CodeArts Artifact

User Guide

Issue	01
Date	2026-08-10

HUAWEI CLOUD COMPUTING TECHNOLOGIES CO., LTD.

Trademarks and Permissions


and other Huawei trademarks are the property of Huawei Technologies Co., Ltd.
All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

The purchased products, services and features are stipulated by the contract made between Huawei Cloud and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

Contents

1 CodeArts Artifact User Guide

CodeArts Artifact assists developers in managing dependencies across different programming languages during development and builds. It stores binary artifacts and centralizes key delivery components. It supports popular package types like Maven and npm. CodeArts Artifact can seamlessly interconnect with local build tools and on-cloud CI/CD so that you can manage software package lifecycle to improve release quality and efficiency. It also provides features such as artifact package version control, granular permission management, and more.

CodeArts Artifact is a service provided within the **CodeArts** solution. For details about its role in the solution, see **CodeArts Architecture**.

Basic operation process of CodeArts Artifact

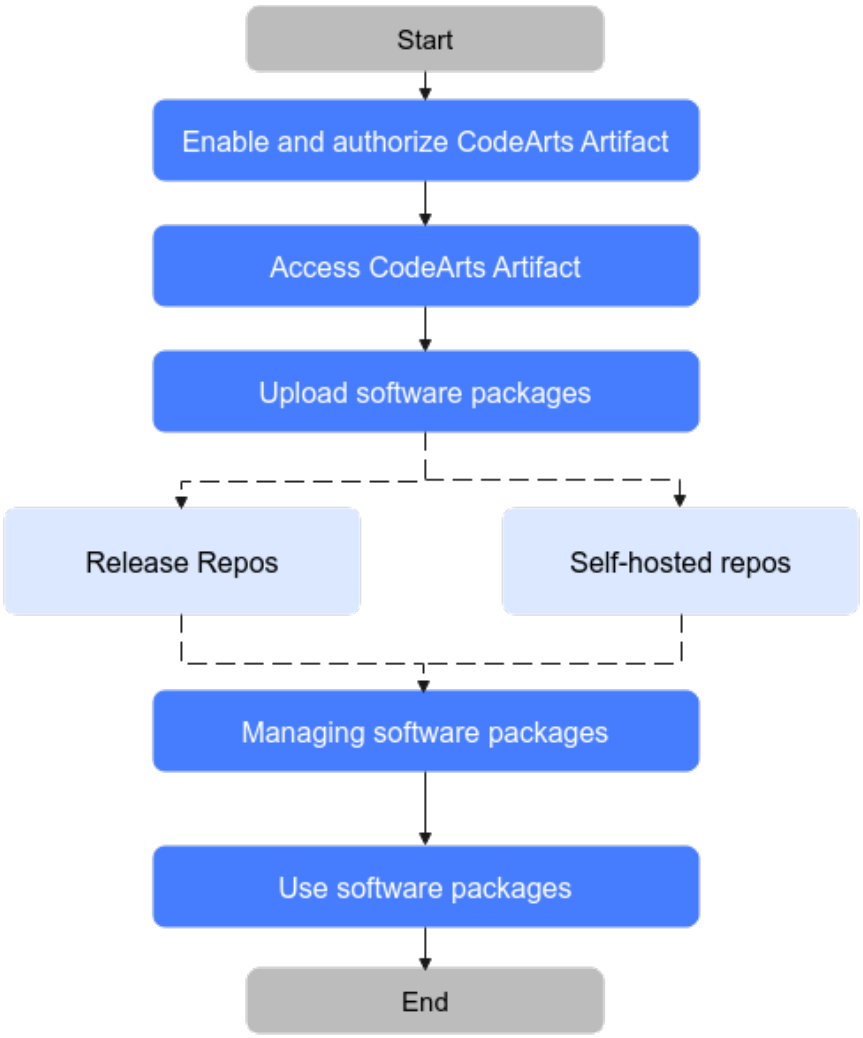


Table 1-1 Steps in using CodeArts Artifact

Step	Description
Purchase CodeArts Artifact	Before using CodeArts Artifact, you need to first **purchase CodeArts Artifact**.
Access CodeArts Artifact	CodeArts Artifact provides two repository types: release repos and self-hosted repos. You can access either **release repos** or **self-hosted repos** as needed.
Upload packages	You can upload packages to **release repos** or **self-hosted repos** for further management and use.
Manage packages	You can manage packages by viewing, downloading, setting status, managing versions, and deleting them in **release repos** and **self-hosted repos**.
Use packages	You can use CodeArts Artifact to store packages during deployment and build. For details, see **CodeArts Artifact Best Practices**.

2 Purchasing CodeArts Artifact

Prerequisites

You have registered with Huawei Cloud and completed real-name authentication. If you do not have a HUAWEI ID yet, follow these steps to create one:

1. Visit the **Huawei Cloud official website**.
2. Click **Sign Up** and create your account as instructed.

   Once your account is created, the system redirects you to your personal information page.
3. Complete individual or enterprise real-name authentication. For details, see **Real-Name Authentication**.

Purchasing CodeArts Artifact

For details, see **Purchasing CodeArts**.

3 Release Repos (New)

3.1 Overview (New)

A release repo is a general-purpose artifact repository that stores and manages artifacts in various formats. In addition to basic storage functions, it also provides important functions such as build and deployment tool integration, version control, and access permission control. It is a standardized way for enterprises to process all artifact types generated during software development.

Constraints

- CodeArts Artifact was upgraded in March 2023. Inventory release repos and resources belonging to users who registered before this update are stored in the old release repos.
- You do not need to manually create release repos. Release repo with the same name as the project is automatically generated when you create the project.

Preparations for Using CodeArts Artifact

- You have subscribed to CodeArts Artifact by referring to **Purchasing a CodeArts Package**.
- You have created a **CodeArts project**.
- You have added CodeArts project members and assigned roles to them. For details, see **Adding Project Members**.

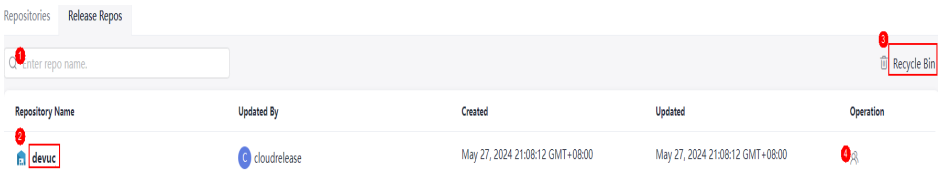
Accessing Release Repos (New)

Step 1 **Log in to the Huawei Cloud console** using your Huawei Cloud account.

Step 2 Click ☰ in the upper left corner of the page and choose **Developer Services** > **CodeArts Artifact** from the service list.

Step 3 Click **Access Service**. The homepage of CodeArts Artifact is displayed.

Step 4 Click the **Release Repos** tab to view the list of repository names under the current tenant. You can then perform the following operations as required.



No.	Operation	Description
1	Search for repositories	Enter the project name in the search box to find the release repos of the project.
2	View folder details	Click any folder to view the list of archived software packages in folders. You can upload, download, and edit software packages or folders.
3	Manage recycle bin	Click **Recycle Bin**. You can delete or restore software packages or folders as required.
4	Set project permissions	Click ··· to go to the **Set Project Permissions** page and edit member permissions. For details, see **Configuring Permissions (New)**.

If you choose **Services** > **Artifact** > **Release Repos** from the top navigation bar, the project list is displayed. But you cannot upload files or create folders there. To perform such operations, click a project name to access it first.

Step 5 Click a repository name to go to the release repo page.

----End

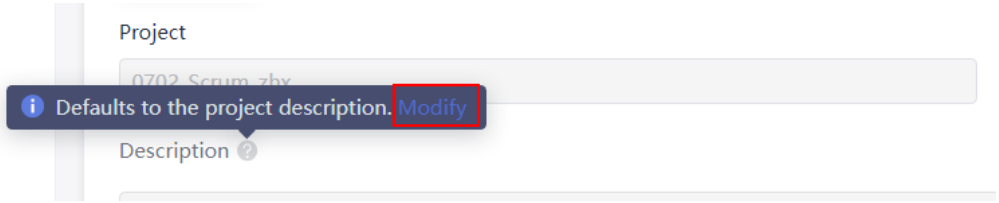
Checking Basic Information

Step 1 **Access release repos** using your Huawei Cloud account.

Step 2 On the release repo page, click **Settings** in the upper right corner of the page.

Step 3 View the repository name, package type, and description.

- The repository name is the same as that of the project and cannot be modified.
- The description is synchronized with that of the project. Click **Modify** to go to the basic project information page and modify the description, as shown in the following figure.



----End

3.2 Configuring Permissions (New)

In CodeArts Artifact, different project roles have different permissions. You can view the default permissions of each project role on the permissions management page of CodeArts Artifact and adjust the role permissions as required.

Constraints

- By default, project administrators have all permissions and their permission scope cannot be modified.
- **Custom roles** created in CodeArts Artifact do not have preset permissions. You can contact the project administrator to configure roles and permissions on corresponding resources.
- By default, the project administrator, project manager, and test manager roles have the **Privilege Config** permission. They can assign other roles permissions for the release repos. If other roles have the **Privilege Config** permission, they can continue to manage permissions for different roles in release repos.

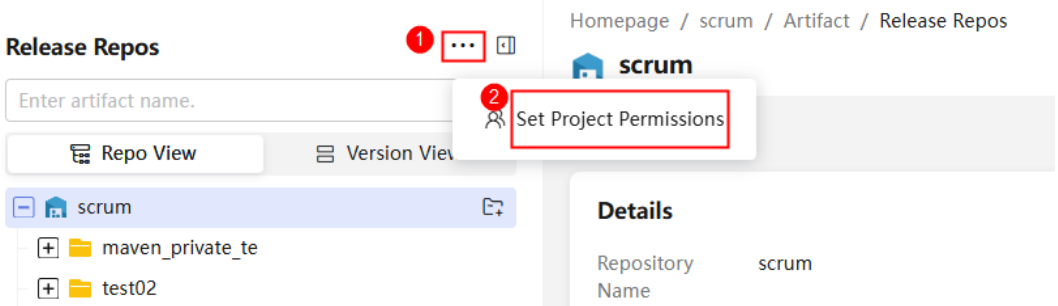
Prerequisites

- You have created a **CodeArts project**.
- You have added users to the CodeArts project and assigned roles to them. For details, see **Adding Project Members**.
- To assign permissions to other roles in the release repos, you must have the **Privilege Config** permission. By default, the project administrator, project manager, and test manager roles have this permission.

Setting Permissions

Step 1 A user with the **Privilege Config** permission accesses the **Release Repos**.

Step 2 Click ··· in the upper right corner and select **Set Project Permissions** from the drop-down list, as shown in the following figure:



Step 3 In the **Roles** area, select the role to configure. On the right of the page, choose **CodeArts Artifact**. The following table lists the default permissions of each role in the release repos.

Table 3-1 Default permissions of each role

Role/ Operation	Change Package Status	Upload	Delete/ Restore (Test Package)	Delete/ Restore (Production Package)	Edit (Test Package)	Create Folder	Download	Restore All	Clear All
Project manager	✓	✓	✓	✗	✓	✓	✓	✓	✓
Product manager	✗	✗	✓	✗	✓	✓	✓	✗	✗
Test manager	✗	✓	✓	✗	✓	✓	✓	✓	✓
Operation manager	✓	✓	✓	✓	✗	✓	✓	✗	✗
System engineer	✗	✓	✓	✗	✓	✓	✓	✗	✗
Committer	✗	✓	✓	✗	✓	✓	✓	✗	✗
Developer	✗	✓	✓	✗	✓	✓	✓	✗	✗
Tester	✗	✓	✗	✗	✗	✓	✓	✗	✗
Participant	✗	✗	✗	✗	✗	✗	✓	✗	✗
Viewer	✗	✗	✗	✗	✗	✗	✗	✗	✗
CI/CD engineer	✓	✓	✗	✗	✓	✗	✓	✗	✗
Project administrator	✓	✓	✓	✓	✓	✓	✓	✓	✓

Step 4 Click **Edit** at the bottom of the page and select or deselect permissions as required.

Step 5 Click **Save**.

Each role can access the **Release Repos** and perform their authorized operations.

----End

3.3 Uploading a Package (New)

Software packages are intermediate products generated during compilation and build in software development. They are an indispensable part of continuous integration and continuous delivery. By uploading packages to release repos for storage and management, you can secure file storage, support software development activities, enable efficient team collaboration, provide reliable software packages for deployment, and provide dependencies for build tasks.

Constraints

When **Multiple files** is selected, a maximum of 20 files can be uploaded at a time.

Prerequisites

- You have been added to a CodeArts project and assigned a role. For details, see **Adding Project Members**.
- You must have permissions to upload packages. For details about the default permissions of each role, see **Table 3-1**. For details about how to obtain permissions, see **Configuring Permissions (New)**.

Procedure

Step 1 **Access release repos** using your Huawei Cloud account.

Step 2 Click **Upload** in the upper right corner of the page to manually upload local packages to release repos.

Alternatively, after selecting a folder, click **Upload** to manually upload local packages to the target folder.

Step 3 In the **Upload** dialog box, set the parameters described in the following table.

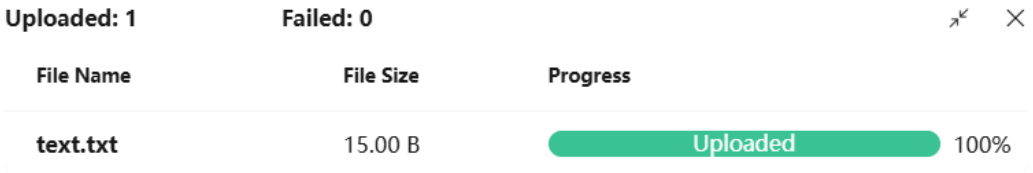
Table 3-2 Parameters for uploading packages

Item	Description
Target Repository	Retain the default settings.
Version	Set the version number for the packages.

Item	Description
Upload Mode	Select **Single file** or **Multiple files** as the upload mode. **Single file** is selected by default here. When **Multiple files** is selected, a maximum of 20 files can be uploaded at a time.
Path	After you set the path name, a folder with that name is created in the **Repo View**, where the uploaded packages will be stored.
File	**CAUTION** **You are advised not to upload files containing sensitive information such as accounts and passwords to the release repos.** Select packages from your local PC to upload.

Step 4 Click **Upload**.

When the progress bar in the lower right corner of the page shows 100% (see the following figure), it indicates the upload is complete. If the upload fails, try again or contact customer service for technical support.



In the **Repo View**, click the name of an uploaded package to view its details.

----End

Follow-Up Operations

After uploading the software packages, you can perform the operations listed in **Table 3-3**.

Table 3-3 Follow-up operations

Operation	Description
Create a subfolder	Click  next to the folder name where a software package is stored to add a subfolder.
Download a package or folder	Click  next to a package or folder name to download it.
Edit a software package or folder	Click  next to a software package or folder name to edit it.

Operation	Description
Delete a software package or folder	Click next to a software package or folder name to either permanently delete it or move it to the recycle bin.

Helpful Links

- CodeArts Artifact allows you to upload software packages either on the release repo page or through CodeArts Build. For details, see **Uploading a Package**.
- If you cannot upload files or create directories on the Release Repos homepage, see **Why Can't I Upload Files or Create Directories on the Release Repos Homepage?**
- If you choose to move a software package or folder to the recycle bin when deleting it, you can restore or permanently delete it from the recycle bin. For details, see **Recycle Bin**.

3.4 Managing Packages

In the release repo, you can view package details, update versions, manage version status, and set folder and package status in the repo or version view. This helps you quickly access resources and enables fine-grained management and release control.

Managing Packages in Repo View

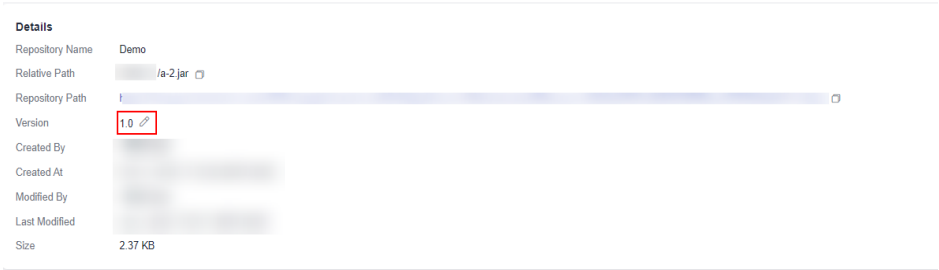
Step 1 **Access release repos** using your Huawei Cloud account.

Step 2 On the **Release Repos** page, click the **Repo View** tab.

Step 3 Click a package name. The details about the package are displayed. The details include the **General**, **Build Metadata**, and **Build Packages** tabs.

- **General**: displays the repository name, relative path, repository URL (for download), version, creator, creation time, modifier, modified time, size, and checksums.

  Click to change the version number of an uploaded package. By default, the release version of a software package archived by CodeArts Build is the version number set when the build task is executed, as shown in the following figure:



- **Build Metadata**
  - **Built-in**: displays the build task, size, build number, builder, code repository, and code branch of the package. Click **Build Task** to link to the task in CodeArts Build.
  - **Custom**: displays custom parameters in the build task. If the value of this field is a URL starting with **http** or **https**, you can click the link to go to the corresponding page.
- **Build Packages**: displays records of packages archived from build tasks. Click to download the package.

----End

Managing Packages in Version View

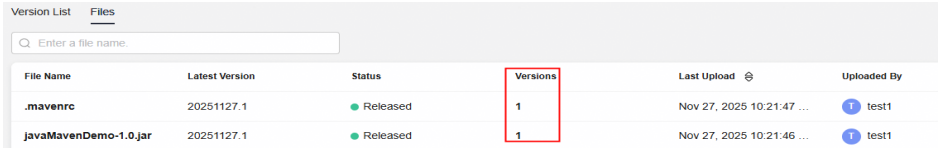
CodeArts Artifact displays packages by version. In **Version View** tab, you can filter and check artifacts by name and version number, and sort files by their update time.

Step 1 Access release repos using your Huawei Cloud account.

Step 2 On the **Release Repos** page, click the **Version View** tab. The list of packages with version numbers is displayed. Release repo stores packages of different versions with the same name in the same file.

Step 3 Click the **Files** tab and perform the following operations:

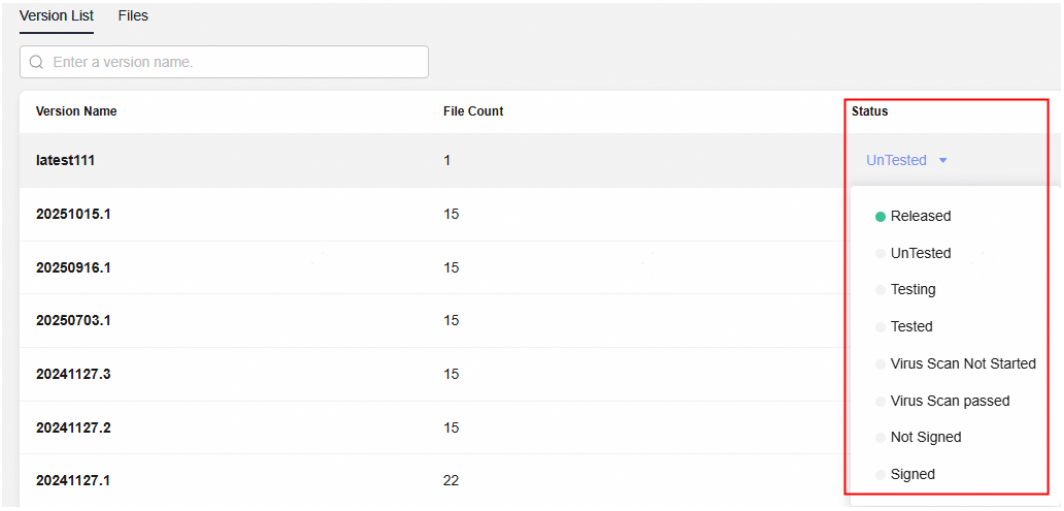
- Click a file name in the **File Name** column. The latest version of the package is displayed.
- Click a number in the **Versions** column. The version list of the package is displayed.



- Click a number in the **Version No.** column. The **Overview** and **Files** pages of the package are displayed. In the **Files** tab, click a name in the **File Name** column. The path of the package is displayed.

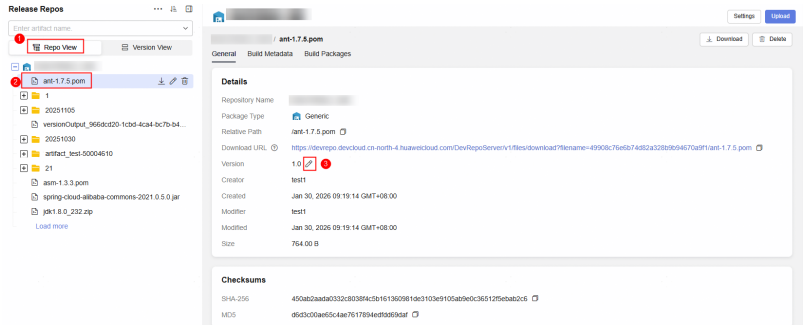
Step 4 Click the **Version List** tab and perform the following operations:

Change the status if needed. The default status is **UnTested** after the version is configured, as shown in the following figure:



CAUTION

The file's release version is latest by default. You cannot change the version status. To do so, first change the **Version** in the repo view (for example, from **latest** to **1.0.0**), as shown in the following figure:



- In the **Version List**, set the status to **Released**. All packages in this version are updated accordingly.

  After the status is changed to Released, the file cannot be edited. This action is irreversible. You can only download or delete the file; you cannot modify it, including its name or version number.
- In the **Version List**, you can set the status to any of the following:

  UnTested, **Testing**, **Tested**, **Virus Scan Not Started**, **Virus Scan passed, Not Signed**, and **Signed**

  You can change between these statuses freely. Once a status is set to **Released**, it cannot be edited. This change cannot be reversed, so proceed with caution.

----End

Setting Status of a Folder

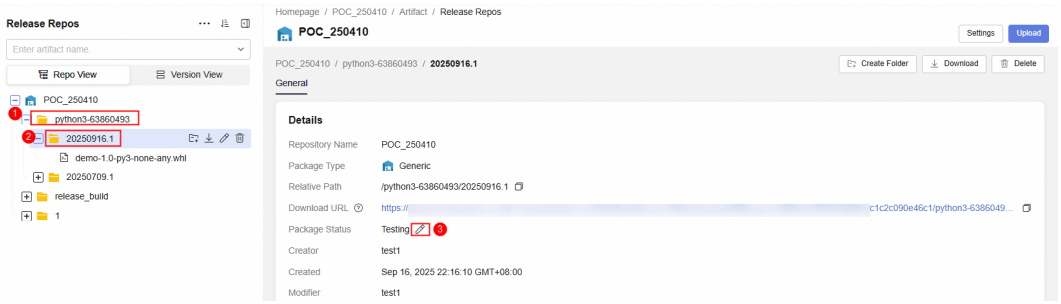
Prerequisites: You have the **changePkgStatus** permission. For details about the default permissions of each role, see **Table 3-1**. For details about how to configure permissions, see **Configuring Permissions (New)**.

Procedure

Step 1 Access release repos using your Huawei Cloud account.

Step 2 On the **Release Repos** page, click the **Repo View** tab.

Step 3 Enter a first-level folder, click the second-level folder, and click next to **Package Status**, (**Testing** by default), as shown in the following figure.



Step 4 Select a new status from the drop-down list.

- If the folder status is **Released**, the folder cannot be changed or edited (changing the folder name, changing the file name in the folder, uploading the folder, changing the version number, or creating a subfolder). You can only download or delete it.
- The folder status can be changed from **Not Released** to **Released**, but such change is irreversible. Exercise caution when performing this operation.

----End

Downloading a Package

Prerequisites: You have the **download** permission. For details about the default permissions of each role, see **Table 3-1**. For details about how to configure permissions, see **Configuring Permissions (New)**.

Procedure

Step 1 Go to the CodeArts homepage.

1. Log in to the **CodeArts console**, click , and select a region where you have enabled CodeArts.
2. Click **Go to Workspace**.

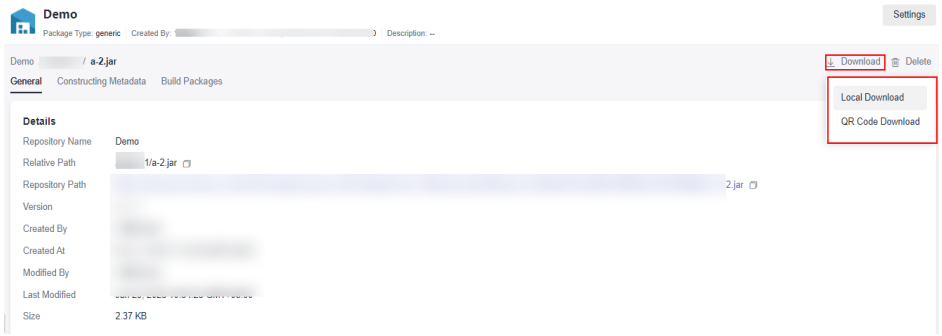
   If your account uses the old billing mode (see **Old Billing Modes**), click **Access Service**.

Log in to the **CodeArts console**, and click **Access Service**.

Step 2 Click a project card to access the project, and choose **Artifact** > **Release Repos** from the menu bar.

Step 3 In the **Repo View** tab, select the package to be downloaded. You can download it in either of the following ways:

- Method 1: Click **Download** on the right of the page and select a download mode from the drop-down list.



  - **Local Download**: Download the package to the localhost.
  - **QR Code Download**: Use your mobile phone to scan the QR code to download the package.

- Method 2: Hover over the package you want to download, and click on the right.

----End

Searching for a Package

Step 1 Go to the CodeArts homepage.

1. Log in to the **CodeArts console**, click , and select a region where you have enabled CodeArts.
2. Click **Go to Workspace**.

   If your account uses the old billing mode (see **Old Billing Modes**), click **Access Service**.

Log in to the **CodeArts console**, and click **Access Service**.

Step 2 Click a project card to access the project, and choose **Artifact** > **Release Repos** from the menu bar.

Step 3 Enter a keyword (a project or file name) in the search box above the list to search for the package whose name contains the keyword.

Step 4 Click the file name to go to its details page.

----End

Deleting a Package

Step 1 Go to the CodeArts homepage.

1. Log in to the **CodeArts console**, click , and select a region where you have enabled CodeArts.
2. Click **Go to Workspace**.

   If your account uses the old billing mode (see **Old Billing Modes**), click **Access Service**.

Log in to the **CodeArts console**, and click **Access Service**.

Step 2 Click a project card to access the project, and choose **Artifact** > **Release Repos** from the menu bar.

Step 3 In the left pane, locate the package you want to delete, and click its name.

If there are too many packages, you can search for the desired package and directory by referring to **Searching for a Package**.

Step 4 Click **Delete** on the right of the package or directory.

Step 5 In the displayed dialog box, enter the name of the package to be deleted and click **OK**.

----End

3.5 Clearing Policies

To keep your release repos organized and efficient, you can enable a scheduled auto-clearing policy to move expired files to the recycle bin or permanently delete outdated ones.

Setting Clearing Policies

Step 1 Go to the CodeArts homepage.

1. Log in to the **CodeArts console**, click , and select a region where you have enabled CodeArts.
2. Click **Go to Workspace**.

   If your account uses the old billing mode (see **Old Billing Modes**), click **Access Service**.

Log in to the **CodeArts console**, and click **Access Service**.

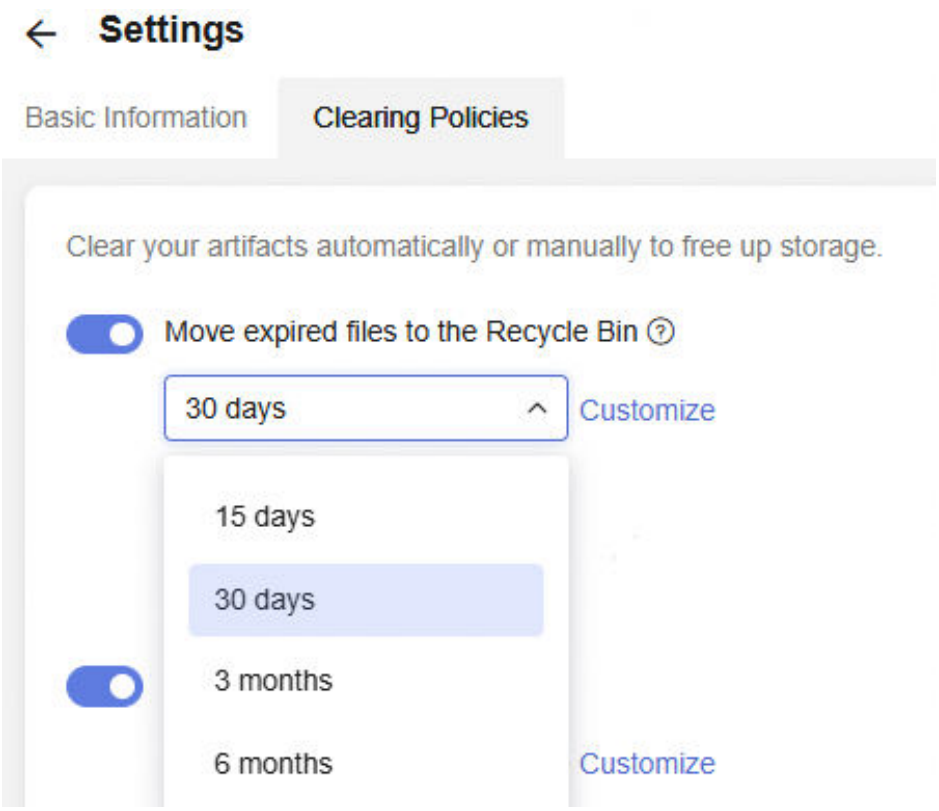
Step 2 Click a project card to access the project, and choose **Artifact** > **Release Repos** from the menu bar.

Step 3 Click **Settings** in the upper right corner of the page and select **Clearing Policies**.

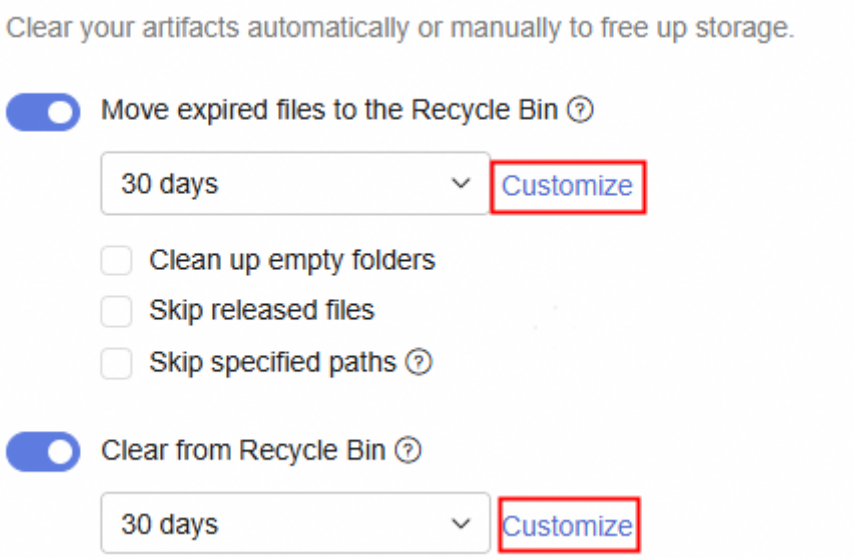
Step 4 Enable **Move expired files to the Recycle Bin** or **Clear from Recycle Bin** as required, and select a retention period from the drop-down list.

Default retention periods:

- **Move expired files to Recycle Bin**: 30 days
- **Clear from Recycle Bin**: 30 days



- You can also set a period. Click **Customize**, enter a number, and click √ to save the setting.



The following configurations are optional.

  - **Clean up empty folders**: The system removes empty folders when files are cleared.
  - **Skip released files**: The system retains files in the production package state when clearing files. For details, see **Setting Production Package Status**.

- **Skip specified paths**: The system retains packages that match the file paths you set when clearing files.
  - You can set multiple file paths, each starting with a slash (/) and separated by semicolons (;) but file path names cannot contain the project name. Example: **/test/folder1/;/1.0.0/test.txt**
  - Spaces before and after the file path name will be automatically removed.
  - The file path uses prefix matching. Ignore directories ending with a slash (/) and ignore files with a full file path. Otherwise, the ignored file may not match.

----End

3.6 Recycle Bin

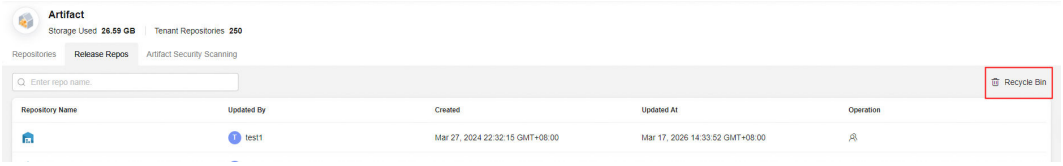
Packages or folders deleted from release repos are moved to the recycle bin. You can restore or permanently delete them there. CodeArts Artifact includes both a global and a project-level recycle bin.

Global Recycle Bin

In the global recycle bin, you can manage packages and folders deleted from any project.

Step 1 **Log in to the CodeArts Artifact homepage** using your Huawei Cloud account.

Step 2 Click the **Release Repos** tab and click **Recycle Bin** on the right of the page, as shown in the figure below.



Step 3 Deleted files from different projects are listed on this page. Perform the following operations on a package or folder as required.

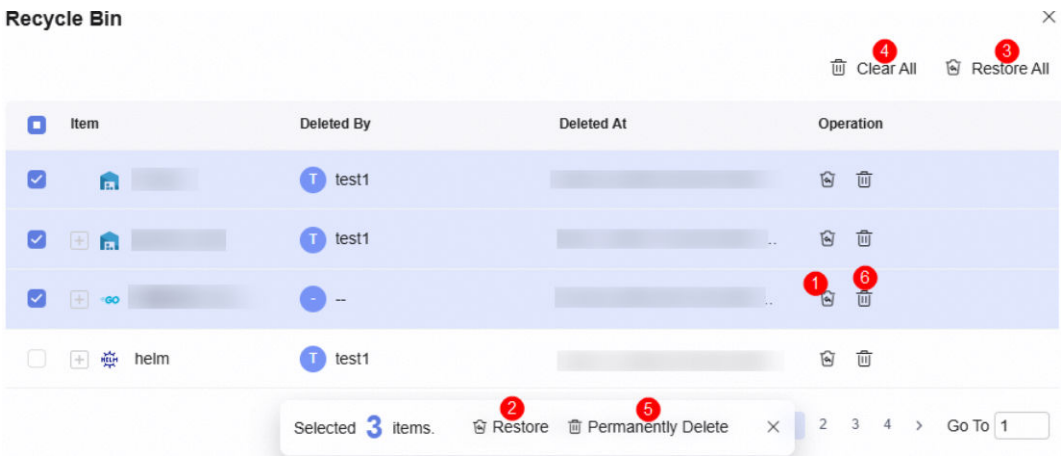


Table 3-4 Operations in the global recycle bin

No.	Operation	Description
1	Individual restore	Click in the **Operation** column to restore the package or folder.
2	Batch restore	Select multiple packages or folders and click **Restore** below the list to restore all the selected items.
3	Restore all	Click **Restore All** to restore all packages or folders in the recycle bin by one click.
4	Clear all	Click **Clear All** to delete all packages or folders from the recycle bin.
5	Batch delete	Select multiple packages or folders and click **Delete** below the list to delete all the selected items.
6	Clear	Click in the **Operation** column to delete the package or folder.

WARNING

Once you delete a package or folder from the recycle bin, it cannot be recovered. Exercise caution when performing this operation.

You cannot batch restore or delete files across projects in the global recycle bin.

----End

Project-Level Recycle Bin

You can manage deleted packages or folders in a project.

Step 1 Go to the CodeArts homepage.

1. Log in to the **CodeArts console**, click , and select a region where you have enabled CodeArts.
2. Click **Go to Workspace**.

   If your account uses the old billing mode (see **Old Billing Modes**), click **Access Service**.

Log in to the **CodeArts console**, and click **Access Service**.

Step 2 Go to the CodeArts homepage.

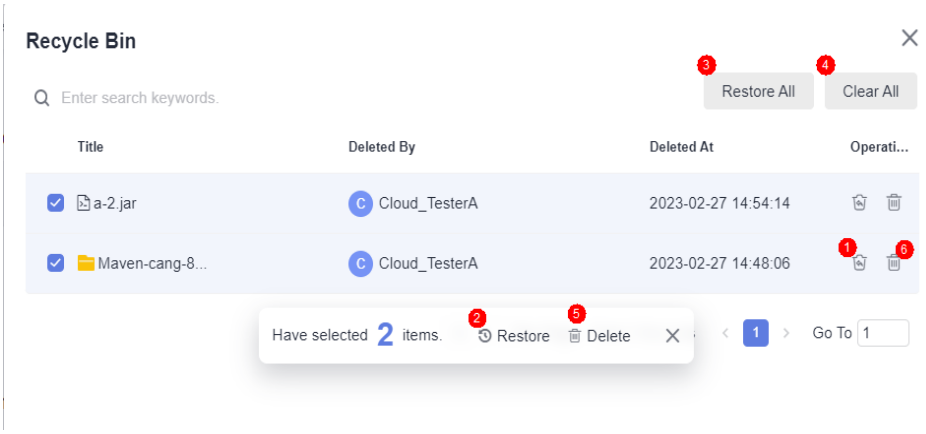
Step 3 Click a project card to access the project, and choose **Artifact** > **Release Repos** from the menu bar.

Step 4 Click **Recycle Bin** in the lower left corner of the page.

Step 5 Deleted files of this project are listed on this page. Perform the following operations on a package or folder as required.

WARNING

Once you delete a package or folder from the recycle bin, it cannot be recovered. Exercise caution when performing this operation.



No.	Operation	Description
1	Individual restore	Click in the **Operation** column to restore the package or folder.
2	Batch restore	Select multiple packages or folders and click **Restore** below the list to restore all the selected items.
3	Restore all	Click **Restore All** to restore all packages or folders in the recycle bin by one click.
4	Clear all	Click **Clear All** to delete all packages or folders from the recycle bin.
5	Batch delete	Select multiple packages or folders and click **Delete** below the list to delete all the selected items.
6	Clear	Click in the **Operation** column to delete the package or folder.

----End

Helpful Links

- A file cannot be restored from the recycle bin page. A message indicating that a duplicate file exists is displayed. For details, see **Can I Restore Files in the Recycle Bin of My Release Repos?**

4 Self-Hosted Repos (New)

4.1 Overview

Developers often need to share packages with their team during development. Self-hosted repos serve as a central place where these packages can be stored and accessed by others, making it easy for team members to obtain packages from repositories.

A self-hosted repo manages private packages, supporting Maven, npm, Go, NuGet, PyPI, RPM, Debian, Conan, Docker, CocoaPods, and OHPM.

In March 2023, CodeArts Artifact was upgraded. Before this upgrade, existing self-hosted repos were not associated to projects. Consequently, repos and their resources from before the upgrade remain in the old repos.

Accessing Self-Hosted Repos

Step 1 Subscribe to CodeArts Artifact by referring to **Purchasing a CodeArts Package**.

Step 2 Add members and assign roles to them. For details, see **Configuring Permissions (New)**.

Step 3 **Log in to the Huawei Cloud console**.

Step 4 Click ☰ in the upper left corner of the page and choose **Developer Services** > **CodeArts Artifact** from the service list.

Step 5 Access CodeArts Artifact in either of the following ways.

- **From the service portal**

  Click **Access Service**. The homepage of CodeArts Artifact is displayed. Click the **Repositories** tab. All self-hosted repos created are displayed.

- **From the project page**

  a. Click **Access Service**. The homepage of CodeArts Artifact is displayed.

  b. Click **Homepage** on the top navigation bar.

  c. Click the name of the project you want to view.

d. Choose **Artifact** > **Self-hosted Repos** from the navigation pane.

Click in the upper left corner of the page and select a region.

----End

4.2 Creating a Repository

4.2.1 Overview

Self-hosted repos include local, proxy, and virtual repositories.

Local Repository

- Definition: A local repository stores all artifacts on the CodeArts Artifact server. It is used to store and manage build outputs, dependencies, and third-party artifacts.
- Features:
  - Privacy: The repository is fully controlled by you and is not shared with external systems.
  - Durability: Data is permanently stored on the CodeArts Artifact server.
  - Performance: Fast access speeds.
  - Usage: It is suitable for storing self-developed components, binary files, and configuration files.

Proxy Repository

This repository type is supported only at region AP-Singapore.

- Definition: A proxy of a mirror, public, or third-party artifact repository for pulling and caching requested resources.
- Features:
  - Caching: The first request pulls resources from the external repository; subsequent requests use the cache.
  - Bandwidth saving: Repeated requests to external repositories are reduced, saving network bandwidth.
  - Offline access: Cached resources remain accessible if the external repository is unavailable.
  - Usage: It is suitable for proxying mirrors or public repositories, such as Maven Central and npmjs.com.

Virtual Repository

- Definition: A virtual repository is a collection of local and proxy repositories accessed through a single URL.
- Features:
  - Aggregation: A virtual repository can include multiple local and proxy repositories.

- Single URL: Resources in multiple repositories can be accessed through a single logical URL.
- Transparency: Clients do not need to know which repository stores the resources.
- Flexibility: Repositories can be added or removed as needed.
- Usage: It is used to simplify dependency management and build processes, and provides a unified view.

4.2.2 Local Repository

4.2.2.1 Creating a Local Repository

A local repository stores all artifacts on the CodeArts Artifact server. It is used to store and manage build outputs, dependencies, and third-party artifacts. If you use this service for the first time, you need to create a repository.

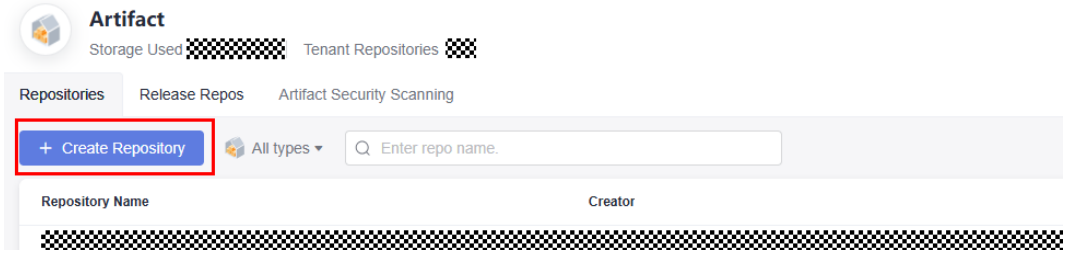
Prerequisites

- You have created a **CodeArts project**.
- You have the **createRepository** permission. For details about how to add this permission, see **Configuring Repository Permissions**.
- You have been added to a CodeArts project and assigned a role. For details, see **Adding Project Members**.

Procedure

Step 1 Use your Huawei Cloud account to access **self-hosted repos**.

Step 2 Click **Create Repository**.



Step 3 On the displayed page, select a repository type. For example, set **Repository Type** to **Local Repo** and then select the package type (such as Maven, npm, or PyPI) for the repository.

Step 4 Configure the basic settings for the local repository by referring to **Table 4-1**.

Table 4-1 Parameters for creating a local repository

Item	Description
Repository Name	You can enter a unique repository name. Enter up to 50 characters. Only letters, digits, underscores (_), hyphens (-), and periods (.) are supported. For example, to create a local repository, you can name it **maven-local**. **WARNING** After a repository is created, its name cannot be changed.
Project	The project to which the current repository belongs. If no project exists, click **New Project** in the drop-down list to create one. **WARNING** The project cannot be changed after being set.
Associated Projects	This parameter is available only when **Package Type** is set to **Maven**. This parameter is used to share artifacts in CodeArts Build. By default, CodeArts Build pulls artifacts only from repositories in the current project. You can configure associated projects to share artifacts across projects. After the target repository is associated with the project where the build task is located, build tasks in that project can pull artifacts from the repository.
Include Patterns	Specify the files to include in the operation. These patterns are usually used to filter in and select specific files or folders. If you configure this parameter, only artifacts that match the added path prefix can be operated on (including artifact uploads and downloads). You can click **+** to add multiple inclusion rules. The relationship between multiple inclusion rules is **OR**.
Exclude Patterns	Specify the files to exclude from the operation. These patterns are usually used to filter out and select specific files or folders. If you configure this parameter, artifacts that match the added path prefix cannot be operated on (including artifact uploads and downloads). You can click **+** to add multiple exclusion rules. The relationship between multiple exclusion rules is **OR**.

Item	Description
Version Policy	This parameter is available only when **Package Type** is set to **Maven**. The options are **Release**, **Snapshot**, and **Release&Snapshot**. Two repositories will be generated: **Release** and **Snapshot**. The **Release** repository is selected by default and the **Snapshot** and **Release&Snapshot** repository can also be selected as required. • The **Release** repository stores release versions with stable functions that will no longer be updated. • The **Snapshot** repository stores development versions with unstable functions in the development stage. • The **Release&Snapshot** repository stores release and snapshot versions simultaneously.
Open To	Set the repository permission scope. • If you select **Project**, the created repository belongs to the associated project. Repositories in a project use roles in the project for authentication. For details about repository permissions by role in a project, see **Configuring Project-Level Permissions**. • If you select **Tenant**, the created repository will belong to the tenant associated with it. Repositories within a tenant use tenant space role authentication. For details about repository permissions by role in a tenant, see **Configuring Repository-Level Permissions**.
Description	Enter up to 200 characters to describe the repository, such as its purpose or usage.

Step 5 Check all configured information and click **OK**.

After the local repository is created, you can view it in the repository list.

----End

4.2.2.2 Configuring a Local Repository

After a local repository is created, you can go to the **Settings** page and configure information in the **Basic Information**, **Project Associations**, **Repository Permissions**, and **Deployment Policies** tabs as required.

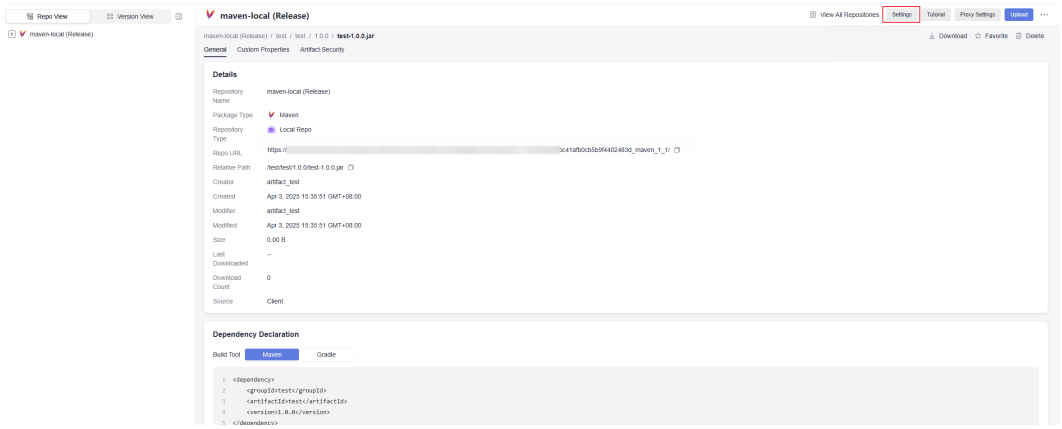
Prerequisites

- You have the **editRepository** permission. For details about how to add this permission, see **Configuring Repository Permissions**.

Procedure

Step 1 Use your Huawei Cloud account to access **self-hosted repos**.

Step 2 Click **Settings** on the right of the repository.



Step 3 Configure repository information by referring to **Table 4-2**.

Table 4-2 Parameters for configuring a local repository

Item		Description
Basic Information	Include Patterns	Specify files or folders to include by configuring a file path pattern. Only artifacts that match the added path prefix can be uploaded or downloaded. **Ensure that the path pattern is correct to avoid disrupting normal operations.**
	Exclude Patterns	Specify files or folders to exclude by configuring a file path pattern. Artifacts that match the added path prefix cannot be operated on. **Ensure that the path pattern is correct to avoid disrupting normal operations.**
	Description	Describe the basic information and purpose of the repository.
Associated Projects		This parameter is available only when **Package Type** is set to **Maven**. This parameter is used to share artifacts in CodeArts Build. By default, CodeArts Build pulls artifacts only from repositories in the current project. You can configure associated projects to share artifacts across projects. After the target repository is associated with the project where the build task is located, build tasks in that project can pull artifacts from the repository.

Item	Description
Repository Permissions	**Configure permissions carefully, preferably with professional guidance.** This function allows you to manage repository permissions of different roles. You can configure access permissions for different users or roles as required. For details, see **User Permissions Management**.
Deployment Policies	**Select a deployment policy carefully, as it affects artifact management.** Deployment policies control how artifacts are deployed in the repository. Available options include: • **Allow redeploy** (default): Artifacts in the same path can be uploaded. • **Disable redeploy**: Artifacts in the same path cannot be uploaded. • **Read-only**: Artifacts cannot be uploaded, updated, or deleted.

Step 4 Click **Submit** to save the configurations.

Changes take effect immediately.

----End

4.2.3 Proxy Repository

4.2.3.1 Creating a Proxy Repository

During software development, teams often need to download dependencies from open-source communities or third-party repositories. However, downloading dependencies directly from these sources may cause network instability and slow download speeds. CodeArts Artifact allows you to create proxy repositories for open-source and third-party dependency repositories. Files downloaded from proxy repositories are cached in CodeArts Artifact. This improves the download speed and stability of dependencies.

Constraints

This repository type is supported only at region AP-Singapore.

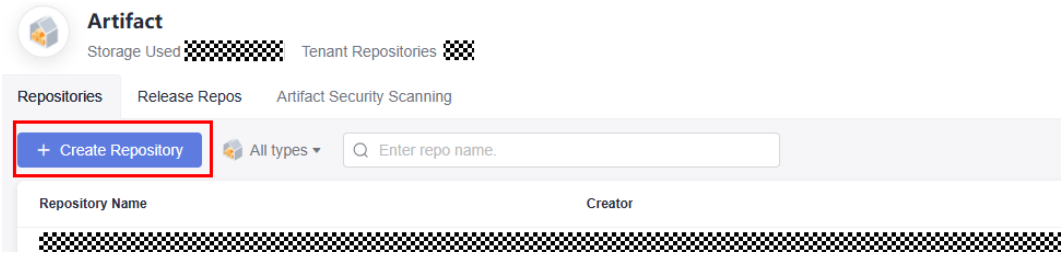
Prerequisites

- You have created a **CodeArts project**.
- You have the **createRepository** permission. For details about how to add this permission, see **Configuring Repository Permissions**.

Procedure

Step 1 Use your Huawei Cloud account to access **self-hosted repos**.

Step 2 Click **New Repository**, as shown in the following figure.



Step 3 On the displayed page, set **Repository Type** to **Proxy Repo**. Select the package type for your repository, for example, Maven, npm, or PyPI.

Step 4 Configure the basic information, as described in the following table.

Table 4-3 Parameters

Item	Description
Repository Name	You can enter a unique repository name. Enter up to 50 characters. Only letters, digits, underscores (_), hyphens (-), and periods (.) are supported. **WARNING** After a repository is created, its repository name cannot be changed.
Proxy Address	Address of the repository that requires proxy. Common proxy repositories include mirrors, public repositories, and third-party repositories. When a resource is requested, CodeArts Artifact pulls the resource from the proxy repository.
PyPI Index Proxy Address	This parameter is available only when **Package Type** is set to **PyPI**. It refers to the index site for Python software packages, also known as the python package index (PyPI). Example: **https://mirrors.tools.****.com/pypi**
Username	If the proxy repository requires authentication, enter the proxy repository username.
Password	If the proxy repository requires authentication, enter the proxy repository password.
Net Proxy Name	If a network proxy is required to connect to CodeArts Artifact, select the proxy to use. Select the proxy from the drop-down list. For details, see **Configuring a Network Proxy**.

Item	Description
Project	The project to which the current repository belongs. If no project exists, click **New Project** in the drop-down list to create one. **WARNING** After the repository is created, its name and project cannot be changed.
Associated Projects	This parameter is available only when **Package Type** is set to **Maven**. This parameter is used to share artifacts in CodeArts Build. By default, CodeArts Build pulls artifacts only from repositories in the current project. You can configure associated projects to share artifacts across projects. After the target repository is associated with the project where the build task is located, build tasks in that project can pull artifacts from the repository.
Include Patterns	Specify the files to include in the operation. These patterns are usually used to filter in and select specific files or folders. If you configure this parameter, only artifacts that match the added path prefix can be operated on (including artifact uploads and downloads). You can click **+** to add multiple inclusion rules. The relationship between multiple inclusion rules is **OR**.
Exclude Patterns	Specify the files to exclude from the operation. These patterns are usually used to filter out and select specific files or folders. If you configure this parameter, artifacts that match the added path prefix cannot be operated on (including artifact uploads and downloads). You can click **+** to add multiple exclusion rules. The relationship between multiple exclusion rules is **OR**.
Open To	Set the repository permission scope.
Description	Enter up to 200 characters to describe the repository, such as its purpose or usage.

Step 5 Confirm the configurations and click **OK**.

After the repository is created, you can view it in the repository list.

----End

4.2.3.2 Configuring a Proxy Repository

After a local repository is created, you can go to **Settings** > **Basic Information** to configure basic information, the proxy address, and the username as needed.

Prerequisites

- You have created a **CodeArts project**.
- You have the **editRepository** permission. For details about how to add this permission, see **Configuring Repository Permissions**.

Procedure

Step 1 Use your Huawei Cloud account to access **self-hosted repos**.

Step 2 Select the created proxy repository from the repository list.

Step 3 Click **Settings** on the right of the repository.

Step 4 Configure the proxy repository parameters by referring to **Table 4-4**.

Table 4-4 Proxy repository configuration parameters

Category	Item	Description
Basic information	Associated Projects	This parameter is available only when **Package Type** is set to **Maven**. This parameter is used to share artifacts in CodeArts Build. By default, CodeArts Build pulls artifacts only from repositories in the current project. You can configure associated projects to share artifacts across projects. After the target repository is associated with the project where the build task is located, build tasks in that project can pull artifacts from the repository.
	Include Patterns	Specify the files to include in the operation. These patterns are usually used to filter in and select specific files or folders. If you configure this parameter, only artifacts that match the added path prefix can be operated on (including artifact uploads and downloads). You can click **+** to add multiple inclusion rules. The relationship between multiple inclusion rules is **OR**.
	Exclude Patterns	Specify the files to exclude from the operation. These patterns are usually used to filter out and select specific files or folders. If you configure this parameter, artifacts that match the added path prefix cannot be operated on (including artifact uploads and downloads). You can click **+** to add multiple exclusion rules. The relationship between multiple exclusion rules is **OR**.

Category	Item	Description
	Open To	Set the repository permission scope. • If you select **Project**, the created repository belongs to the associated project. Repositories in a project use roles in the project for authentication. For details about repository permissions by role in a project, see **Configuring Project-Level Permissions**. • If you select **Tenant**, the created repository will belong to the tenant associated with it. Repositories within a tenant use tenant space role authentication. For details about repository permissions by role in a tenant, see **Configuring Repository-Level Permissions**.
	Description	Enter up to 200 characters to describe the repository, such as its purpose or usage.
Proxy repository configuration	Proxy Address	Address of the repository that requires proxy. Common proxy repositories include mirrors, public repositories, and third-party repositories. When a resource is requested, CodeArts Artifact pulls the resource from the proxy repository.
	Username	If the proxy repository requires authentication, enter the proxy repository username.
	Password	If the proxy repository requires authentication, enter the proxy repository password.
	PyPI Index Proxy Address	This parameter is available only when **Package Type** is set to **PyPI**. It refers to the index site for Python software packages, also known as the python package index (PyPI). Example: **https://mirrors.tools.****.com/pypi**
	Net Proxy Name	If a network proxy is required to connect to CodeArts Artifact, select the proxy to use. Select the proxy from the drop-down list. For details, see **Configuring a Network Proxy**.

Step 5 Click **Submit** to save the configurations.

Changes take effect immediately.

----End

4.2.3.3 Configuring a Network Proxy

This section describes how to configure a network proxy for artifact repository access. If a proxy is required, configure it as described here to ensure developers can access the repository and complete development and build tasks.

Prerequisites

You must have the project manager or project administrator role. For details about how to add this role, see **Managing Project Members**.

Configuring a Network Proxy for the Proxy Repository

Step 1 Use your Huawei Cloud account to access **self-hosted repos**.

Step 2 Click the username in the upper right corner, and select **All Account Settings** from the drop-down list.

Step 3 In the navigation pane, choose **Mirror** > **Mirror**.

Step 4 On the **Proxy Settings** page, click the **Network** tab, and set the proxy name, network proxy address, and port.

- **Proxy Name**: Enter a proxy name. Only digits, letters, hyphens (-), and underscores (_) are supported.
- **Net Proxy Address**: Enter the address of the proxy server installed during the network configuration of the proxy mirror repository. Contact the environment administrator or O&M personnel to obtain the proxy address.
- **Port**: Enter the port of the proxy server installed during the network configuration of the proxy mirror repository. Contact the environment administrator or O&M personnel to obtain the port.

Step 5 Click **OK**. The network proxy is added.

You can view the new network proxy in the proxy list.

Step 6 (Optional) Perform the operations listed in **Table 4-5** for the added network proxy.

Table 4-5 Managing a network proxy

Operation	Description
Edit	Click in the **Operation** column to modify the proxy name, network proxy address, and port.
Delete	Click in the **Operation** column to delete the network proxy. If the network proxy to be deleted is associated with a repository, first remove the proxy on the repository's **Configuring a Proxy Repository** page and then return to this page to delete the proxy.

----End

4.2.4 Virtual Repository

4.2.4.1 Creating a Virtual Repository

A virtual repository aggregates local and proxy repositories, offering a single, unified entry for accessing artifacts.

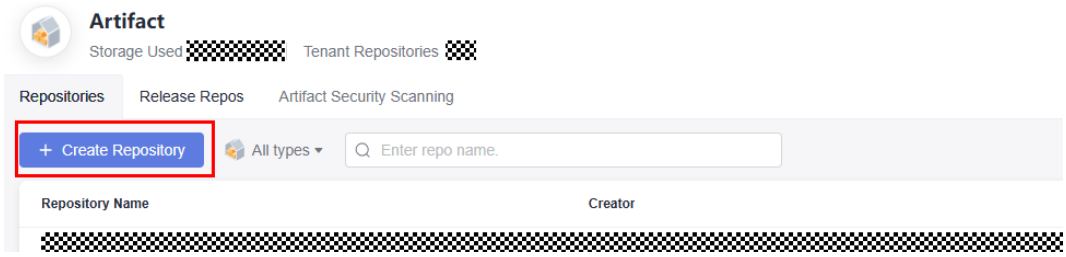
Prerequisites

- You have created a **CodeArts project**.
- You have the **createRepository** permission. For details about how to add this permission, see **Configuring Repository Permissions**.

Procedure

Step 1 Use your Huawei Cloud account to access **self-hosted repos**.

Step 2 Click **Create Repository**.



Step 3 Enter repository information as prompted.

1. Select a repository type.

   Set **Repository Type** to **Virtual Repo**. Select the package type for your repository, for example, Maven, npm, or PyPI.

2. Configure the basic information, as described in the following table.

Table 4-6 Parameters

Item	Description
Repository Name	You can enter a unique repository name. For example, to create a Maven repository, you can name it **maven-virtual**. Enter up to 50 characters. Only letters, digits, underscores (_), hyphens (-), and periods (.) are supported.
Project	The project to which the current repository belongs. If no project exists, click **New Project** in the drop-down list to create one.

Item	Description
Associated Projects	This parameter is available only when **Package Type** is set to **Maven**. This parameter is used to share artifacts in CodeArts Build. By default, CodeArts Build pulls artifacts only from repositories in the current project. You can configure associated projects to share artifacts across projects. After the target repository is associated with the project where the build task is located, build tasks in that project can pull artifacts from the repository.
Include Patterns	Specify the files to include in the operation. These patterns are usually used to filter in and select specific files or folders. If you configure this parameter, only artifacts that match the added path prefix can be operated on (including artifact uploads and downloads). You can click **+** to add multiple inclusion rules. The relationship between multiple inclusion rules is **OR**.
Exclude Patterns	Specify the files to exclude from the operation. These patterns are usually used to filter out and select specific files or folders. If you configure this parameter, artifacts that match the added path prefix cannot be operated on (including artifact uploads and downloads). You can click **+** to add multiple exclusion rules. The relationship between multiple exclusion rules is **OR**.
Version Policy	This parameter is available only when **Package Type** is set to **Maven**. The options are **Release**, **Snapshot**, and **Release&Snapshot**. Two repositories will be generated: **Release** and **Snapshot**. The **Release** repository is selected by default and the **Snapshot** and **Release&Snapshot** repository can also be selected as required. – The **Release** repository stores release versions with stable functions that will no longer be updated. – The **Snapshot** repository stores development versions with unstable functions in the development stage. – The **Release&Snapshot** repository stores release and snapshot versions simultaneously.

Item	Description
Open To	Set the repository permission scope. – If you select **Project**, the created repository belongs to the associated project. Repositories in a project use roles in the project for authentication. For details about repository permissions by role in a project, see **Configuring Project-Level Permissions**. – If you select **Tenant**, the created repository will belong to the tenant associated with it. Repositories within a tenant use tenant space role authentication. For details about repository permissions by role in a tenant, see **Configuring Repository-Level Permissions**.
Description	Enter up to 200 characters to describe the repository, such as its purpose or usage.

Step 4 Confirm the configurations and click **OK**.

After the virtual repository is created, you can view it in the repository list.

----End

4.2.4.2 Configuring a Virtual Repository

After a virtual repository is created, you can modify its configuration as required.

Procedure

Step 1 Use your Huawei Cloud account to access **self-hosted repos**.

Step 2 Select the created virtual repository from the repository list.

Step 3 Click **Settings** in the upper right corner of the repository.

Step 4 Set the parameters by referring to **Table 4-7**.

Table 4-7 Parameter description

Item	Description
Include Patterns	Specify the files to include in the operation. These patterns are usually used to filter in and select specific files or folders. If you configure this parameter, only artifacts that match the added path prefix can be operated on (including artifact uploads and downloads). You can click **+** to add multiple inclusion rules. The relationship between multiple inclusion rules is **OR**.

Item	Description
Exclude Patterns	Specify the files to exclude from the operation. These patterns are usually used to filter out and select specific files or folders. If you configure this parameter, artifacts that match the added path prefix cannot be operated on (including artifact uploads and downloads). You can click **+** to add multiple exclusion rules. The relationship between multiple exclusion rules is **OR**.
Open To	**Project**: Permissions are managed at the project level. **Tenant**: Permissions are managed at the tenant level. For details, see **User Permissions Management**.
Description	Describe the basic information and purpose of the repository.

Step 5 Click **Submit** to save the configurations.

Changes take effect immediately.

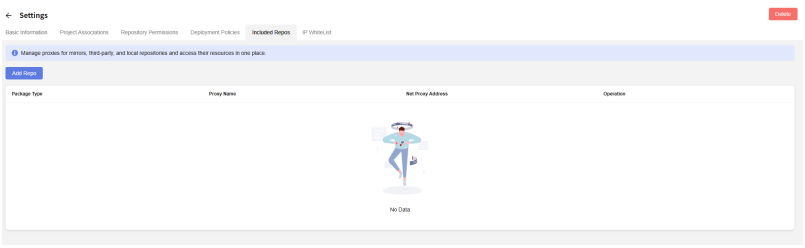
----End

4.2.4.3 Creating Virtual Repository Relationships

You can connect proxy sources to local third-party dependency repositories in the virtual repository and customize a proxy repository for open-source and third-party dependency repositories. After files are downloaded from a proxy repository, they can be cached to CodeArts Artifact, achieving a seamless, local-repository-like download experience.

Accessing the Included Repos Page

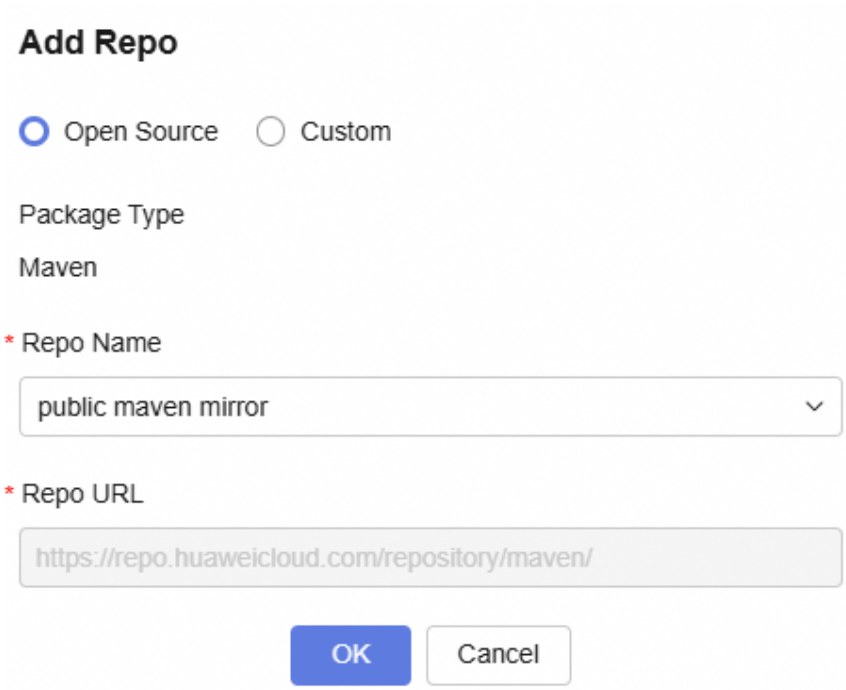
A virtual repository can aggregate and proxy third-party repositories and repositories of the same type. You can click **Settings** on the right of the virtual repository, select the **Included Repos** tab, and click **Add Repo**.



Aggregating and Proxying Open Sources

You can configure the public images that match the package type of the virtual repository. For example, for Maven repositories, you can use the **public maven mirror https://repo.huaweicloud.com/repository/maven/**.

Go to the Included Repos page of the virtual repository. In the **Add Repo** dialog box, select **Open Source** and set **Repo Name** to the public image of the target package type, as shown in the following figure.



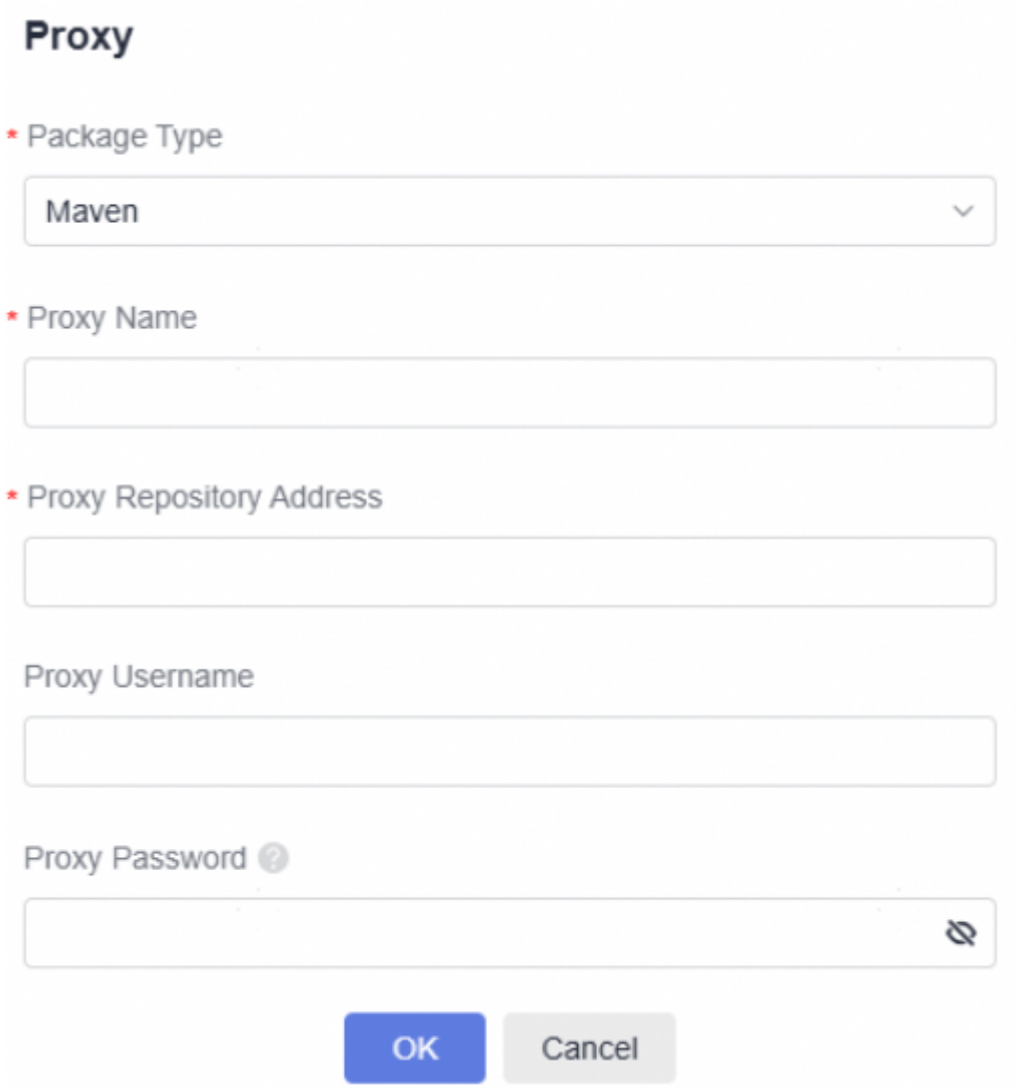
Aggregating and Proxying Custom Sources

You can aggregate and proxy local repositories from user-defined third-party sources or from CodeArts Artifact.

Aggregating and proxying third-party repositories

Step 1 On the CodeArts Artifact home page, click your avatar in the upper right corner and select **All Account Settings**.

Step 2 As the tenant administrator, choose **Mirror** > **Mirror** from the navigation pane. On the **Custom** tab, click **Create Proxy** and enter the required information to add a custom proxy.



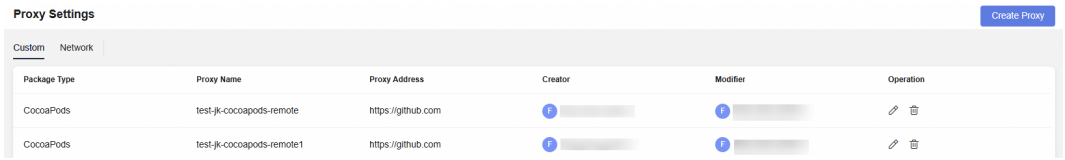
Fill in each proxy input field as follows:

Table 4-8 Proxy configuration information

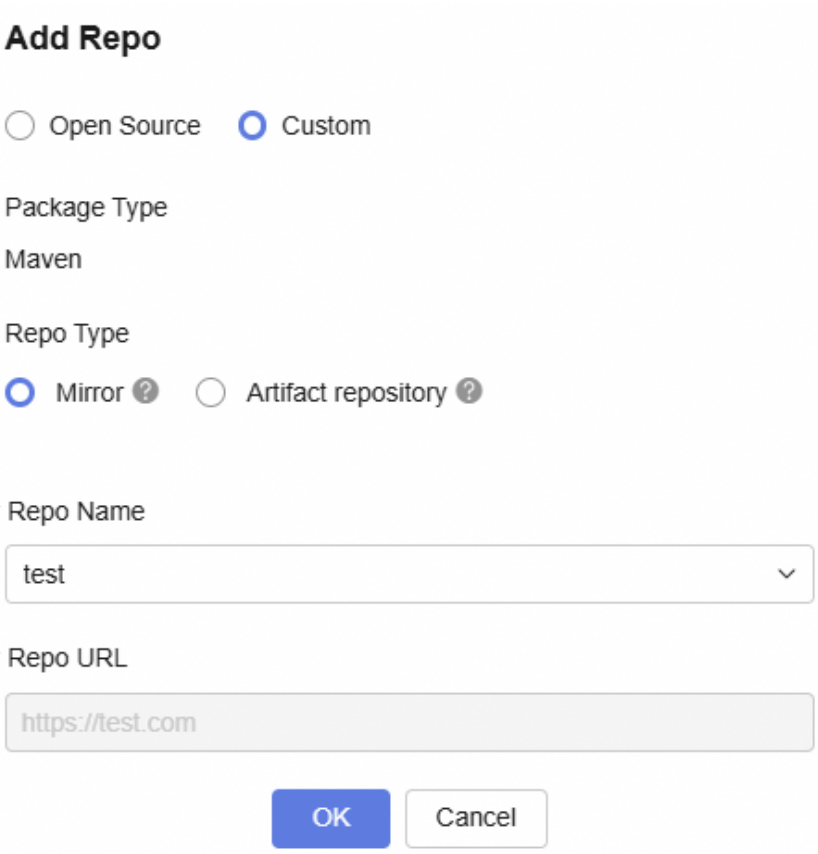
Item	Description
Package Type	Select a package type from the drop-down list, such as Maven, npm, or PyPI.
Repo Name	Name the repository of the proxy.
Repo URL	URL of the repository that requires proxy.
PyPI Index Proxy Address	This parameter is available only when **Package Type** is set to **PyPI**. It refers to the index site for Python software packages, also known as the python package index (PyPI). Example: **https://mirrors.tools.****.com/pypi**

Item	Description
Proxy Username	If the proxy requires authentication, enter the username.
Proxy Password	If the proxy requires authentication, enter the password. If you do not set the proxy password, the password set last time is used by default.

Step 3 After entering the information, click **OK**. The custom proxy is displayed in the list. You can then click or to modify or remove it.



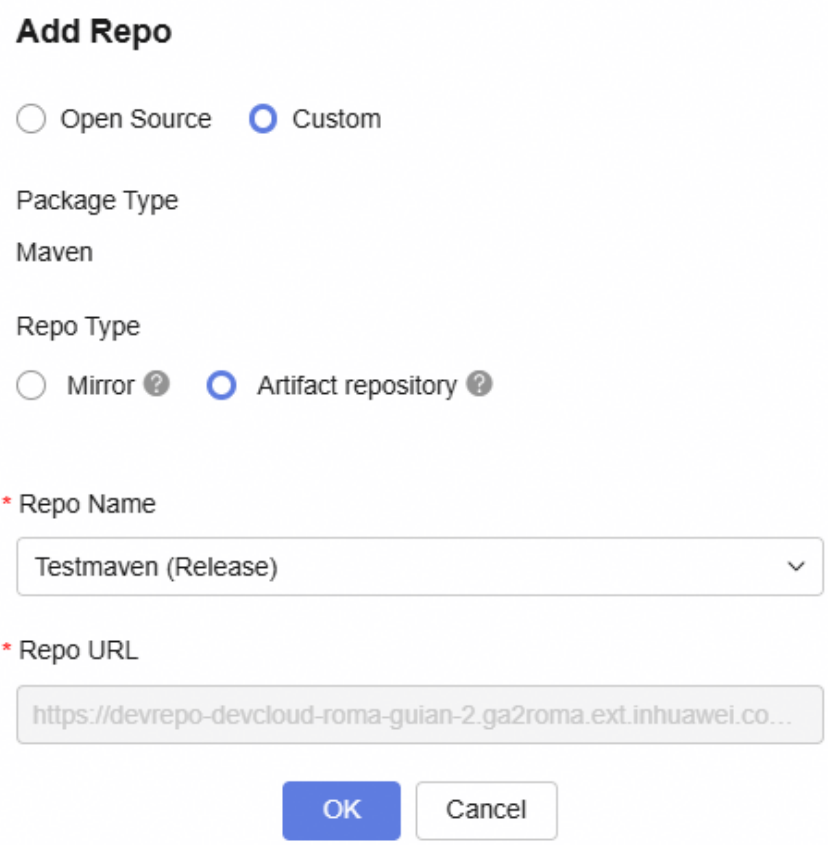
Step 4 Go to the Included Repos page of the virtual repository, select **Custom**, and set **Repo Type** to **Mirror**. Then select the repository to proxy.



----End

Aggregating a local repository

Go to the Included Repos page of the virtual repository, select **Custom**, and set **Repo Type** to **Artifact repository**. Then select the local repository to aggregate.



4.3 Configuring Repository Permissions

By default, new users in self-hosted repos have no permissions. You need to add them to a project and assign roles to them. Each role has different permissions. You can view role permissions on the CodeArts Artifact permissions page and edit user permissions as needed.

Constraints

- By default, project administrators have all permissions and their permission scope cannot be modified.
- **Custom roles** created in CodeArts Artifact do not have preset permissions. You can contact the project administrator to configure roles and permissions on corresponding resources.
- By default, the project administrator, project manager, and test manager can assign permissions for other roles in self-hosted repos. If other roles can assign permissions, they can continue to configure permissions for other roles in self-hosted repos.

Prerequisites

- You have created a **CodeArts project**.
- You have added users to the CodeArts project and assigned roles to them. For details, see **Adding Project Members**.

- To assign permissions to other roles in the self-hosted repo, you must have the **Privilege Config** permission. By default, the project administrator, project manager, and test manager roles have this permission.

Project-Level Permissions

Step 1 Log in as a user with the **Privilege Config** permission, and access the **self-hosted repo**.

Step 2 Choose **Settings** > **General** > **Permissions** from the navigation pane. The **Permissions** page is displayed. Different project roles have different operation permissions.

Step 3 In the **Roles** area, click the role you want to configure permissions, select **CodeArts Artifact** on the right, and click **Edit** at the bottom of the page. In the displayed page, select or deselect the required permissions.

Table 4-9 Project-level permissions

Role/ Operation	Create Repository	Edit Repository	Delete Repository	Restore	Permanently Delete	Restore All	Clear All	Upload Package	Download/ View Package	Delete/ Redeploy Uploaded Package
Project manager	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Product manager	✗	✓	✗	✓	✓	✗	✗	✓	✓	✗
Test manager	✗	✓	✗	✓	✗	✓	✗	✓	✓	✗
Operation manager	✗	✓	✗	✓	✓	✓	✓	✓	✓	✗
System engineer	✗	✓	✗	✓	✓	✗	✗	✓	✓	✗
Committer	✗	✓	✗	✓	✓	✗	✗	✓	✓	✗

Role/ Operation	Create Repository	Edit Repository	Delete Repository	Restore	Permanently Delete	Restore All	Clear All	Upload Package	Download/ View Package	Delete/ Redeploy Uploaded Package
Developer	✗	✓	✗	✓	✓	✗	✓	✓	✓	✗
Tester	✗	✗	✗	✓	✗	✗	✗	✓	✓	✗
Participant	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗
Viewer	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗
CI/CD engineer	✓	✓	✗	✗	✗	✗	✗	✓	✓	✓
Project administrator	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

- Tenant administrators (IAM user with the **Tenant Administrator** permissions) can manage all projects of a tenant. Project creators who are not tenant administrators have the permissions listed in the preceding table.
- IAM users created without being added to any groups do not have permissions. Administrators can assign permissions to these users on the IAM console. Then users can use cloud resources in the account based on the assigned permissions.
- Custom roles do not have preset permissions. You can contact the administrator to grant permissions for the resources needed for your role.

Step 4 Click **Save**.

----End

Repository-Level Permissions

CodeArts Artifact allows you to configure permissions on all self-hosted repos in a project. For details, see **Project-Level Permissions**.

You can also configure permissions for a single self-hosted repo.

To add or remove permissions for self-hosted repo members, perform the following steps:

Step 1 Access self-hosted repos using your Huawei Cloud account and select the target repository from the repository list.

Step 2 Click **Settings** on the right of the page.

Step 3 Click the **Repository Permissions** tab. The roles in the current project are displayed.

Step 4 In the **Roles** area, select or deselect permissions of the target role, and click **Save**.

- By default, created self-hosted repos inherit the **Permissions** in **Settings** in the project. Any changes made to these role permissions in **Permissions** will be synced to the repo's permissions.
- If you do not change the repository permission of a role in the self-hosted repo, any changes made on the **Permissions** page will be synchronized to the repository permission of the self-hosted repo as well.
- If you change the repository permission of a role directly in the self-hosted repo, any changes made on the **Permissions** page will not be synchronized to the repository permission of the self-hosted repo. You need to change the permission of the role in the repo itself.

----End

Tenant-Level Permissions

Step 1 Access self-hosted repos using your Huawei Cloud account and select the target repository from the repository list.

Step 2 Click **Tenant** in the lower-left corner to go to the self-hosted repo of the tenant.

Step 3 Select a self-hosted repo, click **Settings** on the right, and find the **Repository Permissions** tab.

Different roles in the tenant space have different operation permissions, as shown in **Table 4-10**.

Table 4-10 Tenant-level permissions

Permission Description									
Role/ Operation	**Edit Repository**	**Delete Repository**	**Restore Repository**	**Physic Delete**	**Upload**	**Download/View**	**Delete/ Redeploy**	**Import**	**Export**
Tenant space owner	√	√	√	√	√	√	√	√	√
Tenant space admin	√	√	√	√	√	√	√	√	√
Tenant space user	√	×	√	√	√	√	×	√	√

NOTE

- The operation permissions of the tenant space owner cannot be modified.
- For details about how to add members to a tenant space, see **Adding Project Members**.

----End

4.4 Managing Repositories

4.4.1 Viewing, Configuring, and Deleting a Repository

To use repositories in CodeArts Artifact, access the target repository to manage settings, connect to environments, configure rules, and clean up resources. You can also view repository details, obtain repository URLs, and manage packages.

Viewing a Repository

Step 1 Go to the CodeArts homepage.

1. Log in to the **CodeArts console**, click , and select a region where you have enabled CodeArts.
2. Click **Go to Workspace**.

   If your account uses the old billing mode (see **Old Billing Modes**), click **Access Service**.

Log in to the **CodeArts console**, and click **Access Service**.

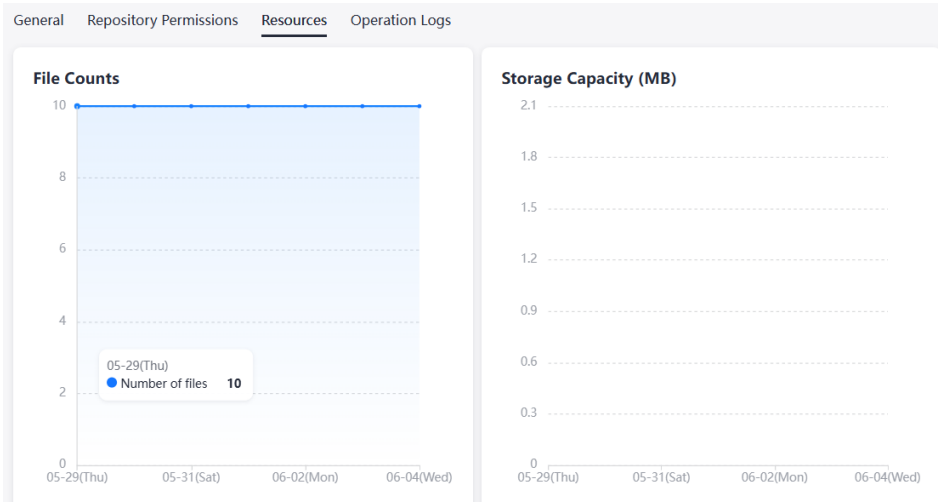
Step 2 Click a project card to access the project. (If no project is available, **create one**.)

Step 3 Choose **Artifact** > **Self-hosted Repos** from the navigation pane.

Step 4 In the left pane, click the desired repository name to go to its details page.

General, **Repository Permissions**, **Resources**, and **Operation Logs** tabs are displayed.

- **General**: displays the repository name, repository type, repository URL, relative path, creator, creation time, modifier, modification time, artifact count, and artifact size.
- **Repository Permissions**: displays the permission scope (repository and package) of the repository and the corresponding permissions. This page is view-only. To edit permissions, choose **Settings** > **Repository Permissions**. For details, see **Repository-Level Permissions**.
- **Resources**: displays dynamic statistics on **File Counts** and **Storage Capacity (Byte)** for uploaded artifacts in the repository.



- **Operation Logs**: displays the operation history of uploading, deleting, and restoring data from the recycle bin in the repository.

----End

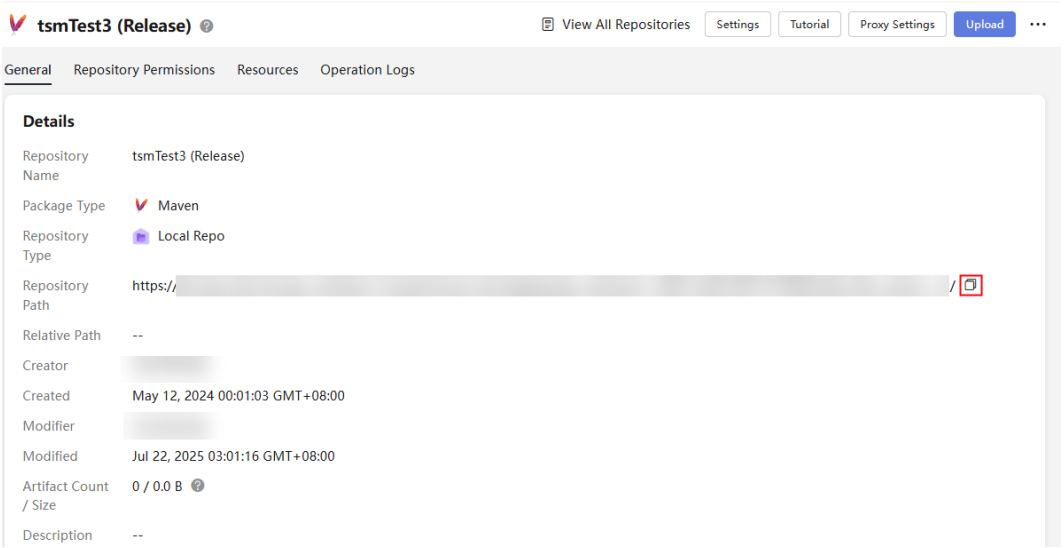
Obtaining Repository URLs

Each time you create a repository, a repository URL is generated for it. You will use this URL to connect the repository to your local development environment. Perform the following operations to obtain it.

Step 1 Access self-hosted repos using your Huawei Cloud account.

Step 2 In the left pane, click the name of the target repository.

Step 3 The repository URL is displayed on the **General** page. Click to obtain the URL.



----End

Configuring a Repository

Step 1 Click a project card to access the project. (If no project is available, **create one**.)

Step 2 Choose **Artifact** > **Self-hosted Repos** from the navigation pane.

Step 3 In the left pane, click the name of the target repository to be edited.

Step 4 Click **Settings** on the right of the page. The **Basic Information** tab is displayed by default.

Step 5 The repository name, package type, project, and version policy cannot be edited. But you can edit the description and include and exclude patterns as required.

On the **Basic Information** page, you can click + to add path rules for Maven, npm, Go, PyPI, RPM, Docker, CocoaPods, OHPM, and Conan repositories.

To delete path rules, click .

Step 6 Click **Submit** to save the settings.

----End

Deleting a Repository

You may delete repositories, and they will be moved to the recycle bin.

Step 1 Go to the self-hosted repo page. In the left pane, click the name of the target repository.

Step 2 Click **Settings** on the right of the page to view the basic information about the repository.

Step 3 Click **Delete** on the right of the page. Check that the deleted repository is no longer displayed in the repository list in the left pane.

----End

4.4.2 Configuring Deployment Policies

You can set policies to control how artifacts are uploaded to your repository, such as whether new artifacts can redeploy existing ones in the same path.

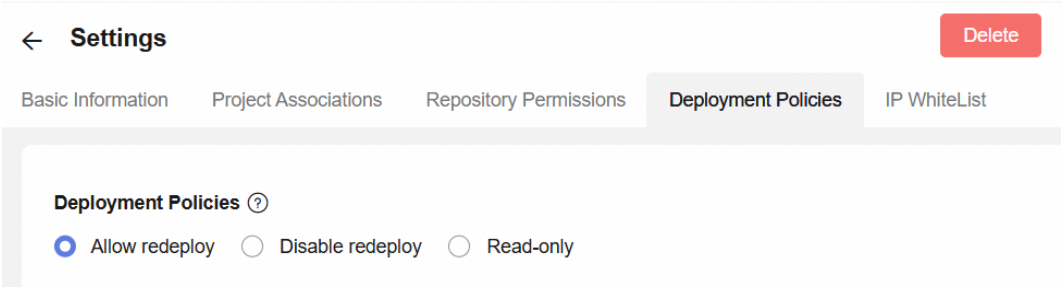
Procedure

Step 1 **Access self-hosted repos** using your Huawei Cloud account.

Step 2 In the left pane, click the name of the target repository.

Step 3 Click **Settings** on the right of the page and click the **Deployment Policies** tab.

Figure 4-1 Configuring deployment policies



- **Allow redeploy** (selected by default): Artifacts in the same path can be uploaded, replacing the original package.
- **Disable redeploy**: Artifacts in the same path cannot be uploaded.
- **Read-only**: Artifacts cannot be uploaded, updated, or deleted. You can download an uploaded artifact.

Step 4 Settings are automatically saved by the system.

----End

4.4.3 Configuring Clearing Policies for a Maven Repository

The clearing policy allows you to clear artifacts in batches. This helps optimize storage, keep artifacts organized, and ensure they are transferred correctly during development, testing, deployment, and release.

Background

A Maven snapshot is a special version that reflects the current state of ongoing development and differs from regular versions. During each build, Maven checks for new snapshots in the remote repository. You can set limits on how many snapshots to retain, as well as automatically clear out expired snapshots.

Snapshot Clearing Policy

Step 1 Go to the CodeArts homepage.

1. Log in to the **CodeArts console**, click , and select a region where you have enabled CodeArts.
2. Click **Go to Workspace**.

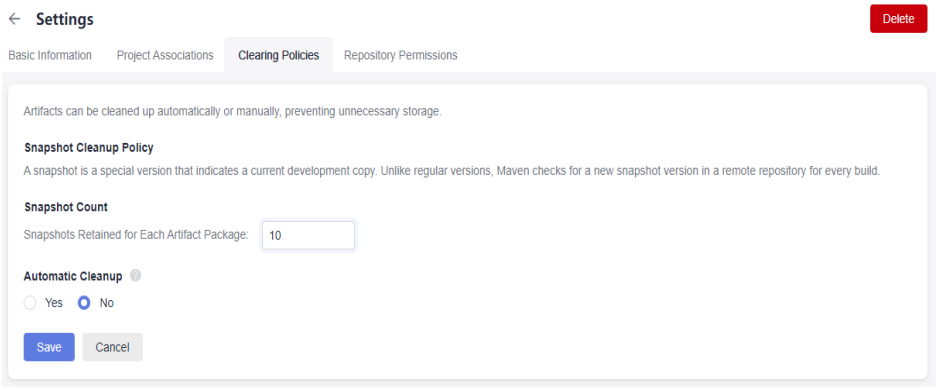
   If your account uses the old billing mode (see **Old Billing Modes**), click **Access Service**.

Log in to the **CodeArts console**, and click **Access Service**.

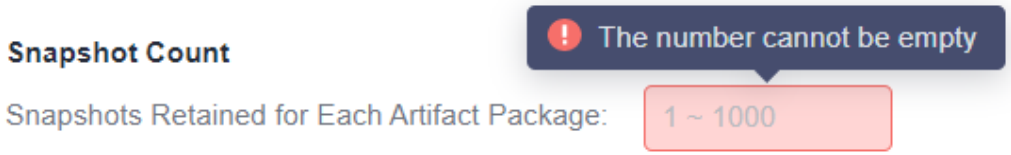
Step 2 Click a project card to access the project and choose **Artifact** > **Self-hosted Repos** from the navigation pane.

Step 3 Select a Maven repository (**Snapshot**) from the list on the left and click **Settings** in the upper right corner of the page.

Step 4 Click the **Clearing Policies** tab.



Step 5 Set the maximum number of **Snapshot Count**. The value ranges from 1 to 1,000.



When the number of retained snapshots exceeds the set limit, the oldest snapshot is replaced by the latest version.

Step 6 Enable automatic cleanup (**No** by default). Click **Yes** and enter the number of days. Snapshots older than the specified number of days will be automatically cleaned up.

The automatic cleanup time must be set between 1 and 100 days.

Step 7 Click **Save** to complete the configuration.

----End

4.4.4 Associating Maven Repositories with Projects

After the Maven repository is associated with multiple projects, you can select this Maven repository in the build step of the build task in the project to store the build product to the repository.

Procedure

Step 1 Access self-hosted repos using your Huawei Cloud account.

Step 2 Click the target Maven repository in the repository list on the left.

Step 3 Click **Settings** on the right of the page, and select **Project Associations**.

Step 4 Find the target project to be associated in the list and click in the **Operation** column of the corresponding row.

Step 5 In the displayed dialog box, select the repository name, and click **OK**.

When the system displays a message indicating that the operation is successful, the value of **Associated Repositories** for the project is updated based on the number of selected repositories.

----End

Helpful Links

In CodeArts Build, you need to upload build products to self-hosted repos. For details, see **Building with Maven**.

4.4.5 Configuring the Encryption Mode of a Maven Repository

Dependency Management Tool: Maven

Prerequisites

- Ensure that you have installed **JDK** and Maven.
- Download the configuration file, or modify the Maven **settings.xml** file in the **conf** or **.m2** directory as follows.

Procedure

Step 1 Create a master password.

```
mvn --encrypt-master-password <password>
```

An encrypted password is generated, for example, **{jSMOWnoPFgsHVpMvz5VrIt5kRbzGpI8u+9EF1iFQyJQ=}**.

Save it to the **${user.home}/.m2/settings-security.xml** file. For example:

```
<settingsSecurity>
<master>{jSMOWnoPFgsHVpMvz5VrIt5kRbzGpI8u+9EF1iFQyJQ=}</master>
</settingsSecurity>
```

Step 2 Encrypt the password.

```
mvn --encrypt-password <password>
```

An encrypted password is generated, for example: **{COQLCE6DU6GtcS5P=}**.

Save it to the **settings.xml** file. For example:

```
<settingsSecurity>
<master>{jSMOWnoPFgsHVpMvz5VrIt5kRbzGpI8u+9EF1iFQyJQ=}</master>
</settingsSecurity>
```

----End

Dependency Management Tool: Gradle

Prerequisites: You have installed **JDK** and **Gradle**.

Step 1 Add the **nu.studer.credentials** plug-in to the **build.gradle** file in the Gradle project.

```
plugins{
 id 'nu.studer.credentials' version '2.1'
}
```

Step 2 Create an encryption password.

```
gradle addCredentials -PcredentialsKey=accountPassword -PcredentialsValue="**"
```

The following encrypted password is generated: **cVe******kg\=\=**.

By default, the data is stored in the **GRADLE_USER_HOME/gradle.encrypted.properties** file. For example:

```
accountPassword=cVe******kg\=\=
```

Step 3 Configure the repository with an encrypted password in the **build.gradle** file.

```
def password = credentials.accountPassword
            repositories {
              maven {
                credentials {
                  username 'cn-
north-1_f9e40463c23845438ca9efd3a7ec854e_67902fb51ded48c2868310a07e1569e7'
                  password password
                }
                url 'https://devrepo.****.com/artgalaxy/cn-
north-1_f9e40463c23845438ca9efd3a7ec854e_maven_1_7/'
              }
            }
```

----End

4.5 Uploading/Downloading Packages on the Self-Hosted Repo Page

Only repository administrators and developers can upload and download private packages. You can set repository roles on the **Repository Permissions** page.

Uploading a Package

Step 1 Click a project card to access the project. (If no project is available, **create one**.)

Step 2 Choose **Artifact** > **Self-hosted Repos** from the navigation pane.

Step 3 In the left pane, click the target repository to which the private package is to be uploaded.

Step 4 Click **Upload** on the right of the page.

Step 5 In the **Upload** dialog box, specify parameters, select the file, and click **Upload**. Detailed configuration for each package type is described below.

- **The Conan package cannot be uploaded on the page.**
- The maximum file size for uploads is 100 MB for Maven, npm, PyPI, RPM, and Debian types, and 20 MB for NuGet.
- Do not upload files containing sensitive information such as plaintext accounts and passwords to self-hosted repos.

----End

Uploading a Maven Artifact

Maven artifact

- **Project Object Model (POM)** is the basic working unit of a Maven project. It is an XML file that contains basic project information to describe how to build a project and declare project dependencies. When a build task is run, Maven searches for the POM in the current directory, reads its content, obtains the required configuration information, and builds the target package.
- **Maven coordinates**: X, Y, and Z are used to uniquely identify a point in the three-dimensional space. In Maven, GAV is used to identify a unique package. It stands for **groupId**, **artifactId**, and **version**. **groupId** indicates a company or organization. For example, Maven core components are in the **org.apache.maven** organization. **artifactId** indicates the name of the package. **version** indicates the version of the package.
- **Maven dependencies** are crucial for POM files. The building and running of most projects depend on the dependency on other components. Add the dependency list to your POM file. If the App component depends on the App-Core and App-Data components, the configuration is as follows:

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
  http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.companyname.groupname</groupId>
    <artifactId>App</artifactId>
    <version>1.0</version>
    <packaging>jar</packaging>
    <dependencies>
      <dependency>
        <groupId>com.companyname.groupname</groupId>
        <artifactId>App-Core</artifactId>
        <version>1.0</version>
      </dependency>
    </dependencies>
    <dependencies>
      <dependency>
        <groupId>com.companyname.groupname</groupId>
        <artifactId>App-Data</artifactId>
        <version>1.0</version>
      </dependency>
    </dependencies>
</project>
```

Procedure

POM and GAV upload modes are supported.

Step 1 **Access self-hosted repos** using your Huawei Cloud account.

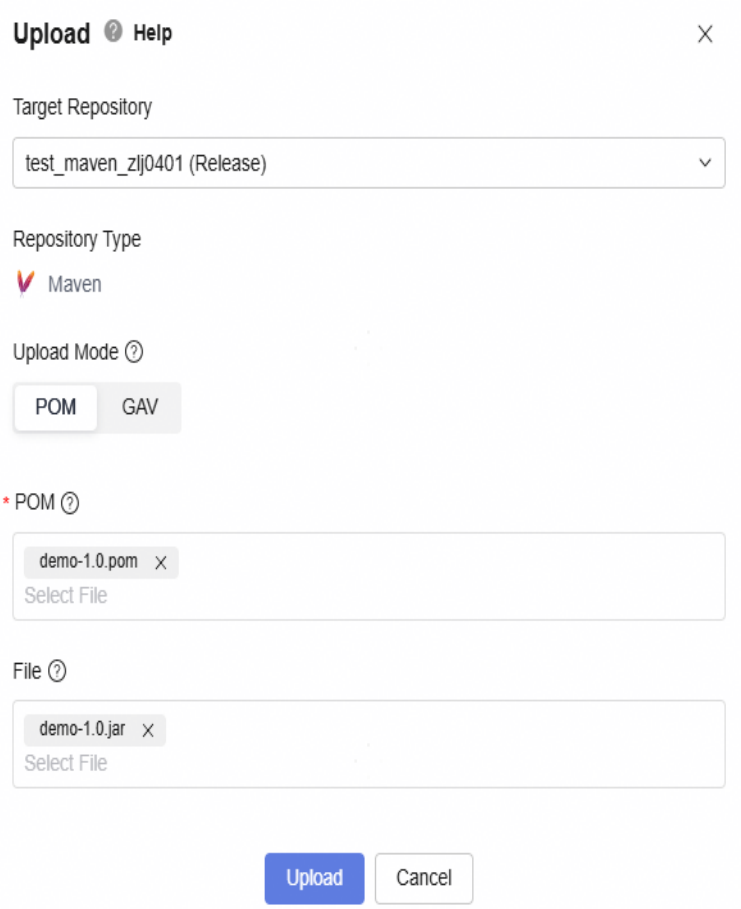
Step 2 In the left pane, click the target Maven repository to which the private package is to be uploaded.

Step 3 Click **Upload** on the right of the page.

Step 4 In the displayed dialog box, select an upload mode and set the parameters, as shown in the following table.

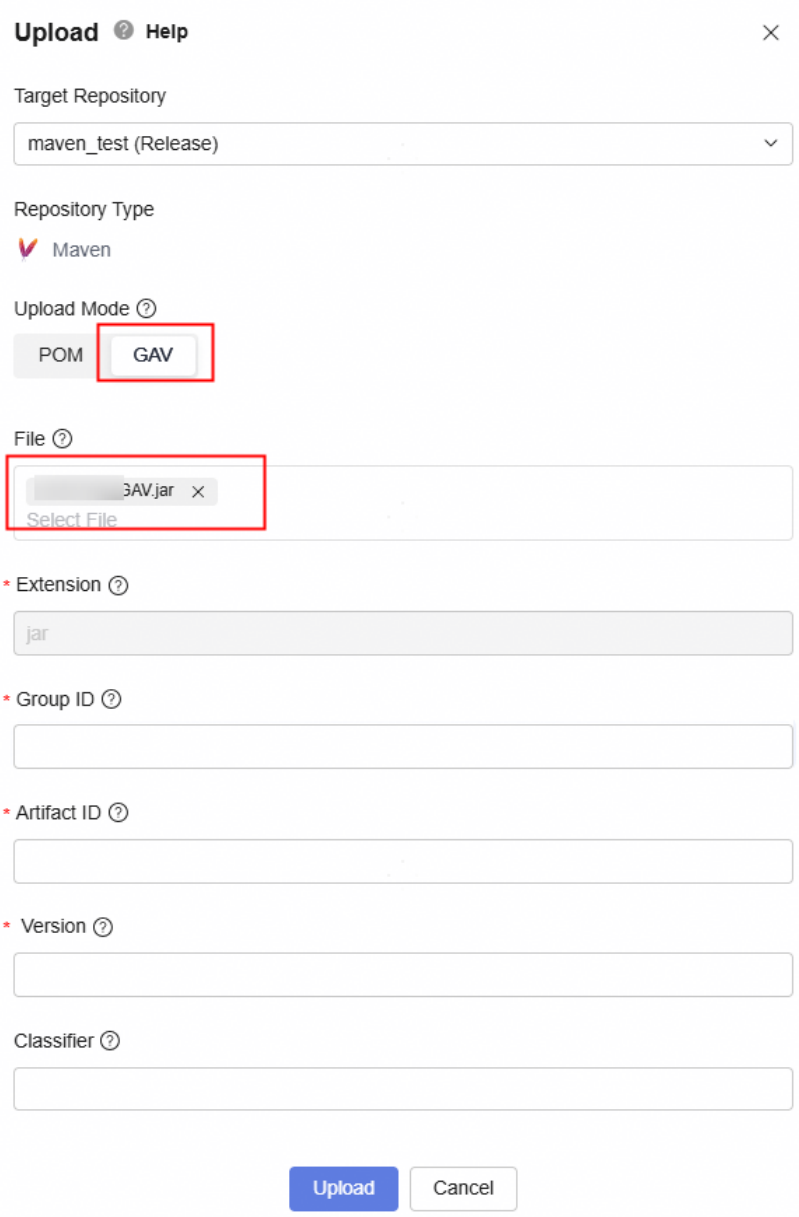
Table 4-11 Parameters for uploading a Maven artifact

Upload Mode	Description

POM	GAV parameters are obtained from the POM file. The system retains transitive dependencies of components.
	In POM mode, you can either upload just the POM file or both the POM file and its related components. The uploaded file must match the **artifactId** and **version** values specified in the POM. As shown in the following figure, if the **artifactId** value is **demo** and the **version** value is **1.0** in the POM, then the uploaded file must be named **demo-1.0.jar**.
	Figure 4-2 Uploading a package in POM mode
	 
	The POM file structure is as follows:
	`<project>`
	`  <modelVersion>4.0.0</modelVersion>`
	`  <groupId>demo</groupId>`
	`  <artifactId>demo</artifactId>`
	`  <version>1.0</version>`
	`</project>`

Upload Mode	Description
	NOTE The **modelVersion** tag must exist and the value must be **4.0.0**, indicating that **Maven2** is used. If you upload files in both the **POM** and **File** area, the **artifactId** and **version** values in **POM** must match the file name in **File**. For example, if the **artifactId** value is **demo** and the **version** value is **1.0** in **POM**, then the uploaded file must be named **demo-1.0** in **File**.

Upload Mode	Description
GAV	GAV, short for **Group ID**, **Artifact ID**, and **Version**, is the unique identifier of a JAR package. In this mode, GAV parameters are manually specified. The system automatically generates a POM file without any transitive dependency. In the GAV mode, the **Group ID**, **Artifact ID**, and **Version** parameters must be manually specified and they determine the name of the file to upload. **Extension** indicates the packaging type and determines the file type to be uploaded. **Classifier** is used to differentiate artifacts that are built from the same POM but contain different contents. This field is optional. It can contain letters, digits, underscores (_), hyphens (-), and dots (.). If you specify this field, it will be appended to the file name. Common usage scenarios • **Differentiate versions by name**, such as **demo-1.0-jdk13.jar** and **demo-1.0-jdk15.jar**. • **Differentiate usage by name**, such as **demo-1.0-javadoc.jar** and **demo-1.0-sources.jar**.

Upload Mode	Description
	Figure 4-3 Uploading a package in GAV mode  • **Extension**: packaging type. The default value is the same as the file name extension. • **Group ID**: project name, which uniquely identifies a project, for example, **org.apache.commons**. It can contain up to 128 characters. Only letters, digits, underscores (_), hyphens (-), and periods (.) are supported. • **Artifact ID**: component name, for example, **commons-math**. It can contain up to 128 characters. Only letters, digits, underscores (_), hyphens (-), and periods (.) are supported.

Upload Mode	Description
	• **Version**: component version, for example, **1.0**. It can contain up to 32 characters. Only letters, digits, underscores (_), hyphens (-), and periods (.) are supported. • **Classifier** is used to differentiate artifacts that are built from the same POM but contain different contents. Common values include **sources**, **javadoc**, **tests**, and **jdk**. If specified, the value is appended to the file name. Setting **Classifier** disables automatic POM file generation.

Step 5 Click **Upload** after the configuration is complete.

View the uploaded package in the repository list.

----End

Uploading an npm Package

npm package

Node Package Manager (npm) is a JavaScript package management tool. An npm package is the item managed by npm, and the npm registry is used to store and manage these packages.

The npm package consists of a structure and file description.

- Package structure organizes various files in a package, such as source code files and resource files.
- Description file describes package information. Example: **package.json**, **bin**, and **lib** files

The **package.json** file in the package is a description file of a project or module package. It contains information such as the name, description, version, and author. The **npm install** command downloads all dependent modules based on this file.

An example of the **package.json** file is as follows:

```
{
  "name": "third_use",            //Package name
  "version": "0.0.1",           //Version number
"description": "this is a test project", // Description
  "main": "index.js",             //Entry file
  "scripts": {                  //Script command
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [                   //Keyword
    "show"
  ],
  "author": "f",                //Developer name
  "license": "ISC",               //License agreement
  "dependencies": {               //Project production dependencies
    "jquery": "^3.6.0",
    "mysql": "^2.18.1"
  },
  "devDependencies": {            //Project development dependencies
    "less": "^4.1.2",
    "sass": "^1.45.0"
```



```
  }
}
```

The **name** and **version** are the most important fields and must exist. Otherwise, the current package cannot be installed. The two attributes together form the unique identifier of an **npm** package.

name indicates the name of the package. The first part of the name, such as **@scope**, is used as the namespace. The other part is **name**. Generally, you can search with the **name** field to install and use the required package.

```
{
  "name": "@scope/name"
}
```

version indicates the version of the package, which is in the x.y.z format.

```
{
  "version": "1.0.0"
}
```

Procedure

Step 1 **Access self-hosted repos** using your Huawei Cloud account.

Step 2 In the left pane, click the target npm repository to which the private package is to be uploaded.

Step 3 npm packages in .tgz format can be uploaded to self-hosted repos. When uploading packages, you need to set the following parameters.

Item	Description
Package Name	The value must be the same as that of **name** in the **package.json** file. When uploading a package, ensure that the package name starts with a path in the path list added during repository creation. For details, see **Table 4-1** in the help guide. Example: The path **@test** is added when you create an npm registry. When uploading the package to the registry, make sure that the **PackageName** starts with **@test**. If a path outside the path list is used, for example, **@npm**, the upload will fail. After the upload is successful, you can find the package in .tgz format in the repository list and the corresponding metadata is generated in the **.npm** directory.
Version	The value must be the same as that of **version** in the **package.json** file.
File	File name must end with .tgz.

Step 4 Click **Upload** after the configuration is complete.

After the upload is successful, you can find the package in .tgz format in the repository list and the corresponding metadata is generated in the **.npm** directory.

----End

Uploading an RPM Package

RPM package

- Red Hat Package Manager (RPM), developed by Red Hat, is used by many Linux distributions. It is a software management system that installs software on Linux in database recording mode.
- You are advised to package and name the RPM binary file according to the following rules:

Software name-*Main version number of the software*.*Minor version number of the software*.*Software revision number*-*Number of software compilation times*.*Hardware platform suitable for the software*.**rpm**

For example, **hello-0.17.2-54.x86_64.rpm**. **hello** is the software name, **0** is the major version number of the software, **17** is the minor version number, **2** is the revision number, **54** is the number of times that the software is compiled, and **x86_64** is the hardware platform suitable for the software.

Software Name	Major Version	Minor Version	Revision No.	Compilation Times	Applicable Hardware Platform
hello	0	17	2	54	x86_64

Procedure

Step 1 **Access self-hosted repos** using your Huawei Cloud account.

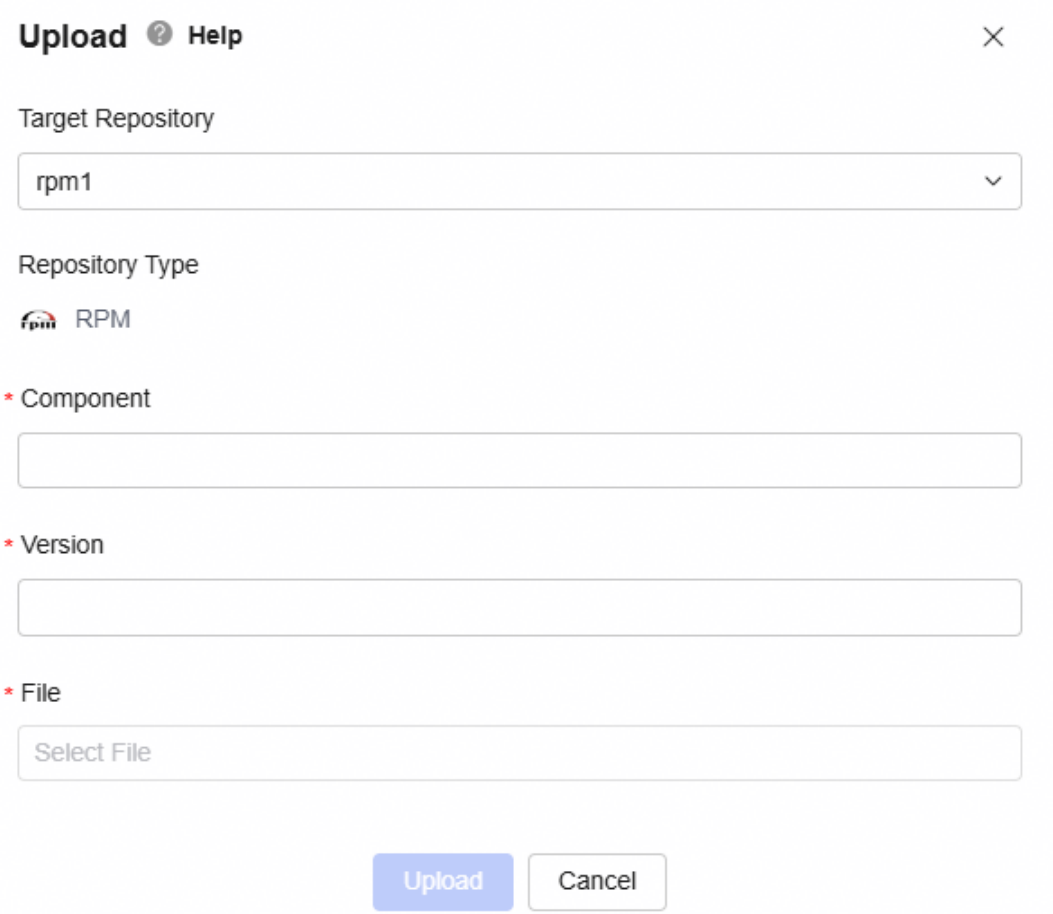
Step 2 In the left pane, click the target RPM repository to which the private package is to be uploaded.

Step 3 Click **Upload** on the right of the page.

Step 4 In the displayed dialog box, set the parameters as described in the following table.

Item	Description
Component	Component name
Version	Version of the RPM binary package
File	The RPM file to be uploaded

Figure 4-4 Uploading an RPM package



Step 5 Click **Upload** after the configuration is complete.

After the upload is successful, you can find the RPM binary package in the repository list and the corresponding metadata **repodata** directory is generated in the package name directory. You can use Yum to install the package.

----End

Uploading a PyPI Package

PyPI package

You are advised to go to the project directory (which must contain the **setup.py** configuration file) and run the following command to compress the package to be uploaded into a wheel (.whl) installation package. By default, this package is generated in the **dist** directory of your project directory. Note that the Python package management tool **pip** supports only wheel installation packages.

```
python setup.py sdist bdist_wheel
```

Procedure

Step 1 Access self-hosted repos using your Huawei Cloud account.

Step 2 In the left pane, click the target PyPI repository to which the private package is to be uploaded.

Step 3 Click **Upload** on the right of the page and set the parameters described in the following table.

Item	Description
Package Name	The value must be the same as that of **name** in the **setup.py** file.
Version	The value must be the same as that of **version** in the **setup.py** file.
File	The PyPI file to be uploaded.

Step 4 Click **Upload** after the configuration is complete.

After the upload is successful, you can find the installation package in **.whl** format in the repository package list. In addition, the corresponding metadata is generated in the **.pypi** directory, which can be used for pip installation.

----End

Uploading a Go Package

Go package

Go, also known as Golang, is a programming language developed by Google. Golang 1.11 or later supports modular package management. A module is a unit for source code exchange and versioning in Go. A MOD file identifies and manages a module. A ZIP file is a source code package. Go modules are divided into two types: v2.0 and later, and earlier than v2.0, each with different management rules.

Procedure

Step 1 **Access self-hosted repos** using your Huawei Cloud account.

Step 2 In the left pane, click the target Go repository to which the package is to be uploaded.

Step 3 Click **Upload** on the right of the page.

Step 4 To upload a Go package, you need to upload both a ZIP file and a MOD file and set the following parameters.

Step	Item	Description
Step 1: Upload a ZIP file	Zip Path	Complete path of the ZIP file. Valid path formats are: • **Versions earlier than v2.0**: *{moduleName}*/@v/*{version}*.zip • **Versions later than v2.0**: – If the ZIP file contains **go.mod** and the path ends with **/vN**, the file path format is: *{moduleName}*/vX/@v/vX.X.X.zip – If the ZIP file does not contain **go.mod** or the first line in **go.mod** does not end with **/vN**, the file path format is: *{moduleName}*/@v/vX.X.X+incompatible.zip
	Zip File	Directory structure of the ZIP file. Valid directory structure formats are: • **Versions earlier than v2.0**: *{moduleName}*@*{version}* • **Versions later than v2.0**: – If the ZIP file contains **go.mod** and the path ends with **/vN**, the directory structure format is: *{moduleName}*/vX@*{version}* – If the ZIP file does not contain **go.mod** or the first line in **go.mod** does not end with **/vN**, the directory structure format is: *{moduleName}*@*{version}*+incompatible.
Step 2: Upload a MOD file	Mod Path	Complete path of the MOD file. Valid path formats are: • **Versions earlier than v2.0**: *{moduleName}*/@v/*{version}*.mod • **Versions later than v2.0**: – If the ZIP file contains **go.mod** and the path ends with **/vN**, the file path format is: *{moduleName}*/vX/@v/vX.X.X.mod – If the ZIP file does not contain **go.mod** or the first line in **go.mod** does not end with **/vN**, the file path format is: *{moduleName}*/@v/vX.X.X+incompatible.mod.

Step	Item	Description
	Mod File	MOD file content. Valid content formats are: • **Versions earlier than v2.0**: module *{moduleName}* • **Versions later than v2.0**: – If the ZIP file contains **go.mod** and the path ends with **/vN**, the content format is: module *{moduleName}*/vX – If the ZIP file does not contain **go.mod** or the first line in **go.mod** does not end with **/vN**, the content format is: module *{moduleName}*

Step 5 Click **Upload** after the configuration is complete.

View the uploaded package in the repository list.

----End

Uploading a Debian Package

Debian package

Debian packages are packages released through the package management system, such as Advanced Package Tool (APT), for the Debian GNU/Linux distribution. These packages contain all files required to install, configure, and use the software.

Procedure

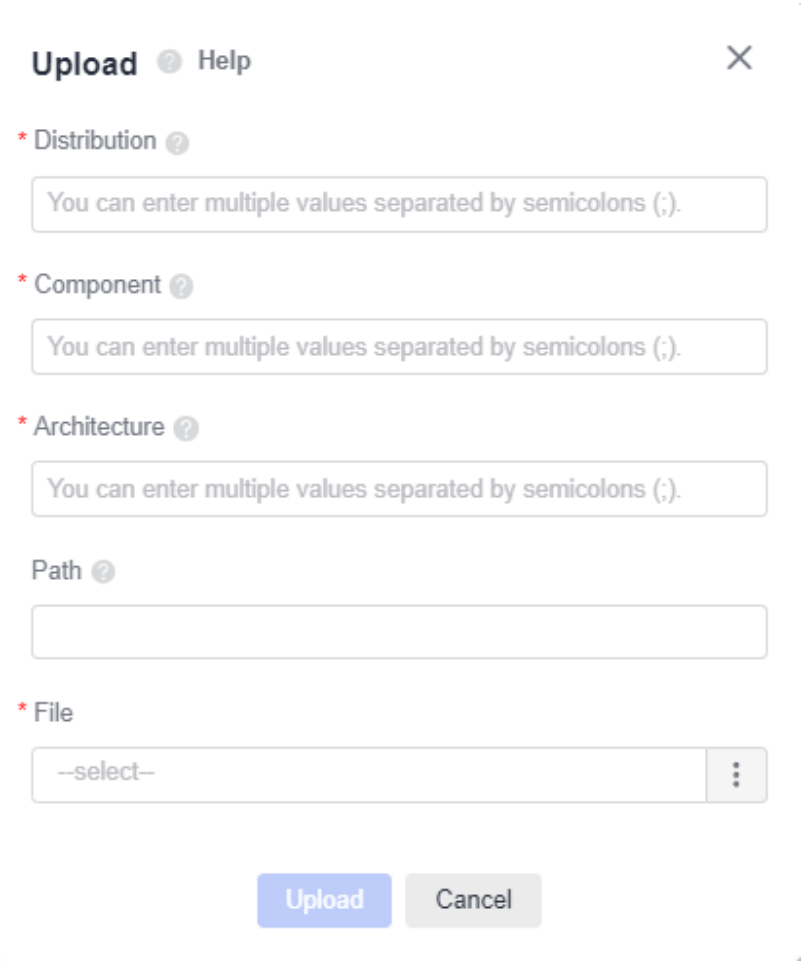
Step 1 **Access self-hosted repos** using your Huawei Cloud account.

Step 2 In the left pane, click the target Debian repository to which the private package is to be uploaded.

Step 3 Click **Upload** on the right of the page.

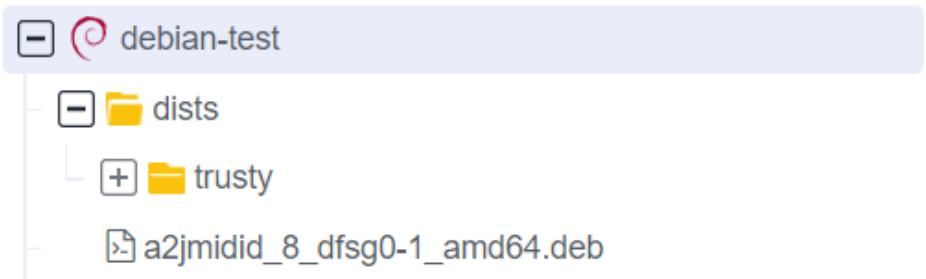
Step 4 When uploading a Debian package, you need to set the following parameters:

Item	Description
Distribution	Release version of the package
Component	Name of the package
Architecture	Package architecture
Path	Path for storing the package. By default, the package is uploaded to the root path.
File	Local storage path of the package to be uploaded



Step 5 Click **Upload** after the configuration is complete.

After the upload is successful, you can find the installation package in **.deb** format in the repository list. In addition, the corresponding metadata is generated in the **dists** directory, which can be used for Debian installation.



----End

Uploading a NuGet Package

NuGet package

A NuGet package is a single ZIP file with a **.nupkg** extension. As a shareable unit of code, developers can publish it to a dedicated server to share it with other team members.

CodeArts Artifact creates a NuGet repository to host and manage NuGet packages.

You are advised to package and name the NuGet file according to the following rules:

Software name-*Major version number of the software*.**nupkg**

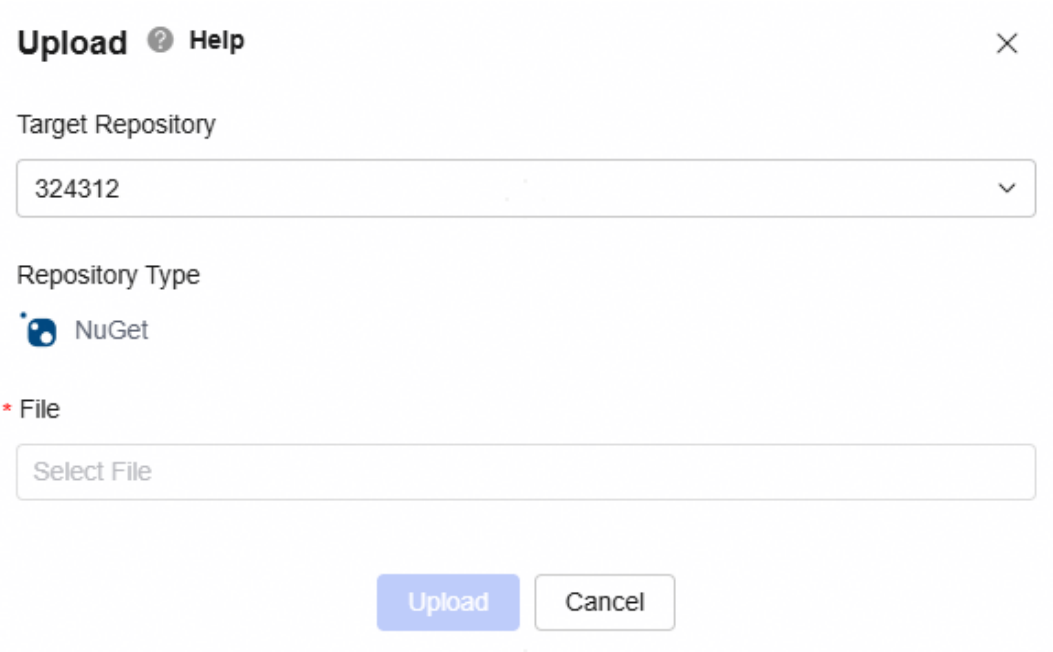
Example: **automapper.12.0.0.nupkg**

Procedure

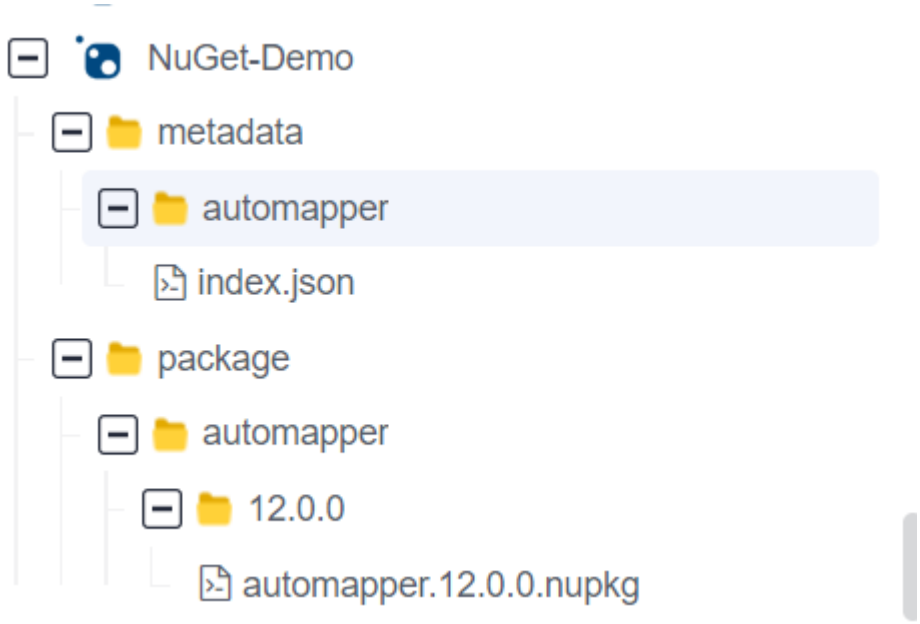
Step 1 Access self-hosted repos using your Huawei Cloud account.

Step 2 In the left pane, click the target NuGet repository to which the private package is to be uploaded.

Step 3 Click **Upload** on the right of the page, select the NuGet file to be uploaded from the local PC, and click **Upload**.



View the uploaded package in the repository list, as shown in the following figure:



- **metadata** stores metadata and is named after the package name. **metadata** cannot be deleted manually. It will be deleted or added automatically when the corresponding package is deleted or restored.
- **package** stores components.

----End

Uploading a CocoaPods Package

CocoaPods pod

CocoaPods is a dependency manager for iOS and macOS projects. It helps developers easily integrate and manage required libraries efficiently. CocoaPods artifacts (Pods) are third-party libraries published via CocoaPods.

Procedure

Step 1 Access self-hosted repos using your Huawei Cloud account.

Step 2 In the left pane, click the target CocoaPods repository to which the private package is to be uploaded.

Step 3 Click **Upload** and set the parameters described in the following table.

Item	Description
Package Name	Enter the package name.
File	Select the CocoaPods file to be uploaded from the local PC.

Step 4 Click **Upload**.

When the progress bar in the lower-right corner of the page shows 100%, it indicates the upload is complete.

----End

Uploading an OHPM Package

OHPM package

OpenHarmony Package Manager (OHPM) is an ArkTS package management tool. OHPM manages packages, which are stored and organized in the OHPM repository.

The OHPM package contains the Harmony Archive (HAR) package and Harmony Shared Package (HSP) package.

- HAR: Static shared package, which is reused in the build phase. It is mainly used as a second-party or third-party library for other applications to depend on. It includes resource files, *.so files, and metadata files.
- HSP: Dynamic shared package, which is reused in the running phase. It provides shared code and resources to improve code reusability and maintainability.

The metadata file, **oh-package.json5**, in the HAR package describes the component, including its name, version, and dependency information. The **ohpm install** command automatically reads the dependencies in **oh-package.json5** and downloads them.

Example: **oh-package.json5**

```
{
  "name": "@scope/demo",                  // Component name
  "version": "0.0.1",                 // Version
  "author": "artifact",                     // Author
  "license": "Apache-2.0",                // License agreement
  "main": "index.ts",                     // Entry file
  "description": "basic information.",      // Description
  "dependencies": {                       // Dependency
    "@scope/demo1": "1.0.0",
    "@scope/demo2": "file:./demo2.har"
  },
  "devDependencies": {                      // Development dependency
    "@scope/demo3": "1.0.0",
    "@scope/demo4": "1.0.0"
  },
  "metadata": {                           // Metadata information
    "sourceRoots": ["./src/main"]
  }
}
```

The **name** and **version** are the most important fields and must exist. Otherwise, the current package cannot be installed. The two attributes together form the unique identifier of an OHPM package.

name indicates the name of the package. The first part of the name, such as **@scope**, is used as the namespace. The other part is **name**. Generally, you can search with the **name** field to install and use the required package.

```
{
  "name": "@scope/name"
}
```


version indicates the version of the package, which is in the x.y.z format.

```
{
  "version": "1.0.0"
}
```

Procedure

You can upload OHPM packages in .har or .hsp format to the self-hosted repo. The procedure is as follows:

Step 1 **Access self-hosted repos** using your Huawei Cloud account.

Step 2 In the left pane, click the target OHPM repository to which the private package is to be uploaded.

Step 3 Click **Upload** on the right of the page and set the parameters described in the following table.

Item	Description
Package Name	Enter the package name. The value must be the same as that of **name** in the **oh-package.json5** file. Ensure that the package name starts with a path in the path list added during repository creation. For details, see **Table 4-1** in the help guide. Example: The path **@test** is added when you create an OHPM repository. When uploading the package to the repository, make sure that the **PackageName** starts with **@test**. If a path outside the path list is used, for example, **@ohos**, the upload will fail.
Version	Enter the version number. The value must be the same as that of **version** in the **oh-package.json5** file.
File	Select the OHPM file to be uploaded from the local PC.

Step 4 Click **Upload**.

When the progress bar in the lower-right corner of the page shows 100%, it indicates the upload is complete.

You can find the package in the repository list and the corresponding metadata is generated in the **.ohpm** directory.

----End

Uploading a Docker Image

Docker image

Docker artifacts refer to container images built and published using Docker. Docker images contain all dependencies and configurations required for running applications, ensuring portability and consistent behavior across environments.

Procedure

Step 1 Access self-hosted repos using your Huawei Cloud account.

Step 2 In the left pane, click the target Docker repository to which the private package is to be uploaded.

Step 3 Click **Upload** on the right of the page and set the parameters described in the following table.

Item	Description
Package Name	Enter the package name.
File	Select the Docker file to be uploaded from the local PC.

Step 4 Click **Upload**.

When the progress bar in the lower-right corner of the page shows 100%, it indicates the upload is complete.

----End

Uploading a Generic Artifact

Generic artifact

Generic artifacts (also called common artifacts) are files of any type generated during the build and release process.

Procedure

Step 1 Access self-hosted repos using your Huawei Cloud account.

Step 2 In the left pane, click the target Generic repository to which the private package is to be uploaded.

Step 3 Click **Upload** on the right of the page and set the parameters described in the following table.

Item	Description
Upload Mode	Select **Single file** or **Multiple files**. **Single file** is selected by default here. When **Multiple files** is selected, a maximum of 20 files can be uploaded at a time.
Path	After you set the path name, a folder with that name is created in the **Repo View**, where the uploaded packages will be stored.
File	Select the Generic artifact to be uploaded from the local PC.

Step 4 Click **Upload**.

When the progress bar in the lower-right corner of the page shows 100%, it indicates the upload is complete.

----End

Uploading a Composer Package

Composer package

- **name**: The value consists of the vendor name and project name, separated by a slash (/). Example: **monolog/monolog**.

  The name must contain lowercase letters, digits, underscores (_), periods (.), and hyphens (-). The regular expression is used: **^[a-z0-9]([_.-]?[a-z0-9]+)*/[a-z0-9](([_.]?|-{0,2})[a-z0-9]+)*$**.
- **version**: The component version number, which follows the format X.Y.Z or vX.Y.Z, and can end with -dev, -patch (-p), -alpha (-a), -beta (-b), and -RC. Examples: **1.0.0**, **v2.0.0**, and **3.0.0-dev**.

Procedure

Step 1 Access self-hosted repos using your Huawei Cloud account.

Step 2 In the left pane, click the target Composer repository to which the private package is to be uploaded.

Step 3 Click **Upload** on the right of the page and set the parameters described in the following table.

Item	Description
Package Name	Enter the package name, for example, **monolog/monolog**.
Version	Enter the version number, for example, **v2.0.0**.
File	Select the Composer file to be uploaded from the localhost.

Step 4 Click **Upload**.

When the progress bar in the lower-right corner of the page shows 100%, it indicates the upload is complete.

----End

Uploading a Helm Chart

Helm chart

Helm is an open-source Kubernetes package manager that allows you to find, share, and use pre-configured deployments of applications on Kubernetes. Helm provides the following functions:

- Simplifies the deployment and management of Kubernetes applications.
- Installs, updates, and uninstalls applications on Kubernetes.
- Provides an easy way to manage application dependencies.
- Shares and reuses application configurations.

You are advised to package and name the Helm file according to the following rules:

Software name-*Software version*.**tgz**

Example: **helloworld-3.0.1.tgz**

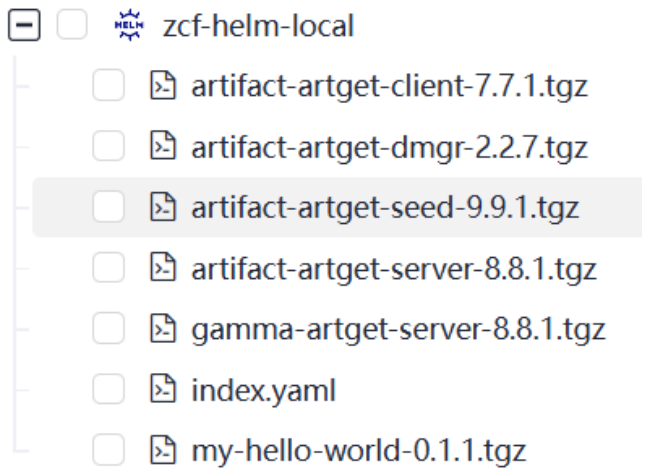
Procedure

Step 1 Access self-hosted repos using your Huawei Cloud account.

Step 2 In the left pane, click the target Helm repository to which the private package is to be uploaded.

Step 3 Click **Upload**, select the Helm file to be uploaded from the localhost, and click **Upload**.

Step 4 The uploaded chart is displayed in the repository list. The **index.yaml** file contains metadata, and the *.tgz file is the Helm software package, as shown in the figure below.



----End

Uploading a Conda Package

Conda package

Conda is an open-source package and environment management system that runs on Windows, macOS, and Linux. Conda can install, run, and update packages and their dependencies. With Conda, you can create, save, load, and switch environments on the localhost. While it is designed for Python programs, it can also package and distribute software for any language.

Procedure

The Conda repository supports .conda and .tar.bz2 packages. You can create multi-level directories and generate metadata files at the package level. The procedure is as follows:

Step 1 Access self-hosted repos using your Huawei Cloud account.

Step 2 In the left pane, click the target Conda repository to which the private package is to be uploaded.

Step 3 Click **Upload** on the right of the page and set the parameters described in the following table.

Item	Description
Path	Mandatory. Specify the storage path of the Conda package. Explicitly specify the platform in the last path segment (for example: **linux-64**, **linux-32**, **win-64**, **win-32**, **osx-64**, **osx-arm64**, **linux-aarch64**, or **noarch**).
File	Select the Conda file to be uploaded from the localhost.

Step 4 Click **Upload**.

When the progress bar in the lower-right corner of the page shows 100%, it indicates the upload is complete.

----End

Downloading a Package

Step 1 **Access self-hosted repos** using your Huawei Cloud account.

Step 2 In the left pane, locate the package to be downloaded and click its name.

If there are too many repositories or packages, you can search for the desired one by referring to **Searching for a Package**.

Step 3 Click **Download** on the right of the page.

----End

4.6 Uploading/Downloading Packages on the Client

4.6.1 Connecting Self-Hosted Repos to Your Local Development Environment

By default, the Maven tool connects to the public repository, and is implemented through configurations in the **setting.xml** file. Self-hosted repos can be connected to your local development environment. You can modify the configuration in the **setting.xml** file to connect a self-hosted repo to your local development environment.

Configuring setting.xml Files

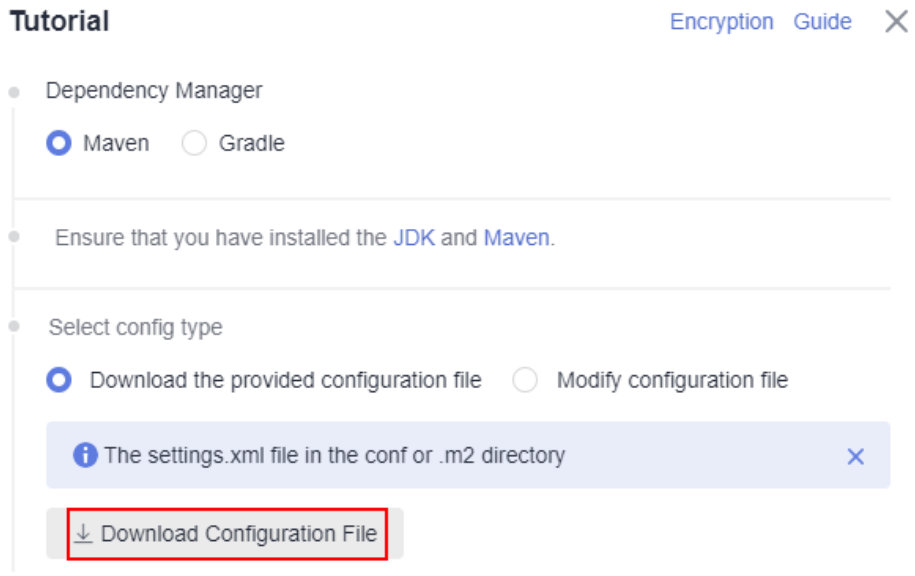
Step 1 **Access self-hosted repos** using your Huawei Cloud account.

Step 2 In the left pane, click the name of the repository to be connected to your local development environment.

Step 3 Click **Tutorial** on the right of the page.

Step 4 In the displayed dialog box, click **Download Configuration File** to download the configuration file to your local directory.

Figure 4-5 Downloading the configuration file



Step 5 Copy the downloaded file to the corresponding Maven directory based on the instructions in the information dialog box.

----End

Resetting Repository Password

WARNING

You can reset the password in the configuration file of self-hosted repo. After the password is reset, download the configuration file again to replace the original file.

Step 1 **Access self-hosted repos** using your Huawei Cloud account.

Step 2 In the left pane, select the repository whose password you want to reset.

Step 3 Click ••• above the repository list on the right and select **Reset Repository Password**.

Step 4 In the displayed dialog box, click **OK**. Check that a message is displayed indicating the password has been reset.

----End

4.6.2 Uploading and Downloading a Maven Artifact via the Client

CodeArts Artifact can connect to local clients. You may upload your private packages from your local client to self-hosted repos and share them with others, who can download these packages through their clients.

Maven Artifact

Maven artifacts are the output files generated during the Maven build process, which usually include compiled code (e.g., JARs, WARs, and EARs) and related metadata, such as dependency information and version numbers. For details, see **Uploading a Maven Artifact**.

Constraints

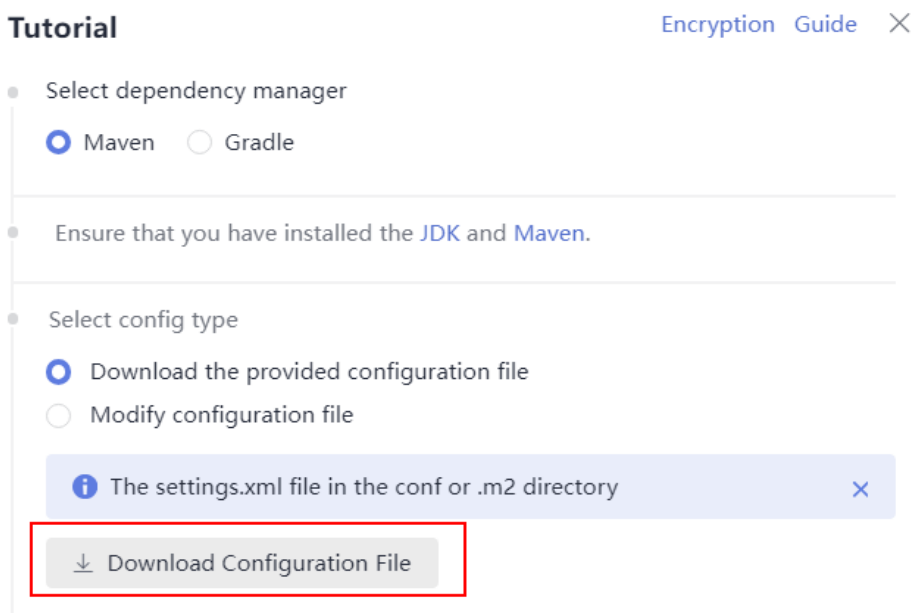
The repository password for a self-hosted repo is account-specific. If you are prompted for a password while using a different account, download the configuration file from the repository's **Tutorial** tab to obtain it.

Prerequisites

- The client tools are Maven and Gradle. You have installed JDK, Maven, and Gradle.
- You have **created a Maven repository**.
- You have the **Download/View** permission on the current repository. For details about how to obtain this permission, see **Configuring Repository Permissions**.

Uploading a Maven Artifact via the Client

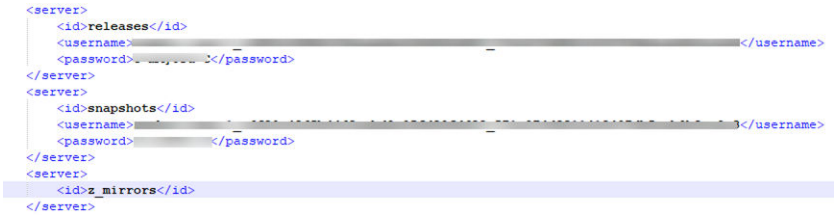
- **The client tool is Maven. Ensure that the JDK and Maven have been installed.**
  a. **Access self-hosted repos** using your Huawei Cloud account.
  b. Select the target Maven repository from the repository list.
  c. Click **Tutorial** on the right of the page.
  d. In the displayed dialog box, set **Select dependency manager** to **Maven**.
  e. Click **Download Configuration File** to download the **settings.xml** file. Replace the downloaded file or modify your Maven **settings.xml** file as prompted.

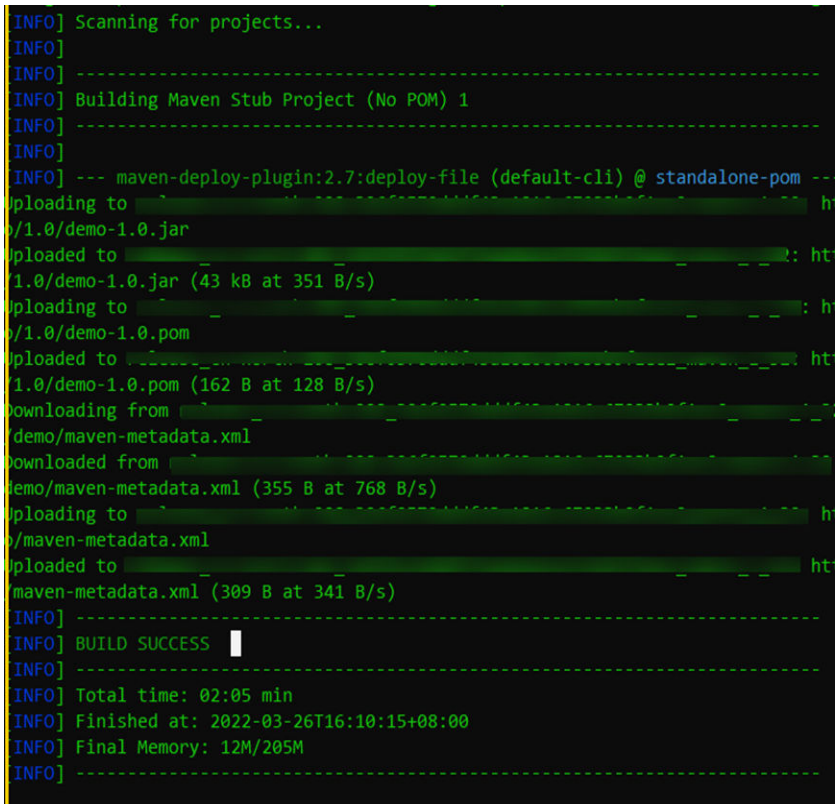


f. Run the following commands (example) in the directory where the uploaded POM file is located:

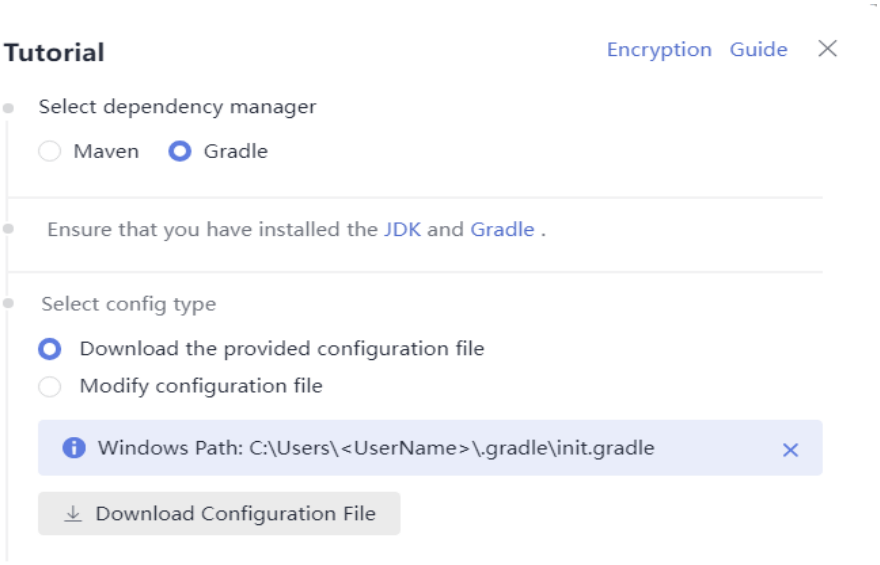
```
mvn deploy:deploy-file -DgroupId={groupId} -DartifactId={artifactId} -Dversion={version} -Dpackaging=jar -Dfile={file_path} -DpomFile={pom_path} -Durl={url} -DrepositoryId={repositoryId} -s {settings_path} -Dmaven.wagon.http.ssl.insecure=true -Dmaven.wagon.http.ssl.allowall=true -Dmaven.wagon.http.ssl.ignore.validity.dates=true
```

- **Parameter description**
  - *DgroupId*: uploaded group ID
  - *DartifactId*: uploaded artifact ID
  - *Dversion*: version of the uploaded file
  - *Dpackaging*: type of the package to be uploaded, such as JAR, ZIP, and WAR
  - *Dfile*: path of the uploaded entity file
  - *DpomFile*: path of the entity POM file to be uploaded. (For a release version, if this parameter does not exist, the system automatically generates a POM file. If there are special requirements for the POM file, specify this parameter.)
  - The values of *DgroupId*, *DartifactId*, and *Dversion* in the POM file must be the same as those outside the POM file. Otherwise, error 409 is reported.
  - Select either *DpomFile* or (*DgroupId*, *DartifactId*, and *Dversion)*.
  - *Durl*: path for uploading files to the repository.
  - *DrepositoryId*: ID corresponding to the username and password configured in settings, as shown in the following figure:





- **The client tool is Gradle. Ensure that JDK and Gradle have been installed.**
  a. **Access self-hosted repos** using your Huawei Cloud account.
  b. In the left pane, click the name of the repository to be connected to your local development environment.
  c. Click **Tutorial** on the right of the page.
  d. In the displayed dialog box, set **Select dependency manager** to **Gradle**.
  e. Click **Download the provided configuration file** to download the **init.gradle** file from the self-hosted repo page.



  f. Find the **build.gradle** file in the local project and add the following commands to the **gradle** file:


```
uploadArchives {
    repositories {
        mavenDeployer {repository(url:"***") {
                authentication(userName: "{repo_name}", password: "{repo_password}")
            }
            // Pom file of the build project
            pom.project {
                name = project.name
                packaging = 'jar'
                description = 'description'
            }
        }
    }
}
```

- *url*: path for uploading files to the repository. You can click on the Maven repository page to obtain the URL.
- *{repo_name}*: username obtained from the **init.gradle** file downloaded from the Maven repository page.
- *{repo_password}*: password obtained from the **init.gradle** file downloaded from the Maven repository page.

g. Run the following command in the directory where the local project is located:

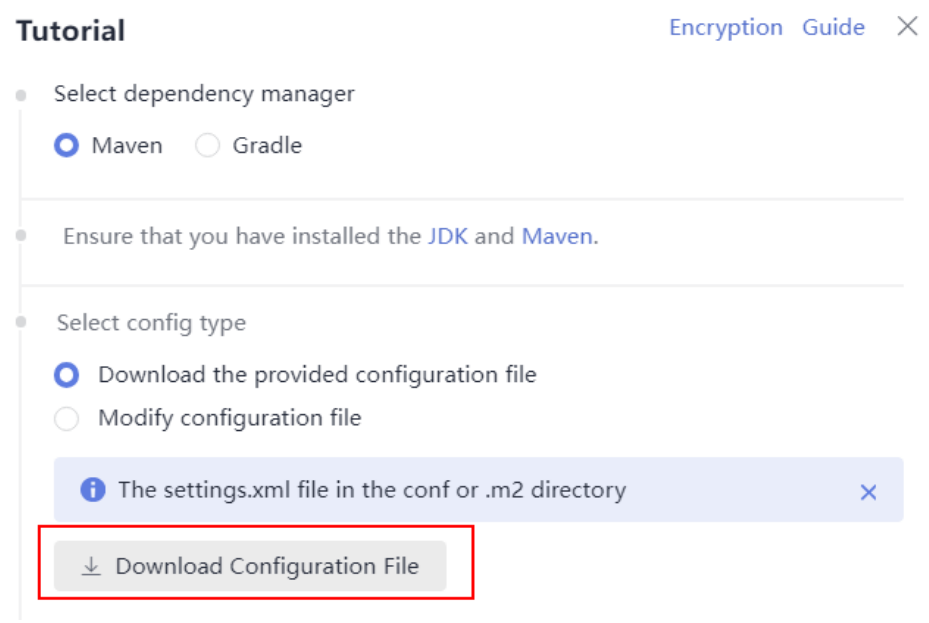
```
gradle uploadArchives
```

h. Return to the target Maven repository to view the uploaded artifact.

Downloading a Maven Artifact via the Client

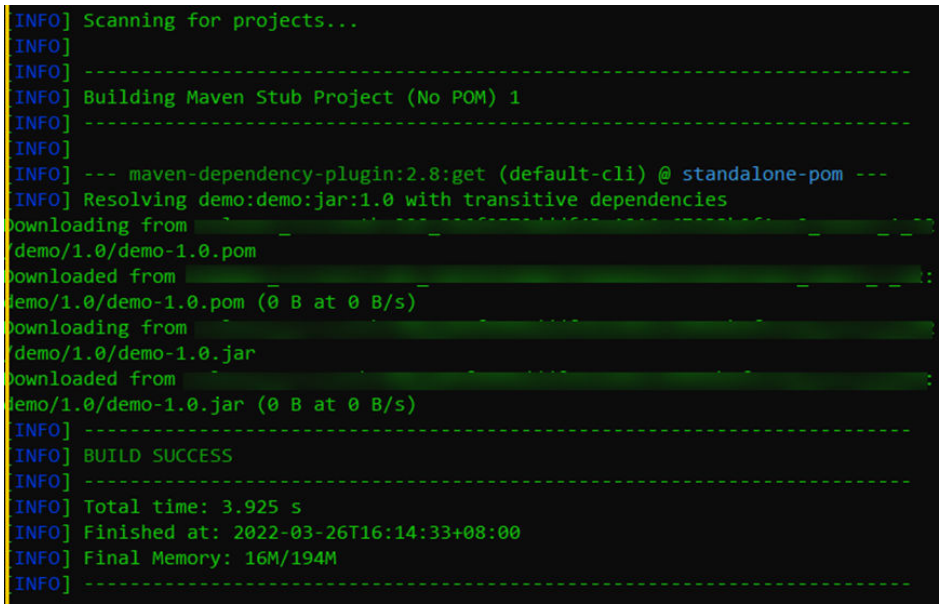
The client tool is Maven. Ensure that the JDK and Maven have been installed.

1. **Access self-hosted repos** using your Huawei Cloud account.
2. Select the target Maven repository on the self-hosted repo page.
3. Click **Tutorial** on the right of the page.
4. In the displayed dialog box, set **Select dependency manager** to **Maven**.
5. Click **Download Configuration File** to download the **settings.xml** file. Replace the **settings.xml** file with the downloaded file or modify your Maven **settings.xml** file as prompted.



6. Download the artifact to the repository using the client.

```
mvn dependency:get -DremoteRepositories={repo_url} -DgroupId={groupId} -DartifactId={artifactId} -Dversion={version} -Dmaven.wagon.http.ssl.insecure=true -Dmaven.wagon.http.ssl.allowall=true -Dmaven.wagon.http.ssl.ignore.validity.dates=true
```



4.6.3 Uploading and Downloading an npm Package via the Client

CodeArts Artifact can connect to local npm clients. You may upload your private packages from your local npm client to self-hosted repos and share them with others, who can download these npm packages through their clients.

npm Package

Node Package Manager (npm) artifacts are packages published through the npm package manager for the JavaScript runtime environment Node.js.

Constraints

The repository password for a self-hosted repo is account-specific. If you are prompted for a password while using a different account, download the configuration file from the repository's **Tutorial** tab to obtain it.

Prerequisites

- The client tool is npm. You have installed Node.js (or io.js) and npm.
- You have **created an npm registry**.
- You have the **Download/View** permission on the current registry. For details about how to obtain this permission, see **Configuring Repository Permissions**.

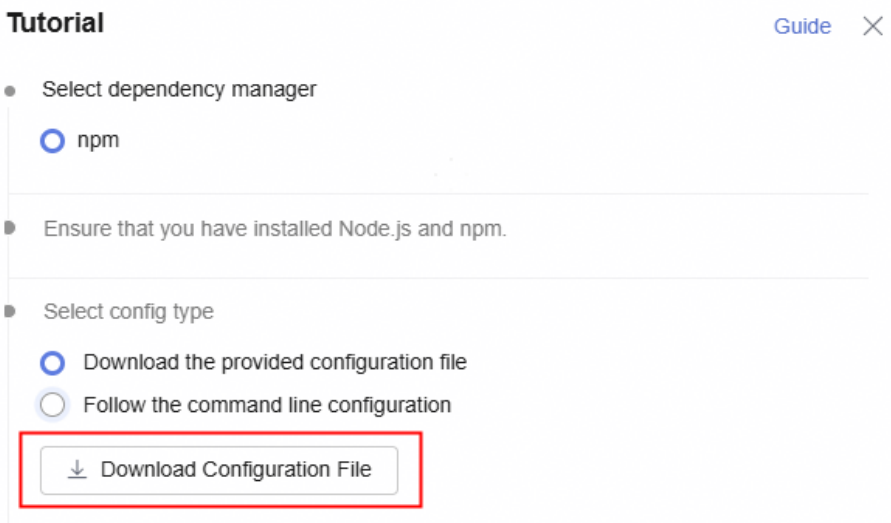
Uploading an npm Package via the Client

Step 1 **Access self-hosted repos** using your Huawei Cloud account.

Step 2 In the left pane, click the name of the repository to be connected to your local development environment.

Step 3 Click **Tutorial** on the right of the page.

Step 4 In the displayed dialog box, click **Download Configuration File** to download the **npmrc** file, and save it as the **.npmrc** file.



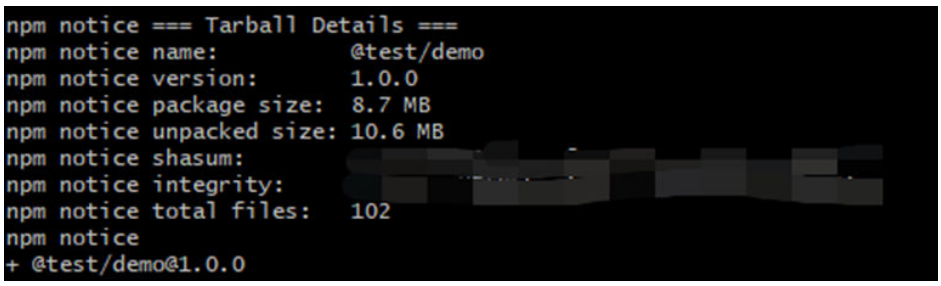
Step 5 Copy the file to the user directory. In Linux, the path is **~/.npmrc** (C:\Users\<*UserName*>\.npmrc).

Step 6 Go to the npm project directory (where the **package.json** file is stored), open the **package.json** file, and add the URL information entered during repository creation to the value of the **name** field.


```
{
  "name": "@test/demo",
  "version": "1.0.0",
  "description": "demo",
  "main": "index.js",
  "engines": {
    "node": ">= 8.0.0",
    "npm": ">= 5.0.0"
  },
```

Step 7 Upload the npm package to the registry.

```
npm config set strict-ssl false
npm publish
```



----End

Downloading an npm Package via the Client

The client tool is npm. Ensure that node.js (or io.js) and npm have been installed.

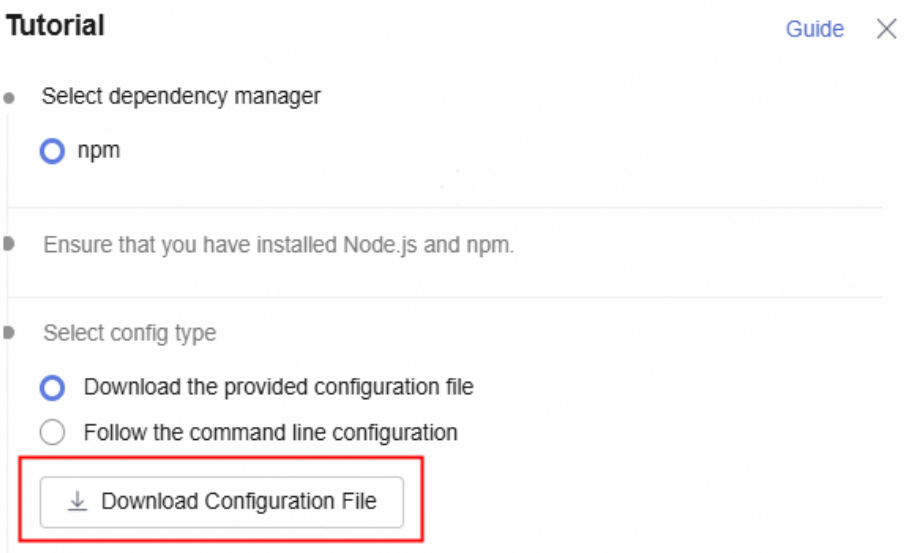
Step 1 Access self-hosted repos using your Huawei Cloud account.

Step 2 Select the target npm registry on the self-hosted repo page.

Step 3 Click **Tutorial** on the right of the page.

Step 4 In the displayed dialog box, click **Download Configuration File** to download the **npmrc** file.

Step 5 Save the downloaded **npmrc** file as the **.npmrc** file.



Step 6 Copy the file to the user directory. In Linux, the path is **~/.npmrc**. In Windows, the path is **C:\Users\<*UserName*>\.npmrc**.

Step 7 Go to the npm project directory (where the **package.json** file is stored) and run the following commands to download the npm dependency.

```
npm config set strict-ssl false
npm install ––verbose
```

----End

4.6.4 Uploading and Downloading an RPM Package via the Client

CodeArts Artifact can connect to local RPM clients. You may upload your private packages from your local RPM client to self-hosted repos and share them with others, who can download these RPM packages through their clients.

RPM Package

Red Hat Package Manager (RPM) is a package format and management tool originally developed by Red Hat and now widely used across Linux distributions such as Red Hat Enterprise Linux (RHEL), Fedora, CentOS, and SUSE. RPM packages are files that contain software and metadata, used to install, upgrade, and remove software packages.

Constraints

The repository password for a self-hosted repo is account-specific. If you are prompted for a password while using a different account, download the configuration file from the repository's **Tutorial** tab to obtain it.

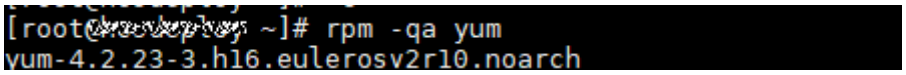
Prerequisites

- The client tool is Linux OS with yum. Ensure that the Linux OS is used and yum has been installed.

  Run the following command on the Linux host to check whether yum is installed:

  ```
  rpm -qa yum
  ```

  If the following information is displayed, yum has been installed on the server.

- You have **created an RPM repository**.
- You have the **Download/View** permission on the current repository. For details about how to obtain this permission, see **Configuring Repository Permissions**.

Uploading an RPM Package via the Client

Step 1 **Access self-hosted repos** using your Huawei Cloud account.

Step 2 Select the target RPM repository on the self-hosted repo page.

Step 3 Click **Tutorial** on the right of the page.

Step 4 In the displayed dialog box, click **Download Configuration File**.

Step 5 On the Linux host, run the following command to upload an RPM package:

```
curl -k -u {{user}}:{{password}} -X PUT https://{{repoUrl}}/{{component}}/{{version}}/ -T {{localFile}}
```

In this command, *user*, *password*, and *repoUrl* can be obtained from the **RPM upload command** in the configuration file downloaded in the **previous step**.

- *user*: character string before the colon (:) between **curl -u** and **-X**
- *password*: character string after the colon (:) between **curl -u** and **-X**
- *repoUrl*: character string between **https://** and **/{{component}}**

component, *version*, and *localFile* can be obtained from the RPM package to be uploaded. The **hello-0.17.2-54.x86_64.rpm** package is used as an example.

- *component*: software name, for example, **hello**.
- *version*: software version, for example, **0.17.2**.
- *localFile*: RPM package, for example, **hello-0.17.2-54.x86_64.rpm**.

The following figure shows the complete command.

After the command is successfully executed, go back to the self-hosted repo and find the uploaded RPM package.

----End

Downloading an RPM Package via the Client

Step 1 Access self-hosted repos using your Huawei Cloud account.

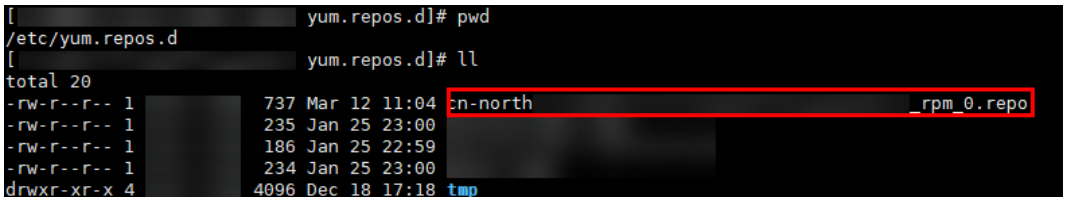
Step 2 Select the target RPM repository on the self-hosted repo page.

Step 3 Click **Tutorial** on the right of the page.

Step 4 In the displayed dialog box, click **Download Configuration File**.

Step 5 Open the configuration file, replace all *{{component}}* placeholders with the value used when uploading the RPM file (for example, **hello** in this document), remove the **RPM upload command**, and save the file.

Step 6 Save the modified configuration file to the **/etc/yum.repos.d/** directory on the Linux host.



Step 7 Run the following command to download the RPM package. Replace **hello** with the actual value of *component*.

```
yum install hello
```

----End

4.6.5 Uploading and Downloading a PyPI Package on the Client

CodeArts Artifact can connect to local PyPI clients. You may upload your private packages from your local PyPI client to self-hosted repos and share them with others, who can download these PyPI packages through their clients.

PyPI Package

The Python Package Index (PyPI) is the official third-party software repository for the Python community. Through PyPI, developers can publish and distribute Python packages.

Constraints

The repository password for a self-hosted repo is account-specific. If you are prompted for a password while using a different account, download the configuration file from the repository's **Tutorial** tab to obtain it.

Prerequisites

- You have installed Python and Twine.
- You have **created a PyPI repository**.
- You have the **Download/View** permission on the current repository. For details about how to obtain this permission, see **Configuring Repository Permissions**.

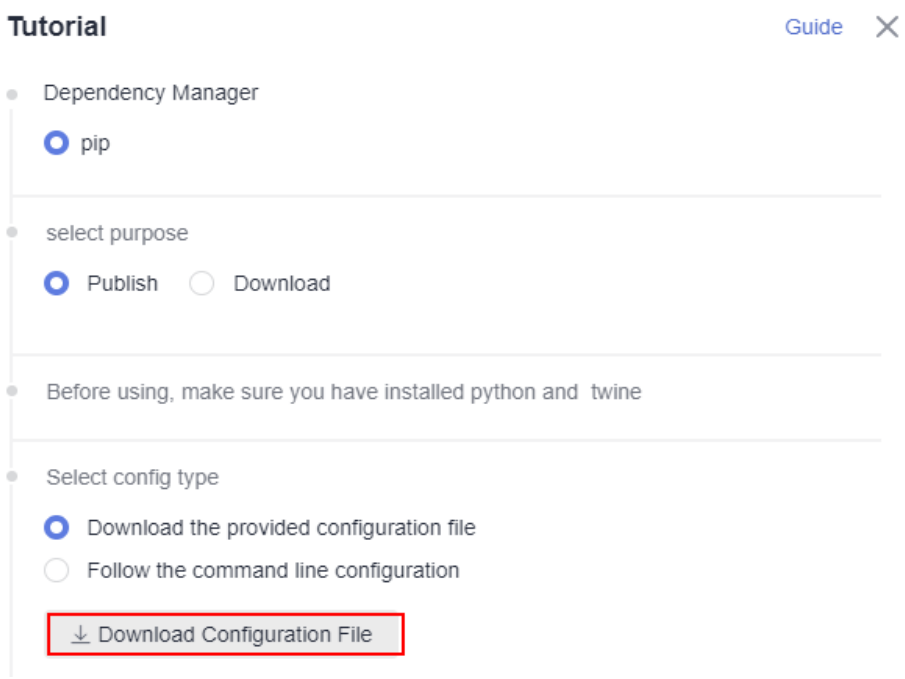
Uploading a PyPI Package from the Client

Step 1 **Access self-hosted repos** using your Huawei Cloud account.

Step 2 Select the target PyPI repository on the self-hosted repo page.

Step 3 Click **Tutorial** on the right of the page.

Step 4 In the displayed dialog box, click **Download Configuration File** to download the **pypirc** file, and save it as the **.pypirc** file.



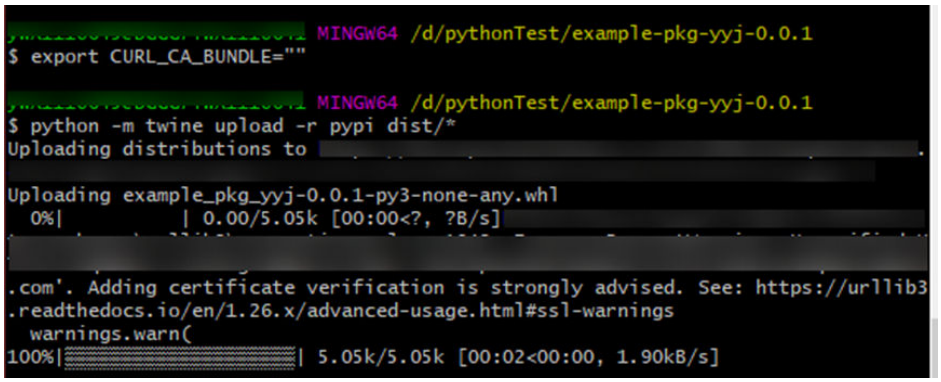
Step 5 Copy the file to the user directory. In Linux, the path is **~/.pypirc**. In Windows, the path is **C:\Users**<*UserName*>**\.pypirc**.

Step 6 Go to the Python project directory and run the following command to compress the Python project into a **.whl** package.

```
python setup.py bdist_wheel
```

Step 7 Upload the file to the repository.

```
python -m twine upload -r pypi dist/*
```

If a certificate error is reported during the upload, run the following command (use Git Bash on Windows) to set environment variables to skip certificate verification (skip this step if you are using Twine 3.8.0 or earlier and Requests 2.27 or earlier.)

```
export CURL_CA_BUNDLE=""
```

The environment variables will be reset if you log in to the server again, switch users, or open a new bash window. Be sure to set the environment variables before each upload.

----End

Downloading a PyPI Package to the Client

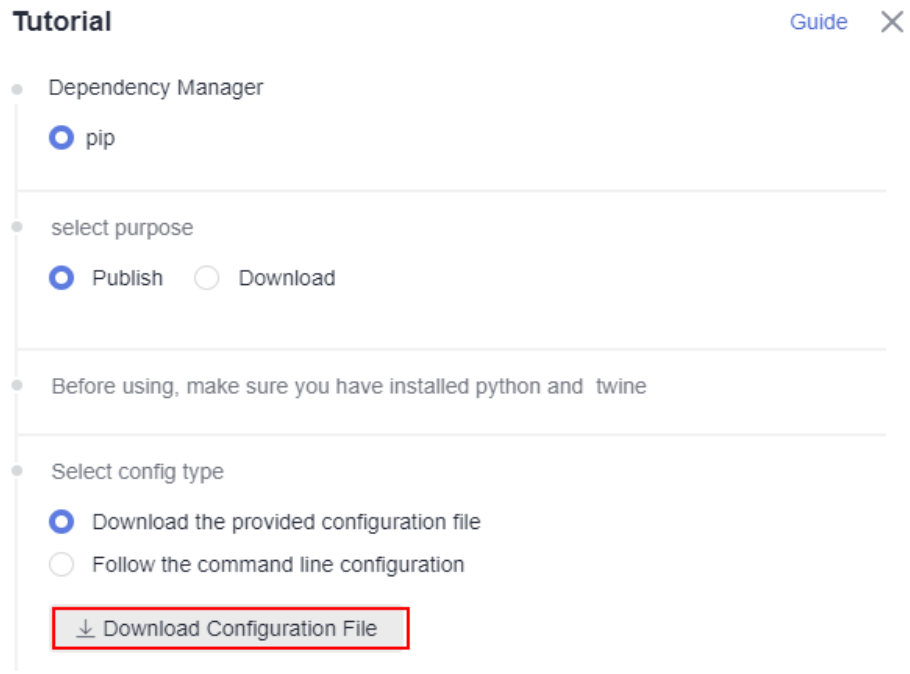
Step 1 **Access self-hosted repos** using your Huawei Cloud account.

Step 2 Select the target PyPI repository on the self-hosted repo page.

Step 3 Click **Tutorial** on the right of the page.

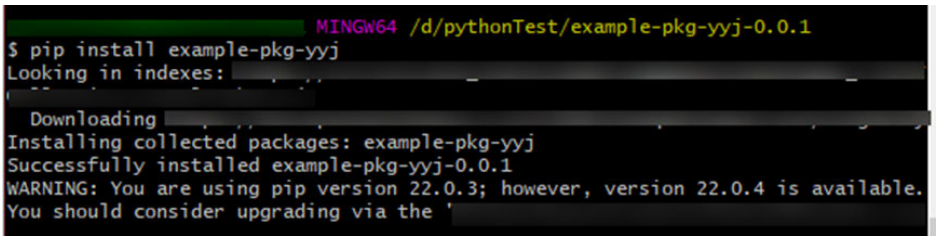
Step 4 In the displayed dialog box, set **Select purpose** to **Download** and click **Download Configuration File**.

Step 5 Download the **pip.ini** file from the self-hosted repo page and copy the file to the user directory. In Linux, the path is **~/.pip/pip.conf** (**C:\Users\<***UserName***>\pip\pip.ini** on Windows)



Step 6 Install the Python package.

```
pip install {package name}
```



----End

4.6.6 Uploading and Downloading a Go Package on the Client

CodeArts Artifact can connect to local Go clients. You may upload your private packages from your local Go client to self-hosted repos and share them with others, who can download these Go packages through their clients.

Go Package

Go (Golang) packages are packages or applications developed and published using the Go programming language. These packages can be libraries (library packages) or executable files (binary files).

Constraints

The repository password for a self-hosted repo is account-specific. If you are prompted for a password while using a different account, download the configuration file from the repository's **Tutorial** tab to obtain it.

Prerequisites

- The client tool is Go. Ensure that V1.13 or a later version has been installed and the project is a Go module project.
- You have **created a Go repository**.
- You have the **Download/View** permission on the current repository. For details about how to obtain this permission, see **Configuring Repository Permissions**.

Uploading a Go Package from the Client

- Go Modules packaging and package upload

  This section describes how to build and upload Go packages through Go modules. The username and password used in the following steps can be obtained from the downloaded configuration file for the Go repository.

  Perform the following steps:

  a. Create a source folder in the working directory.
  ```
  mkdir -p {module}@{version}
  ```
  b. Copy the code source to the source folder.
  ```
  cp -rf . {module}@{version}
  ```
  c. Compress the package into a ZIP package.
  ```
  zip -D -r [package name] [package root directory]
  ```
  d. Upload the ZIP package and the **go.mod** file to the self-hosted repo.
  ```
  curl -k -u {{username}}:{{password}} -X PUT {{repoUrl}}/{filePath} -T {{localFile}}
  ```

  The package directory varies according to the package version. The version can be:

  - Versions earlier than v2.0: The directory is the same as the path of the **go.mod** file. No special directory is required.
  - v2.0 or later:
    - If the first line in the **go.mod** file ends with **/vX**, the directory must contain **/vX**. For example, if the version is v2.0.1, the directory must contain **v2**.
    - If the first line in the **go.mod** file does not end with **/vN**, the directory remains unchanged and the name of the file to be uploaded must contain **+incompatible**.

    The following are examples of package directories for different versions:

  Versions earlier than v2.0

  The **go.mod** file is used as an example.

  go.mod
  ```
  1  module example.com/demo
  ```

  a. Create a source folder in the working directory.

  The value of *module* is **example.com/demo** and that of *version* is **1.0.0**. The command is as follows.
  ```
  mkdir -p ~/example.com/demo@v1.0.0
  ```


b. Copy the code source to the source folder.

The command is as follows (with the same parameter values as the previous command).

```
cp -rf . ~/example.com/demo@v1.0.0/
```

c. Compress the package into a ZIP package.

Run the following command to go to the upper-level directory of the root directory where the ZIP package is located:

```
cd ~
```

Then, use the **zip** command to compress the code into a package. In this command, the package root directory is **example.com** and the package name is **v1.0.0.zip**. The command is as follows.

```
zip -D -r v1.0.0.zip  example.com/
```

d. Upload the ZIP package and the **go.mod** file to the self-hosted repo.

Parameters *username*, *password*, and *repoUrl* can be obtained from the configuration file.

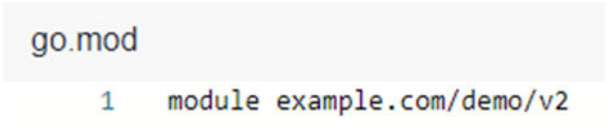
- For the ZIP package, the value of *filePath* is **example.com/demo/@v/v1.0.0.zip** and that of *localFile* is **v1.0.0.zip**.
- For the **go.mod** file, the value of *filePath* is **example.com/demo/@v/v1.0.0.mod** and that of *localFile* is **example.com/demo@v1.0.0/go.mod**.

The commands are as follows (replace *username*, *password*, and *repoUrl* with the actual values).

```
curl -k -u {{username}}:{{password}} -X PUT {{repoUrl}}/example.com/demo/@v/v1.0.0.zip -T v1.0.0.zip
curl -k -u {{username}}:{{password}} -X PUT {{repoUrl}}/example.com/demo/@v/v1.0.0.mod -T example.com/demo@v1.0.0/go.mod
```

- **v2.0 and later, with the first line in the go.mod file ends with /vX**.

The **go.mod** file is used as an example.



a. Create a source folder in the working directory.

The value of *module* is **example.com/demo/v2** and that of *version* is **2.0.0**. The command is as follows.

```
mkdir -p ~/example.com/demo/v2@v2.0.0
```

b. Copy the code source to the source folder.

The command is as follows (with the same parameter values as the previous command).

```
cp -rf . ~/example.com/demo/v2@v2.0.0/
```

c. Compress the package into a ZIP package.

Run the following command to go to the upper-level directory of the root directory where the ZIP package is located:

```
cd ~
```

Then, use the **zip** command to compress the code into a package. In this command, the package root directory is **example.com** and the package name is **v2.0.0.zip**. The command is as follows.

```
zip -D -r v2.0.0.zip  example.com/
```

d. Upload the ZIP package and the **go.mod** file to the self-hosted repo.

Parameters *username*, *password*, and *repoUrl* can be obtained from the configuration file.

- For the ZIP package, the value of *filePath* is **example.com/demo/v2/@v/v2.0.0.zip** and that of *localFile* is **v2.0.0.zip**.
- For the **go.mod** file, the value of *filePath* is **example.com/demo/v2/@v/v2.0.0.mod** and that of *localFile* is **example.com/demo/v2@v2.0.0/go.mod**.

The commands are as follows (replace *username*, *password*, and *repoUrl* with the actual values).

```
curl -u {{username}}:{{password}} -X PUT {{repoUrl}}/example.com/demo/v2/@v/v2.0.0.zip -T v2.0.0.zip
curl -u {{username}}:{{password}} -X PUT {{repoUrl}}/example.com/demo/v2/@v/v2.0.0.mod -T example.com/demo/v2@v2.0.0/go.mod
```

- **v2.0 and later, with the first line in the go.mod file does not end with /vX**.

The **go.mod** file is used as an example.

go.mod

```
1  module example.com/demo
```

a. Create a source folder in the working directory.

The value of *module* is **example.com/demo** and that of *version* is **3.0.0**. The command is as follows.

```
mkdir -p ~/example.com/demo@v3.0.0+incompatible
```

b. Copy the code source to the source folder.

The command is as follows (with the same parameter values as the previous command).

```
cp -rf . ~/example.com/demo@v3.0.0+incompatible/
```

c. Compress the package into a ZIP package.

Run the following command to go to the upper-level directory of the root directory where the ZIP package is located:

```
cd ~
```

Then, use the **zip** command to compress the code into a package. In this command, the package root directory is **example.com** and the package name is **v3.0.0.zip**. The command is as follows.

```
zip -D -r v3.0.0.zip  example.com/
```

d. Upload the ZIP package and the **go.mod** file to the self-hosted repo.

Parameters *username*, *password*, and *repoUrl* can be obtained from the configuration file.

- For the ZIP package, the value of *filePath* is **example.com/demo/@v/v3.0.0+incompatible.zip** and that of *localFile* is **v3.0.0.zip**.
- For the **go.mod** file, the value of *filePath* is **example.com/demo/@v/v3.0.0+incompatible.mod** and that of *localFile* is **example.com/demo@v3.0.0+incompatible/go.mod**.

The commands are as follows (replace *username*, *password*, and *repoUrl* with the actual values).

```
curl -k -u {{username}}:{{password}} -X PUT {{repoUrl}}/example.com/demo/@v/
v3.0.0+incompatible.zip -T v3.0.0.zip
curl -k -u {{username}}:{{password}} -X PUT {{repoUrl}}/example.com/demo/@v/
v3.0.0+incompatible.mod -T example.com/demo@v3.0.0+incompatible/go.mod
```

Downloading a Go Package to the Client

Certificate verification cannot be bypassed on the Go client. You need to add the domain name certificate corresponding to the self-hosted repo to the local certificate trust list and perform the following steps to add the trust certificate:

Step 1 Export the certificates.

```
openssl s_client -connect {host}:443 -showcerts </dev/null 2>/dev/null | sed -ne '/-BEGIN CERTIFICATE-/,/-
END CERTIFICATE-/p' |openssl x509 -outform PEM >mycertfile.pem
openssl x509 -outform der -in mycertfile.pem -out mycertfile.crt
```

mycertfile.pem and **mycertfile.crt** are the downloaded certificates.

Step 2 Add the certificates to the root certificate trust list.

Step 3 Run the go commands to download the dependency.

```
##1. v2.0 and earlier versions
go get -v <moudlename>
##2. v2.0 and later versions
##a. The ZIP package contains go.mod and the path ends with /vN.
go get -v {{moduleName}}/vN@{{version}}
##b. The ZIP package does not contain go.mod or the first line in go.mod does not end with /vN.
go get -v {moduleName}}@{{version}}+incompatible
```

----End

4.6.7 Uploading and Downloading a Debian Package via the Client

CodeArts Artifact can connect to local Debian clients. You may upload your private packages from your local Debian client to self-hosted repos and share them with others, who can download these Debian packages through their clients.

Debian Package

Debian packages are packages released through the package management system, such as Advanced Package Tool (APT), for the Debian GNU/Linux distribution. These packages contain all files required to install, configure, and use the software.

Constraints

The repository password for a self-hosted repo is account-specific. If you are prompted for a password while using a different account, download the configuration file from the repository's **Tutorial** tab to obtain it.

Prerequisites

- The client tool is apt. Ensure that the Ubuntu or Debian system has been installed. The apt repository source configuration file is **sources.list** in **/etc/apt/** of the user root directory.

- You have **created a Debian repository**.
- You have the **Download/View** permission on the current repository. For details about how to obtain this permission, see **Configuring Repository Permissions**.

Debian Configuration

The Debian repository provides the apt configuration, which can be added to the /etc/apt/sources.list file of the OS.

Step 1 **Access self-hosted repos** using your Huawei Cloud account.

Step 2 Select the target Debian repository on the self-hosted repo page.

Step 3 Click **Tutorial** in the upper right corner of the repository page.

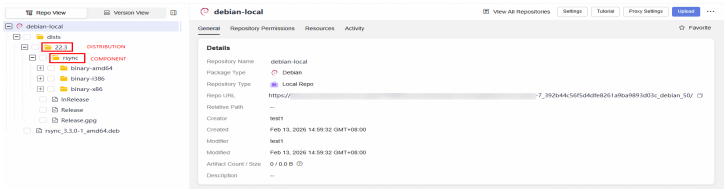
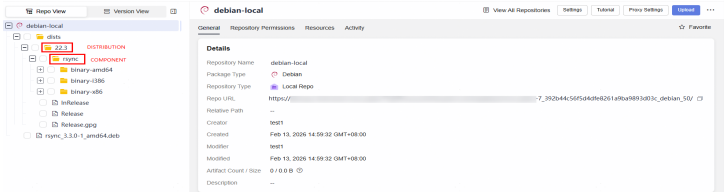
Step 4 In the displayed dialog box, click **Download Configuration File**.

Step 5 Run the commands in the configuration file.

The configuration file provides two methods for adding the Debian repository to the apt configuration. The first method is used as an example.

```
sudo sh -c "echo 'deb <REPO_URL> <DISTRIBUTION> <COMPONENT>' >> /etc/apt/sources.list"
sudo sh -c "echo 'machine <HOST> login <USERNAME> password <PASSWORD>' >> /etc/apt/auth.conf"
```

Table 4-12 Parameters

Item	Description
REPO_URL	Repository address, which can be obtained from the configuration file.
HOST	Domain name in the repository address.
DISTRIBUTION	Release version name, which is the name of the first-level directory under **dists**, as shown in the following figure. 
COMPONENT	Component name, which is the name of the second-level directory under **dists**, as shown in the following figure. 
USERNAME	Username, which can be obtained from the configuration file.

Item	Description
PASSWORD	User password, which can be obtained from the configuration file.

----End

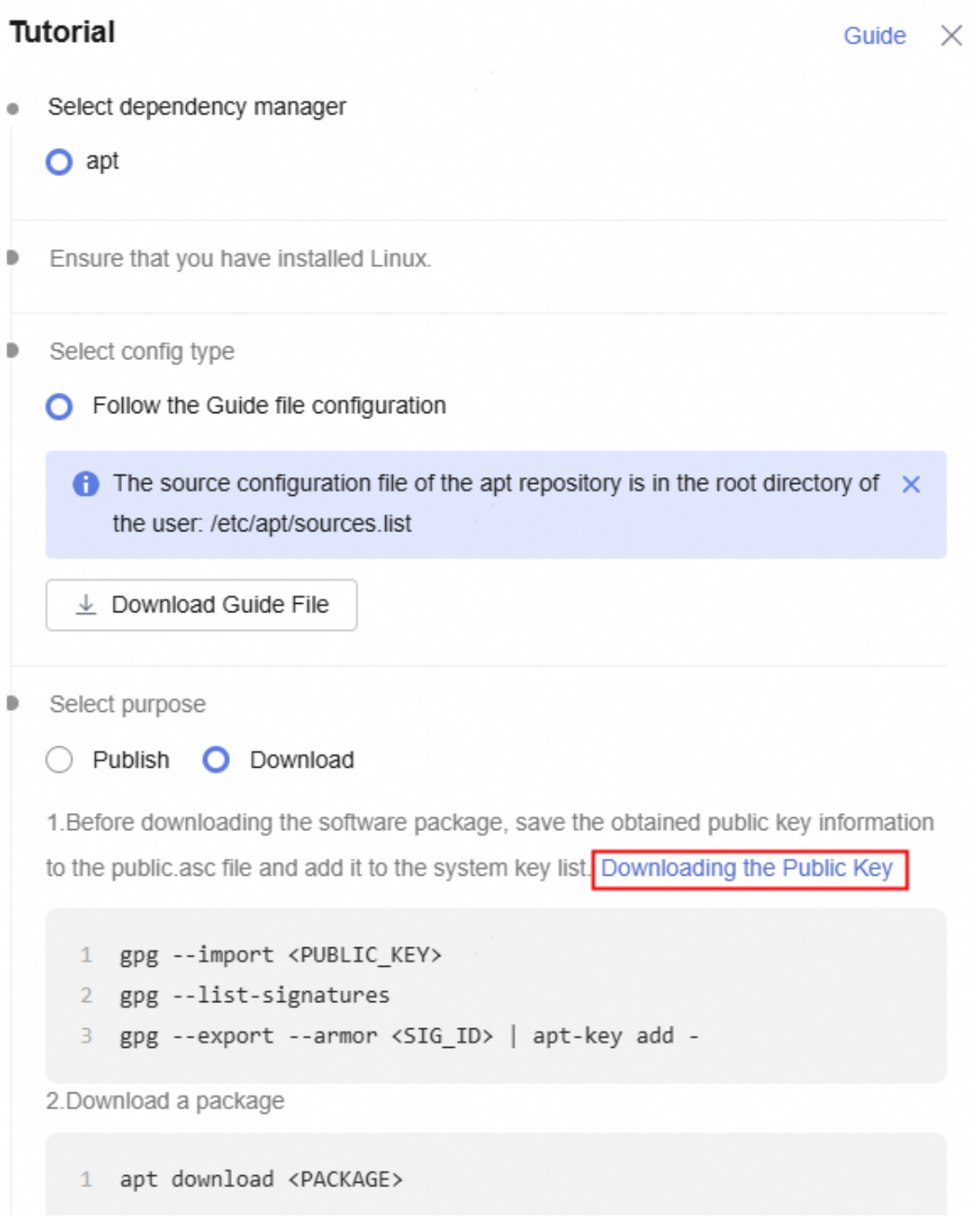
Downloading GPG public keys

Step 1 **Access self-hosted repos** using your Huawei Cloud account.

Step 2 Select the target Debian repository on the self-hosted repo page.

Step 3 Click **Tutorial** in the upper right corner of the repository page.

Step 4 In the displayed dialog box, set **Select purpose** to **Download** and click **Downloading the Public Key**, as shown in the following figure:



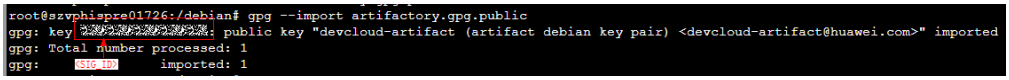
Step 5 Add the GPG public key.

Run the following three commands to add the GPG public key.

```
gpg --import <PUBLIC_KEY>
gpg --list-signatures
gpg --export --armor <SIG_ID> | apt-key add -
```

Parameter description:

- *[PUBLIC_KEY]*: path of the downloaded GPG public key.
- *[SIG_ID]*: obtained after running **gpg --import <PUBLIC_KEY>**. For details, see the following figure:



Step 6 Refresh the local cache.

```
apt-get update
```

----End

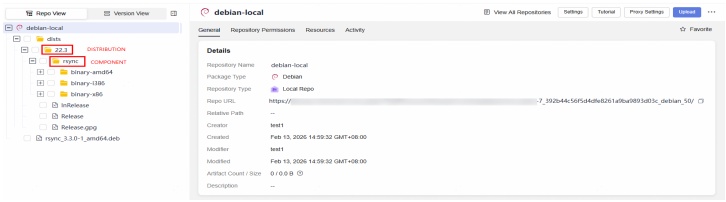
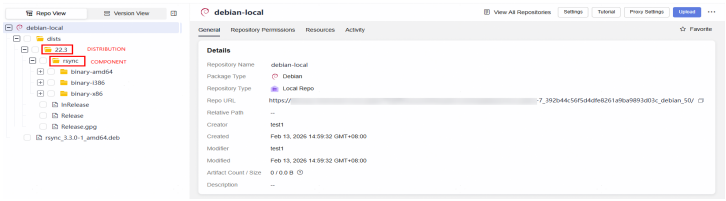
Publishing a Debian Package Using the Client

You can publish a Debian package to self-hosted repos on your local client by using the REST API. For example:

```
curl -k -u "<USERNAME>:<PASSWORD>" -X PUT " <REPO_URL>/<DEBIAN_PACKAGE_NAME>;deb.distribution=<DISTRIBUTION>;deb.component=<COMPONENT>;deb.architecture=<ARCHITECTURE>" -T <PATH_TO_FILE>
```

The table below describes the parameters.

Table 4-13 Parameters

Item	Description
REPO_URL	Repository address, which can be obtained from the configuration file.
DISTRIBUTION	Release version name, which is the name of the first-level directory under **dists**, as shown in the following figure. 
COMPONENT	Component name, which is the name of the second-level directory under **dists**, as shown in the following figure. 
USERNAME	Username, which can be obtained from the configuration file. For details, see **Debian Configuration**.
PASSWORD	User password, which can be obtained from the configuration file. For details, see **Debian Configuration**.
PATH_TO_FILE	Local file path.

Downloading a Debian Package via the Client

Step 1 Configure the current Debian repository as the apt source and add the GPG public key by referring to **Debian Configuration**.

Step 2 Download the package.

```
apt download <PACKAGE>
```

<PACKAGE> specifies the package name.

----End

4.6.8 Uploading and Downloading a Conan Package on the Client

CodeArts Artifact can connect to local Conan clients. You may upload your private packages from your local Conan client to self-hosted repos and share them with others, who can download these Conan packages through their clients.

Conan Package

Conan is a package manager for C and C++ developers. It allows you to manage project dependencies, compile and build processes, and distribute compiled binaries.

Constraints

The repository password for a self-hosted repo is account-specific. If you are prompted for a password while using a different account, download the configuration file from the repository's **Tutorial** tab to obtain it.

Prerequisites

- You have installed the Conan client.
- You have **created a Conan repository**.
- You have the **Download/View** permission on the current repository. For details about how to obtain this permission, see **Configuring Repository Permissions**.

Procedure

Step 1 **Access self-hosted repos** using your Huawei Cloud account.

Step 2 Select the target Conan repository on the self-hosted repo page.

Step 3 Click **Tutorial** in the upper right corner of the repository page.

----End

Tutorial

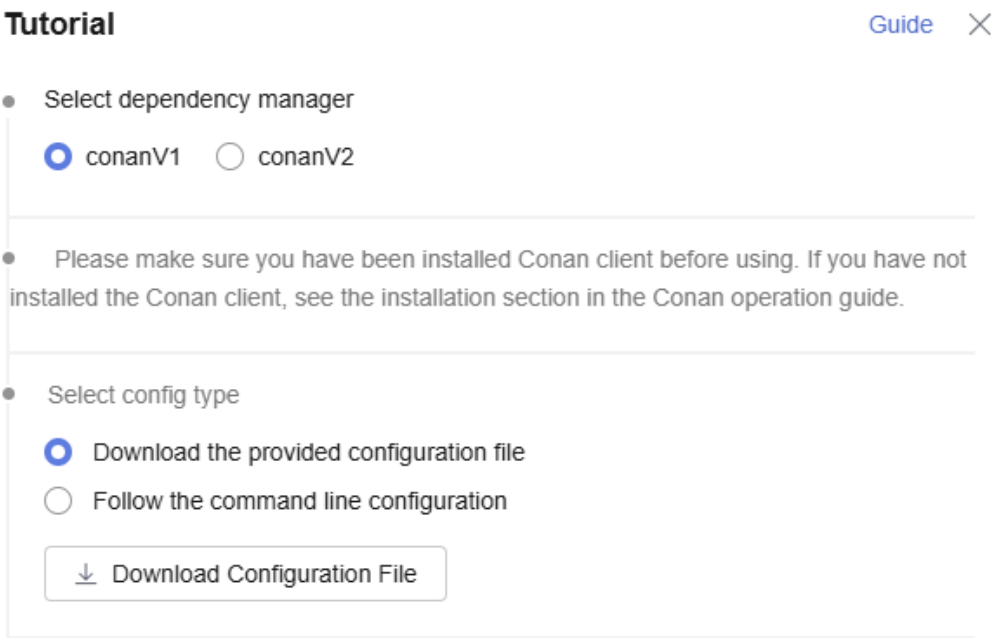
Adding a Conan repository by downloading provided configuration file

Edit the **remotes.json** configuration file for the Conan repository source. By default, the file is stored in the current user directory.

- Path for Conan V1 version: ***{{User}}*/.conan/remotes.json**
- Path for Conan V2 version: ***{{User}}*/.conan2/remotes.json**

Step 1 **Go to the Tutorial page of Conan repository**.

Step 2 In the displayed dialog box, select **Download the provided configuration file**.



Step 3 Add **{"name":"*{{xxx}}*","url":"https://*{{xxx}}*/artgalaxy/api/conan/*{{xxx}}*/","verify_ssl":false}** from the downloaded file to the configuration file. Note that setting **verify_ssl** to **false** means the HTTPS certificate is ignored.

The following is an example of the **remotes.json** file after modification:

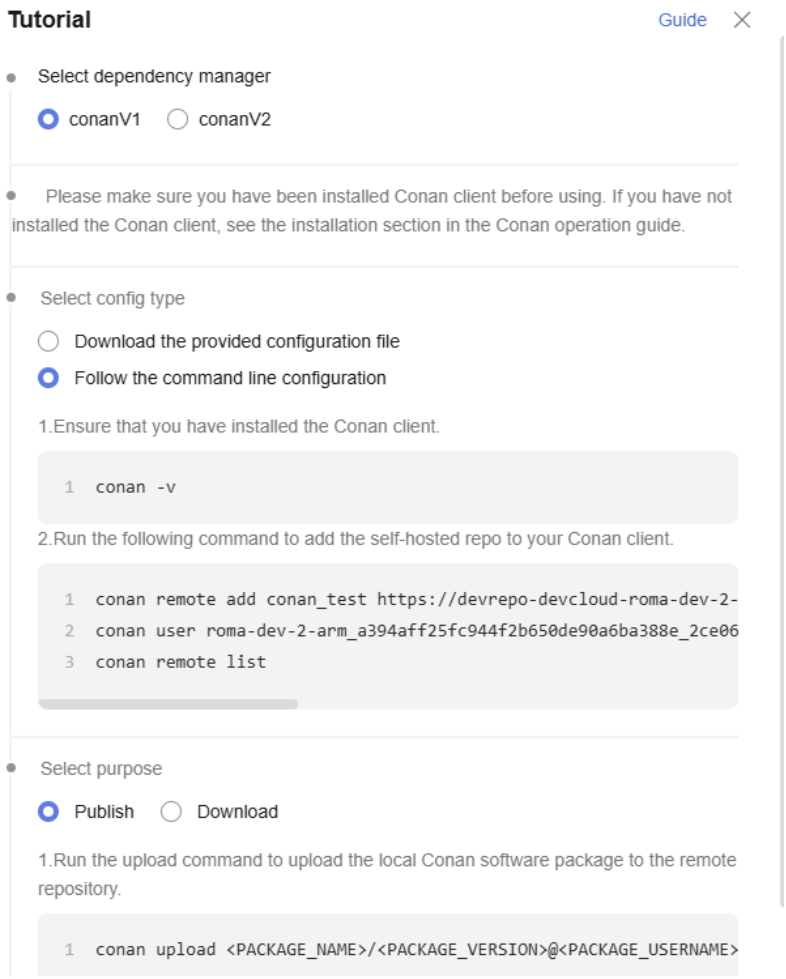
```
{
 "remotes": [
  {
   "name": "center",
   "url": "[redacted]"
   "verify_ssl": false
  },
  {
   "name": "mirror",
   "url": "http://[redacted]",
   "verify_ssl": true
  }
 ]
}
```

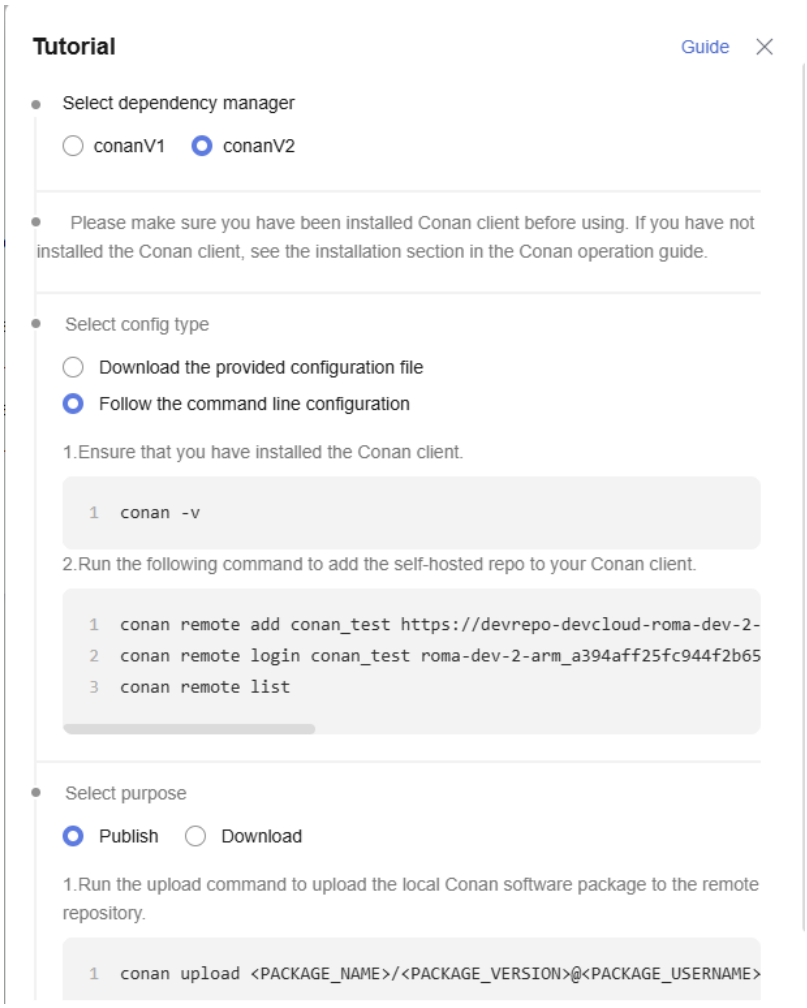
- For the Conan V1 client, run the **conan user** command to configure your account and password.
- For the Conan V2 client, run the **conan remote login** command to configure your account and password.

Step 4 After the configuration is successful, run the **conan remote list** command to verify that the repository appears in the list.

----End

Adding a Conan repository by following command-line configuration

Dependency Manager	Description
conanV1	Set **Select dependency manager** to **conanV1** and copy the command line to your local PC for execution. (The password is hidden by default; hover over the command area and click to automatically include the password in the command line.) 

Dependency Manager	Description
conanV2	Set **Select dependency manager** to **conanV2** and copy the command line to your local PC for execution. (The password is hidden by default; hover over the command area and click to automatically include the password in the command line.) **NOTE** When configuring the Conan repository on your local PC, ensure that the repository address starts with **api/conan**. Example: **https://{{ *Domain name*}}/artgalaxy/api/conan/{{ *Repository name*}}**. 

Publishing a Conan Package Using the Client

Step 1 Complete related configurations by referring to **Tutorial**.

Step 2 Upload the local Conan package to self-hosted repos.

```
conan upload <PATTERN> -r <REMOTE>
```


- Replace *<REMOTE>* with the repository alias you defined when adding the Conan repository by referring to "Adding a Conan repository by following command-line configuration" in **Tutorial**. For example, **my-conan-test**.
- Replace *<PATTERN>* with the package or recipe that you want to upload. The recipe format is ***<NAME>/<VERSION>@<USER>/<CHANNEL>***.

  Example: **conan upload "*" -r my-conan-test**

----End

Downloading a Conan Package to the Client

Step 1 Complete related configurations by referring to **Tutorial**.

Step 2 Download the dependencies declared in the **conanfile.txt** file.

```
conan install . -r <REMOTE>
```

Replace *<REMOTE>* with the repository alias you defined when adding the Conan repository by referring to "Adding a Conan repository by following command-line configuration" in **Tutorial**. For example, if the repository alias is **my-conan-test**, the command is **conan install . -r my-conan-test**.

----End

4.6.9 Uploading and Downloading a NuGet Package via the Client

CodeArts Artifact can connect to local NuGet clients. You may upload your private packages from your local NuGet client to self-hosted repos and share them with others, who can download these NuGet packages through their clients.

NuGet Package

NuGet is the package manager for the .NET ecosystem. It enables developers to publish, share, and consume .NET libraries and components. NuGet packages are the standard format for distributing these artifacts.

Constraints

The repository password for a self-hosted repo is account-specific. If you are prompted for a password while using a different account, download the configuration file from the repository's **Tutorial** tab to obtain it.

Prerequisites

- You have installed .NET and NuGet.
- You have **created a NuGet repository**.
- You have the **Download/View** permission on the current repository. For details about how to obtain this permission, see **Configuring Repository Permissions**.

Uploading a NuGet Package via the Client

Ensure that you have installed the NuGet.

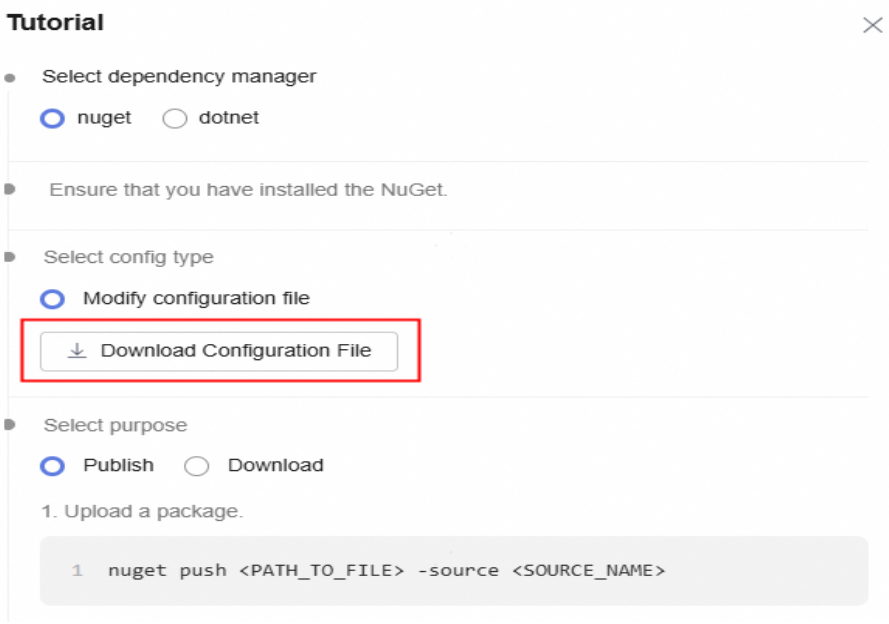
Step 1 **Access self-hosted repos** using your Huawei Cloud account.

Step 2 Select the target NuGet repository on the self-hosted repo page.

Step 3 Click **Tutorial** on the right of the page.

Step 4 In the displayed dialog box, set **Select dependency manager** to **nuget**.

Step 5 Click **Download Configuration File** to download the **NuGet.txt** configuration file.



Step 6 Open the downloaded file and run the following commands to add the source.

```
##----------------------NuGet add source----------------------##
nuget sources add -name {repo_name} -source {repo_url} -username {user_name} -password
{repo_password}
```

Step 7 Run the following commands to upload the package. Replace *<PATH_TO_FILE>* with the path of the file to be uploaded and run the upload statement. (If a configuration source exists, use the configured source name as the parameter following **-source**.)

```
##----------------------NuGet Download----------------------##
nuget push <PATH_TO_FILE> -source <SOURCE_NAME>
```

----End

Downloading a NuGet Package via the Client

Ensure that you have installed the NuGet.

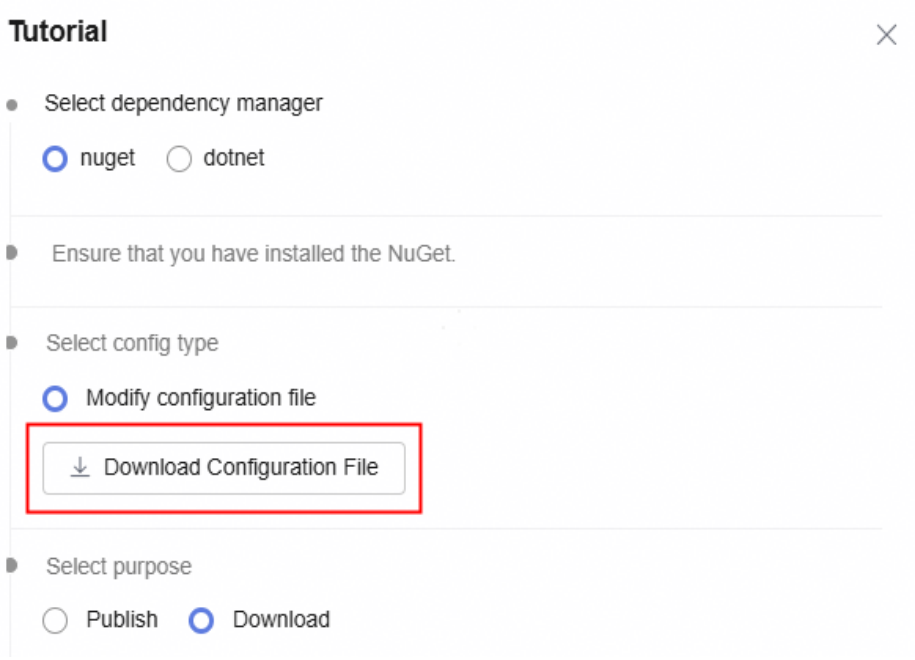
Step 1 **Access self-hosted repos** using your Huawei Cloud account.

Step 2 Select the target NuGet repository on the self-hosted repo page.

Step 3 Click **Tutorial** on the right of the page.

Step 4 In the displayed dialog box, set **Select dependency manager** to **nuget**.

Step 5 Click **Download Configuration File** to download the **NuGet.txt** configuration file.



Step 6 Open the file, find the command under **NuGet add source**, and add the source.

```
##----------------------NuGet add source----------------------##
nuget sources add -name {repo_name} -source{repo_url} -username {user_name} -password
{repo_password}
```

Step 7 Open the file, find the statement under **NuGet Download**, replace *<PACKAGE>* with the name of the package to be downloaded, and run the download statement. (If a configuration source exists, use the configured source name as the parameter following **-source**.)

```
##----------------------NuGet Download----------------------##
nuget install <PACKAGE>
```

----End

Uploading a .NET Package via the Client

Ensure that you have installed .NET.

You can use the .NET client only after you have added the trusted server certificate.

- To obtain the Windows trust certificate, perform the following steps:

  1. Export the server certificate.

  ```
  openssl s_client -connect {host}:443 -showcerts </dev/null 2>/dev/null | sed -ne '/-BEGIN
  CERTIFICATE-/,/-END CERTIFICATE-/p' |openssl x509 -outform PEM >mycertfile.pem
  openssl x509 -outform der -in mycertfile.pem -out mycertfile.crt
  ```

  mycertfile.pem and **mycertfile.crt** are the downloaded certificates.

  2. In Windows, use PowerShell to add the certificate trust.

  Add the certificates.

  ```
  Import-Certificate -FilePath "mycertfile.crt" -CertStoreLocation cert:\CurrentUser\Root
  ```

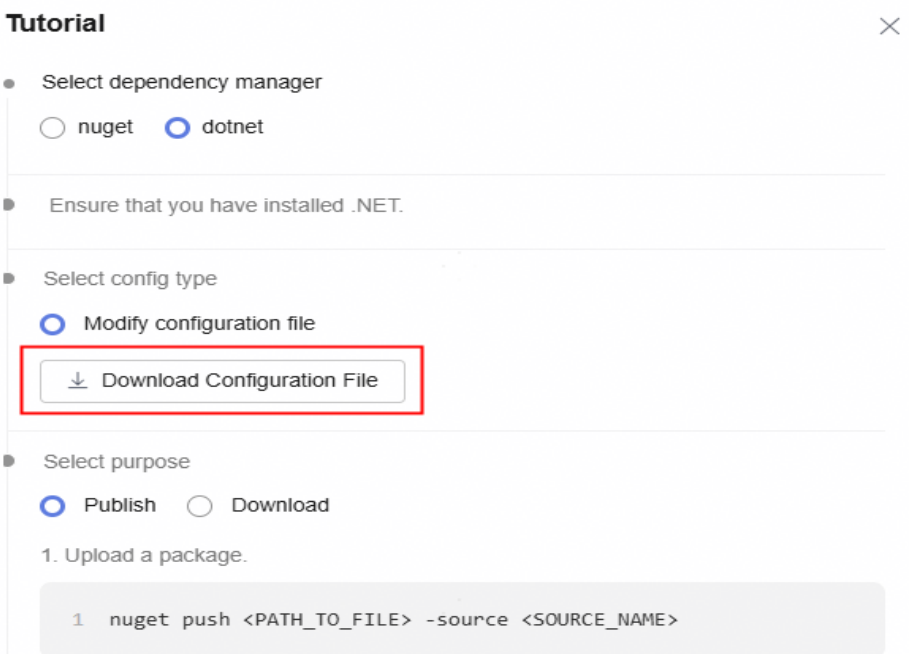
Step 1 **Access self-hosted repos** using your Huawei Cloud account.

Step 2 Select the target NuGet repository on the self-hosted repo page.

Step 3 Click **Tutorial** on the right of the page.

Step 4 In the displayed dialog box, set **Select dependency manager** to **dotnet**.

Step 5 Click **Download Configuration File** to download the **dotnet.txt** configuration file.



Step 6 Open the configuration file, find the command under **dotnet add source**, and add the source.

```
##----------------------dotnet add source----------------------##
dotnet nuget add source {repo_url} add -n {repo_name} -u {user_name} -p {repo_password}
```

Step 7 Find the statement under **dotnet upload**, replace *<PATH_TO_FILE>* with the path of the file to be uploaded, and run the upload statement.

```
##----------------------dotnet upload----------------------##
dotnet nuget push <PATH_TO_FILE> -s {repo_name}
```

----End

Downloading a .NET Package via the Client

Ensure that you have installed .NET.

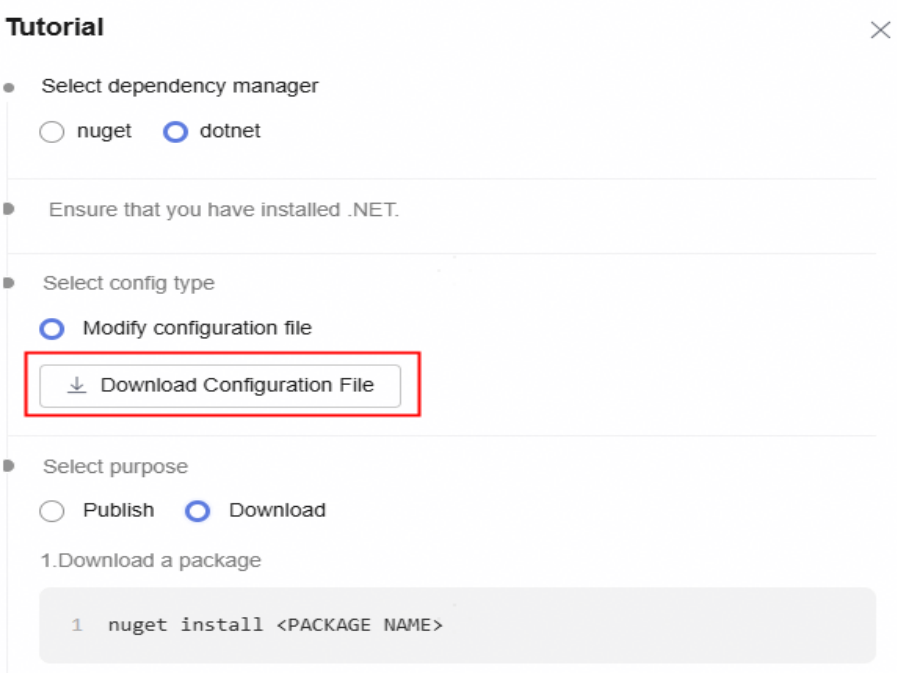
Step 1 **Access self-hosted repos** using your Huawei Cloud account.

Step 2 Select the target NuGet repository on the self-hosted repo page.

Step 3 Click **Tutorial** on the right of the page.

Step 4 In the displayed dialog box, set **Select dependency manager** to **dotnet**.

Step 5 Click **Download Configuration File** to download the **dotnet.txt** configuration file.



Step 6 Open the configuration file, find the command under **dotnet add source**, and add the source.

```
##----------------------dotnet add source----------------------##
dotnet nuget add source {repo_url} add -n {repo_name} -u {user_name} -p {repo_password}
```

Step 7 Find the statement under **dotnet download**, replace *<PACKAGE>* with the name of the package to be downloaded, and run the download statement.

```
##----------------------dotnet download----------------------##
dotnet add package <PACKAGE>
```

----End

4.6.10 Uploading and Downloading a CocoaPods Package on the Client

CodeArts Artifact can connect to local CocoaPods clients. You may upload your private packages from your local CocoaPods client to self-hosted repos and share them with others, who can download these CocoaPods packages through their clients.

CocoaPods Pod

CocoaPods is a dependency manager for iOS and macOS projects. It helps developers easily integrate and manage required libraries efficiently.

Constraints

The repository password for a self-hosted repo is account-specific. If you are prompted for a password while using a different account, download the configuration file from the repository's **Tutorial** tab to obtain it.

Prerequisites

- You have installed the Ruby client and cocoapods-art plug-in.
- You have **created a CocoaPods repository**.
- You have the **Download/View** permission on the current repository. For details about how to obtain this permission, see **Configuring Repository Permissions**.

Uploading a CocoaPods Package from the Client

Uploading a CocoaPods package to self-hosted repos by provided configuration file

Step 1 **Access self-hosted repos** using your Huawei Cloud account.

Step 2 In the left pane, click the target CocoaPods repository.

Step 3 Click **Tutorial** on the right of the page.

Step 4 In the displayed dialog box, select **Download the provided configuration file** and click **Download Configuration File** to download the **cocoapods.txt** file.

Step 5 Obtain *{username}* and *{password}* from the downloaded file.

Step 6 Run the following command on your local client to upload the local CocoaPods package to the target self-hosted repo.

```
curl -k -u "<USERNAME>:<PASSWORD>" -XPUT "{url}/<TARGET_FILE_PATH>/" -T <PATH_TO_FILE>/<FILE_NAME>
```

- *USERNAME*: *{username}* obtained in **Step 5**.
- *PASSWORD*: *{password}* obtained in **Step 5**.
- *url*: repository address of the CocoaPods repository.
- *TARGET_FILE_PATH*: name of the self-hosted repo folder to be stored.
- *PATH_TO_FILE*: local path of the package to be uploaded.
- *FILE_NAME*: name of the self-hosted repo to which the package is uploaded.

Step 7 Check the uploaded package in the CocoaPods repository.

----End

Uploading a CocoaPods package to self-hosted repos by command-line configuration

Step 1 **Access self-hosted repos** using your Huawei Cloud account.

Step 2 In the left pane, click the target CocoaPods repository.

Step 3 Click **Tutorial** on the right of the page.

Step 4 In the displayed dialog box, select **Follow the command line configuration**.

Step 5 Check whether the Ruby client has been installed.

```
ruby -v
```

Step 6 Install the cocoapods-art plug-in.

```
sudo gem install cocoapods-art
```

Step 7 Add the self-hosted repo to your CocoaPods client.


```
pod repo-art add <repo_name> "{url}"
```

- *repo_name*: name of the folder for storing packages of self-hosted repos on your local client.
- *url*: repository address of the CocoaPods repository.

Step 8 In the **Select purpose** area, click **Publish**.

Step 9 Upload the local package to the target CocoaPods repository.

```
curl -k -u "<USERNAME>:<PASSWORD>" -XPUT "{url}/<TARGET_FILE_PATH>/" -T <PATH_TO_FILE>/
<FILE_NAME>
```

- *USERNAME*: *{username}* obtained in **Step 5**.
- *PASSWORD*: *{password}* obtained in **Step 5**.
- *url*: repository address of the CocoaPods repository.
- *TARGET_FILE_PATH*: name of the self-hosted repo folder to be stored.
- *PATH_TO_FILE*: local path of the package to be uploaded.
- *FILE_NAME*: name of the self-hosted repo to which the package is uploaded.

Step 10 Check the uploaded package in the CocoaPods repository.

----End

Downloading a CocoaPods Package to the Client

Downloading a CocoaPods package by provided configuration file

Step 1 **Access self-hosted repos** using your Huawei Cloud account.

Step 2 Select the target CocoaPods repository from the self-hosted repo page and click **Tutorial** on the right of the page.

Step 3 In the displayed dialog box, select **Download the provided configuration file**.

Step 4 In the **Select purpose** area, click **Download**.

Step 5 Run the following commands to download files to your local client.

1. Download the package from the remote repository.

```
pod repo-art add {package_name} {url}
```

package_name: name of the CocoaPods dependency to be downloaded.

url: repository address of the self-hosted repo.

Change the address of the CocoaPods repository.

```
pod repo-art update {package_name} {url}
```

package_name: The CocoaPods dependency cannot be modified.

url: enter the repository address of the target self-hosted repo.

Query the downloaded dependency.

```
pod repo-art list
```

Remove the local dependency.

```
pod repo-art remove <repo-name>//repo_name: name of the CocoaPods dependency
```

----End

Downloading a CocoaPods package by command-line configuration

Step 1 **Access self-hosted repos** using your Huawei Cloud account.

Step 2 Select the target CocoaPods repository from the self-hosted repo page and click **Tutorial** on the right of the page.

Step 3 In the displayed dialog box, select **Follow the command line configuration**.

Step 4 Check whether the Ruby client has been installed.

```
ruby -v
```

Step 5 Install the cocoapods-art plug-in.

```
sudo gem install cocoapods-art
```

Step 6 Add the self-hosted repo to your CocoaPods client.

```
pod repo-art add <repo_name> "{url}"
```

- *repo_name*: name of the folder for storing packages of self-hosted repos on your local client.
- *url*: repository address of the CocoaPods repository.

Step 7 In the **Select purpose** area, click **Download**.

Step 8 Run the following commands to download files to your local client.

Download the package from the remote repository.

```
pod repo-art update {package_name}//package_name: package name.
```

Query the downloaded package.

```
pod repo-art list
```

Remove the local dependency.

```
pod repo-art remove <repo-name>//repo-name: name of the CocoaPods dependency
```

----End

4.6.11 Uploading and Downloading an OHPM Package on the Client

CodeArts Artifact can connect to local OHPM clients. You may upload your private packages from your local OHPM client to self-hosted repos and share them with others, who can download these OHPM packages through their clients.

OHPM Package

OpenHarmony Package Manager (OHPM) is an ArkTS package management tool. OHPM manages packages, which are stored and organized in the OHPM repository.

The OHPM package contains the Harmony Archive (HAR) package and Harmony Shared Package (HSP) package.

- HAR: Static shared package, which is reused in the build phase. It is mainly used as a second-party or third-party library for other applications to depend on. It includes resource files, *.so files, and metadata files.

- HSP: Dynamic shared package, which is reused in the running phase. It provides shared code and resources to improve code reusability and maintainability.

Constraints

The repository password for a self-hosted repo is account-specific. If you are prompted for a password while using a different account, download the configuration file from the repository's **Tutorial** tab to obtain it.

Prerequisites

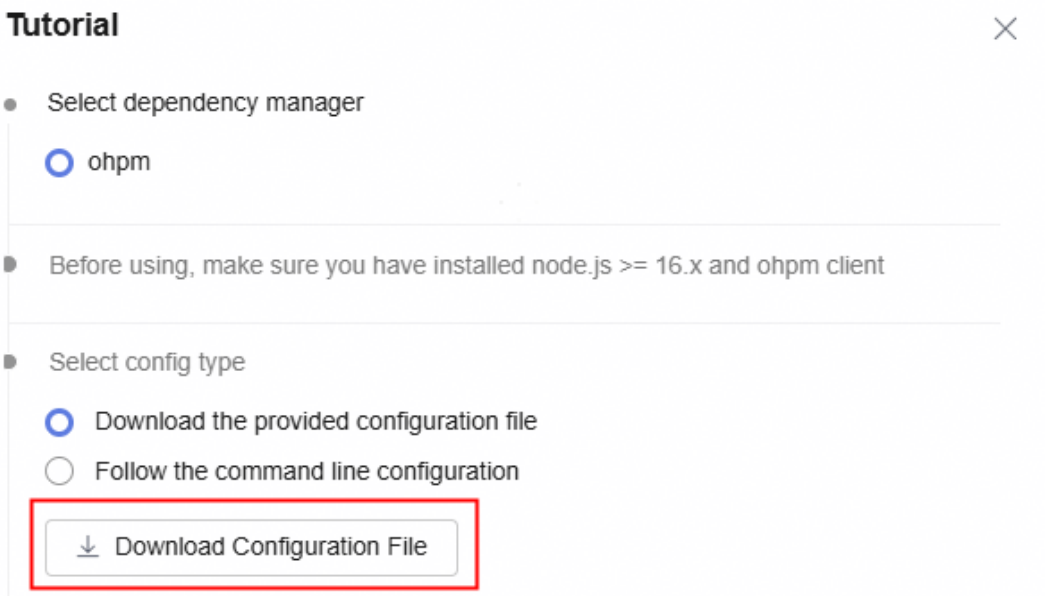
- The client tool is npm, and the dependency management tool is OHPM. You have installed node.js 16.x or later.
- You have **created an OHPM repository**.
- You have the **Download/View** permission on the current repository. For details about how to obtain this permission, see **Configuring Repository Permissions**.

Uploading an OHPM Package from the Client

Step 1 **Access self-hosted repos** using your Huawei Cloud account.

Step 2 Select the target OHPM repository from the self-hosted repo page and click **Tutorial** on the right of the page.

Step 3 In the displayed dialog box, download the **ohpmrc** file and rename it **.ohpmrc**.



Step 4 Copy the file to the user directory. The path is as follows:

- Linux: **~/.ohpmrc**
- Windows: **C:\Users\<*UserName*>\.ohpm\.ohpmrc**

Step 5 Upload the OHPM package to the repository.

```
ohpm publish {file path}\{package name}
```


- *file path*: path of the local OHPM package.
- *package name*: name of the local OHPM package.

----End

Downloading an OHPM Package Using the Client

Download the package Using the client.

```
ohpm install {package name}
```

package name: **name** parameter in the return value in **Step 5**, as shown in the following figure:

The following figure shows that the download is successful.

4.6.12 Uploading and Downloading a Docker Image on the Client

CodeArts Artifact can connect to local Docker clients. You may upload your private images from your local Docker client to self-hosted repos and share them with others, who can download these Docker images through their clients.

Docker Image

Docker artifacts refer to container images built and published using Docker. Docker images contain all dependencies and configurations required for running applications, ensuring portability and consistent behavior across environments.

Constraints

The repository password for a self-hosted repo is account-specific. If you are prompted for a password while using a different account, download the configuration file from the repository's **Tutorial** tab to obtain it.

Prerequisites

- You have installed the Docker client.
- You have **created a Docker registry**.
- You have the **Download/View** permission on the current registry. For details about how to obtain this permission, see **Configuring Repository Permissions**.

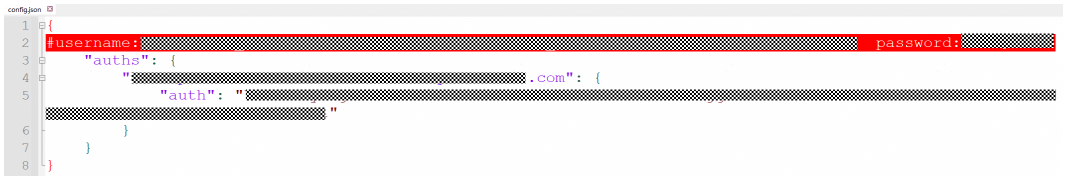
Uploading a Docker Image from the Client

Step 1 **Access self-hosted repos** using your Huawei Cloud account.

Step 2 Select the target Docker registry from the self-hosted repo page and click **Tutorial** on the right of the page.

Step 3 In the displayed dialog box, click **Download Configuration File** to download the **config.json** file.

Step 4 Obtain *{username}* and *{password}* from the downloaded file.



Step 5 Run the following command on your local client to log in to the Docker registry.

```
docker login {url} -u ${username} -p ${password}
```

Variables in the preceding command are as follows:

- *url*: repository address.
- *username*: *{username}* obtained in **Step 4**.
- *password*: *{password}* obtained in **Step 4**.

Step 6 Load a local Docker image.

```
docker load -i registry.k8s.io_pause.tar
```

Step 7 Query the name and version of a local Docker image.

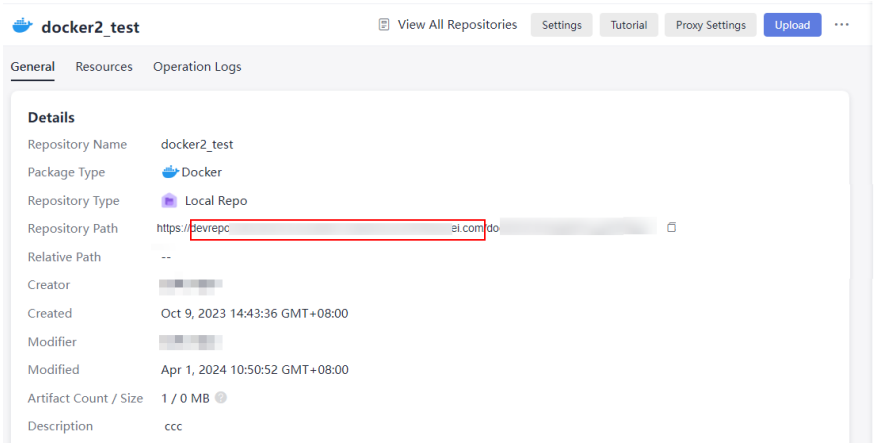
```
docker images
```


Step 8 Run the following command on your local client to package an image.

```
docker tag ${image_name1}:${image_version1} {url}/${image_name2}:${image_version2}
```

Variables in the preceding command are as follows:

- *image_name1*: name of the local Docker image. This is the image name obtained in **Step 7**.
- *image_version1*: version of the local Docker image. This is the image version number obtained in **Step 7**.
- *url*: repository address, as shown in the following figure:



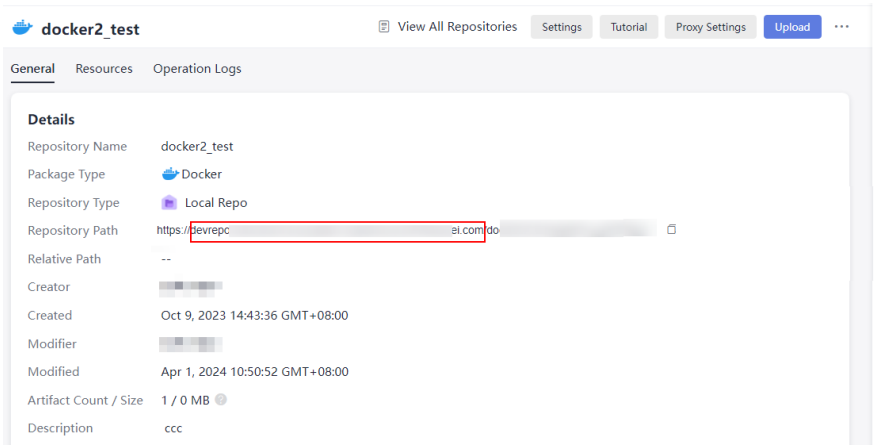
- *image_name2*: You can name the uploaded image. The image name is displayed in the image list of the Docker registry.
- *image_version2*: You can set the version of the uploaded image.

Step 9 Run the following command on the local client to upload the Docker image to the self-hosted repo.

```
docker push {url}/${image_name}:${image_version}
```

Variables in the preceding command are as follows:

- *url*: repository address, as shown in the following figure:



- *image_name*: Enter **image_name2** in **Step 8**.
- *image_version*: Enter **image_version2** in **Step 8**.

Step 10 Check the uploaded image in the Docker registry.

----End

Downloading a Docker Image to the Client

Step 1 **Access self-hosted repos** using your Huawei Cloud account.

Step 2 Select the target Docker registry on the self-hosted repo page.

Step 3 Click **Tutorial** on the right of the page.

Step 4 In the displayed dialog box, click **Download Configuration File** to download the **config.json** file.

Step 5 Obtain *{username}* and *{password}* from the downloaded file.

Step 6 Run the following command on your local client to log in to the Docker registry.

```
docker login {url} -u ${username} -p ${password}
```

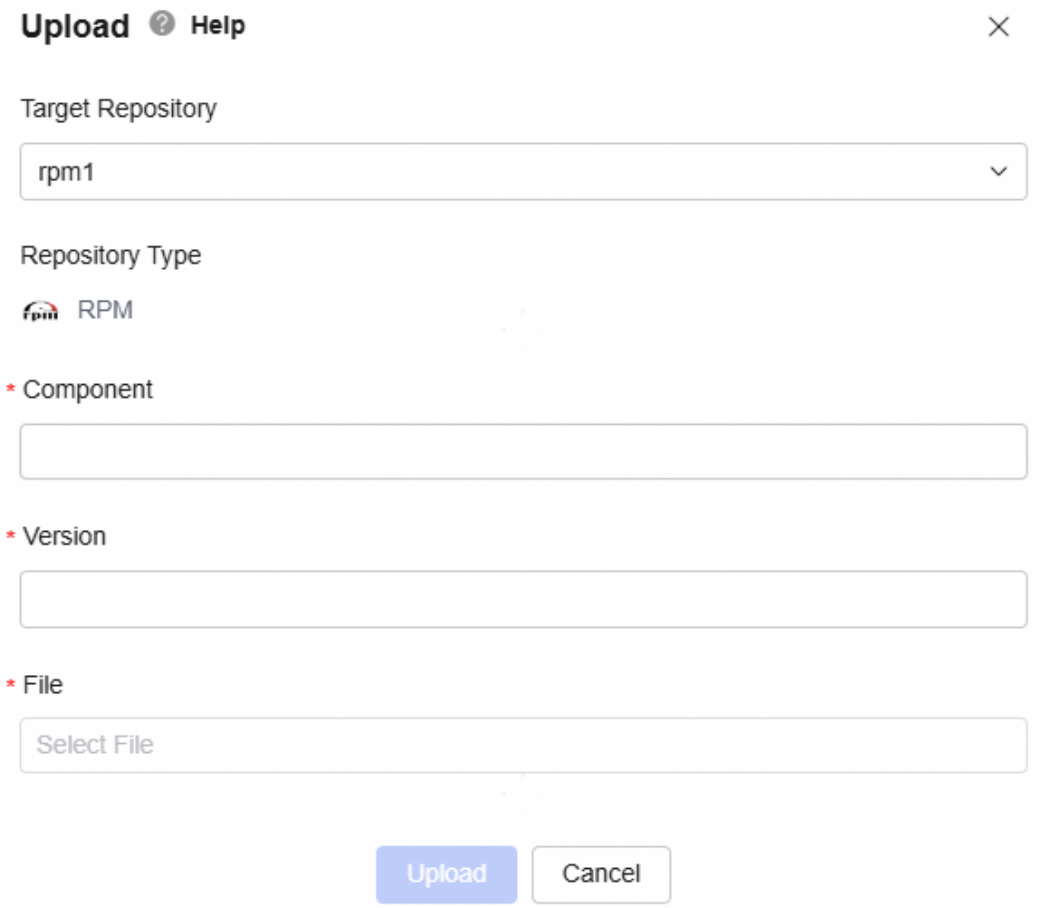
Variables in the preceding command are as follows:

- *url*: repository address.
- *username*: *{username}* obtained in **Step 4**.
- *password*: *{password}* obtained in **Step 4**.

Step 7 Run the following command on your local client to download the Docker image.

```
docker pull {url}/${image_name}:${image_version}
```

url: repository address, as shown in the following figure:



- *image_name*: image name.
- *image_version*: image version.

----End

4.6.13 Uploading and Downloading a Generic Artifact via the Client

CodeArts Artifact can connect to local clients. You may upload your private packages from your local client to self-hosted repos and share them with others, who can download these packages through their clients.

Generic Artifact

Generic artifacts (also called common artifacts) are files of any type generated during the build and release process.

Constraints

The repository password for a self-hosted repo is account-specific. If you are prompted for a password while using a different account, download the configuration file from the repository's **Tutorial** tab to obtain it.

Prerequisites

- You have **created a Generic repository**.
- You have the **Download/View** permission on the current repository. For details about how to obtain this permission, see **Configuring Repository Permissions**.

Uploading a Generic Artifact via the Client

Step 1 **Access self-hosted repos** using your Huawei Cloud account.

Step 2 Select the target Generic repository on the self-hosted repo page.

Step 3 Click **Tutorial** on the right of the page.

Step 4 In the displayed dialog box, click **Download Guide File** to download the **generic.txt** file.

Step 5 Upload the Generic artifact to the repository.

```
curl -k -u "{{username}}:{{password}}" -X PUT {{repo_url}}/{{filePath}} -T {{localFile}}
```

- *file path*: path (including the name) of the Generic repository to be uploaded.
- *localFile*: path (including the name) of the local Generic artifact.
- Obtain the values of *username*, *password*, and *repo_url* from the downloaded **generic.txt** file in **Step 4**, as shown in the following figure:

----End

Downloading a Generic Artifact Using the Client

Step 1 **Access self-hosted repos** using your Huawei Cloud account.

Step 2 Select the target Generic repository on the self-hosted repo page.

Step 3 Click **Tutorial** on the right of the page.

Step 4 In the displayed dialog box, click **Download Guide File** to download the **generic.txt** file.

Step 5 Download the package using the client.

```
curl -o {{localFileName}} -k -u "{{username}}:{{password}}" -X GET {{repo_url}}/{{filePath}}
```

- *localFileName*: local path (including the name) for downloading the Generic artifact.

- *filePath*: path (including the name) of the package in the Generic repository.
- Obtain the values of *username*, *password*, and *repo_url* from the downloaded **generic.txt** file in **Step 4**, as shown in the following figure:

----End

4.6.14 Uploading and Downloading a Composer Package on the Client

CodeArts Artifact can connect to local Composer clients. You may upload your private packages from your local Composer client to self-hosted repos and share them with others, who can download these Composer packages through their clients.

Composer Package

Composer is a dependency manager for the PHP programming language. It allows you to declare the libraries your project depends on and manage the installation of these dependencies. These dependencies are usually referred to as packages or composer packages.

Constraints

The repository password for a self-hosted repo is account-specific. If you are prompted for a password while using a different account, download the configuration file from the repository's **Tutorial** tab to obtain it.

Prerequisites

- You have installed the PHP and Composer clients.
- You have **created a Composer repository**.
- You have the **Download/View** permission on the current repository. For details about how to obtain this permission, see **Configuring Repository Permissions**.

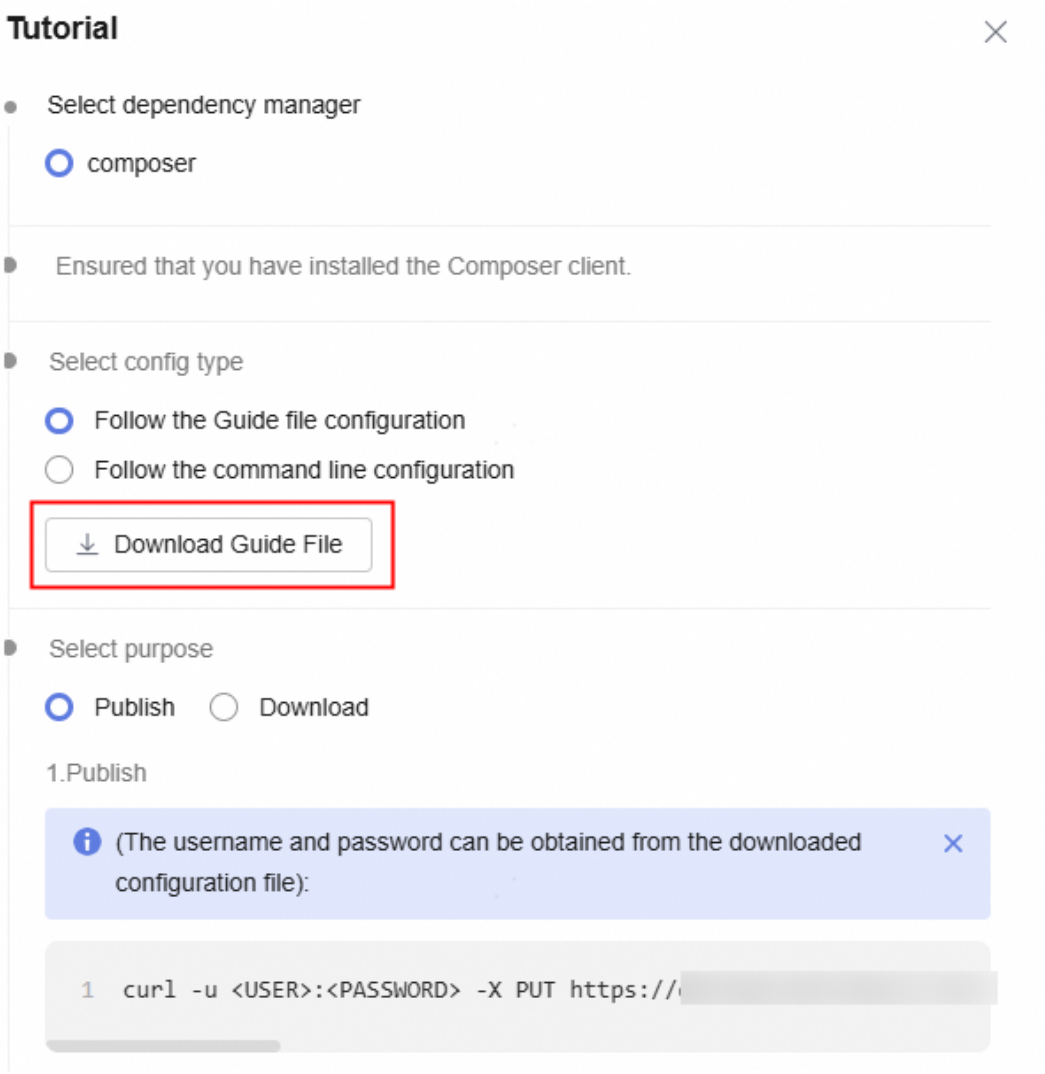
Uploading a Composer Package from the Client

Step 1 **Access self-hosted repos** using your Huawei Cloud account.

Step 2 Select the target Composer repository on the self-hosted repo page.

Step 3 Click **Tutorial** on the right of the page.

Step 4 In the displayed dialog box, click **Download Guide File** to download the **composer.json** file.



Step 5 Obtain *{username}* and *{password}* from the downloaded file.

Step 6 Run the following command on your local client to upload the local Composer package to the target self-hosted repo.

```
curl -u {{user}}:{{password}} -X PUT https://{{url}}/{{vendor_name}}/{{project_name}}/{{version}}/{{vendor_name}}-{{project_name}}-{{version}}.zip;composer.version={{version}} -T {{localFile}}
```

- *user*: *{username}* obtained in **Step 5**.
- *password*: *{password}* obtained in **Step 5**.
- *url*: repository address of the Composer repository.
- *vendor_name*: vendor name of the software package.
- *project_name*: project name of the software package.
- *version*: software package version.
- *localFile*: local path for storing the software package.
- Check the uploaded package in the Composer repository.

----End

Downloading a Composer Package to the Client

Step 1 **Access self-hosted repos** using your Huawei Cloud account.

Step 2 Select the target Composer repository on the self-hosted repo page.

Step 3 Click **Tutorial** on the right of the page.

Step 4 In the displayed dialog box, select **Follow the Guide file configuration**.

Step 5 Replace the repository and credential configurations according to the configuration file.

Step 6 If there is no Composer project, initialize the project.

```
composer init
```

Step 7 Download the software package.

Add the software package dependency declaration.

```
composer require <vendor-name>/<project-name>:<version>
```

Download the installing dependency.

```
composer install
```

----End

4.6.15 Uploading and Downloading a Helm Chart on the Client

CodeArts Artifact can connect to local Helm clients. You may upload your private packages from your local Helm client to self-hosted repos and share them with others, who can download these Helm packages through their clients.

Helm Chart

Helm is a package manager for Kubernetes. Helm Charts enable you to manage and deploy Kubernetes applications.

Constraints

The repository password for a self-hosted repo is account-specific. If you are prompted for a password while using a different account, download the configuration file from the repository's **Tutorial** tab to obtain it.

Prerequisites

- You have installed the Helm client.
- You have **created a Helm repository**.
- You have the **Download/View** permission on the current repository. For details about how to obtain this permission, see **Configuring Repository Permissions**.

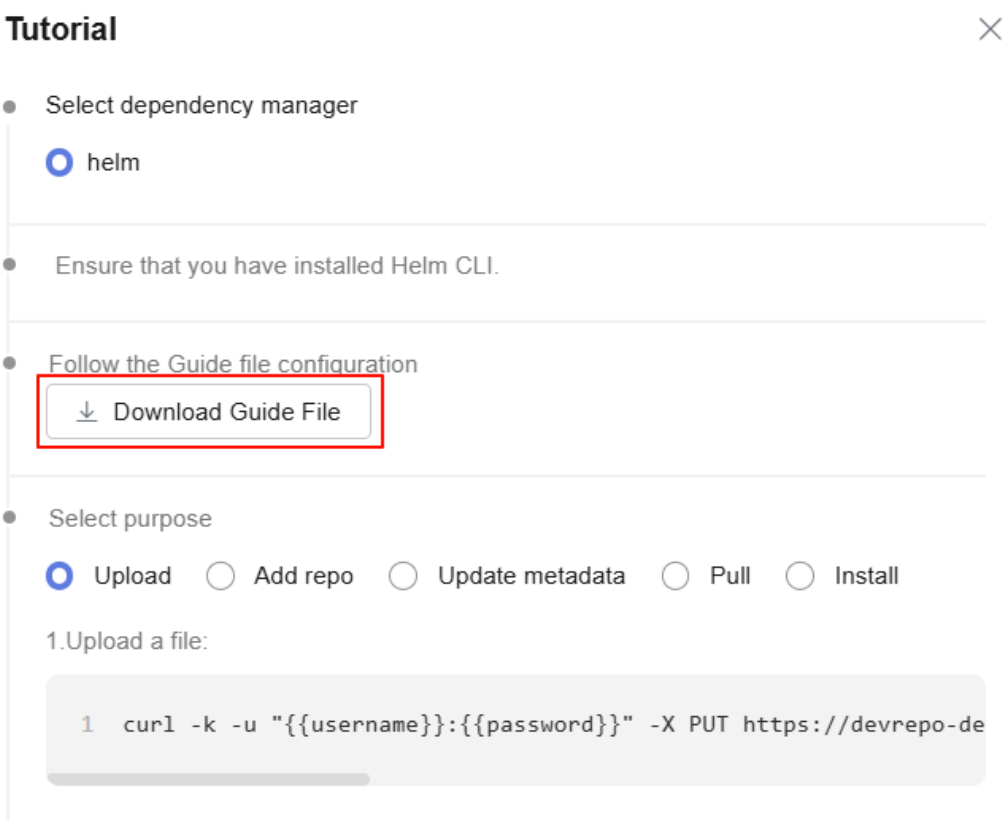
Uploading a Helm Chart from the Client

Step 1 Access self-hosted repos using your Huawei Cloud account.

Step 2 Select the target Helm repository on the self-hosted repo page.

Step 3 Click **Tutorial** on the right of the page.

Step 4 In the displayed dialog box, click **Download Guide File** to download the **repositories.yaml** file.



Step 5 Upload the Helm chart to the repository.

```
curl -k -u "{{username}}:{{password}}" -X PUT url/{{filePath}} -T {{localFile}}
```

- *file path*: path (including the name) of the Helm repository to be uploaded.
- *localFile*: path (including the name) of the local Helm chart.
- Obtain the values of *username*, *password*, and *url* from the **repositories.yaml** file downloaded in **Step 4**, as shown in the following figure:

----End

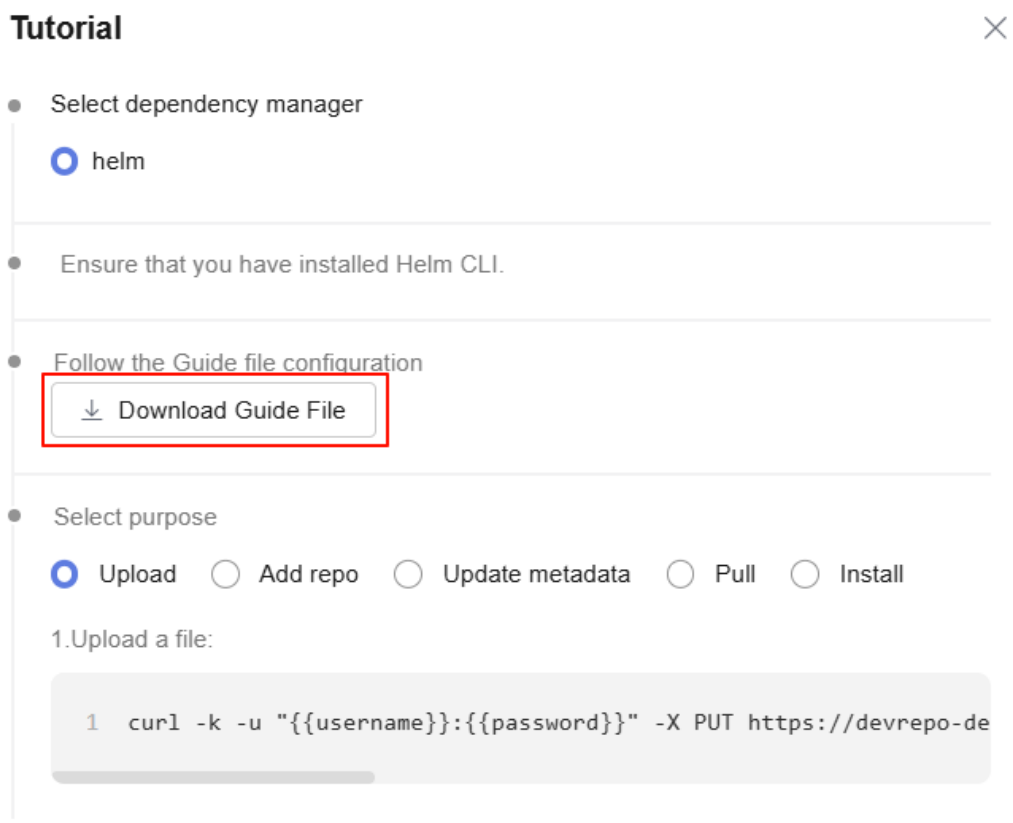
Downloading a Helm Chart to the Client

Step 1 Access self-hosted repos using your Huawei Cloud account.

Step 2 Select the target Helm repository on the self-hosted repo page.

Step 3 Click **Tutorial** on the right of the page.

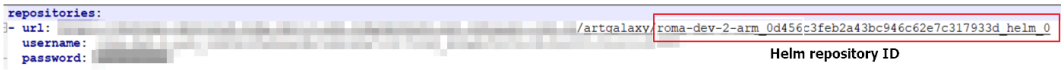
Step 4 In the displayed dialog box, click **Download Guide File** to download the **repositories.yaml** file.



Step 5 Add a Helm repository.

```
helm repo add Helm repository ID Helm repository URL --username {{username}} --password {{password}} --insecure-skip-tls-verify
```

Obtain the values of *username* and *password* from the downloaded **repositories.yaml** file in **Step 4**. Obtain the *Helm repository ID* from the red box in the following figure:



Step 6 Update the metadata.

```
helm repo update
```

Step 7 Download the chart.

```
helm pull Helm repository ID/{{filePath}} --version {{version}} --insecure-skip-tls-verify
```

Obtain the values of *username* and *password* from the **repositories.yaml** file downloaded in **Step 4**, as shown in the following figure:

----End

4.6.16 Uploading and Downloading a Conda Package via the Client

CodeArts Artifact can connect to local Conda clients. You may upload your private packages from your local Conda client to self-hosted repos and share them with others, who can download these Conda packages through their clients.

Conda Package

Conda is an open-source package manager and environment management system widely used in scientific computing, data science, and machine learning. It supports Windows, macOS, and Linux, and manages packages for Python and other languages such as R, Java, and C++. Conda artifacts are packages published via Conda.

Constraints

The repository password for a self-hosted repo is account-specific. If you are prompted for a password while using a different account, download the configuration file from the repository's **Tutorial** tab to obtain it.

Prerequisites

- You have installed the Conda client.
- You have **created a Conda repository**.
- You have the **Download/View** permission on the current repository. For details about how to obtain this permission, see **Configuring Repository Permissions**.

Uploading a Conda Package via the Client

Step 1 **Access self-hosted repos** using your Huawei Cloud account.

Step 2 Select the target Conda repository on the self-hosted repo page.

Step 3 Click **Tutorial** on the right of the page.

Step 4 In the displayed dialog box, click **Download Configuration File** to download the **condarc** file.

Step 5 Upload the Conda package to the repository.

```
curl -k -u "<USERNAME>:<PASSWORD>" -X PUT <REPO_URL>/<FILE_PATH>" -T <PACKAGE_NAME>
```

- *FILE_PATH*: path (including the name) of the Conda repository to be uploaded.
- *PACKAGE_NAME*: path (including the name) of the local Conda package.
- Obtain the values of *USERNAME*, *PASSWORD*, and *REPO_URL* from the downloaded **condarc** file in **Step 4**. In the **condarc** file, the values are provided in **channels**, in the **https://** *<USERNAME>*:*<PASSWORD>*@*<REPO_URL>* format. The value of *PASSWORD* must be decoded in the URL encoding format, as shown below.

----End

Downloading a Conda Package via the Client

Step 1 **Access self-hosted repos** using your Huawei Cloud account.

Step 2 Select the target Conda repository on the self-hosted repo page.

Step 3 Click **Tutorial** on the right of the page.

Step 4 In the displayed dialog box, click **Download Configuration File** to download the **condarc** file.

Step 5 Download the package via the client.

```
conda install {package name}
```

In an offline development environment, the repo.anaconda.com repository is inaccessible.

Edit the **hosts** file and enter the IP address of the repository's independent domain name, as shown in the figure.

package name: **path** parameter in the **Uploading a Conda Package via the Client**.

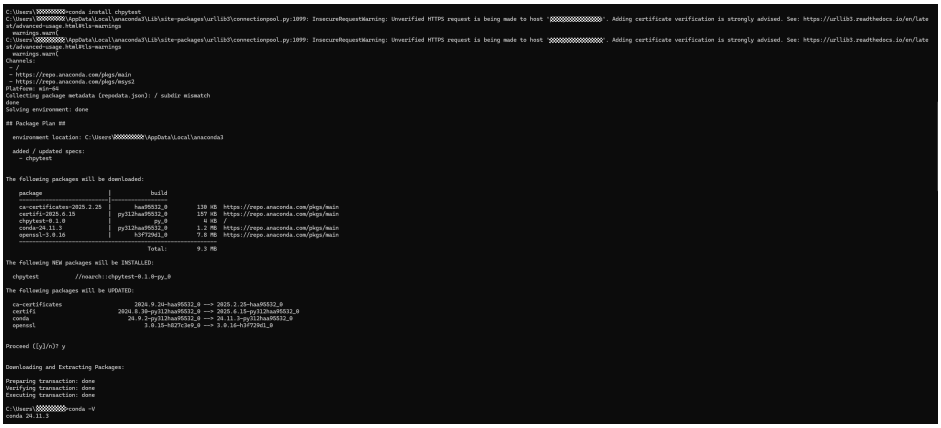
To ignore the certificate, set **ssl_verify** to **false** in the configuration file.

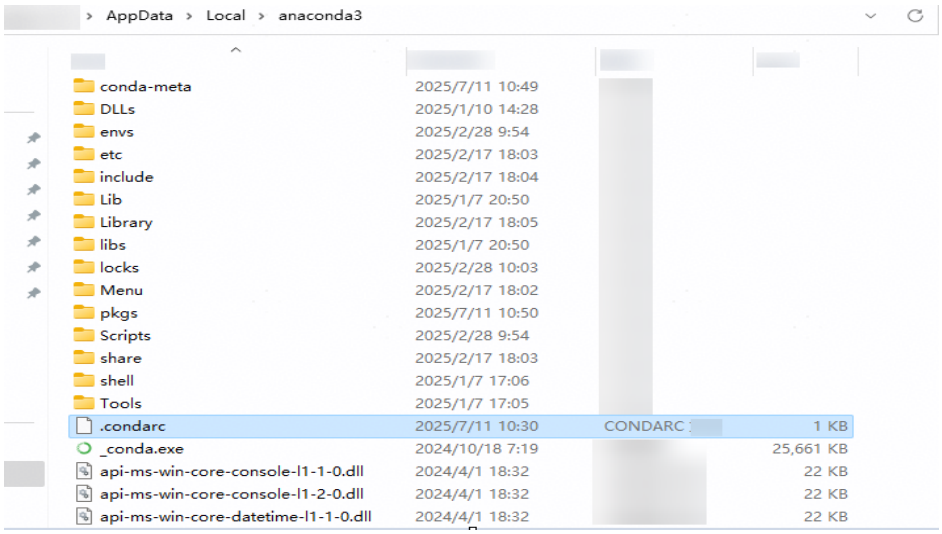
The following figure shows that the download is successful.


```
C:\Users\        }>conda install  pypdf2
Channels:
 - defaults
Platform: win-64
Collecting package metadata (repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: C:\Users\(        }\AppData\Local\anaconda3

  added / updated specs:
    - pypdf2


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    pypdf2-3.0.1               |  py312haa95532_0        12.4 MB
    ------------------------------------------------------------
                                           Total:        12.4 MB

The following NEW packages will be INSTALLED:

  pypdf2             pkgs/main/win-64::pypdf2-3.0.1-py312haa95532_0


Proceed ([y]/n)? y


Downloading and Extracting Packages:

Preparing transaction: done
Verifying transaction: done
Executing transaction: done
```

Common problems

- If the package exists in the repository but fails to be downloaded, update Conda to the latest version.
- If the installation prints the default official channel instead of the expected one, edit the **.condarc** file in **C:\Users*{user}*\AppData\Local\anaconda3**.





----End

4.7 Managing Packages

4.7.1 Viewing a Package

Viewing a Package in the Repo View

After you access the self-hosted repo, the repo view is displayed by default. The uploaded packages are stored in their respective folders within this view.

Step 1 **Access self-hosted repos** using your Huawei Cloud account.

Step 2 Click in front of the repository and folder to find the package.

Step 3 Click the package name. The repository details and checksums page are displayed.

Click in the **Details** and **Checksums** tabs to copy the information. In the search box, find the target package using the pasted information.

----End

Viewing a Package in the Version View

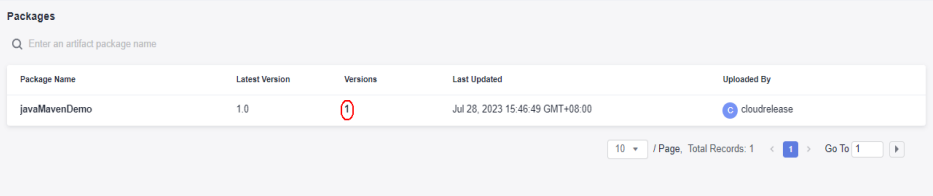
A self-hosted repo displays different package types by version. In **Version View** tab, you can filter and check artifacts by name and version number, and sort files by their update time.

Step 1 **Access self-hosted repos** using your Huawei Cloud account.

Step 2 Select **Version View** in the upper left corner of the page and click a repository name in the list on the left. The package version list is displayed. For details about how to set versions for different packages, see **Uploading/Downloading Packages on the Self-Hosted Repo Page**.

Step 3 Self-hosted repos host packages of different versions with the same name in the same file. Click a name in the **Package Name** column. The overview information about the latest version of the package is displayed.

Step 4 Click a number in the **Versions** column. The version list of the package is displayed.



Click a number in the **Version No.** column. The **Overview** and **Files** pages of the package are displayed. In the **Files** tab, click a name in the **File Name** column. The path of the package is displayed.

----End

4.7.2 Editing a Package

Searching for a Package

Step 1 Access self-hosted repos using your Huawei Cloud account.

Step 2 Click **Advanced Search** in the upper right corner of the page.

Step 3 Select the type of package to be searched for in the upper part of the page. By default, all types are selected. You can search for the package using either of the following methods:

- Select **File Name** mode.
  a. Enter the keyword of the file name in the search box, and click to search for the package.
  b. In the search result list, click a file name to view the package details.
- Select **Checksums** mode.
  a. Click the drop-down list on the left of the search box and select **Checksums** (The default value is **File Name**).
  b. Enter the **MD5/SHA-1/SHA-256/SHA-512 checksum** and click to find the desired package.

----End

Downloading a Package

Step 1 Access self-hosted repos using your Huawei Cloud account.

Step 2 In the left pane, locate the package to be downloaded and click its name.

If there are too many repositories or packages, you can search for the desired one by referring to **Searching for a Package**.

Step 3 Click **Download** on the right of the page.

----End

Deleting a Package

Step 1 Access self-hosted repos using your Huawei Cloud account.

Step 2 In the left pane, locate the package to be deleted and click its name.

If there are too many repositories or packages, you can search for the desired one by referring to **Searching for a Package**.

Step 3 Click **Delete** on the right of a package or directory in the repository to delete it.

Step 4 In the displayed dialog box, enter the name of the package to be deleted and click **OK**.

----End

Favoriting/Unfavoriting

Step 1 Access self-hosted repos using your Huawei Cloud account.

Step 2 In the left pane, locate the package to be favorited and click its name.

If there are too many repositories or packages, you can search for the desired one by referring to **Searching for a Package**.

Step 3 Click **Favorite** on the right of the page.

When the icon changes to  , click **My favorites** in the lower left corner of the page to view the list of favorited items. Click a value in the **Path** column to go to the package details page.

----End

4.8 Recycle Bin

Repositories and packages deleted from a self-hosted repo are moved to the recycle bin, where you can manage them.

The self-hosted repo provides the recycle bin entry both from the homepage and project pages.

Homepage Recycle Bin

You can manage all deleted items in the recycle bin on the homepage.

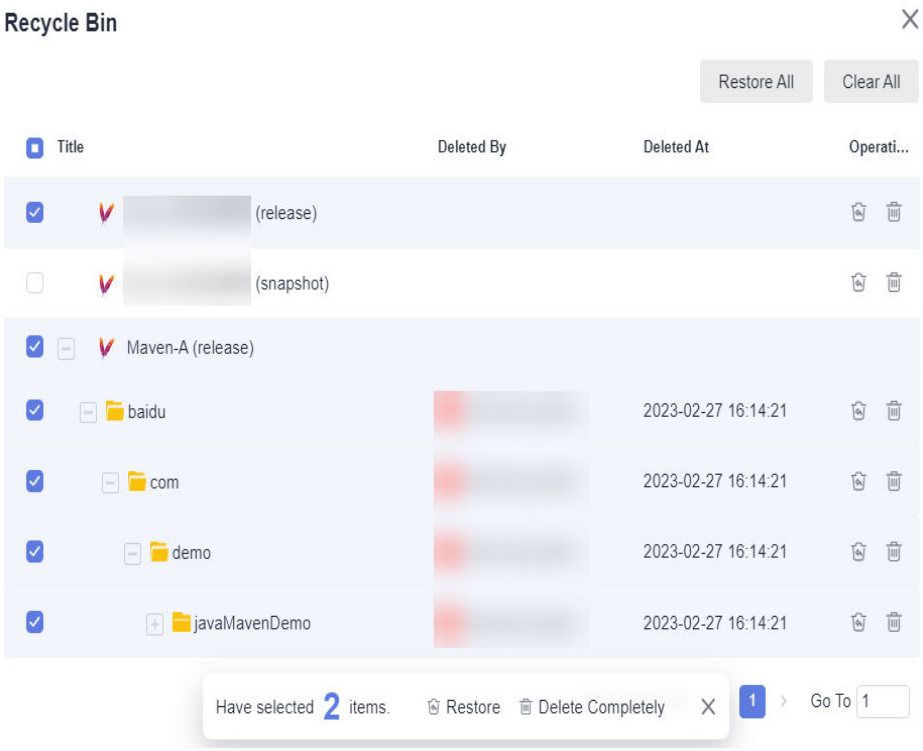
Step 1 Access self-hosted repos using your Huawei Cloud account.

Step 2 Click **Recycle Bin** in the upper right corner of the page, which will appear on the right side.

Step 3 Delete or restore the repositories and packages in the list as required.

If both icons  and  are displayed in the **Operation** column, the repository in that row has been deleted. If only one icon is displayed, the repository still exists,

but its packages have been deleted. Click the repository name to view the deleted packages.



Operations are as follows:

Operation Type	Operation	Description
Restore	Restore a repository	Click in the **Operation** column to restore the repository.
	Restore a package	Go to the repository where the package to restore is located, and click in the **Operation** column to restore it.
	Batch restore packages	Go to the repository where the packages to restore are located, select them, and click **Restore** below the list to restore them in batches.
	Restore all	Click **Restore All** in the upper right corner of the page to restore all selected items in the recycle bin.
Delete	Delete a repository	Click in the **Operation** column to delete the repository.
	Delete a package	Go to the repository where the package to delete is located, and click in the **Operation** column to delete it.

Operation Type	Operation	Description
	Batch delete packages	Go to the repository where the packages to delete are located, select them, and click **Permanently Delete** below the list to permanently delete them in batches.
	Clear all	Click **Clear All** in the upper right corner of the page to delete all selected items in the recycle bin.

----End

Project-Level Recycle Bin

Step 1 Go to the CodeArts homepage.

1. Log in to the **CodeArts console**, click , and select a region where you have enabled CodeArts.
2. Click **Go to Workspace**.

   If your account uses the old billing mode (see **Old Billing Modes**), click **Access Service**.

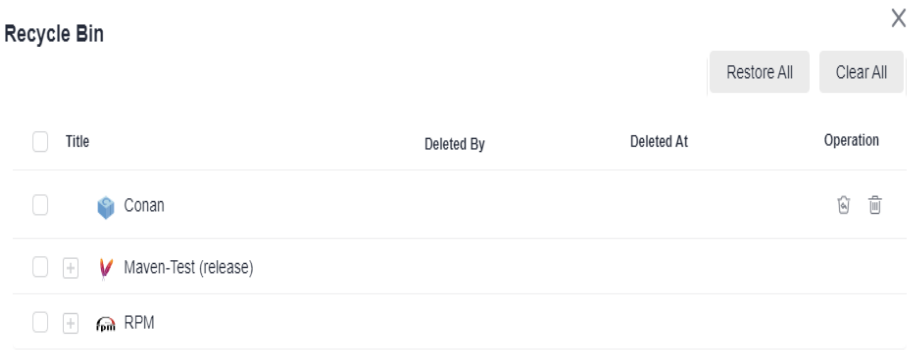
Log in to the **CodeArts console**, and click **Access Service**.

Step 2 Click a project card to access the project, choose **Artifact** > **Self-hosted Repos** from the navigation pane, and click the target repository name.

Step 3 Click **Recycle Bin** in the lower left corner of the page. The **Recycle Bin** page is displayed on the right.

Step 4 Delete or restore the repositories and packages in the list as required.

If both icons and are displayed in the **Operation** column, the repository in that row has been deleted. If only one icon is displayed, the repository still exists, but its packages have been deleted. Click the repository name to view the deleted packages.



Operations are as follows:

Operation Type	Operation	Description
Restore	Restore a repository	Click  in the **Operation** column to restore the repository.
	Restore a package	Go to the repository where the package to restore is located, and click  in the **Operation** column to restore it.
	Batch restore packages	Go to the repository where the packages to restore are located, select them, and click **Restore** below the list to restore them in batches.
	Restore all	Click **Restore All** in the upper right corner of the page to restore all selected items in the recycle bin.
Delete	Delete a repository	Click  in the **Operation** column to delete the repository.
	Delete a package	Go to the repository where the package to delete is located, and click  in the **Operation** column to delete it.
	Batch delete packages	Go to the repository where the packages to delete are located, select them, and click **Permanently Delete** below the list to permanently delete them in batches.
	Clear all	Click **Clear All** in the upper right corner of the page to delete all selected items in the recycle bin.

----End

5 Release Repos (Old)

5.1 Accessing Release Repos

CodeArts Artifact was upgraded in March 2023. Inventory release repos and resources belonging to users who registered before this update are stored in the old release repos.

If you have existing repositories and resources, you can upload and manage software packages in the old release repos. After you create a project, Release repo with the same project name is automatically generated in the new release repos.

Procedure

Step 1 Go to the CodeArts homepage.

1. Log in to the **CodeArts console**, click , and select a region where you have enabled CodeArts.
2. Click **Go to Workspace**.

   If your account uses the old billing mode (see **Old Billing Modes**), click **Access Service**.

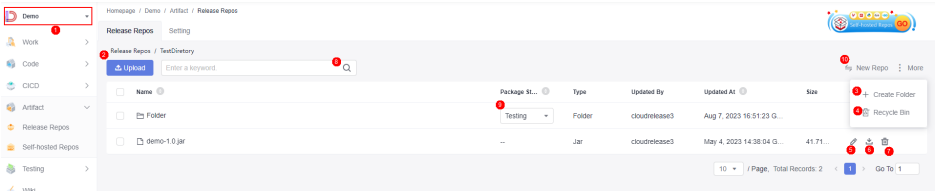
Log in to the **CodeArts console**, and click **Access Service**.

Step 2 Click a project card to access the project.

Step 3 In the navigation pane, choose **Artifact** > **Release Repos**.

Step 4 Click **Old Edition** in the lower left corner.

Step 5 View the existing file list of the current project. You can perform the following operations as required:



No.	Operation	Description
1	Switch project	Click the project name drop-down list to switch the project to view the old release repos.
2	Upload	Click **Upload** in the upper left corner of the list to manually upload local packages to release repos. The size of each file to upload cannot exceed 2 GB. **You are advised not to upload files containing sensitive information such as plaintext accounts and passwords to the release repos.**
3	Create folder	Click **More** and select **Create Folder** in the upper right corner to create a folder on the current page.
4	Recycle bin	Click **More** and select **Recycle Bin** in the upper right corner to access the recycle bin, where you can manage deleted packages or folders.
5	Edit	Click in the **Operation** column to edit the package or folder. • Package: You can change the name and release version of the package. (The release version archived by CodeArts Build is the build sequence number by default.) • Folder: You can edit the folder name.
6	Download	Click in the **Operation** column to download the package.
7	Delete	• Click in the **Operation** column. In the displayed dialog box, click **Move to Recycle Bin** to move the corresponding package or folder to the recycle bin, or click **Permanently Delete** to delete it permanently. • Select multiple packages or folders. In the displayed dialog box, click **Move to Recycle Bin** to move them to the recycle bin, or click **Permanently Delete** to delete them permanently. **Packages in the recycle bin still use space in release repos. The space will be freed up only after a package is permanently deleted from the recycle bin.**
8	Search	Enter a keyword (a folder or file name) in the search box above the list and click to search for the package whose name contains the keyword.

No.	Operation	Description
9	Change status	After entering a level-1 folder, you can change the status of a level-2 folder by clicking the drop-down list in the **Package Status** column. The status can be **Not Released** or **Released**. A folder can change from **Not Released** to **Released**, but such a change is irreversible. If a folder is **Released**, the folder cannot be changed or edited (changing the folder name, changing the file name in the folder, uploading the folder, changing the version number, or creating a subfolder). You can only download or delete it.
10	New repo	Click **New Repo** to return to the new version of release repos.

----End

5.2 Managing Packages

In the release repos (old), you can upload, view, and edit packages.

Uploading a Package

Step 1 Access release repos using your Huawei Cloud account.

Step 2 Click **Upload**.

Step 3 In the displayed dialog box, enter the required information and click **Upload**.

- **Target Repository**: current release repos.
- **Version**: set the version number for the package.
- **Upload Mode**: select **Single file** or **Multiple files**. **Single file** is selected by default here.
- **Path**: After you set the path name, a folder with that name is created in the **Repo View**, where the uploaded package will be stored.
- **File**: select the package from your local PC to upload.

Step 4 The uploaded package is saved in the file list of release repos.

Click in the **Operation** column of a package to change its name.

Click in the **Operation** column of a package to download it to the local PC.

Click in the **Operation** column of a package to delete it.

----End

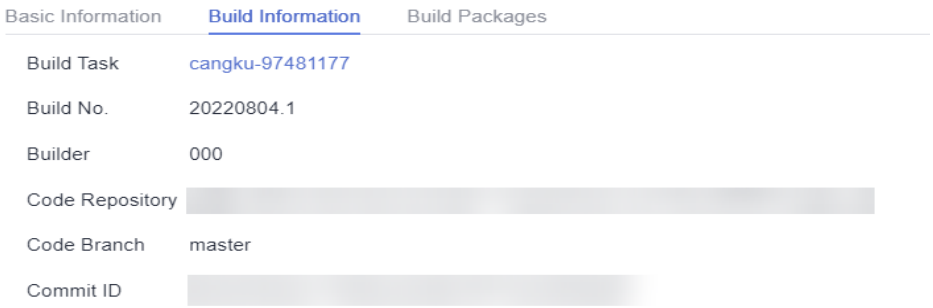
Viewing and Editing a Package

On the **Release Repos** page, you can view or edit the package details, including basic information, build information, and build packages.

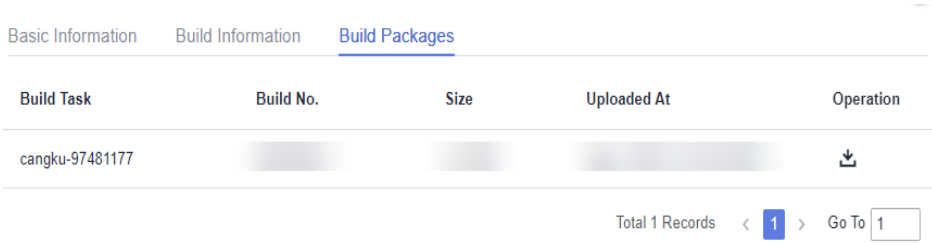
Access the release repo homepage and click the name of a package. A drawer is displayed, showing the package details. The release repos details are displayed on the **Basic Information**, **Build Information**, and **Build Packages** tab pages.

- The **Basic Information** tab page displays the package name, version, size, path, download address, creator, creation time, checksum, and other information.

  You can click  in the upper right corner to change the name and release version of the package. (The release version archived by CodeArts Build is the build sequence number by default.)
- The **Build Information** tab page displays the build task, build No., builder, code repository, code branch, and commit ID of the generated package. Click **Build Task** to link to the task in CodeArts Build.



- The **Build Packages** tab page displays the archiving records of the package uploaded through build tasks. You can click  to download a package.



5.3 Setting Clearing Policies

Release repos automatically clear files on a scheduled basis. You can set a clearing policy to move expired files from the repository to the recycle bin or delete them permanently from the recycle bin based on the specified retention periods.

Procedure

Step 1 Access release repos using your Huawei Cloud account.

Step 2 In the upper right corner of the **Release Repos** page, click the **Setting** tab.

Step 3 Enable **Move expired files to the Recycle Bin** or **Clear from Recycle Bin** as required, and select a retention period from the drop-down list.

Default retention periods:

- **Move expired files to Recycle Bin**: 15 days
- **Clear from Recycle Bin**: 15 days

  You can also set a period. Click **Customize**, enter a number, and click √ to save the setting.

  Parameters below are optional.

  - **Skip released files**: The system retains files in the production package state when clearing files.
  - **Skip specified paths**: The system retains packages that match the file paths you set when clearing files. You can set multiple file paths, each starting with a slash (/) and separated by semicolons (;).

----End

5.4 Managing Recycle Bin

Packages or folders deleted from release repos are moved to the recycle bin, where you can manage them.

Procedure

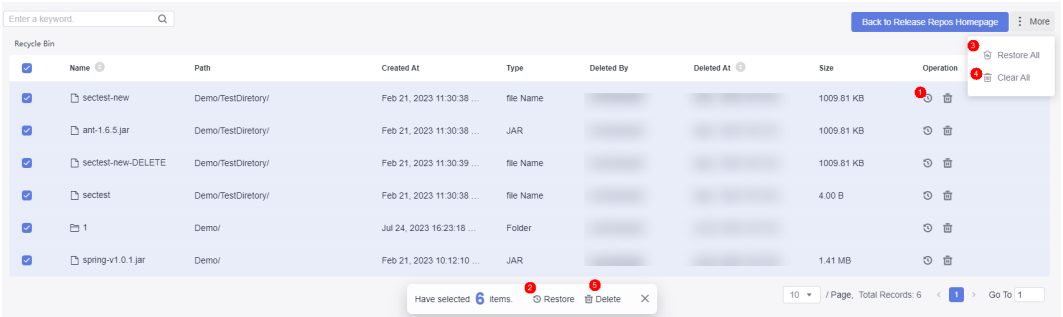
Step 1 Access release repos using your Huawei Cloud account.

Step 2 Click **More** and select **Recycle Bin**.

Step 3 Delete or restore packages or folders as required.

WARNING

Once you delete a package or folder from the recycle bin, it cannot be recovered. Exercise caution when performing this operation.



No.	Operation	Description
1	Individual restore	Click in the **Operation** column to restore the package or folder.
2	Batch restore	Select multiple packages or folders and click **Restore** below the list to restore all the selected items.

No.	Operation	Description
3	Restore all	Choose **More** > **Restore All** to restore all packages or folders in the recycle bin.
4	Clear all	Choose **More** > **Clear All** to delete all packages or folders from the recycle bin.
5	Batch delete	Select multiple packages or folders and click **Delete** below the list to delete all the selected items.
6	Clear	Click  in the **Operation** column to delete the package or folder.

----End

6 IP Address Whitelist

The IP address whitelist includes IP address segments and several access control settings. The whitelist restricts users' access, upload, and download permissions to enhance repository security. You can set the access permission for self-hosted repos by configuring the IP address whitelist.

Creating a Tenant-Level IP Address Whitelist

Step 1 Go to the CodeArts homepage.

1. Log in to the **CodeArts console**, click , and select a region where you have enabled CodeArts.
2. Click **Go to Workspace**.

   If your account uses the old billing mode (see **Old Billing Modes**), click **Access Service**.

Log in to the **CodeArts console**, and click **Access Service**.

Step 2 Click the username in the upper right corner, and select **All Account Settings** from the drop-down list.

Step 3 In the navigation pane, choose **Artifact** > **IP Address Whitelists**.

Step 4 Click **Add Address** in the upper right corner of the page.

Step 5 In the displayed dialog box, select **IP address** or **CIDR** and enter an IP address.

Table 6-1 IP address formats

Format	Description
IP address	This is the simplest IP address format. You can add the IP address of your PC to the whitelist, for example, 100.*.*.123.
CIDR	• If your server is on a LAN and uses the CIDR, you can add a 32-bit egress IP address of the LAN with a specified number of bits for the network prefix. • Requests from the same IP address are accepted if the network prefix is the same as the specified one.

Step 6 Select **I have read and agree to the Privacy Statement and CodeArts Service Statement.**, and click **OK**.

----End

7 Checking Audit Logs

Cloud Trace Service (CTS) records operations on CodeArts Artifact for query, audit, and backtrack.

After you enable CTS, the system starts recording operations on CodeArts Artifact. You can view the operation records of the last seven days on the management console.

CodeArts Artifact Operations Recorded by CTS

Table 7-1 CodeArts Artifact operations recorded by CTS

Operation	Resource Type	Event Name
downloadFile	userName	downloadFile
createFolder	fileName	createFolder
renameFile	fileName	renameFile
modifyFiles	fileName	modifyFiles
modifyFiles	fileName	modifyFiles
restoreTrash	userName	restoreTrash
emptyTrash	userName	emptyTrash
uploadFile	fileName	uploadFile
uploadFile	fileName	uploadFile
uploadFile	fileName	uploadFile
modifyFiles	fileName	modifyFiles
modifyFiles	fileName	modifyFiles
changeVersionCategory	userName	changeVersionCategory
updateAutoDeleteJob-Settings	projectId	updateAutoDeleteJob-Settings

Operation	Resource Type	Event Name
updateProjectRolePermissions	projectId	updateProjectRolePermissions
changeReleaseCategory	userName	changeReleaseCategory
modifyFiles	fileName	modifyFiles
deletePackageVersion	packageVersion	deletePackageVersion
updatePackageVersionStatus	packageVersion	updatePackageVersionStatus
modifyTabInfo	repository	modifyTabInfo
attentionArtifacts	repository	attentionArtifacts
initArtifact	repository	initArtifact
deleteArtifactRepository	repository	deleteArtifactRepository
modifyArtifact	repository	modifyArtifact
deleteArtifactFile	artifact	deleteArtifactFile
clearanceArtifactFile	artifact	clearanceArtifactFile
recoveryArtifactFile	artifact	recoveryArtifactFile
insertProjectRelatedRepository	project	insertProjectRelatedRepository
initMavenRepositories	repository	initMavenRepositories
modifyTabInfo	repository	modifyTabInfo
deleteArtifactFile	artifact	deleteArtifactFile
recoveryArtifactFile	artifact	recoveryArtifactFile
clearanceArtifactFile	artifact	clearanceArtifactFile
downloadMavenSettings	setting	downloadMavenSettings
downloadGradleSettings	setting	downloadGradleSettings
initMavenRepositories	repository	initMavenRepositories
modifyTabInfo	repository	modifyTabInfo
batchDelete	artifact	batchDelete
batchRestore	artifact	batchRestore
resetUserPassword	user	resetUserPassword
createNetProxy	proxy	createNetProxy
updateNetProxy	proxy	updateNetProxy

Operation	Resource Type	Event Name
initVirtualRepositories	repository	initVirtualRepositories
addProxyRepository	repository	addProxyRepository
deleteProxyRepository	repository	deleteProxyRepository
createRemoteRepository	repository	createRemoteRepository
deleteRemoteRepository	repository	deleteRemoteRepository
updateProxyRepoInfo	repository	updateProxyRepoInfo
createRemoteRepository	repository	createRemoteRepository
updateProxyRepoInfo	repository	updateProxyRepoInfo
downloadArtifactFile	file	downloadArtifactFile
uploadArtifact	file	uploadArtifact

Helpful Links

For details about how to query CodeArts Artifact operations on the CTS console, see **Querying Real-Time Traces**.

