

CODING Code Repositories

Operation Guide

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

Operation Guide

Code Repository Management

- Create Repository
- Import or Associate External Repository
- View Repository Details
- Set Basic Repository Information
- Archive Repository
- Delete and Reset Repository
- Restore Deleted Repository
- Manage Code Repository Cards
- Manage Repositories Via Local Command Lines

SVN Repository Usage

- Create SVN Repository
- Access SVN Repository
- Manage SVN Directory Permissions

SSH Protocol Usage

- Configure SSH Public Keys
- Key Fingerprint Authentication
- Push/Pull Code Via SSH Protocol

Branch Management

- Create Branch
- Set Default Branch
- Set Protected Branch
- Set Hidden Branch
- Use Code Owner Mechanism

Merge Request and Code Review

- Adjust Merge Request Settings
- Merge Branch
- Review Merge Request

Versions and Tags

- Manage Version Release
- Manage Version Tag

Code Repository Security

- Inspect Repository Security Risks
- Set Repository Access Method

Sign Git Commits with GPG

Enable Verification of Committers and Commit Messages

Lock Files and Folders

Git Basics

Common Commands

LFS Support

Go Get Support

Personal Settings

Access Token

SSH Public Keys and Project Tokens

Operation Guide

Code Repository Management

Create Repository

Last updated : 2023-12-25 17:08:18

This document describes how to create repositories.

Open Project

1. Log in to the CODING Console and click the team domain name to go to CODING.
2. Click



in the upper-right corner to open the project list page and click a project icon to open the corresponding project.

3. In the menu on the left, select **Code Repositories**.
4. If Code Repositories is not shown on the left, the project admin needs to go to **Project Settings > Projects and Members > Functions** to enable the relevant function.

Create Git and SVN Repositories

Create a Git repository

1. In the code repository list, click **Create Code Repository** in the top-right corner and select **Create from Scratch**.
2. Select Git as the repository type and enter a valid repository name.
3. We recommend you select the **Generate README File** option. When this is enabled, the Git repository is automatically initialized after creation.
4. We recommend you select **Private Repository**. Enable the **Code Scanning** function as needed to discover and avoid potential code issues.
5. Click **Create**.

←

Create Code Repository

Normal

Template

Import

Repository Type *

Repository Name *

Git

Repository

/

The repository name can contain only letters, digits, underscores (_), 0/100

Repository Description

Please enter warehouse description

Quickly Initialize Repository

Generate a README file

☑

Add the .gitignore file.

Select the .gitignore file

Make the repository public or not

☑

 Private warehouse (only visible to warehouse members, who have access to the warehouse)

☐

 Open repository (after open, anyone can access the code repository, please consider carefully!)

Code scanning

☐

Enable code scan to discover code problems such as security vulnerabilities and functional defects in the code. The results will be displayed in the merge request details to assist you in code review. [View details](#)

Submit

Cancel

Note:

After the repository is initialized, you can use Git commands to associate it when a local repository. For details, see [Common Git Commands](#).

Create an SVN repository

CODING supports the creation of SVN repositories. For more information about creating and using SVN repositories, see [SVN Repository Usage](#).

Create from Template

CODING provides a preset code repository module. You can use sample code to learn how the code repository module works with continuous integration and artifacts.

← Create Code Repository | Normal **Template** Import


Repository Name *


The repository name can contain only letters, digits, underscores (_), hyphens (-), and periods (.) 0/100


Repository Description


Please enter warehouse description


Select preset template * ?


 **spring-demo** ☒
Java
Based on a simple Java web application, take you to experience code function modules.

 **ruby-on-rails-demo** ☐
Ruby on Rails
A simple Ruby on Rails web application that takes you through code modules.

 **ruby-sinatra-demo** ☐
Ruby Sinatra
A simple Ruby Sinatra web application that takes you through the code model.

 **express-demo** ☐
Node.js
Based on the simple Node.js web application, take you to experience the code function modules.

 **android-demo** ☐
Android
Based on simple Android APP, take you to experience code function modules.

 **flask-demo** ☐
Python
The Simple Python Flask web app takes you through code modules.

Make the repository public or not
☒ Private warehouse (only visible to warehouse members, who have access to the warehouse)
☐ Open repository (after open, anyone can access the code repository, please consider carefully!)

Code scanning ☐
Enable code scan to discover code problems such as security vulnerabilities and functional defects in the code. The results will be displayed in the merge request details to assist you in code review. [View details](#)

Submit

Cancel

Import an External Repository

You can quickly migrate existing Git repositories to the CODING DevOps platform. For details, see [Import or Associate External Repository](#).

Import or Associate External Repository

Last updated : 2023-12-25 17:08:18

Open Project

1. Log in to the CODING Console and click the team domain name to go to CODING.
2. Click




in the upper-right corner to open the project list page and click a project icon to open the corresponding project.

3. In the menu on the left, select **Code Repositories**.

4. If Code Repositories is not shown on the left, the project admin needs to go to **Project Settings > Projects and Members > Functions** to enable the relevant function.

Import External Code Repositories

CODING-CR provides a quick import function for external open-source repositories and allows you to sync code with external repositories at regular intervals. When creating a code repository, select **Import External Repository** and enter the URL of the open-source Git repository to import.

 **Create Code Repository**

Normal

Template

Import

Git repository URL *

Please enter the warehouse address to clone, such as https://github.com/Coding/WebIDE.git

Repository Name *


The repository name can contain only letters, digits, underscores (_), hyphens (-), and periods (.) 0/100

Make the repository public or not

☒ Private warehouse (only visible to warehouse members, who have access to the warehouse)

☐ Open repository (after open, anyone can access the code repository, please consider carefully!)

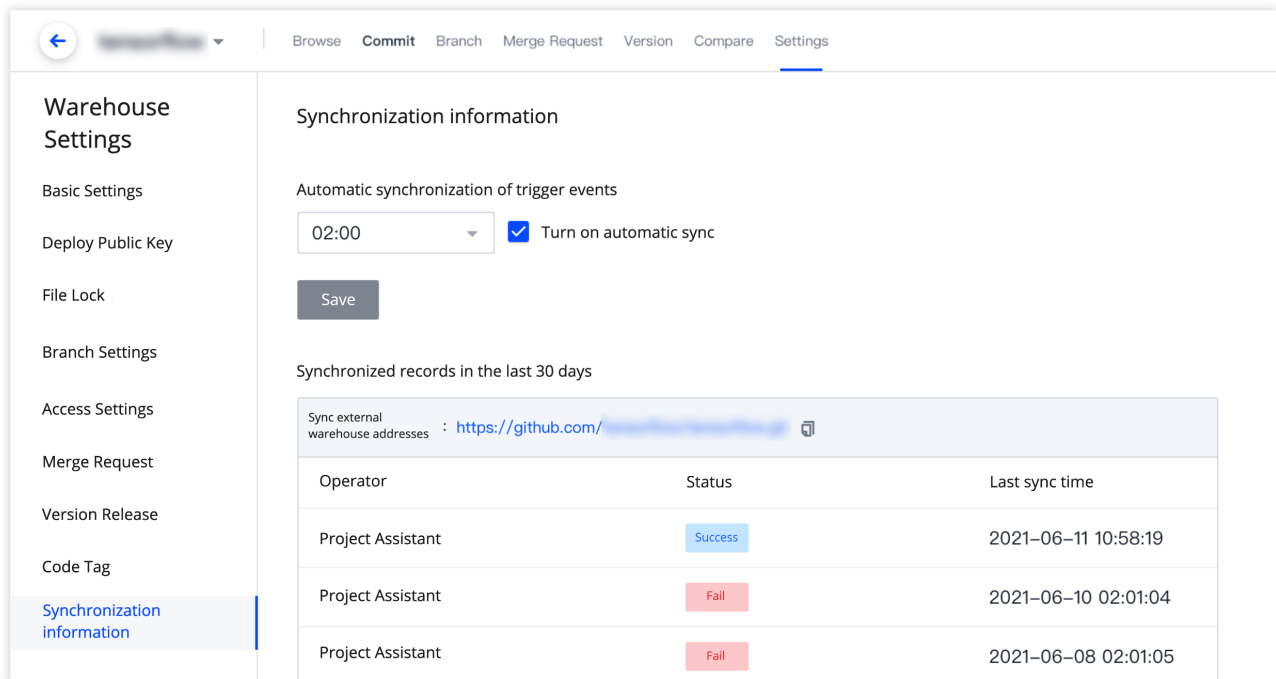
Code scanning ☐

Enable code scan to discover code problems such as security vulnerabilities and functional defects in the code. The results will be displayed in the merge request details to assist you in code review. [View details](#) 

Submit

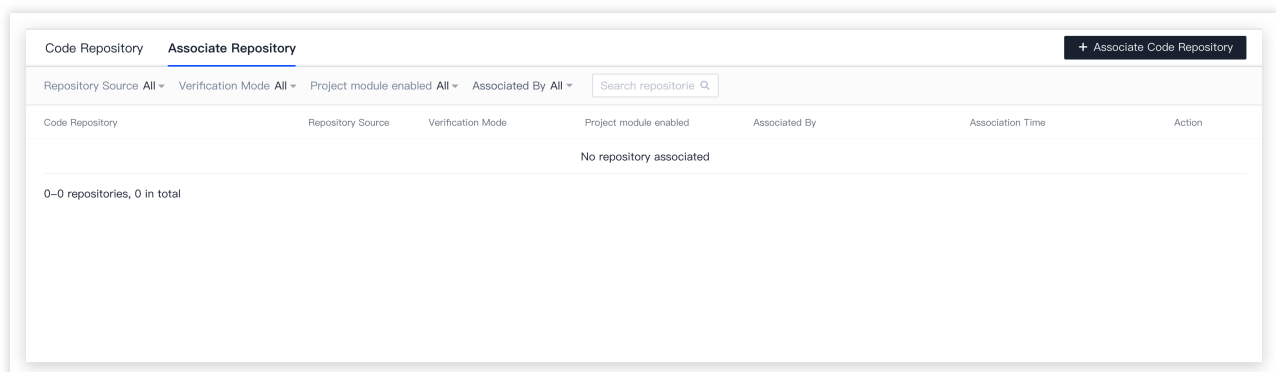
Cancel

You can sync the repository with the source repository and the changes made in CODING will be overwritten. You can change the sync frequency or disable auto sync in the repository settings.



Associate a Code Repository

The **Associate Repository** function allows you to temporarily store the credentials used to access an external repository in CODING. When using **Continuous Integration** or **Continuous Deployment**, you can directly use the third-party repository as a code source without repeated migration.



GitHub, GitLab, private GitLab, Gitee, TGit, and common Git repositories can be associated with CODING repositories. These five repository types support OAuth verification. Common Git repositories support account password verification. After association, repository code will not be stored in the CODING repository.

Associate Code Repository

Repository Source

CODING GitHub GitLab Private GitLab Gitee TGit Common Git Repository

Verification Mode: OAuth

Authorizer: Steven, [Refresh OAuth Verification for GitHub](#)

Code Repository *
Please select

Associate Cancel

Associate a private GitLab repository

To associate a private GitLab repository, you must create an application in GitLab and then the team admin must bind the private GitLab service. For details, see [Bind Private GitLab](#).

Associate a GitLab SaaS repository

To associate a GitLab SaaS repository, Select **GitLab** as the code source on the **Associate Code Repository** page. Then, click **Verify Now** to go to the GitLab Authorization page and click **Authorize** to complete authorization. After successful authorization, select the code repository to associate.

Associate a GitHub repository

On the **Associate Code Repository** page, select the **GitHub** code source and click **Verify Now** to go to GitHub for OAuth authorization. If authorization fails, this may be because you did not enter your username in GitHub. In this case, go to **Settings > Profile > Name** and enter your username.

View Repository Details

Last updated : 2023-12-25 17:08:18

This document describes how to view repository details.

Open Project

1. Log in to the CODING Console and click the team domain name to go to CODING.
2. Click



in the upper-right corner to open the project list page and click a project icon to open the corresponding project.

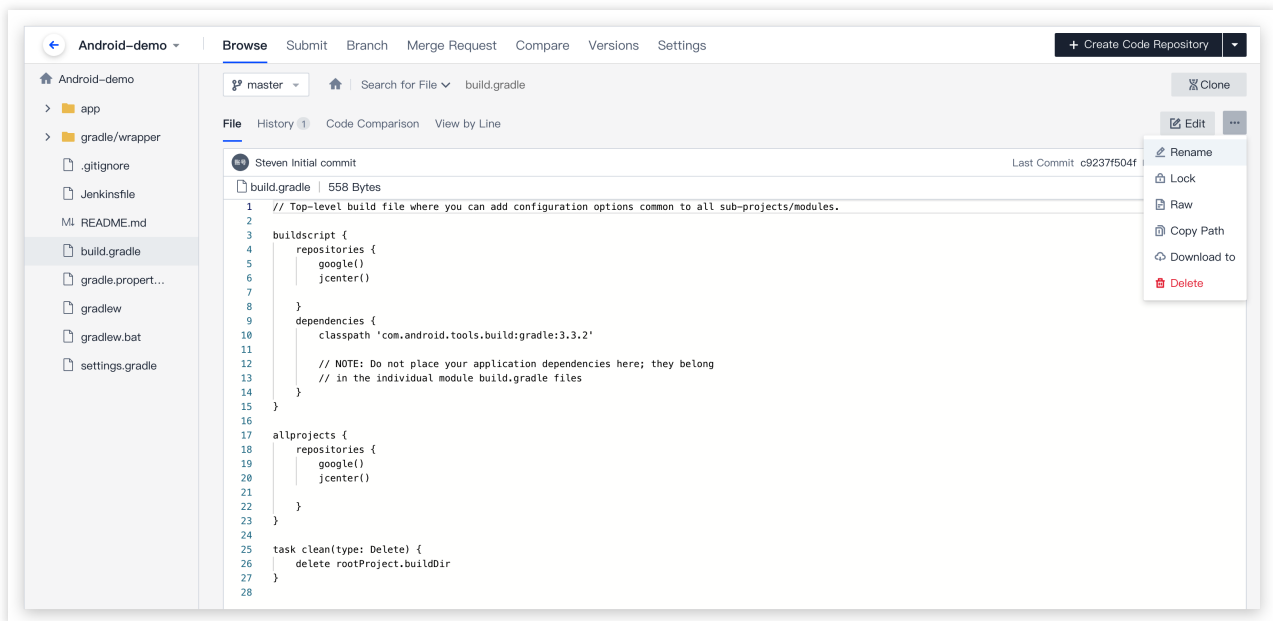
3. In the menu on the left, select **Code Repositories**.
4. If Code Repositories is not shown on the left, the project admin needs to go to **Project Settings > Projects and Members > Functions** to enable the relevant function.

On the **Code Repositories** page, click a repository name to go to the repository details page. The details page shows the files and commit history of the master branch by default.

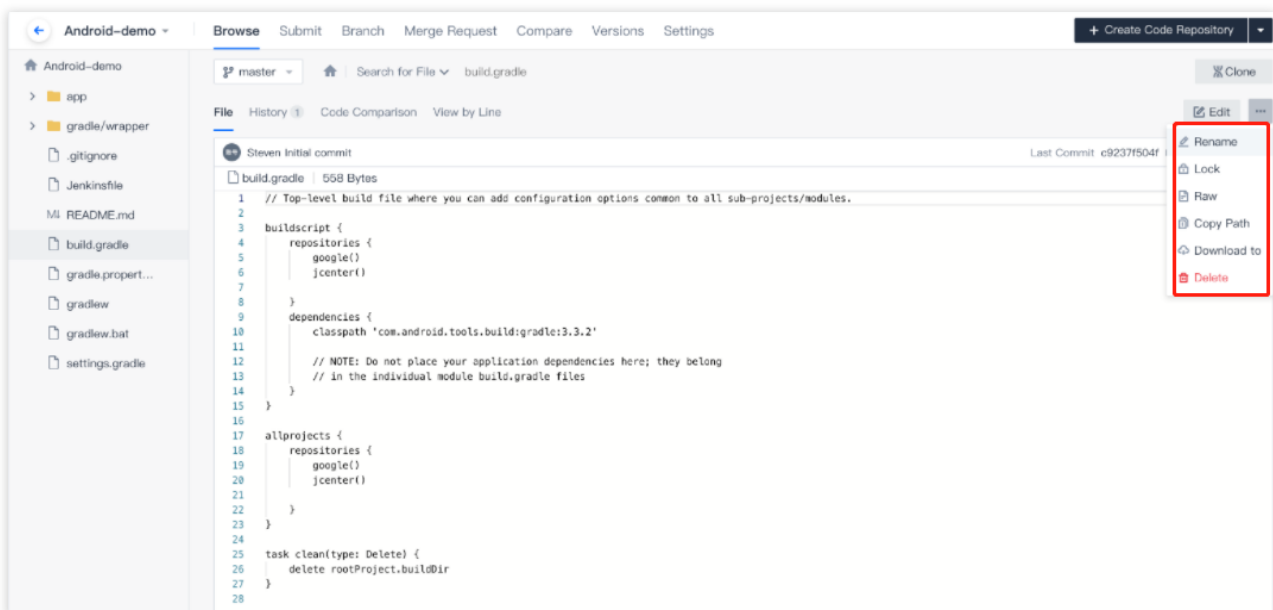
File List

On the repository details page, the file directory and content of the `readme` file of the master branch are shown by default.

Click any folder in the directory on the left to list all the files in this folder in the **Files** tab on the right. You can add, rename, lock, upload, download, or delete files and folders.



Click any file in the directory on the left to display the content of the file in the **Files** tab on the right. You can edit, rename, lock, download, or delete the selected file and view it in raw form.

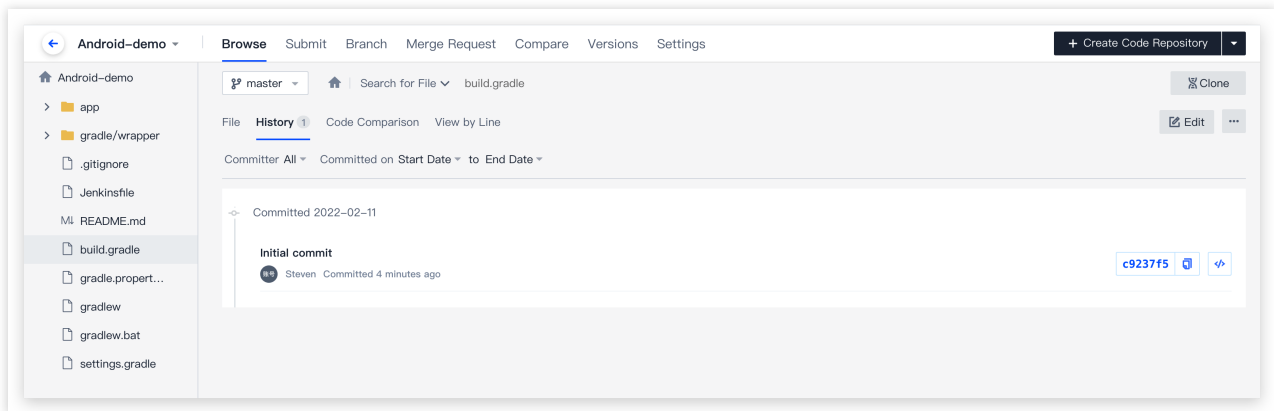


Note:

Hover over a file or folder in the directory to display the More Actions button. You can also click this button to perform the operations.

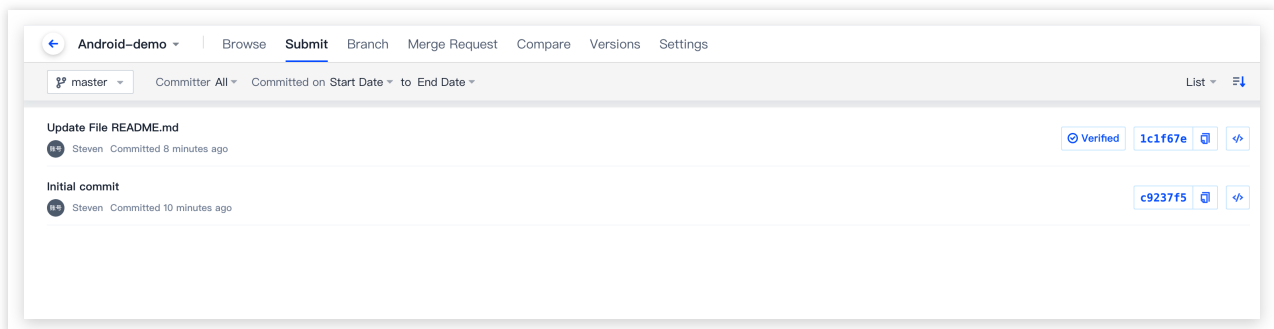
Commit History

1. Click the **History** tab, which shows the commit history of the master branch by default.
2. Commit records are sorted in reverse chronological order. Click the name of a commit record or the SHA ID on the right to go to the **Commits** tab for the current repository and view [Commit Details](#).

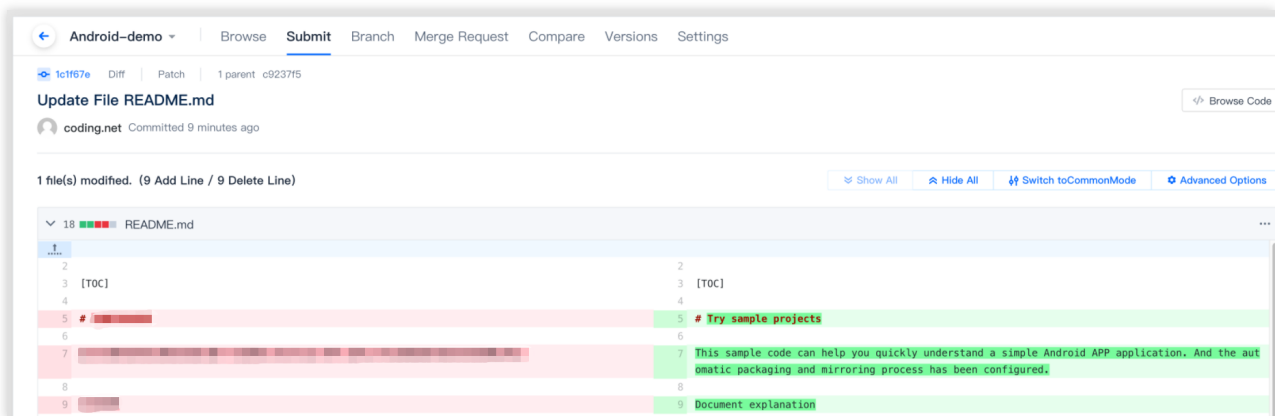


Commit Details

1. On the repository details page, click **Commits** to go to the commit record management page. By default, this page lists the commit records of the master branch in reverse chronological order. You can switch to another branch to view its commit history.



2. Click the name or SHA ID of a commit record to open the details page for this record in a new tab. This page lists all the files changed in this commit.

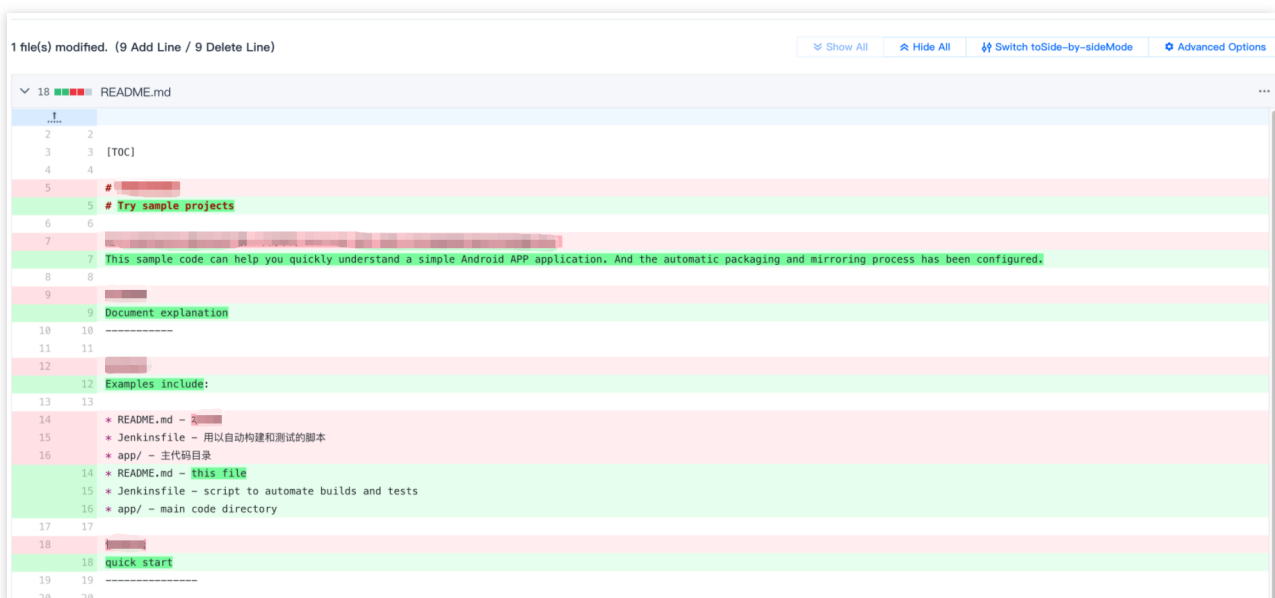


Note:

For line-by-line code comparison, you can use [Normal Mode](#), [Side-by-Side Mode](#), and [Advanced Options](#) to locate code differences easily. You can also comment on specific lines of code.

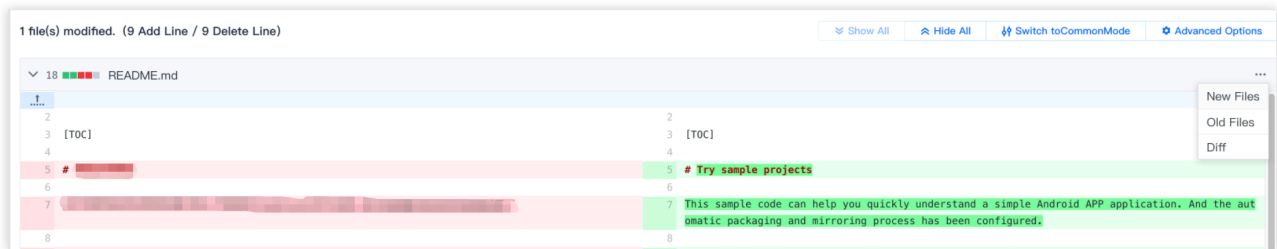
Normal Mode

In normal mode, all files changed in this commit are shown. Expand a file to see specific changes. The specific changes are displayed line by line in a file. Additions are highlighted in green and deletions are highlighted in red.



Side-by-Side Mode

Click **Switch to Side-by-Side** to switch the code comparison mode from Normal to Side-by-Side. This mode gives you a clear view of the content before and after the changes. You can see the lines and content that have been changed. You can choose to browse the files before/after the changes or diff files.



Advanced Options

Advanced Options include word wrapping, show tabs, and performance mode. You can enable or disable these options as needed.



Set Basic Repository Information

Last updated : 2023-12-25 17:08:18

This document describes how to set basic repository information.

Open Project

1. Log in to the CODING Console and click the team domain name to go to CODING.
2. Click



in the upper-right corner to open the project list page and click a project icon to open the corresponding project.

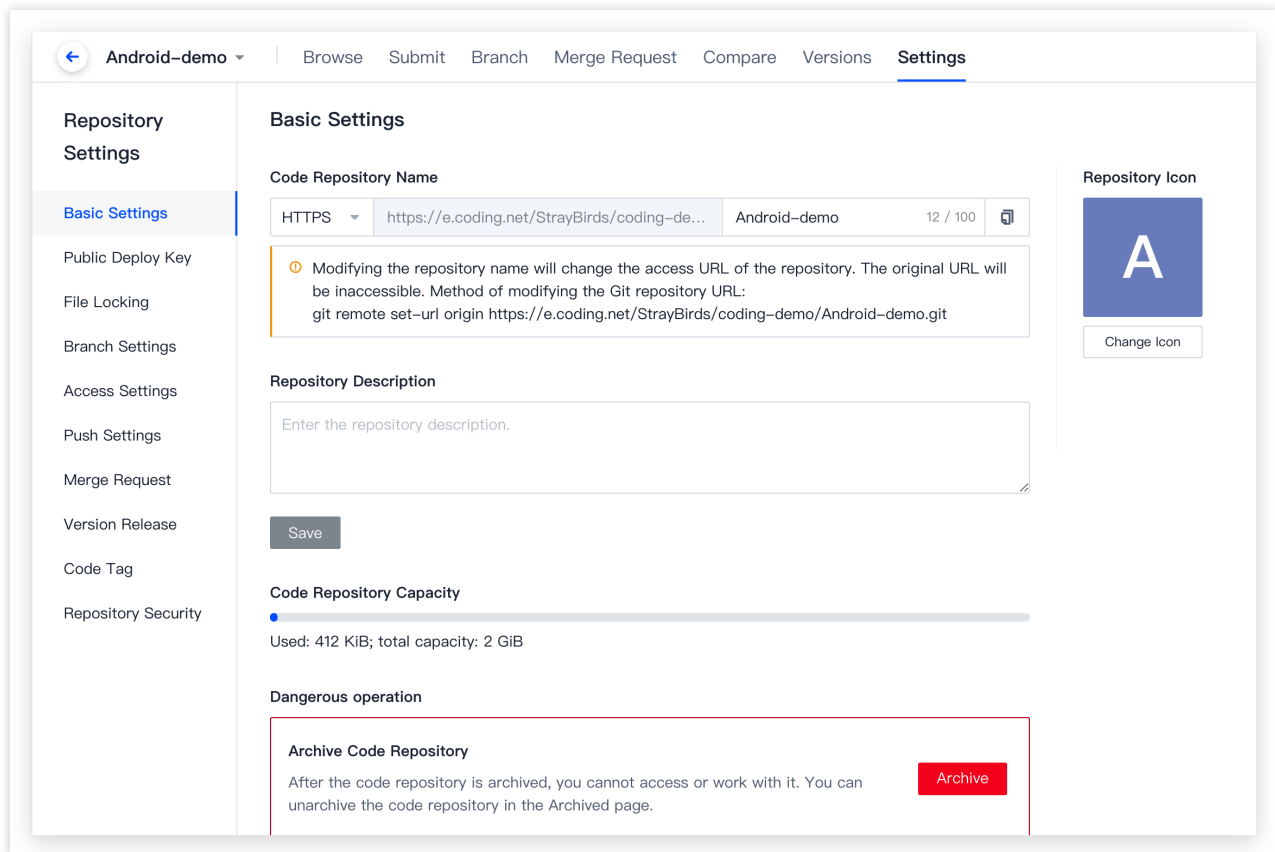
3. In the menu on the left, select **Code Repositories**.
4. If Code Repositories is not shown on the left, the project admin needs to go to **Project Settings > Projects and Members > Functions** to enable the relevant function.

Steps

1. Open a project, go to the **Code Repositories** module, and click a repository to go to its details page. In the **Settings** tab, you can configure the settings of the current repository.

Note:

Only project admins can access the Settings page of a repository.



2. On the **Basic Settings** page, you can modify the name and icon of your repository. Changing the repository name will change its access URL, so the previous URL will no longer work. After you change the name, you must match the new URL in your local repository.



```
git remote set-url origin https://e.coding.net/codingcorp/coding-help-generator/[ne
```

In addition, you can add a repository description and view the code repository capacity on this page. If necessary, you can archive, reset, or delete the repository.

Archive Repository

Last updated : 2023-12-25 17:08:18

This document describes how to archive repositories.

Open Project

1. Log in to the CODING Console and click the team domain name to go to CODING.
2. Click

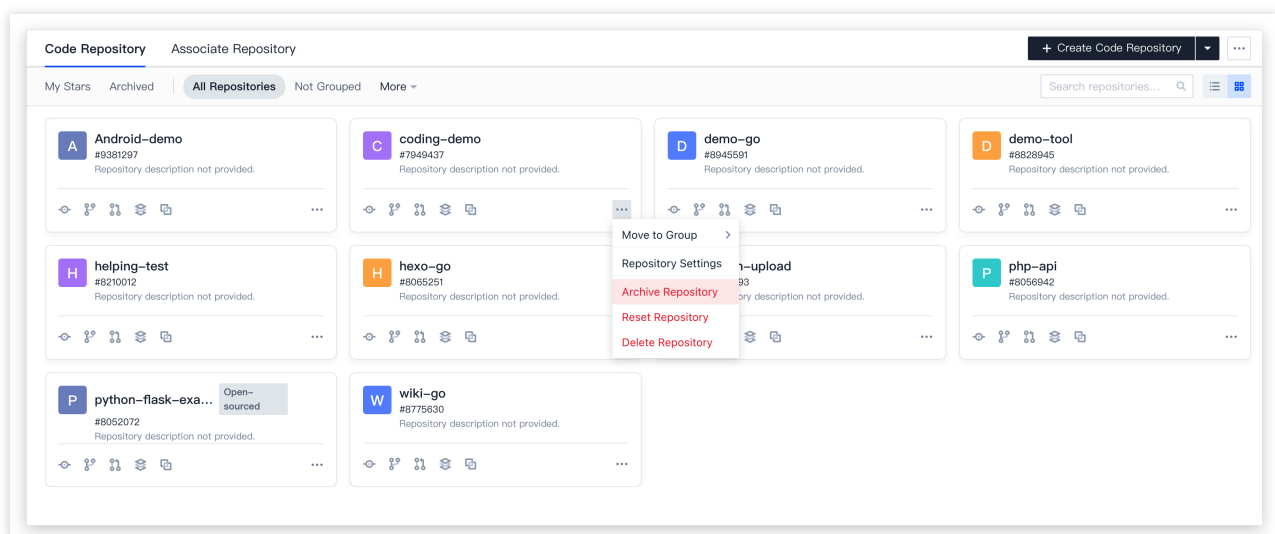


in the upper-right corner to open the project list page and click a project icon to open the corresponding project.

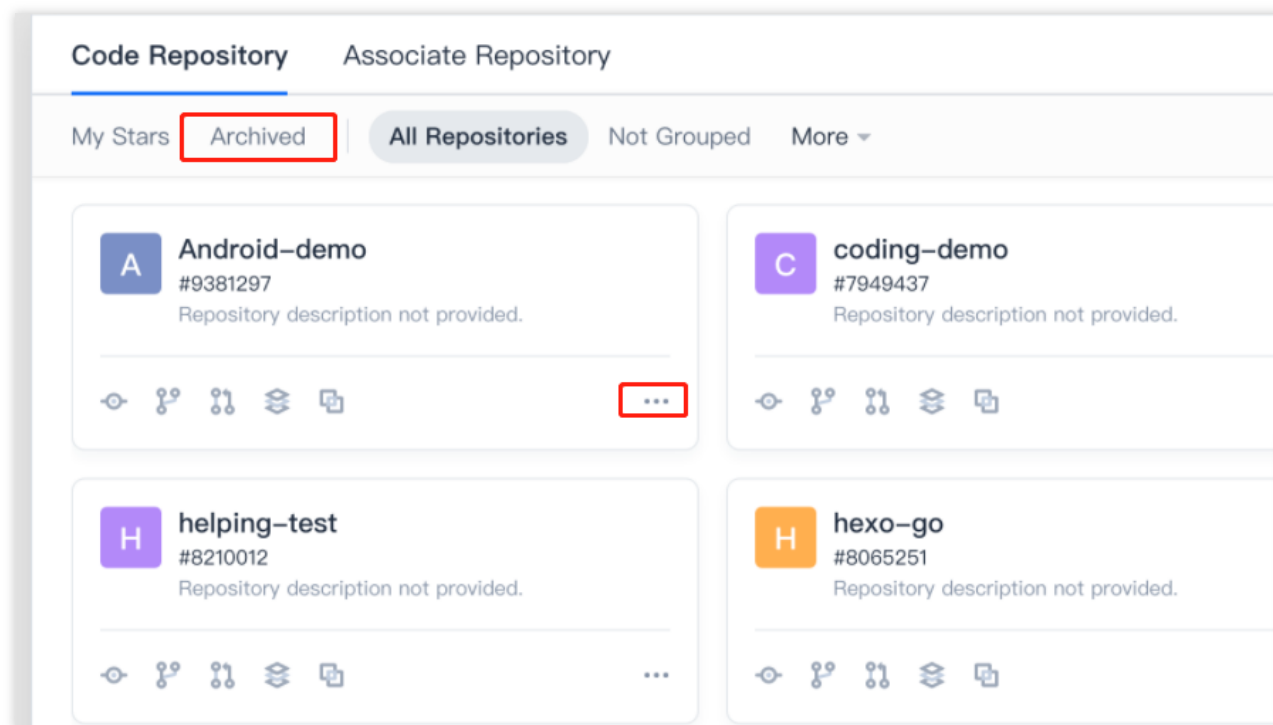
3. In the menu on the left, select **Code Repositories**.
4. If Code Repositories is not shown on the left, the project admin needs to go to **Project Settings > Projects and Members > Functions** to enable the relevant function.

Steps

1. To archive a repository, click the corresponding button in the code repository list and then follow the prompts to confirm the operation.



2. Archived repositories cannot be accessed via Git or web and can only be viewed under the **Archived** category. To restore normal access to the repository, you must unarchive the repository.



Delete and Reset Repository

Last updated : 2023-12-25 17:08:18

This document describes how to delete or reset repositories.

Open Project

1. Log in to the CODING Console and click the team domain name to go to CODING.
2. Click

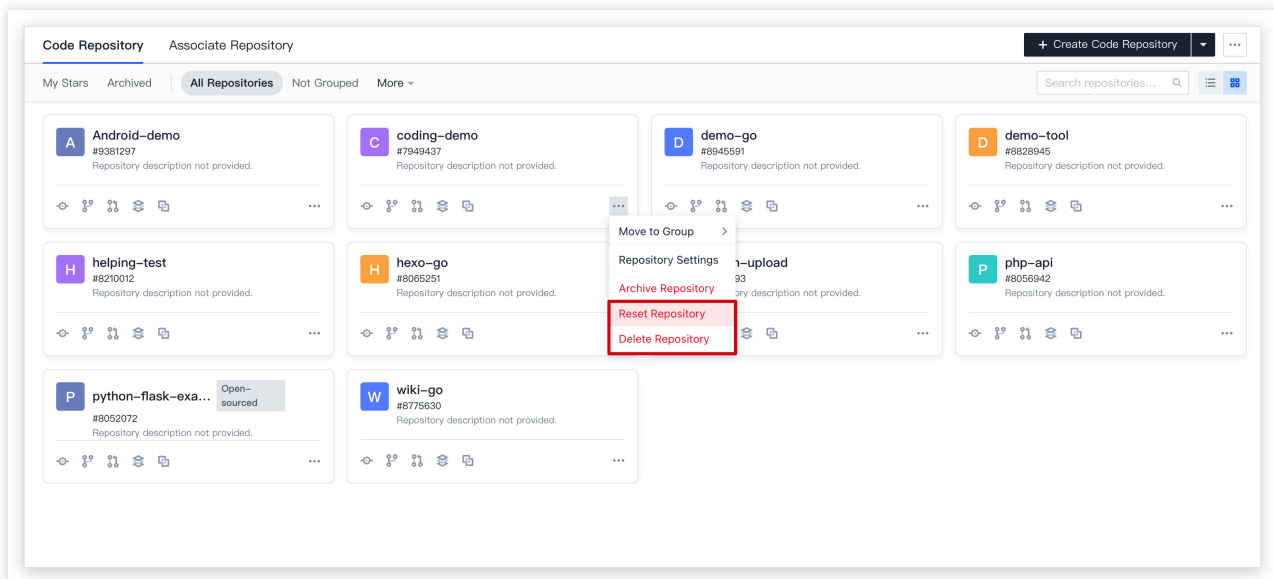


in the upper-right corner to open the project list page and click a project icon to open the corresponding project.

3. In the menu on the left, select **Code Repositories**.
4. If Code Repositories is not shown on the left, the project admin needs to go to **Project Settings > Projects and Members > Functions** to enable the relevant function.

Steps

To reset or delete a repository, click the corresponding button in the code repository list and then follow the prompts to confirm the operation. Deleted repositories are moved to the recycle bin and retained for 30 days. Before they are permanently deleted, project admins can restore them from the recycle bin. After 30 days, they will be permanently deleted and cannot be recovered.



Resetting a code repository will reset all the code in it, including code branches, merge requests, and code versions. This operation cannot be undone. The code repository will be emptied.

Deleting a code repository will delete all the code in the repository permanently, including code branches, merge requests, and code versions. This operation cannot be undone. Deleted code repositories cannot be accessed.

Restore Deleted Repository

Last updated : 2023-12-25 17:08:18

This document describes how to restore deleted repositories.

Open Project

1. Log in to the CODING Console and click the team domain name to go to CODING.
2. Click



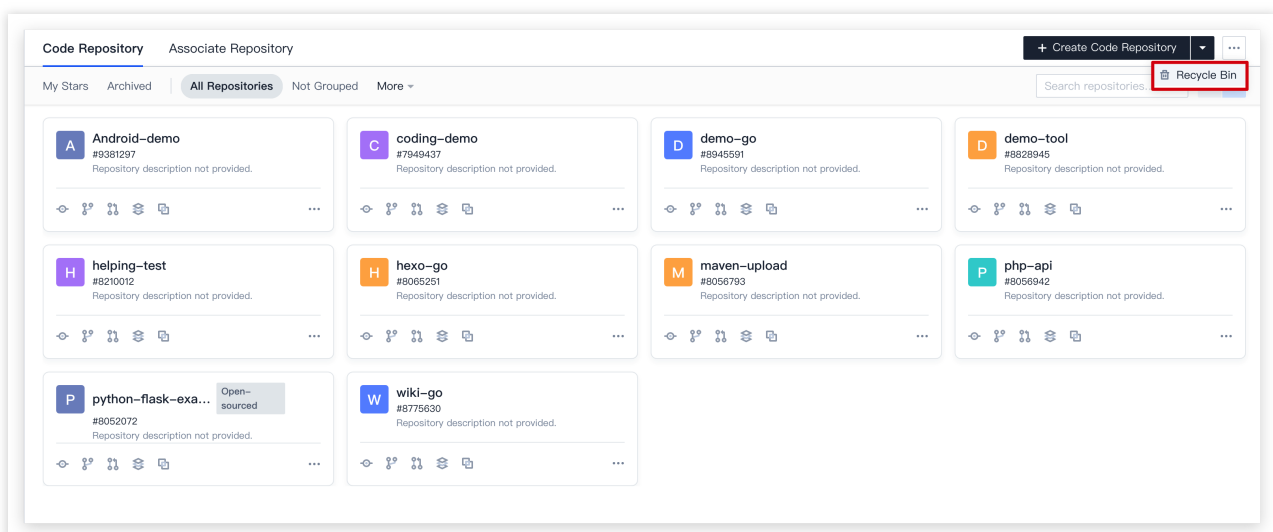
in the upper-right corner to open the project list page and click a project icon to open the corresponding project.

3. In the menu on the left, select **Code Repositories**.
4. If Code Repositories is not shown on the left, the project admin needs to go to **Project Settings > Projects and Members > Functions** to enable the relevant function.

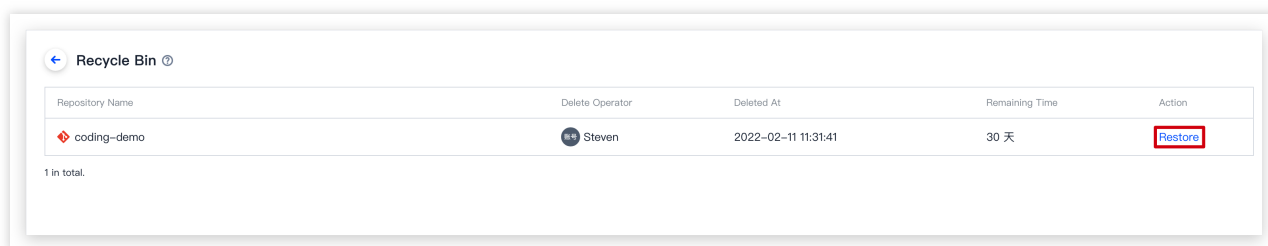
Steps

Project admins can restore deleted repositories from the recycle bin within 30 days.

1. On the **Code Repositories** page, hover over the More Actions button in the top-right corner and click Recycle Bin.



2. In the recycle bin, select the repository to restore and click **Restore** to restore the repository.



Manage Code Repository Cards

Last updated : 2023-12-25 17:08:18

This document describes how to manage code repository cards.

Open Project

1. Log in to the CODING Console and click the team domain name to go to CODING.
2. Click



in the upper-right corner to open the project list page and click a project icon to open the corresponding project.

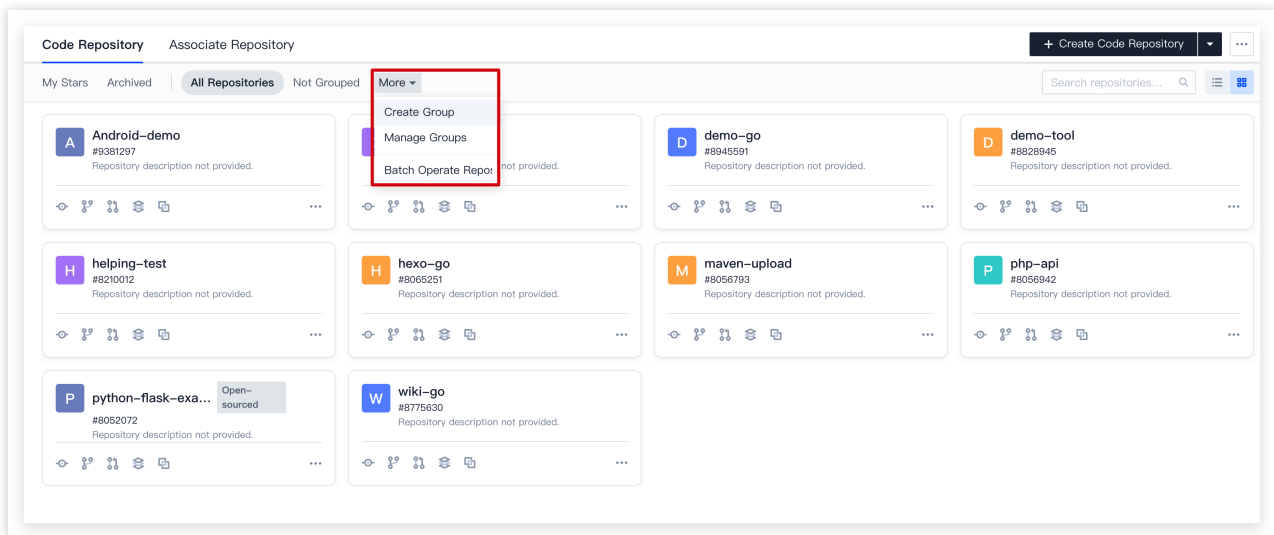
3. In the menu on the left, select **Code Repositories**.

4. If Code Repositories is not shown on the left, the project admin needs to go to **Project Settings > Projects and Members > Functions** to enable the relevant function.

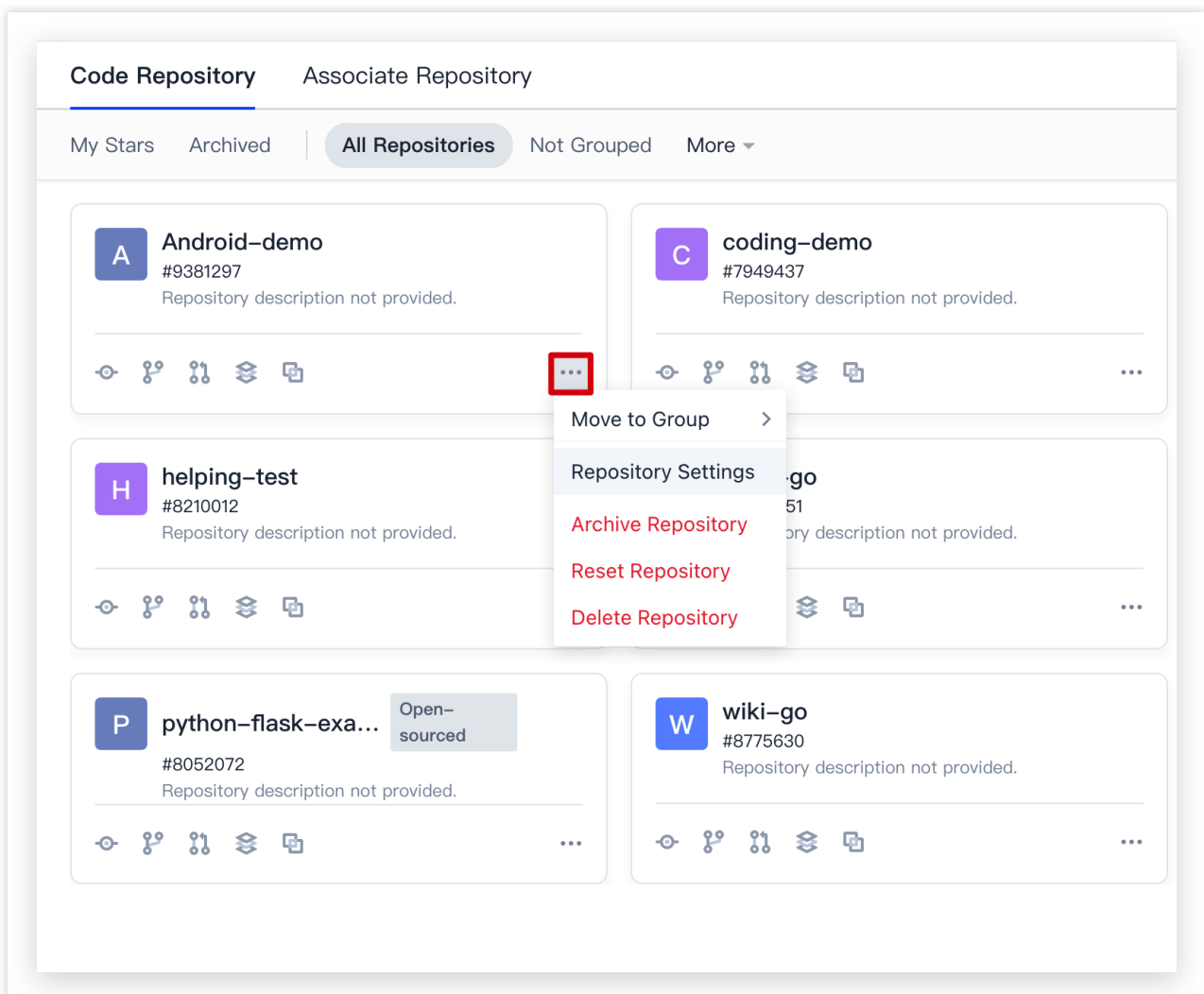
By default, each code repository is displayed as a card. You can use the [quick actions](#) on cards to group and sort repositories.

Group and Sort

On the **Code Repositories** page, you can create repository groups and add multiple repositories to them in batch. Go to **Manage Groups** to adjust the order of repository groups.

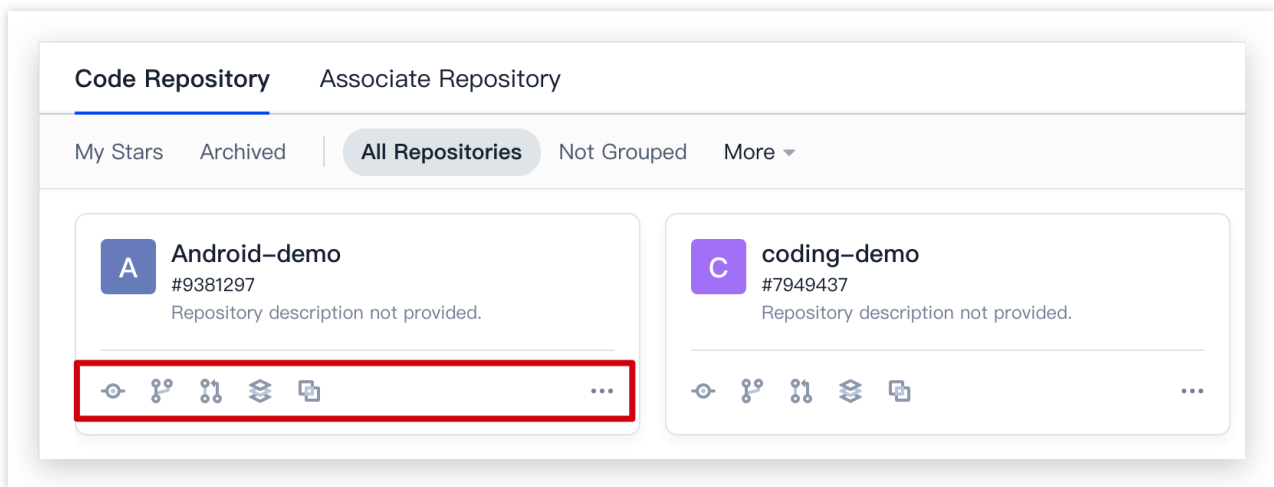


You can add a repository to a group via the More Actions button.

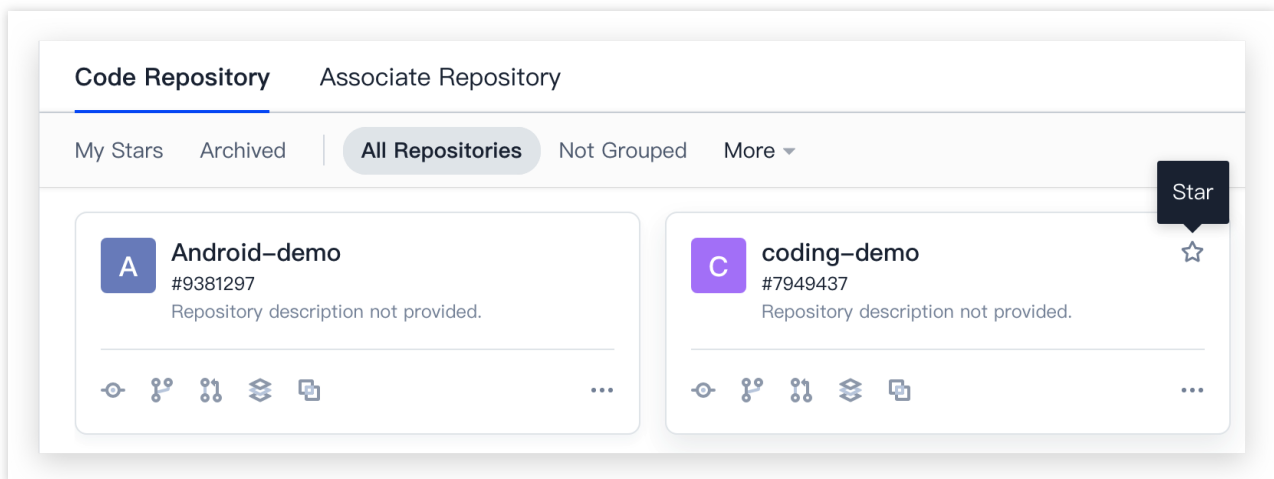


Repository card quick actions

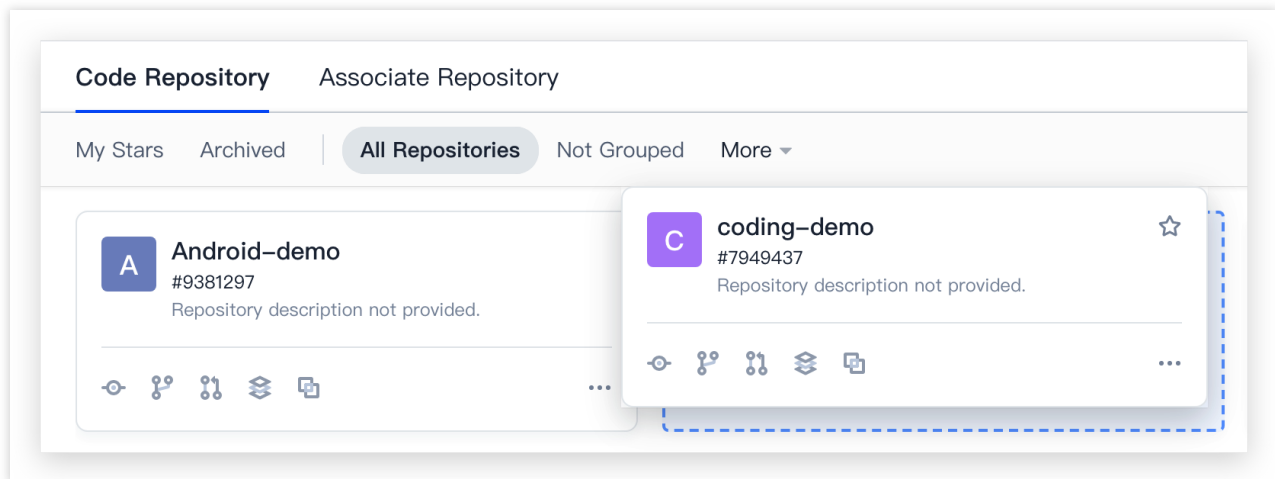
Each repository card has built-in quick actions that allow you to quickly perform operations as shown below:



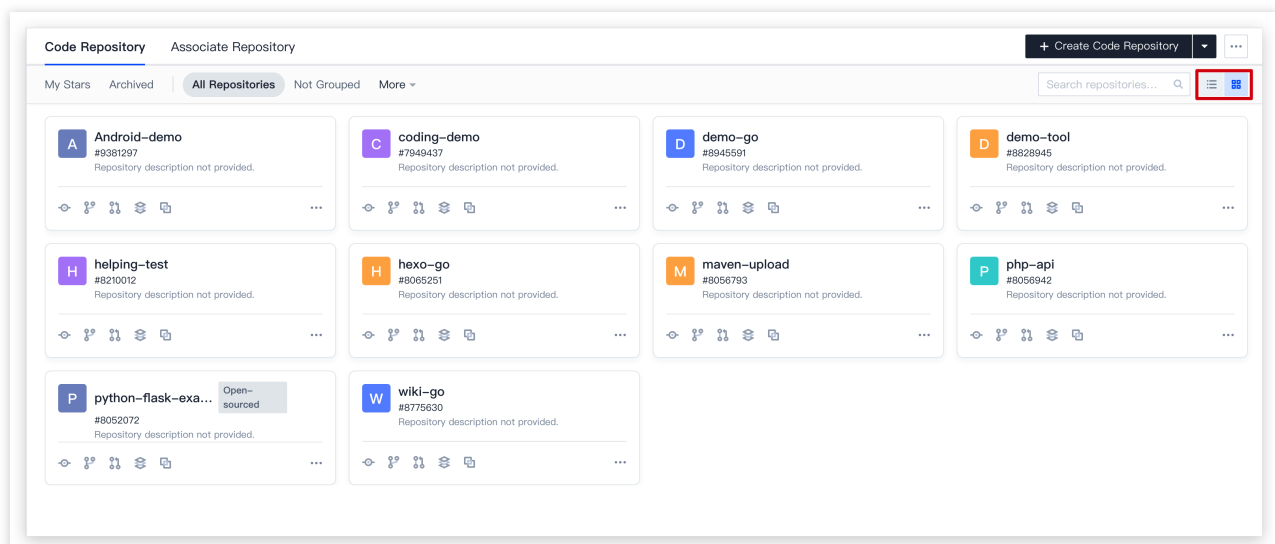
Hover over a card and click the star button in the top-right to star the repository.



Drag the cards to change their order.



Click the switch display mode button in the top-right to switch between list and card display.



Manage Repositories Via Local Command Lines

Last updated : 2023-12-25 17:08:18

This document describes how to use the local command line to manage repositories.

Open Project

1. Log in to the CODING Console and click the team domain name to go to CODING.
2. Click

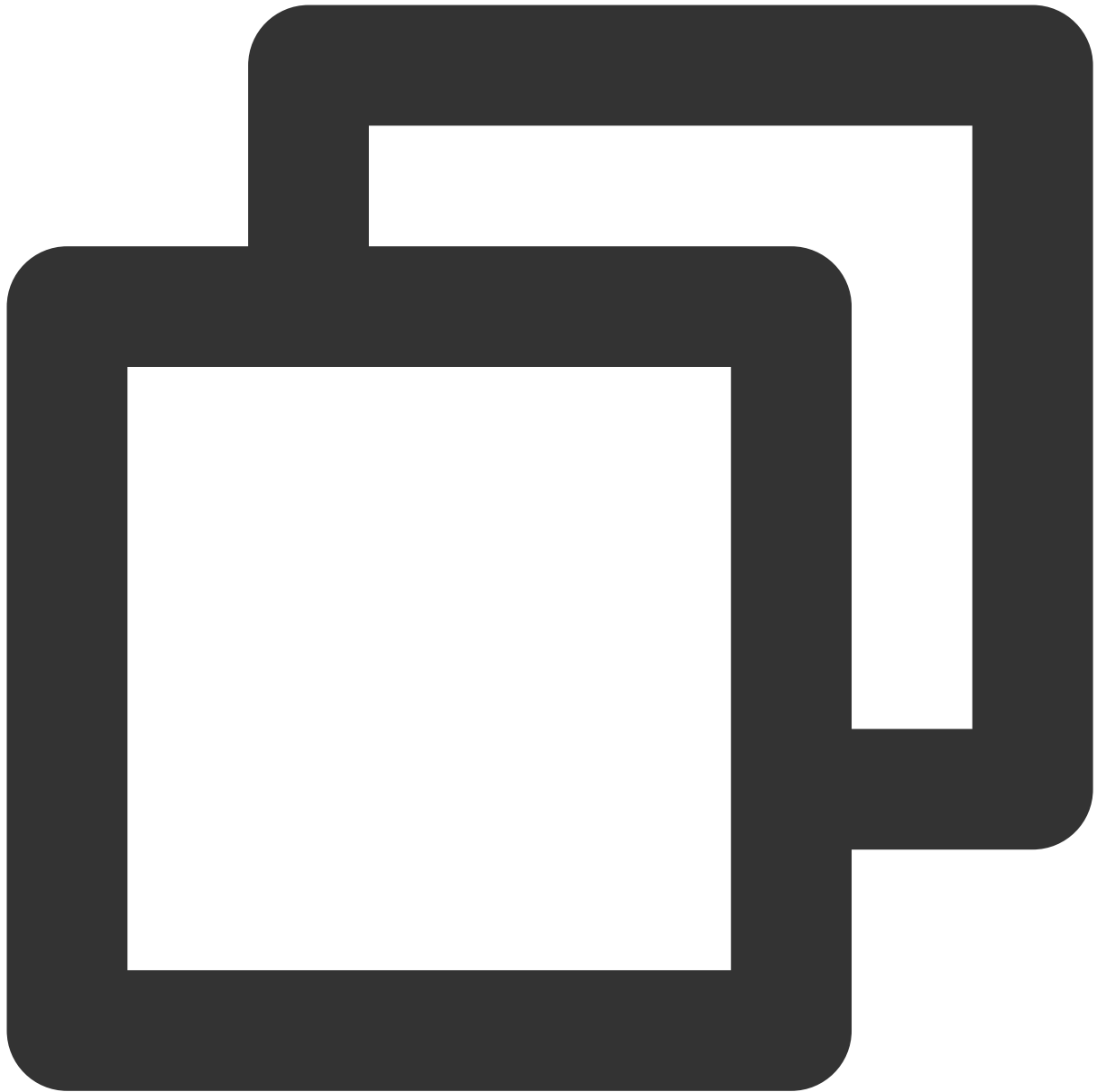


in the upper-right corner to open the project list page and click a project icon to open the corresponding project.

3. In the menu on the left, select **Code Repositories**.
4. If Code Repositories is not shown on the left, the project admin needs to go to **Project Settings > Projects and Members > Functions** to enable the relevant function.

Get Data from Remote Repository

You can use the `git clone` command to clone a remote repository to your local device and automatically associate it.



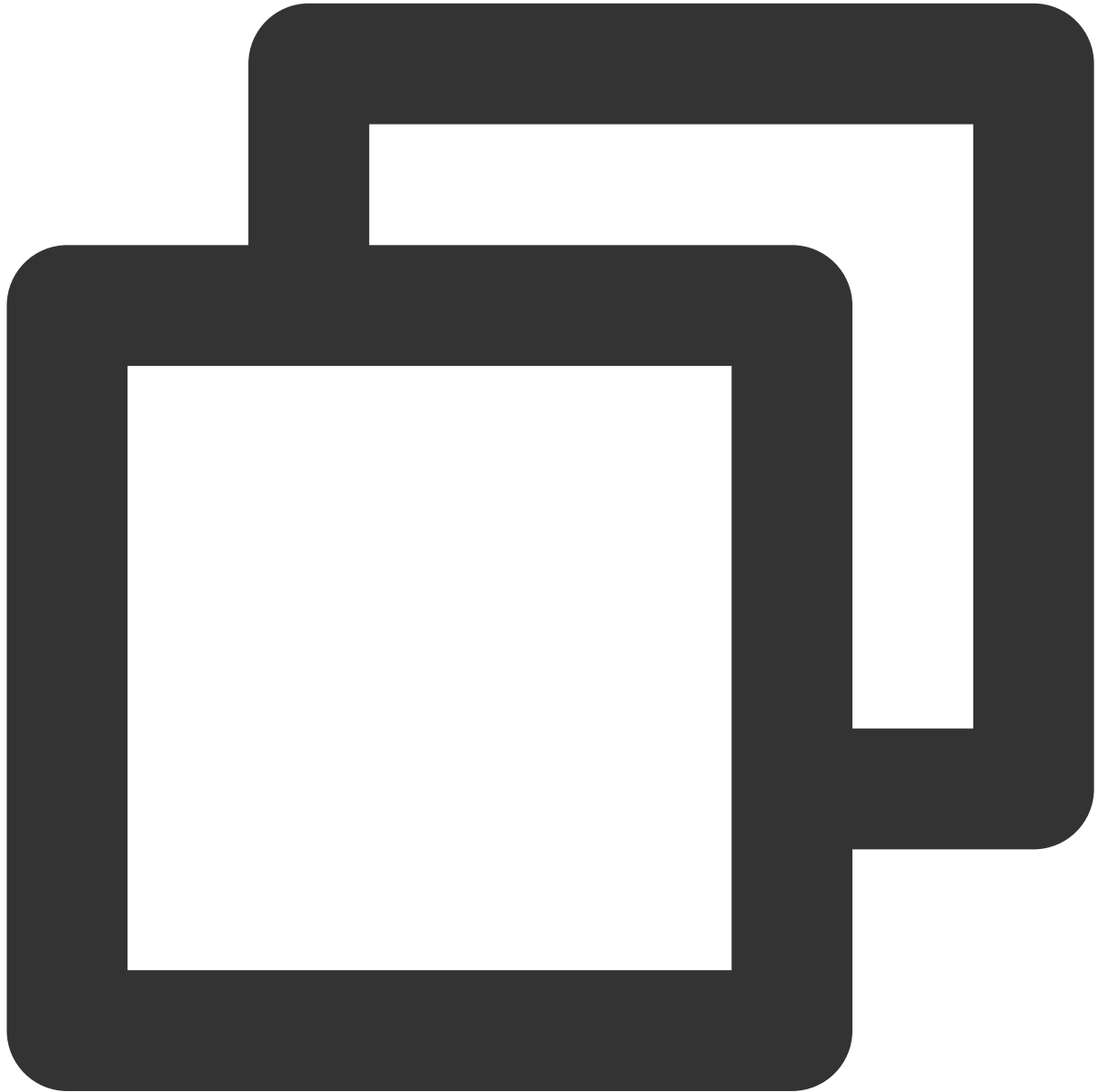
```
git clone [remote-name]
```

Push Data to Remote Repository

Use `git push [remote-name] [branch-name]` to push data from a local repository to a remote repository. For example, `git push learn-git master` will push data from the local repository to the "master" branch of the remote repository.

Rename Remote Repository

Use the `git remote rename [old-name] [new-name]` command to modify the local nickname of a remote repository. For example, to change the repository name from `learn-git` to `origin`, run the following code:

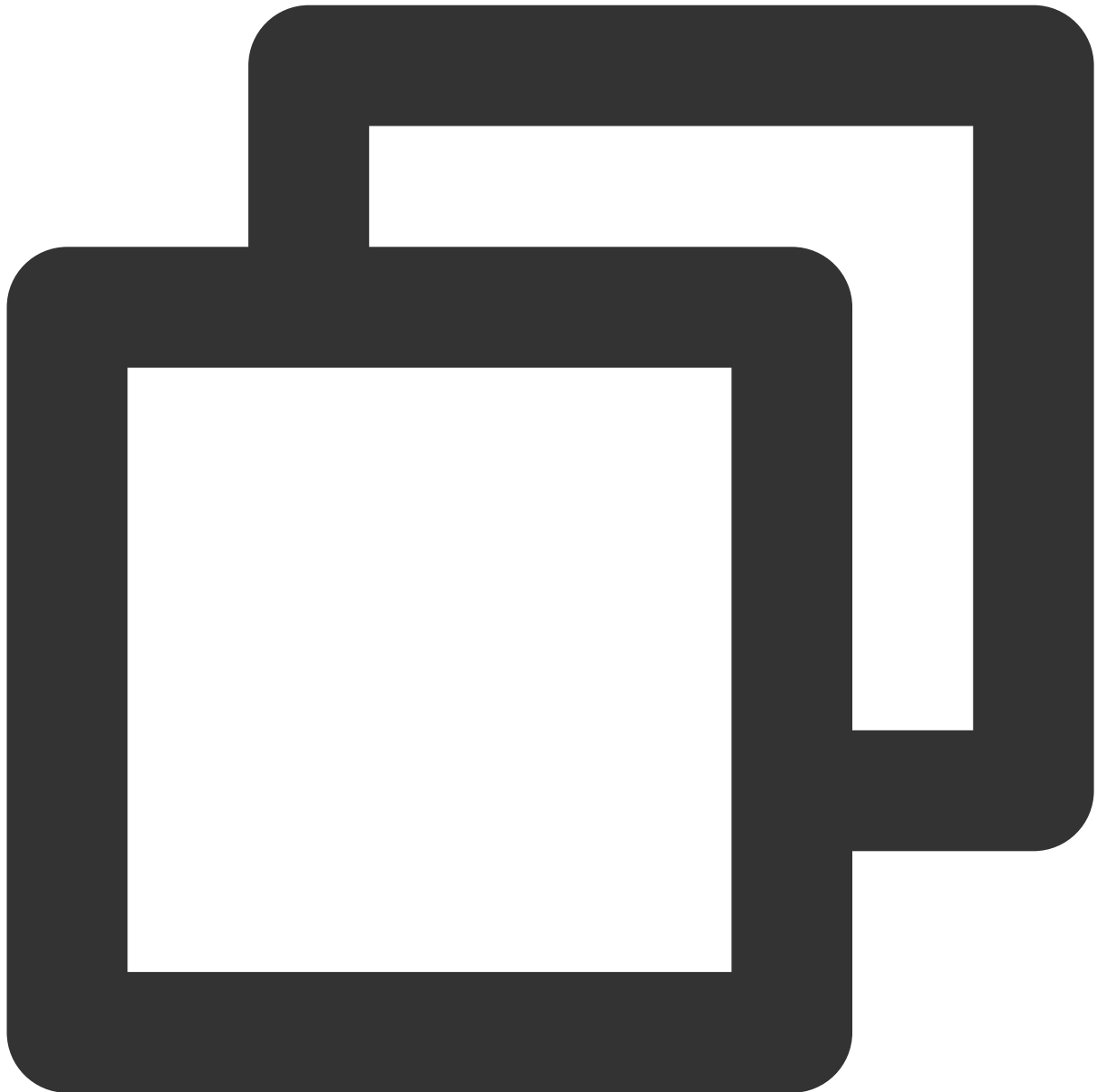


```
git remote rename learn-git origin
```

After you rename the remote repository, remember to use the new name when you need to specify the repository name in a Git command.

Disassociate Remote Repository

To disassociate the remote repository "origin", run the following command:



```
git remote rm origin
```

Note:

This command disassociates the remote repository from the local repository, and does not delete the remote repository data. For more information about Git commands, see [Common Git Commands](#).

SVN Repository Usage

Create SVN Repository

Last updated : 2023-12-25 17:08:18

This document describes how to create an SVN repository.

Open Project

1. Log in to the CODING Console and click the team domain name to go to CODING.
2. Click



in the upper-right corner to open the project list page and click a project icon to open the corresponding project.

3. In the menu on the left, select **Code Repositories**.
4. If Code Repositories is not shown on the left, the project admin needs to go to **Project Settings > Projects and Members > Functions** to enable the relevant function.

Steps

CODING supports native SVN repositories. On the client, use SVN+SSH protocol to connect to the CODING server. All data is transmitted through SSH encrypted channels.

1. Open a project and click **Code Repositories** on the left navigation bar to open the Code Repository Management page.
2. In the upper-right corner of the page, click **Create Code Repository**, and then select **SVN Repository** as the repository type.

[←](#) Create Code Repository | [Create from Scratch](#) [Create from Template](#) [Already have a repository on another website? Click to import](#)

Repository Type *

- SVN Rep...
- Git Repository
- SVN Repository

Repository Name *

The repository name should be a combination of letters, numbers, un 0/100

on

by description.

Quickly Initialize Repository

☐ Create Recommended SVN Repository Layout (tags, branches, and trunk) [SVN Repository Instructions](#)

[Create](#) [Cancel](#)

3. If you select `Create Recommended SVN Repository Layout` , the system will automatically create the `tags` , `branches` , and `trunk` directories. This is the recommended directory layout for most SVN repositories. After completing warehouse initialization, you can view SVN repository content in the Browse Code interface.

[←](#) SVN-Test | [Browse](#) [Submit](#) [Settings](#) [Create Code Repository](#)

SVN-Test

head /

Checkout ▾

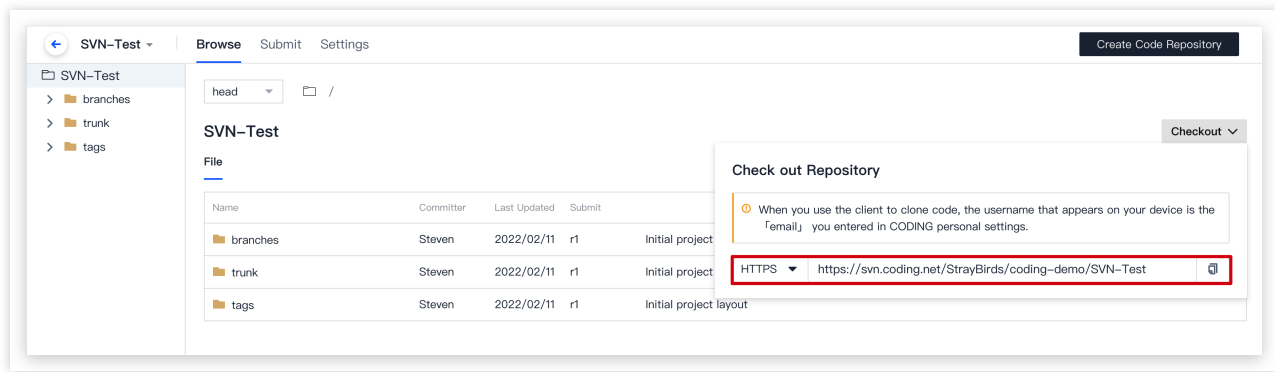
File

Name	Committer	Last Updated	Submit	
branches	Steven	2022/02/11	r1	Initial project layout
trunk	Steven	2022/02/11	r1	Initial project layout
tags	Steven	2022/02/11	r1	Initial project layout

The **Browse Code** interface displays the SVN URL of the repository:



```
svn://subversion.e.coding.net/StrayBirds/svn
```


**Note:**

Currently, you must create SVN repositories in a project. You cannot create an SVN repository in a Git repository.

Access SVN Repository

Last updated : 2023-12-25 17:08:18

The SVN repository service currently supports most mainstream SVN clients. We recommend you use the latest stable version of the client.

Open Project

1. Log in to the CODING Console and click the team domain name to go to CODING.
2. Click



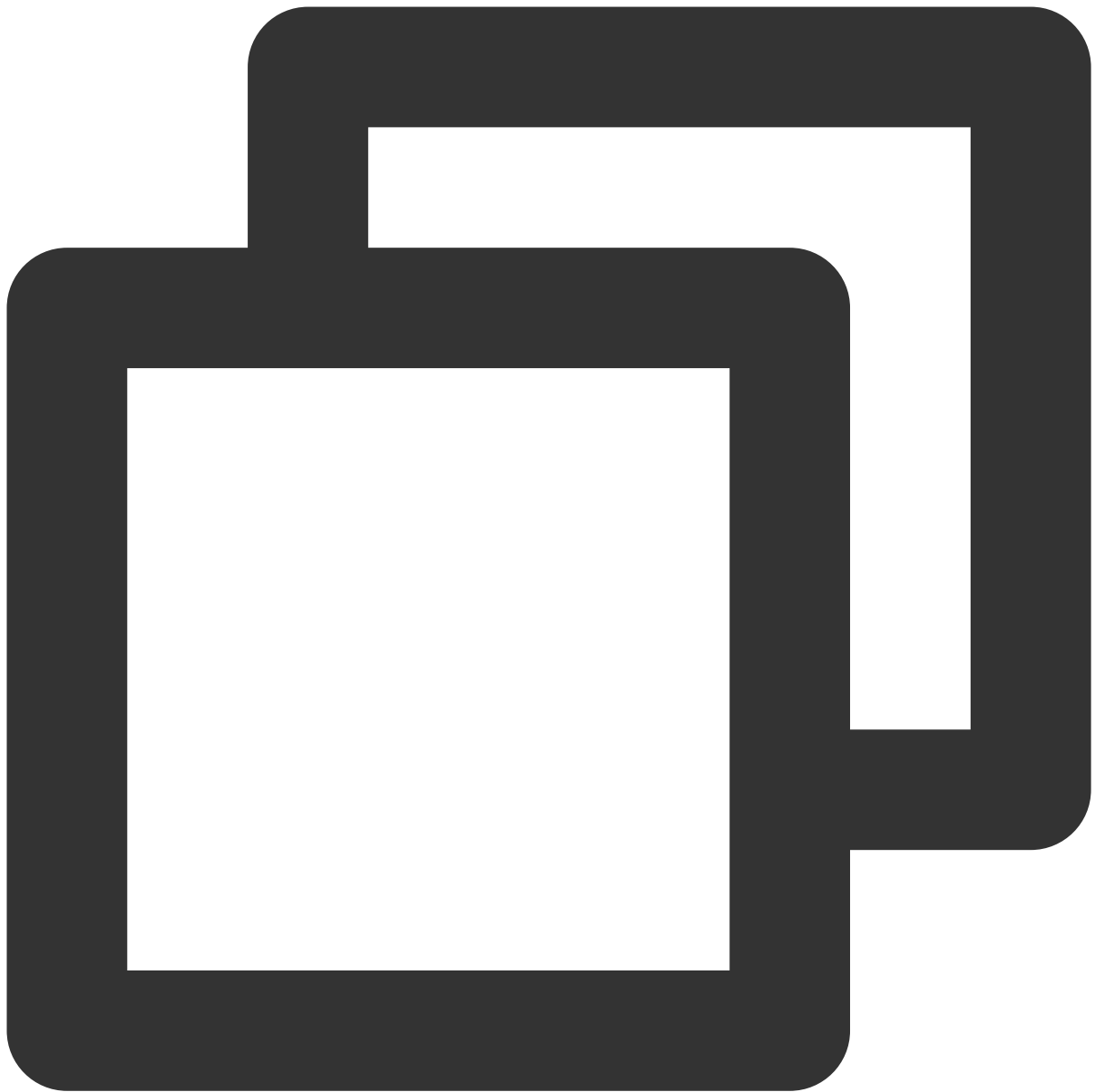
in the upper-right corner to open the project list page and click a project icon to open the corresponding project.

3. In the menu on the left, select **Code Repositories**.
4. If Code Repositories is not shown on the left, the project admin needs to go to **Project Settings > Projects and Members > Functions** to enable the relevant function.

Mac Environment

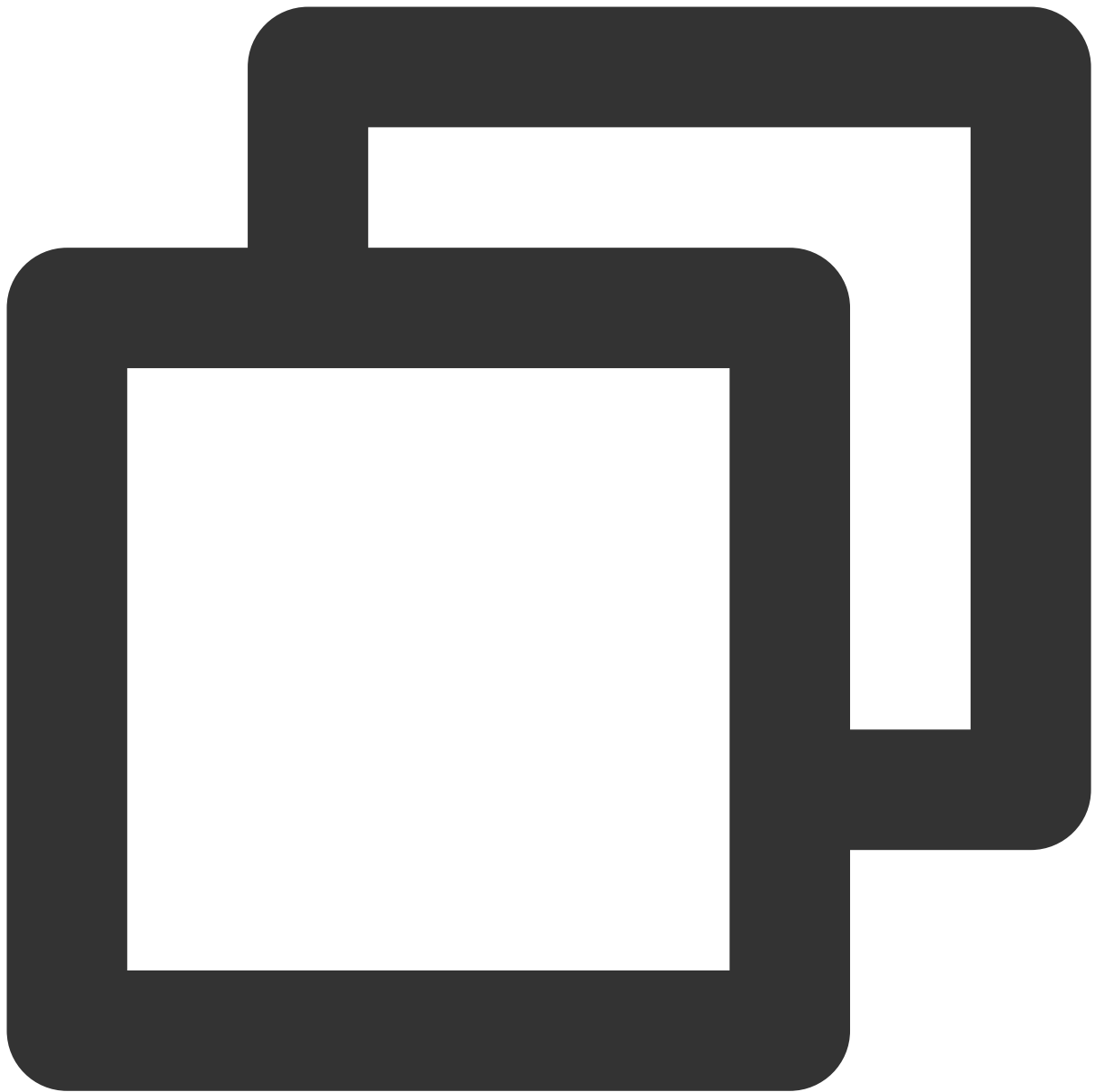
In a Mac environment, you can use Homebrew to install the SVN client.

1. Run the following command to install Homebrew:



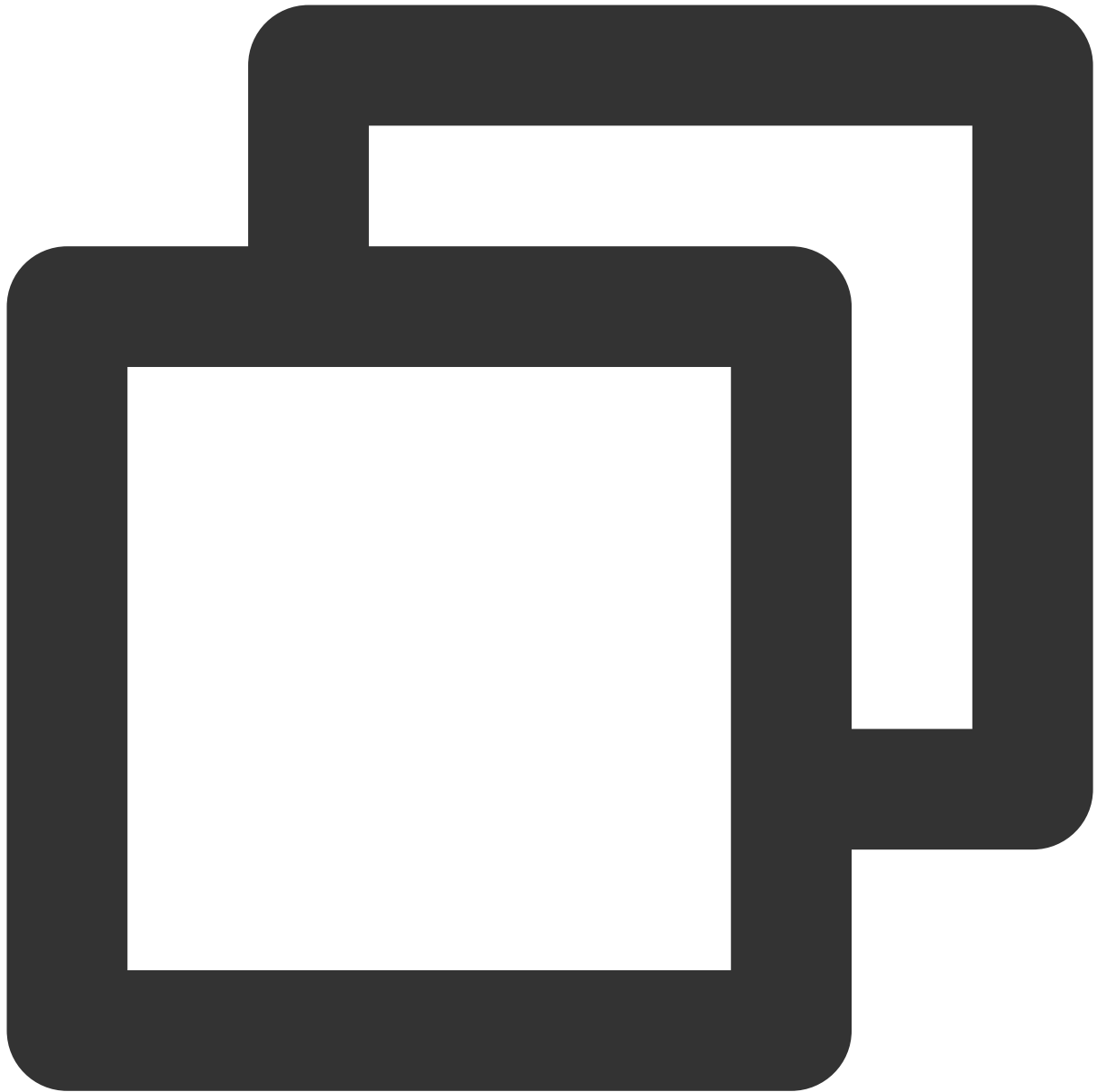
```
/usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/m
```

2. After you install Homebrew, input the following command in your terminal to install SVN:



```
brew install subversion
```

3. Run the `svn --version` command to verify that SVN has been correctly installed:

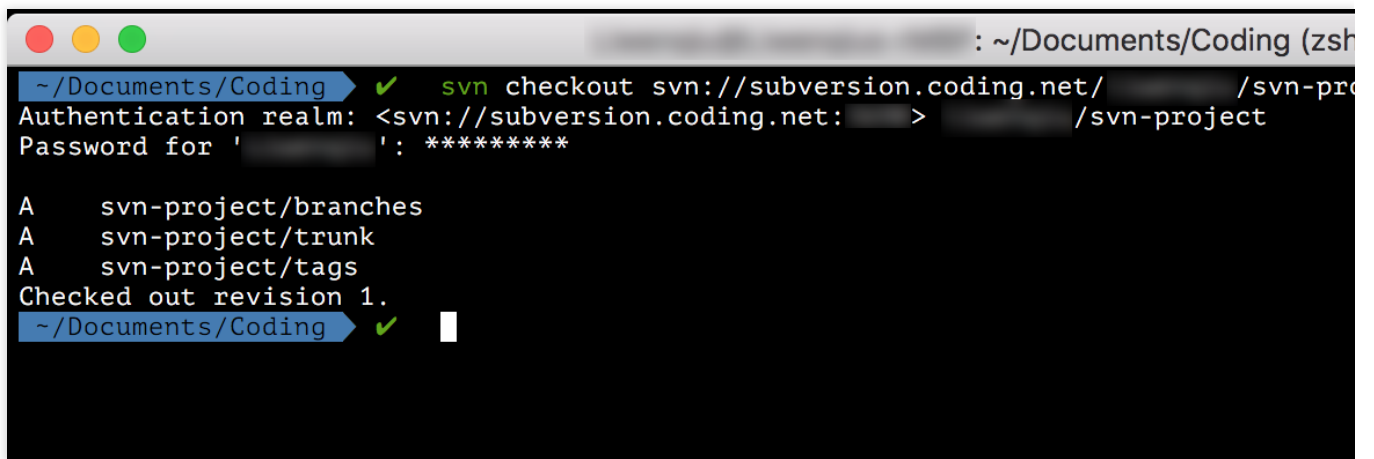


```
svn, version 1.9.7 (r1800392)
compiled Feb 28 2018, 15:54:50 on x86_64-apple-darwin17.3.0
Copyright (C) 2017 The Apache Software Foundation.
This software consists of contributions made by many people;
see the NOTICE file for more information.
Subversion is open source software, see http://subversion.apache.org/
The following repository access (RA) modules are available:

* ra_svn : Module for accessing a repository using the svn network protocol.
- with Cyrus SASL authentication
- handles 'svn' scheme
```

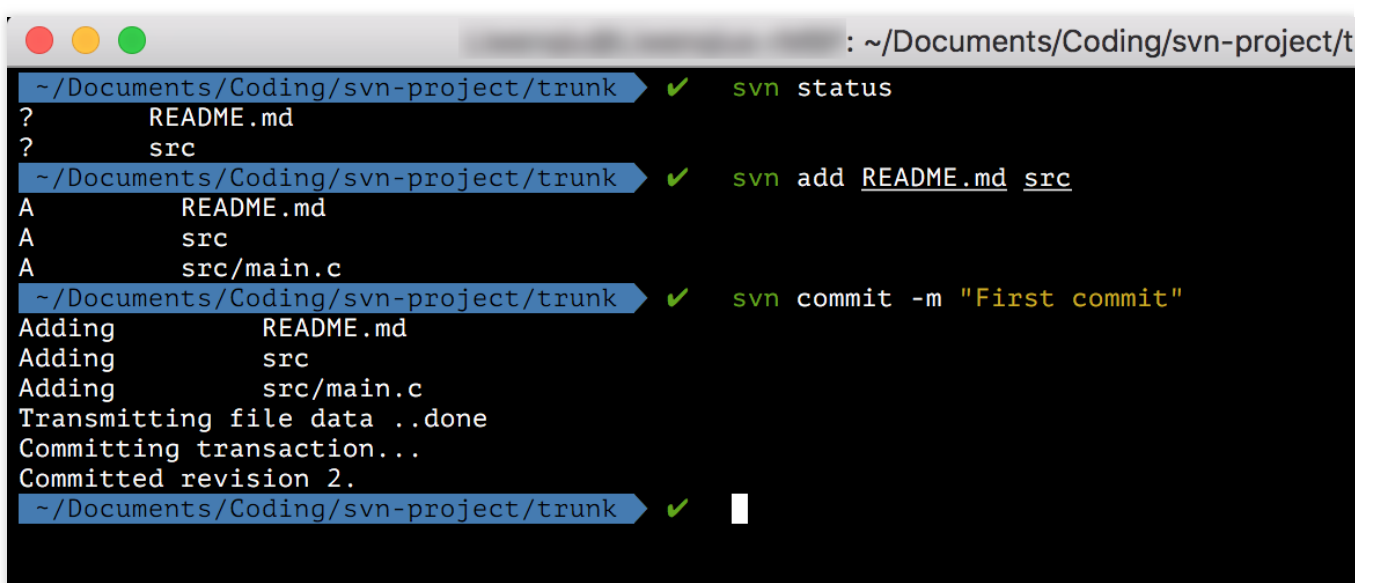
```
* ra_local : Module for accessing a repository on local disk.
- handles 'file' scheme
* ra_serf : Module for accessing a repository via WebDAV protocol using serf.
- using serf 1.3.9 (compiled with 1.3.9)
- handles 'http' scheme
- handles 'https' scheme
The following authentication credential caches are available:
* Plaintext cache in /Users/Liwenqiu/.subversion
* Mac OS X Keychain
```

4. Run the command `svn checkout svn://subversion.e.coding.net/example/example-project` (replacing the URL with your SVN repository URL) to check out the SVN repository:



```
~/Documents/Coding : ~/Documents/Coding (zsh)
~/Documents/Coding ✓ svn checkout svn://subversion.coding.net/ /svn-pro
Authentication realm: <svn://subversion.coding.net: > /svn-project
Password for ' ': *****
A   svn-project/branches
A   svn-project/trunk
A   svn-project/tags
Checked out revision 1.
~/Documents/Coding ✓
```

5. Then, you can use the `add` and `commit` commands to add content to the repository:



```
~/Documents/Coding/svn-project/trunk : ~/Documents/Coding/svn-project/t
~/Documents/Coding/svn-project/trunk ✓ svn status
?   README.md
?   src
~/Documents/Coding/svn-project/trunk ✓ svn add README.md src
A   README.md
A   src
A   src/main.c
~/Documents/Coding/svn-project/trunk ✓ svn commit -m "First commit"
Adding      README.md
Adding      src
Adding      src/main.c
Transmitting file data ..done
Committing transaction...
Committed revision 2.
~/Documents/Coding/svn-project/trunk ✓
```


6. In addition to using SVN protocol, you can use `svn+ssh` protocol to access the repository, as shown below:


```
~/Documents/Coding : ~/Documents/Coding (zsh)
~/Documents/Coding ✓ svn checkout svn+ssh://subversion.coding.net/ /
A   svn-project/trunk
A   svn-project/trunk/src
A   svn-project/trunk/src/main.c
A   svn-project/trunk/README.md
A   svn-project/branches
A   svn-project/tags
Checked out revision 3.
~/Documents/Coding ✓
```


Cornerstone Tool

You can use SVN repositories through Cornerstone.

1. Open Cornerstone and click Add Repository to add an SVN repository reference (replacing the URL with your SVN repository URL):

Cloud Service

File Repository

HTTP Server

SERVER LOCATION

Enter the location of the server. The repository in the source list.

Tunnel:

None

Server:

subversion.coding.net

Port:

Path:

svn-project

Path:

svn:// @subversion.coding.n

Title:

ACCOUNT INFORMATION

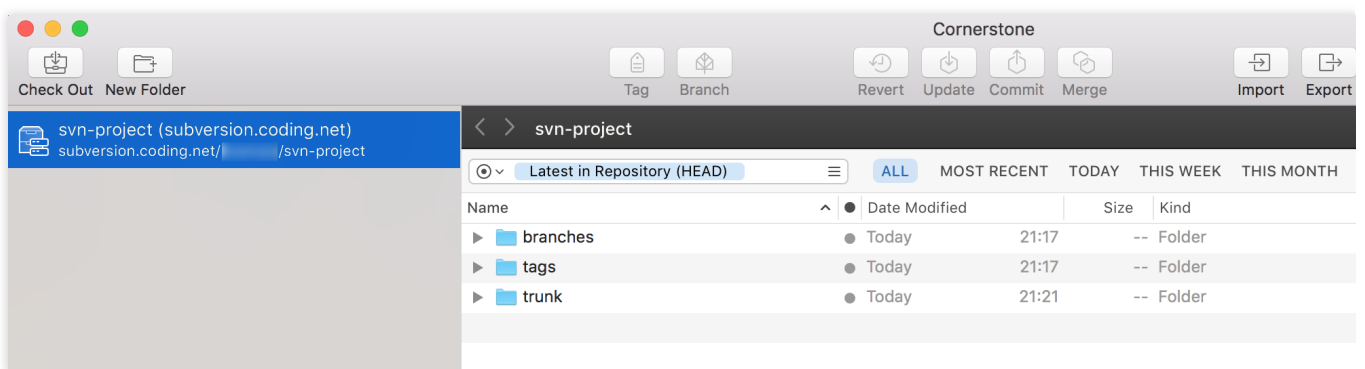
Specify the account you use to access the repository. Leave the fields blank when using a tunnel that does not require an account (such as SSH with private key).

Name:

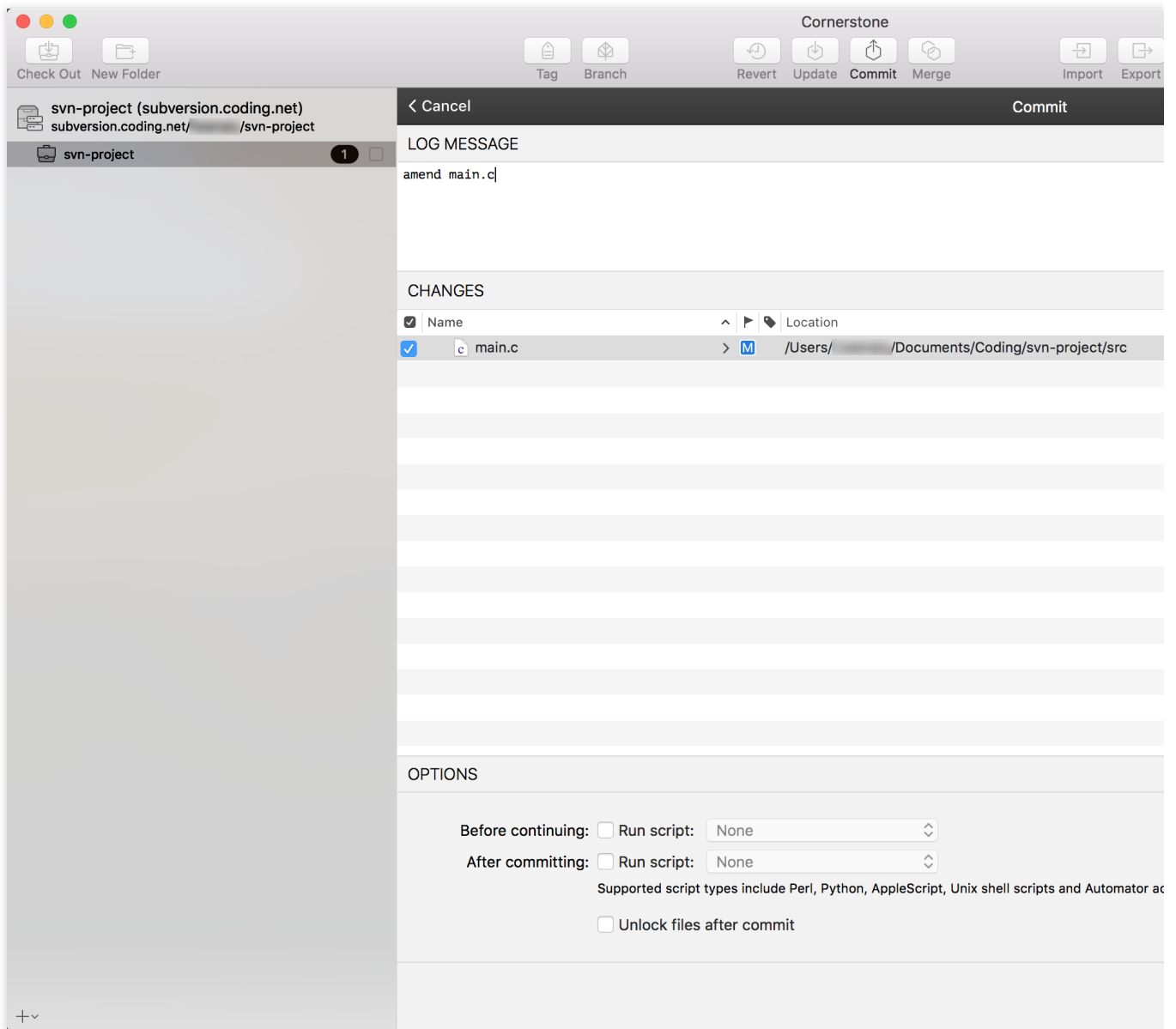
Password:

☒ Save name and password in keychain

Then, you can view the repository content.



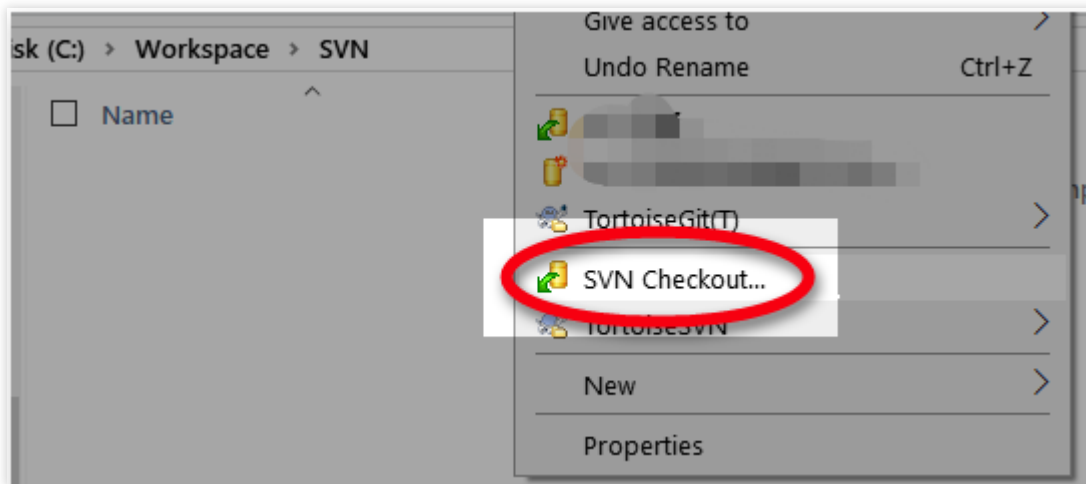
2. Use **Check Out** to check out the repository, edit the files, and use **Commit** to commit the changes, as shown below:



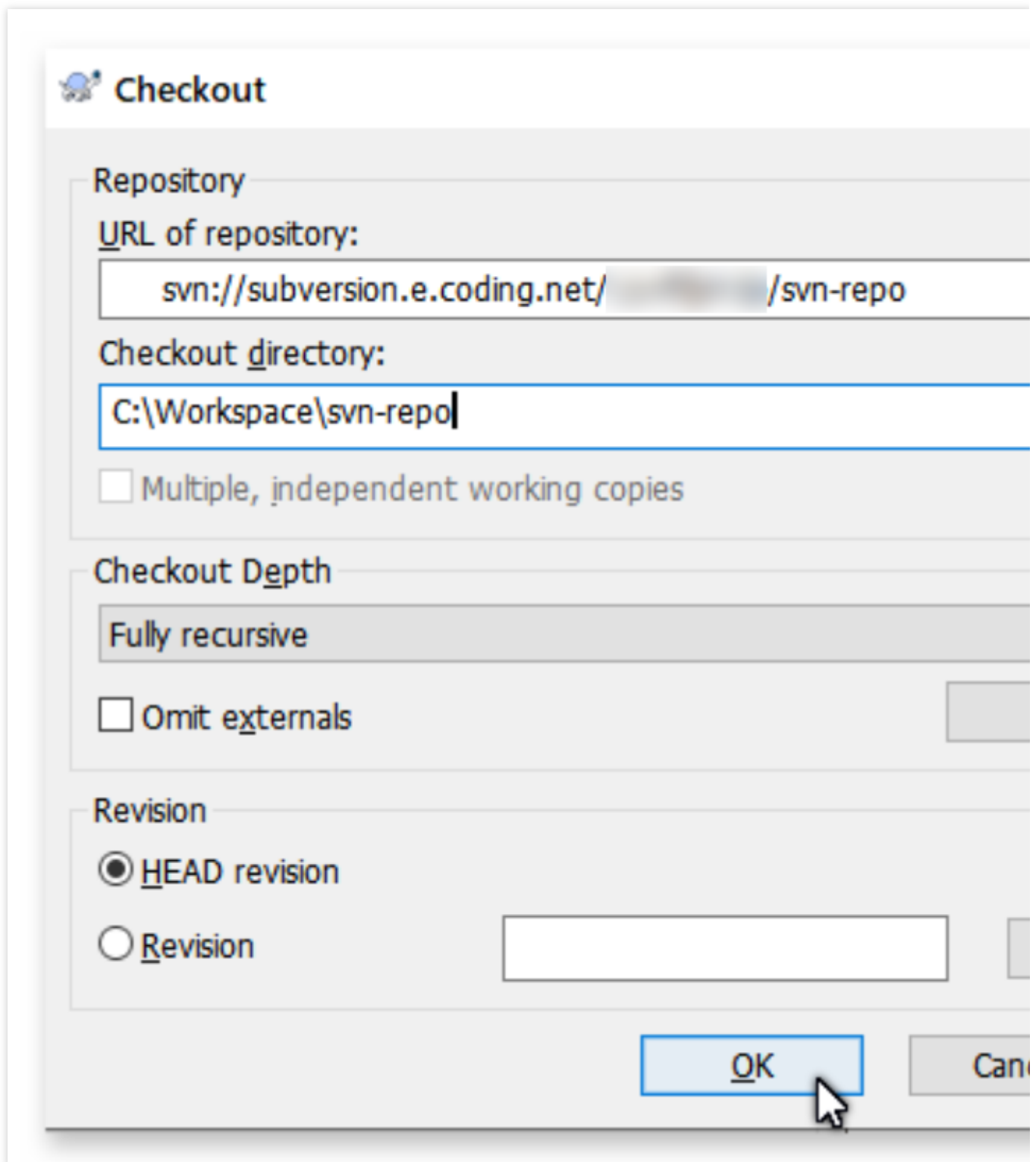
Windows Environment

In Windows, we recommend you use TortoiseSVN.

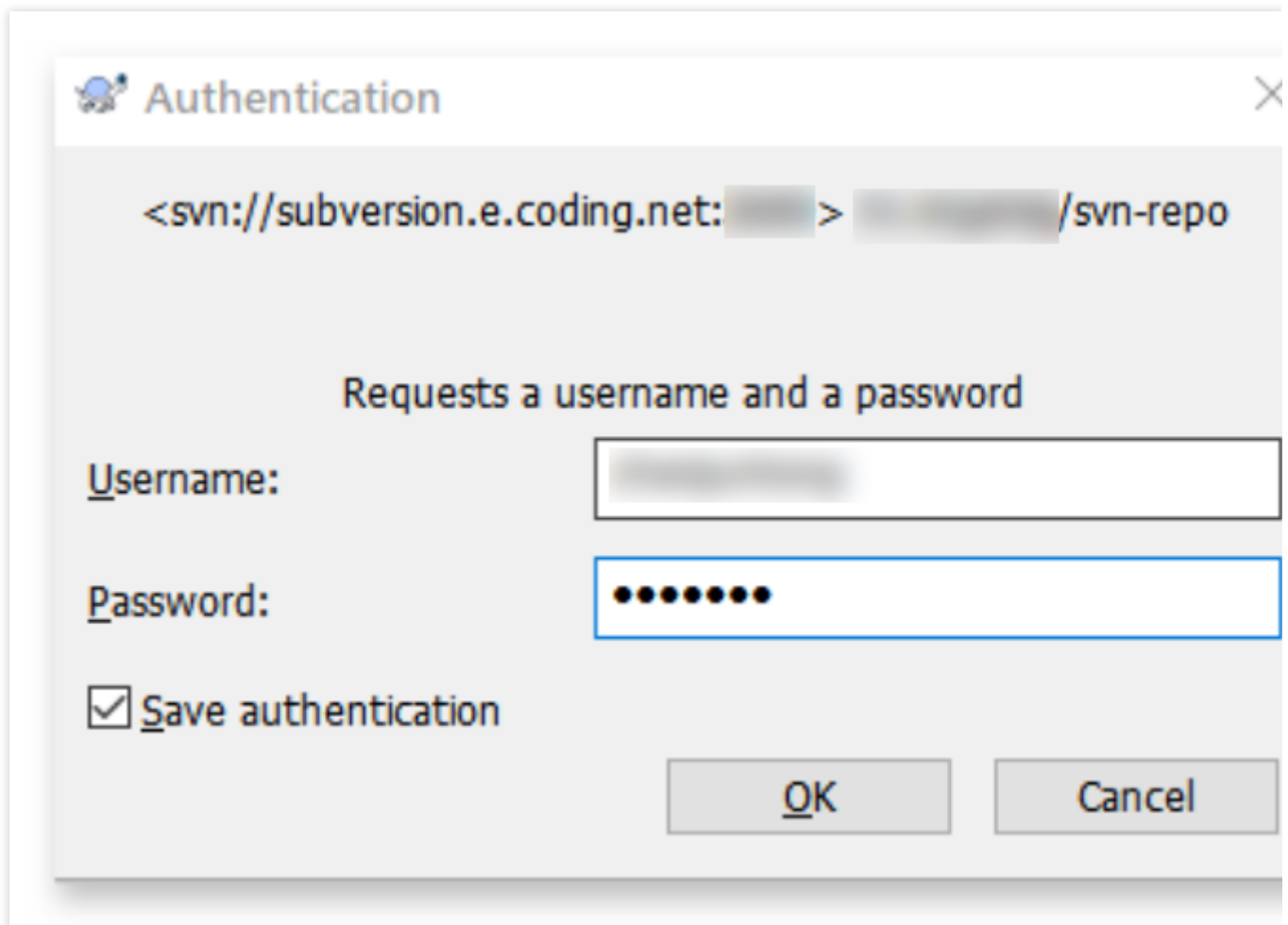
1. After [downloading](#) and installing the tool, right-click on any file directory.



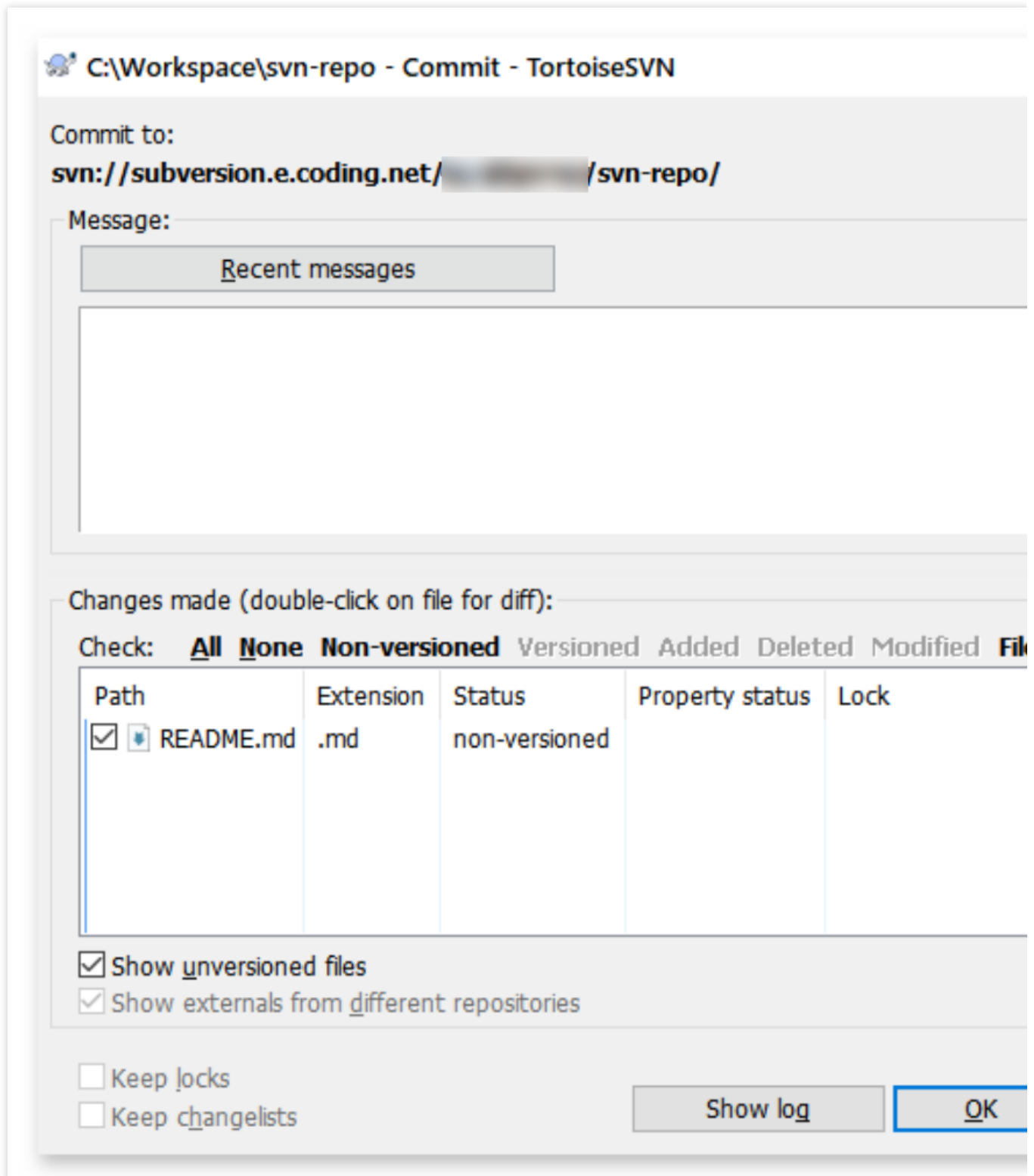
Select `Checkout` to check out the SVN repository (replace the URL with your SVN repository URL).



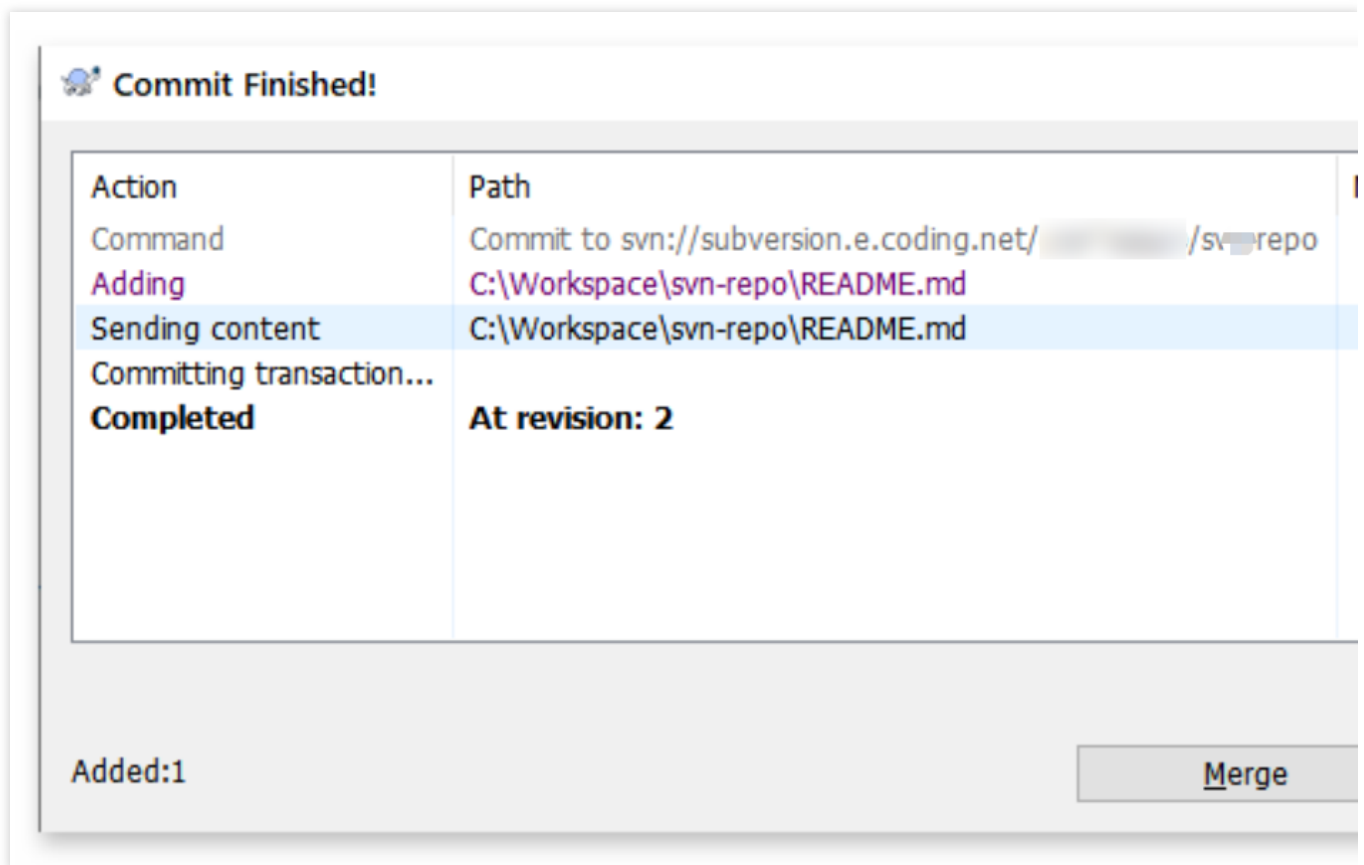
2. The first time you use `Checkout` , you must enter your username and password. Select `Save authentication` to save the authentication information so you will not have to enter your password next time.



3. Open the checked-out folder and create a `README.md` file.



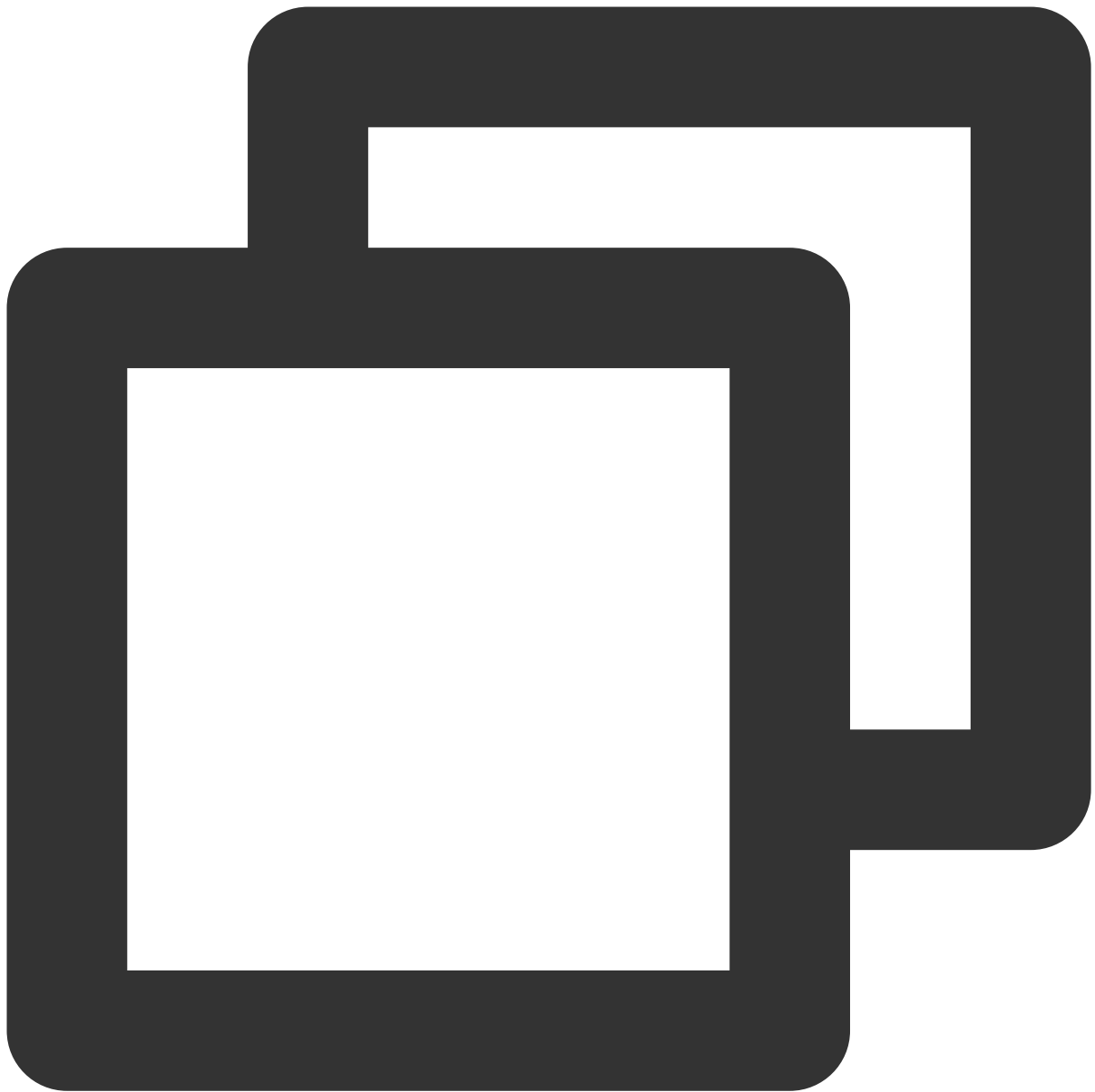
Right-click on a blank space and select `SVN commit...` to save the new file to the version repository:



Linux Environment

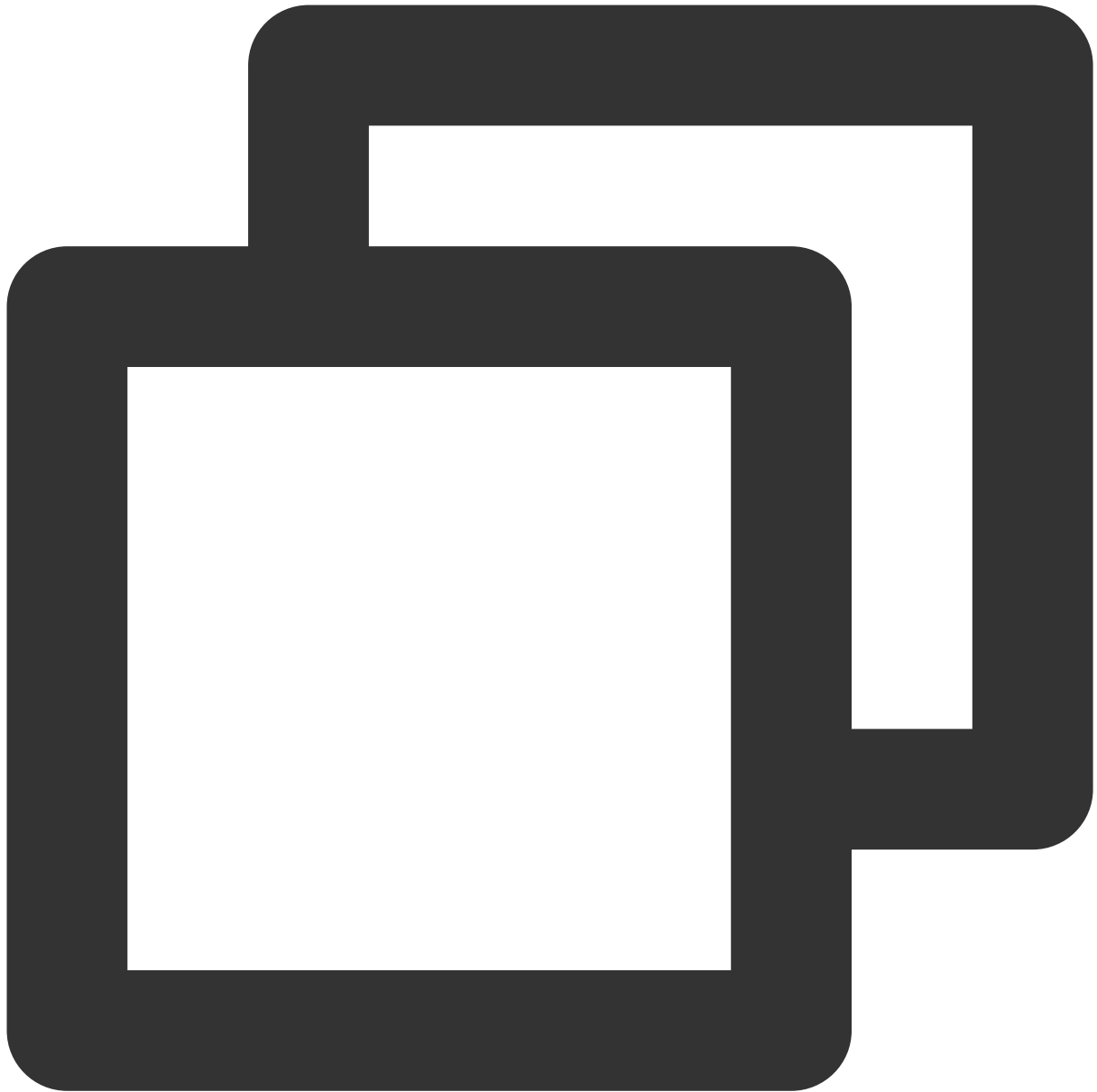
On a Linux system, you can use the system's package management tool to install SVN.

Install with yum in Fedora



```
$ sudo yum install subversion
```

Install with apt-get in Ubuntu or Debian



```
$ sudo apt-get install subversion
```

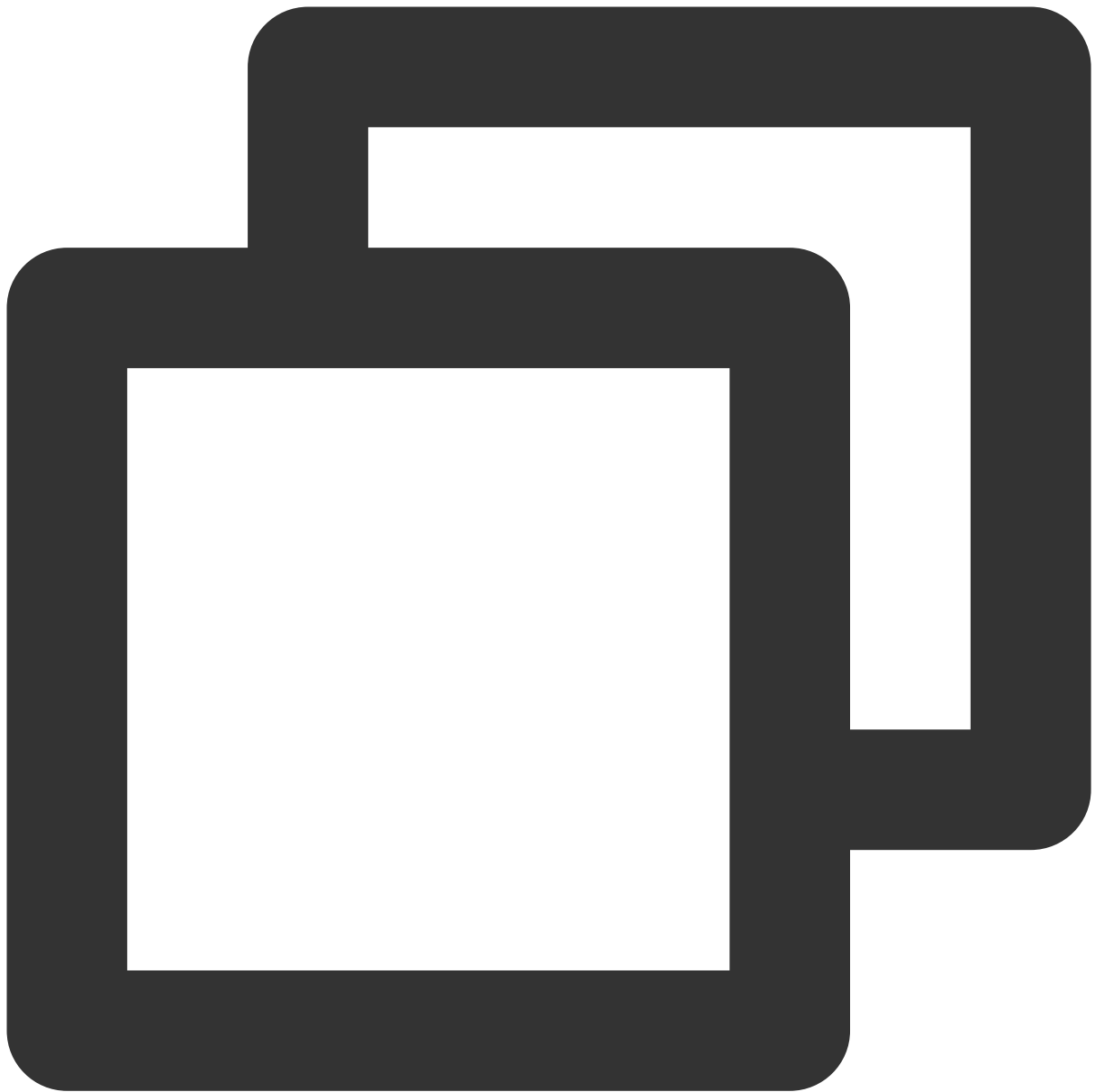
After installation, use `svn checkout / commit` to access the SVN repository.

Note:

This method is similar to the command line used in Mac systems.

"Negotiate authentication mechanism" error when using the SVN command line in Ubuntu

You may see the following error when using the SVN command-line client in Ubuntu:



```
svn: E210007: Cannot negotiate authentication mechanism
```

This occurs because the SVN authentication process uses the SASL library, so you need to run the following command to install the dependent library required to use SASL authentication:



```
$ sudo apt-get install cyrus-sasl2-dbg
```

Manage SVN Directory Permissions

Last updated : 2023-12-25 17:08:18

This document describes how to manage SVN repository permissions.

Open Project

1. Log in to the CODING Console and click the team domain name to go to CODING.
2. Click



in the upper-right corner to open the project list page and click a project icon to open the corresponding project.

3. In the menu on the left, select **Code Repositories**.
4. If Code Repositories is not shown on the left, the project admin needs to go to **Project Settings > Projects and Members > Functions** to enable the relevant function.

SVN code repositories support permission control. Admins can set directory permissions for individual users. Admins can set three types of permissions for repositories and subdirectories:

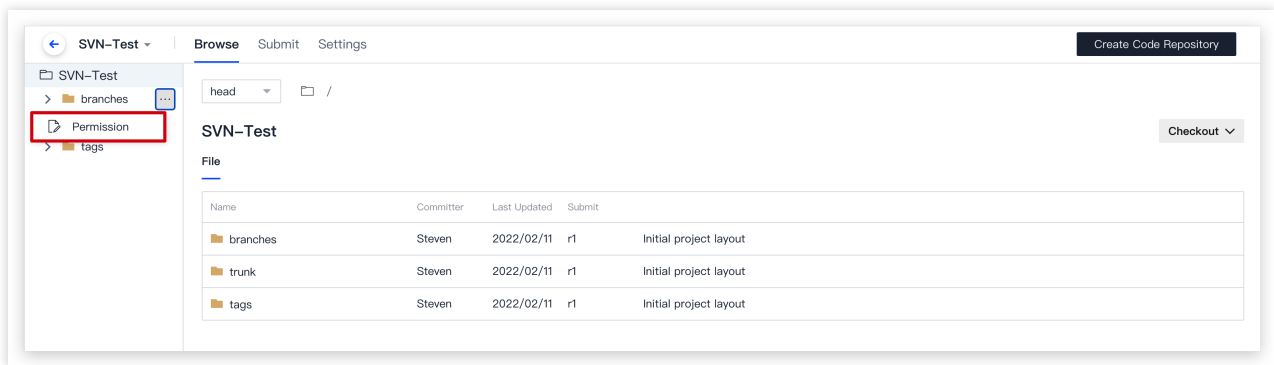
Read-only: Users can check out and view the specified directory, but cannot write to it.

Read/Write: Users can check out, view, and write to the specified directory.

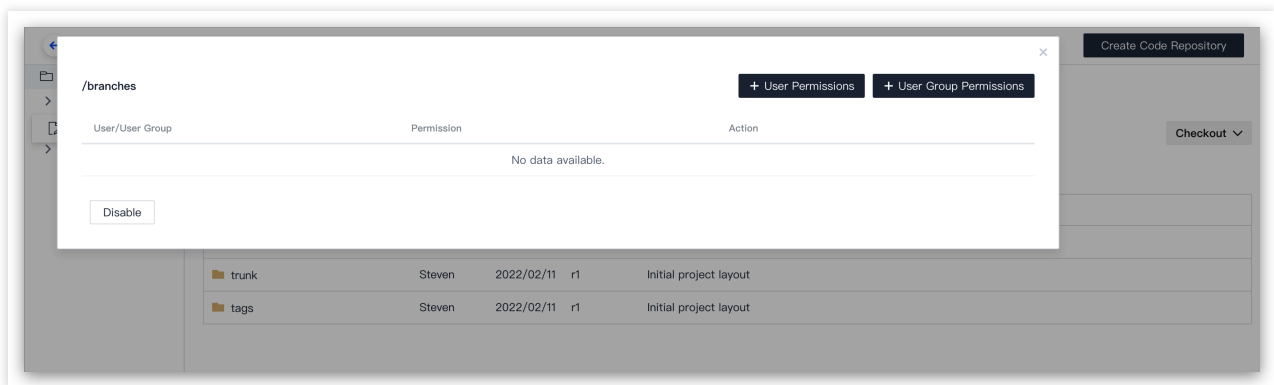
No permission: Users cannot check out, view, or write to the specified directory.

Set Permissions

1. Because each user has read/write permission for a repository by default, to set permission control for a directory in an SVN repository, you must click the More button for the directory and select **Permissions**.



2. On the permission settings page that appears, you can add individual users and corresponding permissions for this directory.

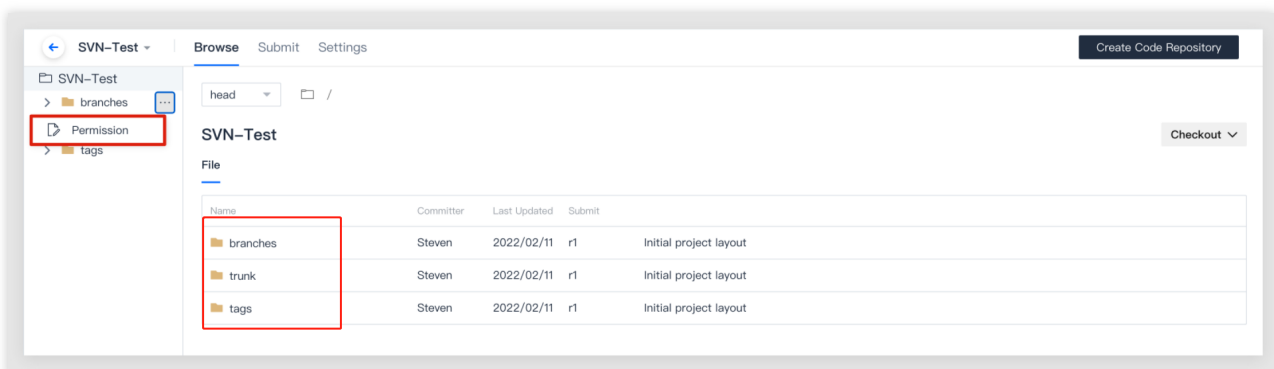


3. After you configure permission control for a directory, the directory is shown in a different color. Directories for which you have not configured permission control are shown in black, and all users have **read/write** permission for this directory.

Read/Write: black (default)

Read-only: yellow

No permission: gray



Permission Precedence

In some scenarios, different permission settings may be configured for a parent directory and its subdirectories. For example, a user may have read-only permission for the parent directory and read/write permission for a subdirectory. For an SVN repository, the precedence rules for parent directory permissions and subdirectory permissions are as follows:

If permissions are set for the parent directory but not its subdirectory, the subdirectory inherits the permission settings of the parent directory.

If permissions are set for both the parent directory and subdirectory, the subdirectory permissions take precedence.

For example:

1.1 If there is **read/write** permission to the parent directory and **read-only** permission to the subdirectory, the effective permission for the subdirectory is **read-only**.

1.2 If there is **read-only** permission to the parent directory and **read/write** permission to the subdirectory, the effective permission for the subdirectory is **read/write**.

1.3 If there is **read/write** or **read-only** permission to the parent directory and there is **no permission** to the subdirectory, the effective permission for the subdirectory is **no permission**.

SSH Protocol Usage

Configure SSH Public Keys

Last updated : 2023-12-25 17:08:18

This document describes how to configure SSH public keys.

Open Project

1. Log in to the CODING Console and click the team domain name to go to CODING.
2. Click



in the upper-right corner to open the project list page and click a project icon to open the corresponding project.

3. In the menu on the left, select **Code Repositories**.
4. If Code Repositories is not shown on the left, the project admin needs to go to **Project Settings > Projects and Members > Functions** to enable the relevant function.

Function Overview

SSH stands for Secure Shell. It is an encryption protocol used for network communication. SSH provides a secure data transmission environment in a public network environment. It is generally used to log in to a remote host and push/pull code.

You can add an SSH public key to a code repository, where it is called a **public deploy key**. Then, this key will grant read-only permission for this project by default. If you add the key to a personal account, it is called an **SSH public key**. It grants read/write permission to all projects under the account. The same SSH public key cannot be added to code repositories or personal accounts more than once.

Generate Public Key

Here, we use the `ssh-keygen` tool to generate an SSH public key. Run the following command:



```
ssh-keygen -m PEM -t rsa -b 4096 -C "your.email@example.com" // Creates a new SSH
Enter file in which to save the key (/Users/you/.ssh/id_rsa): [Press enter] // We
Enter passphrase (empty for no passphrase): // Here, just press Enter. If you set
```

Note:

If you need multiple SSH key pairs, when you are prompted to `Enter file in which to save the key`, enter a new file name, so the existing key pair will not be overwritten. For more information about SSH, see [Wikipedia](#). After the operation succeeds, you will see the following information:



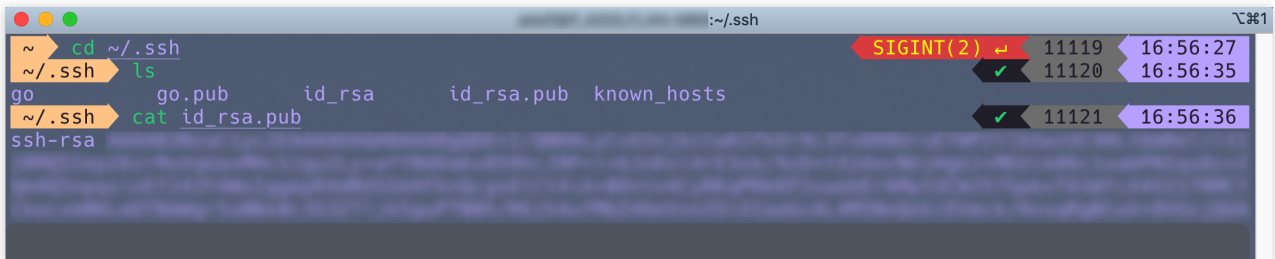
```
Your identification has been saved in /Users/you/.ssh/id_rsa.  
# Your public key has been saved in /Users/you/.ssh/id_rsa.pub.  
# The key fingerprint is:  
# 01:0f:f4:3b:ca:85:d6:17:a1:7d:f0:68:9d:f0:a2:db your.email@example.com
```

Add Public Key

You can add a public key to a code repository or personal account.

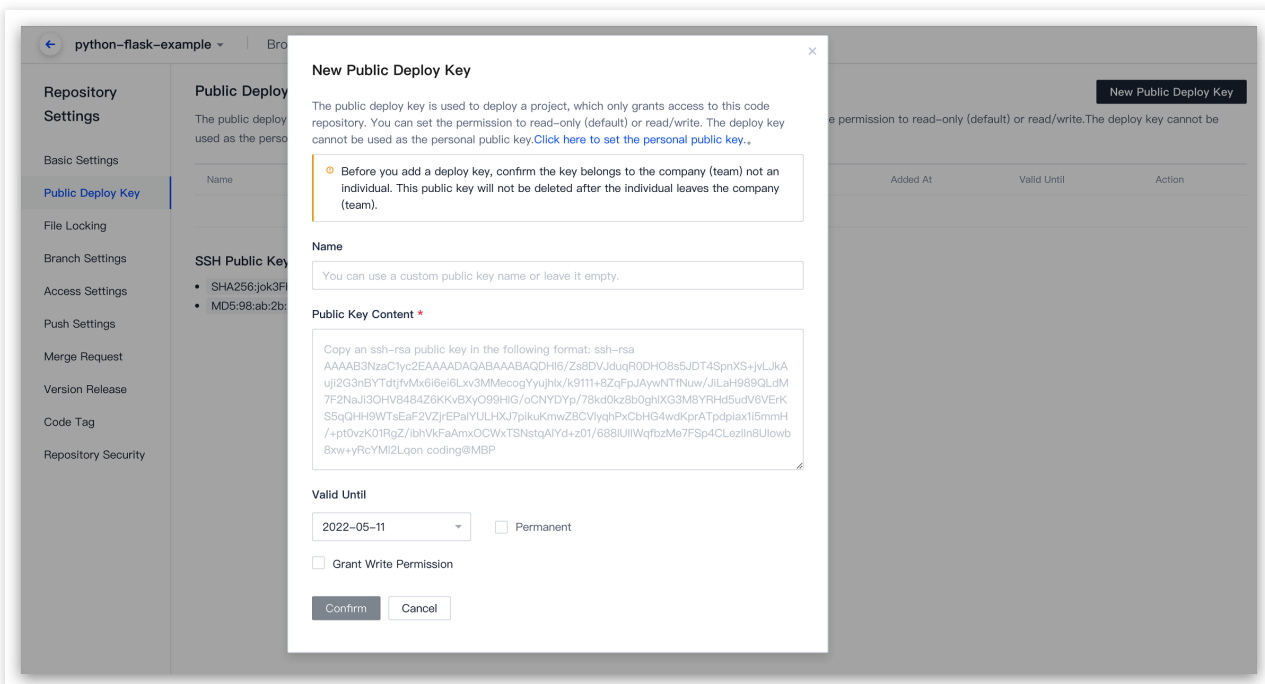
Associate public key with code repository

1. Go to the key pair address generated above (generally `~/ .ssh/`) and find the public key file with the `pub` extension. Use the `cat` command to output all content and copy the content.



```
~ ➜ cd ~/.ssh
~/.ssh ➜ ls
go go.pub id_rsa id_rsa.pub known_hosts
~/.ssh ➜ cat id_rsa.pub
ssh-rsa
```

2. Open the code repository and go to **Settings > Public Deploy Key**. Click **Add Public Deploy Key** and paste the full text of the public key.



Note:

Public deploy keys have read-only permission to the project by default. Select **Grant Push Permission** in the public deploy key settings to obtain push permission.

3. Then, the first time you attempt to connect from your local device, run the public key authentication command: `ssh -T git@e.coding.net`

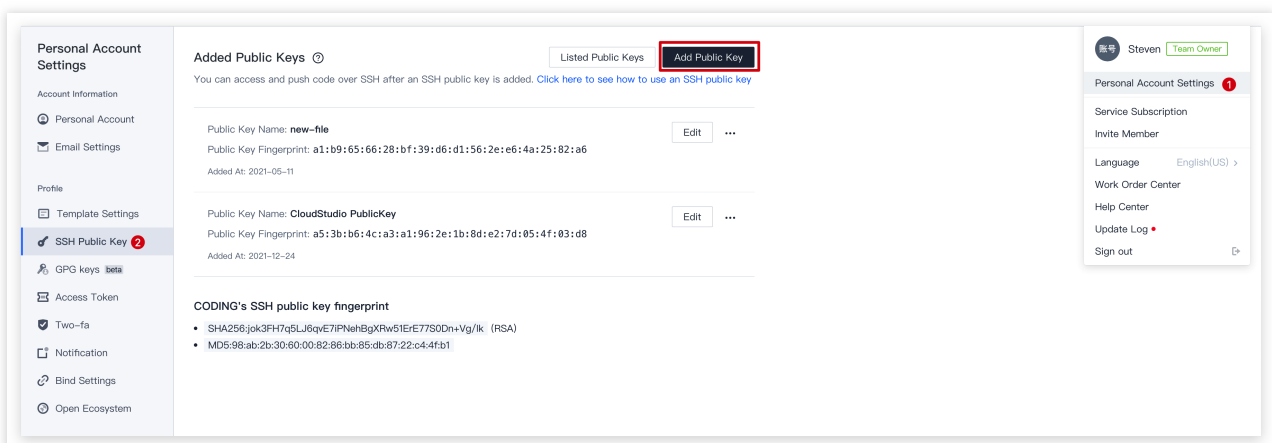
```
~ ssh -T git@e.coding.net SIGINT(2) 11127 10:
~
```

Associate public key with personal account

1. Go to the key pair address generated above (the default address is generally `~/ .ssh/`) and find the public key file with the `pub` extension. Use the `cat` command to output all content and copy the content.

```
~ cd ~/.ssh
~/.ssh ls
go go.pub id_rsa id_rsa.pub known_hosts
~/.ssh cat id_rsa.pub
ssh-rsa
```

2. Log in to CODING, click your profile photo in the upper-right corner, and go to **Personal Account Settings > SSH Public Keys**. Then, click **Create Public Key**.



3. Follow the instructions to paste the public key content you copied and enter a name for the public key.

4. Then, the first time you attempt to connect from your local device, run the public key authentication command: `ssh -T git@e.coding.net`.

```
~ ➤ ssh -T git@e.coding.net SIGINT(2) 11127 10:
~ ➤
```

Key Fingerprint Authentication

Last updated : 2023-12-25 17:08:18

This document describes how to use key fingerprints for code repository authentication. This ensures that the connected remote repository is a genuine CODING repository.

Open Project

1. Log in to the CODING Console and click the team domain name to go to CODING.
2. Click



in the upper-right corner to open the project list page and click a project icon to open the corresponding project.

3. In the menu on the left, select **Code Repositories**.
4. If Code Repositories is not shown on the left, the project admin needs to go to **Project Settings > Projects and Members > Functions** to enable the relevant function.

Function Overview

Code security is always essential. To ensure that the remote repository you are connecting to is an authentic CODING code repository, SSH key fingerprints are now provided for authentication. You simply need to run the command locally and verify the returned result to determine the authenticity of the remote repository.

Verify SHA256 Fingerprint

View the `e.coding.net` SHA256 fingerprint in the local `.ssh/known_hosts` file. If the return value is `jok3FH7q5LJ6qvE7iPNehBgXRw51ErE77S0Dn+Vg/Ik`, this indicates you have connected to the correct CODING server. You can view the result of the command on the terminal.



```
ssh-keygen -lf ~/.ssh/known_hosts
```

```
~ ssh-keygen -lf ~/.ssh/known_hosts
256 SHA256:Bdo9PWvc9YJra+FK28v7oxW0dghA/DI3ZLT3BhDz/nQ [ ]:12400 (ECDSA)
256 SHA256:V8qgXUfieT6Q//G/miMxK+8Dx05gS/2NaNpPYAU629s [ ]:28954 (ECDSA)
256 SHA256:ox9ko2YsRDgwp4C9im0Tha9FWxAXhfe7H7yLIXhcT5A [ ]:11900 (ECDSA)
256 SHA256:u2Xk2ekDfmrav2FALTppnGX9seyiZEK0vsFiGyp/EKo [ ]:28524 (ECDSA)
256 SHA256:za8qm0BYDLKBCx+hG4gT4/Oiq06ZR/w00JhMoqJIWtA [ ]:ECDSA)
2048 SHA256:jok3FH7q5LJ6qvE7iPNehBgXRw51ErE77S0Dn+Vg/Ik e.coding.net, (RSA)
```

Verify MD5 Fingerprint

View the `e.coding.net` MD5 fingerprint in the local `.ssh/known_hosts` file. If the return value is

`98:ab:2b:30:60:00:82:86:bb:85:db:87:22:c4:4f:b1`, this indicates you have connected to the correct CODING server. You can view the result of the command on the terminal.



```
ssh-keygen -E md5 -lf ~/.ssh/known_hosts
```

```
~ ssh-keygen -E md5 -lf ~/.ssh/known_hosts
256 MD5:cd:aa:1f:f5:f7:4d:44:a9:37:93:7d:22:94:6a:bb:b9 [ ]:12400 (ECDSA)
256 MD5:72:c3:89:c4:3e:d3:9d:6d:3e:d8:bc:af:bf:91:61:ce [ ]:28954 (ECDSA)
256 MD5:81:e4:07:c6:ec:38:5c:1a:da:03:d9:fb:34:83:9f:5c [ ]:11900 (ECDSA)
256 MD5:b1:ea:db:6b:04:36:d7:b5:03:57:28:7b:85:33:f4:7e [ ]:28524 (ECDSA)
256 MD5:20:8f:38:63:70:15:70:cc:a7:81:79:ff:ea:e5:11:98 (ECDSA)
2048 MD5:98:ab:2b:30:60:00:82:86:bb:85:db:87:22:c4:4f:b1 e.coding.net, (RSA)
~
```


Push/Pull Code Via SSH Protocol

Last updated : 2023-12-25 17:08:18

This document describes how to use SSH protocol to push/pull code.

Open Project

1. Log in to the CODING Console and click the team domain name to go to CODING.
2. Click

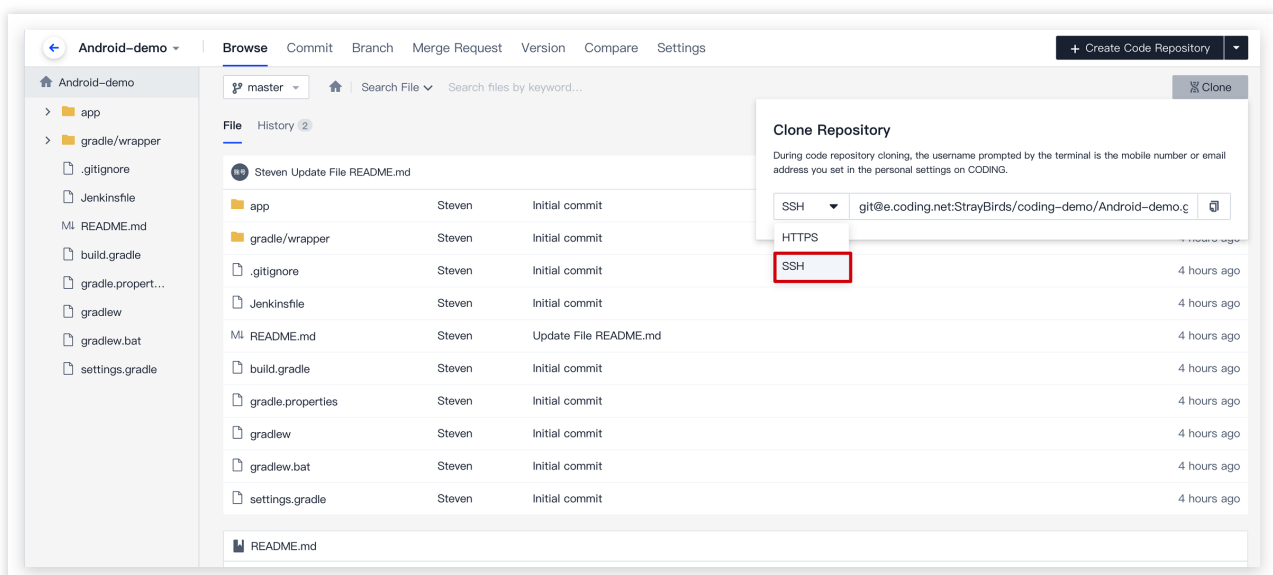


in the upper-right corner to open the project list page and click a project icon to open the corresponding project.

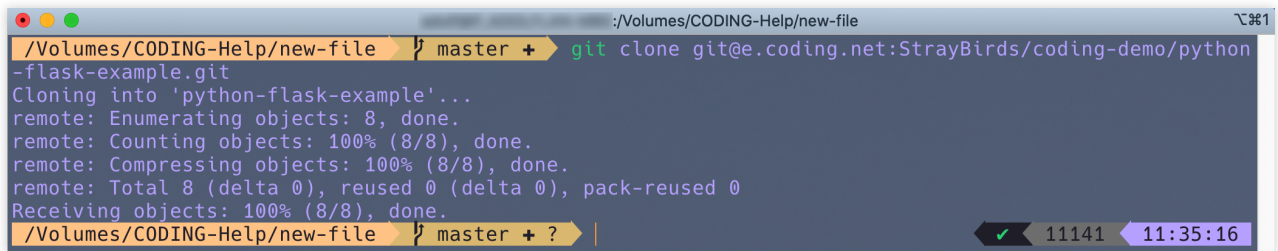
3. In the menu on the left, select **Code Repositories**.
4. If Code Repositories is not shown on the left, the project admin needs to go to **Project Settings > Projects and Members > Functions** to enable the relevant function.

CODING supports pushing/pulling code via SSH protocol.

1. Refer to [Configure Public Keys](#) to generate a public key and add it to a code repository or personal account.
2. Copy the SSH URL on the code repository's Browse page.



3. Run the `git clone + repository URL` command on your local machine to pull code.



```
:/Volumes/CODING-Help/new-file  master +  git clone git@e.coding.net:StrayBirds/coding-demo/python-flask-example.git
Cloning into 'python-flask-example'...
remote: Enumerating objects: 8, done.
remote: Counting objects: 100% (8/8), done.
remote: Compressing objects: 100% (8/8), done.
remote: Total 8 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (8/8), done.
/Volumes/CODING-Help/new-file  master + ?  11141  11:35:16
```

Branch Management

Create Branch

Last updated : 2023-12-25 17:08:18

This document describes how to create branches in code repositories.

Open Project

1. Log in to the CODING Console and click the team domain name to go to CODING.
2. Click



in the upper-right corner to open the project list page and click a project icon to open the corresponding project.

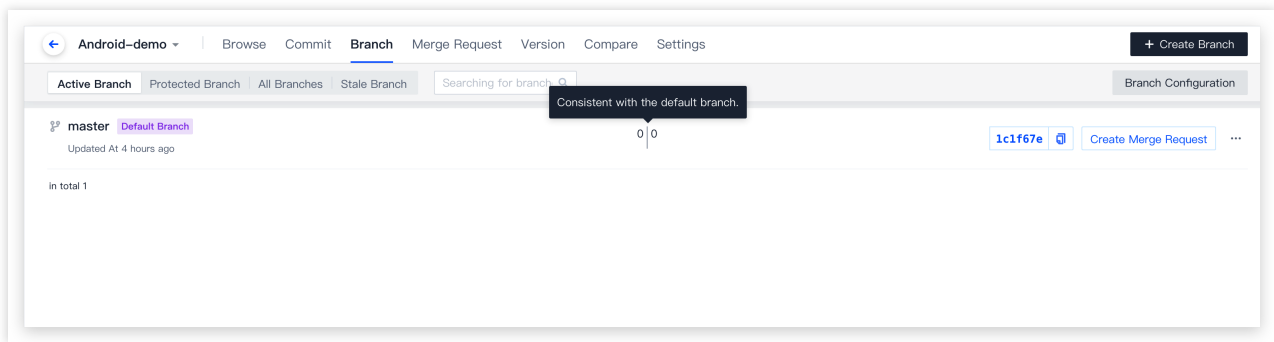
3. Select **Code Repositories** in the menu on the left and click **Branches** to go to the branch management page.

Branches are a common function in Git. The common development approach is to retain one trunk (or master branch), while each development or bug fix task is carried out on its own branch. When finished, the branches are merged back into the trunk. Because development directly on the trunk is very risky, the branch can be viewed as a safety valve, which isolates different development tasks. It can ensure that the stability of the master version is not compromised, while allowing different people to focus on different development tasks.

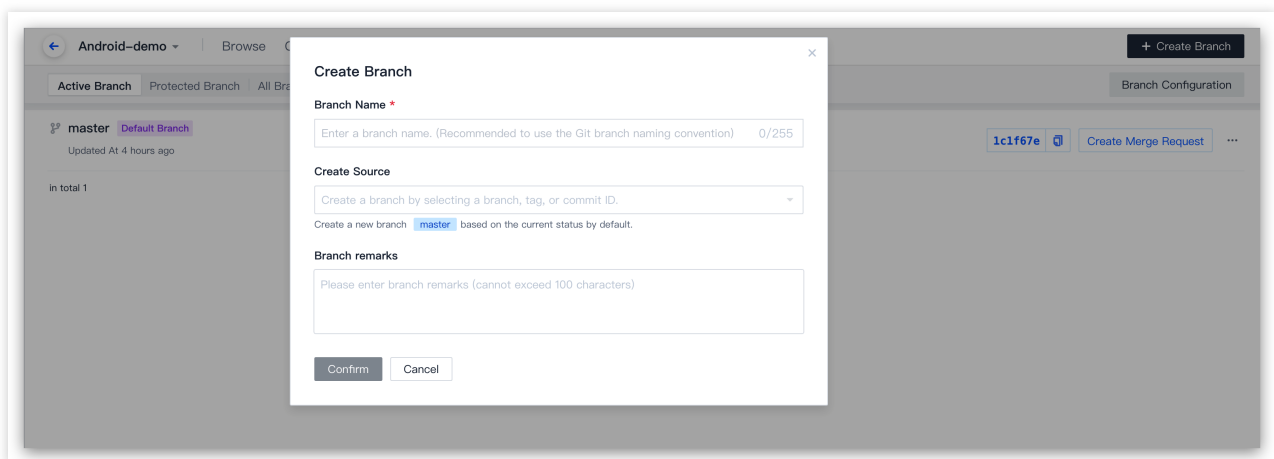
Note:

Below, we will refer to branches in CODING code repositories as remote branches and branches in local Git repositories as local branches. You can run commands on your local terminal to quickly create branches. For details, see [Common Git Commands](#).

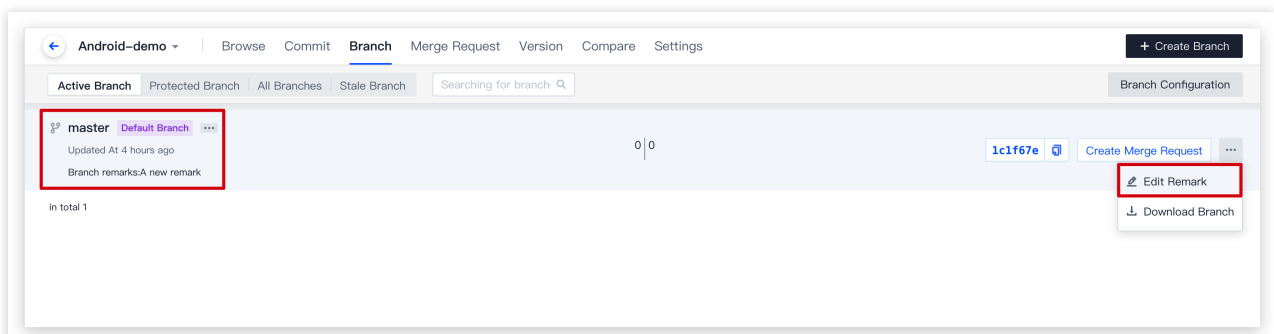
1. Go to the code repository details page and click the **Branches** tab to see all branches in the remote repository. Here, you can create branches and enable protected branches. The branch list tab displays the number of additional or missing commits relative to the default branch at present.



2. Click **Create Branch** in the upper-right corner and follow the instructions in the pop-up window to enter the relevant configuration information. The master branch is used as the source by default, and a new branch is derived from this source.



3. When creating a branch, you can add a simple description to indicate the purpose of the branch. When the branch name cannot fully indicate its purpose, add additional information in the Branch Note.



Set Default Branch

Last updated : 2023-12-25 17:08:18

This document describes how to set default branches.

Open Project

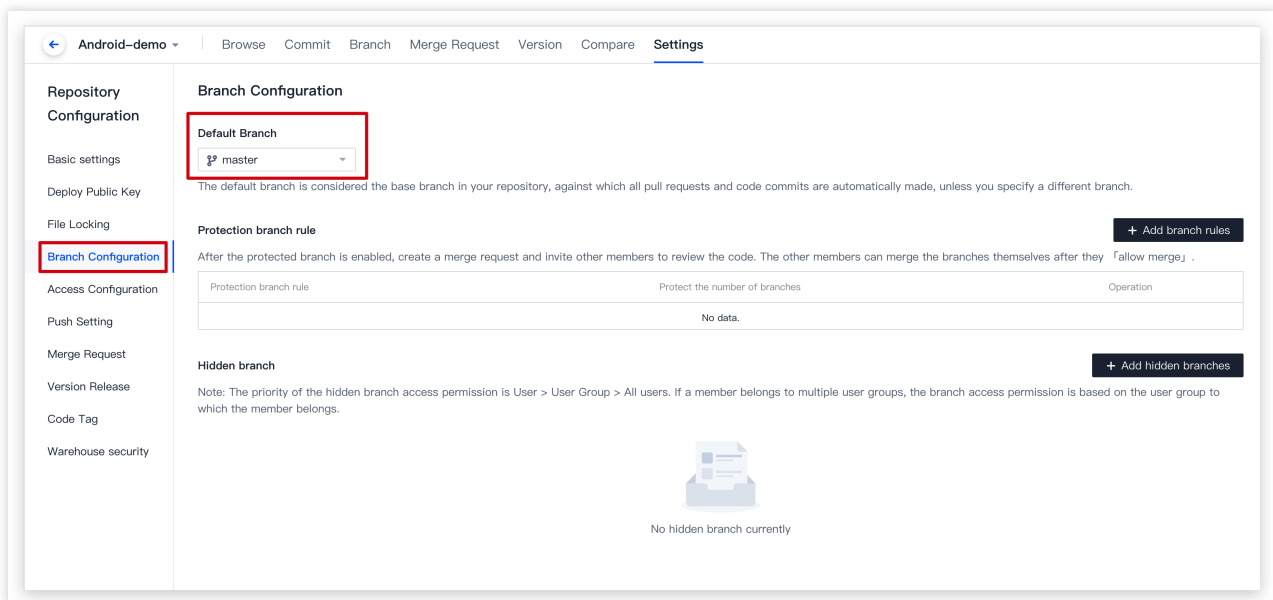
1. Log in to the CODING Console and click the team domain name to go to CODING.
2. Click



in the upper-right corner to open the project list page and click a project icon to open the corresponding project.

3. Select **Code Repositories** in the menu on the left and click **Branches** to go to the branch management page.

Unless you specify a different branch, the default branch in a repository is the base branch for pull requests and code commits. To modify the default branch, go to **Settings > Branch Settings**. Only project admins have permission to change the default branch.



Set Protected Branch

Last updated : 2023-12-25 17:08:18

This document describes how to set protected branches.

Open Project

1. Log in to the CODING Console and click the team domain name to go to CODING.
2. Click

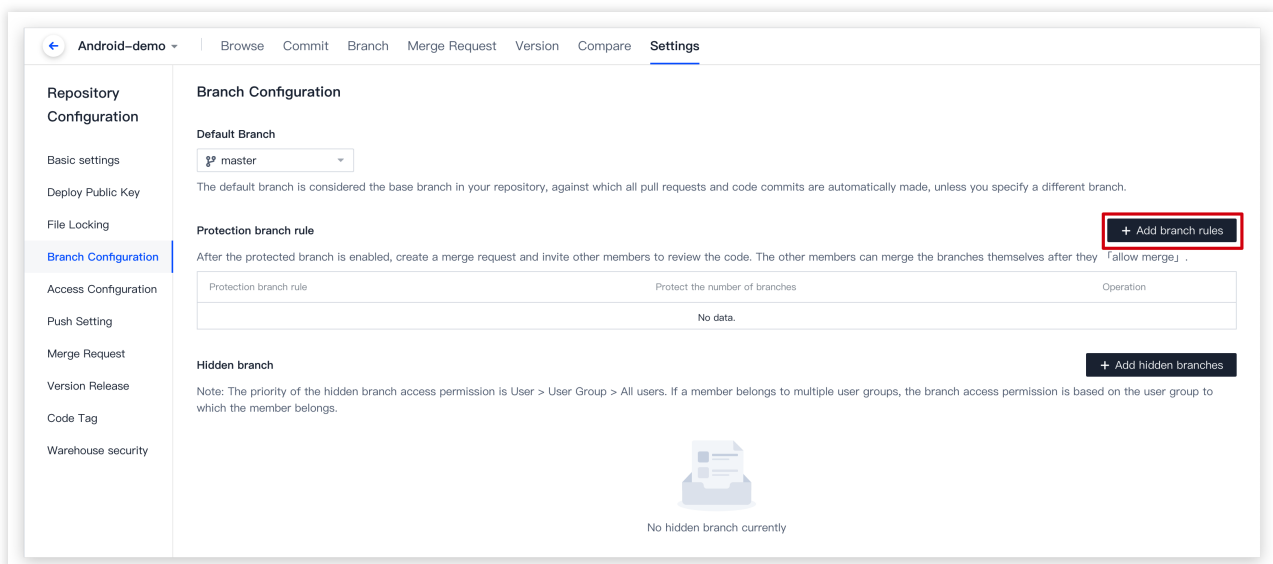


in the upper-right corner to open the project list page and click a project icon to open the corresponding project.

3. Select **Code Repositories** in the menu on the left and click **Branches** to go to the branch management page.

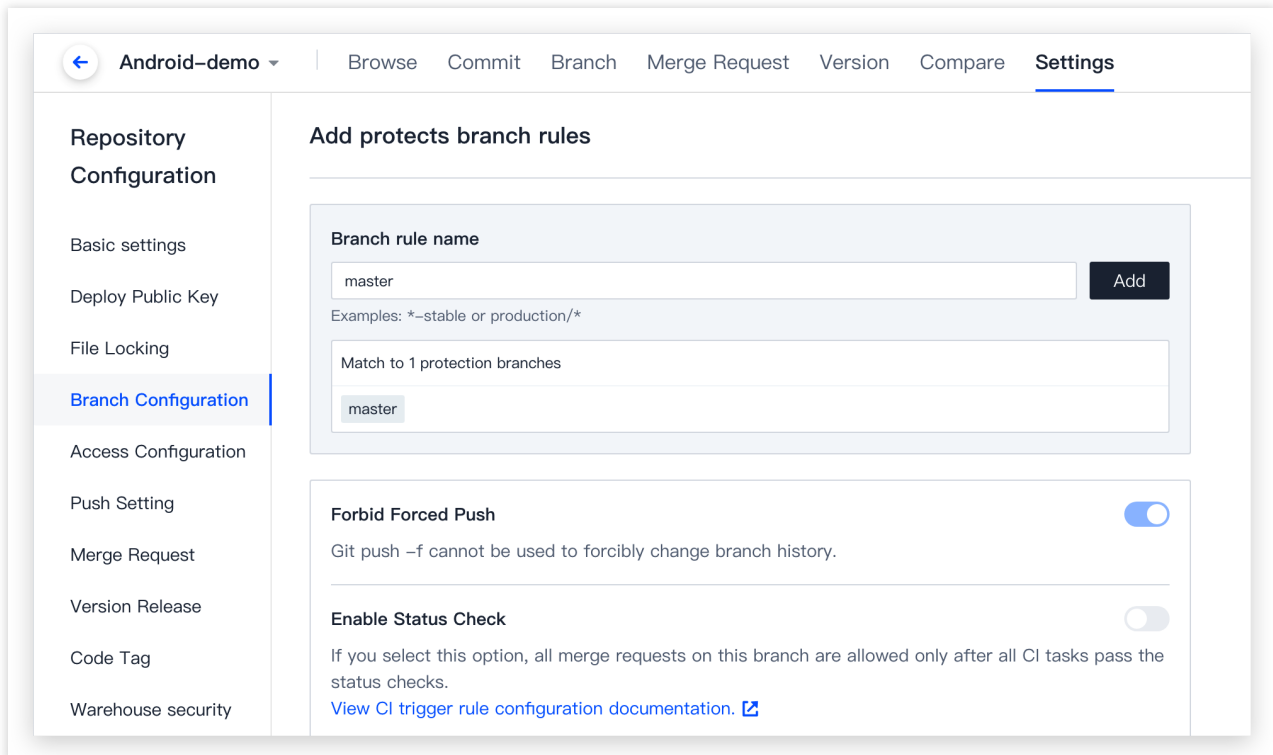
Branch protection is a special function CODING uses to limit Git code permissions. Selected branches can be protected from unreported and unauthorized changes.

When you enable branch protection, protected branches are marked with a green shield in the branch list. To modify a protected branch, members must create a new branch and modify it. Then, they submit a merge request to invite other members to review the code. If the review result is Allow Merge, the merge operation can be performed.



Set Protected Branch Rules

In a code repository, go to **Settings > Branch Settings**. There, you can use **wildcards** to intelligently set protected branches. Branches that match the set name rules are judged to be protected branches.



Disallow Force Push: Enabled by default. Even users with permission to perform git push cannot use git push -f to force modify the branch commit history. We strongly recommend you enable this option when multiple people collaborate on a branch. This ensures that users must use new commits to change the branch history, rather than modifying previous commits.

Enable Status Check: By setting the specification check conditions or setting a code scanning scheme in the CI, merges are allowed only after the CI is successful. For more information, see [Continuous Integration > Trigger Rules](#).

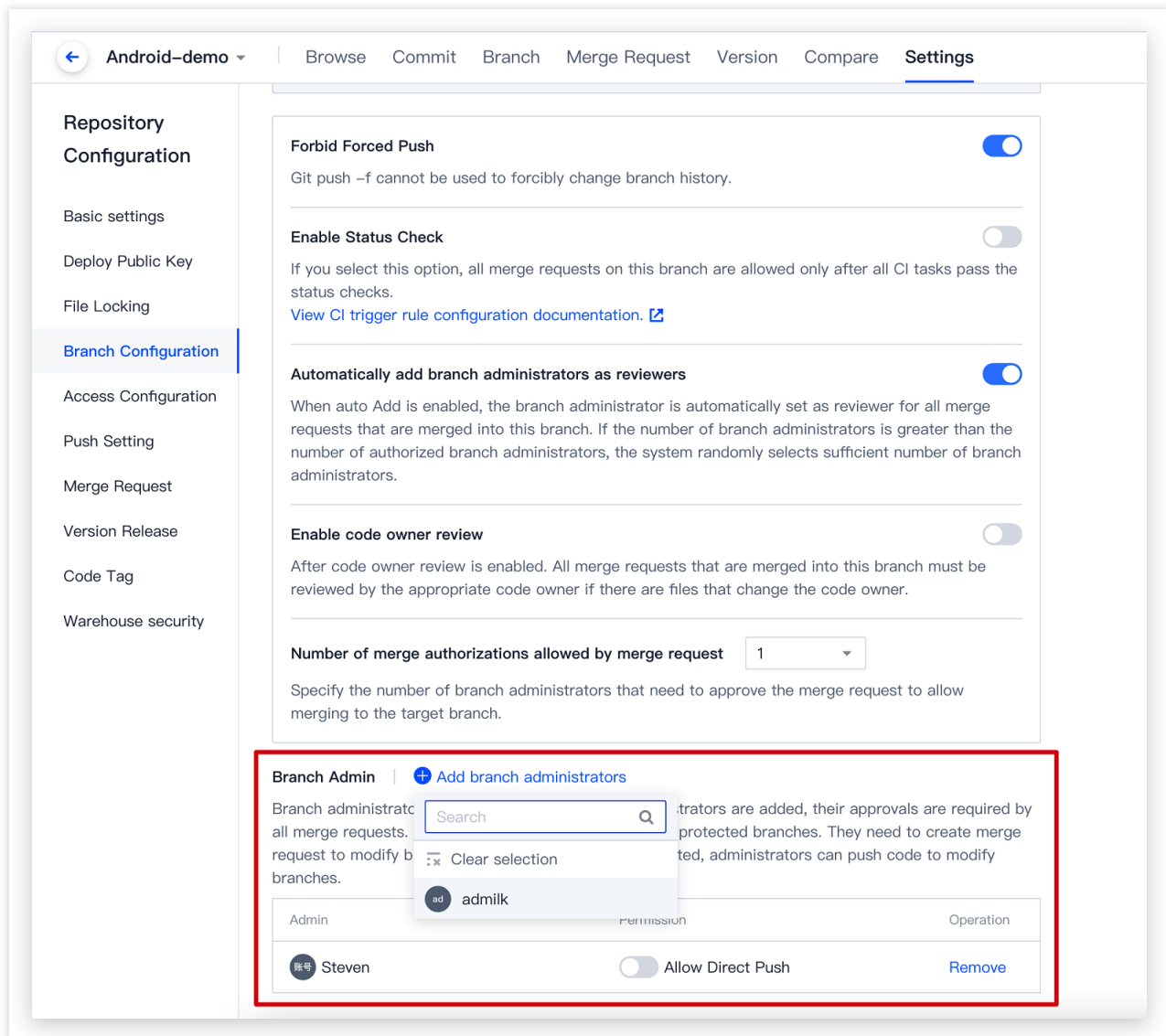
Automatically Add Branch Admin as Reviewer: When this option is enabled, the relevant branch admins are automatically set as reviewers for all merge requests involving protected branches. When the number of branch admins exceeds the **Number of Reviewers Authorizing Merge**, an appropriate number of reviewers are randomly selected from among the branch admins. For example, if there are three branch admins and the number of reviewers is 2, the system randomly selects two of the three admins as reviewers.

Enable Review by Code Owner: When this option is enabled, for merge requests involving protected branches, any modification must be reviewed by the relevant code owner before the merge is allowed. For more information, see [Code Owner](#).

Number of Reviewers Authorizing Merge: This is used to set the number of branch admins that must authorize a merge request. If no branch admins are set for a protected branch, a merge request must be authorized by one ordinary member before the merge is allowed.

Specify Branch Admins

Branch Admin is optional. After an admin is added, the admin must Allow Merge for every merge request. By default, admins are restricted by the conditions of protected branches, and they need to create merge requests to modify the branches. Select **Allow Direct Push** to allow admins to directly modify the content of protected branches.



If members do not have permissions for a branch (e.g., when they are not admins of a protected branch), they will receive the following error message when they try to push to this branch.


```
:/Volumes/CODING/coding-help-generator  api ↑1 ②  git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 378 bytes | 378.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0)
remote: [err=31] You have no permission to update protected branch (refs/heads/a
pi).
remote: (refs/heads/api) , https://coding
.net/help/doc/git/git-branch.html#
remote: error: hook declined to update refs/heads/api
To https://e.coding.net/codingcorp/coding-help-generator.git
! [remote rejected] api -> api (hook declined)
error: failed to push some refs to 'https://e.coding.net/codingcorp/coding-help-
generator.git'
:/Volumes/CODING/coding-help-generator  api ↑1 ②  |
```

Set Hidden Branch

Last updated : 2023-12-25 17:08:18

This document describes how to set hidden branches.

Open Project

1. Log in to the CODING Console and click the team domain name to go to CODING.
2. Click

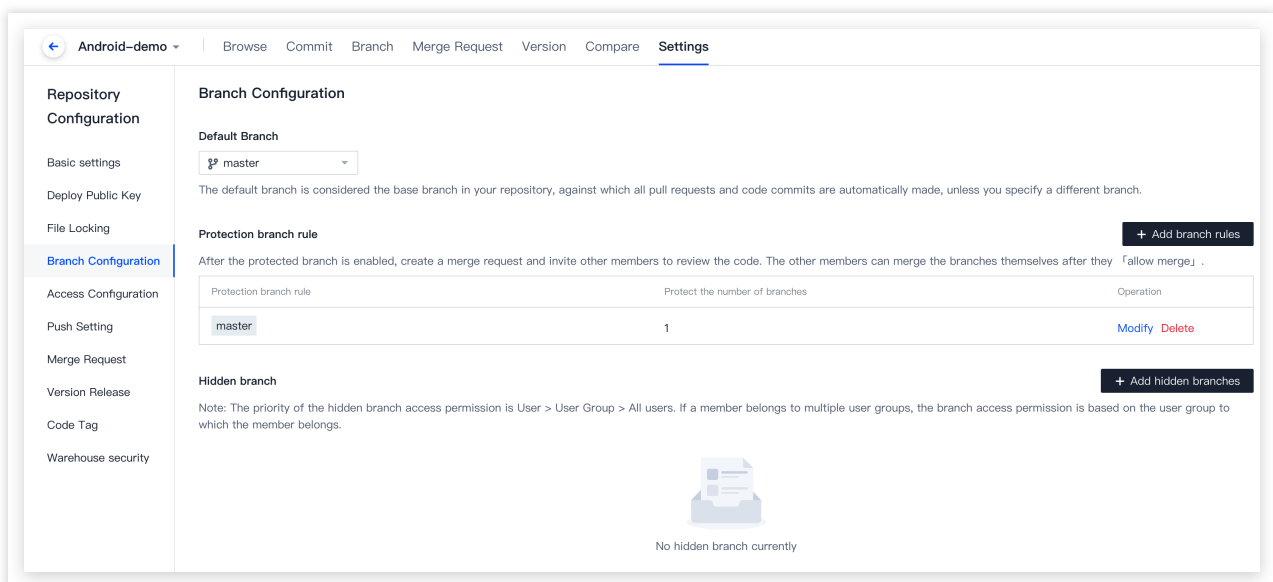


in the upper-right corner to open the project list page and click a project icon to open the corresponding project.

3. Select **Code Repositories** in the menu on the left and click **Branches** to go to the branch management page.

To control access permissions to a single branch, you can make any branch other than the default branch a hidden branch. Only authorized users and user groups can access hidden branches, ensuring the security of the code.

1. On the details page of a code repository, click **Settings** > **Branch Settings** to go to the branch settings page.




2. Click **Add Hidden Branch**, select or enter the branch, and click **Save**. When a branch is hidden, it is shown in the hidden branch list.

3. Use **Add User Group** or **Add Member** to specify the users that are allowed or denied access to the branch.

Hidden branch+ Add hidden branches



Note: The priority of the hidden branch access permission is User > User Group > All users. If a member belongs to multiple user groups, the branch access permission is based on the user group to which the member belongs.

 master

Save

+ Add a user group or member

Delete

  2

☒ Allow access

Note:

The priority of hidden branch access permissions is as follows: user > user group > all users. If a member belongs to multiple user groups, the member can access a hidden branch if any of their groups has access permission.

For example, if User A belongs to User Group 1 (allowed access to the `dev/001` branch) and User Group 2 (denied access to the `dev/001` branch), User A can access the `dev/001` branch.

Use Code Owner Mechanism

Last updated : 2023-12-25 17:08:18

This document describes how to use the code owner mechanism.

Open Project

1. Log in to the CODING Console and click the team domain name to go to CODING.
2. Click



in the upper-right corner to open the project list page and click a project icon to open the corresponding project.

3. Select **Code Repositories** in the menu on the left and click **Branches** to go to the branch management page.

The code owner mechanism must be used together with [Branch Protection](#). Place the declaration file `CODEOWNERS` in a code repository to declare the owner of the code files in this repository. Generally, the code owner is the project owner.

When the target branch of a merge request is a protected branch and the files changed in the request involve the paths or files set in the declaration file, the merge request details will list the corresponding owners and their review statuses.

Merge Status Check: Success

A merge check succeeded

coding-deploy syntax check

 — Build succeeded

Detail

The target branch has no scan task

Code scanning can find code defects and security vulnerabilities hidden in the merged code, and help you evaluate the quality of the merged code.

Go to enable

No violations found in R&D specifications

Normal merge operation

Code Owner:

The change content matches 1 code owner of the target branch, and 0 code owners have passed the review.

charts/repos/**

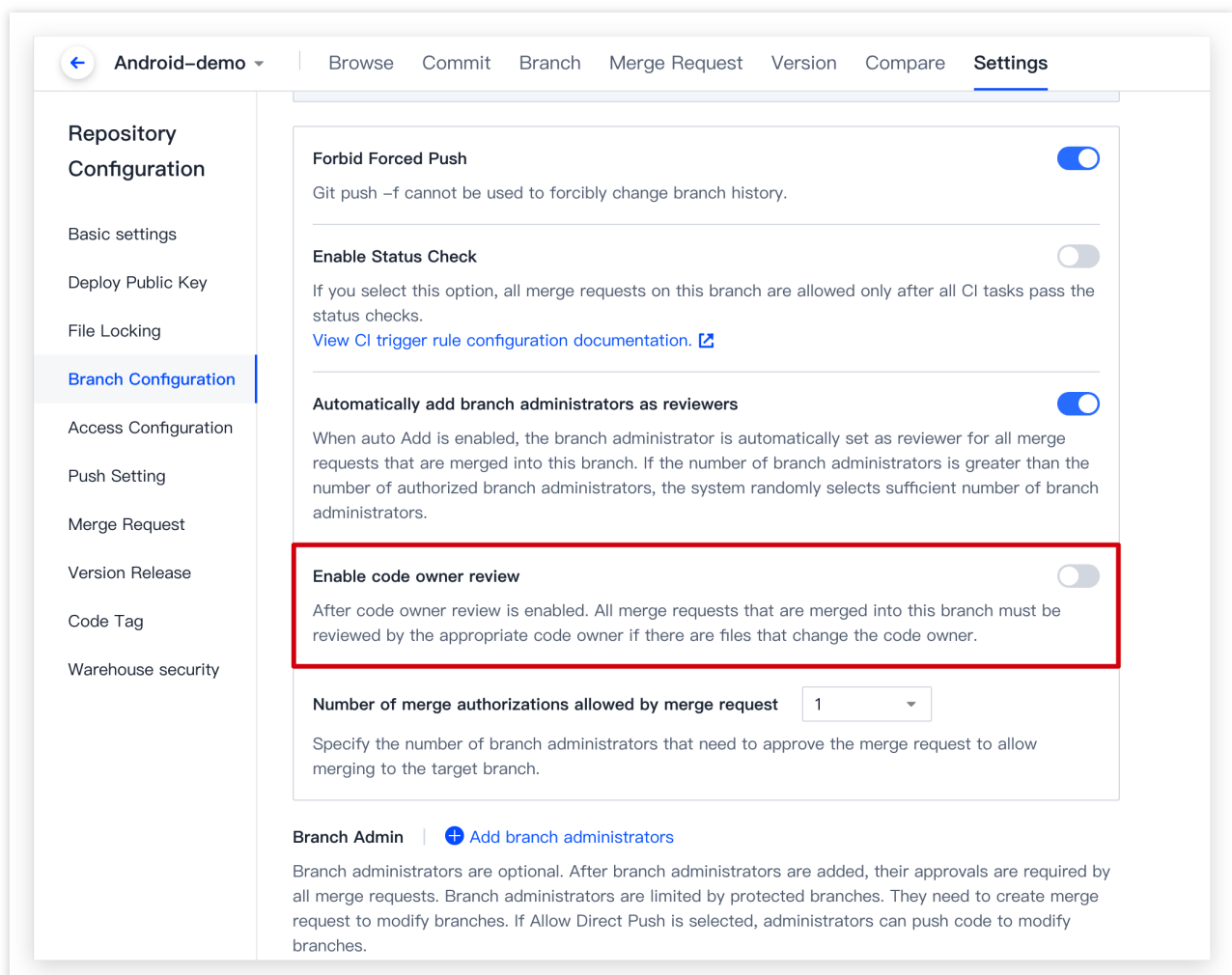
 — Not yet reviewed

In the figure above, the `CODEOWNER` file declares that the owner of the files in the `charts/repos/**` path is Sally. When a merge request is submitted for a protected branch and it involves changes to files in the `charts/repos/` path, Sally is automatically added as a reviewer for this request.

Note:

You can use the continuous integration plugin to automatically add reviewers. For details, see [Automatically Add Merge Request Reviewers](#).

In the **protected branch** settings, toggle on Enable Review by Code Owner. Then, code owners must Allow Merge for every merge request that changes code under their jurisdictions.



Declaration file URL

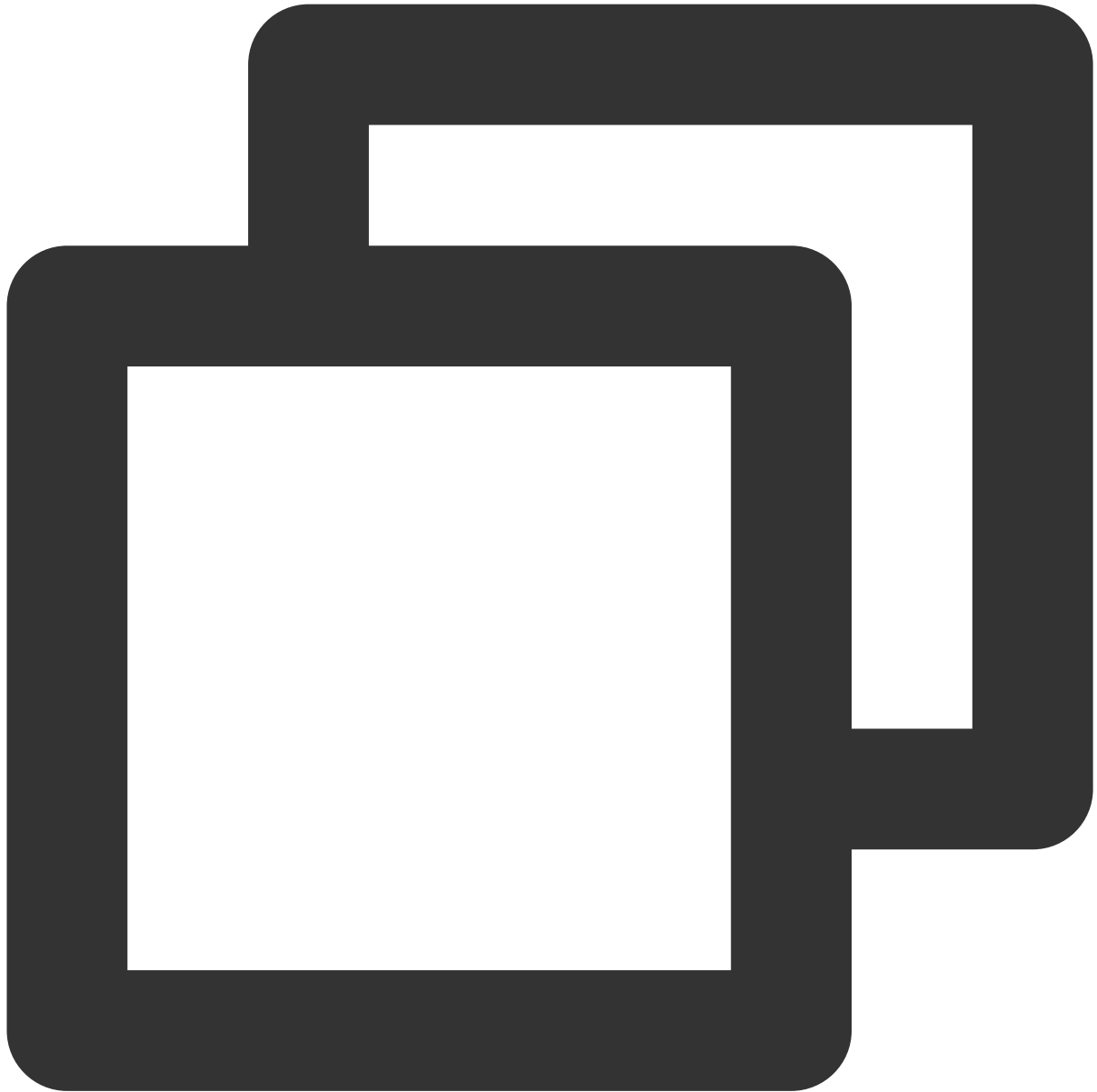
By default, `CODEOWNERS` files are searched for layer by layer from the following locations. The file name must be in uppercase, and the search will stop when one file is found.

Root directory

`docs/` directory

Declaration file format reference

The declaration file is a normal text file. Blank lines and lines starting with `#` are ignored. Each line uses the following format:



```
pattern email email email ...
```

`pattern` indicates a file path mode. `email` is the owner's email. You can enter multiple owners separated by spaces. Sample file:

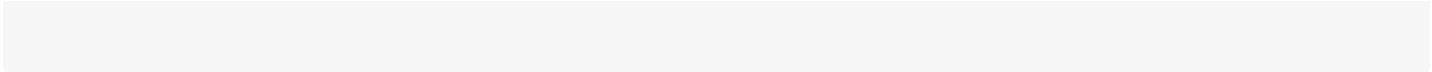


```
# Declares all files with the extension .js
*.js yourname@coding.net

# Declares the files in the build/logs/ directory (including subdirectories) under
/build/logs/ yourname@coding.net

# Declares all files in docs/ folders (not including subdirectories)
docs/* yourname@coding.net

# Declares all files in the docs/ folder (including subdirectories) under the root
/docs/ yourname@coding.net
```



Merge Request and Code Review

Adjust Merge Request Settings

Last updated : 2023-12-25 17:08:18

This document describes how to adjust the settings of merge requests in a code repository.

Open Project

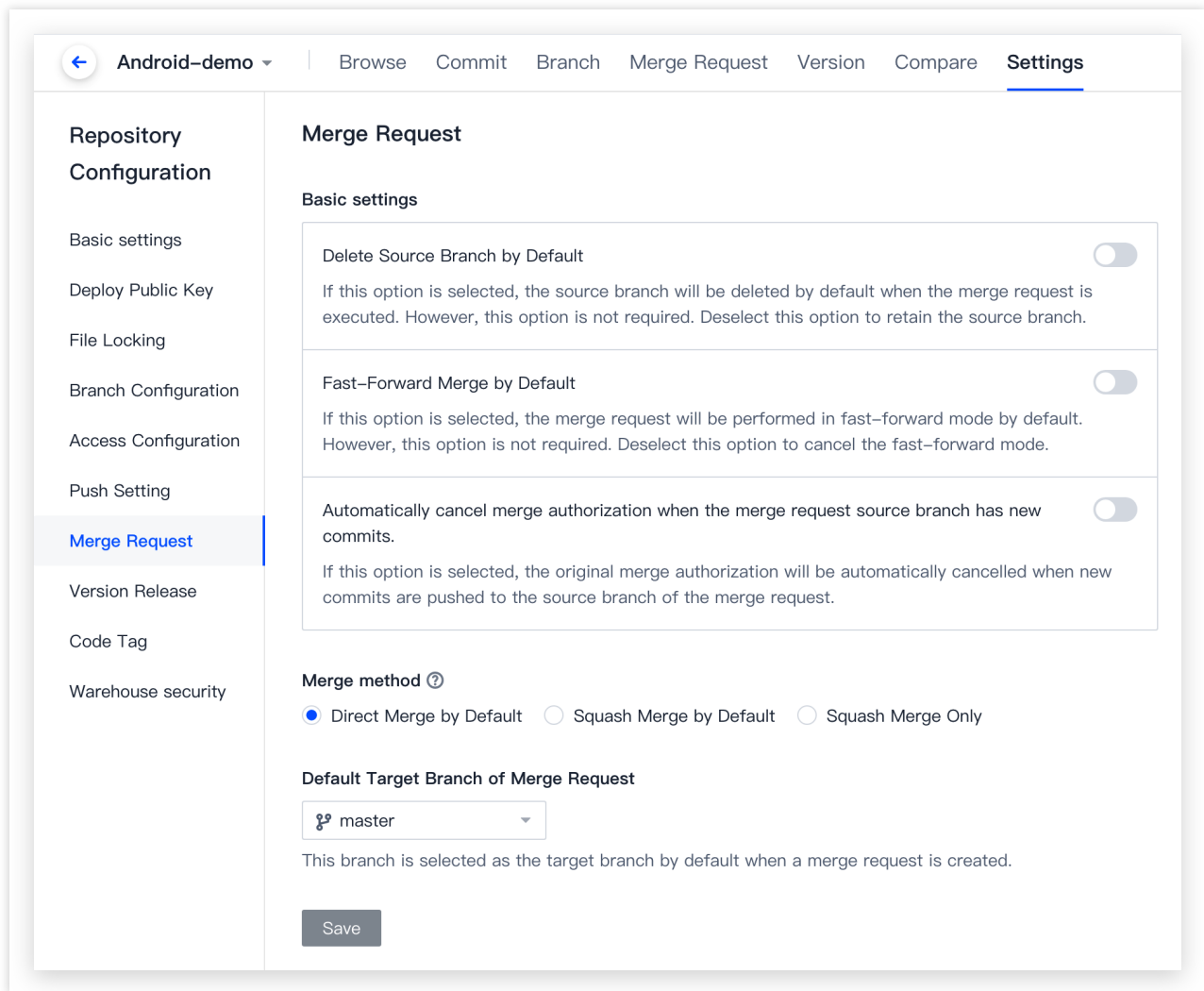
1. Log in to the CODING Console and click the team domain name to go to CODING.
2. Click



in the upper-right corner to open the project list page and click a project icon to open the corresponding project.

3. In the menu on the left, click **Code Repositories > Merge Requests**.

Project admins can configure the basic settings as well as default merge modes and target branches of merge requests in **Settings > Merge Requests**.



Delete Source Branch by Default

If this is enabled, the source branch will be deleted after it is merged into the target branch.

Fast-Forward Merge by Default

If this is enabled, when there is a direct linear path from the source branch to the target branch, the source branch will directly point to the target branch without a merge commit. This process is called the fast-forward merge.

Merge Mode

Three merge modes are available for a source branch with multiple commits:

Direct Merge by Default: Creates a merge commit.

Squash Merge by Default: Combines multiple commits of a source branch into one commit, which can be canceled by users.

Only Squash Merge: Force combines multiple commits of a source branch into one commit, which cannot be canceled.

Default Target Branch

The default target branch for merge requests. We recommend you set the master branch as the default target branch for merge requests.

Merge Branch

Last updated : 2023-12-25 17:08:18

This document describes how to merge branches.

Open Project

1. Log in to the CODING Console and click the team domain name to go to CODING.
2. Click



in the upper-right corner to open the project list page and click a project icon to open the corresponding project.

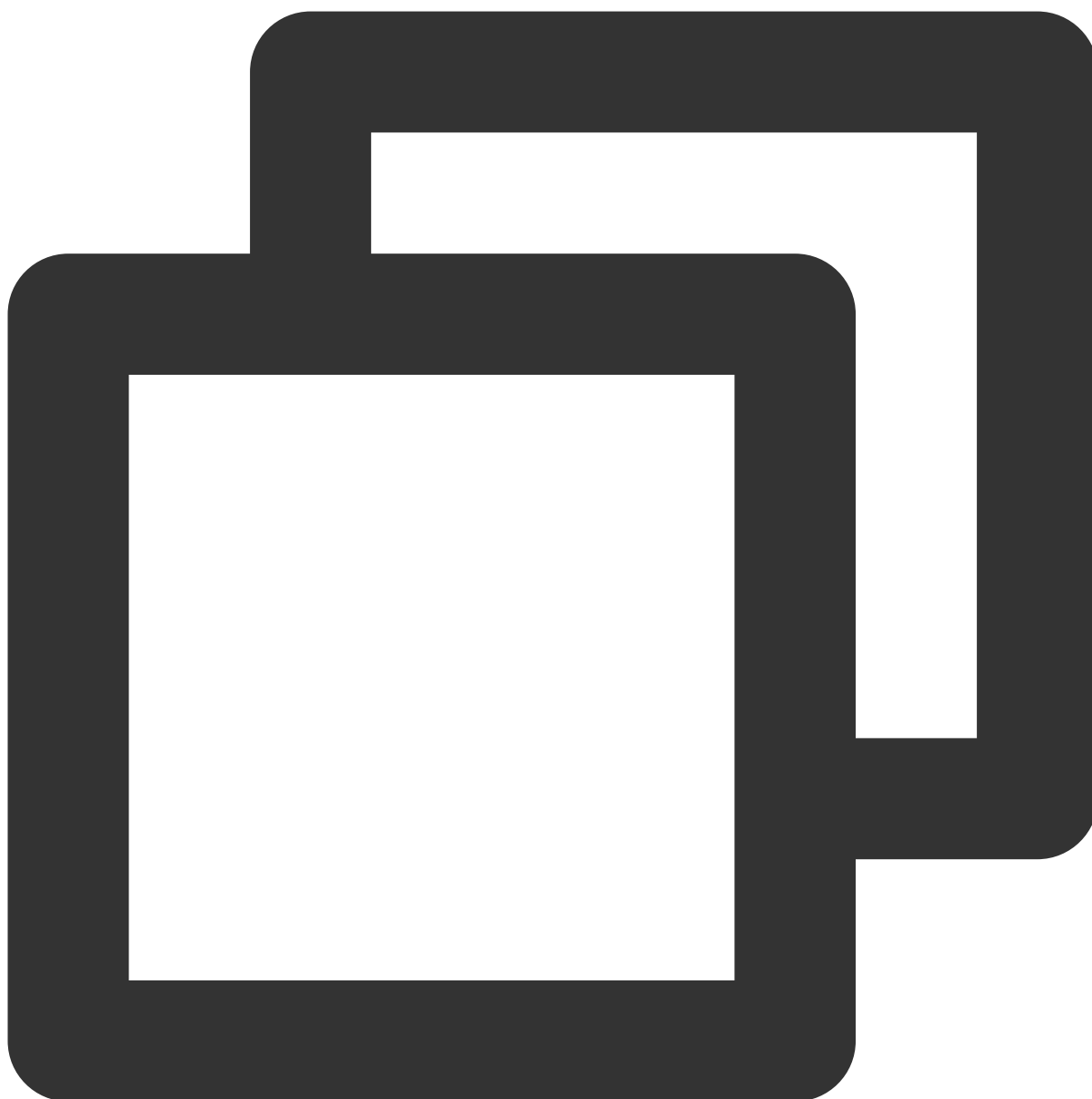
3. Select **Code Repositories** in the menu on the left and click **Branches** to go to the branch management page.

If a multi-branch development workflow is used, we recommend you set the master branch as a [protected branch](#). Developers can create temporary develop branches and initiate merge requests for them. After continuous integration (CI) and code reviews, developers can merge the develop branch into the master branch.

Create Merge Request

You can manually create merge requests on the command line or CODING DevOps platform.

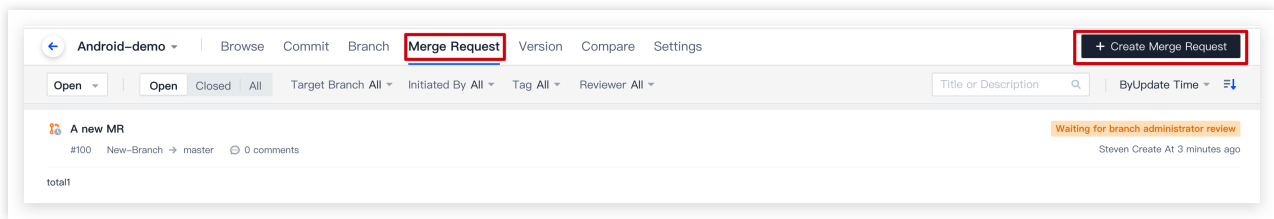
Create via the command line:



```
git push origin local-branch:mr/target-branch/local-branch
```

Create manually:

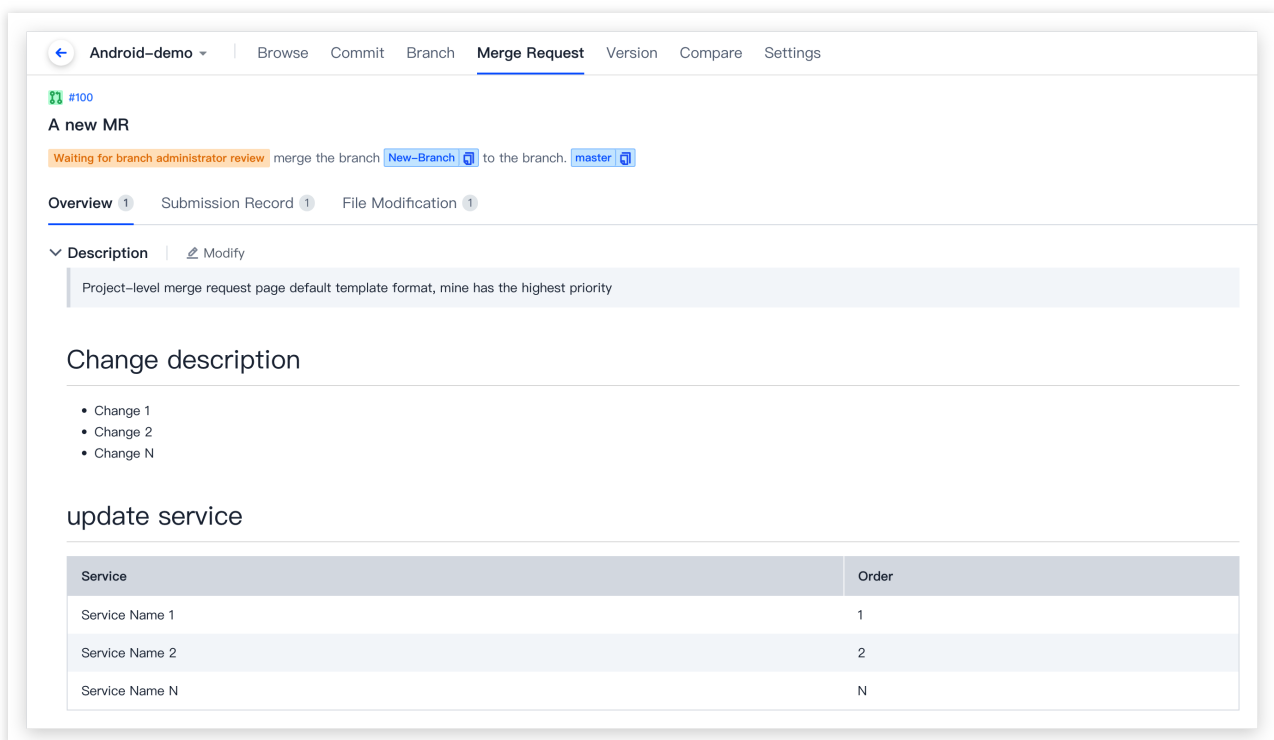
1. On the details page of a code repository, go to the **Merge Requests** tab and click **Create** in the upper-right corner.



2. Specify a source branch and target branch for the merge request. If no conflicts are found after the source branch is compared with the target branch, the system will indicate that they can be merged. You can view the differences between the files on the **File Changes** tab.

Note:

You can specify the default target branch for merge requests in [merge request settings](#).



3. Enter a title, a description, and associated resources for the merge request.

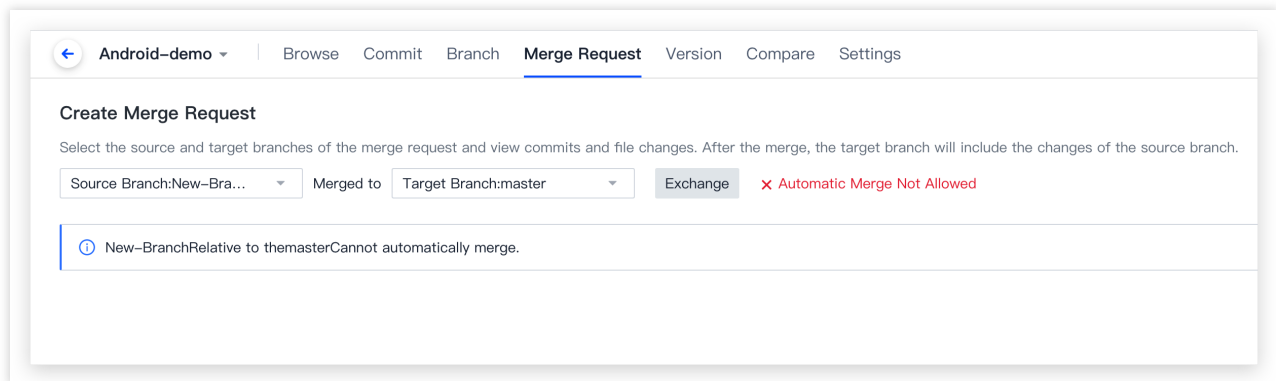
Note:

If the source branch is not ahead of the target branch, merge requests cannot be created.

Resolving Requests That Cannot Be Automatically Merged

If a conflict is found after the source branch is compared with the target branch, the system will indicate that they cannot be automatically merged. You can view the file changes on the **Compare** tab. Resolve the conflicts to continue

merging the branches.



For example, if a conflict is found when merging `branch-01` into `master`, you can switch to the `master` branch locally and run the command:



```
git merge branch-01
```

Find the file with conflicts. The conflicts will be highlighted in the file. You will be prompted to select which content to keep. Select the content you want to keep and save the file before committing it again, and then switch to the `branch-01` branch and enter the command:



```
git merge master
```

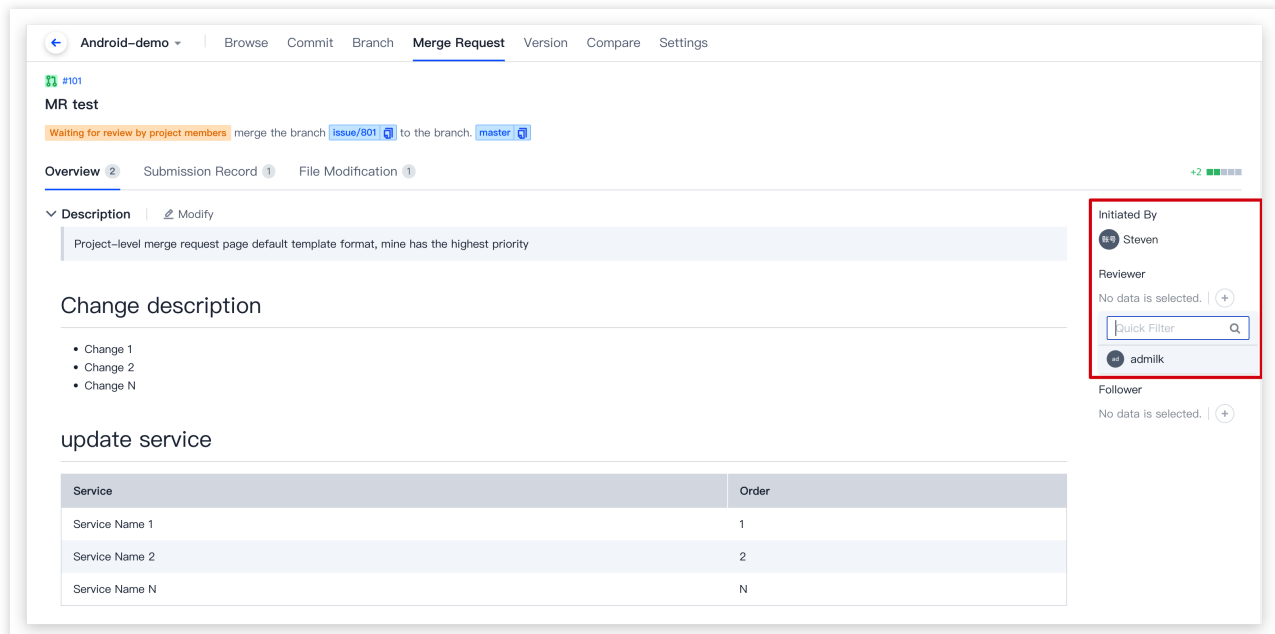
Then, push the modified code to the remote repository.

Initiate Review

When initiating a branch merge request, we recommend that you allow the relevant personnel to review the code to ensure code quality.

Note:

If the target branch of a merge request is a [protected branch](#), the branch admins will be added as reviewers by default. To change the settings, see [Branch Protection Rules](#).



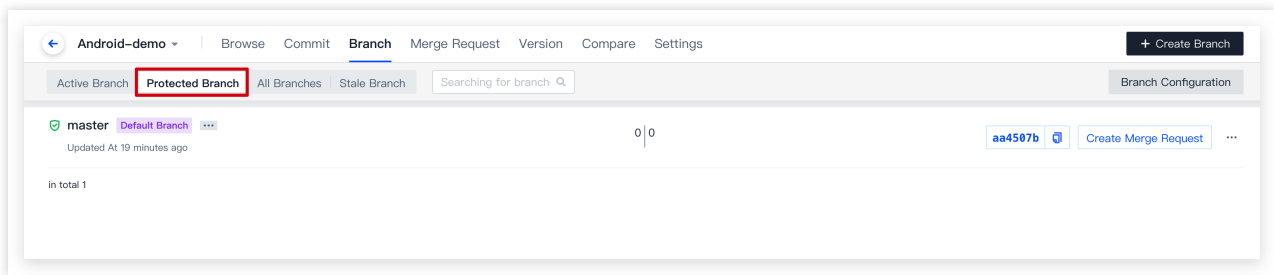
After the reviewers have completed the [merge request review](#), the review result will be shown on the details page of the merge request.

Check Merge Status

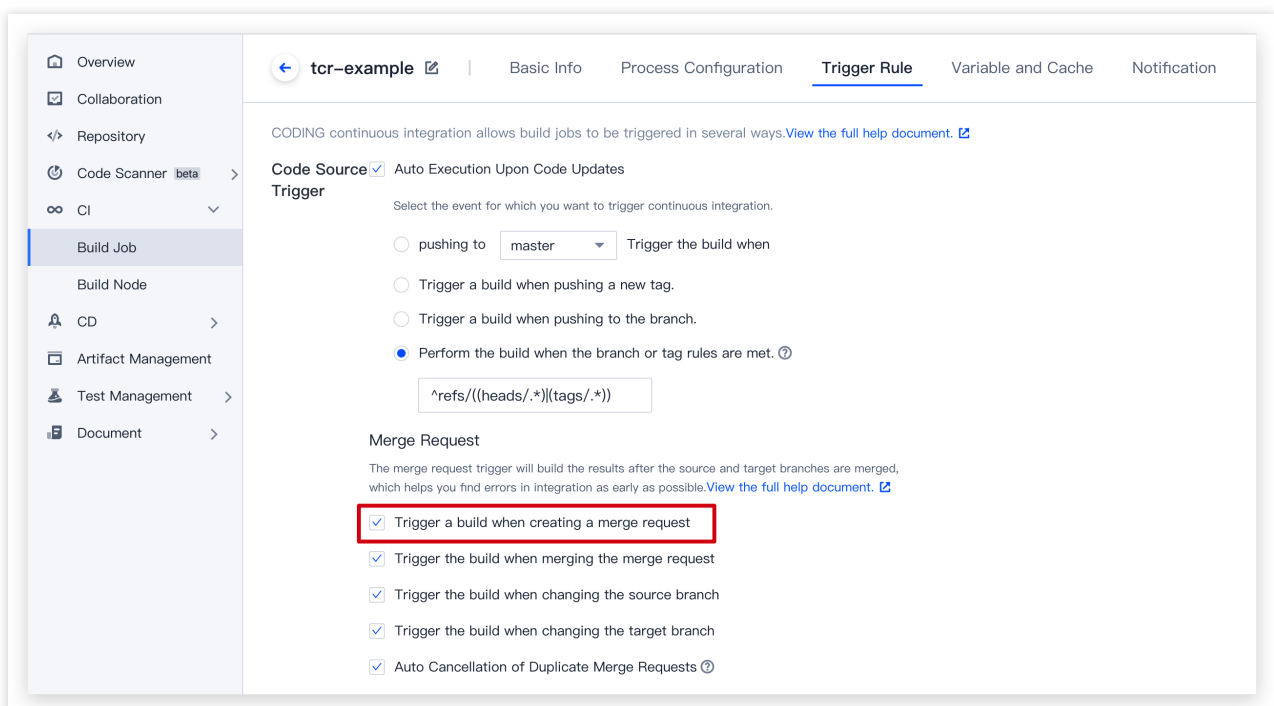
In addition to the common manual code reviews mentioned above, we provide an integrated code review solution with automated tool and CODING Continuous Integration. The code is scanned based on pre-defined rules. When there is an issue with the code quality, the code will not be merged. Only code that has passed checks by the automated tool can be merged, significantly improving the efficiency of code reviews.

Enable status check

Status checks can only be enabled for a [protected branch](#). After the option is selected, merging is allowed only after all status checks (CI tasks) have been run and passed.

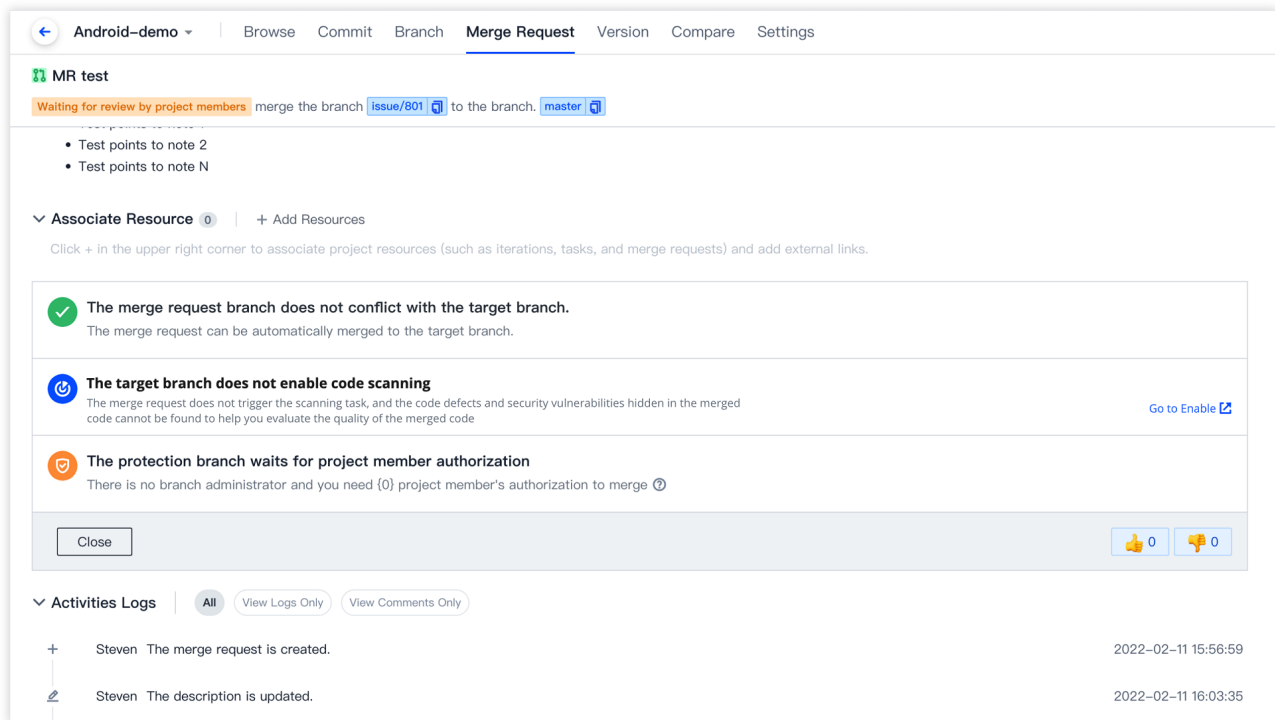


Select the CI trigger rule **Trigger Build on Merge Request** to trigger a build task after a merge request is created.



View status check result

After you have completed the above configuration, you can see the status of the merge check if the build task was correctly triggered. If your page is not similar to the following image, you may need to select **Trigger Build on Merge Request** in the CI build task.



You can click the Refresh button in the upper-right corner to get the latest status. If the check result is successful, a prompt at the bottom will indicate that the branches have been merged. If the result is failed, the merge will be rejected. Status checks can be:

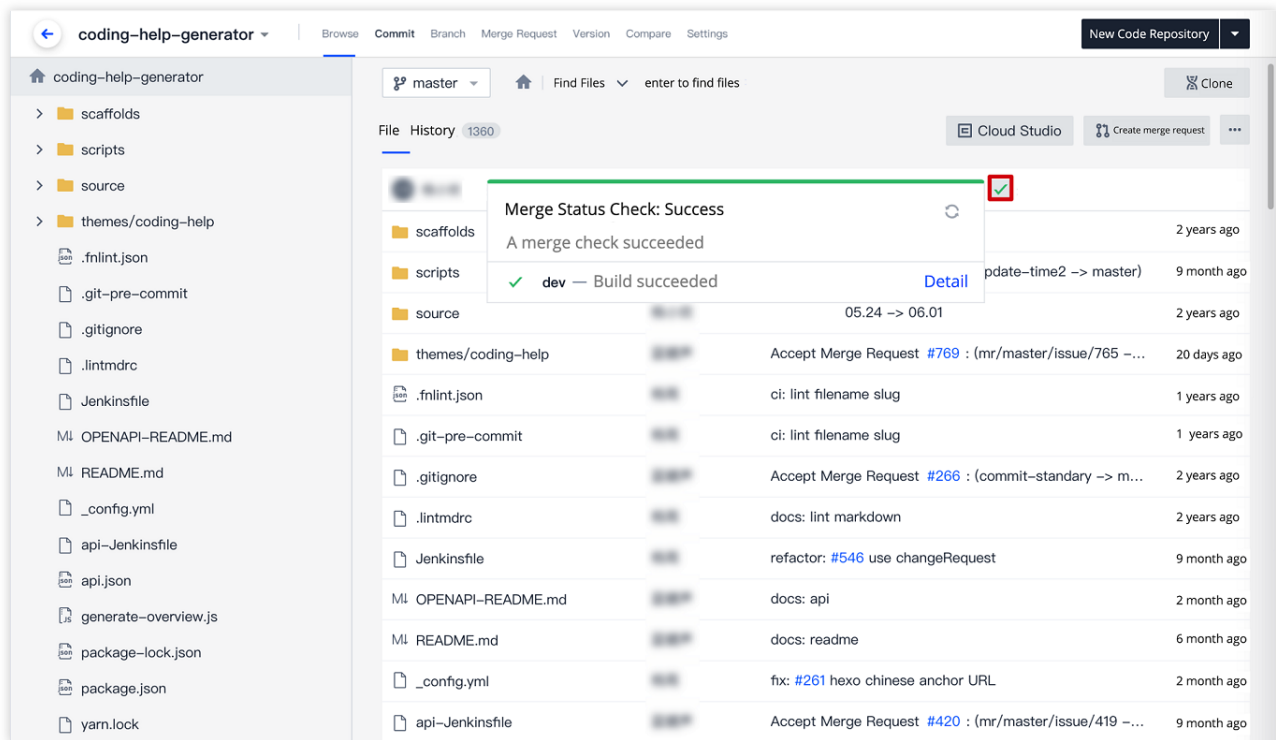
In progress: Wait for the build to finish.

Successful: The merge request can be merged.

Failed: Error occurred during the build process and the merge check failed. You can modify and push the code and trigger new build tasks until the build is completed successfully.

Exception: Exception occurred during the build process. You can try manually triggering it.

If multiple status checks are used, the branches can only be merged after all status checks are passed. You can also view status check processes in Browse Code, Commit History, and the branch list.



Confirm Merge

The target branch of the merge is a protected branch

If the initiator of the merge request is a branch admin, they can perform the merge on their own. If the initiator is an ordinary member, the merge can be completed only after it has passed a review by a branch admin.

The target branch of the merge is not a protected branch

The initiator can initiate and complete a branch merge without review or authorization.

Note:

To learn how to modify the default branch and set protected branches, see [Set the Default Branch](#) or [Set Protected Branches](#).

Delete source branch

When merging branches, select Delete Source Branch to delete the source branch after the merge.

Merge Current Request

Accept Merge Request #101: (issue/801 -> master)

Merge Request: MR test

Created By: @Steven

Accepted By: @Steven

URL: <https://straybirds.coding.net/p/coding-demo/d/Android-demo/git/merge/101>

☒ Delete Source Branch ☒ Fast-Forward Merge

Merge Branches Cancel

Fast-Forward merge

A merge commit record will be created by default during a non-fast-forward merge. If **Fast-Forward Merge** is selected, the remote repository will determine if the fast-forward rules are met. If the rules are met, this merge will not create a new merge commit record. If this mode is not selected, previous development records will be kept and a new merge record will be created during the merge. This option is equivalent to adding the `-ff` parameter when using `git merge .`

Note:

You can enable deletion of the source branch and fast-forward merge by default for merge requests in [merge request settings](#).

Review Merge Request

Last updated : 2023-12-25 17:08:18

This document describes how to review merge requests.

Open Project

1. Log in to the CODING Console and click the team domain name to go to CODING.
2. Click



in the upper-right corner to open the project list page and click a project icon to open the corresponding project.

3. In the menu on the left, select **Code Repositories > Merge Requests**.

Before creating a merge request or performing a merge, developers can update the title and description, add other project members as reviewers, and associate resources in the project (such as tasks, files, merge requests, and Wiki pages). You can also [automatically add reviewers](#) using the CI plugin.

Note:

If the target branch of a merge request is a protected branch and admins have been configured for the protected branch with **Automatically Add Branch Admin as Reviewer** enabled, the specified number of branch admins will be automatically added as reviewers.


Review Content

After a reviewer receives a review notification, they should generally focus on the following:

The title, description, and associated resources of the merge request can clearly describe the code modification. The reviewer can communicate and check with the initiator using comments.

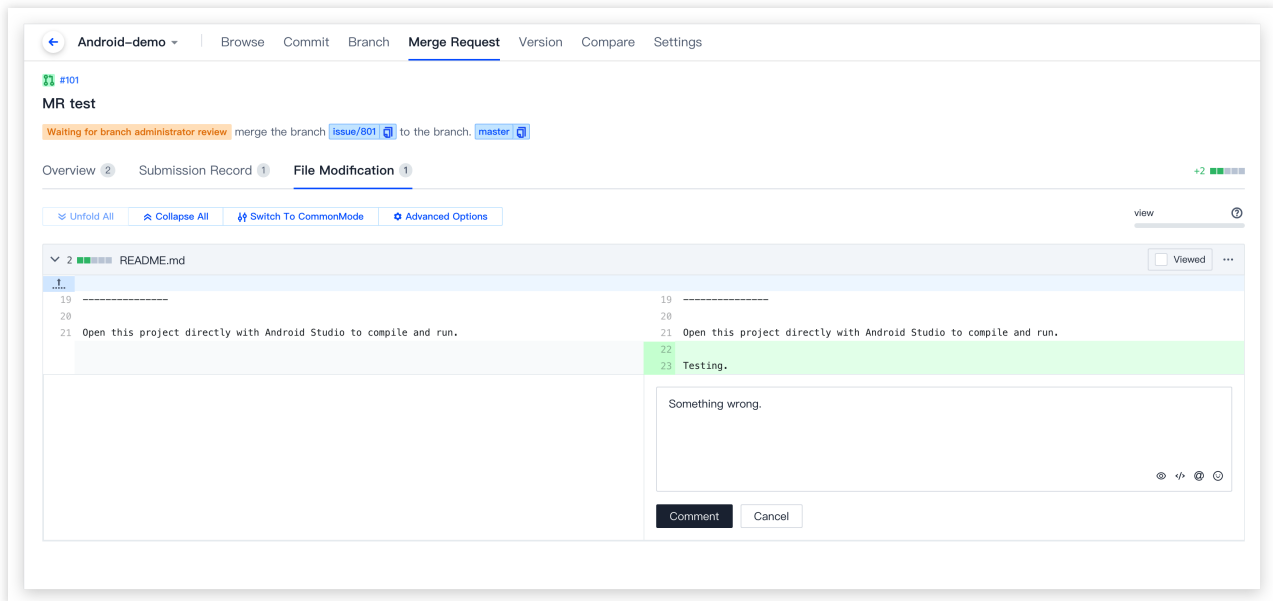
Review (comment) code at the line level for the committed file changes.

Start Review

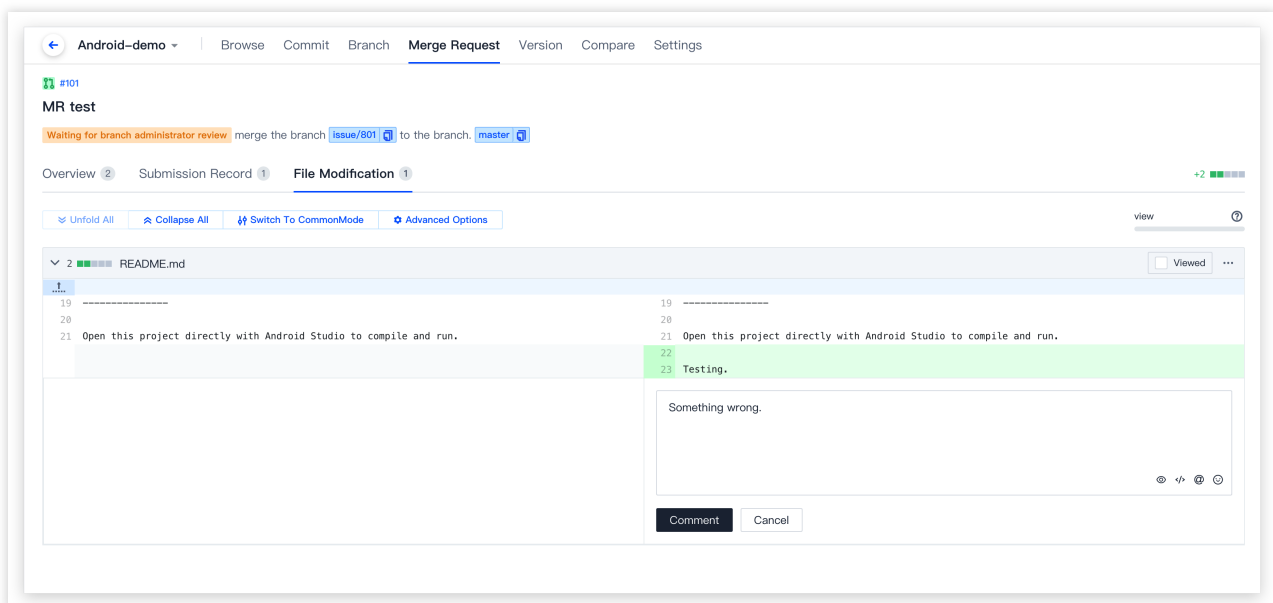
Code reviewers can comment on a code file line by line on the **File Changes** tab of **Merge Requests**. When you hover over a line in a code file, a plus sign  will appear. Click the plus sign to comment on the code. After entering a comment:

Click **Comment** to post the comment.

Click **Start Review** to post the comment and change the request status to **Reviewing**.



After the comment is posted, the comment and review status (if any) will be shown in the action log on the overview page of the merge request.

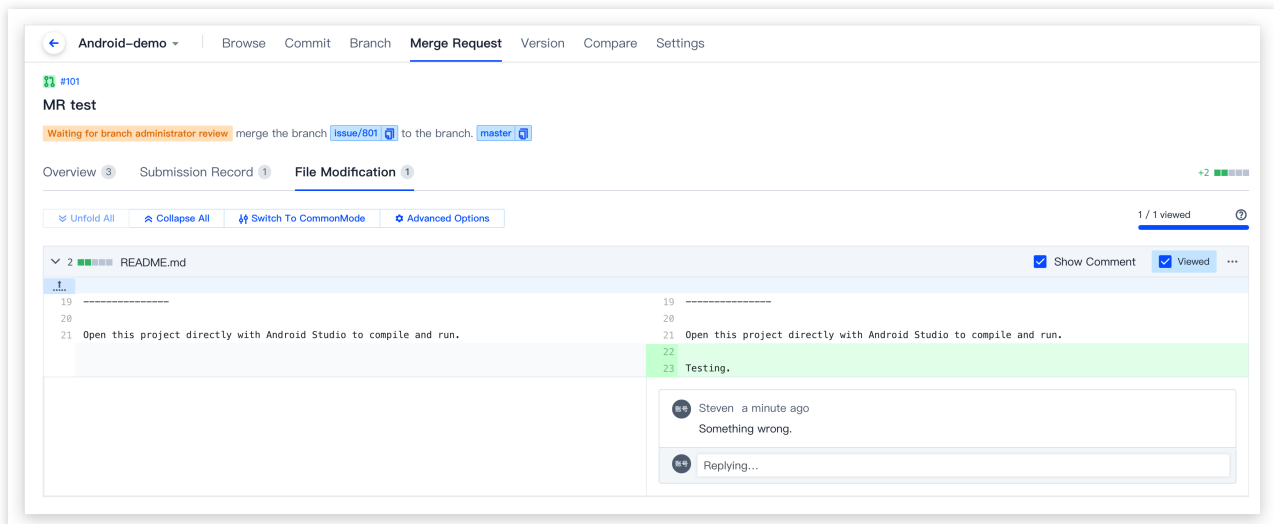


Track File Viewing Progress

When you need to review multiple code files, click *Viewed* after you have finished reviewing a file to mark your progress. The file viewing progress bar will be automatically updated.

Note:

Marking a file as **Viewed** does not affect the review status and only serves to track the file viewing progress. It only applies to the current user.



Complete Review

After you have reviewed all code files, click **Complete Review** in the upper-right corner to publish the review result and end the review.

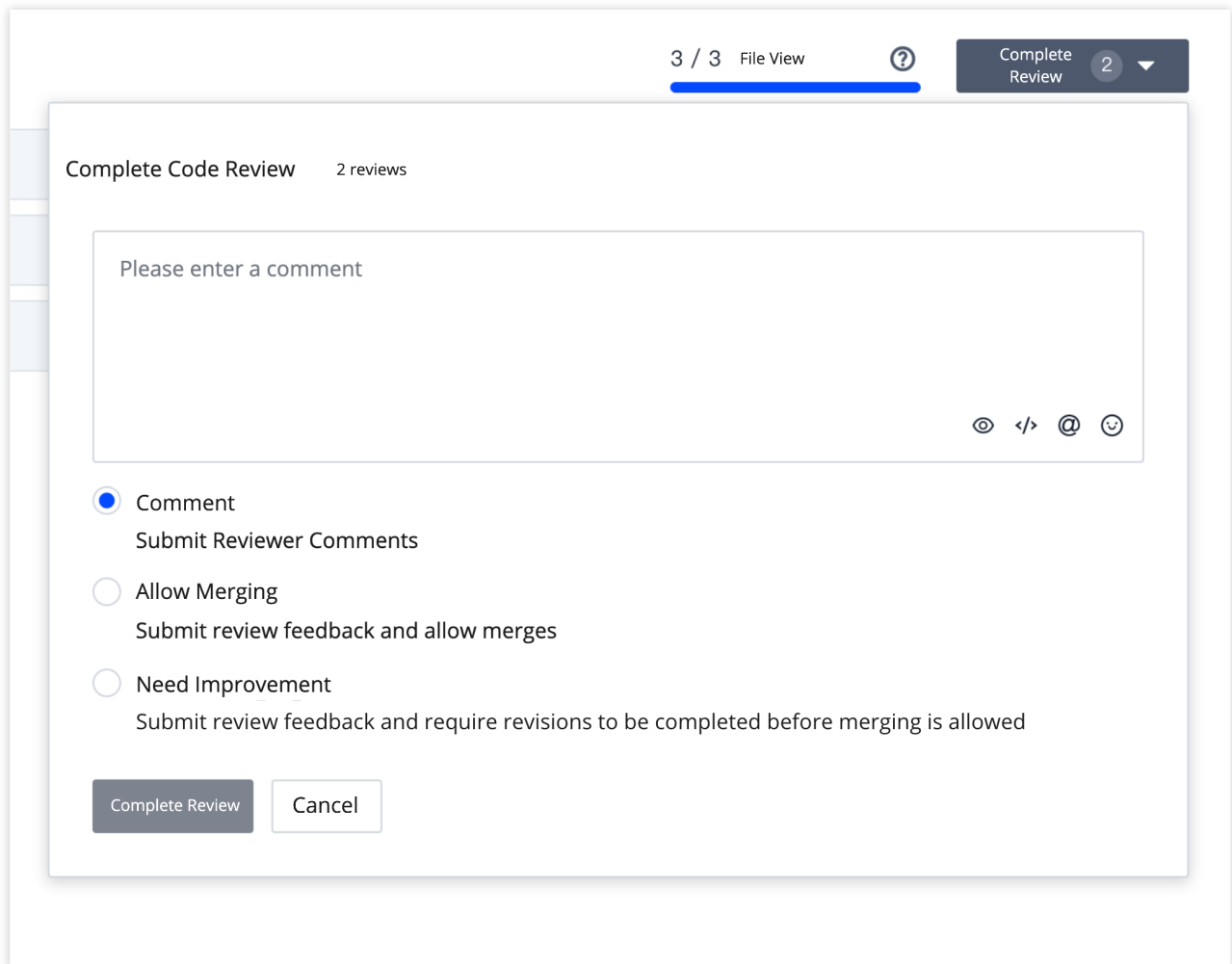
Note:

The number to the right of **Complete Review** indicates the total number of code comments on the current page.

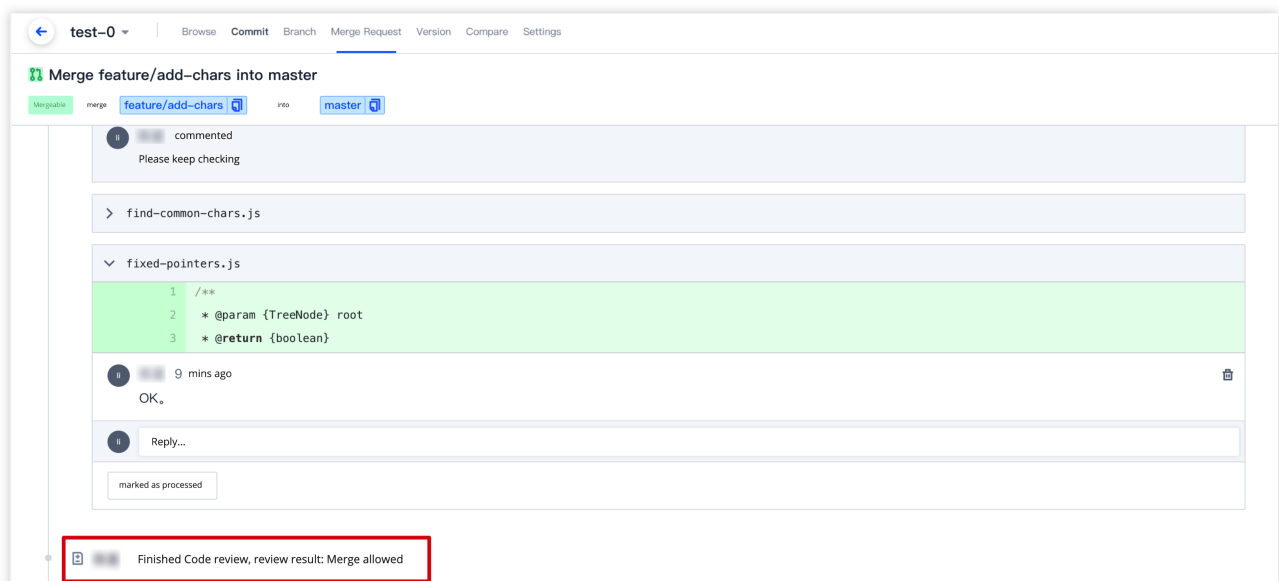
Comment: Comment is required.

Allow Merge: Comment is optional. The review result is published as **Allow Merge**.

Require Changes: Comment is optional. The review result is published as **Require Changes**.

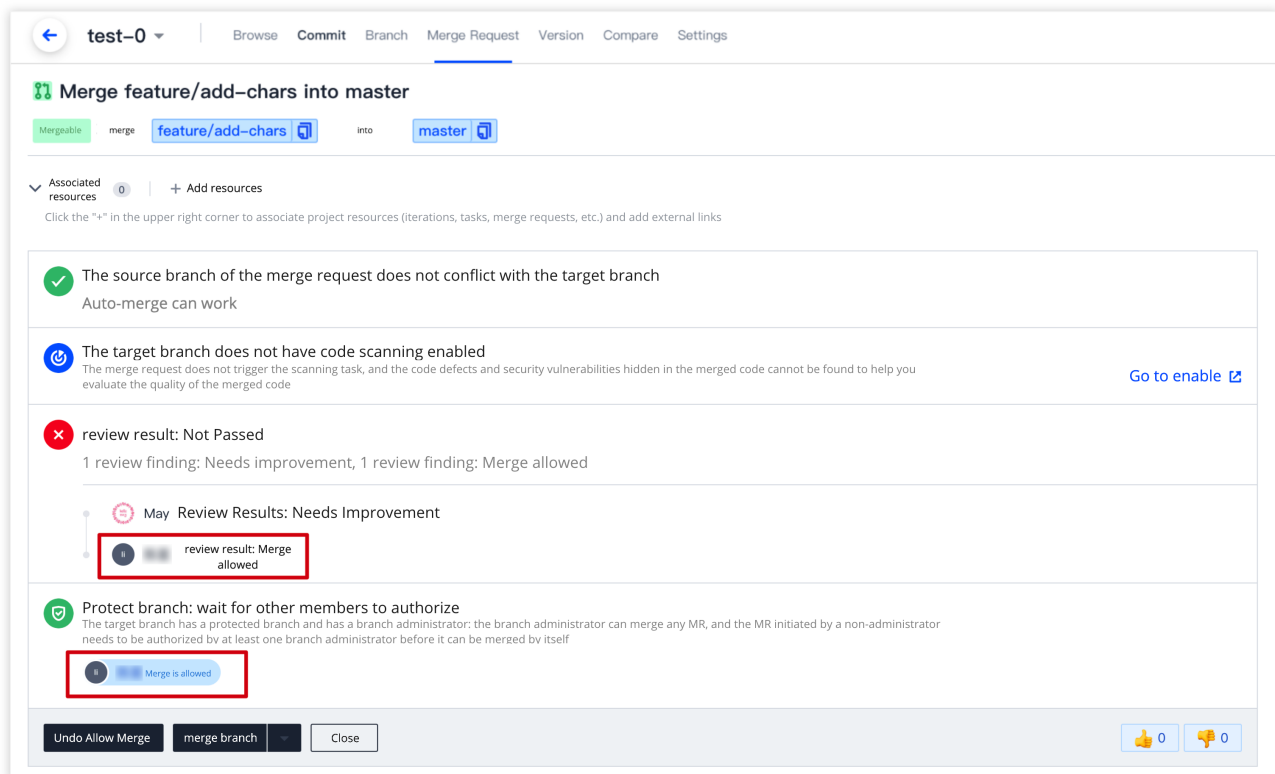


After a review is completed, the comment or review result will be shown in the action log on the overview page of the merge request.



If the target branch of a merge request is a protected branch, the review status of the merge request will be shown (if one of the results is **Require Changes**, the review status will be failed).

If the review result is **Allow Merge**, the result will be shown in the branch status as a tag, regardless of whether the target branch is a protected branch.



Versions and Tags

Manage Version Release

Last updated : 2023-12-25 17:08:18

This document describes how to use the version release feature in a code repository.

Open Project

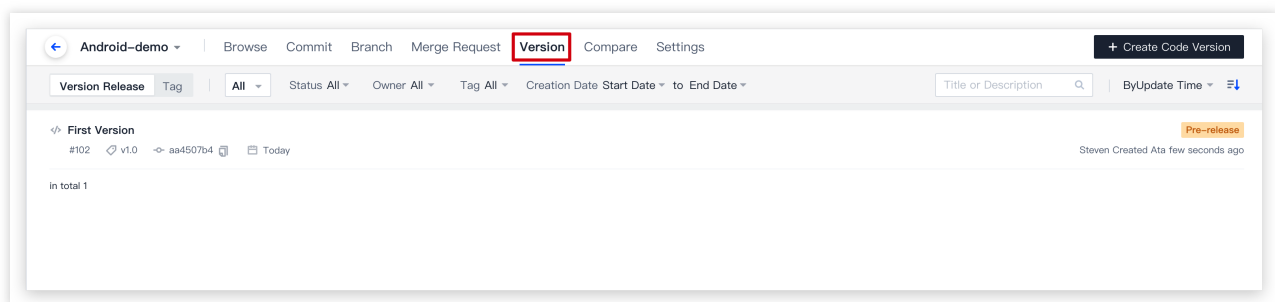
1. Log in to the CODING Console and click the team domain name to go to CODING.
2. Click



in the upper-right corner to open the project list page and click a project icon to open the corresponding project.

3. In the menu on the left, click **Code Repositories** > **Versions** to go to the version page.

In the code repository management list, click a specific code repository and go to the details page, and then click **Versions** > **Version Release** to go to the version release list.



The version release list displays code versions released in a project with their tag names and committed versions by descending order of creation time.

Create Code Version

1. On the version management page, click the Create Code Version button in the upper-right corner.

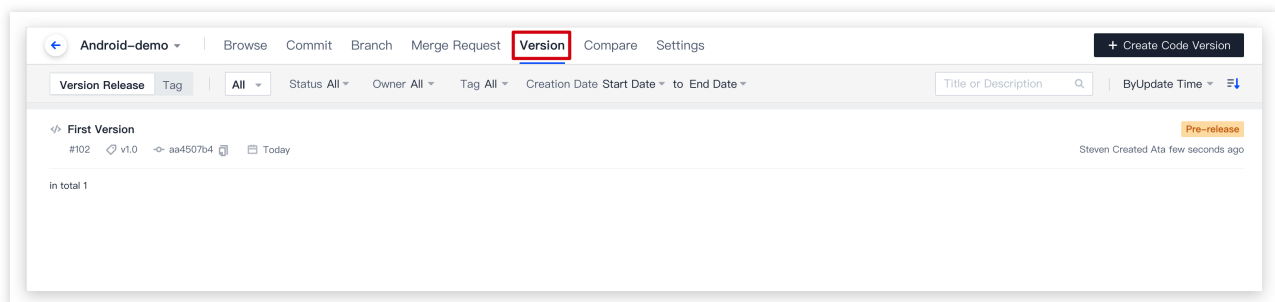
Note:

You can [set the default release branch](#) on the repository settings page.

2. Enter a tag version, release title, and version description. You can upload files smaller than 100 MB and associate resources in the project (such as tasks, files, Wiki pages, and merge requests). Only new tag names are allowed and you need to select a source for new tag versions (branch, tag, or revision number).

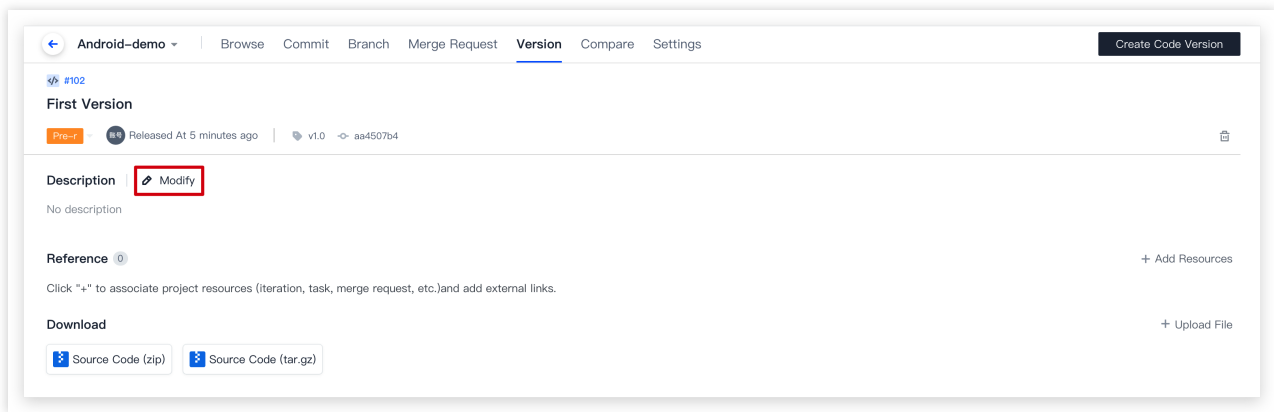
The screenshot shows the 'Create Code Version Release' form. At the top, there's a navigation bar with 'Android-demo' and tabs for 'Browse', 'Commit', 'Branch', 'Merge Request', 'Version' (selected), 'Compare', and 'Settings'. The form has three main sections: 'Tag Version' with a text input 'Please enter the label version' and a 'Create Source' dropdown set to 'master'; 'Version Release Title' with a text input 'Enter the version release title.'; and 'Description' with a rich text editor showing 'Demo Searching...' and a list of project resources on the left. A 'summary.' section on the right explains the version format (Major.Minor.Patch) and provides guidelines for incrementing versions.

3. After entering the above information, you can mark the version as a pre-release, and then create the code release. After the code version is created, the version list will be similar to the following image. The latest release is the latest official version release.



Edit Code Version

Click any code version in the version release list and go to the details page. The release creator or project admins can click the Edit Version Description button to edit all information except the tag name.

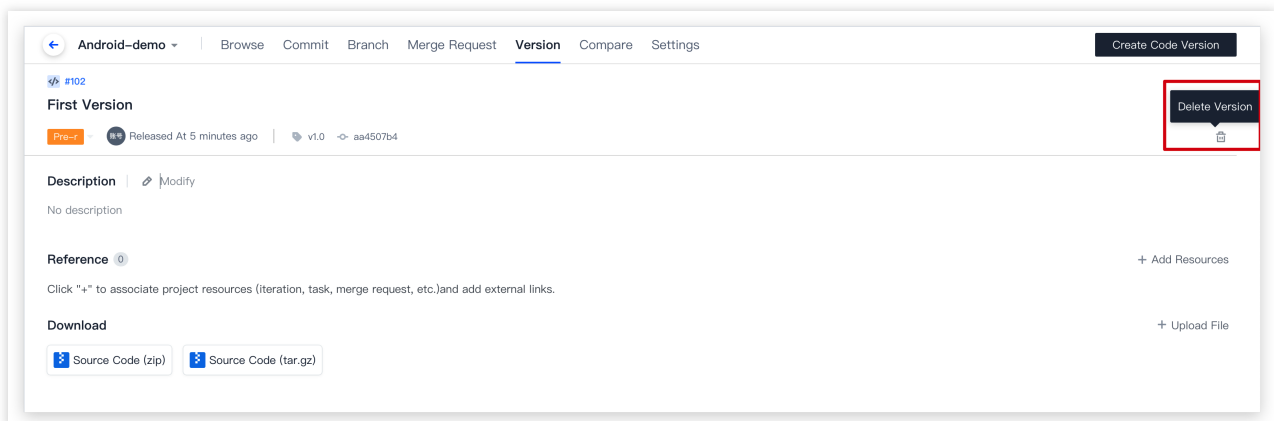


Delete Code Version

On the details page of a code version, the release creator or project admins can delete the version by clicking on the Delete icon.

Note:

You can also delete the linked version tags when deleting a version. You can only use or create a tag with the same name after you have deleted the version tag.



Set the Default Release Branch

Project admins can specify the default branch for version releases in **Settings > Version Release**.

Android-demo

Browse

Commit

Branch

Merge Request

Version

Compare

Settings

Repository Configuration

Basic settings

Deploy Public Key

File Locking

Branch Configuration

Access Configuration

Push Setting

Merge Request

Version Release

Code Tag

Warehouse security

Version Release

Default Branch of Version Release

master

Save

Search

Q

master

New-Branch

issue/801

branch by default when a version release is created.

Manage Version Tag

Last updated : 2023-12-25 17:08:18

This document describes how to manage version tags.

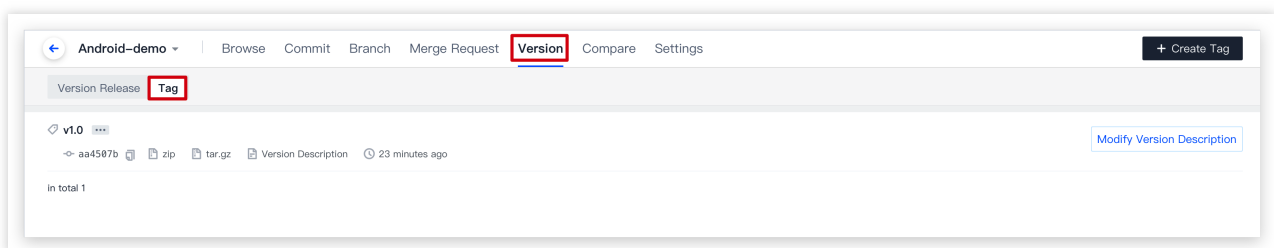
Open Project

1. Log in to the CODING Console and click the team domain name to go to CODING.
2. Click



in the upper-right corner to open the project list page and click a project icon to open the corresponding project.

3. Select **Code Repositories** in the menu on the left and click **Branches** to go to the branch management page. If a multi-branch development workflow is used, we recommend you set the master branch as a protected branch. Developers can create temporary develop branches and initiate merge requests for them. After continuous integration (CI) and code reviews, developers can merge the develop branch into the master branch. In the code repository management list, click a specific code repository to go to the details page, and then click **Versions > Tags** to go to the version tag management list.



All tags in the repository are displayed in the tag list by descending order of creation time. The tag names, tag descriptions, and versions are displayed in the tag list, which also provides entries to download versions as .zip and tar.gz files and delete tags. Click a tag name or version number to go to the details page of the code version.

Create Tag

In the tag management list, click **Create Tag** in the upper-right corner. Enter a tag name, select the code version (branch, tag, and revision number) for the tag, and enter a tag description to create a new tag.

Note:

You can [Set Protected Tags](#) on the repository settings page to standardize the tag operations of members.

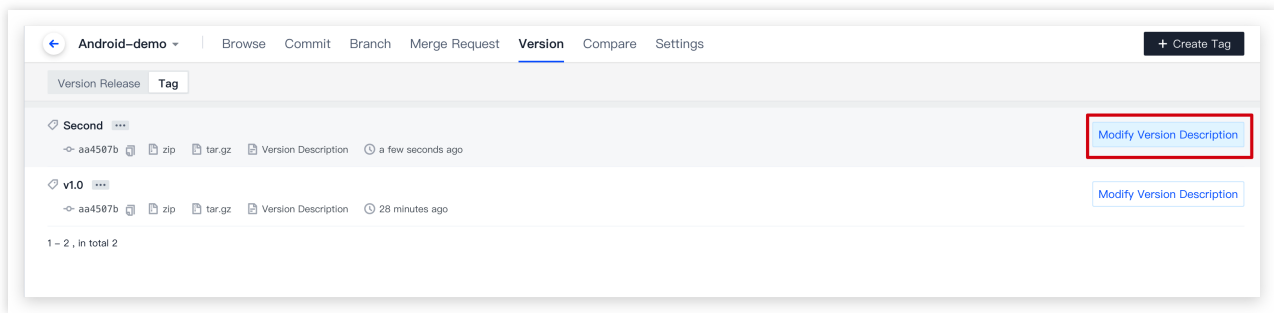
The screenshot shows the 'Create Tag' dialog box. At the top, there's a navigation bar with 'Android-demo' and tabs for 'Browse', 'Commit', 'Branch', 'Merge Request', 'Version' (selected), 'Compare', and 'Settings'. The dialog has two main sections: 'Tag Version' and 'Create Source'. The 'Tag Version' section has a text input field with the placeholder 'Please enter the label version:' and a dropdown menu showing 'master'. The 'Create Source' section has a dropdown menu showing 'master'. Below these is a 'Description' section with a text area for 'Add extra notes (Markdown is not supported)'. At the bottom, there are 'Confirm' and 'Cancel' buttons. On the right side of the dialog, there is a note about the version format: 'The version format is Major.Minor.Patch. The version number is incremented as follows: 1. Major version when you make incompatible API changes. 2. Minor version when you add functionality in a backwards compatible manner. 3. Patch version when you make backwards compatible bug fixes. Additional labels for pre-release and build metadata are available as extensions to the Major.Minor.Patch format.'

View Release Info of a Tag

If a code version is linked to the tag, click **Version Description** to view the release details. Click **Edit Version Description** to edit the release information.

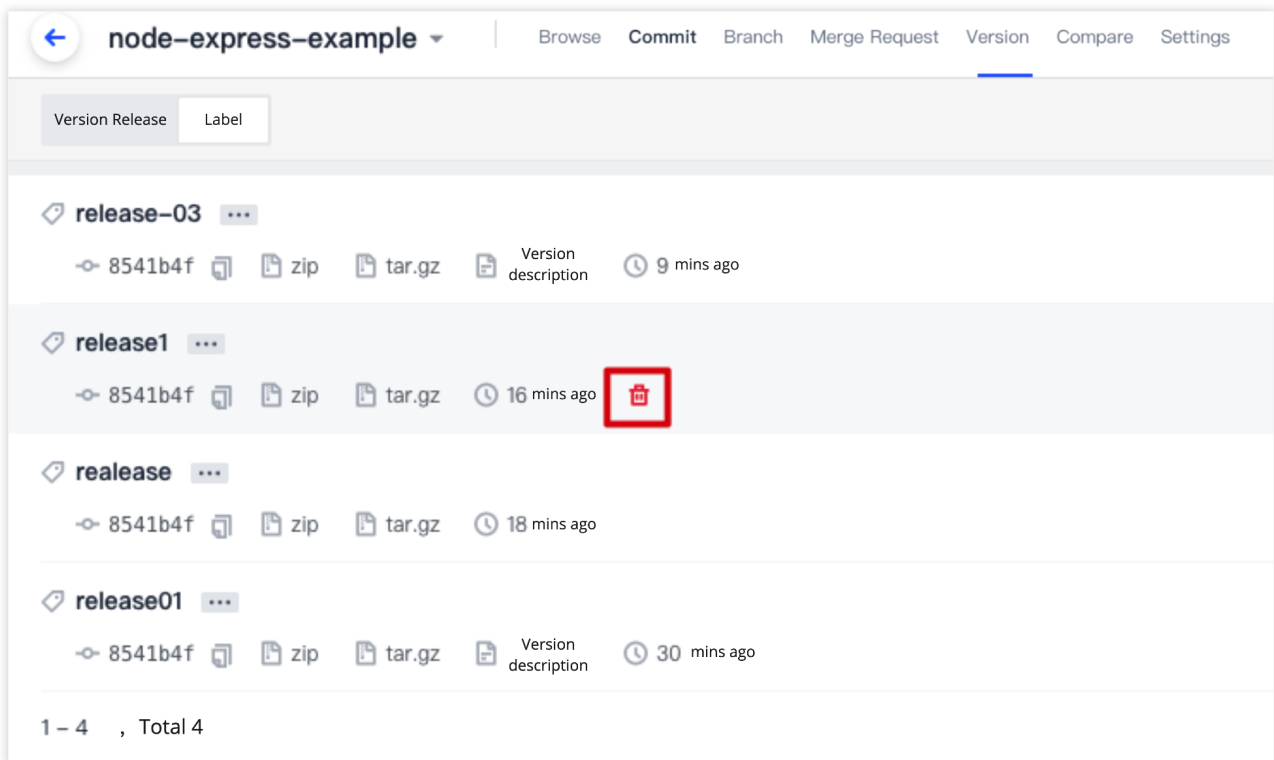
The screenshot shows the 'Version Release' tab in the 'Version' section of the 'Android-demo' repository. The tab is titled 'Version Release' and 'Tag'. Below the title, there are two release entries. The first entry is 'Second' with a dropdown menu. It shows a commit hash 'aa4507b', file icons for 'zip' and 'tar.gz', a blue button labeled 'Version Description' (highlighted with a red box), and a timestamp 'a few seconds ago'. The second entry is 'v1.0' with a dropdown menu. It shows a commit hash 'aa4507b', file icons for 'zip' and 'tar.gz', a blue button labeled 'Version Description', and a timestamp '28 minutes ago'. At the bottom, it says '1 - 2 , in total 2'.

If a tag is not linked to any release, you can click **Create Version Description** to create a release for the tag.



Delete Tag

Only code tags not linked to a release can be deleted on the **Tags** page by the tag creator or admin.

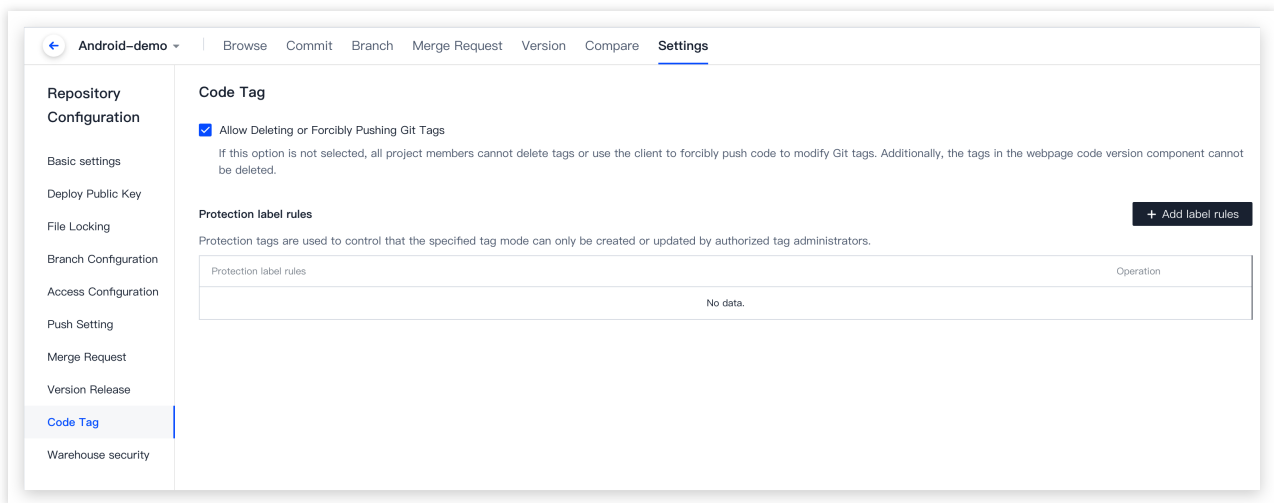


Note:

If a tag is associated to a release, the tag can only be deleted by deleting the linked version in **Version Release**.

Allow Deletion and Force Push of Git Tags

Project admins can check the checkbox to allow deletion and force push of Git tags in **Settings > Code Tags**. If this is disabled, no project member can delete Git tags or force push Git tags for modification, and tags cannot be deleted on the webpage.



Set Protected Tags

Protected tags are used to standardize the creation, update, or deletion of tags by specific members. After protected tag is enabled, only configured tag admins can create tags that match the tag rule. If `*-release` is set as a branch protection rule, non-admins will be prompted with the following when pushing the tag `xxx-release` via Git:

```
git push --tag origin xxx-release
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
remote: [err=32] You have no permission to update protected tag (refs/tags/xxx-release).
remote: (refs/tags/xxx-release) , https://coding.net/help/doc/git/git-branch.html#
remote: error: nook declined to update refs/tags/xxx-release
To https://e.coding.net/vcs-test/coding-demo/coding-demo.git
! [remote rejected] xxx-release -> xxx-release (hook declined)
error: failed to push some refs to 'https://e.coding.net/vcs-test/coding-demo/coding-demo.git'
```

They will also be unable to create tags or versions in Coding for Web.

Create Code Version Release

Tag Version ★

xxx-release

Version Release Title

Enter the version release title.

Description

Write

Please enter a description here (Markdown is supported)

Create Source *

master ▼

own is supported

 Cl

All file formats

for the release.

summary.

The version format is Major.Minor.Patch. The version number is incremented as follows:

1. Major version when you make incompatible API changes.
2. Minor version when you add functionality in a backwards compatible manner.
3. Patch version when you make backwards compatible bug fixes.

Additional labels for pre-release and build metadata are available as extensions to the Major.Minor.Patch format.

A team uses tags as triggers for CI builds. In other words, pushing tags such as v1.0-release is used as a release command in production branches.

Protected tags can be used to only allow the tag admins to create these tags for release and keep the versions organized.

Code Repository Security

Inspect Repository Security Risks

Last updated : 2023-12-25 17:08:18

This document describes how to check for security risks in a code repository.

Open Project

1. Log in to the CODING Console and click the team domain name to go to CODING.
2. Click



in the upper-right corner to open the project list page and click a project icon to open the corresponding project.

3. In the menu on the left, click **Code Repositories** > **Settings** to go to the repository security page.

Project admins can view existing security risks in **Settings** > **Repository Security**.

The screenshot shows the 'Warehouse security' settings page in the CODING Console. The left sidebar contains a menu with 'Warehouse security' selected. The main content area displays three security risks:

- Submitter and submit author check for Git submissions are not turned on**: A warning icon indicates that mailboxes are not verified. A 'Go open' button is present.
- GPG public key is not uploaded**: A warning icon indicates that the GPG key is missing. A 'Go upload' button is present.
- Protecting branch Master has the following risks**: A warning icon indicates that code owner review is not turned on. A 'To set up' button is present.

The following checks on code repositories are available:

Whether the check of the Git committer and author is enabled.

Whether a GPG public key has been uploaded.

Whether a protected branch has been set. Whether branch admins have been set and review by the code owner has been enabled for protected branches.

Note:

To keep your repository secure, we recommend you configure the relevant features with reference to [Push Settings](#), [Using GPG to Sign Commit Records](#), and [Protected Branch](#).

Set Repository Access Method

Last updated : 2023-12-25 17:08:18

This document describes how to set the access mode for a repository.

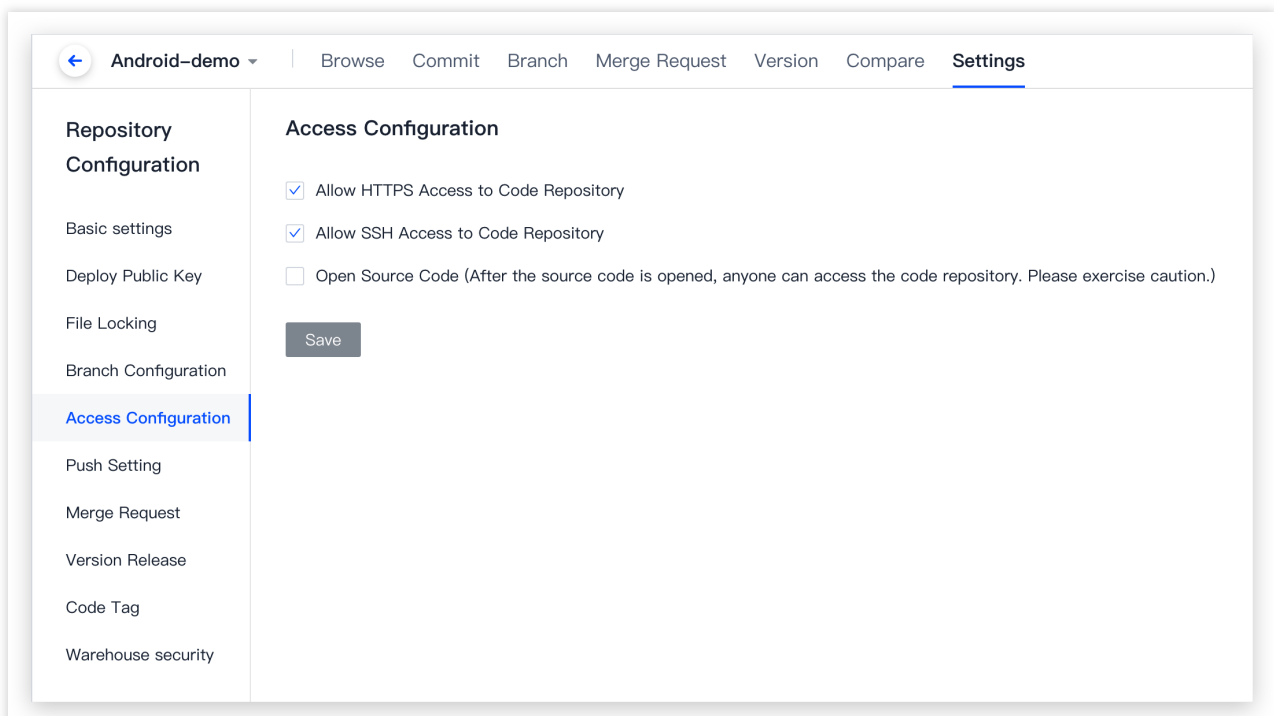
Open Project

1. Log in to the CODING Console and click the team domain name to go to CODING.
2. Click



in the upper-right corner to open the project list page and click a project icon to open the corresponding project.

3. In the menu on the left, select **Code Repositories > Settings** to go to the access settings page of the repository. Project admins can select whether to make the code repository public and whether to allow access to the code repository via HTTPS or SSH in **Settings > Access Settings**.



Sign Git Commits with GPG

Last updated : 2023-12-25 17:08:18

This document describes how to sign commits with GPG.

Open Project

1. Log in to the CODING Console and click the team domain name to go to CODING.
2. Click



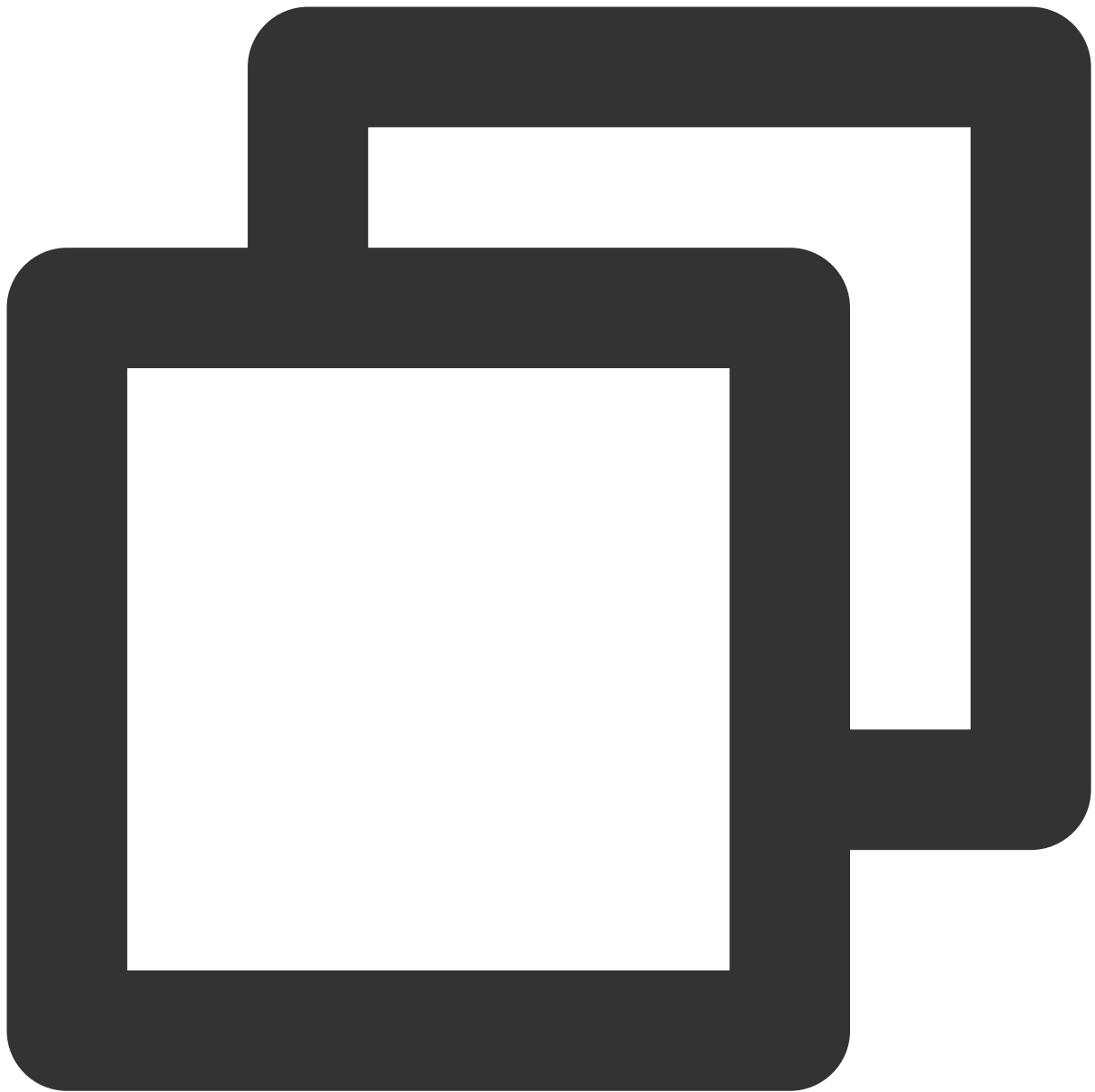
in the upper-right corner to open the project list page and click a project icon to open the corresponding project. CODING allows you to use GPG for Git commit signature verification. Verified commits will be tagged with **Verified**, which ensures that code is committed from reliable sources and enhances code security.

To sign Git commits with GPG:

- Step 1: [Generate a GPG key pair](#)
- Step 2: [Add a GPG public key in your personal account settings](#)
- Step 3: [Associate with the local Git repository](#)
- Step 4: [Sign Git commits](#)
- Step 5: [Verify signatures](#)

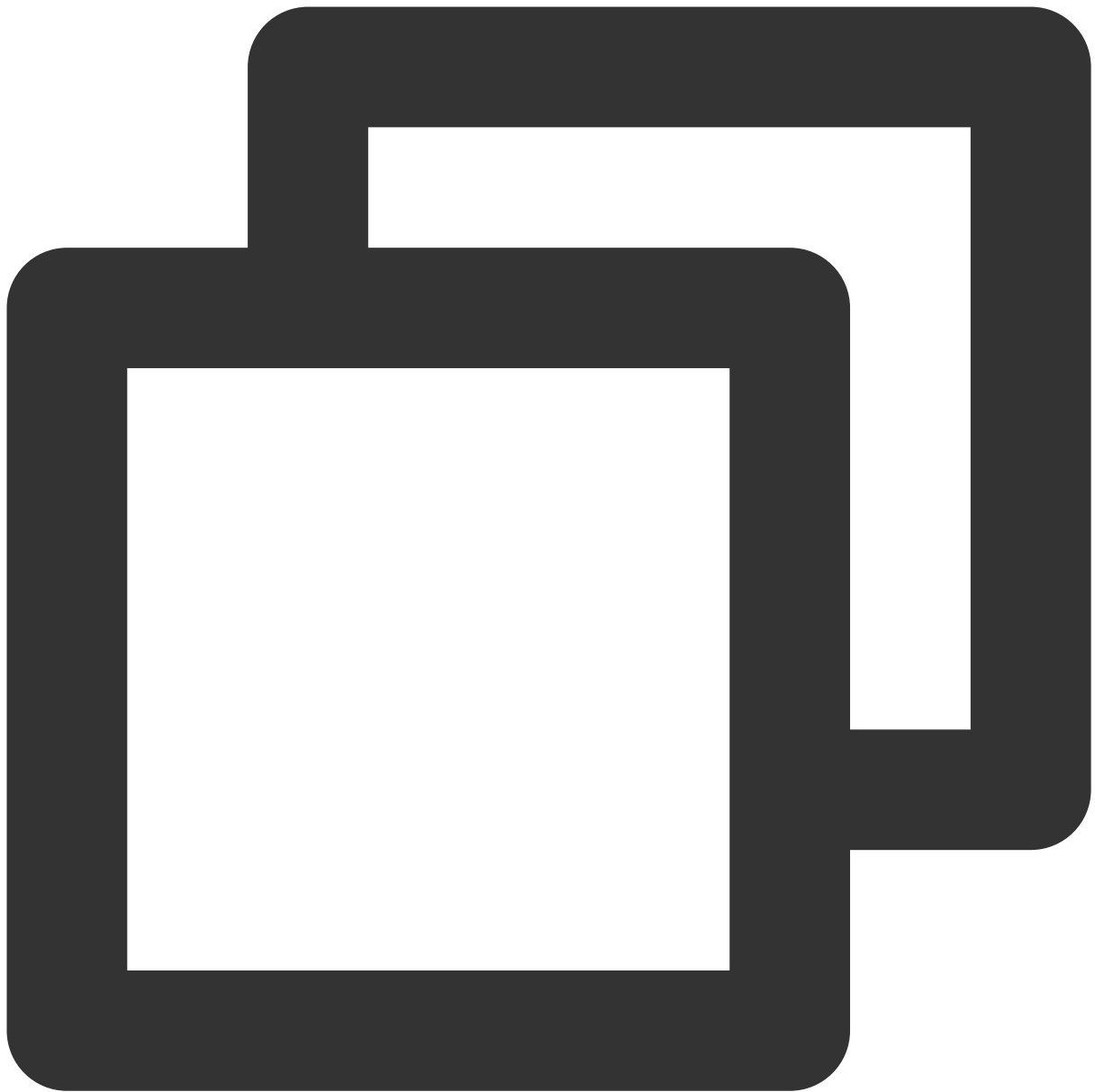
Step 1: Generate a GPG Key Pair

1. [Download](#) and install GPG. If you are using macOS, use the brew package management tool to run the following command:



```
brew install gpg
```

2. Run the following command to generate a GPG key pair (public key/private key):



```
gpg --full-gen-key
```

Note:

In certain scenarios, such as if you are using Windows Gpg4win or other macOS versions, use the `gpg --gen-key` command to generate a key pair.

As the command is interactive, you need to specify the algorithm, validity period, your real name and email address, a password, etc.

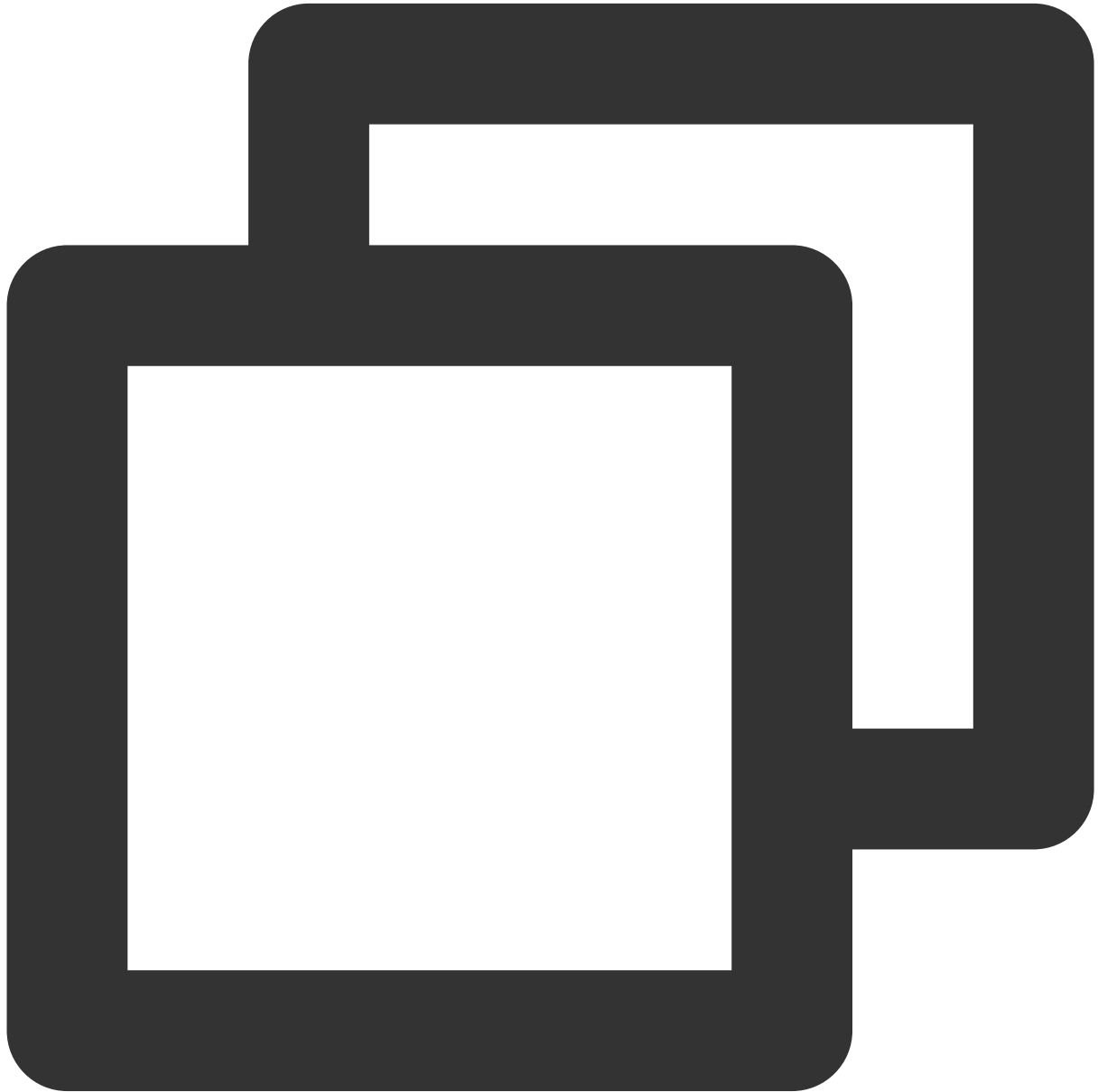
Key type: Select the key type or press `Enter` to select the default type (RSA and RSA).

Elliptic curve: Press `Enter` to select the default `Curve 25519` .

Validity: Specify the validity period of the key as needed or press `Enter` to select the default 'Never expire'.

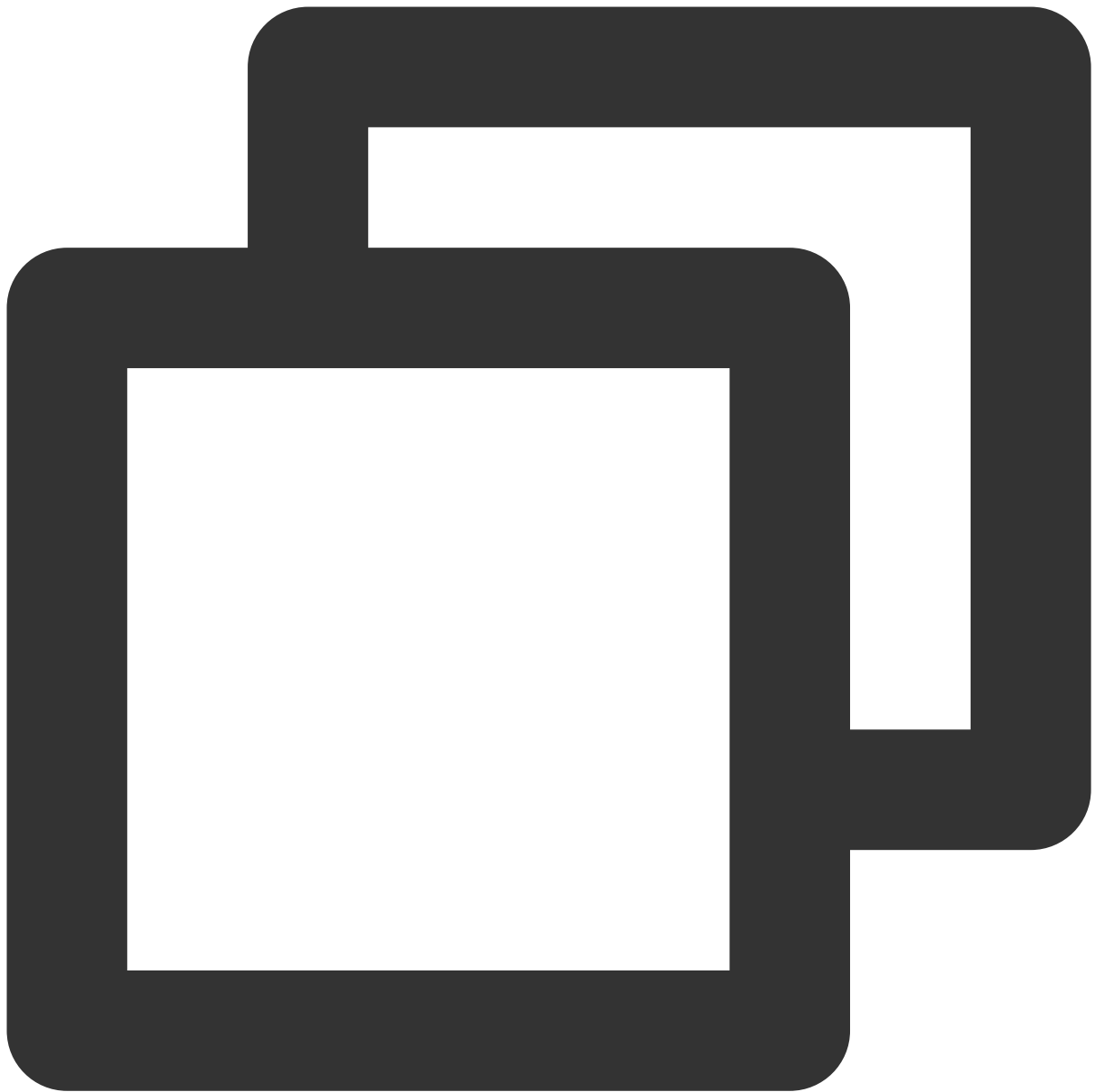
Email address: Enter the email address configured in your CODING account.

3. Run the following command to list the GPG key you just created (replace "your_email" with the email address entered in Step 3):



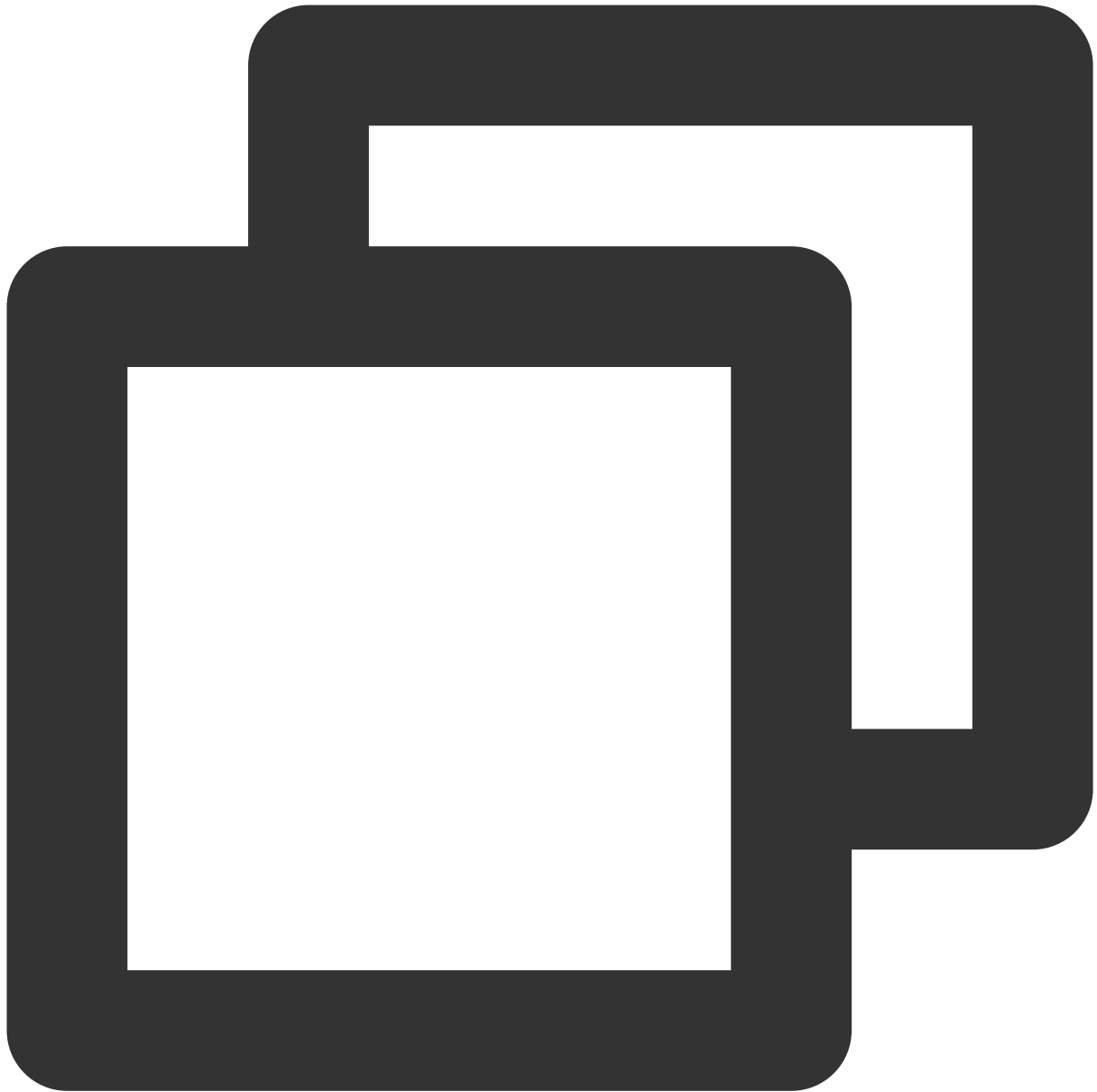
```
gpg --list-secret-keys --keyid-format LONG "your_email"
```

4. Copy the GPG key ID that starts with sec. In the following example, copy 4AEA00A342C24CA3:



```
sec    ed25519/4AEA00A342C24CA3 2021-09-14 [SC]
       6DE3507E82DEB6E8828FAAC34AEA00A342C24BD4
uid          [ ultimate ] your_name "your_email"
ssb    cv25519/812B586FD245B560 2021-09-14 [E]
```

5.Export the public key of that ID (using the above ID as an example):

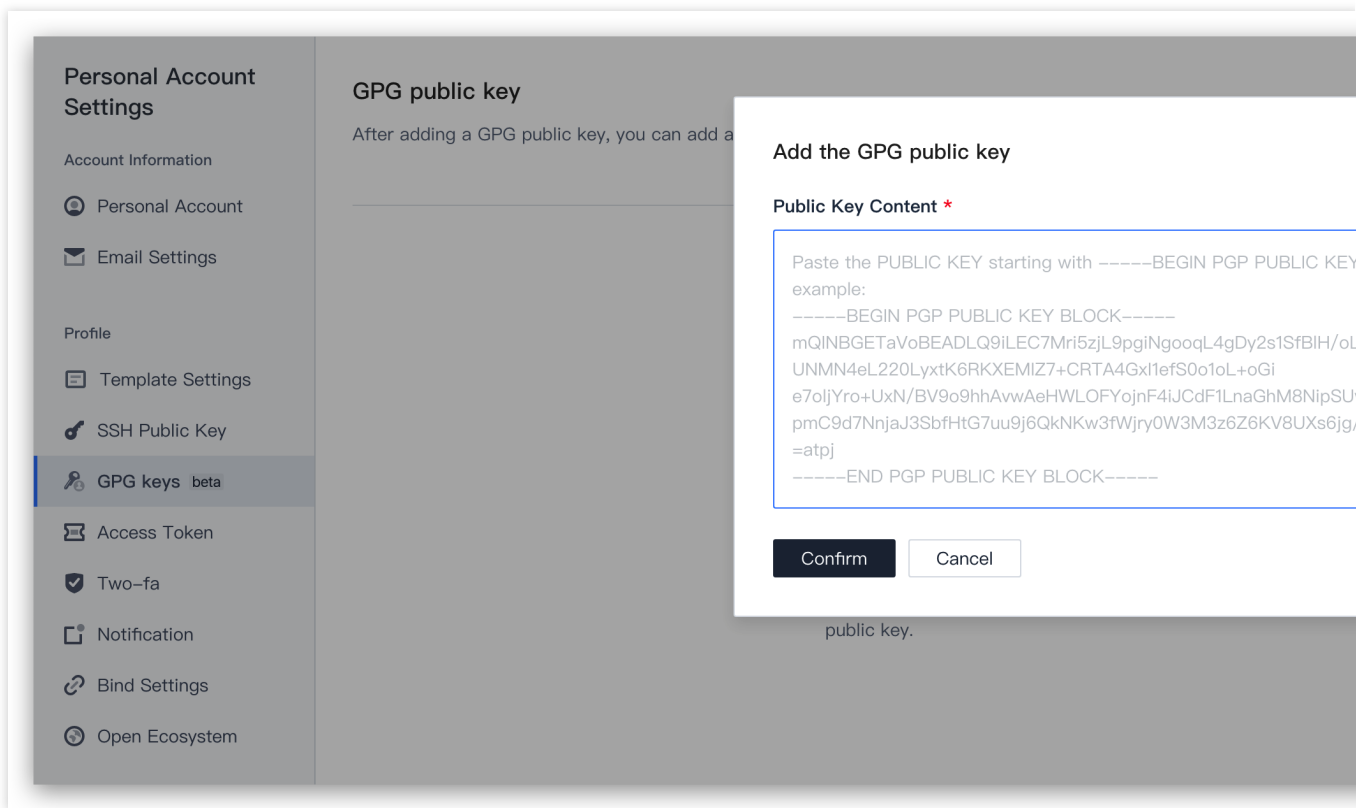


```
gpg --armor --export 4AEA00A342C24CA3
```

6. After the public key is generated, [add it to your CODING account](#).

Step 2: Add a Public Key in Your Personal Account Settings


1. After you have logged in to CODING, click your profile photo in the upper-right corner and select **Personal Account Settings**.
2. In the navigation bar on the left, select **GPG Public Key** to go to the public key management page.
3. Click **Add Public Key** and paste the exported GPG public key in the text box, and then click OK.



After the public key is added, the verification status of the email address, key ID, and subkey will be shown.

GPG Public Key

After adding the GPG public key, you can add a GPG signature to the commit

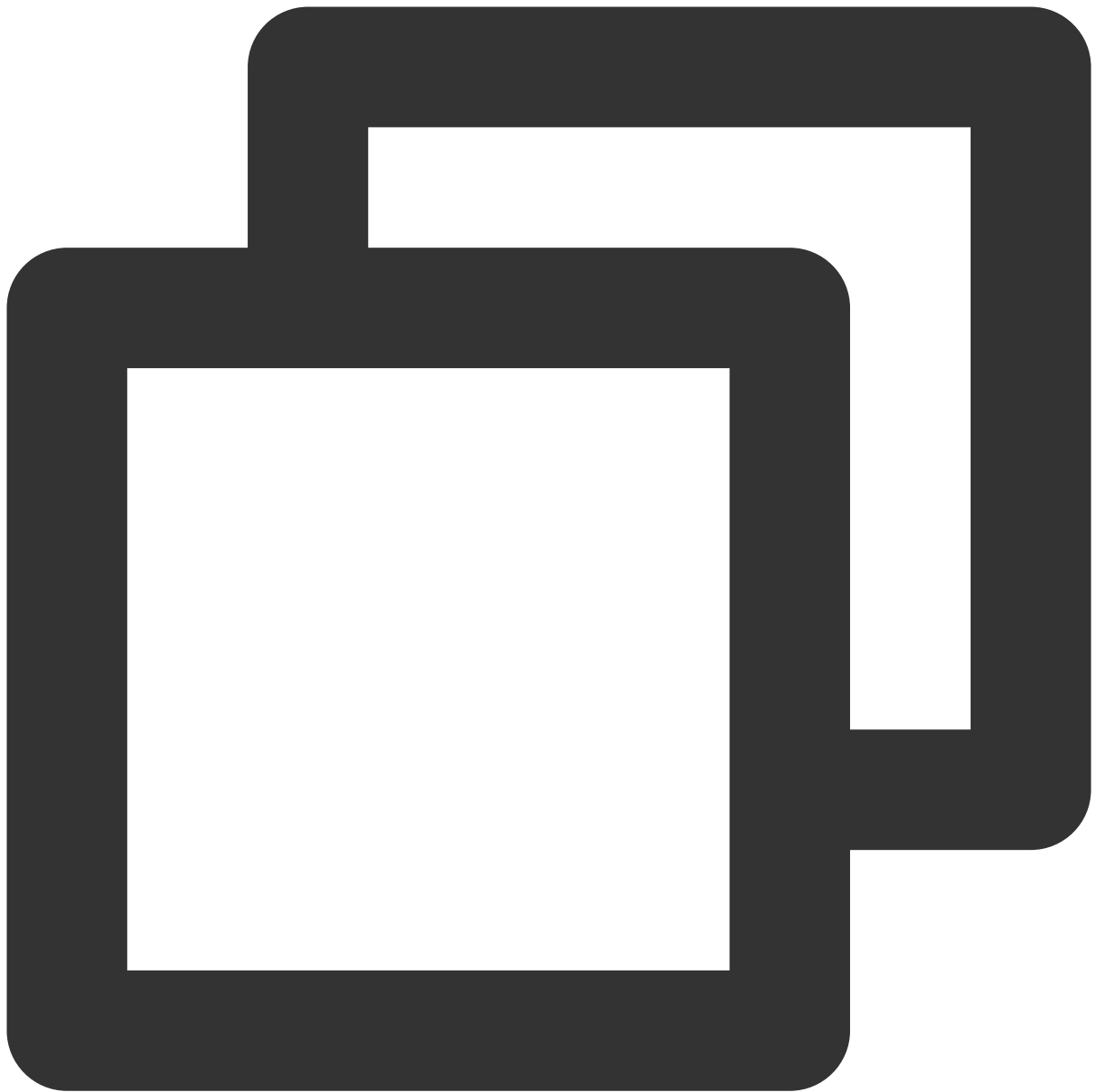
	Email Address	<input type="text"/>	Verified
	Key ID	<input type="text"/>	
	subkey	<input type="text"/>	
	Added at	2021-09-14	

Note:

If the status of the email address is Not Verified, the email address is not configured in the CODING account. Add the email address in **Personal Account Settings > Email Settings**.

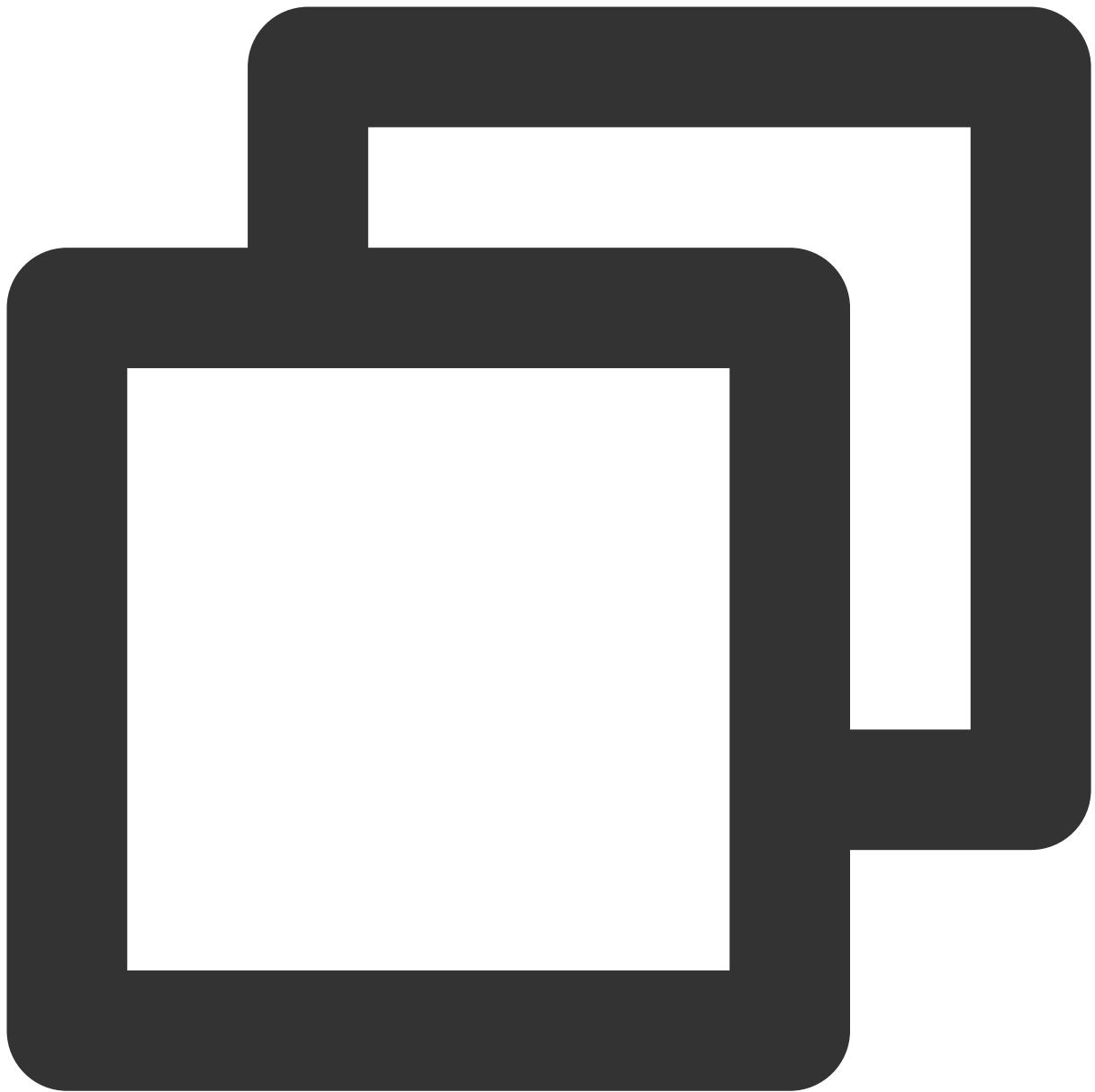
Step 3: Associate with the Local Git Repository

1. Run the following command to list the GPG key you created (replace "your_email" with the email address entered when generating the key):



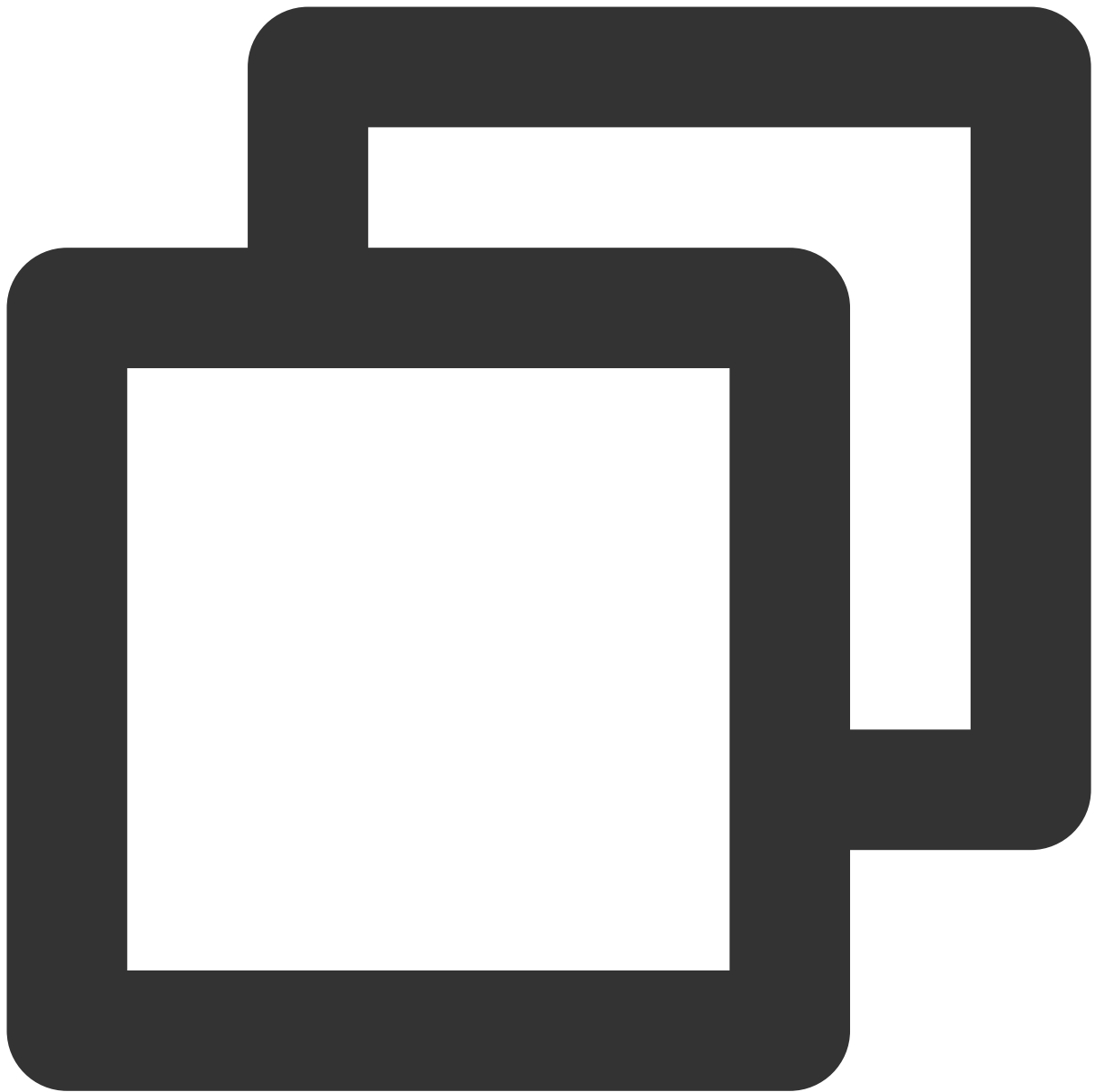
```
gpg --list-secret-keys --keyid-format LONG "your_email"
```

2. Copy the GPG key ID that starts with `sec` . In the following example, copy `4AEA00A342C24CA3` :



```
sec    ed25519/4AEA00A342C24CA3 2021-09-14 [SC]
       6DE3507E82DEB6E8828FAAC34AEA00A342C24BD4
uid          [ ultimate ] your_name "your_email"
ssb    cv25519/812B586FD245B560 2021-09-14 [E]
```

3. Configure the key in the local Git repository to sign the commits with it:



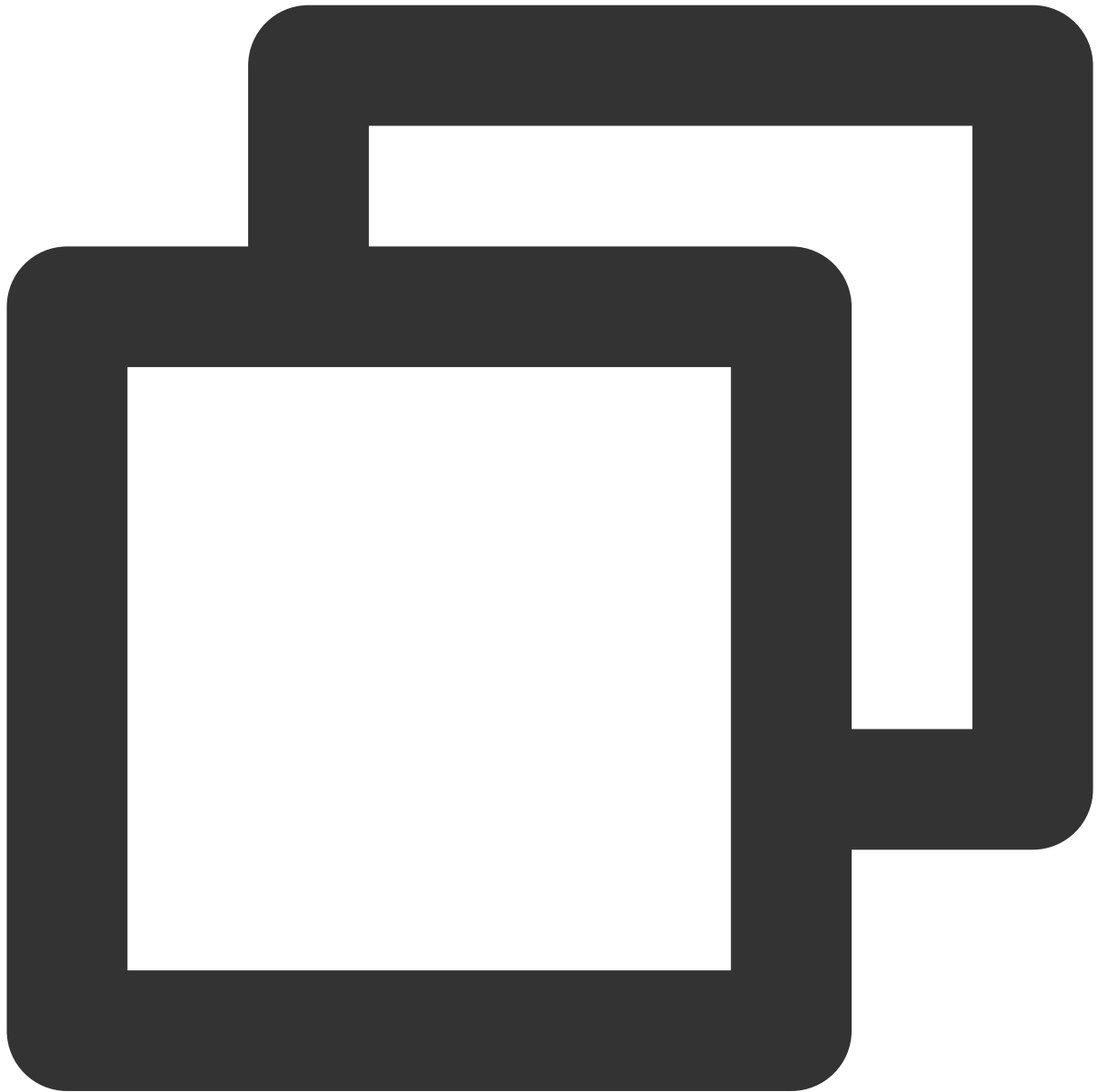
```
git config --global user.signingkey 4AEA00A342C24CA3
```

You have now successfully associated the GPG key with the local Git repository. After modifying code locally, you can sign your Git commits to verify the committer.

Step 4: Sign Git Commits

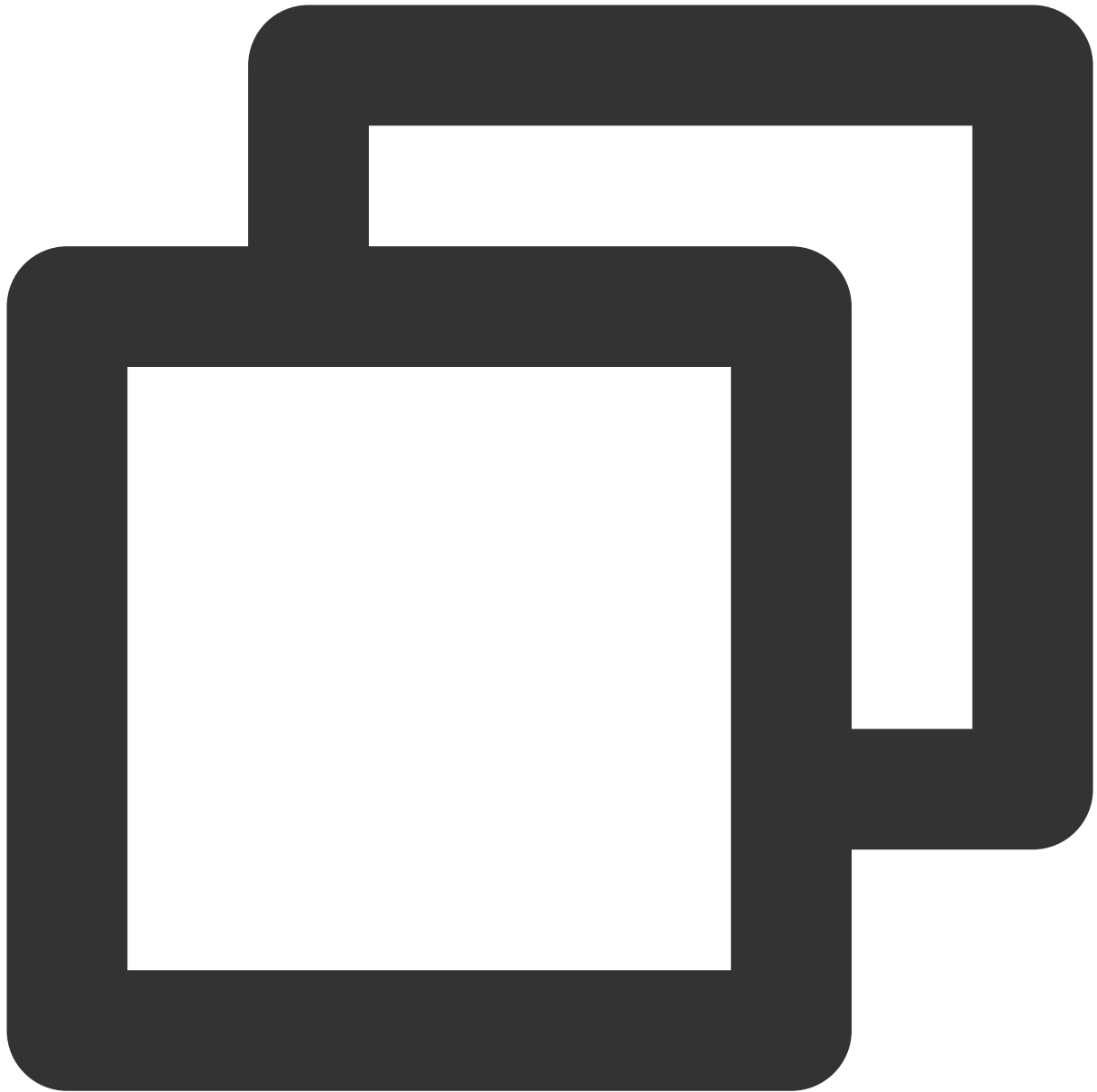
Use the `-S` parameter when running a Git commit command.

1. When you need to commit changes after editing code locally, add the `-S` parameter to the `git commit` command:



```
git commit -S -m "your_commit_message"
```

If you do not wish to enter the `-S` flag every time, you can use the following command to allow Git to sign commits automatically:

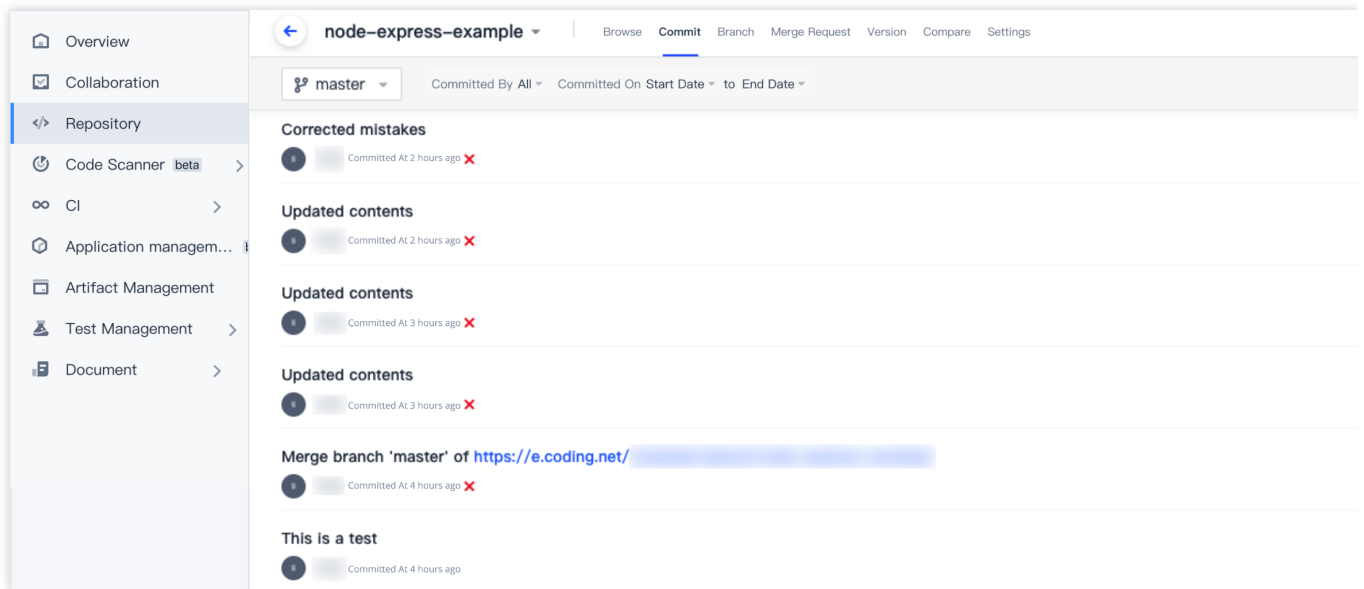


```
git config --global commit.gpgsign true
```

2. When asked, enter the password specified when generating the GPG key.

Step 5: Verify Signature

After pushing the signed commit to the CODING code repository, you can check the verification status of the commit on the **Commits** tab in the code repository.



Commit verification statuses are described below:

Verification Status	Description
Verified	Signed with a GPG private key that corresponds to a public key in a CODING account, and the email address for the public key has been verified.
Not Verified	Signed with a GPG private key that does not correspond to a public key in a CODING account or the email address for the public key has not been verified (if the email address is Not Verified, go to Personal Account Settings > Email Settings to add the email address).
No verification status tag	Not signed with a GPG private key

Delete a GPG Public Key


If your GPG public key has been compromised or is no longer used, you can delete the public key in **Personal Account Settings > GPG Public Key**.

Personal Account Settings

- Account Information
- Personal Account
- Email Settings
- Profile
- Template Settings
- SSH Public Key
- GPG keys beta**
- Access Token
- Two-fa
- Notification
- Bind Settings
- Open Ecosystem

GPG Public Key

After adding the GPG public key, you can add a GPG signature to the commit



Email Address	<input type="text"/>	Verified
Key ID	<input type="text"/>	
subkey	<input type="text"/>	
Added at	2021-09-14	

After the public key is deleted:

Verified commits will change to a Not Verified status.

Future commits using this GPG private key (`git commit -S -m`) will stay unverified.

Unsigned commits (`git commit -m`) will not be verified and will not have a verification status tag.

- Overview
- Collaboration
- Repository**
- Code Scanner beta
- CI
- Application managem..
- Artifact Management
- Test Management
- Document

node-express-example

Browse Commit Branch Merge Request Version Compare Settings

master Committed By All Committed On Start Date to End Date

Corrected mistakes

Committed At 2 hours ago

Corrected mistakes

Committed At 2 hours ago

Updated contents

Committed At 3 hours ago

Updated contents

Committed At 3 hours ago

Updated contents

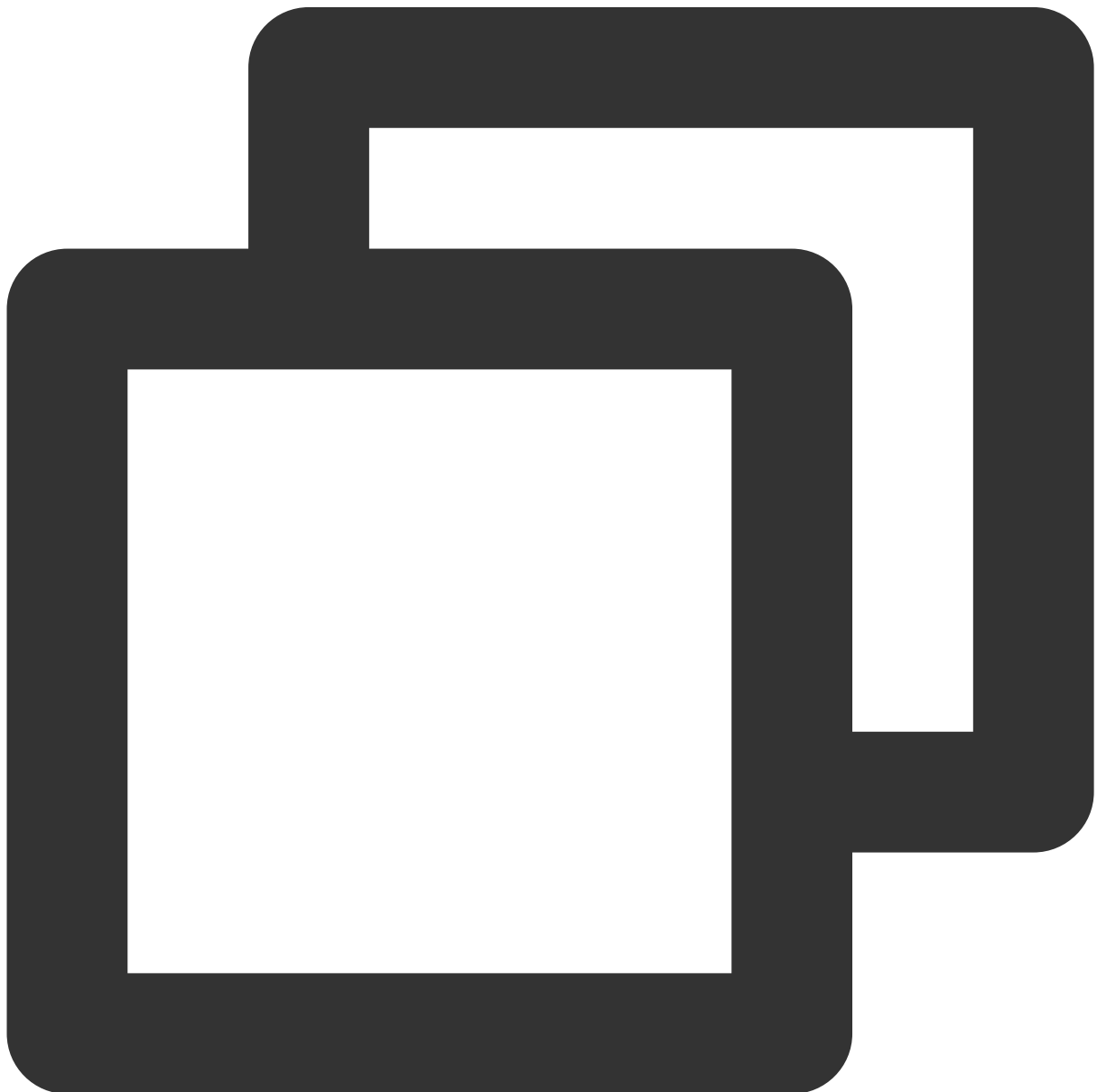
Committed At 4 hours ago

Note:

If you have enabled Git automatic signing, run the `git config --global commit.gpgsign false` command to disable automatic signing. Otherwise, commits pushed to the remote repository will still be **Not Verified** after the GPG public key is deleted.

Fix GPG Signing Errors

If the following error occurs when using `git commit -S -m`, see [Fix GPG Signing Errors](#) and modify the relevant configurations.



```
error: gpg failed to sign the data
fatal: failed to write commit object
```


Enable Verification of Committers and Commit Messages

Last updated : 2023-12-25 17:08:18

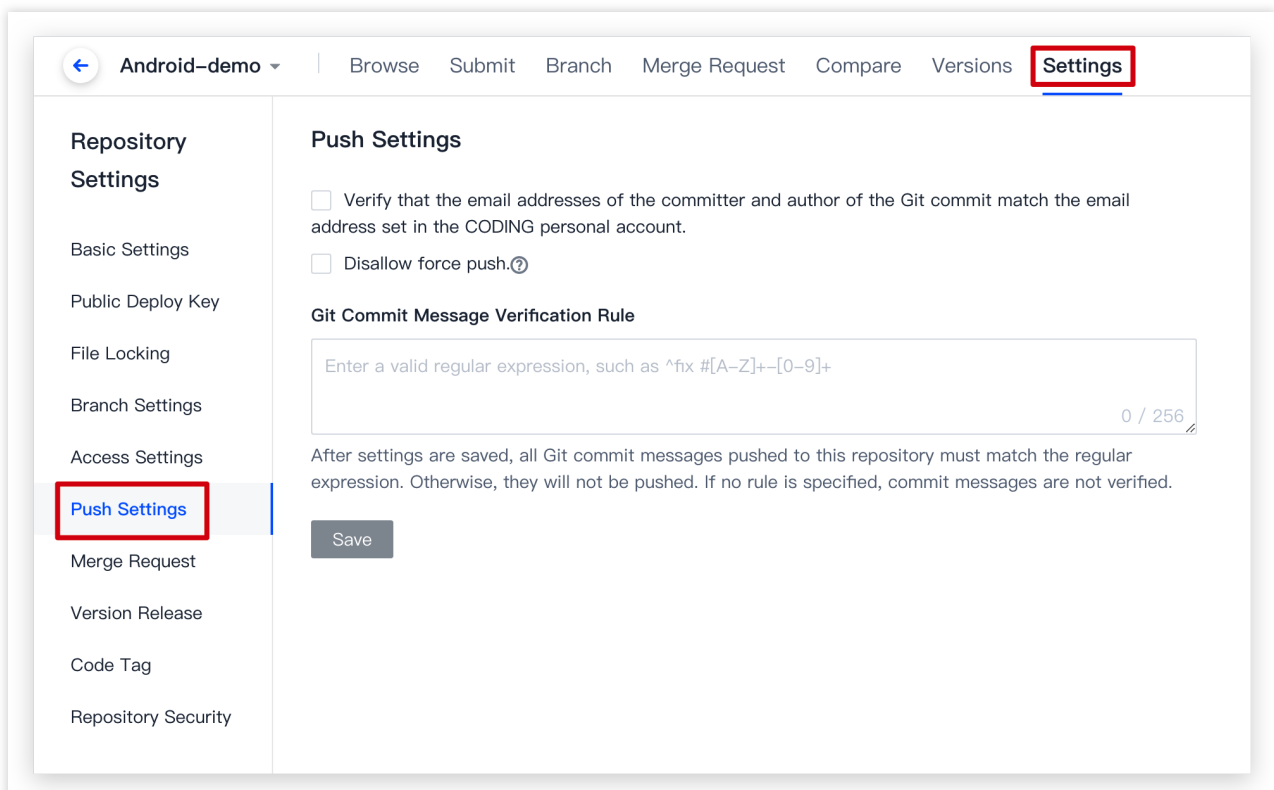
This document describes how to enable verification of committers and commit messages in a code repository.

Open Project

1. Log in to the CODING Console and click the team domain name to go to CODING.
2. Click



in the upper-right corner to open the project list page and click a project icon to open the corresponding project. Project admins can enable verification of Git committers and authors in **Settings > Push Settings**.



Project admins can also set rules for Git commit messages. Commits that do not conform to the rules will be rejected. For example, the rule `^fix #[0-9]+` requires that the commit message must include the associated project issue. `^fix #630` indicates that the commit is associated to issue 630 in the project.

Note:

To learn how to automatically associate issues in commit messages, see [Committing Files](#).

Lock Files and Folders

Last updated : 2023-12-25 17:08:18

This document describes how to lock files and folders in a code repository.

Open Project

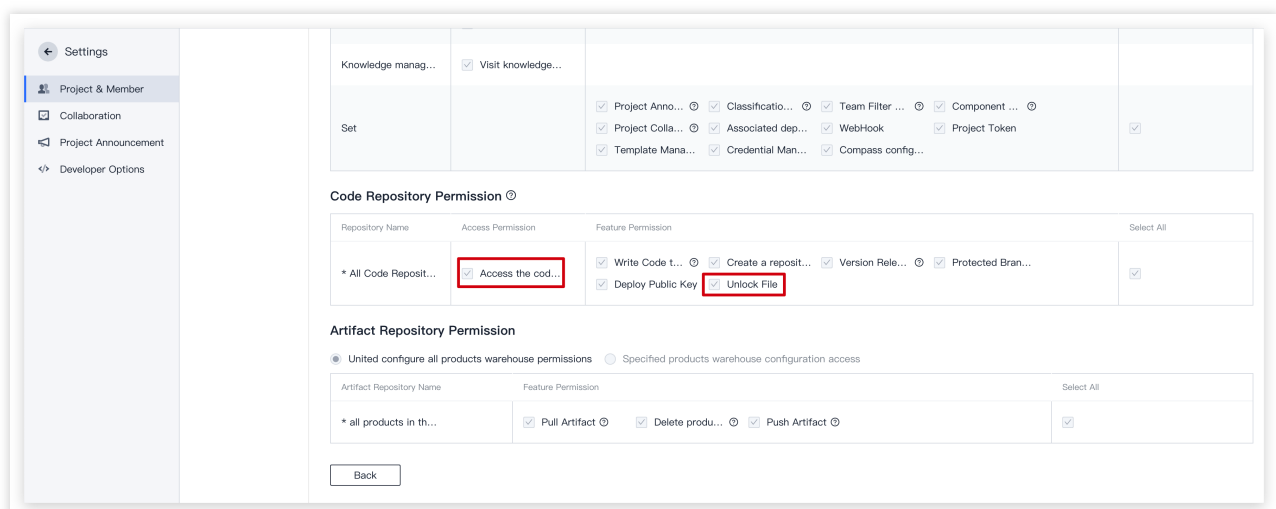
1. Log in to the CODING Console and click the team domain name to go to CODING.
2. Click



in the upper-right corner to open the project list page and click a project icon to open the corresponding project. CODING allows you to lock files or paths of the **Default Branch (usually the master branch)** in code repositories. Only the user who locked the file can modify (edit or delete) it. If a user locks a path, all files in the path will be locked and can only be modified by the user who locked it.

Permission Description

Project admins can enter the project, click **Project Settings** in the lower-left corner, and go to **Projects and Members > User Groups** to set permissions for the repository. After setting the permissions, add team members to the relevant user groups to control the permissions to code repositories in the project.



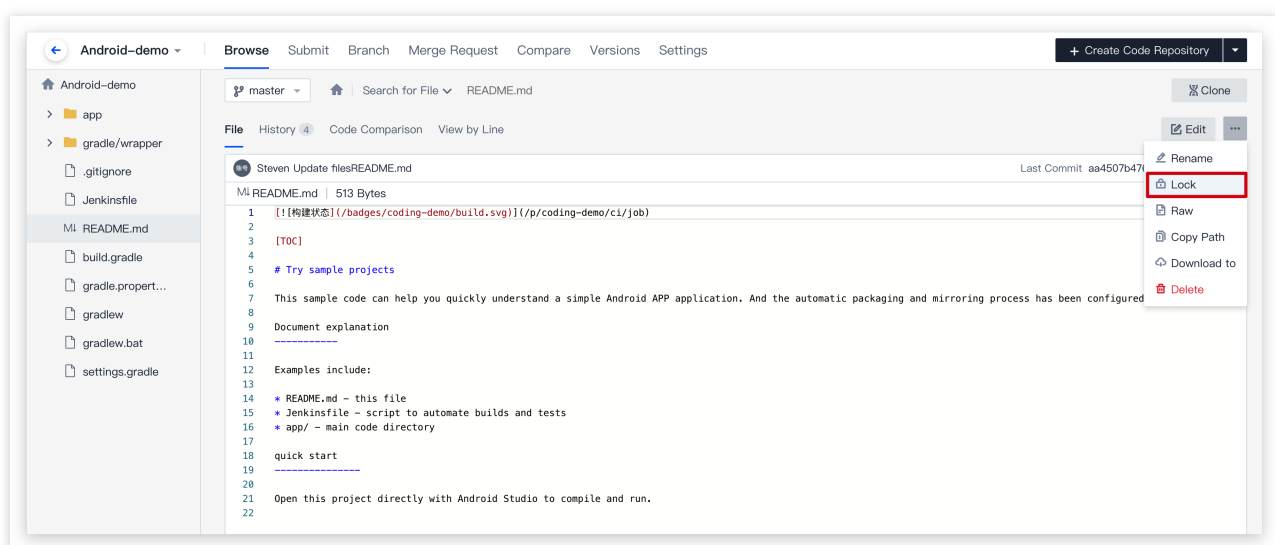
A project member with permission to **Access Code Repositories** can lock files or paths on the Browse page.

Locked files or paths can be unlocked on the Browse page.

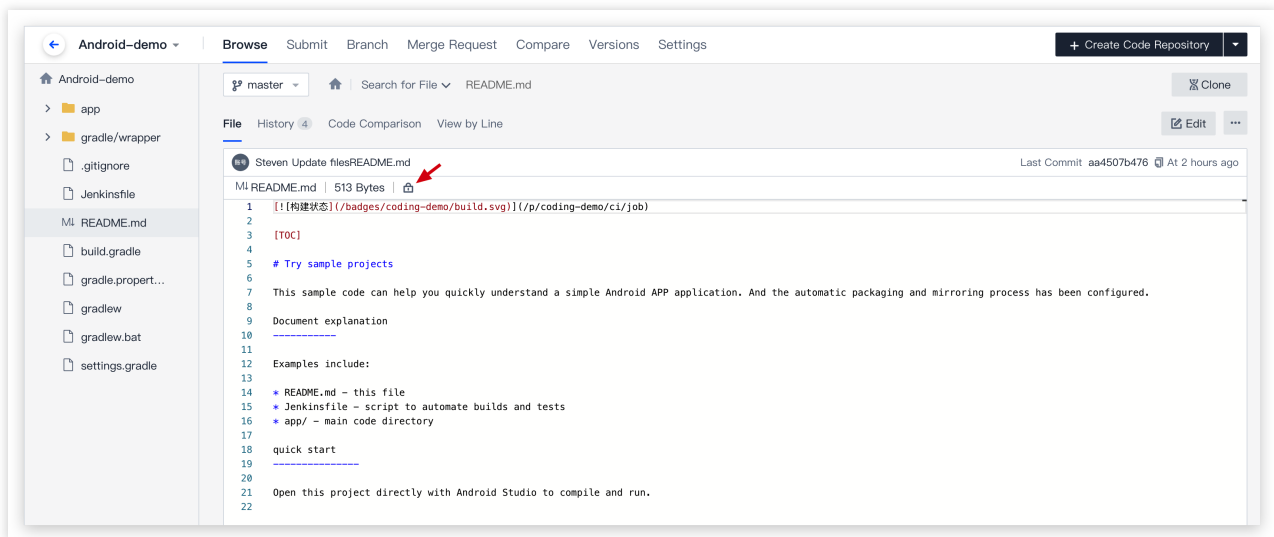
A user can only unlock files or paths they locked; a member with permission to **Unlock Locked Files** can unlock files or paths locked by another user.

Lock a File

Go to the Browse page of a project repository and select a file. Click the More Actions button in the upper-right corner and select Lock.

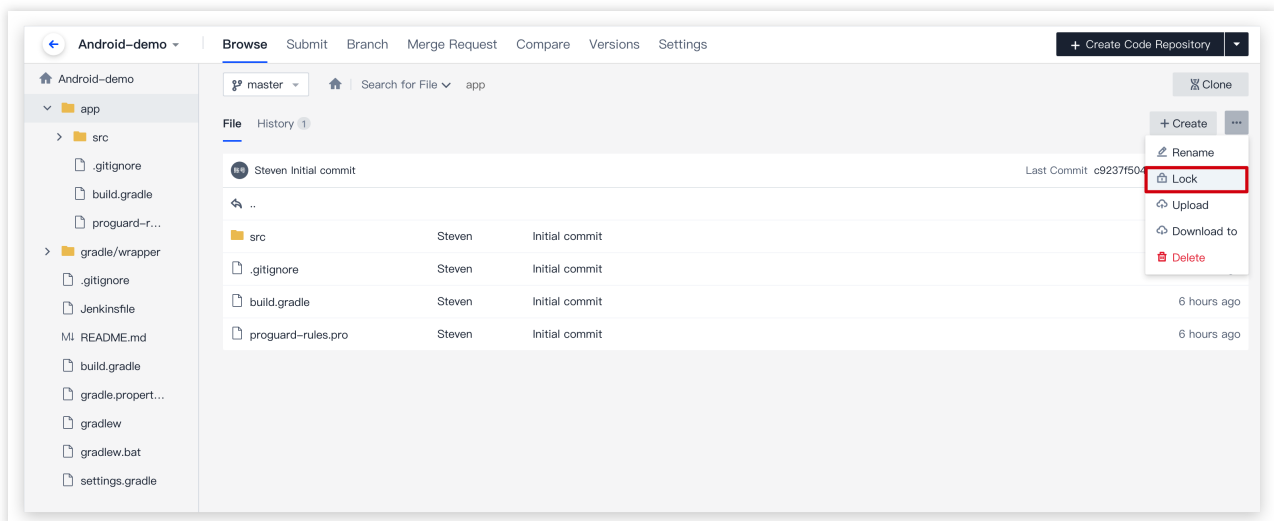


A padlock icon will appear after the file is locked. Only the member who locked this file can edit or delete it.



Lock a Folder

Select a folder in a repository, click the More Actions button in the upper-right corner, and then select Lock to lock all files in the path.

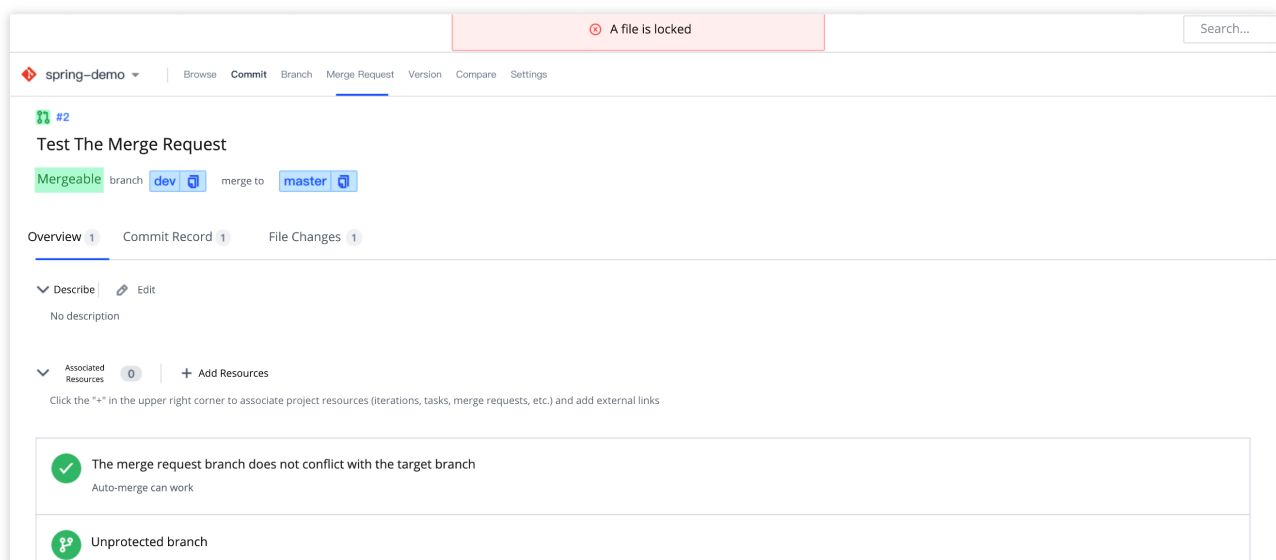


Lock Result

A locked file can only be edited/deleted by the user who locked the file. Only the user who locked the path can create/edit/delete files in it. If another user tries to modify the locked file by pushing, the push will fail.

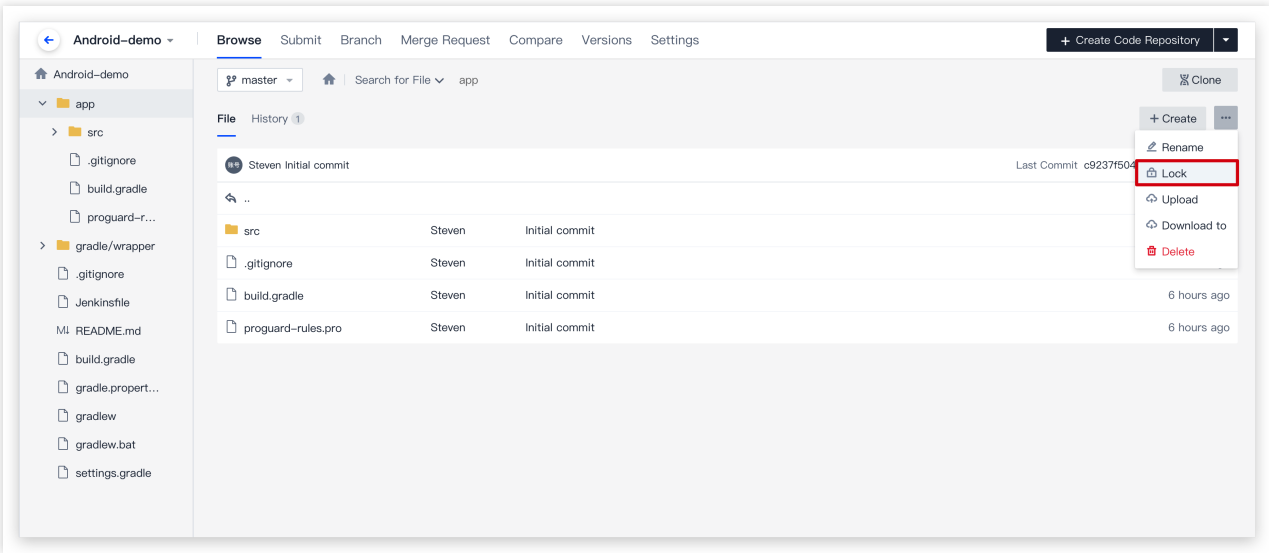
```
Counting objects: 3, done.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 290 bytes | 290.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0)
remote: Permission denied. Your edited files are locked.
remote: 
remote: 
remote: pom.xml
remote: error: hook declined to update refs/heads/master
To https://e.coding.net/
! [remote rejected] master -> master (hook declined)
error: failed to push some refs to 'https://e.coding.net/'
ubuntu@VM-0-14-ubuntu:~/temp/spring-demo$
```

In a merge request, if the target branch is the default branch and the directory to be written includes a locked file or path, the merge can only be performed by the user who locked the file or path. Other members cannot merge the branches.

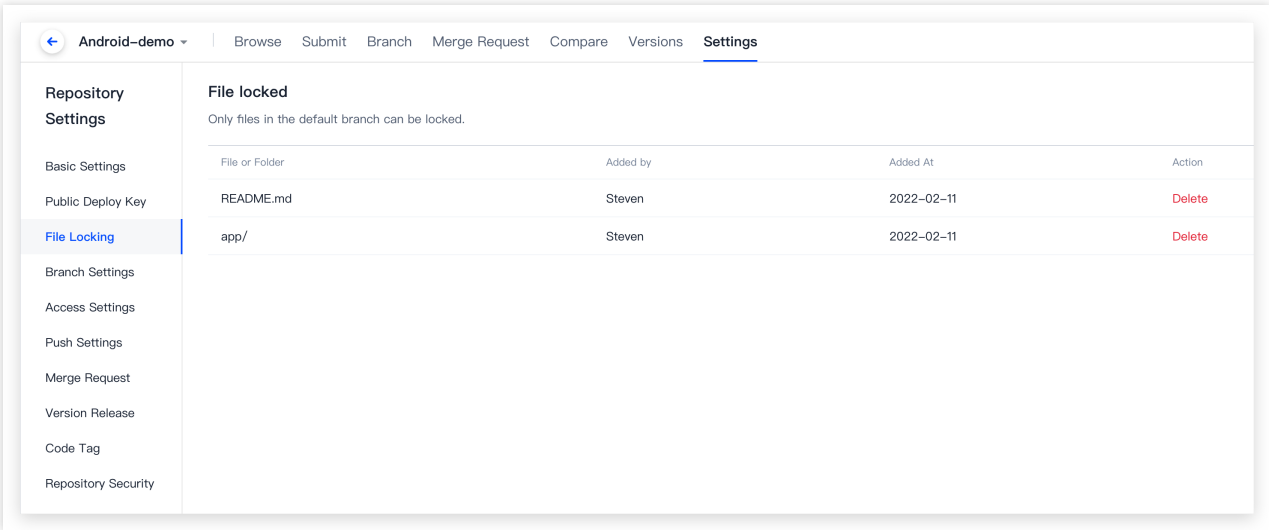


Unlock a Locked File or Folder

A project member can unlock files or folders they locked on the Browse page; a member with permission to **Unlock Locked Files** can unlock files or folders locked by another user.



Project admins can view locked files/folders in **Settings > File Locking**. Click **Delete** to unlock the file or folder.



Git Basics

Common Commands

Last updated : 2023-12-25 17:08:18

This document describes the common Git commands.

Create Repository



```
$ git clone <url>  
$ git init
```

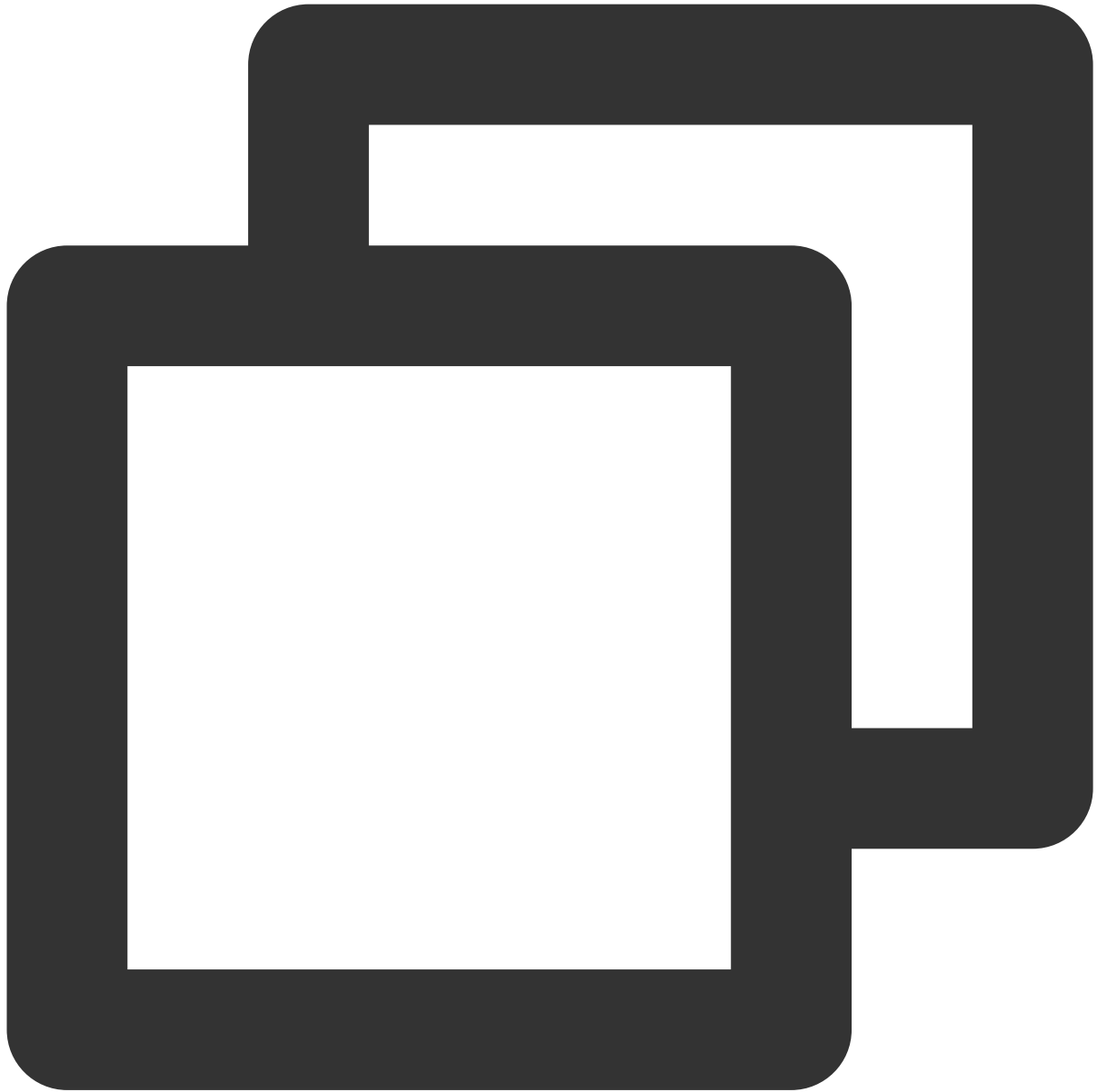
```
# Clone a remote repository  
# Initialize a local repository
```

Modify and Commit



```
$ git status          # View status
$ git diff            # View changes
$ git add .           # Track all files with changes
$ git add <file>      # Track a specific file
$ git mv <old><new>    # Rename a file
$ git rm <file>        # Delete a file
$ git rm --cached <file> # Stop tracking a file without deleting it
$ git commit -m "commit messages" # Commit all updated files
$ git commit --amend   # Modify the last change
```

View Commit History



```
$ git log                # View commit history
$ git log -p <file>      # View the commit history of a specific file
$ git blame <file>       # View the commit history of a specific file as a list
```

Revert



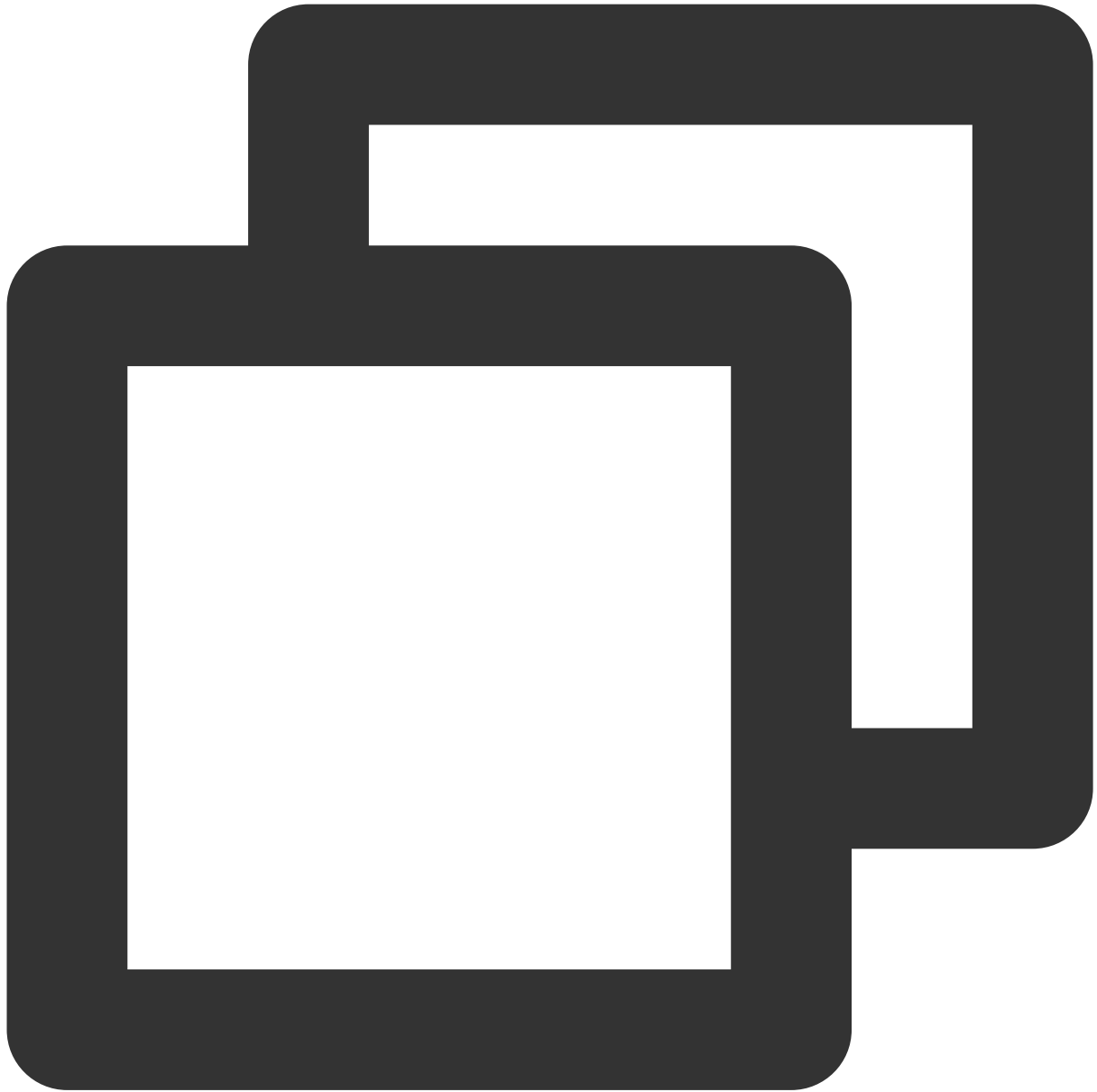
```
$ git reset --hard HEAD      # Discard all uncommitted modifications in the working
$ git checkout HEAD <file>  # Discard uncommitted modifications in a specific file
$ git revert <commit>       # Revert to a specific commit
$ git log --before="1 days" # Show commits created 1 day ago
```

Branch and Tag



```
$ git branch                # Show all local branches
$ git checkout <branch/tag> # Switch to a specific branch and tag
$ git branch <new-branch>   # Create a new branch
$ git branch -d <branch>    # Delete a local branch
$ git tag                   # List all local tags
$ git tag <tagname>         # Create a tag based on the latest commit
$ git tag -d <tagname>      # Delete a tag
```

Merge and Rebase



```
$ git merge <branch>           # Merge a specific branch to the current branch  
$ git rebase <branch>         # Rebase a specific branch to the current branch
```

Remote Operations



```
$ git remote -v                # View remote repository information
$ git remote show <remote>     # View the information of a specific remote repos
$ git remote add <remote> <url> # Add a remote repository
$ git fetch <remote>           # Fetch code from a remote repository
$ git pull <remote> <branch>    # Download code and merge
$ git push <remote> <branch>    # Upload code and merge
$ git push <remote> :<branch/tag-name> # Delete a remote branch or tag
$ git push --tags              # Upload all tags
```

For more information, see the [Git Documentation](#).

LFS Support

Last updated : 2023-12-25 17:08:18

This document describes how to use the Git Large File Storage (LFS) extension.

Feature Overview

CODING supports the Git LFS extension. You can use Git LFS to commit large files of any size without occupying Git repository storage space.

Install

Note:

The Git LFS plugin requires Git 1.8.5 or later.

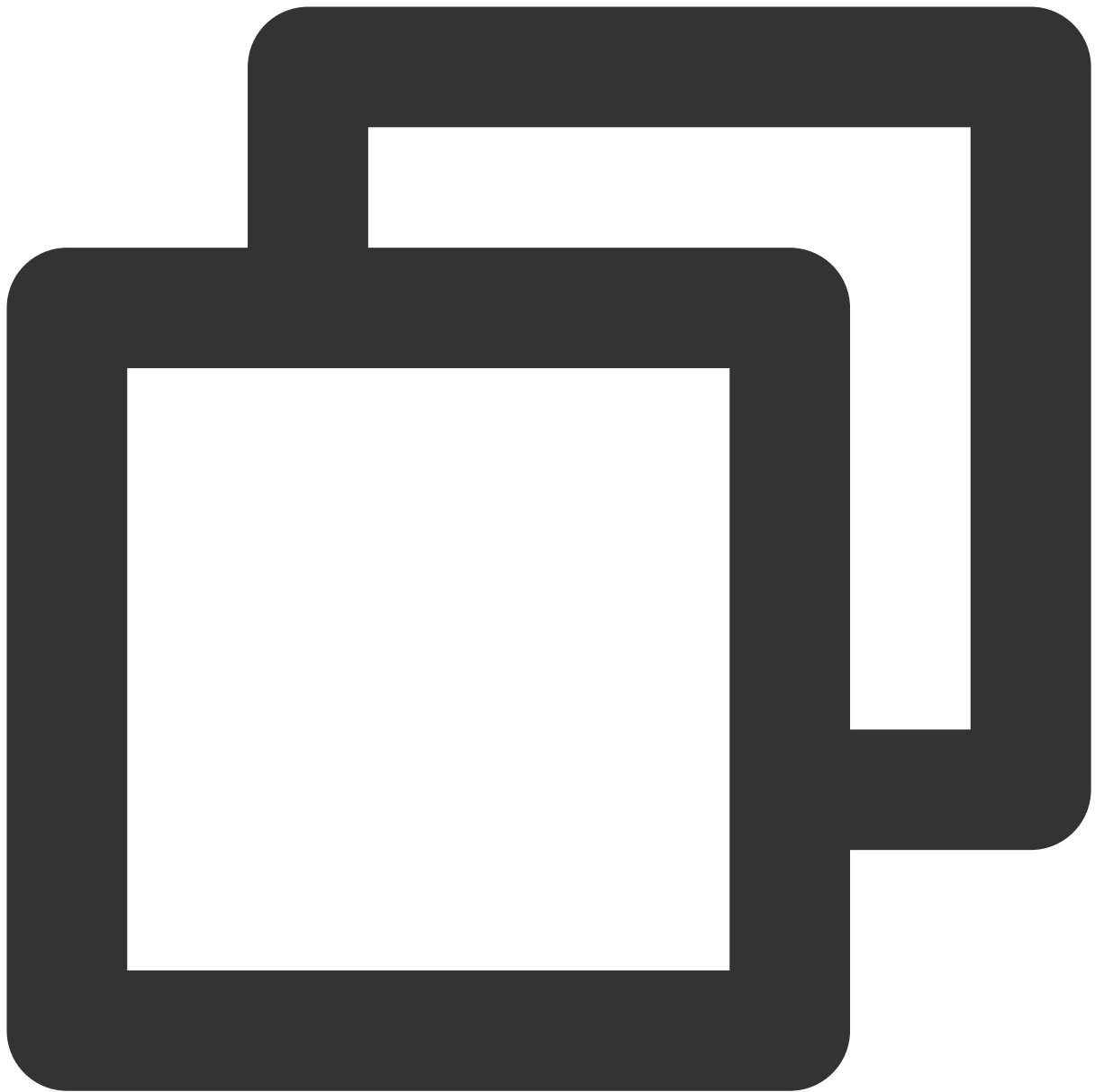
Linux

1. Download the `git-lfs` installation package.



```
curl -s https://packagecloud.io/install/repositories/github/git-lfs/script.deb.s
```

2. Install `git-lfs` .



```
sudo apt-get install git-lfs
```

3. Deploy the LFS tool to Git.



```
git lfs install
```

Mac

1. Install [Homebrew](#).



```
/usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install"
```

2. Install `git-lfs` .



```
brew install git-lfs
```

3. Deploy the LFS tool to Git.



```
git lfs install
```

Windows

1. Download and install the [Windows installer](#).
2. Run the Windows installer.
3. Run `git lfs install` in the command line.

How to Use

For Git commands, see [Common Git Commands](#).

Track files

Git LFS does not process large files by default. Use the `git lfs track` command to track large files.

Track a single file

Use the command `git lfs track "coding.png"` to track the file "coding.png".

Track files with a specific extension

Use the command `git lfs track "*.png"` to track files with the ".png" extension. This tracks both existing and future files with the ".png" extension.

View tracked file patterns

Run the `git lfs track` command:



```
Listing tracked patterns
*.png (.gitattributes)
```

Commit large files

You need to commit the ".gitattributes" file to the repository when committing code. After the commit is complete, run the `git lfs ls-files` command to view the tracked LFS file list.



```
f05131d24d * cat.png  
7db207c488 * dog.png
```

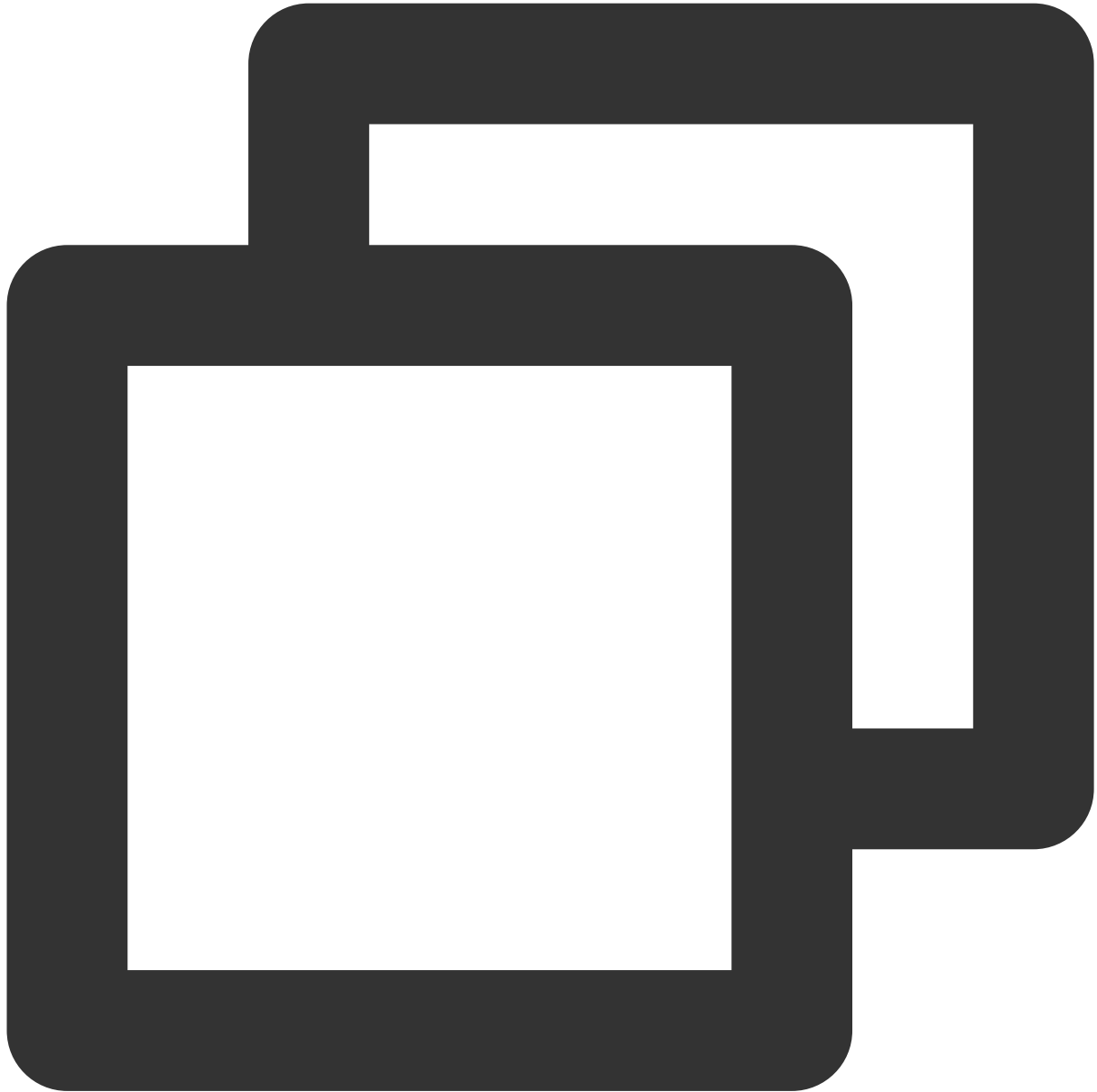
After the code is pushed to the remote repository, tracked LFS files will be shown after "Git LFS":



```
$ git push origin master
Git LFS: (2 of 2 files) 12.58 MB / 12.58 MB
Counting objects: 2, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 548 bytes | 0 bytes/s, done.
Total 5 (delta 1), reused 0 (delta 0)
To https://e.coding.net/coding/coding-manual.git
67fcf6a..47b2002  master -> master
```

Clone a remote repository containing Git LFS files

Use the `git lfs clone` command to clone a remote repository containing "Git LFS" files to a local machine.



```
$ git lfs clone https://e.coding.net/coding/coding-manual.git
Cloning into 'coding-manual'
remote: Counting objects: 16, done.
remote: Compressing objects: 100% (12/12), done.
remote: Total 16 (delta 3), reused 9 (delta 1)
Receiving objects: 100% (16/16), done.
Resolving deltas: 100% (3/3), done.
Checking connectivity...done.
```

```
Git LFS: (4 of 4 files) 0 B / 100 B
```

Note:

To learn more about how to use Git LFS, run the `git lfs help` command.

If you need to store files from the original repository in LFS, see the [tutorial](#).

Go Get Support

Last updated : 2023-12-25 17:08:18

Feature Overview

Golang Get utilizes a code management tool to remotely pull or update code packages and their dependencies and automatically compiles and installs them. The entire process is as simple as installing an application. The **go get** command can be used for multiple code repositories in CODING. The following is a quickstart guide.

Getting Started

Suppose a user has a code repository called Repo A with a Git HTTPS URL:

`https://e.coding.net/{team}/{project}/{repo}.git` (the curly braces are variables). Using the repository `https://e.coding.net/baulk/jackson/mux.git` as an example:

The user can configure module names using the following command.



```
go mod init e.coding.net/baulk/jackson/mux
```

To obtain a module using go get:



```
go get e.coding.net/baulk/jackson/mux
```

The modules of sub-repositories can be obtained when multiple repositories are used:



```
go get e.coding.net/baulk/jackson/mux/dev
```

The Git HTTPS clone URL of some repositories is: `https://e.coding.net/{team}/{project}.git`. The user can configure module names using the following command:



```
go mod init e.coding.net/team/project
```

To obtain a module using go get:



```
go get e.coding.net/team/project
```

Note:

Sub-modules cannot be directly obtained for such repositories. Use `e.coding.net/team/project/project` as the module name to obtain modules and their sub-modules.

Personal Settings

Access Token

Last updated : 2023-12-25 17:08:18

A personal access token is similar to the special passwords for applications in certain systems. You can use the generated token to access specific APIs to create applications or scripts.

Procedure

1. Log in to the CODING Console and click **Use Now** to go to CODING page.
2. Hover over your profile photo in the upper-right corner and click **Personal Settings**.
3. In the menu on the left, click **Access Token**.

Create an access token

1. Click **Create Token**, enter a token description, and select the access permissions for the token.
2. Click **Create Token**. When submitting the request, you will need to enter the service password to verify your identity.
3. Click **OK** to create the access token.
4. After submitting the request, you will be brought back to the list of access tokens, and the new token will be shown.

Note:

The token will only be shown once. As a security precaution, copy and paste it into your application or script and do not save a copy. If you are using the token for testing purposes, generate a new token after the testing and paste it into the final application or script.

Edit an access token

1. Click **Edit** on the right of the token to enter the Edit page.
2. You can edit the token description and permissions.
3. Click **Update Token** and enter the service password to verify your identity.
4. Click **OK** to complete editing the token.

Note:

If you have lost or forgotten the token, click **Regenerate** on the Edit page to update the token.

Delete an access token

1. Click **Delete** on the right of a token to delete it. You can also click **Remove All** to remove all access tokens.
2. Enter the service password to verify your identity, and then click **OK** to delete the token.

Note:

If a token is no longer used, we recommend you delete the token to prevent it from being compromised.

SSH Public Keys and Project Tokens

Last updated : 2023-12-25 17:08:18

In CODING, SSH public keys have different permission scopes depending on the usage scenario. This document describes the differences between SSH public keys and project tokens.

Function Overview

SSH public key files associated with a CODING account are referred to as **SSH Public Keys**. After they are configured, they have read and write permissions to all projects. If they are associated with a certain project, they are referred to as **Project Tokens**. After they are configured, they have read-only permission to the project by default.

Generate a Public Key

Run the following commands:



```
ssh-keygen -m PEM -t rsa -b 4096 -C "your.email@example.com"
# Creates a new ssh key, using the provided email as a label
# Generating public/private rsa key pair.
Enter file in which to save the key (/Users/you/.ssh/id_rsa): [Press enter] // We
Enter passphrase (empty for no passphrase): // Press Enter without entering a pas
```

If you need multiple SSH key pairs (you may be working with multiple code hosting platforms), when you are prompted to "Enter file in which to save the key", enter a new file name, so the default key pair will not be overwritten.

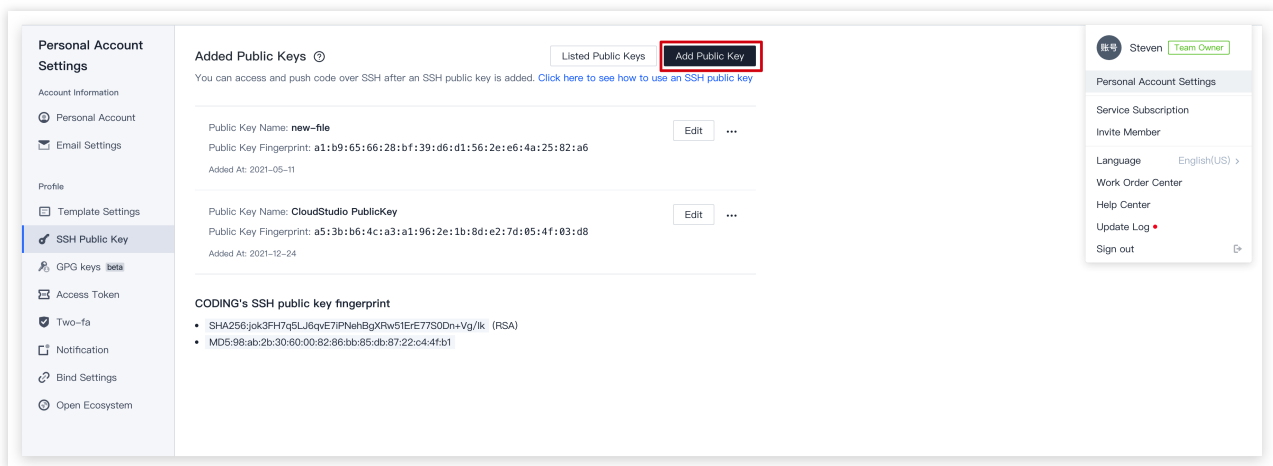
After the operation succeeds, you will see the following information:



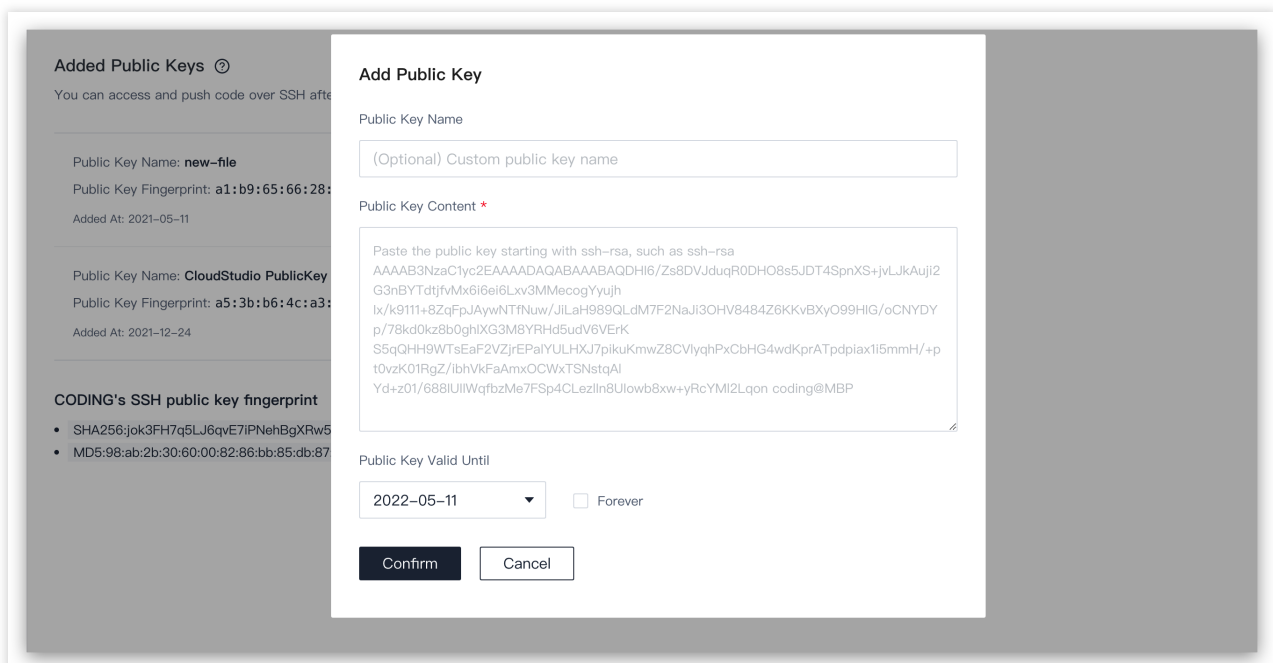
```
Your identification has been saved in /Users/you/.ssh/id_rsa.  
# Your public key has been saved in /Users/you/.ssh/id_rsa.pub.  
# The key fingerprint is:  
# 01:0f:f4:3b:ca:85:d6:17:a1:7d:f0:68:9d:f0:a2:db your.email@example.com
```

Add the SSH Public Key

1. Enter "open ~/.ssh" in the terminal, open the "id_rsa.pub" file with a text editor, and copy all the content. (id_rsa.pub is the default name of the generated public key. Open the corresponding file if you used a different name.)
2. Click your profile photo in the upper-right corner of the page and select **Personal Account Settings**. Go to **Personal Account Settings > Personal Settings > SSH Public Keys**.



3. Paste the content copied in Step 1 into the **Public Key** field and enter a key name.
4. Set the validity of the public key. You can select a specific expiration date or set it to Never expire.

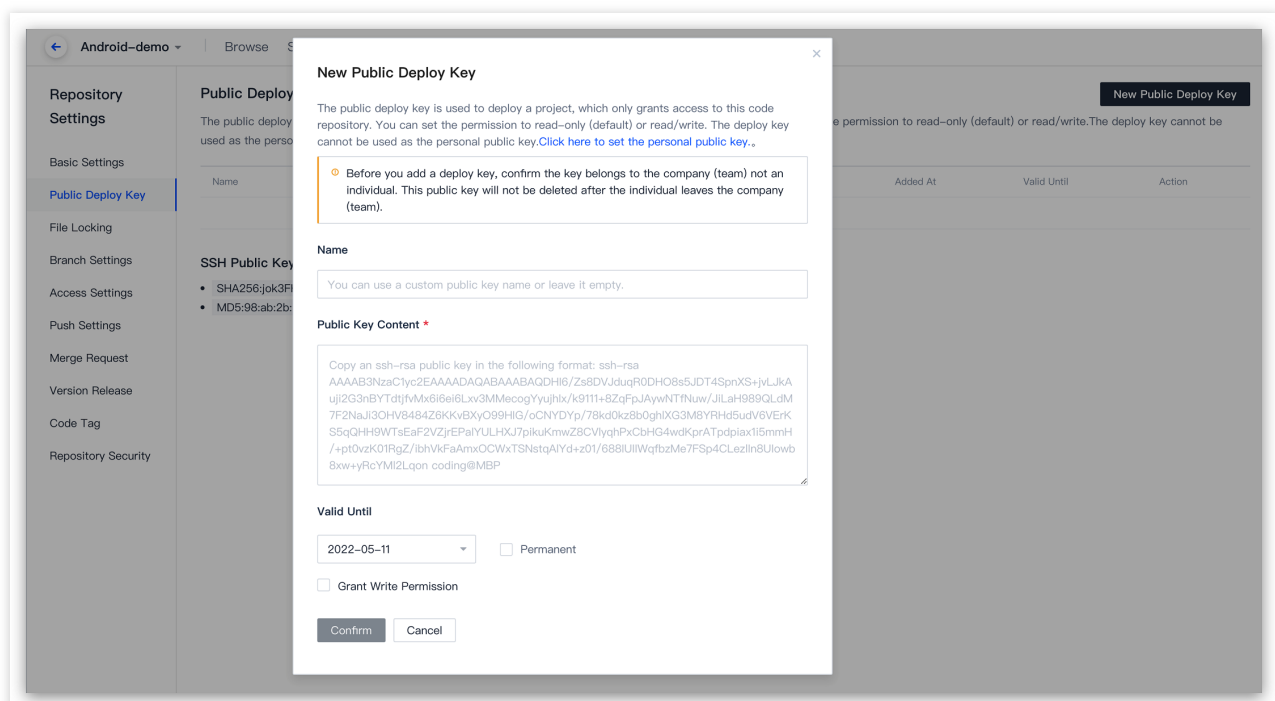


5. Click **Add** and enter the password to add the public key.
6. Then, run a test in the command line. You will need to trust the host when establishing a connection for the first time. Run the `ssh -T git@e.coding.net` command. You can also verify whether the connection with a CODING

remote repository is correct using Key Fingerprint Authentication.

Add the Public Deploy Key

1. Enter "open ~/.ssh" in the terminal, open the "id_deploy.pub" file with a text editor, and copy all the content. (The public deploy key here is named "id_deploy.pub". You can customize the name when generating the public deploy key.)
2. In the target project, go to the code repository > **Public Deploy Key** and click **Create Public Deploy Key**.



3. Paste the content copied in Step 1 into the **Public Key** field and enter a key name.
4. Click **Create** and enter the password to add the public deploy key.

Note:

Public deploy keys have read-only permission to the project by default. Select **Grant Push Permission** to obtain push permission.