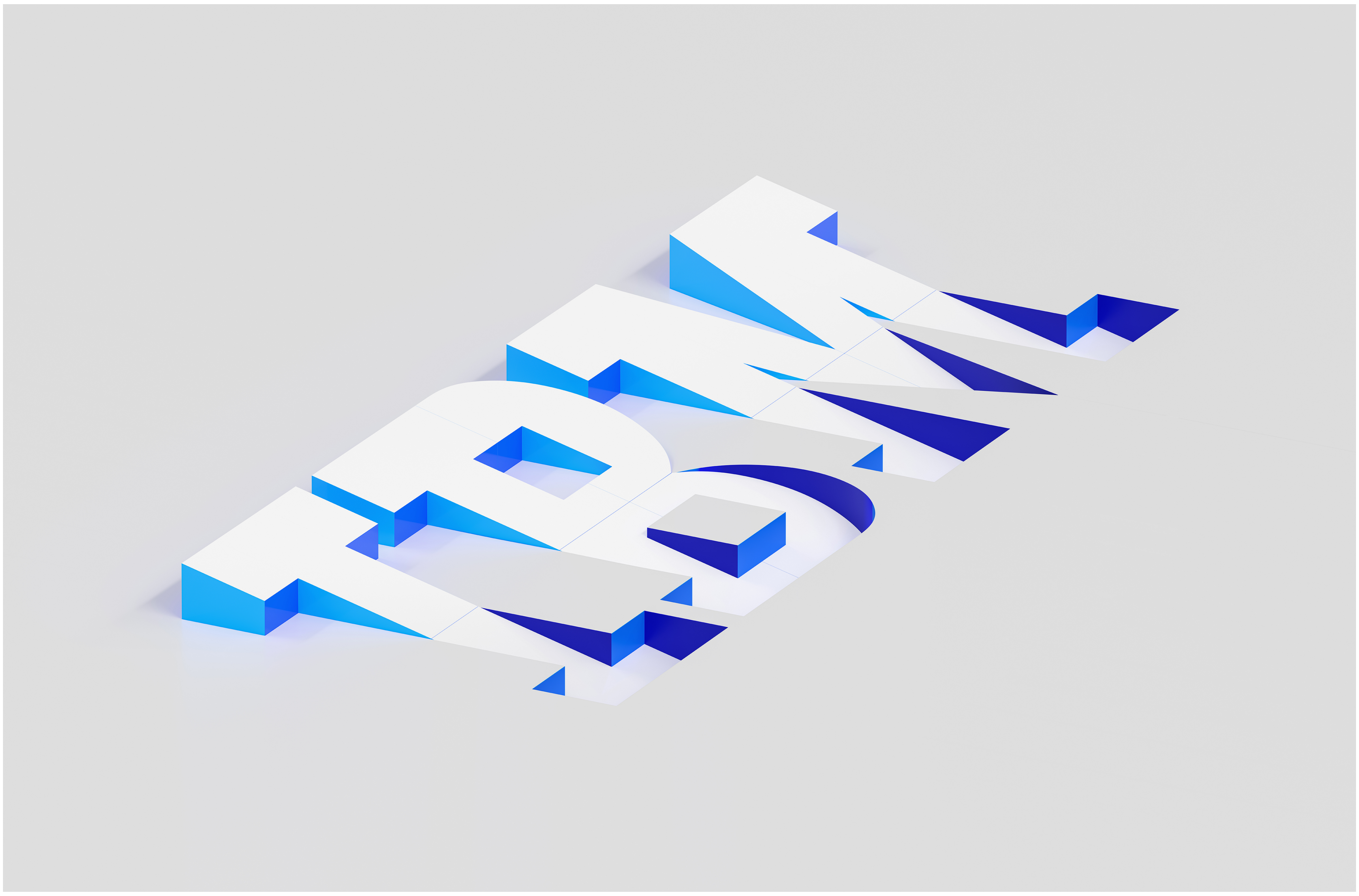
IBM Cloud

Object Storage

Product guide



Edition notices

This PDF was created on 2025-01-07 as a supplement to *Object Storage* in the IBM Cloud docs. It might not be a complete set of information or the latest version. For the latest information, see the IBM Cloud documentation at https://cloud.ibm.com/docs/cloud-object-storage.

Getting started with IBM Cloud Object Storage

IBM Cloud® Object Storage stores encrypted and dispersed data across multiple geographic locations. This getting started tutorial walks through the steps that are needed to use IBM Cloud Object Storage to create buckets, upload objects, and set up access policies to allow other users to work with your data.

Before you begin

You need the following to get started with IBM Cloud Object Storage:

- An IBM Cloud® Platform account
- An instance of IBM Cloud Object Storage
- Some files on your local computer to upload to Object Storage.

This tutorial takes a new user through the first steps with the IBM Cloud Platform console. Developers who want to get started with the API, see the Developer's Guide or API overview.

Create some buckets to store your data

1. Ordering IBM Cloud Object Storage creates a *service instance*. IBM Cloud Object Storage is a multi-tenant system, and all instances of Object Storage share physical infrastructure. You will be automatically redirected to the service instance upon its creation. Your Object Storage instances are listed under **Storage** in the resource list.

   Tip: The terms 'resource instance' and 'service instance' refer to the same concept, and can be used interchangeably.

2. You will need a bucket before you can store data in your new *service instance*. To **Create a bucket**, start by choosing a unique name. All buckets in all regions across the globe share a single namespace. Ensure that you have the correct permissions to create a bucket.

   Tip: When you name buckets or objects, be sure to avoid the use of Personally Identifiable Information (PII). PII is information that can identify any user (natural person) by name, location, or any other means.

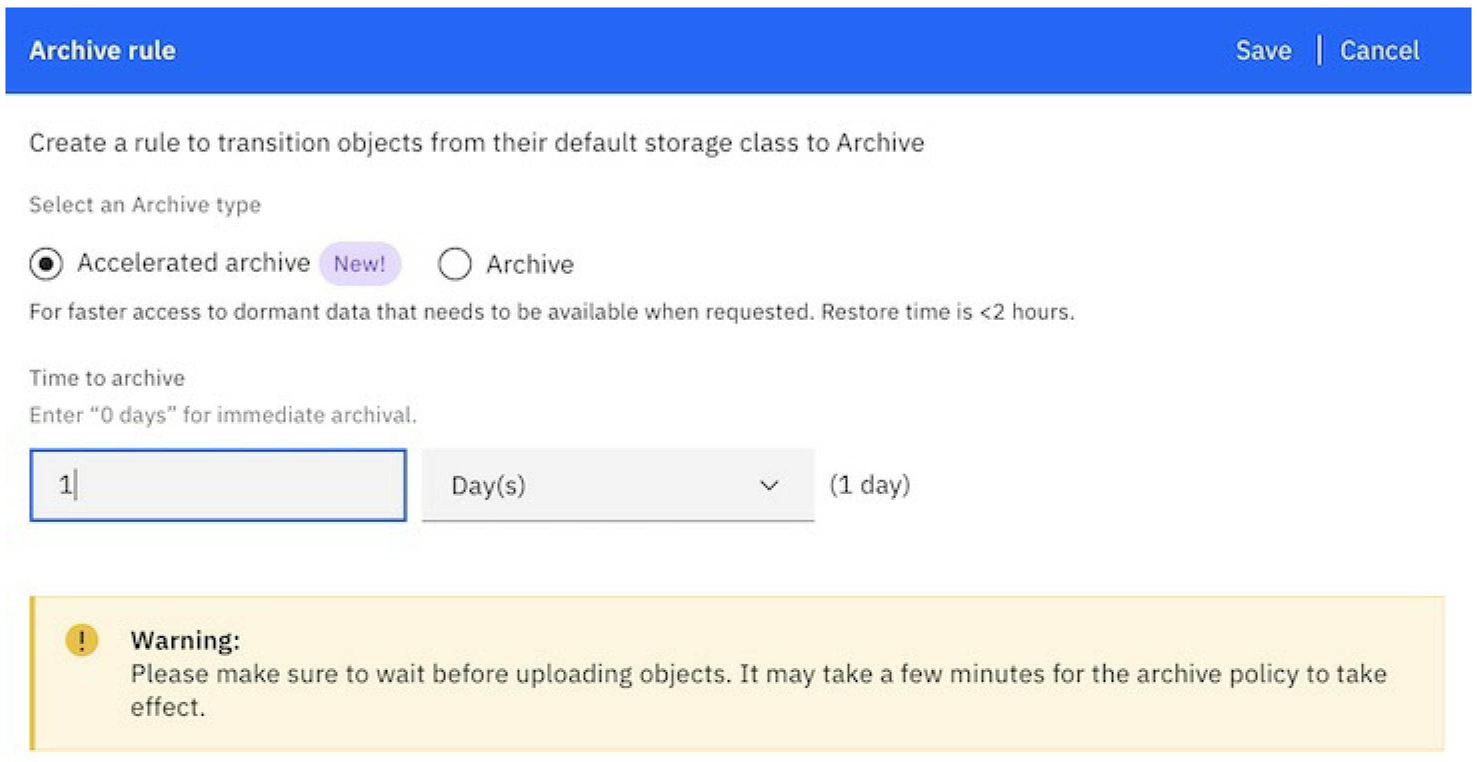
   Tip: Bucket names are required to be DNS addressable and are not case-sensitive.

3. First, choose the level of *resiliency* you want. Then, choose a *location* where you would like your data to be physically stored. Resiliency refers to the scope and scale of the geographic area across which your data is distributed. *Cross Region* resiliency spreads your data across several metropolitan areas, while *Regional* resiliency spreads data across a single metropolitan area. A *Single Data Center* distributes data across devices within a single site only.

4. Choose the bucket's *storage class* to accurately reflect how often you expect to read the stored data. This is important as it determines your billing details. Follow the **Create** link to create and access your new bucket.

5. Determine the advanced configurations, if any, suitable to your content. You can store data by transitioning from any of the storage tiers (Standard, Vault, Cold Vault and Flex) to long-term offline archive or use the online Cold Vault option. See the example in Figure 1 for options in creating an archive policy.

Create an archive policy


Archive rule

Save | Cancel

Create a rule to transition objects from their default storage class to Archive

Select an Archive type

(●) Accelerated archive New! ( ) Archive

For faster access to dormant data that needs to be available when requested. Restore time is <2 hours.

Time to archive

Enter “0 days” for immediate archival.

1 Day(s) (1 day)

Warning:
Please make sure to wait before uploading objects. It may take a few minutes for the archive policy to take effect.

Tip: Buckets are a way to organize your data, but they're not the sole way. Object names (often referred to as *object keys*) can use one or more forward slashes for a directory-like organizational system. You then use the portion of the object name before a delimiter to form an *object prefix*, which is used to list related objects in a single bucket through the Object Storage API.

Add some objects to your buckets

Now go ahead and go to one of your buckets by selecting it from the list. Click **Add Objects**. New objects overwrite existing objects with the same names within the same bucket. When you use the console to upload objects the object name always matches the file name. There doesn't need to be any relationship between the file name and the object key if you're using the API to write data. Go ahead and add a handful of files to this bucket.

Tip: Objects are limited to 200 MB when uploaded through the console unless you use the [Aspera high-speed transfer] plug-in or use Cross-Origin Resource Sharing (CORS), by setting the CORS headers. Larger objects (up to 10 TB) can also be [split into parts and uploaded in parallel using the API]. Object keys can be up to 1024 characters in length, and it's best to avoid any characters that might be problematic in a web address. For example, `?`, `=`, `<`, and other special characters might cause unwanted behavior if not URL-encoded.

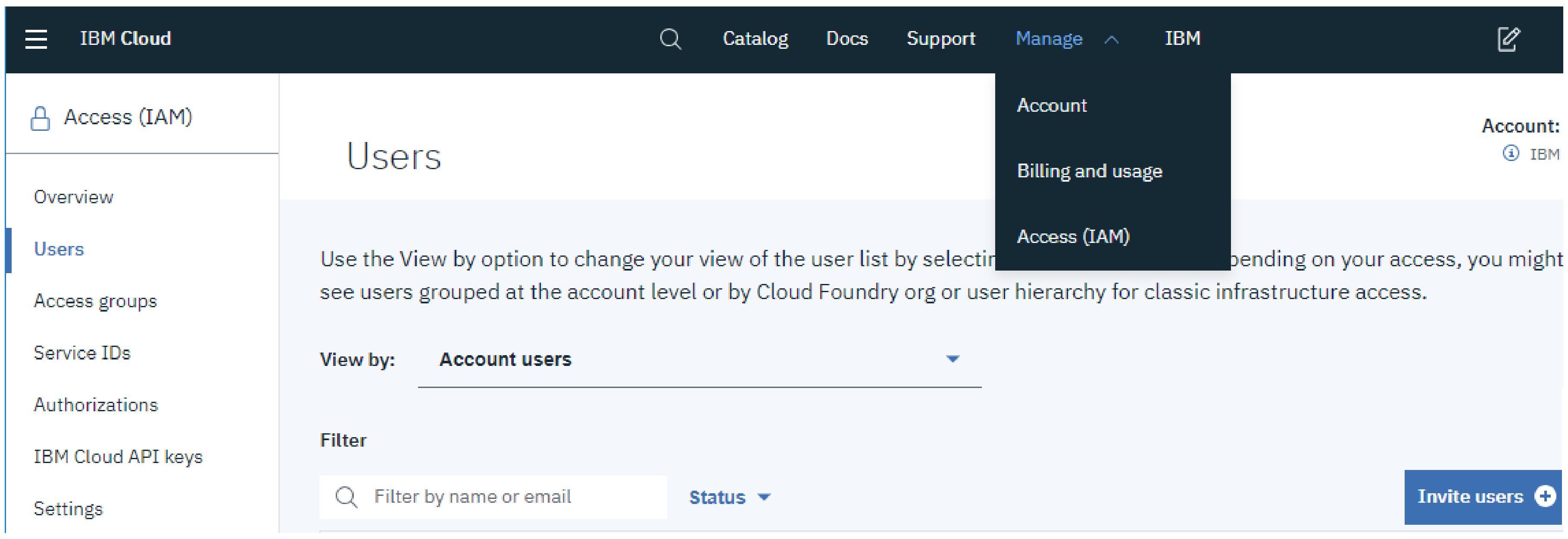
If an object with a special character is uploaded to a bucket, it may cause problems with displaying and accessing it in the UI. In these cases, the object should be deleted and re-uploaded with a more standard name. You may delete these objects with Expiration or Lifecycle rules if the UI and CLI deletions are not successful. Avoid special characters to prevent any difficulties with accessing or deleting the object.

How do I invite a user to administer buckets and data?

Bringing in another user and allow them to act as an administrator for the instance and any data stored in it is an important way to distribute responsibility for administering your IBM Cloud Object Storage instance.

1. To add the new user you first need to leave the current Object Storage interface and head for the IAM console. Go to the **Manage** menu and follow the link at **Access (IAM)** > **Users**. Click **Invite users**.

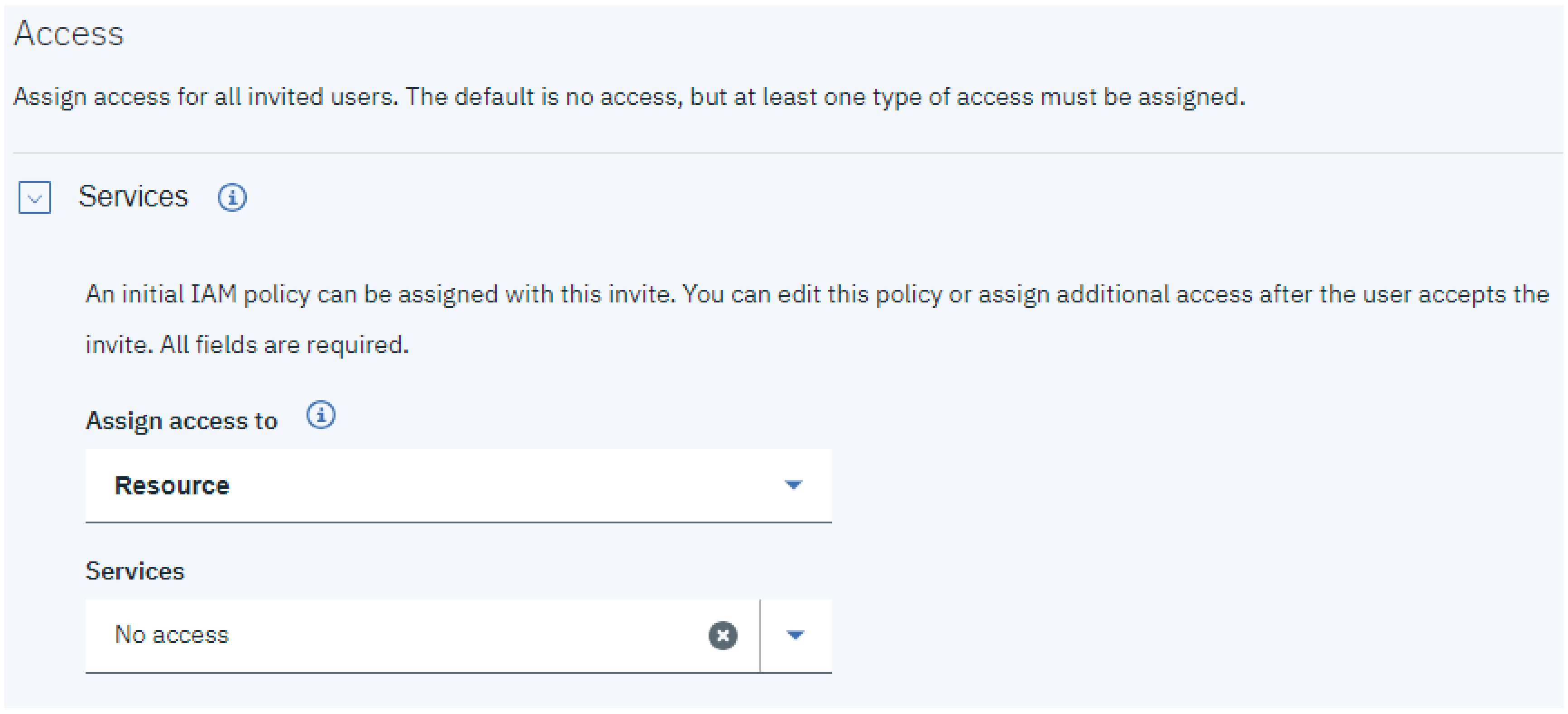
Figure 2: IAM invite users



Use the View by option to change your view of the user list by selectin… …pending on your access, you might see users grouped at the account level or by Cloud Foundry org or user hierarchy for classic infrastructure access.

2. Enter the email address of a user you want to invite to your organization, then expand the **Services** section and select "Resource" from the **Assign access to** menu. Now choose "Cloud Object Storage" from the **Services** menu.

Figure 3: IAM Services



Access

Assign access for all invited users. The default is no access, but at least one type of access must be assigned.

Services

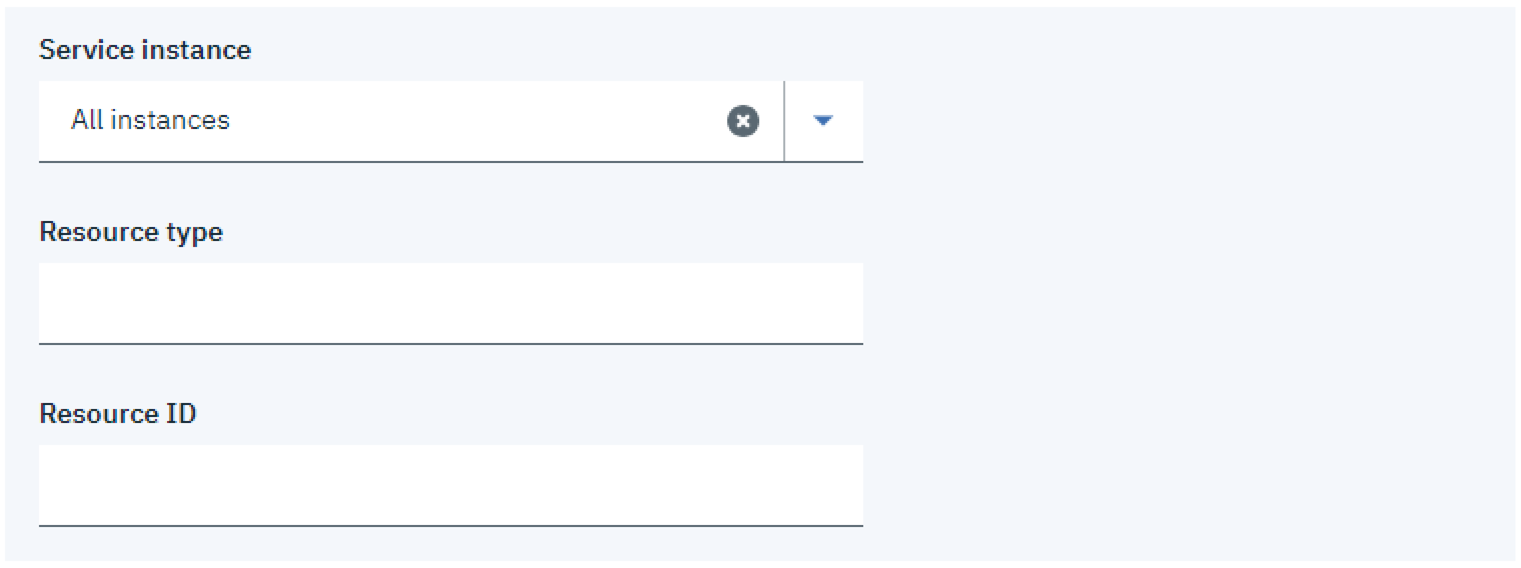
An initial IAM policy can be assigned with this invite. You can edit this policy or assign additional access after the user accepts the invite. All fields are required.

Assign access to: Resource

Services: No access

3. Now, three more fields appear: *Service instance*, *Resource Type*, and *Resource ID*. The first field defines which instance of Object Storage the user can access. It can also be set to grant the same level of access to all instances of Object Storage. We can leave the other fields blank for now.

Figure 4: IAM identifiers for services and resources



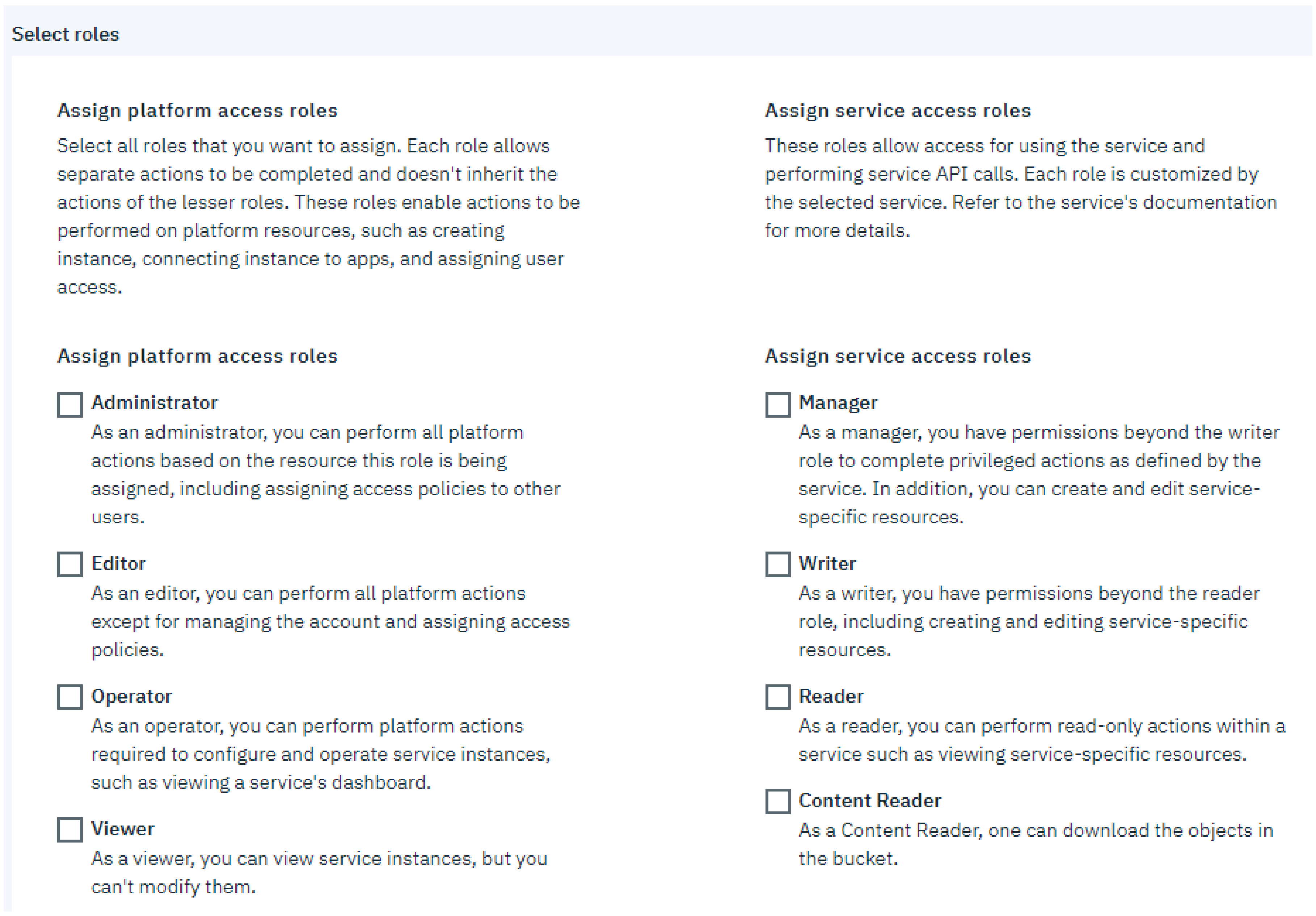
Service instance: All instances

Resource type

Resource ID

4. The check box under **Select roles** determines the set of actions available to the user. Select the "Administrator" platform access role to allow the user grant other users and service IDs access to the instance. Select the "Manager" service access role to allow the user to manage the Object Storage instance as well as create and delete buckets and objects. These combinations of a *Subject* (user), *Role* (Manager), and *Resource* (Object Storage service instance) together form IAM policies. For more detailed guidance on roles and policies, see the IAM documentation.

Figure 5: IAM select roles


Select roles

Assign platform access roles

Select all roles that you want to assign. Each role allows separate actions to be completed and doesn't inherit the actions of the lesser roles. These roles enable actions to be performed on platform resources, such as creating instance, connecting instance to apps, and assigning user access.

Assign platform access roles

- ☐ **Administrator**
  As an administrator, you can perform all platform actions based on the resource this role is being assigned, including assigning access policies to other users.
- ☐ **Editor**
  As an editor, you can perform all platform actions except for managing the account and assigning access policies.
- ☐ **Operator**
  As an operator, you can perform platform actions required to configure and operate service instances, such as viewing a service's dashboard.
- ☐ **Viewer**
  As a viewer, you can view service instances, but you can't modify them.

Assign service access roles

These roles allow access for using the service and performing service API calls. Each role is customized by the selected service. Refer to the service's documentation for more details.

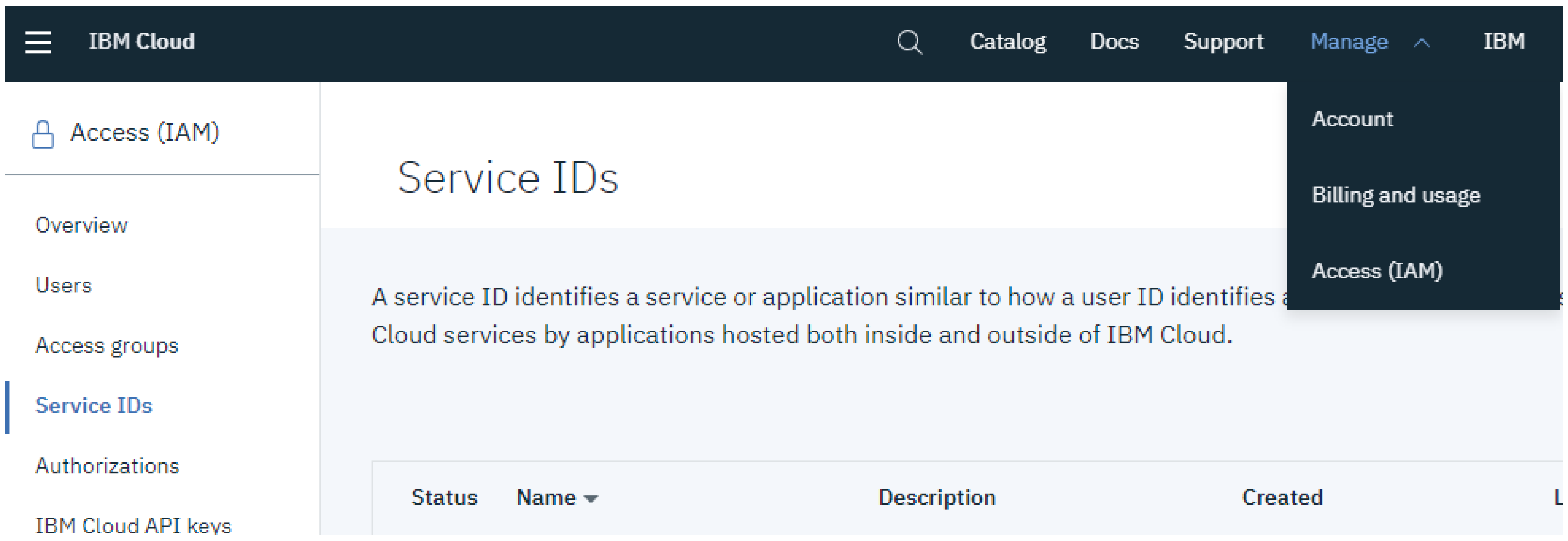
Assign service access roles

- ☐ **Manager**
  As a manager, you have permissions beyond the writer role to complete privileged actions as defined by the service. In addition, you can create and edit service-specific resources.
- ☐ **Writer**
  As a writer, you have permissions beyond the reader role, including creating and editing service-specific resources.
- ☐ **Reader**
  As a reader, you can perform read-only actions within a service such as viewing service-specific resources.
- ☐ **Content Reader**
  As a Content Reader, one can download the objects in the bucket.

Give developers access to a bucket.

1. Navigate to the **Manage** menu and follow the link at **Access(IAM)** > **Service IDs**. Here you can create a *service ID*, which serves as an abstracted identity bound to the account. Service IDs can be assigned API keys and are used in situations where you don't want to tie a particular Developer's identity to a process or component of an application.

Figure 6: IAM Service Ids



Service IDs

A service ID identifies a service or application similar to how a user ID identifies
Cloud services by applications hosted both inside and outside of IBM Cloud.

Status	Name	Description	Created

2. Repeat the above process but in step 3, choose a particular service instance, and enter "bucket" as the *Resource Type* and the full CRN of an existing bucket as the *Resource ID*.

3. Now the service ID can access that particular bucket, and no others.

Next steps

Now that you are familiar with your object storage via the web-based console, you might be interested in doing a similar workflow from the command line. Check out using the `ibmcloud cos` command-line utility to create a service instance and interacting with IAM. And you can further use `curl` for accessing COS directly. Check out the API overview to get started.

What is IBM Cloud Object Storage?

IBM Cloud® Object Storage is a highly available, durable, and secure platform for storing unstructured data. Unstructured data (sometimes called binary or "blob" data) refers to data that is not highly structured in the manner of a database. Object storage is the most efficient way to store PDFs, media files, database backups, disk images, or even large structured datasets.

The files that are uploaded into IBM Cloud Object Storage are called **objects**. Objects can be anywhere from very small (a few bytes) [to very large] (up to 10TB). They are organized into **buckets** that serve as containers for objects, and which can be configured independently from one another in terms of locations, resiliency, billing rates, security, and object lifecycle. Objects themselves have their own metadata in the form of user-defined tags, legal holds, or archive status. Within a bucket, the hierarchy of objects is effectively "flat", although it is possible to add prefixes to object names to provide some organization and to provide flexibility in listing and other operations.

IBM Cloud Object Storage is strongly consistent for all data operations, and eventually consistent for bucket configuration operations. This means that when an object is uploaded, the server responds with a `200 OK` after the object is successfully written, and the object is immediately available for listing and reading. All data stored in IBM Cloud Object Storage is encrypted, erasure-coded, and dispersed across three locations (with the distance between locations ranging from within a single data center, across a Multi-Zone Region or MZR, or even across multiple MZRs). This geographic range of dispersal contributes to a bucket's resiliency.

All requests and responses are made over HTTPS and all requests support the use of hash-based integrity checks using a `Content-MD5` header. If the provided MD5 hash does not match the checksum computed by the storage service, the object is discarded and an error is returned. All `GET` and `HEAD` requests made to objects return an `Etag` value with the MD5 hash of the object to ensure integrity on the client side.

Developers use APIs to interact with their object storage. IBM Cloud Object Storage supports a subset of the S3 API for reading and writing data, as well as for bucket configuration. Additionally, there is a Object Storage Resource Configuration API for reading and configuring bucket metadata. Software development kits (SDKs) are available for the Python, Java, Go, and the Node.js framework. A plug-in is available for the IBM Cloud Command Line Interface.

The IBM Cloud console provides a user interface for most operations and configuration as well.

Cloud Object Storage on IBM Cloud Satellite

Workloads that require object storage on-premise, or in a geographic location not supported by IBM Cloud data centers, can make use of IBM Cloud Satellite. For more information, see get started provides support to provision accounts, to create buckets, to upload objects, and to use a reference of common operations through API interactions.

Getting organized

For administrators

Storage and system administrators familiar with IBM Cloud® Object Storage can easily manage users, create and rotate API keys, and grant roles to users and services.

If you haven't already, go ahead and read through the getting started tutorial to familiarize yourself with the core concepts of buckets, objects, and users.

Setting up your storage

First, you need to have at least one Object Storage resource instance, and some buckets to store data in. How do you want to segment access to your data? Where do you want your data to physically reside? How often will the data is accessed?

Segmenting access

You can segment access at two levels: at the resource instance level and at the bucket level.

Perhaps you want to make sure that only a development team can access particular instances of Object Storage. Or, if you want to ensure that only the application your team is making can edit the data that is stored. You might want your developers with access to the cloud platform to only be able to read data for troubleshooting reasons, but not change anything. These are examples of service level policies.

Now what if the development team, or any individual user, who has viewer access to a storage instance, but should be able to directly edit data in one or more buckets? You can use bucket level policies to elevate the level of access that is granted to users within your account. For instance, a user might not be able to create new buckets, but can create and delete objects within existing buckets.

Managing access

IAM is based on a fundamental concept: A *subject* is granted a *role* on a *resource*.

There are two basic types of subjects: a *user* and a *service ID*.

There is another concept, a *service credential*. A service credential is a collection of important information that is needed to connect to an instance of IBM Cloud® Object Storage. It gives a user an identifier for the instance of IBM Cloud Object Storage (that is, the Resource Instance ID), service and authorization endpoints, and a means of associating the subject with an API key (that is, Service ID). When you create the service credential you have the option of either associating it with an existing service ID, or creating a new service ID.

You might want your development team to be able to use the console to view Object Storage instances and Kubernetes clusters. They would need `Viewer` roles on the Object Storage resources and `Administrator` roles on the Container Service. The `Viewer` role allows for the user to only see that the instance exists, and to view existing credentials, but **not** to view buckets and objects. When the service credentials were created, they were associated with a service ID. This service ID would need to have the `Manager` or `Writer` role on the instance to be able to create and delete buckets and objects.

For more information on IAM roles and permissions, see the IAM overview.

For developers

The powerful features of IBM Cloud® Object Storage are available to a developer directly from the command line.

First, ensure that you have the IBM Cloud® Platform CLI and IBM Developer Tools installed.

Create an instance of IBM Cloud Object Storage

1. First, make sure that you have an API key. Get it from IBM Cloud Identity and Access Management.
2. Log in to IBM Cloud Platform by using the CLI. It's also possible to store the API key in a file or set it as an environment variable.

```
$ ibmcloud login --apikey <value>
```

3. Next, create an instance of IBM Cloud Object Storage specifying the name for the instance and the Standard plan (see Choosing a plan and creating an instance). Now you have a CRN for the instance.

```
$ ibmcloud resource service-instance-create <instance-name> cloud-object-storage <plan> global
```

Tip: When trying to create a new instance, if you encounter the error `No resource group targeted`, it indicates that the default resource group is not available and that a resource group must be explicitly set. A list of available resource groups can be found using `ibmcloud resource groups` and the target can be set with `ibmcloud target -g <resource-group>`.

The Getting Started guide walks through the basic steps of creating buckets and objects, as well as inviting users and creating policies. A list of basic 'curl' commands can be found here.

Learn more about using the IBM Cloud CLI to create applications, manage Kubernetes clusters, and more in the documentation.

Using the API

For managing data stored in Object Storage, you can use S3 API compatible tools like the AWS CLI with HMAC credentials for compatibility. As IAM tokens are relatively easy to work with, `curl` is a good choice for basic testing and interaction with your storage. More information can be found in the `curl` and the API documentation.

Using libraries and SDKs

IBM COS SDKs are available for Python, Java, Go, and Node.js. These libraries are forked and modified versions of the AWS S3 SDKs that support IAM token-based authentication, as well as support for Key Protect.

Building applications on IBM Cloud

IBM Cloud® provides flexibility to developers in choosing the right architectural and deployment options for a given application. Run your code on bare metal, in virtual machines, in containers, or by using a serverless framework.

The Cloud Native Computing Foundation fostered Kubernetes container orchestration framework, which forms the foundation for the IBM Cloud® Kubernetes Service. Developers who want to use Object Storage for persistent storage in their Kubernetes applications can learn more at the following links:

- Choosing a storage solution
- Comparison table for persistent storage options
- Main COS page
- Installing COS
- Creating COS service instance
- Decide on the configuration
- Creating a COS Kubernetes secret
- Kubernetes back up and restore information
- Kubernetes Storage Class reference

Optimizing performance

Network topology

There are many ways to connect to IBM Cloud® Object Storage and the choice of endpoint can have an impact on performance.

Physical distance

When an application makes a request to COS, it needs to cross some amount of physical distance. As this distance increases, the latency of the request will also increase. In order to lessen the latency imposed by physical distance, it is optimal to co-locate compute resources and object storage where possible. If your application is running in the IBM Cloud in the `us-south` region, then in order to optimize performance it would be best to read and write data to a bucket also located in the `us-south` region.

Workloads that require accessing data in far reaching places might benefit from using IBM Aspera, especially if there is significant packet loss. More information about using IBM Aspera High-Speed Transfer and COS can be found in the Aspera guide.

Applications with global reach will benefit from using a Content Delivery Network to cache assets stored in COS in locations closer to their end users. The original files continue to be hosted in their bucket, but copies can be cached in various locations around the world where users are originating requests.

Resilience requirements

Some workloads might require the additional levels of resiliency that comes with writing data to Cross Region buckets, while others might rely on the increased marginal performance found in a Single Data Center bucket. Each application needs to strike a balance between higher availability and faster performance.

When using a Cross Region endpoint, it is possible to direct inbound traffic to a specific access point while still dispersing data across all three regions. When sending requests to an individual access point **there is no automated failover if that region becomes unavailable** . Applications that direct traffic to an access point instead of the `geo` endpoint **must** implement appropriate failover logic internally to achieve the availability advantages of the cross-region storage.

One reason for using an access point is to control where data ingress and egress occurs while still dispersing the data across the widest possible area. Imagine an application running in the `us-south` region that wants to store data in a US cross-region bucket but wants to ensure that all read and write requests remain in the Dallas area:

1. The application creates a client using the `https://s3.private.dal.us.cloud-object-storage.appdomain.cloud` endpoint.
2. The COS service in Dallas suffers an outage.
3. The application detects a persistent failure trying to use the access point.
4. The application recognizes the need to fail over to a different access point, such as San Jose.
5. The application creates a new client using the `https://s3.private.sjc.us.cloud-object-storage.appdomain.cloud` endpoint.
6. Connectivity is resumed, and access can be re-routed to Dallas when service is restored.

For contrast, imagine another application using the normal US cross-region endpoint:

1. The application creates a client using the `https://s3.us.cloud-object-storage.appdomain.cloud` endpoint.
2. The COS service in Dallas suffers an outage.
3. All COS requests are automatically rerouted to San Jose or Washington until service is restored.

Network type

Traffic directed to COS can come from one of three networks: Public, Private, or Direct. The network that is targeted is defined by the COS service endpoint used to access a bucket. While a bucket is created in a single location (be that Cross Region, Regional, or Single Site) it is still possible to access that same bucket via any of the three network types described.

Public traffic traverses the public Internet until it reaches the IBM Cloud and is routed to a load balancer that directs traffic into the COS distributed storage network. Private traffic originates within the IBM Cloud and never touches the public Internet. Direct traffic originates in a Virtual Private Cloud that could contain both local data centers and IBM Cloud resources. This architecture requires IBM Direct Link, and allows users to connect directly to the Private IBM Cloud network from a user's data center (using a reverse proxy) without ever touching the public Internet.

Because the Private network eliminates any variances, congestion, or vulnerabilities found in the Public Internet, it is recommended that all workloads use the Private network whenever possible.

Data IO and encryption

Object size can have significant impacts on IBM Cloud® Object Storage performance. Choose the right approach for your workload.

Multipart transfers

Under typical conditions, multipart uploads and downloads are a very efficient method for breaking up transfers into many parallel transactions. Depending on the object size, a part size of 100MB is generally recommended. In any case, it is most efficient to set the part size to a multiple of 4MiB to optimize the data ingest into and egress out of COS.

As with AWS S3, using multipart transfers provides the following advantages:

- Improved throughput — You can upload parts in parallel to improve throughput.
- Quick recovery from any network issues — Smaller part size minimizes the impact of restarting a failed upload due to a network error.
- Pause and resume object uploads — Upload object parts over time. Once a multipart upload is initiated a multipart upload there is no expiry; it must explicitly complete or the multipart upload has to be aborted.
- Begin an upload before the final object size is known — An object can be uploaded as it is being created.

Due to the additional complexity of multipart transfers, it is recommended to use appropriate S3 libraries, tools, or SDKs that offer support for managed multipart transfers:

- IBM COS SDK for Java
- IBM COS SDK for Python
- IBM COS SDK for Javascript (Node.js)
- IBM COS SDK for Go
- IBM COS Plug-in for IBM Cloud CLI
- S3FS-FUSE

While there is no dedicated API for a multipart download, it is possible to use a `Range` header in a `GET` request to read only a specific part of an object, and many ranged reads can be issued in parallel, just like when uploading parts. After all parts have been downloaded, they can be concatenated and the complete object can be checked for integrity. As mentioned previously, use of SDKs or other tooling is recommended to avoid the complexities of manually managing these transfers.

Workflows that need to store large numbers of very small objects may be better served by aggregating the small files into a larger data structure, such as [Parquet].

For objects greater than 200mb in size, especially in less stable networks or over very long distances where packet loss is a concern, Aspera High-Speed Transfer can deliver excellent performance. Aspera transfers can also upload nested directory structures efficiently within a single request.

Throttling batch deletes

The S3 API provides a mechanism for deleting up to 1,000 objects with a single batch delete request. It is recommended to throttle these requests client-side to minimize the chances of derogatory performance within the COS System. When the number of deletes issued is too high for the system, the client will receive HTTP 503 errors with an error message indicating "slow down".

Consistency impacts

IBM Cloud Object Storage System guarantees immediate consistency for all object operations, which includes object writes, overwrites, deletes, multipart operations, and ACL modifications. Bucket creation is also immediately consistent. Bucket metadata and configuration is eventually consistent, as is the case with other object storage systems, meaning that changes across a highly distributed system may not be synchronized for a short period of time. This occurs because of metadata caching that provides significant performance benefits, and also safeguards against the possibility of denial-of-service attacks.

Some applications will overwrite the same object, or delete and rewrite the same object repeatedly over a short amount of time. This can cause contention in the indices within the COS System and should be avoided. In the rare case where overwriting data with the same object key (name) at a very high frequency and over extended periods of time is a critical aspect of an application design, a different storage platform (file, block, noSQL, etc) may be a better choice.

Existence checks

Applications may want to check if an object exists or has been modified before writing to it. Often this leads to inefficient application logic that will send a HEAD request followed by a PUT or GET request. This anti-pattern results in wasted networking and server resources, and should be discouraged.

Instead of using a HEAD request as an existence check within some function, use a conditional request header. These standard HTTP headers will compare MD5 hashes or timestamps to determine whether the data operation should proceed or not. For more information, see Conditional Requests.

Using conditional requests

When making a request to read or write data, it is possible to set conditions on that request to avoid unnecessary operations. This is accomplished using the following pre-conditional HTTP headers: `If-Match`, `If-None-Match`, `If-Modified-Since`, and `If-Unmodified-Since`.

It is generally preferable to use `If-Match` because the granularity of the `Last-Modified` value is only in seconds, and may not be sufficient to avoid race conditions in some applications.

Using If-Match

On an object `PUT`, `HEAD`, or `GET` request, the `If-Match` header will check to see if a provided `Etag` (MD5 hash of the object content) matches the provided `Etag` value. If this value matches, the operation will proceed. If the match fails, the system will return a `412 Precondition Failed` error.

> If-Match is most often used with state-changing methods (for example, POST, PUT, DELETE) to prevent accidental overwrites when multiple user agents might be acting in parallel on the same resource (that is, to prevent the "lost update" problem).

Using If-None-Match

On an object `PUT`, `HEAD`, or `GET` request, the `If-None-Match` header will check to see if a provided `Etag` (MD5 hash of the object content) matches the provided `Etag` value. If this value does not match, the operation will proceed. If the match succeeds, the system will return a `412 Precondition Failed` error on a `PUT` and a `304 Not Modified` on `GET` or `HEAD`.

> If-None-Match is primarily used in conditional GET requests to enable efficient updates of cached information with a minimum amount of transaction overhead. When a client desires to update one or more stored responses that have entity-tags, the client SHOULD generate an If-None-Match header field containing a list of those entity-tags when making a GET request; this allows recipient servers to send a 304 (Not Modified) response to indicate when one of those stored responses matches the selected representation.

Using If-Modified-Since

On an object `HEAD` or `GET` request, the `If-Modified-Since` header will check to see if the object's `Last-Modified` value (for example `Sat, 14 March 2020 19:43:31 GMT`) is newer than a provided value. If the object has been modified, the operation will proceed. If the object has not been modified, the system will return a `304 Not Modified`.

> If-Modified-Since is typically used for two distinct purposes: to allow efficient updates of a cached representation that does not have an `Etag` and to limit the scope of a web traversal to resources that have recently changed.

Using If-Unmodified-Since

On an object `PUT`, `HEAD`, or `GET` request, the `If-Unmodified-Since` header will check to see if the object's `Last-Modified` value (for example `Sat, 14 March 2020 19:43:31 GMT`) is equal to or earlier than a provided value. If the object has not been modified, the operation will proceed. If the `Last-Modified` value is more recent, the system will return a `412 Precondition Failed` error on a `PUT` and a `304 Not Modified` on `GET` or `HEAD`.

> If-Unmodified-Since is most often used with state-changing methods (for example, POST, PUT, DELETE) to prevent accidental overwrites when multiple user agents might be acting in parallel on a resource that does not supply entity-tags with its representations (that is, to prevent the "lost update" problem). It can also be used with safe methods to abort a request if the selected representation does not match one already stored (or partially stored) from a prior request.

Retry strategy

While most libraries and SDKs will automatically handle retry logic, care must be taken when writing software that uses the API directly to properly handle transient errors. Most importantly, it is critical to provide appropriate retry logic that implements exponential back-off when receiving 503 errors.

Cypher tuning

IBM COS supports a variety of Cipher settings to encrypt data in transit. Not all cipher settings yield the same level performance and using TLS in general leads to small performance degradation. The following cipher settings are recommended (in descending order of priority):

- `TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384`
- `TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256`
- `TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA`
- `TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA`
- `TLS_RSA_WITH_AES_256_CBC_SHA256`
- `TLS_RSA_WITH_AES_128_CBC_SHA256`
- `TLS_RSA_WITH_AES_256_CBC_SHA`
- `TLS_RSA_WITH_AES_128_CBC_SHA`

Client-side bottlenecks

Often, poor performance is investigated and there is no indication of any lag or bottlenecks on the server side. These issues are resolved by making improvements to other aspects of the application architecture.

Application design

Reading and writing data to an object store can use significant resources in order to facilitate integrity checks, especially for applications that are transferring large objects. It is important to ensure that the application design takes into account any potential bottlenecks on CPU, disk IO, memory, and network use in order to minimize any performance impact.

Compute resource power

Containers, virtual machines, or bare metal must have sufficient power for the application to run efficiently. No amount of careful software design and engineering can overcome an underpowered computing environment, and optimal performance is dependent on adequate resources.

Network

NIC speeds can determine the limits of data throughput. It is possible to configure a virtual 10GB NIC, but there are limitations on how many transactions and threads a virtualized NIC can effectively perform and manage. Due to the countless potential configurations and types of virtualized NICs, it isn't possible to provide precise guidance. It is recommended to experiment with a rational amount of transactions to COS on a single virtualized NIC, perhaps 25-50 at first. Then test again, moving up or down in the number to try and find the proper balance and to provide the best performance on a consistent basis.

If encountering network errors, it is useful to provide the specific endpoint where requests are being sent when opening a support ticket. This allows the support team to efficiently investigate the networking to see if there are errors on IBM's network hops.

Your responsibilities when using IBM Cloud Object Storage

Learn about the management responsibilities and terms and conditions that you have when you use IBM Cloud® Object Storage. For overall terms of use, see Cloud Services terms.

Operational responsibilities

IBM responsibilities

- Maintaining SLAs as stated in IBM Cloud Object Storage Service Description.
- Ensure client gets appropriate notifications regarding availability and downtime of its resources.
- Monitor service performance related metrics such as network, storage, and compute capacities.
- Monitor and manage service health and availability.

Your responsibilities

- Ensuring data back-ups if necessary.
- Mitigating potential accidental deletion of data. If applicable the client should store archive data in buckets configured with a data retention policy to help address data loss due to accidental deletion. IBM is unable to “un-delete” data.
- Monitor and manage non-IBM network resources to ensure appropriate access to IBM Cloud service endpoints including capacity and availability.
- Provisioning Object Storage buckets with the appropriate resiliency option, storage class, data locality, and optional configurations necessary for the specific workload and use case.

Managing infrastructure and the cloud environment

IBM responsibilities

- Deployment, provisioning, managing IBM Cloud Object Storage resources offered to clients.
- Ensuring availability of adequate resources to allow for scaling of client resources for client usage requirements.
- All physical and environmental controls.

Your responsibilities

- Ensure proper network access and capacity is available and can reach the designated Object Storage service endpoints. (private or public).
- Ensure applications can interact with the COS S3 API either via native support or with the addition of a hardware or software gateway solution.

Security

IBM responsibilities

- Security and monitoring of the environment of IBM data center security guidelines.
- Encrypting client data when at-rest and in-transit between different IBM Cloud data center locations as part of resiliency options offered to clients.
- Continuous monitoring of resources to check for vulnerabilities and security breaches.

Your responsibilities

- Client controls access to IBM Cloud Object Storage provisioned resources and is responsible for ensuring access to client data is only provided to appropriate resources within the client organization.
- After retrieving data from IBM hosted Object Storage, client takes full responsibility to ensure the security and privacy of any local copies maintained.
- If applicable, manage customer provided encryption keys via S3 API or through IBM hosted key management services.
- Scanning of user data for viruses, malware, etc prior to uploading objects to the IBM Storage platform.

Compliance

IBM responsibilities

- IBM Cloud Object Storage to maintain controls commensurate with the compliance certifications/attestations as stated in official data sheets.

Your responsibilities

- Client responsibility to ensure and seek appropriate legal guidance in order to validate its compliance with pertinent industry compliance certifications and regulations.

- Client to ensure its use of Object Storage resources are inline with the terms and conditions set forth in the IBM Cloud Service Description and any other associated transaction documents.

Notifications

Notifications about IBM Cloud® Object Storage

Notifications about changes that affect Object Storage.

Notifications topics

Notifications about changes.

- IBM Cloud® Object Storage deployment of GSLB in all MultiZone Regions

IBM Cloud® Object Storage deployment of GSLB in all MultiZone Regions

The IBM Cloud® Object Storage team is enabling Global Server Load Balancing (GSLB) in IBM's MZR offerings over the next several months. This change causes the regional endpoints (public, private, and direct) to use new virtual IP addresses. This change affects you when accessing Object Storage with an IP address (rather than URL), or if you have IP-based allowlists or firewall-rules running in your environment.

What you need to know about this change

During the scheduled maintenance, DNS resolution for the Object Storage endpoints listed in the table changes from returning a single IP address to returning one of three possible IP addresses for each region. Previously all connection requests would use a single IP address for each region. With GSLB, your DNS lookup returns the zonal IP address of the Object Storage front end to optimally serve your regional traffic. The IP address can change to a different IP address on subsequent DNS lookups. Approximately 14 days before each maintenance window, the new IP addresses are available for customers to verify connectivity. On the DNS update day, customers may experience temporary name resolution issues, connection failures, and increased latency. These impacts should be short lived.

How you benefit from this change

This change allows your traffic to be routed to the Object Storage front-end servers best positioned to handle traffic at any point in time. It also provides increased reliability for IBM's MZR endpoints.

Understanding if you are impacted by this change

Customers that use hardcoded IP addresses for IBM Object Storage endpoints within their workloads, firewalls, or security components may be affected when new IP addresses are used.

Action may be required. This change affects you when accessing Object Storage:

- By means of an IP address (rather than URL)
- If you have IP-based allowlists or firewall-rules running in your environment
- If you have IP address specific routing

What actions you need to take

Review the IP address information and schedule tables to understand when the changes are made, and what new IP addresses are used. If you are affected by this change, test connectivity to the provided IP addresses or subnets. The new IPs are available 14 days before the DNS maintenance cutover date.

Private and direct networks (for future changes)

In order to avoid future changes when using the private and direct endpoints, IBM recommends updating firewalls and allow lists to include these networks.

- private network: 10.1.129.0/24
- direct network: 161.26.0.0/16

Deployment dates

Region	New addresses available for testing	DNS updated to new IP addresses
Sydney (au-syd)	August 15, 2024	August 29, 2024

San Paulo (br-sao)	October 10, 2024	October 24, 2024
Toronto (ca-tor)	October 14, 2024	October 24, 2024 (Public/Direct)
Frankfurt (eu-de)	October 28, 2024	November 11, 2024
Tokyo (jp-tok)	October 31, 2024	November 14, 2024
London (eu-gb)	November 7, 2024	November 21, 2024
Madrid (eu-es)	November 21, 2024	December 5, 2024
Osaka (jp-osa)	October 17, 2024	December 9, 2024
Toronto (ca-tor)	October 14, 2024	December 9, 2024 (Private)
Washington (us-east)	October 21, 2024	January 13, 2025
Dallas (us-south)	October 24, 2024	January 16, 2025

IP address changes

Region	URL	Current IP	New IPs
Sydney (au-syd) public	s3.au-syd.cloud-object-storage.appdomain.cloud	130.198.118.97	130.198.118.97, 130.198.118.105, 130.198.118.106
Sydney (au-syd) private	s3.private.au-syd.cloud-object-storage.appdomain.cloud	10.1.129.67	10.1.129.67, 10.1.129.189, 10.1.129.190
Sydney (au-syd) direct	s3.direct.au-syd.cloud-object-storage.appdomain.cloud	161.26.0.27	161.26.0.27, 161.26.125.27, 161.26.165.27
San Paulo (br-sao) public	s3.br-sao.cloud-object-storage.appdomain.cloud	13.116.118.49	13.116.118.49, 13.116.118.54, 13.116.118.55
San Paulo (br-sao) private	s3.private.br-sao.cloud-object-storage.appdomain.cloud	10.1.129.165	10.1.129.165, 10.1.129.191, 10.1.129.192
San Paulo (br-sao) direct	s3.direct.br-sao.cloud-object-storage.appdomain.cloud	161.26.0.96	161.26.0.96, 161.26.205.96, 161.26.209.96
Toronto (ca-tor) public	s3.ca-tor.cloud-object-storage.appdomain.cloud	163.66.118.49	163.66.118.49, 163.66.118.51, 163.66.118.52
Toronto (ca-tor) private	s3.private.ca-tor.cloud-object-storage.appdomain.cloud	10.1.129.158	10.1.129.158, 10.1.129.193, 10.1.129.194
Toronto (ca-tor) direct	s3.direct.ca-tor.cloud-object-storage.appdomain.cloud	161.26.0.95	161.26.0.95, 161.26.197.95, 161.26.201.95
Osaka (jp-osa) public	s3.jp-osa.cloud-object-storage.appdomain.cloud	163.68.118.49	163.68.118.49, 163.68.118.57, 163.68.118.58
Osaka (jp-osa) private	s3.private.jp-osa.cloud-object-storage.appdomain.cloud	10.1.129.107	10.1.129.107, 10.1.129.195, 10.1.129.196
Osaka (jp-osa) direct	s3.direct.jp-osa.cloud-object-storage.appdomain.cloud	161.26.0.47	161.26.0.47, 161.26.189.47, 161.26.193.47

Washington (us-east) public	s3.us-east.cloud-object-storage.appdomain.cloud	169.63.118.98	169.63.118.98, 169.63.118.96, 169.63.118.100
Washington (us-east) private	s3.private.us-east.cloud-object-storage.appdomain.cloud	10.1.129.94	10.1.129.94, 10.1.129.50, 10.1.129.197
Washington (us-east) direct	s3.direct.us-east.cloud-object-storage.appdomain.cloud	161.26.0.31	161.26.0.31, 161.26.0.18, 161.26.0.85
Dallas (us-south) public	s3.us-south.cloud-object-storage.appdomain.cloud	169.46.118.100	169.46.118.100, 169.46.118.97, 169.46.118.98
Dallas (us-south) private	s3.private.us-south.cloud-object-storage.appdomain.cloud	10.1.129.97	10.1.129.97, 10.1.129.83, 10.1.129.45
Dallas (us-south) direct	s3.direct.us-south.cloud-object-storage.appdomain.cloud	161.26.0.34	161.26.0.34, 161.26.0.29, 161.26.0.16
Frankfurt (eu-de) public	s3.eu-de.cloud-object-storage.appdomain.cloud	158.177.118.97	158.177.118.97, 158.177.118.101, 158.177.118.102
Frankfurt (eu-de) private	s3.private.eu-de.cloud-object-storage.appdomain.cloud	10.1.129.58	10.1.129.58, 10.1.129.198, 10.1.129.199
Frankfurt (eu-de) direct	s3.direct.eu-de.cloud-object-storage.appdomain.cloud	161.26.0.24	161.26.0.24, 161.26.145.24, 161.26.149.24
Tokyo (jp-tok) public	s3.jp-tok.cloud-object-storage.appdomain.cloud	162.133.118.49	162.133.118.49, 162.133.118.55, 162.133.118.56
Tokyo (jp-tok) private	s3.private.jp-tok.cloud-object-storage.appdomain.cloud	10.1.129.66	10.1.129.66, 10.1.129.200, 10.1.129.201
Tokyo (jp-tok) direct	s3.direct.jp-tok.cloud-object-storage.appdomain.cloud	161.26.0.22	161.26.0.22, 161.26.153.22, 161.26.157.22
London (eu-gb) public	s3.eu-gb.cloud-object-storage.appdomain.cloud	169.50.118.97	169.50.118.97, 169.50.118.100, 169.50.118.101
London (eu-gb) private	s3.private.eu-gb.cloud-object-storage.appdomain.cloud	10.1.129.53	10.1.129.53, 10.1.129.202, 10.1.129.203
London (eu-gb) direct	s3.direct.eu-gb.cloud-object-storage.appdomain.cloud	161.26.0.26	161.26.0.26, 161.26.129.26, 161.26.137.26
Madrid (eu-es) public	s3.eu-es.cloud-object-storage.appdomain.cloud	13.120.118.49	13.120.118.49, 13.120.118.51, 13.120.118.52
Madrid (eu-es) private	s3.private.eu-es.cloud-object-storage.appdomain.cloud	10.1.129.187	10.1.129.187, 10.1.129.204, 10.1.129.205
Madrid (eu-es) direct	s3.direct.eu-es.cloud-object-storage.appdomain.cloud	161.26.0.99	161.26.0.99, 161.26.217.99, 161.26.221.99

IP address changes

Release notes for Object Storage

News on the latest releases from IBM Cloud® Object Storage provide the updates you need on all things related to IBM Cloud Object Storage.

12 December 2024

IBM Cloud® Object Storage for IBM Cloud Satellite® is deprecated

Object Storage for Satellite is a managed object storage service that can be deployed on IBM Satellite for clients to store data closer to their applications and data sources, whether on-premises, at the edge, or in a multi-cloud environment. This service is being deprecated due to changes in market expectations, client fit, and lack of adoption. After December 16, 2025, this service is no longer supported. For more information, see Deprecation overview.

20 August 2024

Object Lock available in the EU and AP Cross Regions and all Single Sites

It is now possible to lock objects to ensure individual object versions are stored in a WORM (Write-Once-Read-Many), non-erasable and non-rewritable manner in the EU and AP Cross Region sites along with all Single Sites.

01 July 2024

Free tier update

IBM Cloud Object Storage currently offers a free evaluation to new clients using the Lite Plan. Effective July 1st, 2024, IBM Cloud will replace the Lite Plan with a new Free Tier available within the Standard (paid) plan. See Cloud Object Storage Lite Plan will be replaced by Free Tier announcement.

14 June 2024

New feature!

You can now do full integration with IBM Cloud Metrics Routing and IBM Cloud Activity Tracker Event Routing for IBM Cloud® Object Storage buckets.

04 June 2024

Aspera available in the Chennai Single Data Center

It is now possible to use Aspera high-speed transfer.

19 April 2024

Cloud Functions update

IBM Cloud® Functions is deprecated. Existing Functions entities such as actions, triggers, or sequences will continue to run, but as of 28 December 2023, you can't create new Functions entities. Existing Functions entities are supported until October 2024. Any Functions entities that still exist on that date will be deleted. For more information, see Deprecation overview.

05 March 2024

Support available for Code Engine in Madrid

Code Engine is supported in the `eu-es` region for Madrid regional using IBM Cloud® Object Storage.

29 January 2024

New feature!

You can configure a bucket into a state of Protection Management using the Resource Configuration API if you have approval from IBM Cloud support and Offering Management.

30 November 2023

New feature!

You can create IAM policies that control access to individual objects within your bucket using fine-grained access control.

30 October 2023

Activity Tracker and Monitoring update

Buckets can be created in the `eu-es` region for Madrid regional using IBM Cloud Activity Tracker and IBM Cloud® Monitoring.

16 October 2023

Archive available in the Chennai Single Site

It is now possible to archive data.

3 October 2023

Object Lock available in the Chennai Single Site

It is now possible to lock objects to ensure individual object versions are stored in a WORM (Write-Once-Read-Many), non-erasable and non-rewritable manner in the CHE01 single site.

Encryption update

Buckets can be created in the `eu-es` region using Hyper Protect Crypto Services managed encryption.

22 September 2023

Encryption update

Buckets created using Key Protect managed encryption can now use Key Protect in Madrid regional. Check out KP Regions and Endpoints.

14 June 2023

New location!

Buckets can now be created in a Regional (MZR) configuration in Madrid, Spain. More information can be found in the Select Regions and Endpoints.

16 March 2023

New feature!

It is now possible to lock objects to ensure individual object versions are stored in a WORM (Write-Once-Read-Many), non-erasable and non-rewritable manner.

26 September 2022

New feature!

It is now possible to create instances using a One Rate plan to lower costs and simplify billing for workloads with high levels of egress.

21 June 2022

New feature!

It is now possible to configure buckets for automated replication of objects to a destination bucket.

7 June 2022

Encryption update

Buckets created with Hyper Protect Crypto Services managed encryption can now use Immutable Object Storage to create retention policies that prevent object deletion or modification.

23 May 2022

Encryption update

Buckets created using Hyper Protect Crypto Services managed encryption can now use .

2 May 2022

New feature!

It is now possible so manage access using context-based restrictions. This offers significant improvements over the existing bucket firewall, and allows for the allowlisting of VPCs and other cloud services, in addition to IP address ranges.

5 April 2022

Encryption update

Buckets can be created in the `eu-gb` region using Hyper Protect Crypto Services managed encryption.

9 March 2022

Encryption update

New buckets created in `eu` Cross Region configuration can now use Key Protect managed encryption.

20 January 2022

IBM Cloud Satellite

You can now use your own compute infrastructure to create a Satellite location. Then, you use the capabilities of Satellite to run IBM Cloud services on your infrastructure, and consistently deploy, manage, and control your software workloads. For details, see Hyper Protect Crypto Services managed encryption.

11 November 2021

Versioning update

Object expiration is now permitted in buckets with versioning enabled.

Lifecycle update

Bucket lifecycle rules can now be created to automatically remove incomplete multipart uploads.

24 September 2021

Encryption update

Buckets created using Key Protect managed encryption can now also use Immutable Object Storage to create retention policies that prevent object deletion or modification.

30 August 2021

New location!

Buckets can now be created in a Regional configuration in São Paulo, Brazil. More information can be found in the Select Regions and Endpoints.

12 August 2021

Encryption update

New buckets created in `us` or `ap` Cross Region configuration can now use Key Protect managed encryption.

7 July 2021

New location!

Buckets can now be created in a Regional configuration in Toronto, Canada. More information can be found in the Select Regions and Endpoints.

Metrics update

In addition to usage metrics, IBM Cloud Monitoring can now track request metrics for buckets.

31 March 2021

New feature!

Buckets can now be configured to version objects, allowing for non-destructive overwrites and deletes.

Compliance update

Rules set in the Security and Compliance Center can now be enforced using the `disallow` action.

New feature!

Buckets can now be configured to have a hard quota to control costs by limiting the maximum amount of storage available for that bucket.

15 March 2021

Activity tracking update

Updates to a bucket's metadata using the Resource Configuration API (such as adding or modifying a firewall) will now show the details of the change in the `requestData` fields shown in Activity Tracker.

15 December 2020

New feature!

Objects can now be efficiently tagged with custom key-value pairs.

5 November 2020

New location!

Buckets can now be created in a Regional configuration in Osaka, Japan. More information can be found in the Select Regions and Endpoints.

27 October 2020

New feature!

Buckets can now be configured to serve static websites.

12 October 2020

Encryption updates

Lifecycle actions on Key Protect and Hyper Protect Crypto Services encryption keys can now generate bucket events in Activity Tracker.

Buckets can now be encrypted using Hyper Protect Crypto Services in the US East region.

18 August 2020

New feature!

Data can now be archived using an Accelerated class that allows restoration of archived objects in under two hours for an additional cost.

30 April 2020

IAM updates

Users and Service IDs can now be granted a new `ObjectWriter` role that allows access to writing objects, but without permissions to download objects or to list the contents of a bucket.

Public Access can now be granted to a new `ObjectReader` role that allows anonymous access to reading objects, but without permissions to list the contents of a bucket.

10 February 2020

New feature!

Buckets can now be created in a new Smart Tier storage class that optimizes costs based on usage patterns.

6 December 2019

New location!

Buckets can now be created in a Single Data Center configuration in Singapore. More information can be found in the Select Regions and Endpoints.

15 November 2019

Lifecycle update

Data can now be archived in buckets located in São Paulo, Brazil (`sao1`).

24 October 2019

Compliance update

Immutable Object Storage is now available for buckets in US Cross Region (`us`).

13 October 2019

New location!

Buckets can now be created in a Single Data Center configuration in Paris, France. More information can be found in the Select Regions and Endpoints.

11 September 2019

New feature!

Changes made to object storage data can be used as an event source for Cloud Functions .

28 August 2019

New feature!

Data can be encrypted using HPCS.

7 August 2019

Activity tracking update

Object-level events can be tracked using Activity Tracker.

11 June 2019

New feature!

Objects can be automatically deleted by adding expiration rules to a bucket's lifecycle configuration .

15 May 2019

New CLI plug-in!

Users can access and interact with object storage using the IBM Cloud CLI .

26 April 2019

New location!

Buckets can now be created in a Single Data Center configuration in Hong Kong. More information can be found in the Select Regions and Endpoints.

28 March 2019

New feature!

User can use COS Firewall to restrict access to the data in COS only if request originates from a list of allowed IP addresses.

IAM update

IAM policies can now grant public access to entire buckets.

New location!

Buckets can now be created in a Single Data Center configuration in Milan, Italy. More information can be found in the Select Regions and Endpoints.

28 February 2019

New location!

Buckets can now be created in a Single Data Center configuration in San Jose, USA. More information can be found in the Select Regions and Endpoints.

18 January 2019

New location!

Buckets can now be created in AP Australia region. More information can be found in the Select Regions and Endpoints.

14 December 2018

New feature!

Users can use Immutable Object Storage to create retention policies that prevent object deletion or modification.

New location!

Buckets can now be created in a Single Data Center configuration in Mexico City, Mexico. More information can be found in the Select Regions and Endpoints.

12 November 2018

New location!

Buckets can now be created in a Single Data Center configuration in Montréal, Canada. More information can be found in the Select Regions and Endpoints.

12 October 2018

New location!

Buckets can now be created in a Single Data Center configuration in Seoul, South Korea. More information can be found in the Select Regions and Endpoints.

20 September 2018

New feature!

Users can archive cold data by setting the proper parameters in a bucket lifecycle configuration policy, either using the console, REST API, or a language-specific SDK.

18 August 2018

New locations!

Buckets can now be created in a Single Data Center configuration in Sao Paolo, Brazil and Oslo, Norway. More information can be found in the Select Regions and Endpoints.

22 June 2018

New locations!

Buckets can now be created in the EU Germany region. Data stored in these buckets is distributed across three availability zones in the EU Germany region. More information can be found in the Select Regions and Endpoints documentation.

Buckets can now be created in a Single Data Center configuration in Chennai, India and Amsterdam, Netherlands. This allows for lower latency when accessing storage from compute resources co-located within the same data center, or for data requiring a specific geographic location. More information can be found in Select Regions and Endpoints.

16 March 2018

New location!

Buckets can now be created in an AP Cross Region configuration. Data stored in these buckets is distributed across the Seoul, Tokyo, and Hong Kong data centers. More information can be found in the Select Regions and Endpoints.

New feature!

Users can run `SELECT` SQL queries directly against structured data objects using IBM Cloud® Data Engine. More information can be found in the Data Engine documentation.

7 March 2018

New feature!

Users who upload or download files using the web-based console have the option to use Aspera high-speed transfer for these operations via a browser plug-in. This allows for transfers of objects larger than 200MB, and also allows for greater control and visibility of uploads and downloads. Additional information can be found in the Uploading Data documentation. Downloads using Aspera high-speed incur additional egress charges. For more information, see the pricing page.

11 February 2018

New location!

Buckets can now be created in a Single Data Center configuration in Toronto, Canada and Melbourne, Australia. This allows for lower latency when accessing storage from compute resources co-located within the same data center, or for data requiring a specific geographic location. More information can be found in the Select Regions and Endpoints documentation.

8 August 2017

Introducing IBM Cloud Object Storage

Object Storage is a highly available, durable, and secure platform for storing unstructured data. Unstructured data (sometimes called binary or "blob" data) refers to data that is not highly structured in the manner of a database. Object storage is the most efficient way to store PDFs, media files, database backups, disk images, or even large structured datasets.

Create a Secure Content Store

Are you looking to store content securely (locally or globally) at an affordable cost, for things like **cloud native apps**, **media storage**, **backup storage** and **archive data**? IBM Secure Content Store powered by IBM Cloud® Object Storage provides unparalleled agility in supporting fast, highly consistent application deployment around the world to help customers securely expand their business into new regions, from business-critical data to video archive solutions. It also offers immutable storage, immutable backup, and archive data with industry-leading security and controls for regulatory/compliance requirements.

- Gain security and control over your data with encryption options, governance policy, access permissions, and context-based restrictions.
- Have immediate consistency across regions or locations for cloud-native apps, disaster recovery, storage backup, video content and delivery, and so on.
- Leverage your own encryption keys (BYOK) with Key Protect.
- Monitor and retain your account & data activity with Activity Tracker and IBM Monitoring.
- APIs & SDKs, Static Web Hosting, High Speed Transfer, Tagging, Replication.

Overview

This tutorial is for customers looking to set up a Secure Content Store using Object Storage, Activity Tracker, and Key Protect. In this tutorial, you are guided through the process of quickly getting started with these essential services to ensure the security and integrity of your content. Secure Content Store is comprised of the following services:

- **Object Storage**: a scalable and flexible storage solution that allows you to store and manage your data securely.
- **Activity Tracker**: a powerful tool that provides comprehensive visibility into the activities happening within your IBM Cloud environment and allows for ease of audit observability.
- **Monitoring**: to provide insights and information about what is happening with your data in your Secure Content Store.
- **Key Protect**: a Key Management Service that enables you to manage and protect your encryption keys in a secure and centralized manner.

Throughout the tutorial, you are provided with step-by-step instructions, along with helpful tips and best practices, which can help you set up a Secure Content Store more efficiently. So, let's get started!

High level steps for the tutorial

1. Set up Object Storage to store and manage your data securely.
2. Configure Activity Tracker for audit observability of relevant events.
3. Add Monitoring for insights and information about what is happening with your data.
4. Finally, use Key Protect to manage encryption keys to secure your data stored in Object Storage.

Before you begin

For this tutorial, you need:

- An IBM Cloud® Platform account
- An instance of IBM Cloud Object Storage (must be a paid service plan instance)

Create a new Object Storage bucket

Step 1: Navigate to your instance of Object Storage

- Go to your instance of IBM Cloud Object Storage.

Step 2: Click Create bucket

1. Select the **Customize your bucket** tile, and click the right arrow.
   a. Name the new bucket. It must start and end in alphanumeric characters (from 3 to 63) that is limited to using lowercase, numbers and nonconsecutive dots, and hyphens.
   b. Choose your desired region and storage class, based on your activity (for example, chose “Standard” storage class for hot data, “Vault” or “Cold Vault” for cold data, or “Smart Tier” for blended or variable data activity.)
2. Add the following services during the bucket creation by scrolling down to **Service integrations (optional)**.

Key Protect

Before you get started, you need:

- An instance of IBM Cloud™ Key Protect
- Grant service authorization to Object Storage in IBM Key Protect.

a. Toggle **Key management disabled** to enable encryption and **click** on **Create new instance**.

b. Choose a region that corresponds with the bucket, give it a memorable name, and click **Create and continue**.

c. Give the `root key` a name and click **Create and continue**.

Activity Tracker

Before you get started, you need:

- An instance of Activity Tracker
- A user ID with administrator platform permissions and the service access writer role.

a. Scroll down to the **Monitoring and activity tracking** section and toggle the radio button to **Activity tracking enabled**. Select an appropriate plan, and give the new instance a memorable name. As you may likely want to create the Activity Tracker instance in the same region as the bucket (for example, `us-east`) you could name the instance something like `US East AT` so that you can easily find it later.

b. Click to enable **Track data events** and select both **read & write** from the drop-down list.

Monitoring

Before you start, you need:

- An instance of IBM Cloud™ Monitoring
- A user ID with administrator platform permissions and the service access writer role.

a. Scroll down to the **Monitoring and activity tracking** section and toggle the radio button to **Monitoring enabled**. Select an appropriate plan, and give the new instance a memorable name. For example, if you are creating the instance in the same region as the bucket (for example, `us-east`) you could name the instance `US East MM` so that you can easily find it later.

b. Enable monitoring for both **usage and request metrics**.

Step 3: Verify the information is correct

Step 4: Click Create bucket to add the new bucket to your instance of Object Storage

After your bucket is created with Activity Tracker and Monitoring, it may take a few minutes for the rules to take effect.

You are now ready to store data in a secure content store with encryption, monitoring, and audit observability!

Get started by uploading data

- See uploading data for more information.

Add capabilities

Add capabilities to protect objects from ransom-ware and accidental deletion such as versioning and immutable retention polices for supporting immutable storage, and immutable backup and archive data.

Library of Object Storage tutorials

Check out the IBM Cloud Tutorials library for more tutorials when deploying solutions with Cloud Object Storage.

Migrating from AWS

There are many tools to assist you to successfully migrate your information from AWS to IBM Cloud® Object Storage, with more secure and globally accessible results.

Before you begin

Determine your goals and process for your migration before starting your migration. You may also consider training and partnerships to be beneficial. Your planning and assessment stage will consider many possibilities, including security and technical capabilities.

Documentation for any project will help keep you keep track of your resources as well as your goals. After assessing your existing projects, you may benefit by updating them to use IBM Cloud Object Storage libraries like those for (Java, Python, Node.js). If you're interested in programmer interfaces, the REST API will provide an in-depth look at operations and configurations.

Refer to the getting started guide to familiarize yourself with key concepts such as endpoints and storage classes.

Provision and configure IBM Cloud Object Storage

1. If you haven't already, create an instance of IBM Cloud Object Storage from the Console.
2. Create any buckets that you anticipate will be needed to store your transferred data.
3. While Object Storage is compatible with the S3 API, it may be necessary to create new Service credentials, or bring your own keys for your projects. In this guide, we will use HMAC credentials similar to the format of AWS credentials.
4. Managing encryption provides insights into security. Refer to product documentation on IBM® Key Protect for IBM Cloud® and Hyper Protect Crypto Services for more information.

Determine your solution

It is true that a massively complex migration requires a complete service to plan and implement migrating your data to IBM Cloud Object Storage. But whatever the size of your data, your goals and timetable take precedence. Once you have provisioned and set your target, it is time to choose a process to achieve your goals on your time.

There are many ways to achieve the goal of migrating your AWS data. Integrated solutions provide comprehensive guides to migration, as shown in the IBM Cloud Pak for Integration. In addition to full-featured migration services, you may also want to investigate third party migration tools as part of your investigation. But don't forget that there are many CLI and GUI tools readily available for use as part of your migration.

- COS CLI can be used for many operations. For example, you may wish to use the CLI to configure your IBM Cloud Object Storage instances, and to create and configure buckets.
- AWS CLI can be used to list your current bucket's contents to prepare for migrating from AWS, among other operations:

```
$ aws s3 ls --recursive s3://<BUCKET_NAME> --summarize > bucket-contents-source.txt
```

- `rclone` has many uses, and we'll look at it specifically, next.

Migrate your data

Based on the process and tools you've chosen, choose a strategy for migrating your data. Here is a simplified process using the command line and the Go-based `rclone` executable as an example.

1. Install `rclone` from either a package manager or pre-compiled binary . There are more configuration options available with explanations at the IBM Cloud Object Storage documentation.

   ```
   $ curl https://rclone.org/install.sh | sudo bash
   ```

Configure `rclone` with your AWS credentials

Start by creating 'profiles' for your source and destination of the migration in `rclone`. A profile contains the configuration and credentials needed for working with your date. To migrate from AWS, those credentials are needed to continue. Also, create a profile for your destination credentials specifically for IBM Cloud Object Storage.

1. There are many options to configuring `rclone` and following the `rclone config` wizard is one way you can create profiles. You can create an `rclone` config file in `~/.rclone.conf` by using the command as shown. Please use the root path of your home directory if the path shown isn't available.

```
$ touch ~/.config/rclone/rclone.conf
```

2. Create the AWS configuration settings by copying the following and pasting into `rclone.conf` using an appropriate editor.

```
$ [AWS]
type = s3
provider = AWS
env_auth = false
access_key_id =
secret_access_key =
region =
```

3. Paste your AWS `access_key_id` and `secret_access_key` as obtained per instructions here into the appropriate fields of your configuration as shown.

Configure `rclone` with your COS credentials

To complement the credentials of the source, we look at configuring the destination profile next.

1. Create the COS configuration settings by copying the following and pasting into `rclone.conf` using an appropriate editor.

```
$ [COS]
type = s3
provider = IBMCOS
env_auth = false
region =
access_key_id =
secret_access_key =
endpoint =
```

2. Paste your HMAC `access_key_id` and `secret_access_key` into the appropriate fields of your configuration as shown in the first step. As noted in the beginning of the guide, you will want to enter the appropriate values for your instance regarding your region and endpoint.

Verify your configurations

1. List the buckets from your source to verify `rclone` is properly configured for retrieval.

```
$ rclone lsd AWS:
```

2. List the COS bucket for your destination you created to verify `rclone` is properly configured for storage.

```
$ rclone lsd COS:
```

Use `rclone` to migrate from AWS

1. Do a dry run (no data copied) of `rclone` to sync the objects in your source bucket (for example, `content-to-be-migrated`) to the target COS bucket (for example, `new-bucket`).

```
$ rclone --dry-run copy AWS:content-to-be-migrated COS:new-bucket
```

2. Check that the files you want to migrate appear after running the command. If everything looks as you expect, remove the `--dry-run` flag and add a `-v` flag to show a verbose output while the data is being copied. Using the optional `--checksum` flag avoids updating any files that have the same MD5 hash and object size in both locations.

```
$ rclone -v copy --checksum AWS:content-to-be-migrated COS:new-bucket
```

As you perform the migration of your data using the process you've outlined, you will want to validate and verify the results.

Validating your migration from AWS

Integrated query-in-place dashboards allows you to see analytics based directly on your data. Using IBM Cloud Monitoring, you can follow up your migration using pre-built charts.

Next Steps

Get started by visiting the catalog, and creating the resources to begin your journey from AWS to IBM Cloud Object Storage with confidence and efficiency.

Limiting access to a single Object Storage bucket using the UI

IBM Cloud IAM resource groups and access policies allow administrators to restrict user access to various service instances. But what if you are using the IBM Cloud user interface and only need to access a limited number of buckets within a service instance? This can be accomplished using a custom role and a narrowly-tailored IAM policy.

This tutorial provides an introduction to granting access to a single Object Storage bucket for a user who needs to use the IBM Cloud UI to access the bucket.

If you are not familiar with IBM Cloud® Object Storage, you can quickly get an overview by getting started with IBM Cloud Object Storage. Also, if you are not familiar with IAM, you may wish to check out how to get started with IAM.

Before you begin

If you are already managing instances of Object Storage or IAM, you do not need to create more. However, as this tutorial will modify and configure the instance you are working with, make sure that any accounts or services are not being used in a production environment.

This tutorial will create a new access policy and a new custom role in the process.

For this tutorial, you need:

- An IBM Cloud® Platform account
- An instance of IBM Cloud Object Storage. If you do not have a COS instance, you can create one. For purposes of this tutorial, name the instance `COS-BUCKET-LIMIT-EX`.
- A bucket to which a user should be constrained. If you have created a new instance for the tutorial, create several buckets after you have created the COS instance. The tutorial will describe how to provide access to only one of the buckets in the COS service instance.
- To complete the steps to manage access to the service, your user ID needs **administrator platform permissions** to configure the IAM policy. You may have to contact or work with an account administrator.

Provide bucket-level access to the individual users

1. Create a custom COS Service role (call it `COS ListBucketsInAccount`, for example) and assign the action `cloud-object-storage.account.get_account_buckets` to this custom role.
2. Create an IAM Access Group and call it `BUCKET_ACCESS_GROUP_1`, for example.
3. In that new access group, create an instance-level access policy for the instance `COS-BUCKET-LIMIT-EX`, and assign the platform role `Viewer` and the custom role you just created, `COS ListBucketsInAccount`.
4. In the same access group, create a bucket-level access policy for one of the buckets in the instance and assign the COS service roles `Content Reader` and `Object Writer`.

   Note: What levels of access you want here are going to determine the roles you specify. This example is for a minimal object list, and upload and download in one bucket. For more information see: Assigning access to an individual bucket

5. Invite a user to the account.
6. Once the user has accepted the invitation to the account, add the user to the access group `BUCKET_ACCESS_GROUP_1`.
7. Now when the user logs in, if they are already members of other IBM Cloud® accounts, ensure that they select the correct IBM Cloud® account in the account selector in the console header.
8. Once the user has selected the correct account in the account selector, the user sees only one COS instance in the console resource view, `COS-BUCKET-LIMIT-EX`.
9. Select the instance.

Note: Both buckets are listed, but users will only be able to list objects in the bucket they are given access to and, depending on the accesses you provided, download objects from that bucket and upload objects to that bucket.

Note: The user can appear to select the other bucket but they cannot list objects, cannot download objects, and cannot upload to that other bucket, and so on.

Next steps

Congratulations, you've just set up a policy to limit user access to a single bucket when they must use the IBM Cloud user interface for their access.

Controlling access to individual objects in a bucket

This tutorial provides examples for how to use IAM access policies with IBM Cloud® Object Storage buckets to grant users access to individual objects within a bucket.

Before you begin

IBM Cloud Object Storage stores data in a flat structure where one bucket can contain billions of distinctly named objects. A folder hierarchy can be simulated by using identical prefixes in related object names. Also, an object name can be referred to as an object key. Here is an example:

```
$ Bucket Name: MyBucket
Objects in MyBucket:

User1/userDoc1.txt
User1/userDoc2.zip
Engineering/project1.git
Engineering/project2.git
Product/2023/roadmap1.ppt
Product/2024/roadmap2.ppt
Orgchart.pdf
```

In this example, the prefix `User1`, `Engineering`, and `Product` can resemble root level folders. In addition, `2023` and `2024` can represent subdirectories. Use the delimiter "/" to represent the file hierarchy. A delimiter can be any supported character. `Orgchart.pdf` is considered a root-level object.

When running a list request on your bucket, you can specify a prefix for listing objects or list the content of the entire bucket. In addition, you can optionally pass a delimiter value in the listing request to simulate a folder structure in the response. For more information, see the examples of using prefix and delimiter condition statements.

Read or write operations typically target a specific object name which is also referred to as the object path.

Scenarios

The following examples show how to use IAM policies and conditions to grant access to individual objects in a bucket. We will continue to use the example bucket shown above. These examples show excerpts of the full access policy with respect to configuring the condition statements. For more information, see Assigning access to objects within a bucket using IAM access conditions .

Scenario 1: Grant Adam read access to all objects in the `User1` folder only.

This will give Adam the ability to read all objects that start with the key name of `User1/`. This will not give Adam the ability to list objects and he therefore cannot navigate the UI to access these objects. Adam can only retrieve objects in the `User1` folder through non-UI methods. Use a wildcard in the policy to give access to all possible objects that begin with `User1/`. Failure to include a wildcard would give Adam only access to the object named `User1/`.

```
"control": {
    "grant": {
      "roles": [
         { "role_id":
             "crn:v1:bluemix:public:cloud-object-storage::::serviceRole:ObjectReader"
         }
       ]
     }
   },
   "rule": {
     "conditions": [
       {
         "key": "{{resource.attributes.path}}",
         "operator": "stringMatch",
         "value": "User1/*"
       }
     ]
   },
   "pattern": "attribute-based-condition:resource:literal-and-wildcard"
```

Scenario 2: Grant Adam list and read access to all objects in the `User1` folder.

This will give Adam the ability to read and list all objects that start with the key name of `User1/`. Also, use a wildcard in the `prefix` condition attribute.

Failure to include the wildcard results in Adam only having List access to the first level of objects or folders in the `User1` folder. This policy will not permit Adam to list the bucket at the root level. If Adam uses the UI, he must search the bucket with the specific prefix of `User1/` to see the objects for which he has access.

```
"control": {
    "grant": {
      "roles": [
        {
          "role_id":
          "crn:v1:bluemix:public:cloud-object-storage::::serviceRole:ContentReader"
        }
      ]
    }
  },
  "rule": {
    "operator": "or",
    "conditions": [
      {
        "operator": "and",
        "conditions": [
          {
            "key": "{{resource.attributes.prefix}}",
            "operator": "stringMatch",
            "value": "User1/*"
          },
          {
            "key": "{{resource.attributes.delimiter}}",
            "operator": "stringEquals",
            "value": "/"
          }
        ]
      },
      {
        "key": "{{resource.attributes.path}}",
        "operator": "stringMatch",
        "value": "User1/*"
      }
    ]
  },
  "pattern": "attribute-based-condition:resource:literal-and-wildcard
```

Scenario 3: Grant Samantha access to list, read, and replicate files in only the `2023` and `2024` subdirectories under the `Product` folder.

These sets of actions will require Samantha to have at least the `Writer` role. The `Writer` role also contains some actions that do not specify a `Path` or a `Prefix` or `Delimiter` such as `cloud-object-storage.bucket.put_replication`. To allow these actions, use the StringExists operator with the resource attributes based conditions.

Samantha will not have access to navigate the UI from the root folder. This situation is shown in Scenario 4.

```
"control": {
    "grant": {
      "roles": [
        {
          "role_id": "crn:v1:bluemix:public:iam::::serviceRole:Writer"
        }
      ]
    }
  },
  "rule": {
    "operator": "or",
    "conditions": [
      {
        "operator": "and",
        "conditions": [
          {
            "key": "{{resource.attributes.prefix}}",
            "operator": "stringMatchAnyOf",
            "value": [
              "Product/2023/*",
              "Product/2024/*"
            ]
          },
          {
            "key": "{{resource.attributes.delimiter}}",
            "operator": "stringEquals",
            "value": "/"
          }
        ]
         },
         {
            "key": "{{resource.attributes.path}}",
            "operator": "stringMatchAnyOf",
            "value": [
              "Product/2023/*",
              "Product/2024/*"
        ]
    },
    {
      "operator": "and",
      "conditions": [
        {
          "key": "{{resource.attributes.delimiter}}",
          "operator": "stringExists",
          "value": false
        },
        {
          "key": "{{resource.attributes.prefix}}",
          "operator": "stringExists",
          "value": false
        },
        {
          "key": "{{resource.attributes.path}}",
          "operator": "stringExists",
          "value": false
        }
      ]
    }
  ]
},
"pattern": "attribute-based-condition:resource:literal-and-wildcard"
```

Scenario 4: Grant Samantha access to navigate the UI to the files in the `2023` and `2024` folders in addition to list, read and replicate files in `2023` and `2024`.

To navigate the UI to `MyBucket`, Samantha needs the platform role `Viewer`. In addition, Samantha is given access to any directories above the target folder. In this case, Samantha needs access to list the root level (defined by the prefix of the empty string) and the `Product/` folder. This allows Samantha to see all root-level folders and objects.

```
"control": {
    "grant": {
      "roles": [
        {
          "role_id": "crn:v1:bluemix:public:iam::::serviceRole:Writer"
        },
        {
          "role_id": "crn:v1:bluemix:public:iam::::role:Viewer"
        }
      ]
    }
  },
  "rule": {
    "operator": "or",
    "conditions": [
      {
        "operator": "and",
        "conditions": [
          {
            "key": "{{resource.attributes.prefix}}",
          "operator": "stringMatchAnyOf",
          "value": [
            "Product/2023/*",
            "Product/2024/*"
        ]
      },
      {
        "key": "{{resource.attributes.delimiter}}",
        "operator": "stringEquals",
        "value": "/"
        }
      ]
    },
    {
      "operator": "and",
      "conditions": [
        {
          "key": "{{resource.attributes.prefix}}",
          "operator": "stringEqualsAnyOf",
          "value": [
            "",
            "Product/"
          ]
        },
        {
          "key": "{{resource.attributes.delimiter}}",
          "operator": "stringEquals",
          "value": "/"
        }
      ]
    },
    {
      "key": "{{resource.attributes.path}}",
      "operator": "stringMatchAnyOf",
      "value": [
        "Product/2023/*",
        "Product/2024/*"
      ]
    },
    {
      "operator": "and",
      "conditions": [
        {
          "key": "{{resource.attributes.delimiter}}",
          "operator": "stringExists",
          "value": false
        },
        {
          "key": "{{resource.attributes.prefix}}",
          "operator": "stringExists",
          "value": false
        },
        {
          "key": "{{resource.attributes.path}}",
          "operator": "stringExists",
          "value": false
        }
      ]
    }
  ]
},
"pattern": "attribute-based-condition:resource:literal-and-wildcard"
```

Encrypting a bucket with Key Protect

While all data stored in Cloud Object Storage is automatically encrypted using randomly generated keys, some workloads require that the keys can be rotated, deleted, or otherwise controlled by a key management system (KMS) like Key Protect.

Before you begin

Before you plan on using Key Protect with Cloud Object Storage buckets, you need:

- An IBM Cloud™ Platform account
- An instance of IBM Cloud Object Storage
- An instance of Key Protect

You will also need to ensure that a service instance is created by using the IBM Cloud catalog and appropriate permissions are granted. This tutorial does not outline the step-by-step instructions to help you get started. This information is found in section Server-Side Encryption with IBM Key Protect (SSE-KP)

Step 1: Create a new encryption key

1. Using the **Navigation Menu**, go to **Resource List** and expand **Security**.
2. Click a **Key Protect** instance.
3. Click the **Add** button.
4. Click the **Root key** tab.
5. Enter a Key name.
6. Click **Advanced Option** and enter a Key description.
7. Click the **Add key** button. Your new encryption key is listed in the **Keys** table.

Step 2: Create a new bucket and associate the key with it

1. Using the **Navigation Menu**, go to **Resource List** and expand **Storage**.
2. Click your **Storage** instance.
3. Click **Create bucket**.
4. Click **Create** in the **Create a Custom Bucket** pane.
5. Enter a unique bucket name.
6. Select **Resiliency>Regional**.
7. Select a **Location**.
8. Select a **Storage Class**.
9. Enable **Service integrations>Encryption>Key management**.
10. Click **Key Protect>Use existing instance**.
11. Select the **Search by instance** tab in the **Key Protect integration** side panel.
12. Select a Key Protect instance from the menu.
13. Select the **Key name** that you just created.
14. Click the **Associate key** button.
15. Click the **Create bucket** button. A popup message displays that a bucket was created successfully.
16. Confirm by clicking the **Configuration** tab.
17. Click **Jump to>Key management** (or scroll down the page).
18. In the **Associated key management services** box see **Service instance** and the **Key** that was associated with the bucket.

Securing data using context-based restrictions

In this tutorial, you will establish context-based restrictions that prevent any access to object storage data unless the request originates from a trusted network zone.

Before you begin

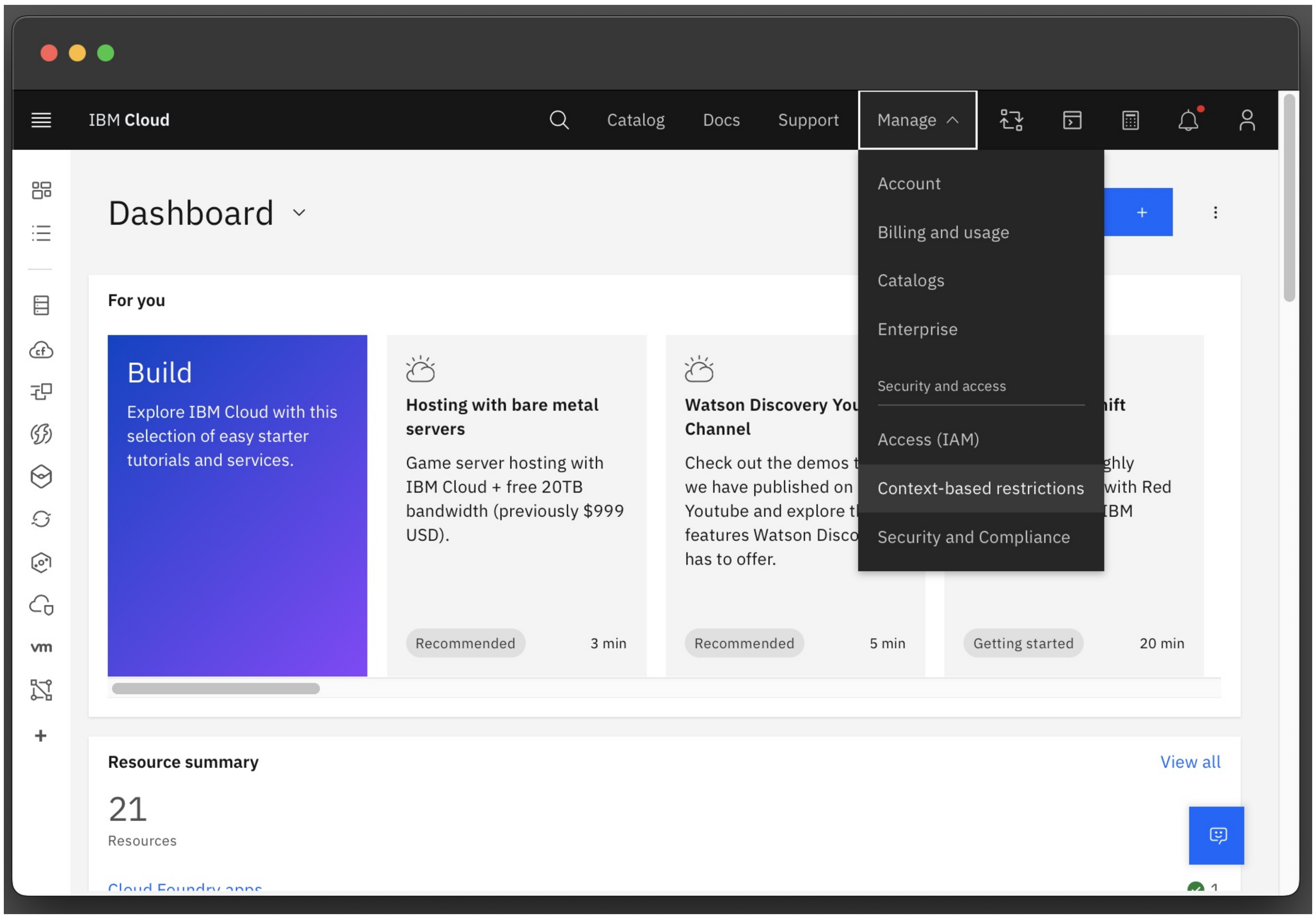
Before you plan on using context-based restrictions with Cloud Object Storage buckets, you need:

- An IBM Cloud™ Platform account
- An instance of IBM Cloud Object Storage
- A role of `Administrator` for context-based restrictions
- A bucket

Step 1: Navigate to the context-based restrictions console

From the **Manage** menu, select **Context-based restrictions**.

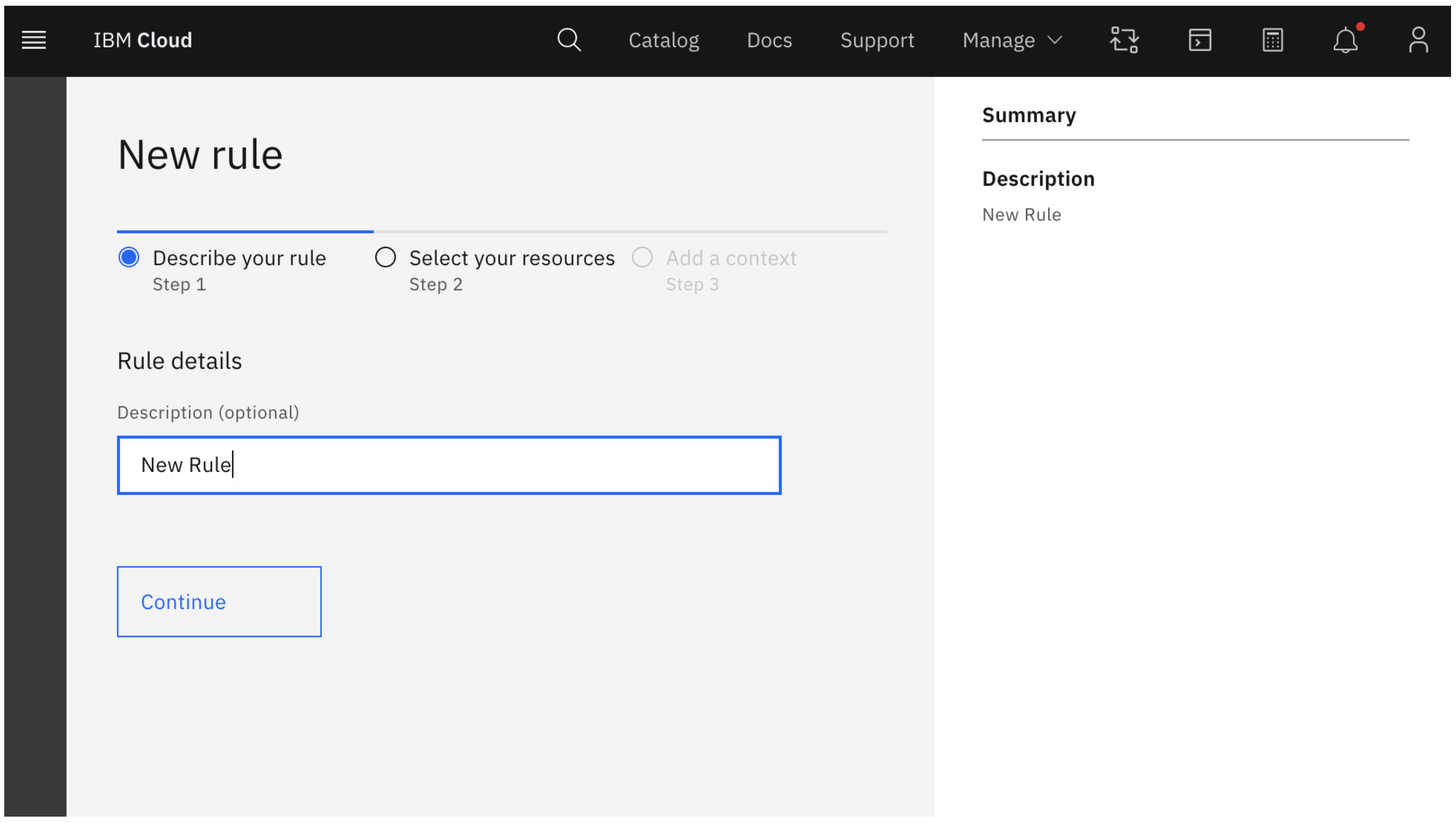
Navigate to CBR


IBM Cloud
Catalog
Docs
Support
Manage
Account
Billing and usage
Catalogs
Enterprise
Security and access
Access (IAM)
Context-based restrictions
Security and Compliance
Dashboard
For you
Build
Explore IBM Cloud with this selection of easy starter tutorials and services.
Hosting with bare metal servers
Game server hosting with IBM Cloud + free 20TB bandwidth (previously $999 USD).
Recommended
3 min
Watson Discovery You Channel
Check out the demos t we have published on Youtube and explore t features Watson Disco has to offer.
Recommended
5 min
Getting started
20 min
Resource summary
View all
21
Resources

Step 2: Create a new rule

1. Click on **Rules**.
2. Choose a name for the rule. This will help keep things organized if you end up with a lot of different rules across all of your cloud services.
3. Click **Continue**.

Name the rule



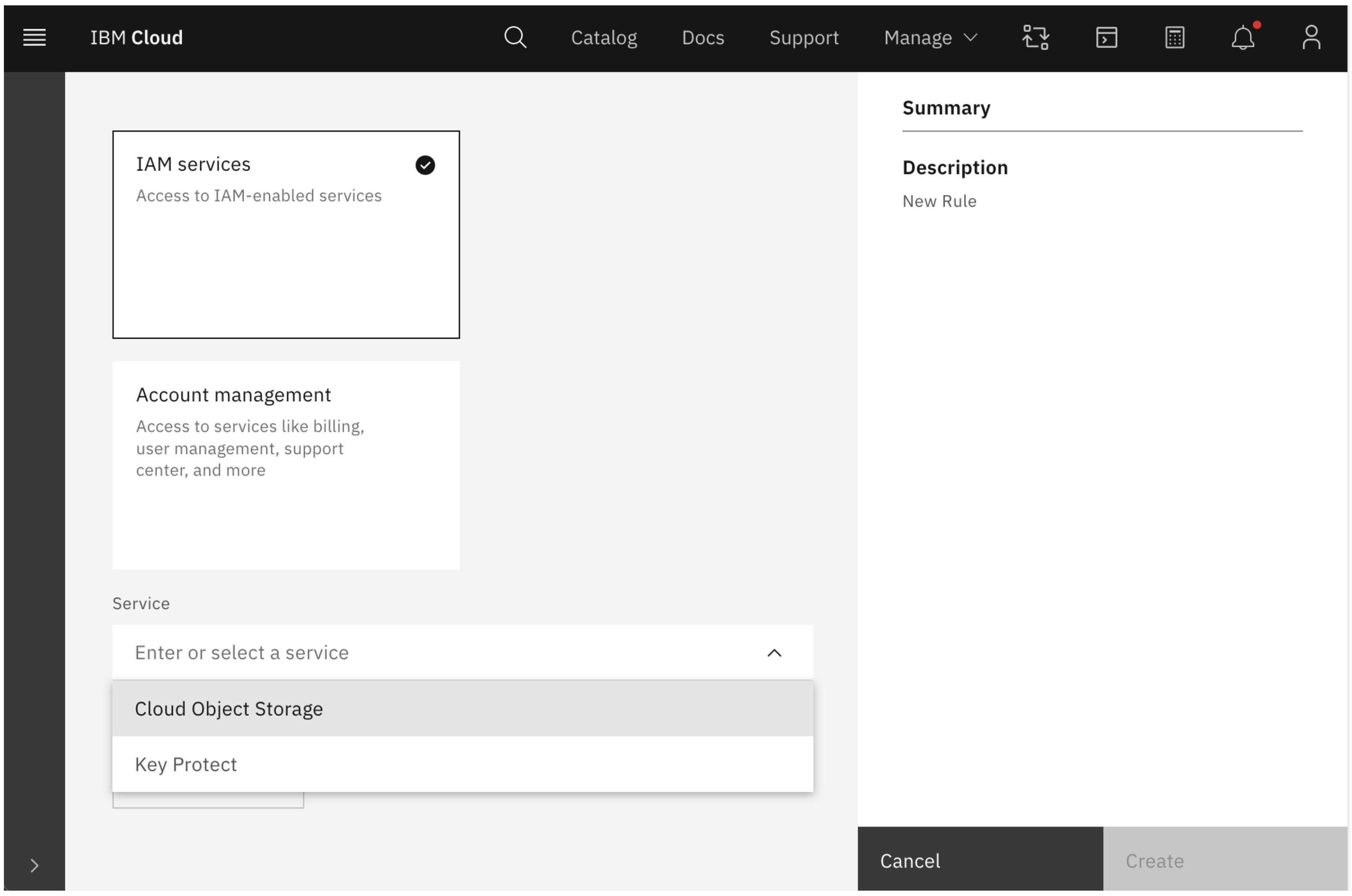
Step 3: Scope the rule

Now you can choose the specific object storage resources to which you would like to apply the context-based restrictions. This can become as specific or generic as you wish - you could apply the rule to all object storage instances and buckets, a specific service instance, or even a specific bucket. Additionally, you can choose which networks (public, private, or direct) you wish to be included.

In this example, we will choose a service instance.

1. Select **IAM services**.
2. Choose **Cloud Object Storage** from the drop down menu.
3. Select the **Resources based on specific attributes** radio button.
4. Check the **Service instance** box.
5. Select the service instance you want the rule to affect.

Scope the rule



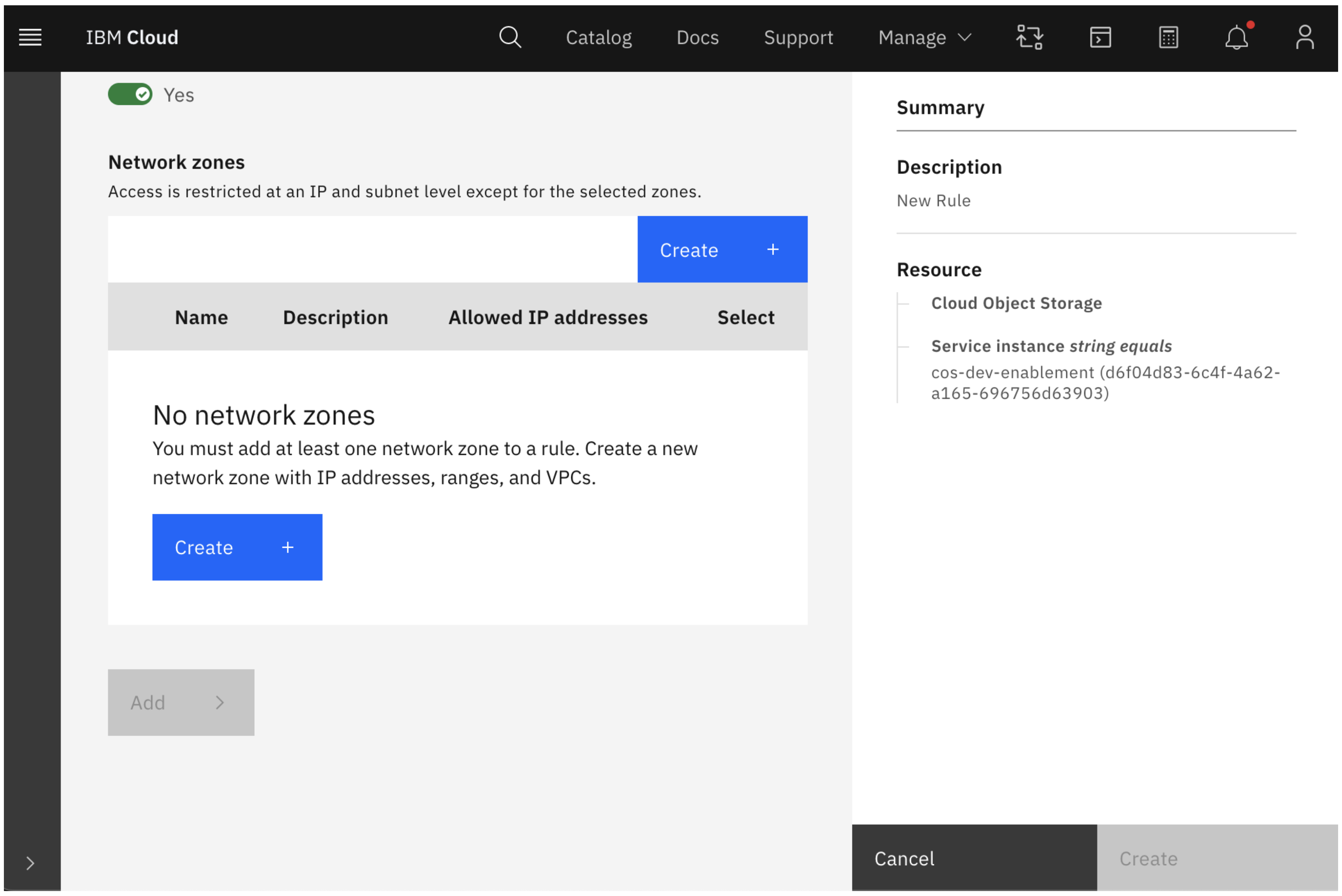
Tip: If you want to instead only limit access to a specific bucket, you can select the **Resource ID** checkbox instead. Provide the name of the bucket in the field - nothing else is necessary.

Step 4: Create a network zone

Now that we know what the rule will affect, we need to decide what the rule will allow. To do this, we'll create a new *network zone* and apply it to the new rule.

1. Click on **Create +**.

Scope the rule



Yes

Network zones

Access is restricted at an IP and subnet level except for the selected zones.

Name	Description	Allowed IP addresses	Select

No network zones

You must add at least one network zone to a rule. Create a new network zone with IP addresses, ranges, and VPCs.

Summary

Description

New Rule

Resource

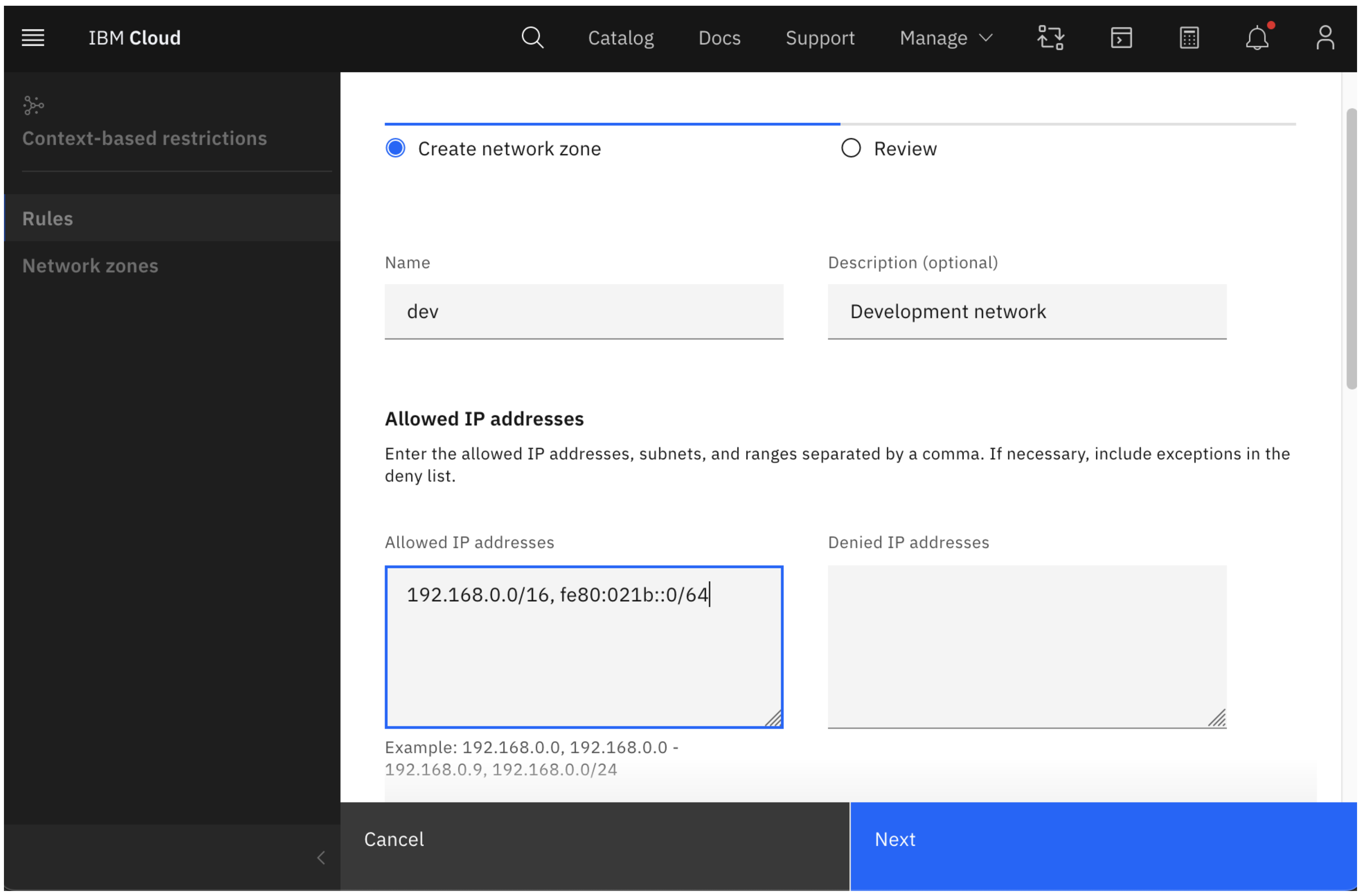
Cloud Object Storage

Service instance *string equals*

cos-dev-enablement (d6f04d83-6c4f-4a62-a165-696756d63903)

2. Give the network zone a helpful name and description.
3. Add some IP ranges to the **Allowed IP addresses** text box.
4. Click **Next**.

Scope the rule



Name: dev

Description (optional): Development network

Allowed IP addresses

Enter the allowed IP addresses, subnets, and ranges separated by a comma. If necessary, include exceptions in the deny list.

Allowed IP addresses: 192.168.0.0/16, fe80:021b::0/64

Denied IP addresses:

Example: 192.168.0.0, 192.168.0.0 - 192.168.0.9, 192.168.0.0/24

Step 5: Finish the rule and verify that it works

Finally, all you need to do is click **Create** and your new rule will be active.

An easy way to check that it works is to [send a simple CLI command] from outside of the allowed network zone, such as a bucket listing (`ic cos buckets`). It will fail with a `403` error code.

Next steps

Learn more about [context-based restrictions and how they relate to legacy bucket firewalls](#).

Building a Static Website

This tutorial shows how to host a static website on IBM Cloud® Object Storage, including creating a bucket, uploading content, and configuring your new website.

Hosting static websites with IBM Cloud Object Storage serves static content for public access giving users flexibility, ease of delivery, and high availability. This tutorial contains instructions for using cURL, the AWS CLI, as well as the Console. Choose your path for this tutorial by using the links for switching between the instructions above the title of this topic.

The Scenario

The scenario for this tutorial simplifies web hosting to its essentials to highlight the steps involved. While not every configuration option will be covered in this tutorial, correctly completing this tutorial results in web-accessible content.

Before you start

Ensure that you have what you need to start:

- An account for the IBM Cloud Platform
- An instance of IBM Cloud Object Storage
- Content in fixed form, like text (HTML would be perfect), and image files

Check that you have the access as appropriate to either the instance of IBM Cloud Object Storage you will be using or the proper permissions for the buckets you will be using for this tutorial.

For use of the IBM Cloud CLI with this tutorial, you will need to configure the Object Storage plug-in to specify the service instance you want to use and the default region where you want your new bucket to be created.

Create a bucket configured for public access

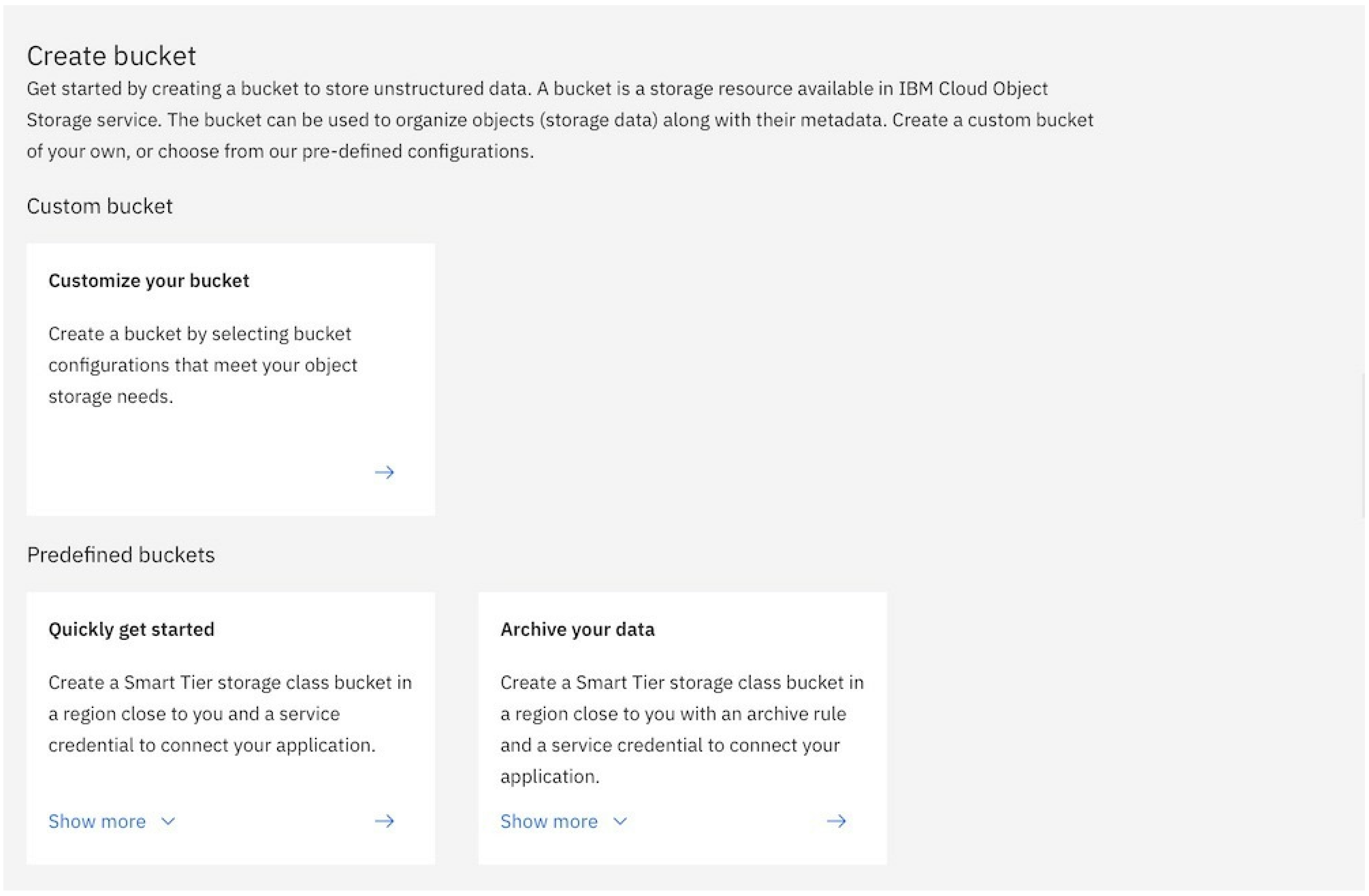
Creating a bucket for a static website will require public access. There are a number of options for configuring public access. Specifically, using the ObjectReader IAM role will prevent the listing of the contents of the bucket while still allowing for the static content to be viewed on the internet. If you want to allow the viewing of the listing of the contents, use the ContentReader IAM role for your bucket.

Create a bucket

After configuring the CLI plug-in, replace the placeholder content as shown in the example command to create a bucket:

```
$ ibmcloud cos bucket-create --bucket <bucketname>
```

Once you login to the Console and after you create an instance of IBM Cloud Object Storage, you can create a bucket. Click on the button labeled "Create bucket" and choose from the options as shown in Figure 1. Select the card that reads "Customize your bucket."


Create bucket

Get started by creating a bucket to store unstructured data. A bucket is a storage resource available in IBM Cloud Object Storage service. The bucket can be used to organize objects (storage data) along with their metadata. Create a custom bucket of your own, or choose from our pre-defined configurations.

Custom bucket

Customize your bucket

Create a bucket by selecting bucket configurations that meet your object storage needs.

Predefined buckets

Quickly get started

Create a Smart Tier storage class bucket in a region close to you and a service credential to connect your application.

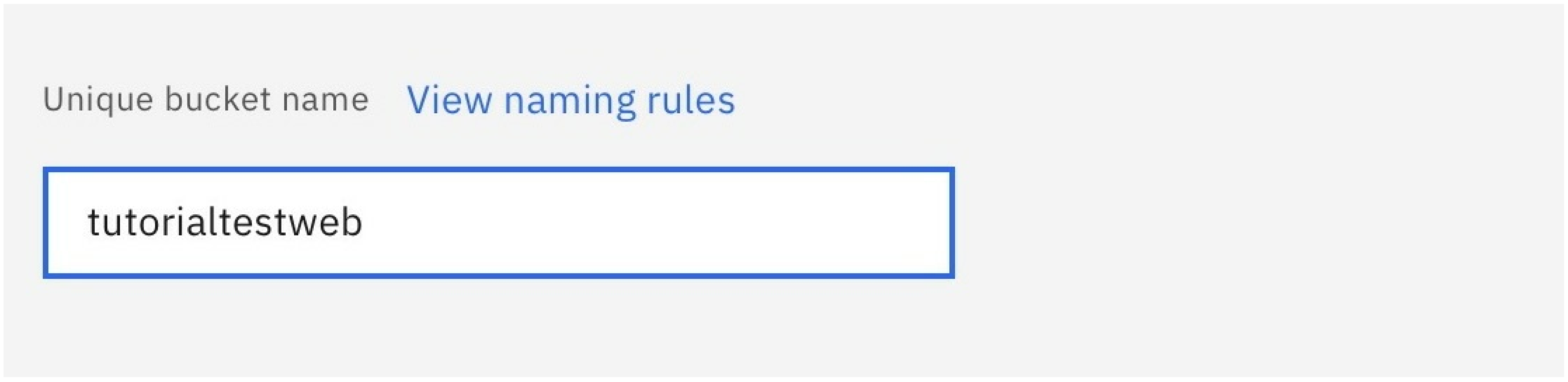
Show more

Archive your data

Create a Smart Tier storage class bucket in a region close to you with an archive rule and a service credential to connect your application.

Show more

The container for the static files in your website will reside in a bucket that you can name. The name you create must be unique, should not contain personal or identifying information, can't have two periods, dots, or hyphens in a row, and must start and end with alphanumeric characters (ASCII character set items 3–63). See Figure 2 for an example.

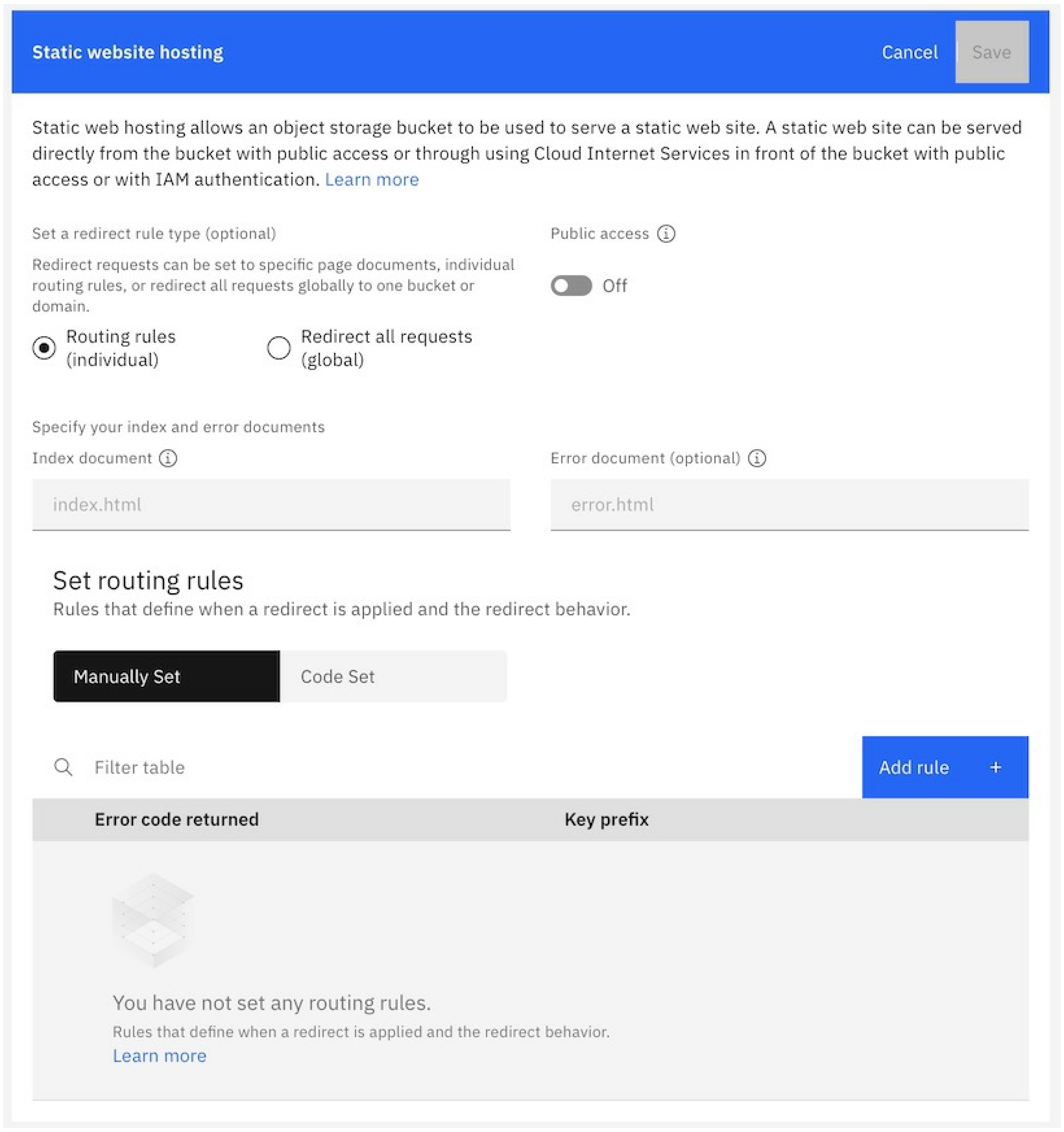


tutorialtestweb

Setting public access

In all scenarios for this tutorial, you will want to use the [UI at the Console] to allow [public access] to your new website.

When creating a bucket for hosting Static Website content, there is an option to enable public access as part of the bucket creation process. See Figure 3 for the option to enable public access to your bucket. For the explanation of the options for the "index document" and "error document" as shown, find more below in the section [Configure the options for your website]. You may complete the basic configuration with this step, before uploading content to your bucket as shown in the next step.



Static website hosting

Static web hosting allows an object storage bucket to be used to serve a static web site. A static web site can be served directly from the bucket with public access or through using Cloud Internet Services in front of the bucket with public access or with IAM authentication. Learn more

Set a redirect rule type (optional)

Redirect requests can be set to specific page documents, individual routing rules, or redirect all requests globally to one bucket or domain.

[✓ Routing rules (individual)] Redirect all requests (global)

Public access: Off

Specify your index and error documents

Index document	index.html
Error document (optional)	error.html

Set routing rules

Rules that define when a redirect is applied and the redirect behavior.

[✓ Manually Set] Code Set

Error code returned	Key prefix

You have not set any routing rules.

Rules that define when a redirect is applied and the redirect behavior.

Learn more

Upload content to your bucket

The content of your hosted static website files focuses naturally on information and media. A popular approach to creating content for static websites are open source generators listed at StaticGen. For the purpose of this tutorial, we only need two files:

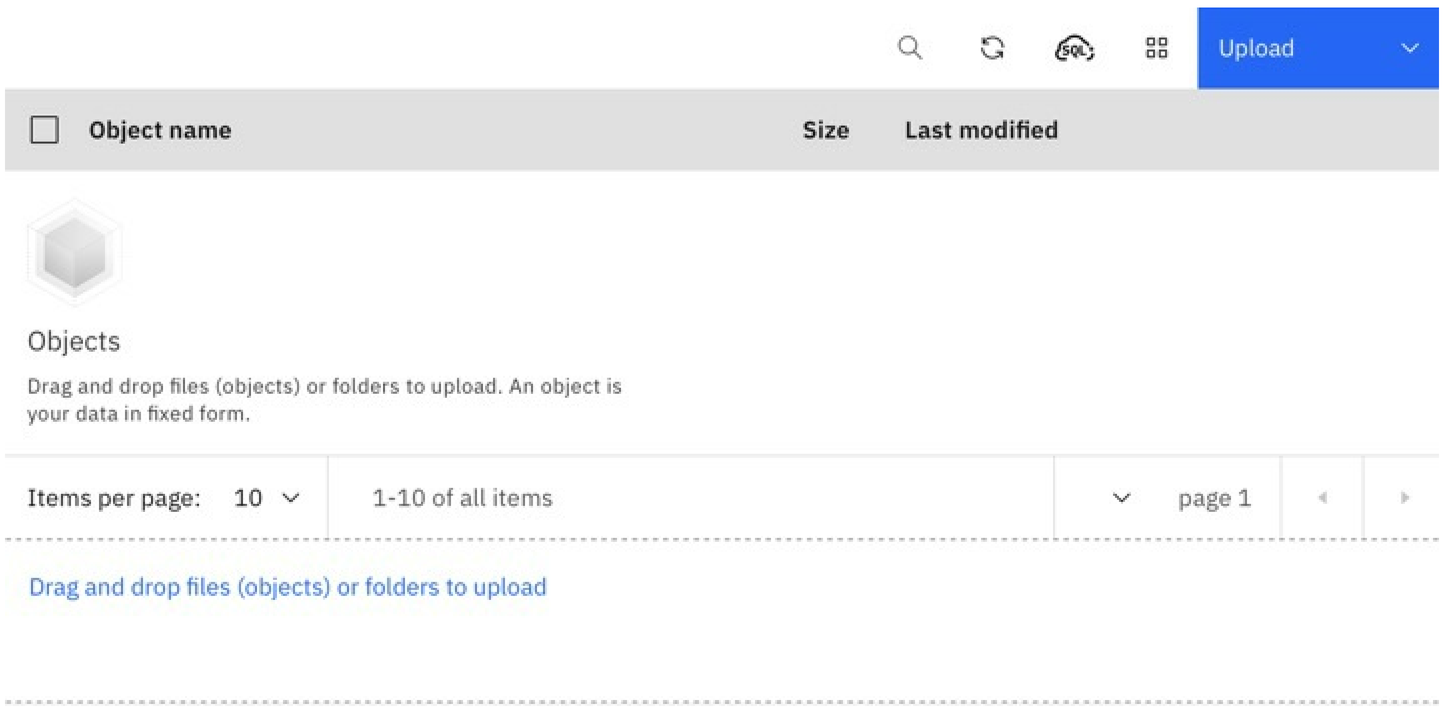
- An index page, typically written in HTML and named `index.html`, that loads by default for visitors to your site
- An error page, also in HTML and here named `error.html`; typically the error page is loaded when a visitor tries to access an object that isn't present or doesn't have public access

Other files, like images, PDFs, or videos, can also be uploaded to your bucket (but this tutorial will focus only on a minimum set of requirements).

For the purpose of this tutorial, place the HTML pages for the index and error handling in a local directory. Replace the placeholder content as shown in the example command to upload your html files:

```
$ ibmcloud cos object-put --bucket BUCKET_NAME --key KEY [--body FILE_PATH]
```

You may have already completed the basic configuration for hosting your static website. Files can be uploaded directly in the Console once you've named and configured your bucket. Note the step is optional as shown in Figure 4, and can occur at any point before the testing of your new hosted website.


Upload
Object name
Size
Last modified
Objects
Drag and drop files (objects) or folders to upload. An object is your data in fixed form.
Items per page: 10
1-10 of all items
page 1
Drag and drop files (objects) or folders to upload

For the rest of the tutorial, we will assume that the object key for the index page is `index.html` and the key for the error document is `error.html` although any appropriate filename can be used for the suffix or key.

Configure the options for your website

There are more options than this tutorial can describe, and for the purpose of this tutorial we only need to set the configuration to start using the static website feature.

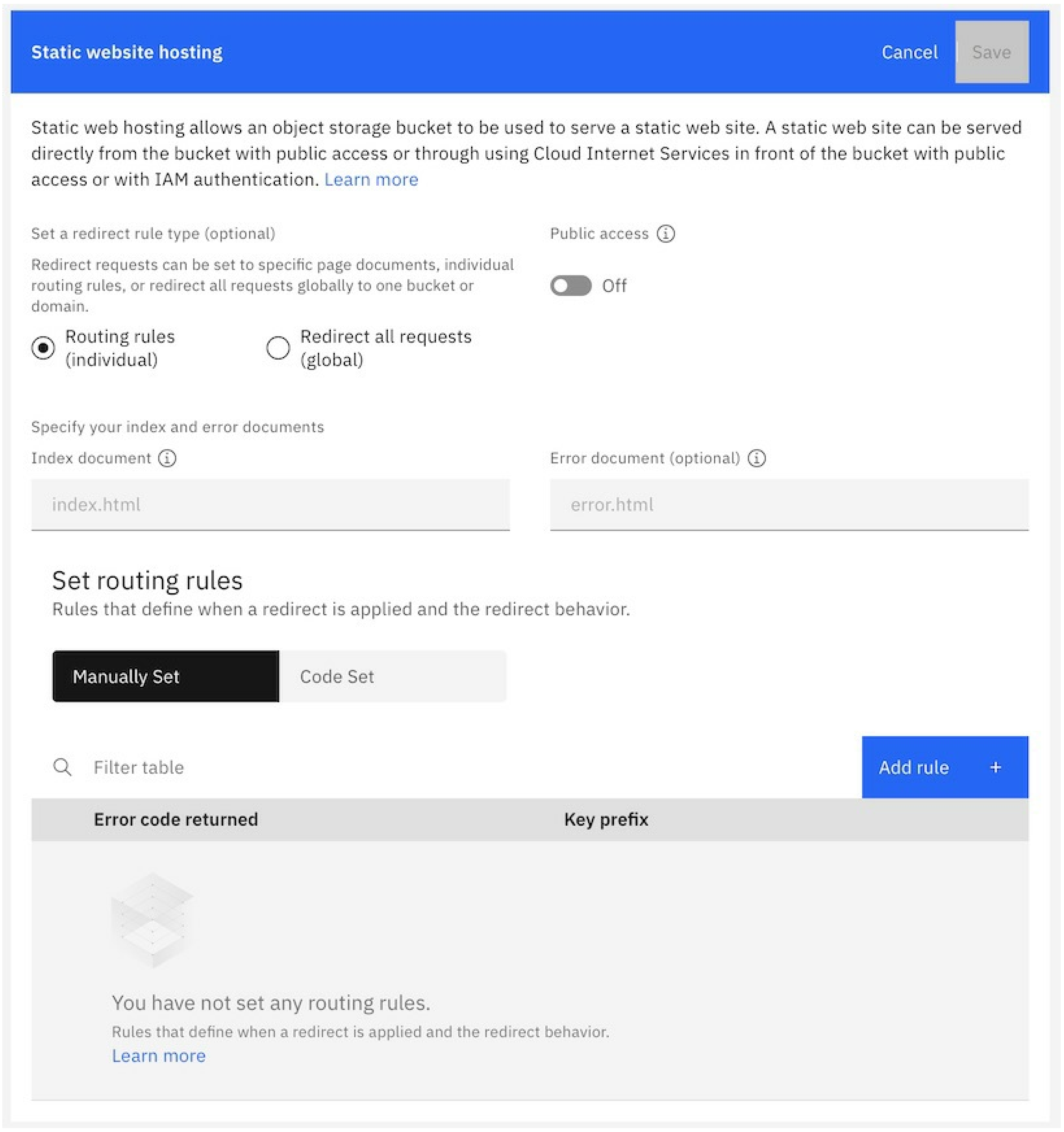
Create a JSON file with the appropriate configuration information:

```
$ {
  "ErrorDocument": {
    "Key": "error.html"
  },
  "IndexDocument": {
    "Suffix": "index.html"
  }
}
```

Replace the placeholder content as shown in the example command to configure the website:

```
$ ibmcloud cos bucket-website-put --bucket BUCKET_NAME --website-configuration file://<filename.json>
```

You may have completed this step during the creation of your bucket, as the basic configuration for your hosted static website determines when and how content is shown. For visitors to your website who fail to provide a key, or web page, the default file will be shown instead. When your users encounter an error, the key for the error page determines what content visitors will receive. The configuration options for the default and error pages are repeated for reference.



Static website hosting

Cancel | Save

Static web hosting allows an object storage bucket to be used to serve a static web site. A static web site can be served directly from the bucket with public access or through using Cloud Internet Services in front of the bucket with public access or with IAM authentication. Learn more

Set a redirect rule type (optional)

Redirect requests can be set to specific page documents, individual routing rules, or redirect all requests globally to one bucket or domain.

[✓ Routing rules (individual)] Redirect all requests (global)

Public access: Off

Specify your index and error documents

Index document: index.html

Error document (optional): error.html

Set routing rules

Rules that define when a redirect is applied and the redirect behavior.

[✓ Manually Set] Code Set

Filter table

Add rule +

Error code returned	Key prefix

You have not set any routing rules.

Rules that define when a redirect is applied and the redirect behavior.

Learn more

Testing and visiting your new website

Once you have configured your bucket to provide HTTP headers using the example command, all you have to do to test your new site is visit the URL for the site. Please note the protocol shown (http), after replacing the placeholders with your own choices made previously in this tutorial:

```
http://<bucketname>.s3-web.<endpoint>/
```

With the successful testing of your new site, you can now explore more options and add more content.

Next steps

The detailed description of configuration options for IBM Cloud Object Storage hosted static websites can be found in the [API Documentation](#).

Developing a web application

This tutorial shows you how to build a simple image gallery using IBM Cloud® Object Storage, bringing together many different concepts and practices key to web development.

From beginning to end, building a web application covers a lot of different concepts and is a great way to introduce yourself to the features of IBM Cloud Object Storage. Your application uses IBM Cloud Object Storage for storage in a Node.js application that allows a user to upload and view JPEG image files.

The Scenario

The scenario for this tutorial involves many moving parts:

- A web server to host the web application
- Use of the command line
- A storage instance for the images in the gallery
- A version control system integrated into continuous delivery
- Client-side application bindings in both scripts and markup
- Images to upload and display

And if you are looking for all that in one package, this tutorial will provide a complete, start-to-finish, example for you. However, this instruction can only temporarily set aside principles of security and secure code. Web applications actually put into production require proper security, or they won't be suitable for possible visitors.

Before you begin

Ensure that you have what you need to start:

- An account for the IBM Cloud Platform
- Docker, as part of the IBM Cloud Developer Tools
- Node.js
- Git (both desktop and command line)

Using the Command Line

Let's start by opening a tool familiar to experienced developers, and a new best friend to those just getting started: the command line. For many, the graphic user interface (GUI) relegated your computer's command-line interface to second-class status. But now, it's time to bring it back (although the GUI isn't going away anytime soon, especially when you need to browse the web to download instructions for the command-line toolset).

Open a shell and create a directory. Change your own reference directory to the new one you created. When created, your application has its own subdirectory with the starter code and configuration needed to get up and running.

Leave the command line and return to your browser so you can follow the instructions to install the IBM Cloud Platform developer tools at the link. The Developer Tools offer an extensible and repeatable approach to building and deploying cloud applications.

Installing Docker

Using containers, like Docker, speeds up development and eases testing and supports automated deployment. A container is a lightweight structure that doesn't need an operating system, just your code and configuration for everything from dependencies to settings.

Docker is installed as part of the Developer Tools, and you need it. Its work takes place mostly in the background within routines that scaffold your new app. Docker must be running for the build commands to work. Go ahead and create a Docker account online at Docker hub, run the Docker app, and sign in.

Installing Node.js

The app that you build uses Node.JS as the server-side engine to run the JavaScript code for this web application. To use the Node Package Manager (`npm`) to manage your app's dependencies, you must install Node locally. Also, a local installation of Node simplifies testing, speeding up development.

Before you start, you might consider a version manager, like Node Version Manager, or `nvm`, to install Node. A version manager reduces the complexity of managing different versions of Node.js.

```
$ curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.34.0/install.sh | bash
```

...or `wget` (just one is necessary, but not both; use whichever is available on your system):

```
$ wget -qO- https://raw.githubusercontent.com/nvm-sh/nvm/v0.34.0/install.sh | bash
```

Or, for Windows, you can use `nvm` for Windows with installers and source code at the link.

Using `nvm`, install Node.

```
$ nvm install v6.17.1
```

Whichever approach you use after you install Node.js and `npm` (included with Node) on your computer, congratulate yourself on a job well started!

Installing Git

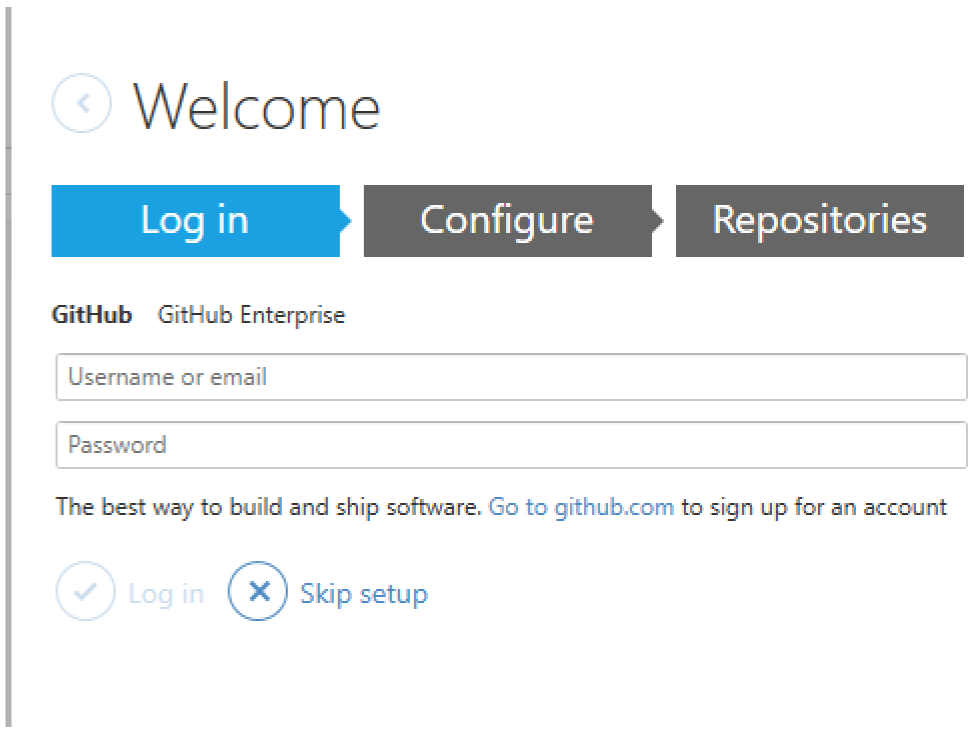
You're probably already familiar with Git, as it's the most widely used source code versioning system. You use Git later when you create a Continuous Deployment (CD) Toolchain in the IBM Cloud Platform for continuous delivery and deployment. If you don't have a GitHub account, create a free public personal account at the GitHub website; otherwise, feel free to log in with any other account you might have.

You need to generate and upload SSH keys to your GitHub profile for secure access to GitHub from the command line. However, doing that now provides good practice, as you repeat the steps for the instance of GitHub used for the IBM Cloud Platform later.

For now, download the GitHub Desktop and run the installer. When the installer finishes, log in to GitHub with your account.

Enter a name and email (this is displayed publicly) for any commits to your repository. Once the application is linked to your account, you might be asked to verify the application connection through your GitHub account online.

GitHub Desktop Login window



The best way to build and ship software. Go to github.com to sign up for an account

Step 1: Creating the Node.js starter app

To start developing your application locally, begin by logging in to the IBM Cloud Platform directly from the command line, as shown in the example. You can specify optional parameters, such as your organization with option `-o` and the space with option `-s`. If you're using a federated account use `--sso`.

```
$ ibmcloud login
```

Type the command as shown in order to download and install the CLI extension used in this tutorial.

```
$ ibmcloud cf install
```

When you log in you might be asked to choose a region. For this exercise, select `us-south` as the region, as that same option is used to build a CD Toolchain later in this tutorial.

Next, set the endpoint (if it isn't set already). Other endpoints are possible, and might be preferable for production use. For now, use the code as shown, if appropriate for your account.

```
$ ibmcloud api cloud.ibm.com
```

Next, create a web application. The `dev` space is a default option for your organization, but you might prefer to create others for isolating different efforts. For example, keeping 'finance' separate from 'development'.

```
$ ibmcloud dev create
```

With that command, you're asked a series of questions. You can go back at many points in the process, so if you feel lost you can start over by deleting the existing directory and creating a new directory. Even when you create your application on the command line, you'll still see the results in your IBM Cloud console.

Note the option for creating a 'Web App'. That's the one you want.

```
================================================================================
Select an application type:

 1. Backend Service / Web App
 2. Mobile App
-------------------------
 0. Exit

================================================================================
? Enter selection number:> 1
```

A number of options are provided, but you want 'Node'. Type '4' and press enter.

```
================================================================================
Select a language:

 1. Go
 2. Java - MicroProfile / Java EE
 3. Java - Spring
 4. Node
 5. Python - Django
 6. Python - Flask
 7. Swift
-------------------------
 0. Return to the previous selection

================================================================================
? Enter selection number:> 4
```

After you make your selection for the programming language and framework, the next selection will have so many options, it might scroll past your wanted service. As you can see in the example, you wish to use a simple Node.js Web App with Express.js. Type '3' and press enter.

```
================================================================================
Select a Starter Kit:

APPSERVICE
-------------------------------------------------------------------------------

 1. Node-RED - A starter to run the Node-RED open-source project on
    IBM Cloud.

 2. Node.js + Cloudant - A web application with Node.js and Cloudant

 3. Node.js Express App - Start building your next Node.js Express
    app on IBM Cloud.


WATSON
-------------------------------------------------------------------------------

 4. Natural Language Understanding Node.js App - Use Watson Natural
    Language Understanding to analyze text to help you understand its
    concepts, entities, keywords, sentiment, and more.

 5. Speech to Text Node.js App - React app using the Watson Speech to
    Text service to transform voice audio into written text.

 6. Text to Speech Node.js App - React app using the Watson Text to
    Speech service to transform text into audio.
 7. Visual Recognition Node.js App - React app using the Watson
    Visual Recognition service to analyze images for scenes, objects, text,
    and other subjects.

------------------------
 0. Return to the previous selection

================================================================================
? Enter selection number:> 3
```

The hardest option for developers everywhere is still required: naming your app. Follow the example and type `webapplication`, then press enter.

```
? Enter a name for your application> webapplication
```

Later, you can add as many services, like data stores or compute functions, as needed or wanted through the web console. However, type 'n' for no when asked if you want to add services now. Also, if you haven't already set a resource group, you may be prompted at this time. You may skip this by typing 'n' at this prompt.

```
Using the resource group Default (default) of your account

? Do you want to select a service to add to this application? [Y/n]> n

```

One way to manage a containerized application is with orchestration software, like Kubernetes, which is a *de facto* standard in development.

Type '4' and press enter to use 'IBM DevOps' for integrating CD within your project lifecycle.

```
================================================================================
Select from the following DevOps toolchain and target runtime environment
options:

 1. IBM DevOps, deploy to Knative-based Kubernetes containers
 2. IBM DevOps, deploy to Helm-based Kubernetes containers
 3. IBM DevOps, deploy to Helm-based Red Hat OpenShift containers
 4. No DevOps, with manual deployment

================================================================================
? Enter selection number:> 4
```

You must choose a region for your automated deployment CD toolchain. Select the option referencing the same region as chosen earlier, '5'.

```
--------------------------------------------------------------------------------
Select a region for your toolchain from the following options:
--------------------------------------------------------------------------------
 1. eu-de (Frankfurt)
 2. eu-gb (London)
 3. jp-tok
 4. us-east (Washington DC)
 5. us-south (Dallas)
--------------------------------------------------------------------------------
 0. Return to the previous selection
--------------------------------------------------------------------------------
? Enter selection number:> 5
```

Generating a new application reminds us that the toolchain used to deploy your app needs some additional configuration. As mentioned earlier, uploading your public key to GitHub (at the CD Toolchain instance on the IBM Cloud Platform), is required to deliver the deployed application by using GitHub.

```
Note: For successful connection to the DevOps toolchain, this machine
must be configured for SSH access to your IBM Cloud GitLab account at
https://git.cloud.ibm.com/profile/keys in order to download the
application code.
```

Further prompts confirm the application and toolchain name that you defined earlier. The example shows how you can alter the host and toolchain names, if you want. The hostname must be unique for the service endpoint of your application, but barring a conflict, you can simply press return when asked for confirmation.

```
The DevOps toolchain for this app will be: webapplication
```

```
? Press [Enter] to accept this, or enter a new value now>


The hostname for this app will be: webapplication
? Press [Enter] to accept this, or enter a new value now>

The app webapplication has been created in IBM Cloud.

DevOps toolchain created at
https://cloud.ibm.com/devops/toolchains/6ffb568a-e48f-4e27-aed0-00ca931dde66?env_id=ibm:yp:us-south
```

If you copy and paste the link that is returned by the `ibmcloud dev create` command, you can also access your CD Toolchain. You can access that from the console later, in case you missed capturing the link. Further information follows, as the process continues creating application entries online, as well as a directory with the sample code.

```
Cloning repository
https://git.cloud.ibm.com/Organization.Name/webapplication...
Cloning into 'webapplication'...
remote: Counting objects: 60, done.
remote: Compressing objects: 100% (54/54), done.
remote: Total 60 (delta 4), reused 0 (delta 0)
Receiving objects: 100% (60/60), 50.04 KiB | 1.52 MiB/s, done.
Resolving deltas: 100% (4/4), done.
OK

The app, webapplication, has been successfully saved into the
current directory.
```

That last statement means that if you view your current directory, a new subdirectory `webapplication` is now visible. This directory holds a scaffold of your new Node.js application. However, while the recipe might be present, the ingredients themselves are still wrapped up in a Docker image and must be combined. Docker is running on your local machine as a consequence of installation, but if you need to restart it do so. If you build your new web application without Docker running it fails, but that's not the only possible error. If you run into trouble, check the resulting error messages, which might have the appropriate link to view result logs in your online portal for your IBM Cloud Platform account.

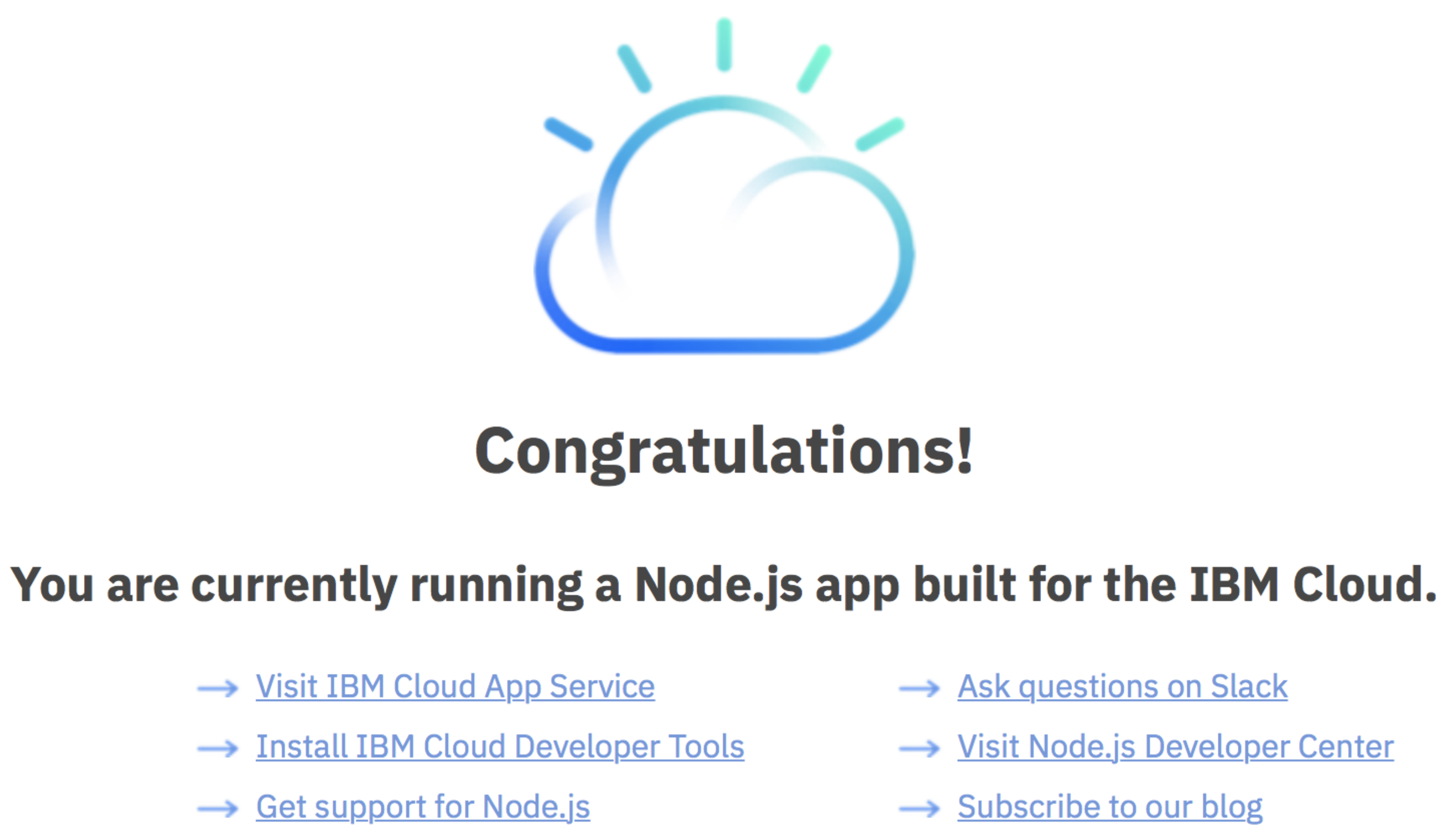
```
$ ibmcloud dev build
```

Now that the app is built, you can run the code locally with the `run` command. When finished, copy and paste the provided URL into your browser's address bar, typically, `http://localhost:3000`.

```
$ ibmcloud dev run
```

Now that the app is created and defined, view your application to confirm it works. If you see the placeholder image as shown in Figure 2, well done! You've created a new Node.js web application and are ready to deploy it to the cloud.

New Node.js Application!


Congratulations!

You are currently running a Node.js app built for the IBM Cloud.

→ Visit IBM Cloud App Service
→ Install IBM Cloud Developer Tools
→ Get support for Node.js
→ Ask questions on Slack
→ Visit Node.js Developer Center
→ Subscribe to our blog

Deploy the app to IBM Cloud Platform with the `deploy` command (as shown in the example).

```
$ ibmcloud dev deploy
```

The URL again is displayed by `ibmcloud dev deploy` based on the regional endpoint and the hostname you specified earlier. You can see links to the logs that are stored in your portal at the IBM Cloud Platform. Go ahead and visit your new web application in the cloud!

Step 2: Creating the Web Gallery app

Let's recall the prerequisites that you needed for developing a Node.js app on IBM Cloud Platform. You already created your IBM Cloud Platform account as well as installed the Developer Tools, which installed Docker. Then, you installed Node.js. The last item listed as a prerequisite for this tutorial was Git, which you dive into now.

We're going to start the specifics of working on the image gallery in Node.js. For now, use GitHub Desktop for this scenario, but you might also use the Git command-line client to complete the same tasks. To get started, clone a starter template for your new web application.

Follow this process:

1. Download the sample here: download. Download the template for your app to your local development environment using your browser. Rather than cloning the sample app from IBM Cloud Platform, use the command in the example to obtain the starter template for the IBM Cloud Object Storage Web Gallery app. After cloning the repo you will find the starter app in the COS-WebGalleryStart directory. Open a Git CMD window and change to a directory where you want to clone Github repo. Once there, use the command shown in the first example of this tutorial to start adding your new files.

   ```
   $ curl images/image-gallery-tutorial.zip -o image-gallery-tutorial.zip
   ```

2. Run the app locally. Open your terminal and change your working directory to the `COS-WebGalleryStart directory`. Note the Node.js dependencies that are listed in the package.json file. Download them into place by using the command shown next.

   ```
   $ npm install
   ```

3. Run the app by using the command shown.

   ```
   $ npm start
   ```

   Open a browser and view your app on the address and port that is output to the console, `http://localhost:3000`.

   Tip: To restart the app locally, kill the node process (Ctrl+C) to stop it, and use `npm start` again. Using `nodemon` instead restarts the app when it detects a change, and saves you time. Install `nodemon` globally like this: `npm install -g nodemon`. Run it from the command line in your app directory by using: `nodemon`, to start your app.

4. Get ready to prepare the app for deployment! Update the application name property value in the `manifest.yml` file from COS-WebGallery, to the name you entered for your app on IBM Cloud Platform and the other information as shown in the example, if necessary. The application `manifest.yml` looks like the following example. You can customize the `package.json` file that is located in the app root directory for your app with the name of your app and your name as the author.

```
applications:
- path: .
  memory: 256M
  instances: 1
  domain: us-south.cf.appdomain.cloud
  name: webapplication
  host: webapplication
  disk_quota: 1024M
  random-route: true
```

Tip: Now is the point where you might need to set up SSH keys to interactively push code to your remote origin. If you set a passphrase for your SSH key, you're required to enter this code each time you push your changes to the remote origin for your repository.

5. Remove and replace the contents of your `webapplication` directory with the contents of the directory you modified, `COS-WebGalleryStart`. Using your finely tuned Git skills, add the files that were deleted and added to the repository with either the CLI or GitHub Desktop. Then, push the changes to the repository origin. In the future, you can make changes to your cloud-based web application just by pushing changes to Git. The CD toolchain will automatically restart the server process after cloning your changes and stashing them on the server.

In essence, you've recoded your application, so repeat the build process. But this time use the new Image Gallery code.

Deploy the app to IBM Cloud Platform.

To get the starter app with your changes to IBM Cloud Platform, deploy it using the Developer Tools by repeating the same steps that you performed earlier.

1. If you haven't already, or if you restarted or logged out, log in to IBM Cloud Platform by using the `login` command.

```
$ ibmcloud login
```

2. Set the API Endpoint for your region by using the `api` command.

```
$ ibmcloud api cloud.ibm.com
```

3. Build the app for delivery that application with the build command (as in the example).

```
$ ibmcloud dev build
```

   a. Let's go ahead and test the application locally. This allows you to run the same code locally with the `run` command.

```
$ ibmcloud dev run
```

4. Deploy the app to IBM Cloud Platform with the `deploy` command.

```
$ ibmcloud dev deploy
```

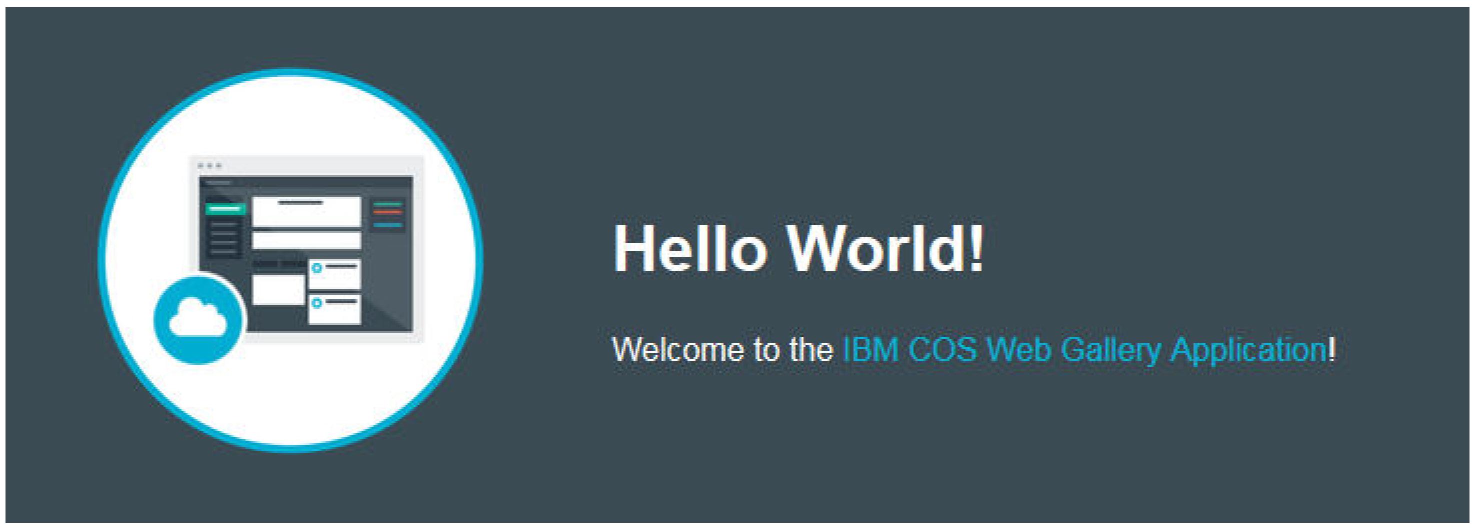
The code shows the sequence of commands that are used in this example to build, test, and deploy the initial web application.

```
$ ibmcloud login --sso
ibmcloud api cloud.ibm.com
ibmcloud target --cf
ibmcloud dev enable
ibmcloud dev build
ibmcloud dev run
ibmcloud dev deploy
```

When the process finishes, the IBM Cloud Platform reports that the app was uploaded, successfully deployed, and started. If you're also logged in to the IBM Cloud Platform web console, you're notified there also of the status of your app. But, most importantly, you can verify that the app was deployed by visiting the app URL reported by IBM Cloud Platform with a browser, or from the web console by clicking View App button.

Test the app. The visible change from the default app template that was deployed at creation to the starter app shown in the following proved that deploying the app to IBM Cloud Platform was successful.

Results of viewing your deployed app.

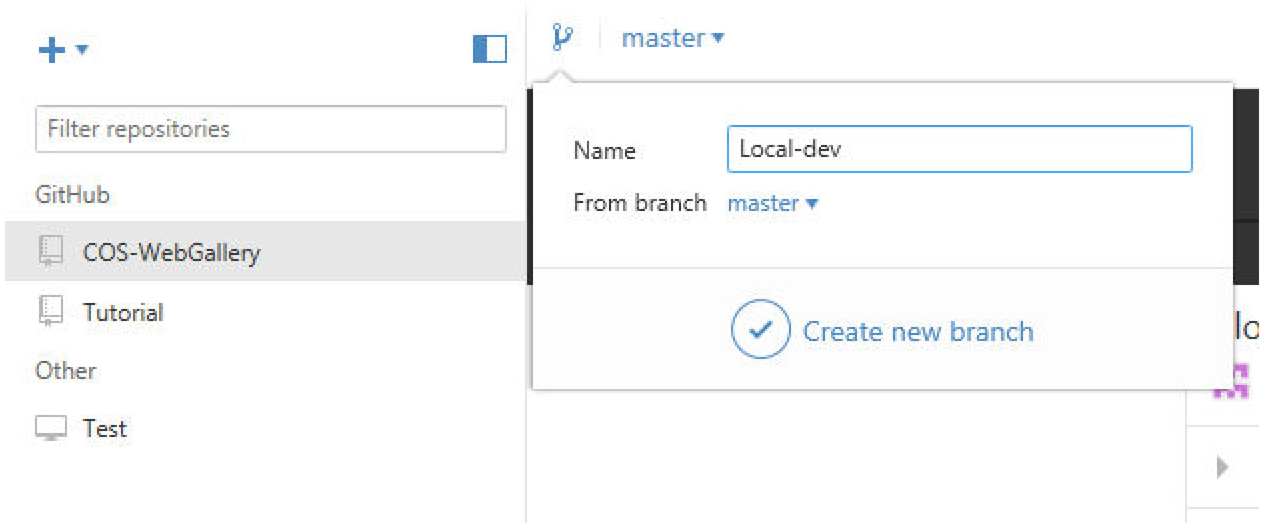


Create a Git branch

Now, you need to create a branch for the local development environment to use for your IBM Cloud Platform Delivery Pipeline Build Stage:

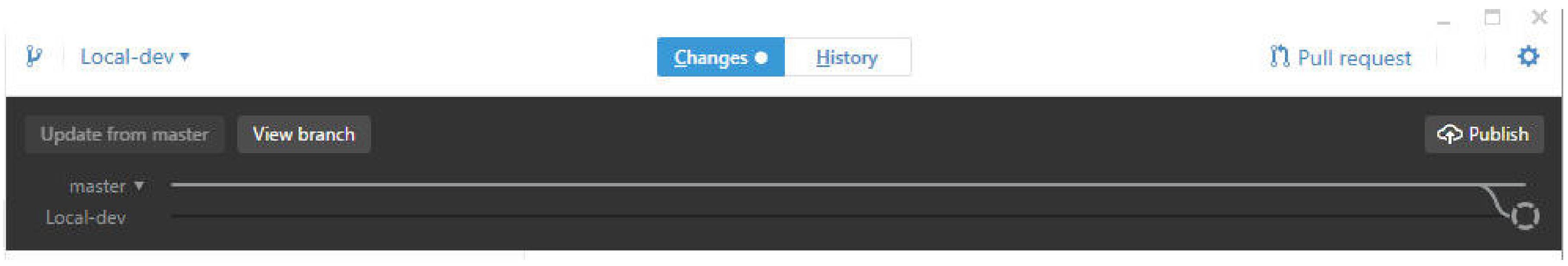
1. If using GitHub Desktop, click the branch icon; you're prompted to enter a name for the branch. This example uses `local-dev` as the name.

Use GitHub Desktop to create a local dev branch



2. After you create the branch, GitHub compares the local files on the Local-dev branch with the files in the repository on the default branch and reports No local changes. You can now click Publish to add the branch you created on your local repo to your GitHub repo (as shown in Figure 5).

Publish your git branch to your repo's remote origin



Now that the Local-dev branch is published to the GitHub repo in your toolchain, the build stage of your IBM Cloud Platform Delivery Pipeline will be triggered followed by the deployment stage anytime you push a commit to it. Deploying the app from the CLI is not necessary, as the deployment has been integrated directly into your workflow.

Setting up your storage credentials

You need to configure Object Storage credentials for your web application, as well as a 'bucket' where it will store and retrieve images. The API key that you will create will need Object Storage HMAC credentials, as defined by your Service Credentials. You might recognize the terms `access_key_id` and `secret_access_key` as you might have an AWS account, and use a credentials file that already has `aws_access_key_id` and `aws_secret_access_key` entries.

After you have completed creating an API key, downloaded, and then copied the HMAC credentials, complete the following steps:

1. On the local development environment, place the credentials in the Windows path `%USERPROFILE%\\.aws\\credentials`. For Mac/Linux users, the credentials should go into `~/.aws/credentials)`. The example shows the contents of a typical credentials file.

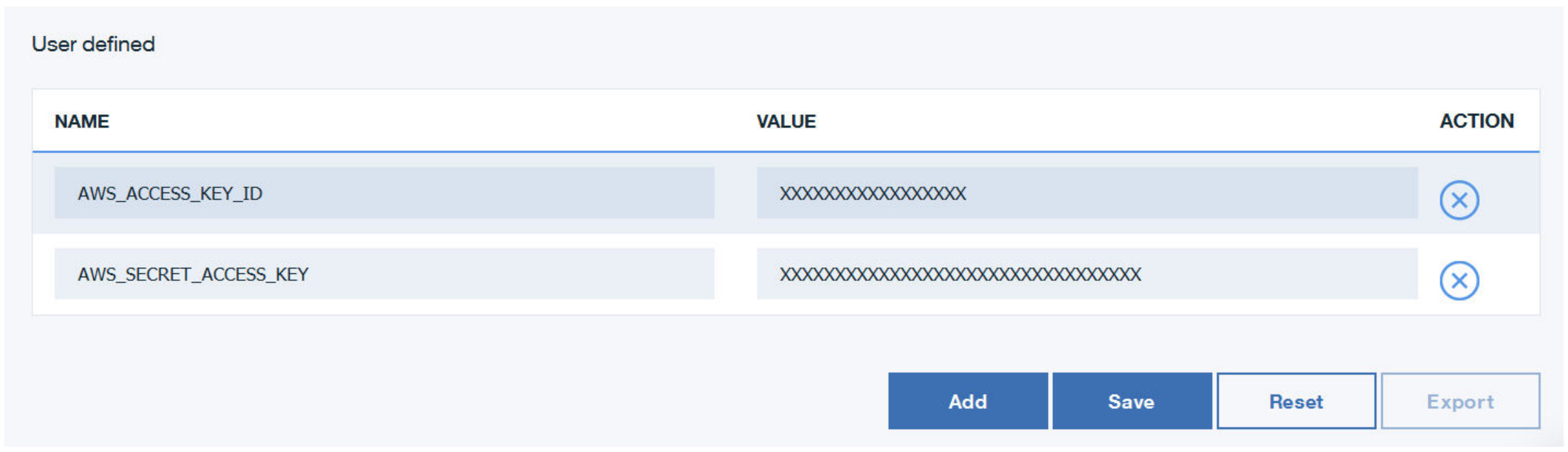
```
[default]
aws_access_key_id = {access_key_id}
aws_secret_access_key = {secret_access_key}
```

2. In the web page for the application you created by using the CLI command on the IBM Cloud Platform, define your required credentials as environment variables per development best practices by logging in to IBM Cloud Platform, and select your app, `webapplication`. From the tabs, click **Runtime**.

3. In the Runtime window, click Environment variables at the beginning of the page and scroll to the User-defined section, which allows you to add the variables.

4. Add two variables: one with the value of your `access_key_id`, using `AWS_ACCESS_KEY_ID` as the name of the key, and another with the value of your secret access key, named `AWS_SECRET_ACCESS_KEY`. These variables and their respective values are what the app uses to authenticate to the Object Storage instance when running on IBM Cloud Platform (see Figure 6). When you finish with the entries, click Save, and IBM Cloud Platform will automatically restart the app for you.

Runtime Environment Variables defined for your app



User defined

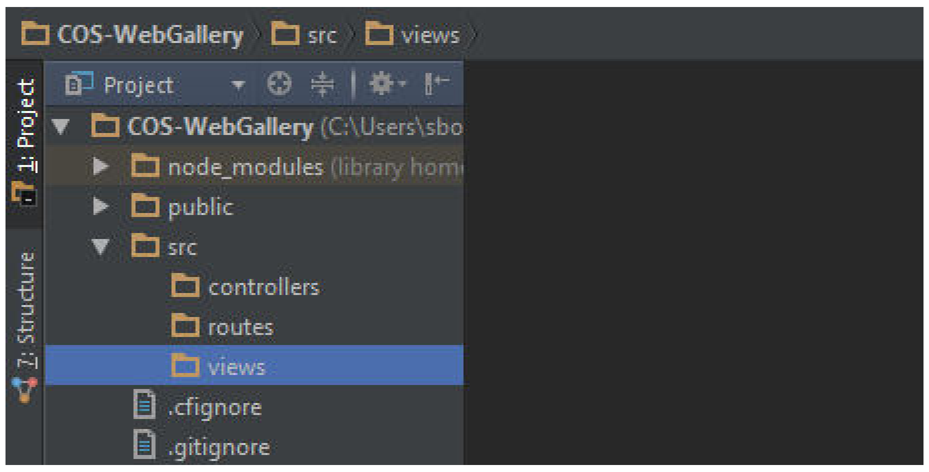
NAME	VALUE	ACTION
AWS_ACCESS_KEY_ID	XXXXXXXXXXXXXXXXXX	⊗
AWS_SECRET_ACCESS_KEY	XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX	⊗

Next, over at the Object Storage Portal for your service instance, add a bucket to contain your images. This scenario uses the bucket that is named `web-images`.

Step 3: Customize your Node.js IBM Cloud Object Storage Image Gallery web Application

Because this example uses an MVC architecture, adjusting the directory structure within your project to reflect this architecture is a convenience as well as a best practice. The directory structure has a views directory to contain the EJS view templates, a routes directory to contain the express routes, and a `controllers` directory as the place to put the controller logic. Place these items under a parent source directory named `src` (see Figure 7).

Source code structure for your app



```
▼ COS-WebGallery (C:\Users\sbo
  ▶ node_modules (library hom
  ▶ public
  ▼ src
      controllers
      routes
      views
    .cfignore
    .gitignore
```

Tip: The repo that you cloned earlier contain a directory that is named `COS-WebGalleryEnd`. Viewing the source code of the completed application in your preferred editor might be helpful as you follow the next steps. This is the version of your `webapplication` that is committed and deployed to IBM Cloud Platform when you complete this tutorial.

Designing the app

These are the two main tasks that a user should be able to do with the simple image gallery web application:

- Upload images from a web browser to the Object Storage bucket.
- View the images in the Object Storage bucket in a web browser.

The next steps focus on how to accomplish these two demonstration functions rather than building a fully developed, production-grade app. Deploying this tutorial and leaving it exposed and running means that anyone who finds the app can perform the same actions: upload files to your IBM Cloud Object Storage bucket and view any JPEG images already there in their browser.

Developing the app

In the `package.json` file, inside the scripts object, you see how "start" is defined. This file is what IBM Cloud Platform uses to tell node to run app.js each time the app starts. Also, use it when testing the app locally. Look at the main application file, which is called `app.js`. This is the code that you told Node.js to process first when you start your app with the `npm start` command (or `nodemon`).

```
{
    "scripts": {
      "start": "node app.js"
    }
}
```

Our `app.js` file uses node to load modules that are needed to get started. The Express framework creates the app as a singleton simply called `app`. The example ends (leaving out most of the code for now) telling the app to listen on the port that is assigned and an environment property, or 3000 by default. When successfully starting at the start, it prints a message with the server URL to the console.

Node

```
var express = require('express');
var cfenv = require('cfenv');
var bodyParser = require('body-parser');
var app = express();
//...

// start server on the specified port and binding host
var port = process.env.PORT || 3000;
app.listen(port, function() {
    console.log("To view your app, open this link in your browser: http://localhost:" + port);
});
//...
```

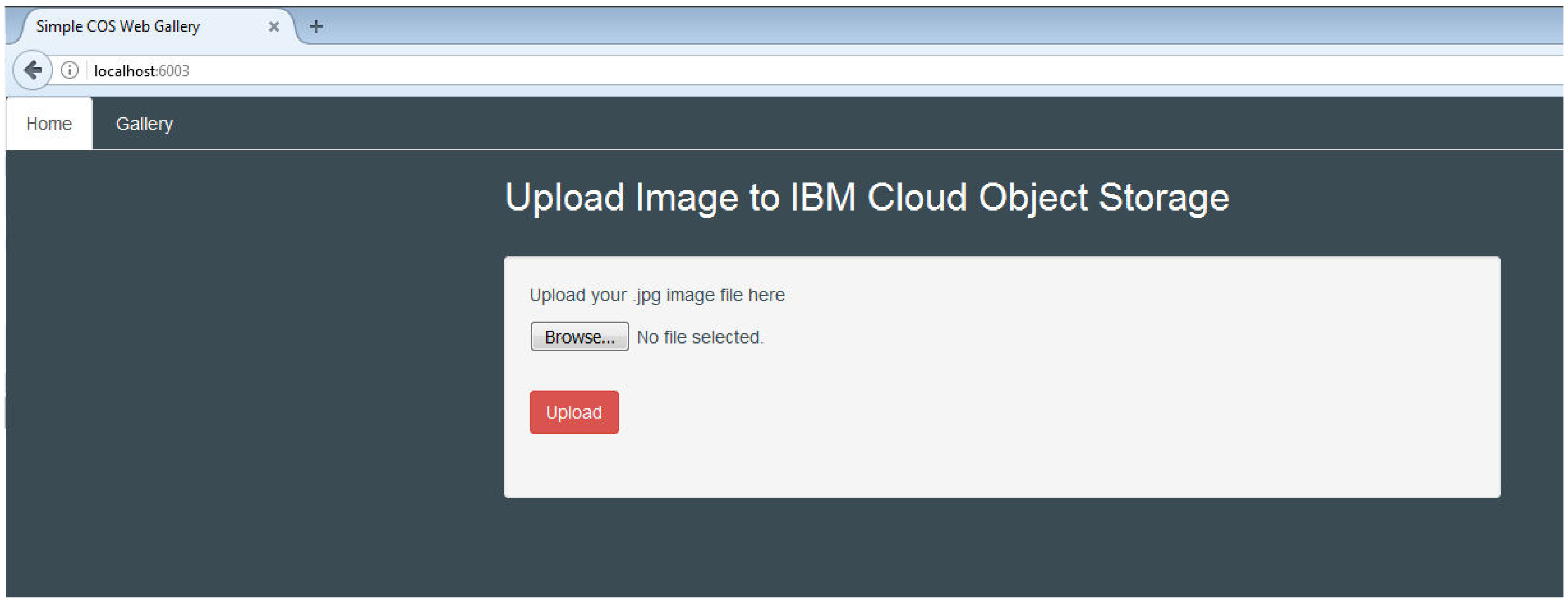
Let's see how to define a path and views. The first line of code tells the Express framework to use the public directory to serve your static files, which include any static images and style sheets you use. The lines that follow tell the app where to find the templates for your views in the `src/views` directory, and set your view engine to be EJS. In addition, the framework uses the body-parser middleware to expose incoming request data to the app as JSON. In the closing lines of the example, the express app responds to all incoming GET requests to your app URL by rendering the `index.ejs` view template.

Node

```
//...
// serve the files out of ./public as your main files
app.use(express.static('public'));
app.set('views', './src/views');
app.set('view engine', 'ejs');
app.use(bodyParser.json());

var title = 'COS Image Gallery Web Application';
// Serve index.ejs
app.get('/', function (req, res) {
  res.render('index', {status: '', title: title});
});

//...
```

The following figure shows what the index view template when rendered and sent to the browser. If you are using , `nodemon` you might have noticed that your browser refreshed when you saved your changes.

Your updated web app by using templates and views for displays



Our view templates share HTML code between the `<head>...</head>`; tags, so you placed it into a separate include template. This template (`head-inc.ejs`) contains a scriptlet (a binding for a JavaScript variable) for the page title on line 1. The `title` variable is set in `app.js`, and passed in as data for your view template in the line below that. Otherwise, you are simply using some CDN addresses to pull in `Bootstrap CSS`, `Bootstrap JavaScript`, and `JQuery`. Finally, add a custom static `styles.css` file from your `pubic/stylesheets` directory.

```
<title><%=title%></title>
<meta charset="utf-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=1">

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css"
      integrity="sha384-BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u"
      crossorigin="anonymous">
<script src="https://code.jquery.com/jquery-3.1.1.min.js"
        integrity="sha256-hVVnYaiADRTO2PzUGmuLJr8BLUSjGIZsDYGmIJLv2b8="
        crossorigin="anonymous">
</script>
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"
        integrity="sha384-Tc5IQib027qvyjSMfHjOMaLkfuWVxZxUPnCJA7l2mCWNIpG9mGCD8wGNIcPD7Txa"
        crossorigin="anonymous">
</script>

<link rel="stylesheet" href="stylesheets/style.css">
```

The body of the index view contains your bootstrap styled navigation tabs, and your upload form in a basic layout that is provided by the CSS styles included with bootstrap.

Consider these two specifications for your app:

- Set your form method to `POST` and the form-data encoding type as multipart/form-data on line 24. For the form action, send the data from your form to the app to the app route "/". Later, do extra work in your router logic to handle `POST` requests to that route.
- Display feedback about the status of the attempted file upload to the user. This feedback is passed to your view in a variable named "status", and is displayed after the upload form.

```
<!DOCTYPE html>
<html>

<head>
    <%- include('head-inc'); %>
</head>

<body>
<ul class="nav nav-tabs">
    <li role="presentation" class="active"><a href="/">Home</a></li>
    <li role="presentation"><a href="/gallery">Gallery</a></li>
</ul>
<div class="container">
    <h2>Upload Image to IBM Cloud Object Storage</h2>
    <div class="row">
        <div class="col-md-12">
            <div class="container" style="margin-top: 20px;">
```

```
                <div class="row">

                    <div class="col-lg-8 col-md-8 well">

                        <p class="wellText">Upload your JPG image file here</p>

                        <form method="post" enctype="multipart/form-data" action="/">
                            <p><input class="wellText" type="file" size="100px" name="img-file" /></p>
                            <br/>
                            <p><input class="btn btn-danger" type="submit" value="Upload" /></p>
                        </form>

                        <br/>
                        <span class="notice"><%=status%></span>
                    </div>
                </div>
            </div>
        </div>
    </div>
</div>
</body>

</html>
```

Let's take a moment to return to `app.js`. The example sets up Express routes to handle extra requests that are made to your app. The code for these routing methods are in two files under the `./src/routes` directory in your project:

- `imageUploadRoutes.js`: This file handles what happens when the user selects an image and clicks Upload.
- `galleryRoutes.js`: This file handles requests when the user clicks the Gallery tab to request the `imageGallery` view.

Node

```
//...
var imageUploadRoutes = require('./src/routes/imageUploadRoutes')(title);
var galleryRouter = require('./src/routes/galleryRoutes')(title);

app.use('/gallery', galleryRouter);
app.use('/', imageUploadRoutes);

//...
```

Image upload

See the code from `imageUploadRoutes.js`. You must create an instance of a new express router and name it `imageUploadRoutes` at the start. Later, create a function that returns `imageUploadRoutes`, and assign it to a variable called `router`. When completed, the function must be exported as a module to make it accessible to the framework and your main code in `app.js`. Separating your routing logic from the upload logic requires a controller file named `galleryController.js`. Because that logic is dedicated to processing the incoming request and providing the appropriate response, put that logic in that function and save it in the `./src/controllers` directory.

The instance of the Router from the Express framework is where your `imageUploadRoutes` is designed to route requests for the root app route ("/") when the HTTP `POST` method is used. Inside the `post` method of your `imageUploadRoutes`, use middleware from the `multer` and `multer-s3` modules that is exposed by the `galleryController` as `upload`. The middleware takes the data and file from your upload form `POST`, processes it, and runs a callback function. In the callback function, check that you get an HTTP status code of `200`, and that you had at least one file in your request object to upload. Based on those conditions, set the feedback in your `status` variable and render the index view template with the new status.

Node

```
var express = require('express');
var imageUploadRoutes = express.Router();
var status = '';

var router = function(title) {

    var galleryController =
        require('../controllers/galleryController')(title);

    imageUploadRoutes.route('/')
     .post(
      galleryController.upload.array('img-file', 1), function (req, res, next) {
                if (res.statusCode === 200 && req.files.length > 0) {
                    status = 'uploaded file successfully';
                }
                else {
                    status = 'upload failed';
                }
                res.render('index', {status: status, title: title});
            });

    return imageUploadRoutes;
};

module.exports = router;
```

In comparison, the code for the `galleryRouter` is a model of simplicity. Follow the same pattern that you did with `imageUploadRouter` and require `galleryController` on the first line of the function, then set up your route. The main difference is that you are routing HTTP `GET` requests rather than `POST`, and sending all the output in the response from `getGalleryImages`, which is exposed by the `galleryController` on the last line of the example.

Node

```
var express = require('express');
var galleryRouter = express.Router();

var router = function(title) {

    var galleryController =
        require('../controllers/galleryController')(title);

    galleryRouter.route('/')
        .get(galleryController.getGalleryImages);

    return galleryRouter;
};
module.exports = router;
```

Next, look at the controller for the gallery.

Note how you set up the `multer` upload, which truncates some code you ignore for now. You require modules `ibm-cos-sdk`, `multer`, and `multer-s3`. The code shows how to configure an S3 object that points to an Object Storage server endpoint. You are statically setting values such as the endpoint address, region, and bucket for simplicity, but they might easily be referenced from an environment variable or JSON configuration file.

Define `upload` in the `imageUploadRouter` by creating a new `multer` instance with `storage` as its only property. This property tells the `multer` where to send the file from your `multipart/form-data`. Since the IBM Cloud Platform uses an implementation of the S3 API, set storage to be an `s3-multer` object. This `s3-multer` object contains an `s3` property that is assigned to your `s3` object. There is also a `bucket` property that is assigned to the `myBucket` variable, which in turn is assigned a value of `web-images`. The `s3-multer` object now has all the data necessary to upload files to your Object Storage bucket when it receives data from the upload form. The name (or key) of the uploaded object is the original file name.

Tip: Use a time stamp as part of the file name to maintain file name uniqueness.

Node

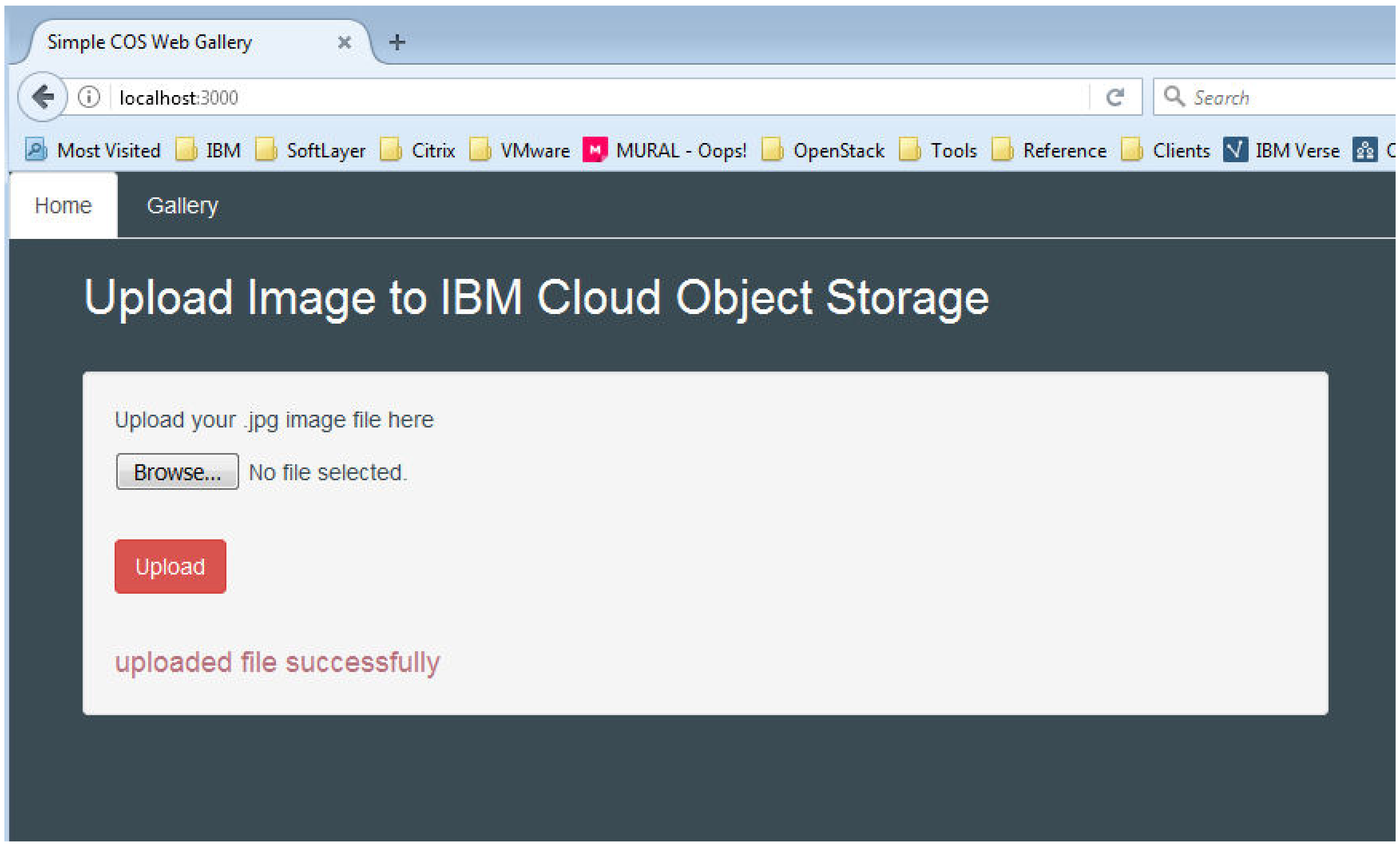
```
var galleryController = function(title) {

    var aws = require('ibm-cos-sdk');
    var multer = require('multer');
    var multerS3 = require('multer-s3');

    var ep = new aws.Endpoint('s3.us-south.cloud-object-storage.appdomain.cloud');
    var s3 = new aws.S3({endpoint: ep, region: 'us-south-1'});
    var myBucket = 'web-images';

    var upload = multer({
        storage: multerS3({
            s3: s3,
            bucket: myBucket,
            acl: 'public-read',
            metadata: function (req, file, cb) {
                cb(null, {fieldName: file.fieldname});
            },
            key: function (req, file, cb) {
                console.log(file);
                cb(null, file.originalname);
            }
        })
    });

    var getGalleryImages = function (req, res) { /* ... shown below ... */ };

    return {
        getGalleryImages: getGalleryImages,
        upload: upload
    };
};

module.exports = galleryController;
```

For local testing, a helpful task is to print the file object to the console, `console.log(file)`. Perform a local test of the upload form and show the output from the console log of the file.

```
{ fieldname: 'img-file',
originalname: 'Chrysanthemum.jpg',
encoding: '7bit',
mimetype: 'image/jpeg' }
```

The feedback from your callback declares the application has "uploaded file successfully" when tested.

Success!



Upload Image to IBM Cloud Object Storage

Upload your .jpg image file here

No file selected.

uploaded file successfully

Image retrieval and display

Remember back in `app.js`, the line of code `app.use('/gallery', galleryRouter);` tells the express framework to use that router when the `/gallery` route is requested. That router uses `galleryController.js`, define the `getGalleryImages` function, the signature of which you have seen previously. Using the same `s3` object that you set up for your image upload function, call the function that is named `listObjectsV2`. This function returns the index data defining each of the objects in your bucket. To display images within HTML, you need an image URL for each JPEG image in your `web-images` bucket to display in your view template. The closure with the data object returned by `listObjectsV2` contains metadata about each object in your bucket.

The code loops through the `bucketContents` searching for any object key ending in ".jpg," and create a parameter to pass to the S3 `getSignedUrl` function. This function returns a signed URL for any object when you provide the object’s bucket name and key. In the callback function, save each URL in an array, and pass it to the HTTP Server response method `res.render` as the value to a property named `imageUrls`.

Node

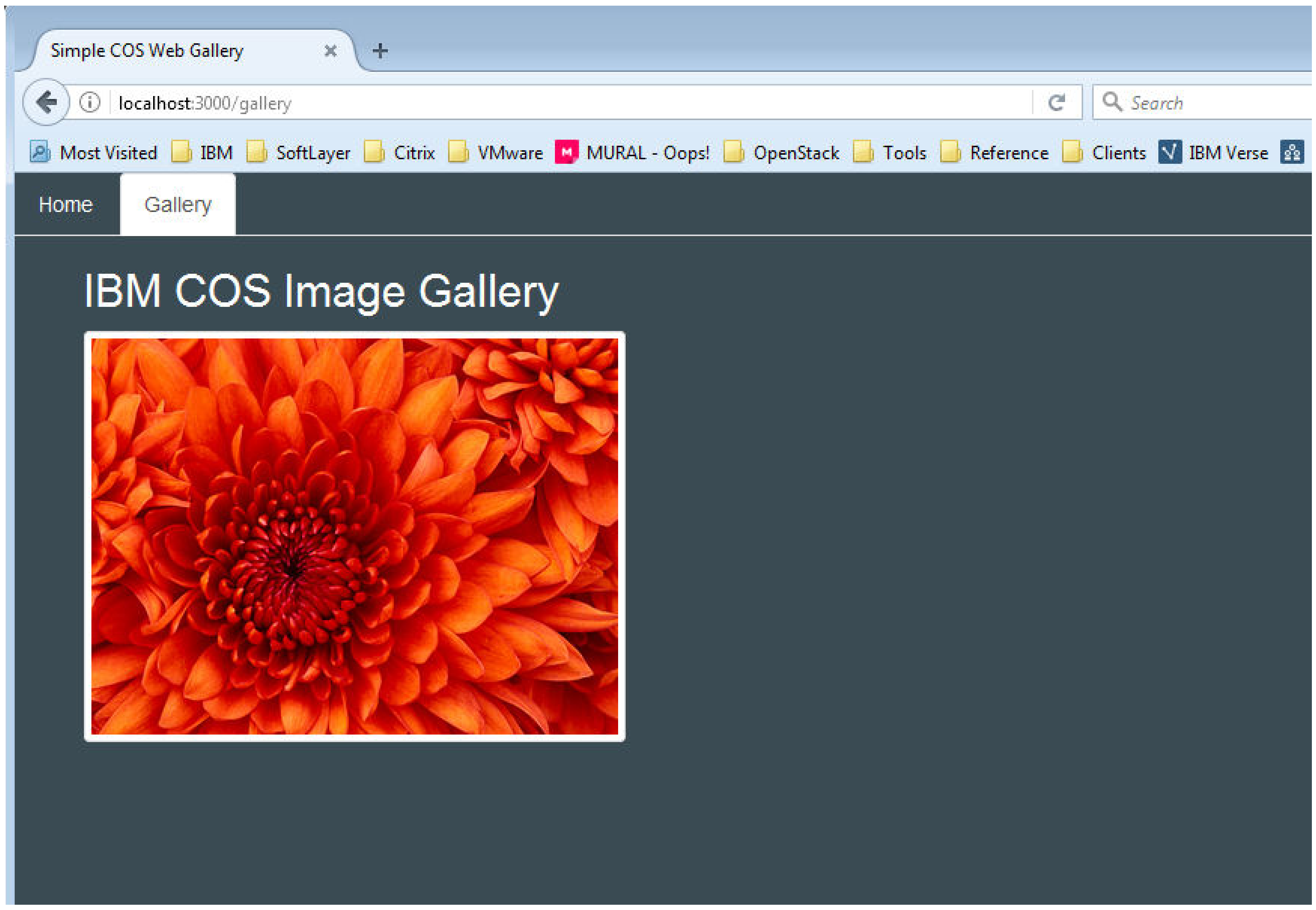
```
//...

    var getGalleryImages = function (req, res) {
        var params = {Bucket: myBucket};
        var imageUrlList = [];

        s3.listObjectsV2(params, function (err, data) {
            if (data) {
                var bucketContents = data.Contents;
                for (var i = 0; i < bucketContents.length; i++) {
                    if (bucketContents[i].Key.search(/.jpg/i) > -1) {
                        var urlParams = {Bucket: myBucket, Key: bucketContents[i].Key};
                        s3.getSignedUrl('getObject', urlParams, function (err, url) {
                            imageUrlList.push(url);
                        });
                    }
                }
            }
            res.render('galleryView', {
                title: title,
                imageUrls: imageUrlList
            });
        });
    };

//...
```

The last code example shows the body of the `galleryView` template with the code that is needed to display your images. Get the `imageUrls` array from the `res.render()` method and iterate over a pair of nested `<div>...</div>` tags. Each sends a `GET` request for the image when the `/gallery` route is requested.

```
<!DOCTYPE html>
<html>

<head>
    <%- include('head-inc'); %>
</head>

<body>
    <ul class="nav nav-tabs">
        <li role="presentation"><a href="/">Home</a></li>
        <li role="presentation" class="active"><a href="/gallery">Gallery</a></li>
    </ul>
    <div class="container">
        <h2>IBM COS Image Gallery</h2>

        <div class="row">
            <% for (var i=0; i < imageUrls.length; i++) { %>
                <div class="col-md-4">
                    <div class="thumbnail">
                            <img src="<%=imageUrls[i]%>" alt="Lights" style="width:100%">
                    </div>
                </div>
            <% } %>
        </div>
    </div>
</body>

</html>
```

Test the app locally from `http://localhost:3000/gallery` and see your image.

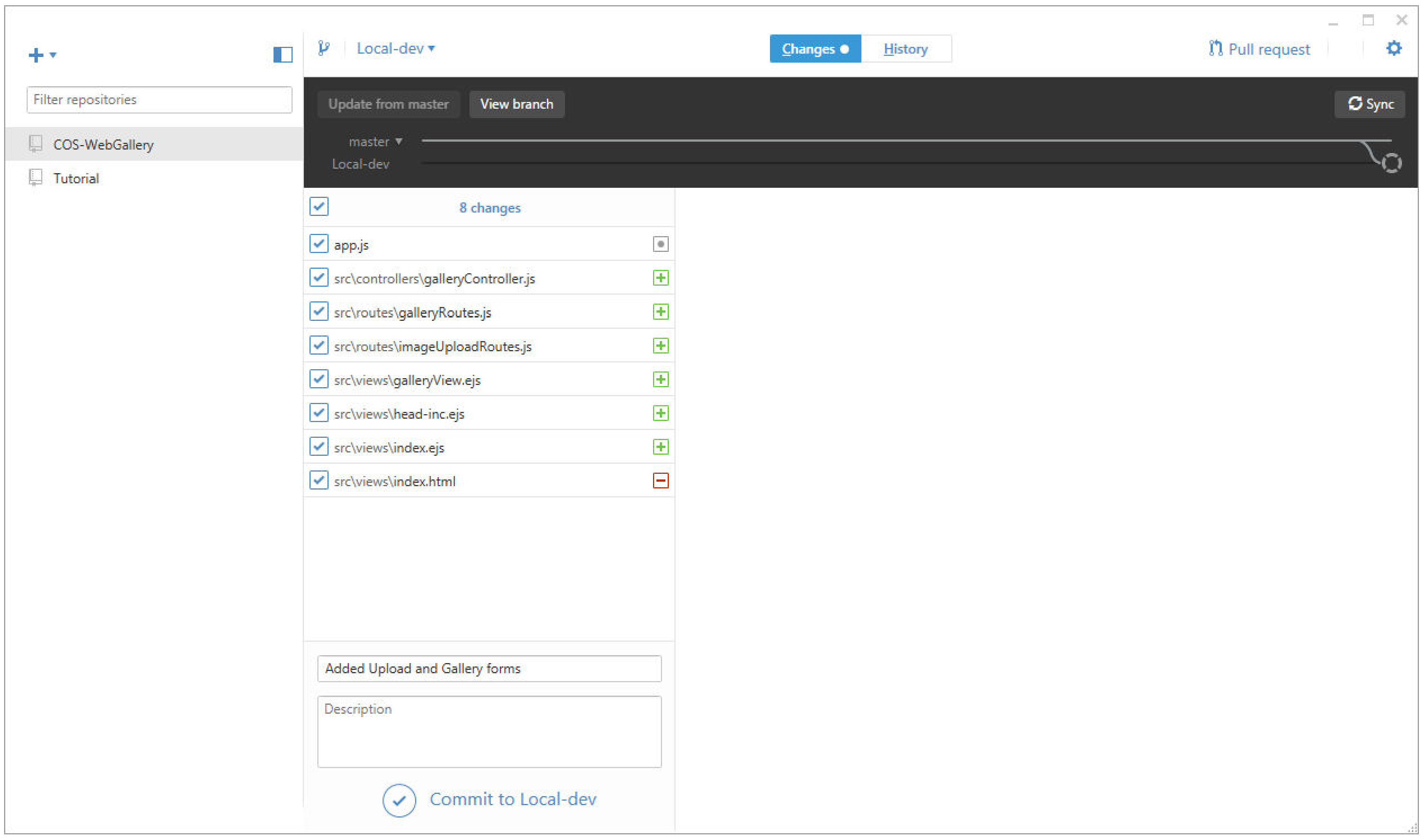
Images uploaded to the bucket are on display



Committing to Git

Now that the basic features of the app are working, commit your code to your local repo, and then push it to GitHub. Using GitHub Desktop, click Changes (see Figure 11), type a summary of the changes in the Summary field, and then click Commit to Local-dev.

Changes ready for commit in Git

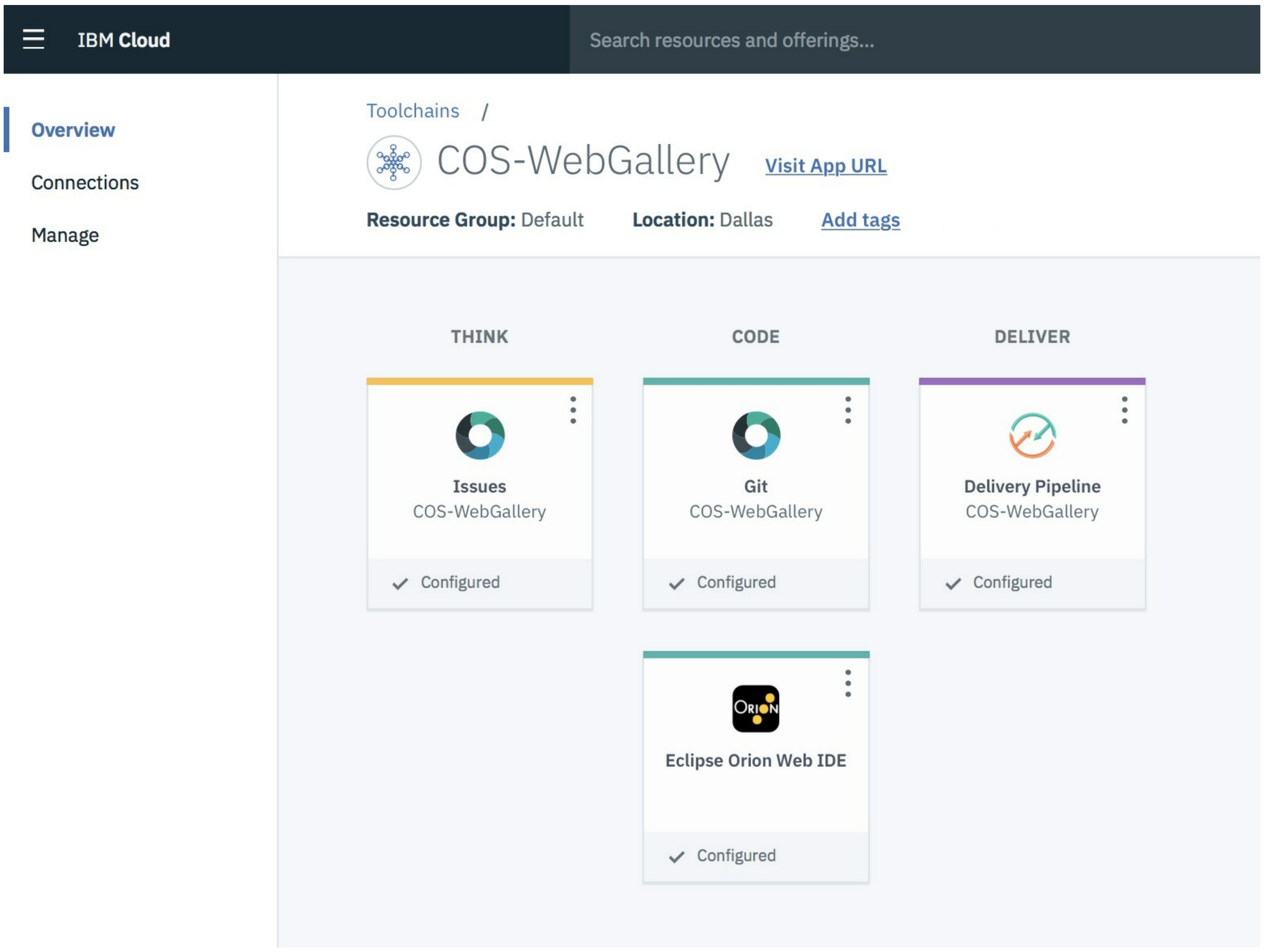


app.js
src\controllers\galleryController.js
src\routes\galleryRoutes.js
src\routes\imageUploadRoutes.js
src\views\galleryView.ejs
src\views\head-inc.ejs
src\views\index.ejs
src\views\index.html

Added Upload and Gallery forms

When you click **sync**, your commit is sent to the remote `local-dev` branch. This action starts the Build and Deploy Stages in your Delivery Pipeline.

CD Delivery Pipeline



Next Steps

You went from beginning to end and built a basic web application image gallery by using the IBM Cloud Platform. Each of the concepts you've covered in this basic introduction can be explored further at [IBM Cloud Object Storage](#).

Good luck!

Provisioning storage

Choosing a plan and creating an instance

Getting data into your instance of IBM Cloud® Object Storage requires just a few steps before you provision your new storage.

About IBM Cloud Object Storage plans

The highest level of organization in IBM Cloud Object Storage is a service instance. Each instance can hold many buckets, and each bucket can hold virtually any number of objects (files). There are four types of Object Storage service instances:

IBM Public Cloud:

- **Standard plan** instances are the most common and are recommended for most workloads.
  - The **Free Tier** allows you to evaluate the Cloud Object Storage service at no cost. You can seamlessly scale up for production use. This includes 5GB of free monthly usage for up to 12 months from the sign-up date. After 12 months or if you exceed the Free Tier limits, you will be billed at standard pay-as-you-go rates.
- **One Rate plan** instances should be used for workloads that involve large volumes of outbound bandwidth (data transferred on public networks outside of IBM Cloud) relative to the amount of total storage capacity.

IBM Cloud Satellite:

- **Satellite** instances are run on hardware outside of IBM Cloud and are typically used for edge computing or for strict data sovereignty requirements

Creating an account

Before you create a new IBM Cloud Object Storage storage instance, it's necessary to create a customer account first.

1. Go to cloud.ibm.com and click **Create a Free Account**.
2. Complete the form with your email address, name, region, and phone number. Choose a password.
3. Follow the link provided by the confirmation email, and follow the links to log in to the IBM Cloud® Platform.

Now that you have a platform account, you can create a new Object Storage service instance.

Creating a service instance

1. Log in to the console.
2. Navigate to the catalog, by clicking **Catalog** in the top navigation bar.
3. In the left menu, Click the **Storage** category. Click the **Object Storage** tile.
4. Give the service instance a name and choose a plan.
5. Click **Create** and you are redirected to your new instance.

It is also possible to manage resources using the IBM Cloud® Platform CLI:

```
ibmcloud resource service-instance-create <instance-name> cloud-object-storage <plan> global
```

Deleting a service instance

When a service instance is deleted, the data is not deleted immediately. Instead, it is scheduled for reclamation (by default this is set to take 7 days), after which the data is irreversibly destroyed, and the bucket names will be made available for reuse. It is also possible to restore a deleted resource that has not yet been reclaimed.

It is possible to check the status of a reclamation, as well as force or cancel a scheduled reclamation using the the IBM Cloud® Platform CLI.

Important: It is impossible to delete a Service Instance if there is a bucket with an active Immutable Object Storage policy or legal hold on any objects. The policy will need to expire before the data can be deleted. It isn't possible to delete a Service Instance if there is a permanent retention policy in place.

Note: Currently, the reclamation can be scheduled only for instances of the IBM Cloud Object Storage **Standard** and **One Rate** plans. The **Lite** plan is not eligible to participate.

Choosing a One Rate plan

The One Rate plan offers a predictable cost of ownership with an all-inclusive flat monthly charge ($/GB/month) that includes capacity, and built-in allowances for outbound bandwidth and operational requests. The One Rate plan is best suited for active workloads with large amounts of outbound bandwidth relative to storage capacity.

The built-in allowances for outbound bandwidth and operational requests (Class A, Class B) depend on the monthly stored capacity. There is no data retrieval charge. The One Rate plan has four pricing regions: North America, Europe, South America, and Asia Pacific. Furthermore, the plan aggregates billing metrics (storage capacity, outbound bandwidth and operational requests) across multiple instances within the One Rate pricing region for determining the allowances (the higher the aggregated storage capacity within a region, the higher the allowances for outbound bandwidth and operational requests for that region).

Why use a One Rate plan?

- Predictable and lower monthly TCO (total cost of ownership) for workloads with high levels of outbound bandwidth to capacity ratios (>20%).
- One Rate plans provide account-level billing that aggregates storage capacity across service instances by region.
- A flat capacity rate with built-in allowances for data access and egress offers a more predictable cost regardless of fluctuating usage patterns.

Terminology

Egress: The measure of outbound bandwidth (GB) read over the public endpoints.

Who should use a One Rate plan?

One Rate plan instances are more expensive when it comes to storage capacity costs, but much less expensive when taking into account egress charges. You should consider using a One Rate instance if:

1. You are a large enterprise or ISV, and most of the data being stored in Object Storage is constantly being read over the public endpoints.
2. You are reading large files from outside of IBM Cloud - for example in post-production film editing, satellite imaging, or music production.

Note: Most workloads, such as for backups/long-term storage, data analysis using IBM Cloud resources, or for small files (such as PNGs for websites) are better served by a Standard plan. One Rate plans are generally best for workloads where more 20% of the total storage is consistently read over the public endpoints each month.

Getting started with One Rate plans

One Rate instances are available in Regional and Single Data Center locations, but are not available in Cross Region locations. There are four pricing tiers based on location:

- North America: `us-south`, `us-east`, `ca-tor`, `mon01`, `sjc04`
- Europe: `eu-de`, `eu-gb`, `eu-es`, `ams03`, `mil01`, `par01`
- Asia: `au-syd`, `jp-osa`, `jp-tok`, `che01`, `sng01`
- South America: `br-sao`

All buckets in a One Rate plan instance **must** use a new `active` storage class specific to One Rate instances.

One Rate plan instances are aggregated and billed at the IBM Cloud account level based on average end-of-month usage. For detailed information and current pricing, please review the detailed cost tables .

Unlike Standard plan instances, One Rate instances provide allowances for Class A and B request charges as well as egress charges. The thresholds for the allowances are dependant on total storage capacity.

Important: It is **not** possible to convert an instance created under a One Rate plan to a Standard plan, or vice-versa.

How allowances are calculated

One Rate plans use an all-inclusive flat monthly rate which includes capacity, operational requests, and outbound bandwidth. The built-in allowances for outbound bandwidth are determined by the total capacity.

> Total Monthly Cost = Capacity Cost + API Cost (if # of API > allowance) + Bandwidth cost (if Bandwidth > allowance)

- Class A allowance: Number of Class A Requests < 100 x Storage (GB)
- Class B allowance: Number of Class B Requests < 1000 x Storage (GB)

- Bandwidth allowance: Bandwidth (GB) < Storage (GB)

Note: Archive is supported but Restore charges are **not** included in the One Rate allowances

How to provision a One Rate instance

A One Rate instance is specified at the point of provisioning, similar to a Lite or Satellite instance.

1. Log in to the console.
2. Navigate to the catalog, by clicking **Catalog** in the navigation bar.
3. Look for the **Object Storage** tile in the storage section and select it.
4. Select **IBM Cloud** from the "Choose an Infrastructure" section.
5. Select **One Rate** from the plans.
6. Choose a name, resource group, and any desired tags.
7. Click **Create** and you're automatically redirected to your new instance.

Special provisioning codes

All buckets created in a One Rate plan must use a specific provisioning code (also known as a storage class or location constraint).

Location	Location Constraint
`us-south`	`us-south-onerate_active`
`us-east`	`us-east-onerate_active`
`ca-tor`	`ca-tor-onerate_active`
`mon01`	`mon01-onerate_active`
`sjc04`	`sjc04-onerate_active`

Location constraint - North America

Location	Location Constraint
`eu-de`	`eu-de-onerate_active`
`eu-gb`	`eu-de-onerate_active`
`eu-es`	`eu-es-onerate_active`
`ams03`	`ams03-onerate_active`
`mil01`	`mil01-onerate_active`
`par01`	`par01-onerate_active`

Location constraint - Europe

Location	Location Constraint
`au-syd`	`au-syd-onerate_active`
`jp-tok`	`jp-tok-onerate_active`
`jp-osa`	`jp-osa-onerate_active`
`sng01`	`sng01-onerate_active`

`che01`	`che01-onerate_active`

Location constraint - Asia

Location	Location Constraint
`br-sao`	`br-sao-onerate_active`

Location constraint - South America

Billing examples

Note: These costs are examples provided to illustrate the mechanics of the billing and are not reflective of actual rates, which can be found here.

Predictable TCO pricing example

Some workloads see steadily increasing traffic as business grows - which can create some billing surprises as egress charges grow as well. A One Rate plan can cap those costs until thresholds are crossed. For example, an account with 10 TB of storage might might see consistent growth until the amount of data being read outside of the IBM Cloud exceeds the amount of data being stored.

Month	Capacity (GB)	Egress (GB)	Capacity:Egress ratio	Standard cost	One Rate cost
1	10 TB	500 GB	5%	$280	$400
2	10 TB	1 TB	10%	$325	$400
3	10 TB	2 TB	20%	$416	$400
4	10 TB	5 TB	50%	$687	$400
5	10 TB	10 TB	100%	$1,139	$400
6	10 TB	15 TB	150%	$1,591	$652
Total				**$4,438**	**$2,652**

Predictable TCO pricing

Aggregation pricing example

Imagine a large enterprise account called "Rainbow Co.". It has a number of subsidiary accounts, such as "Blue", and "Green". Each of these accounts has dozens (or more) Object Storage instances spread out across different regions. Some have large volumes of storage that is rarely read, while others have smaller volumes but very high rates of egress.

Blue (`us-east` , `us-south`):

Metric	Usage	Standard Cost
Storage	100 TB	$2,300
Class A	100	$0
Class B	100	$0
Egress	100 GB	$9
Total cost		**$2,309**

Pricing example for Blue region.

Green (`eu-de` , `mil01`):

Metric	Usage	Standard Cost
Storage	100 GB	$2
Class A	11,000,000	$55
Class B	110,000,000	$44
Egress	120 TB	$10,800
Total cost		**$10,901**

Pricing example for Green region.

Rainbow Co. (Blue and Green):

Metric	Total usage	Total Standard Cost	Allowance	Billable Quantity	One Rate Cost
Storage	100 TB	$2,302	0 GB	100 TB	$4,004
Class A	11,000,100	$55	10,010,000	990,100	$5
Class B	110,000,100	$44	100,100,000	9,900,100	$4
Egress	120 TB	$10,809	100 TB	20 TB	$1,000
Total cost		**$13,210**			**$5,013**

Pricing example for the two regions combined.

Note that the One Rate cost is significantly lower due to the reduced cost for egress. Also note that rather than dozens of individual invoices (one for each service instance), there will only be four invoices - one for each location used.

What next

Additional information can be found in the FAQs, or in the provisioning storage topics.

Choose regions and connect services

Endpoints and storage locations

Sending a REST API request or configuring a storage client requires setting a target endpoint or URL. Each storage location has its own set of URLs.

A bucket's resiliency is defined by the endpoint used to create it. *Cross Region* resiliency will spread your data across several metropolitan areas, while *Regional* resiliency will spread data across a single metropolitan area. *Single Data Center* resiliency spreads data across multiple appliances within a single data center. Regional and Cross Region buckets can maintain availability during a site outage.

Compute workloads co-located with a Regional Object Storage endpoint will see lower latency and better performance. For workloads requiring Cross Region resiliency, performance impacts are mitigated via `geo` endpoint routes connecting to the nearest Cross Region metropolitan area.

Some workloads may benefit from using a Single Data Center endpoint. Data stored in a single site is still distributed across many physical storage appliances, but is contained within a single data center. This can improve performance for compute resources within the same site, but will not maintain availability in the case of a site outage. Single Data Center buckets do not provide automated backup in the case of site destruction, so any applications using a single site should consider using replication for disaster recovery in their design.

All requests must use SSL when using IAM, and the service will reject any plain-text requests.

Note: All IBM Cloud® Object Storage endpoints support TLS 1.2 encryption.

Important: A bucket's resiliency and location that you selected during bucket creation and provisioning cannot be modified thereafter.

Endpoint Types

IBM Cloud® services are connected to a three-tiered network, segmenting public, private, and management traffic.

- **Private endpoints** are not available from a VPC, but are available for most requests originating from within IBM Cloud. Private endpoints provide better performance and do not incur charges for any outgoing or incoming bandwidth even if the traffic is cross regions or across data centers. **Whenever possible, it is best to use a private endpoint.**
- **Public endpoints** can accept requests from anywhere and charges are assessed on outgoing bandwidth. Incoming bandwidth is free. Public endpoints should be used for access not originating from an IBM Cloud cloud computing resource.
- **Direct endpoints** are used for requests originating from resources within VPCs. Like Private endpoints, Direct endpoints provide better performance over Public endpoints and do not incur charges for any outgoing or incoming bandwidth even if the traffic is cross regions or across data centers. Direct endpoints can be accessed through Virtual Private Endpoint gateways as described here.

Requests must be sent to the endpoint associated with a given bucket's location. If you aren't sure where a bucket is located, there is an extension to the bucket listing API that returns the location and storage class information for all buckets in a service instance. Another place to find an endpoint is to open the Bucket configuration tab in the IBM Cloud Console.

Note: When using Virtual Private Endpoints in an application that makes requests to IBM COS, it may be necessary to add some additional configuration for authentication. The IBM COS SDKs will automatically attempt to fetch an IAM token from `https://iam.cloud.ibm.com/identity/token`. If you are using a virtualized endpoint for token acquisition you will need alter the IAM endpoint appropriately.

Regional Endpoints

Buckets that are created at a regional endpoint distribute data across three data centers that are spread across a metro area. Any one of these data centers can suffer an outage or even destruction without impacting availability.

Region	Type	Endpoint
`us-south`	Public	`s3.us-south.cloud-object-storage.appdomain.cloud`
`us-east`	Public	`s3.us-east.cloud-object-storage.appdomain.cloud`
`eu-gb`	Public	`s3.eu-gb.cloud-object-storage.appdomain.cloud`
`eu-de`	Public	`s3.eu-de.cloud-object-storage.appdomain.cloud`

au-syd	Public	s3.au-syd.cloud-object-storage.appdomain.cloud
jp-tok	Public	s3.jp-tok.cloud-object-storage.appdomain.cloud
jp-osa	Public	s3.jp-osa.cloud-object-storage.appdomain.cloud
ca-tor	Public	s3.ca-tor.cloud-object-storage.appdomain.cloud
br-sao	Public	s3.br-sao.cloud-object-storage.appdomain.cloud
eu-es	Public	s3.eu-es.cloud-object-storage.appdomain.cloud

Regional Endpoints

Region	Type	Endpoint
us-south	Private	s3.private.us-south.cloud-object-storage.appdomain.cloud
us-east	Private	s3.private.us-east.cloud-object-storage.appdomain.cloud
eu-gb	Private	s3.private.eu-gb.cloud-object-storage.appdomain.cloud
eu-de	Private	s3.private.eu-de.cloud-object-storage.appdomain.cloud
au-syd	Private	s3.private.au-syd.cloud-object-storage.appdomain.cloud
jp-tok	Private	s3.private.jp-tok.cloud-object-storage.appdomain.cloud
jp-osa	Private	s3.private.jp-osa.cloud-object-storage.appdomain.cloud
ca-tor	Private	s3.private.ca-tor.cloud-object-storage.appdomain.cloud
br-sao	Private	s3.private.br-sao.cloud-object-storage.appdomain.cloud
eu-es	Private	s3.private.eu-es.cloud-object-storage.appdomain.cloud

Regional Endpoints

Region	Type	Endpoint
us-south	Direct	s3.direct.us-south.cloud-object-storage.appdomain.cloud
us-east	Direct	s3.direct.us-east.cloud-object-storage.appdomain.cloud
eu-gb	Direct	s3.direct.eu-gb.cloud-object-storage.appdomain.cloud
eu-de	Direct	s3.direct.eu-de.cloud-object-storage.appdomain.cloud
au-syd	Direct	s3.direct.au-syd.cloud-object-storage.appdomain.cloud
jp-tok	Direct	s3.direct.jp-tok.cloud-object-storage.appdomain.cloud
jp-osa	Direct	s3.direct.jp-osa.cloud-object-storage.appdomain.cloud
ca-tor	Direct	s3.direct.ca-tor.cloud-object-storage.appdomain.cloud
br-sao	Direct	s3.direct.br-sao.cloud-object-storage.appdomain.cloud

`eu-es`	Direct	`s3.direct.eu-es.cloud-object-storage.appdomain.cloud`

Regional Endpoints

Cross-Region Endpoints

Buckets that are created at a cross-region endpoint distribute data across three regions in a geographical location. Any one of these regions can suffer an outage or even destruction without impacting availability. Requests are routed to the nearest cross-region metropolitan area by using Border Gateway Protocol (BGP) routing. In an outage, requests are automatically rerouted to an active region. Advanced users who want to write their own failover logic can do so by sending requests to a tethered endpoint and bypassing the BGP routing.

Geo	Type	Endpoint
`us`	Public	`s3.us.cloud-object-storage.appdomain.cloud`
`eu`	Public	`s3.eu.cloud-object-storage.appdomain.cloud`
`ap`	Public	`s3.ap.cloud-object-storage.appdomain.cloud`

Cross Region Endpoints

Geo	Type	Endpoint
`us`	Private	`s3.private.us.cloud-object-storage.appdomain.cloud`
`eu`	Private	`s3.private.eu.cloud-object-storage.appdomain.cloud`
`ap`	Private	`s3.private.ap.cloud-object-storage.appdomain.cloud`

Cross Region Endpoints

Geo	Type	Endpoint
`us`	Direct	`s3.direct.us.cloud-object-storage.appdomain.cloud`
`eu`	Direct	`s3.direct.eu.cloud-object-storage.appdomain.cloud`
`ap`	Direct	`s3.direct.ap.cloud-object-storage.appdomain.cloud`

Cross Region Endpoints

For example:

- Data in `US` cross-region bucket is distributed only across regions (such as Dallas, WDC, and SJC) in the `US` geographical location.
- Data in `EU` cross-region bucket is distributed only across regions (such as, Amsterdam, FRA, and Milan) in the `EU` geographical location.
- Data in `AP` cross-region bucket is distributed only across regions (such as, TOK, SYD, and OSA) in the `AP` geographical location.

Single Data Center Endpoints

Single data centers are not co-located with IBM Cloud services, such as IAM or Key Protect, and offer no resiliency in a site outage or destruction.

⚠ **Important:** If a networking failure results in a partition where the data center is unable to access IAM, authentication and authorization information is read from a cache that might become stale. This cached data might result in a lack of enforcement of new or altered IAM policies for up to 24 hours.

Region	Type	Endpoint
`ams03`	Public	`s3.ams03.cloud-object-storage.appdomain.cloud`
`che01`	Public	`s3.che01.cloud-object-storage.appdomain.cloud`

`mil01`	Public	`s3.mil01.cloud-object-storage.appdomain.cloud`
`mon01`	Public	`s3.mon01.cloud-object-storage.appdomain.cloud`
`par01`	Public	`s3.par01.cloud-object-storage.appdomain.cloud`
`sjc04`	Public	`s3.sjc04.cloud-object-storage.appdomain.cloud`
`sng01`	Public	`s3.sng01.cloud-object-storage.appdomain.cloud`

Single Data Center Endpoints

Region	Type	Endpoint
`ams03`	Private	`s3.private.ams03.cloud-object-storage.appdomain.cloud`
`che01`	Private	`s3.private.che01.cloud-object-storage.appdomain.cloud`
`mil01`	Private	`s3.private.mil01.cloud-object-storage.appdomain.cloud`
`mon01`	Private	`s3.private.mon01.cloud-object-storage.appdomain.cloud`
`par01`	Private	`s3.private.par01.cloud-object-storage.appdomain.cloud`
`sjc04`	Private	`s3.private.sjc04.cloud-object-storage.appdomain.cloud`
`sjc01`	Private	`s3.private.sjc04.cloud-object-storage.appdomain.cloud`
`sng01`	Private	`s3.private.sng01.cloud-object-storage.appdomain.cloud`

Single Data Center Endpoints

Region	Type	Endpoint
`ams03`	Direct	`s3.direct.ams03.cloud-object-storage.appdomain.cloud`
`che01`	Direct	`s3.direct.che01.cloud-object-storage.appdomain.cloud`
`mil01`	Direct	`s3.direct.mil01.cloud-object-storage.appdomain.cloud`
`mon01`	Direct	`s3.direct.mon01.cloud-object-storage.appdomain.cloud`
`par01`	Direct	`s3.direct.par01.cloud-object-storage.appdomain.cloud`
`sjc04`	Direct	`s3.direct.sjc04.cloud-object-storage.appdomain.cloud`
`sng01`	Direct	`s3.direct.sng01.cloud-object-storage.appdomain.cloud`

Single Data Center Endpoints

EU-Managed Endpoints

The IBM Cloud Activity Tracker can archive to a bucket at specific IBM Cloud Object Storage instances. This table shows the EU-Managed locations of Object Storage instances for archiving events.

Object Storage bucket location	Resiliency	City
`ams03`	Single Site	Amsterdam

`eu-de`	Regional	Frankfurt
`eu-gb`	Regional	London
`mil01`	Single Site	Milan
`par01`	Single Site	Paris
`eu-geo`	Cross Region	Amsterdam, Frankfurt, Milan

EU-managed Endpoints

Resource Configuration Endpoints

Requests made using the Resource Configuration API are sent to a global endpoint, regardless of the bucket's location.

Type	Endpoint
Public	`config.cloud-object-storage.cloud.ibm.com/v1`
Private	`config.private.cloud-object-storage.cloud.ibm.com/v1`
Direct	`config.direct.cloud-object-storage.cloud.ibm.com/v1`

Resource Configuration Endpoints

Decommissioned locations

Over time, it may be necessary for locations to transform from a Single Data Center to a Regional configuration, or for a location to be decommissioned entirely. These situations will require users to migrate data from one bucket to another. Please consult this guide for migrating a bucket using `rclone`.

Region	Type	Endpoint
`mel01`	Public	`s3.mel01.cloud-object-storage.appdomain.cloud`
`mel01`	Private	`s3.private.mel01.cloud-object-storage.appdomain.cloud`
`mel01`	Direct	`s3.direct.mel01.cloud-object-storage.appdomain.cloud`
`mex01`	Public	`s3.mex01.cloud-object-storage.appdomain.cloud`
`mex01`	Private	`s3.private.mex01.cloud-object-storage.appdomain.cloud`
`mex01`	Direct	`s3.direct.mex01.cloud-object-storage.appdomain.cloud`
`tor01`	Public	`s3.tor01.cloud-object-storage.appdomain.cloud`
`tor01`	Private	`s3.private.tor01.cloud-object-storage.appdomain.cloud`
`tor01`	Direct	`s3.direct.tor01.cloud-object-storage.appdomain.cloud`
`osl01`	Public	`s3.osl01.cloud-object-storage.appdomain.cloud`
`osl01`	Private	`s3.private.osl01.cloud-object-storage.appdomain.cloud`
`osl01`	Direct	`s3.direct.osl01.cloud-object-storage.appdomain.cloud`
`hkg02`	Public	`s3.hkg02.cloud-object-storage.appdomain.cloud`

`hkg02`	Private	`s3.private.hkg02.cloud-object-storage.appdomain.cloud`
`hkg02`	Direct	`s3.direct.hkg02.cloud-object-storage.appdomain.cloud`
`seo01`	Public	`s3.seo01.cloud-object-storage.appdomain.cloud`
`seo01`	Private	`s3.private.seo01.cloud-object-storage.appdomain.cloud`
`seo01`	Direct	`s3.direct.seo01.cloud-object-storage.appdomain.cloud`

Decommissioned Endpoints

Using tethered endpoints

When deciding how to configure your IBM Cloud® Object Storage instance, consider how the endpoints reflect your needs for resiliency and access.

When you use a Cross Region bucket, it is possible to direct your accesses to a tethered endpoint associated with a specific Cross Region metropolitan area, rather than connecting to the nearest available Cross Region metropolitan area. In contrast to the `geo` endpoint, when you send requests to a tethered end point **there is no automated failover if that region becomes unavailable** . Applications that direct traffic to a tethered endpoint **must** implement appropriate failover logic internally to achieve the availability advantages of the Cross Region storage.

One reason for using a tethered endpoint is to control where data ingress and egress occurs while still distributing the data across the widest possible area. Imagine an application running in the `us-south` region that wants to store data in a US cross-region bucket but wants to ensure that all read and write requests remain in the Dallas area:

1. The application creates a client using the `https://s3.private.dal.us.cloud-object-storage.appdomain.cloud` endpoint.
2. The Object Storage service in Dallas suffers an outage.
3. The application detects a persistent failure trying to use the tethered endpoint.
4. The application recognizes the need to fail over to a different tethered endpoint, such as San Jose.
5. The application creates a new client using the `https://s3.private.sjc.us.cloud-object-storage.appdomain.cloud` endpoint.
6. Connectivity is resumed, and access can be re-routed to Dallas when service is restored.

Note: When sending requests to a tethered endpoint there is no automated failover if that region becomes unavailable.

For contrast, imagine another application using the normal US cross-region endpoint:

1. The application creates a client using the `https://s3.us.cloud-object-storage.appdomain.cloud` endpoint.
2. The Object Storage service in Dallas suffers an outage.
3. All Object Storage requests are automatically rerouted to San Jose or Washington until service is restored.

Tethered endpoint reference

Region	Type	Endpoint
US: Dallas	Public (Tethered)	`s3.dal.us.cloud-object-storage.appdomain.cloud`
US: San Jose	Public (Tethered)	`s3.sjc.us.cloud-object-storage.appdomain.cloud`
US: Washington, D.C.	Public (Tethered)	`s3.wdc.us.cloud-object-storage.appdomain.cloud`
EU: Amsterdam	Public (Tethered)	`s3.ams.eu.cloud-object-storage.appdomain.cloud`
EU: Frankfurt	Public (Tethered)	`s3.fra.eu.cloud-object-storage.appdomain.cloud`
EU: Milan	Public (Tethered)	`s3.mil.eu.cloud-object-storage.appdomain.cloud`
AP: Tokyo	Public (Tethered)	`s3.tok.ap.cloud-object-storage.appdomain.cloud`
AP: Sydney	Public (Tethered)	`s3.syd.ap.cloud-object-storage.appdomain.cloud`

AP: Osaka	Public (Tethered)	`s3.osa.ap.cloud-object-storage.appdomain.cloud`

Cross Region Public Endpoints

Region	Type	Endpoint
US: Dallas	Private (Tethered)	`s3.private.dal.us.cloud-object-storage.appdomain.cloud`
US: San Jose	Private (Tethered)	`s3.private.sjc.us.cloud-object-storage.appdomain.cloud`
US: Washington, D.C.	Private (Tethered)	`s3.private.wdc.us.cloud-object-storage.appdomain.cloud`
EU: Amsterdam	Private (Tethered)	`s3.private.ams.eu.cloud-object-storage.appdomain.cloud`
EU: Frankfurt	Private (Tethered)	`s3.private.fra.eu.cloud-object-storage.appdomain.cloud`
EU: Milan	Private (Tethered)	`s3.private.mil.eu.cloud-object-storage.appdomain.cloud`
AP: Tokyo	Private (Tethered)	`s3.private.tok.ap.cloud-object-storage.appdomain.cloud`
AP: Sydney	Private (Tethered)	`s3.private.syd.ap.cloud-object-storage.appdomain.cloud`
AP: Osaka	Private (Tethered)	`s3.private.osa.ap.cloud-object-storage.appdomain.cloud`

Cross Region Private Endpoints

Region	Type	Endpoint
US: Dallas	Direct (Tethered)	`s3.direct.dal.us.cloud-object-storage.appdomain.cloud`
US: San Jose	Direct (Tethered)	`s3.direct.sjc.us.cloud-object-storage.appdomain.cloud`
US: Washington, D.C.	Direct (Tethered)	`s3.direct.wdc.us.cloud-object-storage.appdomain.cloud`
EU: Amsterdam	Direct (Tethered)	`s3.direct.ams.eu.cloud-object-storage.appdomain.cloud`
EU: Frankfurt	Direct (Tethered)	`s3.direct.fra.eu.cloud-object-storage.appdomain.cloud`
EU: Milan	Direct (Tethered)	`s3.direct.mil.eu.cloud-object-storage.appdomain.cloud`
AP: Tokyo	Direct (Tethered)	`s3.direct.tok.ap.cloud-object-storage.appdomain.cloud`
AP: Sydney	Direct (Tethered)	`s3.direct.syd.ap.cloud-object-storage.appdomain.cloud`
AP: Osaka	Direct (Tethered)	`s3.direct.osa.ap.cloud-object-storage.appdomain.cloud`

Cross Region Direct Endpoints

Hosted static website endpoint reference

Region	Hosted Static Website Endpoint
US: Dallas	`s3-web.dal.us.cloud-object-storage.appdomain.cloud`
US: San Jose	`s3-web.sjc.us.cloud-object-storage.appdomain.cloud`
US: Washington, D.C.	`s3-web.wdc.us.cloud-object-storage.appdomain.cloud`
EU: Amsterdam	`s3-web.ams.eu.cloud-object-storage.appdomain.cloud`

EU: Frankfurt	`s3-web.fra.eu.cloud-object-storage.appdomain.cloud`
EU: Milan	`s3-web.mil.eu.cloud-object-storage.appdomain.cloud`
AP: Tokyo	`s3-web.tok.ap.cloud-object-storage.appdomain.cloud`
AP: Sydney	`s3-web.syd.ap.cloud-object-storage.appdomain.cloud`
AP: Osaka	`s3-web.osa.ap.cloud-object-storage.appdomain.cloud`

Cross Region Static Web Public Endpoints

Region	Hosted Static Website Endpoint
US: Dallas	`s3-web.private.dal.us.cloud-object-storage.appdomain.cloud`
US: San Jose	`s3-web.private.sjc.us.cloud-object-storage.appdomain.cloud`
US: Washington, D.C.	`s3-web.private.wdc.us.cloud-object-storage.appdomain.cloud`
EU: Amsterdam	`s3-web.private.ams.eu.cloud-object-storage.appdomain.cloud`
EU: Frankfurt	`s3-web.private.fra.eu.cloud-object-storage.appdomain.cloud`
EU: Milan	`s3-web.private.mil.eu.cloud-object-storage.appdomain.cloud`
AP: Tokyo	`s3-web.private.tok.ap.cloud-object-storage.appdomain.cloud`
AP: Sydney	`s3-web.private.syd.ap.cloud-object-storage.appdomain.cloud`
AP: Osaka	`s3-web.private.osa.ap.cloud-object-storage.appdomain.cloud`

Cross Region Static Web Private Endpoints

Region	Hosted Static Website Endpoint
US: Dallas	`s3-web.direct.dal.us.cloud-object-storage.appdomain.cloud`
US: San Jose	`s3-web.direct.sjc.us.cloud-object-storage.appdomain.cloud`
US: Washington, D.C.	`s3-web.direct.wdc.us.cloud-object-storage.appdomain.cloud`
EU: Amsterdam	`s3-web.direct.ams.eu.cloud-object-storage.appdomain.cloud`
EU: Frankfurt	`s3-web.direct.fra.eu.cloud-object-storage.appdomain.cloud`
EU: Milan	`s3-web.direct.mil.eu.cloud-object-storage.appdomain.cloud`
AP: Tokyo	`s3-web.direct.tok.ap.cloud-object-storage.appdomain.cloud`
AP: Sydney	`s3-web.direct.syd.ap.cloud-object-storage.appdomain.cloud`
AP: Osaka	`s3-web.direct.osa.ap.cloud-object-storage.appdomain.cloud`

Cross Region Static Web Direct Endpoints

Next Steps

Different services and the features they support may vary region by region. Check the documentation for more information regarding service availability.

Legacy endpoints

Over time the endpoint URLs used to access Object Storage have changed, and older applications and workflows should be updated to use the newer addresses.

Regional Endpoints

Buckets that are created at a regional endpoint distribute data across three data centers that are spread across a metro area. Any one of these data centers can suffer an outage or even destruction without impacting availability.

Region	Type	Legacy Endpoint	New Endpoint
US South	Public	`s3.us-south.objectstorage.softlayer.net`	`s3.us-south.cloud-object-storage.appdomain.cloud`
US East	Public	`s3.us-east.objectstorage.softlayer.net`	`s3.us-east.cloud-object-storage.appdomain.cloud`
EU United Kingdom	Public	`s3.eu-gb.objectstorage.softlayer.net`	`s3.eu-gb.cloud-object-storage.appdomain.cloud`
EU Germany	Public	`s3.eu-de.objectstorage.softlayer.net`	`s3.eu-de.cloud-object-storage.appdomain.cloud`
AP Australia	Public	`s3.au-syd.objectstorage.softlayer.net`	`s3.au-syd.cloud-object-storage.appdomain.cloud`
AP Japan	Public	`s3.jp-tok.objectstorage.softlayer.net`	`s3.jp-tok.cloud-object-storage.appdomain.cloud`

Regional Endpoints

Region	Type	Legacy Endpoint	New Endpoint
US South	Private	`s3.us-south.objectstorage.service.networklayer.com`	`s3.private.us-south.cloud-object-storage.appdomain.cloud`
US East	Private	`s3.us-east.objectstorage.service.networklayer.com`	`s3.private.us-east.cloud-object-storage.appdomain.cloud`
EU United Kingdom	Private	`s3.eu-gb.objectstorage.service.networklayer.com`	`s3.private.eu-gb.cloud-object-storage.appdomain.cloud`
EU Germany	Private	`s3.eu-de.objectstorage.service.networklayer.com`	`s3.private.eu-de.cloud-object-storage.appdomain.cloud`
AP Australia	Private	`s3.au-syd.objectstorage.service.networklayer.com`	`s3.private.au-syd.cloud-object-storage.appdomain.cloud`
AP Japan	Private	`s3.jp-tok.objectstorage.service.networklayer.com`	`s3.private.jp-tok.cloud-object-storage.appdomain.cloud`

Regional Endpoints

Region	Type	Legacy Endpoint	New Endpoint
US South	Direct	`s3.us-south.objectstorage.adn.networklayer.com`	`s3.direct.us-south.cloud-object-storage.appdomain.cloud`
US East	Direct	`s3.us-east.objectstorage.adn.networklayer.com`	`s3.direct.us-east.cloud-object-storage.appdomain.cloud`
EU United Kingdom	Direct	`s3.eu-gb.objectstorage.adn.networklayer.com`	`s3.direct.eu-gb.cloud-object-storage.appdomain.cloud`
EU Germany	Direct	`s3.eu-de.objectstorage.adn.networklayer.com`	`s3.direct.eu-de.cloud-object-storage.appdomain.cloud`

AP Australia	Direct	`s3.au-syd.objectstorage.adn.networklayer.com`	`s3.direct.au-syd.cloud-object-storage.appdomain.cloud`
AP Japan	Direct	`s3.jp-tok.objectstorage.adn.networklayer.com`	`s3.direct.jp-tok.cloud-object-storage.appdomain.cloud`

Regional Endpoints

Cross Region Endpoints

Buckets that are created at a cross region endpoint distribute data across three regions. Any one of these regions can suffer an outage or even destruction without impacting availability. Requests are routed to the nearest Cross Region metropolitan area by using Border Gateway Protocol (BGP) routing. In an outage, requests are automatically rerouted to an active region. Advanced users who want to write their own failover logic can do so by sending requests to a tethered endpoint and bypassing the BGP routing.

Region	Type	Legacy Endpoint	New Endpoint
US Cross Region	Public	`s3-api.us-geo.objectstorage.softlayer.net`	`s3.us.cloud-object-storage.appdomain.cloud`
EU Cross Region	Public	`s3.eu-geo.objectstorage.softlayer.net`	`s3.eu.cloud-object-storage.appdomain.cloud`
AP Cross Region	Public	`s3.ap-geo.objectstorage.softlayer.net`	`s3.ap.cloud-object-storage.appdomain.cloud`

Cross Region Endpoints

Region	Type	Legacy Endpoint	New Endpoint
US Cross Region	Private	`s3-api.us-geo.objectstorage.service.networklayer.com`	`s3.private.us.cloud-object-storage.appdomain.cloud`
EU Cross Region	Private	`s3.eu-geo.objectstorage.service.networklayer.com`	`s3.private.eu.cloud-object-storage.appdomain.cloud`
AP Cross Region	Private	`s3.ap-geo.objectstorage.service.networklayer.com`	`s3.private.ap.cloud-object-storage.appdomain.cloud`

Cross Region Endpoints

Region	Type	Legacy Endpoint	New Endpoint
US Cross Region	Direct	`s3-api.us-geo.objectstorage.adn.networklayer.com`	`s3.direct.us.cloud-object-storage.appdomain.cloud`
EU Cross Region	Direct	`s3.eu-geo.objectstorage.adn.networklayer.com`	`s3.direct.eu.cloud-object-storage.appdomain.cloud`
AP Cross Region	Direct	`s3.ap-geo.objectstorage.adn.networklayer.com`	`s3.direct.ap.cloud-object-storage.appdomain.cloud`

Cross Region Endpoints

Tethered endpoints

Region	Type	Legacy Endpoint	New Endpoint
US: Dallas	Public (Tethered)	`s3-api.dal-us-geo.objectstorage.softlayer.net`	`s3.dal.us.cloud-object-storage.appdomain.cloud`
US: San Jose	Public (Tethered)	`s3-api.sjc-us-geo.objectstorage.softlayer.net`	`s3.sjc.us.cloud-object-storage.appdomain.cloud`
US: Washington, D.C.	Public (Tethered)	`s3-api.wdc-us-geo.objectstorage.softlayer.net`	`s3.wdc.us.cloud-object-storage.appdomain.cloud`

EU: Amsterdam	Public (Tethered)	`s3.ams-eu-geo.objectstorage.softlayer.net`	`s3.ams.eu.cloud-object-storage.appdomain.cloud`
EU: Frankfurt	Public (Tethered)	`s3.fra-eu-geo.objectstorage.softlayer.net`	`s3.fra.eu.cloud-object-storage.appdomain.cloud`
EU: Milan	Public (Tethered)	`s3.mil-eu-geo.objectstorage.softlayer.net`	`s3.mil.eu.cloud-object-storage.appdomain.cloud`
AP: Tokyo	Public (Tethered)	`s3.tok-ap-geo.objectstorage.softlayer.net`	`s3.tok.ap.cloud-object-storage.appdomain.cloud`
AP: Sydney	Public (Tethered)	`s3.syd-ap-geo.objectstorage.softlayer.net`	`s3.syd.ap.cloud-object-storage.appdomain.cloud`
AP: Osaka	Public (Tethered)	`s3.osa-ap-geo.objectstorage.softlayer.net`	`s3.osa.ap.cloud-object-storage.appdomain.cloud`

Table 2a. Cross Region Endpoints (Tethered)

Region	Type	Legacy Endpoint	New Endpoint
US: Dallas	Private (Tethered)	`s3-api.dal-us-geo.objectstorage.service.networklayer.com`	`s3.private.dal.us.cloud-object-storage.appdomain.cloud`
US: San Jose	Private (Tethered)	`s3-api.sjc-us-geo.objectstorage.service.networklayer.com`	`s3.private.sjc.us.cloud-object-storage.appdomain.cloud`
US: Washington, D.C.	Private (Tethered)	`s3-api.wdc-us-geo.objectstorage.service.networklayer.com`	`s3.private.wdc.us.cloud-object-storage.appdomain.cloud`
EU: Amsterdam	Private (Tethered)	`s3.ams-eu-geo.objectstorage.service.networklayer.com`	`s3.private.ams.eu.cloud-object-storage.appdomain.cloud`
EU: Frankfurt	Private (Tethered)	`s3.fra-eu-geo.objectstorage.service.networklayer.com`	`s3.private.fra.eu.cloud-object-storage.appdomain.cloud`
EU: Milan	Private (Tethered)	`s3.mil-eu-geo.objectstorage.service.networklayer.com`	`s3.private.mil.eu.cloud-object-storage.appdomain.cloud`
AP: Tokyo	Private (Tethered)	`s3.tok-ap-geo.objectstorage.service.networklayer.com`	`s3.private.tok.ap.cloud-object-storage.appdomain.cloud`
AP: Sydney	Private (Tethered)	`s3.syd-ap-geo.objectstorage.service.networklayer.com`	`s3.private.syd.ap.cloud-object-storage.appdomain.cloud`
AP: Osaka	Private (Tethered)	`s3.osa-ap-geo.objectstorage.service.networklayer.com`	`s3.private.osa.ap.cloud-object-storage.appdomain.cloud`

Table 2a. Cross Region Endpoints (Tethered)

Region	Type	Legacy Endpoint	New Endpoint
US: Dallas	Direct (Tethered)	`s3-api.dal-us-geo.objectstorage.adn.networklayer.com`	`s3.direct.dal.us.cloud-object-storage.appdomain.cloud`
US: San Jose	Direct (Tethered)	`s3-api.sjc-us-geo.objectstorage.adn.networklayer.com`	`s3.direct.sjc.us.cloud-object-storage.appdomain.cloud`
US: Washington, D.C.	Direct (Tethered)	`s3-api.wdc-us-geo.objectstorage.adn.networklayer.com`	`s3.direct.wdc.us.cloud-object-storage.appdomain.cloud`

EU: Amsterdam	Direct (Tethered)	`s3.ams-eu-geo.objectstorage.adn.networklayer.com`	`s3.direct.ams.eu.cloud-object-storage.appdomain.cloud`
EU: Frankfurt	Direct (Tethered)	`s3.fra-eu-geo.objectstorage.adn.networklayer.com`	`s3.direct.fra.eu.cloud-object-storage.appdomain.cloud`
EU: Milan	Direct (Tethered)	`s3.mil-eu-geo.objectstorage.adn.networklayer.com`	`s3.direct.mil.eu.cloud-object-storage.appdomain.cloud`
AP: Tokyo	Direct (Tethered)	`s3.tok-ap-geo.objectstorage.adn.networklayer.com`	`s3.direct.tok.ap.cloud-object-storage.appdomain.cloud`
AP: Sydney	Direct (Tethered)	`s3.syd-ap-geo.objectstorage.adn.networklayer.com`	`s3.direct.syd.ap.cloud-object-storage.appdomain.cloud`
AP: Osaka	Direct (Tethered)	`s3.osa-ap-geo.objectstorage.adn.networklayer.com`	`s3.direct.osa.ap.cloud-object-storage.appdomain.cloud`

Table 2a. Cross Region Endpoints (Tethered)

Single Data Center Endpoints

Single data centers are not co-located with IBM Cloud services, such as IAM or Key Protect, and offer no resiliency in a site outage or destruction.

Important: If a networking failure results in a partition where the data center is unable to access IAM, authentication and authorization information is read from a cache that might become stale. This cached data might result in a lack of enforcement of new or altered IAM policies for up to 24 hours.

Region	Type	Legacy Endpoint	New Endpoint
Amsterdam, Netherlands	Public	`s3.ams03.objectstorage.softlayer.net`	`s3.ams03.cloud-object-storage.appdomain.cloud`
Chennai, India	Public	`s3.che01.objectstorage.softlayer.net`	`s3.che01.cloud-object-storage.appdomain.cloud`
Milan, Italy	Public	none	`s3.mil01.cloud-object-storage.appdomain.cloud`
Montrèal, Canada	Public	`s3.mon01.objectstorage.softlayer.net`	`s3.mon01.cloud-object-storage.appdomain.cloud`
Paris, France	Public	`s3.par01.objectstorage.softlayer.net`	`s3.par01.cloud-object-storage.appdomain.cloud`
San Jose, US	Public	none	`s3.sjc04.cloud-object-storage.appdomain.cloud`
São Paulo, Brazil	Public	`s3.sao01.objectstorage.softlayer.net`	`s3.sao01.cloud-object-storage.appdomain.cloud`
Singapore	Public	none	`s3.sng01.cloud-object-storage.appdomain.cloud`

Single Data Center Endpoints

Region	Type	Legacy Endpoint	New Endpoint
Amsterdam, Netherlands	Private	`s3.ams03.objectstorage.service.networklayer.com`	`s3.private.ams03.cloud-object-storage.appdomain.cloud`
Chennai, India	Private	`s3.che01.objectstorage.service.networklayer.com`	`s3.private.che01.cloud-object-storage.appdomain.cloud`
Milan, Italy	Private	none	`s3.private.mil01.cloud-object-storage.appdomain.cloud`

Montrèal, Canada	Private	`s3.mon01.objectstorage.service.networklayer.com`	`s3.private.mon01.cloud-object-storage.appdomain.cloud`
Paris, France	Private	`s3.par01.objectstorage.service.networklayer.com`	`s3.private.par01.cloud-object-storage.appdomain.cloud`
San Jose, US	Private	none	`s3.private.sjc04.cloud-object-storage.appdomain.cloud`
São Paulo, Brazil	Private	`s3.sao01.objectstorage.service.networklayer.com`	`s3.private.sao01.cloud-object-storage.appdomain.cloud`
Singapore	Private	none	`s3.private.sng01.cloud-object-storage.appdomain.cloud`

Single Data Center Endpoints

Region	Type	Legacy Endpoint	New Endpoint
Amsterdam, Netherlands	Direct	`s3.ams03.objectstorage.adn.networklayer.com`	`s3.direct.ams03.cloud-object-storage.appdomain.cloud`
Chennai, India	Direct	`s3.che01.objectstorage.adn.networklayer.com`	`s3.direct.che01.cloud-object-storage.appdomain.cloud`
Milan, Italy	Direct	none	`s3.direct.mil01.cloud-object-storage.appdomain.cloud`
Montrèal, Canada	Direct	`s3.mon01.objectstorage.adn.networklayer.com`	`s3.direct.mon01.cloud-object-storage.appdomain.cloud`
Paris, France	Direct	`s3.par01.objectstorage.adn.networklayer.com`	`s3.direct.par01.cloud-object-storage.appdomain.cloud`
San Jose, US	Direct	none	`s3.direct.sjc04.cloud-object-storage.appdomain.cloud`
São Paulo, Brazil	Direct	`s3.sao01.objectstorage.adn.networklayer.com`	`s3.direct.sao01.cloud-object-storage.appdomain.cloud`
Singapore	Direct	none	`s3.direct.sng01.cloud-object-storage.appdomain.cloud`

Single Data Center Endpoints

Integrated service availability

The document describes the regions where services and the different kinds of availability that are supported.

For more information about the following services, be sure to check out the respective links:

- Aspera high-speed transfer
- Key Protect (SSE-KP)
- Hyper Protect Crypto Services
- Archive Data
- Object Lock
- Immutable Object Storage
- Activity Tracker
- Functions
- Code Engine
- Smart Tier

- Metrics Routing

Tip: Downloads that use Aspera high-speed transfer incur extra egress charges. For more information, see the pricing page.

Cross Region

Region	Aspera	Key Protect	Hyper Protect Crypto Services	Archive Data	Object Lock	Immutable Object Storage	Activity Tracker Routing	Code Engine	Smart Tier	Metrics Routing	Replication	One Rate
`ap`	Yes	Yes (in `jp-tok`)	No	No	Yes	No	`ap-tok`	No	Yes	`ap-tok`	Yes	No
`eu`	Yes	Yes (in `eu-de`)	No	No	Yes	No	`eu-de`	No	Yes	`eu-de`	Yes	No
`us`	Yes	Yes (in `us-south`)	Yes (failover in `us-east`)	No	Yes	Yes	`us-south`	No	Yes	`us-south`	Yes	No

Regional

Region	Aspera	Key Protect	Hyper Protect Crypto Services	Archive Data	Object Lock	Immutable Object Storage	Activity Tracker Routing	Code Engine	Smart Tier	Metrics Routing	Replication	One Rate	Coc Eng
`au-syd`	Yes	Yes	Yes (see note)	Yes	Yes	Yes	`au-syd`	Yes	Yes	`au-syd`	Yes	Yes	`au-`
`jp-tok`	Yes	Yes	Yes (see note)	Yes	Yes	Yes	`ap-tok`	Yes	Yes	`ap-tok`	Yes	Yes	`jp-`
`jp-osa`	No	Yes	No	Yes	Yes	Yes	`ap-osa`	Yes	Yes	`ap-osa`	Yes	Yes	`jp-`
`eu-gb`	Yes	Yes	Yes (see note)	Yes	Yes	Yes	`eu-gb`	Yes	Yes	`eu-gb`	Yes	Yes	`eu-`
`eu-de`	Yes	Yes	Yes (see note)	Yes	Yes	Yes	`eu-de`	Yes	Yes	`eu-de`	Yes	Yes	`eu-`
`us-south`	Yes	Yes	Yes (see note)	Yes	Yes	Yes	`us-south`	Yes	Yes	`us-south`	Yes	Yes	`us-sou`
`us-east`	Yes	Yes	Yes (see note)	Yes	Yes	Yes	`us-east`	Yes	Yes	`us-east`	Yes	Yes	`us-eas`
`ca-tor`	No	Yes	Yes (see note)	Yes	Yes	Yes	`ca-tor`	Yes	Yes	`ca-tor`	Yes	Yes	`ca-`
`br-sao`	No	Yes	Yes (see note)	Yes	Yes	Yes	`br-sao`	Yes	Yes	`br-sao`	Yes	Yes	`br-`
`eu-es`	No	Yes	Yes (see note)	Yes	Yes	Yes	`eu-es`	No	Yes	`eu-es`	Yes	Yes	`eu-`

Note: It is possible to create a bucket and associate any available Key Protect or Hyper Protect Crypto Services instance with any of the listed Cloud Object Storage locations. Hyper Protect Crypto Services is only available in selected locations and it is your responsibility to ensure the location/region you select meets any pertinent requirements. Please refer to [Hyper Protect Crypto Services documentation]() and [IBM Key Protect]() for a list of regions/locations currently available.

Single Data Centers

Region	Aspera	Key Protect	Hyper Protect Crypto Services	Archive Data	Object Lock	Immutable Object Storage	Activity Tracker Routing	Code Engine	Smart Tier	Metrics Routing	Replication	One Rate
`ams03`	No	No	No	No	Yes	No	`eu-de`	No	Yes	`eu-de`	Yes	Yes
`che01`	Yes	No	No	Yes	Yes	No	`che01`	No	Yes	`jp-tok`	Yes	Yes
`mil01`	No	No	No	No	Yes	No	`eu-de`	No	Yes	`eu-de`	Yes	Yes
`mon01`	No	No	No	No	Yes	No	`ca-tor`	No	Yes	`ca-tor`	Yes	Yes
`par01`	No	No	No	No	Yes	No	`eu-de`	No	Yes	`eu-de`	Yes	Yes
`sjc04`	No	No	No	No	Yes	No	`us-south`	No	Yes	`us-south`	Yes	Yes
`sng01`	No	No	No	No	Yes	No	`ap-tok`	No	Yes	`ap-tok`	Yes	Yes

Satellite

Location	Aspera	Key Protect (IBM Cloud)	Hyper Protect Crypto Services	Archive Data	Object Lock	Immutable Object Storage	Activity Tracker Routing	Code Engine	Smart Tier	Metrics Routing
`us-east`	No	Yes	No	No	No	No	No	No	No	No
`eu-de`	No	Yes	No	No	No	No	No	No	No	No
`eu-gb`	No	Yes	No	No	No	No	No	No	No	No
`jp-tok`	No	Yes	No	No	No	No	No	No	No	No

More information

Learn more about how locations are represented by [endpoints]() for users of IBM Cloud Object Storage.

Using Virtual Private Endpoints

IBM Cloud® Virtual Private Endpoint (VPE) for IBM Cloud® Object Storage provides connection points to IBM services on the IBM Cloud® internal network from your VPC network.

Using Virtual Private Endpoints

Note: Virtual Private Endpoints (VPEs) are generally available in all regions.

Before you begin

- You need to have an IBM Cloud account
- You also need an instance of IBM Cloud Object Storage

Setting up your VPE

1. Create an IBM Cloud® Virtual Private Cloud to host the applications that need to access your IBM Cloud Object Storage buckets. See Getting started with VPC.
2. Find the location and the corresponding direct endpoint where your bucket is located.
3. In the IBM Cloud console, click the menu icon and select VPC Infrastructure -> Network -> Virtual private endpoint gateways. Create a VPE for your IBM Cloud instances with the following instructions.
4. After you create your VPE, it may take a few minutes for the new VPE and DNS to complete the process and begin working for your VPC. Completion is confirmed when you see an IP address set in the details view of the VPE.

VPE Discoverability

Following the previous steps results in a VPE that provides access over the internal IBM Cloud® network from your VPC network to all of your buckets in a particular location.

Important: Each access to your buckets from your IBM Cloud VPC will require authorization at the S3 API level. To further restrict this access to specific IP addresses, or ranges of IP addresses, provide the IBM Cloud VPC ID or name when configuring the context-based restrictions.

Tip: The VPE details page will provide you with more information, including IP address, after creation.

More resources

- About virtual private endpoint gateways
- Planning for virtual private endpoint gateways
- Creating an endpoint gateway
- For further assistance, see the FAQs for virtual private endpoints here, and the `Troubleshooting VPE gateways` documentation that includes how to fix communications issues here.

Migrating resources to a different data center

IBM Cloud invests significantly in data center infrastructure. These investments include rolling out newer data centers and multizone regions (MZRs) designed to deliver a more resilient architecture with higher levels of network throughput and redundancy.

Part of this modernization strategy is to close older data centers that are unsuitable for upgrading. As this transition approaches, help is available to assist you in your migration to modern data centers. For a list of the available data centers, see Endpoints and storage locations.

For additional information about data center closings, see Withdrawal of support for some data centers.

To identify your impacted resources, take advantage of special offers, or learn about recommended configurations, use one of the following options to contact the IBM 24x7 Client Success team:

- Live chat (Click 'Lets Talk' in the lower right corner)
- Phone: (US) 866-597-9687; (EMEA) +31 20 308 0540; (APAC) +65 6622 2231

Migrating your resources

To avoid any disruption to your service, please complete the following steps **before any announced deadlines**:

1. Identify your buckets in the data centers that are set to close through viewing your COS UI buckets page. Also, Extended listing can be used for this purpose, as it will return 'LocationConstraint' values that indicate the location in addition to the storage class of a bucket. For more information, contact the Client Success team Live chat.
2. Migrate your data to the new destination bucket using `Rclone`.
3. To avoid being double billed for data in your old and new buckets, empty your old buckets and delete them.

Identifying buckets that require migration

You can use the IBM Cloud CLI to identify buckets located in a given location.

1. First, ensure you have both the [IBM Cloud CLI] and [COS plug-in] installed.
2. After you are logged into the CLI, you can use the `ibmcloud cos buckets-extended` command to list all of the buckets in a given instance.
3. You can filter the results using `grep`. For example, `ibmcloud cos buckets-extended | grep mon01` will return all buckets that are located in the `mon01` single data center.

Bucket management

Managing access

Getting Started with IAM

Access to IBM Cloud® Object Storage service instances for users in your account is controlled by IBM Cloud Identity and Access Management (IAM).

Identity and Access Management roles

Every user that accesses the IBM Cloud® Object Storage service in your account must be assigned an access policy with an IAM user role defined. That policy determines what actions the user can perform within the context of the service or instance you select. The allowable actions are customized and defined by the IBM Cloud service as operations that are allowed to be performed on the service. The actions are then mapped to IAM user roles.

Policies enable access to be granted at different levels. Some of the options include the following:

- Access across all instances of the service in your account
- Access to an individual service instance in your account
- Access to a specific bucket within an instance (see Bucket permissions)
- Access to all IAM-enabled services in your account
- Access to a specific object or group of objects within a bucket

After you define the scope of the access policy, you assign a role. Review the following tables which outline what actions each role allows within the Object Storage service.

The following table details actions that are mapped to platform management roles. Platform management roles enable users to perform tasks on service resources at the platform level, for example assign user access for the service, create or delete service IDs, create instances, and bind instances to applications.

Platform management role	**Description of actions**	**Example actions**
Viewer	View service instances but not modify them	• List available COS service instances • View COS service plan details • View usage details
Editor	Perform all platform actions except for managing the accounts and assigning access policies	• Create and delete COS service instances
Operator	Not used by COS	None
Administrator	Perform all platform actions based on the resource this role is being assigned, including assigning access policies to other users, as well as setting PublicAccess policy on buckets.	• Update user policies • Update pricing plans

IAM user roles and actions

The following table details actions that are mapped to service access roles. Service access roles enable users access to Object Storage as well as the ability to call the Object Storage API.

Service access role	**Description of actions**	**Example actions**

Object Writer	Upload and overwrite objects (including uploading objects in multiple parts).	• Upload objects
Object Reader	Download objects, read object metadata (headers), but not list objects or buckets.	• Download objects
Content Reader	Download and list objects, read object metadata (headers), but not list buckets.	• Download and list objects
Reader	In addition to Content Reader actions, Readers can list buckets and read bucket metadata, but not make modifications.	• List buckets
Writer	In addition to Reader actions, Writers can create buckets and upload objects.	• Create new buckets and objects • Remove buckets and objects
Manager	In addition to Writer actions, Managers can complete privileged actions that affect access control.	• Configure retention policies • Configure bucket firewalls • Block public ACLs

IAM service access roles and actions

For information about assigning user roles in the UI, see [Managing IAM access](#).

Identity and Access Management actions

Action id	Description	Condition attributes supported
`cloud-object-storage.account.get_account_buckets`	List all buckets in a service instance.	none
`cloud-object-storage.bucket.put_bucket`	Create a bucket.	none
`cloud-object-storage.bucket.post_bucket`	Internal use only - unsupported for users.	none
`cloud-object-storage.bucket.delete_bucket`	Delete a bucket.	none
`cloud-object-storage.bucket.get`	List all the objects in a bucket.	prefix, delimiter
`cloud-object-storage.bucket.list_crk_id`	List the IDs of encryption root keys associated with a bucket.	none
`cloud-object-storage.bucket.head`	View bucket metadata.	none
`cloud-object-storage.bucket.get_versions`	List object versions.	prefix, delimiter
`cloud-object-storage.bucket.get_uploads`	List all active multipart uploads for a bucket.	prefix, delimiter
`cloud-object-storage.bucket.put_quota`	Unsupported operation - used for S3 API compatibility only.	none
`cloud-object-storage.bucket.get_acl`	Read a bucket ACL [deprecated].	none

`cloud-object-storage.bucket.put_acl`	Create a bucket ACL [deprecated].	none
`cloud-object-storage.bucket.get_cors`	Read CORS rules.	none
`cloud-object-storage.bucket.put_cors`	Add CORS rules to a bucket.	none
`cloud-object-storage.bucket.delete_cors`	Delete CORS rules.	none
`cloud-object-storage.bucket.get_website`	Read bucket website configuration.	none
`cloud-object-storage.bucket.put_website`	Add bucket website configuration.	none
`cloud-object-storage.bucket.delete_website`	Delete bucket website configuration.	none
`cloud-object-storage.bucket.get_versioning`	Check versioning status of a bucket.	none
`cloud-object-storage.bucket.put_versioning`	Enable versioning on a bucket.	none
`cloud-object-storage.bucket.get_object_lock_configuration`	Get Object Lock Configuration from the bucket.	none
`cloud-object-storage.bucket.put_object_lock_configuration`	Set Object Lock Configuration from the bucket.	none
`cloud-object-storage.bucket.get_fasp_connection_info`	View Aspera FASP connection information.	none
`cloud-object-storage.account.delete_fasp_connection_info`	Delete Aspera FASP connection information.	none
`cloud-object-storage.bucket.get_location`	View the location and storage class of a bucket.	none
`cloud-object-storage.bucket.get_lifecycle`	Read a bucket lifecycle policy.	none
`cloud-object-storage.bucket.put_lifecycle`	Create a bucket lifecycle policy.	none
`cloud-object-storage.bucket.get_basic`	Read bucket metadata (number of objects, etc) using the Resource Configuration API.	none
`cloud-object-storage.bucket.get_activity_tracking`	Read activity tracking configuration.	none
`cloud-object-storage.bucket.put_activity_tracking`	Add activity tracking configuration.	none
`cloud-object-storage.bucket.get_metrics_monitoring`	Read metrics monitoring configuration.	none
`cloud-object-storage.bucket.put_metrics_monitoring`	Add metrics monitoring configuration.	none
`cloud-object-storage.bucket.put_protection`	Add Immutable Object Storage policy.	none
`cloud-object-storage.bucket.get_protection`	Read Immutable Object Storage policy.	none
`cloud-object-storage.bucket.put_firewall`	Add a firewall configuration.	none
`cloud-object-storage.bucket.get_firewall`	Read a firewall configuration.	none

`cloud-object-storage.bucket.put_public_access_block`	Add/Update a public access block configuration for a bucket.	none
`cloud-object-storage.bucket.delete_public_access_block`	Remove public access block configuration for a bucket.	none
`cloud-object-storage.bucket.get_public_access_block`	Retrieve public access block configuration for a bucket.	none
`cloud-object-storage.bucket.list_bucket_crn`	View a bucket CRN.	none
`cloud-object-storage.bucket.get_notifications`	Internal use only - unsupported for users.	none
`cloud-object-storage.bucket.put_notifications`	Internal use only - unsupported for users.	none
`cloud-object-storage.bucket.get_replication`	Read replication configuration of a bucket.	none
`cloud-object-storage.bucket.put_replication`	Add replication configuration to a bucket.	none
`cloud-object-storage.bucket.delete_replication`	Delete replication configuration of a bucket.	none
`cloud-object-storage.object.get`	View and download objects.	path
`cloud-object-storage.object.head`	Read an object's metadata.	path
`cloud-object-storage.object.get_version`	Read a specified version of an object.	path
`cloud-object-storage.object.head_version`	Get headers for a specific version of an object.	path
`cloud-object-storage.object.put`	Write and upload objects.	path
`cloud-object-storage.object.post`	Upload an object using HTML forms [deprecated].	path
`cloud-object-storage.object.post_md`	Update object metadata using HTML forms [deprecated].	path
`cloud-object-storage.object.post_initiate_upload`	Initiate multipart uploads.	path
`cloud-object-storage.object.put_part`	Upload an object part.	path
`cloud-object-storage.object.copy_part`	Copy (write) an object part.	path
`cloud-object-storage.object.copy_part_get`	Copy (read) an object part.	path
`cloud-object-storage.object.copy_part_get_version`	Copy (read) an object part.	path
`cloud-object-storage.object.post_complete_upload`	Complete a multipart upload.	path
`cloud-object-storage.object.copy`	Copy (write) an object from one bucket to another.	path
`cloud-object-storage.object.copy_get`	Copy (read) an object from one bucket to another.	path
`cloud-object-storage.object.copy_get_version`	Copy (read) an object from one bucket to another.	path
`cloud-object-storage.object.get_acl`	Read object ACL [deprecated].	path
`cloud-object-storage.object.get_acl_version`	Read object ACL Version [deprecated]	path

`cloud-object-storage.object.put_acl`	Write object ACL [deprecated].	path
`cloud-object-storage.object.put_acl_version`	Unsupported operation - used for S3 API compatibility only.	path
`cloud-object-storage.object.delete`	Delete an object.	path
`cloud-object-storage.object.delete_version`	Delete a specific version of an object.	path
`cloud-object-storage.object.get_uploads`	List parts of an object.	path
`cloud-object-storage.object.delete_upload`	Abort a multipart upload.	path
`cloud-object-storage.object.restore`	Temporarily restore an archived object.	path
`cloud-object-storage.object.restore_version`	Temporarily restore an archived object.	path
`cloud-object-storage.object.get_tagging`	Read object tag versions	path
`cloud-object-storage.object.put_tagging`	Add/Update object tags	path
`cloud-object-storage.object.delete_tagging`	Delete object tags	path
`cloud-object-storage.object.post_multi_delete`	Delete multiple objects.	none
`cloud-object-storage.object.post_legal_hold`	Add a legal hold to an object.	path
`cloud-object-storage.object.get_legal_hold`	View any legal holds on an object.	path
`cloud-object-storage.object.post_extend_retention`	Extend a retention policy.	path
`cloud-object-storage.object.get_object_lock_retention`	Get object lock retention settings on the object.	path
`cloud-object-storage.object.put_object_lock_retention`	Set object lock retention settings on the object.	path
`cloud-object-storage.object.get_object_lock_legal_hold`	Get object lock legal hold state on the object.	path
`cloud-object-storage.object.put_object_lock_legal_hold`	Set object lock legal hold state on the object.	path
`cloud-object-storage.object.get_object_lock_retention_version`	Get object lock retention version settings on the object.	path
`cloud-object-storage.object.put_object_lock_retention_version`	Set object lock retention version settings on the object.	path
`cloud-object-storage.object.get_object_lock_legal_hold_version`	Get object lock legal hold state on the object.	path
`cloud-object-storage.object.put_object_lock_legal_hold_version`	Set object lock legal hold state on the object.	path
`cloud-object-storage.object.put_tagging_version`	Add/Update object tag versions	path
`cloud-object-storage.object.get_tagging_version`	Read object tag versions	path

`cloud-object-storage.object.delete_tagging_version`	Delete object tag versions	path

Granular IAM action descriptions

IAM overview

IBM Cloud Identity and Access Management (IAM) service securely authenticates users and controls access to all resources consistently in the IBM Cloud Platform.

For more information, see the IAM account overview.

Identity Management

Identity Management includes the interaction of users, services, and resources. Users are identified by their IBMid. Services are identified by their service IDs. And, resources are identified and addressed by using CRNs.

The IBM Cloud IAM Token Service is used to create, update, delete, and use API keys for users and services. Those API keys are created either with API calls or the Identity & Access section of the IBM Cloud® Platform Console. The same key can be used across services. Each user has any number of API keys to support key rotation scenarios, as well as scenarios by using different keys for different purposes to limit the exposure of a single key.

For more information, see the IAM documentation.

Users and API keys

API keys can be created and used by IBM Cloud users for automation and scripting purposes. API keys can be created in the Identity and Access Management UI or by using the `ibmcloud` CLI.

Service IDs and API keys

Users also can create Service IDs and API keys for Service IDs. A Service ID is similar to a "functional ID" or an "application ID" and is used to authenticate services, and not to represent a user.

Users create Service IDs and bind them to scopes, like an IBM Cloud Platform account, a CloudFoundry organization, or a CloudFoundry space. It is best to bind Service IDs to an IBM Cloud Platform account. This binding is done to give the Service ID a container to live in. This container also defines who can update and delete the Service ID and who can create, update, read, and delete API Keys that are associated to that Service ID. It is important to note that a Service ID is NOT related to a user.

Key rotation

API keys can be regularly rotated to prevent any security breaches caused by leaked keys.

Access Management

IAM Access Control provides a common way to assign user roles for IBM Cloud resources and controls the actions that the users can take on those resources. You can view and manage users across the account or organization, depending on the access options that you have been given. For example, account owners are automatically assigned the account Administrator role for Identity and Access Management, which enables them to assign and manage service policies for all members of their account.

Users, roles, resources, and policies

IAM Access Control enables the assignment of policies per service or service instance to allow levels of access for managing resources and users within the assigned context. A policy grants a user a role or roles to a set of resources by using a combination of attributes to define the applicable set of resources. When you assign a policy to a user, you first specify the service then a role or roles to assign. Extra configuration options might be available depending on the service you select.

While roles are a collection of actions, the actions that are mapped to these roles are service specific. Each service determines this role to action mapping during the onboarding process and this mapping effects all users of the service. Roles and access policies are configured through the Policy Administration Point (PAP) and enforced through the Policy Enforcement Point (PEP) and Policy Decision Point (PDP).

See Best practices for organizing resources and assigning access to learn more.

Service credentials

A service credential provides the necessary information to connect an application to Object Storage packaged in a JSON document.

Service credentials are always associated with a Service ID, and new Service IDs can be created along with a new credential.

Important: To view a credential you must be granted the Administrator platform role or a custom role that has the `resource-controller.credential.retrieve_all` action. For more information about this update, see the documentation.

Use the following steps to create a service credential:

1. Log in to the IBM Cloud console and navigate to your instance of Object Storage.
2. In the side navigation, click **Service Credentials**.
3. Click **New credential** and provide the necessary information. If you want to generate HMAC credentials, switch the `Include HMAC Credential` to `On`. Verify the option is switched to `On` before continuing.
4. Click **Add** to generate service credential.

Tip: When creating a service credential, it is possible to provide a value of `None` for the role. This will prevent the creation of unintended or unnecessary IAM access policies. Any access policies for the associated service ID will need to be managed using the IAM console or APIs.

The credential has the following values:

Field name	Value
`apikey`	New API key that is created for the Service ID
`cos_hmac_keys`	Access Key and Secret Key pair for use with S3-compatible tools and libraries
`endpoints`	Link to JSON representation of available endpoints
`iam_apikey_description`	API key description - initially generated but editable
`iam_apikey_name`	API key name - initially generated but editable
`iam_role_crn`	Unique identifier for the assigned role
`iam_serviceid_crn`	Unique identifier for the Service ID
`resource_instance_id`	Unique identifier for the instance of Object Storage the credential accesses. This is also referred to as a service credential.

Credential values

This is an example of a service credential:

```
$ {
  "apikey": "0viPHOY7LbLNa9eLftrtHPpTjoGv6hbLD1QalRXikliJ",
  "cos_hmac_keys": {
      "access_key_id": "347aa3a4b34344f8bc7c7cccdf856e4c",
      "secret_access_key": "gvurfb82712ad14W7a7915h763a6i87155d30a1234364f61"
  },
  "endpoints": "https://control.cloud-object-storage.test.cloud.ibm.com/v2/endpoints",
  "iam_apikey_description": "Auto generated apikey during resource-key operation for Instance - crn:v1:bluemix:public:cloud-
object-storage:global:a/3ag0e9402tyfd5d29761c3e97696b71n:d6f74k03-6k4f-4a82-b165-697354o63903::",
  "iam_apikey_name": "auto-generated-apikey-f9274b63-ef0b-4b4e-a00b-b3bf9023f9dd",
  "iam_role_crn": "crn:v1:bluemix:public:iam::::serviceRole:Manager",
  "iam_serviceid_crn": "crn:v1:bluemix:public:iam-identity::a/3ag0e9402tyfd5d29761c3e97696b71n::serviceid:ServiceId-540a4a41-
7322-4fdd-a9e7-e0cb7ab760f9",
  "resource_instance_id": "crn:v1:bluemix:public:cloud-object-storage:global:a/3ag0e9402tyfd5d29761c3e97696b71n:d6f74k03-6k4f-
4a82-b165-697354o63903::"
}
```

You can also use the IBM Cloud CLI to create a new service credential (which is a subset of something called a *service key*). This example extracts the credential and writes it to a file where the IBM COS SDKs can automatically source the API key and Service Instance ID. First, create the Service Key (called `config-example` and associated with a COS instance called `cos-dev-enablement` in this example):

```
$ ic resource service-key-create config-example --instance-name cos-dev-enablement
```

Then, extract the credential and create the `cos_credential` file:

```
$ ic resource service-key config-example --output JSON | jq '.[].credentials' > ~/.bluemix/cos_credentials
```

Understanding the `endpoints` objects

The `endpoints` URL (`https://control.cloud-object-storage.cloud.ibm.com/v2/endpoints`) provided as part of the service credential provides a list of all possible endpoints that can be used when connecting a client:

```
$ {
   "identity-endpoints":{
      "iam-token":"iam.cloud.ibm.com",
      "iam-policy":"iampap.cloud.ibm.com"
   },
   "service-endpoints":{
      "cross-region":{
         "us":{
            "public":{
               "us-geo":"s3.us.cloud-object-storage.appdomain.cloud",
               "Dallas":"s3.dal.us.cloud-object-storage.appdomain.cloud",
               "Washington":"s3.wdc.us.cloud-object-storage.appdomain.cloud",
               "San Jose":"s3.sjc.us.cloud-object-storage.appdomain.cloud"
            },
            "private":{
               "us-geo":"s3.private.us.cloud-object-storage.appdomain.cloud",
               "Dallas":"s3.private.dal.us.cloud-object-storage.appdomain.cloud",
               "Washington":"s3.private.wdc.us.cloud-object-storage.appdomain.cloud",
               "San Jose":"s3.private.sjc.us.cloud-object-storage.appdomain.cloud"
            },
            "direct":{
               "us-geo":"s3.direct.us.cloud-object-storage.appdomain.cloud",
               "Dallas":"s3.direct.dal.us.cloud-object-storage.appdomain.cloud",
               "Washington":"s3.direct.wdc.us.cloud-object-storage.appdomain.cloud",
               "San Jose":"s3.direct.sjc.us.cloud-object-storage.appdomain.cloud"
            }
         },
         "eu":{
            "public":{
               "eu-geo":"s3.eu.cloud-object-storage.appdomain.cloud",
               "Amsterdam":"s3.ams.eu.cloud-object-storage.appdomain.cloud",
               "Frankfurt":"s3.fra.eu.cloud-object-storage.appdomain.cloud",
               "Milan":"s3.mil.eu.cloud-object-storage.appdomain.cloud"
            },
            "private":{
               "eu-geo":"s3.private.eu.cloud-object-storage.appdomain.cloud",
               "Amsterdam":"s3.private.ams.eu.cloud-object-storage.appdomain.cloud",
               "Frankfurt":"s3.private.fra.eu.cloud-object-storage.appdomain.cloud",
               "Milan":"s3.private.mil.eu.cloud-object-storage.appdomain.cloud"
            },
            "direct":{
               "eu-geo":"s3.direct.eu.cloud-object-storage.appdomain.cloud",
               "Amsterdam":"s3.direct.ams.eu.cloud-object-storage.appdomain.cloud",
               "Frankfurt":"s3.direct.fra.eu.cloud-object-storage.appdomain.cloud",
               "Milan":"s3.direct.mil.eu.cloud-object-storage.appdomain.cloud"
            }
         },
         "ap":{
            "public":{
               "ap-geo":"s3.ap.cloud-object-storage.appdomain.cloud",
               "Tokyo":"s3.tok.ap.cloud-object-storage.appdomain.cloud",
               "Seoul":"s3.seo.ap.cloud-object-storage.appdomain.cloud",
               "Hong Kong":"s3.hkg.ap.cloud-object-storage.appdomain.cloud",
               "Sydney":"s3.syd.ap.cloud-object-storage.appdomain.cloud",
               "Osaka":"s3.osa.ap.cloud-object-storage.appdomain.cloud"
            },
            "private":{
               "ap-geo":"s3.private.ap.cloud-object-storage.appdomain.cloud",
               "Tokyo":"s3.private.tok.ap.cloud-object-storage.appdomain.cloud",
               "Seoul":"s3.private.seo.ap.cloud-object-storage.appdomain.cloud",
               "Hong Kong":"s3.private.hkg.ap.cloud-object-storage.appdomain.cloud",
            "Sydney":"s3.private.syd.ap.cloud-object-storage.appdomain.cloud",
            "Osaka":"s3.private.osa.ap.cloud-object-storage.appdomain.cloud"
         },
         "direct":{
            "ap-geo":"s3.direct.ap.cloud-object-storage.appdomain.cloud",
            "Tokyo":"s3.direct.tok.ap.cloud-object-storage.appdomain.cloud",
            "Seoul":"s3.direct.seo.ap.cloud-object-storage.appdomain.cloud",
            "Hong Kong":"s3.direct.hkg.ap.cloud-object-storage.appdomain.cloud",
            "Sydney":"s3.direct.syd.ap.cloud-object-storage.appdomain.cloud",
            "Osaka":"s3.direct.osa.ap.cloud-object-storage.appdomain.cloud"
         }
      }
   },
   "regional":{
      "us-south":{
         "public":{
            "us-south":"s3.us-south.cloud-object-storage.appdomain.cloud"
         },
         "private":{
            "us-south":"s3.private.us-south.cloud-object-storage.appdomain.cloud"
         },
         "direct":{
            "us-south":"s3.direct.us-south.cloud-object-storage.appdomain.cloud"
         }
      },
      "us-east":{
         "public":{
            "us-east":"s3.us-east.cloud-object-storage.appdomain.cloud"
         },
         "private":{
            "us-east":"s3.private.us-east.cloud-object-storage.appdomain.cloud"
         },
         "direct":{
            "us-east":"s3.direct.us-east.cloud-object-storage.appdomain.cloud"
         }
      },
      "eu-gb":{
         "public":{
            "eu-gb":"s3.eu-gb.cloud-object-storage.appdomain.cloud"
         },
         "private":{
            "eu-gb":"s3.private.eu-gb.cloud-object-storage.appdomain.cloud"
         },
         "direct":{
            "eu-gb":"s3.direct.eu-gb.cloud-object-storage.appdomain.cloud"
         }
      },
      "eu-de":{
         "public":{
            "eu-de":"s3.eu-de.cloud-object-storage.appdomain.cloud"
         },
         "private":{
            "eu-de":"s3.private.eu-de.cloud-object-storage.appdomain.cloud"
         },
         "direct":{
            "eu-de":"s3.direct.eu-de.cloud-object-storage.appdomain.cloud"
         }
      },
      "jp-tok":{
         "public":{
            "jp-tok":"s3.jp-tok.cloud-object-storage.appdomain.cloud"
         },
         "private":{
            "jp-tok":"s3.private.jp-tok.cloud-object-storage.appdomain.cloud"
         },
         "direct":{
            "jp-tok":"s3.direct.jp-tok.cloud-object-storage.appdomain.cloud"
         }
      },
      "jp-osa":{
         "public":{
          "jp-osa":"s3.jp-osa.cloud-object-storage.appdomain.cloud"
        },
        "private":{
          "jp-osa":"s3.private.jp-osa.cloud-object-storage.appdomain.cloud"
        },
        "direct":{
          "jp-osa":"s3.direct.jp-osa.cloud-object-storage.appdomain.cloud"
        }
      },
      "au-syd":{
        "public":{
          "au-syd":"s3.au-syd.cloud-object-storage.appdomain.cloud"
        },
        "private":{
          "au-syd":"s3.private.au-syd.cloud-object-storage.appdomain.cloud"
        },
        "direct":{
          "au-syd":"s3.direct.au-syd.cloud-object-storage.appdomain.cloud"
        }
      },
      "ca-tor":{
        "public":{
          "ca-tor":"s3.ca-tor.cloud-object-storage.appdomain.cloud"
        },
        "private":{
          "ca-tor":"s3.private.ca-tor.cloud-object-storage.appdomain.cloud"
        },
        "direct":{
          "ca-tor":"s3.direct.ca-tor.cloud-object-storage.appdomain.cloud"
        }
      },
      "br-sao":{
        "public":{
          "br-sao":"s3.br-sao.cloud-object-storage.appdomain.cloud"
        },
        "private":{
          "br-sao":"s3.private.br-sao.cloud-object-storage.appdomain.cloud"
        },
        "direct":{
          "br-sao":"s3.direct.br-sao.cloud-object-storage.appdomain.cloud"
        }
      }
    },
    "single-site":{
      "ams03":{
        "public":{
          "ams03":"s3.ams03.cloud-object-storage.appdomain.cloud"
        },
        "private":{
          "ams03":"s3.private.ams03.cloud-object-storage.appdomain.cloud"
        },
        "direct":{
          "ams03":"s3.direct.ams03.cloud-object-storage.appdomain.cloud"
        }
      },
      "che01":{
        "public":{
          "che01":"s3.che01.cloud-object-storage.appdomain.cloud"
        },
        "private":{
          "che01":"s3.private.che01.cloud-object-storage.appdomain.cloud"
        },
        "direct":{
          "che01":"s3.direct.che01.cloud-object-storage.appdomain.cloud"
        }
      },
      "mon01":{
        "public":{
          "mon01":"s3.mon01.cloud-object-storage.appdomain.cloud"
        },
        "private":{
               "mon01":"s3.private.mon01.cloud-object-storage.appdomain.cloud"
            },
            "direct":{
               "mon01":"s3.direct.mon01.cloud-object-storage.appdomain.cloud"
            }
         },
         "mex01":{
            "public":{
               "mex01":"s3.mex01.cloud-object-storage.appdomain.cloud"
            },
            "private":{
               "mex01":"s3.private.mex01.cloud-object-storage.appdomain.cloud"
            },
            "direct":{
               "mex01":"s3.direct.mex01.cloud-object-storage.appdomain.cloud"
            }
         },
         "sjc04":{
            "public":{
               "sjc04":"s3.sjc04.cloud-object-storage.appdomain.cloud"
            },
            "private":{
               "sjc04":"s3.private.sjc04.cloud-object-storage.appdomain.cloud"
            },
            "direct":{
               "sjc04":"s3.direct.sjc04.cloud-object-storage.appdomain.cloud"
            }
         },
         "mil01":{
            "public":{
               "mil01":"s3.mil01.cloud-object-storage.appdomain.cloud"
            },
            "private":{
               "mil01":"s3.private.mil01.cloud-object-storage.appdomain.cloud"
            },
            "direct":{
               "mil01":"s3.direct.mil01.cloud-object-storage.appdomain.cloud"
            }
         },
         "par01":{
            "public":{
               "par01":"s3.par01.cloud-object-storage.appdomain.cloud"
            },
            "private":{
               "par01":"s3.private.par01.cloud-object-storage.appdomain.cloud"
            },
            "direct":{
               "par01":"s3.direct.par01.cloud-object-storage.appdomain.cloud"
            }
         },
         "sng01":{
            "public":{
               "sng01":"s3.sng01.cloud-object-storage.appdomain.cloud"
            },
            "private":{
               "sng01":"s3.private.sng01.cloud-object-storage.appdomain.cloud"
            },
            "direct":{
               "sng01":"s3.direct.sng01.cloud-object-storage.appdomain.cloud"
            }
         }
      }
   },
   "resource-configuration-endpoints":{
      "global":{
         "public":"config.cloud-object-storage.cloud.ibm.com/v1",
         "private":"config.private.cloud-object-storage.cloud.ibm.com/v1",
         "direct":"config.direct.cloud-object-storage.cloud.ibm.com/v1"
      }
   }
}
```

Tip: When creating a client by using a library that requires an "auth" endpoint value, you need to add `/oidc/token` to end of the `iam-token` URL provided above.

Using service credentials for single-bucket access

When a service credential is created, the underlying Service ID is granted a role on the entire instance of Object Storage. If the intention that the credential be used to grant, access to a subset of buckets and not the entire instance, this policy needs to be edited. See the Bucket permissions page for more details.

Using service credentials for single-object/folder access

When a service credential is created, the underlying Service ID is granted a role on the entire instance of Object Storage. If the intention that the credential be used to grant access to a subset of buckets and not the entire instance, this policy needs to be edited. See the Assigning access to objects within a bucket using IAM access conditions page for more details.

Using an API Key for accessing multiple instances

If the intention is to use the same API key for more than one Object Storage instance, you can modify the access policy of the service ID to grant access. Each API key that is associated with a service ID inherits the policy that is assigned to the service ID.

Use the Service ID generated from an existing instance

When you create a service credential for a particular instance, that credential has an API Key value. To use that same API Key with another Object Storage instance, modify the access policy for the service id associated with that API Key to grant it access to the additional instance.

Generate a Service ID directly

If you prefer to limit visibility of the API key after its creation, follow the steps in Creating an API key for a service ID . As noted in that topic, make sure you copy or download the key at time of creation. Then, configure the access policies for the service ID to have access to one or more Object Storage instances.

API Key vs HMAC

In general IAM API Keys are the preferred method of authentication for IBM Cloud® Object Storage. HMAC is supported primarily for compatibility with an earlier version with applications which migrated from IaaS Object Storage and legacy S3 applications. IAM is also natively supported when developing applications with the COS SDKs. Token expiration and refresh are handled automatically to simplify the process.

For more information about IAM visit - Getting started with IAM

For more information about HMAC visit - Using HMAC Credentials

Assigning access to an individual bucket

Assign access roles for users and Service IDs against buckets, by using either the UI or the CLI to create policies.

Access role	Example actions
Manager	Make objects public, create, and destroy buckets and objects
Writer	Create and destroy buckets and objects
Reader	List buckets, list objects, and download objects.
Content Reader	List and download objects
Object Reader	Download objects
Object Writer	Upload objects

Buckets

Granting access to a user

If the user needs to be able to use the console and is able to see the list of all buckets within an instance, it is possible to use a custom platform access role. This allows them to view only the contents of specific buckets. If it is not appropriate for a user to read the names of other buckets then it is necessary to design and implement a custom portal or other user interface using the API.

If the user interacts with data by using the API and doesn't require console access, *and* they are a member of your account, you can grant access to a single bucket without any access to the parent instance using the default roles.

Policy enforcement

IAM policies are enforced hierarchically from greatest level of access to most restricted. Conflicts are resolved to the more permissive policy. For example, if a user has both the `Writer` and `Reader` service access role on a bucket, the policy granting the `Reader` role is ignored.

This is also applicable to service instance and bucket level policies.

- If a user has a policy granting the `Writer` role on a service instance and the `Reader` role on a single bucket, the bucket-level policy is ignored.
- If a user has a policy granting the `Reader` role on a service instance and the `Writer` role on a single bucket, both policies are enforced and the more permissive `Writer` role will take precedence for the individual bucket.

If it is necessary to restrict access to a single bucket (or set of buckets), ensure that the user or Service ID doesn't have any other instance level policies by using either the console or CLI.

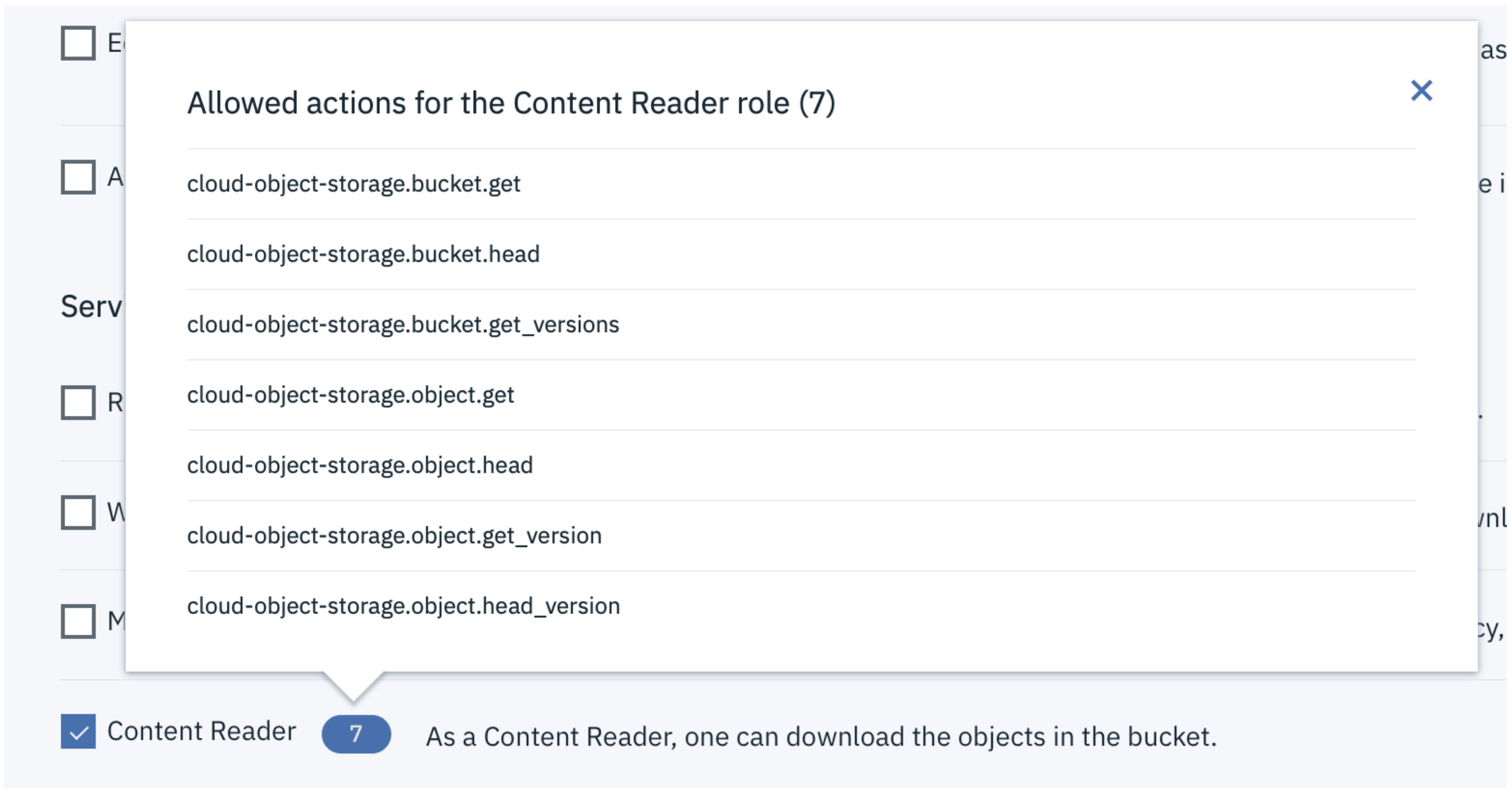
See [Best practices for organizing resources and assigning access](#) to learn more.

Create a new policy for a user

To create a new bucket-level policy:

1. Navigate to the **Access IAM** console from the **Manage** menu.
2. Select **Users** from the left navigation menu.
3. Select a user.
4. Select the **Access Policies** tab to view the user's existing policies, assign a new policy, or edit an existing policy.
5. Click **Assign access** to create a new policy.
6. Choose **Assign access to resources**.
7. First, select **Cloud Object Storage** from the services menu.
8. Then, select the appropriate service instance. Enter `bucket` in the **Resource type** field and the bucket name in the **Resource ID** field.
9. Select the wanted service access role. Selecting the lozenge with the number of actions show the actions available to the role, as exemplified for "Content Reader" in Figure 1.
10. Click **Assign**

Example actions per Content Reader role



Allowed actions for the Content Reader role (7)

cloud-object-storage.bucket.get

cloud-object-storage.bucket.head

cloud-object-storage.bucket.get_versions

cloud-object-storage.object.get

cloud-object-storage.object.head

cloud-object-storage.object.get_version

cloud-object-storage.object.head_version

☑ Content Reader 7 As a Content Reader, one can download the objects in the bucket.

Tip: Note that leaving the **Resource Type** or **Resource** fields blank will create an instance-level policy.

Create a new policy for a user CLI command

From a terminal run the following command:

```
ibmcloud iam user-policy-create <user-name> \
      --roles <role> \
      --service-name cloud-object-storage \
      --service-instance <resource-instance-id> \
      --resource-type bucket \
      --resource <bucket-name>
```

To list existing policies:

```
ibmcloud iam user-policies <user-name>
```

To edit an existing policy:

```
ibmcloud iam user-policy-update <user-name> <policy-id> \
      --roles <role> \
      --service-name cloud-object-storage \
      --service-instance <resource-instance-id> \
      --resource-type bucket \
      --resource <bucket-name>
```

Granting access to a Service ID

If you need to grant access to a bucket for an application or other non-human entity, use a Service ID. The Service ID can be created specifically for this purpose, or can be an existing Service ID already in use.

Create a new policy for a user

1. Navigate to the **Access (IAM)** console from the **Manage** menu.
2. Select **Service IDs** from the left navigation menu.
3. Select a Service ID to view any existing policies, and assign a new policy or edit an existing policy.
4. Select the service instance, service ID, and desired role.
5. Enter `bucket` in the **Resource Type** field and the bucket name in the **Resource** field.
6. Click **Submit**

Tip: Note that leaving the **Resource Type** or **Resource** fields blank will create an instance-level policy.

Create a new policy for a Service ID

From a terminal run the following command:

```
ibmcloud iam service-policy-create <service-id-name> \
      --roles <role> \
      --service-name cloud-object-storage \
      --service-instance <resource-instance-id> \
      --resource-type bucket \
      --resource <bucket-name>
```

To list existing policies:

```
ibmcloud iam service-policies <service-id-name>
```

To edit an existing policy:

```
ibmcloud iam service-policy-update <service-id-name> <policy-id> \
      --roles <role> \
      --service-name cloud-object-storage \
      --service-instance <resource-instance-id>
      --resource-type bucket \
      --resource <bucket-name>
```

Assigning access to objects within a bucket using IAM access conditions

IAM access policies allow granting permissions in a Cloud Object Storage bucket to specific groups of objects. This approach allows for fine-grained access control over data access, making it useful in scenarios where different parts of a bucket need to be accessed by different users or applications.

⚠ **Important:** If access is required to the entire bucket (that is, when fine-grained access control is not required) then follow the information on Assigning access to an individual bucket.

Each object that is stored in a Cloud Object Storage bucket has a unique key, and these keys often follow a hierarchical structure similar to a file system.

Related links to IAM access policies

- Resource attribute-based conditions
- String comparisons
- Checking a policy version in the console

Example

- An individual object with the key *"folder1/subfolder1/file.txt"* can simulate a folder or directory hierarchy, where the directory *"folder1"* contains a subdirectory that is named *"sub-folder1"* containing a file *"file.txt"*. Access can be assigned at any folder level.
- An access policy can be created for all objects and - in the folder named *"folder1"*, or access can be assigned for just objects in the subdirectory named *"subfolder1"*.

A policy administrator can assign access to individual objects and folders by configuring conditions when creating IAM access policies. The next section describes how to construct these types of policies.

Constructing a fine-grained access control policy

The first step to granting access to individual objects within a bucket is to construct an IAM policy. You can find more information on constructing an IAM access policy for Object Storage in the Cloud Object Storage tutorial Limiting access to a single Object Storage bucket. For more general information on building IAM policies, go to How IBM Cloud IAM works. The following items are key components of building an IAM access policy for your Object Storage resources.

Subject

The subject of an access policy can be an individual user, an access group, a Service ID, or a Trusted Profile. See What are IAM policies and who can assign them? for more information on the types of subjects you can apply to a policy.

Service

The service is the IBM Cloud Service that contains the resource you are trying to assign access to. For assigning access to individual objects, use the Cloud Object Storage service.

Resource

IBM Cloud® Object Storage supports the following resource targets:

- Resource group ID
- Service instance
- Resource type with value of *"bucket"*
- Resource ID (bucket name)

Role

IBM Cloud access roles are groups of actions. Access roles allow the subject to complete specific tasks within the context of the target resources that are defined in the policy. Cloud Object Storage supports several pre-defined service roles that make assigning permissions easier. Cloud Object Storage also allows the creation of custom roles. For more information on the supported roles for Cloud Object Storage, see Identity and Access Management roles.

For the list of Cloud Object Storage roles and their interaction with conditions, go to this table.

Condition

When a resource is identified, a condition can be used to further scope access for a subject to individual objects in a bucket, which is referred to as fine-grained access control.

A single IAM Policy can have more than one condition by using an `OR` or `AND` statement to combine the conditions. The condition statement (containing one or more conditions) should evaluate to `TRUE` for the user request to be permitted to perform the action. IAM Policy will deny any action that does not get to be evaluated `TRUE`. The policy statement should contain all condition attributes required by the role. If the policy statement does not contain all condition attributes required by the role, the actions subject to the omitted condition attributes will be denied.

Important: Use a policy with no condition attributes to give full access, as defined by the role, to the target resource.

Tip: Use IAM v2 policy to construct IAM policy with attribute-based conditions.

Using conditions in an IAM policy

Cloud Object Storage supports the following attributes to specify conditions for assigning fine-grained access on Cloud Object Storage resources:

Prefix and Delimiter

Prefix and **Delimiter** are used together to scope all listing permissions to specific objects in a bucket.

- The **Prefix** condition attribute defines the prefix for the set of object keys that this condition should allow for listing of objects or folders. For example, in the object named *"folder1/subfolder1/file.txt"*, both *"folder1/"* and *“folder1/subfolder1/”* are possible prefixes.
- A **Delimiter** helps the user navigate the bucket as if it was a file hierarchy. Assigning a delimiter condition statement restricts the type of folder structure that the user can generate in the listing. In an object named *"folder1/subfolder1/file.txt"*, the delimiter *“/”* can be used to simulate a folder hierarchy where each folder is separated by a *“/”*. If a condition statement allows only a delimiter of *“/”*, then a list request with any other delimiter value is not permitted.

Typically the prefix and delimiter are used together in a condition statement with an `AND` operator. It is possible to use a prefix without a delimiter in a condition statement. If the policy is configured with only a prefix and not a delimiter condition statement, the user can use any or no delimiter to list the objects.

Examples of using Prefix and Delimiter condition statements

Consider the object named *"folder1/subfolder1/file.txt"*:

Prefix of *"folder1/"* `AND` no Delimiter

- You can return a list of every object that starts with *folder1/* by doing a list request on *folder1/* and not providing a delimiter.
- If you use a delimiter of *"/"* in the list request, they'd be restricted to only seeing the first level of objects and sub-folders in *folder1/*.
- If user tries to list the subfolder (requests to list prefix = *“folder1/subfolder1/”*), access is denied.

Prefix of *"folder1/"* `AND` Delimiter of *"/"*

- You can only list the objects and sub-folders in the first level of *folder1*.
- You can only do list requests that specify a delimiter of *"/"*.
- If you try to list the contents of *subfolder1*, access is denied (user would need to have a condition of allowing prefix = *“folder1/subfolder1/"* to allow listing the contents of *subfolder1*).

The following APIs are subject to Prefix/Delimiter conditions:

- GET Bucket (List Objects)
- GET Bucket Object Versions (List Object Versions)
- List Multipart Uploads

Tip: To see the full list of actions and the supported condition attributes, see Identity and Access Management actions.

Tip: To give a fine-grained user access to navigate to their folder in the UI, the user needs access to list the root folder of the bucket. See Scenario 4 for how to construct the policy to enable this.

Path

Path is used to scope all read, write, or management access on specific objects.

For an object named *"folder1/subfolder1/file.txt"*, the full object key is the path. To restrict read, write or management actions to this object, define a condition with Path of *"folder1/subfolder1/file.txt"*.

All Cloud Object Storage APIs that act directly on an object are subject to Path conditions. See Identity and Access Management actions for the list of Cloud Object Storage API actions that support Path.

Note: It is recommended that you define both a Prefix/Delimiter condition and a Path condition when granting read, write or list actions to a user in the same policy. `Manager`, `Writer`, `Reader`, and `Content Reader` are examples of roles where it is recommended to define both a Prefix/Delimiter and Path condition. See Scenario 4 for how to construct the policy to enable this. A condition specifying Prefix/Delimiter and a condition specifying Path should be logically `ORed` in the IAM Policy statement to permit both types of operations (read, write or management of objects or list objects).

Operators used with condition attributes

There are several operators that can be used when defining condition attributes. The full list of operators that can be used for prefix, delimiter, and path condition attributes can be found in Resource attribute-based conditions.

Use of wildcards

A condition attribute's values can include a wildcard when the operator is `stringMatch` or `stringMatchAnyOf`.

Examples of using wildcards in condition statements:

Consider the object named *"folder1/subfolder1/file.txt"*:

Path of *"folder1/*"*

- You will get read, write, or management access, as defined by the role, to all objects that start with *"folder1/"*.

Prefix of *"folder1/*"* `AND` no Delimiter

- For an object list request with prefix set to *"folder1/"* and no delimiter, the user request returns all objects that start with *"folder1/"*.
- For an object list request with prefix set to *"folder1/"* and delimiter of *"/"*, the request returns a view of the objects and folders just in the first level of *folder1*.
- For an object list request with prefix set to "folder1/subfolder1/" and delimiter of *"/"*, the request returns a view of the objects and folders just in the first level of *folder1/subfolder1*.

Prefix of *"folder1/*"* `AND` Delimiter of *"/"*

- For an object list request with prefix set to *"folder1/"* and delimiter of *"/"*, the request returns a view of the objects and folders just in the first level of *folder1*.
- For an object list request with prefix set to "folder1/subfolder1/" and delimiter of *"/"*, the request returns the objects (and any sub-folders) in *folder1/subfolder1*.
- For an object list request with prefix set to *"folder1/"* and no delimiter, the request will not be permitted since a delimiter of *"/"* must be used in the list request for this policy to evaluate to true.

Actions that do not use a Prefix/Delimiter or Path

There are some Cloud Object Storage APIs that do not specify a path or prefix and delimiter. The Cloud Object Storage Service roles: `Manager`, `Writer`, `Reader`, and `Content Reader` are examples of roles that contain these actions. This also applies to custom roles. To allow these actions when using a **Prefix/Delimiter** or **Path** condition, the following condition statement is needed in the IAM policy:

```
$ ((path stringExists = false) AND (prefix stringExists = false) AND (delimiter stringExists= false))
```

See the Identity and Access Management actions table for the full list of API actions that do not support **Prefix/Delimiter** or **Path** conditions and require the statement above when using fine-grained access.

Refer to the section on how to Create a new policy for a user with conditions for using this clause in an IAM policy.

Use of conditions with Cloud Object Storage service roles

Access role	Description of actions	Supported Condition Attributes
Manager	Make objects public, create, and destroy buckets and objects.	See Note
Writer	Create and destroy buckets and objects.	See Note
Reader	List buckets, list objects, and download objects.	See Note
Content Reader	List and download objects.	See Note

Object Reader	Download objects.	Path
Object Writer	Upload objects.	Path

Use of Conditions with COS Service Roles

Note: These roles support Prefix/Delimiter and Path condition attributes. The roles also include actions that do not specify a path or prefix and delimiter. Use the `stringExists` clause in the condition statement to allow these actions.

See link for the full list of actions for each Cloud Object Storage service role and the list of condition attributes that are supported by each action.

Create a new policy for a user with conditions

The following example provides a user with the `“Writer”` Cloud Object Storage Service Role with the ability to:

1. List access to the full object hierarchy within a folder named *"folder1/subfolder1"*.
2. Read, write, or delete access to all objects in a folder named *"subfolder1"*.
3. Perform bucket configuration management such as `HEAD Bucket` and `GET/PUT Bucket Versioning`.

CLI of an IAM policy with a condition

Tip: For general information on how to use the CLI, see the section on using the IBM Cloud® Command Line Interface .

Example

```
policy.json:

{
"type": "access",
  "subject": {
    "attributes": [
      {
        "value": "IBMid-664001QJNU",
        "operator": "stringEquals",
        "key": "iam_id"
      }
    ]
  },
  "resource": {
    "attributes": [
      {
        "value": "cloud-object-storage",
        "operator": "stringEquals",
        "key": "serviceName"
      },
      {
        "value": "e6156134-5ed7-4f73-80d3-d6d1ef56f1f9",
        "operator": "stringEquals",
        "key": "serviceInstance"
      },
      {
        "value": "bucket",
        "operator": "stringEquals",
        "key": "resourceType"
      },
      {
        "value": "fgac-tf-test",
        "operator": "stringEquals",
        "key": "resource"
      }
    ]
  },
  "control": {
    "grant": {
      "roles": [
        {
          "role_id": "crn:v1:bluemix:public:iam::::serviceRole:Writer"
        }      ]
    }
  },
  "rule": {
    "operator": "or",
    "conditions": [
      {
        "operator": "and",
        "conditions": [
          {
            "key": "{{resource.attributes.prefix}}",
            "operator": "stringMatch",
            "value": "folder1/subfolder1/*"
          },
          {
            "key": "{{resource.attributes.delimiter}}",
            "operator": "stringEqualsAnyOf",
            "value": [
              "/",
              ""
            ]
          }
        ]
      },
      {
        "key": "{{resource.attributes.path}}",
        "operator": "stringMatch",
        "value": "folder1/subfolder1/*"
      },
      {
        "operator": "and",
        "conditions": [
          {
            "key": "{{resource.attributes.delimiter}}",
            "operator": "stringExists",
            "value": false
          },
          {
            "key": "{{resource.attributes.prefix}}",
            "operator": "stringExists",
            "value": false
          },
          {
            "key": "{{resource.attributes.path}}",
            "operator": "stringExists",
            "value": false
          }
        ]
      }
    ]
  },
  "pattern": "attribute-based-condition:resource:literal-and-wildcard"
}
ibmcloud iam  user-policy-create hello@ibm.com --file policy.json --api-version v2
```

Note: Use --api-version v2 with resource-based attribute conditions.

API of an IAM policy with a condition

Example

Request

```
curl -X POST 'https://iam.cloud.ibm.com/v2/policies' \
-H 'Authorization: Bearer $TOKEN' \
-H 'Content-Type: application/json' \
-d '{
  "type": "access",
  "description": "access control for RESOURCE_NAME",
  "resource": {
    "attributes": [
      {
        "key": "serviceName",
        "operator": "stringEquals",
        "value": "cloud-object-storage"
      },
      {
        "key": "serviceInstance",
        "operator": "stringEquals",
        "value": "$SERVICE_INSTANCE"
      },
      {
        "key": "accountId",
        "operator": "stringEquals",
        "value": "$ACCOUNT_ID"
      },
      {
        "key": "resourceType",
        "operator": "stringEquals",
        "value": "bucket"
      },
      {
        "key": "resource",
        "operator": "stringEquals",
        "value": "$RESOURCE_NAME"
      }
    ]
  },
  "subject": {
    "attributes": [
      {
        "key": "iam_id",
        "operator": "stringEquals",
        "value": "IBMid-123453user"
      }
    ]
  },
  "control": {
    "grant": {
      "roles": [
        {
          "role_id": "crn:v1:bluemix:public:iam::::serviceRole:Writer"
        }
      ]
    }
  },
  "rule": {
    "operator": "or",
    "conditions": [
      {
        "operator": "and",
        "conditions": [
          {
            "key": "{{resource.attributes.prefix}}",
            "operator": "stringMatch",
            "value": "folder1/subfolder1/*"
          },
          {
            "key": "{{resource.attributes.delimiter}}",
            "operator": "stringEqualsAnyOf",
            "value": [
              "/",
              ""
            ]
          }
        ]
      },
      {
        "key": "{{resource.attributes.path}}",
        "operator": "stringMatch",
        "value": "folder1/subfolder1/*"
      },
      {
        "operator": "and",
        "conditions": [
          {
            "key": "{{resource.attributes.delimiter}}",
            "operator": "stringExists",
            "value": false
          },
          {
            "key": "{{resource.attributes.prefix}}",
            "operator": "stringExists",
            "value": false
          },
          {
            "key": "{{resource.attributes.path}}",
            "operator": "stringExists",
            "value": false
          }
        ]
      }
    ]
  },
  "pattern": "attribute-based-condition:resource:literal-and-wildcard"
}'
```

Response

```
{
    "id": "d4078e99-d78a-4a50-95d6-b528e3c87dff",
    "description": "access control for RESOURCE_NAME",
    "type": "access",
    "subject": {
        "attributes": [
            {
                "key": "iam_id",
                "operator": "stringEquals",
                "value": "IBMid-123453user"
            }
        ]
    },
    "resource": {
        "attributes": [
            {
                "key": "serviceName",
                "operator": "stringEquals",
                "value": "cloud-object-storage"
            },
            {
                "key": "serviceInstance",
                "operator": "stringEquals",
                "value": "$SERVICE_INSTANCE"
            },
            {
                "key": "accountId",
                "operator": "stringEquals",
                "value": "$ACCOUNT_ID"
            },
            {
                "key": "resourceType",
                "operator": "stringEquals",
                "value": "bucket"
            },
            {
                "key": "resource",
                "operator": "stringEquals",
                "value": "$RESOURCE_NAME"
            }
        ]
    },
    "pattern": "attribute-based-condition:resource:literal-and-wildcard",
    "rule": {
        "operator": "or",
        "conditions": [
            {
                "operator": "and",
                "conditions": [
                    {
                        "key": "{{resource.attributes.prefix}}",
                        "operator": "stringMatch",
                        "value": "folder1/subfolder1/*"
                    },
                    {
                        "key": "{{resource.attributes.delimiter}}",
                        "operator": "stringEqualsAnyOf",
                        "value": [
                            "/",
                            ""
                        ]
                    }
                ]
            },
            {
                "key": "{{resource.attributes.path}}",
                "operator": "stringMatch",
                "value": "folder1/subfolder1/*"
            },
            {
                "operator": "and",
                "conditions": [
                    {
                        "key": "{{resource.attributes.delimiter}}",
                        "operator": "stringExists",
                        "value": false
                    },
                    {
                        "key": "{{resource.attributes.prefix}}",
                        "operator": "stringExists",
                        "value": false
                    },
                    {
                        "key": "{{resource.attributes.path}}",
                        "operator": "stringExists",
                        "value": false
                    }
                ]
            }
        ]
    },
    "control": {
        "grant": {
            "roles": [
                {
                    "role_id": "crn:v1:bluemix:public:iam::::serviceRole:Writer"
                }
            ]
        }
    },
    "href": "https://iam.test.cloud.ibm.com/v2/policies/d4078e99-d78a-4a50-95d6-b528e3c87dff",
    "created_at": "2023-10-09T16:24:52.391Z",
    "created_by_id": " IBMid-12345user",
    "last_modified_at": "2023-10-09T16:24:52.391Z",
    "last_modified_by_id": " IBMid-12345user",
    "counts": {
        "account": {
            "current": 785,
            "limit": 4020
        },
        "subject": {
            "current": 1,
            "limit": 1000
        }
    },
    "state": "active",
    "version": "v1.0"
}
```

Terraform of an IAM policy with a condition

Example

```
data "ibm_resource_group" "cos_group" {
      name = "Default"
}

resource "ibm_iam_user_policy" "example" {
 ibm_id = “user_email_id”
 roles = ["Writer"]
 resources {
  service = "cloud-object-storage"
  resource_type = "bucket"
  resource_instance_id = "cos instance guid"
  resource = "bucket-name"
}


rule_conditions {
 operator = "and"
 conditions {
  key = "{{resource.attributes.prefix}}"
  operator = "stringMatch"
  value = ["folder1/subfolder1/*"]
 }
 conditions {
  key = "{{resource.attributes.delimiter}}"
  operator = "stringEqualsAnyOf"
  value = ["/",""]
 }
 }
rule_conditions {
  key = "{{resource.attributes.path}}"
  operator = "stringMatch"
  value = ["folder1/subfolder1/*"]
 }
rule_conditions {
 operator = "and"
  conditions {
   key = "{{resource.attributes.delimiter}}"
   operator = "stringExists"
   value = ["false"]
  }
  conditions {
   key = "{{resource.attributes.prefix}}"
   operator = "stringExists"
   value = ["false"]
  }
  conditions {
   key = "{{resource.attributes.path}}"
   operator = "stringExists"
   value = ["false"]
  }
 }
rule_operator = "or"
 pattern = "attribute-based-condition:resource:literal-and-wildcard"
}
```

Additional information

For examples on how to use **Prefix/Delimiter**, or **Path** condition attributes, see the tutorial on controlling access to individual objects in a bucket.

Allowing public access

Sometimes data is meant to be shared. Buckets might hold open data sets for academic and private research or image repositories that are used by web applications and content delivery networks. Make these buckets accessible using the **Public Access** group.

Tip: There are three IAM roles that can be used for public access to a bucket: `Administrator`, `ContentReader`, and `ObjectReader`. The difference between them is that the `Administrator` and `ContentReader` can list the objects in a bucket, which is useful for applications that require ease of listing (for example, a web UI) in addition to reading objects. For more information, see the IAM reference documentation.

Using the console to set public access

First, make sure that you have a bucket. If not, follow the getting started tutorial to become familiar with the console.

Enable public access

1. From the IBM Cloud console dashboard, select **Storage** to view your resource list.
2. Next, select the service instance with your bucket from within the **Storage** menu. This takes you to the Object Storage Console.
3. Choose the bucket that you want to be publicly accessible. Keep in mind this policy makes *all objects in a bucket* available to download for anyone with the appropriate URL.
4. Select **Access policies** from the navigation menu.
5. Select the **Public access** tab.
6. Click **Create access policy**. After you read the warning, choose **Enable**.
7. Now all objects in this bucket are publicly accessible!

Disable public access

1. From anywhere in the IBM Cloud console, select the **Manage** menu, and the **Access (IAM)**.
2. Select **Access groups** from the navigation menu.
3. Select **Public Access** to see a list of all public access policies currently in use.
4. Find the policy that corresponds to the bucket you want to return to enforced access control.
5. From the list of actions on the far right of the policy entry, choose **Remove**.
6. Confirm the dialog box, and the policy is now removed from the bucket.

Allowing public access on individual objects

To make an object publicly accessible through the REST API, an `x-amz-acl: public-read` header can be included in the request. Setting this header bypasses any IAM policy checks and allow for unauthenticated `HEAD` and `GET` requests. For more information about endpoints, see Endpoints and storage locations.

Additionally, HMAC credentials make it possible to allow temporary public access that uses pre-signed URLs.

Upload a public object

```
curl -X "PUT" "https://{endpoint}/{bucket-name}/{object-name}" \
     -H "x-amz-acl: public-read" \
     -H "Authorization: Bearer {token}" \
     -H "Content-Type: text/plain; charset=utf-8" \
     -d "{object-contents}"
```

Allow public access to an existing object

Using the query parameter `?acl` without a payload and the `x-amz-acl: public-read` header allows public access to the object without needing to overwrite the data.

```
curl -X "PUT" "https://{endpoint}/{bucket-name}/{object-name}?acl" \
     -H "x-amz-acl: public-read" \
     -H "Authorization: Bearer {token}"
```

Make a public object private again

Using the query parameter `?acl` without a payload and an empty `x-amz-acl:` header revokes public access to the object without needing to overwrite the data.

```
curl -X "PUT" "https://{endpoint}/{bucket-name}/{object-name}?acl" \
     -H "Authorization: Bearer {token}" \
     -H "x-amz-acl:"
```

Static websites

While IBM Cloud Object Storage doesn't support automatic static website hosting, it's possible to manually configure a web server and use it to serve publicly accessible content hosted in a bucket. For more information, see the overview of static website options.

Restricting access by network context

Context-based restrictions provide a way for administrators to limit access to resources. What if certain data must be accessed from trusted networks only? A properly configured policy restricts all access to data unless the request originates from an approved network zone and endpoint type (public, private, or direct).

Note: This feature is not currently supported in Object Storage for Satellite. Learn more.

Using context-based restrictions

A context-based restriction is comprised of a **rule** and one or more **contexts** (network zones and/or endpoint type). These restrictions do not replace IAM policies, but simply check that a request is coming from an allowed context, such as a range of IP addresses, VPCs, or service references.

A user must have the `Administrator` role on a service to create, update, or delete rules. A user must have either the `Editor` or `Administrator` role to create, update, or delete network zones.

Context-based restrictions does not support applying context-based restrictions rules to specific objects or folders, only at the bucket level.

You can learn more about how context-based restrictions work in the detailed documentation, or you can follow a quick tutorial.

Tip: Audit log events generated will come from the context-based restrictions service, and not Object Storage.

Important: If no rules are applicable to a particular resource, access is determined by IAM policies and the presence of a legacy bucket firewall.

Important: Context-based restrictions are only applied at the bucket level and not to specific objects or folders.

An account is limited in the number of rules and network zones that can be supported.

Bucket firewalls versus context-based restrictions

Important: Prior to the availability of context-based restrictions, Object Storage itself would enforce access restrictions based on IP addresses. While this method is still supported, it is recommended to use the newer context-based restrictions instead of the legacy bucket firewall.

Bucket firewalls and context-based restrictions operate independently of one another, which means it's possible to have a request permitted by one and denied by the other.

- Bucket creation requests **must** be permitted by any context-based restrictions.
- For all other bucket or object requests, both the context-based restrictions *and* the bucket firewall must allow the request.

Tip: An IP address that is allowed by context-based restrictions can still be denied by the bucket firewall.

About legacy bucket firewalls

There are some rules around setting a firewall:

- A user that sets or views a firewall must have the `Manager` role on the bucket.
- A user with the `Manager` role on the bucket can view and edit the list of allowed IP addresses from any IP address to prevent accidental lockouts.
- The Object Storage Console can still access the bucket, provided the user's IP address is authorized.
- Other IBM Cloud services **are not authorized** to bypass the firewall. This limitation means that other services that rely on IAM policies for bucket access (such as Aspera, SQL Query, Security Advisor, Watson Studio, Cloud Functions, and others) will be unable to do so.

Important: When a firewall is set, the bucket is isolated from the rest of IBM Cloud. Consider how this may impact applications and workflows that depend on other services directly accessing a bucket before enabling the firewall. This can be avoided by using service references and context-based restrictions instead.

Note: Access from a VPC environment can pass `allowed_network_type` checks, and VPC-zone underlay IP addresses can be added to the `allowed_ip` list. It is not possible to restrict access to an overlay IP for an individual VPC VSI or bare-metal server.

First, make sure that you have an instance of Object Storage and have provisioned at least one bucket. If not, follow the getting started tutorial to obtain the prerequisites and become familiar with the console.

Set a list of authorized IP addresses using a legacy firewall

1. Start by selecting **Storage** to view your resource list.
2. Next, select the service instance with your bucket from within the **Storage** menu. This takes you to the Object Storage Console.
3. Choose the bucket that you want to limit access to authorized IP addresses.
4. Select **Access policies** from the navigation menu.
5. Select the **Authorized IPs** tab.
6. Click **Add IP addresses**, then choose **Add**.
7. Specify a list of IP addresses in CIDR notation, for example `192.168.0.0/16, fe80:021b::0/64`. Addresses can follow either IPv4 or IPv6 standards.
8. Click **Add**.
9. The firewall will not be enforced until the address is saved in the console. Click **Save all** to enforce the firewall.
10. Note that all objects in this bucket are only accessible from those IP addresses.

Remove any IP address restrictions using a legacy firewall

1. From the **Authorized IPs** tab, check the boxes next to any IP addresses or ranges to remove from the authorized list.
2. Select **Delete**, and then confirm the dialog box by clicking **Delete** again.
3. The updated list won't be enforced until the changes are saved in the console. Click **Save all** to enforce the new rules.
4. Now all objects in this bucket are only accessible from these IP addresses!

Note: If there are no authorized IP addresses listed this means that normal IAM policies will apply to the bucket, with no restrictions on the user's IP address, unless there are context-based restrictions in place.

Set a legacy firewall through an API

Firewalls are managed with the COS Resource Configuration API. This new REST API is used for configuring buckets.

Tip: Users with the `manager` role can view and edit the list of allowed IP addresses from any network in order to prevent accidental lockouts.

Managing access using Access/Secret Key (HMAC) authentication

Using HMAC credentials

HMAC credentials consist of an Access Key and Secret Key paired for use with S3-compatible tools and libraries that require authentication.

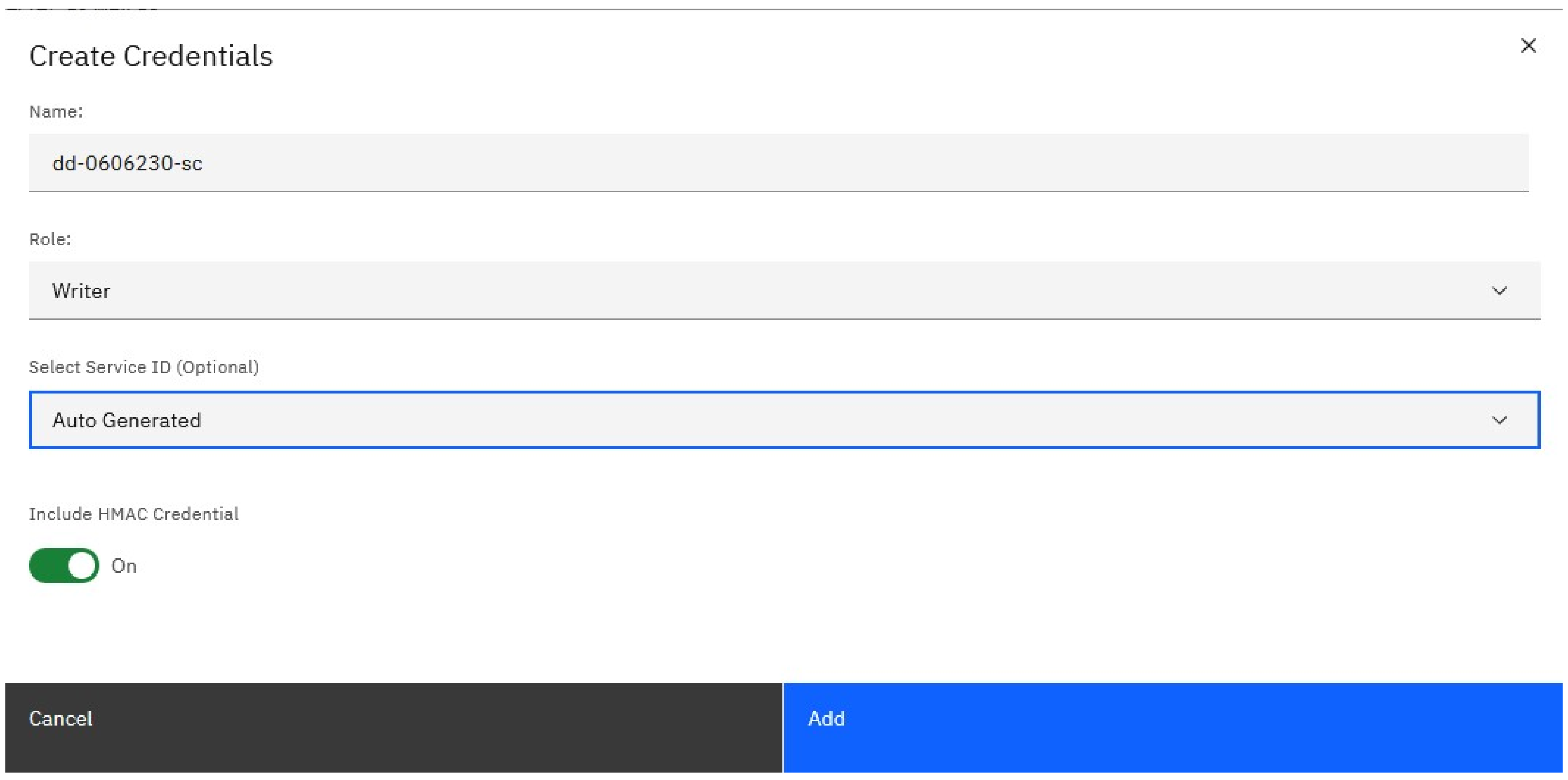
HMAC credentials defined

The IBM Cloud® Object Storage API is a REST-based API for reading and writing objects. It uses IBM Cloud® Identity and Access Management for authentication and authorization, and supports a subset of the S3 API for easy migration of applications to IBM Cloud.

Create HMAC credentials in the console

Users can create a set of HMAC credentials as part of a Service Credential by switching the `Include HMAC Credential` to `On` during credential creation in the console. Figure 1 shows the option for setting the HMAC parameter by choosing "Advanced options."

HMAC setting from advanced options


Create Credentials

Name:

dd-0606230-sc

Role:

Writer

Select Service ID (Optional)

Auto Generated

Include HMAC Credential

On

After the Service Credential is created, the HMAC Key is included in the `cos_hmac_keys` field. These HMAC keys are then associated with a Service ID and can be used to access any resources or operations that are allowed by the Service ID's role.

Tip: When creating a service credential, it is possible to provide a value of `None` for the role. This will prevent the creation of unintended or unnecessary IAM access policies. Any access policies for the associated service ID will need to be managed using the IAM console or APIs.

Create HMAC credentials using the CLI

You can also use the IBM Cloud® Object Storage CLI to create your credentials. You must have the already installed the IBM Cloud Platform Command Line Instructions before you can use the example.

```
$ ibmcloud resource service-key-create <key-name-without-spaces> Writer --instance-name "<instance name--use quotes if your
instance name has spaces>" --parameters '{"HMAC":true}'
```

An example of HMAC credentials

If you want to store the results of the generated key, you can append `> file.skey` to the end of the example. For the purposes of this instruction set, you need only find the `cos_hmac_keys` heading with child keys, `access_key_id`, and `secret_access_key`.

```
    cos_hmac_keys:
        access_key_id:      7exampledonotusea6440da12685eee02
        secret_access_key:  8not8ed850cddbece407exampledonotuse43r2d2586
```

Setting HMAC credentials as environment variables

Once you have created your credentials, you can set them as environment variables (the instructions for which are specific to the operating system involved). For instance, in Example 3, a `.bash_profile` script contains `COS_HMAC_ACCESS_KEY_ID` and `COS_HMAC_SECRET_ACCESS_KEY` that is exported upon starting a shell and used in development.

```
$ export COS_HMAC_ACCESS_KEY_ID="7exampledonotusea6440da12685eee02"
export COS_HMAC_SECRET_ACCESS_KEY="8not8ed850cddbece407exampledonotuse43r2d2586"
```

Next steps

Note that when using HMAC credentials to create signatures to use with direct REST API calls that extra headers are required:

1. All requests must have an `x-amz-date` header with the date in `%Y%m%dT%H%M%SZ` format.
2. Any request that has a payload (object uploads, deleting several objects, and so on) must provide a `x-amz-content-sha256` header with an SHA256 hash of the payload contents.
3. ACLs (other than `public-read`) are unsupported.

Tip: Not all S3-compatible tools are currently supported. Some tools attempt to set ACLs other than `public-read` on bucket creation. Bucket creation through these tools will fail. If a `PUT bucket` request fails with an unsupported ACL error, first use the console as shown in the [getting started with IBM Cloud Object Storage](#) to create the bucket, then configure the tool to read and write objects to that bucket. Tools that set ACLs on object writes are not currently supported.

Constructing an HMAC signature

Instead of token-based authorization, it's possible to use [HMAC credentials](#).

These credentials are used to create an authorization header analogous to AWS Signature Version 4. Calculating signatures provides identity verification and in-transit data integrity. Each signature is tied to the time stamp of the request, so you can't reuse authorization headers. The header is composed of four components: an algorithm declaration, credential information, signed headers, and the calculated signature.

```
$ AWS4-HMAC-SHA256 Credential={access-key}/{date}/{region}/s3/aws4_request,SignedHeaders=host;x-amz-date;{other-required-headers},Signature={signature}
```

The date is provided in `YYYYMMDD` format, and the region corresponds to the location of specified bucket, for example `us`. The `host` and `x-amz-date` headers are always required. Depending on the request other headers might be required as well (for example, `x-amz-content-sha256` in requests with payloads). It's necessary to recalculate the signature for every individual request, so many developers prefer to use a tool or SDK that produces the authorization header automatically.

Creating an `authorization` header

First, we need to create a request in a standardized format.

1. Declare which HTTP method we are using (for example, `PUT`)
2. Define the resource that we are accessing in a standardized fashion. This is the part of the address between `http(s)://` and the query string. For requests at the account level (such as listing buckets) this is simply `/`.
3. If there are any request parameters, they must be standardized by being percent-encoded (for example, spaces are be represented as `%20`) and alphabetized.
4. Headers need to be standardized by removing white space, converting to lowercase, and adding a newline to each, then they must be sorted in ASCII order.
5. After being listed in a standard format, they must be 'signed'. This is taking just the header names, not their values, and listing them in alphabetical order, which is separated by semicolons. `Host` and `x-amz-date` are required for all requests.
6. If the request has a body, such as when uploading an object or creating a new ACL, the request body must be hashed by using the SHA-256 algorithm and represented as base-16 encoded lowercase characters.
7. Combine the HTTP method, standardized resource, standardized parameters, standardized headers, signed headers, and hashed request body each separated by a newline to form a standardized request.

Next, we need to assemble a 'string-to-sign' that is combined with the signature key to form the final signature. The string-to-sign takes the following form:

```
$ AWS4-HMAC-SHA256
{time}
{date}/{string}/s3/aws4_request
{hashed-standardized-request}
```

1. The time must be current Coordinated Universal Time and formatted according to the ISO 8601 specification (for example, `20161128T152924Z`).
2. The date is in `YYYYMMDD` format.
3. The final line is the previously created standardized request hashed that uses the SHA-256 algorithm.

Now we need to calculate the signature.

1. First, the signature key needs to be calculated from the account's secret access key, the current date, and the region and API type being used.
2. The string `AWS4` is added as a prefix to the secret access key, and then that new string is used as the key to hash the date.
3. The resulting hash is used as the key to hash the region.
4. The process continues with the new hash being used as the key to hash the API type.
5. Finally, the newest hash is used as the key to hash the string `aws4_request` creating the signature key.
6. The signature key is then used as the key to hash the string-to-sign generating the final signature.

Now the only step left is assembling the `authorization` header as shown:

```
$ AWS4-HMAC-SHA256 Credential={access-key}/{date}/{region}/s3/aws4_request,SignedHeaders=host;x-amz-date;{other-required-headers},Signature={signature}
```

Generating an `authorization` header

Python Example

```
$ import os
import datetime
import hashlib
import hmac
import requests

# please don't store credentials directly in code
access_key = os.environ.get('COS_HMAC_ACCESS_KEY_ID')
secret_key = os.environ.get('COS_HMAC_SECRET_ACCESS_KEY')

# request elements
http_method = 'GET'
host = 's3.us.cloud-object-storage.appdomain.cloud'
region = 'us-standard'
endpoint = 'https://s3.us.cloud-object-storage.appdomain.cloud'
bucket = '' # add a '/' before the bucket name to list buckets
object_key = ''
request_parameters = ''


# hashing and signing methods
def hash(key, msg):
    return hmac.new(key, msg.encode('utf-8'), hashlib.sha256).digest()

# region is a wildcard value that takes the place of the AWS region value
# as COS doen't use the same conventions for regions, this parameter can accept any string
def createSignatureKey(key, datestamp, region, service):

    keyDate = hash(('AWS4' + key).encode('utf-8'), datestamp)
    keyString = hash(keyDate, region)
    keyService = hash(keyString, service)
    keySigning = hash(keyService, 'aws4_request')
    return keySigning


# assemble the standardized request
time = datetime.datetime.utcnow()
timestamp = time.strftime('%Y%m%dT%H%M%SZ')
datestamp = time.strftime('%Y%m%d')

standardized_resource = '/' + bucket + '/' + object_key
standardized_querystring = request_parameters
standardized_headers = 'host:' + host + '\n' + 'x-amz-date:' + timestamp + '\n'
signed_headers = 'host;x-amz-date'
payload_hash = hashlib.sha256(''.encode('utf-8')).hexdigest()

standardized_request = (http_method + '\n' +
                        standardized_resource + '\n' +
                        standardized_querystring + '\n' +
                        standardized_headers + '\n' +
                        signed_headers + '\n' +
                        payload_hash).encode('utf-8')


# assemble string-to-sign
hashing_algorithm = 'AWS4-HMAC-SHA256'
credential_scope = datestamp + '/' + region + '/' + 's3' + '/' + 'aws4_request'
sts = (hashing_algorithm + '\n' +
       timestamp + '\n' +
       credential_scope + '\n' +
       hashlib.sha256(standardized_request).hexdigest())


# generate the signature
signature_key = createSignatureKey(secret_key, datestamp, region, 's3')
signature = hmac.new(signature_key,
                    (sts).encode('utf-8'),
                    hashlib.sha256).hexdigest()


# assemble all elements into the 'authorization' header
v4auth_header = (hashing_algorithm + ' ' +
                 'Credential=' + access_key + '/' + credential_scope + ', ' +
                 'SignedHeaders=' + signed_headers + ', ' +
                 'Signature=' + signature)


# create and send the request
headers = {'x-amz-date': timestamp, 'Authorization': v4auth_header}
# the 'requests' package autmatically adds the required 'host' header
request_url = endpoint + standardized_resource + standardized_querystring

print('\nSending `%s` request to IBM COS -----------------------' % http_method)
print('Request URL = ' + request_url)
request = requests.get(request_url, headers=headers)

print('\nResponse from IBM COS ----------------------------------')
print('Response code: %d\n' % request.status_code)
print(request.text)
```

Java Example

```
$ import java.io.BufferedReader;
import java.io.DataOutputStream;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;
import java.net.URLEncoder;
import java.nio.charset.StandardCharsets;
import java.security.MessageDigest;
import java.time.format.DateTimeFormatter;
import java.time.ZoneId;
import java.time.ZoneOffset;
import java.time.ZonedDateTime;
import java.util.Formatter;

import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;

public class CosHMAC {
    // please don't store credentials directly in code
    private static final String accessKey = System.getenv("COS_HMAC_ACCESS_KEY_ID");
    private static final String secretKey = System.getenv("COS_HMAC_SECRET_ACCESS_KEY");
    // constants
    private static final String httpMethod = "GET";
    private static final String host = "s3.us.cloud-object-storage.appdomain.cloud";
    private static final String region = "us-standard";
    private static final String endpoint = "https://s3.us.cloud-object-storage.appdomain.cloud";
    private static final String bucket = ""; // add a '/' before the bucket name to list buckets
    private static final String objectKey = "";
    private static final String requestParameters = "";

    public static void main(String[] args) {
        try {
            // assemble the standardized request
            ZonedDateTime time = ZonedDateTime.now(ZoneOffset.UTC);
            String datestamp = time.format(DateTimeFormatter.ofPattern("yyyyMMdd"));
            String timestamp = datestamp + "T" + time.format(DateTimeFormatter.ofPattern("HHmmss")) + "Z";

            String standardizedResource = bucket + "/" + objectKey;
            String standardizedQuerystring = requestParameters;
            String standardizedHeaders = "host:" + host + "\n" + "x-amz-date:" + timestamp + "\n";
            String signedHeaders = "host;x-amz-date";
            String payloadHash = hashHex("");

            String standardizedRequest = httpMethod + "\n" +
                standardizedResource + "\n" +
                standardizedQuerystring + "\n" +
                standardizedHeaders + "\n" +
                signedHeaders + "\n" +
                payloadHash;

            // assemble string-to-sign
            String hashingAlgorithm = "AWS4-HMAC-SHA256";
            String credentialScope = datestamp + "/" + region + "/" + "s3" + "/" + "aws4_request";
            String sts = hashingAlgorithm + "\n" +
                timestamp + "\n" +
                credentialScope + "\n" +
                hashHex(standardizedRequest);

            // generate the signature
            byte[] signatureKey = createSignatureKey(secretKey, datestamp, region, "s3");
            String signature = hmacHex(signatureKey, sts);

            // assemble all elements into the "authorization" header
            String v4auth_header = hashingAlgorithm + " " +
                "Credential=" + accessKey + "/" + credentialScope + ", " +
                "SignedHeaders=" + signedHeaders + ", " +
                "Signature=" + signature;

            // create and send the request
            String requestUrl = endpoint + standardizedResource + standardizedQuerystring;
            URL urlObj = new URL(requestUrl);
            HttpURLConnection con = (HttpURLConnection) urlObj.openConnection();
            con.setRequestMethod(httpMethod);

            //add request headers
            con.setRequestProperty("x-amz-date", timestamp);
            con.setRequestProperty("Authorization", v4auth_header);

            System.out.printf("\nSending %s request to IBM COS -----------------------", httpMethod);
            System.out.println("Request URL = " + requestUrl);

            int responseCode = con.getResponseCode();
            System.out.println("\nResponse from IBM COS ----------------------------------");
            System.out.printf("Response code: %d\n\n", responseCode);

            BufferedReader in = new BufferedReader(new InputStreamReader(con.getInputStream()));
            String inputLine;
            StringBuffer response = new StringBuffer();

            while ((inputLine = in.readLine()) != null) {
                response.append(inputLine);
            }
            in.close();

            //print result
            System.out.println(response.toString());

            con.disconnect();
        }
        catch (Exception ex) {
            System.out.printf("Error: %s\n", ex.getMessage());
        }
    }

    private static String toHexString(byte[] bytes) {
  Formatter formatter = new Formatter();

  for (byte b : bytes) {
   formatter.format("%02x", b);
  }

  return formatter.toString();
 }

    private static byte[] hash(byte[] key, String msg) {
        byte[] returnVal = null;
        try {
            SecretKeySpec signingKey = new SecretKeySpec(key, "HmacSHA256");
            Mac mac = Mac.getInstance("HmacSHA256");
            mac.init(signingKey);
            returnVal = mac.doFinal(msg.getBytes("UTF8"));
        }
        catch (Exception ex) {
            throw ex;
        }
        finally {
            return returnVal;
        }
    }

    private static String hmacHex(byte[] key, String msg) {
        String returnVal = null;
        try {
            returnVal = toHexString(hash(key, msg));
        }
        catch (Exception ex) {
            throw ex;
        }
        finally {
            return returnVal;
        }
    }

    private static String hashHex(String msg) {
        String returnVal = null;
        try {
            MessageDigest digest = MessageDigest.getInstance("SHA-256");
            byte[] encodedhash = digest.digest(msg.getBytes(StandardCharsets.UTF_8));
            returnVal = toHexString(encodedhash);
        }
        catch (Exception ex) {
            throw ex;
        }
        finally {
            return returnVal;
        }
    }

    // region is a wildcard value that takes the place of the AWS region value
    // as COS doesn"t use the same conventions for regions, this parameter can accept any string
    private static byte[] createSignatureKey(String key, String datestamp, String region, String service) {
        byte[] returnVal = null;
        try {
            byte[] keyDate = hash(("AWS4" + key).getBytes("UTF8"), datestamp);
            byte[] keyString = hash(keyDate, region);
            byte[] keyService = hash(keyString, service);
            byte[] keySigning = hash(keyService, "aws4_request");
            returnVal = keySigning;
        }
        catch (Exception ex) {
            throw ex;
        }
        finally {
            return returnVal;
        }
    }
}
```

NodeJS Example

```
$ const crypto = require('crypto');
const moment = require('moment');
const https = require('https');

// please don't store credentials directly in code
```

```
const accessKey = process.env.COS_HMAC_ACCESS_KEY_ID;
const secretKey = process.env.COS_HMAC_SECRET_ACCESS_KEY;

const httpMethod = 'GET';
const host = 's3.us.cloud-object-storage.appdomain.cloud';
const region = 'us-standard';
const endpoint = 'https://s3.us.cloud-object-storage.appdomain.cloud';
const bucket = ''; // add a '/' before the bucket name to list buckets
const objectKey = '';
const requestParameters = '';

// hashing and signing methods
function hash(key, msg) {
    var hmac = crypto.createHmac('sha256', key);
    hmac.update(msg, 'utf8');
    return hmac.digest();
}

function hmacHex(key, msg) {
    var hmac = crypto.createHmac('sha256', key);
    hmac.update(msg, 'utf8');
    return hmac.digest('hex');
}

function hashHex(msg) {
    var hash = crypto.createHash('sha256');
    hash.update(msg);
    return hash.digest('hex');
}

// region is a wildcard value that takes the place of the AWS region value
// as COS doesn't use the same conventions for regions, this parameter can accept any string
function createSignatureKey(key, datestamp, region, service) {
    keyDate = hash(('AWS4' + key), datestamp);
    keyString = hash(keyDate, region);
    keyService = hash(keyString, service);
    keySigning = hash(keyService, 'aws4_request');
    return keySigning;
}

// assemble the standardized request
var time = moment().utc();
var timestamp = time.format('YYYYMMDDTHHmmss') + 'Z';
var datestamp = time.format('YYYYMMDD');

var standardizedResource = bucket + '/' + objectKey;
var standardizedQuerystring = requestParameters;
var standardizedHeaders = 'host:' + host + '\n' + 'x-amz-date:' + timestamp + '\n';
var signedHeaders = 'host;x-amz-date';
var payloadHash = hashHex('');

var standardizedRequest = httpMethod + '\n' +
    standardizedResource + '\n' +
    standardizedQuerystring + '\n' +
    standardizedHeaders + '\n' +
    signedHeaders + '\n' +
    payloadHash;

// assemble string-to-sign
var hashingAlgorithm = 'AWS4-HMAC-SHA256';
var credentialScope = datestamp + '/' + region + '/' + 's3' + '/' + 'aws4_request';
var sts = hashingAlgorithm + '\n' +
    timestamp + '\n' +
    credentialScope + '\n' +
    hashHex(standardizedRequest);

// generate the signature
var signatureKey = createSignatureKey(secretKey, datestamp, region, 's3');
var signature = hmacHex(signatureKey, sts);

// assemble all elements into the 'authorization' header
```

```
var v4authHeader = hashingAlgorithm + ' ' +
    'Credential=' + accessKey + '/' + credentialScope + ', ' +
    'SignedHeaders=' + signedHeaders + ', ' +
    'Signature=' + signature;

// create and send the request
var authHeaders = {'x-amz-date': timestamp, 'Authorization': v4authHeader}
// the 'requests' package autmatically adds the required 'host' header
console.log(authHeaders);
var requestUrl = endpoint + standardizedResource + standardizedQuerystring

console.log(`\nSending ${httpMethod} request to IBM COS -----------------------`);
console.log('Request URL = ' + requestUrl);

var options = {
    host: host,
    port: 443,
    path: standardizedResource + standardizedQuerystring,
    method: httpMethod,
    headers: authHeaders
}

var request = https.request(options, function (response) {
    console.log('\nResponse from IBM COS ---------------------------------');
    console.log(`Response code: ${response.statusCode}\n`);

    response.on('data', function (chunk) {
        console.log(chunk.toString());
    });
});

request.end();
```

Next steps

You can review the documentation for credentials as part of a Service Credential. For an overview on authentication, check out the IBM Cloud Identity and Access Management service.

Creating a pre-signed URL

Pre-signed URLs in IBM Cloud® Object Storage create temporary links that can be used to share an object without requiring additional user credentials when accessed.

Of course, one can also provide a temporary target for sending a PUT request also without needing to provide any more information for authentication. The easiest way to create pre-signed URLs is using the AWS CLI. But first, you may need to run `aws configure` in order to set your Access Key ID and Secret Access Key from your own HMAC-enabled service credential. When you have completed configuring your CLI, use the following example as a template and replace the endpoint and name of your bucket with the appropriate information:

```
$ $ aws --endpoint-url=https://{endpoint} s3 presign s3://{bucket-name}/{new-file-key}
```

Note: If the service credential used to generate the HMAC credentials (used as the Access Key ID and Secret Access Key configuration above) is deleted, the access for the pre-signed URL will fail.

It is also possible to set an expiration time for the URL in seconds (default is 3600):

```
$ $ aws --endpoint-url=https://{endpoint} s3 presign s3://bucket-1/new-file --expires-in 600
```

It is also possible to construct them programmatically. Here are examples for basic `GET` operations written in Python. For more information about endpoints, see Endpoints and storage locations.

Tip: Unlike AWS S3, IBM Cloud Object Storage does not enforce a maximum expiration time of 7 days (604800 seconds). While it is possible to create a pre-signed URL with a long expiration value, most use cases that require extended public access would be better served by implementing a public access policy on a bucket instead.

Create a pre-signed URL to download an object

Python Example

```
import ibm_boto3
import os

bucket_name = '<bucekt name>'
key_name = '<object key name>'
http_method = 'get_object'
expiration = 600  # time in seconds, default:600

access_key = os.environ.get('COS_HMAC_ACCESS_KEY_ID')
secret_key = os.environ.get('COS_HMAC_SECRET_ACCESS_KEY')
# Current list avaiable at https://control.cloud-object-storage.cloud.ibm.com/v2/endpoints
cos_service_endpoint = 'https://s3.<region>.cloud-object-storage.appdomain.cloud'

cos = ibm_boto3.client("s3",
                       aws_access_key_id=access_key,
                       aws_secret_access_key=secret_key,
                       endpoint_url=cos_service_endpoint
                       )

signedUrl = cos.generate_presigned_url(http_method, Params={
                                       'Bucket': bucket_name, 'Key': key_name}, ExpiresIn=expiration)
print("presigned download URL =>" + signedUrl)
```

Java Example

```
$ import java.util.Date;
import com.ibm.cloud.objectstorage.auth.AWSCredentials;
import com.ibm.cloud.objectstorage.auth.AWSStaticCredentialsProvider;
import com.ibm.cloud.objectstorage.client.builder.AwsClientBuilder.EndpointConfiguration;
import com.ibm.cloud.objectstorage.HttpMethod;
import com.ibm.cloud.objectstorage.services.s3.AmazonS3;
import com.ibm.cloud.objectstorage.services.s3.AmazonS3ClientBuilder;
import com.ibm.cloud.objectstorage.services.s3.model.GeneratePresignedUrlRequest;


String bucketName = "<bucket name>";
String keyName = "<object key name>";
HttpMethod httpMethod = HttpMethod.GET;
Date expiration = new Date();
long expTimeMillis = expiration.getTime();
expTimeMillis += 1000 * 60 * 60;
expiration.setTime(expTimeMillis);

String accessKey = "<COS_HMAC_ACCESS_KEY_ID>";
String secretAccessKey = "<COS_HMAC_SECRET_ACCESS_KEY>";
// Current list avaiable at https://control.cloud-object-storage.cloud.ibm.com/v2/endpoints
String cosServiceEndpoint = "https://s3.<region>.cloud-object-storage.appdomain.cloud";

AmazonS3 cosClient = AmazonS3ClientBuilder.standard()
                     .withEndpointConfiguration(new EndpointConfiguration(cosServiceEndpoint, "us-east-1"))
                     .withCredentials(new AWSStaticCredentialsProvider(new BasicAWSCredentials(accessKey,
secretAccessKey))).withPathStyleAccessEnabled(true)
                     .build();

GeneratePresignedUrlRequest generatePresignedUrlRequest = new GeneratePresignedUrlRequest(bucketName, keyName)
                                                          .withMethod(httpMethod)
                                                          .withExpiration(expiration);

URL signedUrl = cosClient.generatePresignedUrl(generatePresignedUrlRequest);
System.out.println(signedUrl);
```

Create a pre-signed URL to upload an object

Python Example

```
import ibm_boto3
import os
```

```
bucket_name = '<bucket name>'
key_name = '<object key name>'
http_method = 'put_object'
expiration = 600  # time in seconds, default:600

access_key = os.environ.get('COS_HMAC_ACCESS_KEY_ID')
secret_key = os.environ.get('COS_HMAC_SECRET_ACCESS_KEY')
# Current list avaiable at https://control.cloud-object-storage.cloud.ibm.com/v2/endpoints
cos_service_endpoint = 'https://s3.<region>.cloud-object-storage.appdomain.cloud'

cos = ibm_boto3.client("s3",
                       aws_access_key_id=access_key,
                       aws_secret_access_key=secret_key,
                       endpoint_url=cos_service_endpoint
                       )

signedUrl = cos.generate_presigned_url(http_method, Params={
                                       'Bucket': bucket_name, 'Key': key_name}, ExpiresIn=expiration)
print("presigned upload URL =>" + signedUrl)
```

Managing a bucket's lifecycle configurations

Archiving and accessing cold data

IBM Cloud® Object Storage "Archive" and "Accelerated Archive" are low cost options for data that is rarely accessed. You can store data by transitioning from any of the storage tiers (Standard, Vault, Cold Vault and Flex) to long-term offline archive or use the online Cold Vault option. With the new "Accelerated Archive" feature you can quickly access dormant data with restoration occurring in less than two hours.

Note: This feature is not currently supported in Object Storage for Satellite. Learn more.

You can archive objects using the web console, REST API, and 3rd party tools that are integrated with IBM Cloud Object Storage.

Tip: For more information about endpoints, see Endpoints and storage locations

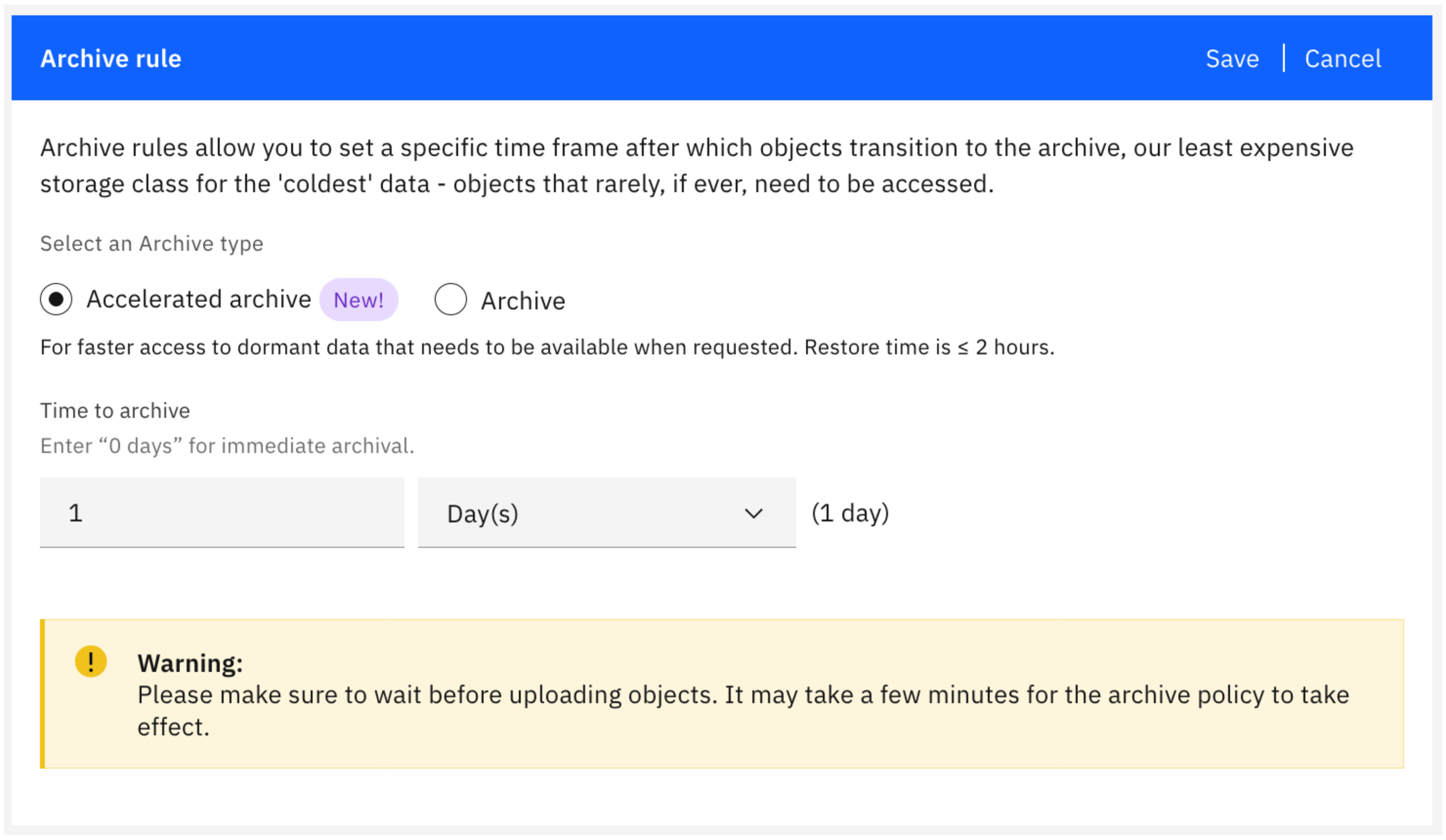
Add or manage an archive policy on a bucket

When creating or modifying an archive policy for a bucket, consider the following:

- An archive policy can be added to a new or existing bucket at any time.
- An existing archive policy can be modified or disabled.
- A newly added or modified archive policy applies to new objects uploaded and does not affect existing objects.

Create a bucket in the console after you've logged in, and you can configure your archive policy using the fields shown in Figure 1.

Create an archive policy



Archive rules allow you to set a specific time frame after which objects transition to the archive, our least expensive storage class for the 'coldest' data - objects that rarely, if ever, need to be accessed.

Select an Archive type

[✓ Accelerated archive] New! Archive

For faster access to dormant data that needs to be available when requested. Restore time is ≤ 2 hours.

Time to archive

Enter “0 days” for immediate archival.

1 Day(s) (1 day)

Warning:
Please make sure to wait before uploading objects. It may take a few minutes for the archive policy to take effect.

Tip: To immediately archive new objects uploaded to a bucket, enter 0 days on the archive policy.

Tip: Archive is available in certain regions only. See Integrated Services for more details.

Restore an archived object

In order to access an archived object, you must restore it to the original storage tier. When restoring an object, you can specify the number of days you want the object to be available. At the end of the specified period, the restored copy is deleted.

Tip: The restoration process for "Accelerated Archive" takes up to 2 hours, while the restoration process for Archive takes up to 12 hours.

The archived object sub-states are:

- Archived: An object in the archived state has been moved from its online storage tier (Standard, Vault, Cold Vault and Flex) to the offline archive tier based on the archive policy on the bucket.
- Restoring: An object in the restoring state is in the process of generating a copy from the archived state to its original online storage tier.
- Restored: An object in the restored state is a copy of the archived object that was restored to its original online storage tier for a specified amount of time. At the end of the period, the copy of the object is deleted, while maintaining the archived object.

Restoring an object using the AWS CLI

The following examples uses environment variables for clarity. These must be set to the desired values, for example `$ENDPOINT` would be set to `https://s3.us.cloud-object-storage.appdomain.cloud`, or `https://s3.eu-de.private.cloud-object-storage.appdomain.cloud`, or any other required value.

1. Check object status: `aws --endpoint-url $ENDPOINT s3api head-object --bucket $BUCKET --key $KEY` The storage class will be shown as `("StorageClass": "GLACIER")`
2. Restore the object: `aws --endpoint-url $ENDPOINT s3api restore-object ---bucket $BUCKET --key $KEY --restore-request '{"Days":25,"GlacierJobParameters":{"Tier":"Bulk"}}'`
3. Check the status: `aws --endpoint-url $ENDPOINT s3api head-object --bucket $BUCKET --key $KEY`

Limitations

Archive policies are implemented using subset of the `PUT Bucket Lifecycle Configuration` S3 API operation.

Supported functionality includes:

- Specifying either a date or the number of days in the future when objects transition to an archived state.

- Setting expiration rules for objects.

Note: Policies specifying a date in the past may take up to a few days to complete.

Unsupported functionality includes:

- Multiple transition rules per bucket.
- Filtering objects to archive using a prefix or object key.
- Tiering between storage classes.

Note: Classic Infrastructure (non-IAM) users are unable to set the transition storage class to `ACCELERATED`.

Using the REST API and SDKs

Create a bucket lifecycle configuration

This implementation of the `PUT` operation uses the `lifecycle` query parameter to set lifecycle settings for the bucket. This operation allows for a single lifecycle policy definition for a given bucket. The policy is defined as a rule consisting of the following parameters: `ID`, `Status`, and `Transition`.

The transition action enables future objects written to the bucket to an archived state after a defined period of time. Changes to the lifecycle policy for a bucket are **only applied to new objects** written to that bucket.

Cloud IAM users must have at a minimum the `Writer` role to add a lifecycle policy to the bucket.

Classic Infrastructure Users must have Owner Permissions and be able to create buckets in the storage account to add a lifecycle policy to the bucket.

This operation does not make use of additional operation specific query parameters.

Header	Type	Description
`Content-MD5`	string	**Required**: The base64 encoded 128-bit MD5 hash of the payload, used as an integrity check to ensure the payload was not altered in transit.

The body of the request must contain an XML block with the following schema:

Element	Type	Children	Ancestor	Constraint
`LifecycleConfiguration`	Container	`Rule`	None	Limit 1.
`Rule`	Container	`ID`, `Status`, `Filter`, `Transition`	`LifecycleConfiguration`	Limit 1.
`ID`	String	None	`Rule`	Must consist of (`a-z`,`A-Z`, `0-9`) and the following symbols: `!` `_` `.` `*` `'` `(` `)` `-`
`Filter`	String	`Prefix`	`Rule`	Must contain a `Prefix` element
`Prefix`	String	None	`Filter`	**Must** be set to `<Prefix/>`.
`Transition`	`Container`	`Days`, `StorageClass`	`Rule`	Limit 1 transition rule, and a maximum of 1000 total rules.
`Days`	Non-negative integer	None	`Transition`	Must be a value equal to or greater than 0.
`Date`	Date	None	`Transistion`	Must be in ISO 8601 Format and the date must be in the future.
`StorageClass`	String	None	`Transition`	`GLACIER` or `ACCELERATED`

Syntax

```
PUT https://{endpoint}/{bucket}?lifecycle # path style
PUT https://{bucket}.{endpoint}?lifecycle # virtual host style
```

```
<LifecycleConfiguration>
 <Rule>
  <ID>{string}</ID>
  <Status>Enabled</Status>
  <Filter>
   <Prefix/>
  </Filter>
  <Transition>
   <Days>{integer}</Days>
   <StorageClass>{StorageClass}</StorageClass>
  </Transition>
 </Rule>
</LifecycleConfiguration>
```

Examples

Sample Request

```
PUT /images?lifecycle HTTP/1.1
Host: s3.us.cloud-object-storage.appdomain.cloud
Date: Wed, 7 Feb 2018 17:50:00 GMT
Authorization: authorization string
Content-Type: text/plain
Content-MD5: 1B2M2Y8AsgTpgAmY7PhCfg==
Content-Length: 305
```

```
<LifecycleConfiguration>
    <Rule>
        <ID>my-archive-policy</ID>
        <Filter>
   <Prefix/>
  </Filter>
        <Status>Enabled</Status>
        <Transition>
            <Days>20</Days>
            <StorageClass>ACCELERATED</StorageClass>
        </Transition>
    </Rule>
</LifecycleConfiguration>
```

Sample Response

```
HTTP/1.1 200 OK
Date: Wed, 7 Feb 2018 17:51:00 GMT
Connection: close
```

Node

```
var params = {
  Bucket: 'STRING_VALUE', /* required */
  LifecycleConfiguration: {
    Rules: [ /* required */
      {
        Status: 'Enabled', /* required */
        ID: 'STRING_VALUE',
        Filter: '', /* required */
        Prefix: '',
        Transitions: [
          {
            Date: DATE, /* required if Days not specified */
            Days: 0, /* required if Date not specified */
            StorageClass: 'STRING_VALUE' /* required */
          },
        ]
      },
    ]
  }
};

s3.putBucketLifecycleConfiguration(params, function(err, data) {
  if (err) console.log(err, err.stack); // an error occurred
  else     console.log(data);           // successful response
});
```

Python

```
response = client.put_bucket_lifecycle_configuration(
    Bucket='string',
    LifecycleConfiguration={
        'Rules': [
            {
                'ID': 'string',
                'Status': 'Enabled',
                'Filter': '',
                'Prefix': '',
                'Transitions': [
                    {
                        'Date': datetime(2015, 1, 1),
                        'Days': 123,
                        'StorageClass': 'GLACIER'
                    },
                ]
            },
        ]
    }
)
```

Java

```
public SetBucketLifecycleConfigurationRequest(String bucketName,
                                              BucketLifecycleConfiguration lifecycleConfiguration)
```

Method Summary

Method	Description
`getBucketName()`	Gets the name of the bucket whose lifecycle configuration is being set.
`getLifecycleConfiguration()`	Gets the new lifecycle configuration for the specified bucket.
`setBucketName(String bucketName)`	Sets the name of the bucket whose lifecycle configuration is being set.
`withBucketName(String bucketName)`	Sets the name of the bucket whose lifecycle configuration is being set, and returns this object so that additional method calls may be chained together.

Retrieve a bucket lifecycle configuration

This implementation of the `GET` operation uses the `lifecycle` query parameter to retrieve the lifecycle settings for the bucket.

Cloud IAM users must have at a minimum the `Reader` role to retrieve a lifecycle for a bucket.

Classic Infrastructure Users must have at minimum `Read` permissions on the bucket to retrieve a lifecycle policy for a bucket.

This operation does not make use of additional operation specific headers, query parameters, or payload.

Syntax

```
GET https://{endpoint}/{bucket}?lifecycle # path style
GET https://{bucket}.{endpoint}?lifecycle # virtual host style
```

Examples

Sample Request

```
GET /images?lifecycle HTTP/1.1
Host: s3.us.cloud-object-storage.appdomain.cloud
Date: Wed, 7 Feb 2018 17:50:00 GMT
Authorization: authorization string
```

Sample Response

```
HTTP/1.1 200 OK
Date: Wed, 7 Feb 2018 17:51:00 GMT
Connection: close
```

```
<LifecycleConfiguration>
    <Rule>
        <ID>my-archive-policy</ID>
        <Filter />
        <Status>Enabled</Status>
        <Transition>
            <Days>20</Days>
            <StorageClass>GLACIER</StorageClass>
        </Transition>
    </Rule>
</LifecycleConfiguration>
```

Node

```
var params = {
  Bucket: 'STRING_VALUE' /* required */
};
s3.getBucketLifecycleConfiguration(params, function(err, data) {
  if (err) console.log(err, err.stack); // an error occurred
  else     console.log(data);           // successful response
});
```

Python

```
response = client.get_bucket_lifecycle_configuration(Bucket='string')
```

Java

```
public GetBucketLifecycleConfigurationRequest(String bucketName)
```

Delete a bucket lifecycle configuration

This implementation of the `DELETE` operation uses the `lifecycle` query parameter to remove any lifecycle settings for the bucket. Transitions defined by the rules will no longer take place for new objects.

Note: Existing transition rules will be maintained for objects that were already written to the bucket before the rules were deleted.

Cloud IAM users must have at a minimum the `Writer` role to remove a lifecycle policy from a bucket.

Classic Infrastructure Users must have `Owner` permissions on the bucket to remove a lifecycle policy from a bucket.

This operation does not make use of additional operation specific headers, query parameters, or payload.

Syntax

```
DELETE https://{endpoint}/{bucket}?lifecycle # path style
DELETE https://{bucket}.{endpoint}?lifecycle # virtual host style
```

Examples

Sample Request

```
DELETE /images?lifecycle HTTP/1.1
Host: s3.us.cloud-object-storage.appdomain.cloud
Date: Wed, 7 Feb 2018 18:50:00 GMT
Authorization: authorization string
```

Sample Response

```
HTTP/1.1 204 No Content
Date: Wed, 7 Feb 2018 18:51:00 GMT
Connection: close
```

Node

```
var params = {
  Bucket: 'STRING_VALUE' /* required */
};
s3.deleteBucketLifecycle(params, function(err, data) {
  if (err) console.log(err, err.stack); // an error occurred
  else     console.log(data);           // successful response
});
```

Python

```
response = client.delete_bucket_lifecycle(Bucket='string')
```

Java

```
public DeleteBucketLifecycleConfigurationRequest(String bucketName)
```

Temporarily restore an archived object

This implementation of the `POST` operation uses the `restore` query parameter to request temporary restoration of an archived object. The user must first restore an archived object before downloading or modifying the object. When restoring an object, the user must specify a period after which the temporary copy of the object will be deleted. The object maintains the storage class of the bucket.

There can be a delay of up to 12 hours before the restored copy is available for access. A `HEAD` request can check if the restored copy is available.

To permanently restore the object, the user must copy the restored object to a bucket that does not have an active lifecycle configuration.

Cloud IAM users must have at a minimum the `Writer` role to restore an object.

Classic Infrastructure users must have at a minimum `Write` permissions on the bucket and `Read` permission on the object to restore it.

This operation does not make use of additional operation specific query parameters.

Header	Type	Description
`Content-MD5`	string	**Required**: The base64 encoded 128-bit MD5 hash of the payload, used as an integrity check to ensure the payload was not altered in transit.

The body of the request must contain an XML block with the following schema:

Element	Type	Children	Ancestor	Constraint
`RestoreRequest`	Container	`Days`, `GlacierJobParameters`	None	None
`Days`	Integer	None	`RestoreRequest`	Specified the lifetime of the temporarily restored object. The minimum number of days that a restored copy of the object can exist is 1. After the restore period has elapsed, temporary copy of the object will be removed.

`GlacierJobParameters`	String	`Tier`	`RestoreRequest`	None
`Tier`	String	None	`GlacierJobParameters`	Optional, and if left blank will default to the value associated with the storage tier of the policy that was in place when the object was written. If this value is not left blank, it **must** be set to `Bulk` if the transition storage class for the bucket's lifecycle policy was set to `GLACIER`, and **must** be set to `Accelerated` if the transition storage class was set to `ACCELERATED`.

A successful response returns a `202` if the object is in the archived state and a `200` if the object is already in the restored state. If the object is already in the restored state and a new request to restore the object is received, the `Days` element will update the expiration time of the restored object.

Syntax

```
POST https://{endpoint}/{bucket}/{object}?restore # path style
POST https://{bucket}.{endpoint}/{object}?restore # virtual host style
```

```
<RestoreRequest>
 <Days>{integer}</Days>
 <GlacierJobParameters>
  <Tier>Bulk</Tier>
 </GlacierJobParameters>
</RestoreRequest>
```

Examples

Sample Request

```
POST /images/backup?restore HTTP/1.1
Host: s3.us.cloud-object-storage.appdomain.cloud
Date: Wed, 7 Feb 2018 19:50:00 GMT
Authorization: {authorization string}
Content-Type: text/plain
Content-MD5: 1B2M2Y8AsgTpgAmY7PhCfg==
Content-Length: 305
```

```
<RestoreRequest>
 <Days>3</Days>
 <GlacierJobParameters>
  <Tier>Bulk</Tier>
 </GlacierJobParameters>
</RestoreRequest>
```

Sample Response

```
HTTP/1.1 202 Accepted
Date: Wed, 7 Feb 2018 19:51:00 GMT
Connection: close
```

Node

```
var params = {
  Bucket: 'STRING_VALUE', /* required */
  Key: 'STRING_VALUE', /* required */
  ContentMD5: 'STRING_VALUE', /* required */
  RestoreRequest: {
   Days: 1, /* days until copy expires */
   GlacierJobParameters: {
     Tier: 'STRING_VALUE' /* required */
   },
  }
 };
 s3.restoreObject(params, function(err, data) {
   if (err) console.log(err, err.stack); // an error occurred
  else     console.log(data);           // successful response
});
```

Python

```
response = client.restore_object(
    Bucket='string',
    Key='string',
    RestoreRequest={
        'Days': 123,
        'GlacierJobParameters': {
            'Tier': 'string'
        },
    }
)
```

Java

```
public RestoreObjectRequest(String bucketName,
                            String key,
                            int expirationInDays)
```

Method Summary

Method	Description
`clone()`	Creates a shallow clone of this object for all fields except the handler context.
`getBucketName()`	Returns the name of the bucket containing the reference to the object to restore.
`getExpirationInDays()`	Returns the time in days from an object's creation to its expiration.
`setExpirationInDays(int expirationInDays)`	Sets the time, in days, between when an object is uploaded to the bucket and when it expires.

Get an object's headers

A `HEAD` given a path to an object retrieves that object's headers. This operation does not make use of operation specific query parameters or payload elements.

Syntax

```
HEAD https://{endpoint}/{bucket-name}/{object-name} # path style
HEAD https://{bucket-name}.{endpoint}/{object-name} # virtual host style
```

Response headers for archived objects

Header	Type	Description
`x-amz-restore`	string	Included if the object has been restored or if a restoration is in progress. If the object has been restored, the expiry date for the temporary copy is also returned.
`x-amz-storage-class`	string	Returns `GLACIER` or `ACCELERATED` if archived or temporarily restored.
`x-ibm-archive-transition-time`	date	Returns the date and time when the object is scheduled to transition to the archive tier.
`x-ibm-transition`	string	Included if the object has transition metadata and returns the tier and original time of transition.
`x-ibm-restored-copy-storage-class`	string	Included if an object is in the `RestoreInProgress` or `Restored` states and returns the storage class of the bucket.

Sample request

```
HEAD /images/backup HTTP/1.1
Authorization: {authorization-string}
x-amz-date: 20160825T183244Z
Host: s3.us.cloud-object-storage.appdomain.cloud
```

Sample response

```
HTTP/1.1 200 OK
Date: Wed, 7 Feb 2018 19:51:00 GMT
X-Clv-Request-Id: da214d69-1999-4461-a130-81ba33c484a6
Accept-Ranges: bytes
Server: 3.x
X-Clv-S3-Version: 2.5
ETag: "37d4c94839ee181a2224d6242176c4b5"
Content-Type: text/plain; charset=UTF-8
Last-Modified: Thu, 25 Aug 2017 17:49:06 GMT
Content-Length: 11
x-ibm-transition: transition="ARCHIVE", date="Mon, 03 Dec 2018 22:28:38 GMT"
x-amz-restore: ongoing-request="false", expiry-date="Thu, 06 Dec 2018 18:28:38 GMT"
x-amz-storage-class: "GLACIER"
x-ibm-restored-copy-storage-class: "Standard"
```

Python

```
response = client.head_object(
    Bucket='string',
    Key='string'
)
```

Node

```
var params = {
  Bucket: 'STRING_VALUE', /* required */
  Key: 'STRING_VALUE', /* required */
};
s3.headObject(params, function(err,data) {
  if (err) console.log(err, err.stack); // an error occurred
  else
    console.log(data);           // successful response
});
```

Java

```
public ObjectMetadata()
```

Method Summary

Method	Description
`clone()`	Returns a clone of this `ObjectMetadata`.
`getRestoreExpirationTime()`	Returns the time at which an object that has been temporarily restored from ARCHIVE will expire, and will need to be restored again in order to be accessed.
`getStorageClass()`	Returns the original storage class of the bucket.
`getIBMTransition()`	Return the transition storage class and time of transition.

Next Steps

In addition to cold storage, IBM Cloud currently provides several additional object storage classes for different user needs, all of which are accessible through web-based portals and RESTful APIs. Learn more about all storage classes available at IBM Cloud Object Storage.

Deleting stale data with expiration rules

An expiration rule deletes objects after a defined period (from the object creation date).

You can set the lifecycle for objects by using the web console, REST API, and third-party tools that are integrated with IBM Cloud Object Storage.

- An expiration rule can be added to a new or existing bucket.
- An existing expiration rule can be modified or disabled.
- A newly added or modified Expiration rule applies to all new and existing objects in the bucket.
- Adding or modifying lifecycle policies requires the `Writer` role.
- Up to 1000 lifecycle rules (archive + expiration) can be defined per bucket.
- Allow up to 24 hours for any changes in Expiration rules to take effect.
- The scope of each expiration rule can be limited by defining an optional prefix filter to apply to only a subset of objects with names that match the prefix.
- An expiration rule without a prefix filter will apply to all objects in the bucket.
- The expiration period for an object, specified in number(s) of days, is calculated from the time the object was created, and is rounded off to the next day's midnight UTC. For example, if you have an expiration rule for a bucket to expire a set of objects ten days after the creation date, an object that was created on 15 April 2019 05:10 UTC will expire on 26 April 2019 00:00 UTC.
- The expiration rules for each bucket are evaluated once every 24 hours. Any object that qualifies for expiration (based on the objects' expiration date) will be queued for deletion. The deletion of expired objects begins the following day and will typically take less than 24 hours. You will not be billed for any associated storage for objects once they are deleted.
- In versioning enabled or suspended buckets, a regular rule retains the current version and creates a delete marker rather than permanently deleting data. To clean up a versioning enabled bucket, you can set two expiration rules. Add a first rule that sets an expiration for the current version and the noncurrent version. Because versioning is enabled, this rule creates a delete marker for the version. Add a second rule with the selection "Clean up expired object delete markers" to delete the data.
- The expiration time of non-current versions is determined by its successor's last modified time, rounded to the next day at midnight UTC.
- If versions are manually deleted from an object that has versions expected to be expired the next day, those expirations may not occur.

Note: Policies specifying a date in the past may take up to a few days to complete.

Important: Without caution, data might be permanently lost with when using expiration rules on a versioned bucket. In cases where **versioning is suspended** and there is a null version present for the expired object, data might be permanently lost. In this case, a `null` delete marker is overwritten, permanently deleting the object.

Important: Objects that are subject to a bucket's Immutable Object Storage retention policy will have any expiration actions deferred until the retention policy is no longer enforced.

Attributes of expiration rules

Each expiration rule has the following attributes:

ID

A rule's ID must be unique within the bucket's lifecycle configuration.

Expiration

The expiration block contains the details that govern the automatic deletion of objects. This could be a specific date in the future, or a period of time after new objects are written.

NoncurrentVersionExpiration

The number of days after which non-current versions of objects are automatically deleted.

Prefix

An optional string that will be matched to the prefix of the object name in the bucket. A rule with a prefix will only apply to the objects that match. You can use multiple rules for different expiration actions for different prefixes within the same bucket. For example, within the same lifecycle configuration, one rule could delete all objects that begin with `logs/` after 30 days, and a second rule could delete objects that begin with `video/` after 365 days.

Status

A rule can either be enabled or disabled. A rule is active only when enabled.

Sample lifecycle configurations

This configuration expires any new objects after 30 days.

```
$ <LifecycleConfiguration>
 <Rule>
  <ID>delete-after-30-days</ID>
  <Filter />
  <Status>Enabled</Status>
  <Expiration>
   <Days>30</Days>
  </Expiration>
 </Rule>
</LifecycleConfiguration>
```

This configuration deletes any objects with the prefix `foo/` on June 1, 2020.

```
$ <LifecycleConfiguration>
 <Rule>
        <ID>delete-on-a-date</ID>
    <Filter>
        <Prefix>foo/</Prefix>
    </Filter>
        <Status>Enabled</Status>
        <Expiration>
            <Date>2020-06-01T00:00:00.000Z</Date>
        </Expiration>
 </Rule>
</LifecycleConfiguration>
```

This configuration expires any non-current versions of objects after 100 days.

```
$ <LifecycleConfiguration>
  <Rule>
    <ID>DeleteAfterBecomingNonCurrent</ID>
    <Filter/>
    <Status>Enabled</Status>
    <NoncurrentVersionExpiration>
      <NoncurrentDays>100</NoncurrentDays>
    </NoncurrentVersionExpiration>
  </Rule>
</LifecycleConfiguration>
```

You can also combine transition and expiration rules. This configuration archives any objects 90 days after creation, and deletes any objects with the prefix `foo/` after 180 days .

```
$ <LifecycleConfiguration>
    <Rule>
        <ID>archive-first</ID>
        <Filter />
        <Status>Enabled</Status>
    <Transition>
        <Days>90</Days>
        <StorageClass>GLACIER</StorageClass>
    </Transition>
    </Rule>
    <Rule>
        <ID>then-delete</ID>
    <Filter>
        <Prefix>foo/</Prefix>
    </Filter>
        <Status>Enabled</Status>
        <Expiration>
            <Days>180</Days>
        </Expiration>
    </Rule>
</LifecycleConfiguration>
```

Using the console

When creating a new bucket, check the **Add expiration rule** box. Next, click **Add rule** to create the new expiration rule. You can add up to five rules during bucket creation, and extra rules can be added later.

For an existing bucket, select **Configuration** from the navigation menu and click **Add rule** under the *Expiration rule* section.

Using the API and SDKs

You can programmatically manage expiration rules by using the REST API or the IBM COS SDKs. Select the format for the examples by selecting a category in the context switcher.

Add an expiration rule to a bucket's lifecycle configuration

REST API reference

This implementation of the `PUT` operation uses the `lifecycle` query parameter to set lifecycle settings for the bucket. This operation allows for a single lifecycle policy definition for a bucket. The policy is defined as a set of rules consisting of the following parameters: `ID`, `Status`, `Filter`, and `Expiration`.

Cloud IAM users must have the `Writer` role to add a lifecycle policy from a bucket.

Classic Infrastructure Users must have `Owner` permissions on the bucket to add a lifecycle policy from a bucket.

Header	Type	Description
`Content-MD5`	String	**Required**: The base64 encoded 128-bit MD5 hash of the payload, which is used as an integrity check to ensure that the payload wasn't altered in transit.

The body of the request must contain an XML block with the following schema:

Element	Type	Children	Ancestor	Constraint
`LifecycleConfiguration`	Container	`Rule`	None	Limit 1.
`Rule`	Container	`ID`, `Status`, `Filter`, `Expiration`	`LifecycleConfiguration`	Limit 1000.
`ID`	String	None	`Rule`	Must consist of (`a-z`,`A-Z`,`0-9`) and the following symbols: `!` `_` `.` `*` `'` `(` `)` `-`
`Filter`	String	`Prefix`	`Rule`	Must contain a `Prefix` element
`Prefix`	String	None	`Filter`	The rule applies to any objects with keys that match this prefix.
`Expiration`	`Container`	`Days` or `Date`	`Rule`	Limit 1.
`Days`	Non-negative integer	None	`Expiration`	Must be a value greater than 0.
`Date`	Date	None	`Expiration`	Must be in ISO 8601 Format.
`NoncurrentVersionExpiration`	Date	`NoncurrentDays`	`Rule`	Limit 1.
`NoncurrentDays`	Non-negative integer	None	`NoncurrentVersionExpiration`	Must be a value greater than 0.

The body of the request must contain an XML block with the schema that is addressed in the table (see Example 1).

```
<LifecycleConfiguration>
 <Rule>
  <ID>id1</ID>
  <Filter />
  <Status>Enabled</Status>
  <Expiration>
   <Days>60</Days>
  </Expiration>
 </Rule>
</LifecycleConfiguration>
```

Syntax

```
PUT https://{endpoint}/{bucket}?lifecycle # path style
PUT https://{bucket}.{endpoint}?lifecycle # virtual host style
```

Example request

```
PUT /images?lifecycle HTTP/1.1
Host: s3.us.cloud-object-storage.appdomain.cloud
Date: Wed, 7 Feb 2018 17:50:00 GMT
Authorization: authorization string
Content-Type: text/plain
Content-MD5: M625BaNwd/OytcM7O5gIaQ==
Content-Length: 305

<LifecycleConfiguration>
 <Rule>
  <ID>id1</ID>
  <Filter />
  <Status>Enabled</Status>
  <Expiration>
   <Days>60</Days>
  </Expiration>
 </Rule>
</LifecycleConfiguration>
```

Code sample for use with NodeJS COS SDK

Using the IBM Cloud® Object Storage SDKs only requires calling the appropriate functions with the correct parameters and proper configuration.

Node

```
var aws = require('ibm-cos-sdk');
var ep = new aws.Endpoint('s3.us-south.cloud-object-storage.appdomain.cloud');
var config = {
    endpoint: ep,
    apiKeyId: 'ZRZDoNoUseOLL7bRO8SAMPLEHPUzUL_-fsampleyYE',
    ibmAuthEndpoint: 'https://iam.cloud.ibm.com/identity/token',
    serviceInstanceId: 'crn:v1:bluemix:public:cloud-object-storage:global:a/<CREDENTIAL_ID_AS_GENERATED>:
<SERVICE_ID_AS_GENERATED>::',
};
var s3 = new aws.S3(config);
var date = new Date('June 16, 2019 00:00:00');

var params = {
  Bucket: 'STRING_VALUE', /* required */
  LifecycleConfiguration: {
    Rules: [ /* required */
      {
        Status: 'Enabled', /* required */
        ID: 'OPTIONAL_STRING_VALUE',
        Filter: {}, /* required */
        Expiration:
        {
          Date: date
        }
      },
    ]
  }
};

s3.putBucketLifecycleConfiguration(params, function(err, data) {
  if (err) console.log(err, err.stack); // an error occurred
  else     console.log(data);           // successful response
});
```

Code sample for use with Python COS SDK

Using the IBM Cloud® Object Storage SDKs only requires calling the appropriate functions with the correct parameters and proper configuration.

Python

```
import sys
import ibm_boto3
from ibm_botocore.client import Config

api_key = "ZRZDoNoUseOLL7bRO8SAMPLEHPUzUL_-fsampleyYE"
service_instance_id = "85SAMPLE-eDOb-4NOT-bUSE-86nnnb31eaxx"
auth_endpoint = "https://iam.cloud.ibm.com/identity/token"
service_endpoint = "https://s3.us-south.cloud-object-storage.appdomain.cloud"

cos = ibm_boto3.client('s3',
                         ibm_api_key_id=api_key,
                         ibm_service_instance_id=service_instance_id,
                         ibm_auth_endpoint=auth_endpoint,
                         config=Config(signature_version='oauth'),
                         endpoint_url=service_endpoint)

response = cos.put_bucket_lifecycle_configuration(
    Bucket='string',
    LifecycleConfiguration={
        'Rules': [
            {
                'Status': 'Enabled',
                'Filter': {},
                'Expiration':
                {
                    'Days': 123
                },
            },
        ]
    }
)

print("Bucket lifecyle: {0}".format(response))
```

Code sample for use with Java COS SDK

Using the IBM Cloud® Object Storage SDKs only requires calling the appropriate functions with the correct parameters and proper configuration.

Java

```
package com.ibm.cloud;

    import java.sql.Timestamp;
    import java.util.List;
    import java.util.Arrays;

    import com.ibm.cloud.objectstorage.ClientConfiguration;
    import com.ibm.cloud.objectstorage.SDKGlobalConfiguration;
    import com.ibm.cloud.objectstorage.auth.AWSCredentials;
    import com.ibm.cloud.objectstorage.auth.AWSStaticCredentialsProvider;
    import com.ibm.cloud.objectstorage.client.builder.AwsClientBuilder.EndpointConfiguration;
    import com.ibm.cloud.objectstorage.services.s3.AmazonS3;
    import com.ibm.cloud.objectstorage.services.s3.AmazonS3ClientBuilder;
    import com.ibm.cloud.objectstorage.services.s3.model.Bucket;
    import com.ibm.cloud.objectstorage.services.s3.model.BucketLifecycleConfiguration;
    import com.ibm.cloud.objectstorage.services.s3.model.ListObjectsRequest;
    import com.ibm.cloud.objectstorage.services.s3.model.ObjectListing;
    import com.ibm.cloud.objectstorage.services.s3.model.S3ObjectSummary;
    import com.ibm.cloud.objectstorage.oauth.BasicIBMOAuthCredentials;

    public class App
    {
        private static AmazonS3 _cosClient;

        /**
         * @param args
         */
        public static void main(String[] args)
        {
            SDKGlobalConfiguration.IAM_ENDPOINT = "https://iam.cloud.ibm.com/identity/token";
            String bucketName = "<sample-bucket-name>";
            String api_key = "ZRZDoNoUseOLL7bRO8SAMPLEHPUzUL_-fsampleyYE";
            String service_instance_id = "85SAMPLE-eDOb-4NOT-bUSE-86nnnb31eaxx"
            String endpoint_url = "https://s3.us-south.cloud-object-storage.appdomain.cloud";
            String storageClass = "us-south";
            String location = "us";

            _cosClient = createClient(api_key, service_instance_id, endpoint_url, location);

            // Define a rule for expiring items in a bucket
            int days_to_delete = 10;
            BucketLifecycleConfiguration.Rule rule = new BucketLifecycleConfiguration.Rule()
                    .withId("Delete rule")
                    .withExpirationInDays(days_to_delete)
                    .withStatus(BucketLifecycleConfiguration.ENABLED);
                    rule.setFilter(new LifecycleFilter());

            // Add the rule to a new BucketLifecycleConfiguration.
            BucketLifecycleConfiguration configuration = new BucketLifecycleConfiguration()
                    .withRules(Arrays.asList(rule));

            // Use the client to set the LifecycleConfiguration on the bucket.
            _cosClient.setBucketLifecycleConfiguration(bucketName, configuration);
        }

        /**
         * @param bucketName
         * @param clientNum
         * @param api_key
         * @param service_instance_id
         * @param endpoint_url
         * @param location
         * @return AmazonS3
         */
        public static AmazonS3 createClient(String api_key, String service_instance_id, String endpoint_url, String location)
        {
            AWSCredentials credentials;
            credentials = new BasicIBMOAuthCredentials(api_key, service_instance_id);

            ClientConfiguration clientConfig = new ClientConfiguration().withRequestTimeout(5000);
            clientConfig.setUseTcpKeepAlive(true);

            AmazonS3 cosClient = AmazonS3ClientBuilder.standard().withCredentials(new AWSStaticCredentialsProvider(credentials))
                    .withEndpointConfiguration(new EndpointConfiguration(endpoint_url,
location)).withPathStyleAccessEnabled(true)
                    .withClientConfiguration(clientConfig).build();
            return cosClient;
        }
    }
```

Examine a bucket’s lifecycle configuration, including expiration

This implementation of the `GET` operation uses the `lifecycle` query parameter to examine lifecycle settings for the bucket. An HTTP `404` response will be returned if no lifecycle configuration is present.

Cloud IAM users must have the `Reader` role to examine a lifecycle policy from a bucket.

Classic Infrastructure Users must have `Read` permissions on the bucket to examine a lifecycle policy from a bucket.

Header	Type	Description
`Content-MD5`	String	**Required**: The base64 encoded 128-bit MD5 hash of the payload, which is used as an integrity check to ensure that the payload wasn't altered in transit.

Syntax

```
GET https://{endpoint}/{bucket}?lifecycle # path style
GET https://{bucket}.{endpoint}?lifecycle # virtual host style
```

Example Header Request

```
GET /images?lifecycle HTTP/1.1
Host: s3.us.cloud-object-storage.appdomain.cloud
Date: Wed, 7 Feb 2018 17:50:00 GMT
Authorization: authorization string
Content-Type: text/plain
Content-MD5: M625BaNwd/OytcM7O5gIaQ==
Content-Length: 305
```

Using the IBM Cloud® Object Storage SDKs only requires calling the appropriate functions with the correct parameters and proper configuration.

Node

```
var aws = require('ibm-cos-sdk');
var ep = new aws.Endpoint('s3.us-south.cloud-object-storage.appdomain.cloud');
var config = {
    endpoint: ep,
    apiKeyId: 'ZRZDoNoUseOLL7bRO8SAMPLEHPUzUL_-fsampleyYE',
    ibmAuthEndpoint: 'https://iam.cloud.ibm.com/identity/token',
    serviceInstanceId: 'crn:v1:bluemix:public:cloud-object-storage:global:a/<CREDENTIAL_ID_AS_GENERATED>:
<SERVICE_ID_AS_GENERATED>::',
};
var s3 = new aws.S3(config);

var params = {
  Bucket: 'STRING_VALUE' /* required */
};

s3.getBucketLifecycleConfiguration(params, function(err, data) {
  if (err) console.log(err, err.stack); // an error occurred
  else     console.log(data);           // successful response
});
```

Python

```
import sys
import ibm_boto3
from ibm_botocore.client import Config

api_key = "ZRZDoNoUseOLL7bRO8SAMPLEHPUzUL_-fsampleyYE"
service_instance_id = "85SAMPLE-eDOb-4NOT-bUSE-86nnnb31eaxx"
auth_endpoint = "https://iam.cloud.ibm.com/identity/token"
service_endpoint = "https://s3.us-south.cloud-object-storage.appdomain.cloud"

cos = ibm_boto3.resource('s3',
                         ibm_api_key_id=api_key,
                         ibm_service_instance_id=service_instance_id,
                         ibm_auth_endpoint=auth_endpoint,
                         config=Config(signature_version='oauth'),
                         endpoint_url=service_endpoint)

response = cos.Bucket('<name-of-bucket>').get_bucket_lifecycle_configuration(
    Bucket='string'
)
```

```
print("Bucket lifecyle: {0}".format(response))
```

Using the IBM Cloud® Object Storage SDKs only requires calling the appropriate functions with the correct parameters and proper configuration.

Java

```
package com.ibm.cloud;

    import java.sql.Timestamp;
    import java.util.List;
    import java.util.Arrays;

    import com.ibm.cloud.objectstorage.ClientConfiguration;
    import com.ibm.cloud.objectstorage.SDKGlobalConfiguration;
    import com.ibm.cloud.objectstorage.auth.AWSCredentials;
    import com.ibm.cloud.objectstorage.auth.AWSStaticCredentialsProvider;
    import com.ibm.cloud.objectstorage.client.builder.AwsClientBuilder.EndpointConfiguration;
    import com.ibm.cloud.objectstorage.services.s3.AmazonS3;
    import com.ibm.cloud.objectstorage.services.s3.AmazonS3ClientBuilder;
    import com.ibm.cloud.objectstorage.services.s3.model.Bucket;
    import com.ibm.cloud.objectstorage.services.s3.model.BucketLifecycleConfiguration;
    import com.ibm.cloud.objectstorage.services.s3.model.ListObjectsRequest;
    import com.ibm.cloud.objectstorage.services.s3.model.ObjectListing;
    import com.ibm.cloud.objectstorage.services.s3.model.S3ObjectSummary;
    import com.ibm.cloud.objectstorage.oauth.BasicIBMOAuthCredentials;

    public class App
    {
        private static AmazonS3 _cosClient;

        /**
         * @param args
         */
        public static void main(String[] args)
        {
            SDKGlobalConfiguration.IAM_ENDPOINT = "https://iam.cloud.ibm.com/identity/token";
            String bucketName = "<sample-bucket-name>";
            String api_key = "ZRZDoNoUseOLL7bRO8SAMPLEHPUzUL_-fsampleyYE";
            String service_instance_id = "85SAMPLE-eDOb-4NOT-bUSE-86nnnb31eaxx"
            String endpoint_url = "https://s3.us-south.cloud-object-storage.appdomain.cloud";

            String storageClass = "us-south";
            String location = "us";

            _cosClient = createClient(api_key, service_instance_id, endpoint_url, location);

            // Use the client to read the configuration
            BucketLifecycleConfiguration config = _cosClient.getBucketLifecycleConfiguration(bucketName);

            System.out.println(config.toString());
        }

        /**
         * @param bucketName
         * @param clientNum
         * @param api_key
         * @param service_instance_id
         * @param endpoint_url
         * @param location
         * @return AmazonS3
         */
        public static AmazonS3 createClient(String api_key, String service_instance_id, String endpoint_url, String location)
        {
            AWSCredentials credentials;
            credentials = new BasicIBMOAuthCredentials(api_key, service_instance_id);

            ClientConfiguration clientConfig = new ClientConfiguration().withRequestTimeout(5000);
            clientConfig.setUseTcpKeepAlive(true);
            AmazonS3 cosClient = AmazonS3ClientBuilder.standard().withCredentials(new AWSStaticCredentialsProvider(credentials))
                    .withEndpointConfiguration(new EndpointConfiguration(endpoint_url,
location)).withPathStyleAccessEnabled(true)
                    .withClientConfiguration(clientConfig).build();
            return cosClient;
        }

    }
```

Delete a bucket's lifecycle configuration, including expiration

This implementation of the `DELETE` operation uses the `lifecycle` query parameter to examine lifecycle settings for the bucket. All lifecycle rules associated with the bucket will be deleted. Transitions defined by the rules will no longer take place for new objects. However, existing transition rules will be maintained for objects that were already written to the bucket before the rules were deleted. Expiration Rules will no longer exist. An HTTP `404` response will be returned if no lifecycle configuration is present.

Cloud IAM users must have the `Writer` role to remove a lifecycle policy from a bucket.

Classic Infrastructure Users must have `Owner` permissions on the bucket to remove a lifecycle policy from a bucket.

Syntax

```
DELETE https://{endpoint}/{bucket}?lifecycle # path style
DELETE https://{bucket}.{endpoint}?lifecycle # virtual host style
```

Example Header Request

```
DELETE /images?lifecycle HTTP/1.1
Host: s3.us.cloud-object-storage.appdomain.cloud
Date: Wed, 7 Feb 2018 17:50:00 GMT
Authorization: authorization string
Content-Type: text/plain
Content-Length: 305
```

Using the IBM Cloud® Object Storage SDKs only requires calling the appropriate functions with the correct parameters and proper configuration.

Node

```
var aws = require('ibm-cos-sdk');
var ep = new aws.Endpoint('s3.us-south.cloud-object-storage.appdomain.cloud');
var config = {
    endpoint: ep,
    apiKeyId: 'ZRZDoNoUseOLL7bRO8SAMPLEHPUzUL_-fsampleyYE',
    ibmAuthEndpoint: 'https://iam.cloud.ibm.com/identity/token',
    serviceInstanceId: 'crn:v1:bluemix:public:cloud-object-storage:global:a/<CREDENTIAL_ID_AS_GENERATED>:
<SERVICE_ID_AS_GENERATED>::',
};
var s3 = new aws.S3(config);

var params = {
  Bucket: 'STRING_VALUE' /* required */
};

s3.deleteBucketLifecycleConfiguration(params, function(err, data) {
  if (err) console.log(err, err.stack); // an error occurred
  else     console.log(data);           // successful response
});
```

Using the IBM Cloud® Object Storage SDKs only requires calling the appropriate functions with the correct parameters and proper configuration.

Python

```
import sys
import ibm_boto3
from ibm_botocore.client import Config

api_key = "ZRZDoNoUseOLL7bRO8SAMPLEHPUzUL_-fsampleyYE"
service_instance_id = "85SAMPLE-eDOb-4NOT-bUSE-86nnnb31eaxx"
auth_endpoint = "https://iam.cloud.ibm.com/identity/token"
```

```
service_endpoint = "https://s3.us-south.cloud-object-storage.appdomain.cloud"

cos = ibm_boto3.resource('s3',
                         ibm_api_key_id=api_key,
                         ibm_service_instance_id=service_instance_id,
                         ibm_auth_endpoint=auth_endpoint,
                         config=Config(signature_version='oauth'),
                         endpoint_url=service_endpoint)

response = cos.Bucket('<name-of-bucket>').delete_bucket_lifecycle_configuration(
    Bucket='string'
)

print("Bucket lifecyle: {0}".format(response))
```

Using the IBM Cloud® Object Storage SDKs only requires calling the appropriate functions with the correct parameters and proper configuration.

Java

```
package com.ibm.cloud;

    import java.sql.Timestamp;
    import java.util.List;
    import java.util.Arrays;

    import com.ibm.cloud.objectstorage.ClientConfiguration;
    import com.ibm.cloud.objectstorage.SDKGlobalConfiguration;
    import com.ibm.cloud.objectstorage.auth.AWSCredentials;
    import com.ibm.cloud.objectstorage.auth.AWSStaticCredentialsProvider;
    import com.ibm.cloud.objectstorage.client.builder.AwsClientBuilder.EndpointConfiguration;
    import com.ibm.cloud.objectstorage.services.s3.AmazonS3;
    import com.ibm.cloud.objectstorage.services.s3.AmazonS3ClientBuilder;
    import com.ibm.cloud.objectstorage.services.s3.model.Bucket;
    import com.ibm.cloud.objectstorage.services.s3.model.BucketLifecycleConfiguration;
    import com.ibm.cloud.objectstorage.services.s3.model.ListObjectsRequest;
    import com.ibm.cloud.objectstorage.services.s3.model.ObjectListing;
    import com.ibm.cloud.objectstorage.services.s3.model.S3ObjectSummary;
    import com.ibm.cloud.objectstorage.oauth.BasicIBMOAuthCredentials;

    public class App
    {
        private static AmazonS3 _cosClient;

        /**
         * @param args
         */
        public static void main(String[] args)
        {
            SDKGlobalConfiguration.IAM_ENDPOINT = "https://iam.cloud.ibm.com/identity/token";
            String bucketName = "<sample-bucket-name>";
            String api_key = "ZRZDoNoUseOLL7bRO8SAMPLEHPUzUL_-fsampleyYE";
            String service_instance_id = "85SAMPLE-eDOb-4NOT-bUSE-86nnnb31eaxx"
            String endpoint_url = "https://s3.us-south.cloud-object-storage.appdomain.cloud";

            String storageClass = "us-south";
            String location = "us";

            _cosClient = createClient(api_key, service_instance_id, endpoint_url, location);

            // Delete the configuration.
            _cosClient.deleteBucketLifecycleConfiguration(bucketName);

            // Verify that the configuration has been deleted by attempting to retrieve it.
            config = _cosClient.getBucketLifecycleConfiguration(bucketName);
            String s = (config == null) ? "Configuration has been deleted." : "Configuration still exists.";
            System.out.println(s);
        }

        /**
         * @param bucketName
         * @param clientNum
         * @param api_key
         * @param service_instance_id
         * @param endpoint_url
         * @param location
         * @return AmazonS3
         */
        public static AmazonS3 createClient(String api_key, String service_instance_id, String endpoint_url, String location)
        {
            AWSCredentials credentials;
            credentials = new BasicIBMOAuthCredentials(api_key, service_instance_id);

            ClientConfiguration clientConfig = new ClientConfiguration().withRequestTimeout(5000);
            clientConfig.setUseTcpKeepAlive(true);

            AmazonS3 cosClient = AmazonS3ClientBuilder.standard().withCredentials(new AWSStaticCredentialsProvider(credentials))
                    .withEndpointConfiguration(new EndpointConfiguration(endpoint_url,
location)).withPathStyleAccessEnabled(true)
                    .withClientConfiguration(clientConfig).build();
            return cosClient;
        }

    }
```

Next Steps

Expiration is just one of many lifecycle concepts available for IBM Cloud Object Storage. Each of the concepts we've covered in this overview can be explored further at the IBM Cloud Platform.

Cleaning up incomplete multipart uploads

This lifecycle rule stops any multipart uploads if the uploads are not completed within a defined number of days after initiation.

You can set lifecycle rules for objects by using the web console, REST API, and third-party tools that are integrated with IBM Cloud Object Storage.

- A new rule can be added to a new or existing bucket at any time.
- An existing rule can be modified or disabled.

These incomplete uploads do not appear in the console, but the uploaded parts continue to accrue usage and billing charges. Setting up lifecycle rules to automatically delete incomplete uploads is the user's responsibility.

Attributes of expiration rules

Each expiration rule has the following attributes:

ID

A rule's ID must be unique within the bucket's lifecycle configuration.

AbortIncompleteMultipartUpload

The `AbortIncompleteMultipartUpload` block contains the details that govern the automatic cancellation of uploads. The block contains a single field: `DaysAfterInitiation`.

Prefix

An optional string that will be matched to the prefix of the object name in the bucket. A rule with a prefix will only apply to the objects that match. You can use multiple rules for different actions for different prefixes within the same bucket.

Status

A rule can either be enabled or disabled. A rule is active only when enabled.

Sample lifecycle configurations

This configuration expires any uploads that haven't completed after 3 days.

```
$ <LifecycleConfiguration>
```

```
    <Rule>
        <ID>delete-after-3-days</ID>
        <Filter />
        <Status>Enabled</Status>
        <AbortIncompleteMultipartUpload>
            <DaysAfterInitiation>3</DaysAfterInitiation>
        </AbortIncompleteMultipartUpload>
    </Rule>
</LifecycleConfiguration>
```

You can also combine rules. This configuration cancels inactive uploads after 5 days, archives any objects 90 days after creation, and deletes any objects with the prefix `foo/` after 180 days .

```
$ <LifecycleConfiguration>
    <Rule>
        <ID>archive-first</ID>
        <Filter />
        <Status>Enabled</Status>
    <Transition>
        <Days>90</Days>
        <StorageClass>GLACIER</StorageClass>
    </Transition>
    </Rule>
    <Rule>
        <ID>then-delete</ID>
    <Filter>
        <Prefix>foo/</Prefix>
    </Filter>
        <Status>Enabled</Status>
        <Expiration>
            <Days>180</Days>
        </Expiration>
    </Rule>
    <Rule>
        <ID>delete-after-3-days</ID>
        <Status>Enabled</Status>
  <AbortIncompleteMultipartUpload>
      <DaysAfterInitiation>3</DaysAfterInitiation>
        </AbortIncompleteMultipartUpload>
    </Rule>
</LifecycleConfiguration>
```

Using the API and SDKs

You can programmatically manage lifecycle rules by using the REST API or the IBM COS SDKs.

REST API reference

This implementation of the `PUT` operation uses the `lifecycle` query parameter to set lifecycle settings for the bucket. This operation allows for a single lifecycle policy definition for a bucket. The policy is defined as a set of rules consisting of the following parameters: `ID`, `Status`, `Filter`, and `Expiration`.

Cloud IAM users must have the `Writer` role to add a lifecycle policy from a bucket.

Classic Infrastructure Users must have `Owner` permissions on the bucket to add a lifecycle policy from a bucket.

Header	Type	Description
`Content-MD5`	String	**Required**: The base64 encoded 128-bit MD5 hash of the payload, which is used as an integrity check to ensure that the payload wasn't altered in transit.

Header

The body of the request must contain an XML block with the following schema:

Element	Type	Children	Ancestor	Constraint
`LifecycleConfiguration`	Container	`Rule`	None	Limit 1.

`Rule`	Container	`ID`, `Status`, `Filter`, `AbortIncompleteMultipartUpload`	`LifecycleConfiguration`	Limit 1000.
`ID`	String	None	`Rule`	Must consist of (`a-z`,`A-Z0-9`) and the following symbols: `!` `_` `.` `*` `'` `(` `)` `-`
`Filter`	String	`Prefix`	`Rule`	Must contain a `Prefix` element or be self-closed (`<Filter />`).
`Prefix`	String	None	`Filter`	The rule applies to any objects with keys that match this prefix.
`AbortIncompleteMultipartUpload`	`Container`	`DaysAfterInitiation`	`Rule`	Limit 1.
`DaysAfterInitiation`	Non-negative integer	None	`AbortIncompleteMultipartUpload`	Must be a value greater than 0.

Body of the request schema

The body of the request must contain an XML block with the schema that is addressed in the table (see Example 1).

```
<LifecycleConfiguration>
 <Rule>
  <ID>delete-after-3-days</ID>
        <Filter />
  <Status>Enabled</Status>
  <AbortIncompleteMultipartUpload>
   <DaysAfterInitiation>3</DaysAfterInitiation>
  </AbortIncompleteMultipartUpload>
 </Rule>
</LifecycleConfiguration>
```

Syntax

```
PUT https://{endpoint}/{bucket}?lifecycle # path style
PUT https://{bucket}.{endpoint}?lifecycle # virtual host style
```

Example request

```
PUT /images?lifecycle HTTP/1.1
Host: s3.us.cloud-object-storage.appdomain.cloud
Date: Wed, 7 Feb 2018 17:50:00 GMT
Authorization: authorization string
Content-Type: text/plain
Content-MD5: M625BaNwd/OytcM7O5gIaQ==
Content-Length: 305

<LifecycleConfiguration>
 <Rule>
  <ID>delete-after-3-days</ID>
        <Filter />
  <Status>Enabled</Status>
  <AbortIncompleteMultipartUpload>
   <DaysAfterInitiation>3</DaysAfterInitiation>
  </AbortIncompleteMultipartUpload>
 </Rule>
</LifecycleConfiguration>
```

Code sample for use with NodeJS COS SDK

Using the IBM Cloud® Object Storage SDKs only requires calling the appropriate functions with the correct parameters and proper configuration.

```
var aws = require('ibm-cos-sdk');
var ep = new aws.Endpoint('s3.us-south.cloud-object-storage.appdomain.cloud');
var config = {
    endpoint: ep,
    apiKeyId: 'ZRZDoNoUseOLL7bRO8SAMPLEHPUzUL_-fsampleyYE',
    ibmAuthEndpoint: 'https://iam.cloud.ibm.com/identity/token',
    serviceInstanceId: 'crn:v1:bluemix:public:cloud-object-storage:global:a/<CREDENTIAL_ID_AS_GENERATED>:
<SERVICE_ID_AS_GENERATED>::',
};
var s3 = new aws.S3(config);

var params = {
  Bucket: 'STRING_VALUE', /* required */
  LifecycleConfiguration: {
    Rules: [ /* required */
      {
        Status: 'Enabled', /* required */
        ID: 'OPTIONAL_STRING_VALUE',
        Filter: {}, /* required */
        AbortIncompleteMultipartUpload: {
          DaysAfterInitiation: 'NUMBER_VALUE'
        }
      },
    ]
  }
};

s3.putBucketLifecycleConfiguration(params, function(err, data) {
  if (err) console.log(err, err.stack); // an error occurred
  else     console.log(data);           // successful response
});
```

Code sample for use with Python COS SDK

Using the IBM Cloud® Object Storage SDKs only requires calling the appropriate functions with the correct parameters and proper configuration.

```
import sys
import ibm_boto3
from ibm_botocore.client import Config

api_key = "ZRZDoNoUseOLL7bRO8SAMPLEHPUzUL_-fsampleyYE"
service_instance_id = "85SAMPLE-eDOb-4NOT-bUSE-86nnnb31eaxx"
auth_endpoint = "https://iam.cloud.ibm.com/identity/token"
service_endpoint = "https://s3.us-south.cloud-object-storage.appdomain.cloud"

cos = ibm_boto3.resource('s3',
                         ibm_api_key_id=api_key,
                         ibm_service_instance_id=service_instance_id,
                         ibm_auth_endpoint=auth_endpoint,
                         config=Config(signature_version='oauth'),
                         endpoint_url=service_endpoint)

response = cos.Bucket('<name-of-bucket>').put_bucket_lifecycle_configuration(
    Bucket='string',
    LifecycleConfiguration={
        'Rules': [
            {
                'Status': 'Enabled',
                'Filter': {},
                'AbortIncompleteMultipartUpload': {
                    'DaysAfterInitiation': <NUMBER_VALUE>
        }
```

```
            },
        ]
    }
)

print("Bucket lifecyle: {0}".format(response))
```

Code sample for use with Java COS SDK

Using the IBM Cloud® Object Storage SDKs only requires calling the appropriate functions with the correct parameters and proper configuration.

```
package com.ibm.cloud;

    import java.sql.Timestamp;
    import java.util.List;
    import java.util.Arrays;

    import com.ibm.cloud.objectstorage.ClientConfiguration;
    import com.ibm.cloud.objectstorage.SDKGlobalConfiguration;
    import com.ibm.cloud.objectstorage.auth.AWSCredentials;
    import com.ibm.cloud.objectstorage.auth.AWSStaticCredentialsProvider;
    import com.ibm.cloud.objectstorage.client.builder.AwsClientBuilder.EndpointConfiguration;
    import com.ibm.cloud.objectstorage.services.s3.AmazonS3;
    import com.ibm.cloud.objectstorage.services.s3.AmazonS3ClientBuilder;
    import com.ibm.cloud.objectstorage.services.s3.model.Bucket;
    import com.ibm.cloud.objectstorage.services.s3.model.BucketLifecycleConfiguration;
    import com.ibm.cloud.objectstorage.services.s3.model.ListObjectsRequest;
    import com.ibm.cloud.objectstorage.services.s3.model.ObjectListing;
    import com.ibm.cloud.objectstorage.services.s3.model.S3ObjectSummary;
    import com.ibm.cloud.objectstorage.oauth.BasicIBMOAuthCredentials;

    public class App
    {
        private static AmazonS3 _cosClient;

        /**
         * @param args
         */
        public static void main(String[] args)
        {
            SDKGlobalConfiguration.IAM_ENDPOINT = "https://iam.cloud.ibm.com/identity/token";
            String bucketName = "<sample-bucket-name>";
            String api_key = "ZRZDoNoUseOLL7bRO8SAMPLEHPUzUL_-fsampleyYE";
            String service_instance_id = "85SAMPLE-eDOb-4NOT-bUSE-86nnnb31eaxx"
            String endpoint_url = "https://s3.us-south.cloud-object-storage.appdomain.cloud";
            String storageClass = "us-south";
            String location = "us";

            _cosClient = createClient(api_key, service_instance_id, endpoint_url, location);

            // Define a rule for expiring items in a bucket
            int days_to_delete = 10;
            BucketLifecycleConfiguration.Rule rule = new BucketLifecycleConfiguration.Rule()
                    .withId("Delete rule")
                    .withExpirationInDays(days_to_delete)
                    .withStatus(BucketLifecycleConfiguration.ENABLED);

                    rule.setFilter(new LifecycleFilter());

            // Add the rule to a new BucketLifecycleConfiguration.
            BucketLifecycleConfiguration configuration = new BucketLifecycleConfiguration()
                    .withRules(Arrays.asList(rule));

            // Use the client to set the LifecycleConfiguration on the bucket.
            _cosClient.setBucketLifecycleConfiguration(bucketName, configuration);
        }

        /**
         * @param bucketName
         * @param clientNum
         * @param api_key
         * @param service_instance_id
         * @param endpoint_url
         * @param location
         * @return AmazonS3
         */
        public static AmazonS3 createClient(String api_key, String service_instance_id, String endpoint_url, String location)
        {
            AWSCredentials credentials;
            credentials = new BasicIBMOAuthCredentials(api_key, service_instance_id);

            ClientConfiguration clientConfig = new ClientConfiguration().withRequestTimeout(5000);
            clientConfig.setUseTcpKeepAlive(true);

            AmazonS3 cosClient = AmazonS3ClientBuilder.standard().withCredentials(new AWSStaticCredentialsProvider(credentials))
                    .withEndpointConfiguration(new EndpointConfiguration(endpoint_url,
location)).withPathStyleAccessEnabled(true)
                    .withClientConfiguration(clientConfig).build();
            return cosClient;
        }
    }
```

Examine a bucket’s lifecycle configuration, including expiration

This implementation of the `GET` operation uses the `lifecycle` query parameter to examine lifecycle settings for the bucket. An HTTP `404` response will be returned if no lifecycle configuration is present.

Cloud IAM users must have the `Reader` role to examine a lifecycle policy from a bucket.

Classic Infrastructure Users must have `Read` permissions on the bucket to examine a lifecycle policy from a bucket.

Syntax

```
GET https://{endpoint}/{bucket}?lifecycle # path style
GET https://{bucket}.{endpoint}?lifecycle # virtual host style
```

Example Header Request

```
GET /images?lifecycle HTTP/1.1
Host: s3.us.cloud-object-storage.appdomain.cloud
Date: Wed, 7 Feb 2018 17:50:00 GMT
Authorization: authorization string
Content-Type: text/plain
Content-Length: 305
```

Using the IBM Cloud® Object Storage SDKs only requires calling the appropriate functions with the correct parameters and proper configuration.

```
var aws = require('ibm-cos-sdk');
var ep = new aws.Endpoint('s3.us-south.cloud-object-storage.appdomain.cloud');
var config = {
    endpoint: ep,
    apiKeyId: 'ZRZDoNoUseOLL7bRO8SAMPLEHPUzUL_-fsampleyYE',
    ibmAuthEndpoint: 'https://iam.cloud.ibm.com/identity/token',
    serviceInstanceId: 'crn:v1:bluemix:public:cloud-object-storage:global:a/<CREDENTIAL_ID_AS_GENERATED>:
<SERVICE_ID_AS_GENERATED>::',
};
var s3 = new aws.S3(config);

var params = {
  Bucket: 'STRING_VALUE' /* required */
};

s3.getBucketLifecycleConfiguration(params, function(err, data) {
  if (err) console.log(err, err.stack); // an error occurred
  else     console.log(data);           // successful response
});
```

```
import sys
import ibm_boto3
```

```
from ibm_botocore.client import Config

api_key = "ZRZDoNoUseOLL7bRO8SAMPLEHPUzUL_-fsampleyYE"
service_instance_id = "85SAMPLE-eDOb-4NOT-bUSE-86nnnb31eaxx"
auth_endpoint = "https://iam.cloud.ibm.com/identity/token"
service_endpoint = "https://s3.us-south.cloud-object-storage.appdomain.cloud"

cos = ibm_boto3.resource('s3',
                         ibm_api_key_id=api_key,
                         ibm_service_instance_id=service_instance_id,
                         ibm_auth_endpoint=auth_endpoint,
                         config=Config(signature_version='oauth'),
                         endpoint_url=service_endpoint)

response = cos.Bucket('<name-of-bucket>').get_bucket_lifecycle_configuration(
    Bucket='string'
)

print("Bucket lifecyle: {0}".format(response))
```

Using the IBM Cloud® Object Storage SDKs only requires calling the appropriate functions with the correct parameters and proper configuration.

Java

```
package com.ibm.cloud;

    import java.sql.Timestamp;
    import java.util.List;
    import java.util.Arrays;

    import com.ibm.cloud.objectstorage.ClientConfiguration;
    import com.ibm.cloud.objectstorage.SDKGlobalConfiguration;
    import com.ibm.cloud.objectstorage.auth.AWSCredentials;
    import com.ibm.cloud.objectstorage.auth.AWSStaticCredentialsProvider;
    import com.ibm.cloud.objectstorage.client.builder.AwsClientBuilder.EndpointConfiguration;
    import com.ibm.cloud.objectstorage.services.s3.AmazonS3;
    import com.ibm.cloud.objectstorage.services.s3.AmazonS3ClientBuilder;
    import com.ibm.cloud.objectstorage.services.s3.model.Bucket;
    import com.ibm.cloud.objectstorage.services.s3.model.BucketLifecycleConfiguration;
    import com.ibm.cloud.objectstorage.services.s3.model.ListObjectsRequest;
    import com.ibm.cloud.objectstorage.services.s3.model.ObjectListing;
    import com.ibm.cloud.objectstorage.services.s3.model.S3ObjectSummary;
    import com.ibm.cloud.objectstorage.oauth.BasicIBMOAuthCredentials;

    public class App
    {
        private static AmazonS3 _cosClient;

        /**
         * @param args
         */
        public static void main(String[] args)
        {
            SDKGlobalConfiguration.IAM_ENDPOINT = "https://iam.cloud.ibm.com/identity/token";
            String bucketName = "<sample-bucket-name>";
            String api_key = "ZRZDoNoUseOLL7bRO8SAMPLEHPUzUL_-fsampleyYE";
            String service_instance_id = "85SAMPLE-eDOb-4NOT-bUSE-86nnnb31eaxx"
            String endpoint_url = "https://s3.us-south.cloud-object-storage.appdomain.cloud";

            String storageClass = "us-south";
            String location = "us";

            _cosClient = createClient(api_key, service_instance_id, endpoint_url, location);

            // Use the client to read the configuration
            BucketLifecycleConfiguration config = _cosClient.getBucketLifecycleConfiguration(bucketName);

            System.out.println(config.toString());
        }
        /**
         * @param bucketName
         * @param clientNum
         * @param api_key
         * @param service_instance_id
         * @param endpoint_url
         * @param location
         * @return AmazonS3
         */
        public static AmazonS3 createClient(String api_key, String service_instance_id, String endpoint_url, String location)
        {
            AWSCredentials credentials;
            credentials = new BasicIBMOAuthCredentials(api_key, service_instance_id);

            ClientConfiguration clientConfig = new ClientConfiguration().withRequestTimeout(5000);
            clientConfig.setUseTcpKeepAlive(true);

            AmazonS3 cosClient = AmazonS3ClientBuilder.standard().withCredentials(new AWSStaticCredentialsProvider(credentials))
                    .withEndpointConfiguration(new EndpointConfiguration(endpoint_url,
location)).withPathStyleAccessEnabled(true)
                    .withClientConfiguration(clientConfig).build();
            return cosClient;
        }

    }
```

Delete a bucket’s lifecycle configuration, including expiration

This implementation of the `DELETE` operation uses the `lifecycle` query parameter to examine lifecycle settings for the bucket. All lifecycle rules associated with the bucket will be deleted. Transitions defined by the rules will no longer take place for new objects. However, existing transition rules will be maintained for objects that were already written to the bucket before the rules were deleted. Expiration Rules will no longer exist. An HTTP `404` response will be returned if no lifecycle configuration is present.

Cloud IAM users must have the `Writer` role to remove a lifecycle policy from a bucket.

Classic Infrastructure Users must have `Owner` permissions on the bucket to remove a lifecycle policy from a bucket.

Syntax

```
DELETE https://{endpoint}/{bucket}?lifecycle # path style
DELETE https://{bucket}.{endpoint}?lifecycle # virtual host style
```

Example Header Request

```
DELETE /images?lifecycle HTTP/1.1
Host: s3.us.cloud-object-storage.appdomain.cloud
Date: Wed, 7 Feb 2018 17:50:00 GMT
Authorization: authorization string
Content-Type: text/plain
Content-Length: 305
```

Using the IBM Cloud® Object Storage SDKs only requires calling the appropriate functions with the correct parameters and proper configuration.

```
var aws = require('ibm-cos-sdk');
var ep = new aws.Endpoint('s3.us-south.cloud-object-storage.appdomain.cloud');
var config = {
    endpoint: ep,
    apiKeyId: 'ZRZDoNoUseOLL7bRO8SAMPLEHPUzUL_-fsampleyYE',
    ibmAuthEndpoint: 'https://iam.cloud.ibm.com/identity/token',
    serviceInstanceId: 'crn:v1:bluemix:public:cloud-object-storage:global:a/<CREDENTIAL_ID_AS_GENERATED>:
<SERVICE_ID_AS_GENERATED>::',
};
var s3 = new aws.S3(config);

var params = {
  Bucket: 'STRING_VALUE' /* required */
};

s3.deleteBucketLifecycleConfiguration(params, function(err, data) {
  if (err) console.log(err, err.stack); // an error occurred
  else     console.log(data);           // successful response
});
```

Using the IBM Cloud® Object Storage SDKs only requires calling the appropriate functions with the correct parameters and proper configuration.

```
import sys
import ibm_boto3
from ibm_botocore.client import Config

api_key = "ZRZDoNoUseOLL7bRO8SAMPLEHPUzUL_-fsampleyYE"
service_instance_id = "85SAMPLE-eDOb-4NOT-bUSE-86nnnb31eaxx"
auth_endpoint = "https://iam.cloud.ibm.com/identity/token"
service_endpoint = "https://s3.us-south.cloud-object-storage.appdomain.cloud"

cos = ibm_boto3.resource('s3',
                         ibm_api_key_id=api_key,
                         ibm_service_instance_id=service_instance_id,
                         ibm_auth_endpoint=auth_endpoint,
                         config=Config(signature_version='oauth'),
                         endpoint_url=service_endpoint)

response = cos.Bucket('<name-of-bucket>').delete_bucket_lifecycle_configuration(
    Bucket='string'
)

print("Bucket lifecyle: {0}".format(response))
```

Using the IBM Cloud® Object Storage SDKs only requires calling the appropriate functions with the correct parameters and proper configuration.

```
package com.ibm.cloud;

    import java.sql.Timestamp;
    import java.util.List;
    import java.util.Arrays;

    import com.ibm.cloud.objectstorage.ClientConfiguration;
    import com.ibm.cloud.objectstorage.SDKGlobalConfiguration;
    import com.ibm.cloud.objectstorage.auth.AWSCredentials;
    import com.ibm.cloud.objectstorage.auth.AWSStaticCredentialsProvider;
    import com.ibm.cloud.objectstorage.client.builder.AwsClientBuilder.EndpointConfiguration;
    import com.ibm.cloud.objectstorage.services.s3.AmazonS3;
    import com.ibm.cloud.objectstorage.services.s3.AmazonS3ClientBuilder;
    import com.ibm.cloud.objectstorage.services.s3.model.Bucket;
    import com.ibm.cloud.objectstorage.services.s3.model.BucketLifecycleConfiguration;
    import com.ibm.cloud.objectstorage.services.s3.model.ListObjectsRequest;
    import com.ibm.cloud.objectstorage.services.s3.model.ObjectListing;
    import com.ibm.cloud.objectstorage.services.s3.model.S3ObjectSummary;
    import com.ibm.cloud.objectstorage.oauth.BasicIBMOAuthCredentials;

    public class App
    {
        private static AmazonS3 _cosClient;

        /**
         * @param args
         */
        public static void main(String[] args)
        {
            SDKGlobalConfiguration.IAM_ENDPOINT = "https://iam.cloud.ibm.com/identity/token";
            String bucketName = "<sample-bucket-name>";
            String api_key = "ZRZDoNoUseOLL7bRO8SAMPLEHPUzUL_-fsampleyYE";
            String service_instance_id = "85SAMPLE-eDOb-4NOT-bUSE-86nnnb31eaxx"
            String endpoint_url = "https://s3.us-south.cloud-object-storage.appdomain.cloud";

            String storageClass = "us-south";
            String location = "us";

            _cosClient = createClient(api_key, service_instance_id, endpoint_url, location);

            // Delete the configuration.
            _cosClient.deleteBucketLifecycleConfiguration(bucketName);

            // Verify that the configuration has been deleted by attempting to retrieve it.
            config = _cosClient.getBucketLifecycleConfiguration(bucketName);
            String s = (config == null) ? "Configuration has been deleted." : "Configuration still exists.";
            System.out.println(s);
        }

        /**
         * @param bucketName
         * @param clientNum
         * @param api_key
         * @param service_instance_id
         * @param endpoint_url
         * @param location
         * @return AmazonS3
         */
        public static AmazonS3 createClient(String api_key, String service_instance_id, String endpoint_url, String location)
        {
            AWSCredentials credentials;
            credentials = new BasicIBMOAuthCredentials(api_key, service_instance_id);

            ClientConfiguration clientConfig = new ClientConfiguration().withRequestTimeout(5000);
            clientConfig.setUseTcpKeepAlive(true);

            AmazonS3 cosClient = AmazonS3ClientBuilder.standard().withCredentials(new AWSStaticCredentialsProvider(credentials))
                    .withEndpointConfiguration(new EndpointConfiguration(endpoint_url,
location)).withPathStyleAccessEnabled(true)
                    .withClientConfiguration(clientConfig).build();
            return cosClient;
        }

    }
```

Next Steps

Expiration is just one of many lifecycle concepts available for IBM Cloud Object Storage. Each of the concepts we've covered in this overview can be explored further at the IBM Cloud Platform.

Managing data immutability

Using Immutable Object Storage to protect buckets

Immutable Object Storage preserves electronic records and maintains data integrity. Retention policies ensure that data is stored in a WORM (Write-Once-Read-Many), non-erasable and non-rewritable manner. This policy is enforced until the end of a retention period and the removal of any legal holds.

Note: This feature is not currently supported in Object Storage for Satellite. Learn more.

Important: Policies are enforced until the end of a retention period, and can not be altered until the retention period has expired. While IBM Cloud Object Storage makes use of the S3 API for most operations, the APIs used for configuring retention policies is not the same as the S3 API, although some terminology may be shared. Read this documentation carefully to prevent any users in your organization from creating objects that can not be deleted, even by IBM Cloud administrators.

This feature can be used by any user that needs long-term data retention in their environment, including but not limited to organizations in the following industries:

- Financial
- Healthcare
- Media content archives
- Anyone looking to prevent privileged modification or deletion of objects or documents

Retention policies can also be used by organizations that deal with financial records management, such as broker-dealer transactions, and might need to store data in a non-rewritable and non-erasable format.

Note: Immutable Object Storage is available in certain regions only, see [Integrated Services] for details. It also requires a Standard pricing plan. See [pricing] for details.

Important: It isn't possible to use Aspera high-speed transfer with buckets with a retention policy.

Terminology and usage

Retention period

The duration of time an object must be stored in the IBM Cloud Object Storage bucket.

Retention policy

A retention policy is enabled at the IBM Cloud Object Storage bucket level. Minimum, maximum and default retention period are defined by this policy and apply to all objects in the bucket.

Minimum retention period is the minimum duration of time an object must be kept unmodified in the bucket.

Maximum retention period is the maximum duration of time an object can be kept unmodified in the bucket.

If an object is stored in the bucket without specifying a custom retention period, the default retention period is used. The minimum retention period must be less than or equal to the default retention period, which in turn must be less than or equal to the maximum retention period.

Tip: A maximum retention period of 1000 years can be specified for the objects.

Important: To create a retention policy on a bucket, you need Manager role. See [Bucket permissions] for more details.

Legal hold

Certain objects might need to be prevented from modification after a retention period expires. An example is an incomplete legal review, where records might need to be accessible for an extended duration beyond the retention period originally set. A legal hold flag can then be applied at the object level. Legal holds can be applied to objects during initial uploads or after an object is written. Note: A maximum of 100 legal holds can be applied per object.

Indefinite retention

Allows the user to set the object to be stored indefinitely until a new retention period is applied. This is set at a per object level.

Important: An object that is retained using indefinite retention is not entirely immutable until the object's retention has been converted from -1 to some positive finite value. While an object written with -1 (Indefinite Retention) cannot be deleted uisng the `DELETE object` or overwritten request, the object can still have the retention updated from -1 to the current date/time, which would make the object immediately deleteable.

Users should consider this behavior when assessing its viability for their storage needs. A common use case for Indefinite Retention is described in [Event-based retention]. In or want to use event-based retention, Immutable Object Storage allows users to set indefinite retention on the object if they are unsure of the retention needs when the object is first uploaded to the system. Once set to indefinite, user applications can then can change the object retention to a finite value when a certain event has taken place.

Example Given that a company has a policy of retaining employee records for three years after the employee leaves the company.

- When an employee joins to start working for a complany, the records that are associated with that employee can be indefinitely retained.
- And when that same employee leaves the company, the indefinite retention is then converted to a finite value of three years from the current time, which is defined by company policy.

Note: A user or third-party application can change the retention period from indefinite to finite retention that uses an SDK or REST API.

Note: Bucket owners and permitted users can limit the new retention that can be configured for an object that is currently retained using indefinite retention. This is done by utilizing the bucket retention minimum and maximum allowed values. By doing so, users can prevent a case where an object retained with indefinite retention has its retention updated to the current time so that it is immediately deletable.

Event-based retention

Immutable Object Storage allows users to set indefinite retention on the object if they are unsure of the final duration of the retention period, or want to use event-based retention. Once set to indefinite, user applications can then can change the object retention to a finite value later. For example, a company has a policy of retaining employee records for three years after the employee leaves the company. When an employee joins the company, the records that are associated with that employee can be indefinitely retained. When the employee leaves the company, the indefinite retention is converted to a finite value of three years from the current time, as defined by company policy. The object is then protected for three years after the retention period change. A user or third-party application can change the retention period from indefinite to finite retention that uses an SDK or REST API.

Permanent retention

⚠ **Important:** Permanent retention ensures that data can not be deleted, ever, by anyone. Read the documentation carefully and do not use permanent retention unless there is a compelling regulatory or compliance need for **permanent** data storage.

Permanent retention can only be enabled at an IBM Cloud Object Storage bucket level with retention policy enabled and users are able to select the permanent retention period option during object uploads. Once enabled, this process can't be reversed and objects uploaded that use a permanent retention period **cannot be deleted**. It's the responsibility of the users to validate at their end if there's a legitimate need to **permanently** store objects by using Object Storage buckets with a retention policy.

⚠ **Important:** When using Immutable Object Storage, you are responsible for ensuring that your IBM Cloud Account is kept in good standing per IBM Cloud policies and guidelines for as long as the data is subject to a retention policy. Refer to IBM Cloud Service terms for more information.

Immutable Object Storage and considerations for various regulations

When using immutable Object Storage, it is the client's responsibility to check for and ensure whether any of the feature capabilities that are discussed can be used to satisfy and comply with the key rules around electronic records storage and retention that is generally governed by:

- Securities and Exchange Commission (SEC) Rule 17a-4(f),
- Financial Industry Regulatory Authority (FINRA) Rule 4511(c), and
- Commodity Futures Trading Commission (CFTC) Rule 1.31(c)-(d)

To assist clients in making informed decisions, IBM engaged Cohasset Associates Inc. to conduct an independent assessment of IBM's Immutable Object Storage. Review Cohasset Associates Inc.'s report that provides details on the assessment of the Immutable Object Storage feature of IBM Cloud Object Storage.

Audit of access and transactions

Access log data for Immutable Object Storage to review changes to retention parameters, object retention period, and application of legal holds is available on a case-by-case basis by opening a customer service ticket.

Using the console

Retention policies can be added to new or existing empty buckets, and cannot be removed. For a new bucket, ensure that you are creating the bucket in a supported region, and then choose the **Add retention policy** option. For an existing bucket, ensure that it has no objects and then navigate to configuration settings and click the **Create policy** button below the bucket retention policy section. In either case, set a minimum, maximum, and default retention periods.

Using the REST API, Libraries, and SDKs

Several new APIs have been introduced to the IBM Cloud Object Storage SDKs to provide support for applications working with retention policies. Select a language (HTTP, Java, JavaScript, or Python) at the beginning of this page to view examples that use the appropriate Object Storage SDK.

All code examples assume the existence of a client object that is called `cos` that can call the different methods. For details on creating clients, see the specific SDK guides.

Note: All date values used to set retention periods are Greenwich mean time. A `Content-MD5` header is required to ensure data integrity, and is automatically sent when using an SDK.

Add a retention policy on an existing bucket

This implementation of the `PUT` operation uses the `protection` query parameter to set the retention parameters for an existing bucket. This operation allows you to set or change the minimum, default, and maximum retention period. This operation also allows you to change the protection state of the bucket.

Objects written to a protected bucket cannot be deleted until the protection period has expired and all legal holds on the object are removed. The bucket's default retention value is given to an object unless an object-specific value is provided when the object is created. Objects in protected buckets that are no longer under retention (retention period has expired and the object does not have any legal holds), when overwritten, will again come under retention. The new retention period can be provided as part of the object overwrite request or the default retention time of the bucket will be given to the object.

The minimum and maximum supported values for the retention period settings `MinimumRetention`, `DefaultRetention`, and `MaximumRetention` are a minimum of 0 days and a maximum of 365243 days (1000 years).

A `Content-MD5` header is required. This operation does not make use of extra query parameters.

Tip: For more information about endpoints, see Endpoints and storage locations

Syntax

```
PUT https://{endpoint}/{bucket-name}?protection= # path style
PUT https://{bucket-name}.{endpoint}?protection= # virtual host style
```

Example request

```
PUT /example-bucket?protection= HTTP/1.1
Authorization: {authorization-string}
x-amz-date: 20181011T190354Z
x-amz-content-sha256: 2938f51643d63c864fdbea618fe71b13579570a86f39da2837c922bae68d72df
Content-MD5: GQmpTNpru0yK6YrxHnpj7g==
Content-Type: text/plain
Host: 67.228.254.193
Content-Length: 299
<ProtectionConfiguration>
  <Status>Retention</Status>
  <MinimumRetention>
    <Days>100</Days>
  </MinimumRetention>
  <MaximumRetention>
    <Days>10000</Days>
  </MaximumRetention>
  <DefaultRetention>
    <Days>2555</Days>
  </DefaultRetention>
</ProtectionConfiguration>
```

Example response

```
HTTP/1.1 200 OK
Date: Wed, 5 Oct 2018 15:39:38 GMT
X-Clv-Request-Id: 7afca6d8-e209-4519-8f2c-1af3f1540b42
Accept-Ranges: bytes
Server: Cleversafe/3.14.1
X-Clv-S3-Version: 2.5
x-amz-request-id: 7afca6d8-e209-4519-8f2c-1af3f1540b42
Content-Length: 0
```

Python

```
def add_protection_configuration_to_bucket(bucket_name):
    try:
        new_protection_config = {
            "Status": "Retention",
            "MinimumRetention": {"Days": 10},
            "DefaultRetention": {"Days": 100},
            "MaximumRetention": {"Days": 1000}
        }

        cos.put_bucket_protection_configuration(Bucket=bucket_name, ProtectionConfiguration=new_protection_config)

        print("Protection added to bucket {0}\n".format(bucket_name))
    except ClientError as be:
        print("CLIENT ERROR: {0}\n".format(be))
    except Exception as e:
        print("Unable to set bucket protection config: {0}".format(e))
```

Node

```
function addProtectionConfigurationToBucket(bucketName) {
    console.log(`Adding protection to bucket ${bucketName}`);
    return cos.putBucketProtectionConfiguration({
        Bucket: bucketName,
        ProtectionConfiguration: {
            'Status': 'Retention',
            'MinimumRetention': {'Days': 10},
            'DefaultRetention': {'Days': 100},
            'MaximumRetention': {'Days': 1000}
        }
    }).promise()
    .then(() => {
        console.log(`Protection added to bucket ${bucketName}!`);
    })
    .catch((e) => {
        console.log(`ERROR: ${e.code} - ${e.message}\n`);
    });
}
```

Java

```
public static void addProtectionConfigurationToBucket(String bucketName) {
    System.out.printf("Adding protection to bucket: %s\n", bucketName);

    BucketProtectionConfiguration newConfig = new BucketProtectionConfiguration()
        .withStatus(BucketProtectionStatus.Retention)
        .withMinimumRetentionInDays(10)
        .withDefaultRetentionInDays(100)
        .withMaximumRetentionInDays(1000);

    cos.setBucketProtection(bucketName, newConfig);

    System.out.printf("Protection added to bucket %s\n", bucketName);
}

public static void addProtectionConfigurationToBucketWithRequest(String bucketName) {
    System.out.printf("Adding protection to bucket: %s\n", bucketName);

    BucketProtectionConfiguration newConfig = new BucketProtectionConfiguration()
        .withStatus(BucketProtectionStatus.Retention)
        .withMinimumRetentionInDays(10)
        .withDefaultRetentionInDays(100)
        .withMaximumRetentionInDays(1000);

    SetBucketProtectionConfigurationRequest newRequest = new SetBucketProtectionConfigurationRequest()
        .withBucketName(bucketName)
        .withProtectionConfiguration(newConfig);

    cos.setBucketProtectionConfiguration(newRequest);

    System.out.printf("Protection added to bucket %s\n", bucketName);
}
```

Check retention policy on a bucket

This implementation of a GET operation fetches the retention parameters for an existing bucket.

Syntax

```
GET https://{endpoint}/{bucket-name}?protection= # path style
GET https://{bucket-name}.{endpoint}?protection= # virtual host style
```

Example request

```
GET /example-bucket?protection= HTTP/1.1
Authorization: {authorization-string}
x-amz-date: 20181011T190354Z
```

```
Content-Type: text/plain
Host: 67.228.254.193
```

Example response

```
HTTP/1.1 200 OK
Date: Wed, 5 Oct 2018 15:39:38 GMT
X-Clv-Request-Id: 7afca6d8-e209-4519-8f2c-1af3f1540b42
Accept-Ranges: bytes
Server: Cleversafe/3.13.1
X-Clv-S3-Version: 2.5
x-amz-request-id: 7afca6d8-e209-4519-8f2c-1af3f1540b42
Content-Length: 299
<ProtectionConfiguration>
  <Status>Retention</Status>
  <MinimumRetention>
    <Days>100</Days>
  </MinimumRetention>
  <MaximumRetention>
    <Days>10000</Days>
  </MaximumRetention>
  <DefaultRetention>
    <Days>2555</Days>
  </DefaultRetention>
</ProtectionConfiguration>
```

If there is no protection configuration on the bucket, the server responds with disabled status instead.

```
<ProtectionConfiguration>
  <Status>Disabled</Status>
</ProtectionConfiguration>
```

Python

```
def get_protection_configuration_on_bucket(bucket_name):
    try:
        response = cos.get_bucket_protection_configuration(Bucket=bucket_name)
        protection_config = response.get("ProtectionConfiguration")

        print("Bucket protection config for {0}\n".format(bucket_name))
        print(protection_config)
        print("\n")
    except ClientError as be:
        print("CLIENT ERROR: {0}\n".format(be))
    except Exception as e:
        print("Unable to get bucket protection config: {0}".format(e))
```

Node

```
function getProtectionConfigurationOnBucket(bucketName) {
    console.log(`Retrieve the protection on bucket ${bucketName}`);
    return cos.getBucketProtectionConfiguration({
        Bucket: bucketName
    }).promise()
    .then((data) => {
        console.log(`Configuration on bucket ${bucketName}:`);
        console.log(data);
    }
    .catch((e) => {
        console.log(`ERROR: ${e.code} - ${e.message}\n`);
    });
}
```

Java

```
public static void getProtectionConfigurationOnBucket(String bucketName) {
    System.out.printf("Retrieving protection configuration from bucket: %s\n", bucketName;
    BucketProtectionConfiguration config = cos.getBucketProtection(bucketName);

    String status = config.getStatus();

    System.out.printf("Status: %s\n", status);

    if (!status.toUpperCase().equals("DISABLED")) {
        System.out.printf("Minimum Retention (Days): %s\n", config.getMinimumRetentionInDays());
        System.out.printf("Default Retention (Days): %s\n", config.getDefaultRetentionInDays());
        System.out.printf("Maximum Retention (Days): %s\n", config.getMaximumRetentionInDays());
    }
}
```

Upload an object to a bucket with retention policy

This enhancement of the `PUT` operation adds three new request headers: two for specifying the retention period in different ways, and one for adding a single legal hold to the new object. New errors are defined for illegal values for the new headers, and if an object is under retention any overwrites will fail.

Objects in buckets with retention policy that are no longer under retention (retention period has expired and the object doesn't have any legal holds), when overwritten, will again come under retention. The new retention period can be provided as part of the object overwrite request or the default retention time of the bucket will be given to the object.

A `Content-MD5` header is required.

These headers apply to POST object and multipart upload requests as well. If uploading an object in multiple parts, each part requires a `Content-MD5` header.

Value	Type	Description
`Retention-Period`	Non-negative integer (seconds)	Retention period to store on the object in seconds. The object can be neither overwritten nor deleted until the amount of time that is specified in the retention period has elapsed. If this field and `Retention-Expiration-Date` are specified a `400` error is returned. If neither is specified the bucket's `DefaultRetention` period will be used. Zero (`0`) is a legal value assuming the bucket's minimum retention period is also `0`.
`Retention-expiration-date`	Date (ISO 8601 Format)	Date on which it is legal to delete or modify the object. You can only specify this or the Retention-Period header. If both are specified a `400` error will be returned. If neither is specified the bucket's `DefaultRetention` period will be used. Supported ISO 8601 format is `[YYYY]-[MM]-[DD]T[hh]:[mm]:[ss]Z` or `[YYYY][MM][DD]T[hh][mm][ss]Z` (for example, `2020-11-28T03:10:01Z` or `20201128T031001Z` are both valid).
`Retention-legal-hold-id`	String	A single legal hold to apply to the object. A legal hold is a Y character long string. The object cannot be overwritten or deleted until all legal holds associated with the object are removed.

Python

```
def put_object_add_legal_hold(bucket_name, object_name, file_text, legal_hold_id):
    print("Add legal hold {0} to {1} in bucket {2} with a putObject operation.\n".format(legal_hold_id, object_name,
bucket_name))

    cos.put_object(
        Bucket=bucket_name,
        Key=object_name,
        Body=file_text,
        RetentionLegalHoldId=legal_hold_id)

    print("Legal hold {0} added to object {1} in bucket {2}\n".format(legal_hold_id, object_name, bucket_name))

def copy_protected_object(source_bucket_name, source_object_name, destination_bucket_name, new_object_name):
    print("Copy protected object {0} from bucket {1} to {2}/{3}.\n".format(source_object_name, source_bucket_name,
destination_bucket_name, new_object_name))

    copy_source = {
        "Bucket": source_bucket_name,
        "Key": source_object_name
    }

    cos.copy_object(
        Bucket=destination_bucket_name,
        Key=new_object_name,
        CopySource=copy_source,
        RetentionDirective="Copy"
    )

    print("Protected object copied from {0}/{1} to {2}/{3}\n".format(source_bucket_name, source_object_name,
destination_bucket_name, new_object_name));

def complete_multipart_upload_with_retention(bucket_name, object_name, upload_id, retention_period):
    print("Completing multi-part upload for object {0} in bucket {1}\n".format(object_name, bucket_name))

    cos.complete_multipart_upload(
        Bucket=bucket_name,
        Key=object_name,
        MultipartUpload={
            "Parts":[{
                "ETag": part["ETag"],
                "PartNumber": 1
            }]
        },
        UploadId=upload_id,
        RetentionPeriod=retention_period
    )

    print("Multi-part upload completed for object {0} in bucket {1}\n".format(object_name, bucket_name))

def upload_file_with_retention(bucket_name, object_name, path_to_file, retention_period):
    print("Uploading file {0} to object {1} in bucket {2}\n".format(path_to_file, object_name, bucket_name))

    args = {
        "RetentionPeriod": retention_period
    }

    cos.upload_file(
        Filename=path_to_file,
        Bucket=bucket_name,
        Key=object_name,
        ExtraArgs=args
    )

    print("File upload complete to object {0} in bucket {1}\n".format(object_name, bucket_name))
```

Node

```
function putObjectAddLegalHold(bucketName, objectName, legalHoldId) {
    console.log(`Add legal hold ${legalHoldId} to ${objectName} in bucket ${bucketName} with a putObject operation.`);
    return cos.putObject({
        Bucket: bucketName,
        Key: objectName,
        Body: 'body',
        RetentionLegalHoldId: legalHoldId
    }).promise()
    .then((data) => {
        console.log(`Legal hold ${legalHoldId} added to object ${objectName} in bucket ${bucketName}`);
    })
    .catch((e) => {
        console.log(`ERROR: ${e.code} - ${e.message}\n`);
    });
}

function copyProtectedObject(sourceBucketName, sourceObjectName, destinationBucketName, newObjectName, ) {
    console.log(`Copy protected object ${sourceObjectName} from bucket ${sourceBucketName} to ${destinationBucketName}/${newObjectName}.`);
    return cos.copyObject({
        Bucket: destinationBucketName,
        Key: newObjectName,
        CopySource: sourceBucketName + '/' + sourceObjectName,
        RetentionDirective: 'Copy'
    }).promise()
    .then((data) => {
        console.log(`Protected object copied from ${sourceBucketName}/${sourceObjectName} to ${destinationBucketName}/${newObjectName}`);
    })
    .catch((e) => {
        console.log(`ERROR: ${e.code} - ${e.message}\n`);
    });
}
```

Java

```
public static void putObjectAddLegalHold(String bucketName, String objectName, String fileText, String legalHoldId) {
    System.out.printf("Add legal hold %s to %s in bucket %s with a putObject operation.\n", legalHoldId, objectName, bucketName);

    InputStream newStream = new ByteArrayInputStream(fileText.getBytes(StandardCharsets.UTF_8));

    ObjectMetadata metadata = new ObjectMetadata();
    metadata.setContentLength(fileText.length());

    PutObjectRequest req = new PutObjectRequest(
        bucketName,
        objectName,
        newStream,
        metadata
    );
    req.setRetentionLegalHoldId(legalHoldId);

    cos.putObject(req);

    System.out.printf("Legal hold %s added to object %s in bucket %s\n", legalHoldId, objectName, bucketName);
}

public static void copyProtectedObject(String sourceBucketName, String sourceObjectName, String destinationBucketName, String
newObjectName) {
    System.out.printf("Copy protected object %s from bucket %s to %s/%s.\n", sourceObjectName, sourceBucketName,
destinationBucketName, newObjectName);

    CopyObjectRequest req = new CopyObjectRequest(
        sourceBucketName,
        sourceObjectName,
        destinationBucketName,
        newObjectName
    );
    req.setRetentionDirective(RetentionDirective.COPY);


    cos.copyObject(req);

    System.out.printf("Protected object copied from %s/%s to %s/%s\n", sourceObjectName, sourceBucketName, destinationBucketName,
newObjectName);
}
```

Add or remove a legal hold to or from an object

This implementation of the `POST` operation uses the `legalHold` query parameter and `add` and `remove` query parameters to add or remove a single legal hold from a protected object in a protected bucket.

The object can support 100 legal holds:

- A legal hold identifier is a string of maximum length 64 characters and a minimum length of one character. Valid characters are letters, numbers, `!`, `_`, `.`, `*`, `(`, `)`, `-`, and `'`.
- If the addition of the given legal hold exceeds 100 total legal holds on the object, the new legal hold will not be added, a `400` error is returned.
- If an identifier is too long, it will not be added to the object and a `400` error is returned.
- If an identifier contains invalid characters, it will not be added to the object and a `400` error is returned.
- If an identifier is already in use on an object, the existing legal hold is not modified and the response indicates that the identifier was already in use with a `409` error.
- If an object does not have retention period metadata, a `400` error is returned and adding or removing a legal hold is not allowed.

The presence of a retention period header is required, otherwise a `400` error is returned.

Syntax

```
$ POST https://{endpoint}/{bucket-name}/{object-name}?legalHold # path style
POST https://{bucket-name}.{endpoint}/{object-name}?legalHold= # virtual host style
```

Example request

```
POST /BucketName/ObjectName?legalHold&add=legalHoldID HTTP/1.1
Host: myBucket.mydsNet.corp.com
Date: Fri, 8 Dec 2018 17:50:00 GMT
Authorization: authorization string
Content-Type: text/plain
```

Example response

```
HTTP/1.1 200 OK
Date: Fri, 8 Dec 2018 17:51:00 GMT
Connection: close
```

Python

```
def add_legal_hold_to_object(bucket_name, object_name, legal_hold_id):
    print("Adding legal hold {0} to object {1} in bucket {2}\n".format(legal_hold_id, object_name, bucket_name))

    cos.add_legal_hold(
        Bucket=bucket_name,
        Key=object_name,
        RetentionLegalHoldId=legal_hold_id
    )

    print("Legal hold {0} added to object {1} in bucket {2}!\n".format(legal_hold_id, object_name, bucket_name))

def delete_legal_hold_from_object(bucket_name, object_name, legal_hold_id):
    print("Deleting legal hold {0} from object {1} in bucket {2}\n".format(legal_hold_id, object_name, bucket_name))

    cos.delete_legal_hold(
        Bucket=bucket_name,
        Key=object_name,
        RetentionLegalHoldId=legal_hold_id
    )

    print("Legal hold {0} deleted from object {1} in bucket {2}!\n".format(legal_hold_id, object_name, bucket_name))
```

Node

```
function addLegalHoldToObject(bucketName, objectName, legalHoldId) {
    console.log(`Adding legal hold ${legalHoldId} to object ${objectName} in bucket ${bucketName}`);
    return cos.client.addLegalHold({
        Bucket: bucketName,
        Key: objectId,
        RetentionLegalHoldId: legalHoldId
    }).promise()
    .then(() => {
        console.log(`Legal hold ${legalHoldId} added to object ${objectName} in bucket ${bucketName}!`);
    })
    .catch((e) => {
        console.log(`ERROR: ${e.code} - ${e.message}\n`);
    });
}

function deleteLegalHoldFromObject(bucketName, objectName, legalHoldId) {
    console.log(`Deleting legal hold ${legalHoldId} from object ${objectName} in bucket ${bucketName}`);
    return cos.client.deleteLegalHold({
        Bucket: bucketName,
        Key: objectId,
        RetentionLegalHoldId: legalHoldId
    }).promise()
    .then(() => {
        console.log(`Legal hold ${legalHoldId} deleted from object ${objectName} in bucket ${bucketName}!`);
    })
    .catch((e) => {
        console.log(`ERROR: ${e.code} - ${e.message}\n`);
    });
}
```

Java

```
public static void addLegalHoldToObject(String bucketName, String objectName, String legalHoldId) {
    System.out.printf("Adding legal hold %s to object %s in bucket %s\n", legalHoldId, objectName, bucketName);

    cos.addLegalHold(
        bucketName,
        objectName,
        legalHoldId
    );

    System.out.printf("Legal hold %s added to object %s in bucket %s!\n", legalHoldId, objectName, bucketName);
}

public static void deleteLegalHoldFromObject(String bucketName, String objectName, String legalHoldId) {
    System.out.printf("Deleting legal hold %s from object %s in bucket %s\n", legalHoldId, objectName, bucketName);

    cos.deleteLegalHold(
        bucketName,
        objectName,
        legalHoldId
    );

    System.out.printf("Legal hold %s deleted from object %s in bucket %s!\n", legalHoldId, objectName, bucketName);
}
```

Extend the retention period of an object

This implementation of the `POST` operation uses the `extendRetention` query parameter to extend the retention period of a protected object in a protected bucket.

The retention period of an object can only be extended. It cannot be decreased from the currently configured value.

The retention expansion value is set in one of three ways:

- additional time from the current value (`Additional-Retention-Period` or similar method)
- new extension period in seconds (`Extend-Retention-From-Current-Time` or similar method)
- new retention expiry date of the object (`New-Retention-Expiration-Date` or similar method)

The current retention period that is stored in the object metadata is either increased by the given extra time or replaced with the new value, depending on the parameter that is set in the `extendRetention` request. In all cases, the extend retention parameter is checked against the current retention period and the extended parameter is only accepted if the updated retention period is greater than the current retention period.

Supported ISO 8601 format for `New-Retention-Expiration-Date` is `[YYYY]-[MM]-[DD]T[hh]:[mm]:[ss]Z` or `[YYYY][MM][DD]T[hh][mm][ss]Z` (for example, `2020-11-28T03:10:01Z` or `20201128T031001Z` are both valid).

Objects in protected buckets that are no longer under retention (retention period has expired and the object does not have any legal holds), when overwritten, will again come under retention. The new retention period can be provided as part of the object overwrite request or the default retention time of the bucket will be given to the object.

Syntax

```
POST https://{endpoint}/{bucket-name}/{object-name}?extendRetention= # path style
POST https://{bucket-name}.{endpoint}/{object-name}?extendRetention= # virtual host style
```

Example request

```
POST /BucketName/ObjectName?extendRetention HTTP/1.1
Host: myBucket.mydsNet.corp.com
Date: Fri, 8 Dec 2018 17:50:00GMT
Authorization: authorization string
Content-Type: text/plain
Additional-Retention-Period: 31470552
```

Example response

```
HTTP/1.1 200 OK
Date: Fri, 8 Dec 2018 17:50:00GMT
Connection: close
```

Python

```python
def extend_retention_period_on_object(bucket_name, object_name, additional_seconds):
    print("Extend the retention period on {0} in bucket {1} by {2} seconds.\n".format(object_name, bucket_name, 
additional_seconds))

    cos.extend_object_retention(
        Bucket=bucket_ame,
        Key=object_name,
        AdditionalRetentionPeriod=additional_seconds
    )

    print("New retention period on {0} is {1}\n".format(object_name, additional_seconds))
```

Node

```javascript
function extendRetentionPeriodOnObject(bucketName, objectName, additionalSeconds) {
    console.log(`Extend the retention period on ${objectName} in bucket ${bucketName} by ${additionalSeconds} seconds.`);
    return cos.extendObjectRetention({
        Bucket: bucketName,
        Key: objectName,
        AdditionalRetentionPeriod: additionalSeconds
    }).promise()
    .then((data) => {
        console.log(`New retention period on ${objectName} is ${data.RetentionPeriod}`);
    })
    .catch((e) => {
        console.log(`ERROR: ${e.code} - ${e.message}\n`);
    });
}
```

Java

```java
public static void extendRetentionPeriodOnObject(String bucketName, String objectName, Long additionalSeconds) {
    System.out.printf("Extend the retention period on %s in bucket %s by %s seconds.\n", objectName, bucketName, 
additionalSeconds);

    ExtendObjectRetentionRequest req = new ExtendObjectRetentionRequest(
        bucketName,
        objectName)
        .withAdditionalRetentionPeriod(additionalSeconds);

    cos.extendObjectRetention(req);

    System.out.printf("New retention period on %s is %s\n", objectName, additionalSeconds);
}
```

List legal holds on an object

This implementation of the `GET` operation uses the `legalHold` query parameter to return the list of legal holds on an object and related retention state in an XML response body.

This operation returns:

- Object creation date
- Object retention period in seconds
- Calculated retention expiration date based on the period and creation date
- List of legal holds
- Legal hold identifier
- Timestamp when legal hold was applied

If there are no legal holds on the object, an empty `LegalHoldSet` is returned. If there is no retention period that is specified on the object, a `404` error is returned.

Syntax

```
$ GET https://{endpoint}/{bucket-name}/{object-name}?legalHold= # path style
GET https://{bucket-name}.{endpoint}/{object-name}?legalHold= # virtual host style
```

Example request

```
GET /BucketName/ObjectName?legalHold HTTP/1.1
Host: myBucket.mydsNet.corp.com
Date: Fri, 8 Dec 2018 17:50:00 GMT
Authorization: {authorization-string}
Content-Type: text/plain
```

Example response

```
HTTP/1.1 200 OK
Date: Fri, 8 Dec 2018 17:51:00 GMT
Connection: close
<?xml version="1.0" encoding="UTF-8"?>
<RetentionState>
  <CreateTime>Fri, 8 Sep 2018 21:33:08 GMT</CreateTime>
  <RetentionPeriod>220752000</RetentionPeriod>
  <RetentionPeriodExpirationDate>Fri, 1 Sep 2023 21:33:08
GMT</RetentionPeriodExpirationDate>
  <LegalHoldSet>
    <LegalHold>
      <ID>SomeLegalHoldID</ID>
      <Date>Fri, 8 Sep 2018 23:13:18 GMT</Date>
    </LegalHold>
    <LegalHold>
    ...
    </LegalHold>
  </LegalHoldSet>
</RetentionState>
```

Python

```
def list_legal_holds_on_object(bucket_name, object_name):
    print("List all legal holds on object {0} in bucket {1}\n".format(object_name, bucket_name));

    response = cos.list_legal_holds(
        Bucket=bucket_name,
        Key=object_name
    )

    print("Legal holds on bucket {0}: {1}\n".format(bucket_name, response))
```

Node

```
function listLegalHoldsOnObject(bucketName, objectName) {
    console.log(`List all legal holds on object ${objectName} in bucket ${bucketName}`);
    return cos.listLegalHolds({
        Bucket: bucketName,
        Key: objectId
    }).promise()
    .then((data) => {
        console.log(`Legal holds on bucket ${bucketName}: ${data}`);
    })
    .catch((e) => {
        console.log(`ERROR: ${e.code} - ${e.message}\n`);
    });
}
```

Java

```
public static void listLegalHoldsOnObject(String bucketName, String objectName) {
    System.out.printf("List all legal holds on object %s in bucket %s\n", objectName, bucketName);

    ListLegalHoldsResult result = cos.listLegalHolds(
        bucketName,
        objectName
    );

    System.out.printf("Legal holds on bucket %s: \n", bucketName);

    List<LegalHold> holds = result.getLegalHolds();
    for (LegalHold hold : holds) {
        System.out.printf("Legal Hold: %s", hold);
    }
}
```

Tracking Object Lock events

Object Lock preserves electronic records and maintains data integrity by ensuring that individual object versions are stored in a WORM (Write-Once-Read-Many), non-erasable and non-rewritable manner. This policy is enforced until a specified date or the removal of any legal holds.

Why use Object Lock?

Object Lock helps customers govern data preservation and retention requirements by enforcing data immutability for their backup, disaster recovery, and cyber resiliency workloads.

Object Lock ensures that **data cannot be deleted by anyone** and there is **no way to suspend retention on an object** . Read the documentation carefully before locking objects with a retention period.

When using Object Lock, it is your responsibility to ensure compliance with any regulations that you (your organization) may be subject to when it comes to preservation and storage of data for long term retention.

Important: When using Object Lock, you are responsible for ensuring that your IBM Cloud Account is kept in good standing per IBM Cloud policies and guidelines for as long as the data is subject to a retention period. Refer to IBM Cloud Service terms for more information.

Terminology

There are two ways use Object Lock to protect data: *retention periods* and *legal holds*.

- A **retention period** defines a timeframe during which an object is unable to be modified or deleted.
- A **legal hold** also prevents an object from being altered, but only remains in place until it is explicitly lifted.

It is possible to make use of any combination of these parameters - an object version can have one, both, or neither.

Retain Until Date (Retention Period)

If you need to protect an object version for a fixed amount of time, you need to specify a *Retain Until Date* which determines the period in which it cannot be altered. The object version can be be deleted after this date is passed (assuming there are no legal holds on the object version).

The retention period for new objects can be inherited from the default value set on the bucket, or it can be explicitly defined when writing the object by specifying a *Retain Until Date*.

When you use bucket default settings, you don’t specify a Retain Until Date. Instead, you specify a duration, in either days or years, for which every object version placed in the bucket should be protected. When you place an object in the bucket, a Retain Until Date is calculated for the object version by adding the specified duration to the time of the object write.

Note: If your request to place an object version in a bucket contains an explicit retention mode and Retain Until Date, those settings override any bucket default settings for that object version.

Like all other Object Lock settings, the Retain Until Date applies to individual object versions. Different versions of a single object can have different retention modes and periods.

Imagine an object that is 60 days into a 90-day retention period, and you overwrite that object with the same name and a two year retention period. The operation will succeed and a new version of the object with a two year retention period is created. Meanwhile, after 30 more days the original version is eligible for deletion.

Extending a retention period

To extend the retention period of an object, simply send a request to set a new, longer, retention period. The old value will be overwritten with the new, assuming the requester has the `cloud-object-storage.object.put_object_lock_retention` and `cloud-object-storage.object.put_object_lock_retention_version` actions.

Legal Hold

A *legal hold* is like a retention period in that it prevents an object version from being overwritten or deleted. However, legal holds are more flexible and don't have a defined temporal component. Instead they simply remain in effect until removed. Legal holds can be freely placed and removed by any user who has the `cloud-object-storage.object.put_object_lock_legal_hold` and `cloud-object-storage.object.put_object_lock_legal_hold_version` actions.

Legal holds have the additional benefit of acting as method for applying indefinite retention on an object.

Legal holds and retention periods operate independently. Legal holds have no impact on retention periods, and vice-versa.

Imagine an object with both a legal hold and a retention period. When the retention period ends, the object version remains protected until the legal hold is removed. If you remove a legal hold while an object version is subject to a retention period it remains protected until the retention period is complete.

⚠ **Important:** Objects locked and stored with a retention period cannot be deleted until retention period expires and any associated legal hold is removed.

Note: Locking objects with a Governance Mode is not supported.

Getting started with Object Lock

To get started, there are some some prerequisites:

- You'll need the `Writer` or `Manager` platform role on a bucket, or a custom role with the appropriate actions (such as `cloud-object-storage.bucket.put_object_lock_configuration`) assigned.
- Object Versioning must be enabled
- You will need to use Standard pricing plan, see pricing for details.
- You will need to pick a region where Object Lock is supported, refer to Integrated Services for details.
- A maximum default retention period of 100 years (or 36500 days) is supported.
- When using the console, it is also possible to set a Retain Until Date in months, in addition to days or years.

Tip: The retention period for an object **cannot be decreased**. If you are using default retention for validation testing please use a lower duration (such as 1 day) as the default retention, increasing it to your desired setting as needed.

Creating and setting up your new bucket for use with Object Lock

1. Navigate to your desired Object Storage instance and use **Create Bucket** with *Customize your bucket option*
2. Enter the required bucket configuration details per your use case requirements
3. Navigate to the *Object Versioning* section and set it to **Enabled**
4. Look for **Immutability**, and under Object Lock click **Add**
5. Set Object Lock to **Enabled**
6. Optionally, set a default retention period.
7. Click on Save
8. Proceed with rest of the configuration settings and click **Create bucket**

Enabling Object Lock on an existing bucket:

A bucket can be set for Object Lock use as follows:

1. Navigate to your bucket **Configuration** section
2. Click on **Object Versioning**
3. At the *Object Versioning* section click on **Edit**, set the configuration option to **Enabled** and **Save**
4. Navigate to *Object Lock* section, click on **Add**
5. Set *Object Lock* to **Enabled**

6. Optionally, set a default retention period.
7. Click on **Save**

Adding a Retain Until Date or Legal Hold to an object

1. Navigate to the bucket with the target object
2. Toggle **Display Versions**
3. Go to the details of the target version
4. Add a retention period and/or toggle on a legal hold.

Using Object Lock for business continuity and disaster recovery

Object Lock can be used to provide continuity of service in the event of a ransomware attack, as protected data is unable to be modified or destroyed.

Consistency and data integrity

While IBM Cloud Object Storage provides strong consistency for all data IO operations, bucket configuration is eventually consistent. After enabling, modifying, or deleting a default retention period on a bucket it may take a few moments for the configuration to propagate across the system. Operations on objects, such as adding a legal hold, are immediately consistent.

Usage and accounting

Locked objects (and their versions) contribute usage just like any other data and you will be responsible for the usage costs for as long as object remains locked with a retention period.

Interactions

Object Lock can be used in combination with several object storage features as per your use case requirements.

Versioning

Enabling versioning is a prerequisite for enabling Object Lock. If a bucket is created using the `x-amz-bucket-object-lock-enabled` header, versioning will automatically be enabled.

Deleting a versioned object creates a *delete marker*. The object may appear to be deleted, but if the object is protected it is impossible to delete the protected version. Delete markers themselves are not protected.

Replication

Object Lock **cannot be used on the source bucket** for replication, only on the destination. Objects will be assigned the default retention period.

Key Management Systems

Protected objects will be encrypted using the root key of the bucket. When Object Lock is enabled on a bucket, the root key hosted by Key Protect or Hyper Protect Crypto Services is protected from deletion as long as an associated bucket has Object Lock enabled. This prevents crypto shredding of protected objects.

Lifecycle configurations

It is possible to enable lifecycle policies that archive locked objects, but naturally not those that expire objects under retention or legal hold (unprotected objects in the bucket can still be expired).

Immutable Object Storage

Object Lock is an alternative to the retention policies available when using Immutable Object Storage. As Object Lock requires versioning to be enabled, and Immutable Object Storage is not compatible with versioning, it is impossible to have both WORM solutions enabled on the same bucket. It is possible to have a mix of buckets in a Service Instance, each using either Immutable Object Storage or Object Lock.

Object Tagging

There are no restrictions on adding or modifying tags on a protected object.

Other interactions

There should be no adverse interactions when using Object Lock with other Object Storage features, such as setting CORS policies, setting IP firewalls or condition based restrictions, bucket quotas, or Code Engine.

IAM actions

There are new IAM actions associated with Object Lock.

IAM Action	Role
`cloud-object-storage.bucket.get_object_lock_configuration`	Manager, Writer, Reader
`cloud-object-storage.bucket.put_object_lock_configuration`	Manager, Writer
`cloud-object-storage.object.get_object_lock_retention`	Manager, Writer, Reader
`cloud-object-storage.object.put_object_lock_retention`	Manager, Writer
`cloud-object-storage.object.get_object_lock_retention_version`	Manager, Writer, Reader
`cloud-object-storage.object.put_object_lock_retention_version`	Manager, Writer
`cloud-object-storage.object.get_object_lock_legal_hold`	Manager, Writer, Reader
`cloud-object-storage.object.put_object_lock_legal_hold`	Manager, Writer
`cloud-object-storage.object.get_object_lock_legal_hold_version`	Manager, Writer, Reader
`cloud-object-storage.object.put_object_lock_legal_hold_version`	Manager, Writer

IAM Actions

Be advised that users with the Writer role are capable of making objects un-deletable for many years (possibly thousand of years). Be careful, and consider crafting custom roles that do not allow most users to set a Retain Until Date.

Activity Tracker events

Object Lock generates additional events.

- `cloud-object-storage.bucket-object-lock.create`
- `cloud-object-storage.bucket-object-lock.read`
- `cloud-object-storage.object-object-lock-legal-hold.create`
- `cloud-object-storage.object-object-lock-legal-hold.read`
- `cloud-object-storage.object-object-lock-retention.create`
- `cloud-object-storage.object-object-lock-retention.read`

For `cloud-object-storage.bucket-object-lock.create` events, the following fields provide extra information:

Field	Description
`requestData.object_lock_configuration.enabled`	Indicates that Object Lock is enabled on the bucket
`requestData.object_lock_configuration.defaultRetention.mode`	Indicates `COMPLIANCE` mode is active - `GOVERNANCE` mode is not yet supported.
`object_lock_configuration.defaultRetention.years`	The default retention period in years.
`object_lock_configuration.defaultRetention.days`	The default retention period in days.

Note: Only `object_lock_configuration.defaultRetention.years` or `object_lock_configuration.defaultRetention.days` will be present, but not both at the same time.

For operations on protected objects, the following fields may be present:

Field	Description
`requestData.object_lock_protection.legal_hold`	Indicates that a legal hold is in force on the object version.
`requestData.object_lock_protection.retention.mode`	Indicates `COMPLIANCE` mode is active on the object version - `GOVERNANCE` mode is not yet supported.
`requestData.object_lock_protection.retention.retain_until_date`	Indicates the date that object version is eligible for deletion. After this date the object is no longer delete protected based on a retention date.

REST API examples

The following examples are shown using cURL for ease of use. Environment variables are used to represent user specific elements such as `$BUCKET`, `$TOKEN`, and `$REGION`. Note that `$REGION` would also include any network type specifications, so sending a request to a bucket in `us-south` using the private network would require setting the variable to `private.us-south`.

Enable object lock on a bucket

The Object Lock configuration is provided as XML in the body of the request. New requests will overwrite any existing replication rules that are present on the bucket.

An Object Lock configuration must include one rule.

Header	Type	Description
`Content-MD5`	String	**Required**: The base64 encoded 128-bit MD5 hash of the payload, which is used as an integrity check to ensure that the payload wasn't altered in transit.

The body of the request must contain an XML block with the following schema:

Element	Type	Children	Ancestor	Constraint
`ObjectLockConfiguration`	Container	`ObjectLockEnabled`, `Rule`	None	**Required** Limit 1.
`ObjectLockEnabled`	String	None	`ObjectLockConfiguration`	**Required** The only valid value is `Enabled` (case-sensitive).
`Rule`	Container	`DefaultRetention`	`ObjectLockConfiguration`	Limit 1
`DefaultRetention`	Container	`Days`, `Mode`, `Years`	`Rule`	Limit 1.
`Days`	Integer	None	`DefaultRetention`	The number of days that you want to specify for the default retention period. Cannot be combined with `Years`.
`Mode`	String	None	`DefaultRetention`	Only `COMPLIANCE` is supported at this time (case-sensitive).
`Years`	Integer	None	`DefaultRetention`	The number of years that you want to specify for the default retention period. Cannot be combined with `Days`.

This example will retain any new objects for at least 30 days.

```
$ curl -X "PUT" "https://$BUCKET.s3.$REGION.cloud-object-storage.appdomain.cloud/?object-lock" \
     -H 'Authorization: bearer $TOKEN' \
     -H 'Content-MD5: exuBoz2kFBykNwqu64JZuA==' \
     -H 'Content-Type: text/plain; charset=utf-8' \
     -d $'<ObjectLockConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
           <ObjectLockEnabled>Enabled</ObjectLockEnabled>
           <Rule>
```

```
                <DefaultRetention>
                  <Days>30</Days>
                  <Mode>COMPLIANCE</Mode>
                </DefaultRetention>
            </Rule>
          </ObjectLockConfiguration>'
```

A successful request returns a `200` response.

View Object Lock configuration for a bucket

```
$ curl -X "GET" "https://$BUCKET.s3.$REGION.cloud-object-storage.appdomain.cloud/?object-lock" \
     -H 'Authorization: bearer $TOKEN'
```

This returns an XML response body with the appropriate schema:

```
$ <ObjectLockConfiguration  xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <ObjectLockEnabled>string</ObjectLockEnabled>
  <Rule>
      <DefaultRetention>
        <Days>30</Days>
        <Mode>COMPLIANCE</Mode>
      </DefaultRetention>
  </Rule>
</ObjectLockConfiguration>
```

Add or extend a retention period for an object

The Object Lock configuration is provided as XML in the body of the request. New requests will overwrite any existing replication rules that are present on the object, provided the `RetainUntilDate` is farther in the future than the current value.

Header	Type	Description
`Content-MD5`	String	**Required**: The base64 encoded 128-bit MD5 hash of the payload, which is used as an integrity check to ensure that the payload wasn't altered in transit.

Optionally, you can specify the version for which to apply the `RetainUntilDate`.

Optional query parameters

Parameter	Required?	Type	Description
`versionID`	Optional	string	Version ID.

The body of the request must contain an XML block with the following schema:

Element	Type	Children	Ancestor	Constraint
`Retention`	Container	`Mode`, `RetainUntilDate`	None	**Required** Limit 1.
`Mode`	String	None	`Retention`	**Required** Only `COMPLIANCE` is supported at this time (case-sensitive).
`RetainUntilDate`	String	None	`Retention`	**Required** The date after which an object is eligible for deletion in ISO8601 Date-Time Format.

This example will retain any new objects for at least until March 12, 2023.

```
$ curl -X "PUT" "https://$BUCKET.s3.$REGION.cloud-object-storage.appdomain.cloud/?retention" \
     -H 'Authorization: Bearer $TOKEN' \
     -H 'Content-MD5: fT0hYstki6zUvEh7abhcTA==' \
     -H 'Content-Type: text/plain; charset=utf-8' \
     -d $'<Retention>
            <Mode>COMPLIANCE</Mode>
```

```
          <RetainUntilDate>2023-03-12T23:01:00.000Z</RetainUntilDate>
        </Retention>'
```

A successful request returns a `200` response.

If the `RetainUntilDate` values is not beyond any existing value, the operation will fail with a `403 Access Denied`.

Add or remove a legal hold for an object

The Object Lock configuration is provided as XML in the body of the request. New requests will overwrite any existing replication rules that are present on the object, provided the `RetainUntilDate` is farther in the future than the current value.

Header	Type	Description
`Content-MD5`	String	**Required**: The base64 encoded 128-bit MD5 hash of the payload, which is used as an integrity check to ensure that the payload wasn't altered in transit.

The body of the request must contain an XML block with the following schema:

Element	Type	Children	Ancestor	Constraint
`legal-hold`	Container	`Status`	None	Limit 1.
`Status`	String	None	`legal-hold`	Supported values are `ON` or `OFF` (case-sensitive)

This example will retain any new objects for at least until March 12, 2023.

```
$ curl -X "PUT" "https://$BUCKET.s3.$REGION.cloud-object-storage.appdomain.cloud/?legal-hold&versionId=$VERSION_ID" \
     -H 'Authorization: Bearer $TOKEN' \
     -H 'Content-MD5: FMh6GxizXUBRaiDuB0vtgQ==' \
     -H 'Content-Type: text/plain; charset=utf-8' \
     -d $'<legal-hold>
          <Status>ON</Status>
        </legal-hold>'
```

A successful request returns a `200` response.

SDK examples

The following examples make use of the IBM COS SDKs for Python, Node.js, and Go, as well as a Terraform script, although the implementation of object versioning should be fully compatible with any S3-compatible library or tool that allows for the setting of custom endpoints. Using third-party tools requires HMAC credentials to calculate AWS V4 signatures.

Python

Enabling Object Lock using the IBM COS SDK for Python can be done using the low-level client syntax.

Using a client:

```
import ibm_boto3
from ibm_botocore.client import Config
from ibm_botocore.exceptions import ClientError
from datetime import datetime, timedelta
import time

# Create new bucket with Object Lock enabled.
def create_bucket_with_objectlock(bucket_name):
        cos_cli.create_bucket(
            Bucket=bucket_name,
            ObjectLockEnabledForBucket=True,
        )
        print("Bucket: {0} created with objectlock enabled".format(bucket_name))

def objectlock_configuration_on_bucket(bucket_name):

    # Putting default retenion on the COS bucket.
    default_retention_rule = {'DefaultRetention': {'Mode': 'COMPLIANCE', 'Years': 1}}
    object_lock_config = {'ObjectLockEnabled': 'Enabled', 'Rule': default_retention_rule}
    cos_cli.put_object_lock_configuration(Bucket=bucket_name, ObjectLockConfiguration=object_lock_config)
    # Reading the objectlock configuration set on the bucket.
    response = cos_cli.get_object_lock_configuration(Bucket=bucket_name)
    print("Objectlock Configuration for {0} =>".format(bucket_name))
    print(response.ObjectLockConfiguration)

def upload_object(bucket_name,object_name,object_content):
        cos_cli.put_object(
            Bucket=bucket_name,
            Key=object_name,
            Body=object_content
        )
        print("Object: {0} uploaded!".format(object_name))

def objectlock_retention(bucket_name,object_name):
        # Put objectlock retenion on the  object uploaded to the bucket.
        date = datetime.now()+timedelta(seconds=5)
        retention_rule = {'Mode': 'COMPLIANCE', 'RetainUntilDate': date}
        cos_cli.put_object_retention(Bucket=bucket_name, Key=object_name, Retention=retention_rule)

        # Get objectlock retention of the above object.
        response = cos_cli.get_object_retention(Bucket=bucket_name, Key=object_name)
        print("Objectlock Retention for {0}=>".format(object_name))
        print(response.Retention)

def objectlock_legal-hold(bucket_name,object_name):
        # Setting the objectlock legal-hold status to ON.
        cos_cli.put_object_legal_hold(Bucket=bucket_name, Key=object_name, legal-hold={'Status': 'ON'})
        # Get objectlock retention of the above object.
        response = cos_cli.get_object_legal_hold(Bucket=bucket_name, Key=object_name)
        print("Objectlock legal-hold for {0}=>".format(object_name))
        print(response.legal-hold)

COS_ENDPOINT = "" #Current list avaiable at https://control.cloud-object-storage.cloud.ibm.com/v2/endpoints -> Ex:https://s3.us-
south.cloud-object-storage.appdomain.cloud
COS_API_KEY_ID = "" #API Key of the cos instance created Ex: W00YixxxxxxxxxxMB-odB-2ySfTrFBIQQWanc--P3byk
COS_RESOURCE_INSTANCE_CRN = "" #API key of cos instance example: xxxd12V2QHXbjaM99G9tWyYDgF_0gYdlQ8aWALIQxXx4

# Create client connection
cos_cli = ibm_boto3.client("s3",
    ibm_api_key_id=COS_API_KEY_ID,
    config=Config(signature_version="oauth"),
    endpoint_url=COS_ENDPOINT,
    ibm_service_instance_id=COS_RESOURCE_INSTANCE_CRN,
    ibm_auth_endpoint="https://iam.test.cloud.ibm.com/identity/token"
)
new_bucket_name = "create-example-python12345" # bucket name should be unique gloablly, or else it will throw an error.
new_text_file_name = "cos_object.txt"
new_text_file_contents = "This is a test file from Python code sample!!!"

# *** Main Program ***
def main():
       create_bucket_with_objectlock(new_bucket_name) # Create a new cos bucket with object lock enabled.
       objectlock_configuration_on_bucket(new_bucket_name) # Put objectlock configuration(i.e. default retention) on COS bucket
and get the configuration.
       upload_object(new_bucket_name,new_text_file_name,new_text_file_contents) # Upload an object to cos bucket.
       objectlock_retention(new_bucket_name,new_text_file_name) # Put objectlock retention(i.e. retain until date) on the object
and get the configured retention.
       objectlock_legal-hold(new_bucket_name,new_text_file_name)  # Put objectlock legal-hold on the object and get the legal-
hold status.

if __name__ == "__main__":
    main()
```

Node.js

Enabling versioning using the IBM COS SDK for Node.js:

```
'use strict';

// Required libraries
const ibm = require('ibm-cos-sdk');
const fs = require('fs');
const crypto = require('crypto');

function logError(e) {
    console.log(`ERROR: ${e.code} - ${e.message}\n`);
}

function logDone() {
    console.log('DONE!\n');
}

const COS_ENDPOINT = "";   //Choose endpoint from https://control.cloud-object-storage.cloud.ibm.com/v2/endpoints. Ex:
https://s3.us-south.cloud-object-storage.appdomain.cloud
const COS_API_KEY_ID = "";  // API key of cos instance example: xxxd12V2QHXbjaM99G9tWyYDgF_0gYdlQ8aWALIQxXx4
const COS_AUTH_ENDPOINT = "";
const COS_RESOURCE_INSTANCE_CRN = ""; // example: crn:v1:bluemix:public:cloud-object-storage:global:a
<CREDENTIAL_ID_AS_GENERATED>:<SERVICE_ID_AS_GENERATED>::

// Client Creation.
var config = {
    endpoint: COS_ENDPOINT,
    apiKeyId: COS_API_KEY_ID,
    ibmAuthEndpoint: COS_AUTH_ENDPOINT,
    serviceInstanceId: COS_RESOURCE_INSTANCE_CRN,
    signatureVersion: 'iam'
};

var cos = new ibm.S3(config);

// Create new bucket with objectlock enabled.
function createBucket(bucketName) {
    console.log(`Creating new bucket: ${bucketName}`);
    return cos.createBucket({
        Bucket: bucketName,
        ObjectLockEnabledForBucket: true,
        CreateBucketConfiguration: {
            LocationConstraint: ''
          },
    }).promise()
    .then((() => {
        console.log(`Bucket: ${bucketName} created!`);
    }))
    .catch((e) => {
        console.error(`ERROR: ${e.code} - ${e.message}\n`);
    });
}

// Create new text file and upload the object to COS bucket.
function createTextFile(bucketName, itemName, fileText) {
    console.log(`Creating new item: ${itemName}`);
    return cos.putObject({
        Bucket: bucketName,
        Key: itemName,
        Body: fileText
    }).promise()
    .then(() => {
        console.log(`Item: ${itemName} created!`);
        logDone();
    })
    .catch(logError);
}

function putObjectLockConfigurationonBucket(bucketName) {
    console.log(`Putting Objectlock Configuration on : ${bucketName}`);
    // Putting objectlock configuration
    var defaultRetention = {Mode: 'COMPLIANCE', Days: 1}
    var objectLockRule = {DefaultRetention : defaultRetention}
    var param = {ObjectLockEnabled: 'Enabled', Rule: objectLockRule}
    return cos.putObjectLockConfiguration({
        Bucket: bucketName,
        ObjectLockConfiguration: param
    }).promise()
    .then(() => {
        console.log(`Object lock Configurtion added!!`);
        logDone();
    })
    .catch(logError);
}

function getObjectLockConfigurationonBucket(bucketName) {
    console.log(`Getting Objectlock Configuration for : ${bucketName}`);
    // Getting objectlock configuration
    return cos.getObjectLockConfiguration({
        Bucket: bucketName,
    }).promise()
    .then((data) => {
        console.log(`objectlock configuration`);
        console.log( JSON.stringify(data.ObjectLockConfiguration, null, "    ") );
        logDone();
    })
    .catch(logError);
}

function putObjectLockRetention(bucketName,keyName) {
    console.log(`Putting Objectlock Retention on : ${keyName}`);
    var inFiveSecond = (new Date(Date.now() + (1000 * 5)))
    var rule = {Mode: 'COMPLIANCE', RetainUntilDate: inFiveSecond}
     // Putting objectlock retention
    return cos.putObjectRetention({
        Bucket: bucketName,
        Key: keyName,
        Retention: rule
    }).promise()
    .then(() => {
        console.log(`Object lock Retention added!!`);
        logDone();
    })
    .catch(logError);
}

function getObjectLockRetention(bucketName,keyName) {
    console.log(`Getting Objectlock Retention for : ${keyName}`);
    // Getting objectlock retention
    return cos.getObjectRetention({
        Bucket: bucketName,
        Key: keyName
    }).promise()
    .then((data) => {
        console.log(`Objectlock retention for : ${keyName} `);
        console.log( JSON.stringify(data.Retention, null, "    ") );
        logDone();
    })
    .catch(logError);
}

function putObjectLocklegal-hold(bucketName,keyName) {
    console.log(`Putting Objectlock legal-hold status ON for  : ${keyName}`);
     // Putting objectlock legal-hold status
    return cos.putObjectlegal-hold({
        Bucket: bucketName,
        Key: keyName,
        legal-hold: {Status: 'ON'}
    }).promise()
    .then(() => {
        console.log(`Object lock legal-hold added!!`);
        logDone();
    })
    .catch(logError);
```

```
}

function getObjectLocklegal-hold(bucketName,keyName) {
    console.log(`Getting Objectlock legal-hold for : ${keyName}`);
    // Getting objectlock legal-hold
    return cos.getObjectlegal-hold({
        Bucket: bucketName,
        Key: keyName
    }).promise()
    .then((data) => {
        console.log(`Objectlock legal-hold for : ${keyName} `);
        console.log( JSON.stringify(data.legal-hold, null, "    ") );
        logDone();
    })
    .catch(logError);
}

// Main app
function main() {
    try {
        var newBucketName = "jscosbucket350";
        var newTextFileName = "js_cos_bucket_file.txt";
        var newTextFileContents = "This is a test file from Node.js code sample!!!";

        createBucket(newBucketName) // Create a new cos bucket with object lock enabled.
        .then(() => putObjectLockConfigurationonBucket(newBucketName)) // Put objectlock configuration(i.e. default retention) on 
COS bucket.
        .then(() => getObjectLockConfigurationonBucket(newBucketName)) // Read objectlock configuration on COS bucket.
        .then(() => createTextFile(newBucketName, newTextFileName, newTextFileContents)) // Upload an object to cos bucket.
        .then(() => putObjectLockRetention(newBucketName, newTextFileName)) // Put objectlock retention(i.e. retain until date) 
on the object.
        .then(() => getObjectLockRetention(newBucketName, newTextFileName)) // Get the configured retention.
        .then(() => putObjectLocklegal-hold(newBucketName,newTextFileName)) // Put objectlock legal-hold on the object.
        .then(() => getObjectLocklegal-hold(newBucketName,newTextFileName)); // Get the legal-hold status.
    }
    catch(ex) {
        logError(ex);
    }
}

main();
```

Go

```
package main

import (
 "bytes"
 "fmt"
 "time"

 "github.com/IBM/ibm-cos-sdk-go/aws"
 "github.com/IBM/ibm-cos-sdk-go/aws/credentials/ibmiam"
 "github.com/IBM/ibm-cos-sdk-go/aws/session"
 "github.com/IBM/ibm-cos-sdk-go/service/s3"
)

const (
 apiKey            = "<apiKey>"
 serviceInstanceID = "<serviceInstanceID>"
 authEndpoint      = "https://iam.cloud.ibm.com/identity/token"
 serviceEndpoint   = "https://<endpoint>.appdomain.cloud"
)

// Create new bucket with objectlock enabled.
func  createBucket(bucketName string, client *s3.S3)  {
 createBucketInput := new(s3.CreateBucketInput)
 createBucketInput.Bucket = aws.String(bucketName)
 createBucketInput.ObjectLockEnabledForBucket = aws.Bool(true)
 _, e := client.CreateBucket(createBucketInput)
 if e != nil {
  fmt.Println(e)
 } else {
  fmt.Println("Bucket Created !!! ")
 }
}

func  uploadObject(bucketName string, client *s3.S3, fileName string, fileContent string)  {
 putInput := &s3.PutObjectInput{
  Bucket: aws.String(bucketName),
  Key:    aws.String(fileName),
  Body:   bytes.NewReader([]byte(fileContent)),
 }

 _, e := client.PutObject(putInput)
 if e != nil {
  fmt.Println(e)
 } else {
  fmt.Println("Object Uploaded!!! ")
 }
}

func  objectLockConfiguration(bucketName string, client *s3.S3)  {
 // Putting default retenion on the COS bucket.
 putObjectLockConfigurationInput := &s3.PutObjectLockConfigurationInput{
  Bucket: aws.String(bucketName),
  ObjectLockConfiguration: &s3.ObjectLockConfiguration{
   ObjectLockEnabled: aws.String(s3.ObjectLockEnabledEnabled),
   Rule: &s3.ObjectLockRule{
    DefaultRetention: &s3.DefaultRetention{
     Mode: aws.String("COMPLIANCE"),
     Days: aws.Int64(1),
    },
   },
  },
 }
 _, e := client.PutObjectLockConfiguration(putObjectLockConfigurationInput)

 // Reading the objectlock configuration set on the bucket.
 getObjectLockConfigurationInput := new(s3.GetObjectLockConfigurationInput)
 getObjectLockConfigurationInput.Bucket = aws.String(bucketName)
 response, e := client.GetObjectLockConfiguration(getObjectLockConfigurationInput)
 if e != nil {
  fmt.Println(e)
 } else {
  fmt.Println("Object Lock Configuration =>", response.ObjectLockConfiguration)
 }
}

func  objectLockRetention(bucketName string, client *s3.S3, keyName string)  {

 // Put objectlock retenion on the  object uploaded to the bucket.
 retention_date := time.Now().Local().Add(time.Second * 5)
 putObjectRetentionInput := &s3.PutObjectRetentionInput{
  Bucket: aws.String(bucketName),
  Key:    aws.String(keyName),
  Retention: &s3.ObjectLockRetention{
   Mode:            aws.String("COMPLIANCE"),
   RetainUntilDate: aws.Time(retention_date),
  },
 }
 _, e := client.PutObjectRetention(putObjectRetentionInput)

 // Get objectlock retention of the above object.
 getObjectRetentionInput := new(s3.GetObjectRetentionInput)
 getObjectRetentionInput.Bucket = aws.String(bucketName)
 getObjectRetentionInput.Key = aws.String(keyName)
 response, e := client.GetObjectRetention(getObjectRetentionInput)
 if e != nil {
  fmt.Println(e)
 } else {
  fmt.Println("Object Lock Retention =>", response.Retention)
 }
}

func  objectLocklegal-hold(bucketName string, client *s3.S3, keyName string)  {

 // Setting the objectlock legal-hold status to ON.
 putObjectlegal-holdInput := &s3.PutObjectlegal-holdInput{
  Bucket: aws.String(bucketName),
  Key:    aws.String(keyName),
  legal-hold: &s3.ObjectLocklegal-hold{
   Status: aws.String("ON"),
  },
 }
 _, e := client.PutObjectlegal-hold(putObjectlegal-holdInput)
 // Get objectlock retention of the above object.
 getObjectlegal-holdInput := new(s3.GetObjectlegal-holdInput)
 getObjectlegal-holdInput.Bucket = aws.String(bucketName)
 getObjectlegal-holdInput.Key = aws.String(keyName)
 response, e := client.GetObjectlegal-hold(getObjectlegal-holdInput)
 if e != nil {
  fmt.Println(e)
 } else {
  fmt.Println("Object Lock legal-hold =>", response.legal-hold)
 }
}

func  main()  {

 bucketName := "gocosbucket353"
 textFileName := "go_cos_bucket_file.txt"
 textFileContents := "This is a test file from Node.js code sample!!!"
 conf := aws.NewConfig().
  WithEndpoint(serviceEndpoint).
  WithCredentials(ibmiam.NewStaticCredentials(aws.NewConfig(),
   authEndpoint, apiKey, serviceInstanceID)).
  WithS3ForcePathStyle(true)

 sess := session.Must(session.NewSession())
 client := s3.New(sess, conf)
 createBucket(bucketName, client)                                 // Create a new cos bucket with object lock enabled.
 objectLockConfiguration(bucketName, client)                      // Put objectlock configuration(i.e. default retention) on COS
bucket and get the configuration.
 uploadObject(bucketName, client, textFileName, textFileContents) // Upload an object to cos bucket.
 objectLockRetention(bucketName, client, textFileName)            // Put objectlock retention(i.e. retain until date) on the
object and get the configured retention.
 objectLocklegal-hold(bucketName, client, textFileName)            // Put objectlock legal-hold on the object and get the legal-
hold status.

}
```

Terraform

```
// Create COS instance.
resource "ibm_resource_instance" "cos_instance" {
  name              = "cos-instance"
  resource_group_id = data.ibm_resource_group.cos_group.id
  service           = "cloud-object-storage"
  plan              = "standard"
  location          = "global"
}

// Create a new bucket with objectlock and object versioning enabled.
resource "ibm_cos_bucket" "bucket" {
  bucket_name           = var.bucket_name
  resource_instance_id  = ibm_resource_instance.cos_instance.id
  region_location  = var.regional_loc
  storage_class          = var.standard_storage_class
  object_versioning {
    enable  = true
  }
  object_lock = true
}

// Set object lock configuration on the bucket by providing the crn of the new COS bucket.
resource ibm_cos_bucket_objectlock_configuration "objectlock" {
 bucket_crn      = ibm_cos_bucket.bucket.crn
 bucket_location = var.regional_loc
 object_lock_configuration{
   objectlockenabled = "Enabled"
   objectlockrule{
     defaultretention{
        mode = "COMPLIANCE"
        days = 6
      }
    }
  }
}

// Upload an object to the COS bucket with objectlock retention and objectlock legal-hold.
resource "ibm_cos_bucket_object" "object_object_lock" {
  bucket_crn      = ibm_cos_bucket.bucket.crn
  bucket_location = ibm_cos_bucket.bucket.region_location
  content         = "Hello World 2"
  key             = "plaintext5.txt"
  object_lock_mode              = "COMPLIANCE"
  object_lock_retain_until_date = "2023-02-15T18:00:00Z"
  object_lock_legal_hold_status = "ON"
  force_delete = true
}
```

Setting a quota on a bucket

A hard quota sets a maximum amount of storage (in bytes) available for a bucket. Once reached, the limit prevents adding any additional objects to the bucket until existing objects are moved or deleted to free up space, or the quota is raised.

Note: This feature is not currently supported in Object Storage for Satellite. Learn more.

There are two types of usage quota: a "hard" quota described above, and a "soft" quota that alerts a user that usage has crossed a threshold, but does not prevent any further object writes. To configure a soft quota, make use of Configure Metrics for IBM Cloud® Object Storage to set usage alerts.

Using the console

You can use the console to add a hard quota to a bucket during creation, or an existing bucket.

Creating a new bucket with a quota

1. After navigating to your object storage instance, click on **Create bucket**.
2. Under *Advanced configurations*, look for **Quota enforcement** and toggle the selector to **Enabled**.
3. Now raise or lower the value and choose the appropriate storage unit. Then, click **Save**.
4. Continue configuring any other rules, setting, or policies on the new bucket.

Adding a quota to an existing bucket

First, make sure that you have a bucket. If not, follow the getting started tutorial to become familiar with the console.

1. Navigate to a bucket, so that you are looking at a list of objects. Select **Configuration** from the navigational menu.
2. Under *Advanced configurations*, look for **Quota enforcement** and toggle the selector to **Enabled**.
3. Now raise or lower the value and choose the appropriate storage unit. Then, click **Save**.

Disabling or editing a quota

1. Navigate to the bucket where you want to change the quota, so that you are looking at a list of objects. Select **Configuration** from the navigational menu.
2. Under *Advanced configurations*, look for **Quota enforcement**.

3. If you want to disable the quota enforcement, toggle the selector to **Disable**. Alternatively, keep the quota enforcement enabled, but edit the values as needed.
4. Click **Save**.

Using an API

Bucket quotas are managed with the COS Resource Configuration API.

To add a quota, you send a `PATCH` request to edit the bucket's metadata:

```
$ curl -X PATCH https://config.cloud-object-storage.cloud.ibm.com/v1/b/my-bucket \
     -H 'authorization: bearer $IAM_TOKEN' \
     -d '{"hard_quota": 10000000000}'
```

To disable the quota, set it to zero:

```
$ curl -X PATCH https://config.cloud-object-storage.cloud.ibm.com/v1/b/my-bucket \
     -H 'authorization: bearer $IAM_TOKEN' \
     -d '{"hard_quota": 0}'
```

To temporarily disable writing new data to the bucket, set the quota to a very small integer:

```
$ curl -X PATCH https://config.cloud-object-storage.cloud.ibm.com/v1/b/my-bucket \
     -H 'authorization: bearer $IAM_TOKEN' \
     -d '{"hard_quota": 1}'
```

To check the quota on a bucket, send a GET request to view the `hard_quota` field in the bucket's metadata:

```
$ curl https://config.cloud-object-storage.cloud.ibm.com/v1/b/my-bucket \
     -H 'authorization: bearer $IAM_TOKEN'
```

Using storage classes

Not all data feeds active workloads. Archival data might sit untouched for long periods of time. For less active workloads, you can create buckets with different storage classes. Objects that are stored in these buckets incur charges on a different schedule than standard storage.

Note: This feature is not currently supported in Object Storage for Satellite. Learn more.

What are the classes?

You can choose from four storage classes:

- **Smart Tier** can be used for any workload, especially dynamic workloads where access patterns are unknown or difficult to predict. Smart Tier provides a simplified pricing structure and automatic cost optimization by classifying the data into "hot", "cool", and "cold" tiers based on monthly usage patterns. All data in the bucket is then billed at the lowest applicable rate. There are no threshold object sizes or storage periods, and there are no retrieval fees. For a detailed explanation of how it works, see the billing topic.
- **Standard** is used for active workloads, with no charge for data retrieved (other than the cost of the operational request itself).
- **Vault** is used for cool workloads where data is accessed less than once a month - an extra retrieval charge ($/GB) is applied each time data is read. The service includes a minimum threshold for object size and storage period consistent with the intended use of this service for cooler, less-active data.
- **Cold Vault** is used for cold workloads where data is accessed every 90 days or less - a larger extra retrieval charge ($/GB) is applied each time data is read. The service includes a longer minimum threshold for object size and storage period consistent with the intended use of this service for cold, inactive data.

Note: Flex has been replaced by Smart Tier for dynamic workloads. Flex users can continue to manage their data in existing Flex buckets, although no new Flex buckets may be created. Existing users can reference pricing information here.

For more information, see the pricing table at ibm.com .

Important: The **Active** storage class is only used with One Rate plans, and cannot be used in a Standard plan instance.

For more information about how to create buckets with different storage classes, see the [API reference](#).

Important: For each storage class, billing is based on aggregated usage across all buckets at the instance level. For example, for Smart Tier, the billing is based on usage across all Smart Tier buckets in a given instance - not on the individual buckets.

How do I create a bucket with a different storage class?

When creating a bucket in the console, there is a menu that allows for storage class selection.

When creating buckets programmatically, it is necessary to specify a `LocationConstraint` that corresponds with the endpoint used. Valid provisioning codes for `LocationConstraint` are

BR São Paulo `br-sao-standard` / `br-sao-vault` / `br-sao-cold` / `br-sao-smart`
US Geo `us-standard` / `us-vault` / `us-cold` / `us-smart`
US East `us-east-standard` / `us-east-vault` / `us-east-cold` / `us-east-smart`
US South `us-south-standard` / `us-south-vault` / `us-south-cold` / `us-south-smart`
EU Geo `eu-standard` / `eu-vault` / `eu-cold` / `eu-smart`
EU Great Britain `eu-gb-standard` / `eu-gb-vault` / `eu-gb-cold` / `eu-gb-smart`
EU Germany `eu-de-standard` / `eu-de-vault` / `eu-de-cold` / `eu-de-smart`
EU Spain `eu-es-standard` / `eu-es-vault` / `eu-es-cold` / `eu-es-smart`
AP Geo `ap-standard` / `ap-vault` / `ap-cold` / `ap-smart`
AP Tokyo `jp-tok-standard` / `jp-tok-vault` / `jp-tok-cold` / `jp-tok-smart`
AP Osaka `jp-osa-standard` / `jp-osa-vault` / `jp-osa-cold` / `jp-osa-smart`
AP Australia `au-syd-standard` / `au-syd-vault` / `au-syd-cold` / `au-syd-smart`
CA Toronto `ca-tor-standard` / `ca-tor-vault` / `ca-tor-cold` / `ca-tor-smart`
Amsterdam `ams03-standard` / `ams03-vault` / `ams03-cold` / `ams03-smart`
Chennai `che01-standard` / `che01-vault` / `che01-cold` / `che01-smart`
Milan `mil01-standard` / `mil01-vault` / `mil01-cold` / `mil01-smart`
Montréal `mon01-standard` / `mon01-vault` / `mon01-cold` / `mon01-smart`
Paris `par01-standard` / `par01-vault` / `par01-cold` / `par01-smart`
San Jose `sjc04-standard` / `sjc04-vault` / `sjc04-cold` / `sjc04-smart`
Singapore `sng01-standard` / `sng01-vault` / `sng01-cold` / `sng01-smart`

For more information about endpoints, see [Endpoints and storage locations](#).

Using the REST API, Libraries, and SDKs

Several new APIs have been introduced to the IBM COS SDKs to provide support for applications working with retention policies. Select a language (curl, Java, JavaScript, Go, or Python) at the beginning of this page to view examples that use the appropriate COS SDK.

All code examples assume the existence of a client object that is called `cos` that can call the different methods. For details on creating clients, see the specific SDK guides.

Create a bucket with a storage class

Java

```
public static void createBucket(String bucketName) {
    System.out.printf("Creating new bucket: %s\n", bucketName);
    _cos.createBucket(bucketName, "us-vault");
    System.out.printf("Bucket: %s created!\n", bucketName);
}
```

Node

```
function createBucket(bucketName) {
    console.log(`Creating new bucket: ${bucketName}`);
    return cos.createBucket({
        Bucket: bucketName,
        CreateBucketConfiguration: {
          LocationConstraint: 'us-standard'
        },
    }).promise()
    .then((() => {
        console.log(`Bucket: ${bucketName} created!`);
    }))
    .catch((e) => {
        console.error(`ERROR: ${e.code} - ${e.message}\n`);
    });
}
```

Python

```
def create_bucket(bucket_name):
    print("Creating new bucket: {0}".format(bucket_name))
    try:
        cos.Bucket(bucket_name).create(
            CreateBucketConfiguration={
                "LocationConstraint":COS_BUCKET_LOCATION
            }
        )
        print("Bucket: {0} created!".format(bucket_name))
    except ClientError as be:
        print("CLIENT ERROR: {0}\n".format(be))
    except Exception as e:
        print("Unable to create bucket: {0}".format(e))
```

Go

```
func main() {

    // Create client
    sess := session.Must(session.NewSession())
    client := s3.New(sess, conf)

    // Bucket Names
    newBucket := "<NEW_BUCKET_NAME>"

    input := &s3.CreateBucketInput{
        Bucket: aws.String(newBucket),
        CreateBucketConfiguration: &s3.CreateBucketConfiguration{
            LocationConstraint: aws.String("us-cold"),
        },
    }
    client.CreateBucket(input)

    d, _ := client.ListBuckets(&s3.ListBucketsInput{})
    fmt.Println(d)
}
```

Curl

```
curl -X "PUT" "https://(endpoint)/(bucket-name)"
 -H "Content-Type: text/plain; charset=utf-8"
 -H "Authorization: Bearer (token)"
 -H "ibm-service-instance-id: (resource-instance-id)"
 -d "<CreateBucketConfiguration>
       <LocationConstraint>(provisioning-code)</LocationConstraint>
     </CreateBucketConfiguration>"
```

It isn't possible to change the storage class of a bucket once the bucket is created. If objects need to be reclassified, it's necessary to move the data to another bucket with the wanted storage class.

Moving data between buckets

At some point it will become necessary to move or backup your data to a different IBM Cloud® Object Storage region. One approach to moving or replicating data across object storage regions is to use a 'sync' or 'clone' tool, such as the open-source `rclone` command-line utility. This utility syncs a file tree between two locations, including cloud object storage. When `rclone` writes data to COS, it uses the COS/S3 API to segment large objects and uploads the parts in parallel according to sizes and thresholds set as configuration parameters.

This guide provides instructions for copying data from one IBM Cloud Object Storage bucket to another Object Storage bucket within the same region or to a second Object Storage bucket in a different Object Storage region. These steps need to be repeated for all the data that you want to copy from each bucket. After the data is migrated you can verify the integrity of the transfer by using `rclone check`, which will produce a list of any objects that don't match either file size or checksum. Additionally, you can keep buckets in sync by regularly running `rclone sync` from your available sources to your chosen destinations.

Create a destination IBM Cloud Object Storage bucket

You have the option of using your existing instance of IBM Cloud Object Storage or creating a new instance. If you want to reuse your existing instance, skip to step #2.

1. Create an instance of IBM Cloud Object Storage from the catalog.
2. Create any buckets that you need to store your transferred data. Read through the getting started guide to familiarize yourself with key concepts such as endpoints and storage classes.
3. **The `rclone` utility will not copy any bucket configurations or object metadata** . Therefore, if you are using any of the Object Storage features such as expiration, archive, key protect, and so on. be sure to configure them appropriately before migrating your data. To view which features are supported at your COS destination, please refer to the feature matrix.

Feature configuration and access policies documentation can be viewed at the IBM Cloud portal pages listed below:

- IBM Cloud Identity and Access Management - IAM
- Activity Tracking Events
- Metrics Monitoring
- Object Expiry
- Cloud Object Storage Firewall
- Content Delivery Network - CDN
- Archive
- Key Protect
- IBM Cloud Functions

Set up a compute resource to run the migration tool

1. Choose a Linux™/macOS™/BSD™ machine or an IBM Cloud Infrastructure Bare Metal or Virtual Server with the best proximity to your data. Selecting a data center in the same region as the destination bucket is generally the best choice (for example, if moving data from `mel01` to `au-syd`, use a VM or Bare Metal in `au-syd`). The recommended Server configuration is: 32 GB RAM, 2-4 core processor, and private network speed of 1000 Mbps.
2. If you are running the migration on an IBM Cloud Infrastructure Bare Metal or Virtual Server use the **private** COS endpoints to avoid network egress charges.
3. Otherwise, use the **public** or **direct** COS endpoints.
4. Install `rclone` from either a package manager or a pre-compiled binary .

```
$ curl https://rclone.org/install.sh | sudo bash
```

Configure `rclone` for COS source data

Create 'profiles' for your source and destination of the migration in `rclone`.

If needed, obtain COS credentials

1. Select your COS instance in the IBM Cloud console.
2. Click **Service Credentials** in the navigation pane.
3. Click **New credential** to generate credential information.
4. Select **Advanced** options.
5. Turn HMAC credentials to **On**.
6. Click **Add**.
7. View the credential that you created, and copy the JSON contents.

Get COS endpoint

1. Click **Buckets** in the navigation pane.
2. Click the migration destination bucket.
3. Click **Configuration** in the navigation pane.
4. Scroll down to the **Endpoints** section and choose the endpoint based on where you are running the migration tool.

5. Create the COS destination by copying the following and pasting into `rclone.conf`.

```
[COS_SOURCE]
type = s3
provider = IBMCOS
env_auth = false
access_key_id =
secret_access_key =
endpoint =
```

Use `[COS_DESTINATION]` as the name of the profile you need to create to configure the destination. Repeat the steps above,

Using your credentials and desired endpoint, complete the following fields:

```
access_key_id = <access_key_id>
secret_access_key = <secret_access_key>
endpoint = <bucket endpoint>
```

Configure `rclone` for COS destination data

Repeat the previous steps for the destination buckets.

Verify that the source and destination are properly configured

- List the buckets associated with the source to verify `rclone` is properly configured.

```
$ rclone lsd COS_SOURCE:
```

- List the buckets associated with the destination to verify `rclone` is properly configured.

```
$ rclone lsd COS_DESTINATION:
```

Note: If you are using the same COS instance for the source and destination, the bucket listings will match.

Run `rclone`

1. Test your configuration with a dry run (where no data is copied) of `rclone` to test the copy of the objects in your source bucket (for example, `source-test`) to target bucket (for example, `destination-test`).

   ```
   $ rclone --dry-run copy COS_SOURCE:source-test COS_DESTINATION:destination-test
   ```

2. Check that the files you want to migrate appear in the command output. If everything looks good, remove the `--dry-run` flag and, optionally add `-v` and/or `-P` flag to copy the data and track progress. Using the optional `--checksum` flag avoids updating any files that have the same MD5 hash and object size in both locations.

   ```
   $ rclone -v -P copy --checksum COS_SOURCE:source-test COS_DESTINATION:destination-test
   ```

Try to max out the CPU, memory, and network on the machine running `rclone` to get the fastest transfer time.

There are other parameters to consider when tuning `rclone`. Different combinations of these values will impact CPU, memory, and transfer times for the objects in your bucket.

Flag	Type	Description
`--checkers`	`int`	Number of checkers to run in parallel (default 8). This is the number of checksums compare threads running. We recommend increasing this to 64 or more.
`--transfers`	`int`	This is the number of objects to transfer in parallel (default 4). We recommend increasing this to 64 or 128 or higher when transferring many small files.
`--multi-thread-streams`	`int`	Download large files (> 250M) in multiple parts in parallel. This will improve the download time of large files (default 4).

`--s3-upload-concurrency`	`int`	The number of parts of large files (> 200M) to upload in parallel. This will improve the upload time of large files (default 4).

{: caption="`rclone` options" caption-side="top"}

Tip: Migrating data using `rclone copy` only copies but does not delete the source data.

The copy process should be repeated for all other source buckets that require migration/copy/backup.

Emptying a bucket

This overview focuses on the steps that are needed to access a list of all items in a bucket within an instance of IBM Cloud® Object Storage for the purpose of deleting each one sequentially.

The process of emptying a bucket is familiar to anyone who has to delete buckets in their instance of IBM Cloud Object Storage because a bucket has to be empty to be deleted. There may be other reasons you may wish to delete items, but want to avoid deleting every object individually. This code pattern for the supported SDKs will allow you to define your configuration, create a client, and then connect with that client to get a list of all the items in an identified bucket to delete them.

Important: If versioning is enabled, then expiration rules create delete markers.

Tip: It is a best practice to avoid putting credentials in scripts. This example is for testing and educational purposes, and your specific setup should be informed by best practices and Developer Guidance.

Before you begin

Specific instructions for downloading and installing SDKs are available for Python, Node.js, Java, and Go. Also, when working with Command Line Instructions (CLI) and your CLI clients, please check out the pertinent information related to Object Storage regarding AWS compatibility, Minio, and `rclone`.

For this code pattern you will need:

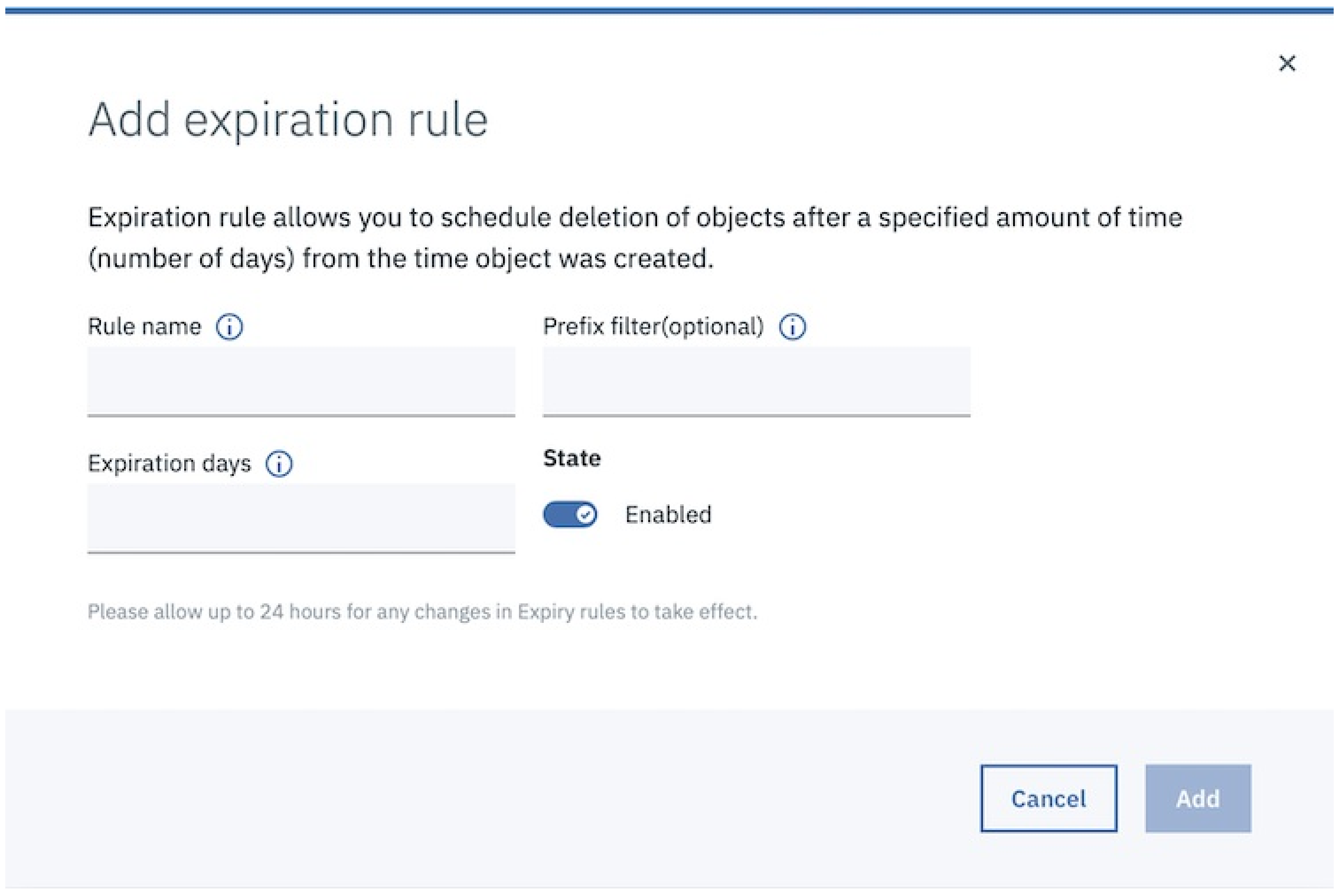
- An IBM Cloud® Platform account
- An instance of IBM Cloud Object Storage
- Configured and operational use of IBM Cloud Object Storage SDKs for your choice of Java, Python, NodeJS, or Go; or, a configured and operational CLI client.

Using the Console

Before getting to the examples, there is one way to empty a bucket via the GUI at the IBM Cloud console: using an expiration rule on the bucket itself. For more information, please see the documentation on deleting stale data with expiration rules.

After logging in to Object Storage, choose your storage instance. Then, select your bucket from the list of your buckets. To set the rule to delete the items, select **Configuration** from the navigation menu and click **Add rule** under the *Expiration rule* section. Set the number of days to '1' to delete all the items after one day.

Add Expiration Rule to delete items


Add expiration rule

Expiration rule allows you to schedule deletion of objects after a specified amount of time (number of days) from the time object was created.

Please allow up to 24 hours for any changes in Expiry rules to take effect.

Tip: The process for rule completion can take up to 24 hours, and is on a set schedule. Please take this into consideration when applying this technique.

CLI Client Examples

There are many tools available to help users make the most of Object Storage and the following CLI clients offer simple ways of emptying buckets.

Tip: Sample instructions are provided for using a client application or command line once your CLI client has been configured and is operational.

`rclone` example

The `rclone` tool is typically used to keep directories synchronized and for migrating data between storage platforms. You can learn more from the documentation on using `rclone`.

```
$ rclone purge {remote}:{path} [flags]
```

Minio example

The open source Minio client allows you to use UNIX-like commands (`ls`, `cp`, `cat`, and so on) with IBM Cloud® Object Storage. For more information, check out using Minio.

```
$ mc rm --recursive --force {instance-alias}/{bucket-name}
```

AWS example

The official command-line interface for AWS is compatible with the IBM Cloud Object Storage S3 API and you can find out more on how to use the AWS CLI.

```
$ aws s3 rm s3://{bucket-name} --recursive
```

Code Example

Deleting an entire directory or removing all the contents of a bucket can be time consuming deleting each object, one at a time. The ability to delete one item at a time can be leveraged to save time and effort by collecting a list of all the items before deletion.

Tip: The topic itself points out the danger of deletion: data **will** be lost. Of course, when that is the goal, caution should be exercised that only the targeted deletions should occur. Check—and double-check—instances, bucket names, and any prefixes or paths that need to be specified.

Overview

The code pattern in this exercise configures a client before creating one for the purpose of gathering a list of items for the purpose of deleting each object.

The code pattern in this exercise configures a client before creating one for the purpose of gathering a list of items for the purpose of deleting each object.

The code pattern in this exercise configures a client before creating one for the purpose of gathering a list of items for the purpose of deleting each object.

The code pattern in this exercise configures a client before creating one for the purpose of gathering a list of items for the purpose of deleting each object.

Node

```
const myCOS = require('ibm-cos-sdk');

var config = {
    endpoint: 's3.us-geo.cloud-object-storage.appdomain.cloud',
    apiKeyId: 'd-2-DO-NOT-USEevplExampleo_t3ZTCJO',
    ibmAuthEndpoint: 'https://iam.cloud.ibm.com/identity/token',
    serviceInstanceId: 'crn:v1:bluemix:public:cloud-object-storage:global:a/SAMPLE253abf6e65dca920c9d58126b:3f656f43-5c2a-941e-
b128-DO-NOT-USE::',
};

var cosClient = new myCOS.S3(config);

function logDone() {
    console.log('COMPLETE!\n');
}

function logError(e) {
    console.log(`ERROR: ${e.code} - ${e.message}\n`);
}

// Retrieve the list of contents from a bucket for deletion
function deleteContents(bucketName) {
    var returnArr = new Array();

    console.log(`Retrieving bucket contents from: ${bucketName}\n`);
    return cosClient.listObjects(
        {Bucket: bucketName},
    ).promise()
    .then((data) => {
        if (data != null && data.Contents != null) {
            for (var i = 0; i < data.Contents.length; i++) {
                returnArr.push(data.Contents[i].Key);
                var itemKey = data.Contents[i].Key;
                var itemSize = data.Contents[i].Size;
                console.log(`Item: ${itemKey} (${itemSize} bytes).\n`)
            }
            deleteItem(bucketName, itemName);
            logDone();
        }
    })
    .catch(logError);
}

// Delete item
function deleteItem(bucketName, itemName) {
    console.log(`Deleting item: ${itemName}`);
    return cosClient.deleteObject({
        Bucket: bucketName,
        Key: itemName
    }).promise()
    .then(() =>{
        console.log(`Item: ${itemName} deleted!`);
    })
    .catch(logError);
}

function main() {
    try {
        var BucketName = "<BUCKET_NAME>";

        deleteContents(BucketName);
    }
    catch(ex) {
        logError(ex);
    }
}

main();
```

Python

```
import ibm_boto3
from ibm_botocore.client import Config, ClientError

# Constants for IBM COS values
COS_ENDPOINT = "<endpoint>" # Current list avaiable at https://control.cloud-object-storage.cloud.ibm.com/v2/endpoints
COS_API_KEY_ID = "<api-key>" # eg "W00YiRnLW4a3fTjMB-odB-2ySfTrFBIQQWanc--P3byk"
COS_AUTH_ENDPOINT = "https://iam.cloud.ibm.com/identity/token"
COS_RESOURCE_CRN = "<resource-instance-id>" # eg "crn:v1:bluemix:public:cloud-object-
storage:global:a/3bf0d9003abfb5d29761c3e97696b71c:d6f04d83-6c4f-4a62-a165-696756d63903::"

# Create resource
cos = ibm_boto3.resource("s3",
    ibm_api_key_id=COS_API_KEY_ID,
    ibm_service_instance_id=COS_RESOURCE_CRN,
    ibm_auth_endpoint=COS_AUTH_ENDPOINT,
    config=Config(signature_version="oauth"),
    endpoint_url=COS_ENDPOINT
)

def get_bucket_contents(bucket_name, max_keys):
    print("Retrieving bucket contents from: {0}".format(bucket_name))
    returnArray = []
    try:
        files = cos.Bucket(bucket_name).objects.all()
        for file in files:
            print("Item: {0} ({1} bytes).".format(file["Key"], file["Size"]))
            returnArray.append(file["Key"])

    except ClientError as be:
        print("CLIENT ERROR: {0}\n".format(be))
    except Exception as e:
        print("Unable to retrieve bucket contents: {0}".format(e))

    return returnArray

def delete_item(bucket_name, item_name):
    print("Deleting item: {0}".format(item_name))
    try:
        cos.Object(bucket_name, item_name).delete()
        print("Item: {0} deleted!".format(item_name))
    except ClientError as be:
        print("CLIENT ERROR: {0}\n".format(be))
    except Exception as e:
        print("Unable to delete item: {0}".format(e))

def main():
    bucket = "<bucket_name>"
    deleteListArray = get_bucket_contents(bucket, 1000)
    for item_name in deleteListArray:
        delete_item(bucket, item_name)
```

```
main()
```

Java

```
package com.cos;

    import java.sql.Timestamp;
    import java.util.List;

    import com.ibm.cloud.objectstorage.ClientConfiguration;
    import com.ibm.cloud.objectstorage.SDKGlobalConfiguration;
    import com.ibm.cloud.objectstorage.auth.AWSCredentials;
    import com.ibm.cloud.objectstorage.auth.AWSStaticCredentialsProvider;
    import com.ibm.cloud.objectstorage.auth.BasicAWSCredentials;
    import com.ibm.cloud.objectstorage.client.builder.AwsClientBuilder.EndpointConfiguration;
    import com.ibm.cloud.objectstorage.services.s3.AmazonS3;
    import com.ibm.cloud.objectstorage.services.s3.AmazonS3ClientBuilder;
    import com.ibm.cloud.objectstorage.services.s3.model.Bucket;
    import com.ibm.cloud.objectstorage.services.s3.model.ListObjectsRequest;
    import com.ibm.cloud.objectstorage.services.s3.model.ListObjectsV2Request;
    import com.ibm.cloud.objectstorage.services.s3.model.ListObjectsV2Result;
    import com.ibm.cloud.objectstorage.services.s3.model.ObjectListing;
    import com.ibm.cloud.objectstorage.services.s3.model.S3ObjectSummary;
    import com.ibm.cloud.objectstorage.oauth.BasicIBMOAuthCredentials;

    public class CosDeleteMultipleItems
    {

        private static AmazonS3 _cosClient;

        /**
         * @param args
         */
        public static void main(String[] args)
        {

            SDKGlobalConfiguration.IAM_ENDPOINT = "https://iam.cloud.ibm.com/identity/token";

            String bucketName = "<BUCKET_NAME>";  // eg my-unique-bucket-name
            String newBucketName = "<NEW_BUCKET_NAME>"; // eg my-other-unique-bucket-name
            String api_key = "<API_KEY>"; // eg "W00YiRnLW4k3fTjMB-oiB-2ySfTrFBIQQWanc--P3byk"
            String service_instance_id = "<SERVICE_INSTANCE_ID"; // eg "crn:v1:bluemix:public:cloud-object-
storage:global:a/3bf0d9003abfb5d29761c3e97696b71c:d6f04d83-6c4f-4a62-a165-696756d63903::"
            String endpoint_url = "https://s3.us.cloud-object-storage.appdomain.cloud"; // this could be any service endpoint

            String storageClass = "us-geo";
            String location = "us";

            _cosClient = createClient(api_key, service_instance_id, endpoint_url, location);

            contentsForDeletion(bucketName, 1000);

            for(Int deletedItem: itemsForDeletion) {
                deleteItem(bucketName, deletedItem.getKey());
                System.out.printf("Deleted item: %s\n", deletedItem.getKey());
            }
        }

        /**
         * @param api_key
         * @param service_instance_id
         * @param endpoint_url
         * @param location
         * @return AmazonS3
         */
        public static AmazonS3 createClient(String api_key, String service_instance_id, String endpoint_url, String location)
        {
            AWSCredentials credentials;
            credentials = new BasicIBMOAuthCredentials(api_key, service_instance_id);
            ClientConfiguration clientConfig = new ClientConfiguration().withRequestTimeout(5000);
            clientConfig.setUseTcpKeepAlive(true);

            AmazonS3 cosClient = AmazonS3ClientBuilder.standard().withCredentials(new AWSStaticCredentialsProvider(credentials))
                    .withEndpointConfiguration(new EndpointConfiguration(endpoint_url,
location)).withPathStyleAccessEnabled(true)
                    .withClientConfiguration(clientConfig).build();
            return cosClient;
        }

        public static List contentsForDeletion(String bucketName, int maxKeys) {
            System.out.printf("Retrieving bucket contents (V2) from: %s\n", bucketName);

            boolean moreResults = true;
            String nextToken = "";

            while (moreResults) {
                ListObjectsV2Request request = new ListObjectsV2Request()
                    .withBucketName(bucketName)
                    .withMaxKeys(maxKeys)
                    .withContinuationToken(nextToken);

                ListObjectsV2Result result = _cosClient.listObjectsV2(request);
                for(S3ObjectSummary objectSummary : result.getObjectSummaries()) {
                    System.out.printf("Item: %s (%s bytes)\n", objectSummary.getKey(), objectSummary.getSize());
                    deleteItem(bucketName, objectSummary.getKey());
                }

                if (result.isTruncated()) {
                    nextToken = result.getNextContinuationToken();
                    System.out.println("...More results in next batch!\n");
                }
                else {
                    nextToken = "";
                    moreResults = false;
                }
            }
            System.out.println("...No more results!");
        }

        public static void deleteItem(String bucketName, String itemName) {
            System.out.printf("Deleting item: %s\n", itemName);
            _cosClient.deleteObject(bucketName, itemName);
            System.out.printf("Item: %s deleted!\n", itemName);
        }

    }
```

Go

```
import (
    "github.com/IBM/ibm-cos-sdk-go/aws/credentials/ibmiam"
    "github.com/IBM/ibm-cos-sdk-go/aws"
    "github.com/IBM/ibm-cos-sdk-go/aws/session"
    "github.com/IBM/ibm-cos-sdk-go/service/s3"
)

// Constants for IBM COS values
const (
    apiKey            = "<API_KEY>"  // eg "0viPHOY7LbLNa9eLftrtHPpTjoGv6hbLD1QalRXikliJ"
    serviceInstanceID = "<RESOURCE_INSTANCE_ID>" // "crn:v1:bluemix:public:cloud-object-
storage:global:a/<CREDENTIAL_ID_AS_GENERATED>:<SERVICE_ID_AS_GENERATED>::"
    authEndpoint      = "https://iam.cloud.ibm.com/identity/token"
    serviceEndpoint   = "<SERVICE_ENDPOINT>" // eg "https://s3.us.cloud-object-storage.appdomain.cloud"
    bucketLocation    = "<LOCATION>" // eg "us"
)

// Create config
conf := aws.NewConfig().
```

```
    WithRegion("us-standard").
    WithEndpoint(serviceEndpoint).
    WithCredentials(ibmiam.NewStaticCredentials(aws.NewConfig(), authEndpoint, apiKey, serviceInstanceID)).
    WithS3ForcePathStyle(true)

func main() {

    // Create client
    sess := session.Must(session.NewSession())
    client := s3.New(sess, conf)

    // Bucket Names
    Bucket := "<BUCKET_NAME>"
    Input := &s3.ListObjectsV2Input{
            Bucket: aws.String(Bucket),
        }

    res, _ := client.ListObjectsV2(Input)

    for _, item := range res.Contents {
        input := &s3.DeleteObjectInput{
                Bucket: aws.String(Bucket),
                Key:    aws.String(*item.Key),
            }
        d, _ := client.DeleteObject(input)
        fmt.Println(d)
    }
}
```

Next Steps

Leveraging new and existing capabilities of the tools covered in this overview can be explored further at the IBM Cloud Platform.

Comparing IBM Cloud Object Storage to FTP

The File Transfer Protocol, (FTP) is a popular way to transfer files, but how does it compare to IBM Cloud® Object Storage?

IBM Cloud Object Storage stores encrypted and dispersed data across multiple geographic locations. The information is accessible over popular protocols like HTTPS using a modern RESTful API. FTP, by contrast, requires both a client and a server application and uses an insecure protocol by default. Object Storage does all the work of the server FTP daemon (`ftpd`) and offers more options for security and validation than can be obtained from `ftpd` or similar services.

How IBM Cloud Object Storage is similar to FTP

If you've used FTP in the past, you have either worked from the command line or from a client application that uses a GUI. Cyberduck is a popular, open source, and easy-to-use graphical interface for either IBM Cloud Object Storage or FTP.

Cyberduck provides full operational visibility in connecting to IBM Cloud Object Storage. Cyberduck is downloaded from cyberduck.io/. Once you have it installed, you can configure it to connect to your instance of Object Storage.

Use Cyberduck to create a connection to IBM Cloud Object Storage. Then, synchronize a folder of local files to a bucket. After you complete getting started Object Storage and obtained your credentials, follow these steps:

1. Download, install, and start Cyberduck.
2. When the application opens, you can create a connection to Object Storage. Click **Open Connection** to configure the connection.
3. A pop-up window opens. From the menu, select the option, `Amazon S3`. Enter your information into the following fields:
    - `Server`: enter the appropriate endpoint for your data at IBM Cloud Object Storage

    ⚠ **Important:** Ensure that the endpoint region matches the intended bucket. For more information about endpoints, see Endpoints and storage locations.

    - `Access Key ID` generated by selecting the appropriate HMAC option when creating a Service Credential;
    - `Secret Access Key` also from the HMAC option.
    - `Add to Keychain`: Save the configuration to the your personal keychain *(optional)*.

    - Ignore the other options like the `Anonymous Login` checkbox, and `SSH Private Key`.

4. Cyberduck takes you to the root of the account where buckets can be created.
    - Right-click within the main pane and select **New Folder**.

    Tip: Cyberduck supports many transfer protocols where Folder is the more common name for a container construct.

    - Enter the bucket name and then click Create.

5. After the bucket is created, double-click the bucket to view it. Within the bucket you can perform various functions such as:
    - Upload files to the bucket
    - List bucket contents
    - Download objects from the bucket
    - Synchronize local files to a bucket
    - Synchronize objects to another bucket
    - Create an archive of a bucket

6. Right-click within the bucket and select **Synchronize**. A pop-up window opens where you can browse to the folder that you want to synchronize to the bucket. Select the folder and click Choose.

7. After you select the folder, a new pop-up window opens. Here, a drop-down menu is available where you select the synchronization operation with the bucket. Three possible synchronize options are available from the menu:
    - `Download`: This downloads changed and missing objects from the bucket.
    - `Upload`: This uploads changed and missing files to the bucket.
    - `Mirror`: This performs both download and upload operations, ensuring that all new and updated files and objects are synchronized between the local folder and the bucket.

How Object Storage is different from FTP

Technically speaking, there are more differences than similarities between FTP and Object Storage. Starting from the convenience of not having to run a server application like `ftpd` and continuing through the security of using a secure protocol like HTTPS, the list of differences is lengthy and significant.

Next Steps

Can FTP provide an API or libraries? We think not! Learn more about what is available for developers of IBM Cloud Object Storage.

Data management

Upload data

After getting your storage organized into buckets, it's time to add some objects by uploading data.

Depending on how you want to use your storage, there are different ways to get data into the system. A data scientist has a few large files that are used for analytics, a systems administrator needs to keep database backups synchronized with local files, and a developer is writing software that needs to read and write millions of files. Each of these scenarios is best served by different methods of data ingest.

Tip: Some applications may wish to restrict a user or Service ID to only uploading data, without any access to reading data in a bucket. This is possible through the Object Writer IAM role.

Using the console

Typically, using the web-based console is not the most common way to use Object Storage. Objects are limited to 200 MB and the file name and key are identical. Multiple objects can be uploaded at the same time, and if the browser allows for multiple threads each object will be uploaded by using multiple parts in parallel. Support for larger object sizes and improved performance (depending on network factors) is provided by Aspera high-speed transfer.

Using a compatible tool

Some users want to use a stand-alone utility to interact with their storage. As the Cloud Object Storage API supports the most common set of S3 API operations, many S3-compatible tools can also connect to Object Storage by using HMAC credentials.

Some examples include file explorers like Cyberduck or Transmit, backup utilities like Cloudberry and Duplicati, command-line utilities like s3cmd or Minio Client, and many others.

Using the API

Most programmatic applications of Object Storage use an SDK (such as Java, node.js, or Python) or the Cloud Object Storage API. Typically objects are uploaded in multiple parts, with part size and number of parts configured by a Transfer Manager class.

Conditional requests

When making a request to read or write data, it is possible to set conditions on that request to avoid unnecessary operations. This is accomplished using the following pre-conditional HTTP headers: `If-Match`, `If-None-Match`, `If-Modified-Since`, and `If-Unmodified-Since`.

Note: It is generally preferable to use `If-Match` because the granularity of the `Last-Modified` value is only in seconds, and may not be sufficient to avoid race conditions in some applications.

Using `If-Match`

On an object PUT, HEAD, or GET request, the `If-Match` header will check to see if a provided `Etag` (MD5 hash of the object content) matches the provided `Etag` value. If this value matches, the operation will proceed. If the match fails, the system will return a `412 Precondition Failed` error.

> If-Match is most often used with state-changing methods (for example, POST, PUT, DELETE) to prevent accidental overwrites when multiple user agents might be acting in parallel on the same resource (that is, to prevent the "lost update" problem).

Using `If-None-Match`

On an object PUT, HEAD, or GET request, the `If-None-Match` header will check to see if a provided `Etag` (MD5 hash of the object content) matches the provided `Etag` value. If this value does not match, the operation will proceed. If the match succeeds, the system will return a `412 Precondition Failed` error on a PUT and a `304 Not Modified` on GET or HEAD.

> If-None-Match is primarily used in conditional GET requests to enable efficient updates of cached information with a minimum amount of transaction overhead. When a client desires to update one or more stored responses that have entity-tags, the client SHOULD generate an If-None-Match header field containing a list of those entity-tags when making a GET request; this allows recipient servers to send a 304 (Not Modified) response to indicate when one of those stored responses matches the selected representation.

Using `If-Modified-Since`

On an object HEAD or GET request, the `If-Modified-Since` header will check to see if the object's `Last-Modified` value (for example `Sat, 14 March 2020 19:43:31 GMT`) is newer than a provided value. If the object has been modified, the operation will proceed. If the object has not been modified, the system will return a `304 Not Modified`.

> If-Modified-Since is typically used for two distinct purposes: 1) to allow efficient updates of a cached representation that does not have an entity-tag and 2) to limit the scope of a web traversal to resources that have recently changed.

Using `If-Unmodified-Since`

On an object PUT, HEAD, or GET request, the `If-Unmodified-Since` header will check to see if the object's `Last-Modified` value (for example `Sat, 14 March 2020 19:43:31 GMT`) is equal to or earlier than a provided value. If the object has not been modified, the operation will proceed. If the `Last-Modified` value is more recent, the system will return a `412 Precondition Failed` error on a PUT and a `304 Not Modified` on GET or HEAD.

> If-Unmodified-Since is most often used with state-changing methods (for example, POST, PUT, DELETE) to prevent accidental overwrites when multiple user agents might be acting in parallel on a resource that does not supply entity-tags with its representations (that is, to prevent the "lost update" problem). It can also be used with safe methods to abort a request if the selected representation does not match one already stored (or partially stored) from a prior request.

Storing large objects

IBM Cloud® Object Storage can support single objects as large as 10 TB when using multipart uploads.

Large objects can also be uploaded by using the console with Aspera high-speed-transfer enabled. Under most scenarios, Aspera high-speed transfer results in significantly increased performance for transferring data, especially across long distances or under unstable network conditions.

Uploading objects in multiple parts

Multipart upload operations are recommended to write larger objects into Object Storage. An upload of a single object is performed as a set of parts and these parts can be uploaded independently in any order and in parallel. Upon upload completion, Object Storage then presents all parts as a single object. This provides many benefits: network interruptions do not cause large uploads to fail, uploads can be paused and restarted over time, and objects can be uploaded as they are being created.

Multipart uploads are only available for objects larger than 5 MB. For objects smaller than 50 GB, a part size of 20 MB to 100 MB is recommended for optimum performance. For larger objects, part size can be increased without significant performance impact. Multipart uploads are limited to no more than 10,000 parts of 5 GB each up to a maximum object size of 10 TB.

Due to the complexity involved in managing and optimizing parallelized uploads, many developers use libraries that provide multipart upload support.

Most tools, such as the CLIs or the IBM Cloud Console, as well as most compatible libraries and SDKs, will automatically transfer objects in multipart uploads.

Using the REST API or SDKs

Tip: Incomplete multipart uploads do persist until the object is deleted or the multipart upload is aborted. If an incomplete multipart upload is not aborted, the partial upload continues to use resources. Interfaces should be designed with this point in mind, and clean up incomplete multipart uploads.

There are three phases to uploading an object in multiple parts:

1. The upload is initiated and an `UploadId` is created.
2. Individual parts are uploaded specifying their sequential part numbers and the `UploadId` for the object.
3. When all parts are finished uploading, the upload is completed by sending a request with the `UploadId` and an XML block that lists each part number and it's respective `Etag` value.

Tip: For more information about endpoints, see Endpoints and storage locations

Initiate a multipart upload

A `POST` issued to an object with the query parameter `upload` creates a new `UploadId` value, which is then be referenced by each part of the object being uploaded.

Syntax

```
POST https://{endpoint}/{bucket-name}/{object-name}?uploads= # path style
POST https://{bucket-name}.{endpoint}/{object-name}?uploads= # virtual host style
```

Example request

```
POST /some-bucket/multipart-object-123?uploads= HTTP/1.1
Authorization: Bearer {token}
Host: s3.us.cloud-object-storage.appdomain.cloud
```

Example response

```
HTTP/1.1 200 OK
Date: Fri, 03 Mar 2017 20:34:12 GMT
X-Clv-Request-Id: 258fdd5a-f9be-40f0-990f-5f4225e0c8e5
Accept-Ranges: bytes
Server: Cleversafe/3.9.1.114
X-Clv-S3-Version: 2.5
Content-Type: application/xml
Content-Length: 276
```

```
<InitiateMultipartUploadResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Bucket>some-bucket</Bucket>
  <Key>multipart-object-123</Key>
  <UploadId>0000015a-95e1-4326-654e-a1b57887784f</UploadId>
</InitiateMultipartUploadResult>
```

Upload a part

A `PUT` request that is issued to an object with query parameters `partNumber` and `uploadId` will upload one part of an object. The parts can be uploaded serially or in parallel, but must be numbered in order.

Syntax

```
PUT https://{endpoint}/{bucket-name}/{object-name}?partNumber={sequential-integer}&uploadId={uploadId}= # path style
PUT https://{bucket-name}.{endpoint}/{object-name}?partNumber={sequential-integer}&uploadId={uploadId}= # virtual host style
```

Example request

```
PUT /some-bucket/multipart-object-123?partNumber=1&uploadId=0000015a-df89-51d0-2790-dee1ac994053 HTTP/1.1
Authorization: Bearer {token}
Content-Type: application/pdf
Host: s3.us.cloud-object-storage.appdomain.cloud
Content-Length: 13374550
```

Example response

```
HTTP/1.1 200 OK
Date: Sat, 18 Mar 2017 03:56:41 GMT
X-Clv-Request-Id: 17ba921d-1c27-4f31-8396-2e6588be5c6d
Accept-Ranges: bytes
Server: Cleversafe/3.9.1.114
X-Clv-S3-Version: 2.5
ETag: "7417ca8d45a71b692168f0419c17fe2f"
Content-Length: 0
```

Complete a multipart upload

A `POST` request that is issued to an object with query parameter `uploadId` and the appropriate XML block in the body will complete a multipart upload.

Syntax

```
$ POST https://{endpoint}/{bucket-name}/{object-name}?uploadId={uploadId}= # path style
POST https://{bucket-name}.{endpoint}/{object-name}?uploadId={uploadId}= # virtual host style
```

```
<CompleteMultipartUpload>
  <Part>
    <PartNumber>{sequential part number}</PartNumber>
    <ETag>{ETag value from part upload response header}</ETag>
  </Part>
```

```
</CompleteMultipartUpload>
```

Example request

```
POST /some-bucket/multipart-object-123?uploadId=0000015a-df89-51d0-2790-dee1ac994053 HTTP/1.1
Authorization: Bearer {token}
Content-Type: text/plain; charset=utf-8
Host: s3.us.cloud-object-storage.appdomain.cloud
Content-Length: 257
```

```
<CompleteMultipartUpload>
  <Part>
    <PartNumber>1</PartNumber>
    <ETag>"7417ca8d45a71b692168f0419c17fe2f"</ETag>
  </Part>
  <Part>
    <PartNumber>2</PartNumber>
    <ETag>"7417ca8d45a71b692168f0419c17fe2f"</ETag>
  </Part>
</CompleteMultipartUpload>
```

Example response

```
HTTP/1.1 200 OK
Date: Fri, 03 Mar 2017 19:18:44 GMT
X-Clv-Request-Id: c8be10e7-94c4-4c03-9960-6f242b42424d
Accept-Ranges: bytes
Server: Cleversafe/3.9.1.114
X-Clv-S3-Version: 2.5
ETag: "765ba3df36cf24e49f67fc6f689dfc6e-2"
Content-Type: application/xml
Content-Length: 364
```

```
<CompleteMultipartUploadResult  xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Location>http://s3.us.cloud-object-storage.appdomain.cloud/zopse/multipart-object-123</Location>
  <Bucket>some-bucket</Bucket>
  <Key>multipart-object-123</Key>
  <ETag>"765ba3df36cf24e49f67fc6f689dfc6e-2"</ETag>
</CompleteMultipartUploadResult>
```

Abort incomplete multipart uploads

A `DELETE` request issued to an object with query parameter `uploadId` deletes all unfinished parts of a multipart upload.

Syntax

```
DELETE https://{endpoint}/{bucket-name}/{object-name}?uploadId={uploadId}= # path style
DELETE https://{bucket-name}.{endpoint}/{object-name}?uploadId={uploadId}= # virtual host style
```

Example request

```
DELETE /some-bucket/multipart-object-123?uploadId=0000015a-df89-51d0-2790-dee1ac994053 HTTP/1.1
Authorization: Bearer {token}
Host: s3.us.cloud-object-storage.appdomain.cloud
```

Example response

```
HTTP/1.1 204 No Content
Date: Thu, 16 Mar 2017 22:07:48 GMT
X-Clv-Request-Id: 06d67542-6a3f-4616-be25-fc4dbdf242ad
Accept-Ranges: bytes
Server: Cleversafe/3.9.1.114
X-Clv-S3-Version: 2.5
```

Using S3cmd (CLI)

S3cmd is a free Linux and Mac command-line tool and client for uploading, retrieving, and managing data in cloud storage service providers that use the S3 protocol. It is designed for power users who are familiar with command-line programs and is ideal for batch scripts and automated backup. S3cmd is written in Python. It's an open source project available under GNU Public License v2 (GPLv2) and is free for both commercial and private use.

S3cmd requires Python 2.6 or newer and is compatible with Python 3. The easiest way to install S3cmd is with the Python Package Index (PyPi).

```
pip install s3cmd
```

Once the package has been installed, grab the IBM Cloud® Object Storage example configuration file here and update it with your Cloud Object Storage (S3) credentials:

```
$ wget -O $HOME/.s3cfg
https://gist.githubusercontent.com/greyhoundforty/676814921b8f4367fba7604e622d10f3/raw/422abaeb70f1c17cd5308745c0e446b047c123e0/s
3cfg
```

The four lines that need to be updated are

- `access_key`
- `secret_key`
- `host_base`
- `host_bucket` {: S3cmd} This is the same whether you use the example file or the one generated by running: `s3cmd --configure`.

Once those lines have been updated with the COS details from the Customer portal, you can test the connection by issuing the command `s3cmd ls`, which will list all the buckets on the account.

```
$ s3cmd ls
2017-02-03 14:52  s3://backuptest
2017-02-06 15:04  s3://coldbackups
2017-02-03 21:23  s3://largebackup
2017-02-07 17:44  s3://winbackup
```

The full list of options and commands along with basic usage information is available on the s3tools site.

Multipart uploads with S3cmd

A `put` command will automatically run a multi-part upload when attempting to upload a file larger than the specified threshold..

```
s3cmd put FILE [FILE...] s3://BUCKET[/PREFIX]
```

The threshold is determined by the `--multipart-chunk-size-mb` option:

```
--multipart-chunk-size-mb=SIZE
    Size of each chunk of a multipart upload. Files bigger
    than SIZE are automatically uploaded as multithreaded-
    multipart, smaller files are uploaded using the
    traditional method. SIZE is in megabytes, default
    chunk size is 15MB, minimum allowed chunk size is 5MB,
    maximum is 5GB.
```

Example:

```
s3cmd put bigfile.pdf s3://backuptest/bigfile.pdf --multipart-chunk-size-mb=5
```

Output:

```
upload: 'bigfile.pdf' -> 's3://backuptest/bigfile.pdf'  [part 1 of 4, 5MB] [1 of 1]
 5242880 of 5242880   100% in    2s  1731.92 kB/s  done
upload: 'bigfile.pdf' -> 's3://backuptest/bigfile.pdf'  [part 2 of 4, 5MB] [1 of 1]
 5242880 of 5242880   100% in    2s  2001.14 kB/s  done
upload: 'bigfile.pdf' -> 's3://backuptest/bigfile.pdf'  [part 3 of 4, 5MB] [1 of 1]
 5242880 of 5242880   100% in    2s  2000.28 kB/s  done
upload: 'bigfile.pdf' -> 's3://backuptest/bigfile.pdf'  [part 4 of 4, 4MB] [1 of 1]
 4973645 of 4973645   100% in    2s  1823.51 kB/s  done
```

Using the Java SDK

The Java SDK provides two ways to run large object uploads:

- Multipart Uploads
- TransferManager

Using the Python SDK

The Python SDK provides two ways to run large object uploads:

- Multipart Uploads
- TransferManager

Using the Node.js SDK

The Node.js SDK provides a single way to run large object uploads:

- Multipart Uploads

Tracking replication events

Replication allows you to define rules for automatic, asynchronous copying of objects from a source bucket to a target bucket in the same account. Also, you can copy objects from a bucket to another bucket in different accounts.

What is replication?

Replication copies newly created objects and object updates from a source bucket to a target bucket.

- Only new objects or new versions of the existing objects (created after the replication rule was added to the bucket) are copied to the target bucket. Existing objects can be replicated by copying them onto themselves, creating a new version that is replicated.
- The metadata of the source object is applied to the replicated object.
- Bi-directional replication between two buckets requires rules to be active on both buckets.
- Filters (composed of prefixes and/or tags) can be used to scope the replication rule to only apply to a subset of objects. Multiple rules can be defined in a single policy and these rules can specify different destinations. In this manner, different objects in the same bucket can be replicated to different destinations.

Why use replication?

- Keep a copy of data in a bucket in a different geographic location.
- Meet compliance regulations for data sovereignty by defining replication rules that store replicas only within the allowable locations.
- Keep production and test data in sync, as replication retains object metadata such as last modified time, version ID, and so on.
- Manage the storage class and lifecycle policies for the replicated objects independent of the source, by defining a different storage class and/or lifecycle rules for the target bucket. Similarly, you can store replicas in a bucket in a separate service instance or even IBM Cloud account, and also independently control access to the replicas.

Getting started with replication

To get started, here are some prerequisites that must be met:

- Set the the `Writer` or `Manager` platform role on the source bucket, or a custom role with the appropriate replication actions (such as `cloud-object-storage.bucket.put_replication`) assigned.
- You do not need to have access to the target bucket, but do need to have sufficient platform roles to create new IAM policies that allow the source bucket to write to the target bucket.
- The target bucket must not have a legacy bucket firewall enabled, but can use context-based restrictions.
- Objects encrypted using SSE-C cannot be replicated, although managed encryption (SSE-KMS) like Key Protect is fully compatible with replication.
- Objects in an archived state cannot be replicated.
- If the source and target buckets are in different IBM accounts, be sure to create the buckets in each account.
- Enable Versioning on both the source and target buckets.

Note: As versioning is a requirement for replication, it is impossible to replicate objects in buckets configured with an Immutable Object Storage policy.

Using one IBM account

To replicate objects between buckets in the same IBM account, do the following:

1. After navigating to your chosen source bucket, click the **Configuration** tab.
2. Look for **Bucket replication** and click the **Setup replication** button.
3. Select **Replication source** and click **Next**.
4. Select the instance and bucket from the drop-down menus. Alternatively, toggle the radio button to **No** and paste in the CRN of the target bucket.
5. Click on the **Check permissions** button.

Now, you'll need to grant the source bucket `Writer` permissions on the target bucket. There are several ways to do this, but the easiest is to use the IBM Cloud Shell and the IBM Cloud CLI.

1. Open an IBM Cloud Shell in a new window or tab.
2. Copy the IBM Cloud CLI command shown in the Object storage console, and paste it into the new shell.
3. Return to the bucket configuration window or tab, and click on the **Check permissions** button again.

Now you'll create a replication rule.

1. Ensure the rule status radio button is set to **Enabled**.
2. Give the rule a name and a priority, as well as any prefix or tag filters that will limit the objects subject to the replication rule.
3. Click **Done**.

Using different IBM accounts

To replicate objects between buckets in different IBM accounts, do the following:

1. Set up an IAM policy on the destination IBM account. For information about creating an IAM policy, see [What are IAM policies and who can assign them](#).
2. Find the account ID and the Service instance ID in CRN format on the Bucket Configuration page.
3. Using the IBM Cloud UI of the destination account, click **Manage**>**Access(IAM)**.
4. Click **Authentication** in the left panel.
5. Click **Create** to create a new IAM policy.
6. Grant a service authorization page configuration. This is the page where you will land after creating a new IAM policy.
7. Select **Another account** and provide the Account ID of the source account.
8. Provide service access as **Cloud Object Storage**.
9. In the Scope of Access, select **Specific Resources**.
10. Select **Source Service Instance** and enter the service instance ID for the source bucket.
11. Under Target, select **Cloud Object Storage** for the source bucket access.
12. For Target Scope, select **Specific resources**>**Service Instance.
13. Select the destination account's service instance ID from the drop down menu.
14. Select the role **Object Writer** or **Writer** as required.

    Note: The **Object writer** role is sufficient to enable replication.

Terminology

Source bucket: The bucket for which a replication policy is configured. It is the source of replicated objects.

Target bucket: The bucket that is defined as the destination in the source bucket replication policy. It is the target of replicated objects. Also referred to as a 'destination' bucket.

Replica: The new object created in a target bucket because of a request made to a source bucket.

What is replicated?

New objects created via `CopyObject`, `PutObject`, or `CompleteMultipartUpload` will be replicated from the source bucket to the target bucket. The replicated objects will inherit the following metadata fields from the source object: `Etag`, `Last Modified Time`, `Version ID`, `user-attributes`, and `Tags`.

Delete markers will be replicated if configured by the replication policy.

Updates to a version's tags will be replicated from the source bucket to the target bucket.

The following are not replicated:

- Actions initiated by lifecycle events
- Objects written directly to archive
- Objects restored from an archive tier
- Objects encrypted via SSE-C
- Object ACLs
- Object-Lock state

Using replication for business continuity and disaster recovery

Replication can be used to provide continuity of service in the event of an outage:

- Ensure that the source and target buckets are in different locations.
- Verify that the latest versions of objects are in sync between both buckets. A tool such as Rclone (the `rclone check` command) can be useful for checking synchronicity from the command line.
- In the event of an outage, an application's traffic can be redirected to the target bucket.

Consistency and data integrity

While IBM Cloud Object Storage provides strong consistency for all data IO operations, bucket configuration is eventually consistent. After enabling replication rules for the first time on a bucket, it may take a few moments for the configuration to propagate across the system and new objects to start being replicated.

IAM actions

There are new IAM actions associated with replication.

IAM Action	Role
`cloud-object-storage.bucket.get_replication`	Manager, Writer, Reader
`cloud-object-storage.bucket.put_replication`	Manager, Writer
`cloud-object-storage.bucket.delete_replication`	Manager, Writer

Activity Tracker events

Replication generates additional events.

- `cloud-object-storage.bucket-replication.create`
- `cloud-object-storage.bucket-replication.read`
- `cloud-object-storage.bucket-replication.delete`
- `cloud-object-storage.object-replication.sync` (generated at the source)
- `cloud-object-storage.object-replication.create` (generated at the target)

For `cloud-object-storage.bucket-replication.create` events, the following fields provide extra information:

Field	Description
`requestData.replication.num_sync_remote_buckets`	The number of target buckets specified in the bucket replication rules.
`requestData.replication.failed_remote_sync`	The CRNs of the buckets that failed the replication check.

When replication is active, operations on objects may generate the following extra information:

Field	Description
`requestData.replication.replication_throttled`	Indicates if the replication of the object was delayed on the source due to a throttling mechanism.
`requestData.replication.destination_bucket_id`	The CRN of the target bucket.
`requestData.replication.sync_type`	The type of sync operation. A `content` sync indicates that the object data *and* any metadata was written to the target, a `metadata` sync indicates that only metadata was written to the target, and a `delete` sync indicates that only delete markers were written to the target.
`responseData.replication.source_bucket_id`	The CRN of the source bucket.
`responseData.replication.result`	Values can be `success`, `failure` (indicates a server error), `user` (indicates a user error).
`responseData.replication.message`	The HTTP response message (such as `OK`).

You can trace an object from when it is written to the source until it is written on the target. Search for the request ID associated with the object write and three events should appear:

- The original `PUT`.
- The sync request from the source.
- The `PUT` request on the target.

Any of these three missing indicates a failure.

Usage and accounting

All replicas are objects themselves, and contribute usage just like any other data. Successful replication results in billable `PUT`, `GET`, and `HEAD` requests, although any bandwidth consumed in the replication process is not billed.

Replication generates additional metrics for use with IBM Cloud Monitoring:

- `ibm_cos_bucket_replication_sync_requests_issued`
- `ibm_cos_bucket_replication_sync_requests_received`

Interactions

Versioning

Versioning is mandatory to enable replication. After you enable versioning on both the source and target buckets and configure replication on the source bucket, you may encounter the following issues:

- If you attempt to disable versioning on the source bucket, Object Storage returns an error. You must remove the replication configuration before you can disable versioning on the source bucket.
- If you disable versioning on the target bucket, replication fails.

Key Protect encryption

Source objects will be encrypted using the root key of the source bucket, and replicas are encrypted using the root key of the target bucket.

Lifecycle configurations

If a lifecycle policy is enabled on a target bucket, the lifecycle actions will be based on the original creation time of the object at the source, not the time that the replica becomes available in the target bucket.

Immutable Object Storage

Using retention policies is impossible on a bucket with versioning enabled, and as versioning is a requirement for replication, it is impossible to replicate objects to or from a bucket with Immutable Object Storage enabled.

Legacy bucket firewalls

Buckets using [legacy firewalls to restrict access based on IP addresses](#) are not able to use replication, as the background services that replicate the objects do not have fixed IP addresses and can not pass the firewall.

It is recommended to instead [use context-based restrictions](#) for controlling access based on network information.

Cloud Functions and Code Engine

Configuring replication does not provide a [trigger for Cloud Functions](#) or Code Engine events at this time, but object writes and deletes will create `Object:Write` and `Object:Delete` notifications for both the source and target buckets. These events are annotated with a `notifications.replication_type` field that indicates if the event triggered a sync, or was triggered by a sync.

Replicating existing objects

A replication rule can only act on objects that are written *after* the rule is configured and applied to a bucket. If there are existing objects in a bucket that should be replicated, the replication processes needs to be made aware of the existence of the objects. This can be easily accomplished by using the `PUT copy` operation to copy objects onto themselves.

Important: This process will reset some object metadata, including creation timestamps. This will impact lifecycle policies and any other services that use creation or modification timestamps (such as content delivery networks). Ensure that any disruptions that may arise from resetting object metadata are dealt with appropriately.

The process involves:

1. Creating a list of all the objects in a bucket that should be subject to replication rules,
2. Iterating over that list, performing a `PUT copy` operation on each object with the source being identical to the target of the request.

Note: This example will only replicate the new version of the object created by the `PUT copy` request. In order to replicate all versions of the object, it would be necessary to copy each individual version as well.

The following example is written in Python, but the algorithm could be applied in any programming language or context.

```
$ import os
import sys
import ibm_boto3
from ibm_botocore.config import Config

# Create client connection
cos = ibm_boto3.client("s3",
                       ibm_api_key_id=os.environ.get('IBMCLOUD_API_KEY'),
                       ibm_service_instance_id=os.environ['SERVICE_INSTANCE_ID'],
                       config=Config(signature_version="oauth"),
                       endpoint_url=os.environ['US_GEO']
                       )

# Define the bucket with existing objects for replication
bucket = os.environ['BUCKET']

def copy_in_place(BUCKET_NAME):
    print("Priming existing objects in " + bucket + " for replication...")

    paginator = cos.get_paginator('list_objects_v2')
    pages = paginator.paginate(Bucket=bucket)

    for page in pages:
        for obj in page['Contents']:
            key = obj['Key']
            print("  * Copying " + key + " in place...")
            try:
                headers = cos.head_object(
                    Bucket=bucket,
                    Key=key
                    )
                md = headers["Metadata"]
                cos.copy_object(
                    CopySource={
                        'Bucket': bucket,
                        'Key': key
                        },
                    Bucket=bucket,
                    Key=key,
                    TaggingDirective='COPY',
                    MetadataDirective='REPLACE',
                    Metadata=md
                    )
                print("    Success!")
            except Exception as e:
                print("    Unable to copy object: {0}".format(e))
    print("Existing objects in " + bucket + " are now subject to replication rules.")

copy_in_place(bucket)
```

REST API examples

The following examples are shown using cURL for ease of use. Environment variables are used to represent user specific elements such as `$BUCKET`, `$TOKEN`, and `$REGION`. Note that `$REGION` would also include any network type specifications, so sending a request to a bucket in `us-south` using the private network would require setting the variable to `private.us-south`.

Enable replication on a bucket

The replication configuration is provided as XML in the body of the request. New requests will overwrite any existing replication rules that are present on the bucket.

A replication configuration must include at least one rule, and can contain a maximum of 1,000. Each rule identifies a subset of objects to replicate by filtering the objects in the source bucket. To choose additional subsets of objects to replicate, add a rule for each subset.

To specify a subset of the objects in the source bucket to apply a replication rule to, add the `Filter` element as a child of the `Rule` element. You can filter objects based on an object key prefix, one or more object tags, or both. When you add the `Filter` element in the configuration, you must also add the following elements: `DeleteMarkerReplication`, `Status`, and `Priority`.

Header	Type	Description
`Content-MD5`	String	**Required**: The base64 encoded 128-bit MD5 hash of the payload, which is used as an integrity check to ensure that the payload wasn't altered in transit.

The body of the request must contain an XML block with the following schema:

Element	Type	Children	Ancestor	Constraint
`ReplicationConfiguration`	Container	`Rule`	None	Limit 1.
`Rule`	Container	`ID`, `Status`, `Filter`, `DeleteMarkerReplication`, `Destination`, `Priority`	`ReplicationConfiguration`	Limit 1000.
`ID`	String	None	`Rule`	Must consist of (`a-z,A-Z0-9`) and the following symbols: `!` `_` `.` `*` `'` `(` `)` `-`
`Destination`	Container	`Bucket`	`Rule`	Limit 1.
`Bucket`	String	None	`Destination`	The CRN of the target bucket.

`Priority`	Integer	None	`Rule`	A priority is associated with each rule. There may be cases where multiple rules may be applicable to an object that is uploaded. In these situations, object storage will apply the applicable rule with the higher priority when replicating that object. Thus only a single replication rule can be applied to any object, irrespective of how many rules in the replication policy may be a match for the object. Note that the higher the number, the higher the priority.
`Status`	String	None	`Rule`	Specifies whether the rule is enabled. Valid values are `Enabled` or `Disabled`.
`DeleteMarkerReplication`	Container	`Status`	`Rule`	Limit 1.
`Status`	String	None	`DeleteMarkerReplication`	Specifies whether Object storage replicates delete markers. Valid values are `Enabled` or `Disabled`.
`Filter`	String	`Prefix`, `Tag`, `AND`	`Rule`	A filter that identifies the subset of objects to which the replication rule applies. A `Filter` must specify exactly one `Prefix`, `Tag`, or an `And` child element.
`Prefix`	String	None	`Filter`	An object key name prefix that identifies the subset of objects to which the rule applies.
`Tag`	String	None	`Filter`	A container for specifying a tag key and value. The rule applies only to objects that have the tag in their tag set.
`And`	String	None	`Filter`	A container for specifying rule filters. The filters determine the subset of objects to which the rule applies. This element is required only if you specify more than one filter.
`Key`	String	None	`Tag`	The tag key.
`Value`	String	None	`Tag`	The tag value.

This example will replicate any new objects, but will not replicate delete markers.

```
$ curl -X "PUT" "https://$BUCKET.s3.$REGION.cloud-object-storage.appdomain.cloud/?replication" \
     -H 'Authorization: bearer $TOKEN' \
     -H 'Content-MD5: exuBoz2kFBykNwqu64JZuA==' \
     -H 'Content-Type: text/plain; charset=utf-8' \
     -d $'<ReplicationConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
            <Rule>
              <ID>SimpleReplication</ID>
              <Priority>1</Priority>
              <Status>Enabled</Status>
              <DeleteMarkerReplication>
                <Status>Disabled</Status>
              </DeleteMarkerReplication>
              <Filter/>
              <Destination>
```

```
                <Bucket>$DESTINATION_CRN</Bucket>
              </Destination>
           </Rule>
          </ReplicationConfiguration>'
```

This example will replicate any objects with a key (name) that begin with `project_a/` to the bucket identified with `$DESTINATION_CRN_A`, and any objects with a key (name) that begin with `project_b/` to the bucket identified with `$DESTINATION_CRN_B`, and any objects that have an object tag with the key `Client` and the value `ACME` to a third bucket identified with `$DESTINATION_CRN_C`, and will replicate delete markers in all cases.

Assume that the following four objects are added to the source bucket. They will be replicated to target buckets as described below:

1. `project_a/foo.mp4`
2. `project_a/bar.mp4`
3. `project_b/baz.pdf`
4. `project_b/acme.pdf`. This fourth object also has an object tag with the key `Client` and the value `ACME`.

Because of the following rules, objects 1 and 2 will be replicated to `$DESTINATION_CRN_A`. Object 3 will be replicated to `$DESTINATION_CRN_B`. Object 4 will only be replicated to `$DESTINATION_CRN_C` because the rule with the ID `AcmeCorp` has a higher priority value than the rule with the ID `ProjectB` and while it meets the requirements for both rules, will only be subject to the former.

```
$ curl -X "PUT" "https://$BUCKET.s3.$REGION.cloud-object-storage.appdomain.cloud/?replication" \
     -H 'Authorization: bearer $TOKEN' \
     -H 'Content-MD5: exuBoz2kFBykNwqu64JZuA==' \
     -H 'Content-Type: text/plain; charset=utf-8' \
     -d $'<ReplicationConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
            <Rule>
              <ID>ProjectA</ID>
              <Priority>10</Priority>
              <Status>Enabled</Status>
              <DeleteMarkerReplication>
                <Status>Enabled</Status>
              </DeleteMarkerReplication>
              <Filter>
                <Prefix>project_a/</prefix>
              </Filter>
              <Destination>
                <Bucket>$DESTINATION_CRN_A</Bucket>
              </Destination>
           </Rule>
            <Rule>
              <ID>ProjectB</ID>
              <Priority>5</Priority>
              <Status>Enabled</Status>
              <DeleteMarkerReplication>
                <Status>Enabled</Status>
              </DeleteMarkerReplication>
              <Filter>
                <Prefix>project_b/</prefix>
              </Filter>
              <Destination>
                <Bucket>$DESTINATION_CRN_B</Bucket>
              </Destination>
           </Rule>
            <Rule>
              <ID>AcmeCorp</ID>
              <Priority>20</Priority>
              <Status>Enabled</Status>
              <DeleteMarkerReplication>
                <Status>Enabled</Status>
              </DeleteMarkerReplication>
              <Filter>
                <Tag>
                  <Key>Client</Key>
                  <Value>ACME</Value>
                </Tag>
              </Filter>
              <Destination>
                <Bucket>$DESTINATION_CRN_C</Bucket>
              </Destination>
           </Rule>
          </ReplicationConfiguration>'
```

A successful request returns a `200` response.

View replication configuration for a bucket

```
$ curl -X "GET" "https://$BUCKET.s3.$REGION.cloud-object-storage.appdomain.cloud/?replication" \
     -H 'Authorization: bearer $TOKEN'
```

This returns an XML response body with the appropriate schema:

```
$ <ReplicationConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Rule>
    <ID>SimpleReplication</ID>
    <Status>ENABLED</Status>
    <DeleteMarkerReplication>
      <Status>DISABLED</Status>
    </DeleteMarkerReplication>
    <Destination>
      <Bucket>crn:v1:bluemix:public:cloud-object-storage:global:a/9978e07eXXXXXXXX66c89c428028654:ef1c725e-XXXX-4967-bcc1-
734c03a2b846:bucket:replication-destination</Bucket>
    </Destination>
    <Priority>1</Priority>
    <Filter/>
  </Rule>
</ReplicationConfiguration>
```

Delete the replication configuration for a bucket

```
$ curl -X "DELETE" "https://$BUCKET.s3.$REGION.cloud-object-storage.appdomain.cloud/?replication" \
     -H 'Authorization: bearer $TOKEN'
```

A successful request returns a `204` response.

SDK examples

The following examples make use of the IBM COS SDKs for Python and Node.js, although the implementation of object versioning should be fully compatible with any S3-compatible library or tool that allows for the setting of custom endpoints. Using third-party tools requires HMAC credentials to calculate AWS V4 signatures. For more information on HMAC credentials, see the documentation.

Python

Enabling versioning using the IBM COS SDK for Python can be done using the low-level client syntax.

Using a client:

```
$ #!/usr/bin/env python3

import ibm_boto3
from ibm_botocore.config import Config
from ibm_botocore.exceptions import ClientError

# Define constants
API_KEY = os.environ.get('IBMCLOUD_API_KEY')
SERVICE_INSTANCE = os.environ.get('SERVICE_INSTANCE_ID')
ENDPOINT = os.environ.get('ENDPOINT')

BUCKET = "my-replication-bucket" # The bucket that will enable replication.

# Create resource client with configuration info pulled from environment variables.
cosClient = ibm_boto3.client("s3",
                         ibm_api_key_id=API_KEY,
                         ibm_service_instance_id=SERVICE_INSTANCE,
                         config=Config(signature_version="oauth"),
                         endpoint_url=ENDPOINT
                         )
```

```
response = cosClient.put_bucket_versioning(
    Bucket=BUCKET,
    ReplicationConfiguration={
        'Rules': [
            {
                'ID': 'string',
                'Priority': 123,
                'Filter': {
                    'Prefix': 'string',
                    'Tag': {
                        'Key': 'string',
                        'Value': 'string'
                    },
                    'And': {
                        'Prefix': 'string',
                        'Tags': [
                            {
                                'Key': 'string',
                                'Value': 'string'
                            },
                        ]
                    }
                },
                'Status': 'Enabled'|'Disabled',
                'Destination': {
                    'Bucket': 'string',
                },
                'DeleteMarkerReplication': {
                    'Status': 'Enabled'|'Disabled'
                }
            },
        ]
    }
)
```

Listing the versions of an object using the same client:

```
$ resp = cosClient.list_object_versions(Prefix='some-prefix', Bucket=BUCKET)
```

Note that the Python APIs are very flexible, and there are many different ways to accomplish the same task.

Node.js

Enabling versioning using the IBM COS SDK for Node.js:

```
$ const IBM = require('ibm-cos-sdk');

var config = {
    endpoint: '<endpoint>',
    apiKeyId: '<api-key>',
    serviceInstanceId: '<resource-instance-id>',
};

var cos = new IBM.S3(config);

var params = {
  Bucket: 'STRING_VALUE', /* required */
  ReplicationConfiguration: { /* required */
    Role: 'STRING_VALUE', /* required */
    Rules: [ /* required */
      {
        Destination: { /* required */
          Bucket: 'STRING_VALUE', /* required */
        },
        Status: Enabled | Disabled, /* required */
        Filter: {
          And: {
            Prefix: 'STRING_VALUE',
            Tags: [
              {
                Key: 'STRING_VALUE', /* required */
                Value: 'STRING_VALUE' /* required */
              },
              /* more items */
            ]
          },
          Prefix: 'STRING_VALUE',
          Tag: {
            Key: 'STRING_VALUE', /* required */
            Value: 'STRING_VALUE' /* required */
          }
        },
        ID: 'STRING_VALUE',
        Prefix: 'STRING_VALUE',
        Priority: 'NUMBER_VALUE',
        }
      }
    ]
  },
  ContentMD5: 'STRING_VALUE',
};
cos.putBucketReplication(params, function(err, data) {
  if (err) console.log(err, err.stack); // an error occurred
  else     console.log(data);           // successful response
});
```

Tagging objects

Your data can be expressly defined, categorized, and classified in IBM Cloud® Object Storage using associated metadata, called "tags." This document will show you how to take full control in "tagging" the objects representing your data.

Objects and metadata

Organizing your data can be a complex task. Basic methods, such as using key prefixes like organizational "folders" are a great start to hierarchical structures. But for more complex organization, you will need custom " *tags*." Your metadata can describe the relationships inherent to your data, and provide more organization than titles or folders. Unlike mere labels, there are two parts to a tag: a `key` and a `value`, defined individually according to your needs.

Tagging Objects

Managing tags describing your objects can be performed through various interfaces and architectures. Using the Console provides a graphical user interface. Using the command line requires tools like `curl` and the knowledge of how it interacts with Object Storage.

Before you begin

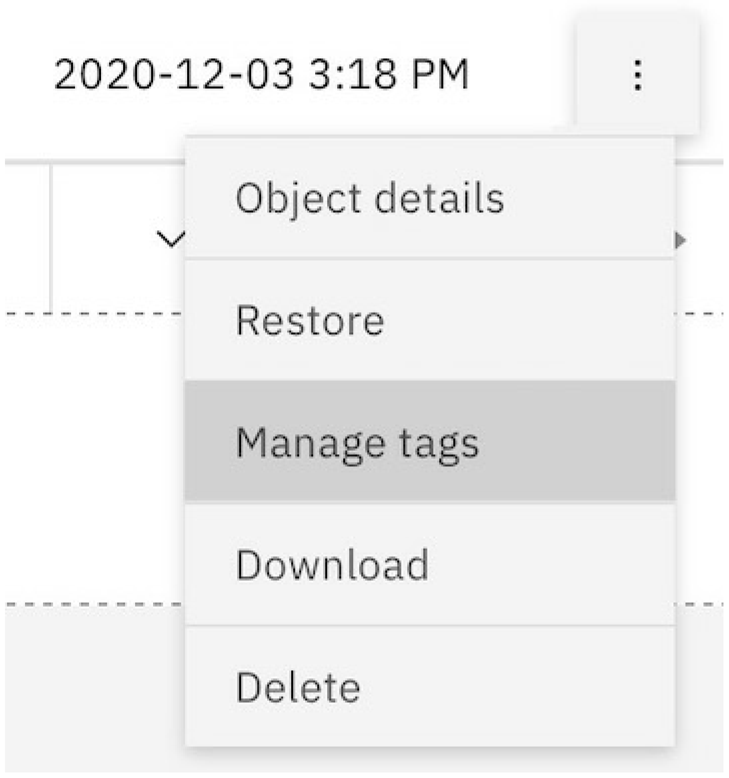
You need:

- An IBM Cloud® Platform account
- An instance of IBM Cloud Object Storage and a bucket created for this purpose
- An IAM API key with Writer access to your Object Storage bucket or instance
- Either existing or new objects that will have tags applied to them.

Reading tags

Tags are accessible throughout an instance with the proper permissions. While the true organizational power of using tags as an organizational principle scales with you, you can access tags on an individual basis as well.

Log in to the console, selecting your instance of IBM Cloud Object Storage and your bucket where your data is represented. After you've uploaded files to your bucket, you can view and manage your tags right in place. Place the cursor over the ellipses at the end of any row representing your data (stored as an object), and select "Manage your tags" from the options in the menu.



A properly formed and authenticated "GET" request with the `?tagging` query parameter is all that is required for accessing the tags for your objects using `curl`. The examples here use bearer tokens generated using this example. In addition to the bucket identifier and object key, you will also need the correct endpoint. The resulting XML object is also shown, where the "Tag" element will be repeated for each tag assigned to the object. If there are no tags, the response will return XML with an empty element, `<TagSet />`.

```
$ curl 'https://<endpoint>/<bucketname>/<objectname>?tagging' \
-H 'Authorization: bearer <token>' \
```

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Tagging xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <TagSet>
    <Tag>
      <Key>Example Key</Key>
      <Value>Value Example</Value>
    </Tag>
  </TagSet>
</Tagging>
```

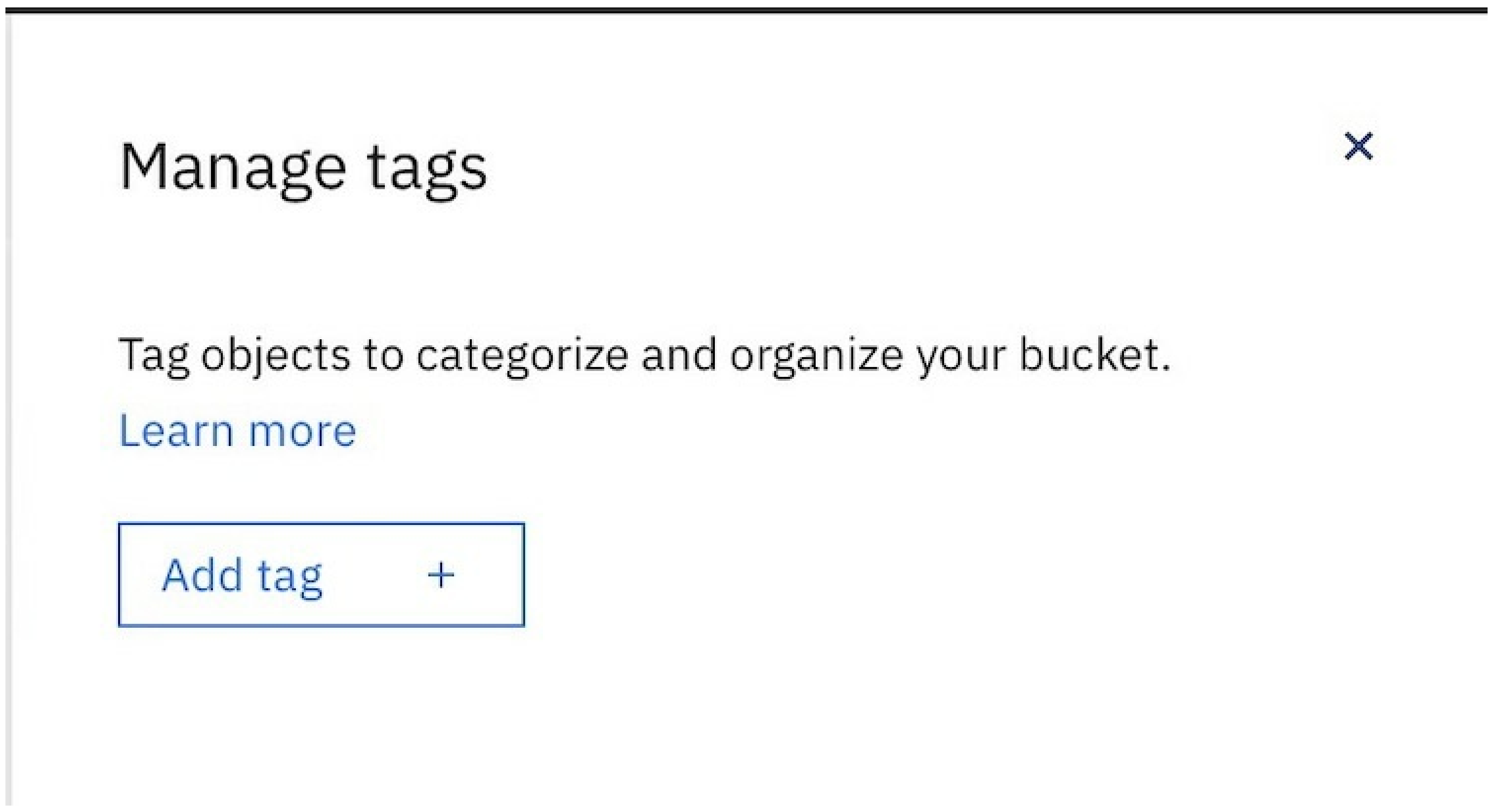
Of course, before tags can be viewed they must be created, which we will turn to next.

Creating tags

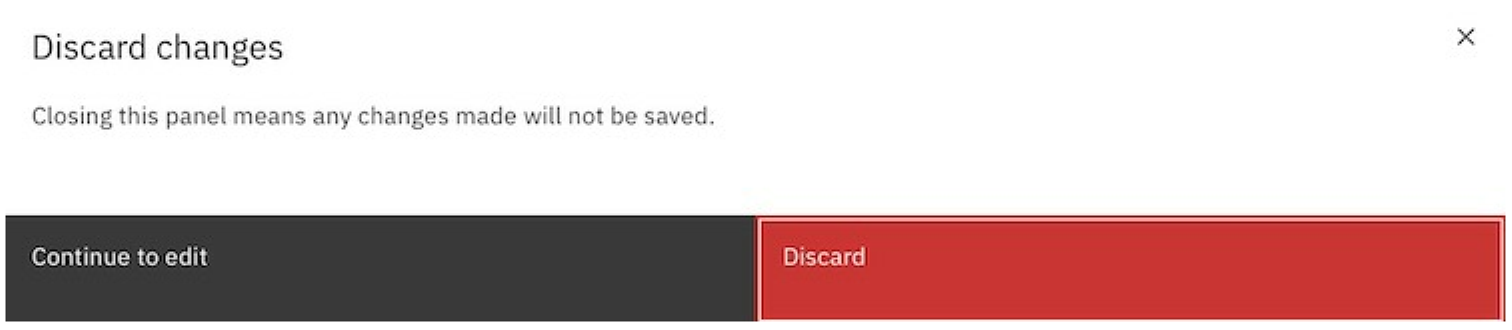
Tags must comply with the following restrictions:

- An object can have a maximum of 10 tags
- For each object, each tag key must be unique, and each tag key can have only one value.
- Minimum key length - 1 Unicode characters in UTF-8
- Maximum key length - 128 Unicode characters in UTF-8
- Maximum key byte size - 256 bytes
- Minimum value length - 0 Unicode characters in UTF-8 (Tag Value can be empty)
- Maximum value length - 256 Unicode characters in UTF-8
- Maximum value byte size - 512 bytes
- A Tag key and value may consist of US Alpha Numeric Characters (`a-z`, `A-Z`, `0-9`), and spaces representable in UTF-8, and the following symbols: `_`, `.`, `*`, `'`, `-`, `:`
- Tag keys and values are case-sensitive
- `ibm:` cannot be used as a key prefix for tags

As noted previously, log in to your instance and navigate to the bucket and object you wish to "tag." In the panel that appears when you select "Manage your tags", start by clicking on the "Add tags +" button. Then, you can add tags by typing text into the `key` and `value` fields as desired. Add more tags one at a time, by repeating the steps you've just completed.



If you do not click on "save" when completing your changes, a dialog box will remind you of the consequences. That is, changes are discarded unless saved.


Discard changes

Closing this panel means any changes made will not be saved.

As noted previously, you will have to authenticate to add tags to your data. If you have questions about bearer tokens, see [this example](). Again, note the query string for working with tags: `?tagging`.

```
$ curl -X "PUT" 'https://<endpoint>/<bucketname>/<objectname>?tagging' \
-H 'Authorization: bearer <token>' \
-H "content-type: text/plain" \
--data "<Tagging><TagSet><Tag><Key>your key</Key><Value>your text</Value></Tag></TagSet></Tagging>"
```

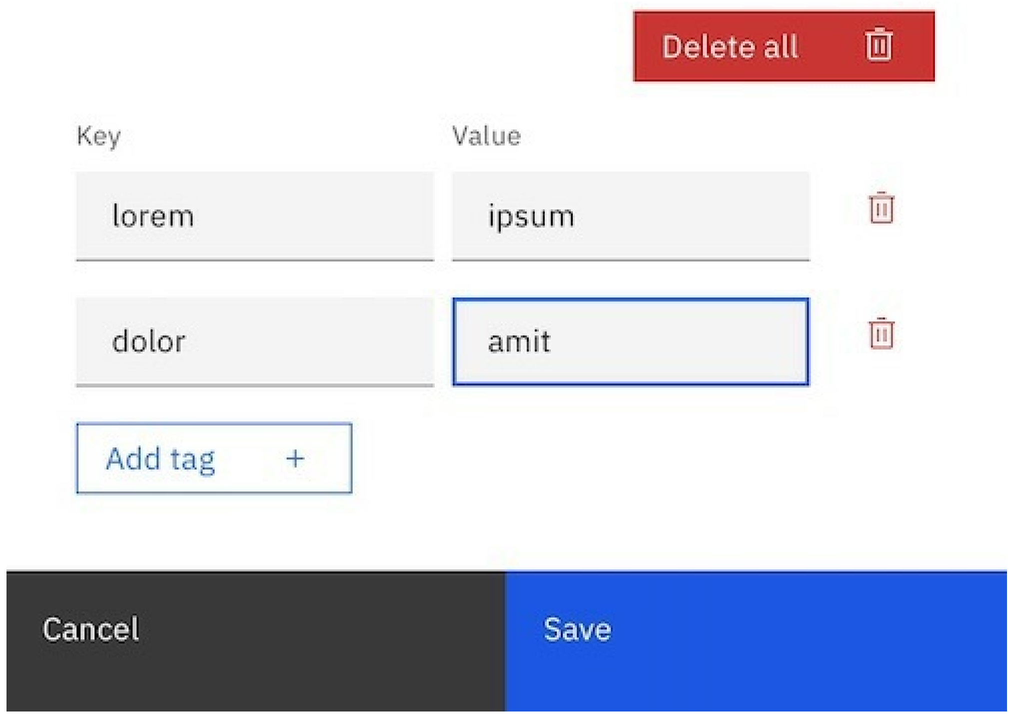
The example describes as shown a tag with a `key` of 'source' and a `value` of 'text' in the XML sent as data in the body of the request. The [schema]() of the XML has to validate upon execution. If you want to add multiple tags, duplicate the 'Tag' node and modify the content of each key and value element to your specifications. There is only one 'TagSet' element for each object, and the 'PUT' command will replace any existing metadata with the values you specified.

```
curl -X "PUT" "https://s3.test.cloud-object-storage.sample.appdomain.cloud/taggingtest/example-file.csv?tagging" -H
"Authorization: bearer ...iOiIyMDIwMTIwNzE0NDkiLCJh..." -H "ibm-service-instance-id: 7nnnnn52-2nn0-nna9-bann-7nnnnn4cc4e7" --data
"<Tagging><TagSet><Tag><Key>source</Key><Value>text</Value></Tag></TagSet></Tagging>"
```

Editing tags

Once your objects have been tagged, over time it may become necessary to modify them.

To edit the tags using the graphic interface, you will have to log into the console and access your objects as described previously. Once you've clicked on the "Manage Tags" option, simply change the contents of the form fields. Remember to press "Save" when complete.



Your requests must be authenticated to tag your data. Also, you will have to programmatically keep any old tags while updating your objects with new information. The example shown repeats the tags from the previous examples while adding a new tag.

⚠ **Important:** Remember that performing "PUT" operations involving tags will overwrite any current tags.

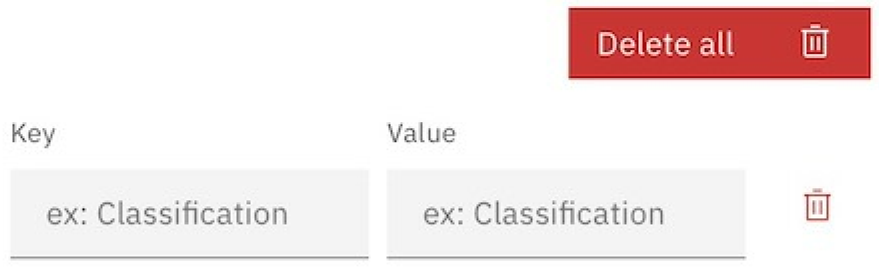
```
$ curl -X "PUT" 'https://<endpoint>/<bucketname>/<objectname>?tagging' \
-H 'Authorization: bearer <token>' \
-H "content-type: text/plain" \
--data "<Tagging><TagSet><Tag><Key>source</Key><Value>text</Value></Tag><Tag><Key>source1</Key><Value>text1</Value></Tag>
</TagSet></Tagging>"
```

Removing tags

After you have added tags to your objects, it may become necessary to remove them.

To delete the tags using the graphic interface, you will have to log into the console and access your objects as previously described. Again, click on the "Manage tags" option, and in the panel that appears, choose either to "delete all" or delete one tag at a time by clicking on the "trash can" icon in the same row as the tag.

Remember to press "Save" when complete.



You will have to authenticate to delete tags from your data. Simply use the "DELETE" HTTP method with the `?tagging` query parameter to delete all tags. If you wish to delete one or more tags while simultaneously keeping one or more tags, use the "edit" instructions to make your changes.

```
$ curl -X "DELETE" 'https://<endpoint>/<bucketname>/<objectname>?tagging' \
-H 'Authorization: bearer <token>' \
-H "content-type: text/plain"
```

Next Steps

Find more details about the operations related to objects in the S3 API documentation and more configuration options in the configuration API.

Versioning objects

Versioning allows multiple revisions of a single object to exist in the same bucket. Each version of an object can be queried, read, restored from an archived state, or deleted. Enabling versioning on a bucket can mitigate data loss from user error or inadvertent deletion. When an object is overwritten, a new version is created, and the previous version of the object is automatically preserved. Therefore, in a versioning-enabled bucket, objects that are deleted as a result of accidental deletion or overwrite can easily be recovered by restoring a previous version of the object. If an object is deleted, it is replaced by a *delete marker* and the previous version is saved (nothing is permanently deleted). To permanently delete individual versions of an object, a delete request must specify a *version ID*. A `GET` request for an object will retrieve the most recently stored version. If the current version is a delete marker, IBM COS returns a `404 Not Found` error.

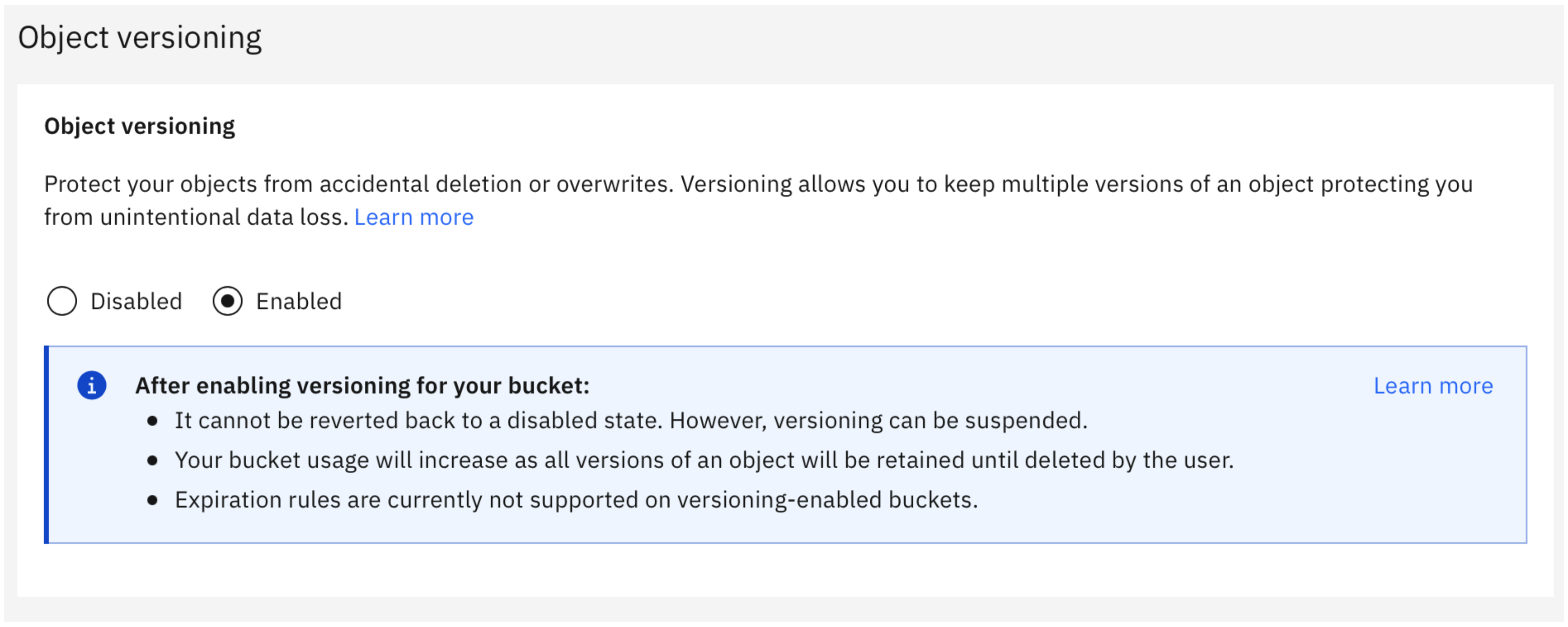
After a bucket has enabled versioning, all the objects in the bucket are versioned. All new objects (created after enabling versioning on a bucket) will receive a permanently assigned version ID. Objects created before versioning was enabled (on the bucket) are assigned a version of `null`. When an object with a `null` version ID is overwritten or deleted it is assigned a new version ID. Suspending versioning does not alter any existing objects, but will change the way future requests are handled by IBM COS. Once enabled, versioning can only be suspended, and not fully disabled. Therefore, a bucket can have three states related to versioning: 1. Default (unversioned), 2. Enabled, or 3. Suspended.

Getting started with versioning

First, create a new bucket with object versioning enabled.

1. After navigating to your object storage instance, click on **Create bucket**.
2. Choose a region and resiliency, then look for **Object versioning** and toggle the selector to **Enabled**.

Enable versioning



Object versioning

Object versioning

Protect your objects from accidental deletion or overwrites. Versioning allows you to keep multiple versions of an object protecting you from unintentional data loss. Learn more

Disabled [✓ Enabled]

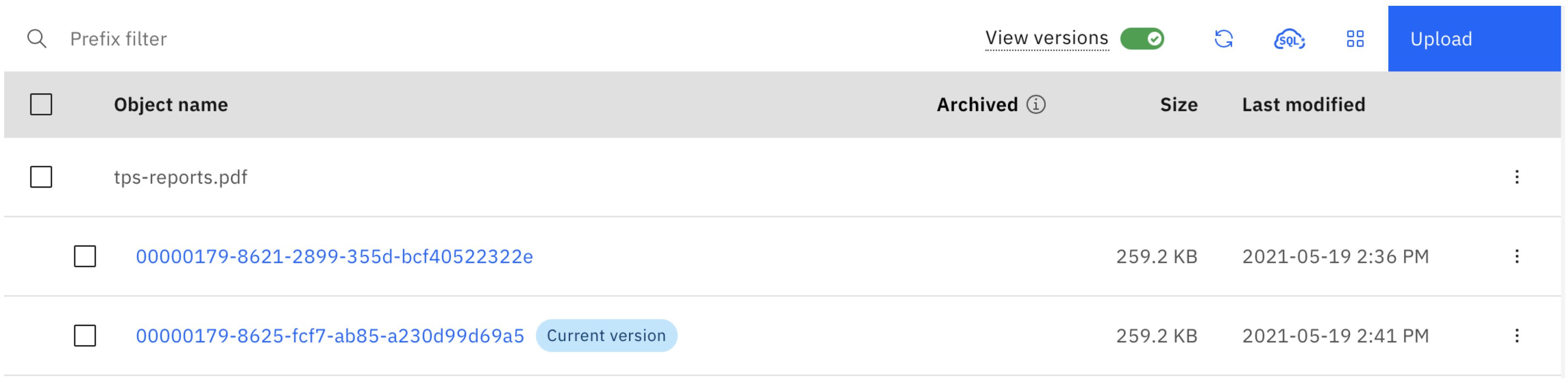
After enabling versioning for your bucket:
- It cannot be reverted back to a disabled state. However, versioning can be suspended.
- Your bucket usage will increase as all versions of an object will be retained until deleted by the user.
- Expiration rules are currently not supported on versioning-enabled buckets.

Learn more

Then create a versioned object.

1. Navigate your new bucket, and upload a file by dragging it onto the browser window.
2. After the object has uploaded successfully, upload another object with the same name. Instead of being overwritten, the file will be assigned a UUID and saved as a non-current version of the object.
3. Toggle **View versions** to see and interact with alternate versions of objects.

View versions



Object name	Archived	Size	Last modified
tps-reports.pdf			
00000179-8621-2899-355d-bcf40522322e		259.2 KB	2021-05-19 2:36 PM
00000179-8625-fcf7-ab85-a230d99d69a5 Current version		259.2 KB	2021-05-19 2:41 PM

Terminology

Delete marker: An 'invisible' object that allows for accessing versions of the deleted object.

Version ID: A Unicode, UTF-8 encoded, URL-safe, opaque string that indicates a unique version of an object and associated metadata, and is used to target requests to that particular version. Version IDs are a maximum of 1,024 bytes long.

'null': A special version ID assigned to objects that existed when versioning was enabled on a bucket.

Consistency and data integrity

While IBM COS provides strong consistency for all data IO operations, bucket configuration is eventually consistent. After enabling versioning for the first time on a bucket, it may take a few moments for the configuration to propagate across the system. Although versioning may appear to be enabled, it is recommended to wait 15 minutes after enabling versioning to make any requests that are expected to create versions or delete markers.

IAM actions

There are new IAM actions associated with versioning.

IAM Action	Role
cloud-object-storage.bucket.put_versioning	Manager, Writer
cloud-object-storage.bucket.get_versioning	Manager, Writer, Reader
cloud-object-storage.object.get_version	Manager, Writer, Reader, Content Reader, Object Reader
cloud-object-storage.object.head_version	Manager, Writer, Reader, Content Reader, Object Reader
cloud-object-storage.bucket.delete_version	Manager, Writer
cloud-object-storage.object.get_versions	Manager, Writer, Reader, Content Reader, Object Reader
cloud-object-storage.object.copy_get_version	Manager, Writer, Reader
cloud-object-storage.object.copy_part_get_version	Manager, Writer, Reader
cloud-object-storage.object.restore_version	Manager, Writer
cloud-object-storage.object.put_tagging_version	Manager, Writer, Object Writer
cloud-object-storage.object.get_tagging_version	Manager, Writer, Reader
cloud-object-storage.object.delete_tagging_version	Manager, Writer

IAM actions associated with versioning

Activity Tracker events

Versioning will generate new events.

- `cloud-object-storage.bucket-versioning.create`
- `cloud-object-storage.bucket-versioning.read`
- `cloud-object-storage.bucket-versioning.list`

Management events for versioned buckets contain a `requestData.versioning.state` field, indicating whether versioning is enabled or suspended on a bucket.

The basic `HEAD`, `GET`, `PUT`, and `DELETE` actions that act on or create versions of objects will include a `target.versionId` field. The `target.versionId` field is also present when completing a multipart upload and when copying objects or parts, if a new version is created because of those actions.

A `responseData.deleteMarker.created` field is present when an object is deleted and a delete marker is created.

Usage and accounting

All versions are metered as if they were equal objects. This means that if a bucket contains a single object with five previous versions, the `object_count` field returned by the Resource Configuration API will be `6`, even though it will appear as if there is only a single object in the bucket. Likewise, accumulated versions contribute to total usage and are billable. In addition to the `object_count` field returned by the Read Bucket Metadata API, the API response body contains several new fields associated with versioning:

- `noncurrent_object_count`: Number of non-current object versions in the bucket in `int64` format.
- `noncurrent_bytes_used`: Total size of all non-current object versions in the bucket in `int64` format.
- `delete_marker_count`: Total number of delete markers in the bucket in `int64` format.

As mentioned, versioning can only be enabled or suspended. If for any reason there is a desire to completely disable versioning, then it is necessary to migrate the contents of the bucket to a new bucket that does not have versioning enabled.

Interactions

The IBM COS implementation of the S3 APIs for versioning is identical to the AWS S3 APIs for versioning, with a few differences.

Archiving and expiring versioned objects

Lifecycle configurations are permitted in a version-enabled bucket. However, unlike Amazon S3, new versions are subject to the archive rule in the same manner as regular objects. Objects are given a transition date when they are created, and are archived on their individual transition date, regardless of whether they are current or non-current versions. Overwriting an object does not affect the transition date of the previous version, and the new (current) version will be assigned a transition date.

It is not possible to use `NoncurrentVersionTransition` rules to archive *only* non-current versions of objects in a lifecycle configuration.

Immutable Object Storage (WORM)

The IBM COS implementation of Immutable Object Storage (that is, retention policies) is not permitted in buckets with versioning enabled. Attempts to create a retention policy will fail, as will attempts to enable versioning on a bucket with an retention policy.

Supported S3 APIs

The following set of REST APIs can interact with versioning in some way:

- `GET Object`
- `HEAD Object`
- `DELETE Object`
- `GET Object ACL`
- `PUT Object ACL`
- `Upload Part Copy`
- `Restore Object`
- `DELETE Objects`
- `List Object Versions`
- `PUT Bucket Versioning`
- `GET Bucket Versioning`
- `PUT Object`
- `POST Object`
- `Copy Object`
- `Complete Multipart Upload`
- `PUT Object Tagging`
- `GET Object Tagging`
- `DELETE Object Tagging`
- `PUT Bucket Lifecycle`
- `GET Bucket Lifecycle`
- `DELETE Bucket Lifecycle`

REST API examples

The following examples are shown using cURL for ease of use. Environment variables are used to represent user specific elements such as `$BUCKET`, `$TOKEN`, and `$REGION`. Note that `$REGION` would also include any network type specifications, so sending a request to a bucket in `us-south` using the private network would require setting the variable to `private.us-south`.

Enable versioning on a bucket

```
$ curl -X "PUT" "https://$BUCKET.s3.$REGION.cloud-object-storage.appdomain.cloud/?versioning" \
     -H 'Authorization: bearer $TOKEN' \
     -H 'Content-MD5: 8qj8HSeDu3APPMQZVG06WQ==' \
     -H 'Content-Type: text/plain; charset=utf-8' \
     -d $'<VersioningConfiguration>
            <Status>Enabled</Status>
          </VersioningConfiguration>'
```

A successful request returns a `200` response.

Suspend versioning on a bucket

```
$ curl -X "PUT" "https://$BUCKET.s3.$REGION.cloud-object-storage.appdomain.cloud/?versioning" \
     -H 'Authorization: bearer $TOKEN' \
     -H 'Content-MD5: hxXDWuCDWB72Be0LG4XniQ==' \
     -H 'Content-Type: text/plain; charset=utf-8' \
     -d $'<VersioningConfiguration>
            <Status>Suspended</Status>
          </VersioningConfiguration>'
```

A successful request returns a `200` response.

List versions of objects in a bucket

```
$ curl -X "GET" "https://$BUCKET.s3.$REGION.cloud-object-storage.appdomain.cloud/?versions" \
     -H 'Authorization: bearer $TOKEN'
```

This returns an XML response body:

```
$ <ListVersionsResult>
    <IsTruncated>boolean</IsTruncated>
    <KeyMarker>string</KeyMarker>
    <VersionIdMarker>string</VersionIdMarker>
    <NextKeyMarker>string</NextKeyMarker>
    <NextVersionIdMarker>string</NextVersionIdMarker>
    <Version>
        <ETag>string</ETag>
        <IsLatest>boolean</IsLatest>
        <Key>string</Key>
        <LastModified>timestamp</LastModified>
        <Owner>
            <DisplayName>string</DisplayName>
            <ID>string</ID>
        </Owner>
        <Size>integer</Size>
        <StorageClass>string</StorageClass>
        <VersionId>string</VersionId>
    </Version>
    ...
    <DeleteMarker>
        <IsLatest>boolean</IsLatest>
        <Key>string</Key>
        <LastModified>timestamp</LastModified>
        <Owner>
            <DisplayName>string</DisplayName>
            <ID>string</ID>
        </Owner>
        <VersionId>string</VersionId>
    </DeleteMarker>
    ...
    <Name>string</Name>
    <Prefix>string</Prefix>
    <Delimiter>string</Delimiter>
    <MaxKeys>integer</MaxKeys>
    <CommonPrefixes>
        <Prefix>string</Prefix>
    </CommonPrefixes>
    ...
    <EncodingType>string</EncodingType>
</ListVersionsResult>
```

`delimiter`: A delimiter is a character that you specify to group keys. All keys that contain the same string between the prefix and the first occurrence of the delimiter are grouped under a single result element in `CommonPrefixes`. These groups are counted as one result against the max-keys limitation. These keys are not returned elsewhere in the response.

`encoding-type`: Requests COS to url-encode the object keys in the response. Object keys may contain any Unicode character; however, XML 1.0 parser cannot parse some characters, such as characters with an ASCII value from 0 to 10. For characters that are not supported in XML 1.0, you can add this parameter to request that COS encodes the keys in the response. Valid value: `url`.

`key-marker`: Specifies the key to start with when listing objects in a bucket.

`max-keys`: Sets the maximum number of keys returned in the response. By default the API returns up to 1,000 key names. The response might contain fewer keys but will never contain more.

`prefix`: Use this parameter to select only those keys that begin with the specified prefix.

`version-id-marker`: Specifies the object version you want to start listing from.

Operations on specific versions of objects

Several APIs make use of a new query parameter (`?versionId=<VersionId>`) that indicates which version of the object you are requesting. This parameter is used in the same manner for reading, deleting, checking metadata and tags, and restoring archived objects. For example, to read a version of an object `foo` with a version ID of `L4kqtJlcpXroDVBH40Nr8X8gdRQBpUMLUo`, the request might look like the following:

```
$ curl -X "GET" "https://$BUCKET.s3.$REGION.cloud-object-storage.appdomain.cloud/foo?
versionId=L4kqtJlcpXroDVBH40Nr8X8gdRQBpUMLUo" \
     -H 'Authorization: bearer $TOKEN'
```

Deleting that object is done in the same manner.

```
$ curl -X "DELETE" "https://$BUCKET.s3.$REGION.cloud-object-storage.appdomain.cloud/foo?
versionId=L4kqtJlcpXroDVBH40Nr8X8gdRQBpUMLUo" \
     -H 'Authorization: bearer $TOKEN'
```

For requests that already make use of a query parameter, the `versionId` parameter can be added to the end.

```
$ curl -X "GET" "https://$BUCKET.s3.$REGION.cloud-object-storage.appdomain.cloud/foo?
tagging&versionId=L4kqtJlcpXroDVBH40Nr8X8gdRQBpUMLUo" \
     -H 'Authorization: bearer $TOKEN'
```

Server-side copying of object versions is supported, but uses a slightly different syntax. The query parameter is not added to the URL itself, but instead is appended to the `x-amz-copy-source` header. This is the same syntax as creating a part for a multipart part from a source object.

```
$ curl -X "PUT" "https://$BUCKET.s3.$REGION.cloud-object-storage.appdomain.cloud/<new-object-key>"
 -H "Authorization: bearer $TOKEN"
 -H "x-amz-copy-source: /<source-bucket>/<object-key>?versionId=L4kqtJlcpXroDVBH40Nr8X8gdRQBpUMLUo"
```

CLI examples

You can use the IBM Cloud CLI with the `cos` plug-in to enable versioning on a bucket.

```
$ cos bucket-versioning-put --bucket $BUCKET --versioning-configuration file://vers.json
```

In this case, `vers.json` is a simple document:

```
$ {
    "Status": "Enabled"
}
```

SDK examples

The following examples make use of the IBM COS SDKs for Python and Node.js, although the implementation of object versioning should be fully compatible with any S3-compatible library or tool that allows for the setting of custom endpoints. Using third-party tools requires HMAC credentials in order to calculate AWS V4 signatures. For more information on HMAC credentials, see the documentation.

Python

Enabling versioning using the IBM COS SDK for Python can be done using either the high-level resource or low-level client syntax.

Using a resource:

```
$ #!/usr/bin/env python3

import ibm_boto3
from ibm_botocore.config import Config
from ibm_botocore.exceptions import ClientError

#Define constants
API_KEY = os.environ.get('IBMCLOUD_API_KEY')
SERVICE_INSTANCE = os.environ.get('SERVICE_INSTANCE_ID')
ENDPOINT = os.environ.get('ENDPOINT')

BUCKET = "my-versioning-bucket" # The bucket that will enable versioning.

#Create resource client with configuration info pulled from environment variables.
cos = ibm_boto3.resource("s3",
                         ibm_api_key_id=API_KEY,
                         ibm_service_instance_id=SERVICE_INSTANCE,
                         config=Config(signature_version="oauth"),
                         endpoint_url=ENDPOINT
                         )

versioning = cos.BucketVersioning(BUCKET)

versioning.enable()
```

Versioning for the bucket can then be suspended using `versioning.suspend()`

Using that same `cos` resource, all versions of objects could be listed using the following:

```
$ versions = s3.Bucket(BUCKET).object_versions.filter(Prefix=key)

for version in versions:
    obj = version.get()
    print(obj.get('VersionId'), obj.get('ContentLength'), obj.get('LastModified'))
```

Using a client:

```
$ #!/usr/bin/env python3

import ibm_boto3
from ibm_botocore.config import Config
from ibm_botocore.exceptions import ClientError

#Define constants
API_KEY = os.environ.get('IBMCLOUD_API_KEY')
SERVICE_INSTANCE = os.environ.get('SERVICE_INSTANCE_ID')
ENDPOINT = os.environ.get('ENDPOINT')

BUCKET = "my-versioning-bucket" # The bucket that will enable versioning.

#Create resource client with configuration info pulled from environment variables.
cosClient = ibm_boto3.client("s3",
                         ibm_api_key_id=API_KEY,
                         ibm_service_instance_id=SERVICE_INSTANCE,
                         config=Config(signature_version="oauth"),
                         endpoint_url=ENDPOINT
                         )

response = cosClient.put_bucket_versioning(
    Bucket=BUCKET,
    VersioningConfiguration={
        'Status': 'Enabled'
    }
)
```

Listing the versions of an object using the same client:

```
$ resp = cosClient.list_object_versions(Prefix='some-prefix', Bucket=BUCKET)
```

Note that the Python APIs are very flexible, and there are many different ways to accomplish the same task.

Node.js

Enabling versioning using the IBM COS SDK for Node.js:

```
$ const IBM = require('ibm-cos-sdk');

var config = {
    endpoint: '<endpoint>',
    apiKeyId: '<api-key>',
    serviceInstanceId: '<resource-instance-id>',
};

var cos = new IBM.S3(config);

var params = {
    Bucket: 'my-versioning-bucket', /* required */
    VersioningConfiguration: { /* required */
    Status: 'Enabled'
   },
};

s3.putBucketVersioning(params, function(err, data) {
   if (err) console.log(err, err.stack); // an error occurred
   else     console.log(data);           // successful response
});
```

Integrated services

Managing encryption

Encrypting your data

IBM Cloud® Object Storage provides several options to encrypt your data.

By default, all objects that are stored in IBM Cloud Object Storage are encrypted by using randomly generated keys and an all-or-nothing-transform (AONT). While this default encryption model provides at-rest security, some workloads need full control over the data encryption keys used. You can manage your keys manually on a per-object basis by providing your own encryption keys - referred to as Server-Side Encryption with Customer-Provided Keys (SSE-C).

With Object Storage you also have a choice to use our integration capabilities with IBM Cloud® Key Management Services like IBM® Key Protect and Hyper Protect Crypto Services. Depending on the security requirements, you can decide whether to use IBM Key Protect or IBM Hyper Protect Crypto Services for your IBM Cloud Object Storage buckets.

IBM® Key Protect for IBM Cloud® helps you provision encrypted keys for apps across IBM Cloud® services. As you manage the lifecycle of your keys, you can benefit from knowing that your keys are secured by FIPS 140-2 Level 3 certified cloud-based hardware security modules (HSMs) that protect against the theft of information.

Hyper Protect Crypto Services is a single-tenant, dedicated HSM that is controlled by you. The service is built on FIPS 140-2 Level 4-certified hardware, the highest offered by any cloud provider in the industry.

Refer to product documentation on IBM® Key Protect for IBM Cloud® and Hyper Protect Crypto Services for a detailed overview of the two services.

Server-Side Encryption with Customer-Provided Keys (SSE-C)

SSE-C is enforced on objects. Requests to read or write objects or their metadata that use customer-managed keys send the required encryption information as headers in the HTTP requests. The syntax is the same as the S3 API, and S3-compatible libraries that support SSE-C work as expected against IBM Cloud® Object Storage.

Any request that uses SSE-C headers must be sent by using SSL. The `ETag` values in response headers are *not* the MD5 hash of the object, but a randomly generated 32-byte hexadecimal string.

A typical PUT object request can make use of the following headers:

Header	Type	Description
`x-amz-server-side-encryption-customer-algorithm`	String	This header is used to specify the algorithm and key size to use with the encryption key stored in `x-amz-server-side-encryption-customer-key` header. This value must be set to the string `AES256`.
`x-amz-server-side-encryption-customer-key`	String	This header is used to transport the base 64 encoded byte string representation of the AES 256 key used in the server-side encryption process.
`x-amz-server-side-encryption-customer-key-MD5`	String	This header is used to transport the base64-encoded 128-bit MD5 digest of the encryption key according to RFC 1321. The object store uses this value to validate the key passes in the `x-amz-server-side-encryption-customer-key` has not been corrupted during transport and encoding process. The digest must be calculated on the key BEFORE the key is base 64 encoded.

A cURL command might look like the following:

```
$ curl -v -T $FILE https://$ENDPOINT/$BUCKET  \
   -H "Authorization: bearer $TOKEN" \
   -H "Content-MD5: $MD5_OBJECT_HASH" \
   -H "x-amz-server-side-encryption-customer-algorithm: AES256" \
   -H "x-amz-server-side-encryption-customer-key:$ENCRYPTION_KEY" \
   -H "x-amz-server-side-encryption-customer-key-MD5:$MD5_KEY_HASH"
```

Server-Side Encryption with IBM Key Protect (SSE-KP)

You can use IBM Key Protect to create, add, and manage keys, which you can then associate with your instance of IBM® Cloud Object Storage to encrypt buckets.

Before you begin

Before you plan on using Key Protect with Cloud Object Storage buckets, you need:

- An IBM Cloud™ Platform account
- An instance of IBM Cloud Object Storage

You will also need to ensure that a service instance is created by using the IBM Cloud catalog and appropriate permissions are granted. This section outlines step-by-step instructions to help you get started.

Provisioning an instance of IBM Key Protect

Refer to the service-specific product pages for instructions on how to provision and setup appropriate service instances.

- Getting started with IBM Key Protect

Once you have an instance of Key Protect, you need to create a root key and note the CRN (Cloud Resource Name) of that key. The CRN is sent in a header during bucket creation.

Before creating the bucket for use with Key Protect, review the relevant guidance around availability and disaster recovery.

⚠ **Important:** Note that managed encryption for a Cross Region bucket **must** use a root key from a Key Protect instance in the nearest high-availability location (`us-south`, `eu-de`, or `jp-tok`).

Create or add a key in Key Protect

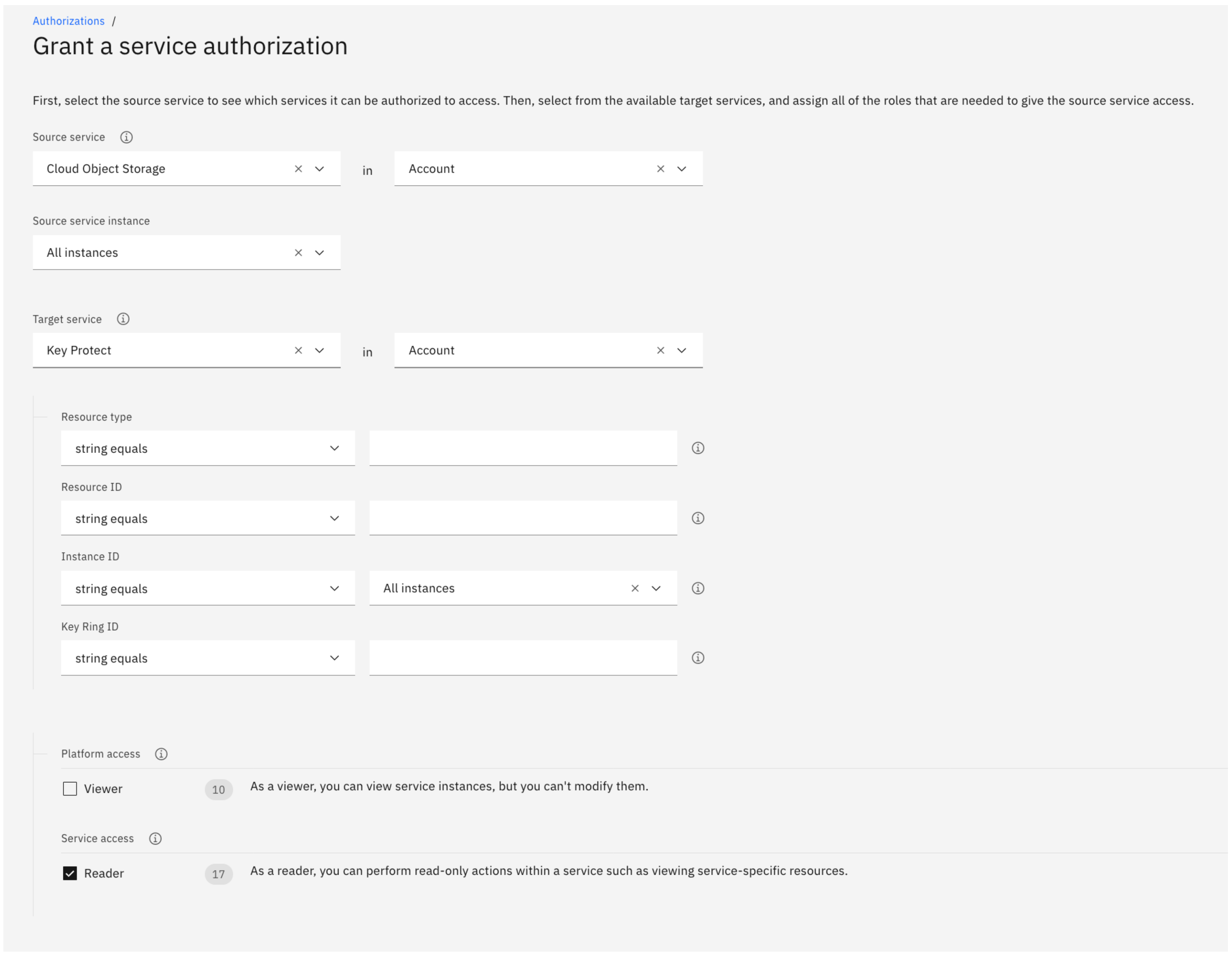
Navigate to your instance of Key Protect and generate or enter a root key.

Grant service authorization

Authorize Key Protect for use with IBM COS:

1. Open your IBM Cloud dashboard.
2. From the menu bar, click **Manage > Access (IAM)**.
3. In the side navigation, click **Authorizations**.
4. Click **Create authorization**.
5. In the **Source service** menu, select **Cloud Object Storage**.
6. In the **Source service instance** menu, select the service instance to authorize.
7. In the **Target service** menu, select **IBM Key Protect**.

Figure 1: Grant service authorization for Key Protect.


Authorizations /

Grant a service authorization

First, select the source service to see which services it can be authorized to access. Then, select from the available target services, and assign all of the roles that are needed to give the source service access.

Source service: Cloud Object Storage in Account

Source service instance: All instances

Target service: Key Protect in Account

Resource type: string equals

Resource ID: string equals

Instance ID: string equals All instances

Key Ring ID: string equals

Platform access

- [ ] Viewer 10 As a viewer, you can view service instances, but you can't modify them.

Service access

- [x] Reader 17 As a reader, you can perform read-only actions within a service such as viewing service-specific resources.

8. In the **Target service instance** menu, select the service instance to authorize. The additional fields may be left blank.
9. Enable the **Reader** role.
10. Click **Authorize**.

Create a bucket

When your key exists in Key Protect and you authorized the service for use with IBM COS, associate the key with a new bucket:

1. Navigate to your instance of Object Storage.
2. Click **Create bucket**.
3. Select **Custom bucket**.
4. Enter a bucket name, select the **Regional** resiliency, and choose a location and storage class.
5. In **Service integrations**, toggle **Key management disabled** to enable encryption key management and click on **Use existing instance**.
6. Select the associated service instance and key, and click **Associate key**.
7. Verify the information is correct.
8. Click **Create**.

Important: You can choose to use Key Protect to manage encryption for a bucket only at the time of creation. It isn't possible to change an existing bucket to use Key Protect.

Tip: If bucket creation fails with a `400 Bad Request` error with the message `The Key CRN could not be found`, ensure that the CRN is correct and that the service to service authorization policy exists.

In the **Buckets** listing, the bucket has a *View* link under **Attributes** where you can verify that the bucket has a Key Protect key enabled.

Tip: Note that the `Etag` value returned for objects encrypted using SSE-KP **will** be the actual MD5 hash of the original decrypted object.

It is also possible to use the REST API or SDKs (Go, Java, Node.js, or Python).

Key lifecycle management

Key Protect offers various ways to manage the lifecycle of encryption keys. For more details, see the Key Protect documentation.

Rotating Keys

Key rotation is an important part of mitigating the risk of a data breach. Periodically changing keys reduces the potential data loss if the key is lost or compromised. The frequency of key rotations varies by organization and depends on a number of variables, such as the environment, the amount of encrypted data, classification of the data, and compliance laws. The National Institute of Standards and Technology (NIST) provides definitions of appropriate key lengths and provides guidelines for how long keys should be used.

For more information, see the documentation for rotating keys in Key Protect.

Disabling and re-enabling keys

As an admin, you might need to temporarily disable a root key if you suspect a possible security exposure, compromise, or breach with your data. When you disable a root key, you suspend its encrypt and decrypt operations. After confirming that a security risk is no longer active, you can reestablish access to your data by enabling the disabled root key.

Note: If a key is disabled, and then re-enabled quickly, requests made to that bucket may be rejected for up to an hour before cached key information is refreshed.

Deleting keys and cryptographic erasure

Important: It isn't possible to delete a root key associated with a bucket that has a retention policy in place. The bucket must be first emptied and destroyed before the root key can be deleted. For more information, see the Key Protect documentation.

Cryptographic erasure (or crypto-shredding) is a method of rendering encrypted data unreadable by deleting the encryption keys rather than the data itself. When a root key is deleted in Key Protect, it will affect all objects in any buckets created using that root key, effectively "shredding" the data and preventing any further reading or writing to the buckets. This process is not instantaneous, but occurs within about 90 seconds after the key is deleted.

Tip: Although objects in a crypto-shredded bucket can not be read, and new object can not be written, existing objects will continue to consume storage until they are deleted by a user.

Restoring a deleted key

As an admin, you might need to restore a root key that you imported to Key Protect so that you can access data that the key previously protected. When you restore a key, you move the key from the Destroyed to the Active key state, and you restore access to any data that was previously encrypted with the key. This must occur within 30 days of deleting a key.

Activity Tracking

When Key Protect root keys are deleted, rotated, suspended, enabled, or restored, an Activity Tracker management event (`cloud-object-storage.bucket-key-state.update`) is generated in addition to any events logged by Key Protect.

Note: In the event of a server-side failure in a lifecycle action on a key, that failure is not logged by COS. If Key Protect does not receive a success from COS for the event handling within four hours of the event being sent, Key Protect will log a failure.

The `cloud-object-storage.bucket-key-state.update` actions are triggered by events taking place in Key Protect, and require that the bucket is registered with the Key Protect service. This registration happens automatically when a bucket is created with a Key Protect root key.

Important: Buckets created prior to February 26th, 2020 are not registered with the Key Protect service and will not receive notifications of encryption key lifecycle events at this time. These buckets can be identified by performing a bucket listing operation and looking at the dates for bucket creation. To ensure that these buckets have the latest key state from Key Protect, it is recommended that some data operation is performed, such as a `PUT`, `GET`, or `HEAD` on an object in each affected bucket. It is recommended that an object operation is done twice, at least an hour apart, to ensure that the key state is properly in synchronization with the Key Protect state.

For more information on Activity Tracker events for object storage, see the reference topic.

Server-Side Encryption with Hyper Protect Crypto Services

You can use Hyper Protect Crypto Services to create, add, and manage keys, which you can then associate with your instance of IBM® Cloud Object

Storage to encrypt buckets.

Note: This feature is not currently supported in Object Storage for Satellite. Learn more.

Before you begin

Before you plan on using Hyper Protect Crypto Services with Cloud Object Storage buckets, you need:

- An IBM Cloud™ Platform account
- An instance of IBM Cloud Object Storage with a *standard* pricing plan.

You will need to ensure that a service instance is created by using the IBM Cloud catalog and appropriate permissions are granted. This section outlines step-by-step instructions to help you get started.

Provisioning an instance of Hyper Protect Crypto Services

Refer to the service-specific product pages for instructions on how to provision and setup appropriate service instances.

- Getting started with Hyper Protect Crypto Services

Once you have an instance of Hyper Protect Crypto Services, you need to create a root key and note the CRN (Cloud Resource Name) of that key. The CRN is sent in a header during bucket creation.

Before creating the bucket for use with Hyper Protect Crypto Services, review the relevant guidance around availability and disaster recovery .

Create or add a key in Hyper Protect Crypto Services

Navigate to your instance of Hyper Protect Crypto Services and initialize the service instance . Once a master key has been created, generate or enter a root key.

Grant service authorization

Authorize Hyper Protect Crypto Services for use with IBM COS:

1. Open your IBM Cloud dashboard.
2. From the menu bar, click **Manage > Access (IAM)**.
3. In the side navigation, click **Authorizations**.
4. Click **Create** to create an authorization.
5. In the **Source service** menu, select **Cloud Object Storage**.
6. In the **Source service instance** menu, select the service instance to authorize.
7. In the **Target service** menu, select **Hyper Protect Crypto Services**.
8. In the **Target service instance** menu, select the service instance to authorize.
9. Enable the **Reader** role.
10. Click **Authorize**.

Create a bucket

When your key exists in Hyper Protect Crypto Services and you authorized the service for use with IBM COS, you can now associate the key with a new bucket:

1. Navigate to your instance of Object Storage.
2. Click **Create bucket**.
3. Select **Custom bucket**.
4. Enter a bucket name, select the resiliency (only Regional and US Cross Region are currently supported), and choose a location and storage class.
5. In **Service integrations**, toggle **Key management disabled** to enable encryption key management and click on **Use existing instance**.
6. Select the associated service instance and key, and click **Associate key**.
7. Verify the information is correct.
8. Click **Create**.

Important: You can choose to use Hyper Protect Crypto Services to manage encryption for a bucket only at the time of creation. It isn't possible to change an existing bucket to use Hyper Protect Crypto Services.

Tip: If bucket creation fails with a `400 Bad Request` error with the message `The Key CRN could not be found`, ensure that the CRN is correct and that the service to service authorization policy exists.

In the `Buckets` listing, the bucket now has a *View* link under `Attributes`, indicating that the bucket has a Hyper Protect Crypto Services key enabled. To view the key details (along with other object metadata), click `View`.

Tip: Note that the `Etag` value returned for objects encrypted using Hyper Protect Crypto Services **will** be the actual MD5 hash of the original decrypted object.

It is also possible to use the REST API or SDKs (Go, Java, Node.js, or Python).

Creating Cross Region buckets

Creating COS Cross Region bucket with a root key from a Hyper Protect Crypto Services instance requires that instance to be configured with failover configuration.

You can confirm that failover is properly configured for the selected Hyper Protect Crypto Services instance correctly using either the IBM Cloud console or CLI.

From IBM Cloud console, navigate to a Hyper Protect Crypto Services instance and click on **Overview**. A "Failover" section will indicate the status of crypto units in the corresponding failover regions.

Ensure the failover section is present, all validation checks are green and there are no warnings for that Hyper Protect Crypto Services instance. If you see any errors or warnings, or if the failover section is not present, refer to the Hyper Protect Crypto Services documentation for further guidance .

You can also use the CLI to list all the crypto units for all instances belong to the targeted resource group:

```
$ ibmcloud tke cryptounits
```

To get status of crypto units for the selected instance, create a list of crypto units associated with that instance and compare them:

```
$ ibmcloud tke cryptounit-add
```

After the units have been selected, you can check their verification patterns:

```
$ ibmcloud tke cryptounit-compare
```

Make sure all are valid and have same verification pattern.

Once the presence of the failover configuration is verified, you may proceed to create the Cross Region bucket using the key from that Hyper Protect Crypto Services instance.

Important: If the Cross Region bucket creation in US Cross Region with a Hyper Protect Crypto Services root key fails with a `500` error, then the user is advised to check if the status of failover configuration for that Hyper Protect Crypto Services instance (using the methods detailed above) before reattempting the bucket creation.

Key lifecycle management

Hyper Protect Crypto Services offers various ways to manage the lifecycle of encryption keys. For more details, see the Hyper Protect Crypto Services documentation.

Rotating Keys

Key rotation is an important part of mitigating the risk of a data breach. Periodically changing keys reduces the potential data loss if the key is lost or compromised. The frequency of key rotations varies by organization and depends on a number of variables, such as the environment, the amount of encrypted data, classification of the data, and compliance laws. The National Institute of Standards and Technology (NIST) provides definitions of appropriate key lengths and provides guidelines for how long keys should be used.

For more information, see the documentation for rotating keys in Hyper Protect Crypto Services.

Disabling and re-enabling keys

As an admin, you might need to temporarily disable a root key if you suspect a possible security exposure, compromise, or breach with your data. When you disable a root key, you suspend its encrypt and decrypt operations. After confirming that a security risk is no longer active, you can reestablish access to your data by enabling the disabled root key.

Deleting keys and cryptographic erasure

Cryptographic erasure (or crypto-shredding) is a method of rendering encrypted data unreadable by deleting the encryption keys rather than the data itself. When a root key is deleted in Hyper Protect Crypto Services , it will affect all objects in any buckets created using that root key, effectively "shredding" the data and preventing any further reading or writing to the buckets. This process is not instantaneous, but occurs within about 90 seconds after the key is deleted.

Tip: Although objects in a crypto-shredded bucket can not be read, and new object can not be written, existing objects will continue to consume storage until they are deleted by a user.

Restoring a deleted key

As an admin, you might need to restore a root key that you imported to Hyper Protect Crypto Services so that you can access data that the key previously protected. When you restore a key, you move the key from the Destroyed to the Active key state, and you restore access to any data that was previously encrypted with the key. This must occur within 30 days of deleting a key.

Important: If a key that was originally uploaded by a user is deleted, and then restored using different key material, it **will result in a loss of data** . It is recommended to keep n-5 keys archived somewhere to ensure that the correct key material is available for restoration.

Activity Tracking

When Hyper Protect Crypto Services root keys are deleted, rotated, suspended, enabled, or restored, an Activity Tracker management event (`cloud-object-storage.bucket-key-state.update`) is generated in addition to any events logged by Hyper Protect Crypto Services.

Note: In the event of a server-side failure in a lifecycle action on a key, that failure is not logged by COS. If Hyper Protect Crypto Services does not receive a success from COS for the event handling within four hours of the event being sent, Hyper Protect Crypto Services will log a failure.

The `cloud-object-storage.bucket-key-state.update` actions are triggered by events taking place in Hyper Protect Crypto Services, and require that the bucket is registered with the Hyper Protect Crypto Services service. This registration happens automatically when a bucket is created with a Hyper Protect Crypto Services root key.

Important: Buckets created before February 26th, 2020 are not registered with the Hyper Protect Crypto Services service and will not receive notifications of encryption key lifecycle events at this time. These buckets can be identified by performing a bucket listing operation and looking at the dates for bucket creation. To ensure that these buckets have the latest key state from Hyper Protect Crypto Services, it is recommended that some data operation is performed, such as a `PUT`, `GET`, or `HEAD` on an object in each affected bucket. It is recommended that an object operation is done twice, at least an hour apart, to ensure that the key state is properly in synchronization with the Hyper Protect Crypto Services state.

For more information on Activity Tracker events for object storage, see the reference topic.

Tracking events on your IBM Cloud Object Storage buckets

IBM Cloud offers centralized logging services to track events performed on your resources. You can use these services to investigate abnormal activity and critical actions and comply with regulatory audit requirements.

Use these services to track events on your IBM Cloud® Object Storage buckets to provide a record of what is happening with your data. Enable these services on your bucket to receive detailed logs about data access and bucket configuration events.

When event tracking is enabled on your bucket, the default target service that captures these events is IBM Cloud® Activity Tracker. Ensure that you have an instance of Activity Tracker at the receiving location corresponding to your bucket location as specified in Object Storage Service Integration.

Alternatively, use IBM Cloud Activity Tracker Event Routing to send events to other target services or to send events to Activity Tracker instances in locations other than the bucket location.

Note: This feature is not currently supported in Object Storage for Satellite.

Note: This feature supports IBM Cloud Activity Tracker to learn more.

Using IBM Cloud Logs to track bucket events (Coming Soon)

IBM Cloud Logs will replace IBM Cloud Activity Tracker hosted event search. See the IBM Announcement to learn more.

IBM Cloud Logs gives you flexibility in how your data is processed for insights and trends, and where data is stored for high-speed search and long-term trend analysis. It provides the tools for you to maximize the value obtained while maintaining control on the total cost.

Migrate from IBM Cloud Activity Tracker to IBM Cloud Logs once available to avoid any disruption in event tracking.

Route Logs with IBM Cloud Activity Tracker Event Routing

Get started with IBM Cloud Activity Tracker Event Routing to configure routing for your IBM Cloud Object Storage auditing events. You can use Activity Tracker Event Routing, a platform service, to manage auditing events at the account-level by configuring targets and routes that define where auditing data is routed.

Activity Tracker Event Routing supports routing IBM COS bucket logs to the following targets

- Another COS Bucket
- Activity Tracker Instance (Deprecated)
- Event Streams
- IBM Cloud Logs (Coming Soon)

Configure Activity Tracking Events on your IBM Cloud Object Storage Bucket (Recommended)

Event tracking can be enabled on your IBM Cloud Object Storage bucket at the time of bucket provisioning, or by updating the bucket configuration after bucket creation. Event tracking will only apply to COS requests made after enablement.

By default, COS events that report on global actions, such as bucket creation, are collected automatically. You can monitor global actions through the Activity Tracker instance located in the Frankfurt location.

IBM COS also optionally supports tracking on these event types:

- Management Events – Requests related to managing bucket and object configuration
- Read Data Events – Requests related to object list and read requests
- Write Data Events – These are all events related to writing and deleting objects

Refer to the COS API events to see the full list of Management, Read Data, and Write Data actions that produce events.

Use the COS Resource Configuration API to configure tracking of these events on your bucket

When event tracking is enabled, all events are sent to the default receiving location for IBM Cloud Activity Tracker Event Router that are based on the location of the bucket. Refer to IBM COS Service Integration to see this default mapping. Use Activity Tracker Event Router rules to route events to an alternative location or target service. See Managing Rules to learn more.

How to configure IBM Cloud Activity Tracker on your bucket (Recommended)

Select the UI, API or Terraform tab at the top of this topic to display the examples that show how to enable tracking of management, data read, and data write events in your bucket.

UI example for how to enable tracking of events in your bucket

1. From the IBM Cloud console resource list, select the service instance that contains the bucket you are interested in adding event tracking. This takes you to the Object Storage Console
2. Choose the bucket for which you want to enable event tracking.
3. Navigate to the configuration tab.
4. Scroll down to the advanced configuration section and toggle on the events you want to track for this bucket.
5. After a few minutes, any activity will be visible in the Activity Tracker web UI.

Examples

JAVA SDK

```
import com.ibm.cloud.objectstorage.config.resource_configuration.v1.ResourceConfiguration;
import com.ibm.cloud.objectstorage.config.resource_configuration.v1.model.BucketPatch;
import com.ibm.cloud.sdk.core.security.IamAuthenticator;
```

```
public class ActivityTrackerExample {
   private static final String BUCKET_NAME = <BUCKET_NAME>;
   private static final String API_KEY = <API_KEY>;

   public static void main(String[] args) {
       IamAuthenticator authenticator = new IamAuthenticator.Builder()
               .apiKey(API_KEY)
               .build();
       ResourceConfiguration RC_CLIENT = new ResourceConfiguration("resource-configuration", authenticator);
       ActivityTracking activityTrackingConfig = new ActivityTracking().Builder()
               .readDataEvents(true)
               .writeDataEvents(true)
               .managementEvents(true)
               .build();
       BucketPatch bucketPatch = new BucketPatch.Builder().activityTracking(activityTrackingConfig).build();
       UpdateBucketConfigOptions update = new UpdateBucketConfigOptions
               .Builder(BUCKET_NAME)
               .bucketPatch(bucketPatch.asPatch())
               .build();

       RC_CLIENT.updateBucketConfig(update).execute();
       GetBucketConfigOptions bucketOptions = new GetBucketConfigOptions.Builder(BUCKET_NAME).build();
       Bucket bucket = RC_CLIENT.getBucketConfig(bucketOptions).execute().getResult();

       ActivityTracking activityTrackingResponse = bucket.getActivityTracking();
       System.out.println("Read Data Events : " + activityTrackingResponse.readDataEvents());
       System.out.println("Write Data Events : " + activityTrackingResponse.writeDataEvents());
       System.out.println("Management Events : " + activityTrackingResponse.managementEvents());
   }
}
```

NodeJS SDK

```
const ResourceConfigurationV1 = require('ibm-cos-sdk-config/resource-configuration/v1');
IamAuthenticator      = require('ibm-cos-sdk-config/auth');

var apiKey = "<API_KEY>"
var bucketName = "<BUCKET_NAME>"

authenticator = new IamAuthenticator({apikey: apiKey})
rcConfig = {authenticator: authenticator}
const client = new ResourceConfigurationV1(rcConfig);

function addAT() {
    console.log('Updating bucket metadata...');

    var params = {
        bucket: bucketName,
        activityTracking: {
          "read_data_events": true,
          "write_data_events": true,
          "management_events": true
          }
    };

    client.updateBucketConfig(params, function (err, response) {
        if (err) {
            console.log("ERROR: " + err);
        } else {
            console.log(response.result);
        }
    });
}

addAT()
```

Python SDK

```
from ibm_cos_sdk_config.resource_configuration_v1 import ResourceConfigurationV1
from ibm_cloud_sdk_core.authenticators import IAMAuthenticator
```

```
api_key = "<API_KEY>"
bucket_name = "<BUCKET_NAME>"

authenticator = IAMAuthenticator(apikey=api_key)
client = ResourceConfigurationV1(authenticator=authenticator)
activity_tracking_config = {
                            'activity_tracking':
                              {
                                'read_data_events':True,
                                'write_data_events':True,
                                'management_events':True
                              }
                            }
client.update_bucket_config(bucket_name, bucket_patch=activity_tracking_config)
```

GO SDK example

```
import (
"github.com/IBM/go-sdk-core/core"
rc "github.com/IBM/ibm-cos-sdk-go-config/v2/resourceconfigurationv1"
)

apiKey := "<ApiKey>"
bucketName := "<BucketName>"

authenticator := new(core.IamAuthenticator)
authenticator.ApiKey = apiKey
optionsRC := new(rc.ResourceConfigurationV1Options)
optionsRC.Authenticator = authenticator
rcClient, _ := rc.NewResourceConfigurationV1(optionsRC)

patchNameMap := make(map[string]interface{})
patchNameMap["activity_tracking"] = &rc.ActivityTracking{
  ReadDataEvents:     core.BoolPtr(true),
  WriteDataEvents:    core.BoolPtr(true),
  ManagementEvents:    core.BoolPtr(true)
}
updateBucketConfigOptions := &rc.UpdateBucketConfigOptions{
  Bucket:      core.StringPtr(bucketName),
  BucketPatch: patchNameMap,
}
rcClient.UpdateBucketConfig(updateBucketConfigOptions)
```

Example

```
resource "ibm_resource_instance" "cos_instance" {
  name              = "cos-instance"
  resource_group_id = data.ibm_resource_group.cos_group.id
  service           = "cloud-object-storage"
  plan              = "standard"
  location          = "global"
}

resource "ibm_cos_bucket" "activity_tracker_bucket" {
  bucket_name          = “Name-of-the-bucket”
  resource_instance_id = ibm_resource_instance.cos_instance.id
  region_location      = “us-south”
  storage_class        = “standard”
  activity_tracking {
  read_data_events     = true
    write_data_events    = true
    management_events    = true
      }
}
```

Configure Activity Tracking Events on your IBM Cloud Object Storage Bucket (Legacy)

Enable IBM Activity Tracking on your COS bucket by specifying the target CRN of the Activity Tracker instance in the [COS Resource Configuration API](#).

Specify the CRN to define the route for COS events.

Management events are always enabled when a CRN is set on the Activity Tracking configuration

The legacy model also supports optionally enabling tracking on the following event types:

- Read Data Events – Requests related to object list and read requests
- Write Data Events – These are all events related to writing and deleting objects

Note: IBM Cloud observability routing services are the standardized way for customers to manage routing of platform observability data. Service-specific routing configurations like COS are being deprecated.

It is recommended that customers remove these legacy routing configurations that use CRNs and instead use the IBM Activity Tracker Event Routing service to route events to other locations.

IBM COS will continue to support legacy configurations where a CRN was specified that differs from the default location.

Upgrading from Legacy to the Recommended Event Tracking on your COS bucket

To upgrade from the legacy configuration using the Resource Configuration API, remove the target Activity Tracker instance CRN. Events will now route to the default Activity Tracker Event Router receiving location as described in COS Service Integration. Provision an instance of Activity Tracker hosted event search at this location or define a routing rule prior to upgrading to ensure there's no interruption in event logging.

Example patch to transition from the Legacy to Recommended event tracking configuration on your COS bucket

Select the UI, API, or Terraform tab at the top of this topic to see examples of patchs.

UI example patch to transition from the Legacy to Recommended event tracking configuration on your COS bucket

1. From the IBM Cloud console resource list, select the service instance that contains the bucket you wish to upgrade to the recommended event tracking configuration. This takes you to the Object Storage Console.
2. Choose the bucket for which you want to upgrade.
3. Navigate to the configuration tab.
4. Scroll down to the advanced configuration section and locate the configuration panel for Activity Tracker.
5. Click on the top right corner of the panel and select upgrade.
6. Confirm you would like to upgrade event tracking for this bucket.

Examples

JAVA SDK

```
import com.ibm.cloud.objectstorage.config.resource_configuration.v1.ResourceConfiguration;
import com.ibm.cloud.objectstorage.config.resource_configuration.v1.model.BucketPatch;
import com.ibm.cloud.sdk.core.security.IamAuthenticator;

public class ActivityTrackerExample {
   private static final String BUCKET_NAME = <BUCKET_NAME>;
   private static final String API_KEY = <API_KEY>;

   public static void main(String[] args) {
       IamAuthenticator authenticator = new IamAuthenticator.Builder()
               .apiKey(API_KEY)
               .build();
       ResourceConfiguration RC_CLIENT = new ResourceConfiguration("resource-configuration", authenticator);
       ActivityTracking activityTrackingConfig = new ActivityTracking().Builder()
             .activityTrackerCrn(AT_CRN)
             .readDataEvents(true)
             .writeDataEvents(true)
             .build();
       BucketPatch bucketPatch = new BucketPatch.Builder().activityTracking(activityTrackingConfig).build();
       UpdateBucketConfigOptions update = new UpdateBucketConfigOptions
               .Builder(BUCKET_NAME)
               .bucketPatch(bucketPatch.asPatch())
               .build();
```

```
        RC_CLIENT.updateBucketConfig(update).execute();
        GetBucketConfigOptions bucketOptions = new GetBucketConfigOptions.Builder(BUCKET_NAME).build();
        Bucket bucket = RC_CLIENT.getBucketConfig(bucketOptions).execute().getResult();

        ActivityTracking activityTrackingResponse = bucket.getActivityTracking();
        System.out.println("Read Data Events : " + activityTrackingResponse.readDataEvents());
        System.out.println("Write Data Events : " + activityTrackingResponse.writeDataEvents());
        System.out.println("Management Events : " + activityTrackingResponse.managementEvents());
    }
}
```

NodeJS SDK

```
const ResourceConfigurationV1 = require('ibm-cos-sdk-config/resource-configuration/v1');
IamAuthenticator      = require('ibm-cos-sdk-config/auth');

var apiKey = "<API_KEY>"
var bucketName = "<BUCKET_NAME>"

authenticator = new IamAuthenticator({apikey: apiKey})
rcConfig = {authenticator: authenticator}
const client = new ResourceConfigurationV1(rcConfig);

function addAT() {
    console.log('Updating bucket metadata...');

    };
    var params = {
        bucket: bucketName,
        activityTracking: {
          "activity_tracker_crn": at_crn,
          "read_data_events": true,
          "write_data_events": true
          }
    };

    client.updateBucketConfig(params, function (err, response) {
        if (err) {
            console.log("ERROR: " + err);
        } else {
            console.log(response.result);
        }
    });
}

addAT()
```

Python SDK

```
from ibm_cos_sdk_config.resource_configuration_v1 import ResourceConfigurationV1
from ibm_cloud_sdk_core.authenticators import IAMAuthenticator

api_key = "<API_KEY>"
bucket_name = "<BUCKET_NAME>"

authenticator = IAMAuthenticator(apikey=api_key)
client = ResourceConfigurationV1(authenticator=authenticator)
activity_tracking_config = {
                            'activity_tracking':
                              {
                                'activity_tracker_crn':at_crn,
                                'read_data_events':True,
                                'write_data_events':True,
                              }
                            }

client.update_bucket_config(bucket_name, bucket_patch=activity_tracking_config)
```

GO SDK

```
import (
"github.com/IBM/go-sdk-core/core"
rc "github.com/IBM/ibm-cos-sdk-go-config/v2/resourceconfigurationv1"
)

apiKey := "<ApiKey>"
bucketName := "<BucketName>"

authenticator := new(core.IamAuthenticator)
authenticator.ApiKey = apiKey
optionsRC := new(rc.ResourceConfigurationV1Options)
optionsRC.Authenticator = authenticator
rcClient, _ := rc.NewResourceConfigurationV1(optionsRC)

patchNameMap := make(map[string]interface{})
patchNameMap["activity_tracking"] = &rc.ActivityTracking{
  ActivityTrackerCrn: core.StringPtr(activityTrackerCrn),
  ReadDataEvents:     core.BoolPtr(true),
  WriteDataEvents:    core.BoolPtr(true),
}
updateBucketConfigOptions := &rc.UpdateBucketConfigOptions{
  Bucket:      core.StringPtr(bucketName),
  BucketPatch: patchNameMap,
}
rcClient.UpdateBucketConfig(updateBucketConfigOptions)
```

Example

```
resource "ibm_resource_instance" "cos_instance" {
  name              = "cos-instance"
  resource_group_id = data.ibm_resource_group.cos_group.id
  service           = "cloud-object-storage"
  plan              = "standard"
  location          = "global"
}

resource "ibm_cos_bucket" "activity_tracker_bucket" {
  bucket_name          = “bucket_name”
  resource_instance_id = ibm_resource_instance.cos_instance.id
  region_location      = “us-south”
  storage_class        = “standard”
  activity_tracking {
  read_data_events     = true
    write_data_events    = true
    activity_tracker_crn = “crn:v1:bluemix:public:logdnaat:us-south:a/2xxxxxxxxxxxxxxxxxxxxxxxxf:3xxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxec::”
      }
}
```

Configure Metrics for IBM Cloud® Object Storage

Use the IBM Cloud® Monitoring service to monitor your IBM Cloud® Object Storage data. IBM Cloud Monitoring is a cloud-native management system. The metrics produced by your COS buckets can be displayed in dashboards built in IBM Monitoring. Documentation from Monitoring can guide you in how to use the comprehensive dashboards. Additionally, specify the conditions when a metrics alert is trigged to set notifications when custom thresholds are exceeded.

When metrics monitoring is enabled on your bucket, the default target service that captures these metrics is IBM Cloud Monitoring. Ensure that you have a platform instance of IBM Cloud Monitoring at the receiving location corresponding to your bucket location as specified in COS Service Integration.

Alternatively, use IBM Cloud Metrics Routing rules to send metrics to other target services or to IBM Cloud Monitoring instances in locations other than the bucket location.

Enable metrics monitoring on your bucket via the IBM Cloud® Object Storage Resource Configuration API or through the UI directly. This is done during bucket provisioning or afterwards by updating the bucket configuration.

IBM COS supports enabling metrics tracking on the following metric types:

- Usage Metrics – These are metrics related to the overall usage of your COS bucket such as total storage consumed in bytes.
- Request Metrics – The metrics report the counts for certain types of API requests made to your bucket

See the IBM Cloud® Object Storage metrics details section below for the full list of metrics sent to IBM Monitoring.

Note: This feature is not currently supported in Object Storage for Satellite.

Note: This feature supports COS Service Integration. You must have an instance of IBM Monitoring at this location or configure a routing rule to another location with a Monitoring instance, to ensure metrics are received.

See Getting started with IBM Cloud Metrics Routing for more information.

Configure Metrics on your IBM Cloud® Object Storage Bucket (Recommended)

Enable metrics tracking on your IBM Cloud® Object Storage bucket at the time of bucket provisioning or by updating the bucket configuration after bucket creation. Metrics monitoring will only apply to IBM Cloud® Object Storage metrics produced after enablement.

Refer to the IBM Cloud® Object Storage Metrics Details to see the full list of Usage and Request metrics available for tracking.

Use the IBM Cloud® Object Storage Resource Configuration API to configure tracking of these metrics for your bucket.

When metrics tracking is enabled, all metrics are sent to the default receiving location for IBM Cloud Metrics Router based on the location of the bucket. Refer to IBM Cloud® Object Storage Service Integration to see this default mapping. Use Metrics Router rules to route metrics to a location other than the bucket location or to another target service. See Managing Routes for more information.

How to configure Metrics for IBM Cloud® Object Storage (Recommended)

Select the UI, API or Terraform tab at the top of this topic to display the examples that show how to configure metrics monitoring to track both usage and request metrics on your bucket.

UI example for how to configure Metrics Monitoring on your bucket

1. From the IBM Cloud console resource list, select the service instance that contains the bucket you are interested in adding metrics monitoring. This takes you to the Object Storage Console
2. Choose the bucket for which you want to enable monitoring.
3. Navigate to the configuration tab.
4. Scroll down to the advanced configuration section and toggle on the metrics you want to monitor for this bucket.
5. After a few minutes, any activity will be visible in the IBM Cloud Monitoring web UI.

Examples

JAVA SDK

```
import com.ibm.cloud.objectstorage.config.resource_configuration.v1.ResourceConfiguration;
import com.ibm.cloud.objectstorage.config.resource_configuration.v1.model.BucketPatch;
import com.ibm.cloud.sdk.core.security.IamAuthenticator;

public class MetricsMonitoringExample {

   private static final String BUCKET_NAME = <BUCKET_NAME>;
   private static final String API_KEY = <API_KEY>;

   public static void main(String[] args) {
      IamAuthenticator authenticator = new IamAuthenticator.Builder()
               .apiKey(API_KEY)
               .build();
      ResourceConfiguration RC_CLIENT = new ResourceConfiguration("resource-configuration", authenticator);
      MetricsMonitoring metricsMonitoringConfig = new MetricsMonitoring().Builder()
            .requestMetricsEnabled(true)
            .usageMetricsEnabled(true)
            .build();
      BucketPatch bucketPatch = new BucketPatch.Builder().metricsMonitoring(metricsMonitoringConfig).build();
      UpdateBucketConfigOptions update = new UpdateBucketConfigOptions
               .Builder(BUCKET_NAME)
               .bucketPatch(bucketPatch.asPatch())
```

```
                .build();
        RC_CLIENT.updateBucketConfig(update).execute();

        GetBucketConfigOptions bucketOptions = new GetBucketConfigOptions.Builder(BUCKET_NAME).build();
        Bucket bucket = RC_CLIENT.getBucketConfig(bucketOptions).execute().getResult();
        MetricsMonitoring metricsMonitoringResponse = bucket.getMetricsMonitoring();
        System.out.println("Usage Metrics Enabled  : " + metricsMonitoringResponse.usageMetricsEnabled());
        System.out.println("Request Metrics Enabled : " + metricsMonitoringResponse.requestMetricsEnabled());
    }
}
```

NodeJS SDK

```
const ResourceConfigurationV1 = require('ibm-cos-sdk-config/resource-configuration/v1');
IamAuthenticator      = require('ibm-cos-sdk-config/auth');

var apiKey = "<API_KEY>"
var bucketName = "<BUCKET_NAME>"

authenticator = new IamAuthenticator({apikey: apiKey})
rcConfig = {authenticator: authenticator}
const client = new ResourceConfigurationV1(rcConfig);

function addMM() {
   console.log('Updating bucket metadata...');

   var params = {
      bucket: bucketName,
      metricsMonitoring: {
         "request_metrics_enabled": true,
         "usage_metrics_enabled": true
         }
   };

   client.updateBucketConfig(params, function (err, response) {
      if (err) {
            console.log("ERROR: " + err);
      } else {
            console.log(response.result);
      }
   });
}

addMM()
```

Python SDK

```
from ibm_cos_sdk_config.resource_configuration_v1 import ResourceConfigurationV1
from ibm_cloud_sdk_core.authenticators import IAMAuthenticator

api_key = "<API_KEY>"
bucket_name = "<BUCKET_NAME>"

authenticator = IAMAuthenticator(apikey=api_key)
client = ResourceConfigurationV1(authenticator=authenticator)
metrics_monitoring_config = {'metrics_monitoring':
                           {
                              'request_metrics_enabled':True,
                              'usage_metrics_enabled':True
                              }
                           }
client.update_bucket_config(bucket_name, bucket_patch=metrics_monitoring_config
```

GO SDK

```
import (
"github.com/IBM/go-sdk-core/core"
rc "github.com/IBM/ibm-cos-sdk-go-config/v2/resourceconfigurationv1"
)
```

```
apiKey := "<ApiKey>"
bucketName := "<BucketName>"

authenticator := new(core.IamAuthenticator)
authenticator.ApiKey = apiKey
optionsRC := new(rc.ResourceConfigurationV1Options)
optionsRC.Authenticator = authenticator
rcClient, _ := rc.NewResourceConfigurationV1(optionsRC)

patchNameMap := make(map[string]interface{})
patchNameMap["metrics_monitoring"] = &rc.MetricsMonitoring{
RequestMetricsEnabled:     core.BoolPtr(true),
UsageMetricsEnabled:    core.BoolPtr(true)
}
updateBucketConfigOptions := &rc.UpdateBucketConfigOptions{
Bucket:      core.StringPtr(bucketName),
BucketPatch: patchNameMap,
}
rcClient.UpdateBucketConfig(updateBucketConfigOptions)
```

Example

```
resource "ibm_resource_instance" "cos_instance" {
   name              = "cos-instance"
   resource_group_id = data.ibm_resource_group.cos_group.id
   service           = "cloud-object-storage"
   plan              = "standard"
   location          = "global"
}

resource "ibm_cos_bucket" "metric_monitoring_bucket" {
   bucket_name          = “bucket_name”
   resource_instance_id = ibm_resource_instance.cos_instance.id
   region_location      = “us-south”
   storage_class        = “standard”
   metrics_monitoring {
      usage_metrics_enabled   = true
      request_metrics_enabled = true
}
}
```

Configure Metrics on your IBM Cloud® Object Storage Bucket (Legacy)

Enable IBM Metrics Monitoring on your IBM Cloud® Object Storage bucket by specifying the target CRN of the Monitoring instance in the IBM Cloud® Object Storage Resource Configuration API. Specify the CRN to define the route for COS metrics.

Note: IBM Cloud Metrics Routing is the standardized way for customers to manage routing of platform observability data. Service-specific routing configurations like IBM Cloud® Object Storage are being deprecated.

It is recommended that customers remove these legacy routing configurations (make this a link to upgrade section below) that use CRNs and instead use the IBM Metrics Router service to route metrics to other locations.

IBM Cloud® Object Storage will continue to support legacy configurations where a CRN was specified that differs from the default location.

Upgrading from Legacy to the Recommended Metrics Monitoring on your COS bucket:

To upgrade from the legacy configuration using the Resource Configuration API, remove the target Metrics Monitoring instance CRN. Metrics will now route to the default Metrics Router receiving location as described in COS Service Integration. Provision an instance of Monitoring at this location or define a routing rule prior to upgrading to ensure there’s no interruption in metrics monitoring.

Example patch to transition from the Legacy to Recommend event tracking configuration on your COS bucket

Select the UI, API or Terraform tab at the top of this topic to see examples of patches.

UI example patch to transition from the Legacy to Recommend event tracking configuration on your COS

bucket

Example patch to transition from the Legacy to Recommend metrics monitoring configuration on your IBM Cloud® Object Storage bucket (SDK, RC API, UI, Terraform)

1. From the IBM Cloud console resource list, select the service instance that contains the bucket you wish to upgrade to the recommended metrics monitoring configuration. This takes you to the Object Storage Console.
2. Choose the bucket for which you want to upgrade.
3. Navigate to the configuration tab.
4. Scroll down to the advanced configuration section and locate the configuration panel for metrics monitoring.
5. Click on the top right corner of the panel and select upgrade.
6. Confirm you would like to upgrade metrics monitoring for this bucket.

Examples

JAVA SDK

```
import com.ibm.cloud.objectstorage.config.resource_configuration.v1.ResourceConfiguration;
import com.ibm.cloud.objectstorage.config.resource_configuration.v1.model.BucketPatch;
import com.ibm.cloud.sdk.core.security.IamAuthenticator;

public class MetricsMonitoringExample {

   private static final String BUCKET_NAME = <BUCKET_NAME>;
   private static final String API_KEY = <API_KEY>;

   public static void main(String[] args) {
      IamAuthenticator authenticator = new IamAuthenticator.Builder()
               .apiKey(API_KEY)
               .build();
      ResourceConfiguration RC_CLIENT = new ResourceConfiguration("resource-configuration", authenticator);
      MetricsMonitoring metricsMonitoringConfig = new MetricsMonitoring().Builder()
            .metricsMonitoringCrn(MM_CRN)
            .requestMetricsEnabled(true)
            .usageMetricsEnabled(true)
            .build();
      BucketPatch bucketPatch = new BucketPatch.Builder().metricsMonitoring(metricsMonitoringConfig).build();
      UpdateBucketConfigOptions update = new UpdateBucketConfigOptions
               .Builder(BUCKET_NAME)
               .bucketPatch(bucketPatch.asPatch())
               .build();
      RC_CLIENT.updateBucketConfig(update).execute();

      GetBucketConfigOptions bucketOptions = new GetBucketConfigOptions.Builder(BUCKET_NAME).build();
      Bucket bucket = RC_CLIENT.getBucketConfig(bucketOptions).execute().getResult();
      MetricsMonitoring metricsMonitoringResponse = bucket.getMetricsMonitoring();
      System.out.println("Usage Metrics Enabled  : " + metricsMonitoringResponse.usageMetricsEnabled());
      System.out.println("Request Metrics Enabled : " + metricsMonitoringResponse.requestMetricsEnabled());
   }
}
```

NodeJS SDK

```
const ResourceConfigurationV1 = require('ibm-cos-sdk-config/resource-configuration/v1');
IamAuthenticator      = require('ibm-cos-sdk-config/auth');

var apiKey = "<API_KEY>"
var bucketName = "<BUCKET_NAME>"

authenticator = new IamAuthenticator({apikey: apiKey})
rcConfig = {authenticator: authenticator}
const client = new ResourceConfigurationV1(rcConfig);

function addMM() {
   console.log('Updating bucket metadata...');

   var params = {
      bucket: bucketName,
      metricsMonitoring: {
               "metrics_monitoring_crn": metricsCrn,
         "request_metrics_enabled": true,
         "usage_metrics_enabled": true
         }
   };

   client.updateBucketConfig(params, function (err, response) {
      if (err) {
            console.log("ERROR: " + err);
      } else {
            console.log(response.result);
      }
   });
}

addMM()
```

Python SDK

```
from ibm_cos_sdk_config.resource_configuration_v1 import ResourceConfigurationV1
from ibm_cloud_sdk_core.authenticators import IAMAuthenticator

api_key = "<API_KEY>"
bucket_name = "<BUCKET_NAME>"

authenticator = IAMAuthenticator(apikey=api_key)
client = ResourceConfigurationV1(authenticator=authenticator)
metrics_monitoring_config = {'metrics_monitoring':
                           {
                        'metrics_monitoring_crn': mm_crn,
                              'request_metrics_enabled':True,
                              'usage_metrics_enabled':True
                              }
                           }

client.update_bucket_config(bucket_name, bucket_patch=metrics_monitoring_config
```

GO SDK

```
import (
"github.com/IBM/go-sdk-core/core"
rc "github.com/IBM/ibm-cos-sdk-go-config/v2/resourceconfigurationv1"
)

apiKey := "<ApiKey>"
bucketName := "<BucketName>"

authenticator := new(core.IamAuthenticator)
authenticator.ApiKey = apiKey
optionsRC := new(rc.ResourceConfigurationV1Options)
optionsRC.Authenticator = authenticator
rcClient, _ := rc.NewResourceConfigurationV1(optionsRC)

patchNameMap["metrics_monitoring"] = &rc.MetricsMonitoring{
MetricsMonitoringCrn: core.StringPtr(MMCrn),
RequestMetricsEnabled:     core.BoolPtr(true),
UsageMetricsEnabled:    core.BoolPtr(true)
}

updateBucketConfigOptions := &rc.UpdateBucketConfigOptions{
Bucket:      core.StringPtr(bucketName),
BucketPatch: patchNameMap,
}
rcClient.UpdateBucketConfig(updateBucketConfigOptions)
```

Example

```
resource "ibm_resource_instance" "cos_instance" {
   name             = "cos-instance"
   resource_group_id = data.ibm_resource_group.cos_group.id
   service          = "cloud-object-storage"
   plan             = "standard"
   location         = "global"
}

resource "ibm_cos_bucket" "metric_monitoring_bucket" {
   bucket_name          = “bucket_name”
   resource_instance_id = ibm_resource_instance.cos_instance.id
   region_location      = “us-south”
   storage_class        = “standard”
   metrics_monitoring {
      usage_metrics_enabled   = true
      request_metrics_enabled = true
      metrics_monitoring_crn = "crn:v1:bluemix:public:sysdig-monitor:us-east:a/xxxxxxxxxxxxxxxxxxxxxxxx:4xxxxxxxx-xxxx-xxxx-
xxxx-fxxxxxxxxx4c::"
}
}
```

Cloud Object Storage metrics details

Usage metrics

There are a set of basic metrics that track usage:

- `ibm_cos_bucket_used_bytes`
- `ibm_cos_bucket_object_count`
- `ibm_cos_bucket_hard_quota_bytes`

Request metrics

There are metrics that report the aggregates for different classes of HTTP requests:

- `ibm_cos_bucket_all_requests`
- `ibm_cos_bucket_get_requests`
- `ibm_cos_bucket_put_requests`
- `ibm_cos_bucket_delete_requests`
- `ibm_cos_bucket_post_requests`
- `ibm_cos_bucket_list_requests`
- `ibm_cos_bucket_head_requests`

Errors are also collected, with server-side (5xx) errors broken out:

- `ibm_cos_bucket_4xx_errors`
- `ibm_cos_bucket_5xx_errors`

The minimum, maximum, and average bytes transferred by network type are reported:

- `ibm_cos_bucket_bytes_download_public_min`
- `ibm_cos_bucket_bytes_download_public_max`
- `ibm_cos_bucket_bytes_download_public_avg`
- `ibm_cos_bucket_bytes_download_private_min`
- `ibm_cos_bucket_bytes_download_private_max`
- `ibm_cos_bucket_bytes_download_private_avg`
- `ibm_cos_bucket_bytes_download_direct_min`
- `ibm_cos_bucket_bytes_download_direct_max`
- `ibm_cos_bucket_bytes_download_direct_avg`
- `ibm_cos_bucket_bytes_upload_public_min`
- `ibm_cos_bucket_bytes_upload_public_max`
- `ibm_cos_bucket_bytes_upload_public_avg`
- `ibm_cos_bucket_bytes_upload_private_min`

- `ibm_cos_bucket_bytes_upload_private_max`
- `ibm_cos_bucket_bytes_upload_private_avg`
- `ibm_cos_bucket_bytes_upload_direct_min`
- `ibm_cos_bucket_bytes_upload_direct_max`
- `ibm_cos_bucket_bytes_upload_direct_avg`

Latency metrics (first byte and general) for requests are broken down by request type:

- `ibm_cos_bucket_first_byte_latency_read_min`
- `ibm_cos_bucket_first_byte_latency_read_max`
- `ibm_cos_bucket_first_byte_latency_read_avg`
- `ibm_cos_bucket_first_byte_latency_write_min`
- `ibm_cos_bucket_first_byte_latency_write_max`
- `ibm_cos_bucket_first_byte_latency_write_avg`
- `ibm_cos_bucket_first_byte_latency_misc_min`
- `ibm_cos_bucket_first_byte_latency_misc_max`
- `ibm_cos_bucket_first_byte_latency_misc_avg`
- `ibm_cos_bucket_request_latency_read_min`
- `ibm_cos_bucket_request_latency_read_max`
- `ibm_cos_bucket_request_latency_read_avg`
- `ibm_cos_bucket_request_latency_write_min`
- `ibm_cos_bucket_request_latency_write_max`
- `ibm_cos_bucket_request_latency_write_avg`
- `ibm_cos_bucket_request_latency_misc_min`
- `ibm_cos_bucket_request_latency_misc_max`
- `ibm_cos_bucket_request_latency_misc_avg`

All metrics are reported as `float64` numeric values:

Attributes for Segmentation

You can filter your results by attributes. In this guide, we'll look at some general examples as well as those specific to IBM Cloud Object Storage.

Global Attributes

The following attributes are available for segmenting all the metrics listed above

Attribute	Attribute Name	Attribute Description
`Cloud Type`	`ibm_ctype`	public, dedicated or local
`Location`	`ibm_location`	The location of the monitored resource. This may be a Cross Region, Regional, or Single Site bucket.
`Resource`	`ibm_resource`	COS bucket name
`Resource Type`	`ibm_resource_type`	COS bucket
`Scope`	`ibm_scope`	The scope is the account associated with this metric.
`Service name`	`ibm_service_name`	cloud-object-storage

Table 4: IBM global attributes

Additional Attributes

The following attributes are available for segmenting one or more attributes as described in the reference above. Please see the individual metrics for segmentation options.

Attribute	Attribute Name	Attribute Description
`IBM COS Bucket storage class`	`ibm_cos_bucket_storage_class`	Storage class of the bucket
`Service instance`	`ibm_service_instance`	The service instance segment identifies the guide of the instance the metric is associated with.

Table 5: COS specific attributes

Using IBM Cloud Code Engine

Code Engine is a fully managed, serverless platform that runs your containerized workloads, including web apps, micro-services, event-driven functions, or batch jobs. Code Engine even builds container images for you from your source code. All these workloads can seamlessly work together because they are all hosted within the same Kubernetes infrastructure. The Code Engine experience is designed so that you can focus on writing code and not on the infrastructure that is needed to host it.

Using Object Storage as an event source

Using Object Storage as an event source Code Engine is an event-driven compute platform (also referred to as Serverless computing). Actions (small bits of code) run in response to triggers (some category of event), and rules associate certain actions with certain triggers. Configure IBM Cloud® Object Storage to be an event source, and anytime an object in a particular bucket is written or deleted an action is triggered. You can further tailor the changes feed to only corral events for objects which match a particular prefix or suffix. See Working with the IBM Cloud Object Storage event producer for more information.

Using Aspera high-speed transfer

Aspera high-speed transfer overcomes the limitations of traditional FTP and HTTP transfers to improve data transfer performance under most conditions, especially in networks with high latency and packet loss.

Note: This feature is not currently supported in Object Storage for Satellite. Learn more.

Instead of the standard HTTP `PUT` operation, Aspera high-speed transfer uploads the object by using the FASP protocol. Using Aspera high-speed transfer for uploads and downloads offers the following benefits:

- Faster transfer speeds
- Transfer large object uploads over 200 MB in the console and 1 GB by using an SDK or library
- Upload entire folders of any type of data, such as multi-media files, disk images, and any other structured or unstructured data
- Customize transfer speeds and default preferences
- Transfers can be viewed, paused, resumed, or cancelled independently

Aspera high-speed transfer is available in the IBM Cloud console and can also be used programmatically by using the Aspera Transfer SDK.

Tip: Aspera high-speed transfer is available in certain regions only. See Integrated Services for more details.

Important: It isn't possible to use Aspera high-speed transfer if a targeted bucket has an Immutable Object Storage policy.

Using the console

If you add objects by using the console in a supported region, you are prompted with an option to install the Aspera Connect client. This browser plug-in provides Aspera high-speed transfer to upload files or folders.

Install Aspera Connect

1. Select **Install Aspera Connect** client.
2. Follow the installation instructions for your operating system and browser.
3. Resume file or folder upload.

The Aspera Connect plug-in can also be installed from the Aspera website directly. For help troubleshooting issues with the Aspera Connect plug-in, see the documentation.

After the plug-in is installed, you have the option to set Aspera high-speed transfer as the default for any uploads to the target bucket that use the same browser. Select **Remember my browser preferences**. Options are also available in the bucket configuration page under **Transfer options**. These options allow you to choose between Standard and High speed as the default transport for uploads and downloads.

Typically, using the IBM Cloud Object Storage web-based console isn't the most common way to use Object Storage. The Standard transfer option limits objects size to 200 MB and the file name and key will be the same. Support for larger object sizes and improved performance (depending on network factors) is provided by Aspera high-speed transfer.

Important: An Aspera server runs one SSH server on a configurable TCP port (33001 by default). The firewall on the server side must allow this one TCP port to reach the Aspera server. No servers are listening on UDP ports. When a transfer is initiated by an Aspera client, the client opens an SSH session to the SSH server on the designated TCP port and negotiates the UDP port over which the data will travel. By default, Aspera clients and servers are configured to use UDP port 33001. After the session initiation step, both the client and the server will send and receive UDP traffic on the negotiated port. To allow the UDP session to start, the firewall on the Aspera server side must allow port UDP 33001 to reach the Aspera server. For more information, see Firewall Considerations.

Transfer status

Active: Once you initiate a transfer, the transfer status displays as active. While the transfer is active, you can pause, resume, or cancel an active transfer.

Completed: Upon completion of your transfer, information about this and all transfers in this session display on the completed tab. You can clear this information. You will only see information about transfers that are completed in the current session.

Preferences: You can set the default for uploads and downloads to High speed.

Tip: Downloads that use Aspera high-speed transfer incur egress charges. For more information, see the pricing page.

Advanced Preferences: You can set bandwidth for uploads and downloads.

Using the Aspera Transfer SDK

1. Download the Aspera Transfer SDK from the IBM API Hub. The SDK is a collection of binaries (command line utilities and a daemon to listen for transfer requests), configuration files, and language specific connectors.
2. Install the grpc dependencies from the appropriate package manager (pip, maven, gem, etc).
3. Launch the daemon and import the relevant programming language connector files to your project.
4. Instantiate an Aspera client by passing it the local port used by the daemon.
5. Create a `transfer_spec` containing all the information needed for the transfer:
    a. `icos` information:
        a. API key
        b. Service instance ID
        c. Target endpoint
        d. Bucket name
        e. Transfer direction
        f. Remote host (you find this by sending a GET request to a bucket with a `?faspConnectionInfo` query parameter)
        g. Assets for transfer (basically a set of file paths)
6. Pass the transfer specification and configuration info to a transfer request.

The following is an example using Python:

```
$ import random
import string

import grpc
import json
import os.path

from urllib3.connectionpool import xrange

import transfer_pb2 as transfer_manager
import transfer_pb2_grpc as transfer_manager_grpc


def run():
    # create a connection to the transfer manager daemon
    client = transfer_manager_grpc.TransferServiceStub(
        grpc.insecure_channel('localhost:55002'))

    # create file
    file_path = generate_source_file()

    # create transfer spec
    transfer_spec = {
        "session_initiation": {
            "icos": {
                "api_key": os.environ.get('IBMCLOUD_API_KEY'),
                "bucket": os.environ.get('IBMCLOUD_BUCKET'),
                "ibm_service_instance_id": os.environ.get('IBMCLOUD_COS_INSTANCE'),
                "ibm_service_endpoint": os.environ.get('IBMCLOUD_COS_ENDPOINT')
            }
        },
        "direction": "send",
        "remote_host": "https://ats-sl-dal.aspera.io:443",
        "title": "strategic",
        "assets": {
            "destination_root": "/aspera/file",
            "paths": [
                {
                    "source": file_path
                }
            ]
        }
    }
    transfer_spec = json.dumps(transfer_spec)

    # create a transfer request
    transfer_request = transfer_manager.TransferRequest(
        transferType=transfer_manager.FILE_REGULAR,
        config=transfer_manager.TransferConfig(),
        transferSpec=transfer_spec)

    # send start transfer request to transfer manager daemon
    transfer_response = client.StartTransfer(transfer_request)
    transfer_id = transfer_response.transferId
    print("transfer started with id {0}".format(transfer_id))

    # monitor transfer status
    for transfer_info in client.MonitorTransfers(
            transfer_manager.RegistrationRequest(
                filters=[transfer_manager.RegistrationFilter(
                    transferId=[transfer_id]
                )])):
        print("transfer info {0}".format(transfer_info))

        # check transfer status in response, and exit if it's done
        status = transfer_info.status
        if status == transfer_manager.FAILED or status == transfer_manager.COMPLETED:
            print("finished {0}".format(status))
            break


def generate_source_file(name='file'):
    with open(name, 'w') as file:
        # file.write('Hello World!')
        file.write(''.join(random.choice(string.ascii_lowercase) for i in xrange(10 ** 10)))
    return os.path.abspath(name)


if __name__ == '__main__':
    run()
```

Using IBM Cloud® Data Engine

Deprecated: IBM Cloud® Data Engine (formerly SQL Query) is now end of market. No new instances of Data Engine can be created. Existing instances can still be used until end of support. See Deprecation of Data Engine for more information.

IBM Cloud® Data Engine is a fully managed service that lets you run SQL queries (that is, `SELECT` statements) to analyze, transform, or clean up rectangular data using the full ANSI SQL standard.

Querying Object Storage with SQL Query

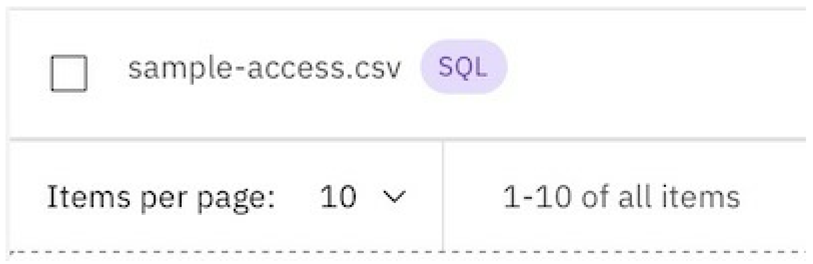
Tip: You can use SQL Query to create `SELECT` statements only; actions such as `CREATE`, `DELETE`, `INSERT`, and `UPDATE` are impossible.

Input data for your queries are read from ORC, CSV, JSON, or Parquet files located in one or more IBM Cloud Object Storage instances. Each query result is written by default to a CSV file in a Cloud Object Storage instance where you created the integration. But you can freely override and customize the format and Object Storage location as part of the SQL statement that you run.

Note: You can use a custom `INTO` clause of a `SELECT` statement to control where and how result data from a `SELECT` statement is written to IBM Cloud Object Storage.

Getting started using SQL Query `SELECT` statements from inside your instance is as easy as creating an integration. Objects of data formats that can be queried, as well as folders with multiple objects of a consistent format that can be queried (when shown in the "folders" view) are labeled as shown in Figure 1.

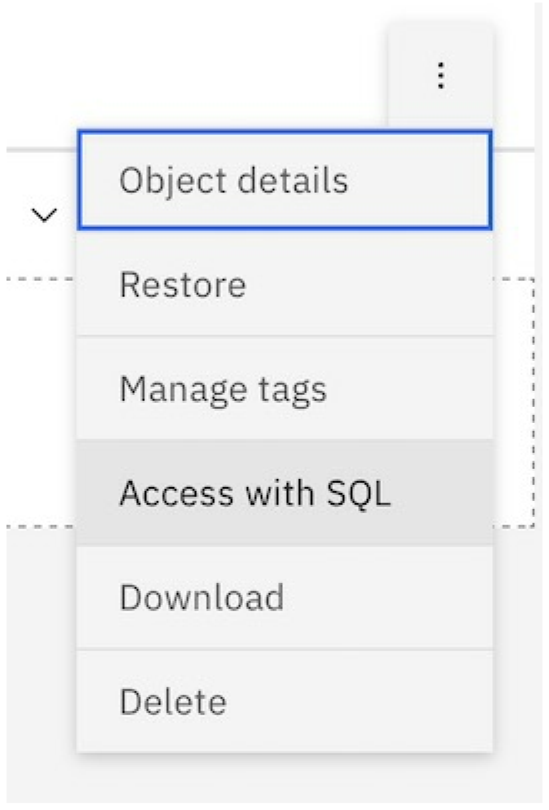
SQL label shows objects that can be queried



Tip: You can retrieve an SQL URL that can be queried for objects for a selected individual object (Object SQL URL) or for all objects currently displayed with an active prefix filter (Filtered SQL URL). You can use this URL inside the SQL statement as the table name.

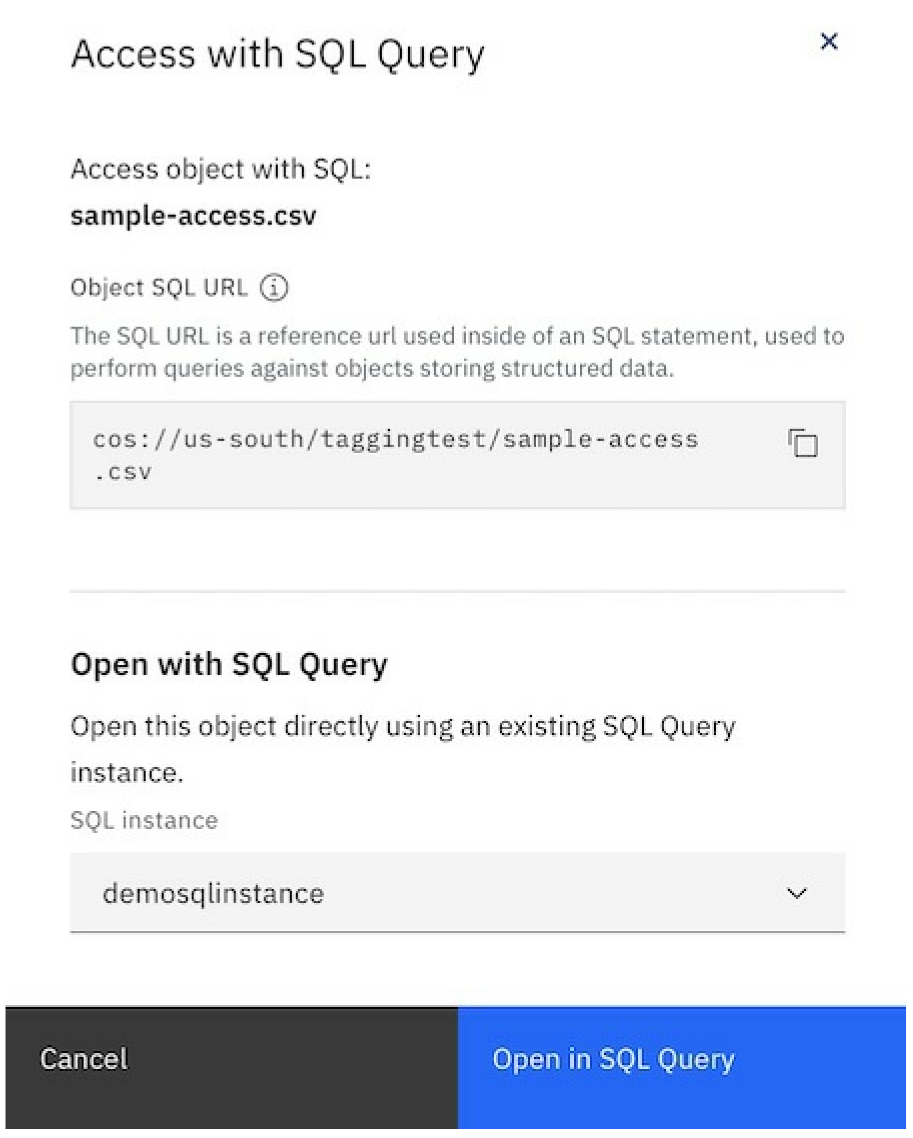
Figure 1 shows how to access your data using Data Engine. When you click on the ellipses at the end of a row of an object that you can query, you will see a menu where you can "Access with SQL" by selecting that option.

Access with SQL shows objects that can be queried



The panel shown in Figure 3 shows how to access your data using Data Engine. The location of your object appears in the panel for reference outside of the console. The instances to which you have access appear in the dropdown list in the panel. After you specify the instance, click on "Open in SQL Query" to launch your instance already pre-populated with a sample query written in the appropriate SQL.

Access with SQL launch panel


Access with SQL Query

Access object with SQL:
sample-access.csv

Object SQL URL

The SQL URL is a reference url used inside of an SQL statement, used to perform queries against objects storing structured data.

```
cos://us-south/taggingtest/sample-access
.csv
```

Open with SQL Query

Open this object directly using an existing SQL Query instance.

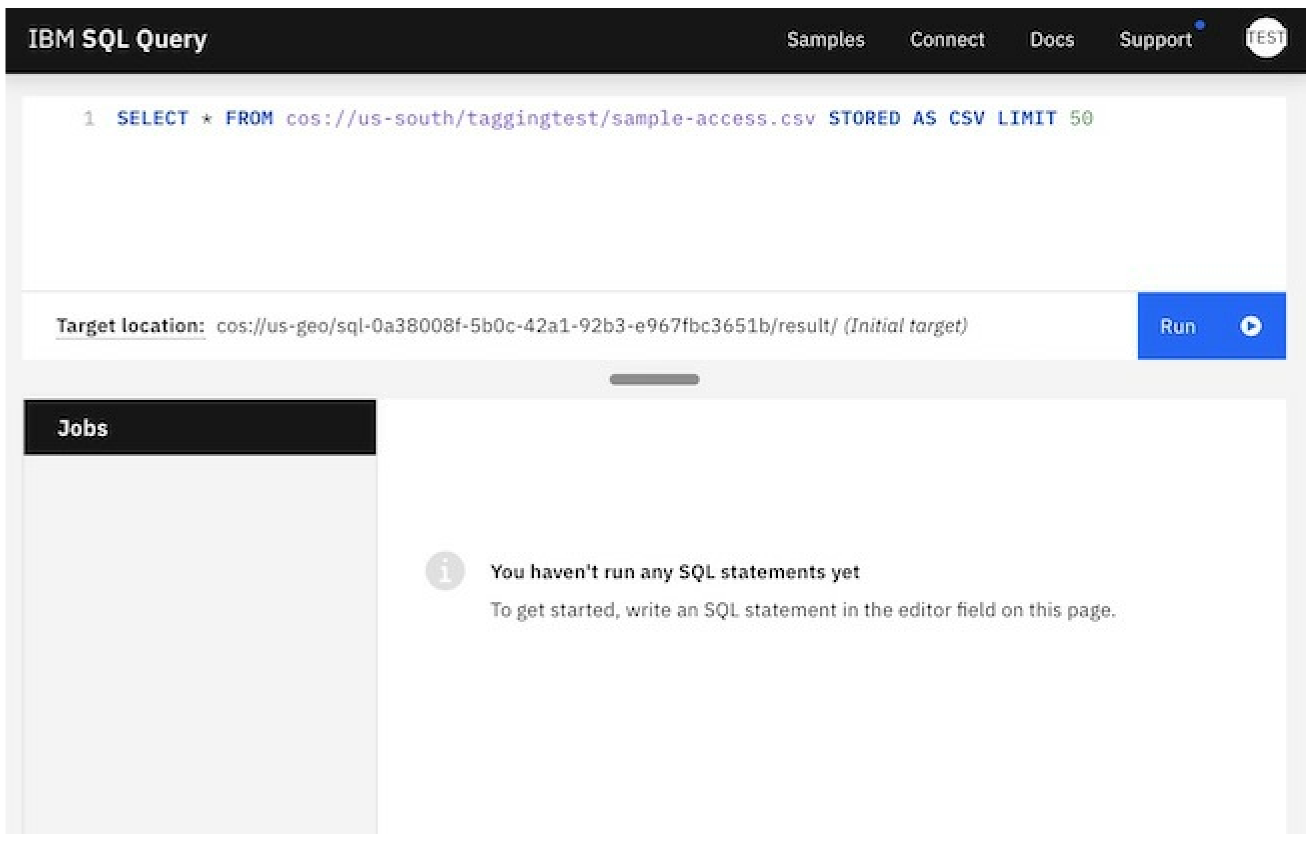
SQL instance: demosqlinstance

Important: Access is based on permissions, and you may wish to study more about authentication and access.

Getting Results

Figure 4 shows a sample SQL query you can modify. By pressing the "Run" button, the list below the query will populate with a new entry that links to your results. The results will be stored in the location shown beneath the query.

Access with SQL query window



```
SELECT * FROM cos://us-south/taggingtest/sample-access.csv STORED AS CSV LIMIT 50
```

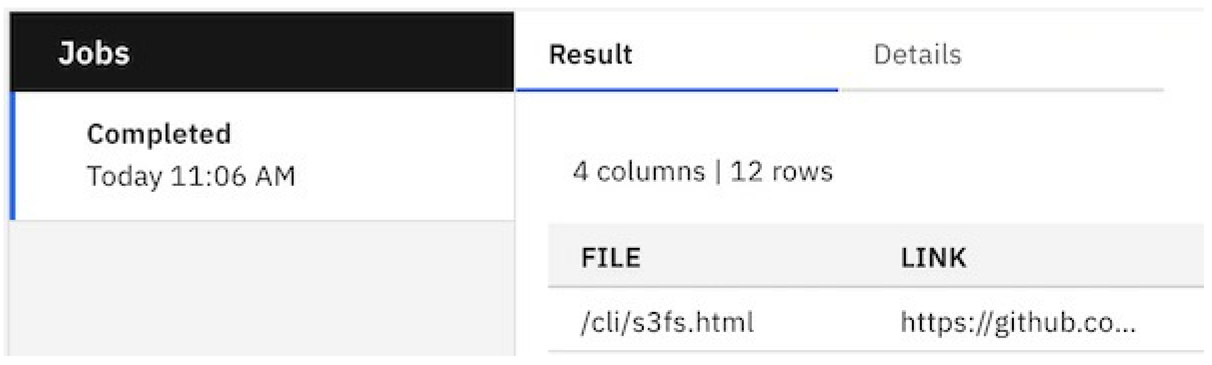
Target location: cos://us-geo/sql-0a38008f-5b0c-42a1-92b3-e967fbc3651b/result/ *(Initial target)*

You haven't run any SQL statements yet

To get started, write an SQL statement in the editor field on this page.

The entry representing the job of the `SELECT` statement run previously is shown in Figure 5. There are two tabs, "Results" and "Details," at the top of the list that allow you to switch between seeing the results and more detailed information.

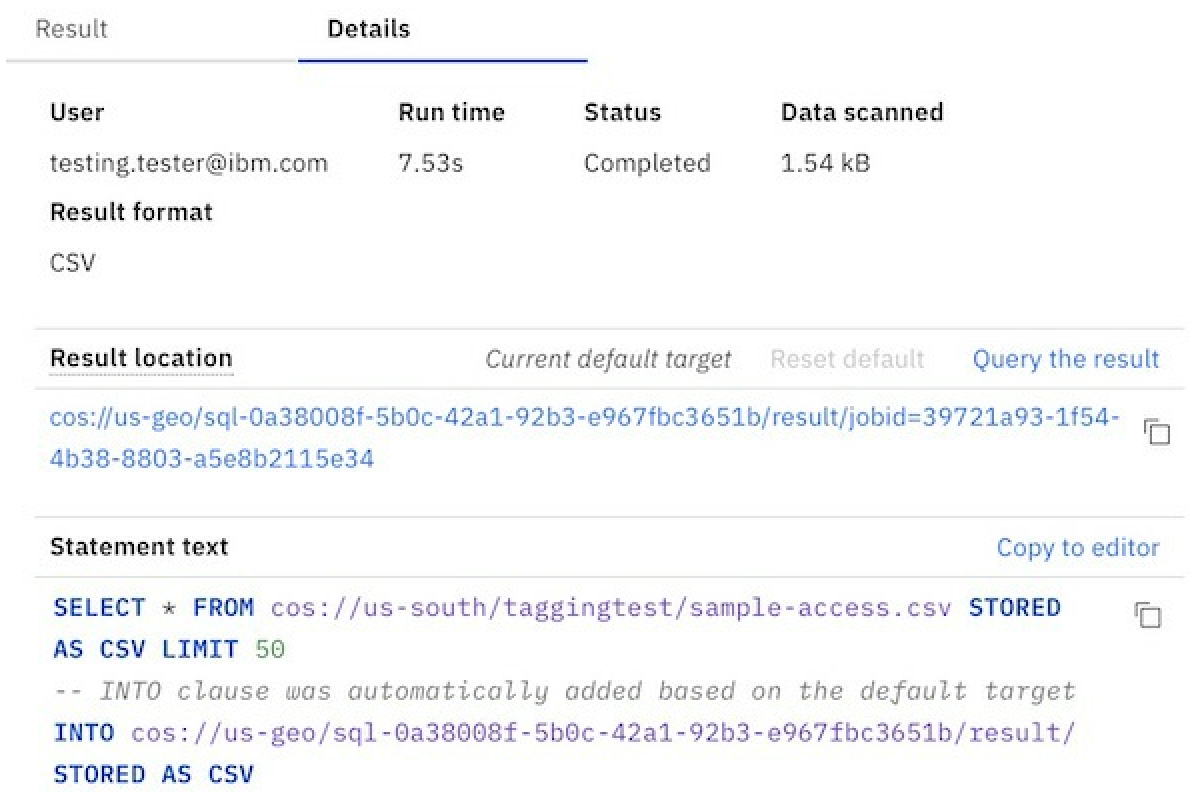
Access with SQL query jobs



FILE	LINK
/cli/s3fs.html	https://github.co...

The entry representing the details of running the `SELECT` statement run previously is shown in Figure 6.

Access with SQL query jobs



User	Run time	Status	Data scanned
testing.tester@ibm.com	7.53s	Completed	1.54 kB

Result format: CSV

Result location

cos://us-geo/sql-0a38008f-5b0c-42a1-92b3-e967fbc3651b/result/jobid=39721a93-1f54-4b38-8803-a5e8b2115e34

Statement text

```
SELECT * FROM cos://us-south/taggingtest/sample-access.csv STORED
AS CSV LIMIT 50
-- INTO clause was automatically added based on the default target
INTO cos://us-geo/sql-0a38008f-5b0c-42a1-92b3-e967fbc3651b/result/
STORED AS CSV
```

Next Steps

For more information on using Data Engine see the Data Engine documentation.

Using Cloud Functions

Deprecated: IBM Cloud® Functions is deprecated. Existing Functions entities such as actions, triggers, or sequences will continue to run, but as of 28 December 2023, you can't create new Functions entities. Existing Functions entities are supported until October 2024. Any Functions entities that still exist on that date will be deleted. For more information, see Deprecation overview.

With IBM Cloud® Functions, you can use your favorite programming language to write lightweight code that runs app logic in a scalable way. You can run code on-demand with HTTP-based API requests from applications or run code in response to IBM Cloud services and third-party events, like updates made to a bucket. The Function-as-a-Service (FaaS) programming platform is based on the open source project Apache OpenWhisk.

Note: This feature is not currently supported in Object Storage for Satellite. Learn more.

Using Object Storage as an event source

Cloud Functions is an event-driven compute platform (also referred to as Serverless computing). Actions (small bits of code) run in response to triggers (some category of event), and rules associate certain actions with certain triggers. Configure IBM Cloud® Object Storage to be an event source, and anytime an object in a particular bucket is written or deleted an action is triggered. You can further tailor the changes feed to only corral events for objects which match a particular prefix or suffix.

1. Set the option to allow Cloud Functions access to listen for changes that are made to your bucket. This involves creating a service-to-service authorization, and uses the new Notifications Manager IAM role.
2. Then, create a trigger to respond to the changes feed.
3. Then use the IBM Cloud Object Storage package to bind credentials and easily script common tasks.

For more information about using Cloud Functions with IBM Cloud Object Storage, see the Functions documentation.

Important: It is not possible to use a bucket with a firewall enabled as an event source for IBM Cloud® Functions actions.

Next Steps

Be sure to identify the appropriate [region and endpoint] for your service. Then, verify your operations with specific [testing].

Serving static websites

A new hosted static website can be created with IBM Cloud® Object Storage in minutes using [this simple tutorial]. This topic contains the details and some advanced configuration options for hosting static websites.

Overview

Modern web development requires modern tools and secure infrastructure. Static websites represent the latest developments in high-availability, SEO improvement, and increased security. While covering every available option is beyond the scope of this hands-on overview, the ease of serving static content on IBM Cloud Object Storage allows for many possible strategies.

Hosted static websites focus on the content your users desire: information and media. By removing the administration of web servers like Apache or Nginx, management of your website focuses directly on content, from generation to deployment.

Static content differs substantially from dynamic web content. However, if you don't need to generate dynamic content on the web or if your workflow results in content saved to a fixed form, then the hosted static solution featured here presents the best choice.

Capabilities

Creating static website hosting in IBM Cloud Object Storage can be accomplished with [cURL], as well as libraries for [Java], [Go], [Python], and [NodeJS]. In addition, S3 compatibility means that the [AWS CLI] can also be used to define static website functionality from the command line. Also, creating and configuring a new hosted static website solution can be created using a GUI in the [Console] just by adding the option for Static Website when creating a bucket.

Basic Configuration

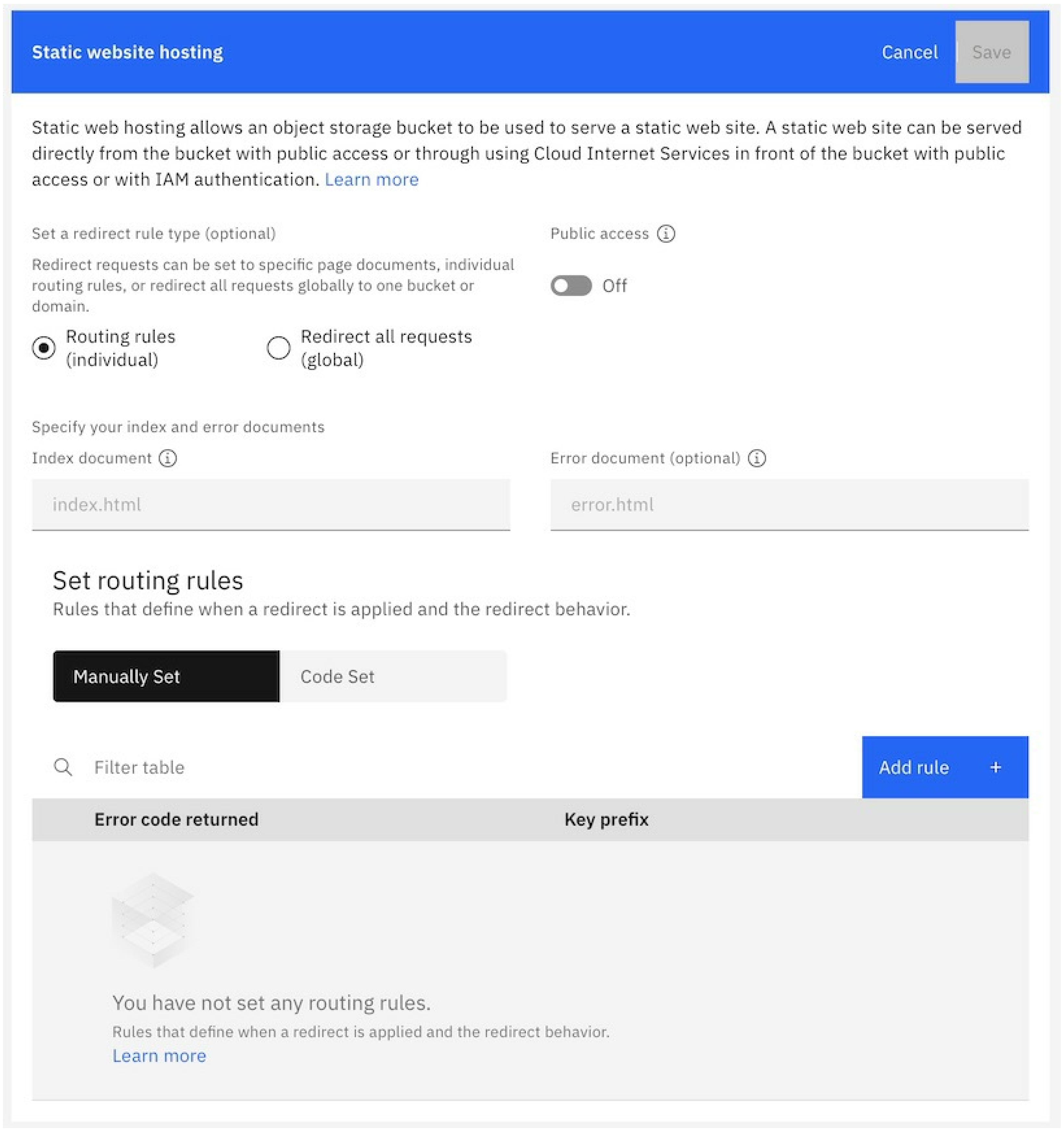
Hosting a static website on IBM Cloud® Object Storage starts with [creating a bucket] and configuring it for public access. Then, [upload] your website content to your bucket. Finally, [configure the website] to use your documents as an index for the site and to potentially display errors.

At minimum, your configuration should consist of a required index document for visitors to view by default, usually written in HTML and named `index.html`. An optional error document can help your visitors stay on track when they stray. Of course, you can always try for yourself using [this tutorial].

Advanced Configuration

When you create and configure a new hosted static website, you may also wish to use IBM Cloud Internet Services to configure more advanced options including [routing rules for your domain]. But you don't even have to go further than configuring your bucket during creation to start customizing your new site.

Initial configuration options



Static web hosting allows an object storage bucket to be used to serve a static web site. A static web site can be served directly from the bucket with public access or through using Cloud Internet Services in front of the bucket with public access or with IAM authentication. Learn more

Set a redirect rule type (optional)

Redirect requests can be set to specific page documents, individual routing rules, or redirect all requests globally to one bucket or domain.

[✓ Routing rules (individual)] Redirect all requests (global)

Public access: Off

Specify your index and error documents

Index document: index.html

Error document (optional): error.html

Set routing rules

Rules that define when a redirect is applied and the redirect behavior.

[✓ Manually Set] Code Set

Error code returned	Key prefix

You have not set any routing rules.

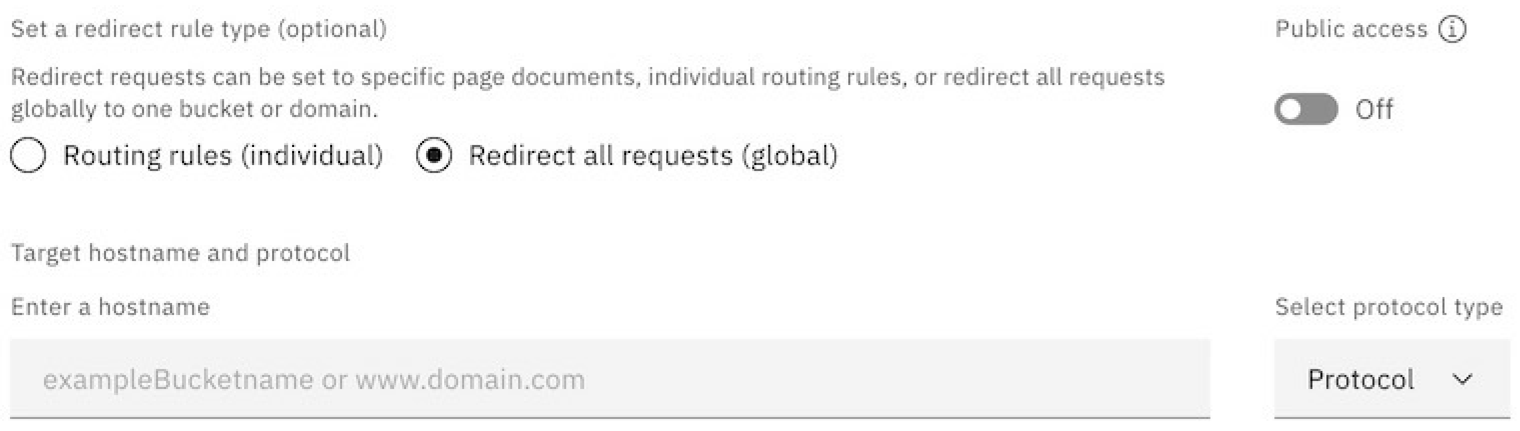
Rules that define when a redirect is applied and the redirect behavior.

Learn more

Routing

Routing gives you control over the requests coming from your visitors. For example, you could globally redirect all your traffic from using one protocol to another, like replacing HTTP with the secure HTTPS. Or, you can create individual rules that process incoming requests for specific files and provide responses to your visitors based on the rules you define.

Global routing rule



Set a redirect rule type (optional)

Redirect requests can be set to specific page documents, individual routing rules, or redirect all requests globally to one bucket or domain.

Routing rules (individual) [✓ Redirect all requests (global)]

Public access: Off

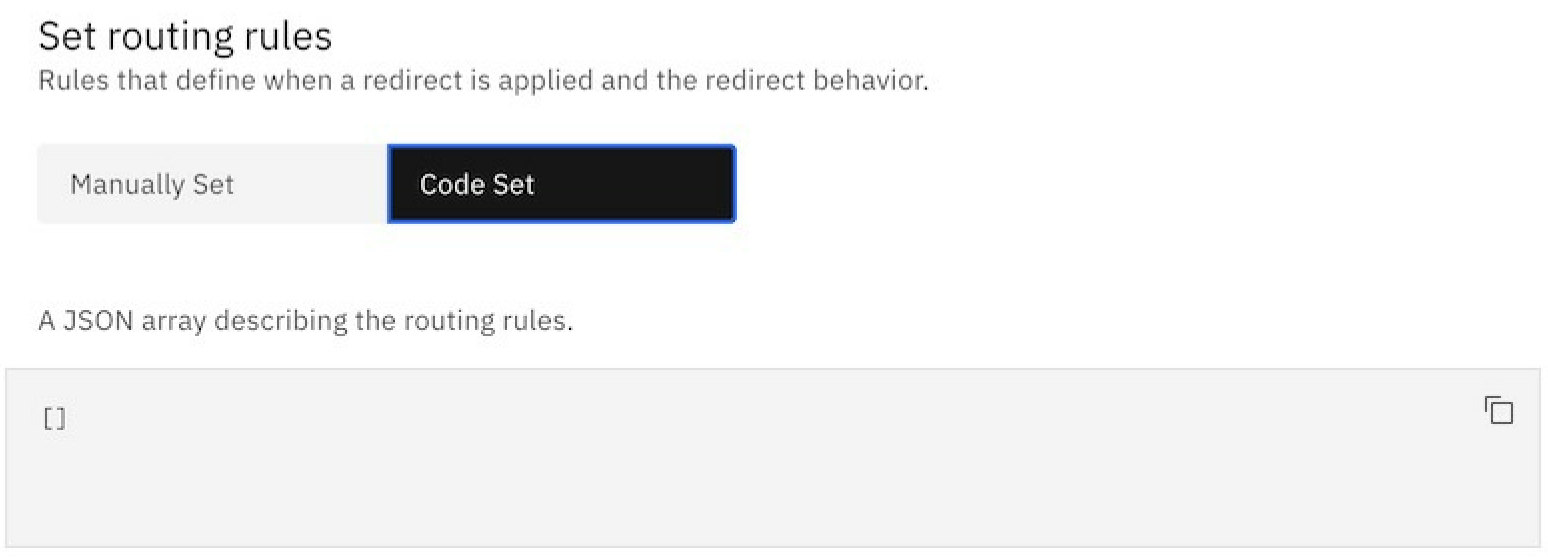
Target hostname and protocol

Enter a hostname: exampleBucketname or www.domain.com

Select protocol type: Protocol

If you already have a hosted static website that you wish to migrate, you can bring a set of the routing rules that you have already set and import the set as code. The input shown in Figure 3 requires a JSON array formatted for the website configuration rules.

Import configuration as code


Set routing rules

Rules that define when a redirect is applied and the redirect behavior.

A JSON array describing the routing rules.

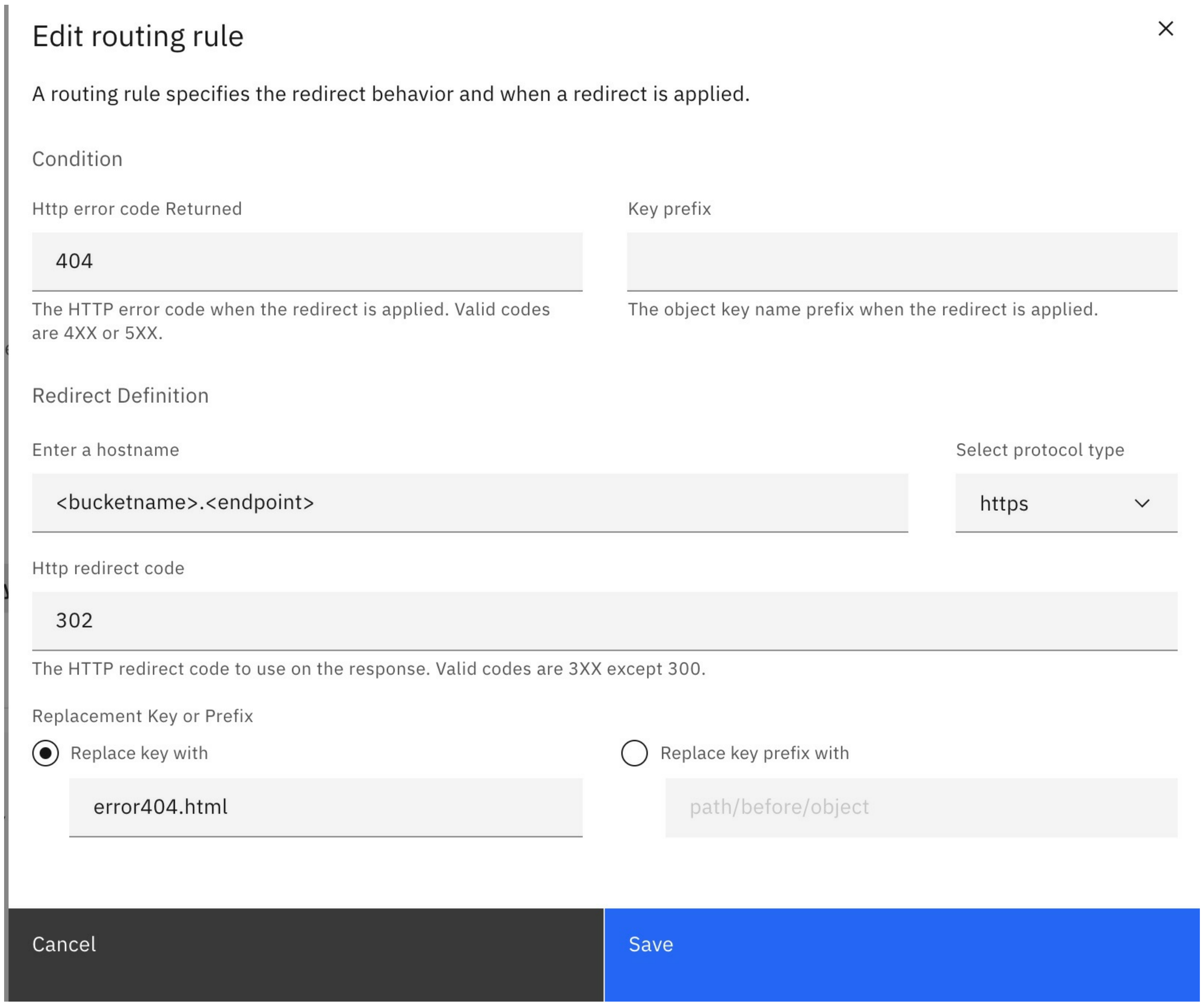
```
[]
```

An example of JSON code exemplifies the possibilities. The following shows a rule that redirects visitors from missing pages or possible malformed request resulting in a `404` error code and redirecting the visitor to a specific error page. The JSON can contain multiple objects representing the definition of the rules as needed.

```
$ [
  {
    "Condition": {
      "HttpErrorCodeReturnedEquals": "404"
    },
    "Redirect": {
      "HostName": "<bucketname>.<endpoint>",
      "HttpRedirectCode": "302",
      "Protocol": "https",
      "ReplaceKeyWith": "error404.html"
    }
  }
]
```

The same rule codified previously can be added as an individual rule using the Console, and shown in Figure 3.

Add individual rules


Edit routing rule

A routing rule specifies the redirect behavior and when a redirect is applied.

Condition

Http error code Returned: 404

The HTTP error code when the redirect is applied. Valid codes are 4XX or 5XX.

Key prefix:

The object key name prefix when the redirect is applied.

Redirect Definition

Enter a hostname: <bucketname>.<endpoint>

Select protocol type: https

Http redirect code: 302

The HTTP redirect code to use on the response. Valid codes are 3XX except 300.

Replacement Key or Prefix

[✓ Replace key with] error404.html

Replace key prefix with: path/before/object

IBM Cloud Internet, Domain, and Delivery Services

One of the benefits of using IBM Cloud Internet Services pertains to setting up your own domains. A "domain" is part of the overall web address, consisting of a Top Level Domain (TLD) and one or more unique words separated by dots, like `example.com` where the TLD is `com`. You can choose to skip this step, but if your DNS records are not configured properly using CIS (or other service providing domain name resolution), it might leave all or part of your website inaccessible.

Static websites are meant to be fast and secure. Serving up static content is easy with the right tools that deliver the content to your customers. Many deployment tools have built-in support for CDN support. Getting started configuring your domains using IBM Cloud® Internet Services. When creating redirect rules, you will be adding a `CNAME`, a "canonical (domain) name", or alias. Just like files on an operating system can have an alias for convenience, your hosted static website can be just as convenient.

The process for delivering static content through dedicated networks starts with this overview of CDN options. Content Delivery moves your static content closer to your customer's own location, extending your reach without having to manage copies of your content.

Endpoints for hosting static website content

The following tables match most of the regions, locations, and type of connections used in IBM Cloud Object Storage to the new specific endpoints used for sourcing and testing hosted static websites. For tethered endpoints not listed here, find more information on using tethered endpoints.

Regional endpoints

Region	Hosted Static Website Endpoint
US South	`s3-web.us-south.cloud-object-storage.appdomain.cloud`
US East	`s3-web.us-east.cloud-object-storage.appdomain.cloud`
EU United Kingdom	`s3-web.eu-gb.cloud-object-storage.appdomain.cloud`

EU Germany	`s3-web.eu-de.cloud-object-storage.appdomain.cloud`
AP Australia	`s3-web.au-syd.cloud-object-storage.appdomain.cloud`
AP Tokyo	`s3-web.jp-tok.cloud-object-storage.appdomain.cloud`
AP Osaka	`s3-web.jp-osa.cloud-object-storage.appdomain.cloud`

Regional Endpoints

Region	Hosted Static Website Endpoint
US South	`s3-web.private.us-south.cloud-object-storage.appdomain.cloud`
US East	`s3-web.private.us-east.cloud-object-storage.appdomain.cloud`
EU United Kingdom	`s3-web.private.eu-gb.cloud-object-storage.appdomain.cloud`
EU Germany	`s3-web.private.eu-de.cloud-object-storage.appdomain.cloud`
AP Australia	`s3-web.private.au-syd.cloud-object-storage.appdomain.cloud`
AP Tokyo	`s3-web.private.jp-tok.cloud-object-storage.appdomain.cloud`
AP Osaka	`s3-web.private.jp-osa.cloud-object-storage.appdomain.cloud`

Regional Endpoints

Region	Hosted Static Website Endpoint
US South	`s3-web.direct.us-south.cloud-object-storage.appdomain.cloud`
US East	`s3-web.direct.us-east.cloud-object-storage.appdomain.cloud`
EU United Kingdom	`s3-web.direct.eu-gb.cloud-object-storage.appdomain.cloud`
EU Germany	`s3-web.direct.eu-de.cloud-object-storage.appdomain.cloud`
AP Australia	`s3-web.direct.au-syd.cloud-object-storage.appdomain.cloud`
AP Tokyo	`s3-web.direct.jp-tok.cloud-object-storage.appdomain.cloud`
AP Osaka	`s3-web.direct.jp-osa.cloud-object-storage.appdomain.cloud`

Regional Endpoints

Cross Region endpoints

Region	Hosted Static Website Endpoint
US Cross Region	`s3-web.us.cloud-object-storage.appdomain.cloud`
EU Cross Region	`s3-web.eu.cloud-object-storage.appdomain.cloud`
AP Cross Region	`s3-web.ap.cloud-object-storage.appdomain.cloud`

Cross Region Endpoints

Region	Hosted Static Website Endpoint
US Cross Region	`s3-web.private.us.cloud-object-storage.appdomain.cloud`

EU Cross Region	`s3-web.private.eu.cloud-object-storage.appdomain.cloud`
AP Cross Region	`s3-web.private.ap.cloud-object-storage.appdomain.cloud`

Cross Region Endpoints

Region	Hosted Static Website Endpoint
US Cross Region	`s3-web.direct.us.cloud-object-storage.appdomain.cloud`
EU Cross Region	`s3-web.direct.eu.cloud-object-storage.appdomain.cloud`
AP Cross Region	`s3-web.direct.ap.cloud-object-storage.appdomain.cloud`

Cross Region Endpoints

Single site endpoints

Location	Hosted Static Website Endpoint
Amsterdam, Netherlands	`s3-web.ams03.cloud-object-storage.appdomain.cloud`
Chennai, India	`s3-web.che01.cloud-object-storage.appdomain.cloud`
Mexico City, Mexico	`s3-web.mex01.cloud-object-storage.appdomain.cloud`
Milan, Italy	`s3-web.mil01.cloud-object-storage.appdomain.cloud`
Montrèal, Canada	`s3-web.mon01.cloud-object-storage.appdomain.cloud`
Paris, France	`s3-web.par01.cloud-object-storage.appdomain.cloud`
San Jose, US	`s3-web.sjc04.cloud-object-storage.appdomain.cloud`
São Paulo, Brazil	`s3-web.sao01.cloud-object-storage.appdomain.cloud`
Singapore	`s3-web.sng01.cloud-object-storage.appdomain.cloud`

Single Data Center Endpoints

Location	Hosted Static Website Endpoint
Amsterdam, Netherlands	`s3-web.private.ams03.cloud-object-storage.appdomain.cloud`
Chennai, India	`s3-web.private.che01.cloud-object-storage.appdomain.cloud`
Mexico City, Mexico	`s3-web.private.mex01.cloud-object-storage.appdomain.cloud`
Milan, Italy	`s3-web.private.mil01.cloud-object-storage.appdomain.cloud`
Montrèal, Canada	`s3-web.private.mon01.cloud-object-storage.appdomain.cloud`
Paris, France	`s3-web.private.par01.cloud-object-storage.appdomain.cloud`
San Jose, US	`s3-web.private.sjc04.cloud-object-storage.appdomain.cloud`
São Paulo, Brazil	`s3-web.private.sao01.cloud-object-storage.appdomain.cloud`
Singapore	`s3-web.private.sng01.cloud-object-storage.appdomain.cloud`

Single Data Center Endpoints

Location	Hosted Static Website Endpoint
Amsterdam, Netherlands	`s3-web.direct.ams03.cloud-object-storage.appdomain.cloud`
Chennai, India	`s3-web.direct.che01.cloud-object-storage.appdomain.cloud`
Mexico City, Mexico	`s3-web.direct.mex01.cloud-object-storage.appdomain.cloud`
Milan, Italy	`s3-web.direct.mil01.cloud-object-storage.appdomain.cloud`
Montrèal, Canada	`s3-web.direct.mon01.cloud-object-storage.appdomain.cloud`
Paris, France	`s3-web.direct.par01.cloud-object-storage.appdomain.cloud`
San Jose, US	`s3-web.direct.sjc04.cloud-object-storage.appdomain.cloud`
São Paulo, Brazil	`s3-web.direct.sao01.cloud-object-storage.appdomain.cloud`
Singapore	`s3-web.direct.sng01.cloud-object-storage.appdomain.cloud`

Single Data Center Endpoints

Next steps

Making the most of modern web development requires modern tools and secure infrastructure, but shouldn't be a barrier to the success of your projects. If you haven't already tried the tutorial, check out for yourself how hosting a static website can work for you.

Domain routing for static website hosting

A static website hosted with IBM Cloud® Object Storage can be configured using IBM Cloud Internet Services. Configuring routing rules for domains hosted in IBM Cloud Object Storage will be explored in this advanced "how to."

Important: These instructions are subject to change and are provided here for review.

Overview

When hosting static website content on IBM Cloud Object Storage, you can configure how the public accesses your site by specifying a custom domain. Using IBM Cloud Internet Services configures Object Storage to modify the HTTP `host` header in response to public requests of your content. For this example, we will use `example.com` as the domain configured in Cloud Internet Services (CIS), and a desired subdomain, `web` that you wish your public visitors to view as `web.example.com`.

Before you start

Prerequisites:

- An account for the IBM Cloud Platform
- An instance of Cloud Internet Services (CIS) with an applicable plan and permissions
- An instance of Object Storage with a bucket configured as a hosted static website
- An Internet domain managed through IBM Cloud Internet Services

Note: These instructions require an account with the correct plan in order to access the services as described.

IBM Cloud Internet, Domain, and Delivery Services

Create a Page Rule to target your bucket

Creating a "Page Rule" in your instance of IBM Cloud Internet Services will take several steps, and require your endpoint information

1. Select Performance from the Navigation

2. Select the Page rules Tab from the options.

3. In the table of rules (empty if this is the first), Select the "Create rule" button.

4. In the URL match field Enter `<sub-domain>.<custom-domain>/*`. For example, `web.example.com/*`.

5. Select from the options for "Rule Behavior Setting" the "Host Header Override." For your custom bucket hosting your static website content as a sub-domain and endpoint: `<bucket-name>.s3-web.<bucket-region>.cloud-object-storage.appdomain.cloud`. For example, using `web-example-com` as the name of the bucket hosting your static website, the example would appear as: `web-example-com.s3-web.us-east.cloud-object-storage.appdomain.cloud`.

   Note: You can find this information in your bucket configuration, or Quick View in the Console.

6. When you have confirmed your configuration options, select "Create."

7. Next, you will create the DNS CNAME record to forward traffic to your content in IBM Cloud Object Storage.

Create a domain alias to proxy your content

After you have directed your visitors to the right location using a "Page Rule," you will want to create an alias to guide your visitors to the location. For this example, we want to send your visitors to your new subdomain `web` to the existing domain, `example.com` that will point to

1. Select Reliability from the Navigation

2. Select the DNS Tab from the options.

3. Add a new DNS record, substituting your configuration for the exemplified values shown. The desired subdomain should be added in the "name" field. in this example, we used `web` as a new subdomain value. The "alias domain name" is the same as entered earlier, which in this example comprised a bucket name followed by a dot and then the endpoint, for example, `web-example-com.s3-web.us-east.cloud-object-storage.appdomain.cloud`.

   - Type: CNAME
   - Name: `<sub-domain>`
   - TTL: Automatic
   - Alias Domain Name: `<bucket-name>.s3-web.<bucket-region>.cloud-object-storage.appdomain.cloud`

4. Click "Add" to save the DNS entry when you've completed the configuration.

5. In the table of rules where your new entry appears, enable the Proxy option as "on."

To test the rule you just created, allow some time for the configuration to propagate. Then, use a browser to visit the subdomain exemplified by `web.example.com` to validate the settings.

Next steps

Learn more about IBM Cloud Internet Services, or jump right in using Cloud Internet Services (CIS) to get started managing your presence on the Internet.

Using IBM Cloud Satellite

Deprecation Object Storage for Satellite

IBM Cloud continues to evaluate its service offerings periodically, keeping in perspective our client requirements and market direction. As a result, as of December 16, 2024, the IBM Cloud® Object Storage for IBM Cloud Satellite® offering is being deprecated.

Important dates

Stage	Date	Description
Deprecation announcement	16 December 2024	Announcement of Object Storage for Satellite deprecation. Existing instances are serviced as per terms of offering.
End of Marketing	13 January 2025	No new requests for Object Storage for Satellite can be created or submitted. Existing instances are serviced as per terms of offering.
End of Support	16 December 2025	Support for this service ends on this date, after which all instances will be permanently disabled and inaccessible, and no new support cases can be opened.

Important dates

Deprecation details

- The service is removed from the IBM Cloud console on 13 January 2025, and no new instances can be created after that date. Your existing instances that were created before this date will continue to run as planned.
- This deprecation means that support, including updates and technical support for the product, is no longer available, effective 16 December 2025.
- Any remaining instances will be permanently disabled and inaccessible as of 16 December 2025, including any user data.
- No support cases can be opened after 16 December 2025.

Next steps for current users

Please complete your in-progress activities promptly and ensure the transition to another object storage service on IBM Satellite (as offered through alternatives). If you need help, please open a service request with IBM Cloud support.

Migrating to an alternative service

Clients seeking an Object Storage solution on Satellite must transition to alternative options.

1. Connecting to S3-compatible Object Storage using the CSI driver .
2. Using Object Storage native to ROKS (for locality, disconnected mode) using ODF.
3. For larger deployments, utilize our on-premises Cloud Object Storage solution .

Help

If you have questions, comments, or concerns, you can contact the team through IBM Cloud Support.

About Object Storage for Satellite

Deprecated: IBM Cloud continues to evaluate its service offerings periodically, keeping in perspective our client requirements and market direction. As a result, as of December 16, 2024, the Object Storage for Satellite offering is being deprecated. For more information, see Deprecation overview.

IBM Cloud Object Storage for IBM Cloud Satellite offers users the flexibility to run a managed Object Storage service on client-owned on-premises infrastructure, edge locations or third-party public cloud infrastructure.

Important: This introductory offering of Object Storage for Satellite is limited in capabilities and will be expanded on in the future. Keep in mind that not all APIs or connected services may work in the same fashion as Object Storage on IBM Cloud.

Essentially, provisioning an instance of Object Storage for Satellite provides the same familiar interfaces of IBM Cloud Object Storage outside of IBM Cloud.

Note: Object Storage is integrated into Satellite in three different ways: configuration data and backup storage for the Satellite instance itself, as a persistent volume that allows for file-like access, and as a local instance of an IBM Cloud Object Storage service instance. This documentation focuses on the latter - setting up and accessing an instance of Object Storage running on Satellite hardware.

Typical use cases of Object Storage for Satellite

Low latency workloads that need to be run in close proximity to on-premises data and applications including workloads running on factory floors for automated operations in manufacturing, real-time patient diagnosis, and media streaming.

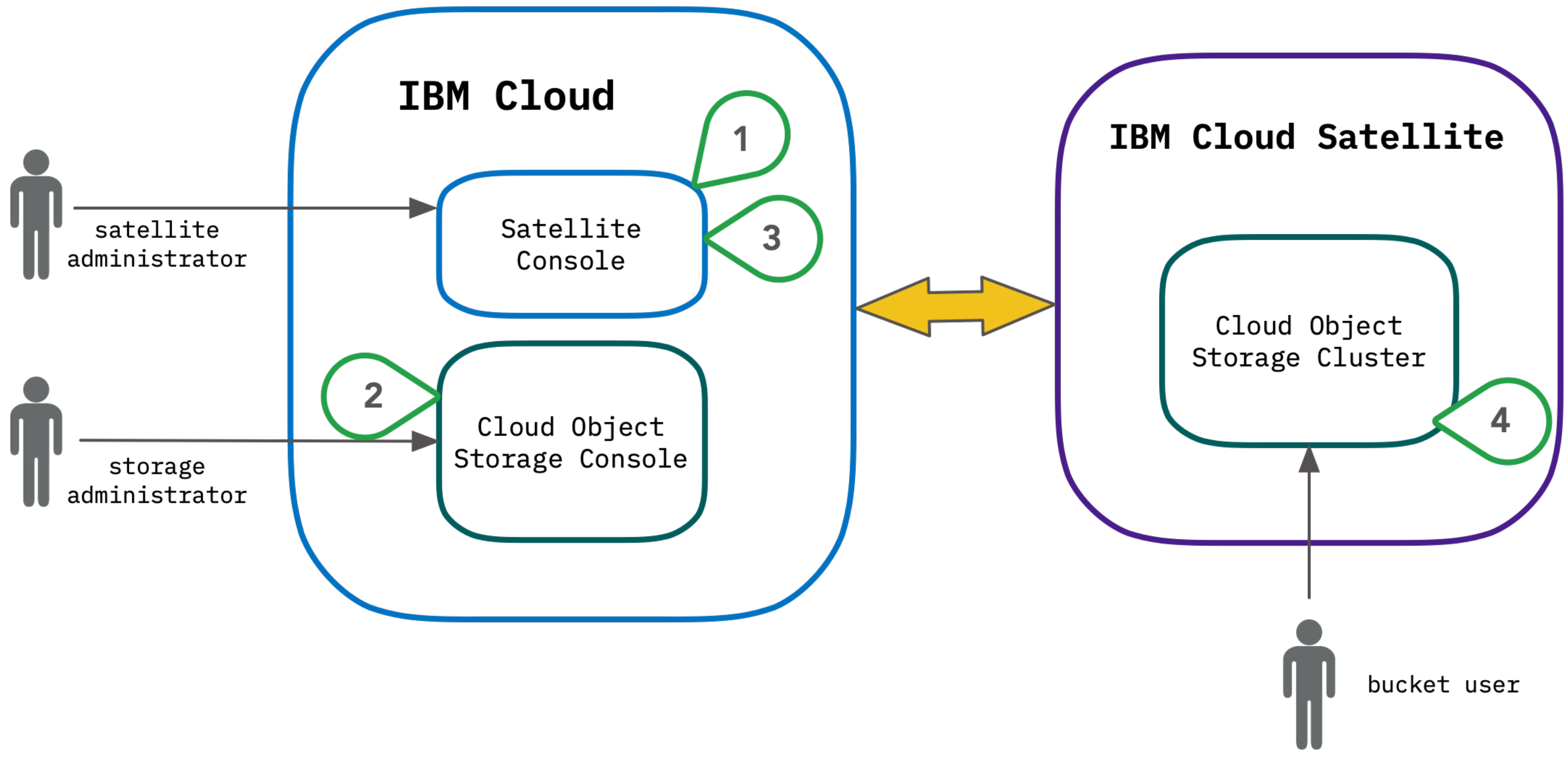
Data residency requirements or those in regulated industries that need to securely store and process customer data that needs to remain on-premises or in locations where there is no public Cloud Object Storage service.

Edge or IOT applications that collect and process data on the edge of network for new workloads from devices and users such as data collection and processing, location-based media, autonomous vehicle data, analytics, and machine data controls for manufacturing.

Hybrid workloads that require management of data between on-premises infrastructure, edge, public cloud or any multi-cloud installation.

How Object Storage for Satellite works

Object Storage for Satellite Architecture



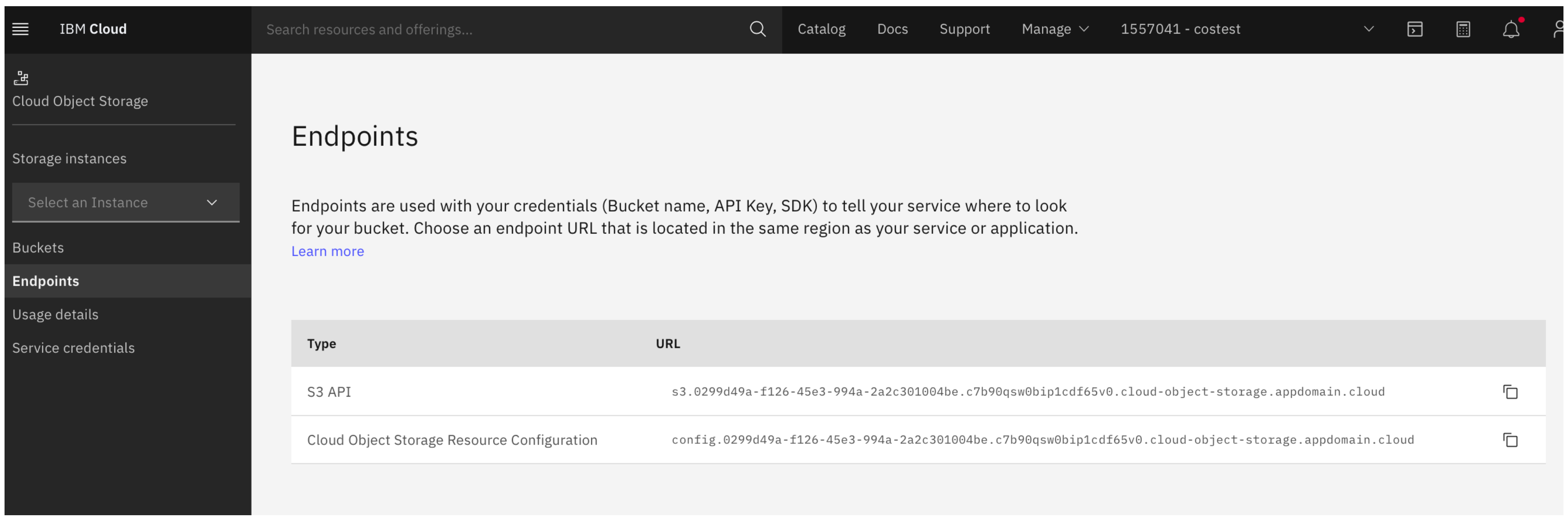
1. A Satellite administrator needs to configure a new "Location" using the Satellite console and assigns hosts for the Satellite Control Plane.
2. After the new location is created and accessible, an Object Storage administrator provisions the Object Storage instance in the new location.
3. The Satellite administrator assigns the appropriate hosts and block storage to the new Object Storage for Satellite cluster.
4. The new instance is available for both Object Storage bucket configuration and data operations.

Connecting to Object Storage for Satellite

In order to interact with object storage, a client makes API calls to a *service endpoint*. In a Satellite configuration, these should not be confused with *link endpoints* which are used for communication between services.

The *service endpoint* that is used for reading and writing data typically takes the form of `https://s3.{cos-instance-uuid}.{location-id}.cloud-object-storage.appdomain.cloud` and can be found under the **Endpoints** section of the object storage console.

Object Storage for Satellite Endpoints



Endpoints

Endpoints are used with your credentials (Bucket name, API Key, SDK) to tell your service where to look for your bucket. Choose an endpoint URL that is located in the same region as your service or application.

Type	URL
S3 API	s3.0299d49a-f126-45e3-994a-2a2c301004be.c7b90qsw0bip1cdf65v0.cloud-object-storage.appdomain.cloud
Cloud Object Storage Resource Configuration	config.0299d49a-f126-45e3-994a-2a2c301004be.c7b90qsw0bip1cdf65v0.cloud-object-storage.appdomain.cloud

Note: Keep in mind that requests made to Object Storage for Satellite infrastructure must originate within the satellite location as the service endpoint may not be accessible from the outside of that location.

What features are currently supported?

- IBM Cloud IAM access policies
- Object Expiration
- Object Versioning
- Object Tagging
- Static Web hosting
- Key Protect managed encryption

Important: Any Key Protect instances must be in IBM Cloud and must be located in the same IBM Cloud region from where the Satellite location is managed.

Important: Activity Tracking events are produced for service instance creation and deletion, but not any actions specific to object storage, such as listing buckets or reading/writing data.

Other features that are currently not supported (such as Activity Tracking, Metrics Monitoring, Compliance, Security and Compliance Center) will be added in the future.

Provisioning Object Storage for Satellite

Deprecated: IBM Cloud continues to evaluate its service offerings periodically, keeping in perspective our client requirements and market direction. As a result, as of December 16, 2024, the Object Storage for Satellite offering is being deprecated. For more information, see Deprecation overview.

You can provision Object Storage for Satellite using the IBM Cloud console.

Before you begin

Before deploying Object Storage in a Satellite location, you must first deploy a Satellite location with sufficient computing hosts and raw block storage allocated for provisioning Object Storage.

Note: Object Storage for Satellite only supports RHEL8.

Object Storage capacity	Raw storage required	Minimum host requirements
Small (12 TB)	18 TB	9 nodes of 4 vCPU and 16 GiB memory
Medium (24 TB)	36 TB	9 nodes of 4 vCPU and 16 GiB memory
Large (48 TB)	72 TB	9 nodes of 4 vCPU and 16 GiB memory

Extra Large (96 TB)	144 TB	18 nodes of 4 vCPU and 16 GiB memory

For more information on configuring hosts for storage, see the Satellite documentation.

Unlike cloud storage which scales elastically, there may be negative performance impacts when an instance gets near capacity. Workloads that demand higher performance may benefit from the additional computing power provided by the Extra Large plan, regardless of total storage required.

Tip: When provisioning block storage, is recommended to use a "Silver" storage class at a minimum to ensure adequate performance.

Configure a satellite location

1. Follow the documentation to create a new Satellite location with the necessary hosts and storage resources.
2. Grant the necessary service authorizations.
   a. Configure your IAM Authorizations under the Manage tab.
   b. Choose the **Authorizations** tab from the left hand menu.
   c. Click the **create** button to create an authorization that will allow a service instance access to another service instance. The source service is the service that is granted access to the target service. The roles you select define the level of access for this service. The target service is the service you are granting permission to be accessed by the source service based on the assigned roles.
   d. In the **Source Service** field, select **Cloud Object Storage**.
   e. In the **Target Service** field, select **Satellite**.
      a. Select all options: *Satellite Cluster Creator*, *Satellite Link Administrator*, *Satellite Link Source Access* Controller
   f. Then **Authorize**.

Provision an object storage service instance

1. Log in to the console.
2. Navigate to the catalog, by clicking **Catalog** in the navigation bar.
3. Look for the **Object Storage** tile in the storage section and select it.
4. Select **Satellite** from the "Choose an Infrastructure" section.
5. Choose an existing Satellite location.
6. Choose a capacity for your new Object Storage instance.
7. Click **Create** and you're automatically redirected to your new instance.

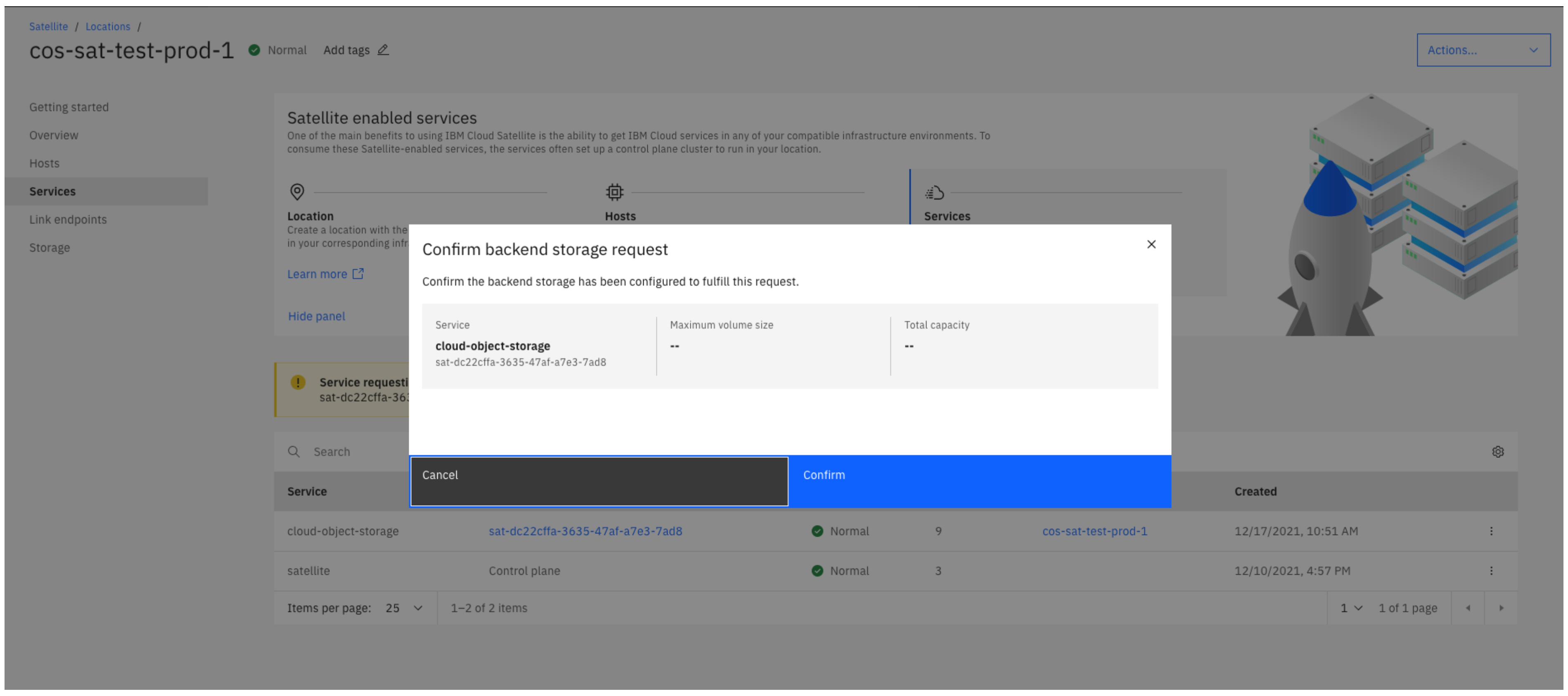
Assign hosts and storage to object storage cluster (using Satellite Storage UI)

Important: To access the Storage UI for Satellite, you must be added to the allowlist. Contact IBM to learn more.

If the location chosen for the new instance of Object Storage for Satellite was correctly configured with the required hosts and storage available, they will be automatically queued for assignment. This assignment requires confirmation from a Satellite administrator.

1. Log in to the console.
2. Navigate to Satellite, by clicking **Satellite** > **Locations** in the navigation bar.
3. Select the **Services** tab.
4. Look for the confirmation pop-up and approve the assignment.

Confirming host and storage assignment.



cos-sat-test-prod-1 Normal Add tags

Satellite enabled services

One of the main benefits to using IBM Cloud Satellite is the ability to get IBM Cloud services in any of your compatible infrastructure environments. To consume these Satellite-enabled services, the services often set up a control plane cluster to run in your location.

Create a location with the
in your corresponding infr

Service requesti
sat-dc22cffa-363

Confirm backend storage request

Confirm the backend storage has been configured to fulfill this request.

Service	Maximum volume size	Total capacity
cloud-object-storage sat-dc22cffa-3635-47af-a7e3-7ad8	--	--

Service					Created	
cloud-object-storage	sat-dc22cffa-3635-47af-a7e3-7ad8	Normal	9	cos-sat-test-prod-1	12/17/2021, 10:51 AM	⋮
satellite	Control plane	Normal	3		12/10/2021, 4:57 PM	⋮

Items per page: 25 1–2 of 2 items 1 1 of 1 page

Billing for Object Storage for Satellite

Deprecated: IBM Cloud continues to evaluate its service offerings periodically, keeping in perspective our client requirements and market direction. As a result, as of December 16, 2024, the Object Storage for Satellite offering is being deprecated. For more information, see Deprecation overview.

Object Storage for Satellite has a different pricing model than the typical pay-as-you-go scheme used in the public cloud.

Instead, storage is allocated at a fixed capacity using a "T-shirt size" model. The available sizes are:

Object Storage capacity	Raw storage required	Monthly price
Small (12 TB)	18 TB	$502
Medium (24 TB)	36 TB	$878
Large (48 TB)	72 TB	$1254
Extra Large (96 TB)	144 TB	$2006

Available sizes

The total cost for using Object Storage for Satellite is a combination of:

1. Cost of infrastructure
2. Base Satellite charge
3. Object Storage fixed capacity pricing

Choosing capacity

The storage instance capacity is set during the provisioning process. You will need to work with your Satellite administrator to ensure that enough raw capacity exists in the underlying Satellite infrastructure.

As each application has unique storage needs, it is not possible to provide much in the way of generic guidance for choosing a capacity. Take into account the nature of the application (for example, processing high-resolution satellite imagery or 4K video will require a greater capacity than a simple document repository) and the expected scaling of storage needs, and consult IBM Cloud support as needed.

Adding capacity

At this point, it is not possible to extend capacity once an instance is provisioned. Instead, you will need to provision an additional instance and create a new bucket for the overflow.

Supported APIs

Deprecated: IBM Cloud continues to evaluate its service offerings periodically, keeping in perspective our client requirements and market direction. As a result, as of December 16, 2024, the Object Storage for Satellite offering is being deprecated. For more information, see Deprecation overview.

Object Storage for Satellite supports most S3 APIs.

Supported S3 APIs

- `AbortMultipartUpload`
- `CompleteMultipartUpload`
- `CopyObject`
- `CreateBucket`
- `CreateMultipartUpload`
- `DeleteBucket`
- `DeleteBucketCors`
- `DeleteBucketLifecycle`
- `DeleteBucketWebsite`
- `DeleteObject`
- `DeleteObjects`
- `DeleteObjectTagging`
- `DeletePublicAccessBlock`
- `GetBucketAcl`
- `GetBucketCors`
- `GetBucketLifecycle`
- `GetBucketLocation`
- `GetBucketVersioning`
- `GetBucketWebsite`
- `GetObject`
- `GetObjectAcl`
- `GetObjectTagging`
- `GetPublicAccessBlock`
- `HeadBucket`
- `HeadObject`
- `ListBuckets`
- `ListMultipartUploads`
- `ListObjects`
- `ListObjectsV2`
- `ListObjectVersions`
- `ListParts`
- `PutBucketAcl`
- `PutBucketCors`
- `PutBucketLifecycle` (expiration rules only)
- `PutBucketVersioning`
- `PutBucketWebsite`
- `PutObject`
- `PutObjectAcl`
- `PutObjectTagging`
- `PutPublicAccessBlock`
- `UploadPart`
- `UploadPartCopy`

Unsupported S3 APIs

- `PutBucketLifecycle` (archive rules only)
- `RestoreObject`

API reference

About the IBM Cloud Object Storage S3 API

The IBM Cloud® Object Storage API is a REST-based API for reading and writing objects.

It uses IBM Cloud® Identity and Access Management for authentication and authorization, and supports a subset of the S3 API for easy migration of applications to IBM Cloud.

This reference documentation is being continuously improved. If you have technical questions about using the API in your application, post them on StackOverflow. Add both `ibm-cloud-platform` and `object-storage` tags and help improve this documentation thanks to your feedback.

As Cloud Identity and Access Management tokens are relatively easy to work with, `curl` is a good choice for basic testing and interaction with your storage. More information can be found in the `curl` reference.

The following tables describe the complete set of operations of the IBM Cloud Object Storage API. For more information, see the API reference page for buckets or objects.

Bucket operations

These operations create, delete, get information about, and control behavior of buckets.

Bucket operation	Note
`GET` Buckets	Used to retrieve a list of all buckets that belong to an account.
`DELETE` Bucket	Deletes an empty bucket.
`DELETE` Bucket CORS	Deletes any CORS (cross-origin resource sharing) configuration set on a bucket.
`GET` Bucket	Lists objects in a bucket. Limited to listing 1,000 objects at a time.
`GET` Bucket CORS	Retrieves any CORS configuration set on a bucket.
`HEAD` Bucket	Retrieves a bucket's headers.
`GET` Multipart Uploads	Lists multipart uploads that aren't completed or canceled.
`PUT` Bucket	Buckets have naming restrictions. Accounts are limited to 100 buckets.
`PUT` Bucket CORS	Creates a CORS configuration for a bucket.

Bucket operation

Object operations

These operations create, delete, get information about, and control behavior of objects.

Object operation	Note
`DELETE` Object	Deletes an object from a bucket.
`DELETE` Batch	Deletes many objects from a bucket with one operation.
`GET` Object	Retrieves an object from a bucket.
`HEAD` Object	Retrieves an object's headers.
`OPTIONS` Object	Checks CORS configuration to see whether a specific request can be sent.

`PUT` Object	Adds an object to a bucket.
`PUT` Object (Copy)	Creates a copy of an object.
Begin Multipart Upload	Creates an upload ID for a set of parts to be uploaded.
Upload Part	Uploads a part of an object that is associated with an upload ID.
Upload Part (Copy)	Uploads a part of an existing object that is associated with an upload ID.
Complete Multipart Upload	Assembles an object from parts that are associated with an upload ID.
Cancel Multipart Upload	Cancels upload and deletes outstanding parts that are associated with an upload ID.
List Parts	Returns a list of parts that are associated with an upload ID

Object operation

More information about IBM Cloud Object Storage features and use-cases can be found at [ibm.com](ibm.com).

Common headers and error codes

Data transfers use many standard protocols and have unique requirements. Keep up-to-date with the reference to common headers and some error codes.

Common Request Headers

The following table describes supported common request headers. IBM Cloud® Object Storage ignores any common headers that are not listed below if sent as part of a request, although some requests might support extra headers as defined in this document.

Header	Note
Authorization	**Required** for all requests (OAuth2 `bearer` token).
`ibm-service-instance-id`	**Required** for requests to create or list buckets.
`Content-MD5`	The base64 encoded 128-bit binary MD5 hash of the payload, which is used as an integrity check to ensure that the payload was not altered in transit. The base64 encoding must be performed on the binary output of the MD5 hash, not the hexadecimal representation.
`Expect`	The value `100-continue` waits for acknowledgment from the system that the headers are appropriate before sending the payload.
`host`	Either the endpoint or the 'virtual host' syntax of `{bucket-name}.{endpoint}`. Typically, this header is automatically added. For more information about endpoints, see [Endpoints and storage locations](#)
`Cache-Control`	Can be used to specify caching behavior along the request/reply chain. For more information, go to http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.9

Custom metadata

A benefit of using Object Storage is the ability to add custom metadata by sending key-value pairs as headers. These headers take the form of `x-amz-meta-{KEY}`. Note that unlike AWS S3, IBM Cloud Object Storage combines multiple headers with the same metadata key into a comma-separated list of values.

Common Response Headers

The following table describes common response headers.

Header	Note

`Content-Length`	The length of the request body in bytes.
`Connection`	Indicates whether the connection is open or closed.
`Date`	Timestamp of the request.
`ETag`	MD5 hash value of the request.
`Server`	Name of the responding server.
`X-Clv-Request-Id`	Unique identifier generated per request.

Lifecycle Response Headers

The following table describes response headers for archived objects

Header	Note
`x-amz-restore`	Included if the object has been restored or if a restoration is in progress.
`x-amz-storage-class`	Returns `GLACIER` or `ACCELERATED` if archived or temporarily restored.
`x-ibm-archive-transition-time`	Returns the date and time when the object is scheduled to transition to the archive tier.
`x-ibm-transition`	Included if the object has transition metadata and returns the tier and original time of transition.
`x-ibm-restored-copy-storage-class`	Included if an object is in the `RestoreInProgress` or `Restored` states and returns the storage class of the bucket.

Error Codes

Error Code	Description	HTTP Status Code
AccessDenied	Access Denied	403 Forbidden
BadDigest	The Content-MD5 that you specified did not match what we received.	400 Bad Request
BucketAlreadyExists	The requested bucket name isn't available. The bucket namespace is shared by all users of the system. Please select a different name and try again.	409 Conflict
BucketAlreadyOwnedByYou	Your previous request to create the named bucket that is succeeded and you already own it.	409 Conflict
BucketNotEmpty	The bucket that you tried to delete isn't empty.	409 Conflict
CredentialsNotSupported	This request does not support credentials.	400 Bad Request
EntityTooSmall	Your proposed upload is smaller than the minimum allowed object size.	400 Bad Request
EntityTooLarge	Your proposed upload exceeds the maximum allowed object size.	400 Bad Request
IncompleteBody	You did not provide the number of bytes specified by the Content-Length HTTP header.	400 Bad Request

IncorrectNumberOfFilesInPostRequest	POST requires exactly one file upload per request.	400 Bad Request
InlineDataTooLarge	Inline data exceeds the maximum allowed size.	400 Bad Request
InternalError	We encountered an internal error. Please try again.	500 Internal Server Error
InvalidAccessKeyId	The AWS access key Id that you provided does not exist in our records.	403 Forbidden
InvalidArgument	Invalid Argument	400 Bad Request
InvalidBucketName	The specified bucket is not valid.	400 Bad Request
InvalidBucketState	The request is not valid with the current state of the bucket.	409 Conflict
InvalidDigest	The Content-MD5 that you specified is not valid.	400 Bad Request
InvalidLocationConstraint	The specified location constraint is not valid. For more information about regions, see How to Select a Region for Your Buckets.	400 Bad Request
InvalidObjectState	The operation is not valid for the current state of the object.	403 Forbidden
InvalidPart	One or more of the specified parts might not be found. The part might not have been uploaded, or the specified entity tag might not have matched the part's entity tag.	400 Bad Request
InvalidPartOrder	The list of parts was not in ascending order. Parts list must specified in order by part number.	400 Bad Request
InvalidRange	The requested range cannot be satisfied.	416 Requested Range Not Satisfiable
InvalidRequest	Please use AWS4-HMAC-SHA256.	400 Bad Request
InvalidSecurity	The provided security credentials are not valid.	403 Forbidden
InvalidURI	Mightn't parse the specified URI.	400 Bad Request
KeyTooLong	Your key is too long.	400 Bad Request
MalformedPOSTRequest	The body of your POST request is not well-formed multipart/form-data.	400 Bad Request
MalformedXML	The XML you provided was not well-formed or did not validate against our published schema.	400 Bad Request
MaxMessageLengthExceeded	Your request was too large.	400 Bad Request
MaxPostPreDataLengthExceededError	Your POST request fields preceding the upload file were too large.	400 Bad Request

MetadataTooLarge	Your metadata headers exceed the maximum allowed metadata size.	400 Bad Request
MethodNotAllowed	The specified method is not allowed against this resource.	405 Method Not Allowed
MissingContentLength	You must provide the Content-Length HTTP header.	411 Length Required
MissingRequestBodyError	This happens when the user sends an empty xml document as a request. The error message is, "Request body is empty."	400 Bad Request
NoSuchBucket	The specified bucket does not exist.	404 Not Found
NoSuchKey	The specified key does not exist.	404 Not Found
NoSuchUpload	The specified multipart upload does not exist. The upload ID might be invalid, or the multipart upload might have been aborted or completed.	404 Not Found
NotImplemented	A header that you provided implies functionality that is not implemented.	501 Not Implemented
OperationAborted	A conflicting conditional operation is currently in progress against this resource. Try again.	409 Conflict
PreconditionFailed	At least one of the preconditions you specified did not hold.	412 Precondition Failed
Redirect	Temporary redirect.	307 Moved Temporarily
RequestIsNotMultiPartContent	Bucket POST must be of the enclosure-type multipart/form-data.	400 Bad Request
RequestTimeout	Your socket connection to the server was not read from or written to within the timeout period.	400 Bad Request
RequestTimeTooSkewed	The difference between the request time and the server's time is too large.	403 Forbidden
ServiceUnavailable	Reduce your request rate.	503 Service Unavailable
SlowDown	Reduce your request rate.	503 Slow Down
TemporaryRedirect	You are being redirected to the bucket while DNS updates.	307 Moved Temporarily
TooManyBuckets	You have attempted to create more buckets than allowed.	400 Bad Request
UnexpectedContent	This request does not support content.	400 Bad Request
UserKeyMustBeSpecified	The bucket POST must contain the specified field name. If it is specified, check the order of the fields.	400 Bad Request

Bucket operations

The modern capabilities of IBM Cloud® Object Storage are conveniently available through a RESTful API. Operations and methods that are used to interact with buckets (where objects are stored) are documented here.

Tip: For more information about the permissions and access, see [Bucket permissions](#).

A note about Access/Secret Key (HMAC) authentication

When authenticating to your instance of IBM Cloud® Object Storage by [using HMAC credentials](#), you need the information that is represented in Table 1 when [constructing an HMAC signature](#).

Key	Value	Example
{access_key}	Access key that is assigned to your Service Credential	`cf4965cebe074720a4929759f57e1214`
{date}	The formatted date of your request (`yyyymmdd`)	`20180613`
{region}	The location code for your endpoint	`us-standard`
{signature}	The hash that is created by using the secret key, location, and date	`ffe2b6e18f9dcc41f593f4dbb39882a6bb4d26a73a04326e62a8d344e07c1a3e`
{timestamp}	The formatted date and time of your request	`20180614T001804Z`

HMAC signature components

List buckets

A `GET` request that is sent to the endpoint root returns a list of buckets that are associated with the specified service instance. For more information about endpoints, see [Endpoints and storage locations](#).

Note: Not all operations are supported in Satellite environments. For more information, see [supported Satellite APIs](#)

Header	Type	Required?	Description
`ibm-service-instance-id`	String	Yes	List buckets that were created in this service instance.

Headers

Query Parameter	Value	Required?	Description
`extended`	None	No	Provides `LocationConstraint` and `CreationTemplateId` metadata in the listing.

Query parameters

Syntax

```
GET https://{endpoint}/
```

Example request

```
$ GET / HTTP/1.1
Authorization: Bearer {token}
Content-Type: text/plain
Host: s3.us.cloud-object-storage.appdomain.cloud
ibm-service-instance-id: {ibm-service-instance-id}
```

Example request

```
$ GET / HTTP/1.1
Authorization: 'AWS4-HMAC-SHA256 Credential={access-key}/{date}/{region}/s3/aws4_request,SignedHeaders=host;x-amz-
date;,Signature={signature}'
x-amz-date: {timestamp}
```

```
Content-Type: text/plain
Host: s3.us.cloud-object-storage.appdomain.cloud
```

Example response

```
$ <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ListAllMyBucketsResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
    <Owner>
        <ID>{account-id}</ID>
        <DisplayName>{account-id}</DisplayName>
    </Owner>
    <Buckets>
        <Bucket>
            <Name>bucket-27200-lwx4cfvcue</Name>
            <CreationDate>2016-08-18T14:21:36.593Z</CreationDate>
        </Bucket>
        <Bucket>
            <Name>bucket-27590-drqmydpfdv</Name>
            <CreationDate>2016-08-18T14:22:32.366Z</CreationDate>
        </Bucket>
        <Bucket>
            <Name>bucket-27852-290jtb0n2y</Name>
            <CreationDate>2016-08-18T14:23:03.141Z</CreationDate>
        </Bucket>
        <Bucket>
            <Name>bucket-28731-k0o1gde2rm</Name>
            <CreationDate>2016-08-18T14:25:09.599Z</CreationDate>
        </Bucket>
    </Buckets>
</ListAllMyBucketsResult>
```

Getting an extended listing

Syntax

```
GET https://{endpoint}/?extended
```

Example request

```
$ GET /?extended HTTP/1.1
Authorization: Bearer {token}
Content-Type: text/plain
Host: s3.us.cloud-object-storage.appdomain.cloud
ibm-service-instance-id: {ibm-service-instance-id}
```

Example request

```
$ GET /?extended HTTP/1.1
Authorization: 'AWS4-HMAC-SHA256 Credential={access-key}/{date}/{region}/s3/aws4_request,SignedHeaders=host;x-amz-
date;,Signature={signature}'
x-amz-date: {timestamp}
Content-Type: text/plain
Host: s3.us.cloud-object-storage.appdomain.cloud
```

Example response

```
$ <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ListAllMyBucketsResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
    <Owner>
        <ID>{account-id}</ID>
        <DisplayName>{account-id}</DisplayName>
    </Owner>
    <IsTruncated>false</IsTruncated>
    <MaxKeys>1000</MaxKeys>
    <Prefix/>
    <Marker/>
    <Buckets>
        <Bucket>
```

```
            <Name>bucket-27200-lwx4cfvcue</Name>
            <CreationDate>2016-08-18T14:21:36.593Z</CreationDate>
            <LocationConstraint>us-south-standard</LocationConstraint>
        </Bucket>
        <Bucket>
            <Name>bucket-27590-drqmydpfdv</Name>
            <CreationDate>2016-08-18T14:22:32.366Z</CreationDate>
            <LocationConstraint>us-standard</LocationConstraint>
        </Bucket>
        <Bucket>
            <Name>bucket-27852-290jtb0n2y</Name>
            <CreationDate>2016-08-18T14:23:03.141Z</CreationDate>
            <LocationConstraint>eu-standard</LocationConstraint>
        </Bucket>
        <Bucket>
            <Name>bucket-28731-k0o1gde2rm</Name>
            <CreationDate>2016-08-18T14:25:09.599Z</CreationDate>
            <LocationConstraint>us-cold</LocationConstraint>
        </Bucket>
    </Buckets>
</ListAllMyBucketsResult>
```

Create a bucket

A `PUT` request that is sent to the endpoint root and followed by a string creates a bucket. For more information about endpoints, see Endpoints and storage locations. Bucket names must be globally unique and DNS-compliant. Names between 3 and 63 characters long must be made of lowercase letters, numbers, dots (periods), and dashes (hyphens). Bucket names must begin and end with a lowercase letter or number. Bucket names can't contain consecutive dots or dashes. Bucket names that resemble IP addresses are not allowed. This operation doesn't use operation-specific query parameters.

Important: Bucket names must be unique because all buckets in the public cloud share a global namespace. This requirement allows for access to a bucket without needing to provide any service instance or account information. It is also not possible to create a bucket with a name beginning with `cosv1-` or `account-` as these prefixes are reserved by the system.

Note: Not all operations are supported in Satellite environments. For more information, see supported Satellite APIs

Header	Type	Required?	Description
`ibm-service-instance-id`	String	Yes	This header references the service instance where the bucket is to be created and to which data usage can be billed.
`x-amz-bucket-object-lock-enabled`	Boolean	No	Specifies whether you want to enable Object Lock on the new bucket. This header automatically enables versioning.

Headers

Tip: When setting Object Lock on a new bucket, ensure that no typographical errors are in the `x-amz-bucket-object-lock-enabled` header. If either the header or the value is misspelled, the bucket is created, but Object Lock and Versioning is **not** enabled.

Note: Personally Identifiable Information (PII): When creating buckets or adding objects, do not use any information that can identify any user (natural person) by name, location, or any other means in the name of the bucket or object.

Syntax

```
$ PUT https://{endpoint}/{bucket-name} # path style
PUT https://{bucket-name}.{endpoint} # virtual host style
```

Example request

The following example creates a bucket that is called 'images'.

```
$ PUT /images HTTP/1.1
Authorization: Bearer {token}
Content-Type: text/plain
```

```
Host: s3.us.cloud-object-storage.appdomain.cloud
ibm-service-instance-id: {ibm-service-instance-id}
```

Example request

```
$ PUT /images HTTP/1.1
Authorization: 'AWS4-HMAC-SHA256 Credential={access-key}/{date}/{region}/s3/aws4_request,SignedHeaders=host;x-amz-
date;,Signature={signature}'
x-amz-date: {timestamp}
Content-Type: text/plain
Host: s3.us.cloud-object-storage.appdomain.cloud
```

Example response

```
$ HTTP/1.1 200 OK
Date: Wed, 24 Aug 2016 17:45:25 GMT
X-Clv-Request-Id: dca204eb-72b5-4e2a-a142-808d2a5c2a87
Accept-Ranges: bytes
Server: Cleversafe/3.9.0.115
X-Clv-S3-Version: 2.5
x-amz-request-id: dca204eb-72b5-4e2a-a142-808d2a5c2a87
Content-Length: 0
```

Create a bucket with a different storage class

To create a bucket with a different storage class, send an XML block that specifies a bucket configuration with a `LocationConstraint` of `{provisioning code}` in the body of a `PUT` request to a bucket endpoint. For more information about endpoints, see Endpoints and storage locations. Standard bucket naming rules apply. This operation doesn't use operation-specific query parameters.

Note: Not all operations are supported in Satellite environments. For more information, see supported Satellite APIs

||Header | Type | Description | |-------------------------|--------|---| `ibm-service-instance-id` | String | This header references the service instance where the bucket is to be created and to which data usage can be billed.

Syntax

```
$ PUT https://{endpoint}/{bucket-name} # path style
PUT https://{bucket-name}.{endpoint} # virtual host style
```

The body of the request must contain an XML block with the following schema:

Element	Type	Children	Ancestor	Constraint
`CreateBucketConfiguration`	Container	`LocationConstraint`	•	•
`LocationConstraint`	String	•	`CreateBucketConfiguration`	Valid location code

Body of the request schema

```
$ <CreateBucketConfiguration>
   <LocationConstraint>us-vault</LocationConstraint>
</CreateBucketConfiguration>
```

A list of valid provisioning codes for `LocationConstraint` can be referenced in the Storage Classes guide.

Example request

The following example creates a bucket that is called 'vault-images'.

```
$ PUT /vault-images HTTP/1.1
```

```
Authorization: Bearer {token}
Content-Type: text/plain
Host: s3.us.cloud-object-storage.appdomain.cloud
ibm-service-instance-id: {ibm-service-instance-id}
Content-Length: 110
```

Example request

```
$ PUT /vault-images HTTP/1.1
Authorization: 'AWS4-HMAC-SHA256 Credential={access-key}/{date}/{region}/s3/aws4_request,SignedHeaders=host;x-amz-
date;,Signature={signature}'
x-amz-date: {timestamp}
Content-Type: text/plain
Host: s3.us.cloud-object-storage.appdomain.cloud
```

```
$ <CreateBucketConfiguration>
  <LocationConstraint>us-vault</LocationConstraint>
</CreateBucketConfiguration>
```

Example response

```
$ HTTP/1.1 200 OK
Date: Fri, 17 Mar 2017 17:52:17 GMT
X-Clv-Request-Id: b6483b2c-24ae-488a-884c-db1a93b9a9a6
Accept-Ranges: bytes
Server: Cleversafe/3.9.0.115
X-Clv-S3-Version: 2.5
Content-Length: 0
```

Create a bucket with Key Protect or Hyper Protect Crypto Services managed encryption keys (SSE-KP)

To create a bucket where the encryption keys are managed by Key Protect or Hyper Protect Crypto Services, it is necessary to have access to an active Key Protect or Hyper Protect Crypto Services service instance. This operation doesn't use operation-specific query parameters.

For more information about using Key Protect to manage your encryption keys, see the documentation for Key Protect.

For more information about Hyper Protect Crypto Services, see the documentation.

Important: Managed encryption for a Cross Region bucket **must** use a root key from a Key Protect instance in the nearest high-availability location (`us-south` or `jp-tok`).

Note: Not all operations are supported in Satellite environments. For more information, see supported Satellite APIs

Header	Type	Description
`ibm-service-instance-id`	String	This header references the service instance where the bucket is to be created and to which data usage can be billed.
`ibm-sse-kp-encryption-algorithm`	String	This header is used to specify the algorithm and the key size to use with the encryption key that is stored by using Key Protect. This value must be set to the string `AES256`.
`ibm-sse-kp-customer-root-key-crn`	String	This header is used to reference the specific root key that is used by Key Protect or Hyper Protect Crypto Services to encrypt this bucket. This value must be the full CRN of the root key.

Headers

Syntax

```
PUT https://{endpoint}/{bucket-name} # path style
PUT https://{bucket-name}.{endpoint} # virtual host style
```

Example request

The following example creates a bucket that is called 'secure-files'.

```
$ PUT /secure-files HTTP/1.1
Authorization: Bearer {token}
Content-Type: text/plain
Host: s3.us-south.objectstorage.s3.us-south.cloud-object-storage.appdomain.cloud.net
ibm-service-instance-id: {ibm-service-instance-id}
ibm-sse-kp-encryption-algorithm: "AES256"
ibm-sse-kp-customer-root-key-crn: {customer-root-key-id}
```

Example request

```
$ PUT /secure-files HTTP/1.1
Authorization: 'AWS4-HMAC-SHA256 Credential={access-key}/{date}/{region}/s3/aws4_request,SignedHeaders=host;x-amz-
date;,Signature={signature}'
x-amz-date: {timestamp}
Content-Type: text/plain
Host: s3.us.cloud-object-storage.appdomain.cloud
ibm-sse-kp-encryption-algorithm: "AES256"
ibm-sse-kp-customer-root-key-crn: {customer-root-key-id}
```

Example response

```
$ HTTP/1.1 200 OK
Date: Wed, 24 Aug 2016 17:45:25 GMT
X-Clv-Request-Id: dca204eb-72b5-4e2a-a142-808d2a5c2a87
Accept-Ranges: bytes
Server: Cleversafe/3.9.0.115
X-Clv-S3-Version: 2.5
x-amz-request-id: dca204eb-72b5-4e2a-a142-808d2a5c2a87
Content-Length: 0
```

Retrieve a bucket's headers

A `HEAD` issued to a bucket returns the headers for that bucket.

Tip: `HEAD` requests don't return a body and thus can't return specific error messages such as `NoSuchBucket`, only `NotFound`.

Note: Not all operations are supported in Satellite environments. For more information, see [supported Satellite APIs]

Syntax

```
HEAD https://{endpoint}/{bucket-name} # path style
HEAD https://{bucket-name}.{endpoint} # virtual host style
```

Example request

The following example fetches the headers for the 'images' bucket.

```
$ HEAD /images HTTP/1.1
Content-Type: text/plain
Host: s3.us.cloud-object-storage.appdomain.cloud
Authorization:Bearer {token}
```

Example request

```
$ HEAD /images HTTP/1.1
Authorization: 'AWS4-HMAC-SHA256 Credential={access-key}/{date}/{region}/s3/aws4_request,SignedHeaders=host;x-amz-
date;,Signature={signature}'
x-amz-date: {timestamp}
Content-Type: text/plain
Host: s3.us.cloud-object-storage.appdomain.cloud
```

Example response

```
$ HTTP/1.1 200 OK
Date: Wed, 24 Aug 2016 17:46:35 GMT
X-Clv-Request-Id: 0c2832e3-3c51-4ea6-96a3-cd8482aca08a
Accept-Ranges: bytes
Server: Cleversafe/3.9.0.115
X-Clv-S3-Version: 2.5
x-amz-request-id: 0c2832e3-3c51-4ea6-96a3-cd8482aca08a
Content-Length: 0
```

Example request

`HEAD` requests on buckets with Key Protect encryption return extra headers.

```
$ HEAD /secure-files HTTP/1.1
Content-Type: text/plain
Host: s3.us.cloud-object-storage.appdomain.cloud
Authorization:Bearer {token}
```

Example request

```
$ HEAD /secure-files HTTP/1.1
Authorization: 'AWS4-HMAC-SHA256 Credential={access-key}/{date}/{region}/s3/aws4_request,SignedHeaders=host;x-amz-
date;,Signature={signature}'
x-amz-date: {timestamp}
Content-Type: text/plain
Host: s3.us.cloud-object-storage.appdomain.cloud
```

Example response

```
$ HTTP/1.1 200 OK
Date: Wed, 24 Aug 2016 17:46:35 GMT
X-Clv-Request-Id: 0c2832e3-3c51-4ea6-96a3-cd8482aca08a
Accept-Ranges: bytes
Server: Cleversafe/3.9.0.115
X-Clv-S3-Version: 2.5
x-amz-request-id: 0c2832e3-3c51-4ea6-96a3-cd8482aca08a
Content-Length: 0
ibm-sse-kp-enabled: True
ibm-sse-kp-crk-id: {customer-root-key-id}
```

List objects in a specific bucket (Version 2)

A `GET` request addressed to a bucket returns a list of objects, limited to 1,000 at a time and returned in non-lexicographical order. The `StorageClass` value that is returned in the response is a default value as storage class operations are not implemented in Object Storage. This operation doesn't use operation-specific headers or payload elements.

Note: Not all operations are supported in Satellite environments. For more information, see supported Satellite APIs

Syntax

```
GET https://{endpoint}/{bucket-name}?list-type=2 # path style
GET https://{bucket-name}.{endpoint}?list-type=2 # virtual host style
```

Optional query parameters

Name	Type	Description
`list-type`	String	Indicates version 2 of the API and the value must be 2.
`prefix`	String	Constrains response to object names that begin with `prefix`.

`delimiter`	String	Groups objects between the `prefix` and the `delimiter`.
`encoding-type`	String	If Unicode characters that are not supported by XML are used in an object name, this parameter can be set to `url` to properly encode the response.
`max-keys`	String	Restricts the number of objects to display in the response. The default and maximum value is 1,000.
`fetch-owner`	String	Version 2 of the API does not include the `Owner` information by default. Set this parameter to `true` if `Owner` information is wanted in the response.
`continuation-token`	String	Specifies the next set of objects to be returned when your response is truncated (`IsTruncated` element returns `true`). Your initial response includes the `NextContinuationToken` element. Use this token in the next request as the value for `continuation-token`.
`start-after`	String	Returns key names after a specific key object. *This parameter is only valid in your initial request.* If a `continuation-token` parameter is included in your request, this parameter is ignored.

Optional query parameters

Example request (simple)

This request lists the objects inside the "apiary" bucket.

```
$ GET /apiary?list-type=2 HTTP/1.1
Content-Type: text/plain
Host: s3.us.cloud-object-storage.appdomain.cloud
Authorization: Bearer {token}
```

Sample request (simple)

```
$ GET /apiary?list-type=2 HTTP/1.1
Authorization: 'AWS4-HMAC-SHA256 Credential={access-key}/{date}/{region}/s3/aws4_request,SignedHeaders=host;x-amz-
date;,Signature={signature}'
x-amz-date: {timestamp}
Content-Type: text/plain
Host: s3.us.cloud-object-storage.appdomain.cloud
```

Example response (simple)

```
$ HTTP/1.1 200 OK
Date: Wed, 24 Aug 2016 17:36:24 GMT
X-Clv-Request-Id: 9f39ff2e-55d1-461b-a6f1-2d0b75138861
Accept-Ranges: bytes
Server: Cleversafe/3.13.3.57
X-Clv-S3-Version: 2.5
x-amz-request-id: 9f39ff2e-55d1-461b-a6f1-2d0b75138861
Content-Type: application/xml
Content-Length: 814
```

```
$ <ListBucketResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Name>apiary</Name>
  <Prefix/>
  <KeyCount>3</KeyCount>
  <MaxKeys>1000</MaxKeys>
  <Delimiter/>
  <IsTruncated>false</IsTruncated>
  <Contents>
    <Key>drone-bee</Key>
    <LastModified>2016-08-25T17:38:38.549Z</LastModified>
    <ETag>"0cbc6611f5540bd0809a388dc95a615b"</ETag>
    <Size>4</Size>
    <StorageClass>STANDARD</StorageClass>
  </Contents>
  <Contents>
    <Key>soldier-bee</Key>
    <LastModified>2016-08-25T17:49:06.006Z</LastModified>
    <ETag>"37d4c94839ee181a2224d6242176c4b5"</ETag>
    <Size>11</Size>
    <StorageClass>STANDARD</StorageClass>
  </Contents>
  <Contents>
    <Key>worker-bee</Key>
    <LastModified>2016-08-25T17:46:53.288Z</LastModified>
    <ETag>"d34d8aada2996fc42e6948b926513907"</ETag>
    <Size>467</Size>
    <StorageClass>STANDARD</StorageClass>
  </Contents>
</ListBucketResult>
```

Example request (max-keys parameter)

This request lists the objects inside the "apiary" bucket with a max key returned set to 1.

```
$ GET /apiary?list-type=2&max-keys=1 HTTP/1.1
Content-Type: text/plain
Host: s3.us.cloud-object-storage.appdomain.cloud
Authorization: Bearer {token}
```

Sample request (max-keys parameter)

```
$ GET /apiary?list-type=2&max-keys=1 HTTP/1.1
Authorization: 'AWS4-HMAC-SHA256 Credential={access-key}/{date}/{region}/s3/aws4_request,SignedHeaders=host;x-amz-
date;,Signature={signature}'
x-amz-date: {timestamp}
Content-Type: text/plain
Host: s3.us.cloud-object-storage.appdomain.cloud
```

Example response (Truncated Response)

```
$ HTTP/1.1 200 OK
Date: Wed, 24 Aug 2016 17:36:24 GMT
X-Clv-Request-Id: 9f39ff2e-55d1-461b-a6f1-2d0b75138861
Accept-Ranges: bytes
Server: Cleversafe/3.13.3.57
X-Clv-S3-Version: 2.5
x-amz-request-id: 9f39ff2e-55d1-461b-a6f1-2d0b75138861
Content-Type: application/xml
Content-Length: 598
```

```
$ <ListBucketResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Name>apiary</Name>
  <Prefix/>

<NextContinuationToken>1dPe45g5uuxjyASPegLq80sQsZKL5OB2by4Iz_7YGR5NjiOENBPZXqvKJN6_PgKGVzZYTlws7qqdWaMklzb8HX2iDxxl72ane3rUFQrvNMeIih49MZ4APUjrAu

  <KeyCount>1</KeyCount>
  <MaxKeys>1</MaxKeys>
  <Delimiter/>
  <IsTruncated>true</IsTruncated>
  <Contents>
    <Key>drone-bee</Key>
    <LastModified>2016-08-25T17:38:38.549Z</LastModified>
    <ETag>"0cbc6611f5540bd0809a388dc95a615b"</ETag>
    <Size>4</Size>
    <StorageClass>STANDARD</StorageClass>
  </Contents>
</ListBucketResult>
```

Example request (continuation-token parameter)

This request lists the objects inside the "apiary" bucket with a continuation token specified.

```
$ GET /apiary?list-type=2&max-keys=1&continuation-
```

```
token=1dPe45g5uuxjyASPegLq80sQsZKL5OB2by4Iz_7YGR5NjiOENBPZXqvKJN6_PgKGVzZYTlws7qqdWaMklzb8HX2iDxxl72ane3rUFQrvNMeIih49MZ4APUjrAuYI83KxSMmfKHGZ
g HTTP/1.1
Content-Type: text/plain
Host: s3.us.cloud-object-storage.appdomain.cloud
Authorization: Bearer {token}
```

Sample request (continuation-token parameter)

```
$ GET /apiary?list-type=2&max-keys=1&continuation-
token=1dPe45g5uuxjyASPegLq80sQsZKL5OB2by4Iz_7YGR5NjiOENBPZXqvKJN6_PgKGVzZYTlws7qqdWaMklzb8HX2iDxxl72ane3rUFQrvNMeIih49MZ4APUjrAuYI83KxSMmfKHGZ
g  HTTP/1.1
Authorization: 'AWS4-HMAC-SHA256 Credential={access-key}/{date}/{region}/s3/aws4_request,SignedHeaders=host;x-amz-
date;,Signature={signature}'
x-amz-date: {timestamp}
Content-Type: text/plain
Host: s3.us.cloud-object-storage.appdomain.cloud
```

Example response (Truncated Response, continuation-token parameter)

```
$ HTTP/1.1 200 OK
Date: Wed, 24 Aug 2016 17:36:24 GMT
X-Clv-Request-Id: 9f39ff2e-55d1-461b-a6f1-2d0b75138861
Accept-Ranges: bytes
Server: Cleversafe/3.13.3.57
X-Clv-S3-Version: 2.5
x-amz-request-id: 9f39ff2e-55d1-461b-a6f1-2d0b75138861
Content-Type: application/xml
Content-Length: 604
```

```
$ <ListBucketResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Name>apiary</Name>
  <Prefix/>

<ContinuationToken>1dPe45g5uuxjyASPegLq80sQsZKL5OB2by4Iz_7YGR5NjiOENBPZXqvKJN6_PgKGVzZYTlws7qqdWaMklzb8HX2iDxxl72ane3rUFQrvNMeIih49MZ4APUjrAuYI83

  <NextContinuationToken>1a8j20CqowRrM4epIQ7fTBuyPZWZUeA8Epog16wYu9KhAPNoYkWQYhGURsIQbll1lP7c-OO-V5Vyzu6mogiakC4NSwlK4LyRDdHQgY-
yPH4wMB76MfQR61VyxI4TJLxIWTPSZA0nmQQWcuV2mE4jiDA</NextContinuationToken>
  <KeyCount>1</KeyCount>
  <MaxKeys>1</MaxKeys>
  <Delimiter/>
  <IsTruncated>true</IsTruncated>
  <Contents>
    <Key>soldier-bee</Key>
    <LastModified>2016-08-25T17:49:06.006Z</LastModified>
    <ETag>"37d4c94839ee181a2224d6242176c4b5"</ETag>
    <Size>11</Size>
    <StorageClass>STANDARD</StorageClass>
  </Contents>
</ListBucketResult>
```

List objects in a specific bucket (deprecated)

Note: *This API is included for compatibility with an earlier version.* See Version 2 for the recommended method of retrieving objects in a bucket.

Note: Not all operations are supported in Satellite environments. For more information, see supported Satellite APIs

A `GET` request addressed to a bucket returns a list of objects, limited to 1,000 at a time and returned in non-lexicographical order. The `StorageClass` value that is returned in the response is a default value as storage class operations are not implemented in Object Storage. This operation doesn't use operation-specific headers or payload elements.

Syntax

```
GET https://{endpoint}/{bucket-name} # path style
GET https://{bucket-name}.{endpoint} # virtual host style
```

Optional query parameters for list object method

Name	Type	Description
`prefix`	String	Constrains response to object names that begin with `prefix`.
`delimiter`	String	Groups objects between the `prefix` and the `delimiter`.
`encoding-type`	String	If Unicode characters that are not supported by XML are used in an object name, this parameter can be set to `url` to properly encode the response.
`max-keys`	String	Restricts the number of objects to display in the response. The default and maximum value is 1,000.
`marker`	String	Specifies the object from where the listing is to begin, in UTF-8 binary order.

Optional query parameters

Example request

This request lists the objects inside the "apiary" bucket.

```
$ GET /apiary HTTP/1.1
Content-Type: text/plain
Host: s3.us.cloud-object-storage.appdomain.cloud
Authorization: Bearer {token}
```

Example request

```
$ GET /apiary HTTP/1.1
Authorization: 'AWS4-HMAC-SHA256 Credential={access-key}/{date}/{region}/s3/aws4_request,SignedHeaders=host;x-amz-
date;,Signature={signature}'
x-amz-date: {timestamp}
Content-Type: text/plain
Host: s3.us.cloud-object-storage.appdomain.cloud
```

Example response

```
$ HTTP/1.1 200 OK
Date: Wed, 24 Aug 2016 17:36:24 GMT
X-Clv-Request-Id: 9f39ff2e-55d1-461b-a6f1-2d0b75138861
Accept-Ranges: bytes
Server: Cleversafe/3.9.0.115
X-Clv-S3-Version: 2.5
x-amz-request-id: 9f39ff2e-55d1-461b-a6f1-2d0b75138861
Content-Type: application/xml
Content-Length: 909
```

```
$ <ListBucketResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Name>apiary</Name>
  <Prefix/>
  <Marker/>
  <MaxKeys>1000</MaxKeys>
  <Delimiter/>
  <IsTruncated>false</IsTruncated>
  <Contents>
    <Key>drone-bee</Key>
    <LastModified>2016-08-25T17:38:38.549Z</LastModified>
    <ETag>"0cbc6611f5540bd0809a388dc95a615b"</ETag>
    <Size>4</Size>
    <Owner>
      <ID>{account-id}</ID>
      <DisplayName>{account-id}</DisplayName>
    </Owner>
    <StorageClass>STANDARD</StorageClass>
  </Contents>
  <Contents>
    <Key>soldier-bee</Key>
    <LastModified>2016-08-25T17:49:06.006Z</LastModified>
    <ETag>"37d4c94839ee181a2224d6242176c4b5"</ETag>
    <Size>11</Size>
    <Owner>
      <ID>{account-id}</ID>
      <DisplayName>{account-id}</DisplayName>
    </Owner>
    <StorageClass>STANDARD</StorageClass>
  </Contents>
  <Contents>
    <Key>worker-bee</Key>
    <LastModified>2016-08-25T17:46:53.288Z</LastModified>
    <ETag>"d34d8aada2996fc42e6948b926513907"</ETag>
    <Size>467</Size>
    <Owner>
      <ID>{account-id}</ID>
      <DisplayName>{account-id}</DisplayName>
    </Owner>
    <StorageClass>STANDARD</StorageClass>
  </Contents>
</ListBucketResult>
```

Delete a bucket

A `DELETE` request that is issued to an empty bucket deletes the bucket. The name of the bucket is held in reserve by the system for 10 minutes after the deletion. After 10 minutes, the name is released for re-use. *Only empty buckets can be deleted.*

If the Object Storage service instance is deleted, all bucket names in that instance are held in reserve by the system for 7 days. After 7 days, the names are released for re-use.

Note: Not all operations are supported in Satellite environments. For more information, see supported Satellite APIs

Syntax

```
DELETE https://{endpoint}/{bucket-name} # path style
DELETE https://{bucket-name}.{endpoint} # virtual host style
```

Optional headers

Name	Type	Description
`aspera-ak-max-tries`	String	Specifies the number of times to attempt the delete operation. The default value is 2.

Optional headers

Example request

```
$ DELETE /apiary HTTP/1.1
Host: s3.us.cloud-object-storage.appdomain.cloud
Authorization: Bearer {token}
```

Example request

```
$ DELETE /apiary HTTP/1.1
Authorization: 'AWS4-HMAC-SHA256 Credential={access-key}/{date}/{region}/s3/aws4_request,SignedHeaders=host;x-amz-
date;,Signature={signature}'
x-amz-date: {timestamp}
Content-Type: text/plain
Host: s3.us.cloud-object-storage.appdomain.cloud
```

The server responds with `204 No Content`.

If a non-empty bucket is requested for deletion, the server responds with `409 Conflict`.

Example response

```
$ <Error>
  <Code>BucketNotEmpty</Code>
  <Message>The bucket you tried to delete is not empty.</Message>
  <Resource>/apiary/</Resource>
  <RequestId>9d2bbc00-2827-4210-b40a-8107863f4386</RequestId>
  <httpStatusCode>409</httpStatusCode>
</Error>
```

Configure Object Lock on an existing bucket

Note: Not all operations are supported in Satellite environments. For more information, see supported Satellite APIs

A `PUT` request that is addressed to an empty bucket with the `?object-lock` query parameter sets a new object lock configuration on a bucket.

Syntax

```
PUT https://{endpoint}/{bucket-name}?object-lock # path style
PUT https://{bucket-name}.{endpoint}?object-lock # virtual host style
```

The Object Lock configuration is provided as XML in the body of the request. New requests overwrite any existing replication rules that are present on the bucket.

An Object Lock configuration must include one rule.

Header	Type	Description
`Content-MD5`	String	**Required**: The Base64 encoded 128-bit MD5 hash of the payload, which is used as an integrity check to ensure that the payload wasn't altered in transit.

Headers

The body of the request must contain an XML block with the following schema:

Element	Type	Children	Ancestor	Constraint
`ObjectLockConfiguration`	Container	`ObjectLockEnabled`, `Rule`	None	Limit 1.
`ObjectLockEnabled`	String	None	`ObjectLockConfiguration`	The only valid value is `ENABLED`.
`Rule`	Container	`DefaultRetention`	`ObjectLockConfiguration`	Limit 1
`DefaultRetention`	Container	`Days`, `Mode`, `Years`	`Rule`	Limit 1.
`Days`	Integer	None	`DefaultRetention`	The number of days that you want to specify for the default retention period. It can't be combined with `Years`.
`Mode`	String	None	`DefaultRetention`	Only `COMPLIANCE` is supported currently.
`Years`	Integer	None	`DefaultRetention`	The number of years that you want to specify for the default retention period. It can't be combined with `Days`.

Body of the request schema

Example request

This request lists the objects inside the "apiary" bucket.

```
$ GET /apiary HTTP/1.1
Content-Type: text/plain
Host: s3.us.cloud-object-storage.appdomain.cloud
Authorization: Bearer {token}
```

Example response

```
$ HTTP/1.1 200 OK
Date: Wed, 24 Aug 2016 17:36:24 GMT
X-Clv-Request-Id: 9f39ff2e-55d1-461b-a6f1-2d0b75138861
Accept-Ranges: bytes
Server: Cleversafe/3.9.0.115
X-Clv-S3-Version: 2.5
x-amz-request-id: 9f39ff2e-55d1-461b-a6f1-2d0b75138861
Content-Type: application/xml
Content-Length: 909
```

```
$ <ObjectLockConfiguration  xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <ObjectLockEnabled>ENABLED</ObjectLockEnabled>
  <Rule>
      <DefaultRetention>
        <Days>30</Days>
        <Mode>COMPLIANCE</Mode>
      </DefaultRetention>
  </Rule>
</ObjectLockConfiguration>
```

List canceled or incomplete multipart uploads for a bucket

A `GET` issued to a bucket with the proper parameters retrieves information about any canceled or incomplete multipart uploads for a bucket.

Note: Not all operations are supported in Satellite environments. For more information, see [supported Satellite APIs]

Syntax

```
GET https://{endpoint}/{bucket-name}?uploads= # path style
GET https://{bucket-name}.{endpoint}?uploads= # virtual host style
```

Parameters

Name	Type	Description
`prefix`	String	Constrains response to object names that begin with `{prefix}`.
`delimiter`	String	Groups objects between the `prefix` and the `delimiter`.
`encoding-type`	String	If Unicode characters that are not supported by XML are used in an object name, this parameter can be set to `url` to properly encode the response.
`max-uploads`	Integer	Restricts the number of objects to display in the response. The default and maximum value is 1,000.
`key-marker`	String	Specifies from where the listing is to begin.
`upload-id-marker`	String	Ignored if `key-marker` is not specified, otherwise sets a point at which to begin listing the parts above `upload-id-marker`.

Parameters

Example request

The following example retrieves all current canceled and incomplete multipart uploads.

```
$ GET /apiary?uploads= HTTP/1.1
Authorization: Bearer {token}
Host: s3.us.cloud-object-storage.appdomain.cloud
```

Example request

```
$ GET /apiary?uploads= HTTP/1.1
Authorization: 'AWS4-HMAC-SHA256 Credential={access-key}/{date}/{region}/s3/aws4_request,SignedHeaders=host;x-amz-
date;,Signature={signature}'
x-amz-date: {timestamp}
Content-Type: text/plain
Host: s3.us.cloud-object-storage.appdomain.cloud
```

Example response (no multipart uploads in progress)

```
$ HTTP/1.1 200 OK
Date: Wed, 5 Oct 2016 15:22:27 GMT
X-Clv-Request-Id: 9fa96daa-9f37-42ee-ab79-0bcda049c671
Accept-Ranges: bytes
Server: Cleversafe/3.9.0.129
X-Clv-S3-Version: 2.5
x-amz-request-id: 9fa96daa-9f37-42ee-ab79-0bcda049c671
Content-Type: application/xml
Content-Length: 374
```

```
$ <ListMultipartUploadsResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Bucket>apiary</Bucket>
  <KeyMarker/>
  <UploadIdMarker/>
  <NextKeyMarker>multipart-object-123</NextKeyMarker>
  <NextUploadIdMarker>0000015a-df89-51d0-2790-dee1ac994053</NextUploadIdMarker>
  <MaxUploads>1000</MaxUploads>
  <IsTruncated>false</IsTruncated>
  <Upload>
    <Key>file</Key>
    <UploadId>0000015a-d92a-bc4a-c312-8c1c2a0e89db</UploadId>
    <Initiator>
      <ID>d4d11b981e6e489486a945d640d41c4d</ID>
      <DisplayName>d4d11b981e6e489486a945d640d41c4d</DisplayName>
    </Initiator>
    <Owner>
      <ID>d4d11b981e6e489486a945d640d41c4d</ID>
      <DisplayName>d4d11b981e6e489486a945d640d41c4d</DisplayName>
    </Owner>
    <StorageClass>STANDARD</StorageClass>
    <Initiated>2017-03-16T22:09:01.002Z</Initiated>
  </Upload>
  <Upload>
    <Key>multipart-object-123</Key>
    <UploadId>0000015a-df89-51d0-2790-dee1ac994053</UploadId>
    <Initiator>
      <ID>d4d11b981e6e489486a945d640d41c4d</ID>
      <DisplayName>d4d11b981e6e489486a945d640d41c4d</DisplayName>
    </Initiator>
    <Owner>
      <ID>d4d11b981e6e489486a945d640d41c4d</ID>
      <DisplayName>d4d11b981e6e489486a945d640d41c4d</DisplayName>
    </Owner>
    <StorageClass>STANDARD</StorageClass>
    <Initiated>2017-03-18T03:50:02.960Z</Initiated>
  </Upload>
</ListMultipartUploadsResult>
```

List any cross-origin resource sharing configuration for a bucket

A `GET` issued to a bucket with the proper parameters retrieves information about cross-origin resource sharing (CORS) configuration for a bucket.

Note: Not all operations are supported in Satellite environments. For more information, see supported Satellite APIs

Syntax

```
GET https://{endpoint}/{bucket-name}?cors= # path style
GET https://{bucket-name}.{endpoint}?cors= # virtual host style
```

Example request

The following example lists a CORS configuration on the "apiary" bucket.

```
$ GET /apiary?cors= HTTP/1.1
Authorization: Bearer {token}
Host: s3.us.cloud-object-storage.appdomain.cloud
```

Example request

```
$ GET /apiary?cors= HTTP/1.1
Authorization: 'AWS4-HMAC-SHA256 Credential={access-key}/{date}/{region}/s3/aws4_request,SignedHeaders=host;x-amz-
date;,Signature={signature}'
x-amz-date: {timestamp}
Content-Type: text/plain
Host: s3.us.cloud-object-storage.appdomain.cloud
```

Example response No CORS configuration set

```
$ HTTP/1.1 200 OK
Date: Wed, 5 Oct 2016 15:20:30 GMT
X-Clv-Request-Id: 0b69bce1-8420-4f93-a04a-35d7542799e6
Accept-Ranges: bytes
Server: Cleversafe/3.9.0.129
X-Clv-S3-Version: 2.5
x-amz-request-id: 0b69bce1-8420-4f93-a04a-35d7542799e6
Content-Type: application/xml
Content-Length: 123
```

```
$ <CORSConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <CORSRule>
    <AllowedMethod>GET</AllowedMethod>
    <AllowedMethod>PUT</AllowedMethod>
    <AllowedMethod>POST</AllowedMethod>
    <AllowedOrigin>http://www.ibm.com</AllowedOrigin>
  </CORSRule>
</CORSConfiguration>
```

Create a cross-origin resource sharing configuration for a bucket

A `PUT` issued to a bucket with the proper parameters creates or replaces a cross-origin resource sharing (CORS) configuration for a bucket.

Note: Not all operations are supported in Satellite environments. For more information, see [supported Satellite APIs](#)

Syntax

```
PUT https://{endpoint}/{bucket-name}?cors= # path style
PUT https://{bucket-name}.{endpoint}?cors= # virtual host style
```

Payload Elements

The body of the request must contain an XML block with the following schema:

Element	Type	Children	Ancestor	Constraint
`CORSConfiguration`	Container	`CORSRule`	•	•
`CORSRule`	Container	`AllowedOrigin`, `AllowedMethod`	Delete	•

`AllowedOrigin`	String	●	`CORSRule`	Valid origin string
`AllowedMethod`	String	●	`CORSRule`	Valid method string

Body of the request schema

The required `Content-MD5` header needs to be the binary representation of a base64-encoded MD5 hash. The following snippet shows one way to achieve the content for that particular header.

```
echo -n (XML block) | openssl dgst -md5 -binary | openssl enc -base64
```

Example request

The following example adds a CORS configuration that allows requests from `www.ibm.com` to issue `GET`, `PUT`, and `POST` requests to the bucket.

```
$ PUT /apiary?cors= HTTP/1.1
Authorization: Bearer {token}
Content-Type: text/plain
Host: s3.us.cloud-object-storage.appdomain.cloud
Content-MD5: M625BaNwd/OytcM7O5gIaQ==
Content-Length: 237
```

Example request

```
$ PUT /apiary?cors= HTTP/1.1
Authorization: 'AWS4-HMAC-SHA256 Credential={access-key}/{date}/{region}/s3/aws4_request,SignedHeaders=host;x-amz-
date;,Signature={signature}'
x-amz-date: {timestamp}
Content-Type: text/plain
Host: s3.us.cloud-object-storage.appdomain.cloud
Content-MD5: M625BaNwd/OytcM7O5gIaQ==
Content-Length: 237
```

```
$ <CORSConfiguration>
  <CORSRule>
    <AllowedOrigin>http://www.ibm.com</AllowedOrigin>
    <AllowedMethod>GET</AllowedMethod>
    <AllowedMethod>PUT</AllowedMethod>
    <AllowedMethod>POST</AllowedMethod>
  </CORSRule>
</CORSConfiguration>
```

Example response

```
$ HTTP/1.1 200 OK
Date: Wed, 5 Oct 2016 15:39:38 GMT
X-Clv-Request-Id: 7afca6d8-e209-4519-8f2c-1af3f1540b42
Accept-Ranges: bytes
Server: Cleversafe/3.9.0.129
X-Clv-S3-Version: 2.5
x-amz-request-id: 7afca6d8-e209-4519-8f2c-1af3f1540b42
Content-Length: 0
```

Delete any cross-origin resource sharing configuration for a bucket

A `DELETE` issued to a bucket with the proper parameters creates or replaces a cross-origin resource sharing (CORS) configuration for a bucket.

Note: Not all operations are supported in Satellite environments. For more information, see [supported Satellite APIs]

Syntax

```
DELETE https://{endpoint}/{bucket-name}?cors= # path style
DELETE https://{bucket-name}.{endpoint}?cors= # virtual host style
```

Example request

The following example deletes a CORS configuration for a bucket.

```
$ DELETE /apiary?cors= HTTP/1.1
Authorization: Bearer {token}
Host: s3.us.cloud-object-storage.appdomain.cloud
```

Example request

```
$ DELETE /apiary?cors= HTTP/1.1
Authorization: 'AWS4-HMAC-SHA256 Credential={access-key}/{date}/{region}/s3/aws4_request,SignedHeaders=host;x-amz-
date;,Signature={signature}'
x-amz-date: {timestamp}
Content-Type: text/plain
Host: s3.us.cloud-object-storage.appdomain.cloud
```

The server responds with `204 No Content`.

List the location constraint for a bucket

A `GET` issued to a bucket with the proper parameter retrieves the location information for a bucket.

Note: Not all operations are supported in Satellite environments. For more information, see supported Satellite APIs

Syntax

```
GET https://{endpoint}/{bucket-name}?location # path style
GET https://{bucket-name}.{endpoint}?location # virtual host style
```

Example request

The following example retrieves the location of the "apiary" bucket.

```
$ GET /apiary?location= HTTP/1.1
Authorization: Bearer {token}
Host: s3.us.cloud-object-storage.appdomain.cloud
```

Example request

```
$ GET /apiary?location= HTTP/1.1
Authorization: 'AWS4-HMAC-SHA256 Credential={access-key}/{date}/{region}/s3/aws4_request,SignedHeaders=host;x-amz-
date;,Signature={signature}'
x-amz-date: {timestamp}
Content-Type: text/plain
Host: s3.us.cloud-object-storage.appdomain.cloud
```

Example response

```
$ HTTP/1.1 200 OK
Date: Tue, 12 Jun 2018 21:10:57 GMT
X-Clv-Request-Id: 0e469546-3e43-4c6b-b814-5ad0db5b638f
Accept-Ranges: bytes
Server: Cleversafe/3.13.3.57
X-Clv-S3-Version: 2.5
x-amz-request-id: 0e469546-3e43-4c6b-b814-5ad0db5b638f
Content-Type: application/xml
Content-Length: 161
```

```
$ <LocationConstraint xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  us-south-standard
</LocationConstraint>
```

Create a bucket lifecycle configuration

A `PUT` operation uses the lifecycle query parameter to set lifecycle settings for the bucket. A `Content-MD5` header is required as an integrity check for the payload.

Note: Not all operations are supported in Satellite environments. For more information, see supported Satellite APIs

Syntax

```
PUT https://{endpoint}/{bucket-name}?lifecycle # path style
PUT https://{bucket-name}.{endpoint}?lifecycle # virtual host style
```

Payload Elements

The body of the request must contain an XML block with the following schema:

Element	Type	Children	Ancestor	Constraint
`LifecycleConfiguration`	Container	`Rule`	None	Limit 1
`Rule`	Container	`ID`, `Status`, `Filter`, `Transition`	`LifecycleConfiguration`	Limit 1
`ID`	String	None	`Rule`	**Must** consist of (`a-z`,`A-Z`,`0-9`) and the following symbols:`!` `_` `.` `*` `'` `(` `)` `-`
`Filter`	String	`Prefix`	`Rule`	**Must** contain a `Prefix` element.
`Expiration`	Container	`Days`, `Date`, `ExpiredObjectDeleteMarker`	`Rule`	Limit 1
`Prefix`	String	None	`Filter`	If using a transition (archive) rule, the value **must** be set to `<Prefix/>`. This limitation does not apply to expiration rules.
`Transition`	Container	`Days`, `StorageClass`	`Rule`	Limit 1 transition rule, and 1000 rules in total.
`Days`	Non-negative integer	None	`Transition`	**Must** be a value equal to or greater than 0.
`Date`	Date	None	`Transition`	**Must** be in ISO 8601 Format and the date must be in the f future.
`StorageClass`	String	None	`Transition`	**Must** be set to `GLACIER` or `ACCELERATED`.

`ExpiredObjectDeleteMarker`	Boolean	None	`Expiration`	**Must** be `true` or `false`.
`NoncurrentVersionExpiration`	Container	`NoncurrentDays`	`NoncurrentVersionExpiration`	Limit 1
`NoncurrentDays`	Positive Integer	None	`Transition`	**Must** be a value greater than 0.
`AbortIncompleteMultipartUpload`	Container	`DaysAfterInitiation`	`Rule`	Limit 1
`DaysAfterInitiation`	Non-negative Integer	None	`AbortIncompleteMultipartUpload`	**Must** be a value greater than 0.

Body of the request schema

Note: IBM Cloud® Object Storage IaaS (non-IAM) accounts are unable to set the transition storage class to `ACCELERATED`.

```
$ <LifecycleConfiguration>
    <Rule>
        <ID>{string}</ID>
        <Status>Enabled</Status>
        <Filter>
            <Prefix/>
        </Filter>
        <Transition>
            <Days>{integer}</Days>
            <StorageClass>GLACIER</StorageClass>
        </Transition>
    </Rule>
</LifecycleConfiguration>
```

The required `Content-MD5` header needs to be the binary representation of a base64-encoded MD5 hash. The following snippet shows one way to achieve the content for that particular header.

```
echo -n (XML block) | openssl dgst -md5 -binary | openssl enc -base64
```

Example request

```
PUT /apiary?lifecycle HTTP/1.1
Content-Type: text/plain
Host: s3.us.cloud-object-storage.appdomain.cloud
Authorization: {authorization-string}
Content-Type: text/plain
Content-MD5: M625BaNwd/OytcM7O5gIaQ==
Content-Length: 305
```

Example request

```
$ PUT /apiary?lifecycle HTTP/1.1
Authorization: 'AWS4-HMAC-SHA256 Credential={access-key}/{date}/{region}/s3/aws4_request,SignedHeaders=host;x-amz-
date;,Signature={signature}'
x-amz-date: {timestamp}
Content-Type: text/plain
Content-MD5: M625BaNwd/OytcM7O5gIaQ==
Content-Length: 305
Host: s3.us.cloud-object-storage.appdomain.cloud
```

```
<LifecycleConfiguration>
    <Rule>
        <ID>my-archive-policy</ID>
        <Filter>
            <Prefix/>
        </Filter>
        <Status>Enabled</Status>
        <Transition>
            <Days>20</Days>
            <StorageClass>GLACIER</StorageClass>
        </Transition>
    </Rule>
</LifecycleConfiguration>
```

The server responds with `200 OK`.

```
$ <LifecycleConfiguration>
    <Rule>
        <ID>{string}</ID>
        <Status>Enabled</Status>
        <Filter>
            <Prefix/>
        </Filter>
        <Expiration>
            <Days>{integer}</Days>
        </Expiration>
    </Rule>
</LifecycleConfiguration>
```

The required `Content-MD5` header needs to be the binary representation of a base64-encoded MD5 hash. The following snippet shows one way to achieve the content for that particular header.

```
echo -n (XML block) | openssl dgst -md5 -binary | openssl enc -base64
```

Example request

```
$ PUT /cit-test?lifecycle HTTP/1.1
Host: 192.168.35.22
Date: Fri, 28 Feb 2020 14:12:06 +0000
Authorization: AWS MOfXYiHQ9QTyD2ALoiOh:WrlFRE2KMmhutBf3CxIZoNLl/ko=
Content-MD5: To3JYtaVNR3+aGYtl1dlmw==
Content-Length: 321
```

```
<LifecycleConfiguration  xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Rule>
    <ID>ID1</ID>
    <Status>Enabled</Status>
    <Filter>
      <Prefix/>
    </Filter>
    <Expiration>
      <Days>100</Days>
    </Expiration>
  </Rule>
</LifecycleConfiguration>
```

Example response The server responds with `200 OK`.

```
$ We are completely uploaded and fine
HTTP/1.1 200 OK
Date: Fri, 28 Feb 2020 14:12:06 GMT
X-Clv-Request-Id: 587d909f-4939-41ef-8c16-80aea16a0587
Server: Cleversafe/3.14.9.53
X-Clv-S3-Version: 2.5
x-amz-request-id: 587d909f-4939-41ef-8c16-80aea16a0587
Content-Length: 0
```

Example request

```
PUT /{bucket-name}?lifecycle HTTP/1.1
```

```
Authorization: Bearer {token}
Content-Type: text/plain
Host: s3.us.cloud-object-storage.appdomain.cloud
ibm-service-instance-id: {ibm-service-instance-id}
Content-Length: 123
```

Example request

```
$ PUT /{bucket-name}?lifecycle HTTP/1.1
Authorization: 'AWS4-HMAC-SHA256 Credential={access-key}/{date}/{region}/s3/aws4_request,SignedHeaders=host;x-amz-
date;,Signature={signature}'
x-amz-date: {timestamp}
Content-Type: text/plain
Host: s3.us.cloud-object-storage.appdomain.cloud
Content-Length: 123
```

Retrieve a bucket lifecycle configuration

A `GET` operation uses the lifecycle query parameter to retrieve lifecycle settings for the bucket.

Syntax

```
GET https://{endpoint}/{bucket-name}?lifecycle # path style
GET https://{bucket-name}.{endpoint}?lifecycle # virtual host style
```

Example request

```
$ GET /apiary?lifecycle HTTP/1.1
Content-Type: text/plain
Host: s3.us.cloud-object-storage.appdomain.cloud
Authorization: {authorization-string}
```

Example request

```
$ GET /apiary?lifecycle HTTP/1.1
Authorization: 'AWS4-HMAC-SHA256 Credential={access-key}/{date}/{region}/s3/aws4_request,SignedHeaders=host;x-amz-
date;,Signature={signature}'
x-amz-date: {timestamp}
Content-Type: text/plain
Host: s3.us.cloud-object-storage.appdomain.cloud
```

Example Response

```
<LifecycleConfiguration>
    <Rule>
        <ID>my-archive-policy</ID>
        <Filter>
            <Prefix/>
        </Filter>
        <Status>Enabled</Status>
        <Transition>
            <Days>20</Days>
            <StorageClass>GLACIER</StorageClass>
        </Transition>
    </Rule>
</LifecycleConfiguration>
```

Example request

```
$ GET /cit_dump-log?lifecycle HTTP/1.1
Host: 192.168.35.22
User-Agent: curl/7.64.1
Accept: */*
Date: Fri, 28 Feb 2020 14:00:43 +0000
Authorization: AWS MOfXYiHQ9QTyD2ALoiOh:iKm2QNetyW740kylP6ja2pze3DM=
Content-MD5: 1B2M2Y8AsgTpgAmY7PhCfg==
```

Example Response

```
$ HTTP/1.1 200 OK
Date: Fri, 28 Feb 2020 14:00:43 GMT
X-Clv-Request-Id: ecbf9294-284d-4169-b2cd-5d52b2450808
Server: Cleversafe/3.14.9.53
X-Clv-S3-Version: 2.5
Accept-Ranges: bytes
x-amz-request-id: ecbf9294-284d-4169-b2cd-5d52b2450808
Content-Type: application/xml
Content-Length: 276
```

```
$ <LifecycleConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
    <Rule>
      <ID>ID1</ID>
      <Status>Enabled</Status>
      <Filter>
        <Prefix/>
      </Filter>
      <Expiration>
        <Days>270</Days>
      </Expiration>
    </Rule>
</LifecycleConfiguration>
```

Delete stale data with expiration rules

Note: Any expiration actions for objects that are subject to a bucket's Immutable Object Storage retention policy are deferred until the retention policy is no longer enforced.

For more about using lifecycle configuration to delete objects, check out the documentation.

This implementation of the `PUT` operation uses the `lifecycle` query parameter to set lifecycle settings for the bucket. This operation allows for a single lifecycle policy definition for a bucket. The policy is defined as a set of rules that consists of the following parameters: `ID`, `Status`, `Filter`, and `Expiration`.

Note: Not all operations are supported in Satellite environments. For more information, see supported Satellite APIs

Header	Type	Description
`Content-MD5`	String	**Required**: The Base64 encoded 128-bit MD5 hash of the payload, which is used as an integrity check to ensure that the payload wasn't altered in transit.

Body of the request schema

The following snippet shows one way to achieve the content for that particular header.

```
echo -n (XML block) | openssl dgst -md5 -binary | openssl enc -base64
```

The body of the request must contain an XML block with the following schema:

Element	Type	Children	Ancestor	Constraint
`LifecycleConfiguration`	Container	`Rule`	None	Limit 1.
`Rule`	Container	`ID`, `Status`, `Filter`, `Expiration`	`LifecycleConfiguration`	Limit 1000.
`ID`	String	None	`Rule`	Must consist of (`a-z`,`A-Z`,`0-9`) and the following symbols: `!` `_` `.` `*` `'` `(` `)` `-`

`Filter`	String	`Prefix`	`Rule`	Must contain a `Prefix` element
`Prefix`	String	None	`Filter`	The rule applies to any objects with keys that match this prefix.
`Expiration`	Container	`Days` or `Date`	`Rule`	Limit 1.
`Days`	Non-negative integer	None	`Expiration`	Must be a value greater than 0.
`Date`	Date	None	`Expiration`	Must be in ISO 8601 Format.

Body of the request schema

Syntax

```
PUT https://{endpoint}/{bucket}?lifecycle # path style
PUT https://{bucket}.{endpoint}?lifecycle # virtual host style
```

Example request

```
PUT /images?lifecycle HTTP/1.1
Host: s3.us.cloud-object-storage.appdomain.cloud
Date: Wed, 7 Feb 2018 17:50:00 GMT
Authorization: authorization string
Content-Type: text/plain
Content-MD5: M625BaNwd/OytcM7O5gIaQ==
Content-Length: 305

<LifecycleConfiguration>
 <Rule>
  <ID>id1</ID>
  <Filter />
  <Status>Enabled</Status>
  <Expiration>
   <Days>60</Days>
  </Expiration>
 </Rule>
</LifecycleConfiguration>
```

Delete the lifecycle configuration for a bucket

A `DELETE` issued to a bucket with the proper parameters removes any lifecycle configurations for a bucket.

Syntax

```
DELETE https://{endpoint}/{bucket-name}?lifecycle # path style
DELETE https://{bucket-name}.{endpoint}?lifecycle # virtual host style
```

Example request

```
$ DELETE /apiary?lifecycle HTTP/1.1
Authorization: {authorization-string}
Host: s3.us.cloud-object-storage.appdomain.cloud
```

Example request

```
$ DELETE /apiary?lifecycle HTTP/1.1
Authorization: 'AWS4-HMAC-SHA256 Credential={access-key}/{date}/{region}/s3/aws4_request,SignedHeaders=host;x-amz-
date;,Signature={signature}'
x-amz-date: {timestamp}
Content-Type: text/plain
Host: s3.us.cloud-object-storage.appdomain.cloud
```

The server responds with `204 No Content`.

Example request

```
$ DELETE /cit-test?lifecycle HTTP/1.1
Host: 192.168.35.22
User-Agent: curl/7.64.1
Accept: */*
Date: Fri, 28 Feb 2020 14:16:47 +0000
Authorization: AWS MOfXYiHQ9QTyD2ALoiOh:n25GU28DiBgkNVgET5hKmLmp938=
Content-MD5: 1B2M2Y8AsgTpgAmY7PhCfg==
```

Example response

```
$ HTTP/1.1 204 No Content
Date: Fri, 28 Feb 2020 14:16:47 GMT
X-Clv-Request-Id: 3e8bdf1e-b611-4b83-a404-e7d3e58e60b0
Server: Cleversafe/3.14.9.53
X-Clv-S3-Version: 2.5
x-amz-request-id: 3e8bdf1e-b611-4b83-a404-e7d3e58e60b0
```

The server responds with `204 No Content`.

Add a retention policy on an existing bucket

Note: Immutable Object Storage is available in certain regions only, see Integrated Services for details. The service also requires a Standard pricing plan. See pricing for details.

Find out more about Immutable Object Storage in the documentation.

The minimum and maximum supported values for the retention period settings `MinimumRetention`, `DefaultRetention`, and `MaximumRetention` are a minimum of 0 days and a maximum of 365243 days (1000 years).

This operation doesn't use extra query parameters. The required `Content-MD5` header needs to be the binary representation of a base64-encoded MD5 hash. The following snippet shows one way to achieve the content for that particular header.

Important: Policies are enforced until the end of a retention period, and cannot be altered until the retention period has expired. While IBM Cloud® Object Storage uses the S3 API for most operations, the APIs that are used for configuring retention policies are not the same as the S3 API, although some terminology might be shared. Read this documentation carefully to prevent any users in your organization from creating objects that can't be deleted, even by IBM Cloud administrators.

Note: Not all operations are supported in Satellite environments. For more information, see supported Satellite APIs

```
echo -n (XML block) | openssl dgst -md5 -binary | openssl enc -base64
```

Syntax

```
PUT https://{endpoint}/{bucket-name}?protection= # path style
PUT https://{bucket-name}.{endpoint}?protection= # virtual host style
```

The body of the request must contain an XML block with the following schema:

Element	Type	Children	Ancestor	Constraint
`ProtectionConfiguration`	Container	`Status`, `MinimumRetention`, `MaximumRetention`, `DefaultRetention`	•	•
`Status`	String	•	`ProtectionConfiguration`	Valid status string

`MinimumRetention`	Container	`Days`	`ProtectionConfiguration`	•
`MaximumRetention`	Container	`Days`	`ProtectionConfiguration`	•
`DefaultRetention`	Container	`Days`	`ProtectionConfiguration`	•
`Days`	Integer	•	`MinimumRetention`, `MaximumRetention`, `DefaultRetention`	Valid retention integer

Body of the request schema

Example request

```
PUT /example-bucket?protection= HTTP/1.1
Authorization: {authorization-string}
x-amz-date: 20181011T190354Z
x-amz-content-sha256: 2938f51643d63c864fdbea618fe71b13579570a86f39da2837c922bae68d72df
Content-MD5: GQmpTNpruOyK6YrxHnpj7g==
Content-Type: text/plain
Host: 67.228.254.193
Content-Length: 299

<ProtectionConfiguration>
  <Status>Retention</Status>
  <MinimumRetention>
    <Days>100</Days>
  </MinimumRetention>
  <MaximumRetention>
    <Days>10000</Days>
  </MaximumRetention>
  <DefaultRetention>
    <Days>2555</Days>
  </DefaultRetention>
</ProtectionConfiguration>
```

Example response

```
$ HTTP/1.1 200 OK
Date: Wed, 5 Oct 2018 15:39:38 GMT
X-Clv-Request-Id: 7afca6d8-e209-4519-8f2c-1af3f1540b42
Accept-Ranges: bytes
Server: Cleversafe/3.14.1
X-Clv-S3-Version: 2.5
x-amz-request-id: 7afca6d8-e209-4519-8f2c-1af3f1540b42
Content-Length: 0
```

Configure a bucket for static website hosting

A `PUT` issued to a bucket with the proper parameters creates or replaces a static website configuration for a bucket.

Note: Not all operations are supported in Satellite environments. For more information, see supported Satellite APIs

Syntax

```
PUT https://{endpoint}/{bucket-name}?website # path style
PUT https://{bucket-name}.{endpoint}?website # virtual host style
```

Payload Elements

The body of the request must contain an XML block with the following schema:

Element	Type	Children	Ancestor	Notes
`WebsiteConfiguration`	Container	`ErrorDocument`, `IndexDocument`, `RedirectAllRequestsTo`, `RoutingRule`	•	Required
`ErrorDocument`	Container	`Key`	`WebsiteConfiguration`	•
`Key`	String	•	`ErrorDocument`	•
`IndexDocument`	Container	`Suffix`	`WebsiteConfiguration`	•
`Suffix`	String	•	`IndexDocument`	•
`RedirectAllRequestsTo`	Container	`HostName`, `Protocol`	`WebsiteConfiguration`	If given, it must be the only element that is specified
`HostName`	String	•	`RedirectAllRequestsTo`	•
`Protocol`	String	•	`RedirectAllRequestsTo`	•
`RoutingRules`	Container	`RoutingRule`	`WebsiteConfiguration`	•
`RoutingRule`	Container	`Condition`, `Redirect`	`RoutingRules`	•
`Condition`	Container	`HttpErrorCodeReturnedEquals`, `KeyPrefixEquals`	`RoutingRule`	•
`HttpErrorCodeReturnedEquals`	String	•	`Condition`	•
`KeyPrefixEquals`	String	•	`Condition`	•
`Redirect`	Container	`HostName`, `HttpRedirectCode`, `Protocol`, `ReplaceKeyPrefixWith`, `ReplaceKeyWith`	`RoutingRule`	•
`HostName`	String	•	`Redirect`	•
`HttpRedirectCode`	String	•	`Redirect`	•

`Protocol`	String	●	`Redirect`	●
`ReplaceKeyPrefixWith`	String	●	`Redirect`	●
`ReplaceKeyWith`	String	●	`Redirect`	●

Body of the request schema

Example request

The following example adds a website configuration that serves a basic website that looks for an `index.html` file in each prefix. For example, a request that is made to `/apiary/images/` serves the content in `/apiary/images/index.html` without the need for specifying the actual file.

```
$ PUT /apiary?website HTTP/1.1
Authorization: Bearer {token}
Content-Type: text/plain
Host: s3.us.cloud-object-storage.appdomain.cloud
Content-Length: 119
```

```
$ PUT /apiary?website HTTP/1.1
Authorization: 'AWS4-HMAC-SHA256 Credential={access-key}/{date}/{region}/s3/aws4_request,SignedHeaders=host;x-amz-
date;,Signature={signature}'
x-amz-date: {timestamp}
Content-Type: text/plain
Host: s3.us.cloud-object-storage.appdomain.cloud
Content-Length: 119
```

```
$ <WebsiteConfiguration>
    <IndexDocument>
        <Suffix>index.html</Suffix>
    </IndexDocument>
</WebsiteConfiguration>
```

Example response

```
$ HTTP/1.1 200 OK
Date: Wed, 5 Oct 2020 15:39:38 GMT
X-Clv-Request-Id: 7afca6d8-e209-4519-8f2c-1af3f1540b42
Accept-Ranges: bytes
Content-Length: 0
```

Delete any website configuration for a bucket

A `DELETE` request that is issued to a bucket with the proper parameters removes the website configuration for a bucket.

Note: Not all operations are supported in Satellite environments. For more information, see supported Satellite APIs

Syntax

```
DELETE https://{endpoint}/{bucket-name}?website # path style
DELETE https://{bucket-name}.{endpoint}?website # virtual host style
```

Example request

The following example deletes a website configuration for a bucket.

```
$ DELETE /apiary?website HTTP/1.1
Authorization: Bearer {token}
Host: s3.us.cloud-object-storage.appdomain.cloud
```

Example request

```
$ DELETE /apiary?website HTTP/1.1
Authorization: 'AWS4-HMAC-SHA256 Credential={access-key}/{date}/{region}/s3/aws4_request,SignedHeaders=host;x-amz-
date;,Signature={signature}'
x-amz-date: {timestamp}
Content-Type: text/plain
Host: s3.us.cloud-object-storage.appdomain.cloud
```

The server responds with `204 No Content`.

Block public ACLs on a bucket

A `PUT` request that is issued to a bucket with the proper parameters prevents adding public access ACLs on a bucket. It can be set either to fail new ACL requests, or to ignore them. `BlockPublicAcls` does not affect existing ACLs, but `IgnorePublicAcls` ignores existing ACLs. **This operation does not affect IAM Public Access policies.**

Note: Not all operations are supported in Satellite environments. For more information, see supported Satellite APIs

Syntax

```
PUT https://{endpoint}/{bucket-name}?publicAccessBlock # path style
PUT https://{bucket-name}.{endpoint}?publicAccessBlock # virtual host style
```

Payload Elements

The body of the request must contain an XML block with the following schema:

Element	Type	Children	Ancestor	Notes
PublicAccessBlockConfiguration	Container	BlockPublicAcls, IgnorePublicAcls	•	Required
BlockPublicAcls	Boolean	•	PublicAccessBlockConfiguration	•
IgnorePublicAcls	Boolean	•	PublicAccessBlockConfiguration	•

Body of the request schema

Example request

```
$ PUT /apiary?publicAccessBlock HTTP/1.1
Authorization: Bearer {token}
Host: s3.us.cloud-object-storage.appdomain.cloud
Content-Length: 155
```

```
$ PUT /apiary?publicAccessBlock HTTP/1.1
Authorization: 'AWS4-HMAC-SHA256 Credential={access-key}/{date}/{region}/s3/aws4_request,SignedHeaders=host;x-amz-
date;,Signature={signature}'
x-amz-date: {timestamp}
Host: s3.us.cloud-object-storage.appdomain.cloud
Content-Length: 155
```

```
$ <PublicAccessBlockConfiguration>
   <BlockPublicAcls>True</BlockPublicAcls>
   <IgnorePublicAcls>True</IgnorePublicAcls>
</PublicAccessBlockConfiguration>
```

Example response

```
$ HTTP/1.1 200 OK
Date: Mon, 02 Nov 2020 15:39:38 GMT
X-Clv-Request-Id: 7afca6d8-e209-4519-8f2c-1af3f1540b42
Accept-Ranges: bytes
Content-Length: 0
```

Check a public ACL block for a bucket

A `GET` issued to a bucket with the proper parameters returns the ACL block configuration for a bucket.

Syntax

```
GET https://{endpoint}/{bucket-name}?publicAccessBlock # path style
GET https://{bucket-name}.{endpoint}?publicAccessBlock # virtual host style
```

Example request

The following example reads a public access block for a bucket.

```
$ GET /apiary?publicAccessBlock HTTP/1.1
Authorization: Bearer {token}
Host: s3.us.cloud-object-storage.appdomain.cloud
```

Example request

```
$ GET /apiary?publicAccessBlock HTTP/1.1
Authorization: 'AWS4-HMAC-SHA256 Credential={access-key}/{date}/{region}/s3/aws4_request,SignedHeaders=host;x-amz-
date;,Signature={signature}'
x-amz-date: {timestamp}
Content-Type: text/plain
Host: s3.us.cloud-object-storage.appdomain.cloud
```

```
$ HTTP/1.1 200 OK
Date: Mon, 02 Nov 2020 19:52:56 GMT
X-Clv-Request-Id: 7c9079b1-2833-4abc-ba10-466ef06725b2
Server: Cleversafe/3.15.2.31
X-Clv-S3-Version: 2.5
Accept-Ranges: bytes
Content-Type: application/xml
Content-Length: 248
```

```
$ <PublicAccessBlockConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
   <BlockPublicAcls>true</BlockPublicAcls>
   <IgnorePublicAcls>true</IgnorePublicAcls>
</PublicAccessBlockConfiguration>
```

Delete a public ACL block from a bucket

A `DELETE` issued to a bucket with the proper parameters removes the public ACL block from a bucket.

Note: Not all operations are supported in Satellite environments. For more information, see [supported Satellite APIs](#)

Syntax

```
DELETE https://{endpoint}/{bucket-name}?publicAccessBlock # path style
DELETE https://{bucket-name}.{endpoint}?publicAccessBlock # virtual host style
```

Example request

The following example deletes an ACL block for a bucket.

```
$ DELETE /apiary?publicAccessBlock HTTP/1.1
Authorization: Bearer {token}
Host: s3.us.cloud-object-storage.appdomain.cloud
```

Example request

```
$ DELETE /apiary?publicAccessBlock HTTP/1.1
Authorization: 'AWS4-HMAC-SHA256 Credential={access-key}/{date}/{region}/s3/aws4_request,SignedHeaders=host;x-amz-
date;,Signature={signature}'
x-amz-date: {timestamp}
Content-Type: text/plain
Host: s3.us.cloud-object-storage.appdomain.cloud
```

The server responds with `204 No Content`.

Configure a PUT bucket inventory

A `PutBucketInventoryConfiguration` issued to a bucket with the proper parameters.

Syntax

```
PUT {bucket}?inventory&id={id}
```

Example request

The following example is of a `PutBucketInventoryConfiguration` request for a bucket.

```
$
```

Example request

```
$ PUT /mybucket?inventory&id=myid HTTP/1.1
<?xml version="1.0" encoding="UTF-8"?>
<InventoryConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
        <Id>myid</Id>
    <IsEnabled>true</IsEnabled>
    <Filter>
      <Prefix>my-filter-prefix</Prefix>
    </Filter>
        <IncludedObjectVersions>Current</IncludedObjectVersions>
    <Schedule>
      <Frequency>Daily</Frequency>
    </Schedule>
    <OptionalFields>
              <Field>Size</Field>
            <Field>LastModifiedDate</Field>
      <Field>ETag</Field>
          <Field>IsMultipartUploaded</Field>
          <Field>EncryptionStatus</Field>
          <Field>ObjectOwner</Field>
    </OptionalFields>
        <Destination>
            <S3BucketDestination>
                <Bucket>mybucket</Bucket>
                <Format>CSV</Format>
                <Prefix>my-destination-prefix</Prefix>
              </S3BucketDestination>
        </Destination>
</InventoryConfiguration>
```

The server responds with `204 No Content`.

Configure a GET bucket inventory

A `GetBucketInventoryConfiguration` issued to a bucket with the proper parameters.

Syntax

```
GET {bucket}?inventory&id={id}
```

Example request

The following example is of a `GetBucketInventoryConfiguration` request for a bucket.

```
$ GET mybucket?inventory&id=myid HTTP/1.1
```

Example response

```
$ <?xml version="1.0" encoding="UTF-8"?>
<InventoryConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
        <Id>myid</Id>
    <IsEnabled>true</IsEnabled>
    <Filter>
      <Prefix>my-filter-prefix</Prefix>
    </Filter>
        <IncludedObjectVersions>Current</IncludedObjectVersions>
    <Schedule>
      <Frequency>Daily</Frequency>
    </Schedule>
    <OptionalFields>
              <Field>Size</Field>
            <Field>LastModifiedDate</Field>
      <Field>ETag</Field>
          <Field>IsMultipartUploaded</Field>
          <Field>EncryptionStatus</Field>
          <Field>ObjectOwner</Field>
    </OptionalFields>
        <Destination>
            <S3BucketDestination>
                <Bucket>mybucket</Bucket>
                <Format>CSV</Format>
                <Prefix>my-destination-prefix</Prefix>
              </S3BucketDestination>
        </Destination>
</InventoryConfiguration>
```

Configure a LIST bucket inventory

A `ListBucketInventoryConfigurations` issued to a bucket with the proper parameters.

Syntax

```
GET {bucket}?inventory&continuation-token={continuation-token}
```

Example request

The following example is of `ListBucketInventoryConfigurations` request for a bucket.

```
$ GET /mybucket?inventory HTTP/1.1
```

Example response

```
$ <?xml version="1.0" encoding="UTF-8"?>
<ListInventoryConfigurationsResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
      <InventoryConfiguration>
          <Id>goodinventoryid</Id>
          <IsEnabled>true</IsEnabled>
          <Filter>
              <Prefix>goodFilterPrefix</Prefix>
          </Filter>
          <Destination>
              <S3BucketDestination>
                  <Format>CSV</Format>
                  <Bucket>mybucketCRN</Bucket>
                  <Prefix>goodPrefix</Prefix>
              </S3BucketDestination>
          </Destination>
```

```
        <Schedule>
            <Frequency>Daily</Frequency>
        </Schedule>
        <IncludedObjectVersions>All</IncludedObjectVersions>
        <OptionalFields>
            <Field>Size</Field>
        </OptionalFields>
    </InventoryConfiguration>
    <InventoryConfiguration>
        <Id>goodinventoryid1</Id>
        ...
    </InventoryConfiguration>
    <IsTruncated>true</IsTruncated>
        <NextContinuationToken>{continuation-token}</NextContinuationToken>
</ListInventoryConfigurationsResult>
```

Configure a DELETE bucket inventory

A `DeleteBucketInventoryConfiguration` issued to a bucket with the proper parameters.

Syntax

```
DELETE {bucket}?inventory&id={id}
```

Example request

The following example is of a `DeleteBucketInventoryConfiguration` request for a bucket.

```
$ DELETE mybucket?inventory&id=myid HTTP/1.1
```

Example response

```
$ 204 No Content
```

Next Steps

For more information, see Object operations.

Object operations

The modern capabilities of IBM Cloud® Object Storage are conveniently available via a RESTful API. Operations and methods for reading, writing, and configuring objects (stored within a bucket), are documented here.

Tip: For more information about endpoints, see Endpoints and storage locations.

A note regarding Access/Secret Key (HMAC) authentication

When authenticating to your instance of IBM Cloud Object Storage by using HMAC credentials, you need the information that is represented in Table 1 when constructing an HMAC signature.

Key	Value	Example
{access_key}	Access key assigned to your Service Credential	cf4965cebe074720a4929759f57e1214
{date}	The formatted date of your request (`yyyymmdd`)	20180613
{region}	The location code for your endpoint	us-standard
{signature}	The hash created using the secret key, location, and date	ffe2b6e18f9dcc41f593f4dbb39882a6bb4d26a73a04326e62a8d344e07c1a3e

{timestamp}	The formatted date and time of your request	20180614T001804Z

HMAC signature components

Upload an object

A `PUT` given a path to an object uploads the request body as an object. All objects uploaded in a single thread should be smaller than 500 MB to minimize the risk of network disruptions. (objects that are uploaded in multiple parts can be as large as 10 TB).

Note: Personally Identifiable Information (PII): When *naming* buckets or objects, do not use any information that can identify any user (natural person) by name, location, or any other means.

Note: Not all operations are supported in Satellite environments. For details, see supported Satellite APIs

Tip: It is possible to stream objects as large as 5 GB using a single `PUT` request. Multipart uploads are more reliable and can upload more efficiently by using multiple threads to upload parts in parallel. Uploading larger objects in a single `PUT` request results in the performance limitations of a single thread, and in the event of any failures single-threaded uploads will need to be retried in their entirety (whereas with MPU only the specific part(s) that failed need to be retried). The precise throughput that can be achieved by a single thread varies depending on the network bandwidth from the client to the IBM Cloud endpoint, the rate of packet loss (if any) on that connection, the use of HTTP vs HTTPS, the specific ciphers used in the connection and specific TCP connection parameters (such as window size), as well as other factors. While these factors can be optimized for a single-threaded upload, the optimizations would apply equally to any multi-threaded (multipart) uploads as well.

Note: Personally Identifiable Information (PII): When creating buckets or adding objects, please ensure to not use any information that can identify any user (natural person) by name, location, or any other means.

Note: Not all operations are supported in Satellite environments. For details, see supported Satellite APIs

Syntax

```
PUT https://{endpoint}/{bucket-name}/{object-name} # path style
PUT https://{bucket-name}.{endpoint}/{object-name} # virtual host style
```

Optional headers

Header	Type	Description
`x-amz-tagging`	string	A set of tags to apply to the object, formatted as query parameters (`"SomeKey=SomeValue"`).
`x-amz-object-lock-mode`	string	Valid value is `COMPLIANCE` - required if `x-amz-object-lock-retain-until-date` is present.
`x-amz-object-lock-retain-until-date`	ISO8601 Date and Time	Required if `x-amz-object-lock-mode` is present.
`x-amz-object-lock-legal-hold`	string	Valid values are `ON` or `OFF`.

Example request

```
$ PUT /apiary/queen-bee HTTP/1.1
Authorization: Bearer {token}
Content-Type: text/plain; charset=utf-8
Host: s3.us.cloud-object-storage.appdomain.cloud

Content-Length: 533

 The 'queen' bee is developed from larvae selected by worker bees and fed a
 substance referred to as 'royal jelly' to accelerate sexual maturity. After a
 short while the 'queen' is the mother of nearly every bee in the hive, and
 the colony will fight fiercely to protect her.
```

Example request

```
$ PUT /apiary/queen-bee HTTP/1.1
Authorization: 'AWS4-HMAC-SHA256 Credential={access-key}/{date}/{region}/s3/aws4_request,SignedHeaders=host;x-amz-
date;,Signature={signature}'
x-amz-date: {timestamp}
x-amz-content-sha256: {payload_hash}
Content-Type: text/plain; charset=utf-8
Host: s3.us.cloud-object-storage.appdomain.cloud

Content-Length: 533

 The 'queen' bee is developed from larvae selected by worker bees and fed a
 substance referred to as 'royal jelly' to accelerate sexual maturity. After a
 short while the 'queen' is the mother of nearly every bee in the hive, and
 the colony will fight fiercely to protect her.
```

Example response

```
HTTP/1.1 200 OK
Date: Thu, 25 Aug 2016 18:30:02 GMT
X-Clv-Request-Id: 9f0ca49a-ae13-4d2d-925b-117b157cf5c3
Accept-Ranges: bytes
Server: Cleversafe/3.9.0.121
X-Clv-S3-Version: 2.5
x-amz-request-id: 9f0ca49a-ae13-4d2d-925b-117b157cf5c3
ETag: "3ca744fa96cb95e92081708887f63de5"
Content-Length: 0
```

Get an object's headers

A `HEAD` given a path to an object retrieves that object's headers.

Note: The `Etag` value returned for objects encrypted using SSE-KP **is** the MD5 hash of the original decrypted object.

Note: Not all operations are supported in Satellite environments. For details, see [supported Satellite APIs]

Syntax

```
HEAD https://{endpoint}/{bucket-name}/{object-name} # path style
HEAD https://{bucket-name}.{endpoint}/{object-name} # virtual host style
```

Example request

```
$ HEAD /apiary/soldier-bee HTTP/1.1
Authorization: Bearer {token}
Host: s3-api.sjc-us-geo.objectstorage.s3.us-south.cloud-object-storage.appdomain.cloud.net
```

Example request

```
$ HEAD /apiary/soldier-bee HTTP/1.1
Authorization: 'AWS4-HMAC-SHA256 Credential={access-key}/{date}/{region}/s3/aws4_request,SignedHeaders=host;x-amz-
date;,Signature={signature}'
x-amz-date: {timestamp}
Host: s3.us.cloud-object-storage.appdomain.cloud
```

Example response

```
HTTP/1.1 200 OK
Date: Thu, 25 Aug 2016 18:32:44 GMT
```

```
X-Clv-Request-Id: da214d69-1999-4461-a130-81ba33c484a6
Accept-Ranges: bytes
Server: Cleversafe/3.9.0.121
X-Clv-S3-Version: 2.5
x-amz-request-id: da214d69-1999-4461-a130-81ba33c484a6
ETag: "37d4c94839ee181a2224d6242176c4b5"
Content-Type: text/plain; charset=UTF-8
Last-Modified: Thu, 25 Aug 2016 17:49:06 GMT
Content-Length: 11
```

Download an object

A `GET` given a path to an object downloads the object.

Note: The `Etag` value that is returned for objects encrypted using SSE-C/SSE-KP will **not** be the MD5 hash of the original decrypted object.

Note: Not all operations are supported in Satellite environments. For details, see supported Satellite APIs

Syntax

```
GET https://{endpoint}/{bucket-name}/{object-name} # path style
GET https://{bucket-name}.{endpoint}/{object-name} # virtual host style
```

Optional headers

Header	Type	Description
`range`	String	Returns the bytes of an object within the specified range.

Example request

```
$ GET /apiary/worker-bee HTTP/1.1
Authorization: Bearer {token}
Host: s3.us.cloud-object-storage.appdomain.cloud
```

Example request

```
$ GET /apiary/worker-bee HTTP/1.1
Authorization: 'AWS4-HMAC-SHA256 Credential={access-key}/{date}/{region}/s3/aws4_request,SignedHeaders=host;x-amz-
date;,Signature={signature}'
x-amz-date: {timestamp}
Host: s3.us.cloud-object-storage.appdomain.cloud
```

Example response

```
HTTP/1.1 200 OK
Date: Thu, 25 Aug 2016 18:34:25 GMT
X-Clv-Request-Id: 116dcd6b-215d-4a81-bd30-30291fa38f93
Accept-Ranges: bytes
Server: Cleversafe/3.9.0.121
X-Clv-S3-Version: 2.5
x-amz-request-id: 116dcd6b-215d-4a81-bd30-30291fa38f93
ETag: "d34d8aada2996fc42e6948b926513907"
Content-Type: text/plain; charset=UTF-8
Last-Modified: Thu, 25 Aug 2016 17:46:53 GMT
Content-Length: 467

 Female bees that are not fortunate enough to be selected to be the 'queen'
 while they were still larvae become known as 'worker' bees. These bees lack
 the ability to reproduce and instead ensure that the hive functions smoothly,
 acting almost as a single organism in fulfilling their purpose.
```

Delete an object

A `DELETE` given a path to an object deletes an object.

Note: Not all operations are supported in Satellite environments. For details, see [supported Satellite APIs]

Syntax

```
DELETE https://{endpoint}/{bucket-name}/{object-name} # path style
DELETE https://{bucket-name}.{endpoint}/{object-name} # virtual host style
```

Example request

```
$ DELETE /apiary/soldier-bee HTTP/1.1
Authorization: Bearer {token}
Host: s3-api.sjc-us-geo.objectstorage.s3.us-south.cloud-object-storage.appdomain.cloud.net
```

Example request

```
$ DELETE /apiary/soldier-bee HTTP/1.1
Authorization: 'AWS4-HMAC-SHA256 Credential={access-key}/{date}/{region}/s3/aws4_request,SignedHeaders=host;x-amz-
date;,Signature={signature}'
x-amz-date: {timestamp}
Host: s3.us.cloud-object-storage.appdomain.cloud
```

Example response

```
$ HTTP/1.1 204 No Content
Date: Thu, 25 Aug 2016 17:44:57 GMT
X-Clv-Request-Id: 8ff4dc32-a6f0-447f-86cf-427b564d5855
Accept-Ranges: bytes
Server: Cleversafe/3.9.0.121
X-Clv-S3-Version: 2.5
x-amz-request-id: 8ff4dc32-a6f0-447f-86cf-427b564d5855
```

Delete multiple objects

A `POST` given a path to a bucket and proper parameters deletes a specified set of objects. A `Content-MD5` header that specifies the base64 encoded MD5 hash of the request body is required.

The required `Content-MD5` header needs to be the binary representation of a base64 encoded MD5 hash.

Note: When an object that is specified in the request is not found the result returns as deleted.

Note: Not all operations are supported in Satellite environments. For details, see [supported Satellite APIs]

Note: Multiple object deletes involve a `POST operation` that is charged as Class A. The cost of the `POST` request for multiple deletes varies depending on the storage class of the objects, and the amount of data that is deleted. For more information about pricing, see the [IBM Cloud Object Storage pricing page].

Optional Elements

Header	Type	Description
`Quiet`	Boolean	Enable quiet mode for the request.

Header

Tip: The request can contain a maximum of 1000 keys that you want to delete. While this is useful in reducing the number of requests, be mindful when deleting many keys. Also, take into account the sizes of the objects to ensure suitable performance.

The following code shows one example of how to create the necessary representation of the header content:

```
echo -n (XML block) | openssl dgst -md5 -binary | openssl enc -base64
```

Syntax

```
POST https://{endpoint}/{bucket-name}?delete= # path style
POST https://{bucket-name}.{endpoint}?delete= # virtual host style
```

The body of the request must contain an XML block with the following schema:

Element	Type	Children	Ancestor	Constraint
Delete	Container	Object	•	•
Object	Container	Key	Delete	•
Key	String	•	Object	Valid key string

Body of the request schema

Example request

```
$ POST /apiary?delete= HTTP/1.1
Authorization: Bearer {token}
Host: s3.us.cloud-object-storage.appdomain.cloud
Content-Type: text/plain; charset=utf-8
Content-MD5: xj/vf7lD7vbIe/bqHTaLvg==
```

Example request

```
$ POST /apiary?delete= HTTP/1.1
Authorization: 'AWS4-HMAC-SHA256 Credential={access-key}/{date}/{region}/s3/aws4_request,SignedHeaders=host;x-amz-
date;,Signature={signature}'
x-amz-date: {timestamp}
Content-Type: text/plain; charset=utf-8
Content-MD5: xj/vf7lD7vbIe/bqHTaLvg==
Host: s3.us.cloud-object-storage.appdomain.cloud
```

Example response

```
$ HTTP/1.1 200 OK
Date: Wed, 30 Nov 2016 18:54:53 GMT
X-Clv-Request-Id: a6232735-c3b7-4c13-a7b2-cd40c4728d51
Accept-Ranges: bytes
Server: Cleversafe/3.9.0.137
X-Clv-S3-Version: 2.5
x-amz-request-id: a6232735-c3b7-4c13-a7b2-cd40c4728d51
Content-Type: application/xml
Content-Length: 207
```

```
$ <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<DeleteResult  xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
     <Deleted>
          <Key>surplus-bee</Key>
     </Deleted>
     <Deleted>
          <Key>unnecessary-bee</Key>
     </Deleted>
</DeleteResult>
```

Add or extend retention on an object

A `PUT` issued to an object with the proper parameters adds or extends retention period of the object.

Note: Not all operations are supported in Satellite environments. For details, see [supported Satellite APIs]

Syntax

```
PUT https://{endpoint}/{bucket-name}/{object-name}?retention # path style
PUT https://{bucket-name}.{endpoint}/{object-name}?retention # virtual host style
```

Payload Elements

The body of the request must contain an XML block with the following schema:

Element	Type	Children	Ancestor	Notes
Retention	Container	Mode, RetainUntilDate	•	Required
Mode	String	•	Retention	Required - valid value is COMPLIANCE
RetainUntilDate	Timestamp	•	Retention	Required

Body of the request schema

The following code shows one example of how to create the necessary representation of the header content:

```
echo -n (XML block) | openssl dgst -md5 -binary | openssl enc -base64
```

Example request

This is an example of adding or extending retention on an object.

```
$ PUT /apiary/myObject?retention HTTP/1.1
Authorization: Bearer {token}
Content-Type: text/plain
Content-MD5: cDeRJIdLuEXWmLpA79K2kg==
Host: s3.us.cloud-object-storage.appdomain.cloud
Content-Length: 119
```

```
$ <Retention>
    <Mode>COMPLIANCE</Mode>
    <RetainUntilDate>2023-04-12T23:01:00.000Z</RetainUntilDate>
</Retention>
```

Example response

```
$ HTTP/1.1 200 OK
Date: Wed, 5 Oct 2020 15:39:38 GMT
X-Clv-Request-Id: 7afca6d8-e209-4519-8f2c-1af3f1540b42
Accept-Ranges: bytes
Content-Length: 0
```

Add tags to an object

A `PUT` issued to an object with the proper parameters creates or replaces a set of key-value tags associated with the object.

Note: Not all operations are supported in Satellite environments. For details, see [supported Satellite APIs]

Syntax

```
PUT https://{endpoint}/{bucket-name}/{object-name}?tagging # path style
PUT https://{bucket-name}.{endpoint}/{object-name}?tagging # virtual host style
```

Payload Elements

The body of the request must contain an XML block with the following schema:

Element	Type	Children	Ancestor	Notes
Tagging	Container	TagSet	•	Required
TagSet	Container	Tag	Tagging	Required
Tag	String	Key, Value	TagSet	Required
Key	Container	•	Tag	Required
Value	String	•	Tag	Required

Body of the request schema

Tags must comply with the following restrictions:

- An object can have a maximum of 10 tags
- For each object, each tag key must be unique, and each tag key can have only one value.
- Minimum key length - 1 Unicode characters in UTF-8
- Maximum key length - 128 Unicode characters in UTF-8
- Maximum key byte size - 256 bytes
- Minimum value length - 0 Unicode characters in UTF-8 (Tag Value can be empty)
- Maximum value length - 256 Unicode characters in UTF-8
- Maximum value byte size - 512 bytes
- A Tag key and value may consist of US Alpha Numeric Characters (`a-z`, `A-Z`, `0-9`), and spaces representable in UTF-8, and the following symbols: `!`, `_`, `.`, `*`, `'`, `(`, `)`, `-`, `:`
- Tag keys and values are case-sensitive
- `ibm:` cannot be used as a key prefix for tags

Example request

This is an example of adding a set of tags to an object.

```
$ PUT /apiary/myObject?tagging HTTP/1.1
Authorization: Bearer {token}
Content-Type: text/plain
Host: s3.us.cloud-object-storage.appdomain.cloud
Content-Length: 119
```

```
$ PUT /apiary/myObject?tagging HTTP/1.1
Authorization: 'AWS4-HMAC-SHA256 Credential={access-key}/{date}/{region}/s3/aws4_request,SignedHeaders=host;x-amz-
date;,Signature={signature}'
x-amz-date: {timestamp}
Content-Type: text/plain
Host: s3.us.cloud-object-storage.appdomain.cloud
Content-Length: 128
```

```
$ <Tagging>
    <TagSet>
```

```
        <Tag>
            <Key>string</Key>
            <Value>string</Value>
        </Tag>
    </TagSet>
</Tagging>
```

Example response

```
$ HTTP/1.1 200 OK
Date: Wed, 5 Oct 2020 15:39:38 GMT
X-Clv-Request-Id: 7afca6d8-e209-4519-8f2c-1af3f1540b42
Accept-Ranges: bytes
Content-Length: 0
```

Read an object's tags

A `GET` issued to an object with the proper parameters returns the set of key-value tags associated with the object.

Note: Not all operations are supported in Satellite environments. For details, see [supported Satellite APIs]

Syntax

```
GET https://{endpoint}/{bucket-name}/{object-name}?tagging # path style
GET https://{bucket-name}.{endpoint}/{object-name}?tagging # virtual host style
```

Example request

This is an example of reading a set of object tags.

```
$ GET /apiary/myObject?tagging HTTP/1.1
Authorization: Bearer {token}
Content-Type: text/plain
Host: s3.us.cloud-object-storage.appdomain.cloud
Content-Length: 0
```

```
$ GET /apiarymyObject?tagging HTTP/1.1
Authorization: 'AWS4-HMAC-SHA256 Credential={access-key}/{date}/{region}/s3/aws4_request,SignedHeaders=host;x-amz-
date;,Signature={signature}'
x-amz-date: {timestamp}
Content-Type: text/plain
Host: s3.us.cloud-object-storage.appdomain.cloud
Content-Length: 0
```

Example response

```
$ HTTP/1.1 200 OK
Date: Wed, 5 Oct 2020 15:39:38 GMT
X-Clv-Request-Id: 7afca6d8-e209-4519-8f2c-1af3f1540b42
Accept-Ranges: bytes
Content-Length: 128
```

```
$ <Tagging>
    <TagSet>
        <Tag>
            <Key>string</Key>
            <Value>string</Value>
        </Tag>
    </TagSet>
</Tagging>
```

Delete an object's tags

A `DELETE` issued to a bucket with the proper parameters removes an object's tags.

Note: Not all operations are supported in Satellite environments. For details, see [supported Satellite APIs]()

Syntax

```
DELETE https://{endpoint}/{bucket-name}{object-name}?tagging # path style
DELETE https://{bucket-name}.{endpoint}{object-name}?tagging # virtual host style
```

Example request

This is an example of deleting an object's tags.

```
$ DELETE /apiary/myObject?tagging HTTP/1.1
Authorization: Bearer {token}
Host: s3.us.cloud-object-storage.appdomain.cloud
```

Example request

```
$ DELETE /apiary/myObject?tagging HTTP/1.1
Authorization: 'AWS4-HMAC-SHA256 Credential={access-key}/{date}/{region}/s3/aws4_request,SignedHeaders=host;x-amz-
date;,Signature={signature}'
x-amz-date: {timestamp}
Content-Type: text/plain
Host: s3.us.cloud-object-storage.appdomain.cloud
```

The server responds with `204 No Content`.

Copy an object

A `PUT` given a path to a new object creates a new copy of another object that is specified by the `x-amz-copy-source` header. Unless otherwise altered the metadata remains the same.

Note: Personally Identifiable Information (PII): When *naming* buckets or objects, do not use any information that can identify any user (natural person) by name, location, or any other means.

Note: Copying objects (even across locations) does not incur the public outbound bandwidth charges. All data remains inside the COS internal network.

Note: Not all operations are supported in Satellite environments. For details, see [supported Satellite APIs]()

Syntax

```
PUT https://{endpoint}/{bucket-name}/{object-name} # path style
PUT https://{bucket-name}.{endpoint}/{object-name} # virtual host style
```

Optional headers

Header	Type	Description
`x-amz-metadata-directive`	string (`COPY` or `REPLACE`)	A `REPLACE` overwrites original metadata with new metadata that is provided.
`x-amz-tagging`	string	A set of tags to apply to the object, formatted as query parameters (`"SomeKey=SomeValue"`).
`x-amz-tagging-directive`	string (`COPY` or `REPLACE`)	A `REPLACE` overwrites original tags with new tags that is provided.
`x-amz-copy-source-if-match`	String (`ETag`)	Creates a copy if the specified `ETag` matches the source object.

`x-amz-copy-source-if-none-match`	String (`ETag`)	Creates a copy if the specified `ETag` is different from the source object.
`x-amz-copy-source-if-unmodified-since`	String (time stamp)	Creates a copy if the source object has not been modified since the specified date. Date must be a valid HTTP date (for example, `Wed, 30 Nov 2016 20:21:38 GMT`).
`x-amz-copy-source-if-modified-since`	String (time stamp)	Creates a copy if the source object has been modified since the specified date. Date must be a valid HTTP date (for example, `Wed, 30 Nov 2016 20:21:38 GMT`).

Example request

This basic example takes the `bee` object from the `garden` bucket, and creates a copy in the `apiary` bucket with the new key `wild-bee`.

```
$ PUT /apiary/wild-bee HTTP/1.1
Authorization: Bearer {token}
x-amz-copy-source: /garden/bee
Host: s3.us.cloud-object-storage.appdomain.cloud
```

Example request

```
$ PUT /apiary/wild-bee HTTP/1.1
Authorization: 'AWS4-HMAC-SHA256 Credential={access-key}/{date}/{region}/s3/aws4_request,SignedHeaders=host;x-amz-
date;,Signature={signature}'
x-amz-date: {timestamp}
x-amz-copy-source: /garden/bee
Host: s3.us.cloud-object-storage.appdomain.cloud
```

Example response

```
HTTP/1.1 200 OK
Date: Wed, 30 Nov 2016 19:52:52 GMT
X-Clv-Request-Id: 72992a90-8f86-433f-b1a4-7b1b33714bed
Accept-Ranges: bytes
Server: Cleversafe/3.9.0.137
X-Clv-S3-Version: 2.5
x-amz-request-id: 72992a90-8f86-433f-b1a4-7b1b33714bed
ETag: "853aab195ce770b0dfb294a4e9467e62"
Content-Type: application/xml
Content-Length: 240
```

```
<CopyObjectResult  xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <LastModified>2016-11-30T19:52:53.125Z</LastModified>
  <ETag>"853aab195ce770b0dfb294a4e9467e62"</ETag>
</CopyObjectResult>
```

Check an object's CORS configuration

An `OPTIONS` given a path to an object along with an origin and request type checks to see whether that object is accessible from that origin by using that request type. Unlike all other requests, an OPTIONS request does not require the `authorization` or `x-amx-date` headers.

Note: Not all operations are supported in Satellite environments. For details, see [supported Satellite APIs]

Syntax

```
OPTIONS https://{endpoint}/{bucket-name}/{object-name} # path style
OPTIONS https://{bucket-name}.{endpoint}/{object-name} # virtual host style
```

Example request

```
$ OPTIONS /apiary/queen-bee HTTP/1.1
Access-Control-Request-Method: PUT
Origin: http://ibm.com
Host: s3.us.cloud-object-storage.appdomain.cloud
```

Example request

```
$ OPTIONS /apiary/queen-bee HTTP/1.1
Authorization: 'AWS4-HMAC-SHA256 Credential={access-key}/{date}/{region}/s3/aws4_request,SignedHeaders=host;x-amz-
date;,Signature={signature}'
x-amz-date: {timestamp}
Access-Control-Request-Method: PUT
Origin: http://ibm.com
Host: s3.us.cloud-object-storage.appdomain.cloud
```

Example response

```
$ HTTP/1.1 200 OK
Date: Wed, 07 Dec 2016 16:23:14 GMT
X-Clv-Request-Id: 9a2ae3e1-76dd-4eec-a8f2-1a7f60f63483
Accept-Ranges: bytes
Server: Cleversafe/3.9.0.137
X-Clv-S3-Version: 2.5
x-amz-request-id: 9a2ae3e1-76dd-4eec-a8f2-1a7f60f63483
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: PUT
Access-Control-Allow-Credentials: true
Vary: Origin, Access-Control-Request-Headers, Access-Control-Allow-Methods
Content-Length: 0
```

Uploading objects in multiple parts

When working with larger objects, multipart upload operations are recommended to write objects into IBM Cloud® Object Storage. An upload of a single object can be performed as a set of parts and these parts can be uploaded independently in any order and in parallel. Upon upload completion, Object Storage then presents all parts as a single object. This provides many benefits: network interruptions do not cause large uploads to fail, uploads can be paused and restarted over time, and objects can be uploaded as they are being created.

Multipart uploads are only available for objects larger than 5 MB. For objects smaller than 50 GB, a part size of 20 MB to 100 MB is recommended for optimum performance. For larger objects, part size can be increased without significant performance impact.

Due to the additional complexity involved, it is recommended that developers make use of a library that provides multipart upload support.

Tip: Incomplete multipart uploads do persist until the object is deleted or the multipart upload is aborted with `AbortIncompleteMultipartUpload`. If an incomplete multipart upload is not aborted, the partial upload continues to use resources. Interfaces should be designed with this point in mind, and clean up incomplete multipart uploads.

There are three phases to uploading an object in multiple parts:

1. The upload is initiated and an `UploadId` is created.
2. Individual parts are uploaded specifying their sequential part numbers and the `UploadId` for the object.
3. When all parts are finished uploading, the upload is completed by sending a request with the `UploadId` and an XML block that lists each part number and its respective `Etag` value.

Initiate a multipart upload

A `POST` issued to an object with the query parameter `upload` creates a new `UploadId` value, which is then be referenced by each part of the object being uploaded.

Note: Personally Identifiable Information (PII): When *naming* buckets or objects, do not use any information that can identify any user (natural person) by name, location, or any other means.

Note: Not all operations are supported in Satellite environments. For details, see supported Satellite APIs

Syntax

```
POST https://{endpoint}/{bucket-name}/{object-name}?uploads= # path style
POST https://{bucket-name}.{endpoint}/{object-name}?uploads= # virtual host style
```

Example request

```
$ POST /some-bucket/multipart-object-123?uploads= HTTP/1.1
Authorization: Bearer {token}
Host: s3.us.cloud-object-storage.appdomain.cloud
```

Example request

```
$ POST /some-bucket/multipart-object-123?uploads= HTTP/1.1
Authorization: 'AWS4-HMAC-SHA256 Credential={access-key}/{date}/{region}/s3/aws4_request,SignedHeaders=host;x-amz-
date;,Signature={signature}'
x-amz-date: {timestamp}
Host: s3.us.cloud-object-storage.appdomain.cloud
```

Example response

```
$ HTTP/1.1 200 OK
Date: Fri, 03 Mar 2017 20:34:12 GMT
X-Clv-Request-Id: 258fdd5a-f9be-40f0-990f-5f4225e0c8e5
Accept-Ranges: bytes
Server: Cleversafe/3.9.1.114
X-Clv-S3-Version: 2.5
Content-Type: application/xml
Content-Length: 276
```

```
$ <InitiateMultipartUploadResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Bucket>some-bucket</Bucket>
  <Key>multipart-object-123</Key>
  <UploadId>0000015a-95e1-4326-654e-a1b57887784f</UploadId>
</InitiateMultipartUploadResult>
```

Upload a part

A `PUT` request that is issued to an object with query parameters `partNumber` and `uploadId` will upload one part of an object. The parts can be uploaded serially or in parallel, but must be numbered in order.

Note: Personally Identifiable Information (PII): When *naming* buckets or objects, do not use any information that can identify any user (natural person) by name, location, or any other means.

Note: Not all operations are supported in Satellite environments. For details, see supported Satellite APIs

Syntax

```
PUT https://{endpoint}/{bucket-name}/{object-name}?partNumber={sequential-integer}&uploadId={uploadId}= # path style
PUT https://{bucket-name}.{endpoint}/{object-name}?partNumber={sequential-integer}&uploadId={uploadId}= # virtual host style
```

Example request

```
$ PUT /some-bucket/multipart-object-123?partNumber=1&uploadId=0000015a-df89-51d0-2790-dee1ac994053 HTTP/1.1
Authorization: Bearer {token}
Content-Type: application/pdf
Host: s3.us.cloud-object-storage.appdomain.cloud
Content-Length: 13374550
```

Example request

```
$ PUT /some-bucket/multipart-object-123?partNumber=1&uploadId=0000015a-df89-51d0-2790-dee1ac994053 HTTP/1.1
Authorization: 'AWS4-HMAC-SHA256 Credential={access-key}/{date}/{region}/s3/aws4_request,SignedHeaders=host;x-amz-
date;,Signature={signature}'
x-amz-date: {timestamp}
x-amz-content-sha256: STREAMING-AWS4-HMAC-SHA256-PAYLOAD
```

```
Content-Encoding: aws-chunked
x-amz-decoded-content-length: 13374550
Content-Type: application/pdf
Host: s3.us.cloud-object-storage.appdomain.cloud
Content-Length: 13374550
```

Example response

```
HTTP/1.1 200 OK
Date: Sat, 18 Mar 2017 03:56:41 GMT
X-Clv-Request-Id: 17ba921d-1c27-4f31-8396-2e6588be5c6d
Accept-Ranges: bytes
Server: Cleversafe/3.9.1.114
X-Clv-S3-Version: 2.5
ETag: "7417ca8d45a71b692168f0419c17fe2f"
Content-Length: 0
```

List parts

A `GET` given a path to a multipart object with an active `UploadID` specified as a query parameter returns a list of all of the object's parts.

Note: Not all operations are supported in Satellite environments. For details, see supported Satellite APIs

Syntax

```
GET https://{endpoint}/{bucket-name}/{object-name}?uploadId={uploadId} # path style
GET https://{bucket-name}.{endpoint}/{object-name}?uploadId={uploadId} # virtual host style
```

Query parameters

Parameter	Required?	Type	Description
`uploadId`	Required	string	Upload ID returned when initializing a multipart upload.
`max-parts`	Optional	string	Defaults to 1,000.
`part-number -marker`	Optional	string	Defines where the list of parts begins.

Parameters

Example request

```
$ GET /farm/spaceship?uploadId=01000162-3f46-6ab8-4b5f-f7060b310f37 HTTP/1.1
Authorization: bearer {token}
Host: s3.us.cloud-object-storage.appdomain.cloud
```

Example request

```
$ GET /farm/spaceship?uploadId=01000162-3f46-6ab8-4b5f-f7060b310f37 HTTP/1.1
Authorization: 'AWS4-HMAC-SHA256 Credential={access-key}/{date}/{region}/s3/aws4_request,SignedHeaders=host;x-amz-
date;,Signature={signature}'
x-amz-date: {timestamp}
Host: s3.us.cloud-object-storage.appdomain.cloud
```

Example response

```
$ HTTP/1.1 200 OK
Date: Mon, 19 Mar 2018 17:21:08 GMT
X-Clv-Request-Id: 6544044d-4f88-4bb6-9ee5-bfadf5023249
Server: Cleversafe/3.12.4.20
X-Clv-S3-Version: 2.5
Accept-Ranges: bytes
Content-Type: application/xml
Content-Length: 743
```

```
<ListPartsResult  xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Bucket>farm</Bucket>
  <Key>spaceship</Key>
  <UploadId>01000162-3f46-6ab8-4b5f-f7060b310f37</UploadId>
  <Initiator>
    <ID>d6f04d83-6c4f-4a62-a165-696756d63903</ID>
    <DisplayName>d6f04d83-6c4f-4a62-a165-696756d63903</DisplayName>
  </Initiator>
  <Owner>
    <ID>d6f04d83-6c4f-4a62-a165-696756d63903</ID>
    <DisplayName>d6f04d83-6c4f-4a62-a165-696756d63903</DisplayName>
  </Owner>
  <StorageClass>STANDARD</StorageClass>
  <MaxParts>1000</MaxParts>
  <IsTruncated>false</IsTruncated>
  <Part>
    <PartNumber>1</PartNumber>
    <LastModified>2018-03-19T17:20:35.482Z</LastModified>
    <ETag>"bb03cf4fa8603fe407a65ee1dba55265"</ETag>
    <Size>7128094</Size>
  </Part>
</ListPartsResult>
```

Complete a multipart upload

A `POST` request that is issued to an object with query parameter `uploadId` and the appropriate XML block in the body will complete a multipart upload.

Note: Not all operations are supported in Satellite environments. For details, see supported Satellite APIs

Syntax

```
POST https://{endpoint}/{bucket-name}/{object-name}?uploadId={uploadId}= # path style
POST https://{bucket-name}.{endpoint}/{object-name}?uploadId={uploadId}= # virtual host style
```

The body of the request must contain an XML block with the following schema:

Element	Type	Children	Ancestor	Constraint
CompleteMultipartUpload	Container	Part	•	•
Part	Container	PartNumber, `ETag`	Delete	•
PartNumber	String	•	Object	Valid part number
`ETag`	String	•	Object	Valid `ETag` value string

Body of the request schema

```
<CompleteMultipartUpload>
  <Part>
    <PartNumber>{sequential part number}</PartNumber>
    <ETag>{ETag value from part upload response header}</ETag>
  </Part>
</CompleteMultipartUpload>
```

Example request

```
POST /some-bucket/multipart-object-123?uploadId=0000015a-df89-51d0-2790-dee1ac994053 HTTP/1.1
```

```
Authorization: Bearer {token}
Content-Type: text/plain; charset=utf-8
Host: s3.us.cloud-object-storage.appdomain.cloud
Content-Length: 257
```

Example request

```
POST /some-bucket/multipart-object-123?uploadId=0000015a-df89-51d0-2790-dee1ac994053 HTTP/1.1
Authorization: 'AWS4-HMAC-SHA256 Credential={access-key}/{date}/{region}/s3/aws4_request,SignedHeaders=host;x-amz-
date;,Signature={signature}'
x-amz-date: {timestamp}
Content-Type: text/plain; charset=utf-8
Host: s3.us.cloud-object-storage.appdomain.cloud
Content-Length: 257
```

```
<CompleteMultipartUpload>
  <Part>
    <PartNumber>1</PartNumber>
    <ETag>"7417ca8d45a71b692168f0419c17fe2f"</ETag>
  </Part>
  <Part>
    <PartNumber>2</PartNumber>
    <ETag>"7417ca8d45a71b692168f0419c17fe2f"</ETag>
  </Part>
</CompleteMultipartUpload>
```

Example response

```
HTTP/1.1 200 OK
Date: Fri, 03 Mar 2017 19:18:44 GMT
X-Clv-Request-Id: c8be10e7-94c4-4c03-9960-6f242b42424d
Accept-Ranges: bytes
Server: Cleversafe/3.9.1.114
X-Clv-S3-Version: 2.5
ETag: "765ba3df36cf24e49f67fc6f689dfc6e-2"
Content-Type: application/xml
Content-Length: 364
```

```
<CompleteMultipartUploadResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Location>http://s3.us.cloud-object-storage.appdomain.cloud/zopse/multipart-object-123</Location>
  <Bucket>some-bucket</Bucket>
  <Key>multipart-object-123</Key>
  <ETag>"765ba3df36cf24e49f67fc6f689dfc6e-2"</ETag>
</CompleteMultipartUploadResult>
```

Abort incomplete multipart uploads

A `DELETE` request issued to an object with query parameter `uploadId` deletes all unfinished parts of a multipart upload.

Note: Not all operations are supported in Satellite environments. For details, see supported Satellite APIs

Syntax

```
DELETE https://{endpoint}/{bucket-name}/{object-name}?uploadId={uploadId}= # path style
DELETE https://{bucket-name}.{endpoint}/{object-name}?uploadId={uploadId}= # virtual host style
```

Example request

```
DELETE /some-bucket/multipart-object-123?uploadId=0000015a-df89-51d0-2790-dee1ac994053 HTTP/1.1
Authorization: Bearer {token}
Host: s3.us.cloud-object-storage.appdomain.cloud
```

Example request

```
DELETE /some-bucket/multipart-object-123?uploadId=0000015a-df89-51d0-2790-dee1ac994053 HTTP/1.1
Authorization: 'AWS4-HMAC-SHA256 Credential={access-key}/{date}/{region}/s3/aws4_request,SignedHeaders=host;x-amz-
date;,Signature={signature}'
x-amz-date: {timestamp}
Host: s3.us.cloud-object-storage.appdomain.cloud
```

Example response

```
HTTP/1.1 204 No Content
Date: Thu, 16 Mar 2017 22:07:48 GMT
X-Clv-Request-Id: 06d67542-6a3f-4616-be25-fc4dbdf242ad
Accept-Ranges: bytes
Server: Cleversafe/3.9.1.114
X-Clv-S3-Version: 2.5
```

Temporarily restore an archived object

A `POST` request that is issued to an object with query parameter `restore` to request temporary restoration of an archived object. A `Content-MD5` header is required as an integrity check for the payload.

An archived object must be restored before downloading or modifying the object. The lifetime of the object must be specified after which the temporary copy of the object will be deleted.

For buckets with a lifecycle policy transition storage class of `GLACIER`, there can be a delay of up to 12 hours before the restored copy is available for access. If the transition storage class was set to `ACCELERATED`, there can be a delay of up to two (2) hours before the restored object is available. A HEAD request can check whether the restored copy is available.

To permanently restore the object, it must be copied to a bucket that doesn't have an active lifecycle configuration.

Note: Not all operations are supported in Satellite environments. For details, see supported Satellite APIs

Syntax

```
POST https://{endpoint}/{bucket-name}/{object-name}?restore # path style
POST https://{bucket-name}.{endpoint}/{object-name}?restore # virtual host style
```

Payload Elements

The body of the request must contain an XML block with the following schema:

Element	Type	Children	Ancestor	Constraint
RestoreRequest	Container	Days, GlacierJobParameters	None	None
Days	Integer	None	RestoreRequest	Specified the lifetime of the temporarily restored object. The minimum number of days that a restored copy of the object can exist is 1. After the restore period has elapsed, temporary copy of the object will be removed.
GlacierJobParameters	String	Tier	RestoreRequest	None
Tier	String	None	GlacierJobParameters	Optional, and if left blank will default to the value associated with the storage tier of the policy that was in place when the object was written. If this value is not left blank, it **must** be set to `Bulk` if the transition storage class for the bucket's lifecycle policy was set to `GLACIER`, and **must** be set to `Accelerated` if the transition storage class was set to `ACCELERATED`.

```
<RestoreRequest>
    <Days>{integer}</Days>
    <GlacierJobParameters>
```

```
        <Tier>Bulk</Tier>
    </GlacierJobParameters>
</RestoreRequest>
```

Example request

```
POST /apiary/queenbee?restore HTTP/1.1
Authorization: {authorization-string}
Content-Type: text/plain
Content-MD5: rgRRGfd/OytcM7O5gIaQ==
Content-Length: 305
Host: s3.us.cloud-object-storage.appdomain.cloud
```

Example request

```
POST /apiary/queenbee?restore HTTP/1.1
Authorization: 'AWS4-HMAC-SHA256 Credential={access-key}/{date}/{region}/s3/aws4_request,SignedHeaders=host;x-amz-
date;,Signature={signature}'
x-amz-date: {timestamp}
Content-MD5: rgRRGfd/OytcM7O5gIaQ==
Content-Length: 305
Host: s3.us.cloud-object-storage.appdomain.cloud
```

```
<RestoreRequest>
    <Days>3</Days>
    <GlacierJobParameters>
        <Tier>Bulk</Tier>
    </GlacierJobParameters>
</RestoreRequest>
```

Example response

```
HTTP/1.1 202 Accepted
Date: Thu, 16 Mar 2017 22:07:48 GMT
X-Clv-Request-Id: 06d67542-6a3f-4616-be25-fc4dbdf242ad
Accept-Ranges: bytes
Server: Cleversafe/3.9.1.114
X-Clv-S3-Version: 2.5
```

Updating metadata

There are two ways to update the metadata on an existing object:

- A `PUT` request with the new metadata and the original object contents
- Running a `COPY` request with the new metadata specifying the original object as the copy source

Tip: All metadata key must be prefixed with `x-amz-meta-`

Note: Not all operations are supported in Satellite environments. For details, see supported Satellite APIs

Using PUT to update metadata

Important: The `PUT` request requires a copy of existing object as the contents are overwritten.

Syntax

```
PUT https://{endpoint}/{bucket-name}/{object-name} # path style
PUT https://{bucket-name}.{endpoint}/{object-name} # virtual host style
```

Example request

```
PUT /apiary/queen-bee HTTP/1.1
Authorization: Bearer {token}
```

```
Content-Type: text/plain; charset=utf-8
Host: s3.us.cloud-object-storage.appdomain.cloud
x-amz-meta-key1: value1
x-amz-meta-key2: value2

Content-Length: 533

 The 'queen' bee is developed from larvae selected by worker bees and fed a
 substance referred to as 'royal jelly' to accelerate sexual maturity. After a
 short while the 'queen' is the mother of nearly every bee in the hive, and
 the colony will fight fiercely to protect her.
```

Example request

```
PUT /apiary/queen-bee HTTP/1.1
Authorization: Bearer {token}
Content-Type: text/plain; charset=utf-8
Content-MD5: M625BaNwd/OytcM7O5gIaQ==
Host: s3.us.cloud-object-storage.appdomain.cloud
x-amz-meta-key1: value1
x-amz-meta-key2: value2

Content-Length: 533

 The 'queen' bee is developed from larvae selected by worker bees and fed a
 substance referred to as 'royal jelly' to accelerate sexual maturity. After a
 short while the 'queen' is the mother of nearly every bee in the hive, and
 the colony will fight fiercely to protect her.
```

Example response

```
HTTP/1.1 200 OK
Date: Thu, 25 Aug 2016 18:30:02 GMT
X-Clv-Request-Id: 9f0ca49a-ae13-4d2d-925b-117b157cf5c3
Accept-Ranges: bytes
Server: Cleversafe/3.9.0.121
X-Clv-S3-Version: 2.5
x-amz-request-id: 9f0ca49a-ae13-4d2d-925b-117b157cf5c3
ETag: "3ca744fa96cb95e92081708887f63de5"
Content-Length: 0
```

Using COPY to update metadata

The complete details about the `COPY` request are here.

Syntax

```
PUT https://{endpoint}/{bucket-name}/{object-name} # path style
PUT https://{bucket-name}.{endpoint}/{object-name} # virtual host style
```

Example request

```
PUT /apiary/queen-bee HTTP/1.1
Authorization: Bearer {token}
Content-Type: text/plain; charset=utf-8
Host: s3.us.cloud-object-storage.appdomain.cloud
x-amz-copy-source: /apiary/queen-bee
x-amz-metadata-directive: REPLACE
x-amz-meta-key1: value1
x-amz-meta-key2: value2
```

Example request

```
PUT /apiary/queen-bee HTTP/1.1
Authorization: 'AWS4-HMAC-SHA256 Credential={access-key}/{date}/{region}/s3/aws4_request,SignedHeaders=host;x-amz-
date;,Signature={signature}'
```

```
x-amz-date: {timestamp}
Content-Type: text/plain
Host: s3.us.cloud-object-storage.appdomain.cloud
x-amz-copy-source: /apiary/queen-bee
x-amz-metadata-directive: REPLACE
x-amz-meta-key1: value1
x-amz-meta-key2: value2
```

Example response

```
HTTP/1.1 200 OK
Date: Thu, 25 Aug 2016 18:30:02 GMT
X-Clv-Request-Id: 9f0ca49a-ae13-4d2d-925b-117b157cf5c3
Accept-Ranges: bytes
Server: Cleversafe/3.9.0.121
X-Clv-S3-Version: 2.5
x-amz-request-id: 9f0ca49a-ae13-4d2d-925b-117b157cf5c3
ETag: "3ca744fa96cb95e92081708887f63de5"
Content-Length: 0
```

Next Steps

Learn more about bucket operations at the documentation.

Using `Postman`

Here's a basic `Postman` setup for the IBM Cloud® Object Storage REST API. More detail can be found in the API reference for buckets or objects.

Using `Postman` assumes a certain amount of familiarity with Object Storage and the necessary information from a service credential or the console as shown in the getting started with IBM Cloud Object Storage. If any terms or variables are unfamiliar, they can be found in the FAQ.

Tip: Personally Identifiable Information (PII): When *naming* buckets or objects, do not use any information that can identify any user (natural person) by name, location, or any other means.

REST API client overview

Interacting with a REST API isn't as simple as using a standard internet browser. Simple browsers do not allow any manipulation of the URL request. A REST API client can help quickly put together both simple and complex HTTP requests.

Prerequisites

- IBM Cloud account
- Cloud Storage resource created (lite plan works fine)
- IBM Cloud COS CLI installed and configured
- Service Instance ID for your Cloud Storage
- IAM (Identity and Access Management) Token
- Endpoint for your COS bucket

Create a bucket

1. Start Postman
2. In the **New** tab, select `PUT`.
3. Enter the endpoint in the address bar and add the name for your new bucket. a. Bucket names must be unique across all buckets, so choose something specific.
4. In the **Type** menu, select Bearer Token.
5. Add the IAM Token in the Token box.
6. Click **Preview Request**. a. You should see a confirmation message that the headers were added.
7. Click the **Header** tab where you should see an existing entry for Authorization.
8. Add a key. a. Key: `ibm-service-instance-id` b. Value: Resource Instance ID for your cloud storage service.
9. Click Send.
10. You'll receive a status `200 OK` message.

Create a text file object

1. Create a tab by clicking the Plus (+) icon.
2. Select `PUT` from the list.
3. In the address bar, enter the endpoint address with the bucket name from previous section and a file name.
4. In the Type list select Bearer Token.
5. Add the IAM Token in the token box.
6. Select the Body tab.
7. Select raw option and ensure that Text is selected.
8. Enter text in the provided space.
9. Click Send.
10. You receive a status `200 OK` message.

List the contents of a bucket

1. Create a new tab by selecting the Plus (+) icon.
2. Verify `GET` is selected in the list.
3. In the address bar, enter the endpoint address with the bucket name from the previous section.
4. In the Type list, select Bearer Token.
5. Add the IAM Token in the token box.
6. Click Send.
7. You receive a status `200 OK` message.
8. In the Body of the Response section is an XML message with the list of files in your bucket.

Using the sample collection

This Postman collection is provided as a starting point for experimenting with the API, and it not intended for production use:

```
$ {
 "info": {
  "_postman_id": "56d99641-9ad6-4218-b3d4-18ac8f3361e0",
  "name": "IBM COS",
  "description": "IBM COS samples",
  "schema": "https://schema.getpostman.com/json/collection/v2.1.0/collection.json"
 },
 "item": [
  {
   "name": "Retrieve list of buckets",
   "event": [
    {
     "listen": "test",
     "script": {
      "id": "d67afcf2-6d35-4a2a-9542-b8d5df78eded",
      "type": "text/javascript",
      "exec": [
       "pm.test(\"Request was successful\", function() {",
       "   pm.response.to.be.success; ",
       "});",
       "",
       "pm.test(\"Response contains expected content\", function() {",
       "    pm.expect(pm.response.text()).to.include(\"ListAllMyBucketsResult\");",
       "});"
      ]
     }
    }
   ],
   "request": {
    "auth": {
     "type": "bearer",
     "bearer": [
      {
       "key": "token",
       "value": "{{iamtoken}}",
       "type": "string"
    }
   ]
  },
  "method": "GET",
  "header": [
   {
    "key": "ibm-service-instance-id",
    "value": "{{serviceid}}"
   }
  ],
  "body": {},
  "url": {
   "raw": "https://{{endpoint-region}}cloud-object-storage.appdomain.cloud",
   "protocol": "https",
   "host": [
    "{{endpoint-region}}",
    "objectstorage",
    "softlayer",
    "net"
   ]
  }
 },
 "response": []
},
{
 "name": "Create new bucket",
 "event": [
  {
   "listen": "test",
   "script": {
    "id": "3cebb9d7-90ee-42c0-9154-b26bd229e179",
    "type": "text/javascript",
    "exec": [
     "pm.test(\"Request was successful\", function() {",
     "   pm.response.to.be.success; ",
     "});"
    ]
   }
  }
 ],
 "request": {
  "auth": {
   "type": "bearer",
   "bearer": [
    {
     "key": "token",
     "value": "{{iamtoken}}",
     "type": "string"
    }
   ]
  },
  "method": "PUT",
  "header": [
   {
    "key": "ibm-service-instance-id",
    "value": "{{serviceid}}"
   }
  ],
  "body": {
   "mode": "raw",
   "raw": ""
  },
  "url": {
   "raw": "https://{{endpoint-region}}cloud-object-storage.appdomain.cloud/{{bucket}}",
   "protocol": "https",
   "host": [
    "{{endpoint-region}}",
    "objectstorage",
    "softlayer",
    "net"
   ],
    "path": [
     "{{bucket}}"
    ]
   },
   "description": "Create new bucket"
  },
  "response": []
 },
 {
  "name": "Create new text file",
  "event": [
   {
    "listen": "test",
    "script": {
     "id": "0a54c09b-0032-4933-ae99-81911935cb4d",
     "type": "text/javascript",
     "exec": [
      "pm.test(\"Request was successful\", function() {",
      "   pm.response.to.be.success; ",
      "});",
      "",
      "pm.test(\"Response contains expected header\", function() {",
      "    pm.response.to.have.header(\"ETag\");",
      "});"
     ]
    }
   }
  ],
  "request": {
   "auth": {
    "type": "bearer",
    "bearer": [
     {
      "key": "token",
      "value": "{{iamtoken}}",
      "type": "string"
     }
    ]
   },
   "method": "PUT",
   "header": [
    {
     "key": "Content-Type",
     "value": "application/x-www-form-urlencoded"
    }
   ],
   "body": {
    "mode": "raw",
    "raw": "This is test data for the text file."
   },
   "url": {
    "raw": "https://{{endpoint-region}}cloud-object-storage.appdomain.cloud/{{bucket}}/testfile.txt",
    "protocol": "https",
    "host": [
     "{{endpoint-region}}",
     "objectstorage",
     "softlayer",
     "net"
    ],
    "path": [
     "{{bucket}}",
     "testfile.txt"
    ]
   },
   "description": "Create a new text file in the bucket"
  },
  "response": []
 },
 {
  "name": "Create new binary file",
  "event": [
   {
    "listen": "test",
    "script": {
     "id": "80e3e8bd-3fb5-4874-a638-510cf5b8c872",
     "type": "text/javascript",
     "exec": [
      "pm.test(\"Request was successful\", function() {",
      "   pm.response.to.be.success; ",
      "});",
      "",
      "pm.test(\"Response contains expected header\", function() {",
      "    pm.response.to.have.header(\"ETag\");",
      "});"
     ]
    }
   }
  ],
  "request": {
   "auth": {
    "type": "bearer",
    "bearer": [
     {
      "key": "token",
      "value": "{{iamtoken}}",
      "type": "string"
     }
    ]
   },
   "method": "PUT",
   "header": [
    {
     "key": "Content-Type",
     "value": "image/jpeg"
    }
   ],
   "body": {
    "mode": "file",
    "file": {}
   },
   "url": {
    "raw": "https://{{endpoint-region}}cloud-object-storage.appdomain.cloud/{{bucket}}/testimage.jpg",
    "protocol": "https",
    "host": [
     "{{endpoint-region}}",
     "objectstorage",
     "softlayer",
     "net"
    ],
    "path": [
     "{{bucket}}",
     "testimage.jpg"
    ]
   },
   "description": "Create a new binary (image) file in the bucket"
  },
  "response": []
 },
 {
  "name": "Retrieve list of files from bucket",
  "event": [
   {
    "listen": "test",
    "script": {
     "id": "aae11dfd-89e5-464d-a6da-44c05652ccec",
     "type": "text/javascript",
     "exec": [
      "pm.test(\"Request was successful\", function() {",
      "   pm.response.to.be.success; ",
      "});",
      "",
      "pm.test(\"Response contains expected content\", function() {",
      "    pm.expect(pm.response.text()).to.include(\"ListBucketResult\");",
      "});"
     ]
    }
   }
  ],
  "request": {
   "auth": {
    "type": "bearer",
    "bearer": [
     {
      "key": "token",
      "value": "{{iamtoken}}",
      "type": "string"
     }
    ]
   },
   "method": "GET",
   "header": [],
   "body": {
    "mode": "raw",
    "raw": ""
   },
   "url": {
    "raw": "https://{{endpoint-region}}cloud-object-storage.appdomain.cloud/{{bucket}}",
    "protocol": "https",
    "host": [
     "{{endpoint-region}}",
     "objectstorage",
     "softlayer",
     "net"
    ],
    "path": [
     "{{bucket}}"
    ]
   },
   "description": "Retrieve the list of files available in the bucket"
  },
  "response": []
 },
 {
  "name": "Retrieve list of files from bucket (filter by prefix)",
  "event": [
   {
    "listen": "test",
    "script": {
     "id": "aae11dfd-89e5-464d-a6da-44c05652ccec",
     "type": "text/javascript",
     "exec": [
      "pm.test(\"Request was successful\", function() {",
      "   pm.response.to.be.success; ",
      "});",
      "",
      "pm.test(\"Response contains expected content\", function() {",
      "    pm.expect(pm.response.text()).to.include(\"ListBucketResult\");",
      "});"
     ]
    }
   }
  ],
  "request": {
   "auth": {
    "type": "bearer",
    "bearer": [
     {
      "key": "token",
      "value": "{{iamtoken}}",
      "type": "string"
     }
    ]
   },
   "method": "GET",
   "header": [],
   "body": {
    "mode": "raw",
    "raw": ""
   },
   "url": {
    "raw": "https://{{endpoint-region}}cloud-object-storage.appdomain.cloud/{{bucket}}?prefix=new",
    "protocol": "https",
    "host": [
     "{{endpoint-region}}",
     "objectstorage",
     "softlayer",
     "net"
    ],
    "path": [
     "{{bucket}}"
    ],
    "query": [
     {
      "key": "prefix",
      "value": "new"
     }
    ]
   },
   "description": "Retrieve the list of files available in the bucket"
  },
  "response": []
 },
 {
  "name": "Retrieve text file",
  "event": [
   {
    "listen": "test",
    "script": {
     "id": "58817cc7-7d15-45a9-b372-7712d2fd389d",
     "type": "text/javascript",
     "exec": [
      "pm.test(\"Request was successful\", function() {",
      "   pm.response.to.be.success; ",
      "});",
      "",
      "pm.test(\"Response contains expected body content\", function() {",
      "    pm.expect(pm.response.text()).to.include(\"This is test data\");",
      "});",
      "",
      "pm.test(\"Response contains expected header\", function() {",
      "    pm.response.to.have.header(\"ETag\");",
      "});"
     ]
    }
   }
  ],
  "request": {
   "auth": {
    "type": "bearer",
    "bearer": [
     {
      "key": "token",
      "value": "{{iamtoken}}",
      "type": "string"
     }
    ]
   },
   "method": "GET",
   "header": [],
   "body": {
    "mode": "raw",
    "raw": ""
   },
   "url": {
    "raw": "https://{{endpoint-region}}cloud-object-storage.appdomain.cloud/{{bucket}}/testfile.txt",
    "protocol": "https",
    "host": [
     "{{endpoint-region}}",
     "objectstorage",
     "softlayer",
     "net"
    ],
    "path": [
     "{{bucket}}",
     "testfile.txt"
    ]
   },
   "description": "Retrieving a file from bucket"
  },
  "response": []
 },
 {
  "name": "Retrieve binary file",
  "event": [
   {
    "listen": "test",
    "script": {
     "id": "db6fa324-080a-43bc-a301-32e70d9bbd65",
     "type": "text/javascript",
     "exec": [
      "pm.test(\"Request was successful\", function() {",
      "   pm.response.to.be.success; ",
      "});",
      "",
      "pm.test(\"Response contains expected header\", function() {",
      "    pm.response.to.have.header(\"ETag\");",
      "});"
     ]
    }
   }
  ],
  "request": {
   "auth": {
    "type": "bearer",
    "bearer": [
     {
      "key": "token",
      "value": "{{iamtoken}}",
      "type": "string"
     }
    ]
   },
   "method": "GET",
   "header": [],
   "body": {
    "mode": "raw",
    "raw": ""
   },
   "url": {
    "raw": "https://{{endpoint-region}}cloud-object-storage.appdomain.cloud/{{bucket}}/testimage.jpg",
    "protocol": "https",
    "host": [
     "{{endpoint-region}}",
     "objectstorage",
     "softlayer",
     "net"
    ],
    "path": [
     "{{bucket}}",
     "testimage.jpg"
    ]
   },
   "description": "Retrieve a binary file from the bucket"
  },
  "response": []
```

```
},
{
 "name": "Retrieve list of failed multipart uploads",
 "event": [
  {
   "listen": "test",
   "script": {
    "id": "835bcaf6-6e7d-4fa9-883e-1a5c5538ac7e",
    "type": "text/javascript",
    "exec": [
     "pm.test(\"Request was successful\", function() {",
     "   pm.response.to.be.success; ",
     "});",
     "",
     "pm.test(\"Response contains expected content\", function() {",
     "    pm.expect(pm.response.text()).to.include(\"ListMultipartUploadsResult\");",
     "});"
    ]
   }
  }
 ],
 "request": {
  "auth": {
   "type": "bearer",
   "bearer": [
    {
     "key": "token",
     "value": "{{iamtoken}}",
     "type": "string"
    }
   ]
  },
  "method": "GET",
  "header": [],
  "body": {
   "mode": "raw",
   "raw": ""
  },
  "url": {
   "raw": "https://{{endpoint-region}}cloud-object-storage.appdomain.cloud/{{bucket}}?uploads=",
   "protocol": "https",
   "host": [
    "{{endpoint-region}}",
    "objectstorage",
    "softlayer",
    "net"
   ],
   "path": [
    "{{bucket}}"
   ],
   "query": [
    {
     "key": "uploads",
     "value": ""
    }
   ]
  },
  "description": "Retrieve the list of files available in the bucket"
 },
 "response": []
},
{
 "name": "Retrieve list of failed multipart uploads (filter by name)",
 "event": [
  {
   "listen": "test",
   "script": {
    "id": "835bcaf6-6e7d-4fa9-883e-1a5c5538ac7e",
    "type": "text/javascript",
    "exec": [
     "pm.test(\"Request was successful\", function() {",
     "   pm.response.to.be.success; ",
     "});",
     "",
     "pm.test(\"Response contains expected content\", function() {",
     "    pm.expect(pm.response.text()).to.include(\"ListMultipartUploadsResult\");",
     "});"
    ]
   }
  }
 ],
 "request": {
  "auth": {
   "type": "bearer",
   "bearer": [
    {
     "key": "token",
     "value": "{{iamtoken}}",
     "type": "string"
    }
   ]
  },
  "method": "GET",
  "header": [],
  "body": {
   "mode": "raw",
   "raw": ""
  },
  "url": {
   "raw": "https://{{endpoint-region}}cloud-object-storage.appdomain.cloud/{{bucket}}?uploads=&prefix=my",
   "protocol": "https",
   "host": [
    "{{endpoint-region}}",
    "objectstorage",
    "softlayer",
    "net"
   ],
   "path": [
    "{{bucket}}"
   ],
   "query": [
    {
     "key": "uploads",
     "value": ""
    },
    {
     "key": "prefix",
     "value": "my"
    }
   ]
  },
  "description": "Retrieve the list of files available in the bucket"
 },
 "response": []
},
{
 "name": "Set CORS enabled bucket",
 "event": [
  {
   "listen": "test",
   "script": {
    "id": "5cf3d531-13f1-4cf6-a4fc-2f8d2d8e48af",
    "type": "text/javascript",
    "exec": [
     "pm.test(\"Request was successful\", function() {",
     "   pm.response.to.be.success; ",
     "});"
    ]
   }
  }
 ],
 "request": {
```

```
    "auth": {
     "type": "bearer",
     "bearer": [
      {
       "key": "token",
       "value": "{{iamtoken}}",
       "type": "string"
      }
     ]
    },
    "method": "PUT",
    "header": [
     {
      "key": "ibm-service-instance-id",
      "value": "{{serviceid}}"
     },
     {
      "key": "Content-MD5",
      "value": "GQmpTNpruOyK6YrxHnpj7g=="
     }
    ],
    "body": {
     "mode": "raw",
     "raw": "<CORSConfiguration>\n  <CORSRule>\n    <AllowedOrigin>http:www.ibm.com</AllowedOrigin>\n
<AllowedMethod>GET</AllowedMethod>\n    <AllowedMethod>PUT</AllowedMethod>\n    <AllowedMethod>POST</AllowedMethod>\n
</CORSRule>\n</CORSConfiguration>"
    },
    "url": {
     "raw": "https://{{endpoint-region}}cloud-object-storage.appdomain.cloud/{{bucket}}?cors=",
     "protocol": "https",
     "host": [
      "{{endpoint-region}}",
      "objectstorage",
      "softlayer",
      "net"
     ],
     "path": [
      "{{bucket}}"
     ],
     "query": [
      {
       "key": "cors",
       "value": ""
      }
     ]
    }
   },
   "response": []
  },
  {
   "name": "Retrieve bucket CORS config",
   "event": [
    {
     "listen": "test",
     "script": {
      "id": "d9c0dce3-decd-4f3e-b5b1-e3f4a89f1283",
      "type": "text/javascript",
      "exec": [
       "pm.test(\"Request was successful\", function() {",
       "   pm.response.to.be.success; ",
       "});",
       "",
       "pm.test(\"Response contains expected content\", function() {",
       "    pm.expect(pm.response.text()).to.include(\"CORSConfiguration\");",
       "});"
      ]
     }
    }
   ],
   "request": {
    "auth": {
```

```
    "type": "bearer",
    "bearer": [
     {
      "key": "token",
      "value": "{{iamtoken}}",
      "type": "string"
     }
    ]
   },
   "method": "GET",
   "header": [],
   "body": {
    "mode": "raw",
    "raw": ""
   },
   "url": {
    "raw": "https://{{endpoint-region}}cloud-object-storage.appdomain.cloud/{{bucket}}?cors=",
    "protocol": "https",
    "host": [
     "{{endpoint-region}}",
     "objectstorage",
     "softlayer",
     "net"
    ],
    "path": [
     "{{bucket}}"
    ],
    "query": [
     {
      "key": "cors",
      "value": ""
     }
    ]
   },
   "description": "Retrieve the list of files available in the bucket"
  },
  "response": []
 },
 {
  "name": "Delete bucket CORS config",
  "event": [
   {
    "listen": "test",
    "script": {
     "id": "83ce0bed-572f-4c16-b45f-67a7a9cc5550",
     "type": "text/javascript",
     "exec": [
      "pm.test(\"Request was successful\", function() {",
      "   pm.response.to.be.success; ",
      "});"
     ]
    }
   }
  ],
  "request": {
   "auth": {
    "type": "bearer",
    "bearer": [
     {
      "key": "token",
      "value": "{{iamtoken}}",
      "type": "string"
     }
    ]
   },
   "method": "DELETE",
   "header": [],
   "body": {
    "mode": "raw",
    "raw": ""
   },
   "url": {
    "raw": "https://{{endpoint-region}}cloud-object-storage.appdomain.cloud/{{bucket}}?cors=",
    "protocol": "https",
    "host": [
     "{{endpoint-region}}",
     "objectstorage",
     "softlayer",
     "net"
    ],
    "path": [
     "{{bucket}}"
    ],
    "query": [
     {
      "key": "cors",
      "value": ""
     }
    ]
   },
   "description": "Retrieve the list of files available in the bucket"
  },
  "response": []
 },
 {
  "name": "Delete text file",
  "event": [
   {
    "listen": "test",
    "script": {
     "id": "78b76a84-9562-4692-9634-e10912574a89",
     "type": "text/javascript",
     "exec": [
      "pm.test(\"Request was successful\", function() {",
      "   pm.response.to.be.success; ",
      "});",
      ""
     ]
    }
   }
  ],
  "request": {
   "auth": {
    "type": "bearer",
    "bearer": [
     {
      "key": "token",
      "value": "{{iamtoken}}",
      "type": "string"
     }
    ]
   },
   "method": "DELETE",
   "header": [],
   "body": {
    "mode": "raw",
    "raw": ""
   },
   "url": {
    "raw": "https://{{endpoint-region}}cloud-object-storage.appdomain.cloud/{{bucket}}/testfile.txt",
    "protocol": "https",
    "host": [
     "{{endpoint-region}}",
     "objectstorage",
     "softlayer",
     "net"
    ],
    "path": [
     "{{bucket}}",
     "testfile.txt"
    ]
   },
   "description": "Retrieving a file from bucket"
  },
  "response": []
 },
 {
  "name": "Delete binary file",
  "event": [
   {
    "listen": "test",
    "script": {
     "id": "a8363887-2b04-4deb-a75b-10220d30e856",
     "type": "text/javascript",
     "exec": [
      "pm.test(\"Request was successful\", function() {",
      "   pm.response.to.be.success; ",
      "});"
     ]
    }
   }
  ],
  "request": {
   "auth": {
    "type": "bearer",
    "bearer": [
     {
      "key": "token",
      "value": "{{iamtoken}}",
      "type": "string"
     }
    ]
   },
   "method": "DELETE",
   "header": [],
   "body": {
    "mode": "raw",
    "raw": ""
   },
   "url": {
    "raw": "https://{{endpoint-region}}cloud-object-storage.appdomain.cloud/{{bucket}}/testimage.jpg",
    "protocol": "https",
    "host": [
     "{{endpoint-region}}",
     "objectstorage",
     "softlayer",
     "net"
    ],
    "path": [
     "{{bucket}}",
     "testimage.jpg"
    ]
   },
   "description": "Retrieve a binary file from the bucket"
  },
  "response": []
 },
 {
  "name": "Delete bucket",
  "event": [
   {
    "listen": "test",
    "script": {
     "id": "3cebb9d7-90ee-42c0-9154-b26bd229e179",
     "type": "text/javascript",
     "exec": [
      "pm.test(\"Request was successful\", function() {",
      "   pm.response.to.be.success; ",
      "});"
     ]
    }
   }
  ],
  "request": {
   "auth": {
    "type": "bearer",
    "bearer": [
     {
      "key": "token",
      "value": "{{iamtoken}}",
      "type": "string"
     }
    ]
   },
   "method": "DELETE",
   "header": [
    {
     "key": "ibm-service-instance-id",
     "value": "{{serviceid}}"
    }
   ],
   "body": {
    "mode": "raw",
    "raw": ""
   },
   "url": {
    "raw": "https://{{endpoint-region}}cloud-object-storage.appdomain.cloud/{{bucket}}",
    "protocol": "https",
    "host": [
     "{{endpoint-region}}",
     "objectstorage",
     "softlayer",
     "net"
    ],
    "path": [
     "{{bucket}}"
    ]
   },
   "description": "Create new bucket"
  },
  "response": []
 },
 {
  "name": "Create new bucket (different storage class)",
  "event": [
   {
    "listen": "test",
    "script": {
     "id": "5cf3d531-13f1-4cf6-a4fc-2f8d2d8e48af",
     "type": "text/javascript",
     "exec": [
      "pm.test(\"Request was successful\", function() {",
      "   pm.response.to.be.success; ",
      "});"
     ]
    }
   }
  ],
  "request": {
   "auth": {
    "type": "bearer",
    "bearer": [
     {
      "key": "token",
      "value": "{{iamtoken}}",
      "type": "string"
     }
    ]
   },
   "method": "PUT",
   "header": [
    {
     "key": "ibm-service-instance-id",
     "value": "{{serviceid}}"
     },
     {
      "key": "Content-Type",
      "value": "application/x-www-form-urlencoded"
     }
    ],
    "body": {
     "mode": "raw",
     "raw": "<CreateBucketConfiguration>\n\t<LocationConstraint>{{bucketlocationvault}}
</LocationConstraint>\n</CreateBucketConfiguration>"
    },
    "url": {
     "raw": "https://{{endpoint-region}}cloud-object-storage.appdomain.cloud/{{bucket}}vault",
     "protocol": "https",
     "host": [
      "{{endpoint-region}}",
      "objectstorage",
      "softlayer",
      "net"
     ],
     "path": [
      "{{bucket}}vault"
     ]
    }
   },
   "response": []
  },
  {
   "name": "Delete bucket (different storage class)",
   "event": [
    {
     "listen": "test",
     "script": {
      "id": "5cf3d531-13f1-4cf6-a4fc-2f8d2d8e48af",
      "type": "text/javascript",
      "exec": [
       "pm.test(\"Request was successful\", function() {",
       "   pm.response.to.be.success; ",
       "});"
      ]
     }
    }
   ],
   "request": {
    "auth": {
     "type": "bearer",
     "bearer": [
      {
       "key": "token",
       "value": "{{iamtoken}}",
       "type": "string"
      }
     ]
    },
    "method": "DELETE",
    "header": [
     {
      "key": "ibm-service-instance-id",
      "value": "{{serviceid}}"
     }
    ],
    "body": {
     "mode": "raw",
     "raw": ""
    },
    "url": {
     "raw": "https://{{endpoint-region}}cloud-object-storage.appdomain.cloud/{{bucket}}vault",
     "protocol": "https",
     "host": [
      "{{endpoint-region}}",
      "objectstorage",
     "softlayer",
     "net"
    ],
    "path": [
     "{{bucket}}vault"
    ]
   }
  },
  "response": []
 },
 {
  "name": "Create new bucket (key protect)",
  "event": [
   {
    "listen": "test",
    "script": {
     "id": "5cf3d531-13f1-4cf6-a4fc-2f8d2d8e48af",
     "type": "text/javascript",
     "exec": [
      "pm.test(\"Request was successful\", function() {",
      "   pm.response.to.be.success; ",
      "});"
     ]
    }
   }
  ],
  "request": {
   "auth": {
    "type": "bearer",
    "bearer": [
     {
      "key": "token",
      "value": "{{iamtoken}}",
      "type": "string"
     }
    ]
   },
   "method": "PUT",
   "header": [
    {
     "key": "ibm-service-instance-id",
     "value": "{{serviceid}}"
    },
    {
     "key": "ibm-sse-kp-encryption-algorithm",
     "value": "AES256"
    },
    {
     "key": "ibm-sse-kp-customer-root-key-crn",
     "value": "{{rootkeycrn}}"
    }
   ],
   "body": {
    "mode": "raw",
    "raw": ""
   },
   "url": {
    "raw": "https://{{endpoint-region}}cloud-object-storage.appdomain.cloud/{{bucket}}kp",
    "protocol": "https",
    "host": [
     "{{endpoint-region}}",
     "objectstorage",
     "softlayer",
     "net"
    ],
    "path": [
     "{{bucket}}kp"
    ]
   }
  },
  "response": []
  },
  {
   "name": "Delete bucket (key protect)",
   "event": [
    {
     "listen": "test",
     "script": {
      "id": "5cf3d531-13f1-4cf6-a4fc-2f8d2d8e48af",
      "type": "text/javascript",
      "exec": [
       "pm.test(\"Request was successful\", function() {",
       "   pm.response.to.be.success; ",
       "});"
      ]
     }
    }
   ],
   "request": {
    "auth": {
     "type": "bearer",
     "bearer": [
      {
       "key": "token",
       "value": "{{iamtoken}}",
       "type": "string"
      }
     ]
    },
    "method": "DELETE",
    "header": [
     {
      "key": "ibm-service-instance-id",
      "value": "{{serviceid}}"
     }
    ],
    "body": {
     "mode": "raw",
     "raw": ""
    },
    "url": {
     "raw": "https://{{endpoint-region}}cloud-object-storage.appdomain.cloud/{{bucket}}kp",
     "protocol": "https",
     "host": [
      "{{endpoint-region}}",
      "objectstorage",
      "softlayer",
      "net"
     ],
     "path": [
      "{{bucket}}kp"
     ]
    }
   },
   "response": []
  }
 ],
 "event": [
  {
   "listen": "prerequest",
   "script": {
    "id": "08caf505-3991-4273-8027-db00d867680f",
    "type": "text/javascript",
    "exec": [
     ""
    ]
   }
  },
  {
   "listen": "test",
   "script": {
    "id": "6e61fa13-e3b9-42f0-8c38-9d3468748922",
    "type": "text/javascript",
    "exec": [
     ""
    ]
   }
  }
 ],
 "variable": [
  {
   "id": "643d5480-e629-4e26-b244-1b3c3a85195a",
   "key": "bucket",
   "value": "jsaitocosbucketapitest41",
   "type": "string",
   "description": ""
  },
  {
   "id": "7554e40a-f4d5-4938-972e-43b28ce52ad1",
   "key": "serviceid",
   "value": "crn:v1:bluemix:public:cloud-object-storage:global:a/1d524cd94a0dda86fd8eff3191340732:8888b05b-a143-4917-9d8e-
9d5b326a1604::",
   "type": "string",
   "description": ""
  },
  {
   "id": "f06bc8b2-3476-47a6-aacb-c603dd808310",
   "key": "iamtoken",
   "value":
"eyJraWQiOiIyMDE3MTAzMC0wMDowMDowMCIsImFsZyI6IlJTMjU2In0.eyJpYW1faWQiOiJJQk1pZC01MFkxNjdNOFRZIiwiaWQiOiJJQk1pZC01MFkxNjdNOFRZIiwicmVhbG1pZCI6I
-nWst3O9o6iEJfm0AzJwYNTZqWVru1pjI-KcAPXFBe503DPIf6cYolAw4rarU5booW-pdzk8-R5HZ7MJK7b8sxJtm7PAilXVZvl5yFE-
tJsFkeMH6XCIj_R9i6dwSemDBL3Juq79_x3KsgJGFg37p5f2vck1_7gR7nSb03m8m3mCvrrx7zGkLDuM8NXAVlwwxKcitwlG_UfEBwSbX3krF04zF2tFCpGkcWAnuaFdaVTOL6uaULxCi5

   "type": "string",
   "description": ""
  },
  {
   "id": "dba24a46-e1df-4201-9a9b-f247fff22315",
   "key": "endpoint-region",
   "value": "s3.us-south",
   "type": "string",
   "description": ""
  },
  {
   "id": "ca751949-eea0-4f36-9f58-aea1ed324cd5",
   "key": "rootkeycrn",
   "value": "crn:v1:bluemix:public:kms:us-south:a/1d524cd94a0dda86fd8eff3191340732:90b7a1db-0fe2-4de9-b90e-
922c127ff530:key:0b43e36e-a863-40e2-b713-5caa2bf99288",
   "type": "string",
   "description": ""
  },
  {
   "id": "e486732e-c9bc-4ec4-8bf9-04b66d513e5a",
   "key": "bucketlocationvault",
   "value": "us-south-vault",
   "type": "string",
   "description": ""
  }
 ]
}
```

Import the collection to Postman

1. In Postman click **Import** in the upper right corner
2. Import the Collection file by using either of these methods:
   - From the Import window drag the Collection file into the window labeled **Drop files here**
   - Click the Choose Files button and browse to the folder and select the Collection file
3. *IBM COS* now appears in the Collections window
4. Expand the Collection and see 20 sample requests
5. Click the three dots to the right of the collection to expand the menu and click **Edit**

6. Edit the variables to match your Cloud Storage environment
    - `bucket` - Enter the name for the new bucket you want to create (bucket names must be unique across Cloud Storage).
    - `serviceid` - Enter the CRN of your Cloud Storage service. Instructions to obtain your CRN are available here.
    - `iamtoken` - Enter the OAUTH token for your Cloud Storage service. Instructions to obtain your OAUTH token are available here.
    - `endpoint` - Enter the regional endpoint for your Cloud Storage service. Obtain the available endpoints from the IBM Cloud Dashboard
        - *Ensure that your selected endpoint matches your key protect service to ensure that the samples run correctly*
    - `rootkeycrn` - The CRN of the Root Key created in your primary Key Protect service.
        - The CRN resembles `crn:v1:bluemix:public:kms:us-south:a/3d624cd74a0dea86ed8efe3101341742:90b6a1db-0fe1-4fe9-b91e-962c327df531:key:0bg3e33e-a866-50f2-b715-5cba2bc93234`
        - *Ensure the Key Protect service that is selected matches the region of the Endpoint*
    - `bucketlocationvault` - Enter the location constraint value for the bucket creation for the *Create New Bucket (different storage class)* API request.
        - Acceptable values include:
            - `us-south-vault`
            - `us-standard-flex`
            - `eu-cold`
7. Click Update

Running the samples

The API sample requests are fairly straightforward and easy to use. They're designed to run in order and demonstrate how to interact with Cloud Storage. You can also run a functional test against your Cloud Storage service to ensure proper operation.

Request	Expected Result	Test Results
Retrieve list of buckets	In the Body, you set an XML list of the buckets in your cloud storage.	Request was successful. Response contains expected content
Create new bucket	Status Code 200 OK	Request was successful
Create new text file	Status Code 200 OK	Request was successful. Response contains expected header
Create new binary file	Click **Body** and click **Choose File** to select an image to upload.	Request was successful. Response contains expected header
Retrieve list of files from bucket	In the Body of the response you see the two files you created in the previous requests.	Request was successful. Response contains expected header
Retrieve list of files from bucket (filter by prefix)	Change the query string value to `prefix=<some text>`. In the body of the response you see the files with names that start with the prefix specified.	Request was successful. Response contains expected header
Retrieve text file	In the Body of the response you see the text you entered in the previous request	Request was successful. Response contains expected body content. Response contains expected header
Retrieve binary file	In the Body of the response you see the image you chose in the previous request.	Request was successful. Response contains expected header
Retrieve list of failed multipart uploads	In the Body of the response you see any failed multipart uploads for the bucket.	Request was successful. Response contains expected content
Retrieve list of failed multipart uploads (filter by name)	Change the query string value to `prefix=<some text>`. In the body of the response you see any failed multipart uploads for the bucket with names that start with the prefix specified.	Request was successful. Response contains expected content
Set CORS enabled bucket	Status Code 200 OK	Request was successful

Retrieve bucket CORS config	In the body of the response you see the CORS configuration set for the bucket	Request was successful. Response contains expected content
Delete bucket CORS config	Status Code 200 OK	Request was successful
Delete text file	Status Code 200 OK	Request was successful
Delete binary file	Status Code 200 OK	Request was successful
Delete bucket	Status Code 200 OK	Request was successful
Create new bucket (different storage class)	Status Code 200 OK	Request was successful
Delete bucket (different storage class)	Status Code 200 OK	Request was successful
Create new bucket (key protect)	Status Code 200 OK	Request was successful
Delete bucket (key protect)	Status Code 200 OK	Request was successful

Using the Postman Collection Runner

The Postman Collection Runner provides a user interface for testing a collection and allows you to run all requests in a Collection at once.

1. Click the Runner button in the upper right corner on the main Postman window.
2. In the Runner window, select the IBM COS collection and click the big blue **run IBM COS** button at the bottom of the screen.
3. The Collection Runner window will show the iterations as the requests are run. You will see that the test results appear below each of the requests.
    - The **Run Summary** displays a grid view of the requests and allows filtering of the results.
    - You can also click **Export Results** to save the results to a JSON file.

Libraries

About IBM COS SDKs

IBM Cloud® Object Storage provides SDKs for Java, Python, NodeJS, and Go featuring capabilities to make the most of IBM Cloud Object Storage.

These SDKs are based on the official AWS S3 API SDKs, but are modified to use IBM Cloud features like IAM, Key Protect, Immutable Object Storage, and others.

Feature	Java	Python	NodeJS	GO	CLI	Terraform
IAM API key support	✓	✓	✓	✓	✓	✓
Managed multipart uploads	✓	✓	✓	✓	✓	
Managed multipart downloads	✓	✓	✓	✓		
Extended bucket listing	✓	✓	✓	✓	✓	
Version 2 object listing	✓	✓	✓	✓		
Key Protect	✓	✓	✓	✓		✓
SSE-C	✓	✓	✓	✓		
Archive rules	✓	✓	✓	✓		✓
Retention policies	✓	✓	✓	✓		✓
Aspera high-speed transfer	✓	✓				

Features supported per SDK

IAM API key support

Allows for creating clients with an API key instead of a pair of Access and Secret keys. Token management is handled automatically, and tokens are automatically refreshed during long-running operations.

Managed multipart uploads

Using a `TransferManager` class, the SDK handles all the necessary logic for uploading objects in parallel parts.

Managed multipart downloads

Using a `TransferManager` class, the SDK handles all the necessary logic for downloading objects in parallel parts.

Extended bucket listing

This extension to the S3 API returns a list of buckets with their `LocationConstraint`. All buckets in a service instance are always returned on a list request, not just the subset that is located in the region of the target endpoint. This API is useful for finding where a bucket is located.

Version 2 object listing

Version 2 listing allows for more powerful scoping of object listings.

Key Protect

IBM® Key Protect for IBM Cloud® helps you create encrypted keys for apps across IBM Cloud® services. Keys are secured by FIPS 140-2 Level 3 cloud-based hardware security modules (HSMs) that protect against the theft of information. Hyper Protect Crypto Services is a single-tenant, dedicated HSM that is controlled by you. The service is built on FIPS 140-2 Level 4 hardware, the highest offered by any cloud provider in the industry.

SSE-C

IBM Cloud® Object Storage provides several options to encrypt your data. By default, all objects that are stored in IBM Cloud Object Storage are encrypted by using randomly generated keys and an all-or-nothing-transform (AONT). While this default encryption model provides at-rest security, some workloads need full control over the data encryption keys used. You can manage your keys manually by supplying your own encryption keys - referred to as Server-Side Encryption with Customer-Provided Keys (SSE-C).

Archive rules

IBM Cloud® Object Storage Archive is a low-cost option for data that is rarely accessed. You can migrate data from any of the storage tiers (Standard, Vault, Cold Vault, and Flex) to a long-term offline archive.

Retention policies

Immutable Object Storage maintains data integrity in a WORM (Write-Once-Read-Many) manner. Objects can't be modified until the end of their retention period and the removal of any legal holds.

Aspera high-speed transfer

Aspera high-speed transfer improves data transfer performance under most conditions, especially in networks with high latency or packet loss. Instead of the standard HTTP `PUT`, Aspera high-speed transfer uploads the object by using the FASP protocol.

Getting Started with the SDKs

IBM Cloud® Object Storage provides SDKs for Java, Python, NodeJS, and Go which can help you to make the most of Object Storage.

This Quick Start guide provides a code example that demonstrates the following operations:

- Create a new bucket
- List the available buckets
- Create a new text file
- List the available files
- Retrieve the text file contents
- Upload a large binary file
- Delete a file
- Delete a bucket

Before you begin

You need:

- An IBM Cloud® Platform account
- An instance of IBM Cloud Object Storage
- An IAM API key with Writer access to your Object Storage

Getting the SDK

Specific instructions for downloading and installing the SDK is available in Using Python Using Node.js Using Java Using Go.

Code Example

The code examples below provide introductory examples of running the basic operations with Object Storage. For simplicity, the code example can be run multiple times as it uses Universally Unique Identifiers (UUIDs) for bucket/item names to prevent potential conflicts.

Note: In your code, you must remove the angled brackets or any other excess characters that are provided here as illustration.

To complete the code example, you need to replace the following values:

Value	Description	Example

`<endpoint>`	Regional endpoint for your COS instance	`s3.us-south.cloud-object-storage.appdomain.cloud`
`<api-key>`	IAM API Key with at least `Writer` permissions	`xxxd12V2QHXbjaM99G9tWyYDgF_0gYdlQ8aWALIQxXx4`
`<resource-instance-id>`	Unique ID for the Service Instance	`crn:v1:bluemix:public:cloud-object-storage:global:a/xx999cd94a0dda86fd8eff3191349999:9999b05b-x999-4917-xxxx-9d5b326a1111::`
`<storage-class>`	Storage class for a new bucket	`us-south-standard`

For more information about endpoints, see [Endpoints and storage locations]().

Code examples are tested on supported release versions of Python.

Python

```
import os
import uuid
import ibm_boto3
from ibm_botocore.client import Config
from ibm_botocore.exceptions import ClientError
import ibm_s3transfer.manager

def log_done():
    print("DONE!\n")

def log_client_error(e):
    print("CLIENT ERROR: {0}\n".format(e))

def log_error(msg):
    print("UNKNOWN ERROR: {0}\n".format(msg))

def get_uuid():
    return str(uuid.uuid4().hex)

def generate_big_random_file(file_name, size):
    with open('%s'%file_name, 'wb') as fout:
        fout.write(os.urandom(size))

# Retrieve the list of available buckets
def get_buckets():
    print("Retrieving list of buckets")
    try:
        bucket_list = cos_cli.list_buckets()
        for bucket in bucket_list["Buckets"]:
            print("Bucket Name: {0}".format(bucket["Name"]))
        log_done()
    except ClientError as be:
        log_client_error(be)
    except Exception as e:
        log_error("Unable to retrieve list buckets: {0}".format(e))

# Retrieve the list of contents for a bucket
def get_bucket_contents(bucket_name):
    print("Retrieving bucket contents from: {0}".format(bucket_name))
    try:
        file_list = cos_cli.list_objects(Bucket=bucket_name)
        for file in file_list.get("Contents", []):
            print("Item: {0} ({1} bytes).".format(file["Key"], file["Size"]))
        else:
            print("Bucket {0} has no items.".format(bucket_name))
        log_done()
    except ClientError as be:
        log_client_error(be)
    except Exception as e:
        log_error("Unable to retrieve bucket contents: {0}".format(e))

# Retrieve a particular item from the bucket
def get_item(bucket_name, item_name):
    print("Retrieving item from bucket: {0}, key: {1}".format(bucket_name, item_name))
    try:
        file = cos_cli.get_object(Bucket=bucket_name, Key=item_name)
        print("File Contents: {0}".format(file["Body"].read()))
        log_done()
    except ClientError as be:
        log_client_error(be)
    except Exception as e:
        log_error("Unable to retrieve file contents for {0}:\n{1}".format(item_name, e))

# Create new bucket
def create_bucket(bucket_name):
    print("Creating new bucket: {0}".format(bucket_name))
    try:
        cos_cli.create_bucket(
            Bucket=bucket_name,
            CreateBucketConfiguration={
                "LocationConstraint":COS_STORAGE_CLASS
            }
        )
        print("Bucket: {0} created!".format(bucket_name))
        log_done()
    except ClientError as be:
        log_client_error(be)
    except Exception as e:
        log_error("Unable to create bucket: {0}".format(e))

# Create new text file
def create_text_file(bucket_name, item_name, file_text):
    print("Creating new item: {0} in bucket: {1}".format(item_name, bucket_name))
    try:
        cos_cli.put_object(
            Bucket=bucket_name,
            Key=item_name,
            Body=file_text
        )
        print("Item: {0} created!".format(item_name))
        log_done()
    except ClientError as be:
        log_client_error(be)
    except Exception as e:
        log_error("Unable to create text file: {0}".format(e))

# Delete item
def delete_item(bucket_name, item_name):
    print("Deleting item: {0} from bucket: {1}".format(item_name, bucket_name))
    try:
        cos_cli.delete_object(
            Bucket=bucket_name,
            Key=item_name
        )
        print("Item: {0} deleted!".format(item_name))
        log_done()
    except ClientError as be:
        log_client_error(be)
    except Exception as e:
        log_error("Unable to delete item: {0}".format(e))

# Delete bucket
def delete_bucket(bucket_name):
    print("Deleting bucket: {0}".format(bucket_name))
    try:
        cos_cli.delete_bucket(Bucket=bucket_name)
        print("Bucket: {0} deleted!".format(bucket_name))
        log_done()
    except ClientError as be:
        log_client_error(be)
    except Exception as e:
        log_error("Unable to delete bucket: {0}".format(e))

def upload_large_file(bucket_name, item_name, file_path):
    print("Starting large file upload for {0} to bucket: {1}".format(item_name, bucket_name))

    # set the chunk size to 5 MB
    part_size = 1024 * 1024 * 5

    # set threadhold to 5 MB
    file_threshold = 1024 * 1024 * 5

    # set the transfer threshold and chunk size in config settings
    transfer_config = ibm_boto3.s3.transfer.TransferConfig(
        multipart_threshold=file_threshold,
        multipart_chunksize=part_size
    )

    # create transfer manager
    transfer_mgr = ibm_boto3.s3.transfer.TransferManager(cos_cli, config=transfer_config)

    try:
        # initiate file upload
        future = transfer_mgr.upload(file_path, bucket_name, item_name)

        # wait for upload to complete
        future.result()

        print ("Large file upload complete!")
    except Exception as e:
        print("Unable to complete large file upload: {0}".format(e))
    finally:
        transfer_mgr.shutdown()

# Constants for IBM COS values
COS_ENDPOINT = "<endpoint>" # example: https://s3.us-south.cloud-object-storage.appdomain.cloud
COS_API_KEY_ID = "<api-key>" # example: xxxd12V2QHXbjaM99G9tWyYDgF_0gYdlQ8aWALIQxXx4
COS_AUTH_ENDPOINT = "https://iam.cloud.ibm.com/identity/token"
COS_SERVICE_CRN = "<resource-instance-id>" # example: crn:v1:bluemix:public:cloud-object-
storage:global:a/xx999cd94a0dda86fd8eff3191349999:9999b05b-x999-4917-xxxx-9d5b326a1111::
COS_STORAGE_CLASS = "<storage-class>" # example: us-south-standard

# Create client connection
cos_cli = ibm_boto3.client("s3",
    ibm_api_key_id=COS_API_KEY_ID,
    ibm_service_instance_id=COS_SERVICE_CRN,
    ibm_auth_endpoint=COS_AUTH_ENDPOINT,
    config=Config(signature_version="oauth"),
    endpoint_url=COS_ENDPOINT
)

# *** Main Program ***
def main():
    try:
        new_bucket_name = "py.bucket." + get_uuid()
        new_text_file_name = "py_file_" + get_uuid() + ".txt"
        new_text_file_contents = "This is a test file from Python code sample!!!"
        new_large_file_name = "py_large_file_" + get_uuid() + ".bin"
        new_large_file_size = 1024 * 1024 * 20

        # create a new bucket
        create_bucket(new_bucket_name)

        # get the list of buckets
        get_buckets()

        # create a new text file
        create_text_file(new_bucket_name, new_text_file_name, new_text_file_contents)

        # get the list of files from the new bucket
        get_bucket_contents(new_bucket_name)
        # get the text file contents
        get_item(new_bucket_name, new_text_file_name)

        # create a new local binary file that is 20 MB
        generate_big_random_file(new_large_file_name, new_large_file_size)

        # upload the large file using transfer manager
        upload_large_file(new_bucket_name, new_large_file_name, new_large_file_name)

        # get the list of files from the new bucket
        get_bucket_contents(new_bucket_name)

        # remove the two new files
        delete_item(new_bucket_name, new_large_file_name)
        delete_item(new_bucket_name, new_text_file_name)

        # remove the new bucket
        delete_bucket(new_bucket_name)
    except Exception as e:
        log_error("Main Program Error: {0}".format(e))

if __name__ == "__main__":
    main()
```

Node

```
'use strict';

// Required libraries
const ibm = require('ibm-cos-sdk');
const fs = require('fs');
const async = require('async');
const uuidv1 = require('uuid/v1');
const crypto = require('crypto');

function logError(e) {
    console.log(`ERROR: ${e.code} - ${e.message}\n`);
}

function logDone() {
    console.log('DONE!\n');
}

function getUUID() {
    return uuidv1().toString().replace(/-/g, "");
}

function generateBigRandomFile(fileName, size) {
    return new Promise(function(resolve, reject) {
        crypto.randomBytes(size, (err, buf) => {
            if (err) reject(err);

            fs.writeFile(fileName, buf, function (err) {
                if (err) {
                    reject(err);
                }
                else {
                    resolve();
                }
            });
        });
    });
}

// Retrieve the list of available buckets
function getBuckets() {
    console.log('Retrieving list of buckets');
    return cos.listBuckets()
    .promise()
    .then((data) => {
        if (data.Buckets != null) {
            for (var i = 0; i < data.Buckets.length; i++) {
                console.log(`Bucket Name: ${data.Buckets[i].Name}`);
            }
            logDone();
        }
    })
    .catch((logError));
}

// Retrieve the list of contents for a bucket
function getBucketContents(bucketName) {
    console.log(`Retrieving bucket contents from: ${bucketName}`);
    return cos.listObjects(
        {Bucket: bucketName},
    ).promise()
    .then((data) => {
        if (data != null && data.Contents != null) {
            for (var i = 0; i < data.Contents.length; i++) {
                var itemKey = data.Contents[i].Key;
                var itemSize = data.Contents[i].Size;
                console.log(`Item: ${itemKey} (${itemSize} bytes).`)
            }
            logDone();
        }
    })
    .catch(logError);
}

// Retrieve a particular item from the bucket
function getItem(bucketName, itemName) {
    console.log(`Retrieving item from bucket: ${bucketName}, key: ${itemName}`);
    return cos.getObject({
        Bucket: bucketName,
        Key: itemName
    }).promise()
    .then((data) => {
        if (data != null) {
            console.log('File Contents: ' + Buffer.from(data.Body).toString());
            logDone();
        }
    })
    .catch(logError);
}

// Create new bucket
function createBucket(bucketName) {
    console.log(`Creating new bucket: ${bucketName}`);
    return cos.createBucket({
        Bucket: bucketName,
        CreateBucketConfiguration: {
          LocationConstraint: COS_STORAGE_CLASS
        },
    }).promise()
    .then((() => {
        console.log(`Bucket: ${bucketName} created!`);
        logDone();
    }))
    .catch(logError);
}

// Create new text file
function createTextFile(bucketName, itemName, fileText) {
    console.log(`Creating new item: ${itemName}`);
    return cos.putObject({
        Bucket: bucketName,
        Key: itemName,
        Body: fileText
    }).promise()
    .then(() => {
        console.log(`Item: ${itemName} created!`);
        logDone();
    })
    .catch(logError);
}

// Delete item
function deleteItem(bucketName, itemName) {
    console.log(`Deleting item: ${itemName}`);
    return cos.deleteObject({
        Bucket: bucketName,
        Key: itemName
    }).promise()
    .then(() =>{
        console.log(`Item: ${itemName} deleted!`);
        logDone();
    })
    .catch(logError);
}

// Delete bucket
function deleteBucket(bucketName) {
    console.log(`Deleting bucket: ${bucketName}`);
    return cos.deleteBucket({
        Bucket: bucketName
    }).promise()
    .then(() => {
        console.log(`Bucket: ${bucketName} deleted!`);
        logDone();
    })
    .catch(logError);
}

// Multi part upload
function multiPartUpload(bucketName, itemName, filePath) {
    var uploadID = null;

    if (!fs.existsSync(filePath)) {
        logError(new Error(`The file \'${filePath}\' does not exist or is not accessible.`));
        return;
    }

    return new Promise(function(resolve, reject) {
        console.log(`Starting multi-part upload for ${itemName} to bucket: ${bucketName}`);
        return cos.createMultipartUpload({
            Bucket: bucketName,
            Key: itemName
        }).promise()
        .then((data) => {
            uploadID = data.UploadId;

            //begin the file upload
            fs.readFile(filePath, (e, fileData) => {
                //min 5MB part
                var partSize = 1024 * 1024 * 5;
                var partCount = Math.ceil(fileData.length / partSize);
                async.timesSeries(partCount, (partNum, next) => {
                    var start = partNum * partSize;
                    var end = Math.min(start + partSize, fileData.length);
                    partNum++;

                    console.log(`Uploading to ${itemName} (part ${partNum} of ${partCount})`);

                    cos.uploadPart({
                        Body: fileData.slice(start, end),
                        Bucket: bucketName,
                        Key: itemName,
                        PartNumber: partNum,
                        UploadId: uploadID
                    }).promise()
                    .then((data) => {
                        next(e, {ETag: data.ETag, PartNumber: partNum});
                    })
                    .catch((e) => {
                        cancelMultiPartUpload(bucketName, itemName, uploadID);
                        logError(e);
                        reject(e);
                    });
                }, (e, dataPacks) => {
                    cos.completeMultipartUpload({
                        Bucket: bucketName,
                        Key: itemName,
                        MultipartUpload: {
                            Parts: dataPacks
                        },
                        UploadId: uploadID
                    }).promise()
                    .then(() => {
                        logDone();
                        resolve();
                    })
                    .catch((e) => {
                        cancelMultiPartUpload(bucketName, itemName, uploadID);
                        logError(e);
                        reject(e);
                    });
                });
            });
        })
        .catch((e) => {
            logError(e);
            reject(e);
        });
    });
}

function cancelMultiPartUpload(bucketName, itemName, uploadID) {
    return cos.abortMultipartUpload({
        Bucket: bucketName,
        Key: itemName,
        UploadId: uploadID
    }).promise()
    .then(() => {
        console.log(`Multi-part upload aborted for ${itemName}`);
    })
    .catch(logError);
}

// Constants for IBM COS values
const COS_ENDPOINT = "<endpoint>";  // example: s3.us-south.cloud-object-storage.appdomain.cloud
const COS_API_KEY_ID = "<api-key>";  // example: xxxd12V2QHXbjaM99G9tWyYDgF_0gYdlQ8aWALIQxXx4
const COS_AUTH_ENDPOINT = "https://iam.cloud.ibm.com/identity/token";
const COS_SERVICE_CRN = "<resource-instance-id>"; // example: crn:v1:bluemix:public:cloud-object-
storage:global:a/<CREDENTIAL_ID_AS_GENERATED>:<SERVICE_ID_AS_GENERATED>::
const COS_STORAGE_CLASS = "<storage-class>"; // example: us-south-standard

// Init IBM COS library
var config = {
    endpoint: COS_ENDPOINT,
    apiKeyId: COS_API_KEY_ID,
    ibmAuthEndpoint: COS_AUTH_ENDPOINT,
    serviceInstanceId: COS_SERVICE_CRN,
    signatureVersion: 'iam'
};

var cos = new ibm.S3(config);

// Main app
function main() {
    try {
        var newBucketName = "js.bucket." + getUUID();
        var newTextFileName = "js_file_" + getUUID() + ".txt";
        var newTextFileContents = "This is a test file from Node.js code sample!!!";
        var newLargeFileName = "js_large_file_" + getUUID() + ".bin";
        var newLargeFileSize = 1024 * 1024 * 20;

        createBucket(newBucketName) // create a new bucket
            .then(() => getBuckets()) // get the list of buckets
            .then(() => createTextFile(newBucketName, newTextFileName, newTextFileContents)) // create a new text file
            .then(() => getBucketContents(newBucketName)) // get the list of files from the new bucket
            .then(() => getItem(newBucketName, newTextFileName)) // get the text file contents
            .then(() => generateBigRandomFile(newLargeFileName, newLargeFileSize)) // create a new local binary file that is 20
MB
            .then(() => multiPartUpload(newBucketName, newLargeFileName, newLargeFileName)) // upload the large file using
transfer manager
            .then(() => getBucketContents(newBucketName)) // get the list of files from the new bucket
            .then(() => deleteItem(newBucketName, newLargeFileName)) // remove the large file
            .then(() => deleteItem(newBucketName, newTextFileName)) // remove the text file
            .then(() => deleteBucket(newBucketName)); // remove the new bucket
    }
    catch(ex) {
        logError(ex);
    }
}

main();
```

Java

```
// Required libraries
import com.ibm.cloud.objectstorage.ClientConfiguration;
import com.ibm.cloud.objectstorage.SDKGlobalConfiguration;
import com.ibm.cloud.objectstorage.SdkClientException;
import com.ibm.cloud.objectstorage.auth.AWSCredentials;
import com.ibm.cloud.objectstorage.auth.AWSStaticCredentialsProvider;
import com.ibm.cloud.objectstorage.client.builder.AwsClientBuilder;
import com.ibm.cloud.objectstorage.oauth.BasicIBMOAuthCredentials;
import com.ibm.cloud.objectstorage.services.s3.AmazonS3;
import com.ibm.cloud.objectstorage.services.s3.AmazonS3ClientBuilder;
import com.ibm.cloud.objectstorage.services.s3.model.*;
import com.ibm.cloud.objectstorage.services.s3.transfer.TransferManager;
import com.ibm.cloud.objectstorage.services.s3.transfer.TransferManagerBuilder;
import com.ibm.cloud.objectstorage.services.s3.transfer.Upload;

import java.io.ByteArrayInputStream;
import java.io.File;
import java.io.IOException;
import java.io.InputStream;
import java.nio.charset.Charset;
import java.sql.Timestamp;
import java.util.List;
import java.util.UUID;

public class JavaExampleCode {
    private static AmazonS3 _cosClient;
    private static String api_key;
    private static String service_instance_id;
    private static String endpoint_url;
    private static String location;

    public static void main(String[] args) throws IOException
    {
        // Creating a random UUID (Universally unique identifier).
        UUID uuid = UUID.randomUUID();

        // Constants for IBM COS values
        SDKGlobalConfiguration.IAM_ENDPOINT = "https://iam.cloud.ibm.com/oidc/token";
        api_key = "<api-key>"; // example: xxxd12V2QHXbjaM99G9tWyYDgF_0gYdlQ8aWALIQxXx4
        service_instance_id = "<resource-instance-id>"; // example: crn:v1:bluemix:public:cloud-object-
storage:global:a/xx999cd94a0dda86fd8eff3191349999:9999b05b-x999-4917-xxxx-9d5b326a1111::
        endpoint_url = "<endpoint>"; // example: https://s3.us-south.cloud-object-storage.appdomain.cloud
        location = "<storage-class>"; // example: us-south-standard
        // Create client connection details
        _cosClient = createClient(api_key, service_instance_id, endpoint_url, location);

        // Setting string values
        String bucketName = "java.bucket" + UUID.randomUUID().toString().replace("-","");
        String itemName = UUID.randomUUID().toString().replace("-","") + "_java_file.txt";
        String fileText = "This is a test file from the Java code sample!!!";

        // create a new bucket
        createBucket(bucketName, _cosClient);

        // get the list of buckets
        listBuckets(_cosClient);

        // create a new text file & upload
        createTextFile(bucketName, itemName, fileText);

        // get the list of files from the new bucket
        listObjects(bucketName, _cosClient);

        // remove new file
        deleteItem(bucketName, itemName);

        // create & upload the large file using transfer manager & remove large file
        createLargeFile(bucketName);

        // remove the new bucket
        deleteBucket(bucketName);
    }

    private static void createLargeFile(String bucketName)  throws IOException {
        String fileName = "Sample"; //Setting the File Name

        try {
            File uploadFile = File.createTempFile(fileName,".tmp");
            uploadFile.deleteOnExit();
            fileName = uploadFile.getName();

            largeObjectUpload(bucketName, uploadFile);
        } catch (InterruptedException e) {
            System.out.println("object upload timed out");
        }

        deleteItem(bucketName, fileName); // remove new large file
    }

    // Create client connection
    public static AmazonS3 createClient(String api_key, String service_instance_id, String endpoint_url, String location)
    {
        AWSCredentials credentials;
        credentials = new BasicIBMOAuthCredentials(api_key, service_instance_id);

        ClientConfiguration clientConfig = new ClientConfiguration().withRequestTimeout(5000);
        clientConfig.setUseTcpKeepAlive(true);

        AmazonS3 cosClient = AmazonS3ClientBuilder.standard().withCredentials(new AWSStaticCredentialsProvider(credentials))
                .withEndpointConfiguration(new AwsClientBuilder.EndpointConfiguration(endpoint_url,
location)).withPathStyleAccessEnabled(true)
                .withClientConfiguration(clientConfig).build();
        return cosClient;
    }

    // Create a new bucket
    public static void createBucket(String bucketName, AmazonS3 cosClient)
    {
        cosClient.createBucket(bucketName);
        System.out.printf("Bucket: %s created!\n", bucketName);
    }

    // Retrieve the list of available buckets
    public static void listBuckets(AmazonS3 cosClient)
    {
        System.out.println("Listing buckets:");
        final List<Bucket> bucketList = _cosClient.listBuckets();
        for (final Bucket bucket : bucketList) {
            System.out.println(bucket.getName());
        }
        System.out.println();
    }

    // Retrieve the list of contents for a bucket
    public static void listObjects(String bucketName, AmazonS3 cosClient)
    {
        System.out.println("Listing objects in bucket " + bucketName);
        ObjectListing objectListing = cosClient.listObjects(new ListObjectsRequest().withBucketName(bucketName));
        for (S3ObjectSummary objectSummary : objectListing.getObjectSummaries()) {
            System.out.println(" - " + objectSummary.getKey() + "  " + "(size = " + objectSummary.getSize() + ")");
        }
        System.out.println();
    }

    // Create file and upload to new bucket
    public static void createTextFile(String bucketName, String itemName, String fileText) {
        System.out.printf("Creating new item: %s\n", itemName);

        InputStream newStream = new ByteArrayInputStream(fileText.getBytes(Charset.forName("UTF-8")));

        ObjectMetadata metadata = new ObjectMetadata();
        metadata.setContentLength(fileText.length());

        PutObjectRequest req = new PutObjectRequest(bucketName, itemName, newStream, metadata);
        _cosClient.putObject(req);

        System.out.printf("Item: %s created!\n", itemName);
    }

    // Delete item
    public static void deleteItem(String bucketName, String itemName) {
        System.out.printf("Deleting item: %s\n", itemName);
        _cosClient.deleteObject(bucketName, itemName);
        System.out.printf("Item: %s deleted!\n", itemName);
    }

    // Delete bucket
    public static void deleteBucket(String bucketName) {
        System.out.printf("Deleting bucket: %s\n", bucketName);
        _cosClient.deleteBucket(bucketName);
        System.out.printf("Bucket: %s deleted!\n", bucketName);
    }

    //  Upload large file to new bucket
    public static void largeObjectUpload(String bucketName, File uploadFile) throws IOException, InterruptedException {

        if (!uploadFile.isFile()) {
            System.out.printf("The file does not exist or is not accessible.\n");
            return;
        }

        System.out.println("Starting large file upload with TransferManager");

        //set the part size to 5 MB
        long partSize = 1024 * 1024 * 20;

        //set the threshold size to 5 MB
        long thresholdSize = 1024 * 1024 * 20;

        AmazonS3 s3client = createClient( api_key, service_instance_id, endpoint_url, location);

        TransferManager transferManager = TransferManagerBuilder.standard()
                .withS3Client(s3client)
                .withMinimumUploadPartSize(partSize)
                .withMultipartCopyThreshold(thresholdSize)
                .build();

        try {
            Upload lrgUpload = transferManager.upload(bucketName, uploadFile.getName(), uploadFile);
            lrgUpload.waitForCompletion();
            System.out.println("Large file upload complete!");
        } catch (SdkClientException e) {
            System.out.printf("Upload error: %s\n", e.getMessage());
        } finally {
            transferManager.shutdownNow();
        }
    }
}
```

Go

```
package main
import (
    "bytes"
    "fmt"
    "github.com/IBM/ibm-cos-sdk-go/aws"
    "github.com/IBM/ibm-cos-sdk-go/aws/credentials/ibmiam"
    "github.com/IBM/ibm-cos-sdk-go/aws/session"
    "github.com/IBM/ibm-cos-sdk-go/service/s3"
    "io"
    "math/rand"
    "os"
    "time"
)

// Constants for IBM COS values
const (
    apiKey            = "<api-key>" // example: xxxd12V2QHXbjaM99G9tWyYDgF_0gYdlQ8aWALIQxXx4
    serviceInstanceID = "<resource-instance-id>" // example: crn:v1:bluemix:public:cloud-object-
storage:global:a/xx999cd94a0dda86fd8eff3191349999:9999b05b-x999-4917-xxxx-9d5b326a1111::
    authEndpoint      = "https://iam.cloud.ibm.com/identity/token"
    serviceEndpoint   = "<endpoint>" // example: https://s3.us-south.cloud-object-storage.appdomain.cloud
)

// UUID
func  random(min int, max int)  int {
    return rand.Intn(max-min) + min
}

func  main() {

    // UUID
    rand.Seed(time.Now().UnixNano())
    UUID := random(10, 2000)

    // Variables
    newBucket := fmt.Sprintf("%s%d", "go.bucket", UUID) // New bucket name
    objectKey := fmt.Sprintf("%s%d%s", "go_file_", UUID, ".txt") // Object Key
    content := bytes.NewReader([]byte("This is a test file from Go code sample!!!"))
    downloadObjectKey := fmt.Sprintf("%s%d%s", "downloaded_go_file_", UUID, ".txt") // Downloaded Object Key

    //Setting up a new configuration
    conf := aws.NewConfig().
        WithRegion("<storage-class>"). // Enter your storage class (LocationConstraint) - example: us-standard
        WithEndpoint(serviceEndpoint).
        WithCredentials(ibmiam.NewStaticCredentials(aws.NewConfig(), authEndpoint, apiKey, serviceInstanceID)).
        WithS3ForcePathStyle(true)

    // Create client connection
    sess := session.Must(session.NewSession()) // Creating a new session
    client := s3.New(sess, conf)               // Creating a new client

    // Create new bucket
```

```
   _, err := client.CreateBucket(&s3.CreateBucketInput{
       Bucket: aws.String(newBucket), // New Bucket Name
   })
   if err != nil {
       exitErrorf("Unable to create bucket %q, %v", newBucket, err)
   }

   // Wait until bucket is created before finishing
   fmt.Printf("Waiting for bucket %q to be created...\n", newBucket)

   err = client.WaitUntilBucketExists(&s3.HeadBucketInput{
       Bucket: aws.String(newBucket),
   })
   if err != nil {
       exitErrorf("Error occurred while waiting for bucket to be created, %v", newBucket)
   }

   fmt.Printf("Bucket %q successfully created\n", newBucket)

   // Retrieve the list of available buckets
   bklist, err := client.ListBuckets(nil)
   if err != nil {
       exitErrorf("Unable to list buckets, %v", err)
   }

   fmt.Println("Buckets:")

   for _, b := range bklist.Buckets {
       fmt.Printf("* %s created on %s\n",
           aws.StringValue(b.Name), aws.TimeValue(b.CreationDate))
   }

// Uploading an object
input3 := s3.CreateMultipartUploadInput{
 Bucket: aws.String(newBucket), // Bucket Name
 Key:    aws.String(objectKey), // Object Key
}

upload, _ := client.CreateMultipartUpload(&input3)

uploadPartInput := s3.UploadPartInput{
 Bucket:     aws.String(newBucket), // Bucket Name
 Key:        aws.String(objectKey), // Object Key
 PartNumber: aws.Int64(int64(1)),
 UploadId:   upload.UploadId,
 Body:       content,
}

var completedParts []*s3.CompletedPart
completedPart, _ := client.UploadPart(&uploadPartInput)

completedParts = append(completedParts, &s3.CompletedPart{
 ETag:       completedPart.ETag,
 PartNumber: aws.Int64(int64(1)),
})

completeMPUInput := s3.CompleteMultipartUploadInput{
 Bucket: aws.String(newBucket), // Bucket Name
 Key:    aws.String(objectKey), // Object Key
 MultipartUpload: &s3.CompletedMultipartUpload{
  Parts: completedParts,
 },
 UploadId: upload.UploadId,
}

d, _ := client.CompleteMultipartUpload(&completeMPUInput)
fmt.Println(d)

// List objects within a bucket
resp, err := client.ListObjects(&s3.ListObjectsInput{Bucket: aws.String(newBucket)})
if err != nil {
  exitErrorf("Unable to list items in bucket %q, %v", newBucket, err)
 }
 for _, item := range resp.Contents {
  fmt.Println("Name:         ", *item.Key)          // Print the object's name
  fmt.Println("Last modified:", *item.LastModified) // Print the last modified date of the object
  fmt.Println("Size:         ", *item.Size)         // Print the size of the object
  fmt.Println("")
 }

 fmt.Println("Found", len(resp.Contents), "items in bucket", newBucket)


 // Download an object
 input4 := s3.GetObjectInput{
  Bucket: aws.String(newBucket), // The bucket where the object is located
  Key:    aws.String(objectKey), // Object you want to download
 }

 res, err := client.GetObject(&input4)
 if err != nil {
  exitErrorf("Unable to download object %q from bucket %q, %v", objectKey, newBucket, err)
 }

 f, _ := os.Create(downloadObjectKey)
 defer f.Close()
 io.Copy(f, res.Body)

 fmt.Println("Downloaded", f.Name())


 // Delete object within the new bucket
 _, err = client.DeleteObject(&s3.DeleteObjectInput{Bucket: aws.String(newBucket), Key: aws.String(objectKey)})
 if err != nil {
  exitErrorf("Unable to delete object %q from bucket %q, %v", objectKey, newBucket, err)
 }

 err = client.WaitUntilObjectNotExists(&s3.HeadObjectInput{
  Bucket: aws.String(newBucket),
  Key:    aws.String(objectKey),
 })
 if err != nil {
  exitErrorf("Error occurred while waiting for object %q to be deleted, %v", objectKey)
 }

 fmt.Printf("Object %q successfully deleted\n", objectKey)

 // Delete the new bucket
 // It must be empty or else the call fails
 _, err = client.DeleteBucket(&s3.DeleteBucketInput{
  Bucket: aws.String(newBucket),
 })
 if err != nil {
  exitErrorf("Unable to delete bucket %q, %v", newBucket, err)
 }

 // Wait until bucket is deleted before finishing
 fmt.Printf("Waiting for bucket %q to be deleted...\n", newBucket)

 err = client.WaitUntilBucketNotExists(&s3.HeadBucketInput{
  Bucket: aws.String(newBucket),
 })
 if err != nil {
  exitErrorf("Error occurred while waiting for bucket to be deleted, %v", newBucket)
 }

 fmt.Printf("Bucket %q successfully deleted\n", newBucket)
}


func exitErrorf(msg string, args ...interface{}) {
 fmt.Fprintf(os.Stderr, msg+"\n", args...)
 os.Exit(1)
```

Running the Code Example

To run the code sample, copy the code blocks above and run the following:

Python

```
python python-example.py
```

Node

```
node node-example.js
```

Java

```
java javaexamplecode
```

Go

```
go run go_example.go
```

Output from the Code Example

The output from the Code Example should resemble the following:

Python

```
Creating new bucket: py.bucket.779177bfe41945edb458294d0b25440a
Bucket: py.bucket.779177bfe41945edb458294d0b25440a created!
DONE!

Retrieving list of buckets
Bucket Name: py.bucket.779177bfe41945edb458294d0b25440a
DONE!

Creating new item: py_file_17b79068b7c845658f2f74249e14e267.txt in bucket: py.bucket.779177bfe41945edb458294d0b25440a
Item: py_file_17b79068b7c845658f2f74249e14e267.txt created!
DONE!

Retrieving bucket contents from: py.bucket.779177bfe41945edb458294d0b25440a
Item: py_file_17b79068b7c845658f2f74249e14e267.txt (46 bytes).
DONE!

Retrieving item from bucket: py.bucket.779177bfe41945edb458294d0b25440a, key: py_file_17b79068b7c845658f2f74249e14e267.txt
File Contents: This is a test file from Python code sample!!!
DONE!

Starting large file upload for py_large_file_722319147bba4fc4a6c111cc21eb11b5.bin to bucket:
py.bucket.779177bfe41945edb458294d0b25440a
Large file upload complete!
Retrieving bucket contents from: py.bucket.779177bfe41945edb458294d0b25440a
Item: py_file_17b79068b7c845658f2f74249e14e267.txt (46 bytes).
Item: py_large_file_722319147bba4fc4a6c111cc21eb11b5.bin (20971520 bytes).
DONE!

Deleting item: py_large_file_722319147bba4fc4a6c111cc21eb11b5.bin from bucket: py.bucket.779177bfe41945edb458294d0b25440a
Item: py_large_file_722319147bba4fc4a6c111cc21eb11b5.bin deleted!
DONE!

Deleting item: py_file_17b79068b7c845658f2f74249e14e267.txt from bucket: py.bucket.779177bfe41945edb458294d0b25440a
Item: py_file_17b79068b7c845658f2f74249e14e267.txt deleted!
DONE!

Deleting bucket: py.bucket.779177bfe41945edb458294d0b25440a
Bucket: py.bucket.779177bfe41945edb458294d0b25440a deleted!
DONE!
```

Node

```
Creating new bucket: js.bucket.c697b4403f8211e9b1228597cf8e3a32
Bucket: js.bucket.c697b4403f8211e9b1228597cf8e3a32 created!
DONE!

Retrieving list of buckets
Bucket Name: js.bucket.c697b4403f8211e9b1228597cf8e3a32
DONE!

Creating new item: js_file_c697db503f8211e9b1228597cf8e3a32.txt
Item: js_file_c697db503f8211e9b1228597cf8e3a32.txt created!
DONE!

Retrieving bucket contents from: js.bucket.c697b4403f8211e9b1228597cf8e3a32
Item: js_file_c697db503f8211e9b1228597cf8e3a32.txt (47 bytes).
DONE!

Retrieving item from bucket: js.bucket.c697b4403f8211e9b1228597cf8e3a32, key: js_file_c697db503f8211e9b1228597cf8e3a32.txt
File Contents: This is a test file from Node.js code sample!!!
DONE!

Starting multi-part upload for js_large_file_c697db513f8211e9b1228597cf8e3a32.bin to bucket:
js.bucket.c697b4403f8211e9b1228597cf8e3a32
Uploading to js_large_file_c697db513f8211e9b1228597cf8e3a32.bin (part 1 of 4)
Uploading to js_large_file_c697db513f8211e9b1228597cf8e3a32.bin (part 2 of 4)
Uploading to js_large_file_c697db513f8211e9b1228597cf8e3a32.bin (part 3 of 4)
Uploading to js_large_file_c697db513f8211e9b1228597cf8e3a32.bin (part 4 of 4)
DONE!

Retrieving bucket contents from: js.bucket.c697b4403f8211e9b1228597cf8e3a32
Item: js_file_c697db503f8211e9b1228597cf8e3a32.txt (47 bytes).
Item: js_large_file_c697db513f8211e9b1228597cf8e3a32.bin (20971520 bytes).
DONE!

Deleting item: js_large_file_c697db513f8211e9b1228597cf8e3a32.bin
Item: js_large_file_c697db513f8211e9b1228597cf8e3a32.bin deleted!
DONE!

Deleting item: js_file_c697db503f8211e9b1228597cf8e3a32.txt
Item: js_file_c697db503f8211e9b1228597cf8e3a32.txt deleted!
DONE!

Deleting bucket: js.bucket.c697b4403f8211e9b1228597cf8e3a32
Bucket: js.bucket.c697b4403f8211e9b1228597cf8e3a32 deleted!
DONE!
```

Java

```
Bucket: java.bucket71bd68d087b948f5a1f1cbdd86e4fda2 created!
DONE!

Listing buckets:
java.bucket71bd68d087b948f5a1f1cbdd86e4fda2

Creating new item: 4e69e627be7e4e10bf8d39e3fa10058f_java_file.txt
Item: 4e69e627be7e4e10bf8d39e3fa10058f_java_file.txt created!

Listing objects in bucket java.bucket71bd68d087b948f5a1f1cbdd86e4fda2
 - 4e69e627be7e4e10bf8d39e3fa10058f_java_file.txt  (size = 48)

Deleting item: 4e69e627be7e4e10bf8d39e3fa10058f_java_file.txt
Item: 4e69e627be7e4e10bf8d39e3fa10058f_java_file.txt deleted!

Starting large file upload with TransferManager
Large file upload complete!
Deleting item: Sample5438677733541671254.tmp
Item: Sample5438677733541671254.tmp deleted!

Deleting bucket: java.bucket71bd68d087b948f5a1f1cbdd86e4fda2
```

```
Bucket: java.bucket71bd68d087b948f5a1f1cbdd86e4fda2 deleted!
```

Go

```
Waiting for bucket "go.bucket645" to be created...
Bucket "go.bucket645" successfully created

Listing buckets:
* go.bucket645 created on 2019-03-10 13:25:12.072 +0000 UTC

{
  Bucket: "go.bucket645",
  ETag: "\"686d1d07d6de02e920532342fcbd6d2a-1\"",
  Key: "go_file_645.txt",
  Location: "http://s3.us.cloud-object-storage.appdomain.cloud/go.bucket645/go_file_645.txt"
}

Name:          go_file_645.txt
Last modified: 2019-03-10 13:25:14 +0000 UTC
Size:          42

Found 1 items in bucket go.bucket645

Downloaded downloaded_go_file_645.txt

Object "go_file_645.txt" successfully deleted

Waiting for bucket "go.bucket645" to be deleted...
Bucket "go.bucket645" successfully deleted
```

Using Java

The IBM Cloud® Object Storage SDK for Java provides features to make the most of IBM Cloud Object Storage.

The IBM Cloud Object Storage SDK for Java is comprehensive, with many features and capabilities that exceed the scope and space of this guide. For detailed class and method documentation see the Javadoc. Source code can be found in the GitHub repository.

Getting the SDK

The easiest way to use the IBM Cloud Object Storage Java SDK is to use Maven to manage dependencies. If you aren't familiar with Maven, you can get up and running by using the Maven in 5-Minutes guide.

Maven uses a file that is called `pom.xml` to specify the libraries (and their versions) needed for a Java project. Here is an example `pom.xml` file for using the IBM Cloud Object Storage Java SDK to connect to Object Storage.

```
<project  xmlns="http://maven.apache.org/POM/4.0.0"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.cos</groupId>
    <artifactId>docs</artifactId>
    <version>1.0.0-SNAPSHOT</version>
    <dependencies>
        <dependency>
            <groupId>com.ibm.cos</groupId>
            <artifactId>ibm-cos-java-sdk</artifactId>
            <version>2.8.0</version>
        </dependency>
    </dependencies>
</project>
```

Creating a client and sourcing credentials

In the following example, a client `cos` is created and configured by providing credential information (API key and service instance ID). These values can also be automatically sourced from a credentials file or from environment variables.

After generating a Service Credential, the resulting JSON document can be saved to `~/.bluemix/cos_credentials`. The SDK will automatically source credentials from this file unless other credentials are explicitly set during client creation. If the `cos_credentials` file contains HMAC keys the client authenticates with a signature, otherwise the client uses the provided API key to authenticate by using a bearer token.

If migrating from AWS S3, you can also source credentials data from `~/.aws/credentials` in the format:

```
$ [default]
aws_access_key_id = {API_KEY}
aws_secret_access_key = {SERVICE_INSTANCE_ID}
```

If both `~/.bluemix/cos_credentials` and `~/.aws/credentials` exist, `cos_credentials` takes preference.

For more details on client construction, see the Javadoc.

Code Examples

Note: In your code, you must remove the angled brackets or any other excess characters that are provided here as illustration.

Let's start with an complete example class that will run through some basic functionality, then explore the classes individually. This `CosExample` class will list objects in an existing bucket, create a new bucket, and then list all buckets in the service instance.

Gather required information

- `bucketName` and `newBucketName` are unique and DNS-safe strings. Because bucket names are unique across the entire system, these values need to be changed if this example is run multiple times. Note that names are reserved for 10 - 15 minutes after deletion.
- `apiKey` is the value found in the Service Credential as `apikey`.
- `serviceInstanceId` is the value found in the Service Credential as `resource_instance_id`.
- `endpointUrl` is a service endpoint URL, inclusive of the `https://` protocol. This is **not** the `endpoints` value found in the Service Credential. For more information about endpoints, see Endpoints and storage locations.
- `storageClass` is a valid provisioning code that corresponds to the `endpoint` value. This is then used as the S3 API `LocationConstraint` variable.
- `location` should be set to the location portion of the `storageClass`. For `us-south-standard`, this would be `us-south`. This variable is used only for the calculation of HMAC signatures, but is required for any client, including this example that uses an IAM API key.

```
    package com.cos;

    import java.time.LocalDateTime;
    import java.util.List;

    import com.ibm.cloud.objectstorage.ClientConfiguration;
    import com.ibm.cloud.objectstorage.auth.AWSCredentials;
    import com.ibm.cloud.objectstorage.auth.AWSStaticCredentialsProvider;
    import com.ibm.cloud.objectstorage.client.builder.AwsClientBuilder.EndpointConfiguration;
    import com.ibm.cloud.objectstorage.services.s3.AmazonS3;
    import com.ibm.cloud.objectstorage.services.s3.AmazonS3ClientBuilder;
    import com.ibm.cloud.objectstorage.services.s3.model.Bucket;
    import com.ibm.cloud.objectstorage.services.s3.model.ListObjectsRequest;
    import com.ibm.cloud.objectstorage.services.s3.model.ObjectListing;
    import com.ibm.cloud.objectstorage.services.s3.model.S3ObjectSummary;
    import com.ibm.cloud.objectstorage.oauth.BasicIBMOAuthCredentials;

    public class CosExample
    {
        public static void main(String[] args)
        {
            String bucketName = "<BUCKET_NAME>";  // eg my-unique-bucket-name
            String newBucketName = "<NEW_BUCKET_NAME>"; // eg my-other-unique-bucket-name
            String apiKey = "<API_KEY>"; // eg "W00YiRnLW4k3fTjMB-oiB-2ySfTrFBIQQWanc--P3byk"
            String serviceInstanceId = "<SERVICE_INSTANCE_ID"; // eg "crn:v1:bluemix:public:cloud-object-
storage:global:a/3bf0d9003abfb5d29761c3e97696b71c:d6f04d83-6c4f-4a62-a165-696756d63903::"
            String endpointUrl = "https://s3.us-south.cloud-object-storage.appdomain.cloud"; // this could be any service
endpoint

            String storageClass = "us-south-standard";
            String location = "us"; // not an endpoint, but used in a custom function below to obtain the correct URL

            System.out.println("Current time: " + LocalDateTime.now());
            AmazonS3 cosClient = createClient(apiKey, serviceInstanceId, endpointUrl, location);
            listObjects(cosClient, bucketName);
            createBucket(cosClient, newBucketName, storageClass);
            listBuckets(cosClient);
        }

        public static AmazonS3 createClient(String apiKey, String serviceInstanceId, String endpointUrl, String location)
        {
            AWSCredentials credentials = new BasicIBMOAuthCredentials(apiKey, serviceInstanceId);
            ClientConfiguration clientConfig = new ClientConfiguration()
                    .withRequestTimeout(5000)
                    .withTcpKeepAlive(true);

            return AmazonS3ClientBuilder
                    .standard()
                    .withCredentials(new AWSStaticCredentialsProvider(credentials))
                    .withEndpointConfiguration(new EndpointConfiguration(endpointUrl, location))
                    .withPathStyleAccessEnabled(true)
                    .withClientConfiguration(clientConfig)
                    .build();
        }

        public static void listObjects(AmazonS3 cosClient, String bucketName)
        {
            System.out.println("Listing objects in bucket " + bucketName);
            ObjectListing objectListing = cosClient.listObjects(new ListObjectsRequest().withBucketName(bucketName));
            for (S3ObjectSummary objectSummary : objectListing.getObjectSummaries()) {
                System.out.println(" - " + objectSummary.getKey() + "  " + "(size = " + objectSummary.getSize() + ")");
            }
            System.out.println();
        }

        public static void createBucket(AmazonS3 cosClient, String bucketName, String storageClass)
        {
            cosClient.createBucket(bucketName, storageClass);
        }

        public static void listBuckets(AmazonS3 cosClient)
        {
            System.out.println("Listing buckets");
            final List<Bucket> bucketList = cosClient.listBuckets();
            for (final Bucket bucket : bucketList) {
                System.out.println(bucket.getName());
            }
            System.out.println();
        }
    }
```

Initializing configuration

```
$ private static String COS_ENDPOINT = "<endpoint>"; // eg "https://s3.us.cloud-object-storage.appdomain.cloud"
private static String COS_API_KEY_ID = "<api-key>"; // eg "0viPHOY7LbLNa9eLftrtHPpTjoGv6hbLD1QalRXikliJ"
private static String COS_AUTH_ENDPOINT = "https://iam.cloud.ibm.com/identity/token";
private static String COS_SERVICE_CRN = "<resource-instance-id>"; // "crn:v1:bluemix:public:cloud-object-
storage:global:a/<CREDENTIAL_ID_AS_GENERATED>:<SERVICE_ID_AS_GENERATED>::"
private static String COS_BUCKET_LOCATION = "<location>"; // eg "us"

public static void main(String[] args)
{
    SDKGlobalConfiguration.IAM_ENDPOINT = COS_AUTH_ENDPOINT;

    try {
        _cos = createClient(COS_API_KEY_ID, COS_SERVICE_CRN, COS_ENDPOINT, COS_BUCKET_LOCATION);
    } catch (SdkClientException sdke) {
        System.out.printf("SDK Error: %s\n", sdke.getMessage());
    } catch (Exception e) {
        System.out.printf("Error: %s\n", e.getMessage());
    }
}
```

```
public static AmazonS3 createClient(String api_key, String service_instance_id, String endpoint_url, String location)
{
    AWSCredentials credentials = new BasicIBMOAuthCredentials(api_key, service_instance_id);
    ClientConfiguration clientConfig = new ClientConfiguration().withRequestTimeout(5000);
    clientConfig.setUseTcpKeepAlive(true);

    AmazonS3 cos = AmazonS3ClientBuilder.standard().withCredentials(new AWSStaticCredentialsProvider(credentials))
            .withEndpointConfiguration(new EndpointConfiguration(endpoint_url, location)).withPathStyleAccessEnabled(true)
            .withClientConfiguration(clientConfig).build();

    return cos;
}
```

Key Values

- `<endpoint>` - public endpoint for your cloud Object Storage (available from the IBM Cloud Dashboard). For more information about endpoints, see Endpoints and storage locations.
- `<api-key>` - api key generated when creating the service credentials (write access is required for creation and deletion examples)
- `<resource-instance-id>` - resource ID for your cloud Object Storage (available through IBM Cloud CLI or IBM Cloud Dashboard)
- `<location>` - default location for your cloud Object Storage (must match the region that is used for `<endpoint>`)

SDK References

Classes

- AmazonS3ClientBuilder
- AWSCredentials
- AWSStaticCredentialsProvider
- BasicAWSCredentials
- BasicIBMOAuthCredentials
- ClientConfiguration
- EndpointConfiguration
- SdkClientException

Determining Endpoint

The methods below can be used to determine the service endpoint based on the bucket location, endpoint type (public or private), and specific region (optional). For more information about endpoints, see Endpoints and storage locations.

```
$ /**
* Returns a service endpoint based on the
* storage class location (i.e. us-standard, us-south-standard),
* endpoint type (public or private)
*/
public static String getEndpoint(String location, String endPointType) {
    return getEndpoint(location, "", endPointType);
}

/**
* Returns a service endpoint based on the
* storage class location (i.e. us-standard, us-south-standard),
* specific region if desired (i.e. sanjose, amsterdam) - only use if you want a specific regional endpoint,
* endpoint type (public or private)
*/
public static String getEndpoint(String location, String region, String endpointType) {
    HashMap locationMap = new HashMap<String, String>();
    locationMap.put("us", "s3-api.us-geo");
    locationMap.put("us-dallas", "s3-api.dal-us-geo");
    locationMap.put("us-sanjose", "s3-api.sjc-us-geo");
    locationMap.put("us-washington", "s3-api.wdc-us-geo");
    locationMap.put("us-south", "s3.us-south");
    locationMap.put("us-east", "s3.us-east");
    locationMap.put("eu", "s3.eu-geo");
    locationMap.put("eu-amsterdam", "s3.ams-eu-geo");
    locationMap.put("eu-frankfurt", "s3.fra-eu-geo");
    locationMap.put("eu-milan", "s3.mil-eu-geo");
    locationMap.put("eu-gb", "s3.eu-gb");
    locationMap.put("eu-germany", "s3.eu-de");
    locationMap.put("ap", "s3.ap-geo");
    locationMap.put("ap-tokyo", "s3.tok-ap-geo");
    locationMap.put("che01", "s3.che01");
    locationMap.put("mel01", "s3.mel01");
    locationMap.put("tor01", "s3.tor01");

    String key = location.substring(0, location.lastIndexOf("-")) + (region != null && !region.isEmpty() ? "-" + region : "");
    String endpoint = locationMap.getOrDefault(key, null).toString();

    if (endpoint != null) {
        if (endpointType.toLowerCase() == "private")
            endpoint += ".objectstorage.service.networklayer.com";
        else
            endpoint += ".objectstorage.s3.us-south.cloud-object-storage.appdomain.cloud.net";
    }

    return endpoint;
}
```

Creating a new bucket

```
$ public static void createBucket(String bucketName) {
    System.out.printf("Creating new bucket: %s\n", bucketName);
    _cos.createBucket(bucketName);
    System.out.printf("Bucket: %s created!\n", bucketName);
}
```

Create a bucket with a different storage class

A list of valid provisioning codes for `LocationConstraint` can be referenced in [the Storage Classes guide].

```
$ cos.createBucket("sample", "us-vault"); // the name of the bucket, and the storage class (LocationConstraint)
```

SDK References

- `createBucket`

Creating a new text file

```
$ public static void createTextFile(String bucketName, String itemName, String fileText) {
    System.out.printf("Creating new item: %s\n", itemName);

    byte[] arr = fileText.getBytes(StandardCharsets.UTF_8);
    InputStream newStream = new ByteArrayInputStream(arr);

    ObjectMetadata metadata = new ObjectMetadata();
    metadata.setContentLength(arr.length);

    PutObjectRequest req = new PutObjectRequest(bucketName, itemName, newStream, metadata);
    _cos.putObject(req);

    System.out.printf("Item: %s created!\n", itemName);
}
```

Tip: Note that when adding custom metadata to an object, it is necessary to create an `ObjectMetadata` object by using the SDK, and not to manually send a custom header containing `x-amz-meta-{key}`. The latter can cause issues when authenticating by using HMAC credentials.

Upload object from a file

This example assumes that the bucket `sample` exists.

```
$ cos.putObject(
    "sample", // the name of the destination bucket
    "myfile", // the object key
    new File("/home/user/test.txt") // the file name and path of the object to be uploaded
);
```

Upload object by using a stream

This example assumes that the bucket `sample` exists.

```
$ String obj = "An example"; // the object to be stored
ByteArrayOutputStream theBytes = new ByteArrayOutputStream(); // create a new output stream to store the object data
ObjectOutputStream serializer = new ObjectOutputStream(theBytes); // set the object data to be serialized
serializer.writeObject(obj); // serialize the object data
serializer.flush();
serializer.close();
InputStream stream = new ByteArrayInputStream(theBytes.toByteArray()); // convert the serialized data to a new input stream to
store
ObjectMetadata metadata = new ObjectMetadata(); // define the metadata
metadata.setContentType("application/x-java-serialized-object"); // set the metadata
metadata.setContentLength(theBytes.size()); // set metadata for the length of the data stream
cos.putObject(
    "sample", // the name of the bucket to which the object is being written
    "serialized-object", // the name of the object being written
    stream, // the name of the data stream writing the object
    metadata // the metadata for the object being written
);
```

Alternatively, you can use a `CipherInputStream` to more easily encrypt the data stream without needing to overload the existing `InputStream` object.

```
$ public CipherInputStream encryptStream(InputStream inputStream) {
       // Generate key
       KeyGenerator kgen = KeyGenerator.getInstance("AES");
       kgen.init(128);
       SecretKey aesKey = kgen.generateKey();
       // Encrypt cipher
       Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
       cipher.init(Cipher.ENCRYPT_MODE, aesKey);
       CipherInputStream cis = new CipherInputStream(inputStream, cipher);
       return cis;
}
```

Download object to a file

This example assumes that the bucket `sample` exists.

```
$ GetObjectRequest request = new // create a new request to get an object
GetObjectRequest( // request the new object by identifying
    "sample", // the name of the bucket
    "myFile" // the name of the object
);

s3Client.getObject( // write the contents of the object
    request, // using the request that was just created
    new File("retrieved.txt") // to write to a new file
);
```

Download object by using a stream

This example assumes that the bucket `sample` exists.

```
$ S3Object returned = cos.getObject( // request the object by identifying
    "sample", // the name of the bucket
    "serialized-object" // the name of the serialized object
);
S3ObjectInputStream s3Input = returned.getObjectContent(); // set the object stream
```

Copy objects

```
$ // copy an object within the same Bucket
cos.copyObject( // copy the Object, passing…
    "sample",  // the name of the Bucket in which the Object to be copied is stored,
    "myFile.txt",  // the name of the Object being copied from the source Bucket,
    "sample",  // the name of the Bucket in which the Object to be copied is stored,
    "myFile.txt.backup"    // and the new name of the copy of the Object to be copied
);
```

```
$ // copy an object between two Buckets
cos.copyObject( // copy the Object, passing…
    "sample", // the name of the Bucket from which the Object will be copied,
    "myFile.txt", // the name of the Object being copied from the source Bucket,
    "backup", // the name of the Bucket to which the Object will be copied,
    "myFile.txt" // and the name of the copied Object in the destination Bucket
);
```

SDK References

Classes

- ObjectMetadata
- PutObjectRequest

*Methods

- `putObject`

`putObject` Exception

The `putObject` method might throw the following exception even if the new object upload was successful:

```
$ Exception in thread "main" java.lang.NoClassDefFoundError: javax/xml/bind/JAXBException
    at com.ibm.cloud.objectstorage.services.s3.AmazonS3Client.putObject(AmazonS3Client.java:1597)
    at ibmcos.CoSExample.createTextFile(CoSExample.java:174)
    at ibmcos.CoSExample.main(CoSExample.java:65)
Caused by: java.lang.ClassNotFoundException: javax.xml.bind.JAXBException
    at java.base/jdk.internal.loader.BuiltinClassLoader.loadClass(BuiltinClassLoader.java:582)
    at java.base/jdk.internal.loader.ClassLoaders$AppClassLoader.loadClass(ClassLoaders.java:190)
    at java.base/java.lang.ClassLoader.loadClass(ClassLoader.java:499)
    ... 3 more
```

Root Cause: The JAXB APIs are considered to be Java EE APIs, and are no longer contained on the default class path in Java SE 9.

Fix: Add the following entry to the pom.xml file in your project folder and repackage your project

```
$ <dependency>
    <groupId>javax.xml.bind</groupId>
    <artifactId>jaxb-api</artifactId>
    <version>2.3.0</version>
</dependency>
```

List available buckets

```
$ public static void getBuckets() {
    System.out.println("Retrieving list of buckets");

    final List<Bucket> bucketList = _cos.listBuckets();
    for (final Bucket bucket : bucketList) {
        System.out.printf("Bucket Name: %s\n", bucket.getName());
    }
}
```

SDK References

Classes

- Bucket

Methods

- `listBuckets`

List items in a bucket (v2)

The AmazonS3 object contains an updated method to list the contents (listObjectsV2). This method allows you to limit the number of records that are returned and retrieve the records in batches. This might be useful for paging your results within an application and improve performance.

```
$ public static void getBucketContentsV2(String bucketName, int maxKeys) {
    System.out.printf("Retrieving bucket contents (V2) from: %s\n", bucketName);

    boolean moreResults = true;
    String nextToken = "";

    while (moreResults) {
        ListObjectsV2Request request = new ListObjectsV2Request()
            .withBucketName(bucketName)
            .withMaxKeys(maxKeys)
            .withContinuationToken(nextToken);

        ListObjectsV2Result result = _cos.listObjectsV2(request);
        for(S3ObjectSummary objectSummary : result.getObjectSummaries()) {
            System.out.printf("Item: %s (%s bytes)\n", objectSummary.getKey(), objectSummary.getSize());
        }
        if (result.isTruncated()) {
            nextToken = result.getNextContinuationToken();
            System.out.println("...More results in next batch!\n");
        }
        else {
            nextToken = "";
            moreResults = false;
        }
    }
    System.out.println("...No more results!");
}
```

SDK References

Classes

- ListObjectsV2Request
- ListObjectsV2Result
- S3ObjectSummary

Methods

- getObjectSummaries
- getNextContinuationToken
- listObjectsV2

Get file contents of particular item

```
$ public static void getItem(String bucketName, String itemName) {
    System.out.printf("Retrieving item from bucket: %s, key: %s\n", bucketName, itemName);

    S3Object item = _cos.getObject(new GetObjectRequest(bucketName, itemName));

    try {
        final int bufferSize = 1024;
        final char[] buffer = new char[bufferSize];
        final StringBuilder out = new StringBuilder();
        InputStreamReader in = new InputStreamReader(item.getObjectContent());

        for (; ; ) {
            int rsz = in.read(buffer, 0, buffer.length);
            if (rsz < 0)
                break;
            out.append(buffer, 0, rsz);
        }
        System.out.println(out.toString());
    } catch (IOException ioe){
        System.out.printf("Error reading file %s: %s\n", name, ioe.getMessage());
    }
}
```

SDK References

Classes

- GetObjectRequest

Methods

- `getObject`

Delete an item from a bucket

```
$ public static void deleteItem(String bucketName, String itemName) {
    System.out.printf("Deleting item: %s\n", itemName);
    _cos.deleteObject(bucketName, itemName);
    System.out.printf("Item: %s deleted!\n", itemName);
}
```

SDK References

Methods

- `deleteObject`

Delete multiple items from a bucket

Tip: The delete request can contain a maximum of 1000 keys that you want to delete. While this is very useful in reducing the per-request performance hit, be mindful when deleting a large number of keys. Also take into account the sizes of the objects to ensure suitable performance.

```
$ public static void deleteItems(String bucketName) {
    DeleteObjectsRequest req = new DeleteObjectsRequest(bucketName);
    req.withKeys(
        "deletetest/testfile1.txt",
        "deletetest/testfile2.txt",
        "deletetest/testfile3.txt",
        "deletetest/testfile4.txt",
        "deletetest/testfile5.txt"
    );

    DeleteObjectsResult res = _cos.deleteObjects(req);

    System.out.printf("Deleted items for %s\n", bucketName);

    List<DeleteObjectsResult.DeletedObject> deletedItems = res.getDeletedObjects();
    for(DeleteObjectsResult.DeletedObject deletedItem : deletedItems) {
        System.out.printf("Deleted item: %s\n", deletedItem.getKey());
    }
}
```

SDK References

Classes

- `DeleteObjectsRequest`
- `DeleteObjectsResult`
- `DeleteObjectsResult.DeletedObject`

Methods

- `deleteObjects`

Delete a bucket

```
$ public static void deleteBucket(String bucketName) {
    System.out.printf("Deleting bucket: %s\n", bucketName);
    _cos.deleteBucket(bucketName);
    System.out.printf("Bucket: %s deleted!\n", bucketName);
}
```

SDK References

Methods

- deleteBucket

Check if an object is publicly readable

```
$ public static void getItemACL(String bucketName, String itemName) {
    System.out.printf("Retrieving ACL for %s from bucket: %s\n", itemName, bucketName);

    AccessControlList acl = _cos.getObjectAcl(bucketName, itemName);

    List<Grant> grants = acl.getGrantsAsList();

    for (Grant grant : grants) {
        System.out.printf("User: %s (%s)\n", grant.getGrantee().getIdentifier(), grant.getPermission().toString());
    }
}
```

SDK References

Classes

- AccessControlList
- Grant

Methods

- getObjectAcl

Execute a multi-part upload

```
$ public static void multiPartUpload(String bucketName, String itemName, String filePath) {
    File file = new File(filePath);
    if (!file.isFile()) {
        System.out.printf("The file '%s' does not exist or is not accessible.\n", filePath);
        return;
    }

    System.out.printf("Starting multi-part upload for %s to bucket: %s\n", itemName, bucketName);

    InitiateMultipartUploadResult mpResult = _cos.initiateMultipartUpload(new InitiateMultipartUploadRequest(bucketName,
itemName));
    String uploadID = mpResult.getUploadId();

    //begin uploading the parts
    //min 5MB part size
    long partSize = 1024 * 1024 * 5;
    long fileSize = file.length();
    long partCount = ((long)Math.ceil(fileSize / partSize)) + 1;
    List<PartETag> dataPacks = new ArrayList<PartETag>();

    try {
        long position = 0;
        for (int partNum = 1; position < fileSize; partNum++) {
            partSize = Math.min(partSize, (fileSize - position));

            System.out.printf("Uploading to %s (part %s of %s)\n", name, partNum, partCount);

            UploadPartRequest upRequest = new UploadPartRequest()
                    .withBucketName(bucketName)
                    .withKey(itemName)
                    .withUploadId(uploadID)
                    .withPartNumber(partNum)
                    .withFileOffset(position)
                    .withFile(file)
                    .withPartSize(partSize);

            UploadPartResult upResult = _cos.uploadPart(upRequest);
            dataPacks.add(upResult.getPartETag());

            position += partSize;
        }

        //complete upload
        _cos.completeMultipartUpload(new CompleteMultipartUploadRequest(bucketName, itemName, uploadID, dataPacks));
        System.out.printf("Upload for %s Complete!\n", itemName);
    } catch (SdkClientException sdke) {
        System.out.printf("Multi-part upload aborted for %s\n", itemName);
        System.out.printf("Upload Error: %s\n", sdke.getMessage());
        _cos.abortMultipartUpload(new AbortMultipartUploadRequest(bucketName, itemName, uploadID));
    }
}
```

SDK References

Classes

- AbortMultipartUploadRequest
- CompleteMultipartUploadRequest
- InitiateMultipartUploadRequest
- InitiateMultipartUploadResult
- SdkClientException
- UploadPartRequest
- UploadPartResult

Methods

- abortMultipartUpload
- completeMultipartUpload
- initiateMultipartUpload
- `uploadPart`

Upload larger objects using a Transfer Manager

The `TransferManager` simplifies large file transfers by automatically incorporating multi-part uploads whenever necessary setting configuration parameters.

```
$ public static void largeObjectUpload(String bucketName, String itemName, String filePath) throws IOException,
InterruptedException {
    File uploadFile = new File(filePath);

    if (!uploadFile.isFile()) {
        System.out.printf("The file '%s' does not exist or is not accessible.\n", filePath);
        return;
    }

    System.out.println("Starting large file upload with TransferManager");

    //set the part size to 5 MB
    long partSize = 1024 * 1024 * 5;

    //set the threshold size to 5 MB
    long thresholdSize = 1024 * 1024 * 5;

    String endPoint = getEndpoint(COS_BUCKET_LOCATION, "public");
    AmazonS3 s3client = createClient(COS_API_KEY_ID, COS_SERVICE_CRN, endPoint, COS_BUCKET_LOCATION);

    TransferManager transferManager = TransferManagerBuilder.standard()
```

```
        .withS3Client(s3client)
        .withMinimumUploadPartSize(partSize)
        .withMultipartCopyThreshold(thresholdSize)
        .build();

    try {
        Upload lrgUpload = transferManager.upload(bucketName, itemName, uploadFile);

        lrgUpload.waitForCompletion();

        System.out.println("Large file upload complete!");
    }
    catch (SdkClientException e) {
        System.out.printf("Upload error: %s\n", e.getMessage());
    }
    finally {
        transferManager.shutdownNow();
    }
```

SDK References

Classes

- TransferManager
- TransferManagerBuilder
- Upload

Methods

- `shutdownNow`
- upload
- waitForCompletion

Using Key Protect

Key Protect can be added to a storage bucket to encrypt sensitive data at rest in the cloud.

Before You Begin

The following items are necessary in order to create a bucket with Key-Protect enabled:

- A Key Protect service provisioned
- A Root key available (either generated or imported)

Retrieving the Root Key CRN

1. Retrieve the instance ID for your Key Protect service
2. Use the Key Protect API to retrieve all your available keys
   - You can either use `curl` commands or an API REST Client such as Postman to access the Key Protect API.
3. Retrieve the CRN of the root key you will use to enabled Key Protect on the your bucket. The CRN will look similar to below:

`crn:v1:bluemix:public:kms:us-south:a/3d624cd74a0dea86ed8efe3101341742:90b6a1db-0fe1-4fe9-b91e-962c327df531:key:0bg3e33e-a866-50f2-b715-5cba2bc93234`

Creating a bucket with key-protect enabled

```
$ private static String COS_KP_ALGORITHM = "<algorithm>";
private static String COS_KP_ROOTKEY_CRN = "<root-key-crn>";

public static void createBucketKP(String bucketName) {
    System.out.printf("Creating new encrypted bucket: %s\n", bucketName);

    EncryptionType encType = new EncryptionType();
    encType.setKmsEncryptionAlgorithm(COS_KP_ALGORITHM);
    encType.setIBMSSEKMSCustomerRootKeyCrn(COS_KP_ROOTKEY_CRN);

    CreateBucketRequest req = new CreateBucketRequest(bucketName).withEncryptionType(encType);
    _cos.createBucket(req);

    System.out.printf("Bucket: %s created!", bucketName);
}
```

Key Values

- `<algorithm>` - The encryption algorithm used for new objects added to the bucket (Default is AES256).
- `<root-key-crn>` - CRN of the Root Key obtained from the Key Protect service.

SDK References

Classes

- `CreateBucketRequest`
- `EncryptionType`

Methods

- `createBucket`

New Headers for Key Protect

The additional headers have been defined within `Headers` class:

```
$ public static final String IBM_SSE_KP_ENCRYPTION_ALGORITHM = "ibm-sse-kp-encryption-algorithm";
public static final String IBM_SSE_KP_CUSTOMER_ROOT_KEY_CRN = "ibm-sse-kp-customer-root-key-crn";
```

The same section of the create bucket implementation which already adds IAM service instance headers will add the 2 new encryption headers:

```
$ //Add IBM Service Instance Id & Encryption to headers
if ((null != this.awsCredentialsProvider ) && (this.awsCredentialsProvider.getCredentials() instanceof IBMOAuthCredentials)) {
    IBMOAuthCredentials oAuthCreds = (IBMOAuthCredentials)this.awsCredentialsProvider.getCredentials();
    if (oAuthCreds.getServiceInstanceId() != null) {
        request.addHeader(Headers.IBM_SERVICE_INSTANCE_ID, oAuthCreds.getServiceInstanceId());
        request.addHeader(Headers.IBM_SSE_KP_ENCRYPTION_ALGORITHM,
createBucketRequest.getEncryptionType().getKpEncryptionAlgorithm());
        request.addHeader(Headers.IBM_SSE_KP_CUSTOMER_ROOT_KEY_CRN,
createBucketRequest.getEncryptionType().getIBMSSEKPCustomerRootKeyCrn());
    }
}
```

The `ObjectListing` and `HeadBucketResult` objects have been updated to include boolean `IBMSSEKPEnabled` & String `IBMSSEKPCustomerRootKeyCrn` variables with getter & setter methods. These will store the values of the new headers.

GET bucket

```
$ public ObjectListing listObjects(String bucketName)
public ObjectListing listObjects(String bucketName, String prefix)
public ObjectListing listObjects(ListObjectsRequest listObjectsRequest)
```

The `ObjectListing` class will require 2 additional methods:

```
$ ObjectListing listing = s3client.listObjects(listObjectsRequest)
String KPEnabled = listing.getIBMSSEKPEnabled();
String crkId = listing.getIBMSSEKPCrkId();
```

The additional headers have been defined within the `Headers` class:

```
$ Headers.java
public static final string IBM_SSE_KP_ENABLED = "ibm-sse-kp-enabled";
public static final String IBM_SSE_KP_CUSTOMER_ROOT_KEY_CRN = "ibm-sse-kp-customer-root-key-crn";
```

The S3XmlResponseHandler which is responsible for unmarshalling all xml responses. A check has been added that the result is an instance of `ObjectListing` and the retrieved headers will be added to the `ObjectListing` object:

```
$ if (result instanceof ObjectListing) {
    if (!StringUtils.isNullOrEmpty(responseHeaders.get(Headers.IBM_SSE_KP_ENABLED)){
            ((ObjectListing) result).setIBMSSEKPEnabled(Boolean.parseBoolean(responseHeaders.get(Headers.IBM_SSE_KP_ENABLED)));
        }
    if (!StringUtils.isNullOrEmpty(responseHeaders.get(Headers.IBM_SSE_KP_CUSTOMER_ROOT_KEY_CRN))) {
            ((ObjectListing) result).setIBMSSEKPCrk(responseHeaders.get(Headers.IBM_SSE_KP_CUSTOMER_ROOT_KEY_CRN));
        }
}
```

HEAD bucket

The additional headers have been defined within Headers class:

```
$ Headers.java
public static final String IBM_SSE_KP_ENABLED = "ibm-sse-kp-enabled";
public static final String IBM_SSE_KP_CUSTOMER_ROOT_KEY_CRN = "ibm-sse-kp-customer-root-key-crn";
```

These variables are populated in the HeadBucketResponseHandler.

```
$ HeadBucketResultHandler
result.setIBMSSEKPEnabled(response.getHeaders().get(Headers.IBM_SSE_KP_ENABLED));
result.setIBMSSEKPCrk(response.getHeaders().get(Headers. IBM_SSE_KP_CUSTOMER_ROOT_KEY_CRN));

Head Bucket Example
HeadBucketResult result = s3client.headBucket(headBucketRequest)
boolean KPEnabled = result.getIBMSSEKPEnabled();
String crn = result.getIBMSSEKPCUSTOMERROOTKEYCRN();
```

Using Aspera High-Speed Transfer

By installing the Aspera high-speed transfer library you can utilize high-speed file transfers within your application. The Aspera library is closed-source, and thus an optional dependency for the COS SDK (which uses an Apache license).

Tip: Each Aspera high-speed transfer session spawns an individual `ascp` process that runs on the client machine to perform the transfer. Ensure that your computing environment can allow this process to run.

You will need instances of the S3 Client and IAM Token Manager classes to initialize the `AsperaTransferManager`. The `s3Client` is required to get FASP connection information for the COS target bucket. The `tokenManager` is required to allow the Aspera high-speed transfer SDK to authenticate with the COS target bucket.

Initializing the `AsperaTransferManager`

Before initializing the `AsperaTransferManager`, make sure you've got working `s3Client` and `tokenManager` objects.

Note: It is advised to use `TokenManager tokenManager = new DefaultTokenManager(new DelegateTokenProvider(apiKey));` and avoid `.withTokenManager(tokenManager)` when building `AsperaTransferManager` with `AsperaTransferManagerBuilder`.

There isn't a lot of benefit to using a single session of Aspera high-speed transfer unless you expect to see significant noise or packet loss in the network. So we need to tell the `AsperaTransferManager` to use multiple sessions using the `AsperaConfig` class. This will split the transfer into a number of parallel **sessions** that send chunks of data whose size is defined by the **threshold** value.

The typical configuration for using multi-session should be:

- 2500 Mbps target rate
- 100 MB threshold (*this is the recommended value for most applications*)

```
$ AsperaTransferManagerConfig transferConfig = new AsperaTransferManagerConfig()
    .withMultiSession(true);

AsperaConfig asperaConfig = new AsperaConfig()
    .withTargetRateMbps(2500L)
    .withMultiSessionThresholdMb(100);

TokenManager tokenManager = new DefaultTokenManager(new DelegateTokenProvider(API_KEY));
```

```
AsperaTransferManager asperaTransferMgr = new AsperaTransferManagerBuilder(API_KEY, s3Client)
    .withAsperaTransferManagerConfig(transferConfig)
    .withAsperaConfig(asperaConfig)
    .build();
```

In the above example, the sdk will spawn enough sessions to attempt to reach the target rate of 2500 Mbps.

Alternatively, session management can be explicitly configured in the sdk. This is useful in cases where more precise control over network utilization is desired.

The typical configuration for using explicit multi-session should be:

- 2 or 10 sessions
- 100 MB threshold (*this is the recommended value for most applications*)

```
$ AsperaConfig asperaConfig = new AsperaConfig()
    .withMultiSession(2)
    .withMultiSessionThresholdMb(100);

TokenManager tokenManager = new DefaultTokenManager(new DelegateTokenProvider(API_KEY));

AsperaTransferManager asperaTransferMgr = new AsperaTransferManagerBuilder(API_KEY, s3Client)
    .withAsperaConfig(asperaConfig)
    .build();
```

Tip: For best performance in most scenarios, always make use of multiple sessions to minimize any processing associated with instantiating an Aspera high-speed transfer. **If your network capacity is at least 1 Gbps you should use 10 sessions.** Lower bandwidth networks should use two sessions.

Key Values

- `API_KEY` - An API key for a user or service ID with Writer or Manager roles

Tip: You need to provide an IAM API Key for constructing an `AsperaTransferManager`. HMAC Credentials are **NOT** currently supported. For more information on IAM, click here.

File Upload

```
$ String filePath = "<absolute-path-to-source-data>";
String bucketName = "<bucket-name>";
String itemName = "<item-name>";

// Load file
File inputFile = new File(filePath);

// Create AsperaTransferManager for FASP upload
AsperaTransferManager asperaTransferMgr = new AsperaTransferManagerBuilder(API_KEY, s3Client).build();

// Upload test file and report progress
Future<AsperaTransaction> asperaTransactionFuture = asperaTransferMgr.upload(bucketName, itemName, inputFile);
AsperaTransaction asperaTransaction = asperaTransactionFuture.get();
```

Key Values

- `<bucket-name>` - name of the bucket in your Object Storage service instance that has Aspera enabled.
- `<absolute-path-to-source-data>` - directory and file name to upload to Object Storage.
- `<item-name>` - name of the new object added to the bucket.

File Download

```
$ String bucketName = "<bucket-name>";
String outputPath = "<absolute-path-to-file>";
String itemName = "<item-name>";

// Create local file
```

```
File outputFile = new File(outputPath);
outputFile.createNewFile();

// Create AsperaTransferManager for FASP download
AsperaTransferManager asperaTransferMgr = new AsperaTransferManagerBuilder(COS_API_KEY_ID, s3Client)
    .withTokenManager(tokenManager)
    .withAsperaConfig(asperaConfig)
    .build();

// Download file
Future<AsperaTransaction> asperaTransactionFuture = asperaTransferMgr.download(bucketName, itemName, outputPath);
AsperaTransaction asperaTransaction = asperaTransactionFuture.get();
```

Key Values

- `<bucket-name>` - name of the bucket in your Object Storage service instance that has Aspera enabled.
- `<absolute-path-to-file>` - directory and file name to save from Object Storage.
- `<item-name>` - name of the object in the bucket.

Directory Upload

```
$ String bucketName = "<bucket-name>";
String directoryPath = "<absolute-path-to-directory-for-new-file>";
String directoryPrefix = "<virtual-directory-prefix>";
boolean includeSubDirectories = true;

// Load Directory
File inputDirectory = new File(directoryPath);

// Create AsperaTransferManager for FASP upload
AsperaTransferManager asperaTransferMgr = new AsperaTransferManagerBuilder(COS_API_KEY_ID, s3Client)
    .withTokenManager(tokenManager)
    .withAsperaConfig(asperaConfig)
    .build();

// Upload test directory
Future<AsperaTransaction> asperaTransactionFuture  = asperaTransferMgr.uploadDirectory(bucketName, directoryPrefix,
inputDirectory, includeSubDirectories);
AsperaTransaction asperaTransaction = asperaTransactionFuture.get();
```

Key Values

- `<bucket-name>` - name of the bucket in your Object Storage service instance that has Aspera enabled.
- `<absolute-path-to-directory>` - directory of the files to be uploaded to Object Storage.
- `<virtual-directory-prefix>` - name of the directory prefix to be added to each file upon upload. Use null or empty string to upload the files to the bucket root.

Directory Download

```
$ String bucketName = "<bucket-name>";
String directoryPath = "<absolute-path-to-directory>";
String directoryPrefix = "<virtual-directory-prefix>";
boolean includeSubDirectories = true;

// Load Directory
File outputDirectory = new File(directoryPath);

// Create AsperaTransferManager for FASP download
AsperaTransferManager asperaTransferMgr = new AsperaTransferManagerBuilder(COS_API_KEY_ID, s3Client)
    .withTokenManager(tokenManager)
    .withAsperaConfig(asperaConfig)
    .build();

// Download test directory
Future<AsperaTransaction> asperaTransactionFuture   = asperaTransferMgr.downloadDirectory(bucketName, directoryPrefix,
outputDirectory, includeSubDirectories);
AsperaTransaction asperaTransaction = asperaTransactionFuture.get();
```

Key Values

- `<bucket-name>` - name of the bucket in your Object Storage service instance that has Aspera enabled.
- `<absolute-path-to-directory>` - directory to save downloaded files from Object Storage.
- `<virtual-directory-prefix>` - name of the directory prefix of each file to download. Use null or empty string to download all files in the bucket.

Overriding Session Configuration on a Per Transfer Basis

You can override the multi-session configuration values on a per transfer basis by passing an instance of `AsperaConfig` to the upload and download overloaded methods. Using `AsperaConfig` you can specify the number of sessions and minimum file threshold size per session.

```
$ String bucketName = "<bucket-name>";
String filePath = "<absolute-path-to-file>";
String itemName = "<item-name>";

// Load file
File inputFile = new File(filePath);

// Create AsperaTransferManager for FASP upload
AsperaTransferManager asperaTransferMgr = new AsperaTransferManagerBuilder(API_KEY, s3Client)
.withTokenManager(TOKEN_MANAGER)
.withAsperaConfig(asperaConfig)
.build();

// Create AsperaConfig to set number of sessions
// and file threshold per session.
AsperaConfig asperaConfig = new AsperaConfig().
withMultiSession(10).
withMultiSessionThresholdMb(100);

// Upload test file and report progress
Future<AsperaTransaction> asperaTransactionFuture  = asperaTransferMgr.upload(bucketName, itemName, inputFile, asperaConfig,
null);
AsperaTransaction asperaTransaction = asperaTransactionFuture.get();
```

Monitoring Transfer Progress

The simplest way to monitor the progress of your file/directory transfers is to use the `isDone()` property that returns `true` when your transfer is complete.

```
$ Future<AsperaTransaction> asperaTransactionFuture  = asperaTransferMgr.downloadDirectory(bucketName, directoryPrefix,
outputDirectory, includeSubDirectories);
AsperaTransaction asperaTransaction = asperaTransactionFuture.get();

while (!asperaTransaction.isDone()) {
    System.out.println("Directory download is in progress");

    //pause for 3 seconds
    Thread.sleep(1000 * 3);
}
```

You can also check if a transfer is queued for processing by calling the `onQueue` method on the `AsperaTransaction`. `onQueue` will return a Boolean with `true` indicating that the transfer is queued.

```
$ Future<AsperaTransaction> asperaTransactionFuture  = asperaTransferMgr.downloadDirectory(bucketName, directoryPrefix,
outputDirectory, includeSubDirectories);
AsperaTransaction asperaTransaction = asperaTransactionFuture.get();

while (!asperaTransaction.isDone()) {
    System.out.println("Directory download is in queueing: " + asperaTransaction.onQueue());

    //pause for 3 seconds
    Thread.sleep(1000 * 3);
}
```

To check if a transfer is in progress call the progress method in `AsperaTransaction`.

```
$ Future<AsperaTransaction> asperaTransactionFuture  = asperaTransferMgr.downloadDirectory(bucketName, directoryPrefix,
outputDirectory, includeSubDirectories);
AsperaTransaction asperaTransaction = asperaTransactionFuture.get();

while (!asperaTransaction.isDone()) {
    System.out.println("Directory download is in progress: " + asperaTransaction.progress());

    //pause for 3 seconds
    Thread.sleep(1000 * 3);
}
```

Every transfer by default will have a `TransferProgress` attached to it. The `TransferProgress` will report the number of bytes transferred and the percentage transferred of the total bytes to transfer. To access a transfer's `TransferProgress` use the `getProgress` method in `AsperaTransaction`.

```
$ Future<AsperaTransaction> asperaTransactionFuture  = asperaTransferMgr.downloadDirectory(bucketName, directoryPrefix,
outputDirectory, includeSubDirectories);
AsperaTransaction asperaTransaction = asperaTransactionFuture.get();

while (!asperaTransaction.isDone()) {
    TransferProgress transferProgress = asperaTransaction.getProgress();

    //pause for 3 seconds
    Thread.sleep(1000 * 3);
}
```

To report the number of bytes transferred call the `getBytesTransferred` method on `TransferProgress`. To report the percentage transferred of the total bytes to transfer call the `getPercentTransferred` method on `TransferProgress`.

```
$ Future<AsperaTransaction> asperaTransactionFuture  = asperaTransferMgr.downloadDirectory(bucketName, directoryPrefix,
outputDirectory, includeSubDirectories);
AsperaTransaction asperaTransaction = asperaTransactionFuture.get();

while (!asperaTransaction.isDone()) {
    TransferProgress transferProgress = asperaTransaction.getProgress();

    System.out.println("Bytes transferred: " + transferProgress.getBytesTransferred());
    System.out.println("Percent transferred: " + transferProgress.getPercentTransferred());


    //pause for 3 seconds
    Thread.sleep(1000 * 3);
}
```

Pause/Resume/Cancel

The SDK provides the ability to manage the progress of file/directory transfers through the following methods of the `AsperaTransfer` object:

- `pause()`
- `resume()`
- `cancel()`

Tip: There are no side-effects from calling either of the methods outlined above. Proper clean up and housekeeping is handled by the SDK.

The following example shows a possible use for these methods:

```
$ String bucketName = "<bucket-name>";
String directoryPath = "<absolute-path-to-directory>";
String directoryPrefix = "<virtual-directory-prefix>";
boolean includeSubDirectories = true;

AsperaTransferManager asperaTransferMgr = new AsperaTransferManagerBuilder(COS_API_KEY_ID, _cos)
    .withTokenManager(TOKEN_MANAGER)
    .build();

File outputDirectory = new File(directoryName);
```

```
System.out.println("Starting directory download...");

//download the directory from cloud storage
Future<AsperaTransaction> asperaTransactionFuture  = asperaTransferMgr.downloadDirectory(bucketName, directoryPrefix,
outputDirectory, includeSubDirectories);
AsperaTransaction asperaTransaction = asperaTransactionFuture.get();

int pauseCount = 0;

while (!asperaTransaction.isDone()) {
    System.out.println("Directory download in progress...");

    //pause the transfer
    asperaTransfer.pause();

    //resume the transfer
    asperaTransfer.resume();

    //cancel the transfer
    asperaTransfer.cancel();
}

System.out.println("Directory download complete!");
```

Troubleshooting Aspera Issues

Issue: developers using the Oracle JDK on Linux or Mac OS X may experience unexpected and silent crashes during transfers

Cause: The native code requires its own signal handlers which could be overriding the JVM's signal handlers. It might be necessary to use the JVM's signal chaining facility.

IBM® JDK users or Microsoft® Windows users are not affected.

Solution: Link and load the JVM's signal chaining library.

- On Linux locate the `libjsig.so` shared library and set the following environment variable:
  - `LD_PRELOAD=<PATH_TO_SHARED_LIB>/libjsig.so`
- On Mac OS X locate the shared library `libjsig.dylib` and set the following environment variables:
  - `DYLD_INSERT_LIBRARIES=<PATH_TO_SHARED_LIB>/libjsig.dylib`
  - `DYLD_FORCE_FLAT_NAMESPACE=0`

Visit the Oracle® JDK documentation for more information about signal chaining.

Issue: `UnsatisfiedLinkError` on Linux

Cause: System unable to load dependent libraries. Errors such as the following may be seen in the application logs:

```
$ libfaspmanager2.so: libawt.so: cannot open shared object file: No such file or directory
```

Solution: Set the following environment variable:

`LD_LIBRARY_PATH=<JAVA_HOME>/jre/lib/amd64/server:<JAVA_HOME>/jre/lib/amd64`

Updating Metadata

There are two ways to update the metadata on an existing object:

- A `PUT` request with the new metadata and the original object contents
- Executing a `COPY` request with the new metadata specifying the original object as the copy source

Using PUT to update metadata

Note: The `PUT` request overwrites the existing contents of the object so it must first be downloaded and re-uploaded with the new metadata.

```
$ public static void updateMetadataPut(String bucketName, String itemName, String key, String value) throws IOException {
    System.out.printf("Updating metadata for item: %s\n", itemName);

    //retrieve the existing item to reload the contents
    S3Object item = _cos.getObject(new GetObjectRequest(bucketName, itemName));
    S3ObjectInputStream itemContents = item.getObjectContent();

    //read the contents of the item in order to set the content length and create a copy
    ByteArrayOutputStream output = new ByteArrayOutputStream();
    int b;
    while ((b = itemContents.read()) != -1) {
        output.write(b);
    }

    int contentLength = output.size();
    InputStream itemCopy = new ByteArrayInputStream(output.toByteArray());

    //set the new metadata
    HashMap<String, String> userMetadata = new HashMap<String, String>();
    userMetadata.put(key, value);

    ObjectMetadata metadata = new ObjectMetadata();
    metadata.setContentLength(contentLength);
    metadata.setUserMetadata(userMetadata);

    PutObjectRequest req = new PutObjectRequest(bucketName, itemName, itemCopy, metadata);

    _cos.putObject(req);

    System.out.printf("Updated metadata for item %s from bucket %s\n", itemName, bucketName);
}
```

Using COPY to update metadata

```
$ public static void updateMetadataCopy(String bucketName, String itemName, String key, String value) {
    System.out.printf("Updating metadata for item: %s\n", itemName);

    //set the new metadata
    HashMap<String, String> userMetadata = new HashMap<String, String>();
    userMetadata.put(key, value);

    ObjectMetadata metadata = new ObjectMetadata();
    metadata.setUserMetadata(userMetadata);

    //set the copy source to itself
    CopyObjectRequest req = new CopyObjectRequest(bucketName, itemName, bucketName, itemName);
    req.setNewObjectMetadata(metadata);

    _cos.copyObject(req);

    System.out.printf("Updated metadata for item %s from bucket %s\n", itemName, bucketName);
}
```

Using Immutable Object Storage

Add a protection configuration to an existing bucket

This implementation of the `PUT` operation uses the `protection` query parameter to set the retention parameters for an existing bucket. This operation allows you to set or change the minimum, default, and maximum retention period. This operation also allows you to change the protection state of the bucket.

Objects written to a protected bucket cannot be deleted until the protection period has expired and all legal holds on the object are removed. The bucket's default retention value is given to an object unless an object specific value is provided when the object is created. Objects in protected buckets that are no longer under retention (retention period has expired and the object does not have any legal holds), when overwritten, will again come under retention. The new retention period can be provided as part of the object overwrite request or the default retention time of the bucket will be given to the object.

The minimum and maximum supported values for the retention period settings `MinimumRetention`, `DefaultRetention`, and `MaximumRetention` are a minimum of 0 days and a maximum of 365243 days (1000 years).

```
public static void addProtectionConfigurationToBucket(String bucketName) {
    System.out.printf("Adding protection to bucket: %s\n", bucketName);

    BucketProtectionConfiguration newConfig = new BucketProtectionConfiguration()
        .withStatus(BucketProtectionStatus.Retention)
        .withMinimumRetentionInDays(10)
        .withDefaultRetentionInDays(100)
        .withMaximumRetentionInDays(1000);

    cos.setBucketProtection(bucketName, newConfig);

    System.out.printf("Protection added to bucket %s\n", bucketName);
}

public static void addProtectionConfigurationToBucketWithRequest(String bucketName) {
    System.out.printf("Adding protection to bucket: %s\n", bucketName);

    BucketProtectionConfiguration newConfig = new BucketProtectionConfiguration()
        .withStatus(BucketProtectionStatus.Retention)
        .withMinimumRetentionInDays(10)
        .withDefaultRetentionInDays(100)
        .withMaximumRetentionInDays(1000);

    SetBucketProtectionConfigurationRequest newRequest = new SetBucketProtectionConfigurationRequest()
        .withBucketName(bucketName)
        .withProtectionConfiguration(newConfig);

    cos.setBucketProtectionConfiguration(newRequest);

    System.out.printf("Protection added to bucket %s\n", bucketName);
}
```

Check protection on a bucket

```
public static void getProtectionConfigurationOnBucket(String bucketName) {
    System.out.printf("Retrieving protection configuration from bucket: %s\n", bucketName;

    BucketProtectionConfiguration config = cos.getBucketProtection(bucketName);

    String status = config.getStatus();

    System.out.printf("Status: %s\n", status);

    if (!status.toUpperCase().equals("DISABLED")) {
        System.out.printf("Minimum Retention (Days): %s\n", config.getMinimumRetentionInDays());
        System.out.printf("Default Retention (Days): %s\n", config.getDefaultRetentionInDays());
        System.out.printf("Maximum Retention (Days): %s\n", config.getMaximumRetentionInDays());
    }
}
```

Upload a protected object

Objects in protected buckets that are no longer under retention (retention period has expired and the object does not have any legal holds), when overwritten, will again come under retention. The new retention period can be provided as part of the object overwrite request or the default retention time of the bucket will be given to the object.

Value	Type	Description
`Retention-Period`	Non-negative integer (seconds)	Retention period to store on the object in seconds. The object can be neither overwritten nor deleted until the amount of time specified in the retention period has elapsed. If this field and `Retention-Expiration-Date` are specified a `400` error is returned. If neither is specified the bucket's `DefaultRetention` period will be used. Zero (`0`) is a legal value assuming the bucket's minimum retention period is also `0`.
`Retention-expiration-date`	Date (ISO 8601 Format)	Date on which it will be legal to delete or modify the object. You can only specify this or the Retention-Period header. If both are specified a `400` error will be returned. If neither is specified the bucket's DefaultRetention period will be used.

`Retention-legal-hold-id`	string	A single legal hold to apply to the object. A legal hold is a Y character long string. The object cannot be overwritten or deleted until all legal holds associated with the object are removed.

```
public static void putObjectAddLegalHold(String bucketName, String objectName, String fileText, String legalHoldId) {
    System.out.printf("Add legal hold %s to %s in bucket %s with a putObject operation.\n", legalHoldId, objectName, bucketName);

    InputStream newStream = new ByteArrayInputStream(fileText.getBytes(StandardCharsets.UTF_8));

    ObjectMetadata metadata = new ObjectMetadata();
    metadata.setContentLength(fileText.length());

    PutObjectRequest req = new PutObjectRequest(
        bucketName,
        objectName,
        newStream,
        metadata
    );
    req.setRetentionLegalHoldId(legalHoldId);

    cos.putObject(req);

    System.out.printf("Legal hold %s added to object %s in bucket %s\n", legalHoldId, objectName, bucketName);
}

public static void copyProtectedObject(String sourceBucketName, String sourceObjectName, String destinationBucketName, String
newObjectName) {
    System.out.printf("Copy protected object %s from bucket %s to %s/%s.\n", sourceObjectName, sourceBucketName,
destinationBucketName, newObjectName);

    CopyObjectRequest req = new CopyObjectRequest(
        sourceBucketName,
        sourceObjectName,
        destinationBucketName,
        newObjectName
    );
    req.setRetentionDirective(RetentionDirective.COPY);


    cos.copyObject(req);

    System.out.printf("Protected object copied from %s/%s to %s/%s\n", sourceObjectName, sourceBucketName, destinationBucketName,
newObjectName);
}
```

Add or remove a legal hold to or from a protected object

The object can support 100 legal holds:

- A legal hold identifier is a string of maximum length 64 characters and a minimum length of 1 character. Valid characters are letters, numbers, `!`, `_`, `.`, `*`, `(`, `)`, `-` and `'`.
- If the addition of the given legal hold exceeds 100 total legal holds on the object, the new legal hold will not be added, a `400` error will be returned.
- If an identifier is too long it will not be added to the object and a `400` error is returned.
- If an identifier contains invalid characters, it will not be added to the object and a `400` error is returned.
- If an identifier is already in use on an object, the existing legal hold is not modified and the response indicates the identifier was already in use with a `409` error.
- If an object does not have retention period metadata, a `400` error is returned and adding or removing a legal hold is not allowed.

The presence of a retention period header is required, otherwise a `400` error is returned.

The user making adding or removing a legal hold must have `Manager` permissions for this bucket.

```
public static void addLegalHoldToObject(String bucketName, String objectName, String legalHoldId) {
    System.out.printf("Adding legal hold %s to object %s in bucket %s\n", legalHoldId, objectName, bucketName);

    cos.addLegalHold(
        bucketName,
        objectName,
        legalHoldId
    );

    System.out.printf("Legal hold %s added to object %s in bucket %s!\n", legalHoldId, objectName, bucketName);
}

public static void deleteLegalHoldFromObject(String bucketName, String objectName, String legalHoldId) {
    System.out.printf("Deleting legal hold %s from object %s in bucket %s\n", legalHoldId, objectName, bucketName);

    cos.deleteLegalHold(
        bucketName,
        objectName,
        legalHoldId
    );

    System.out.printf("Legal hold %s deleted from object %s in bucket %s!\n", legalHoldId, objectName, bucketName);
}
```

Extend the retention period of a protected object

The retention period of an object can only be extended. It cannot be decreased from the currently configured value.

The retention expansion value is set in one of three ways:

- additional time from the current value (`Additional-Retention-Period` or similar method)
- new extension period in seconds (`Extend-Retention-From-Current-Time` or similar method)
- new retention expiry date of the object (`New-Retention-Expiration-Date` or similar method)

The current retention period stored in the object metadata is either increased by the given additional time or replaced with the new value, depending on the parameter that is set in the `extendRetention` request. In all cases, the extend retention parameter is checked against the current retention period and the extended parameter is only accepted if the updated retention period is greater than the current retention period.

Objects in protected buckets that are no longer under retention (retention period has expired and the object does not have any legal holds), when overwritten, will again come under retention. The new retention period can be provided as part of the object overwrite request or the default retention time of the bucket will be given to the object.

```
public static void extendRetentionPeriodOnObject(String bucketName, String objectName, Long additionalSeconds) {
    System.out.printf("Extend the retention period on %s in bucket %s by %s seconds.\n", objectName, bucketName,
additionalSeconds);

    ExtendObjectRetentionRequest req = new ExtendObjectRetentionRequest(
        bucketName,
        objectName)
        .withAdditionalRetentionPeriod(additionalSeconds);

    cos.extendObjectRetention(req);

    System.out.printf("New retention period on %s is %s\n", objectName, additionalSeconds);
}
```

List legal holds on a protected object

This operation returns:

- Object creation date
- Object retention period in seconds
- Calculated retention expiration date based on the period and creation date
- List of legal holds
- Legal hold identifier
- Timestamp when legal hold was applied

If there are no legal holds on the object, an empty `LegalHoldSet` is returned. If there is no retention period specified on the object, a `404` error is returned.

```
public static void listLegalHoldsOnObject(String bucketName, String objectName) {
    System.out.printf("List all legal holds on object %s in bucket %s\n", objectName, bucketName);
    ListLegalHoldsResult result = cos.listLegalHolds(
        bucketName,
        objectName
    );

    System.out.printf("Legal holds on bucket %s: \n", bucketName);

    List<LegalHold> holds = result.getLegalHolds();
    for (LegalHold hold : holds) {
        System.out.printf("Legal Hold: %s", hold);
    }
}
```

Create a hosted static website

This operation requires an import statement to be added:

```
import com.ibm.cloud.objectstorage.services.s3.model.model.BucketWebsiteConfiguration;
```

This operation provides the following upon configuration and requires a correctly configured client:

- Bucket configuration for suffix (index document)
- Bucket configuration for key (error document)

```
cosClient.setBucketWebsiteConfiguration("<bucket_name>", new BucketWebsiteConfiguration("index.html", "error.html"));
```

Next Steps

For more information, see the Javadoc. The source code for the project can be found in the GitHub repository.

Using Python

Python support is provided through a fork of the `boto3` library with features to make the most of IBM Cloud® Object Storage.

It can be installed from the Python Package Index through `pip install ibm-cos-sdk`.

Source code can be found at GitHub.

The `ibm_boto3` library provides complete access to the IBM Cloud® Object Storage API. Endpoints, an API key, and the instance ID must be specified during creation of a service resource or low-level client as shown in the following basic examples.

Tip: The service instance ID is also referred to as a *resource instance ID*. The value can be found by creating a service credential, or through the CLI.

Detailed documentation can be found at here.

Creating a client and sourcing credentials

To connect to COS, a client is created and configured using credential information (API key and service instance ID). These values can also be automatically sourced from a credentials file or from environment variables.

After generating a Service Credential, the resulting JSON document can be saved to `~/.bluemix/cos_credentials`. The SDK will automatically source credentials from this file unless other credentials are explicitly set during client creation. If the `cos_credentials` file contains HMAC keys the client authenticates with a signature, otherwise the client uses the provided API key to authenticate by using a bearer token (using an API key still requires the `config=Config(signature_version="oauth")` to be included during client creation).

If migrating from AWS S3, you can also source credentials data from `~/.aws/credentials` in the format:

```
[default]
aws_access_key_id = {API_KEY}
aws_secret_access_key = {SERVICE_INSTANCE_ID}
```

Note: If both `~/.bluemix/cos_credentials` and `~/.aws/credentials` exist, `cos_credentials` takes preference.

Gather required information

The following variables appear in the examples:

- `bucket_name` must be a unique and DNS-safe string. Because bucket names are unique across the entire system, these values need to be changed if this example is run multiple times. Note that names are reserved for 10 - 15 minutes after deletion.
- `ibm_api_key_id` is the value found in the Service Credential as `apikey`.
- `ibm_service_instance_id` is the value found in the Service Credential as `resource_instance_id`.
- `endpoint_url` is a service endpoint URL, inclusive of the `https://` protocol. This value is **not** the `endpoints` value that is found in the Service Credential. For more information about endpoints, see Endpoints and storage locations.
- `LocationConstraint` is a valid provisioning code that corresponds to the `endpoint` value.

Code Examples

Code examples are tested on supported release versions of Python.

Note: In your code, you must remove the angled brackets or any other excess characters that are provided here as illustration.

Initializing configuration

This example creates a `resource` object. A resource provides an object-oriented interface to COS. This allows for a higher level of abstraction than the low-level calls provided by a client object.

Important: Note that some operations (such as Aspera high-speed transfer) require a `client` object. Aspera itself requires Python version 3.6.

Important: Legacy Notice: Support for Aspera is considered legacy. Instead, use the Aspera Transfer SDK.

Python

```
import ibm_boto3
from ibm_botocore.client import Config, ClientError

# Constants for IBM COS values
COS_ENDPOINT = "<endpoint>" # Current list avaiable at https://control.cloud-object-storage.cloud.ibm.com/v2/endpoints
COS_API_KEY_ID = "<api-key>" # eg "W00YixxxxxxxxxxMB-odB-2ySfTrFBIQQWanc--P3byk"
COS_INSTANCE_CRN = "<service-instance-id>" # eg "crn:v1:bluemix:public:cloud-object-
storage:global:a/3bf0d9003xxxxxxxxxx1c3e97696b71c:d6f04d83-6c4f-4a62-a165-696756d63903::"

# Create resource
cos_resource = ibm_boto3.resource("s3",
    ibm_api_key_id=COS_API_KEY_ID,
    ibm_service_instance_id=COS_INSTANCE_CRN,
    config=Config(signature_version="oauth"),
    endpoint_url=COS_ENDPOINT
)
```

A client provides a low-level interface to the COS S3 API. This allows for processing HTTP responses directly, rather than making use of abstracted methods and attributes provided by a resource to access the information contained in headers or XML response payloads.

```
$
import ibm_boto3
from ibm_botocore.client import Config, ClientError

# Constants for IBM COS values
COS_ENDPOINT = "<endpoint>" # Current list avaiable at https://control.cloud-object-storage.cloud.ibm.com/v2/endpoints
COS_API_KEY_ID = "<api-key>" # eg "W00YixxxxxxxxxxMB-odB-2ySfTrFBIQQWanc--P3byk"
COS_INSTANCE_CRN = "<service-instance-id>" # eg "crn:v1:bluemix:public:cloud-object-
storage:global:a/3bf0d9003xxxxxxxxxx1c3e97696b71c:d6f04d83-6c4f-4a62-a165-696756d63903::"

# Create client
cos_client = ibm_boto3.client("s3",
    ibm_api_key_id=COS_API_KEY_ID,
    ibm_service_instance_id=COS_INSTANCE_CRN,
    config=Config(signature_version="oauth"),
    endpoint_url=COS_ENDPOINT
)
```

Key Values

- `<endpoint>` - public endpoint for your cloud Object Storage with schema prefixed ('https://') (available from the IBM Cloud Dashboard). For more information about endpoints, see Endpoints and storage locations.
- `<api-key>` - api key generated when creating the service credentials (write access is required for creation and deletion examples)
- `<service-instance-id>` - resource ID for your cloud Object Storage (available through IBM Cloud CLI or IBM Cloud Dashboard)
- `<location>` - default location for your cloud Object Storage (must match the region that is used for `<endpoint>`)

SDK References

- `ServiceResource`

Creating a new bucket

The examples below uses client which is a low level interface.

A list of valid provisioning codes for `LocationConstraint` can be referenced in the Storage Classes guide.

Python

```
def create_bucket(bucket_name):
    print("Creating new bucket: {0}".format(bucket_name))
    try:
        cos_client.create_bucket(
            Bucket=bucket_name,
            CreateBucketConfiguration={
                "LocationConstraint":COS_BUCKET_LOCATION
            }
        )
        print("Bucket: {0} created!".format(bucket_name))
    except ClientError as be:
        print("CLIENT ERROR: {0}\n".format(be))
    except Exception as e:
        print("Unable to create bucket: {0}".format(e))
```

SDK References

Methods

- `create_bucket`

Creating a new text file

Python

```
def create_text_file(bucket_name, item_name, file_text):
    print("Creating new item: {0}".format(item_name))
    try:
        cos_client.put_object(
            Bucket=bucket_name,
            Key=item_name,
            Body=file_text
        )
        print("Item: {0} created!".format(item_name))
    except ClientError as be:
        print("CLIENT ERROR: {0}\n".format(be))
    except Exception as e:
        print("Unable to create text file: {0}".format(e))
```

SDK References

Methods

- `put_object`

List available buckets

Python

```
def get_buckets():
    print("Retrieving list of buckets")
    try:
        buckets = cos_client.list_buckets()
        for bucket in buckets["Buckets"]:
            print("Bucket Name: {0}".format(bucket["Name"]))
    except ClientError as be:
        print("CLIENT ERROR: {0}\n".format(be))
    except Exception as e:
        print("Unable to retrieve list buckets: {0}".format(e))
```

SDK References

Methods

- list_buckets

List items in a bucket

Python

```
def get_bucket_contents(bucket_name):
    print("Retrieving bucket contents from: {0}".format(bucket_name))
    try:
        files = cos_client.list_objects(Bucket=bucket_name)
        for file in files.get("Contents", []):
            print("Item: {0} ({1} bytes).".format(file["Key"], file["Size"]))
    except ClientError as be:
        print("CLIENT ERROR: {0}\n".format(be))
    except Exception as e:
        print("Unable to retrieve bucket contents: {0}".format(e))
```

SDK References

Methods

- list_objects

Get file contents of particular item

Python

```
def get_item(bucket_name, item_name):
    print("Retrieving item from bucket: {0}, key: {1}".format(bucket_name, item_name))
    try:
        file = cos_client.get_object(Bucket=bucket_name, Key=item_name)
        print("File Contents: {0}".format(file["Body"].read()))
    except ClientError as be:
        print("CLIENT ERROR: {0}\n".format(be))
    except Exception as e:
        print("Unable to retrieve file contents: {0}".format(e))
```

SDK References

Methods

- get_object

Delete an item from a bucket

Python

```
def delete_item(bucket_name, object_name):
    try:
        cos_client.delete_object(Bucket=bucket_name, Key=object_name)
        print("Item: {0} deleted!\n".format(object_name))
    except ClientError as be:
        print("CLIENT ERROR: {0}\n".format(be))
    except Exception as e:
        print("Unable to delete object: {0}".format(e))
```

SDK References

Methods

- delete_object

Delete multiple items from a bucket

Tip: The delete request can contain a maximum of 1000 keys that you want to delete. While this is useful in reducing the per-request performance hit, be mindful when deleting many keys. Also, take into account the sizes of the objects to ensure suitable performance.

Python

```
def delete_items(bucket_name):
    try:
        delete_request = {
            "Objects": [
                { "Key": "deletetest/testfile1.txt" },
                { "Key": "deletetest/testfile2.txt" },
                { "Key": "deletetest/testfile3.txt" },
                { "Key": "deletetest/testfile4.txt" },
                { "Key": "deletetest/testfile5.txt" }
            ]
        }

        response = cos_client.delete_objects(
            Bucket=bucket_name,
            Delete=delete_request
        )

        print("Deleted items for {0}\n".format(bucket_name))
        print(json.dumps(response.get("Deleted"), indent=4))
    except ClientError as be:
        print("CLIENT ERROR: {0}\n".format(be))
    except Exception as e:
        print("Unable to copy item: {0}".format(e))
```

SDK References

Methods

- delete_objects

Delete a bucket

Python

```
def delete_bucket(bucket_name):
    print("Deleting bucket: {0}".format(bucket_name))
    try:
        cos_client.delete_bucket(Bucket=bucket_name)
        print("Bucket: {0} deleted!".format(bucket_name))
    except ClientError as be:
        print("CLIENT ERROR: {0}\n".format(be))
    except Exception as e:
        print("Unable to delete bucket: {0}".format(e))
```

SDK References

Methods

- delete_bucket

Note: The bucket names are reserved for 10 - 15 minutes after deletion.

Run a multi-part upload

Upload binary file (preferred method)

The upload_fileobj method of the S3 Object automatically runs a multi-part upload when necessary. The TransferConfig class is used to determine the threshold for using the multi-part upload.

Python

```
def multi_part_upload(bucket_name, item_name, file_path):
    try:
        print("Starting file transfer for {0} to bucket: {1}\n".format(item_name, bucket_name))
        # set 5 MB chunks
        part_size = 1024 * 1024 * 5

        # set threadhold to 15 MB
        file_threshold = 1024 * 1024 * 15

        # set the transfer threshold and chunk size
        transfer_config = ibm_boto3.s3.transfer.TransferConfig(
            multipart_threshold=file_threshold,
            multipart_chunksize=part_size
        )

        # the upload_fileobj method will automatically execute a multi-part upload
        # in 5 MB chunks for all files over 15 MB
        with open(file_path, "rb") as file_data:
            cos_client.upload_fileobj(
                Bucket=bucket_name,
                Key=item_name,
                Fileobj=file_data,
                Config=transfer_config
            )

        print("Transfer for {0} Complete!\n".format(item_name))
    except ClientError as be:
        print("CLIENT ERROR: {0}\n".format(be))
    except Exception as e:
        print("Unable to complete multi-part upload: {0}".format(e))
```

SDK References

Methods

- upload_fileobj

Manually run a multi-part upload

If wanted, the S3.Client class can be used to perform a multi-part upload. This can be useful if more control over the upload process is necessary.

Python

```
def multi_part_upload_manual(bucket_name, item_name, file_path):
    try:
        # create client object
        cos_client = ibm_boto3.client("s3",
            ibm_api_key_id=COS_API_KEY_ID,
            ibm_service_instance_id=COS_SERVICE_CRN,
            config=Config(signature_version="oauth"),
            endpoint_url=COS_ENDPOINT
        )

        print("Starting multi-part upload for {0} to bucket: {1}\n".format(item_name, bucket_name))
        # initiate the multi-part upload
        mp = cos_client.create_multipart_upload(
            Bucket=bucket_name,
            Key=item_name
        )

        upload_id = mp["UploadId"]

        # min 20MB part size
        part_size = 1024 * 1024 * 20
        file_size = os.stat(file_path).st_size
        part_count = int(math.ceil(file_size / float(part_size)))
        data_packs = []
        position = 0
        part_num = 0

        # begin uploading the parts
        with open(file_path, "rb") as file:
            for i in range(part_count):
                part_num = i + 1
                part_size = min(part_size, (file_size - position))

                print("Uploading to {0} (part {1} of {2})".format(item_name, part_num, part_count))

                file_data = file.read(part_size)

                mp_part = cos_client.upload_part(
                    Bucket=bucket_name,
                    Key=item_name,
                    PartNumber=part_num,
                    Body=file_data,
                    ContentLength=part_size,
                    UploadId=upload_id
                )

                data_packs.append({
                    "ETag":mp_part["ETag"],
                    "PartNumber":part_num
                })

                position += part_size

        # complete upload
        cos_client.complete_multipart_upload(
            Bucket=bucket_name,
            Key=item_name,
            UploadId=upload_id,
            MultipartUpload={
                "Parts": data_packs
            }
        )
        print("Upload for {0} Complete!\n".format(item_name))
    except ClientError as be:
        # abort the upload
        cos_client.abort_multipart_upload(
            Bucket=bucket_name,
            Key=item_name,
            UploadId=upload_id
        )
        print("Multi-part upload aborted for {0}\n".format(item_name))
        print("CLIENT ERROR: {0}\n".format(be))
    except Exception as e:
        print("Unable to complete multi-part upload: {0}".format(e))
```

SDK References continued

Classes

- S3.Client

Methods

- abort_multipart_upload
- complete_multipart_upload
- create_multipart_upload
- upload_part

Large Object Upload by using TransferManager

The `TransferManager` provides another way to run large file transfers by automatically incorporating multi-part uploads whenever necessary setting configuration parameters.

Python

```
def upload_large_file(bucket_name, item_name, file_path):
    print("Starting large file upload for {0} to bucket: {1}".format(item_name, bucket_name))

    # set the chunk size to 5 MB
    part_size = 1024 * 1024 * 5

    # set threadhold to 5 MB
    file_threshold = 1024 * 1024 * 5

    # Create client connection
    cos_client = ibm_boto3.client("s3",
        ibm_api_key_id=COS_API_KEY_ID,
        ibm_service_instance_id=COS_SERVICE_CRN,
        config=Config(signature_version="oauth"),
        endpoint_url=COS_ENDPOINT
    )

    # set the transfer threshold and chunk size in config settings
    transfer_config = ibm_boto3.s3.transfer.TransferConfig(
        multipart_threshold=file_threshold,
        multipart_chunksize=part_size
    )

    # create transfer manager
    transfer_mgr = ibm_boto3.s3.transfer.TransferManager(cos_client, config=transfer_config)

    try:
        # initiate file upload
        future = transfer_mgr.upload(file_path, bucket_name, item_name)

        # wait for upload to complete
        future.result()

        print ("Large file upload complete!")
    except Exception as e:
        print("Unable to complete large file upload: {0}".format(e))
    finally:
        transfer_mgr.shutdown()
```

List items in a bucket (v2)

The S3.Client object has an updated method to list the contents (list_objects_v2). This method allows you to limit the number of records that are returned and retrieve the records in batches. This might be useful for paging your results within an application and improve performance.

Python

```
def get_bucket_contents_v2(bucket_name, max_keys):
    print("Retrieving bucket contents from: {0}".format(bucket_name))
    try:
        # create client object
        cos_client = ibm_boto3.client("s3",
            ibm_api_key_id=COS_API_KEY_ID,
            ibm_service_instance_id=COS_SERVICE_CRN,
            config=Config(signature_version="oauth"),
            endpoint_url=COS_ENDPOINT)
        more_results = True
        next_token = ""

        while (more_results):
            response = cos_client.list_objects_v2(Bucket=bucket_name, MaxKeys=max_keys, ContinuationToken=next_token)
            files = response["Contents"]
            for file in files:
                print("Item: {0} ({1} bytes).".format(file["Key"], file["Size"]))

            if (response["IsTruncated"]):
                next_token = response["NextContinuationToken"]
                print("...More results in next batch!\n")
            else:
                more_results = False
                next_token = ""

    except ClientError as be:
        print("CLIENT ERROR: {0}\n".format(be))
    except Exception as e:
        print("Unable to retrieve bucket contents: {0}".format(e))
```

SDK References

Methods

- `list_objects_v2`

Using Key Protect

Key Protect can be added to a storage bucket to encrypt sensitive data at rest in the cloud.

Before You Begin

The following items are necessary in order to create a bucket with Key-Protect enabled:

- A Key Protect service provisioned
- A Root key available (either generated or imported)

Retrieving the Root Key CRN

1. Retrieve the instance ID for your Key Protect service
2. Use the Key Protect API to retrieve all your available keys
    - You can either use `curl` commands or an API REST Client such as Postman to access the Key Protect API.
3. Retrieve the CRN of the root key you use to enabled Key Protect on your bucket. The CRN looks similar to below:

`crn:v1:bluemix:public:kms:us-south:a/3d624cd74a0dea86ed8efe3101341742:90b6a1db-0fe1-4fe9-b91e-962c327df531:key:0bg3e33e-a866-50f2-b715-5cba2bc93234`

Creating a bucket with key-protect enabled

Python

```
COS_KP_ALGORITHM = "<algorithm>"
COS_KP_ROOTKEY_CRN = "<root-key-crn>"

# Create a new bucket with key protect (encryption)
def create_bucket_kp(bucket_name):
    print("Creating new encrypted bucket: {0}".format(bucket_name))
    try:
        cos_client.create_bucket(
            Bucket=bucket_name,
            CreateBucketConfiguration={
                "LocationConstraint":COS_BUCKET_LOCATION
            },
            IBMSSEKPEncryptionAlgorithm=COS_KP_ALGORITHM,
            IBMSSEKPCustomerRootKeyCrn=COS_KP_ROOTKEY_CRN
        )
        print("Encrypted Bucket: {0} created!".format(bucket_name))
    except ClientError as be:
        print("CLIENT ERROR: {0}\n".format(be))
    except Exception as e:
        print("Unable to create encrypted bucket: {0}".format(e))
```

Key Values

- `<algorithm>` - The encryption algorithm that is used for new objects added to the bucket (Default is AES256).
- `<root-key-crn>` - CRN of the Root Key that is obtained from the Key Protect service.

SDK References

Methods

- [`create_bucket`]

Using Aspera High-Speed Transfer

Legacy Notice: Support for Aspera is considered legacy. Users are recommended to use Aspera Transfer SDK[https://developer.ibm.com/apis/catalog/aspera--aspera-transfer-sdk/API Reference].

Legacy Notice: Support for Aspera is considered legacy. Users are recommended to use Aspera Transfer SDK.

By installing the Aspera high-speed transfer library, you can use high-speed file transfers within your application. The Aspera library is closed-source, and thus an optional dependency for the COS SDK (which uses an Apache license).

Tip: Each Aspera session creates an individual `ascp` process that runs on the client machine to perform the transfer. Ensure that your computing environment can allow this process to run.

Initializing the AsperaTransferManager

Important: Before initializing the `AsperaTransferManager`, make sure that you have a working `client` (not a `resource` or `session`) object.

Python

```
import ibm_boto3
from ibm_botocore.client import Config
from ibm_s3transfer.aspera.manager import AsperaTransferManager

COS_ENDPOINT = "<endpoint>" # Current list avaiable at https://control.cloud-object-storage.cloud.ibm.com/v2/endpoints
COS_API_KEY_ID = "<api-key>"
COS_RESOURCE_CRN = "<resource-instance-id>"
COS_BUCKET_LOCATION = "<location>"

# Create resource
cos_client = ibm_boto3.client("s3",
    ibm_api_key_id=COS_API_KEY_ID,
    ibm_service_instance_id=COS_RESOURCE_CRN,
    config=Config(signature_version="oauth"),
    endpoint_url=COS_ENDPOINT
)

transfer_manager = AsperaTransferManager(cos)
```

Tip: You need to provide an IAM API Key for Aspera high-speed transfers. HMAC Credentials are **NOT** currently supported. For more information on IAM, click here.

To get the highest throughput, split the transfer into a specified number of parallel **sessions** that send chunks of data whose size is defined by a **threshold** value.

The typical configuration for using multi-session should be:

- 2500 Mbps target rate
- 100 MB threshold (*this is the recommended value for most applications*)

Python

```
ms_transfer_config = AsperaConfig(multi_session="all",
                                  target_rate_mbps=2500,
                                  multi_session_threshold_mb=100)
```

In the above example, the sdk spawns enough sessions to attempt to reach the target rate of 2500 Mbps.

Session management can also be explicitly configured in the SDK. This is useful in cases where more precise control over network utilization is wanted.

The typical configuration for using explicit multi-session should be:

- 2 or 10 sessions
- 100 MB threshold (*this is the recommended value for most applications*)

Python

```
from ibm_s3transfer.aspera.manager import AsperaConfig
# Configure 2 sessions for transfer
ms_transfer_config = AsperaConfig(multi_session=2,
                                  multi_session_threshold_mb=100)

# Create the Aspera Transfer Manager
transfer_manager = AsperaTransferManager(client=client,
                                         transfer_config=ms_transfer_config)
```

Tip: For best performance in most scenarios, always make use of multiple sessions to minimize any processing that is associated with instantiating an Aspera high-speed transfer. **If your network capacity is at least 1 Gbps, you should use 10 sessions.** Lower bandwidth networks should use two sessions.

File Upload

Python

```
bucket_name = "<bucket-name>"
upload_filename = "<absolute-path-to-file>"
object_name = "<item-name>"

# Create Transfer manager
with AsperaTransferManager(client) as transfer_manager:

    # Perform upload
    future = transfer_manager.upload(upload_filename, bucket_name, object_name)

    # Wait for upload to complete
    future.result()
```

Key Values

- `<bucket-name>` - name of the target bucket
- `<absolute-path-to-file>` - directory path and file name to the file to be uploaded
- `<item-name>` - name of the new file added to the bucket

File Download

Python

```
bucket_name = "<bucket-name>"
download_filename = "<absolute-path-to-file>"
object_name = "<object-to-download>"

# Create Transfer manager
with AsperaTransferManager(client) as transfer_manager:

    # Get object with Aspera
    future = transfer_manager.download(bucket_name, object_name, download_filename)

    # Wait for download to complete
    future.result()
```

Key Values

- `<bucket-name>` - name of the bucket in your Object Storage service instance that has Aspera enabled.
- `<absolute-path-to-file>` - directory and file name where save the file to the local system.
- `<object-to-download>` - name of the file in the bucket to download.

Directory Upload

Python

```
bucket_name = "<bucket-name>"
# THIS DIRECTORY MUST EXIST LOCALLY, and have objects in it.
local_upload_directory = "<absolute-path-to-directory>"
# THIS SHOULD NOT HAVE A LEADING "/"
remote_directory = "<object prefix>"

# Create Transfer manager
with AsperaTransferManager(client) as transfer_manager:

    # Perform upload
    future = transfer_manager.upload_directory(local_upload_directory, bucket_name, remote_directory)

    # Wait for upload to complete
    future.result()
```

Key Values

- `<bucket-name>` - name of the bucket in your Object Storage service instance that has Aspera enabled
- `<absolute-path-to-directory>` - local directory that contains the files to be uploaded. Must have leading and trailing `/` (that is, `/Users/testuser/Documents/Upload/`)
- `<object prefix>` - name of the directory in the bucket to store the files. Must not have a leading slash `/` (that is, `newuploads/`)

Directory Download

Python

```
bucket_name = "<bucket-name>"
# THIS DIRECTORY MUST EXIST LOCALLY
local_download_directory = "<absolute-path-to-directory>"
remote_directory = "<object prefix>"

# Create Transfer manager
with AsperaTransferManager(client) as transfer_manager:

    # Get object with Aspera
    future = transfer_manager.download_directory(bucket_name, remote_directory, local_download_directory)

    # Wait for download to complete
    future.result()
```

Key Values

- `<bucket-name>` - name of the bucket in your Object Storage service instance that has Aspera enabled
- `<absolute-path-to-directory>` - local directory to save the downloaded files. Must have leading and trailing slash `/` (that is `/Users/testuser/Downloads/`)
- `<object prefix>` - name of the directory in the bucket to store the files. Must not have a leading slash `/` (that is, `todownload/`)

Using Subscribers

Subscribers provide observability into transfers by attaching custom callback methods. All transfers transition between the following phases:

`Queued - In Progress - Done`

There are three available subscribers for each phase:

- `CallbackOnQueued()` - called when a new transfer has been added to the `AsperaTransferManager`
- `CallbackOnProgress()` - called when a transfer has transmitted data (fired repeatedly while the transfer is in progress).
- `CallbackOnDone()` - called once the transfer is completed

Python

```
bucket_name = "<bucket-name>"
local_download_directory = "<absolute-path-to-directory>"
remote_directory = "<object prefix>"

# Subscriber callbacks
class CallbackOnQueued(AsperaBaseSubscriber):
    def __init__(self):
        pass

    def on_queued(self, future, **kwargs):
        print("Directory download queued.")

class CallbackOnProgress(AsperaBaseSubscriber):
    def __init__(self):
        pass

    def on_progress(self, future, bytes_transferred, **kwargs):
        print("Directory download in progress: %s bytes transferred" % bytes_transferred)

class CallbackOnDone(AsperaBaseSubscriber):
    def __init__(self):
        pass

    def on_done(self, future, **kwargs):
        print("Downloads complete!")

# Create Transfer manager
transfer_manager = AsperaTransferManager(client)

# Attach subscribers
subscribers = [CallbackOnQueued(), CallbackOnProgress(), CallbackOnDone()]

# Get object with Aspera
future = transfer_manager.download_directory(bucket_name, remote_directory, local_download_directory, None, subscribers)

# Wait for download to complete
future.result()
```

Key Values

- `<bucket-name>` - name of the bucket in your Object Storage service instance that has Aspera enabled
- `<absolute-path-to-directory>` - local directory to save the downloaded files. Must have leading and trailing slash `/` (that is, `/Users/testuser/Downloads/`)
- `<object prefix>` - name of the directory in the bucket to store the files. Must not have a leading slash `/` (that is, `todownload/`)

The sample code above produces the following output:

```
$ Directory download queued.
Directory download in progress: 5632 bytes transferred
Directory download in progress: 1047552 bytes transferred
...
Directory download in progress: 53295130 bytes transferred
Directory download in progress: 62106855 bytes transferred
Download complete!
```

Pause/Resume/Cancel

The SDK provides the ability to manage the progress of file/directory transfers through the following methods of the `AsperaTransferFuture` object:

- `pause()`
- `resume()`

- `cancel()`

Tip: There are no side-effects from calling either of the methods outlined above. Proper clean up and housekeeping is handled by the SDK.

Python

```
# Create Transfer manager
bucket_name = "<bucket-name>"
local_download_directory = "<absolute-path-to-directory>"
remote_directory = "<object prefix>"

with AsperaTransferManager(client) as transfer_manager:

    # download a directory with Aspera
    future = transfer_manager.download_directory(bucket_name, remote_directory, local_download_directory, None, None)

    # pause the transfer
    future.pause()

    # resume the transfer
    future.resume()

    # cancel the transfer
    future.cancel()
```

Troubleshooting Aspera Issues

Issue: Developers using any version of Python besides 3.6 may experience failures when installing or using Aspera SDK.

Cause: If there are different versions of Python installed on your environment, then you might encounter installation failures when you try to install the Aspera SDK. This can be caused by a missing DLL files or wrong DLL in path.

Solution: The first step to resolving this issue would be to reinstall the Aspera libraries. There might have been a failure during the installation. As a result this might have affected the DLL files. If that does not resolve the issues, then you will be required to update your version of Python. If you are unable to do this, then you can use installation Intel® Distribution for Python*. This should allow you to install the Aspera SDK on Python 3.6.x without any issues.

Updating metadata

There are two ways to update the metadata on an existing object:

- A `PUT` request with the new metadata and the original object contents
- Running a `COPY` request with the new metadata specifying the original object as the copy source

Using PUT to update metadata

Note: The `PUT` request overwrites the existing contents of the object so it must first be downloaded and re-uploaded with the new metadata.

Python

```
def update_metadata_put(bucket_name, item_name, key, value):
    try:
        # retrieve the existing item to reload the contents
        response = cos_client.get_object(Bucket=bucket_name, Key=item_name)
        existing_body = response.get("Body").read()

        # set the new metadata
        new_metadata = {
            key: value
        }

        cos_client.put_object(Bucket=bucket_name, Key=item_name, Body=existing_body, Metadata=new_metadata)

        print("Metadata update (PUT) for {0} Complete!\n".format(item_name))
    except ClientError as be:
        print("CLIENT ERROR: {0}\n".format(be))
    except Exception as e:
        log_error("Unable to update metadata: {0}".format(e))
```

Using COPY to update metadata

Python

```
def update_metadata_copy(bucket_name, item_name, key, value):
    try:
        # set the new metadata
        new_metadata = {
            key: value
        }

        # set the copy source to itself
        copy_source = {
            "Bucket": bucket_name,
            "Key": item_name
        }

        cos_client.copy_object(Bucket=bucket_name, Key=item_name, CopySource=copy_source, Metadata=new_metadata,
MetadataDirective="REPLACE")

        print("Metadata update (COPY) for {0} Complete!\n".format(item_name))
    except ClientError as be:
        print("CLIENT ERROR: {0}\n".format(be))
    except Exception as e:
        log_error("Unable to update metadata: {0}".format(e))
```

Using Immutable Object Storage

Add a protection configuration to an existing bucket

Objects written to a protected bucket cannot be deleted until the protection period has expired and all legal holds on the object are removed. The bucket's default retention value is given to an object unless an object-specific value is provided when the object is created. Objects in protected buckets that are no longer under retention (retention period has expired and the object does not have any legal holds), when overwritten, will again come under retention. The new retention period can be provided as part of the object overwrite request or the default retention time of the bucket will be given to the object.

The minimum and maximum supported values for the retention period settings `MinimumRetention`, `DefaultRetention`, and `MaximumRetention` are a minimum of 0 days and a maximum of 365243 days (1000 years).

Python

```
def add_protection_configuration_to_bucket(bucket_name):
    try:
        new_protection_config = {
            "Status": "Retention",
            "MinimumRetention": {"Days": 10},
            "DefaultRetention": {"Days": 100},
            "MaximumRetention": {"Days": 1000}
        }

        cos_client.put_bucket_protection_configuration(Bucket=bucket_name, ProtectionConfiguration=new_protection_config)

        print("Protection added to bucket {0}\n".format(bucket_name))
    except ClientError as be:
        print("CLIENT ERROR: {0}\n".format(be))
    except Exception as e:
        print("Unable to set bucket protection config: {0}".format(e))
```

Check protection on a bucket

Python

```
def get_protection_configuration_on_bucket(bucket_name):
    try:
        response = cos_client.get_bucket_protection_configuration(Bucket=bucket_name)
        protection_config = response.get("ProtectionConfiguration")

        print("Bucket protection config for {0}\n".format(bucket_name))
        print(protection_config)
        print("\n")
    except ClientError as be:
        print("CLIENT ERROR: {0}\n".format(be))
    except Exception as e:
        print("Unable to get bucket protection config: {0}".format(e))
```

Upload a protected object

Objects in protected buckets that are no longer under retention (retention period has expired and the object does not have any legal holds), when overwritten, will again come under retention. The new retention period can be provided as part of the object overwrite request or the default retention time of the bucket will be given to the object.

Value	Type	Description
`Retention-Period`	Non-negative integer (seconds)	Retention period to store on the object in seconds. The object can be neither overwritten nor deleted until the amount of time that is specified in the retention period has elapsed. If this field and `Retention-Expiration-Date` are specified a `400` error is returned. If neither is specified the bucket's `DefaultRetention` period will be used. Zero (`0`) is a legal value assuming the bucket's minimum retention period is also `0`.
`Retention-expiration-date`	Date (ISO 8601 Format)	Date on which it will be legal to delete or modify the object. You can only specify this or the Retention-Period header. If both are specified a `400` error will be returned. If neither is specified the bucket's DefaultRetention period will be used.
`Retention-legal-hold-id`	string	A single legal hold to apply to the object. A legal hold is a Y character long string. The object cannot be overwritten or deleted until all legal holds associated with the object are removed.

Python

```
def put_object_add_legal_hold(bucket_name, object_name, file_text, legal_hold_id):
    print("Add legal hold {0} to {1} in bucket {2} with a putObject operation.\n".format(legal_hold_id, object_name,
bucket_name))
    cos_client.put_object(
        Bucket=bucket_name,
        Key=object_name,
        Body=file_text,
        RetentionLegalHoldId=legal_hold_id)
    print("Legal hold {0} added to object {1} in bucket {2}\n".format(legal_hold_id, object_name, bucket_name))

def copy_protected_object(source_bucket_name, source_object_name, destination_bucket_name, new_object_name):
    print("Copy protected object {0} from bucket {1} to {2}/{3}.\n".format(source_object_name, source_bucket_name,
destination_bucket_name, new_object_name))

    copy_source = {
        "Bucket": source_bucket_name,
        "Key": source_object_name
    }

    cos_client.copy_object(
        Bucket=destination_bucket_name,
        Key=new_object_name,
        CopySource=copy_source,
        RetentionDirective="Copy"
    )

    print("Protected object copied from {0}/{1} to {2}/{3}\n".format(source_bucket_name, source_object_name,
destination_bucket_name, new_object_name));

def complete_multipart_upload_with_retention(bucket_name, object_name, upload_id, retention_period):
    print("Completing multi-part upload for object {0} in bucket {1}\n".format(object_name, bucket_name))
    cos_client.complete_multipart_upload(
        Bucket=bucket_name,
        Key=object_name,
        MultipartUpload={
            "Parts":[{
                "ETag": part["ETag"],
                "PartNumber": 1
            }]
        },
        UploadId=upload_id,
        RetentionPeriod=retention_period
    )

    print("Multi-part upload completed for object {0} in bucket {1}\n".format(object_name, bucket_name))

def upload_file_with_retention(bucket_name, object_name, path_to_file, retention_period):
    print("Uploading file {0} to object {1} in bucket {2}\n".format(path_to_file, object_name, bucket_name))

    args = {
        "RetentionPeriod": retention_period
    }

    cos_client.upload_file(
        Filename=path_to_file,
        Bucket=bucket_name,
        Key=object_name,
        ExtraArgs=args
    )

    print("File upload complete to object {0} in bucket {1}\n".format(object_name, bucket_name))
```

Add or remove a legal hold to or from a protected object

The object can support 100 legal holds:

- A legal hold identifier is a string of maximum length 64 characters and a minimum length of 1 character. Valid characters are letters, numbers, and the symbols `!`, `_`, `.`, `*`, `(`, `)`, and `-`.
- If the addition of the given legal hold exceeds 100 total legal holds on the object, the new legal hold will not be added, a `400` error is returned.
- If an identifier is too long, it will not be added to the object and a `400` error is returned.
- If an identifier contains invalid characters, it will not be added to the object and a `400` error is returned.
- If an identifier is already in use on an object, the existing legal hold is not modified and the response indicates that the identifier was already in use with a `409` error.
- If an object does not have retention period metadata, a `400` error is returned and adding or removing a legal hold is not allowed.

To add or remove a legal hold, you must have `Manager` permissions for this bucket.

Python

```
def add_legal_hold_to_object(bucket_name, object_name, legal_hold_id):
    print("Adding legal hold {0} to object {1} in bucket {2}\n".format(legal_hold_id, object_name, bucket_name))

    cos_client.add_legal_hold(
        Bucket=bucket_name,
        Key=object_name,
        RetentionLegalHoldId=legal_hold_id
    )

    print("Legal hold {0} added to object {1} in bucket {2}!\n".format(legal_hold_id, object_name, bucket_name))

def delete_legal_hold_from_object(bucket_name, object_name, legal_hold_id):
    print("Deleting legal hold {0} from object {1} in bucket {2}\n".format(legal_hold_id, object_name, bucket_name))

    cos_client.delete_legal_hold(
        Bucket=bucket_name,
        Key=object_name,
        RetentionLegalHoldId=legal_hold_id
    )

    print("Legal hold {0} deleted from object {1} in bucket {2}!\n".format(legal_hold_id, object_name, bucket_name))
```

Extend the retention period of a protected object

The retention period of an object can only be extended. It cannot be decreased from the currently configured value.

The retention expansion value is set in one of three ways:

- additional time from the current value (`Additional-Retention-Period` or similar method)
- new extension period in seconds (`Extend-Retention-From-Current-Time` or similar method)
- new retention expiry date of the object (`New-Retention-Expiration-Date` or similar method)

The current retention period that is stored in the object metadata is either increased by the given more time or replaced with the new value, depending on the parameter that is set in the `extendRetention` request. In all cases, the extend retention parameter is checked against the current retention period and the extended parameter is only accepted if the updated retention period is greater than the current retention period.

Objects in protected buckets that are no longer under retention (retention period has expired and the object does not have any legal holds), when overwritten, will again come under retention. The new retention period can be provided as part of the object overwrite request or the default retention time of the bucket will be given to the object.

Python

```
def extend_retention_period_on_object(bucket_name, object_name, additional_seconds):
    print("Extend the retention period on {0} in bucket {1} by {2} seconds.\n".format(object_name, bucket_name, 
additional_seconds))

    cos_client.extend_object_retention(
        Bucket=bucket_ame,
        Key=object_name,
        AdditionalRetentionPeriod=additional_seconds
    )

    print("New retention period on {0} is {1}\n".format(object_name, additional_seconds))
```

List legal holds on a protected object

This operation returns:

- Object creation date
- Object retention period in seconds
- Calculated retention expiration date based on the period and creation date
- List of legal holds
- Legal hold identifier
- Timestamp when legal hold was applied

If there are no legal holds on the object, an empty `LegalHoldSet` is returned. If there is no retention period that is specified on the object, a `404` error is returned.

Python

```
def list_legal_holds_on_object(bucket_name, object_name):
    print("List all legal holds on object {0} in bucket {1}\n".format(object_name, bucket_name));

    response = cos_client.list_legal_holds(
        Bucket=bucket_name,
        Key=object_name
    )

    print("Legal holds on bucket {0}: {1}\n".format(bucket_name, response))
```

Create a hosted static website

This operation requires permissions, as only the bucket owner is typically permitted to configure a bucket to host a static website. The parameters determine the default suffix for visitors to the site as well as an optional error document.

Python

```
def putBucketWebsiteConfiguration(bucket_name):
    website_defaults = {
        'ErrorDocument': {'Key': 'error.html'},
        'IndexDocument': {'Suffix': 'index.html'},
    }
    cos_client.put_bucket_website(Bucket=bucket_name, WebsiteConfiguration=website_defaults)
    print("Website configuration set on bucket {0}\n".format(bucket_name))
```

Next Steps

For more information, the source code can be found at GitHub.

Using Node.js

The IBM Cloud® Object Storage SDK for Node.js provides modern capabilities that make the most of IBM Cloud Object Storage.

Installing the SDK

Node.js is an excellent way to build web applications, and customize your instance of Object Storage for your end users. The preferred way to install the Object Storage SDK for Node.js is to use the `npm` package manager for Node.js. Type the following command into a command line:

```
npm install ibm-cos-sdk
```

To download the SDK directly, the source code is hosted on GitHub.

More detail on individual methods and classes can be found in the API documentation for the SDK.

Getting Started

Minimum requirements

To run the SDK, you need **Node 4.x+**.

Creating a client and sourcing credentials

To connect to COS, a client is created and configured by providing credential information (API Key, Service Instance ID, and IBM Authentication Endpoint). These values can also be automatically sourced from a credentials file or from environment variables.

After generating a Service Credential, the resulting JSON document can be saved to `~/.bluemix/cos_credentials`. The SDK will automatically source credentials from this file unless other credentials are explicitly set during client creation. If the `cos_credentials` file contains HMAC keys the client authenticates with a signature, otherwise the client uses the provided API key to authenticate with a bearer token.

The `default` section heading specifies a default profile and associated values for credentials. You can create more profiles in the same shared configuration file, each with its own credential information. The following example shows a configuration file with the default profile:

```
[default]
ibm_api_key_id = <DEFAULT_IBM_API_KEY>
ibm_service_instance_id = <DEFAULT_IBM_SERVICE_INSTANCE_ID>
ibm_auth_endpoint = <DEFAULT_IBM_AUTH_ENDPOINT>
```

If migrating from AWS S3, you can also source credentials data from `~/.aws/credentials` in the format:

```
aws_access_key_id = <DEFAULT_ACCESS_KEY_ID>
aws_secret_access_key = <DEFAULT_SECRET_ACCESS_KEY>
```

If both `~/.bluemix/cos_credentials` and `~/.aws/credentials` exist, `cos_credentials` takes preference.

Code Examples

Note: In your code, you must remove the angled brackets or any other excess characters that are provided here as illustration.

Getting started with Node.js—once it's installed—usually involves configuration and invocation, like in this example from Nodejs.org. We'll follow a similar model

Initializing configuration

```
$ const IBM = require('ibm-cos-sdk');

var config = {
    endpoint: '<endpoint>',
    apiKeyId: '<api-key>',
    serviceInstanceId: '<resource-instance-id>',
    signatureVersion: 'iam',
};

var cos = new IBM.S3(config);
```

Key Values

- `<endpoint>` - public endpoint for your cloud object storage (available from the [IBM Cloud Dashboard](#)). For more information about endpoints, see [Endpoints and storage locations](#).
- `<api-key>` - API key generated when creating the service credentials (write access is required for creation and deletion examples)
- `<resource-instance-id>` - resource ID for your cloud object storage (available through [IBM Cloud CLI](#) or [IBM Cloud Dashboard](#))

Creating a bucket

A list of valid provisioning codes for `LocationConstraint` can be referenced in [the Storage Classes guide](#).

Node

```
function createBucket(bucketName) {
    console.log(`Creating new bucket: ${bucketName}`);
    return cos.createBucket({
        Bucket: bucketName,
        CreateBucketConfiguration: {
          LocationConstraint: 'us-standard'
        },
    }).promise()
    .then((() => {
        console.log(`Bucket: ${bucketName} created!`);
    }))
    .catch((e) => {
        console.error(`ERROR: ${e.code} - ${e.message}\n`);
    });
}
```

SDK References

- [`createBucket`](#)

Creating a text object

Node

```
function createTextFile(bucketName, itemName, fileText) {
    console.log(`Creating new item: ${itemName}`);
    return cos.putObject({
        Bucket: bucketName,
        Key: itemName,
        Body: fileText
    }).promise()
    .then(() => {
        console.log(`Item: ${itemName} created!`);
    })
    .catch((e) => {
        console.error(`ERROR: ${e.code} - ${e.message}\n`);
    });
}
```

SDK References

- [`putObject`](#)

List buckets

Node

```
function getBuckets() {
    console.log('Retrieving list of buckets');
    return cos.listBuckets()
    .promise()
    .then((data) => {
        if (data.Buckets != null) {
            for (var i = 0; i < data.Buckets.length; i++) {
                console.log(`Bucket Name: ${data.Buckets[i].Name}`);
            }
        }
    })
    .catch((e) => {
        console.error(`ERROR: ${e.code} - ${e.message}\n`);
    });
}
```

SDK References

- listBuckets

List items in a bucket

Node

```
function getBucketContents(bucketName) {
    console.log(`Retrieving bucket contents from: ${bucketName}`);
    return cos.listObjects(
        {Bucket: bucketName},
    ).promise()
    .then((data) => {
        if (data != null && data.Contents != null) {
            for (var i = 0; i < data.Contents.length; i++) {
                var itemKey = data.Contents[i].Key;
                var itemSize = data.Contents[i].Size;
                console.log(`Item: ${itemKey} (${itemSize} bytes).`)
            }
        }
    })
    .catch((e) => {
        console.error(`ERROR: ${e.code} - ${e.message}\n`);
    });
}
```

SDK References

- listObjects

Get file contents of particular item

Node

```
function getItem(bucketName, itemName) {
    console.log(`Retrieving item from bucket: ${bucketName}, key: ${itemName}`);
    return cos.getObject({
        Bucket: bucketName,
        Key: itemName
    }).promise()
    .then((data) => {
        if (data != null) {
            console.log('File Contents: ' + Buffer.from(data.Body).toString());
        }
    })
    .catch((e) => {
        console.error(`ERROR: ${e.code} - ${e.message}\n`);
    });
}
```

SDK References

- getObject

Delete an item from a bucket

Node

```
function deleteItem(bucketName, itemName) {
    console.log(`Deleting item: ${itemName}`);
    return cos.deleteObject({
        Bucket: bucketName,
        Key: itemName
    }).promise()
    .then(() =>{
        console.log(`Item: ${itemName} deleted!`);
    })
    .catch((e) => {
        console.error(`ERROR: ${e.code} - ${e.message}\n`);
    });
}
```

SDK References

- `deleteObject`

Delete multiple items from a bucket

Tip: The delete request can contain a maximum of 1000 keys that you want to delete. While deleting objects in batches is very useful in reducing the per-request overhead, be mindful when deleting many keys that the request may take some time to complete. Also, consider the sizes of the objects to ensure suitable performance.

Node

```
function deleteItems(bucketName) {
    var deleteRequest = {
        "Objects": [
            { "Key": "deletetest/testfile1.txt" },
            { "Key": "deletetest/testfile2.txt" },
            { "Key": "deletetest/testfile3.txt" },
            { "Key": "deletetest/testfile4.txt" },
            { "Key": "deletetest/testfile5.txt" }
        ]
    }
    return cos.deleteObjects({
        Bucket: bucketName,
        Delete: deleteRequest
    }).promise()
    .then((data) => {
        console.log(`Deleted items for ${bucketName}`);
        console.log(data.Deleted);
    })
    .catch((e) => {
        console.log(`ERROR: ${e.code} - ${e.message}\n`);
    });
}
```

SDK References

- `deleteObjects`

Delete a bucket

Node

```
function deleteBucket(bucketName) {
    console.log(`Deleting bucket: ${bucketName}`);
    return cos.deleteBucket({
        Bucket: bucketName
    }).promise()
    .then(() => {
        console.log(`Bucket: ${bucketName} deleted!`);
    })
    .catch((e) => {
        console.error(`ERROR: ${e.code} - ${e.message}\n`);
    });
}
```

SDK References

- `deleteBucket`

Execute a multi-part upload

Node

```
function multiPartUpload(bucketName, itemName, filePath) {
    var uploadID = null;

    if (!fs.existsSync(filePath)) {
        log.error(new Error(`The file \'${filePath}\' does not exist or is not accessible.`));
        return;
    }

    console.log(`Starting multi-part upload for ${itemName} to bucket: ${bucketName}`);
    return cos.createMultipartUpload({
        Bucket: bucketName,
        Key: itemName
    }).promise()
    .then((data) => {
        uploadID = data.UploadId;

        //begin the file upload
        fs.readFile(filePath, (e, fileData) => {
            //min 5MB part
            var partSize = 1024 * 1024 * 5;
            var partCount = Math.ceil(fileData.length / partSize);

            async.timesSeries(partCount, (partNum, next) => {
                var start = partNum * partSize;
                var end = Math.min(start + partSize, fileData.length);

                partNum++;

                console.log(`Uploading to ${itemName} (part ${partNum} of ${partCount})`);

                cos.uploadPart({
                    Body: fileData.slice(start, end),
                    Bucket: bucketName,
                    Key: itemName,
                    PartNumber: partNum,
                    UploadId: uploadID
                }).promise()
                .then((data) => {
                    next(e, {ETag: data.ETag, PartNumber: partNum});
                })
                .catch((e) => {
                    cancelMultiPartUpload(bucketName, itemName, uploadID);
                    console.error(`ERROR: ${e.code} - ${e.message}\n`);
                });
            }, (e, dataPacks) => {
                cos.completeMultipartUpload({
                    Bucket: bucketName,
                    Key: itemName,
                    MultipartUpload: {
                        Parts: dataPacks
                    },
                    UploadId: uploadID
                }).promise()
                .then(console.log(`Upload of all ${partCount} parts of ${itemName} successful.`))
                .catch((e) => {
                    cancelMultiPartUpload(bucketName, itemName, uploadID);
                    console.error(`ERROR: ${e.code} - ${e.message}\n`);
                });
            });
        });
    })
    .catch((e) => {
        console.error(`ERROR: ${e.code} - ${e.message}\n`);
    });
}

function cancelMultiPartUpload(bucketName, itemName, uploadID) {
    return cos.abortMultipartUpload({
        Bucket: bucketName,
        Key: itemName,
        UploadId: uploadID
    }).promise()
    .then(() => {
        console.log(`Multi-part upload aborted for ${itemName}`);
    })
    .catch((e)=>{
        console.error(`ERROR: ${e.code} - ${e.message}\n`);
    });
}
```

SDK References

- abortMultipartUpload
- completeMultipartUpload
- createMultipartUpload
- `uploadPart`

Using Key Protect

Key Protect can be added to a storage bucket to manage encryption keys. All data is encrypted in IBM COS, but Key Protect provides a service for generating, rotating, and controlling access to encryption keys using a centralized service.

Before You Begin

The following items are necessary to create a bucket with Key-Protect enabled:

- A Key Protect service provisioned
- A Root key available (either generated or imported)

Retrieving the Root Key CRN

1. Retrieve the instance ID for your Key Protect service
2. Use the Key Protect API to retrieve all your available keys
    - You can either use `curl` commands or an API REST Client such as Postman to access the Key Protect API.
3. Retrieve the CRN of the root key you will use to enabled Key Protect on the your bucket. The CRN will look similar to below:

`crn:v1:bluemix:public:kms:us-south:a/3d624cd74a0dea86ed8efe3101341742:90b6a1db-0fe1-4fe9-b91e-962c327df531:key:0bg3e33e-a866-50f2-b715-5cba2bc93234`

Creating a bucket with Key Protect enabled

Node

```
function createBucketKP(bucketName) {
    console.log(`Creating new encrypted bucket: ${bucketName}`);
    return cos.createBucket({
        Bucket: bucketName,
        CreateBucketConfiguration: {
          LocationConstraint: '<bucket-location>'
        },
        IBMSSEKPEncryptionAlgorithm: '<algorithm>',
        IBMSSEKPCustomerRootKeyCrn: '<root-key-crn>'
    }).promise()
    .then((() => {
        console.log(`Bucket: ${bucketName} created!`);
        logDone();
    }))
    .catch(logError);
}
```

Key Values

- `<bucket-location>` - Region or location for your bucket (Key Protect is only available in certain regions. Ensure your location matches the Key Protect service) A list of valid provisioning codes for `LocationConstraint` can be referenced in [the Storage Classes guide](). .
- `<algorithm>` - The encryption algorithm used for new objects added to the bucket (Default is AES256).
- `<root-key-crn>` - CRN of the Root Key that is obtained from the Key Protect service.

SDK References

- [`createBucket`]()

Using Archive Feature

Archive Tier allows users to archive stale data and reduce their storage costs. Archival policies (also known as *Lifecycle Configurations*) are created for buckets and applies to any objects added to the bucket after the policy is created.

View a bucket's lifecycle configuration

Node

```
function getLifecycleConfiguration(bucketName) {
    return cos.getBucketLifecycleConfiguration({
        Bucket: bucketName
    }).promise()
    .then((data) => {
        if (data != null) {
            console.log(`Retrieving bucket lifecycle config from: ${bucketName}`);
            console.log(JSON.stringify(data, null, 4));
        }
        else {
            console.log(`No lifecycle configuration for ${bucketName}`);
        }
    })
    .catch((e) => {
        console.log(`ERROR: ${e.code} - ${e.message}\n`);
    });
}
```

SDK References

- [getBucketLifecycleConfiguration]()

Create a lifecycle configuration

Detailed information about structuring the lifecycle configuration rules are available in the [API Reference]()

Node

```
function createLifecycleConfiguration(bucketName) {
    //
    var config = {
        Rules: [{
            Status: 'Enabled',
            ID: '<policy-id>',
            Filter: {
                Prefix: ''
            },
            Transitions: [{
                Days: <number-of-days>,
                StorageClass: 'GLACIER'
            }]
        }]
    };
    return cos.putBucketLifecycleConfiguration({
        Bucket: bucketName,
        LifecycleConfiguration: config
    }).promise()
    .then(() => {
        console.log(`Created bucket lifecycle config for: ${bucketName}`);
    })
    .catch((e) => {
        console.log(`ERROR: ${e.code} - ${e.message}\n`);
    });
}
```

Key Values

- `<policy-id>` - Name of the lifecycle policy (must be unique)
- `<number-of-days>` - Number of days to keep the restored file

SDK References

- putBucketLifecycleConfiguration

Delete a bucket's lifecycle configuration

Node

```
function deleteLifecycleConfiguration(bucketName) {
    return cos.deleteBucketLifecycle({
        Bucket: bucketName
    }).promise()
    .then(() => {
        console.log(`Deleted bucket lifecycle config from: ${bucketName}`);
    })
    .catch((e) => {
        console.log(`ERROR: ${e.code} - ${e.message}\n`);
    });
}
```

SDK References

- `deleteBucketLifecycle`

Temporarily restore an object

Detailed information about the restore request parameters are available in the API Reference

Node

```
function restoreItem(bucketName, itemName) {
    var params = {
        Bucket: bucketName,
        Key: itemName,
        RestoreRequest: {
            Days: <number-of-days>,
            GlacierJobParameters: {
                Tier: 'Bulk'
            },
        }
    };

    return cos.restoreObject(params).promise()
    .then(() => {
        console.log(`Restoring item: ${itemName} from bucket: ${bucketName}`);
    })
    .catch((e) => {
        console.log(`ERROR: ${e.code} - ${e.message}\n`);
    });
}
```

Key Values

- `<number-of-days>` - Number of days to keep the restored file

SDK References

- [`restoreObject`]

View HEAD information for an object

Node

```
function getHEADItem(bucketName, itemName) {
    return cos.headObject({
        Bucket: bucketName,
        Key: itemName
    }).promise()
    .then((data) => {
        console.log(`Retrieving HEAD for item: ${itemName} from bucket: ${bucketName}`);
        console.log(JSON.stringify(data, null, 4));
    })
    .catch((e) => {
        console.log(`ERROR: ${e.code} - ${e.message}\n`);
    });
}
```

SDK References

- [`headObject`]

Updating Metadata

There are two ways to update the metadata on an existing object:

- A `PUT` request with the new metadata and the original object contents
- Executing a `COPY` request with the new metadata specifying the original object as the copy source

Using PUT to update metadata

Note: The `PUT` request overwrites the existing contents of the object so it must first be downloaded and re-uploaded with the new metadata.

Node

```
function updateMetadataPut(bucketName, itemName, metaValue) {
    console.log(`Updating metadata for item: ${itemName}`);

    //retrieve the existing item to reload the contents
    return cos.getObject({
        Bucket: bucketName,
        Key: itemName
    }).promise()
    .then((data) => {
        //set the new metadata
        var newMetadata = {
            newkey: metaValue
        };

        return cos.putObject({
            Bucket: bucketName,
            Key: itemName,
            Body: data.Body,
            Metadata: newMetadata
        }).promise()
        .then(() => {
            console.log(`Updated metadata for item: ${itemName} from bucket: ${bucketName}`);
        })
    })
    .catch((e) => {
        console.log(`ERROR: ${e.code} - ${e.message}\n`);
    });
}
```

Using COPY to update metadata

Node

```
function updateMetadataCopy(bucketName, itemName, metaValue) {
    console.log(`Updating metadata for item: ${itemName}`);

    //set the copy source to itself
    var copySource = bucketName + '/' + itemName;

    //set the new metadata
    var newMetadata = {
        newkey: metaValue
    };

    return cos.copyObject({
        Bucket: bucketName,
        Key: itemName,
        CopySource: copySource,
        Metadata: newMetadata,
        MetadataDirective: 'REPLACE'
    }).promise()
    .then((data) => {
        console.log(`Updated metadata for item: ${itemName} from bucket: ${bucketName}`);
    })
    .catch((e) => {
        console.log(`ERROR: ${e.code} - ${e.message}\n`);
    });
}
```

Using Immutable Object Storage

Add a protection configuration to an existing bucket

Objects written to a protected bucket cannot be deleted until the protection period has expired and all legal holds on the object are removed. The bucket's default retention value is given to an object unless an object specific value is provided when the object is created. Objects in protected buckets that are no longer under retention (retention period has expired and the object does not have any legal holds), when overwritten, will again come under retention. The new retention period can be provided as part of the object overwrite request or the default retention time of the bucket will be given to the object.

The minimum and maximum supported values for the retention period settings `MinimumRetention`, `DefaultRetention`, and `MaximumRetention` are 0 days and 365243 days (1000 years) respectively.

Node

```
function addProtectionConfigurationToBucket(bucketName) {
    console.log(`Adding protection to bucket ${bucketName}`);
    return cos.putBucketProtectionConfiguration({
        Bucket: bucketName,
        ProtectionConfiguration: {
            'Status': 'Retention',
            'MinimumRetention': {'Days': 10},
            'DefaultRetention': {'Days': 100},
            'MaximumRetention': {'Days': 1000}
        }
    }).promise()
    .then(() => {
        console.log(`Protection added to bucket ${bucketName}!`);
    })
    .catch((e) => {
        console.log(`ERROR: ${e.code} - ${e.message}\n`);
    });
}
```

Check protection on a bucket

Node

```
function getProtectionConfigurationOnBucket(bucketName) {
    console.log(`Retrieve the protection on bucket ${bucketName}`);
    return cos.getBucketProtectionConfiguration({
        Bucket: bucketName
    }).promise()
    .then((data) => {
        console.log(`Configuration on bucket ${bucketName}:`);
        console.log(data);
    })
    .catch((e) => {
        console.log(`ERROR: ${e.code} - ${e.message}\n`);
    });
}
```

Upload a protected object

Objects in protected buckets that are no longer under retention (retention period has expired and the object does not have any legal holds), when overwritten, will again come under retention. The new retention period can be provided as part of the object overwrite request or the default retention time of the bucket will be given to the object.

Value	Type	Description
`Retention-Period`	Non-negative integer (seconds)	Retention period to store on the object in seconds. The object can be neither overwritten nor deleted until the amount of time specified in the retention period has elapsed. If this field and `Retention-Expiration-Date` are specified a `400` error is returned. If neither is specified the bucket's `DefaultRetention` period will be used. Zero (`0`) is a legal value assuming the bucket's minimum retention period is also `0`.
`Retention-expiration-date`	Date (ISO 8601 Format)	Date on which it will be legal to delete or modify the object. You can only specify this or the Retention-Period header. If both are specified a `400` error will be returned. If neither is specified the bucket's DefaultRetention period will be used.
`Retention-legal-hold-id`	string	A single legal hold to apply to the object. A legal hold is a Y character long string. The object cannot be overwritten or deleted until all legal holds associated with the object are removed.

Node

```
function putObjectAddLegalHold(bucketName, objectName, legalHoldId) {
    console.log(`Add legal hold ${legalHoldId} to ${objectName} in bucket ${bucketName} with a putObject operation.`);
    return cos.putObject({
        Bucket: bucketName,
        Key: objectName,
        Body: 'body',
        RetentionLegalHoldId: legalHoldId
    }).promise()
    .then((data) => {
        console.log(`Legal hold ${legalHoldId} added to object ${objectName} in bucket ${bucketName}`);
    })
    .catch((e) => {
        console.log(`ERROR: ${e.code} - ${e.message}\n`);
    });
}

function copyProtectedObject(sourceBucketName, sourceObjectName, destinationBucketName, newObjectName, ) {
    console.log(`Copy protected object ${sourceObjectName} from bucket ${sourceBucketName} to ${destinationBucketName}/${newObjectName}.`);
    return cos.copyObject({
        Bucket: destinationBucketName,
        Key: newObjectName,
        CopySource: sourceBucketName + '/' + sourceObjectName,
        RetentionDirective: 'Copy'
    }).promise()
    .then((data) => {
        console.log(`Protected object copied from ${sourceBucketName}/${sourceObjectName} to ${destinationBucketName}/${newObjectName}`);
    })
    .catch((e) => {
        console.log(`ERROR: ${e.code} - ${e.message}\n`);
    });
}
```

Add or remove a legal hold to or from a protected object

The object can support 100 legal holds:

- A legal hold identifier is a string of maximum length 64 characters and a minimum length of 1 character. Valid characters are letters, numbers, `!`, `_`, `.`, `*`, `(`, `)`, `-` and `'`.
- If the addition of the given legal hold exceeds 100 total legal holds on the object, the new legal hold will not be added, a `400` error will be returned.
- If an identifier is too long it will not be added to the object and a `400` error is returned.
- If an identifier contains invalid characters, it will not be added to the object and a `400` error is returned.
- If an identifier is already in use on an object, the existing legal hold is not modified and the response indicates the identifier was already in use with a `409` error.
- If an object does not have retention period metadata, a `400` error is returned and adding or removing a legal hold is not allowed.

The user making adding or removing a legal hold must have `Manager` permissions for this bucket.

Node

```
function addLegalHoldToObject(bucketName, objectName, legalHoldId) {
    console.log(`Adding legal hold ${legalHoldId} to object ${objectName} in bucket ${bucketName}`);
    return cos.client.addLegalHold({
        Bucket: bucketName,
        Key: objectId,
        RetentionLegalHoldId: legalHoldId
    }).promise()
    .then(() => {
        console.log(`Legal hold ${legalHoldId} added to object ${objectName} in bucket ${bucketName}!`);
    })
    .catch((e) => {
        console.log(`ERROR: ${e.code} - ${e.message}\n`);
    });
}

function deleteLegalHoldFromObject(bucketName, objectName, legalHoldId) {
    console.log(`Deleting legal hold ${legalHoldId} from object ${objectName} in bucket ${bucketName}`);
    return cos.client.deleteLegalHold({
        Bucket: bucketName,
        Key: objectId,
        RetentionLegalHoldId: legalHoldId
    }).promise()
    .then(() => {
        console.log(`Legal hold ${legalHoldId} deleted from object ${objectName} in bucket ${bucketName}!`);
    })
    .catch((e) => {
        console.log(`ERROR: ${e.code} - ${e.message}\n`);
    });
}
```

Extend the retention period of a protected object

The retention period of an object can only be extended. It cannot be decreased from the currently configured value.

The retention expansion value is set in one of three ways:

- additional time from the current value (`Additional-Retention-Period` or similar method)
- new extension period in seconds (`Extend-Retention-From-Current-Time` or similar method)
- new retention expiry date of the object (`New-Retention-Expiration-Date` or similar method)

The current retention period stored in the object metadata is either increased by the given additional time or replaced with the new value, depending on the parameter that is set in the `extendRetention` request. In all cases, the extend retention parameter is checked against the current retention period and the extended parameter is only accepted if the updated retention period is greater than the current retention period.

Objects in protected buckets that are no longer under retention (retention period has expired and the object does not have any legal holds), when overwritten, will again come under retention. The new retention period can be provided as part of the object overwrite request or the default retention time of the bucket will be given to the object.

Node

```
function extendRetentionPeriodOnObject(bucketName, objectName, additionalSeconds) {
    console.log(`Extend the retention period on ${objectName} in bucket ${bucketName} by ${additionalSeconds} seconds.`);
    return cos.extendObjectRetention({
        Bucket: bucketName,
        Key: objectName,
        AdditionalRetentionPeriod: additionalSeconds
    }).promise()
    .then((data) => {
        console.log(`New retention period on ${objectName} is ${data.RetentionPeriod}`);
    })
    .catch((e) => {
        console.log(`ERROR: ${e.code} - ${e.message}\n`);
    });
}
```

List legal holds on a protected object

This operation returns:

- Object creation date
- Object retention period in seconds
- Calculated retention expiration date based on the period and creation date
- List of legal holds
- Legal hold identifier
- Timestamp when legal hold was applied

If there are no legal holds on the object, an empty `LegalHoldSet` is returned. If there is no retention period specified on the object, a `404` error is returned.

Node

```
function listLegalHoldsOnObject(bucketName, objectName) {
    console.log(`List all legal holds on object ${objectName} in bucket ${bucketName}`);
    return cos.listLegalHolds({
        Bucket: bucketName,
        Key: objectId
    }).promise()
    .then((data) => {
        console.log(`Legal holds on bucket ${bucketName}: ${data}`);
    })
    .catch((e) => {
        console.log(`ERROR: ${e.code} - ${e.message}\n`);
    });
}
```

Create a hosted static website

This operation requires permissions, as only the bucket owner is typically permitted to configure a bucket to host a static website. The parameters determine the default suffix for visitors to the site as well as an optional error document.

Node

```
var websiteParams = {
  Bucket: "bucketName",
  WebsiteConfiguration: {
    ErrorDocument: {
      Key: "error.html"
    },
    IndexDocument: {
      Suffix: "index.html"
    }
  }
};
function putBucketWebsiteConfiguration(websiteParams) {
  return cos.putBucketWebsite({websiteParams}).promise()
    .then((data) => {
       console.log(`Website configured for ${bucketName}`);
    })
    .catch((e) => {
        console.log(`ERROR: ${e.code} - ${e.message}\n`);
    });
}
```

Next Steps

More detail on individual methods and classes can be found in [the SDK's API documentation](). Check out the source code on [GitHub]().

Using Go

The IBM Cloud® Object Storage SDK for Go provides features to make the most of IBM Cloud Object Storage.

The IBM Cloud Object Storage SDK for Go is comprehensive, with many features and capabilities that exceed the scope and space of this guide. For detailed class and method documentation [see the Go API documentation](). Source code can be found in the [GitHub repository]().

Getting the SDK

Use `go get` to retrieve the SDK to add it to your GOPATH workspace, or project's Go module dependencies. The SDK requires a minimum version of Go 1.10 and maximum version of Go 1.12. Future versions of Go will be supported once our quality control process has been completed.

```
$ go get github.com/IBM/ibm-cos-sdk-go
```

To update the SDK use `go get -u` to retrieve the latest version of the SDK.

```
$ go get -u github.com/IBM/ibm-cos-sdk-go
```

Import packages

After you have installed the SDK, you will need to import the packages that you require into your Go applications to use the SDK, as shown in the following example:

```
import (
    "github.com/IBM/ibm-cos-sdk-go/aws/credentials/ibmiam"
    "github.com/IBM/ibm-cos-sdk-go/aws"
    "github.com/IBM/ibm-cos-sdk-go/aws/session"
    "github.com/IBM/ibm-cos-sdk-go/service/s3"
)
```

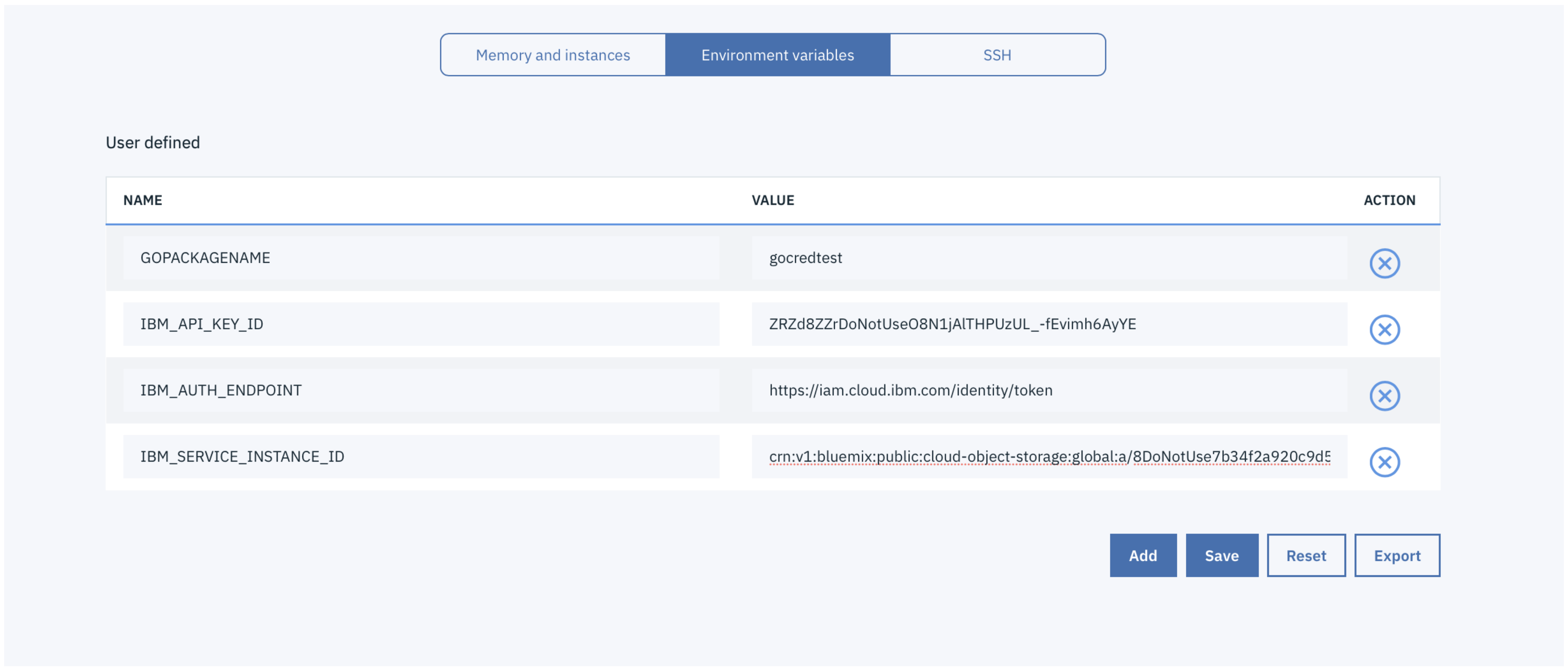
Creating a client and sourcing Service credentials

To connect to IBM Cloud Object Storage, a client is created and configured by providing credential information (API key and service instance ID). These values can also be automatically sourced from a credentials file or from environment variables.

The credentials can be found by creating a [Service Credential](), or through the CLI.

Figure 1 shows an example of how to define environment variables in an application runtime at the IBM Cloud Object Storage portal. The required variables are `IBM_API_KEY_ID` containing your Service Credential `apikey`, `IBM_SERVICE_INSTANCE_ID` holding the `resource_instance_id` also from your Service Credential, and an `IBM_AUTH_ENDPOINT` with a value appropriate to your account, like `https://iam.cloud.ibm.com/identity/token`. If using environment variables to define your application credentials, use `WithCredentials(ibmiam.NewEnvCredentials(aws.NewConfig())).`, replacing the similar method used in the configuration example.

Environment Variables



User defined

NAME	VALUE	ACTION
GOPACKAGENAME	gocredtest	⊗
IBM_API_KEY_ID	ZRZd8ZZrDoNotUseO8N1jAlTHPUzUL_-fEvimh6AyYE	⊗
IBM_AUTH_ENDPOINT	https://iam.cloud.ibm.com/identity/token	⊗
IBM_SERVICE_INSTANCE_ID	crn:v1:bluemix:public:cloud-object-storage:global:a/8DoNotUse7b34f2a920c9d5	⊗

If migrating from AWS S3, you can also source credentials data from `~/.aws/credentials` in the format:

```
$ [default]
aws_access_key_id = {ACCESS_KEY}
aws_secret_access_key = {SECRET_ACCESS_KEY}
```

If both `~/.bluemix/cos_credentials` and `~/.aws/credentials` exist, `cos_credentials` takes preference.

Initializing configuration

```

// Constants for IBM COS values
const (
    apiKey            = "<API_KEY>"  // eg "0viPHOY7LbLNa9eLftrtHPpTjoGv6hbLD1QalRXikliJ"
    serviceInstanceID = "<RESOURCE_INSTANCE_ID>" // eg "crn:v1:bluemix:public:cloud-object-
storage:global:a/3bf0d9003xxxxxxxxxx1c3e97696b71c:d6f04d83-6c4f-4a62-a165-696756d63903::"
    authEndpoint      = "https://iam.cloud.ibm.com/identity/token"
    serviceEndpoint   = "<SERVICE_ENDPOINT>" // eg "https://s3.us.cloud-object-storage.appdomain.cloud"
    bucketLocation    = "<LOCATION>" // eg "us"
)

// Create config

conf := aws.NewConfig().
    WithRegion("us-standard").
    WithEndpoint(serviceEndpoint).
    WithCredentials(ibmiam.NewStaticCredentials(aws.NewConfig(), authEndpoint, apiKey, serviceInstanceID)).
    WithS3ForcePathStyle(true)

```

Creating a client and sourcing Trusted Profile credentials

A client can be created by provding service credentials or trusted profile credentials. This section provides information to create a client using trusted profile credentials.

To connect to IBM Cloud Object Storage, a client is created and can also be configured by providing trusted profile credential information (Trusted Profile Id and CR Token file path). These values can also be automatically sourced from environment variables.

To create a Trusted Profile, establishing trust with compute resources based on specific attributes, and to define a policy to assign access to resources, see [Managing access for apps in compute resources](#).

To learn more about establishing trust with a Kubernetes cluster, see [Using Trusted Profiles in your Kubernetes and OpenShift Clusters](#)

Note: GO SDK supports authentication using trusted profile only in kubernetes and openshift clusters.

Trusted profile credentials can be set as environment variables during application runtime. The required variables are `TRUSTED_PROFILE_ID` containing your Trusted profile Id `trusted profile id`, `CR_TOKEN_FILE_PATH` holding the `service account token file path`, `IBM_SERVICE_INSTANCE_ID` holding the `resource_instance_id` from your Service Credential, and an `IBM_AUTH_ENDPOINT` with a value appropriate to your account, like `https://iam.cloud.ibm.com/identity/token` . If using environment variables to define your application credentials, use `WithCredentials(ibmiam.NewEnvCredentials(aws.NewConfig())).` , replacing the similar method used in the configuration example.

Initializing configuration

```
// Constants for IBM COS values
const (
    trustedProfileID  = "<TRUSTED_PROFILE_ID>"  // eg "Profile-5790481a-8fc5-46a4-bae3-d0e64ff6e0ad"
    crTokenFilePath   = "<SERVICE_ACCOUNT_TOKEN_FILE_PATH>" // "/var/run/secrets/tokens/service-account-token"
    serviceInstanceID = "<RESOURCE_INSTANCE_ID>" // "crn:v1:bluemix:public:cloud-object-
storage:global:a/<CREDENTIAL_ID_AS_GENERATED>:<SERVICE_ID_AS_GENERATED>::"
    authEndpoint      = "https://iam.cloud.ibm.com/identity/token"
    serviceEndpoint   = "<SERVICE_ENDPOINT>" // eg "https://s3.us.cloud-object-storage.appdomain.cloud"
    bucketLocation    = "<LOCATION>" // eg "us-standard"
)

// Create config
conf := aws.NewConfig().
    WithRegion(bucketLocation).
    WithEndpoint(serviceEndpoint).
    WithCredentials(ibmiam.NewTrustedProfileCredentialsCR(aws.NewConfig(), authEndpoint, trustedProfileID, crtokenFilePath,
serviceInstanceID)).
    WithS3ForcePathStyle(true)
```

Note: Both API-Key and Trusted-Profile-Id can't be set as environmental variables. Only one of them should be set, otherwise GO sdk throws an error.

For more information about endpoints, see [Endpoints and storage locations](#).

Code Examples

Creating a new bucket

A list of valid provisioning codes for `LocationConstraint` can be referenced in [the Storage Classes guide](#). Please note that the sample uses the appropriate location constraint for the Cold Vault storage based on the sample configuration. Your locations and configuration may vary.

```
func main() {

    // Create client
    sess := session.Must(session.NewSession())
    client := s3.New(sess, conf)

    // Bucket Names
    newBucket := "<NEW_BUCKET_NAME>"
    newColdBucket := "<NEW_COLD_BUCKET_NAME>"
    input := &s3.CreateBucketInput{
        Bucket: aws.String(newBucket),
    }
    client.CreateBucket(input)

    input2 := &s3.CreateBucketInput{
        Bucket: aws.String(newColdBucket),
        CreateBucketConfiguration: &s3.CreateBucketConfiguration{
            LocationConstraint: aws.String("us-cold"),
        },
    }
    client.CreateBucket(input2)

    d, _ := client.ListBuckets(&s3.ListBucketsInput{})
    fmt.Println(d)
}
```

List available buckets

```
func main() {
    // Create client
    sess := session.Must(session.NewSession())
    client := s3.New(sess, conf)

    // Call Function
    d, _ := client.ListBuckets(&s3.ListBucketsInput{})
    fmt.Println(d)
}
```

Upload an object to a bucket

```
func main() {

    // Create client
    sess := session.Must(session.NewSession())
    client := s3.New(sess, conf)

    // Variables and random content to sample, replace when appropriate
    bucketName := "<BUCKET_NAME>"
    key := "<OBJECT_KEY>"
    content := bytes.NewReader([]byte("<CONTENT>"))

    input := s3.PutObjectInput{
        Bucket:        aws.String(bucketName),
        Key:           aws.String(key),
        Body:          content,
    }

    // Call Function to upload (Put) an object
    result, _ := client.PutObject(&input)
    fmt.Println(result)
}
```

List items in a bucket (List Objects V2)

```
func main() {

    // Create client
    sess := session.Must(session.NewSession())
    client := s3.New(sess, conf)

    // Bucket Name
    Bucket := "<BUCKET_NAME>"

    // Call Function
    Input := &s3.ListObjectsV2Input{
            Bucket: aws.String(Bucket),
        }

    l, e := client.ListObjectsV2(Input)
    fmt.Println(l)
    fmt.Println(e) // prints "<nil>"
}

// The response should be formatted like the following example:
//{
//  Contents: [{
//   ETag: "\"dbxxxxx53xxx7d06378204e3xxxxxx9f\"",
//   Key: "file1.json",
//   LastModified: 2019-10-15 22:22:52.62 +0000 UTC,
//   Size: 1045,
//   StorageClass: "STANDARD"
//    },{
//   ETag: "\"6e1xxxxx63xxxdefb440f72axxxxxxc2\"",
//   Key: "file2.json",
//   LastModified: 2019-10-15 23:08:10.074 +0000 UTC,
//   Size: 1045,
//   StorageClass: "STANDARD"
//    }],
//  Delimiter: "",
//  IsTruncated: false,
//  KeyCount: 2,
//  MaxKeys: 1000,
//  Name: "<BUCKET_NAME>",
//  Prefix: ""
//}
```

Get an object's contents

```
func main() {
    // Create client
    sess := session.Must(session.NewSession())
    client := s3.New(sess, conf)

    // Variables
    bucketName := "<NEW_BUCKET_NAME>"
    key := "<OBJECT_KEY>"

    // users will need to create bucket, key (flat string name)
    Input := s3.GetObjectInput{
        Bucket: aws.String(bucketName),
        Key:    aws.String(key),
    }

    // Call Function
    res, _ := client.GetObject(&Input)

    body, _ := ioutil.ReadAll(res.Body)
    fmt.Println(body)
}
```

Delete an object from a bucket

```
func main() {
    // Create client
    sess := session.Must(session.NewSession())
    client := s3.New(sess, conf)
    // Bucket Name
    bucket := "<BUCKET_NAME>"
    input := &s3.DeleteObjectInput{
        Bucket: aws.String(bucket),
        Key:    aws.String("<OBJECT_KEY>"),
    }
    d, _ := client.DeleteObject(input)
    fmt.Println(d)
}
```

Delete multiple objects from a bucket

```
func main() {

    // Create client
    sess := session.Must(session.NewSession())
    client := s3.New(sess, conf)

    // Bucket Name
    bucket := "<BUCKET_NAME>"

    input := &s3.DeleteObjectsInput{
        Bucket: aws.String(bucket),
        Delete: &s3.Delete{
            Objects: []*s3.ObjectIdentifier{
                {
                    Key: aws.String("<OBJECT_KEY1>"),
                },
                {
                    Key: aws.String("<OBJECT_KEY2>"),
                },
                {
                    Key: aws.String("<OBJECT_KEY3>"),
                },
            },
            Quiet: aws.Bool(false),
        },
    }

    d, _ := client.DeleteObjects(input)
    fmt.Println(d)
}
```

Delete a bucket

```
func main() {

    // Bucket Name
    bucket := "<BUCKET_NAME>"

    // Create client
    sess := session.Must(session.NewSession())
    client := s3.New(sess, conf)

    input := &s3.DeleteBucketInput{
        Bucket: aws.String(bucket),
    }
    d, _ := client.DeleteBucket(input)
    fmt.Println(d)
}
```

Run a manual multi-part upload

```
func main() {

    // Variables
    bucket := "<BUCKET_NAME>"
    key := "<OBJECT_KEY>"
    content := bytes.NewReader([]byte("<CONTENT>"))

    input := s3.CreateMultipartUploadInput{
        Bucket: aws.String(bucket),
        Key:    aws.String(key),
    }

    // Create client
    sess := session.Must(session.NewSession())
    client := s3.New(sess, conf)
    upload, _ := client.CreateMultipartUpload(&input)

    uploadPartInput := s3.UploadPartInput{
        Bucket:     aws.String(bucket),
        Key:        aws.String(key),
        PartNumber: aws.Int64(int64(1)),
        UploadId:   upload.UploadId,
        Body:          content,
    }

    var completedParts []*s3.CompletedPart
    completedPart, _ := client.UploadPart(&uploadPartInput)

    completedParts = append(completedParts, &s3.CompletedPart{
        ETag:       completedPart.ETag,
        PartNumber: aws.Int64(int64(1)),
    })

    completeMPUInput := s3.CompleteMultipartUploadInput{
        Bucket: aws.String(bucket),
        Key:    aws.String(key),
        MultipartUpload: &s3.CompletedMultipartUpload{
            Parts: completedParts,
        },
        UploadId: upload.UploadId,
    }

    d, _ := client.CompleteMultipartUpload(&completeMPUInput)
    fmt.Println(d)
}
```

Using Key Protect

Key Protect can be added to a storage bucket to manage encryption keys. All data is encrypted in IBM COS, but Key Protect provides a service for generating, rotating, and controlling access to encryption keys by using a centralized service.

Before You Begin

The following items are necessary to create a bucket with Key-Protect enabled:

- A Key Protect service provisioned
- A Root key available (either generated or imported)

Retrieving the Root Key CRN

1. Retrieve the instance ID for your Key Protect service
2. Use the Key Protect API to retrieve all your available keys
    - You can either use `curl` commands or an API REST Client such as Postman to access the Key Protect API.
3. Retrieve the CRN of the root key you use to enabled Key Protect on your bucket. The CRN looks similar to below:

`crn:v1:bluemix:public:kms:us-south:a/3d624cd74a0dea86ed8efe3101341742:90b6a1db-0fe1-4fe9-b91e-962c327df531:key:0bg3e33e-a866-50f2-b715-5cba2bc93234`

Creating a bucket with Key Protect enabled

```
func main() {

    // Create client
    sess := session.Must(session.NewSession())
    client := s3.New(sess, conf)

    // Bucket Names
    newBucket := "<NEW_BUCKET_NAME>"
    fmt.Println("Creating new encrypted bucket:", newBucket)

    input := &s3.CreateBucketInput{
        Bucket: aws.String(newBucket),
        IBMSSEKPCustomerRootKeyCrn: aws.String("<ROOT-KEY-CRN>"),
        IBMSSEKPEncryptionAlgorithm:aws.String("<ALGORITHM>"),
    }
    client.CreateBucket(input)

    // List Buckets
    d, _ := client.ListBuckets(&s3.ListBucketsInput{})
    fmt.Println(d)
}
```

Key Values

- `<NEW_BUCKET_NAME>` - The name of the new bucket.
- `<ROOT-KEY-CRN>` - CRN of the Root Key that is obtained from the Key Protect service.
- `<ALGORITHM>` - The encryption algorithm that is used for new objects added to the bucket (Default is AES256).

Use the transfer manager

```
func main() {

    // Variables
    bucket := "<BUCKET_NAME>"
    key := "<OBJECT_KEY>"

    // Create client
    sess := session.Must(session.NewSession())
    client := s3.New(sess, conf)

    // Create an uploader with S3 client and custom options
    uploader := s3manager.NewUploaderWithClient(client, func(u *s3manager.Uploader) {
        u.PartSize = 5 * 1024 * 1024 // 64MB per part
    })

    // make a buffer of 5MB
    buffer := make([]byte, 15*1024*1024, 15*1024*1024)
    random := rand.New(rand.NewSource(time.Now().Unix()))
    random.Read(buffer)

    input := &s3manager.UploadInput{
        Bucket: aws.String(bucket),
        Key:    aws.String(key),
        Body:   io.ReadSeeker(bytes.NewReader(buffer)),
    }

    // Perform an upload.
    d, _ := uploader.Upload(input)
    fmt.Println(d)
    // Perform upload with options different than the those in the Uploader.
    f, _ := uploader.Upload(input, func(u *s3manager.Uploader) {
        u.PartSize = 10 * 1024 * 1024 // 10MB part size
        u.LeavePartsOnError = true    // Don't delete the parts if the upload fails.
    })
    fmt.Println(f)
}
```

Getting an extended listing

```
func main() {
// Create client
        sess := session.Must(session.NewSession())
        client := s3.New(sess, conf)

        input := new(s3.ListBucketsExtendedInput).SetMaxKeys(<MAX_KEYS>).SetMarker("<MARKER>").SetPrefix("<PREFIX>")
        output, _ := client.ListBucketsExtended(input)

        jsonBytes, _ := json.MarshalIndent(output, " ", " ")
        fmt.Println(string(jsonBytes))
}
```

Key Values

- `<MAX_KEYS>` - Maximum number of buckets to retrieve in the request.
- `<MARKER>` - The bucket name to start the listing (Skip until this bucket).
- `<PREFIX` - Only include buckets whose name start with this prefix.

Getting an extended listing with pagination

```
func main() {

 // Create client
 sess := session.Must(session.NewSession())
 client := s3.New(sess, conf)

    i := 0
    input := new(s3.ListBucketsExtendedInput).SetMaxKeys(<MAX_KEYS>).SetMarker("<MARKER>").SetPrefix("<PREFIX>")
 output, _ := client.ListBucketsExtended(input)
 for _, bucket := range output.Buckets {
  fmt.Println(i, "\t\t", *bucket.Name, "\t\t", *bucket.LocationConstraint, "\t\t", *bucket.CreationDate)
 }

}
```

Key Values

- `<MAX_KEYS>` - Maximum number of buckets to retrieve in the request.
- `<MARKER>` - The bucket name to start the listing (Skip until this bucket).
- `<PREFIX` - Only include buckets whose name start with this prefix.

Archive Tier Support

You can automatically archive objects after a specified length of time or after a specified date. Once archived, a temporary copy of an object can be restored for access as needed. Please note the time required to restore the temporary copy of the object(s) may take up to 12 hours.

To use the example provided, provide your own configuration—including replacing `<apikey>` and other bracketed `<...>` information, while keeping in mind that using environment variables are more secure, and you should not put credentials in code that will be versioned.

An archive policy is set at the bucket level by calling the `PutBucketLifecycleConfiguration` method on a client instance. A newly added or modified archive policy applies to new objects uploaded and does not affect existing objects.

```
func main() {

 // Create Client
 sess := session.Must(session.NewSession())
 client := s3.New(sess, conf)

 // PUT BUCKET LIFECYCLE CONFIGURATION
 // Replace <BUCKET_NAME> with the name of the bucket
 lInput := &s3.PutBucketLifecycleConfigurationInput{
  Bucket: aws.String("<BUCKET_NAME>"),
  LifecycleConfiguration: &s3.LifecycleConfiguration{
   Rules: []*s3.LifecycleRule{
    {
     Status: aws.String("Enabled"),
     Filter: &s3.LifecycleRuleFilter{},
     ID:     aws.String("id3"),
     Transitions: []*s3.Transition{
      {
       Days:         aws.Int64(5),
       StorageClass: aws.String("Glacier"),
      },
     },
    },
   },
  },
 }
 l, e := client.PutBucketLifecycleConfiguration(lInput)
 fmt.Println(l) // should print an empty bracket
 fmt.Println(e) // should print <nil>

 // GET BUCKET LIFECYCLE CONFIGURATION
 gInput := &s3.GetBucketLifecycleConfigurationInput{
  Bucket: aws.String("<bucketname>"),
 }
 g, e := client.GetBucketLifecycleConfiguration(gInput)
 fmt.Println(g)
 fmt.Println(e) // see response for results

    // RESTORE OBJECT
    // Replace <OBJECT_KEY> with the appropriate key
    rInput := &s3.RestoreObjectInput{
        Bucket: aws.String("<BUCKET_NAME>"),
        Key:    aws.String("<OBJECT_KEY>"),
        RestoreRequest: &s3.RestoreRequest{
            Days: aws.Int64(100),
            GlacierJobParameters: &s3.GlacierJobParameters{
                Tier: aws.String("Bulk"),
            },
        },
    }
    r, e := client.RestoreObject(rInput)
    fmt.Println(r)
    fmt.Println(e)

}
```

The typical response is exemplified here.

```
{
  Rules: [{
      Filter: {

      },
      ID: "id3",
      Status: "Enabled",
      Transitions: [{
          Days: 5,
          StorageClass: "GLACIER"
        }]
    }]
}
```

Immutable Object Storage

Users can configure buckets with an Immutable Object Storage policy to prevent objects from being modified or deleted for a defined period of time. The retention period can be specified on a per-object basis, or objects can inherit a default retention period set on the bucket. It is also possible to set open-ended and permanent retention periods. Immutable Object Storage meets the rules set forth by the SEC governing record retention, and IBM Cloud administrators are unable to bypass these restrictions.

Note: Immutable Object Storage does not support Aspera transfers via the SDK to upload objects or directories at this stage.

```
func main() {

 // Create Client
 sess := session.Must(session.NewSession())
 client := s3.New(sess, conf)

 // Create a bucket
 input := &s3.CreateBucketInput{
  Bucket: aws.String("<BUCKET_NAME>"),
 }
 d, e := client.CreateBucket(input)
 fmt.Println(d) // should print an empty bracket
 fmt.Println(e) // should print <nil>

 // PUT BUCKET PROTECTION CONFIGURATION
 pInput := &s3.PutBucketProtectionConfigurationInput{
  Bucket: aws.String("<BUCKET_NAME>"),
  ProtectionConfiguration: &s3.ProtectionConfiguration{
   DefaultRetention: &s3.BucketProtectionDefaultRetention{
    Days: aws.Int64(100),
   },
   MaximumRetention: &s3.BucketProtectionMaximumRetention{
    Days: aws.Int64(1000),
   },
   MinimumRetention: &s3.BucketProtectionMinimumRetention{
    Days: aws.Int64(10),
   },
   Status: aws.String("Retention"),
  },
 }
 p, e := client.PutBucketProtectionConfiguration(pInput)
 fmt.Println(p)
 fmt.Println(e) // see response for results

 // GET BUCKET PROTECTION CONFIGURATION
 gInput := &s3.GetBucketProtectionConfigurationInput{
  Bucket: aws.String("<BUCKET_NAME>"),
 }
 g, e := client.GetBucketProtectionConfiguration(gInput)
 fmt.Println(g)
 fmt.Println(e)
}
```

The typical response is exemplified here.

```
 {
   ProtectionConfiguration: {
     DefaultRetention: {
       Days: 100
     },
     MaximumRetention: {
       Days: 1000
     },
     MinimumRetention: {
       Days: 10
     },
     Status: "COMPLIANCE"
   }
 }
```

Create a hosted static website

This operation requires permissions, as only the bucket owner is typically permitted to configure a bucket to host a static website. The parameters determine the default suffix for visitors to the site as well as an optional error document included here to complete the example.

```
func  main() {

 // Create Client
 sess := session.Must(session.NewSession())
 client := s3.New(sess, conf)

 // Create a bucket
 input := &s3.CreateBucketInput{
  Bucket: aws.String("<BUCKET_NAME>"),
 }
 d, e := client.CreateBucket(input)
 fmt.Println(d) // should print an empty bracket
 fmt.Println(e) // should print <nil>

 // PUT BUCKET WEBSITE
 pInput := s3.PutBucketWebsiteInput{
        Bucket: input,
        WebsiteConfiguration: &s3.WebsiteConfiguration{
            IndexDocument: &s3.IndexDocument{
                Suffix: aws.String("index.html"),
            },
        },
    }

    pInput.WebsiteConfiguration.ErrorDocument = &s3.ErrorDocument{
        Key: aws.String("error.html"),
    }

    p, e := client.PutBucketWebsite(&params)
 fmt.Println(p)
 fmt.Println(e) // see response for results

}
```

Next Steps

If you haven't already, please see the detailed class and method documentation available at the Go API documentation.

About Terraform

Terraform is an open source project that lets you specify your cloud infrastructure resources and services by using the high-level scripting HashiCorp Configuration Language (HCL). With HCL, you have one common language to declare the cloud resources that you want and the state that you want your resources to be in

Terraform on IBM Cloud enables predictable and consistent provisioning of IBM Cloud platform, classic infrastructure, and VPC infrastructure resources so that you can rapidly build complex, multi-tier cloud environments, and enable Infrastructure as Code (IaC).

How does Terraform on IBM Cloud work

Let's say you want to spin up multiple copies of your cloud environment that uses a cluster of virtual servers, a load balancer, and a database server on IBM Cloud. You could learn how to create each resource, review the API or the commands that you need, and write a bash script to spin up these components. But it's easier, faster, and more orderly to use one language to declare all your requirements, document them in a configuration file, and let Terraform on IBM Cloud do it all for you.

How to provision Terraform on IBM Cloud and manage cloud services?

To use Terraform on IBM Cloud, you must create a Terraform configuration file that describes the IBM Cloud resources that you need and how you want to configure them. Based on your configuration, Terraform creates an execution plan and describes the actions that need to be executed to get to the required state. You can review the execution plan, change it, or simply execute the plan. When you change your configuration, Terraform on IBM Cloud can determine what changed and create incremental execution plans that you can apply to your existing IBM Cloud resources.

The following steps show how Terraform on IBM Cloud provisions your services in IBM Cloud.

1. You declare the IBM Cloud resources that you want in a Terraform configuration file by using HashiCorp Configuration Language (HCL). Store this configuration file in a source code repository that is version-controlled and that allows teams to collaborate, such as GitHub or GitLab.
2. Configure the IBM Cloud Provider plug-in.
3. Create a Terraform execution plan that summarizes all the actions that Terraform needs to run to create, update, or delete the IBM Cloud resources in your Terraform template.
4. Apply the Terraform configuration file in IBM Cloud.

Using cloudyr for data science

When you use the R programming language for your projects, get the most out of the features for supporting data science from IBM Cloud® Object Storage by using cloudyr.

This tutorial shows you how to integrate data from the IBM Cloud® Platform within your `R` project. Your project will use IBM Cloud Object Storage for storage with S3-compatible connectivity in your project.

Before you begin

We need to make sure that we have the prerequisites before continuing:

```
$ - IBM Cloud Platform account
- An instance of IBM Cloud Object Storage
- `R` installed and configured
- S3-compatible authentication configuration
```

Create HMAC credentials

Before we begin, we might need to create a set of HMAC credentials as part of a Service Credential by using the configuration parameter `{"HMAC":true}` when we create credentials. For example, use the IBM Cloud Object Storage CLI as shown here.

```
$ ibmcloud resource service-key-create <key-name-without-spaces> Writer --instance-name "<instance name--use quotes if your
instance name has spaces>" --parameters '{"HMAC":true}'
```

To store the results of the generated key, append the text, `> cos_credentials` to the end of the command in the example. For the purposes of this tutorial you need to find the `cos_hmac_keys` heading with child keys, `access_key_id`, and `secret_access_key`.

```
    cos_hmac_keys:
        access_key_id:      7xxxxxxxxxxxxxxa6440da12685eee02
        secret_access_key:  8xxxx8ed850cddbece407xxxxxxxxxxxxxx43r2d2586
```

While it is best practices to set credentials in environment variables, you can also set your credentials inside your local copy of your `R` script itself. Environment variables can alternatively be set before you start `R` using an `Renviron.site` or `.Renviron` file, used to set environment variables in `R` during startup.

Note: You will need to set the actual values for the `access_key_id` and `secret_access_key` in your code along with the IBM Cloud Object Storage endpoint for your instance.

Add credentials to your `R` project

As it is beyond the scope of this tutorial, it is assumed you already installed the `R` language and suite of applications. Before you add any libraries or code to your project, ensure that you have credentials available to connect to IBM Cloud Object Storage. You will need the appropriate region for your bucket and endpoint.

```
Sys.setenv("AWS_ACCESS_KEY_ID" = "access_key_id",
           "AWS_SECRET_ACCESS_KEY" = "secret_access_key",
           "AWS_S3_ENDPOINT" = "myendpoint",
           "AWS_DEFAULT_REGION" = "")
```

Add libraries to your `R` project

We used a `cloudyr` S3-compatible client to test our credentials resulting in listing your buckets. To get additional packages, we use the source code collective known as CRAN that operates through a series of mirrors.

For this example, we use aws.s3 as shown in the example and added to the code to set or access your credentials.

```
library("aws.s3")
bucketlist()
```

Use library methods in your `R` project

You can learn a lot from working with sample packages. For example, the package for Cosmic Microwave Background Data Analysis presents a conundrum. The executable of the project for local compiling are small enough to work on one's personal machine, but working with the source data would be constrained due to the size of the data.

Tip: When using version `0.3.21` of the package, it is necessary to add `region=""` in a request to connect to COS.

In addition to PUT, HEAD, and other compatible API commands, we can GET objects as shown with the S3-compatible client we included earlier.

```
# return object using 'S3 URI' syntax, with progress bar
get_object("s3://mybucketname-only/example.csv", show_progress = TRUE)
```

Add data to your `R` project

As you can guess, the library discussed earlier has a `save_object()` method that can write directly to your bucket. While there are many ways to load data, we can use cloudSimplifieR to work with an open data set.

```
library(cloudSimplifieR)
d <- as.data.frame(csvToDataframe("s3://mybucket/example.csv"))
plot(d)
```

Next steps

In addition to creating your own projects, you can also use R Studio to analyze data.

Use the command line

IBM Cloud Object Storage CLI

The IBM Cloud® Object Storage plug-in extends the IBM Cloud command line interface (CLI) with an API wrapper for working with object storage resources.

Installation and configuration

The plugin is compatible with Linux (x86_64, arm64, ppc64le, s390x), Windows® (x64), and macOS® (amd64, arm64) platforms that run on 64-bit processors.

Install the plug-in by using the `plugin install` command.

```
$ ibmcloud plugin install cloud-object-storage
```

Once the plug-in is installed, you can configure the plug-in by using the `ibmcloud cos config` command. This can be used to populate the plug-in with your credentials, default download location, choosing your authentication, and so on.

Important: For optimal performance, ensure that tracing is disabled by setting the `IBMCLOUD_TRACE` environment variable to `false`.

The program also offers the ability for you to set the default local directory for downloaded files, and to set a default region. To set the default download location, type `ibmcloud cos config ddl` and input into the program a valid file path. To set a default region, type `ibmcloud cos config region` and provide an input into the program a region code, such as `us-south`. By default, this value is set to `us-geo`.

You can view your current IBM Cloud Object Storage credentials by prompting `ibmcloud cos config list`. As the config file is generated by the plug-in, it's best not to edit the file manually.

```
$ $ ibmcloud cos config list
Key                     Value
Last Updated            Tuesday, April 28 2020 at 19:35:57
Default Region          us-south
Download Location       /home/ibmuser/Downloads
CRN                     8f275e7b-c076-49e2-b9c5-f985704cf678
AccessKeyID             9eib1eejar6HaezaohveV5hikei4aNg2ooV0qu
SecretAccessKey         ********************************************
Authentication Method   IAM
URL Style               VHost
```

IAM Authentication

If you are using IAM authentication, then you then you must configure your client with an instance ID to use some of the commands. To retrieve the instance ID you can type `ibmcloud resource service-instance <INSTANCE_NAME> --id`, replace `<INSTANCE_NAME>` with the unique alias that you assigned to your service instance. In the below examples, the `8f275e7b-c076-49e2-b9c5-f985704cf678` value is an example instance ID.

First, retrieve the CRN and id with the name of your instance. Be sure to use quotes (`'`) on your instance name and that you are logged in to IBM Cloud. Only the last piece of the CRN is needed, the part after `::`.

```
$ $ ibmcloud resource service-instance 'My Awesome Cloud Object Storage'  --id
Retrieving service instance My Awesome Cloud Object Storage in all resource groups under account IBM as ibmuser@us.ibm.com...
crn:v1:bluemix:public:cloud-object-storage:global:a/94400e98c553415c9599db39b9be9219:3b7d66c8-9fdf-4f81-b7e6-08d187f07288::
8f275e7b-c076-49e2-b9c5-f985704cf678
```

Set the CRN with the `ibmcloud cos config crn` command. It may warn you about overwriting. If you don't want to provide the CRN interactively, you can provide it on the same command with the `--crn` flag.

```
$ $ ibmcloud cos config crn
Resource Instance ID CRN:  ()> 8f275e7b-c076-49e2-b9c5-f985704cf678
Saving new Service Instance ID...
OK
Successfully stored your service instance ID.
```

Verify the configuration:

```
$ $ ibmcloud cos config crn --list
Key   Value
CRN   8f275e7b-c076-49e2-b9c5-f985704cf678
```

Alternatively, you might open the web-based console, select **Service credentials** in the sidebar, and create a new set of credentials (or view an existing credential file that you already created).

HMAC Credentials

If preferred, a Service ID's HMAC credentials can be used instead of your API key. Run `ibmcloud cos config hmac` to input the HMAC credentials, and then switch the authorization method by using `ibmcloud cos config auth`.

Note: If you choose to use token authentication with your own API key, you don't need to provide any credentials as the program authenticates you automatically.

At any time, to switch between HMAC and IAM authentication, you can type `ibmcloud cos config auth`. For more information about authentication and authorization in IBM Cloud, see the Identity and Access Management documentation.

Enable tracing in the command line interface

Tracing can be enabled by setting `IBMCLOUD_TRACE` environment variable to `true` (case ignored). When trace is enabled, additional debugging information is printed to the terminal.

On Linux/macOS terminal:

```
$ export IBMCLOUD_TRACE=true
```

On Windows prompt:

```
$ SET IBMCLOUD_TRACE=true
```

To disable tracing, set the `IBMCLOUD_TRACE` environment variable to `false` (case ignored).

Command index

Each operation has an explanation of what it does, how to use it, and any optional or required parameters. Unless specified as optional, any listed parameters are mandatory.

Note: The CLI plug-in doesn't yet support the full suite of features available in Object Storage. Aspera High-Speed Transfer, Immutable Object Storage, creating Key Protect buckets, or Bucket Firewalls cannot be used by the CLI.

Abort a multipart upload

- **Action:** Abort a multipart upload instance by ending the upload to the bucket in the user's IBM Cloud Object Storage account.
- **Usage:** `ibmcloud cos multipart-upload-abort --bucket BUCKET_NAME --key KEY --upload-id ID [--region REGION] [--output FORMAT]`
- **Parameters to provide:**
  - The name of the bucket.
    - Flag: `--bucket BUCKET_NAME`
  - The KEY of the object.
    - Flag: `--key KEY`
  - Upload ID identifying the multipart upload.
    - Flag: `--upload-id ID`
  - *Optional*: The REGION where the bucket is present. If this flag is not provided, the program uses the default option that is specified in config.
    - Flag: `--region REGION`
  - *Optional*: Output FORMAT can be only json or text.
    - Flag: `--output FORMAT`

Configure a static website

- **Action:** Configures a bucket to host a static website.

- **Usage:** `ibmcloud cos bucket-website-put --bucket BUCKET_NAME [--region REGION] [--output FORMAT]`
- **Parameters to provide:**
  - The name of the bucket. =======
  - Flag: `--bucket BUCKET_NAME`
  - The website configuration in the form of a JSON structure. The `file://` prefix is used to load the JSON structure from the specified file, such as `--website-configuration file://<filename.json>`.
  - Flag: `--website-configuration STRUCTURE` The following parameters are available for configuring static website behavior. None are required. For more details, see the documentation.

    ```
    $ {
      "ErrorDocument": {
        "Key": "string"
      },
      "IndexDocument": {
        "Suffix": "string"
      },
      "RoutingRules": [
        {
          "Condition": {
            "HttpErrorCodeReturnedEquals": "string",
            "KeyPrefixEquals": "string"
          },
          "Redirect": {
            "HostName": "string",
            "HttpRedirectCode": "string",
            "Protocol": "http"|"https",
            "ReplaceKeyPrefixWith": "string",
            "ReplaceKeyWith": "string"
          }
        }
        ...
      ]
    }
    ```

    Alternatively, if the bucket website is configured to redirect traffic, it must be the only parameter configured:

    ```
    $   "RedirectAllRequestsTo": {
        "HostName": "string",
        "Protocol": "http"|"https"
      }
      ```
    ```
  - *Optional*: The REGION where the bucket is present. If this flag is not provided, the program uses the default option that is specified in config.
  - Flag: `--region REGION`
  - *Optional*: Output FORMAT can be only json or text.
  - Flag: `--output FORMAT`

## Copy object from bucket

⚠ **Important:** If you want to add metadata to an object during the copying (using the `--metadata` feature), you must add the attribute `--metadata-directive REPLACE` as metadata is copied during the operation by default (an implicit `--metadata-directive COPY`).

- **Action:** Copy an object from source bucket to destination bucket.
- **Usage:** `ibmcloud cos object-copy --bucket BUCKET_NAME --key KEY --copy-source SOURCE [--cache-control CACHING_DIRECTIVES] [--content-disposition DIRECTIVES] [--content-encoding CONTENT_ENCODING] [--content-language LANGUAGE] [--content-type MIME] [--copy-source-if-match ETAG] [--copy-source-if-modified-since TIMESTAMP] [--copy-source-if-none-match ETAG] [--copy-source-if-unmodified-since TIMESTAMP] [--metadata MAP] [--metadata-directive DIRECTIVE] [--region REGION] [--output FORMAT]`
- **Parameters to provide:**
    ```

- The name of the destination bucket.
- Flag: `--bucket BUCKET_NAME`
- The KEY of the object.
- Flag: `--key KEY`
- (SOURCE) The name of the source bucket and key name of the source object, which is separated by a slash (/). Must be URL-encoded.
- Flag: `--copy-source SOURCE`
- *Optional*: Specifies `CACHING_DIRECTIVES` for the request and reply chain.
- Flag: `--cache-control CACHING_DIRECTIVES`
- *Optional*: Specifies presentation information (`DIRECTIVES`).
- Flag: `--content-disposition DIRECTIVES`
- *Optional*: Specifies what content encodings (CONTENT_ENCODING) are applied to the object and thus what decoding mechanisms must be applied to obtain the media-type referenced by the Content-Type header field.
- Flag: `--content-encoding CONTENT_ENCODING`
- *Optional*: The LANGUAGE the content is in.
- Flag: `--content-language LANGUAGE`
- *Optional*: A standard MIME type describing the format of the object data.
- Flag: `--content-type MIME`
- *Optional*: Copies the object if its entity tag (`Etag`) matches the specified tag (`ETAG`).
- Flag: `--copy-source-if-match ETAG`
- *Optional*: Copies the object if it has been modified since the specified time (TIMESTAMP).
- Flag: `--copy-source-if-modified-since TIMESTAMP`
- *Optional*: Copies the object if its entity tag (`ETag`) is different than the specified tag (`ETAG`).
- Flag: `--copy-source-if-none-match ETAG`
- *Optional*: Copies the object if it hasn't been modified since the specified time (TIMESTAMP).
- Flag: `--copy-source-if-unmodified-since TIMESTAMP`
- *Optional*: A MAP of metadata to store.
- Flag: `--metadata MAP` JSON Syntax: The `--metadata` flag takes the `file://` prefix that is used to load the JSON structure from the specified file.

```
$       {
        "file_name": "file_20xxxxxxxxxxxx45.zip",
        "label": "texas",
        "state": "Texas",
        "Date_to": "2019-11-09T16:00:00.000Z",
        "Sha256sum": "9e39dxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx8ce6b68ede3a47",
        "Timestamp": "Thu, 17 Oct 2019 09:22:13 GMT"
      }
      ```

   * _Optional_: Specifies whether the metadata is copied from the source object or replaced with metadata provided in the
request. DIRECTIVE values: COPY,REPLACE.
        * Flag: ` --metadata-directive DIRECTIVE`
    * _Optional_: The REGION where the bucket is present. If this flag is not provided, the program uses the default option that
is specified in config.
        * Flag: `--region REGION`
    * _Optional_: Output FORMAT can be only json or text.
        * Flag: `--output FORMAT`
```

## Create a new bucket

- **Action:** Create a bucket in an IBM Cloud Object Storage instance.
- **Usage:** `ibmcloud cos bucket-create --bucket BUCKET_NAME [--class CLASS_NAME][--class onerate_active] [--ibm-service-instance-id ID] [--region REGION] [--output FORMAT]`
  - Note that you must provide a CRN if you are using IAM authentication. This can be set by using the [`ibmcloud cos config crn`] command.
- **Parameters to provide:**
  - The name of the bucket.
    - Flag: `--bucket BUCKET_NAME`
```

- *Optional*: The name of the Class.
  - Flag: `--class CLASS_NAME`
- User must specify onerate_active when creating a bucket.
  - Flag: `--class onerate_active`
- *Optional*: Sets the IBM Service Instance ID in the request.
  - Flag: `--ibm-service-instance-id ID`
- *Optional*: The REGION where the bucket is present. If this flag is not provided, the program uses the default option that is specified in config.
  - Flag: `--region REGION`
- *Optional*: Output FORMAT can be only json or text.
  - Flag: `--output FORMAT`

Create a new bucket with Key Protect

- **Action:** Create a bucket with Key Protect in an IBM Cloud Object Storage instance.
- **Usage:** `bucket-create --bucket BUCKET_NAME [--ibm-service-instance-id ID] [--class CLASS_NAME] [--region REGION] --kms-root-key-crn CUSTOMERROOTKEYCRN --kms-encryption-algorithm ALGORITHM [--output FORMAT] [--json]`
  - Note that you must provide a CRN if you are using IAM authentication. This can be set by using the `ibmcloud cos config crn` command.
- **Parameters to provide:**
  - The name of the bucket.
    - Flag: `--bucket BUCKET_NAME`
  - The CUSTOMERROOTKEYCRN of the KMS root key to be associated with the bucket for data encryption.
    - Flag: `--kms-root-key-crn CUSTOMERROOTKEYCRN`
  - *Optional*: The ALGORITHM and SIZE to use with the encryption key stored by using key protect.
    - Flag: `--kms-encryption-algorithm ALGORITHM`
  - *Optional*: The name of the Class.
    - Flag: `--class CLASS_NAME`
  - *Optional*: Sets the IBM Service Instance ID in the request.
    - Flag: `--ibm-service-instance-id ID`
  - *Optional*: The REGION where the bucket is present. If this flag is not provided, the program uses the default option that is specified in config.
    - Flag: `--region REGION`
  - *Optional*: Output FORMAT can be only json or text.
    - Flag: `--output FORMAT`
  - (Deprecated): Output returned in raw JSON format..
    - Flag: `--json`

Example:

```
$ ibmcloud cos bucket-create --bucket bucket-name --kms-root-key-crn crn:v1:staging:public:kms:us-
south:a/9978e0xxxxxxxxxxxxxxxxxxxxxx8654:dfdxxxxx-xxxx-xxxx-xxxx-xxxxxxba6eb0:key:7cea005e-75d4-4a08-ad2f-5e56141f6a96 --kms-
encryption-algorithm AES256
```

Create a new bucket with Hyper Protect Crypto Services

- **Action:** Create a new bucket with Hyper Protect Crypto Services.
- **Usage:** `bucket-create --bucket BUCKET_NAME [--ibm-service-instance-id ID] [--class CLASS_NAME] [--region REGION] --kms-root-key-crn CUSTOMERROOTKEYCRN --kms-encryption-algorithm ALGORITHM [--output FORMAT] [--json]`
- **Parameters to provide:**
  - The name of the bucket.
    - Flag: `--bucket BUCKET_NAME`
  - The CUSTOMERROOTKEYCRN of the KMS root key to be associated with the bucket for data encryption.
    - Flag: `--kms-root-key-crn CUSTOMERROOTKEYCRN`
  - *Optional*: The ALGORITHM and SIZE to use with the encryption key stored by using key protect.
    - Flag: `--kms-encryption-algorithm ALGORITHM`
  - *Optional*: The name of the Class.
    - Flag: `--class CLASS_NAME`

    - *Optional*: Sets the IBM Service Instance ID in the request.
        - Flag: `--ibm-service-instance-id ID`
    - *Optional*: The REGION where the bucket is present. If this flag is not provided, the program uses the default option that is specified in config.
        - Flag: `--region REGION`
    - *Optional*: Output FORMAT can be only json or text.
        - Flag: `--output FORMAT`
    - (Deprecated): Output returned in raw JSON format..
        - Flag: `--json`

Example:

```
$ ibmcloud cos bucket-create --bucket bucket-name --kms-root-key-crn crn:v1:bluemix:public:hs-crypto:us-
south:a/ee747e4xxxxxxxxxxxxxxxxxxxxxx7559:ac6xxxxx-xxxx-xxxx-xxxx-xxxxxx1bea99:key:e7451f36-d7ea-4f55-bc1c-ce4bcceb7018
```

Create a new multipart upload

- **Action:** Begin the multipart file upload process by creating a new multipart upload instance.
- **Usage:** `ibmcloud cos multipart-upload-create --bucket BUCKET_NAME --key KEY [--cache-control CACHING_DIRECTIVES] [--content-disposition DIRECTIVES] [--content-encoding CONTENT_ENCODING] [--content-language LANGUAGE] [--content-type MIME] [--metadata MAP] [--region REGION] [--output FORMAT]`
- **Parameters to provide:**
    - The name of the bucket.
        - Flag: `--bucket BUCKET_NAME`
    - The KEY of the object.
        - Flag: `--key KEY`
    - *Optional*: Specifies `CACHING_DIRECTIVES` for the request and reply chain.
        - Flag: `--cache-control CACHING_DIRECTIVES`
    - *Optional*: Specifies presentation information (`DIRECTIVES`).
        - Flag: `--content-disposition DIRECTIVES`
    - *Optional*: Specifies the content encoding (`CONTENT_ENCODING`) of the object..
        - Flag: `--content-encoding CONTENT_ENCODING`
    - *Optional*: The LANGUAGE the content is in.
        - Flag: `--content-language LANGUAGE`
    - *Optional*: A standard MIME type describing the format of the object data.
        - Flag: `--content-type MIME`
    - *Optional*: A MAP of metadata to store.
        - Flag: `--metadata MAP` JSON Syntax: The `--metadata` flag takes the `file://` prefix that is used to load the JSON structure from the specified file.

```
$ {
  "file_name": "file_20xxxxxxxxxxxx45.zip",
  "label": "texas",
  "state": "Texas",
  "Date_to": "2019-11-09T16:00:00.000Z",
  "Sha256sum": "9e39dxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx8ce6b68ede3a47",
  "Timestamp": "Thu, 17 Oct 2019 09:22:13 GMT"
}
```

    - *Optional*: The REGION where the bucket is present. If this flag is not provided, the program uses the default option that is specified in config.
        - Flag: `--region REGION`
    - *Optional*: Output FORMAT can be only json or text.
        - Flag: `--output FORMAT`

Delete an existing bucket

- **Action:** Delete an existing bucket in an IBM Cloud Object Storage instance.

- **Usage:** `ibmcloud cos bucket-delete --bucket BUCKET_NAME [--region REGION] [--force] [--output FORMAT]`
- **Parameters to provide:**
  - The name of the bucket.
    - Flag: `--bucket BUCKET_NAME`
  - *Optional*: The REGION where the bucket is present. If this flag is not provided, the program uses the default option that is specified in config.
    - Flag: `--region REGION`
  - *Optional*: The operation will do not ask for confirmation.
    - Flag: `--force`
  - *Optional*: Output FORMAT can be only json or text.
    - Flag: `--output FORMAT`

Delete bucket CORS

- **Action:** Delete CORS configuration on a bucket in a user's IBM Cloud Object Storage account.
- **Usage:** `ibmcloud cos bucket-cors-delete --bucket BUCKET_NAME [--region REGION] [--output FORMAT]`
- **Parameters to provide:**
  - The name of the bucket.
    - Flag: `--bucket BUCKET_NAME`
  - *Optional*: The REGION where the bucket is present. If this flag is not provided, the program uses the default option that is specified in config.
    - Flag: `--region REGION`
  - *Optional*: Output FORMAT can be only json or text.
    - Flag: `--output FORMAT`

Delete a static website configuration

- **Action:** Removes a bucket's static website configuration.
- **Usage:** `ibmcloud cos bucket-website-delete --bucket BUCKET_NAME [--region REGION] [--output FORMAT]`
- **Parameters to provide:**
  - The name of the bucket. =======
    - Flag: `--bucket BUCKET_NAME`
  - *Optional*: The REGION where the bucket is present. If this flag is not provided, the program uses the default option that is specified in config.
    - Flag: `--region REGION`
  - *Optional*: Output FORMAT can be only json or text.
    - Flag: `--output FORMAT`

Delete an object

- **Action:** Delete an object from a bucket in a user's IBM Cloud Object Storage account.
- **Usage:** `ibmcloud cos object-delete --bucket BUCKET_NAME --key KEY [--region REGION] [--force] [--output FORMAT]`
- **Parameters to provide:**
  - The name of the bucket.
    - Flag: `--bucket BUCKET_NAME`
  - The KEY of the object.
    - Flag: `--key KEY`
  - *Optional*: The REGION where the bucket is present. If this flag is not provided, the program uses the default option that is specified in config.
    - Flag: `--region REGION`
  - *Optional*: The operation will do not ask for confirmation.
    - Flag: `--force`
    - *Optional*: Output FORMAT can be only json or text.
      - Flag: `--output FORMAT`

Delete multiple objects

- **Action:** Delete multiple objects from a bucket in a user's IBM Cloud Object Storage account.

- **Usage:** `ibmcloud cos objects-delete --bucket BUCKET_NAME --delete STRUCTURE [--region REGION] [--output FORMAT]`
- **Parameters to provide:**
  - The name of the bucket.
    - Flag: `--bucket BUCKET_NAME`
  - A STRUCTURE using either shorthand or JSON syntax.
    - Flag: `--delete STRUCTURE`
    - Shorthand Syntax: `--delete 'Objects=[{Key=string},{Key=string}],Quiet=boolean'`
    - JSON Syntax: `--delete file://<filename.json>` ======= The `--delete` command takes a JSON structure listing the objects to delete. In this example, the `file://` prefix is used to load the JSON structure from the specified file.

```
$ {
"Objects": [
 {
 "Key": "string",
 "VersionId": "string"
 }
...
],
"Quiet": true|false
}
```

  - *Optional*: The REGION where the bucket is present. If this flag is not provided, the program uses the default option that is specified in config.
    - Flag: `--region REGION`
  - *Optional*: Output FORMAT can be only json or text.
    - Flag: `--output FORMAT`

Download an object

- **Action:** Download an object from a bucket in a user's IBM Cloud Object Storage account.
- **Usage:** `ibmcloud cos object-get --bucket BUCKET_NAME --key KEY [--if-match ETAG] [--if-modified-since TIMESTAMP] [--if-none-match ETAG] [--if-unmodified-since TIMESTAMP] [--range RANGE] [--response-cache-control HEADER] [--response-content-disposition HEADER] [--response-content-encoding HEADER] [--response-content-language HEADER] [--response-content-type HEADER] [--response-expires HEADER] [--region REGION] [--output FORMAT] [OUTFILE]`
- **Parameters to provide:**
  - The name of the bucket.
    - Flag: `--bucket BUCKET_NAME`
  - The KEY of the object.
    - Flag: `--key KEY`
  - *Optional*: Return the object only if its entity tag (`ETag`) is the same as the `ETAG` specified, otherwise return a 412 (precondition failed).
    - Flag: `--if-match ETAG`
  - *Optional*: Return the object only if it has been modified since the specified TIMESTAMP, otherwise return a 304 (not modified).
    - Flag: `--if-modified-since TIMESTAMP`
  - *Optional*: Return the object only if its entity tag (`ETag`) is different from the `ETAG` specified, otherwise return a 304 (not modified).
    - Flag: `--if-none-match ETAG`
  - *Optional*: Return the object only if it has not been modified since the specified TIMESTAMP, otherwise return a 412 (precondition failed).
    - Flag: `--if-unmodified-since TIMESTAMP`
  - *Optional*: Downloads the specified RANGE bytes of an object.
    - Flag: `--range RANGE`
  - *Optional*: Sets the Cache-Control HEADER of the response.
    - Flag: `--response-cache-control HEADER`
  - *Optional*: Sets the Content-Disposition HEADER of the response.
    - Flag: `--response-content-disposition HEADER`
  - *Optional*: Sets the Content-Encoding HEADER of the response.
    - Flag: `--response-content-encoding HEADER`
  - *Optional*: Sets the Content-Language HEADER of the response.
    - Flag: `--response-content-language HEADER`

        - Flag: `--response-content-language HEADER`
    - *Optional*: Sets the Content-Type HEADER of the response.
        - Flag: `--response-content-type HEADER`
    - *Optional*: Sets the Expires HEADER of the response.
        - Flag: `--response-expires HEADER`
    - *Optional*: The REGION where the bucket is present. If this flag is not provided, the program uses the default option that is specified in config.
        - Flag: `--region REGION`
    - *Optional*: Output FORMAT can be only json or text.
        - Flag: `--output FORMAT`
    - *Optional*: The location where to save the content of the object. If this parameter is not provided, the program uses the default location.
        - Parameter: `OUTFILE`

Download objects by using S3Manager

- **Action:** Download objects from S3 concurrently.
- **Usage:** `ibmcloud cos download --bucket BUCKET_NAME --key KEY [--concurrency value] [--part-size SIZE] [--if-match ETAG] [--if-modified-since TIMESTAMP] [--if-none-match ETAG] [--if-unmodified-since TIMESTAMP] [--range RANGE] [--response-cache-control HEADER] [--response-content-disposition HEADER] [--response-content-encoding HEADER] [--response-content-language HEADER] [--response-content-type HEADER] [--response-expires HEADER] [--region REGION] [--output FORMAT] [OUTFILE]`
- **Parameters to provide:**
    - The name (BUCKET_NAME) of the bucket.
        - Flag: `--bucket BUCKET_NAME`
    - The KEY of the object.
        - Flag: `--key KEY`
    - *Optional*: The number of go routines to spin up in parallel per call to download when sending parts. Default value is 5.
        - Flag: `--concurrency value`
    - *Optional*: The buffer SIZE (in bytes) to use when buffering data into chunks and ending them as parts to S3. The minimum allowed part size is 5MB.
        - Flag: `--part-size SIZE`
    - *Optional*: Return the object only if its entity tag (`ETag`) is the same as the `ETAG` specified, otherwise return a 412 (precondition failed).
        - Flag: `--if-match ETAG`
    - *Optional*: Return the object only if it has been modified since the specified TIMESTAMP, otherwise return a 304 (not modified).
        - Flag: `--if-modified-since TIMESTAMP`
    - *Optional*: Return the object only if its entity tag(`ETag`) is different from the `ETAG` specified, otherwise return a 304 (not modified).
        - Flag: `--if-none-match ETAG`
    - *Optional*: Return the object only if it has not been modified since the specified TIMESTAMP, otherwise return a 412 (precondition failed).
        - Flag: `--if-unmodified-since TIMESTAMP`
    - *Optional*: Downloads the specified RANGE bytes of an object. For more information about the HTTP Range header, click here.
        - Flag: `--range RANGE`
    - *Optional*: Sets the Cache-Control HEADER of the response.
        - Flag: `--response-cache-control HEADER`
    - *Optional*: Sets the Content-Disposition HEADER of the response.
        - Flag: `--response-content-disposition HEADER`
    - *Optional*: Sets the Content-Encoding HEADER of the response.
        - Flag: `--response-content-encoding HEADER`
    - *Optional*: Sets the Content-Language HEADER of the response.
        - Flag: `--response-content-language HEADER`
    - *Optional*: Sets the Content-Type HEADER of the response.
        - Flag: `--response-content-type HEADER`
    - *Optional*: Sets the Expires HEADER of the response.
        - Flag: `--response-expires HEADER`
    - *Optional*: The REGION where the bucket is present. If this flag is not provided, the program will use the default option specified in config.
        - Flag: `--region REGION`
    - *Optional*: Output FORMAT can be only json or text.

        - Flag: `--output FORMAT`
    - *Optional*: The location where to save the content of the object. If this parameter is not provided, the program uses the default location.
        - Parameter: `OUTFILE`

Find a bucket

- **Action:** Determine the region and class of a bucket in an IBM Cloud Object Storage instance.
- **Usage:** `ibmcloud cos bucket-location-get --bucket BUCKET_NAME [--output FORMAT]`
- **Parameters to provide:**
    - The name of the bucket.
        - Flag: `--bucket BUCKET_NAME`
    - *Optional*: Output FORMAT can be only json or text.
        - Flag: `--output FORMAT`

Get a bucket's class

- **Action:** Determine the class of a bucket in an IBM Cloud Object Storage instance.
- **Usage:** `ibmcloud cos bucket-class-get --bucket BUCKET_NAME [--output FORMAT]`
- **Parameters to provide:**
    - The name of the bucket.
        - Flag: `--bucket BUCKET_NAME`
    - *Optional*: Output FORMAT can be only json or text.
        - Flag: `--output FORMAT`

Get bucket CORS

- **Action:** Returns the CORS configuration for the bucket in a user's IBM Cloud Object Storage account.
- **Usage:** `ibmcloud cos bucket-cors-get --bucket BUCKET_NAME [--region REGION] [--output FORMAT]`
- **Parameters to provide:**
    - The name of the bucket.
        - Flag: `--bucket BUCKET_NAME`
    - *Optional*: The REGION where the bucket is present. If this flag is not provided, the program uses the default option that is specified in config.
        - Flag: `--region REGION`
    - *Optional*: Output FORMAT can be only json or text.
        - Flag: `--output FORMAT`

Get a bucket's headers

- **Action:** Determine if a bucket exists in an IBM Cloud Object Storage instance.
- **Usage:** `ibmcloud cos bucket-head --bucket BUCKET_NAME [--region REGION] [--output FORMAT]`
- **Parameters to provide:**
    - The name of the bucket.
        - Flag: `--bucket BUCKET_NAME`
    - *Optional*: The REGION where the bucket is present. If this flag is not provided, the program uses the default option that is specified in config.
        - Flag: `--region REGION`
    - *Optional*: Output FORMAT can be only json or text.
        - Flag: `--output FORMAT`

Complete a multipart upload

- **Action:** Complete a multipart upload instance by assembling the currently uploaded parts and uploading the file to the bucket in the user's IBM Cloud Object Storage account.
- **Usage:** `ibmcloud cos multipart-upload-complete --bucket BUCKET_NAME --key KEY --upload-id ID --multipart-upload STRUCTURE [--region REGION] [--output FORMAT]`
- **Parameters to provide:**
    - The name of the bucket.

        - Flag: `--bucket BUCKET_NAME`
    - The KEY of the object.
        - Flag: `--key KEY`
    - Upload ID identifying the multipart upload.
        - Flag: `--upload-id ID`
    - The STRUCTURE of MultipartUpload to set.
        - Flag: `--multipart-upload STRUCTURE`
        - Shorthand Syntax: `--multipart-upload 'Parts=[{ETag=string,PartNumber=integer},{ETag=string,PartNumber=integer}]'`
        - JSON Syntax: `--multipart-upload file://<filename.json>` ======= The `--multipart-upload` command takes a JSON structure that describes the parts of the multipart upload that should be reassembled into the complete file. In this example, the `file://` prefix is used to load the JSON structure from the specified file.

```
$  {
 "Parts": [
  {
   "ETag": "string",
   "PartNumber": integer
  }
  ...
  ]
 }
```

    - *Optional*: The REGION where the bucket is present. If this flag is not provided, the program uses the default option that is specified in config.
        - Flag: `--region REGION`
    - *Optional*: Output FORMAT can be only json or text.
        - Flag: `--output FORMAT`

Configure the Program

- **Action:** Configure the program's preferences.
- **Usage:** `ibmcloud cos config [COMMAND]`
- **Commands:**
    - Switch between HMAC and IAM authentication.
        - Command: `auth`
    - Store CRN in the config.
        - Command: `crn`
    - Store Default Download Location in the config.
        - Command: `ddl`
    - Store HMAC credentials in the config.
        - Command: `hmac`
    - List configuration.
        - Command: `list`
    - Store Default Region in the config.
        - Command: `region`
    - Switch between `VHost` and Path URL style.
        - Command: `url-style`
    - Set Default Service Endpoint.
        - Command: `endpoint-url`
            - Parameters: =======
                - `--list` displays the current default Service Endpoint, if it has been set. Otherwise, it will be empty.
                - `--url some.end.point.url` will change the Service Endpoint to the value as given.
                - `--clear` removes the default Service Endpoint URL that has been set.

Get a static website configuration

- **Action:** Gets a bucket's static website configuration.
- **Usage:** `ibmcloud cos bucket-website-get --bucket BUCKET_NAME [--region REGION] [--output FORMAT]`
- **Parameters to provide:**
  - The name of the bucket. =======
    - Flag: `--bucket BUCKET_NAME`
  - *Optional*: The REGION where the bucket is present. If this flag is not provided, the program uses the default option that is specified in config.
    - Flag: `--region REGION`
  - *Optional*: Output FORMAT can be only json or text.
    - Flag: `--output FORMAT`

Get an object's headers

- **Action:** Determine if a file exists in a bucket in a user's IBM Cloud Object Storage account.
- **Usage:** `ibmcloud cos object-head --bucket BUCKET_NAME --key KEY [--if-match ETAG] [--if-modified-since TIMESTAMP] [--if-none-match ETAG] [--if-unmodified-since TIMESTAMP] [--range RANGE] [--region REGION] [--output FORMAT]`
- **Parameters to provide:**
  - The name of the bucket.
    - Flag: `--bucket BUCKET_NAME`
  - The KEY of the object.
    - Flag: `--key KEY`
  - *Optional*: Return the object only if its entity tag (`ETag`) is the same as the `ETAG` specified, otherwise return a 412 (precondition failed).
    - Flag: `--if-match ETAG`
  - *Optional*: Return the object only if it has been modified since the specified TIMESTAMP, otherwise return a 304 (not modified).
    - Flag: `--if-modified-since TIMESTAMP`
  - *Optional*: Return the object only if its entity tag (`ETag`) is different from the `ETAG` specified, otherwise return a 304 (not modified).
    - Flag: `--if-none-match ETAG`
  - *Optional*: Return the object only if it has not been modified since the specified TIMESTAMP, otherwise return a 412 (precondition failed).
    - Flag: `--if-unmodified-since TIMESTAMP`
  - Downloads the specified RANGE bytes of an object.
    - Flag: `--range RANGE`
  - *Optional*: The REGION where the bucket is present. If this flag is not provided, the program uses the default option that is specified in config.
    - Flag: `--region REGION`
  - *Optional*: Output FORMAT can be only json or text.
    - Flag: `--output FORMAT`

List all buckets

- **Action:** Print a list of all the buckets in a user's IBM Cloud Object Storage account. Buckets might be located in different regions.
- **Usage:** `ibmcloud cos buckets [--ibm-service-instance-id ID] [--output FORMAT]`
  - Note that you must provide a CRN if you are using IAM authentication. This can be set by using the `ibmcloud cos config crn` command.
- **Parameters to provide:**
  - No parameters to provide.
    - *Optional*: Sets the IBM Service Instance ID in the request.
      - Flag: `--ibm-service-instance-id`
    - *Optional*: Output FORMAT can be only json or text.
      - Flag: `--output FORMAT`

Extended Bucket Listing

- **Action:** Print a list of all the buckets in a user's IBM Cloud Object Storage account. Buckets might be located in different regions.
- **Usage:** `ibmcloud cos buckets-extended [--ibm-service-instance-id ID] [--marker KEY] [--prefix PREFIX] [--page-size SIZE] [--max-items NUMBER] [--output FORMAT]`
  - Note that you must provide a CRN if you are using IAM authentication. This can be set by using the `ibmcloud cos config crn` command.

- **Parameters to provide:**
  - No parameters to provide.
    - *Optional*: Sets the IBM Service Instance ID in the request.
      - Flag: `--ibm-service-instance-id`
    - *Optional*: Specifies the KEY to start with when listing objects in a bucket.
      - Flag: `--marker KEY`
    - *Optional*: Limits the response to keys that begin with the specified PREFIX.
      - Flag: `--prefix PREFIX`
    - *Optional*: The SIZE of each page to get in the service call. This does not affect the number of items returned in the command's output. Setting a smaller page size results in more calls to the service, retrieving fewer items in each call. This can help prevent the service calls from timing out.
      - Flag: `--page-size SIZE`
    - *Optional*: The total NUMBER of items to return in the command's output.
      - Flag: `--max-items NUMBER`
    - *Optional*: Output FORMAT can be only json or text.
      - Flag: `--output FORMAT`

List in-progress multipart uploads

- **Action:** Lists in-progress multipart uploads.
- **Usage:** `ibmcloud cos multipart-uploads --bucket BUCKET_NAME [--delimiter DELIMITER] [--encoding-type METHOD] [--prefix PREFIX] [--key-marker value] [--upload-id-marker value] [--page-size SIZE] [--max-items NUMBER] [--region REGION] [--output FORMAT]`
- **Parameters to provide:**
  - The name of the bucket.
    - Flag: `--bucket BUCKET_NAME`
  - *Optional*: A DELIMITER is a character that you use to group keys.
    - Flag: `--delimiter DELIMITER`
  - *Optional*: Requests to encode the object keys in the response and specifies the encoding METHOD to use.
    - Flag: `--encoding-type METHOD`
  - *Optional*: Limits the response to keys that begin with the specified PREFIX.
    - Flag: `--prefix PREFIX`
  - *Optional*: Together with upload-id-marker, this parameter specifies the multipart upload after which listing should begin.
    - Flag: `--key-marker value`
  - *Optional*: Together with key-marker, specifies the multipart upload after which listing should begin. If key-marker is not specified, the upload-id-marker parameter is ignored.
    - Flag: `--upload-id-marker value`
  - *Optional*: The SIZE of each page to get in the service call. This does not affect the number of items returned in the command's output. Setting a smaller page size results in more calls to the service, retrieving fewer items in each call. This can help prevent the service calls from timing out. (default: 1000).
    - Flag: `--page-size SIZE`
  - *Optional*: The total NUMBER of items to return in the command's output. If the total number of items available is more than the value specified, a NextToken is provided in the command's output. To resume pagination, provide the NextToken value in the starting-token argument of a subsequent command. (default: 0).
    - Flag: `--max-items NUMBER`
  - *Optional*: The REGION where the bucket is present. If this flag is not provided, the program uses the default option that is specified in config.
    - Flag: `--region REGION`
  - *Optional*: Output FORMAT can be only json or text.
    - Flag: `--output FORMAT`

List objects

- **Action:** List files present in a bucket in a user's IBM Cloud Object Storage Account. This operation is currently limited to the 1000 most recently created objects and can't be filtered.
- **Usage:** `ibmcloud cos objects --bucket BUCKET_NAME [--delimiter DELIMITER] [--encoding-type METHOD] [--prefix PREFIX] [--starting-token TOKEN] [--page-size SIZE] [--max-items NUMBER] [--region REGION] [--output FORMAT]`

- **Parameters to provide:**
  - The name of the bucket.
    - Flag: `--bucket BUCKET_NAME`
  - *Optional*: A DELIMITER is a character that you use to group keys.
    - Flag: `--delimiter DELIMITER`
  - *Optional*: Requests to encode the object keys in the response and specifies the encoding METHOD to use.
    - Flag: `--encoding-type METHOD`
  - *Optional*: Limits the response to keys that begin with the specified PREFIX.
    - Flag: `--prefix PREFIX`
  - *Optional*: A TOKEN to specify where to start paginating. This is the NextToken from a previously truncated response.
    - Flag: `--starting-token TOKEN`
  - *Optional*: The SIZE of each page to get in the service call. This does not affect the number of items returned in the command's output. Setting a smaller page size results in more calls to the service, retrieving fewer items in each call. This can help prevent the service calls from timing out. (default: 1000)
    - Flag: `--page-size SIZE`
  - *Optional*: The total NUMBER of items to return in the command's output. If the total number of items available is more than the value specified, a NextToken is provided in the command's output. To resume pagination, provide the NextToken value in the starting-token argument of a subsequent command. (default: 0)
    - Flag: `--max-items NUMBER`
  - *Optional*: The REGION where the bucket is present. If this flag is not provided, the program uses the default option that is specified in config.
    - Flag: `--region REGION`
  - *Optional*: Output FORMAT can be only json or text.
    - Flag: `--output FORMAT`

List objects v2

- **Action:** List all objects in a specific bucket.
- **Usage:** `list-objects-v2 --bucket BUCKET_NAME [--starting-token Starting Token] [--delimiter DELIMITER] [--encoding-type METHOD] [--fetch-owner Boolean] [--max-items NUMBER] [--prefix PREFIX] [--start-after Start After] [--page-size SIZE] [--region REGION] [--output FORMAT] [--json]`
- **Parameters to provide:**
  - The name of the bucket.
    - Flag: `--bucket BUCKET_NAME`
  - *Optional*: A DELIMITER is a character that you use to group keys.
    - Flag: `--delimiter DELIMITER`
  - *Optional*: Requests to encode the object keys in the response and specifies the encoding METHOD to use.
    - Flag: `--encoding-type METHOD`
  - *Optional*: Limits the response to keys that begin with the specified PREFIX.
    - Flag: `--prefix PREFIX`
  - *Optional*: A TOKEN to specify where to start paginating. This is the NextToken from a previously truncated response.
    - Flag: `--starting-token TOKEN`
  - *Optional*: The SIZE of each page to get in the service call. This does not affect the number of items returned in the command's output. Setting a smaller page size results in more calls to the service, retrieving fewer items in each call. This can help prevent the service calls from timing out. (default: 1000)
    - Flag: `--page-size SIZE`
  - *Optional*: The total NUMBER of items to return in the command's output. If the total number of items available is more than the value specified, a NextToken is provided in the command's output. To resume pagination, provide the NextToken value in the starting-token argument of a subsequent command. (default: 0)
    - Flag: `--max-items NUMBER`
  - *Optional*: The REGION where the bucket is present. If this flag is not provided, the program uses the default option that is specified in config.
    - Flag: `--region REGION`
  - *Optional*: Output FORMAT can be only json or text.
    - Flag: `--output FORMAT`
  - *Optional*: The Boolean is not present in listV2 by default, if you want to return owner field with each key in the result then set the fetch owner

field to true.
        - Flag: `--fetch-owner Boolean`
    - *Optional*: Start After is where you want S3 to start listing from. S3 starts listing after this specified key. StartAfter can be any key in the bucket.
        - Flag: `--start-after Start After`
    - *Deprecated*: Output returned in raw JSON format.
        - Flag: `--json`

List parts

- **Action:** Print out information about an in progress multipart upload instance.
- **Usage:** `ibmcloud cos parts --bucket BUCKET_NAME --key KEY --upload-id ID --part-number-marker VALUE [--page-size SIZE] [--max-items NUMBER] [--region REGION] [--output FORMAT]`
- **Parameters to provide:**
    - The name of the bucket.
        - Flag: `--bucket BUCKET_NAME`
    - The KEY of the object.
        - Flag: `--key KEY`
    - Upload ID identifying the multipart upload.
        - Flag: `--upload-id ID`
    - Part number VALUE after which listing begins (default: 1)
        - Flag: `--part-number-marker VALUE`
    - *Optional*: The SIZE of each page to get in the service call. This does not affect the number of items returned in the command's output. Setting a smaller page size results in more calls to the service, retrieving fewer items in each call. This can help prevent the service calls from timing out. (default: 1000)
        - Flag: `--page-size SIZE`
    - *Optional*: The total NUMBER of items to return in the command's output. If the total number of items available is more than the value specified, a NextToken is provided in the command's output. To resume pagination, provide the NextToken value in the starting-token argument of a subsequent command. (default: 0)
        - Flag: `--max-items NUMBER`
    - *Optional*: The REGION where the bucket is present. If this flag is not provided, the program uses the default option that is specified in config.
        - Flag: `--region REGION`
    - *Optional*: Output FORMAT can be only json or text.
        - Flag: `--output FORMAT`

Set bucket CORS

- **Action:** Sets the CORS configuration for a bucket in the user's IBM Cloud Object Storage account.

- **Usage:** `ibmcloud cos bucket-cors-put --bucket BUCKET_NAME [--cors-configuration STRUCTURE] [--region REGION] [--output FORMAT]`

- **Parameters to provide:**

    - The name of the bucket.
        - Flag: `--bucket BUCKET_NAME`
    - *Optional*: A STRUCTURE using JSON syntax in a file.
        - Flag: `--cors-configuration STRUCTURE`
        - JSON Syntax: `--cors-configuration file://<filename.json>` ======= The `--cors-configuration` command takes a JSON structure that describes the CORS configuration. In this example, the `file://` prefix is used to load the JSON structure from the specified file.

```
$ {
"CORSRules": [
 {
   "AllowedHeaders": ["string", ...],
   "AllowedMethods": ["string", ...],
   "AllowedOrigins": ["string", ...],
   "ExposeHeaders": ["string", ...],
   "MaxAgeSeconds": integer
 }
 ...
]
}
```

- *Optional*: The REGION where the bucket is present. If this flag is not provided, the program uses the default option that is specified in config.
  - Flag: `--region REGION`
- *Optional*: Output FORMAT can be only json or text.
  - Flag: `--output FORMAT`

Put object

- **Action:** Upload an object to a bucket in a user's IBM Cloud Object Storage account.
- **Usage:** `ibmcloud cos object-put --bucket BUCKET_NAME --key KEY [--body FILE_PATH] [--cache-control CACHING_DIRECTIVES] [--content-disposition DIRECTIVES] [--content-encoding CONTENT_ENCODING] [--content-language LANGUAGE] [--content-length SIZE] [--content-md5 MD5] [--content-type MIME] [--metadata MAP] [--region REGION] [--output FORMAT]`
- **Parameters to provide:**
  - The name of the bucket.
    - Flag: `--bucket BUCKET_NAME`
  - The KEY of the object.
    - Flag: `--key KEY`
  - *Optional*: Object data location (`FILE_PATH`).
    - Flag: `--body FILE_PATH`
  - *Optional*: Specifies `CACHING_DIRECTIVES` for the request and reply chain.
    - Flag: `--cache-control CACHING_DIRECTIVES`
  - *Optional*: Specifies presentation information (`DIRECTIVES`).
    - Flag: `--content-disposition DIRECTIVES`
  - *Optional*: Specifies the content encoding (`CONTENT_ENCODING`) of the object.
    - Flag: `--content-encoding CONTENT_ENCODING`
  - *Optional*: The LANGUAGE the content is in.
    - Flag: `--content-language LANGUAGE`
  - *Optional*: SIZE of the body in bytes. This parameter is useful when the size of the body cannot be determined automatically. (default: 0)
    - Flag: `--content-length SIZE`
  - *Optional*: The base64-encoded 128-bit MD5 digest of the data.
    - Flag: `--content-md5 MD5`
  - *Optional*: A standard MIME type describing the format of the object data.
    - Flag: `--content-type MIME`
  - *Optional*: A MAP of metadata to store.
    - Flag: `--metadata MAP` JSON Syntax: The `--metadata` flag takes the `file://` prefix that is used to load the JSON structure from the specified file.

```
$ {
  "file_name": "file_20xxxxxxxxxxxx45.zip",
  "label": "texas",
  "state": "Texas",
  "Date_to": "2019-11-09T16:00:00.000Z",
  "Sha256sum": "9e39dxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx8ce6b68ede3a47",
  "Timestamp": "Thu, 17 Oct 2019 09:22:13 GMT"
}
```

  - *Optional*: The REGION where the bucket is present. If this flag is not provided, the program uses the default option that is specified in config.
    - Flag: `--region REGION`
  - *Optional*: Output FORMAT can be only json or text.
    - Flag: `--output FORMAT`

Upload objects by using S3Manager

- **Action:** Upload objects to COS concurrently.
- **Usage:** `ibmcloud cos upload --bucket BUCKET_NAME --key KEY --file PATH [--concurrency value] [--max-upload-parts PARTS] [--part-size SIZE] [--leave-parts-on-errors] [--cache-control CACHING_DIRECTIVES] [--content-disposition DIRECTIVES] [--content-encoding CONTENT_ENCODING] [--content-language LANGUAGE] [--content-length SIZE] [--content-md5 MD5] [--content-type MIME] [--metadata MAP] [--region REGION] [--output FORMAT]`
- **Parameters to provide:**
  - The name (BUCKET_NAME) of the bucket.
    - Flag: `--bucket BUCKET_NAME`
  - The KEY of the object.
    - Flag: `--key KEY`
  - The PATH to the file to upload.
    - Flag: `--file PATH`
  - *Optional*: The number of go routines to spin up in parallel per call to Upload when sending parts. Default value is 5.
    - Flag: `--concurrency value`
  - *Optional*: Max number of PARTS which will be uploaded to S3 that calculates the part size of the object to be uploaded. Limit is 10,000 parts.
    - Flag: `--max-upload-parts PARTS`
  - *Optional*: The buffer SIZE (in bytes) to use when buffering data into chunks and ending them as parts to S3. The minimum allowed part size is 5MB.
    - Flag: `--part-size SIZE`
  - *Optional*: Setting this value to true will cause the SDK to avoid calling AbortMultipartUpload on a failure, leaving all successfully uploaded parts on S3 for manual recovery.
    - Flag: `--leave-parts-on-errors`
  - *Optional*: Specifies CACHING_DIRECTIVES for the request/reply chain.
    - Flag: `--cache-control CACHING_DIRECTIVES`
  - *Optional*: Specifies presentational information (DIRECTIVES).
    - Flag: `--content-disposition DIRECTIVES`
  - *Optional*: Specifies what content encodings (CONTENT_ENCODING) have been applied to the object and thus what decoding mechanisms must be applied to obtain the media-type referenced by the Content-Type header field.
    - Flag: `--content-encoding CONTENT_ENCODING`
  - *Optional*: The LANGUAGE the content is in.
    - Flag: `--content-language LANGUAGE`
  - *Optional*: SIZE of the body in bytes. This parameter is useful when the size of the body cannot be determined automatically.
    - Flag: `--content-length SIZE`
  - *Optional*: The base64-encoded 128-bit MD5 digest of the data.
    - Flag: `--content-md5 MD5`
  - *Optional*: A standard MIME type describing the format of the object data.
    - Flag: `--content-type MIME`
  - *Optional*: A MAP of metadata to store.
    - Flag: `--metadata MAP` JSON Syntax: The `--metadata` flag takes the `file://` prefix that is used to load the JSON structure from the specified file.

```
$ {
  "file_name": "file_20xxxxxxxxxxxx45.zip",
  "label": "texas",
  "state": "Texas",
  "Date_to": "2019-11-09T16:00:00.000Z",
  "Sha256sum": "9e39dxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx8ce6b68ede3a47",
  "Timestamp": "Thu, 17 Oct 2019 09:22:13 GMT"
}
```

  - *Optional*: The REGION where the bucket is present. If this flag is not provided, the program will use the default option specified in config.
    - Flag: `--region REGION`
  - *Optional*: Output FORMAT can be only json or text.
    - Flag: `--output FORMAT`

Manually controlling multipart uploads

The IBM Cloud Object Storage CLI provides the ability for users to upload large files in multiple parts by using the AWS multipart upload functions. To initiate a new multipart upload, run the `multipart-upload-create` command, which returns the new upload instance's upload ID. To continue with the upload process, you must save the upload ID for each subsequent command. This command requires you to generate an MD5 hash:

```
$ {object data} | openssl dgst -md5 -binary | openssl enc -base64
```

After running the `multipart-upload-complete` command, run `part-upload` for each file part you want to upload. **For multipart uploads, every file part (except for the last part) must be at least 5 MB.** To split a file into separate parts, you can run `split` in a terminal window. For example, if you have a 13 MB file that is named `TESTFILE` on your Desktop, and you would like to split it into file parts of 5 MB each, you can run `split -b 3m ~/Desktop/TESTFILE part-file-`. This command generates three file parts into two file parts of 5 MB each, and one file part of 3 MB, with the names `part-file-aa`, `part-file-ab`, and `part-file-ac`. `As each file part is uploaded, the CLI print its`ETag`. You must save this `ETag`into a formatted JSON file, along with the part number. Use this template to create your own`ETag JSON data file.

```
$ {
    "Parts": [
    {
      "PartNumber": 1,
      "ETag": "The ETag of the first file part goes here."
    },
    {
      "PartNumber": 2,
      "ETag": "The ETag of the second file part goes here."
    }
    ]
}
```

Add more entries to this JSON template as necessary.

To see the status of your multipart upload instance, you can always run the `part-list` command, providing the bucket name, key, and the upload ID. This print raw information about your multipart upload instance. Once you have completed uploading each part of the file, run the `multipart-upload-complete` command with the necessary parameters. If all goes well, you receive a confirmation that the file uploaded successfully to the wanted bucket.

Upload a part

- **Action:** Upload a part of a file in an existing multipart upload instance.
- **Usage:** `ibmcloud cos part-upload --bucket BUCKET_NAME --key KEY --upload-id ID --part-number NUMBER [--body FILE_PATH] [--region REGION] [--output FORMAT]`
    - Note that you must save each uploaded file part's number and `ETag` (which the CLI will print for you) for each part into a JSON file. Refer to the "Multipart Upload Guide" below for more information.
- **Parameters to provide:**
    - The bucket name where the multipart upload is taking place.
        - Flag: `--bucket BUCKET_NAME`
    - The KEY of the object.
        - Flag: `--key KEY`
    - Upload ID identifying the multipart upload.
        - Flag: `--upload-id ID`
    - Part NUMBER of part being uploaded. This is a positive integer in the range 1 - 10,000. (default: 1)
        - Flag: `--part-number NUMBER`
    - *Optional*: Object data location (`FILE_PATH`).
        - Flag: `--body FILE_PATH`
    - *Optional*: The REGION where the bucket is present. If this flag is not provided, the program uses the default option that is specified in config.
        - Flag: `--region REGION`
    - *Optional*: Output FORMAT can be only json or text.
        - Flag: `--output FORMAT`

Upload a part copy

- **Action:** Upload a part by copying data from an existing object.
- **Usage:** `ibmcloud cos part-upload-copy --bucket BUCKET_NAME --key KEY --upload-id ID --part-number NUMBER --copy-source SOURCE`

`[--copy-source-if-match ETAG] [--copy-source-if-modified-since TIMESTAMP] [--copy-source-if-none-match ETAG] [--copy-source-if-unmodified-since TIMESTAMP] [--copy-source-range value] [--region REGION] [--output FORMAT]`
    - Note that you must save each uploaded file part's number and `ETag` (which the CLI will print for you) for each part into a JSON file. Refer to the "Multipart Upload Guide" for more information.
- **Parameters to provide:**
    - The name of the bucket.
        - Flag: `--bucket BUCKET_NAME`
    - The KEY of the object.
        - Flag: `--key KEY`
    - Upload ID identifying the multipart upload.
        - Flag: `--upload-id ID`
    - Part NUMBER of part being uploaded. This is a positive integer between 1 and 10,000.
        - Flag: `--part-number PART_NUMBER`
    - (SOURCE) The name of the source bucket and key name of the source object, which is separated by a slash (/). Must be URL-encoded.
        - Flag: `--copy-source SOURCE`
    - *Optional*: Copies the object if its entity tag (`Etag`) matches the specified tag (`ETAG`).
        - Flag: `--copy-source-if-match ETAG`
    - *Optional*: Copies the object if it has been modified since the specified time (TIMESTAMP).
        - Flag: `--copy-source-if-modified-since TIMESTAMP`
    - *Optional*: Copies the object if its entity tag (`ETag`) is different than the specified tag (`ETAG`).
        - Flag: `--copy-source-if-none-match ETAG`
    - *Optional*: Copies the object if it hasn't been modified since the specified time (TIMESTAMP).
        - Flag: `--copy-source-if-unmodified-since TIMESTAMP`
    - *Optional*: The range of bytes to copy from the source object. The range value must use the form bytes=first-last, where the first and last are the zero-based byte offsets to copy. For example, bytes=0-9 indicates that you want to copy the first ten bytes of the source. You can copy a range only if the source object is greater than 5 MB.
        - Flag: `--copy-source-range value`
    - *Optional*: The REGION where the bucket is present. If this flag is not provided, the program uses the default option that is specified in config.
        - Flag: `--region REGION`
    - *Optional*: Output FORMAT can be only json or text.
        - Flag: `--output FORMAT`

Object Lock configuration

Put Object Lock configuration

Note: In default retention Days and Years cannot be provided at the same time.

- **Action:** Set the object lock configuration on a bucket.
- **Usage:** `object-lock-configuration-put --bucket BUCKET_NAME [--object-lock-configuration STRUCTURE] [--region REGION] [--output FORMAT]`
- **Parameters to provide:**
    - The name of the bucket.
        - Flag: `--bucket BUCKET_NAME`
    - A STRUCTURE using JSON syntax. See IBM Cloud Documentation.
        - Flag: `--object-lock-configuration STRUCTURE`

```
$ {
"ObjectLockEnabled": "Enabled",
"Rule": {
 "DefaultRetention": {
 "Mode": "COMPLIANCE",
 "Days": integer,
 "Years": integer
 }
}
}
```

- *Optional*: The REGION where the bucket is present. If this flag is not provided, the program uses the default option that is specified in config.
  - Flag: `--region REGION`
- *Optional*: Output FORMAT can be only json or text.
  - Flag: `--output FORMAT`

Example:

```
$ ibmcloud cos object-lock-configuration-put --bucket bucket-name --object-lock-configuration '{ "ObjectLockEnabled": "Enabled",
"Rule": { "DefaultRetention": { "Mode": "COMPLIANCE", "Days": 30 }}}'
```

Get Object Lock configuration

- **Action:** Get the object lock configuration on a bucket.
- **Usage:** `object-lock-configuration-get --bucket BUCKET_NAME [--region REGION] [--output FORMAT]`
- **Parameters to provide:**
  - The name of the bucket.
    - Flag: `--bucket BUCKET_NAME`
  - A STRUCTURE using JSON syntax. See IBM Cloud Documentation.
    - Flag: `--object-lock-configuration STRUCTURE`

      ```
      $ {
      "ObjectLockEnabled": "Enabled",
      "Rule": {
       "DefaultRetention": {
       "Mode": "COMPLIANCE",
       "Days": integer,
       "Years": integer
       }
      }
      }
      ```
  - *Optional*: The REGION where the bucket is present. If this flag is not provided, the program uses the default option that is specified in config.
    - Flag: `--region REGION`
  - *Optional*: Output FORMAT can be only json or text.
    - Flag: `--output FORMAT`

      ```
      $ {
      "ObjectLockEnabled": "Enabled",
      "Rule": {
       "DefaultRetention": {
       "Mode": "COMPLIANCE",
       "Days": integer,
       "Years": integer
       }
      }
      }
      ```

Example:

```
$ ibmcloud cos object-lock-configuration-get --bucket bucket-name --region us-south
```

Object Retention

Put Object Retention

- **Action:** Set retention on a object.
- **Usage:** `object-retention-put --bucket BUCKET_NAME --key KEY [--retention STRUCTURE] [--region REGION] [--output FORMAT]`
- **Parameters to provide:**
  - The name of the bucket.

        - Flag: `--bucket BUCKET_NAME`
    - The KEY of the object.
        - Flag: `--key KEY`
    - A STRUCTURE using JSON syntax. See [IBM Cloud Documentation](#).
        - Flag: `--retention STRUCTURE`

          ```
          $ {
            "Mode": "COMPLIANCE",
            "RetainUntilDate": timestamp
          }
          ```

    - *Optional*: The REGION where the bucket is present. If this flag is not provided, the program uses the default option that is specified in config.
        - Flag: `--region REGION`
    - *Optional*: Output FORMAT can be only json or text.
        - Flag: `--output FORMAT`

Example:

```
$ ibmcloud cos object-retention-put --bucket bucket-name --key file-name.txt --retention '{ "Mode": "COMPLIANCE",
"RetainUntilDate": "2024-02-02T00:00:00"}’
```

Get Object Retention

- **Action:** Get retention on a object.
- **Usage:** `object-retention-get --bucket BUCKET_NAME --key KEY [--region REGION] [--output FORMAT]`
- **Parameters to provide:**
    - The name of the bucket.
        - Flag: `--bucket BUCKET_NAME`
    - The KEY of the object.
        - Flag: `--key KEY`
    - *Optional*: The REGION where the bucket is present. If this flag is not provided, the program uses the default option that is specified in config.
        - Flag: `--region REGION`
    - *Optional*: Output FORMAT can be only json or text.
        - Flag: `--output FORMAT`

          ```
          $ {
           "Retention": {
            "Mode": "COMPLIANCE",
            "RetainUntilDate": "2024-02-02T00:00:00.000Z"
           }
          }
          ```

Example:

```
$ ibmcloud cos object-retention-put --bucket bucket-name --key file-name.txt --region us-south
```

Object Legal Hold

Put Object Legal Hold

- **Action:** Set the legal hold on a object.
- **Usage:** `object-legal-hold-put --bucket BUCKET_NAME --key KEY [--legal-hold STRUCTURE] [--region REGION] [--output FORMAT]`
- **Parameters to provide:**
    - The name of the bucket.
        - Flag: `--bucket BUCKET_NAME`
    - The KEY of the object.
        - Flag: `--key KEY`
    - A STRUCTURE using JSON syntax. See [IBM Cloud Documentation](#).

        - Flag: `--legalhold STRUCTURE`

          ```
          $ {
             "Status": "ON"|"OFF"
          }
          ```

    - *Optional*: The REGION where the bucket is present. If this flag is not provided, the program uses the default option that is specified in config.
        - Flag: `--region REGION`
    - *Optional*: Output FORMAT can be only json or text.
        - Flag: `--output FORMAT`

Example:

```
$ ibmcloud cos object-legal-hold-put --bucket bucket-name --key file-name.txt --legal-hold '{"Status": "ON"}'
```

Get Object Legal Hold

- **Action:** Get legal hold for a object.
- **Usage:** `object-legal-hold-get --bucket BUCKET_NAME --key KEY [--region REGION] [--output FORMAT]`
- **Parameters to provide:**
    - The name of the bucket.
        - Flag: `--bucket BUCKET_NAME`
    - The KEY of the object.
        - Flag: `--key KEY`
    - *Optional*: The REGION where the bucket is present. If this flag is not provided, the program uses the default option that is specified in config.
        - Flag: `--region REGION`
    - *Optional*: Output FORMAT can be only json or text.

        - Flag: `--output FORMAT`

          ```
          $ {
           "LegalHold": {
            "Status": "ON"
           }
          }
          ```

Example:

```
$ ibmcloud cos object-retention-get --bucket bucket-name --key file-name.txt --region us-south
```

Configure bucket replication

Setup for configuring a replicated bucket.

Put bucket replication

- **Action:** Set the replication configuration on a bucket.
- **Usage:** `bucket-replication-put --bucket BUCKET_NAME [--replication-configuration STRUCTURE] [--region REGION] [--output FORMAT]`
- **Parameters to provide:**
    - The name of the bucket.
        - Flag: `--bucket BUCKET_NAME`
    - *Optional*: The REGION where the bucket is present. If this flag is not provided, the program uses the default option that is specified in config.
        - Flag: `--region REGION`
    - *Optional*: Output FORMAT can be only json or text.

        - Flag: `--output FORMAT`

          ```
          $ {
           "Rules": [
            {
          ```

```
    "Status": "Enabled",
    "Priority": 1,
    "Filter" : { "Prefix": ""},
    "DeleteMarkerReplication": {
     "Status": "Disabled"
    },
    "Destination": {
     "Bucket": "DEST-BUCKET-NAME"
    }
   }
  ]
 }
```

Example:

```
$ ibmcloud cos bucket-replication-put --bucket SOURCE-BUCKET-NAME --replication-configuration file://replication.json
```

Get bucket replication

- **Action:** Get the replication configuration for a bucket.
- **Usage:** `bucket-replication-get --bucket BUCKET_NAME [--region REGION] [--output FORMAT]`
- **Parameters to provide:**
  - The name of the bucket.
    - Flag: `--bucket BUCKET_NAME`
  - *Optional*: The REGION where the bucket is present. If this flag is not provided, the program uses the default option that is specified in config.
    - Flag: `--region REGION`
  - *Optional*: Output FORMAT can be only json or text.
    - Flag: `--output FORMAT`

```
$ {
 "ReplicationConfiguration": {
  "Rules": [
   {
    "Status": "Enabled",
    "Prefix": "",
    "Destination": {
     "Bucket": "DEST-BUCKET-NAME",
     "StorageClass": "STANDARD"
    },
   }
  ],
 }
}
```

Example:

```
$  ibmcloud cos bucket-replication-get --bucket SOURCE-BUCKET-NAME
```

Delete bucket replication

- **Action:** Delete the replication configuration from a bucket.
- **Usage:** `bucket-replication-delete --bucket BUCKET_NAME [--region REGION] [--output FORMAT]`
- **Parameters to provide:**
  - The name of the bucket.
    - Flag: `--bucket BUCKET_NAME`
  - *Optional*: The REGION where the bucket is present. If this flag is not provided, the program uses the default option that is specified in config.
    - Flag: `--region REGION`
  - *Optional*: Output FORMAT can be only json or text.
    - Flag: `--output FORMAT`

Example:

```
$  ibmcloud cos bucket-replication-delete --bucket SOURCE-BUCKET-NAME
```

Next Steps

As every procedure always goes exactly as planned, you might not have seen any of the common header and error codes. For more reference, check the API reference.

Using `cURL`

You can get the most out working with the command line in most environments with IBM Cloud® Object Storage and `cURL`.

Here's a 'cheat sheet' of basic `curl` commands for the IBM Cloud® Object Storage REST API. More detail can be found in the API reference for buckets or objects.

Using `curl` assumes a certain amount of familiarity with the command line and Object Storage, and have the necessary information from a service credential, the endpoints reference, or the console. If any terms or variables are unfamiliar, they can be found in the glossary.

Tip: Note: Personally Identifiable Information (PII): When *naming* buckets or objects, do not use any information that can identify any user (natural person) by name, location, or any other means.

Request an IAM Token

Two ways you can generate an IAM `oauth` token for authenticating requests are using a `curl` command with an API key (described later), or from the command line by using IBM Cloud® CLI.

Request an IAM token by using an API key

Ensure that you have an API key. You can get one from IBM Cloud® Identity and Access Management.

```
curl -X "POST" "https://iam.cloud.ibm.com/oidc/token" \
     -H 'Accept: application/json' \
     -H 'Content-Type: application/x-www-form-urlencoded' \
     --data-urlencode "apikey={api-key}" \
     --data-urlencode "response_type=cloud_iam" \
     --data-urlencode "grant_type=urn:ibm:params:oauth:grant-type:apikey"
```

Get your resource instance ID

Some of the following commands require an `ibm-service-instance-id` parameter. To find this value, go to the **Service credentials** tab of your Object Storage instance in the cloud console. Create a credential if needed, then use the *View credentials* menu to see the JSON format. Use the value of `resource_instance_id`.

Tip: For use with curl APIs, you need only the UUID that starts after the last single colon and ends before the final double colon. For example, the ID `crn:v1:bluemix:public:cloud-object-storage:global:a/81caa0254631ce5f9330ae427618f209:39d8d161-22c4-4b77-a856-f11db5130d7d::` can be abbreviated to `39d8d161-22c4-4b77-a856-f11db5130d7d`.

List buckets

```
curl "https://(endpoint)/"
 -H "Authorization: bearer (token)"
 -H "ibm-service-instance-id: (resource-instance-id)"
```

Add a bucket

```
curl -X "PUT" "https://(endpoint)/(bucket-name)"
 -H "Authorization: Bearer (token)"
 -H "ibm-service-instance-id: (resource-instance-id)"
```

Add a bucket (storage class)

```
curl -X "PUT" "https://(endpoint)/(bucket-name)"
```

```
 -H "Content-Type: text/plain; charset=utf-8"
 -H "Authorization: Bearer (token)"
 -H "ibm-service-instance-id: (resource-instance-id)"
 -d "<CreateBucketConfiguration>
       <LocationConstraint>(provisioning-code)</LocationConstraint>
     </CreateBucketConfiguration>"
```

A list of valid codes for `LocationConstraint` can be referenced in [the Storage Classes guide](#).

Create a bucket CORS

```
curl -X "PUT" "https://(endpoint)/(bucket-name)/?cors"
 -H "Content-MD5: (md5-hash)"
 -H "Authorization: bearer (token)"
 -H "Content-Type: text/plain; charset=utf-8"
 -d "<CORSConfiguration>
      <CORSRule>
        <AllowedOrigin>(url)</AllowedOrigin>
        <AllowedMethod>(request-type)</AllowedMethod>
        <AllowedHeader>(url)</AllowedHeader>
      </CORSRule>
     </CORSConfiguration>"
```

The `Content-MD5` header needs to be the binary representation of a base64-encoded MD5 hash.

```
echo -n (XML block) | openssl dgst -md5 -binary | openssl enc -base64
```

Get a bucket CORS

```
curl "https://(endpoint)/(bucket-name)/?cors"
 -H "Authorization: bearer (token)"
```

Delete a bucket CORS

```
curl -X "DELETE" "https://(endpoint)/(bucket-name)/?cors"
 -H "Authorization: bearer (token)"
```

List objects

```
curl "https://(endpoint)/(bucket-name)"
 -H "Authorization: bearer (token)"
```

Get bucket headers

```
curl --head "https://(endpoint)/(bucket-name)/"
 -H "Authorization: bearer (token)"
```

Get bucket metadata

Note: The use of the config API endpoint isn't the same as the endpoint for your bucket itself. Use of this command returns metadata for the specified bucket.

```
curl https://config.cloud-object-storage.cloud.ibm.com/v1/b/{my-bucket} \
                        -H 'authorization: bearer <IAM_token>'
```

Delete a bucket

```
curl -X "DELETE" "https://(endpoint)/(bucket-name)/"
 -H "Authorization: bearer (token)"
```

Upload an object

```
curl -X "PUT" "https://(endpoint)/(bucket-name)/(object-key)" \
 -H "Authorization: bearer (token)" \
 -H "Content-Type: (content-type)" \
 -d "(object-contents)"
```

Get an object's headers

```
curl --head "https://(endpoint)/(bucket-name)/(object-key)"
 -H "Authorization: bearer (token)"
```

Copy an object

```
curl -X "PUT" "https://(endpoint)/(bucket-name)/(object-key)"
 -H "Authorization: bearer (token)"
 -H "x-amz-copy-source: /(bucket-name)/(object-key)"
```

Check CORS information

```
curl -X "OPTIONS" "https://(endpoint)/(bucket-name)/(object-key)"
 -H "Access-Control-Request-Method: PUT"
 -H "Origin: http://(url)"
```

Download an object

```
curl "https://(endpoint)/(bucket-name)/(object-key)"
 -H "Authorization: bearer (token)"
```

Check object's ACL

```
curl "https://(endpoint)/(bucket-name)/(object-key)?acl"
 -H "Authorization: bearer (token)"
```

Enable a firewall

Note: The use of the config API endpoint isn't the same as the endpoint for your bucket itself. Use of this command enables a firewall for the specified bucket. No other IBM Cloud® services can access the bucket when the firewall is active.

```
curl -X PATCH https://config.cloud-object-storage.cloud.ibm.com/v1/b/{my-bucket} \
                        -H 'authorization: bearer $IAM_TOKEN' \
                        -d '{"firewall": {"allowed_ip": ["10.142.175.0/22", "10.198.243.79"]}}'
```

Enable activity tracking

Note the use of the config API endpoint isn't the same as the endpoint for your bucket itself. Use of this command enables activity tracking for the specified bucket.

```
curl -X PATCH https://config.cloud-object-storage.cloud.ibm.com/v1/b/{my-bucket} \
                        -H 'authorization: bearer <IAM_token>' \
                        -d '{"activity_tracking": { \
                                "read_data_events": True, \
                                "write_data_events": True}'
```

Allow anonymous access to an object

```
curl -X "PUT" "https://(endpoint)/(bucket-name)/(object-key)?acl"
 -H "Content-Type: (content-type)"
 -H "Authorization: bearer (token)"
 -H "x-amz-acl: public-read"
```

Delete an object

```
curl -X "DELETE" "https://(endpoint)/(bucket-name)/(object-key)"
 -H "Authorization: bearer (token)"
```

Delete many objects

```
curl -X "POST" "https://(endpoint)/(bucket-name)?delete"
 -H "Content-MD5: (md5-hash)"
 -H "Authorization: bearer (token)"
 -H "Content-Type: text/plain; charset=utf-8"
 -d "<?xml version="1.0" encoding="UTF-8"?>
         <Delete>
           <Object>
             <Key>(first-object)</Key>
           </Object>
           <Object>
             <Key>(second-object)</Key>
           </Object>
         </Delete>"
```

The `Content-MD5` header needs to be the binary representation of a base64-encoded MD5 hash.

```
echo -n (XML block) | openssl dgst -md5 -binary | openssl enc -base64
```

Start a multipart upload

```
curl -X "POST" "https://(endpoint)/(bucket-name)/(object-key)?uploads"
 -H "Authorization: bearer (token)"
```

Upload a part

```
curl -X "PUT" "https://(endpoint)/(bucket-name)/(object-key)?partNumber=(sequential-integer)&uploadId=(upload-id)"
 -H "Authorization: bearer (token)"
 -H "Content-Type: (content-type)"
```

Complete a multipart upload

```
curl -X "POST" "https://(endpoint)/(bucket-name)/(object-key)?uploadId=(upload-id)"
 -H "Authorization: bearer (token)"
 -H "Content-Type: text/plain; charset=utf-8"
 -d "<CompleteMultipartUpload>
         <Part>
           <PartNumber>1</PartNumber>
           <ETag>(etag)</ETag>
         </Part>
         <Part>
           <PartNumber>2</PartNumber>
           <ETag>(etag)</ETag>
         </Part>
       </CompleteMultipartUpload>"
```

Get incomplete multipart uploads

```
curl "https://(endpoint)/(bucket-name)/?uploads"
 -H "Authorization: bearer (token)"
```

Stop incomplete multipart uploads

```
curl -X "DELETE" "https://(endpoint)/(bucket-name)/(object-key)?uploadId"
 -H "Authorization: bearer (token)"
```

Configure a Static Website

```
$ curl --location --request PUT 'https://<endpoint>/<bucketname>?website' \
--header 'Authorization: bearer <token>' --header 'ibm-service-instance-id: <resource_instance_id> \
--header 'Content-MD5: <hashed-output>' --header 'Content-Type: text/plain' \
--data-raw '<WebsiteConfiguration>
    <IndexDocument>
        <Suffix>index.html</Suffix>
    </IndexDocument>
    <ErrorDocument>
        <Key>error.html</Key>
    </ErrorDocument>
</WebsiteConfiguration>'
```

As a reminder, the `Content-MD5` header needs to be the binary representation of a base64-encoded MD5 hash.

```
$ echo -n (XML block) | openssl dgst -md5 -binary | openssl enc -base64
```

Next Steps

The detailed description of the RESTful API for IBM Cloud Object Storage can be found in the S3 Compatibility API Documentation or the Configuration API Documentation.

Using the AWS CLI

The official command-line interface for AWS is compatible with the IBM Cloud® Object Storage S3 API.

Written in Python, it can be installed from the Python Package Index via `pip install awscli`. By default, access keys are sourced from `~/.aws/credentials`, but can also be set as environment variables.

These examples were generated by using version 1.14.2 of the CLI. To check the version installed, run `aws --version`.

Configure the CLI to connect to Object Storage

To configure the AWS CLI, type `aws configure`. Provide your HMAC credentials and a default region name. The "region name" used by AWS S3 corresponds to the code (`LocationConstraint`) that Object Storage uses to define a storage class.

A list of valid codes for `LocationConstraint` can be referenced in the Storage Classes guide.

```
$ aws configure
AWS Access Key ID [None]: {Access Key ID}
AWS Secret Access Key [None]: {Secret Access Key}
Default region name [None]: {Provisioning Code}
Default output format [None]: json
```

This creates two files:

`~/.aws/credentials`:

```
[default]
aws_access_key_id = {Access Key ID}
aws_secret_access_key = {Secret Access Key}
```

`~/.aws/config`:

```
[default]
region = {Provisioning Code}
output = json
```

You can also use environment variables to set HMAC credentials:

```
export AWS_ACCESS_KEY_ID="{Access Key ID}"
export AWS_SECRET_ACCESS_KEY="{Secret Access Key}"
```

The IBM COS endpoint must be sourced by using the `--endpoint-url` option, and can't be set in the credentials file.

High-level syntax commands

Simple use cases can be accomplished by using `aws --endpoint-url {endpoint} s3 <command>`. For more information about endpoints, see [Endpoints and storage locations](). Objects are managed by using familiar shell commands, such as `ls`, `mv`, `cp`, and `rm`. Buckets can be created by using `mb` and deleted by using `rb`.

List all buckets within a service instance

```
$ aws --endpoint-url {endpoint} s3 ls
2016-09-09 12:48  s3://bucket-1
2016-09-16 21:29  s3://bucket-2
```

List objects within a bucket

```
$ aws --endpoint-url {endpoint} s3 ls s3://bucket-1
2016-09-28 15:36       837   s3://bucket-1/c1ca2-filename-00001
2016-09-09 12:49       533   s3://bucket-1/c9872-filename-00002
2016-09-28 15:36     14476   s3://bucket-1/98837-filename-00003
2016-09-29 16:24     20950   s3://bucket-1/abfc4-filename-00004
```

Make a new bucket

Tip: Note: Personally Identifiable Information (PII): When *naming* buckets or objects, do not use any information that can identify any user (natural person) by name, location, or any other means.

If the default region in the `~/.aws/config` file corresponds the same location as the chosen endpoint, then bucket creation is straightforward.

```
$ aws --endpoint-url {endpoint} s3 mb s3://bucket-1
make_bucket: s3://bucket-1/
```

Add an object to a bucket

```
$ aws --endpoint-url {endpoint} s3 cp large-dataset.tar.gz s3://bucket-1
upload: ./large-dataset.tar.gz to s3://bucket-1/large-dataset.tar.gz
```

You can also set a new object key that is different from the file name:

```
$ aws --endpoint-url {endpoint} s3 cp large-dataset.tar.gz s3://bucket-1/large-dataset-for-project-x
upload: ./large-dataset.tar.gz to s3://bucket-1/large-dataset-for-project-x
```

Copying an object from one bucket to another within the same region:

```
$ $ aws --endpoint-url {endpoint} s3 cp s3://bucket-1/new-file s3://bucket-2/
copy: s3://bucket-1/new-file to s3://bucket-2/new-file
```

Delete an object from a bucket

```
$ aws --endpoint-url {endpoint} s3 rm s3://mybucket/argparse-1.2.1.tar.gz
delete: s3://mybucket/argparse-1.2.1.tar.gz
```

Remove a bucket

```
$ aws --endpoint-url {endpoint} s3 rb s3://bucket-1
remove_bucket: s3://bucket-1/
```

Create pre-signed URLs

The CLI can create pre-signed URLs. These URLs allow for temporary public access to objects without changing any existing access controls.

```
$ $ aws --endpoint-url {endpoint} s3 presign s3://bucket-1/new-file
```

It's also possible to set a time-to-live for the URL in seconds (default is 3600):

```
$ $ aws --endpoint-url {endpoint} s3 presign s3://bucket-1/new-file --expires-in 600
```

Low-level syntax commands

The AWS CLI also allows direct API calls that provide the same responses as direct HTTP requests by using the `s3api` command.

Listing buckets:

```
$ $ aws --endpoint-url {endpoint} s3api list-buckets
{
    "Owner": {
        "DisplayName": "{storage-account-uuid}",
        "ID": "{storage-account-uuid}"
    },
    "Buckets": [
        {
            "CreationDate": "2016-09-09T12:48:52.442Z",
            "Name": "bucket-1"
        },
        {
            "CreationDate": "2016-09-16T21:29:00.912Z",
            "Name": "bucket-2"
        }
    ]
}
```

Listing objects within a bucket

```
$ $ aws --endpoint-url {endpoint} s3api list-objects --bucket bucket-1
```

```
$ {
    "Contents": [
        {
            "LastModified": "2016-09-28T15:36:56.807Z",
            "ETag": "\"13d567d518c650414c50a81805fff7f2\"",
            "StorageClass": "STANDARD",
            "Key": "c1ca2-filename-00001",
            "Owner": {
                "DisplayName": "{storage-account-uuid}",
                "ID": "{storage-account-uuid}"
            },
            "Size": 837
        },
        {
            "LastModified": "2016-09-09T12:49:58.018Z",
            "ETag": "\"3ca744fa96cb95e92081708887f63de5\"",
            "StorageClass": "STANDARD",
            "Key": "c9872-filename-00002",
            "Owner": {
                "DisplayName": "{storage-account-uuid}",
                "ID": "{storage-account-uuid}"
            },
            "Size": 533
        },
        {
            "LastModified": "2016-09-28T15:36:17.573Z",
            "ETag": "\"a54ed08bcb07c28f89f4b14ff54ce5b7\"",
            "StorageClass": "STANDARD",
            "Key": "98837-filename-00003",
            "Owner": {
                "DisplayName": "{storage-account-uuid}",
                "ID": "{storage-account-uuid}"
            },
            "Size": 14476
        },
        {
            "LastModified": "2016-10-06T14:46:26.923Z",
            "ETag": "\"2bcc8ee6bc1e4b8cd2f9a1d61d817ed2\"",
            "StorageClass": "STANDARD",
            "Key": "abfc4-filename-00004",
            "Owner": {
                "DisplayName": "{storage-account-uuid}",
                "ID": "{storage-account-uuid}"
            },
            "Size": 20950
        }
    ]
}
```

Configure a Static Website

```
$ aws --endpoint-url=https://<endpoint> s3 website s3://<bucketname>/ --index-document index.html --error-document error.html
```

Next Steps

The detailed description of the RESTful API for IBM Cloud Object Storage can be found in the S3 Compatibility API Documentation or the Configuration API Documentation.

Mounting a bucket using `s3fs`

Applications that expect to read and write to a NFS-style filesystem can use `s3fs`, which can mount a bucket as directory while preserving the native object format for files.

This allows you to interact with your cloud storage using familiar shell commands, like `ls` for listing or `cp` to copy files, as well as providing access to legacy applications that rely on reading and writing from local files. For a more detailed overview, visit the project's official README.

Tip: Looking for instructions for how to use IBM Cloud® Object Storage in an IBM Cloud Kubernetes Service cluster? Go to the IBM Cloud Kubernetes Service documentation instead.

Prerequisites

- IBM Cloud account and an instance of IBM Cloud® Object Storage
- A Linux or macOS environment
- Credentials (either an IAM API key or HMAC credentials)

Installation

On Debian or Ubuntu:

```
sudo apt-get install automake autotools-dev fuse g++ git libcurl4-openssl-dev libfuse-dev libssl-dev libxml2-dev make pkg-config
```

On RHEL and CentOS 7 or newer by means of EPEL:

```
sudo yum install epel-release
sudo yum install s3fs-fuse
```

Important: Your device must have public connection to pull this EPEL repo, as it is not available in IBM's private repo. See How to install EPEL on RHEL and CentOS Stream for more information.

Tip: The official `s3fs` documentation suggests using `libcurl4-gnutls-dev` instead of `libcurl4-openssl-dev`. Either work, but the OpenSSL version may result in better performance.

For macOS, you will need to build `s3fs` from source:

Ensure you have the following packages installed (all are available via Homebrew):

- `macfuse`
- `automake`
- `gcc`
- `curl`
- `libxml2`

- `pkg-config`
- `openssl`

And as noted in the output of the `openssl` install, you'll need to set these environment variables:

```
export LDFLAGS="-L/usr/local/opt/openssl@3/lib"
export CPPFLAGS="-I/usr/local/opt/openssl@3/include"
export PKG_CONFIG_PATH="/usr/local/opt/openssl@3/lib/pkgconfig"
```

⚠ **Important:** Be aware that macFUSE is closed-source software containing a kernel extension, and may require a license for commercial use.

First clone the Github repository:

```
git clone https://github.com/s3fs-fuse/s3fs-fuse.git
```

Then build `s3fs`:

```
cd s3fs-fuse
./autogen.sh
./configure
make
```

And install the binary:

```
sudo make install
```

Configuration

Store your credentials in a file containing either `<access_key>:<secret_key>` or `:<api_key>`. This file needs to have limited access so run:

```
chmod 0600 <credentials_file>
```

Now you can mount a bucket using:

```
s3fs <bucket> <mountpoint> -o url=http{s}://<endpoint> -o passwd_file=<credentials_file>
```

If the credentials file only has an API key (no HMAC credentials), you'll need to add the `ibm_iam_auth` flag as well:

```
s3fs <bucket> <mountpoint> -o url=http{s}://<endpoint> -o passwd_file=<credentials_file> -o ibm_iam_auth
```

The `<bucket>` in the example refers to an existing bucket and the `<mountpoint>` is the local path where you want to mount the bucket. The `<endpoint>` must correspond to the bucket's location. The `credentials_file` is the file created with the API key or HMAC credentials.

Now, `ls <mountpoint>` will list the objects in that bucket as if they were local files (or in the case of object prefixes, as if they were nested directories).

Performance optimization

While performance will never be equal to a true local filesystem, it is possible to use some advanced options to increase throughput.

```
s3fs <bucket_name> <mountpoint> -o url=http{s}://<COS_endpoint> –o passwd_file=<credentials_file> \
-o cipher_suites=AESGCM \
-o kernel_cache \
-o max_background=1000 \
-o max_stat_cache_size=100000 \
-o multipart_size=52 \
-o parallel_count=30 \
-o multireq_max=30 \
-o dbglevel=warn
```

1. `cipher_suites=AESGCM` is only relevant when using an HTTPS endpoint. By default, secure connections to IBM COS use the `AES256-SHA` cipher suite. Using an `AESGCM` suite instead greatly reduces the CPU load on your client machine, caused by the TLS crypto functions, while offering the same level of cryptographic security.
2. `kernel_cache` enables the kernel buffer cache on your `s3fs mountpoint`. This means that objects will only be read once by `s3fs`, as repetitive

reading of the same file can be served from the kernel's buffer cache. The kernel buffer cache will only use free memory which is not in use by other processes. This option is not recommend if you expect the bucket objects to be overwritten from another process/machine while the bucket is mounted, and your use-case requires live accessing the most up-to-date content.

3. `max_background=1000` improves `s3fs` concurrent file reading performance. By default, FUSE supports file read requests of up to 128 KB. When asking to read more than that, the kernel split the large request to smaller sub-requests and lets s3fs process them asynchronously. The `max_background` option sets the global maximum number of such concurrent asynchronous requests. By default, it is set to 12, but setting it to an arbitrary high value (1000) prevents read requests from being blocked, even when reading many files simultaneously.
4. `max_stat_cache_size=100000` reduces the number of redundant HTTP `HEAD` requests sent by `s3fs` and reduces the time it takes to list a directory or retrieve file attributes. Typical file system usage makes frequent access to a file's metadata via a `stat()` call which maps to `HEAD` request on the object storage system. By default, `s3fs` caches the attributes (metadata) of up to 1000 objects. Each cached entry takes up to 0.5 KB of memory. Ideally, you would want the cache to be able to hold the metadata for all the objects in your bucket. However, you may want to consider the memory usage implications of this caching. Setting it to `100000` will take no more than 0.5 KB * 100000 = 50 MB.
5. `multipart_size=52` will set the maximum size of requests and responses sent and received from the COS server, in MB scale. `s3fs` sets this to 10 MB by default. Increasing this value also increases the throughput (MB/s) per HTTP connection. On the other hand, the latency for the first byte served from the file will also increase. Therefore, if your use-case only reads a small amount of data from each file, you probably do not want to increase this value. Furthermore, for large objects (say, over 50 MB) throughput increases if this value is small enough to allow the file to be fetched concurrently using multiple requests. I find that the optimal value for this option is around 50 MB. COS best practices suggest using requests that are multiples of 4 MB, and thus the recommendation is to set this option to 52 (MB).
6. `parallel_count=30` sets the maximum number of requests sent concurrently to COS, per single file read/write operation. By default, this is set to 5. For very large objects, you can get more throughput by increasing this value. As with the previous option, keep this value low if you only read a small amount of data of each file.
7. `multireq_max=30` When listing a directory, an object metadata request (`HEAD`) is sent per each object in the listing (unless the metadata is found in cache). This option limits the number of concurrent such requests sent to COS, for a single directory listing operation. By default it is set to 20. Note that this value must be greater or equal to the `parallel_count` option above.
8. `dbglevel=warn` sets the debug level to `warn` instead of the default (`crit`) for logging messages to /var/log/syslog.

Limitations

It is important to remember that s3fs may not be suitable for all applications, as object storage services have high-latency for time to first byte and lack random write access. Workloads that only read big files, like deep learning workloads, can achieve good throughput using `s3fs`.

Using Minio Client

The open source Minio Client could be your solution for using UNIX-like commands (`ls` , `cp` , `cat` , and so on) with IBM Cloud® Object Storage.

Installation

You can find installation instructions for each operating system is available in the Quick Start guide on the Minio website.

Configuration

Adding your Object Storage is accomplished by running the following command:

```
$ mc config host add <ALIAS> <COS-ENDPOINT> <ACCESS-KEY> <SECRET-KEY>
```

- `<ALIAS>` - short name for referencing Object Storage in commands
- `<COS-ENDPOINT>` - endpoint for your Object Storage instance. For more information about endpoints, see Endpoints and storage locations.
- `<ACCESS-KEY>` - access key that is assigned to your Service Credential
- `<SECRET-KEY>` - secret key that is assigned to your Service Credential

Tip: The `<ACCESS-KEY>` and `<SECRET-KEY>` can be accessed/generated using HMAC

The configuration information is stored in a JSON file that is at `~/.mc/config.json`

```
$ mc config host add cos https://s3.us-south.cloud-object-storage.appdomain.cloud xx1111cfbe094710x4819759x57e9999
9f99fc08347d1a6xxxxx0b7e0a9ee7b0c9999c2c08ed0000
```

Sample Commands

A complete list of commands and optional flags and parameters are documented in the Minio Client Complete Guide

`mb` - Make a Bucket

```
$ mc mb cos/my_test_bucket
```

`ls` - List Buckets

Tip: Though all your available buckets are listed, not all objects might be accessible depending on the specified endpoint's region.

```
$ mc ls cos
```

```
$ [2018-06-05 09:55:08 HST]     0B testbucket1/
[2018-05-24 04:17:34 HST]     0B testbucket_south/
[2018-10-15 16:14:28 HST]     0B my_test_bucket/
```

`ls` - List Objects

```
$ mc ls cos/testbucket1
```

```
$ [2018-11-12 08:09:53 HST]    34B mynewfile1.txt
[2018-05-31 01:49:26 HST]    34B mynewfile12.txt
[2018-08-10 09:49:08 HST]  20MiB newbigfile.pdf
[2018-11-29 09:53:15 HST]    31B testsave.txt
```

`find` - Search for Objects by Name

Tip: A full list of search options is available in the complete guide

```
$ mc find cos/testbucket1 --name my*
```

```
$ [2018-11-12 08:09:53 HST]    34B mynewfile1.txt
[2018-05-31 01:49:26 HST]    34B mynewfile12.txt
```

`head` - Display few lines of object

```
$ mc head cos/testbucket1/mynewfile1.txt
```

`cp` - Copy objects

This command copies an object between two locations. These locations can be different hosts (such as different endpoints or storage services) or local file system locations (such as `~/foo/filename.pdf`).

```
$ mc cp cos/testbucket1/mynewfile1.txt cos/my_test_bucket/cp_from_minio.txt
```

```
$ ...1/mynewfile1.txt:  34 B / 34 B  ▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓▓  100.00% 27 B/s 1s
```

`rm` - Remove objects

More removal options are available on the complete guide

```
$ mc rm cos/my_test_bucket/cp_from_minio.txt
```

`pipe` - Copies STDIN to an object

```
$ echo -n 'this is a test' | mc pipe cos/my_test_bucket/stdin_pipe_test.txt
```

Using `rclone`

Getting the most out of IBM Cloud® Object Storage when you have access to tools and solutions like `rclone` and the command-line interface (cli).

Install `rclone`

The `rclone` tool is useful for keeping directories synchronized and for migrating data between storage platforms. It's a Go program and comes as a single binary file.

Quick start Installation

- Download the relevant binary.
- Extract the `rclone` or `rclone.exe` binary from the archive.
- Run `rclone config` to set up.

Installation by using a script

Install `rclone` on Linux/macOS/BSD systems:

```
$ curl https://rclone.org/install.sh | sudo bash
```

Beta versions are available as well:

```
$ curl https://rclone.org/install.sh | sudo bash -s beta
```

Note: The installation script checks the version of `rclone` installed first, and skips downloading if the current version is already up-to-date.

Linux installation from pre-compiled binary

1. Fetch and unpack the binary:

   ```
   $ curl -O https://downloads.rclone.org/rclone-current-linux-amd64.zip
   unzip rclone-current-linux-amd64.zip
   cd rclone-*-linux-amd64
   ```

2. Copy the binary file to a sensible location:

   ```
   $ sudo cp rclone /usr/bin/
   sudo chown root:root /usr/bin/rclone
   sudo chmod 755 /usr/bin/rclone
   ```

3. Install the documentation:

   ```
   $ sudo mkdir -p /usr/local/share/man/man1
   sudo cp rclone.1 /usr/local/share/man/man1/
   sudo mandb
   ```

4. Run `rclone config` to set up:

   ```
   $ rclone config
   ```

macOS installation from pre-compiled binary

1. Download the `rclone` package:

   ```
   $ cd && curl -O https://downloads.rclone.org/rclone-current-osx-amd64.zip
   ```

2. Extract the downloaded file and `cd` to the extracted folder:

   ```
   $ unzip -a rclone-current-osx-amd64.zip && cd rclone-*-osx-amd64
   ```

3. Move `rclone` to your `$PATH` and enter your password when prompted:

   ```
   $ sudo mkdir -p /usr/local/bin
   sudo mv rclone /usr/local/bin/
   ```

Tip: The `mkdir` command is safe to run, even if the directory exists.

4. Remove the leftover files.

```
$ cd .. && rm -rf rclone-*-osx-amd64 rclone-current-osx-amd64.zip
```

5. Run `rclone config` to set up:

```
$ rclone config
```

Configure access to IBM COS

1. Run `rclone config` and select `n` for a new remote.

```
$ No remotes found - make a new one
   n) New remote
   s) Set configuration password
   q) Quit config
   n/s/q> n
```

2. Enter the name for the configuration:

```
$ name> <YOUR NAME>
```

3. Select “s3” storage.

```
$ Choose a number from below, or type in your own value
   1 / Alias for a existing remote
     \ "alias"
   2 / Amazon Drive
     \ "amazon cloud drive"
   3 / Amazon S3 Complaint Storage Providers (Dreamhost, Ceph, Minio, IBM COS)
     \ "s3"
   4 / Backblaze B2
     \ "b2"
[snip]
   23 / http Connection
     \ "http"
   Storage> 3
```

4. Select IBM COS as the S3 Storage Provider.

```
$ Choose the S3 provider.
Enter a string value. Press Enter for the default ("")
Choose a number from below, or type in your own value
   1 / Amazon Web Services (AWS) S3
     \ "AWS"
   2 / Ceph Object Storage
     \ "Ceph"
   3 / Digital Ocean Spaces
     \ "Digital Ocean"
   4 / Dreamhost DreamObjects
     \ "Dreamhost"
   5 / IBM COS S3
     \ "IBMCOS"
[snip]
     Provider>5
```

5. Enter **False** to enter your credentials.

```
$ Get AWS credentials from the runtime (environment variables or EC2/ECS meta data if no env vars).
Only applies if access_key_id and secret_access_key is blank.
Enter a boolean value (true or false). Please Enter for the default ("false").
Choose a number from below, or type in your own value
   1 / Enter AWS credentials in the next step
```

```
     \ "false"
   2 / Get AWS credentials from the environment (env vars or IAM)
     \ "true"
   env_auth>false
```

6. Enter the Access Key and Secret.

```
$ AWS Access Key ID - leave blank for anonymous access or runtime credentials.
  access_key_id> <>
AWS Secret Access Key (password) - leave blank for anonymous access or runtime credentials.
  secret_access_key> <>
```

7. Specify the endpoint for IBM COS. For Public IBM COS, choose from the provided options. For more information about endpoints, see Endpoints and storage locations.

```
$ Endpoint for IBM COS S3 API.
Choose a number from below, or type in your own value
   1 / US Cross Region Endpoint
     \ "s3.us.cloud-object-storage.appdomain.cloud"
   2 / US Cross Region Dallas Endpoint
     \ "s3-api.dal.us-geo.objectstorage.s3.us-south.cloud-object-storage.appdomain.cloud.net"
   3 / US Cross Region Washington DC Endpoint
     \ "s3-api.wdc-us-geo.objectstorage.s3.us-south.cloud-object-storage.appdomain.cloud.net"
   4 / US Cross Region San Jose Endpoint
     \ "s3-api.sjc-us-geo.objectstorage.s3.us-south.cloud-object-storage.appdomain.cloud.net"
   5 / US Cross Region Private Endpoint
     \ "s3-api.us-geo.objectstorage.service.networklayer.com"
[snip]
   34 / Toronto Single Site Private Endpoint
     \ "s3.tor01.objectstorage.service.networklayer.com"
endpoint>1
```

8. Specify an IBM COS Location Constraint. The location constraint must match the endpoint. For more information about endpoints, see Endpoints and storage locations.

```
$    1 / US Cross Region Standard
     \ "us-standard"
   2 / US Cross Region Vault
     \ "us-vault"
   3 / US Cross Region Cold
     \ "us-cold"
   4 / US Cross Region Flex
     \ "us-flex"
   5 / US East Region Standard
     \ "us-east-standard"
[snip]
  32 / Toronto Flex
  \ "tor01-flex"
  location_constraint>1
```

9. Specify an ACL. Only `public-read` and `private` are supported.

```
$ Canned ACL used when creating buckets or storing objects in S3.
Choose a number from below, or type in your own value
   1 "private"
   2 "public-read"
acl>1
```

10. Review the displayed configuration and accept to save the “remote” then quit. The config file should look like this

```
$ [YOUR NAME]
type = s3
Provider = IBMCOS
access_key_id = xxx
secret_access_key = yyy
endpoint = s3.us.cloud-object-storage.appdomain.cloud
location_constraint = us-standard
acl = private
```

Command reference

Create a bucket

```
$ rclone mkdir RemoteName:newbucket
```

List available buckets

```
$ rclone lsd RemoteName:
```

List contents of a bucket

```
$ rclone ls RemoteName:newbucket
```

Copy a file from local to remote

```
$ rclone copy /Users/file.txt RemoteName:newbucket
```

Copy a file from remote to local

```
$ rclone copy RemoteName:newbucket/file.txt /Users/Documents/
```

Delete a file on remote

```
$ rclone delete RemoteName:newbucket/file.txt
```

List Commands

There are several related list commands:

- `ls` to list size and path of objects only
- `lsl` to list modification time, size, and path of objects only
- `lsd` to list directories only
- `lsf` to list objects and directories in easy to parse format
- `lsjson` to list objects and directories in JSON format

`rclone sync`

The `sync` operation makes the source and destination identical, and modifies the destination only. Syncing doesn’t transfer unchanged files, testing by size and modification time or MD5SUM. Destination is updated to match source, including deleting files if necessary.

Important: Since this operation can cause data loss, test first with the `--dry-run` flag to see exactly what would be copied and deleted.

Files in the destination aren't deleted if there are errors at any point.

The *contents* of the directory are synced, not the directory itself. When `source:path` is a directory, it’s the contents of `source:path` that are copied, not the directory name and contents. For more information, see the extended explanation in the `copy` command.

If `dest:path` doesn’t exist, it is created and the `source:path` contents go there.

```
$ rclone sync source:path dest:path [flags]
```

Using `rclone` from multiple locations at the simultaneously

You can use `rclone` from multiple places simultaneously if you choose different subdirectory for the output:

```
$ Server A> rclone sync /tmp/whatever remote:ServerA
Server B> rclone sync /tmp/whatever remote:ServerB
```

If you `sync` to the same directory than you can use `rclone copy`, otherwise the two processes might delete each other's others files:

```
$ Server A> rclone copy /tmp/whatever remote:Backup
Server B> rclone copy /tmp/whatever remote:Backup
```

`--backup-dir=DIR`

When using `sync`, `copy` or `move` any files that would have been overwritten or deleted are moved in their original hierarchy into this directory.

If `--suffix` is set, then the moved files have the suffix added to them. If there is a file with the same path (after the suffix has been added) in the directory, it is overwritten.

The remote in use must support server-side move or copy and you must use the same remote as the destination of the sync. The backup directory must not overlap the destination directory.

```
$ rclone sync /path/to/local remote:current --backup-dir remote:old
```

will `sync` `/path/to/local` to `remote:current`, but for any files that would have been updated or deleted will be stored in `remote:old`.

If running `rclone` from a script you might want to use today's date as the directory name passed to `--backup-dir` to store the old files, or you might want to pass `--suffix` with today's date.

`rclone` daily sync

Scheduling a backup is important to automating backups. Depending on your platform depends on how you do this. Windows can use Task Scheduler while MacOS and Linux can use crontabs.

Syncing a Directory

`Rclone` syncs a local directory with the remote container, storing all the files in the local directory in the container. `Rclone` uses the syntax, `rclone sync source destination`, where `source` is the local folder and `destination` is the container within your IBM COS.

```
$ rclone sync /path/to/my/backup/directory RemoteName:newbucket
```

You might already have a destination that is created, but if you don't then you can create a new bucket by using the steps above.

Scheduling a Job

Before scheduling a job, make sure that you have done your initial upload and it has completed.

Windows

1. Create a text file that is called `backup.bat` somewhere on your computer and paste in the command you used in the section about syncing a directory. Specify the full path to the `rclone.exe` and don't forget to save the file.

   ```
   $  C:\full\path\to\rclone.exe sync "C:\path\to\my\backup\directory" RemoteName:newbucket
   ```

2. Use `schtasks` to schedule a job. This utility takes a number of parameters.

- /RU – the user to run the job as. This is needed if the user you want to use is logged out.
- /RP – the password for the user.
- /SC – set to DAILY
- /TN – the name of the job. Call it backup
- /TR – the path to the backup.bat file you created.
- /ST – the time to start the task. This is in the 24-hour time format. 01:05:00 is 1:05 AM. 13:05:00 would be 1:05 PM.

   ```
   $ schtasks /Create /RU username /RP "password" /SC DAILY /TN Backup /TR C:\path\to\backup.bat /ST 01:05:00
   ```

Mac and Linux

1. Create a text file called `backup.sh` somewhere on your computer, and paste the command that you used in the section syncing a Directory. It looks something like the following. Specify the full path to the `rclone` executable and don't forget to save the file.

   ```
   $ #!/bin/sh
   ```

```
/full/path/to/rclone sync /path/to/my/backup/directory RemoteName:newbucket
```

2. Make the script executable with `chmod`.

   ```
   $ chmod +x backup.sh
   ```

3. Edit crontabs.

   ```
   $ sudo crontab -e
   ```

4. Add an entry to the bottom of the crontabs file. Crontabs are straight forward: the first five fields describe in order minutes, hours, days, months, and weekdays. Using * denotes all. To make the `backup.sh` run at Daily at 1:05 AM, use something that looks like this:

   ```
   $ 5 1 * * * /full/path/to/backup.sh
   ```

5. Save the crontabs and you're ready to go.

Security and compliance

Data security

IBM Cloud® Object Storage uses an innovative approach for cost-effectively storing large volumes of unstructured data that ensures security, availability, and reliability.

This level of security is accomplished by using Information Dispersal Algorithms (IDA) to separate data into unrecognizable “slices”. The system distributes these slices across a network of data centers, making transmission and storage of data inherently private and secure. No complete copy of the data resides in any single storage node, and only a subset of nodes needs to be available to fully retrieve the data on the network.

All data in IBM Cloud Object Storage is encrypted at rest. This technology individually encrypts each object by using per-object generated keys. These keys are secured and reliably stored by using the same Information Dispersal Algorithms that protect object data by using an All-or-Nothing Transform (AONT). Key data is impossible to recover, even if individual nodes or hard disks are compromised.

If it's necessary for a user to control encryption keys, root keys can be provided on a per-object basis that uses SSE-C, or a per-bucket basis that uses SSE-KP.

Storage can be accessed over HTTPS, and internally storage devices communicate with each other using TLS.

Credential and encryption key rotation

Credentials, such as HMAC and API keys, do not naturally expire. Over time, it is possible that employee turnover or an accidental mishandling of information can result in unintended or unwanted access to cloud resources. Following a rotation schedule can help to prevent this scenario. Read more about rotation of encryption keys and access credentials.

Access Control Lists

Access control lists (often referred to as ACLs) are an outdated method for controlling access to object storage resources. Some APIs exist for setting individual objects to a `public-read` status, but this is discouraged in favor of using IAM to allow unauthenticated access to an entire bucket, and using these buckets to serve any open data.

Data deletion

IBM Cloud Object Storage data is erasure coded and distributed to multiple individual storage devices in multiple data centers. When data is deleted, various mechanisms exist which prevent recovery or reconstruction of the deleted objects. Deletion of an object undergoes various stages. First, the metadata is marked to indicate the object is deleted, then, the data is removed. Eventually, deleted metadata is overwritten by a process of compaction and the deleted data blocks are overwritten with new data in the course of normal operations. As soon as the metadata is marked deleted, it is impossible to read an object remotely. IBM's provider-managed encryption and erasure coding prevents data (both before and after deletion) from being accessible from within individual data centers.

Note: Cross regional and regional resiliency buckets distribute information across multiple data centers. For single site resiliency, data is dispersed to the same number of storage devices but they are all located in the same data center.

Data can be made more secure by using one of several available methods to protect the encryption keys including SSE-C, Key Protect or Hyper Protect Crypto Services. Please visit the manage encryption topic to learn more about the encryption methods.

Tenant isolation

IBM Cloud Object Storage is a multi-tenant Object Storage product. If your workload requires dedicated or isolated storage, see IBM Cloud® for more information.

Compliance

IBM Cloud® Object Storage actively participates in several industry compliance programs.

Note: This feature is not currently supported in Object Storage for Satellite. Learn more.

An updated list of our compliance certifications can always be obtained by referencing the Data Processing and Protection Datasheet available from IBM Software Product Compatibility Reports.

Note: Clients are responsible for ensuring their own compliance with any applicable laws and regulations and are solely responsible for obtaining advice of competent legal counsel as to the identification and interpretation of any relevant laws and regulations that may affect the clients' business as well as any actions the clients may need to take to comply with such laws and regulations. IBM does not provide legal, accounting, or auditing advice or represent or warrant that its services or products will ensure that clients are in compliance with any law or regulation.

For a listing of available certificates and instructions on obtaining pertinent reports please visit IBM Cloud Compliance page or contact an IBM Sales representative.

International Organization for Standardization (ISO)

IBM Cloud Object Storage is certified for ISO 27001, ISO 27017, and ISO 27018.

For available certificates, please refer to listings titled "IBM Cloud Services (PaaS and SaaS) certificates" on the IBM Cloud Compliance page.

System and Organization Controls (SOC)

IBM Cloud Object Storage is certified for SOC 1 Type 2, SOC 2 Type 2, and SOC 3.

Payment Card Industry (PCI) data security standards

IBM Cloud Object Storage is compliant with the PCI data security standards.

HIPAA readiness

IBM Cloud Object Storage meets the required IBM controls that are commensurate with the Health Insurance Portability and Accountability Act of 1996 (HIPAA) Security and Privacy Rule requirements. These requirements include the appropriate administrative, physical, and technical safeguards required of Business Associates in 45 CFR Part 160 and Subparts A and C of Part 164. HIPAA readiness for IBM Cloud Object Storage applies to the following plan:

- IBM Cloud Object Storage – Standard pricing plan

Note: If you or your company is a covered entity as defined by HIPAA, you must enable the HIPAA Supported setting on your IBM Cloud account if you run sensitive workloads that are regulated under HIPAA and the HITECH Act. HIPAA support from IBM requires that you agree to the terms of the Business Associate Addendum (BAA) agreement with IBM for your IBM Cloud account.

After you enable HIPAA Supported setting in your IBM Cloud account, you cannot disable it. See IBM Cloud Docs: Enabling the HIPAA Supported setting for additional information.

General Data Protection Regulation (GDPR) readiness

Please visit IBM's commitment to GDPR readiness page to learn about IBM's GDPR readiness journey and our GDPR capabilities and offerings to support your compliance journey.

- IBM Data Processing Addendum (DPA)

Privacy shield

IBM Cloud Object Storage is privacy shield certified. For more information please visit IBM Privacy Shield Privacy Policy for Certified IBM Cloud Services.

European Union support

IBM Cloud Object Storage has additional controls in place in the European Union (EU) to ensure access to your object storage resources is restricted and controlled by the IBM Cloud support team in the EU region. Review Enabling the EU Supported setting for information on how to enable the setting for your IBM Cloud account. For information on the support process, visit Requesting support for resources in the European Union.

Note: If you enable EU Supported setting for your IBM Cloud account, ensure that IBM Cloud Object Storage resources (buckets) are created in the EU region. For more information, visit Endpoints and Storage locations.

Activity Tracker events

Use the IBM Cloud® Activity Tracker service to track how users and applications interact with IBM Cloud Object Storage (COS).

Note: This feature is not currently supported in Object Storage for Satellite. Learn more.

The IBM Cloud Activity Tracker service records user-initiated activities that change the state of a service in IBM Cloud. For more information, see IBM Cloud Activity Tracker.

By default, COS events that report on global actions such as creation of a bucket are collected automatically. You can monitor global actions through the Activity Tracker instance that is located in the Frankfurt location.

In IBM Cloud Object Storage, you can also monitor management events and COS data events.

- Collection of these events in your account is optional.
- You must configure each bucket to enable management events, or data events, or both.
- Each action that a user performs on a COS resource has a unique ID that is included in the event in the `responseData.requestId` field.

You can use this service to investigate abnormal activity and critical actions, and to comply with regulatory audit requirements. In addition, you can be alerted about actions as they happen. The events that are collected comply with the Cloud Auditing Data Federation (CADF) standard.

For guidance on how to use IBM Cloud® Activity Tracker with Object Storage see Tracking events on your IBM Cloud Object Storage buckets . Below we list all of the events that are available.

Management events

Management events are classified in the following categories:

- Global events
- Resource configuration events
- Bucket events
- Object events

Global events

The following table lists the COS actions that generate a global event. You can monitor this events through the Activity Tracker instance that is available in the Frankfurt location.

Action	Description
`cloud-object-storage.instance.list`	List the buckets in the service instance
`cloud-object-storage.bucket.create`	Create a bucket in the service instance
`cloud-object-storage.bucket.delete`	Delete a bucket in the service instance

Object Storage actions that generate global events

Resource configuration events

The following table lists the COS resource configuration events:

Action	Description
`cloud-object-storage.resource-configuration.read`	Read the resource configuration for the bucket
`cloud-object-storage.resource-configuration.update`	Update the resource configuration for the bucket

Resource Configuration events

Bucket events

The following table lists the COS bucket events:

Action	Description
`cloud-object-storage.bucket-cors.read`	Get the CORS configuration
`cloud-object-storage.bucket-cors.create`	Create the CORS configuration
`cloud-object-storage.bucket-cors.delete`	Delete the CORS configuration
`cloud-object-storage.bucket-lifecycle.read`	Get the bucket lifecycle configuration
`cloud-object-storage.bucket-lifecycle.create`	Create the bucket lifecycle configuration
`cloud-object-storage.bucket-lifecycle.delete`	Delete the bucket lifecycle configuration
`cloud-object-storage.bucket-acl.read`	Get the bucket *ACL*
`cloud-object-storage.bucket-acl.create`	Create the bucket *ACL*
`cloud-object-storage.bucket-crn.read`	Get the bucket CRN
`cloud-object-storage.bucket-location.read`	Get the bucket location
`cloud-object-storage.bucket-retention.read`	Get the bucket retention
`cloud-object-storage.bucket-retention.create`	Create the bucket retention
`cloud-object-storage.bucket-key-state.update`	Updating a Key Protect root encryption key
`cloud-object-storage.bucket-public-access-block.create`	Add a public ACL block configuration
`cloud-object-storage.bucket-public-access-block.read`	Read a public ACL block configuration
`cloud-object-storage.bucket-public-access-block.delete`	Delete a public ACL block configuration

Bucket events

For `cloud-object-storage.bucket-key-state.update` events, the following fields include extra information:

Field	Description
`requestData.eventType`	The type of lifecycle event that occurred, such as deletion, rotation, and so on
`requestData.requestedKeyState`	The the requested state of the key (enabled or disabled).
`requestData.requestKeyVersion`	The requested version of the key.
`requestData.bucketLocation`	The location of the bucket that uses the key.
`responseData.eventID`	The unique identifier associated with the key lifecycle event.
`responseData.adopterKeyState`	The current state the key (enabled or disabled).
`responseData.adopterKeyVersion`	The current version of the key.

Table 3a. Additional fields for bucket-key-state.update events

Object events

The following table lists the COS object events:

Action	Description
`cloud-object-storage.object-cors.read`	Get the CORS configuration
`cloud-object-storage.object-acl.read`	Get the object ACL
`cloud-object-storage.object-acl.create`	Create the object ACL
`cloud-object-storage.object-retention-legal-hold.list`	List the legal holds on the object
`cloud-object-storage.object-retention-legal-hold.update`	Add or remove object legal hold
`cloud-object-storage.object-retention.update`	Extend the retention time
`cloud-object-storage.object-expire.read`	Get when the object will expire

Object events

Data Events

Data events are classified in the following categories:

- Bucket access events
- Object access events
- Multipart events
- Bucket versioning events

Bucket access events

The following table lists the COS bucket access events:

Action	Description
`cloud-object-storage.bucket.list`	List the objects in the bucket
`cloud-object-storage.bucket-metadata.read`	Get the metadata for the bucket

Bucket access events

Object access events

The following table lists the COS object access events:

Action	Description
`cloud-object-storage.object-metadata.read`	Get the metadata for the object
`cloud-object-storage.object.read`	Read the object
`cloud-object-storage.object.create`	Create the object
`cloud-object-storage.object.delete`	Delete the object
`cloud-object-storage.objects.delete`	Delete multiple objects
`cloud-object-storage.object-batch.delete`	Delete an object in a batch

`cloud-object-storage.object-copy.read`	Read the source object to copy
`cloud-object-storage.object-copy.create`	Create the target object from the copy
`cloud-object-storage.object-restore.read`	Read the source object to restore
`cloud-object-storage.object-restore.create`	Create the target object from the restore

Object access events

If versioning is enabled for a bucket, then `target.versionId` will be present for operations that make use of object versions.

For `cloud-object-storage.object.delete` and `cloud-object-storage.object-batch.delete` events, the following fields include extra information:

Field	Description
`responseData.deleteMarker.created`	The object has been versioned and replaced with a delete marker.

Table 6a. Additional fields for deletion events

Multipart events

The following table lists the COS multipart events:

Action	Description
`cloud-object-storage.bucket-multipart.list`	List multipart uploads of objects in a bucket
`cloud-object-storage.object-multipart.list`	List parts of an object
`cloud-object-storage.object-multipart.start`	Initiate a multipart upload of an object
`cloud-object-storage.object-multipart.create`	Create a part of a multipart upload of an object
`cloud-object-storage.object-multipart.complete`	Complete a multipart upload of an object
`cloud-object-storage.object-multipart.delete`	Abort an incomplete multipart upload of an object

Multipart events

Bucket versioning events

The following table lists the COS versioning events:

Action	Description
`cloud-object-storage.bucket-versioning.create`	Enable versioning on a bucket
`cloud-object-storage.bucket-versioning.read`	Check versioning status of a bucket
`cloud-object-storage.bucket-versioning.list`	List versions of objects in a bucket

Versioning events

For `cloud-object-storage.bucket-versioning.create` events, the following fields include extra information:

Field	Description
`requestData.newValue.versioning.state`	The versioning state of the bucket (enabled or suspended).

Table 8a. Additional fields for bucket-versioning.create events

Viewing events

You can view the Activity Tracker events that are associated with your Object Storage instance by using [IBM Cloud Activity Tracker](#).

You can only provision 1 instance of the IBM Cloud Activity Tracker service per location.

To view events, you must identify the location where events are collected and available for monitoring. Then, you must access the web UI of the IBM Cloud Activity Tracker instance in that location. For more information, see [Launching the web UI through the IBM Cloud UI](#) .

Management events

Object Storage global events are forwarded to the IBM Cloud Activity Tracker service instance that is located in Frankfurt.

All other Object Storage management events are forwarded to the IBM Cloud Activity Tracker instance that is associated with the bucket.

To view events, you must access the web UI of the IBM Cloud Activity Tracker instance in the location that is associated with the bucket.

Data events

Object Storage data events are forwarded to the IBM Cloud Activity Tracker instance that is associated with the bucket.

To view events, you must access the web UI of the IBM Cloud Activity Tracker instance in the location that is associated with the bucket.

Analyzing events

Identifying the COS instance ID that generates the event

In the IBM Cloud, you can have 1 or more COS instances.

To quickly identify the COS instance ID in your account that has generated an event, check the field `responseData.serviceInstanceId` that is set in the `responseData` field.

Identifying the bucket location

To quickly identify the bucket location, check the field `responseData.bucketLocation` that is set in the `responseData` field.

Getting the unique ID of a request

Each action that a user performs on a COS resource has a unique ID.

To get the unique ID of a request to a COS resource, check the field `responseData.requestId` that is set in the `responseData` field.

Getting all events for a multipart upload operations

When you upload a large object by using *multipart upload operations*, each operation generates an event. In each event, the field `responseData.uploadId` is set to the same value.

To search for all events that are part of a multipart upload operation, you can search for a specific `responseData.uploadId` value.

Getting all events that are generated for a restore request

A request to restore an object from an archive generates multiple events in COS:

1. A read action of the source object. This action generates an event with action **cloud-object-storage.object-restore.read**.
2. A create action of the object into a bucket. This action generates an event with action **cloud-object-storage.object-restore.create**.

You can use the `responseData.requestId` field to identify the events that are generated when you restore an object.

Getting all events that are generated for copying an object from one bucket to another

A request to copy an object from one bucket to a different one generates multiple events in COS:

1. A read action of the source object. This action generates an event with action **cloud-object-storage.object-copy.read**.
2. A create action of the object into the new bucket. This action generates an event with action **cloud-object-storage.object-copy.create**.

To collect and monitor all events that report on a copy action across buckets, consider configuring each bucket to collect and forward events to the same Activity Tracker instance in your account.

- If one bucket is not enabled to collect management and data events, you will not receive the event that reports any copy action on that bucket.
- If you configure different Activity Tracker instances for each bucket, you will have one event in 1 instance and the other event in a different instance.

You can use the `responseData.requestId` field to identify the events that are generated when you copy an object from one bucket to another.

Getting the details of a firewall update

Updating a bucket's firewall will generate a `cloud-object-storage.resource-configuration.update` event.

To get the details of what was changed, check for fields `requestData.allowedIp`, `requestData.deniedIp`, and `requestData.allowedNetworkTypes` that appear in the `requestData` field.

IAM and Activity Tracker actions by API

Find detailed information on IBM Cloud events from IAM and IBM Cloud® Activity Tracker actions, listed here by API method.

Resource Configuration API

The first table details the API for configuring IBM Cloud® Object Storage buckets: COS Resource Configuration API

- Note the endpoint URL: `https://config.cloud-object-storage.cloud.ibm.com/v1`

Action	Method	IAM Action	Activity Tracker action
Returns metadata for the specified bucket	`GET {endpoint}/b/{bucket}`	`cloud-object-storage.bucket.list_bucket_crn`, `cloud-object-storage.bucket.get_firewall`, `cloud-object-storage.bucket.get_activity_tracking`, `cloud-object-storage.bucket.get_metrics_monitoring`	`cloud-object-storage.resource-configuration.read`
Make changes to a bucket's configuration	`PATCH {endpoint}/b/{bucket}`	`cloud-object-storage.bucket.put_firewall`, `cloud-object-storage.bucket.put_activity_tracking`, `cloud-object-storage.bucket.put_metrics_monitoring`	`cloud-object-storage.resource-configuration.update`

RC API actions

S3 API

The next table describes the API for reading and writing objects as defined in the COS Compatibility S3 API

- Note that the endpoint URL for S3 operations differs by region: Endpoints

Action	Method	IAM Action	Activity Tracker action
List buckets	`GET {endpoint}/`	`cloud-object-storage.account.get_account_buckets`	`cloud-object-storage.instance.list`
Create a bucket	`PUT {endpoint}/{bucket}`	`cloud-object-storage.bucket.put_bucket`	`cloud-object-storage.bucket.create`
List objects	`GET {endpoint}/{bucket}`	`cloud-object-storage.bucket.get`	`cloud-object-storage.bucket.list`
Check a bucket's headers	`HEAD {endpoint}/{bucket}`	`cloud-object-storage.bucket.head`, `cloud-object-storage.bucket.list_crk_id`	`cloud-object-storage.bucket-metadata.read`
Delete a bucket	`DELETE {endpoint}/{bucket}`	`cloud-object-storage.bucket.delete_bucket`	`cloud-object-storage.bucket.delete`
Upload an object	`PUT {endpoint}/{bucket}/{key}`	`cloud-object-storage.object.put`	`cloud-object-storage.object.create`
Download an object	`GET {endpoint}/{bucket}/{key}`	`cloud-object-storage.object.get`	`cloud-object-storage.object.read`
Check an object's headers	`HEAD {endpoint}/{Bucket}/{key}`	`cloud-object-storage.object.head`	`cloud-object-storage.object-metadata.read`

Delete an object	`DELETE {endpoint}/{bucket}/{key}`	`cloud-object-storage.object.delete`	`cloud-object-storage.object.delete`
Add a CORS configuration	`PUT {endpoint}/{bucket}?cors`	`cloud-object-storage.bucket.put_cors`	`cloud-object-storage.bucket-cors.create`
Read a CORS configuration	`GET {endpoint}/{bucket}?cors`	`cloud-object-storage.bucket.get_cors`	`cloud-object-storage.bucket-cors.read`
Delete a CORS configuration	`DELETE {endpoint}/{bucket}?cors`	`cloud-object-storage.bucket.delete_cors`	`cloud-object-storage.bucket-cors.delete`
Add/edit a bucket's lifecycle configuration	`PUT {endpoint}/{bucket}?lifecycle`	`cloud-object-storage.bucket.put_lifecycle`	`cloud-object-storage.bucket-lifecycle.create`
Read a bucket's lifecycle configuration	`GET {endpoint}/{bucket}?lifecycle`	`cloud-object-storage.bucket.get_lifecycle`	`cloud-object-storage.bucket-lifecycle.read`
Delete a bucket's lifecycle configuration	`DELETE {endpoint}/{bucket}?lifecycle`	`cloud-object-storage.bucket.put_lifecycle`	`cloud-object-storage.bucket-lifecycle.delete`
Add/edit/remove a bucket's Immutable Storage policy	`PUT {endpoint}/{bucket}?protection`	`cloud-object-storage.bucket.put_protection`	`cloud-object-storage.bucket-retention.create`
Read a bucket's Immutable Storage policy	`GET {endpoint}/{bucket}?protection`	`cloud-object-storage.bucket.get_protection`	`cloud-object-storage.bucket-retention.read`
Initiate a multipart upload	`POST {endpoint}/{bucket}/{key}?uploads`	`cloud-object-storage.object.post_initiate_upload`	`cloud-object-storage.object-multipart.start`
Upload a part	`PUT {endpoint}/{bucket}/{key}?uploadId={uploadId}&partNumber={partNumber}`	`cloud-object-storage.object.put_part`	`cloud-object-storage.object-multipart.create`
Complete a multipart upload	`POST {endpoint}/{bucket}/{key}?uploadID={uploadId}`	`cloud-object-storage.object.post_initiate_upload`	`cloud-object-storage.object-multipart.complete`
Add a public ACL block configuration	`PUT {endpoint}/{bucket}?publicAccessBlock`	`cloud-object-storage.bucket.put_public_access_block`	`cloud-object-storage.bucket-public-access-block.create`
Read a public ACL block configuration	`GET {endpoint}/{bucket}?publicAccessBlock`	`cloud-object-storage.bucket.get_public_access_block`	`cloud-object-storage.bucket-public-access-block.read`
Delete a public ACL block configuration	`DELETE {endpoint}/{bucket}?publicAccessBlock`	`cloud-object-storage.bucket.delete_public_access_block`	`cloud-object-storage.bucket-public-access-block.delete`

S3 API actions

Understanding data portability

Data portability involves a set of tools and procedures that enable you to export the digital artifacts that are needed to implement similar workload and data processing on different service providers, or on-premises software. It includes procedures for copying and storing your service content, including the related configuration that is used by the service to store and process the data on your own location.

Responsibilities

IBM Cloud services provide interfaces and instructions to guide you to copy and store your service content, including the related configuration, on your own selected location.

You are responsible for the use of the exported data and configuration for data portability to other infrastructures, which includes:

- The planning and execution for setting up alternative infrastructure on different cloud providers, or on-premises software that provide similar capabilities to the IBM services.
- The planning and execution for the porting of the required application code on the alternative infrastructure, including the adaptation of your application code, deployment automation, and more.
- The conversion of the exported data and configuration to the format that's required by the alternative infrastructure and adapted applications.

For more information about your responsibilities when using IBM Cloud® Object Storage, see Shared responsibilities for IBM Cloud Object Storage.

Data export procedures

IBM Cloud Object Storage provides mechanisms to export your content that is uploaded, stored, and processed using this service.

About the IBM Cloud Object Storage S3 API documents the commands to interact with your data held in Object Storage buckets.

Further, more detailed information and examples for commands to extract your information for detailed topics are below:

- Bucket operations provides detailed examples about bucket operations.
- Object operations discusses object operations.
- IBM Cloud Object Storage S3 API discusses s3 API detail and how to use the methods.

Moreover IBM Cloud Object Storage provides mechanisms to export settings and configuration used to process the customer's content.

- COS Resource Configuration API details bucket configuration.

Exported data formats

As IBM Cloud Object Storage is a data architecture for storing unstructured data securely, without a schema. You must manage how you store data in your buckets.

The format of the data output exported using the methods outlined in the Data export procedures are also covered in the IBM Cloud Object Storage S3 API documentation.

Data ownership

All exported data is classified as your content. Apply your full ownership and licensing rights, as stated in the IBM Cloud Service Agreement.

Understanding high availability and disaster recovery for IBM Cloud® Object Storage

High availability (HA) is the ability for a service to remain operational and accessible in the presence of unexpected failures. *Disaster recovery* is the process of recovering the service instance to a working state.

IBM Cloud Object Storage is a global service that allows you to configure storage data resiliency while maintaining high availability. For more information, see Service Level Agreement (SLA). You can also find the available region and data center locations in the Service and infrastructure availability by local documentation.

High availability architecture

Object Storage is a global service and you have a choice to configure the storage resiliency. A bucket's resiliency is defined by the endpoint that is used to create it, i.e., Cross Region, Regional, and Single Site.

- Cross-Region resiliency will spread your data across several metropolitan areas
- Regional resiliency will spread data across a single metropolitan area
- Single Data Center resiliency spreads data across multiple appliances within a single data center

Regional and Cross Region buckets can maintain availability during a site or zone outage without any configuration changes required so it is recommended to use these storage bucket resiliency settings when configuring your workloads for high availability. Data that is stored in a single site is still distributed across many physical storage appliances, but is contained within a single data center without any zonal support.

High availability features

Object Storage provides the following capabilities to help you plan for high availability in the event of an outage:

Feature	Description	Consideration
Storage Bucket Resiliency	Ability to configure specific resiliency choice for customer data.	Object Storage buckets that are created at a regional endpoint distribute data across three or more zones that are contained in a metro area. Any one of these zones can suffer an outage or even destruction without impacting availability. Buckets that are created at a cross-region endpoint distribute data across three regions in a geographical location. Any one of these regions can suffer an outage or even destruction without impacting availability. Requests are routed to the nearest cross-region metropolitan area by using Global Server Load Balancing (GSLB). Refer to Endpoints and storage locations for more information.
Replication	Replication copies newly created objects and object updates from a source bucket to a target bucket and allows you to define rules for automatic, asynchronous copying of objects.	To ensure that you have a backup copy available in the event of a disaster, it is recommended that replication be configured and setup. Learn more about Tracking replication events.

HA features for Object Storage

Disaster recovery architecture

Cross-region, regional, and single site buckets offer varying levels of tolerance against specific disaster scenarios. Choose the right resiliency model for your bucket that aligns with your business's disaster recovery requirements. For many disaster scenarios at the data center or regional level, IBM plans to target a recovery time of the service and associated content in less than 24 hours with an RPO of 1 hour.

Additional optional architectures can be implemented by the customer to improve recovery times.

For example, to recover from the unlikely event of a complete COS regional outage where restoration of the original data is not possible, a duplicated bucket could be created in an alternate region. Waiting for IBM Cloud to recover an affected region or service is also a valid path, but remember it can take many hours or longer and there may be data loss depending on the disaster scenario.

The duplicated bucket can be configured to mirror the production bucket but with updated references to regional services. For instance, if using Key Protect, a duplicated bucket in Madrid should reference root keys that are stored in a Madrid Key Protect Instance. In the case of a disaster scenario, where waiting for IBM to fully recover the region is not possible, customers can repopulate this duplicated bucket with a backup copy of the original data in the source bucket. Alternatively, customers can set up replication rules prior to any outage to keep data in-sync between the source bucket and the duplicated bucket. For the highest level of resiliency, such a duplicated bucket should be created before the fact (before any potential disaster) and kept in-sync with the source bucket that uses replication. Use of the object replication feature should be considered along with the resiliency model of your source bucket and your overall business disaster recovery objectives.

Plan for the recovery into a recovery region. The replicated bucket should align with the workload disaster recovery approaches within IBM Cloud. If the disaster does not impact the production source bucket's configuration or availability (for example just data loss), it may be possible for a customer to repair the data in the source bucket in place. In the case that failover to a replicated bucket is required, the client application will need to be reconfigured to call the endpoint of the target-replicated bucket.

Disaster recovery features

IBM COS provides the following disaster recovery features that can be configured by customers:

Feature	Description	Consideration
Object Replication	Replication copies newly created objects and object updates from a source bucket to a target bucket and allows you to define rules for automatic, asynchronous copying of objects.	To ensure that you have a second copy available in the event of a disaster, you can configure replication between a production bucket and a target recovery bucket. Depending on your business's resiliency requirements, replication may not be required if using Cross-Region or Regional buckets. Learn more about Tracking replication events.
Object Versioning	Enable object versioning to maintain previous versions of objects that can be restored in the case of data corruption or deletion.	Customers can enable object versioning on buckets and restore old versions in the event of data corruption. The bucket must be available to perform version recovery. Learn More
Object Lock	Object Lock prevents deletion of object versions during a specified retention period.	Enable object lock to protect against accidental or unauthorized object deletions or overwrites. Ensure that safe object versions are available for recovery. Learn More

DR features for Object Storage

Other disaster recovery options are created and supported by the customer.

Feature	Description	Consideration
Backup and restore	Use scripts or 3rd-party backup applications to back up data in source buckets to a recovery region.	Customer must host and manage any script or 3rd-party backup solution to back up data stored in COS buckets.

Customer DR features for Object Storage

Planning for DR

The DR steps must be practiced regularly. As you build your plan, consider the following failure scenarios and resolutions.

Failure	Resolution
Hardware failure (single point)	Object Storage buckets are resilient from single point hardware failures within a zone. No configuration required.
Data center failure	Cross-region and regional COS buckets are resilient to individual data center failures. No configuration or failover is required by the customer. Customers with buckets in single data center zones can configure replication or use 3rd party backup solutions to ensure that a safe copy of data is available outside the zone. Waiting for IBM Cloud to recover an affected region or service is also a valid path, but remember it can take many hours or longer depending on the nature of the data center outage.
Data corruption	Use object versioning, object replication, or 3rd party back up solutions to ensure that uncorrupted versions of objects exist to recover from in the case of data corruption or accidental deletion.

Regional failure	Cross-region COS buckets are resilient to regional failures. Some integrated regional services like Key Protect may require additional failover steps for cross-region buckets. Customers with buckets in regional or single data center COS buckets should follow the disaster recovery steps above in the case of total regional failures. Waiting for IBM Cloud to recover an affected region or service is also a valid path, but remember it can take many hours or longer depending on the nature of the regional outage.

DR scenarios for Object Storage

Use of IBM Cloud Key Management Service for adding envelop encryption:

If you are using any other IBM Cloud service integration, for example IBM Cloud Key Management Service like Key Protect or Hyper Protect, to add envelop encryption, you will need to ensure that the appropriate configuration plan for key replica is used. This is essential when using a Cross Region configuration which ensures a replica key is available in the event of an outage. Refer to Key Protect documentation for [high availability and disaster recovery](#).

Your responsibilities for HA and DR

Responsibility	Description
Resiliency	Provision Object Storage buckets with the appropriate resiliency option, storage class, data locality, and optional configurations necessary for the specific workload and use case.
Data Backup	Ensure customer data backups if required as per your organization requirements.
Network	Monitor and manage non-IBM network resources to ensure appropriate access to IBM Cloud service endpoints including capacity and availability.
Using IBM Cloud KMS for adding envelop encryption	If you are using IBM Cloud Key Protect or Hyper Protect Crypto Services to add envelop encryption, ensure to review the respective High Availability and Disaster Recovery documentation to fully understand the implications. You may be required to use a key instance location that has a key replica which can be used in the event of a failover. Please also ensure to review the appropriate licensing and plan information.

Your responsibilities for HA and DR

To find out more about responsibility ownership between the customer and Object Storage, refer to [Your responsibilities when using Object Storage(/docs/cloud-object-storage?topic=cloud-object-storage-responsibilities)].

Recovery time objective (RTO) and recovery point objective (RPO)

IBM Cloud Object Storage offering has plans in place to provide for the recovery of both the Cloud Service, and the associated Content, which happens within hours in the event of a corresponding disaster.

Feature	RTO and RPO
Recover from hardware failure (single point)	RTO = 0, RPO = 0 for all resiliency models
Recover from data center outage	RTO = 0, RPO = 0 for Cross-Region and Regional resiliency models
Restore previous object version	RTO = seconds, RPO = near 0
Recover to bucket in separate region with active replication	RTO = minutes, scripting may improve time further and also consider time to adjust workloads to target the recovery bucket, , RPO = near 1 hour
Recover to new bucket in new region without active replication	RTO = minutes to days, consider the amount of time to reconfigure a new bucket and to adjust workloads to target the new bucket endpoint. Also consider time to populate the bucket with a copy of the original data. RPO is subject to the customer’s backup and recovery plan

RTO/RPO features for Object Storage

Change management

Change management includes tasks such as upgrades, configuration changes, and deletion. In order to ensure that users are given access as per role requirements, please review Getting Started with IAM.

It is recommended that you grant users and processes the IAM roles and actions with the least privilege that is required for their work. See How can I prevent accidental deletion of services?.

How IBM® helps ensure disaster recovery

IBM® takes specific recovery actions in the case of a disaster.

- Recovery from zone or regional failures
  In the event of a zone failure IBM Cloud will resolve the zone outage and when the zone comes back on-line, the global load balancer will resume sending API requests to the restored instance node without need for customer action.
- IBM® conducts annual tests of various disaster scenarios and continuously refines our recovery documentation based on findings that are found during these tests.
- 24 × 7 global support is available to customers with IBM® Subject Matter Experts who are on call to help in the case of a disaster.
  All IBM® Subject Matter Experts are trained annually on business continuity and disaster recovery policies and procedures to ensure preparedness in the event of a disaster.

How IBM maintains services

All upgrades follow the IBM service best practices and have a recovery plan and rollback process in-place. Regular upgrades for new features and maintenance occur as part of normal operations. Such maintenance can occasionally cause short interruption intervals that are handled by client availability retry logic. Changes are rolled out sequentially, region by region and zone by zone within a region. Updates are backed out at the first sign of a defect.

Complex changes are enabled and disabled with feature flags to control exposure.

Changes that impact customer workloads are detailed in notifications. For more information, see monitoring notifications and status for planned maintenance, announcements, and release notes that impact this service.

Frequently asked questions

FAQ - General

Frequently asked questions can produce helpful answers and insight into best practices for working with IBM Cloud® Object Storage.

Can I use AWS S3 SDKs with IBM Cloud Object Storage?

IBM Cloud Object Storage supports the most commonly used subset of Amazon S3 API operations. IBM makes a sustained best effort to ensure that the IBM Cloud Object Storage APIs stay compatible with the industry standard S3 API. IBM Cloud Object Storage also produces several native core COS SDKs that are derivatives of publicly available AWS SDKs. These core COS SDKs are explicitly tested on each new IBM Cloud Object Storage upgrade. When using AWS SDKs, use HMAC authorization and an explicit endpoint. For details, see About IBM COS SDKs.

Does data consistency in Object Storage come with a performance impact?

Consistency with any distributed system comes with a cost, because the efficiency of the IBM Cloud Object Storage dispersed storage system is not trivial, but is lower compared to systems with multiple synchronous copies.

Aren't there performance implications if my application needs to manipulate large objects?

For performance optimization, objects can be uploaded and downloaded in multiple parts, in parallel.

What is the difference between 'Class A' and 'Class B' requests?

'Class A' requests are operations that involve modification or listing. This includes creating buckets, uploading or copying objects, creating or changing configurations, listing buckets, and listing the contents of buckets.'Class B' requests are those related to retrieving objects or their associated metadata/configurations from the system. There is no charge for deleting buckets or objects from the system.

Can you confirm that Object Storage is ‘immediately consistent’, as opposed to ‘eventually consistent’?

Object Storage is ‘immediately consistent’ for data and ‘eventually consistent’ for usage accounting.

Can a web browser display the content of files stored in IBM Cloud Object Storage?

Web browsers can display web content in IBM Cloud Object Storage files, using the COS endpoint as the file location. To create a functioning website, however, you need to set up a web environment; for example, elements such as a CNAME record. IBM Cloud Object Storage does not support automatic static website hosting. For information, see Static websites and this tutorial.

Why do `CredentialRetrievalError` occur while uploading data to Object Storage or while retrieving credentials?

`CredentialRetrievalError` can occur due to the following reasons:

- The API key is not valid.
- The IAM endpoint is incorrect.

However, if the issue persists, contact IBM customer support.

How do I ensure communication with Object Storage?

You can check the communication with Object Storage by using one of the following:

- Use a `COS API HEAD` call to a bucket that will return the headers for that bucket. See api-head-bucket.
- Use SDK : See `headbucket` property.

Why can I not create or delete a service instance?

A user is required to have have at a minimum the platform role of `editor` for all IAM enabled services, or at least for Cloud Object Service. For more information, see the IAM documentation on roles .

What is the maximum number of characters that can be used in a key, or Object name?

Keys have a 1024-character limit.

How can I track events in Object Storage?

The Object Storage Activity Tracker service records user-initiated activities that change the state of a service in Object Storage. For details, see IBM Cloud Activity Tracker.

What are some tools unable to render object names?

Object names that contain unicode characters that are not allowed by the XML standard will result in "Malformed XML" messages. For more information, see the XML reference documentation.

Is Object Storage HIPAA compliant to host PHI data?

Yes, Object Storage is HIPAA compliant.

Is there any option in Object Storage to enable `accelerate data transfer`?

Object Storage offers **Aspera** service for high speed data transfer.

How can I access a private COS endpoint in a data center from another date center?

Use Object Storage Direct Link Connection to create a global direct link.

How can I monitor Object Storage resources?

Use the Activity Tracker service to capture and record Object Storage activities and monitor the activity of your IBM Cloud account. Activity Tracker is used to track how users and applications interact with Object Storage.

How can I move data into the archive tier?

You can archive objects using the web console, REST API, and third-party tools that are integrated with IBM Cloud Object Storage. For details, see COS Archive.

Can I use the same Object Storage instance across multiple regions?

Yes, the Object Storage instance is a global service. Once an instance is created, you can choose the region while creating the bucket.

Is it possible to form a Hadoop cluster using Object Storage?

No, Object Storage is used for the object storage service. For a Hadoop cluster, you need the processing associated with each unit of storage. You may consider the Hadoop-as-a-Service setup.

Can I generate a "Pre-signed URL" to download a file and review?

A Pre-signed URL is not generated using the IBM Cloud UI; however, you can use CyberDuck to generate the “pre-signed URL”. It is free.

How can I generate a Auth Token using the IAM API Key using REST?

For more information on working with the API, see Creating IAM token for API Key and Configuration Authentication.

What are the libraries that the Object Storage SDK supports?

Object Storage provides SDKs for Java, Python, NodeJS, and Go featuring capabilities to make the most of IBM Cloud Object Storage. For information about the features supported by each SDK, see the feature list.

When a file is uploaded to a cross region bucket using the ‘us-geo’ endpoint, how long is the delay before the file is available at the other US sites?

The data are spread immediately without delay and the uploaded files are available once the write is successful.

Why am I unable to delete a Object Storage instance?

It isn't possible to delete an instance if the API key or Service ID being used is locked. You'll need to navigate in the console to **Manage** > **Access (IAM)** and unlock the API Key or Service ID. The error provided may seem ambiguous but is intended to increase security:

> An error occurred during an attempt to complete the operation. Try fixing the issue or try the operation again later. Description: 400

This is intentionally vague to prevent any useful information from being conveyed to a possible attacker. For more information on locking API keys or Service IDs, [see the IAM documentation](#).

How do I download the Root CA certificate for Object Storage?

Object Storage root CA certificates can be downloaded from https://www.digicert.com/kb/digicert-root-certificates.htm. Please download PEM or DER/CRT format from "DigiCert TLS RSA SHA256 2020 CA1" that is located under "Other intermediate certificates."

How to I find my current active Object Storage instance/resources?

Login to the IBM Cloud shell: https://cloud.ibm.com/shell and enter at the prompt `ibmcloud resource search "service_name:cloud-object-storage AND type:resource-instance"`.

The response you receive includes information for the name of your instance, location, family, resource type, resource group ID, CRN, tags, service tags, and access tags.

Does IBM Cloud Object Storage rate limit?

IBM Cloud Object Storage may rate-limit your workload based on its specific characteristics and current system capacity. Rate-limiting will be seen as a 429 or 503 response, in which case retries with exponential back-off are suggested.

What's the difference between the service and the deployable architecture for IBM Cloud® Object Storage? {: #faq-deployable-architecture faq}

The IBM Cloud® Object Storage service is a SaaS offering in the catalog. It displays in the Storage category on your Resource list. The deployable architecture can be configured, updated, monitored, and deployed across accounts by using IBM Cloud projects. The deployable architecture can be used to link together multiple architectures to create an end-to-end solution.

FAQ - Plans

Frequently asked questions can produce helpful answers and insight into best practices for working with IBM Cloud® Object Storage.

Which one of my instances uses a Lite plan?

An account is limited to a single instance of IBM Cloud Object Storage that uses a Lite plan. You can find this instance three different ways:

1. Navigate to the [catalog](#) and attempt to make a new Lite instance. An error will pop up prompting you to delete the existing instance, and provides a link to the current Lite instance.
2. Navigate to the storage section of the resource list, and click on any area of blank space to select an instance of Object Storage. An informational sidebar will appear and provide the plan name: either Lite or Standard.
3. Use the CLI to search for the resource:

```
$ ibmcloud resource search "service_name:cloud-object-storage AND 2fdf0c08-2d32-4f46-84b5-32e0c92fffd8"
```

How do I upgrade a service instance from a Lite Plan to a Standard Plan?

- Using the Console
    1. Upgrade your account to a [Pay-As-You-Go account](#).
    2. Go to the [IBM Cloud Object Storage console](#).
    3. Using the left navigation panel, select the name of the Cloud Object Storage instance you want to upgrade.
    4. Click `Plan` in the navigation menu, located after `Instance Usage`. The `Plan` tab for a Lite Plan instance displays a `Change Pricing Plan` section.
    5. Select the "Standard" plan and click `Save`.
- Using the CLI
    1. Use the plan ID for a standard Object Storage instance:

        744bfc56-d12c-4866-88d5-dac9139e0e5d

    2. Using the name of the instance that you are trying to upgrade (for example, to upgrade the instance “"My Object Storage"), issue the command:

        ```
        $ ic resource service-instance-update "My Object Storage" --service-plan-id 744bfc56-d12c-4866-88d5-dac9139e0e5d
        ```

    3. Use the plan ID for a standard Object Storage instance:

744bfc56-d12c-4866-88d5-dac9139e0e5d

4. Using the name of the instance that you are trying to upgrade (for example, to upgrade the instance “"My Object Storage"), issue the command:

```
$ ic resource service-instance-update "My Object Storage" --service-plan-id 744bfc56-d12c-4866-88d5-dac9139e0e5d
```

Can I create more than one Object Storage service with a Lite plan?

If you already have a Lite plan instance created, you may create other Standard plan instances, but only one Lite plan instance is allowed.-->

What if my Lite Plan instance is locked?

In cases where a Lite Plan instance has exceeded the size limit, your account is locked or deactivated.

- To continue using the service instance, follow the steps to upgrade it to a Standard plan . Effective December 15th, 2024, support will end for all Lite Plan instances. To avoid loss of data, Lite Plan users need to convert their Lite Plan instance to the Standard (paid) plan before that date.
- If necessary, you can create a support case to request that the Cloud Support team unlock the account and provide a one-time reactivation of the instance to allow time for you to convert the plan.

How does frequency of data access impact the pricing of Object Storage?

Storage cost for Object Storage is determined by the total volume of data stored, the amount of public outbound bandwidth used, and the total number of operational requests processed by the system. For details, see cloud-object-storage-billing.

What are the considerations for choosing the correct storage class in Object Storage?

You can choose the correct storage class based on your requirement. For details, see billing-storage-classes.

What is Free Tier?

Free Tier is a no-cost option that allows you to use Object Storage for free, within certain allowances, for 12 months. It enables you to easily evaluate and explore all the features of Object Storage without any upfront costs. To get Free Tier, you must create a Smart Tier bucket in any location, in an instance provisioned under the Standard plan.

What are the specific allowances included in the Free Tier?

Free Tier includes free monthly usage in the Smart Tier storage class under the Standard plan. Free Tier allowances include up to 5 GB of Smart Tier storage capacity, 2,000 Class A (PUT, COPY, POST, and LIST) requests, 20,000 Class B (GET and all others) requests, 10 GB of data retrieval, and 5GB of egress (public outbound bandwidth) each month.

When does Free Tier expire?

The Free Tier provides free usage for the specified allowances for 12 months from the date when the Object Storage instance was initially created.

What happens if I exceed the Free Tier usage limits or after the 12-month period ends?

If you exceed the Free Tier monthly allowances within the 12-month period, you are only charged for the portion above the allowance and only in the months when they are exceeded.

What happens after the 12-month Free Tier period ends?

Once the 12-month Free Tier period ends, you are charged at the standard pay-as-you-go rates (see pricing).

How can I transition from Free Tier to production use?

Free Tier enables you to seamlessly transition to production use when you are ready to scale up. No further action is needed. You are billed for any usage over the Free Tier usage allowances.

How are the Free Tier allowances applied across multiple Smart Tier buckets in my account?

The Free Tier limits apply to the total usage across all Smart Tier buckets in the Standard Plan.

How can I transition from my current Lite Plan instance to Free Tier?

There is no direct path to transition from the old Lite Plan to the Free Tier. First, upgrade your Lite Plan to a Standard plan. Then you can enable the Free Tier by either creating a Smart Tier bucket in the Standard plan or, if you already had a Smart Tier bucket in the Lite Plan, the Free Tier will apply to it once the Lite Plan is upgraded to the Standard plan.

FAQ - One Rate plans

Frequently asked questions can produce helpful answers and insight into best practices for working with IBM Cloud® Object Storage.

What is the difference between a Standard and One Rate plan?

- Standard plan is our most popular public Cloud pricing plan, that meets the requirements of majority of the enterprise workloads. The Standard plan is best suited for workloads that have large amount of storage and relatively small Outbound bandwidth (Outbound bandwidth < 20% of Storage capacity). The plan offers flexible choices for storage class based on data access patterns (lower the cost, the less frequently data is accessed). The Standard plan bills for every stored capacity ($/GB/month), Outbound bandwidth ($/GB), class A ($/1,000), class B ($/10,000) and retrieval ($/GB) metrics, where applicable.
- One Rate plan is suited for active workloads with large amounts of Outbound bandwidth (or varying Outbound bandwidth) as a percentage of their Storage capacity (Outbound bandwidth > 20% of Storage capacity). Typical workloads belong to large enterprises and ISV's which may have sub-accounts with multiple divisions/departments or end-users. The plan offers a predictable TCO with an all-inclusive flat monthly charge ($/GB/month) that includes capacity, and built-in allowances for Outbound bandwidth and Operational requests. The built-in allowances for Outbound bandwidth and Operational requests (Class A, Class B) depend on the monthly stored capacity. There is no data retrieval charge.

How are the allowance thresholds (for Outbound bandwidth, class A and class B) calculated for the One-Rate plan?

For each of the One-Rate plan pricing regions(North America, Europe, South America,and Asia Pacific), the total aggregated Storage capacity across all instances (within a region) is used to determine the allowance thresholds.

- Outbound bandwidth: No charge if Outbound bandwidth ≤ 100% of Storage capacity in GB, then list prices apply ($0.05/GBfor North America and Europe, $0.08/GB for South America and Asia Pacific). For example, for an account with aggregated monthly Storage capacity of 100 GB in North America, there are no Outbound bandwidth charges up to 100 GB of transferred data within that month.
- Class A: No charge if class A requests ≤ 100 x Storage capacity in GB, then list prices apply ($0.005/1000). For example, for an account with aggregated monthly Storage capacity of 100 GB in North America, there are no Outbound bandwidth charges up to 10,000 class A requests that month in North America.
- Class B: No charge for class B ≤ 1000 x Storage(GB), then list prices apply ($0.004/1000)For example, for an account with aggregated monthly Storage capacity of 100 GB in North America, there are no Outbound bandwidth charges up to 100,000 class A requests that month in North America.

Which storage classes are supported in the One-Rate plan?

There is only one storage class available in the One-Rate plan: One-Rate Active

What are the One-Rate pricing regions?

There are four One-Rate pricing regions: North America, Europe, South America and Asia Pacific. The following Regional and Single Sites are included in the four One-Rate pricing regions:

North America:

- Regional: `us-south`, `us-east`, `ca-tor`
- Single Sites: `mon01`, `sjc04`

Europe:

- Regional: `eu-gb`, `eu-de`
- Single Sites: `ams03`, `mil01`, `par01`

South America:

- Regional: `br-sao`

Asia Pacific:

- Regional: `au-syd`, `jp-osa`, `jp-tok`

- Single Sites: `che01`, `sng01`

Is the pricing different for the four One-Rate pricing regions?

The pricing rates are same for North America and Europe, similarly for South America and Asia Pacific. See [One Rate pricing plan details](#).

Are all Cloud Object Storage features available in the One-Rate Plan?

All Cloud Object Storage features (Versioning, Archive, Replication, WORM, Expiration, and so on) are available in the One-Rate Plan.

Is the One-Rate plan available in all Cloud Object Storage regions?

The One-Rate plan is available in all Cloud Object Storage Regional and Single sites.

Can I configure a lifecycle policy to archive or expire my objects in the One Rate Active buckets?

Yes, you can set a lifecycle policy to archive objects from the One Rate Active buckets to either Archive (restore ≤ 12 hours) or Accelerated Archive (restore ≤ 2 hours). Similarly, you can set expiration rules to expire objects based on the date of creation of the object, or a specific date.

What pricing rates will apply to objects archived from the One Rate Active buckets?

For Archive and Accelerated Archive, standard pricing applies based on the bucket location. For example, a bucket created in `us-south` will incur archive pricing for `us-south`. Similarly, a bucket in `ca-tor` will incur archive pricing for `ca-tor`.

Can I move my existing buckets from Standard plan to One-Rate plan?

Existing buckets in the Standard plan cannot be moved to the One-rate plan. Clients must first enroll in the One-Rate plan, create a new service instance and new buckets before data can be populated.

Can I upgrade my Cloud Object Storage Lite plan instance to One-Rate plan?

No, Lite Plan instances can only be upgraded to the Cloud Object Storage Standard plan.

Are there any minimum object size or minimum duration requirements for objects stored with the One-Rate plan?

There are no minimum object size or minimum duration requirements for the One-Rate plan.

What is the cost of data retrieval from One Rate Active buckets?

There is no data retrieval charge for the One Rate Active buckets.

What happens if I exceed my monthly allowance for Outbound bandwidth and Operational requests?

For any usage (Outbound bandwidth or Operational requests) that exceeds the allowance determined by aggregated monthly capacity, a small overage fee applies based on the One Rate pricing regions. See [One Rate pricing plan details](#).

Is the overage pricing tiered for Outbound bandwidth and Operational requests?

No, the overage pricing for the One Rate plan has flat rates regardless of excess usage. See [One Rate pricing plan details](#).

I already have a Cloud Object Storage Standard plan in my IBM Cloud account. Can I add a One Rate plan for my new workloads?

Yes, you can add a One Rate plan to your existing account in addition to the Standard plan. If you are a new to Cloud Object Storage, you can add either Standard or One Rate plan (or both) based on your workload requirements.

FAQ - Encryption

Frequently asked questions can produce helpful answers and insight into best practices for working with IBM Cloud® Object Storage.

What types of authentication can I use to access IBM Cloud® Object Storage?

You can use an OAuth 2 token or an HMAC key for authentication. The HMAC key can be used for S3-compatible tools such as `rclone`, `Cyberduck`, and others.

- For instructions to obtain an OAuth token, see Generating an IBM Cloud IAM token by using an API key .
- For instructions to obtain the HMAC credentials, see Using HMAC Credentials.

Also, see API Key vs HMAC.

Does Object Storage provide encryption at rest and in motion?

Yes. Data at rest is encrypted with automatic provider-side Advanced Encryption Standard (AES) 256-bit encryption and the Secure Hash Algorithm (SHA)-256 hash. Data in motion is secured by using the built-in carrier grade Transport Layer Security/Secure Sockets Layer (TLS/SSL) or SNMPv3 with AES encryption.

If you want more control over encryption, you can make use of IBM Key Protect to manage generated or "bring your own" keying. For details, see Key-protect COS Integration.

Is there additional encryption processing if a customer wants to encrypt their data?

Server-side encryption is always on for customer data. Compared to the hashing required in S3 authentication and the erasure coding, encryption is not a significant part of the processing cost of Object Storage.

Does Object Storage encrypt all data?

Yes, Object Storage encrypts all data.

How do I encrypt my data?

1. Go to the IBM Cloud Object Storage documentation for managing encryption to research the encryption topic.
2. Choose between IBM® Key Protect for IBM Cloud® and Hyper Protect Crypto Services for your encryption needs.
3. Remember that customer-provided keys are enforced on objects.
4. Use IBM Key Protect or Hyper Protect Crypto Services to create, add, and manage keys, which you can then associate with your instance of IBM Cloud Object Storage.
5. Grant service authorization
   a. Open your IBM Cloud dashboard.
   b. From the menu bar, click **Manage > Access**.
   c. In the side navigation, click **Authorizations**.
   d. Click **Create authorization**.
   e. In the **Source service** menu, select **Cloud Object Storage**.
   f. In the **Source service instance** menu, select the service instance to authorize.
   g. In the **Target service** menu, select **IBM Key Protect** or **Hyper Protect Crypto Services**.
   h. In the **Target service instance** menu, select the service instance to authorize.
   i. Enable the **Reader** role.
   j. Click **Authorize**

Does Object Storage have FIPS 140-2 compliance for the encryption algorithms?

Yes, the IBM COS Federal offering is approved for FedRAMP Moderate Security controls, which require a validated FIPS configuration. IBM COS Federal is certified at FIPS 140-2 level 1. For more information on COS Federal offering, contact us via our Federal site.

Is client-key encryption supported?

Yes, client-key encryption is supported by using SSE-C, Key Protect, or HPCS.

Is encryption applied to a bucket by default?

Yes, by default, all objects stored in Object Storage are encrypted using randomly generated keys and an all-or-nothing-transform (AONT). You can get the encryption details using IBM Cloud UI/CLI. For details, see Cloud Storage Encryption.

FAQ - Bucket management

Frequently asked questions can produce helpful answers and insight into best practices for working with IBM Cloud® Object Storage.

How can I find out the total size of my bucket by using the API?

You can use the [Resource Configuration API](#) to get the bytes used for a given bucket.

How can I view my buckets?

You can view and navigate your buckets using the console, CLI or the API.

For example, the CLI command `ibmcloud cos buckets` will list all buckets associated with the targeted service instance.

Is there a 100-bucket limit to an account? What happens if I need more?

Yes, 100 is the current bucket limit. Generally, prefixes are a better way to group objects in a bucket, unless the data needs to be in a different region or storage class. For example, to group patient records, you would use one prefix per patient. If this is not a workable solution and you require additional buckets, contact IBM customer support.

When I create a bucket by using the API, how do I set the storage class?

The storage class (for example, `us-smart`) is assigned to the `LocationConstraint` configuration variable for that bucket. This is because of a key difference between the way AWS S3 and IBM Cloud Object Storage handle storage classes. Object Storage sets storage classes at the bucket level, while AWS S3 assigns a storage class to an individual object. For a list of valid provisioning codes for `LocationConstraint`, see [the Storage Classes guide](#).

Can the storage class of a bucket be changed? For example, if you have production data in 'standard', can we easily switch it to 'vault' for billing purposes if we are not using it frequently?

You can change the storage class by manually moving or copying the data from one bucket to another bucket with the wanted storage class.

Can the location of a bucket be changed?

To change a location, create a new bucket in the desired location and move existing data to the new bucket.

How many objects can fit in a single bucket?

There is no practical limit to the number of objects in a single bucket.

Can I nest buckets inside one another?

No, buckets cannot be nested. If a greater level of organization is required within a bucket, the use of prefixes is supported: `{endpoint}/{bucket-name}/{object-prefix}/{object-name}`. The object's key remains the combination `{object-prefix}/{object-name}`.

Can I restore a bucket from a specific back-up file?

It is possible to overwrite an existing bucket. Restore options depend on the capabilities provided by the back-up tool you use; check with your back-up provider. As described in [Your responsibilities when using IBM Cloud Object Storage](#), you are responsible for ensuring data back-ups if necessary. IBM Cloud® Object Storage does not provide a back-up service.

If I set an archive policy on an existing bucket, does the policy apply to existing files?

The policy applies to the new objects uploaded but does not affect existing objects on a bucket. For details, see [Add or manage an archive policy on a bucket](#).

Can I create a bucket, in the same or different region, with a deleted bucket name?

A bucket name can be reused as soon as 15 minutes after the contents of the bucket have been deleted and the bucket has been deleted. Then, the objects and bucket are irrevocably deleted and **can not** be restored.

If you do not first empty and then delete the bucket, and instead [delete or schedule the Object Storage service instance for deletion](#), the bucket names will be held in reserve for a [default period of seven (7) days until the account reclamation process](#) is completed. Until the reclamation process is complete, it is possible to restore the instance, along with the buckets and objects. After reclamation is complete, all buckets and objects will be irrevocably deleted and **can not** be restored, although the bucket names will be made available for new buckets to reuse.

How do I select an endpoint?

1. Go to the IBM Cloud Object Storage documentation for [endpoints](#) to research the desired levels of resiliency for your data and appropriate location.
2. Follow the steps to provision your instance in order to create a bucket, choosing a unique name. All buckets in all regions across the globe share a

single namespace.

3. Choose your desired level of resiliency, then a location where you would like your data to be physically stored. Resiliency refers to the scope and scale of the geographic area across which your data is distributed. Cross Region resiliency spreads your data across several metropolitan areas, while Regional resiliency spreads data across a single metropolitan area. A Single Data Center distributes data across devices within a single site only.

How do I find a bucket’s name?

To find a bucket’s name, go to the IBM Cloud console, select **Storage**, and then select the name of your Object Storage instance from within the **Storage** category. The Object Storage Console opens with a list of buckets, their names, locations, and other details. This name is the one you can use when prompted for a bucket name value by another service.

How do I find the details for a bucket?

To find the details for a bucket, go to the IBM Cloud console, select **Storage**, and then select the name of your Object Storage instance from within the **Storage** category. The Object Storage Console opens with a list of buckets. Find the bucket you want to see the details, and go to the end of the row and select the options list represented by the three-dot colon. Click the three-dot colon and select **Configuration** to see the details for the bucket.

How do I find a bucket’s location and endpoint?

You can view the bucket location in the IBM Cloud console with these steps:

1. From the IBM Cloud console, select **Storage** to view your resource list.
2. Next, select the service instance with your bucket from within the **Storage** category. This takes you to the Object Storage Console.
3. Choose the bucket for which you want to see location and endpoints.
4. Select **Configuration** from the navigation menu to view the page with Location and Endpoints data.

Or you can list bucket information with a GET request that includes the “extended” parameter as shown in Getting an extended listing.

Do Object Storage endpoints support IPv6 connections?

No.

How do I restrict access to a single bucket using IAM?

1. Go to the IBM Cloud Object Storage page for using service credentials to research the authentication topic.
2. Create a bucket, but do not add any public or other permissions to it.
3. To add the new user you first need to leave the current Object Storage interface and head for the IAM console. Go to the **Manage** menu and follow the link at **Access (IAM)** > **Users**. Click **Service Credentials**.
4. Click **New credential** and provide the necessary information. If you want to generate HMAC credentials, click the 'Include HMAC Credential' check box. Select the "Manager" service access role to allow the user to manage the bucket that you will select next.
5. Click **Add** to generate service credential.

How do I resolve a 404 error when using the command line?

You can view a bucket or object in the IBM Cloud console but the following error occurs when you use a command line interface to access that same bucket:

- Cloud CLI error: “The specified bucket was not found in your IBM Cloud account. This may be because you provided the wrong region. Provide the bucket's correct region and try again.”
- AWS CLI error: “An error occurred (NoSuchBucket) when calling the ListObjectsV2 operation: The specified bucket does not exist.”

The bucket’s location must correspond to the endpoint used by the CLI. This error occurs when the bucket or object cannot be found at the default endpoint for the CLI.

To avoid the error, make sure the bucket location matches the endpoint used by the CLI. For the parameters to set a region or endpoint, refer to the documentation for Cloud Object Storage CLI or AWS CLI.

How do I copy or move files to another bucket in a different location?

Refer to Move data between buckets for an example of how to use the `rclone` command line utility for copying data. If you use other 'sync' or 'clone' tools, be aware that you might need to implement a script to move files to a bucket in a different location because multiple endpoints are not allowed in a command.

Can I migrate a bucket from one COS instance to another?

Yes, You can achieve the same by creating a bucket in the target Object Storage instance and perform a sync. For details see cloud-object-storage-region-copy.

After deleting a Object Storage instance, is it possible to reuse the same bucket names that were part of the deleted COS instance?

When an empty bucket is deleted, the name of the bucket is held in reserve by the system for 10 minutes after the delete operation. After 10 minutes the name is released for re-use.

Can I enable Object Storage replication between two different regions for DR purposes?

Yes, it is possible to configure buckets for automated replication of objects to a destination bucket.

How can I setup notifications when objects are updated or written to a bucket?

You can use Code Engine to receive events about actions taken on your bucket.

Does Object Storage have rate limits when writing to or reading from buckets?

Yes, Object Storage has rate limiting. For details, see COS support.

How can I compare various attributes of an object in two different buckets?

Use `rclone`. It enables you to compare various attributes.

What is the default retention period for buckets?

There is no default retention period applied. You can set it while creating the bucket.

Can we add a retention policy to an existing bucket?

Yes, Retention policies can be added to an existing bucket; however, the retention period can only be extended. It cannot be decreased from the currently configured value.

Why is there a "legal hold" concept on top of the "retention period"?

A legal hold prevents an object from being overwritten or deleted. However, a legal hold does not have to be associated with a retention period and remains in effect until the legal hold is removed. For details, see Legal hold and retention period.

How to invoke IBM Cloud Object Storage bucket operations using cURL?

You have the most power by using the command line in most environments with IBM Cloud Object Storage and cURL. However using cURL assumes a certain amount of familiarity with the command line and Object Storage. For details, see Using cURL.

How can I list all permissions of a bucket?

The IAM feature creates a report at the instance level which may extend to their buckets. It does not specifically report at the bucket level. For details, see Account Access Report.

How do I get bucket information without using the web console?

Use the Object Storage Resource Configuration API to get bucket information. For details, see COS configuration and COS Integration.

How can I manage service credentials for Object Storage instances?

When a service credential is created, the underlying Service ID is granted a role on the entire instance of Object Storage. For details, see Managing Service credentials.

Why are parts of my credentials hidden or not viewable?

There may be an issue where the viewer does not have sufficient roles to view the credential information. For more information, see the account credentials documentation.

Is there a way to enable Key Protect on a Object Storage bucket after the bucket is created?

No, it is impossible to add Key Protect after creating a bucket. Key Protect can only be added while creating the bucket.

Can I host a website using a Object Storage bucket?

You can use Object Storage bucket to host a static website. For details, see Hosting Website using COS.

Are REST and cURL commands supported for Object Storage bucket creation using HMAC credentials?

Yes, you should setup an authorization header. For details, see Using HMAC Signature.

What kind of IAM authorization is required to edit a bucket's authorized IPs list?

You must have 'Manager' privilege on the bucket to manage the firewall and to set the authorizations.

Can I convert a single region Object Storage bucket to cross region without having to copy objects?

No, you must copy objects to the target bucket. For details, see COS Region Copy.

How can I set a notification when usage in a Object Storage instance is near a certain billing amount?

You can use a "soft" bucket quota feature by integrating with Metrics Monitoring and configuring for notifications. For details on establishing a hard quota that prevents usage beyond a set bucket size, see Using Bucket Quota.

How do I delete a non-empty bucket when I do not see any objects in it?

There may be versioned objects or incomplete multipart uploads that are still within the bucket but aren't being displayed. Both of these can be cleaned up by setting an expiry policy to delete stale data.

Also, you can delete multipart uploads directly using the Minio client command: `mc rm s3/ -I -r --force`

Why do I receive an error when I try to create a bucket?

Check IAM permissions because a user must have "Writer" permissions to create buckets.

Content-based restrictions may be preventing the user from acting on the service.

How do cross-origin resource sharing (CORS) and a bucket firewall differ in limiting access to data?

CORS allows interactions between resources from different origins that are normally prohibited. A bucket firewall allows access only to requests from a list of allowed IP addresses. For more information on CORS, see What is CORS?.

How do I allow Aspera High-Speed Transfer through a bucket with context-based restrictions or a firewall?

The full list (in JSON) of Aspera High-Speed Transfer IP addresses that are used with IBM Cloud Object Storage can be found using this API endpoint.

FAQ - Data management

Frequently asked questions can produce helpful answers and insight into best practices for working with IBM Cloud® Object Storage.

Can I migrate data from AWS S3 into IBM Cloud Object Storage?

Yes, you can use your existing tools to read and write data into IBM Cloud Object Storage. You need to configure HMAC credentials allow your tools to authenticate. Not all S3-compatible tools are currently unsupported. For details, see Using HMAC credentials.

How does IBM Cloud® Object Storage delete expired data?

Deletion of an object undergoes various stages to prevent data from being accessible (both before and after deletion). For details, see Data deletion.

What is the best way to structure your data by using Object Storage so you can 'look' at it and find what you are looking for?

You can use metadata that is associated with each object to find the objects you are looking for. The biggest advantage of Object Storage is the metadata that is associated with each object. Each object can have up to 4 MB of metadata in Object Storage. When offloaded to a database, metadata provides excellent search capabilities. Many (key, value) pairs can be stored in 4 MB. You can also use Prefix searching to find what you are looking for. For example, if you use buckets to separate each customer data, you can use prefixes within buckets for organization. For example: /bucket1/folder/object where 'folder/' is the prefix.

Can Object Storage partition the data automatically using HDFS, so I can read the partitions in parallel, for example, with Spark?

Object Storage supports a ranged GET on the object, so an application can do a distributed striped-read-type operation. Doing the striping is managed by the application.

Can I unzip a file after I upload it?

A feature to unzip or decompress files is not part of the service. For large data transfer, consider using Aspera high-speed transfer, multi-part uploads, or threads to manage multi-part uploads. See Store large objects.

How can I archive and restore objects in Object Storage?

Archived objects must be restored before you can access them. While restoring, specify the time limit the objects should remain available before being re-archived. For details, see archive-restore data.

Does an object in a bucket get overwritten if the same object name is used again in the same bucket?

Yes, the object is overwritten.

Are files scanned for viruses, while being uploaded to COS?

While there is no built in antivirus scanning in Object Storage, customers could enable a scanning workflow employing their own anti-virus technology that is deployed on Code Engine(/docs/codeengine?topic=codeengine-getting-started).

How can I use the Object Storage web console to download and upload large objects?

You can use IBM Cloud CLI or the API to download large objects. Alternatively, plugins such as Aspera /rclone can be used.

How do I access the reclaimed resources?

Create a new set of credentials to access the restored resources.

Is there a way to verify an object’s integrity during an upload to Object Storage?

Object Storage supports object integrity and ensures that the payload is not altered during transit.

Support

If you have problems or questions when you use IBM Cloud® Object Storage, you can get help starting right here.

Whether by searching for information or by asking questions through a forum, you can find what you need. If you don't, you can also open a support ticket.

Other support options

- If you have technical questions about Object Storage, post your question on [Stack Overflow](#) and tag your question with `ibm` and `object-storage`.

Next steps

For more information about asking questions, see [Contacting support](#).

See [Getting help](#) for more details about using the forums.

For more information about opening an IBM support ticket, see how to [create a request](#).

If you experience an issue or have questions when you deploy a Cloud Object Storage deployable architecture, you can use the following resources before you open a support case.

Review the FAQs. IBM Cloud icon Check the status of the IBM Cloud platform and resources by going to the Status page. GitHub icon Review the GitHub issues to see whether other users experienced the same problem. Review the troubleshooting documentation to troubleshoot and resolve common issues. If you still can't resolve the problem, you can open a support case. For more information, see Creating support cases. If you're looking to provide feedback, see Submitting feedback.

Providing support case details

To ensure that the support team can start investigating your case to provide a timely resolution, include details from the Schematics logs:

Find the Schematics log:

In the IBM Cloud console, go to Schematics > Workspaces > deployable architecture instance. From the workspace Activity page, select the Schematics apply action that failed. Click Jobs to see the detailed log output. Provide errors from the Schematics log:

In the log file, find the last action that Schematics started before the error occurred. For example, in the following log output, Schematics tried to run a copy script in the instances_module module by using the Terraform null_resource.

2021/05/24 05:03:41 Terraform apply | module.instances_module.module.compute_remote_copy_rpms.null_resource.remote_copy[0]: Still creating... [5m0s elapsed] 2021/05/24 05:03:41 Terraform apply | 2021/05/24 05:03:42 Terraform apply | 2021/05/24 05:03:42 Terraform apply | 2021/05/24 05:03:42 Terraform apply | Error: timeout - last error: ssh: rejected: connect failed (Connection timed out) Paste the errors into the case details. Provide the architecture name, source URL, and version from the log:

In the log file, find the architecture information. In the following example, you see the Related Workspace, sourcerelease, and sourceurl:

2024/11/19 09:14:44 Related Workspace: name=deploy-arch-ibm-cos-9758, sourcerelease=(not specified), sourceurl=, folder=terraform-ibm-cos-8.14.5/solutions/instance Copy the architecture information and paste it into the case details. Routing your support case

To route your support case correctly to speed up resolution, select the applicable product when you open the case.

Routing when you can't deploy successfully

If you can't deploy your deployable architecture, open a support case with the most likely cause of the issue:

If you identified the service that you think is causing the error from the Schematics log, use the name of that service as the product name in the case. If you can't identify the error from the Schematics log, use the name of the deployable architecture as it is listed in the IBM Cloud catalog. Routing when you deployed successfully

If you successfully deployed, yet have an issue with a service in the deployable architecture, open a support case and use the name of that service.

Billing and pricing

Billing

Information on pricing can be found at [IBM Cloud®](#).

Invoices

Find your account invoices at **Manage** > **Billing and Usage** in the navigation menu.

Under a Standard plan, service instance receives a single bill. If you need separate billing for different sets of buckets, then creating multiple instances is necessary.

Important: For each storage class, billing is based on aggregated usage across all buckets at the instance level. For example, for Smart Tier, the billing is based on usage across all Smart Tier buckets in a given instance - not on the individual buckets.

IBM Cloud Object Storage pricing

Storage costs for IBM Cloud® Object Storage are determined by the average monthly stored volume of data, the amount of public outbound bandwidth used, and the total number of operational requests processed by the system.

Tip: Infrastructure offerings are connected to a three-tiered network, segmenting public, private, and management traffic. Infrastructure services can transfer data between one another across the private network at no cost. Infrastructure offerings (such as bare metal servers, virtual servers, and cloud storage) connect to other applications and services in the IBM Cloud Platform catalog (such as Watson services) across the public network, so data transfer between those two types of offerings is metered and charged at standard public network bandwidth rates.

Request classes

'Class A' requests involve modification or listing. This category includes creating buckets, uploading or copying objects, creating or changing configurations, listing buckets, and listing the contents of buckets.

'Class B' requests are related to retrieving objects or their associated metadata or configurations from the system.

Deleting buckets or objects from the system does not incur a charge. For charges related to Multiple Deletes, see [Delete multiple objects](#).

Class	Requests	Examples
Class A	PUT, COPY, and POST requests, as well as GET requests used to list buckets and objects	Creating buckets, uploading or copying objects, listing buckets, listing contents of buckets, setting ACLs, and setting CORS configurations
Class B	GET (excluding listing), HEAD, and OPTIONS requests	Retrieving objects and metadata

Request classes

Note: Requests made using the Resource Configuration API are not charged for requests and do not accrue usage for billing purposes.

Aspera transfers

[Aspera high-speed transfer](#) incurs extra egress charges. For more information, see the [pricing page](#).

Storage classes

Not all data that is stored needs to be accessed frequently, and some archival data might be rarely accessed if at all. For less active workloads, buckets can be created in a different storage class and objects that are stored in these buckets incur charges on a different schedule than standard storage.

There are six classes:

- **Smart Tier** can be used for any workload, especially dynamic workloads where access patterns are unknown or difficult to predict. Smart Tier provides a simplified pricing structure and automatic cost optimization by classifying the data into "hot", "cool", and "cold" tiers based on monthly usage patterns. All data in the bucket is then billed at the lowest applicable rate. There are no threshold object sizes or storage periods, and there are no retrieval fees.

- **Standard** is used for active workloads, with no charge for data retrieved (other than the cost of the operational request itself).
- **Vault** is used for cool workloads where data is accessed less than once a month - an extra retrieval charge ($/GB) is applied each time data is read. The service includes a minimum threshold for object size and storage period consistent with the intended use of this service for cooler, less-active data.
- **Cold Vault** is used for cold workloads where data is accessed every 90 days or less - a larger extra retrieval charge ($/GB) is applied each time data is read. The service includes a longer minimum threshold for object size and storage period consistent with the intended use of this service for cold, inactive data.

Note: Flex has been replaced by Smart Tier for dynamic workloads. Flex users can continue to manage their data in existing Flex buckets, although no new Flex buckets may be created. Existing users can reference pricing information here.

For more information about pricing, see the pricing table at ibm.com .

Important: The **Active** storage class is only used with One Rate plans, and cannot be used in Standard or Lite plan instances.

For more information about creating buckets with different storage classes, see the API reference.

Smart Tier pricing details

Based on monthly averages, data in a Smart Tier bucket is classified into one of three tiers based on the following variables:

Variable	Description
`storage`	Total volume of data stored in GB
`retrievals`	Total volume of data retrieved in GB
`requests`	Sum of the number of Class A (write) requests plus 1/10 of the number of Class B (read) requests

Smart Tier bucket classification

- Data is classified **hot** if the total `requests > 1000 x (storage - retrievals)` .
- Data is classified **cold** if the total `requests < (storage - retrievals)` .
- Data is classified **cool** if neither of the above equations are true.

For example, let's imagine a bucket in the `us-south` region with an access pattern that changes from month to month. The bucket stores 1 TB of data, but some objects are very large and others are very small.

1. In the first month, there is a lot of activity but mostly with smaller objects. In total, there are 4 million requests and 100 GB is retrieved. This month the bucket is classified as **hot**.
2. In the second month activity slows down, but the focus is on larger objects. This month there are only 4 thousand requests but 200 GB is retrieved. Now the bucket is classified as **cool**.
3. In the third month activity slows to a near stop. There are only 400 requests and 10 GB is retrieved. This month the bucket is classified as **cold**.

Let's see how the costs might compare to the other storage classes.

Month	`storage`	`requests`	`retrieval`	Classification	Standard	Vault	Cold Vault	Smart Tier
1	1,000 GB	4,000,000	100 GB	Hot	$41	$53	$111	**$41**
2	1,000 GB	4,000	200 GB	Cool	$21	$14	$16	**$12**
3	1,000 GB	400	10 GB	Cold	$21	$12	$7	**$8**
Total	•	•	•	•	$83	$79	$134	**$61**

Cost comparison

Note that in situations where data is very cold, it is possible to get a lower rate with a Cold Vault bucket, although unexpected spikes in access could accrue significant costs. In this scenario, if the data doesn't require on-demand access, it might be better to archive the objects instead.

Free Tier monthly allowances

The following Free Tier allowances apply to each month for up to 12 months and apply to the total usage across all Smart Tier buckets in the Standard Plan:

- Up to 5 GB of Smart Tier storage capacity
- 2,000 Class A (PUT, COPY, POST, and LIST) requests
- 20,000 Class B (GET and all others) requests
- 10 GB of data retrieval
- 5GB of egress (public outbound bandwidth) each month

Get bucket metadata

In order to determine your current usage, you may wish to query a bucket to see `bytes_used` and `object_count`. Use of this command returns metadata containing that information for the specified bucket.

```
curl https://config.cloud-object-storage.cloud.ibm.com/v1/b/{my-bucket} \
                        -H 'authorization: bearer <IAM_token>'
```

The appropriate response to the request should contain `bytes_used` and `object_count`.

```
{
  "name": "{my-bucket}",
  "crn": "crn:v1:bluemix:public:cloud-object-storage:global:a/3bf0d9003abfb5d29761c3e97696b71c:d6f04d83-6c4f-4a62-a165-
696756d63903:bucket:my-new-bucket",
  "service_instance_id": "d6f04d83-6c4f-4a62-a165-696756d63903",
  "service_instance_crn": "crn:v1:bluemix:public:cloud-object-storage:global:a/3bf0d9003abfb5d29761c3e97696b71c:d6f04d83-6c4f-
4a62-a165-696756d63903::",
  "time_created": "2018-03-26T16:23:36.980Z",
  "time_updated": "2018-10-17T19:29:10.117Z",
  "object_count": 764265234,
  "bytes_used": 28198745752445144
}
```

Get resource information from an API

The resource controller is the next-generation IBM Cloud Platform provisioning layer that manages the lifecycle of Object Storage resources in a customer account. The API can provide actual billable metrics, such as types of requests and charges for storage, to get you started. More information can be found at the documentation

```
curl -X GET https://resource-controller.cloud.ibm.com/v2/resource_instances -H 'Authorization: Bearer <IAM_TOKEN>'
```

An appropriate response should list metadata for your resources as shown in the example.

```
{
  "rows_count": 1,
  "next_url": "/v2/resource_instances?
next_docid=g1AAAACkeJzLYWBgYMpgTmFQSklKzi9KdUhJMtTLTMrVTSouNjAw1EvOyS9NScwr0ctLLckBqc1jAZIMC4DU____92eBxdycyiQ6O2sOMCQxMLHnZKEaZ0
qEcQ8gxv2HG-fo9M_-Asg4-TVZWQCZcDI1&limit=2&account_id=d86af7367f70fba4f306d3c19c7344b2",
  "resources": [
    {
      "id": "crn:v1:bluemix:public:cloud-object-storage:global:a/4329073d16d2f3663f74bfa955259139:8d7af921-b136-4078-9666-
081bd8470d94::",
      "guid": "8d7af921-b136-4078-9666-081bd8470d94",
      "url": "/v2/resource_instances/8d7af921-b136-4078-9666-081bd8470d94",
      "created_at": "2018-04-19T00:18:53.302077457Z",
      "updated_at": "2018-04-19T00:18:53.302077457Z",
      "deleted_at": null,
      "name": "my-instance",
      "region_id": "global",
      "account_id": "4329073d16d2f3663f74bfa955259139",
      "resource_plan_id": "2fdf0c08-2d32-4f46-84b5-32e0c92fffd8",
      "resource_group_id": "0be5ad401ae913d8ff665d92680664ed",
      "resource_group_crn": "crn:v1:bluemix:public:resource-controller::a/4329073d16d2f3663f74bfa955259139::resource-
group:0be5ad401ae913d8ff665d92680664ed",
      "target_crn": "crn:v1:bluemix:public:resource-catalog::a/9e16d1fed8aa7e1bd73e7a9d23434a5a::deployment:2fdf0c08-2d32-4f46-
84b5-32e0c92fffd8%3Aglobal",
      "crn": "crn:v1:bluemix:public:cloud-object-storage:global:a/4329073d16d2f3663f74bfa955259139:8d7af921-b136-4078-9666-
081bd8470d94::",
      "state": "active",
      "type": "service_instance",
      "resource_id": "dff97f5c-bc5e-4455-b470-411c3edbe49c",
      "dashboard_url": "/objectstorage/crn%3Av1%3Abluemix%3Apublic%3Acloud-object-
storage%3Aglobal%3Aa%2F4329073d16d2f3663f74bfa955259139%3A8d7af921-b136-4078-9666-081bd8470d94%3A%3A",
      "last_operation": null,
      "resource_aliases_url": "/v2/resource_instances/8d7af921-b136-4078-9666-081bd8470d94/resource_aliases",
      "resource_bindings_url": "/v2/resource_instances/8d7af921-b136-4078-9666-081bd8470d94/resource_bindings",
      "resource_keys_url": "/v2/resource_instances/8d7af921-b136-4078-9666-081bd8470d94/resource_keys",
      "plan_history": [
        {
          "resource_plan_id": "2fdf0c08-2d32-4f46-84b5-32e0c92fffd8",
          "start_date": "2018-04-19T00:18:53.302077457Z"
        }
      ],
      "migrated": false,
      "controlled_by": ""
    }
  ]
}
```

Flex storage class pricing

Storage used	US South	US East	EU United Kingdom	EU Germany	AP Australia	AP Japan	São Paulo, Brazil	Toronto, Canada
0 - 499.99 TB	$0.009	$0.009	$0.0096	$0.0099	$0.0108	$0.0102	$0.0108	$0.0093
500+ TB	$0.009	$0.009	$0.0096	$0.0099	$0.0108	$0.0102	$0.0108	$0.0093

Storage Capacity (GB/month)

Storage used	US Cross Region	EU Cross Region	AP Cross Region
0 - 499.99 TB	$0.014	$0.0148	$0.0158
500+ TB	$0.014	$0.0148	$0.0158

Storage Capacity (GB/month)

Storage used	Amsterdam, Netherlands	Chennai, India	Melbourne, Australia	Milan, Italy	Montrèal, Canada	Paris, France	San Jose, US	Singapore
0 - 499.99 TB	$0.0093	$0.0108	$0.0108	$0.0099	$0.0093	$0.0099	$0.009	$0.0102
500+ TB	$0.0093	$0.0108	$0.0108	$0.0099	$0.0093	$0.0099	$0.009	$0.0102

Storage Capacity (GB/month)

Bandwidth used	US South	US East	EU United Kingdom	EU Germany	AP Australia	AP Japan	São Paulo, Brazil	Toronto, Canada
0 - 50 TB	$0.09	$0.09	$0.09	$0.09	$0.14	$0.14	$0.18	$0.09
Next 100 TB	$0.07	$0.07	$0.07	$0.07	$0.11	$0.11	$0.14	$0.07
Next 350 TB	$0.05	$0.05	$0.05	$0.05	$0.08	$0.08	$0.10	$0.05
Greater than 500 TB	Contact us	Contact us	Contact us	Contact us	Contact us	Contact us	Contact us	Contact us

Public outbound bandwidth (GB/month)

Bandwidth used	US Cross Region	EU Cross Region	AP Cross Region
0 - 50 TB	$0.09	$0.09	$0.14
Next 100 TB	$0.07	$0.07	$0.11
Next 350 TB	$0.05	$0.05	$0.08
Greater than 500 TB	Contact us	Contact us	Contact us

Public outbound bandwidth (GB/month)

Bandwidth used	Amsterdam, Netherlands	Chennai, India	Melbourne, Australia	Milan, Italy	Montrèal, Canada	Paris, France	San Jose, US	Singapore
0 - 50 TB	$0.09	$0.18	$0.14	$0.12	$0.09	$0.12	$0.09	$0.12
Next 100 TB	$0.07	$0.14	$0.11	$0.09	$0.07	$0.09	$0.07	$0.09
Next 350 TB	$0.05	$0.10	$0.08	$0.07	$0.05	$0.07	$0.05	$0.07
Greater than 500 TB	Contact us	Contact us	Contact us	Contact us	Contact us	Contact us	Contact us	Contact us

Public outbound bandwidth (GB/month)

Request type	US South	US East	EU United Kingdom	EU Germany	AP Australia	AP Japan	São Paulo, Brazil	Toronto, Canada
Class A: PUT, COPY, POST and LIST (per 1,000)	$0.01	$0.01	$0.01	$0.01	$0.01	$0.01	$0.01	$0.01
Class B: GET and all others (per 10,000)	$0.01	$0.01	$0.01	$0.01	$0.01	$0.01	$0.01	$0.01
Delete requests	No charge	No charge	No charge	No charge	No charge	No charge	No charge	No charge
Data retrieval (per GB)	$0.029	$0.029	$0.029	$0.029	$0.029	$0.029	$0.029	$0.029

Operational Requests

Request type	US Cross Region	EU Cross Region	AP Cross Region
Class A: PUT, COPY, POST and LIST (per 1,000)	$0.01	$0.01	$0.01
Class B: GET and all others (per 10,000)	$0.01	$0.01	$0.01
Delete requests	No charge	No charge	No charge
Data retrieval (per GB)	$0.029	$0.029	$0.029

Operational Requests

Request type	Amsterdam, Netherlands	Chennai, India	Melbourne, Australia	Milan, Italy	Montrèal, Canada	Paris, France	San Jose, US	Singapore
Class A: PUT, COPY, POST and LIST (per 1,000)	$0.01	$0.01	$0.01	$0.01	$0.01	$0.01	$0.01	$0.01

Class B: GET and all others (per 10,000)	$0.01	$0.01	$0.01	$0.01	$0.01	$0.01	$0.01	$0.01
Delete requests	No charge	No charge	No charge	No charge	No charge	No charge	No charge	No charge
Data retrieval (per GB)	$0.029	$0.029	$0.029	$0.029	$0.029	$0.029	$0.029	$0.029

Operational Requests

Flex cap	US South	US East	EU United Kingdom	EU Germany	AP Australia	AP Japan	São Paulo, Brazil	Toronto, Canada
Total GB stored and retrieved	$0.029	$0.029	$0.0296	$0.0299	$0.0308	$0.0302	$0.0308	$0.0293

Flex charge model for combined (storage capacity and data retrieval) is calculated using the lowest value of (A) storage capacity charge + data retrieval charge, or (B) capacity x Flex cap charge.

Request type	US Cross Region	EU Cross Region	AP Cross Region
Total GB stored and retrieved	$0.034	$0.0348	$0.0358

Flex charge model for combined (storage capacity and data retrieval) is calculated using the lowest value of (A) storage capacity charge + data retrieval charge, or (B) capacity x Flex cap charge.

Request type	Amsterdam, Netherlands	Chennai, India	Melbourne, Australia	Milan, Italy	Montrèal, Canada	Paris, France	San Jose, US	Singapore
Total GB stored and retrieved	$0.0293	$0.0308	$0.0308	$0.0102	$0.0299	$0.0299	$0.0290	$0.0302

Flex charge model for combined (storage capacity and data retrieval) is calculated using the lowest value of (A) storage capacity charge + data retrieval charge, or (B) capacity x Flex cap charge.

Aspera HST egress	US South	US East	EU United Kingdom	EU Germany	AP Australia	AP Japan	São Paulo, Brazil	Toronto, Canada
0 - 50 TB	$0.08	$0.08	$0.08	$0.08	$0.08	$0.08	$0.08	$0.08
Next 100 TB	$0.06	$0.06	$0.06	$0.06	$0.06	$0.06	$0.06	$0.06
Next 350 TB	$0.04	$0.04	$0.04	$0.04	$0.05	$0.04	$0.04	$0.04
Greater than 500 TB	Contact us	Contact us	Contact us	Contact us	Contact us	Contact us	Contact us	Contact us

Aspera High-Speed Transfer outbound bandwidth (GB/month)

Aspera HST egress	US Cross Region	EU Cross Region	AP Cross Region
0 - 50 TB	$0.08	$0.08	$0.08
Next 100 TB	$0.06	$0.06	$0.06
Next 350 TB	$0.04	$0.04	$0.04
Greater than 500 TB	Contact us	Contact us	Contact us

Aspera High-Speed Transfer outbound bandwidth (GB/month)

Aspera HST egress	Amsterdam, Netherlands	Chennai, India	Melbourne, Australia	Milan, Italy	Montrèal, Canada	Paris, France	San Jose, US	Singapore

0 - 50 TB	$0.08	$0.08	$0.08	$0.08	$0.08	$0.08	$0.08	$0.08
Next 100 TB	$0.06	$0.06	$0.06	$0.06	$0.06	$0.06	$0.06	$0.06
Next 350 TB	$0.04	$0.04	$0.04	$0.05	$0.04	$0.05	$0.05	$0.05
Greater than 500 TB	Contact us	Contact us	Contact us	Contact us	Contact us	Contact us	Contact us	Contact us

Aspera High-Speed Transfer outbound bandwidth (GB/month)