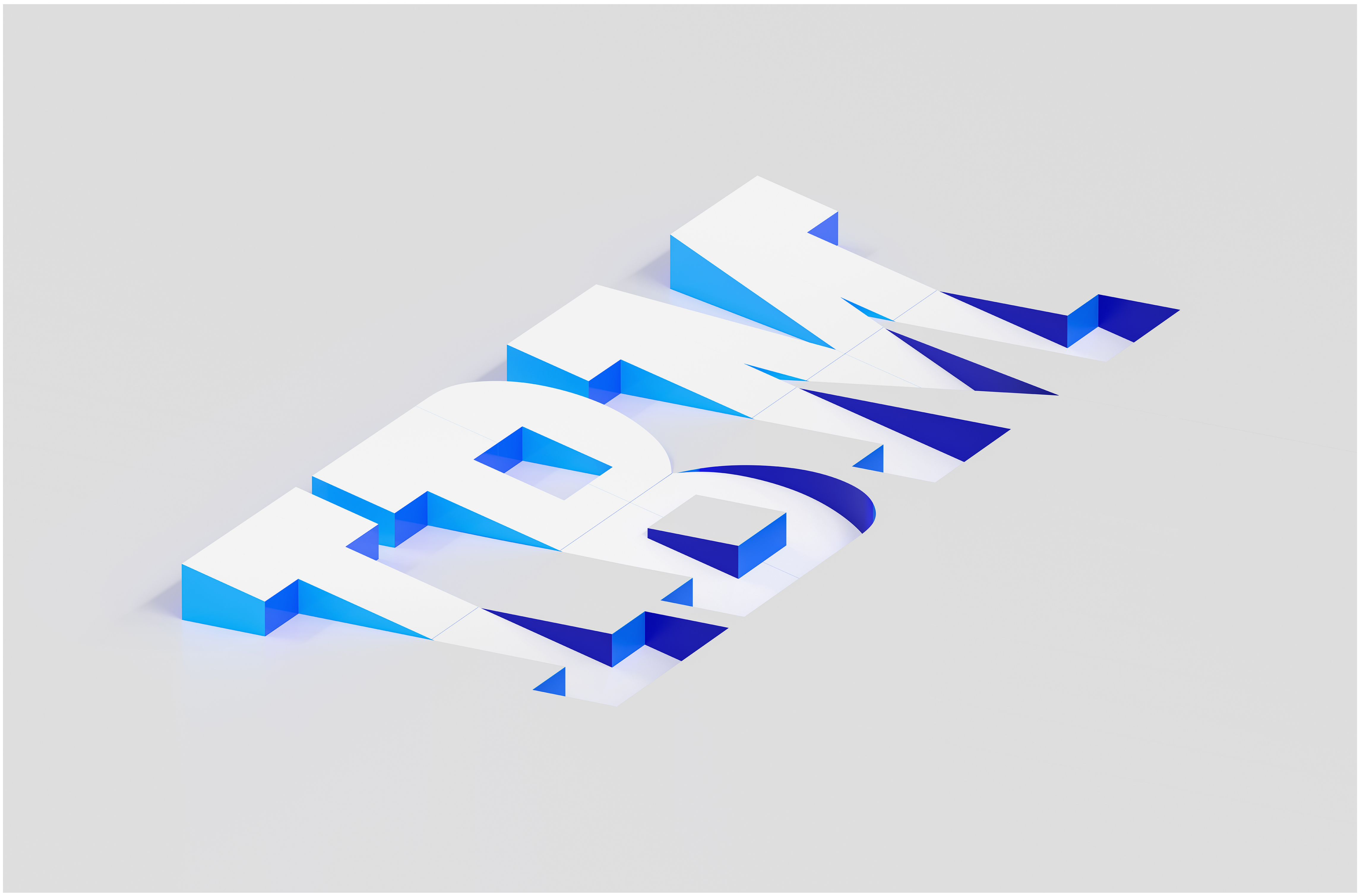
IBM Cloud

IBM Cloud Load Balancer

Product guide



Edition notices

Getting started with IBM Cloud Load Balancer

The IBM Cloud® Load Balancer service helps customers improve availability of their business-critical applications by distributing traffic among multiple application server instances, and by forwarding traffic to healthy instances only.

To use IBM Cloud® Load Balancer, you require two main items:

- An account with IBM - IBMid
- An IBM server, either Bare Metal or Virtual Server Instance (VSI)

If you need assistance in obtaining an IBMid account, contact your IBM Sales representative for extra guidance.

Ordering a load balancer

To get started with IBM Cloud® Load Balancer, see Using an elastic load balancer for server load-balancing overview.

About IBM Cloud Load Balancer

The IBM Cloud® Load Balancer service offers the following features:

- Public (internet-facing) load balancer
  - Publicly accessible service through its fully qualified domain name (FQDN)
  - Back-end server instances on private subnets
- Internal load balancer
  - Privately accessible service through its fully qualified domain name (FQDN)
  - Client requests are routed over the private network
  - Back-end server instances on private subnets
- Basic load balancing
  - Traffic distribution based on layer 4 application port information
  - Support for HTTP, HTTPS, and TCP-based applications
  - A variety of load-balancing methods, such as round robin, weighted round robin, and least connections
  - Load balancing among virtual server and bare metal compute instances that reside locally within a data center
- Server health checks
  - Periodic monitoring of server health to ensure that traffic is forwarded to healthy servers only
  - Layer 4 health checks for TCP ports and Layer-7 health checks for HTTP ports
- SSL offload: Termination of incoming SSL (HTTPS) traffic that uses plain-text HTTP communication with back-end servers
- Advanced traffic management
  - Client stickiness (session persistence)
  - Maximum connections per virtual port
- Easy management that uses an intuitive graphical interface and API
- Built-in reliability
- Usage-based pricing
- Monitoring: Monitors the throughput, active connections and connection rate metrics for HTTP, HTTPS, and TCP protocols over user-specified time intervals.
- Layer 7 support
  - HTTP/HTTPS traffic is routed to different back-end services based on the HTTP header, and is done by using policies and rules. Rules are used to classify the traffic and are based on the HTTP header fields. When the traffic matches all the rules, an action that is specified by the policy is taken.
- Multi-Zone Region (MZR) support: Load balancer nodes are instantiated in different data centers of an MZR. For more information, see Multi-Zone Region overview.

Pricing metrics

IBM Cloud Load Balancer determines its pricing based on the following metrics.

Instance hour per month: Measures the number of hours CLB is used per calendar month.

Data Processed: Measures how much data, in gigabytes (GB), that is processed per calendar month.

Bandwidth Usage: Measures the amount of bandwidth, in gigabytes (GB), used per calendar month.

Note: This pricing metric has multiple pricing tiers that differ based on the amount of data used.

Tip: You can estimate the cost of a service by using the cost estimator on the provisioning pages for IBM Cloud Load Balancer. SelectIBM Cloud® Load Balancer from the Load Balancer page of the IBM Cloud catalog, then click Create.

Load balancer basics

The IBM Cloud® Load Balancer service distributes traffic among multiple server instances (bare metal and virtual server) that reside locally, within the same data center.

Public-to-private load balancer

A publicly accessible, fully qualified domain name is assigned to your load balancer service instance. Use this domain name to access your applications hosted behind the load balancer service. This domain name can be registered with one or more public IP addresses. The public IP addresses and number of public IP addresses can change over time based on maintenance and scaling activities, which are transparent to users. Back-end compute instances hosting your application must be on an IBM Cloud Private network.

Note: As a good practice, it is recommended that you provision your back-end servers as `private-only`, unless they require direct public connectivity. This practice helps achieve better security, and it preserves your public IP address. The applications that are hosted on these back-end servers are still accessible over public networks by using the load balancer.

There are two ways to allocate public IP addresses to the load balancer:

- Allocate from IBM system pool: The default method. This method allocates IP addresses from IBM system pool.
- Allocate from a public subnet in this account: This method allocates IP addresses from a public subnet from your account. By default, the subnet selection is automatic. If you are using API, you can select a particular subnet by passing the public subnet ID when placing the order.

Private-to-private load balancer

The internal load balancer is only accessible within the IBM Cloud Private network.

Just like a public-to-private load balancer, your internal load balancer service instance is also assigned a fully qualified domain name. However, this domain name is registered with one or more private IP addresses.

Similar to a public load balancer, the private IP addresses and their numbers can change over time based on maintenance and scaling activities, which are transparent to users.

Note: The back-end compute instances hosting your application must also be on the IBM Cloud private network.

Public-to-public load balancer

The public-to-public load balancer is publicly accessible.

Just like other load balancer types, your public-to-public load balancer service instance is assigned a fully qualified domain name. This domain name can be registered with one or more public IP addresses.

After the load balancer is provisioned, only the public IP addresses of the members in the server tables context are visible. This is applicable for both layer 4 and layer 7 load balancer members.

The public IP addresses and their numbers can change over time based on maintenance and scaling activities, which are transparent to users.

Note: The back-end compute instances hosting your application must be on the IBM Cloud public network.

Front-end and back-end application ports/protocols

You can define up to 10 front-end application ports (protocols) and map them to respective ports (protocols) on the back-end application servers. The fully qualified domain name that is assigned to your load balancer service instance and the front-end application ports are exposed to the external world. The incoming user requests are received on these ports.

The back-end ports are only known internally. These back-end ports might or might not be the same as the front-end ports. As an example, the load balancer can be configured to receive incoming web/HTTP traffic on front-end port 80, while the back-end servers are listening on custom port 81.

The supported front-end ports/protocols are HTTP, HTTPS, and TCP. The supported back-end ports/protocols are also HTTP, HTTPS, and TCP. Incoming HTTPS traffic must be terminated at the load balancer to allow for plain text HTTP communication with the back-end server. If the back-end protocol is HTTPS, the traffic is encrypted between load balancer and back-end servers.

Considerations

- During the initial configuration, you can define up to two front-end ports only. After a load balancer is created, you can edit the port configuration to define extra ports, up to the maximum of 10 ports.
- All 10 front-end ports must map to the same set of back-end server instances.
- The maximum number of server instances that can be placed behind a load balancer is 50.
- The port range of 56500 to 56520 is reserved for management purposes and cannot be used as front-end virtual ports.
- TCP port 56501 is used for management. If you choose to allocate load balancer public IP addresses from your public VLAN, ensure that traffic to this management port as well as your application's ports are not blocked by any firewalls you might have deployed on your public VLANs. This is not required if you choose the IBM system pool option to allocate load balancer public IP addresses.

Load-balancing methods

The following three load-balancing methods are available for distributing traffic among back-end application servers:

- Round Robin - Round Robin is the default load-balancing method. With this method, the load balancer forwards incoming client connections in round-robin fashion to the back-end servers. As a result, all back-end servers receive roughly an equal number of client connections.

- Weighted Round Robin - With this method, the load balancer forwards incoming client connections to the back-end servers in proportion to the weight assigned to these servers. Each server is assigned a default weight of 50, which you can customize to any value in the range 0 - 100.

  As an example, if there are three application servers A, B and C, and their weights are customized to 60, 60 and 30 respectively, then servers A and B receive an equal number of connections, while server C receives half that number of connections.

  Note: Resetting a server weight to 0 means that no new connections are forwarded to that server, but any existing traffic continues to flow as long as it is active. Using a weight of 0 can help bring down a server gracefully and remove it from service rotation.

  Note: The server weight values are applicable only with the weighted round robin method. Server weight values are ignored with round-robin and least connection load-balancing methods.

- Least Connections - With this method, the server instance serving the least number of connections at a given time receives the next client connection.

Horizontal scaling

The load balancer adjusts its capacity automatically according to the load. When this occurs, you might see a change in the number of IP addresses associated with the load balancer's DNS name.

Multizone region requirements

A Multizone Region (MZR) ensures high availability by deploying your load balancer appliances in multiple data centers. When you create a load balancer, you select the subnet for deployment. If the data center is part of an MZR, one appliance is deployed in your selected data center, and the other is deployed in a different data center within the same region.

For example, Dallas (`us-south`) is an MZR that contains the data centers `dal10`, `dal12`, and `dal13`. Subnet A is in `dal10`, subnets B and C are in `dal12`, and subnets D and E are in `dal13`. If you create a load balancer in the `dal13` data center, the first appliance is deployed in `dal13` and the second appliance is deployed in the subnet with the most available IPs between `dal10` or `dal12` data centers.

Currently, the Cloud Load Balancer service is available in the data centers listed in Classic infrastructure.

MZRs have the following requirements:

- The selected data center must be part of an MZR.
- VLAN spanning or VRF must be enabled in your account.
- Private subnets must exist in your account in the data centers of the MZR. Creation of compute devices in data centers results in the instantiation of private subnets.

Note: If the selected data center is not part of an MZR, or if VLAN spanning or VRF is not enabled in your account, the load balancer will be created with all nodes instantiated in the specified data center, following the original deployment behavior.

For more information about multizone regions, see IBM Cloud region and data center locations for resource deployment. For information about the mapping between regions and zones, see Zone mapping per account.

Exploring IBM Cloud load balancers

IBM Cloud offers several load balancing solutions to choose from. The following table compares the load balancing solutions to help you choose the one that's right for you. To learn more about the individual offering, click its name in the table.

	IBM Cloud® Load Balancer	Citrix NetScaler VPX (Standard)	Citrix NetScaler VPX (Platinum)
Public VIP	✓	✓	✓
Private VIP	✓	✓	✓
Layer 4 load balancer	✓	✓	✓
Layer 7 load balancer	✓	✓	✓
Health Checks	✓	✓	✓
Horizontal Scaling	✓		
SSL offload	✓	✓	✓
Management	IBM console	Self-manage (Vendor GUI)	Self-manage (Vendor GUI)
High Availability	Built-in	Optional	Optional
Advance LB (TCP Optimization, Compress, Caching, WAF)		Limited	✓
Global LB			✓

A comparison of IBM's load balancer offerings

Demo: Network as a service

IBM Cloud® network services, including load balancers and security groups, make it easy to build cloud-native applications with greater security, performance, and resiliency. In this demo, learn how you can use these services to create a simple three-tiered web app in a matter of clicks.

See how to set up a web app with a load balancer and security groups

Other resources for IBM Cloud Load Balancer

IBM and some of its partners and customers created the following resources:

- Education - [Load Balancing: How load balancing optimizes website and application performance]

Using an elastic load balancer for server load balancing

Creating a load balancer service overview

The IBM Cloud® Load Balancer service helps improve scalability and availability of your business-critical applications. It monitors the health of your application servers, uses smart load-balancing methods to distribute traffic among multiple servers, and dynamically adjusts its system capacity to handle varying traffic load.

In this step-by-step guide, learn how to create and configure a new load balancer service.

Task	Description
Selecting the service and configuring basic options	Configure basic options, such as the name, description, and type of your load balancer.
Configuring load-balancing options and placing your order	Configure protocols, health checks, and back-end servers.
Monitoring and managing your service	Edit your configuration and monitor your load balancer performance.

Configure a load balancer

Important: Mandatory management port 56501 must be allowlisted in order for load balancer provisioning (as well as customer and service triggered operations) to succeed. For more information, refer to this FAQ.

Selecting the service and configuring basic options

To begin creating your new IBM Cloud® Load Balancer, selectIBM Cloud® Load Balancer from the Load Balancer page of the IBM Cloud catalog, then click Create.

When the configuration page appears, follow these steps:

1. Enter the name for the load balancer instance. The fully qualified domain name (FQDN) of the load balancer is based on this name.
2. Enter a description for the load balancer (optional).
3. Select the data center where the load balancer instance is to be created.

   Tip: For multi-zone availability, all requirements for MZR must be satisfied.

4. Select the type of load balancer that you want to create from the options **Public-to-Private,Private-to-Private, or Public-to-Public. Refer to IBM Cloud® Load Balancer basics for details on each type.
5. Select the subnet where you want to deploy your new load balancer.

   Note: This option applies to only Public-to-Private and Private-to-Private load balancer types. Your load balancer service instance has one of its network interfaces on this subnet. Ensure that your application servers are either on this subnet or reachable from this subnet. If necessary, enable VLAN spanning.

6. Select the allocation of public IPs for the load balancer.

   Note: This option applies to only the Public-to-Private load balancer type. If you select Allocate from a public subnet in this account, you must have at least two public IPs available in a public subnet in the same data center. Also, ensure that traffic to TCP management port 56501 and your application's own ports are not blocked by firewalls that are deployed on your public VLANs.

What's next

Configuring load-balancing options and placing your order.

Configuring load-balancing options and placing your order

Configure your load balancer and finalize your order.

Step 1 - Adding protocols

Add the protocols for your load balancer:

1. On the load balancer configuration page, identify the protocols and ports your application is listening on, and then input the details into your new application profile. You can use the same configuration for both front-end and back-end, or expose a different front-end port (for security purposes, for instance).
2. The default is Round Robin. You can change it to either Weighted Round Robin or Least Connections from the list, depending on your application needs.
3. Optionally, you can enable Session stickiness, which sends all requests from a user (for example, one with the same source IP) to the same back-end server for a system defined "sticky" time. For more information on session stickiness, see Session persistence
4. You can also set the Maximum connection limit against your application. For more information, see Max connections
5. Click Add Protocol to specify extra ports and protocols your application might be listening on. Be sure that all front-end ports are unique. You can choose HTTP, HTTPS, or TCP as your front-end protocol.

   Note: A maximum of two ports can be defined at the time of initial configuration. Extra ports can be added after creating the service instance. Refer to Limitations on number of ports for more information on the maximum number of ports allowed.

6. If you chose HTTPS for your front-end protocol, you must upload your SSL Certificate. Select one of your available certificates from the drop-down list.

   The IBM Cloud® Load Balancer terminates incoming HTTPS connections and can communicate in plain text HTTP with the back-end application servers when HTTP is selected as the backend protocol. This offloads processor intensive SSL tasks from your servers to the load balancer. You can also choose to have HTTPS as the backend protocol when the backend application servers are configured to receive HTTPS traffic. In this case, traffic is encrypted between the load balancer and the backend servers.

   Note: If you do not have an existing certificate, go to the IBM Cloud Certificate service and either purchase a new certificate or upload an existing one. After adding the certificate, return to the load balancer configuration page and click the refresh icon next to the SSL Certificate drop-down list to view and add your newly added certificate.

   Important: Never delete any certificates associated with HTTPS listeners as this can cause issues.

Step 2 - Configuring health checks

The health check definition is mandatory for each of your application ports (the back-end ports that are identified in the protocols section).

The system pre-populates a default health check configuration for these back-end ports, and you can customize these settings to suit your application needs:

- Interval: Interval in seconds between two consecutive health check attempts
- Timeout: Maximum amount of time the system waits for a response against a health check request
- Max Trials: Maximum number of extra health check attempts the system makes before declaring a port unhealthy
- Path: The HTTP URL path for the health check

Step 3 - Adding server instances

Select your server instance from the dropdown in the table, and use Attach Server to add more servers. You can select from the IBM© Cloud Virtual Server Instances (VSIs) and Bare Metal Servers in your account.

These server instances must be local to the data center where you deploy the load balancer service. Also, server instances from the neighboring data centers within the same city can also be added (for instance, if the first three letters of the data center name are the same).

If the load balancer type is Public to Public, the server instances are added with their Public IP, as shown in the following image.

Note: Server weights are relevant only when using the Weighted Round Robin load-balancing method. The default weight is 50 and the range is 0-100. The weights are disabled with other load-balancing methods.

Tip: Refer to Limitations on number of application servers for more information on the maximum limit for the number of application servers.

Step 4 - Placing your order

Finally, to place your order:

1. Review the Order Information in the right column of the page.
2. Click the checkbox after reading the Master Service Agreement.
3. Click Create.

   The system creates your load balancer and takes you to the Load Balancers summary page. Refresh your browser window to see the entry for your new load balancer and its state change from offline to online, which usually takes a few minutes. Offline load balancers are disabled, while online load balancers can be clicked to manage and monitor them, as shown in the following image.

What's next

Managing and monitoring your service to edit configuration and monitor service performance.

Monitoring and managing your service

You can edit your configuration or monitor your service performance by clicking the load balancer name in the load balancer summary page.

The fully qualified domain name (FQDN) address of your load balancer instance can be seen by clicking the Details button. Your users are able to connect to your application by using this FQDN address.

Note: The public and private IP addresses of the load balancer service are not exposed to the outside world; only the FQDN address is exposed.

The Details list also displays the type, location, and logging enablement of the load balancer.

The Overview tab on the left of the page provides high-level information about your service. It displays the current health of your application servers and their ports, and also provides a quick summary of system performance - throughput, connection rate, concurrent connections, and so on.

If you already set up your IBM Cloud Monitoring instance, you can select the Launch monitoring button in the Monitoring section to view real-time charts of your system performance. You can view these graphs per individual application port and for various time durations. To work with the IBM Cloud Monitoring dashboard, follow these instructions.

If you have not already set up your IBM Cloud Monitoring instance, you can select the Configure monitoring button in the Monitoring section. Follow these instructions to enable metric monitoring.

You can edit your existing configuration by using the Protocols, Server Instances, and Health Checks tabs. For example, the Protocols tab can be used to define extra application ports or to customize the SSL cipher lists when there is an existing HTTPS protocol.

After every configuration change, the load balancer goes into an UPDATE PENDING state. In this state, datapath traffic is not affected, but no further updates can be made to that load balancer. Click the refresh button next to the load balancer's state to check the latest status.

You can also use the Layer 7 tab to configure Layer 7 load balancing.

Setting up Terraform for IBM Cloud Load Balancer

Terraform on IBM Cloud® enables predictable and consistent provisioning of IBM Cloud services so that you can rapidly build complex, multitier cloud environments following Infrastructure as Code (IaC) principles. Similar to using the IBM Cloud CLI or API and SDKs, you can automate the provisioning, update, and deletion of your load balancer instances by using HashiCorp Configuration Language (HCL).

Tip: Looking for a managed Terraform on IBM Cloud® solution? Try out IBM Cloud® Schematics. With Schematics, you can use the Terraform scripting language that you are familiar with, but you don't must worry about setting up and maintaining the Terraform command line and the IBM Cloud® Provider plug-in. Schematics also provides pre-defined Terraform templates that you can easily install from the IBM Cloud® catalog.

Installing Terraform and configuring resources for IBM Cloud Load Balancer

1. Follow the Terraform on IBM Cloud® getting started tutorial to install the Terraform CLI and configure the IBM Cloud® Provider plug-in for Terraform. The plug-in abstracts the IBM Cloud® APIs that are used to provision, update, or delete IBM Cloud Load Balancer service instances and resources.

2. Create a Terraform configuration file that is named `main.tf`. In this file, you add the configuration to create a load balancer and to assign a user an access policy in Identity and Access Management (IAM) for that instance by using HashiCorp Configuration Language (HCL). For more information, see the Terraform documentation.

   The load balancer instance in the following example is named `test_lb_local_service` and is enabled on port `80`. The ID of the local load balancer service group is `ibm_lb_service_group.test_service_group.service_group_id` and the weight for the load balancer service group is `1`.

   Note: For more information, see the ibm_lb_service usage example.

   ```
   resource "ibm_lb_service" "test_lb_local_service" {
    port = 80
    enabled = true
    service_group_id = ibm_lb_service_group.test_service_group.service_group_id
    weight = 1
    health_check_type = "DNS"
    ip_address_id = ibm_compute_vm_instance.test_server.ip_address_id
    }
   ```

3. Initialize the Terraform CLI.

   ```
   terraform init
   ```

4. Create a Terraform execution plan. The Terraform execution plan summarizes all the actions that need to be run to create the load balancer in your account.

   ```
   terraform plan
   ```

5. Create the load balancer instance and IAM access policy in IBM Cloud.

   ```
   terraform apply
   ```

6. From the IBM Cloud resource list, select the load balancer instance that you created and note the instance ID.

Performing load balancer advanced tasks

Choosing a preferred cipher suite for your HTTPS application

Cipher algorithms help the IBM Cloud® Load Balancer form secure connections with its HTTP clients. IBM offers a suite of approved ciphers for you to choose from so that you can secure the communication between your load balancer and your clients.

To choose a preferred cipher suite for an existing load balancer:

1. Access the Cipher table from the Protocols tab.

   Note: There must be at least one existing HTTPS protocol.

2. Click Edit to enable editing, and select (or clear) the required ciphers.
3. When you're done, click Save to save your changes.

   Note: If you defined an HTTPS protocol when you create your load balancer, it has all ciphers in the suite enabled. The selection of a particular set of ciphers can be done only after the load balancer is provisioned.

   Note: For a list of supported ciphers, refer to SSL offload.

Performing health checks

The load balancer conducts periodic health checks to monitor the health of the back-end ports and forwards client traffic to them. If a given back-end server port is found unhealthy, then new connections are not forwarded to it. The load balancer continues to monitor the health of these unhealthy ports, and resumes their use if they are found healthy again by successfully passing two consecutive health check attempts.

Health check definitions are mandatory for each of the back-end application ports. The port and protocol under health check configuration must match with the defined back-end port and protocol. Otherwise, the configuration is rejected.

The health checks against HTTP，HTTPS and TCP ports are conducted as follows:

- HTTP - An `HTTP GET` request against a pre-specified URL is sent to the back-end server port. The server port is marked healthy upon receiving a `200 OK` response. The default `GET` URL is “/” via the GUI, and it can be customized.
- HTTPS - Only applicable when back-end encryption is enabled and the back-end protocol is set to HTTPS. The mechanism is the same as HTTP, except that all health check messages are SSL encrypted. For more information, see Setting back-end encryption.
- TCP - The load balancer attempts to open a TCP connection with the back-end server on a specified TCP port. The server port is marked healthy if the connection attempt is successful. Then, the connection is closed.

  Note: The default health check interval is 5 seconds. The default timeout against a health check request is 2 seconds, and the default number of retry attempts is 2.

SSL offload with IBM Cloud Load Balancer

For all incoming HTTPS connections, the load balancer service ends the SSL connection and establishes a plain text HTTP communication with the back-end server. CPU-intensive SSL handshakes and encryption or decryption tasks are shifted away from the back-end servers, allowing them to use all their CPU cycles for processing application traffic.

An SSL certificate is required for the load balancer to perform SSL offload tasks. You can use a pre-existing SSL certificate or purchase a new one, and manage it through the SSL Certificates page.

SSL cipher suites

The load balancer service supports TLS version 1.2 with SSL offload.

The following SSL ciphers are supported by your load balancer:

- ECDHE-RSA-AES256-GCM-SHA384
- ECDHE-RSA-AES256-SHA384

- AES256-GCM-SHA384
- AES256-SHA256
- ECDHE-RSA-AES128-GCM-SHA256
- ECDHE-RSA-AES128-SHA256
- AES128-GCM-SHA256
- AES128-SHA256

If your load balancer has one or more HTTPS front-end application ports (protocols) configured, by default, all these predefined SSL ciphers are enabled for your load balancer.

Note: You can choose to enable different SSL ciphers for your load balancer if needed. For more information, seeChoosing a preferred cipher suite for your HTTPS application.

Advanced traffic management with IBM Cloud Load Balancer

Learn about the various advanced traffic-management features available with the IBM Cloud® Load Balancer service.

Max connections

Use the `max connections` configuration to limit the maximum number of concurrent connections against a given front-end virtual port. The maximum concurrent connections against a given front-end virtual port or system-wide across all front-end virtual ports is `15000`. By default, it is set to the maximum value of `15000`.

Session persistence

The load balancer supports session persistence based on the source IP of the connection. As an example, if you have `source IP` type session persistence that is enabled for port `80` (HTTP), then subsequent HTTP connection attempts from the same source IP client are persistent on the same back-end server. This feature is available for all three supported protocols (HTTP, HTTPS, and TCP).

The load balancer also supports session persistence based on the HTTP Cookie. For example, if you have `HTTP Cookie` type session persistence that is enabled for port `80` (HTTP), when the load balancer receives its first response from the back-end server, it adds a cookie with the name `IBMCLB` and a value of `back-end server UUID` in the response header. All subsequent HTTP requests with this cookie that arrives at the load balancer are persistent on the same back-end server. This feature is available for both HTTP and HTTPS.

HTTP keep alive

The load balancer supports `HTTP keep alive` when it is enabled on both the client and back-end servers. The load balancer attempts to reuse the server-side HTTP connections to increase connection efficiency and reduce latency.

Connection timeouts

The following timeout values are used by the load balancer:

Name	Description	Default timeout	User configurable
Server-side connection attempt	The maximum time window that the load balancer can use to establish a TCP connection with the back-end server. If the connection attempt is unsuccessful, the load balancer tries the next available server, according to the load-balancing method that you configured.	5 seconds	No
Client-side idle connection	The maximum idle time after which the load balancer brings down the client-side connection, if the client failed to close its connection properly.	50 seconds	Yes
Server-side idle connection	The maximum idle time (with back-end protocol configuration of TCP) after which the load balancer closes the server-side connection. When HTTP is used as the back-end protocol, if the load balancer doesn't receive an HTTP response within the idle timeout period, it returns an error message to the client.	50 seconds	Yes

Load Balancer Timeout Values

Server-side and client-side idle connection timeout values can be configured by using the API, cURL or from the CLI.

Note: You can configure the server timeout (`ParameterName: serverTimeout`) and client timeout (`ParameterName: clientTimeout`) value in seconds up to 2 hours (Range: `1 - 7200` seconds) by using `UpdateLoadBalancerProtocols` method of `SoftLayer_Network_LBaaS_Listener` service. If you do not provide the server or client timeout values, the load balancer uses the default value (mentioned in the table) for the corresponding timeout.

Setting the timeout value with the API

You can set the client and server timeout values with the API and cURL.

To set the timeout value with the API, follow these steps:

```
import SoftLayer
from pprint import pprint

#Your load balancer UUID
uuid = 'set me'

#New protocols to add
protocolConfigurations = [
  {
  "listenerUuid": "69fad83a-e850-4b72-a4d3-af94d5bf5437",
  "serverTimeout": 60,
  "clientTimeout": 60
  },
  {
   "listenerUuid": "e4b8cfd0-1e27-4d3e-a8ed-595b198cd683",
   "frontendPort": 1450,
   "maxConn": 1002,
   "serverTimeout": 80,
   "clientTimeout": 80
  }
]

#Create the API client
client = SoftLayer.Client()
listener_service = client['Network_LBaaS_Listener']

_mask = "mask[listeners]"

try:
  response = listener_service.updateLoadBalancerProtocols(uuid, protocolConfigurations, mask=mask)
  pprint(response)
except SoftLayer.SoftLayerAPIError as e:
  print("Unable to add protocols: %s, %s" % (e.faultCode, e.faultString))
```

For more information, see updating a protocol in the IBM Cloud Load Balancer API reference page.

To set the timeout values by using cURL, follow these steps:

1. Get load balancer information:

   ```
     curl -g -u $SL_USER:$SL_APIKEY -X POST -d '{"parameters":["CLB_UUID"]}'
   'https://api.softlayer.com/rest/v3.1/SoftLayer_Network_LBaaS_LoadBalancer/getLoadBalancer' | jq
   ```

2. Get load balancer listeners:

   ```
     curl -g -u $SL_USER:$SL_APIKEY -X GET 'https://api.softlayer.com/rest/v3.1/SoftLayer_Network_LBaaS_LoadBalancer/CLB_ID/getListeners' | jq
   ```

3. Use the listener ID you obtained in the previous step to get listener details:

   ```
     curl -g -u $SL_USER:$SL_APIKEY -X GET 'https://api.softlayer.com/rest/v3.1/SoftLayer_Network_LBaaS_Listener/CLB_LISTENER_ID/getObject' | jq
   ```

4. Set client or server timeout values:

   ```
     curl -g -u $SL_USER:$SL_APIKEY -X POST -d '{"parameters": ["LB_UUID", [{"listenerUuid": "LISTENER_UUID", "clientTimeout": 1000}]]}'
   'https://api.softlayer.com/rest/v3.1/SoftLayer_Network_LBaaS_Listener/updateLoadBalancerProtocols.json' | jq
   ```

Note: Obtain the load balancer UUID and listener UUID from the previous steps.

Setting the timeout value from the CLI

To set the timeout values from the CLI, follow these steps:

1. Get the load balancer list in your account:

   ```
   $   ibmcloud sl loadbal list
   ```

2. Get the ID of the required load balancer for which you want set the timeout value:

   ```
   $   ibmcloud sl loadbal detail <LB-id>
   ```

3. Edit the parameters obtained from previous command:

   ```
   $   ibmcloud sl call-api SoftLayer_Network_LBaaS_Listener updateLoadBalancerProtocols --parameters '["LB_UUID",
   [{"tlsCertificateId":null,"listenerUuid":"LISTENER_UUID","clientTimeout": <Value in seconds>}]]'
   ```

4. Verify your changes:

   ```
   $   ibmcloud sl loadbal detail <LB-id>
   ```

Tip: Setting long idle connection timeout values can cause your data path traffic to experience latencies or be blocked because the idle connections are counted toward the maximum concurrent connections of 15,000. Consider the idle connection timeout value along with the maximum number of concurrent connections to ensure that your data path traffic is not disrupted.

Preserving end-client IP address

IBM Cloud Load Balancer works as a reverse proxy, which handles incoming traffic from the client. It establishes a separate connection to the back-end server instance by using its own IP address. For HTTP connections with the back-end servers (against front-end HTTP or HTTPS connections), the load balancer preserves the original client IP address by including it inside the `X-Forwarded-For HTTP` header. For TCP connections, the original client IP information is not preserved.

Preserving end-client protocol

IBM Cloud Load Balancer preserves the original protocol that is used by the client for front-end HTTP and HTTPS connections by including it inside the `X-Forwarded-Proto` HTTP header. This behavior does not apply to TCP protocols because the load balancer does not look at Layer-7 traffic when the TCP protocol is used.

Monitoring metrics by using IBM Cloud Monitoring

Support for the IBM Cloud Monitoring service ended 31 March 2020. IBM Cloud® Load Balancer monitoring is now provided with IBM Cloud Monitoring, a third-party monitoring tool that specializes in data aggregation, usage alerts, and in-depth visualizations. For more information, see IBM Cloud Monitoring.

Load balancers calculate the metrics and send those metrics to your IBM Cloud Monitoring instance, which reflects different types of use and traffic. You can visualize and analyze metrics from either the IBM Cloud Monitoring dashboard, or its API.

Metrics available by service plan

The supported monitoring metrics include:

- Active connections to your load balancer at a given time.
- Throughput of data passing through your load balancer over a given time.
- Connection rate, or an analysis of when more or less connections are made to your load balancer.

These metrics help track the traffic and usage patterns for your load balancers and can provide insight about peak traffic hours, usage dropouts, and overall usage patterns.

Each metric is composed of the following metadata types:

- Metric name - The name for the collected metric.
- Metric type - Determines whether the metric value is a counter metric or a gauge metric. Each of these metrics is of the type `gauge`, which represents a single numerical value that can arbitrarily fluctuate over time.
- Value type - A unit of measurement for a specific metric. Examples include bytes or counts. A value type of `none` means that the metric value represents individual occurrences of that metric type.
- Segment - How you want IBM Cloud Monitoring to divide and display the monitoring metrics.

Active connections

Active connections are the number of connections that are established on a load balancer at a specific time.

The active connection metric contains the following metadata:

Metadata	Description
Metric name	`ibm_cloud_load_balancer_active_connections`
Metric type	`gauge`
Value type	`none`
Segment by	`IBM Cloud Load Balancer appliance metrics` and `IBM Cloud Load Balancer listener metrics`

IBM Cloud Load Balancer active connections metrics metadata

Connection rate

Connection rate is the number of new incoming active connections per second to your load balancer.

Metadata	Description
Metric name	`ibm_cloud_load_balancer_connection_rate`
Metric type	`gauge`
Value type	`none`
Segment by	`IBM Cloud Load Balancer appliance metrics` and `IBM Cloud Load Balancer listener metrics`

IBM Cloud Load Balancer connection rate metric metadata

Throughput

Throughput is the amount of data that passes in and out of a load balancer over a period.

Metadata	Description
Metric name	`ibm_cloud_load_balancer_throughput`
Metric type	`gauge`
Value type	`byte`
Segment by	`IBM Cloud Load Balancer appliance metrics` or `IBM Cloud Load Balancer listener metrics`

IBM Cloud Load Balancer throughput metric metadata

Metric segmentation

You can split the data that IBM Cloud Monitoring presents into various visualizations in the IBM Cloud Monitoring dashboard, allowing views of different metrics based on your preferences. For example, if you have multiple load balancers or accounts with different load balancers in each account, you might want to focus on a particular listener (front-end protocol) port.

As an example, you can segment the `active connections` by `IBM Cloud Load Balancer listener port` to show how many active users are connected to the load balancer through each listener type. To illustrate this, let's assume that your load balancer has two different listener protocols one HTTP on port 80 and another for TCP on port 8080. The dashboard would contain different lines showing 10 users who are connected through HTTP on Port 80 in one color, and 6 users connected through TCP on port 8080 in another color.

Global attributes

The following attributes are available for segmenting all three of the IBM Cloud Monitoring metrics.

Attribute	Attribute Name	Attribute Description
Resource	ibm_resource	A load balancer's unique ID
Scope	ibm_scope	The account that is associated with a given load balancer
Service name	ibm_service_name	ibm-cloud-load-balancer

IBM Cloud Monitoring global attributes

Extra attributes

The following attributes are available to segment one or more of the global attributes. See the individual metrics for any segmentation options.

Attribute	Attribute Name	Attribute Description
IBM Cloud Load Balancer appliance metrics	ibm_cloud_load_balancer_appliance_ip	The metrics coming from the load balancer back end. Because the load balancer is highly available, multiple appliances support each load balancer for redundancy.
IBM Cloud Load Balancer listener metrics	ibm_cloud_load_balancer_listener_port	The metrics that are gathered from individual listeners and their ports. Configure the listeners in your load balancer settings. The monitoring metrics reflect the metrics coming from those listeners.

IBM Cloud Monitoring additional attributes

The displayed metrics contain a timestamp and the metric value for the time interval ending at that timestamp. You can specify different scopes, as well as the time interval over which to report the metrics.

The supported protocols include:

- HTTP
- HTTPS
- TCP

Specifying a listener port filters the metric by that listener. For example, if you don't specify a port, and the metric is `Throughput`, then IBM Cloud Monitoring reports the total throughput for all listener protocols. However, if the listener protocol is HTTP on port 80, then IBM Cloud Monitoring reports the throughput for HTTP traffic only.

You can also specify the time interval over which to report your metrics. Time intervals that are supported in the IBM Cloud Monitoring dashboard are:

- 10 seconds
- 1 minute
- 10 minutes
- 1 hour
- 6 hours
- 2 weeks
- Custom

The number of data points you can report is roughly the same for each time interval. For example, if the interval is 1 hour, then each data point represents 5 minutes of data. If the interval is 2 weeks, then each data point represents 24 hours of data.

Enabling metrics monitoring

To receive monitoring metrics, you must set up your IBM Cloud Monitoring instance.

To do so, follow these steps:

1. Navigate to the metrics monitoring portal, then click Create a monitoring instance.
2. Select a region for your IBM Cloud Monitoring instance.

   Tip: If you do not have an existing load balancer, see Using an elastic IBM Cloud Load Balancer for server load balancing to provision one.

   Important: The region must match the location of your existing load balancer.

3. Choose your pricing plan.

   Pricing plan details are explained in the selection window. Select the plan that best meets your requirements.

4. Provide a service name for your instance. It can be any name that you want, and has no impact on functions.

   Important: Do not create multiple IBM Cloud Monitoring instances with the same name.

5. Optionally, select a resource group. A resource group is a way to organize account resources in customizable groupings. Any account resource that is managed by using IBM Cloud Identity and Access Management (IAM) access control belongs to a resource group within your account.

   Note: If you do not have any pre-configured resource groups, or have no reason to share this resource selectively, use the default selection.

   Tip: If your account has multiple resource groups, you can choose which one has access to this IBM Cloud Monitoring instance. This allows you to have metrics available to some resource groups and not to others.

6. Select the Enable Platform Metrics checkbox. Select this to receive metrics from your load balancer.
7. Click Create. You are taken back to the monitoring metrics home page.

Within a few minutes, your new instance displays. You might have to refresh your browser to see it.

Working with the IBM Cloud Monitoring dashboard

To view and work with your IBM Cloud Monitoring metrics, follow these steps:

1. Navigate to the metrics monitoring portal.
2. Click View IBM Cloud Monitoring next to the service name of the IBM Cloud Monitoring instance you want to work with.

   Note: The first time that you access your IBM Cloud Monitoring instance, several windows display as part of the internal setup. Leave these selections with their default entries, and click through the pages until you reach the IBM Cloud Monitoring main page.

3. Select Dashboards on the left sidebar to open the IBM Load Balancer Monitoring Metrics dashboard. Then, clickDefault Dashboards > IBM > Load Balancer Monitoring Metrics. The default dashboard is not editable.
4. Three main metrics in the dashboard are shown: Throughput, Active Connections, and Connection Rate. To modify options and segment your metrics by load balancer ID or listener port, you must create a custom dashboard.

   Tip: You can choose what time window that you want to see your metrics by using the time selection bar. You can also zoom in and out for more granularity and drag the mouse to create a selection of a specific time window.

Creating a custom metrics dashboard

You can create your own dashboard to customize your monitoring metrics, such as viewing information about particular load balancers, or seeing only traffic that comes through HTTPS listeners.

To customize your dashboard, follow these steps:

1. Navigate to the metrics monitoring portal.
2. Click View IBM Cloud Monitoring next to the service name of the IBM Cloud Monitoring instance you want to work with. The dashboard opens.

3. On the left sidebar, select Dashboards. Then, click the green + sign in the page.

4. Select the Blank dashboard, then select the type of visual representation you want.

   IBM Cloud Monitoring offers eight different visualizations for your dashboard. Read the description for each visualization, then choose the one that best meets your requirements.

   > Note: Line ("View trends over time") is the easiest and most basic option. It is also the most frequently selected option. The following examples show a Line-based visualization.

5. Configure your custom dashboard.

   - In the Metrics field, enter `ibm_cloud` to display the IBM IBM Cloud Monitoring load balancer metrics. The ones discussed so far in this topic are `ibm_cloud_load_balancer_active_connections`, `ibm_cloud_load_balancer_connection_rate`, and `ibm_cloud_load_balancer_throughput`. After you click and add each metric, a new dropdown menu appears to select the next one. Repeat this process until you added all three.

   > Tip: You can monitor listener port traffic by enabling the `ibm_cloud_load_balancer_listener_port` metric.

   - You can choose a scope to display in your dashboard by clicking Override Dashboard Scope. For example, you can display the metrics for a particular load balancer.

   - You can also set a segment to compare metrics across the scope you defined. For example, you can look at throughput for a particular load balancer segmented by listener port.

6. Click Save for your new custom dashboard to be accessible.

   > Tip: By default, the dashboard begins with the name "blank dashboard". You can change the name by selecting Dashboards from the sidebar, then clicking the Pencil icon next to the name.

To return to the default IBM Cloud Monitoring dashboard at any time, selectDashboards > Default Dashboards > IBM > Load Balancer Monitoring Metrics.

Working with IBM Cloud Monitoring using the APIs

You can also work with the IBM Cloud Monitoring instance by using the metric query API. You might want to do this if you need raw data points or want to consume your metrics from a command-line interface rather than using the IBM Cloud Monitoring dashboard.

After you create your IIBM Cloud Monitoring instance, you must collect the following two pieces of information.

- The IBM Cloud Monitoring Monitor API token
- The endpoint of your IBM Cloud Monitoring IBM Cloud Monitoring instance

To collect this information and start working with your IBM Cloud Monitoring instance by using the metric query API, follow these steps:

1. Access the Monitoring home page, and click View IBM Cloud Monitoring next to the instance you want to work with. After the IBM Cloud Monitoring dashboard shows, select your Account Profile icon on the left sidebar, then select Settings. Your account settings display.

2. Your API token is an alphanumeric string that is located in the IBM Cloud Monitoring Monitor API Token field. Click the Copy button to the right of the key to transfer it to your clipboard.

   > Important: Do not share this key. Anyone who has this key has full access to your metrics.

3. To get the endpoint of your IBM Cloud Monitoring instance, navigate to your main IBM Cloud Monitoring dashboard in your browser. Then, select the URL to the dashboard, which appears similar to:

   ```
   $ https://us-south.monitoring.cloud.ibm.com/#/default-dashboard/ibm_cloud_load_balancer?last=3600
   ```

   The first part of the URL (in this case, `us-south.monitoring.cloud.ibm.com`) is your endpoint. Make note of it.

4. After you have both the API token and the endpoint, you can format your POST request. The following POST request is an example, with all the options that you can modify. These options are:

   - The IBM Cloud Monitoring Monitor API token.

   - The endpoint of your IBM Cloud Monitoring instance.

- The value for `ibm_resource` (this is the load balancer ID you want to see metrics for).

  > Tip: If you want to see this metric for all of your load balancers, do not enter a value for the `scope` attribute. For example, use `"scope" : ""`.

- The metric type that you want to see the results for. This example uses `ibm_cloud_load_balancer_throughput`, but `ibm_cloud_load_balancer_active_connections` and `ibm_cloud_load_balancer_connection_rate` are also valid options.
- The `from` and `to` attributes define the times to focus the scan, set in Epoch Time and in microseconds.
- The `sampling` and `value` attributes set the granularity of which data is returned in the POST request.

  Because a large volume of data is stored in IBM Cloud Monitoring, choosing the specific level of granularity is important. IBM Cloud Monitoring can return only 600 data points at any time with a given request. As a result, the `sampling` and `value` attributes are important. Leaving these two lines out of your request returns an aggregate sum over that time period instead.

  If the time range specified by `from` and `to` is large (for example, 4 days), but you define a `sampling` and `value` of 10 seconds, this means that you receive 4 days worth of data that is split into 10-second chunks. This is not a useful sampling due to the large amount of data returned. Specifying a larger chunk is recommended (for example, 1 hour instead of 10 seconds).

```
$ curl \
-H 'Authorization: Bearer <API_TOKEN>' \
-H 'Content-Type: application/json' \
https://us-south.monitoring.cloud.ibm.com/api/data/batch  \
-d '{
 "requests": [
    {
        "format": {
            "type": "data"
        },
        "scope": "ibm_resource=\"908461\"",
        "metrics": {
            "k0": "timestamp",
            “v1”: "ibm_cloud_load_balancer_throughput"
        },
        "time": {
            "from": 1584396900000000,
            "to": 1584402600000000,
              “sampling”: 600000000
        },
        "group": {
            "by": [
                {
                    "metric": "k0",
                    “value” : 600000000
                }
            ],
            "aggregations": {
                “v1”: "sum"
            },
            "groupAggregations": {
                “v1”: "sum"
            }
        }
    }
  ]
 }
```

Setting back-end encryption

Back-end encryption is supported to allow end-to-end data traffic encryption. Not only is the traffic between the load balancer and the client encrypted, but so is the traffic between the load balancer and the back-end server.

To enable back-end encryption:

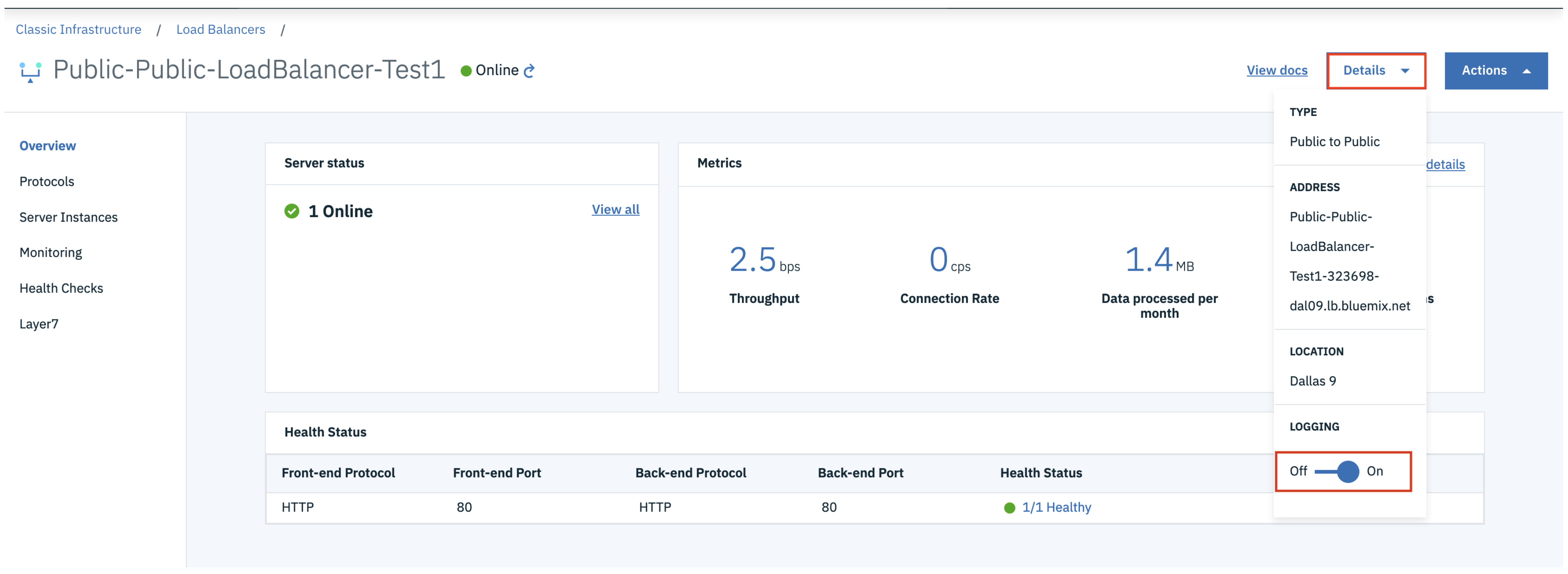
- If you add a new HTTPS protocol, set the front end and back end toHTTPS.
- For existing HTTPS protocols, set the back end to HTTPS.

Logging for IBM Cloud Load Balancer

Data and health check logs are valuable for debugging and maintenance purposes. With the data logging feature enabled, IBM Cloud Load Balancer forwards these logs to the [IBM Cloud Logs](https://cloud.ibm.com/docs/cloud-logs) under your account.

You can enable or disable this feature by:

- Creating a load balancer and setting this feature to on:



Data logging

- Using the API: `enableOrDisableDataLogs`.

Viewing logs in the IBM Cloud Logs service

Log in to the [IBM Cloud Logs](https://cloud.ibm.com/docs/cloud-logs) with your IBM Cloud account. Logs can be viewed from the IBM Cloud Logs instance. Refer to [Getting started with IBM Cloud Logs](https://cloud.ibm.com/docs/cloud-logs) for more information.

Note: Data logs are only sent if your Softlayer and IBM Cloud accounts are linked.

To create a IBM Cloud Logs instance, perform the following procedure:

1. Select the IBM Cloud account associated with your Softlayer account, then select Create a logging instance. The logging instance creation dialog shows.

2. Choose the region from the dropdown list that corresponds to the data center where you provisioned the load balancer.

   Tip: For a load balancer in SYD01, you would choose the region of Sydney.

   For information on the mapping between regions and data center, refer to [IBM Cloud global data centers](https://www.ibm.com/cloud/data-centers/).

3. You can use IBM Cloud Logs Routing, a platform service, to route platform logs in your account to a destination of your choice by configuring a tenant that defines where platform logs are sent. For more information, see [About Logs Routing](https://cloud.ibm.com/docs/logs-router).

Log output examples

Sub-system name = `Cloud Load Balancer`

Field	Type	Description
`msgTimestamp`	Required	The timestamp that indicates when the log was generated.
`logSourceCRN`	Required	The load balancer UUID can be obtained from `logSourceCRN`.
`message`	Required	Datapath log message.

Log output examples

The following output is an example of an IBM Cloud® Load Balancer data log:

```
{
datetime:2025-01-22T14:11:18.847230+00:00
msgTimestamp:2025-01-22T14:11:18.847230+00:00
host:loadbalancer-wdc04-323716-1137319-1367523
process:Cloud Load Balancer
message:Connect from 5.181.190.248:37328 to 52.116.108.53:80 (fabf4139-50b0-48ba-b399-277c09162832/HTTP)
logSourceCRN:crn:v1:bluemix:public:cloud-load-balancer:us-east:a/5c59f412bc914beb390b080e07e5e6a2:6239a1a7-6da4-4370-881c-2dec099b8623::
saveServiceCopy:false
platformSource:Cloud Load Balancer
}
```

Details of the datapath log fieds are:

- `msgTimestamp` is Coordinated Universal Time.
- `loadbalancer-wdc04-323716-1137319-1367523` is the load balancer name, and `wdc04` is the data center.
- `323716` is the account ID. `1137319` is the load balancer ID. `1367523` is the load balancer instance ID.

The following output is an example of a health check log seen in the IBM Cloud Logs service:

```
{
  datetime:2025-01-16T11:32:05.100888+00:00
  msgTimestamp:2025-01-16T11:32:05.100888+00:00
  host:loadbalancer-wdc04-323716-1137319-1367279
  process:Cloud Load Balancer
  message:Health check for server ea3b5aa1-cd85-4374-8387-cbf155d35643/c8d09bae-4d1c-4579-8af1-6e2fa75e81b5-10.171.112.26 failed, reason: Layer4 timeout, check
duration: 5001ms, status: 0/2 DOWN.
  logSourceCRN:crn:v1:bluemix:public:cloud-load-balancer:us-east:a/5c59f412bc914beb390b080e07e5e6a2:6239a1a7-6da4-4370-881c-2dec099b8623::
  saveServiceCopy:false
  platformSource:Cloud Load Balancer
  }
```

Selecting Point of Delivery (PoD) based locations by using the CLI and API

Each physical data center location in IBM Cloud® is divided into multiple, discrete networking LANs, otherwise known as PoDs. Each of these has its own separate network infrastructure, a limited number of VLANs, as well as public and private IP subnets. These VLANs and subnets are shared by all clients that have compute capacity that is provisioned within that PoD space.

Currently, you can use the UI to select the deploying data center location at the specific zone, such as `tok02`. You can also select the deploying VLAN and subnet when you have the existing one. However, you cannot specify PoD-based locations except for a gateway appliance and VLAN in the console.

Because you cannot specify PoD locations, and there is no VLAN or subnets, your provisioning can be distributed over multiple PoDs, which can be problematic.

To prevent this issue, you can order a "Premium VLAN" in advance, then specify it as your network VLAN; however, this option incurs an extra cost.

A better option is to use the CLI and API.

Using the CLI and API to specify PoD

You can use the CLI and API to specify PoD using the following option:

```
--extras '{"virtualGuests": [{"hostname": "test", "domain": "softlayer.com", "primaryBackendNetworkComponent": {"router": {"id": 1673467}}}]}'
```

The `primaryBackendNetworkComponentId` can be obtained through the following API call:

```
slcli call-api SoftLayer_Network_Pod getAllObjects
```

Layer 7 support

Layer 7 load balancing

The IBM Cloud® Load Balancer service distributes traffic among multiple server instances, including bare-metal and virtual-server instances, by using Layer 7 (application layer) data.

- The data traffic to be distributed is classified by using policies and rules.
- Policies define what action to take when the data traffic matches all the rules that are associated with a policy.
- Layer 7 (L7) load balancing is supported for HTTP and HTTPS traffic only.

Layer 7 policies and rules

A Layer 7 policy is associated with a front-end application port. Multiple policies can be associated with a front-end port.

- These policies are evaluated in order, based on the priority assigned to each policy.
- An associated action is taken when the policy is matched.
- Each L7 rule is associated with a policy.
- If all the rules that are associated with the policy evaluate to `true`, the policy is matched, so the associated action is taken.

For more information, see L7 Policy and Rules.

Layer 7 pools

Each Layer 7 load balancer pool contains one or more logical server instances.

- Each logical server instance is identified by an IP address and port number.
- Each pool has a health monitor associated with it, which monitors the health of all the servers in the pool.
- A pool can be configured for session persistence.
- Use the source IP address of the client or HTTP cookie to configure session persistence.

For more information, see L7 Pools.

Layer 7 policy

A Layer 7 (L7) policy is used to classify traffic by matching its L7 information with L7 rules, and then taking specific actions if those rules match.

- A policy is applied to a front-end application port (protocol).
- Multiple policies can be applied to the same protocol.

Since multiple policies can be applied to a protocol, a priority is associated with each policy.

- Policies with the lowest set priority are evaluated first.
- If the rules associated with the policy do not match the traffic, the next lowest policy on the priority list is evaluated.

If the traffic does not match any of the policy rules, the traffic is redirected to a default pool, which is the pool that was configured when the basic load balancer was deployed.

Each policy is associated with an action that runs when all rules in the policy match the traffic.

The actions can be:

- Reject
- Redirect to HTTPS
- Redirect to URL
- Redirect to pool

Policies set to `reject` are evaluated first.

If the action is set to `REDIRECT_HTTPS`, then the HTTP traffic redirects to the HTTPS listener port. Only one configuration of this action is supported per listener. This action cannot configure L7 rules, and you must configure it using the API. You can configure the L7 policy by using the action `REDIRECT_HTTPS` with the `addL7Policies` method of the `SoftLayer_Network_LBaaS_L7Policy` service. A `redirectUrl` data type provides the HTTPS listener's universal unique identifier (UUID) for HTTPS redirection.

The `Redirect to https` policy is evaluated after a `Reject`. If this policy exists, then any existing `Redirect to url` and `Redirect to pool` policies do not apply. Also, if there are any existing `Redirect to https` policies, then you cannot create any new `Redirect to url` and `Redirect to pool` policies.

If no HTTPS redirect policy exists, then any policies set to `Redirect to url` are evaluated after `Reject`.

Finally, policies set to `Redirect to pool` are evaluated last.

Within each action category, policies are evaluated in ascending order of priority (lowest to highest). Only one action of `REDIRECT_HTTPS` is allowed per listener, and it takes precedence over all other policies except `REJECT`. As a result, the concept of "priority" does not apply.

Layer 7 policy properties

Property	Description
Name	The name of the policy. Each policy must have a unique name.
Action	The action to take when the rules match. The actions are `REJECT`, `REDIRECT_HTTPS`, `REDIRECT_URL`, and `REDIRECT_POOL`. `REDIRECT_HTTPS` action is supported by API only.
Priority	Within each action category, policies are evaluated in ascending order of priority. This field is not applicable for `REDIRECT_HTTPS` because only one such action is applicable per listener.
Redirect URL	The URL to which traffic is redirected, if the action is set to `REDIRECT_URL`. Also, the HTTPS listener UUID to which traffic is to be redirected, if the action is set to `REDIRECT_HTTPS`.
Redirect L7 Pool	The pool of servers to which traffic is sent, if the action is set to `REDIRECT_POOL`.
Protocol	The front-end application port to which the policy is applied.

Layer 7 properties

Layer 7 rule

Layer 7 rules define a portion of the incoming traffic that is to be matched with specific values.

- Adding L7 rules is not allowed for any L7 policy with `REDIRECT_HTTPS` action.
- If the incoming traffic matches the specified value of a rule, then the rule evaluates to `true`.
- Layer 7 rules are always associated with a Layer 7 policy. Multiple Layer 7 rules can be associated with the same Layer 7 policy.
- If multiple rules are associated with a policy, then each rule is evaluated to be `true` or `false`.
- If all the rules that are associated to a policy evaluate to `true`, then the policy action is applied to the request. Otherwise, the load balancer evaluates the next policy.

Rules have types, which indicate the portion of the Layer 7 traffic to be matched with the rule.

Type	Field to be extracted and evaluated
`HOST_NAME`	The hostname part of the URL (for example, `api.my_company.com`)
`FILE_TYPE`	The end of the URL, representing the file type (for example, `jpg`)
`HEADER`	A field in the HTTP header
`COOKIE`	A named cookie in the HTTP header
`PATH`	The part of the URL that follows the hostname (for example, `/index.html`)

Layer 7 rules

Rules also have a comparison type, which indicates how they are to be evaluated.

Comparison Type	Type of evaluation

REGEX	Match the extracted field (for example, hostname) with the supplied regular expression
STARTS_WITH	Verify whether the extracted field starts with the supplied string
ENDS_WITH	Verify whether the extracted field ends with the supplied string
CONTAINS	Verify whether the extracted field contains the supplied string
EQUAL_TO	Verify whether the extracted field is identical to the supplied string

Comparison types

Tip: Not all rule types support all comparison types. For example, if you are using `FILE_TYPE`, it is best to use comparison types `REGEX` and `ENDS_WITH`.

Layer 7 rule properties

Property	Description
Type	Specifies the type of rule. Rule types can be HOST_NAME, FILE_TYPE, HEADER, COOKIE, or PATH.
Comparison Type	Comparison types are used in association with the rule type, key, and value to define a rule and classify traffic. Comparison types can be: REGEX, STARTS_WITH, ENDS_WITH, CONTAINS, and EQUAL_TO.
Key	The description key for the rule types HEADER and COOKIE.
Value	For the rule types HEADER and COOKIE, the value is compared against the key.
Invert	If you set the value to 1, the value of this L7 rule comparison is set totrue whenever the specified rule is not matched.
Layer 7 Policy ID	The unique identifier of the policy to which the rules are attached.

Layer 7 rule properties

Layer 7 rules

Layer 7 rules define a portion of the incoming traffic that is to be matched with specific values.

- If the incoming traffic matches the specified value of a rule, then the rule evaluates to `true`.
- Layer 7 rules are always associated with a Layer 7 policy. Multiple Layer 7 rules can be associated with the same Layer 7 policy.
- If multiple rules are associated with a policy, then each rule is evaluated to be `true` or `false`.
- If all the rules that are associated to a policy evaluate to `true`, then the policy action is applied to the request. Otherwise, the load balancer evaluates the next policy.

Rules have types, which indicate the portion of the Layer 7 traffic to be matched with the rule.

Type	Field to be extracted and evaluated
HOST_NAME	The hostname part of the URL (for example, api.my_company.com)
FILE_TYPE	The end of the URL, representing the file type (for example, jpg)
HEADER	A field in the HTTP header
COOKIE	A named cookie in the HTTP header
PATH	The part of the URL that follows the hostname (for example, /index.html)

Rule types

Rules also have a comparison type, which indicates how they are to be evaluated. Rules can have following comparison types:

Comparison Type	Type of evaluation
`REGEX`	Match the extracted field (for example, `hostname`) with the supplied regular expression
`STARTS_WITH`	Verify whether the extracted field starts with the supplied string
`ENDS_WITH`	Verify whether the extracted field ends with the supplied string
`CONTAINS`	Verify whether the extracted field contains the supplied string
`EQUAL_TO`	Verify whether the extracted field is identical to the supplied string

Rule comparison types

Tip: Not all rule types support all comparison types. For example, if you are using `FILE_TYPE`, it is best to use comparison types `REGEX` and `ENDS_WITH`.

Layer 7 rule properties

Property	Description
Type	Specifies the type of rule. Rule types can be `HOST_NAME`, `FILE_TYPE`, `HEADER`, `COOKIE`, or `PATH`.
Comparison Type	Comparison types are used in association with the rule type, key, and value to define a rule and classify traffic. Comparison types can be: `REGEX`, `STARTS_WITH`, `ENDS_WITH`, `CONTAINS`, and `EQUAL_TO`.
Key	The description key for the rule types `HEADER` and `COOKIE`.
Value	For the rule types `HEADER` and `COOKIE`, the value is compared against the key.
Invert	If you set the value to 1, the value of this L7 rule comparison is set to`true` whenever the specified rule is not matched.
Layer 7 Policy ID	The unique identifier of the policy to which the rules are attached.

Rule properties

Layer 7 pool

A Layer 7 (L7) pool is a logical grouping of the servers (members) for handling incoming requests.

The Layer 7 load balancing feature can direct the incoming traffic to different back-end pools based on the policies and rules. This feature is supported by associating multiple L7 pools with a load balancer. Layer 7 pools are used with the Layer 7 policies whose action is `redirect to pool`.

L7 pools support both HTTP and HTTPS as the back-end protocol.

Session persistence

Session persistence can be configured for each Layer 7 pool. For more information, see [Advanced traffic management with IBM Cloud Load Balancer](#).

Layer 7 members

Back-end servers that are associated with a Layer 7 pool are called Layer 7 members.

The same back-end server can be added multiple times to L7 pools, by specifying a different port number each time.

Configure health checks

The health check definition is mandatory for each Layer 7 pool. The system pre-populates a default health check configuration for L7 pools.

You can customize these settings to suit your application needs:

- Interval - The interval in seconds between two consecutive health check attempts.
- Timeout - The maximum amount of time the system waits for a response to the health check request.
- MaxRetries - The maximum number of extra health check attempts that the system makes before declaring the service unhealthy.
- UrlPath - The HTTP URL path for the Layer 7 health check.

IBM Cloud Load Balancer API reference

The SoftLayer application programming interface (API) is the development interface that gives developers and system administrators direct interaction with the SoftLayer back-end system.

The SoftLayer API powers many of the features in the Customer Portal. Generally, if an interaction is possible in the Customer Portal, it can also be run in the API. Because you can programmatically interact with all portions of the SoftLayer environment within the API, you can use the API to automate tasks.

The SLAPI is a Remote Procedure Call system. Each call involves sending data towards an API endpoint and receiving structured data in return. The format used to send and receive data with the SLAPI depends on which implementation of the API you choose. The SLAPI currently uses SOAP, XML-RPC or REST for data transmission.

For more information about the SoftLayer API, IBM Cloud® Load Balancer Service APIs, see the following resources in the SoftLayer Development Network:

- Getting Started with the SoftLayer API
- SoftLayer_Product_Package API
- SoftLayer_Network_LBaaS_LoadBalancer API
- SoftLayer_Network_LBaaS_Listener API
- SoftLayer_Network_LBaaS_Member API
- SoftLayer_Network_LBaaS_HealthMonitor API
- SoftLayer_Network_LBaaS_SSLCipher API
- SoftLayer_Network_LBaaS_L7Policy API
- SoftLayer_Network_LBaaS_L7Rule API
- SoftLayer_Network_LBaaS_L7Pool API
- SoftLayer_Network_LBaaS_L7Member API

The following examples are using softlayer-python client.

Set an api username and apikey as following for each example in case that you don't have a `~/.softlayer` file that is configured in your environment.

```
$ client = SoftLayer.Client(username='set me', api_key='set me')
```

Creating a load balancer

The following example retrieves the package ID, subnet ID, and prices for a "Load Balancer As A Service (LBaaS)" package, builds the order data and places the order.

```
import SoftLayer
from pprint import pprint

class LBaaSExample():
    def __init__(self):
        self.client = SoftLayer.Client()

    def get_package_id(self, pkg_name):
        """Returns the packageId for the LBaaS"""
        _filter = {"name":{"operation": pkg_name}}

        pkg_list = self.client['Product_Package'].getAllObjects(filter=_filter)

        return pkg_list[0]['id']


    def get_item_prices(self, pkg_id):
        """Returns the standard prices"""

        item_list = self.client['Product_Package'].getItems(id=pkg_id)
        prices = []

        for item in item_list:
            price_id = [p['id'] for p in item['prices']
                        if not p['locationGroupId']][0]
            prices.append(price_id)
        return prices

    def get_subnet_id(self, datacenter):
        """Find and returns the first subnet in the datacenter"""

        _filter = {"privateSubnets": {
                        "networkVlan": {
                            "primaryRouter": {
                                "datacenter": {
                                    "regions": { "keyname": { "operation": datacenter}}
                                }
                            },
                            "type": { "keyName": { "operation": "STANDARD" } }
                        },
                        "routingTypeKeyName": {"operation": "PRIMARY" }
                    }
                }

        subnets = self.client['Account'].getPrivateSubnets(filter=_filter)

        return subnets[0]['id']

    def order_lbaas(self, pkg_name, datacenter, name, desc, protocols,
                    subnet_id=None, public=False, verify=False):
        """Allows to order a Load Balancer"""

        package_id = self.get_package_id(pkg_name)
        prices = self.get_item_prices(package_id)

        # Find and select a subnet id if it was not specified.
        if subnet_id is None:
            subnet_id = self.get_subnet_id(datacenter)

        # Build the configuration of the order
        orderData = {
            'complexType': 'SoftLayer_Container_Product_Order_Network_LoadBalancer_AsAService',
            'name': name,
            'description': desc,
            'location': datacenter,
            'packageId': package_id,
            'useHourlyPricing': True,       # Required since LBaaS is an hourly service
            'prices': [{'id': price_id} for price_id in prices],
            'protocolConfigurations': protocols,
            'subnets': [{'id': subnet_id}]
        }

        try:
            # If verify=True it checks your order for errors.
            # It orders the lbaas if False.
            if verify:
                response = self.client['Product_Order'].verifyOrder(orderData)
            else:
                response = self.client['Product_Order'].placeOrder(orderData)

            return response
        except SoftLayer.SoftLayerAPIError as e:
            print("Unable to place the order: %s, %s" % (e.faultCode, e.faultString))


if __name__ == "__main__":
    lbaas = LBaaSExample()

    package_name = 'Load Balancer As A Service (LBaaS)'
    location = 'MEXICO'
    name = 'My-LBaaS-name'
    description = 'A description sample'

    # Set False for private network
    is_public = True

    protocols = [
        {
            "backendPort": 80,
            "backendProtocol": "HTTP",
            "frontendPort": 8080,
            "frontendProtocol": "HTTP",
            "loadBalancingMethod": "ROUNDROBIN",    # ROUNDROBIN, LEASTCONNECTION, WEIGHTED_RR
            "maxConn": 1000
        }
    ]

    # remove verify=True to place the order
    receipt = lbaas.order_lbaas(package_name, location, name, description,
                                protocols, public=is_public, verify=True)

    pprint(receipt)
```

Order a public to private load balancer with specified public subnet

The following example focuses on the order data that is required to create a load balancer with a specified public subnet. It is only valid for public to private load balancer types, and `useSystemPublicIpPool` must be `false`.

```
import SoftLayer
from prettytable import PrettyTable

class LBaasExample():
  ...
  def order_lbaas(self, pkg_name, datacenter, name, desc, protocols,
                  subnet_id=None, public_subnet_id=None, public=False, verify=False):
      """Allows to order a Load Balancer"""

      package_id = self.get_package_id(pkg_name)
      prices = self.get_item_prices(package_id)

      # Build the configuration of the order
      orderData = {
          'complexType': 'SoftLayer_Container_Product_Order_Network_LoadBalancer_AsAService',
          'name': name,
          'description': desc,
          'location': datacenter,
          'packageId': package_id,
          'useHourlyPricing': True,       # Required since LBaaS is an hourly service
          'prices': [{'id': price_id} for price_id in prices],
          'protocolConfigurations': protocols,
          'subnets': [{'id': subnet_id}],
          'useSystemPublicIpPool': False,
          'type': 1, # public to private load balancer
          'publicSubnets': [{'id': publicSubnet_id}]
      }

      try:
          # If verify=True it checks your order for errors.
          # It orders the lbaas if False.
          if verify:
              response = self.client['Product_Order'].verifyOrder(orderData)
          else:
              response = self.client['Product_Order'].placeOrder(orderData)

          return response
      except SoftLayer.SoftLayerAPIError as e:
          print("Unable to place the order: %s, %s" % (e.faultCode, e.faultString))
    ...
```

Retrieving details on load balancers

List all load balancers

```
import SoftLayer
from prettytable import PrettyTable
```

```
class LBaasExample():
    def __init__(self):
        client = SoftLayer.Client()
        self.lbaas_service = client['Network_LBaaS_LoadBalancer']

    def get_list(self, dc=None):
        _filter = None
        lbaas_list = None

        # Use filters if datacenter is set
        if dc:
            _filter = {"datacenter":{"name":{"operation": dc}}}

        try:
            # Retrieve load balancer objects
            lbaas_list = self.lbaas_service.getAllObjects(filter=_filter)
        except SoftLayer.SoftLayerAPIError as e:
            print("Unable to get the LBaaS list: %s, %s" % (e.faultCode, e.faultString))

        return lbaas_list


if __name__ == "__main__":
    table = PrettyTable(['ID','UUID','Name', 'Description',
                         'Address', 'Type', 'Location', 'Status'])

    # To find all lbaas in Mexico
    datacenter = "mex01"

    lbaas = LBaasExample()
    # remove dc=datacenter to retrieve all the load balancers in the account
    lbaas_list = lbaas.get_list(dc=datacenter)

    # add lbaas data to the table
    for i in lbaas_list:
        isPublic = "Public" if i['isPublic'] == 1 else "Private"
        table.add_row([i['id'], i['uuid'], i['name'], i['description'], i['address'],
                       isPublic,i['datacenter']['longName'],i['operatingStatus']])

    print (table)
```

Retrieve details of a specific load balancer

```
import SoftLayer
from pprint import pprint

# Your load balancer UUID
uuid = 'set me'
# mask to retrieve the load balancer's listeners and healthMonitors
_mask = "mask[listeners, healthMonitors]"

# Create the api client
client = SoftLayer.Client()
lbaas_service = client['Network_LBaaS_LoadBalancer']

try:
    # Retrieve a specific load balancer object
    details = lbaas_service.getLoadBalancer(uuid, mask=_mask)
    pprint(details)
except SoftLayer.SoftLayerAPIError as e:
    print("Unable to retrieve LBaaS details: %s, %s" % (e.faultCode, e.faultString))
```

Updating a load balancer

Add a member

```
import SoftLayer
```

```
from pprint import pprint

# Your load balancer UUID
uuid = 'set me'

# Update with the correct IP addresses
serverInstances = [
    { "privateIpAddress": "10.131.11.46", "weight": 80 },
    { "privateIpAddress": "10.131.11.6" } # Default weight=50
]

# Create the api client
client = SoftLayer.Client()
member_service = client['Network_LBaaS_Member']

_mask = "mask[members]"

try:
    response = member_service.addLoadBalancerMembers(uuid, serverInstances, mask=_mask)
    pprint(response)
except SoftLayer.SoftLayerAPIError as e:
    print("Unable to add members: %s, %s" % (e.faultCode, e.faultString))
```

Add a protocol

```
import SoftLayer
from pprint import pprint

# Your load balancer UUID
uuid = 'set me'

# New protocols to add
protocolConfigurations = [
    {
        "backendPort": 1350,
        "backendProtocol": "TCP",
        "frontendPort": 1450,
        "frontendProtocol": "TCP",
        "loadBalancingMethod": "WEIGHTED_RR",    # ROUNDROBIN, LEASTCONNECTION, WEIGHTED_RR
        "maxConn": 500,
        "sessionType": "SOURCE_IP"
    },
    {
        "backendPort": 1200,
        "backendProtocol": "HTTP",
        "frontendPort": 1150,
        "frontendProtocol": "HTTP",
        "loadBalancingMethod": "ROUNDROBIN",    # ROUNDROBIN, LEASTCONNECTION, WEIGHTED_RR
        "maxConn": 1000,
        "sessionType": "SOURCE_IP",
        "serverTimeout": 70,
        "clientTimeout": 70
    }
]

# Create the api client
client = SoftLayer.Client()
listener_service = client['Network_LBaaS_Listener']

_mask = "mask[listeners]"

try:
    response = listener_service.updateLoadBalancerProtocols(uuid, protocolConfigurations, mask=mask)
    pprint(response)
except SoftLayer.SoftLayerAPIError as e:
    print("Unable to add protocols: %s, %s" % (e.faultCode, e.faultString))
```

Update a protocol

```
import SoftLayer
from pprint import pprint

# Your load balancer UUID
uuid = 'set me'

# New protocols to add
protocolConfigurations = [
    {
        "listenerUuid": "69fad83a-e850-4b72-a4d3-af94d5bf5437",
        "serverTimeout": 60,
        "clientTimeout": 60
    },
    {
        "listenerUuid": "e4b8cfd0-1e27-4d3e-a8ed-595b198cd683",
        "frontendPort": 1450,
        "maxConn": 1002,
        "serverTimeout": 80,
        "clientTimeout": 80
    }
]

# Create the api client
client = SoftLayer.Client()
listener_service = client['Network_LBaaS_Listener']

_mask = "mask[listeners]"

try:
    response = listener_service.updateLoadBalancerProtocols(uuid, protocolConfigurations, mask=mask)
    pprint(response)
except SoftLayer.SoftLayerAPIError as e:
    print("Unable to add protocols: %s, %s" % (e.faultCode, e.faultString))
```

Canceling a load balancer

```
import SoftLayer

# Your load balancer UUID
uuid = 'set me'

# Create the api client
client = SoftLayer.Client()
lbaas_service = client['Network_LBaaS_LoadBalancer']

try:
    result = lbaas_service.cancelLoadBalancer(uuid)

    if result:
        print("The cancellation request is accepted")
except SoftLayer.SoftLayerAPIError as e:
    print("Failed to cancel load balancer: %s, %s" % (e.faultCode, e.faultString))
```

Viewing monitoring metrics of load balancers

Get throughput of HTTP traffic

```
import SoftLayer
from pprint import pprint

# UUID of load balancer
uuid = 'set me'

# Avaiable options: Throughput, ActiveConnections, and ConnectionRate.
metric_name = 'Throughput'

# Options are: 1hour, 6hours, 12hours, 24hour, 1week, or 2weeks
```

```
time_range = '1hour'

# UUID of the listener whose throughput your requesting
protocol_uuid = 'set me'

# Create the api client
client = SoftLayer.Client()
lbaas_service = client['Network_LBaaS_LoadBalancer']

try:
    # Remove protocol_uuid to retrieve the traffic metrics of the entire load balancer
    time_series = lbaas_service.getListenerTimeSeriesData(uuid, metric_name,
                                                time_range, protocol_uuid)

    pprint(time_series)
except SoftLayer.SoftLayerAPIError as e:
    print("Unable to retrieve the traffic: %s, %s" % (e.faultCode, e.faultString))
```

Layer 7 APIs

Create Multiple L7 policies and L7 rules

```
import SoftLayer
from pprint import pprint

# UUID of the listener
listener_uuid = "set me"

# Build the rules
l7RuleArray1 = [
    {
        "type": "HEADER",
        "comparisonType": "EQUAL_TO",
        "key": "headerkey",
        "value": "header_key3",
        "invert": 0
    },
    {
        "type": "PATH",
        "comparisonType": "STARTS_WITH",
        "value": "/secret_key",
        "invert": 0
    }
]

# Bulk policies and rules configuration
policies_rules = [
    {
        "l7Policy": {
            "name": "traf_test1",
            "action": "REDIRECT_URL",
            "priority": 101,
            "redirectUrl": "http://example.com"
        },
        "l7Rules": l7RuleArray1
    }
]

client = SoftLayer.Client()
networkLBaaSL7PolicyService = client['SoftLayer_Network_LBaaS_L7Policy']

try:
    result = networkLBaaSL7PolicyService.addL7Policies(listener_uuid, policies_rules)
    pprint(result)
except SoftLayer.SoftLayerAPIError as e:
    print("Unable to addL7Policies: %s, %s " % (e.faultCode, e.faultString))
```

Update Layer 7 policy

```
import SoftLayer
from pprint import pprint

# The ID of policy you want to update
networkLBaaSL7PolicyId = 11111111

# Update policy configuration by specifying the variable name and value , e.g. "<name>": "<value>"
policyConfiguration = {
    "name": "<value>"
}

client = SoftLayer.Client()
networkLBaaSL7PolicyService = client['SoftLayer_Network_LBaaS_L7Policy']

try:
    result = networkLBaaSL7PolicyService.editObject(policyConfiguration, id=networkLBaaSL7PolicyId)
    pprint(result)
except SoftLayer.SoftLayerAPIError as e:
    print("Unable to update the Layer 7 policy: %s, %s" % (e.faultCode, e.faultString))
```

Add rules to a Layer 7 policy

```
import SoftLayer
from pprint import pprint

# UUID of the Layer 7 policy
policyUuid = "set me"

# New rules to add
rules = [
    {
        "type": "FILE_TYPE",
        "comparisonType": "CONTAINS",
        "value": "some_value",
        "invert": 1
    },
    {
        "type": "PATH",
        "comparisonType": "EQUAL_TO",
        "value": "some_value",
        "invert": 0
    }
]

client = SoftLayer.Client()
networkLBaaSL7RuleService = client['SoftLayer_Network_LBaaS_L7Rule']

try:
    result = networkLBaaSL7RuleService.addL7Rules(policyUuid, rules)
    pprint(result)
except SoftLayer.SoftLayerAPIError as e:
    print("Unable to add rules to a Layer 7 policy: %s, %s " % (e.faultCode, e.faultString))
```

Update multiple Layer 7 rules that are attached to the same Layer 7 policy

```
# For nice debug output:
import SoftLayer
from pprint import pprint

# UUID of the Layer 7 policy
policyUuid = "set me"

# New rules to add
rules = [
    {
        'uuid': '<UUID of the L7 Rule being updated>',
        'type': 'FILE_TYPE',
        'comparisonType': 'CONTAINS',
        'value': 'some_newvalue',
        'invert': 1
    },
    {
        'uuid': '<UUID of the L7 Rule being updated>',
        'type': 'PATH',
        'comparisonType': 'EQUAL_TO',
        'value': 'some_newvalue2',
        'invert': 0
    }
]

client = SoftLayer.Client()
networkLBaaSL7RuleService = client['SoftLayer_Network_LBaaS_L7Rule']

try:
    result = networkLBaaSL7RuleService.updateL7Rules(policyUuid, rules)
    pprint(result)
except SoftLayer.SoftLayerAPIError as e:
    print("Unable to update multiple Layer 7 Rules attached to the same Layer 7 policy: %s, %s"
          % (e.faultCode, e.faultString))
```

Create a Layer 7 pool with servers, health monitoring, and session affinity

```
import SoftLayer
from pprint import pprint

# UUID of load balancer to be updated
loadBalancerUuid = "set me"

# Layer 7 pool to be added
l7Pool = {
    'name': 'image_pool',
    'protocol': 'HTTP',  # only supports HTTP
    'loadBalancingAlgorithm': 'ROUNDROBIN'
}

# Layer 7 Backend servers to be added
l7Members = [
    {
        'address': '10.131.11.46',
        'port': 80,
        'weight': 10
    },
    {
        'address': '10.130.188.130',
        'port': 81,
        'weight': 11
    }
]

# Layer 7 Health monitor to be added
l7HealthMonitor = {
    'interval': 10,
    'timeout': 5,
    'maxRetries': 3,
    'urlPath': '/'
}

# Layer 7 session affinity to be added. Only supports SOURCE_IP as of now
l7SessionAffinity = {
    'type': 'SOURCE_IP'
}

client = SoftLayer.Client()
networkLBaaSL7PoolService = client['SoftLayer_Network_LBaaS_L7Pool']

try:
    result = networkLBaaSL7PoolService.createL7Pool(loadBalancerUuid, l7Pool, l7Members,
                                                    l7HealthMonitor, l7SessionAffinity)
    pprint(result)
```

```
except SoftLayer.SoftLayerAPIError as e:
    print("Unable to create a Layer 7 Pool with servers: %s, %s"
          % (e.faultCode, e.faultString))
```

Update a Layer 7 pool along with health monitoring and session affinity

```
import SoftLayer
from pprint import pprint

# UUID of L7 pool to be updated
l7PoolUuid = 'set me'

# New Layer 7 pool values to be updated
l7Pool = {'loadBalancingAlgorithm': 'LEASTCONNECTION'}

# New Layer 7 Health monitor values to be updated
l7HealthMonitor = {'urlPath': '/index'}

# New Layer 7 session affinity values to be updated.
# If not given it deletes the existing session affinity
# If given and session affinity doesn't exist, it creates one.
l7SessionAffinity = {'type': 'SOURCE_IP'}

client = SoftLayer.Client()
networkLBaaSL7PoolService = client['SoftLayer_Network_LBaaS_L7Pool']

try:
    result = networkLBaaSL7PoolService.updateL7Pool(l7PoolUuid, l7Pool,
                                                     l7HealthMonitor, l7SessionAffinity)
    pprint(result)
except SoftLayer.SoftLayerAPIError as e:
    print("Unable to update a Layer 7 pool along with health monitoring and session affinity: %s, %s"
          % (e.faultCode, e.faultString))
```

Add servers to a Layer 7 Pool

```
import SoftLayer
from pprint import pprint

# UUID of the L7 pool to which members should be added.
l7PoolUuid = 'set me'

# Back-end servers to be added
memberInstances = [
    {
        'address': '10.131.11.46',
        'port': 80,
        'weight': 10
    },
    {
        'address': '10.130.188.130',
        'port': 81,
        'weight': 11
    }
]

client = SoftLayer.Client()
networkLBaaSL7MemberService = client['SoftLayer_Network_LBaaS_L7Member']

try:
    result = networkLBaaSL7MemberService.addL7PoolMembers(l7PoolUuid, memberInstances)
    pprint(result)
except SoftLayer.SoftLayerAPIError as e:
    print("Unable to add servers to a Layer 7 Pool: %s, %s" % (e.faultCode, e.faultString))
```

Update severs belonging to a Layer 7 pool

```
import SoftLayer
```

```
from pprint import pprint

# UUID of the L7 pool who's member we need to update
l7PoolUuid = 'set me'

# Back-end servers to be added
members = [
    {
        'uuid': '1e433123-ceae-4bbd-a4e3-2539ceb8b46f',  # UUID of the member to be updated.
        'address': '10.73.67.83',
        'port': 90,
        'weight': 10
    },
    {
        'uuid': 'bd3524db-26c8-4662-982e-04a68a420fba',
        'address': '10.73.67.83',
        'port': 91,
        'weight': 11
    }
]

client = SoftLayer.Client()
networkLBaaSL7MemberService = client['SoftLayer_Network_LBaaS_L7Member']

try:
    result = networkLBaaSL7MemberService.updateL7PoolMembers(l7PoolUuid, members)
    pprint(result)
except SoftLayer.SoftLayerAPIError as e:
    print("Unable to update severs belonging to a Layer 7 pool: %s, %s"
          % (e.faultCode, e.faultString))
```

Enable or disable data logs for a specific load balancer

```
$ from zeep import Client, xsd

# Username and apikey for SLAPI call
username = ''
apiKey = ''
# UUID of the load balancer
uuid = ''
# enabled = 1 when enable, enable = 0 when disable
enabled = 0

# WSDL for SoftLayer_Network_LBaaS_LoadBalancer API
wsdl = 'https://api.softlayer.com/soap/v3/SoftLayer_Network_LBaaS_LoadBalancer?wsdl'
client = Client(wsdl)

# XSD for authentication
xsdUserAuth = xsd.Element(
    '{http://api.softlayer.com/soap/v3/}authenticate',
    xsd.ComplexType([
        xsd.Element('{http://api.softlayer.com/soap/v3/}username', xsd.String()),
        xsd.Element('{http://api.softlayer.com/soap/v3/}apiKey', xsd.String())
    ])
)

# Create XSD value objects
userAuthValue = xsdUserAuth(username=username, apiKey=apiKey)

# Enable or disable data logs for a specific load balancer
loadbalancer = client.service.enableOrDisableDataLogs(_soapheaders=[userAuthValue], uuid=uuid, enabled=enabled)
#print loadbalancer
```

Known issues and limitations with IBM Cloud Load Balancer

Provides information on the currently known issues and limitations with the IBM Cloud® Load Balancer service.

Known issues

The IBM Cloud® Load Balancer service currently has the following issues:

- The Edit button in the server instances and protocols tabs applies to all entries and is not restricted to rows selected by using a checkbox.
- During the initial creation of the load balancer service, your custom health check settings are lost if you go back and forth between various pages.
- You might experience some issues while using Internet Explorer 11, Edge, or Safari browsers for administering the load balancer service. As an alternative, use either the Firefox or Chrome browser.
- During the initial service creation, the list for the data centers might be skewed. Regardless, you can still select your data center.
- The pricing information on the review page is rounded off to two decimal digits. For correct pricing, refer to the pricing displayed on the plan page.

Known limitations

The IBM Cloud® Load Balancer service currently has the following limitations:

- Maximum number of virtual ports/protocols - 10
- Maximum number of servers behind load balancer - 50

FAQ for IBM Cloud Load Balancer

The following frequently asked questions relate to the IBM Cloud® Load Balancer service.

How many load-balancing options are available in IBM Cloud?

For a detailed comparison of IBM's load balancer offerings, see Exploring IBM Cloud load balancers.

Does IBM Cloud Load Balancer support HTTP/2 protocol?

Yes. The load balancer supports HTTP/2 protocol for both HTTP requests and responses.

Can I use a different DNS name for my load balancer?

While you can't customize the loads balancer auto-assigned DNS name, you can add a Canonical Name (CNAME) record that points your preferred DNS name to the auto-assigned DNS name.

For example, if your account number is `123456`, your load balancer is deployed in the `dal09` data center, and its name is `myapp`, the system assigns it the DNS name `myapp-123456-dal09.lb.bluemix.net`.

If you want to use `www.myapp.com` as your preferred name instead, you can create a CNAME record with your DNS provider to point `www.myapp.com` to the load balancer DNS name `myapp-123456-dal09.lb.bluemix.net`.

What's the maximum number of virtual ports that I can define with my load balancer service?

While you try to create a load balancer service, you can define up to two virtual ports. After the service is created, you can define extra virtual ports, up to a maximum of `10` ports.

What's the maximum number of compute instances that I can associate with my load balancer?

When you create a load balancer service, you can configure up to 10 compute instances as back-end servers. After the load balancer is created, you can add more servers, up to a maximum of `50` back-end members.

Can my back-end compute instances sit on a subnet different from the load balancer's subnet?

Yes. The load balancer and the compute instances that are connected to the load balancer can be in different subnets. However, you must enableVLAN spanning for the load balancer to communicate and forward requests to the compute instance. For more information, see VLAN spanning.

What are the default settings and allowed values for various health check options?

The default settings and allowed values are as follows.

- Health check interval - The default is 5 seconds, and the range is 2 – 60 seconds
- Health check response timeout - The default is 2 seconds, and the range is 1 – 59 seconds
- Max retry attempts - The default is two retry attempts, and the range is 1 - 10 retries

Note: The health check response timeout value must always be less than the health check interval value.

Can I use compute instances that reside in remote data centers with this service?

It's best to keep your load balancer service and compute instances in the same data center. The console shows only compute instances from other data centers within the same city (for example, data centers whose names share the first three letters, such as `DALxx`). If you need to add instances that reside in remote data centers, you can do that through the API.

Which TLS version does SSL offload support?

The IBM Cloud Load Balancer service supports TLS 1.2 with SSL termination.

The following list details the supported ciphers (listed in order of precedence):

- ECDHE-RSA-AES256-GCM-SHA384
- ECDHE-RSA-AES256-SHA384
- AES256-GCM-SHA384
- AES256-SHA256
- ECDHE-RSA-AES128-GCM-SHA256
- ECDHE-RSA-AES128-SHA256
- AES128-GCM-SHA256
- AES128-SHA256

What is the maximum number of load balancer service instances that I can create within my account?

Currently, you can create up to 50 service instances. If you need more instances, contact IBM Support.

Can the IBM Cloud Load Balancer service be used with VMWare?

VMWare virtual machines that are assigned IBM Cloud portable private addresses can be specified as back-end servers to the load balancer. This feature is available only from the API, but not from the console. Portable private IPs added by using the API appear as "Unknown" in the console as they are not assigned by IBM Cloud. This configuration can be used with other hypervisors, such as Xen and KVM.

VMWare virtual machines that are assigned non-IBM Cloud addresses, such as those that are on VMWare NSX networks can't be added directly as back-end servers to the load balancer. But depending on your setup, you might use an intermediary, such as an NSX gateway. This gateway has an IBM Cloud private address and acts as the back-end server for the load balancer, while the actual servers are VMs on VMware NSX-managed networks.

If I deploy a load balancer and a firewall on a public VLAN in my account, what configurations do I need on the firewall to make it work with the load balancer?

The TCP port 56501 is used for management. Ensure that incoming traffic to this port is not blocked by your firewall. Otherwise, load balancer provisioning, customer operations, and service triggered operations, might fail. More specifically, ports 56501 (management), 443 (monitoring), 8834 and 10514 (security and compliance) must be always allowed for the load balancer to successfully manage customer workloads. Some outbound traffic is also required to be open to make sure the load balancer functions properly.

In summary, the following are needed for the firewall configuration:

Inbound and Outbound	Protocol	Source IP	Source Port	Destination IP	Destination Port
Inbound	TCP	AnyIP	AnyPort	AnyIP	56501
Inbound	TCP	AnyIP	443	AnyIP	AnyPort
Inbound	TCP	AnyIP	10514	AnyIP	AnyPort
Inbound	TCP	AnyIP	8834	AnyIP	AnyPort
Outbound	TCP	AnyIP	56501	AnyIP	AnyPort
Outbound	TCP	AnyIP	AnyPort	AnyIP	443
Outbound	TCP	AnyIP	AnyPort	AnyIP	10514
Outbound	TCP	AnyIP	AnyPort	AnyIP	8834

Required firewall configuration

Also, ensure your application's ports are open to accept traffic.

What if I can't see the monitoring metrics of an existing load balancer after I link my Softlayer

account to IBM Cloud?

Monitoring metrics aren't available for existing load balancers after you link the accounts. Re-create the load balancers or contact IBM Support. Monitoring metrics are available for all newly created load balancers.

Are the load balancer IP addresses fixed?

No. The load balancer is built to adjust automatically as needed, which means its IP addresses might change over time. As the service scales up or down, the IPs linked to the FQDN of your instance might also change.

Note: Use FQDN and not cached IP addresses.

If I deploy a firewall in front of my backend servers, what configurations do I need to make it work with my load balancer?

The available range of possible IPs for `public to public` load balancers cannot be predicted. Because of this condition, you must open all back-end member ports that are added to the load balancer and set the source IP to `any`.

`Public to private` and `private to private` load balancers use your private subnets to communicate with your back-end members. Because of this setup, you can set the source IP with your subnet's CIDR.

If the data center where you created the load balancer is part of an MZR, one load balancer deploys in the selected data center, while a second deploys in a different data center within the same region. This means that they exist in two different subnets.

Can the IBM Cloud Load Balancer service be used with Terraform?

Yes. You can use Terraform to create, update, and delete an IBM Cloud Load Balancer service resource.

Is it possible to add members with secondary IP addresses?

Yes. You can add members with secondary IP addresses by using the API.

What happens when the load balancer is in `Maintenance Pending` state?

The load balancer can go into `Maintenance Pending` state due to the following reasons:

- A horizontal scaling event takes place.
- The load balancer has a firewall that's blocking the management traffic.
- Security patches are being installed.

When the load balancer is in the `Maintenance Pending` state, the data path is not affected.

What is a nonsystem pool?

A nonsystem pool is applicable only with public to private load balancers. The public IP addresses of the load balancer appliances are allocated from your public subnet. Select the Allocate from a public subnet in this accountoption when you provision a load balancer.

Does IBM Cloud Load Balancer support UDP?

No. The IBM Cloud Load Balancer service does not support UDP. It supports only TCP, HTTP, and HTTPS.

Is autoscaling supported by IBM Cloud Load Balancer?

No. Currently, the IBM Cloud Load Balancer service doesn't support autoscaling.

How can I monitor my IBM Cloud Load Balancer metrics?

The IBM Cloud Load Balancer service supports monitoring with IBM Cloud Monitoring. For more information, see Monitoring metrics that use IBM Cloud Monitoring.

What information do I need to file an IBM Support ticket?

To file an IBM Support ticket, provide the product name ("IBM Cloud® Load Balancer"), the UUID of your load balancer (if possible) and your IBM Cloud account number. You can find the UUID in the URL by going to the overview page of the specific load balancer.

Where can I find pricing information for IBM Cloud Load Balancer?

Pricing metrics for IBM Cloud Load Balancer are detailed in the pricing metrics page. You can estimate the cost of a service by using the cost estimator on the provisioning pages for IBM Cloud Load Balancer. Select IBM Cloud® Load Balancer from the Load Balancer page of the IBM Cloud catalog, then click Create.

Can I use my Classic Bandwidth Pool to cover bandwidth costs for my IBM Cloud Load Balancer?

No. Currently, IBM Cloud Load Balancer bandwidth usage isn't covered by Classic Bandwidth Pools.

Do I need extra IPs in the subnet for IBM Cloud load balancer operations?

Yes. To support scaling and maintenance, it is best to allocate 8 extra IP addresses per subnet used by the load balancer. If your application load balancer uses only one subnet, allocate 16 extra IPs.

Troubleshooting

Why is my back-end server's health failing?

What’s happening

Your back-end server's health is failing.

How to fix it

To rectify this problem, check the following possible issues:

- Does the port of the configured back-end protocol match the port that your application is listening on?
- Does the configured health-check have the correct protocol (TCP or HTTP), port, and URL (for HTTP) information? For HTTP, ensure that your application responds with `200 OK` for the configured health check URL.
- Is the back-end server on a different subnet than the load balancer? If not, ensure that VLAN spanning is enabled.
- Is the back-end server a virtual server with an enabled security group? If so, ensure that the security group rules allow traffic between the load balancer and the virtual server.

Common issues when creating a new load balancer

This topic provides information on common issues that you might encounter when creating a new instance of IBM Cloud® Load Balancer.

Insufficient IP addresses in your subnet

IBM Cloud® Load Balancer requires at least two free IP addresses from its private subnet. In addition, if the load balancer is a public load balancer and the IBM system pool option is not used, then at least two free IP addresses are needed from your public subnet as well. This IP address requirement is not only required when you create a new load balancer, but also when performing maintenance on an existing load balancer.

Issues with firewalls on public and private VLANs

For more information, see IBM Cloud IP ranges.

Viewing load balancer error messages

To view error messages for your load balancer, follow these steps:

1. Click the load balancer from the list page to view its details.
2. Mouseover the error symbol next to the load balancer's status.

If the load balancer is in an `ERROR` state, it cannot be recovered and must be deleted.

Error message troubleshooting

This topic provides information on common error messages that you might encounter while creating or updating an instance of IBM Cloud® Load Balancer.

Error	Explanation	Solution
`The maximum number of load balancers 'n' has been reached.`	Only 50 load balancer instances are allowed per account.	Ensure that you have 50 or less load balancer instances per account.
`The maximum number of given protocols 'n' in load balancer product order has been reached.`	Only two protocols can be added while provisioning a load balancer.	If you need more protocols, then after provisioning, you can add up to ten from the Protocols tab in the Load Balancer management flow. For more information, see monitoring and managing your service.
`The maximum number of given server instances 'n' in load balancer product order has been reached.`	Only ten servers can be added while provisioning a load balancer.	If you need extra servers, then after provisioning, you can add up to 50 from the Server Instances tab in the Load Balancer management flow. For more information, see monitoring and managing your service.

Load balancer name must be a string and have at least 1 and up to 40 characters.	The load balancer name is mandatory. Use only alphanumeric characters (or special characters '-' and '.') for the name. The name must not begin or end with a special character, and must be limited to 40 characters in length.	Enter a unique load balancer name. For example, "ui-servers" or "backend-servers".
Format error found in given load balancer name.	The load balancer name is mandatory. Use only alphanumeric characters (or special characters '-' and '.') for the name. The name must not begin or end with a special character, and must be limited to 40 characters in length.	Enter a unique load balancer name. For example, "ui-servers" or "backend-servers".
Load balancer with the same name (case insensitive) already exists.	A load balancer with the same name exists.	Enter a unique load balancer name to proceed further, for example, "ui-servers" or "backend-servers".
Load balancer description must be a string and not exceed 255 characters.	The load balancer description must be a string that does not exceed 255 characters.	Limit the description to 255 characters.
Frontend port 80 is already used.	You might have entered a front-end port that is already in use.	Enter a unique front end port.
Backend port must be an integer.	You might have entered an invalid back-end port value.	Enter a back-end port number between 1 and 65535.
Since the protocol and port are not editable in health monitor, this error is impossible from UI.	Your private subnet is not a standard type and cannot be used to create the load balancer.	Contact IBM Support.
Private subnet 'xyz' must have at least 'n' free IP addresses.	The selected private subnet does not have any free IP addresses.	Check these troubleshooting steps for more information.
Specified private subnet VLAN is on router 'xyz'. However, no public subnet with 'n' free IPs was found with VLAN on the same router.	This happens because you selected the Allocate from public subnet from this account option during provisioning.	Select Allocate from IBM System Pool instead, or contact IBM Support.
Given new description must be a string.	You might have entered an invalid description.	Enter a valid load balancer description, no larger than 255 characters.
A billing item is required to process a cancellation for load balancer uuid=aaaa-bbbb-cccc-dddd.	Billing information is missing or invalid for your account, and the load balancer cannot be cancelled.	Contact IBM Support.
An internal error occurred. Data could not be retrieved.	Metrics data cannot be retrieved	Reload the page. If the issue persists, then contact IBM Support.
Front-end port must be an integer between 1 and 65535 excluding the range 56500-56520.	You might have entered a front-end port number between 56500-56520.	Enter a unique port number between 1 to 65535, excluding the range of 56500 to 56520.
Back-end port must be an integer.	You might have entered an invalid back-end port value.	Enter a back-end port number between 1 and 65535.
Back-end port must be an integer between 1 and 65535 excluding the range 56500-56520.	You might have entered a back-end port between 56500 and 56520.	Enter a back-end port number between 1 and 65535, excluding the range of 56500 to 56520.

`Backend protocol HTTP is not compatible with frontend protocol TCP.`	You might have selected an incompatible back-end protocol and front-end protocol combination.	Select a valid front-end and back-end protocol combination, in the form: `<br> HTTP-HTTP <br> HTTPS-HTTP <br> TCP-TCP`
Member weight <some value> is provided for member abcd-xxxx-yyyy-2222. The allowed weight value is 0-100	You might have entered an invalid weight.	Enter a weight value between 0 and 100.
`There was a problem fetching the servers.`	This can happen as a result of network timeout issues.	Reload the page. If the issue persists, contact IBM Support.

Error messages

Getting help and support

If you experience an issue or have questions when using IBM Cloud® Load Balancer service, you can use the following resources before you open a support case.

- Ask a question in the AI assistant from the console or the IBM Cloud CLI.
- Review the FAQs in the product documentation.
- Review the troubleshooting documentation to troubleshoot and resolve common issues.
- Check the status of the IBM Cloud platform and resources by going to the Status page.
- Review Stack Overflow to see whether other users experienced the same problem. When you ask a question, tag the question with `ibm-cloud` and `load-balancer-service`, so that it's seen by the IBM Cloud development teams.

If you still can't resolve the problem, you can open a support case. For more information, see Creating support cases. And, if you're looking to provide feedback, see Submitting feedback.