

CODING Artifact Repositories

Getting Started

Product Documentation



Copyright Notice

©2013-2024 Tencent Cloud. All rights reserved.

Copyright in this document is exclusively owned by Tencent Cloud. You must not reproduce, modify, copy or distribute in any way, in whole or in part, the contents of this document without Tencent Cloud's the prior written consent.

Trademark Notice



All trademarks associated with Tencent Cloud and its services are owned by Tencent Cloud Computing (Beijing) Company Limited and its affiliated companies. Trademarks of third parties referred to in this document are owned by their respective proprietors.

Service Statement

This document is intended to provide users with general information about Tencent Cloud's products and services only and does not form part of Tencent Cloud's terms and conditions. Tencent Cloud's products or services are subject to change. Specific products and services and the standards applicable to them are exclusively provided for in Tencent Cloud's applicable terms and conditions.

Contents

Getting Started

Basic Operations

Generic Repository

Docker Repository

npm Repository

Maven Repository

Helm Repository

PyPI Repository

Composer Repository

NuGet Repository

RPM Repository

Conan Repository

CocoaPods Repository

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**.

Create Repository

Artifact Repository*

Generic Docker Maven npm PyPI

Helm Composer NuGet Conan CocoaPods

Rpm

Repository Only letters, numbers, underscores (_), da

URL*

Repository Description

Permission External Permissions of the Artifact Repository ⓘ [View the complete description of artifact permissions.](#)

Scope

In project
Permission Scope
Pull ✓ members of the project X Team Members X Anonymous User
Push ✓ members of the project X Team Members X Anonymous User

In team

Public

Only members of this project can push or pull the artifacts of this repository. The project members' push and pull permissions can be [project settings](#) configured

Enable Proxy Allows to obtain the proxy source's Image ⓘ [What is artifact proxy?](#)

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**.

Create Repository

Artifact Repository*

Generic Docker Maven npm PyPI

Helm Composer NuGet Conan CocoaPods

Rpm

Repository URL* Only letters, numbers, underscores (_), da:

Repository Description

Permission External Permissions of the Artifact Repository ⓘ [View the complete description of artifact permissions.](#)

Scope In project In team Public

Permission Scope

Pull ✓ members of the project X Team Members X Anonymous User

Push ✓ members of the project X Team Members X Anonymous User

Only members of this project can push or pull the artifacts of this repository. The project members' push and pull permissions can be [project settings](#) configured

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**.

Operation Guide

- Configure Credentials
- Push**
- Pull
- Delete

[? Help Center](#)

Push

Enter the following push information to generate the push command.

Artifact Local Path:

Artifact Name:

Artifact Version:

Please execute the following command on the command line to push the product:

```
curl -T <LOCAL_FILE_NAME> -u galaxydolf@gmail.com "https://StrayBirds-generi
```

Please use the CODING Generic plug-in for resuming the oversized product from a breakpoint:

1. Install CODING Generic plugin (please install Node.js first)

```
npm install coding-generic -g
```

2. Push artifact

```
coding-generic -u=galaxydolf@gmail.com --path=<LOCAL_FILE_NAME> --regist
```

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

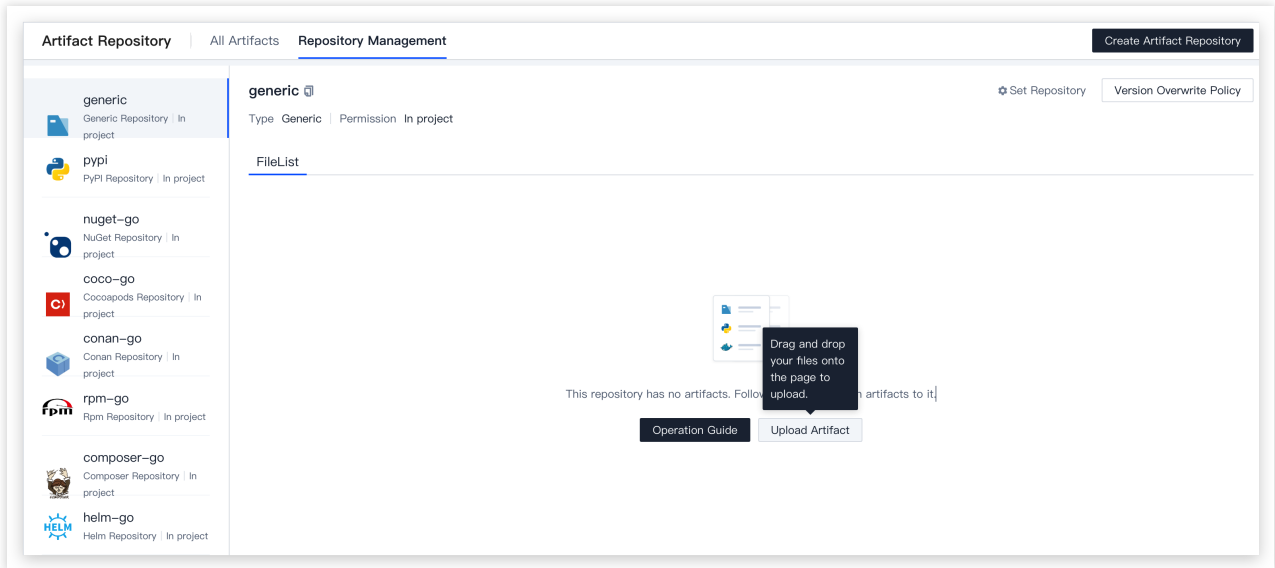


```
curl -T demo.properties -u username "https://*****-generic.pkg.coding.net/my-pro
```

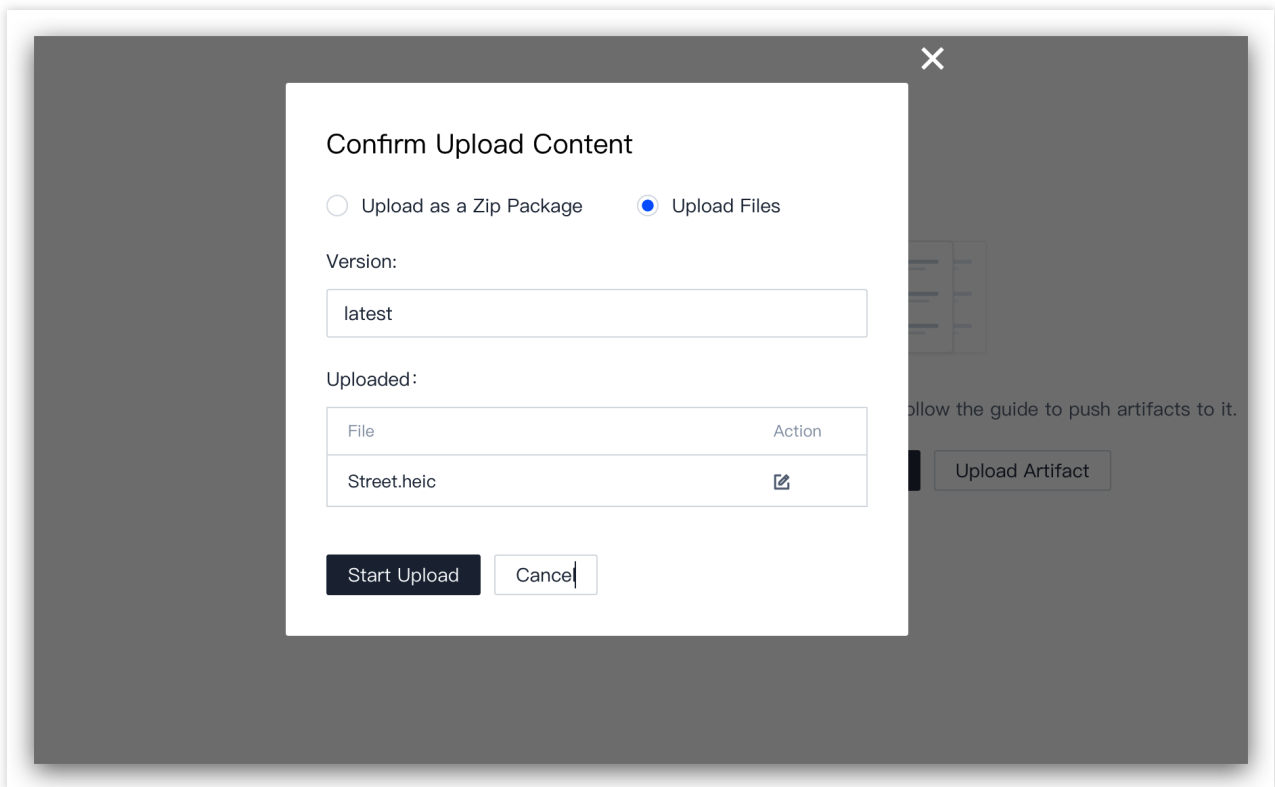
```
~/Downloads $ curl -T demo.properties -u ***** "https://
/*****-generic.pkg.coding.net/my-projects/my-generic/demo.properties?version=0.1"
Enter host password for user '*****':
success
```

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.

The screenshot displays the 'generic' artifact repository page. At the top, there are options for 'Set Repository' and 'Version Overwrite Policy'. Below this, the page is titled 'FileList' and includes filters for 'Release Status All' and '+ Artifact Field'. A search bar is labeled 'Search by artifact'. On the right, there are buttons for 'Operation Guide' and 'Upload Artifact'. The main content is a table with the following columns: '文件名' (File Name), 'Latest Push Version', 'Last Updated', 'Number of Versions', and 'Action'. The table contains one entry: 'Testing.zip' with version 'latest', updated on '2022-02-15 16:59:19', and 1 version. Below the table, it shows '1-1', 'Total 1', and 'Rows per page 15'. An 'Upload Tasks of Artifact Repository' dialog box is open in the bottom right, showing a table with columns 'File', 'Version', 'Progress', and 'Action'. It lists 'Testing.zip' with version 'latest' and a progress status of 'Uploaded Successfully'.

文件名	Latest Push Version	Last Updated	Number of Versions	Action
Testing.zip	latest	2022-02-15 16:59:19	1	...

File	Version	Progress	Action
Testing.zip	latest	Uploaded Successfully	⊗

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**.

Operation Guide

- Configure Credentials
- Push
- Pull**
- Delete

Pull

Enter the following pull information to generate the pull command:

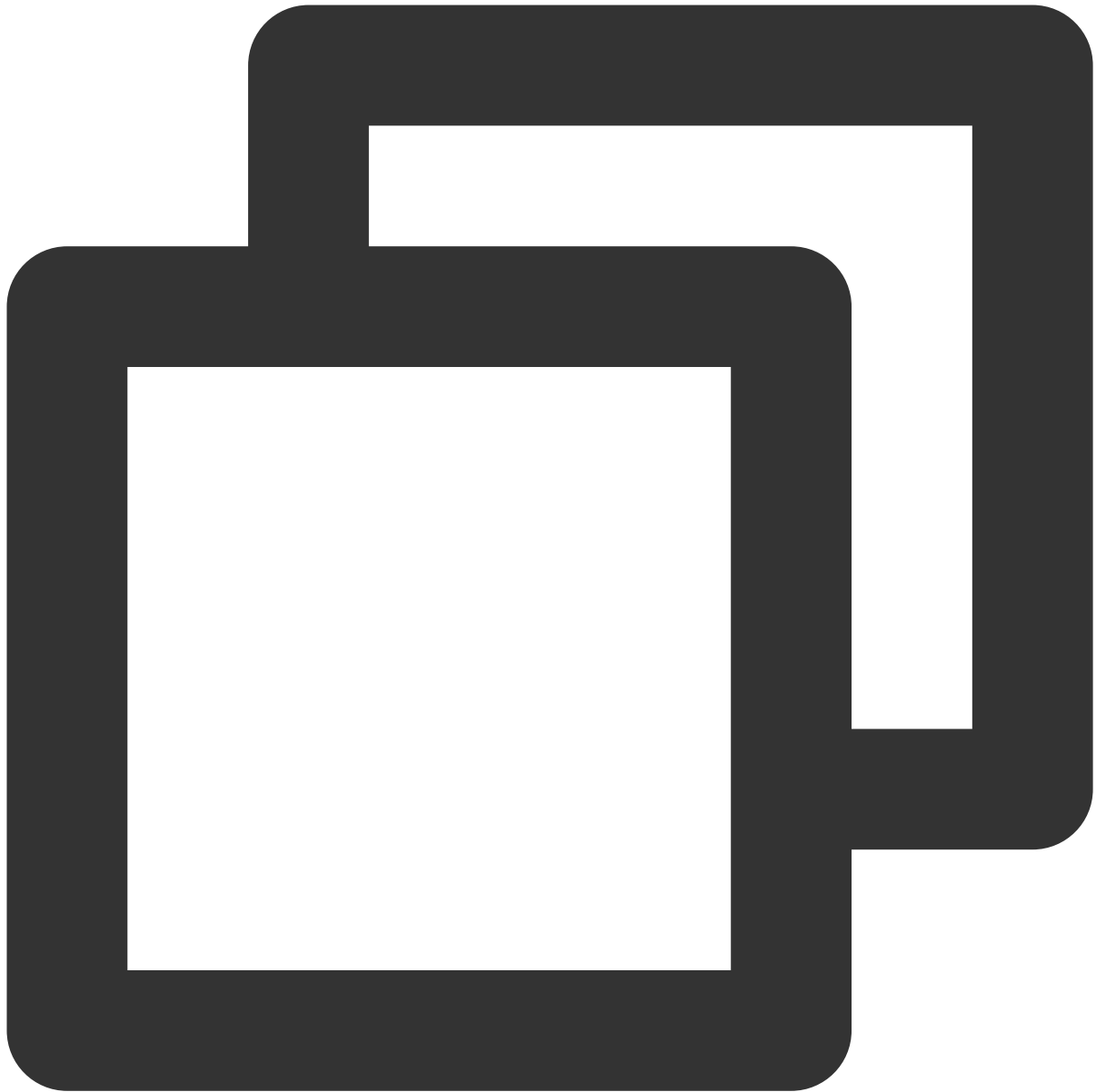
Artifact Name:

Artifact Version:

The file name after pulling to the local:

Please execute the following command on the command line to pull the product:

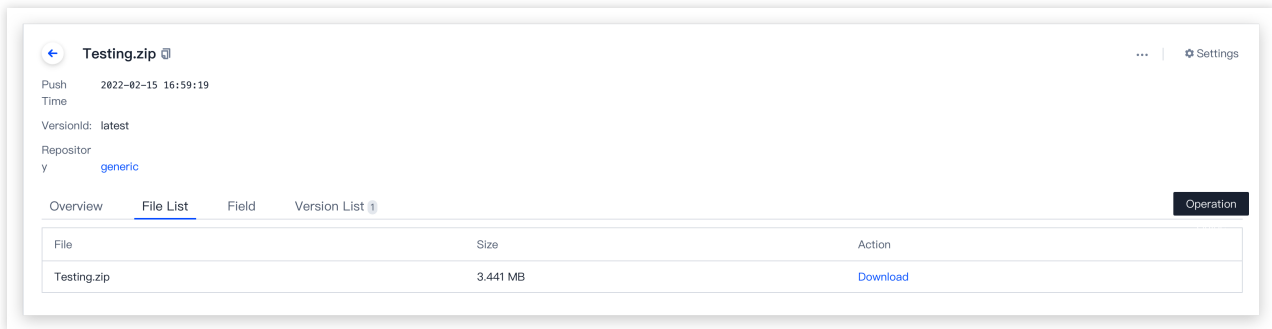
```
curl -fL -u galaxydolf@gmail.com "https://StrayBirds-generic.pkg.coding.net/
```



```
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-
```

```
Downloads $ curl -X DELETE -u ***** "https://
-generic.pkg.coding.net/my-projects/my-generic/demo.properties?version=0.1"
Enter host password for user '*****':
success
```

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](#)).

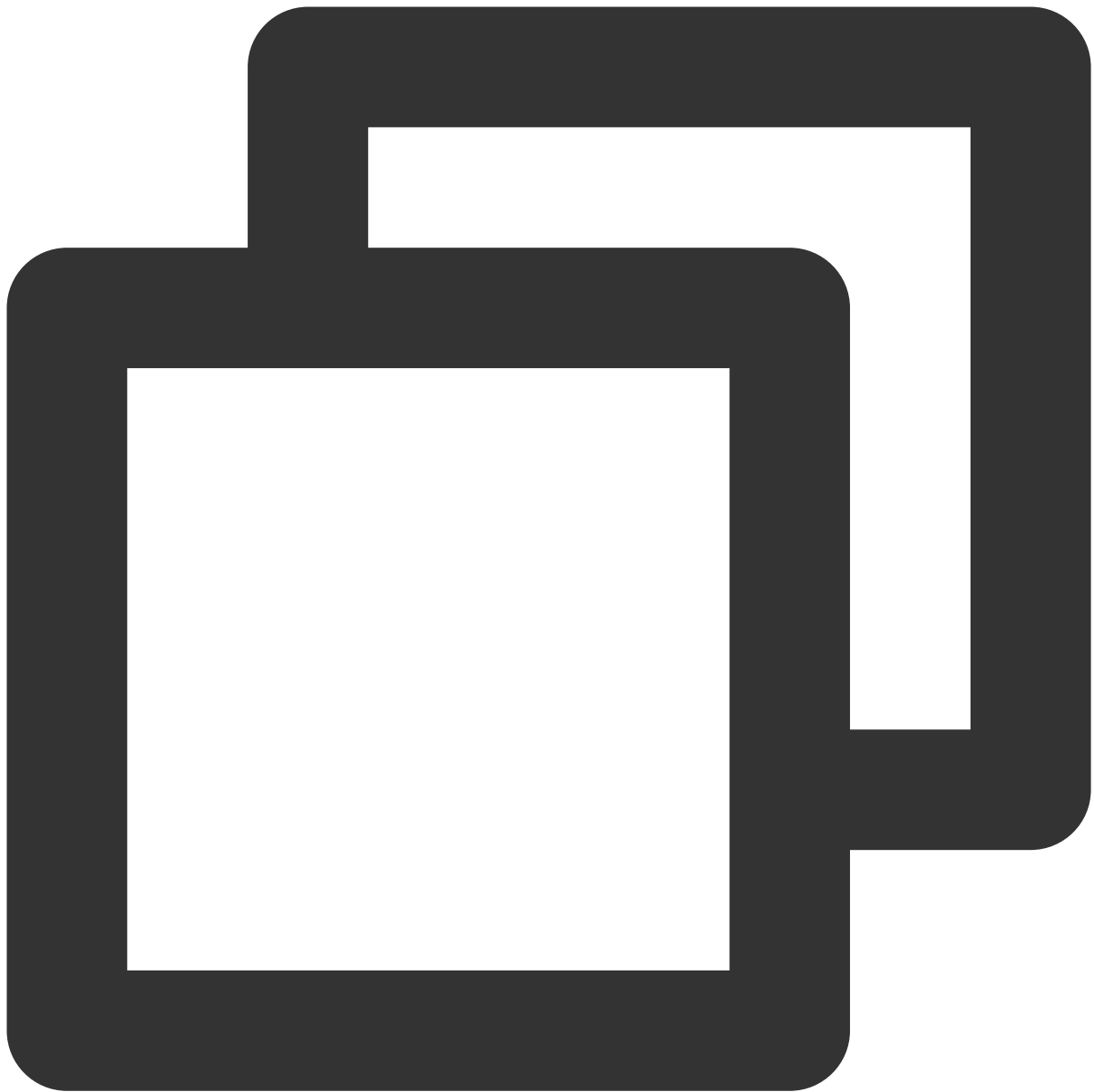
Select Docker as the repository type.

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.

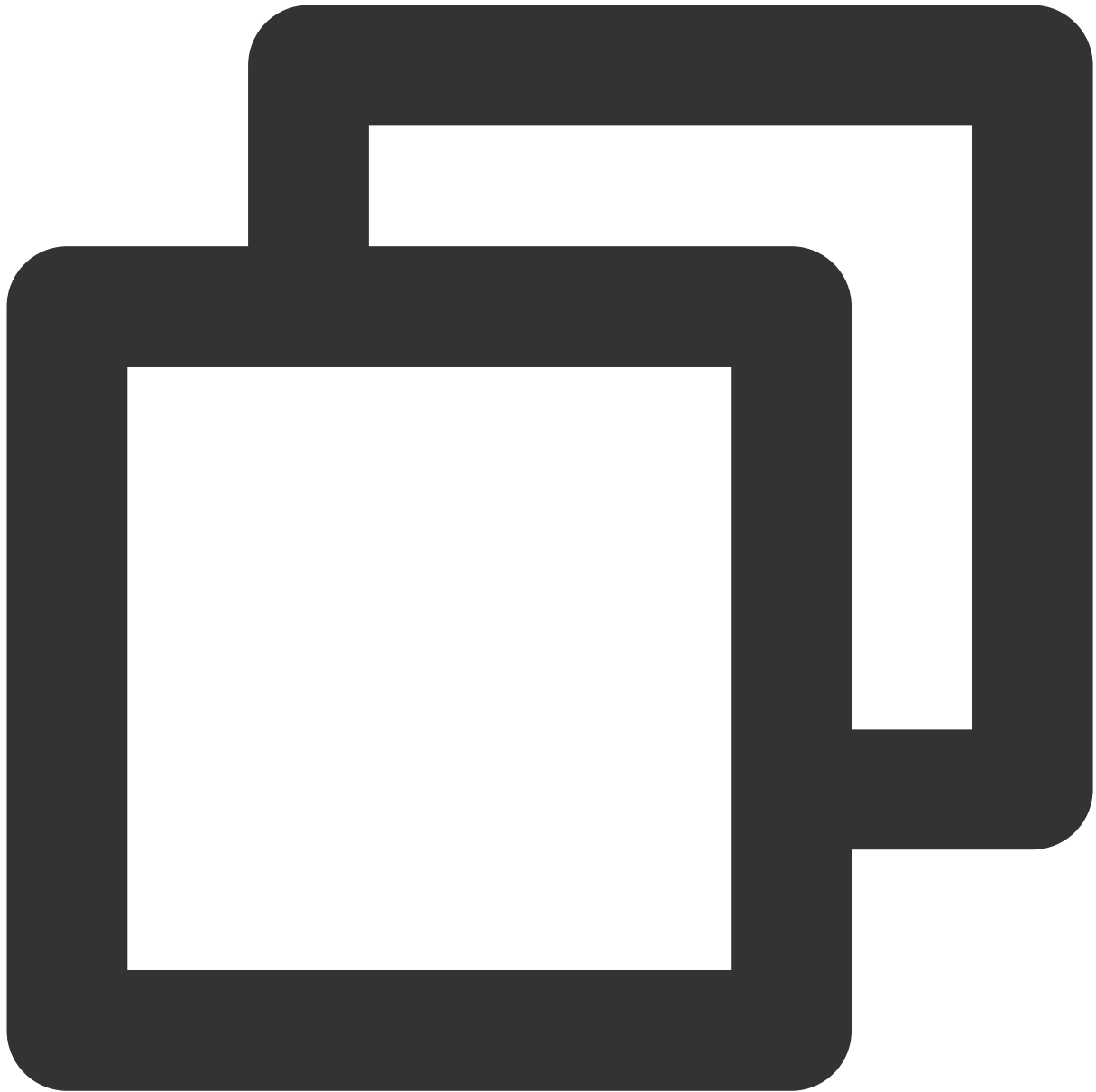
Method 1: create an image locally

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.



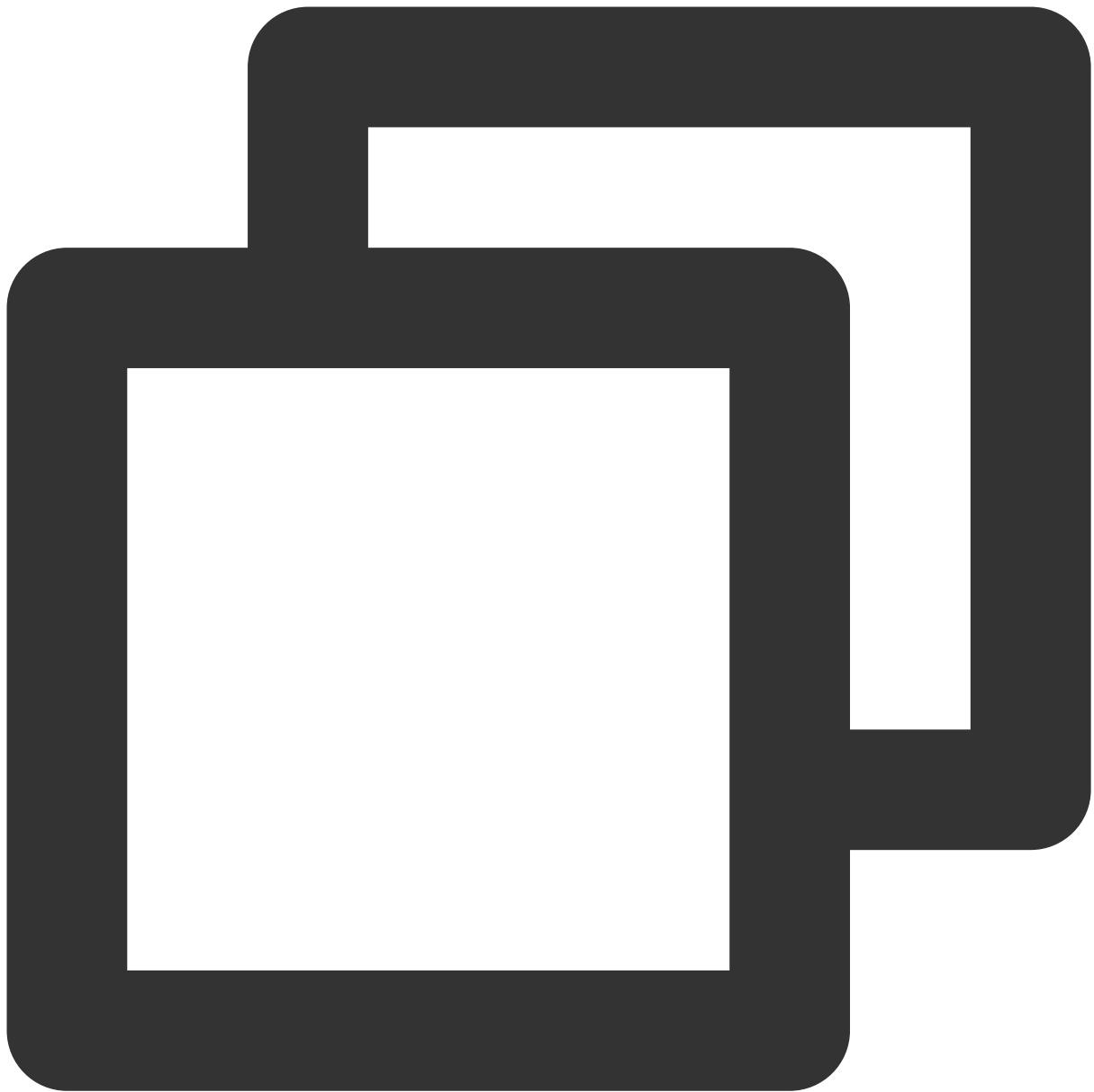
```
docker build -t hello-world .
```

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
```

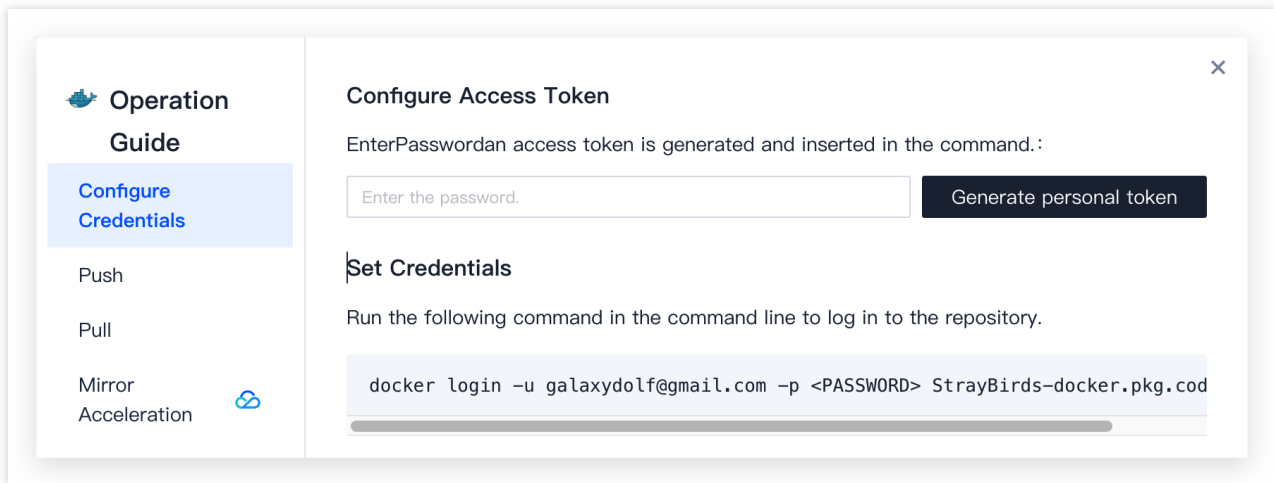
Configure Authentication Information

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.



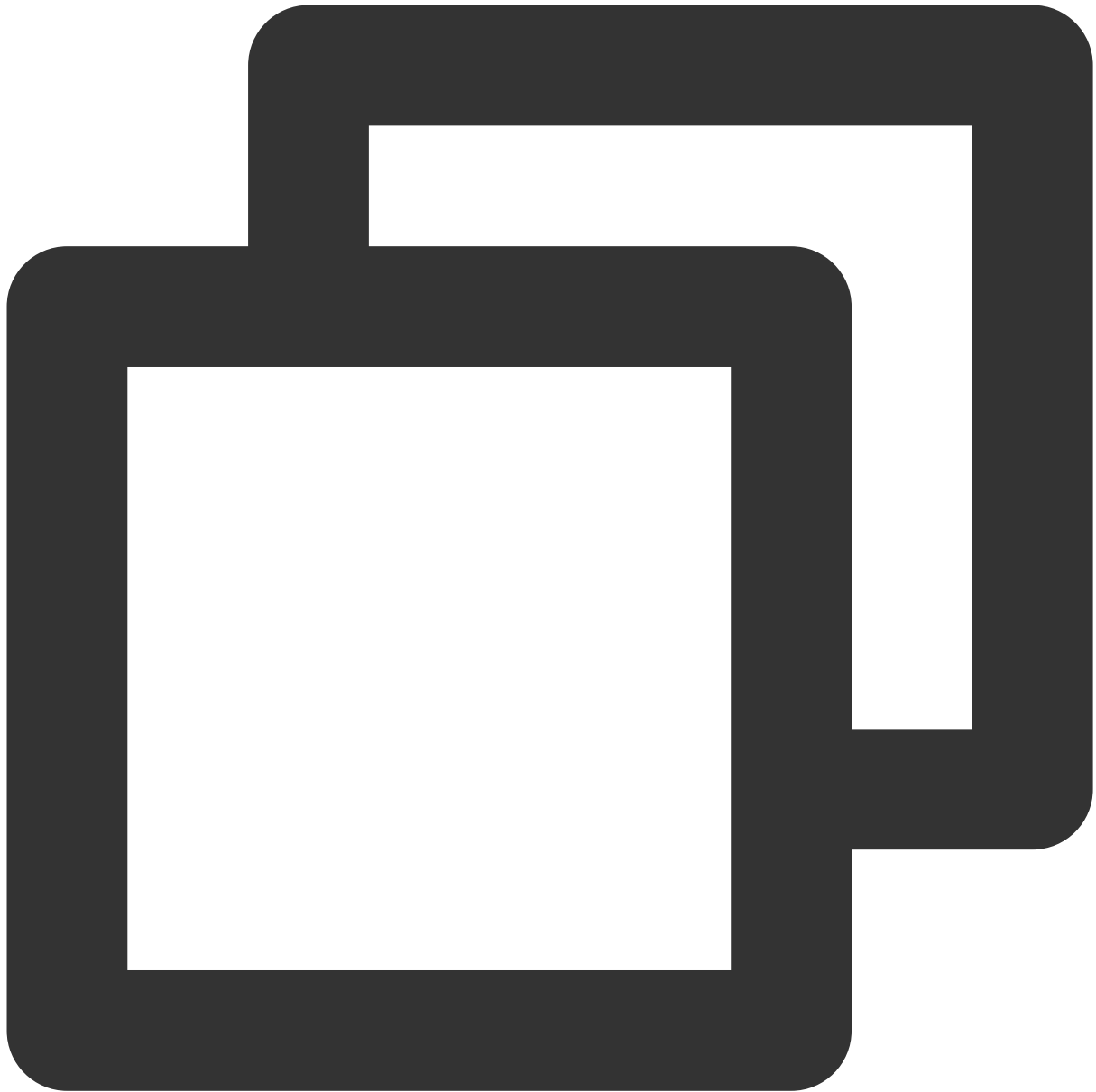
3. Paste and run the command in the local Docker environment to complete the authentication.

```
→ ~ docker login -u -p artifacts-docker.pkg.coding.net
WARNING! Using --password via the CLI is insecure. Use --password-stdin.
Login Succeeded
→ ~ █
```

Push an Image

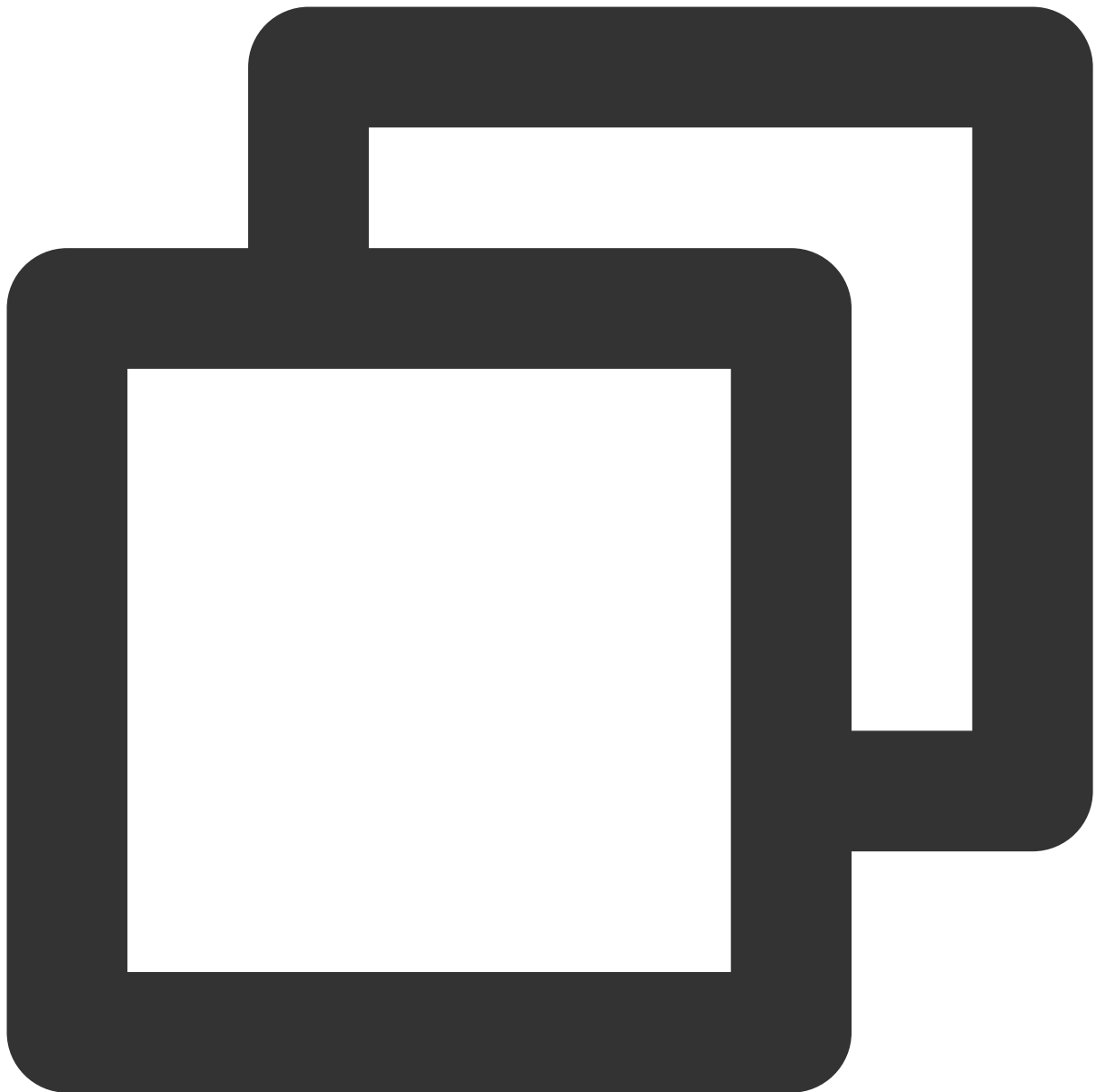
The following commands are for reference only. Use the commands generated in your project.

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: size: 
```

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

Operation Guide

- Configure Credentials
- Push**
- Pull
- Mirror Acceleration

Push

Enter the following push information to generate the push command.

Local Image Tag:

Artifact Name:

Artifact Version:

1. Run the following command in the command line to tag the local image.

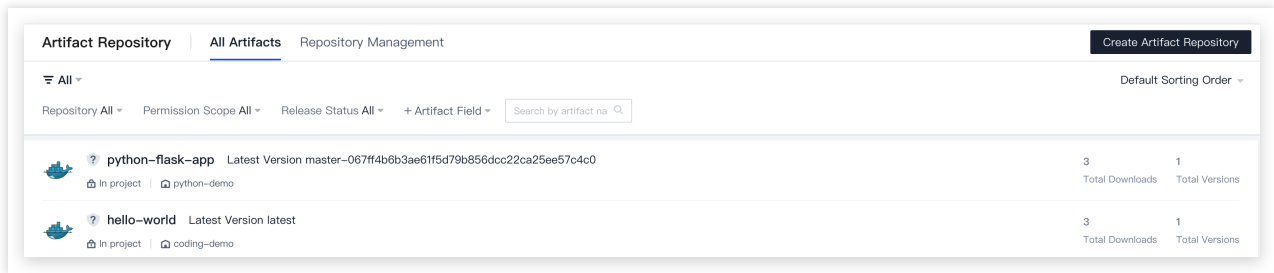
```
docker tag <LOCAL_IMAGE_TAG> StrayBirds-docker.pkg.coding.net/coding-dem
```
2. Run the following command in the command line to push.

```
docker push StrayBirds-docker.pkg.coding.net/coding-demo/python-demo/<PA
```

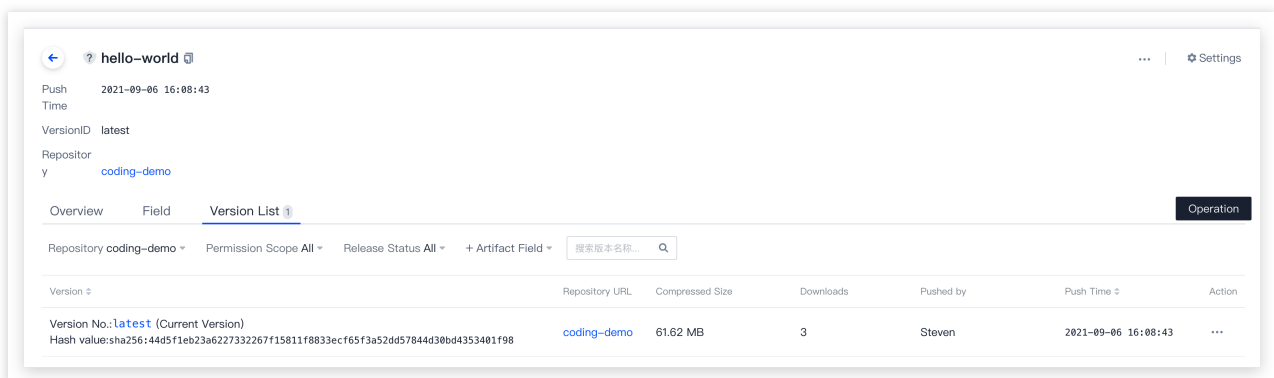
View an Image

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

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

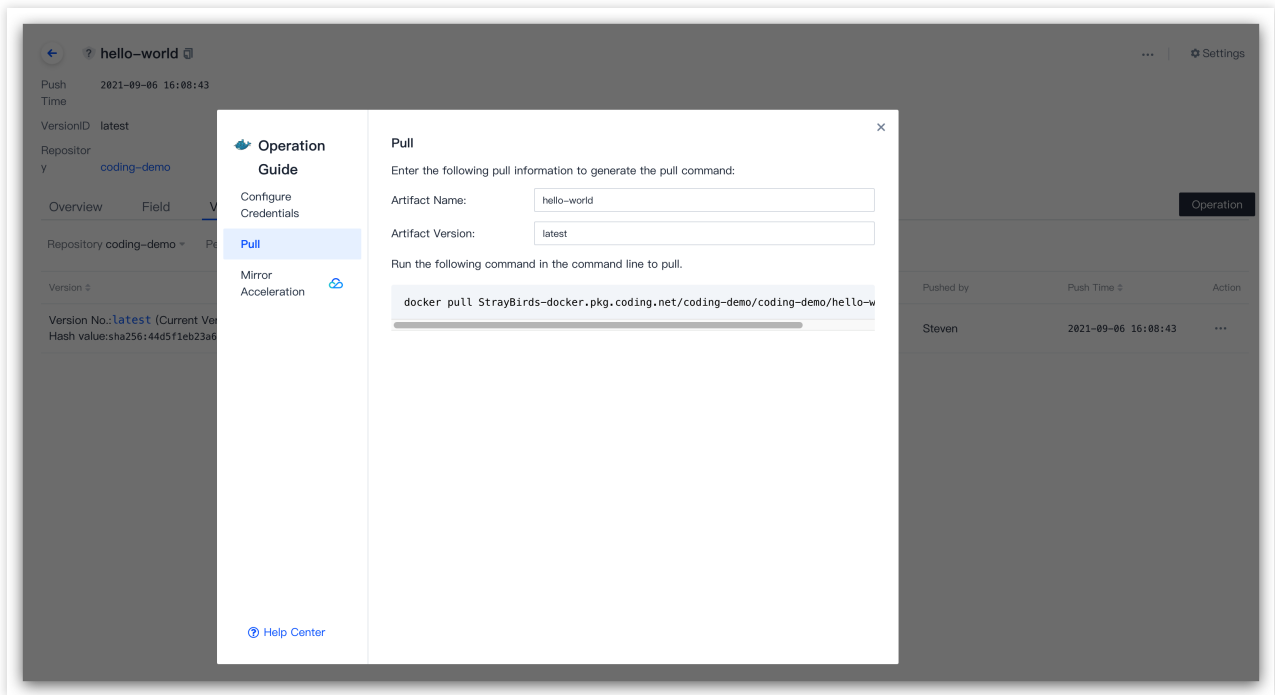


Click the image name to view its information, including overview, guide, properties, and version list.

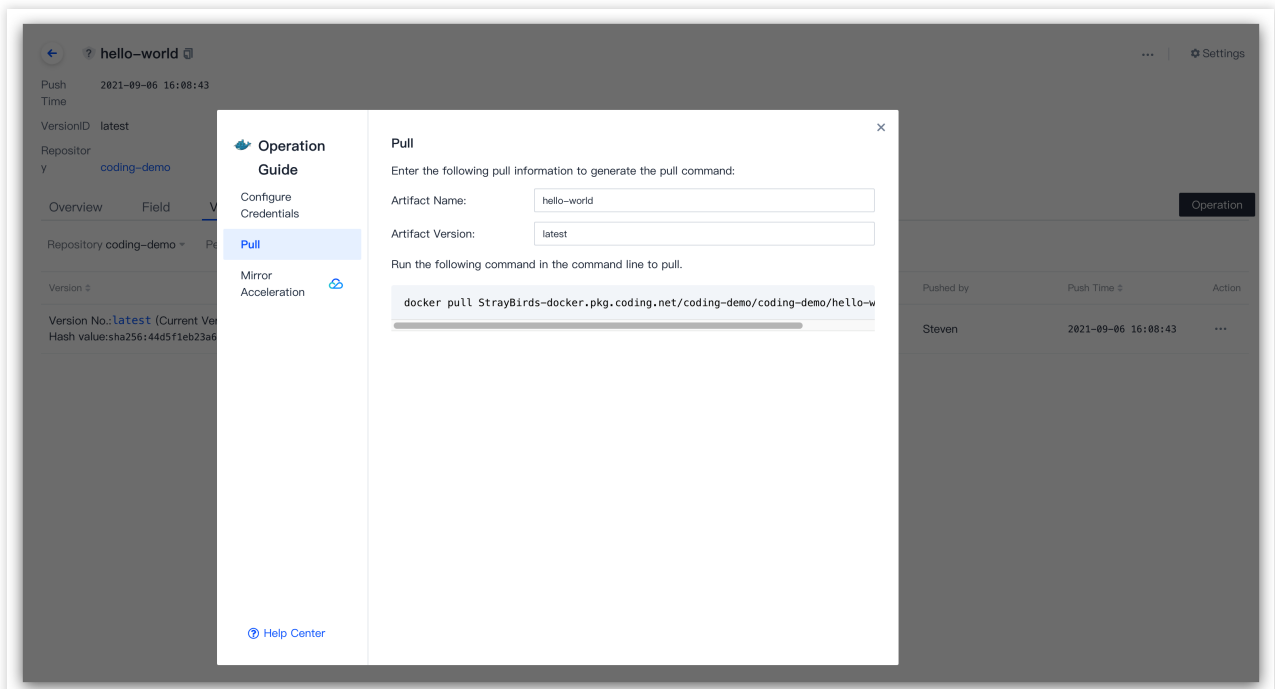


Pull an Image

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.

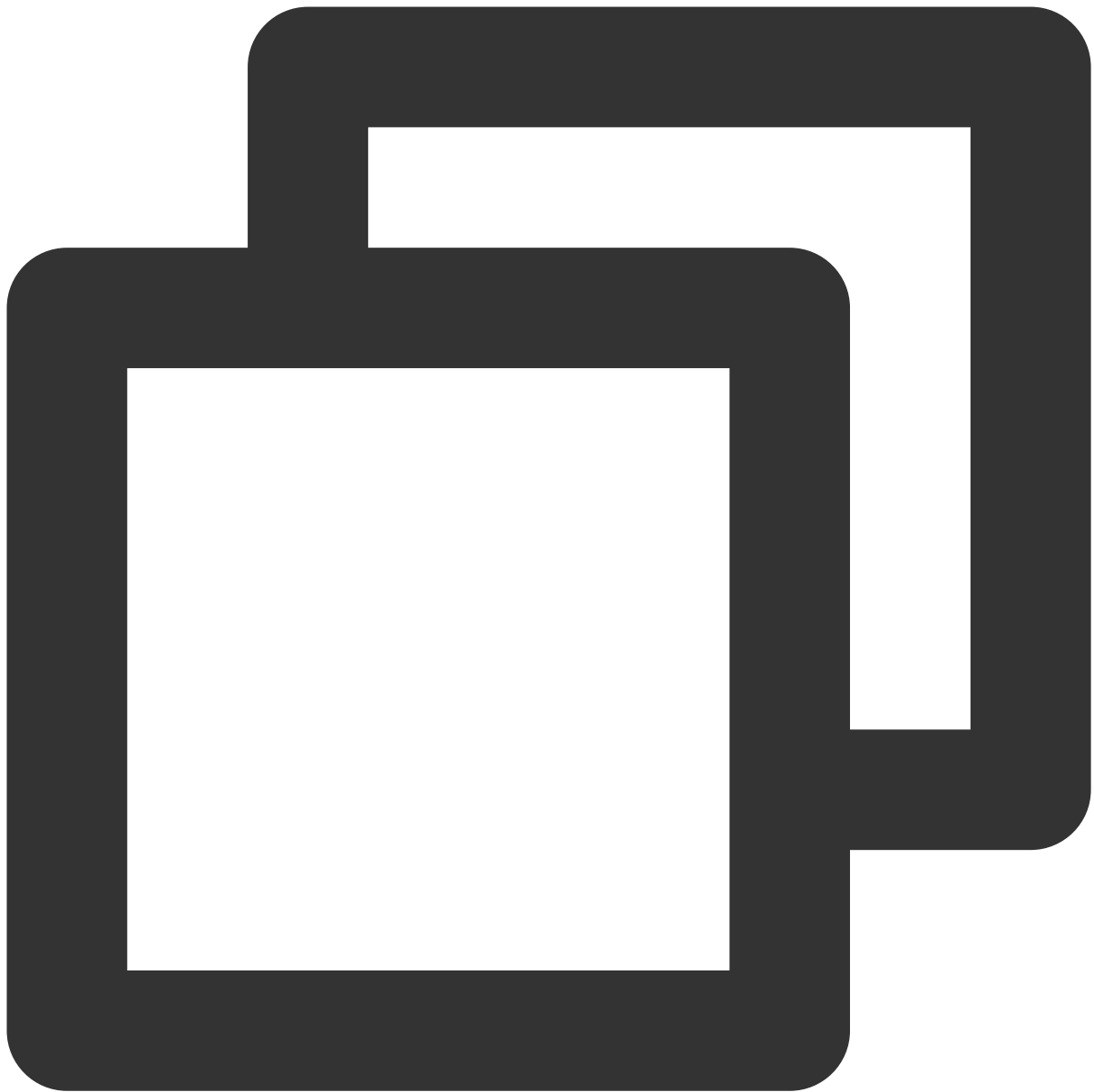
Create an artifact repository (see [Basic Operations](#)).

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.



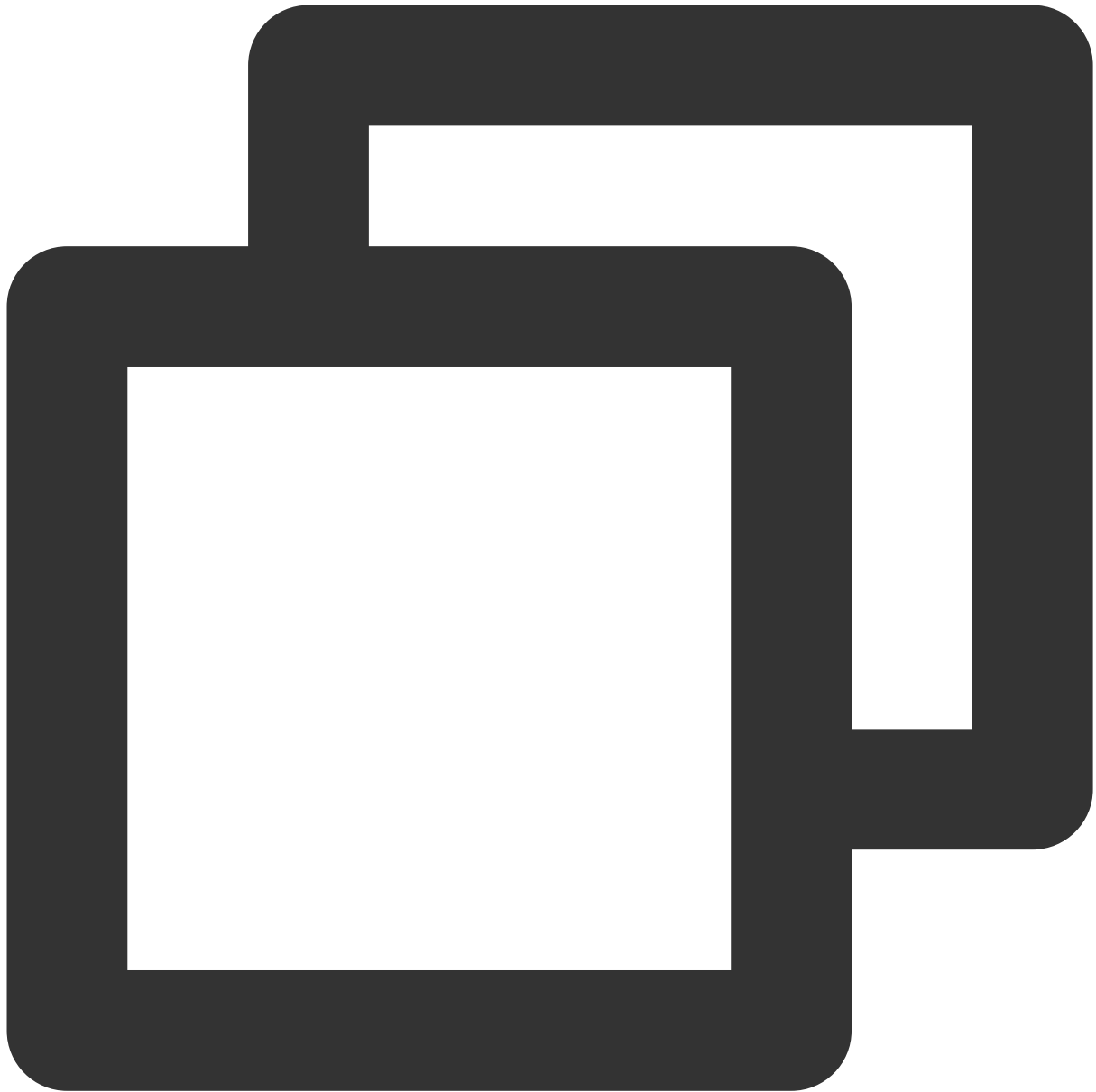
```
mkdir npm-demo
```

2. Initialize the npm project.



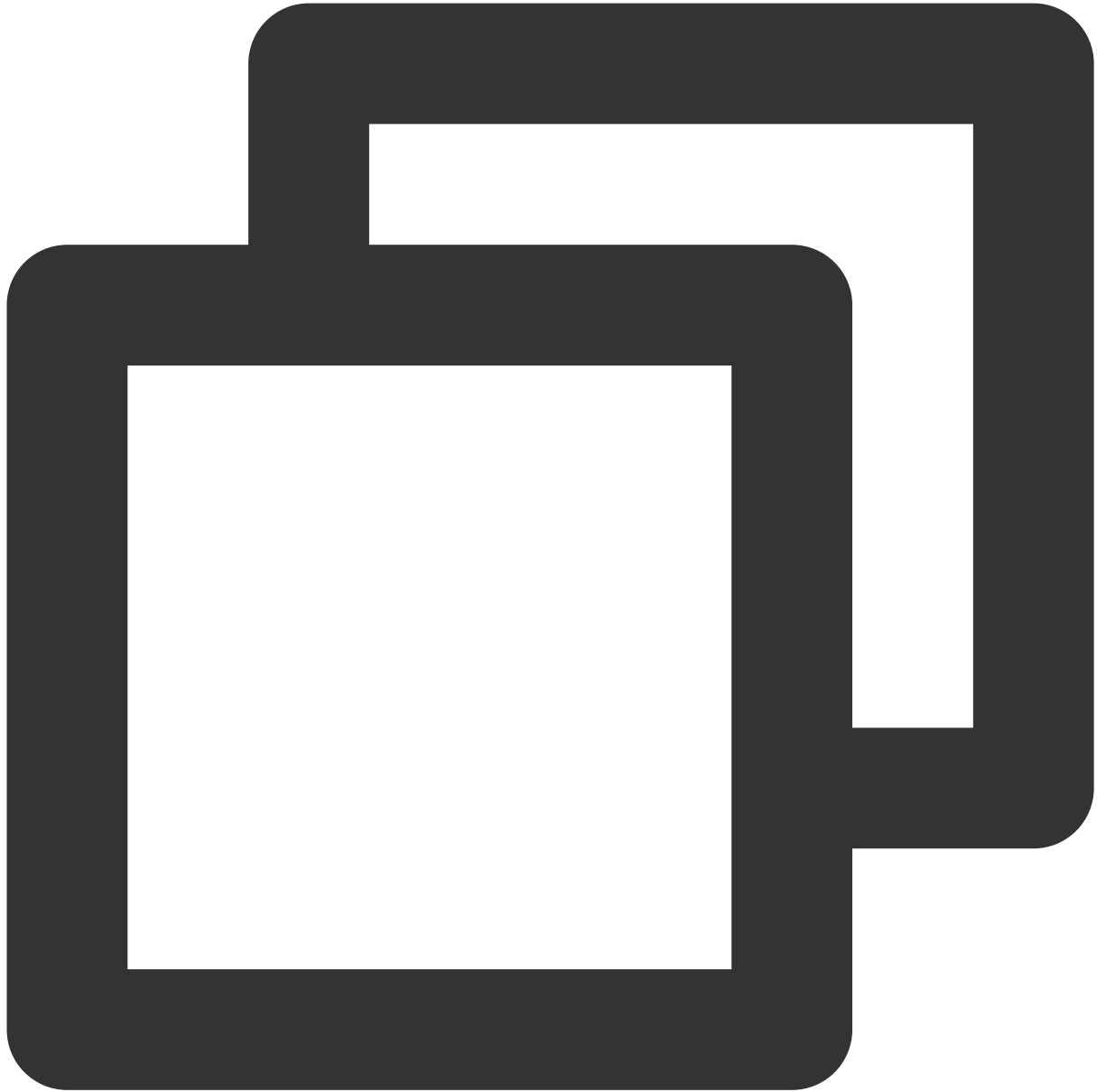
```
cd npm-demo && npm init
```

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.

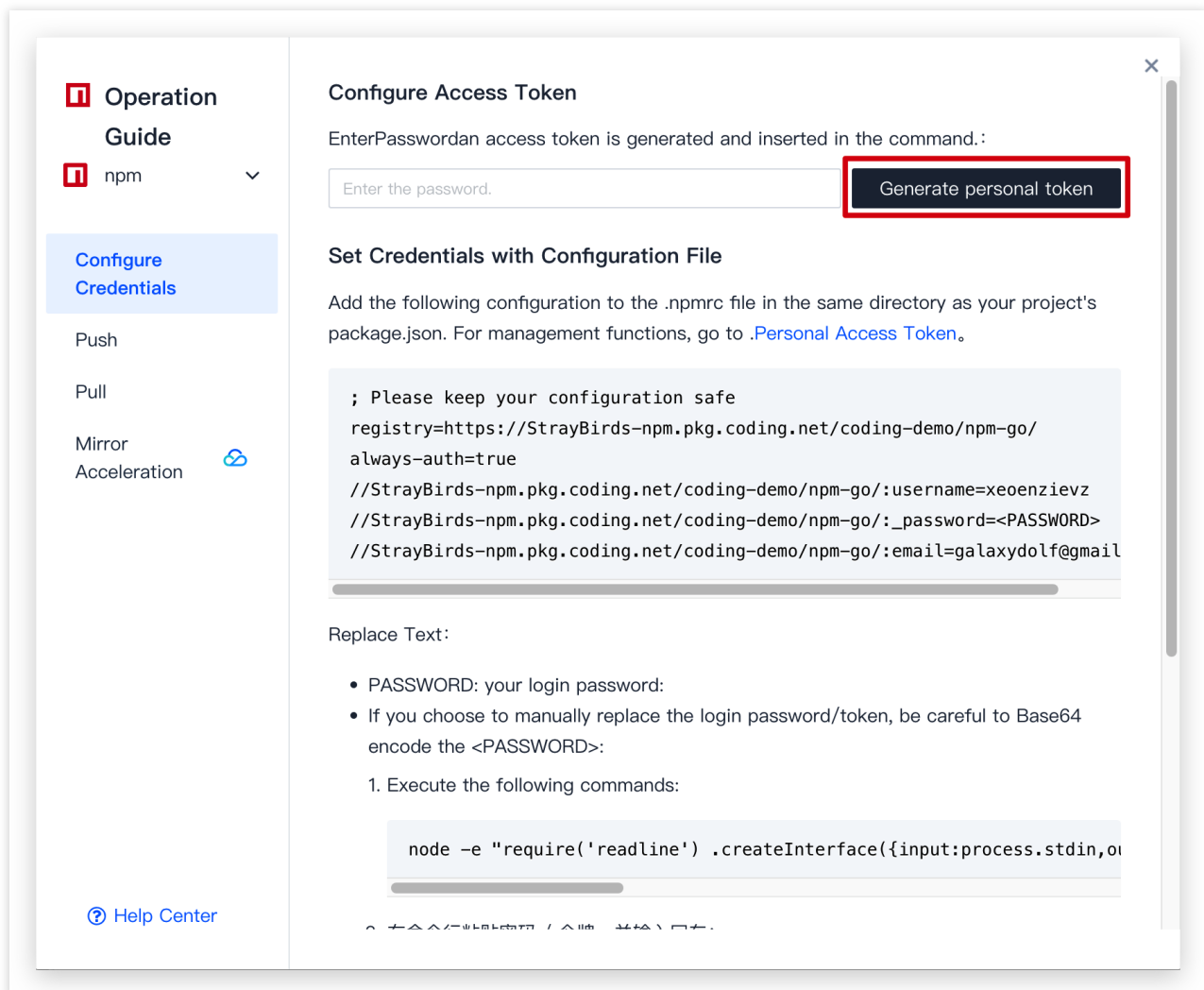
Configure authentication information in either of the following ways:

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.



Operation Guide

npm

Configure Credentials

Push

Pull

Mirror Acceleration

[Help Center](#)

Configure Access Token

Enter a password and an access token is generated and inserted in the command.:

Enter the password. **Generate personal token**

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=xoenzioevz
//StrayBirds-npm.pkg.coding.net/coding-demo/npm-go/:_password=<PASSWORD>
//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
```

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.

Operation Guide

npm

Configure Credentials

Push

Pull

Mirror Acceleration

[Help Center](#)

Configure Access Token

Enter a password and an access token is generated and inserted in the command.:

Enter the password. Generate personal token

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=xoenzievz
//StrayBirds-npm.pkg.coding.net/coding-demo/npm-go/:_password=<PASSWORD>
//StrayBirds-npm.pkg.coding.net/coding-demo/npm-go/:email=galaxydolf@gmail
```

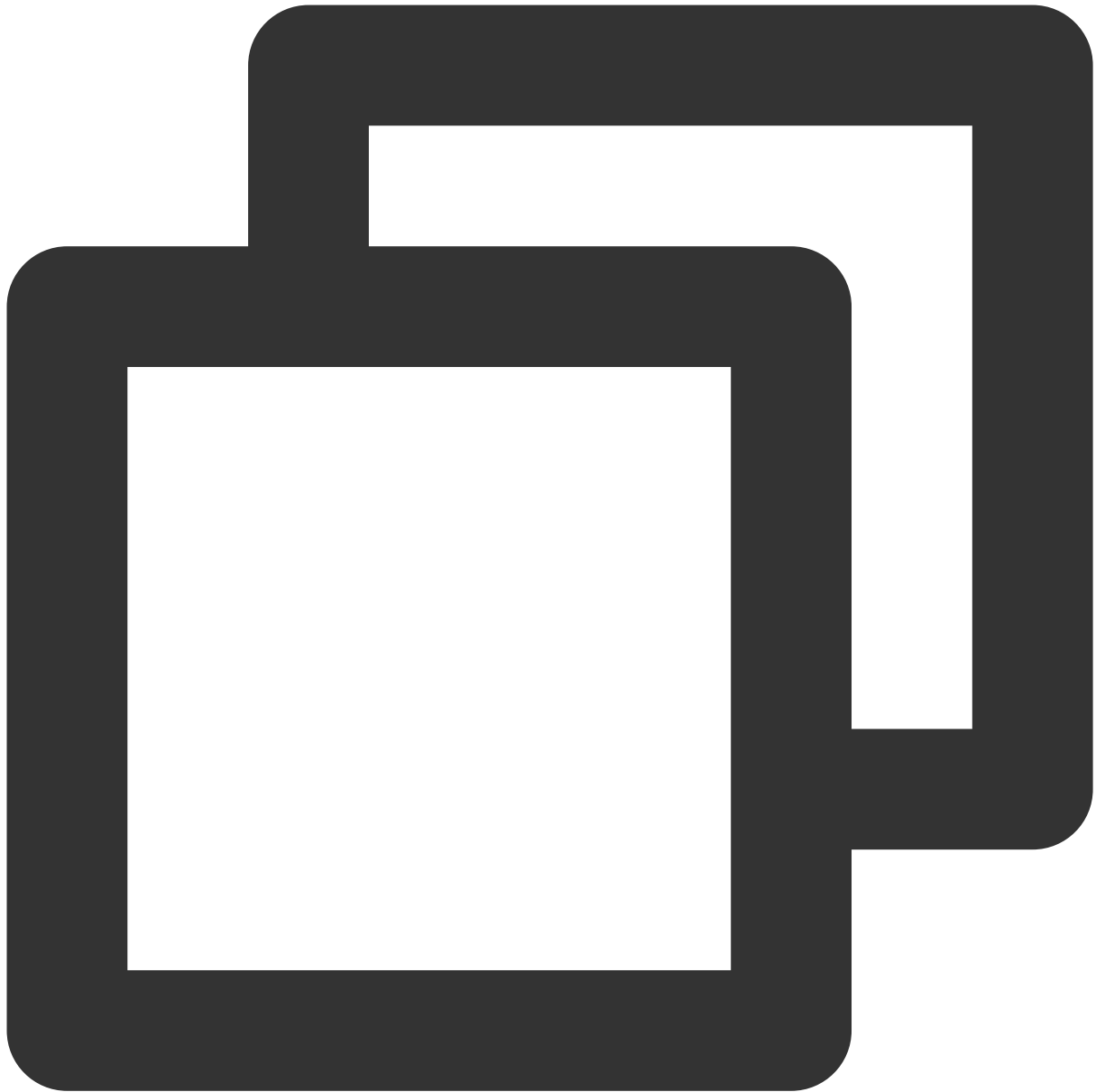
Replace text:

- If you choose to manually replace the login password/token, be careful to Base64 encode the `<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
```

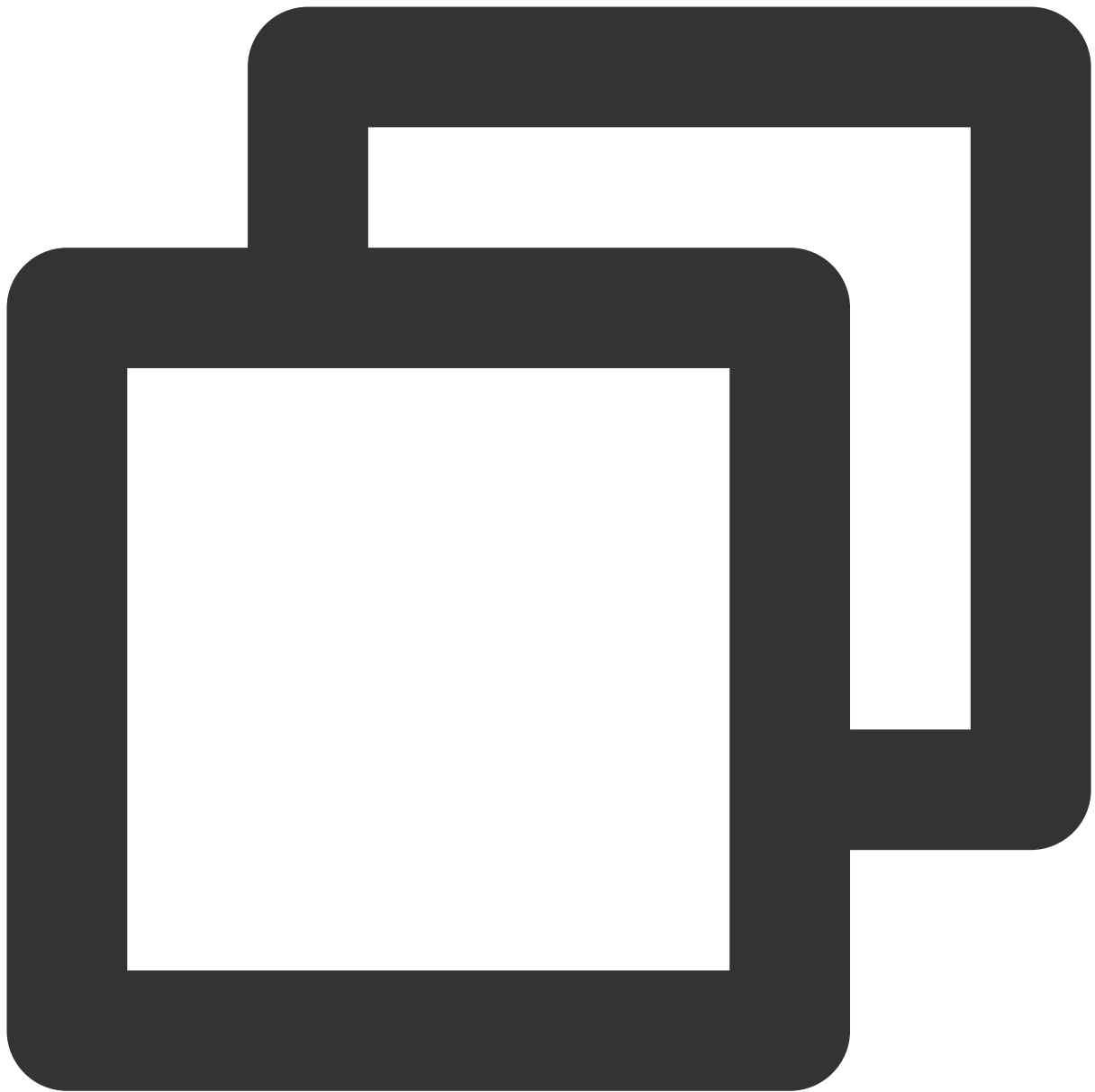
2. Run `npm login` and enter the account, password, and email address when asked.



```
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>
```

```

-MB0:demo $ npm publish --registry=https://anywhere-npm.pkg.coding.net
t/coding-demo/my-npm/
npm notice
npm notice 📦 demo@0.0.1
npm notice === Tarball Contents ===
npm notice 213B package.json
npm notice === Tarball Details ===
npm notice name:          demo
npm notice version:       0.0.1
npm notice package size:  248 B
npm notice unpacked size: 213 B
npm notice shasum:
npm notice integrity:
npm notice total files:   1
npm notice
+ demo@0.0.1
    
```

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

The screenshot shows the 'Artifact Repository' management page. On the left is a sidebar with repository types like 'generic', 'pypi', 'nuget-go', 'coco-go', 'conan-go', 'rpm-go', 'composer-go', 'helm-go', 'npm-go', and 'python-demo'. The 'npm-go' repository is selected. The main area shows the 'PackageList' for the 'npm-go' repository. A table lists various artifacts with columns for Package name, Latest Push Version, Last Updated, Number of Versions, and Action. A red box highlights the table content.

Package name	Latest Push Version	Last Updated	Number of Versions	Action
react	17.0.1	2021-01-19 15:41:56	1	...
js-tokens	4.0.0	2021-01-19 15:41:54	1	...
loose-envify	1.4.0	2021-01-19 15:41:53	1	...
object-assign	4.1.1	2021-01-19 15:41:53	1	...
mustache	4.1.0	2021-01-19 14:20:30	1	...
underscore	1.12.0	2021-01-19 14:20:30	1	...
linkify-it	2.2.0	2021-01-19 14:20:30	1	...
markdown-it	8.4.1	2021-01-19 14:20:29	1	...
backbone	1.4.0	2021-01-19 14:20:29	1	...

Pull an npm Artifact

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

Operation Guide

npm

Configure Credentials

Push

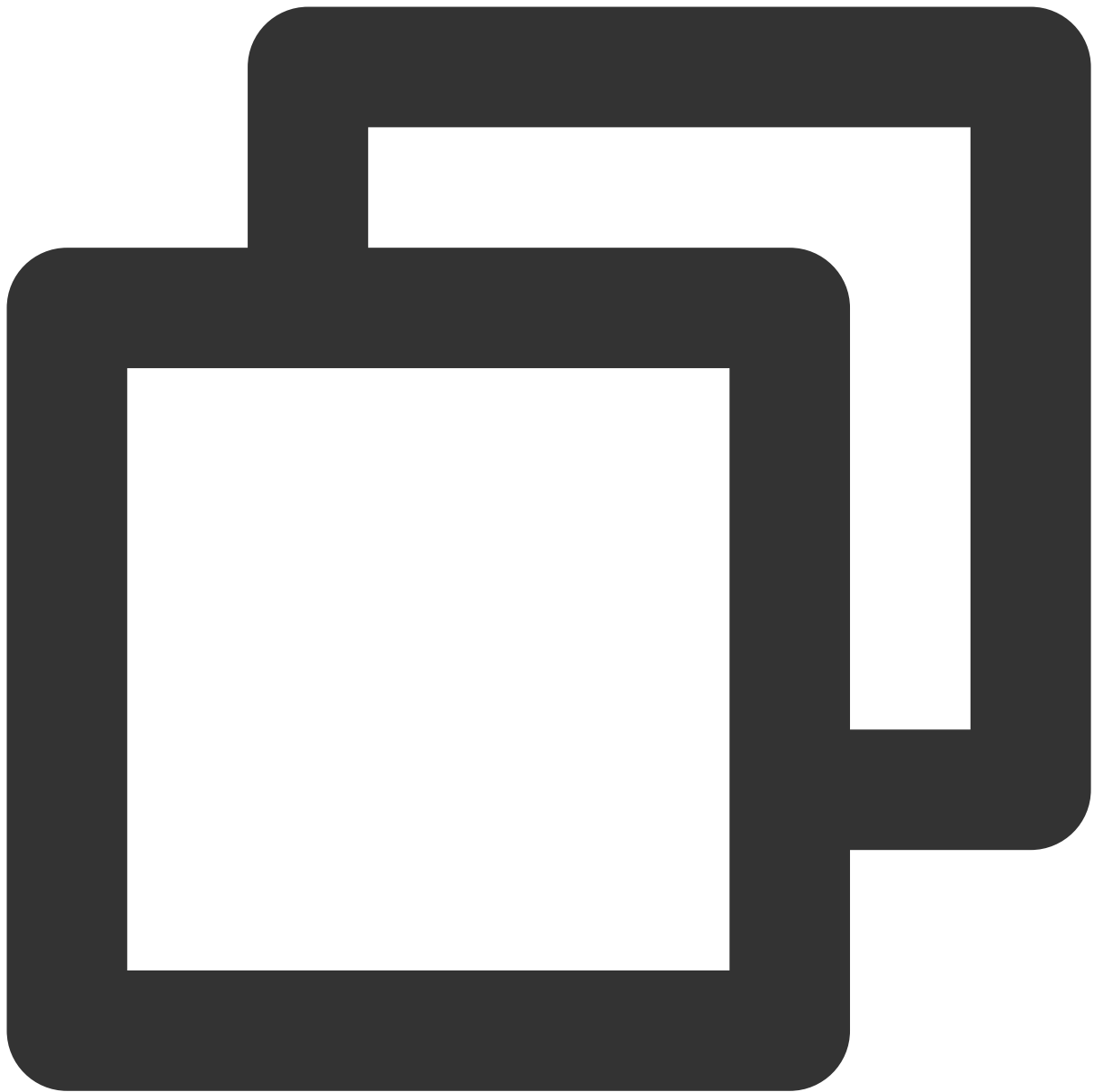
Pull

Mirror Acceleration

Pull Artifact

Enter artifact_name@artifact_version to generate a pull command:

```
npm install latest@<VERSION> --registry=https://StrayBirds-npm.pkg.coding.ne
```



```
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.

The screenshot shows the 'Repository Management' page for the 'npm-go' repository. The page includes a sidebar with a list of repositories, a main content area with a 'PackageList' table, and a 'Proxy Settings' modal window.

PackageList Table:

Package name	Latest Push Version
react	17.0.1
js-tokens	4.0.0
loose-envify	1.4.0
object-assign	4.1.1
mustache	4.1.0
underscore	1.12.0
linkify-it	2.2.0
markdown-it	8.4.1
backbone	1.4.0

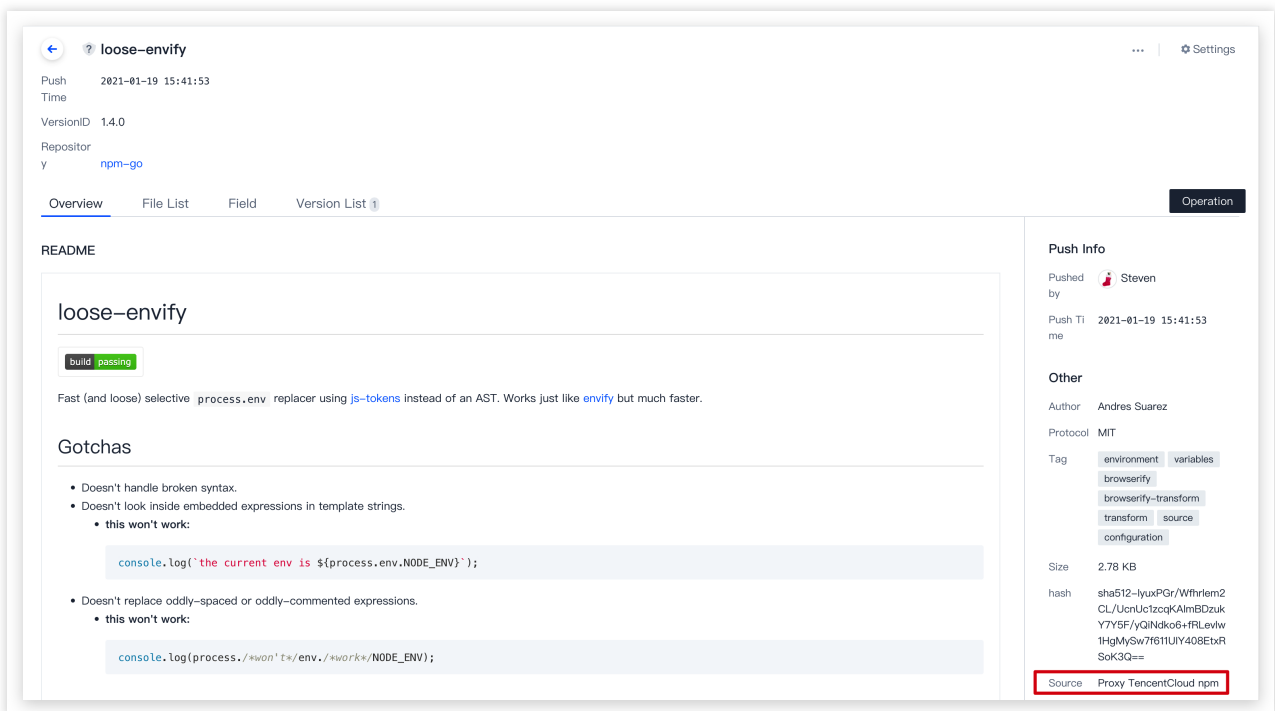
Proxy Settings Table:

Source	URL	Action
TencentCloud npm	http://mirrors.cloud.tencent.com/npm/	Configuration
cnpm	https://registry.npm.taobao.org/	Configuration
npmsjs	https://registry.npmsjs.org/	Configuration

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**.

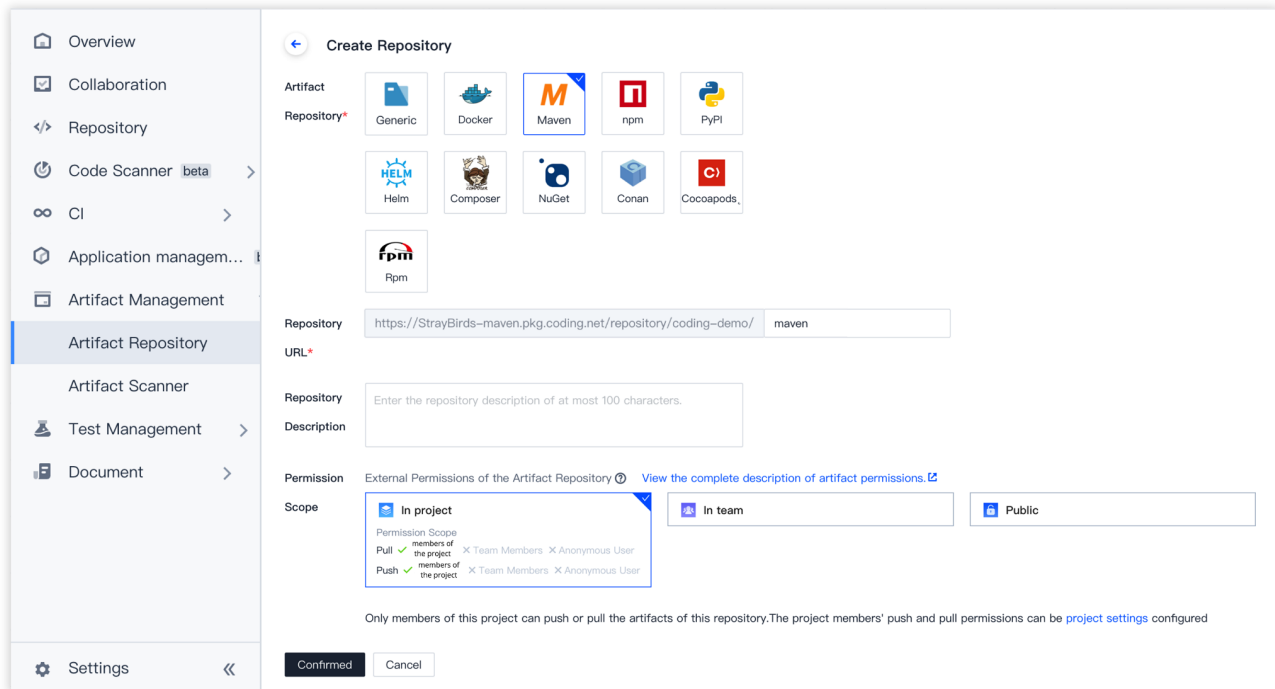
Select Maven as the repository type.

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.

M Operation Guide

M Apache Ma... ▾

Configure Credentials

Push

Pull

Mirror Acceleration

[Help Center](#)

Configure Access Token

Enter a password and an access token is generated and inserted in the command.:

Enter the password. **Generate personal token**

Set Credentials

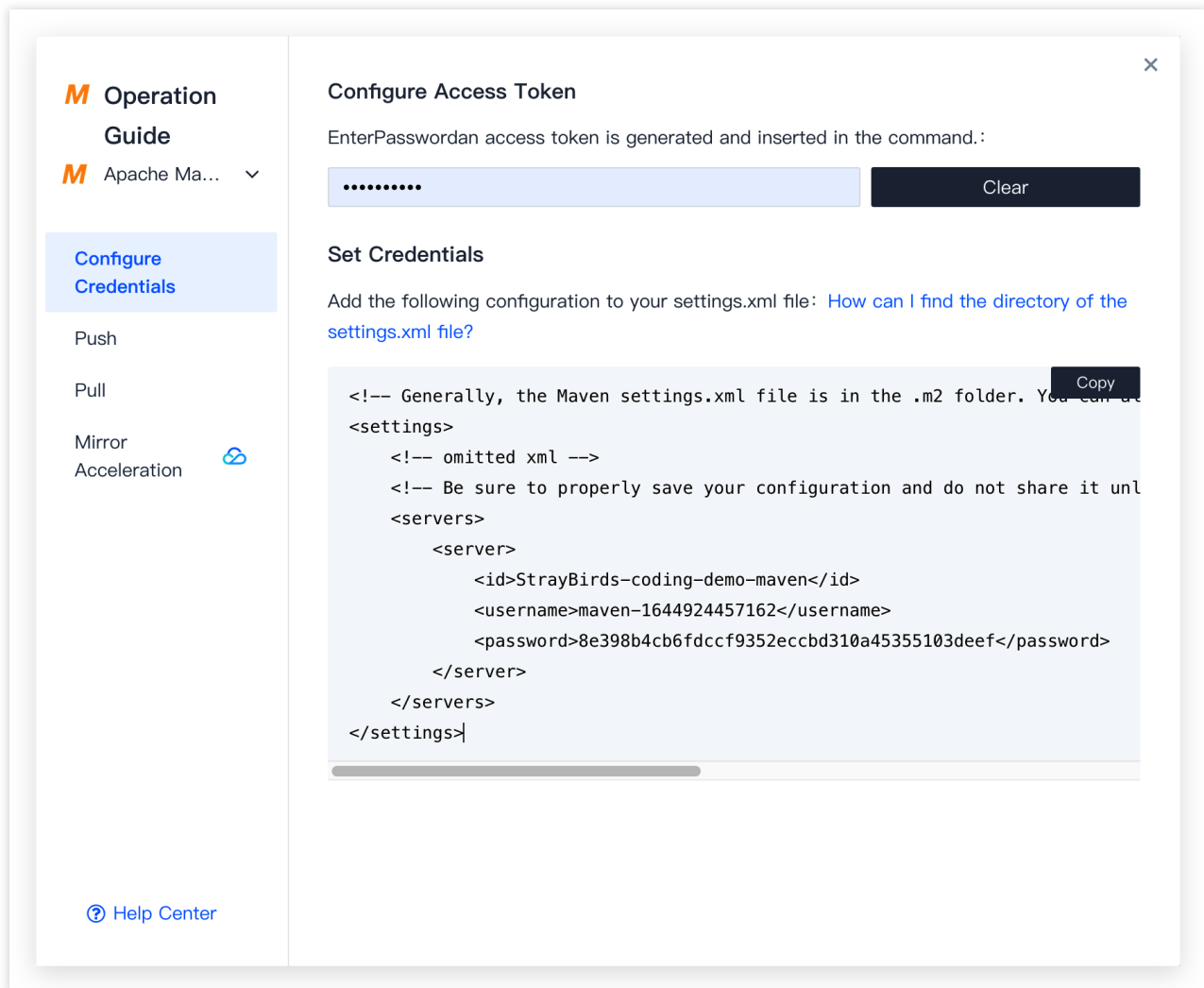
Add the following configuration to your settings.xml file: [How can I find the directory of the settings.xml file?](#)

```
<!-- Generally, the Maven settings.xml file is in the .m2 folder. You can al
<settings>
  <!-- omitted xml -->
  <!-- Be sure to properly save your configuration and do not share it unl
  <servers>
    <server>
      <id>StrayBirds-coding-demo-maven</id>
      <username>galaxydolf@gmail.com</username>
      <password>[PASSWORD]</password>
    </server>
  </servers>
</settings>
```

Replace text:

- PASSWORD: Your login password

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




M Operation Guide

M Apache Ma... ▾

Configure Credentials

Push

Pull

Mirror Acceleration 

[Help Center](#)

Configure Access Token

Enter password and an access token is generated and inserted in the command.:

..... Clear

Set Credentials

Add the following configuration to your settings.xml file: [How can I find the directory of the settings.xml file?](#)

```
<!-- Generally, the Maven settings.xml file is in the .m2 folder. You can use the following command to find the directory: mvn help:effective-settings.xml -DoutputDirectory=. -DoutputFile=settings.xml -->
<settings>
  <!-- omitted xml -->
  <!-- Be sure to properly save your configuration and do not share it unless necessary -->
  <servers>
    <server>
      <id>StrayBirds-coding-demo-maven</id>
      <username>maven-1644924457162</username>
      <password>8e398b4cb6fdccf9352eccbd310a45355103deef</password>
    </server>
  </servers>
</settings>
```

Copy

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.

M Operation Guide

M Apache Ma...

Configure Credentials

Push

Pull

Mirror Acceleration

[Help Center](#)

Configure Access Token

Enter a password and an access token is generated and inserted in the command.:

Enter the password. Generate personal token

Set Credentials

Add the following configuration to your settings.xml file: [How can I find the directory of the settings.xml file?](#)

```
<!-- Generally, the Maven settings.xml file is in the .m2 folder. You can al
<settings>
  <!-- omitted xml -->
  <!-- Be sure to properly save your configuration and do not share it unl
  <servers>
    <server>
      <id>StrayBirds-coding-demo-maven</id>
      <username>[REDACTED]@gmail.com</username>
      <password>[PASSWORD]</password>
    </server>
  </servers>
</settings>
```

Replace text:

- PASSWORD: Your login password

Compile and Upload a Maven Artifact

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

```
:0.0.1-SNAPSHOT $ ls -l
total 32
-rw-r--r--  1  staff  220 10 28 13:42 _remote.repositories
-rw-r--r--  1  staff 2254 10 28 13:42 demo-for-artifacts-0.0.1-SNAPSHOT.jar
-rw-r--r--  1  staff  766 10 28 11:03 demo-for-artifacts-0.0.1-SNAPSHOT.pom
-rw-r--r--  1  staff  710 10 28 13:42 maven-metadata-local.xml
:0.0.1-SNAPSHOT $ pwd
/Users/ / .m2/repository/coding/demo-for-artifacts/0.0.1-SNAPSHOT
```

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

The screenshot shows a web interface for configuring a Maven push. On the left is a sidebar with navigation options: 'Operation Guide', 'Apache Ma...', 'Configure Credentials', 'Push' (highlighted), 'Pull', 'Mirror Acceleration', and 'Help Center'. The main content area is titled 'Push' and contains three input fields: 'Artifact GROUP_ID' with value '1', 'Artifact ARTIFACT_ID' with value '1', and 'Artifact VERSION' with value 'latest'. Below these fields is a numbered instruction: '1. Please add the following configuration to your pom.xml file:'. A code block displays the XML configuration to be added to the pom.xml file, including repository management details.

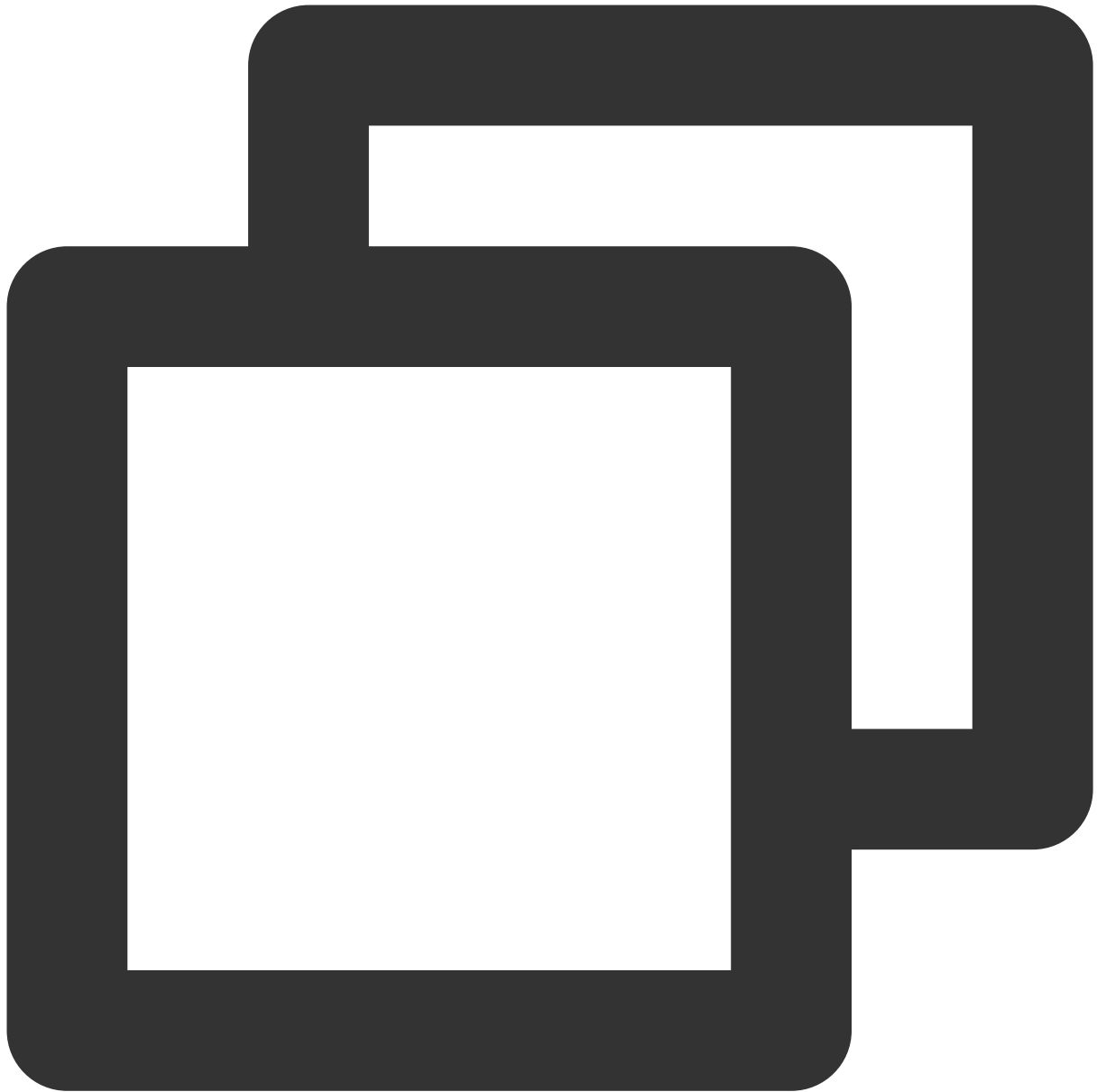
```
<project>
  <!-- Essential attributes -->
  <groupId>1</groupId>
  <artifactId>1</artifactId>
  <version>latest</version>

  <!-- omitted xml -->
  <distributionManagement>
    <repository>
      <!--Must be consistent with the ID in settings.xml.-->
      <id>StrayBirds-coding-demo-maven</id>
      <name>maven</name>
      <url>https://StrayBirds-maven.pkg.coding.net/repository/co
    </repository>
  </distributionManagement>
</project>
```

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
1 <project xmlns="http://maven.apache.org/POM/4.0.0"
2   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.x
4   <modelVersion>4.0.0</modelVersion>
5
6   <groupId>coding</groupId>
7   <artifactId>demo-for-artifacts</artifactId>
8   <version>0.0.1-SNAPSHOT</version>
9   <packaging>jar</packaging>
10
11  <name>demo-for-artifacts</name>
12  <url>http://maven.apache.org</url>
13
14  <properties>
15    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
16  </properties>
17
18  <dependencies>
19    <dependency>
20      <groupId>junit</groupId>
21      <artifactId>junit</artifactId>
22      <version>3.8.1</version>
23      <scope>test</scope>
24    </dependency>
25  </dependencies>
26
27  <distributionManagement>
28    <repository>
29      <!-- settings.xml -->
30      <id>anywhere-coding-demo-my-maven</id>
31      <name>my-maven</name>
32      <url>https://anywhere-maven.pkg.coding.net/repository/coding-demo/my-maven/</url>
33    </repository>
34  </distributionManagement>
35 </project>
36
```

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.

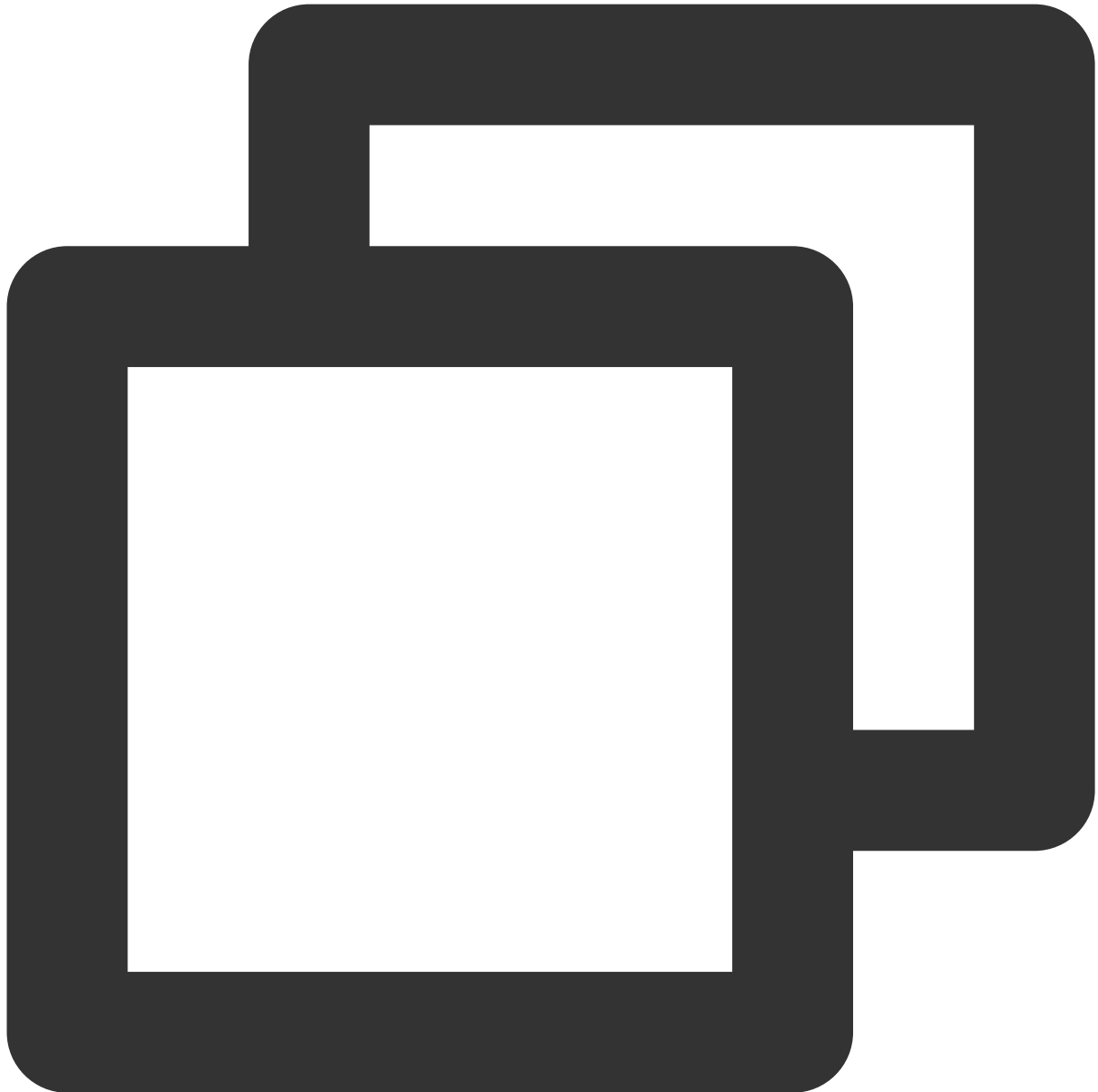


```
mvn deploy -s "/Users/somebody/software/apache-maven-3.6.2/conf/settings.xml"
```

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:

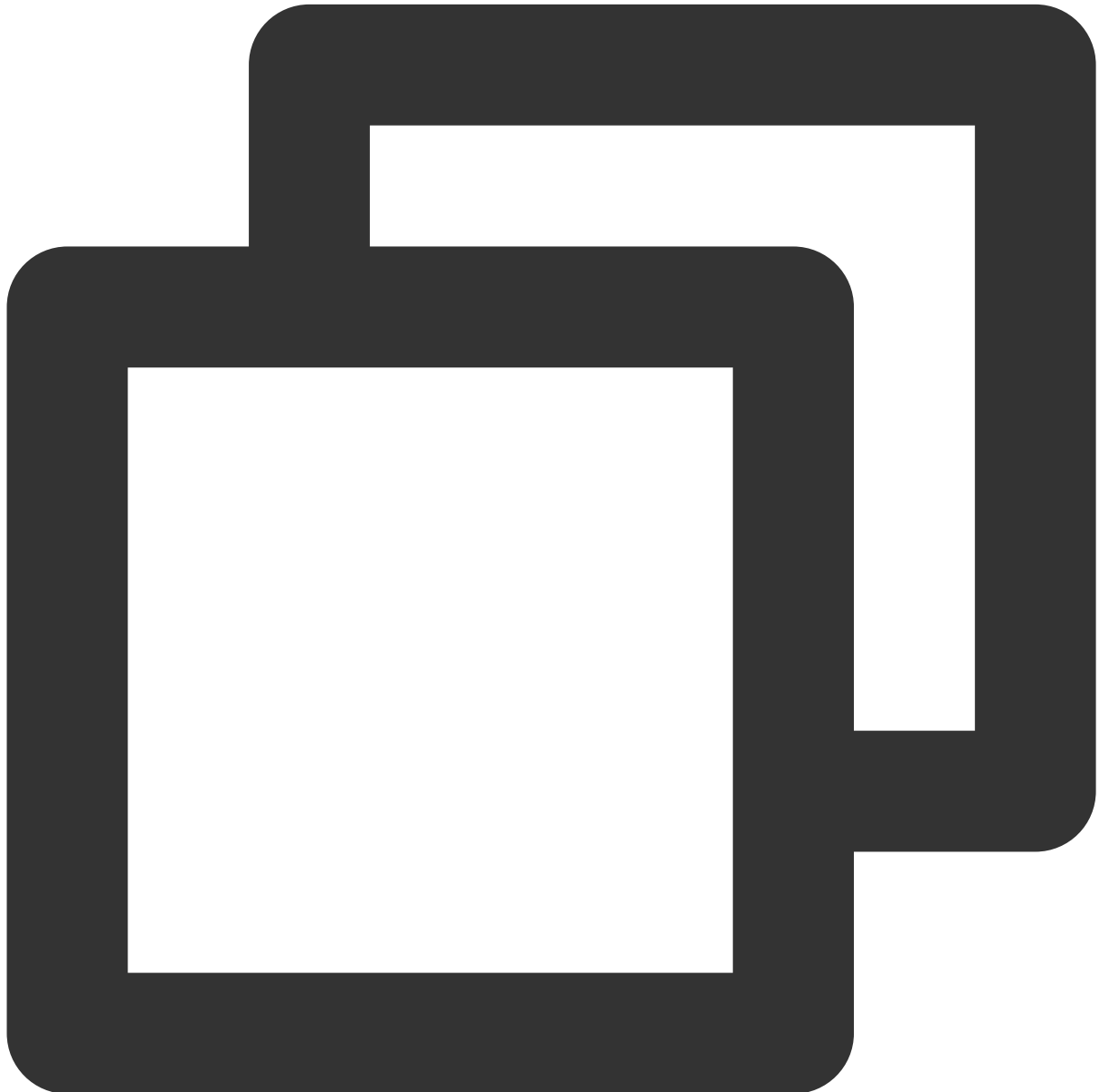


```
mvn deploy:deploy-file -Durl=file://C:\m2-repo \\  
-DrepositoryId=some.id \\  
-Dfile=your-artifact-1.0.jar \\  
[-DpomFile=your-pom.xml] \\  
[-DgroupId=org.some.group] \\  
[-DartifactId=your-artifact] \\  
[-Dversion=1.0] \\  
[-Dpackaging=jar] \\  

```

```
[-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>
```

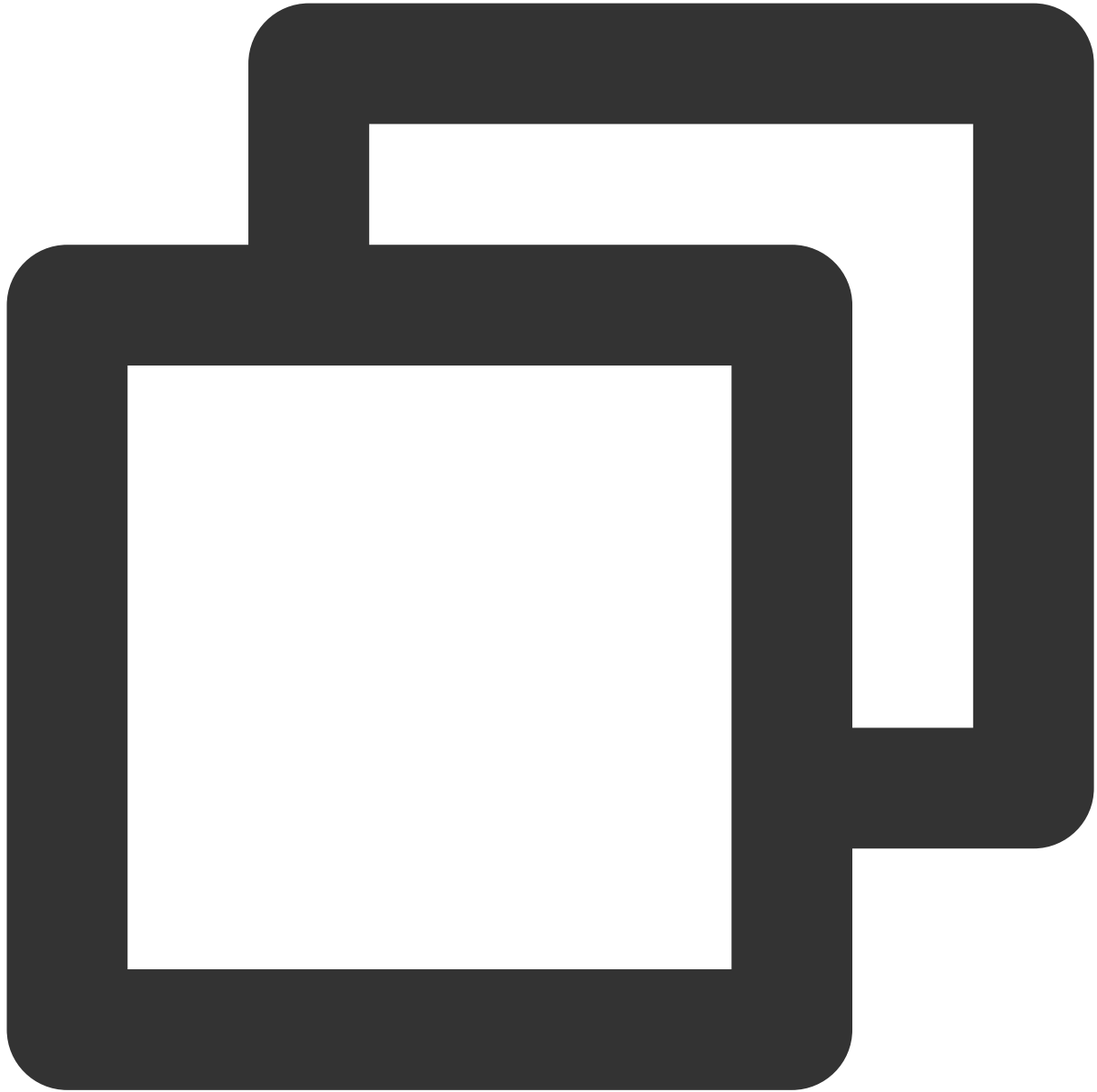
```
<name>tencent</name>
<url>https://coding-public-maven.pkg.coding.net/repository/tenc
<releases>
  <enabled>>true</enabled>
</releases>
<snapshots>
  <enabled>>true</enabled>
</snapshots>
</repository>
</repositories>
</profile>
</profiles>
</settings>
```

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.

```
pom.xml src      target
demo-for-artifacts-consumer $ mvn install
[INFO] Scanning for projects...
[INFO]
[INFO] -----< coding:demo-for-artifacts-consumer >-----
[INFO] Building demo-for-artifacts-consumer 0.0.1-SNAPSHOT
[INFO] -----[ jar ]-----
Downloading from anywhere-coding-demo-my-maven: https://anywhere-maven.pkg.coding.net/repos
itory/coding-demo/my-maven/coding/demo-for-artifacts/0.0.1-SNAPSHOT/maven-metadata.xml
Downloaded from anywhere-coding-demo-my-maven: https://anywhere-maven.pkg.coding.net/reposi
tory/coding-demo/my-maven/coding/demo-for-artifacts/0.0.1-SNAPSHOT/maven-metadata.xml (774
B at 332 B/s)
```

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](#)).

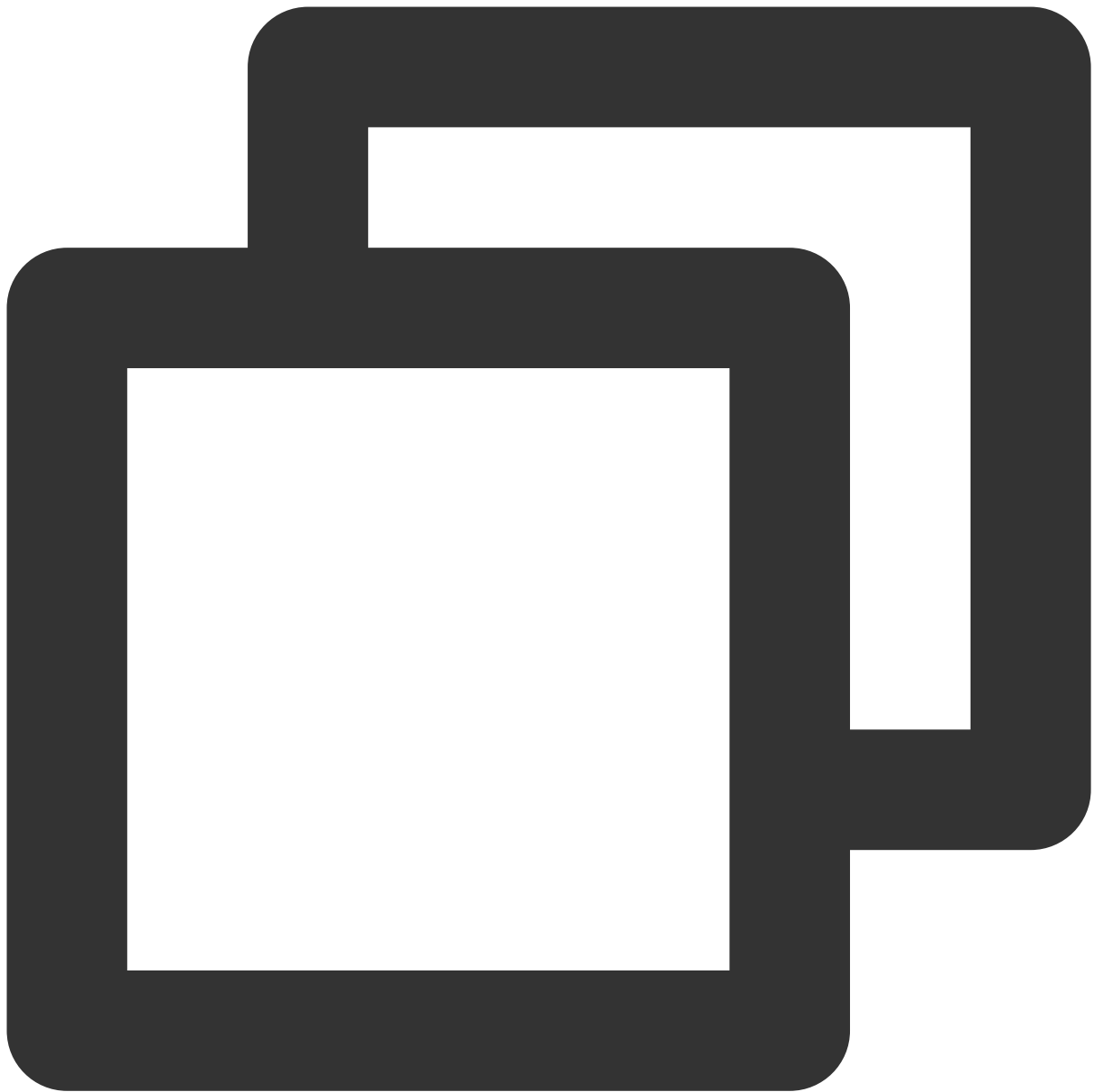
Select Helm as the repository type.

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.



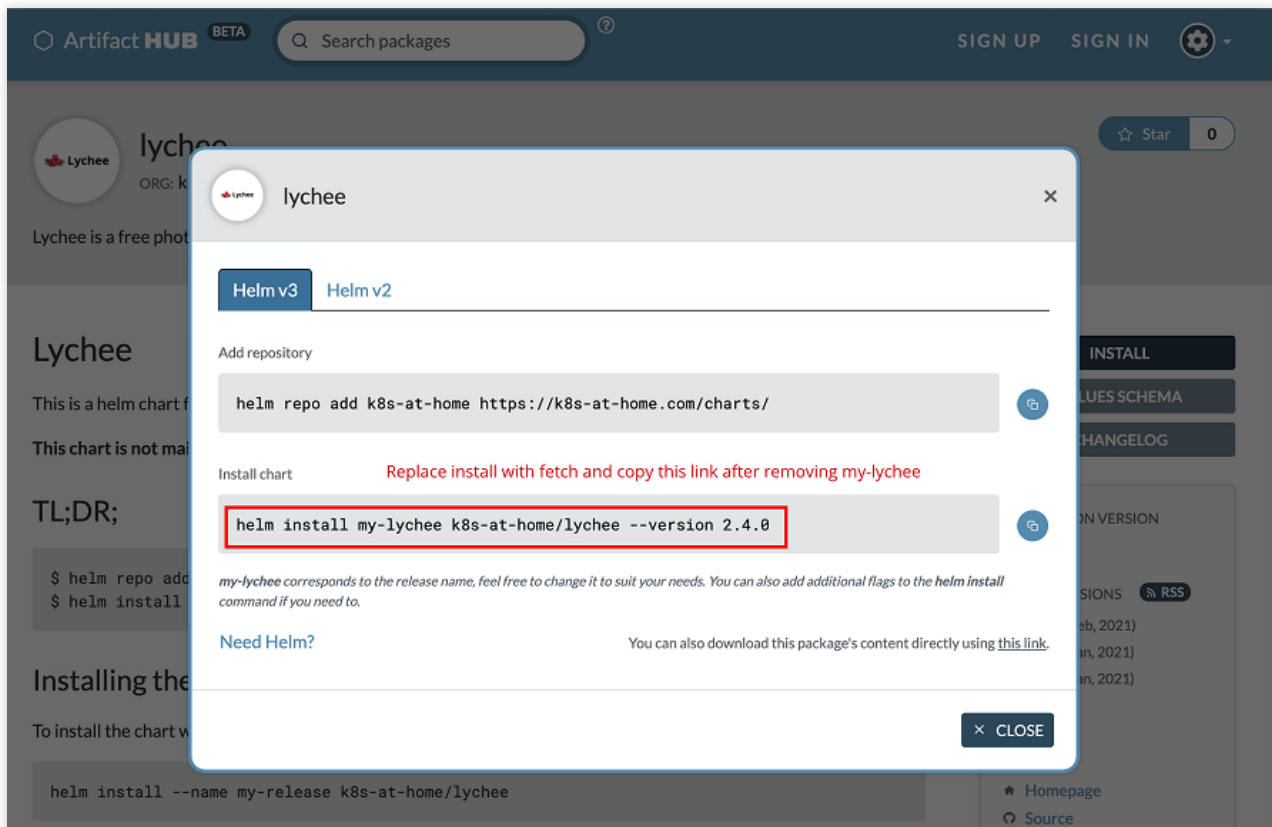
```
helm package [name]
```

Method 2: pull an artifact from Artifact Hub

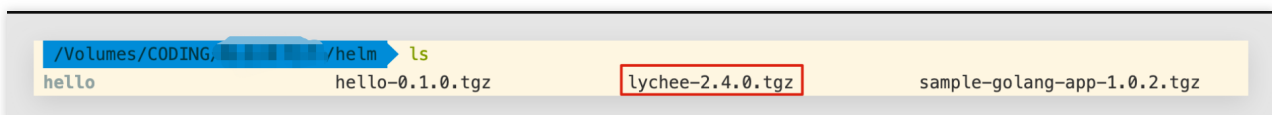
Search for a Helm chart and run the download command in a local directory.



```
$ 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 screenshot shows a dialog window titled "Configure Access Token" with a close button (X) in the top right corner. On the left is a sidebar with the "HELM" logo and "Operation Guide" header. Below it, "Helm + COD..." is selected. The sidebar has three items: "Configure Credentials" (highlighted in blue), "Push", and "Pull". The main content area has the title "Configure Access Token" and a subtitle "Enter Password and an access token is generated and inserted in the command.:". Below this is a text input field containing "Enter the password." and a "Generate personal token" button. Underneath is the "Set Credentials" section with two steps: 1. "Install CODING Helm plugin." followed by a code block:

```
helm plugin install https://e.coding.net/coding-public/helm-push
```

 2. "Please execute the following command on the command line to configure the artifacts warehouse credentials:" followed by a code block:

```
helm repo add --username galaxydolf@gmail.com --password <PASSWORD> helm
```

The following example uses the CODING Helm plugin:

This screenshot is similar to the one above but shows the next step in the process. The "Generate personal token" button has been replaced by a "Clear" button. The text input field now contains a series of dots representing a masked access token. The "Set Credentials" section remains the same, but the code block for the second step now shows the generated token:

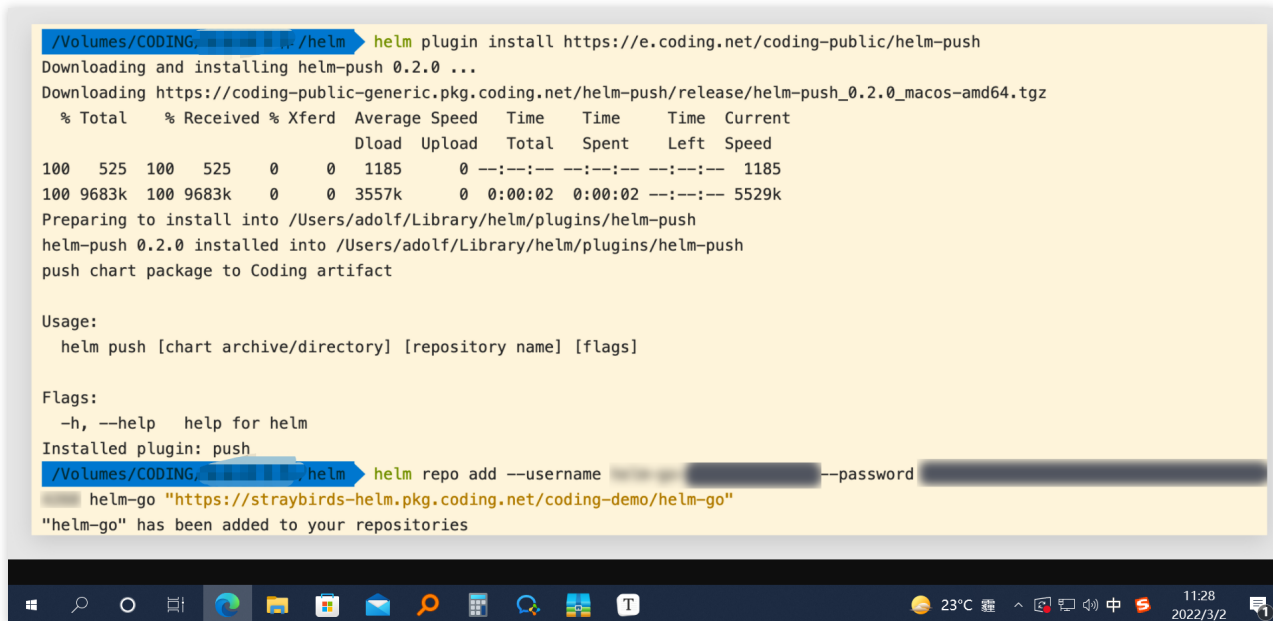
```
helm repo add --username helm-go-1644979353912 --password 783a0ff79e42b4
```

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.

```
/Volumes/CODING: helm plugin install https://e.coding.net/coding-public/helm-push
Downloading and installing helm-push 0.2.0 ...
Downloading https://coding-public-generic.pkg.coding.net/helm-push/release/helm-push_0.2.0_macos-amd64.tgz
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload  Total   Spent    Left   Speed
100  525    100  525    0    0   1185      0  --:--:-- --:--:-- --:--:-- 1185
100 9683k   100 9683k    0    0  3557k      0  0:00:02 0:00:02 --:--:-- 5529k
Preparing to install into /Users/adolf/Library/helm/plugins/helm-push
helm-push 0.2.0 installed into /Users/adolf/Library/helm/plugins/helm-push
push chart package to Coding artifact

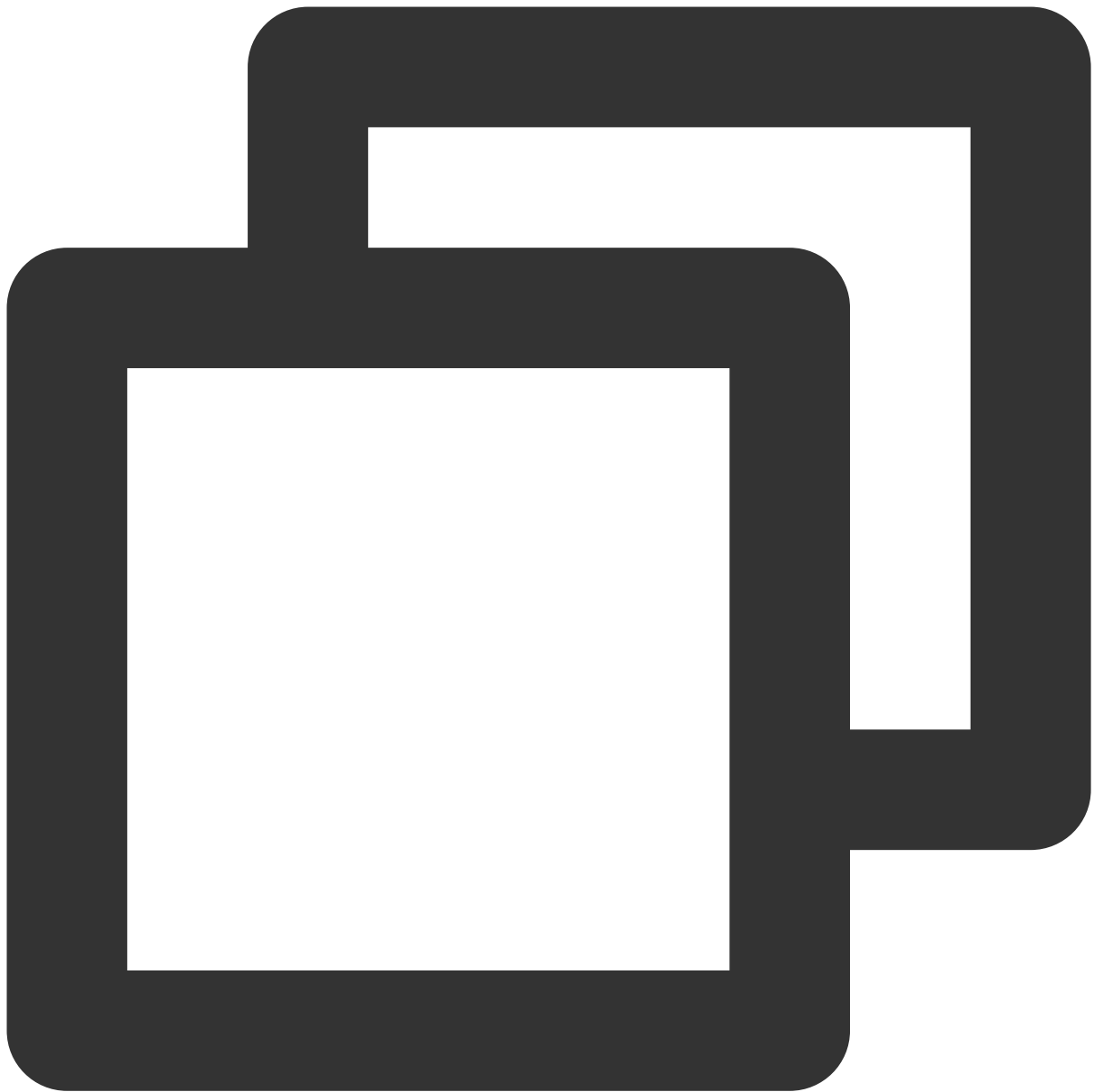
Usage:
  helm push [chart archive/directory] [repository name] [flags]

Flags:
  -h, --help  help for helm
Installed plugin: push
/Volumes/CODING: helm repo add --username  --password 
helm-go "https://straybirds-helm.pkg.coding.net/coding-demo/helm-go"
"helm-go" has been added to your repositories
```



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]
```

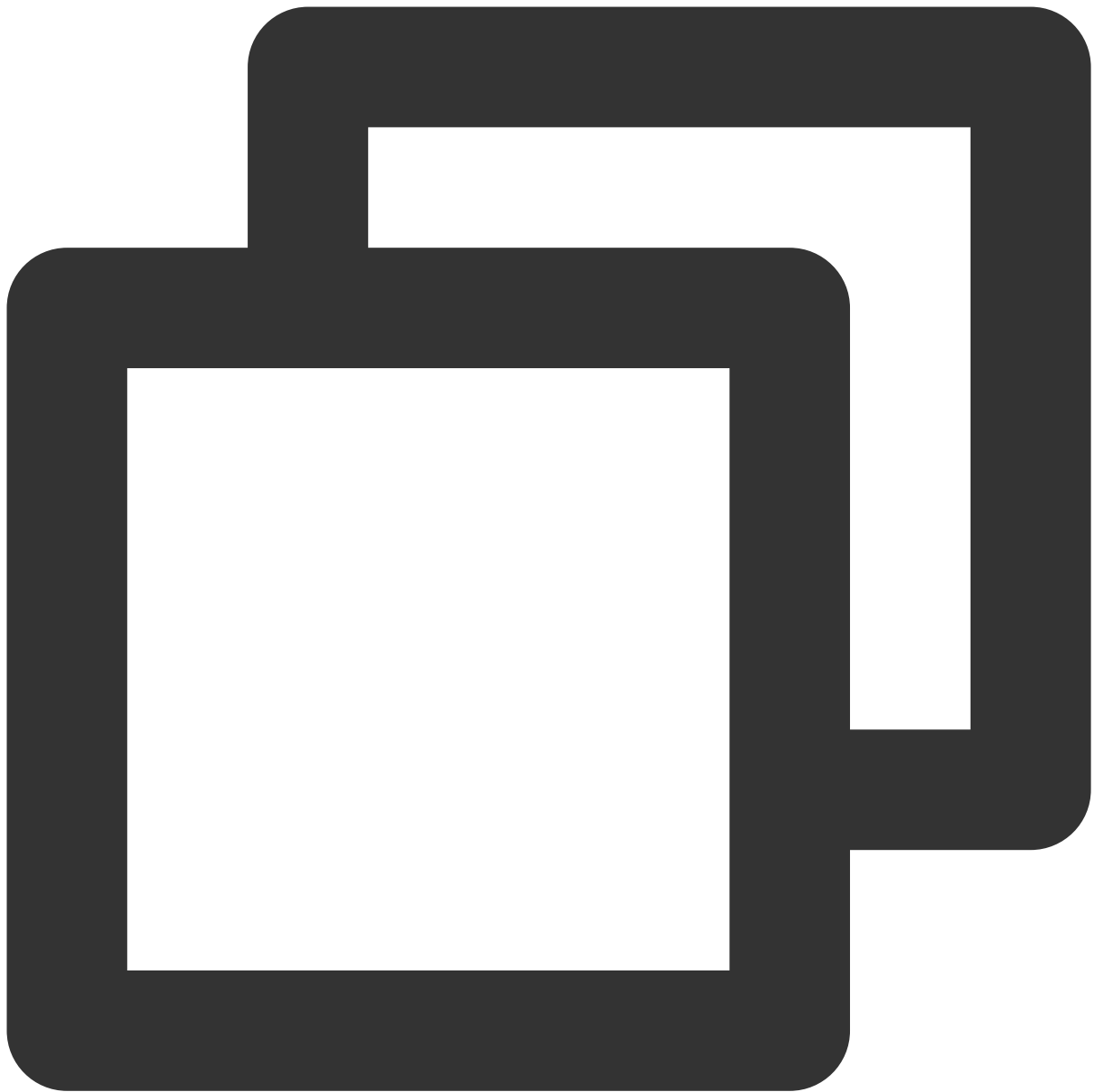
After the artifact is pushed successfully, refresh the page to view the latest artifacts.

The screenshot displays the 'Repository Management' page for an artifact repository named 'helm-go'. The interface includes a sidebar with a list of repository types (generic, pypi, nuget-go, coco-go, conan-go, rpm-go, composer-go, helm-go, npm-go, python-demo) and a main content area for the selected 'helm-go' repository. The main area shows the repository type as 'Helm', its permission as 'In project', and options to 'Set Repository', 'Proxy Settings', and 'Version Overwrite Policy'. Below this, the 'PackageList' section features a search bar and a table of packages. The table lists two packages: 'lychee' and 'hello', both with a latest push version of 2.4.0 and 0.1.0 respectively, and a last updated time of 2021-02-08. The interface also includes a pagination control showing '1-2' items and a 'Total 2' count.

Package name	Latest Push Version	Last Updated	Number of Versions	Action
lychee	2.4.0	2021-02-08 14:09:37	1	...
hello	0.1.0	2021-02-08 14:08:25	1	...

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:

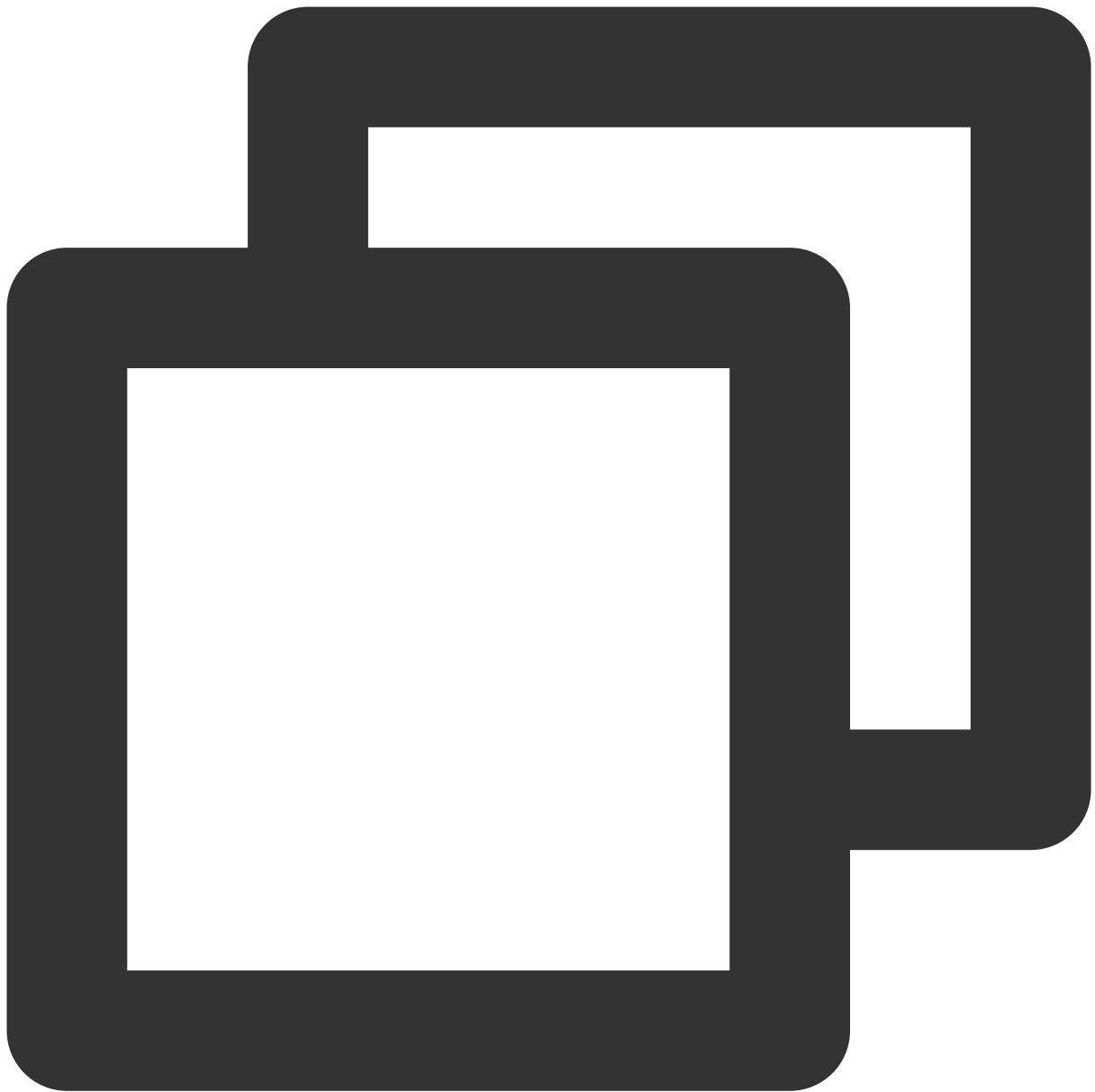
Install Python3.

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.



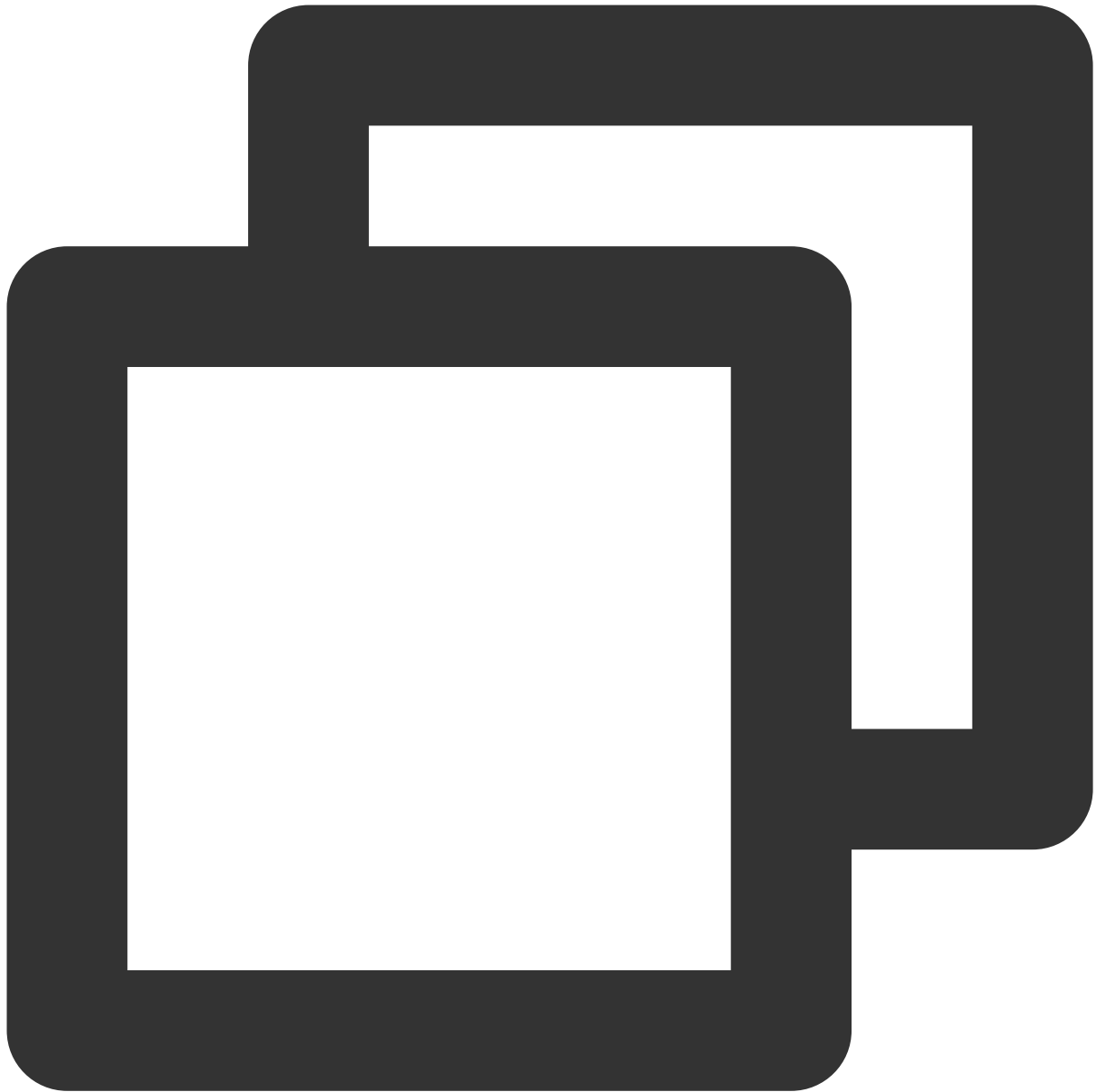
```
mkdir -p demo/example_pkg/__init__.py
```

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



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

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

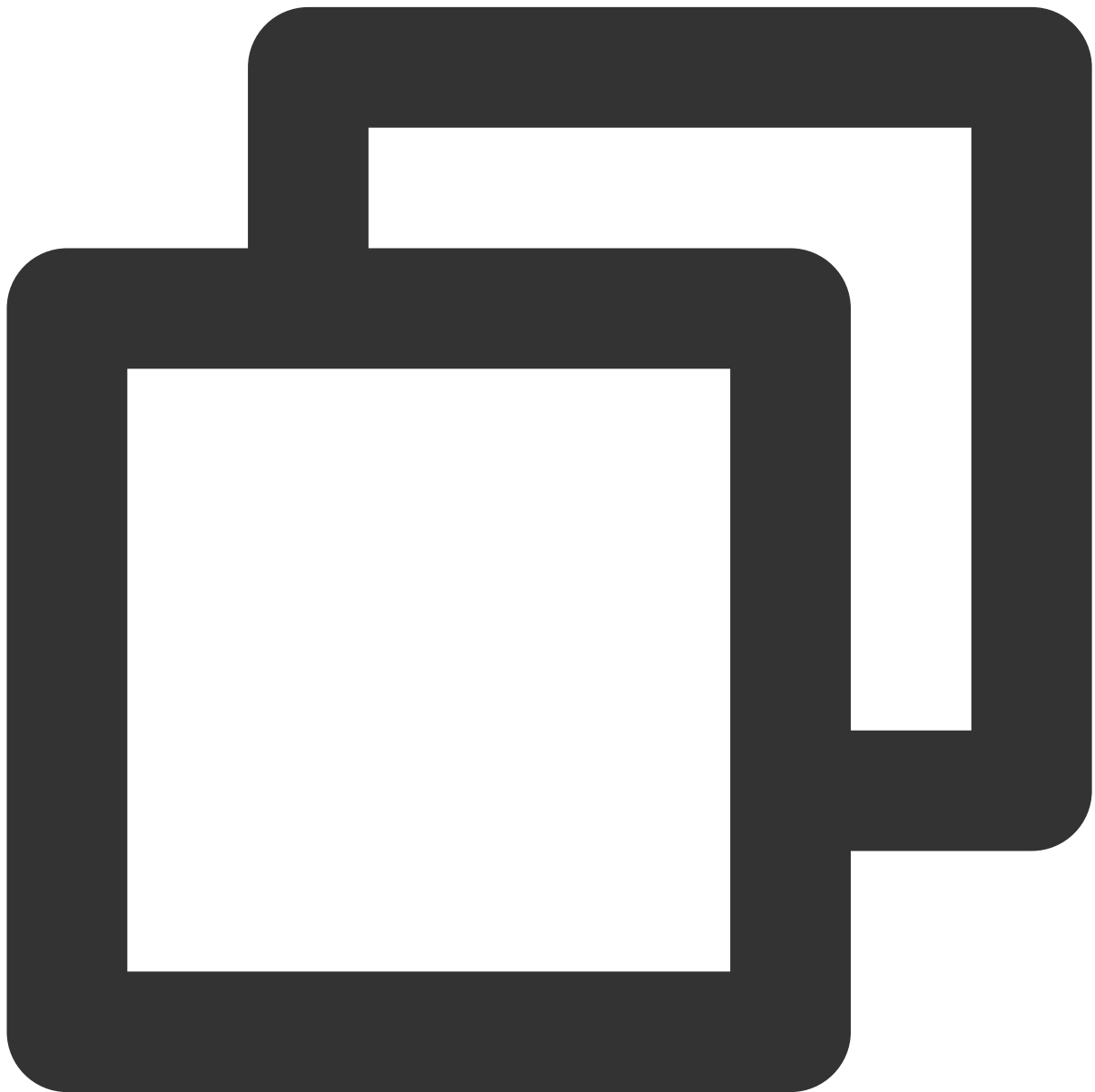


```
import setuptools

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=[
```

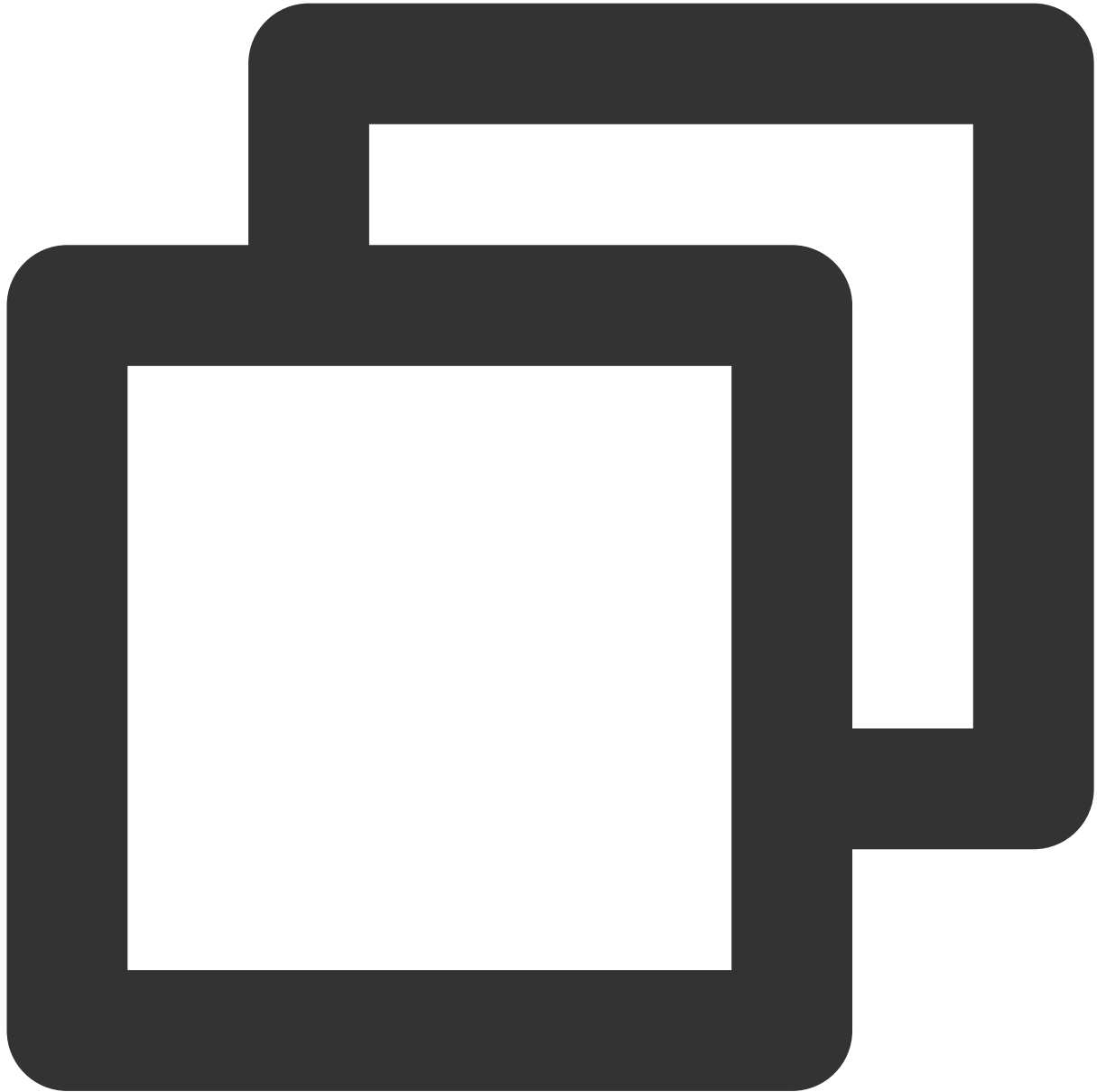
```
    "Programming Language :: Python :: 3",  
    "License :: OSI Approved :: MIT License",  
    "Operating System :: OS Independent",  
],  
python_requires='>=3.6',  
)
```

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:

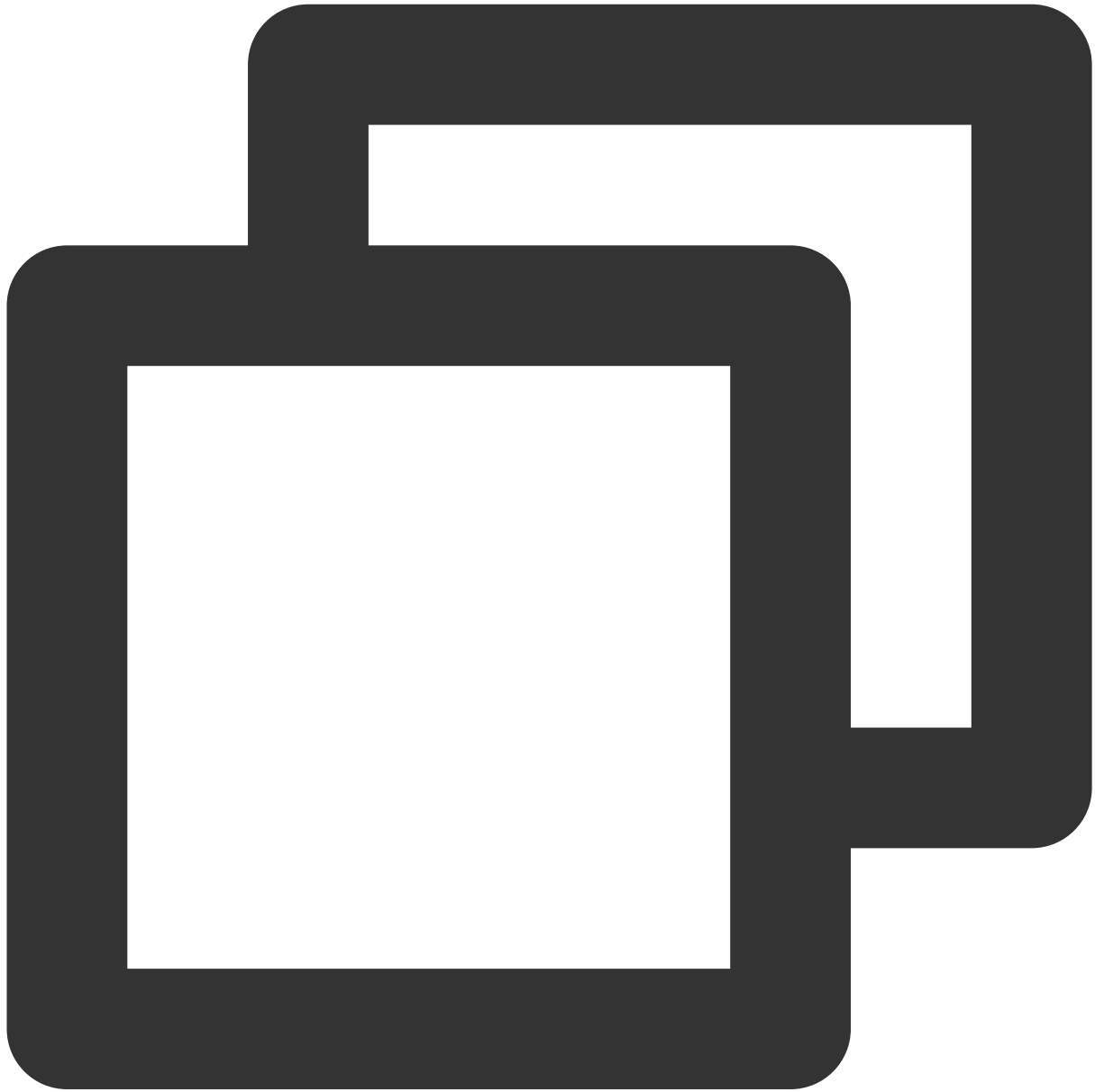


```
└─npm  
  └─example_pkg_YOUR_USERNAME_HERE-0.0.1-py3-none-any.whl  
  └─example_pkg_YOUR_USERNAME_HERE-0.0.1.tar.gz
```

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.

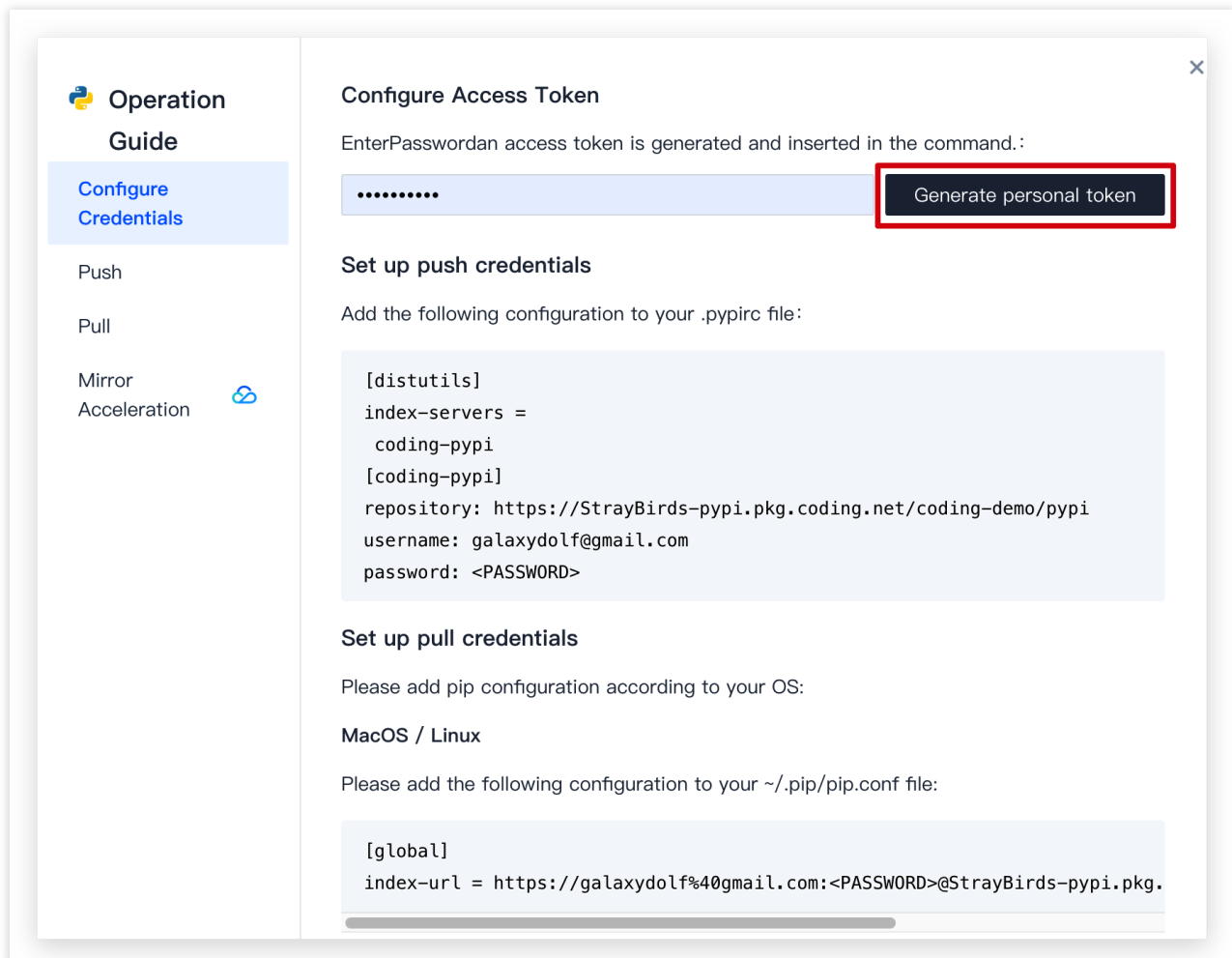
If such files are not found, run the following command to create these files.



```
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**.



The screenshot shows a dialog box titled "Configure Access Token" with a close button (X) in the top right corner. On the left is a sidebar with the heading "Operation Guide" and several menu items: "Configure Credentials" (highlighted in blue), "Push", "Pull", "Mirror Acceleration" (with a cloud icon), and "Acceleration".

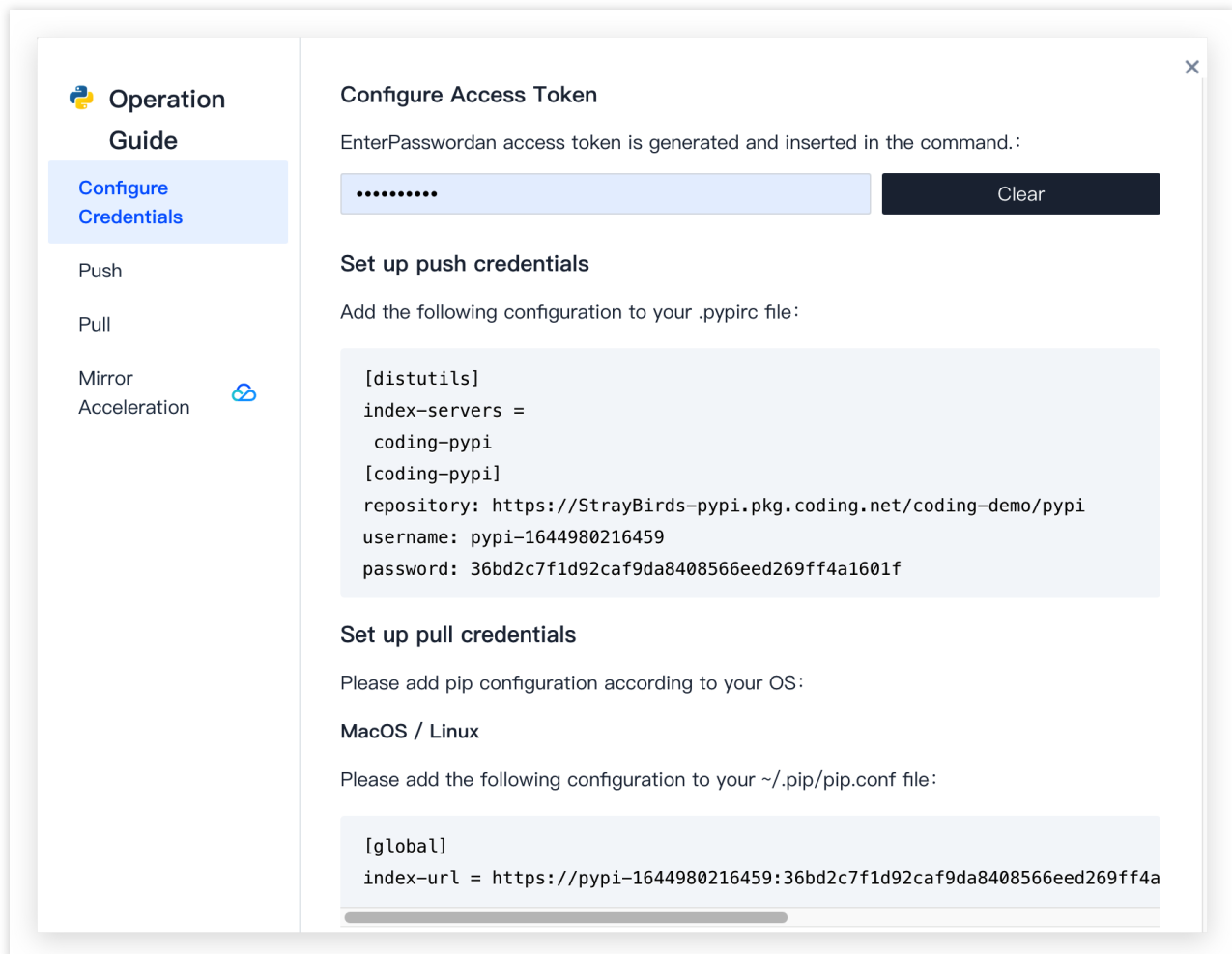
The main content area of the dialog is divided into sections:

- Configure Access Token**: A text prompt says "Enter a password and an access token is generated and inserted in the command.:". Below this is a text input field containing a series of dots. To the right of the input field is a dark button labeled "Generate personal token", which is highlighted with a red rectangular border.
- Set up push credentials**: A text prompt says "Add the following configuration to your .pypirc file:". Below this is a code block containing the following text:

```
[distutils]
index-servers =
  coding-pypi
[coding-pypi]
repository: https://StrayBirds-pypi.pkg.coding.net/coding-demo/pypi
username: galaxydolf@gmail.com
password: <PASSWORD>
```
- Set up pull credentials**: A text prompt says "Please add pip configuration according to your OS:". Below this is a sub-section for "MacOS / Linux" with a text prompt: "Please add the following configuration to your ~/.pip/pip.conf file:". Below this is another code block containing the following text:

```
[global]
index-url = https://galaxydolf%40gmail.com:<PASSWORD>@StrayBirds-pypi.pkg.
```

2. Enter your login password to obtain the configuration.



Operation Guide

- Configure Credentials
- Push
- Pull
- Mirror Acceleration

Configure Access Token

Enter Password an access token is generated and inserted in the command.:

..... Clear

Set up push credentials

Add the following configuration to your .pypirc file:

```
[distutils]
index-servers =
  coding-pypi
[coding-pypi]
repository: https://StrayBirds-pypi.pkg.coding.net/coding-demo/pypi
username: pypi-1644980216459
password: 36bd2c7f1d92caf9da8408566eed269ff4a1601f
```

Set up pull credentials

Please add pip configuration according to your OS:

MacOS / Linux

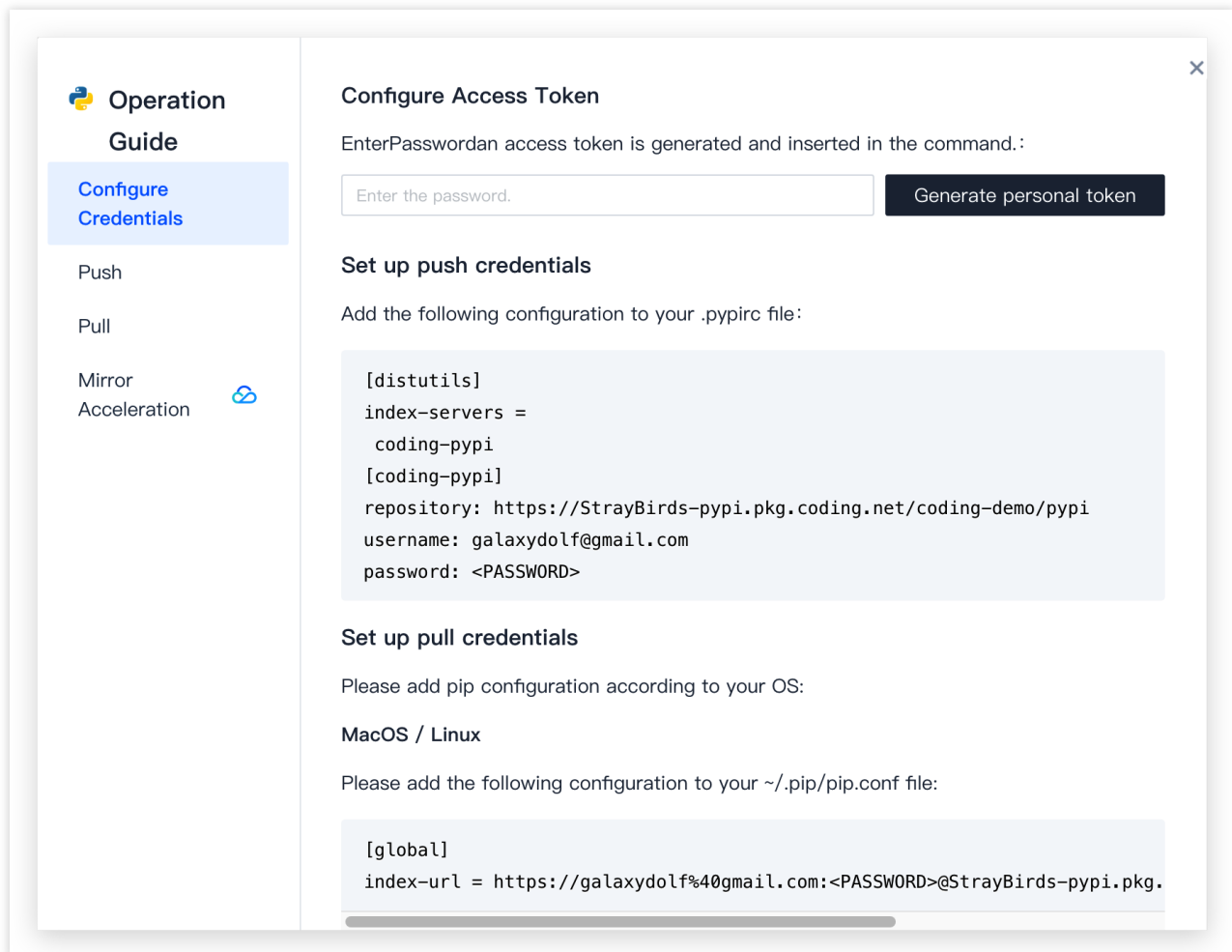
Please add the following configuration to your ~/.pip/pip.conf file:

```
[global]
index-url = https://pypi-1644980216459:36bd2c7f1d92caf9da8408566eed269ff4a
```

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.



Operation Guide

- Configure Credentials
- Push
- Pull
- Mirror Acceleration

Configure Access Token

Enter a password and an access token is generated and inserted in the command.:

Enter the password. Generate personal token

Set up push credentials

Add the following configuration to your .pypirc file:

```
[distutils]
index-servers =
  coding-pypi
[coding-pypi]
repository: https://StrayBirds-pypi.pkg.coding.net/coding-demo/pypi
username: galaxydolf@gmail.com
password: <PASSWORD>
```

Set up pull credentials

Please add pip configuration according to your OS:

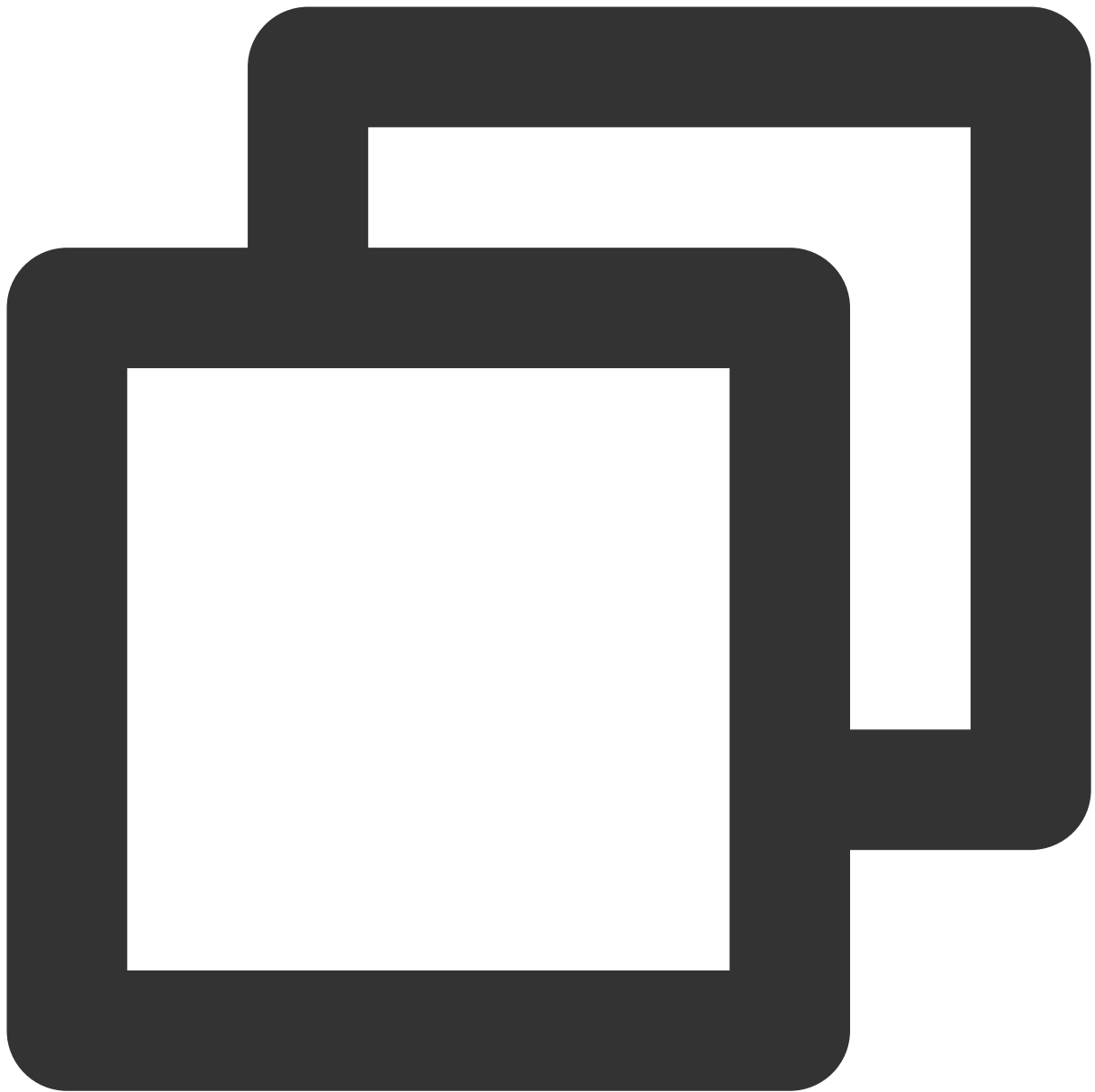
MacOS / Linux

Please add the following configuration to your ~/.pip/pip.conf file:

```
[global]
index-url = https://galaxydolf%40gmail.com:<PASSWORD>@StrayBirds-pypi.pkg.
```

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.



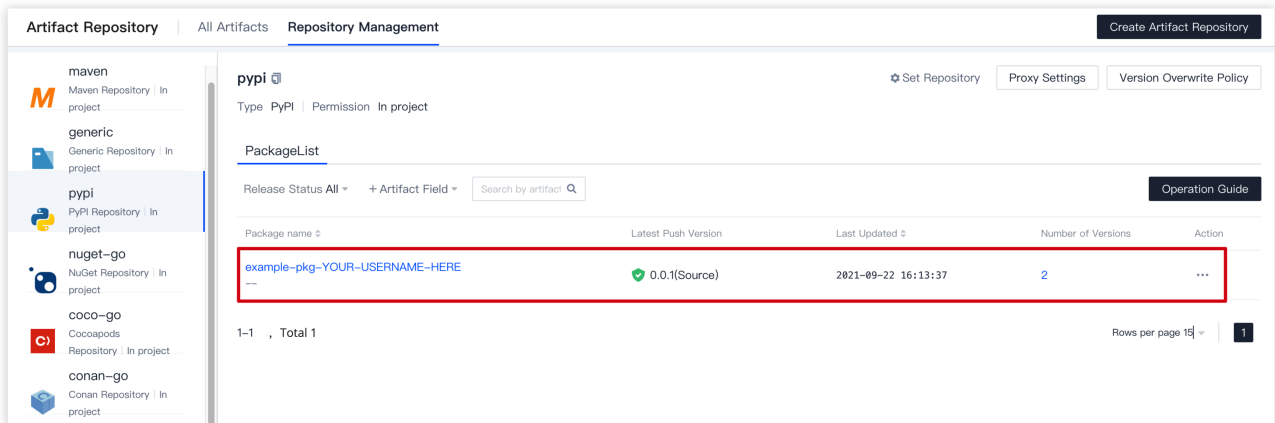
```
twine upload -r coding-pypi dist/*
```

```

/Volumes/CODING/pypi/packaging_tutorial twine upload -r coding-pypi dist/*
Uploading distributions to https://codingcorp-pypi.pkg.coding.net/coding-help-generat
or/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.

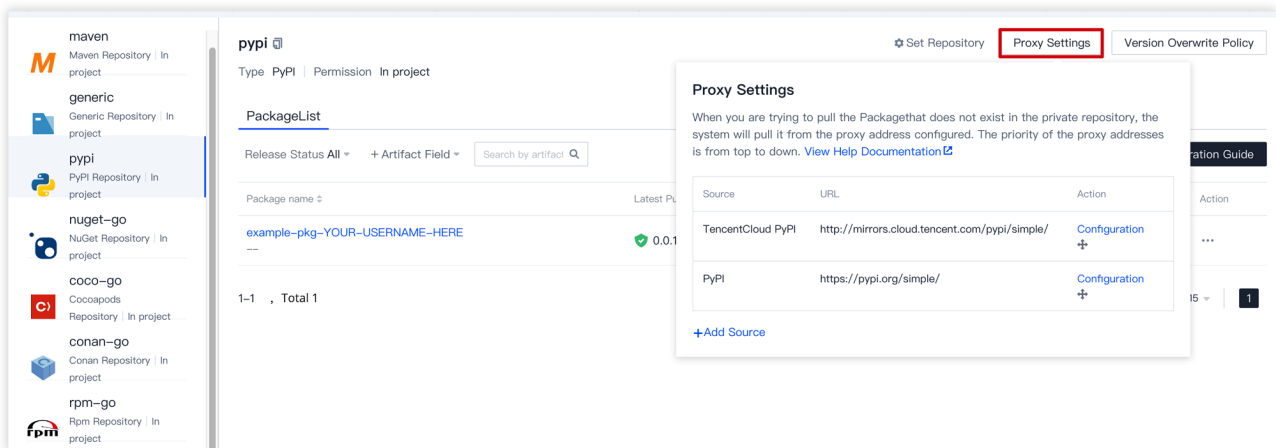


```
pip install <Artifact Name>
```

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 init
```

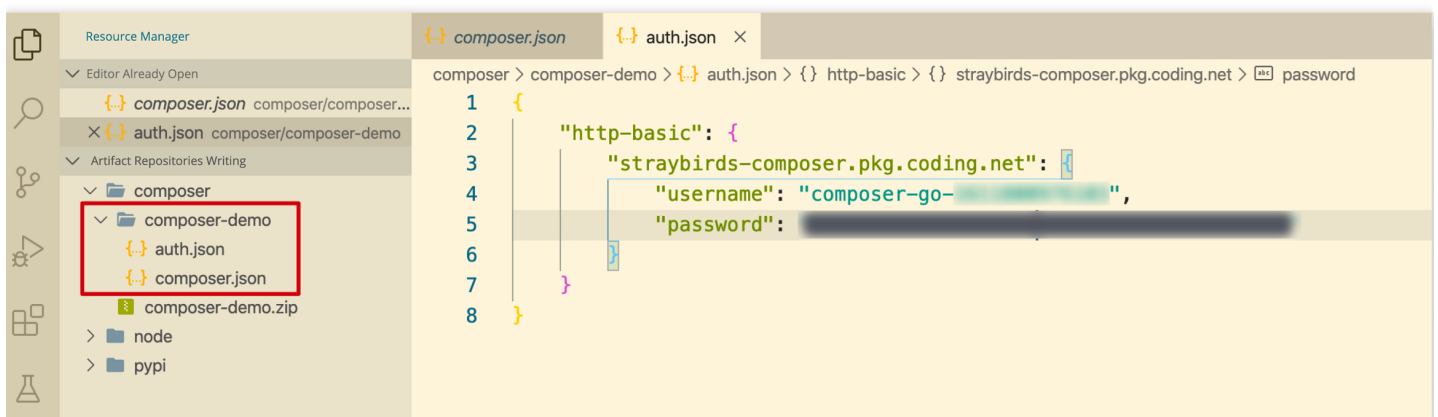
After initialization, a `composer.json` configuration file will be added to the directory.

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`.

 **Operation Guide**

Configure Credentials

Push

Pull

[Help Center](#)

Configure Access Token

Enter a password and an access token is generated and inserted in the command.:

Generate personal token

Configure push credentials

Please add the following configuration to your `~/.netrc` file:

```

machine StrayBirds-composer.pkg.coding.net
login galaxydolf@gmail.com
password <PASSWORD>
```

Replace Text:

- PASSWORD: your login password

Configure Pull Credentials

Please go to the Composer package file directory and add the following configuration to the `auth.json` file:

```

{
  "http-basic": {
    "StrayBirds-composer.pkg.coding.net": {
      "username": "galaxydolf@gmail.com",
      "password": "<PASSWORD>"
    }
  }
}
```

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

```


/Volumes/CODING/.../composer $ zip -r composer-demo.zip . -x "./vendor/*"
updating: composer-demo/ (stored 0%)
updating: composer-demo/auth.json (deflated 31%)
updating: composer-demo/composer.json (deflated 38%)
/Volumes/CODING/.../composer $ curl -T composer-demo.zip -u @qq.com "https://straybirds-composer.pkg.coding.net/coding-demo/composer-go?version=0.0.2"
Enter host password for user '@qq.com':
success
```


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

The screenshot displays the CODING Artifact Repository interface. On the left, a sidebar lists various repository types: maven, generic, pypi, nuget-go, coco-go, conan-go, rpm-go, composer-go (selected), helm-go, and npm-go. The main content area is titled 'composer-go' and includes options for 'Set Repository', 'Proxy Settings', and 'Version Overwrite Policy'. Below this, it shows 'Type Composer | Permission In project' and a 'PackageList' section. The 'PackageList' section features a search bar and a table with columns for 'Package name', 'Latest Push Version', 'Last Updated', 'Number of Versions', and 'Action'. A single entry is visible: 'adolf/composer-demo' with version '0.0.2', updated on '2021-01-28 10:58:45', and 2 versions. At the bottom right of the table, it indicates '1-1 , Total 1' and 'Rows per page 15'.

Pull an Artifact

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

**Operation Guide**

- Configure Credentials
- Push
- Pull**

Pull

Enter the following pull information to generate the pull command:

Artifact Name:

Artifact Version:

1. In the Composer package file directory, execute the following command to set the repository address:

```
composer config repos.composer-go composer https://StrayBirds-composer.p
```
2. In the Composer package file directory, execute the following command to pull:

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

If you want to use this product repository to represent the official Composer product source, please execute the following command in the Composer package file directory before pulling it:

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

[Help Center](#)

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/CODING.../testing composer require adolf/composer-demo:0.0.2 Authentication required (straybirds-composer.pkg.coding.net):
Username: @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: @qq.com
Password:
Do you want to store credentials for straybirds-composer.pkg.coding.net in /Users/adolff/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.


The screenshot shows the 'Repository Management' page for 'composer-go'. On the left, there is a sidebar with various repository types like maven, generic, pypi, nuget-go, coco-go, conan-go, rpm-go, composer-go, and helm-go. The main area shows the 'PackageList' for 'composer-go' with a search bar and a table of packages. The table has columns for 'Package name' and 'Latest Push Version'. One package is listed: 'adolff/composer-demo' with version '0.0.2'. A 'Proxy Settings' dialog box is overlaid on the right, showing a table with columns 'Source', 'URL', and 'Action'. The table contains one entry: 'Aliyun Composer' with URL 'https://mirrors.aliyun.com/composer/' and a 'Configuration' link. A '+Add Source' button is at the bottom of the dialog.

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
```

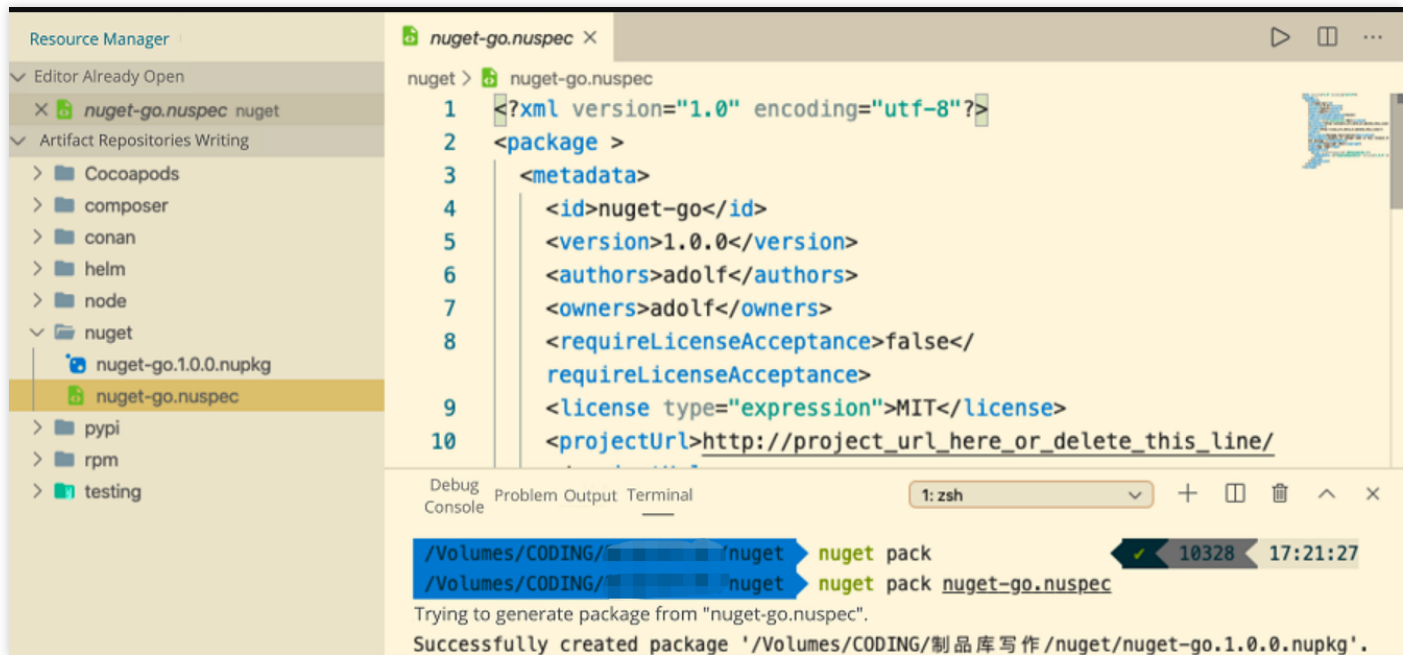
2. Create a `.nuspec` package.

```
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.

Newtonsoft.Json 13.0.1

Json.NET is a popular high-performance JSON framework for .NET

Requires NuGet 2.12 or higher.

Package Manager .NET CLI PackageReference Paket CLI Script & Interactive Cake

```
PM> Install-Package Newtonsoft.Json -Version 13.0.1
```

> Dependencies

> Used By

∨ Version History

Version	Downloads	Last updated
13.0.1	10,247,917	4 months ago
12.0.3	143,709,834	9/11/2019
12.0.2	127,120,363	22/4/2019
12.0.1	70,142,630	27/11/2018
11.0.2	125,789,560	24/3/2018

+ Show more

Info

- last updated 4 months ago
- Project Site
- Source repository
- License: MIT
- Contact owners
- Report
- Download package (1.97 MB)**
- Download symbols (733.23 KB)
- Open in FuGet Package Explorer

Statistics

- 1,151,898,081 total downloads
- 10,247,917 downloads of current version
- 299,505 downloads per day (avg)
- View full stats

Owners

jamesnk

Pull an artifact via the command line:

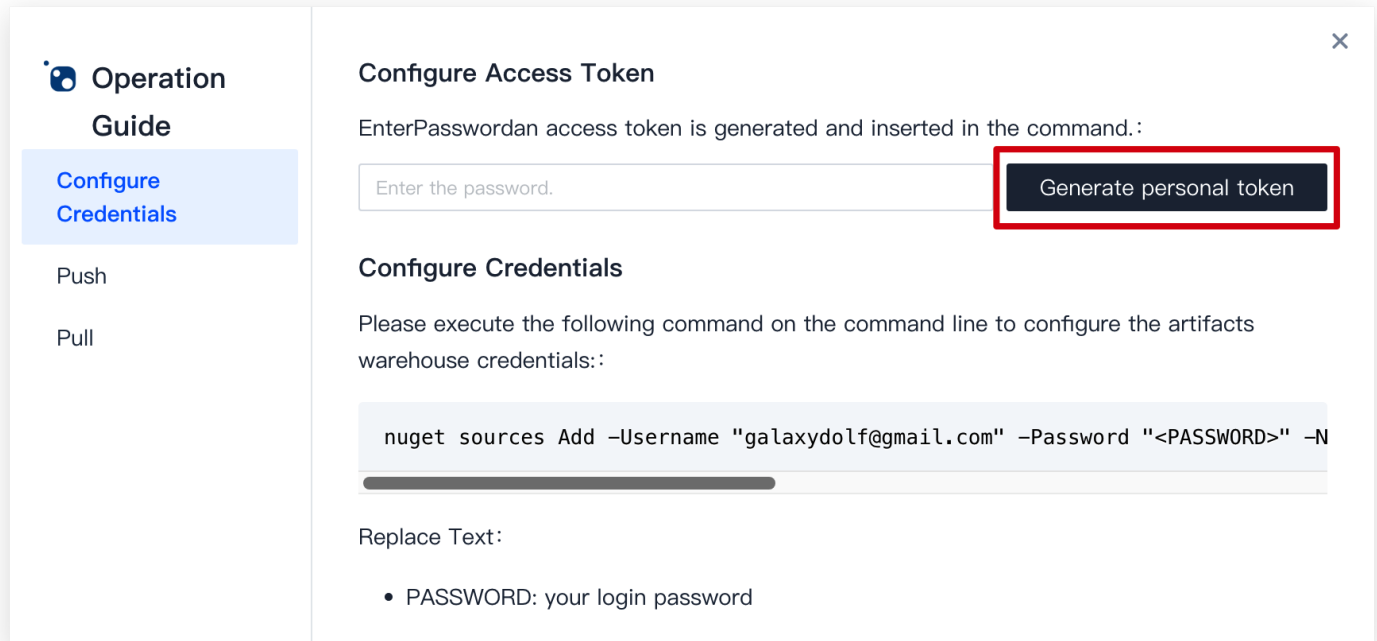
```
nuget install [Artifact Name] -OutputDirectory packages
```

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**.



Operation Guide

- Configure Credentials
- Push
- Pull

Configure Access Token

EnterPasswordan access token is generated and inserted in the command.:

Enter the password. **Generate personal token**

Configure Credentials

Please execute the following command on the command line to configure the artifacts warehouse credentials::

```
nuget sources Add -Username "galaxydolf@gmail.com" -Password "<PASSWORD>" -N
```

Replace Text:

- PASSWORD: your login password

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 artifact 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 Version] [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.

The screenshot shows the 'Artifact Repository' management interface. On the left, a sidebar lists various repository types like 'maven', 'generic', 'pypi', and 'nuget-go'. The main area displays the 'nuget-go' repository details, including a 'PackageList' table with columns for 'Package name' and 'Latest Push Version'. A 'Proxy Settings' dialog box is overlaid on the right, containing a table with columns for 'Source', 'URL', and 'Action'. The table lists 'nuget.org' with the URL 'https://api.nuget.org/v3/index.json'. A '+Add Source' button is highlighted with a red box at the bottom of the table.

Run the following command to pull an artifact:

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

The screenshot shows a dialog box titled 'Operation Guide' with a sidebar on the left containing options like 'Configure Credentials', 'Push', and 'Pull'. The 'Pull' option is selected. The main content area is titled 'Pull' and contains the instruction: 'Enter the following pull information to generate the pull command:'. Below this are two input fields: 'Artifact Name:' and 'Artifact Version:'. At the bottom, a code block displays the command: `nuget install -Source "nuget-go" -Version <VERSION> <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.

The screenshot shows the details page for the Serilog artifact. The page is titled "Serilog" and includes a back arrow, a settings icon, and a "Settings" link. The push time is 2021-07-19 19:50:47. The version ID is 2.10.0, and the repository is nuget-go. The page has tabs for Overview, File List, Field, and Version List. The Overview tab is active, showing a dependency list and a description. The dependency list includes .NETFramework4.5, .NETStandard1.0, .NETStandard1.3, .NETStandard2.0, .NETFramework4.6, .NETStandard1.3, and .NETStandard2.0. The description is "Simple .NET logging with fully-structured events". The push info shows it was pushed by Steven on 2021-07-19 19:50:47. The other section lists the author as Serilog Contributors, the protocol as Apache-2.0, and tags as serilog, logging, semantic, and structured. The size is 480.44 KB and the source is Proxy https://api.nuget.org.

Serilog Settings

Push Time 2021-07-19 19:50:47

VersionID 2.10.0

Repository nuget-go

Overview File List Field Version List Operation

Dependency

.NETFramework4.5 Version No.:---	.NETFramework4.6 Version No.:---
.NETStandard1.0 : NETStandard.Library Version No.:>= 1.6.1	.NETStandard1.3 : NETStandard.Library Version No.:>= 1.6.1
.NETStandard1.3 : System.Collections.NonGeneric Version No.:>= 4.3.0	.NETStandard2.0 Version No.:---
.NETStandard2.1 Version No.:---	

Description

Simple .NET logging with fully-structured events

Push Info

Pushed by Steven

Push Time 2021-07-19 19:50:47

Other

Author Serilog Contributors

Protocol Apache-2.0

Tag serilog logging semantic structured

Size 480.44 KB

Source Proxy https://api.nuget.org

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:

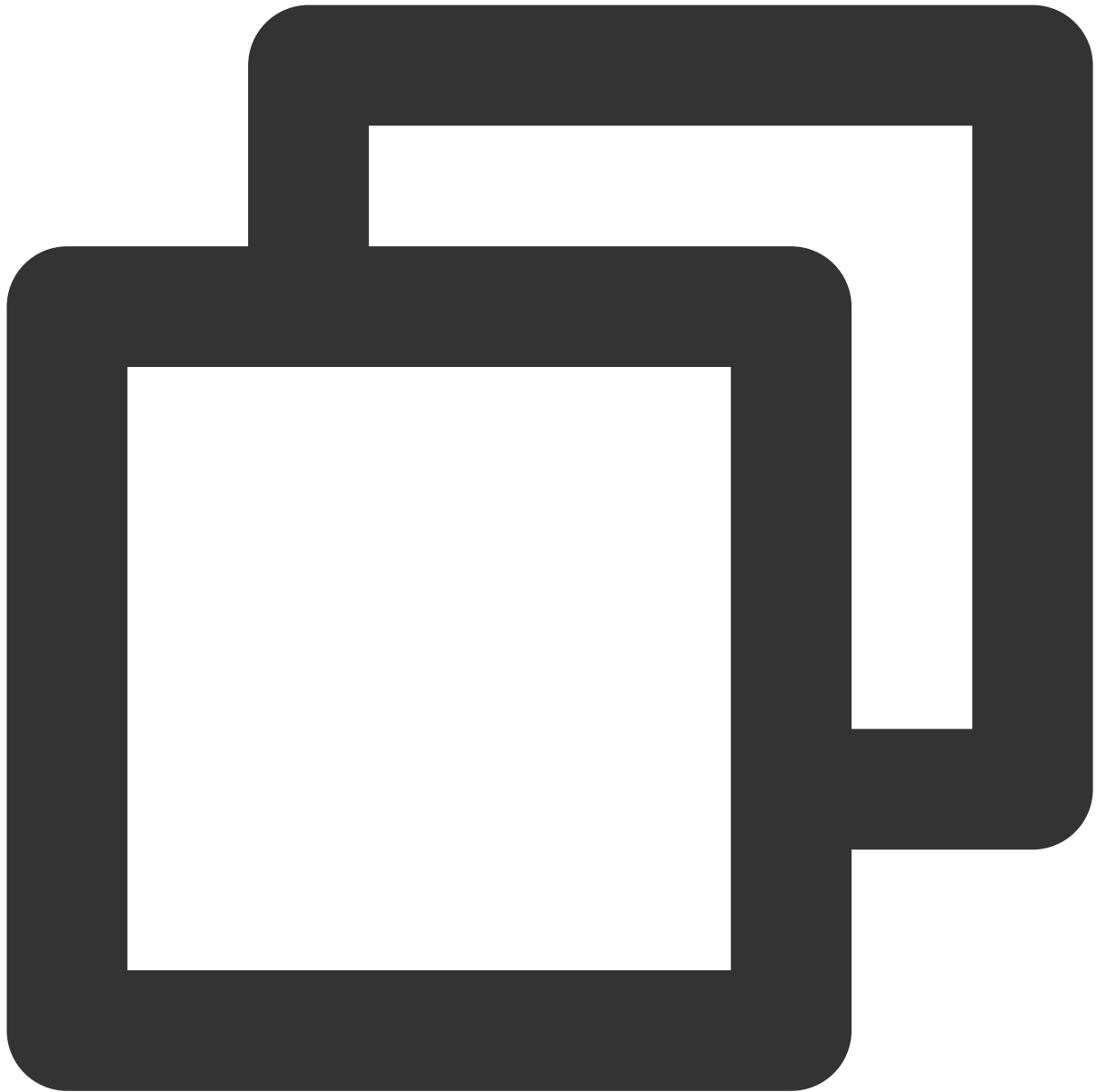
Prepare a Linux environment.

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:

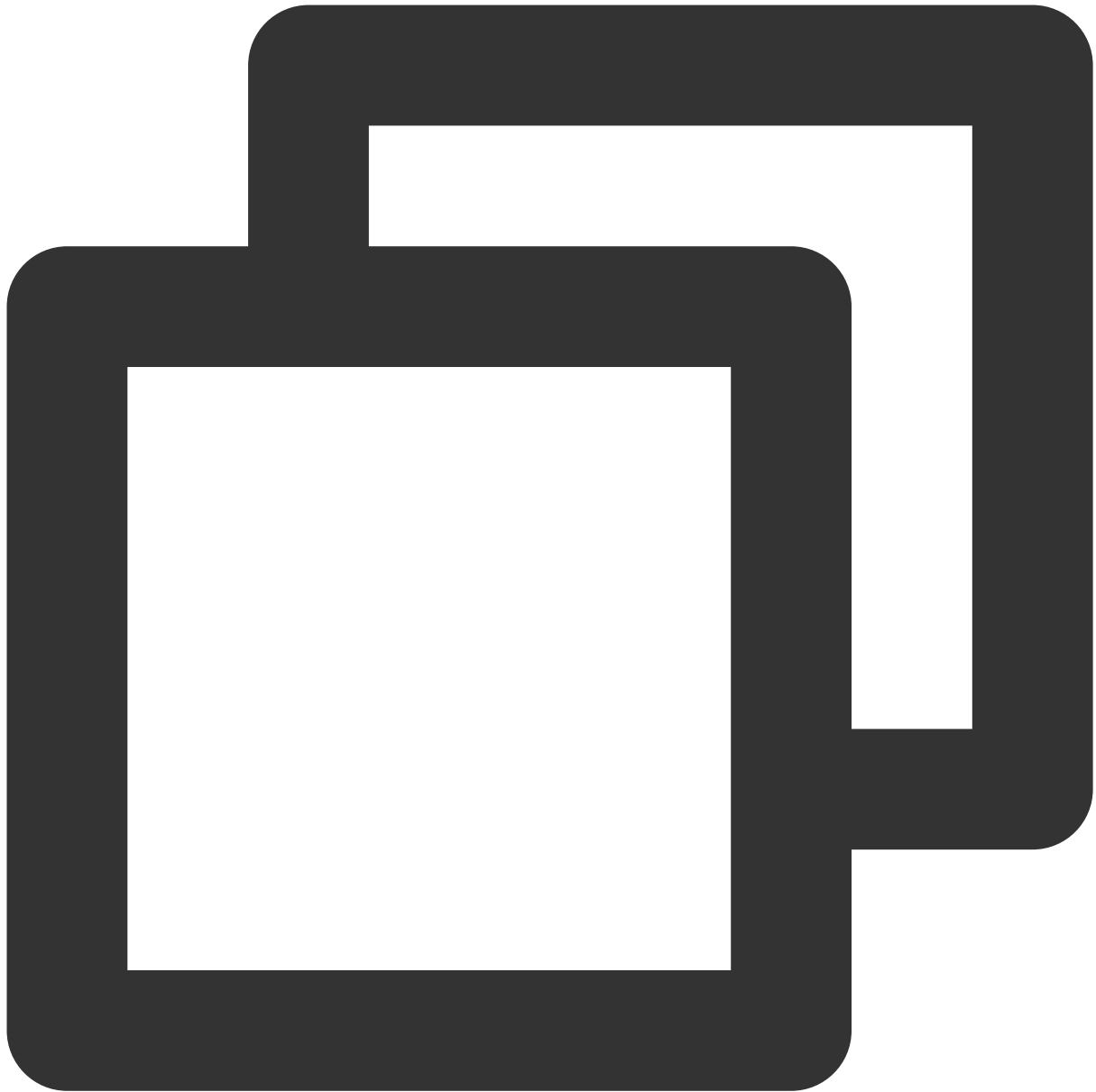


```
docker run -it --name centos centos:8 /bin/bash
```

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**.

 **Operation****Guide****Configure
Credentials**

Push

Pull

Configure Access Token

Enter Password and an access token is generated and inserted into the configuration.

Set CredentialsPlease add the following configuration to your `/etc/yum.repos.d/rpm-go.repo` file.

```
[rpm]
name=rpm-go
baseurl=https://StrayBirds-rpm.pkg.coding.net/coding/rpm-go
enabled=1
username=galaxydolf@gmail.com
password=<PASSWORD>
repo_gpgcheck=0
gpgcheck=0
```

Replace Text:

- **PASSWORD:** your login password

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.

Debug Console Problem Output Terminal

```
[rpm-go]
```

```
name=rpm-go
```

```
baseurl=https://straybirds-rpm.pkg.coding.net/c
```

```
enabled=1
```

```
username=rpm-go-
```

```
password=
```

```
repo_gpgcheck=0
```

```
gpgcheck=0
```

```
~
```

```
~
```

```
~
```

```
~
```

```
~
```

```
~
```

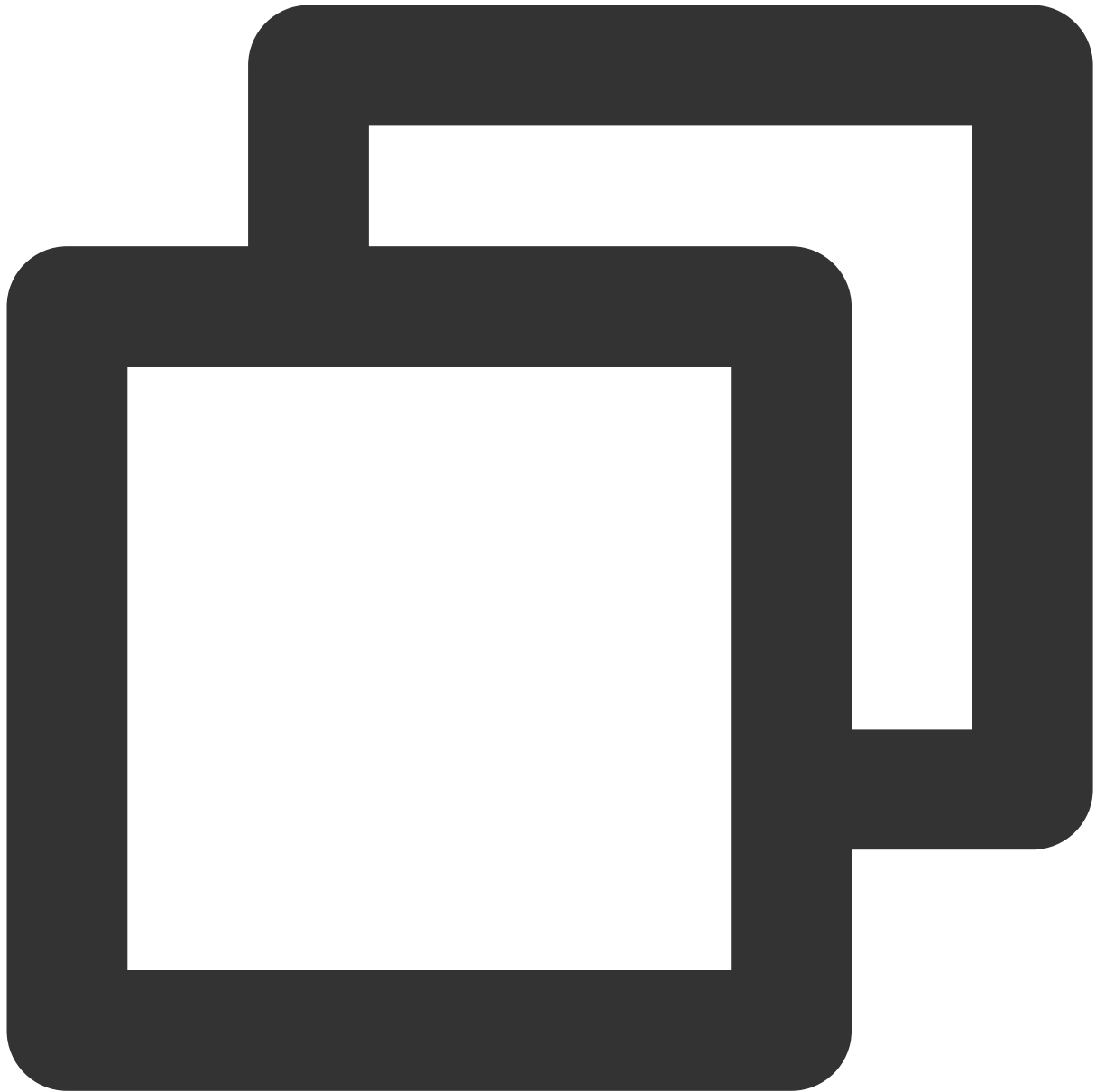
```
~
```

```
~
```

```
"/etc/yum-repos.d/rpm-go.repo" 81 - 2040
```

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.

Artifact Repository | All Artifacts | **Repository Management**

- 
maven
 Maven Repository | In project
- 
generic
 Generic Repository | In project
- 
pypi
 PyPI Repository | In project
- 
nuget-go
 NuGet Repository | In project
- 
coco-go
 CocoaPods Repository | In project
- 
conan-go
 Conan Repository | In project
- 
rpm-go
 Rpm Repository | In project

rpm-go 

Type Rpm | Permission In project

PackageList

Release Status All ▾ + Artifact Field ▾

Package name ↕	Latest Push Version	Last Updated ↕
hello	2.10-5.fc34.aarch64	2021-02-09 15:28:52

1-1 , Total 1

Pull an Artifact

Run the command in the guide to pull an artifact.

Operation Guide

- Configure Credentials
- Push
- Pull**

Pull

Enter the following pull information to generate the pull co

Artifact Name:

Artifact Version:

Pull using yum command

```
yum install --repo rpm-go <PACKAGE>
```

Pull using the rpm command

```
rpm -i https://galaxydolf%40gmail.com:<PASSWORD>
```








Replace Text:


- PASSWORD: your login passowrd

Configure a Proxy

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

Artifact Repository
All Artifacts
Repository Management

-  **maven**
Maven Repository | In project
-  **generic**
Generic Repository | In project
-  **pypi**
PyPI Repository | In project
-  **nuget-go**
NuGet Repository | In project
-  **coco-go**
Cocoapods Repository | In project
-  **conan-go**
Conan Repository | In project
-  **rpm-go**
Rpm Repository | In project

rpm-go 

Type **Rpm** | Permission **In project**

PackageList

Release Status **All** ▾ + Artifact Field ▾

Package name	Latest Push Version
hello	2.10-5.fc34.aarch64

1-1 , Total 1

Proxy Settings

When you are trying to pull the Pack system will pull it from the proxy address from top to down. [View Help Doc](#)

Source	URL
No data available.	

[+Add Source](#)
Sync Index

✓ Last synced on 2021-02-09 15:2

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**.

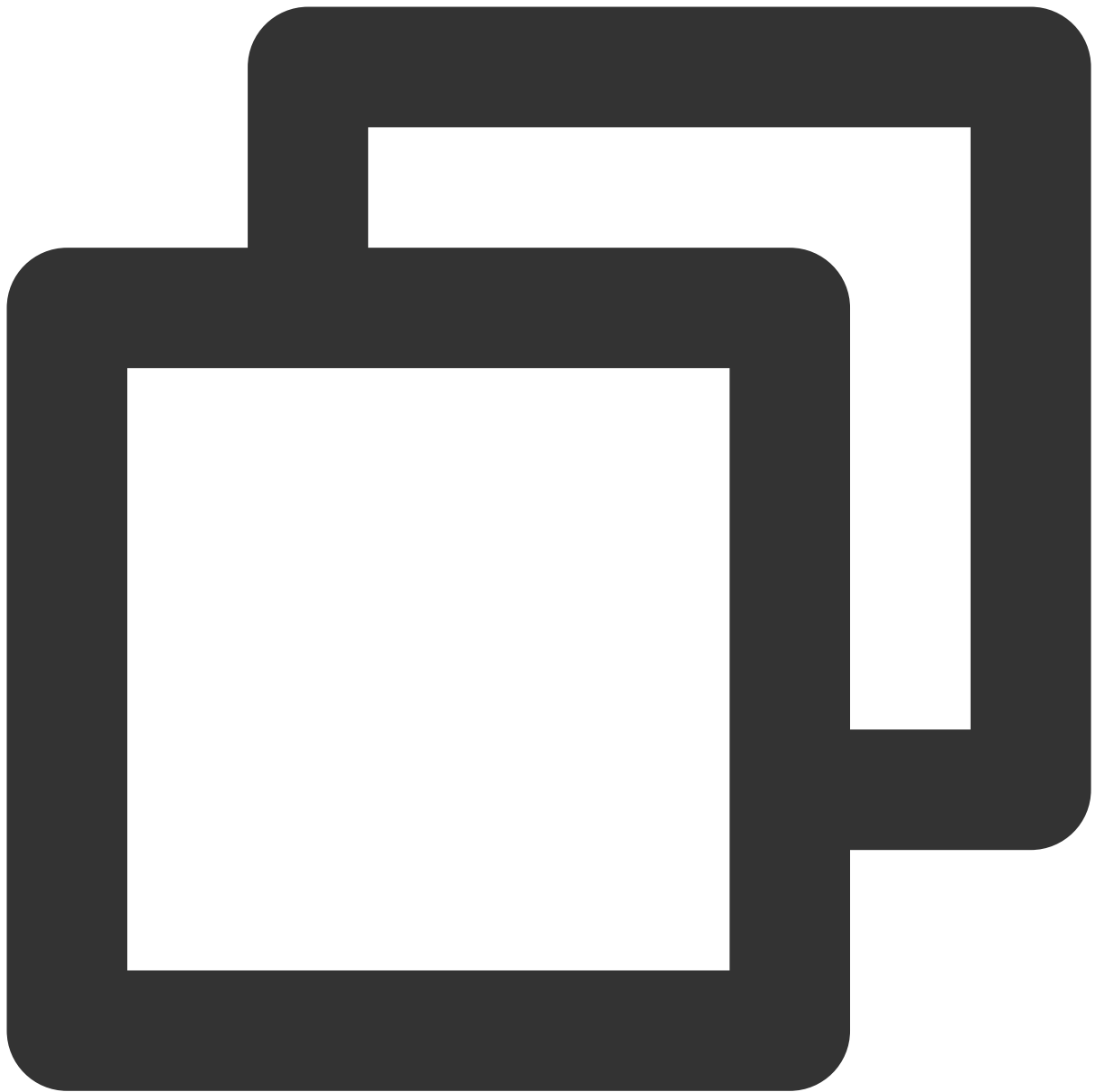
Preparations

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
```

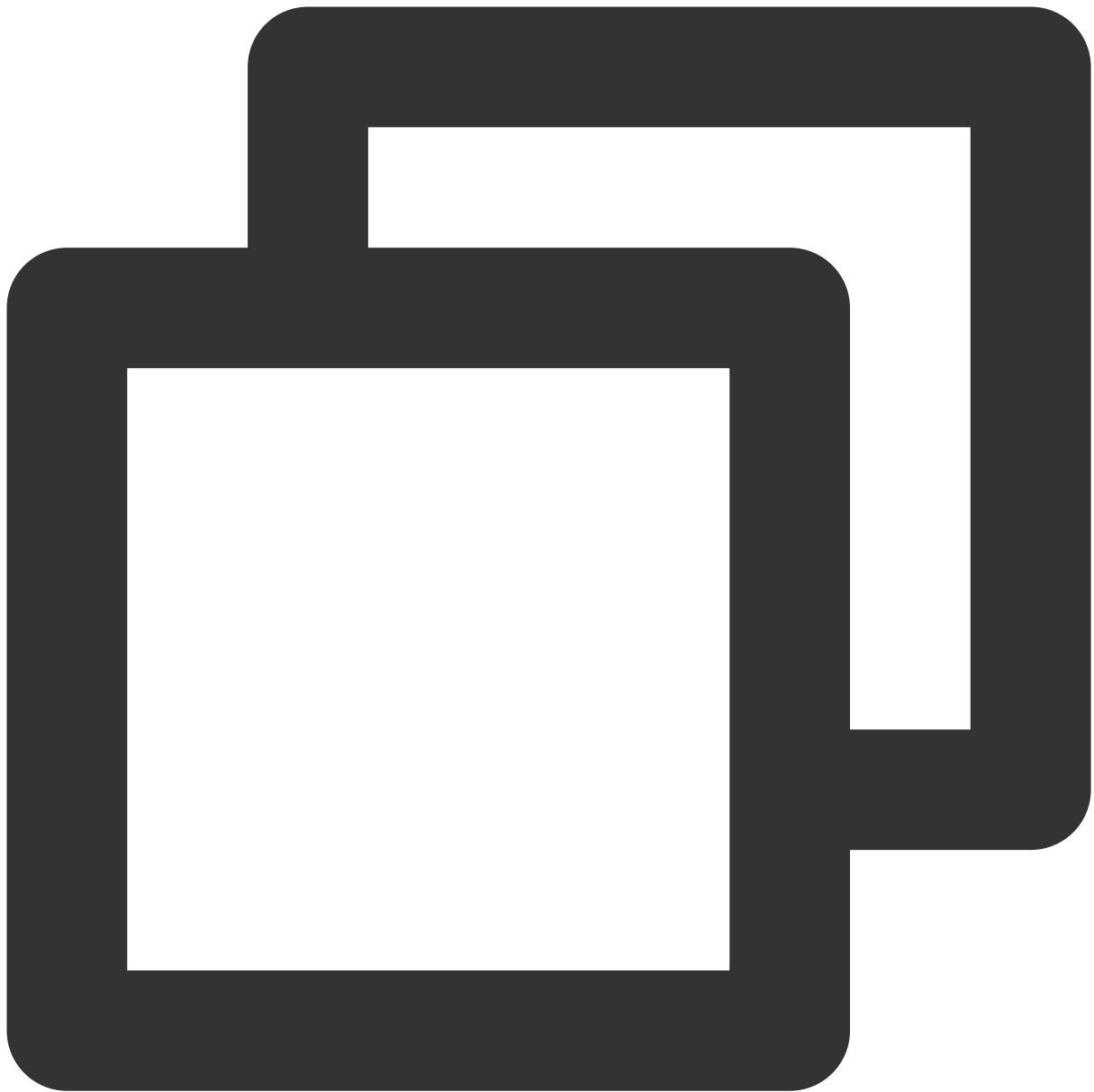
Create a Conan Package

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` :

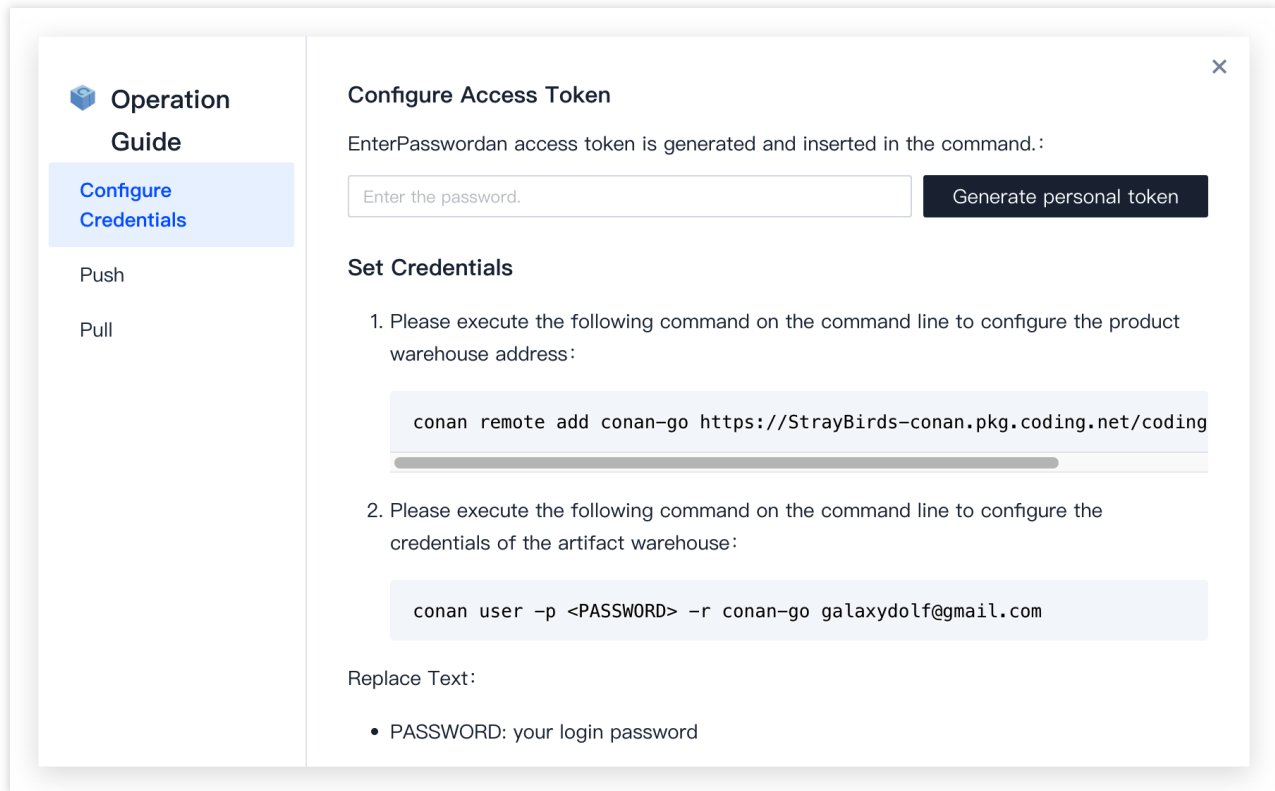


```
$ pip3 install cmake  
# or  
$ brew install cmake
```

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**.



The screenshot shows a dialog box titled "Configure Access Token" with a close button (X) in the top right corner. On the left, there is a sidebar with the following items: "Operation Guide" (with a sub-item "Configure Credentials" highlighted in blue), "Push", and "Pull".

The main content area of the dialog box is titled "Configure Access Token" and contains the following instructions:

Enter a password and an access token is generated and inserted in the command.:

Enter the password.

Set Credentials

1. Please execute the following command on the command line to configure the product warehouse address:

```
conan remote add conan-go https://StrayBirds-conan.pkg.coding.net/coding
```

2. Please execute the following command on the command line to configure the credentials of the artifact warehouse:

```
conan user -p <PASSWORD> -r conan-go galaxydolf@gmail.com
```

Replace Text:

- PASSWORD: your login password

Run the commands as prompted.

The screenshot shows a dialog box titled "Configure Access Token" with a close button (X) in the top right corner. On the left side, there is a sidebar menu under "Operation Guide" with options: "Configure Credentials" (highlighted in blue), "Push", and "Pull".

The main content area of the dialog box contains the following sections:

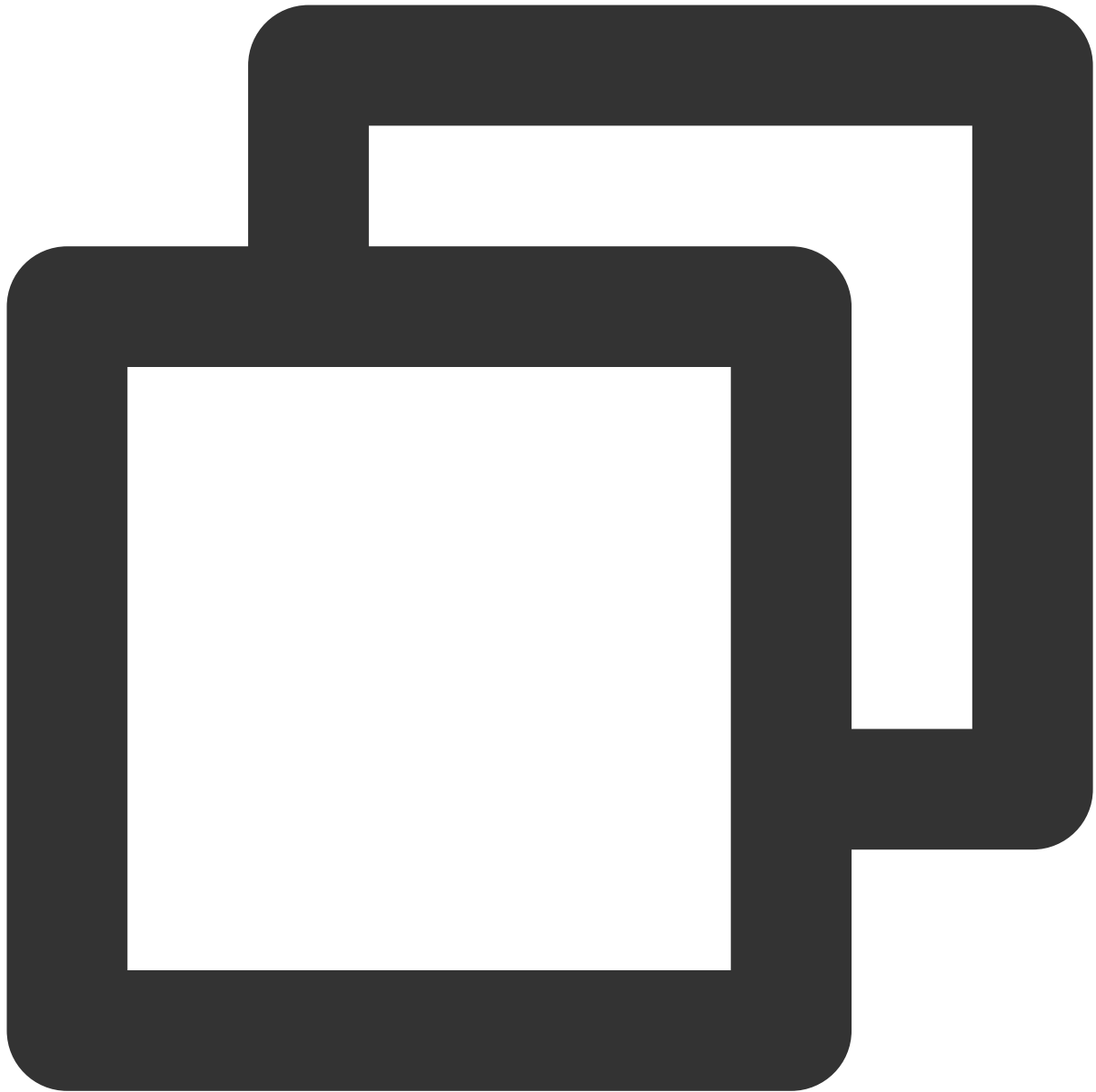
- Configure Access Token**: A text prompt "Enter Password an access token is generated and inserted in the command.:" followed by a password input field (displayed as dots) and a "Clear" button.
- Set Credentials**: A numbered list of instructions:
 1. Please execute the following command on the command line to configure the product warehouse address:

```
conan remote add conan-go https://StrayBirds-conan.pkg.coding.net/coding
```
 2. Please execute the following command on the command line to configure the artifacts warehouse credentials:

```
conan user -p dca236afda951d6d1980700719c0479bfe10b8a1 -r conan-go conan
```

Push an Artifact

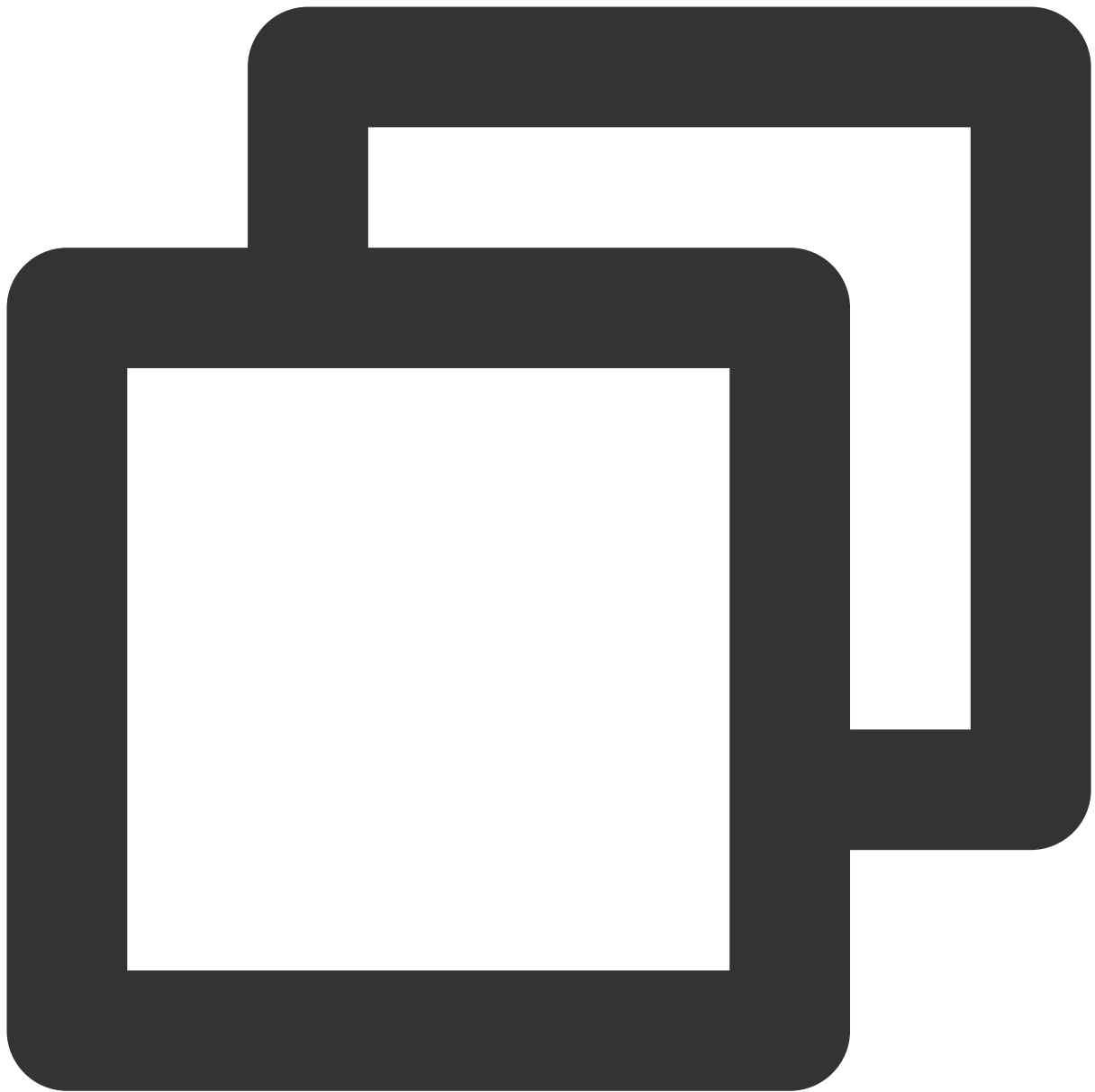
Run the command in the guide. Replace the variables with the name and version of the artifact to be pushed.



```
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.

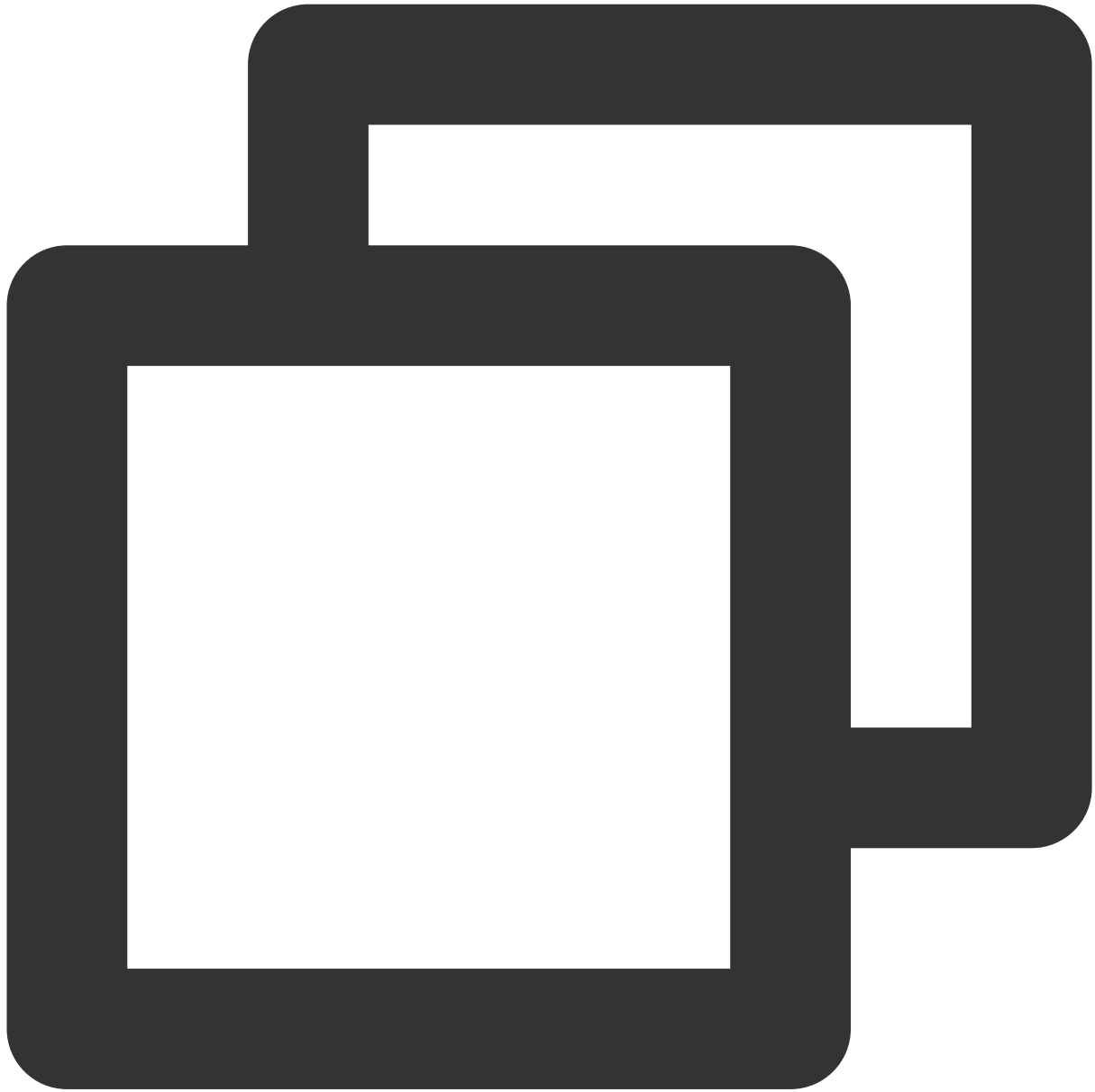


```
$ sudo gem install cocoapods  
-- or  
$ brew install cocoapods
```

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.

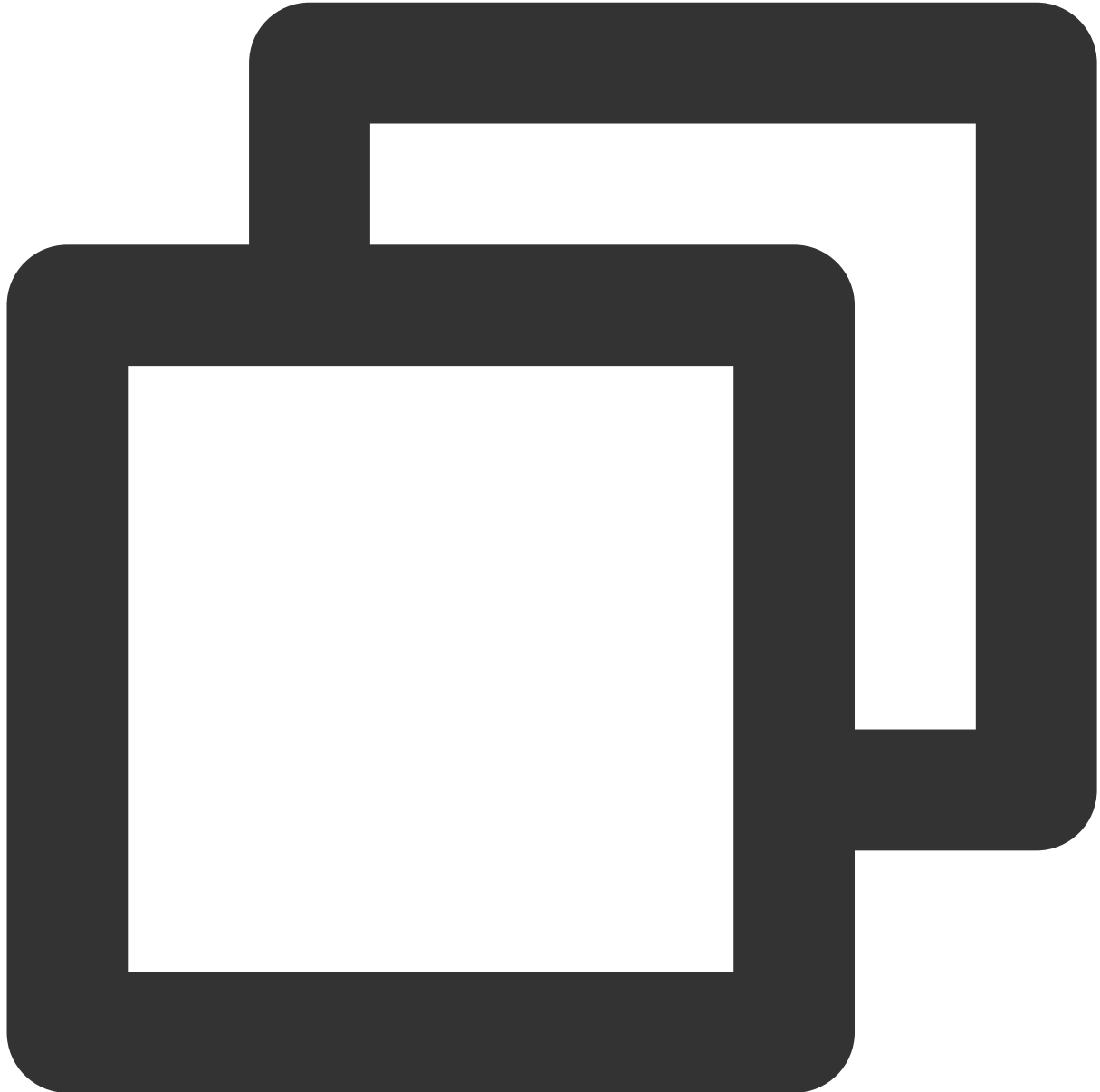


```
pod lib create <custom pod name>
```

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

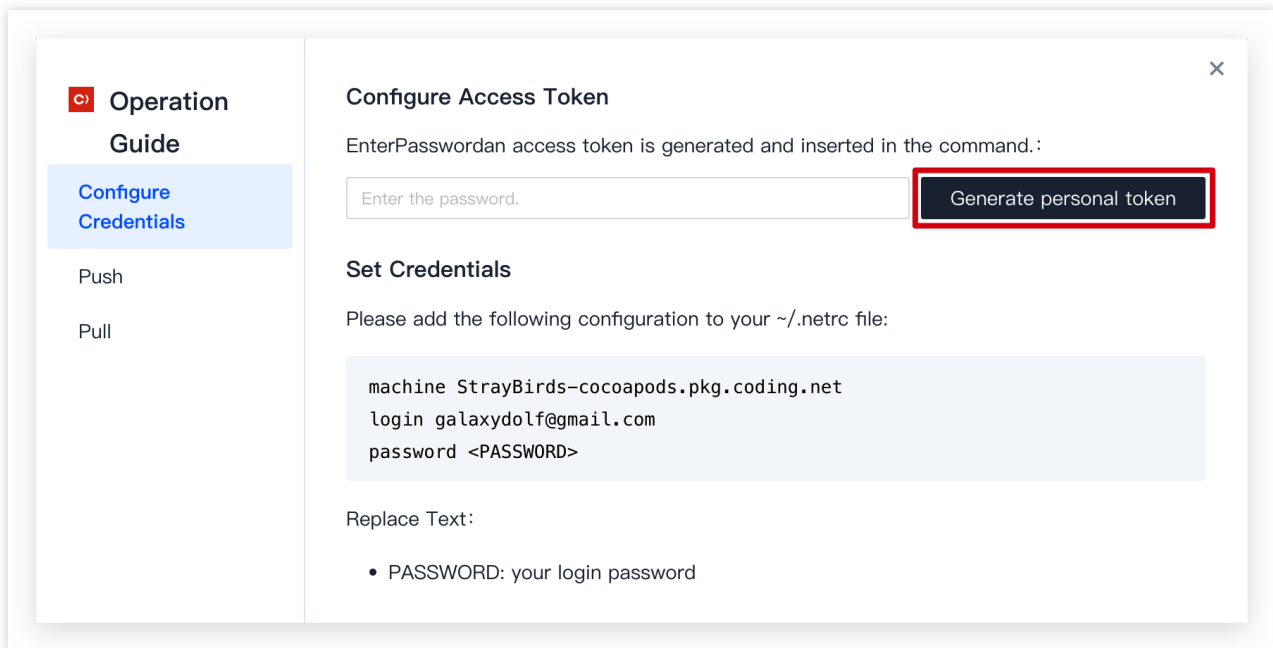
Configure Authentication Information

Install the CODING CocoaPods plugin.

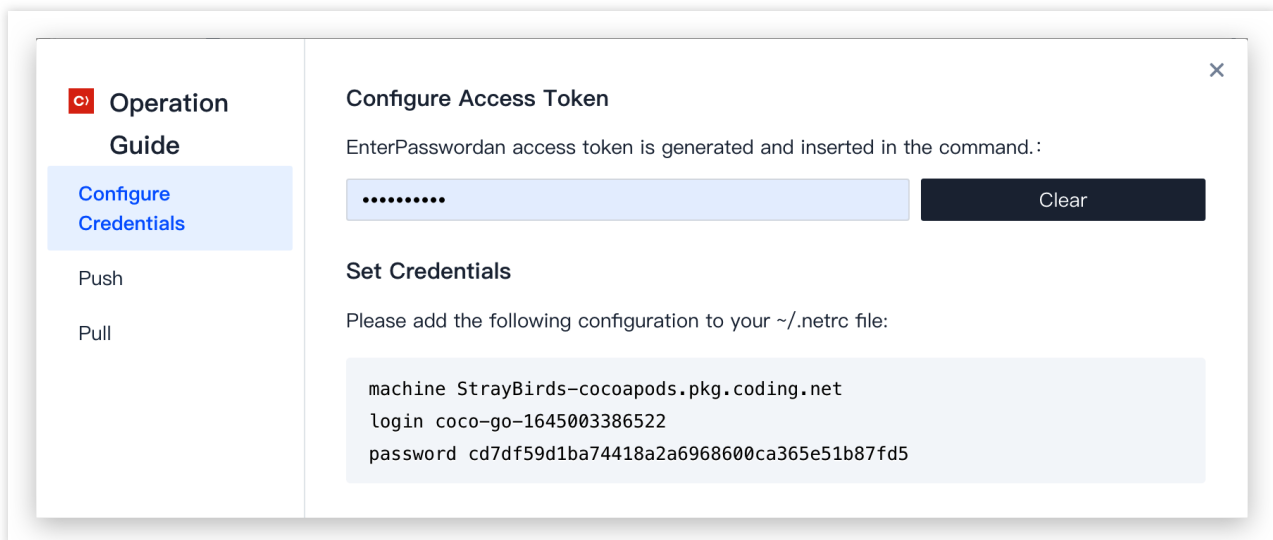


```
sudo gem install cocoapods-coding-ar
```

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.

The screenshot shows a web interface for the 'Push' operation. On the left is a sidebar with a red 'C' logo and the text 'Operation Guide'. Under 'Operation Guide', there are three items: 'Configure Credentials', 'Push' (highlighted in blue), and 'Pull'. The main content area is titled 'Push' and contains the following instructions:

Enter the following push information to generate the push command.

Artifact Name:

1. Install the CODING Cocoapods plugin

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

2. Please execute the following command on the command line to add the current product repository locally:

```
pod repo add-coding-ar coco-go https://StrayBirds-cocoapods.pkg.coding.n
```

3. Run the following command in the command line to push.

```
pod repo push-coding-ar coco-go <PACKAGE>.podspec
```

Pull an Artifact

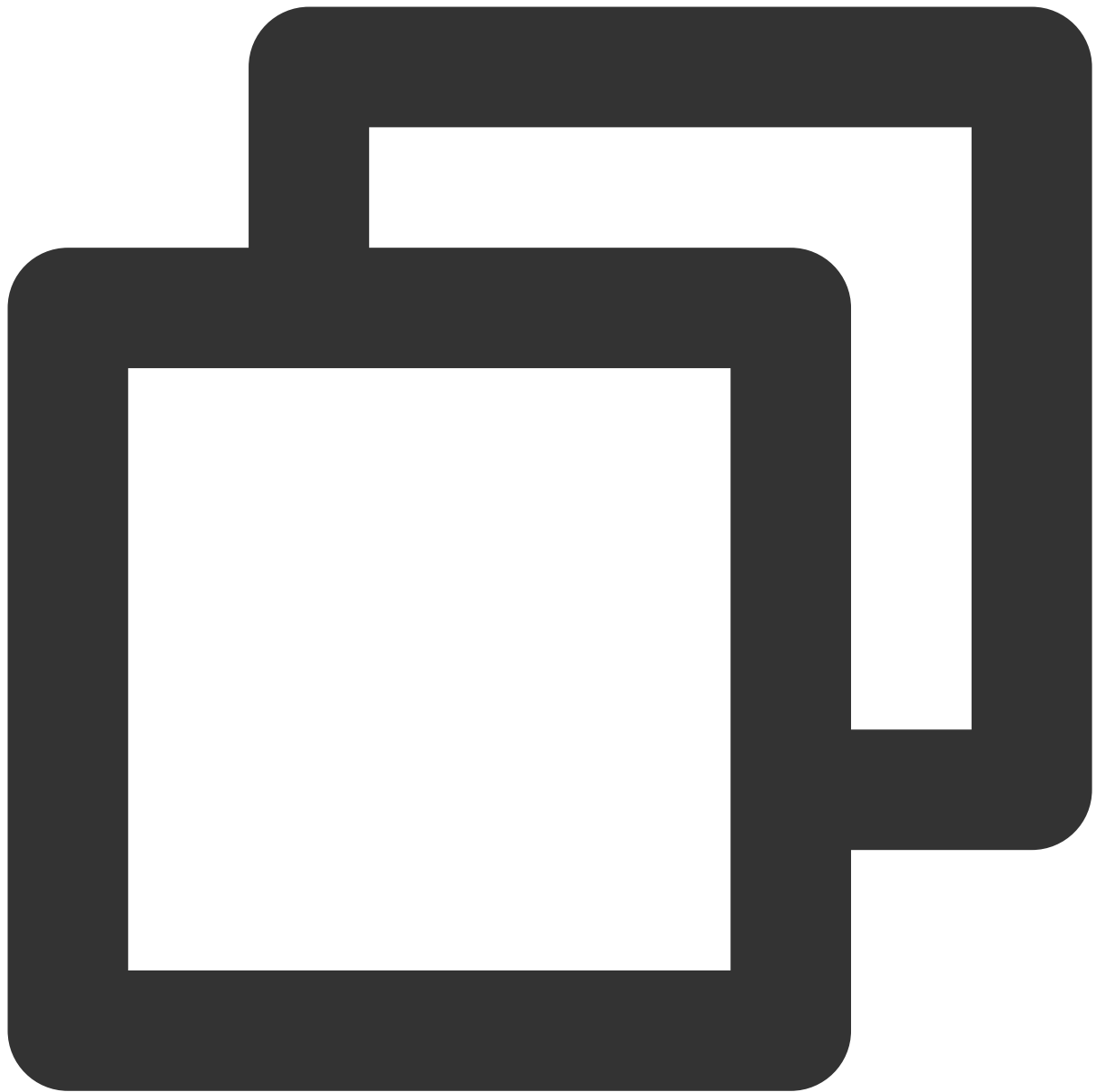
Pull a specific CocoaPods artifact by referring to the specific guide.

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
```