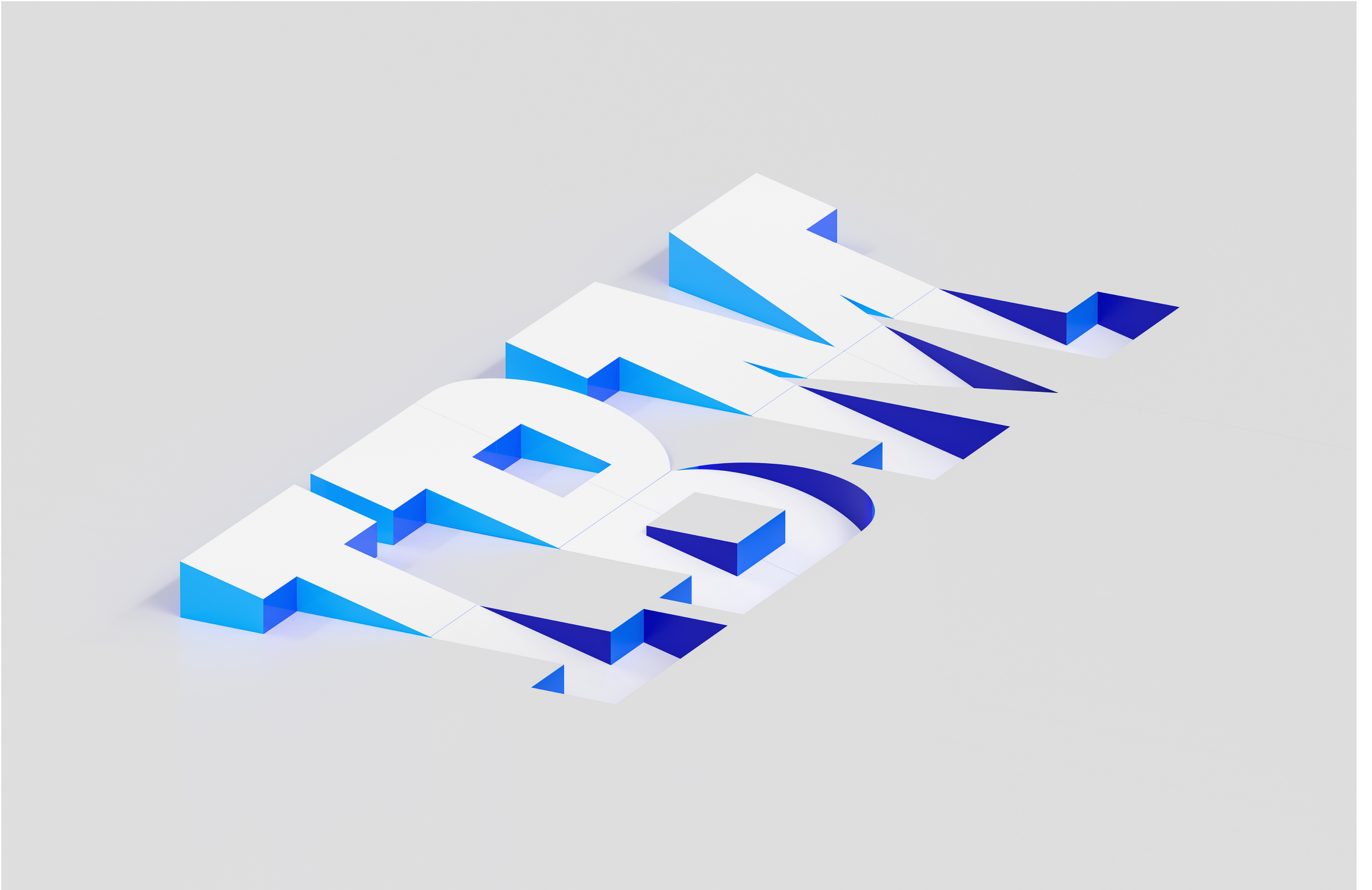


IBM Cloud

IBM watsonx Code Assistant

Product guide



Edition notices

This PDF was created on 2026-03-23 as a supplement to *IBM watsonx Code Assistant* in the IBM Cloud docs. It might not be a complete set of information or the latest version. For the latest information, see the IBM Cloud documentation at <https://cloud.ibm.com/docs/watsonx-code-assistant>.

Overview of watsonx Code Assistant

IBM watsonx Code Assistant is a portfolio of purpose-built, generative AI-assisted products that accelerate code and content generation and increase developer productivity, with trust, security, and compliance at its core. It provides models based on specific programming languages to provide trust and efficiency for accurate code and content generation.

IBM watsonx Code Assistant: Harness generative AI and advanced automation to create enterprise-ready code

watsonx Code Assistant

IBM watsonx Code Assistant is an innovative, generative AI coding companion that offers robust, contextually aware assistance for popular programming languages, including Go, C, C++, Java, JavaScript, Python, TypeScript, and more. Seamlessly integrated into your IDE, you can boost productivity and simplify coding tasks, all with trust, security, and compliance.

- **AI-powered coding tasks**: Code generation, explanation, unit test, creation, AI-derived code documentation, and more.
- **Chat for code**: Natural language AI conversational chat for planning applications and writing code.
- **Prebuilt chat commands**: Use optimized prompts for specific tasks.
- **Integrated directly into IDEs**: Available directly within Visual Studio Code and Eclipse, integrating seamlessly with your workflow.
- **Enterprise Java modernization**: Generative AI and automation-assisted modernization for Java enterprise applications (requires the Standard plan).
- **retrieval augmented generation**: Enhances response quality of user queries in relevant, up-to-date context from code bases and documentation (requires the Standard plan).

Overview video

Watch YouTuber **Tech With Lucy** demonstrate watsonx Code Assistant in the overview video [How to build technical projects faster using AI](#).

IBM Cloud

To get started:

1. [Set up your watsonx Code Assistant service in IBM Cloud](#)
2. [Install the IDE extension or plug-in](#)

IBM Cloud plan	Description
Trial plan	Limited no-cost trial with integrated generative AI chat for code
Essentials plan	Integrated generative AI for code suggestions, code explanations, code documentation, and unit tests
Standard plan	All the features of the Essentials plan, plus enterprise Java modernization capabilities, enhanced code explanation, and enhanced unit test generation

IBM Cloud plans

For complete plan details, see:

- [Pricing plans](#)
- [IBM Cloud catalog](#).

On-premises

You can also use IBM Software Hub to install watsonx Code Assistant as an on-premises service at your organization. For more information, see the [IBM Software Hub documentation](#).

IBM watsonx Code Assistant features

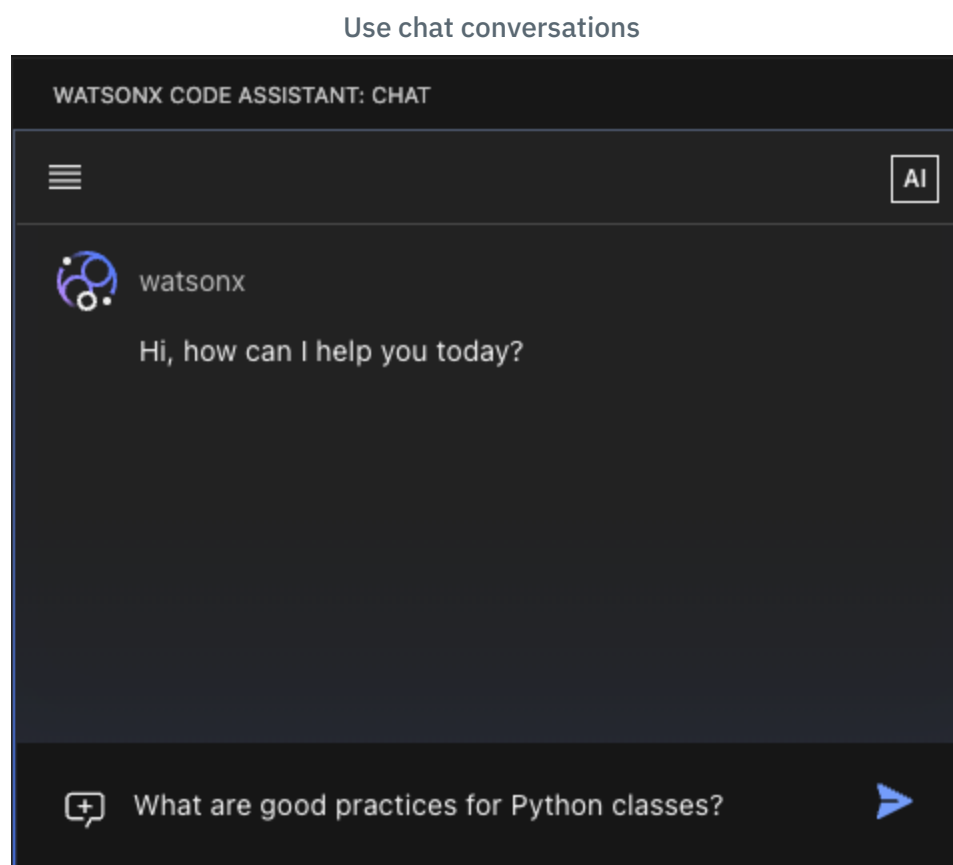
watsonx Code Assistant

You can use watsonx Code Assistant to:

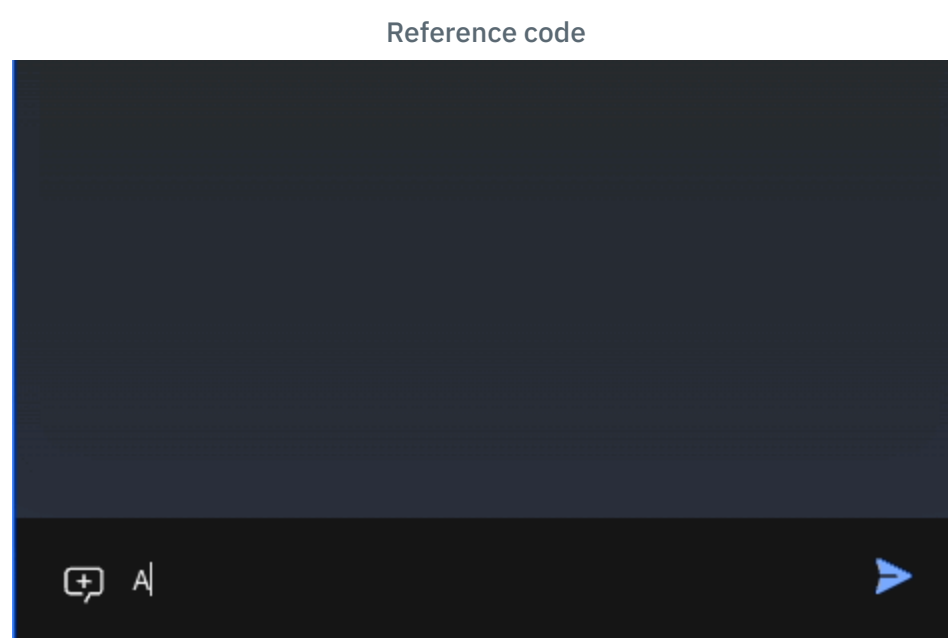
- [Get code suggestions](#)
- [Explain code](#)
- [Document code](#)
- [Generate unit tests](#)
- [Translate code from one language to another](#)
- [Get answers from IBM documentation](#)
- [Modernize or upgrade Enterprise Java applications](#)
- [Improve context by using retrieval augmented generation \(RAG\)](#)

Get code suggestions

Use chat conversations: Use natural language prompts to generate code suggestions. Use a chat conversation to enter a prompt that explains the code you need, and watsonx Code Assistant generates something you can choose to use.



Reference code: To ask questions or refine a specific file, class, function, or method in your workspace, you can use a code reference. These references provide important context and can help to increase the accuracy of the answer. As part of your chat message, type the @ symbol to see a list of files, classes, and methods from your workspace. Click to select the reference, and watsonx Code Assistant sends the contents of the reference as part of your message.



Code completion: Or, complete code in the editor. Start typing a line of code, then pause. IBM watsonx Code Assistant adds a code suggestion to complete the line that you typed.

Code completion

For more information, see the documentation for [Getting code suggestions](#).

Explain code

Use generative AI to analyze and summarize your code to understand what the code does. Click the **Explain** option that precedes a code block or enter `/explain` in a chat conversation. IBM watsonx Code Assistant analyzes the code and provides a detailed explanation of what the code does.

For more information, see [Explaining code](#).

Document code

Generate comment lines that document what your code does. Click the **Document** option that precedes a code block or enter `/document` in a chat conversation. IBM watsonx Code Assistant analyzes the code and adds comments that document what the code does.

For more information, see [Documenting code](#).

Generate unit tests

Create unit tests to evaluate your code functions. Click the **Unit Test** option that precedes a code block or enter `/unit-test` in a chat conversation. IBM watsonx Code Assistant analyzes the code and creates a unit test.

For more information, see [Generating unit tests](#).

Translate code from one language to another

Use watsonx Code Assistant to translate code. In a chat conversation, use the syntax `translate [from <source language >] to <target_language> <code reference>`

For more information, see [Translating code from one language to another](#).

Get answers from IBM documentation

Use chat to ask questions that reference IBM and Red Hat product documentation.

The chat syntax is: `/docs [collection:<collection_name>] <your question>`

The `collection_name` is optional. If no collection is specified, `ibm_docs` is used.

For more information, see [Getting answers from IBM documentation](#)

Modernize or upgrade Enterprise Java applications

Standard plan

For use with the Standard plan, IBM watsonx Code Assistant for Enterprise Java Applications adds features for Java application upgrades and modernization, and enhanced code explanation, unit test generation, and fixing Java unit tests.

Java runtime modernization

Analyze your Java application runtime and modernize it to a more lightweight, flexible, and efficient runtime. Receive a prescriptive plan that describes the changes that are needed to modernize your application, with a detailed assessment of complexity and required development effort. Use automation to quickly implement code and configuration changes. Transform code with generative AI assistance to resolve more complex issues.

For more information, see the documentation for [Modernizing Java applications](#).

Java version upgrade

Identify changes required to upgrade Java code, automatically apply fixes, and use generative AI to transform Java code. Receive a prescriptive plan that describes the changes that are needed to upgrade your application, with a detailed assessment of complexity and required development effort. Use automation to quickly implement code and configuration changes. Transform code with generative AI assistance to resolve more complex issues.

For more information, see the documentation for [Upgrading the Java version for your application](#).

Explain Java applications

With the Standard plan, you can generate an explanation of your Java application. For more information, see the documentation for [Explaining Java applications](#).

Generate and fix Java unit tests

Use generative AI to create tests that help maintain critical application functions. With the Standard plan, create enhanced unit tests to evaluate your code functions. You can also use generative AI to fix Java unit tests.

For more information, see the documentation for [Generating unit tests](#) and [Fixing unit tests](#).

Improve context by using retrieval augmented generation (RAG)

Standard plan

retrieval augmented generation (RAG) is the process of optimizing the large language model (LLM) output through the prompt augmentation with the additional context. If you submit a query, watsonx Code Assistant uses the RAG tools to retrieve the information from your code bases or documentation. This relevant context is appended to the query before it is sent to the LLM model. The RAG system determines the sources that need to be included or excluded to generate a response with the most useful information. For more information, see [Setting up retrieval augmented generation \(RAG\)](#).

Related products

- [IBM watsonx Code Assistant for Red Hat Ansible Lightspeed](#)
- [IBM watsonx Code Assistant for Z](#)

IBM watsonx Code Assistant for Red Hat Ansible Lightspeed: Writing Ansible playbooks with AI-generated recommendations

Red Hat Ansible Lightspeed

You can use watsonx Code Assistant for Red Hat Ansible Lightspeed to write Ansible playbooks with AI-generated recommendations. Purpose-built to accelerate IT Automation, the service delivers Ansible content recommendations that adhere to accepted enterprise standards.

Watsonx Code Assistant for Red Hat Ansible Lightspeed uses IBM foundation models to provide code recommendations directly in your Visual Studio Code development environment. You can create task prompts from natural language requests to generate Ansible code.

For more information about pricing plan options, see [IBM watsonx Code Assistant for Red Hat Ansible Lightspeed pricing plans](#).

For cloud administrator setup instructions, see [Setting up watsonx Code Assistant for Red Hat Ansible Lightspeed in IBM Cloud](#).

For more information about watsonx Code Assistant for Red Hat Ansible Lightspeed, including Visual Studio Code configuration instructions, see the [Red Hat Ansible Lightspeed with IBM watsonx Code Assistant User Guide](#).

IBM watsonx Code Assistant for Z: Refactoring and transforming COBOL code

watsonx Code Assistant for Z

Use the power of generative AI to help you modernize COBOL business services to object-oriented Java code.

Watsonx Code Assistant for Z provides an AI-driven interface to interpret, refactor, and transform your code. Use your existing IBM Z tools and the Z Open Editor Visual Studio Code extension to convert your COBOL code to Java.

Before you get started, make sure that you have alignment with your peers. To get started, you need your OS admins and DevOps specialists to install and configure the solution. Then, developers can use watsonx Code Assistant for Z to modernize your apps.

For cloud administrator setup instructions, see [Setting up watsonx Code Assistant for Z in IBM Cloud](#).

After you complete your initial setup, use the [watsonx Code Assistant for Z documentation](#) for full instructions about configuration and getting started as quickly as possible.

Setting up watsonx Code Assistant

Setting up your watsonx Code Assistant service in IBM Cloud

watsonx Code Assistant

Set up your service instance of watsonx Code Assistant.

Provision the service instance

To get started, you need to create your own service instance of watsonx Code Assistant for you or your organization.

To provision a service instance:

1. Open the [IBM Cloud catalog](#).
2. Use the catalog to provision a service instance of watsonx Code Assistant.

The service is available in these data centers:

- Dallas (`us-south`)
- Frankfurt (`eu-de`)
- Sydney (`au-syd`)

For more information about data centers, see the IBM Cloud documentation for [Service and infrastructure availability by location](#).

3. When you finish provisioning your instance, click **Setup** to open an onboarding checklist page to help you with configuration. Or, you can select **Administration**, and then **Setup** to access this onboarding checklist.
4. Items in the onboarding checklist correspond to each of the following steps. Click the action arrows on the checklist page that are associated with each item to complete the steps.

The setup steps are different depending on how you're going to use the service instance.

Type	Description
Organizational users	Multiple developers
Single user	One developer

Installation types

Create an access group

Organizational users

Create an access group where you can add your developers.

1. Open the **Access groups** page in [Cloud Identity and Access Management](#).
2. Click **Create**.
3. Enter a name for your access group, for example `wca-instance-access`, and then click **Create**.
4. In your new access group, click the **Access** tab.
5. Click **Assign access** to start a policy.
6. For **Service**, select **IBM watsonx Code Assistant**, then click **Next**.
7. For **Resources**, select **Specific resources**.
8. Select the resource group where you created your instance, then click **Next**.
9. For **Resource Group Access**, select **Viewer**, then click **Next**.
10. For **Roles and actions**, select **Viewer**.
11. **Conditions** aren't required, so click **Add**.

12. In the **Access summary**, click **Assign**.

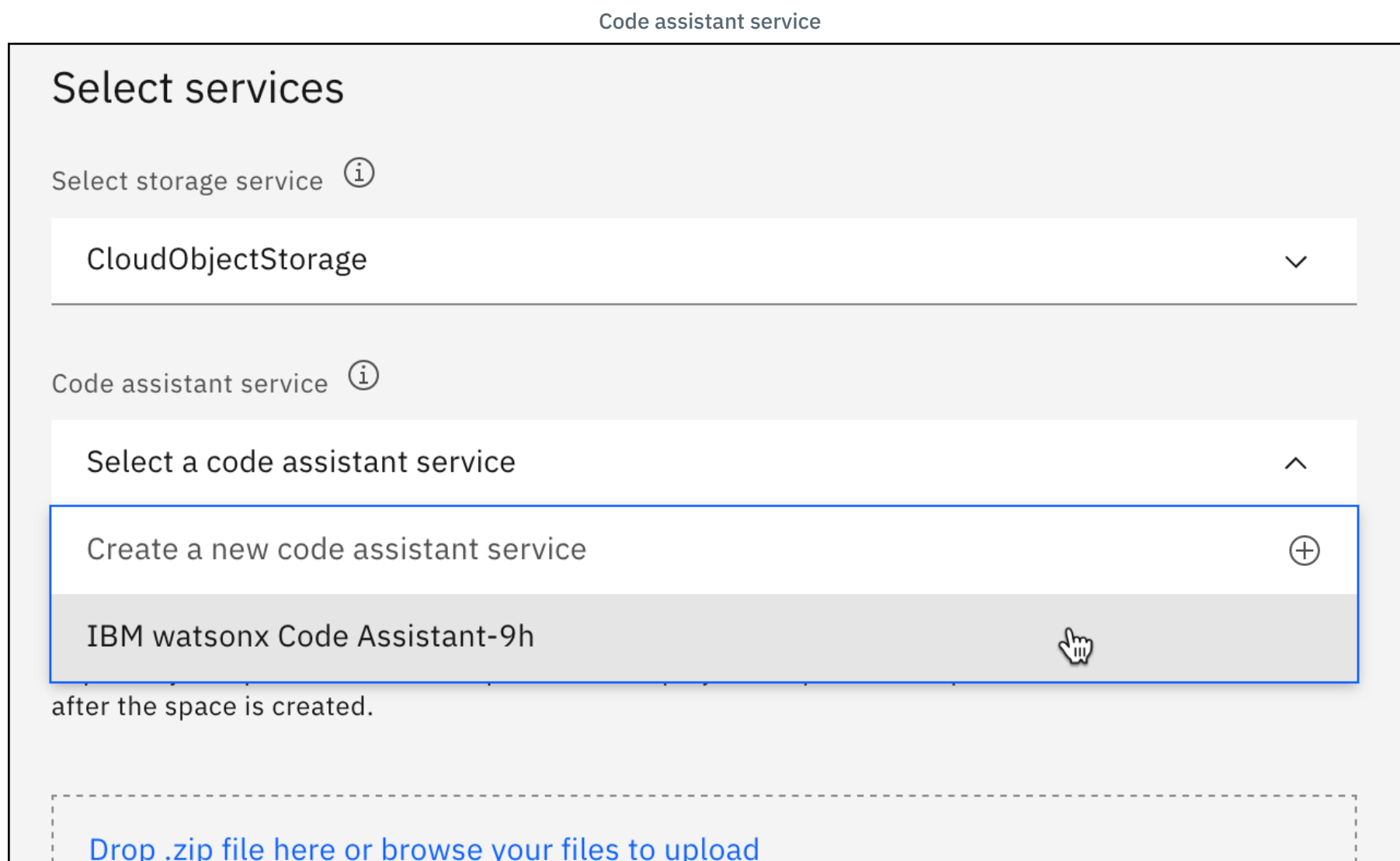
Create a deployment space

Single user Organizational users

The deployment space is the serving environment for watsonx Code Assistant.

1. Open your watsonx Code Assistant service instance.
2. Click the **Navigation Menu** ☰ icon, then select **Deployments**.
3. Click **New deployment space**.
4. Enter `Code Assistant` as the name for your deployment space. Optionally, enter a description.
5. In **Code assistant service**, select your instance name.

The following example displays the **Select services** page. For the **Code assistant service** option on the page, **Select a code assistant service** is expanded and the user is selecting the `IBM watsonx Code Assistant-9h` instance from the expanded list.



6. Click **Create**.
7. A status window appears. After the space is created, click **Close**.

Add the access group to the deployment space

Organizational users

Add your new access group to the deployment space.

1. In your watsonx Code Assistant instance, click the **Navigation Menu** ☰ icon, then select **Deployments**.
2. Open the deployment space for your service instance.
3. On the **Manage** tab of your deployment space, click **Access control**.
4. Click **Add collaborators** and select **Add access groups** from the menu.
5. Select your new access group.
6. For **Role**, select **Editor**, then click **Add**.

Select your license preferences for code suggestions

Single user

Organizational users

Watsonx Code Assistant checks whether a generated code suggestion is similar to other code sources. By default, the code suggestion includes a reference to the similar code source and its license.

You can set how code suggestions are displayed in the IDE. For more information, see [Configuring how code suggestions are displayed](#).

Add developers to your access group

Organizational users

Use Cloud Identity and Access Management to add developers to your access group.

1. Open the **Access groups** page in [Cloud Identity and Access Management](#).
2. Select the access group that you created.
3. On the **Users** tab, click **Add**.
4. Select the members of your IBM Cloud account who should get access to watsonx Code Assistant.
5. Click **Add**.

Set up your network

To ensure that your developers can connect the IDE extension or plug-in, you need to allow access to IBM Cloud URLs. Make sure your organization's network allows access to:

- `https://iam.cloud.ibm.com/`
- `https://api.dataplatform.cloud.ibm.com/`



Note: If your organization uses an HTTP proxy server, your developers need to update their settings to connect correctly. For more information, see [Update HTTP proxy settings for Visual Studio Code](#) or [Update HTTP proxy settings for Eclipse IDE](#).

Instruct developers to add an IDE extension or plug-in

Organizational users

Developers need to add the watsonx Code Assistant Microsoft Visual Studio Code extension or the Eclipse IDE plug-in.

For the instructions, see [Installing the IDE extension or plug-in](#).

Copy this link to send the instructions to your developers:

`https://cloud.ibm.com/docs/watsonx-code-assistant?topic=watsonx-code-assistant-cloud-setup-wca-extensions`

Configuring how code suggestions are displayed

watsonx Code Assistant

Watsonx Code Assistant checks whether a generated code suggestion is similar to other code sources.

The generated code must be 10 or more lines in these languages:


- C
- C++
- Go
- Java
- JavaScript
- Python
- TypeScript

By default, watsonx Code Assistant blocks code suggestions that are similar to other code sources. You can set whether code suggestions that are similar to code from specific licenses are displayed in the IDE with a reference to the similar code source and its license.

The code suggestion display is set per license. The following table lists the code similarity settings and a description of each:

Setting	Description
Block code suggestion (default)	No code suggestion
Show code suggestion with reference	Show the code suggestion with a license reference

Code similarity settings

 **Important:** IBM recommends that you block the display of code suggestions that are similar to other code sources. If you choose to show code suggestions that are similar to other code sources with a reference, your eligibility for contractual IP protection (if available) under your customer agreement might be affected.

To change settings:

1. Open your watsonx Code Assistant instance.
2. On the home page, click the **Settings for code generation licenses** tile.
3. Select the licenses that you want to change.
4. Click the setting in the action bar. The action bar choices are **Block** or **Show with reference**.

For a list of licenses that can be configured, see the [Code suggestion license reference](#).

Installing the IDE extension or plug-in

watsonx Code Assistant

Add the watsonx Code Assistant extension or plug-in to your editor.

After the administrator completes the [setup of the service instance](#) of watsonx Code Assistant, developers can:

- Create an IBM Cloud API key
- Add the watsonx Code Assistant extension for Microsoft Visual Studio Code or plug-in for the Eclipse IDE

Open the instructions for the editor that you're using.

IDE	Create API key	Install
Microsoft Visual Studio Code	Create an IBM Cloud API key	Installing the extension for Visual Studio Code
Eclipse IDE	Create an IBM Cloud API key	Installing the plug-in for the Eclipse IDE

Instructions per IDE

Installing the IDE extension or plug-in

watsonx Code Assistant

Add the watsonx Code Assistant extension or plug-in to your editor.

After the administrator completes the [setup of the service instance](#) of watsonx Code Assistant, developers can:

- Create an IBM Cloud API key
- Add the watsonx Code Assistant extension for Microsoft Visual Studio Code or plug-in for the Eclipse IDE

Open the instructions for the editor that you're using.

IDE	Create API key	Install
Microsoft Visual Studio Code	Create an IBM Cloud API key	Installing the extension for Visual Studio Code
Eclipse IDE	Create an IBM Cloud API key	Installing the plug-in for the Eclipse IDE

Instructions per IDE

Installing the extension for Visual Studio Code

watsonx Code Assistant

Create an API key and add the watsonx Code Assistant extension to your Microsoft Visual Studio Code editor.

Compatibility

The extension is compatible with Visual Studio Code 1.101 and later, using Node.js version 22.15 or later.

To verify your Node.js version in Visual Studio Code, click the **Code** menu, then choose **About Visual Studio Code**. Make sure Node.js is version 22.15 or later.

Installing the extension in a development container is not supported.

Operating systems:

- macOS
- Windows 11
- Fedora Linux 41
- Red Hat Enterprise Linux 8
- Ubuntu 20.04

Create an API key on IBM Cloud

Instead of a username and password, each developer uses a personal API key to log in.

To create an API key:

1. Log in to [IBM Cloud API Keys](#).



Note: Ensure that you are using the IBM Cloud account where your organization's instance of watsonx Code Assistant is provisioned.

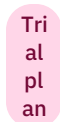
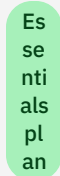
2. Click **Create**.
3. Enter `IBM watsonx Code Assistant API key` as the name for your API key. Optionally, enter a description.
4. Click **Create**.
5. Click **Download** or **Copy** and store the API key in a safe location.



Important: The key is only available for 296 seconds. Store it where you can find it later.

Install the extension

Choose the installation for the plan that you are using.

Plan	Description	Extension name	Display language support
	Limited no-cost trial with integrated generative AI chat for code.	watsonx Code Assistant	English (en), Japanese (ja), Korean (ko)
	Integrated generative AI for code suggestions, code explanations, code documentation, and unit tests.	watsonx Code Assistant	English (en), Japanese (ja), Korean (ko)

Standard plan

All the features of the Essentials plan, plus enterprise Java modernization capabilities, enhanced code explanation, enhanced unit test generation, and retrieval augmented generation (RAG).

For the Standard plan installation, two entries appear in your Visual Studio Code Extensions panel, one for **watsonx Code Assistant** and one for **watsonx Code Assistant for Enterprise Java Applications**. The two extensions are dependencies for this installation. Version 1.6.2 or later of watsonx Code Assistant and version 1.1.0 or later of watsonx Code Assistant for Enterprise Java Applications are required.

[watsonx Code Assistant for Enterprise Java Applications](#) English (en) only

Plan descriptions

For complete plan details, see:

- [Pricing plans](#)
- [IBM Cloud catalog](#)

From the Visual Studio Marketplace

To install from the Visual Studio Marketplace:

1. Choose the installation for the plan that you are using.
 - **Trial plan** **Essentials plan** To install for Trial and Essentials plans, open the [watsonx Code Assistant](#) page in the Visual Studio Marketplace.
 - **Standard plan** To install for the Standard plan, open the [watsonx Code Assistant for Enterprise Java Applications](#) page in the Visual Studio Marketplace.
2. Click **Install** on the Marketplace page.
3. In Visual Studio Code, click **Install** on the extension.


From the Visual Studio Code editor

To install from your Visual Studio Code editor:

1. In your Visual Studio Code editor, click the **Extensions** icon.
2. Choose the installation for the plan that you are using:
 - **Trial plan** **Essentials plan** To install for Trial and Essentials plans, search for `watsonx Code Assistant`.
 - **Standard plan** To install for the Standard plan, search for `watsonx Code Assistant for Enterprise Java Applications`.
3. Select the extension that you need, then click **Install**.

Open the extension

To open watsonx Code Assistant:

1. Click the **watsonx Code Assistant**  icon.
2. Click **Log in with your API key**, then enter your IBM watsonx Code Assistant API key.



Note: If your account is associated with more than one IBM Cloud data center, also choose the location.

Set up your enterprise Java environment

Standard plan If you're using the Standard plan and want to work with enterprise Java applications, you need to set up your environment. For more information, see [Setting up your enterprise Java environment](#).

Update HTTP proxy settings

If your organization uses an HTTP proxy server, you need to update your Visual Studio Code settings so it can connect correctly. You might receive console errors when you try to connect and this setting is missing.

1. Open the [http.proxy.setting](#) in Visual Studio Code settings.
2. Enter the URL of your HTTP proxy server in the format `https://proxy_server:port`. You might need to include authentication information, such as `http://user:pass@proxy_server:port`.

Setting up verification for custom certificates

If your proxy server uses custom certificates, you need to configure verification for Visual Studio Code. Follow these steps:

To export your server SSL certificate:

1. Use Google Chrome to open your watsonx Code Assistant instance on IBM Cloud.
2. Click the **View site information** icon next to the web address in the browser.
3. Click **Connection is secure**, then click **Certificate is valid** to open the Certificate Viewer.
4. Click **Details**, select the certificate of the proxy, and then click **Export**.

To set up for MacOS:

1. Use Spotlight to search for and open Keychain Access.
2. Click **System**, then click the **Certificates** tab.
3. Drag the certificate file into the Certificates tab.
4. Right-click the certificate, then click **Get info**.
5. In the Trust section, set **When using this certificate** to **Always Trust**.
6. Restart the extension to apply the changes. You should be able to log in by using your API key.

To set up for Windows:

1. Open the Microsoft Management Console.
2. Open **Certificates (Local Computer) > Trusted Root Certification Authorities > Certificates**.
3. Right-click **Certificates**, click **All Tasks**, then choose **Import**.
4. Click **Details**, select the certificate of the proxy, and then click **Export**.
5. Restart Visual Studio Code and the watsonx Code Assistant extension to apply the changes. You should be able to log in by using your API key.

Secure your setup

IBM watsonx Code Assistant doesn't provide additional security controls. Use these steps to properly secure your setup.

Your Visual Studio Code environment

Apply all Visual Studio Code updates to help ensure that you have the latest security and bug fixes. For more information, see the [Microsoft Documentation](#).

The watsonx Code Assistant extension logs are stored in *.log files under `<your home directory>/wca`. These files are not encrypted, other than the encryption that your file system provides. Safeguard the logs against improper access.

Chat conversation storage

IBM watsonx Code Assistant stores all your chat conversation history locally in your file system in `<your home directory>/wca/chat.db`, in a database format defined by [SQLite](#). IBM watsonx Code Assistant does *not* share these conversations with anyone. This file is not encrypted, other than the encryption that your file system provides. Safeguard this file against improper access.

Telemetry data

IBM watsonx Code Assistant does *not* collect any telemetry data. In general, watsonx Code Assistant doesn't send any data that it processes to a third party, IBM included.

Logging out of your API key

If necessary, you can log out of your API key. Logging out and reentering your API key might help with any authentication issues you encounter.

To log out:

1. In Visual Studio Code, click the **Accounts** icon.
2. Click the **Your Name (WCA API Key)** setting, then click **Sign out**.

Using the Output view to see activity details

You can use the Output view of Visual Studio Code to see details about what the extension is doing and logging, and check for any errors.

To open the Output view:

1. In Visual Studio Code, click **View**, then click **Output**.
2. In the Output view, use the drop-down list to select **WCA** or **WCA Language Server**. Both provide more details about what the extension is doing.

To adjust the Output view and logging for the Visual Studio Code extension:

1. In Visual Studio Code, open the extension settings for watsonx Code Assistant.
2. In **Wca: Log level**, switch from the default of **INFO** to another setting such as **WARN**.

Disabling or uninstalling the extension


If necessary, you can disable the extension and keep it installed, or you can uninstall it completely.

To disable or uninstall the extension:

1. In Visual Studio Code, click the **Extensions** icon to open the panel.
2. Click the **watsonx Code Assistant** extension.
3. Click **Disable** or **Uninstall**.

Changing the display language

The watsonx Code Assistant extension is available in English (**en**), Japanese (**ja**), and Korean (**ko**).

 **Note:** The watsonx Code Assistant for Enterprise Java Applications extension is available only in English (**en**).

To change the display language:

1. In Visual Studio Code, click **View**, then click **Command Palette**.
2. Search for **display**, then click **Configure Display Language**.
3. Choose either English (**en**), Japanese (**ja**), or Korean (**ko**).

For more information, see [Display Language](#) in the Visual Studio Code documentation.

Installing the plug-in for the Eclipse IDE

watsonx Code Assistant

Create an API key and add the watsonx Code Assistant plug-in to your Eclipse IDE.

Compatibility

The plug-in is compatible with:

- Eclipse IDE for Java Developers or Eclipse IDE for Enterprise Java and Web Developers
- Versions 2024-06, 2024-09, 2024-12, 2025-03, 2025-06, 2025-09
- On Windows, the plug-in requires the Microsoft Edge WebView2 runtime. In newer versions of Windows this might already be installed.

Operating systems:

- macOS
- Windows 11
- Fedora Linux 41
- Red Hat Enterprise Linux 8
- Ubuntu 20.04

Create an API key on IBM Cloud

Instead of a username and password, each developer uses a personal API key to log in.

To create an API key:

1. Log in to [IBM Cloud API Keys](#).



Note: Ensure that you are using the IBM Cloud account where your organization's instance of watsonx Code Assistant is provisioned.

2. Click **Create**.
3. Enter `IBM watsonx Code Assistant API key` as the name for your API key. Optionally, enter a description.
4. Click **Create**.
5. Click **Download** or **Copy** and store the API key in a safe location.



Important: The key is only available for 296 seconds. Make sure to store it somewhere where you can find it later.

Install the plug-in

The plug-in that you need depends upon the plan you are using.

Plan	Description	Plug-in name	Display language support
Trial plan	Limited free trial with integrated generative AI chat for code.	watsonx Code Assistant	English (en), Japanese (ja), Korean (ko)
Essentials plan	Integrated generative AI for code suggestions, code explanations, code documentation, and unit tests.	watsonx Code Assistant	English (en), Japanese (ja), Korean (ko)
Standard plan	All the features of the Essentials plan, plus enterprise Java modernization capabilities, enhanced code explanation, and enhanced unit test generation. For the Standard plan installation, two entries appear in your list of installed software, one for watsonx Code Assistant and one for watsonx Code Assistant for Enterprise Java Applications . The two plug-ins are dependencies for this installation. Version 1.3.1 or later of watsonx Code Assistant and version 1.1.0 or later of watsonx Code Assistant for Enterprise Java Applications are required.	IBM watsonx Code Assistant for Enterprise Java Applications	English (en) only

Plug-ins by plan

For complete plan details, see:

- [Pricing plans](#)
- [IBM Cloud catalog](#).

From the Eclipse Marketplace

To install the plug-in from the Eclipse Marketplace:

1. Open your Eclipse IDE.
2. Choose the installation for the plan that you are using:
 - **Trial plan** **Essentials plan** To install the plug-in for Trial and Essentials plans, open the [watsonx Code Assistant](#) page in the Eclipse Marketplace.
 - **Standard plan** To install the plug-in for Standard plans, open the [IBM watsonx Code Assistant for Enterprise Java Applications](#) page in the Eclipse Marketplace.
3. Click and drag the **Install** button into the workspace in your Eclipse IDE.

4. On Confirm Features, review the items, then click **Confirm**.
5. On Review Licenses, read the license, click to agree, then click **Finish**.
6. On Trust Authorities, click the checkbox, then click **Trust Selected**.
7. On Trust Artifacts, click the **Type** checkbox, then click **Trust Selected**.
8. You need to restart Eclipse. On Software Updates, click **Restart Now**.

From the Eclipse IDE

To install the plug-in from your Eclipse IDE:

1. In your Eclipse IDE, click the **Help** menu, then choose **Eclipse Marketplace**.
2. Choose the installation for the plan that you are using:
 - **Trial plan** **Essentials plan** To install the plug-in for Trial and Essentials plans, search for `watsonx Code Assistant`.
 - **Standard plan** To install the plug-in for Standard plans, search for `watsonx Code Assistant for Enterprise Java Applications`.
3. Click the **Install** button for the plug-in you need.
4. On Confirm Features, review the items, then click **Confirm**.
5. On Review Licenses, read the license, click to agree, then click **Finish**.
6. On Trust Authorities, click the checkbox, then click **Trust Selected**.
7. On Trust Artifacts, click the **Type** checkbox, then click **Trust Selected**.
8. You need to restart Eclipse. On Software Updates, click **Restart Now**.

Open the plug-in

To open watsonx Code Assistant:

1. After Eclipse restarts, click **Window**, select **Show View**, then click **Other**.
2. In the Show View dialog, open the **watsonx Code Assistant** folder, select **Chat**, then click **Open**.
3. Click **Log in with your API key**.
4. In the Security Preferences, enter your IBM watsonx Code Assistant API key.
5. If your account is associated with more than one IBM Cloud data center, also select the preferred region from the drop-down list.
6. Click **Apply and close** to exit the Security Preferences.

Set up your enterprise Java environment

Standard plan If you're using the Standard plan and want to work with enterprise Java applications, you need to set up your environment. For more information, see [Setting up your enterprise Java environment](#).

Update HTTP proxy settings

If your organization uses an HTTP proxy server, you need to update your Eclipse IDE settings so it can connect correctly. You might receive console errors when you try to connect and this setting is missing.

1. Open the Eclipse IDE settings.
2. In the General section, click **Network Connections**.
3. Set **Active Provider** to `Manual`.
4. Select the schema, such as `HTTPS`, then click **Edit**.
5. Enter the host URL and port of your HTTP proxy server.
6. If you need to include authentication information, click **Requires Authentication**, then enter your username and password.
7. Click **OK** to apply the changes.

Secure your setup

IBM watsonx Code Assistant doesn't provide any additional security controls. Use these steps to properly secure your setup.

Your Eclipse environment

Apply all Eclipse IDE updates to help ensure you have the latest security and bug fixes.

The watsonx Code Assistant plug-in logs are stored in *.log files under `<your home directory>/wca`. These files are not encrypted, other than the encryption that your file system provides. Safeguard the logs against improper access.

Chat conversation storage

IBM watsonx Code Assistant stores all your chat conversation history locally in your file system in `<your home directory>/wca/chat.db`, in a database format defined by [SQLite](#). IBM watsonx Code Assistant does *not* share these conversations with anyone. This file is not encrypted, other than the encryption that your file system provides. Safeguard this file against improper access.

Telemetry data

IBM watsonx Code Assistant does *not* collect any telemetry data. In general, watsonx Code Assistant doesn't send any data that it processes to a third party, IBM included.

Log out of your API key

If necessary, you can log out of your API key. Logging out and reentering your API key might help with any authentication issues you encounter.

To log out:

1. Open the chat.
2. Click the **Logout** icon.

Use the Console to see activity details

You can use the Console view of the Eclipse IDE to see details about what the plug-in is doing and logging, and check for any errors.

To open the Console view:

1. In Eclipse, click **Window**, select **Show View**, then click **Console**.
2. In the Console view, click the **Display Selected Console** icon to select **watsonx Code Assistant** or **watsonx Code Assistant: LSP**. Both provide more details about what the plug-in is doing.

To adjust the Console view and logging for the Eclipse plug-in:

1. Open the settings for the Eclipse IDE.
2. Click the **watsonx Code Assistant Settings** entry.
3. In **Log level**, switch from the default of `INFO` to another setting such as `WARN`.

Uninstall the plug-in

If you need to uninstall watsonx Code Assistant, follow these steps:

1. In your Eclipse editor, click **Help**, then click **Install New Software**.
2. Click **What is already installed?**.
3. On the **Installed Software** tab, select **watsonx Code Assistant**, then click **Uninstall**.
4. On Uninstall, review the items, then click **Finish**.
5. You need to restart Eclipse. On Software Updates, click **Restart Now**.

Setting up your enterprise Java environment

watsonx Code Assistant

Standard plan

If you're using the Standard plan and want to work with enterprise Java applications, you need to set up your environment for your IDE.

Software requirements

The requirements to use watsonx Code Assistant for Enterprise Java Applications in Visual Studio Code or in Eclipse are:

Item	Details
Java	Java SE 11 with Java Developer Kit 11 or a later version
Maven	Use Maven to build your application. If you are using Liberty Tools and the Liberty Maven plug-in to build your app or project, use Maven 3.8.6 or a later version.

System requirements

Visual Studio Code setup

- [Set environment variables](#)
- [Determining the Java developer kit](#)
- [Set logging level](#)

Set environment variables

Set up your environment variables.

JAVA_HOME

Set the `JAVA_HOME` environment variable the same way that you do for Maven. Set it to the JDK directory that contains the `/bin` directory so your Java executable file is in the `$JAVA_HOME/bin/java` path.

You can set the `JAVA_HOME` environment variable to build your application at a different version, for example, Java 11.

PATH

Set the `PATH` environment variable to include the `mvn` executable file.

- Windows: `mvn.cmd`
- macOS: `mvn`



Note: If you use the Visual Studio Code terminal to set the environment variables, you need to restart Visual Studio Code to apply the changes.

Determining the Java developer kit

For Visual Studio Code, if you are using the [Extension Pack for Java](#), watsonx Code Assistant determines the Java developer kit to use in this order of precedence:

Order of precedence	Description
1	Globally-configured Java developer kit for automatic building in the Red Hat extension, which is <code>java.configuration.runtimes</code> in the <code>IDE settings.json</code> file.
2	<code>java.jdt.ls.java.home</code> or <code>java.home</code> IDE settings if they are present.
3	<code>JDK_HOME</code> system environment variable.
4	<code>JAVA_HOME</code> system environment variable. Required if items 1 through 3 aren't set.

Java developer kit order

When watsonx Code Assistant for Enterprise Java Applications features are used, such as Java modernization or upgrade, you can verify which Java developer kit is being used:

1. In Visual Studio Code, click **View**, then click **Output**.
2. In the Output view, use the drop-down list to select **WCA**.
3. Check for a message such as `Using the Java developer kit that is defined in <location> to run watsonx Code Assistant components. The path is: <path>`.

Set logging level

To adjust logging for the Visual Studio Code extension:

1. In Visual Studio Code, open the extension settings for watsonx Code Assistant.
2. In **Wca: Log level**, switch from the default of `INFO` to another setting such as `WARN`.

Eclipse IDE setup

- [Determining the Java installation](#)
- [Maven setup](#)

Determining the Java installation

For Eclipse, the Java installation is determined in this order:

- The installation that is configured in the project's build path
- The default Java installation for the IDE

For application modernization and upgrade, the automated fixes use the Java developer kit installation that best matches the target Java version that you chose. You need to:

1. Install the Java developer kit.
2. In the Eclipse IDE settings, open the **Java** section and add the installation to the **Installed JREs**.

Maven setup in Eclipse

Make sure to use Maven to build your application.

Maven is required to be set on the PATH.

Set the `PATH` environment variable to include the `mvn` executable file.

Operating system	Executable file
Windows	<code>mvn.cmd</code>
macOS	<code>mvn</code>

Maven executable files

Note: For macOS operating systems, the environment variables might not be set as expected when you run Eclipse. To fix the problem, you can restart Eclipse by using the Finder tool to right-click your Eclipse application and choose Show Package Contents. Open the `Contents` folder, select `MacOS`, and then run Eclipse by clicking the executable code.

How is the pom.xml file of a Maven project located?

Watsonx Code Assistant uses the aggregator `pom.xml` file to build and manage the entire multimodule Maven project. When watsonx Code Assistant attempts to do builds and other Maven-related activity, it uses the multi-module root (MMR) to locate the aggregator `pom.xml` file.

- The MMR first searches the highest level project directory for the aggregator `pom.xml` file.
- If the MMR doesn't find the file, it searches the project directory by going from the highest to the next highest directory structure, and so on, until it finds an aggregator `pom.xml` file.
- If the MMR again does not find an aggregator `pom.xml` file, it searches the project directory by going from the highest to the next highest directory structure, and so on, until it finds a regular `pom.xml` file. A regular `pom.xml` file indicates that the Maven project is a single module project instead of a multi-module project.

However, there is an extra restriction. The multimodule project must have one `WAR` file or one `EAR` file. There can be multiple WAR files if there is a single EAR file. But, if there are multiple WAR files and no EAR file, then the project is not supported. The reason is so that watsonx Code Assistant can analyze the correct binary and place the Liberty configuration file (`server.xml`) in the correct submodule.

Configuring retrieval augmented generation (RAG)

Setting up retrieval augmented generation (RAG)

RAG is the process of optimizing the large language model (LLM) output through the prompt augmentation with the additional context. If you submit a query, watsonx Code Assistant uses the RAG tools to retrieve the information from your code bases or documentation. This relevant context is appended to the query before it is sent to the LLM model. The RAG system determines the sources that need to be included or excluded to generate a response with the most useful information.



Important: RAG requires the watsonx Code Assistant Standard plan and the Visual Studio Code extension. It isn't available for use with the Eclipse plug-in.

IBM watsonx Code Assistant supports RAG that enhances response quality of user queries in relevant, up-to-date context from code bases and documentation. RAG reduces model hallucinations and improves the accuracy of generated responses.

You can configure watsonx Code Assistant with specific code repositories and project documentation that are not stored in the Git repository by using RAG to extract relevant information for the chat message. You can configure documentation such as API documents, readme files, technical and design documents, Markdown, PDFs, Word, and PowerPoint documents.

The following figure illustrates the procedure to configure RAG for watsonx Code Assistant:

Procedure for setting up RAG in watsonx Code Assistant



Enabling RAG

To enable RAG:

1. Provision a vector store on IBM Cloud.
 - a. Provision a Milvus or Elasticsearch vector store instance on the IBM Cloud. If the vector store is already available on your IBM Cloud, you can skip this step.
 - To provision an Elasticsearch vector store instance, see [Provisioning Databases for Elasticsearch](#).
 - To provision a Milvus vector store instance, see [Provisioning watsonx.data](#).



Note: Milvus is a part of IBM watsonx.data. You can provision a Milvus instance with the [watsonx.data enterprise plan](#).

- b. Optional: Provision an instance of IBM watsonx.ai Studio to index the code repository.
2. Index or refresh your code repositories in Milvus or Elasticsearch. For more information, see [Indexing code repositories with IBM watsonx.ai Studio](#).

Set up the connection assets:

- a. Use the existing deployment space in your watsonx Code Assistant service instance.
 - b. Create a connection asset for each index that is created in the vector store. For more information, see [Creating a connection asset](#).
3. Set up the Git personal access token in Visual Studio Code and use the RAG-enabled prompts.
 - a. Click your profile icon in GitHub and go to **Settings > Developer Settings > Personal Access Tokens > Tokens (Classic)**.
 - b. Copy your personal access token.
 - c. In Visual Studio Code, click **View**, then click **Command Palette**.
 - d. Search for `WCA`, then click **Enter GitHub Personal Access Token for WCA**.
 - e. Enter your GitHub personal access token and press `Enter` or `Return`.
 - f. In chat messages, enter `@repo < instruction >` or `@docs < instruction >` to generate a response that uses the context from the referenced repository or documents in the vector store. Replace the `< instruction >` parameter with a prompt message.
 - Sample syntax to use the referenced repository:

```
@repo how is a chat message processed?
```

watsonx Code Assistant uses the indexed repositories based on the following conditions:

- If one repository is opened in Visual Studio Code, watsonx Code Assistant searches for the context in the opened repository as default.
- If multiple repositories are opened in Visual Studio Code, watsonx Code Assistant searches for the context from the repository that is associated with the recently accessed file.

- Watsonx Code Assistant checks for the `repo.yaml` file in the indexed repository when you enter `@repo < instruction >` syntax in the chat. If one or more YAML configuration files are configured, watsonx Code Assistant uses all the configured repositories to generate a response. If YAML configuration is not configured, watsonx Code Assistant uses the selected repository.
- Sample syntax to use the referenced document collection:

```
@docs What are the steps to setup a connection to the user data store?
```

Watsonx Code Assistant uses the indexed document collections based on the following conditions:

- If the document is opened in Visual Studio Code, watsonx Code Assistant searches for the context in the opened repository as default.
 - If multiple document collections are opened in Visual Studio Code, watsonx Code Assistant searches for the context from the recently accessed document.
 - Watsonx Code Assistant checks for the `docs.yaml` file in the indexed repository when you enter `@docs < instruction >` syntax in the chat. If one or more YAML configuration files are configured, watsonx Code Assistant uses all the configured document collections to generate a response. If YAML configuration is not configured, watsonx Code Assistant uses all the document with the `docs_<filename>` name in your deployment space.

Optional: If you need to set up YAML configuration for indexed repositories or document collections, see [Setting up YAML configuration for RAG](#).

Setting up YAML configuration for RAG

You can set up YAML configuration optionally to allow watsonx Code Assistant to search for multiple repositories at the same time or use the specific indexed code repository or document in the vector store. If you do not set up YAML configuration, watsonx Code Assistant uses the repository that is opened in Visual Studio Code as default.



Note: watsonx Code Assistant uses the API key authorization method to ensure that you can access only the authorized repositories or document collections.

To set up YAML configuration for specific indexed repository, complete the following steps:

1. Create a `.wca/repo` folder at the root level of the repository.
2. Create a YAML file with the following fields:

```
repo:
- url: git@github.ibm.com:code-assistant/<my-code>.git
```

If you need to configure watsonx Code Assistant with multiple repositories, create a `repo.yaml` file for each repository that needs to be used to generate a response.

To set up YAML configuration for specific document collections, complete the following steps:

1. Create a `.wca/docs` folder at the root level of the repository.
2. Create a YAML file with the following fields:

```
docs:
- url: my_collection
```

Use case scenarios

The use case scenarios explain how RAG works and how data is retrieved from indexed repositories or documents.

Using a single code repository

If you are working in a single repository and want to use the code from this repository as context for chat conversations, complete the following steps:

1. Ensure that you have access to the repository in GitHub.
2. Index the repository in the vector store. For more information, see [Indexing code repository with IBM watsonx.ai Studio](#)
3. Create a connection asset for the repository. For more information, see [Creating a connection asset](#).
4. Generate a Git personal access token from your GitHub account and complete the setup in Visual Studio Code.

5. Open the repository in Visual Studio Code.
6. Use the `@repo` command in chat to generate a response that uses the context from the repository.

Using multiple code repositories

If you are working on a repository that has dependencies on another repository, you can use the code from both the repositories as context for chat conversations. To use both repositories, complete the following steps:

1. Ensure that you have access to both repositories in GitHub.
2. Index each repository in the vector store separately. For more information, see [Indexing code repository with IBM watsonx.ai Studio](#).
3. Create two connection assets for each vector store. For more information, see [Creating a connection asset](#).
4. Generate a Git personal access token from your GitHub account and complete the setup in Visual Studio Code.
5. In the first repository, set up the YAML configuration to list both repositories.
6. Use the `@repo` command in chat to generate a response that uses the context from both repositories.

Note: If you use same index for both repositories, the GitHub access check is not used for the second repository. If you do not have access for one of the repositories, watsonx Code Assistant generates the context from the authorized repository only.

Enabling all the users in a team to access project documentation repositories

You can allow all the users in a team to access the project documentation repositories and use the documents as context for chat conversations.

To enable access for all users:

1. Index the project documentation repositories in the vector store. For more information, see [Indexing code repository with IBM watsonx.ai Studio](#).
You can use the same index for all documentation repositories if all the users have access for the indexed project documentation repositories. If the users have an access restriction for the indexed repository, see [Enabling the users in a different subteam to access project documentation repositories](#).
2. Create a connection asset for the documentation index in the deployment space that includes all the users of the team. For more information, see [Creating a connection asset](#).
3. Use the `@docs` command in chat to generate a response that uses the context from the documentation index.

Note: The GitHub personal access tokens are not used to verify access to documentation content. You can combine several documentation repositories in the same index in the vector store. The documentation content is less restrictive than code repositories and a single index for all the documentation repositories simplifies the index management process. The authorization to access the documentation repositories is only allowed to the onboarded users in the deployment space.

Enabling the users in a different subteam to access project documentation repositories

In this use case, different subteams within a large team maintain the project documentation repositories. Access to the documentation repositories is restricted to the different subteams. Other subteams don't have access for these documentation repositories.

To enable each subteam to access the related project documentation repositories:

1. Index the documentation repositories separately in the vector store. For more information, see [Indexing code repository with IBM watsonx.ai Studio](#).
2. Ensure that the deployment space is created for each subteam that includes its users.
3. For each documentation repository index, create a connection asset in the related deployment space. For more information, see [Creating a connection asset](#).
4. Use the `@docs` command in chat to generate a response that uses the context from the documentation index.

Watsonx Code Assistant uses the documentation repository index based on the deployment space that is assigned to the users. The client-side configuration is not required.

Enabling role-based users to access project documentation repositories

In this use case, users in a team require access to different sets of documentation based on their roles. The scope of documentation that is used as context in varies across the users with no access restrictions. For example, developers need technical and API documentation only, while business analysts focus

on business process documents.

To customize the documentation repositories that are used as context:

1. Index each documentation repository in the vector store separately. For more information, see [Indexing code repository with IBM watsonx.ai Studio](#).
2. Create a connection asset for each index in the deployment space of the team. For more information, see [Creating a connection asset](#).
3. Set up YAML configuration for the required documentation index that is used as context. If you need to use multiple documentation indexes, set up YAML configuration for each documentation index.
4. Use the `@docs` command in chat to generate a response that uses the context from the documentation index.

Setting up connection assets

watsonx Code Assistant

Standard plan

A connection asset contains the connection details of your vector store. Watsonx Code Assistant uses the connection asset to connect to your vector store and fetch the context for your instructions in the chat.

Before you begin

- Review the use case scenarios. For more information, see [Use case scenarios](#).
- Ensure that you have an IBM Cloud account.
- Provision an instance of Milvus or Elasticsearch DB on IBM Cloud.
- Obtain the list of indexed code repositories and document in the vector store.

Creating a connection asset


Watsonx Code Assistant uses connection assets in your deployment space to access code repository content that is indexed in your vector store. Create a connection asset for each index in your vector store.



Note: The name of the connection asset must match the name of the code repository. For example, if the repository url is `git@github.ibm.com:my-org/mycode.git`, the name of the connection asset must be `mycode`. If you create connection asset for the documentation indexes, the name of the connection asset must be in the `docs_<name>` format. You can replace `<name>` with the name of the indexed document with the prefix `docs`.

Watsonx Code Assistant uses the description section of the connection asset to read more configuration details about your vector store.

To create a connection asset:

1. Open your watsonx Code Assistant service instance.
2. Click the **Navigation Menu** icon , then select **Deployments**.
3. Select your deployment space.
4. Go to the **Assets** tab of your deployment space.
5. Go to **Import assets > Data access > Connection**.
6. Select the Elasticsearch or Milvus connection type in **Add connection**.
 - For the Elasticsearch connection type, add the following fields:

Parameter	Description
Name	Enter the name of the Elasticsearch connection. The Elasticsearch connection name must be the same as the repository name to use the <code>@repo</code> command or start with <code>docs_</code> to use the <code>@docs</code> command.

Description Enter the additional information of the connection asset in the JSON format with the following fields:

```
$
{
  "index_name": "add the name of your index",
  "db_type": "elasticsearch",
  "ssl_assert_fingerprint": "your_fingerprint",
  "embedding_details": {
    "model_id": "msmarco-MiniLM-L-6-v3"
  }
}
```

URL Enter the connection URL for your Elasticsearch instance.

Username and Password Enter the credentials of your Elasticsearch instance.

SSL Certificate Optional: Enter the SSL certificate to create a secure connection with the Elasticsearch instance. You can't test your Elasticsearch connection without an SSL Certificate, but you can click **Save connection anyway** to save the connection.

Elasticsearch connection type settings

- For the Milvus connection type, add the following fields:

Parameter	Description
-----------	-------------

Name	Enter the name of the Milvus connection. The Milvus connection name must be the same as the repository name the <code>@repo</code> command or start with <code>docs_</code> to use the <code>@docs</code> command.
------	--

Description Enter the additional information of the connection asset in the JSON format with the following fields:

```
$
{
  "index_name": ""add the name of your index",
  "db_type": "milvus",
  "embedding_details": {
    "model_id": "msmarco-MiniLM-L-6-v3"
  }
}
```

Host Enter the host URL for your Milvus instance.

Port Enter the port number for your Milvus instance.

Database Set the Database parameter to default.

Username and Password Set the username to `ibmIhapikey` and enter an API key that is generated from the IBM Cloud account with the IBM watsonx.data instance in the Password field.

SSL Certificate	Optional: Enter the SSL certificate to create a secure connection with the Milvus instance. You can test your Milvus connection without an SSL Certificate.
-----------------	---

Milvus connection type settings

7. Click **Create**.

Indexing code repositories with IBM watsonx.ai Studio

watsonx Code Assistant

Standard plan

You can use IBM watsonx.ai Studio to index your code repositories to enable retrieval augmented generation (RAG).

Before you begin

- Provision an instance of Milvus or Elasticsearch DB on IBM Cloud before you run the notebook.
- Provision an instance of watsonx.ai Studio to run the notebook. To create a watsonx.ai Studio instance, see [watsonx.ai Studio catalog](#).
- Ensure that you have an API key to access watsonx Code Assistant to generate explanations for the functions in your code. The explanations are indexed with the code for better search results. For more information, see [Create an IBM Cloud API key](#).

Planning your index

Effective indexing is essential for optimizing the performance of watsonx Code Assistant when you use RAG. Consider the following factors when you index the repository or documents:

- Ensure that the storage space in the disk is double the size of the content that is being indexed.
- Index code repositories when you reuse existing code to generate new code, locating implementations of specific functions, answering how-to questions related to the code base. You can include main code projects and their dependencies such as API implementation.
- Index documentation repositories such as product or project documentation, technical guides, and design documents if you need watsonx Code Assistant to respond to queries based on these documents.
- Do not index rarely used code repositories or documents to reduce storage usage and minimize the time that is required for index creation and maintenance.
- Use one index per repository, especially for source code to limit the context search to specific repositories.
- Combine multiple documentation repositories in a single index if you commonly search across all documents. You can create separate indexes if a significant keyword overlap between repositories and different team or users need to search the context in the specific documentation repositories.
- Do not merge code from different repositories in the same index. Merged code reduces the accuracy of the response generation for the prompt and leads to unauthorized access of the code if the access restriction for the users is not implemented properly.
- Ensure that you manage security and authorization correctly during the indexing process. For different usage scenarios, see [Use case scenarios](#).
- Ensure that all necessary organizational approvals are obtained before you index the confidential or sensitive content. The source code and documents that are used for RAG are saved as plain text and vectorized format in a vector store outside your GitHub repository.

Indexing code repositories

1. Log in to this project: [Create RAG vector stores for watsonx Code Assistant](#).
2. Click **Create project**.
3. Select a Cloud Object Storage instance in the **Storage** field. If the Cloud Object Storage instance is not available in the drop-down list, create it.
4. Click **Create**.
5. Go to **Asset** tab in your project and click **Populate vector store** notebook.
6. Follow the instructions in the [Populate vector store](#) notebook to index the code or document.

Using watsonx Code Assistant Individual with a local IBM Granite model

watsonx Code Assistant Individual

For individual users, watsonx Code Assistant can access a local model through [Ollama](#), which is a widely used local inferencing engine for large language models. Ollama wraps the underlying model-serving project [llama.cpp](#).



Note: For increased performance and a full set of features for your organization, provision a trial of watsonx Code Assistant on IBM Cloud. For more information, see [Setting up your watsonx Code Assistant service in IBM Cloud](#).

Install the watsonx Code Assistant extension

You can set up Ollama for use within Microsoft Visual Studio Code.



Note: This setup is not available for the Eclipse IDE plug-in. It is only available with the Visual Studio Code extension.

1. Open the [watsonx Code Assistant](#) page in the Visual Studio Marketplace.
2. Click **Install** on the Marketplace page.
3. In Visual Studio Code, click **Install** on the extension.
4. In the extension settings, set **Wca: Backend Provider** to **ollama**.

Note for IBM employees: Changing the backend provider will always revert to `wcaCore` if you are an IBM or Red Hat employee using the internal WCA@IBM extension. To enable changing the backend provider:

- In the WCA@IBM extension settings, clear **Enable WCA@IBM mode for watsonx Code Assistant**
- As an alternative, disable the WCA@IBM extension

Install Ollama

- Download and run the [ollama installer](#).
- On macOS, you can also use [Homebrew](#) to install Ollama:

```
$ brew install ollama
```

Start the Ollama inference server

In a console window, run:

```
$ ollama serve
```

Leave that window open while you use Ollama.

If you receive the message `Error: listen tcp 127.0.0.1:11434: bind: address already in use`, the Ollama server is already started.

Install the IBM Granite code model

Get started by installing the `granite-code:8b` model available in the [Ollama library](#).

1. Open a new console window.
2. On the command line, type `ollama run granite-code:8b` to download and deploy the model. You see output similar to the following example:

```
$ pulling manifest
pulling 8718ec280572... 100% ██████████ 4.6 GB
pulling e50df8490144... 100% ██████████ | 123 B
pulling 58d1e17ffe51... 100% ██████████ 11 KB
pulling 9893bb2c2917... 100% ██████████ 108 B
pulling 0e851433eda0... 100% ██████████ 485 B
verifying sha256 digest
writing manifest
removing any unused layers
success
>>>
```

3. Type `/bye` after the `>>>` to exit the Ollama command shell.
4. Try out the model by typing:

```
$ ollama run granite-code:8b "How do I create a python class?"
```

5. You should see a response similar to:

```
$ To create a Python class, you can define a new class using the "class" keyword followed by the name of the class and a colon. Inside the class definition, you can specify the methods and attributes that the class will have. Here is an example: ...
```

Configure the Ollama host

By default, the Ollama server runs on IP address `127.0.0.1`, port `11434`, and http as a protocol. If you change the IP address or the port where Ollama is available:

1. In Visual Studio Code, open the extension settings for watsonx Code Assistant.
2. In **Wca > Local: API Host**, add the host IP and port.

Configure the Granite model to use

By default, watsonx Code Assistant uses the `granite-code:8b` model for both chat and code completion. If your environment has enough capacity, install the `granite-code:8b-base` model.

To use a different model:

1. Install the `granite-code:8b-base` model. See [Install the IBM Granite code model](#).
2. In Visual Studio Code, open the extension settings for watsonx Code Assistant.
3. In **Wca > Local: Code Gen Model**, enter `granite-code:8b-base`.

Securing your setup

By default, the Ollama server runs on IP address 127.0.0.1, port 11434, using http as a protocol, on your local device. To use https instead, or go through a proxy server, see the [Ollama documentation](#).

Switch from a local model to IBM Cloud

You might decide to switch from a local model to use a service instance on IBM Cloud. You can then configure Visual Studio Code to switch from a local model to IBM Cloud.

For more information, see [Setting up your watsonx Code Assistant service in IBM Cloud](#).

To update your Visual Studio Code editor to use IBM Cloud instead of Ollama:

1. Quit Visual Studio Code.
2. Quit the Ollama application.
3. Start Visual Studio Code, then open watsonx Code Assistant. You should see the message `Ollama is not running in your IDE`.
4. Click `Switch to watsonx Code Assistant on IBM Cloud`.

For an alternative method, you can change the extension settings:

1. In Visual Studio Code, open the extension settings for watsonx Code Assistant.
2. In **Wca: Backend Provider**, switch from `ollama` to `wcaCore`.
3. Restart extensions to apply the change.

Setting up your watsonx Code Assistant for Red Hat Ansible Lightspeed service in IBM Cloud

Red Hat Ansible Lightspeed

Set up your IBM Cloud service instance of watsonx Code Assistant for Red Hat Ansible Lightspeed so you can use it with your full installation of Red Hat Ansible Lightspeed.

Before you begin

Provision an instance of watsonx Code Assistant for Red Hat Ansible Lightspeed through the [IBM Cloud catalog](#) page or by working with an IBM sales representative.

The service is available in these data centers:

- Dallas (`us-south`)
- Frankfurt (`eu-de`)
- Sydney (`au-syd`) (Essentials plan only)

For more information about data centers, see the IBM Cloud documentation for [Service and infrastructure availability by location](#).

When you finish provisioning your instance, click **Setup** to open an onboarding checklist page to help you configure the instance. Or, you can select **Administration** and then **Setup** to access this onboarding checklist.

Items in that checklist correspond to each of the following steps. Click the action arrows on the checklist page that are associated with each item to complete the steps.


Create a service ID and API key

You use this API key later to enable Visual Studio Code to communicate with watsonx Code Assistant.

1. On the [Service IDs](#) page, click **Create**.
2. Enter `IBM watsonx Code Assistant user` as the name for your service ID. Optionally, enter a description.
3. Click **Create**.


You are redirected to the dashboard for your service ID, which includes tabs for **Access** and **API keys**.

4. Click the **API keys** tab.
5. Click **Create**.
6. Enter `IBM watsonx Code Assistant API key` as the name for your API key. Optionally, enter a description.
7. Click **Create**.
8. Click **Download** or **Copy** and store the API key in a safe location.

 **Important:** The key is only available for 296 seconds. Make sure to store it somewhere you can find it later.

Create a deployment space and model ID

The deployment space is the serving environment for watsonx Code Assistant. The model ID determines which model generates code recommendations in the Ansible extension for Visual Studio Code.

1. In your watsonx Code Assistant instance, click the **Navigation Menu**  icon, then select **Deployments**.
2. Click **New deployment space**.
3. Enter `Code Assistant Models` as the name for your deployment space. Optionally, enter a description.
4. In **Code assistant service**, select your instance name.
5. Click **Create**.
6. A status window appears. After the space is created, click **Close**.

7. Copy your model ID. After you create the deployment space, your model ID is provided in a field after the **Create a deployment space** checklist item. Copy this ID and store it in a safe location as you need to add it to the Red Hat Ansible Lightspeed Admin portal in a later step.

Add the service ID to the deployment space

Enable the API to connect to your deployment space.

1. On the **Manage** tab of your deployment space, click **Access control**.
2. Click **Add collaborators** and select **Add service IDs** from the menu.

A list of service IDs that are associated with your account appears.

3. Click the checkbox for the **IBM watsonx Code Assistant user** service ID that you created, and select **Editor** for the role.
4. Click **Add**.

The service ID is added to the list of collaborators.

Configure Red Hat Ansible Lightspeed to connect with watsonx Code Assistant

Add the API key and the model ID to the Red Hat Ansible Lightspeed Admin Portal to connect watsonx Code Assistant with the Ansible extension for Visual Studio Code. Follow the onboarding checklist and the instructions from Red Hat.

To add the API key and model ID to the Red Hat Ansible Lightspeed Admin Portal, see [Configuring Red Hat Ansible Lightspeed cloud service](#) in the Red Hat documentation.

Set up your network

To ensure that your developers can connect the IDE extension or plug-in, you need to allow access to IBM Cloud URLs. Make sure your organization's network allows access to:

- `https://iam.cloud.ibm.com/`
- `https://api.dataplatform.cloud.ibm.com/`

Next steps

See [Installing and configuring the Ansible VS Code extension](#) to start generating code recommendations.

Setting up your watsonx Code Assistant for Z service in IBM Cloud

watsonx Code Assistant for Z

Set up your IBM Cloud service instance of watsonx Code Assistant for Z so you can use it with your full installation of the product.

Before you begin

- Make sure that you have an [IBM Cloud account](#).
- Provision an instance of watsonx Code Assistant for Z through the [IBM Cloud catalog](#) page or by working with an IBM sales representative.

The service is available in these data centers:

- Dallas (`us-south`)
- Frankfurt (`eu-de`)

For more information about data centers, see the IBM Cloud documentation for [Service and infrastructure availability by location](#).

- Download the most recent `IBM watsonx Code Assistant for Z Multiplatform Multilingual eAssembly` from [IBM Passport Advantage](#).

When you finish provisioning your instance, click **Setup** to open an onboarding checklist page to help you configure the instance. Or, you can select **Administration** and then **Setup** to access this onboarding checklist.

Items in that checklist correspond to each of the following steps. Click the action arrows on the checklist page that are associated with each item to complete the steps.

Create a service ID


Create a service ID that will be used to connect the deployment space to the watsonx Code Assistant service.

1. On the [Service IDs](#) page, click **Create**.
2. Enter `IBM watsonx Code Assistant user` as the name for your service ID. Optionally, enter a description.
3. Click **Create**.

You are redirected to the dashboard for your service ID, which includes tabs for **Access** and **API keys**.

Create a deployment space

The deployment space is the serving environment for watsonx Code Assistant.

1. In your watsonx Code Assistant instance, click the **Navigation Menu**  icon, then select **Deployments**.
2. Click **New deployment space**.
3. Enter `Code Assistant` as the name for your deployment space. Optionally, enter a description.
4. In **Code assistant service**, select your instance.
5. Click **Create**.
6. A status window appears. After the space is created, click **Close**.

Add the service ID as a space collaborator

Enable the API to connect to your deployment space.

1. On the **Manage** tab of your deployment space, click **Access control**.
2. Click **Add collaborators** and select **Add service IDs** from the menu.

A list of service IDs that are associated with your account appears.
3. Click the checkbox for the **IBM watsonx Code Assistant user** service ID that you created, and select **Editor** for the role.
4. Click **Add**.

The service ID is added to the list of collaborators.

Generate an API key

You need an API key to get Visual Studio Code to communicate with watsonx Code Assistant.

1. Navigate to the dashboard for your service ID. Find and click the **API keys** tab, then click **Create**.
2. Enter `IBM watsonx Code Assistant API key` as the name for your API key. Optionally, enter a description. Click **Create**.
3. Click **Download** or **Copy** and store the API key in a safe location.



Important: The key is only available for 296 seconds after creation. Make sure to store it somewhere you can find it later.

For more information on adding users and managing access, see [Managing IAM access for watsonx Code Assistant for Z](#).

Set up your network

To ensure that your developers can connect the IDE extension or plug-in, you need to allow access to IBM Cloud URLs. Make sure your organization's network allows access to:

- `https://iam.cloud.ibm.com/`
- `https://api.dataplatform.cloud.ibm.com/`

Next steps

You deployed and connected your IBM Cloud components to your provisioned watsonx Code Assistant instance. You're ready to take your next steps and integrate with your IBM Z Open Editor in Visual Studio Code. For information, see [Set up a development environment](#).


Release notes for IBM watsonx Code Assistant

Use these release notes to learn about the latest updates.

25 September 2025

watsonx Code Assistant

Chat input improvements

For Eclipse, you can now use the **Copy as Markdown** icon  to copy the content of a response and preserve its formatting.

Korean display language in Eclipse

The watsonx Code Assistant plug-in for Eclipse, version 1.4.2, now supports Korean as a display language.

11 September 2025

watsonx Code Assistant


Korean display language in Visual Studio Code

The watsonx Code Assistant extension for Visual Studio Code, version 1.9.1, now supports Korean as a display language. For more information, see [Changing the display language](#).

26 August 2025

watsonx Code Assistant

Chat input improvements

For Visual Studio Code, you can now use the **Copy as Markdown** icon  to copy the content of a response and preserve its formatting.

19 August 2025

watsonx Code Assistant

Chat input improvements

In Eclipse, you can now use the **Copy prompt** icon  to copy the content of previous prompt requests.

8 August 2025

watsonx Code Assistant

Chat input improvements

For Visual Studio Code, you can now use the **Copy prompt** icon  to copy the content of previous prompt requests.

1 August 2025

Japanese display language in Eclipse

The watsonx Code Assistant plug-in for Eclipse, version 1.4.0, now supports Japanese as a display language.

Chat session names now use part of the first prompt in a chat conversation

In Eclipse, chat sessions are now named by using a portion of the first prompt in a chat. Previously each chat name used `Chat session` and a number. For more information, see [Managing chat conversation sessions](#).

25 July 2025

Retrieval augmented generation (RAG) is now available in the Standard plan

RAG is the process of optimizing the large language model (LLM) output through the prompt augmentation with the additional context. If you submit a query, watsonx Code Assistant uses the RAG tools to retrieve the information from your code bases or documentation. This relevant context is appended to the query before it is sent to the LLM model. The RAG system determines the sources that need to be included or excluded to generate a response with the most useful information. For more information, see [Setting up retrieval augmented generation \(RAG\)](#).

8 July 2025

Japanese display language in Visual Studio Code

The watsonx Code Assistant extension for Visual Studio Code, version 1.7.0, now supports Japanese as a display language. For more information, see [Changing the display language](#).

Chat session names now use part of the first prompt in a chat conversation

In Visual Studio Code, chat sessions are now named by using a portion of the first prompt in a chat. Previously each chat name used `Chat session` and a number. For more information, see [Managing chat conversation sessions](#).

26 June 2025

Fix Java unit tests

You can now start a unit test fix for any Java file that is in an application test folder of a Java project. For more information, see [Fixing unit tests](#).

19 June 2025

IBM watsonx Code Assistant and watsonx Code Assistant for Red Hat Ansible Lightspeed plans now available in the Sydney data center

All plans for watsonx Code Assistant and the Essentials plan for watsonx Code Assistant for Red Hat Ansible Lightspeed are now available in the Sydney, Australia data center on IBM Cloud. For more information, see the [IBM Cloud catalog](#).

6 June 2025

watsonx Code Assistant

Delete multiple chat sessions in Eclipse

When managing your chat conversation sessions in the Eclipse IDE, you can now delete multiple chat sessions at a time. For more information, see [Deleting chat sessions](#).

3 June 2025

watsonx Code Assistant

Delete multiple chat sessions in Visual Studio Code

When managing your chat conversation sessions in Visual Studio Code, you can now delete multiple chat sessions at a time. For more information, see [Deleting chat sessions](#).

30 April 2025

watsonx Code Assistant

Red Hat Ansible Lightspeed

watsonx Code Assistant for Z

IBM watsonx Code Assistant plans now available in the Frankfurt data center

All plans for watsonx Code Assistant, watsonx Code Assistant for Red Hat Ansible Lightspeed, and watsonx Code Assistant for Z{tag-dark-teal} are now available in the Frankfurt, Germany data center on IBM Cloud. For more information, see the [IBM Cloud catalog](#).

20 March 2025

watsonx Code Assistant

Setting to enable or disable CodeLens in Eclipse IDE

In the IDE editor, the CodeLens shows a line of generative AI options that precedes code blocks and snippets. CodeLens is available for code explanation, code documentation, and unit test generation. The Eclipse IDE plug-in now includes a new `Enable CodeLens` setting that you can use to enable or disable the CodeLens options. The default setting is enabled.

For more information, see:

- [Using the CodeLens in the editor to explain code](#)
- [Using the CodeLens in the editor to generate documentation](#)
- [Using the CodeLens in the editor to generate unit tests](#)

14 March 2025

watsonx Code Assistant

Setting to enable or disable CodeLens in Visual Studio Code

In the IDE editor, the CodeLens shows a line of generative AI options that precedes code blocks and snippets. CodeLens is available for code explanation, code documentation, and unit test generation. The Visual Studio Code extension now includes a new `Enable CodeLens` setting that you can use to enable or disable the CodeLens options. The default setting is enabled.

For more information, see:

- [Using the CodeLens in the editor to explain code](#)
- [Using the CodeLens in the editor to generate documentation](#)
- [Using the CodeLens in the editor to generate unit tests](#)

Setting to disable code completion per code language

The Visual Studio Code extension now includes a new `Completion: Disabled Languages` setting that you can use to disable code completion for specific code languages. For more information, see [Disabling code completion for specific code languages](#).

4 March 2025

watsonx Code Assistant

Chat input improvements

For Visual Studio Code, when you enter chat messages, you can now press the up arrow to reuse a previous message, use Shift+Enter for a new line, and press Enter to send a message.

27 February 2025

watsonx Code Assistant

Transferring code between the chat and the editor is now available in Eclipse

In Eclipse, when you get a code suggestion that you want to use, you can now use the **Insert at cursor** icon to insert a code block from the chat into your editor. Or, if you have code in the editor that you want to use as context in the chat, you can use the new menu choice **Add selected code to chat**. For more information, see [Transferring code between the chat and the editor](#).

5 February 2025

watsonx Code Assistant

Transferring code between the chat and the editor is now available in Visual Studio Code

In Visual Studio Code, when you get a code suggestion that you want to use, you can now use the **Insert at cursor** icon to insert a code block from the chat into your editor. Or, if you have code in the editor that you want to use as context in the chat, you can use the new menu choice **Add selected code to chat**. For more information, see [Transferring code between the chat and the editor](#).

11 December 2024

watsonx Code Assistant

Eclipse IDE plug-in now available

You can now use watsonx Code Assistant with the Eclipse IDE. For more information, see [Installing the plug-in for the Eclipse IDE](#).

IBM watsonx Code Assistant now available for on-premises installation

You can now use the watsonx Code Assistant service in an on-premises installation. For more information, see the [IBM Software Hub documentation](#).

22 November 2024

watsonx Code Assistant

Standard plan adds enterprise Java features, plus enhanced code explanation and unit test generation

The Standard plan adds new features for Java application upgrades and WebSphere to Liberty modernization, as well as enhanced code explanation and unit test generation. For more information, see: [Pricing plans](#) and the [IBM Cloud catalog](#).

15 November 2024

watsonx Code Assistant

Introducing watsonx Code Assistant

IBM watsonx Code Assistant is an innovative, generative AI coding companion that offers robust, contextually aware assistance for popular programming languages including Go, C, C++, Java, JavaScript, Python, TypeScript, and more. Seamlessly integrated into your IDE, you can accelerate your productivity and simplify coding tasks, all with trust, security, and compliance. For more information, see [Overview of watsonx Code Assistant](#).

4 September 2024

watsonx Code Assistant for Z

Release notes for **watsonx Code Assistant for Z** have moved

Release notes for **watsonx Code Assistant for Z** have moved. For the most up-to-date information, see [What's new in IBM watsonx Code Assistant for Z](#).

21 August 2024

Red Hat Ansible Lightspeed

IBM watsonx Code Assistant for Red Hat Ansible Lightspeed one-click 90-day Trial plan

The Trial plan includes the updated Red Hat Ansible Lightspeed service, an updated Visual Studio Code extension, watsonx Code Assistant for Red Hat Ansible Lightspeed, new IBM terms and conditions, updated Red Hat Ansible Lightspeed documentation, and a new Red Hat Ansible Lightspeed landing page. To use the Trial plan, see [Activate Red Hat Ansible Lightspeed with IBM watsonx Code Assistant](#).

24 June 2024

watsonx Code Assistant for Z

New capabilities

New capabilities include bug fixes.

Known issue

Generate Java classes is not supported for COBOL programs that have a hyphen (-) character in their `PROGRAM-ID` paragraph. This issue will be fixed in a future release.

If the `PROGRAM-ID` of the COBOL program for which you want to run Generate Java classes contains a hyphen character, follow the steps in the

User Action required section of this release note.

User action required

If you generate a Java class for a COBOL program with a hyphen character in its PROGRAM-ID, it fails with the error `Class Mapping Command Utilities error handler: Server error occurred while retrieving Java classes. Try again. For more information, check the full log in the Output view.`

To mitigate this issue:

1. Change the PROGRAM-ID to something that doesn't include a hyphen
2. In Visual Studio Code, right click the edited COBOL, then select watsonx Code Assistant for Z
3. Select Prepare COBOL for Transformation.
4. After you prepare the edited COBOL file, right click the same COBOL file, select watsonx Code Assistant for Z, then select Select COBOL for Transformation.

You can now generate Java classes successfully.

If you generate or regenerate a Java method and:

- It fails with the error message `Server error occurred while translating COBOL to Java. Try again. For more information, check the full log in the Output view.`
- The last message in the Output view includes `Select from Catalog db failed to find any entry for the given inputs for app name EZSCH, program name.`

You need to delete the Java class design data from your database.

1. Run two SQL queries against your database, replacing `$program-id` with the PROGRAM-ID of your COBOL program:

```
DELETE FROM EZSCH.WCAZCATALOG WHERE appName = 'EZSCH' AND fileKey = '$program-id' AND moduleName = 'oodesigner' AND secondaryKey = 'CH';
DELETE FROM EZSCH.WCAZCATALOG WHERE appName = 'EZSCH' AND fileKey = '$program-id' AND moduleName = 'semantic-mapping' AND secondaryKey = 'mapping';
```

If you use the Db2 Connect extension for Visual Studio Code, and you connected the extension to your database, you can use it to run these SQL queries:

1. Create a `cleanup_jcd.sql` file that contains the SQL queries.
2. Open the Db2 Connect extension and click the **Execute SQL File** icon
3. Choose the `cleanup_jcd.sql` file and click **Execute**.

After you run the SQL queries, you need to generate Java classes first before you can generate Java methods.

If you generate Java classes and:

- It fails with the error message `Class Mapping Command Utilities error handler: Server error occurred while retrieving Java classes. Try again. For more information, check the full log in the Output view.`
- The last message in the Output view includes `Deserializing from class hierarchy bytes from db failed with, then you need to delete Java transformation data from your database.`

You need to delete the Java transformation data from your database:

1. Run two SQL queries against your database, replacing `$program-id` with the PROGRAM-ID of your COBOL program:

```
DELETE FROM EZSCH.WCAZCATALOG WHERE appName = 'EZSCH' AND fileKey = '$program-id' AND moduleName = 'oodesigner' AND secondaryKey = 'CH';
DELETE FROM EZSCH.WCAZCATALOG WHERE appName = 'EZSCH' AND fileKey = '$program-id' AND moduleName = 'oodesigner' AND secondaryKey = 'TranslateValidateInput';
DELETE FROM EZSCH.WCAZCATALOG WHERE appName = 'EZSCH' AND fileKey = '$program-id' AND moduleName = 'semantic-mapping' AND secondaryKey = 'mapping';
```

If you use the Db2 Connect extension for Visual Studio Code, and you connected the extension to your database, you can use it to run these SQL queries:

1. Create a `cleanup_db.sql` file that contains the SQL queries.
2. Open the Db2 Connect extension and click the **Execute SQL File** icon
3. Choose the `cleanup_db.sql` file and click **Execute**.

After you run the SQL queries, you can generate Java classes for your program.

17 June 2024

watsonx Code Assistant for Z

New capabilities

New capabilities include:

- Bug fixes
- Performance improvements

3 June 2024

watsonx Code Assistant for Z

New capabilities

New capabilities include:

- Bug fixes
- Support preparation for transformation of COBOL programs to Java by using the watsonx Code Assistant for Z extension for Microsoft Visual Studio Code. You need to use IBM Passport Advantage to download version 1.1.3 of the extension, then install it in Visual Studio Code.

User action required

A defect in the backend component that causes watsonx Code Assistant for Z to infrequently return empty methods during the Generate Java methods process.

If you generate a Java method and:

- It fails with the error message `Error locating metadata needed for transformation. Try again. For more information, check the full log in the Output view.`
- The last message in the full log is `Core Mapping not found for project EZSCH`.

You need to regenerate your Java classes.

1. Click the **Generate Java classes** icon.
2. Choose the directory where you want to generate Java class files.
3. Optional: Edit any names in the Java class design view that you want to change.
4. Click **Generate Java classes**.
5. If you choose a directory that already contains Java class files with duplicate file names, you need to confirm if you want to overwrite existing files with the new files.

If you see a message similar to `Could not find OO Designer config in database for project EZSCH, program name LGACDB01` :

1. Check that the file name of the COBOL program file matches the PROGRAM-ID paragraph within the COBOL source.
2. If they differ, update the file name to match the value of PROGRAM-ID.
3. Rescan by using IBM Application Discovery and Delivery Intelligence (ADDI).

For example, a COBOL program with the following PROGRAM-ID paragraph is in a source file named `LGACDB01.cbl` .

```
PROGRAM-ID. LGACDB01.
```

16 May 2024

watsonx Code Assistant for Z

New capabilities

New capabilities include bug fixes.

User action required

If you generate a Java method and:

- It fails with the error message `Error locating metadata needed for transformation. Try again. For more information, check the full log in the Output view.`
- The last message in the full log is `Core Mapping not found for project EZSCH`.

You need to regenerate your Java classes.

1. Click the **Generate Java classes** icon.
2. Choose the directory where you want to generate Java class files.
3. Optional: Edit any names in the Java class design view that you want to change.
4. Click **Generate Java classes**.
5. If you choose a directory that already contains Java class files with duplicate file names, you need to confirm if you want to overwrite existing files with the new files.

If you see a message similar to `Could not find OO Designer config in database for project EZSCH, program name LGACDB01` :

1. Check that the file name of the COBOL program file matches the PROGRAM-ID paragraph within the COBOL source.
2. If they differ, update the file name to match the value of PROGRAM-ID.
3. Rescan by using IBM Application Discovery and Delivery Intelligence (ADDI).

For example, a COBOL program with the following PROGRAM-ID paragraph is in a source file named `LGACDB01.cbl`.

```
PROGRAM-ID. LGACDB01.
```

2 May 2024

watsonx Code Assistant for Z

New capabilities

New capabilities include:

- Bug fixes
- Performance enhancements for transform APIs (generate Java classes)
- Quality improvements for generated Java methods

User action required

The generated Java class might include a `reset()` method.

If you see a message similar to `Could not find OO Designer config in database for project EZSCH, program name LGACDB01` :

1. Check that the file name of the COBOL program file matches the PROGRAM-ID paragraph within the COBOL source.
2. If they differ, update the file name to match the value of PROGRAM-ID.
3. Rescan by using IBM Application Discovery and Delivery Intelligence (ADDI).

For example, a COBOL program with the following PROGRAM-ID paragraph is in a source file named `LGACDB01.cbl`.

```
PROGRAM-ID. LGACDB01.
```

22 April 2024

watsonx Code Assistant for Z

New capabilities

New capabilities include:

- Bug fixes

- Performance enhancements to transform APIs (generate Java classes)

User action required

If you generate a Java method and it fails with the error message `Invalid Java selected for transformation or error locating COBOL needed for transformation. Try selecting valid Java to translate. For more information, check the full log in the Output view.`, then you need to regenerate your Java classes.

1. Click the **Generate Java classes** icon.
2. Choose the directory where you want to generate Java class files.
3. Optional: Edit any names in the Java class design view that you want to change.
4. Click **Generate Java classes**.
5. If you choose a directory that already contains Java class files with duplicate file names, you need to confirm if you want to overwrite existing files with the new files.

The generated Java class might include a `reset()` method, which can't be generated with the Generate Java method flow, instead returning exceptions such as: `Failed to get COBOL paragraph name for method reset. For more information, check the full log in the Output view.`

You need to delete the `reset()` method from the generated Java code. This method will not be automatically generated in a future release of watsonx Code Assistant for Z.

If you see a message similar to `Could not find OO Designer config in database for project EZSCH, program name LGACDB01` check that the file name of the COBOL program file matches the PROGRAM-ID paragraph within the COBOL source and, if they differ, update the file name to match the value of PROGRAM-ID, and then rescan by using IBM Application Discovery and Delivery Intelligence (ADDI).

For example, a COBOL program with the following PROGRAM-ID paragraph is in a source file named `LGACDB01.cbl`.

```
PROGRAM-ID. LGACDB01.
```

5 April 2024

watsonx Code Assistant for Z

New capabilities

New capabilities include:

- REST API enhancements
- Bug and security fixes

User action required

Method generation might fail with the message `Cannot find COBOL paragraph`. The cause is that the source code is missing a section name for the program. The format of the paragraph name that is used to distinguish the paragraph location is missing. To fix the issue:

1. To capture the paragraph, add a section name in `PROCEDURE DIVISION`.
2. Rescan the file with ADDI to restart the process for transforming the program.
3. To verify, when you generate the classes at the start of the Z Open Editor plug-in process, check the method and its linked paragraph name. For example, look for something similar to `<SectionName>:FIRST_SENTENCES`.

18 March 2024

watsonx Code Assistant for Z

New capabilities

New capabilities include:

- REST API enhancements
- Bug and security fixes

User action required

When you use the IBM Z Open Editor extension to generate Java classes, the main method might be in the list of methods eligible for code generation from the LLM. If so, you need to delete your saved Java class design from your Db2 database.

To delete your class design, run a SQL query against your database. Replace `$program-id` with the PROGRAM-ID of your COBOL program.

```
DELETE FROM EZSCH.WCAZCATALOG WHERE appName = 'EZSCH' AND moduleName = 'semantic-mapping' AND secondaryKey = 'mapping' AND fileKey = '$program-id' AND version = '1.0.1';
```

If you use the Db2 Connect extension for Visual Studio Code, and connected the extension to your database, you can use it to run the SQL query:

1. Create a `delete_mapping.sql` file that contains the SQL query.
2. Open the DB2 Connect extension and click the **Execute SQL File** icon.
3. Choose the `delete_mapping.sql` file and click **Execute**.

After you run the SQL query, regenerate the Java class design and then generate the Java classes in the IBM Z Open Editor. The list of methods eligible for code generation omit the main method.

If you generate a Java method and it fails with the error message `Invalid Java selected for transformation or error locating COBOL needed for transformation. Try selecting valid Java to translate. For more information, check the full log in the Output view`, then you need to regenerate your Java classes.

1. Click the **Generate Java classes** icon.
2. Choose the directory where you want to generate Java class files.
3. Optional: Edit any names in the Java class design view that you want to change.
4. Click **Generate Java classes**.
5. If you choose a directory that already contains Java class files with duplicate file names, you need to confirm if you want to overwrite existing files with the new files.

During class generation, some suggested Java variables might be missing their type, for example:

	Type		
payment	int	PAYMENT	variable
commission	int	COMMISSION	variable
customer		CUSTOMER	variable

Example of a missing type for a Java variable

A missing type doesn't affect the actual class generation / Java class design process, or the conversion itself, and will be addressed in a future release.

Also, from the example, Java variable name suggestion might have an initial uppercase character (Customer rather than customer). If you accept the suggested variable name, and continue to generate classes, the Java code is incorrect. You need to override the suggested variable names, for example, change `Customer` to `customer`, before you click **Generate Java classes**.

15 February 2024

Red Hat Ansible Lightspeed

New pricing plans

Watsonx Code Assistant for Red Hat Ansible Lightspeed is now available in three different pricing plans, each of which offers different features and is subject to different resource constraints. For more information about the Lite, Essentials, and Standard plans, see [IBM watsonx Code Assistant for Red Hat Ansible Lightspeed pricing plans](#).

02 February 2024

New capabilities

New capabilities include:

- A subset of IMS commands within COBOL code can be converted to Java
- A subset of CICS commands within COBOL code can be converted to Java
- REST API enhancements to enable more capabilities to be added to future IBM Z Open Editor releases

User action required

Generation of classes must be rerun by the VS Code user to enable successful conversion calls.

29 October 2023

Introducing Watsonx Code Assistant for Red Hat Ansible Lightspeed and watsonx Code Assistant for Z

Watsonx Code Assistant for Red Hat Ansible Lightspeed and watsonx Code Assistant for Z are now available.

Prompting guide for IBM watsonx Code Assistant

Learn about prompting techniques that maximize your use of watsonx Code Assistant.

IBM watsonx Code Assistant:

- Captures and transmits prompt and context information for inference and captures feedback from the user to improve model and service quality
- Combines and processes natural language and code to get code suggestions from large language models (LLMs)
- Finds training examples that are similar to the code suggestions
- Collects and processes feedback data to improve model quality and user experience

One of the key advantages of watsonx Code Assistant is its ability to use context and industry best practices to generate intelligent suggestions. As you use watsonx Code Assistant, it incorporates context from the overall source file and the specific paragraph to provide more accurate recommendations. It analyzes the code that you wrote and suggests improvements or additions based on established best practices.

Prompt engineering primer

Parameterization, in the context of prompt engineering, covers the adjusted settings that go into the behavior of your model for your use case. Here are some key prompt engineering parameters and how they work:

- **Temperature:** Controls the randomness of responses. Lowering it to approximately 0.1 yields consistent, predictable outputs, while raising it (approximately 0.9) produces creative responses with varying degrees of variability.
- **Stop sequences:** Tells the model when to end its response and acknowledge when a question has been fully answered.
- **Top-k:** Sets a hard limit on how many word choices the model considers at each step of its thought process. For example, if the top-k is set to 50, the model picks from only the 50 most likely next words, ignoring less probable ones.
- **Top-p sampling:** Also known as nucleus sampling, it works by considering words based on their cumulative probability. With a lower value, the model sticks to high-confidence words, giving you more focused and reliable answers.
- **Frequency versus presence penalties:** Prevents repetition. Frequency penalizes reusing words, while presence encourages exploring new topics.
- **Logit bias:** You can select specific words to be more or less likely to show up in ones responses. This is helpful when you want to steer the model toward or away from certain terminologies.

For the purposes of code generation, watsonx Code Assistant has optimized backend parameters like *temperature*, where the value threshold is set low enough to provide consistent and predictable outputs while allowing creative responses when appropriate. Additionally, watsonx Code Assistant couples this with *stop sequences* that are clear and produce structured functional code that adheres to proper syntax rules.

Because watsonx Code Assistant doesn't expose these parameters, you are effectively practicing *hard prompting* and *soft prompting* by using inputs in *prompt chat* and *inline in the code directly*.

Hard prompting

Code generation is inherently an exercise in hard prompting, requiring you to provide specific instructions as prompts to the AI domain expert to receive an accurate response. As a warmup, ask watsonx Code Assistant to do something generic. For example:

```
Write Python code to sort an array using the bubble sort algorithm.
```

Watsonx Code Assistant might respond with:

```
$ Here's a Python implementation of the bubble sort algorithm:
```

```
# Assisted by watsonx Code Assistant
```

```
def bubble_sort(arr):  
    n = len(arr)  
  
    for i in range(n):  
        for j in range(0, n - i - 1):  
            if arr[j] > arr[j + 1]:  
                arr[j], arr[j + 1] = arr[j + 1], arr[j]
```

```
# Example usage:
```

```
arr = [64, 34, 25, 12, 22, 11, 90]
```

```
bubble_sort(arr)
print("Sorted array is:", arr)
```

This code defines a function `bubble_sort` that takes an array as input and sorts it using the bubble sort algorithm. The outer loop iterates through the array, and the inner loop compares adjacent elements and swaps them if they are in the wrong order. The process repeats until the array is sorted. The example usage demonstrates how to use the `bubble_sort` function to sort an array.



Note: The nature of the question was so generic yet specialized in its domain that the LLM provided an example usage without being prompted for it.

The challenge with hard prompting is that it's possible to make the prompt too hard. If you use hardcoded information in your prompt, there's an excellent chance that the response is anonymized or otherwise different from what you prompted. For example, if you prompt with:

```
$ You are a Cisco NXOS engineer. Write a script that sets the secondary interface IPv4 address for Ethernet1/2 to 194.99.2.254/24.
```

You might get a response that uses a different number in the fourth octet of the IPv4 address, or a different IPv4 address entirely.

This is where the other definition of parameterization applies, which is the use of parameters within the syntax of the language you are using. Consider this improvement:

```
$ You are a Cisco NXOS engineer. Write a script that sets the secondary interface IPv4 address for a parameterized layer 1 interface name to a parameterized IPv4 address formatted as CIDR.
```

You might even give an example:

```
$ You are a Cisco NXOS engineer. Write a script that sets the secondary interface IPv4 address for a parameterized layer 1 interface name like "Ethernet1/2" to a parameterized IPv4 address formatted as CIDR like "192.168.1.200/24."
```

Experiment with different approaches, but the use of hardcoded data is an anti-pattern.

Soft prompting

Conversely, when code generation occurs without explicit instructions and the AI is provided with subtle guidance, the AI generates prompts as prefix vector embeddings that are incomprehensible to humans. This phenomenon is commonly referred to as *soft prompting*.

Watsonx Code Assistant benefits from this technique in the form of single and multiline code completion, which makes the solution so powerful. For example, prompt Watsonx Code Assistant to do something traditional in the context of object-oriented programming:

```
$ Create a python class Customer. Include attributes for firstName, lastName, and age.
```

Watsonx Code Assistant would respond with:

```
$ Here's a Python class named Customer with attributes for firstName, lastName, and age:
```

```
# Assisted by watsonx Code Assistant
```

```
class Customer:
```

```
    def __init__(self, first_name, last_name, age):
        self.first_name = first_name
        self.last_name = last_name
        self.age = age
```

```
    def full_name(self):
        return f"{self.first_name} {self.last_name}"
```

```
    def is_adult(self):
        return self.age >= 18
```

This class has three attributes: `first_name`, `last_name`, and `age`. The `__init__` method initializes these attributes when a new `Customer` object is created. The `full_name` method returns the customer's full name, and the `is_adult` method checks if the customer is an adult (18 years or older).



Note: The example method is deliberately omitted in the response to articulate the following point.

Now for the fun part. You've your hard prompting efforts from the prompt chat into your code file. Your results, which in effect act as a parameterization for soft prompting, take over when asked to provide an example usage of your code inline, for example:

```
class Customer:
    Explain | Document | Unit Test
    ... def __init__(self, first_name, last_name, age):
    ...     self.first_name = first_name
    ...     self.last_name = last_name
    ...     self.age = age

    Explain | Document | Unit Test
    ... def full_name(self):
    ...     return f"{self.first_name} {self.last_name}"

    Explain | Document | Unit Test
    ... def is_adult(self):
    ...     return self.age >= 18

# Example usage
customer1 = Customer("John", "Doe", 25)
customer2 = Customer("Jane", "Doe", 15)

print(customer1.full_name()) # Output: John Doe
print(customer1.is_adult()) # Output: True
print(customer2.is_adult()) # Output: False
```

Soft prompting example

As expected, it instantiates customer object examples and provides what those example results would be.

Summary

With hard prompting, parameterization is driven by you. Watsonx Code Assistant facilitates this in the chat. Conversely, with soft prompting your entire contextualized code is one significant parameter that is driven by the IBM Granite model, which is used by watsonx Code Assistant and facilitated within the inline code directly.

Simple ways to enhance your experience

Use these approaches to enhance your experience when you write prompts.

Use personas or role names to provide context

The LLM supporting watsonx Code Assistant is tuned to support 116 different programming languages. When you prompt for content, explanations, or opportunities for improvement, it's important to provide context that includes the role or persona that would otherwise be expected to produce the content without AI.

Start with a basic query for language recommendations. Suppose you work for an operational technology company that produces embedded systems, sensors, and other instruments. You want to implement the UNIX `cat` command, and ask the following question:

Good: `Between C, C++, and Ada, which language is best for implementing the UNIX "cat" command?`

This is a good start, and one improvement would be to start with a sentence that provides more context. Consider the following:

Better: You are an embedded systems programmer. Between C, C++, and Ada, which language is best for implementing the UNIX "cat" command?

This is better because the needed skill set is now much clearer.

Finally, it would help to include any specific requirements. For example:

Best: You are an embedded systems programmer building secure fielded instruments. Between C, C++, and Ada, which language is best for implementing the UNIX "cat" command?

This is an improvement because the prompt includes the security requirement, preventing the need for a follow-up prompt such as **What if I need robust security capabilities?**

You can use these examples to build a strong persona line:

Element	Example
Role	Backend developer, site-reliability engineer
Seniority	Junior, senior, principal
Domain	Fintech, healthcare, edge devices
Key constraints	FIPS, low latency, 256 MB RAM cap
Audience or voice	Mentoring a junior, writing a blog post

Persona examples

Terraform module examples

Level	Prompt
Good	Write Terraform to deploy an S3 bucket.
Better	You are a cloud engineer. Write Terraform to deploy an encrypted S3 bucket.
Best	You are a senior cloud engineer in a healthcare firm. Write Terraform that deploys an S3 bucket with SSE-KMS, public access blocked, and versioning on.

Terraform module examples

Python ETL script examples

Level	Prompt
Good	Write a Python script that loads a CSV into PostgreSQL.
Better	You are a data engineer. Write a Python script that loads a CSV into PostgreSQL using psycopg2.
Best	You are a data engineer at a pharma company. Write a Python script that loads a large CSV into PostgreSQL using COPY, tracks row count, and logs failures to CloudWatch.

Python ETL script examples

Security review examples

Level	Prompt
Good	Review this Node.js code for bugs.
Better	You are an application security auditor. Review this Node.js code for common OWASP issues.
Best	You are an application security auditor preparing a PCI report. Review this Node.js Express handler for OWASP Top 10 issues and suggest fixes inline.

Security review examples

Postgres tuning examples

Level	Prompt
Good	Suggest Postgres settings for high load.
Better	You are a database administrator. Suggest Postgres settings for steady 5 k writes per second.
Best	You are a senior DBA at an ad-tech firm. Suggest Postgres 15 settings to sustain 5 k writes per second on an m6i.2xlarge, 64 GB RAM, 2 TB gp3.

Postgres tuning examples

Generate API version-specific code

When you work with certain APIs such as Azure API, the code uses an API with a specific version date. Given the following prompt:

```
$ Write a bicep to create a keyvault.
```

The code output that is generated by watsonx Code Assistant includes the following dated Azure API version:

```
$ ...
resource keyVault 'Microsoft.KeyVault/vaults@2019-09-01' = {
  name: keyVaultName
```

But is that the newest API version available in the training data? You can ask the model:

```
$ List only the dates of the KeyVault Azure API available to you.
```

Produces output similar to this example:

```
$ Here are the dates for the available Key Vault API versions as of April 2024:
```

```
2015-06-01
2016-10-01
2017-09-01
2018-02-14
2018-11-30
2019-09-01
2019-10-01
2020-01-01
2021-04-01
2021-10-01
2023-02-01
```

You can take this information and request that a specific API version is used:

```
$ Write a bicep to create a keyvault using this API version Microsoft.KeyVault/vaults@2023-02-01
```

Produces output similar to this example:

```
$ ...
resource keyVault 'Microsoft.KeyVault/vaults@2023-02-01'
```

Encapsulate separate contexts into separate chats


The watsonx Code Assistant extension for Microsoft Visual Studio Code is based on a chat-oriented core. When you start a new chat, there is no context until after you submit your first prompt. Using the context and persona practices, you can establish context early and link to specific files in your repo by using a reference command in the syntax `@<filename>`. Your subsequent prompts can then refer to previous outputs, and you can use relative language without needing to reinput your progress or generated content.

For example, by using the initial prompt:

```
$ You are an Ansible engineer. Begin developing a new Ansible role called mq-ocp that deploys IBM MQ to OpenShift 4 with an exposed route. Use the kubernetes.core collection. Use fully qualified collection names. Use parameterization for the kubernetes and MQ resources wherever possible. Format the output as tasks only, suitable for tasks/main.yml.
```

When you are happy with the generated content, you can follow up with subsequent prompts such as:

```
$ Generate defaults/main.yml.  
  
Generate argSpec.  
  
Generate meta/main.yml.  
  
Generate a README.md that includes a synopsis, prerequisites, dependencies, up & running, task explanations, parameter explanations, author, and license information.  
  
Generate a top-level play called ibm-mq-day0-playbook.yml that runs this role on an inventory called rhel9servers.
```

 **Note:** Before you start a new coding session, be sure to start a new chat if you need to start from a clean slate.

Add important details to your prompts

Remember when you first learned how to code, when your instructor said computers do exactly what you tell them to do? LLMs aren't different in that regard. Be sure to add important details to your prompts. For example:

Good: `You are an Ansible engineer. Write a playbook that deploys IBM MQ on OpenShift.`

A good start, but you might get content that supports older versions of Red Hat OpenShift like OpenShift Container Platform 3. If you need Red Hat OpenShift 4 support, modify your prompt by adding a `4` at the end.

Better: `You are an Ansible engineer. Write a playbook that deploys IBM MQ on OpenShift 4.`

This is more specific, but then you notice that the proposed content suggests passing Red Hat OpenShift CLI commands instead of using a bespoke Ansible module, which might not be idempotent. Further, the proposed content uses short module names, which is a linting violation. In Red Hat OpenShift, services are not exposed as routes by default.

Best: `You are an Ansible engineer. Write a playbook that deploys IBM MQ on OpenShift 4 with an exposed route. Use the kubernetes.core collection. Use fully qualified collection names.`

By being more specific, you can generate more useful code and save time making corrections. The good news here is that watsonx Code Assistant retains context in each chat, which means you can propose corrections conversationally instead of having to rewrite the entire prompt.

Kubernetes manifest examples

Level	Prompt
Good	<code>Generate a Deployment for nginx.</code>
Better	<code>Generate a Kubernetes Deployment for nginx 1.25.</code>
Best	<code>You are a platform engineer. Generate a Kubernetes Deployment for nginx 1.25 in namespace web, 2 replicas, CPU 250m, memory 256Mi, plus a ClusterIP Service.</code>

Kubernetes manifest examples

Java unit test examples

Level	Prompt
Good	<code>Write a JUnit test for my Calculator class.</code>
Better	<code>You are a Java developer. Write a JUnit 5 test for the add() method in Calculator.</code>
Best	<code>You are a senior Java developer. Write a JUnit 5 test for Calculator.add() covering positive, negative, and overflow cases. Mock external calls with Mockito.</code>

Java unit test examples

PowerShell backup script examples

Level	Prompt
-------	--------

Good Create a script to back up files.

Better You are a sysadmin. Create a PowerShell script that zips C:/Logs nightly.

Best You are a sysadmin. Create a PowerShell script that zips C:/Logs nightly, retains 30 archives, and writes to the Windows event log.

PowerShell backup script examples

Also good to remember

Neatness counts: Be sure to capitalize the beginning of your sentences, separate sections by using commas, and end sentences with periods or question marks. You might even find that good manners get you different results. Try using "please" and see what happens!

Record your good prompts: It's good practice to create a simple `prompts.md` file to keep your prompts that produce good results. You can use them to test updated LLMs as well.

Test your knowledge so far

Here are some example prompts for you to consider. First, how might you consolidate these two prompts into one, and then how would you improve the final version? What might you need to clarify later?

```
$ Update the @deploy.bicep for keyvault to include an event grid system topic which will stream to an azure logic app uri
```

The section above has a source and topic type missing. Can you please update this to include those parameters?

Consider this more complex prompt to build an SSH factory by using Rust. What do you notice about it? How specific is it? What would you change?

```
$ You are a Rust programmer. Generate a Rust struct named SshFactory for managing SSH connections. The struct should have fields for inventory, a flag to forward the SSH agent, and an optional login password. Implement the ConnectionFactory trait with methods get_local_connection and get_connection to create local and remote SSH connections. Use the ssh2 crate for SSH functionality and ensure the code is thread-safe with appropriate use of Arc and RwLock.
```

Final tips for an optimal experience

1. Although generative AI can be incredibly useful, it can sometimes produce answers that aren't exactly what you're looking for, or it can give you suggestions that seem off track. If this happens, don't hesitate to rewrite or reformulate your request. Your opinion is fundamental in guiding the AI to provide more precise, meaningful, and relevant answers.
2. Don't hesitate to use extensive prompts. In certain scenarios, this approach can enhance the accuracy of prompt suggestions that are generated by the AI. A well-crafted prompt is crucial for optimal outcomes.

Using watsonx Code Assistant



Getting code suggestions

watsonx Code Assistant



Generate code suggestions by using chat conversations or completing code in the editor.

Using chat conversations

Enter a prompt that explains the code you need, and watsonx Code Assistant generates something you can choose to use.


1. To start a new chat conversation, click the **New chat** icon .
2. Ask a question or give an instruction.
3. Click the **Send** icon . Watsonx Code Assistant sends your input to the code model, and shows the response in the chat.

Select the message where watsonx Code Assistant generated the last good result, and continue the chat conversation from there in a new chat.

1. To use a previous or an existing message in a new chat, click the **Start new chat** icon .
2. Ask a question or give an instruction.
3. Click the **Send** icon . Watsonx Code Assistant sends your input to the code model, and shows the response in the chat.

Transferring code between the chat and the editor

When you get a code suggestion that you want to use, you can insert a code block from the chat into your editor.

1. Position your cursor in the editor.
2. In the chat, click the **Insert at cursor** icon .

If you have code in the editor that you want to use as context in the chat, you can copy the code into the chat.

1. Highlight the code in the editor.
2. Right click to show the pop-up menu.
3. Choose **watsonx Code Assistant**.
4. Choose **Add selected code to chat**.

Using natural language

Each chat message is part of a chat conversation. Keep conversations focused around a specific subject or task. When you switch your context to another programming language, another project in your workspace, or a different programming task, create a new chat conversation. You get more relevant results for your questions.



Examples:

- Create a Python class Customer. Include attributes for firstName, lastName and age.
- Create a JavaScript function that fetches data from a REST API endpoint, dynamically populates an HTML table with the results, and updates the UI asynchronously

Using the prompt library

Watsonx Code Assistant includes a prompt library that contains a set of common, natural-language coding requests. You can use them as-is in your chat, or modify them for your use case.

To access the prompt library:

1. Click the **Open menu** icon .
2. Click **Prompt library** to open it.
3. Click the **Copy to clipboard** icon .
4. Click **Prompt library** again to close it.
5. Paste the prompt into the chat.

Writing effective chat messages

For the best results in your conversation:

- Write your chat messages in English.

- Start with a simple and clear instruction.
- Use subsequent chat messages to refine the code output of the model.
- Be as specific and detailed as you can for each step. You can gradually enhance the created code through subsequent chat messages.
- Use file and method references in your message to provide relevant context. For example, if you want the model to create a method that is similar to another method, add `similar to @<method>` to your message.
- If you find the answers become less relevant, or if you start a new task, create a new chat conversation, and work from there. It is better to have many short chat conversations, each with a specific context, rather than one large conversation that might confuse the model with different and unrelated chat messages.

Referencing code in your workspace

To ask questions or refine a specific file, class, function, or method in your local workspace, you can use a *code reference*. These references provide important context and can help to increase the accuracy of the answer.

1. Type the `@` symbol as part of your chat message to see a list of files, classes, and methods from your workspace.
2. Filter the list by typing characters of the file, class, or method name you want to reference.
3. Click to select the reference. Watsonx Code Assistant sends the contents of the reference as part of your message.

Note: Entries in `.gitignore` are not included as code reference choices. If you have sensitive data, such as `.env` files, make sure to add them to `.gitignore`.

When you open a local workspace folder, watsonx Code Assistant creates an index of these items in memory so you can reference these files and functions in the chat. Your editor also indexes files that you add or change during your session. The index contains up to 1,000 of the most recent files in these programming languages:

- C
- C++
- Go
- Java
- JavaScript
- Python
- TypeScript

Note: File indexing works only on local repositories.

When you reference a method, or use CodeLens on a method, watsonx Code Assistant supports certain code languages. Referencing a full file works for all languages. For more information, see [Language support when you work with methods](#).

When you reference a file, the size limit is 50 KB. If you reach this limit, split the file into individual functions and reference each function. Or, split the file at 49 KB, taking care of function boundaries, and reference the file at each split. With either approach, you need to merge the results.

Use cases with examples that reference code:

Use case	Example message
Generate a function based on an existing function	Create a method <code>send_translate_message</code> that is similar to <code>@send_code_explanation_message</code>
Generate a unit test that follows existing unit tests	Create a unit test for <code>@getName</code> that is similar to the unit tests in <code>@testLoadTablesChildConnectionReceiverJob.h</code>
Enhance existing functions	Add error handling and log statements to <code>@format_documents</code>
Enhance existing functions	Update <code>@setEmail</code> with error handling for null strings
Explain code	What does <code>@main.py</code> do
Explain code	Explain the logic of <code>@send_invoice</code>

Generate documentation for functions and classes

Add JavaDoc to @Customer


Chat messages that reference code

Retrieval augmented generation

Standard plan

With the Standard plan, your organization can configure retrieval augmented generation (RAG), which is the process of optimizing the large language model (LLM) output through the prompt augmentation with the additional context. For more information, see [Setting up retrieval augmented generation \(RAG\)](#).

If you submit a query, watsonx Code Assistant uses the RAG tools to retrieve the information from your code bases or documentation. This relevant context is appended to the query before it is sent to the LLM model. The RAG system determines the sources that need to be included or excluded to generate a response with the most useful information.

 **Important:** RAG requires the watsonx Code Assistant Standard plan and the Visual Studio Code extension. It isn't available for use with the Eclipse plug-in.

When RAG is enabled for your Git repositories, enter `@repo < instruction >` or `@docs < instruction >` in chat conversations to generate a response that uses the context from the referenced repository or documents in the vector store. Replace the `< instruction >` parameter with a prompt message.

- Sample syntax to use the referenced code base repository:

```
@repo how is a chat message processed?
```

- Sample syntax to use the referenced document collection:

```
@docs What are the steps to setup a connection to the user data store?
```

Using chat commands

Chat commands that you can use to perform certain tasks:


Command	Description	For more information
<code>/document</code>	Generate documentation for a referenced file or a method or a class in the active workspace	Generating documentation
<code>/docs</code>	Ask questions that reference IBM and Red Hat product documentation	Getting answers from IBM documentation
<code>/explain</code>	Use generative AI to analyze and summarize your code to understand what the code does	Explaining code
<code>/help</code>	Get quick assistance about tasks	Type <code>/help</code> in the chat
<code>/unit-test</code>	Generate unit tests for various programming languages	Generating unit tests
<code>/translate</code>	Translate code from one language to the other	Translating code
<code>@repo</code>	For use with retrieval augmented generation (RAG)	Enabling RAG
<code>@docs</code>	For use with retrieval augmented generation (RAG)	Enabling RAG

Chat commands

Managing chat conversation sessions

You can use the **Chat sessions** menu to manage your existing chat conversation sessions.

To open the menu:


1. Click the **Open menu** icon .
2. Click **Chat sessions**.

Switching chat conversation sessions

To switch between chat conversation sessions, select the session from the chat sessions list.


Renaming a chat session

To rename a chat conversation session:



1. On the chat session, click the overflow menu icon .
2. Click **Rename**.
3. Type the new name, then press Enter.

Deleting chat sessions

To delete a single chat conversation session:

1. On the chat session, click the overflow menu icon .
2. Click **Delete**.

To delete multiple chat conversation sessions:

1. Click the **Select chats** icon .
2. Select the chats you want to delete, or click **Select all**.
3. Click the delete icon , then click the **Delete** button to confirm.

Completing code in the editor

Get code suggestions by entering code or comments in your editor.

Using the editor in Eclipse

For C, C++, Go, JavaScript, Python, and TypeScript:

1. Right-click the file.
2. Click **Open with**, then click **Text Editor**.

For Java, inline code generation with the Java Editor in the Eclipse IDE is supported in watsonx Code Assistant but first needs to be set up.

1. Go to Content Assist in the Eclipse settings.

Operating system	Path
MacOS	Eclipse > Settings > Java > Editor > Content Assist
Windows	Window > Preference > Java > Editor > Content Assist

Navigation to Content Assist

2. In the **Insertion** section, clear **Insert single proposals automatically**.
3. In the **Auto Activation** section, add `._abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ` to **Auto activation triggers for Java**, and confirm that **Enable auto activation** is checked.
4. Click **Apply and Close**.

Single-line completion

As you write a line of code in your editor, watsonx Code Assistant can suggest code to complete it.

1. Start typing a line of code, then pause.
2. Watsonx Code Assistant adds a code suggestion to complete the line that you typed.
3. In Visual Studio Code, press **Tab** to accept the suggestion. In Eclipse, press **Return** or **Enter** to accept the suggestion. Or press **Esc** to reject.

Multiline completion

You can get a multiline code suggestion based on a line of code or a comment that you enter.

1. Start typing a line of code, or enter a comment that describes the code you want.

For best results, include the expected programming language in your comment, for example:

```
$ // Sort an array with a fast sort function (TypeScript programming language)
```

2. Use a keyboard shortcut.

Operating system	Shortcut
MacOs	Option+.
Windows	Alt+.

Default keyboard shortcuts

3. Watsonx Code Assistant adds a multiline code suggestion.
4. In Visual Studio Code, press **Tab** to accept the suggestion. In Eclipse, press **Return** or **Enter** to accept the suggestion. Or press **Esc** to reject.

Disabling code completion for specific code languages

If necessary, you can use the `Completion: Disabled Languages` setting for the Visual Studio Code extension to disable code completion for specific code languages.

1. Open the watsonx Code Assistant extension settings.
2. Click **Add Item** to select a code language. Choices are:

Choice	Code language
<code>c</code>	C
<code>cpp</code>	C++
<code>go</code>	Go
<code>java</code>	Java
<code>javascript</code>	JavaScript
<code>javascriptreact</code>	JavaScript React
<code>python</code>	Python
<code>typescript</code>	TypeScript
<code>typescriptreact</code>	TypeScript React

Code languages

To re-enable a code language, you can remove it from the list.

1. Click the code language in the list.
2. Click the **Remove Item** icon to remove it from the list of disabled languages.

Changing keyboard shortcuts

You can change the keyboard shortcuts if the default combinations are already assigned to other tasks.

To change keyboard shortcuts in Visual Studio Code:

1. Click the **Code** menu.
2. Select **Settings**.
3. Select **Keyboard Shortcuts**.
4. Search for `watsonx`.
5. For the shortcut that you want to change, click the **Change Keybinding** edit pencil icon and assign a different key combination.


To change keyboard shortcuts in Eclipse:

1. Click the **Eclipse** menu.
2. Select **Settings**.
3. Type `Keys` to filter the settings, then click **Keys**.
4. Type `WCA` to filter the list.
5. Select the keyboard shortcut that you want to change.
6. In the **Binding** field, enter a new key combination.

Disabling code completion

If necessary, you can disable the completion of code in the editor window.

To disable code completion in Visual Studio Code:

1. In the lower right corner of the editor, click the **Disable watsonx Code Assistant Completions** icon .
2. You can re-enable completions by clicking the **Enable watsonx Code Assistant Completions** icon again.

To disable code completion in the Eclipse IDE:

1. In the editor toolbar, click the **Disable watsonx Code Assistant Completions** icon.
2. You can re-enable completions by clicking the **Enable watsonx Code Assistant Completions** icon again.

Best practices for generating code suggestions

Use these tips to get the best results.

Use chat for longer code blocks. Use in-editor code generation for refinement and boilerplate code.

Use chat to:

- Generate an outline of a class
- Specify a method in detail
- Refine the suggested code with more instructions (for example, rename or replace)

Use in-editor code generation to:

- Complete the line of code that you started to type
- Automatically create boilerplate code, such as getters and setters
- Create methods and code that can be described with a single comment line

Comment-to-code: use descriptive comments, not instructions

When you use comment-to-code in the editor, write a comment that describes the intended behavior, similar to how you write comments for code you wrote. For example, use `//return even numbers from an arraylist` or `//method that returns even numbers from an arraylist`.

Don't write your comment as an instruction, such as `//write a method that returns even numbers from an arraylist`. Watsonx Code Assistant uses models that are trained to complete code on data that contains many typical, descriptive comments, so these types of comments yield better results.

Context used for in-editor code generation

The key ingredient for code generation is context, that is, the surrounding code that you pass to the model. For in-editor code generation, watsonx Code Assistant uses:

1. The 20 lines of code before the line where the generation is triggered.
2. The 20 lines of code after the line where the generation is triggered.
3. Up to 200 lines from the beginning of the current file where generation is triggered.

- Up to 5 code snippets that are similar to the code that surrounds the line where generation was triggered. These snippets are taken from the last 10 files that you opened in the same programming language as the current file.

To improve the results of in-editor code generation:

- Open files in your workspace that have code you want watsonx Code Assistant to know about. Watsonx Code Assistant tries to find code in these files that is similar to where you trigger completion, for example, methods that use similar classes and variables, and then sends this code to the model as context.
- Help watsonx Code Assistant by starting with a class definition or adding import statements, or use the watsonx Code Assistant chat to create the initial outline, copy the result into a file in your workspace, and then continue with in-editor code generation. Don't use in-editor code generation when you start from zero or almost empty files, as these conditions do not provide enough context.

Explaining code

watsonx Code Assistant

Use generative AI to analyze and summarize your code to understand what the code does.

The following table lists the type of explanation for each plan.

Plan	Explanation	Description
Trial plan	Basic	Uses generative AI to provide a basic explanation. No extra code analysis is required.
Essentials plan		
Standard plan	Enhanced	Requires a built application, and uses a code analysis and generative AI to provide an enhanced code explanation for Java methods/classes and enterprise Java applications.

Explanation types



Note: Eclipse IDE only: For best results with a multimodule Java application, ensure that you import the aggregator module and each submodule as separate projects within the Eclipse workspace. Watsonx Code Assistant is unlikely to successfully introspect your project if you import the top-level directory as a single project.

Language support

Code explanation is available for the following languages:

- C
- C++
- Go
- Java
- JavaScript
- Python
- TypeScript

When you reference a method, or use CodeLens on a method, watsonx Code Assistant supports certain code languages. Referencing a full file works for all languages. For more information, see [Language support when you work with methods](#).

Using a chat command to explain code

You can use the `/explain` command in chat to explain code for a referenced class, file, function, or method in the active workspace.

Use this syntax:

```
/explain <code reference> [additional instructions]
```

- Make sure to start the prompt with `/explain`, followed by the rest of the syntax.
- For `<code reference>`, type the `@` symbol to see a list of classes, files, functions, and methods from your workspace. Use one class, file, function, or method reference at a time.
- The `[additional instructions]` are optional. Add instructions if you want specific details.

When you reference a file, the size limit is 50 KB. If you reach this limit, split the file into individual functions and reference each function. Or, split the file at 49 KB, taking care of function boundaries, and reference the file at each split. With either approach, you need to merge the results.

Using the CodeLens in the editor to explain code

In the IDE editor, the CodeLens shows a line of generative AI options that precedes code blocks and snippets.

Note: Known issue If you are using the watsonx Code Assistant for Enterprise Java Applications extension, upon startup it might take a few seconds for the enhanced Java capabilities to be available from the CodeLens.

1. Click the **Explain** option that immediately precedes a code block to generate an explanation.

In the following code example, the `Explain | Document | Unit Test` options immediately precede the `protected void` keywords.

CodeLens example

```
87 | Explain | Document | Unit Test
    | protected void doPost(HttpServletRequest request, HttpServletResponse response)
88 |     | throws ServletException, IOException {
89 |     |
90 |     |     doGet(request, response);
91 |     | }
92 |
93 | Explain | Document | Unit Test
    | protected int exportReverervations(String selectedDateStr) {
94 |     | File fileToZip = IOUtils.getFileFromRelativePath("reservations.json");
95 |     | String userDirectory = System.getProperty("user.home");
96 |     | String zipPath = userDirectory + "/reservations.zip";
97 |     |
98 |     | FileOutputStream fos;
99 |     | try {
100 |    |     | fos = new FileOutputStream(zipPath);
101 |    |     | ZipOutputStream zipOut = new ZipOutputStream(fos);
102 |    |     |
103 |    |     | FileInputStream fis = new FileInputStream(fileToZip);
104 |    |     | ZipEntry zipEntry = new ZipEntry(fileToZip.getName());
105 |    |     | zipOut.putNextEntry(zipEntry);
```

2. The watsonx Code Assistant chat window opens, displays the `/explain @<item name>` command, runs the command, and displays the explanation.

Disabling CodeLens

If you want to disable the CodeLens options, you can change the setting for the extension or plug-in.

In Visual Studio Code:

1. Open the settings for the extension.
2. Clear the `Enable CodeLens` setting.

In Eclipse:

1. Open the settings for the Eclipse IDE.
2. In the **watsonx Code Assistant Settings** entry, clear the `Enable CodeLens` setting.
3. Click **Apply and close**.

Using the explorer to explain code

To generate an explanation from the Explorer (Visual Studio Code) or Project Explorer (Eclipse):

1. Expand your application to the code for which you want to generate an explanation.
2. Right-click the code, click **watsonx Code Assistant**, then click **Explain**.
3. The watsonx Code Assistant chat displays the `/explain` command for the code that you selected. The following syntax is used in the command:

```
/explain <code syntax>
```

4. Watsonx Code Assistant processes the request and in the chat displays the explanation for the code that you selected.

Explaining Java applications

Standard plan Application explanation is only available with the watsonx Code Assistant for Enterprise Java Applications extension.

Note: Application explanation is only supported for Java applications that contain one or more classes that extend `javax.servlet.http.HttpServlet`. These classes need to implement any one of the following methods: `doGet`, `doPost`, or `doPut`.

Note: For large applications, the explanation considers the 40 entry points with the [highest cyclomatic complexity](#). If the application exceeds 40 entry points, the explanation contains the section `The following methods were not considered:` that provides details about which entry points were not considered.

To request and view an explanation for an application:

1. In your IDE, right-click any item in the hierarchy in the directory of the application that you want to explain, and then click **Explain Application**.

Note: In the Eclipse IDE, if you have a multimodule application with a peer multimodule root, the application explanation applies only to the selected module.

2. Watsonx Code Assistant scans the application to generate an overview and a list of main services with a description of functions for each method.
3. Click **Save** to retain a copy, or the explanation is discarded.
4. Click **Open explanation** and review, and then you can click **Save application explanation** to store in a local file.

Documenting code

watsonx Code Assistant

Generate comment lines that document what your code does.

Language support and standards

The following table lists code languages and the corresponding documentation standard for each code language.

Code	Documentation standard
C	doxygen
C++	doxygen
Go	GoDoc
Java	JavaDoc
JavaScript	JSDoc
TypeScript	JSDoc
All others	Markdown

Documentation standards

When you reference a method, or use CodeLens on a method, watsonx Code Assistant supports certain code languages. Referencing a full file works for all languages. For more information, see [Language support when you work with methods](#).

Using a chat command to generate documentation

You can use the `/document` command in chat to generate documentation for a referenced file or a method or a class in the active workspace.

Use this syntax:

```
/document <code reference> [additional instructions]
```

- Make sure to start the prompt with `/document`, followed by the rest of the syntax.
- For `<code reference>`, type the `@` symbol to see a list of files, classes, and methods from your workspace. Use one file, method, class reference at a time.
- The `[additional instructions]` are optional. Add instructions if you want specific details.

When you reference a file, the size limit is 50 KB. If you reach this limit, split the file into individual functions and reference each function. Or, split the file at 49 KB, taking care of function boundaries, and reference the file at each split. With either approach, you need to merge the results.

Using the CodeLens in the editor to generate documentation

In the IDE editor, the CodeLens shows a line of generative AI options that precedes code blocks and snippets.

1. Click the **Document** option that immediately precedes a code block or snippet to generate its documentation.

In the following code example, the `Explain | Document | Unit Test` options immediately precede the `protected void` keywords.

CodeLens example

```

87 Explain | Document | Unit Test
88 protected void doPost(HttpServletRequest request, HttpServletResponse response)
89     throws ServletException, IOException {
90     doGet(request, response);
91 }
92
93 Explain | Document | Unit Test
94 protected int exportReverervations(String selectedDateStr) {
95     File fileToZip = IOUtils.getFileFromRelativePath("reservations.json");
96     String userDirectory = System.getProperty("user.home");
97     String zipPath = userDirectory + "/reservations.zip";
98
99     FileOutputStream fos;
100    try {
101        fos = new FileOutputStream(zipPath);
102        ZipOutputStream zipOut = new ZipOutputStream(fos);
103
104        FileInputStream fis = new FileInputStream(fileToZip);
105        ZipEntry zipEntry = new ZipEntry(fileToZip.getName());
106        zipOut.putNextEntry(zipEntry);

```

2. The watsonx Code Assistant chat window opens, displays the `/document @<item name*>` command, runs the command, and displays the code with documentation added as comments.

Disabling CodeLens

If you want to disable the CodeLens options, you can change the setting for the extension or plug-in.

In Visual Studio Code:

1. Open the settings for the extension.
2. Clear the `Enable CodeLens` setting.

In Eclipse:

1. Open the settings for the Eclipse IDE.
2. In the **watsonx Code Assistant Settings** entry, clear the `Enable CodeLens` setting.
3. Click **Apply and close**.

Generating unit tests

watsonx Code Assistant

Generate unit tests for various programming languages from the chat, the CodeLens in the editor, or the explorer.

The following table lists the type of unit test for each plan.

Plan	Unit test	Description
<p>Trial plan</p> <p>Essentials plan</p>	Basic	Uses generative AI to provide a basic unit test. No extra code analysis is required.
Standard plan	Enhanced	Requires a built application, and uses a code analysis and generative AI to provide an enhanced unit test for Java methods and classes.

Unit test types

Note: Eclipse IDE only: For best results with a multimodule Java application, ensure that you import the aggregator module and each submodule as separate projects within the Eclipse workspace. Watsonx Code Assistant is unlikely to successfully introspect your project if you import the top-level directory as a single project.

Language support

Test generation is available for the following languages:

- C
- C++
- Go
- Java
- JavaScript
- Python
- TypeScript

When you reference a method, or use CodeLens on a method, watsonx Code Assistant supports certain code languages. Referencing a full file works for all languages. For more information, see [Language support when you work with methods](#).

Dependencies

Standard plan For Java, make sure that the `pom.xml` file includes the following dependencies:

- JUnit - A minimum of JUnit 5 is required
- Mockito - A minimum of Java 9 is required

The following sample `pom.xml` file shows the dependencies with example versions. In your file, include the dependencies and versions that you are using.

```
$ <dependencies>
  <dependency>
    <groupId>org.junit.jupiter</groupId>
    <artifactId>junit-jupiter-api</artifactId>
    <version>5.10.0</version>
    <scope>test</scope>
  </dependency>
  <dependency>
    <groupId>org.mockito</groupId>
    <artifactId>mockito-core</artifactId>
    <version>5.12.0</version>
    <scope>test</scope>
  </dependency>
  <dependency>
    <groupId>org.mockito</groupId>
    <artifactId>mockito-junit-jupiter</artifactId>
    <version>5.12.0</version>
    <scope>test</scope>
  </dependency>
</dependencies>
```

Using a chat command to generate a unit test

You can use the `/unit-test` command in chat to generate a test for a referenced class, file, function, or method in the active workspace.

Use this syntax:

```
/unit-test [using <test_framework>] <code reference> [similar to (<unit test reference>)] [additional instructions]
```

- Make sure to start the prompt with `/unit-test`, followed by the rest of the syntax.
- `[using <test_framework>]` is optional. If you don't specify the framework, watsonx Code Assistant uses the test framework from your settings.
- For `<code reference>`, type the `@` symbol to see a list of classes, files, functions, and methods from your workspace. Use one class, file, function, or method reference at a time.

When you reference a file, the size limit is 50 KB. If you reach this limit, split the file into individual functions and reference each function. Or, split the file at 49 KB, taking care of function boundaries, and reference the file at each split. With either approach, you need to merge the results.

- `[similar to (<unit test reference>)]` is optional. Use to provide an example of how the generated unit test is expected to look. You can use a method reference in the `similar to` option if required.
- The `[additional instructions]` are optional. Add instructions if you want specific details.

Next, watsonx Code Assistant processes the request and in the chat displays the generated unit test for the code that you selected.

- **Essentials plan** Copy the unit test to a file where you have unit tests for your application.
- **Standard plan** Copy the unit test to a file in the high-level directory structure `src/test/java`. For example, copy the unit test file to `.../src/test/java/com/acme/modres/WCA_OrdersAlertFilter_Test.java`.

Using the CodeLens in the editor to generate unit tests

In the IDE editor, the CodeLens shows a line of generative AI options that precedes code blocks and snippets.

Note: Known issue If you are using the watsonx Code Assistant for Enterprise Java Applications extension, upon startup it might take a few seconds for the enhanced Java capabilities to be available from the CodeLens.

1. Click the **Unit Test** option that immediately precedes a code block to generate a unit test.

In the following code example, the `Explain | Document | Unit Test` options immediately precede the `protected void` keywords.

CodeLens example

```
00 | Explain | Document | Unit Test
87 | protected void doPost(HttpServletRequest request, HttpServletResponse response)
88 |     throws ServletException, IOException {
89 |
90 |     doGet(request, response);
91 | }
92 |
93 | Explain | Document | Unit Test
94 | protected int exportReservations(String selectedDateStr) {
95 |     File fileToZip = IOUtils.getFileFromRelativePath("reservations.json");
96 |     String userDirectory = System.getProperty("user.home");
97 |     String zipPath = userDirectory + "/reservations.zip";
98 |
99 |     FileOutputStream fos;
100 |     try {
101 |         fos = new FileOutputStream(zipPath);
102 |         ZipOutputStream zipOut = new ZipOutputStream(fos);
103 |
104 |         FileInputStream fis = new FileInputStream(fileToZip);
105 |         ZipEntry zipEntry = new ZipEntry(fileToZip.getName());
106 |         zipOut.putNextEntry(zipEntry);
```

The watsonx Code Assistant chat window opens, displays the `/unit-test @<item name*>` command, runs the command, and displays the unit test.

2. **Essentials plan** Copy the unit test to a file where you have unit tests for your application.
3. **Standard plan** Copy the unit test to a file in the high-level directory structure `src/test/java`. For example, copy the unit test file to `.../src/test/java/com/acme/modres/WCA_OrdersAlertFilter_Test.java`.

Disabling CodeLens

If you want to disable the CodeLens options, you can change the setting for the extension or plug-in.

In Visual Studio Code:

1. Open the settings for the extension.
2. Clear the `Enable CodeLens` setting.

In Eclipse:

1. Open the settings for the Eclipse IDE.
2. In the **watsonx Code Assistant Settings** entry, clear the `Enable CodeLens` setting.
3. Click **Apply and close**.

Using the explorer to generate unit tests

To generate a unit test from the Explorer (Visual Studio Code) or Project Explorer (Eclipse):

1. Expand your application to the code for which you want to generate a unit test.
2. Right-click the code, click **watsonx Code Assistant**, then click **Unit Test**.
3. The watsonx Code Assistant chat displays the `/unit-test` command for the code that you selected for unit testing. The following syntax is used in the command:

```
/unit-test @<*item syntax*>
```

The following syntax examples are for a file, a class, and a method.

- File: `/unit-test @OrdersAlertFilter.java`
- Class: `/unit-test @OrdersAlertFilter`
- Method: `/unit-test @DoFilter()`

Watsonx Code Assistant processes the request and in the chat displays the generated unit test for the code that you selected.

4. **Essentials plan** Copy the unit test to a file where you have unit tests for your application.
5. **Standard plan** Copy the unit test to a file in the high-level directory structure `src/test/java`. For example, copy the unit test file to `.../src/test/java/com/acme/modres/WCA_OrdersAlertFilter_Test.java`.

Fixing unit tests

Standard plan You can start a unit test fix for tests generated by watsonx Code Assistant. Fixing unit tests is only available with the watsonx Code Assistant for Enterprise Java Applications Visual Studio Code extension or Eclipse plug-in, version 1.1.0 and later.



Note: If you are using watsonx Code Assistant on-premises as a service of IBM Software Hub, version 5.2 or later is required.

Requirements:

- The Fix Unit Test feature is available for tests generated by watsonx Code Assistant that start with the prefix `WCA_`. Don't remove the `WCA_` prefix from the generated tests.
- The unit tests need to be stored in the `src/test/java` folder of a Java project or application.

To fix a unit test:

1. Open the Explorer view.
2. In the test folder, right-click an existing test file, click **watsonx Code Assistant**, and then click **Fix unit test**.

The process runs in two phases:

- **Compile fix phase:** Updates the file in the editor with clean compiled code
- **Runtime fix phase:** Checks the unit test for runtime errors



Note: If failures occur during the process, the code is commented out. For example, if compilation errors are fixed but the resulting code then has runtime errors.

3. When the two phases are complete, the unit test file is updated and can be run.

Keep the following points in mind:

- If a reference with a fully qualified class name or a full qualified method is missing from the generated test case, the unit test fix is canceled.
- Your application must be at Java version 9 or later to fix a unit test.
- Your application must be analyzed before the unit test can be fixed.
- Test generation sometimes generates unnecessary imported classes, interfaces, or packages. If the imports cause compile or runtime errors, you can comment them out.

Translating code from one language to another

watsonx Code Assistant

Use the `/translate` command in the chat window to translate code from one language to the other.

Chat syntax for translating code

The chat syntax is: `/translate [from <source_language>] to <target_language> (<code reference> | <code snippet>) [additional instructions]`

- Make sure to start the prompt with `/translate`, followed by the rest of the syntax.
- `[from <source_language>]` is optional and is automatically detected if not specified in the command. It is recommended to specify the source language if the quality of the translation response is not good enough.
- `<target_language>` is required. Provide complete language names without spaces, such as `TypeScript` instead of Type Script or `JavaScript` instead of Java Script.
- `(<code reference> | <code snippet>)`: Along with a target language, you can include either a code reference or a code snippet. Use one reference or snippet at a time.
 - `<code reference>`: Refer to a file, method, or class name from active workspace by using an `@` symbol. For example:

```
$ /translate to java @get_metric_type
```

When you reference a method, watsonx Code Assistant supports certain code languages. Referencing a full file works for all languages. For more information, see [Language support when you work with methods](#).

When you reference a file, the size limit is 50 KB. If you reach this limit, split the file into individual functions and reference each function. Or, split the file at 49 KB, taking care of function boundaries, and reference the file at each split. With either approach, you need to merge the results.

- `<code snippet>`: Use a code snippet from your editor or paste a code snippet.

To use code snippet from your editor:

1. Highlight the code in the editor.
2. Right click to show the pop-up menu.
3. Choose **watsonx Code Assistant**.
4. Choose **Add selected code to chat** to insert the code into the chat where you are starting the `/translate` command.

To paste a code snippet:

1. Find the code you want to translate.
2. Paste the code into the chat where you are starting the `/translate` command.

- `[additional instructions]` are optional.

For example: `/translate to java @get_user_list Copy the comments from this function to the translated code`

Getting answers from IBM documentation

watsonx Code Assistant

Trial plan

Essentials plan

Standard plan

Use chat to ask questions that reference IBM and Red Hat product documentation.



Note: This feature requires an IBM Cloud Trial, Essentials, or Standard plan. It isn't available if you are using watsonx Code Assistant Individual to access a local IBM Granite model.

Chat syntax for documentation questions

The chat syntax is: `/docs [collection:<collection_name>] <your question>`

The `collection_name` is optional. If no collection is specified, `ibm_docs` is used.

Available documentation collections are:

Collection	Source	URL reference
<code>ibm_docs</code>	IBM Documentation	https://www.ibm.com/docs
<code>cloud_docs</code>	IBM Cloud Docs	https://cloud.ibm.com/docs/
<code>redhat</code>	Red Hat Documentation	https://docs.redhat.com/products
<code>ibm_redbooks</code>	IBM Redbooks publications	https://www.redbooks.ibm.com/
<code>support_docs</code>	IBM Support content	https://www.ibm.com/support/
<code>general_technical_docs</code>	IBM Research and SecurityIntelligence	https://research.ibm.com/ and https://securityintelligence.com/articles

Documentation collections

Examples

Type	Example question
General product information	<code>/docs what are the different IBM Cognos Analytics offerings for which IBM provides commands to?</code>
Perform a task	<code>/docs What is the command to back up a database in Db2? Enclose the command in code blocks.</code>
Troubleshoot an error code	<code>/docs collection:support_docs I am getting this error: Internal error occurred: SCAPI error: The value on row 1,029 is not a valid string while previewing a data asset using import node. Can you provide a workaround for this?</code>
Query a specific collection	<code>/docs collection:redhat What are Machine pools in Cluster administration nodes in Red Hat OpenShift dedicated?</code>

Examples

Tips

These tips can help improve your `/docs` questions.

Use the product name

Be sure to specify the product name. `How to back up a database` might retrieve results from Db2, but `How to back up a Db2 database` works better.

Version numbers

The source that watsonx Code Assistant uses contains documentation about the latest versions of IBM products. Questions about older versions might work if you specify the version number in your query.

Context

When watsonx Code Assistant answers a `/docs` question, it doesn't use any other context from your chat window or open files. The context is retrieved from IBM product documentation.

The response from a `/docs` question becomes part of your chat context. Your next chat question can use the documentation response as context to help guide watsonx Code Assistant.

For example, you can ask: `/docs what is the command to back up a Db2 database. Format the command in Markdown.`

When you get the response with the command, you can ask: `Create a script to back up the Db2 database REVENUE` and you should receive a response that uses the command syntax from the documentation.

Modernizing Java applications

watsonx Code Assistant

Standard plan

You can modernize your application to Liberty. To modernize your application, you first upload a migration bundle or build and analyze your application, and then fix your application issues.

Before you begin, [set up your enterprise Java environment](#).

Application server sources that you can choose from are:

- Apache Tomcat Application Server
- JBoss Application Server
- WebLogic Application Server
- WebSphere Application Server traditional V6.1
- WebSphere Application Server traditional V7.0
- WebSphere Application Server traditional V8.0
- WebSphere Application Server traditional V8.5.5
- WebSphere Application Server traditional V9.0
- Other Third-Party Application Servers



Note: Eclipse IDE only: For best results with a multimodule Java application, ensure that you import the aggregator module and each submodule as separate projects within the Eclipse workspace. Watsonx Code Assistant is unlikely to successfully introspect your project if you import the top-level directory as a single project.

Complete either the steps to upload your application migration bundle or the steps to analyze your application. After you complete one set of steps, you can then fix issues with the migration. The steps to fix the issues are the same for both the migration bundle steps and application analysis. After you fix the issues, you have a modernized application.

Time required to analyze your code

The amount of time to analyze your application varies. The following times are typical.

Time	Application details
Seconds to a few minutes	Fewer than 100 classes
2 to 10 minutes	20,000 to 50,000 lines of code
10 or more minutes	300,000 lines of code

Time required to analyze



Important: When watsonx Code Assistant analyzes your application, it excludes some third-party packages from code analysis by default. For more information, see [Third-party packages that are excluded from application analysis](#).

Using a migration bundle to modernize your application

The following steps assume that you are using [IBM Cloud® Transformation Advisor](#) version 3.10 or later to generate a migration bundle. The migration bundle contains a list of modernization issues that need to be addressed. It also contains automatically generated configuration information to enable the application to function properly in Liberty.



Important: If you already uploaded and analyzed a migration bundle for your project, uploading a new migration bundle overwrites the existing analysis.



Note: Before you modernize your application, do a full build of your application to ensure that you are starting from a known good application

build. This build also ensures that class files are up to date with the latest source code, which is a requirement before you run automations that add changes for Java version upgrades.

If you revert the changes to your code, you must click **Return to upload / analyze** and do a full analysis if you want to modernize your application again.

1. In your IDE, right-click any item in the hierarchy in the directory of the application that you want to modernize.
2. Use either path to start the modernization process. Both are the same:
 - Click **watsonx Code Assistant**, then click **Modernize to Liberty**.
 - Click **Modernize Java Applications**, then click **Modernize to Liberty**.
3. On the Modernize <your application name> page, upload the migration bundle.

The IDE analyzes your project folder, gathers information about your application, and then uploads the migration bundle that Transformation Advisor generates.

- a. Click **Upload migration bundle**.
- b. Select the application bundle `.zip` file from the list and click **Open**.

When the the process to upload the migration bundle completes, the Modernize < your application name> page displays the following configuration files that are in the bundle:

- `server.xml` is the configuration file for Liberty and is required to start your application.
 - `Containerfile` is required only to build a container image for your Liberty application.
- c. Select the files to add to your project and then click **Proceed**.
 - (Required) Select the `server.xml` file.
 - (Optional) Select the `Containerfile` file only if you are building a container image for your Liberty application.

When the upload of the migration bundle completes, the issues to fix or review are displayed.

4. [Fix issues with the application.](#)

Depending on the issues to fix, the ability to fix them can be automated, assisted, or self-directed. The issues on the **Additional information** tab don't require a code change, but you might need to verify or test them and make a code change if you want.

Analyzing your application to modernize it


The following steps assume that you are analyzing your application instead of a using a Transformation Advisor migration bundle to modernize your application. The analysis provides you with a list of issues to address and a limited server configuration in the `server.xml` file. The `server.xml` file contains a list of features that your application uses, but it doesn't include other configuration information such as database connections or JMS configurations.

If you revert the changes to your code, you must click **Return to upload / analyze** and do a full analysis if you want to modernize your application again.

1. In your IDE, right-click any item in the hierarchy in the directory of the application that you want to modernize.
2. Use either path to start the modernization process. Both are the same:
 - Click **watsonx Code Assistant**, then click **Modernize to Liberty**.
 - Click **Modernize Java Applications**, then click **Modernize to Liberty**.
3. On your Modernize <your application name> page, expand **Analyze application**, and click **Start analysis**.
4. Select the analysis parameters, which define the rules that the application server is analyzed against.
 - a. Select the application server source and the application server target.
 - b. Optionally change the Java SE source, which is displayed automatically with a recommended value.

The recommended value is based on the Java SE version that watsonx Code Assistant recommends for the application server source. Your application server might be using a different Java SE version than what is recommended. If it is, then change the Java SE source to the version that your application server uses.

For example, watsonx Code Assistant recommends [Java SE 8 for WebSphere Application Server V8.5.5](#) even though your application server might be using an earlier version of Java SE.

 **Important:** The minimum-supported Java version is Java 5.

- c. Select the Java SE target.
- d. Click either **Analyze** or **Build and analyze**.

Application analysis is done on the application binary. The application must be built before analysis begins. You can either automatically or manually build the application.

- To automatically build and analyze the application, click **Build and analyze**.
- If you manually build the application, first build it and then click **Analyze**.

After your application is built and analyzed, the issues to fix or review are displayed.

When you click **Build and analyze**, the application is built at its current Java source version so it can be analyzed for Java upgrade issues. Make sure that the IDE has the Java developer kit of the Java source version that is configured so the IDE can successfully build the application.

5. [Fix issues with the application.](#)

Depending on the issues to fix, the ability to fix them can be automated, assisted, or self-directed. The issues on the **Additional information** tab don't require a code change, but you might need to verify or test them and make a code change if you want.

Why is a .class file included in an analysis report instead of the .java file?

When you run an analysis of a binary (JAR/WAR/EAR), the issues are always discovered in the class files. When the binary scanner is run outside of watsonx Code Assistant, it has no access to the source code, so the .class file is included in the analysis report. When you import a migration plan, or when you run an analysis inside watsonx Code Assistant, the reports show references to class files, but watsonx Code Assistant links the .class file to the .java file, and the .java files are shown in the watsonx Code Assistant UI.

Fixing issues and reviewing additional information for your application

The Modernize *<your application name>* page lists the following information:

- The application server source and the application server target
- The Java SE version source and the Java SE version target
- Links to a technology report, an inventory report, and an analysis report
- Tabs for issues to fix, which can be automated, assisted, or self-directed fixes, and additional information.

The target Java developer kit is required when you run automated fixes and when you build the application after the Java version target changes are made. Make sure that the target Java developer kit is configured for your IDE.

- The Eclipse IDE extension automatically tries to find the correct Java developer kit that matches the Java version target from the set of all configured Java developer kits in the IDE.
- In the Visual Studio Code IDE, you must manually change the Java developer kit configuration.

Completing automated fixes

If you have an **Automated fixes** tab with one or more fixes, select the tab.

Because watsonx Code Assistant completes the fixes for you, you do not need to make any code changes. The large language model (LLM) is not used for automated fixes.

1. Expand the Instructions and read them.
2. Optionally open the issue to review the description of the issue.
3. Click **Run automated fixes**.
 - The Automated fixes are fixed.
 - External issues remain in the list until you update the associated dependency.
4. When **Run automated fixes** completes, click **Rebuild and refresh** to update the list of issues.



Note: If the automated fixes fail to run and produce compilation errors in the console output, make sure to update the version of the `maven-compiler` plug-in to 3.13.0 or newer.

Completing assisted fixes

If you have an **Assisted fixes** tab with one or more fixes, select the tab.

Although you fix your code, watsonx Code Assistant assists you by providing suggestions in the chat on how to change your code.

The level of assistance available for the issues varies with the issue. The assisted fixes help you to fix the issue, and sometimes a code suggestion gets you all the way or most of the way. Other times, the code suggestion might be as simple as removing an old API to prevent a compilation failure, and no further action is required from you. It is important to follow steps for each assisted fix to understand the value and limitations of each assisted fix. In cases where a complete code suggestion is not provided, you can think of the assisted fix as a conversation starter in the chat. Continue to engage with the chat to build your understanding of the issue and its resolution.

For a list of issues associated with assisted fixes, see the [Assisted fixes reference](#).

To use assisted fixes:

1. Expand the Instructions and follow the instructions.
2. For each issue, expand it.
3. Click **Read more** to expand the issue.
4. Click the file that is listed in the issue.

The file opens in the editor.

5. Expand the issue to view the issue details.
6. If in the details you are directed to select a code snippet, select it.
7. Click **Help me** for assistance in fixing the issue.

If you are using Visual Studio Code, the following restrictions apply:

- Make sure that you select Java code in only one active editor.
- Make sure that the active editor in which you selected Java code is in a different editor group than the **Modernize to Liberty** or **Java Upgrade** view.

8. Follow the code suggestions in the chat so you modify the code to fix the issues.

Sometimes, the chat provides an informational response instead of a code suggestion.

9. As you fix issues, click **Rebuild and refresh** until all assisted fixes are fixed.

Completing self-directed fixes

If you have a **Self-directed fixes** tab with one or more fixes, select the tab.

You change the code and determine how to fix the code, but you can get help from the chat.

1. Expand the Instructions and follow the instructions.
2. Expand each issue.
3. Click **Read more** to expand the issue.
4. Click the file that is listed in the issue.

The file opens in the editor.

5. Click **Start chat**, which is displayed for each occurrence of the issue.

The chat window opens so you can get help for the occurrence. For example, you can ask the code assistant what the current code does. Or, you can ask if an alternative implementation exists for the problematic code.

6. As you fix issues, click **Rebuild and refresh** until all self-directed fixes are fixed.

Addressing additional information

If you have an **Additional information** tab with a number of one or more on it, select the tab.

The issues listed on the tab are determined the same way that issues on other tabs are determined. Although the additional information issues do not require a change to your code, review the issues so you can ensure that your application is working when your code is migrated.

The additional information might indicate issues, such as minor behavioral changes in an API or connectivity concerns in databases or user registries, and what to do about them.

For example, you have a database that your application connects to. From reading the issue for this situation, you make sure that the host and port configured for the application are valid and accessible when the application is migrated.

- Expand the instructions and follow them.
- Click each issue.
 - Click **Read more** to expand the issue.
 - Click the link on the file that is listed in the issue to review the code.

Working on external issues outside the current application

While you work on fixes, some might be categorized as external issue occurrences outside the current application. The source code for these issues are in specified archives that are not part of the current source application.

If the archive is a library that your organization owns:

1. You can open its associated source code as a separate application and use watsonx Code Assistant to remediate the issues there.
2. Then, you can update your current application to use the new remediated version.

If the archive is a library that your organization doesn't own:

1. You need to update to a version that does not contain this issue. If no such version exists, you might need to use an alternative technology.
2. You can start a conversation with watsonx Code Assistant for help with determining a replacement. To start, try asking `What is the library X used for?` or `Is it easy to replace?`. Continue the conversation to get more information for your specific usage of the library.

Upgrading the Java version for your application

watsonx Code Assistant

Standard plan

To upgrade your Java code, you first build and analyze your application, and then fix any issues that are identified. Depending on the issues, the ability to fix them can be automated, assisted, or self-directed.

Before you begin, [set up your enterprise Java environment](#).



Note: Eclipse IDE only: For best results with a multimodule Java application, ensure that you import the aggregator module and each submodule as separate projects within the Eclipse workspace. Watsonx Code Assistant is unlikely to successfully introspect your project if you import the top-level directory as a single project.

Time required to analyze your code

The amount of time to analyze your application varies. The following times are typical.

Time	Application details
Seconds to a few minutes	Fewer than 100 classes
2 to 10 minutes	20,000 to 50,000 lines of code
10 or more minutes	300,000 lines of code

Time required to analyze



Important: When watsonx Code Assistant analyzes your application, it excludes some third-party packages from code analysis by default. For more information, see [Third-party packages that are excluded from application analysis](#).

Analyzing your application to upgrade the Java version

The analysis provides you with a list of issues to address. After you fix the issues, the Java version for your application is upgraded.

If you revert the changes to your code, you must click **Return to analyze** and do a full analysis if you want to do the Java upgrade again.

1. In your IDE, right-click any item in the hierarchy in the directory of the application that you want to upgrade.

2. Use either path to start the upgrade process. Both are the same:


- Click **watsonx Code Assistant**, then click **Upgrade Java Version**.
- Click **Modernize Java Applications**, then click **Upgrade Java Version**.

3. On your Upgrade *<your application name>* page, select the analysis parameters, which define the rules that the application is analyzed against.

a. Optionally change the Java source, which is displayed automatically with a recommended value.

The Java source version is the Java version that you are upgrading from. Based on the Java version detected in the build configuration, watsonx Code Assistant recommends a Java version to use as the source. You can choose a different Java version from the list of supported Java versions if that is appropriate for your application. The build configuration is in the Maven `pom.xml` file.

The Java version that is recommended is either the Java version that is detected in the build configuration, or is the best available Java version given the Java version that was detected. For example, if Java 9, which is not supported, is specified in the build configuration, Java 8 is the recommended Java version. Java 8 is recommended since it is the closest supported Java version for the analysis, and since it is earlier than the detected Java version.

 **Important:** The minimum-supported Java version is Java 6.

b. Select the Java target.

The Java target version is the Java version that you are upgrading your Java application to. The target versions that you can select from are long-term (LTS) Java releases.


c. Click either **Analyze** or **Build and analyze**.

Application analysis is done on the application binary. The application must be built before analysis begins. You can either automatically or manually build the application.

- Click **Build and analyze** to automatically build and analyze the application.
- Click **Analyze** after you manually build the application.

After your application is built and analyzed, the issues to fix or review are displayed. Depending on the issues to fix, the ability to fix them can be automated, assisted, or self-directed. These issues to review don't require a code change to migrate your code. However, they might need to be verified or tested.

When you click **Build and analyze**, the application is built at its current Java source version, so it can be analyzed for Java upgrade issues. Make sure to configure the IDE with the Java developer kit of the Java source version, so the IDE can successfully build the application.

 **Note:** When building and analyzing, if you have issues with a multimodule project that includes a submodule with `liberty-assembly` packaging, run `mvn install` at the root level of the project and try again.

Why is a .class file included in an analysis report instead of the .java file?

When you run an analysis of a binary (JAR/WAR/EAR), the issues are always discovered in the class files. When the binary scanner is run outside of watsonx Code Assistant, it has no access to the source code, so the .class file is included in the analysis report. When you import a migration plan, or when you run an analysis inside watsonx Code Assistant, the reports show references to class files, but watsonx Code Assistant links the .class file to the .java file, and the .java files are shown in the watsonx Code Assistant UI.

Fixing issues and reviewing additional information for your application

The Upgrade *<your application name>* page lists the following information:

- The Java version source and the Java version target
- Links to a technology report, an inventory report, and an analysis report
- Tabs for issues to fix, which can be automated, assisted, or self-directed fixes, and for additional information

The target Java developer kit is required when you run automated fixes and when you build the application after the Java version target changes are made. Make sure that the target Java developer kit is configured for your IDE.


- The Eclipse IDE extension automatically tries to find the correct Java developer kit that matches the Java version target from the set of all configured Java developer kits in the IDE.
- In the Visual Studio Code IDE, you must manually change the Java developer kit configuration.


Completing automated fixes

If you have an **Automated fixes** tab with one or more fixes, select the tab.

Because watsonx Code Assistant completes the fixes for you, you do not need to make any code changes. The large language model (LLM) is not used for automated fixes.

1. Expand the Instructions and read them.
2. Optionally open the issue to review the description of the issue.
3. Click **Run automated fixes**.
 - The Automated fixes are fixed.
 - External issues remain in the list until you update the associated dependency.
4. When **Run automated fixes** completes, click **Rebuild and refresh** to update the list of issues.

 **Note:** If the automated fixes fail to run and produce compilation errors in the console output, make sure to update the version of the `maven-compiler` plug-in to 3.13.0 or newer.

 **Note:** When running automations, if you have issues with a multimodule project that includes a submodule with `liberty-assembly` packaging, run `mvn install` at the root level of the project and try again.

Completing assisted fixes

If you have an **Assisted fixes** tab with one or more fixes, select the tab.

Although you fix your code, watsonx Code Assistant assists you by providing suggestions in the chat on how to change your code.

The level of assistance available varies with the issue. The assisted fixes help you to fix the issue, and sometimes a code suggestion gets you all the way or most the way. Other times, the code suggestion might be as simple as removing an old API to prevent a compilation failure, and no further action is required from you. It is important to follow steps for each assisted fix to understand the value and limitations of each assisted fix. In cases where a complete code suggestion is not provided, you can think of the assisted fix as a conversation starter in the chat. Continue to engage with the chat to build your understanding of the issue and its resolution.

For a list of issues associated with assisted fixes, see the [Assisted fixes reference](#).

To use assisted fixes:

1. Expand the Instructions and follow the instructions.
2. For each issue, expand it.
3. Click **Read more** to expand the issue.
4. Click the file that is listed in the issue. The file opens in the editor.
5. Expand the issue to view the issue details.
6. If in the details you are directed to select a code snippet, select it.
7. Click **Help me** for assistance in fixing the issue.

If you are using Visual Studio Code, the following restrictions apply:

- Make sure that you select Java code in only one active editor.
 - Make sure that the active editor in which you selected Java code is in a different editor group than the **Modernize to Liberty** or **Java Upgrade** view.
8. Follow the code suggestions in the chat to modify the code and fix the issue.

Sometimes, the chat provides an informational response instead of a code suggestion.

9. As you fix issues, click **Rebuild and refresh** until all assisted fixes are fixed.

Completing self-directed fixes

If you have a **Self-directed fixes** tab with one or more fixes, select the tab.

You change the code and determine how to fix the code, but you can get help from the chat.

1. Expand the Instructions and follow the instructions.
2. Expand each issue.
3. Click **Read more** to expand the issue.
4. Click the file that is listed in the issue. The file opens in the editor.
5. Click **Start chat**, which is displayed for each occurrence of the issue. The chat window opens so you can get help for the occurrence. For example, you can ask the code assistant what the current code does. Or, you can ask if an alternative implementation exists for the problematic code.
6. As you fix issues, click **Rebuild and refresh** until all self-directed fixes are fixed.

Addressing additional information

If you have an **Additional information** tab with a number of one or more on it, select the tab.

The issues listed on the tab are determined the same way that issues on other tabs are determined. Although the additional information issues do not require a change to your code, review the issues so you can ensure that your application is working when your code is migrated.

The additional information might indicate issues, such as minor behavioral changes in an API and what to do about them.

- Expand the instructions and follow them.
- Click each issue.
 - Click **Read more** to expand the issue.
 - Click the link on the file that is listed in the issue to review the code.

Working on external issues outside the current application

While you work on fixes, some might be categorized as external issue occurrences outside the current application. The source code for these issues are in specified archives that are not part of the current source application.

If the archive is a library that your organization owns:

1. You can open its associated source code as a separate application and use watsonx Code Assistant to remediate the issues there.
2. Then, you can update your current application to use the new remediated version.

If the archive is a library that your organization doesn't own:

1. You need to update to a version that does not contain this issue. If no such version exists, you might need to use an alternative technology.
2. You can start a conversation with watsonx Code Assistant for help with determining a replacement. To start, try asking **What is the library X used for?** or **Is it easy to replace?**. Continue the conversation to get more information for your specific usage of the library.

Configuring watsonx Code Assistant for Z

Updating your Db2 database

watsonx Code Assistant for Z

You might need to change or update your Db2 connection in watsonx Code Assistant. For example, either your user ID or password changed, or you want to point to a different database for a different IBM Application Discovery and Delivery Intelligence (ADDI) project.

To update your Db2 connection, complete the following steps:

1. Go to the watsonx Code Assistant dashboard.
2. Click **Deployments** from the navigation menu.
3. Click **Spaces**, then click the space that you created.
4. Click **Assets** in your space, then click the connection to Db2 that you created for your deployment.
5. Update the connection information (for example, update your username and password).
6. Click **Test Connection**. If the test completes successfully, click **Create**.

Supporting more ADDI projects

watsonx Code Assistant for Z

Watsonx Code Assistant for Z can work with multiple IBM Application Discovery and Delivery Intelligence (ADDI) projects in your organization.

Each project requires a dedicated:

- Service ID
- IBM Db2 instance

If you want to create another ADDI project, follow the same instructions as in [Setting up watsonx Code Assistant for Z in IBM Cloud](#), but make sure that you create new:

- Service ID. Do not reuse the same Service ID for two ADDI projects.
- Db2 instance. Do not reuse the Db2 instance from your existing ADDI project.
- Deployment space. Do not reuse the deployment space that you created for your existing ADDI project.

Managing IAM access for watsonx Code Assistant for Z

watsonx Code Assistant for Z

Access to watsonx Code Assistant for Z for users in your account is controlled by IBM Cloud® Identity and Access Management (IAM). Every user that accesses IBM watsonx Code Assistant in your account must be assigned an access policy with an IAM role.

To add a user to your instance, you must assign the **Editor** access role for the user in the deployment space. With this role, the user has full access to view, modify, and create resources, but cannot create users or access policies.

Adding users

The following steps must be completed by the administrator for the instance.

1. In the IBM Cloud console, invite the user to the account for the instance.

You must provide an email address for the user that you want to invite. For more information, see [Inviting users in the console](#).

Specify the following details:

- **Service:** IBM watsonx Code Assistant
- **Resources:** All resources

When the user responds to this invitation, they are prompted to sign in to or create an IBM Cloud account. From there, they can [create an API key](#) to connect to the instance from their local Visual Studio Code environment.

2. Go to the [watsonx user interface](#).
3. Open the Navigation menu and click **Deployments**.

4. Go to the **Spaces** tab and click the **Code Assistant Models** space that you created during setup.

5. Select the **Manage** tab and click **Access Control**.

6. Click **Add collaborators** and select **Add user IDs** from the menu.

A list of user IDs that are associated with your account appears.

7. Click the checkbox for the user ID that you want to add and select **Editor** for the role.

8. Click **Add**.

9. After the user accepts the invitation that you sent in the first step of this procedure, they can sign in to IBM Cloud and [create an API key](#). The user must then provide this API key in the settings for the Open Z Editor Visual Studio Code extension in their local environment.

For more information, see [Setting up your watsonx Code Assistant for Z service in IBM Cloud](#).

For more information about IAM in IBM Cloud, see [Managing access to resources](#).

Understanding your responsibilities when you use watsonx Code Assistant

Learn about the management responsibilities and terms and conditions that you have when you use watsonx Code Assistant.

Review the following sections for the specific responsibilities for you and for IBM. For the overall terms of use, see [IBM Cloud Notices](#) and [IBM Cloud Terms of Use](#). For a high-level view of the service types in IBM Cloud and the breakdown of responsibilities between the client and IBM for each type, see [Shared responsibilities for IBM Cloud offerings](#).

Incident and operations management

Incident and operations management includes tasks such as monitoring, event management, high availability, problem determination, recovery, and full state backup and recovery.

Task	IBM responsibilities	Your responsibilities
IBM Db2	<ul style="list-style-type: none"> For paid plans, provide daily encrypted backups of the database. Daily backups are kept for 14 days. For Enterprise and Standard HADR plans, provide failover capability. 	<ul style="list-style-type: none"> Provide more manual backups if required. Provide end-of-backup and point-in-time restore operations in the Db2 web console UI.

Incident and operations management roles and responsibilities

Change management

Change management includes tasks such as deployment, configuration, upgrades, patching, configuration changes, and deletion.

Task	IBM responsibilities	Your responsibilities
watsonx Code Assistant Service	Provide updates for the watsonx Code Assistant service and base model.	Subscribe to change notifications for watsonx Code Assistant.
IBM Db2 as a Service: watsonx Code Assistant for Z	Verify that watsonx Code Assistant for Z works with the current version of IBM Db2 SaaS.	<ul style="list-style-type: none"> Provision a required plan level of IBM Db2 SaaS in your cloud account, and set up encryption of Db2 data and user management according to your policies. Apply required updates for IBM Db2 SaaS. Ensure network connectivity between your instance of IBM Db2 SaaS and watsonx Code Assistant. Rotate the username and password for Db2 according to your requirements. Update the Db2 connection in watsonx Code Assistant when the username and password change.
IBM Cloud Object Storage	<ul style="list-style-type: none"> Verify that watsonx Code Assistant works with the current version of IBM Cloud Object Storage. Create Object Storage buckets within your Object Storage instance. 	<ul style="list-style-type: none"> Provision an instance of Object Storage with one of the required plans in your IBM Cloud account, and set up data encryption and user management according to your policies. Grant watsonx Code Assistant access to your Object Storage instance to ensure that it can write and read from your Object Storage buckets. Delete Object Storage buckets that were created by watsonx Code Assistant after you delete your instance of watsonx Code Assistant.

Change management roles and responsibilities

Identity and access management

Identity and access management includes tasks such as authentication, authorization, access control policies, and approving, granting, and revoking access.

Task	IBM responsibilities	Your responsibilities
Cloud Identity and Access Management (IAM)	Provide the function to restrict access to resources through the IBM Cloud console and REST APIs.	Manage access to resources through Cloud Identity and Access Management (IAM).
Service authentication: Ansible	Ensure that only authenticated users have access to your instance of watsonx Code Assistant.	<ul style="list-style-type: none"> • Create a Service ID and API key, enter the API key into Red Hat Ansible Lightspeed, and safeguard your API key from unauthorized access. • Rotate and update your API keys according to your security policy requirements.
Service authentication: watsonx Code Assistant for Z	Ensure that only authenticated users have access to your instance of watsonx Code Assistant.	<ul style="list-style-type: none"> • Create a Service ID and API key, enter the API key into the IBM Open Editor for Z settings UI, and safeguard your API key from unauthorized access. • Rotate and update your API keys according to your security policy requirements.

IAM roles and responsibilities

For more information about identity and access management, see [Managing IAM access for watsonx Code Assistant](#).

Security and regulatory compliance

Security and regulatory compliance includes tasks such as security controls implementation and compliance certification.

Task	IBM responsibilities	Your responsibilities
General	<ul style="list-style-type: none"> • Maintain controls commensurate to various industry compliance standards. • Monitor, isolate, and recover instances. • Monitor and report the health of instances in the various interfaces. • Secure cluster access through TLS/SSH (data plane in the IBM Services account). • Integrate watsonx Code Assistant with Cloud Identity and Access Management (IAM). 	Set up and maintain security and regulation compliance for the watsonx Code Assistant instances.
Vulnerabilities	Monitor stock images to detect vulnerability and security compliance issues.	Keep informed about possible vulnerabilities and security issues through security bulletins that provide potential remediation actions. You can check the IBM Cloud status website to find announcements and security bulletins about key events that affect the IBM Cloud platform, infrastructure, and major services.
Encryption	Encrypt data when it is transmitted on any public networks and within the Cloud Service's private data center network. Encryption methods such as HTTPS, SSL, and TLS are used to protect data in motion.	Ensure, as required, that your connection is encrypted end-to-end.

Security and regulatory compliance roles and responsibilities

Disaster recovery

Disaster recovery includes tasks such as providing dependencies on disaster recovery sites, provisioning disaster recovery environments, data and configuration backup, replicating data and configuration, and fail over on disaster events.

Task	IBM responsibilities	Your responsibilities
General	<ul style="list-style-type: none"> Restore or rebuild the provisioning environments in the affected regions. Restore existing watsonx Code Assistant instances, where possible. 	<ul style="list-style-type: none"> Track instance state. Provision new watsonx Code Assistant instances in alternatively available regions.
Service recovery	Restore the watsonx Code Assistant service.	No action required.

Disaster recovery roles and responsibilities

Data security

Securing your data in watsonx Code Assistant

To ensure that you can securely manage your data when you use watsonx Code Assistant, it is important to know what data is stored and encrypted and how you can delete any stored data. Depending on your security requirements, you can encrypt data with customer-managed keys by integrating with IBM Cloud key management services. Such services include Key Protect, which supports the bring-your-own-key (BYOK) method, or Hyper Protect Crypto Services, which supports the keep-your-own-key (KYOK) method.

Securing your IDE extension setup

watsonx Code Assistant

Developers should follow instructions to secure the setup of the watsonx Code Assistant IDE extension.

For more information, see:

IDE	Instructions
Visual Studio Code	Securing your setup
Eclipse IDE	Securing your setup

Securing extension setup

Chat conversation storage

IBM watsonx Code Assistant stores all your chat conversation history locally in your file system in `<your home directory>/wca/chat.db`, in a database format defined by [SQLite](#). IBM watsonx Code Assistant does *not* share these conversations with anyone. This file is not encrypted, other than the encryption that your file system provides. Safeguard this file against improper access.

Telemetry data

IBM watsonx Code Assistant does *not* collect any telemetry data. In general, watsonx Code Assistant doesn't send any data that it processes to a third party, IBM included.

How your data is stored and encrypted in IBM watsonx Code Assistant

Watsonx Code Assistant stores customer-specific metadata, such as the connection asset for Db2, in a deployment space that the customer's cloud administrator creates. This space is reflected within IBM Cloud Object Storage as a folder in a Object Storage bucket. This bucket uses an IBM Cloud Object Storage instance that is owned by the customer. The customer can encrypt this data at rest by BYOK, or use the automatic encryption with keys that are provided by Object Storage. Watsonx Code Assistant accesses this data by using the customer-provided credentials. Watsonx Code Assistant has no way to access this data without these credentials.

Configuring IBM Cloud Object Storage

IBM Cloud Object Storage provides storage for projects, catalogs, and deployment spaces. You are required to associate an IBM Cloud Object Storage instance when you create projects, catalogs, or deployment spaces to store files for assets, such as uploaded data files or notebook files. The Lite plan instance is a no-cost option with storage capacity up to 25 GB per month.

You can also access data sources in an IBM Cloud Object Storage instance. To access IBM Cloud Object Storage, you create a Object Storage connection when you want to connect to data stored in Object Storage. A Object Storage connection has a different purpose from the Object Storage instance that you associate with a project, deployment space, or catalog. For more information, see [Getting started with IBM Cloud Object Storage](#).

The Cloud Identity and Access Management service securely authenticates users and controls access to IBM Cloud Object Storage. For instructions to set up access control for IBM Cloud Object Storage on IBM Cloud®, see [How do I invite a user to administer buckets and data?](#).

Encrypting data at rest

Red Hat Ansible Lightspeed

By default, data at rest is encrypted with randomly generated keys that IBM manages. If the default keys are sufficient protection for your data, no additional action is needed. To provide more protection for at rest data, you can create and manage your own keys with [IBM® Key Protect for IBM Cloud®](#), which is an encryption solution that securely stores data in IBM Cloud Object Storage.

For more information, see [Encrypting data with your own keys](#).

Encrypting data in motion

IBM encrypts data that is transmitted on any public networks and within the Cloud Service's private data center network. Encryption methods such as HTTPS, SSL, and TLS are used to protect data in motion.

Event logging

IBM watsonx Code Assistant services on IBM Cloud do not have event logging available.

Security and privacy for foundation models

Watsonx Code Assistant uses IBM Granite foundation models to help you code more quickly and more accurately by using natural language prompts. Your work with these foundation models is secure and private, in the same way that all your work on watsonx Code Assistant is secure and private.

The foundation models that you interact are hosted in IBM Cloud®. Your data is not sent to any third-party or open source platforms.

Any prompts that you create for are accessible only by you. Your prompt text is not accessible or used by IBM or any other person or organization.

Any data that you save is stored in a dedicated IBM Cloud Object Storage bucket that is associated with your project.

Data that is stored in your project storage bucket is encrypted at rest and in motion. You can delete your stored data at any time.

Privacy and security of saved work

Only people with Admin or Editor role access to the project or the project storage bucket can view saved assets. You control who can access your project and its associated Object Storage bucket.

For more information about asset security, see [Securing your data in watsonx Code Assistant](#).

For more information about managing project access, see [Setting up watsonx Code Assistant for Red Hat Ansible Lightspeed in IBM Cloud](#) or [Managing IAM access for watsonx Code Assistant for Z](#).

Ownership of your content and foundation model output

Content that you upload into watsonx Code Assistant is yours.

IBM does not use the content that you upload to or the output that is generated by a foundation model to further train or improve any IBM-developed models.

IBM does not claim to have any ownership rights to any foundation model outputs. You remain solely responsible for your content and the output of any foundation model.

Data privacy and IBM watsonx Code Assistant for Red Hat Ansible Lightspeed

Red Hat Ansible Lightspeed

When you request code recommendations from watsonx Code Assistant for Red Hat Ansible Lightspeed, the prompt data that you submit is transient, and IBM does not retain or use it for any other purpose.

What data does watsonx Code Assistant for Red Hat Ansible Lightspeed collect?

When you request a code recommendation, a prompt is sent to the service with the following information:

- The Ansible tasks that you are requesting a recommendation for
- Playbook context, that is, the playbook name and variables mentioned in that playbook

Once the data is submitted for processing to the model, the prompt data is discarded.

IBM does not:

- Use the input prompt to gather feedback about the model's performance.
- Use the input prompt to improve models.

Data privacy and IBM watsonx Code Assistant for Z

When you request code recommendations from watsonx Code Assistant for Z, the prompt data that you submit is transient and IBM does not retain or use it for any other purpose.

What data does watsonx Code Assistant for Z collect?

When you request a code recommendation, a prompt is sent to the service with the following information:

- The COBOL paragraph that you are requesting a recommendation for
- Metadata that provides content about your COBOL program

The following information clarifies how watsonx Code Assistant for Z handles the data that it collects in each of these cases.

The COBOL paragraph

Although the prompt includes the code snippet that you are requesting a recommendation for, IBM does not store the input prompt in any way. It is transient information that is available in the memory of the watsonx Code Assistant microservices only during the request.

IBM does not:

- Use the input prompt to gather feedback about the model performance.
- Use the input prompt to improve models.
- Log the input prompt in any observability tools, whether as part of an explicit log message or as part of exception messages.



Note: Exception -- Program names (COBOL Program ID) and paragraph names are stored when transforming.

The context data

Your prompt also includes metadata on the overall COBOL program, which helps watsonx Code Assistant for Z make more accurate recommendations in the context of your code. This metadata does not include the actual source code.

This metadata is stored in a Db2 instance on IBM Cloud, which is both managed and provided by you.

Only the specific IBM Cloud user and service IDs that you explicitly authorize have access to the Db2 instance that stores your metadata.

Watsonx Code Assistant does not cache the connection information for this Db2 instance in any way. The connection is retrieved specifically for each request, and kept in memory only during that particular request.

Code suggestion license reference

watsonx Code Assistant

Watsonx Code Assistant checks whether a generated code suggestion is similar to other code sources.

The generated code must be 10 or more lines in these languages:

- C
- C++
- Go
- Java
- JavaScript
- Python
- TypeScript

By default, watsonx Code Assistant blocks code suggestions that are similar to other code sources.

For some licenses, you can allow code suggestions that include a license reference. For more information, see [Configuring how code suggestions are displayed](#).

This table lists the licenses for which you can allow code suggestions:

Group	License
MIT Licenses	MIT No Attribution (mit-0)
MIT Licenses	Enlightenment License (e16) (MIT-advertising)
MIT Licenses	CMU License (MIT-CMU)
MIT Licenses	enna License (MIT-enna)
MIT Licenses	feh License (MIT-feh)
MIT Licenses	MIT License (MIT)
MIT Licenses	MIT +no-false-attribs license (MITNFA)
Apache Software Licenses	Apache License 1.0 (Apache-1.0)
Apache Software Licenses	Apache License 1.1 (Apache-1.1)
Apache Software Licenses	Apache License 2.0 (Apache-2.0)
Lesser General Public Licenses	GNU Lesser General Public License v2.1 only (LGPL-2.1)
Lesser General Public Licenses	GNU Lesser General Public License v3.0 only (LGPL-3.0)
General Public Licenses	GNU General Public License v2.0 only (GPL-2.0)
General Public Licenses	GNU General Public License v3.0 only (GPL-3.0)
General Public Licenses	GNU Affero General Public License v3.0 (AGPL-3.0)
Creative Commons	Creative Commons Attribution 1.0 Generic (CC-BY-1.0)
Creative Commons	Creative Commons Attribution 2.0 Generic (CC-BY-2.0)

Creative Commons	Creative Commons Attribution 3.0 Unported (CC-BY-3.0)
Creative Commons	Creative Commons Attribution 4.0 International (CC-BY-4.0)
Creative Commons	Creative Commons Zero v1.0 Universal (CC0-1.0)
Creative Commons	Creative Commons Attribution Share Alike 4.0 International (CC-BY-SA-4.0)
Berkeley Software Distribution	BSD 1-Clause License (BSD-1-Clause)
Berkeley Software Distribution	BSD 2-Clause "Simplified" License (BSD-2-Clause)
Berkeley Software Distribution	BSD 2-Clause FreeBSD License (BSD-2-Clause-FreeBSD)
Berkeley Software Distribution	BSD 2-Clause NetBSD License (BSD-2-Clause-NetBSD)
Berkeley Software Distribution	BSD-2-Clause Plus Patent License (BSD-2-Clause-Patent)
Berkeley Software Distribution	BSD 3-Clause "New" or "Revised" License (BSD-3-Clause)
Berkeley Software Distribution	BSD with attribution (BSD-3-Clause-Attribution)
Berkeley Software Distribution	BSD 3-Clause Clear License (BSD-3-Clause-Clear)
Berkeley Software Distribution	Lawrence Berkeley National Labs BSD variant license (BSD-3-Clause-LBNL)
Berkeley Software Distribution	BSD 3-Clause No Nuclear License (BSD-3-Clause-No-Nuclear-License)
Berkeley Software Distribution	BSD 3-Clause No Nuclear License 2014 (BSD-3-Clause-No-Nuclear-License-2014)
Berkeley Software Distribution	BSD 3-Clause No Nuclear Warranty (BSD-3-Clause-No-Nuclear-Warranty)
Berkeley Software Distribution	BSD 3-Clause Open MPI variant (BSD-3-Clause-Open-MPI)
Berkeley Software Distribution	BSD 4-Clause "Original" or "Old" License (BSD-4-Clause)
Berkeley Software Distribution	BSD-4-Clause (University of California-Specific) (BSD-4-Clause-UC)
Berkeley Software Distribution	BSD Source Code Attribution (BSD-Source-Code)
Berkeley Software Distribution	BSD Zero Clause License (0BSD)
Public Domain	The Unlicense (Unlicense)
Business Source Licenses	Boost Software License 1.0 (BSL-1.0)
Internet Systems Consortium	ISC License (ISC)

Licenses that can be allowed for code suggestions

This table lists the licenses for which code suggestions are always blocked. This setting can't be configured.

License

adobe-2006

adobe-glyph

adsl

amdplpa

aml

ampas

antlr-pd

apafml

artistic-2.0

barr

blueoak-1.0.0

boost software license 1.0

bzip2-1.0.6

cecill-2.1

cecill-b

cnri-python-gpl-compatible

condor-1.1

curl

do what the f*ck you want to public license

doc

ecl-1.0

ecl-2.0

eclipse public license 1.0

eclipse public license 2.0

educational community license v2.0

efl-2.0

egenix

epl-1.0

epl-2.0

european union public license 1.2

fair

fsfap

fsful

ftl

giftware

hpnd

ibm internal

ibm-pibs

icu

ijg

imagemagick

info-zip

intel

jasper-2.0

leptonica

libpng

libpng-2.0

libtiff

linux-openib

microsoft public license

miros

mozilla public license 2.0

mpich2

mpl-2.0

ms-pl

ms-rl

mtll

mulanpsl-1.0

mulanpsl-2.0

multics

mup

nasa-1.3

naumen

nca

net-snmp

netcdf

npl

nrl

ntp

ofl-1.1

oldap-2.2.1

oldap-2.3

oldap-2.4

oldap-2.5

oldap-2.6

oldap-2.7

oldap-2.8

oml

openssl

osl-3.0

other

php-3.0

php-3.01

plexus

postgresql

psf-2.0

python-2.0

qhull

ruby

sgi-b-2.0

spencer-86

spencer-94

swl

tcl

tcp-wrappers

tu-berlin-1.0

tu-berlin-2.0

upl-1.0

vim

vsl-1.0

w3c

w3c-19980720

w3c-20150513

wtfpl

x11

xerox

xfree86-1.1

xinetd

xnet

xpp

zed

zend-2.0

zlib

zlib-acknowledgement

zpl-1.1

zpl-2.0

zpl-2.1

Licenses that are always blocked

Third-party packages that are excluded from application analysis

watsonx Code Assistant

Standard plan

When watsonx Code Assistant analyzes your application, it excludes some third-party packages from code analysis by default.

Your applications are analyzed when you modernize or upgrade Java applications. For more information, see:

- [Modernizing Java applications](#)
- [Upgrading the Java version for your application](#)

The following packages are excluded:

- `ch.qos`
- `com.cloudant`
- `com.cognos`
- `com.fasterxml`
- `com.filenet`
- `com.ibatis`
- `com.ibm`
- `com.informix`
- `com.lowagie`
- `com.mchange`
- `com.meterware`
- `com.microsoft`
- `com.mongodb`
- `com.mysql`
- `com.sun`
- `com.sybase`
- `filenet`
- `freemarker`
- `groovy`
- `jakarta`
- `java`
- `javax`
- `net`
- `oracle`
- `org`
- `sqlj`
- `sun`
- `twitter4j`
- `_ibmjsp`

Assisted fixes reference

watsonx Code Assistant

Standard plan

When you modernize or upgrade your Java application, watsonx Code Assistant assists you by providing suggestions in the chat on how to change your code. This reference lists the code issues that are associated with assisted fixes.

For more information, see:

- [Completing assisted fixes when you modernize Java applications](#)
- [Completing assisted fixes when you upgrade Java applications](#)

Issues when you modernize WebSphere Application Server to Liberty

- The WebSphere ServletChain API was superseded by a newer implementation
- The WebSphere Servlet API was superseded by a newer implementation (match criteria: com.ibm.websphere.servlet.response.ResponseUtils)
- The WebSphere Startup Beans Service API was superseded by a newer implementation
- The WebSphere Servlet API was superseded by a newer implementation (ServletInputStreamAdapter)
- The WebSphere Servlet API was superseded by a newer implementation (ChainerServlet)
- The WebSphere Servlet API was superseded by a newer implementation (ServletOutputStreamAdapter)
- The WebSphere Servlet API was superseded by a newer implementation (ResponseErrorReport)
- Some WebSphere Security APIs and SPIs are unavailable (security Exception)
- Some WebSphere Security APIs and SPIs are unavailable (com.ibm.websphere.crypto)
- Review use of the dynamic cache service

Issues when you upgrade Java SE version

- Avoid using default charset when the output stream is a PrintStream
- The defineAnonymousClass method from sun.misc.Unsafe has been removed
- The netscape.javascript.JSObject.getWindow(Applet) method has been removed
- The sun.nio.cs.map system property was removed
- Behavior change in the javax.management.MBeanOperationInfo constructor
- The JavaFX modules were removed from JDK 11
- Do not use the removed javax.crypto.ExemptionMechanism.finalize() method
- Behavior change in java.util.concurrent.ThreadPoolExecutor.finalize() method
- Do not use removed runFinalizersOnExit methods
- Check for removed methods on Pack200.Packer and Pack200.Unpacker interfaces
- The java.util.jar.Pack200 APIs were removed
- Do not use APIs from the com.sun.image.codec.jpeg package
- The ObjectInputStream.GetField.get(name, object) method was changed to throw a ClassNotFoundException
- Weak named curves have been removed
- Do not use com.sun.org.apache JAXP internal classes
- Do not use com.sun.org.apache JAXP package names in string literals
- Do not use the removed com.sun.java.browser.plugin2.DOM and sun.plugin.dom.DOMObject APIs
- Do not override the Path2D getPathIterator methods
- Do not use the java.lang.Compiler class
- Avoid using the removed RMI Activation package
- Do not use the netdoc URL protocol
- The java.security.acl package was removed
- Do not use the com.sun.awt.SecurityWarning class
- Do not use the ThreadGroup.allowThreadSuspension method
- The java.awt.peer and java.awt.dnd.peer packages are not accessible
- Do not use the removed methods addPropertyChangeListener or removePropertyChangeListener from LogManager
- Avoid using the degraded ThreadGroup methods
- java.lang.Thread.stop(java.lang.Throwable) is disabled

- Avoid using `Thread.stop()`, `Thread.resume()`, and `Thread.suspend()` methods
- Check for configuration changes related to the `com.ibm.security.auth.module.Krb5LoginModule` class

Issues when you upgrade Java EE version

- Do not use OpenJPA strings for query hints or properties
- OpenJPA and WebSphere JPA configuration properties must be migrated
- Check for a behavior change in the `InjectionPoint` `getAnnotated` method
- Check for a behavior change regarding duplicate elements in web descriptors
- The `openjpa.jdbc.Schema` configuration property must be migrated to the mapping file
- Classes that use both the `Specializes` and `Alternative` annotations are not injected into other modules

Language support when you work with methods

watsonx Code Assistant

When you reference a method, or use CodeLens on a method, watsonx Code Assistant supports certain code languages. Referencing a full file works for all languages.

The following languages are supported when you work with methods:

- Bash
- C
- C++
- C#
- Go
- Groovy
- Java
- JavaScript
- Kotlin
- PHP
- Python
- Ruby
- Rust
- SQL
- TSX
- TypeScript

IBM watsonx Code Assistant model details

watsonx Code Assistant

IBM watsonx Code Assistant is designed to accelerate the software development lifecycle and is built on IBM Granite code models.

General Programming

The public data sources used to train the models include:

- GitHub
- StarCoder
- CommitPack
- Glaive-code-assistant

Java

Enhanced enterprise Java model for remediation, explanation, unit test generation and fix. This model takes advantage of new enterprise Java data tailored to the these use cases and a code jam by hundreds of IBM WebSphere developers.

The public data sources used to train the models include:

- methods2test
- Jakarta EE source code and documentation
- MicroProfile source code and documentation

Jakarta EE Specifications

Copyright © 2016-2023, Eclipse Foundation and Copyright © 2016-2024, Eclipse Foundation AISBL. This software or document includes material copied from or derived from the JakartaEE Specifications available from <https://jakarta.ee/specifications/>. This software may generate material that implements, is copied from or derived from the documents referenced herein. User is responsible for determining whether the requirements of the Eclipse Foundation Specification License (<https://www.eclipse.org/legal/efsl/>) apply to the generated materials.

MicroProfile Specifications and API documentation

Copyright © 2016-2023, Eclipse Foundation and Copyright © 2016-2024 Eclipse Foundation AISBL. This software or document includes material copied from or derived from Eclipse Foundation MicroProfile Specification documents and Eclipse Foundation MicroProfile API documentation available from <https://microprofile.io/specifications/>. This software may generate material that implements, is copied from or derived from the documents referenced herein. User is responsible for determining whether the requirements of the Eclipse Foundation Specification License (<https://www.eclipse.org/legal/efsl/>) apply to the generated materials.

Other (Such as natural language and math)

The public data sources used to train the models include:

- MathInstruct
- OpenWebMath
- S2ORC: The Semantic Scholar Open Research Corpus
- RedPajama 1T
- Stack Exchange
- Wikimedia

IBM watsonx Code Assistant for Red Hat Ansible Lightspeed model details

Red Hat Ansible Lightspeed

Watsonx Code Assistant for Red Hat Ansible Lightspeed base models are derived from the [IBM Granite models](#) and were trained on Ansible YAML data to improve their performance in Ansible tasks.

The following sources were used for training the base model on Ansible code:

- Ansible Galaxy (Ansible Collections)
- Ansible YAMLS from public repositories in GitLab and GitHub

The following sources were used for training the base model on the English language:

- Ansible documentation
- StackExchange
- Wikipedia
- CommonCrawl

IBM watsonx Code Assistant for Red Hat Ansible Lightspeed pricing plans

Red Hat Ansible Lightspeed

Watsonx Code Assistant for Red Hat Ansible Lightspeed is available on IBM Cloud in three different pricing plans, each of which offers different features and is subject to different resource constraints. All three plans require that you have entitlement to the [Red Hat Ansible Automation Platform](#), which is sold by Red Hat.

When you provision an instance of watsonx Code Assistant for Red Hat Ansible Lightspeed, you can choose from the following plans. Resource usage in these plans is allocated by code tokens. For each plan, you can track your token usage from the watsonx Code Assistant resource page in IBM Cloud.

Red Hat Ansible Lightspeed Trial plan

The Trial plan is a 90 day, no-cost trial of the capabilities that are available in the Essentials plan, with no resource limits. The plan includes task prompt capabilities and a preset limit of code tokens. To use the Trial plan, see [Activate Red Hat Ansible Lightspeed with IBM watsonx Code Assistant](#).

Red Hat Ansible Lightspeed Essentials plan

The Essentials plan is a pay-as-you-go plan with no preset limit that includes task prompt capabilities. With this plan, you can create task prompts from natural language requests to generate Ansible code.

Red Hat Ansible Lightspeed Standard plan

The Standard plan is a prepaid monthly plan. With this plan, you can create task prompts from natural language requests to generate Ansible code.



Note: The Standard plan isn't available in the Sydney (`au-syd`) data center. You need to choose either the Dallas (`us-south`) or Frankfurt (`eu-de`) data centers.

Provisioning a plan

Provision an instance of watsonx Code Assistant through the [IBM Cloud catalog](#) page or by working with an IBM sales representative.

Understanding IBM Cloud subscriptions and resource units for watsonx Code Assistant

watsonx Code Assistant

The watsonx Code Assistant service works on a subscription basis. Learn about the relationship among your subscription, resource units, and tokens.

In generative AI, tokens are discrete pieces of data that, for language or code, can be a word, a character, or even a phrase (or sequence of words or characters). The models that underlie generative AI solutions predict content based on an understanding of context (which are previous tokens). When you query a model, it uses a number of tokens to accurately generate a response to your query.

When you use the [IBM watsonx Code Assistant catalog page](#), what you're purchasing is entitlement to a number of resource units in your IBM Cloud subscription.

- One resource unit covers approximately 20 task prompts that use 500 tokens each
- A single developer requires approximately 15 resource units per month
- Use the [catalog page](#) to see what each plan offers and to estimate your costs



Important: Your IBM Cloud subscription credits don't expire and you can use them on any service instead of, or in addition to, watsonx Code Assistant. Resource units are tied to usage, not time. When you run out, you can resubscribe by renewing your contract.

Understanding IBM Cloud subscriptions and resource units for watsonx Code Assistant for Z

watsonx Code Assistant for Z

The watsonx Code Assistant cloud service works on a subscription basis. Learn about the relationship among your subscription, resource units, and tokens.

In generative AI, tokens are discrete pieces of data that, for language or code, can be a word, a character, or even a phrase (or sequence of words or characters). The models that underlie generative AI solutions predict content based on an understanding of context (which are previous tokens). When you query a model, it uses a number of tokens to accurately generate a response to your query.

When you use the [IBM watsonx Code Assistant for Z catalog page](#), what you're purchasing is entitlement to a number of resource units in your IBM Cloud subscription. Each resource unit is equal to 150,000 tokens. For watsonx Code Assistant for Z, COBOL applications require approximately 20-30 tokens per line of code. That value might vary depending on the application. Your tokens yield approximately 5000-7000 lines of code. That number can vary because the number of tokens that are required to generate a good response varies. When you run out of tokens, you can renew your contract to refill your credits.



Important: Your IBM Cloud subscription credits don't expire and you can use them on any service instead of, or in addition to, watsonx Code Assistant for Z. Resource units are tied to usage, not time. When you run out, you can resubscribe by renewing your contract.

Getting help and support for watsonx Code Assistant

If you experience an issue or have questions when you use watsonx Code Assistant, try 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 [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 `service-Name`, 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](#).

FAQ for watsonx Code Assistant

Frequently asked questions about watsonx Code Assistant.

What is watsonx Code Assistant?

IBM® watsonx™ Code Assistant is an innovative, generative AI coding companion that offers robust, contextually aware assistance for popular programming languages that include Go, C, C++, Java, JavaScript, Python, TypeScript, and more. Seamlessly integrated into your IDE, you can accelerate your productivity and simplify coding tasks, all with trust, security, and compliance.

How does watsonx Code Assistant integrate with my current development environment?

IBM watsonx Code Assistant seamlessly integrates with popular development environments like Visual Studio Code and Eclipse, so you can use your existing workflow.

For more information, see the documentation for your IDE:

- [Installing the extension for Visual Studio Code](#)
- [Installing the plug-in for the Eclipse IDE](#)

Standard plan

For use with the Standard plan, you can configure retrieval augmented generation (RAG), which is the process of optimizing the large language model (LLM) output through the prompt augmentation with additional context. If you submit a query, watsonx Code Assistant uses the RAG tools to retrieve the information from your code bases or documentation. This relevant context is appended to the query before it is sent to the LLM model. The RAG system determines the sources that need to be included or excluded to generate a response with the most useful information. For more information, see [Setting up retrieval augmented generation \(RAG\)](#).

What coding tasks can I do with watsonx Code Assistant?

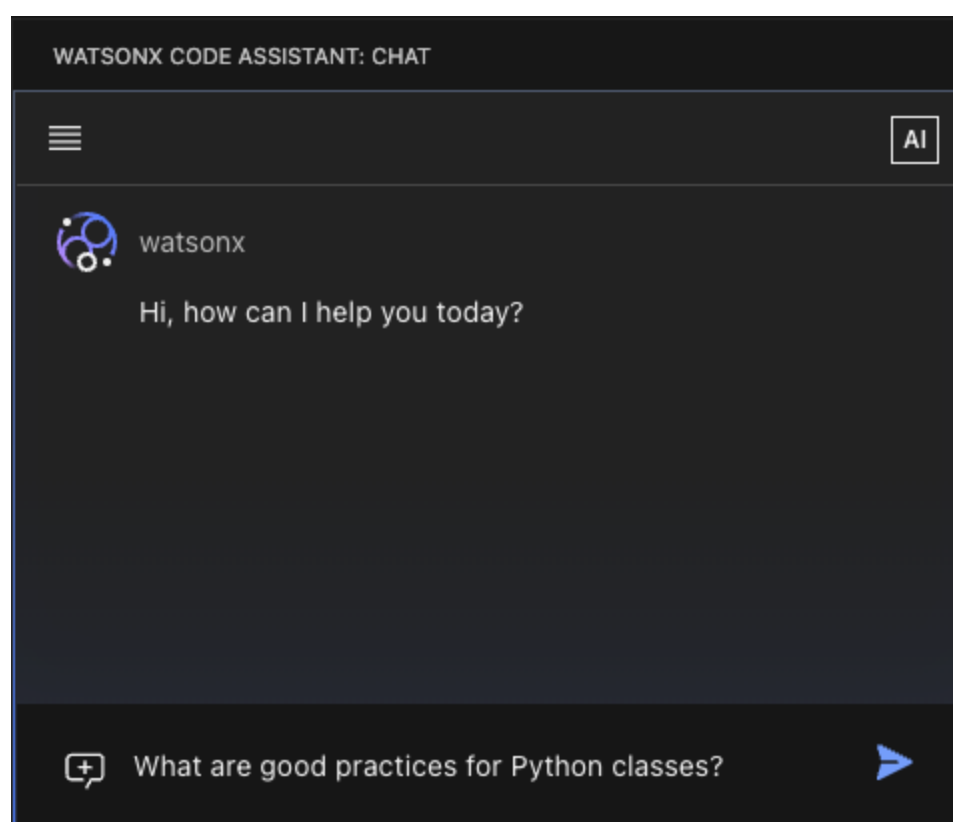
You can use watsonx Code Assistant to:

- [Get code suggestions](#)
- [Explain code](#)
- [Document code](#)
- [Generate unit tests](#)
- [Translate code from one language to another](#)
- [Get answers from IBM documentation](#)

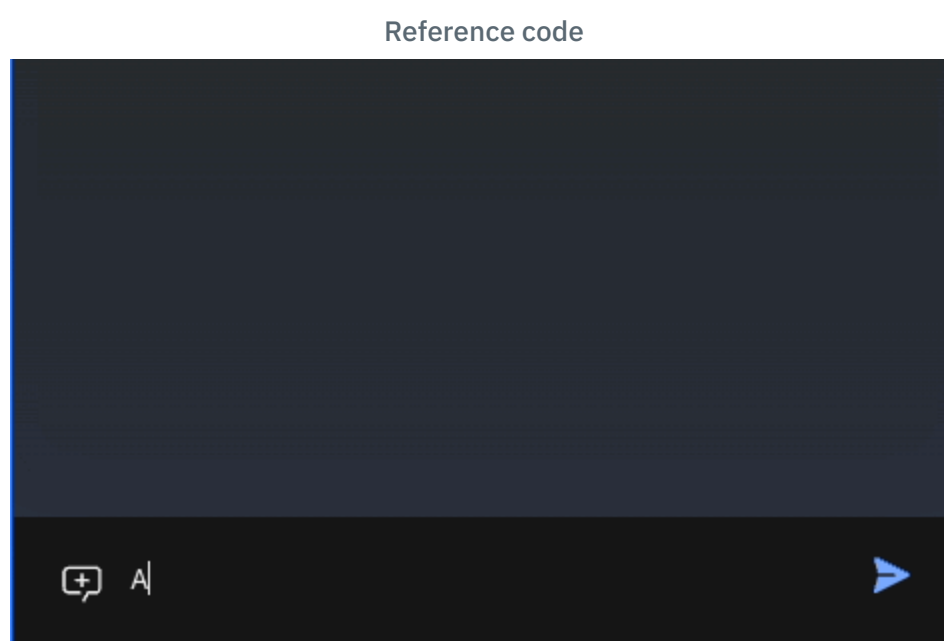
How do I get code suggestions?

Use chat conversations: Use natural language prompts to generate code suggestions. Use a chat conversation to enter a prompt that explains the code you need, and watsonx Code Assistant generates something you can choose to use.

Use chat conversations



Reference code: To ask questions or refine a specific file, class, function, or method in your workspace, you can use a code reference. These references provide important context and can help to increase the accuracy of the answer. As part of your chat message, type the @ symbol to see a list of files, classes, and methods from your workspace. Click to select the reference, and watsonx Code Assistant sends the contents of the reference as part of your message.



Code completion: Or, complete code in the editor. Start typing a line of code, then pause. IBM watsonx Code Assistant adds a code suggestion to complete the line that you typed.

Code completion

```
public class Customer {  
    private String firstName;  
    private String lastName;  
    private String email;  
    private String phoneNumber;  
    private int age;  
    ..  
    ..  
    ..  
}
```

A small yellow lightbulb icon is positioned at the end of the code block, indicating a code suggestion.

You can also get a multiline code suggestion. Start typing a line of code, then use a keyboard shortcut, and watsonx Code Assistant adds a multiline code suggestion.

Multiline code completion

```
J Customer.java 9+ ●
3 import java.util.Comparator;
4 import java.util.List;
5 import java.util.Random;
6
7 public class Customer {
8
9     private String firstName;
10    private String lastName;
11    private String email;
12    private String phoneNumber;
13    private int age;
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
```

Ln 15, Col 3 Spaces: 2 UTF-8 LF {} Java 1.22 of

Or, enter a comment that describes the code you want.

Comment-to-code generation

```

1
2 //
3
4
```

Ln 61, Col 5 Spaces: 2 UTF-8 LF {} Java 1.09 of

For more information, see the documentation for [Getting code suggestions](#).

How do I get a code explanation?

Use generative AI to analyze and summarize your code to understand what the code does. Click the **Explain** option that precedes a code block or enter `/explain` in a chat conversation. IBM watsonx Code Assistant analyzes the code and provides a detailed explanation of what the code does.

For more information, see the documentation for [Explaining code](#).

How do I add code documentation?

Generate comment lines that document what your code does. Click the **Document** option that precedes a code block or enter `/document` in a chat conversation. IBM watsonx Code Assistant analyzes the code and adds comments that document what the code does.

For more information, see the documentation for [Documenting code](#).

How do I generate unit tests?

Create unit tests to evaluate your code functions. Click the **Unit Test** option that precedes a code block or enter `/unit-test` in a chat conversation. IBM watsonx Code Assistant analyzes the code and creates a unit test.

For more information, see the documentation for [Generating unit tests](#).

How do I translate code from one language to another?

Use watsonx Code Assistant to translate code. In a chat conversation, use the syntax `translate [from <source language >] to <target_language> <code reference>`

For more information, see the documentation for [Translating code from one language to another](#).

How do I get answers from IBM documentation?

Use chat to ask questions that reference IBM and Red Hat product documentation.

The chat syntax is: `/docs [collection:<collection_name>] <your question>`

The `collection_name` is optional. If no collection is specified, `ibm_docs` is used.

Available documentation collections are:

Collection	Source	URL reference
<code>ibm_docs</code>	IBM Documentation	https://www.ibm.com/docs
<code>cloud_docs</code>	IBM Cloud Docs	https://cloud.ibm.com/docs/
<code>redhat</code>	Red Hat Documentation	https://docs.redhat.com/products
<code>ibm_redbooks</code>	IBM Redbooks publications	https://www.redbooks.ibm.com/
<code>support_docs</code>	IBM Support content	https://www.ibm.com/support/
<code>general_technical_docs</code>	IBM Research and SecurityIntelligence	https://research.ibm.com/ and https://securityintelligence.com/articles

Documentation collections

For more information, see the documentation for [Getting answers from IBM documentation](#)

What are the built-in commands that I can use when I write prompts?

Use these built-in commands to start or construct a prompt:

Command	Description	For more information
<code>@</code>	Reference code in your workspace	Referencing code in your workspace
<code>/document</code>	Generate documentation for a referenced file or a method or a class in the active workspace	Generating documentation
<code>/docs</code>	Ask questions that reference IBM and Red Hat product documentation	Getting answers from IBM documentation
<code>/explain</code>	Use generative AI to analyze and summarize your code to understand what the code does	Explaining code
<code>/help</code>	Get quick assistance about tasks	Type <code>/help</code> in the chat
<code>/unit-test</code>	Generate unit tests for various programming languages	Generating unit tests
<code>/translate</code>	Translate code from one language to the other	Translating code

Chat commands

Can I modernize or upgrade Enterprise Java applications?

Standard plan

For use with the Standard plan, IBM watsonx Code Assistant for Enterprise Java Applications adds features for Java application upgrades and modernization, and enhanced code explanation, unit test generation, and fixing Java unit tests.

Java runtime modernization

Analyze your Java application runtime and modernize it to a more lightweight, flexible, and efficient runtime. Receive a prescriptive plan that describes the changes that are needed to modernize your application, with a detailed assessment of complexity and required development effort. Use automation to quickly implement code and configuration changes. Transform code with generative AI assistance to resolve more complex issues.

For more information, see the documentation for [Modernizing Java applications](#).

Java version upgrade

Identify changes required to upgrade Java code, automatically apply fixes, and use generative AI to transform Java code. Receive a prescriptive plan that describes the changes that are needed to upgrade your application, with a detailed assessment of complexity and required development effort. Use automation to quickly implement code and configuration changes. Transform code with generative AI assistance to resolve more complex issues.

For more information, see the documentation for [Upgrading the Java version for your application](#).

Explain Java applications

With the Standard plan, you can generate an explanation of your Java application. For more information, see the documentation for [Explaining Java applications](#).

Generate and fix Java unit tests

Use generative AI to create tests that help maintain critical application functions. With the Standard plan, create enhanced unit tests to evaluate your code functions. You can also use generative AI to fix Java unit tests.

For more information, see the documentation for [Generating unit tests](#) and [Fixing unit tests](#).

How does watsonx Code Assistant differentiate between similar code contexts to avoid incorrect suggestions?

IBM watsonx Code Assistant uses context-aware generation, meaning it analyzes surrounding code, function names, parameters, and even previous edits to tailor suggestions. This minimizes out-of-context completions and improves relevance, especially in complex or multi-module applications.

How scalable is watsonx Code Assistant for larger projects or teams?

IBM watsonx Code Assistant is available in both on-premises and on SaaS models that provide you with the flexibility you need. It's highly scalable and suitable for individual developers and large teams that work on complex projects.

For more information, see the documentation:

- [SaaS: Setting up your watsonx Code Assistant service in IBM Cloud](#)
- [On-premises: watsonx Code Assistant on IBM Software Hub](#)

What security features does watsonx Code Assistant include to help ensure the safety and integrity of my code?

IBM watsonx Code Assistant is built on IBM Granite models and includes robust security features, such as IP indemnity, code similarity checks, and hardware-accelerated transparent memory encryption to protect your organization's intellectual property and help ensure code integrity.

Where are my chat conversations stored?

IBM watsonx Code Assistant stores all your chat conversations locally in your file system under `<your home directory>/wca/chat.db`, in a database format defined by [SQLite](#). IBM watsonx Code Assistant does *not* share these conversations with anyone. This file is not encrypted, other than the encryption that your file system provides. Safeguard this file against improper access.

Does watsonx Code Assistant collect telemetry data?

IBM watsonx Code Assistant does *not* collect any telemetry data. In general, watsonx Code Assistant does not send any data that it processes to a third party, IBM included.

What are the pricing plans, and what does each plan include?

IBM offers flexible pricing plans, including subscription-based options for different budgetary needs. Refer to the plan listings in the [IBM Cloud catalog](#).

Which programming languages are supported by watsonx Code Assistant?

IBM watsonx Code Assistant supports 116 programming languages, including Python, Java, C, C++, Go, JavaScript, and TypeScript.

How does watsonx Code Assistant support enterprise innovation at scale?

By removing routine friction in the development process, like writing boilerplate, onboarding new developers, and maintaining code quality, watsonx Code Assistant enables teams to focus more on business logic and experimentation.

How does watsonx Code Assistant maintain code consistency across a distributed development team?

IBM watsonx Code Assistant promotes consistency by generating code that is aligned with shared organizational patterns, such as naming conventions, architecture standards, and commenting styles. By grounding watsonx Code Assistant with your team's existing repositories and by using watsonx Code Assistant within IDEs integrated with version control systems, teams can reduce mistakes and rework during code reviews.

What data sources are used to train the watsonx Code Assistant models?

The watsonx Code Assistant models gather training data from various sources depending on which platform it's supporting. For more information, see [IBM watsonx Code Assistant model details](#)

Can I provide feedback about the models?

You can provide feedback on your experiences, including suggestions for when your results don't match your expectations. For more information about providing feedback, see the [IBM Data and AI Ideas Portal for Customers](#).

Accessibility

IBM strives to provide products with usable access for everyone, regardless of age or ability.

You can interact with all functions of the IBM watsonx Code Assistant content by using only the keyboard.

For more information about the accessibility compliance of the product, see the [Product Accessibility Conformance Reports](#), and then search for IBM watsonx Code Assistant.

Accessibility features in the product documentation

Accessibility features help people with a physical disability, such as restricted mobility or limited vision, or with other special needs, to use information technology products successfully.

The accessibility features in this product documentation allow users to do the following things:

- Use screen-reader software and digital speech synthesizers to hear what is displayed on the screen. Consult the product documentation of the assistive technology for details on using assistive technologies with HTML-based information.
- Use screen magnifiers to magnify what is displayed on the screen.
- Operate specific or equivalent features by using only the keyboard.

The documentation content is published in the IBM Cloud Docs site. For more information about the accessibility of the site, see [Accessibility features for IBM Cloud](#).

Troubleshooting

Why can't I register for the trial of watsonx Code Assistant?

watsonx Code Assistant

You are unsuccessful when you try to register for the free trial.

What's happening

You try to register for the [free trial](#) of watsonx Code Assistant but end up opening the home page for an existing service instance.

Why it's happening

If you already provisioned or have access to any existing service instance, for example, of watsonx Code Assistant for Red Hat Ansible Lightspeed or watsonx Code Assistant for Z, then you can't use the free trial registration process.

How to fix it

To fix this issue, you can use the [IBM Cloud catalog](#) and select the Trial plan.

I receive console errors when I try to connect to watsonx Code Assistant

watsonx Code Assistant

You are unsuccessful when you try to connect to watsonx Code Assistant and receive console errors.

What's happening

When you try to use the Visual Studio Code extension or Eclipse IDE plug-in, you can't connect and receive console errors that include `Request failed. AggregateError`.

Why it's happening

If your organization uses an HTTP proxy server, you need to update your Visual Studio Code or Eclipse IDE settings so that it can connect correctly. You might receive console errors when you try to connect and this setting is missing.

For Visual Studio Code:

1. Open the [http.proxy_setting](#) in Visual Studio Code settings.
2. Enter the URL of your HTTP proxy server in the format `https://proxy_server:port`. You might need to include authentication information, such as `http://user:pass@proxy_server:port`.

For Eclipse IDE:

1. Open the Eclipse IDE settings.
2. In the General section, click **Network Connections**.
3. Set **Active Provider** to `Manual`.
4. Select the schema, such as `HTTPS`, then click **Edit**.
5. Enter the host URL and port of your HTTP proxy server.
6. If you need to include authentication information, click **Requires Authentication**, then enter your username and password.
7. Click **OK** to apply the changes.

Why is the chat window not loading, and other features aren't working?

watsonx Code Assistant

In Visual Studio Code, the chat conversion window doesn't load, and other features aren't working.

What's happening

When you try to use the Visual Studio Code extension the chat window doesn't load, features aren't working, and the WCA Language Server logs include the message `bad option: --use-system-ca`

Why it's happening

You're using an older version of Visual Studio Code.

The watsonx Code Assistant extension is compatible with Visual Studio Code 1.101 and later, using Node.js version 22.15 or later.

To verify your Node.js version in Visual Studio Code, click the **Code** menu, then choose **About Visual Studio Code**. Make sure Node.js is version 22.15 or later.

For more information, see [Installing the extension for Visual Studio Code](#).

When I use multiline code completion, why is the result not in the same language as my file?

watsonx Code Assistant

When you enter a comment to generate a multiline code suggestion, the resulting code is not in the same language as your file.

What's happening

The multiline code suggestion that you receive isn't in the same language as the file you are editing.

Why it's happening

There is a known issue with multiline code suggestions, where the code suggestion might not be in the correct language.

For best results, include the expected programming language in your comment, for example:

```
$ \ \ Sort an array with a fast sort function (TypeScript programming language)
```

For more information, see [Multiline completion](#).


How can I troubleshoot the Java modernization process?

watsonx Code Assistant

If you need to troubleshoot the Java modernization process, it might be useful to examine the files in the application modernization workspace.

When you modernize a Java application, watsonx Code Assistant maintains a workspace in your home directory named `.ibm_java_app_mod`. It is used to store temporary files, results of analysis, and logs. The workspace is sub-divided for each application that you work with where each unique application location corresponds to a UUID string.

In advanced troubleshooting scenarios, you might want to inspect the files and logs in the workspace.

 **Important:** Manually modifying the files in the workspace will lead to unpredictable results.

The structure of the workspace is:

```
$ .ibm_java_app_mod
├── 44daa5e5-6e44-30de-a8bb-8d4ee92b8391. (Unique string corresponding to an application location)
│   ├── JAVA (Files relating to the Upgrade Java version flow)
│   │   ├── binary_scanner_output (Raw output from the binary scanner component)
│   │   │   └── logs
│   │   │       ├── std_err_25.01.27_11.48.48.log
│   │   │       └── trace_25.01.27_11.48.48.log
│   │   └── HTML, JSON, OTHER files
│   └── migration_bundle (Migration bundle files)
│       └── HTML and JSON reports, generated artifacts
├── LIBERTY (Files relating to the Modernize to Liberty flow)
└── app-metadata.json (Meta data for application. Contains UUID to application location mapping)
```

Why can't I connect watsonx Code Assistant for Z to my Db2 database?

watsonx Code Assistant for Z

During the initial cloud setup of your watsonx Code Assistant for Z instance, the automated population of your Db2 account fails.

If your database is associated with a different account than your IBM Cloud instance, or if the automated population in the onboarding checklist fails, use these steps to manually connect your database.

First, you need to collect authentication and hostname information from your Db2 service instance:

1. Log in to cloud.ibm.com.
2. Open the IBM Cloud account that contains your Db2 instance.
3. Click the **Resource list** ☰ icon.
4. Open the **Databases** section, then click your Db2 instance.
5. Click **Service Credentials**.



Note: If you don't have service credentials, click **New credential**, then click **Add**.

6. Click the expand ▼ icon for the credentials.
7. Locate the **authentication** section and make note of the:
 - username
 - password
8. Locate the **hosts** section and make note of the:
 - hostname
 - port

Second, create a connection in your deployment space:

1. In your watsonx Code Assistant for Z instance, open your deployment space.
2. On the **Assets** tab of your deployment space, click **New asset**.
3. In **New asset**, choose **Connect to a data source**.
4. In **Add connection**, select the IBM Db2 on Cloud connector, then click **Next**.
5. Enter the following values to create your connection:
 - **Name:** A meaningful name for your connection
 - **Database:** `bludb`
 - **Hostname or IP address:** The hostname from the service credentials
 - **Port:** The port number from the service credentials
 - **Username:** The username from the service credentials
 - **Password:** The password from the service credentials
6. Click **Test Connection**. If the test completes successfully, click **Create**.

© Copyright IBM Corporation 2026

IBM Corporation
New Orchard Road
Armonk, NY 10504

Produced in the United States of America
2026-03-23

IBM, the IBM logo, and [ibm.com](https://www.ibm.com) are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at <https://www.ibm.com/legal/copytrade>.

This document is current as of the initial date of publication and may be changed by IBM at any time. Not all offerings are available in every country in which IBM operates.

THE INFORMATION IN THIS DOCUMENT IS PROVIDED "AS IS" WITHOUT ANY WARRANTY, EXPRESS OR IMPLIED, INCLUDING WITHOUT ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND ANY WARRANTY OR CONDITION OF NON-INFRINGEMENT.

IBM products are warranted according to the terms and conditions of the agreements under which they are provided.

