CODING Artifact Repositories

Getting Started

Product Documentation

Contents

Getting Started
Basic Operations

Last updated：2024-01-02 10:29:54

This document describes how to use CODING Artifact Repositories (CODING-AR).

Prerequisites

You must activate the CODING DevOps service for your Tencent Cloud account before you can use CODING-AR.

Open CODING-AR

1. Log in to the CODING Console and click **Use Now** to go to CODING page.
2. Click

in the upper-right corner to open the project list page and click a project icon to open the project.
3. In the menu on the left, click **Artifact Management**.
Refer to the following sections for initialization as needed. Operations described in the following sections are not mandatory.

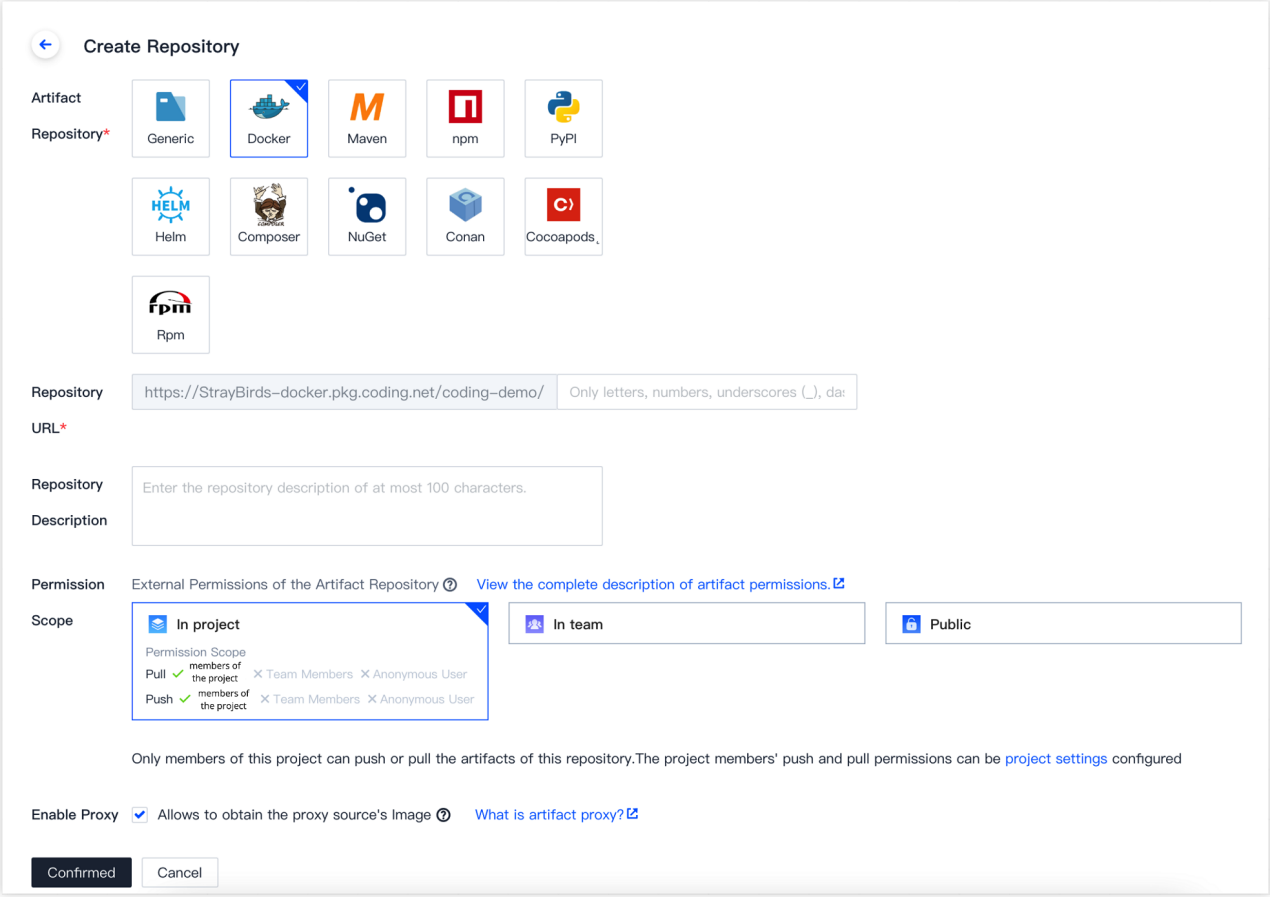
Create a Project

Before using CODING-AR, create a **DevOps project**.
Enter the required information to create a project. Then you can go to Artifact Repository from the left menu. If the function is hidden, enable it in **Project Settings** > **Functions**.

Create an Artifact Repository

Select an artifact repository type.
Enter a repository name.

Permission Scope: Configure the permissions of different roles on the current repository. By default, all project members have the **Pull** and **Push** permissions.
Click **Create**.



After creating an artifact repository, refer to the corresponding quick start guide to get started with CODING-AR.

Generic Repository
Docker Repository
Maven Repository
npm Repository
Helm Repository
PyPI Repository
Composer Repository
NuGet Repository
RPM Repository
Conan Repository
CocoaPods Repository

Generic Repository

Last updated：2024-01-02 10:30:46

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

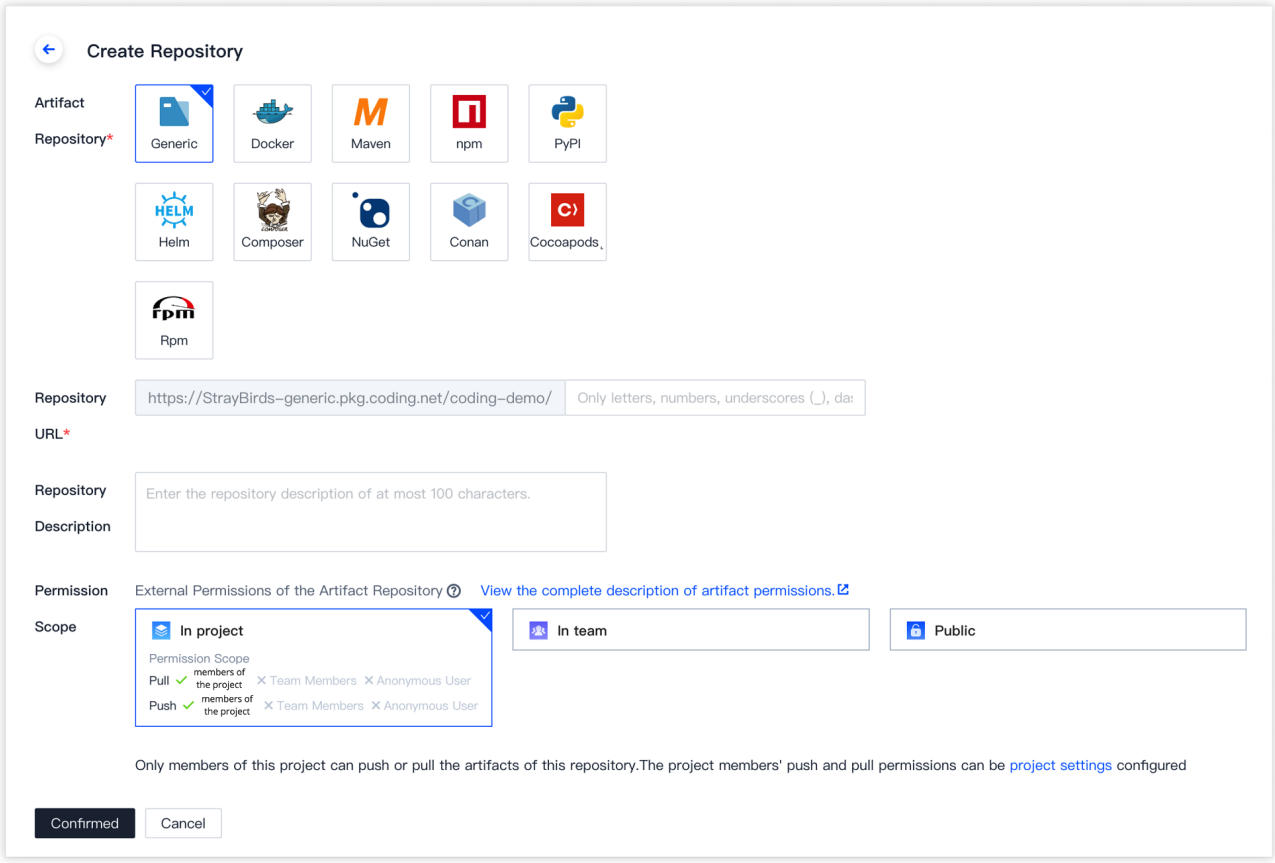
Open CODING-AR

1. Log in to the CODING Console and click **Use Now** to go to CODING page.
2. Click

in the upper-right corner to open the project list page and click a project icon to open the project.
3. In the menu on the left, click **Artifact Management**.

Create an Artifact Repository

Click **Create Repository**.
Select Generic as the repository type.
Enter a repository name.
Enter a repository description (optional).
Configure the permissions of different roles on the current repository. By default, all project members have the **Pull** and **Push** permissions.
Click **Create**.



Push a Generic Artifact

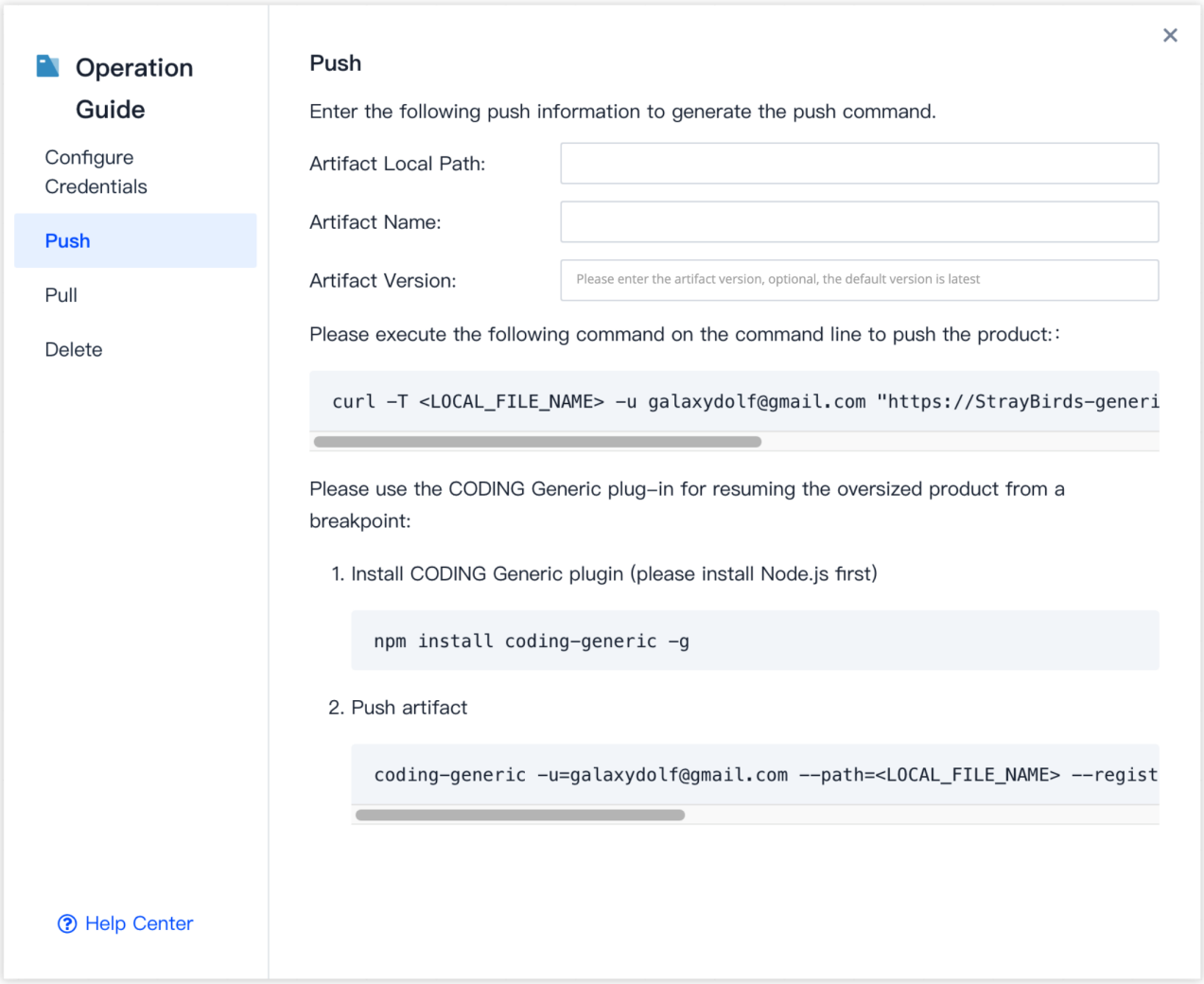
You can push a generic artifact in either of the following ways:

Command line

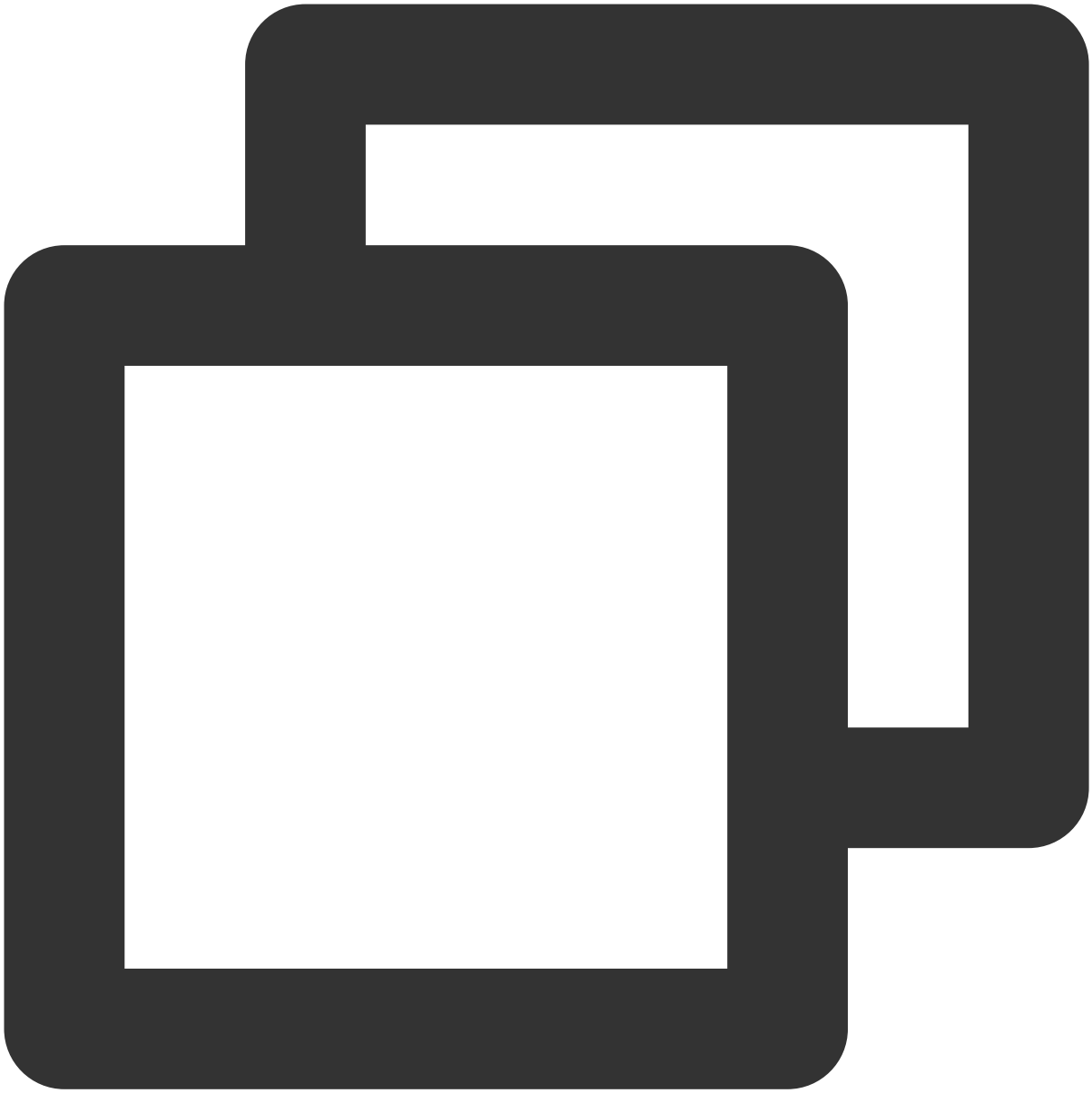
Direct upload

Upload via command line

Push an artifact using the command in the **Guide**.



For example, to push a file named demo.properties:

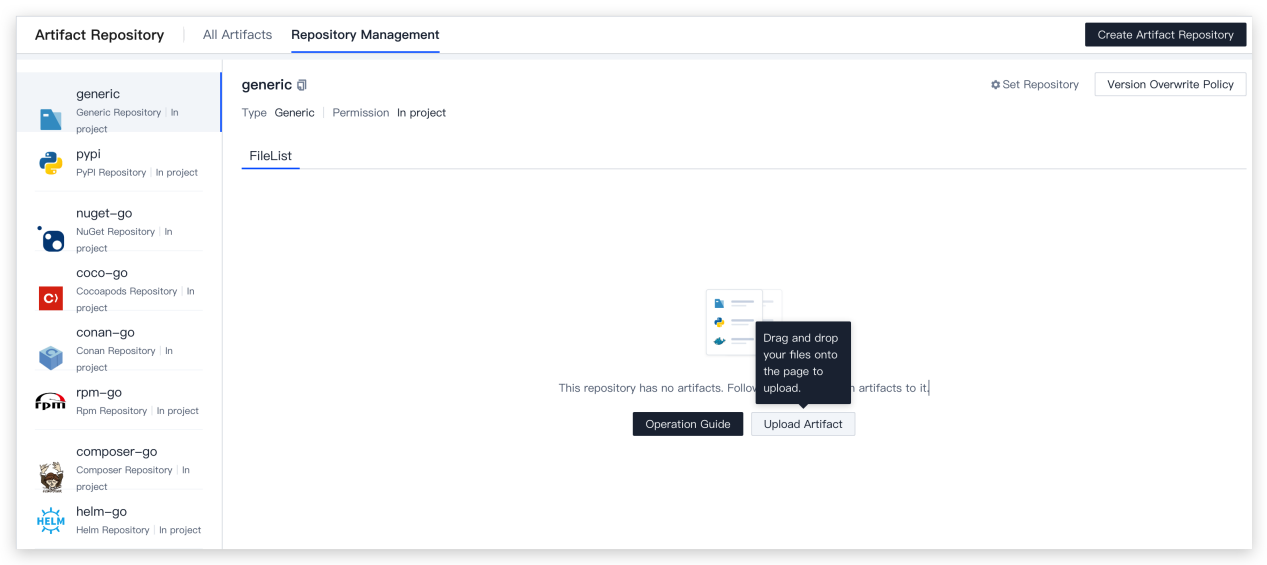


```
curl -T demo.properties -u username "https://********-generic.pkg.coding.net/my-pro
```

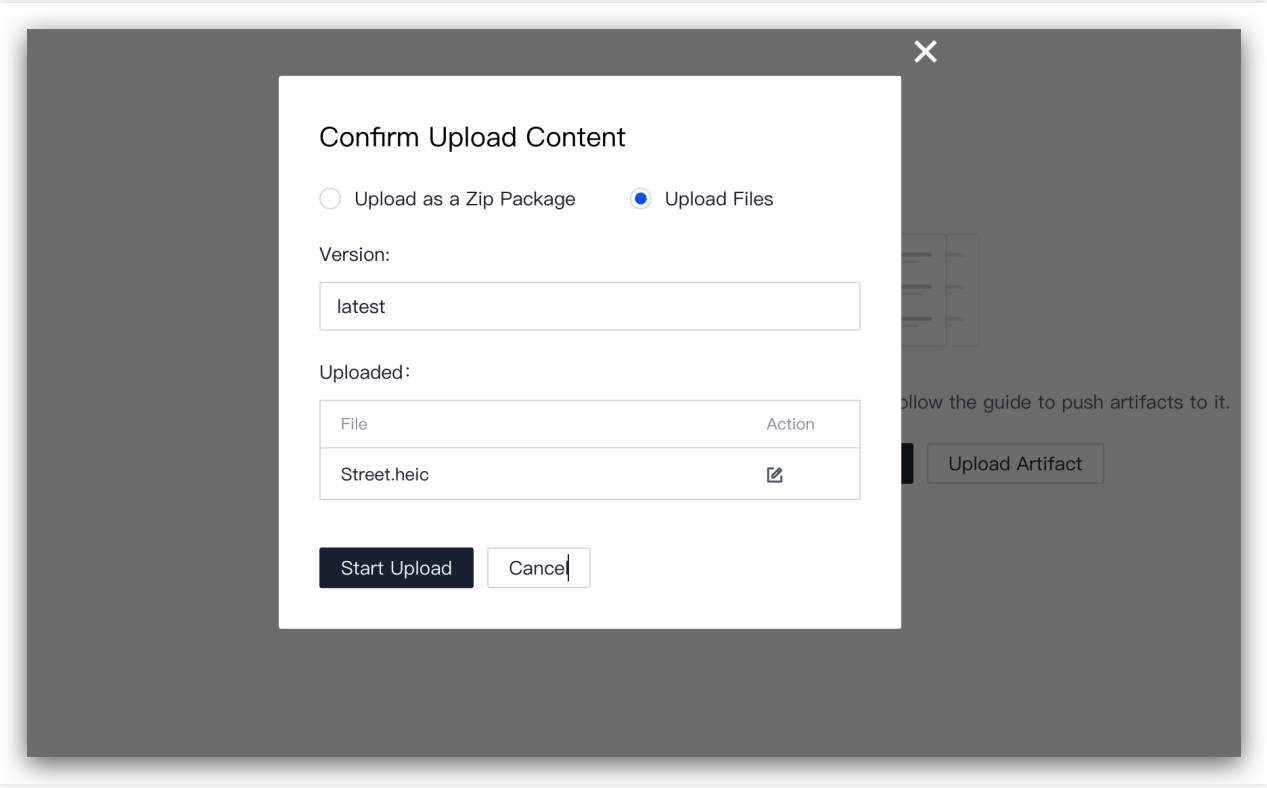
```
:Downloads $ curl -T demo.properties -u "https:/
/ -generic.pkg.coding.net/my-projects/my-generic/demo.properties?version=0.1"
Enter host password for user '':
success
```

Direct upload

On the repository page, click **Direct Upload** or drag a file onto the current page.

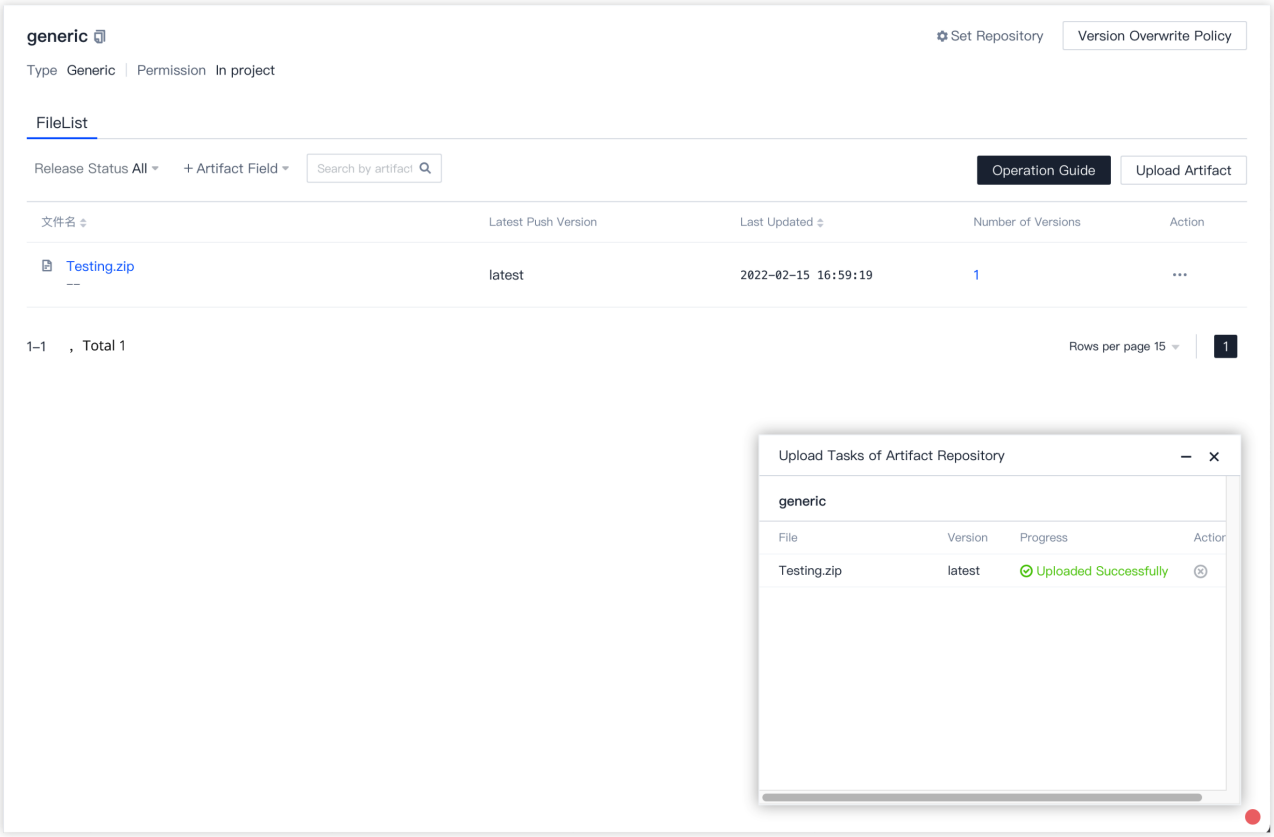


Select a file, specify the upload mode and version, confirm the package name, and click **Upload**.



View a Generic Artifact

After an artifact is uploaded successfully, a success message will be displayed in the lower right corner. Click **Package List** to view the package uploaded.



Pull a Generic Artifact

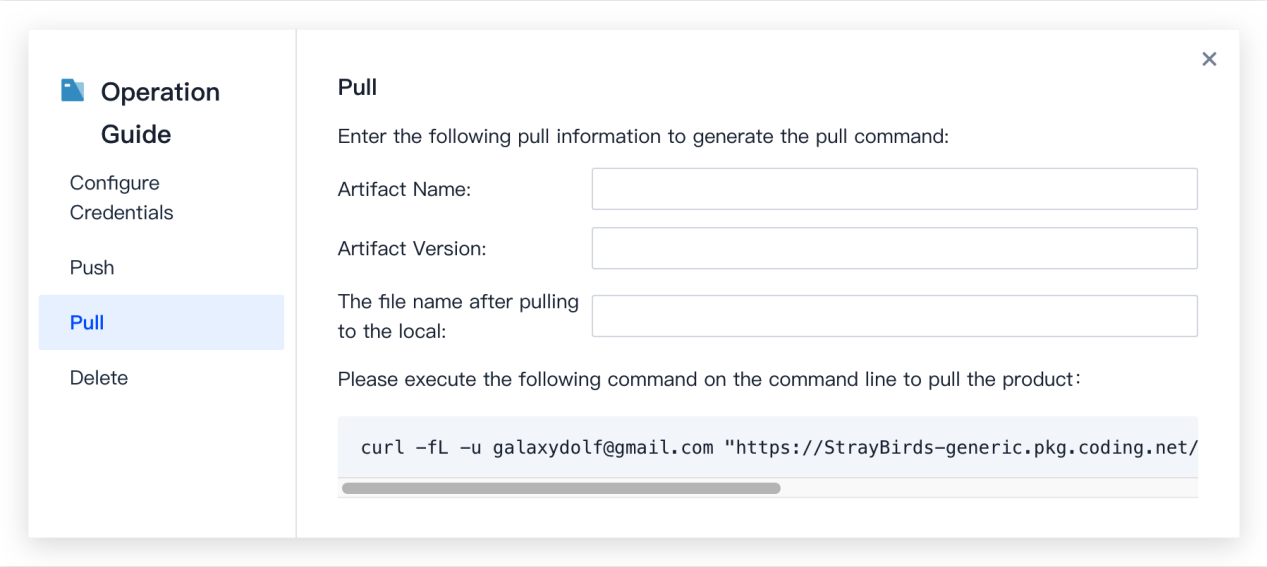
You can pull a generic artifact in either of the following ways:

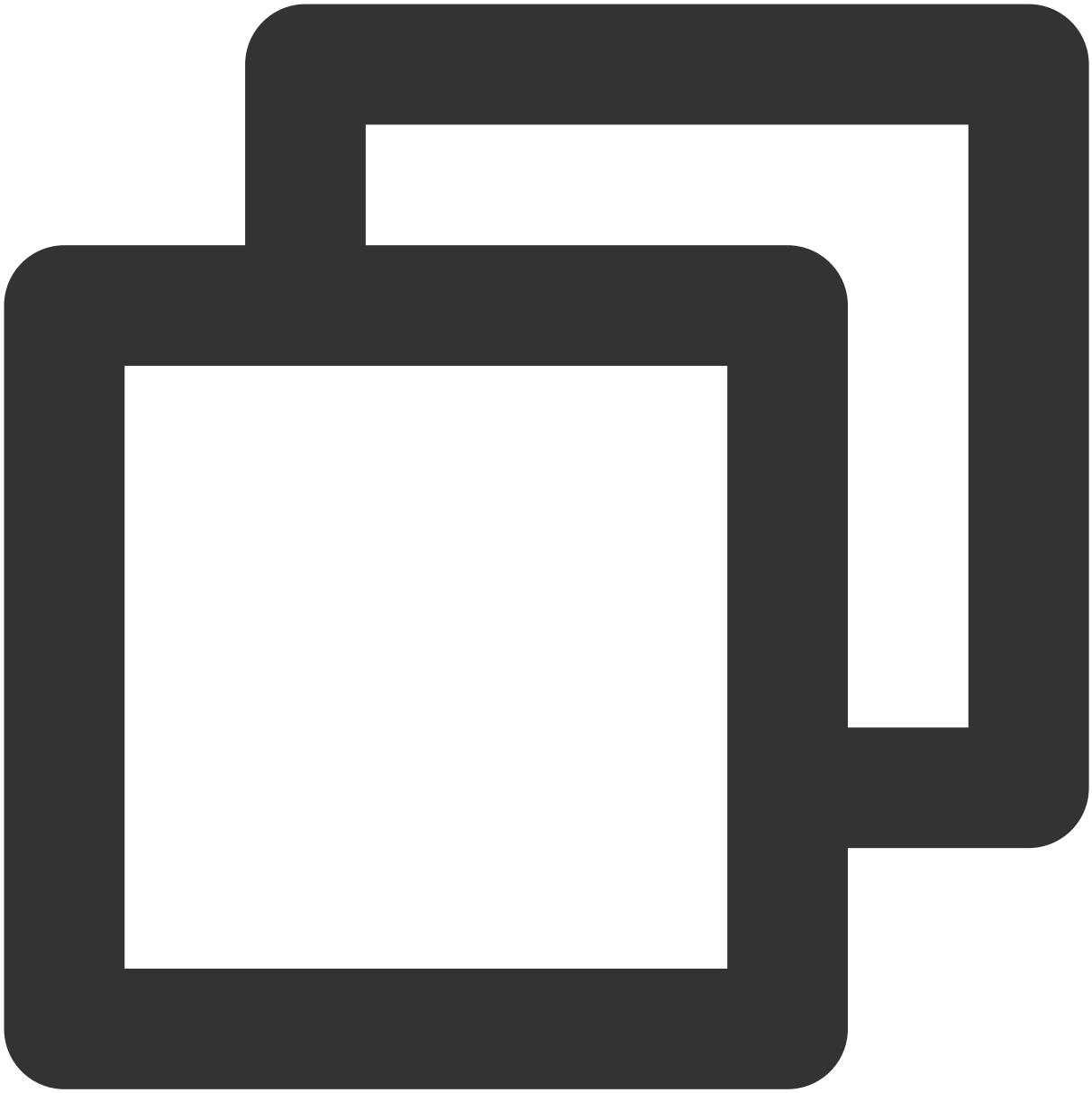
Command line

Direct download

Pull via command line

Pull an artifact using the command in the **Guide**.

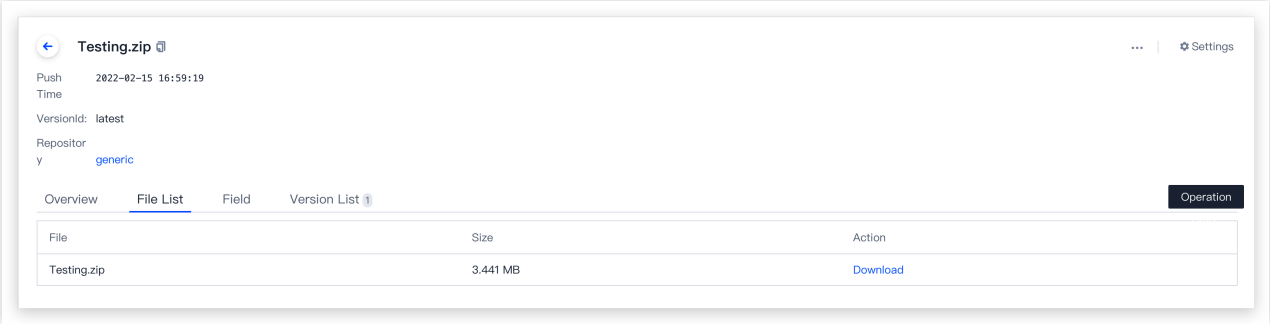




```
curl -L -u username "https://********-generic.pkg.coding.net/my-projects/my-generic
```

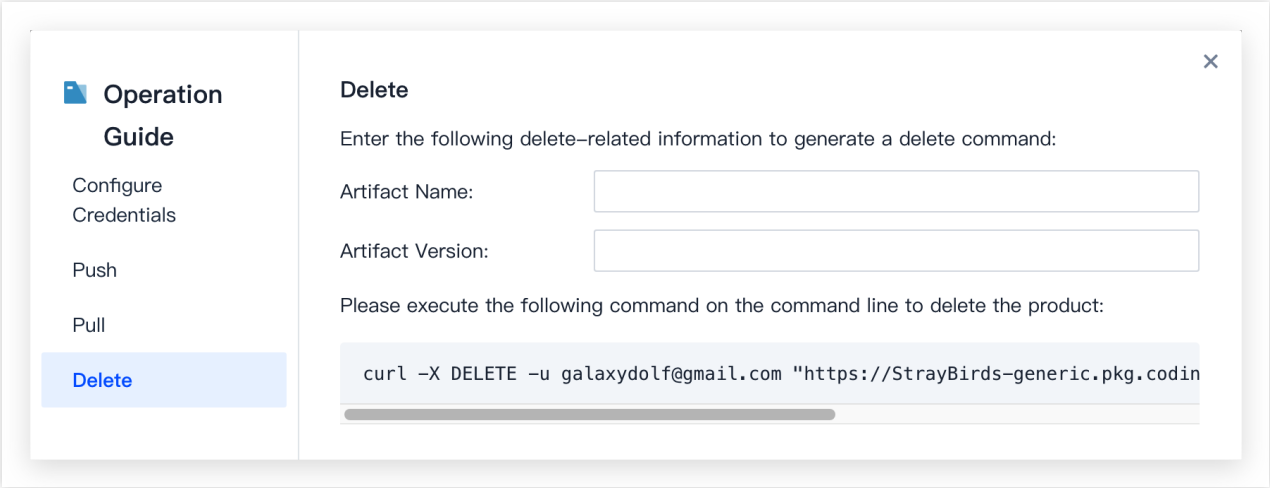
Direct download

On the repository page, select a package in **Package List** and download a version in **Version List** as needed.

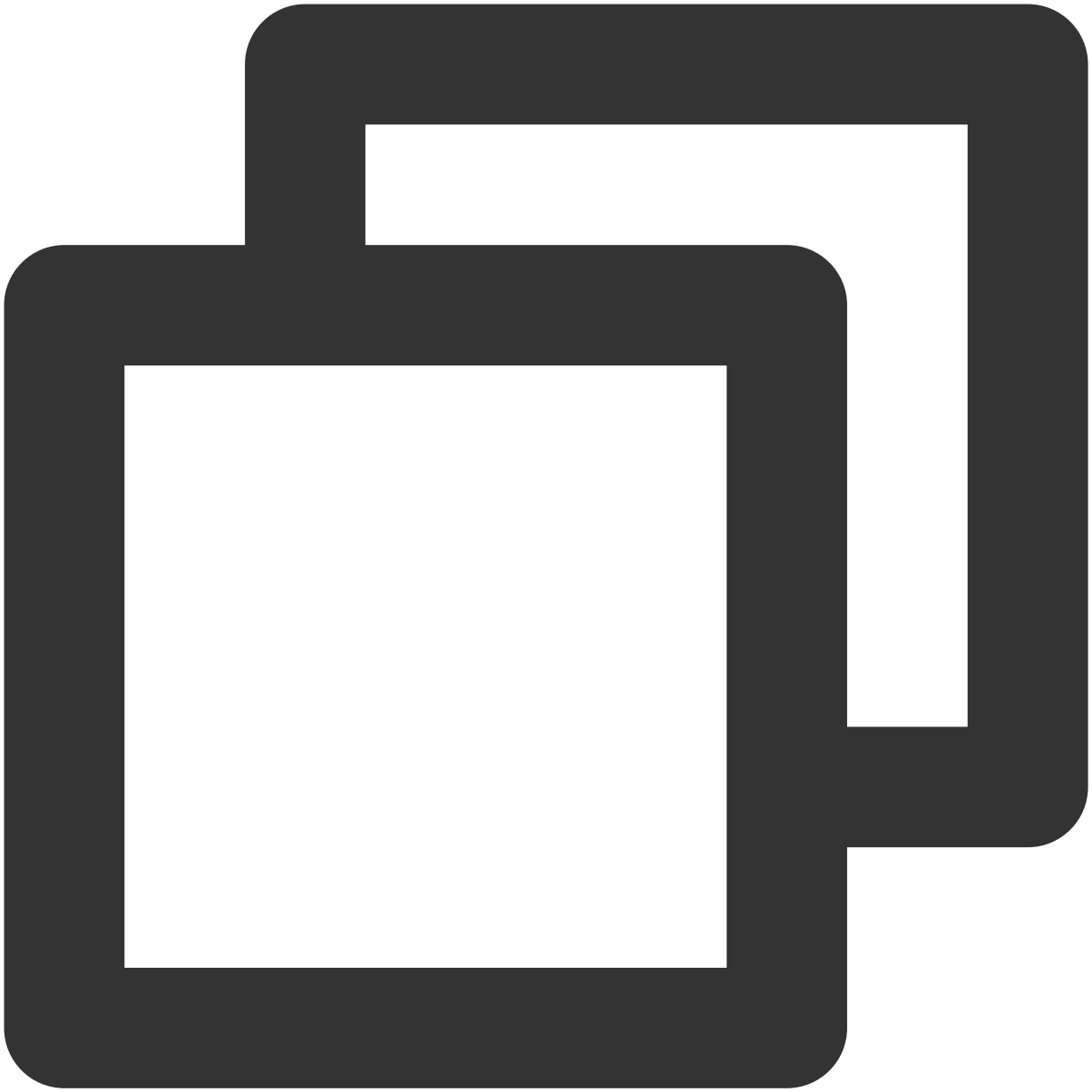


Delete a Generic Artifact

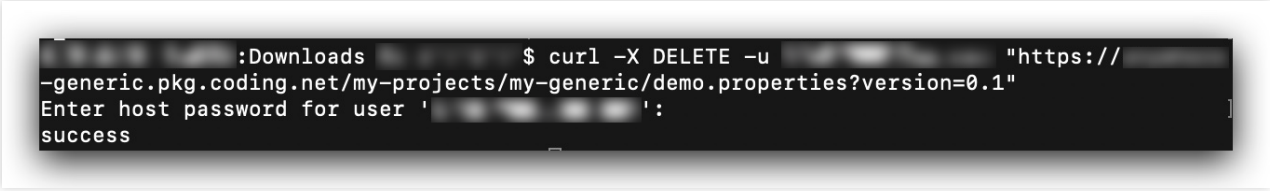
Delete a generic artifact via the command line.



Run the command in the guide.



```
curl -X DELETE -u username "https://********-generic.pkg.coding.net/my-projects/my-
```



Docker Repository

Last updated：2024-01-02 10:31:18

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

Open CODING-AR

1. Log in to the CODING Console and click **Use Now** to go to CODING page.
2. Click

in the upper-right corner to open the project list page and click a project icon to open the project.
3. In the menu on the left, click **Artifact Management**.

Preparations

Note:

Before you begin:

Install Docker.

Create an artifact repository (see Basic Operations).

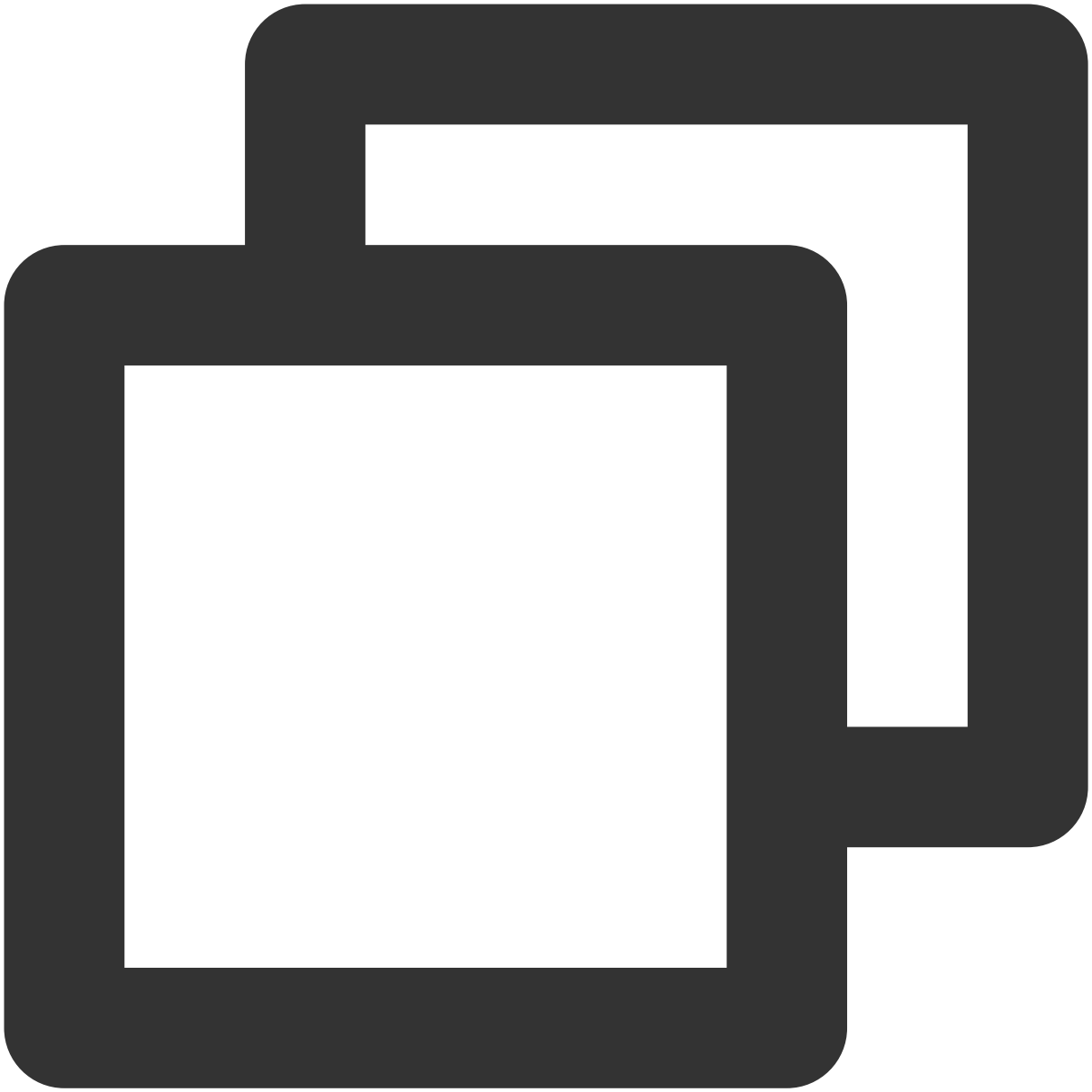
Select Docker as the repository type.

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.

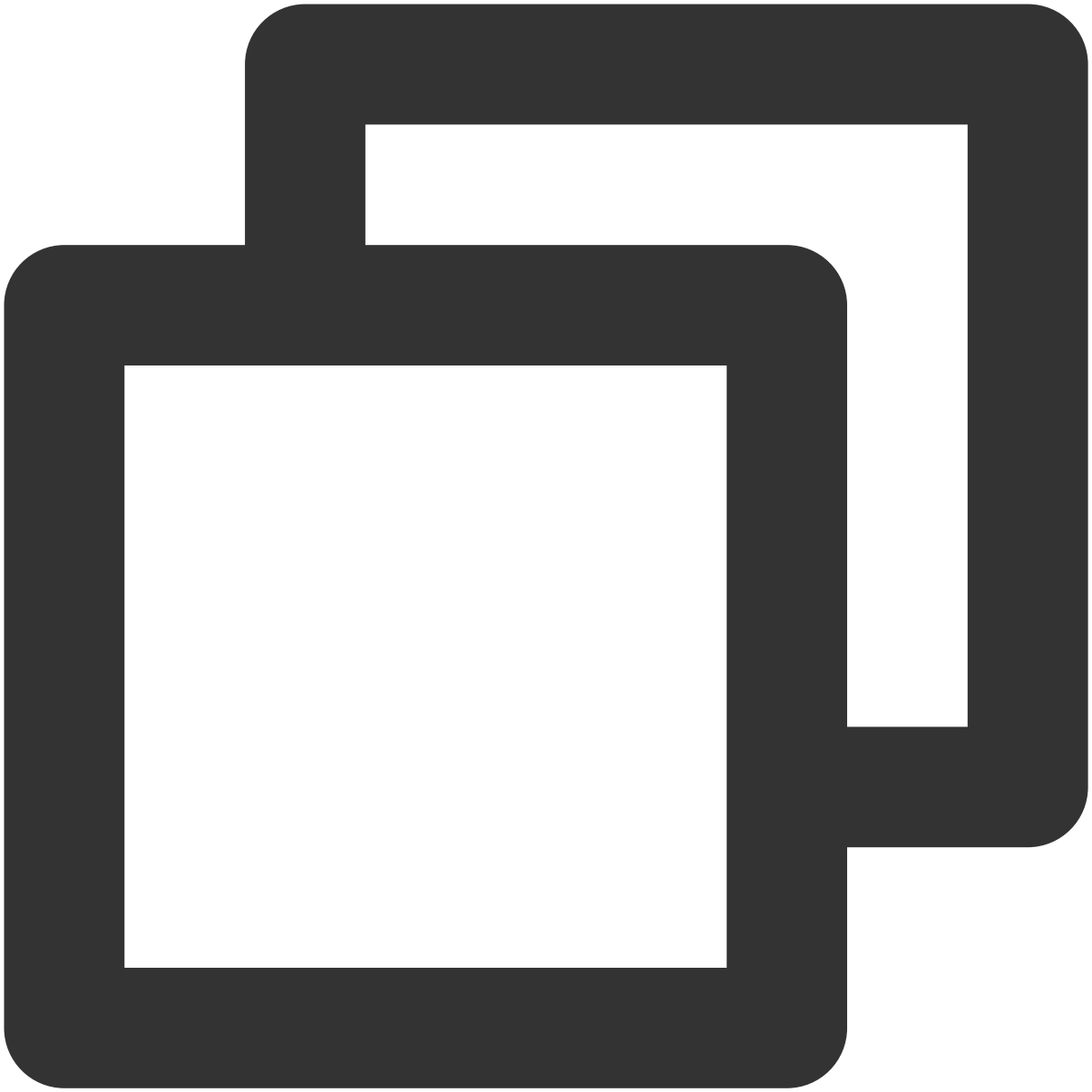
Method 1: create an image locally

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.



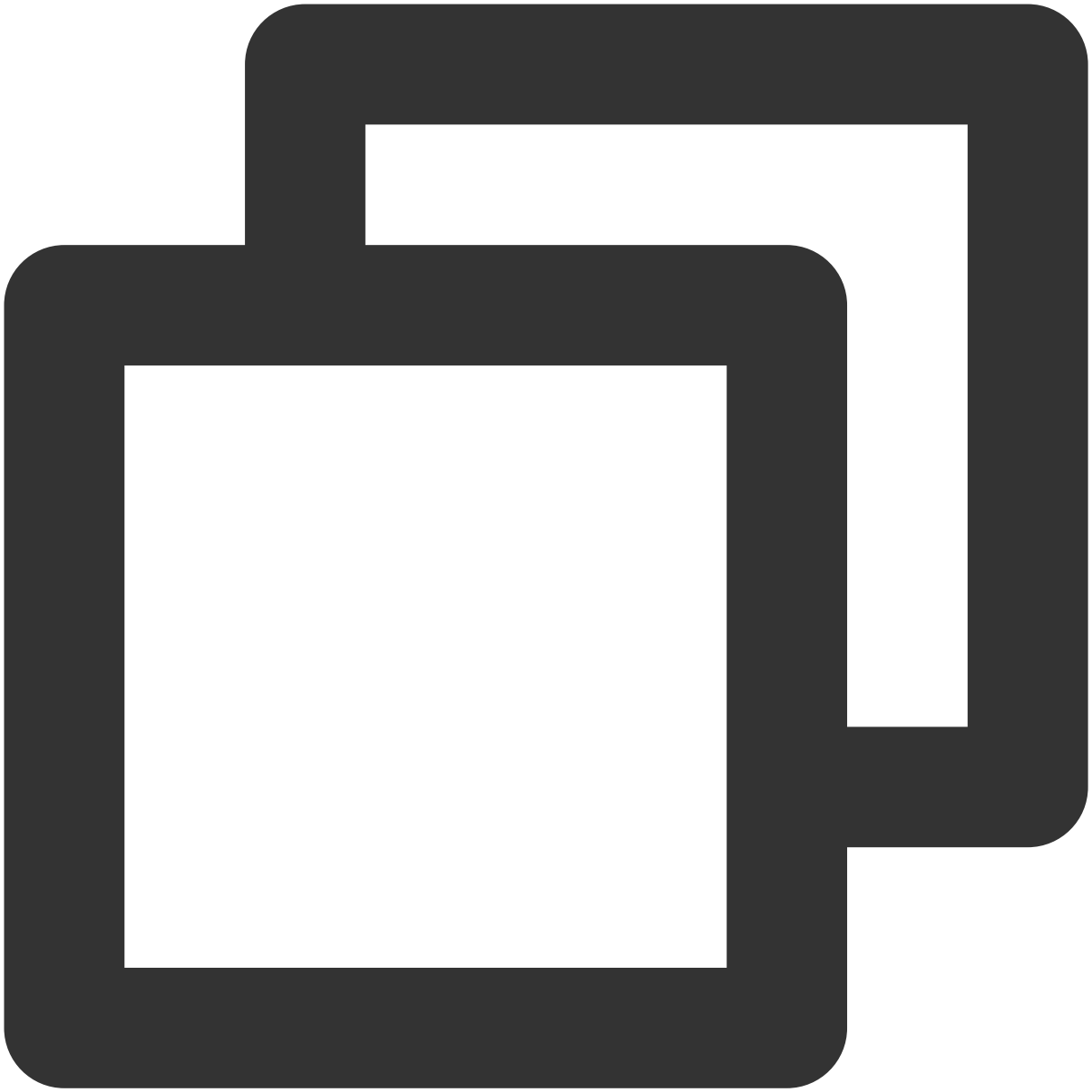
```
docker build -t hello-world .
```

The image is created with a default tag `hello-world:latest` . Refer to the Docker documentation for customized tags in the format of `<Image Name>:<Version>` .

```
/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
```

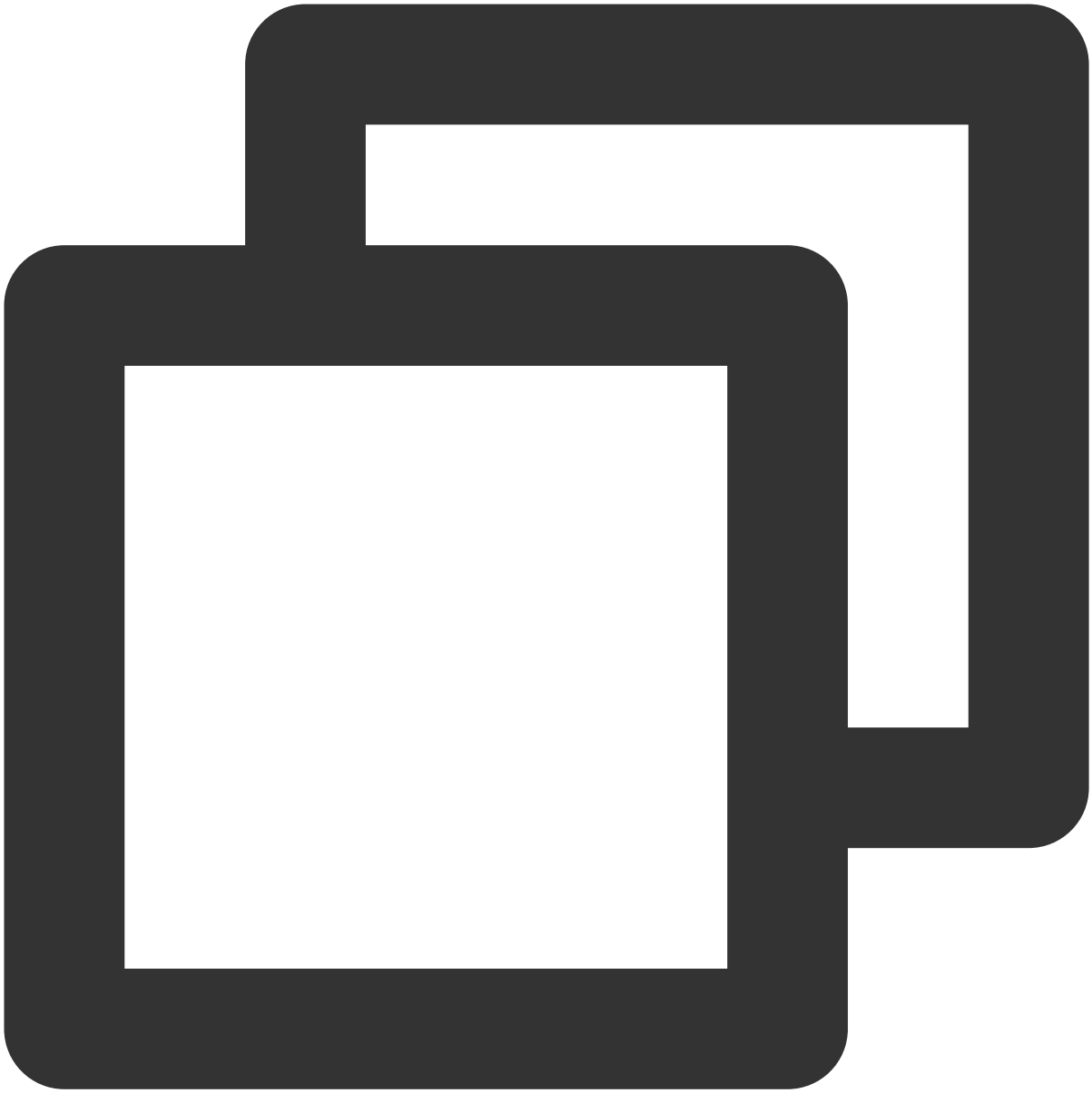
Method 2: pull an image from Docker Hub

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
```

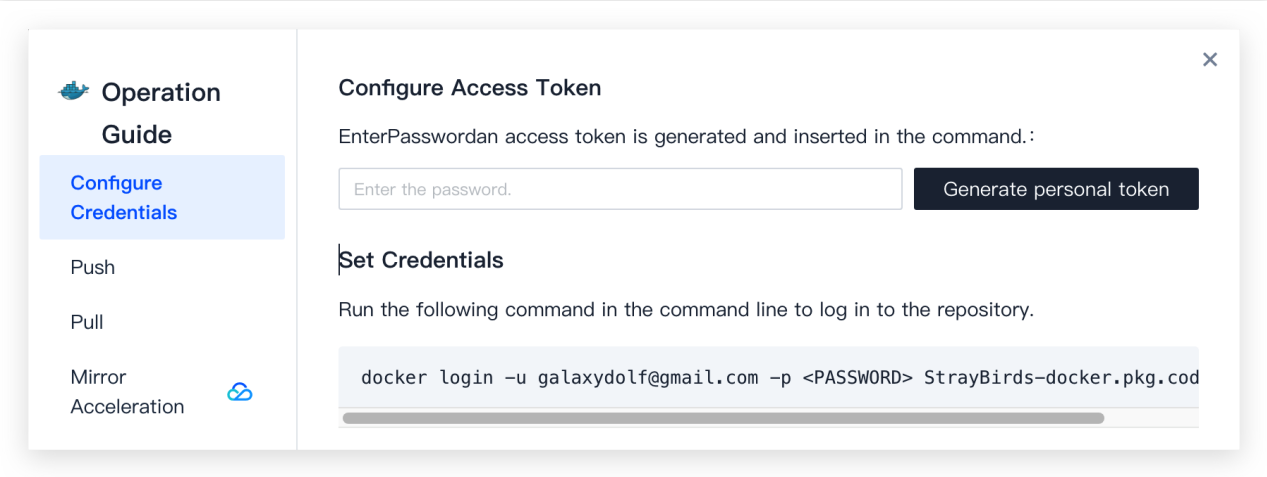
Configure Authentication Information

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.

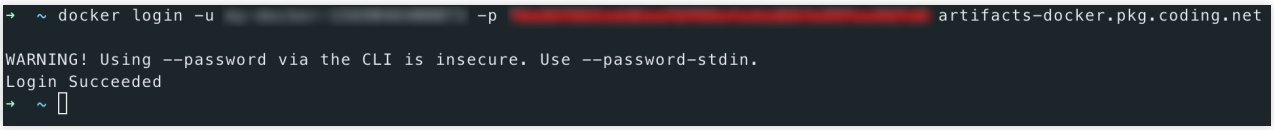
Access token

We recommend you use an **access token** to generate the authentication configuration.

1. Click **Operation Guide** on the repository page.
2. Enter the login password/two-step verification code and then copy the command generated.



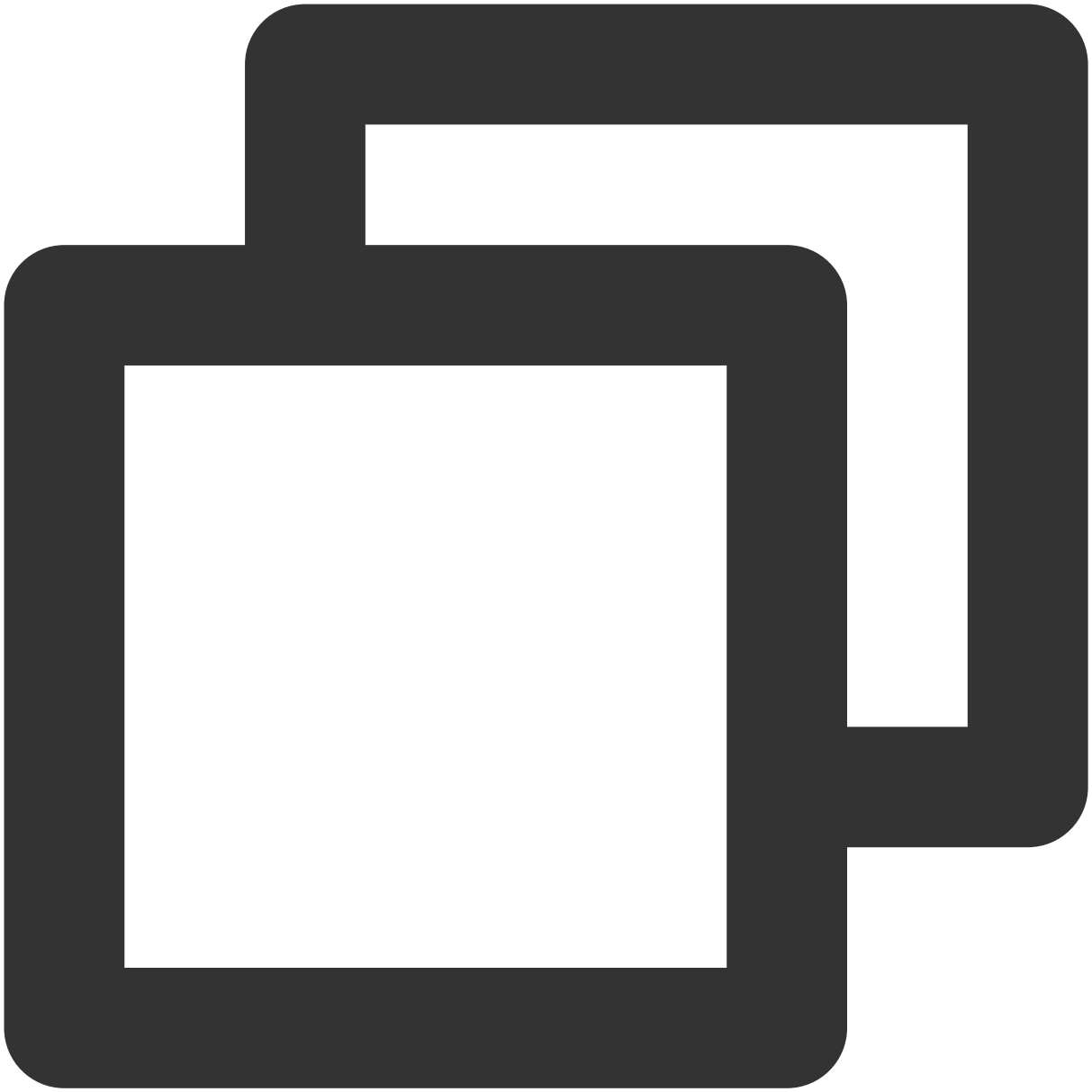
3. Paste and run the command in the local Docker environment to complete the authentication.



Push an Image

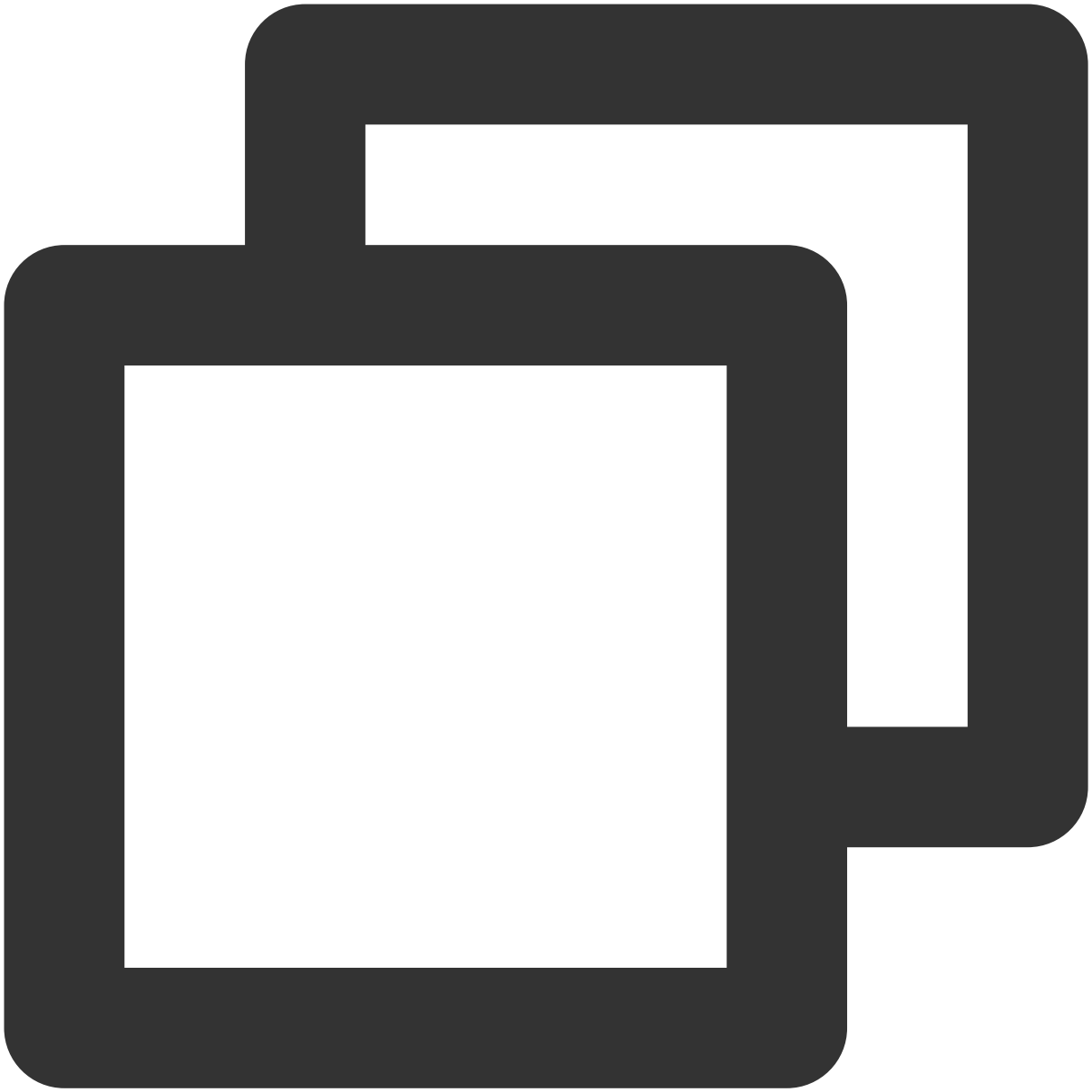
The following commands are for reference only. Use the commands generated in your project.

1. Tag the `hello-world` image pulled in the above section.



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

2. Push your docker image to CODING-AR.



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

The following information will be displayed after the image is pushed successfully.

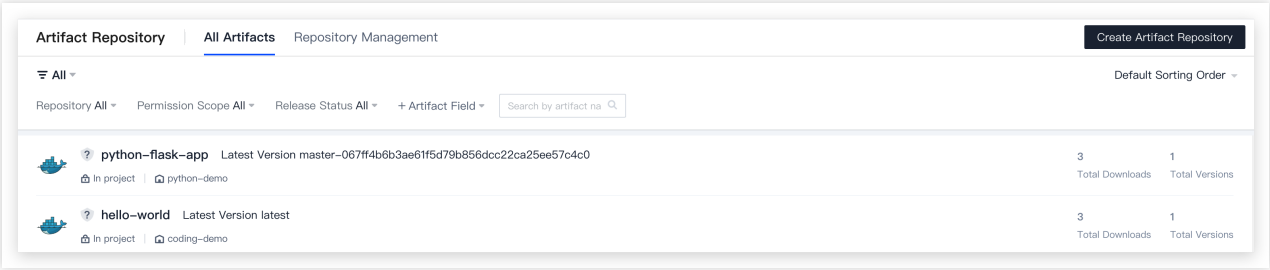
```
/Volumes/CODING/Docker-learning docker push straybirds-docker.pkg.coding.net/coding-demo/coding-demo/hello-world
The push refers to repository [straybirds-docker.pkg.coding.net/coding-demo/coding-demo/hello-world]
cc471758abdf: Pushed
a25d159becc3: Pushed
06cfd7503045: Pushed
beee9f30bc1f: Pushed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

All the commands described above are displayed in the operation guide. Replace the variables and then run the commands.

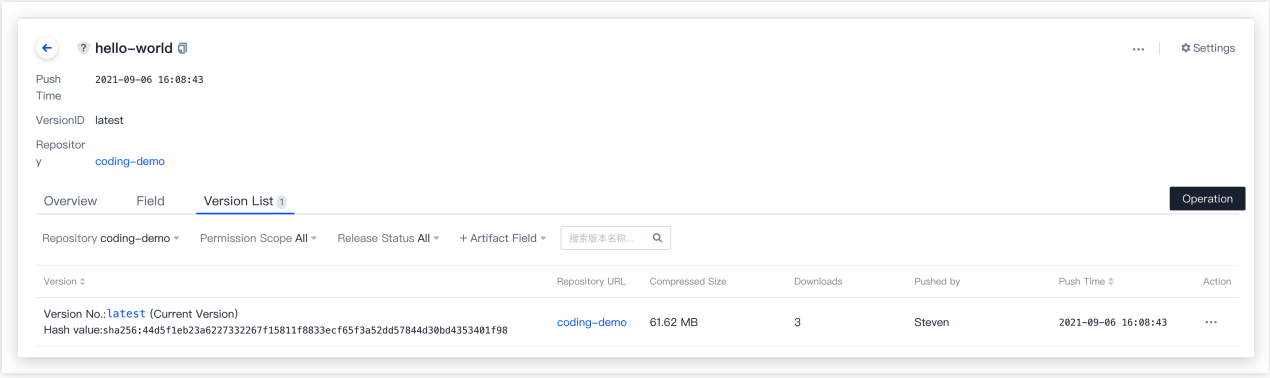
View an Image

After the image is pushed, the activity will be shown in **Project Overview**.

You can see the **hello-world** image in the artifact list.

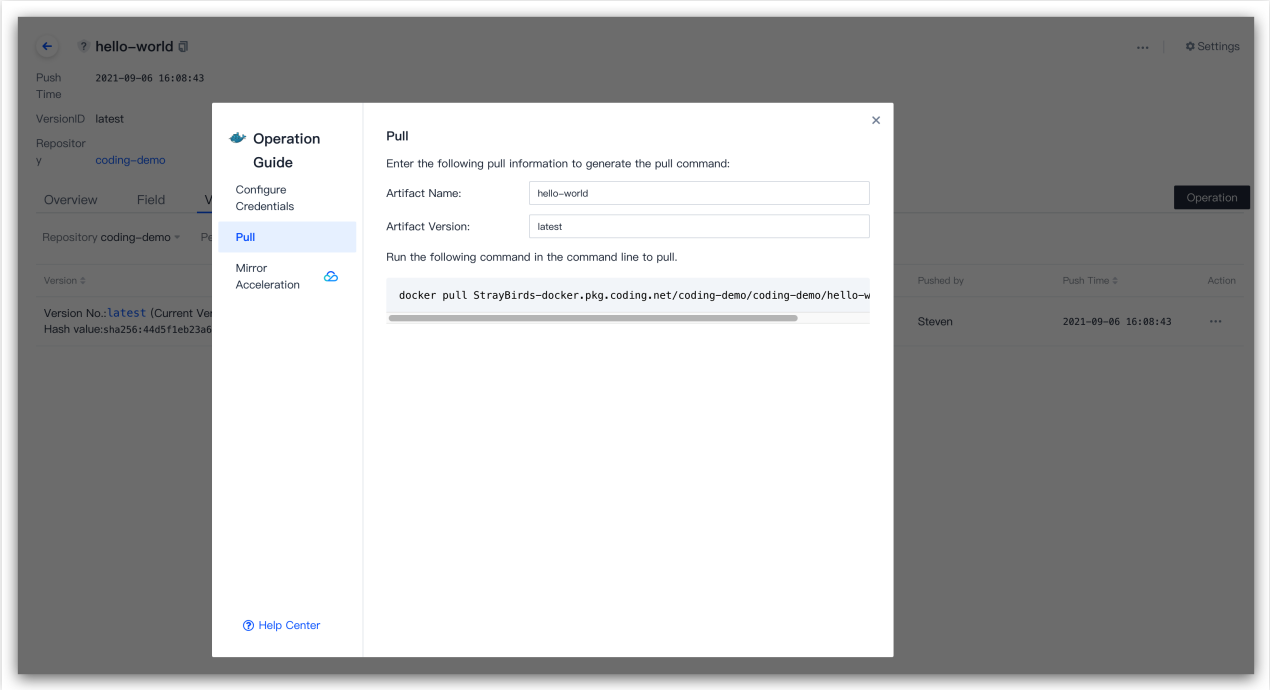


Click the image name to view its information, including overview, guide, properties, and version list.

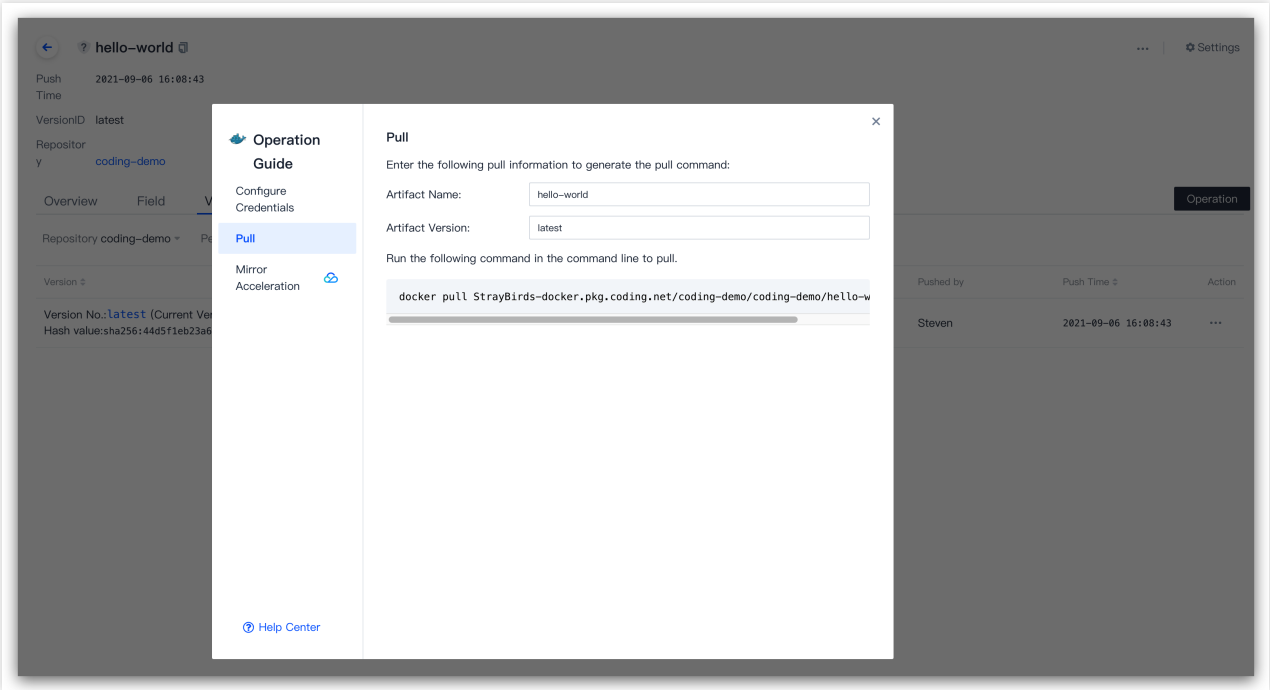


Pull an Image

Run `docker pull` to pull a Docker image from CODING-AR. You can use the command generated in the **Guide**.



The following information will be displayed after the image is pulled successfully.



npm Repository

Last updated：2024-01-02 17:38:14

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.

Open CODING-AR

1. Log in to the CODING Console and click **Use Now** to go to CODING page.
2. Click

in the upper-right corner to open the project list page and click a project icon to open the project.
3. In the menu on the left, click **Artifact Management**.

Preparations

Note:

Before you begin:

Install Node.js.

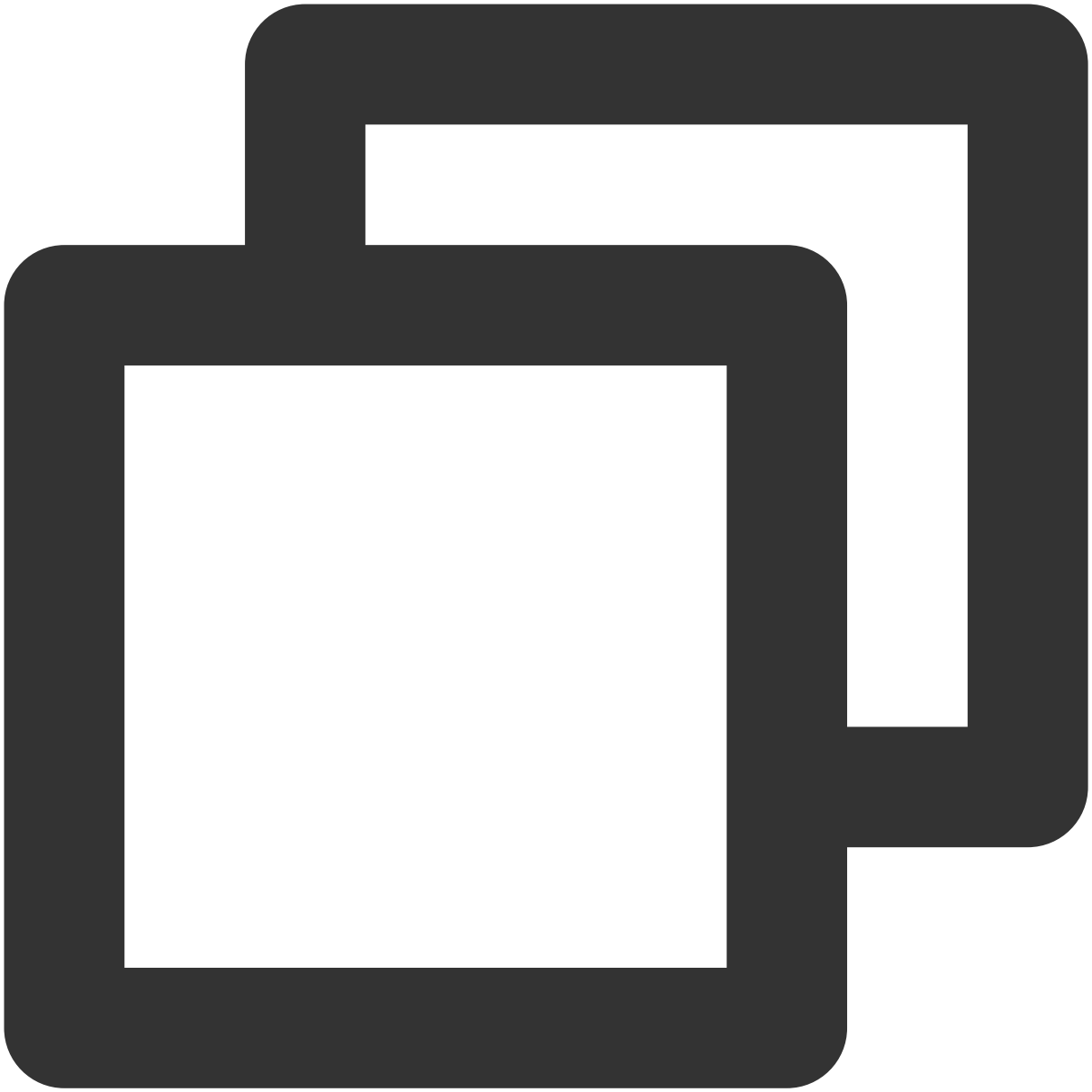
Create an artifact repository (see Basic Operations).

Select npm as the repository type.

Initialize a Local npm Project (Optional)

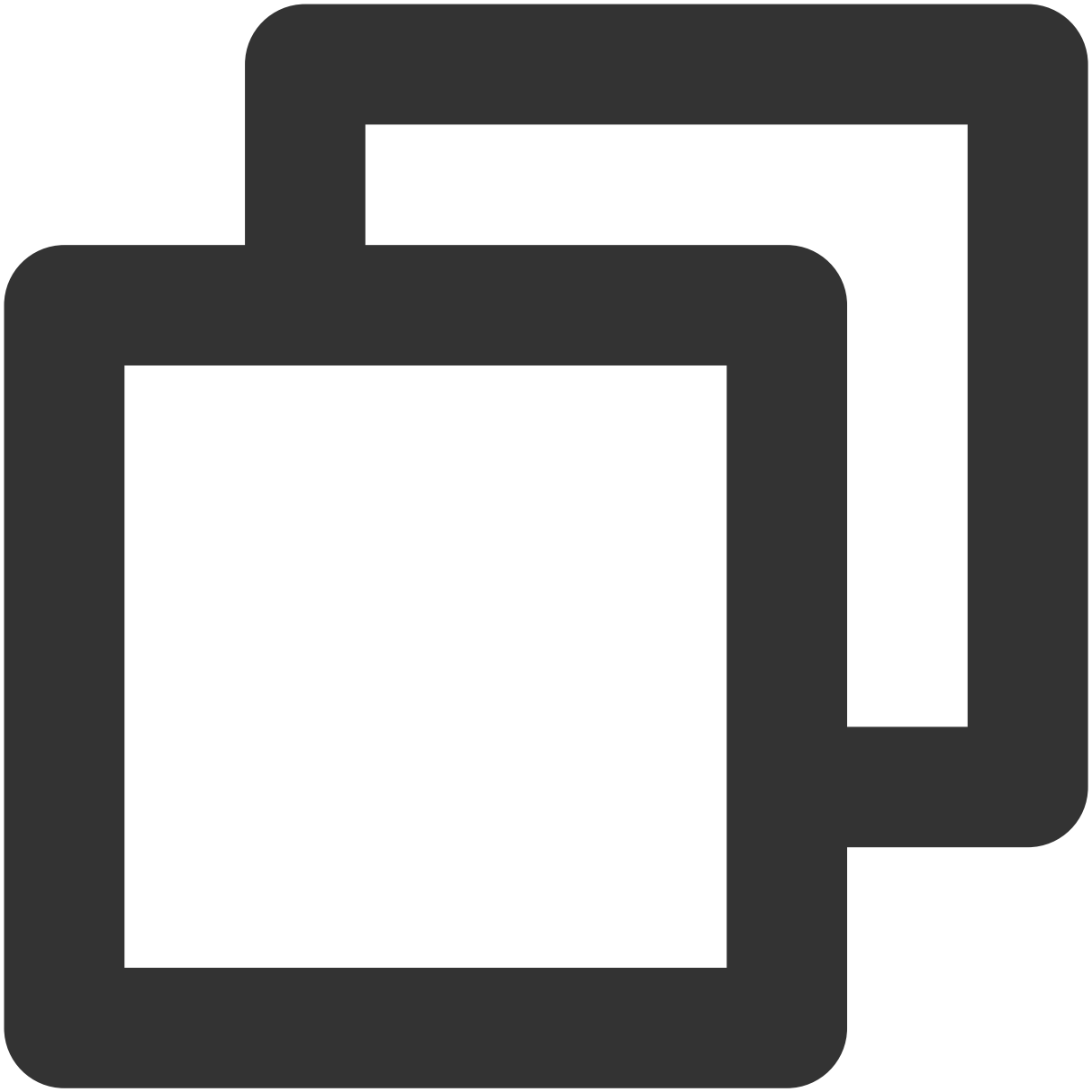
You can skip this section if you are familiar with npm artifacts.

1. Create a demo directory for the npm project.



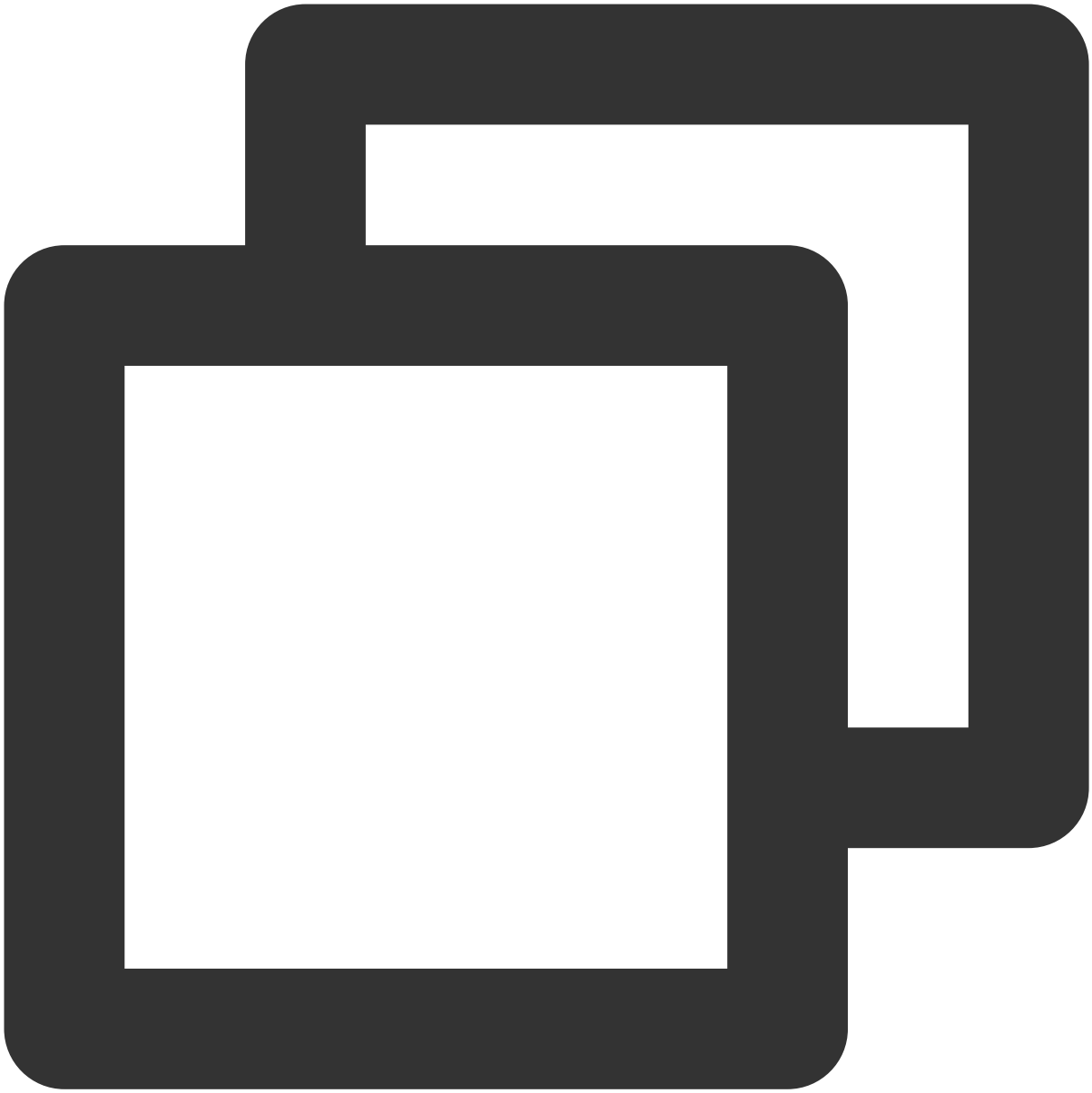
```
mkdir npm-demo
```

2. Initialize the npm project.



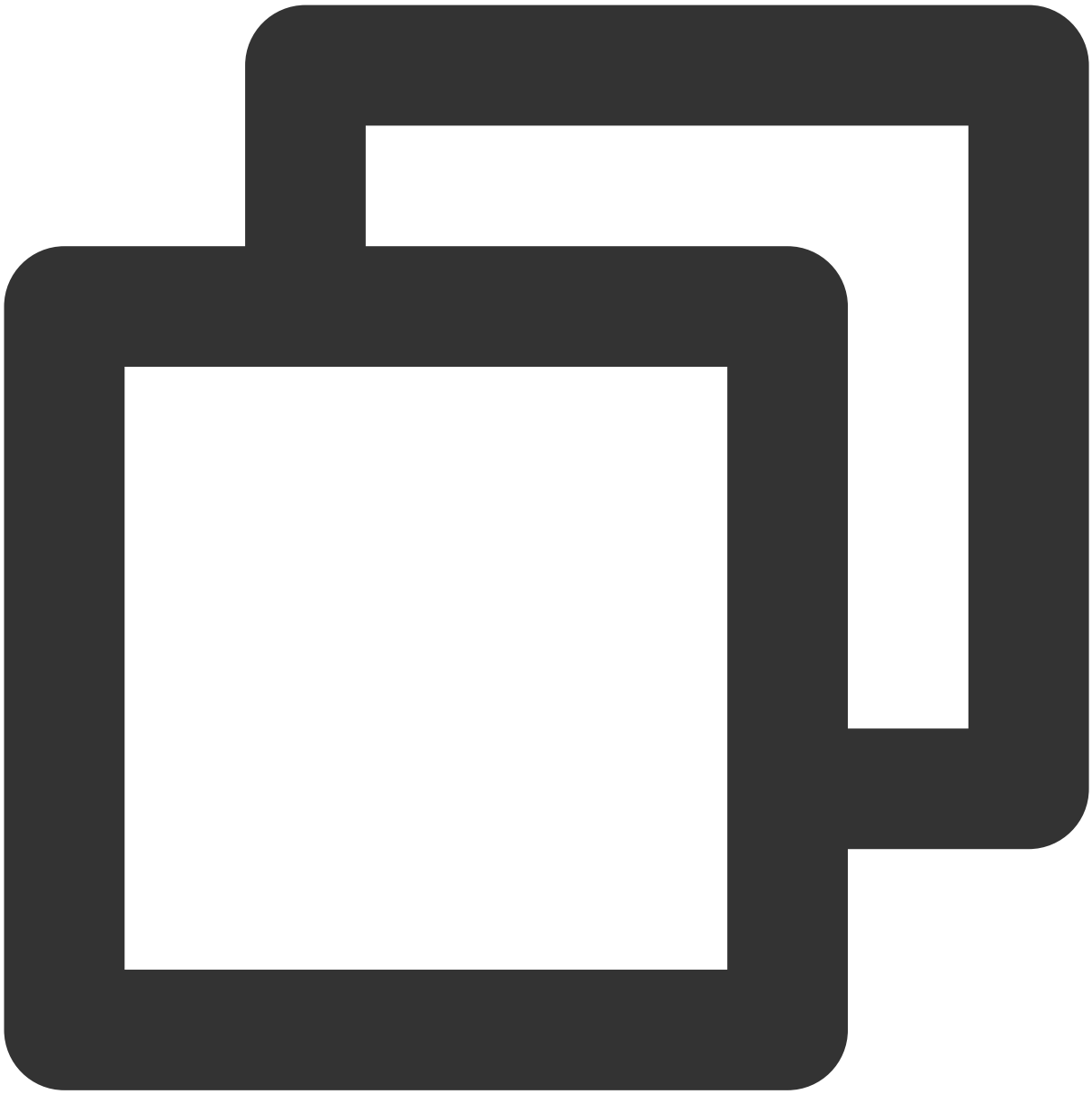
```
cd npm-demo && npm init
```

Enter the npm configuration in the new `package.json` when asked. 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

Authentication information must be configured before you can pull artifacts from or push artifacts to CODING-AR.

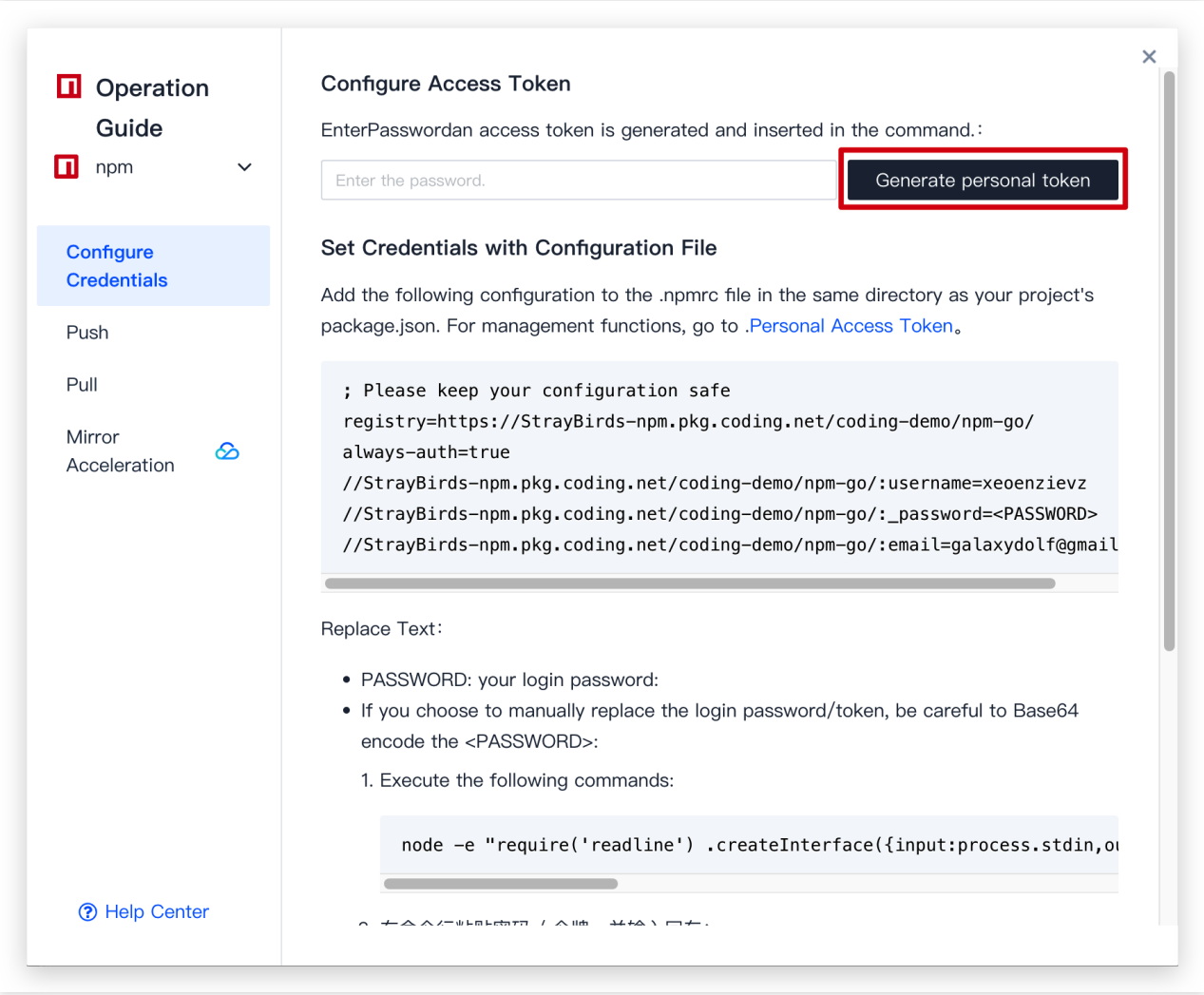
Configure authentication information in either of the following ways:

Set credentials with the configuration file

Set credentials with the interactive command line

Method 1: setting credentials with configuration file

1. On the guide page, click **Generate configuration from access token** and enter the account login password in the pop-up window.



2. Copy the configuration to the `.npmrc` file in the same directory as your project's `package.json` .

Set Credentials with Configuration File

Add the following configuration to the .npmrc file in the same directory as your project's package.json. For management functions, go to .Personal Access Token。

```
; Please keep your configuration safe
registry=https://StrayBirds-npm.pkg.coding.net/coding-demo/npm-go/
always-auth=true
//StrayBirds-npm.pkg.coding.net/coding-demo/npm-go/:username=npm-go-164497
//StrayBirds-npm.pkg.coding.net/coding-demo/npm-go/:_password=ZDdmY2M0ZWQ5
//StrayBirds-npm.pkg.coding.net/coding-demo/npm-go/:email=galaxydolf@gmail
```

Replace Text:

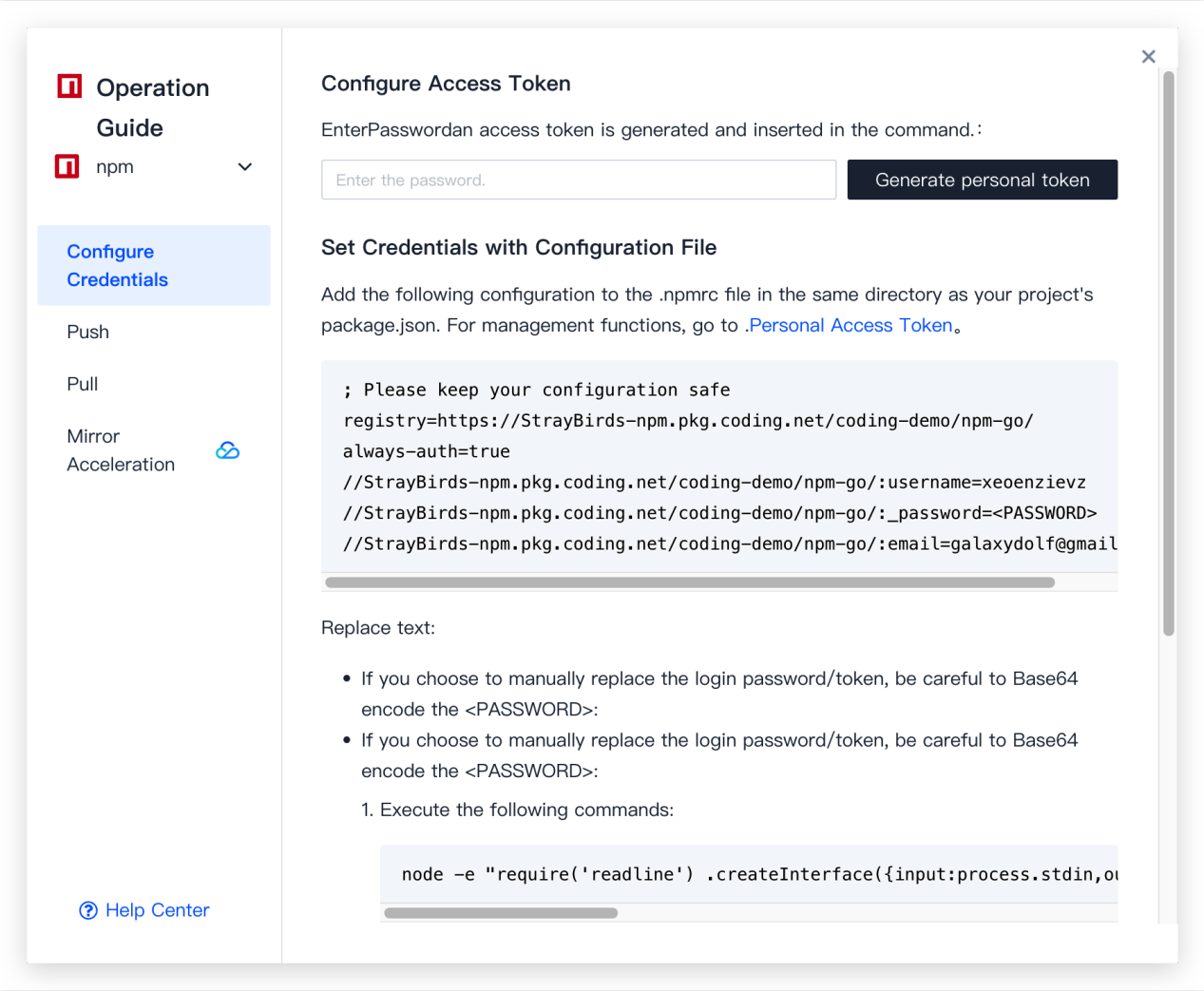
- PASSWORD: Your login password
- If you choose to manually replace the login password/token, be careful to Base64 encode the <PASSWORD>:

  1. Execute the following commands:

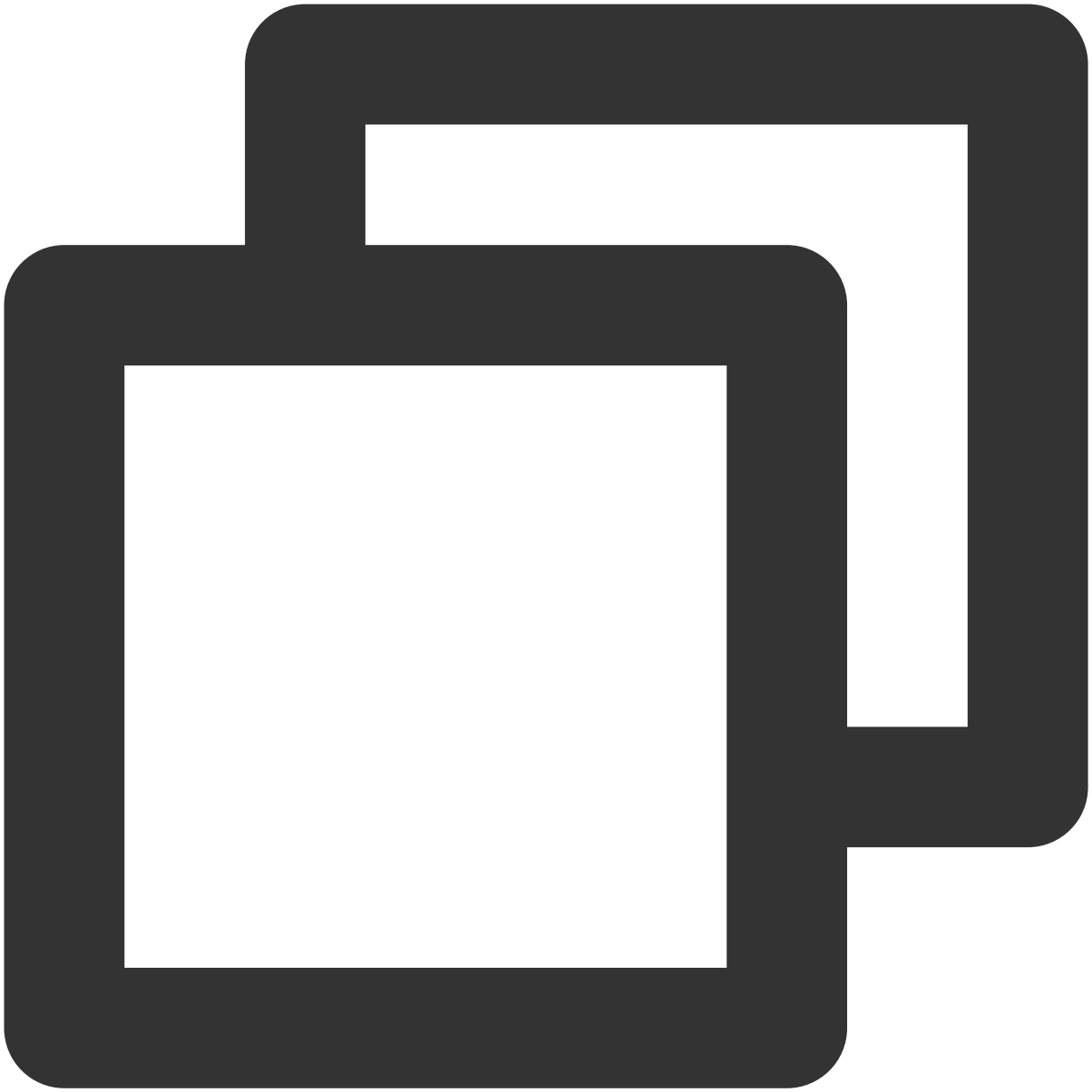
```
node -e "require('readline') .createInterface({input:process.stdin,ou
```

Method 2: set credentials with interactive command line

1. Copy and run the `npm config` to set the `npm registry` to the current artifact repository.



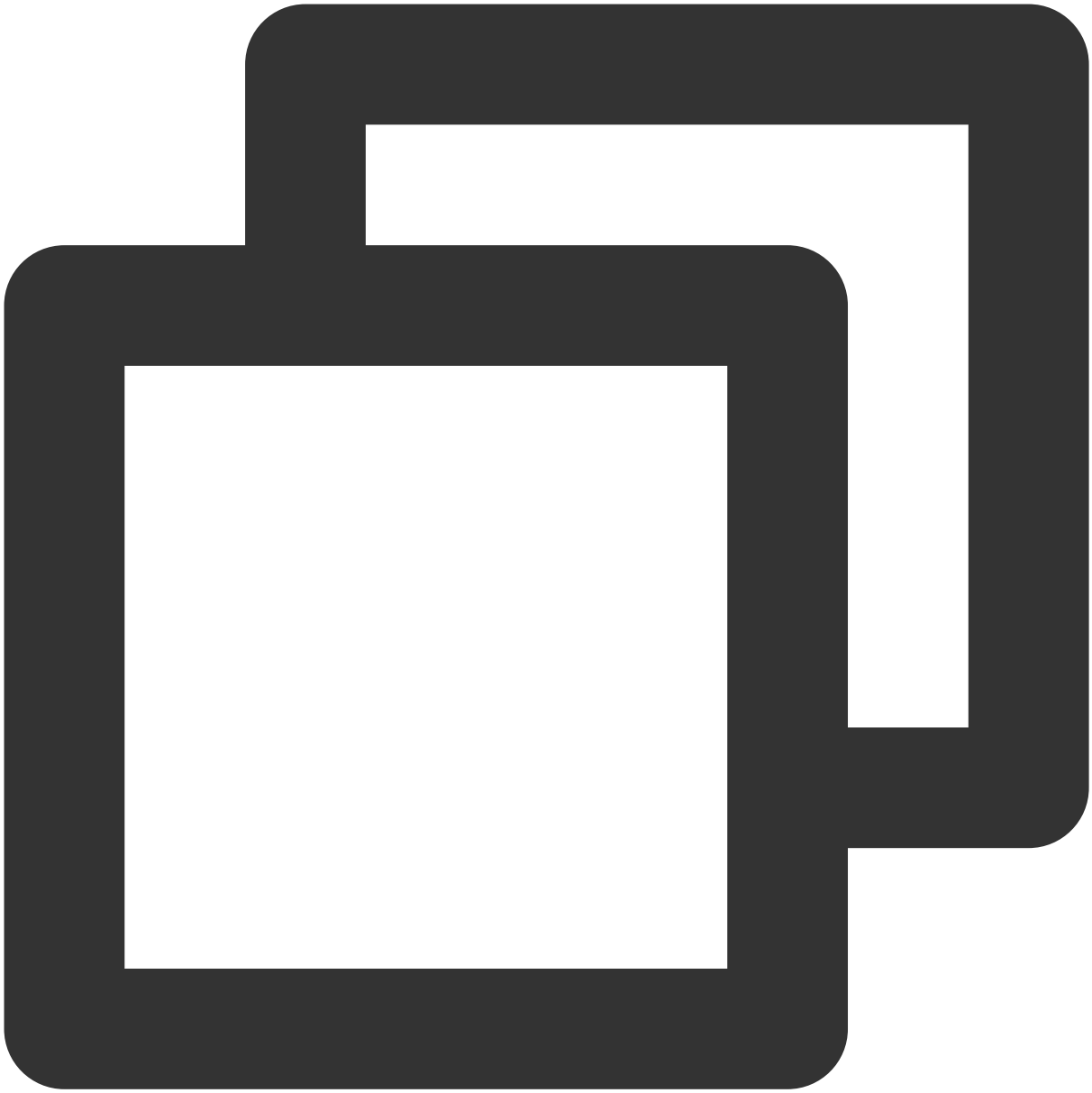
2. Run `npm login` and enter the account, password, and email address when asked.



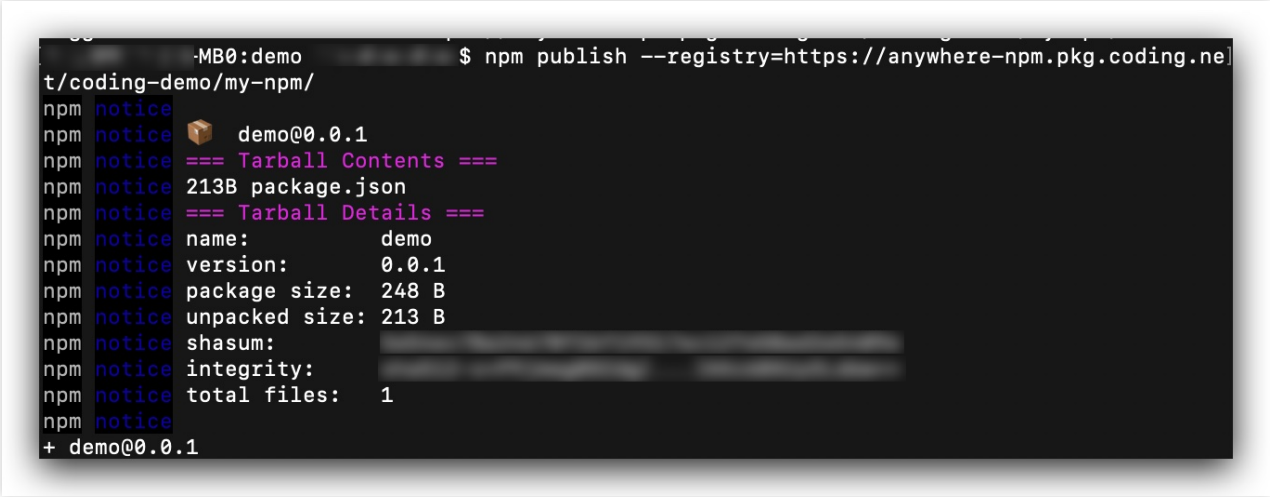
```
npm login
```

Push an npm Artifact

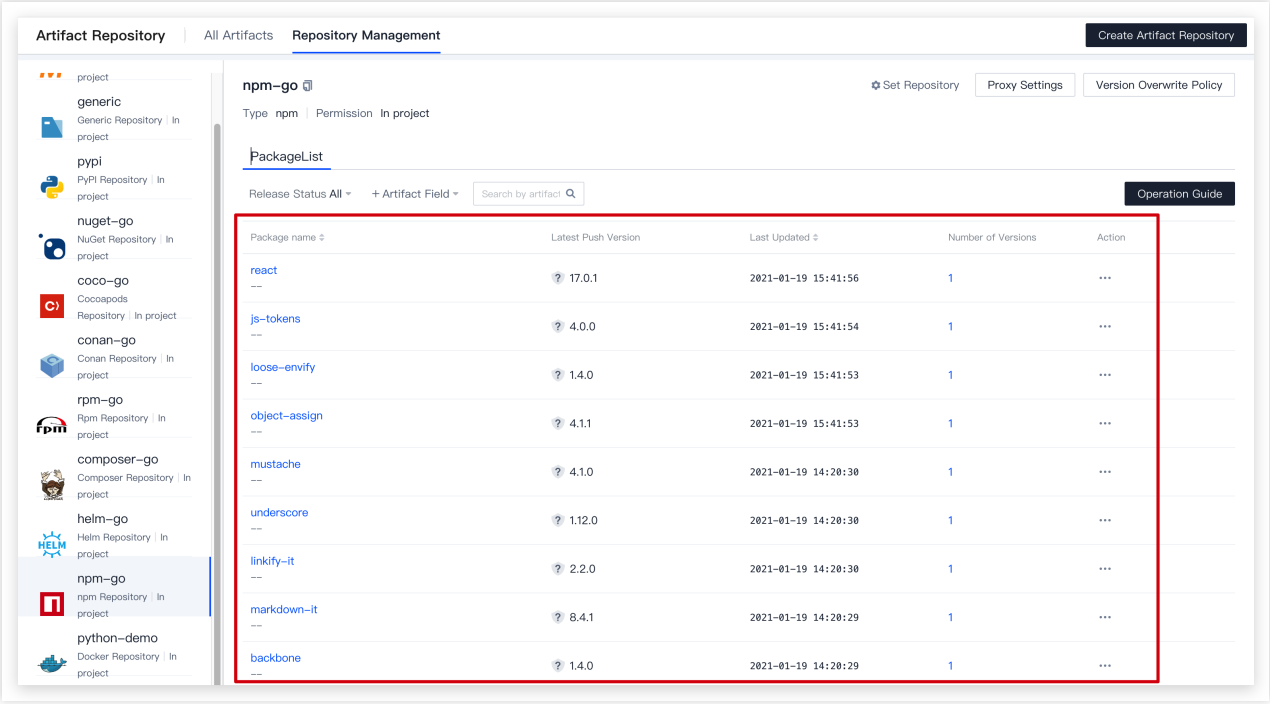
Copy and run the **push** command on the page to push a local artifact to the remote repository.



```
npm publish --registry=<Repository URL in the Push Guide>
```

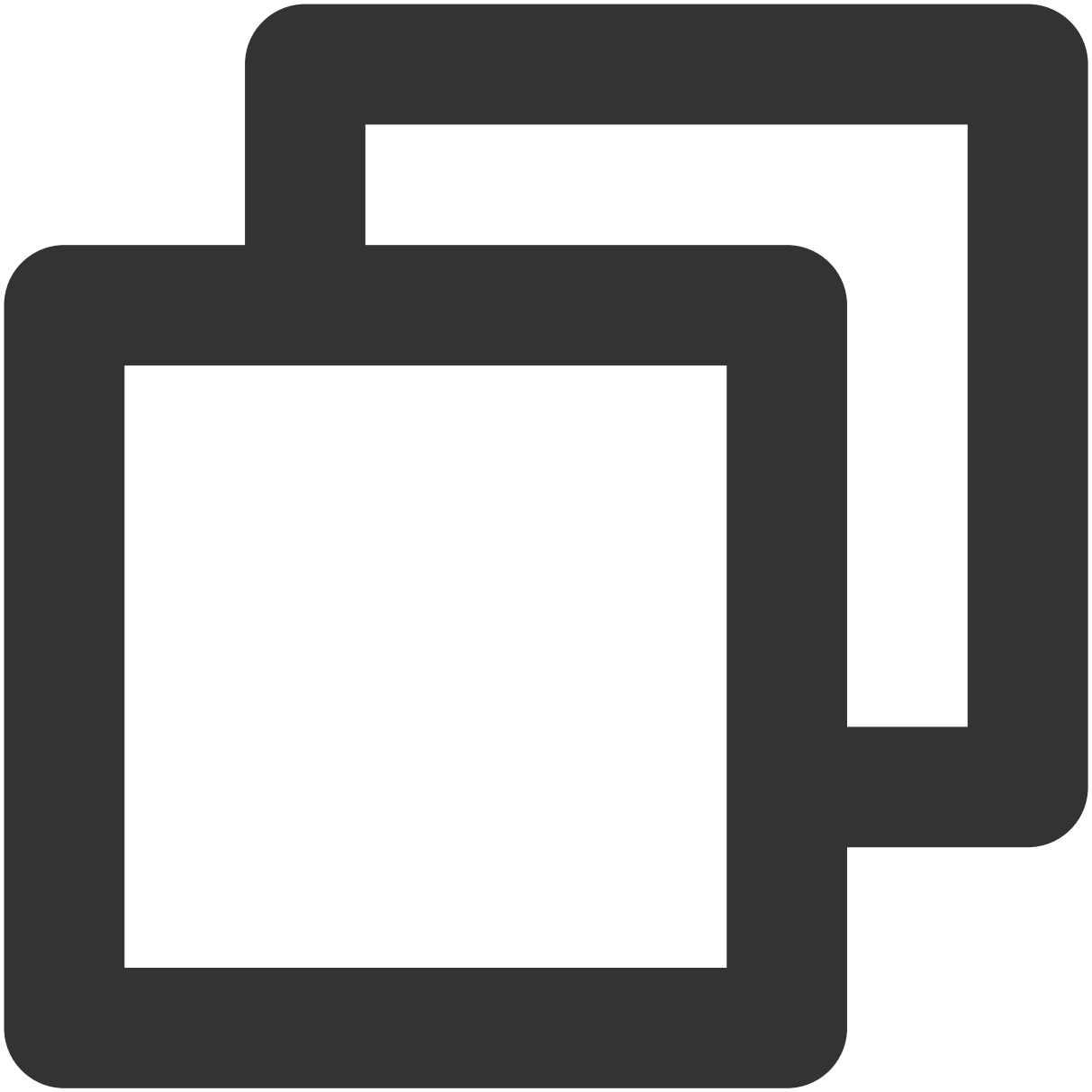


After the artifact is pushed successfully, refresh the page to view the latest artifacts.



Pull an npm Artifact

Copy and run the `npm install` command to pull an artifact:



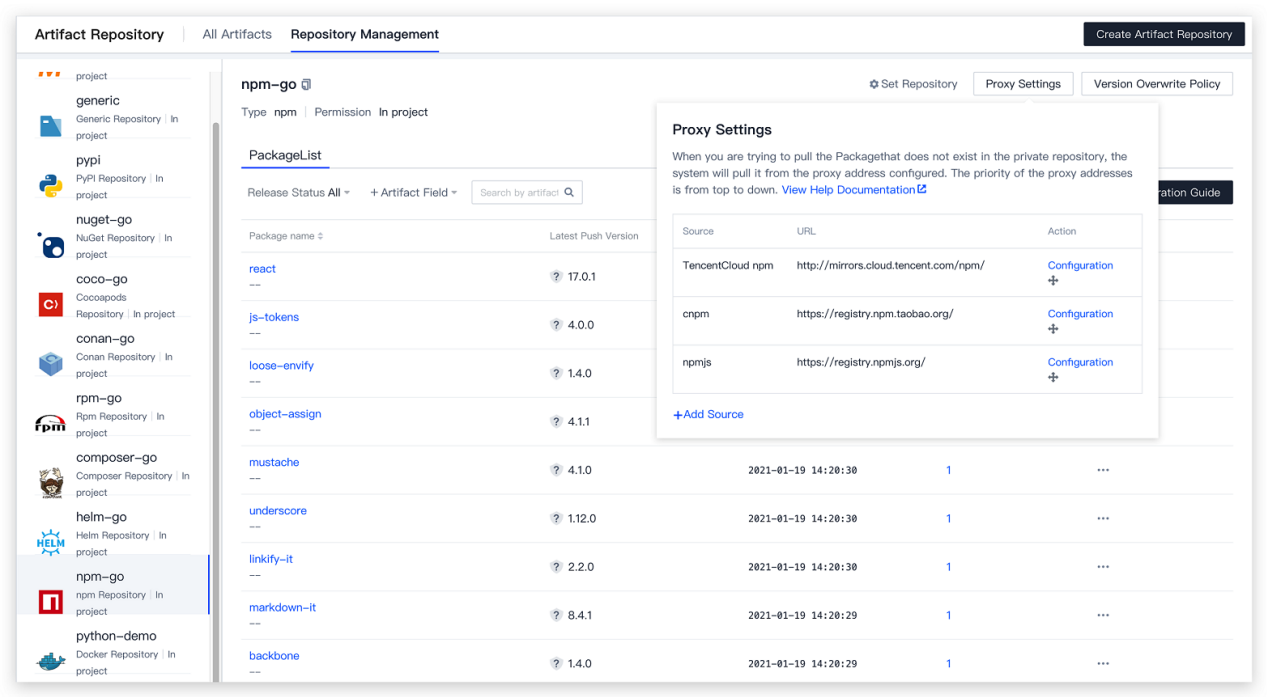
```
npm install <Package Name> --registry=<Repository URL in the Pull Guide>
```

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
```

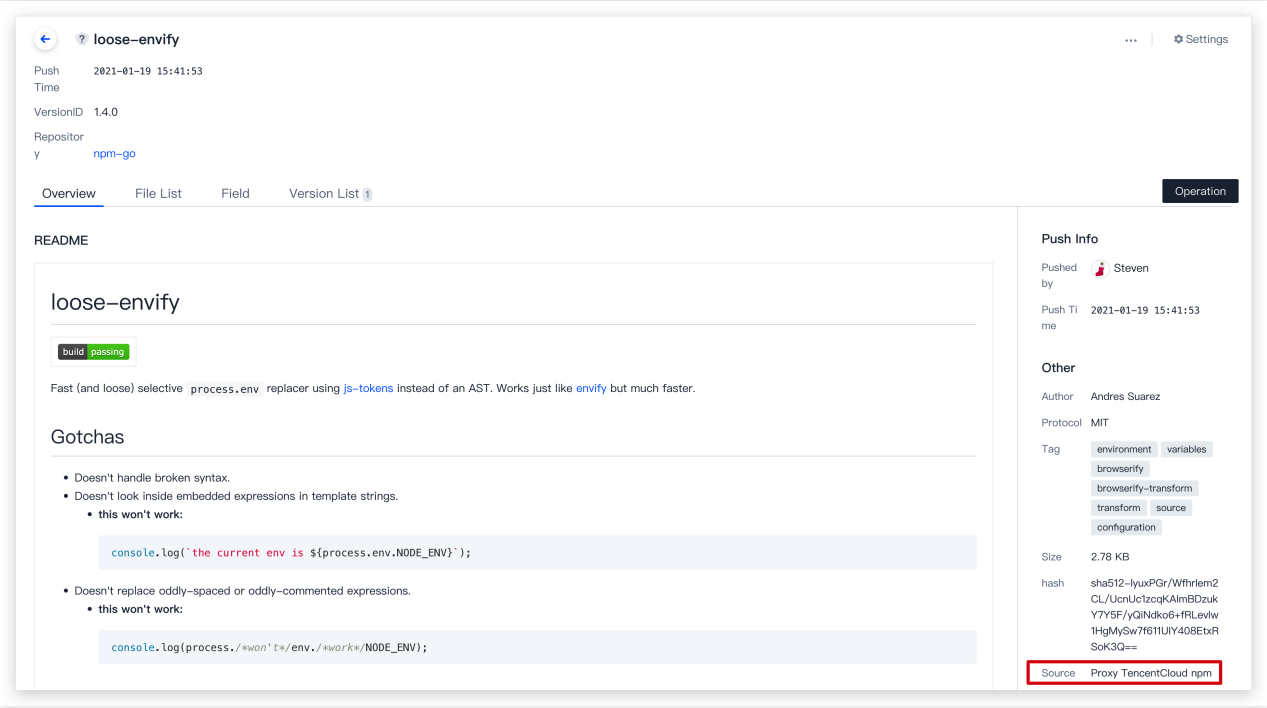
Configure a Proxy

If 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.



Replace `<package>` with the package name and run the command generated on the page to pull the package.

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.



Maven Repository

Last updated：2024-01-03 11:31:26

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

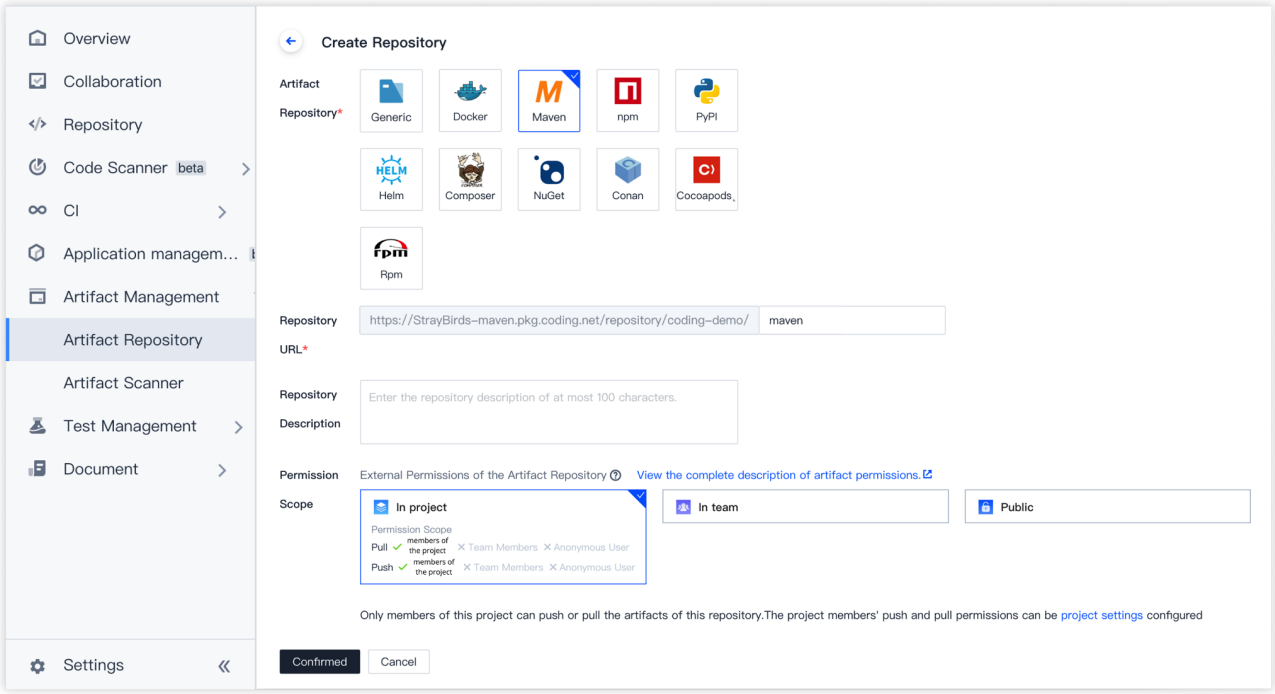
Open CODING-AR

1. Log in to the CODING Console and click **Use Now** to go to CODING page.
2. Click

in the upper-right corner to open the project list page and click a project icon to open the project.
3. In the menu on the left, click **Artifact Management**.

Create an Artifact Repository

Click **Create Repository**.
Select Maven as the repository type.
Enter a repository name.
Enter a repository description (optional).
Configure the permissions of different roles on the current repository. By default, all project members have the **Pull** and **Push** permissions.
Click **Create**.



Configure Authentication Information

Authentication information must be configured before you can pull artifacts from or push artifacts to CODING-AR.

Configure authentication information in either of the following ways:

Configure the access token in settings.xml.

Configure your CODING account and password in settings.xml.

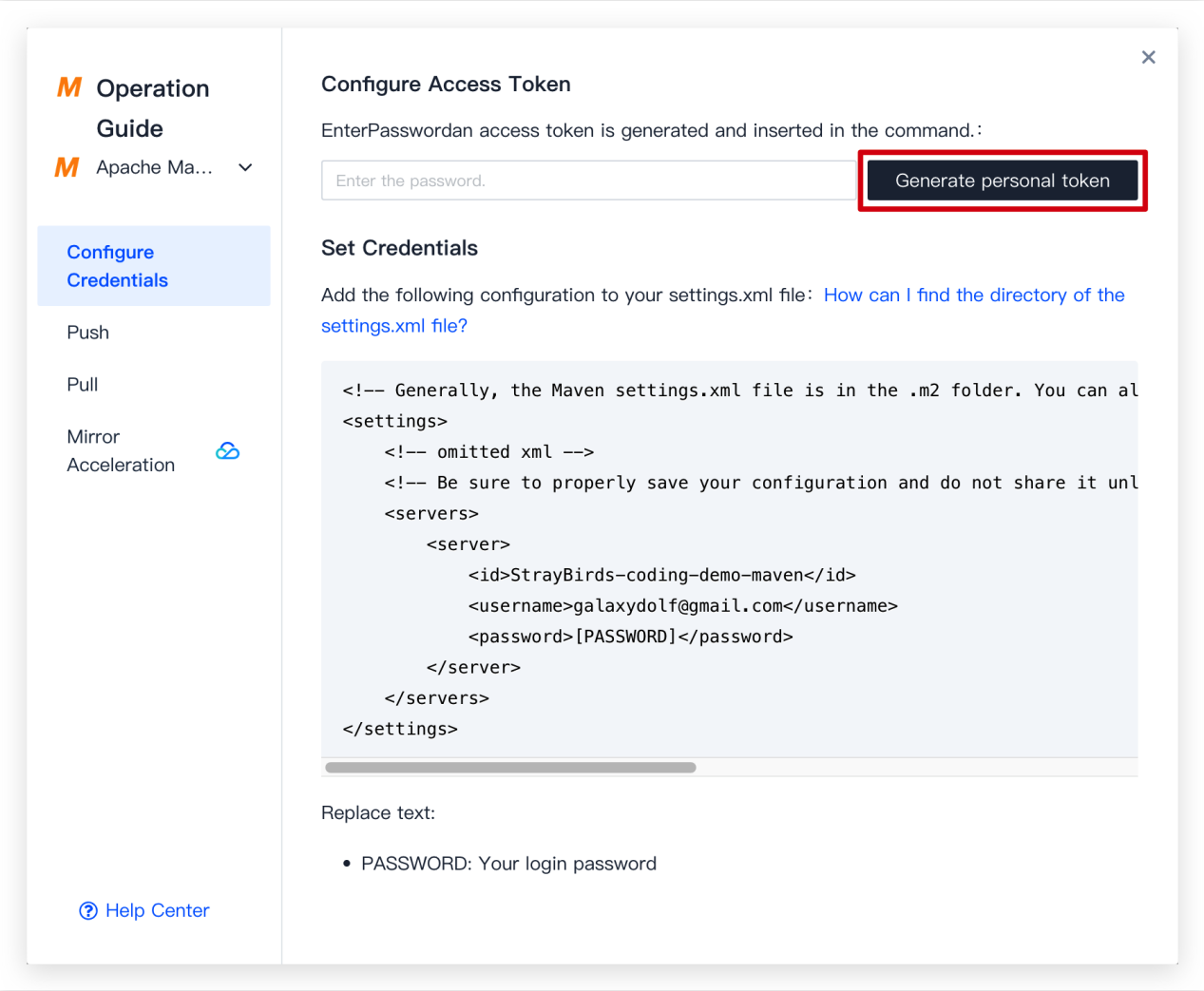
We recommend you use an **access token** to generate the authentication configuration.

Note:

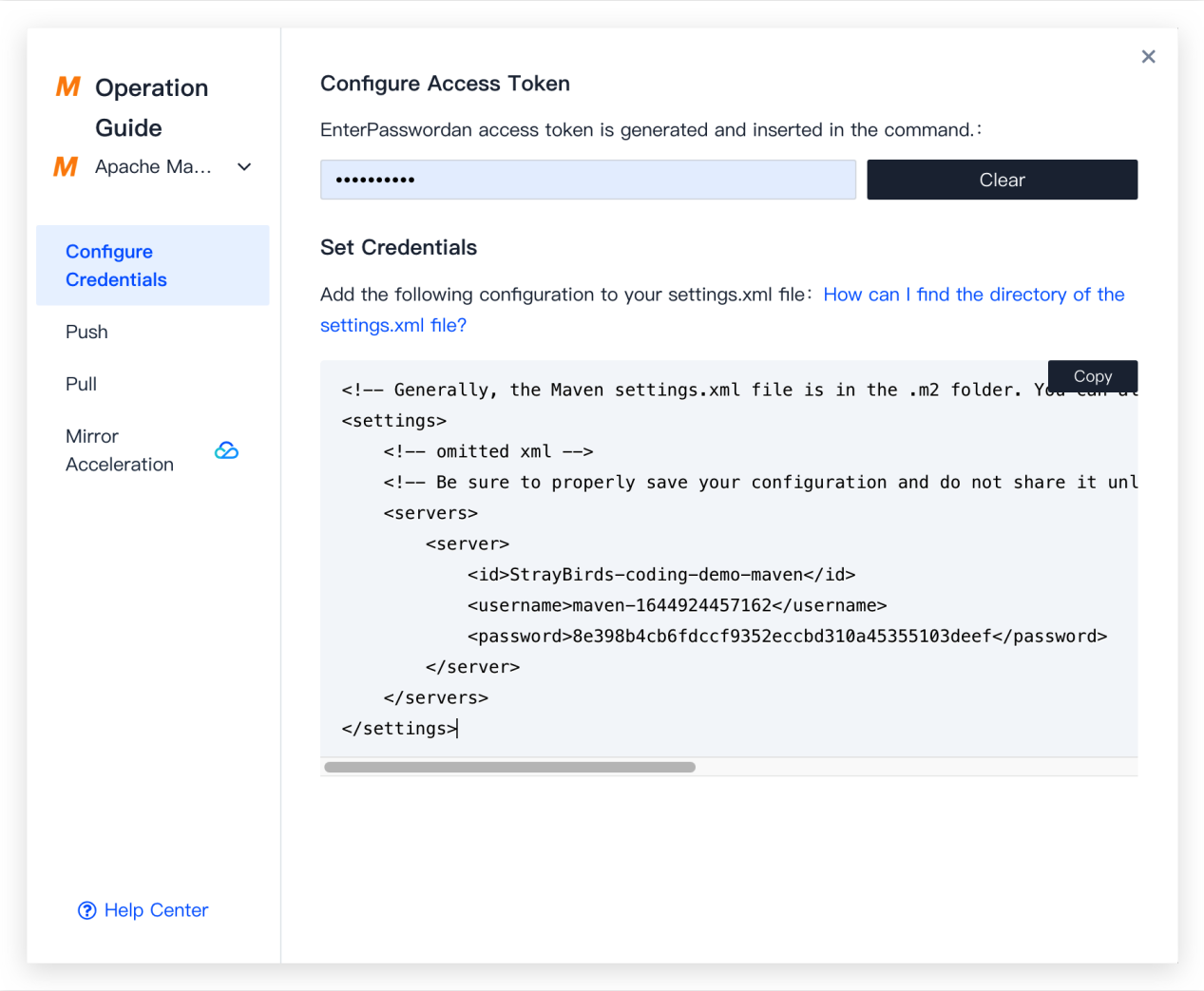
For details about the Maven settings.xml, refer to FAQs.

Method 1: generate configuration from access token

1. On the guide page, click **Generate configuration from access token** and enter the account login password in the pop-up window.

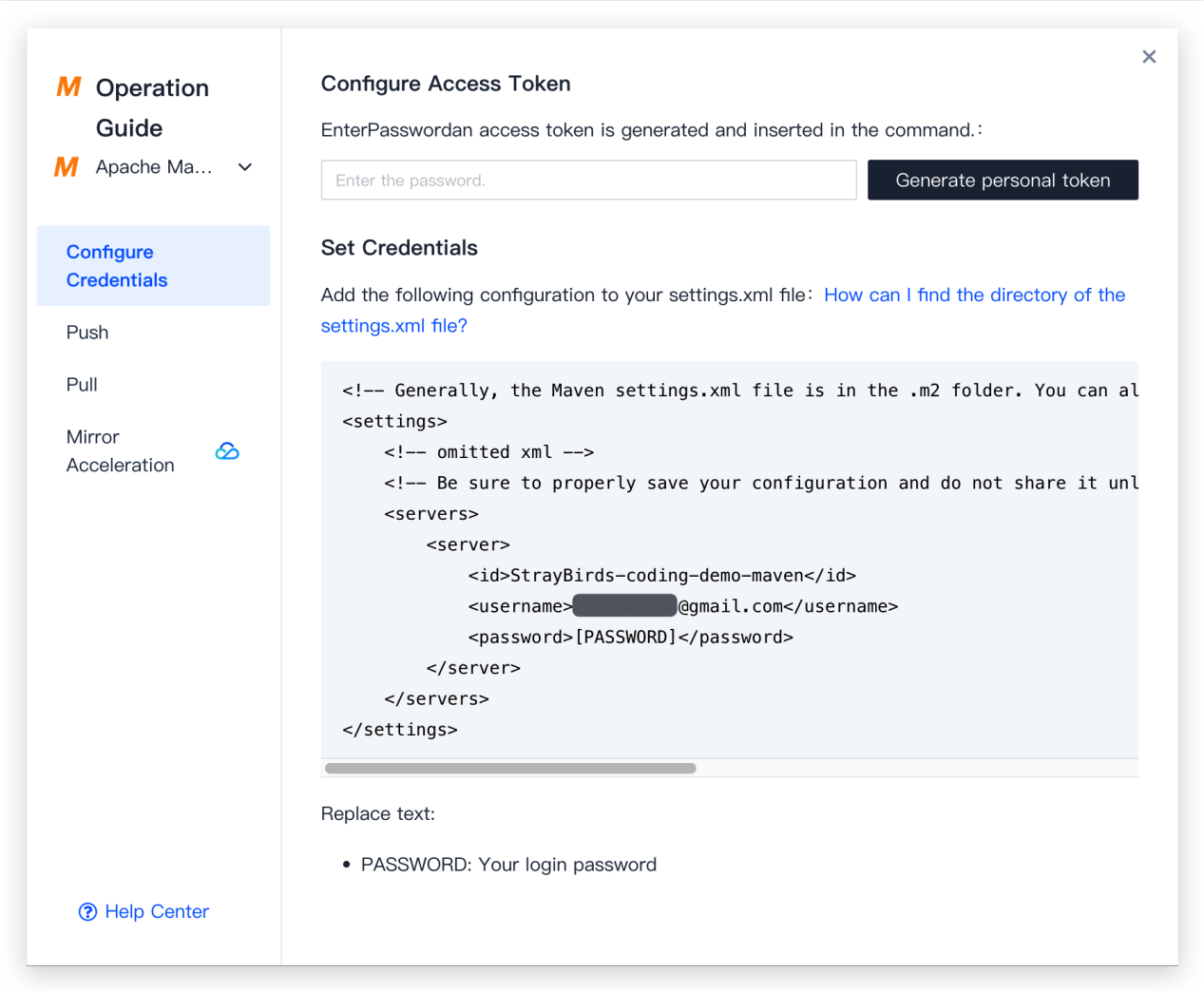


2. Copy the configuration generated and add it to settings.xml.



Method 2: configure account password manually

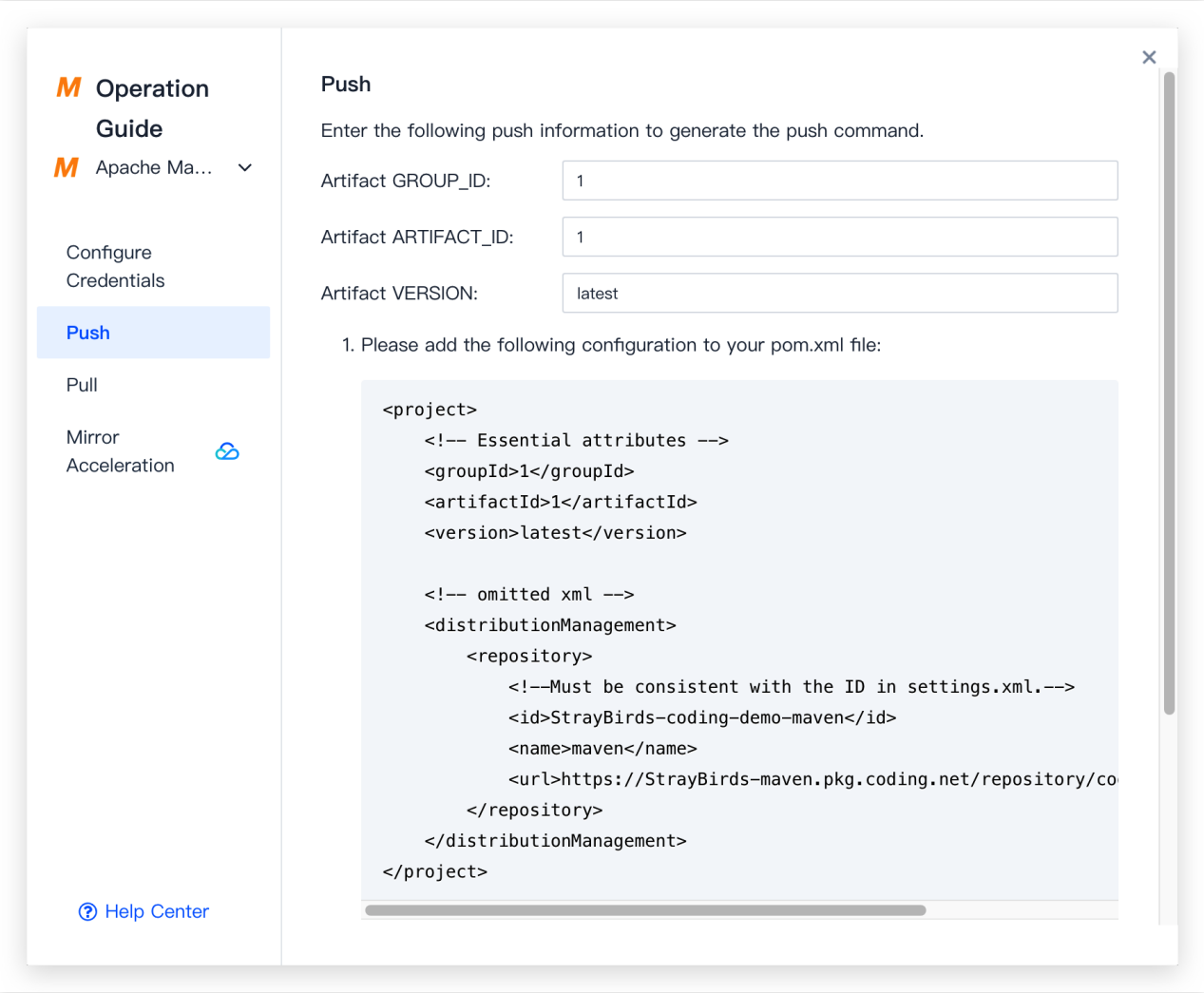
On the guide page, copy the following configuration, replace **PASSWORD** with your login password, and then add the configuration to settings.xml.



Compile and Upload a Maven Artifact

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

1. On the guide page, copy the following configuration to pom.xml.

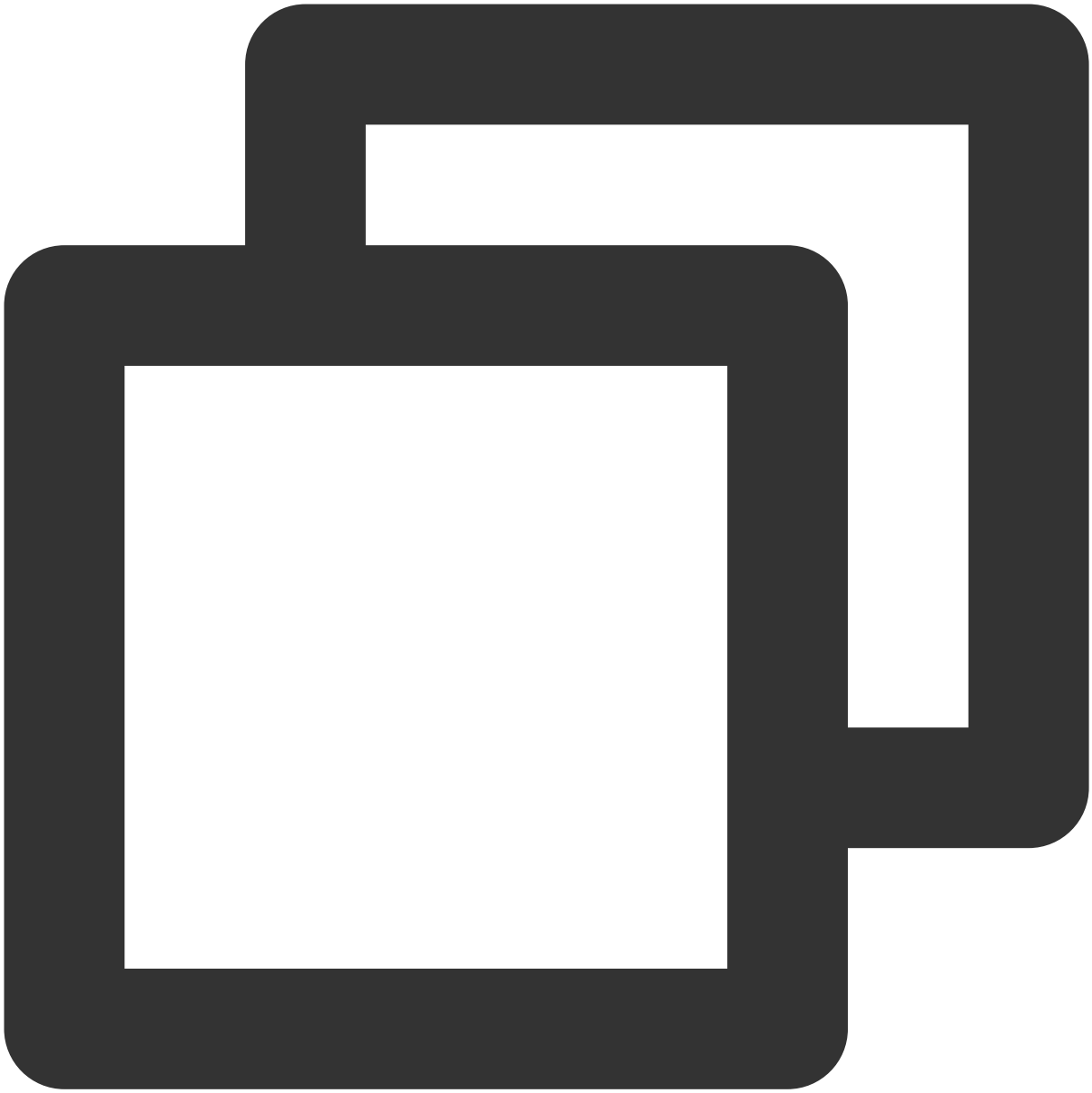


Generally, groupId, artifactId, and version configurations already exist for a Maven project. You only need to add `distributionManagement` to it.

demo-for-artifacts/pom.xml

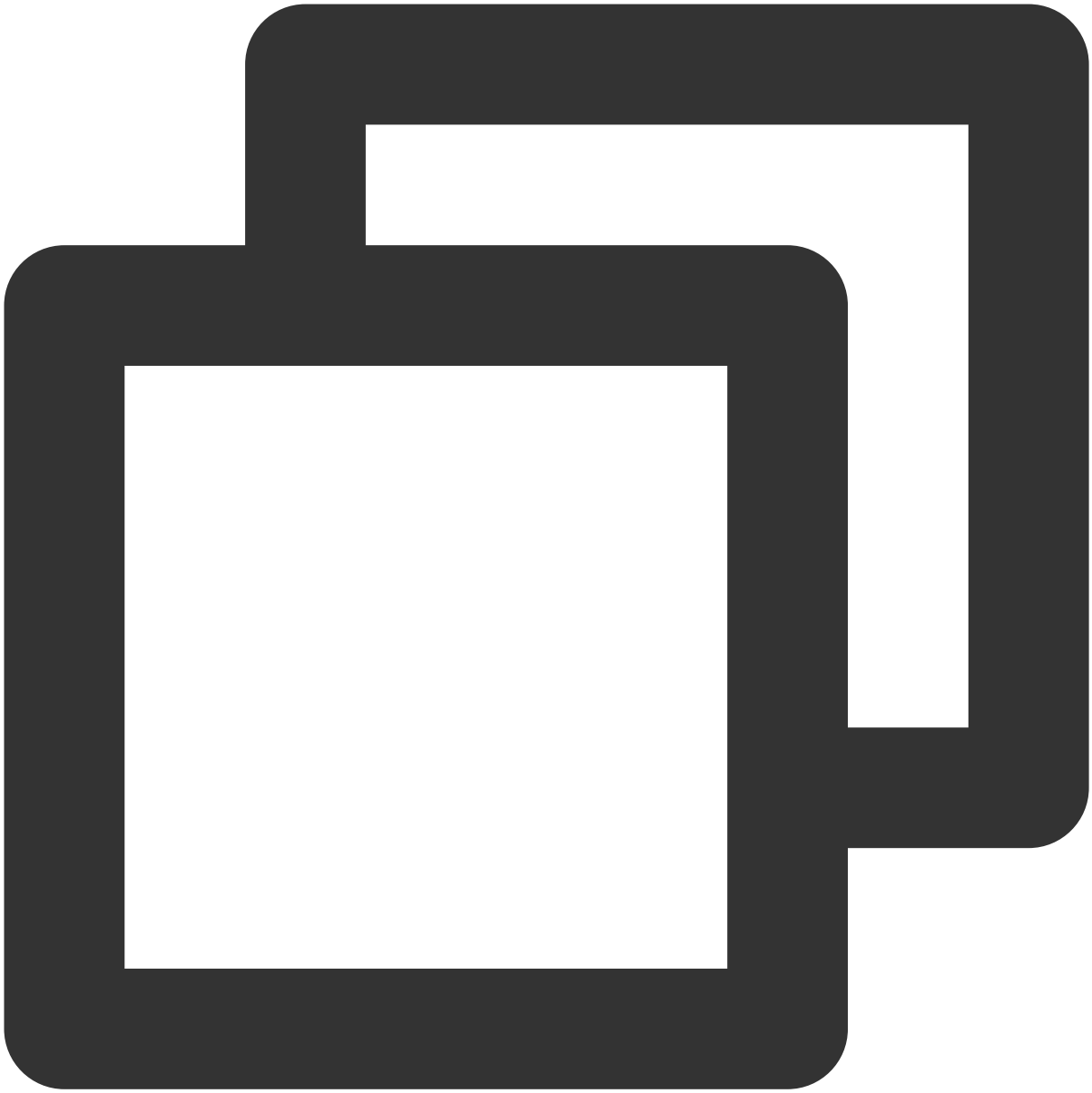
```xml
<project xmlns="http://maven.apache.org/POM/4.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.x
    <modelVersion>4.0.0</modelVersion>

    <groupId>coding</groupId>
    <artifactId>demo-for-artifacts</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <packaging>jar</packaging>

    <name>demo-for-artifacts</name>
    <url>http://maven.apache.org</url>

    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    </properties>

    <dependencies>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>3.8.1</version>
            <scope>test</scope>
        </dependency>
    </dependencies>

    <distributionManagement>
        <repository>
            <!-- settings.xml -->
            <id>anywhere-coding-demo-my-maven</id>
            <name>my-maven</name>
            <url>https://anywhere-maven.pkg.coding.net/repository/coding-demo/my-maven/</url>
        </repository>
    </distributionManagement>
</project>
```

2. Run the mvn deploy command.



```
mvn deploy
```

If settings.xml is not found, add `-s` to the end of the command and add the path of the settings file.

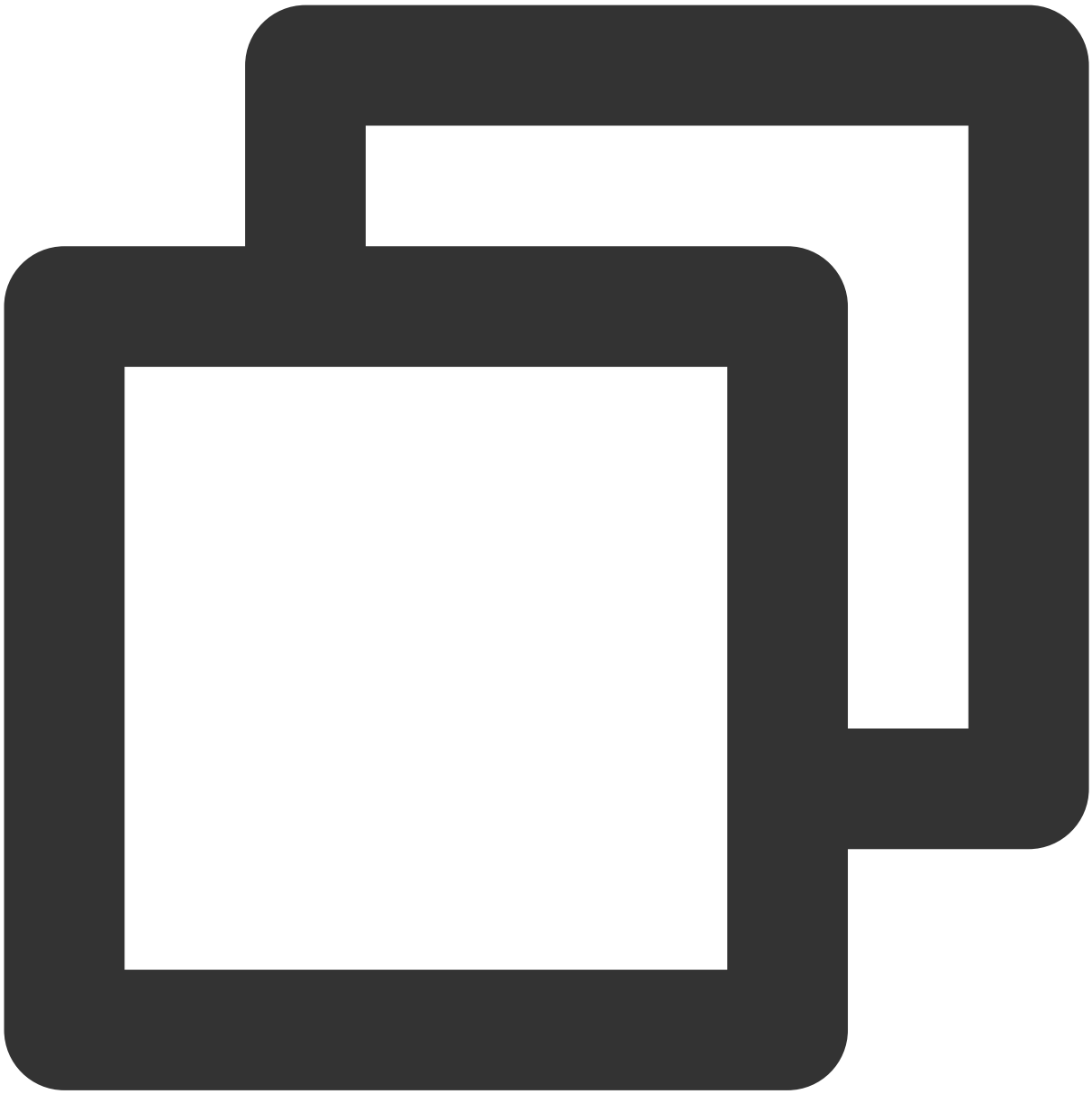


```
mvn deploy -s "/Users/somebody/software/apache-maven-3.6.2/conf/settings.xml"
```

3. If a build success message is displayed, refresh the repository page to view the latest artifacts.

Upload a Maven Package Without Source Code

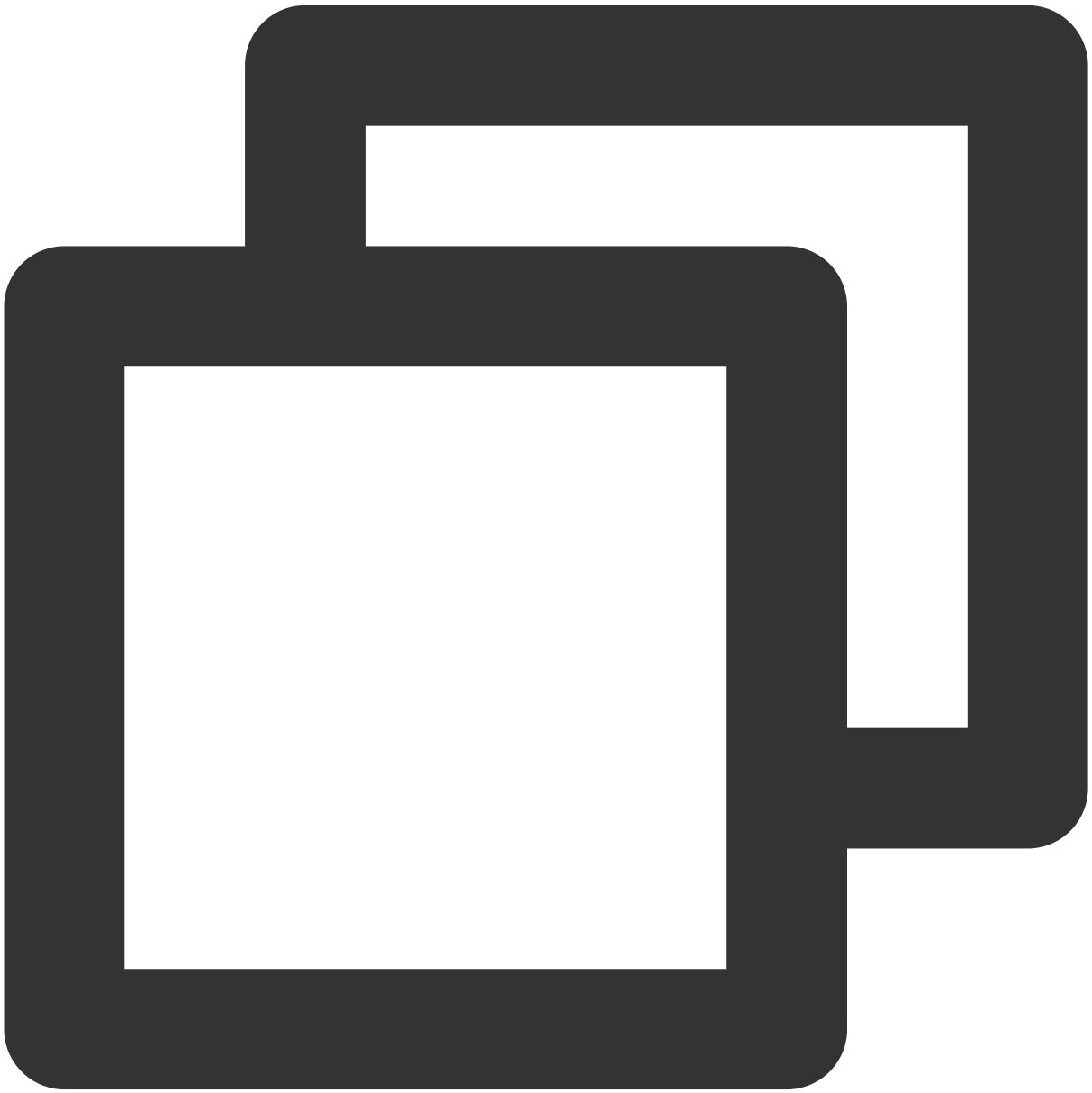
If a third-party Maven package is not officially released to a repository and only a JAR package is provided without source code, you can upload the JAR package to the repository by running the following command:



```
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 `pom.xml` is provided by the third party, you can obtain group, artifact, and version information. For example, use the following command for the **WeChat Cloud Pay Java SDK**:



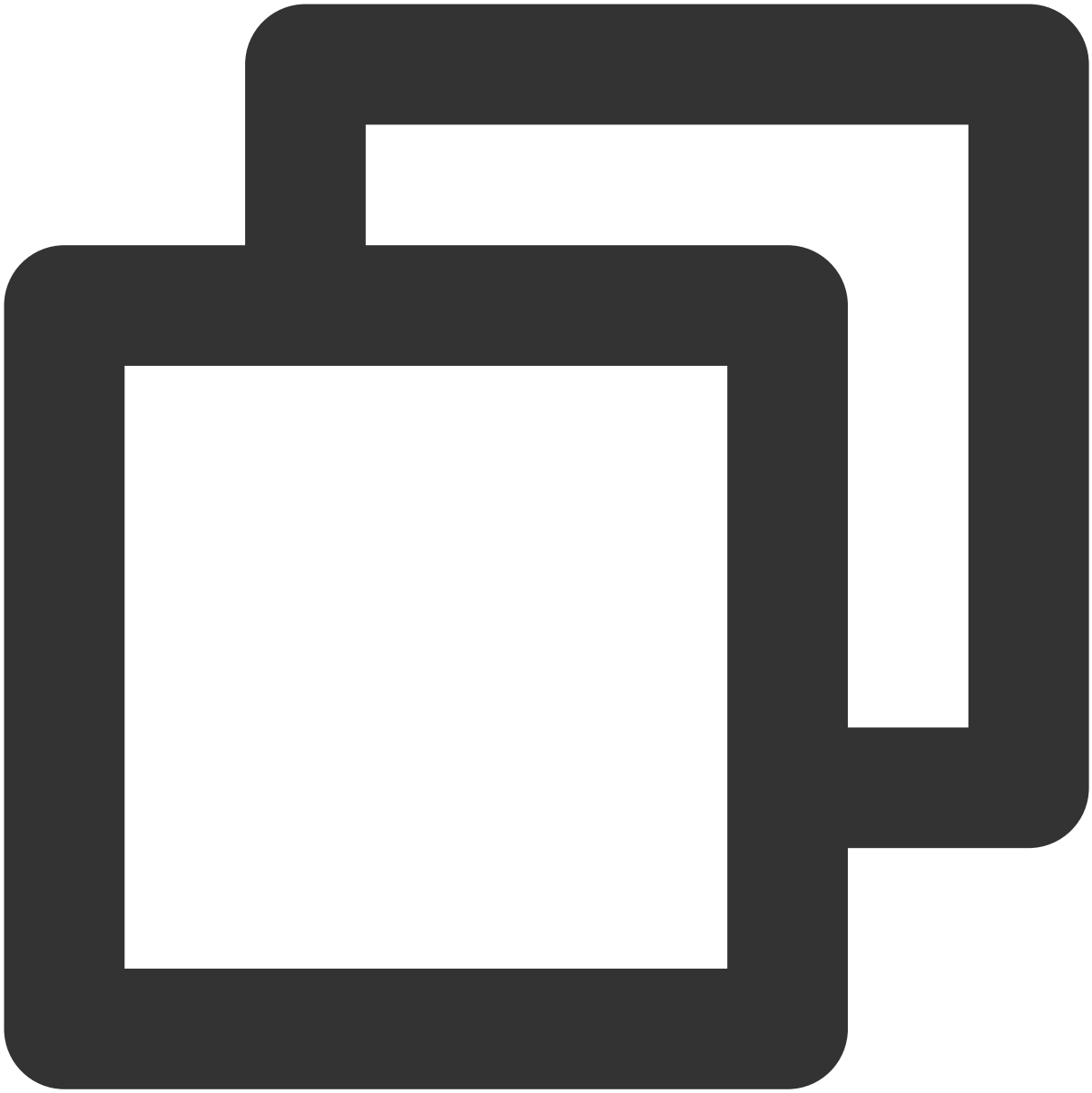
```
mvn deploy:deploy-file --settings ./settings.xml -Durl=https://coding-public-maven.
                       -DrepositoryId=coding-public-tencent-cloud-pay-sdk-java-tenc
```

```
                        -Dfile=../cloudpay.jar \\
                        -DpomFile=pom.xml
```

The following illustrates the JAR package upload page.

Pull a Maven Artifact

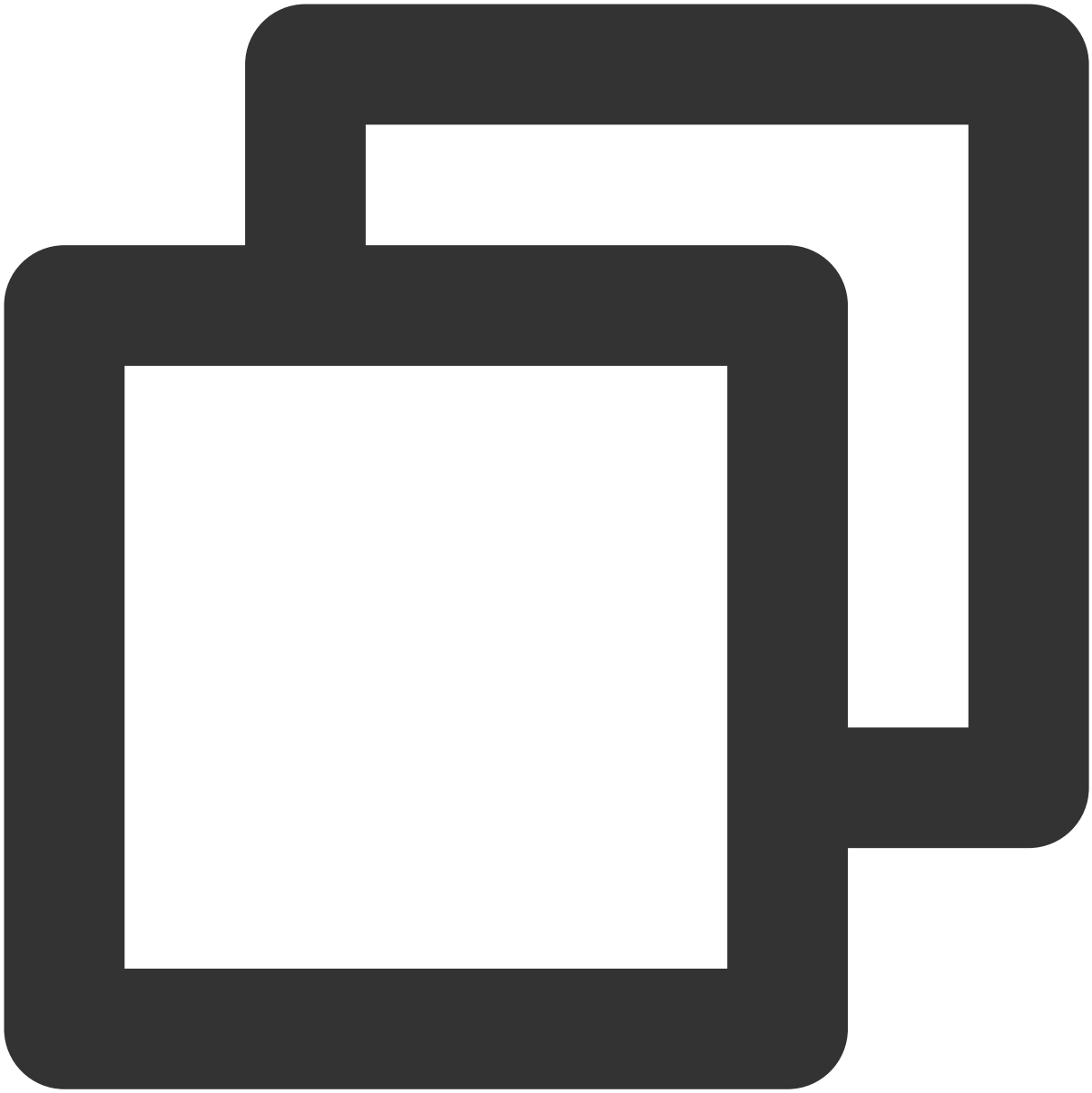
1. On the guide page, copy the configuration to settings.xml. For example, the configuration for the **WeChat Cloud Pay Java SDK** is as follows:



```
<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>
```

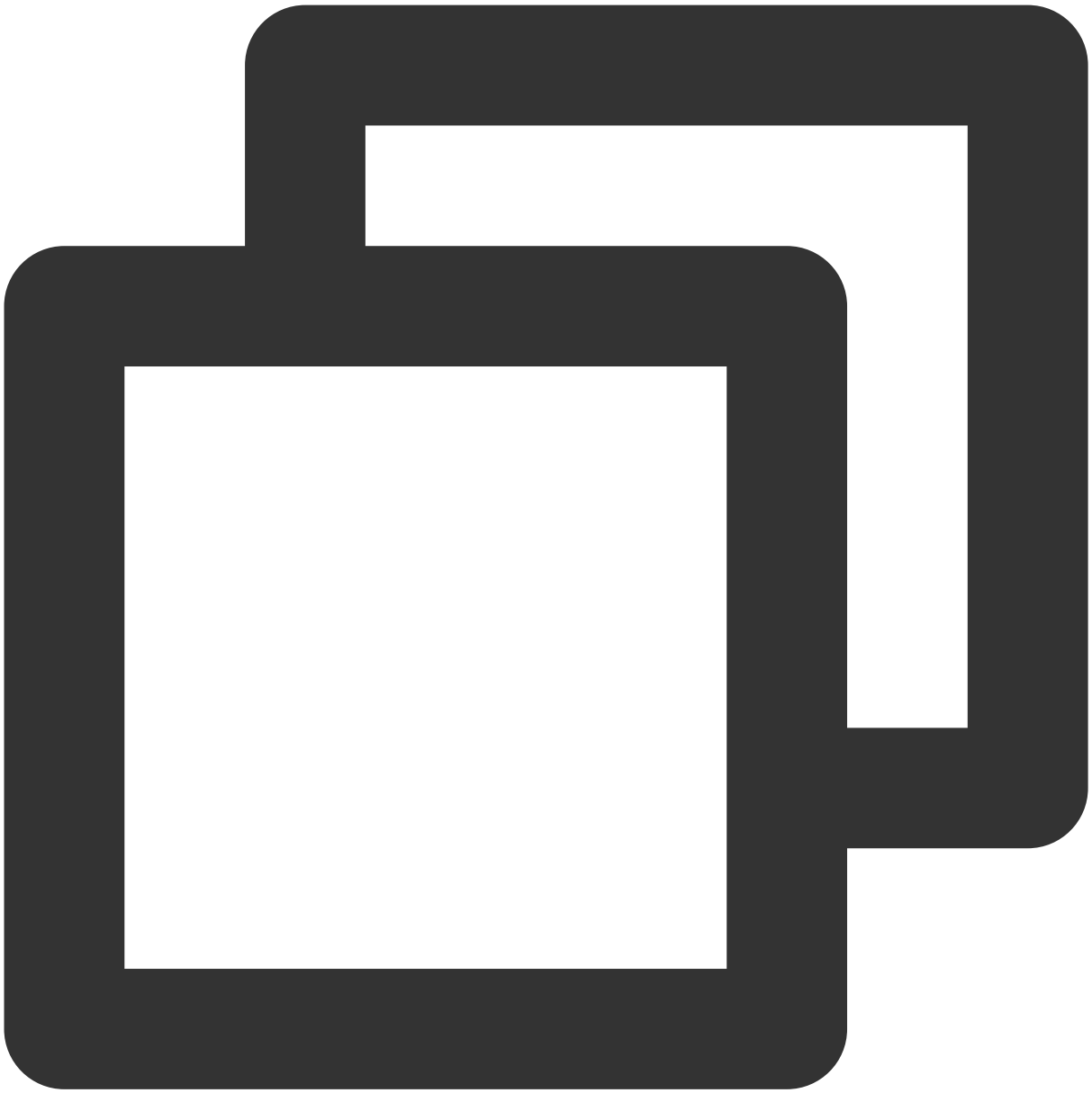
```
                    <name>tencent</name>
                    <url>https://coding-public-maven.pkg.coding.net/repository/tenc
                    <releases>
                        <enabled>true</enabled>
                    </releases>
                    <snapshots>
                        <enabled>true</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 Cloud Pay Java SDK**, the configuration is as follows:



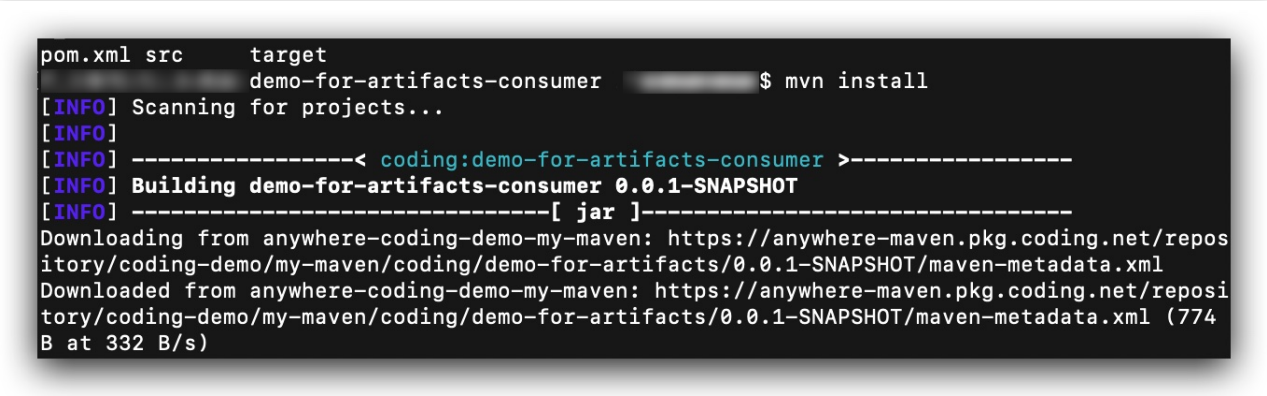
```
<project>
    <dependencies>
        <dependency>
            <groupId>com.tencent</groupId>
            <artifactId>cloudpay</artifactId>
            <version>1.6</version>
        </dependency>
    </dependencies>
</project>
```

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.



Helm Repository

Last updated：2024-01-02 17:58:34

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

Open CODING-AR

1. Log in to the CODING Console and click **Use Now** to go to CODING page.
2. Click

in the upper-right corner to open the project list page and click a project icon to open the project.
3. In the menu on the left, click **Artifact Management**.

Preparations

Note:

Before you begin:

Install Helm.

Create an artifact repository (see Basic Operations).

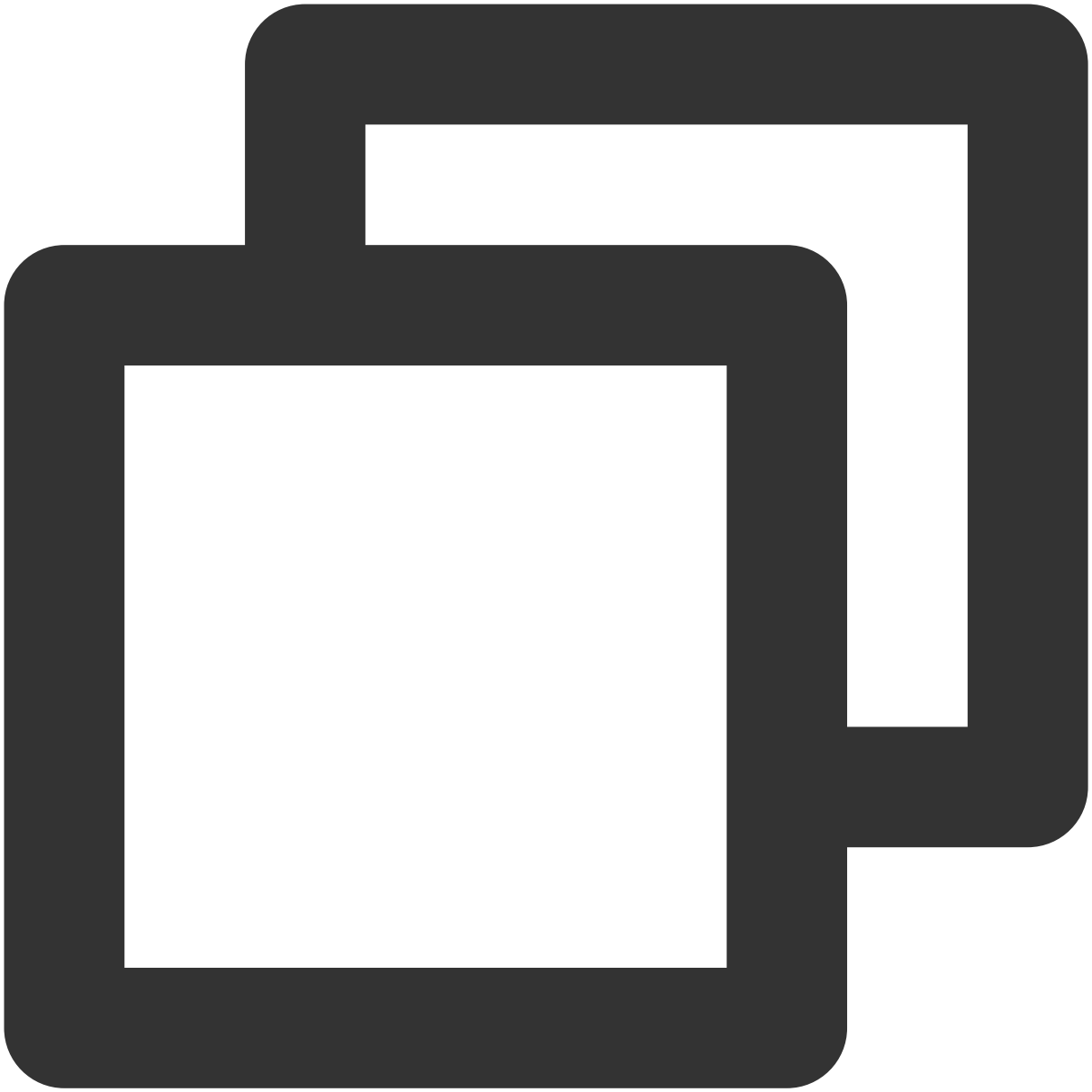
Select Helm as the repository type.

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.

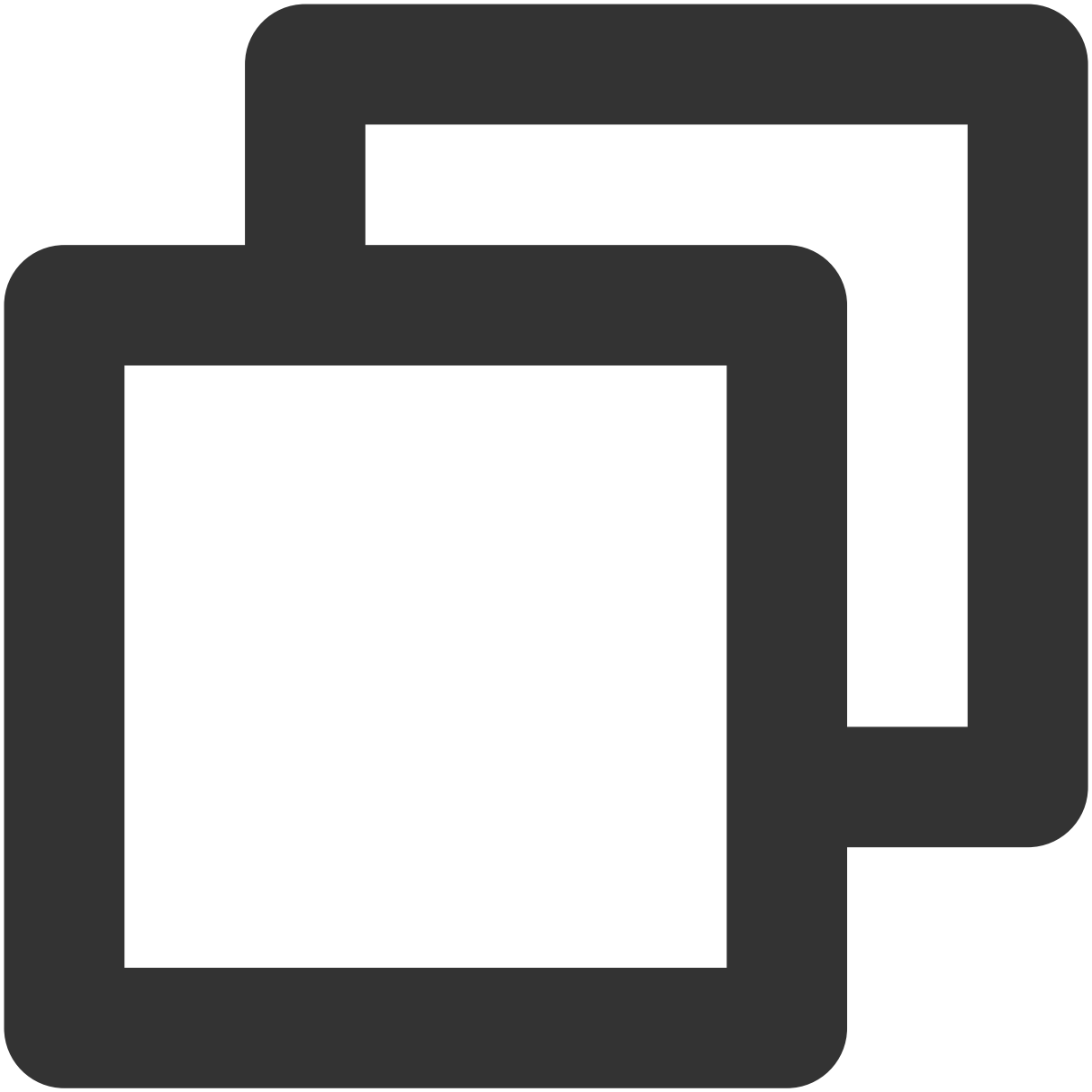
Method 1: create an image locally

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



```
helm create [name]
```

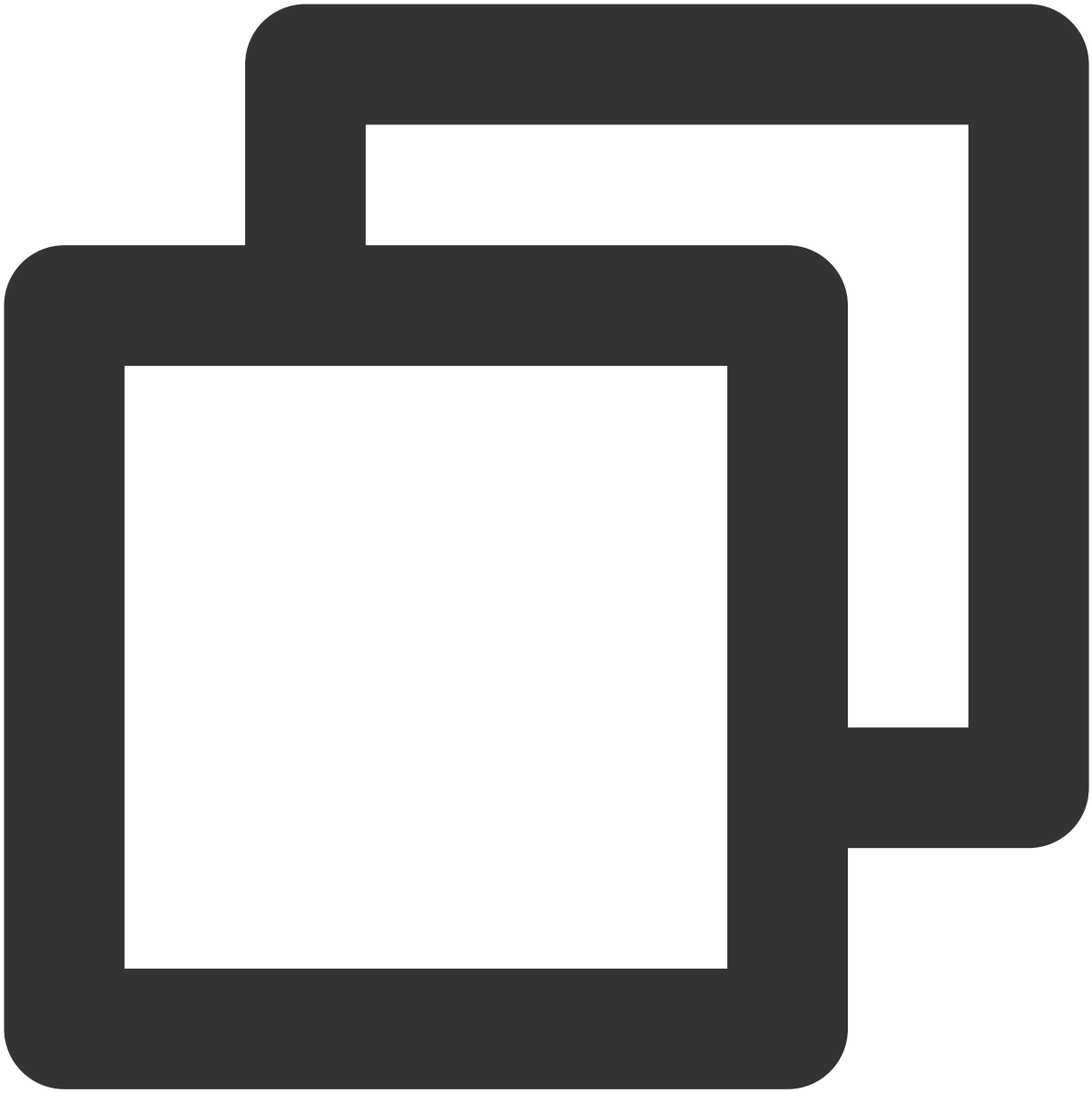
2. Package the chart.



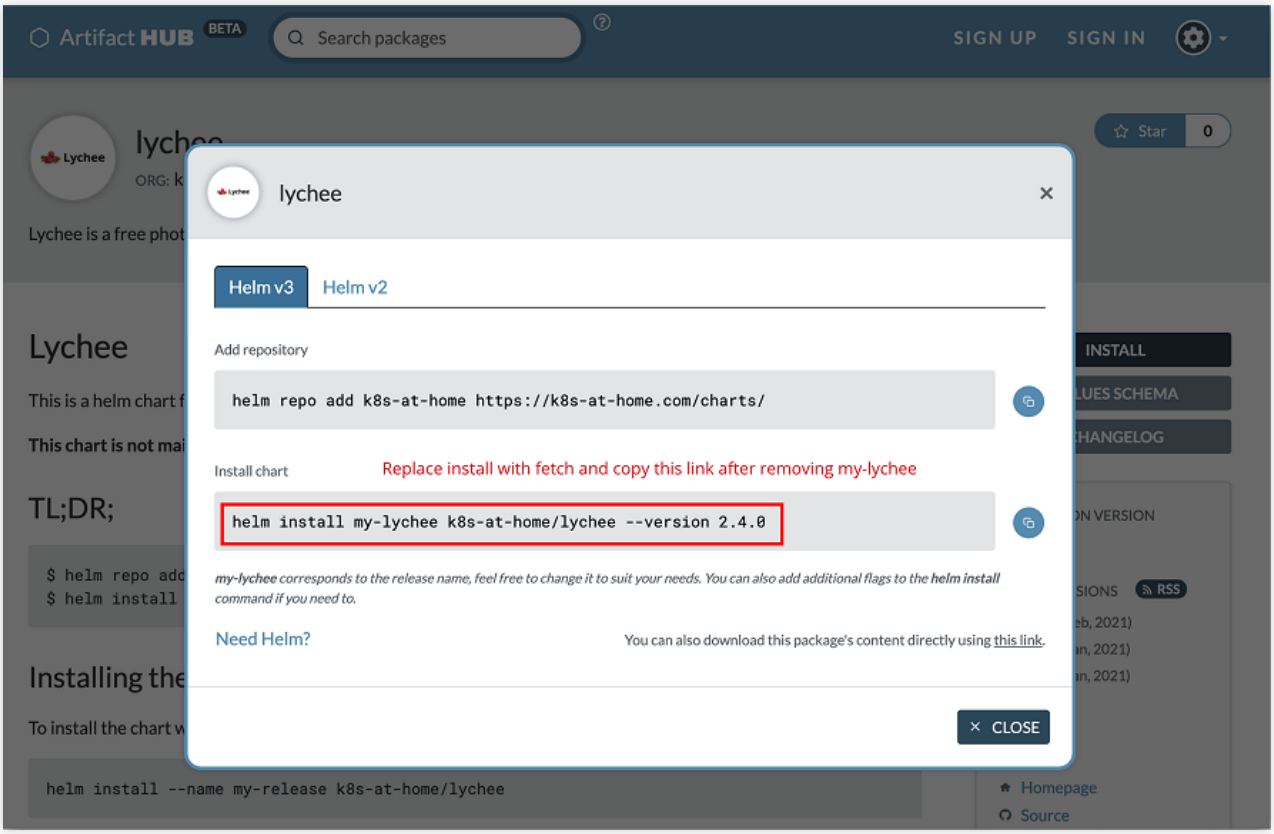
```
helm package [name]
```

Method 2: pull an artifact from Artifact Hub

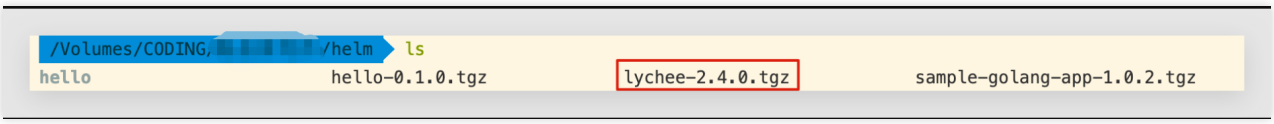
Search for a Helm chart and run the download command in a local directory.



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

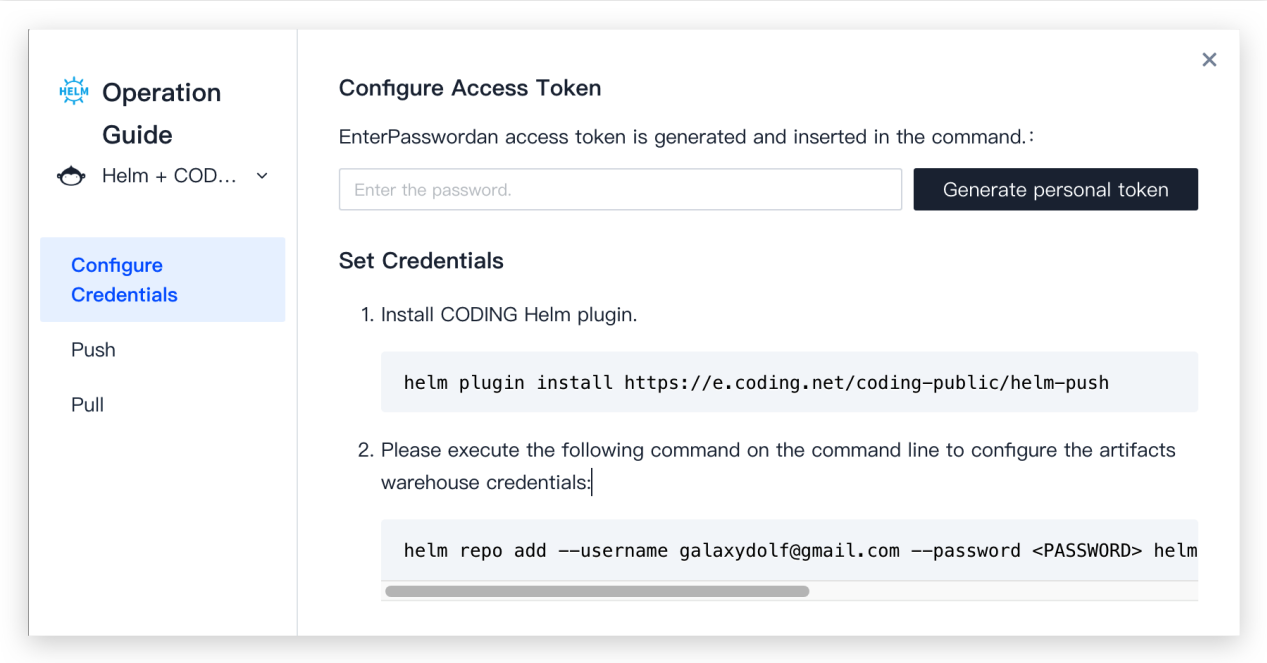


After the commands are successfully run, you can view the artifact locally.

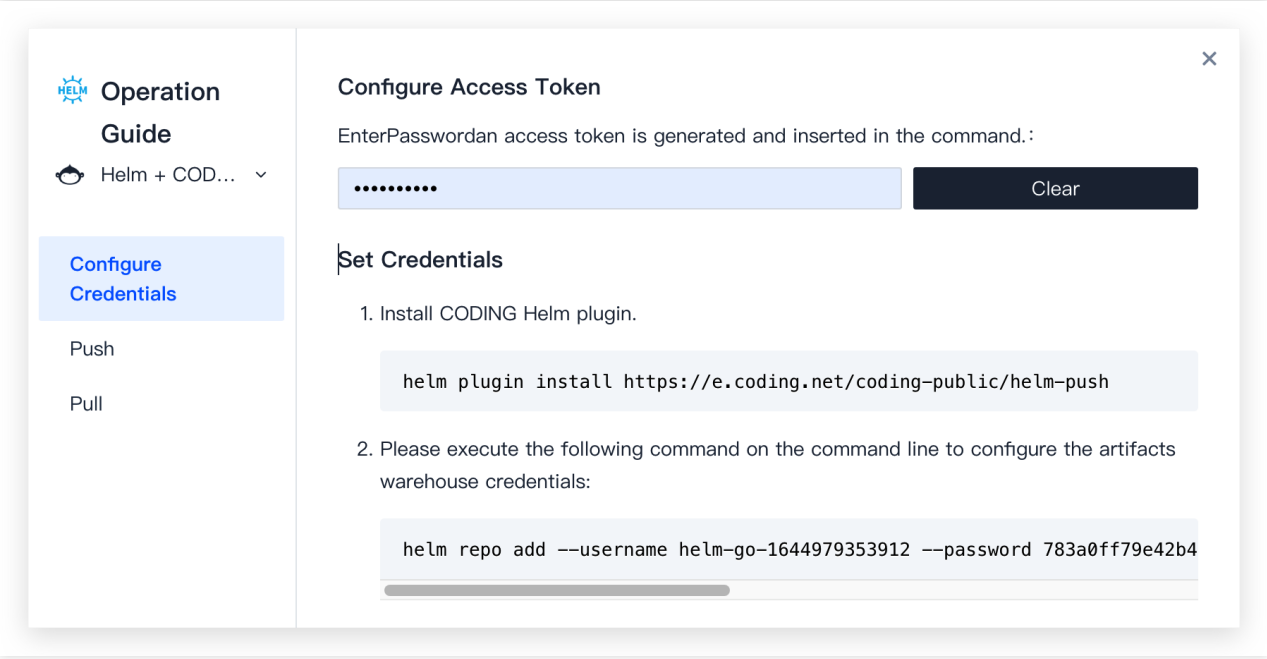


Configure Authentication Information

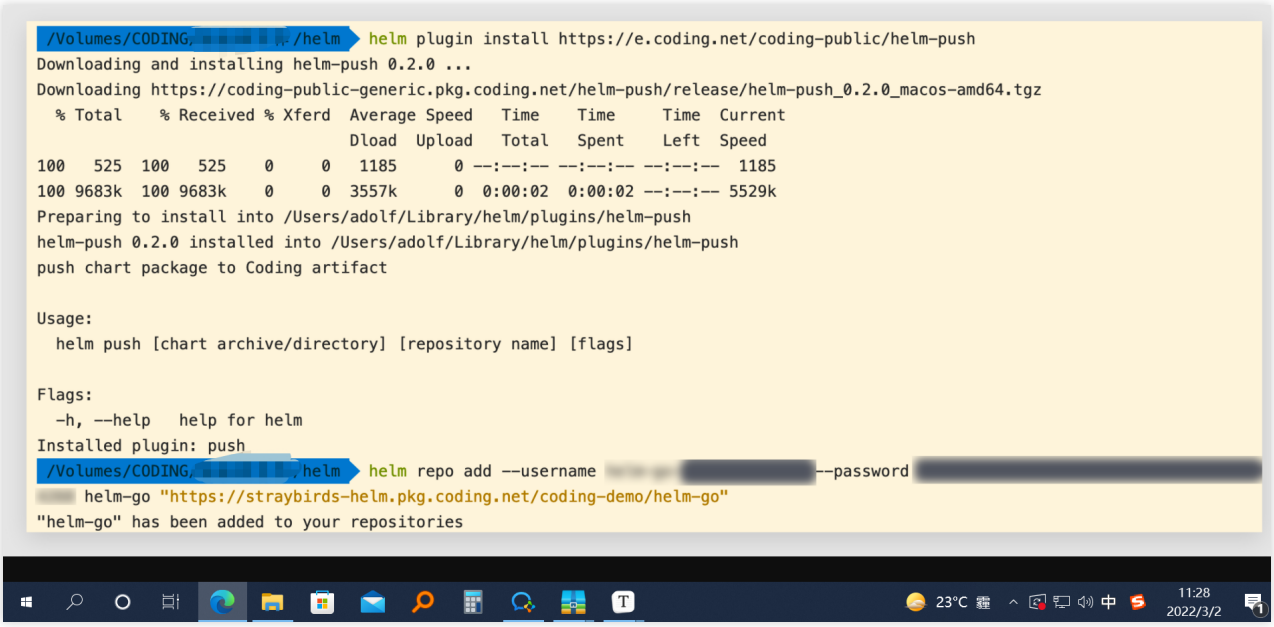
After creating a local artifact, you can push it to the remote repository. Before the push, you need to locally configure the authentication information of the remote repository. You can pull or push an artifact by using Helm + cURL or Helm + CODING Helm plugin as instructed in the **Guide**.



The following example uses the CODING Helm plugin:

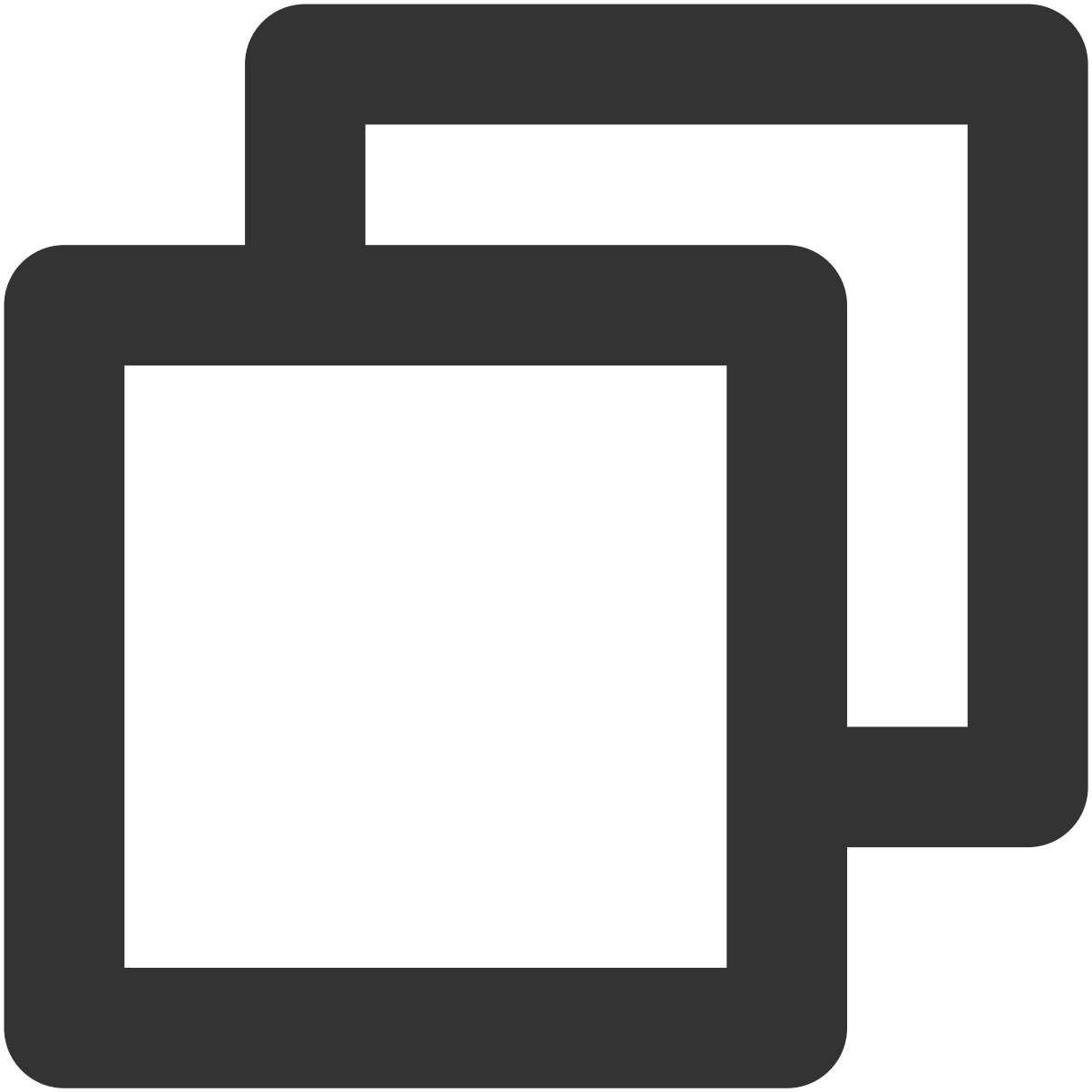


Copy and run the command in the guide to install the plugin. Then click **Generate configuration from access token**, and copy and run the configuration command in the package directory.



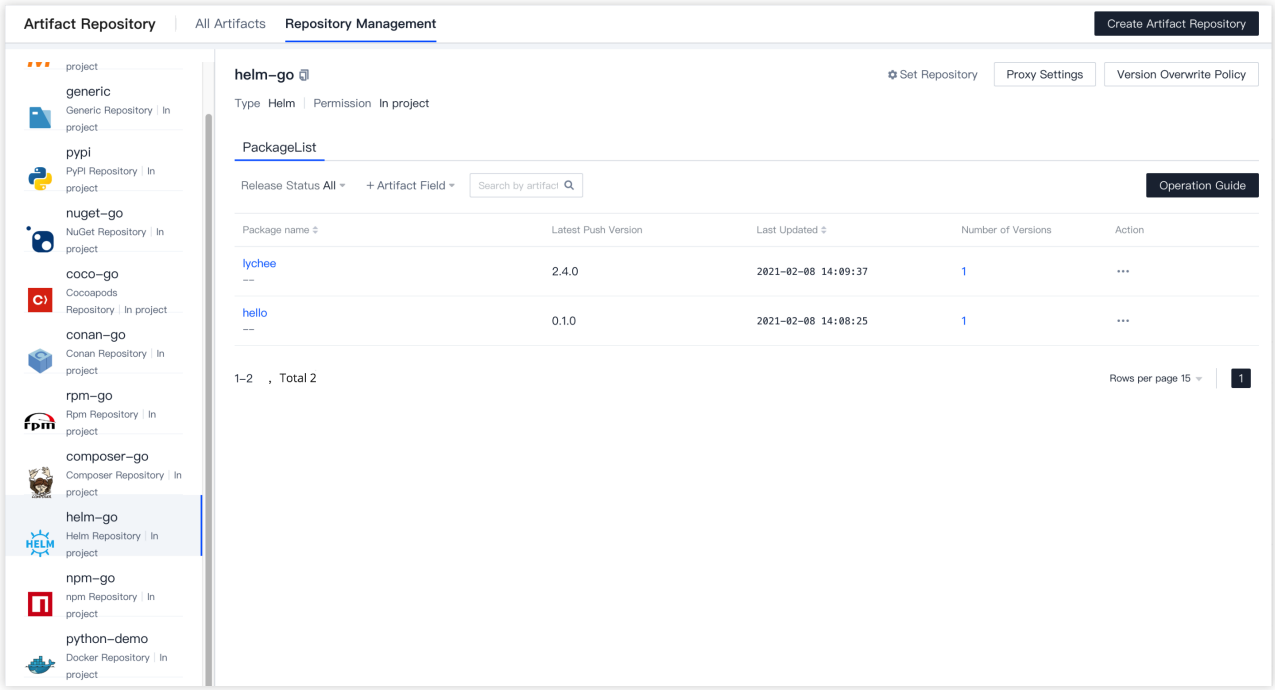
Push an Artifact

Go to the directory where the Helm chart is stored and run the command in the **Guide** to push the specified Helm chart to the repository.



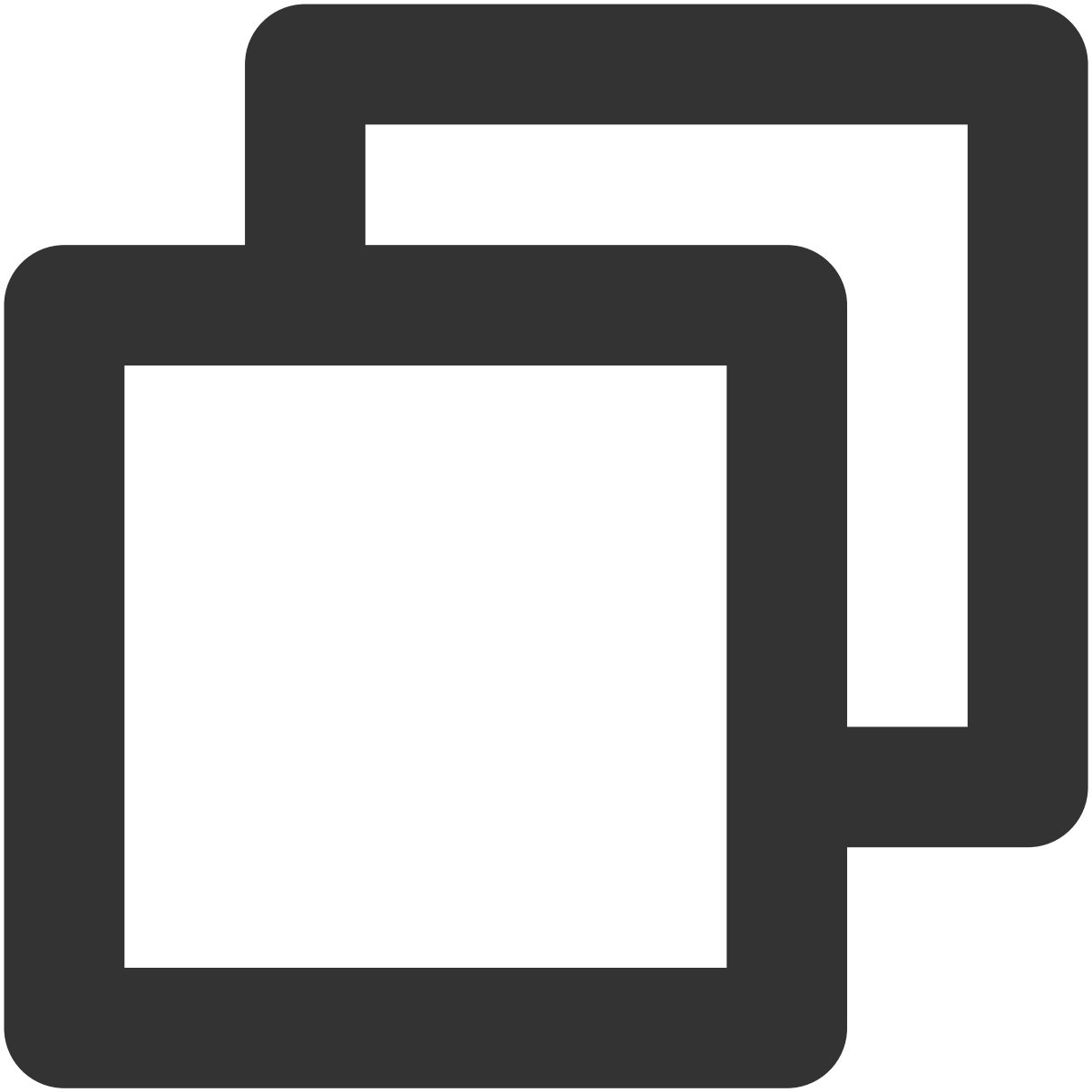
```
helm push [Package Name.tgz] [Repository Info in the Push Guide]
```

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



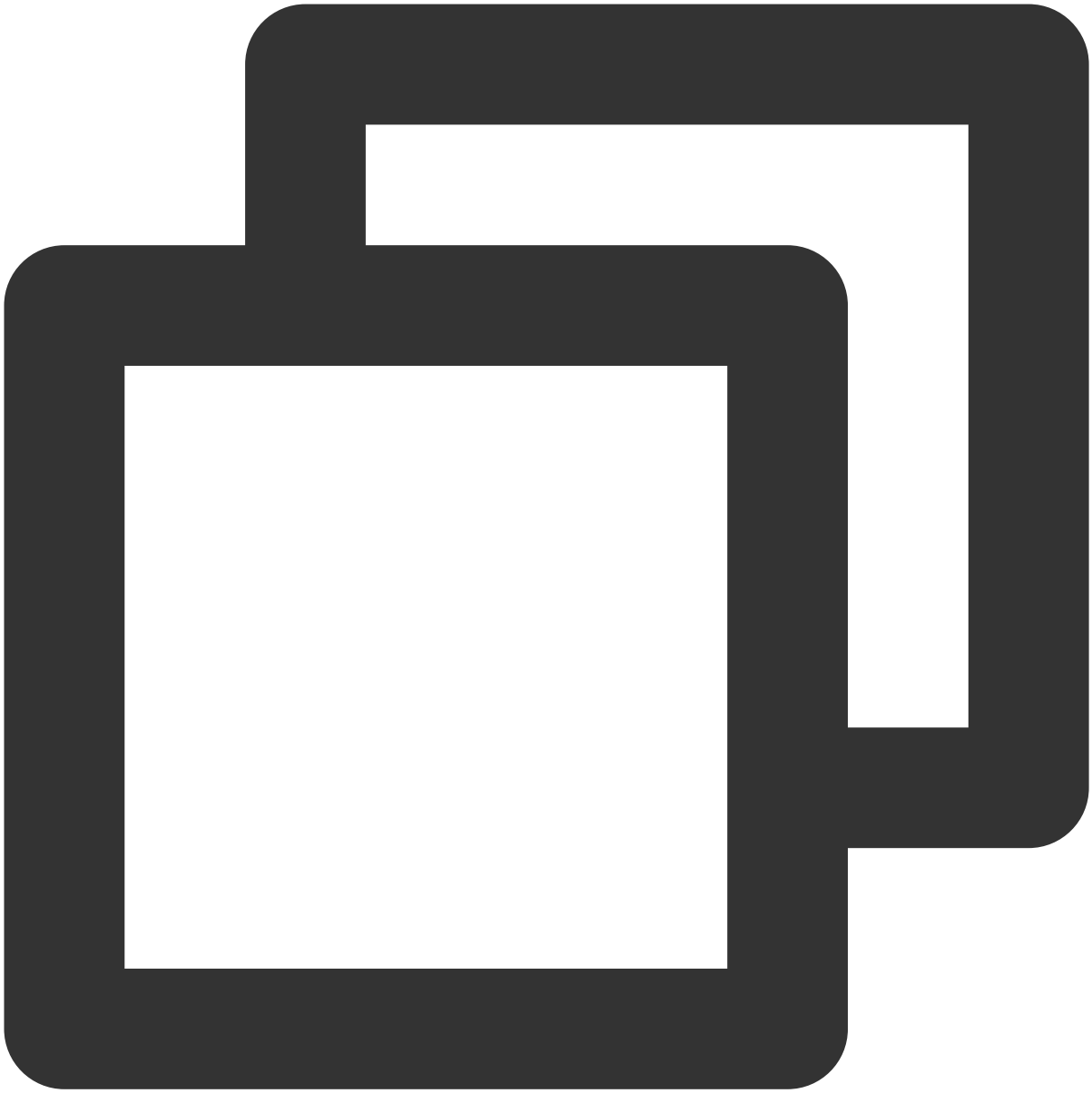
Pull an Artifact

If your artifact repository is updated, run the update command before pulling an artifact.



```
helm repo update
```

After the update is successful, run the pull command.



```
helm fetch [Artifact Info in the pull guide] --version [Version]
```

PyPI Repository

Last updated：2024-01-02 17:58:50

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.

Open CODING-AR

1. Log in to the CODING Console and click **Use Now** to go to CODING page.
2. Click

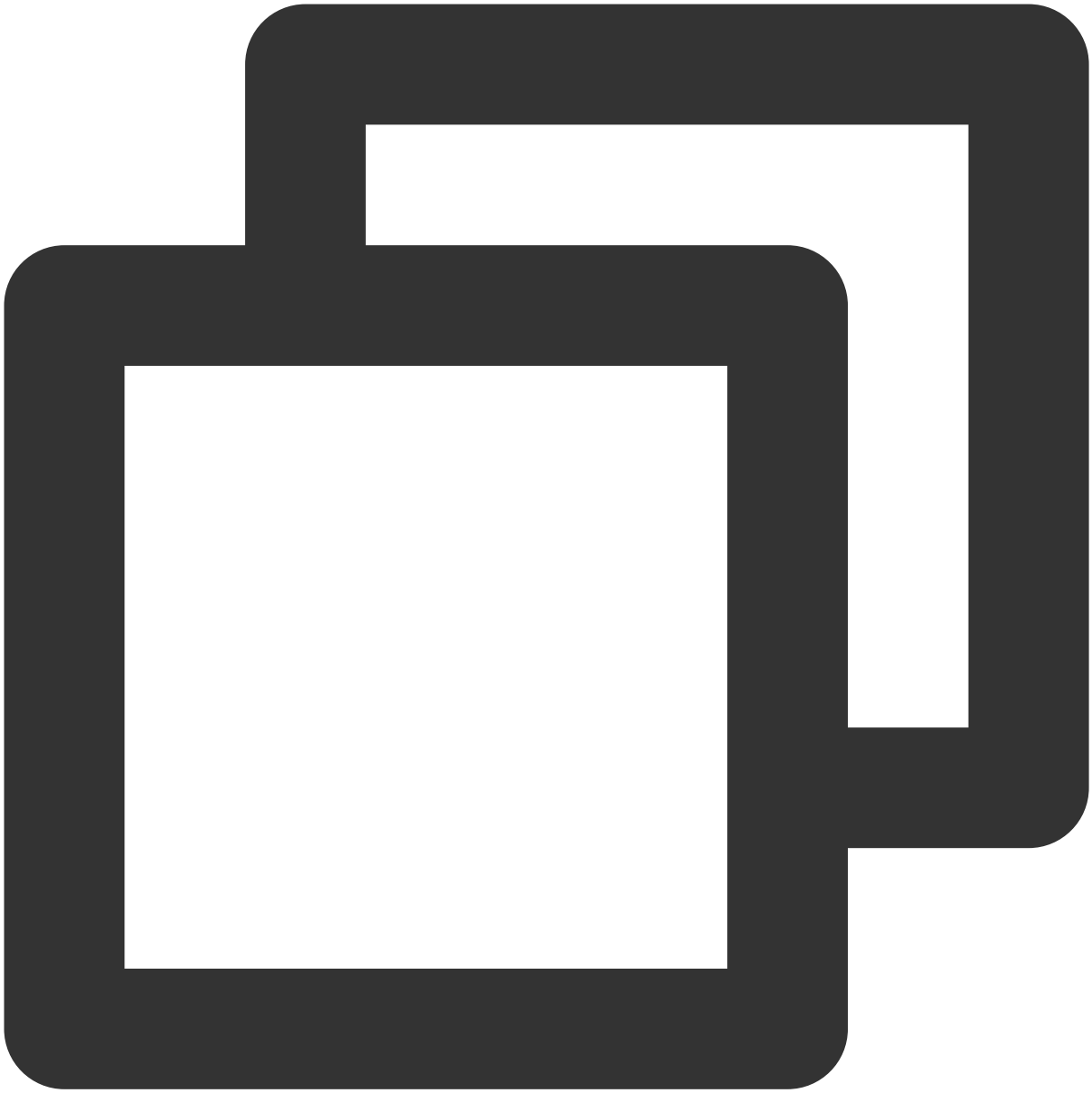
in the upper-right corner to open the project list page and click a project icon to open the project.
3. In the menu on the left, click **Artifact Management**.

Preparations

Note:
Before you begin:
Install Python3.
Create an artifact repository (see Basic Operations).
Select PyPI as the repository type.

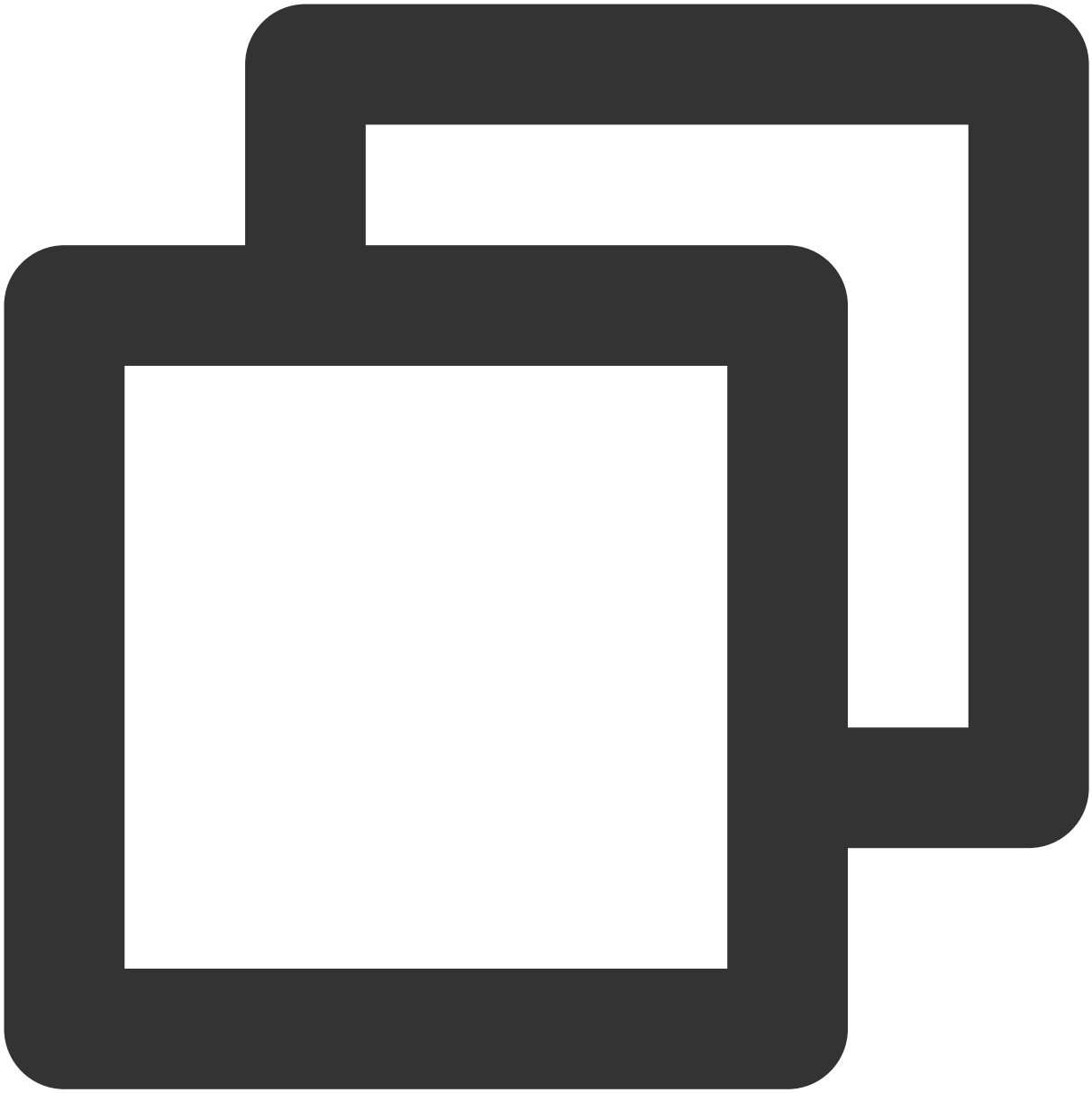
Initialize a Local PyPI Project

1. Create a demo directory for the PyPI project. Run the following command in the terminal to create a demo project folder.



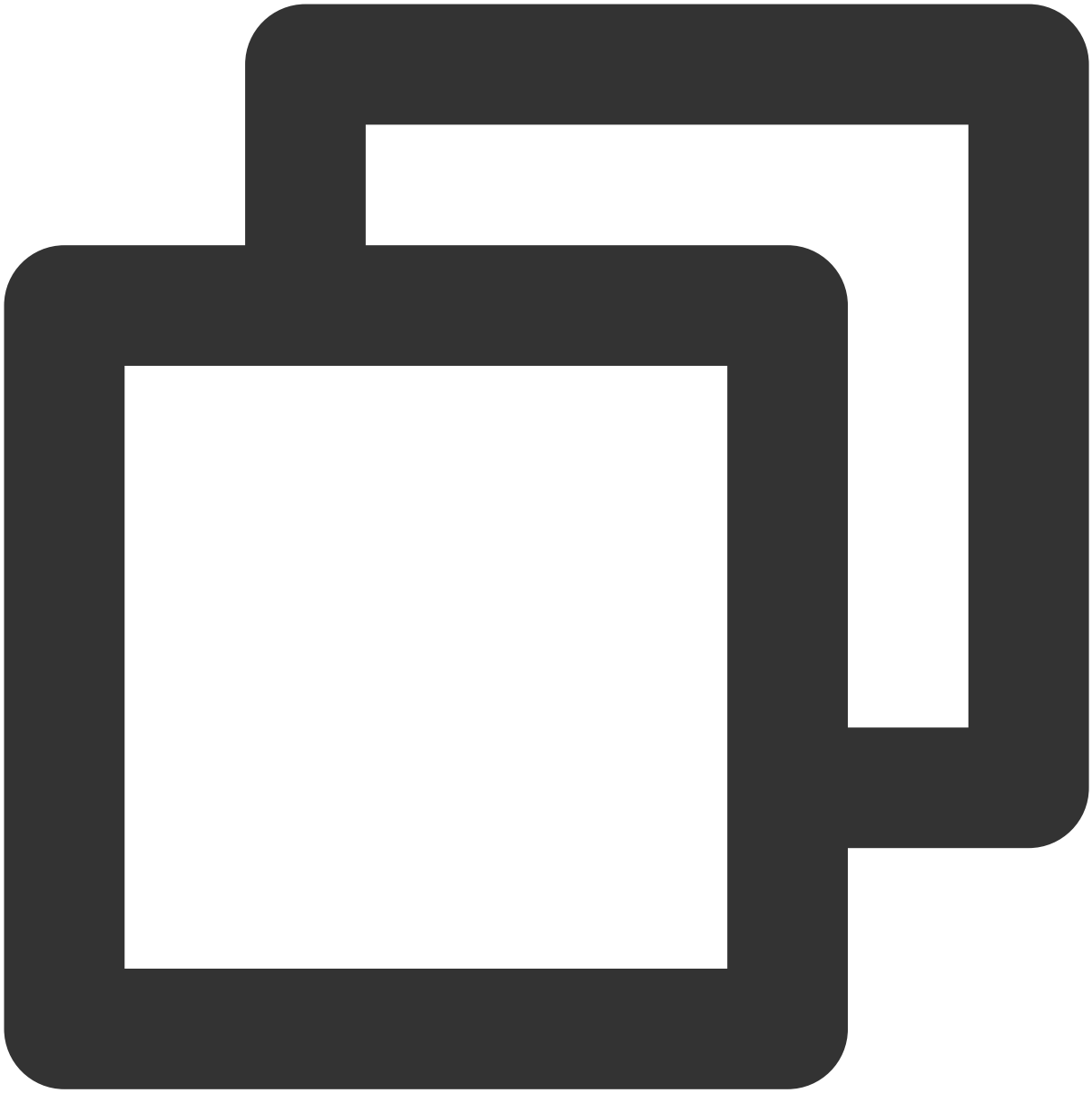
```
mkdir -p demo/example_pkg/__init__.py
```

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



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

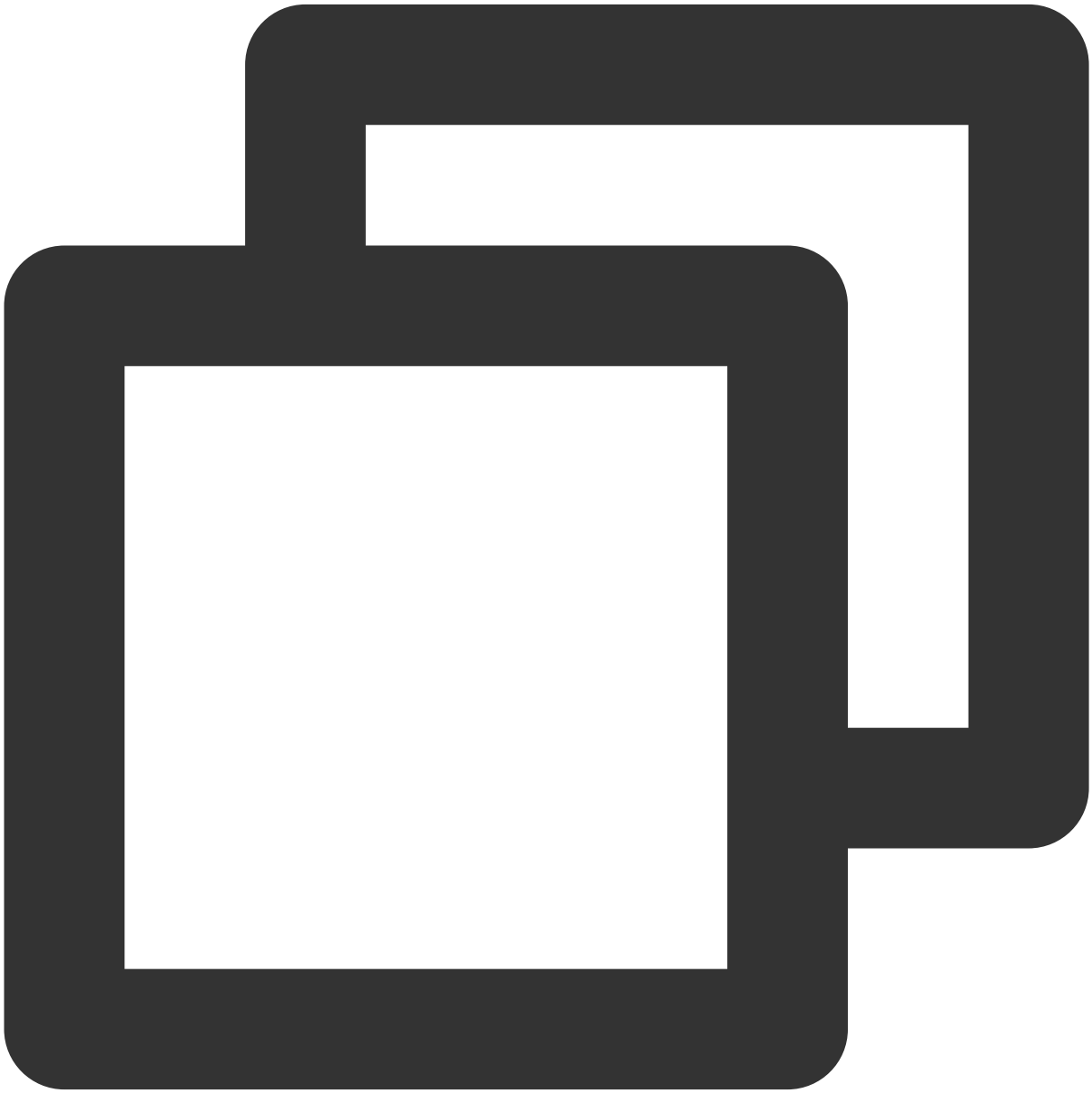
3. Paste the configuration to `setup.py` .



```
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=[
```

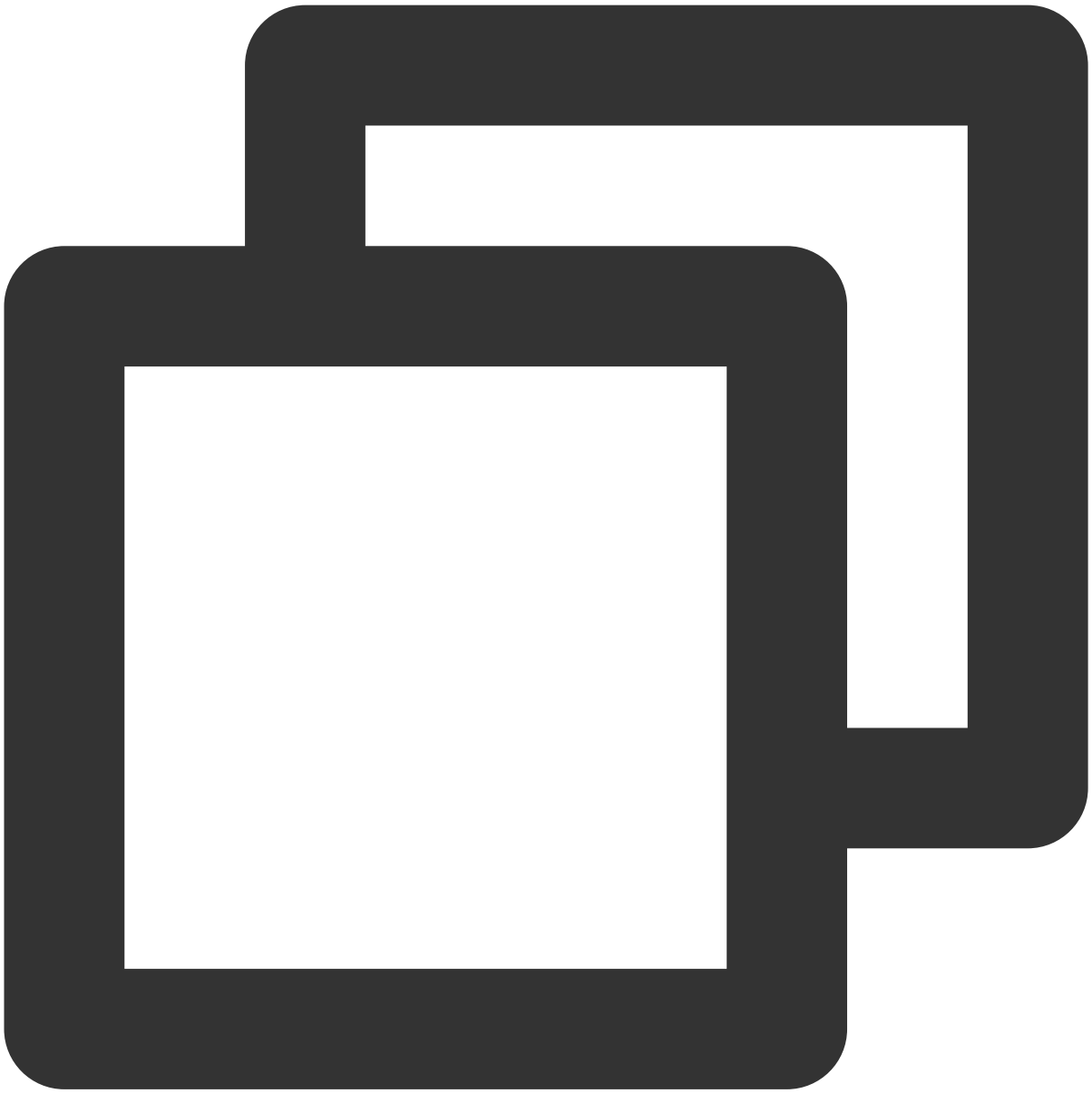
```
        "Programming Language :: Python :: 3",
        "License :: OSI Approved :: MIT License",
        "Operating System :: OS Independent",
    ],
    python_requires='>=3.6',
)
```

4. Install `setuptools` and `wheel` .



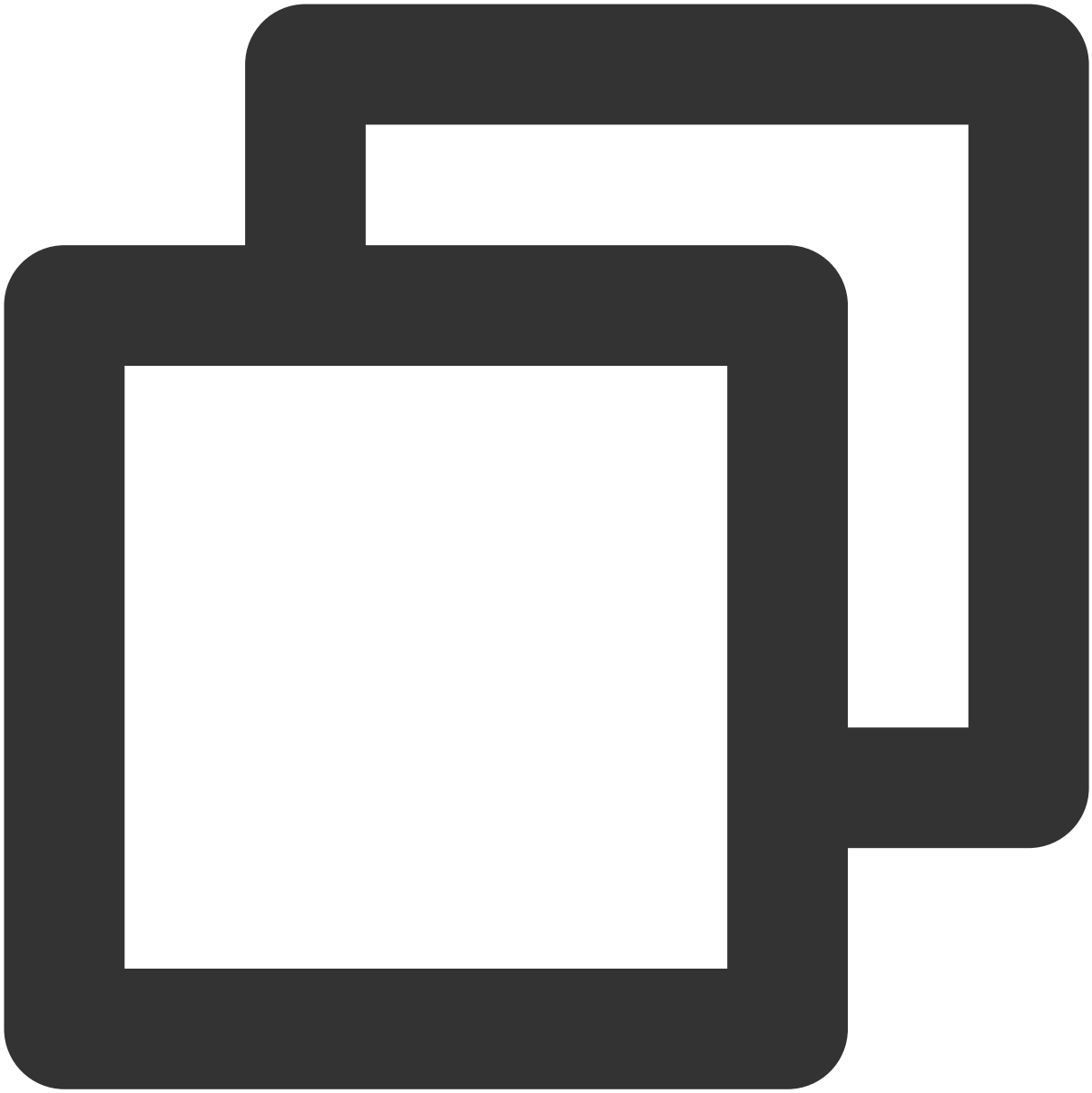
```
python3 -m pip install --user --upgrade setuptools wheel
```

5. Package the project.



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

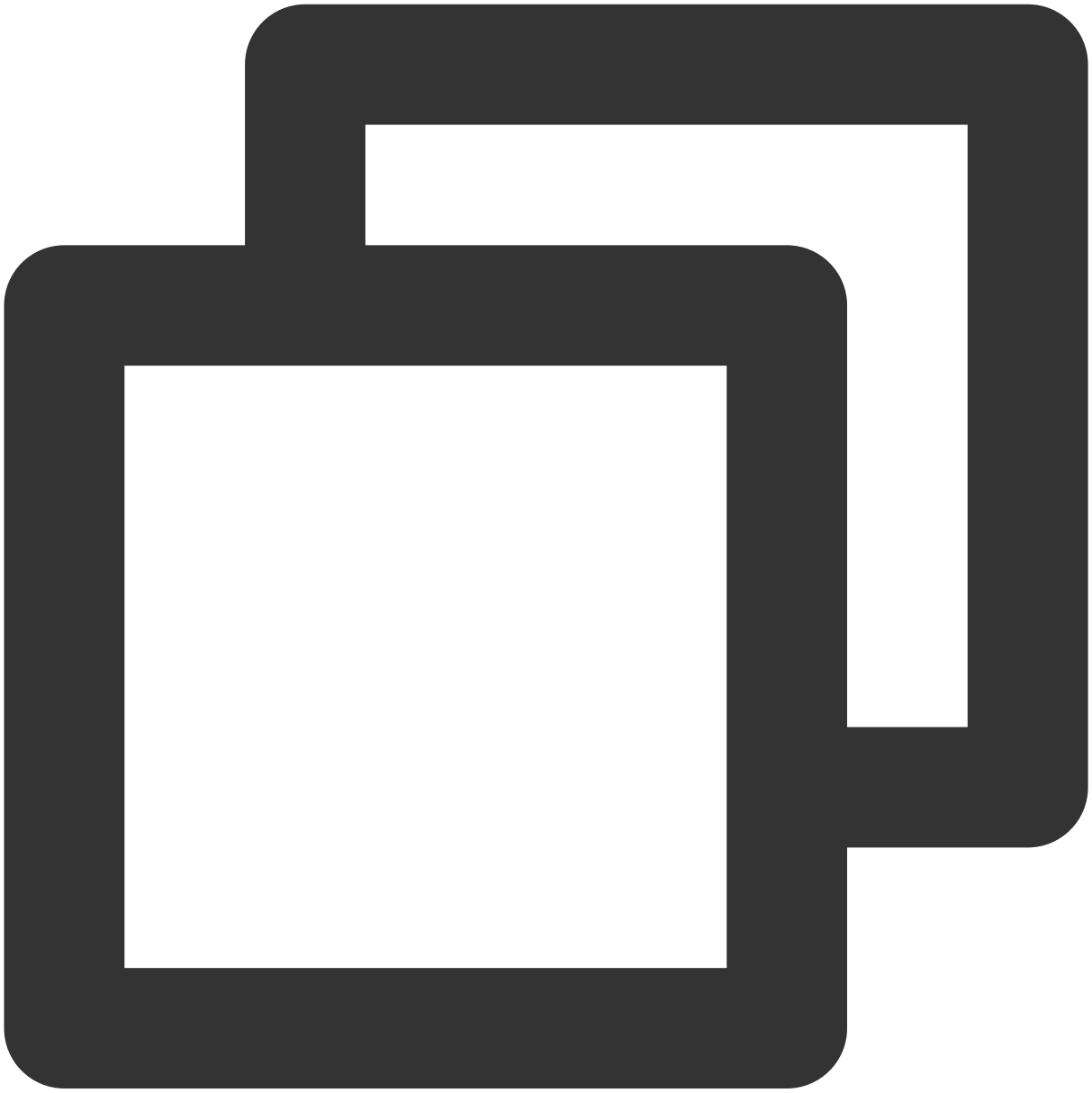
After the project is packaged, the following files will be generated in `/dist` for pushing:



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

Configure Authentication Information

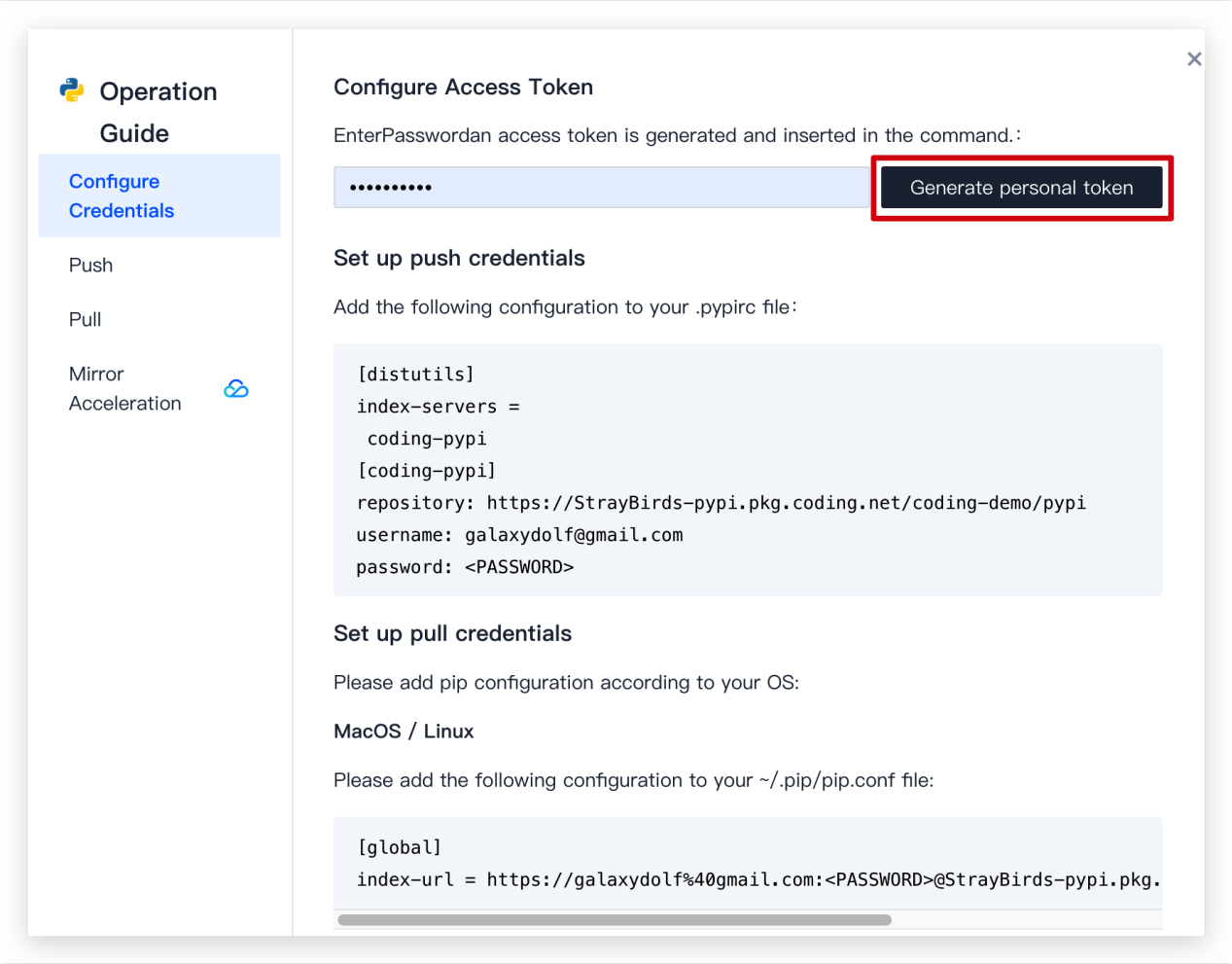
Authentication information must be configured before you can pull artifacts from or push artifacts to CODING-AR. You can select **automatic configuration** or **manual configuration**. Before configuration, run `cd /` to go to the root directory, and then run `ls -a` to check whether `.pypirc` and `pip.conf` files exist.
If such files are not found, run the following command to create these files.



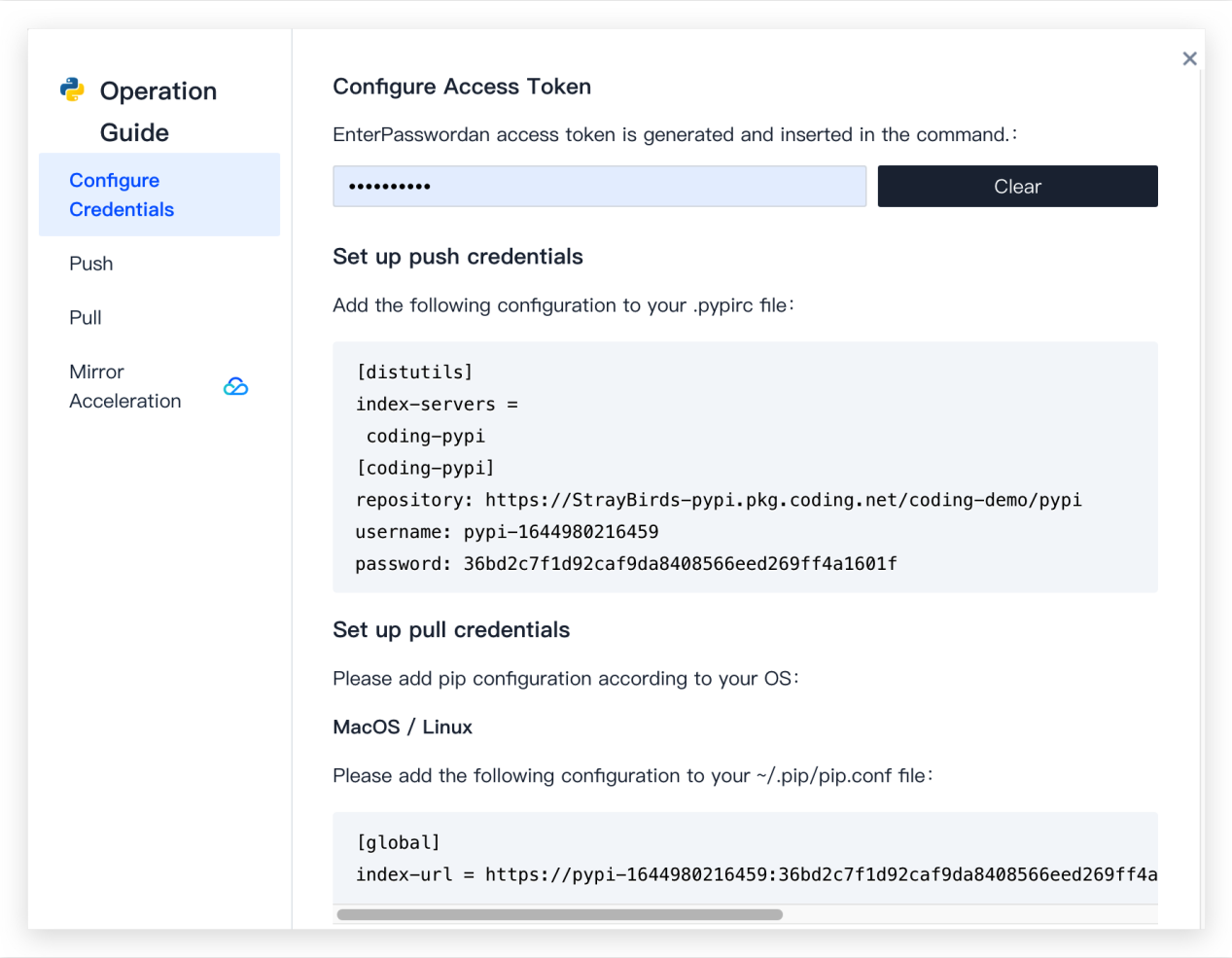
```
touch .pypirc && touch pip.conf
```

Automatic configuration

1. Click **Generate configuration from access token**. An access credential will be generated for you. To view your personal token, go to **Personal Account Settings** > **Access Token**.



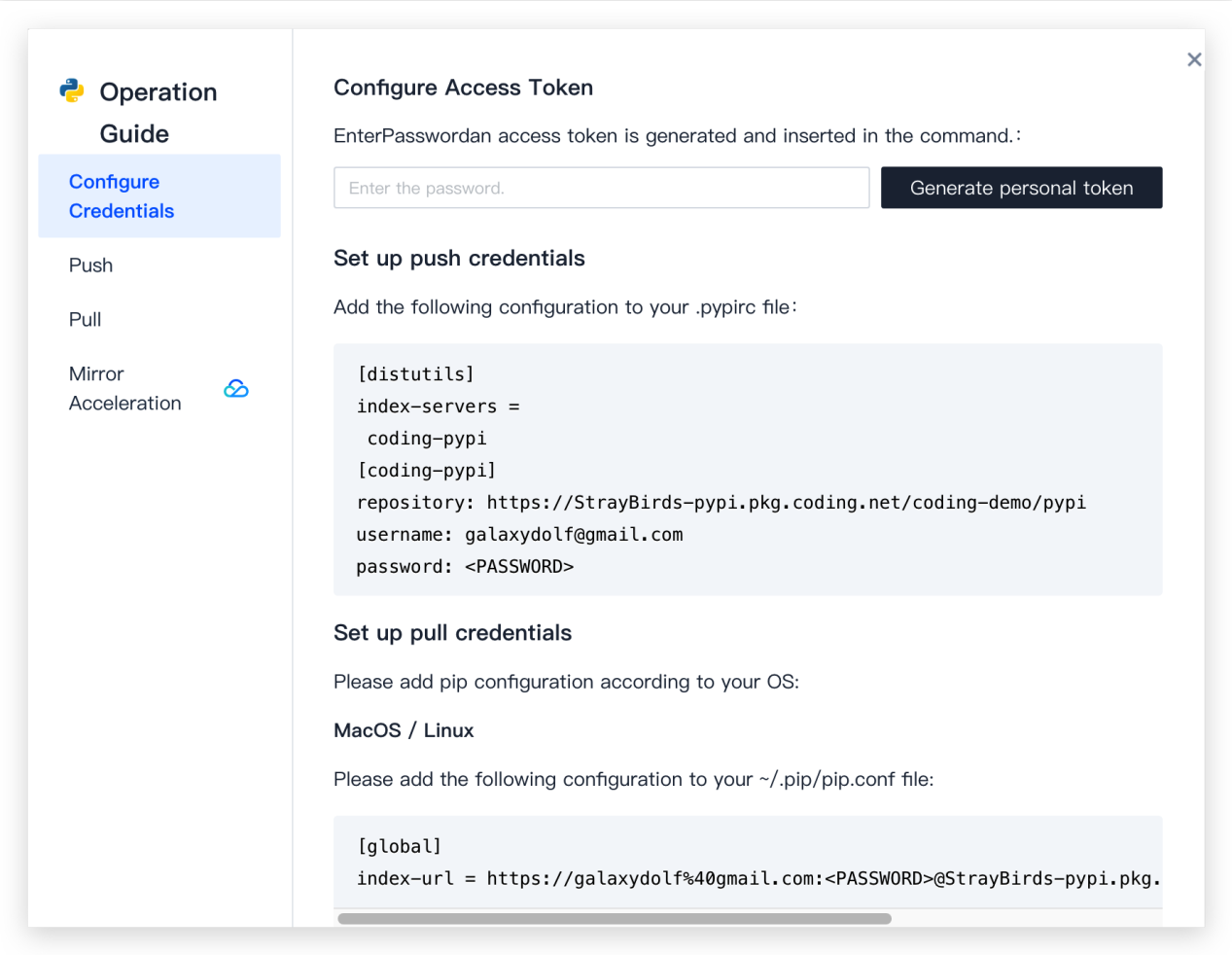
2. Enter your login password to obtain the configuration.



Copy the configuration to the `.pypirc` and `pip.conf` files in the root directory.

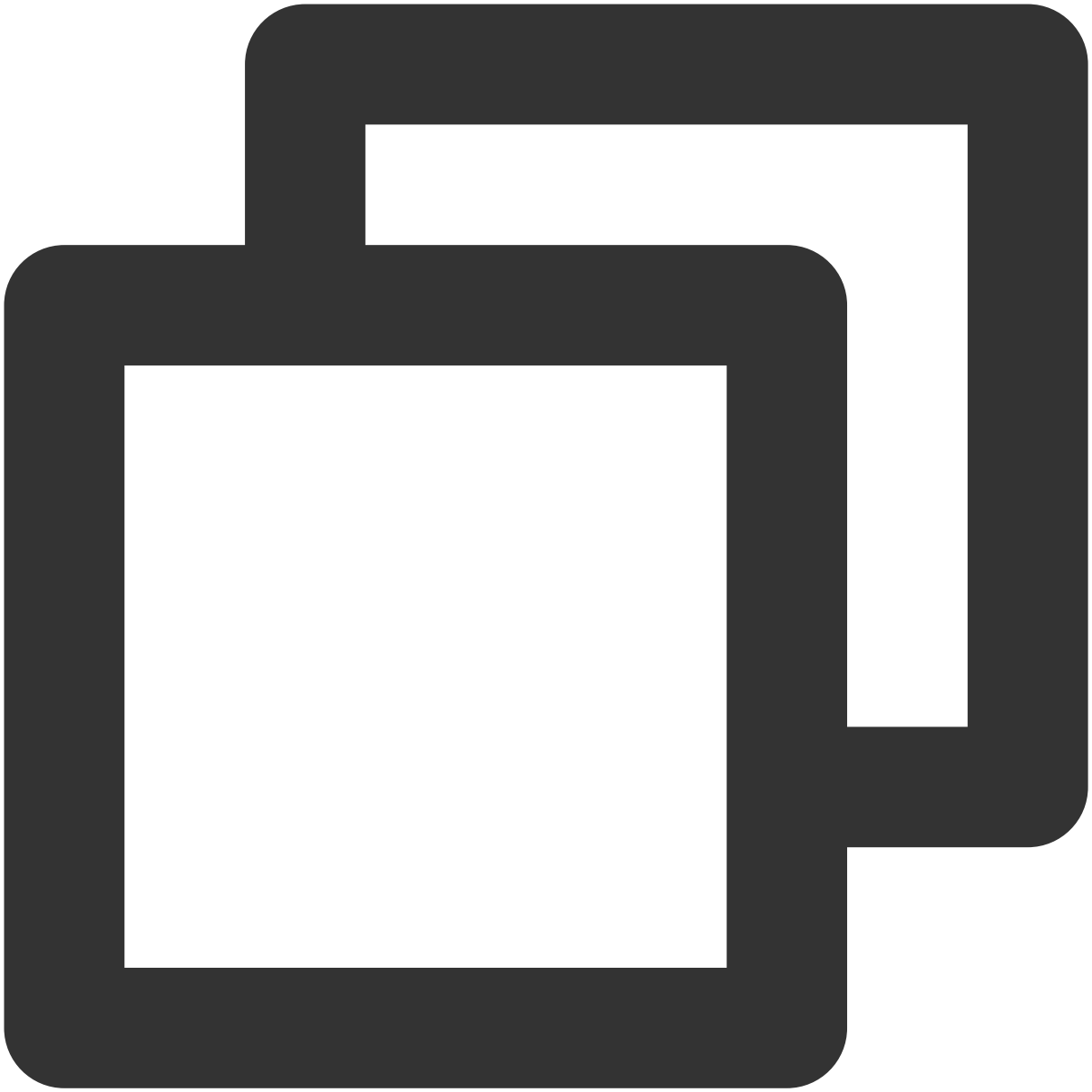
Manual configuration

Copy the code in the guide to the specific files.



Push an Artifact

Go to the project directory and copy and run the command in the terminal. For example, go to the `Demo` directory created above and run the command to push all artifacts in `Demo/dist` to CODING-AR.

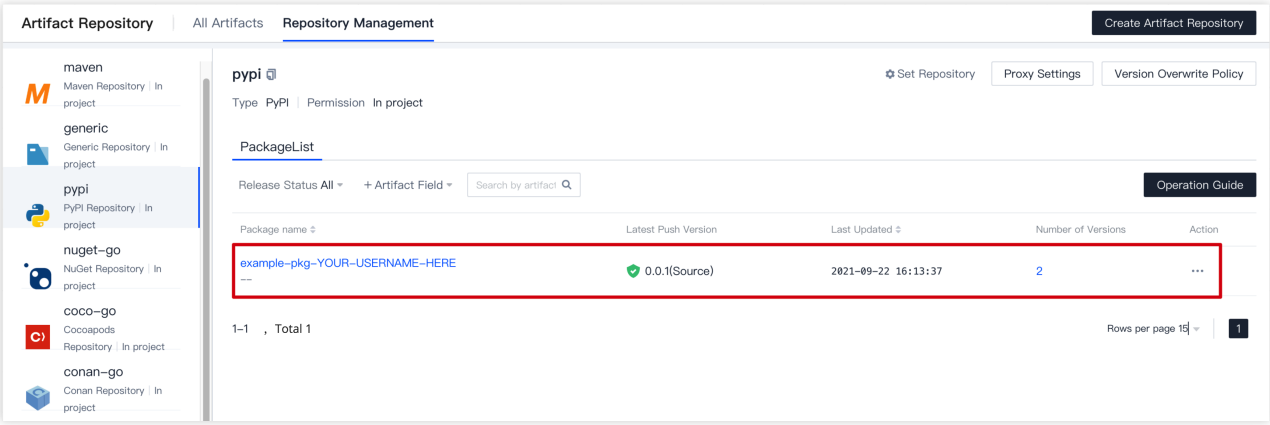


```
twine upload -r coding-pypi dist/*
```

```
/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
```

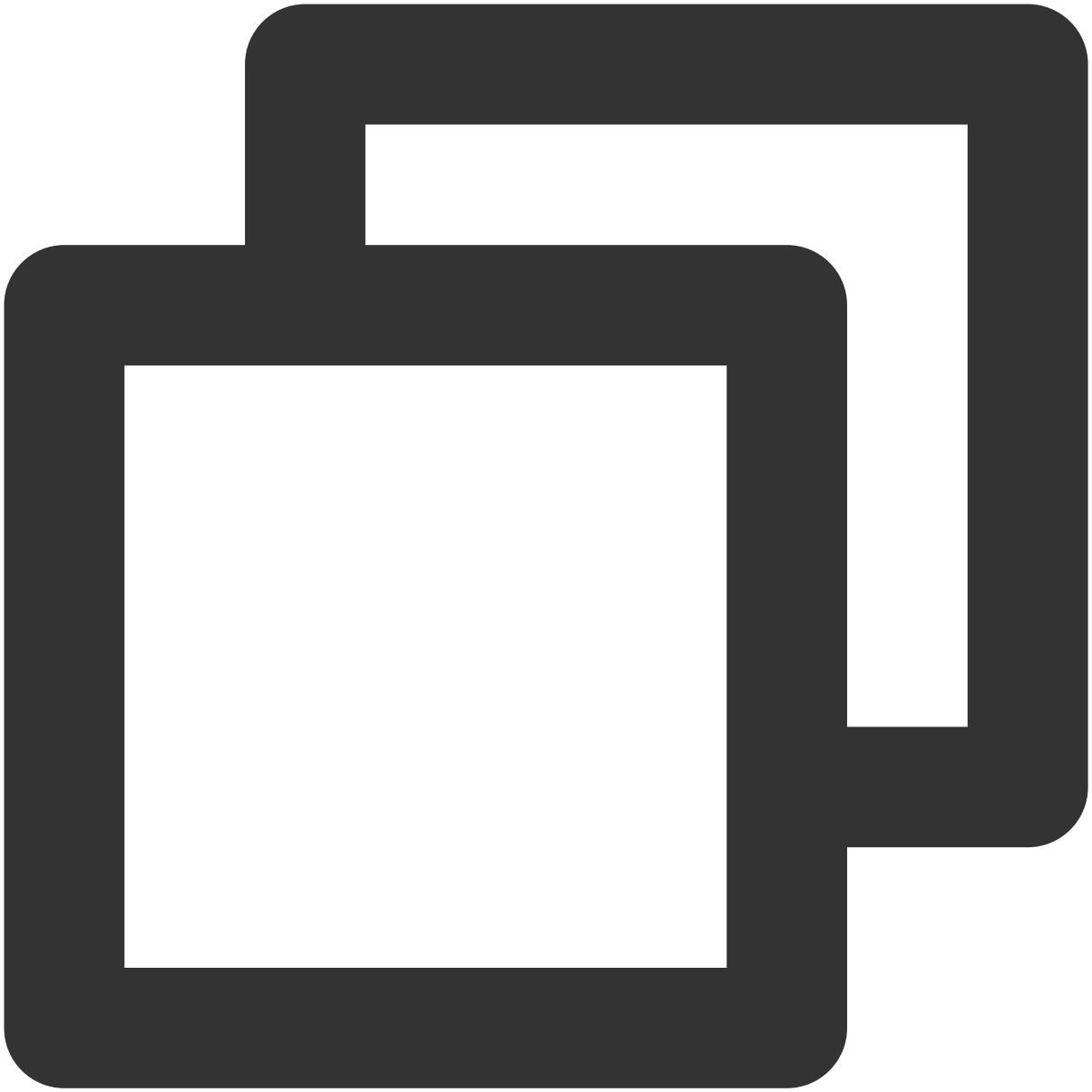
After the artifact is pushed successfully, refresh the page to view the latest artifacts.

</dx-codeblock>



Pull an Artifact

Run the `pip install` command in the PyPI repository guide to pull an artifact.

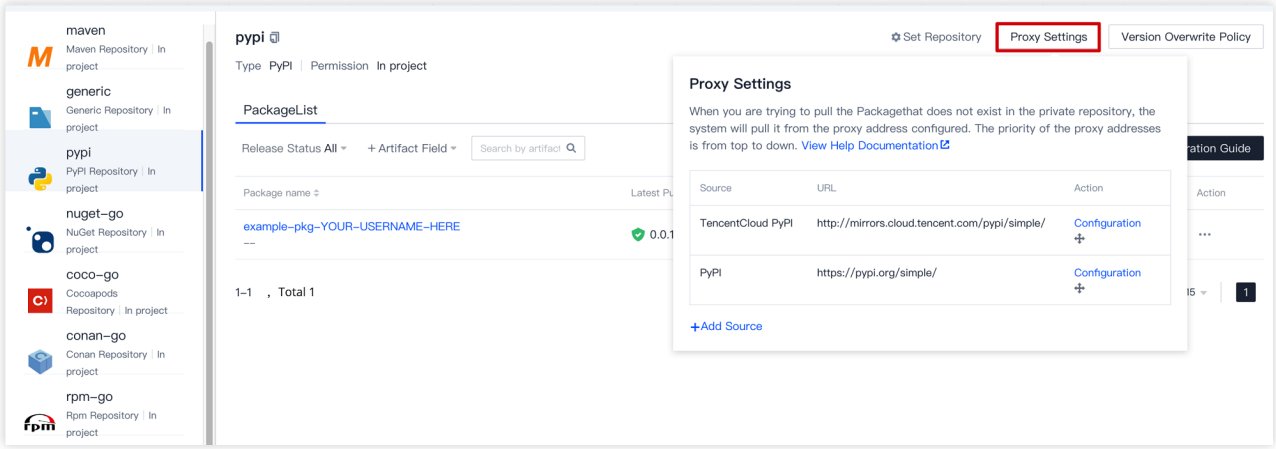


```
pip install <Artifact Name>
```

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.



Replace `<package>` with the package name and run the command generated on the page to pull the package.

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.

Composer Repository

Last updated：2022-03-30 10:08:16

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.

Open CODING-AR

1. Log in to the CODING Console and click **Use Now** to go to CODING page.
2. Click  in the upper-right corner to open the project list page and click a project icon to open the project.
3. In the menu on the left, click **Artifact Management**.

Preparations

Note：
Before you begin:

- Install Composer.
- Create an artifact repository (see Basic Operations).
- Select Composer as the repository type.

Create a Composer Package (Optional)

Install Composer

Run the following command to install Composer.

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

Add an environment variable for global access.

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

Initialize

Create a demo directory.

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

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

```
composer inits
```

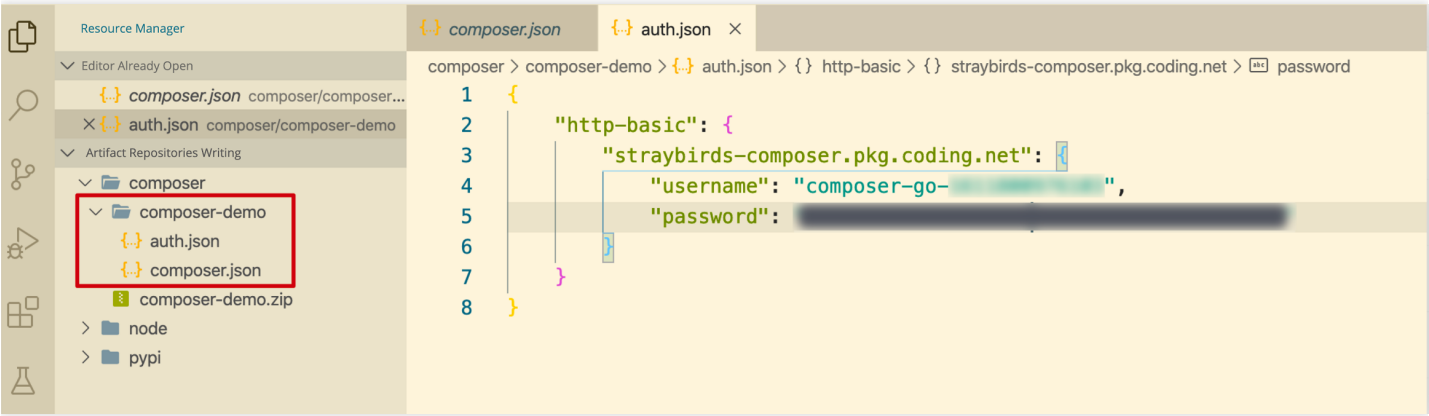
After initialization, a `composer.json` configuration file will be added to the directory.

Configure the Authentication File

Go to the directory where the Composer package is located and create an `auth.json` file. You can select manual or automatic configuration.

Automatic configuration

Click **Generate configuration from access token** and then copy the content to the `auth.json` file.

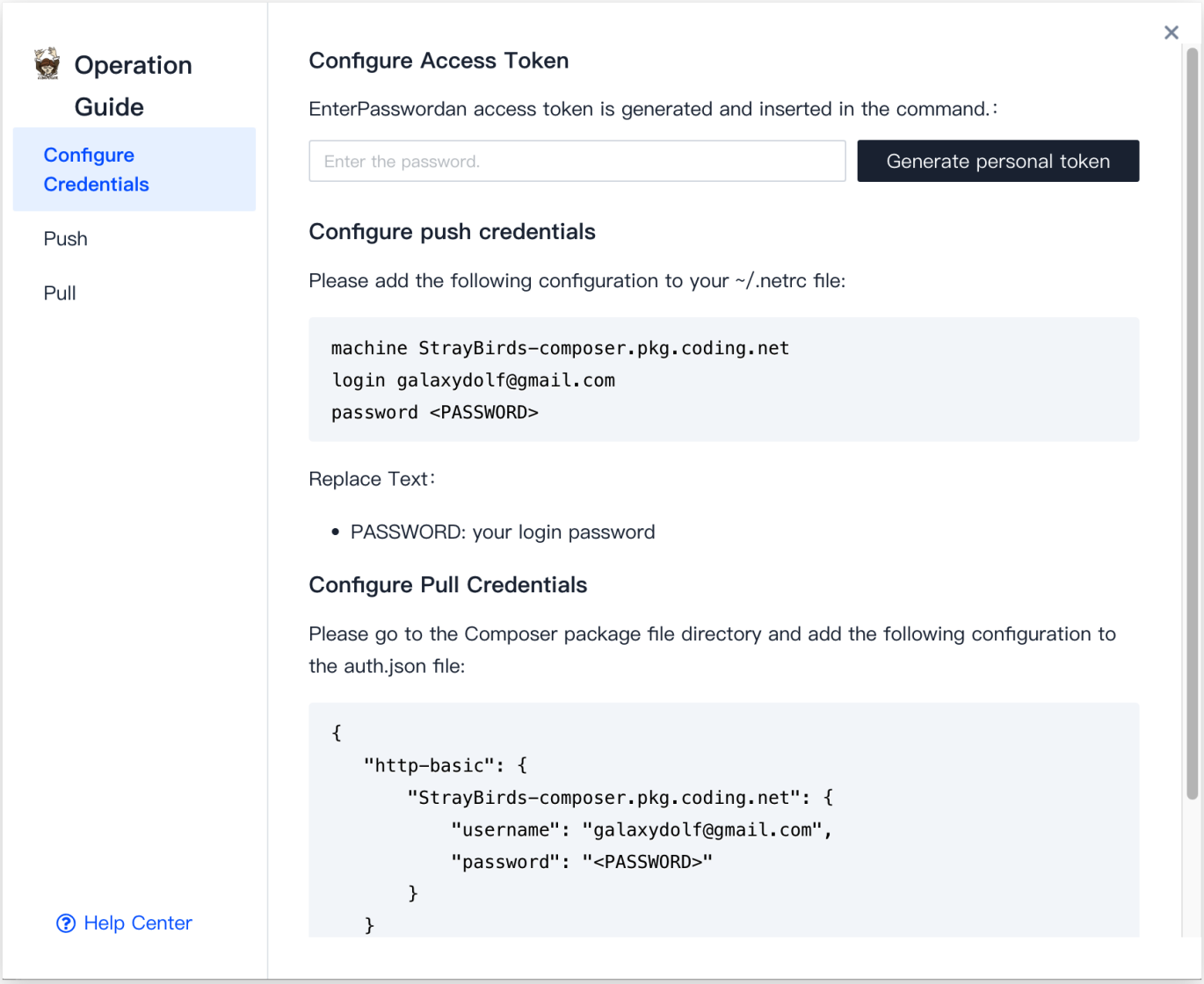


Manual configuration

Copy the commands on the webpage and replace `[PASSWORD]` with your service password.

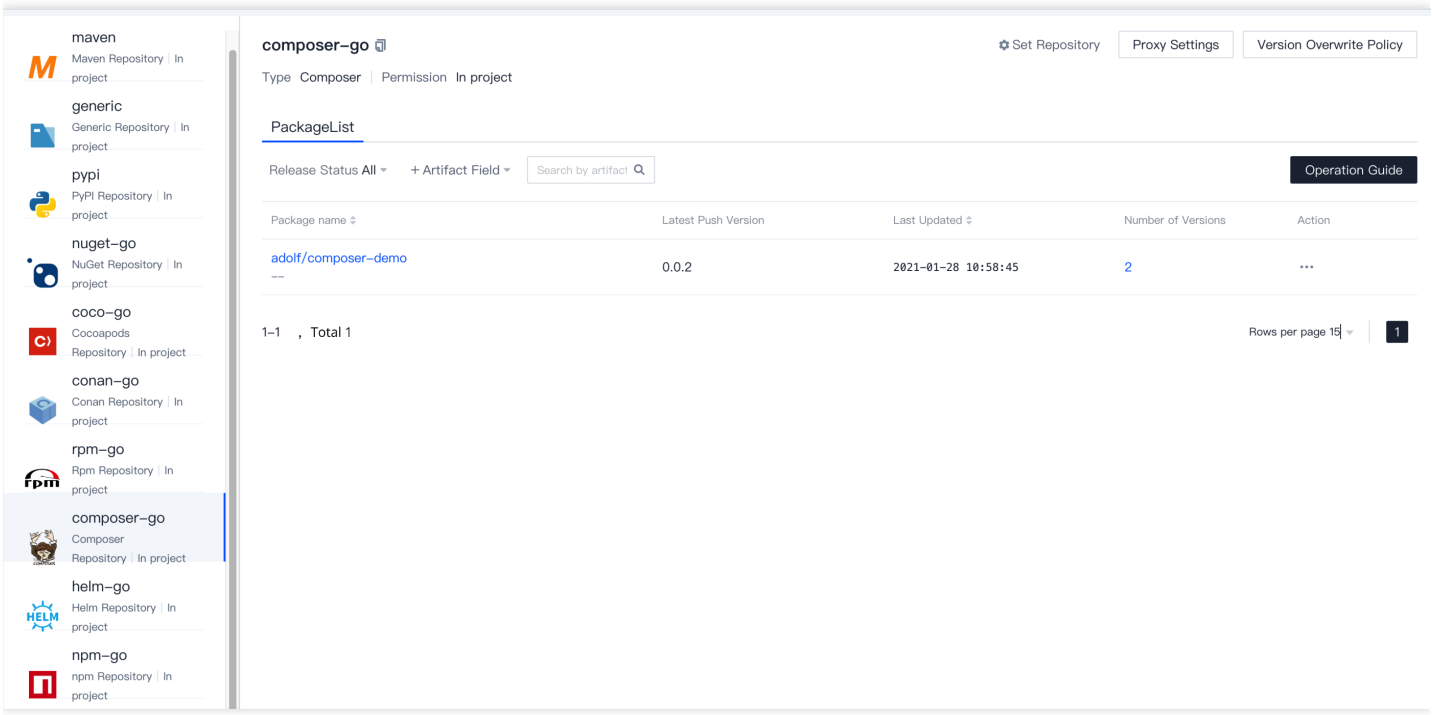
Push an Artifact

Run the commands on the CODING-AR page to compress Composer package files into a `zip` package and then push it to the repository using tools such as `cURL` .



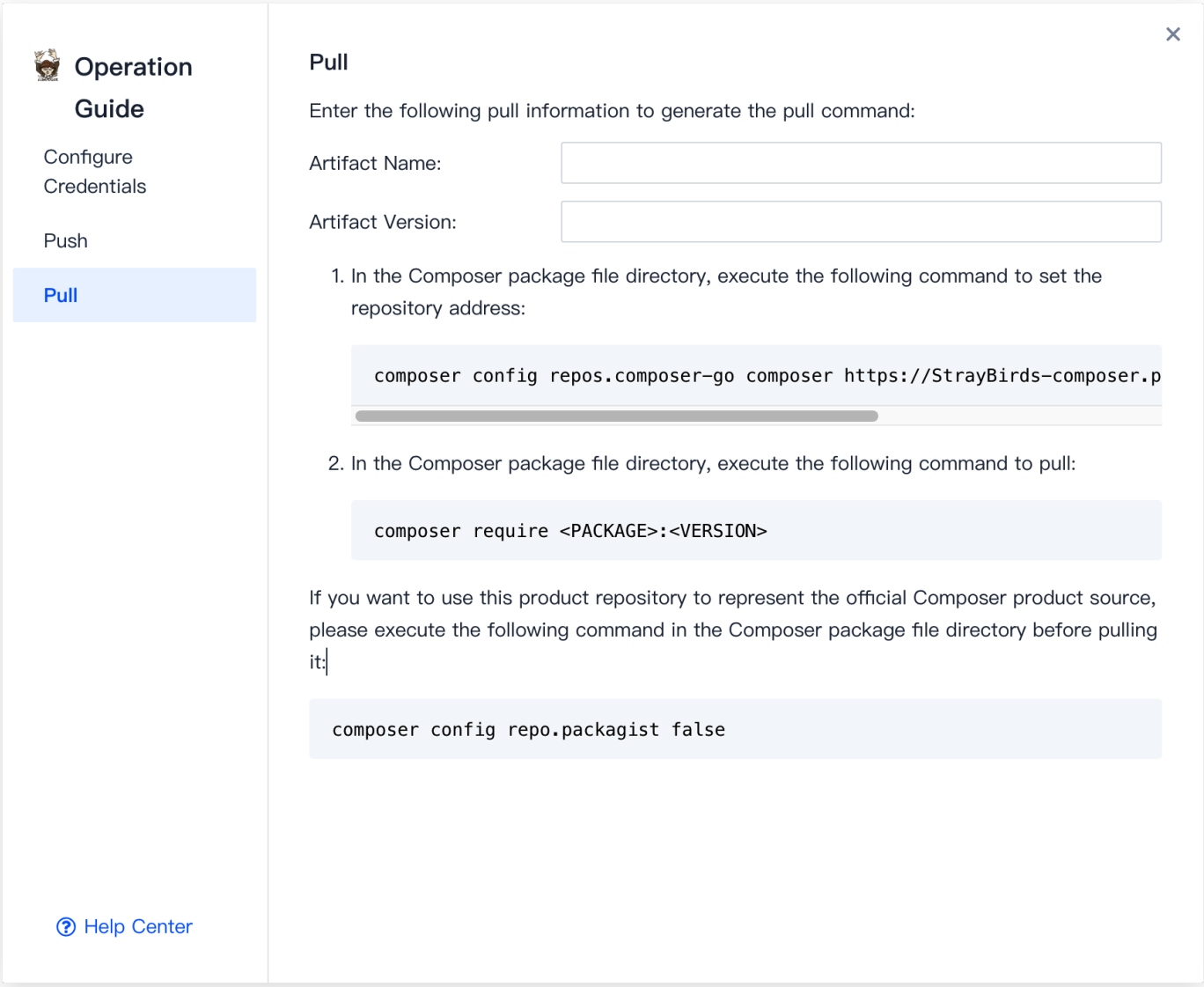
Enter the password configured in `auth.json` and replace `<version>` with the specific version number.

After the push is successful, you can view the package in CODING-AR.



Pull an Artifact

Go to the Composer package directory and set the repository URL. You can use the commands in CODING.



Replace the Composer source with the CODING repository (optional).

```
composer config repo.packagist false
```

Replace `` with the package to be pulled and run the pull command.

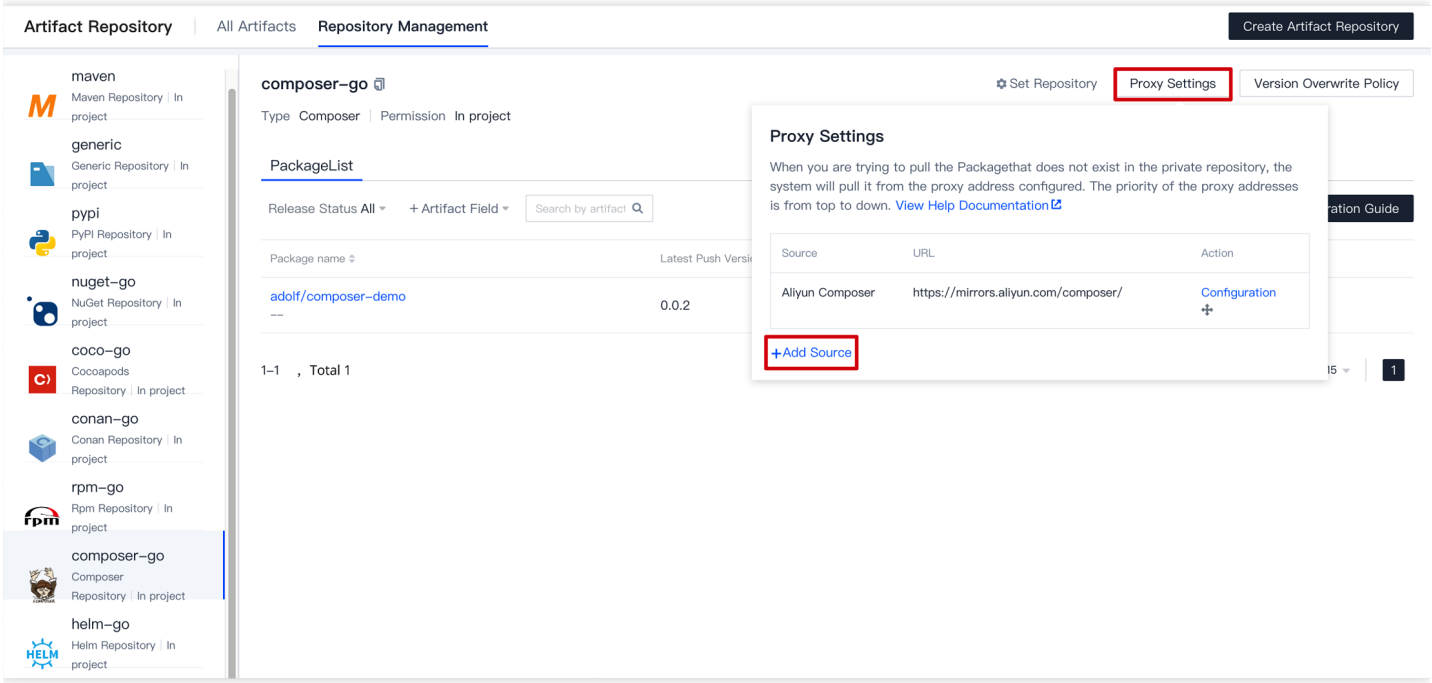
```
composer require < PACKAGE >:< VERSION >
```

Enter your service email address and password to pull the package.

```
/Volumes/CODINC[redacted]F/testing composer require adolf/composer-demo:0.0.2        Authentication required (straybirds-composer.pkg.coding.net):
      Username: [redacted]@qq.com
      Password:
https://straybirds-composer.pkg.coding.net/coding-demo/composer-go could not be fully loaded (Invalid credentials for 'https://straybirds-composer.pkg.coding.net/coding-demo/composer-go/packages.json', aborting.), package information was loaded from the local cache and may be out of date
./composer.json has been updated
Running composer update adolf/composer-demo
Loading composer repositories with package information
    Authentication required (straybirds-composer.pkg.coding.net):
      Username: [redacted]@qq.com
      Password:
Do you want to store credentials for straybirds-composer.pkg.coding.net in /Users/adolf/.composer/auth.json ? [Yn] y
Updating dependencies
Lock file operations: 1 install, 0 updates, 0 removals
  - Locking adolf/composer-demo (0.0.2)
Writing lock file
Installing dependencies from lock file (including require-dev)
Package operations: 1 install, 0 updates, 0 removals
  - Downloading adolf/composer-demo (0.0.2)
  - Installing adolf/composer-demo (0.0.2): Extracting archive
Generating autoload files
```

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.



NuGet Repository

Last updated：2022-03-30 10:08:16

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.

Open CODING-AR

1. Log in to the CODING Console and click **Use Now** to go to CODING page.
2. Click  in the upper-right corner to open the project list page and click a project icon to open the project.
3. In the menu on the left, click **Artifact Management**.

Preparations

Note：
Before you begin:

- Install NuGet.
- Create an artifact repository (see Basic Operations).
- Select NuGet as the repository type.

Initialize a NuGet Artifact (Optional)

Visit the NuGet website to and 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
```

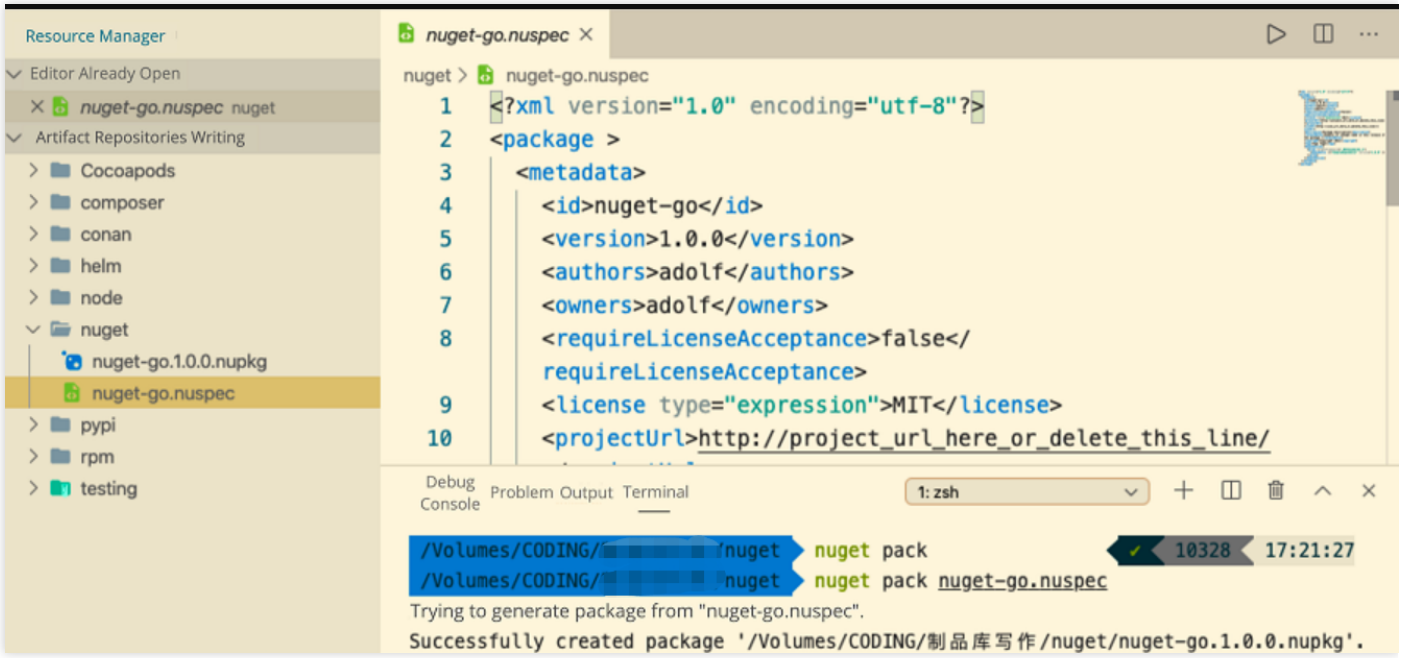
2. Create a `.nuspec` package.

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

3. Package the artifact.

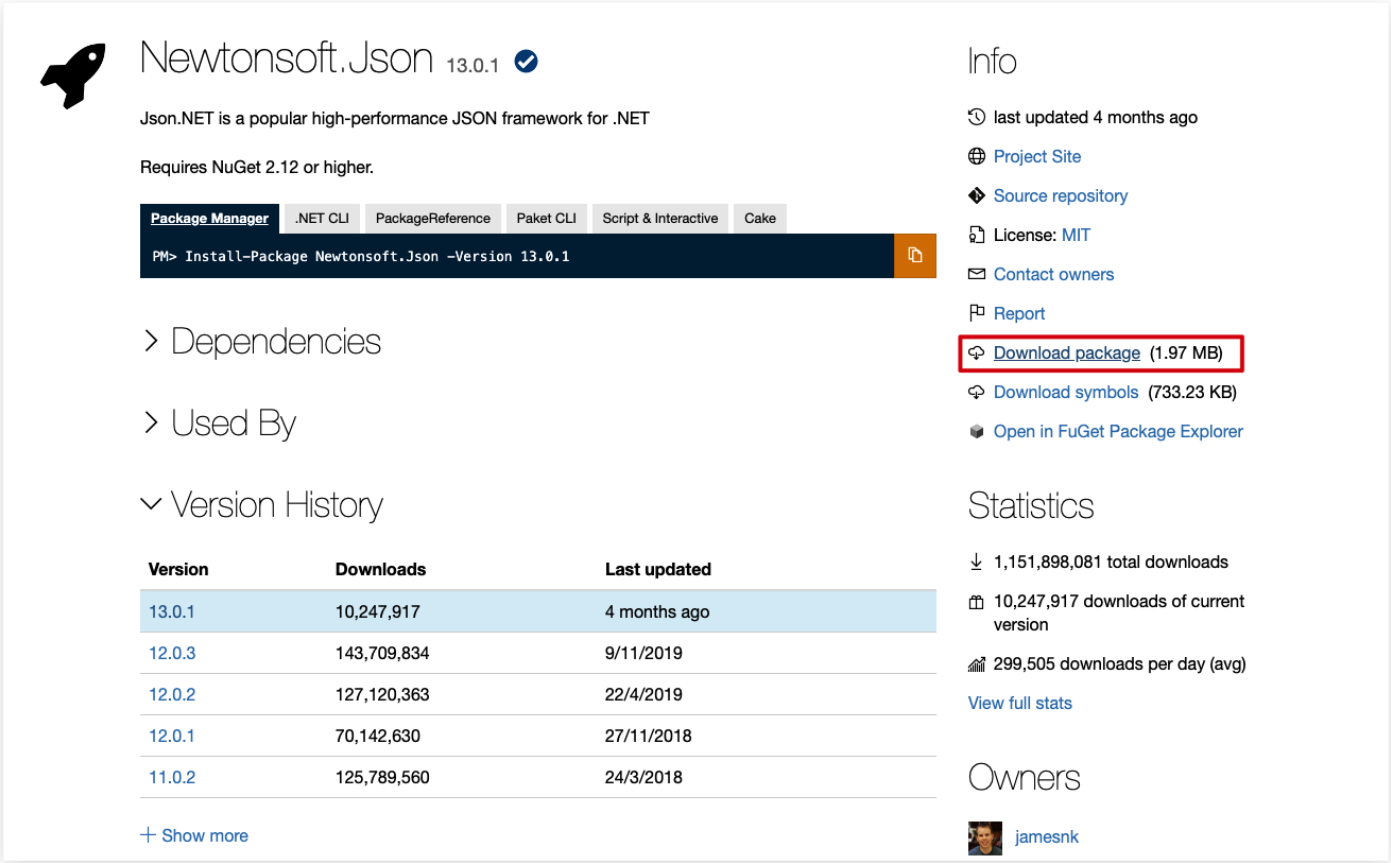
```
nuget pack <artifact name="">.nuspec
```

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



Pull an artifact from an online source

Visit the NuGet website, search for a NuGet artifact, and download it or pull it via the command line.



Pull an artifact via the command line:

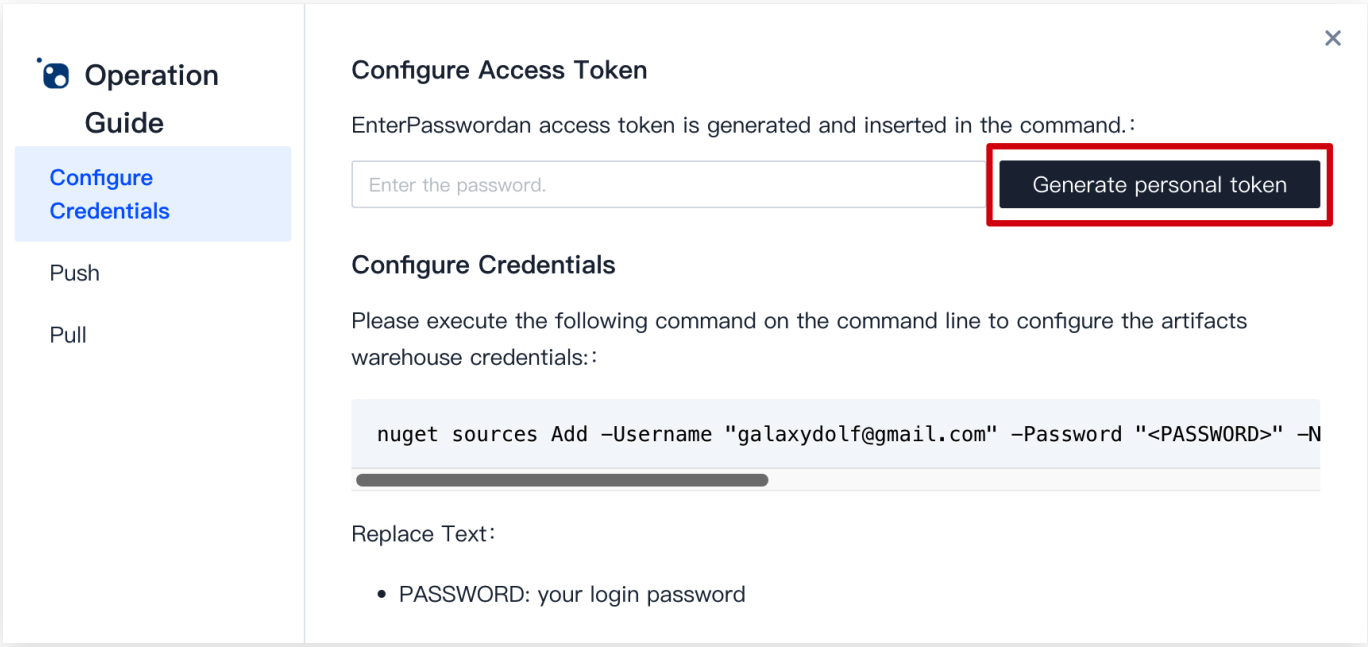
```
nuget install [Artifact Name] -OutputDirectory packages
```

Configure Authentication Information

Authentication information must be configured before you can pull artifacts from or push artifacts to CODING-AR. The following example uses **automatic configuration**.

Click **Generate configuration from access token** and enter your password to obtain the configuration command. Then, copy and run the command in the directory where the NuGet artifact is stored. During this process, **permission**

control is implemented with the **personal access token**.



Push an Artifact

Replace the variables and run the command push an artifact.

```
nuget push -ApiKey api -Source [Repository name in the push guide] [Local artifa
ct name].nupkg
```

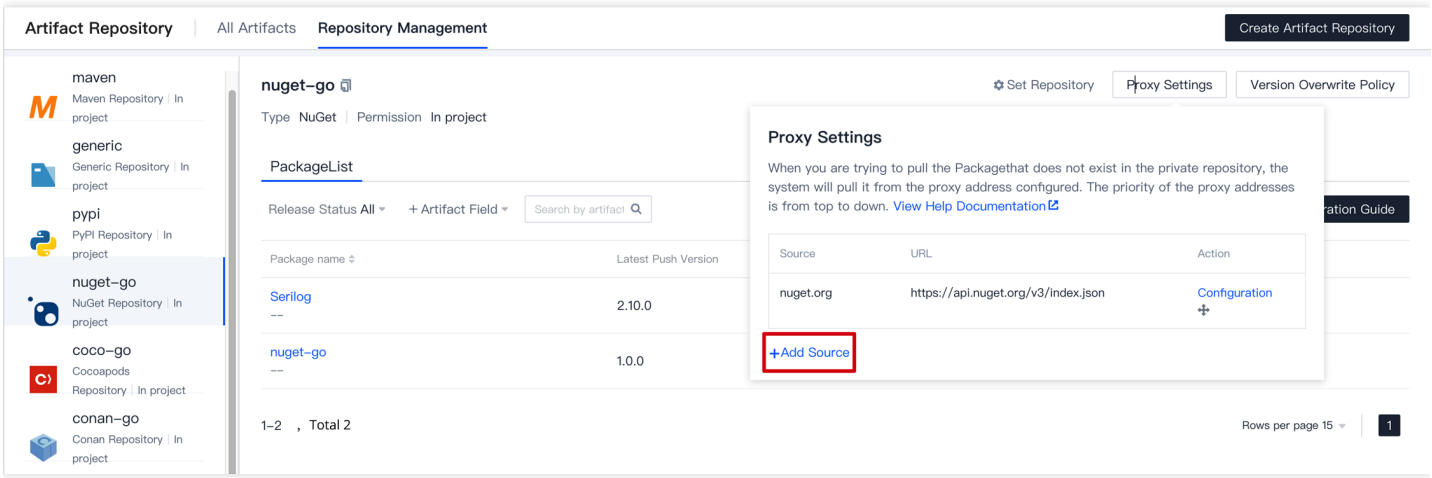
Pull an Artifact

Replace the variables and run the command pull an artifact.

```
nuget install -Source [Repository name in the pull guide] -Version [Artifact Ver
sion] [Artifact Name]
```

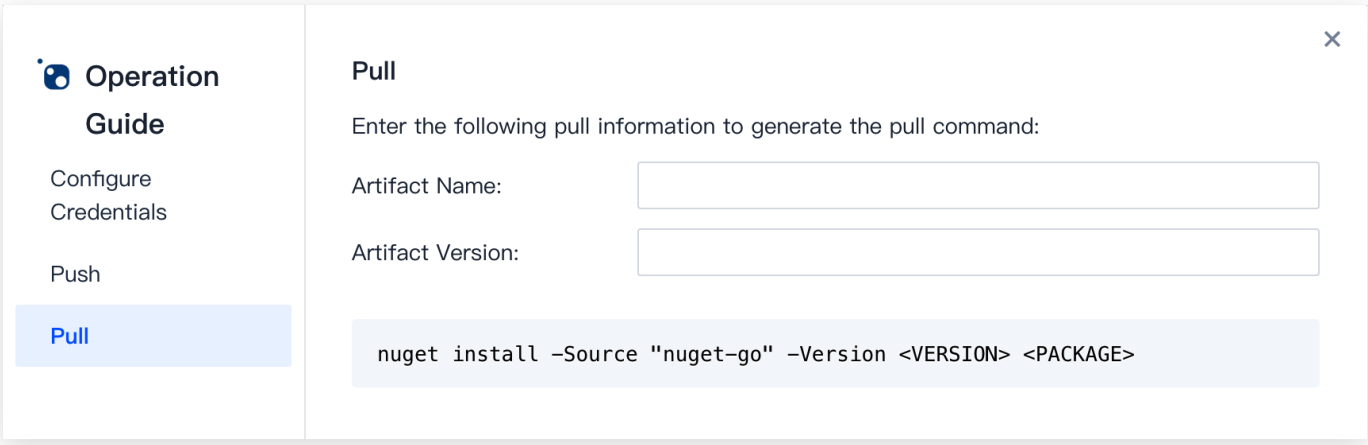
Configure a Proxy

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.



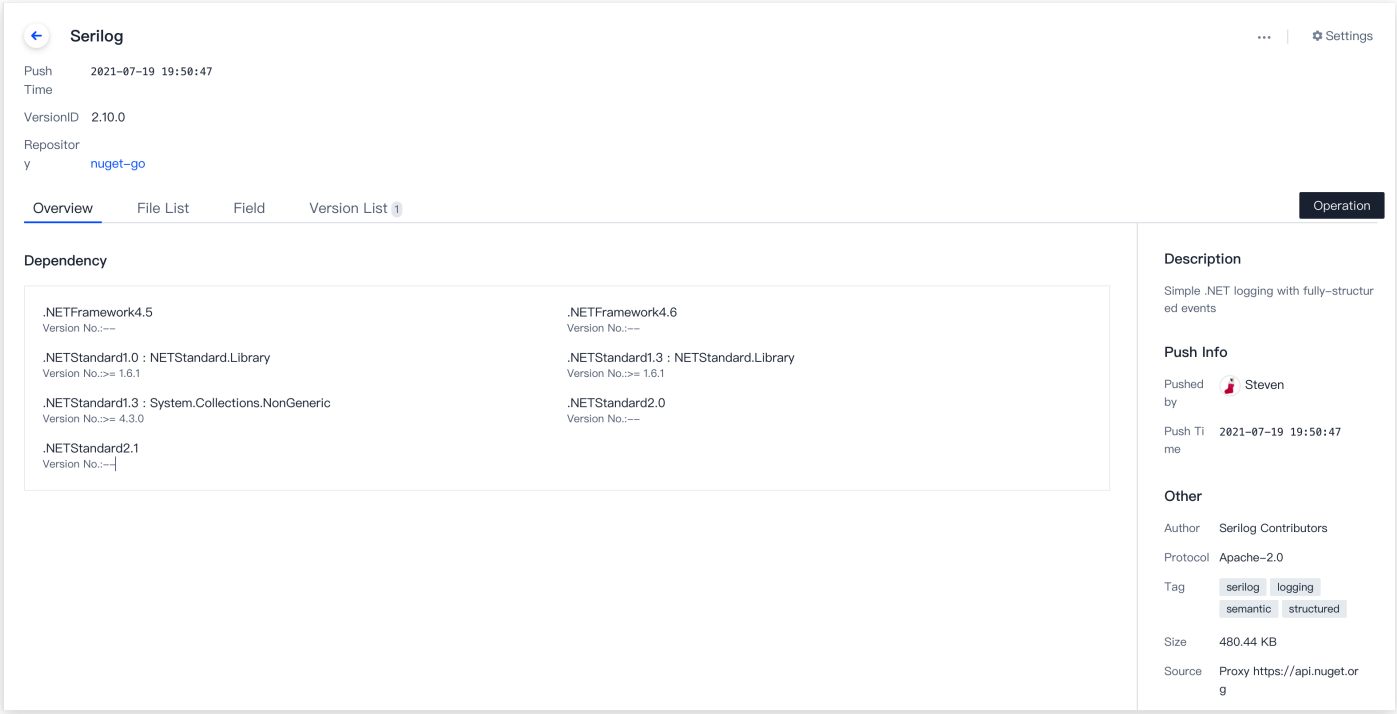
Run the following command to pull an artifact:

```
nuget install -Source [Repository Name] -Version [Artifact Version] [Artifact Name]
```



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.



Note

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.

RPM Repository

Last updated：2024-01-02 11:12:38

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.

Open CODING-AR

1. Log in to the CODING Console and click **Use Now** to go to CODING page.
2. Click  in the upper-right corner to open the project list page and click a project icon to open the project.
3. In the menu on the left, click **Artifact Management**.

Preparations

Note:

Before you begin:

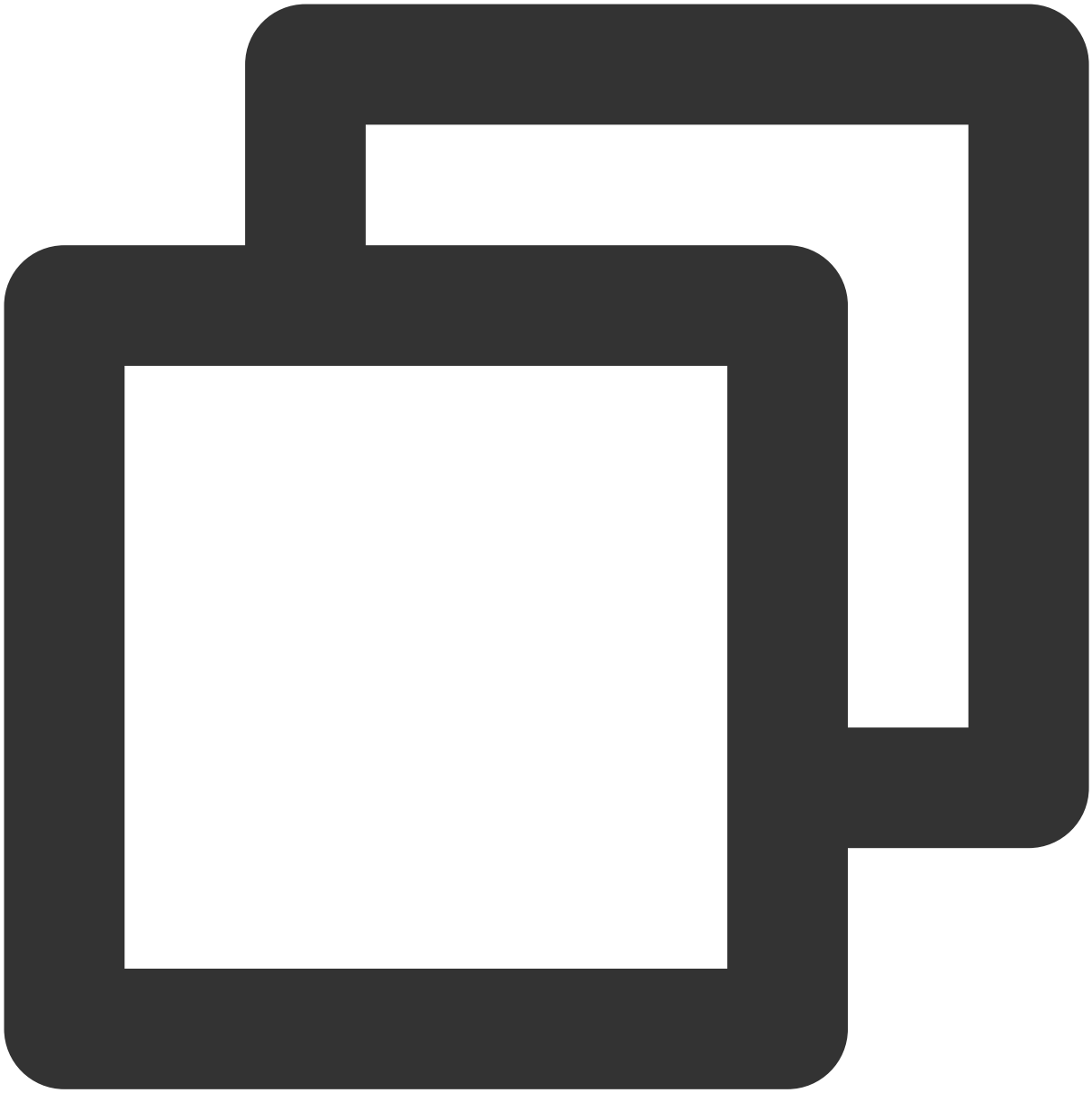
Prepare a Linux environment.

Create an artifact repository (see Basic Operations).

Select RPM as the repository type.

Initialize a Local RPM Project

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:

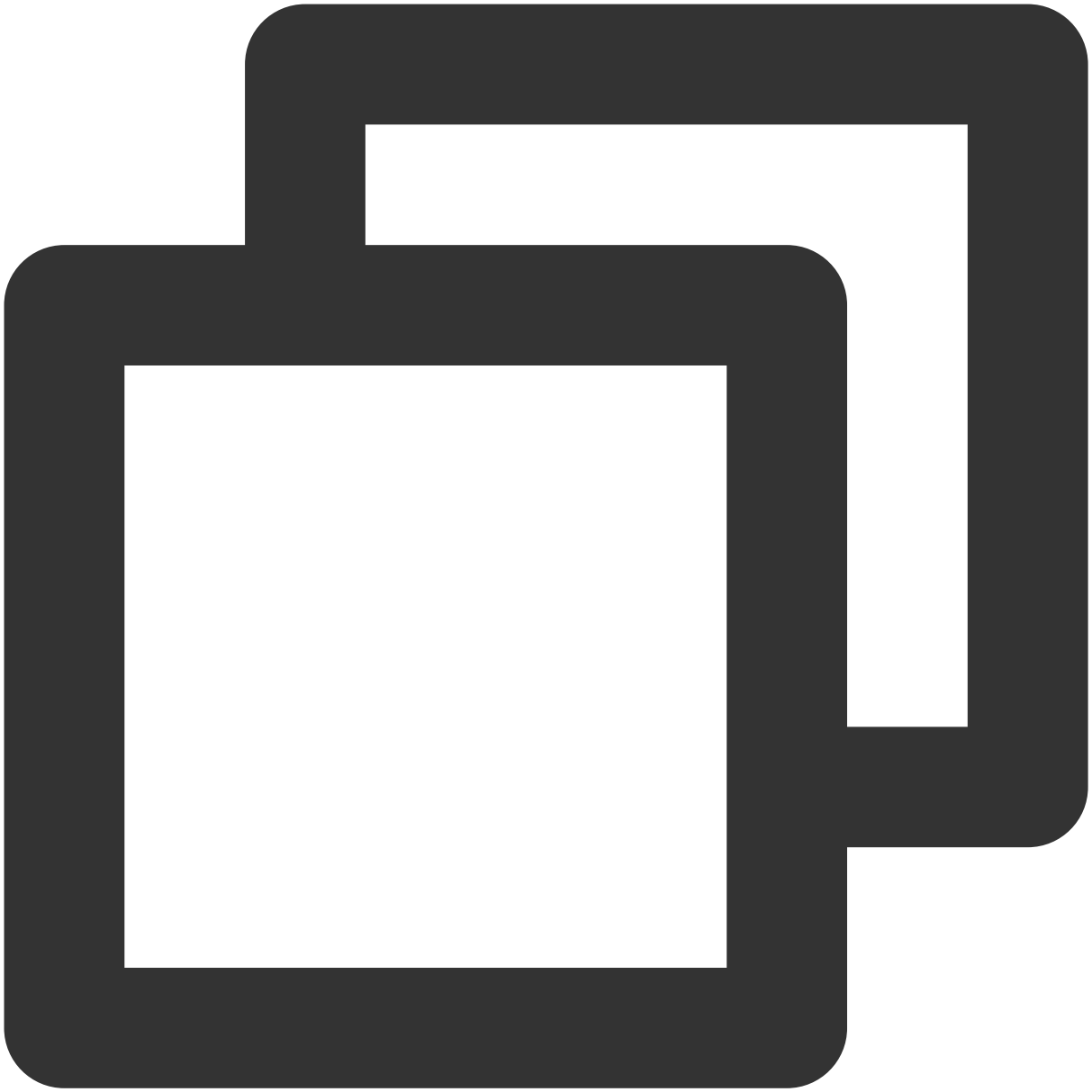


```
docker run -it --name centos centos:8 /bin/bash
```

Download a Demo Project

Visit the RPM artifact website to search for an artifact and download and install it.

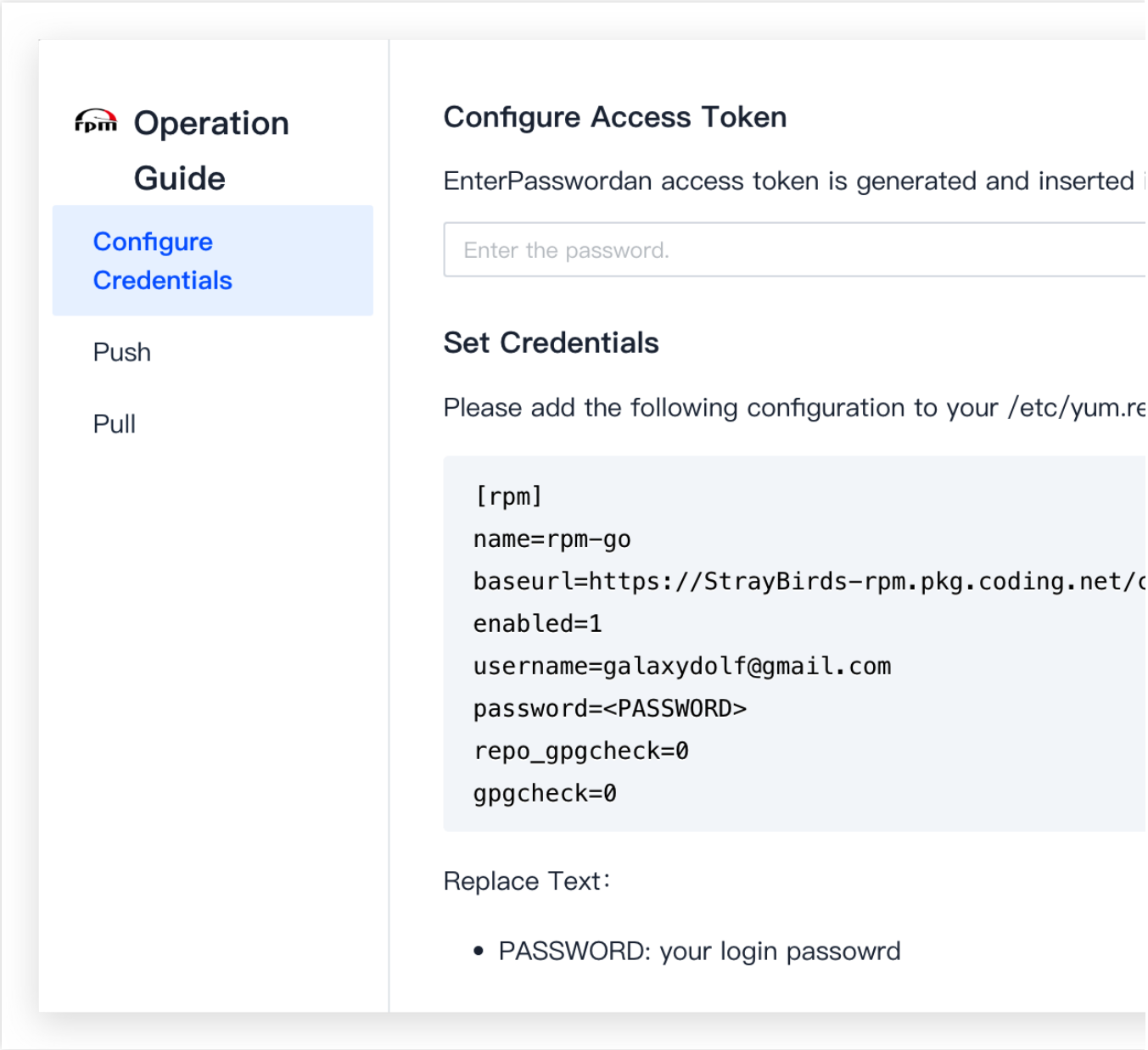
For example:



```
wget -N --no-check-certificate "https://www.rpmfind.net/linux/fedora/linux/developm
```

Configure Authentication Information

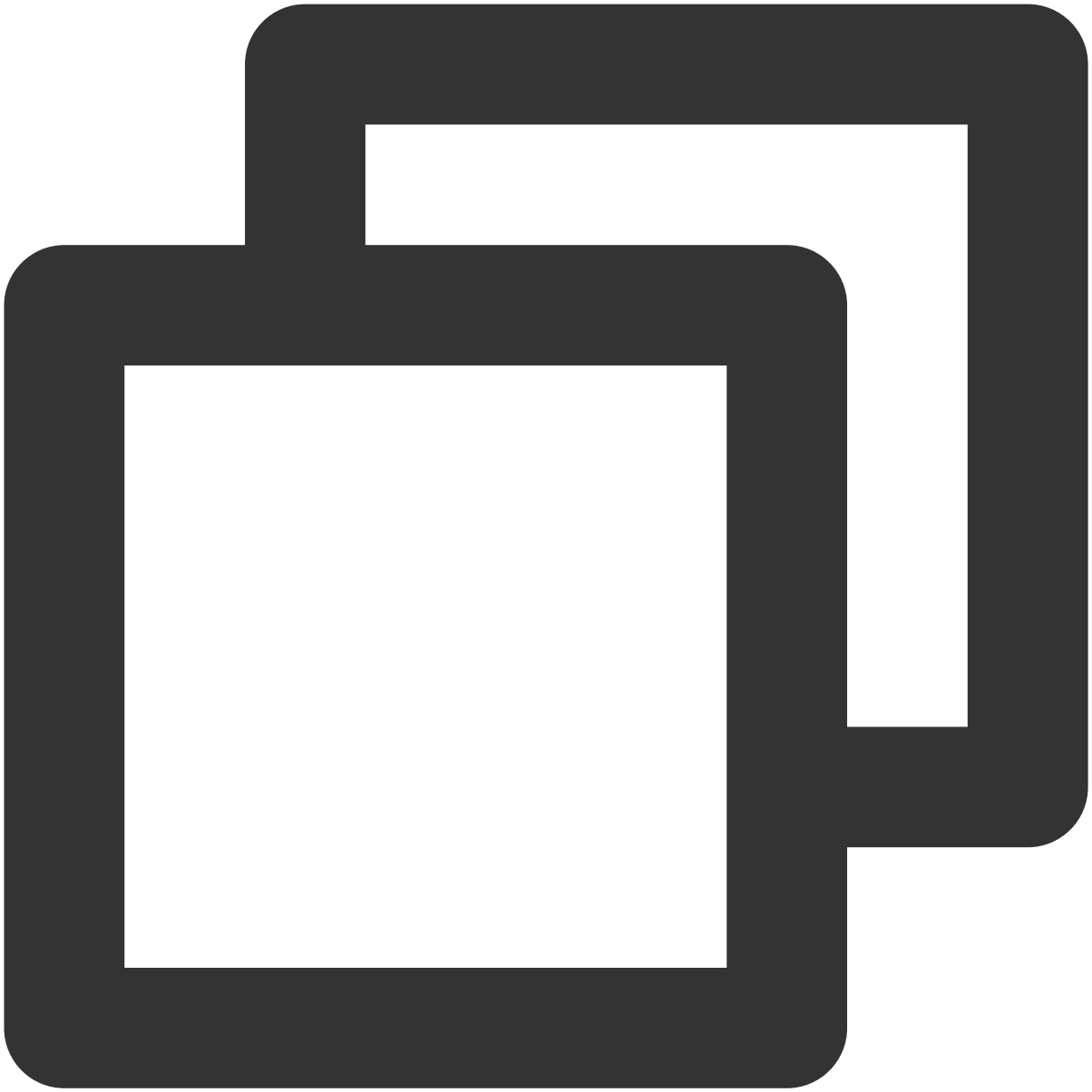
Click **Generate configuration from access token** in "Guide". A personal token will be generated as your access credential. You can manage your personal token in **Personal Account Settings** > **Access Token**.



After entering your login password, copy the configuration generated to the local `/etc/yum.repos.d/rpm-go.repo` file. If this file does not exist, create one.

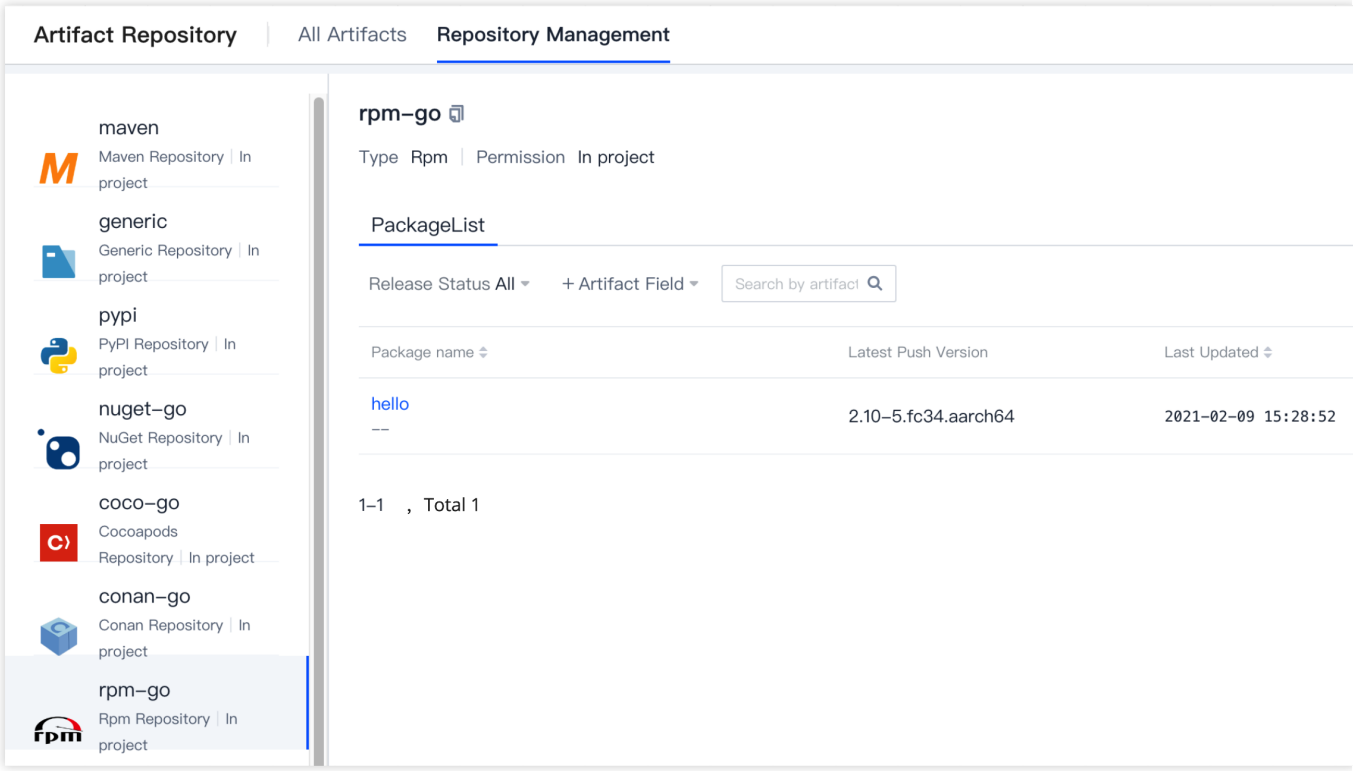
Push an Artifact

Run the rpm publish command to push an RPM package.



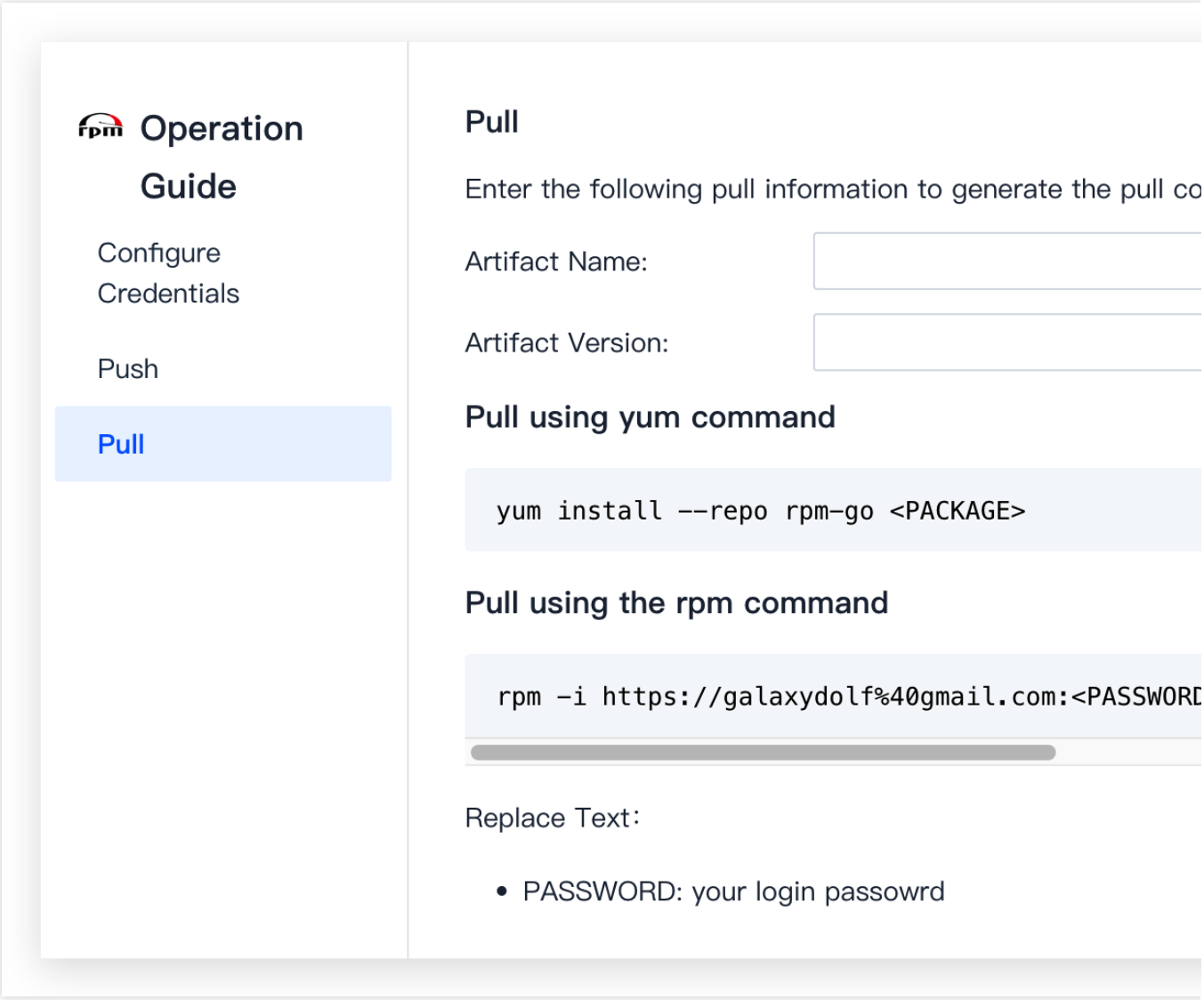
```
curl -u [Username/Email] -X POST [Repository URL in the Push Guide] -T [Artifact Na
```

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



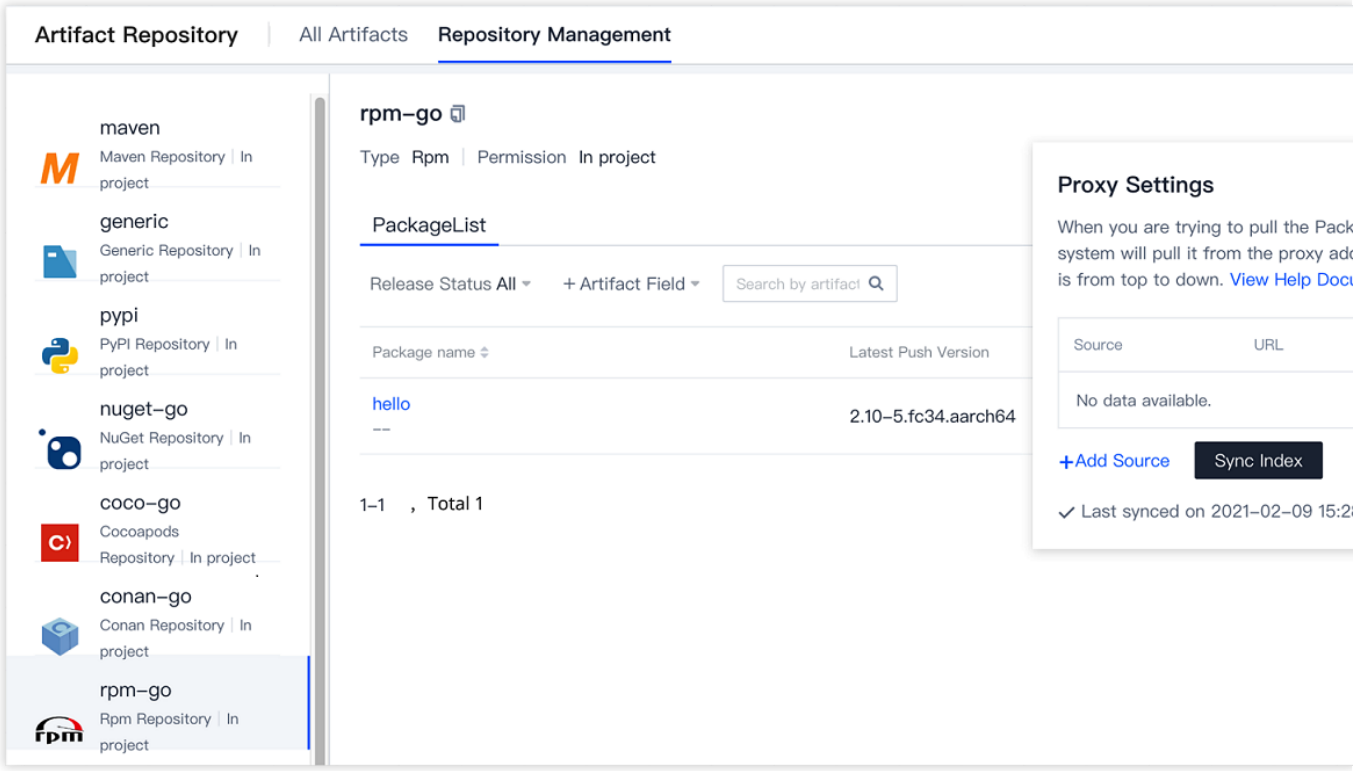
Pull an Artifact

Run the command in the guide to pull an artifact.



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.

Note:

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

Whether you have the permission to push artifacts to this repository.

Whether the artifacts already exist in your local cache.

Conan Repository

Last updated：2024-01-03 11:31:03

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.

Open CODING-AR

1. Log in to the CODING Console and click **Use Now** to go to CODING page.
2. Click

in the upper-right corner to open the project list page and click a project icon to open the project.
3. In the menu on the left, click **Artifact Management**.

Preparations

Note:

Before you begin:

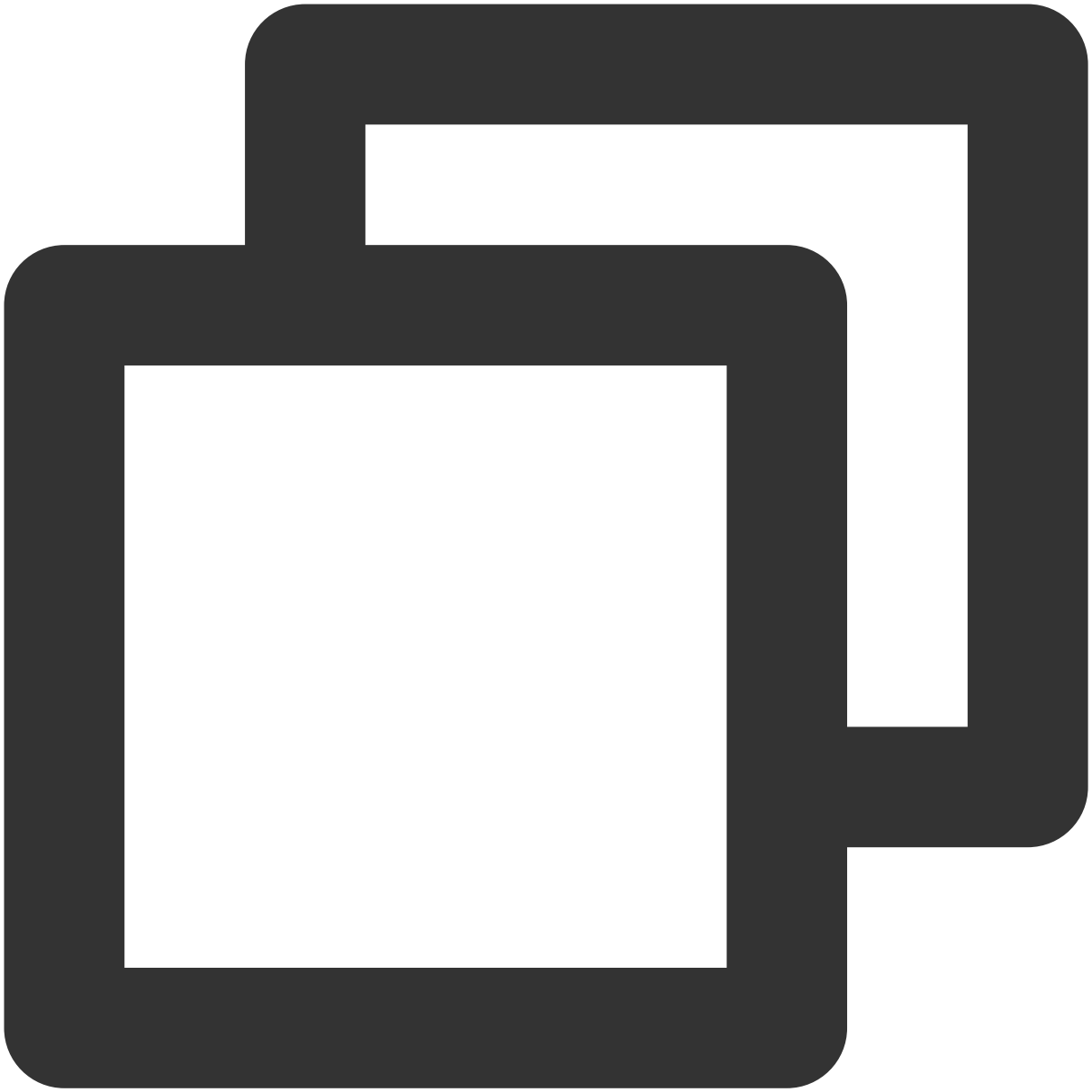
Install Python3.

Create an artifact repository (see Basic Operations).

Select Conan as the repository type.

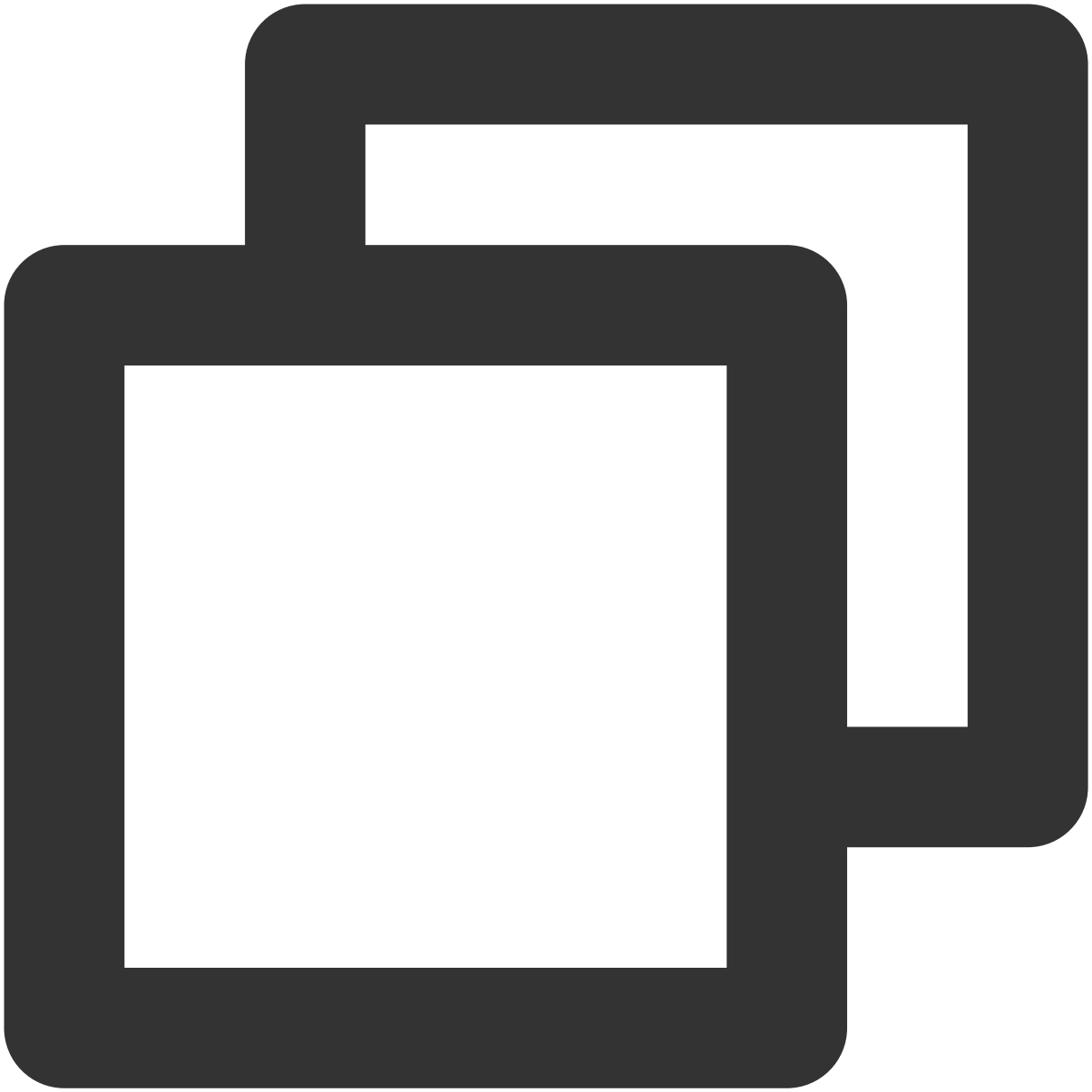
Install Conan

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



```
pip install
```

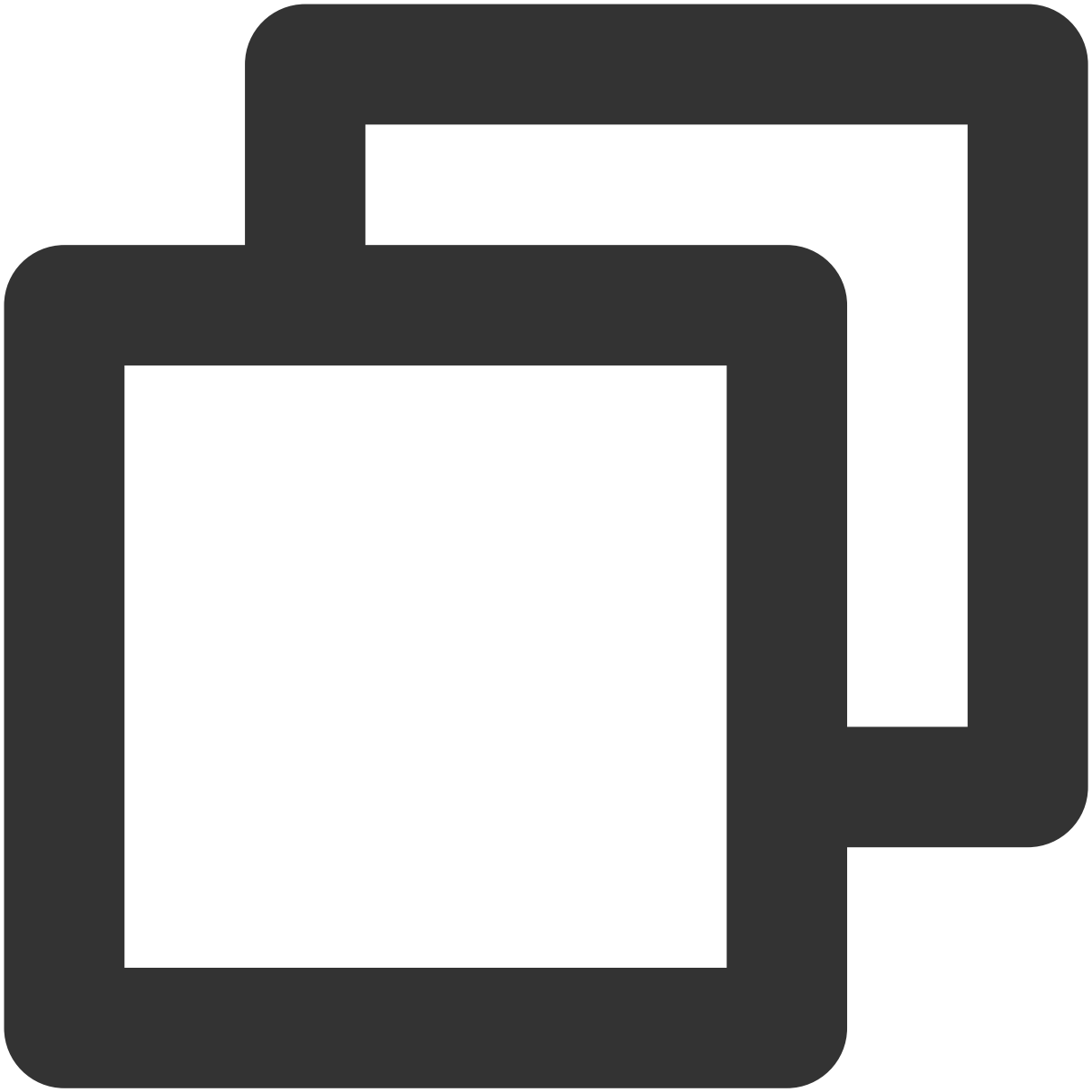
Install Conan with brew.



```
brew install conan
```

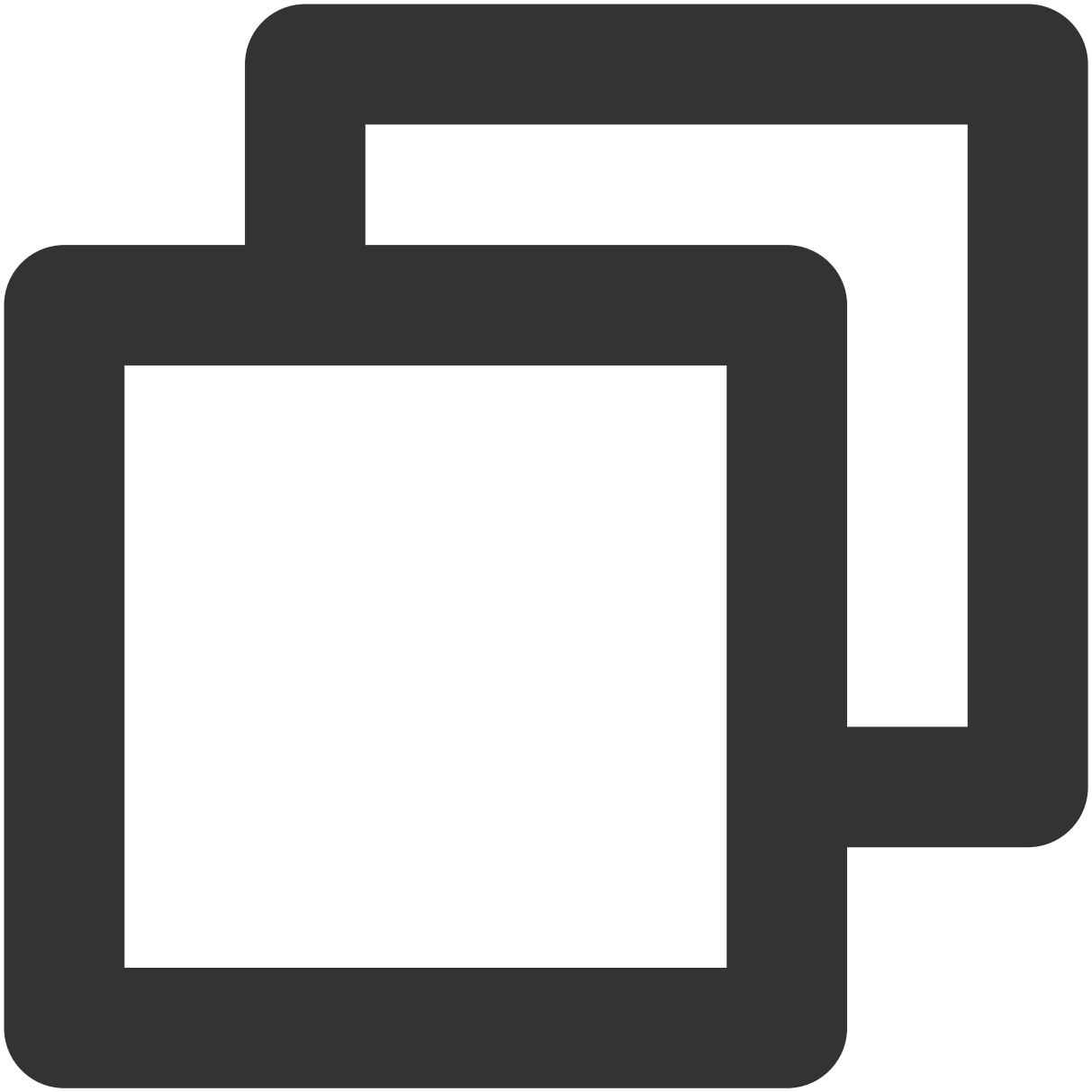
Create a Conan Package

Create a local demo directory.



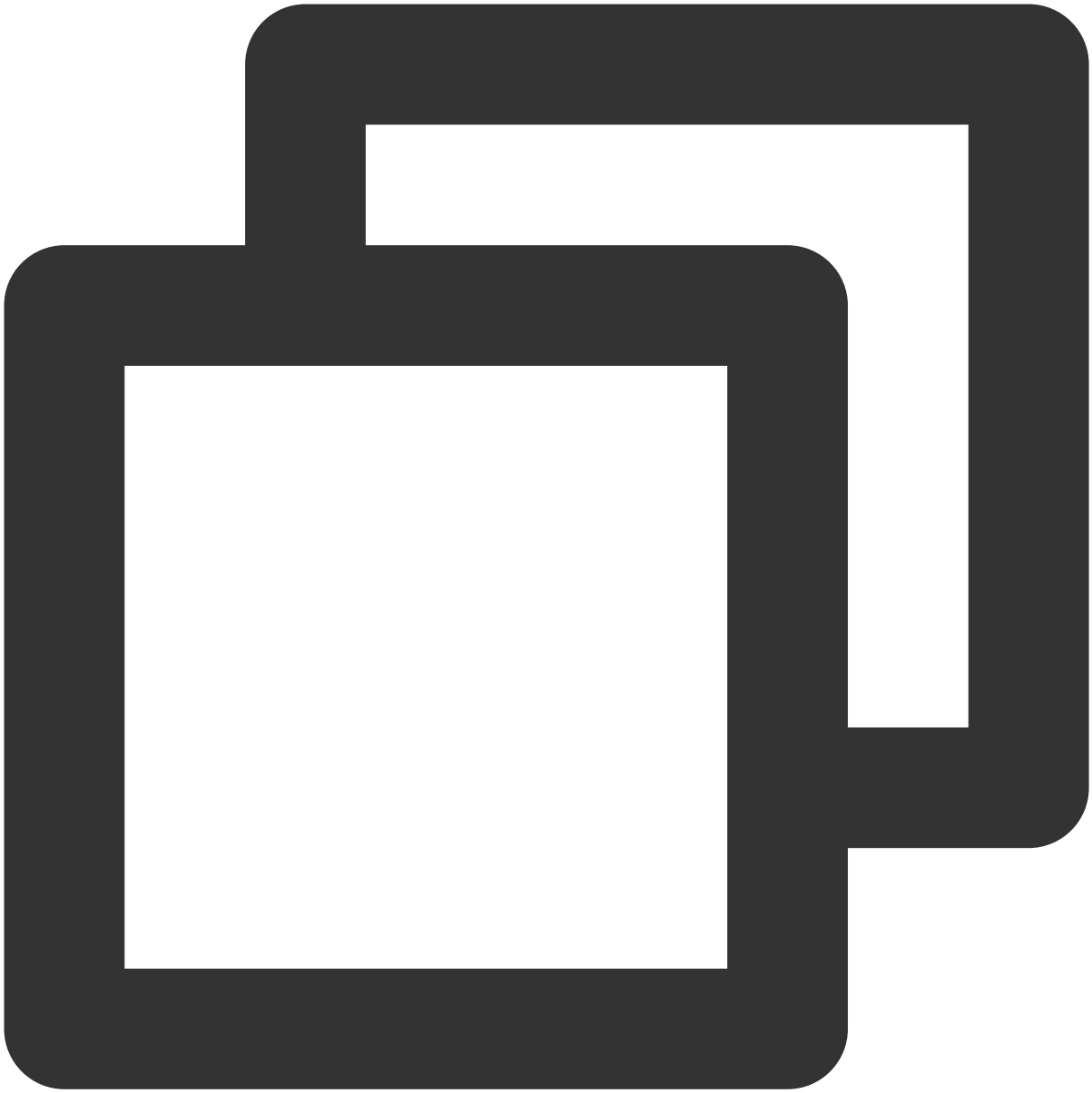
```
mkdir mypkg && cd mypkg
```

Create a demo project.



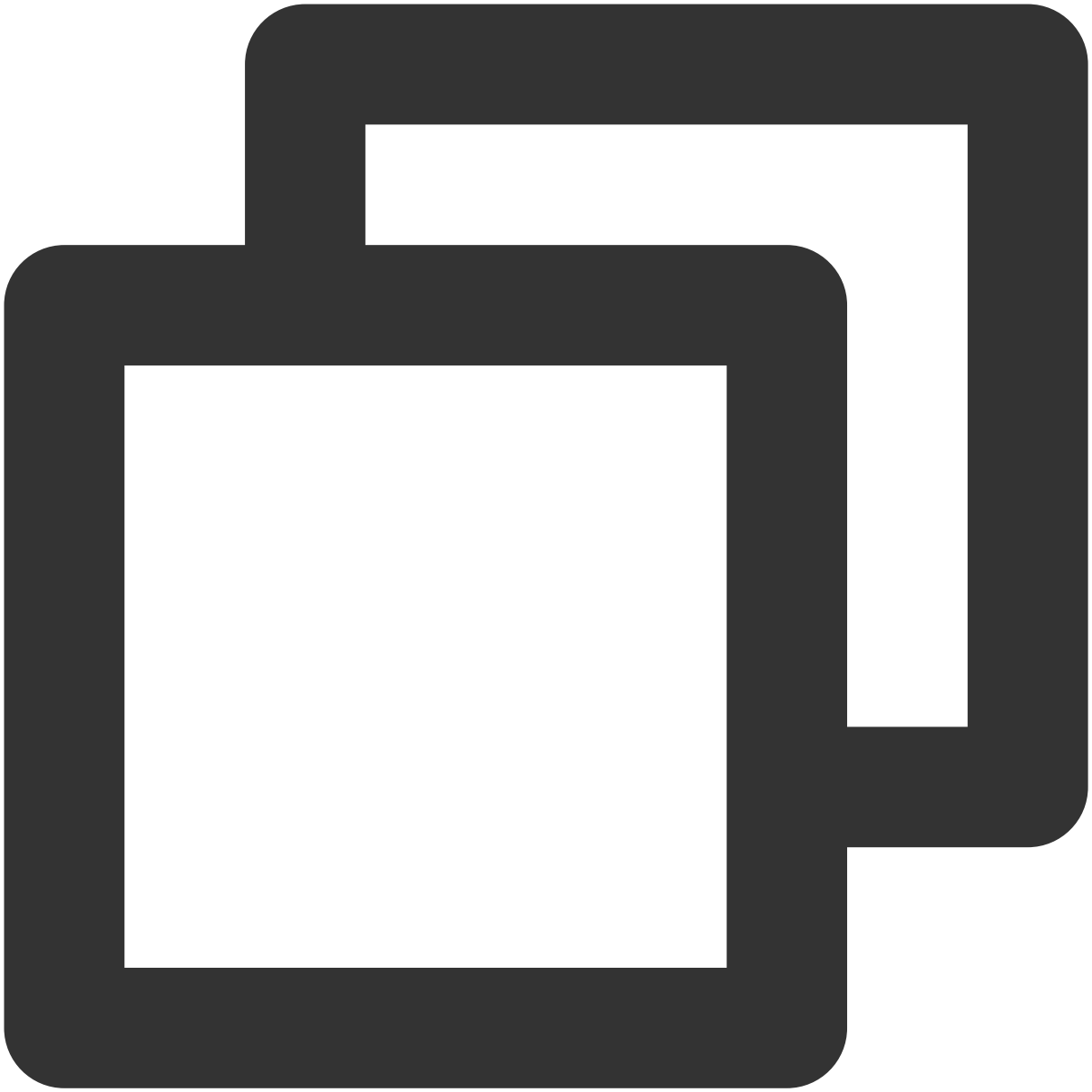
```
conan new hello/0.1 -t
```

Build a binary package for the project.



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

If an error message `/bin/sh: cmake: command not found` is displayed, run the following command to install `cmake` :

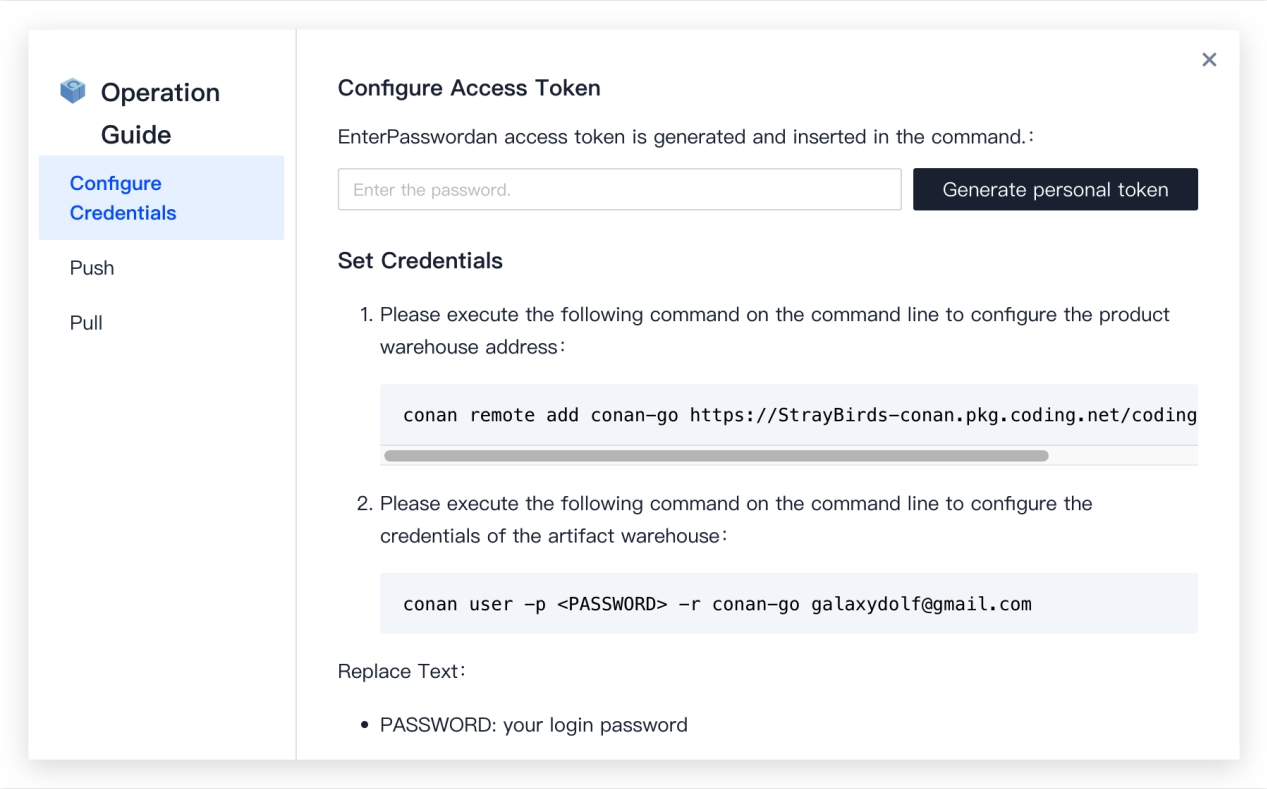


```
$ pip3 install cmake
# or
$ brew install cmake
```

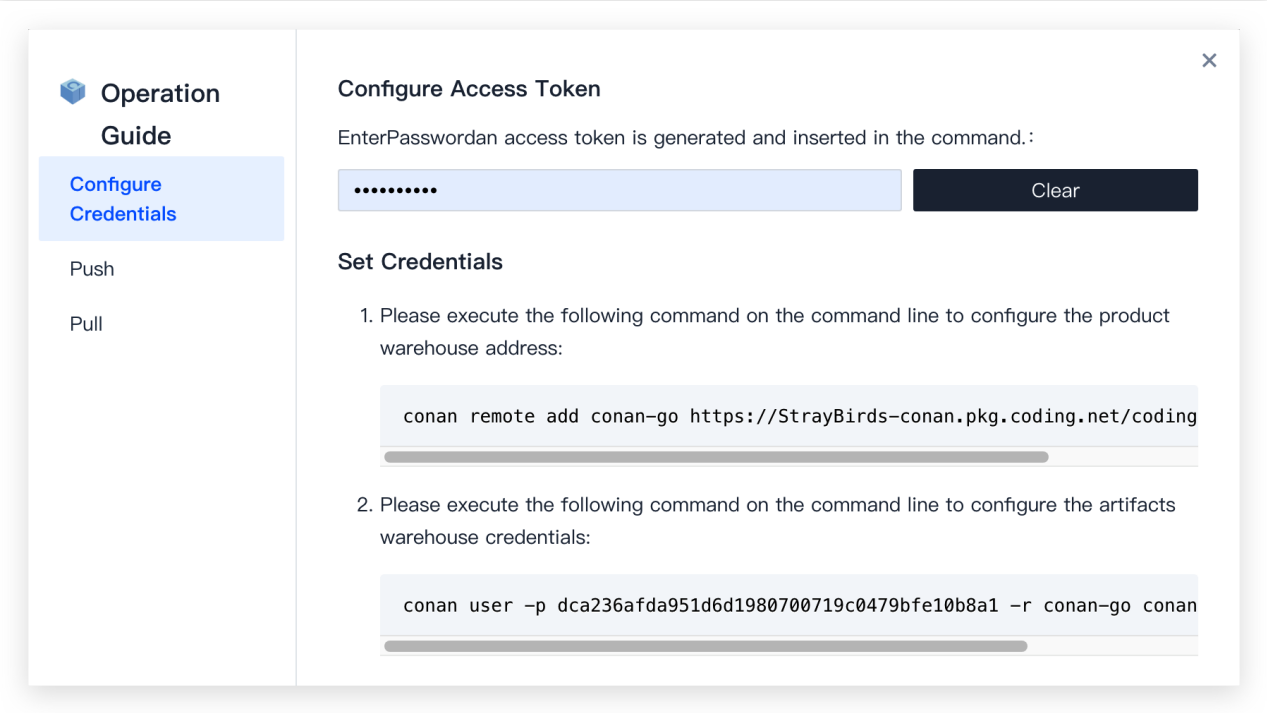
Configure Authentication Information

You can select manual or automatic configuration. Automatic configuration is used in the following example.

Click **Generate configuration from access token** in "Guide". A personal token will be generated as your access credential. You can manage your personal token in **Personal Account Settings** > **Access Token**.

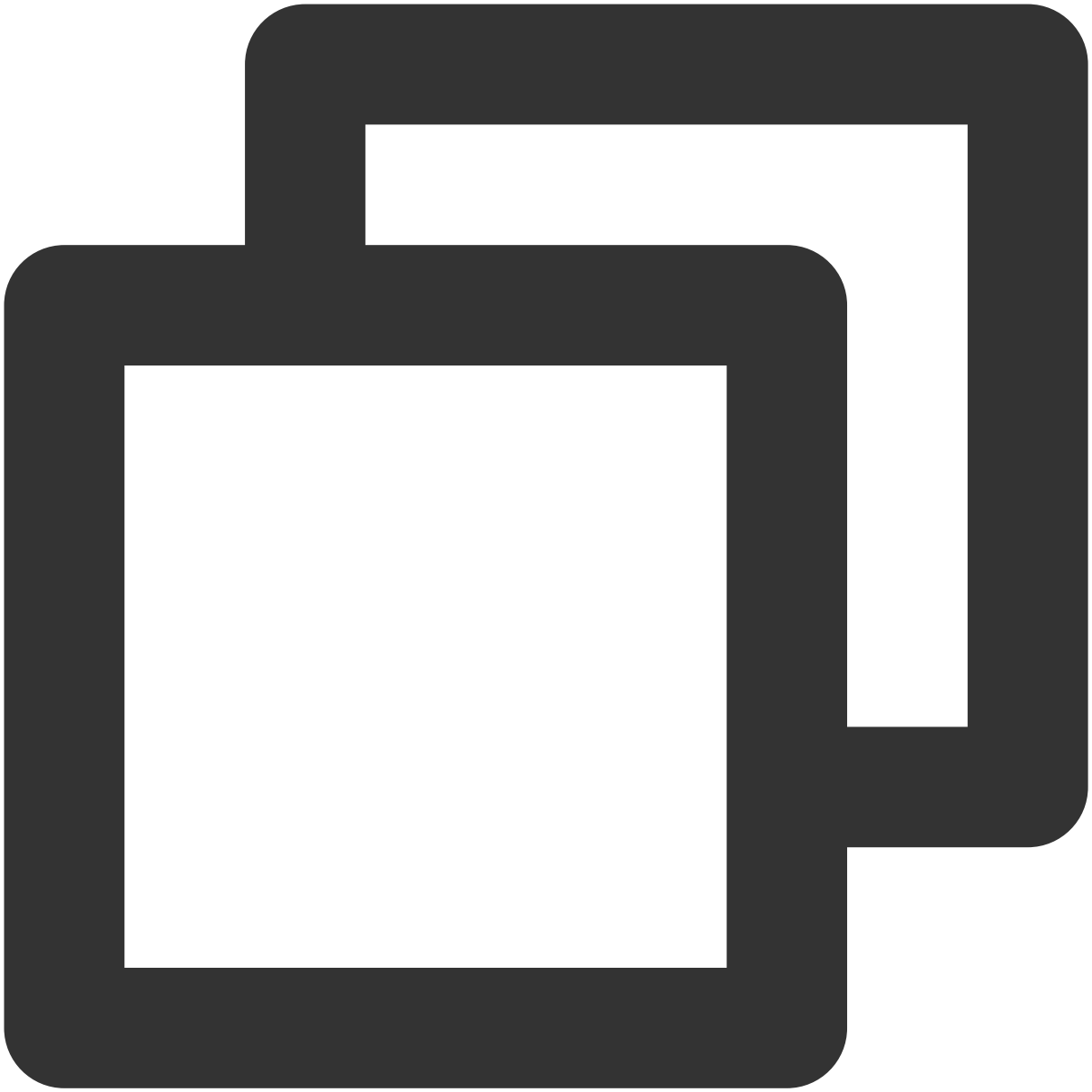


Run the commands as prompted.



Push an Artifact

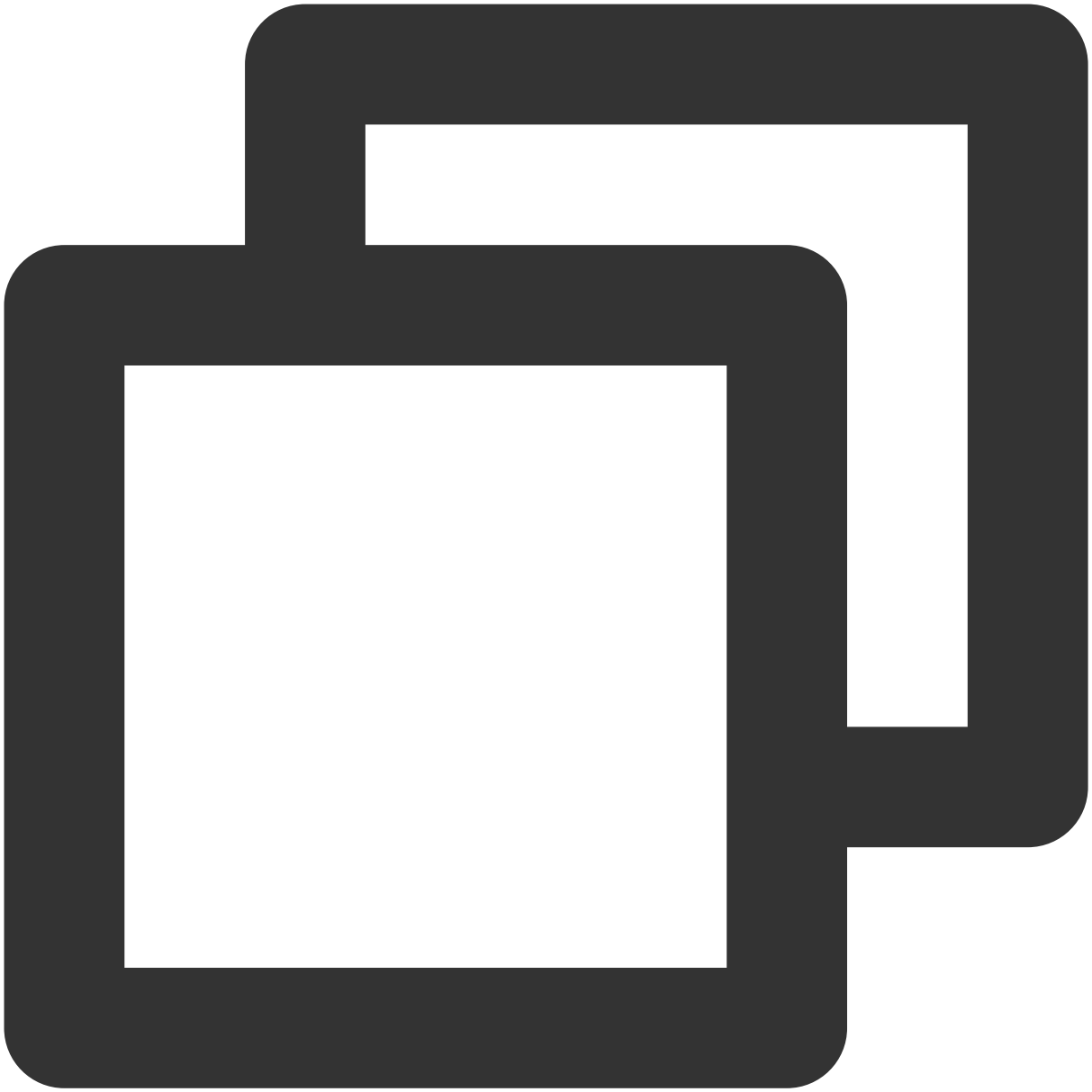
Run the command in the guide. Replace the variables with the name and version of the artifact to be pushed.



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

Pull an Artifact

Run the pull command in the guide.



```
conan install [package name]/[custom vesion number]@ -r conan-go
```

CocoaPods Repository

Last updated：2024-01-03 11:31:14

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.

Open CODING-AR

1. Log in to the CODING Console and click **Use Now** to go to CODING page.
2. Click

in the upper-right corner to open the project list page and click a project icon to open the project.
3. In the menu on the left, click **Artifact Management**.

Preparations

Note:

Before you begin:

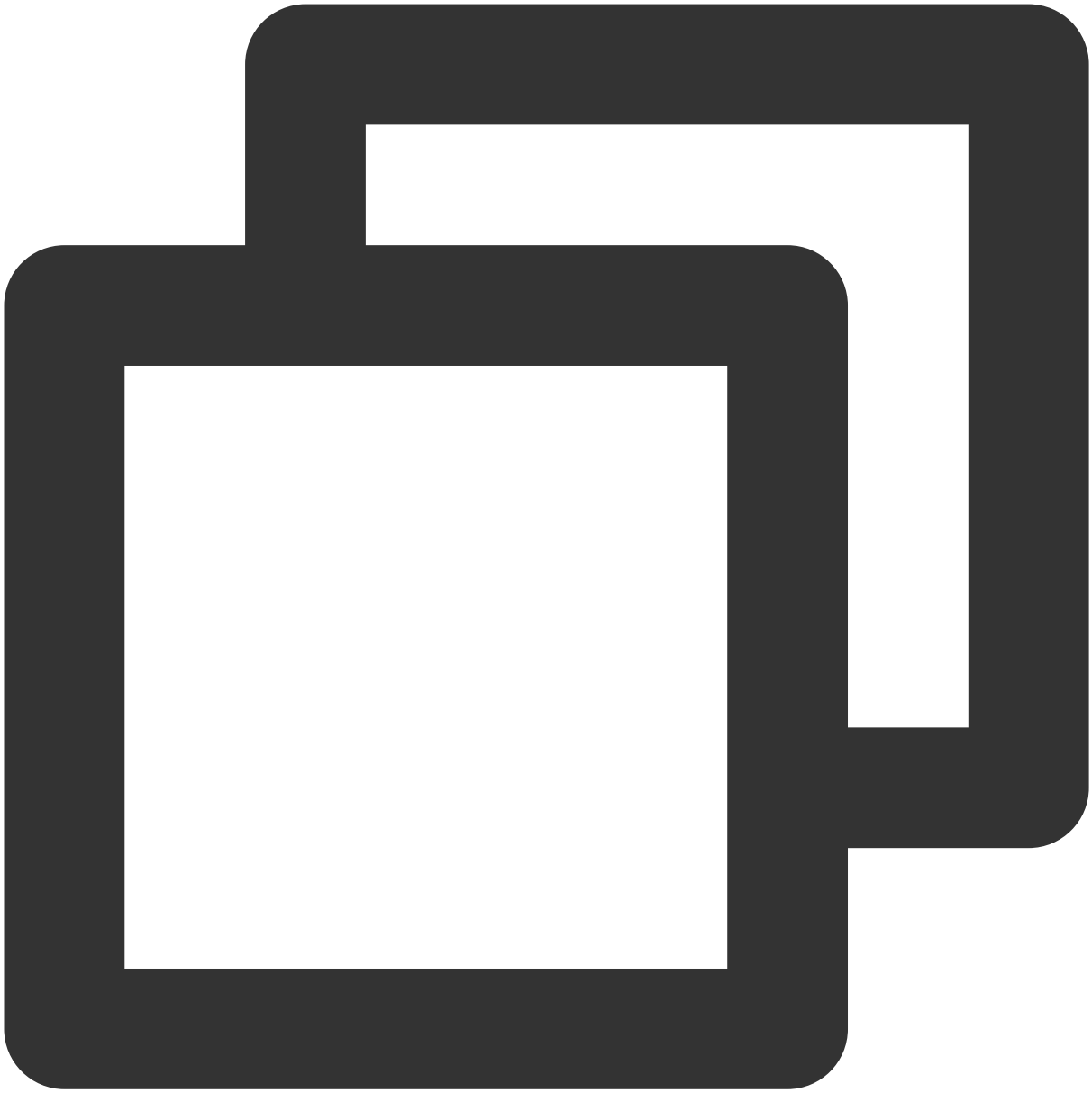
Install CocoaPods.

Create an artifact repository (see Basic Operations).

Select CocoaPods as the repository type.

Install CocoaPods

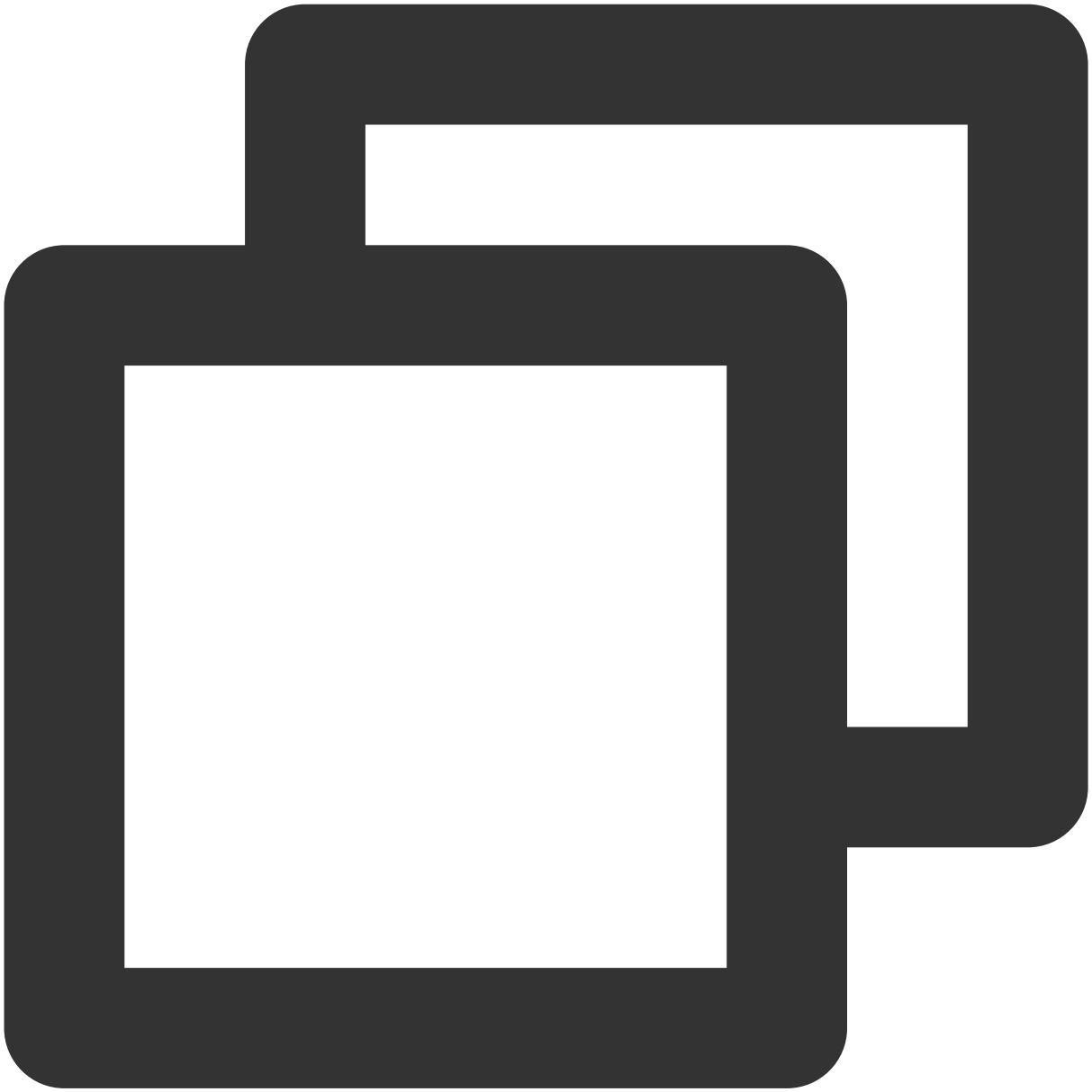
Run either of the following commands to install CocoaPods.



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

Create a Demo Project

This section describes how to quickly create a demo CocoaPods artifact. You can skip this section if you are familiar with CocoaPods artifacts.
Run the following command in a directory and select a template as needed when asked.

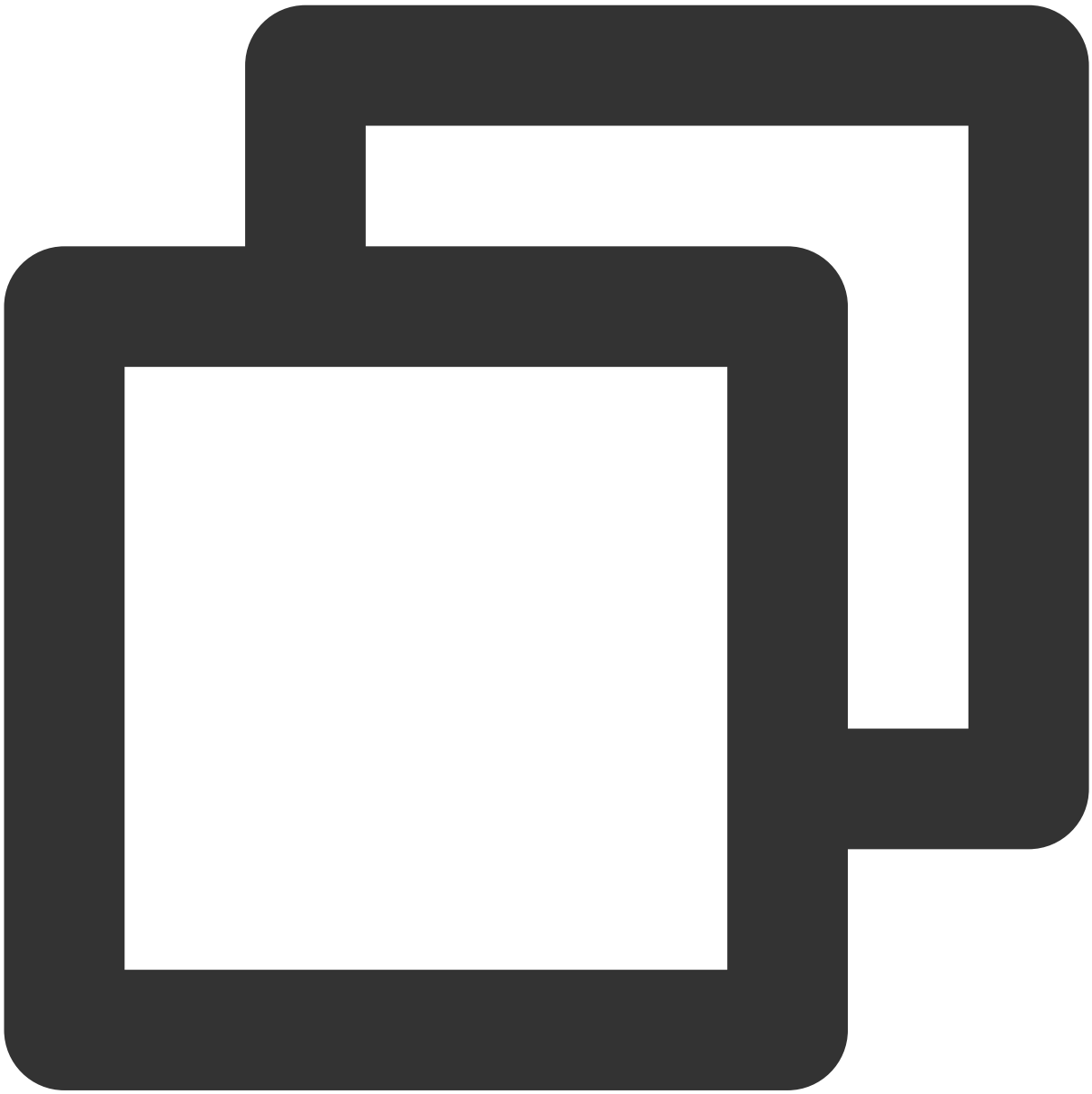


```
pod lib create <custom pod name>
```

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

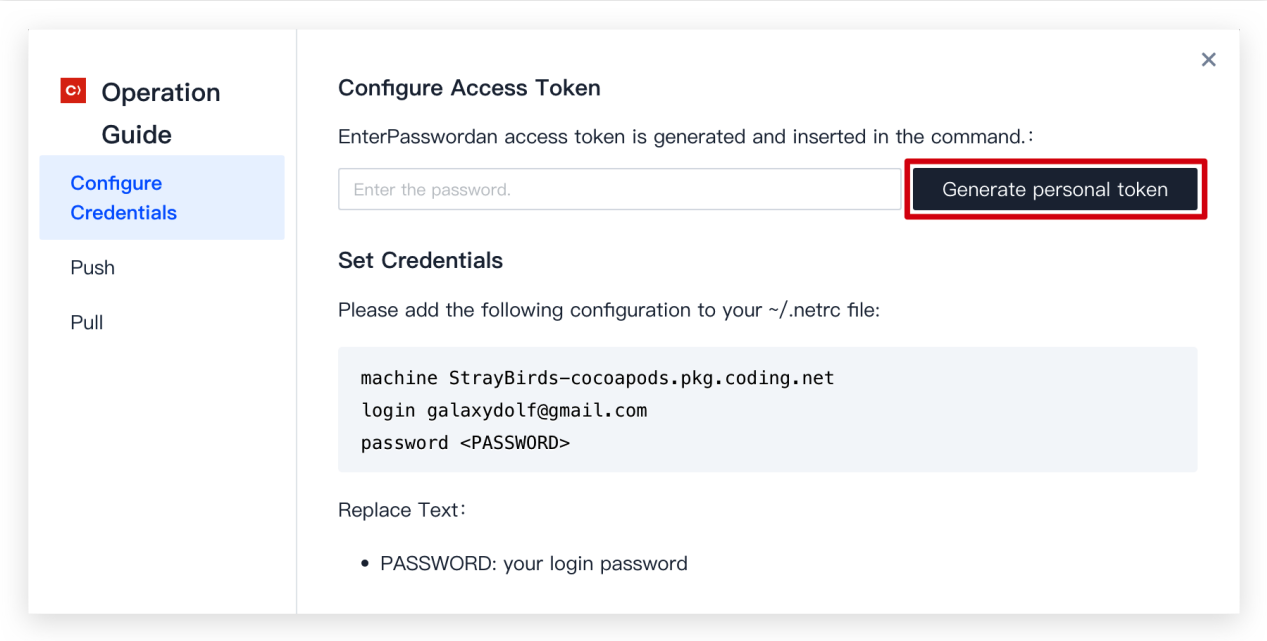
Configure Authentication Information

Install the CODING CocoaPods plugin.

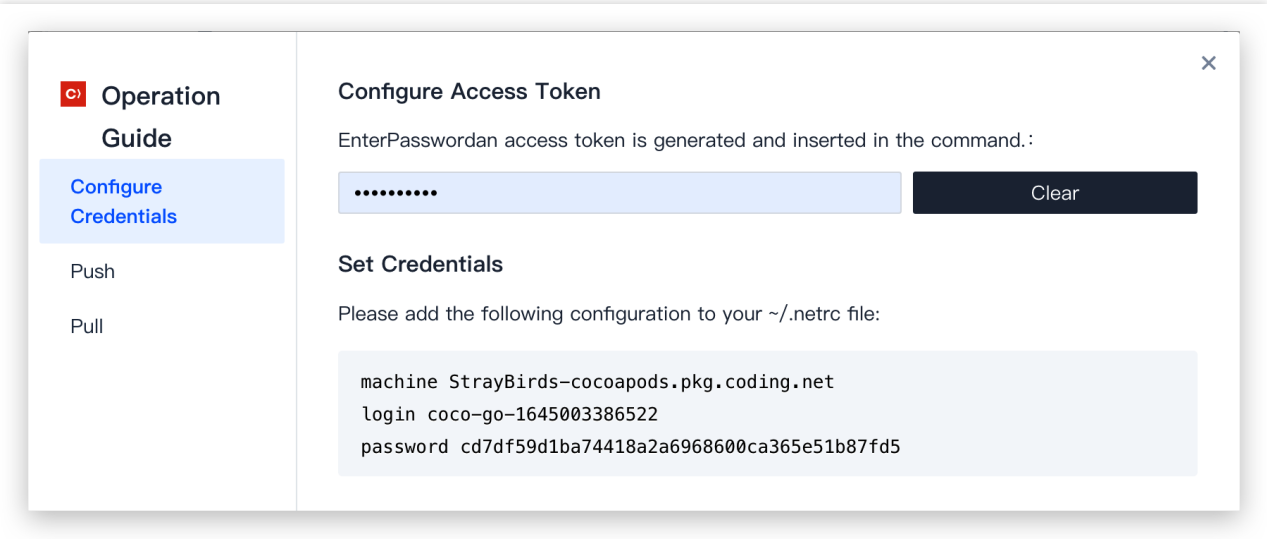


```
sudo gem install cocoapods-coding-ar
```

You can select manual or automatic configuration. Automatic configuration is used in the following example. Click **Generate configuration from access token** in "Guide". A personal token will be generated as your access credential. You can manage your personal token in **Personal Account Settings** > **Access Token**.

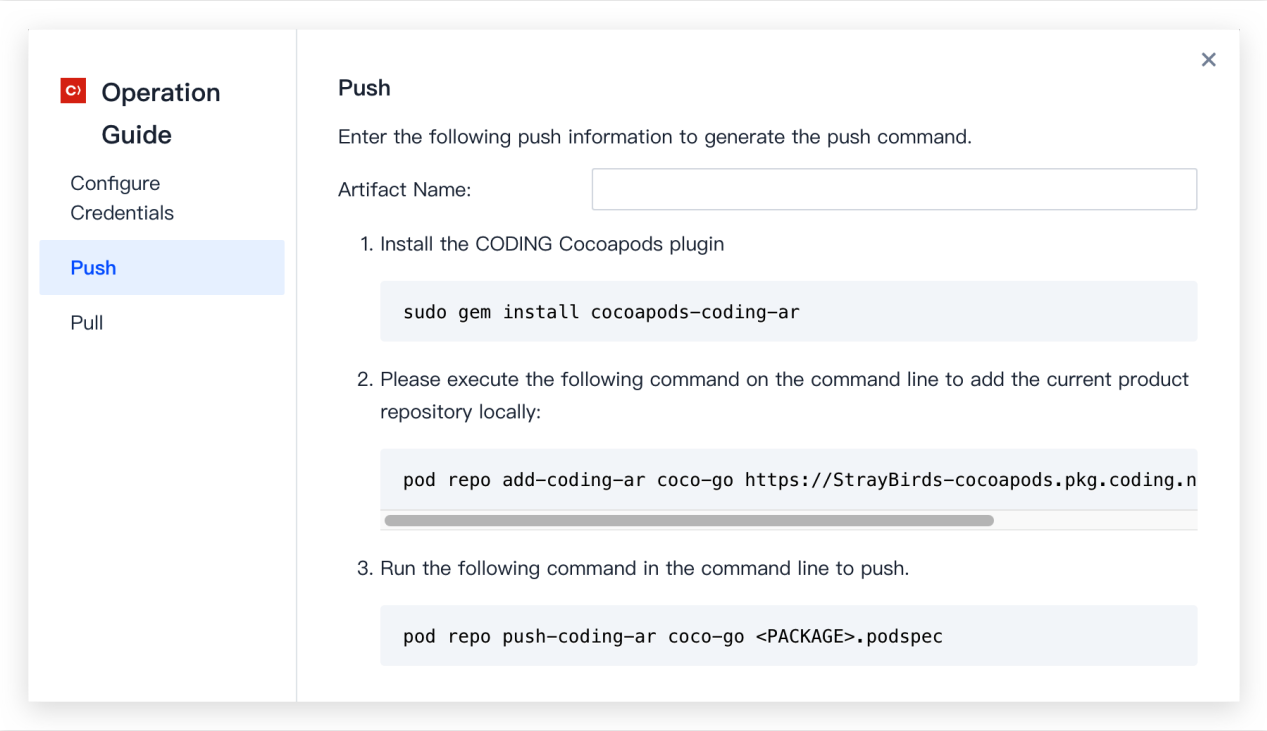


Copy the commands to your `~/.netrc` file. If this file does not exist, run `vim ~/.netrc` to create it and then copy the commands.



Push an Artifact

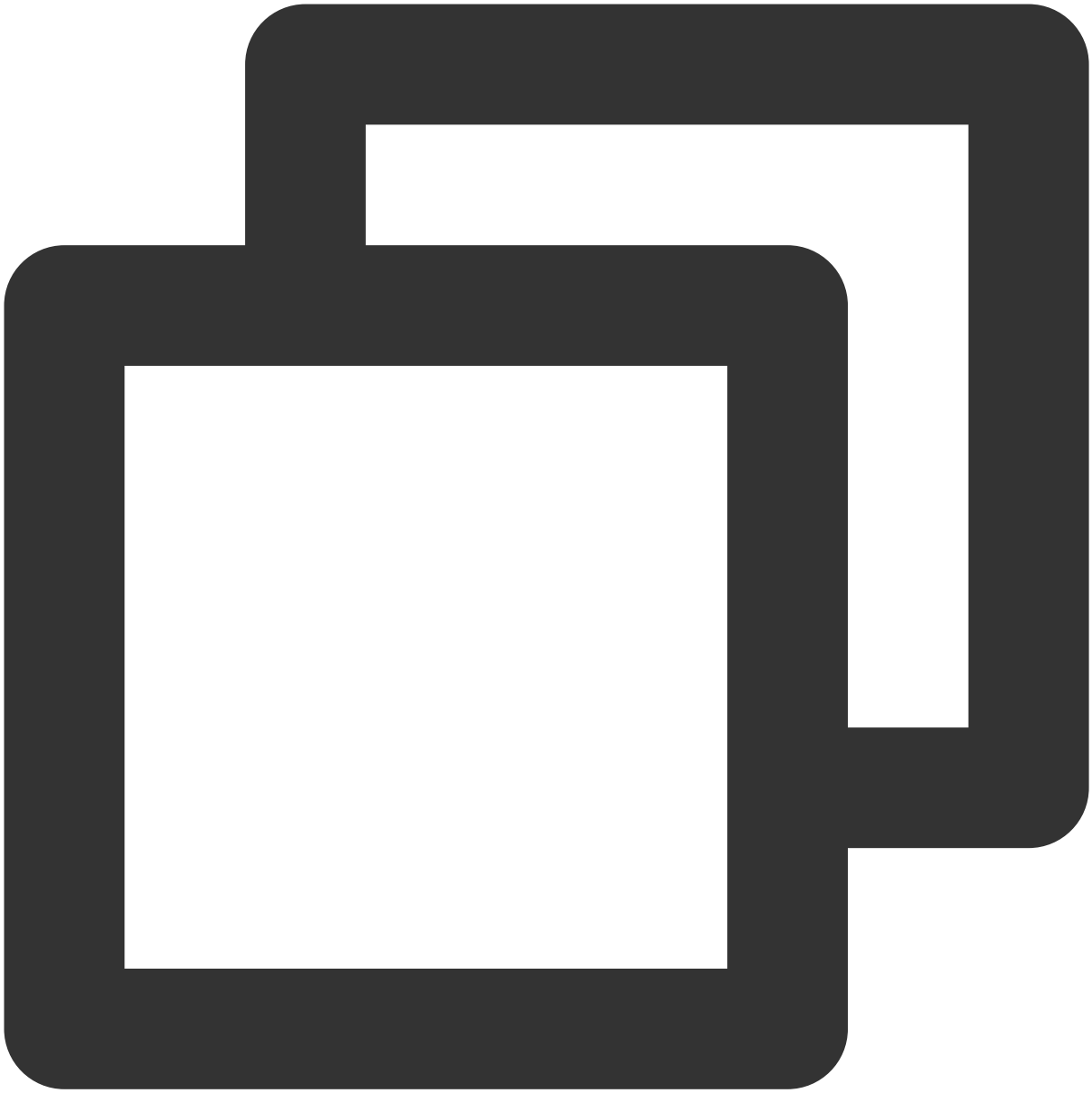
Run the commands in the guide.



Pull an Artifact

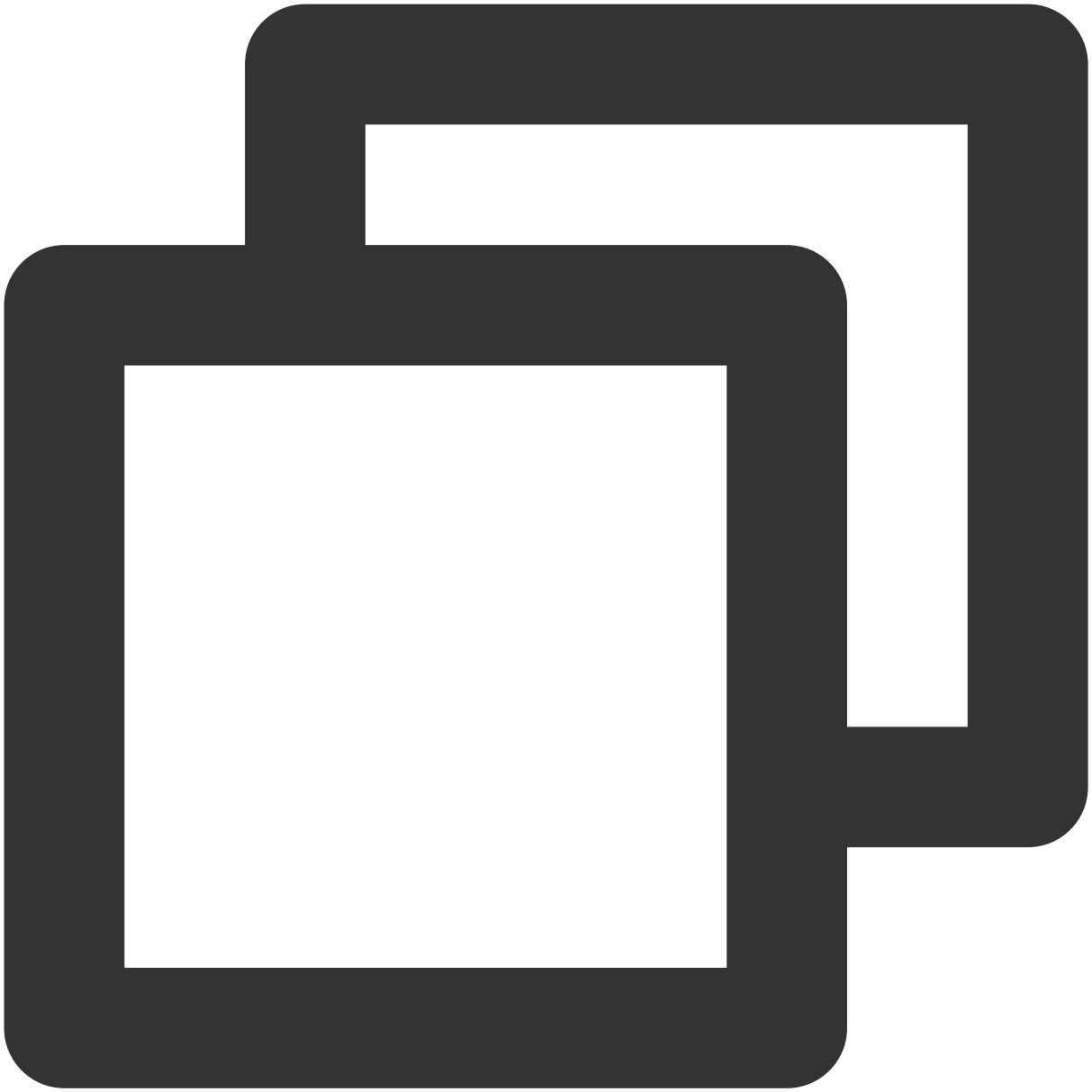
Pull a specific CocoaPods artifact by referring to the specific guide.

1. Modify the Podfile.



```
source '<repository URL in the guide>'
target 'MyApp' do
    pod '<Artifact Name>', '~> <Version>'
end
```

2. Run the pull command.



```
pod install
```