CODING DevOps

Artifact Repository

Copyright Notice

Trademark Notice

Service Notice

This document provides an overview of the as-is details of Tencent Cloud's products and services in their entirety or part. The descriptions of certain products and services may be subject to adjustments from time to time.

The commercial contract concluded by you and Tencent Cloud will provide the specific types of Tencent Cloud products and services you purchase and the service standards. Unless otherwise agreed upon by both parties, Tencent Cloud does not make any explicit or implied commitments or warranties regarding the content of this document.

Contact Us

We are committed to providing personalized pre-sales consultation and technical after-sale support. Don't hesitate to contact us at 4009100100 or 95716 for any inquiries or concerns.

Contents

Artifact Repository Quick Start Basic Operations

Last updated: 2024-09-05 16:34:28

Before officially operating the Artifact repository, you can refer to the following content for feature initialization. The steps and preparations mentioned below are not mandatory and can be selectively read based on actual needs.

Create a project

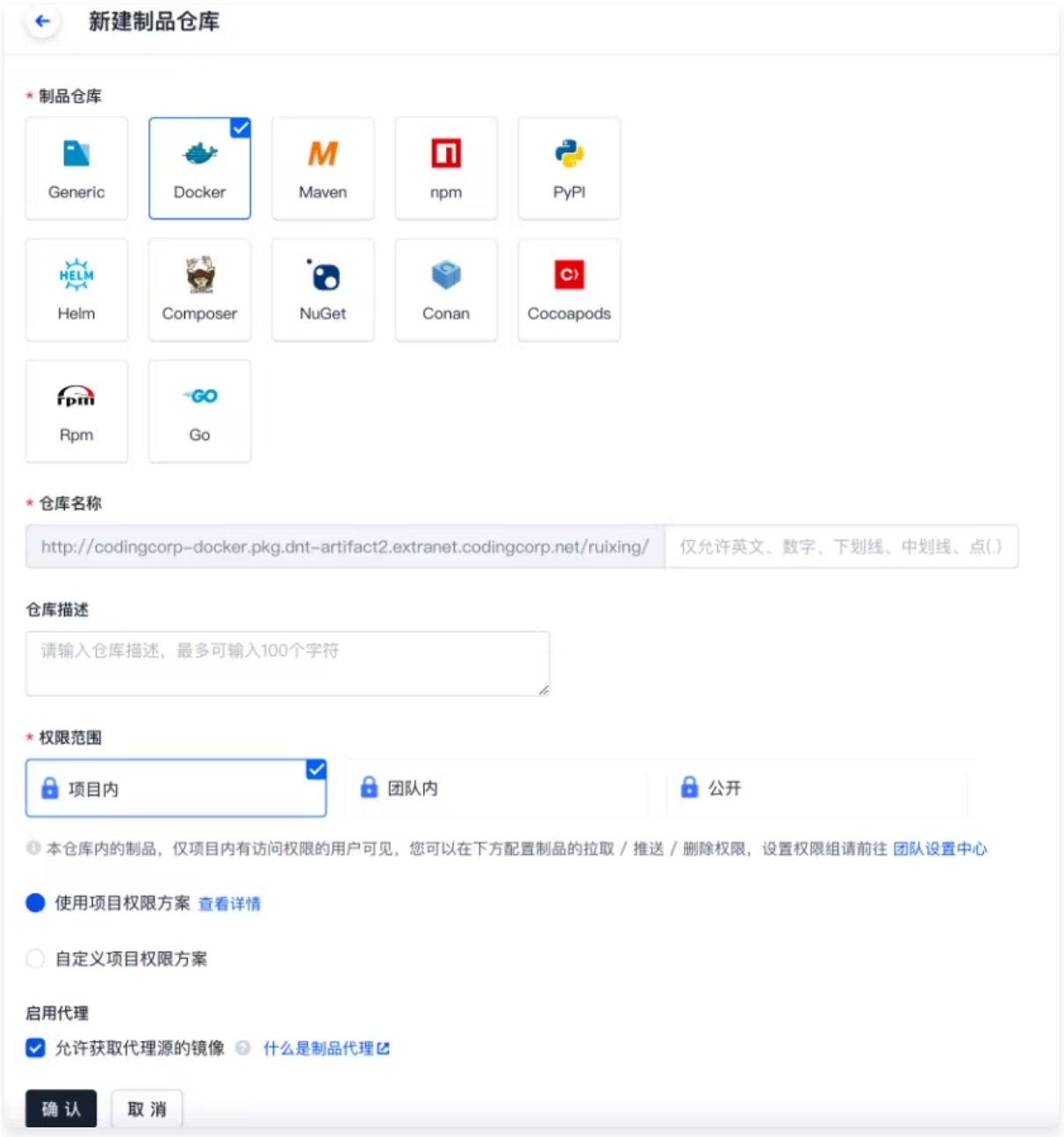
Before using the Artifact repository, create a DevOps project. For detailed steps, refer to Create a Project.
After creating the project, you can enter it and access the Artifact repository feature from the left menu. If the feature is hidden, click the **Project Settings > Project and Members > Menu Management** in the bottom left corner to enable the Artifact repository feature.

Create an Artifact Repository

After entering the Artifact repository, click **Create Artifact Repository** in the top right corner and select the repository type.

- Provide a repository name.
- Select the type of Artifact repository.
- Permission Scope: Configure the permissions of different roles for the current repository. By default, all project members have **Push** and **Pull** permissions.

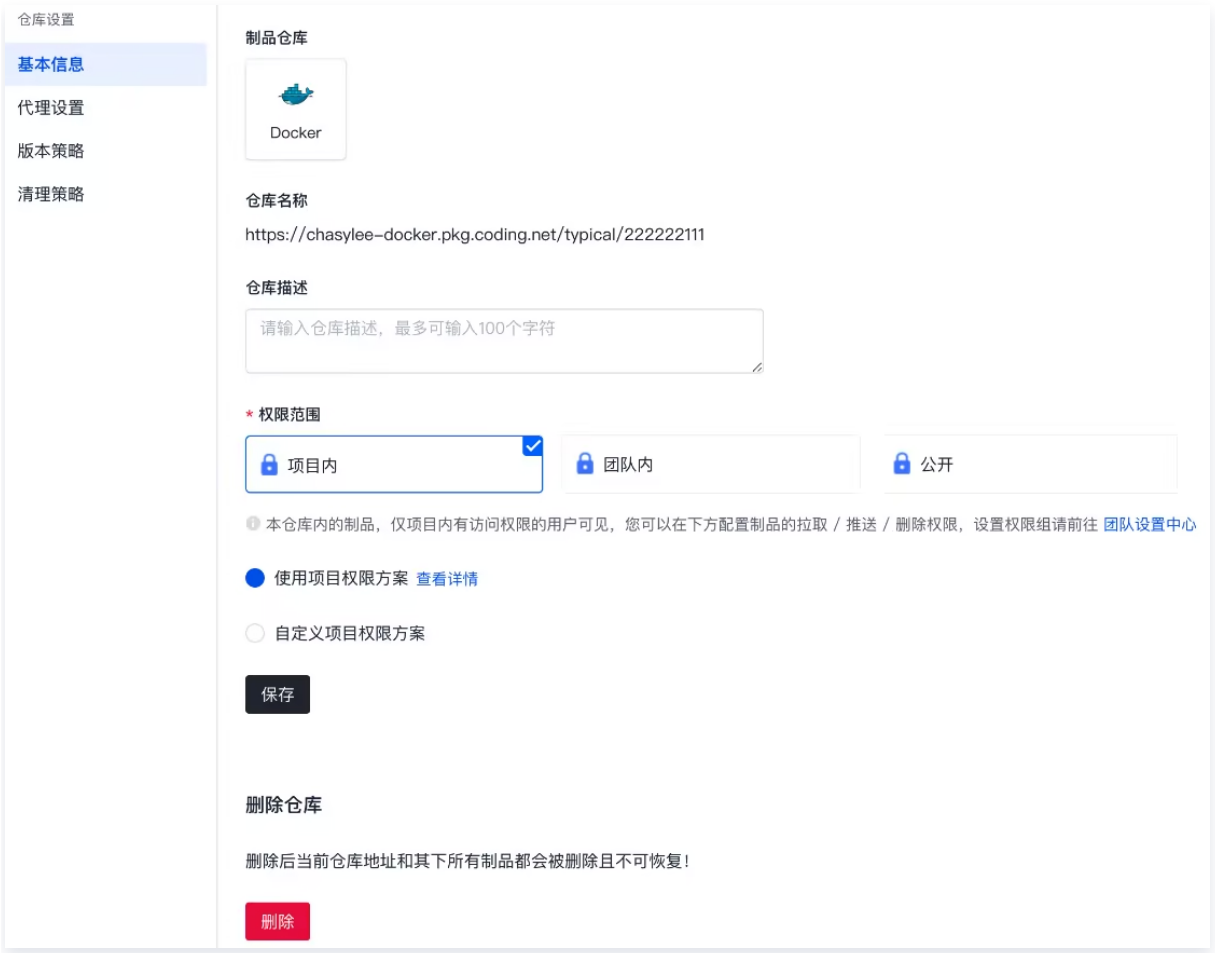
> **Note:**
> After real-name authentication of the team, you can select **Disclosure** as the permission scope of artifacts. The team leader or administrator should perform authentication in the Team Settings Center.



After creating an Artifact repository, you can refer to the quick start guides for each type of artifact from the documentation list on the left.

Delete an Artifact Repository

Click **Repository Settings** in any repository and perform the deletion in the basic information section.



Filter Artifacts

Enter keywords in the artifact search box or adjust search conditions to quickly find the required artifacts. After searching, you can save the search conditions as filters for easier future queries.

Generic

Last updated: 2026-04-03 17:21:44

This document describes how to store generic artifacts in CODING-AR, including how to create a repository and push, pull, and delete artifacts.

Create an Artifact Repository

See Basic Operations to create a Generic type artifact repository in the project.

Push an Artifact

> **Note:**
> The download restriction capacity for the Generic type artifact of the Standard Edition team is 10G per day, and 50G per day for the Advanced Edition team.

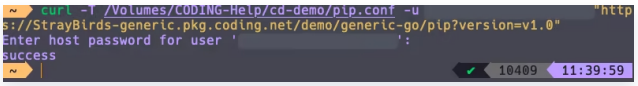
There are two ways to push artifacts:

- Command line
- Upload directly via the page

Upload via command line

Enter the local path, artifact name, and artifact version, and the push command will be automatically generated. Just copy and run it directly in the terminal. The upload process supports breakpoint resume. Follow the prompts in the illustration to install the plugin and then push.

Enter the service password, authenticate and complete the push.



Upload directly via the page

Simply drag and drop the artifact file onto the page button to complete the upload operation.

View Artifact

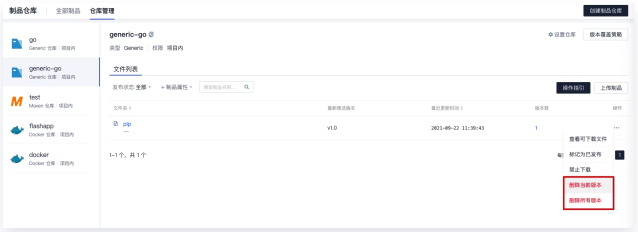
After a successful upload, you can view the version number information in the package list.

Pull an Artifact

Click **Operation Guide** for the uploaded artifact, enter the version number, name, and file name to automatically generate the command, and run the command in the terminal to pull it:

Delete Artifact

Click the **Delete Version** button on the page to delete the artifact version.



If you need to delete the artifact repository, click Basic Operations for more information.

Docker

Last updated: 2026-04-03 17:19:13

This document describes how to store Docker images in CODING-AR for centralized artifact management and version control. The following sections cover image creation, authentication configuration, and artifact push/pull.

> **Note:**
> To read this document, please prepare the following:
> - Installing Docker
> - A created project and artifact repository. For detailed operations, refer to the guidelines in Basic Operations on creating a project and a new artifact repository.
> - Select Docker as the repository type.

Create an Image (Optional)

This section describes how to quickly create a demo Docker image. You can skip this section if you are familiar with Docker images.

Method 1: Create an image locally

1. In a local directory, create a Dockerfile with the following content:

```
FROM coding-public-docker.pkg.coding.net/public/docker/nodejs:12
```

2. Run the following command in the directory to build an image.

```
docker build -t hello-world .
```

The image is created successfully with the default tag `hello-world:latest`. Refer to <Docker Official Documentation> for custom tag formats in the structure of `<ImageName>:<Version>`. This section will not expand further.

```
/Volumes/CODING/Docker-learning docker build -t hello-world .
Sending build context to Docker daemon  2.048kB
Step 1/1 : FROM fanvinga/docker-ssrmu
 ---> 90ec15c0d38d
Successfully built 90ec15c0d38d
Successfully tagged hello-world:latest
```

Method 2: Pull an image from Docker Hub

1. Run the following command in the terminal to pull an image.

```
docker pull hello-world
```

2. Run the following command to view the images pulled.

```
docker images
```

Configure Authentication Information

After creating a local artifact, you can push it to the remote repository. Before the push, you need to locally configure the authentication information of the remote repository.

Access Token

We recommend using an **Access Token** to generate the authentication configuration.

1. Click **Operation Guide** on the repository page.

2. Enter your account's login password or two-step verification code to confirm, then copy the generated command.

3. Paste and run the command in the local Docker environment to complete the authentication.

Push an Image

> **Note:**

> Want to automate pushing Docker images? See how to use the Artifact Repository Plugin in continuous integration.

The following commands are for reference only. Use the commands generated in your project.

1. Tag the the previously pulled hello-world image to local.

```
docker tag hello-world straybirds-docker.pkg.coding.net/coding-demo/coding-demo/hello-world
```

2. Push your Docker images to the artifact repository.

```
docker push straybirds-docker.pkg.coding.net/coding-demo/coding-demo/hello-world
```

Upon successful push, you will see the following content.

```
/Volumes/CODING/Docker-learning> docker push straybirds-docker.pkg.coding.net/coding-demo/coding-demo/hello-world
The push refers to repository [straybirds-docker.pkg.coding.net/coding-demo/coding-demo/hello-world]
cc471758abdf: Pushed
a25d159becc3: Pushed
06cfd7503045: Pushed
beee9f30bc1f: Pushed
latest: digest: sha256:44d5f1eb23a6227332267f15811f8833ecf65f3a52dd57844d30bd4353401f98 size: 1157
```

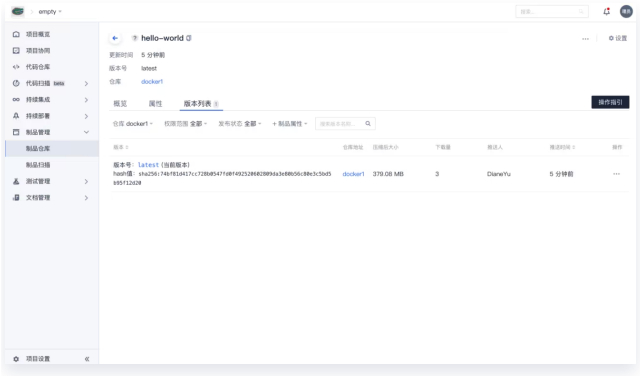
The above operation commands will be displayed directly in the operation guide, where you can input the replacement values and copy the command.

View an Image

After the push is completed, the **Project Overview** in the left menu will broadcast the push activity within the project.

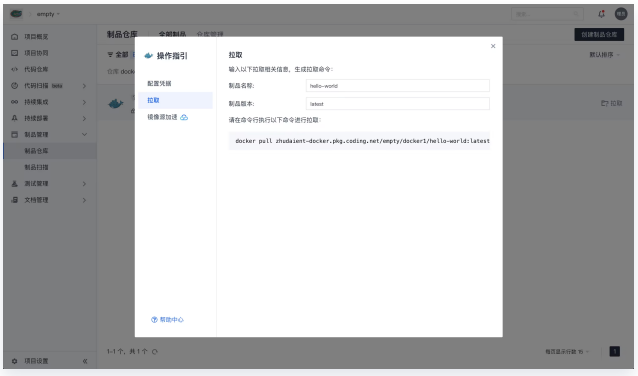
In the project's product list, you can see the pushed **hello-world** image.

Click the image name to view the complete information of the package in the right sidebar, including overview, guide, attributes, version list, and more.



Pull an Image

Use the `docker pull` command to pull a Docker image hosted in CODING-AR. The guide page will automatically generate the corresponding pull command.



Maven

Last updated: 2026-04-03 17:18:21

This document describes how to store Maven artifacts in CODING-AR, including how to create a repository and push and pull artifacts.

> **Note:**
> To read this document, prepare the following:
> - <Basic Operations>—Create a project.
> - Select Maven as the repository type.

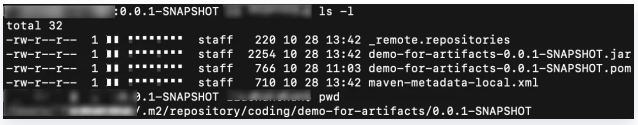
The Maven repository type supports three file formats: Apache Maven, Gradle Groovy, and Gradle Kotlin DSL.

Configure Authentication Information

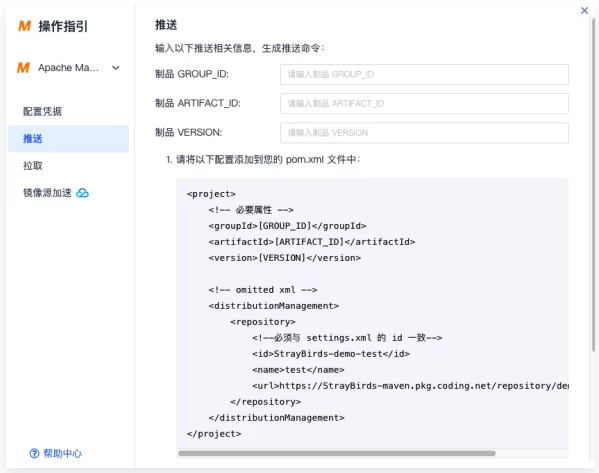
Before pushing or pulling Maven artifacts, you need to configure authentication information. Click in the operation guide **Generate a personal token as credentials** and add it to the `settings.xml` file.

Compile and Upload a Maven Artifact

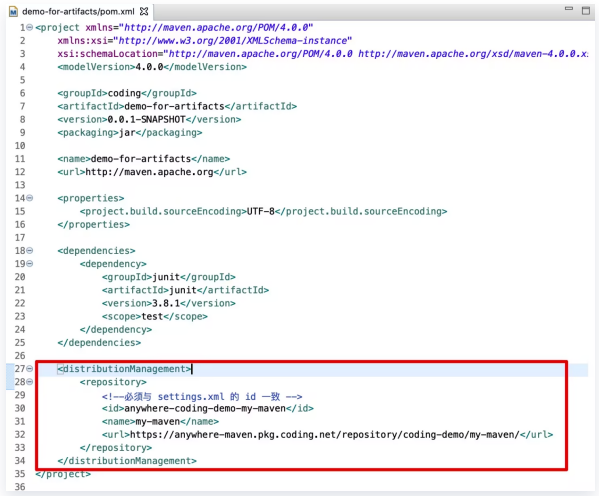
This section describes how to push a demo Maven package to the repository created above.



1. On the repository guide page, copy the following configuration into the project's pom.xml file.



Generally, groupId, artifactId, and version configurations already exist for a Maven project. You only need to add distributionManagement to it.



2. Run the mvn deploy command.

```
mvn deploy
```

If the settings.xml file is not found, add the -s parameter at the end of the command to specify the path where you place the settings file. With the parameter added:

```
 mvn deploy -s "/Users/somebody/software/apache-maven-3.6.2/conf/setti
ngs.xml"
```

3. After the mvn command prompts 'build success', refresh the artifact repository page to see the latest artifacts.

Upload a Maven Package Without Source Code

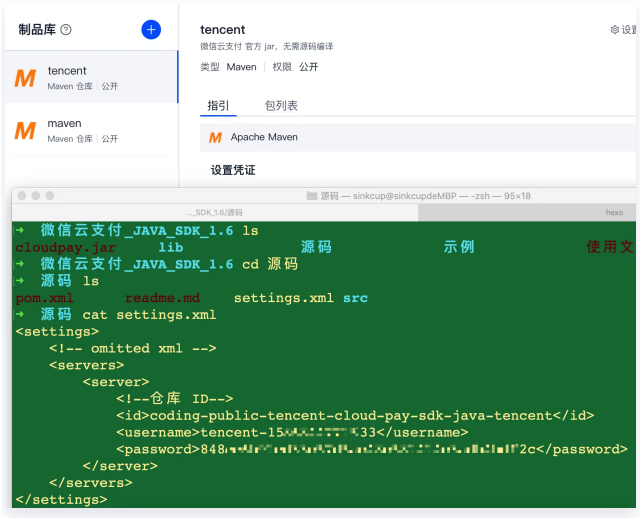
If a third-party Maven package is not officially released to a repository and only a JAR package is provided without source code, you can upload the JAR package to the repository by running the following command:

```
mvn deploy:deploy-file -Durl=file://C:\m2-repo \
                       -DrepositoryId=some.id \
                       -Dfile=your-artifact-1.0.jar \
                       [-DpomFile=your-pom.xml] \
                       [-DgroupId=org.some.group] \
                       [-DartifactId=your-artifact] \
                       [-Dversion=1.0] \
                       [-Dpackaging=jar] \
                       [-Dclassifier=test] \
                       [-DgeneratePom=true] \
```

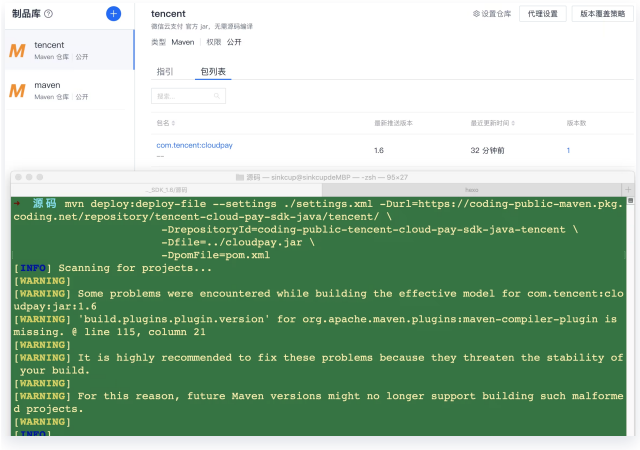
```
                        [-DgeneratePom.description="My Project
Description"] \
                        [-DrepositoryLayout=legacy]
```

If a third party provides `pom.xml`, you can extract the group, artifact, and version fields from it. For example, use the following command for the WeChat CPay Java SDK:

```
mvn deploy:deploy-file --settings ./settings.xml -Durl=https://coding-
public-maven.pkg.coding.net/repository/tencent-cloud-pay-sdk-
java/tencent/ \
                        -DrepositoryId=coding-public-tencent-cloud-pay-
sdk-java-tencent \
                        -Dfile=../cloudpay.jar \
                        -DpomFile=pom.xml
```



WeChat CPay Java SDK jar upload page.



Pull Maven Artifact

1. On the guide page, copy the configuration to settings.xml. For example, the configuration for the WeChat CPay Java SDK is as follows:

```
<settings>
    <!--   omitted xml -->
    <profiles>
        <profile>
            <id>Repository Proxy</id>
            <activation>
                <activeByDefault>true</activeByDefault>
            </activation>
            <repositories>
                <repository>
                    <id>coding-public-tencent-cloud-pay-sdk-java-
tencent</id>
                    <name>tencent</name>
                    <url>https://coding-public-
maven.pkg.coding.net/repository/tencent-cloud-pay-sdk-
java/tencent/</url>
                    <releases>
                        <enabled>true</enabled>
                    </releases>
                    <snapshots>
                        <enabled>true</enabled>
                    </snapshots>
                </repository>
            </repositories>
        </profile>
    </profiles>
</settings>
```

2. Configure the dependencies in the pom.xml file of your Java project. For example, for the WeChat CPay Java SDK, the configuration is as follows:

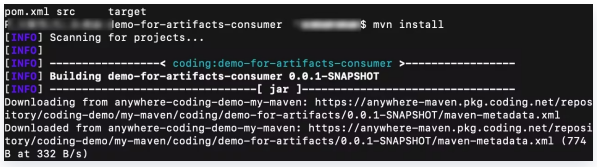
```
<project>
    <dependencies>
        <dependency>
            <groupId>com.tencent</groupId>
            <artifactId>cloudpay</artifactId>
```

```
            <version>1.6</version>
        </dependency>
    </dependencies>
</project>
```

3. Compile the project.

```
mvn install -s ./settings.xml
```

You can view the package is being pulled during the execution. Alternatively, view the pulled package in the local maven cache after the execution is completed.



Npm

Last updated: 2026-04-03 17:16:15

This document describes how to store npm artifacts in CODING-AR for centralized artifact management and version control. The following sections introduce how to create an artifact, configure authentication, and pull and push artifacts.

> **Note:**
> To read this document, prepare the following:
> - Install Node.js.
> - <Basic Operations>—Create a project.
> - Select npm as the repository type.

Initialize a Local npm Project (Optional)

You can skip this section if you are familiar with npm artifacts.

1. Create a demo directory for the npm project.

```
mkdir npm-demo
```

2. Initialize the npm project.

```
cd npm-demo && npm init
```

Follow the prompts to fill in the configuration file of the npm package in the newly added `package.json`.
For example:

```
{
    "name": "example",
    "version": "1.0.0",
    "description": "",
    "main": "index.js",
    "author": "",
    "license": "MIT"
}
```

3. Create a `.npmrc` file.

```
touch .npmrc
```

Configure Authentication Information

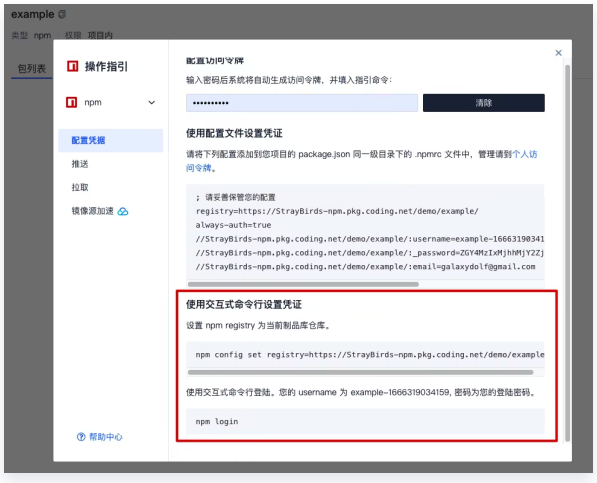
Configure authentication before pushing or pulling an artifact.
Configure authentication information in either of the following ways:

- Set credentials with the configuration file
- Set credentials with the interactive command line

Method 1: Set credentials using a configuration file

1. On the artifact repository guide page, enter the password and click Generate Personal Token as a credential.

2. Copy the interactive command line instructions newly added to the page, and add them to the `.npmrc` file in the same directory as your project's `package.json`.



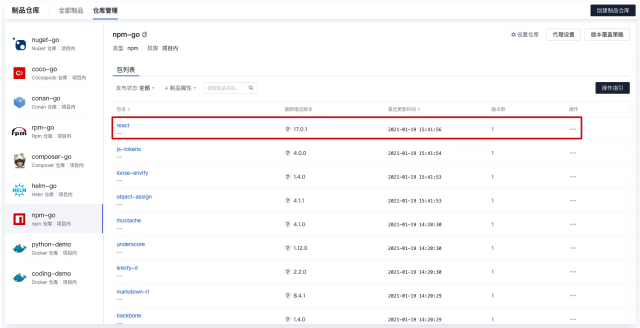
Method 2: Set credentials using interactive command line

Follow the page instructions to configure credentials in the local artifact repository.

Push an Artifact

Enter the artifact name and version in the web guide, initialize first, and then push the artifact as prompted.

After the artifact is pushed successfully, refresh the page to view the latest artifacts.



Pull an Artifact

Enter the artifact name and version in the guide to auto-generate the pull command.

After the artifact is pulled, you can see a success message.

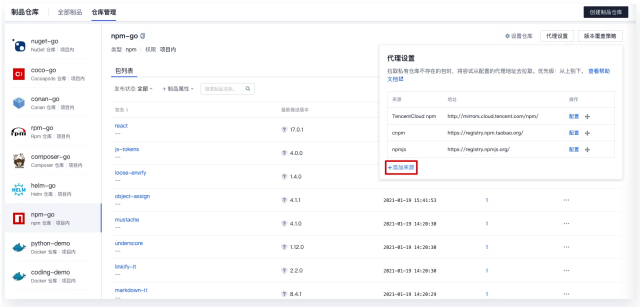
```
/Volumes/CODING/node npm install hello-world --registry=https://straybirds-npm.pkg
.coding.net/coding-demo/npm-go/
npm WARN node@1.0.0 No description
npm WARN node@1.0.0 No repository field.

+ hello-world@0.0.2
updated 1 package in 35.936s

3 packages are looking for funding
  run `npm fund` for details
```

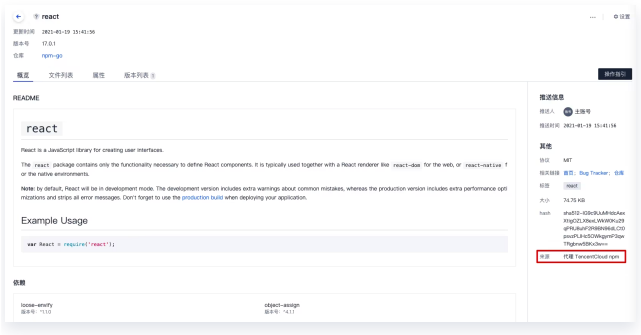
Configure a Proxy

If the desired artifact is not hosted in the CODING Artifact Repository, it will attempt to pull from the configured proxy address. You can add a third-party artifact source to get artifacts from a specific repository. No additional settings are needed; the repository will sequentially retrieve the respective artifact packages.



Use the pull command in the operation guidelines, replace the package name in `<package>`, to complete the pull.

The artifact and its dependencies will be pulled to the local machine and synchronized to CODING-AR. The package source will be shown on the details page.



For more information, see Artifact Repository Proxy.

Helm

Last updated: 2026-04-03 17:21:09

This document describes how to store Helm artifacts in CODING-AR, including how to create a repository and push, pull, and delete artifacts.

> **Note:**
> To read this document, prepare the following:
> - Install Helm.
> - Basic Operations—Create a project.
> - Select Helm as the repository type.

Create a Package (Optional)

This section describes how to quickly create a Helm chart. You can skip this section if you are familiar with Helm charts.

Method 1: Create an image locally

1. In a local directory, create a Helm chart.

```
helm create [name]
```

2. Package the chart.

```
helm package [name]
```

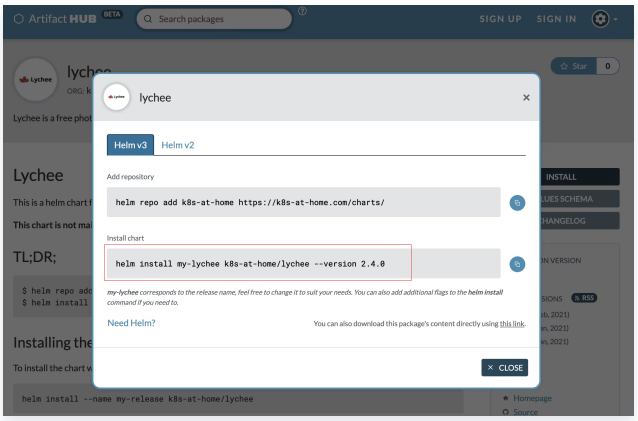
Method 2: pull an artifact from Artifact Hub

Search for a Helm chart and run the download command in a local directory.

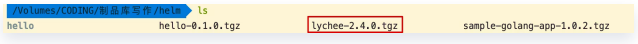
```
$ helm repo add [Remote Repository Name] [Remote Repository URL]

$ helm fetch [Helm Chart URL in the Remote Repository>]--version
[Version]
```

Replace install with fetch, and delete my-lychee before copying this link.



After the commands are successfully run, you can view the artifact locally.



Configure Authentication Information

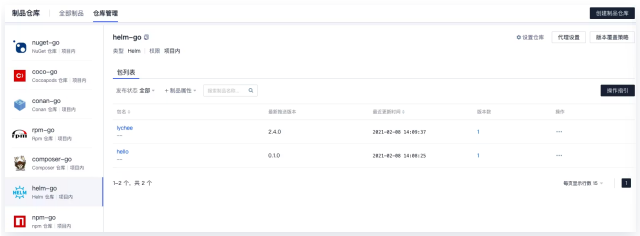
After creating a local artifact, you can push it to the remote repository. You can pull or push an artifact using **Helm + cURL** or **Helm + CODING Helm** plugin.

According to the guide, enter password for information authentication.

Push an Artifact

Input the artifact name in the operation guide page to auto-generate the push command. After copying it, use the terminal to navigate to the Helm Chart directory and execute it.

After the artifact is pushed successfully, refresh the page to view the latest artifacts.



Pull an Artifact

If your artifact repository is updated, generate a pull command in the operation guide to update.

PyPI

Last updated: 2026-04-03 17:17:44

This document describes how to store PyPI artifacts in CODING-AR for centralized artifact management and version control. The following sections introduce how to create an artifact, configure authentication, and pull and push artifacts.

Note:
To read this document, prepare the following:
- Install Python3.
- Basic Operations—Create a project.
- Select PyPI as the repository type.

Initialization

1. Create a demo directory as the address for the local PyPI package. Run the command in the terminal to create the demo project folder.

```
mkdir -p demo/example_pkg/__init__.py
```

2. Go to the demo directory and create a setup.py file.

```
cd demo && touch setup.py
```

3. Paste the configuration into the setup.py file.

```
import setuptools

setuptools.setup(
    name="example-pkg-YOUR-USERNAME-HERE", # Replace with your own
username
    version="0.0.1",
    author="Example Author",
    author_email="author@example.com",
    description="A small example package",
    url="https://github.com/pypa/sampleproject",
    packages=setuptools.find_packages(),
```

```
    classifiers=[
        "Programming Language :: Python :: 3",
        "License :: OSI Approved :: MIT License",
        "Operating System :: OS Independent",
    ],
    python_requires='>=3.6',
)
```

4. Install setuptools and wheel tools.

```
python3 -m pip install --user --upgrade setuptools wheel
```

5. Package the project.

```
python3 setup.py sdist bdist_wheel
```

After packaging the project, the following two files will be generated in the `/dist` directory for pushing to the artifact repository.

```
└─dist
    ├─example_pkg_YOUR_USERNAME_HERE-0.0.1-py3-none-any.whl
    ├─example_pkg_YOUR_USERNAME_HERE-0.0.1.tar.gz
```

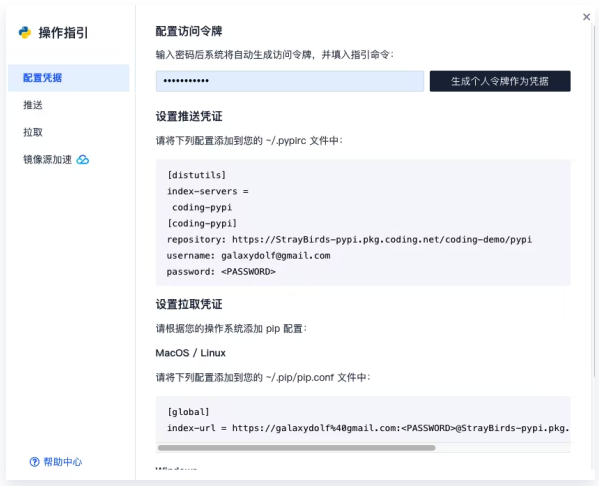
Configure artifact repository authentication information

Before pushing to the CODING artifact repository, you need to add the corresponding authentication information in the local file. You can authenticate through either **automatic configuration** or **manual configuration**. Before proceeding, please use the command `cd /` to go to the root directory and enter `ls -a` to check if the `.pypirc` and `pip.conf` files exist. If not, enter the following commands to create the two files.

```
touch .pypirc && touch pip.conf
```

Automatic configuration

1. Click on the website to **generate a personal token as credentials**, and the system will automatically generate access credentials for you. To view the personal token, go to **Account Settings > Access Tokens** for management.



2. Enter your login password to confirm and obtain the configuration.

   Copy the configuration content into the root directory's `.pypirc` and `pip.conf` files.

Push an Artifact

Enter the project directory, such as the previously created Demo directory, copy the command from the website and execute it in the terminal to push all artifacts in the `Demo/dist` directory to the artifact repository.

```
twine upload -r coding-pypi dist/*
```

```
/Volumes/CODING/pypi/packaging_tutorial twine upload -r coding-pypi dist/*
Uploading distributions to https://codingcorp-pypi.pkg.coding.net/coding-help-generator/pypi-go
Uploading example_pkg_YOUR_USERNAME_HERE-0.0.1-py3-none-any.whl
100%|████████████████████████████████| 4.41k/4.41k [00:00<00:00, 6.96kB/s]
Uploading example-pkg-YOUR-USERNAME-HERE-0.0.1.tar.gz
```

After the artifact is pushed successfully, refresh the page to view the latest artifacts.

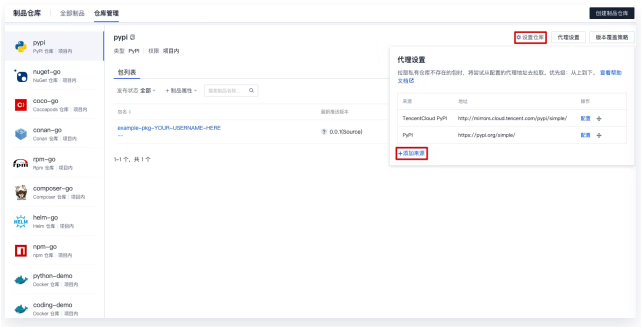
Pull

Based on the specific artifact retrieval guidelines in the PyPI artifact repository, you can execute `pip install` to retrieve artifacts.

```
pip install <artifact name>
```

Configure a Proxy

If you try to pull an artifact that does not exist in the CODING private repository, the system will try to pull from the configured proxy. You can add a third-party artifact source to obtain artifacts from the specific repository. Without the need for configuration, CODING will retrieve artifacts in sequence from top to bottom.



Use the command generated on the webpage, replace `<package>` with the package name to complete the pull. The artifact and its dependencies will be successfully pulled to the local machine and synchronized to the CODING Artifact Repository. The package source will be shown on the details page.

For detailed information about proxy settings, please refer to Artifact Repository Proxy.

Composer

Last updated: 2026-04-03 17:24:20

This document describes how to store Composer artifacts in CODING-AR for centralized artifact management and version control. The following sections introduce how to create an artifact, configure authentication, and pull and push artifacts.

> **Note:**
> To read this document, prepare the following:
> - Install Composer.
> - Basic Operations—Create a project.
> - Select Composer as the repository type.

Create a Composer Package (Optional)

Install Composer

Run the download Composer command in the terminal.

```
curl -sS https://getcomposer.org/installer | php
```

Add an environment variable for global access.

```
mv composer.phar /usr/local/bin/composer
```

Initialization

Create a demo directory.

```
mkdir composer-demo && cd composer-demo
```

Initialize the Composer package. Enter the required information when asked.

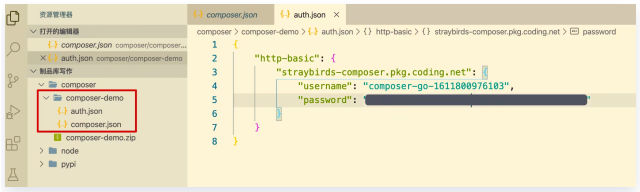
```
composer inits
```

After initialization, a composer.json configuration file will be added to the directory as the configuration file for the Composer package.

Configure the Authentication File

Go to the Composer package directory and create an auth.json file. Enter the password, click the **Generate Personal Token** button on the guide page to automatically generate push credentials.

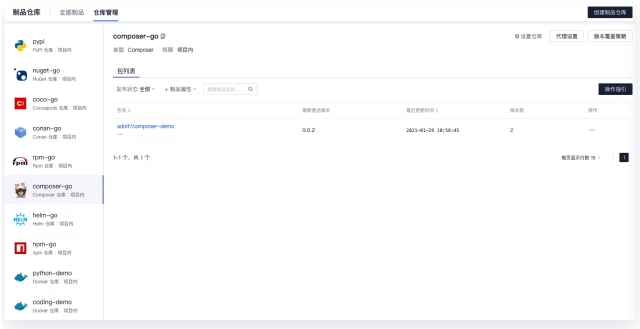
Copy the credentials and paste them into the auth.json file.



Push an Artifact

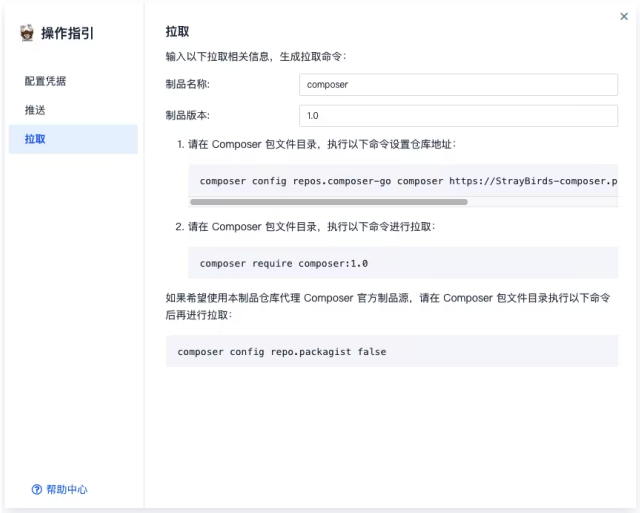
Enter the artifact name and version on the guide page to automatically generate the push command, then run it in the terminal.

After the push is complete, you can see the pushed package in CODING-AR.



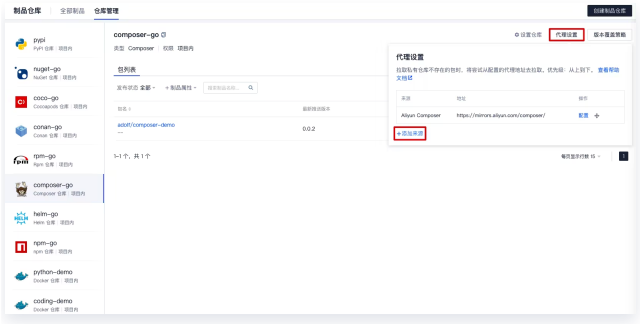
Pull an Artifact

Enter the artifact name and version on the guide page to automatically generate the pull command.



Configure a Proxy

If you try to pull an artifact that does not exist in the CODING private repository, the system will try to pull from the configured proxy. You can add a third-party artifact source to obtain artifacts from the specific repository. Without the need for configuration, CODING will retrieve artifacts in sequence from top to bottom.



For detailed instructions on proxy settings, please refer to Artifact Repository Proxy.

Nuget

Last updated: 2026-04-03 17:23:01

This document describes how to store NuGet artifacts in CODING-AR for centralized artifact management and version control. The following sections introduce how to create an artifact repository and NuGet package, pull and push artifacts, and configure proxies.

> **Note:**
> To read this document, prepare the following:
> - Install NuGet.
> - Basic Operations—Create a project.
> - Select NuGet as the repository type.

Initialize a NuGet Artifact (Optional)

Visit Official website to download and install NuGet.

Generate a NuGet artifact locally

You can skip this section if you are familiar with NuGet artifacts.

1. Create a demo directory.

```
mkdir nuget-demo && cd nuget-demo
```

2. Create a `.nuspec` package.

```
nuget spec [Artifact Name]
```

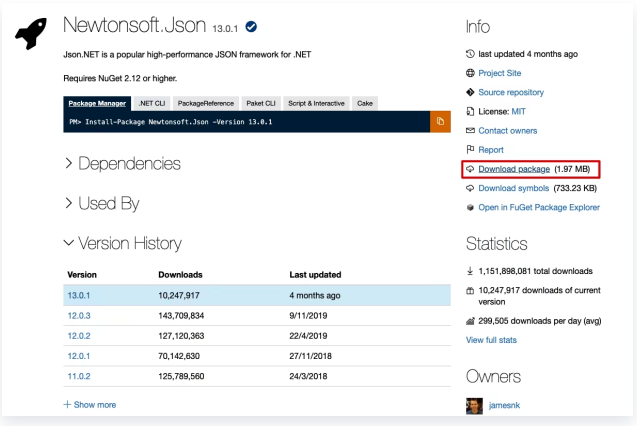
3. Package the artifact.

```
nuget pack <Artifact Name>.nuspec
```

4. After the artifact is packaged, you can view the package file generated in the local directory.

Pull Online

Click visit Official website , search for any NuGet artifact, and download it via the online link or command line.

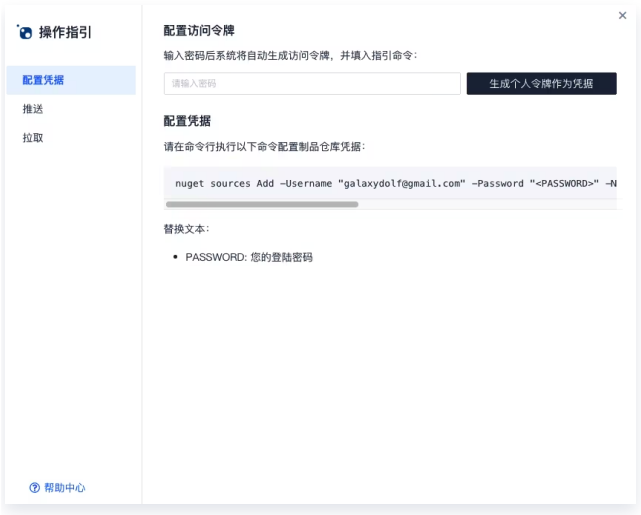


Pull an artifact via the command line:

```
nuget install [Artifact Name] -OutputDirectory packages
```

Configure Authentication Information

You need to locally configure the authentication information for accessing NuGet artifact repositories in CODING. Here, we use**Automatically Generate Configuration** to complete the authentication process.
After entering the password, click **Generate Personal Token as Credentials** on the operation guide. After entering the password, obtain the configuration command. Copy and execute it in the directory of the NuGet artifact you want to push. The **Permission Mechanism** utilized in this process uses the **Personal Access Token** feature. For more details and control, please refer to Access Token .



Push an Artifact

Enter the command line, replacing the corresponding names with local content to complete the push.

```
nuget push -ApiKey api -Source "nuget-go" [Artifact Name].nupkg
```

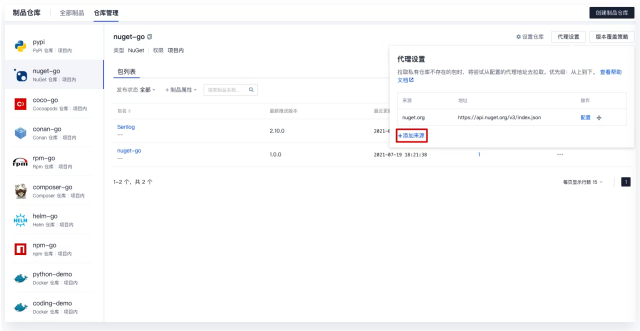
Pull an Artifact

Enter the command line, replacing the corresponding names with local content to complete the pull.

```
nuget install -Source "nuget-go" -Version [Artifact Version] [Artifact N
ame]
```

Configure a Proxy

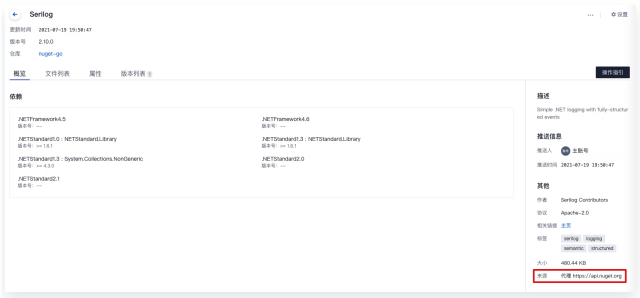
If you try to pull an artifact that does not exist in the CODING private repository, the system will try to pull from the configured proxy. You can add a third-party artifact source to obtain artifacts from the specific repository. Without the need for configuration, CODING will retrieve artifacts in sequence from top to bottom.



Run the following command to pull an artifact:

```
nuget install -Source "nuget-go" -Version [Artifact Version] [Artifact N
ame]
```

The artifact and its dependencies will be pulled to the local machine and synchronized to CODING-AR. The package source will be shown on the details page.



If CODING-AR does not automatically store NuGet artifacts pulled from the proxy, check:

1. Whether you have the permission to push artifacts to this repository.
2. Whether the artifacts already exist in your local cache.

For detailed information about proxy settings, please refer to Artifact Repository Proxy.

Rpm

Last updated: 2026-04-03 17:25:08

This document describes how to store RPM artifacts in CODING-AR for centralized artifact management and version control. The following sections introduce how to create an artifact, configure authentication, and pull and push artifacts.

> **Note:**
> To read this document, prepare the following:
> - Prepare a Linux environment.
> - Basic Operations—Create a project.
> - Select RPM as the repository type.

Initialization

RPM is installed in Linux by default. You can run rpm commands in a Linux terminal directly. To use rpm commands in other operating systems, use Docker to install CentOS:

```
docker run -it --name centos centos:8 /bin/bash
```

Download a Demo Project

Go to the rpm artifact download page, search for the artifact package, download it to the local machine, and then install it.
For example:

```
wget -N --no-check-certificate
"https://www.rpmfind.net/linux/fedora/linux/development/rawhide/Everythi
ng/aarch64/os/Packages/h/hello-2.12.1-4.fc40.aarch64.rpm"
```

Configure Authentication Information

Click on **Operation Guide** and the **Generate Personal Token as Credential** option will automatically generate and set the credential.

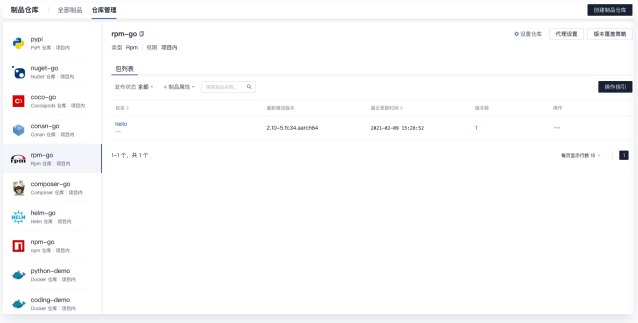
Copy the generated code to the local `/etc/yum.repos.d/rpm-go.repo` file. If the file does not exist, please create it.

Push an Artifact

Run the rpm publish command to push an RPM package.

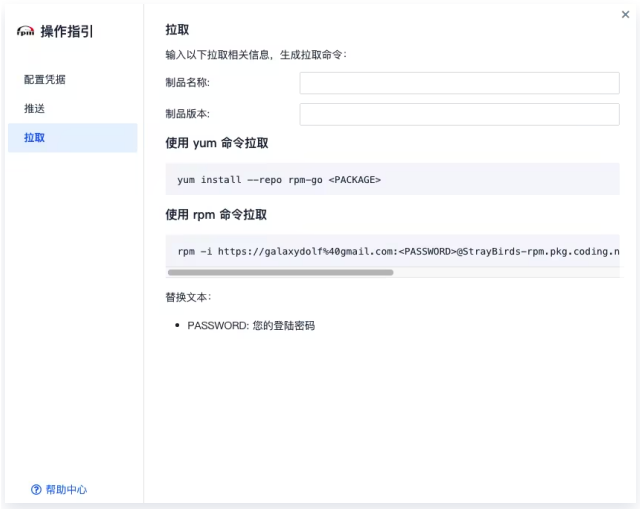
```
curl -u [ username /email] -X POST [repository URL provided in the push guide] -T [artifact name].rpm
```

After the artifact is pushed successfully, refresh the page to view the latest artifacts.



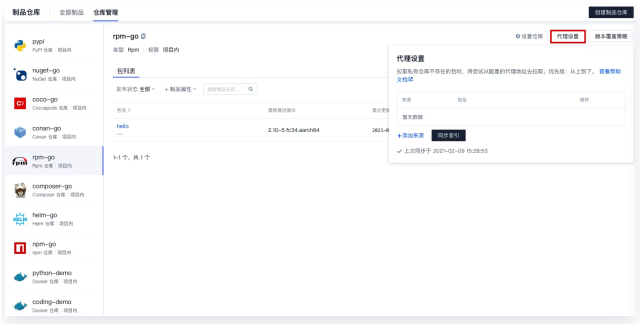
Pull an Artifact

Run the commands provided in the page guide to complete the pull operation.



Configure a Proxy

RPM repositories have a default proxy address. You can configure other addresses.



Configure the remote proxy repository URL, pull artifacts in the repository to the local machine, and the artifacts will be automatically backed up to CODING-AR.
If the RPM Artifact Repository does not have a Storage Proxy RPM Artifact, there could be two reasons:

1. Whether you have the permission to push artifacts to this repository.
2. Whether the artifacts already exist in your local cache.

Conan

Last updated: 2026-04-03 17:25:50

This document describes how to store Conan artifacts in CODING-AR for centralized artifact management and version control. The following sections introduce how to create an artifact, configure authentication, and pull and push artifacts.

Note:
To read this document, prepare the following:
- Install Python3.
- Basic Operations—Create a project.
- Select Conan as the repository type.

Install Conan

Python 3.5 or later is required to install Conan using pip.

```
pip3 install conan
```

Install Conan with brew.

```
brew install conan
```

Create a Conan Package

Create a local demo directory.

```
mkdir mypkg && cd mypkg
```

Create a demo project.

```
conan new hello/0.1 -t
```

Build a binary package for the project.

```
conan create . demo/testing
```

If an error occurs `/bin/sh: cmake: command not found`, you need to run the command to install cmake.

```
$ pip3 install cmake
# or
$ brew install cmake
```

Configure Authentication Information

After entering password in the guide, click **Generate Personal Token as Credentials** to automatically generate and execute the command.

Push an Artifact

Run the command in the guide. Replace the variables with the name and version of the artifact to be pushed.

```
conan upload [package name]/[custom version number] --all -r=conan-go
```

Pull an Artifact

Run the pull command from the guide to pull an artifact from the current repository.

```
conan install <package name>/<custom version number>@ -r conan-go
```

Cocoapods

Last updated: 2026-04-03 17:23:36

This document describes how to store CocoaPods artifacts in CODING-AR for centralized artifact management and version control. The following sections introduce how to create an artifact, configure authentication, and pull and push artifacts.

> **Note:**
> To read this document, prepare the following:
> - Install CocoaPods.
> - Basic Operations—Create a project.
> - Select CocoaPods as the repository type.

Installation

Run either of the following commands to install CocoaPods.

```
$ sudo gem install cocoapods
-- or
$ brew install cocoapods
```

Create a Demo Project

Execute the create command in any directory and select the required example template based on the command line prompts:

```
pod lib create <Custom Pod Name>
```

The sample CocoaPods code will be cloned from GitHub to your local machine.

Configure Authentication Information

Install the CODING CocoaPods plugin.

```
sudo gem install cocoapods-coding-ar
```

Enter your password in the operation instructions and click **Generate Personal Token as Credential**. The system will automatically generate the configuration credentials.

Copy the command to your `~/.netrc` file. If the file does not exist, run `vim ~/.netrc` to create it and paste the content.

Push an Artifact

Enter the product name in the operation instructions and follow the guided steps to execute the commands.

Pull an Artifact

Pull the specified CocoaPods artifact and version by referring to the specific guide in the CocoaPods Artifact Repository.

Go

Last updated: 2026-04-03 17:22:22

This document describes how to store Go artifacts in CODING-AR for centralized artifact management and version control within a team project. The following sections introduce creating a Go artifact repository, locally packaging Go artifacts, and using the push-pull features.

> **Note:**
> To read this document, prepare the following:
> - Install Go version 1.13 or above.
> - A created project and artifact repository. For detailed operations, refer to the guidelines in Basic Operations on creating a project and a new artifact repository.
> - Select Go as the repository type.

Configure Authentication Information

Configure authentication before pushing or pulling an artifact.

1. In the **Warehouse Management** tab of the Artifacts Repository, click **Operation Guide**.

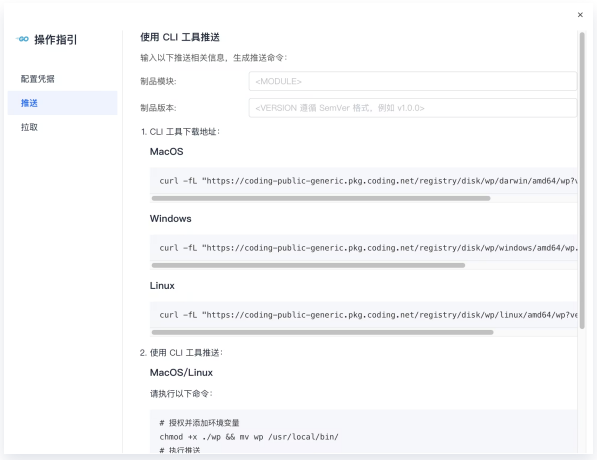
2. On the **Operation Guide**'s **Configure Credentials** page, enter your personal account's log in to password, then click **Generate Personal Token as Credentials**.

3. Add the GOPROXY configuration according to the operating system, and copy the command from the popup to execute it in the terminal.

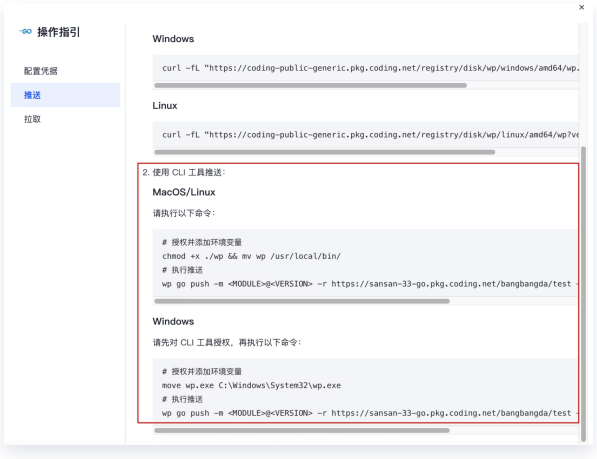
Push Go Artifacts

Push Go artifacts using the CLI tool.

1. On the **Operation Guide**'s **Push** page, copy and execute the corresponding command in the terminal according to different operating systems to complete the CLI tool download and installation.

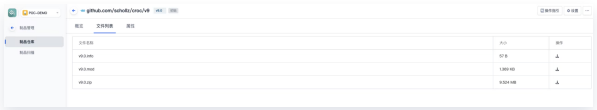


2. Enter the **Artifact Module**, **Artifact Version**, and the system automatically generates the push command. According to different operating systems, copy and execute the corresponding command in the terminal to complete the push of the Go artifact package to the artifact repository.



3. Upon successful push of the artifact package, the terminal interface will display the following information.

The pushed Go artifact can be viewed in the artifact list on the artifact repository page.



Pull Go Artifacts

Pull Go artifacts using one of the following methods.

- On the **Operation Guide**'s **Push** page, enter the Go artifact's module information, copy and execute the command in the terminal to pull the Go artifact.

> **Note:**
> - The entered **Artifact Module** information must match the module information in the Go artifact's `go.mod` file. Otherwise, it will result in a failure to pull the Go artifact repository.
> - When using the `go get` command in a non-https environment, the Go proxy must not include authentication information and must be a public repository.

- Execute the following command in the directory where the terminal's `go.mod` file is located to pull the dependency artifact.

Upon successful pull of the artifact package, the terminal interface will display the following information.

```
→  croc git:(main) ✗ go get github.com/schollz/croc/v9
go: added github.com/cespare/xxhash v1.1.0
```

Configuring permission

Last updated: 2024-09-05 16:40:44

As an important asset of the team, the permission management of products is crucial. For example, some products need to be open for third-party external use (such as open-source component packages), some products need to be open to other project team members within the team (such as basic public component packages), and some products only need to be open to the members of this project (such as application installation packages). CODING-AR provides comprehensive permission management to meet the permission needs in different scenarios.

Permission rules

1. CODING-AR provides three permission scopes.
   - Within the project
   - Within the team
   - Public

   For public products, the team needs to perform real-name operations. The team leader/administrator must go to the team settings center for real-name authentication.

2. Different operations by members on the product.
   - Pull: Pull any specified product from the product library.
   - Push: Push any product to the product library.
   - Agency: Synchronize products not in the cache from the agency.
3. Permission schemes
   - Project permission scheme:

If the project permission scheme is selected, the user's permission in the product library depends on the product library permissions owned by the associated project permission group.

- Custom permission scheme:

  If you select a custom permission scheme, the specified custom permissions will apply.

> **Note:**
> Project Administrators have all artifact operation permissions by default, not limited by the custom permission scheme or project permission scheme.

Setting permissions

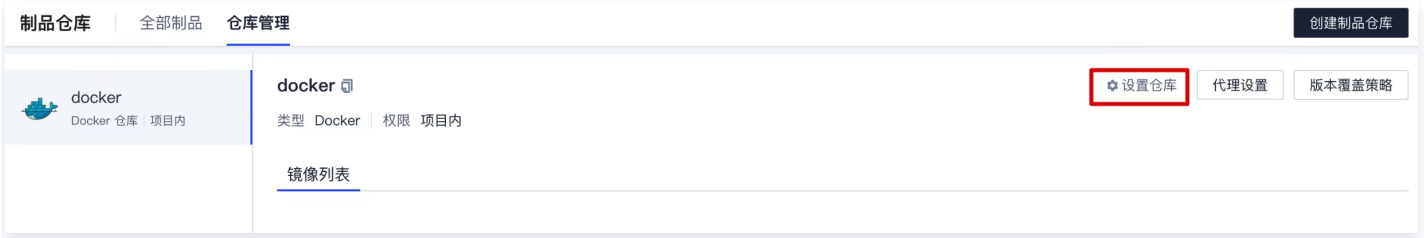
After understanding the authentication rules corresponding to the above permissions, you can set the permission scope as needed when creating an artifact repository. You can also modify the permission scope for existing artifact repositories.

Set permissions when creating a repository

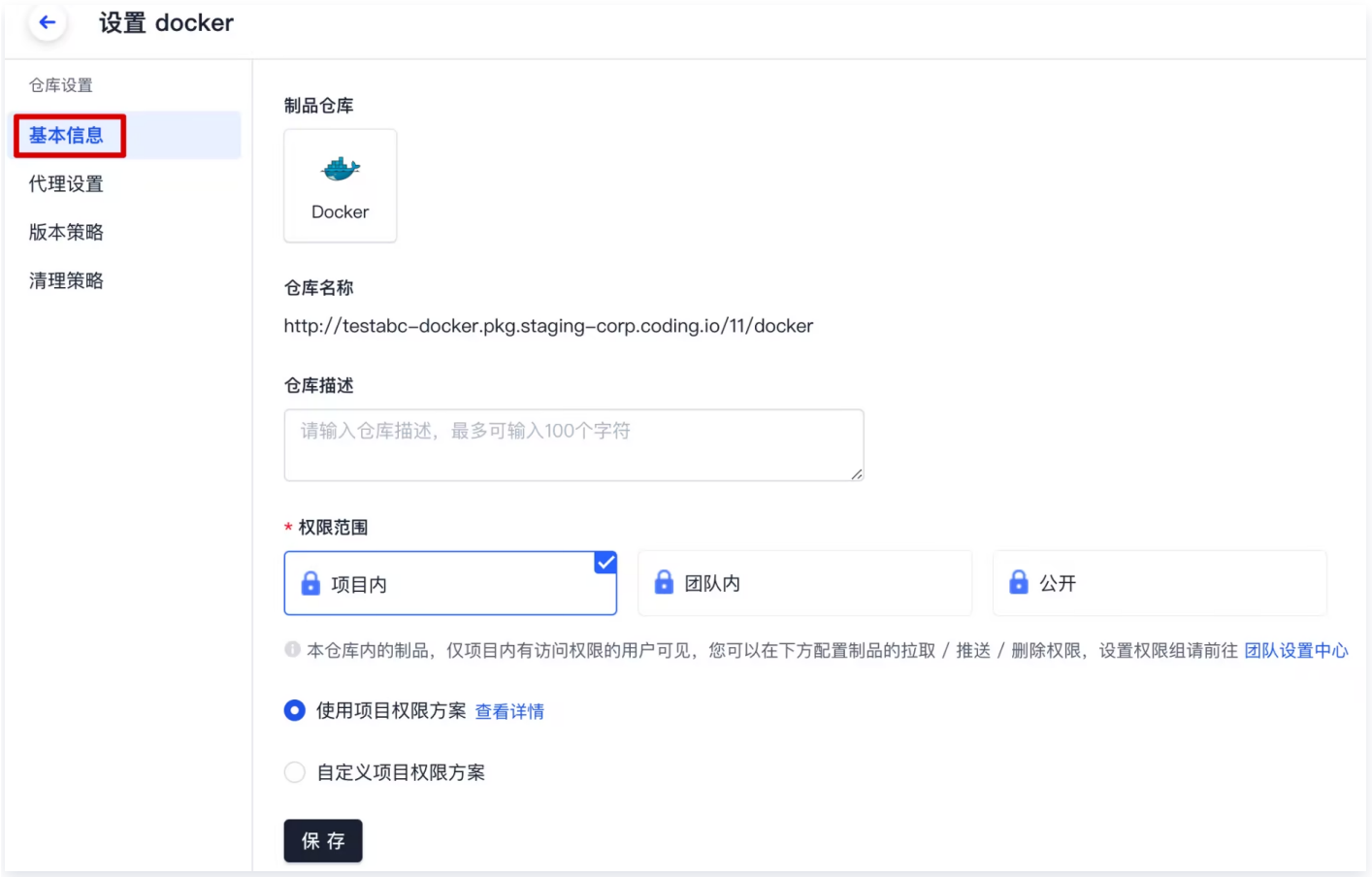
On the **Artifact Repository** page, you can set the permission scope when creating a new artifact repository.

Modify permissions for an existing repository

Click **Repository Settings** at the top right of the repository.



You can update the permission scope in **Basic Information**.

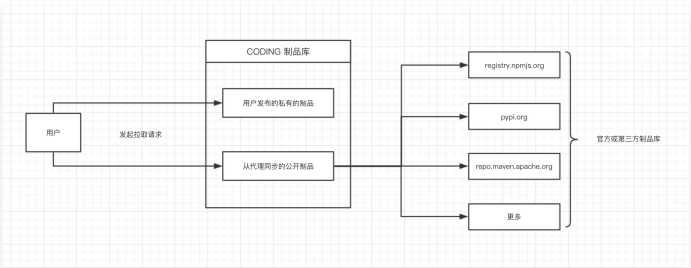


Repository Proxy

Last updated: 2026-03-27 17:56:30

Description of the Feature

The repository proxy feature supports users in configuring multiple sources for the repository proxy. When the corresponding package cannot be found in a private repository, it will attempt to fetch the relevant package from the configured sources and return it to the user. Additionally, users can configure the account information for authentication of the proxy sources. The repository proxy feature can serve as a unified entry point to help users manage their third-party product dependencies.

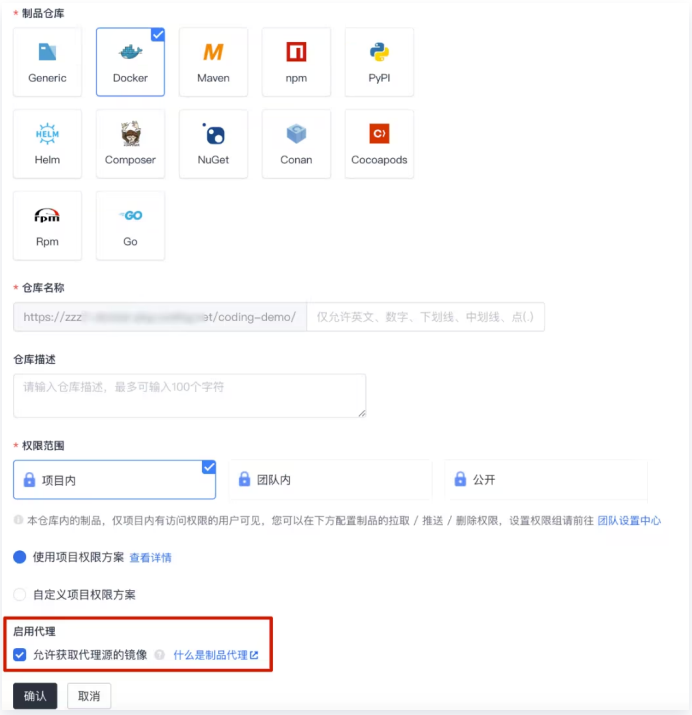


The order of package fetching is as follows:

1. Priority is given to fetching packages from the private repository.
2. If the package cannot be found in the private repository, it will then search through the configured proxy sources in order from top to bottom.

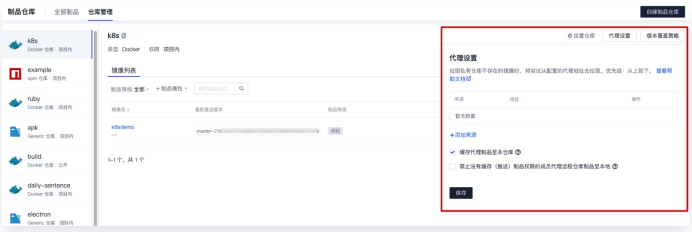
Enabling proxy

When creating a new repository under **Repository**, you can choose to **Enable Proxy**, which is enabled by default.



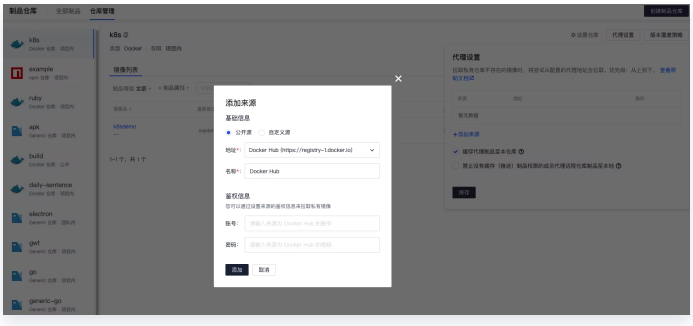
Configure Proxy

Select a specific repository, click **Proxy Settings**, where you can add or remove proxy sources, adjust the priority of proxy sources, and decide whether to cache to the remote product repository, among other operations.



Add Source

After clicking **Add Source**, you will enter the create source page, provide the address and name, and if necessary, fill in the authentication configuration information. Click **Add** to complete.



Modify Authentication

To modify the authentication information of the proxy source, go to the proxy source list page, click **Configure**, and make the necessary changes. The built-in proxy addresses of the repository are as follows:

Type	Name	Address
npm	npmjs	https://registry.npmjs.org
npm	TencentCloud npm	https://mirrors.cloud.tencent.com/npm
PyPI	PyPI	https://pypi.org/simple
PyPI	TencentCloud PyPI	https://mirrors.cloud.tencent.com/pypi/simple
Maven	Maven Central	https://repo.maven.apache.org/maven2
Maven	TencentCloud Maven	https://mirrors.cloud.tencent.com/nexus/repository/maven-public
Composer	TencentCloud Composer	https://mirrors.cloud.tencent.com/composer/

Allow Proxy IP

If there are issues with proxy fetching, it may be because the target product source is blocking network requests from CODING IP. Allowing the relevant IP addresses will resolve this issue. The service IP exit used by the CODING platform is:

```
212.129.144.0/24
212.64.105.0/24
49.234.127.0/24
49.235.224.0/24
49.234.65.0/24
81.69.101.0/24
```

Use Proxy

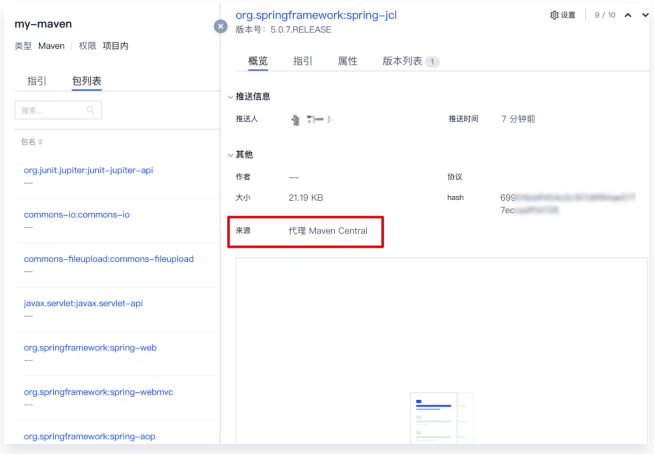
After successfully configuring the proxy, you can use the proxy source to fetch dependencies.

How to identify if artifacts in the repository are synchronized from a proxy?

1. For example, with Maven artifacts, you can see similar logs when you execute maven install locally:

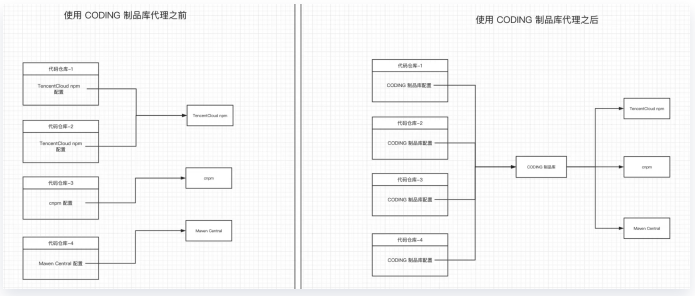
```
[INFO] Downloading from : https://xxxxxxxx-
maven.pkg.coding.net/repository/coding-demo/my-
maven/org/springframework/spring-jcl/5.0.7.RELEASE/spring-jcl-
5.0.7.RELEASE.pom
[INFO] Downloaded from : https://xxxxxxxx-
maven.pkg.coding.net/repository/coding-demo/my-
maven/org/springframework/spring-jcl/5.0.7.RELEASE/spring-jcl-
5.0.7.RELEASE.pom (1.9 kB at 735 B/s)
```

2. Additionally, on CODING-AR, you can also see the source of the artifact.



What is the difference between fetching artifacts directly from third-party sources and fetching through CODING-AR proxy?

The artifact repository helps you manage the configuration of artifact sources within your team uniformly. You can see the usage by team members within CODING-AR and use CODING artifact scanning to detect any security vulnerabilities, directly auditing the artifact security within the team.



Authentication of the Artifact Repository

Last updated: 2024-09-05 16:41:26

Description of the Feature

When a user accesses the artifact repository, the repository verifies the credentials provided by the user to ensure that the user has the appropriate permissions.
The artifact repository supports multiple authentication methods:

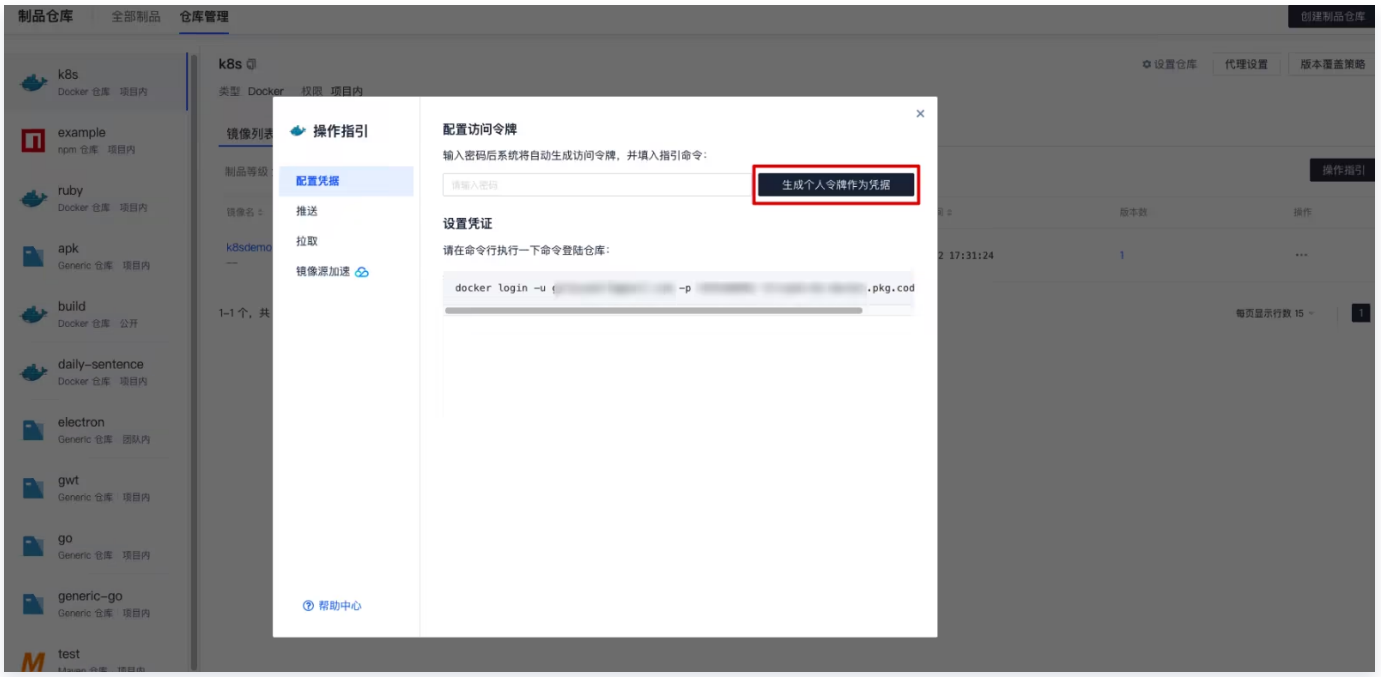
- Personal Access Token
- User Account Password
- Project Token

Each artifact repository has a different command to configure credentials locally, but the logic is similar. There are guides for setting credentials on the artifact repository page. This document uses the Docker artifact repository as an example to demonstrate three methods for configuring authentication credentials.

Personal Access Token

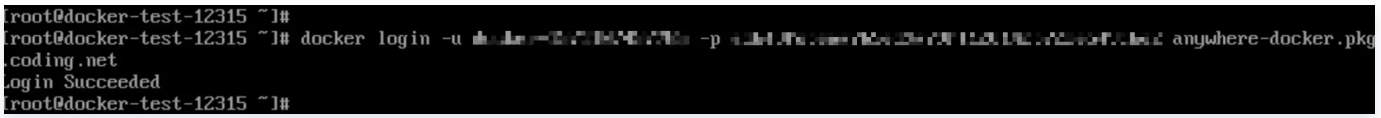
A Personal Access Token (Access token) contains a user's security authentication information, allowing the user to view or operate corresponding resources. Using a personal access token for artifact repository authentication is recommended as it is safer than directly configuring a user account password.

1. Click **Generate Personal Token as Credential** on the artifact repository guide page.



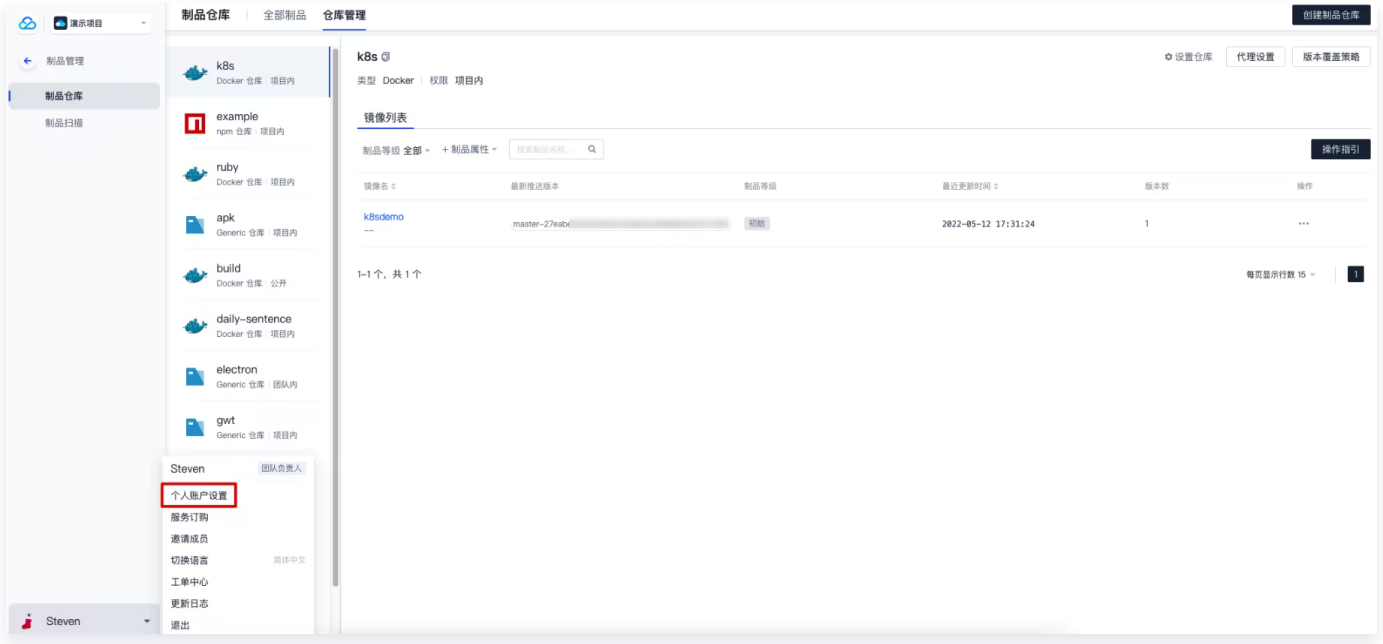
2. After entering the authentication information, you will see the command that carries the new access token. Click **Copy** to copy the command.

3. In the local Docker environment, execute the copied Docker login command. If you successfully log in, you can proceed with the next Push/Pull operations.



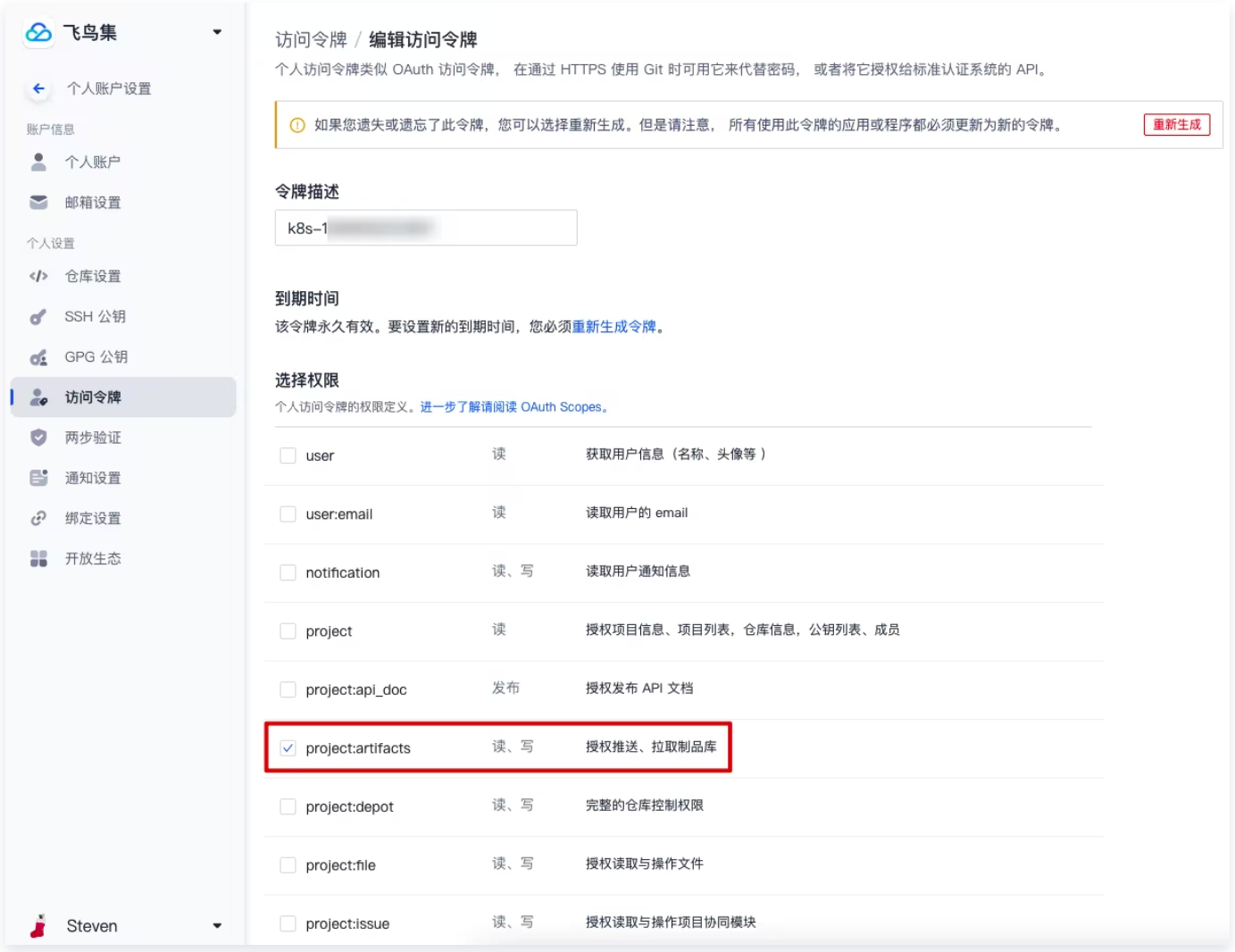
View Personal Tokens

1. Click **Personal Account Settings** in the lower-left avatar frame.



2. On the personal account page, click **Access Token** to see the personal access token information created through the artifact repository in the previous steps. You can also view the usage records of the personal access token on this page.

3. Click **Edit** on the right side of the token to view or modify the permission information of the access token.



User Account Password

Credential information can also be set by using the user account (phone number or email) password.

1. On the Artifact Repository guide page, directly copy the default provided Docker command and replace the password with the login password of the personal account.

2. Execute the command on the local Docker page, enter the password to authenticate successfully for the next push or pull operation.

Project Token

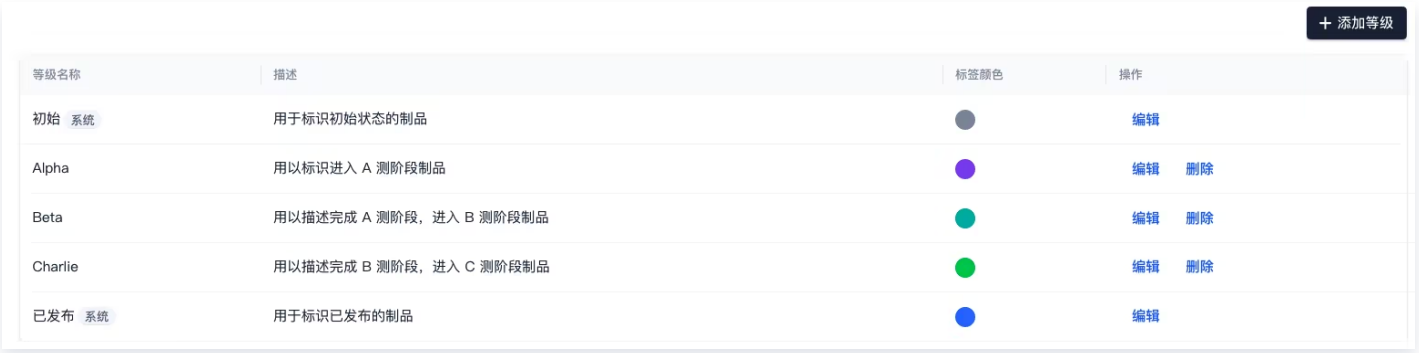
CODING provides the project token feature, allowing project administrators to configure the push-pull permissions of the artifact repository within the project based on actual needs. For detailed operations, please refer to Project Token.

Artifact Promotion

Last updated：2024-09-05 16:41:42

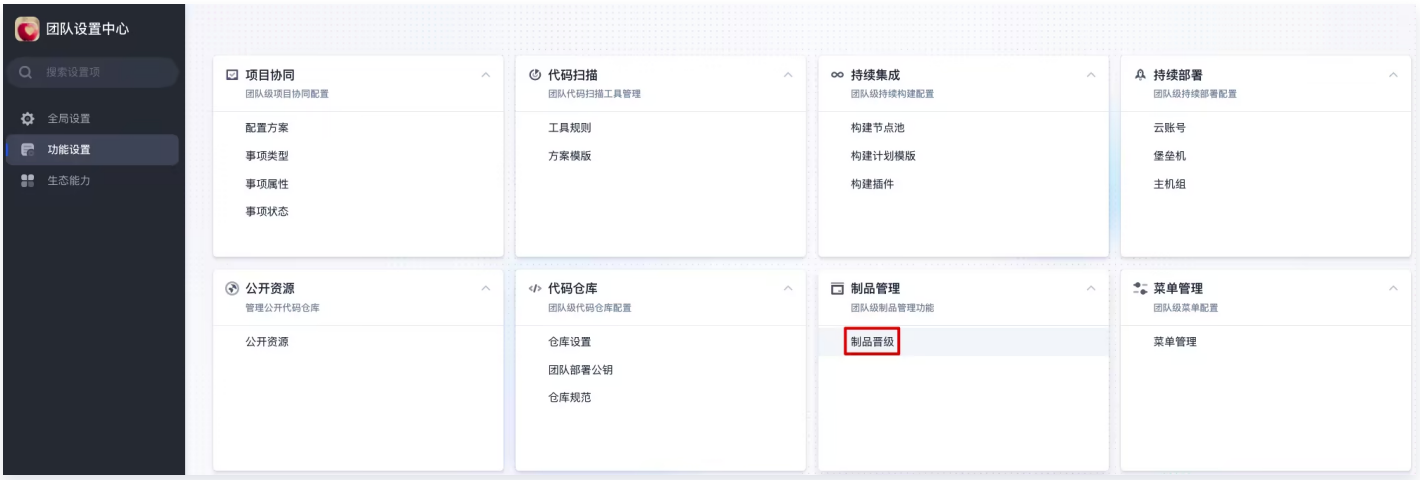
Description of the Feature

Simply using the artifact version number is not sufficient to effectively determine if an artifact has reached a deliverable level. The Artifact Promotion feature provides definition artifact levels and promotion rules to help R&D teams effectively and intuitively distinguish the maturity of artifact versions, ensuring that the final delivered artifact version is qualified and reliable. Team members can configure different promotion rules for the artifact repository based on business needs to improve artifact quality and version differentiation.



Set Artifact Levels

Similar to issue flow in project collaboration, artifact promotion essentially describes the delivery status of artifacts intuitively. Go to **Team Settings Center > Feature Settings > Artifact Promotion** to create artifact levels.



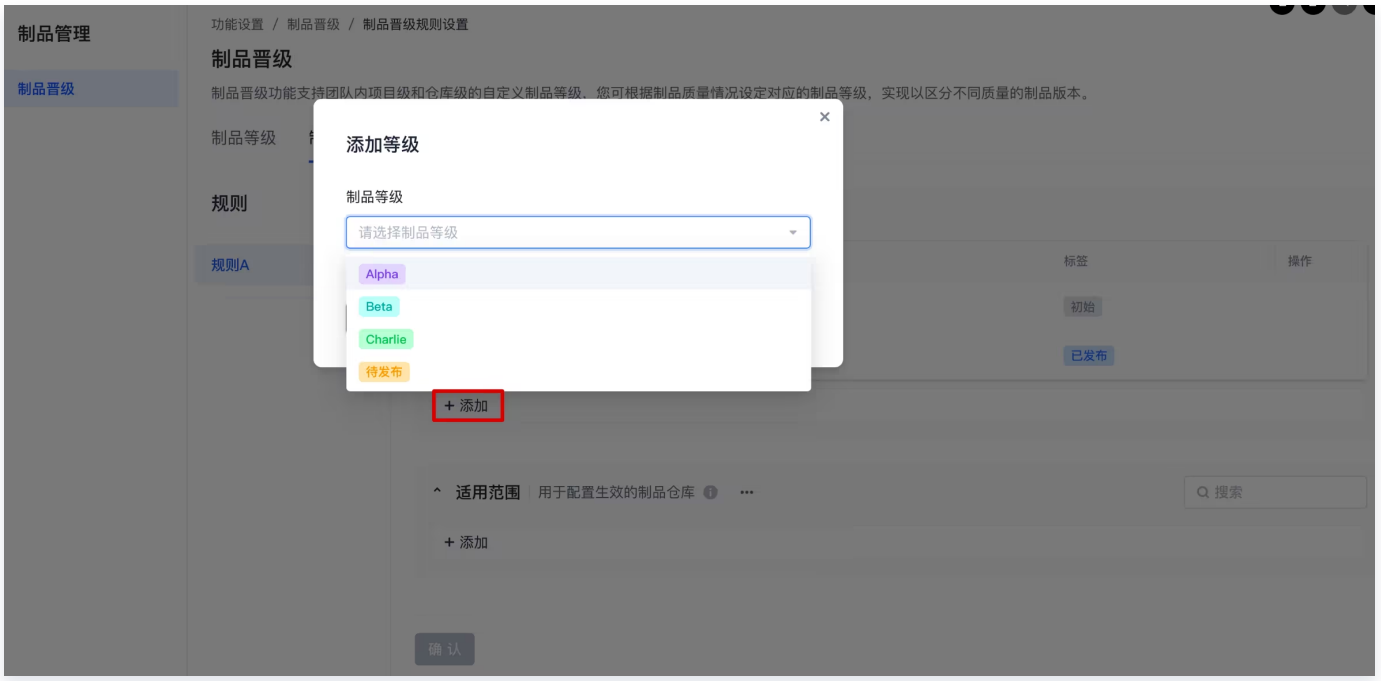
You can define artifact levels according to the test delivery process, such as Initial > Alpha Test > Beta Test > To be Published > Published; or establish levels based on the team’s internal consensus on artifact quality. Enter the name and select the color to complete the creation.

After creating the levels, you need to further set promotion rules to apply them to the artifact repository.

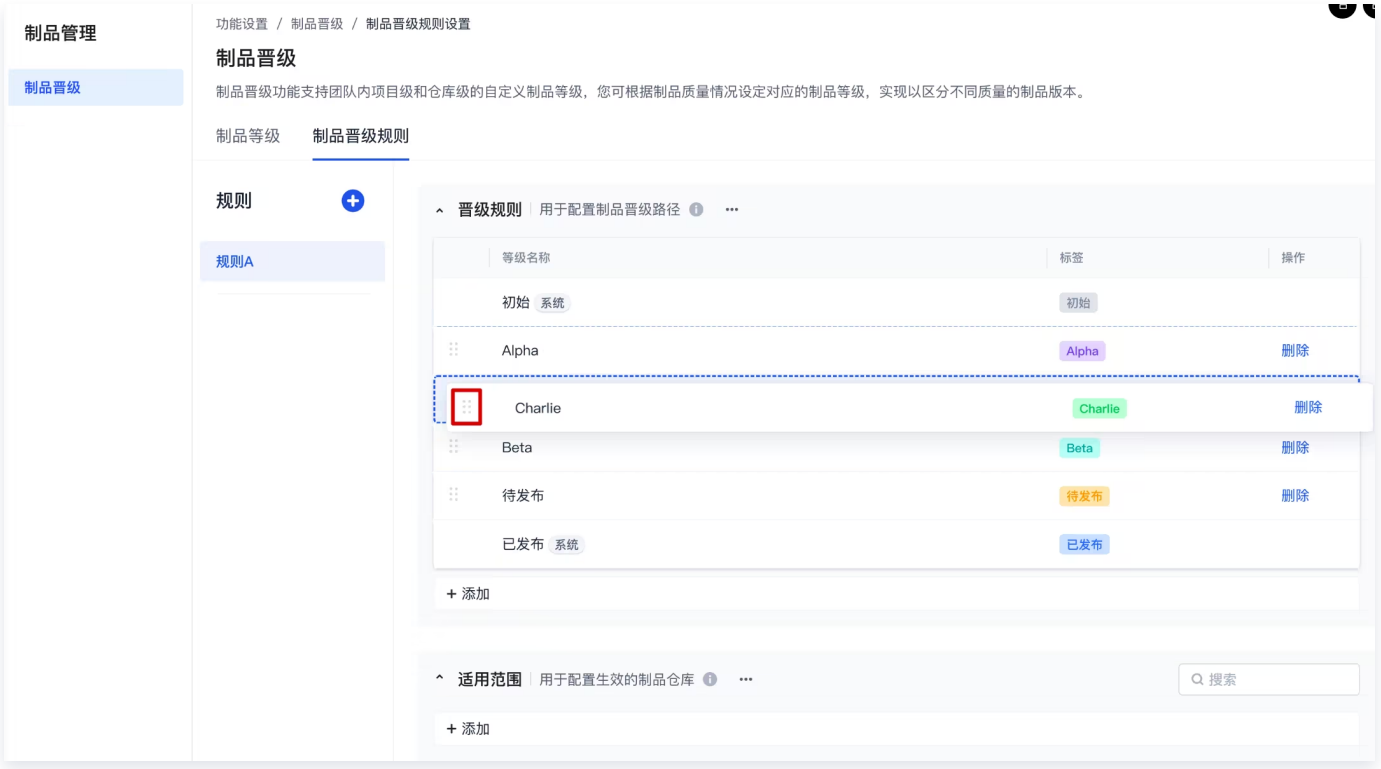
Set Artifact Promotion Rules

Promotion rules are used to define the flow sequence of artifact levels, for example, to stipulate that the artifact status must go through various tests from the **Initial** level to reach the **To be Published** level. After entering the artifact promotion option, go to the **Artifact Promotion Rules** page to create promotion rules.

1. Circle the artifact levels into the **Promotion Rules**.

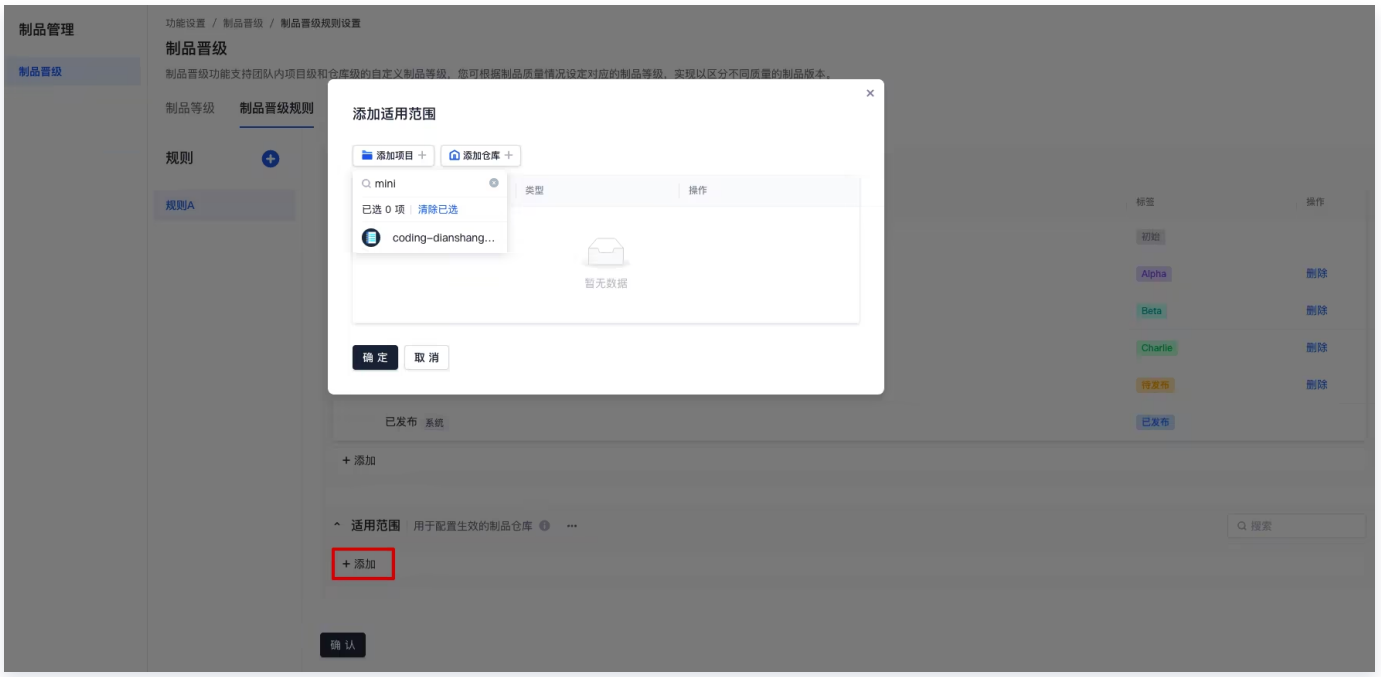


2. Drag the button on the left to adjust the flow sequence of the levels.



3. Check the artifact repository that the promotion rules apply to. After completion, click **Confirm** at the bottom to save.

> **Note:**
> A single project or artifact repository can support multiple rules, allowing the selection of a level in the actual artifact flow process.

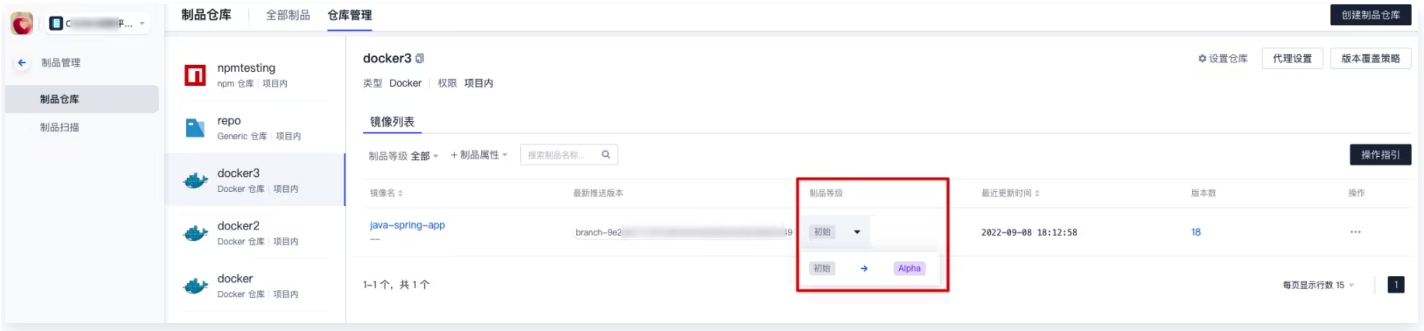


Apply Artifact Promotion Rules

In the artifact management's artifact list and artifact historical versions page, adjust the artifact levels to help team members understand the current status of artifacts and quickly ascertain if the current version's artifact quality has reached the to be published level.

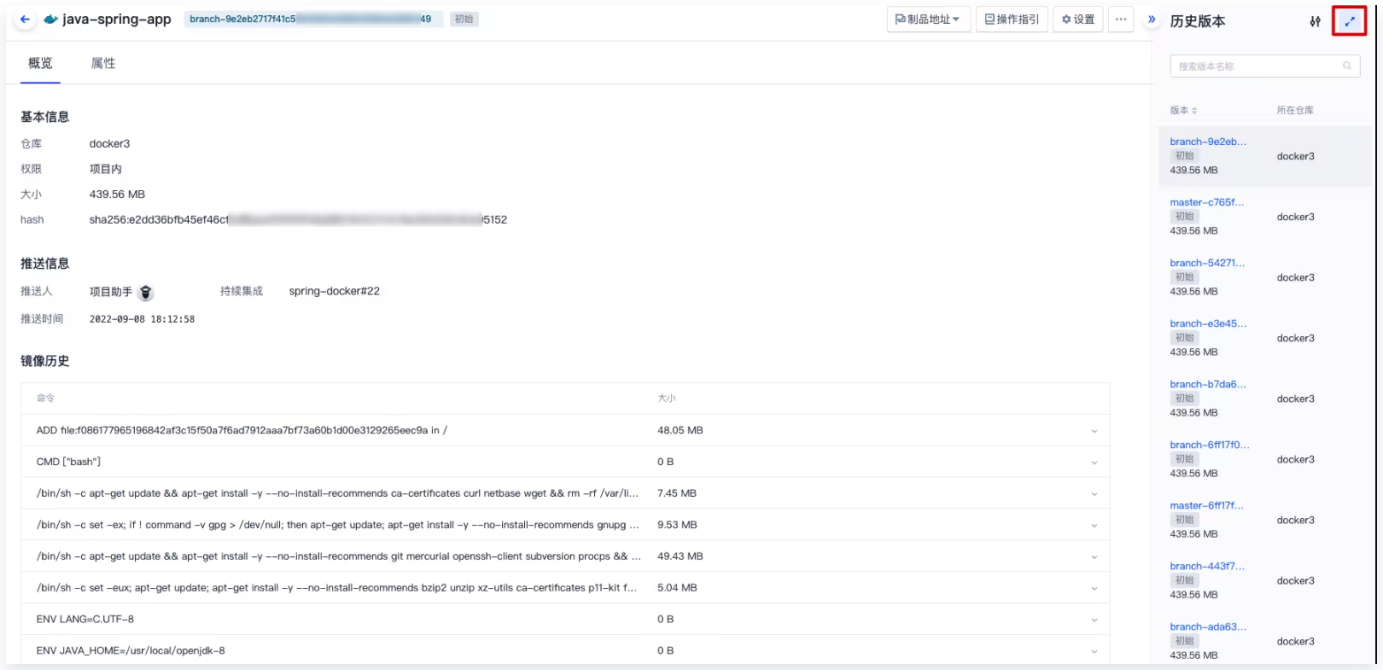
Artifact List

Enter the Repository Management page, manually adjust the artifact levels in the list.

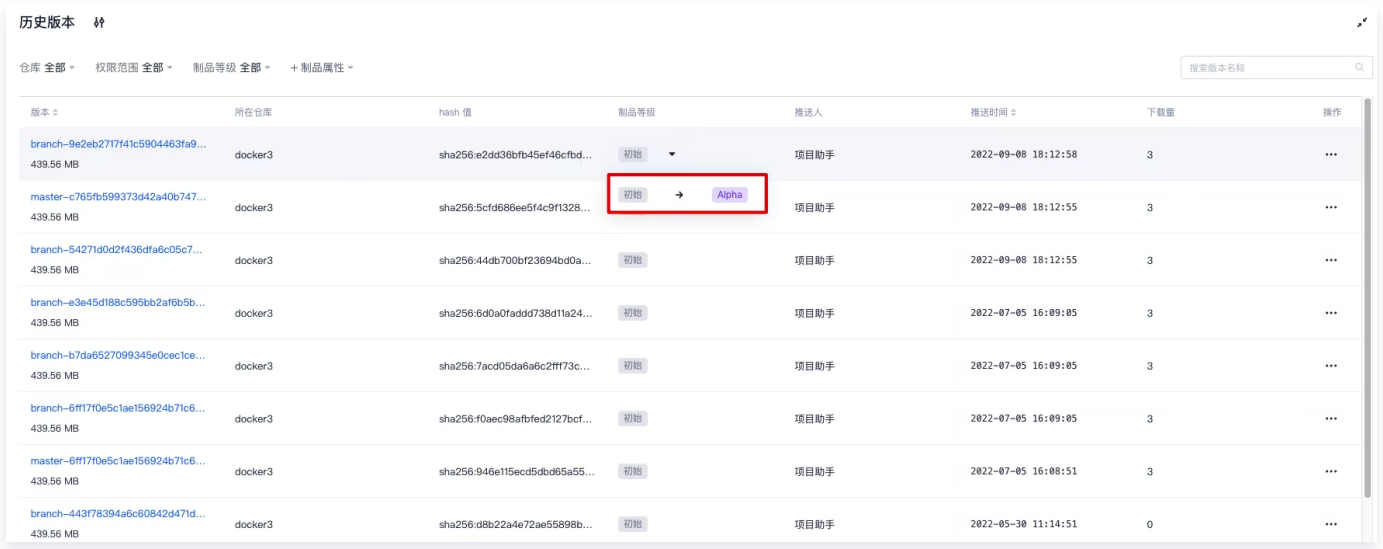


Historical version

1. Click the fullscreen button at the top right of the Artifact Overview to expand and view the historical versions of the artifact.



2. Manually adjust the artifact status in the artifact levels.



Product Version Override Strategy

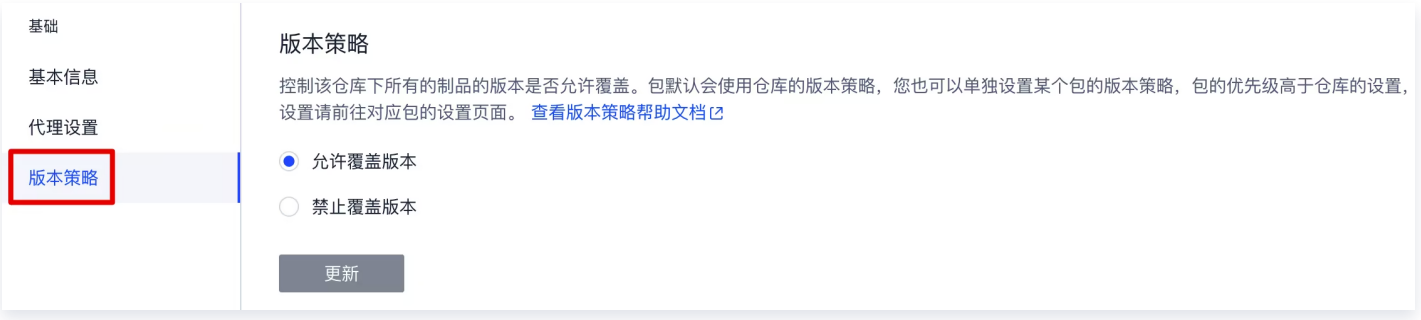
Last updated: 2024-09-05 16:41:57

During the development phase, each code modification may produce a new product. Developers might need to frequently change version numbers to use the latest version. This can be very detrimental to development and debugging, as arbitrarily overriding the version of the same product in the production phase can lead to management chaos. **Ensuring each product has a unique version number guarantees the same behavior for the same version product, which is very meaningful for deployment and application lifecycle management.** CODING-AR provides flexible version override strategies, ensuring the uniqueness of Docker image versions while allowing repeated publishing of the same npm package version. You can set the **Repository/Package/Version** strategy according to the product's lifecycle needs. This article will introduce how to set up product version override strategies at these three levels, as well as the default version override strategies provided by the artifact repository.

Repository Version Override Strategy

Click **Artifact Repository** > **Set Repository**.

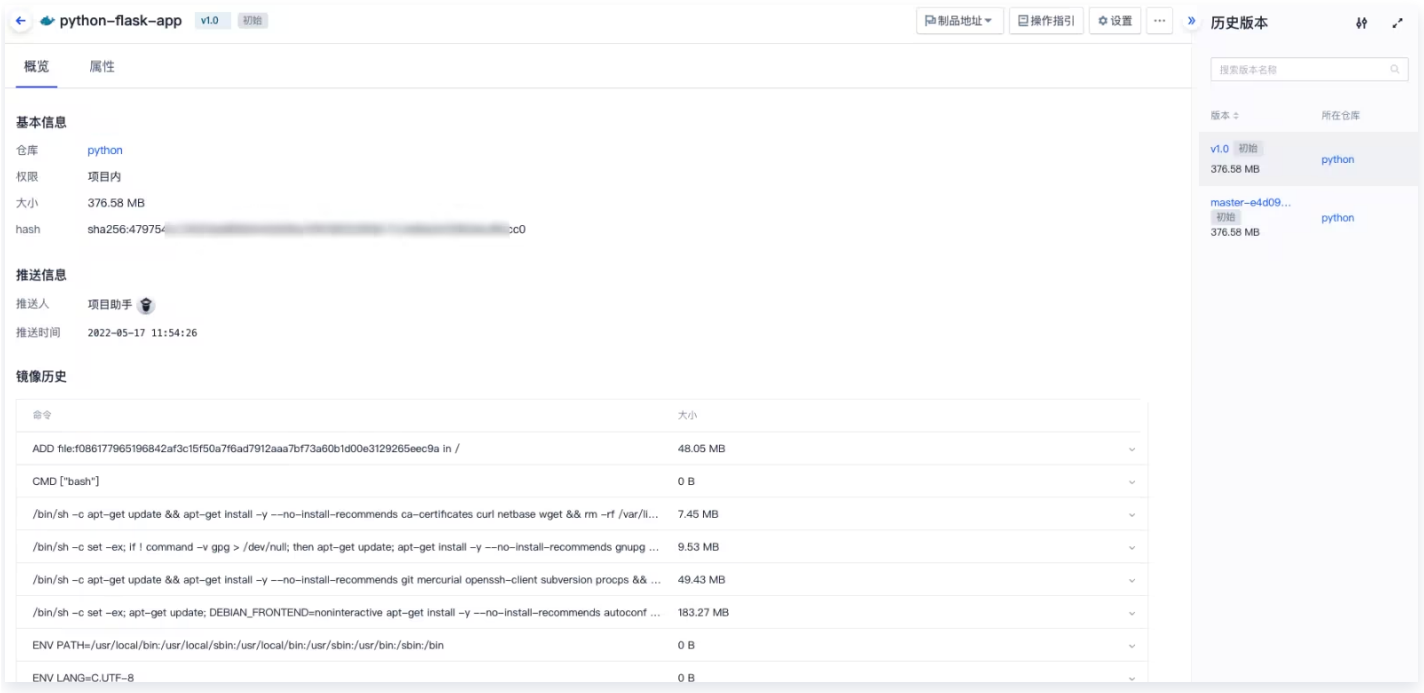
Click **Version Strategy** to set whether versions of all products in the repository are allowed to be overridden.



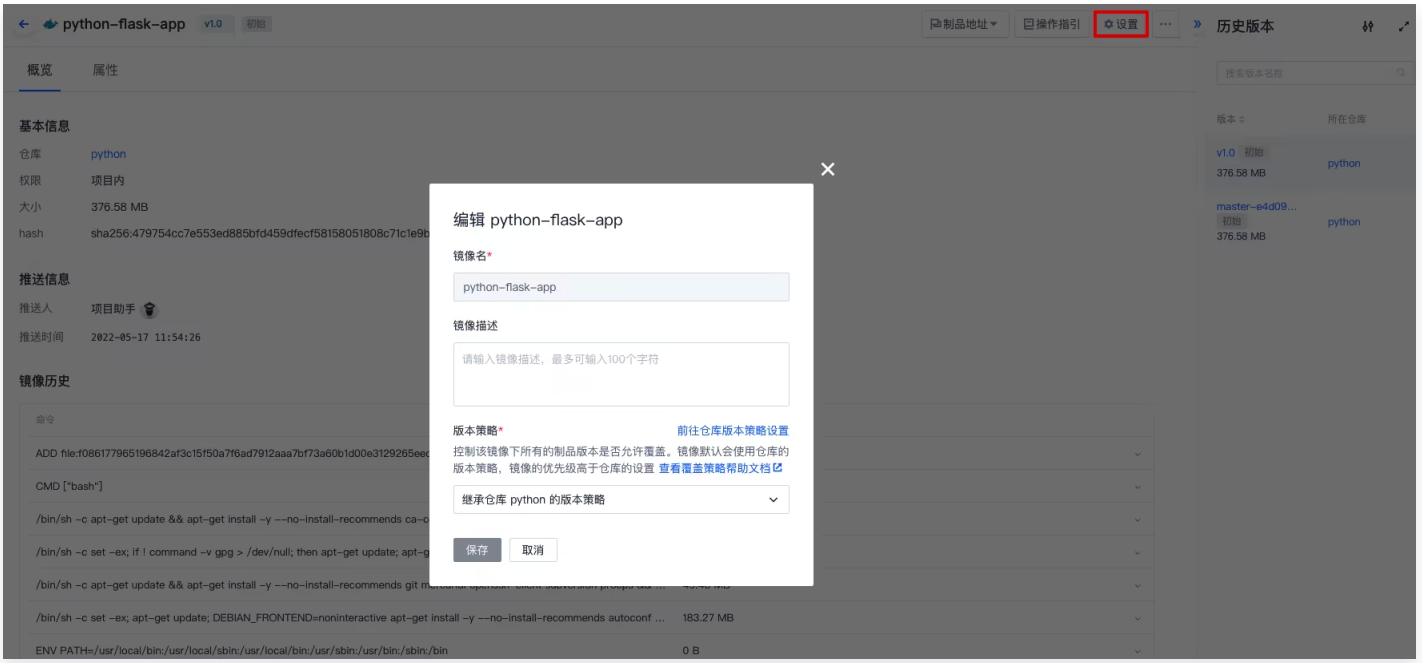
> **Note:**
> Currently, Maven is special and has an additional feature: using Maven SNAPSHOT.

Package Version Override Strategy

Click the specific package name to see the package details page on the right.



Click **Settings** to select the version strategy for the package. By default, the package uses the repository's version override strategy.



Default Version Override Strategy

The artifact repository provides a default version override strategy according to the native logic of the product type, as follows:

Product Type	Repository	Package	Version
Docker	Allow same version to be released	Inherit repository rules	Unreleased
Maven	Maven SNAPSHOT	Inherit repository rules	Unreleased
npm	Do not allow same version to be released	Inherit repository rules	Unreleased
PyPI	Do not allow same version to be released	Inherit repository rules	Unreleased
Generic	Allow same version to be released	Inherit repository rules	Unreleased
Helm	Allow same version to be released	Inherit repository rules	Unreleased
Composer	Do not allow same version to be released	Inherit repository rules	Unreleased
NuGet	Do not allow same version to be released	Inherit repository rules	Unreleased
Conan	Allow same version to be released	Inherit repository rules	Unreleased

Artifact Clean-up Policy

Last updated: 2024-09-05 16:42:11

The clean-up policy of the artifact repository can promptly clear old versions of artifacts. You can quickly clean up excess artifacts by setting a clean-up policy to free up storage space. Currently supported artifact types for clean-up policy settings:

- Docker
- Generic
- Gradle
- Helm
- npm

Configure Clean-up Policy

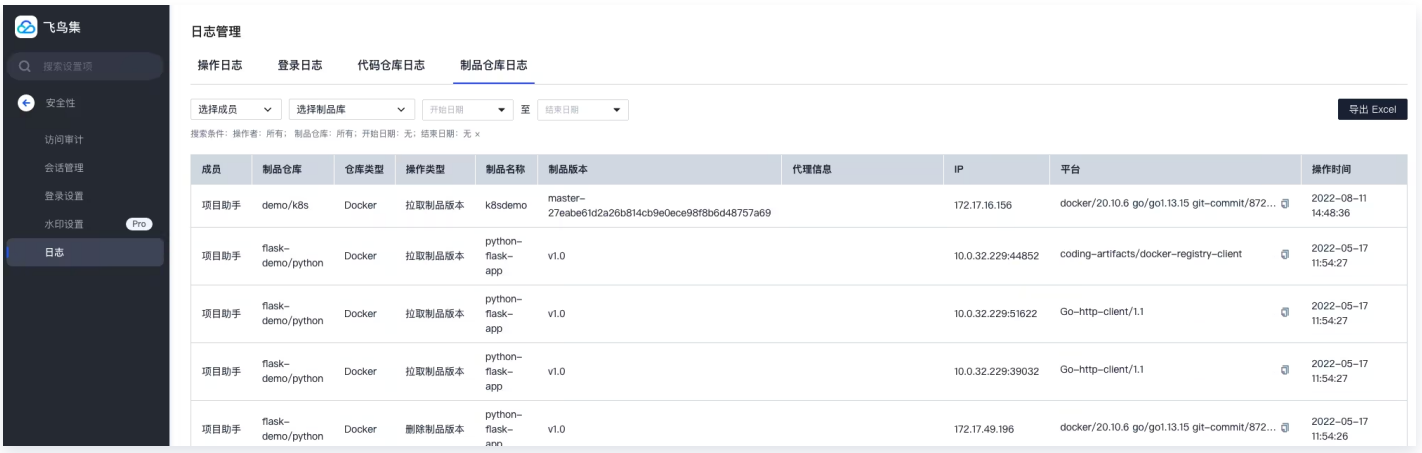
After entering the artifact repository page of the above types, click **Repository Settings > Clean-up Policy**.

You need to fill in two trigger conditions in the clean-up settings, then choose to execute automatic or manual clean-up. Only artifacts that meet both trigger conditions will be included in the clean-up list.

View Clean-up Records

Go to the Team Settings Center and view the operation log of the artifact repository in the logs.

The recent artifact repository logs will be displayed on the **Artifact Repository Log** page.



Product Scan Overview

Last updated: 2024-09-05 16:42:37

The scanning feature of the CODING Artifact Repository can detect vulnerabilities in binary components and their metadata without accessing the source code. The artifact scanning feature supports integration with Continuous Integration/Continuous Deployment modules. You can preset quality redline standards in the solution to prevent problematic components from being released to the production environment. Additionally, the scanning solution also provides detailed scan records and defect statistics.

Product Features

Scanning Solution

Defines the scanning rules, quality redlines, and the artifact packages to be scanned. Scanning solutions can only be applied to artifact repositories within the current project, and each scanning solution has a unique scan ID. A scanning solution can be applied to any repository, artifact package, and artifact version within the current project. It can be triggered automatically when the artifact package is updated or manually for specific artifact versions.

Scanning Rules

Defines the defect contents to be focused on during scanning. You can specify the vulnerability levels of concern and set up a vulnerability allowlist.

- Vulnerability Risk Level Rules:

  Definition The vulnerability severity levels that this scanning plan focuses on. If the **low risk** level is not selected, then all vulnerabilities of low risk level will not be detected and not included in the vulnerability statistics.

- Vulnerability Allowlist (Security Scanning Only):

  After including specific CVE vulnerability numbers in the allowlist, all public components containing those CVE vulnerabilities will not be scanned. If specific dependent components are included in the allowlist, all vulnerabilities in those components will not be scanned.

Quality Redline

Defines the standards for passing the scan results. You can set redline standards according to your team's security requirements, including limitations on vulnerability levels, open source licenses, and quantities.

Component Analysis (SBOM)

Currently, analysis features are only provided for Generic, Maven, Docker, and npm artifacts. The Component Analysis (SBOM) feature helps enterprises analyze dependencies within artifacts and provides information on component versions, vulnerabilities, and licenses. Meanwhile, the system also analyzes the relationships between artifacts and their dependencies, including the components that artifacts depend on and the artifacts associated with those components, providing effective basis for tracing software supply chain risks and for security governance.

Open Source License Risk

Developers must be aware of license violation risks when using open-source software. Artifact Scanning not only performs vulnerability scans but also scans open-source

components' licenses, displaying information on license risk levels, sources, constraints, and associated components.

Solution Application

A single scanning scheme can be reused for any versions of artifacts in various types of repositories. It can be triggered automatically when an artifact package is updated or manually.

Plan Types

The scan plan supports two types of scans: security vulnerability scanning and mobile app package quality check. The mobile app package quality check is primarily provided by Tencent Cloud Package Quality Inspection IPT. In addition to these two types, artifact scanning will continue to expand its capabilities.

- **Security vulnerability scanning:**

  Scans for security vulnerabilities in artifacts and their dependent public components. Definitions of vulnerabilities use the internationally standardized CVE and ratings; the component vulnerability database used by artifact scanning is maintained by Yunding Lab, providing over 20 years of industry experience to Tencent Security and updated in real-time by a dedicated security operations team.
  Supports the following types of artifact repositories: Docker, Maven, npm, PyPI, Generic.
- **Mobile app package quality scan:**

  Conducts standard checks on installation packages for Android/iOS operating systems, including package size, duplicate files, images, etc.
  Supports .ipa and .apk format artifact files in Generic type repositories.

Security Vulnerability

Vulnerability definitions use the internationally standardized CVE and follow the CVSS standard for danger levels. CODING Artifact Scanning exposes security vulnerabilities in artifacts and their dependencies. Vulnerability information includes CVE ID, introduced dependency components, danger level, and vulnerability description.
The definition of a vulnerability is determined by its CVE ID and the public component it exists in. For example, if a security vulnerability with the ID CVE-2021-45105 is found in the public component Log4j version 2.17, the scan result will display this vulnerability as:

- CVE ID: CVE-2021-45105
- Dependent Component: Log4j
- Introduced Version: 2.17

Artifact Vulnerability Library

CODING artifact scanning utilizes the Tencent Security Vulnerability Feature Library.
The Tencent Security Vulnerability Feature Library is a proprietary, commercially maintained vulnerability database developed by Tencent's Keen Lab. It contains information from both domestic and international open-source vulnerability databases (including CVE, NVD, CNVD, CNNVD, etc.) and unique commercial vulnerability information discovered by Tencent. The Tencent Security team verifies open-source vulnerability information for false positives, translates it into Chinese, and provides repair priority and suggestions, ensuring industry-leading scanning accuracy. For unique commercial vulnerabilities discovered in-house, it provides a unique vulnerability ID, detailed description, and repair recommendations, covering basic vulnerability information.
For private clients, we offer both real-time and offline update methods.

Real-time Updates

In terms of real-time updates, it supports configuring the domestic Internet update address and Tencent Cloud account key authentication in the local vulnerability database. During each scan, it checks if the vulnerability database is the latest version. If not, it updates the database before executing the scan.

Offline Updates

For offline updates, it also supports offline packaging and deployment of the vulnerability library to the company's intranet, providing update services.

Quick Start

Last updated: 2026-04-03 17:28:04

This article uses the sample artifact fastjson as the scanning target. If there are already artifacts in your project, you can scan them directly.
Thanks to the CODING Artifact Proxy feature. When pulling artifacts to the local, they will be automatically uploaded to the Artifact Repository. Click Sample Artifact for relevant information. If you are not sure how to use the Maven Repository, please refer to Quick Start Maven Repository.

Create Scanning Plan

Go to **Artifact Management > Artifact Scanning**, click the blue + in the upper left corner to create a new scanning plan, enter the plan name, description, scanning rules, and Quality Red Line Standards to complete the creation.

Plan Types

The scanning plan supports two types: Security Vulnerability Scanning and Mobile Installation Package Quality Check. The Security Vulnerability Scanning type allows for filtering the vulnerability level and CVE vulnerability allowlist. The Mobile Installation Package Quality Check is primarily provided by **Tencent Cloud Installation Package Quality Check IPT**.

Scanning Rules

Scanning rules determine the vulnerabilities that can be detected in the scanning plan. If only the **high-risk** vulnerability level is selected, other levels of vulnerabilities will not be counted, even if there are vulnerabilities higher than **high-risk**.

CVE Vulnerability Allowlist

It is used to ignore specific types of vulnerabilities during the scan. You can define a CVE vulnerability number for a certain type of vulnerability and fill it in the allowlist. If there is a CVE-2020-9794 type of vulnerability in the artifact, it will not be included in the statistics. Click to view CVE Vulnerability Inclusion.

> **Note:**
> During the scan, the vulnerability scan will only be performed within the range of levels selected in the scanning rules. Other levels will be filtered out, and then the specific vulnerability number specified in the CVE vulnerability allowlist will be determined. For example, if the user only selects the **Critical** vulnerability level in the vulnerability scan

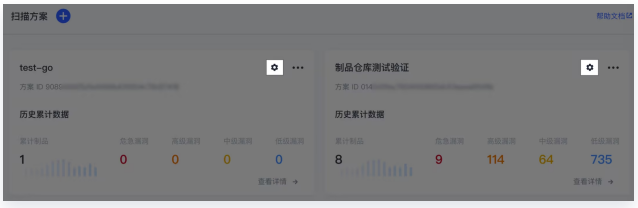
> but fills in a **low-risk** level vulnerability in the CVE vulnerability allowlist, this level of vulnerability will be ignored.

Advanced Options

In the advanced configuration, artifacts that have not completed scanning or have quality issues can be automatically prohibited from being downloaded to prevent vulnerable artifacts from being accidentally used.

Edit Scanning Plan

For the created scanning plan, click the **Settings** or **Rule Configuration** in the upper right corner to enter the settings page.



Here you can edit the plan name or description, review and reselect the vulnerability levels and Quality Red Line Standards in the scanning rules.

Trigger Scanning Plan

The scanning plan supports both automatic and manual triggering.

Automatic Scanning

Click the ••• in the upper right corner to enter the Solution Application page, where you can configure the automatic triggering scanning plan.

Once you turn on the automatic scanning switch, the current plan will automatically trigger a scan when there are updates in the applied artifact repository.
Scan and Filter support both all and by condition:

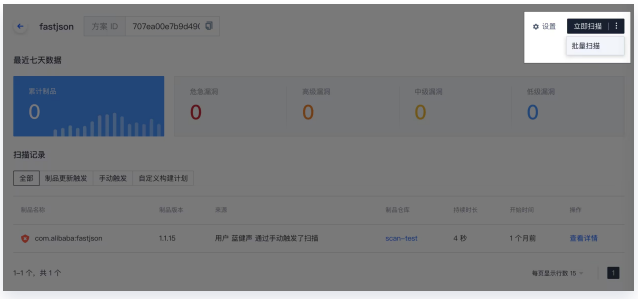
- The default is **All**, which means the scanning plan will be automatically triggered whenever any artifact repository gets updated.
- After selecting **Filter by conditions**, you can set the scope of application and triggering conditions for the scanning plan. For example, as shown below, the scanning plan will be

automatically triggered whenever there are any updates in the pypi-go and write-go artifact repositories.

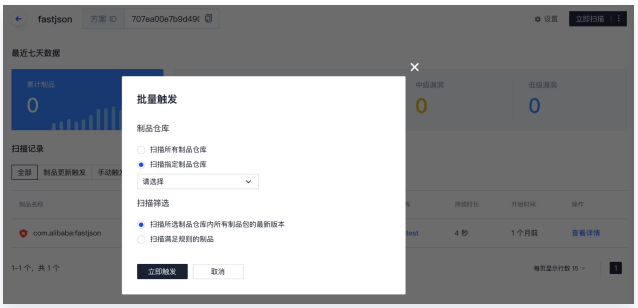
You can also add more detailed artifact filtering conditions to the scanning plan. As shown below, when the release version of the test artifact gets updated, it will be scanned individually.

Manual Scanning

There are three ways to manually trigger a scan: trigger a single scanning plan, batch trigger artifact scanning, and add artifact scanning in the continuous integration pipeline.



Click the batch scan in the upper right corner of the scanning plan to perform the scan in all artifact repositories, or set the scan range and scan filtering conditions.



In the continuous integration pipeline, manual addition of artifact scanning units is also supported.

Analyze Scan Results

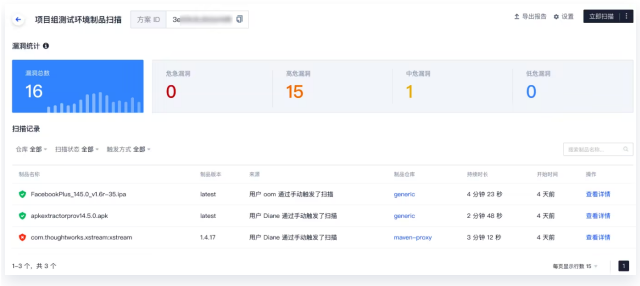
Icon Labels

The meaning of various result icons:

Hover the mouse over the marker to pop up the scan results.

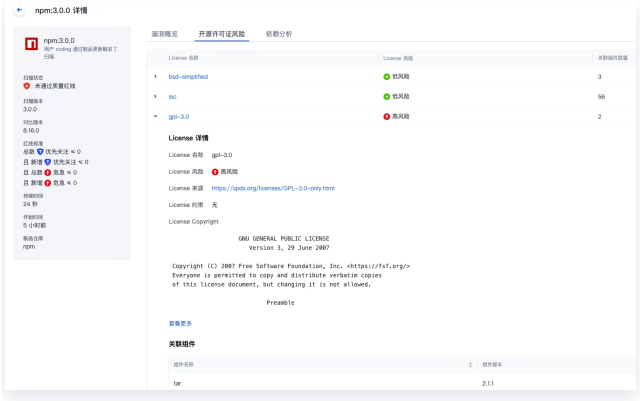
View Scan Results

In the scanning scheme details, you can view all historical cumulative data and scan records applied to the scheme.



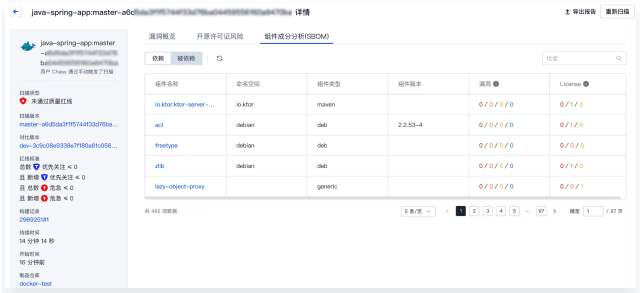
Open Source License Risk

Click **Open Source License Risks** to see the open source license names, risk levels, source information, and associated components of all components within the artifact.



Component Analysis (SBOM)

Currently, analysis features are only provided for Generic, Maven, Docker, and npm artifacts. The Component Analysis (SBOM) feature helps enterprises analyze dependencies within artifacts and provides information on component versions, vulnerabilities, and licenses. Meanwhile, the system also analyzes the relationships between artifacts and their dependencies, including the components that artifacts depend on and the artifacts associated with those components, providing effective basis for tracing software supply chain risks and for security governance.



Fix Vulnerabilities and Record

Click **View Details** or the artifact name to see the vulnerability details for that artifact version. The details page displays comprehensive information on all vulnerabilities, including vulnerability coding, level, dependent relationships, versions, remediation suggestions, and more.

For exposed vulnerabilities, you can follow the remediation suggestions in the vulnerability overview to fix them.

If a vulnerability does not cause serious consequences, you can mark it as **Ignore**; if you have taken actions to fix the vulnerability, you can mark it as **Repaired**.