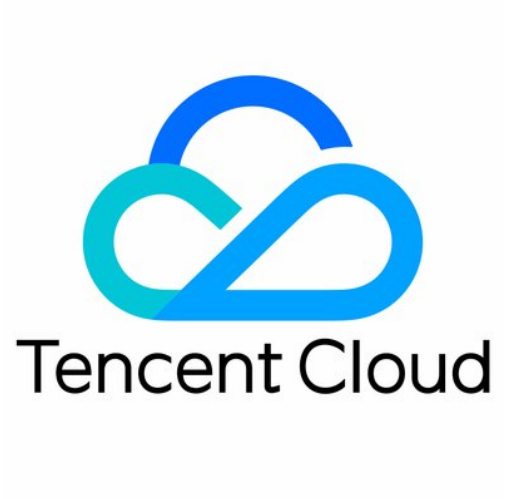
CODING Artifact Repositories

FAQs

Product Documentation



FAQs

Last updated：2024-01-02 17:37:03

What types of artifact repositories are supported?

CODING-AR supports popular repository types, including Docker, Maven, npm, Generic, PyPI, and Helm.

What are the terms and their relationships in an artifact repository?

The terms in CODING-AR are **repository > package > version**:
Repository: used to manage different types of repositories and package resources under them. You can set their external access permissions.
Package: the basic artifact unit for external access and is used to describe the purpose and use instructions of the artifact.
Version: lists all artifacts under a package and records in detail the iterative updates and changes of each artifact.

How do external permissions work for artifact repositories?

Project: Project members have read and write permissions. Other members do not have read or write permission.
Team: Project members have read and write permissions. Other members in the company have read permission, but not write permission. Other members do not have read or write permission.
Public: Project members have read and write permissions. Non-project members and anonymous members have read permission, but not write permission.

What are the package naming rules in artifact repositories?

Package names must be composed of 1 to 31 letters, numbers, underscores (_), hyphens (-), and periods (.).
Package names must be unique within the repository.

What are the package settings?

Package settings include: license, package description, maturity, website URL, issue tracking URL, version control URL, etc.

Maven Issues

Where can I find the Maven settings.xml configuration file?

When creating a Maven artifact, you must configure your settings file. This file is usually stored in the locations below. Files in different locations have different scopes and priorities. You can use the following locations as needed:

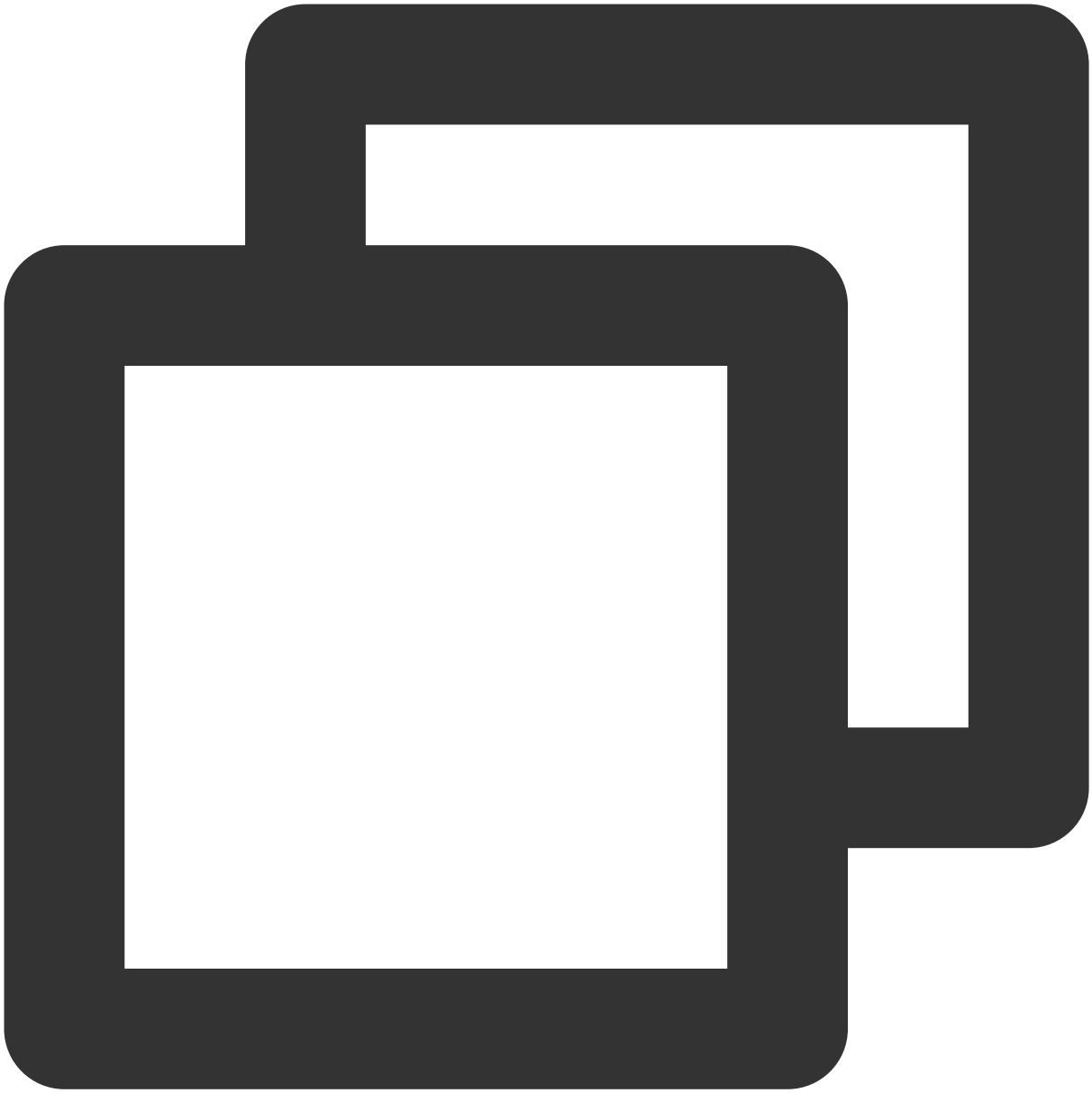
1. Global configuration: `${M2_HOME}/conf/settings.xml`

   If you do not remember the Maven installation directory `${M2_HOME}`, you can run `echo ${M2_HOME}` or `mvn -version` in the terminal to see the Maven home path.

2. User configuration: `${user.home}/.m2/settings.xml`

   You can use the echo environment variable to find this file directory. Sometimes, this directory does not contain a settings.xml file. In this case, copy the global settings.xml file to this directory and modify it.

3. settings.xml under a specified path



```
mvn deploy --settings settings.xml
```

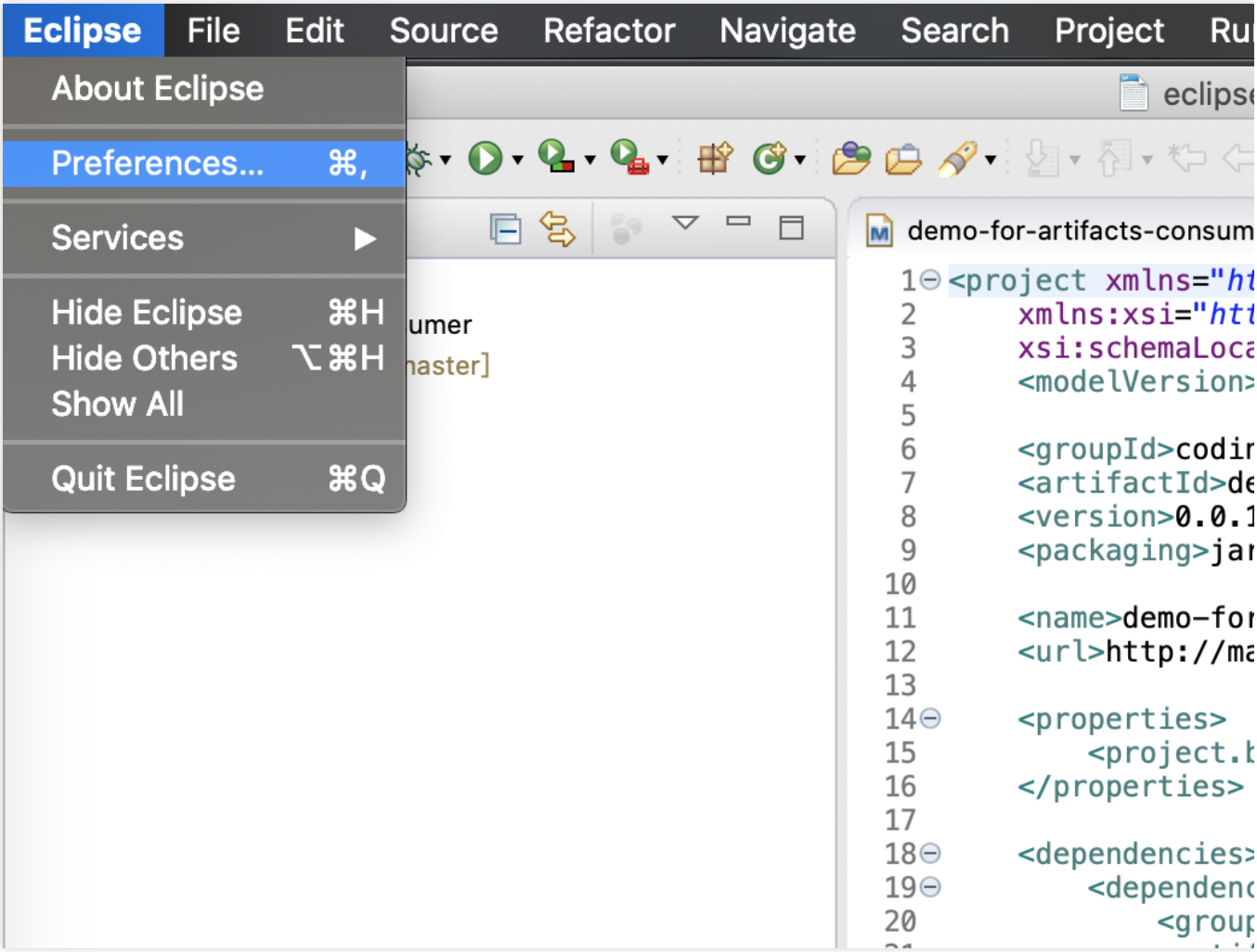
Note:

For the mvn command, the priorities of settings.xml files are: Specified path > User configuration > Global configuration.

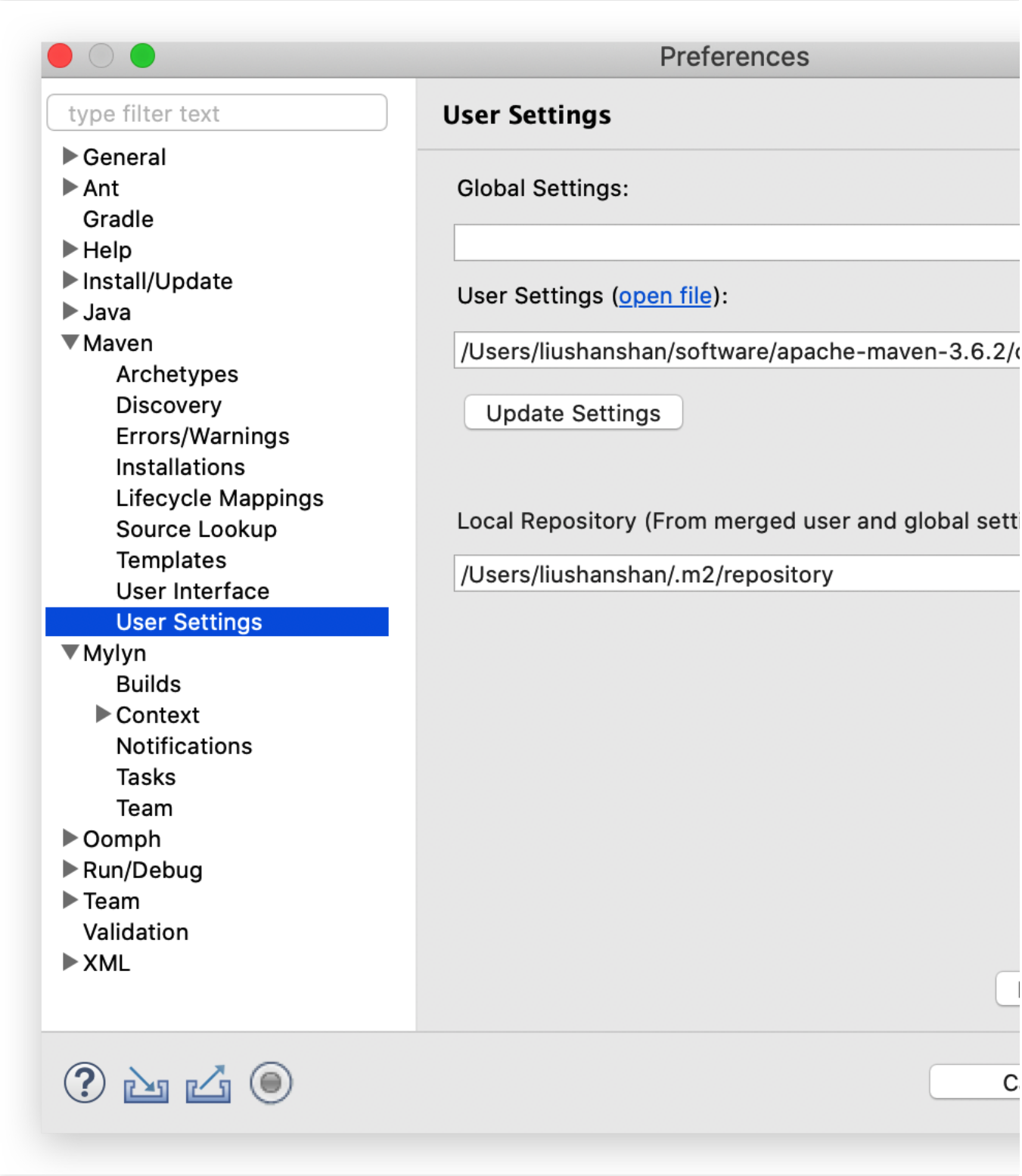
In addition to the mvn command, how do I modify the settings.xml file in IDEs such as Eclipse?

We will use Eclipse as an example (for other IDEs, you can search for the references online):

1. Click **Preferences**.



2. Go to **Maven** > **User Settings** to view your configuration file path and modify the configuration file.



npm Issues

How can I specify npm @scope to a CODING private artifact repository?

1. You can configure **.npmrc** to specify the @scope registry.

For example: For an npm package with the following location information:

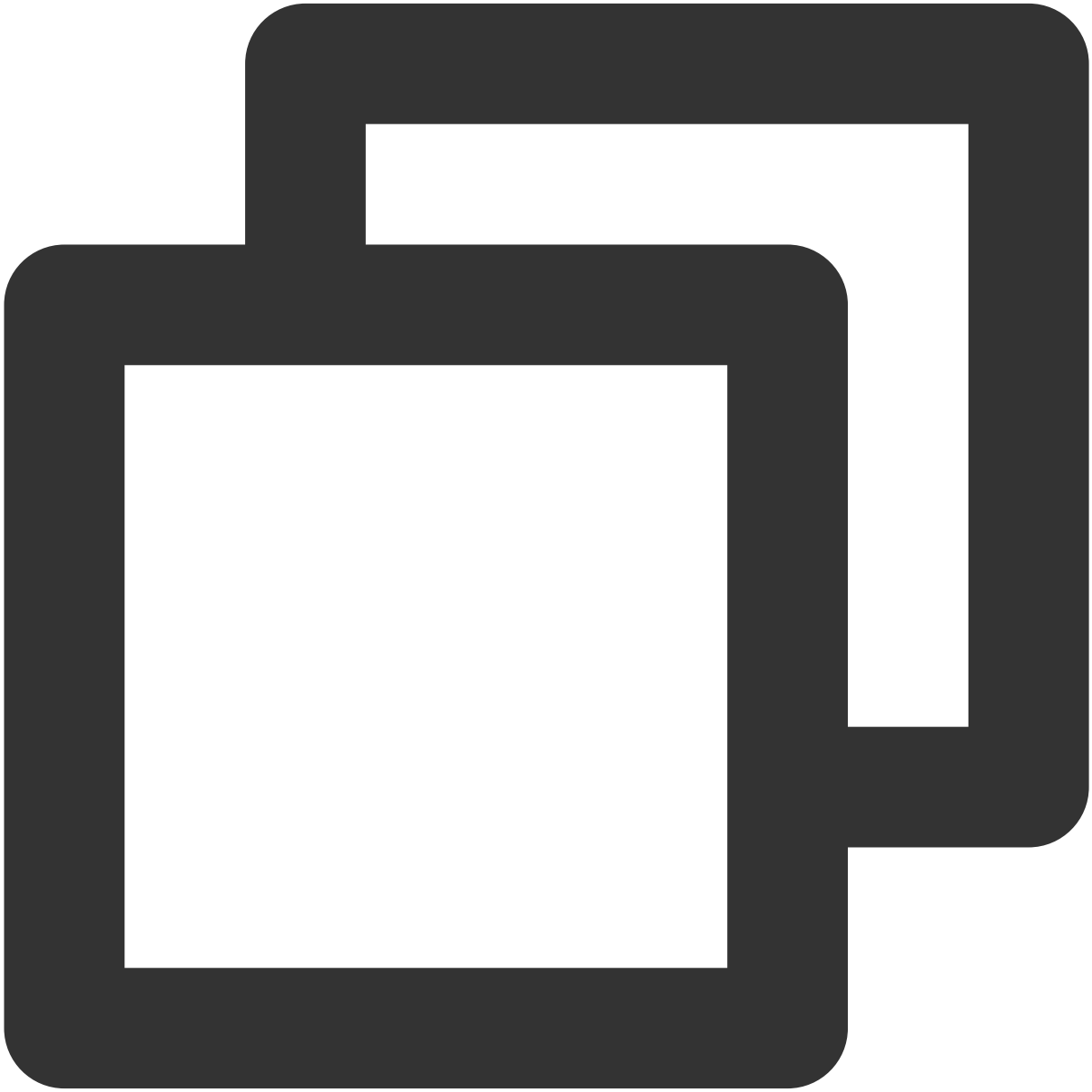
Company: my-team

Project: my-project

Artifact repository: my-npm-repo

Name: @my-scope/my-pkg.

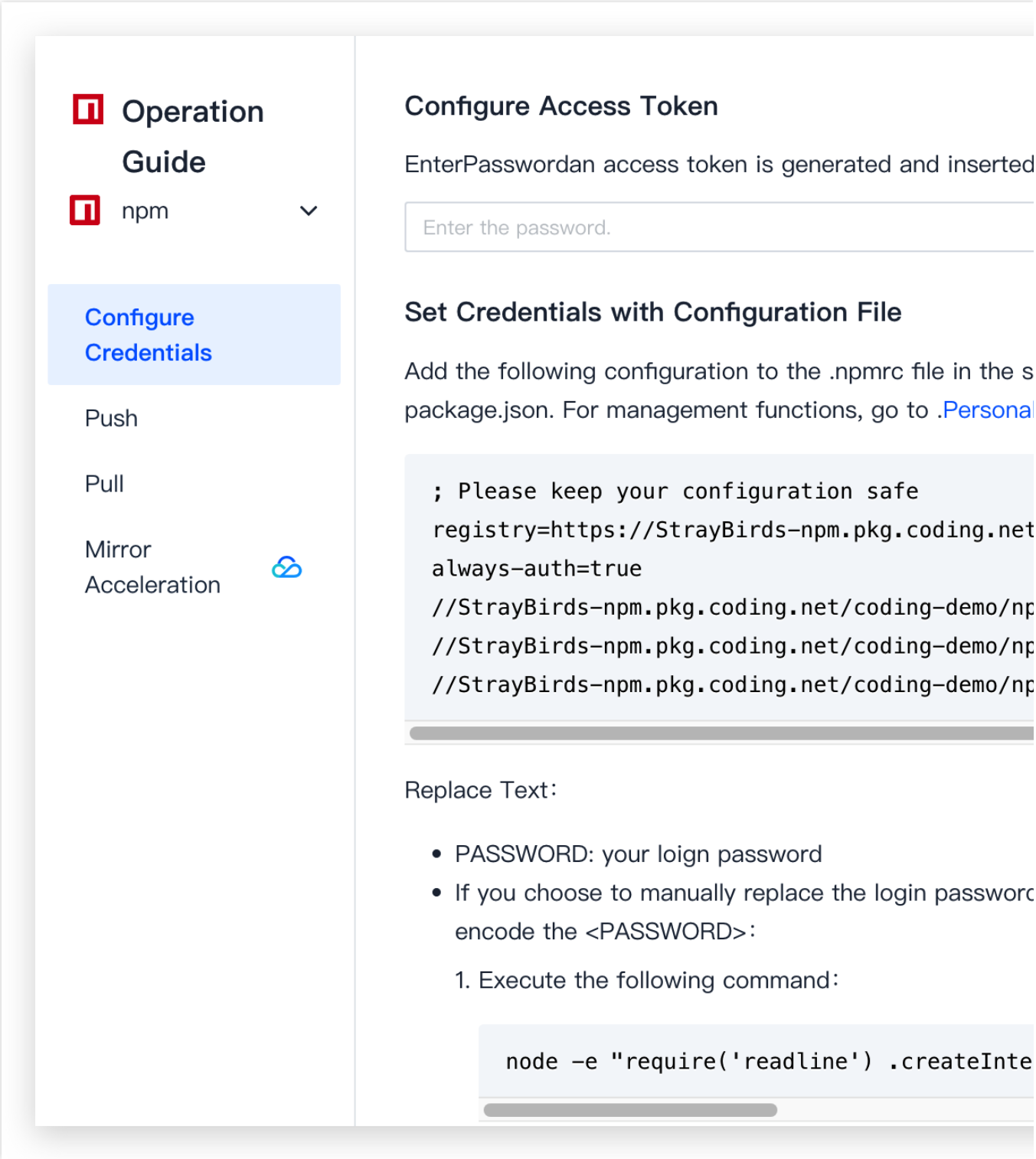
You can configure `.npmrc` to set `@my-scope/my-pkg` in package.json to this URL:



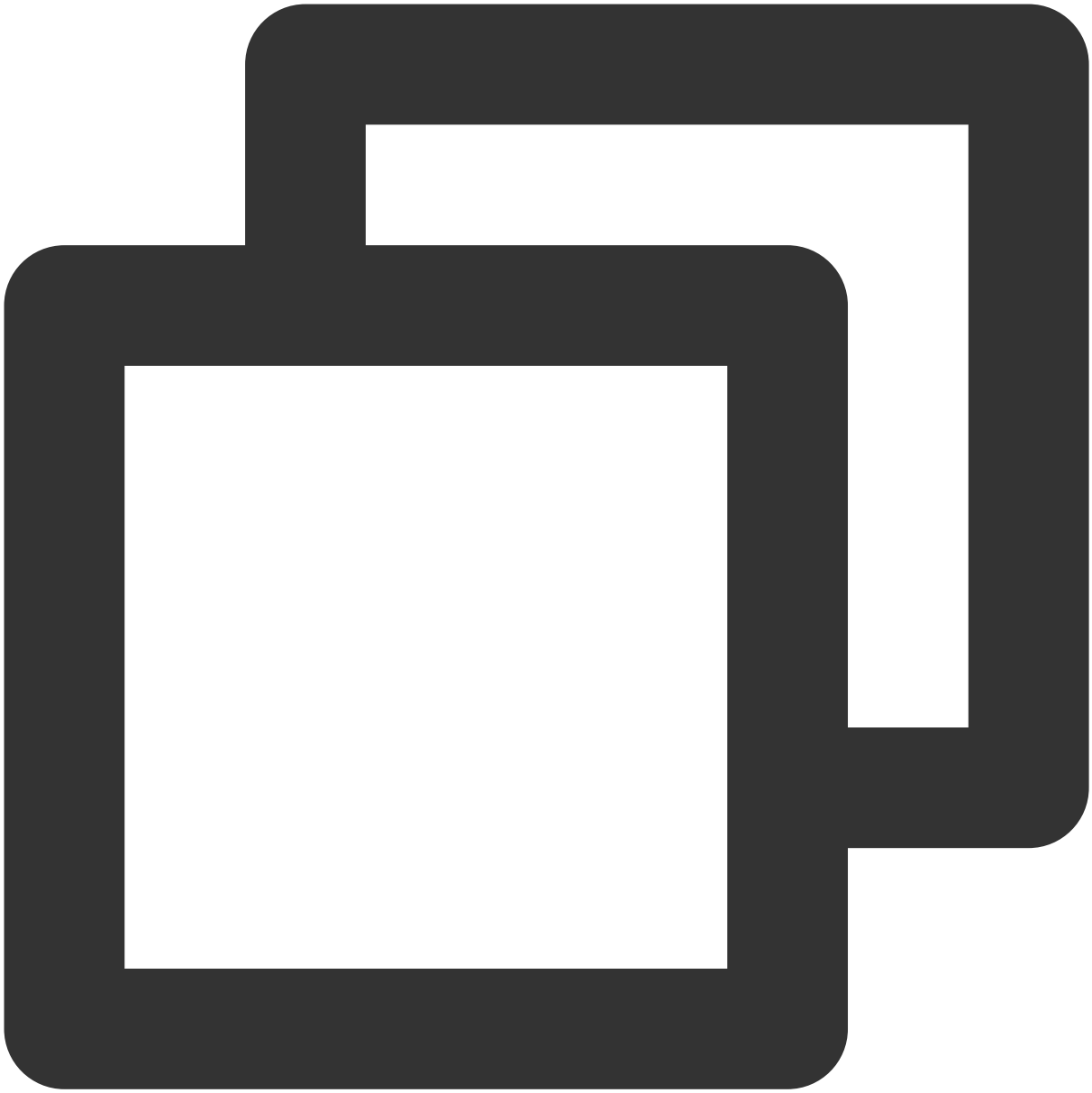
```
https://my-team-npm.pkg.coding.net/my-project/my-npm-repo/
```

2. Set the npm package registry to the CODING-AR repository.

Click **Generate configuration from access token** on the npm repository guide page to generate the .npmrc file.



Be sure to save the generated configuration:



```
registry=https://my-team-npm.pkg.coding.net/my-project/my-npm-repo/
always-auth=true
//my-team-npm.pkg.coding.net/my-project/my-npm-repo/:username=xxxxxx
//my-team-npm.pkg.coding.net/my-project/my-npm-repo/:_password=xxxxx
//my-team-npm.pkg.coding.net/my-project/my-npm-repo/:email=xxxxx
```

CODING npm repository supports proxies. You can also set the npm registry to the CODING-AR repository to pull public artifacts.

Note:

For how to use npm repositories in CODING Continuous Integration, see **Continuous Integration** > **Build npm Artifacts**.

Permission Issues

How can I pull artifacts from the repositories of other CODING projects?

You can use project tokens to pull artifacts from other CODING repositories.

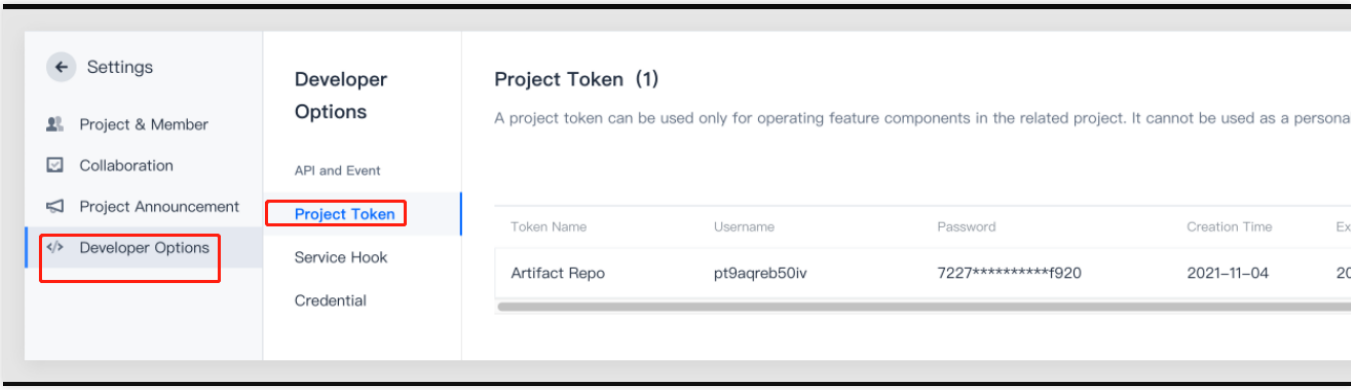
In this example, we will use two different projects:

The project from which artifacts are pulled is "Project A".

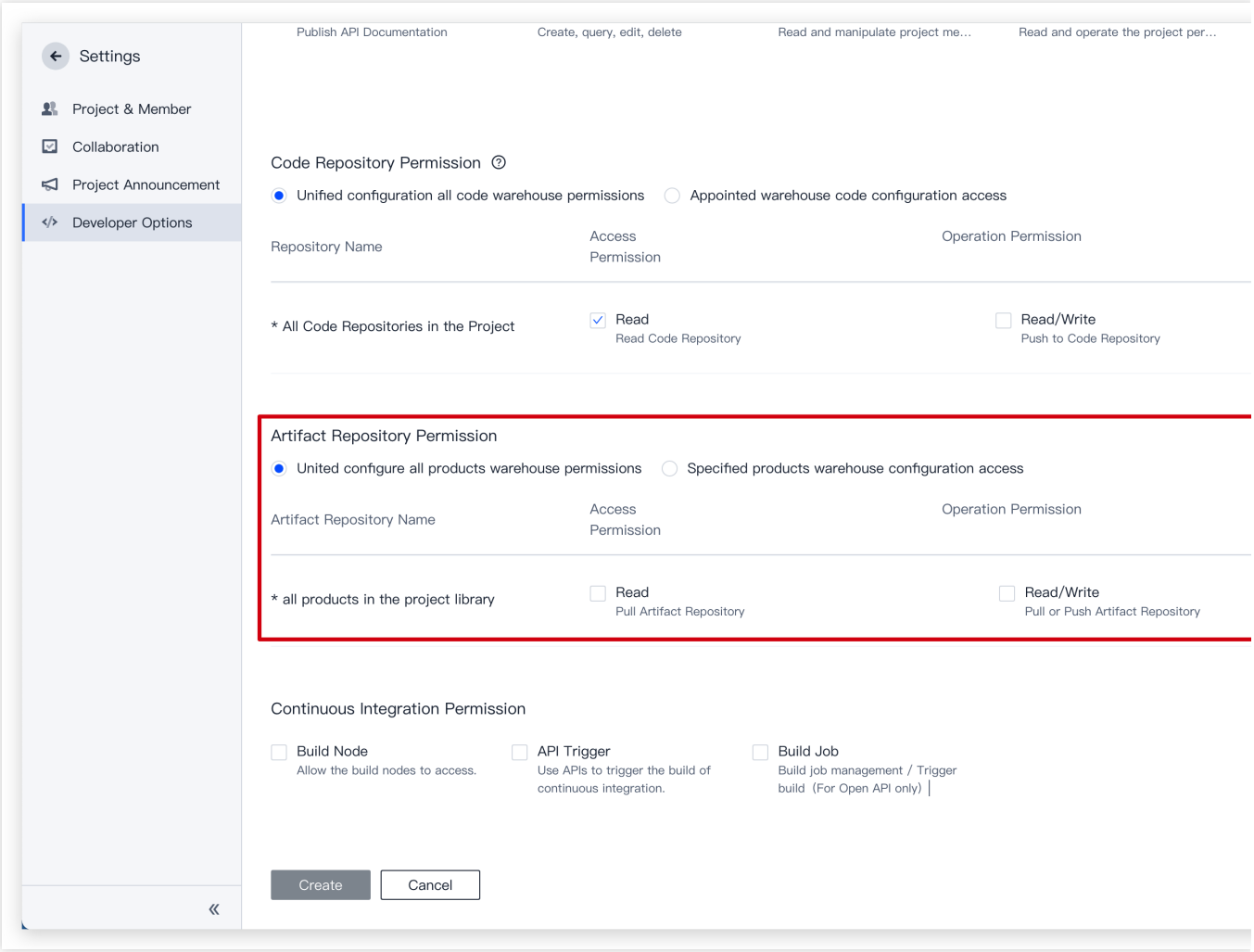
The project that pulls artifacts is "Project B".

Step 1: create a project token in Project A

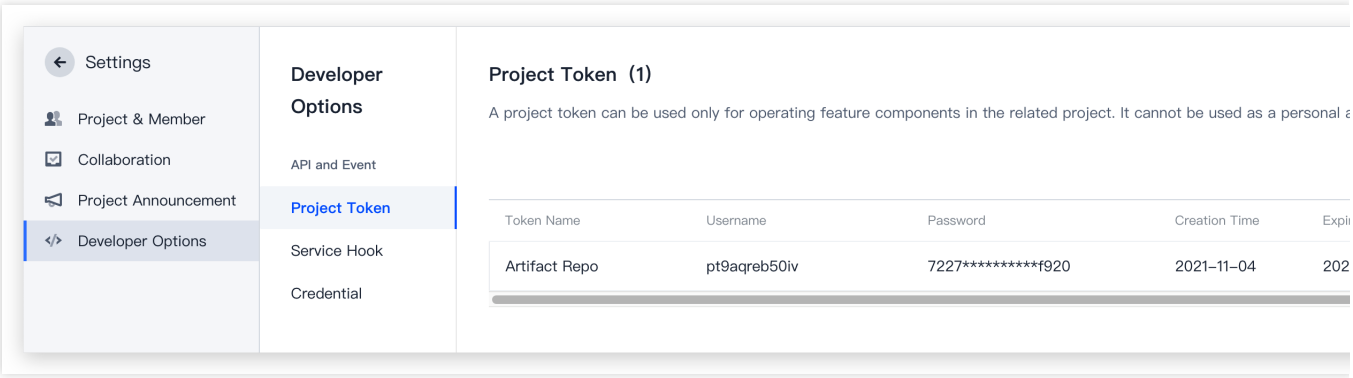
1. Open project A, go to **Project Settings** > **Developer Options** > **Project Token**, and click **Create Project Token**.



2. Configure the artifact repository permissions.

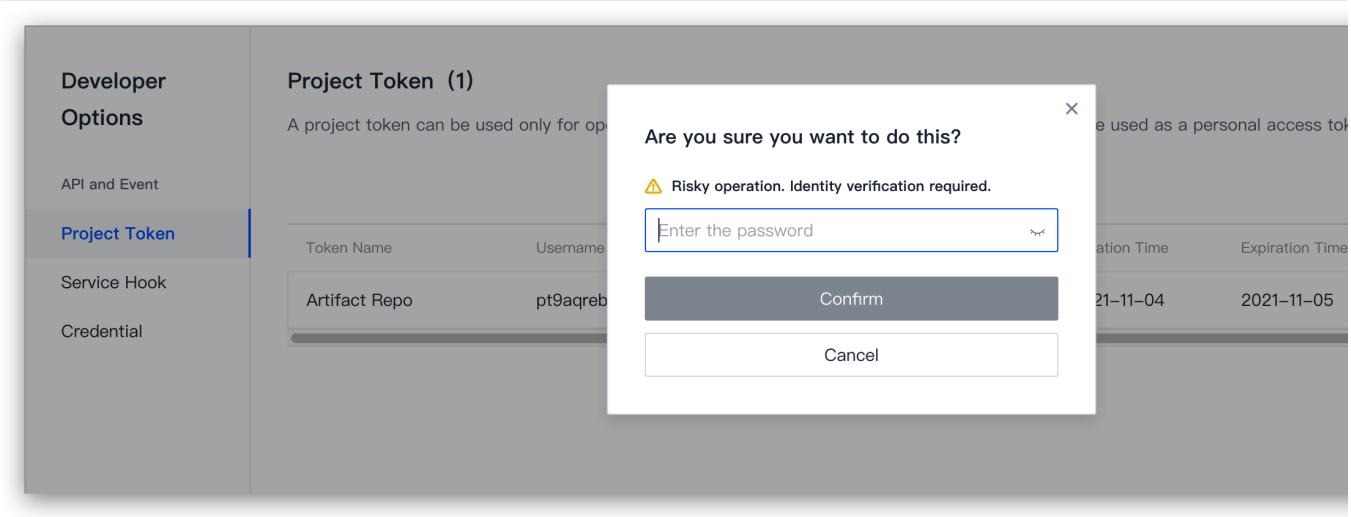


3. Click **OK** to create the token.



Step 2: use the token created in Project A as the username and password in Project B to pull artifacts

1. Configure the authentication information based on your artifact type.
2. Go back to the token page in Project A and click "View Password".



3. When configuring authentication information for the repository in Project B, use the project token username and token password as the username and password.
4. After entering the correct information, you can pull artifacts from other CODING project repositories.

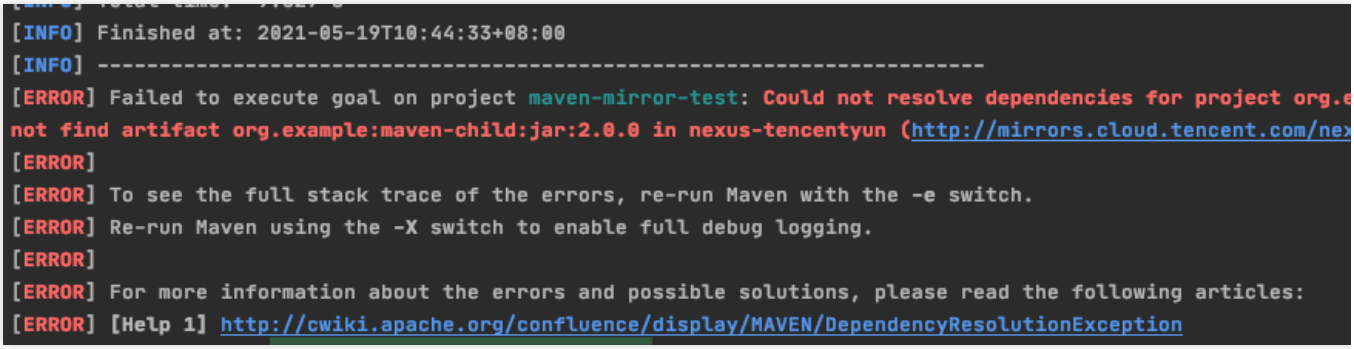
Why does the artifact repository contain dependencies that were not pushed?

This occurs when the proxy feature is enabled in the repository. If a proxy is used, the artifact repository will serve as a centralized platform to help you manage third-party artifact dependencies by pulling dependencies from the proxy. You can track dependencies in your team in CODING-AR and perform dependency vulnerability scans for dependency audits.

Mirror Issues

Why can't I pull dependencies from an artifact repository?

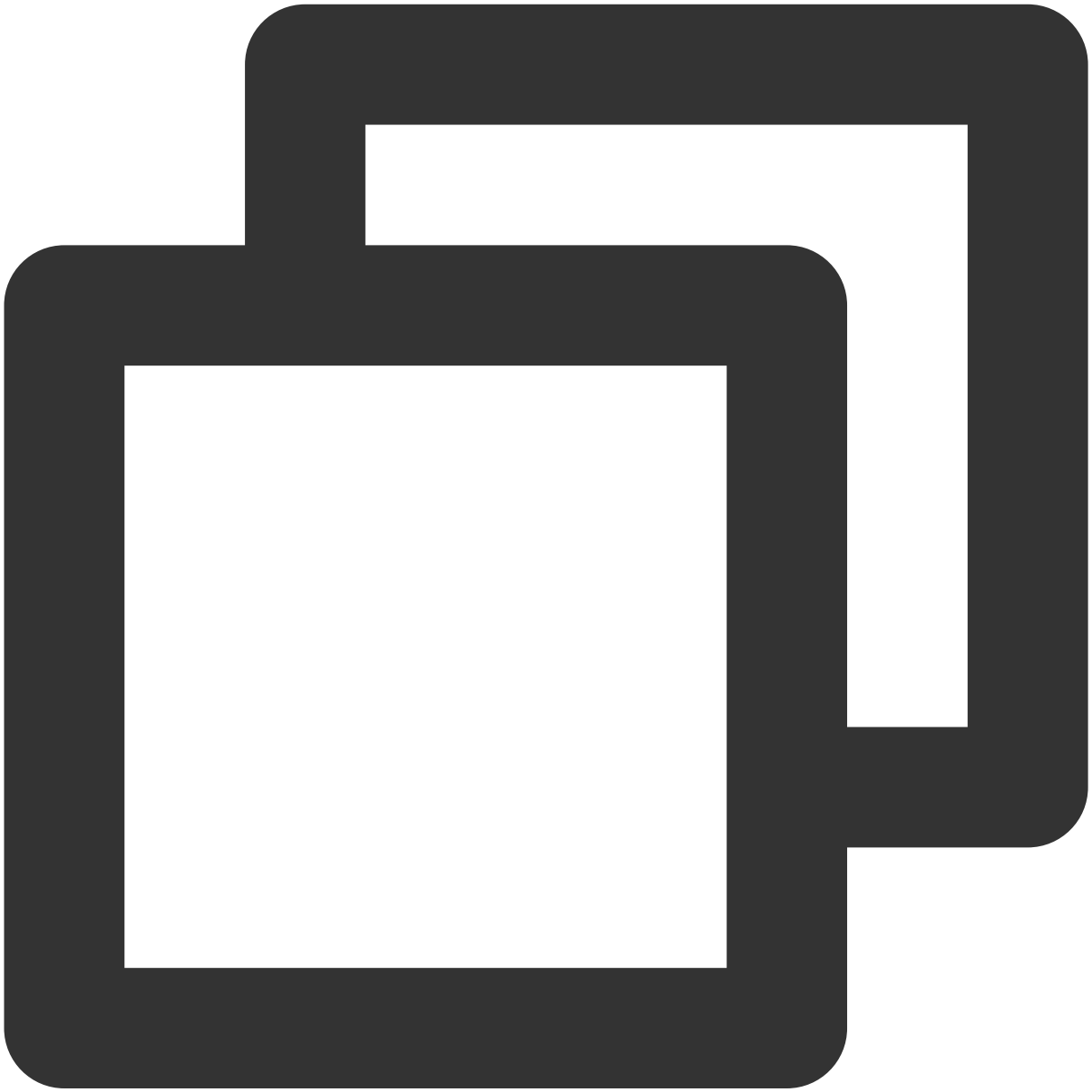
If the mirrors parameter is used for acceleration, you may fail to pull the dependency from the artifact repository for builds, as shown in the screenshot below:



This problem may occur when the mirror is configured to be used for pulling in `<mirrorOf>*</mirrorof>` in

`<mirror>` , but the mirror does not contain the dependency in CODING-AR. There are two solutions to this problem.

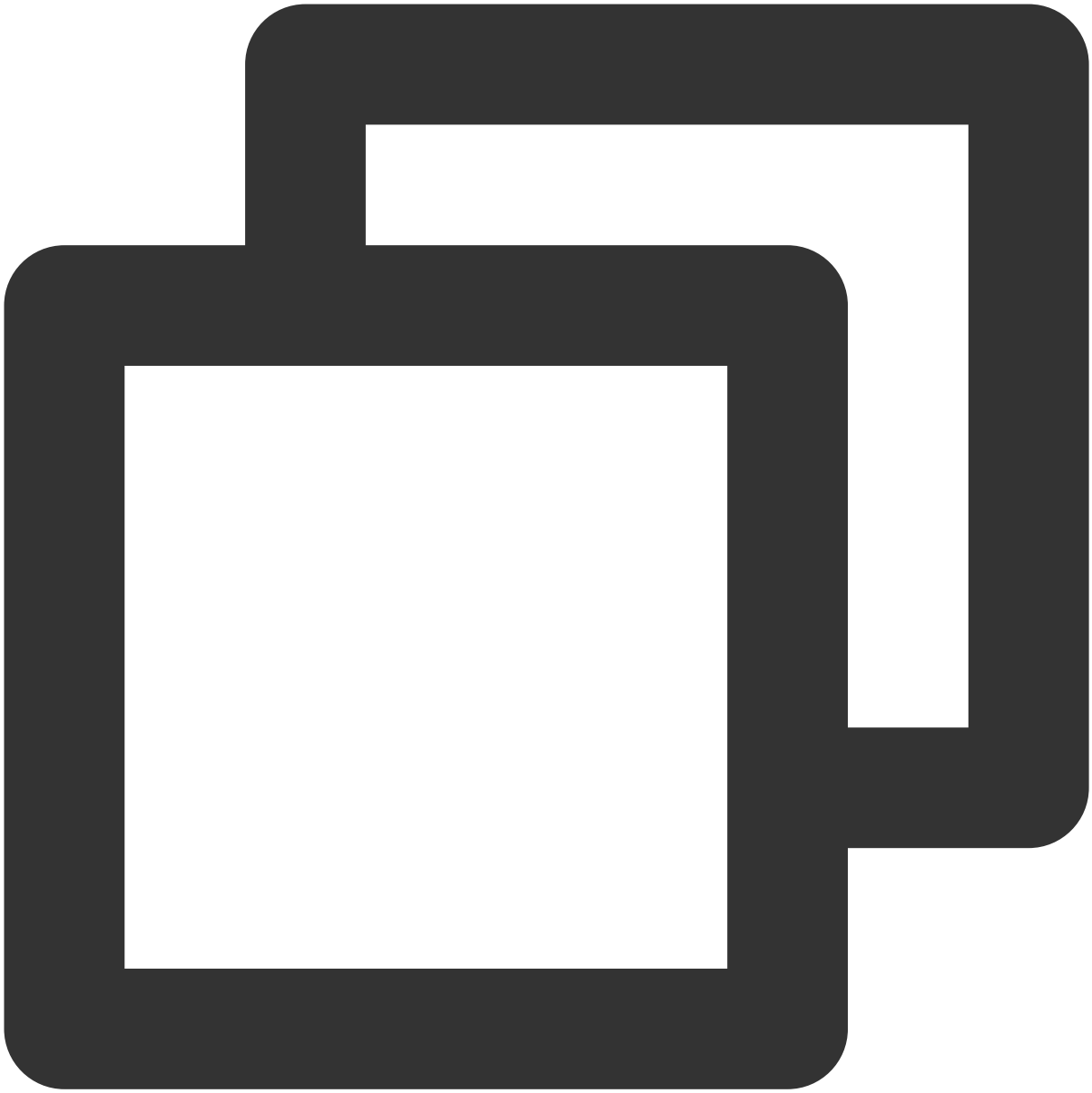
Method 1: modify the parameter configuration to only allow artifacts for non-CODING repositories to be pulled from the mirror.



```
<settings>
    <!-- Configure profiles according to the pull guide in the CODING repository --
    <profiles>
        <profile>
            <id>Repository Proxy</id>
```

```
            <activation>
                <activeByDefault>true</activeByDefault>
            </activation>
            <repositories>
                <repository>
                    <id>coding-maven-repo-id</id>
                    <name>coding-maven-repo-name</name>
                    <url>https://coding-maven-repo-url</url>
                    <releases>
                        <enabled>true</enabled>
                    </releases>
                    <snapshots>
                        <enabled>true</enabled>
                    </snapshots>
                </repository>
            </repositories>
        </profile>
    </profiles>
    <mirrors>
        <mirror>
            <id>nexus-tencentyun</id>
            <!-- Only artifacts not from the coding-maven-repo-id source can be pul
            <mirrorOf>!coding-maven-repo-id</mirrorOf>
            <name>Nexus tencentyun</name>
            <url>http://mirrors.cloud.tencent.com/nexus/repository/maven-public/</u
        </mirror>
    </mirrors>
</settings>
```

Method 2: delete the `<mirrors>` configuration and add the mirror in the CODING-AR repository. This will proxy and save all open-source dependencies to CODING-AR.

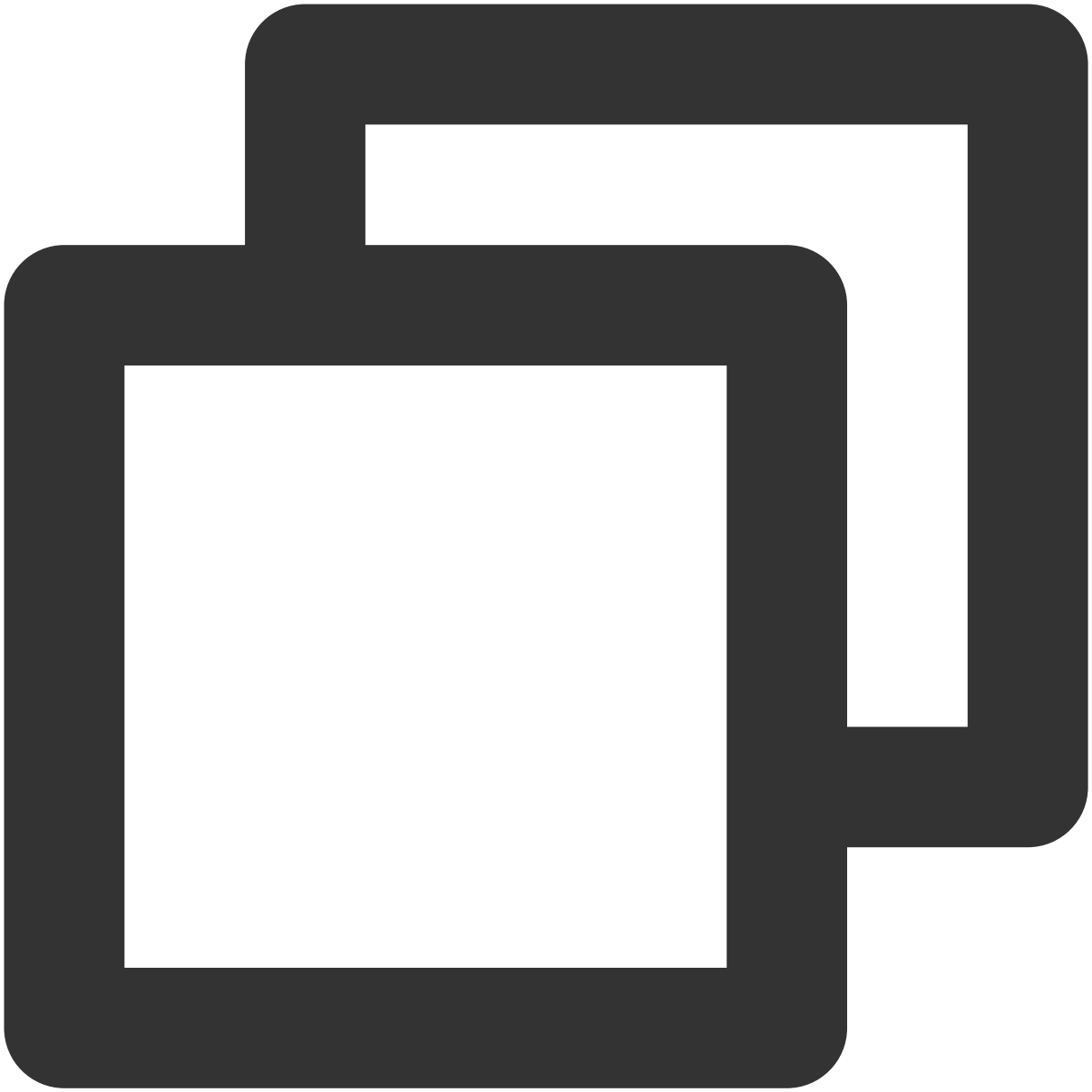


```
<settings>
    <!-- Configure profiles according to the pull guide in the CODING repository --
    <profiles>
        <profile>
            <id>Repository Proxy</id>
            <activation>
                <activeByDefault>true</activeByDefault>
            </activation>
            <repositories>
                <repository>
                    <id>coding-maven-repo-id</id>
```

```
                    <name>coding-maven-repo-name</name>
                    <url>https://coding-maven-repo-url</url>
                    <releases>
                        <enabled>true</enabled>
                    </releases>
                    <snapshots>
                        <enabled>true</enabled>
                    </snapshots>
                </repository>
            </repositories>
        </profile>
    </profiles>
</settings>
```

Note:

If the CI mirror acceleration is configured in the global environment configuration `${M2_HOME}/conf/settings.xml`, you must overwrite it with the project's `settings.xml` configuration.



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