代码托管

操作指南

产品文档

腾讯云

文档目录

操作指南
代码仓库管理
创建仓库

最近更新时间：2023-12-25 17:08:18

本文为您详细介绍如何创建仓库。

进入项目

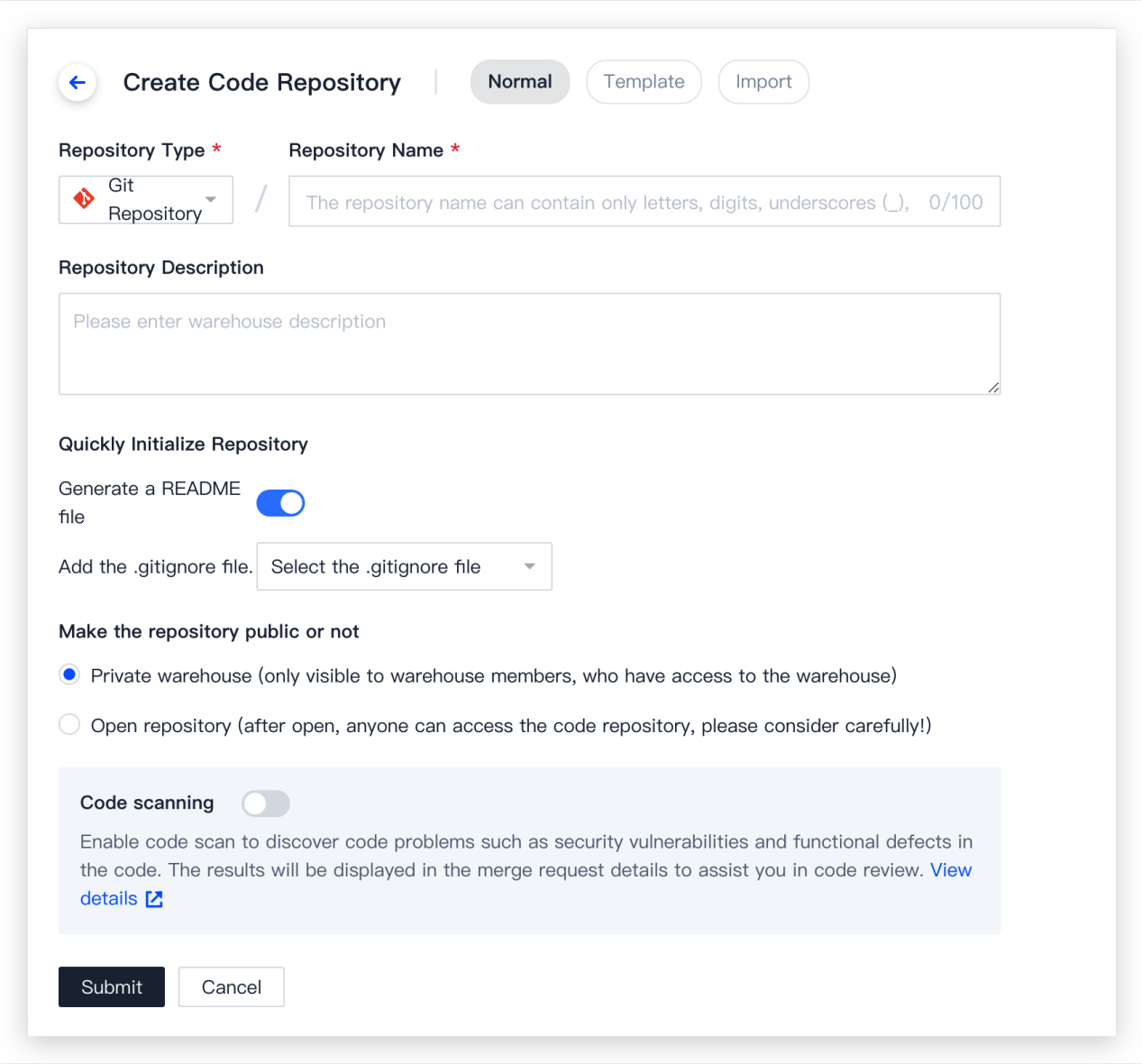
1. 登录 CODING 控制台，单击团队域名进入 CODING 使用页面。
2. 单击页面右上角的

，进入项目列表页面，单击项目图标进入目标项目。
3. 选择左侧菜单**代码仓库**，进入代码仓库首页。
4. 若左侧未显示代码仓库，需项目管理员前往**项目设置** > **项目与成员** > **功能开关**打开功能开关。

创建 Git 和 SVN 仓库

创建 Git 仓库

1. 在代码仓库列表，单击右上角**创建代码仓库**，选择**普通创建**。
2. 仓库类型选择 Git，输入符合条件的仓库名称。
3. 建议打开**生成 README 文件**开关。该功能开启之后，Git 仓库创建完成后会自动初始化。
4. 建议选择**私有仓库**，按需开启**代码扫描**功能， 以及时发现和规避潜在的代码问题。
5. 单击**完成创建**。



说明：

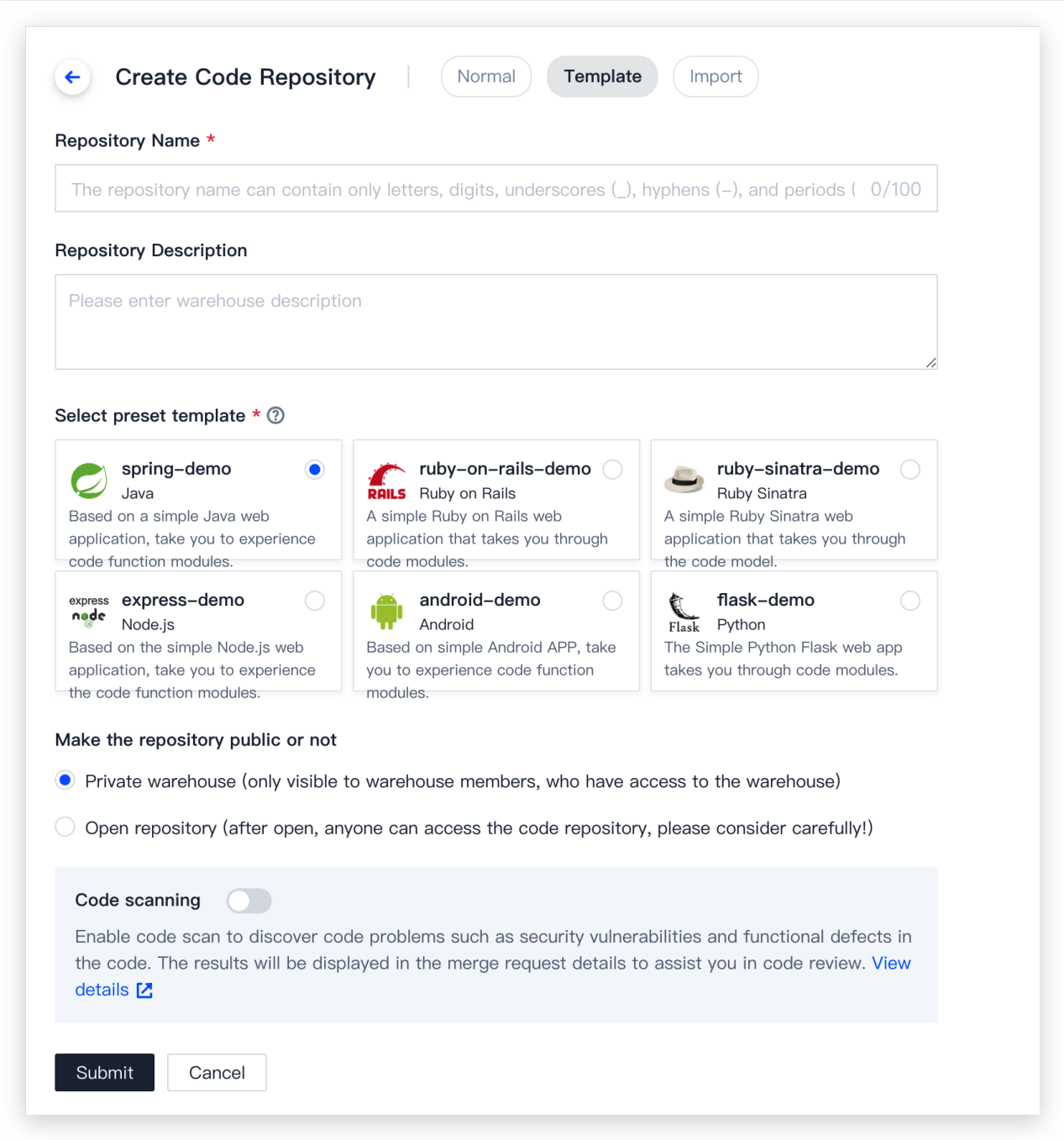
在完成代码仓库初始化后，您可以使用 Git 命令来与本地仓库进行联动。详情请参见 Git 常用命令。

创建 SVN 仓库

CODING 支持创建 SVN 仓库。有关如何创建与使用 SVN 仓库，请参见 SVN 仓库使用。

模板创建

CODING 提供预置的代码仓库模块。您可以通过示例代码快速体验代码仓库模块是如何与持续集成或构建产物进行关联的。



导入外部仓库

您可以将已有的 Git 仓库快速迁移至 CODING DevOps 平台。详情请参见 导入或关联外部仓库。

导入或关联外部仓库

最近更新时间：2023-12-25 17:08:18

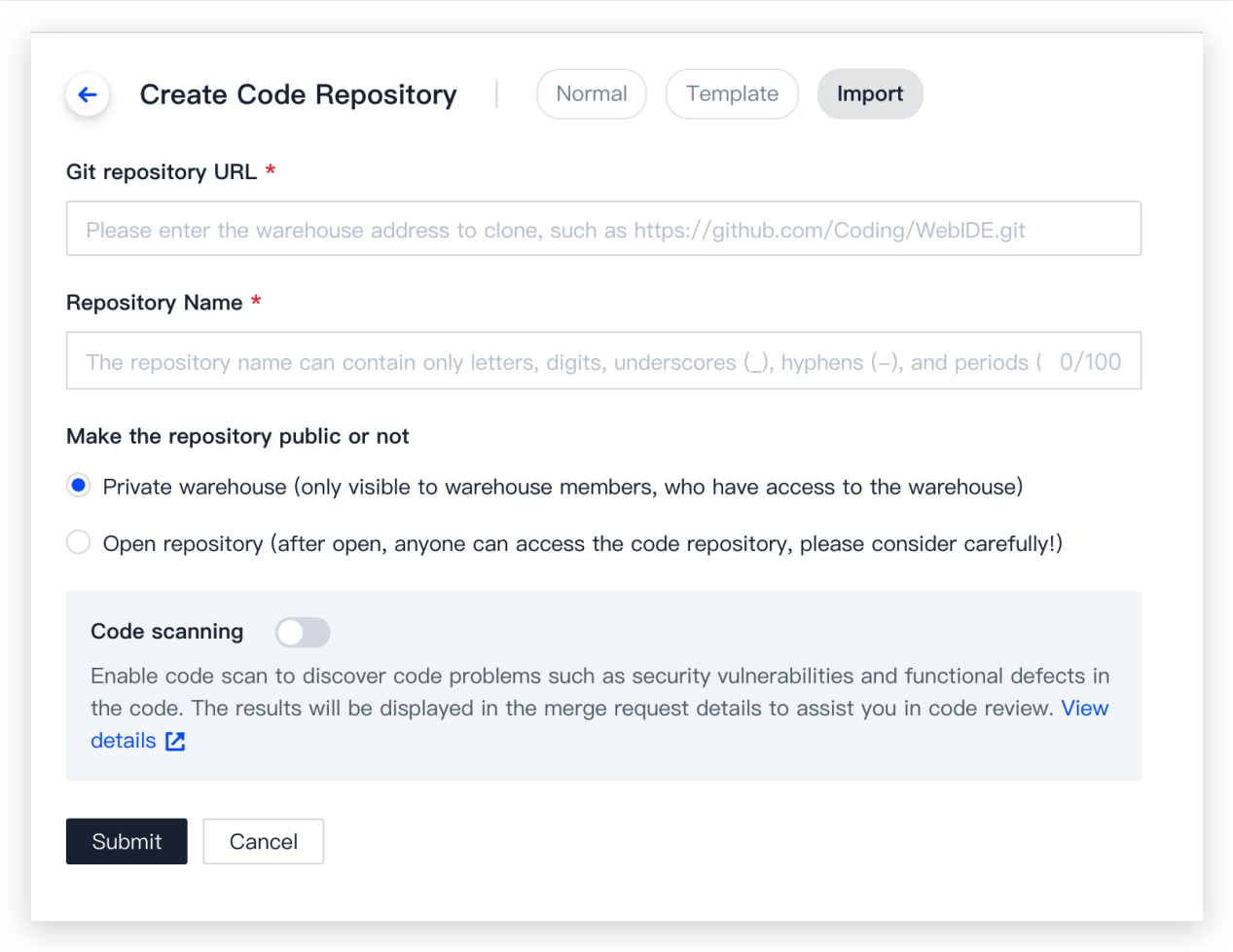
进入项目

1. 登录 CODING 控制台，单击团队域名进入 CODING 使用页面。
2. 单击页面右上角的

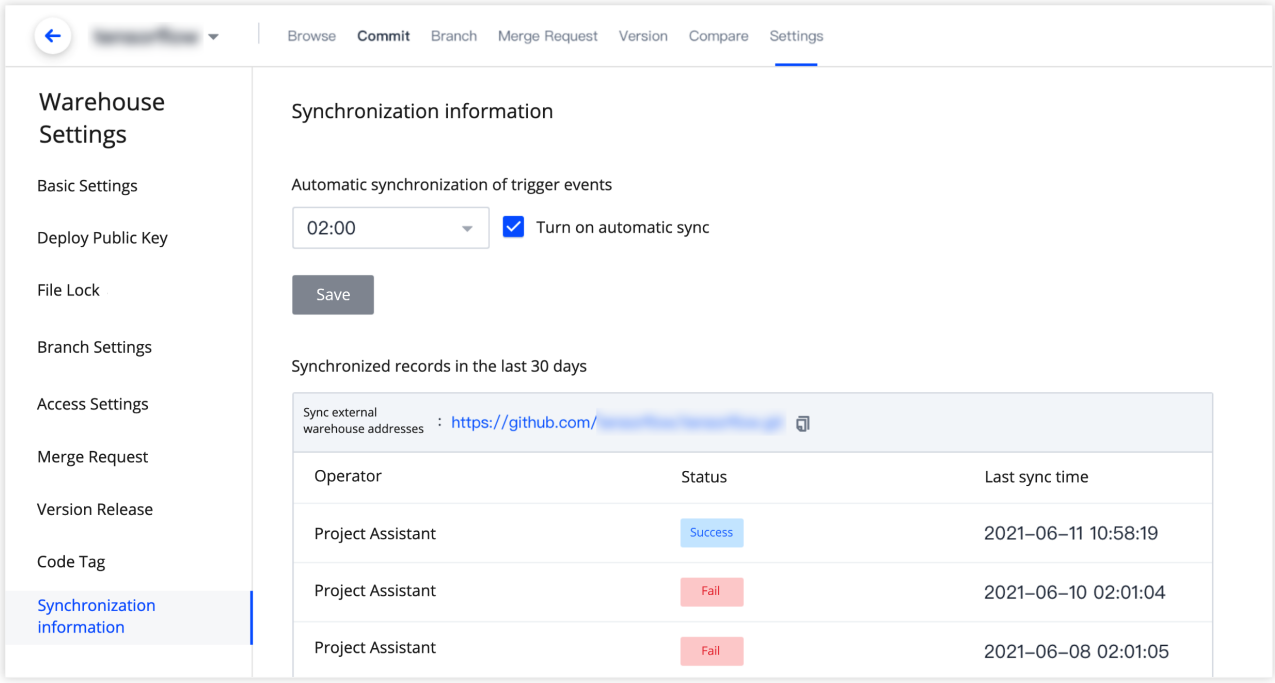
，进入项目列表页面，单击项目图标进入目标项目。
3. 选择左侧菜单**代码仓库**，进入代码仓库首页。
4. 若左侧未显示代码仓库，需项目管理员前往**项目设置 > 项目与成员 > 功能开关**打开功能开关。

导入外部代码仓库

CODING 代码仓库不仅提供了一键导入外部开源仓库功能，还支持定时同步外部仓库。在创建代码仓库时选择**导入外部仓库**，填入 Git 开源仓库的 URL 地址即可导入。

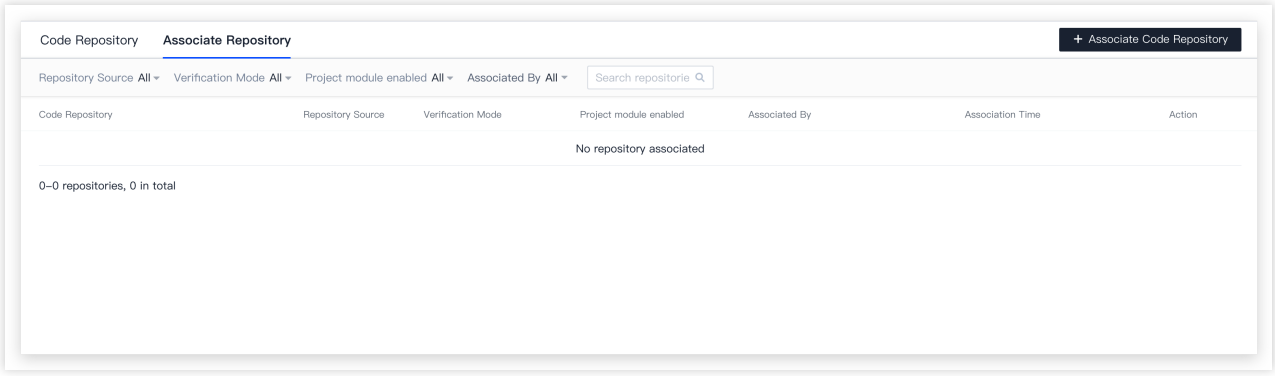


同步意味着和源仓库保持一致，将覆盖您在 CODING 仓库中做出的改动。您可以在仓库设置中修改同步信息频率或关闭自动同步功能。

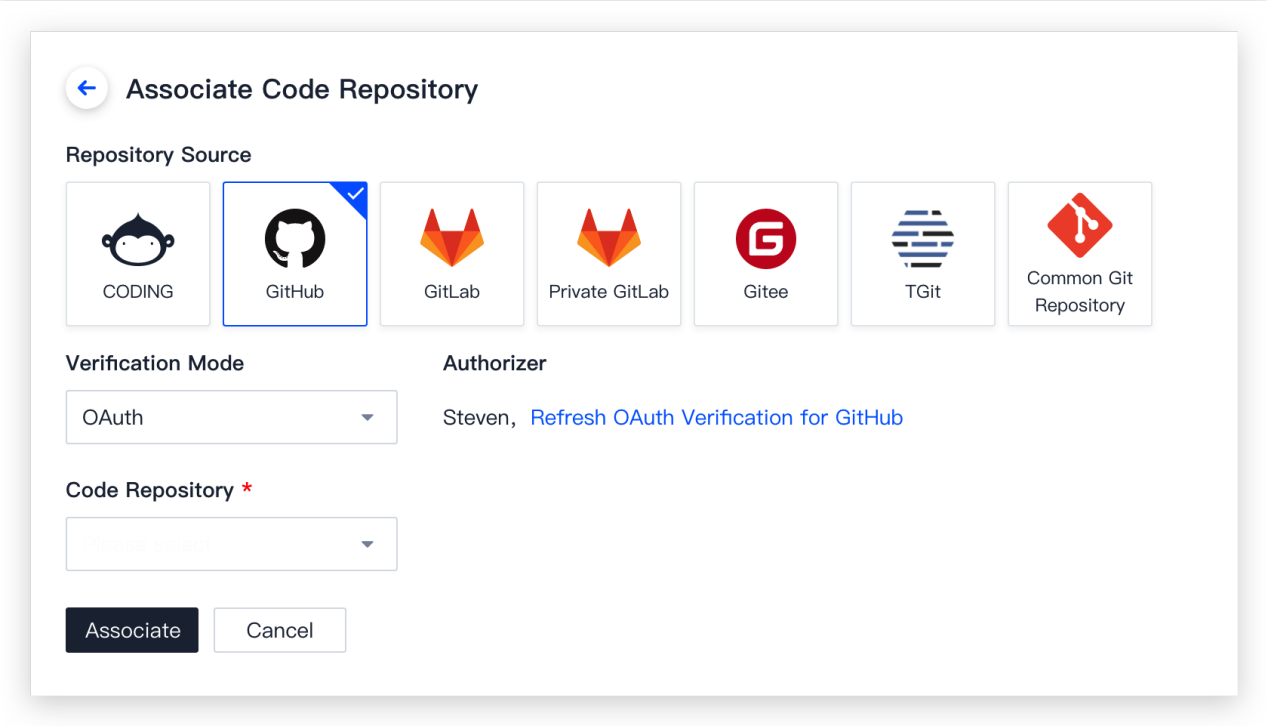


关联代码仓库

关联仓库功能本质上是将访问外部仓库的凭据“暂存”至 CODING，当您使用**持续集成**或**持续部署**时，能够直接调用第三方仓库作为代码源，而省去了频繁迁移的繁琐流程。



支持的关联仓库类型有 GitHub 、GitLab 、私有 GitLab、Gitee、工蜂、通用 Git 仓库与其他项目中的 CODING 仓库。前五种仓库类型支持 OAuth 认证方式，通用 Git 仓库支持账号密码认证，关联后的仓库代码不会存储至 CODING 代码仓库。



关联 GitLab 私有仓库

如需关联私有 GitLab 仓库，需要在 GitLab 创建应用然后由团队管理员绑定私有 GitLab 服务。具体操作请参见 绑定私有 GitLab。

关联 GitLab SaaS 仓库

如需关联 GitLab SaaS 版本上的仓库，在**关联代码仓库**页面选择 **GitLab** 代码源，然后单击**前往认证**，在跳转的 GitLab 验证页面单击 **Authorize** 完成授权。授权成功后，选择需要关联的代码仓库即可完成操作。

关联 GitHub 仓库

在**关联代码仓库**页面，选择 **GitHub** 代码源，然后单击**前往认证**使用 OAuth 认证即可跳转至 GitHub 进行授权认证。若提示失败，有可能是因为您未在 GitHub 中填写用户名。请前往 **Settings** > **Profile** > **Name** 进行填写。

查看仓库详情

最近更新时间：2023-12-25 17:08:18

本文为您详细介绍如何查看仓库详情。

进入项目

1. 登录 CODING 控制台，单击团队域名进入 CODING 使用页面。
2. 单击页面右上角的

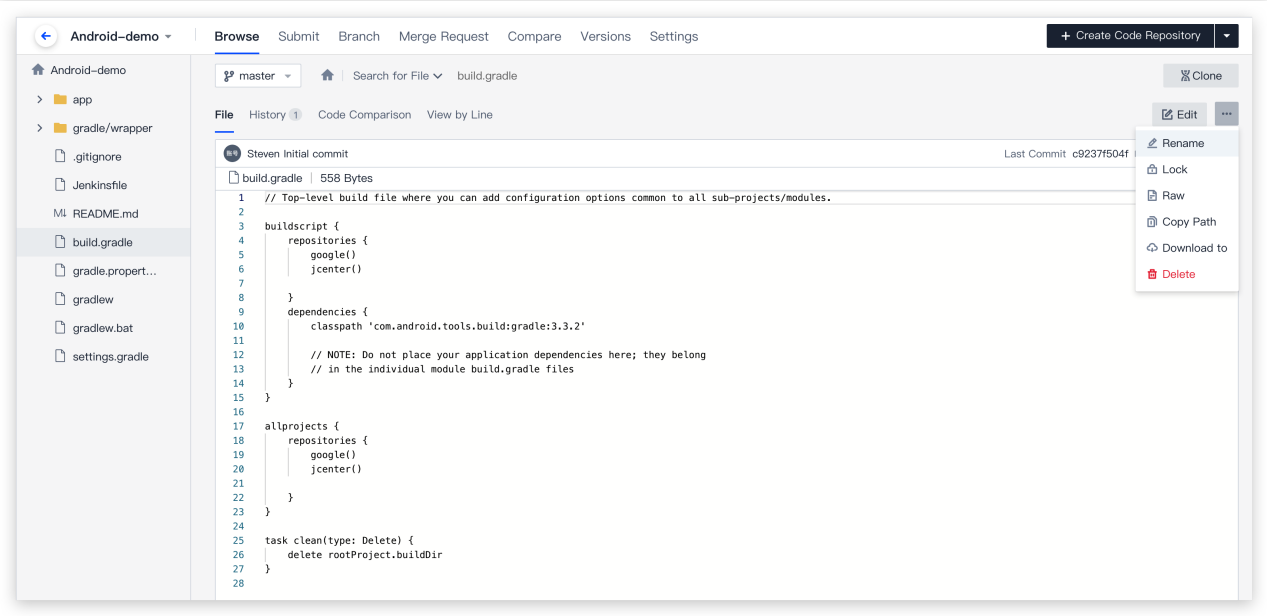
，进入项目列表页面，单击项目图标进入目标项目。
3. 选择左侧菜单**代码仓库**，进入代码仓库首页。
4. 若左侧未显示代码仓库，需项目管理员前往**项目设置** > **项目与成员** > **功能开关**打开功能开关。

在**代码仓库**页面，单击任一仓库名称即可进入该仓库的详情页面。详情页面默认显示 master 分支的文件和提交历史等信息。

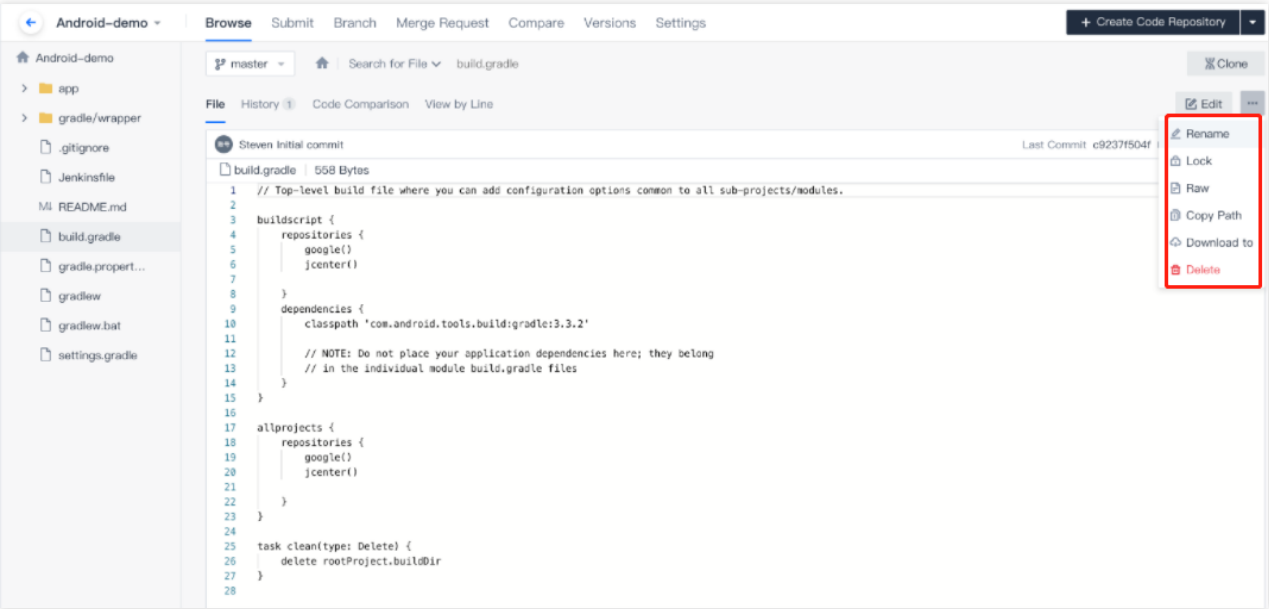
文件列表

进入仓库的详情页面之后，默认显示仓库中 master 分支的文件目录结构，并展开 `readme` 文件的具体内容。

单击左侧目录结构中的任一文件夹，该文件夹中包含的所有文件将会在右侧**文件**页签中列出。您可以新增文件或文件夹，或执行重命名、锁定、上传、下载或删除操作。



单击左侧目录结构中的任一文件，该文件的内容将会显示在右侧**文件**页签。您可以对当前文件进行编辑，或执行重命名、锁定、RAW、下载或删除操作。

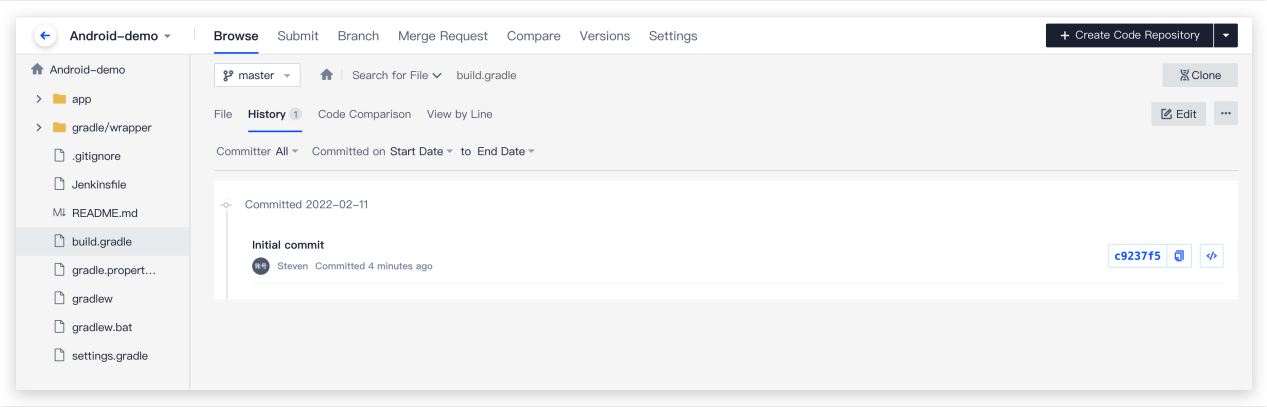


说明：

将鼠标悬浮在目录结构中的任一文件或文件夹上，会出现更多操作按钮。您也可以通过该按钮执行对应操作。

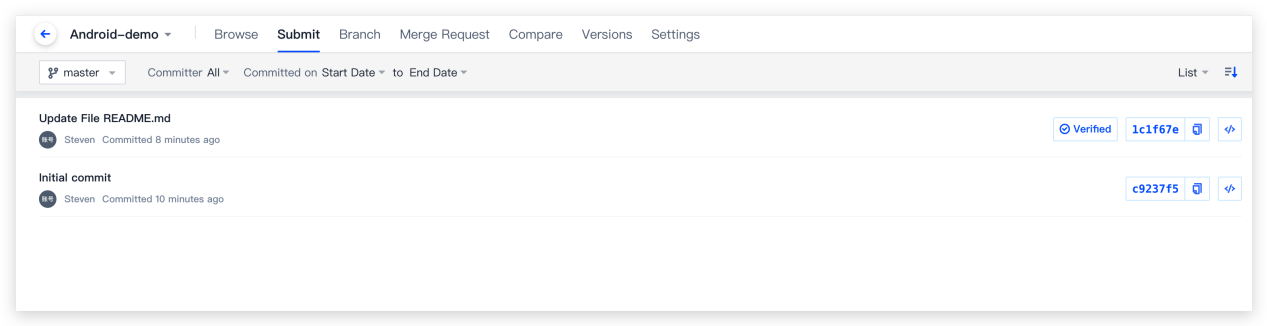
提交历史

1. 单击**历史**页签，默认进入 master 分支的提交历史页面。
2. 提交记录按照日期倒序排列。单击任一提交记录的名称或右侧的 SHA ID 可以跳转至该代码仓库的**提交**页签查看提交详情。

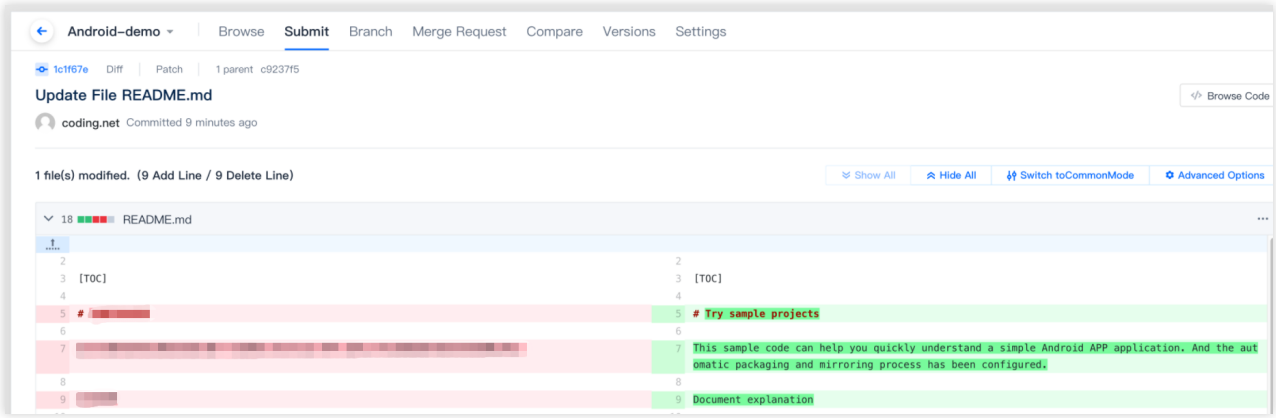


提交详情

1. 在代码仓库详情页，单击**提交**即可进入提交记录管理页面。该页面默认按照时间倒序列出 master 分支的提交记录。您可以切换为其他分支，查看相应的提交历史记录。



2. 单击任一提交记录的名称或 SHA ID，将会在新页面打开该记录的详情页面，清晰地列出本次提交中所有更改过的文件。

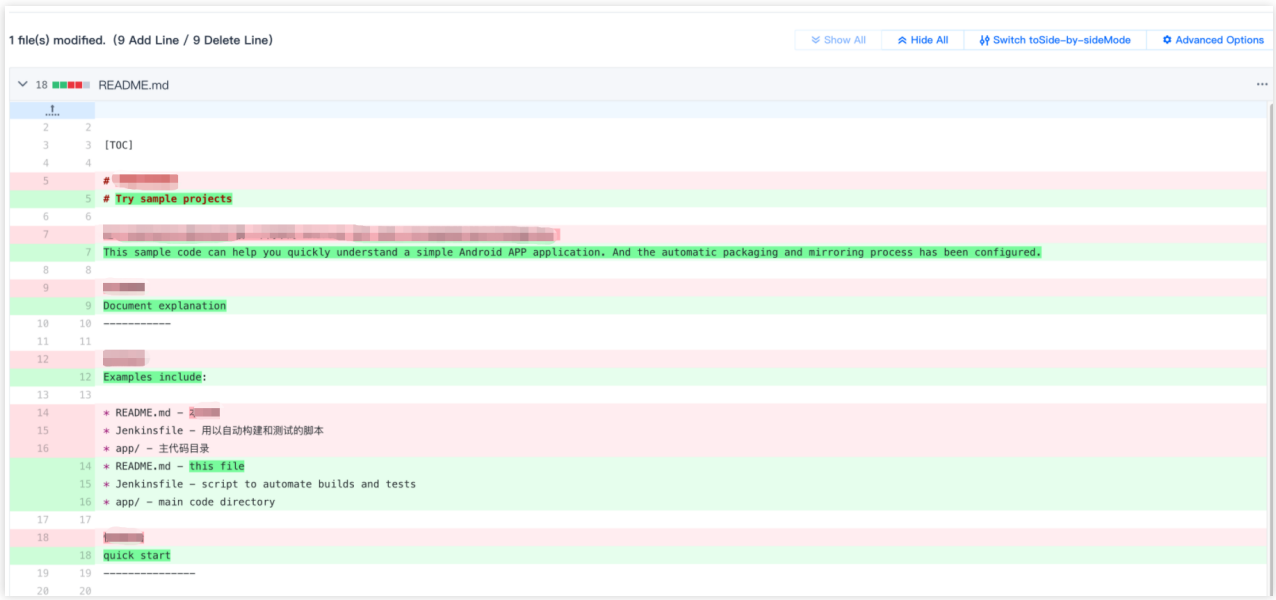


说明：

代码的行级对比支持使用 普通模式、左右模式 与 高级选项 以便更快发现代码间的异同，还支持对每行代码发起评论。

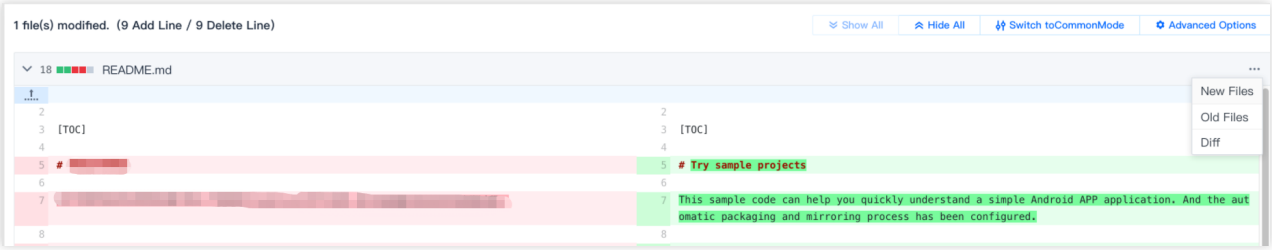
普通模式

普通模式下，代码文件中会显示该提交改动过的所有文件。展开任一文档，可查看具体的提交改动。每个文件内会显示具体的行级增删修改信息。其中，绿色代表增加，红色表示减少。



左右模式

单击**切换为左右模式**，代码行级对比由普通模式切换为左右模式。用户可清晰地对比代码改动前后的内容。改了哪一行、修改了什么内容均一目了然。您还可以选择浏览改动前/后的文件或 Diff 文件。



高级选项

高级选项支持折行显示、显示制表符及性能模式。您可以按需打开或关闭对应选项。

设置仓库基本信息

最近更新时间：2023-12-25 17:08:18

本文为您详细介绍如何设置仓库的基本信息。

进入项目

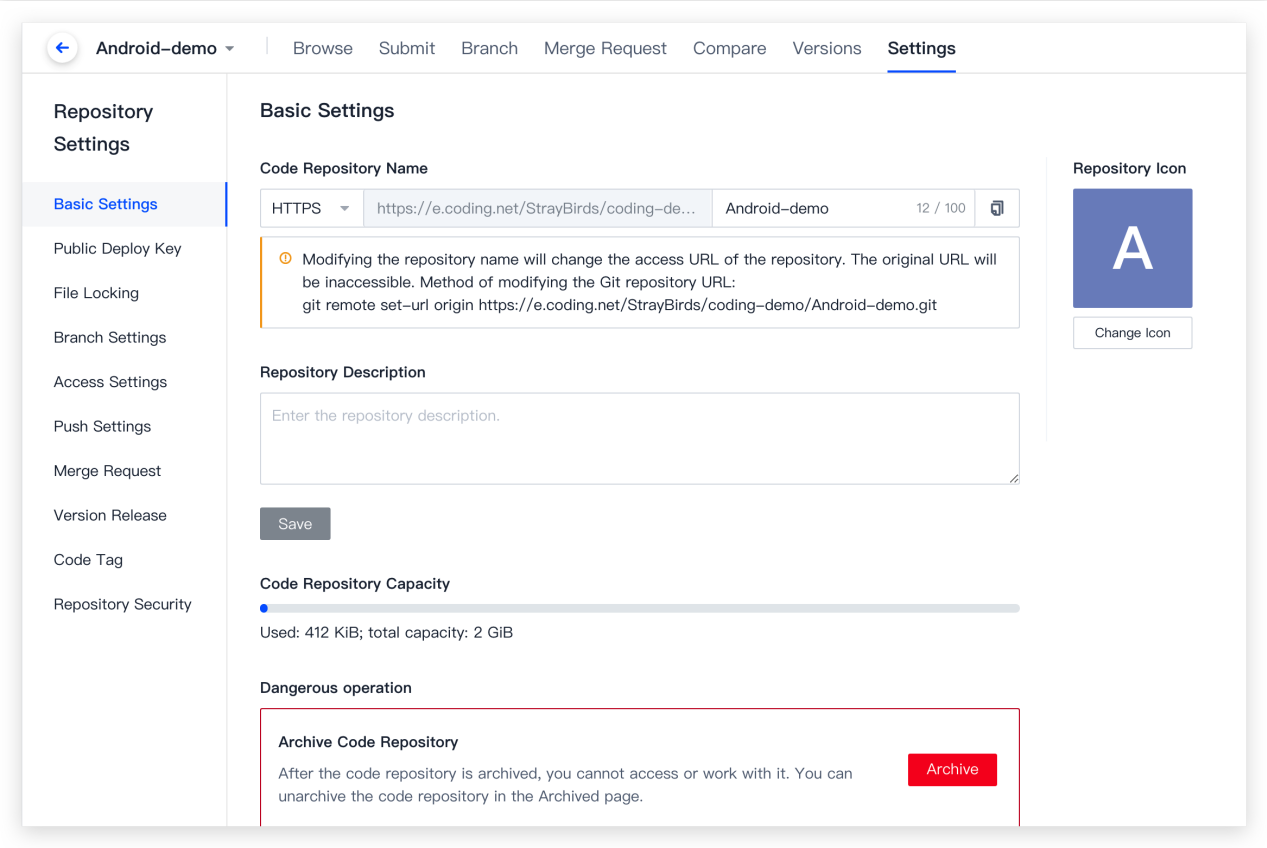
1. 登录 CODING 控制台，单击团队域名进入 CODING 使用页面。
2. 单击页面右上角的

，进入项目列表页面，单击项目图标进入目标项目。
3. 选择左侧菜单**代码仓库**，进入代码仓库首页。
4. 若左侧未显示代码仓库，需项目管理员前往**项目设置** > **项目与成员** > **功能开关**打开功能开关。

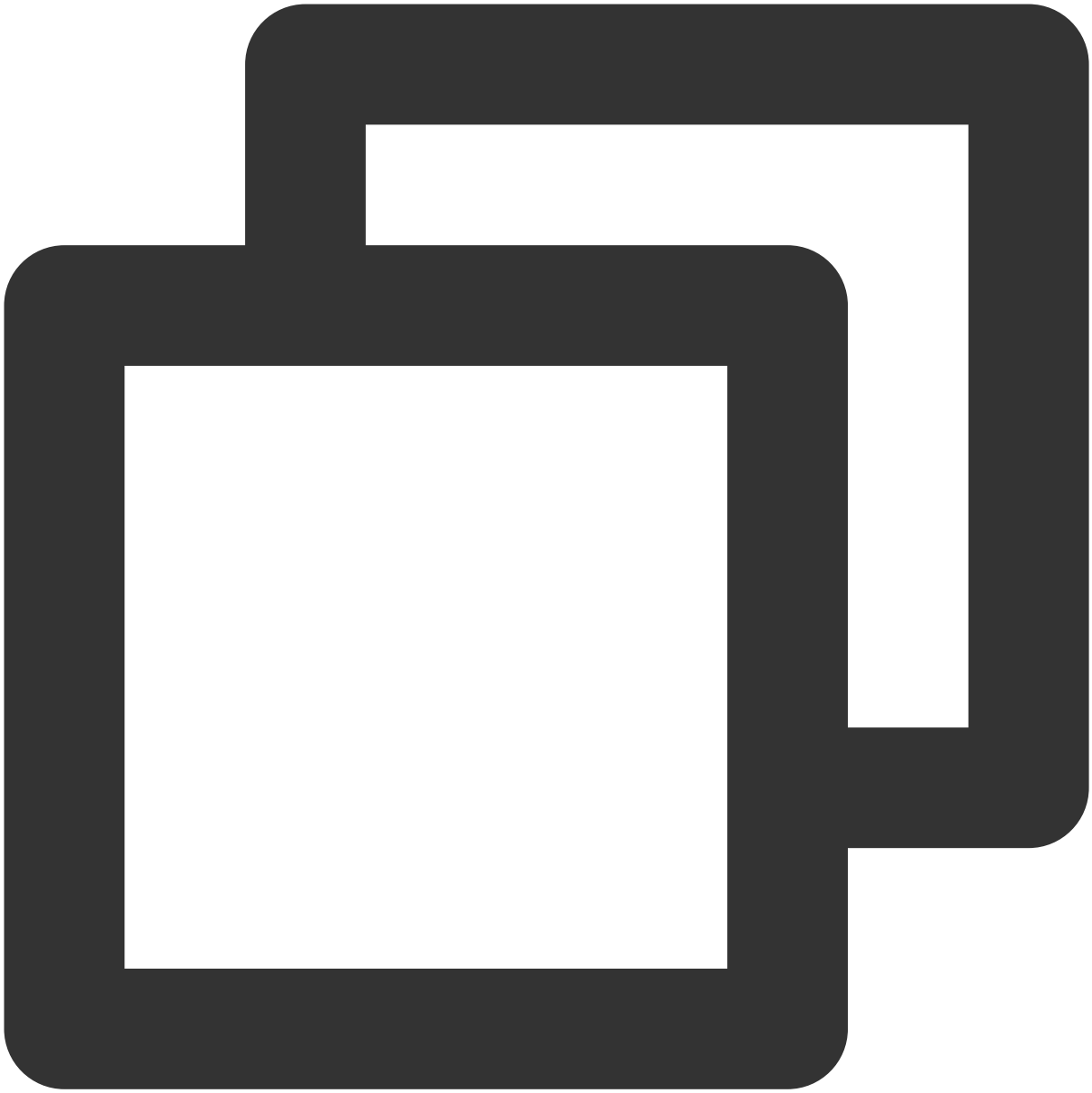
操作说明

1. 进入一个项目之后，在**代码仓库**模块，单击任一仓库进入其详情页面。在**设置**页签即可对当前代码仓库进行设置。

说明：
只有项目管理员才能进入仓库的设置页面。



2. 您可以在**基本设置**页面修改您的仓库名称与图标。修改名称后会导致仓库的访问 URL 改变，在此之前的 URL 地址将失效。修改名称后需在您的本地仓库与新的地址相匹配。



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

除此之外，您还可以在该页面添加仓库描述、查看代码仓库用量情况。如有必要，您还可以归档、重置或删除代码仓库。

归档仓库

最近更新时间：2023-12-25 17:08:18

本文为您详细介绍如何归档仓库。

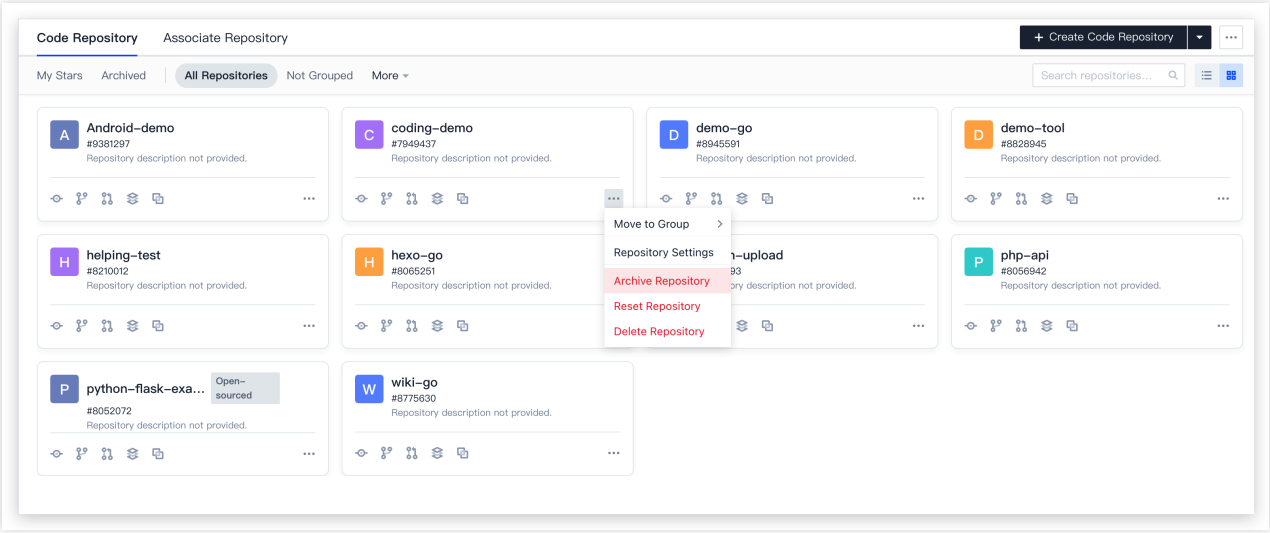
进入项目

1. 登录 CODING 控制台，单击团队域名进入 CODING 使用页面。
2. 单击页面右上角的

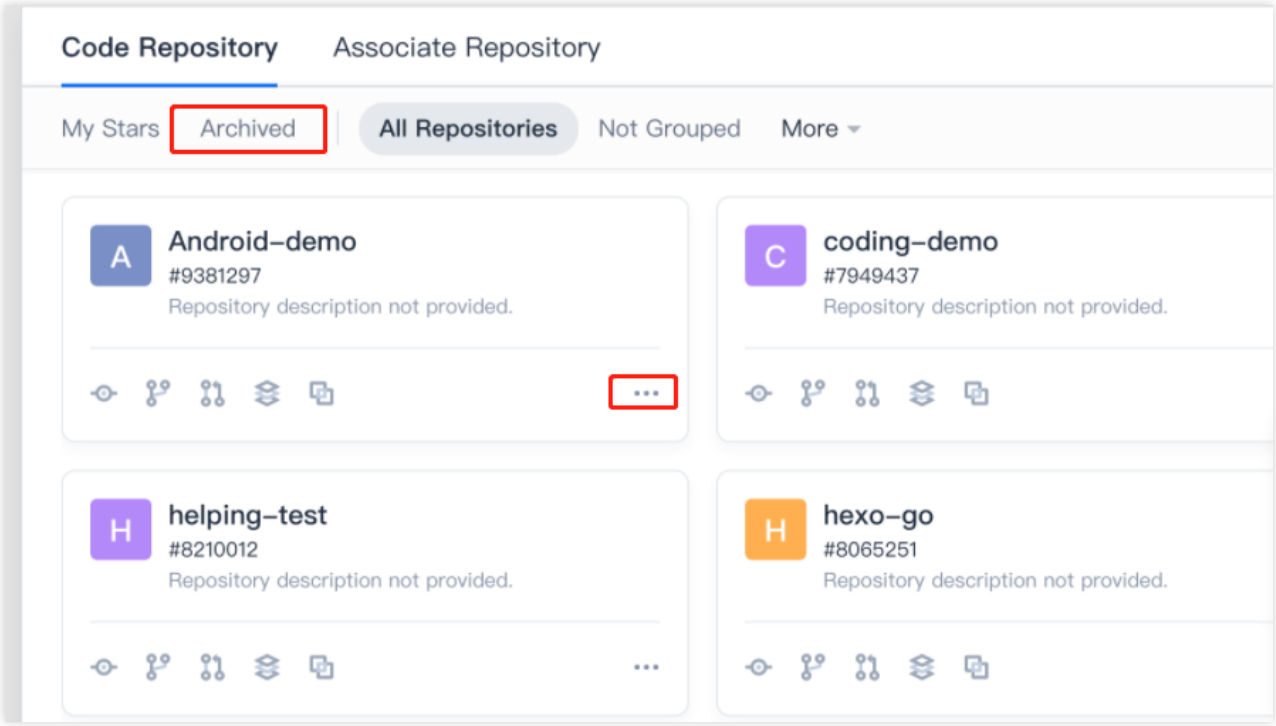
，进入项目列表页面，单击项目图标进入目标项目。
3. 选择左侧菜单**代码仓库**，进入代码仓库首页。
4. 若左侧未显示代码仓库，需项目管理员前往**项目设置** > **项目与成员** > **功能开关**打开功能开关。

操作说明

1. 如需归档仓库，在代码仓库列表单击操作按钮并按照提示确认即可。



2. 归档代码仓库之后，将会自动阻隔 Git 或 Web 端的代码仓库访问请求，用户无法继续访问和操作该仓库。已归档的仓库只能在**已归档**分类内查看，若希望恢复仓库的正常访问，需重新解除归档。


Code Repository
Associate Repository
My Stars
Archived
All Repositories
Not Grouped
More
A
Android-demo
#9381297
Repository description not provided.
C
coding-demo
#7949437
Repository description not provided.
H
helping-test
#8210012
Repository description not provided.
H
hexo-go
#8065251
Repository description not provided.

删除与重置仓库

最近更新时间：2023-12-25 17:08:18

本文为您详细介绍如何删除或重置仓库。

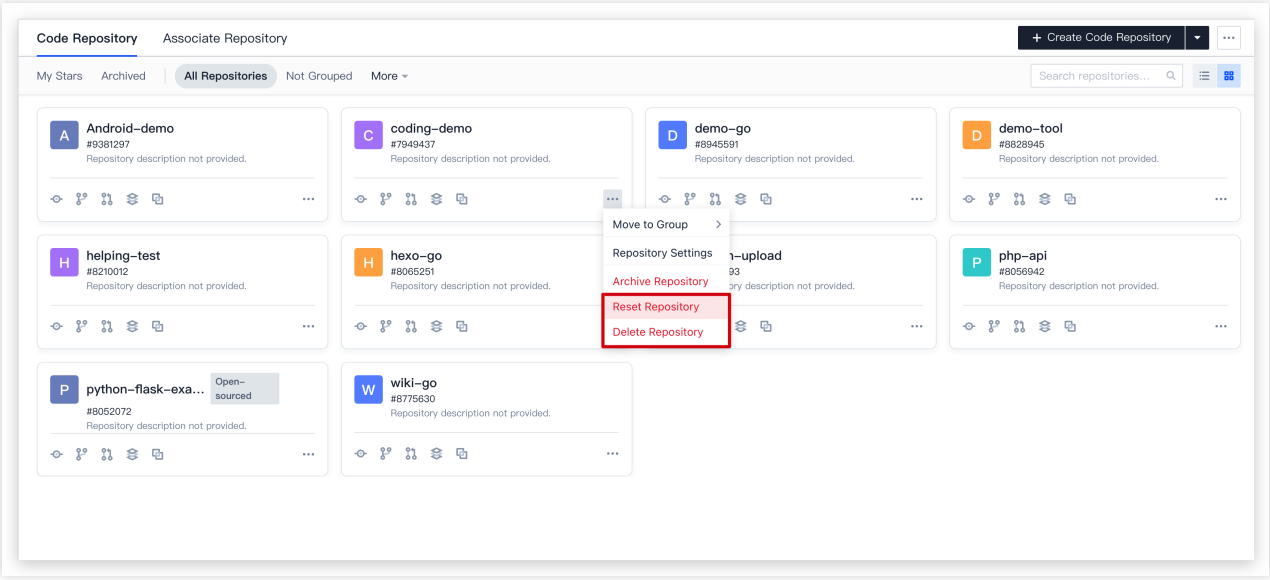
进入项目

1. 登录 CODING 控制台，单击团队域名进入 CODING 使用页面。
2. 单击页面右上角的

，进入项目列表页面，单击项目图标进入目标项目。
3. 选择左侧菜单**代码仓库**，进入代码仓库首页。
4. 若左侧未显示代码仓库，需项目管理员前往**项目设置 > 项目与成员 > 功能开关**打开功能开关。

操作说明

如需重置或删除仓库，在代码仓库列表单击操作按钮并按照提示确认即可。删除后的代码仓库将移入回收站保留 30 天，项目管理员可在到期前进入回收站恢复。逾期将会彻底删除仓库，不可恢复。



重置代码仓库后，将重置仓库内的所有代码，包括代码分支、合并请求、代码版本。重置后数据无法恢复。代码仓库将会被重置为空仓库。

删除代码仓库将会永久删除当前代码仓库内的所有代码，包括代码分支、合并请求、代码版本。删除后数据无法恢复。代码仓库将无法通过任何途径访问。

恢复已删除仓库

最近更新时间：2023-12-25 17:08:18

本文为您详细介绍如何恢复已删除仓库。

进入项目

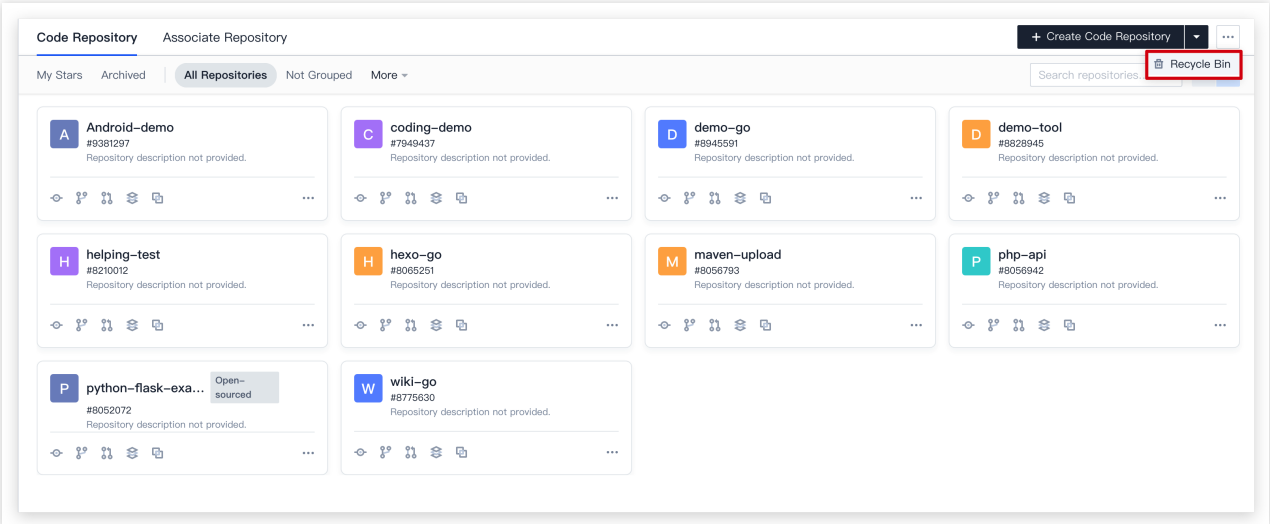
1. 登录 CODING 控制台，单击团队域名进入 CODING 使用页面。
2. 单击页面右上角的

，进入项目列表页面，单击项目图标进入目标项目。
3. 选择左侧菜单**代码仓库**，进入代码仓库首页。
4. 若左侧未显示代码仓库，需项目管理员前往**项目设置** > **项目与成员** > **功能开关**打开功能开关。

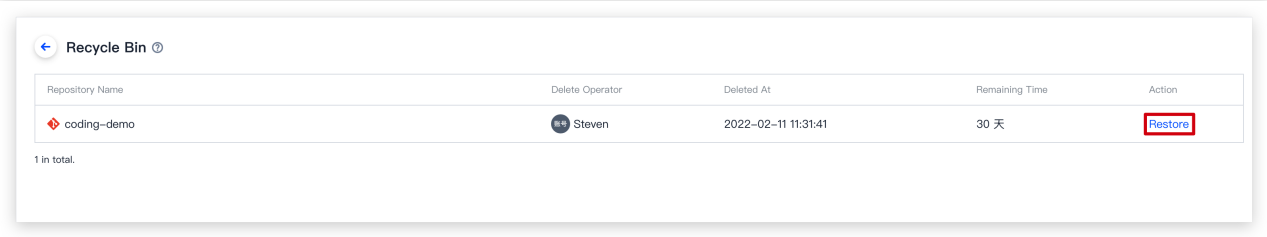
操作说明

对于已删除的代码仓库，项目管理员可在 30 天内进入回收站恢复。

1. 在**代码仓库**页面，将鼠标悬浮在右上角的更多操作按钮，将出现回收站图标。



2. 进入回收站后，选择要恢复的代码仓库，单击**恢复**即可恢复该仓库。


Recycle Bin
Repository Name
Delete Operator
Deleted At
Remaining Time
Action
coding-demo
Steven
2022-02-11 11:31:41
30 天
Restore
1 in total.

管理代码仓库卡片

最近更新时间：2023-12-25 17:08:18

本文为您详细介绍如何管理代码仓库卡片。

进入项目

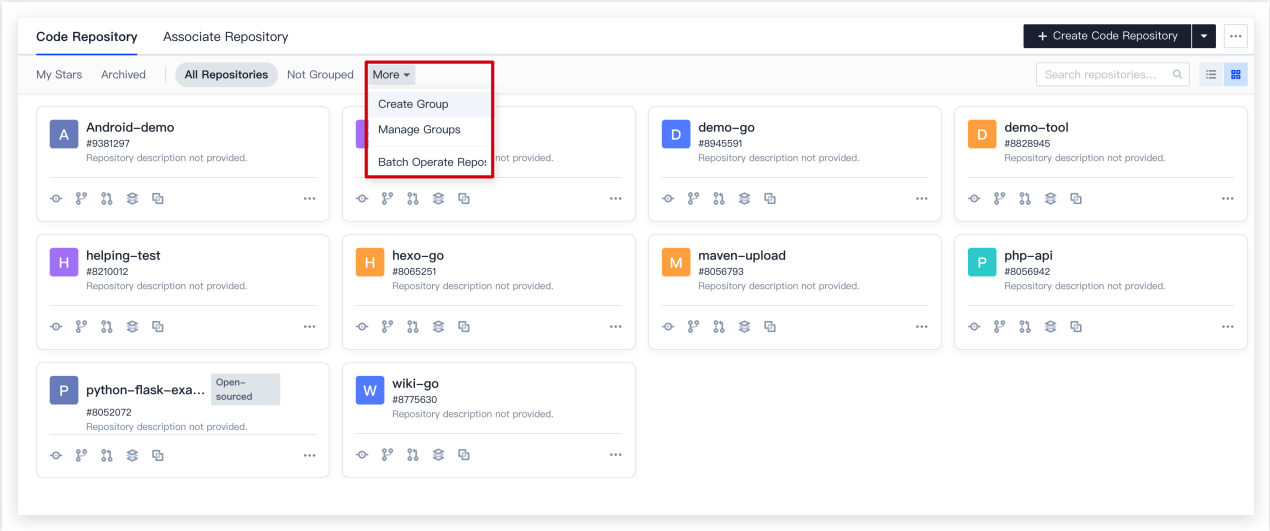
1. 登录 CODING 控制台，单击团队域名进入 CODING 使用页面。
2. 单击页面右上角的

，进入项目列表页面，单击项目图标进入目标项目。
3. 选择左侧菜单**代码仓库**，进入代码仓库首页。
4. 若左侧未显示代码仓库，需项目管理员前往**项目设置 > 项目与成员 > 功能开关**打开功能开关。

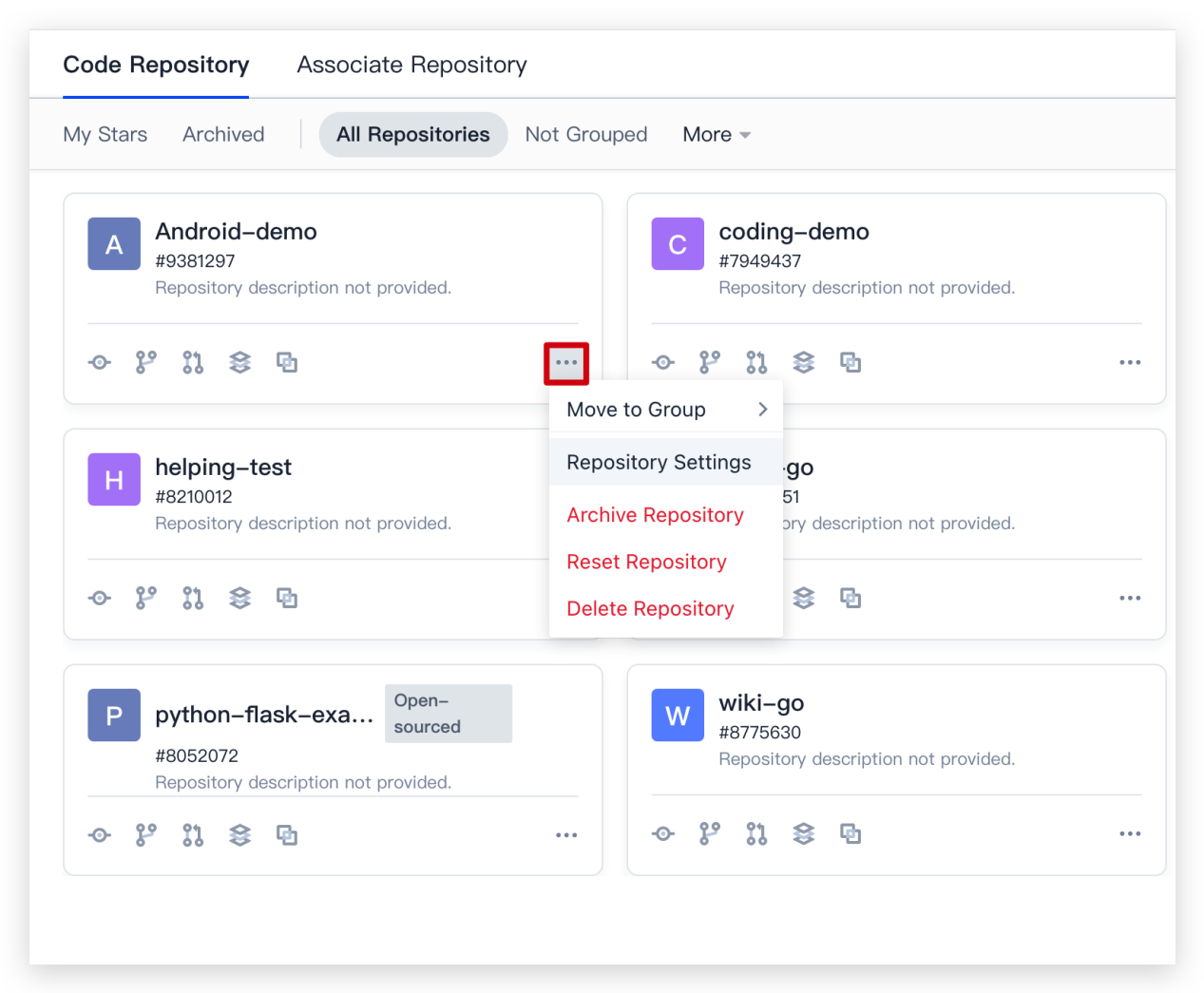
每个代码仓库默认以卡片的形式展示。您可以对仓库进行分组与排序，并通过卡片上的 快速入口 进行对应的操作。

分组与排序

在**代码仓库**页面，您可以创建仓库分组，然后通过批量操作将多个仓库同时添加到同一分组。通过**管理分组**，您可以调整仓库分组的展示顺序。

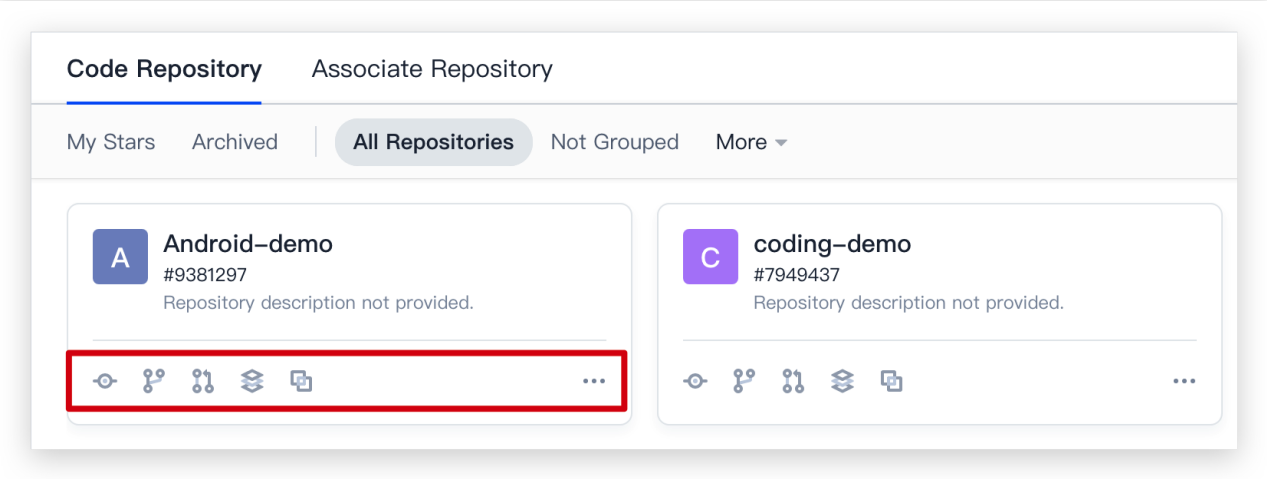


针对单个代码仓库，可通过更多操作按钮将其添加至任一分组。

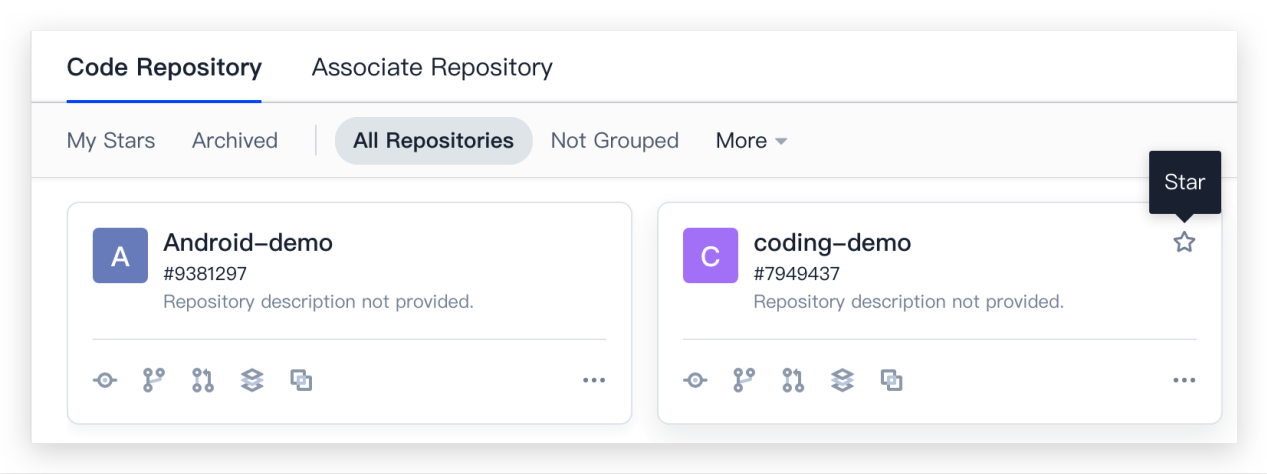


仓库卡片快速操作

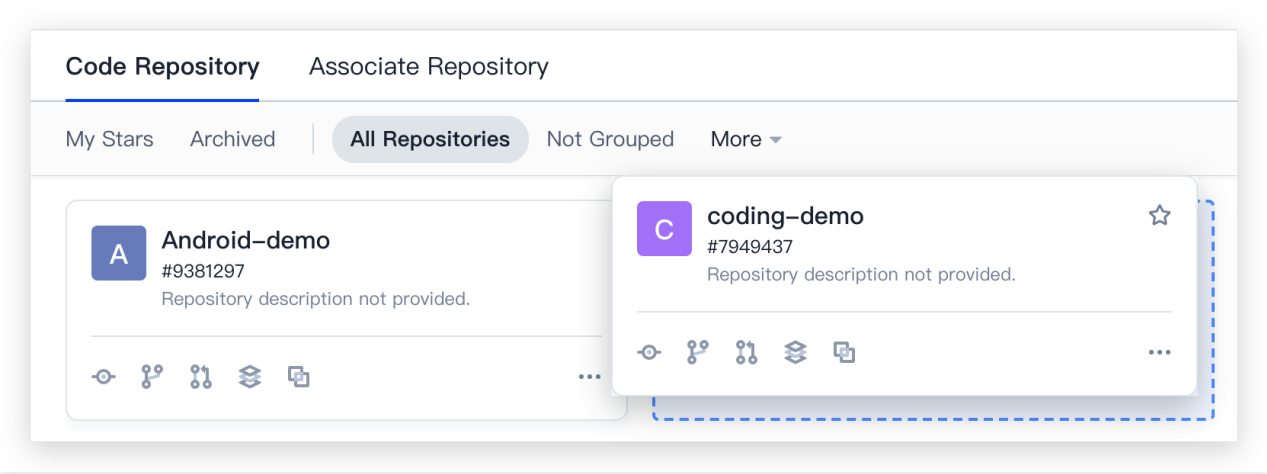
每一个代码仓库卡片内置快捷入口，以便快速执行所需操作。快捷入口的说明如下图所示：



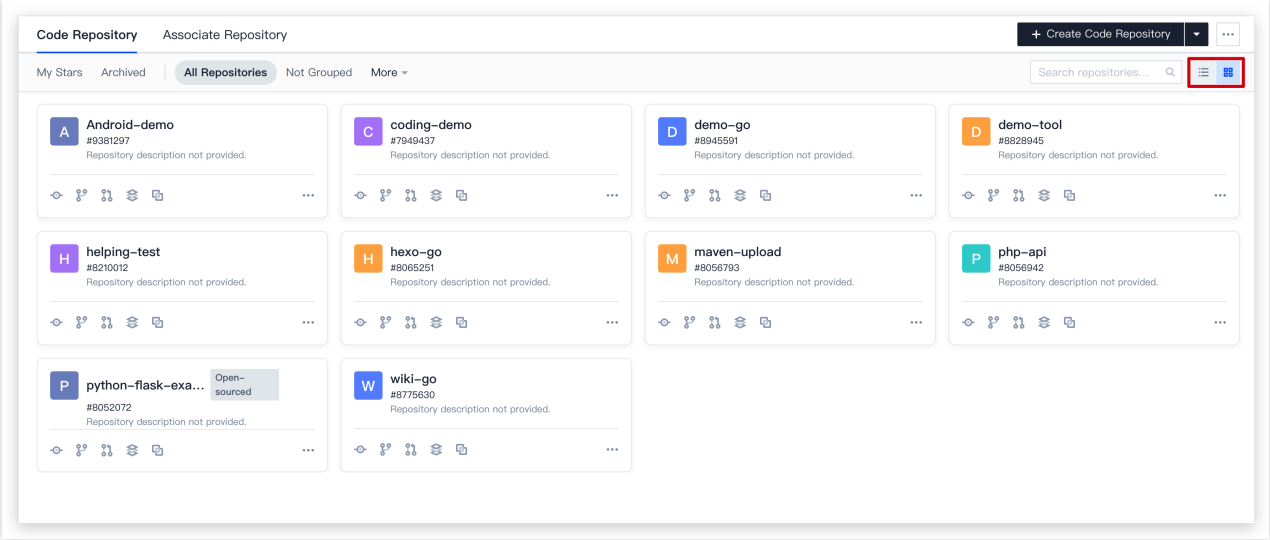
鼠标移至某一卡片上，单击右上角的星按钮可以将其设为星标仓库。



拖动卡片可以调整显示顺序。



单击右上方的显示切换按钮，可以在列表与卡片展示模式之间进行切换。



通过本地命令行管理仓库

最近更新时间：2023-12-25 17:08:18

本文为您详细介绍如何通过本地命令行管理仓库。

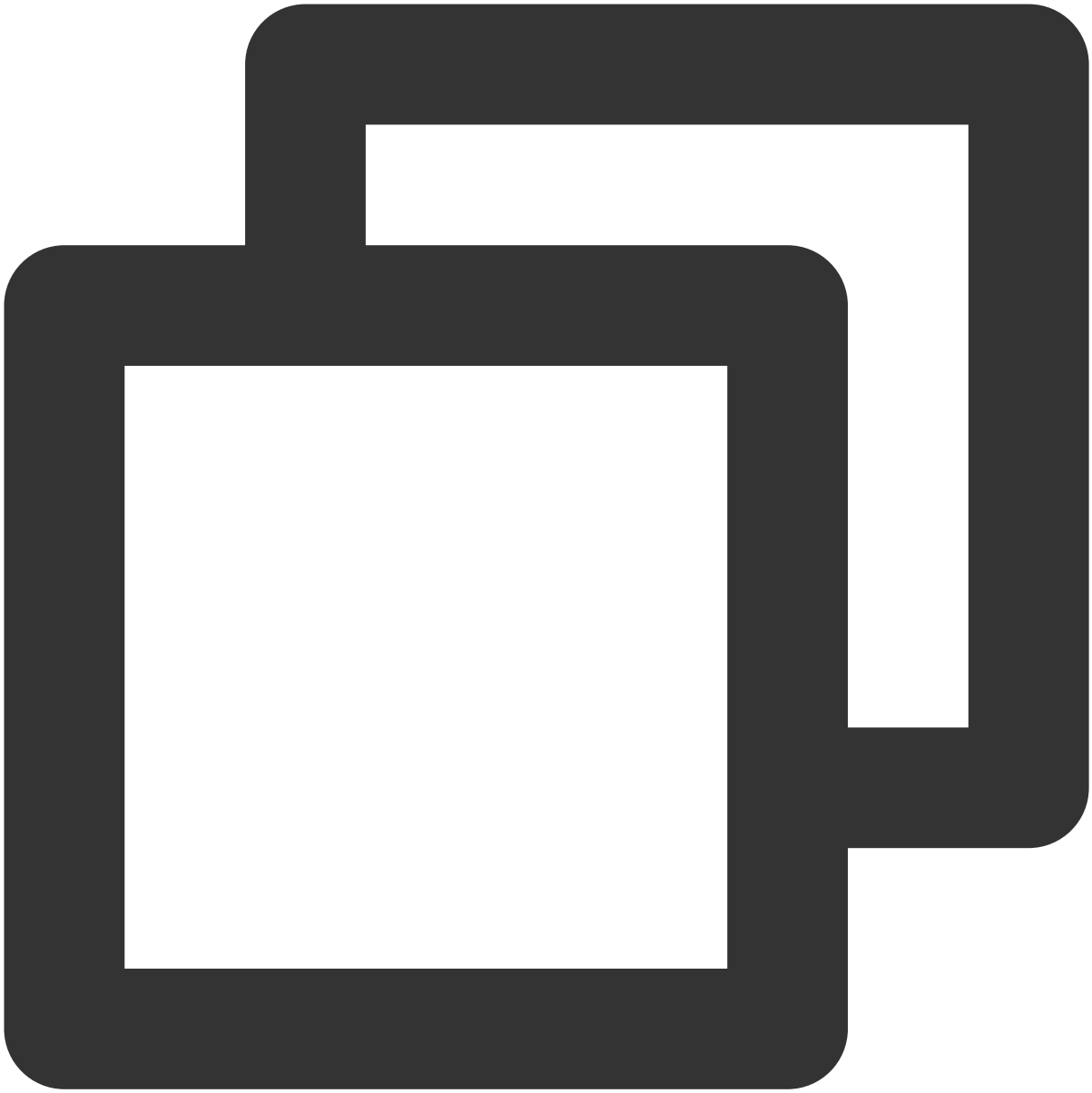
进入项目

1. 登录 CODING 控制台，单击团队域名进入 CODING 使用页面。
2. 单击页面右上角的

，进入项目列表页面，单击项目图标进入目标项目。
3. 选择左侧菜单**代码仓库**，进入代码仓库首页。
4. 若左侧未显示代码仓库，需项目管理员前往**项目设置** > **项目与成员** > **功能开关**打开功能开关。

从远程仓库获取数据

您可以使用 `git clone` 命令克隆远程仓库至本地，并自动与之关联。



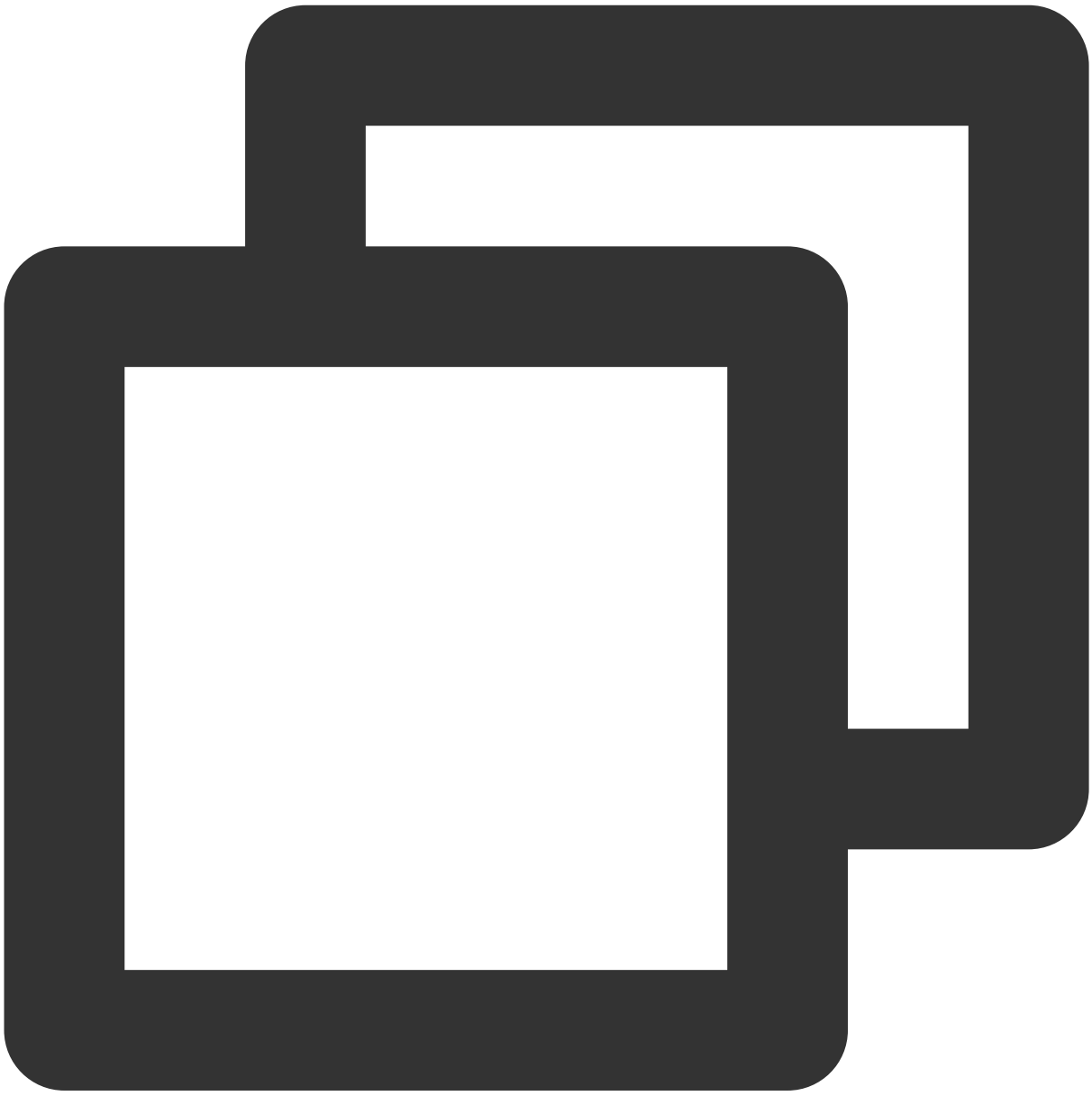
```
git clone [remote-name]
```

推送数据到远程仓库

使用 `git push [remote-name] [branch-name]` 可以将本地仓库中的数据推送到远程仓库，如： `git push learn-git master` 会将本地的仓库数据推送到远程仓库的 “master” 分支。

重命名远程仓库

使用 `git remote rename [old-name] [new-name]` 命令修改某个远程仓库在本地的简称，例如想把 `learn-git` 改成 `origin`，可以运行：

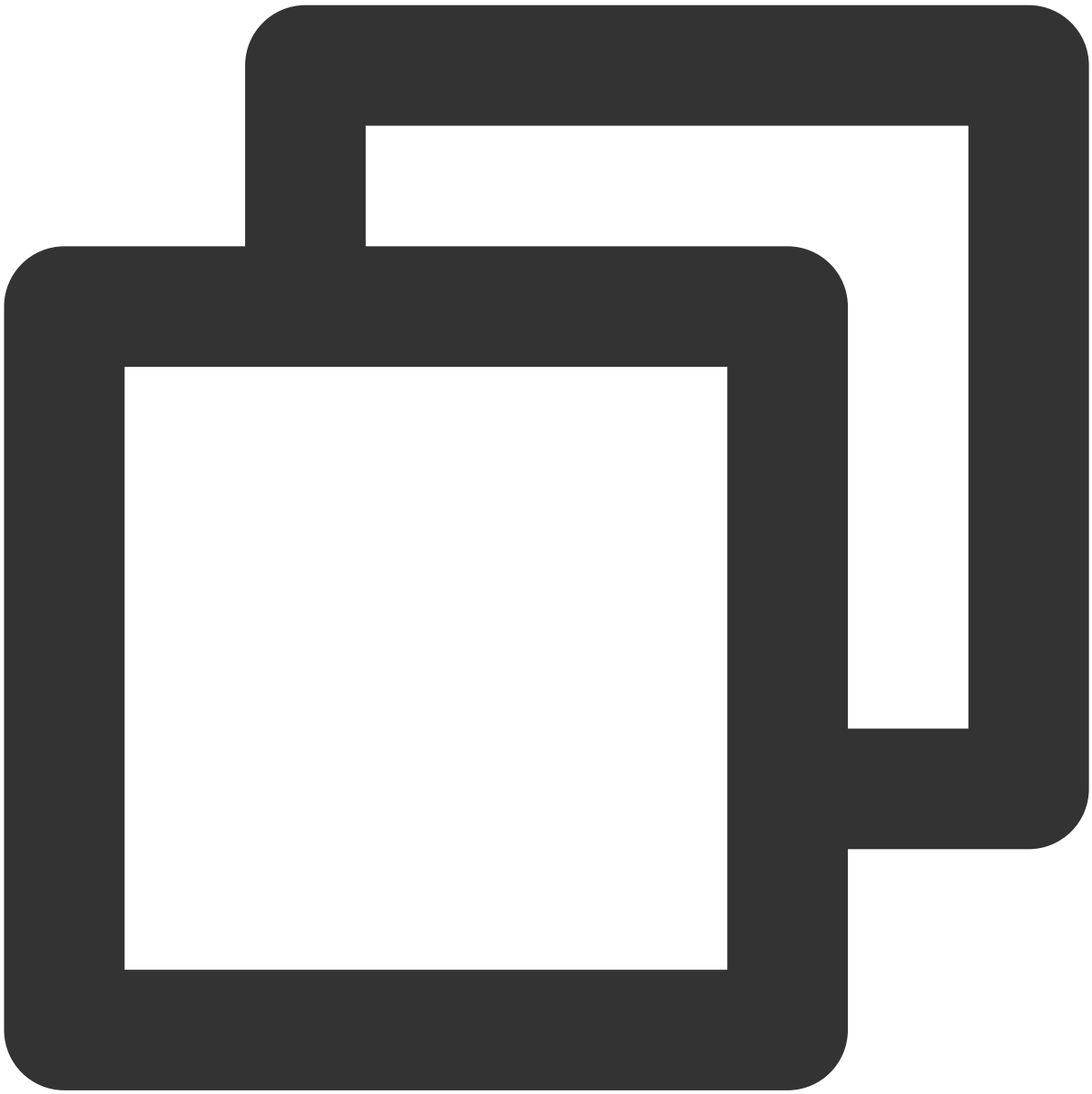


```
git remote rename learn-git origin
```

重命名远程仓库之后，若在使用 Git 命令时需要指定远程仓库的名字，请使用更新后的命名。

解除远程仓库关联

例如要解除和远程仓库 “origin” 的关联，运行：



```
git remote rm origin
```

注意：

此命令是解除了本地仓库和远程仓库的关联，并不是删除了远程仓库的数据。如需了解更多常用的 Git 命令，参见 Git 常用命令。

SVN 仓库使用
创建 SVN 仓库

最近更新时间：2023-12-25 17:08:18

本文为您详细介绍如何创建 SVN 仓库。

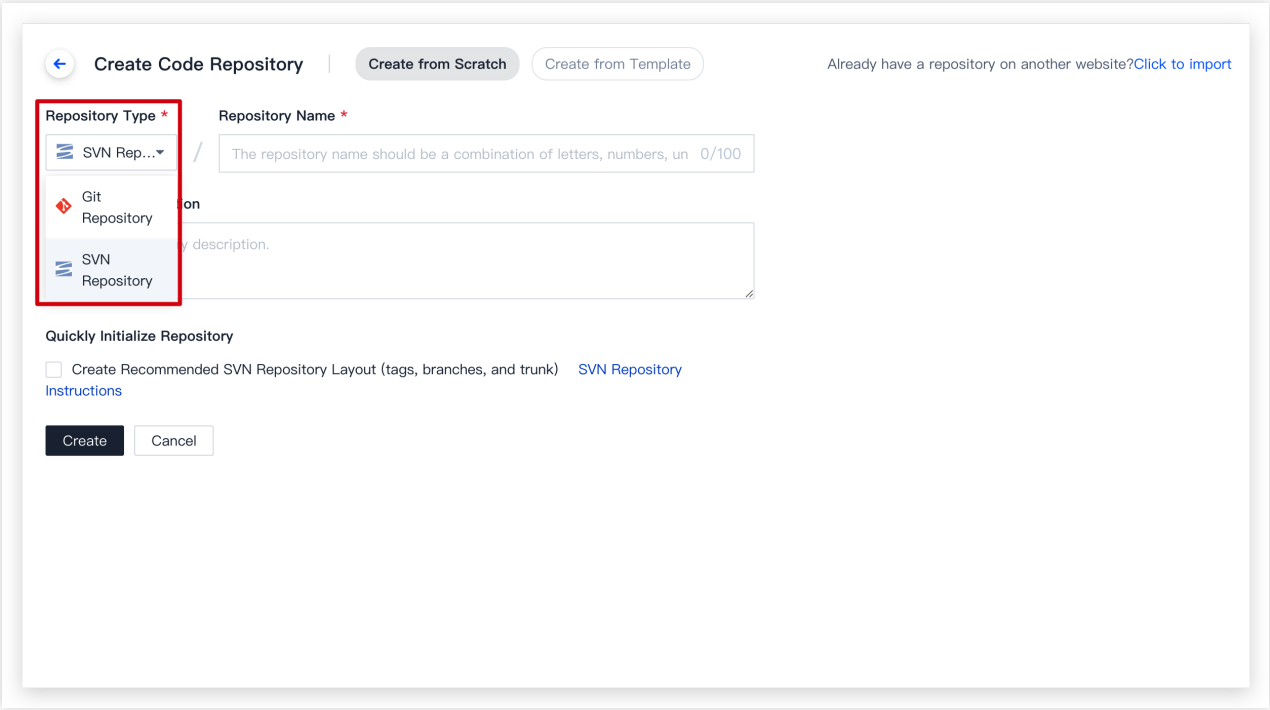
进入项目

1. 登录 CODING 控制台，单击团队域名进入 CODING 使用页面。
2. 单击页面右上角的

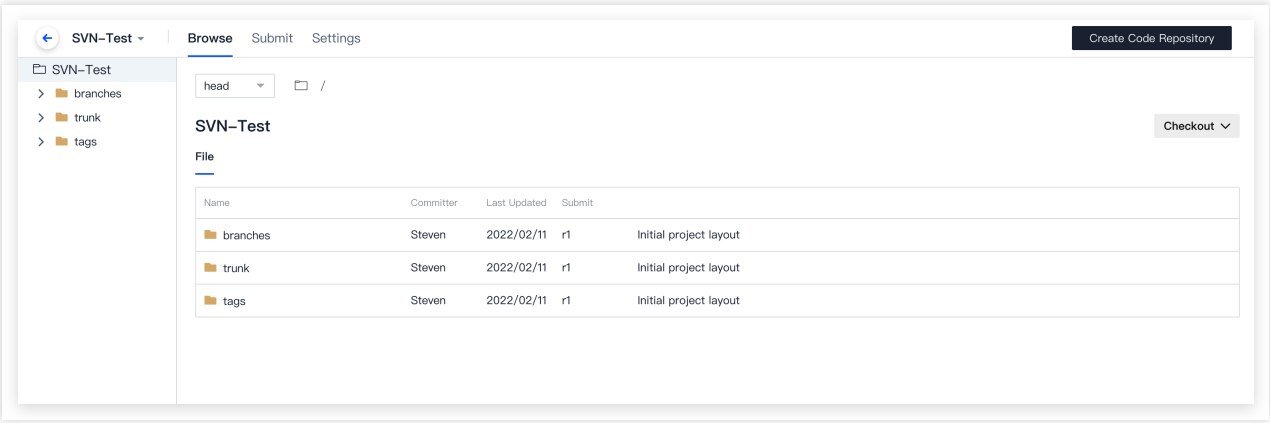
，进入项目列表页面，单击项目图标进入目标项目。
3. 选择左侧菜单**代码仓库**，进入代码仓库首页。
4. 若左侧未显示代码仓库，需项目管理员前往**项目设置** > **项目与成员** > **功能开关**打开功能开关。

操作说明

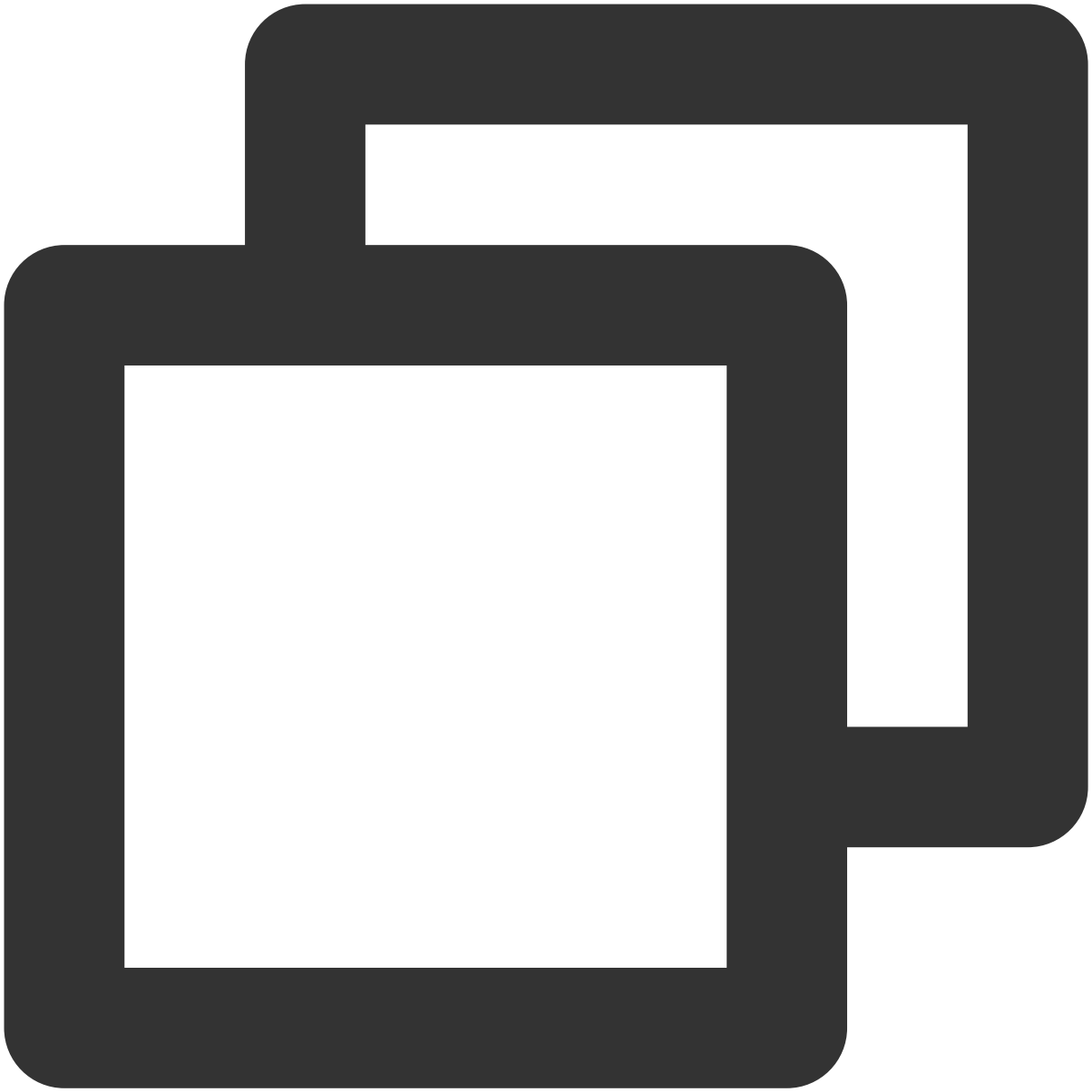
目前 CODING 已支持原生的 SVN 仓库，客户端通过 SVN+SSH 协议连接到 CODING 的服务器，数据传输全程走 SSH 加密通道。
1. 进入一个项目之后，单击左侧导航栏**代码仓库**进入代码仓库管理页面。
2. 单击页面右上角**创建代码仓库**，选择仓库类型为 **SVN 仓库**。



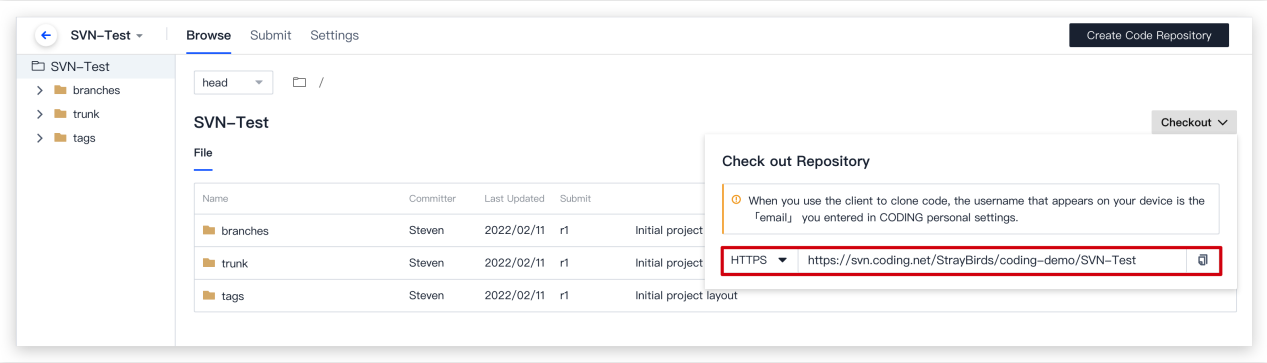
3. 选择 `创建 SVN 仓库推荐布局` 时，将会自动创建 `tags` 、 `branches` 和 `trunk` 三个目录。这是多数 SVN 仓库的推荐目录布局。仓库初始化完成之后，即可在代码浏览界面看到 SVN 仓库的内容。



在**代码浏览**界面可以看到这个仓库的 SVN 地址：



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



注意：

目前只支持在创建项目中开启 SVN 仓库，不支持在 Git 仓库中新建 SVN 仓库。

访问 SVN 仓库

最近更新时间：2023-12-25 17:08:18

SVN 仓库服务目前支持大多数主流 SVN 客户端。推荐使用各客户端的最新稳定版本。

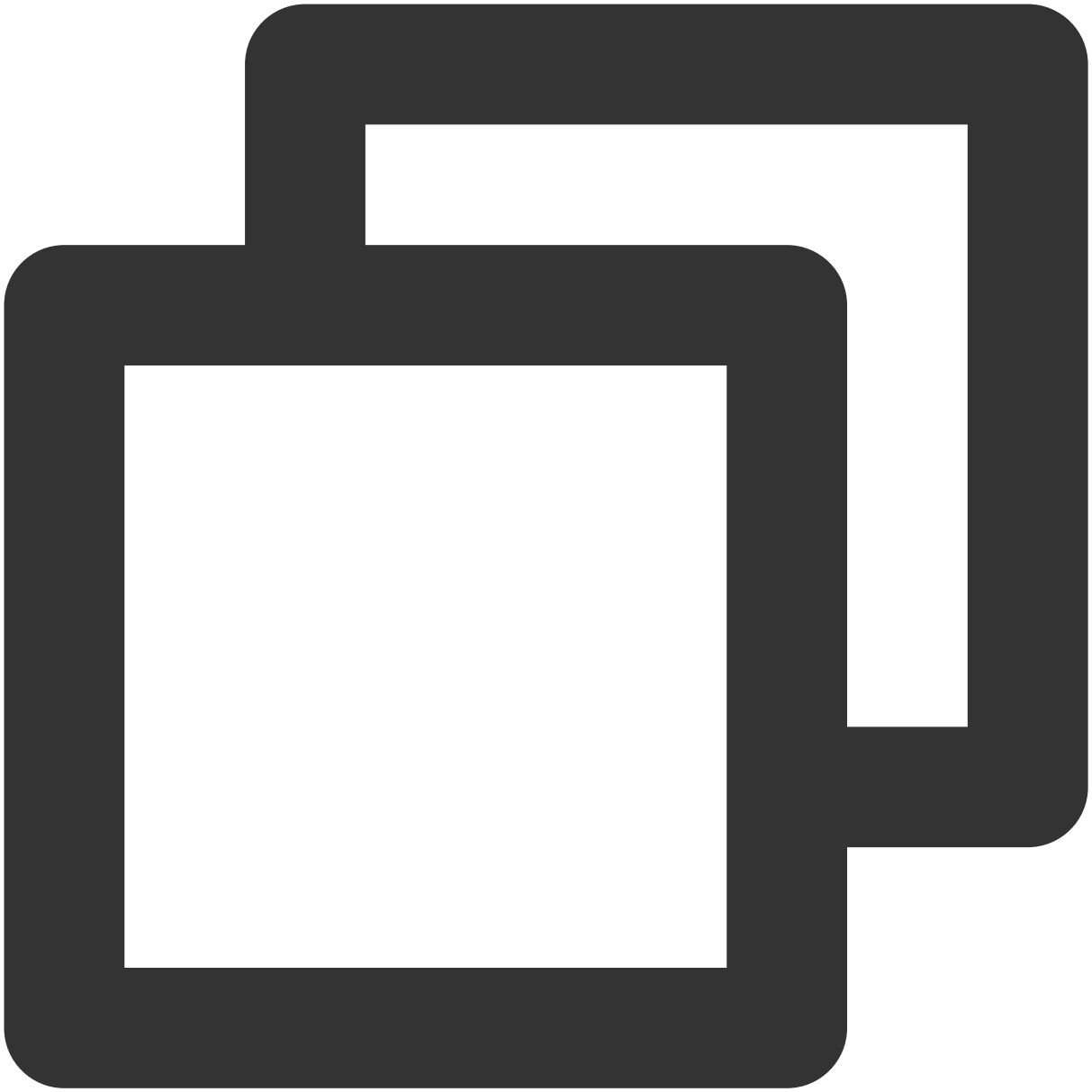
进入项目

1. 登录 CODING 控制台，单击团队域名进入 CODING 使用页面。
2. 单击页面右上角的

，进入项目列表页面，单击项目图标进入目标项目。
3. 选择左侧菜单**代码仓库**，进入代码仓库首页。
4. 若左侧未显示代码仓库，需项目管理员前往**项目设置** > **项目与成员** > **功能开关**打开功能开关。

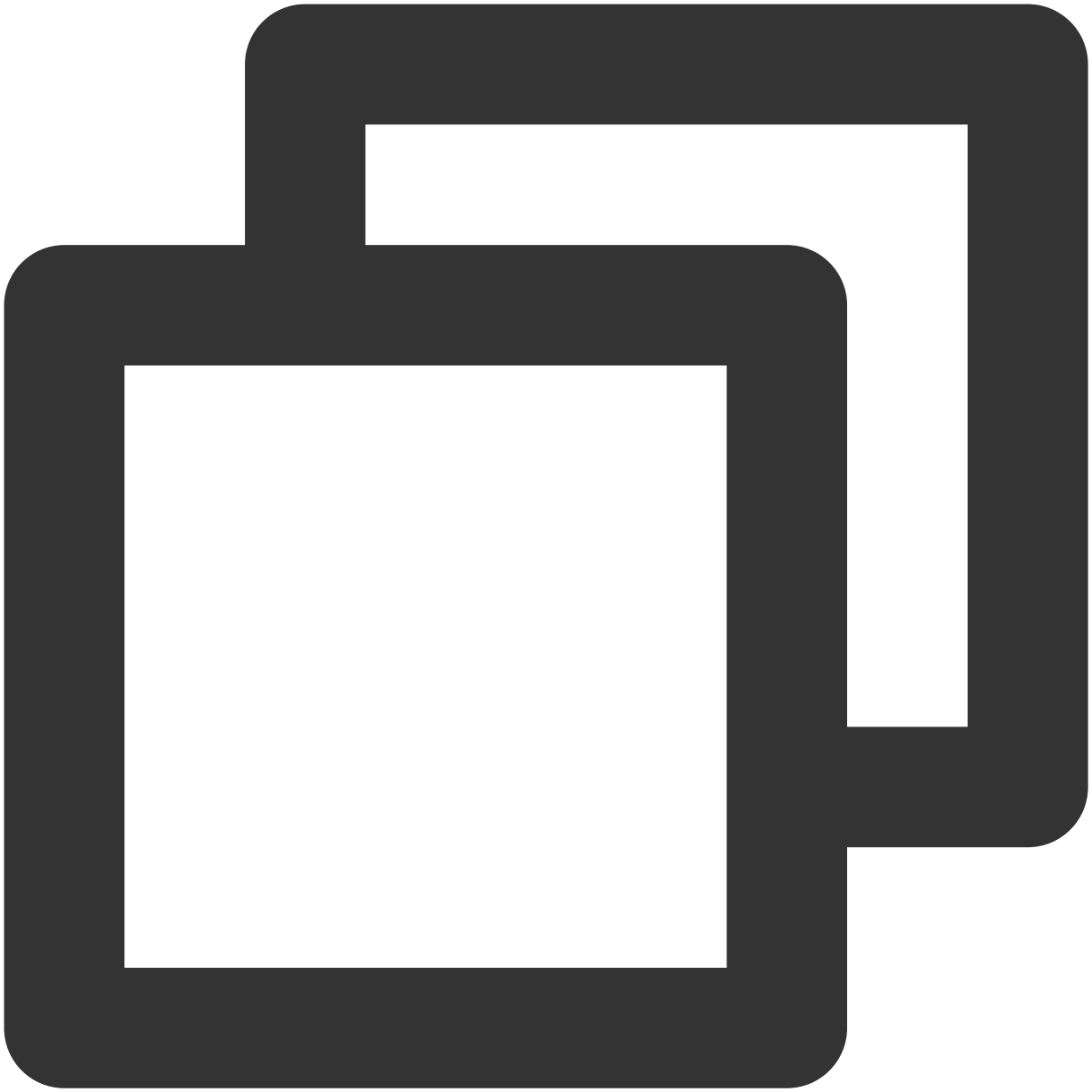
Mac 环境

在 Mac 环境，可使用 Homebrew 安装 SVN 客户端。
1. 运行下面命令安装 Homebrew：



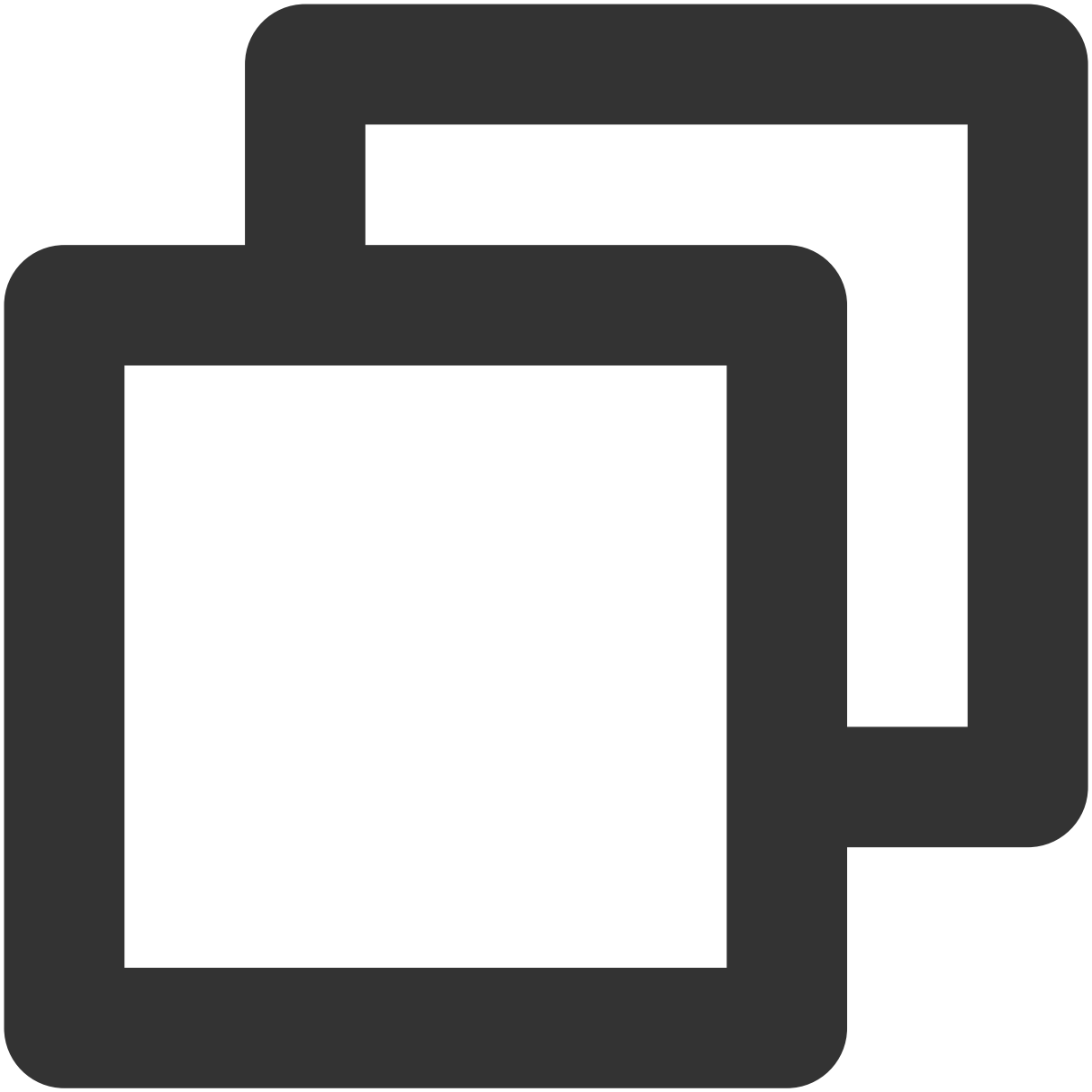
```
/usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/m
```

2. Homebrew 安装好之后，在终端输入以下命令完成 SVN 安装：



```
brew install subversion
```

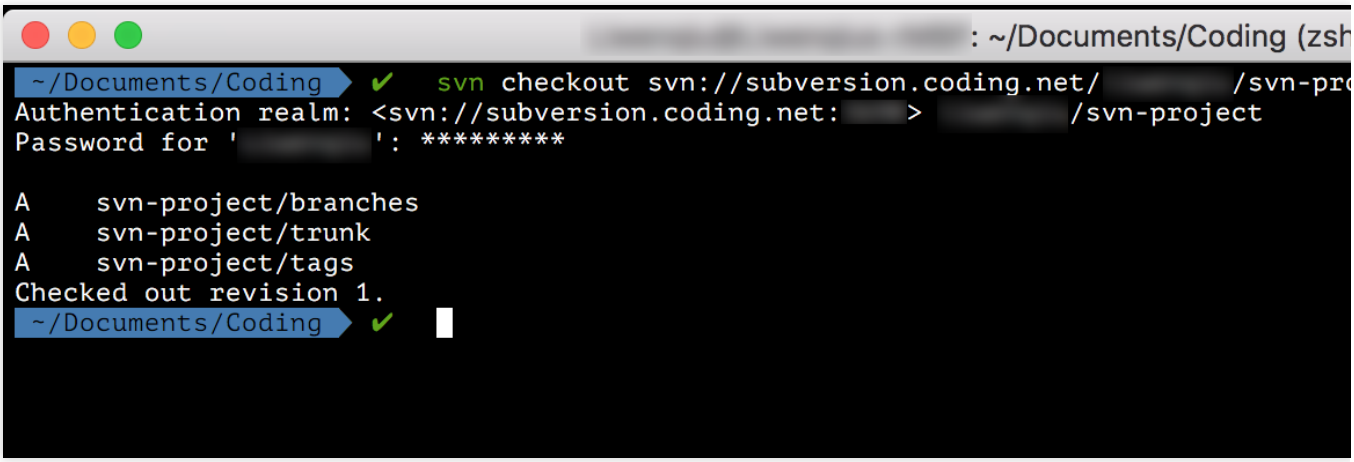
3. 使用命令 `svn --version` 验证 SVN 是否已正确安装：



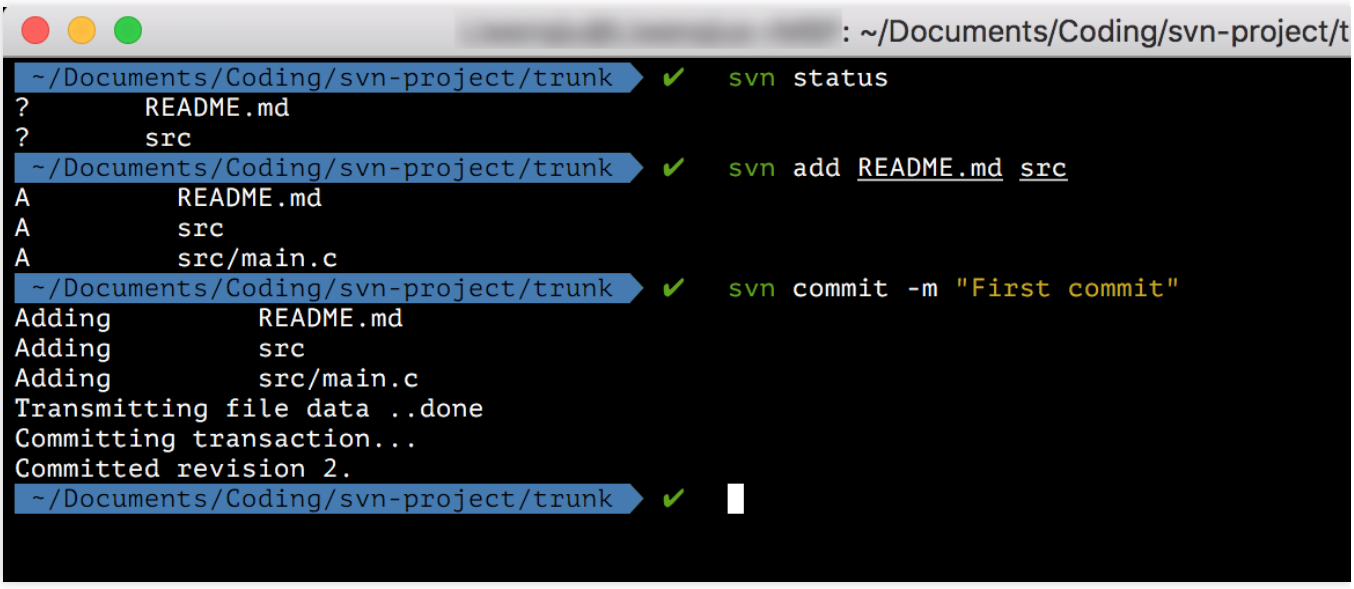
```
svn, version 1.9.7 (r1800392)
compiled Feb 28 2018, 15:54:50 on x86_64-apple-darwin17.3.0
Copyright (C) 2017 The Apache Software Foundation.
This software consists of contributions made by many people;
see the NOTICE file for more information.
Subversion is open source software, see http://subversion.apache.org/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

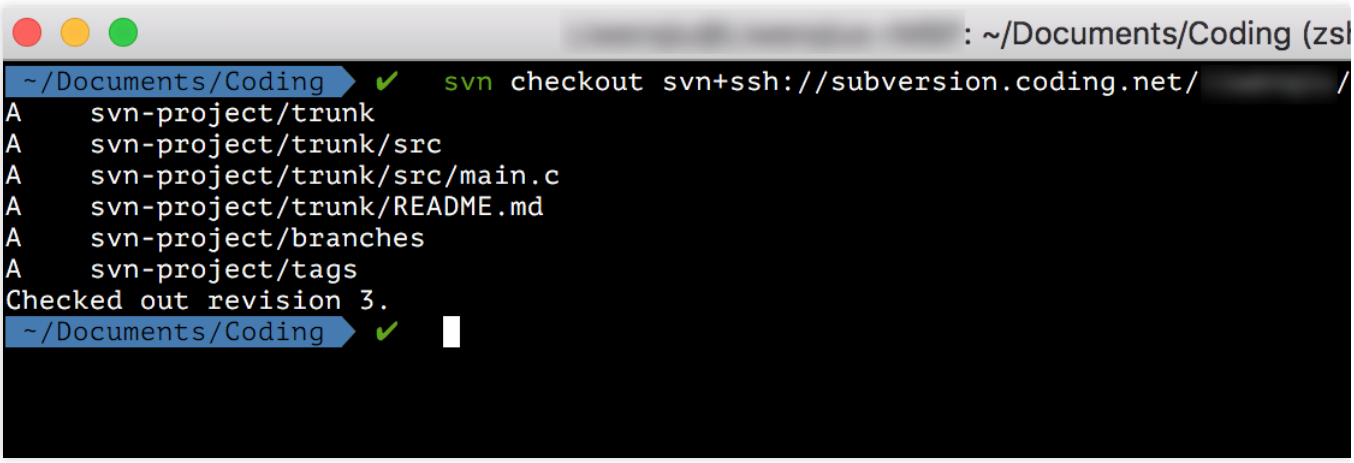
4. 使用命令 `svn checkout svn://subversion.e.coding.net/example/example-project` （请将地址替换为你的 SVN 仓库地址）来检出 SVN 仓库：



5. 接下来可以使用 `add` 、 `commit` 命令往仓库中新添加内容：



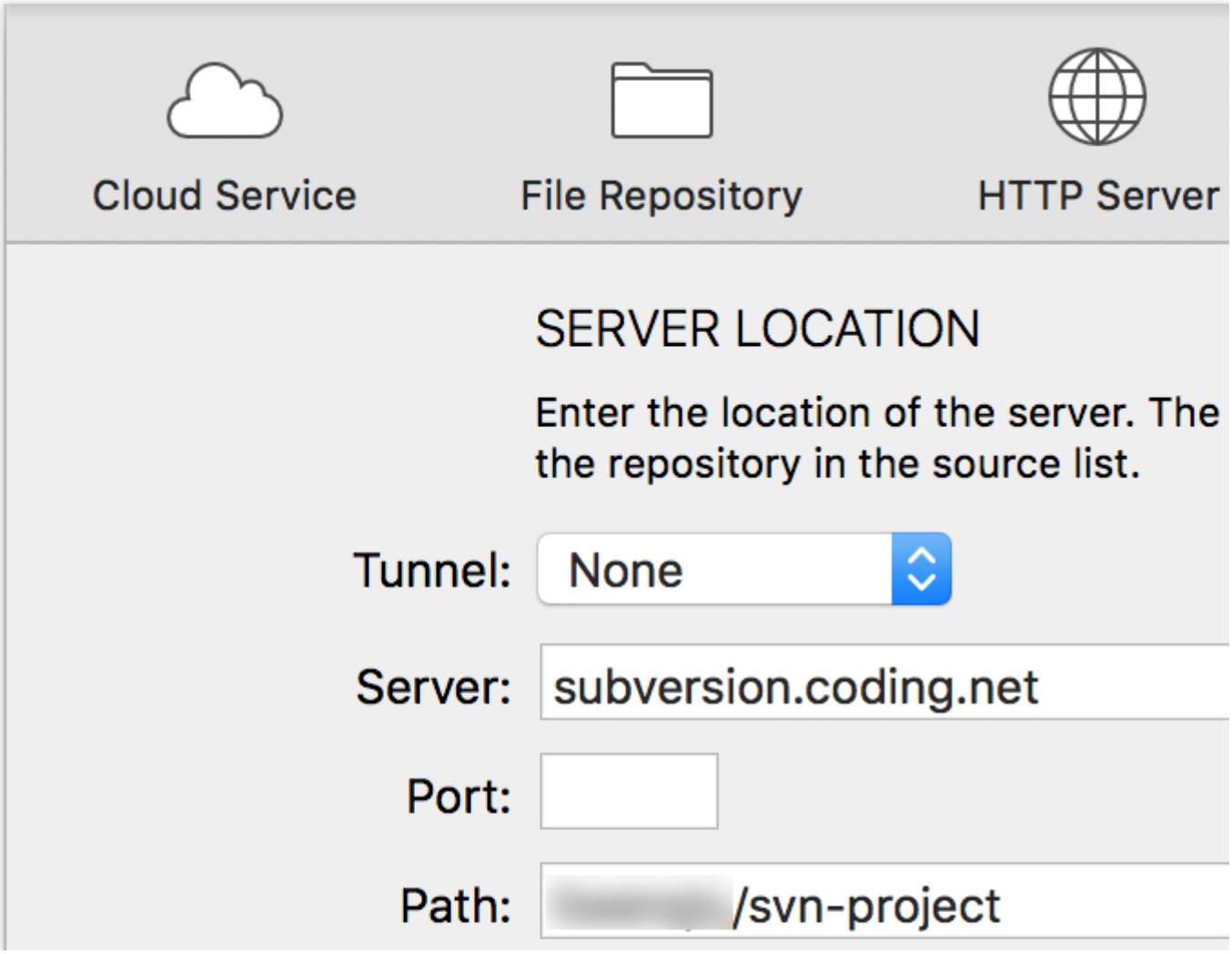
6. 除了使用 SVN 协议之外，还可以使用 `svn+ssh` 协议来访问仓库，如下图所示：

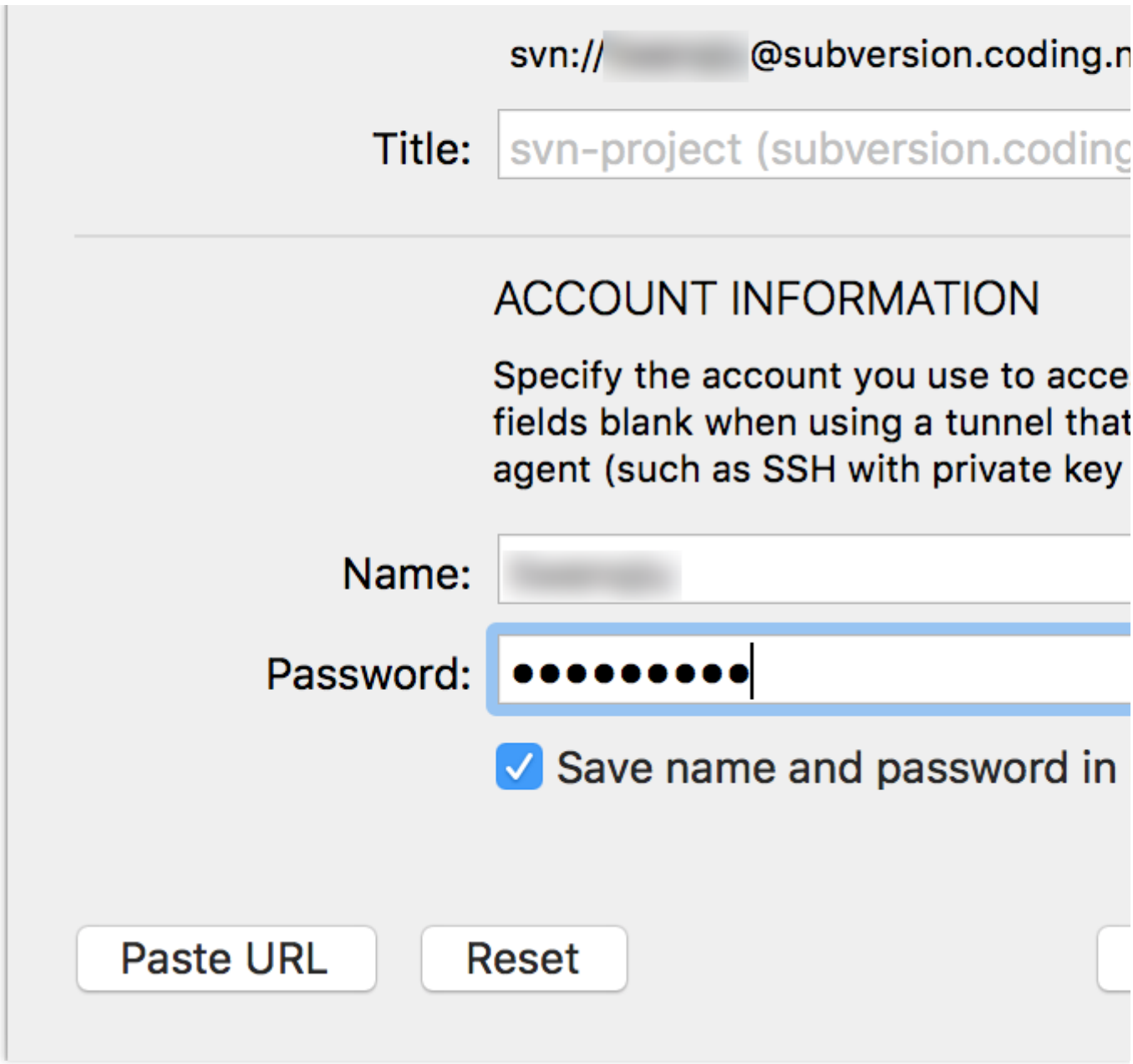


Cornerstone 工具

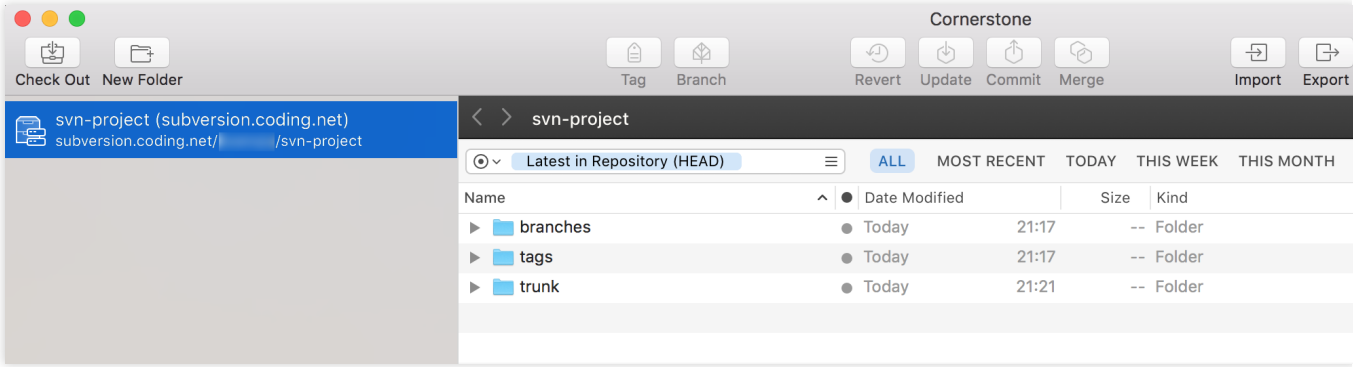
您可以通过 Cornerstone 来使用 SVN 仓库。

1. 打开 Cornerstone 后，单击 Add Repostory 来添加 SVN 仓库（请将地址替换为你的 SVN 仓库地址）引用：

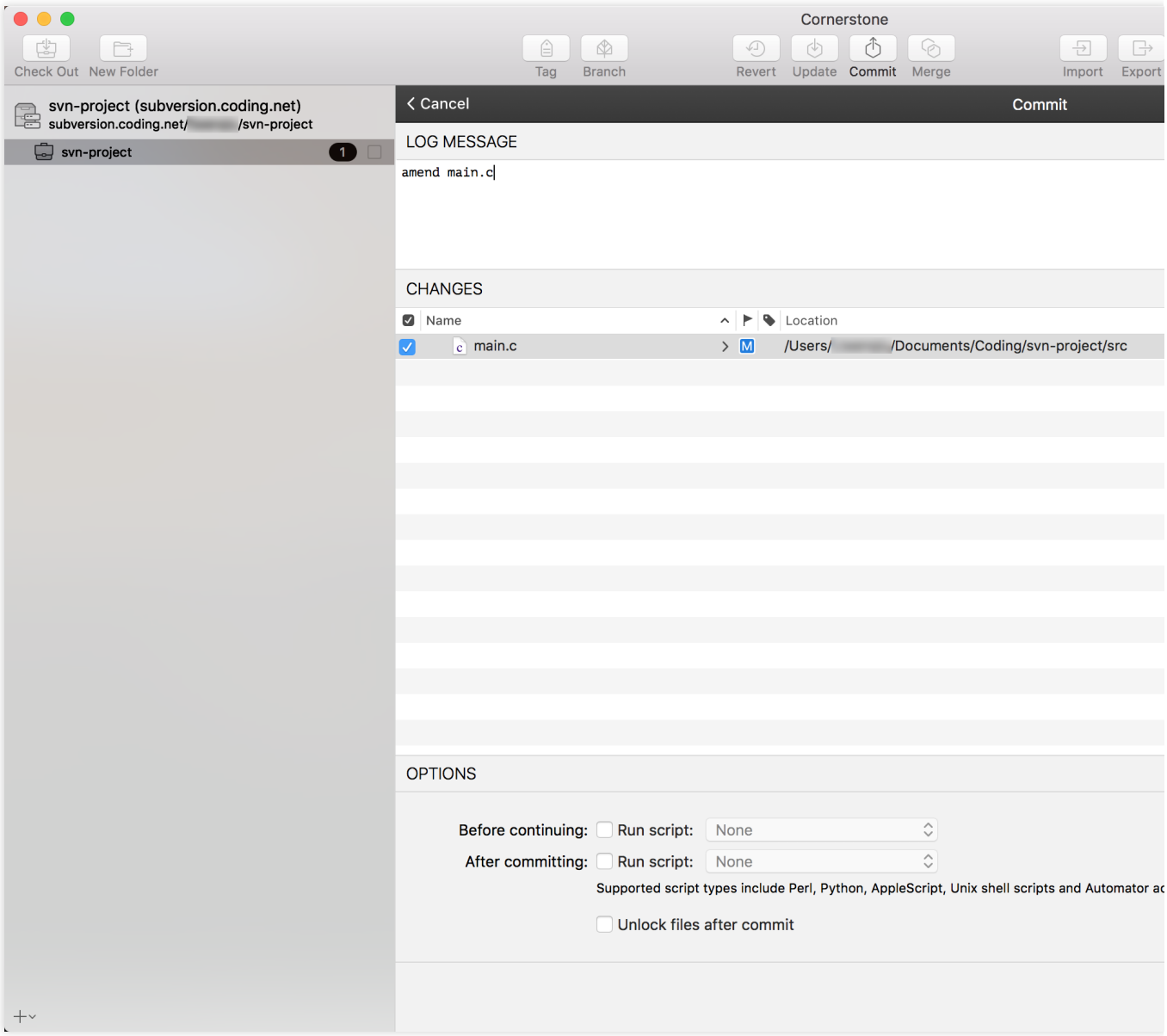




然后可以看到仓库的内容：



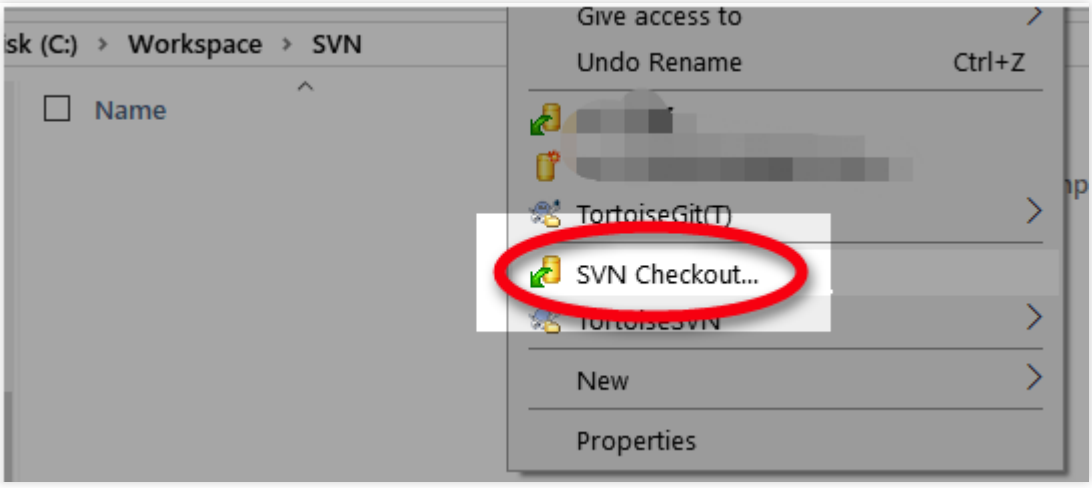
2. 把仓库 `checkout` 出来，并且编辑文件之后，就可以 `commit` 进仓库，如下图：



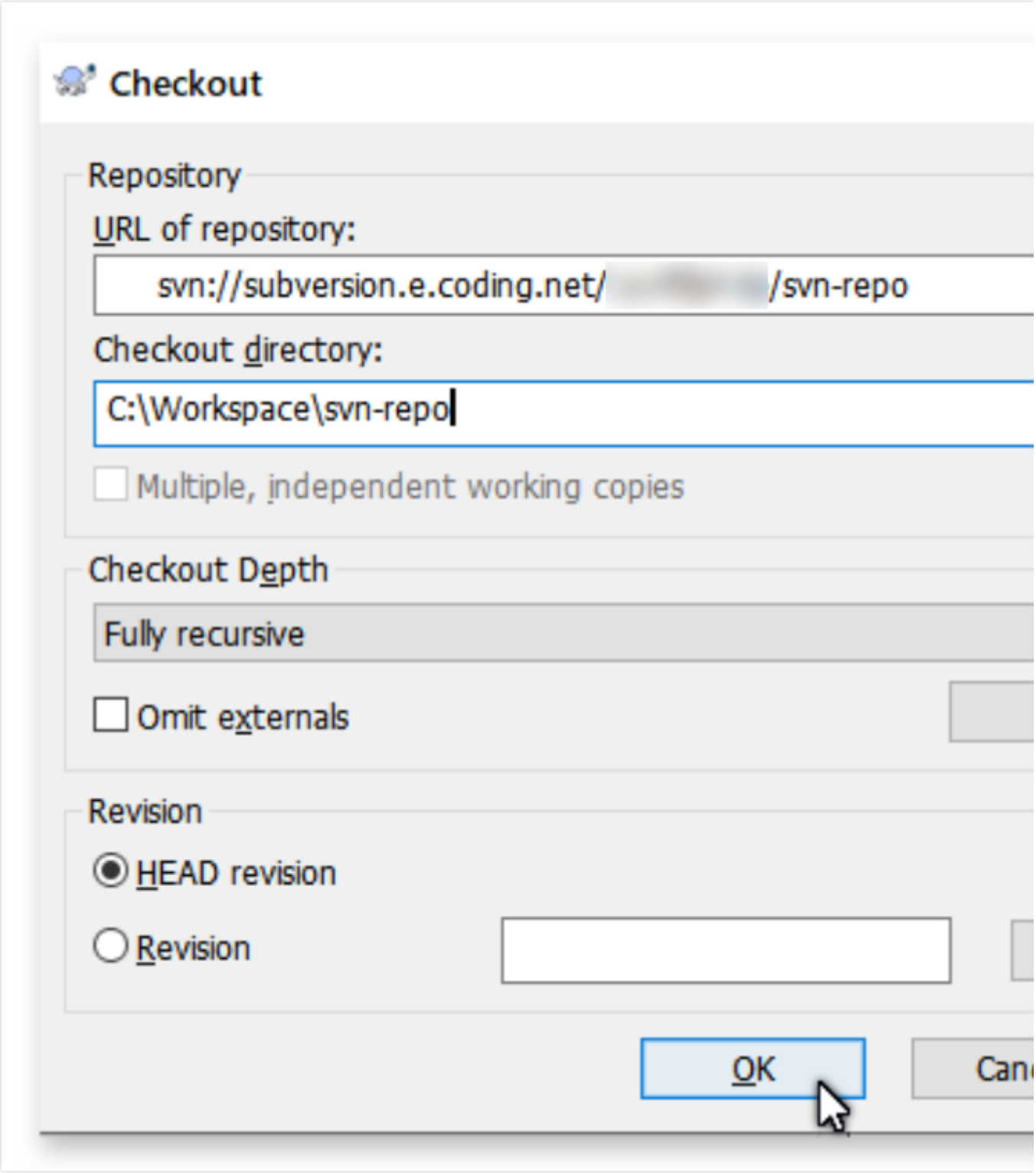
Windows 环境

在 Windows 平台，推荐使用 TortoiseSVN。

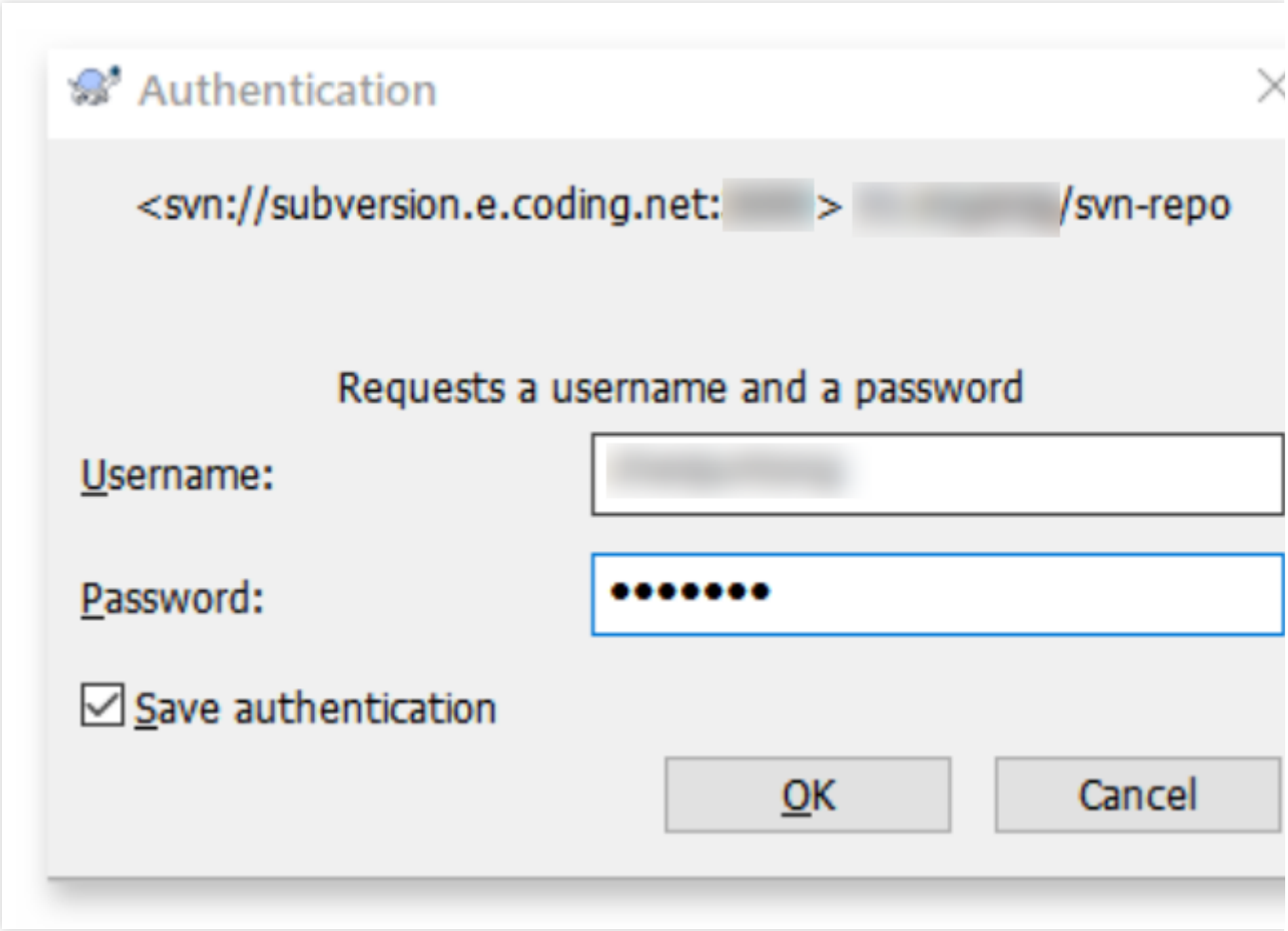
1. 下载 安装完成之后，在任意文件目录单机鼠标右键。



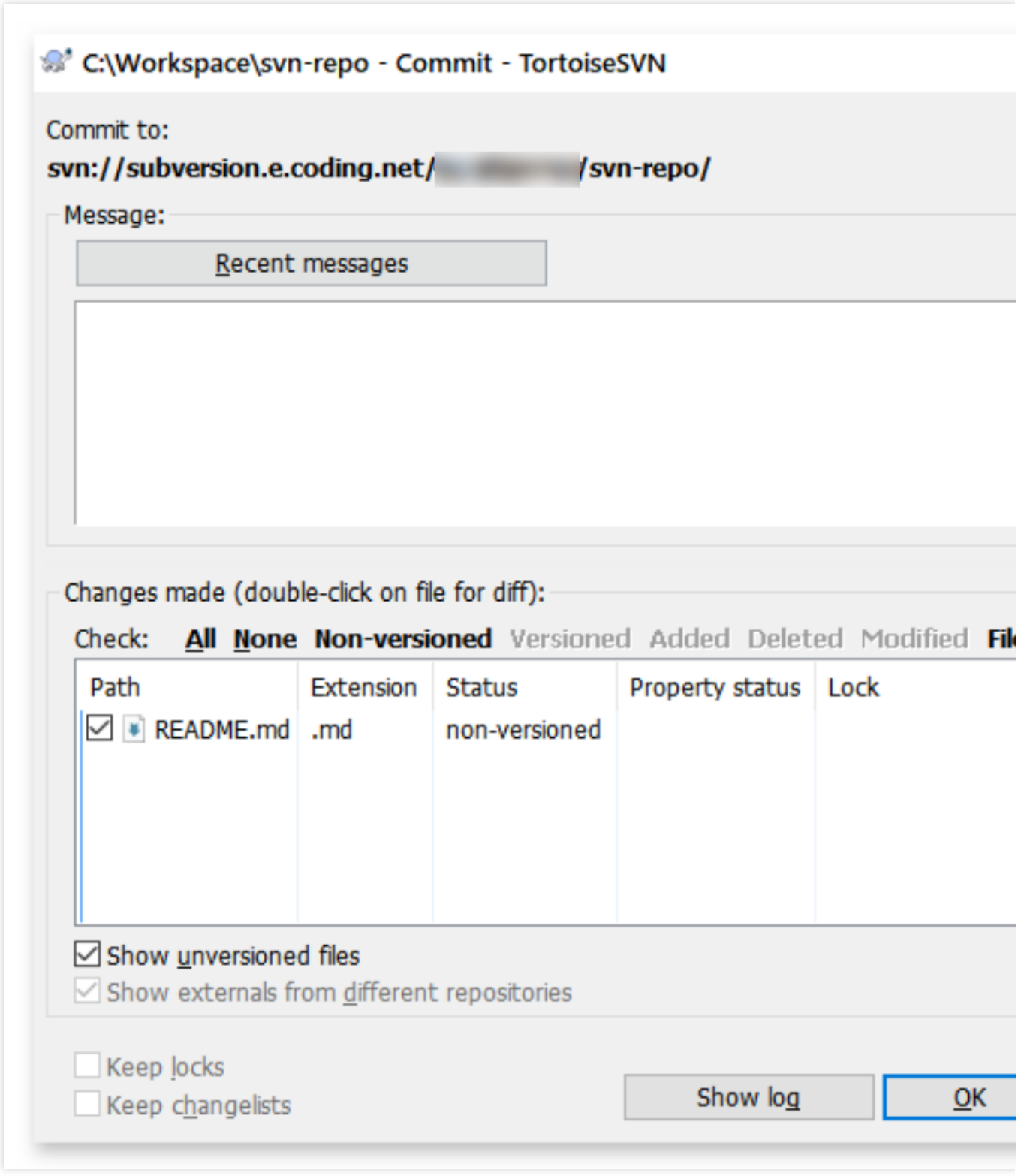
选择 `checkout` 把 SVN 仓库 `checkout` 到本地（请将地址替换为你的 SVN 仓库地址）。



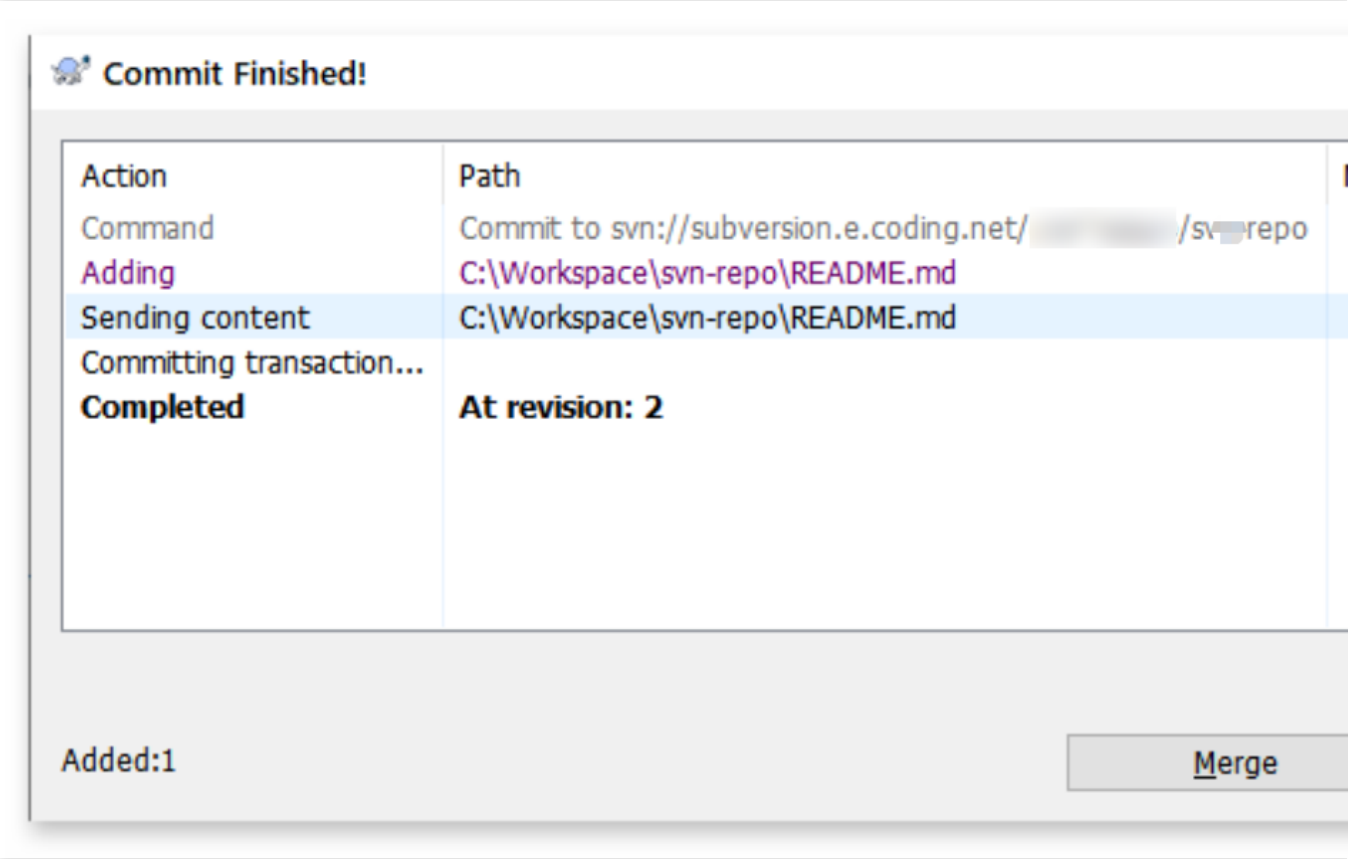
2. 第一次 `checkout` 需要输入用户名和密码。勾选 `Save authentication` 保存认证信息之后就不需要每次都输入密码。



3. 进入检出的文件夹，新建 `README.md` 文件。



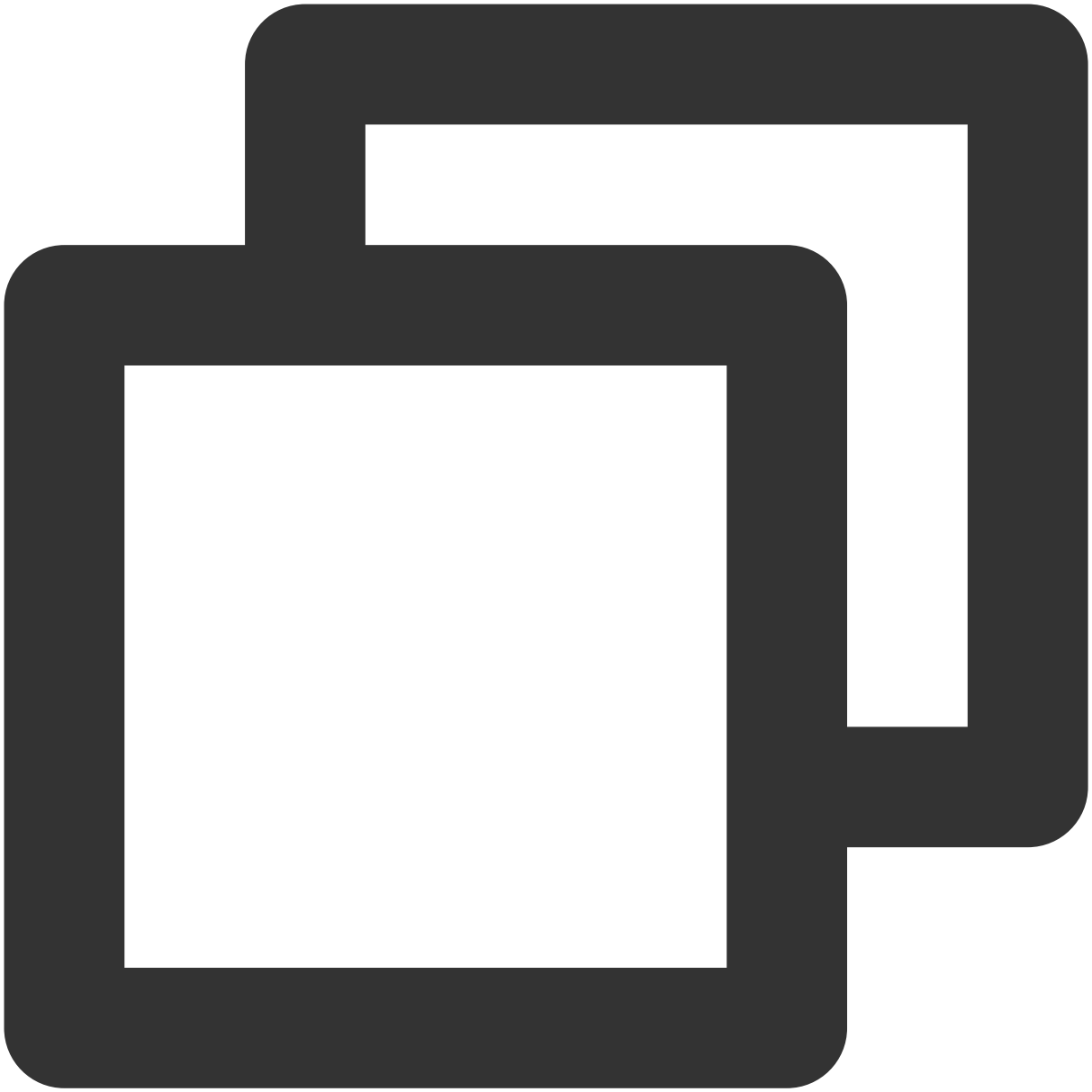
在空白处右键鼠标，选择 `SVN commit...` 将新建的文件保存进版本库：



Linux 环境

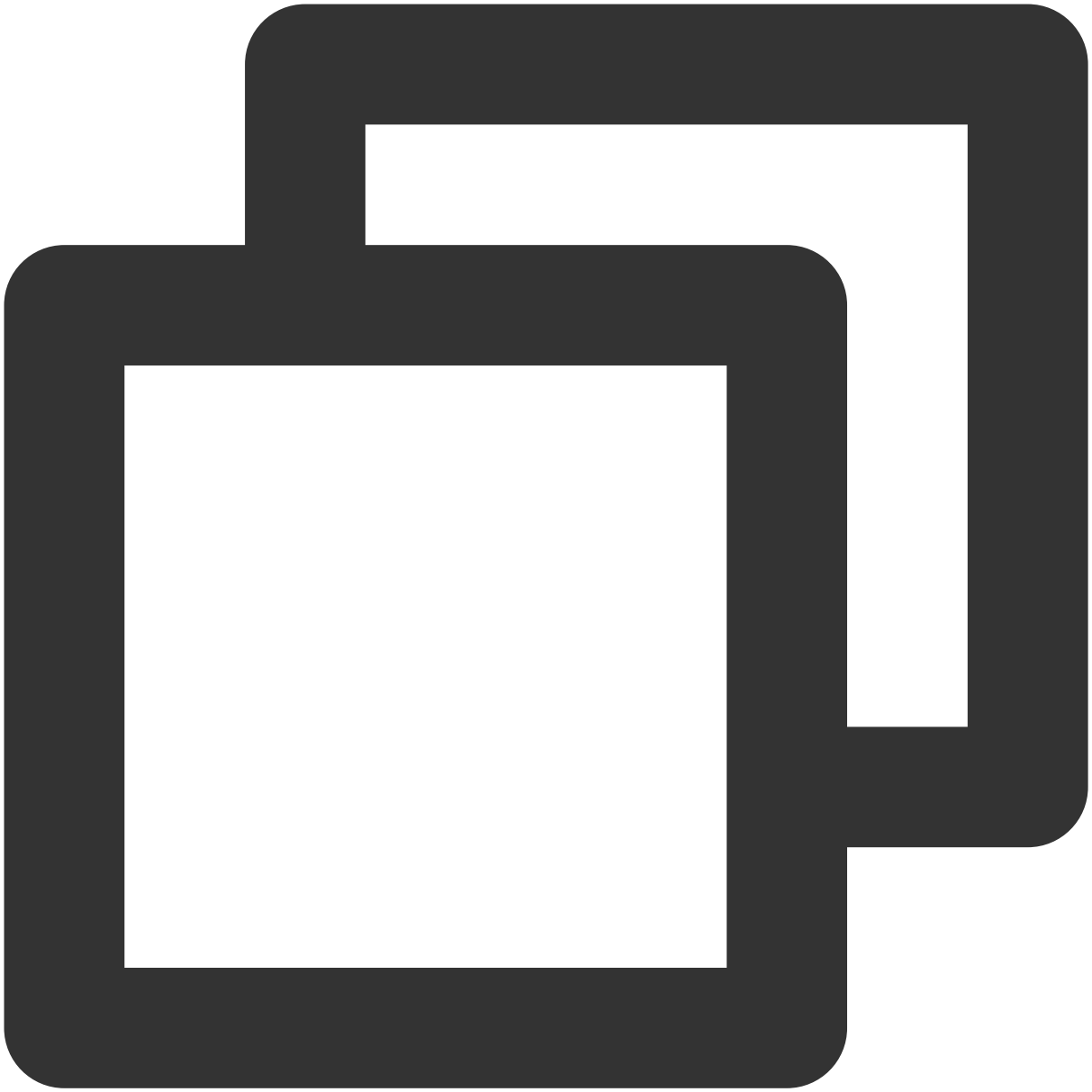
在 Linux 下可以直接用系统的包管理工具安装 SVN。

在 Fedora 上用 yum 安装



```
$ sudo yum install subversion
```

在 Ubuntu 或 Debian 上用 apt-get 安装



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

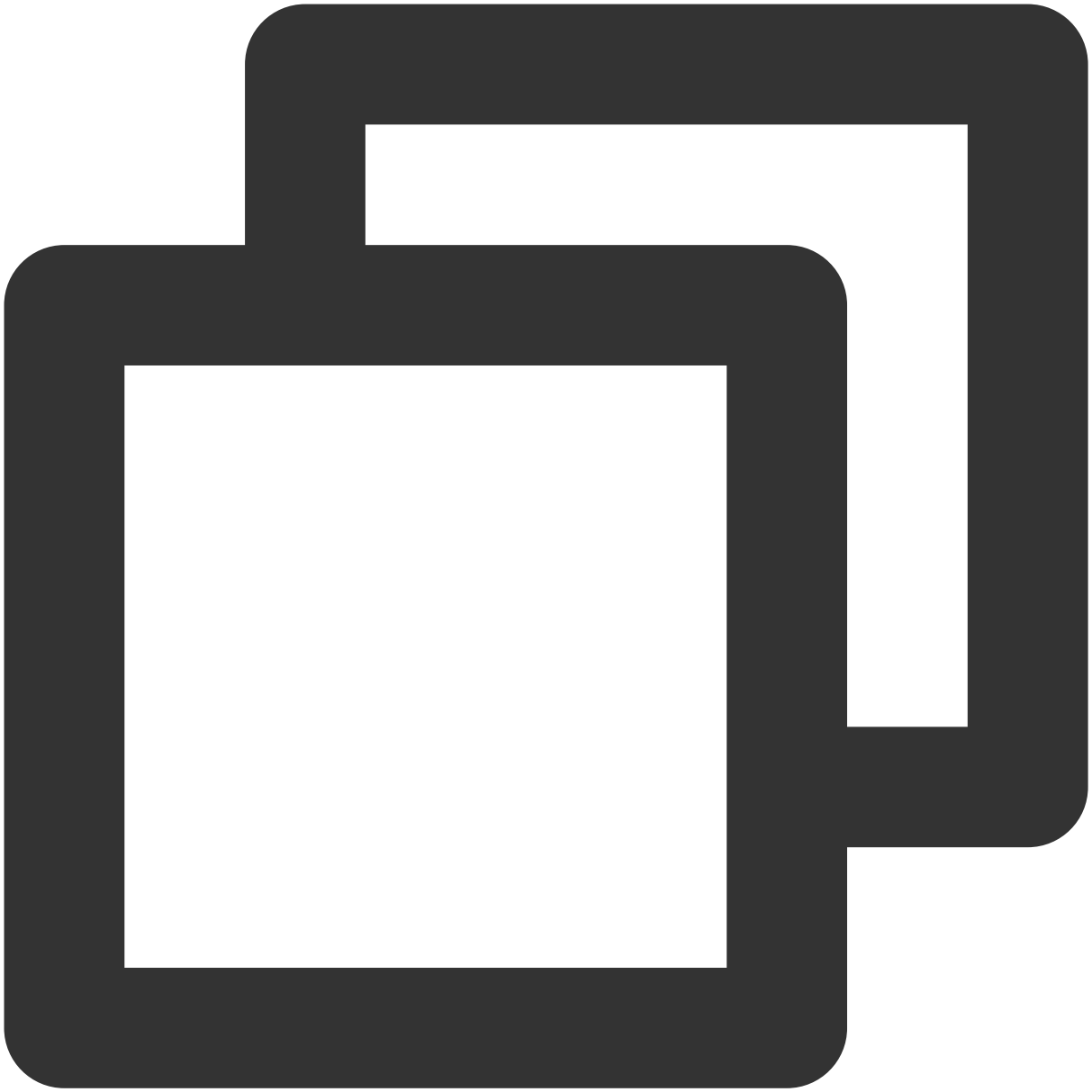
安装成功之后，就可以用 `svn checkout / commit` 来访问 SVN 仓库。

说明：

使用方法与在 Mac 平台使用命令行没有太大区别。

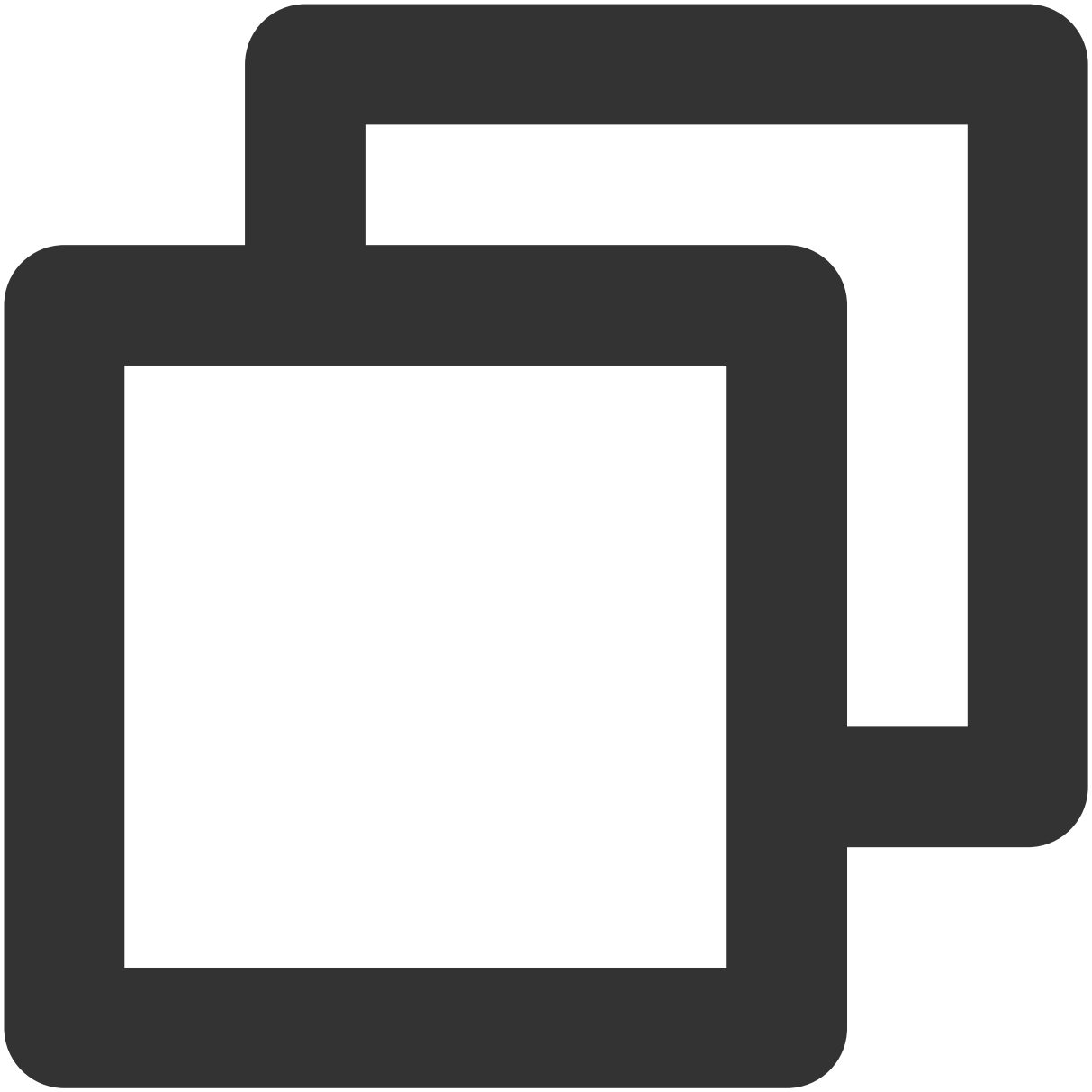
Ubuntu 下使用 SVN 命令行出现协商认证机制错误

在 ubuntu 下使用 SVN 命令行客户端可能出现以下错误：



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

这是由于 SVN 的认证过程使用到了 SASL 库来完成，所以需要运行以下命令安装依赖库来使用 SASL 认证：



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

管理 SVN 目录权限

最近更新时间：2023-12-25 17:08:18

本文为您详细介绍如何管理 SVN 仓库权限。

进入项目

1. 登录 CODING 控制台，单击团队域名进入 CODING 使用页面。
2. 单击页面右上角的

，进入项目列表页面，单击项目图标进入目标项目。
3. 选择左侧菜单**代码仓库**，进入代码仓库首页。
4. 若左侧未显示代码仓库，需项目管理员前往**项目设置** > **项目与成员** > **功能开关**打开功能开关。

SVN 仓库现支持权限控制，管理员能够为单独的用户设置指定目录的权限。管理员可以为仓库及子目录单独设置以下 3 种权限：

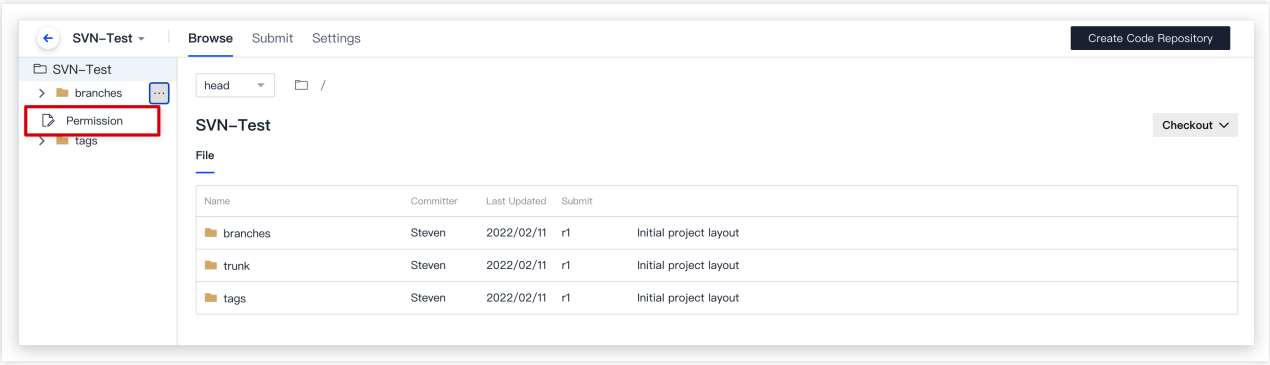
只读：只能查看设置的目录，不能写入，允许检出。

读写：可对设置的目录进行查看和写入，允许检出。

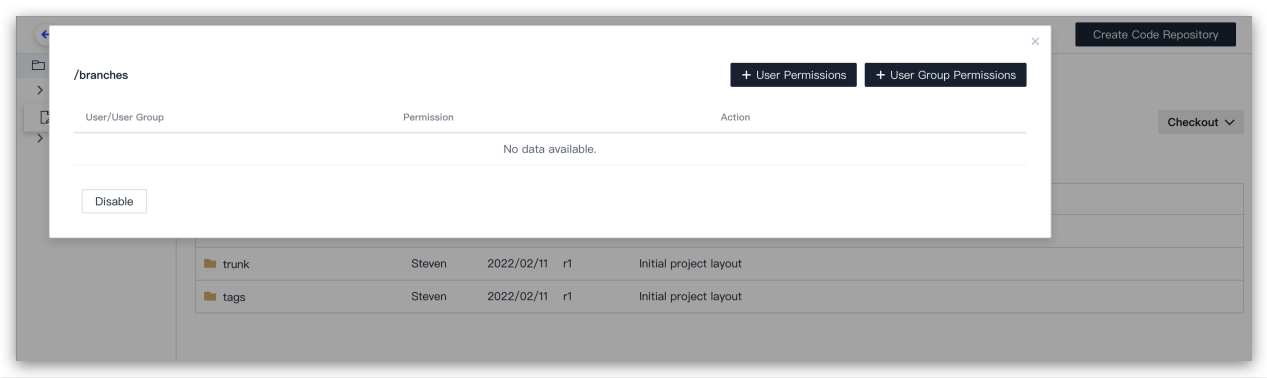
无权限：不能查看也不能写入，禁止检出。

设置权限

1. 因每个用户默认对仓库都有读写权限，若需对 SVN 仓库中的某一目录进行权限控制，单击该目录的更多操作按钮，选择**权限**。



2. 在弹出的权限设置页面，您可以为该目录添加单独的用户与相应权限。

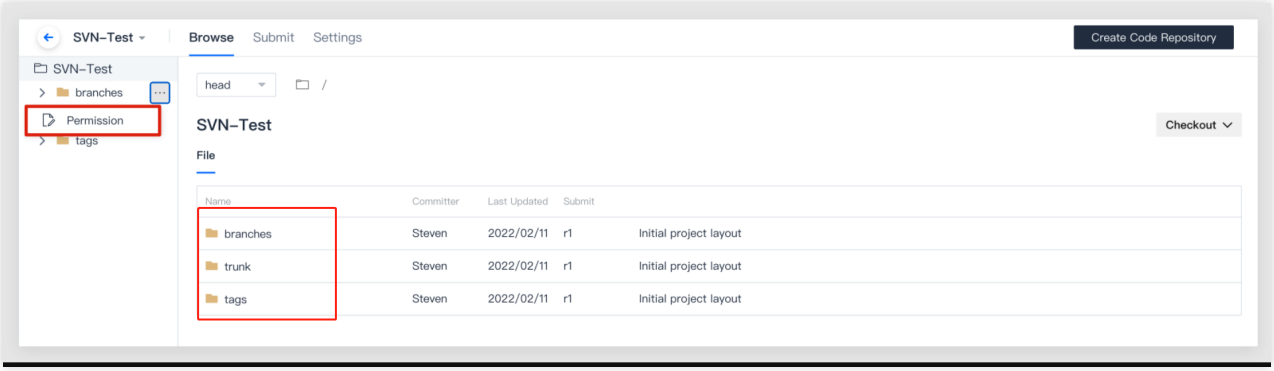


3. 设置完成后，已配置了权限控制的目录将以不同颜色来区分。若没有对目录配置过权限控制，目录默认显示为黑色，用户拥有对其的**读写**权限。

读写：黑色（默认）

只读：黄色

无权限：灰色



权限覆盖说明

在某些场景下，可能会存在父目录与子级目录均设置了权限，且权限不一致的情况（例如，父目录权限为只读而子目录权限为读写）。SVN 仓库中父目录与子目录的权限覆盖规则如下：

若父目录设置了权限，子目录未设置权限，则子目录继承父目录权限。

若父目录与子目录均设置了权限，以子目录的权限为准。例如：

1.1 父目录权限为**读写**，子目录权限为**只读**，则子目录实际为**只读**权限。

1.2 父目录权限为**只读**，子目录权限为**读写**，则子目录实际为**读写**权限。

1.3 父目录权限为**读写**或**只读**，子目录**无权限**，则子目录实际为**无权限**。

SSH 协议使用
配置 SSH 公钥

最近更新时间：2023-12-25 17:08:18

本文为您详细介绍如何配置 SSH 公钥。

进入项目

1. 登录 CODING 控制台，单击团队域名进入 CODING 使用页面。
2. 单击页面右上角的

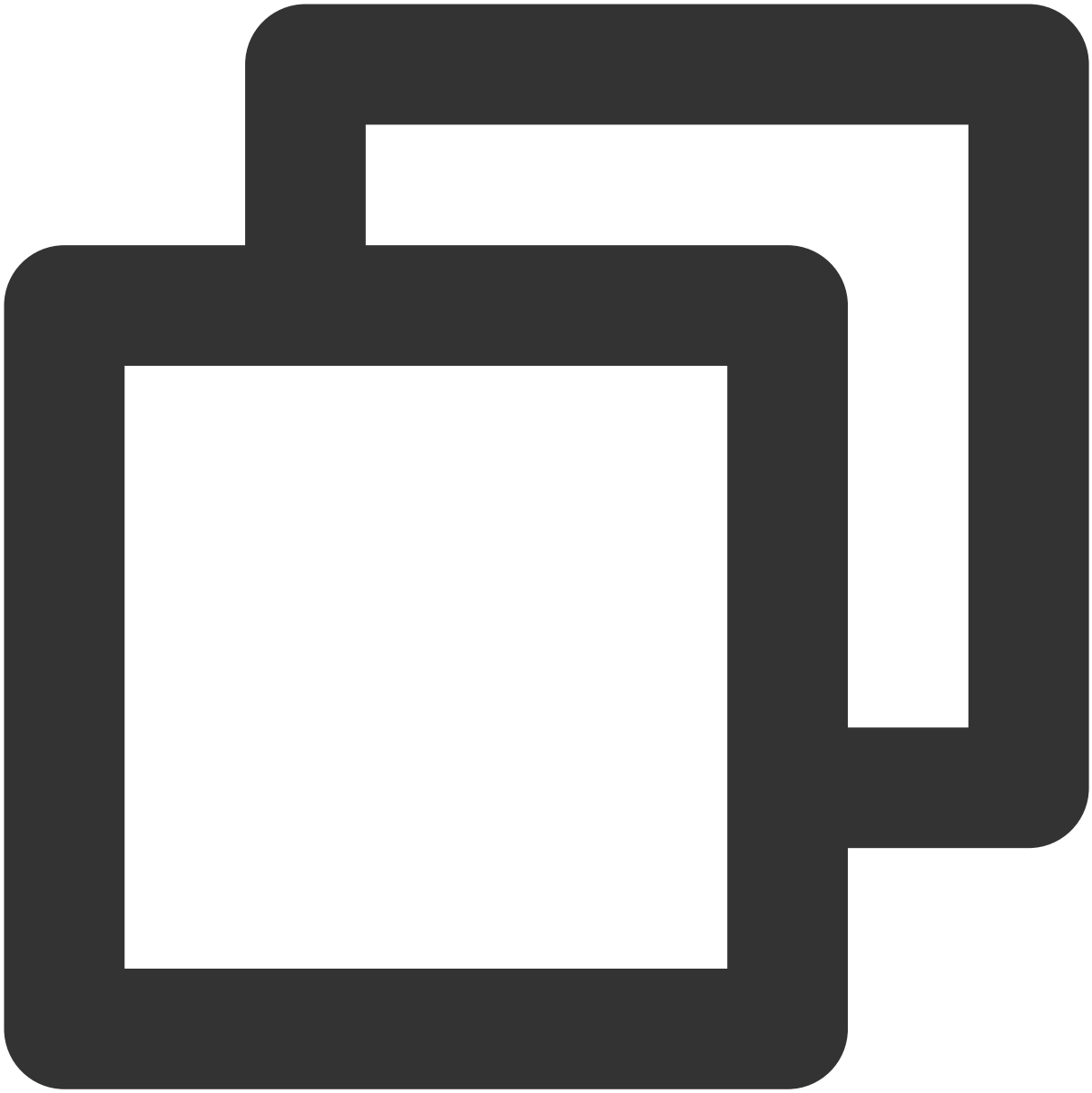
，进入项目列表页面，单击项目图标进入目标项目。
3. 选择左侧菜单**代码仓库**，进入代码仓库首页。
4. 若左侧未显示代码仓库，需项目管理员前往**项目设置** > **项目与成员** > **功能开关**打开功能开关。

功能介绍

SSH 的全称为 Secure Shell 安全外壳协议，是一种加密的网络传输协议。它能够在公开的网络环境中提供安全的数据传输环境，通常用于登录远程主机与推拉代码。
同样一个 SSH 公钥文件，如果添加至某一个代码仓库，则称为**部署公钥**，配置后默认拥有该项目的只读权限；如果添加至个人账户，称为**账户 SSH 公钥**，配置后拥有账户下所有项目的读写权限。同一个 SSH 公钥无法重复添加至代码仓库和个人账户。

生成公钥

本文使用 `ssh-keygen` 工具生成 SSH 公钥，执行命令：

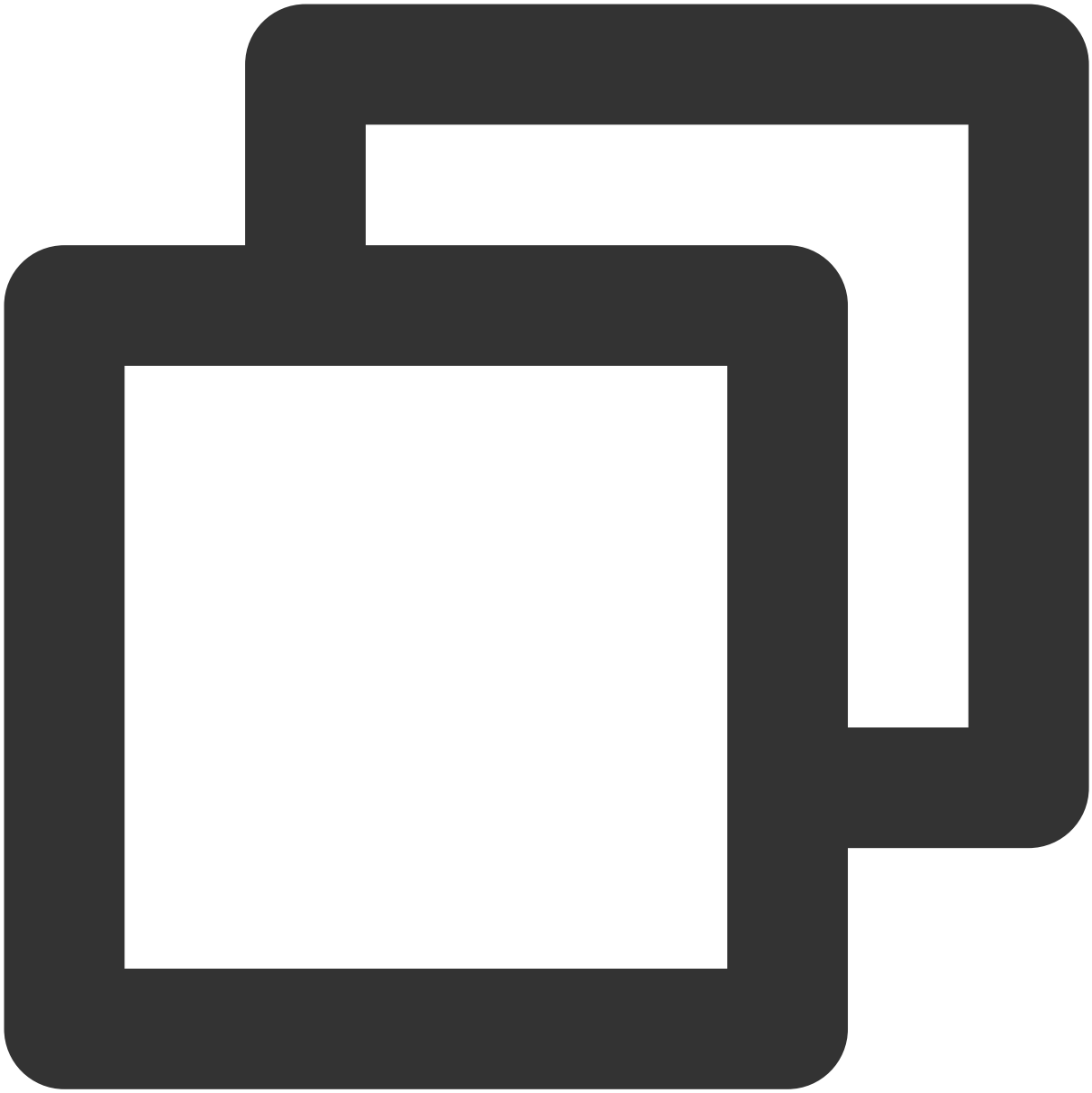


```
ssh-keygen -m PEM -t rsa -b 4096 -C "your.email@example.com"  // 创建新的 SSH 私钥与公
Enter file in which to save the key (/Users/you/.ssh/id_rsa): [Press enter]  // 推荐
Enter passphrase (empty for no passphrase):  // 此处直接回车即可；若设置密码，则每次使用 S
```

说明：

若您需要使用多个 SSH 密钥对，在 `Enter file in which to save the key` 步骤时，输入一个新的文件名称就可以避免覆盖已有的密钥对。有关 SSH 更多信息可参见 维基百科。

成功之后显示如下信息：



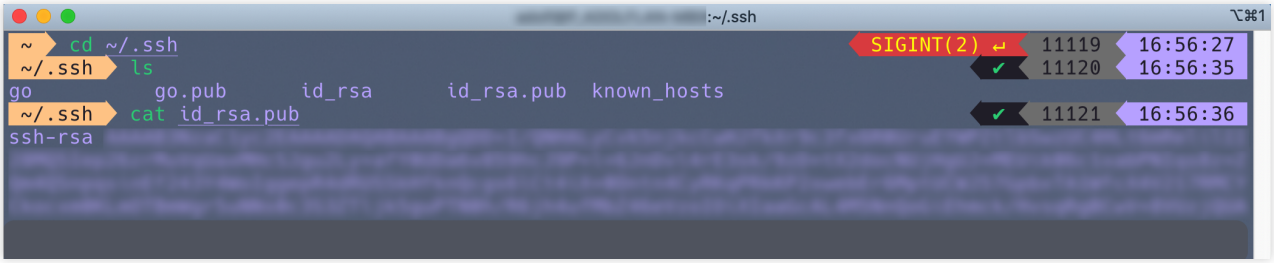
```
Your identification has been saved in /Users/you/.ssh/id_rsa.
# Your public key has been saved in /Users/you/.ssh/id_rsa.pub.
# The key fingerprint is:
# 01:0f:f4:3b:ca:85:d6:17:a1:7d:f0:68:9d:f0:a2:db your.email@example.com
```

添加公钥

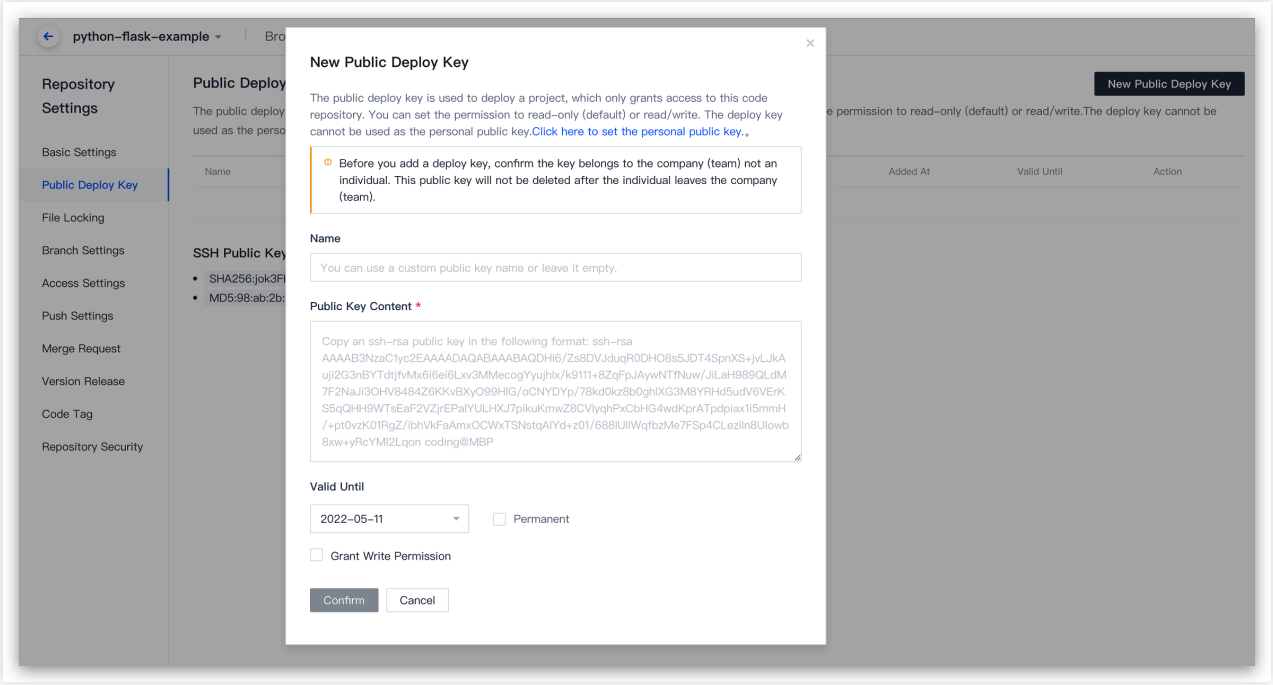
您可以按需添加公钥至单个仓库或个人账户，相同的 SSH 公钥无法重复添加。

关联公钥至单个代码仓库

1. 打开上文中生成的密钥对的地址（通常为 `~/.ssh/` ）找到后缀为 `pub` 的公钥文件，使用 `cat` 命令输出所有内容并复制。



2. 前往代码仓库的**设置** > **部署公钥**页面，单击添加部署公钥，粘贴复制的公钥全文。



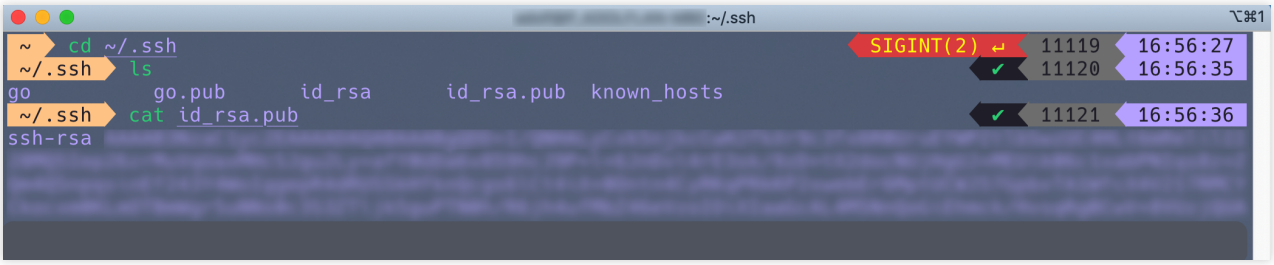
说明：

部署公钥默认拥有该项目的只读权限。如果需要获取推送权限，请勾选部署公钥设置里的**授予推送权限**。

3. 完成后，在本地运行首次连接时的公钥认证命令： `ssh -T git@e.coding.net`

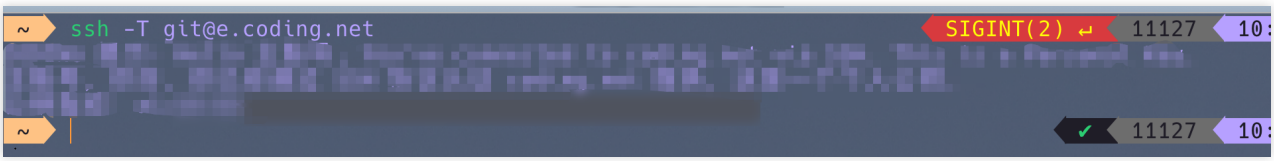
关联公钥至个人账户

1. 打开上文中生成的密钥对的地址（默认地址通常为 `~/.ssh/` ）找到后缀为 `pub` 的公钥文件，使用 `cat` 命令输出所有内容并复制。



2. 登录 CODING，单击右上角个人头像进入**个人账户设置 > SSH 公钥**页面，然后单击**新建公钥**。

3. 根据提示粘贴已复制的公钥内容，按需填写公钥名称。
4. 完成后，在本地运行首次连接时的公钥认证命令： `ssh -T git@e.coding.net` 。



密钥指纹鉴权

最近更新时间：2023-12-25 17:08:18

本文为您详细介绍如何使用密钥指纹进行代码仓库鉴权，用以确认所连接的远程仓库是否为真正的 CODING 仓库。

进入项目

1. 登录 CODING 控制台，单击团队域名进入 CODING 使用页面。
2. 单击页面右上角的

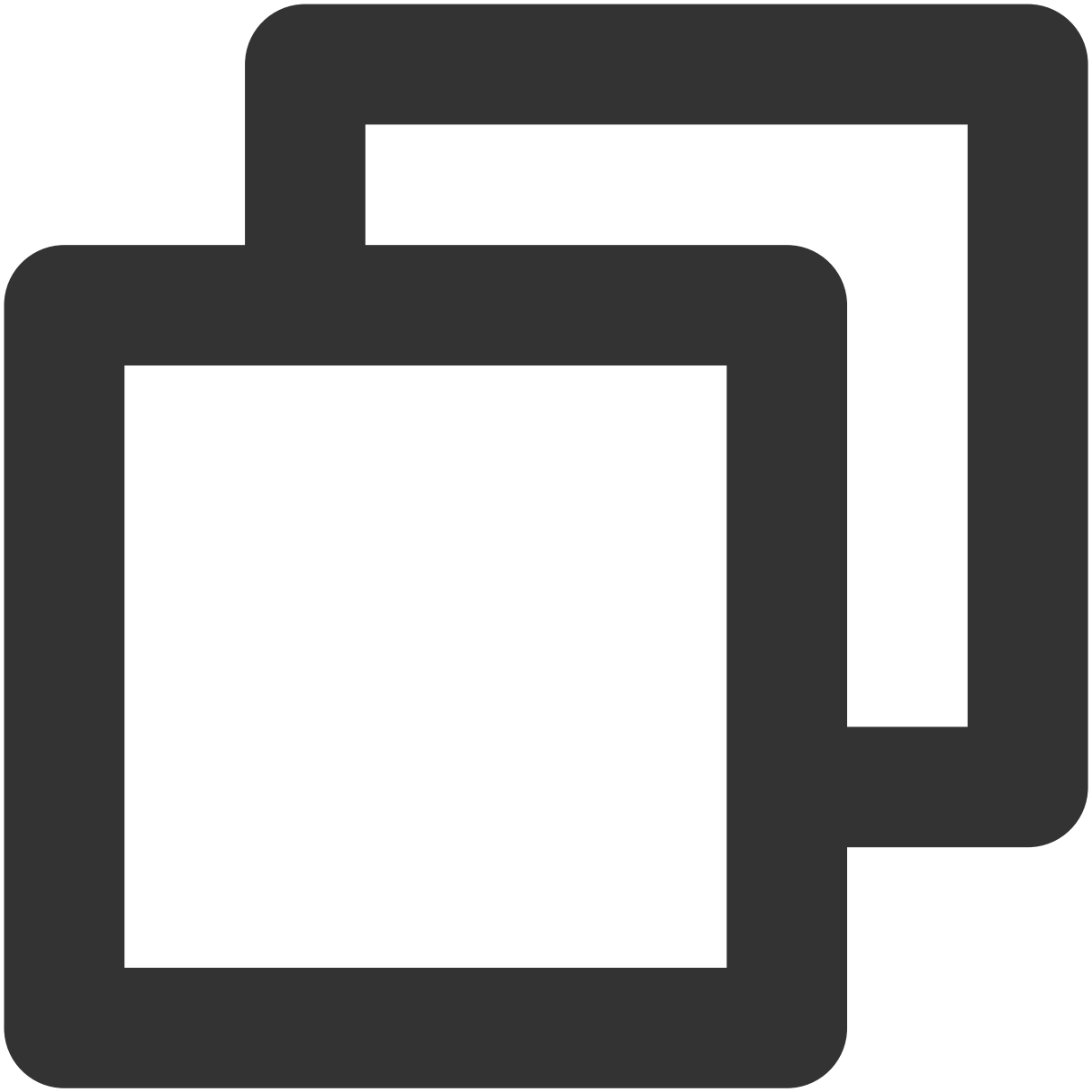
，进入项目列表页面，单击项目图标进入目标项目。
3. 选择左侧菜单**代码仓库**，进入代码仓库首页。
4. 若左侧未显示代码仓库，需项目管理员前往**项目设置** > **项目与成员** > **功能开关**打开功能开关。

功能介绍

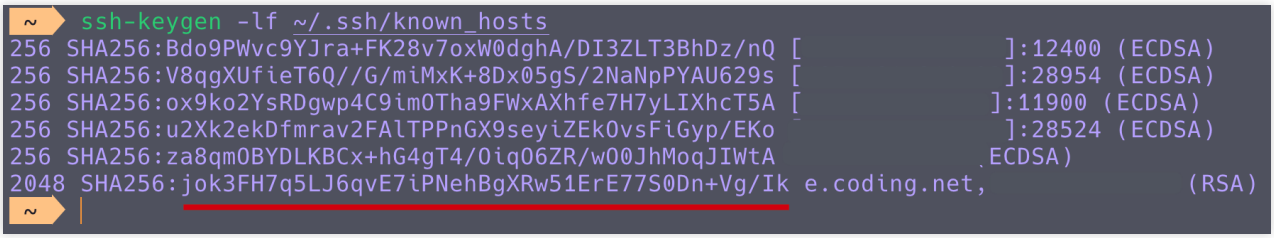
代码安全是永不过时的议题，为了保证您所连接的远端仓库是真正的 CODING 代码仓库，现提供 SSH 密钥指纹用于鉴权。您只需要在本地运行命令后，验证返回的结果就可以知晓远端代码仓库的真实性。

验证 SHA256 算法指纹

查看本地 `.ssh/know_hosts` 文件中关于 `e.coding.net` 的 SHA256 算法的指纹，如果返回值为 `jok3FH7q5LJ6qvE7iPNehBgXRw51ErE77S0Dn+Vg/Ik` ，则证明您连接到了正确的 CODING 服务器。可在终端中运行命令查看结果。

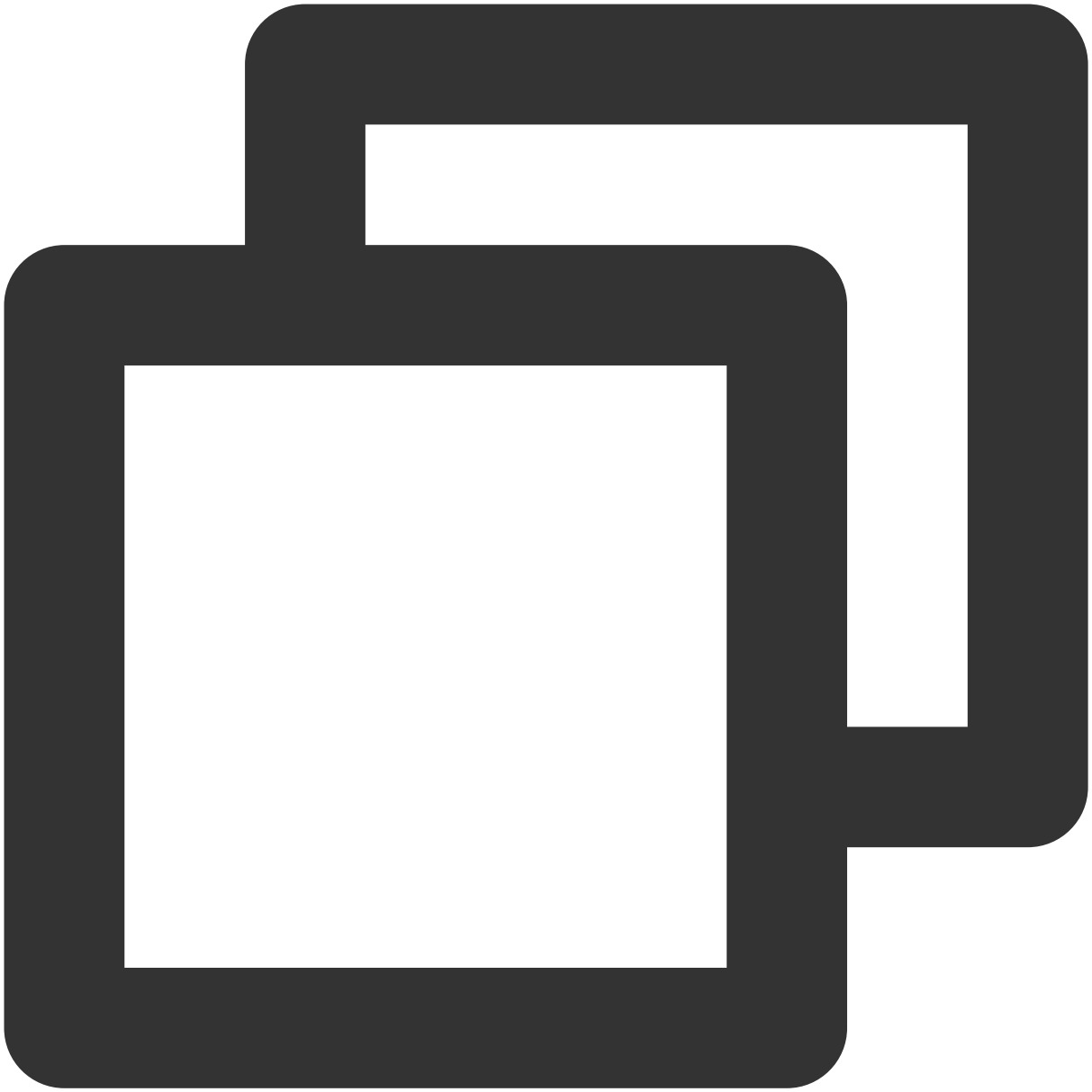


```
ssh-keygen -lf ~/.ssh/known_hosts
```

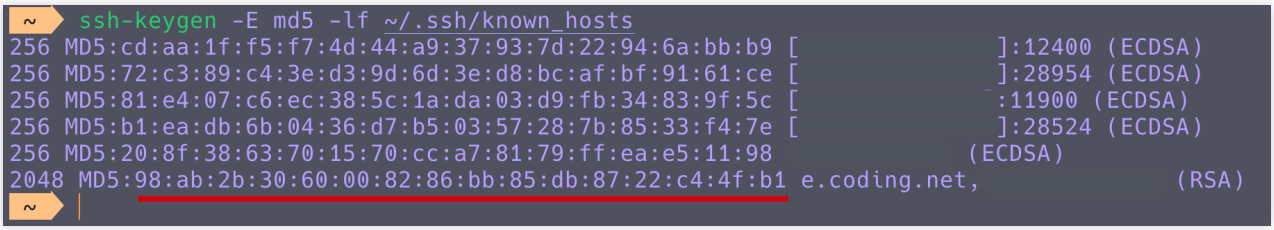


验证 MD5 算法指纹

查看本地 `.ssh/know_hosts` 文件中关于 `e.coding.net` 的 MD5 算法的指纹，如果返回值是 `98:ab:2b:30:60:00:82:86:bb:85:db:87:22:c4:4f:b1`，则证明您连接到了正确的 CODING 服务器。可在终端中运行命令查看结果。



```
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)
~

通过 SSH 协议推拉代码

最近更新时间：2023-12-25 17:08:18

本文为您详细介绍如何通过 SSH 协议进行代码推拉。

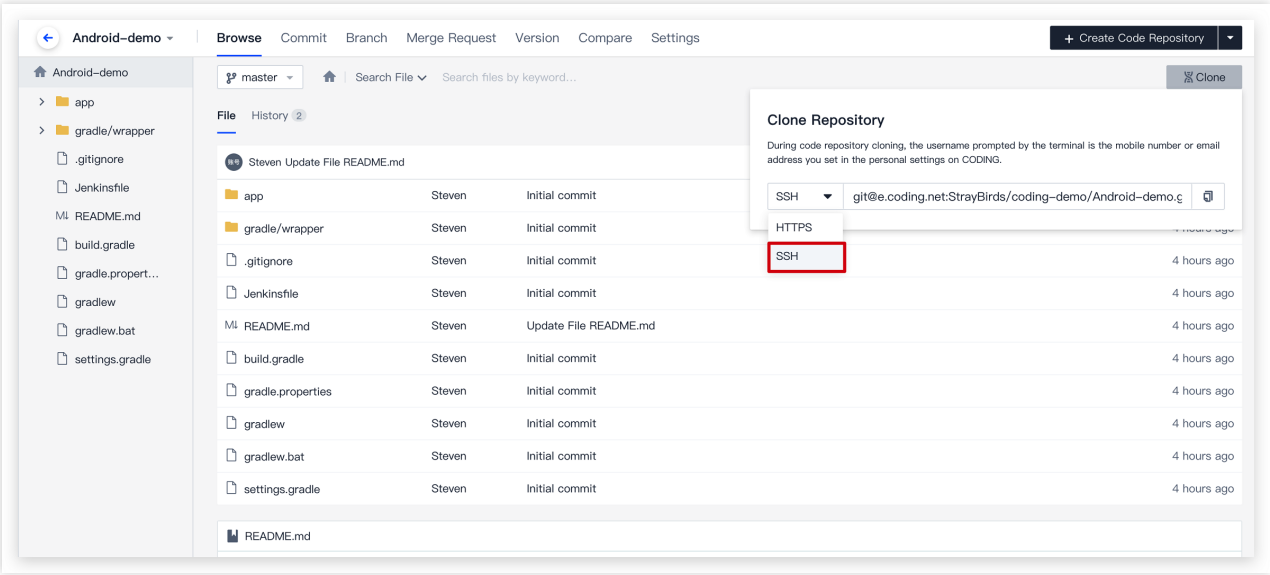
进入项目

1. 登录 CODING 控制台，单击团队域名进入 CODING 使用页面。
2. 单击页面右上角的

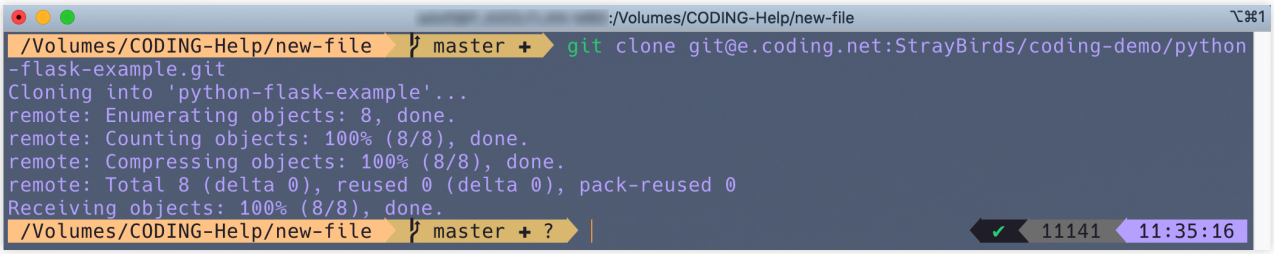
，进入项目列表页面，单击项目图标进入目标项目。
3. 选择左侧菜单**代码仓库**，进入代码仓库首页。
4. 若左侧未显示代码仓库，需项目管理员前往**项目设置** > **项目与成员** > **功能开关**打开功能开关。

CODING 支持使用 SSH 协议推拉代码。

1. 参见 配置公钥 生成公钥并添加至代码仓库或个人账户。
2. 在代码仓库的浏览页面复制其 SSH 地址。



3. 在本地运行 `git clone + 仓库地址` 命令行即可完成拉取。


:/Volumes/CODING-Help/new-file
⌥⌘1
/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

分支管理
创建分支

最近更新时间：2023-12-25 17:08:18

本文为您详细介绍如何使用在代码仓库中创建分支。

进入项目

1. 登录 CODING 控制台，单击团队域名进入 CODING 使用页面。
2. 单击页面右上角的

，进入项目列表页面，单击项目图标进入目标项目。
3. 选择左侧菜单**代码仓库** > **分支**，进入分支管理页面。

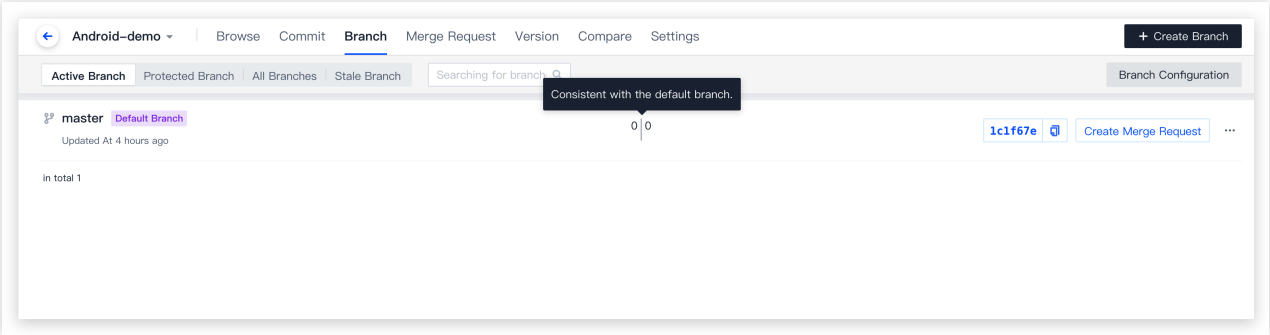
分支是 Git 中的常用功能。常用的开发模式是保留一个主干，各项开发或修补工作在其他分支进行，完成后再合并入主干。因为直接在主干开发是一件危险系数较高的活动，分支功能可以视为一道安全的阀门，将各项开发工作分隔开来，它能够保证主要版本的稳定性不被破坏，不同的人可以专注于不同的开发任务。

说明：

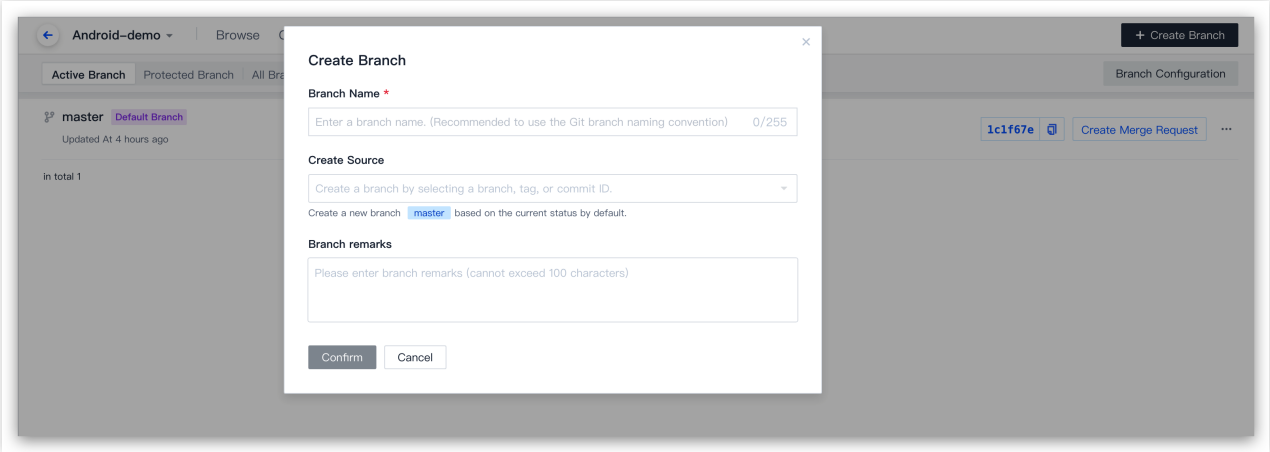
下文中简称 CODING 代码仓库中的分支为远程分支、本地 Git 代码仓库的分支为本地分支。在本地终端运行相关命令也可以快速创建分支，请参见 Git 常用命令。

1. 进入代码仓库的详情页面之后，单击**分支**页签即可查看目前远端仓库中的所有分支。

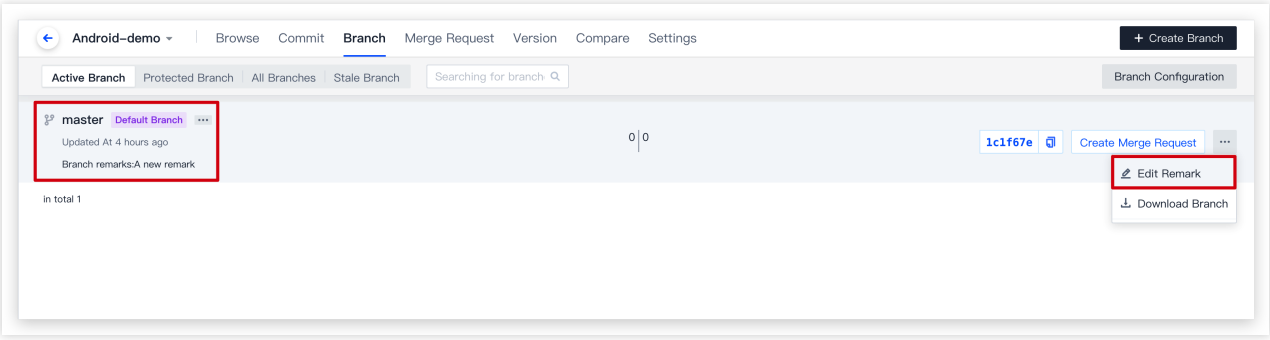
您可以在此处进行新建分支、启用保护分支等操作，分支列表页会显示当前相比于默认分支提前或落后多少提交。



2. 单击右上角的**新建分支**，在弹窗中按照提示输入相关配置信息，默认以 master 分支作为创建来源，相当于基于源分支衍生出新分支。



3. 新建分支时可以添加一个简单的描述信息用以记录分支用途，当分支名称不能完整描述分支用途时，就可以使用分支备注来完善信息。



设置默认分支

最近更新时间：2023-12-25 17:08:18

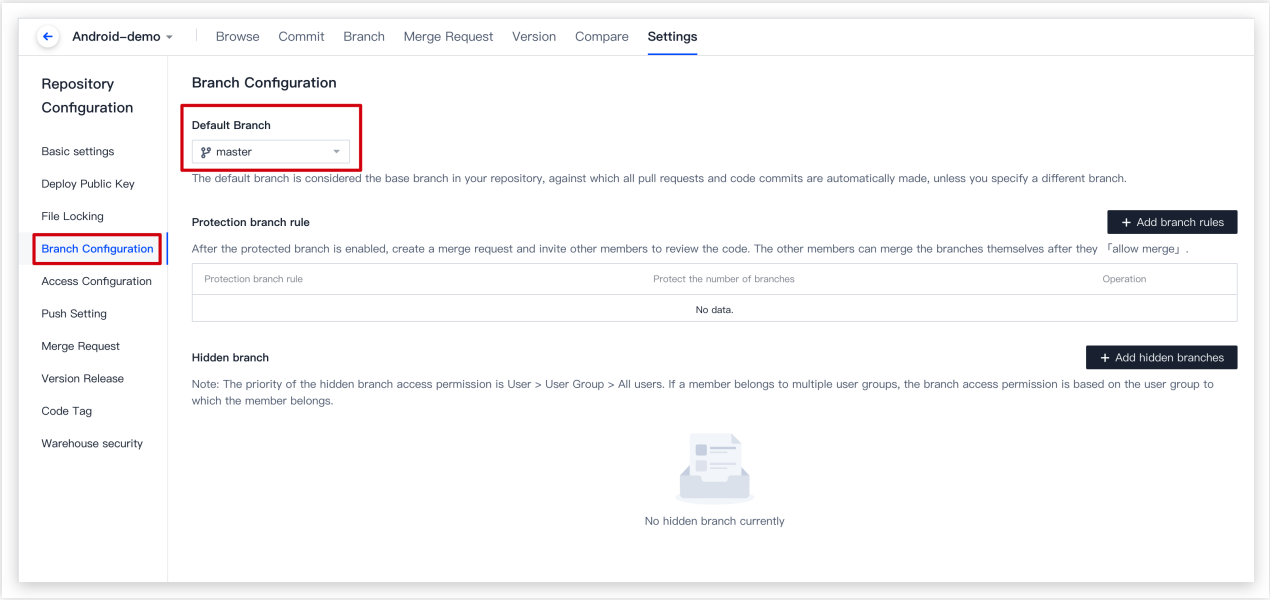
本文为您详细介绍如何设置默认分支。

进入项目

1. 登录 CODING 控制台，单击团队域名进入 CODING 使用页面。
2. 单击页面右上角的

，进入项目列表页面，单击项目图标进入目标项目。
3. 选择左侧菜单**代码仓库 > 分支**，进入分支管理页面。

默认分支被视为仓库中的基本分支，除非您指定其他分支，否则将基于默认分支自动生成所有拉取请求和代码提交。默认分支可在代码仓库的**设置 > 分支设置**页面中修改。仅项目管理员有权限作出变更修改。



设置保护分支

最近更新时间：2023-12-25 17:08:18

本文为您详细介绍如何设置保护分支。

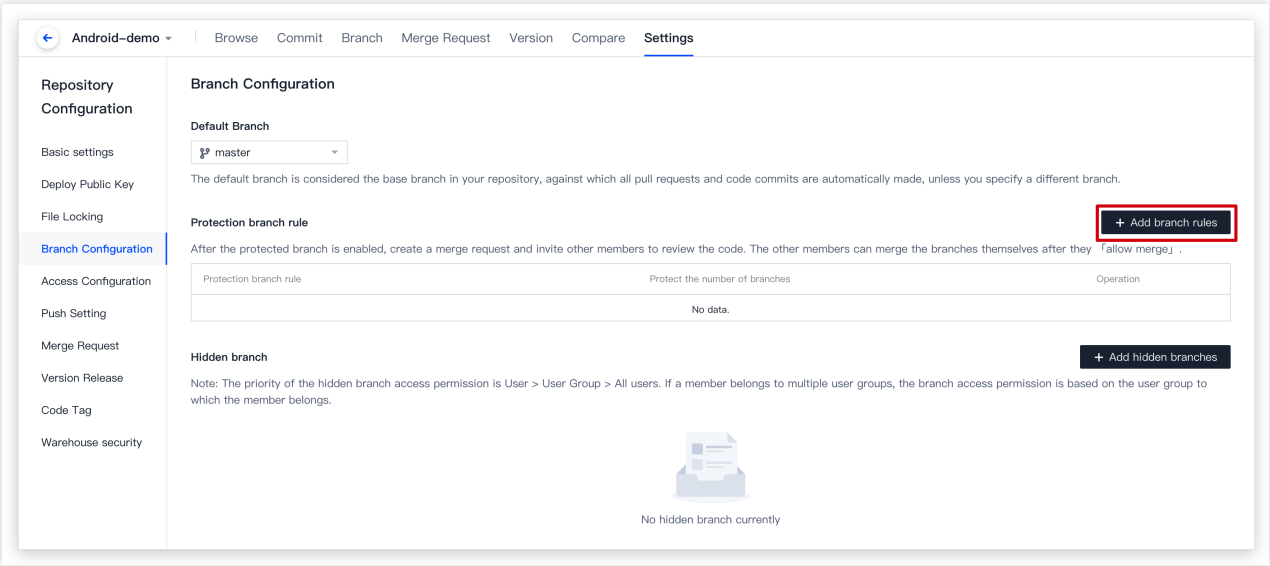
进入项目

1. 登录 CODING 控制台，单击团队域名进入 CODING 使用页面。
2. 单击页面右上角的

，进入项目列表页面，单击项目图标进入目标项目。
3. 选择左侧菜单**代码仓库 > 分支**，进入分支管理页面。

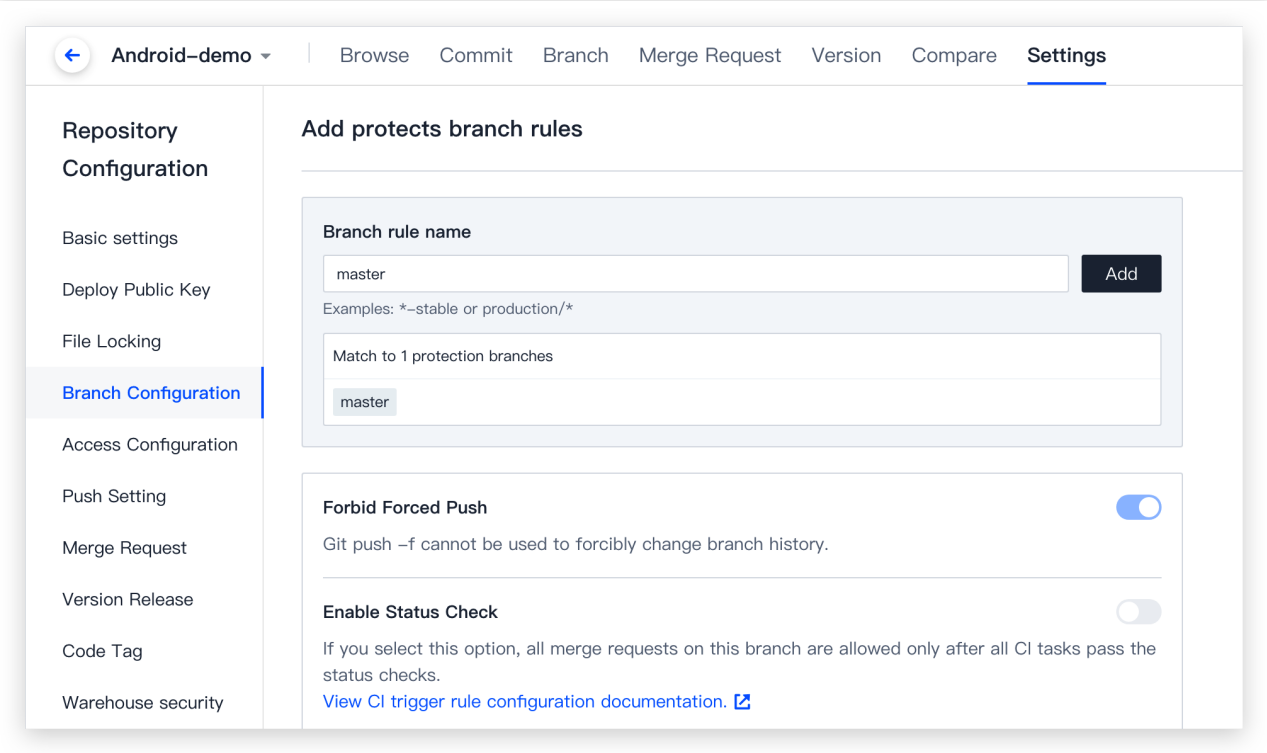
保护分支是 CODING 针对 Git 中有关代码权限开发的一个特色功能，可以将选中的分支保护起来，防止未经报备、允许的更改。

开启后，保护分支在分支列表中将以绿色盾牌为标志。成员修改保护分支时需新建一个分支并在其中进行修改，创建合并请求后邀请其他成员评审代码，评审完成并允许合并后才能执行合并操作。



设置保护分支规则

在代码仓库的**设置 > 分支设置**页面，您可以使用**通配符**更加智能地设置保护分支，符合命名规则的分支都会被视为保护分支。



禁止强制推送：默认打开。即使有 git push 的权限，也不允许通过 git push -f 的方式强制修改分支的提交历史。对于多人合作的分支，强烈建议打开此选项。它确保了只能通过增加新的提交来改变分支内容，而不是修改历史提交的方式来提交变更。

开启状态检查：通过在 CI 中设置规范性检查条件或设置代码扫描方案，运行 CI 成功后才被允许合并，详情请参见持续集成 > 触发规则。

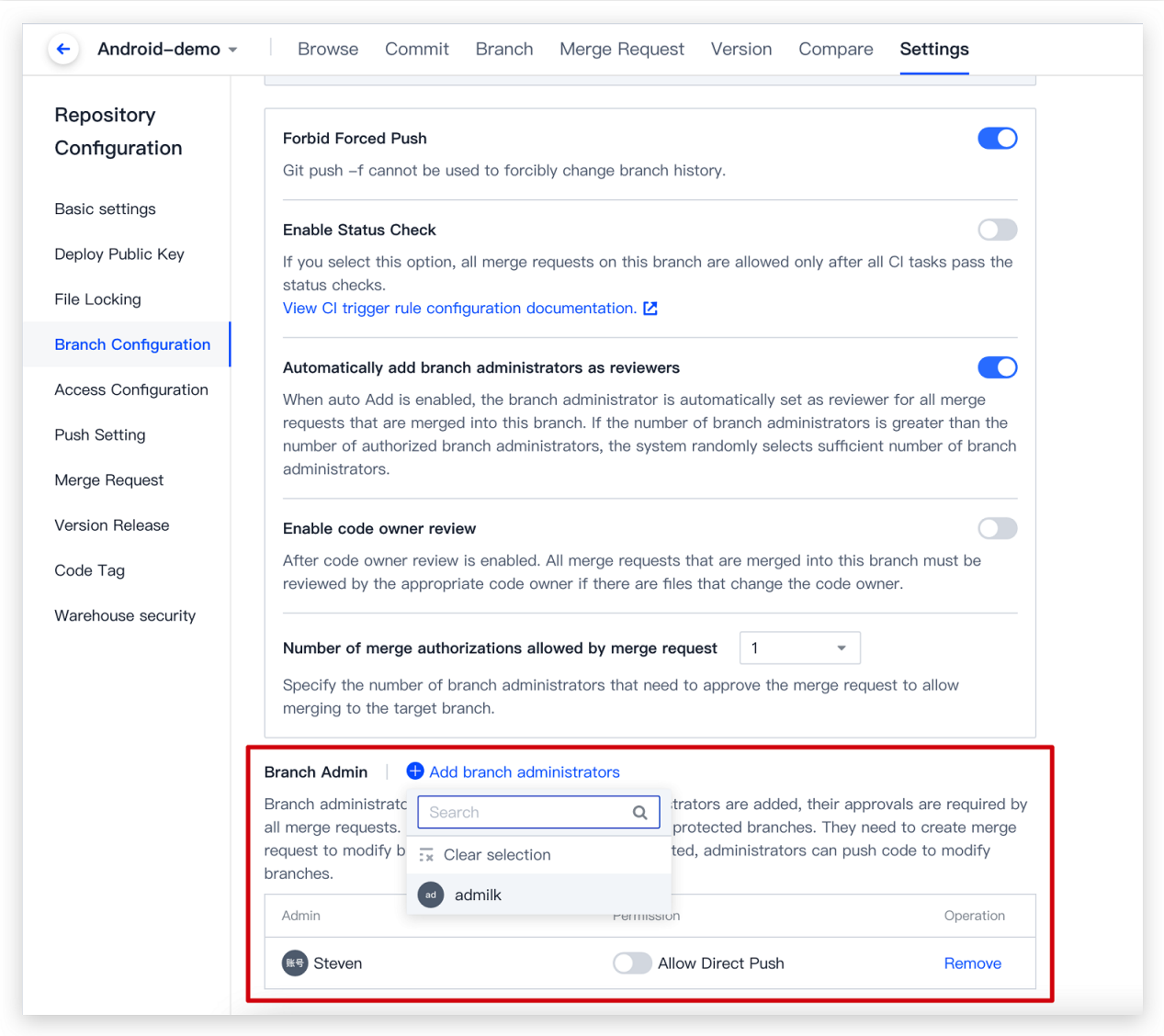
自动添加分支管理员为评审者：该功能开启之后，针对所有合并到此保护分支的合并请求，都会自动将分支管理员设置为评审者。当分支管理员的数量大于设置的**合并请求允许合并授权数量**时，会随机选择对应授权数量的分支管理员作为评审者。例如，当配置了 3 位分支管理员而授权数量为 2 时，系统会从 3 位分支管理员中随机选取 2 位作为评审者。

开启代码所有者评审：该功能开启之后，针对合并到该保护分支的合并请求，如果存在对代码所有者的文件的修改，则必须经过代码所有者的评审之后才允许合并。详情请参见 代码所有者。

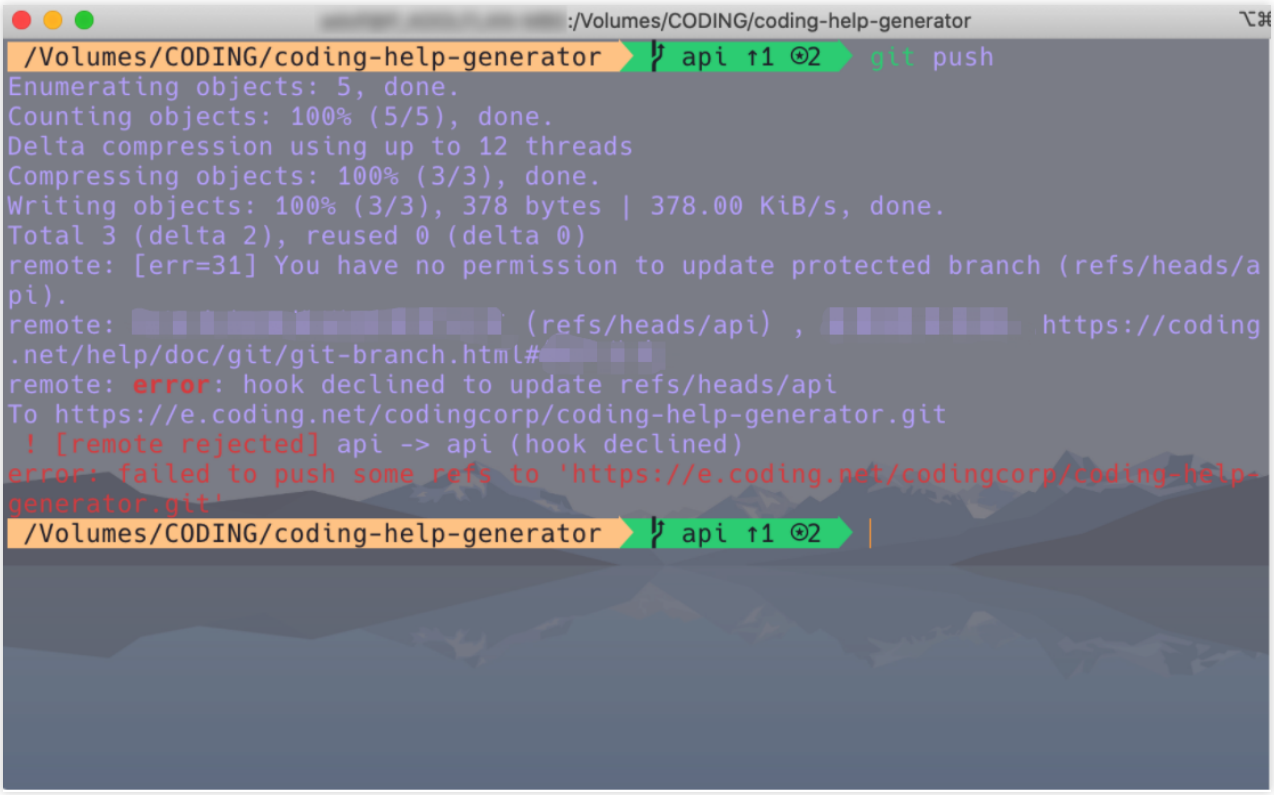
合并请求允许合并授权数量：用于设置合并请求必须经过多少位分支管理员的授权之后才允许合并到目标分支。如果该保护分支没有设置分支管理员，需经过 1 位普通成员授权之后才允许合并。

指定分支管理员

分支管理员为可选项。添加管理员后，所有的合并请求需得到管理员的允许才能被允许合并。管理员默认受到保护分支的条件限制，需创建合并请求才可修改分支。勾选**允许直接推送**，管理员将不受保护限制，可以直接修改保护分支内容。



若成员没有权限（即保护分支的非分支设置员） push 至该分支，当其尝试 push 至该分支的时候，会得到如下错误提示：


:/Volumes/CODING/coding-help-generator
/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/api).
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

设置隐藏分支

最近更新时间：2023-12-25 17:08:18

本文为您详细介绍如何设置隐藏分支。

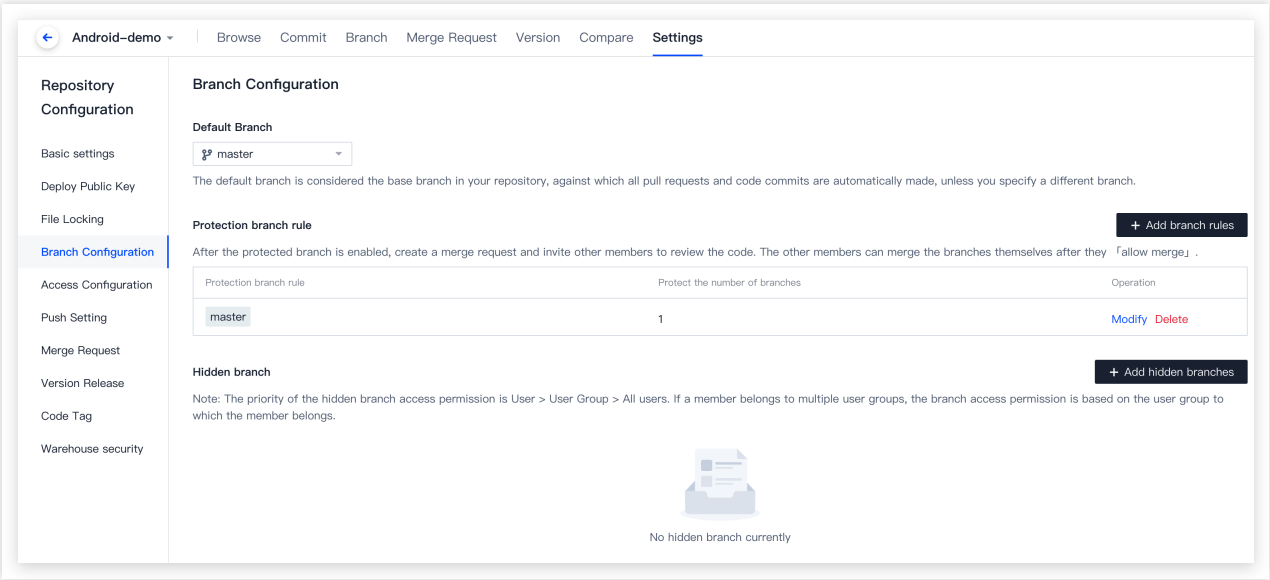
进入项目

1. 登录 CODING 控制台，单击团队域名进入 CODING 使用页面。
2. 单击页面右上角的

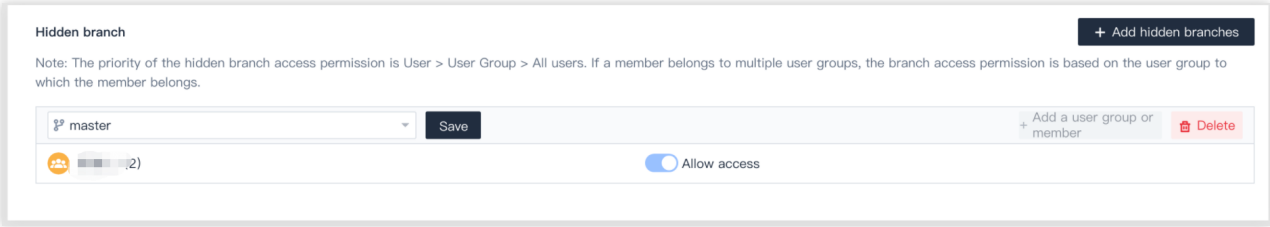
，进入项目列表页面，单击项目图标进入目标项目。
3. 选择左侧菜单**代码仓库 > 分支**，进入分支管理页面。

如需控制单个分支的访问权限，您可以将默认分支以外的任一分支设为隐藏分支，只有经授权的用户或用户组才能访问该分支，保证代码的安全性。

1. 在代码仓库的详情页面，单击**设置 > 分支设置**进入分支设置页面。



2. 单击**添加隐藏分支**，选择或输入分支后单击**保存**。添加成功的隐藏分支会显示在隐藏分支列表。
3. 通过**新增用户组**或**新增成员**添加允许/拒绝分支访问的用户。



说明：

隐藏分支访问权限优先级为 用户 > 用户组 > 所有用户。当某个成员归属多个用户组时，若某个用户组具备访问权限，那么其能够访问隐藏分支。

例如，用户 A 同时归属于用户组 1 (允许访问 `dev/001` 分支) 和用户组 2 (拒绝访问 `dev/001` 分支)，在该种情况下用户 A 可以访问 `dev/001` 分支。

使用代码所有者机制

最近更新时间：2023-12-25 17:08:18

本文为您详细介绍如何使用代码所有者机制。

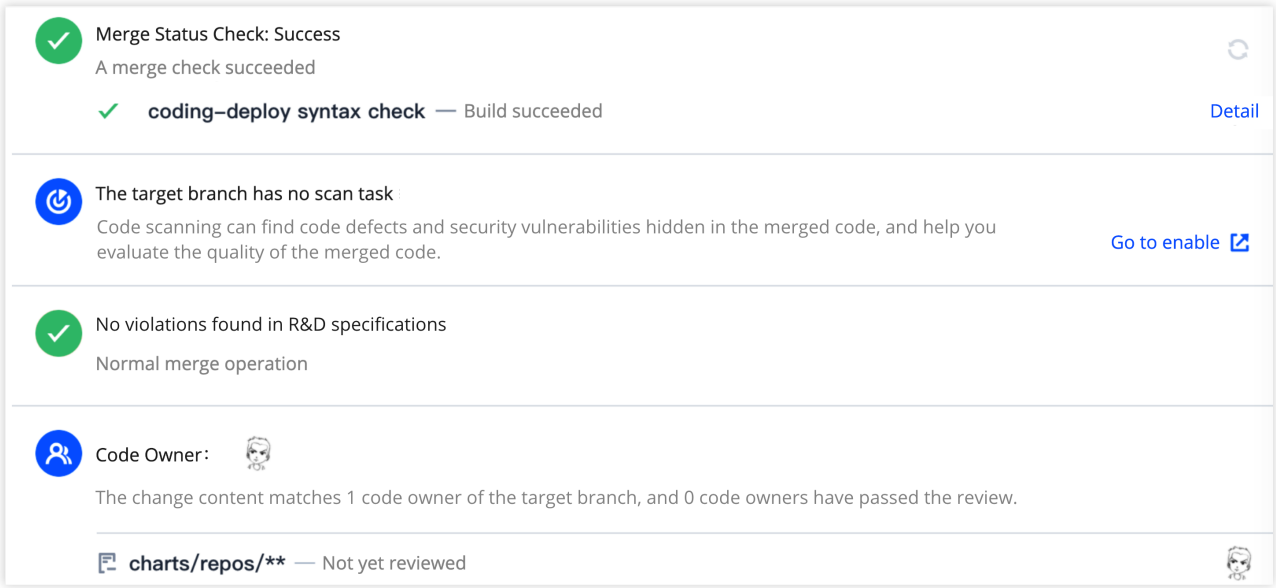
进入项目

1. 登录 CODING 控制台，单击团队域名进入 CODING 使用页面。
2. 单击页面右上角的

，进入项目列表页面，单击项目图标进入目标项目。
3. 选择左侧菜单**代码仓库 > 分支**，进入分支管理页面。

代码所有者机制需配合 保护分支 功能一起使用。在代码仓库中放置声明文件 `CODEOWNERS` 后，就可以声明此仓库内代码文件的所有者，通常为项目负责人。

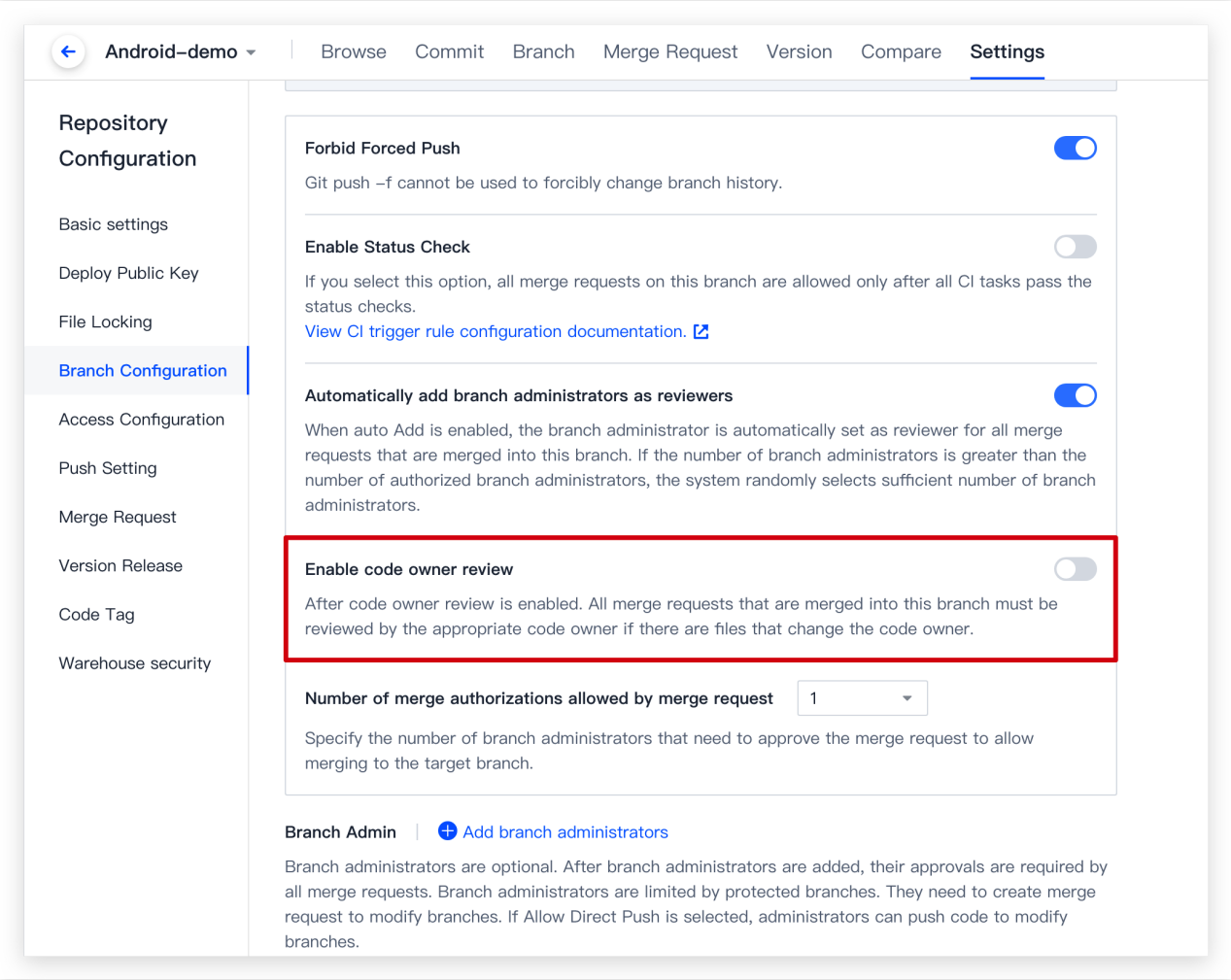
当合并请求的目标分支为保护分支，且在请求中的改动文件涉及到声明文件中设定的路径或文件，合并请求详情中将列出对应的所有者与他们的评审状态。



如上图所示：`CODEOWNER` 文件内声明了 `charts/repos/**` 路径内文件的所有者是小白。若针对保护分支提交了合并请求且涉及到 `chars/repos/` 路径下的文件变动将自动添加小白为此请求的评审者。

说明：

亦支持使用持续集成插件自动添加评审者，详情请参见 合并请求自动添加评审者。
在**保护分支**设置中勾选开启代码所有者评审后，管辖范围的代码变动需通过代码所有者的评审后方能合并。



声明文件地址

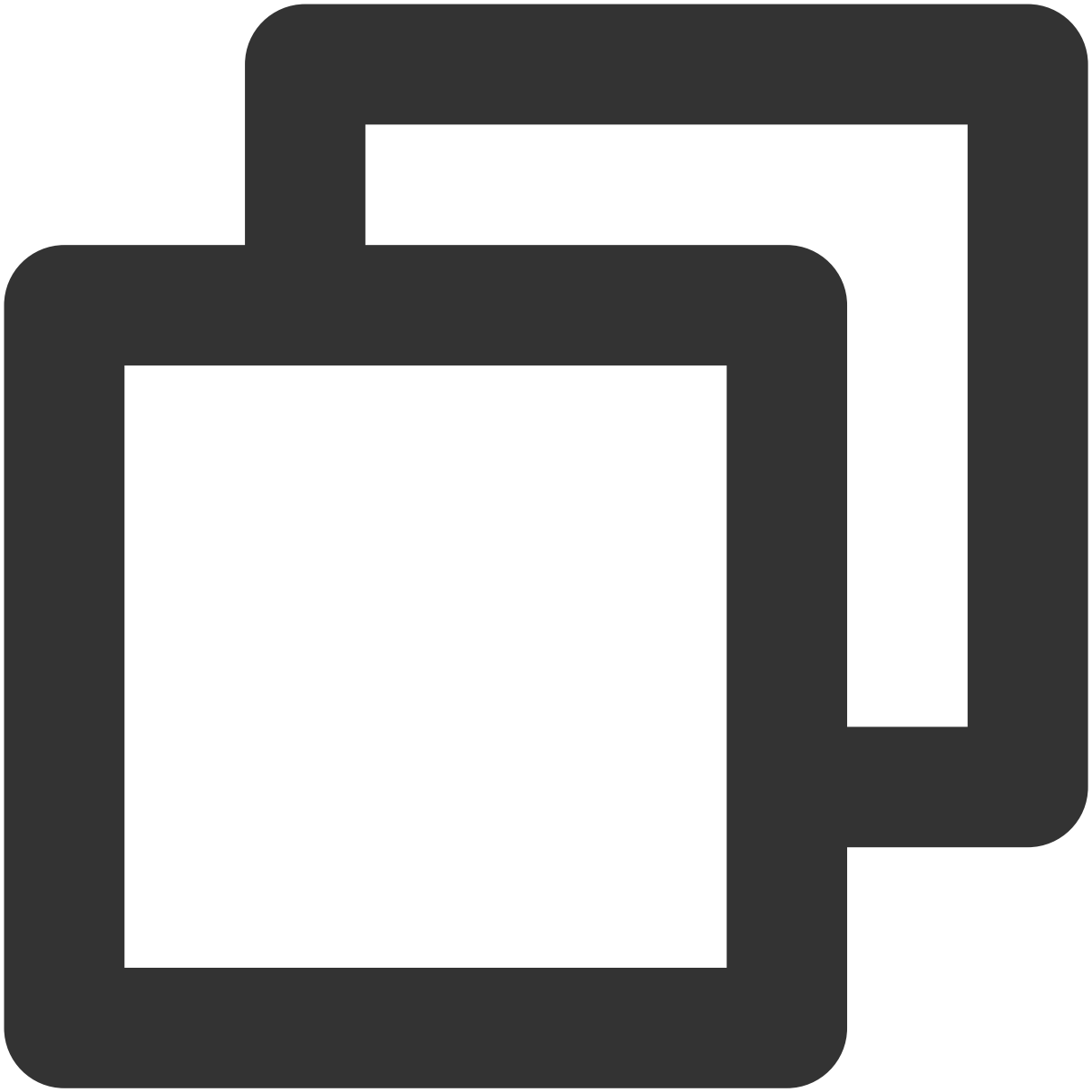
代码所有者的声明文件 `CODEOWNERS` 默认从以下位置逐层检索，文件名必须是大写格式，找到一个后就会停止搜寻。

根目录

`docs/` 目录

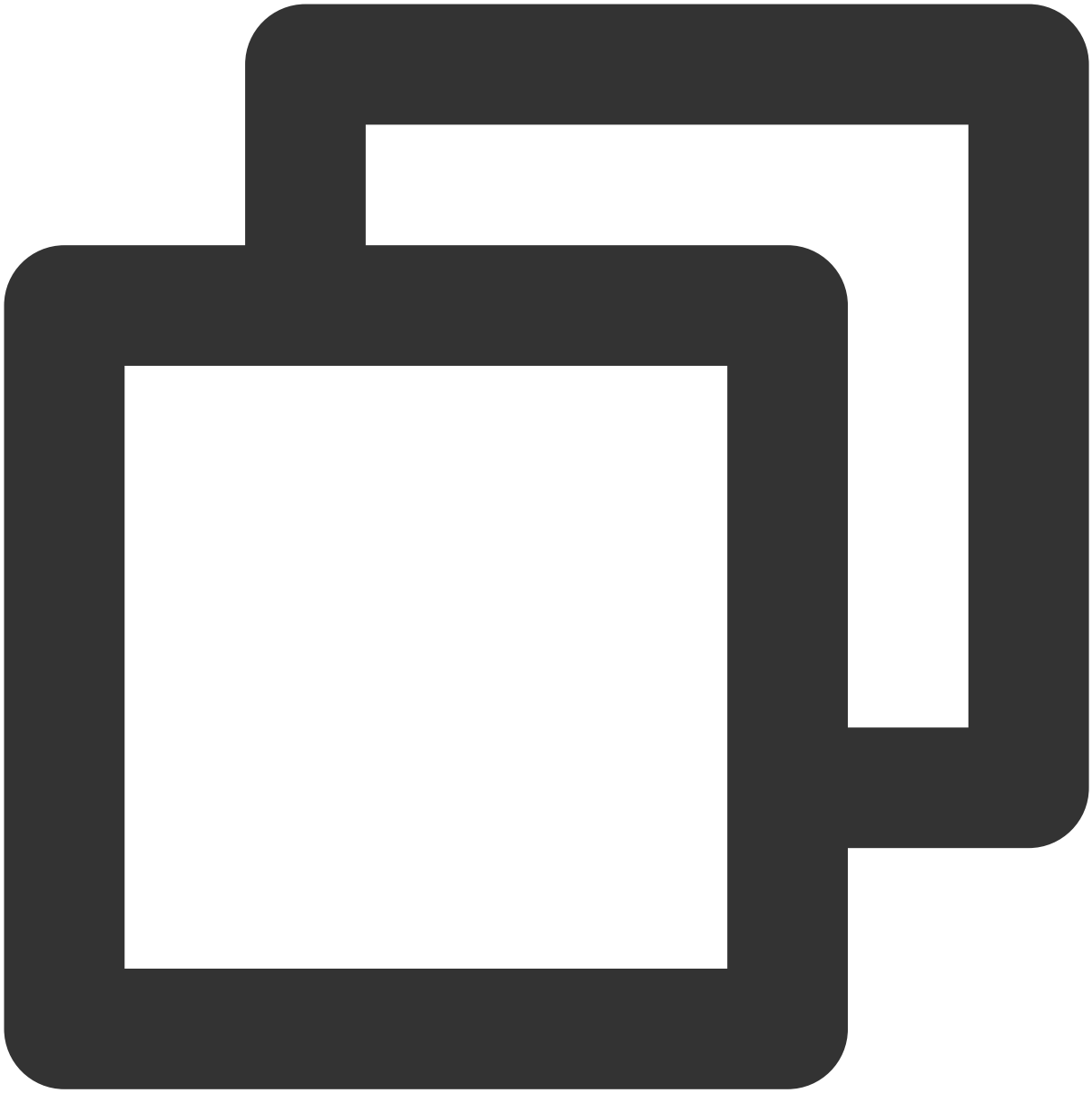
声明文件格式参见

声明文件是一个普通的文本文件，空行与 `#` 开始的行会被忽略，每行的格式为：



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

`pattern` 指定了一个文件路径模式。email 为所有者邮箱，可以填写多个所有者，以空格分隔。示例文件：



```
# 声明所有后缀名是 js 的文件
*.js yourname@coding.net

# 声明仓库根目录下 build/logs/ 目录内的文件（包括子目录）
/build/logs/ yourname@coding.net

# 声明所有 docs/ 文件夹内的文件（不包括子目录）
docs/* yourname@coding.net

# 声明所有根目录下 docs/ 文件夹内的文件（包括子目录）
/docs/ yourname@coding.net
```

合并请求与代码评审
调整合并请求设置

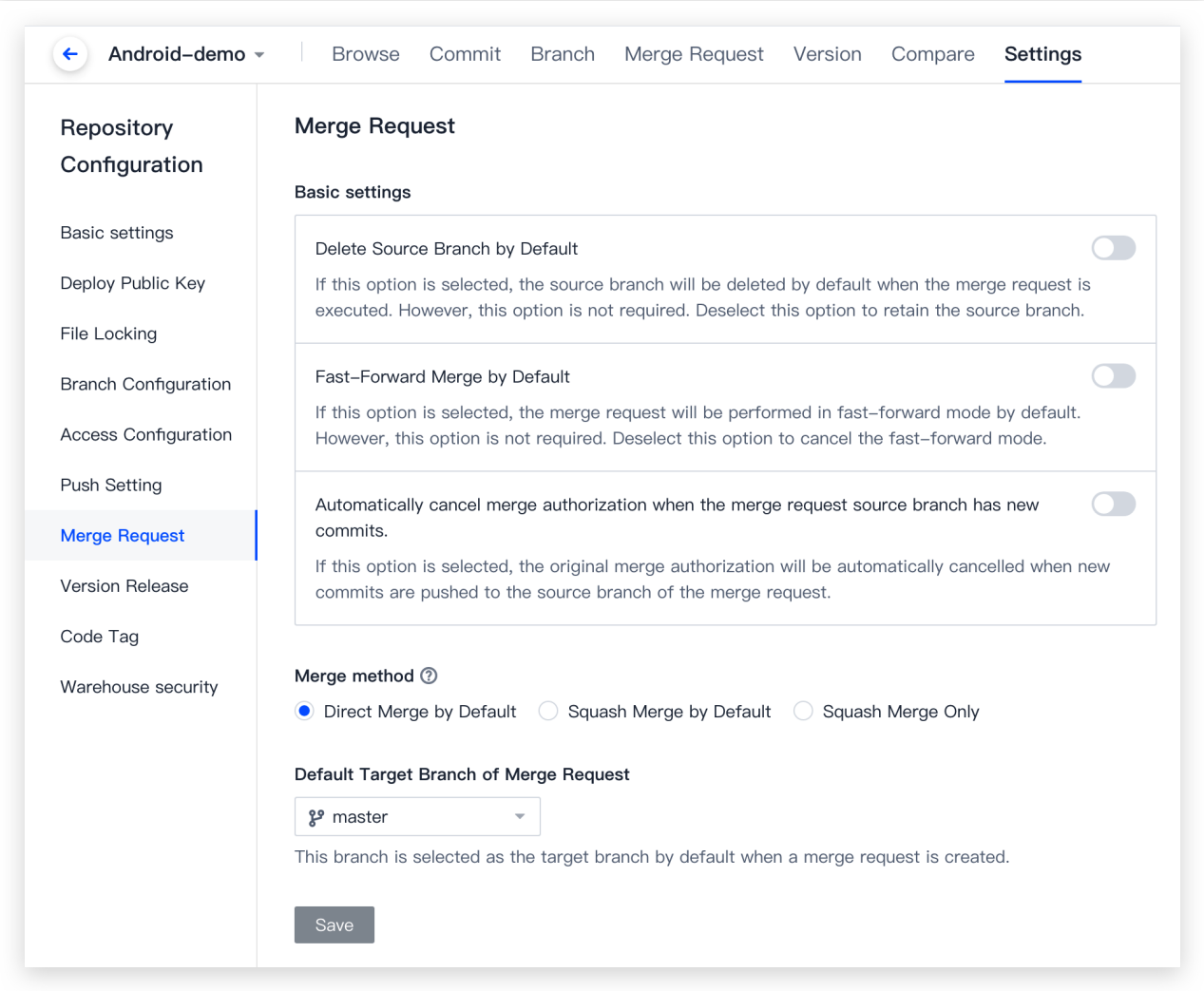
最近更新时间：2023-12-25 17:08:18

本文为您详细介绍如何在代码仓库中调整合并请求设置。

进入项目

1. 登录 CODING 控制台，单击团队域名进入 CODING 使用页面。
2. 单击页面右上角的

，进入项目列表页面，单击项目图标进入目标项目。
3. 单击左侧菜单栏的**代码仓库** > **合并请求**，进入功能页面。
项目管理员可以在代码仓库的**设置** > **合并请求**页面设置合并请求的基础设置、默认的合并方式与目标分支。



是否默认删除源分支

如果开启此项开关，那么当发起请求的源分支被并入至目标分支后，源分支会被自动删除。

是否默认以 Fast-Forward 模式合并

如果该功能开启，那么当源分支是目标分支的直接上游时，源分支会直接指向目标分支，而不会产生一个合并提交。此过程称为 Fast-Forward 合并模式。

合并方式

当源分支有多个提交的时候，我们会提供三种合并模式：

默认直接合并：会产生一个合并提交
默认 Squash 合并：会把源分支的多个提交合并成一个提交，用户可以取消这个行为
只能 Squash 合并：强制把源分支的多个提交合并成一个提交，用户不能取消

默认目标分支

指定了默认分支之后，在创建合并请求时会自动填充该分支为目标分支。建议使用主干分支为合并请求的默认目标分支。

合并分支

最近更新时间：2023-12-25 17:08:18

本文为您详细介绍如何合并分支。

进入项目

1. 登录 CODING 控制台，单击团队域名进入 CODING 使用页面。
2. 单击页面右上角的

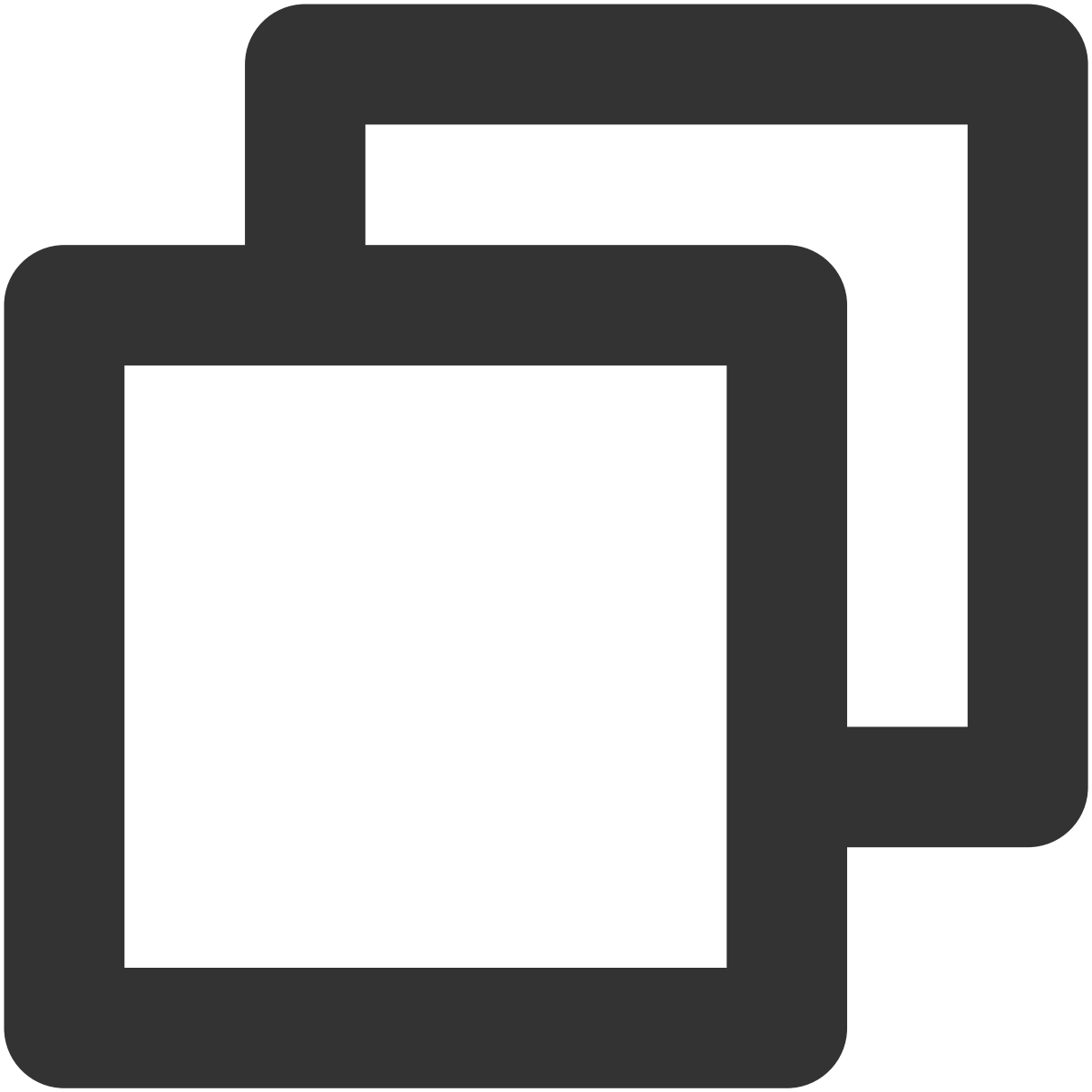
，进入项目列表页面，单击项目图标进入目标项目。
3. 选择左侧菜单**代码仓库 > 分支**，进入分支管理页面。

在采用多分支开发工作流时，建议开发组长将主分支设置为 保护分支，开发者创建临时开发分支，完成后向主分支发起合并请求。通过持续集成和代码评审后，开发者将开发分支合并至主分支。

创建合并请求

支持通过命令行或在 CODING DevOps 平台手动创建合并请求。

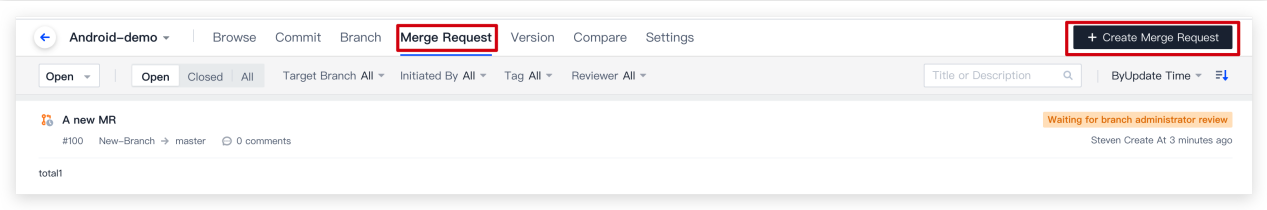
通过命令行创建：



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

手动创建：

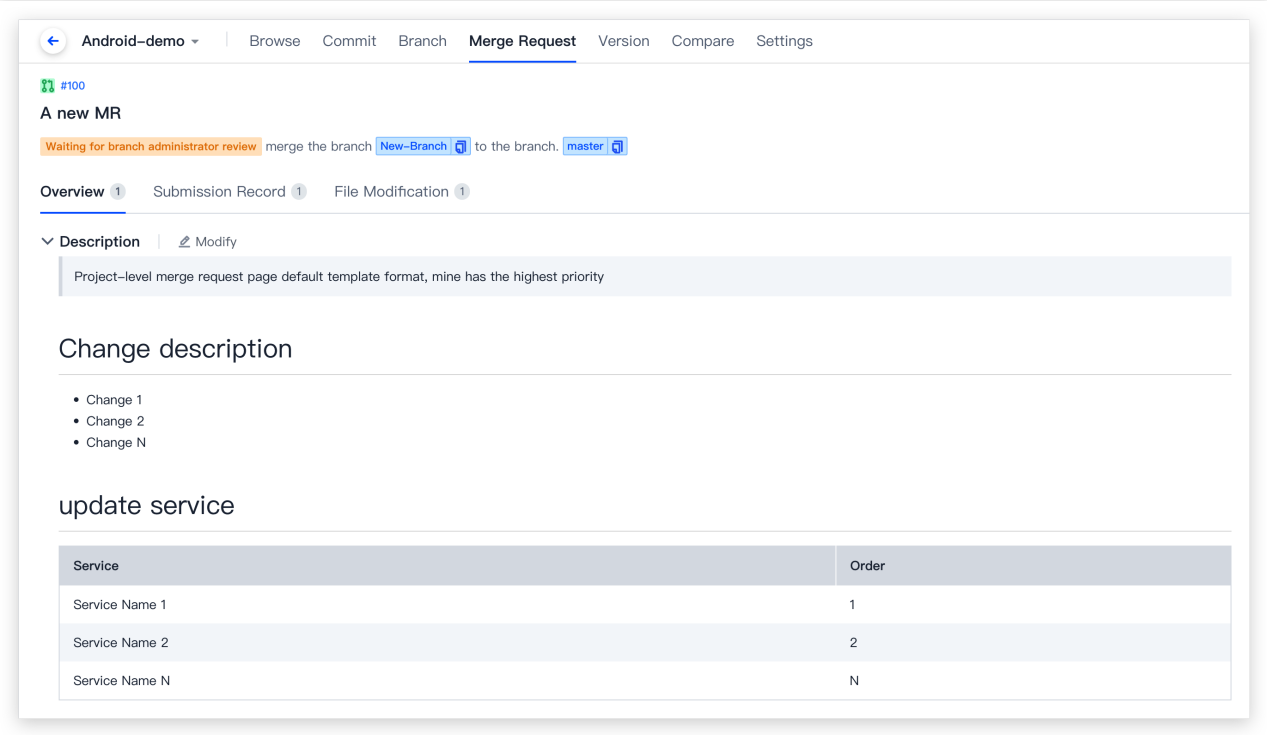
1. 在代码仓库详情页面，进入**合并请求**页签，然后单击右上角**创建**。



2. 指定合并请求的源分支与目标分支。当源分支与目标分支对比后没有发现冲突，系统提示源分支可合并到目标分支。您可以通过**文件改动**页签查看文件差异。

说明：

通过 设置合并请求 可指定合并请求的默认目标分支。



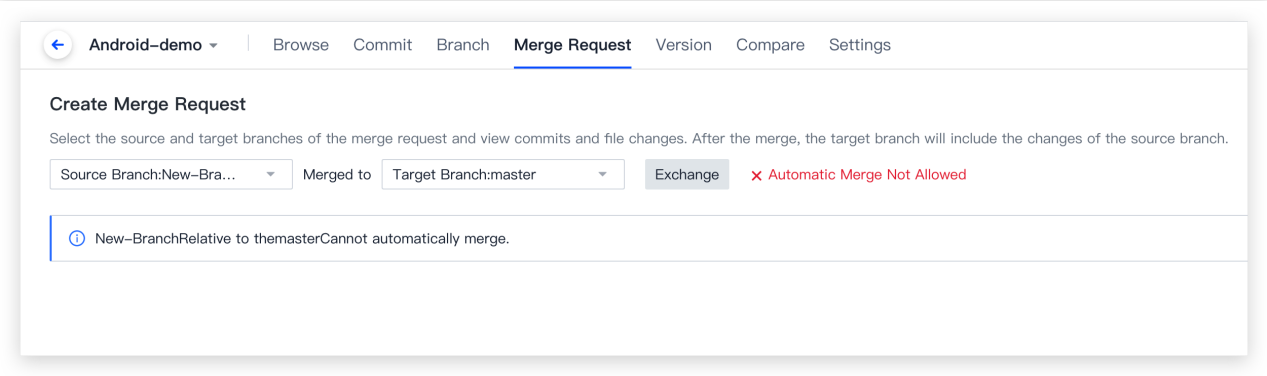
3. 填写合并请求的标题、描述，关联资源。

说明：

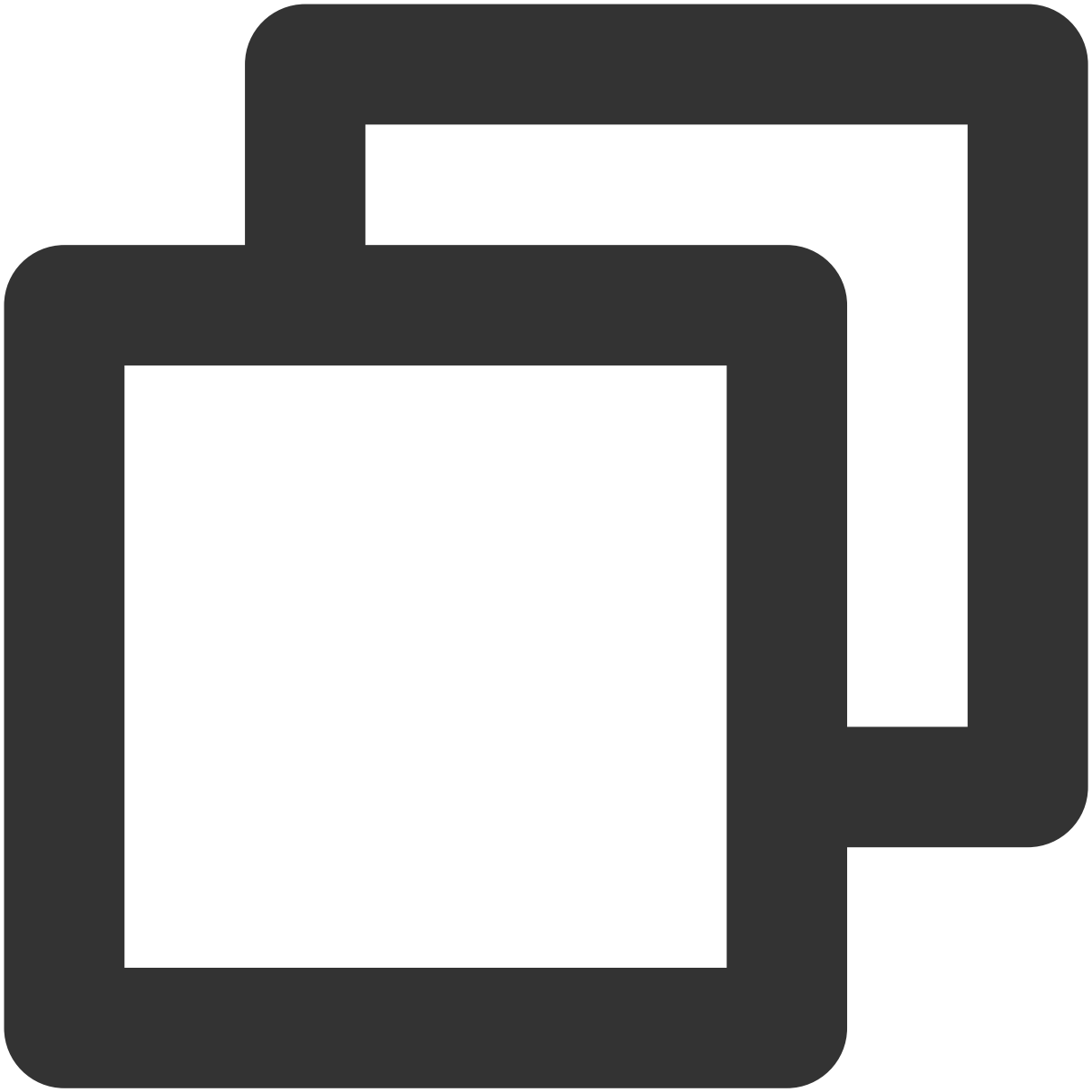
当源分支的提交不领先于目标分支时，不可创建合并请求。

解决不可自动合并的请求

当源分支与目标分支对比后发现存在冲突，系统提示源分支和目标分支不可自动合并。您可以在**对比**页签查看文件改动记录，解决冲突后才能继续合并分支。

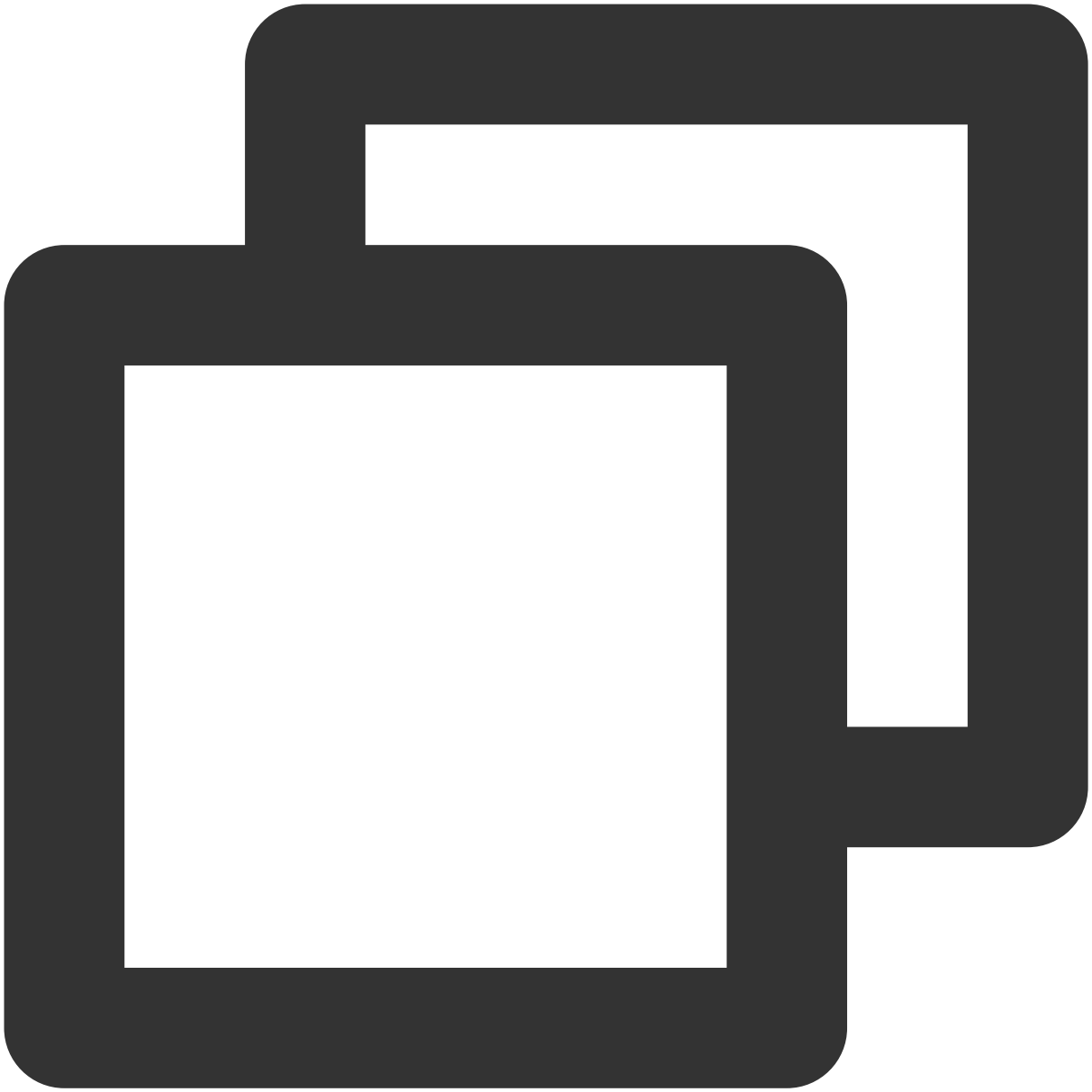


例如：当 `branch-01` 合并入 `master` 分支时提示有冲突时，可以先在本地切换至 `master` 分支并运行命令：



```
git merge branch-01
```

找到冲突文件，此时冲突文件会标识冲突内容并询问保留何种内容。选择需保留的内容，保存后重新提交 commit，完成后切换至 `branch-01` 分支并输入命令：



```
git merge master
```

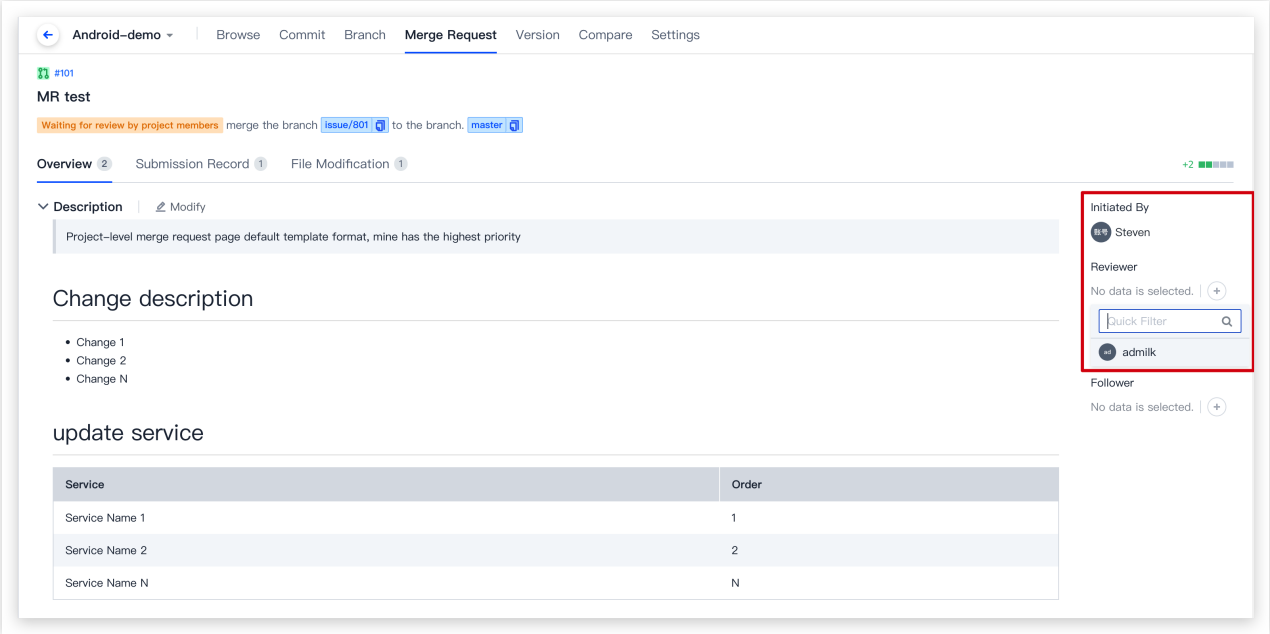
再将修改后的代码推送至远端仓库即可。

发起评审

在发起一次分支合并请求时，建议让相关人员参与到代码审视中，以确保准合并代码的正确性。

说明：

如果合并请求的目标分支为 保护分支，默认会自动添加分支管理员为评审者。如需更改设置，参见 保护分支规则。



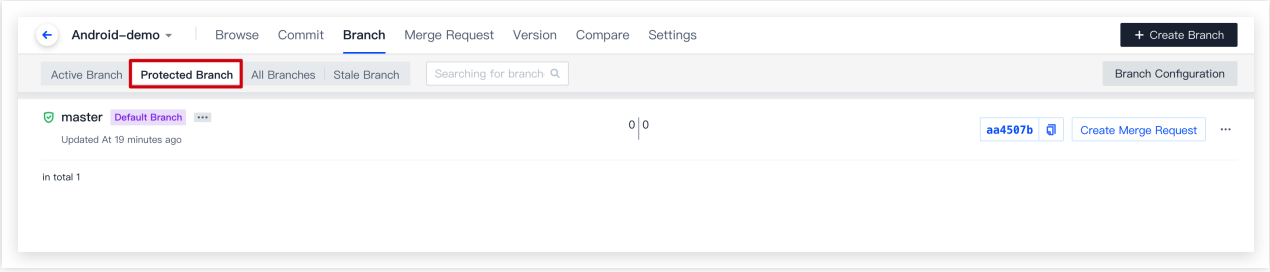
评审者在完成 合并请求评审 之后，将会在该合并请求详情页面显示评审结果。

检查合并状态

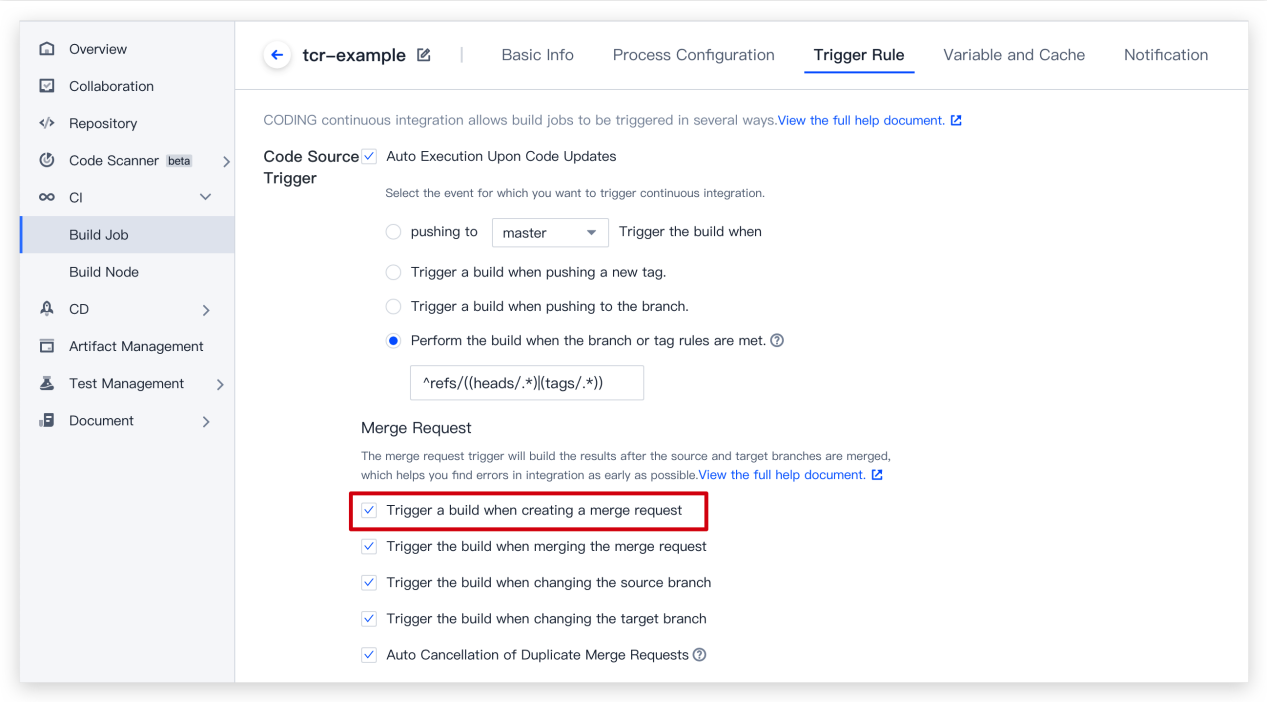
除了上述常见的人工代码审查外，结合 CODING 持续集成，我们还提供了自动化代码评审工具集成的解决方案。基于预定义的规则先行对代码进行扫描，当代码质量有问题时会拒绝合并代码。只有通过了自动化工具检查的代码才允许合并，显著提高代码审查效率。

开启状态检查

仅 保护分支 支持开启状态检查。勾选之后，要求状态检查（CI 任务）全部运行通过后才允许合并。

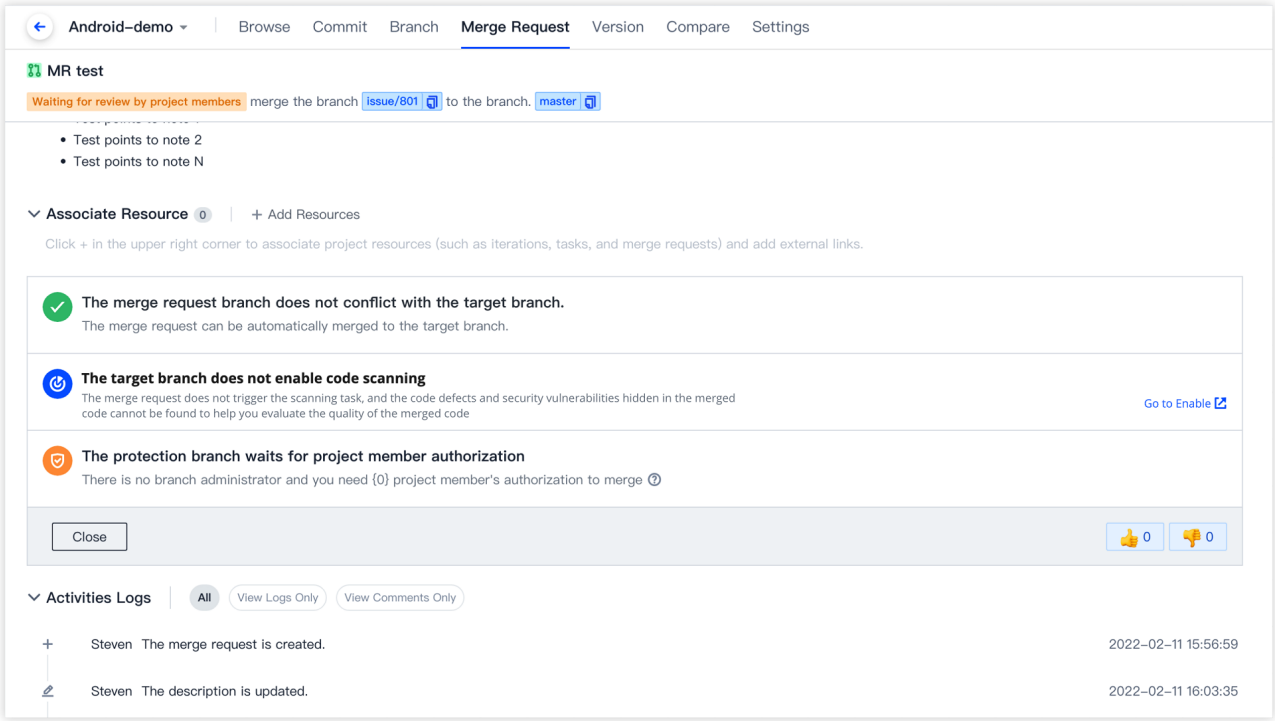


在持续集成中的触发规则内需要选中**创建合并请求时触发构建**，这样才能在创建合并后立即触发构建任务。



查看状态检查结果

完成上述设置后，在正确触发构建任务的情况下，可以看到合并检查的状态。如果没有显示如图界面，可能是没有在 CI 构建任务中选择**创建合并请求时触发构建**。



您可以单击右上角的刷新按钮随时获取最新状态，成功后会在下方出现分支已经合并的提示，失败则会拒绝合并。

状态检查有四种状态：

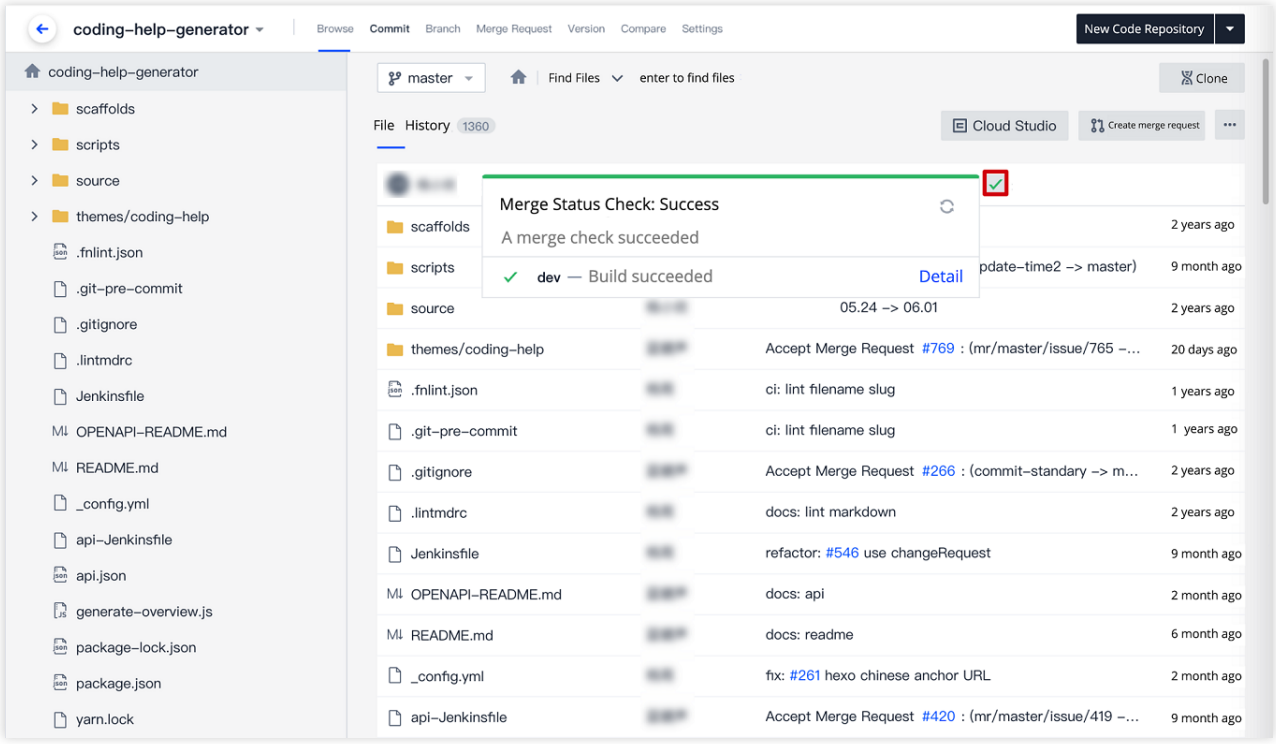
进行中：您可以耐心地等待构建完成。

成功：此时合并请求可以正常合并。

失败：构建过程发生错误，合并检查不通过，您可以修改代码、推送触发新的构建任务直到构建成功完成。

异常：构建过程发生了异常，可以尝试进行手动触发。

如果存在多个状态检查的情况下，只有所有的状态检查都成功通过后才会允许分支合并。您也可以在代码浏览、提交历史、分支列表中查看状态检查过程。



确认合并

合并的目标分支为保护分支

若合并请求的发起者为分支管理员，那么可以自行合并。若发起者为普通成员，那么需要通过分支管理员的评审才能完成合并。

合并的目标分支为非保护分支

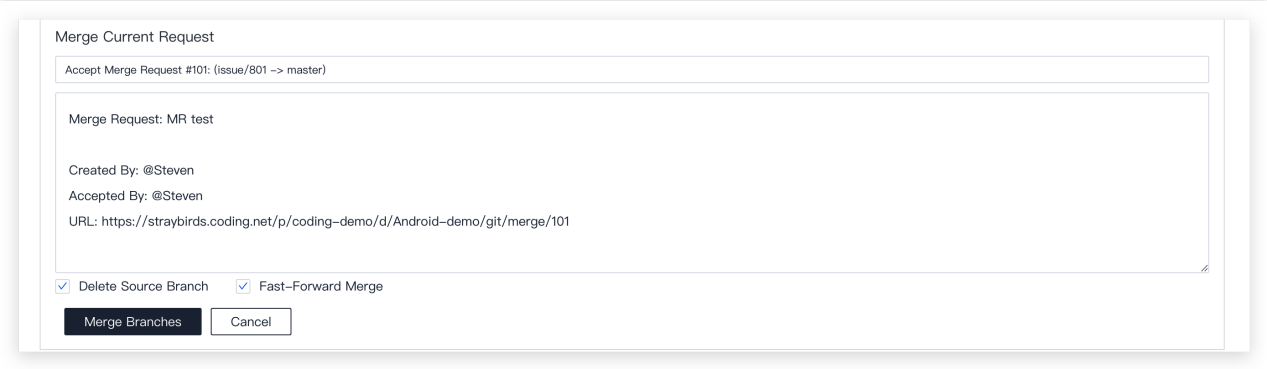
发起者无需经过评审与授权，可以自行发起并完成分支合并。

说明：

如需了解如何修改默认分支与设置保护分支，参见 设置默认分支 或 设置保护分支。

删除源分支

合并分支时，勾选删除源分支，可在合并分支时删除源分支。



Fast-Forward 模式合并

常规的分支合并时会默认产生一个合并提交记录。若勾选了 **Fast-Forward 模式合并**，远端仓库会在合并时判断是否符合 Fast-Forward 规则。若符合则此合并不会产生新的合并提交记录；若不勾选此模式，则会在合并时保留过往开发记录并产生一个新的合并记录。此选项相当于使用 `git merge` 时添加 `-ff` 参数。

说明：

通过 设置合并请求 可实现合并请求默认删除源分支及默认以 Fast-Forward 模式合并。

评审合并请求

最近更新时间：2023-12-25 17:08:18

本文为您详细介绍如何评审合并请求。

进入项目

1. 登录 CODING 控制台，单击团队域名进入 CODING 使用页面。
2. 单击页面右上角的

，进入项目列表页面，单击项目图标进入目标项目。
3. 选择左侧菜单**代码仓库** > **合并请求**，进入合并请求功能页面。

在新建合并请求或合并请求合并前，开发者可更新标题和描述，添加其他项目成员为评审者，并关联项目内资源（如任务、文件、合并请求、Wiki）。您还可以通过 CI 插件 自动添加评审者。

说明：

如果合并请求的目标分支为保护分支，且为保护分支配置了管理员并开启**自动添加分支管理员为评审者**，对应授权数量的分支管理员会自动添加为评审者。

评审内容

评审者收到邀请评审通知后，对代码进行评审的内容一般会聚焦于以下两点：

该合并请求的标题、描述及关联资源是否对代码改动作出充分说明，评审者可通过评论和发起者进行沟通确认。

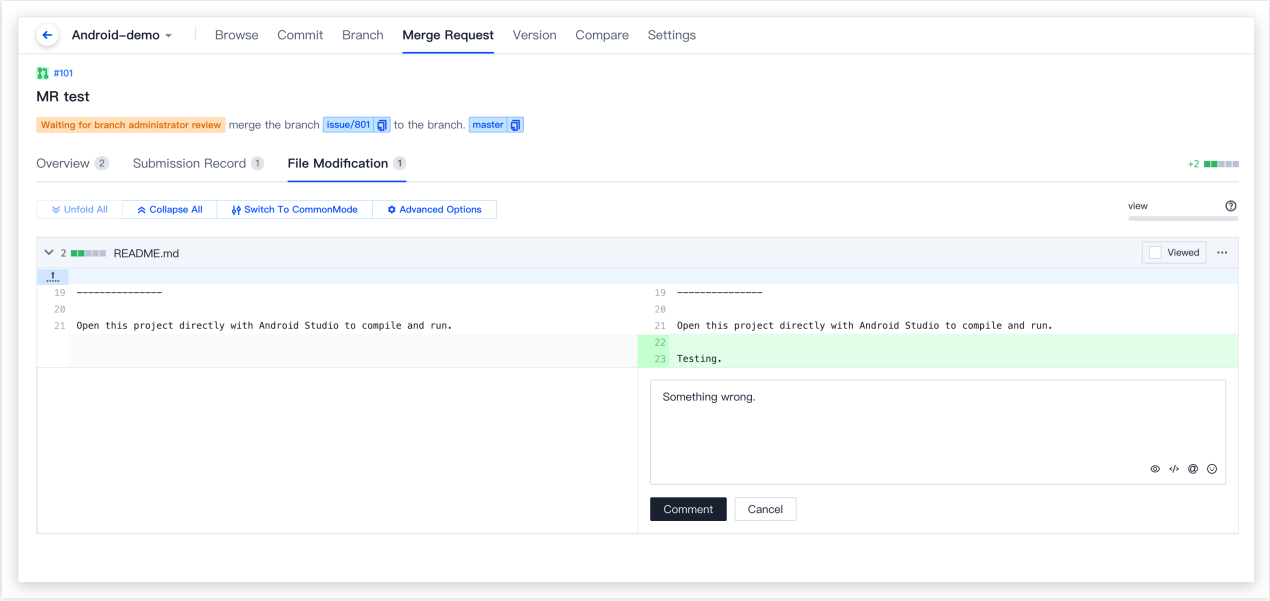
针对提交记录对应的文件改动，进行代码行级评审（评论）。

开始评审

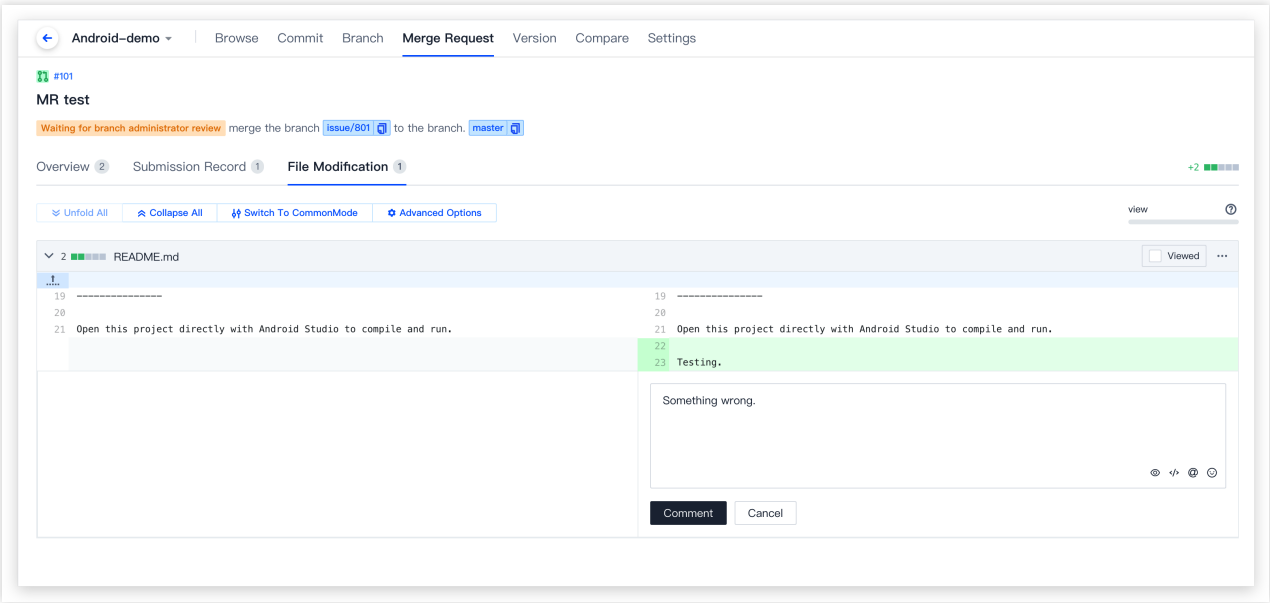
在**合并请求**的**文件改动**页签，代码评审者可以针对代码文件进行逐行评论以完成评审。鼠标悬停在代码文件中的某一行，界面会显示出 `+` 号。单击即可针对该代码行进行评论。填写评论之后：

单击**评论**，直接发布该评论。

单击**开始评审**，则发布评论的同时，该合并请求还会显示**评审中**的状态。



评论发布成功之后，评论内容和评审状态（如有）会同步显示在合并请求概览页面的活动日志。

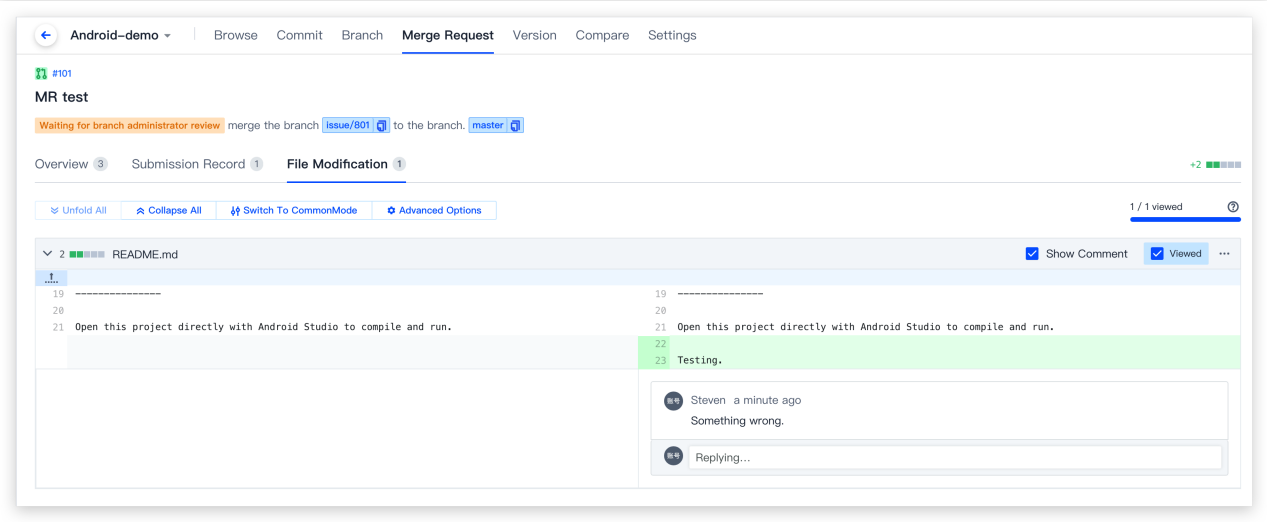


跟踪文件查看进度

当需要评审多个代码文件时，可在评审完一个文件之后单击**已查看**进行跟踪与标记，文件查看进度条会随之自动更新。

说明：

文件**已查看**并不影响评审状态，仅用于跟踪文件查阅进度，且只针对当前用户生效。



完成评审

待所有代码文件均已被评审之后，可单击右上角**完成评审**发布评审结果并结束评审。

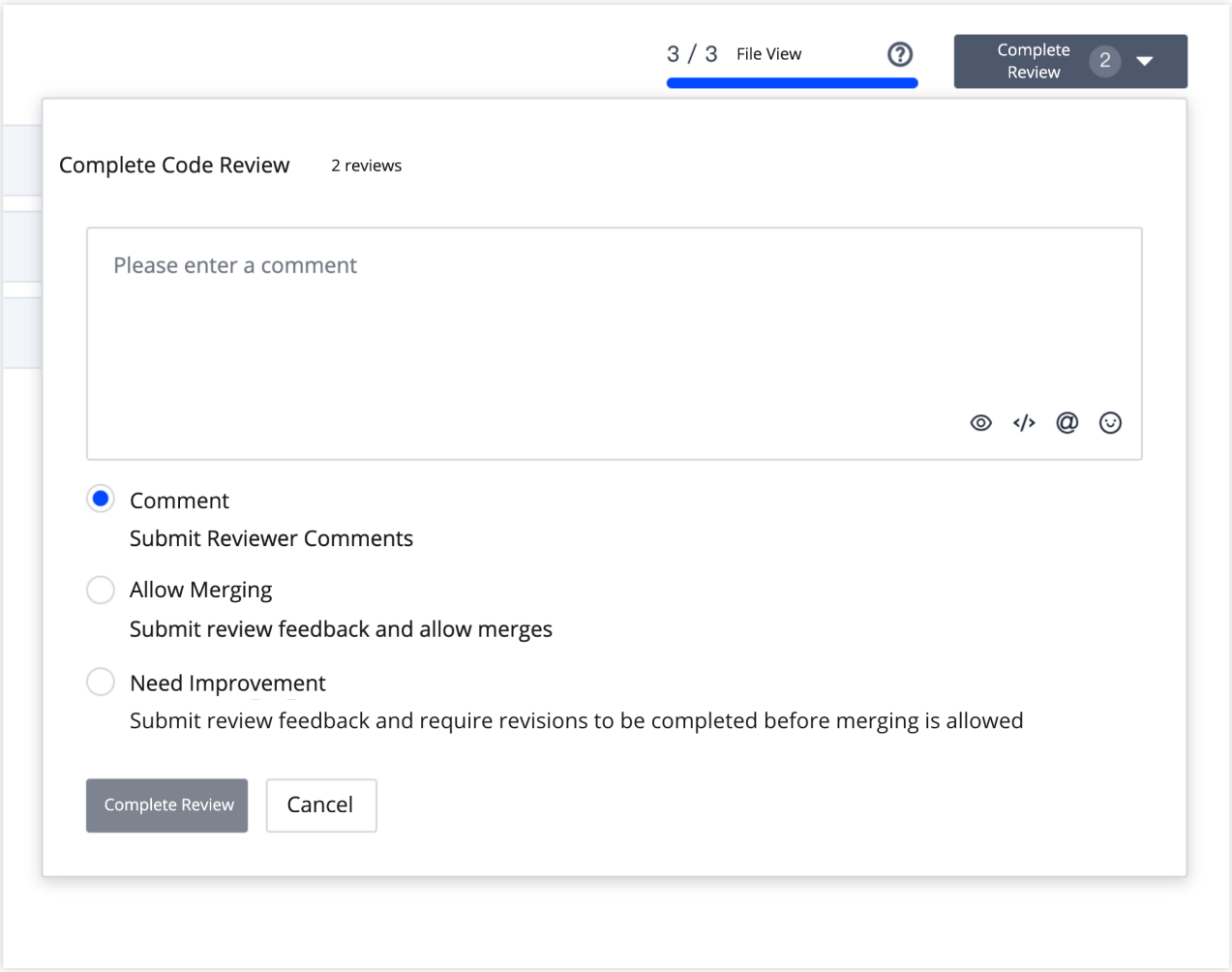
说明：

完成评审上的数字代表当前页面所有代码评论的数量。

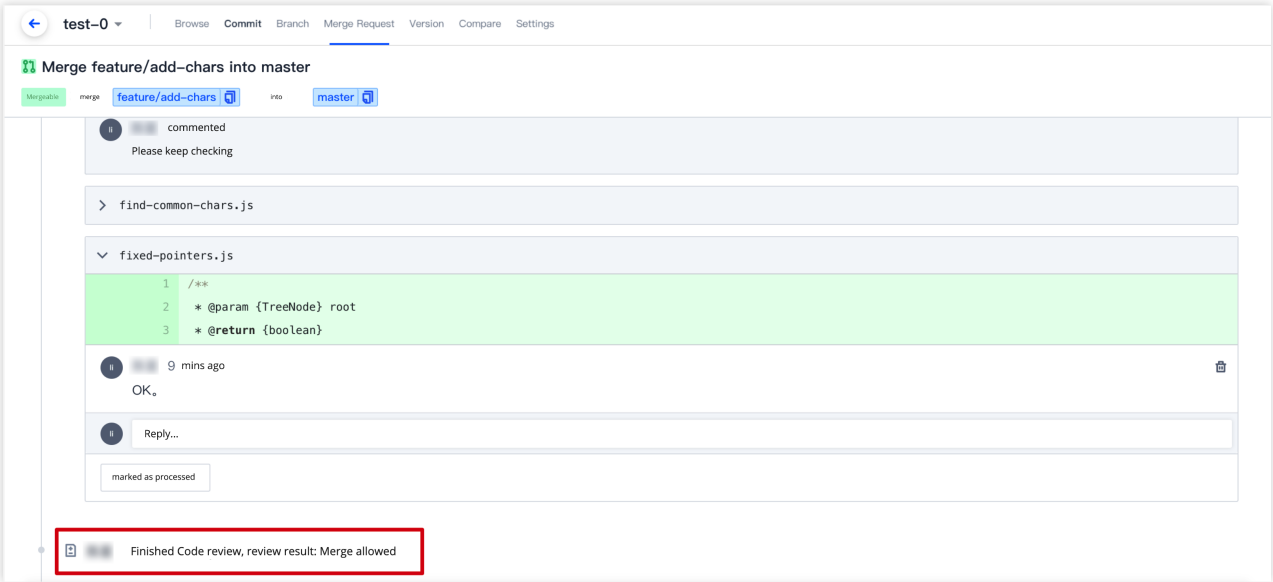
评论：必需填写评论。评论内容为填写内容。

允许合并：可不输入评论，直接发布评审结果为：**允许合并**。

需要改进：可不输入评论，直接发布评审结果为：**需要改进**。

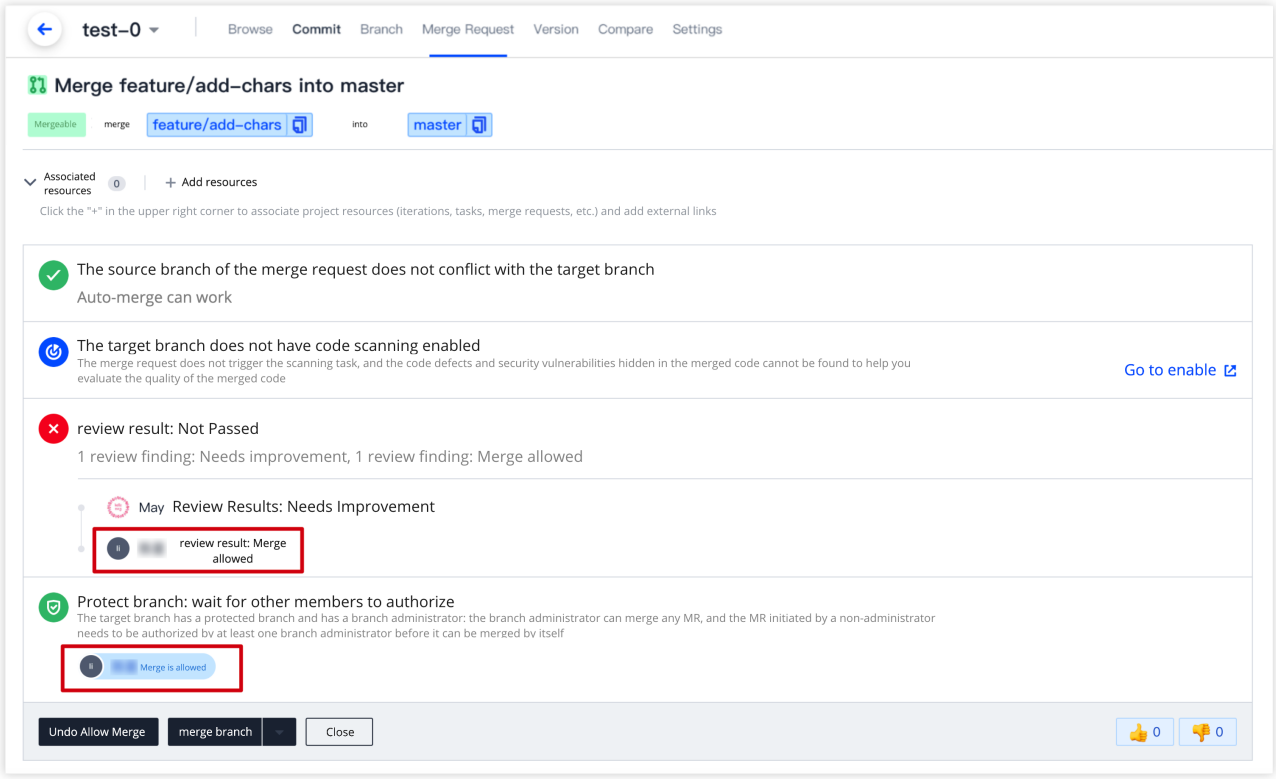


完成评审之后，评论内容或评审结果会同步显示在合并请求概览页面的活动日志。



若合并请求的目标分支为保护分支，将会显示合并请求的评审状态（只要存在**需要改进**的评审结果，评审状态即为不通过）。

若评审结果为**允许合并**，无论目标分支是否是保护分支，该结果还会作为标签显示在分支状态中。



版本与标签
管理版本发布

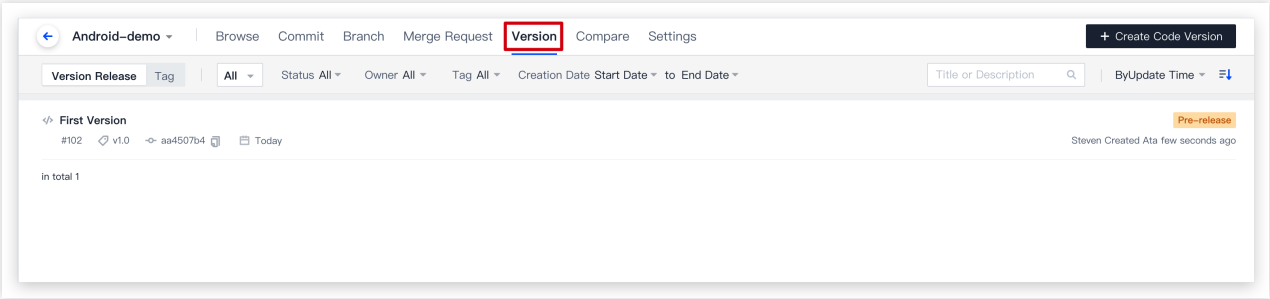
最近更新时间：2023-12-25 17:08:18

本文为您详细介绍如何使用代码仓库中的版本发布功能。

进入项目

1. 登录 CODING 控制台，单击团队域名进入 CODING 使用页面。
2. 单击页面右上角的

，进入项目列表页面，单击项目图标进入目标项目。
3. 单击左侧菜单栏的**代码仓库 > 版本**，进入版本功能页面。
在代码仓库管理列表，单击指定代码仓库进入其详情页面之后，再单击**版本 > 版本发布**即可进入其版本发布列表。



版本发布列表按创建时间倒序列出项目已发布的代码版本，并展示了代码版本对应的标签名、提交的代码版本等信息。

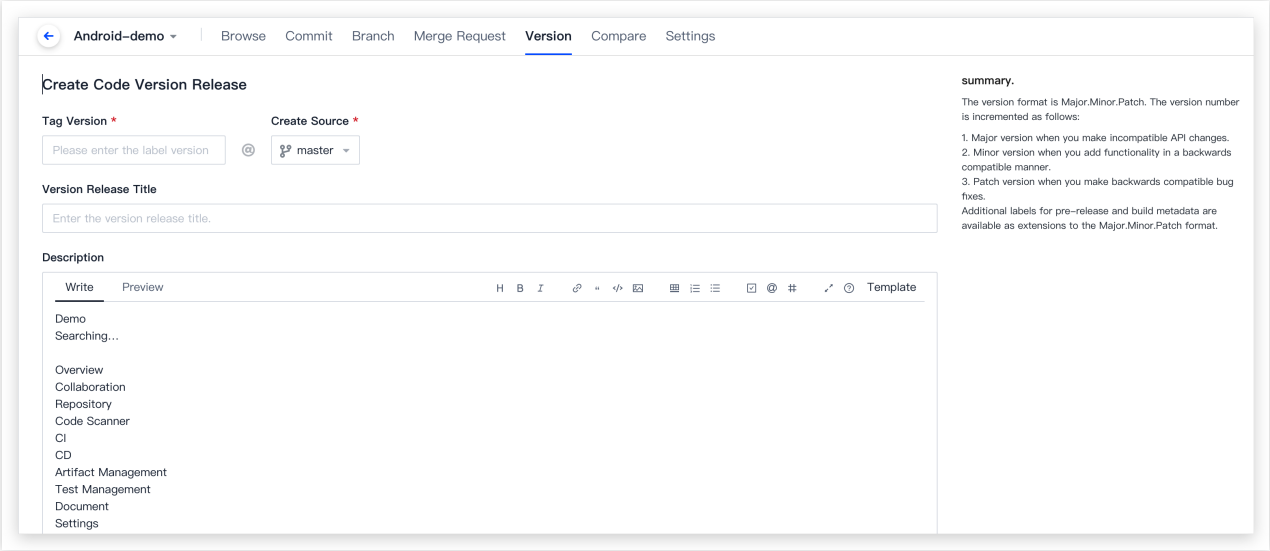
新建代码版本

1. 在版本管理页面，单击右上角的新建代码版本按钮。

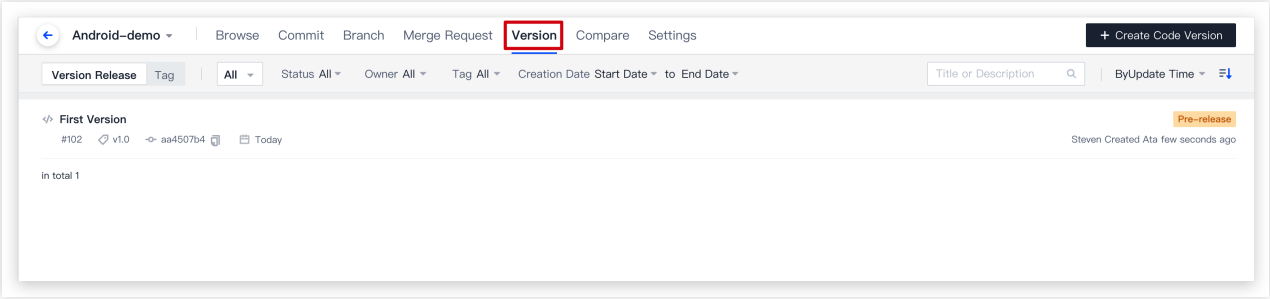
说明：

您可以在代码仓库设置页面 设置版本发布默认分支。

2. 输入标签版本、版本发布标题、版本描述等，支持上传 100 MB 以下所有格式的文件以及关联项目内的资源（如任务、文件、Wiki、合并请求）。其中，标签名只能输出新标签名，需要给新标签版本选择创建来源（分支、标签或修订版本）。

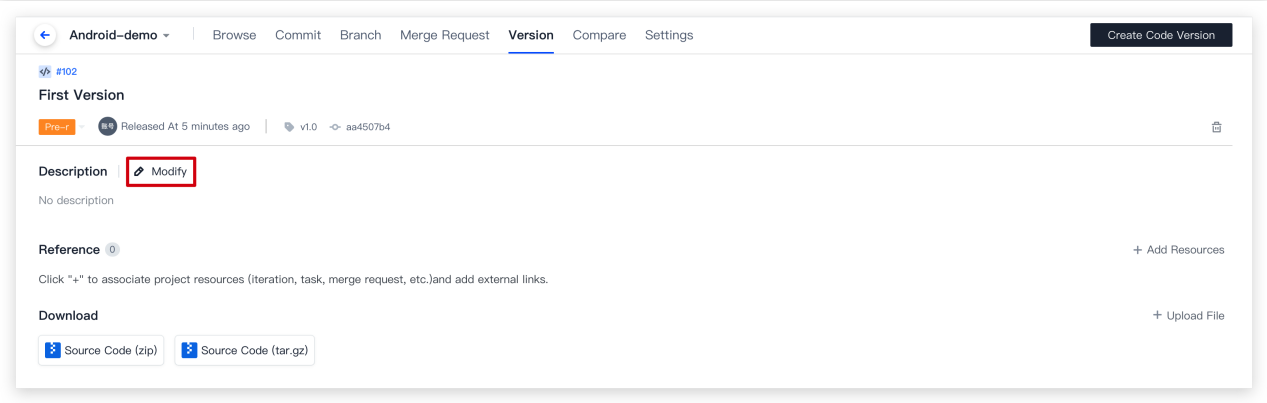


3. 在填写以上信息后，您可以标记为该版本为预发布状态，然后创建代码版本发布。代码版本发布后，显示情况如下图。最近一次正式版本发布为最新发布。



编辑代码版本

在版本发布列表单击任一代码版本进入其详情页面之后，版本发布创建者或项目管理员单击编辑版本描述按钮，即可编辑除标签名以外的其他所有信息。

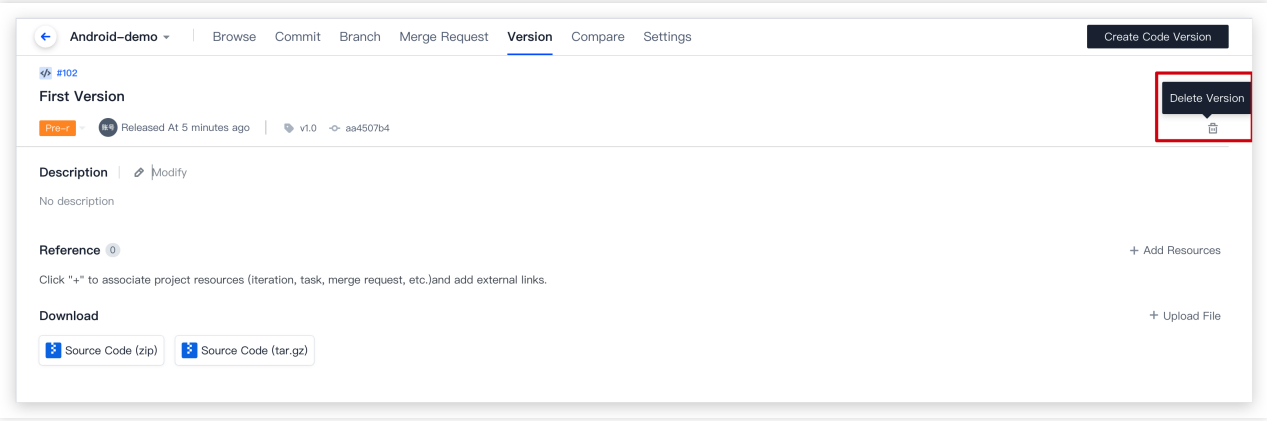


删除代码版本

在代码版本详情页面，版本发布创建者或项目管理员单击删除图标，即可删除该版本。

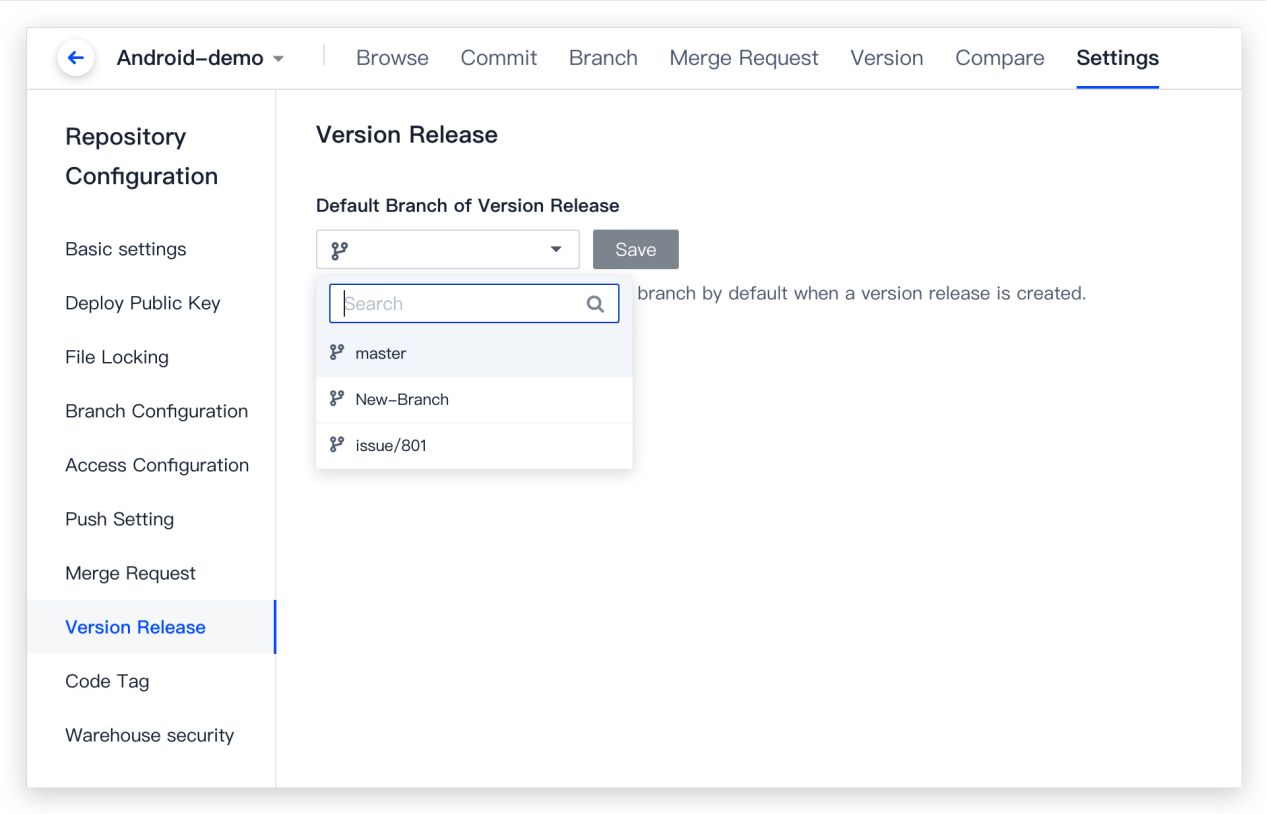
说明：

在删除版本时，可同时将对应的版本标签删除。只有删除了对应的版本标签，才能使用或创建同名的标签。



设置版本发布默认分支

在代码仓库的**设置** > **版本发布**页面，项目管理员可以指定版本发布时默认选择的分支。


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
Save
Search
master
New-Branch
issue/801
branch by default when a version release is created.

管理版本标签

最近更新时间：2023-12-25 17:08:18

本文为您详细介绍如何管理版本标签。

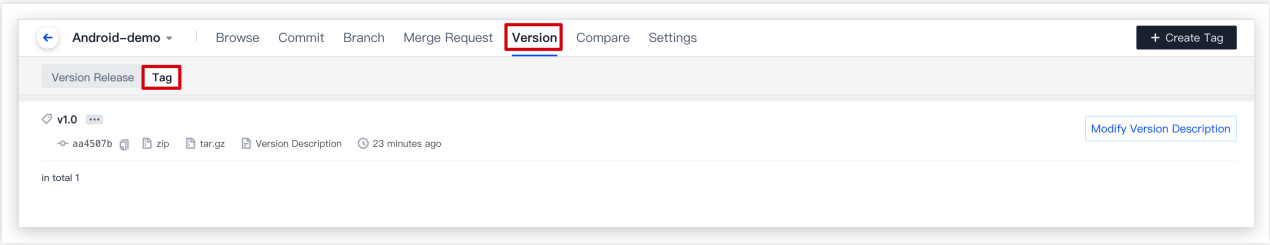
进入项目

1. 登录 CODING 控制台，单击团队域名进入 CODING 使用页面。
2. 单击页面右上角的

，进入项目列表页面，单击项目图标进入目标项目。
3. 选择左侧菜单**代码仓库 > 分支**，进入分支管理页面。

在采用多分支开发工作流时，建议开发组长将主分支设置为保护分支，开发者创建临时开发分支，完成后向主分支发起合并请求。通过持续集成和代码评审后，开发者将开发分支合并至主分支。

在代码仓库管理列表，单击指定代码仓库进入其详情页面之后，再单击**版本 > 标签**即可进入其版本标签管理列表。



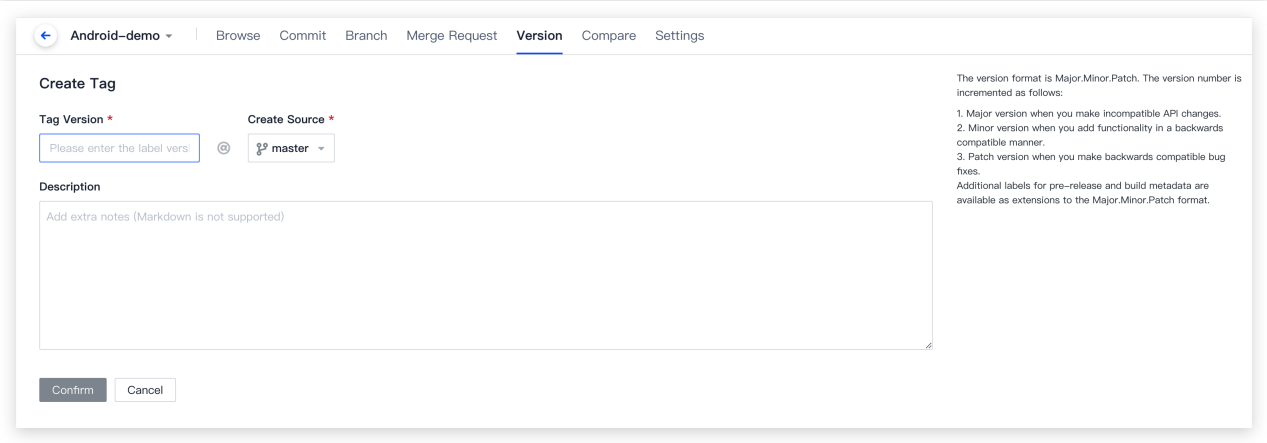
标签列表显示了该仓库所有标签，按照创建顺序倒序排列。标签列表显示了标签名、标签说明、标签对应版本，并提供 zip 和 tar.gz 下载入口和删除标签入口。点击标签名或者版本号可进入对应的代码版本详情页。

新建标签

在标签管理列表，单击右上角**创建建标签**，输入标签名并选择待创建标签对应的代码版本（分支、标签、修订版本号），输入标签说明即可创建新标签。

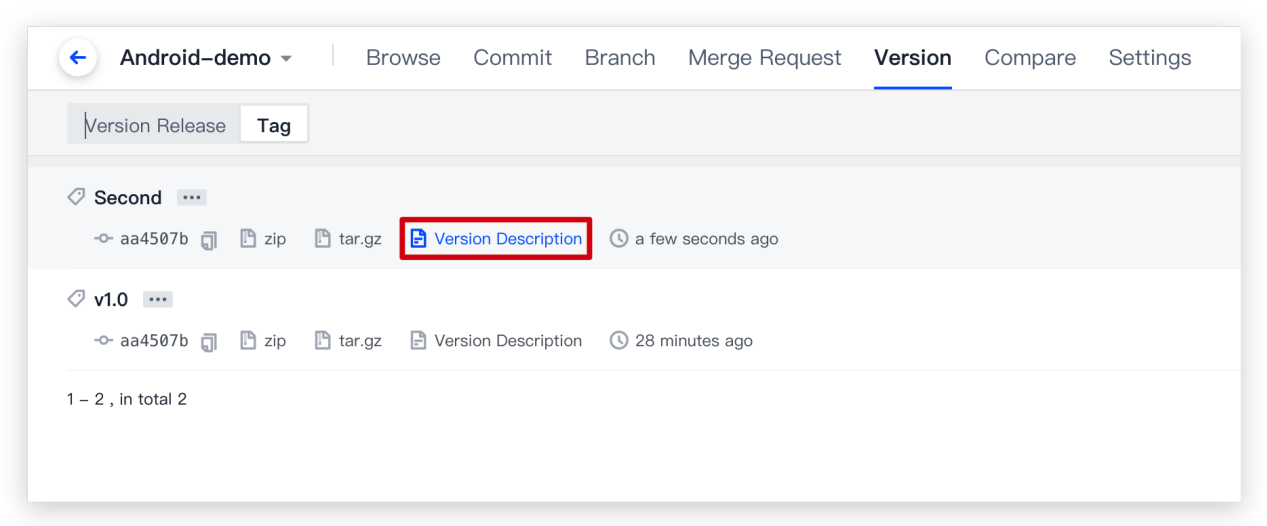
说明：

您可以在代码仓库设置页面 设置保护标签，以规范成员的标签操作。

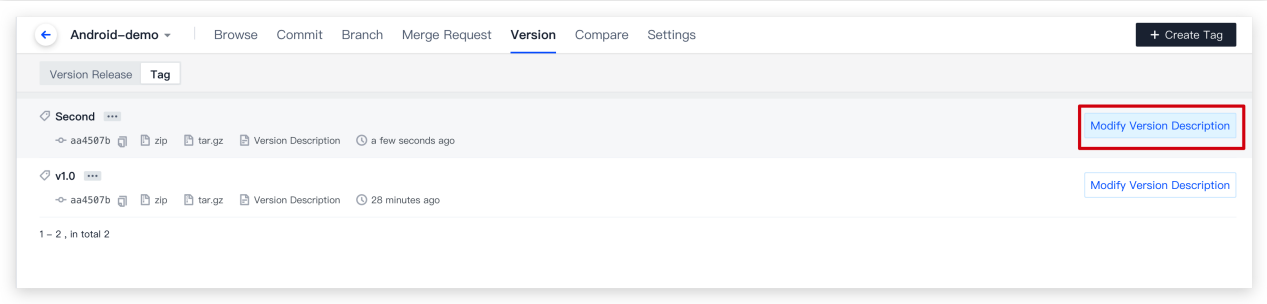


查看标签对应的版本发布信息

若某一代码版本对应该标签，单击**版本描述**可以查看该版本发布详情。单击**编辑版本描述**，即可编辑该版本发布信息。

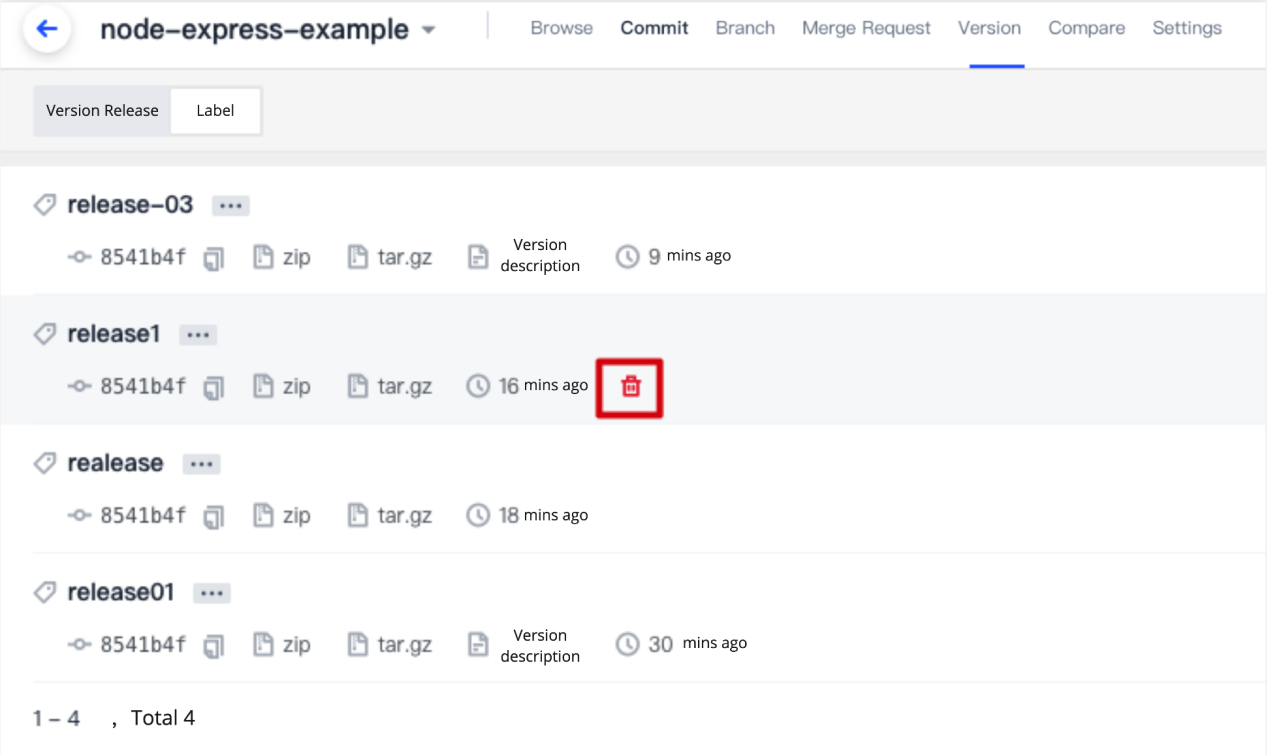


若一个标签并没有对应任何版本发布，可单击**创建版本描述**快速为该标签创建一个发布版本。



删除标签

在**标签**页面，标签创建者或管理员只能删除未与任何版本发布对应的代码标签。

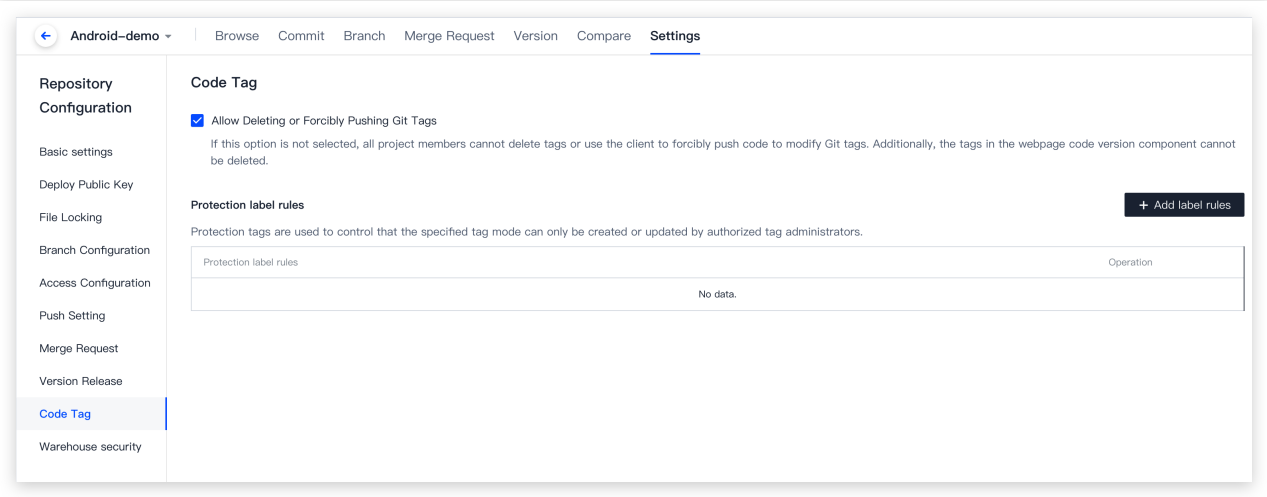


说明：

若任一标签与版本发布关联，只能在**版本发布**删除对应版本的同时将标签删除。

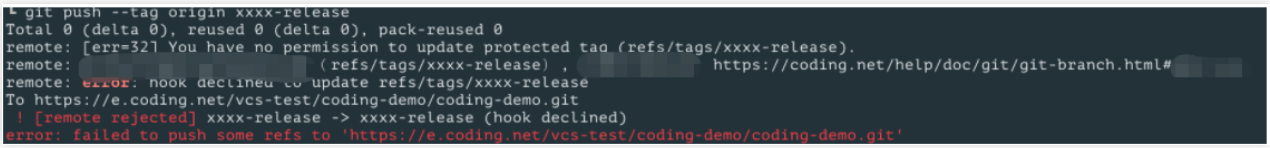
设置是否允许删除或强制推送 Git 标签

在代码仓库的**设置 > 代码标签**页面，项目管理员可以勾选是否允许删除或者强制推送 Git 标签。关闭此选项后，项目中的成员均不可删除或者通过强制推送修改 Git 标签，同时网页上的标签不提供删除功能。

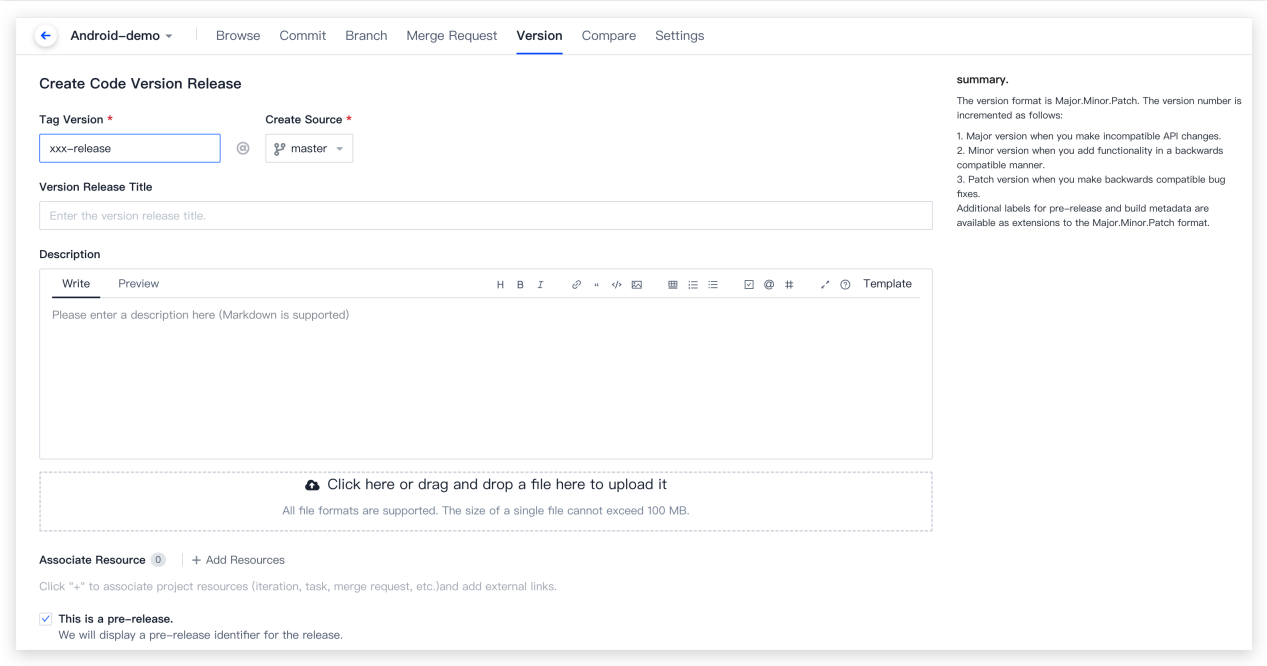


设置保护标签

保护标签主要用于规范特定的成员进行创建、更新或删除标签等操作。开启保护标签后需设置标签管理员，仅管理员被允许在此标签下创建匹配标签规则的标签。当设置了 `*-release` 为保护分支规则之后，非管理员用户通过 Git 推送标签 `xxx-release` 的时候有如下提示：



在 Web 端创建标签或新建版本时也同样会失败。



使用场景举例

某团队使用标签作为触发 CI 构建的条件，即在生产分支中，通过推送 v1.0-release 字样的标签作为发布命令。设置保护标签后，仅标签管理员能够创建此类型标签并完成发版动作，保持了代码仓库内各版本序列的整洁与规范。

代码仓库安全
检查仓库安全风险

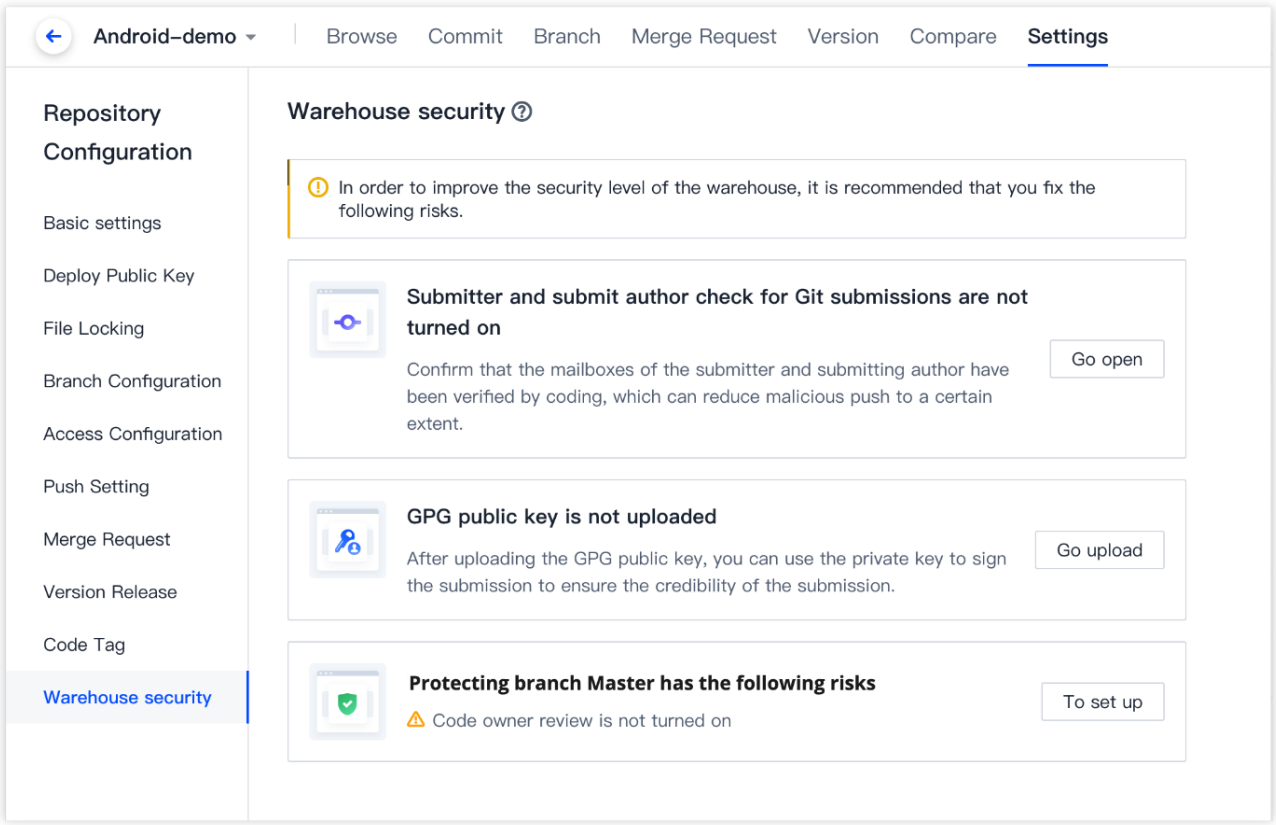
最近更新时间：2023-12-25 17:08:18

本文为您详细介绍如何检查代码仓库中的安全风险。

进入项目

1. 登录 CODING 控制台，单击团队域名进入 CODING 使用页面。
2. 单击页面右上角的

，进入项目列表页面，单击项目图标进入目标项目。
3. 单击左侧菜单栏的**代码仓库** > **设置**，进入仓库安全页。
项目管理员可以在代码仓库的**设置** > **仓库安全**页面查看该仓库存在的安全风险。



目前，系统支持对代码仓库进行以下检查：

是否开启了 Git 提交者和提交作者检查

是否上传了 GPG 公钥

是否设置了保护分支；若已设置保护分支，是否设置了分支管理员和开启代码所有者评审

说明：

为了提高仓库的安全等级，建议您参见 推送设置、使用 GPG 签名 commit 记录、保护分支 配置对应功能。

设置仓库访问方式

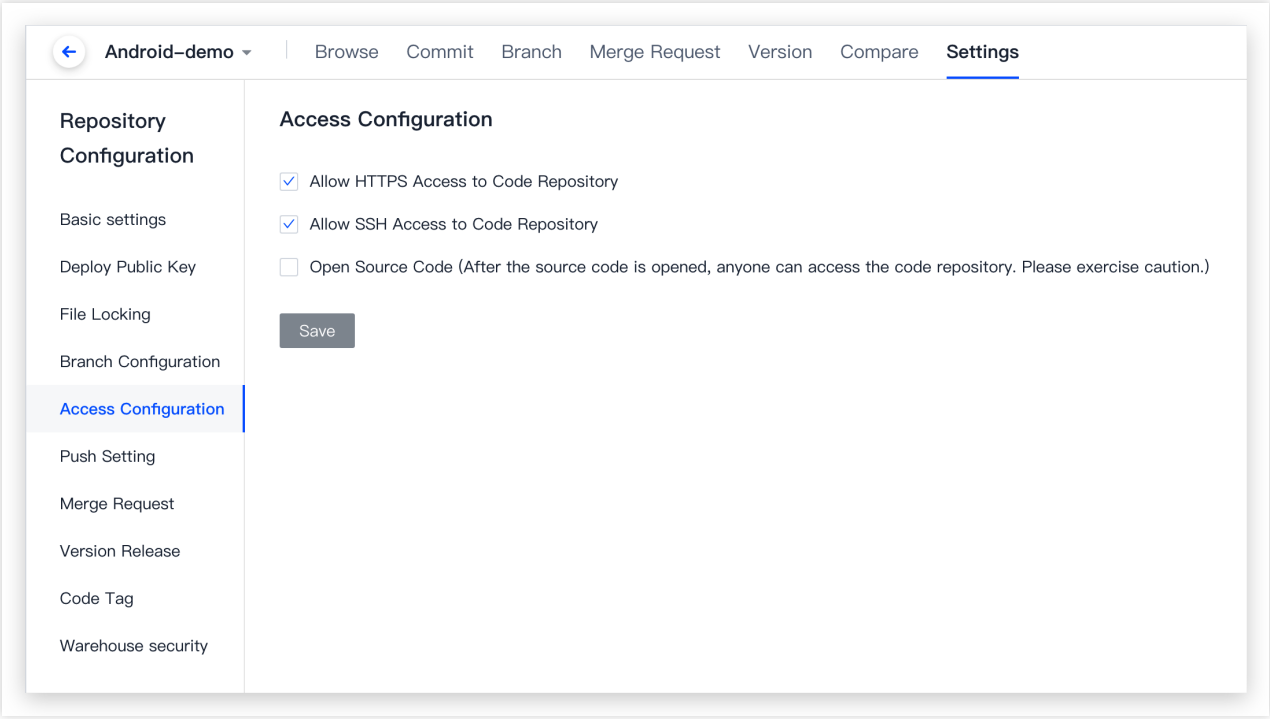
最近更新时间：2023-12-25 17:08:18

本文为您详细介绍如何设置仓库的访问方式。

进入项目

1. 登录 CODING 控制台，单击团队域名进入 CODING 使用页面。
2. 单击页面右上角的

，进入项目列表页面，单击项目图标进入目标项目。
3. 选择左侧菜单**代码仓库 > 设置**，进入仓库的访问设置页面。
项目管理员可以在代码仓库的**设置 > 访问设置**页面选择是否开源代码仓库，还可以设置是否允许 HTTPS 或 SSH 协议访问代码仓库。



使用 GPG 签名 commit 记录

最近更新时间：2023-12-25 17:08:18

本文为您详细介绍如何使用 GPG 签名 commit 记录。

进入项目

1. 登录 CODING 控制台，单击团队域名进入 CODING 使用页面。
2. 单击页面右上角的

，进入项目列表页面，单击项目图标进入目标项目。

CODING 支持使用 GPG 对 Git commit 进行签名验证。对于验证通过的 commit 提交记录，将会打上**已验证**标签，确保代码提交者是可靠来源，增强代码安全性。

使用 GPG 签名 Git commit 时需要执行以下步骤：

步骤 1：生成 GPG 密钥对

步骤 2：添加 GPG 公钥至个人账户设置

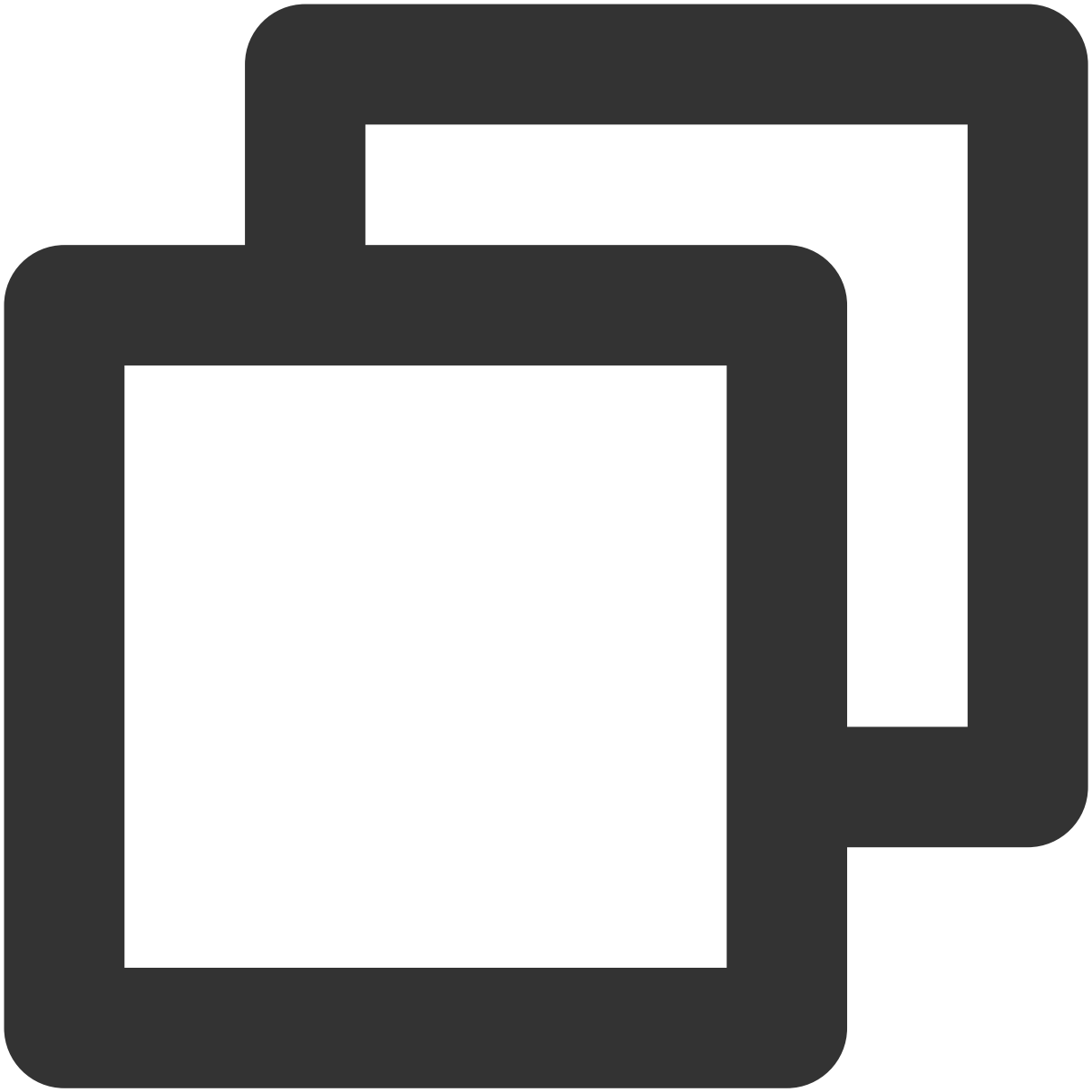
步骤 3：与本地 Git 仓库关联

步骤 4：签名 Git commit

步骤 5：验证签名

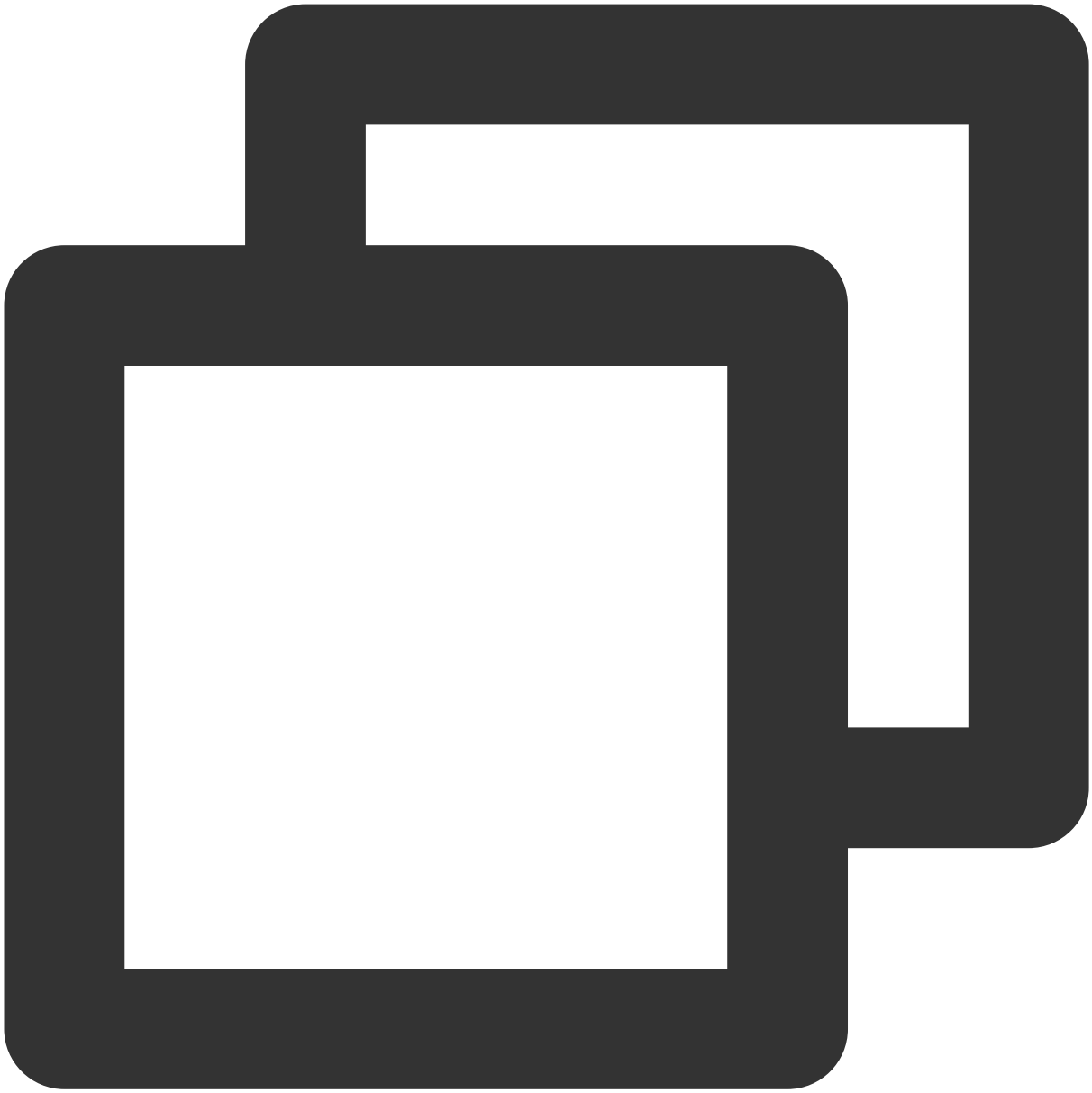
步骤1：生成 GPG 密钥对

1. 下载 并安装 GPG。如果使用 macOS，可直接使用 brew 包管理工具运行以下命令：



```
brew install gpg
```

2. 运行以下命令生成 GPG 密钥对（公钥/私钥）：



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

说明：

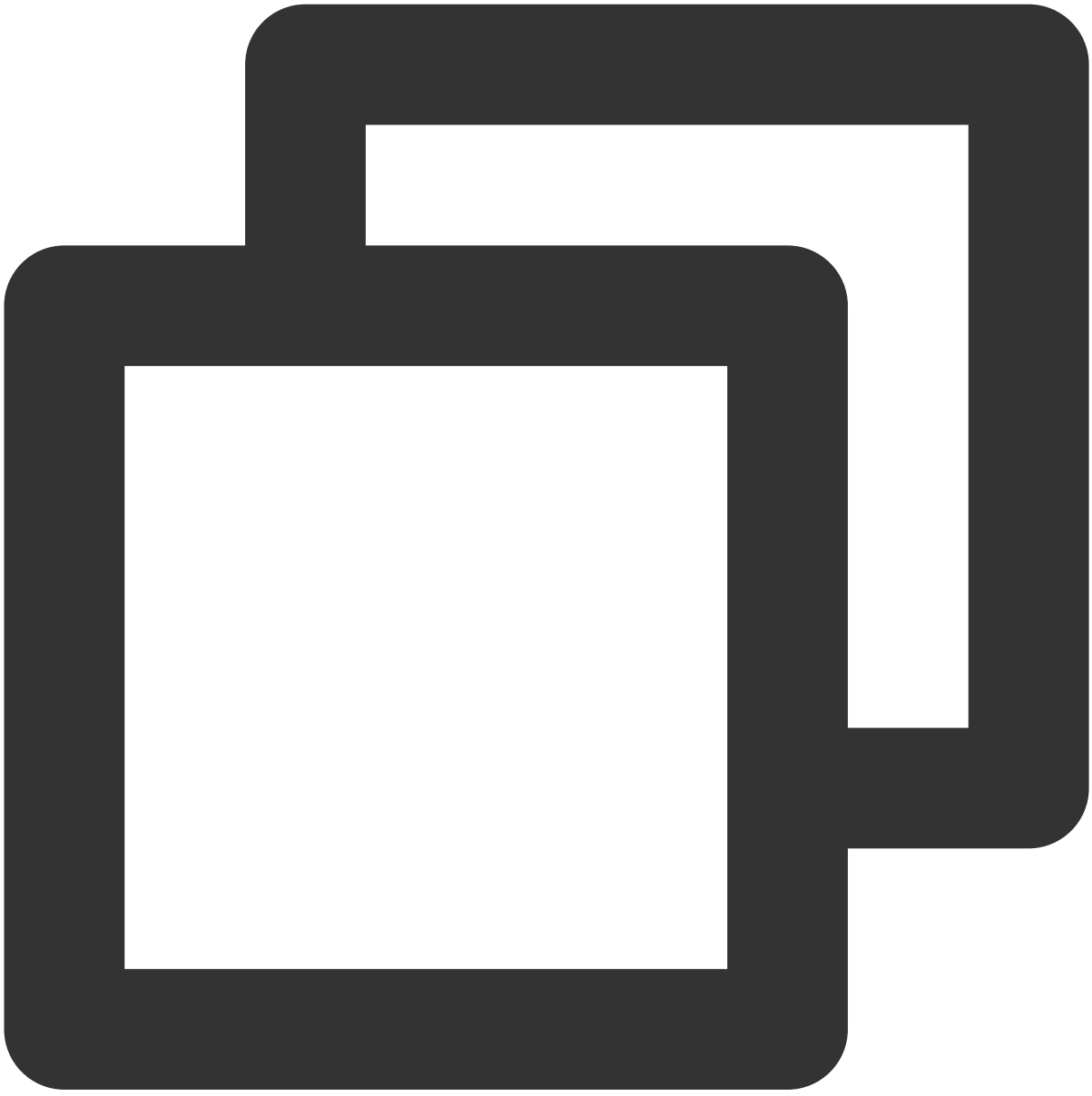
在某些场景下，例如使用了 Windows Gpg4win 或其他 macOS 版本，使用 `gpg --gen-key` 命令生成密钥对。该命令为交互式命令，需要根据提示选择算法类型，指定密钥的有效期，输入您的真实姓名和电子邮件，设置密码等。<

密钥类型：选择使用的密钥类型，或按 `Enter` 键选择默认的 RSA 和 RSA。

椭圆曲线：按 `Enter` 键选择默认的椭圆曲线 `Curve 25519` 。

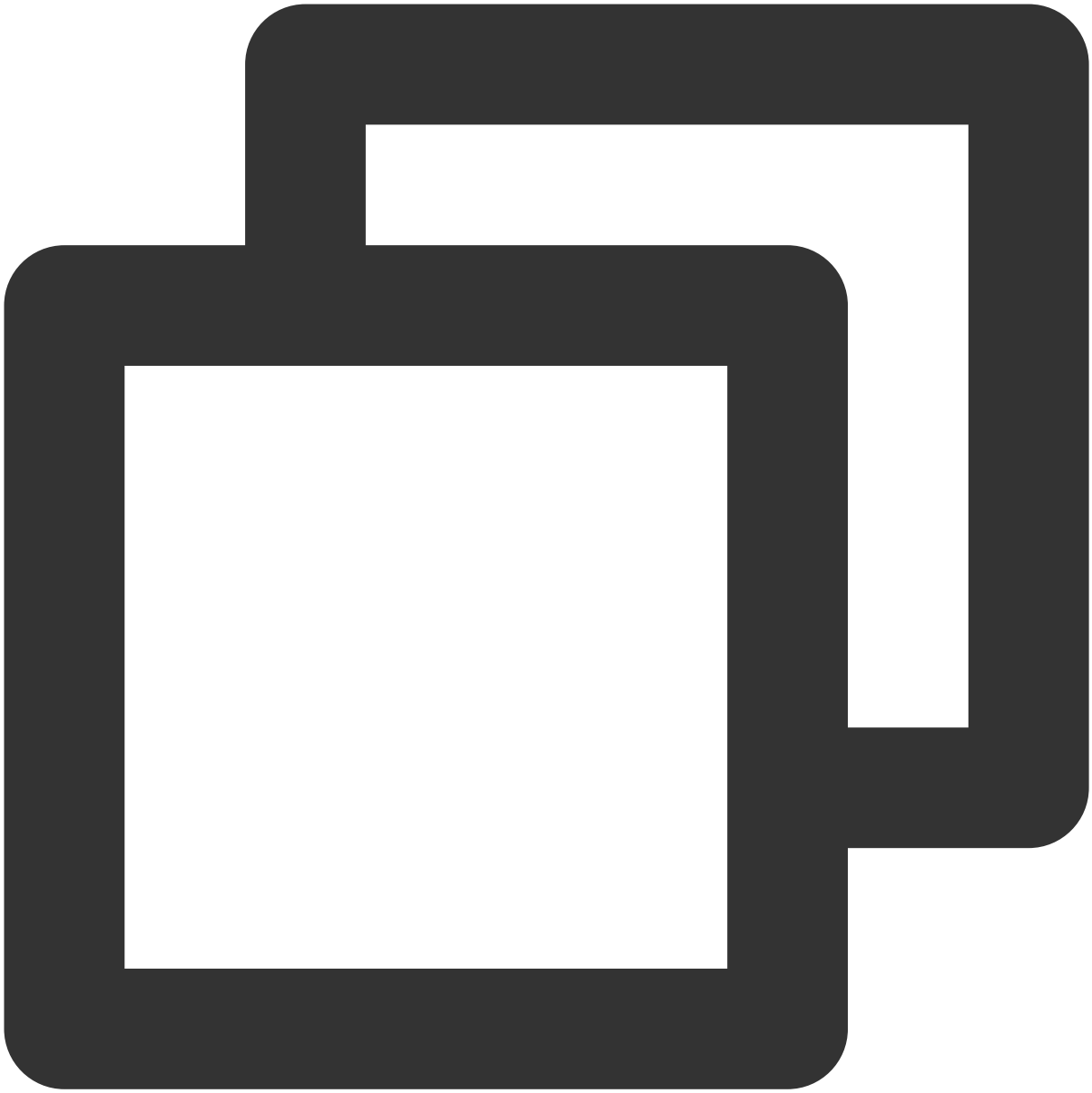
有效期限：按需指定密钥有效期，或按 `Enter` 键选择默认的`永不过期`。
电子邮件地址：需为 CODING 账户内配置的邮箱地址。

3. 运行以下命令列出已创建的 GPG 密钥（命令中的邮箱地址需填写步骤 3 中指定的邮件地址）：



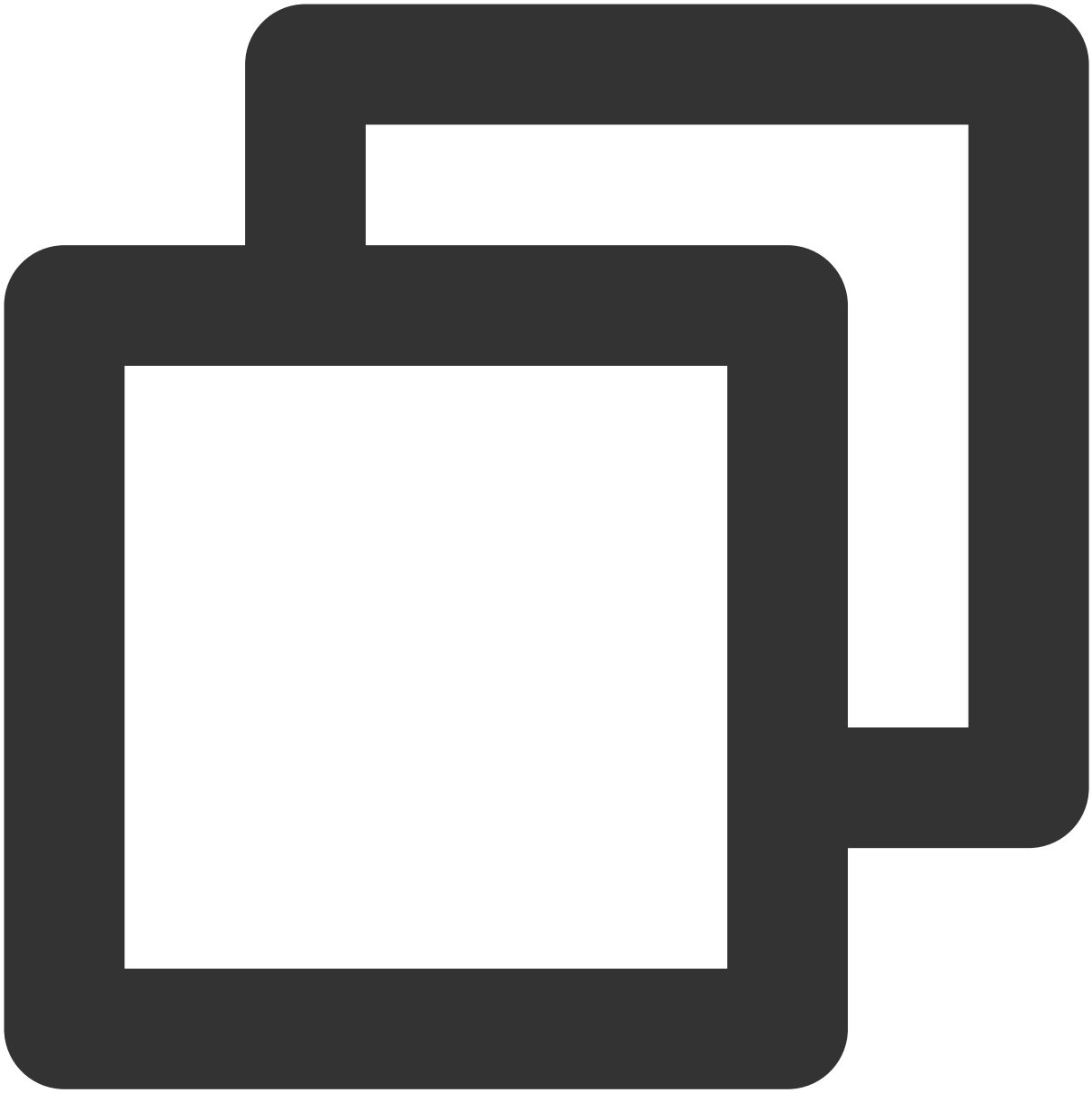
```
gpg --list-secret-keys --keyid-format LONG "your_email"
```

4.复制以 sec开头的 GPG 密钥 ID。以下示例中，复制4AEA00A342C24CA3：



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

5.利用复制的 ID 导出该 ID 的公钥（以上述 ID 为例）：



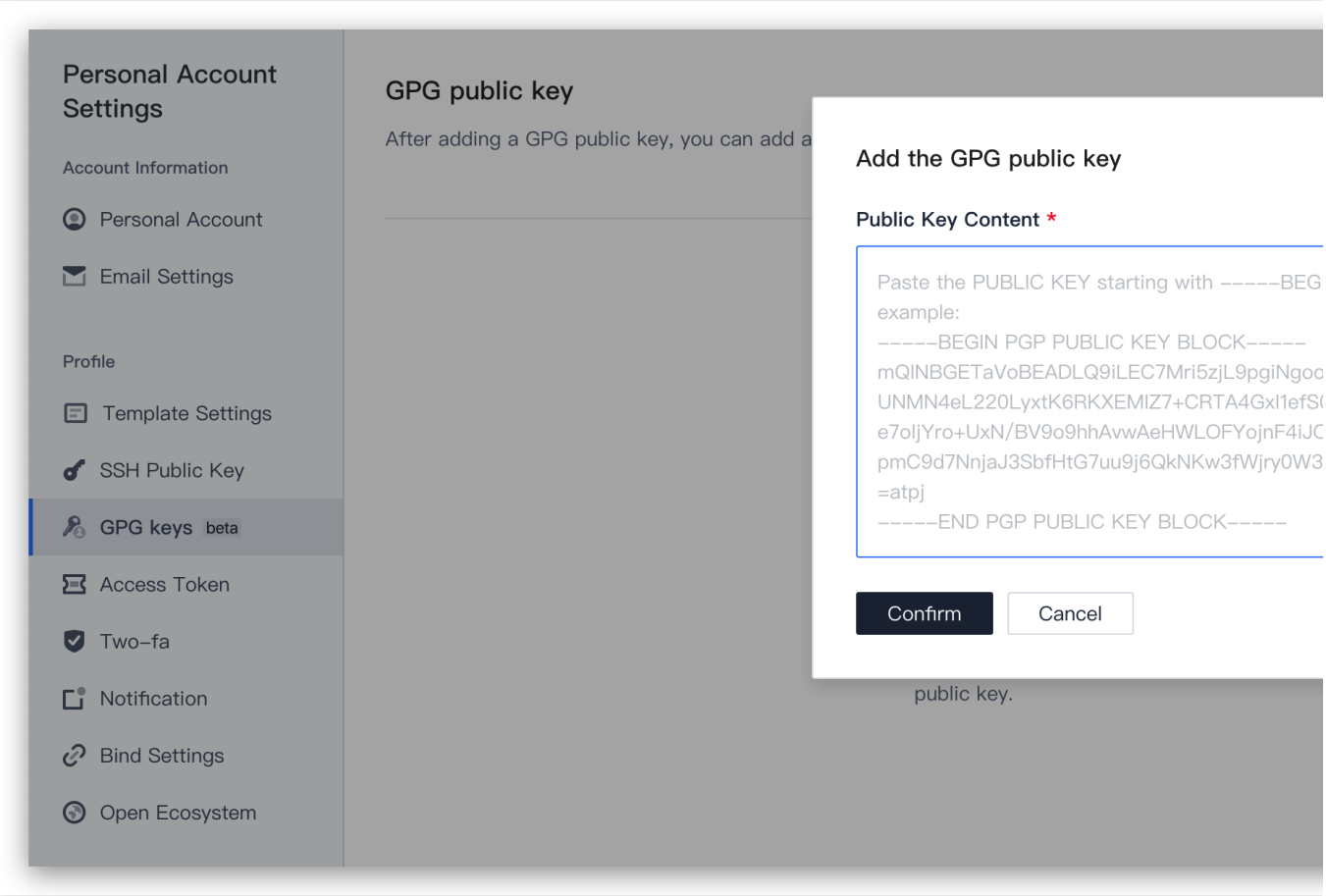
```
gpg --armor --export 4AEA00A342C24CA3
```

6.生成公钥之后，可将其添加至您的 CODING 账户。

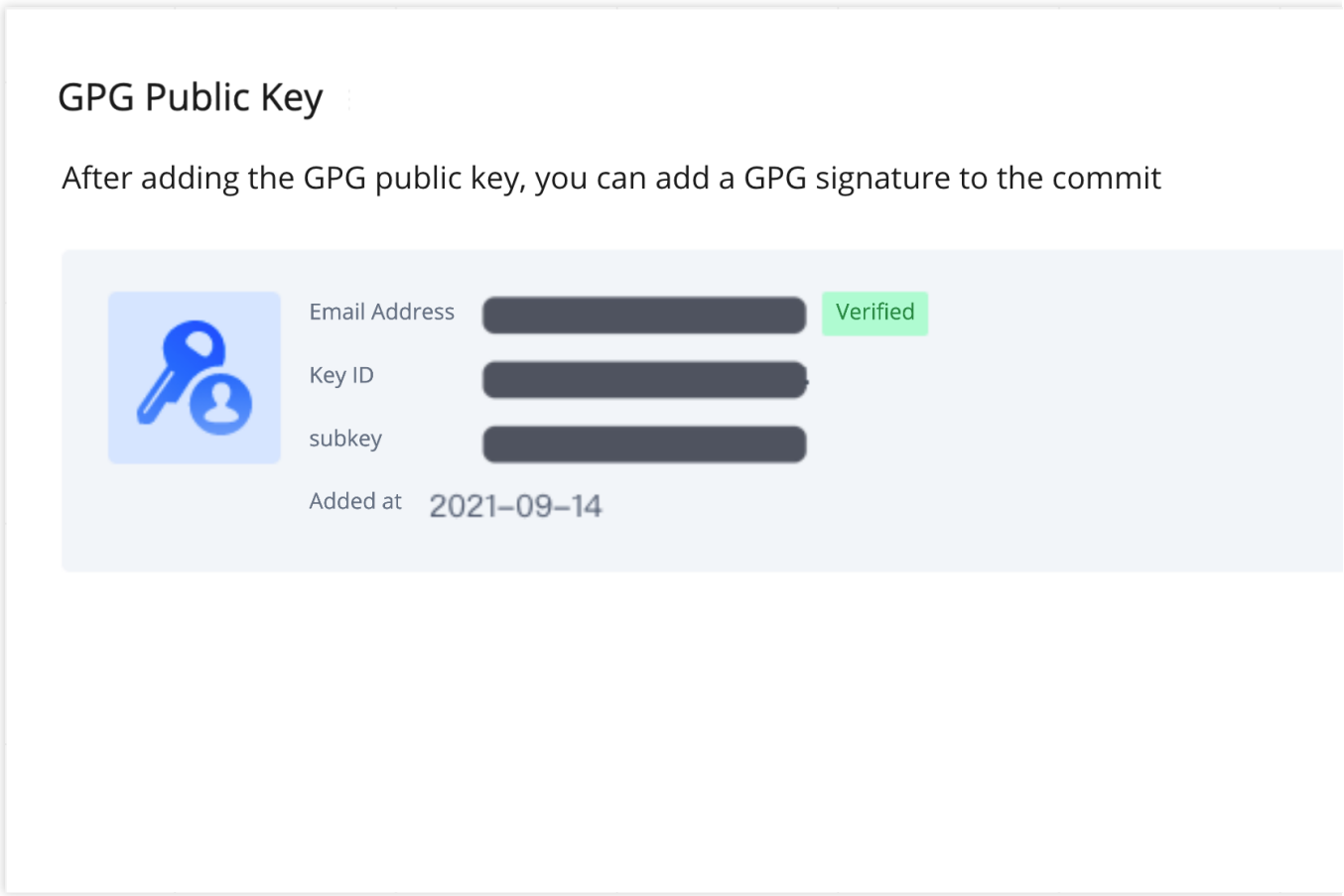
步骤2：添加公钥至个人账户设置

1. 登录 CODING 之后，单击页面右上角个人头像，选择**个人账户设置**。

2. 在左侧导航栏选择**GPG 公钥**，进入公钥管理页面。
3. 单击**新增公钥**，将导出的 GPG 公钥粘贴至内容框，完成确认。



公钥成功添加之后，将会显示邮箱地址的验证状态、密钥 ID 和子密钥。

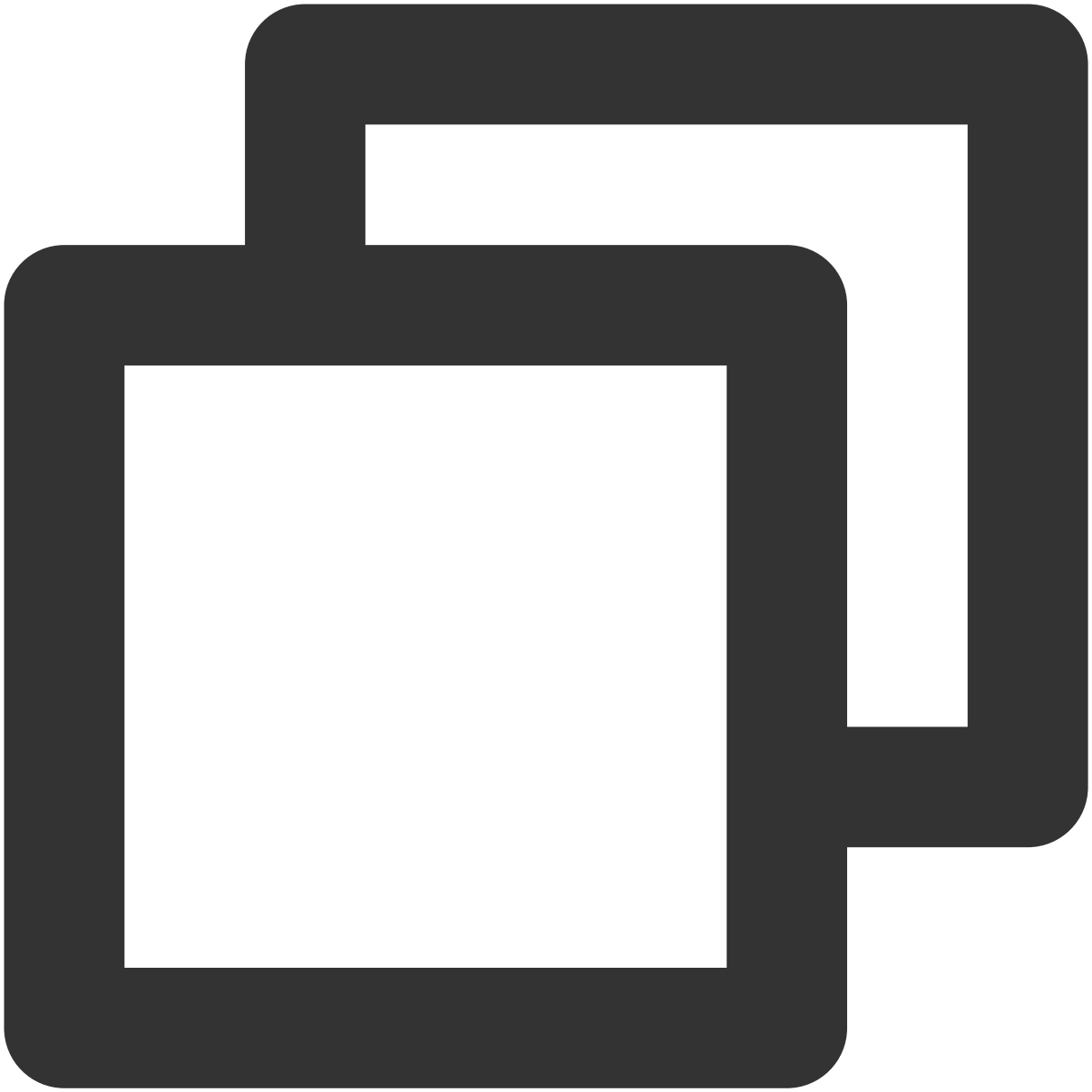


说明：

若邮箱地址显示未验证状态，意味着该邮箱没有在 CODING 账户中配置。请在**个人账户设置 > 邮箱设置**中添加该邮箱。

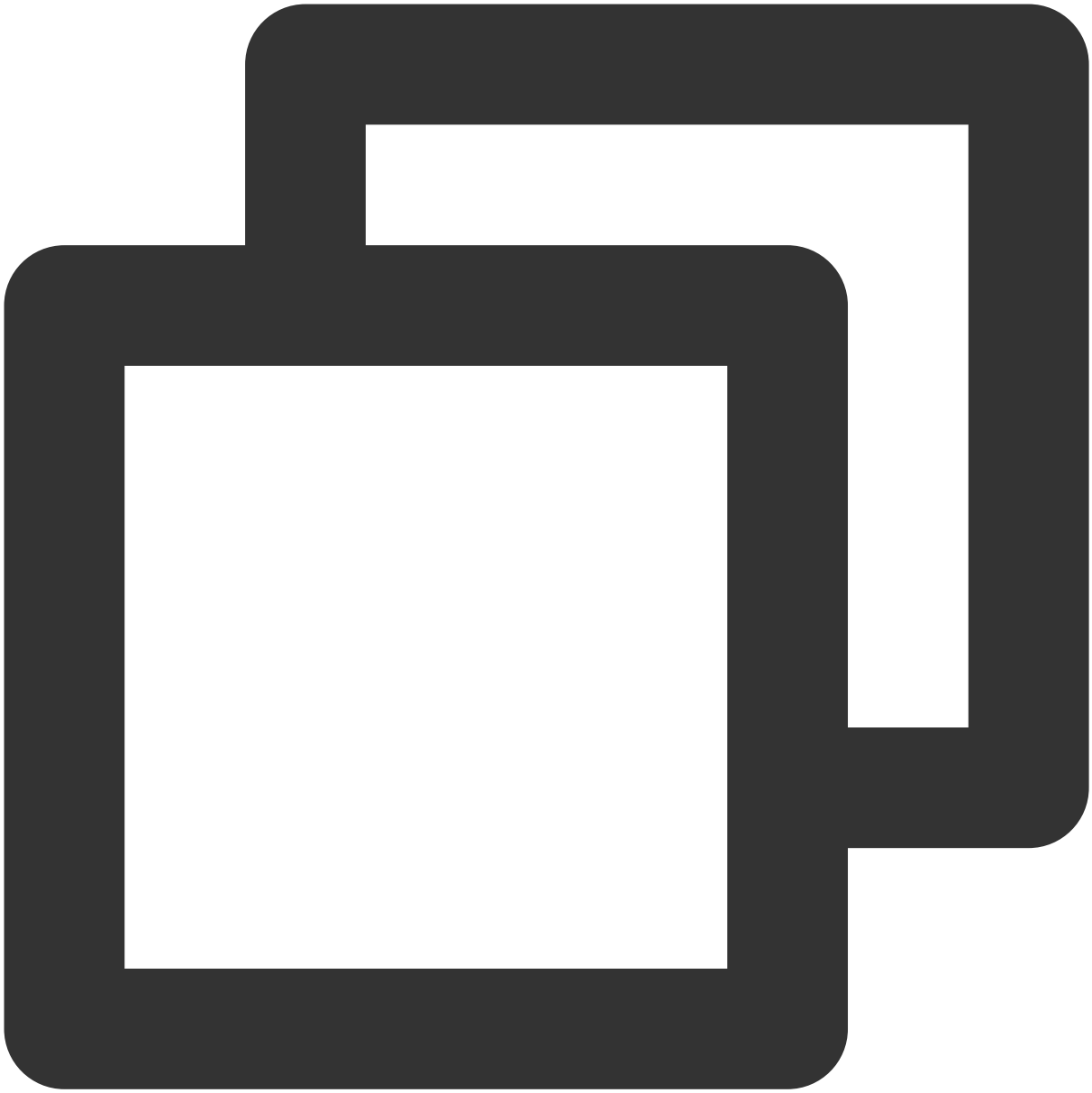
步骤3：与本地 Git 仓库关联

1. 运行以下命令列出您已创建的 GPG 密钥（命令中的邮箱地址需填写生成密钥时指定的邮件地址）：



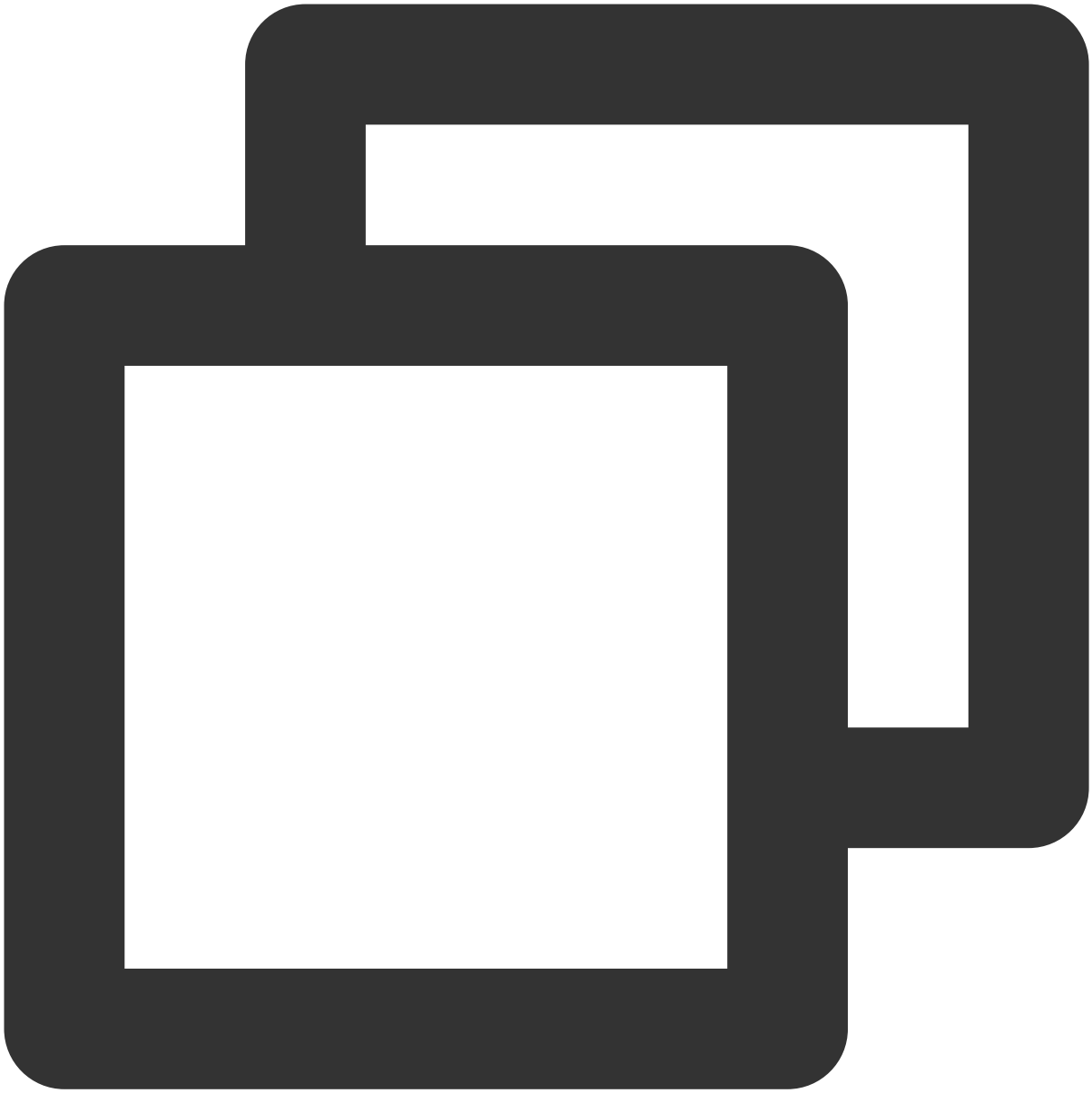
```
gpg --list-secret-keys --keyid-format LONG "your_email"
```

2. 复制 `sec` 开头的 GPG 密钥 ID。以下示例中，复制 `4AEA00A342C24CA3` ：



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

3. 在本地 Git 仓库中配置该密钥，对 commit 提交进行签名：



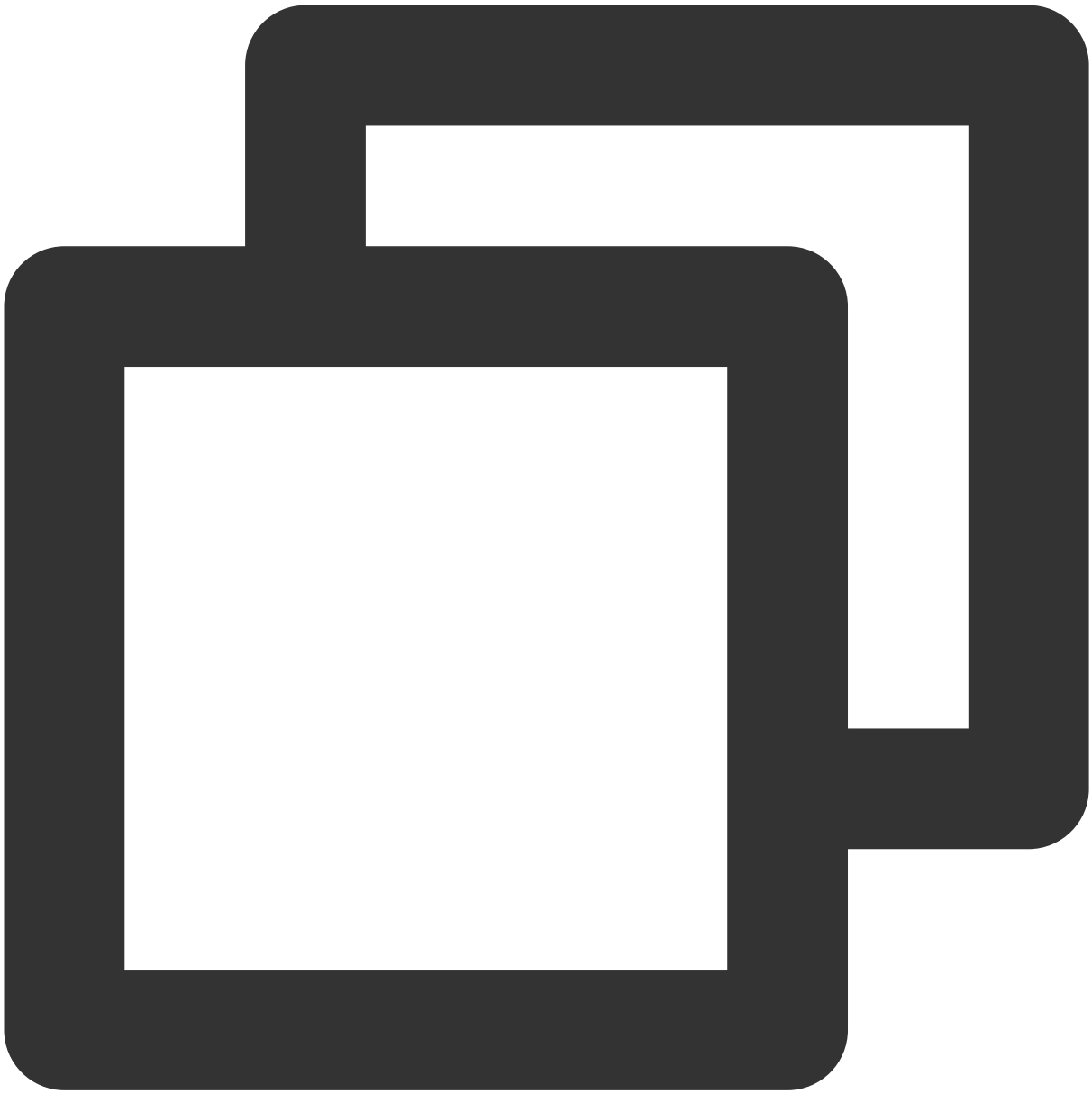
```
git config --global user.signingkey 4AEA00A342C24CA3
```

至此，您已经成功将创建的 GPG 密钥与本地 Git 仓库进行关联。在本地修改完代码后书写 Git commit message 时进行签名，以此验明提交者的真实性。

步骤4：签名 Git commit

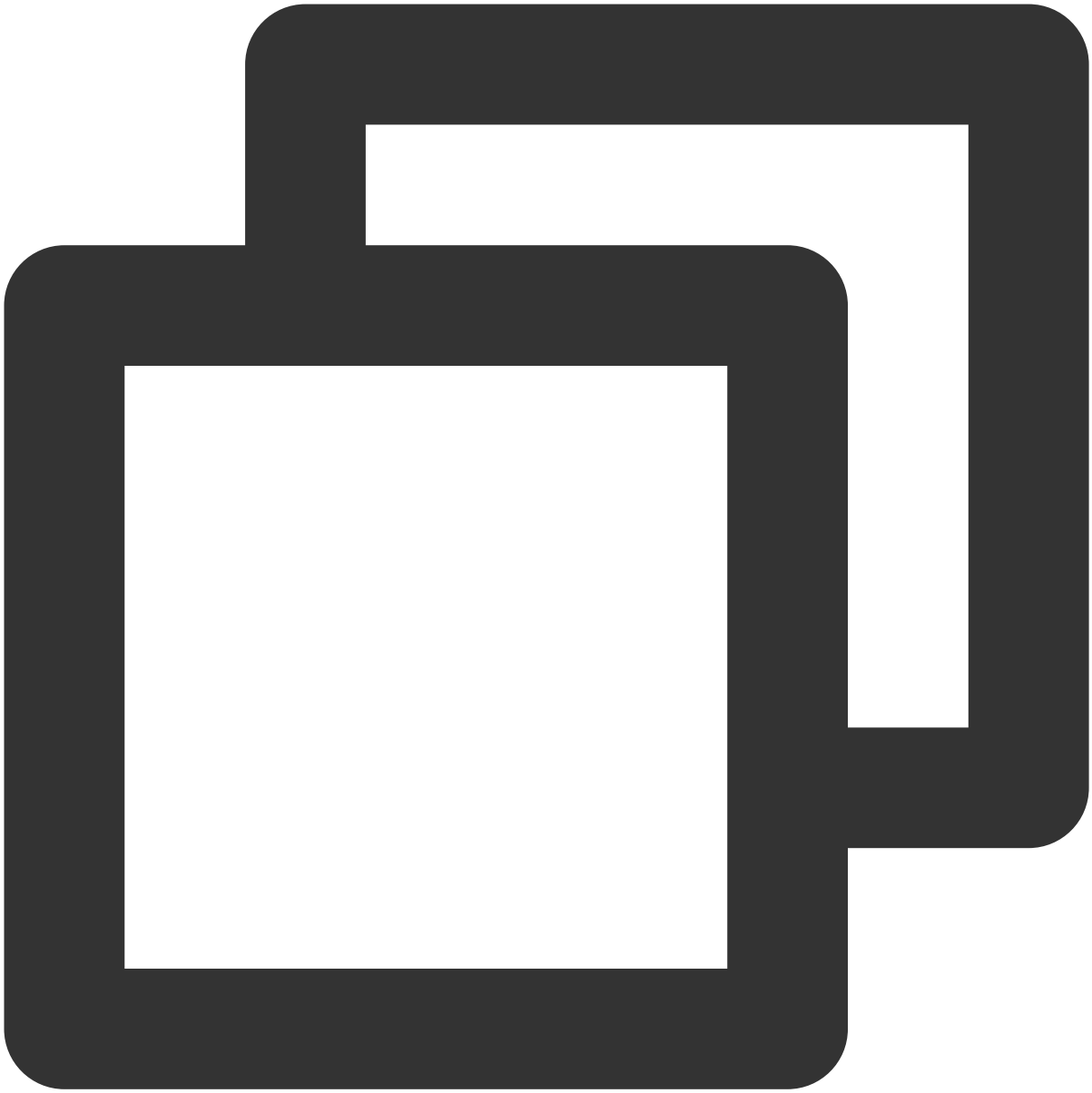
运行 Git commit 命令时需要用到 `-S` 参数。

1. 在本地完成代码编辑需要提交更改时，将 `-S` 参数添加到 `git commit` 命令中：



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

如果不希望每次都要输入 `-S` 标志，您可以使用以下命令行设置 Git 自动为 commit 签名：

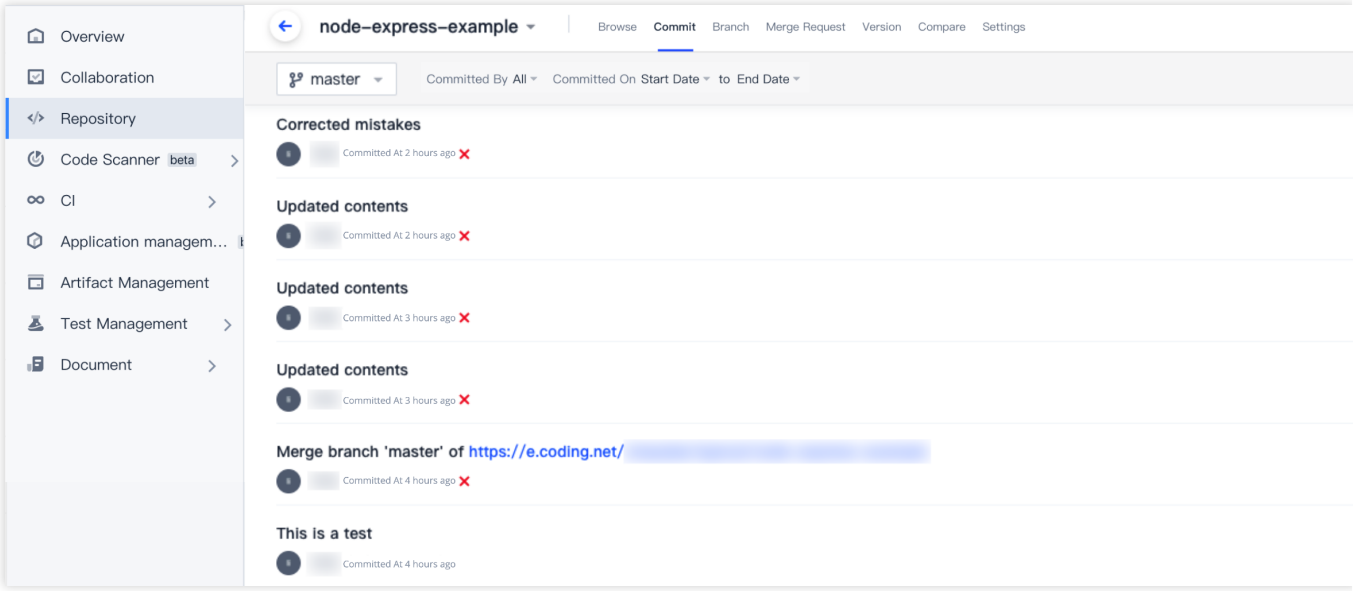


```
git config --global commit.gpgsign true
```

2. 如提示输入密码，则提供生成 GPG 密钥时设置的密码。

步骤5：验证签名

将签了名的提交推送至 CODING 代码仓库后，您可以在在代码仓库的**提交**tab 页查看提交验证是否签名成功。

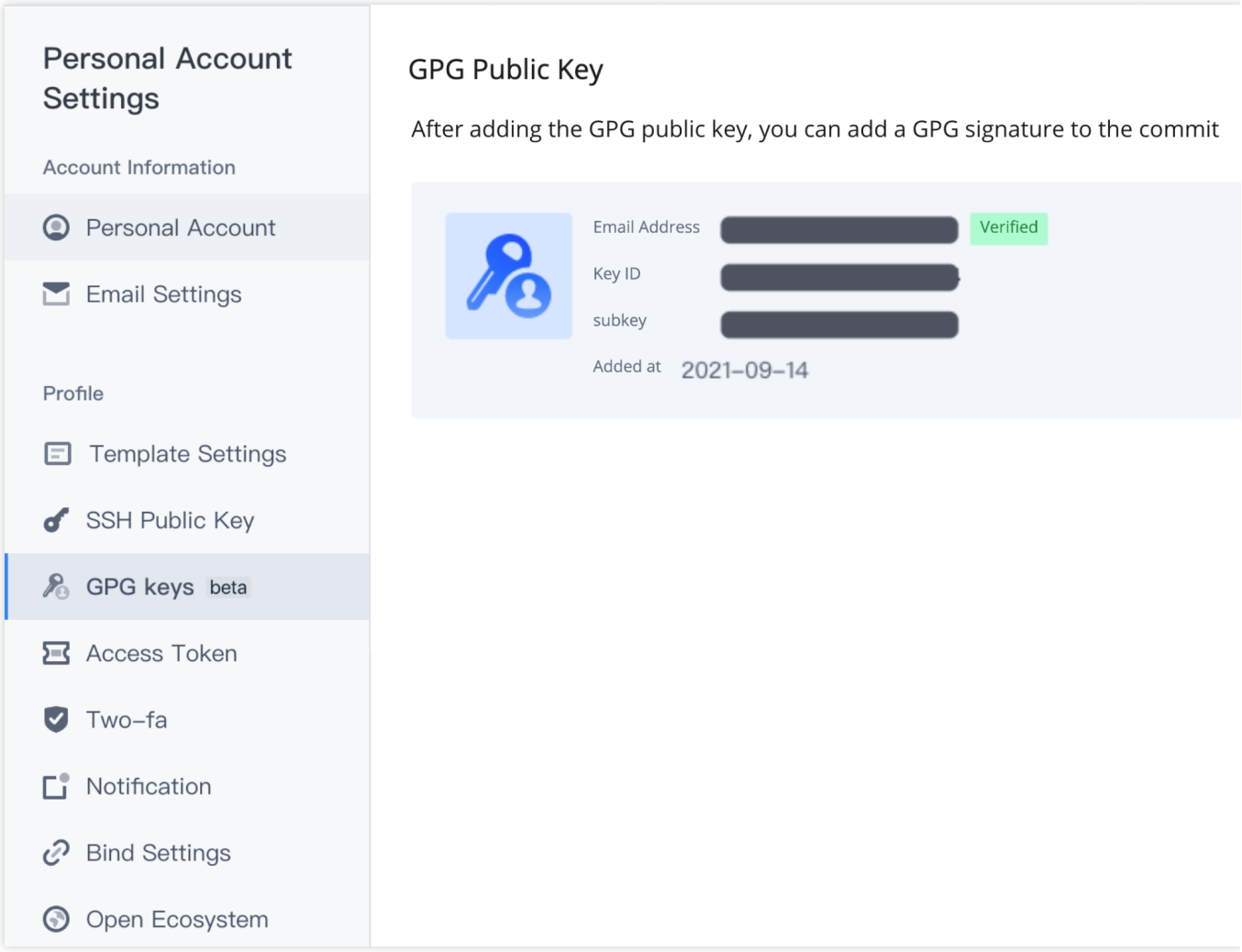


提交验证状态的说明如下：

验证状态	说明
已验证	使用 GPG 私钥签名，CODING 账户中有对应公钥，且公钥邮箱已验证
未验证	使用 GPG 私钥签名，但 CODING 账户中无对应公钥或公钥邮箱未验证 （若出现未验证的邮箱，请前往**个人账户设置 > 邮箱设置**中添加该邮箱。）
无验证状态标签	没有使用 GPG 私钥签名

删除 GPG 公钥

如果您的 GPG 公钥有泄露风险或已不再使用 GPG 签名，可在**个人账户设置 > GPG 公钥**中删除该公钥。

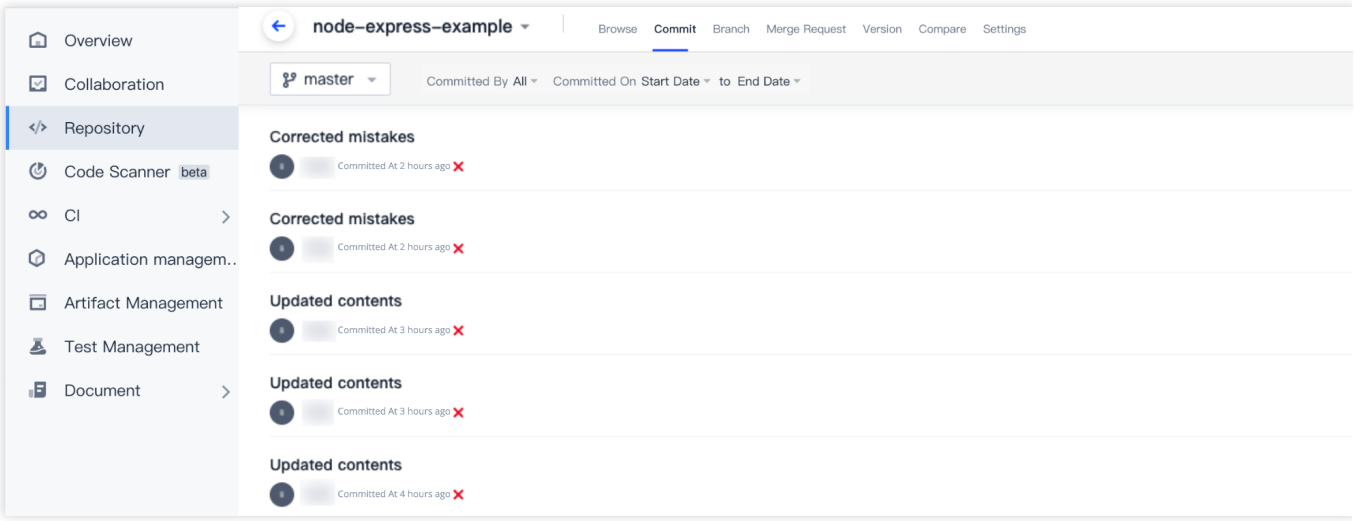


公钥删除之后：

已验证的提交变成未验证状态。

仍使用 GPG 私钥签名的提交（即使用 `git commit -S -m` ）将变成未验证状态。

无签名的提交（即使用 `git commit -m` ）将不被验证，无验证状态标签。

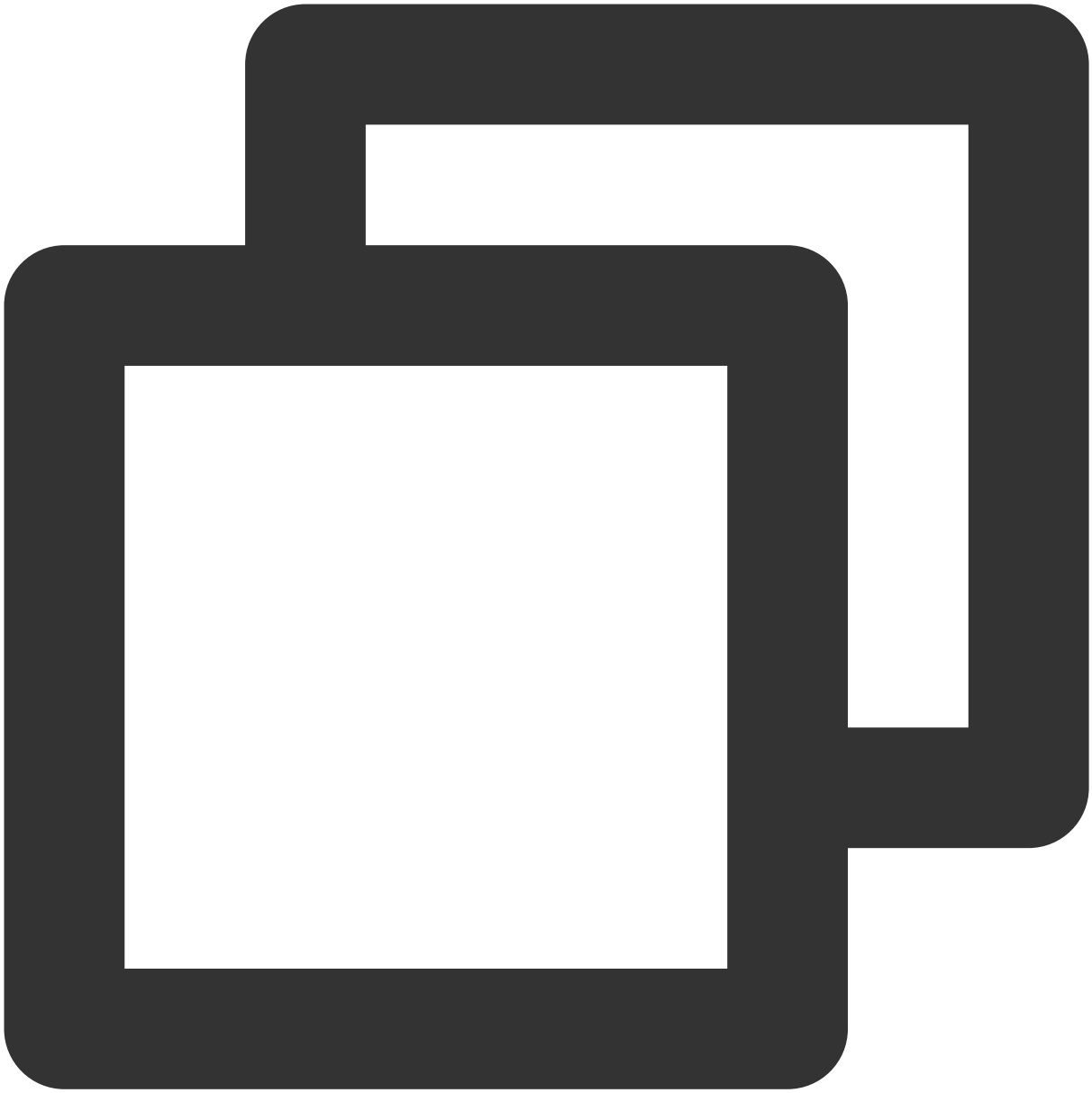


说明：

若已配置 Git 自动签名，可运行 `git config --global commit.gpgsign false` 命令取消自动签名。否则 GPG 公钥删除后，推送至远端仓库的提交依然显示**未验证**状态。

GPG 签名提交报错处理

如果参见上文完成所有操作之后，在使用 `git commit -S -m` 签名提交时出现以下报错，可以参见 解决 GPG 签名失败的问题 修改相关配置。



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

开启提交者及提交注释验证

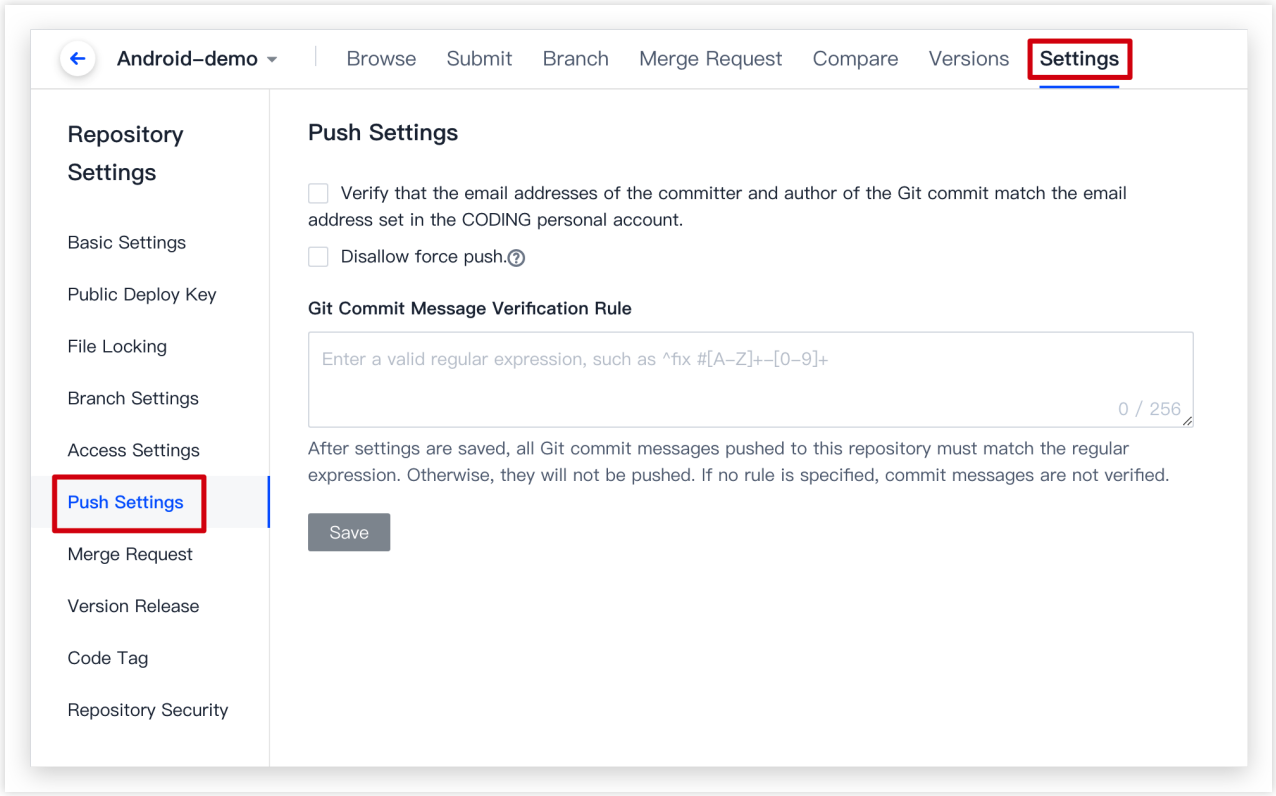
最近更新时间：2023-12-25 17:08:18

本文为您详细介绍如何在代码仓库中开启提交者及提交注释验证。

进入项目

1. 登录 CODING 控制台，单击团队域名进入 CODING 使用页面。
2. 单击页面右上角的

，进入项目列表页面，单击项目图标进入目标项目。
项目管理员可以在代码仓库的**设置 > 推送设置**页面开启针对 Git 提交者 (Committer) 和 提交作者 (Author）的验证。



项目管理员还可以设置 Git 提交注释的规则，不符合该规则的提交将被拦截。例如，将提交注释规则设置为 `^fix #[0-9]+`，则规定了每次提交时必须要在提交注释中关联项目事项；如 `^fix #630` 表明该次提交关联当前项目中 ID 为 630 的事项。

说明：

如需了解如何在提交信息中自动关联事项，请参见 提交文件。

锁定文件与文件夹

最近更新时间：2023-12-25 17:08:18

本文为您详细介绍如何锁定代码仓库中的文件与文件夹。

进入项目

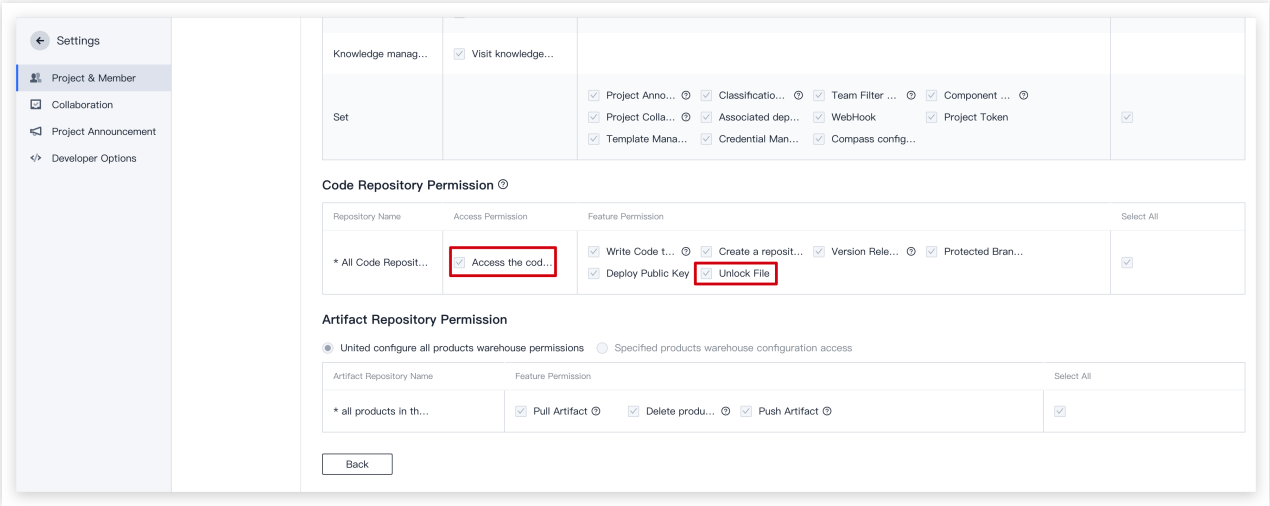
1. 登录 CODING 控制台，单击团队域名进入 CODING 使用页面。
2. 单击页面右上角的

，进入项目列表页面，单击项目图标进入目标项目。

针对代码仓库中的**默认分支（通常为 master 分支）**，CODING 支持对该分支下的文件或路径进行锁定。锁定后的文件仅限锁定者修改（编辑、删除），若选择锁定路径，则该路径下的所有文件同样会被锁定并仅限锁定者修改。

权限说明

项目管理员可进入项目，单击左下角的**项目设置**，前往**项目与成员 > 用户组**设置代码仓库权限。设置完成后将团队成员添加至相关用户组中即可完成项目内的代码仓库权限控制。

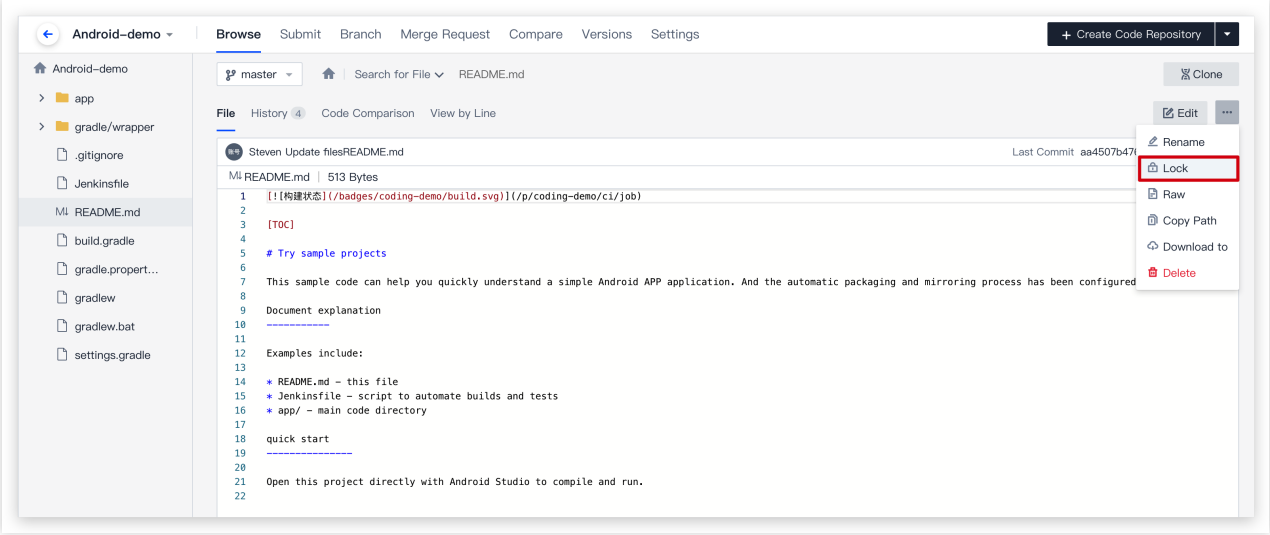


具备**访问代码仓库**权限的项目成员可以在代码的浏览页面锁定相关文件或路径。锁定后，如果希望取消锁定，可以在代码浏览页面进行解锁。

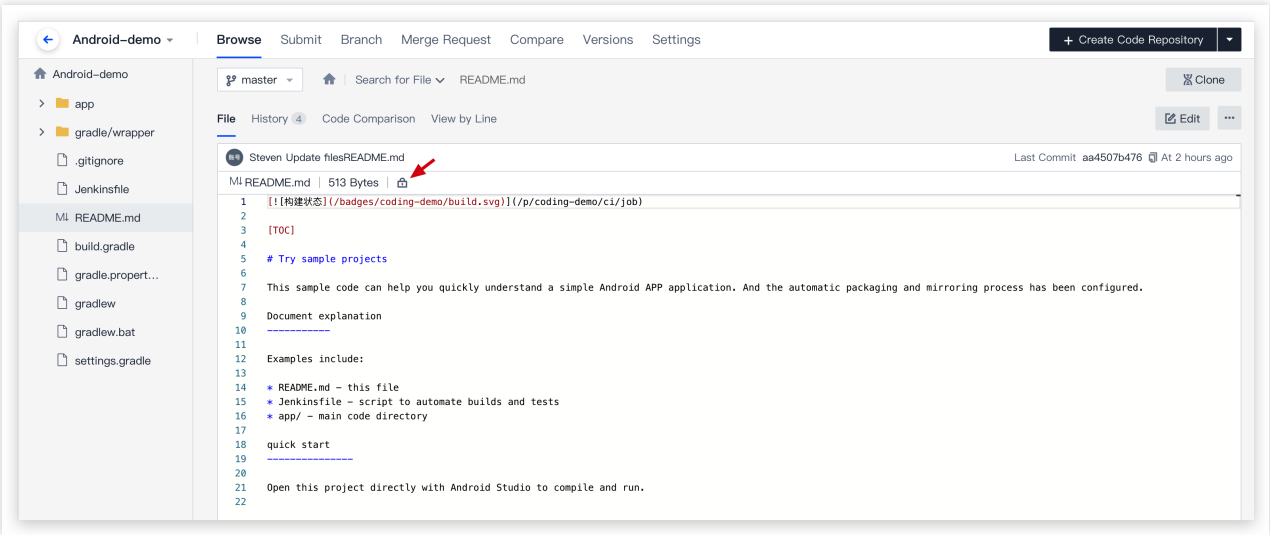
普通用户只能解锁自己锁定的文件或路径；拥有**解锁锁定文件**权限的成员可以解锁他人锁定的文件或路径。

锁定文件

进入任一项目的代码仓库的浏览页，选择拟锁定文件，单击右上角更多操作按钮，选择锁定操作。

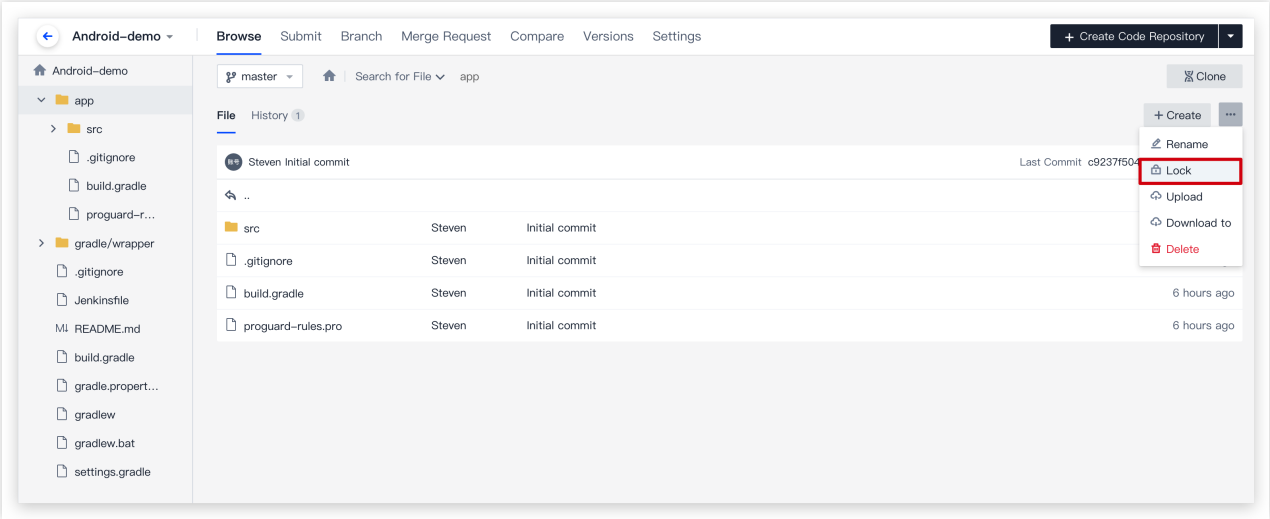


锁定后文件会出现小锁标志，除了锁定者以外的成员将无法编辑或删除此文件。



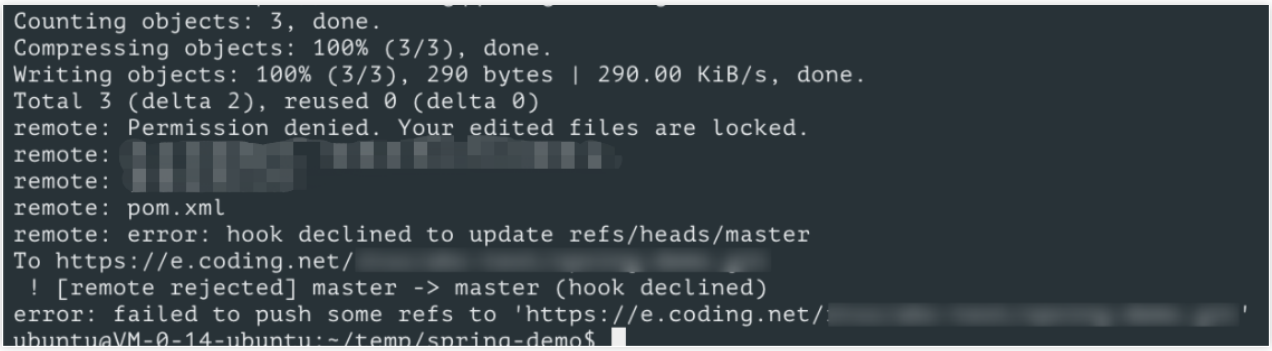
锁定文件夹

选择仓库的任一文件夹，单击右上角更多操作按钮，选择锁定操作，即可锁定该路径下所有文件。

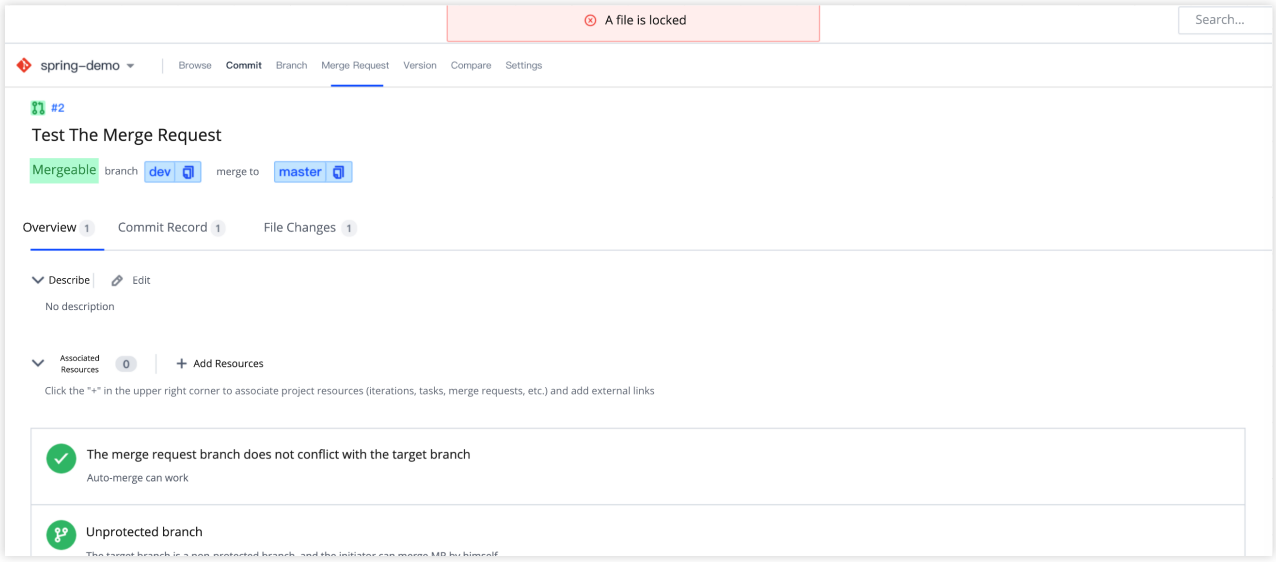


锁定效果

锁定后的文件只有锁定者才能进行编辑/删除操作。锁定后的路径只有锁定者才能在路径下新建/编辑/删除文件。若其他用户推送的代码中包含对锁定文件的操作记录，则会提示推送失败。

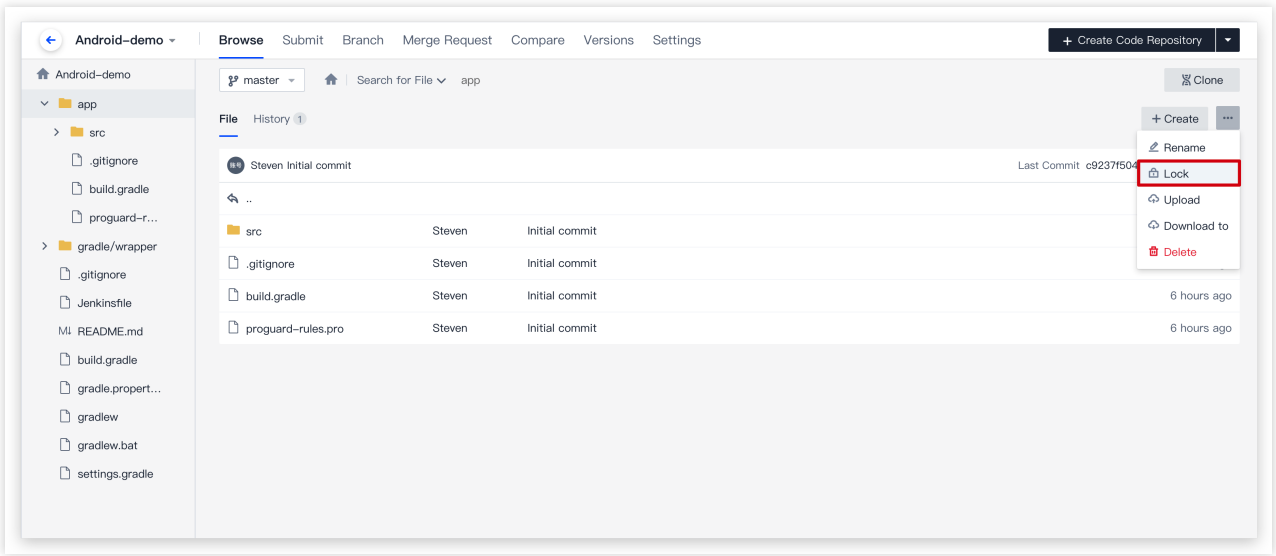


在合并请求中，若目标分支为默认分支且待写入的目录中包含被锁定的文件或路径，那么该合并请求只能由被锁定的文件或路径的锁定者进行合并。其他成员单击合并会提示失败。

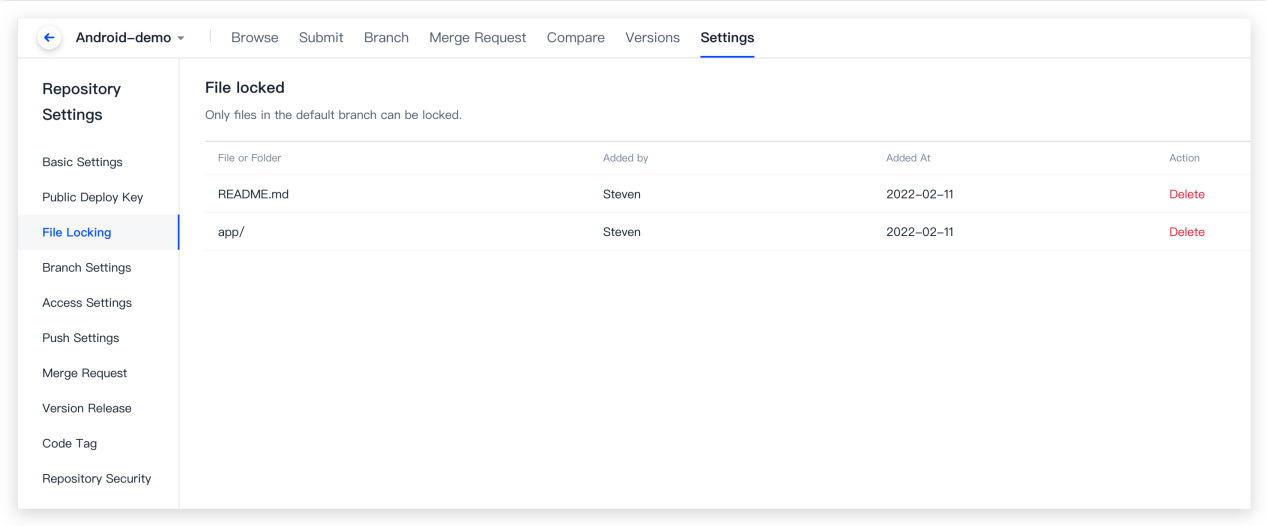


解锁被锁定文件或文件夹

项目普通成员可直接在代码浏览页面解锁自己锁定的文件或文件夹；拥有**解锁锁定文件**权限的成员还可以解锁他人锁定的文件或文件夹。



项目管理员可以在仓库的**设置 > 文件锁定**页面中查看被锁定的文件/文件夹。单击**删除**即可解锁文件或文件夹。

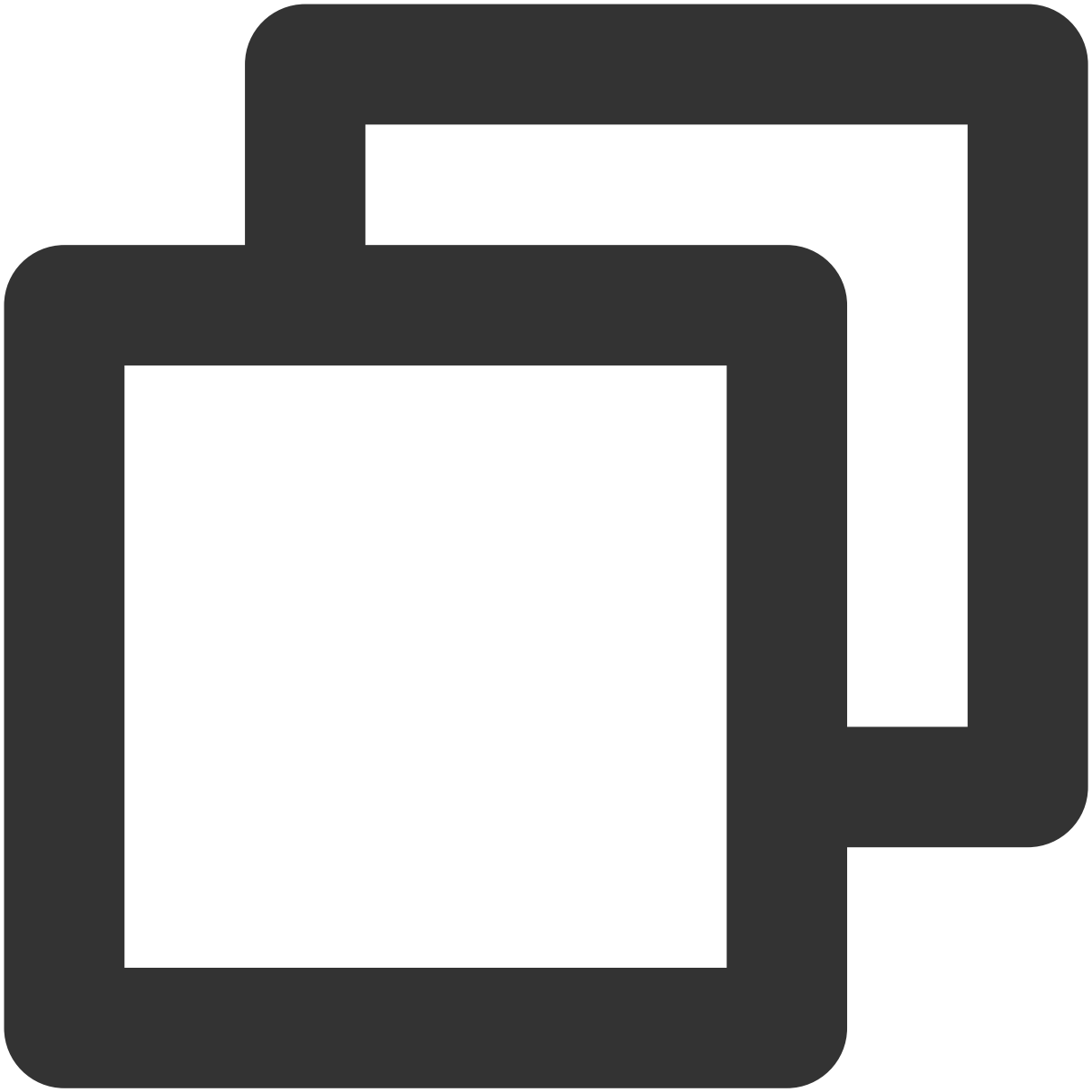

Android-demo
Browse
Submit
Branch
Merge Request
Compare
Versions
Settings
Repository Settings
Basic Settings
Public Deploy Key
File Locking
Branch Settings
Access Settings
Push Settings
Merge Request
Version Release
Code Tag
Repository Security
File locked
Only files in the default branch can be locked.
File or Folder
Added by
Added At
Action
README.md
Steven
2022-02-11
Delete
app/
Steven
2022-02-11
Delete

Git 通识
常用命令

最近更新时间：2023-12-25 17:08:18

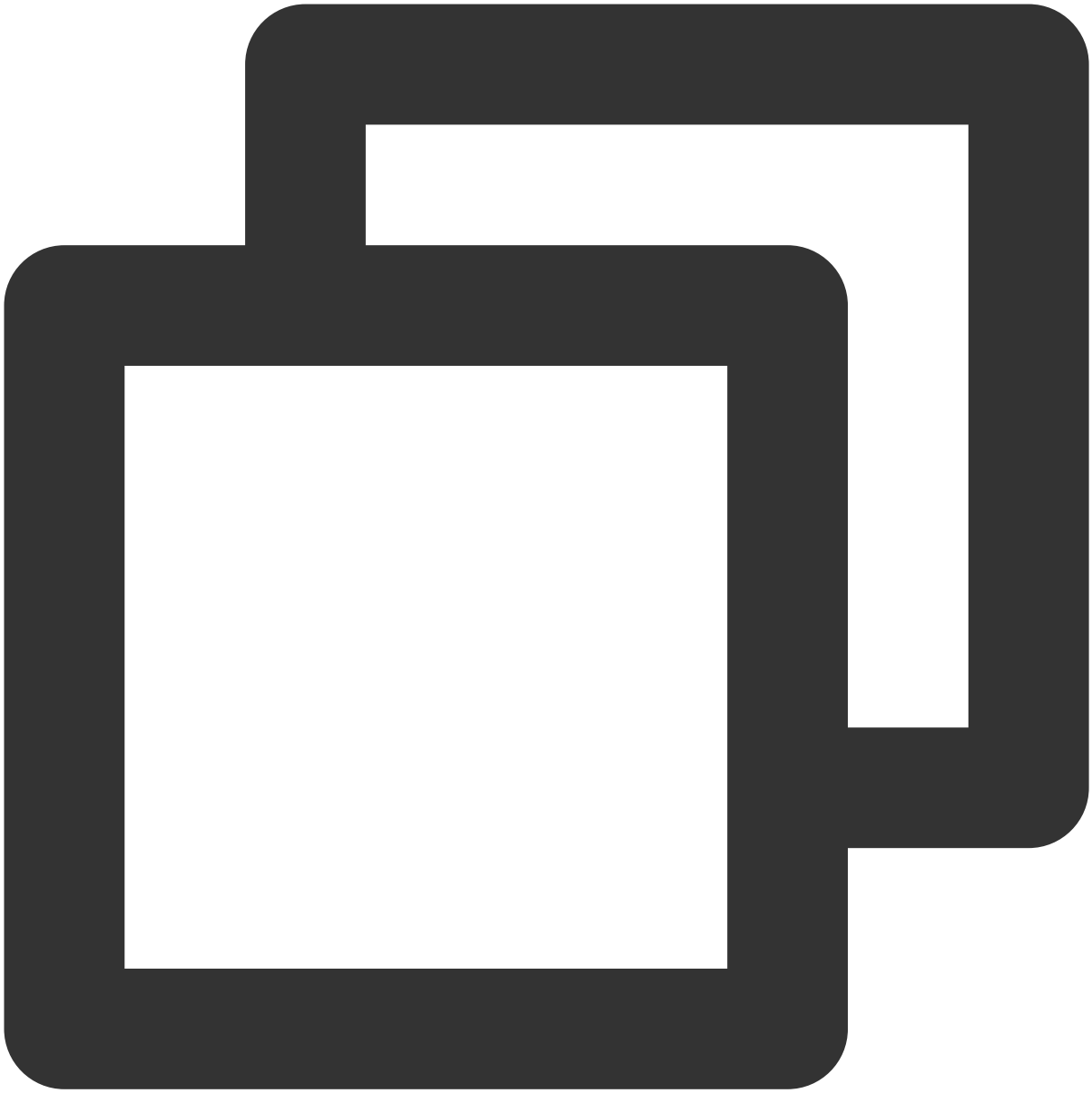
本文为您详细介绍 Git 中的常用命令。

创建版本库



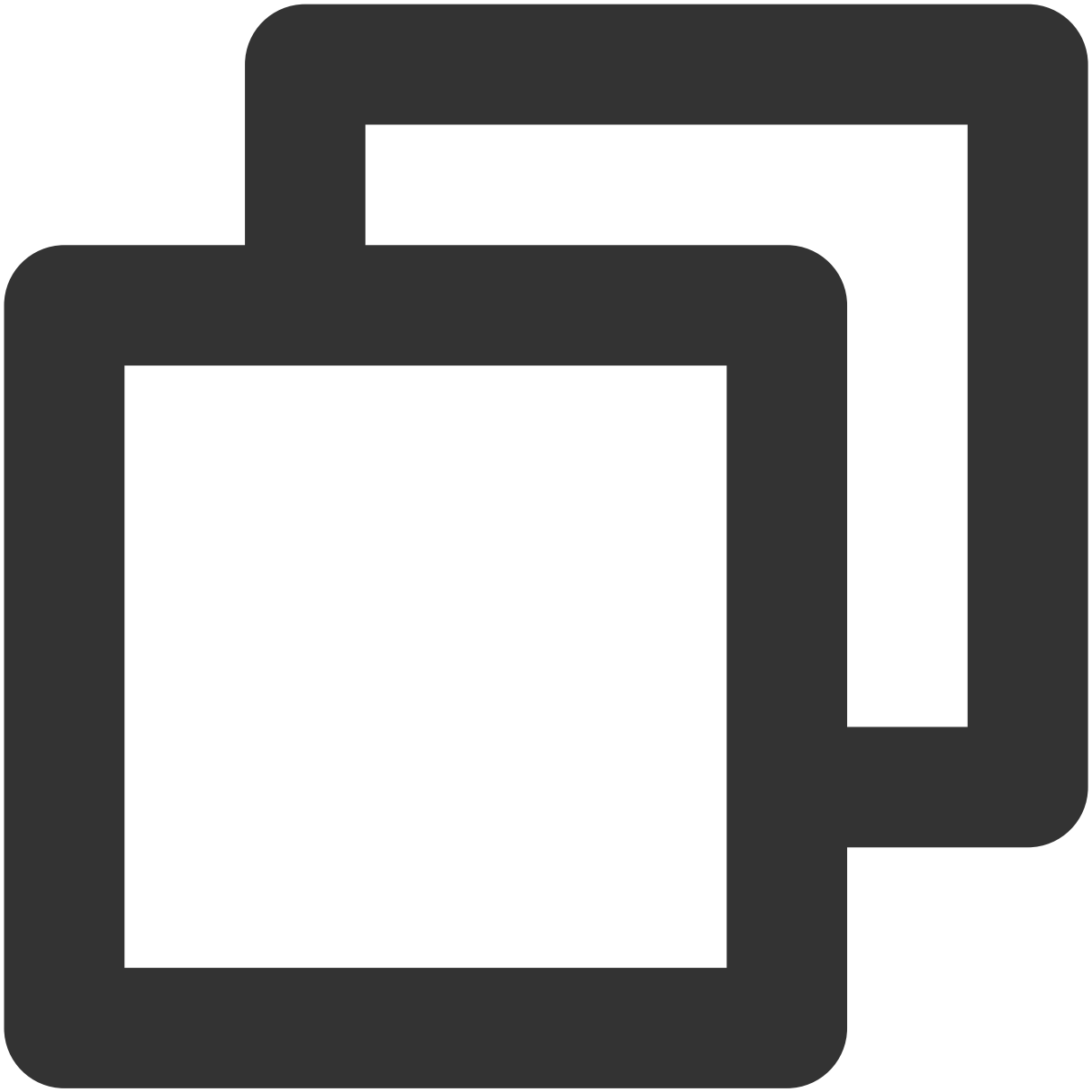
```
$ git clone <url>                    #克隆远程版本库
$ git init                           #初始化本地版本库
```

修改和提交



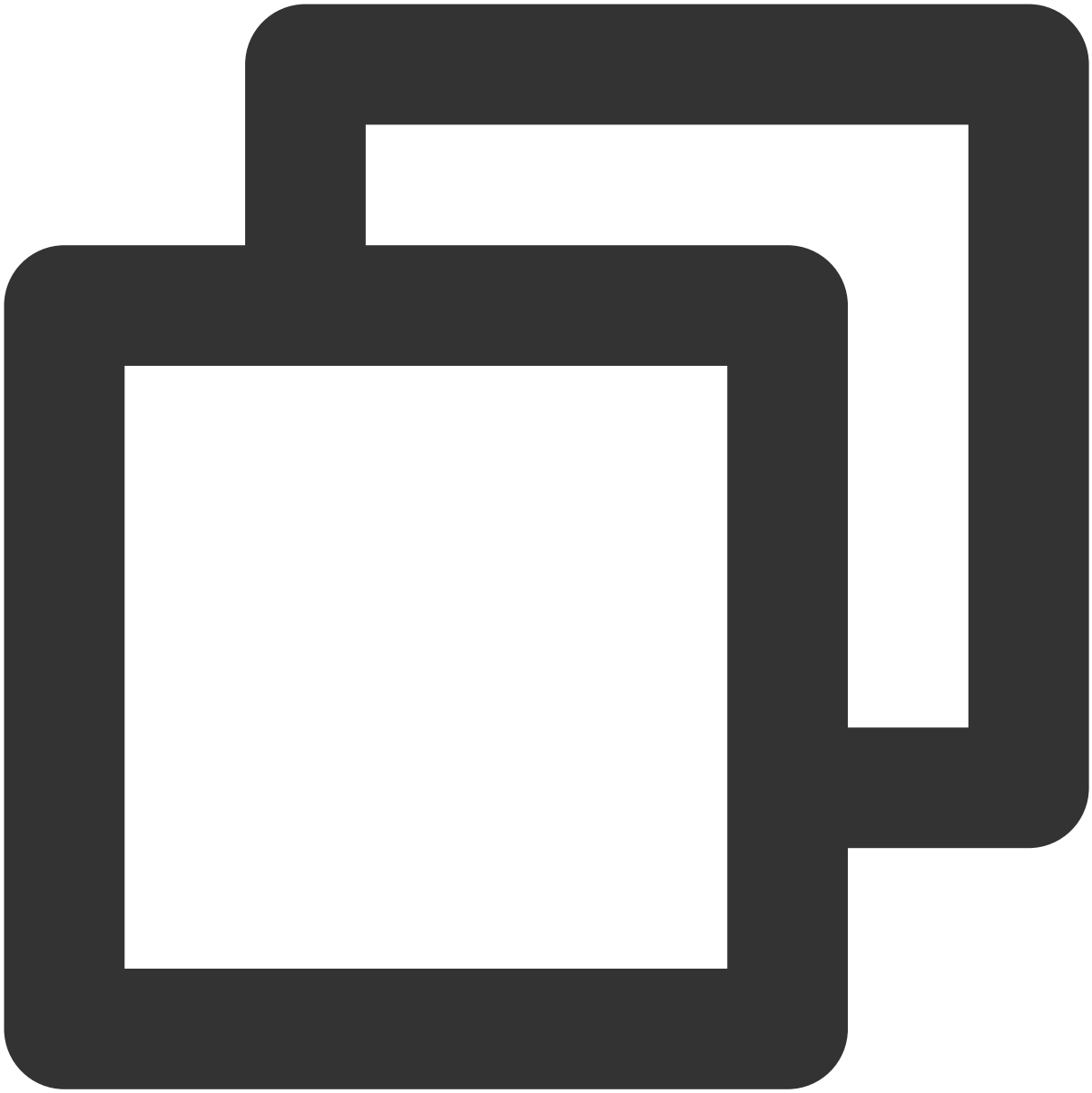
```
$ git status                          #查看状态
$ git diff                            #查看变更内容
$ git add .                           #跟踪所有改动过的文件
$ git add <file>                      #跟踪指定的文件
$ git mv <old><new>                   #文件改名
$ git rm<file>                        #删除文件
$ git rm --cached<file>               #停止跟踪文件但不删除
$ git commit -m "commit messages"     #提交所有更新过的文件
$ git commit --amend                  #修改最后一次改动
```

查看提交历史



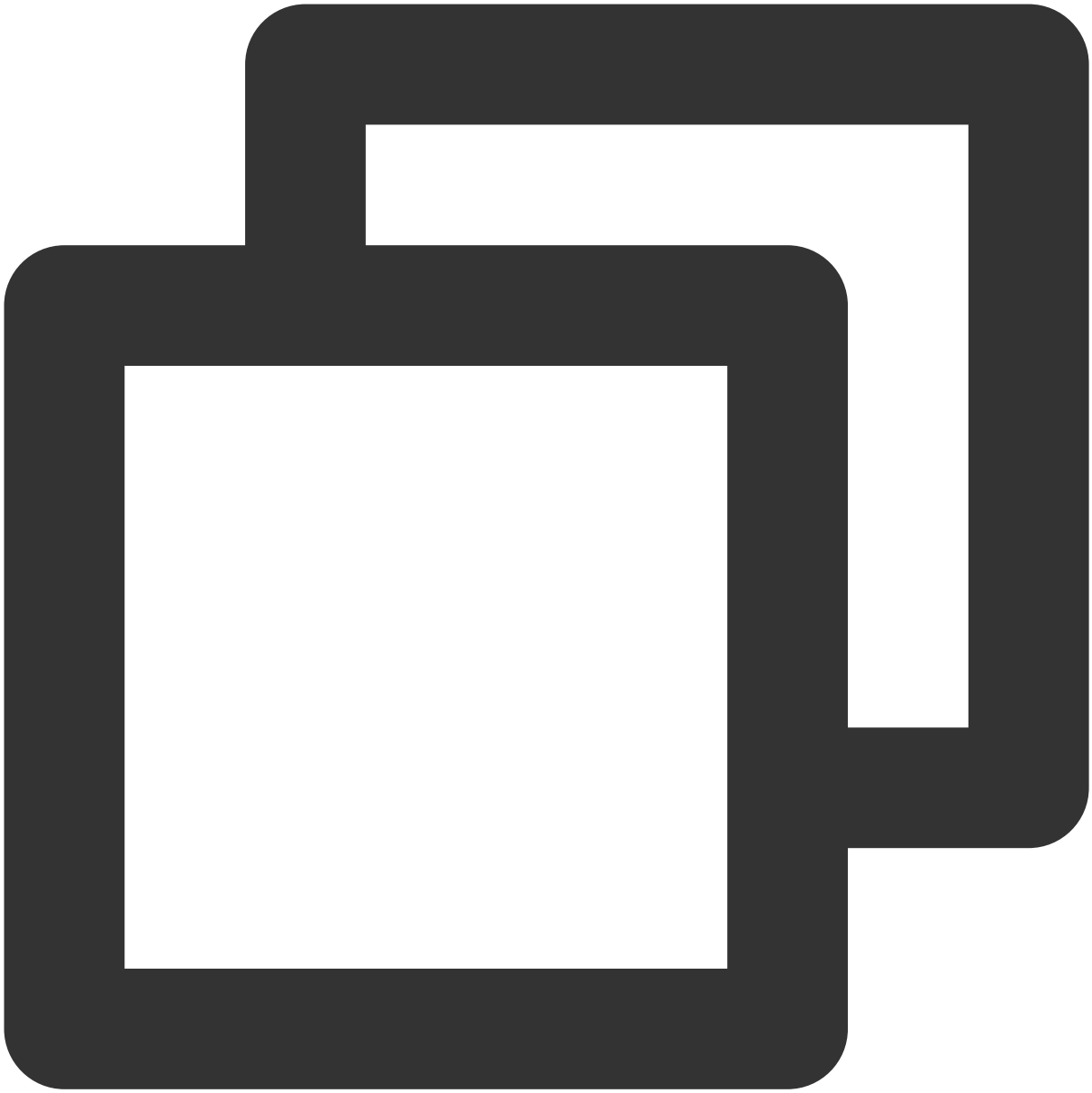
```
$ git log                    #查看提交历史
$ git log -p <file>          #查看指定文件的提交历史
$ git blame <file>           #以列表方式查看指定文件的提交历史
```

撤销



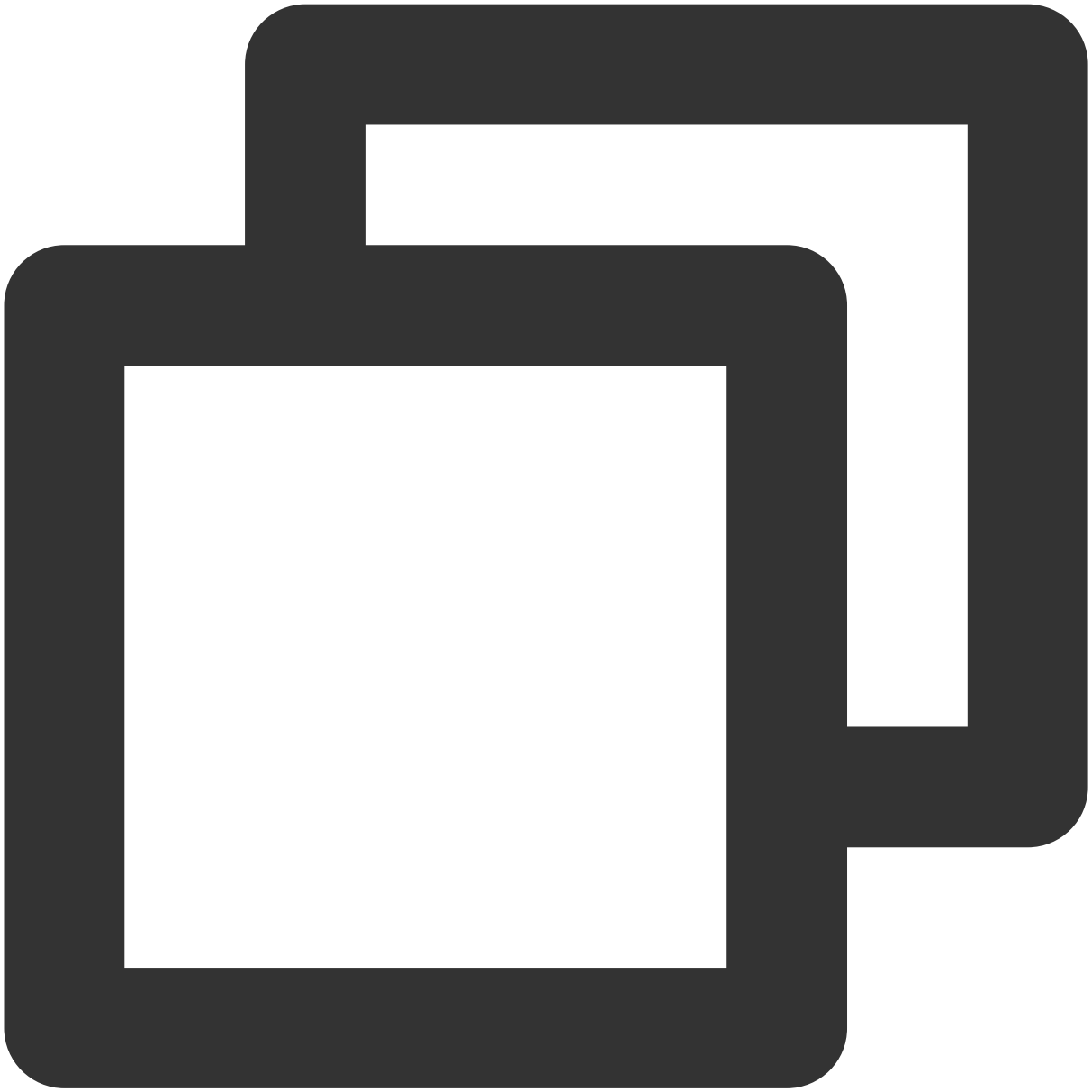
```
$ git reset --hard HEAD       #撤销工作目录中所有未提交文件的修改内容
$ git checkout HEAD <file>    #撤销指定的未提交文件的修改内容
$ git revert <commit>         #撤销指定的提交
$ git log --before="1 days"   #退回到之前1天的版本
```

分支与标签



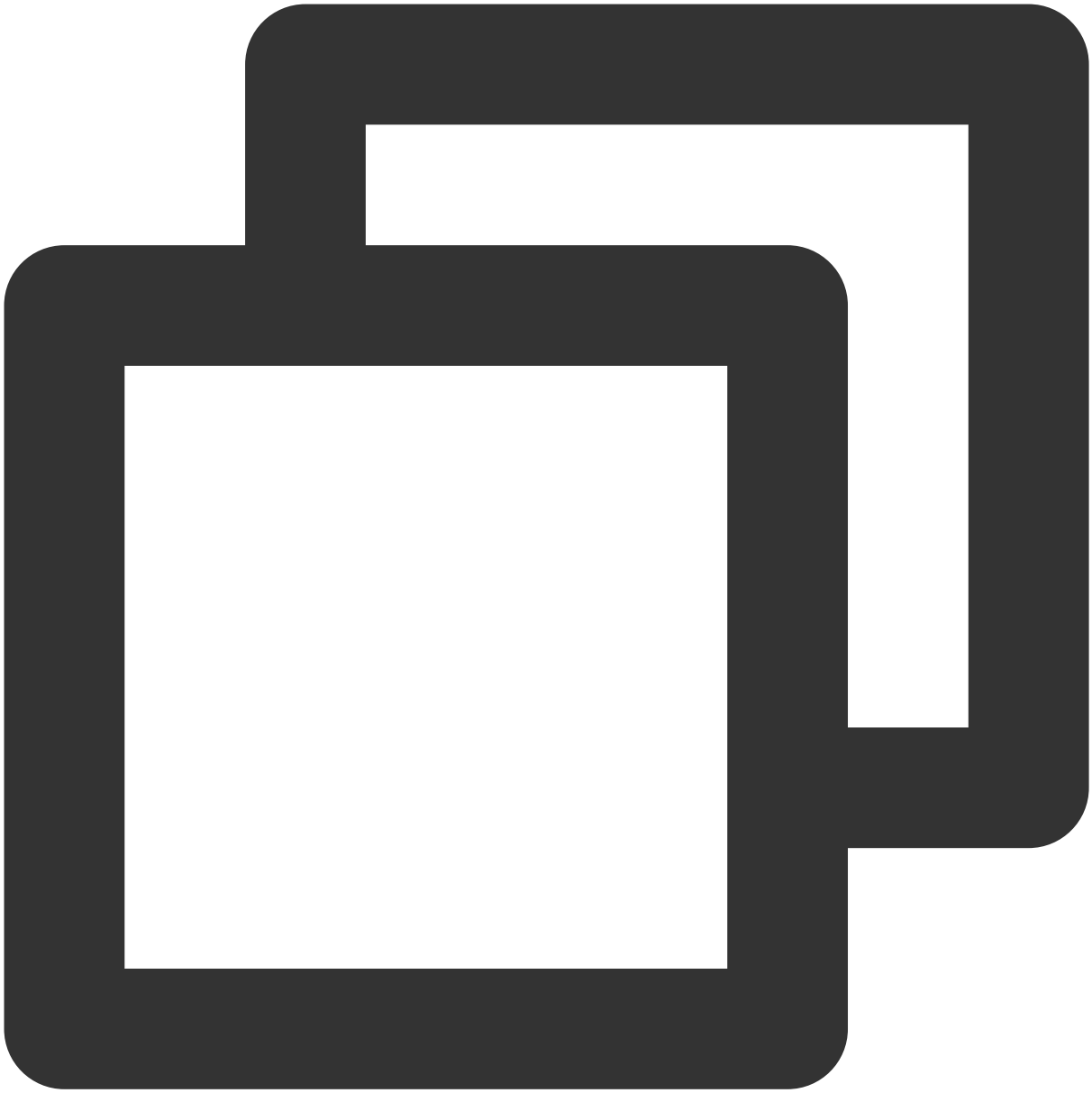
```
$ git branch                    #显示所有本地分支
$ git checkout <branch/tag>     #切换到指定分支和标签
$ git branch <new-branch>       #创建新分支
$ git branch -d <branch>        #删除本地分支
$ git tag                       #列出所有本地标签
$ git tag <tagname>             #基于最新提交创建标签
$ git tag -d <tagname>          #删除标签
```

合并与衍合



```
$ git merge <branch>          #合并指定分支到当前分支
$ git rebase <branch>         #衍合指定分支到当前分支
```

远程操作



```
$ git remote -v                      #查看远程版本库信息
$ git remote show <remote>           #查看指定远程版本库信息
$ git remote add <remote> <url>      #添加远程版本库
$ git fetch <remote>                 #从远程库获取代码
$ git pull <remote> <branch>         #下载代码及快速合并
$ git push <remote> <branch>         #上传代码及快速合并
$ git push <remote\\> :<branch/tag-name\\>   #删除远程分支或标签
$ git push --tags                            #上传所有标签
```

更多内容请参见 Git 文档。

LFS 大文件支持

最近更新时间：2023-12-25 17:08:18

本文为您详细介绍如何使用 Git LFS 扩展。

功能介绍

CODING 支持 Git LFS（Git Large File Storage）扩展，使用 Git LFS 提交的大文件不占用 Git 仓库存储空间，理论上可以提交的单个文件大小无上限。

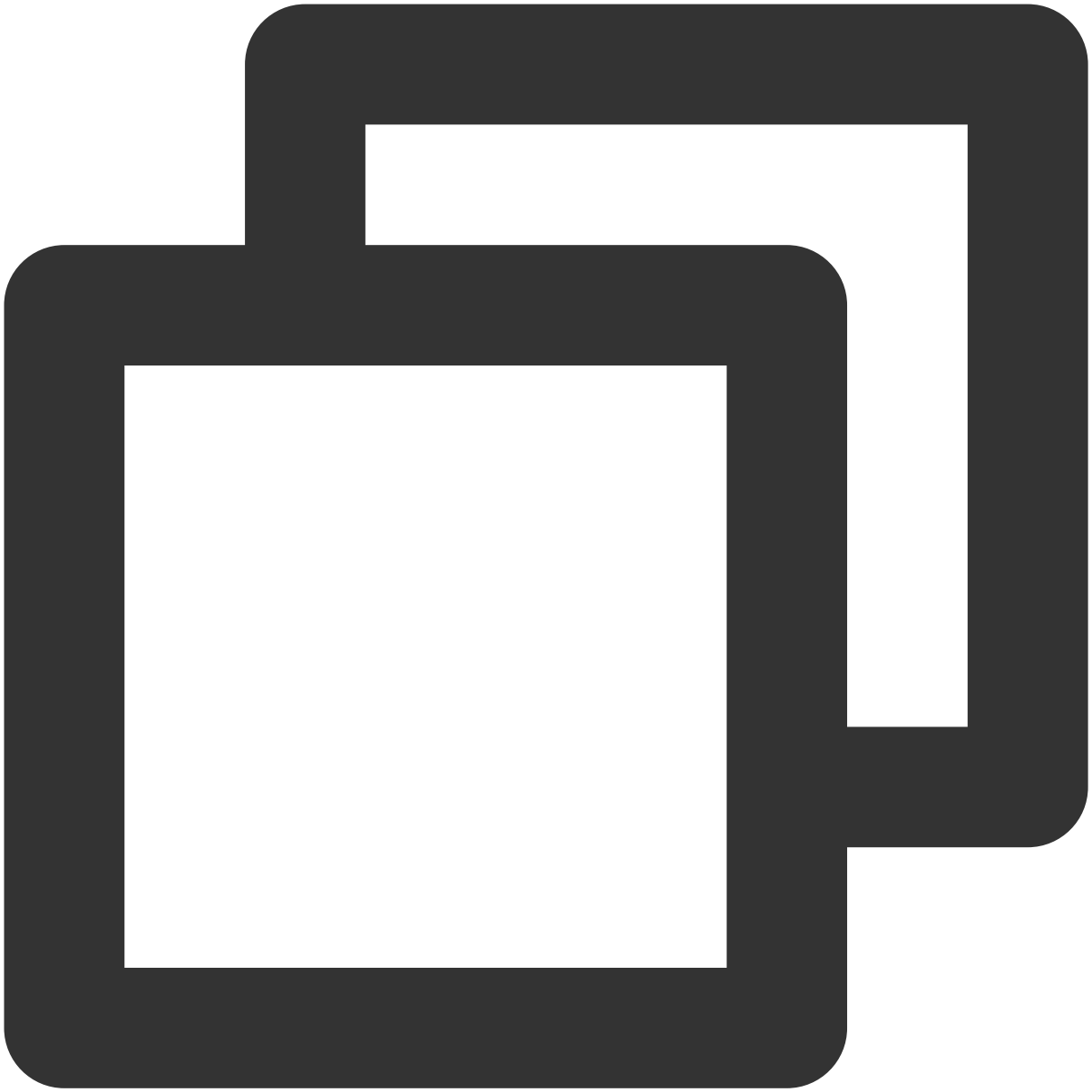
安装

注意：

Git LFS 插件时的 Git 版本需为 1.8.5 及以上。

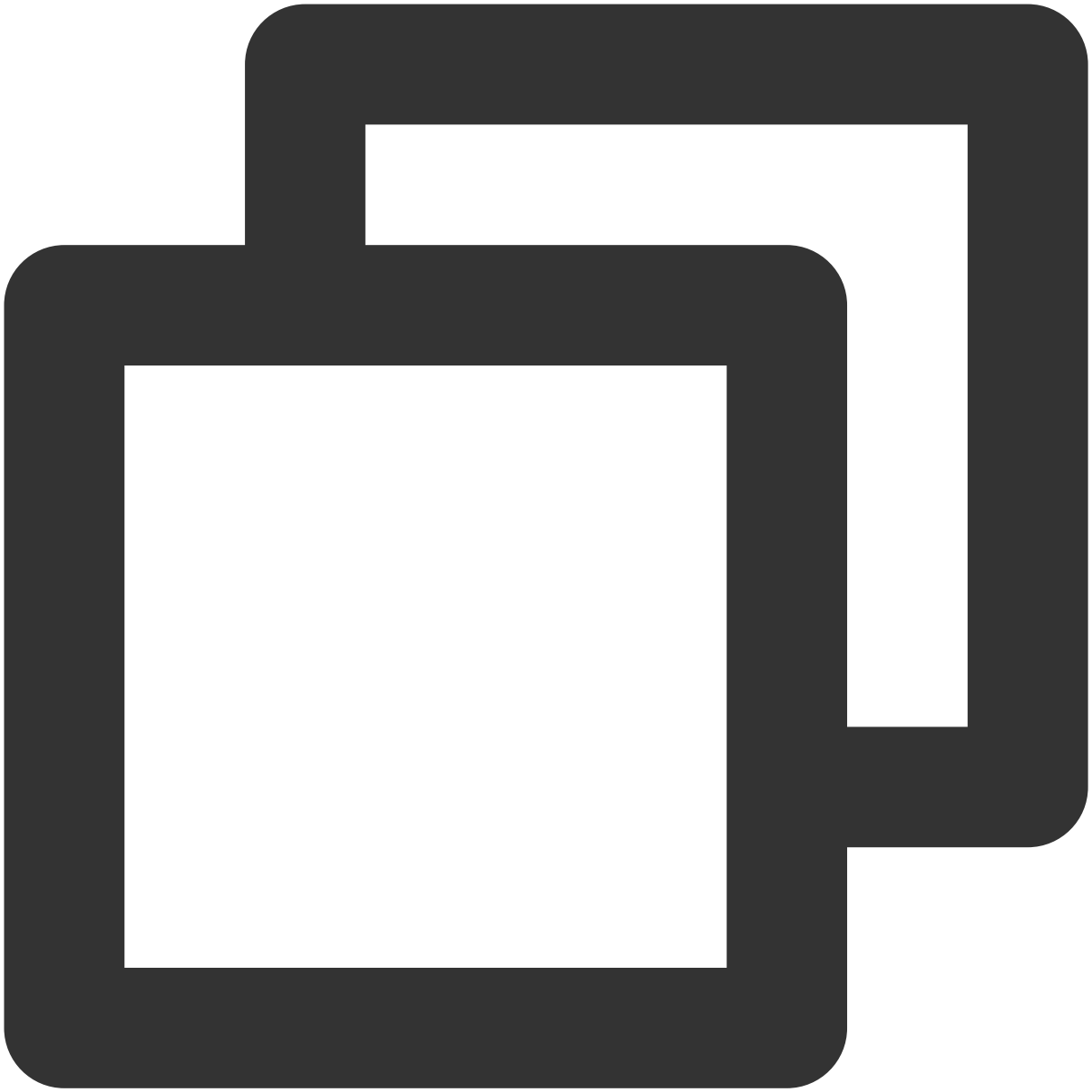
Linux

1. 下载 `git-lfs` 安装包。



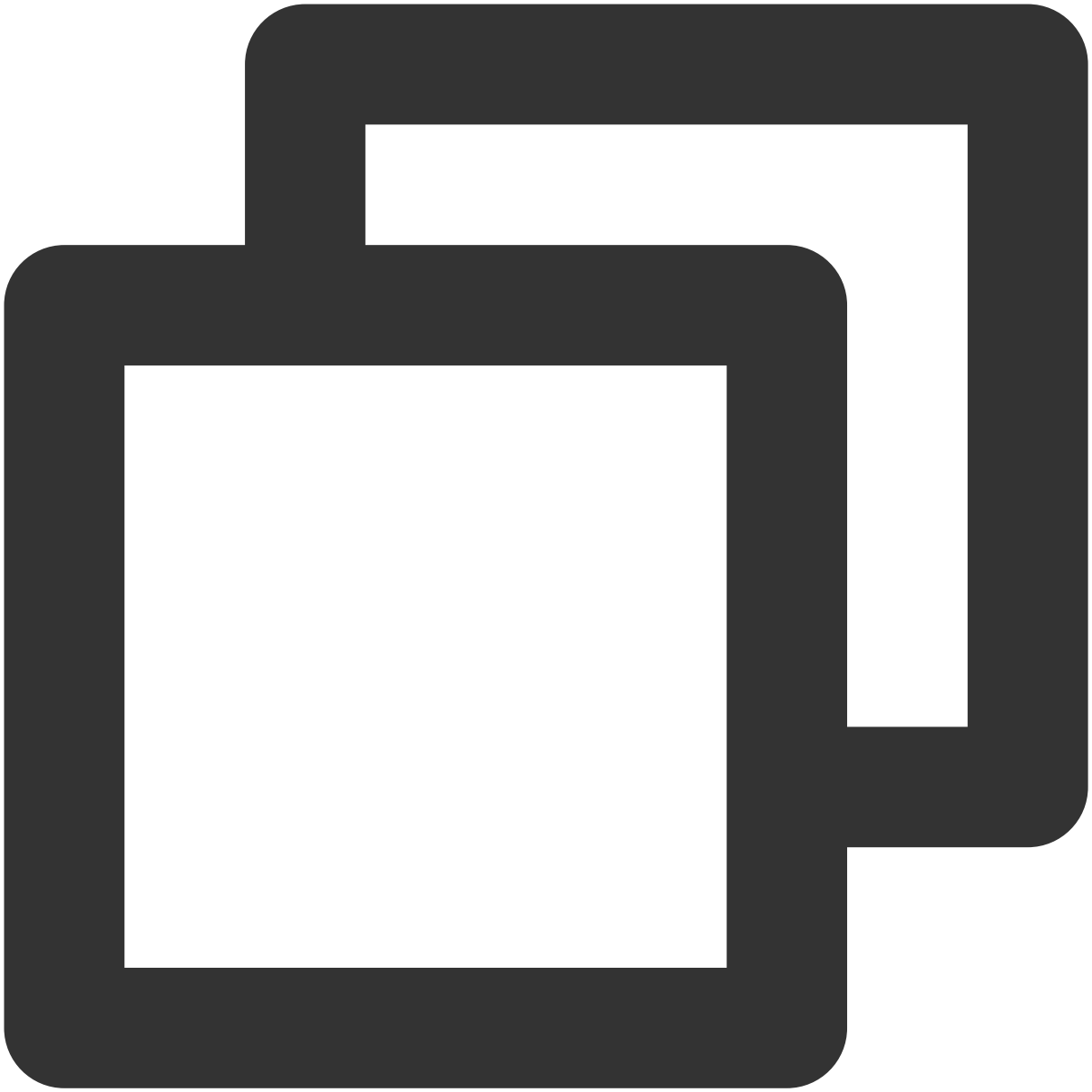
```
curl -s https://packagecloud.io/install/repositories/github/git-lfs/script.deb.s
```

2. 安装 `git-lfs` 。



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

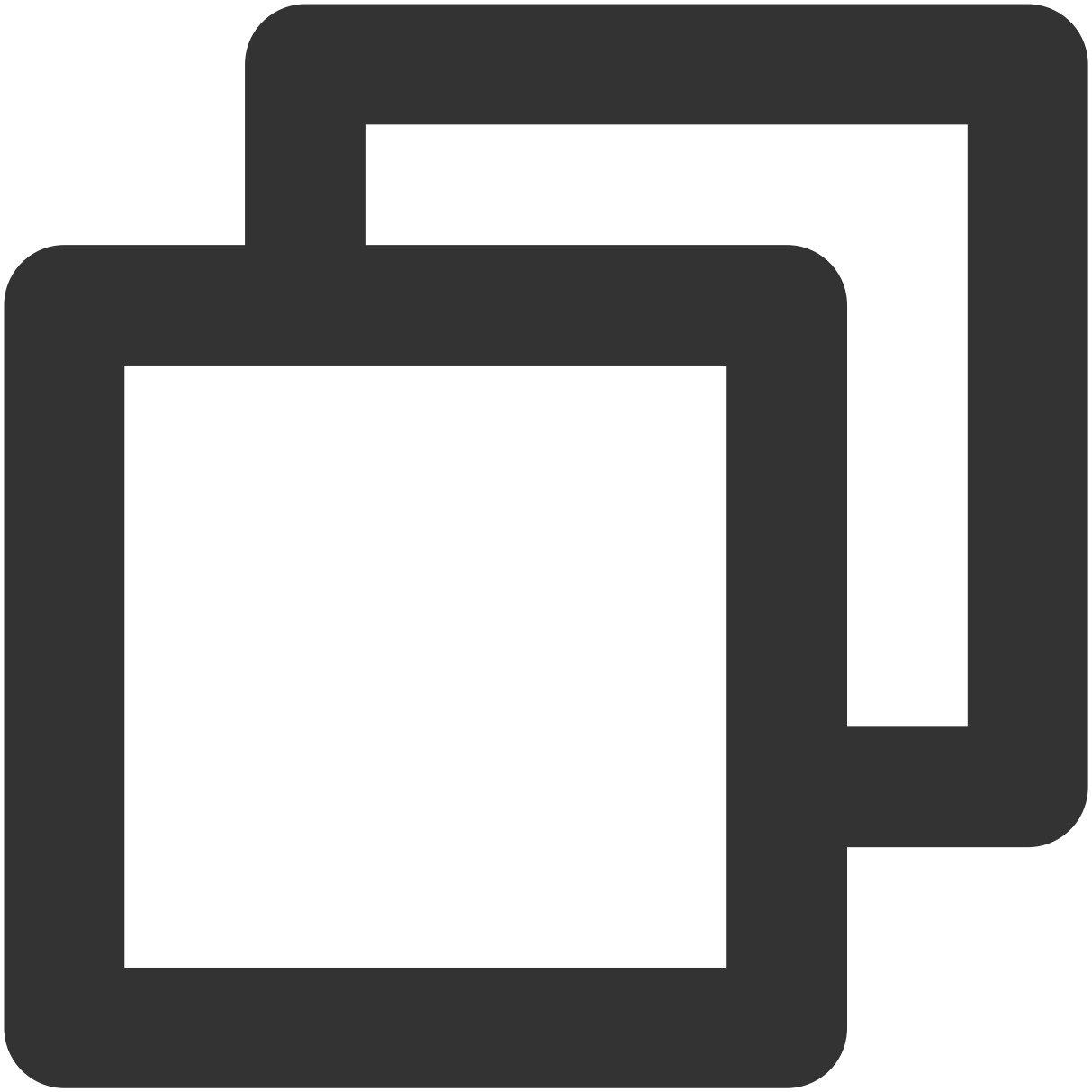
3. 将 LFS 工具部署到 Git 上。



```
git lfs install
```

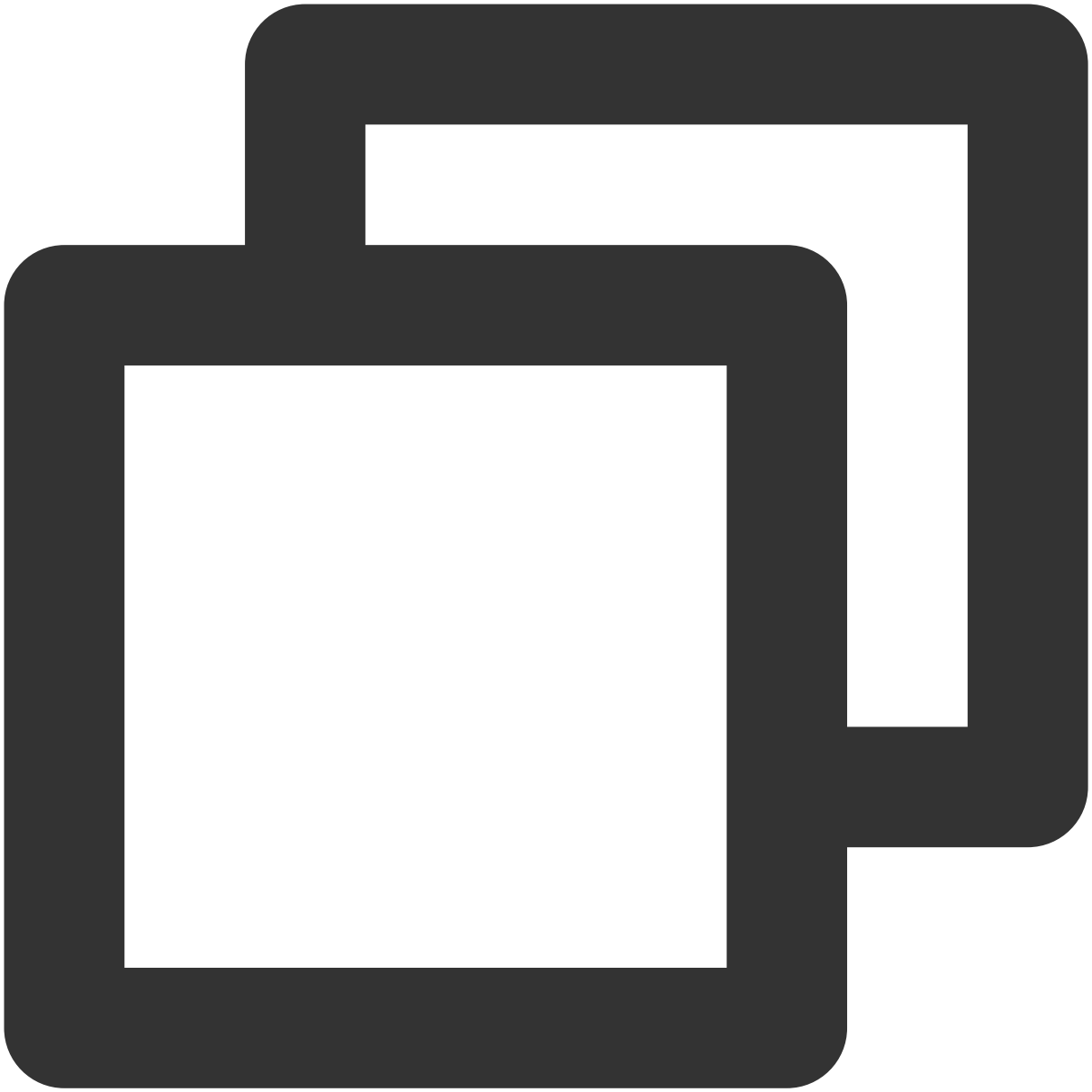
Mac

1. 安装 Homebrew。



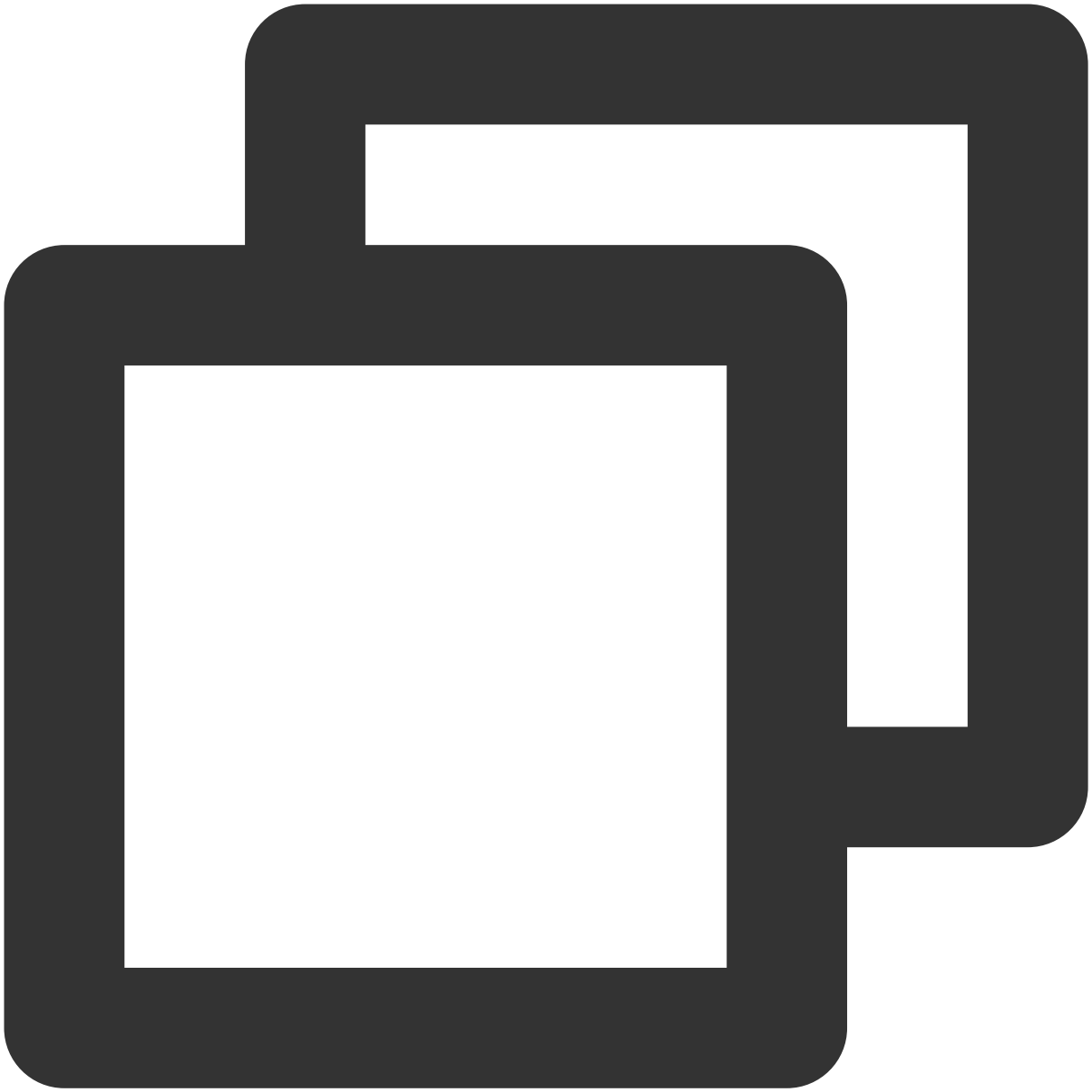
```
/usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/instal
```

2. 安装 `git-lfs` 。



```
brew install git-lfs
```

3. 将 LFS 工具部署到 Git 上。



```
git lfs install
```

Windows

1. 下载安装 Windows installer。
2. 运行 Windows installer。
3. 在命令行执行 `git lfs install` 。

使用

有关 Git 操作请查看 Git 常用命令。

追踪文件

没有特别说明的情况下，Git LFS 不会处理大文件问题，使用 `git lfs track` 命令进行大文件追踪。

追踪单个文件

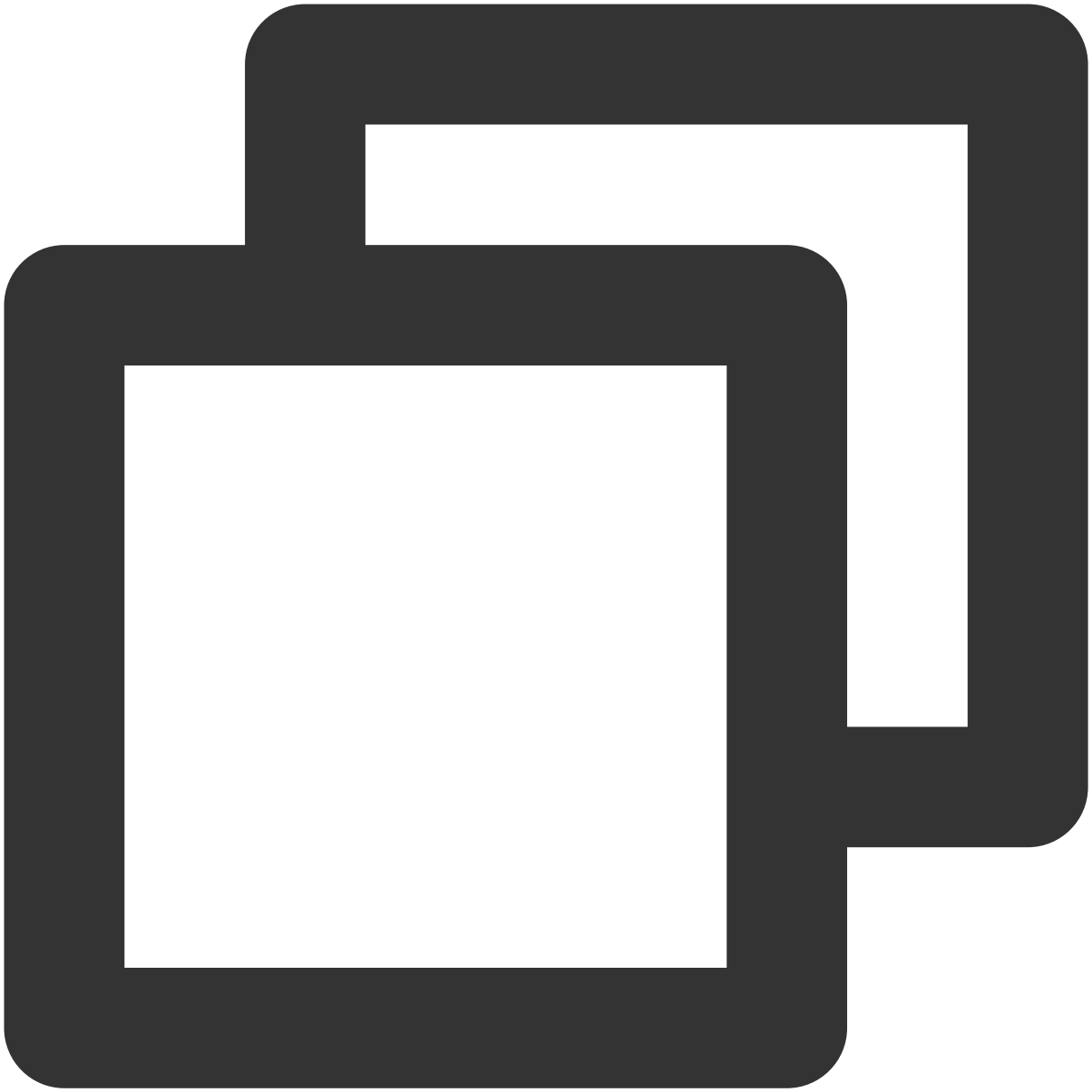
例如追踪一个名为 "coding.png" 的文件，使用 `git lfs track "coding.png"` 命令

追踪同一后缀的所有文件

如果要追踪所有后缀为 "png" 的文件，使用 `git lfs track "*.png"` 命令。运行此命令后，不但会追踪已存在的所有后缀为 "png" 的文件，也包括以后创建的 "png" 文件。

查看正在追踪的文件模式（patterns）

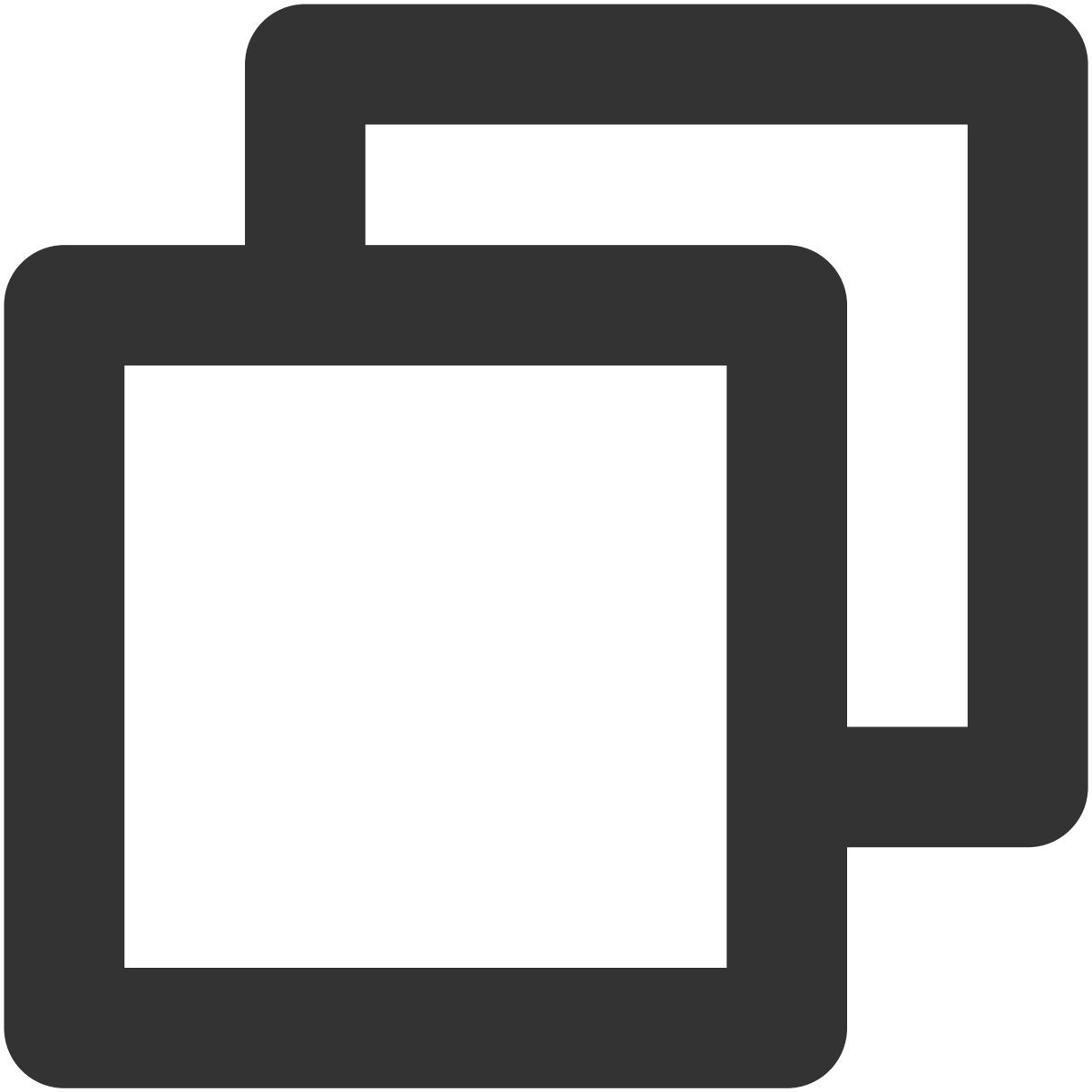
运行命令 `git lfs track` ：



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

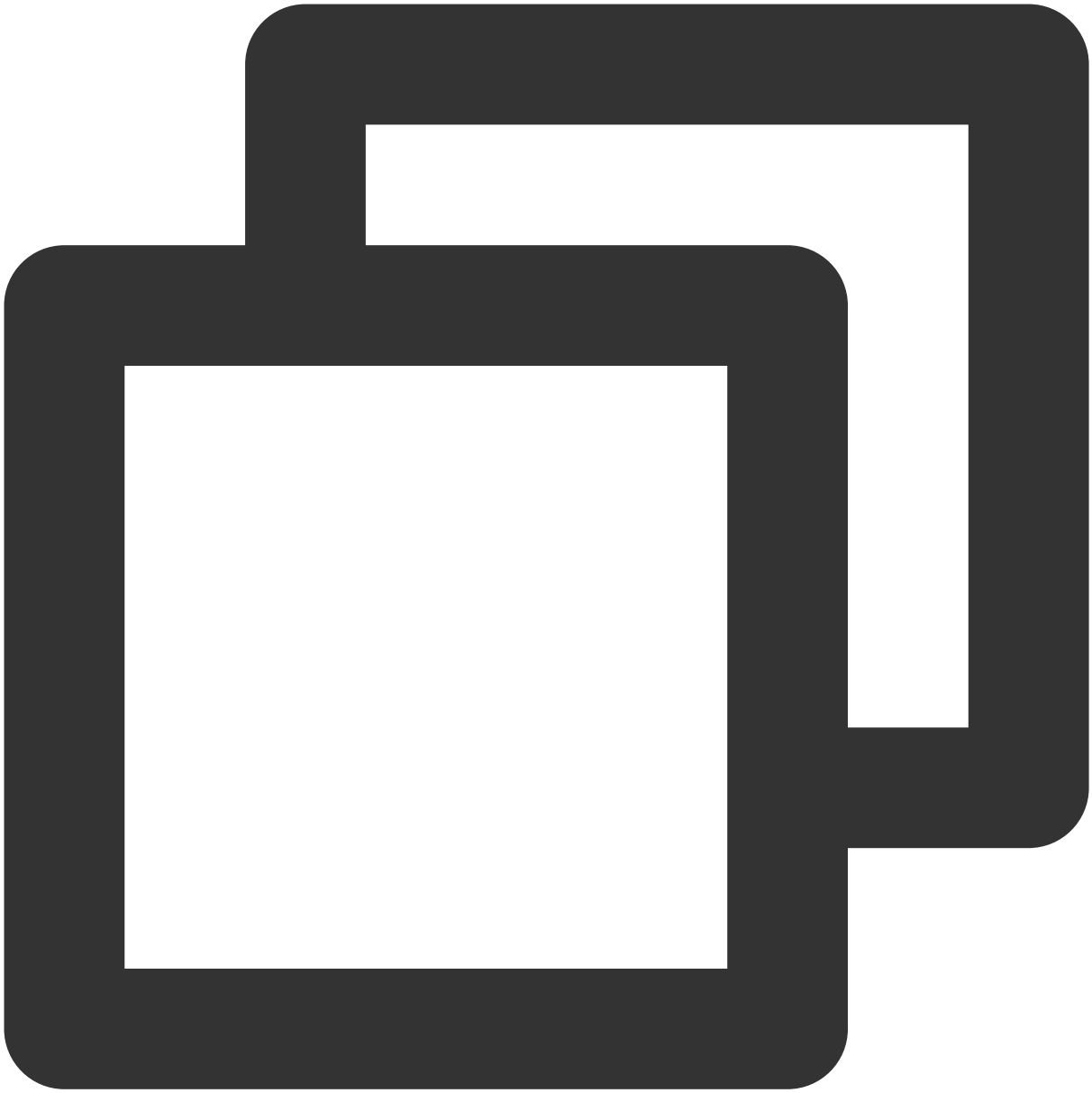
提交大文件

提交代码时需要将 ".gitattributes" 文件也提交到仓库，提交完成后，执行 `git lfs ls-files` 命令可以查看 LFS 跟踪的文件列表。



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

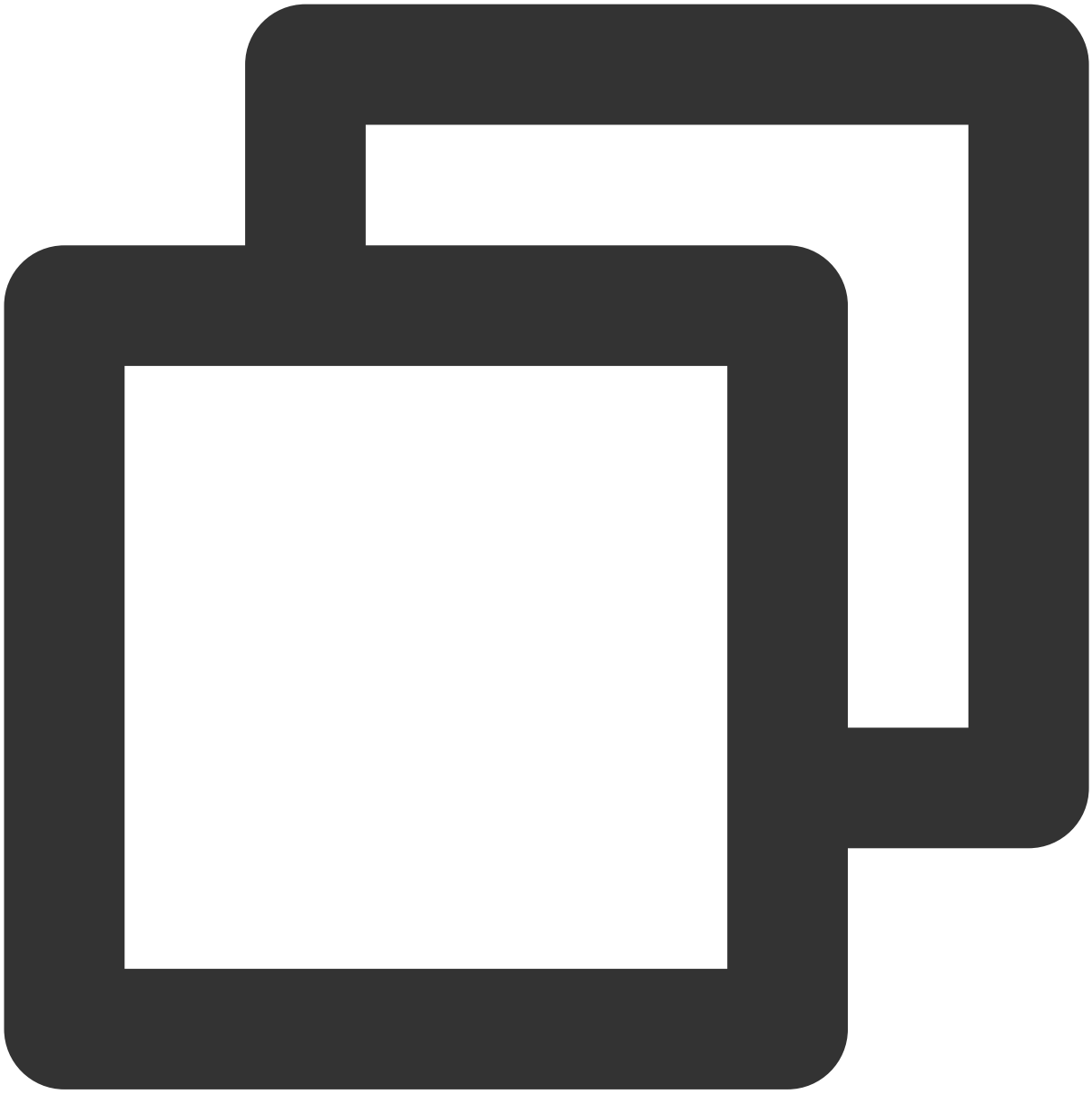
将代码 push 到远程仓库后，LFS 跟踪的文件会以 "Git LFS" 的形式显示:



```
$ 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
```

克隆包含 Git LFS 文件的远程仓库

使用 `git lfs clone` 命令 clone 包含 "Git LFS" 文件的远程仓库到本地。



```
$ 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 connectively...done.
```

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

说明：

了解更多 Git LFS 的使用，可执行 `git lfs help` 命令查看帮助。

如需将原有仓库的文件以 LFS 方式存储，请参见 参见教程。

Go get 支持

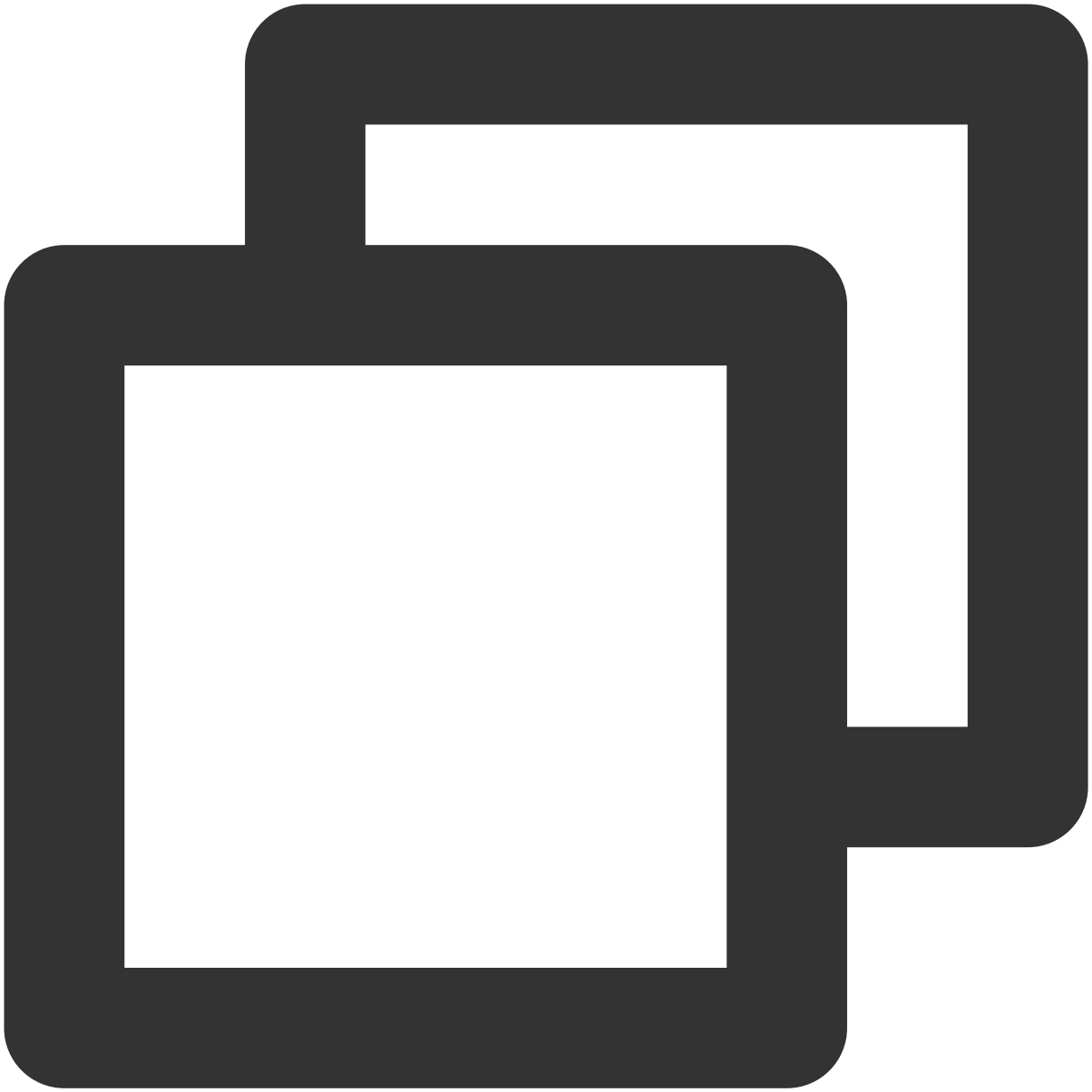
最近更新时间：2023-12-25 17:08:18

功能介绍

Golang Get 可以借助代码管理工具通过远程拉取或更新代码包及其依赖包，并自动完成编译和安装，使得整个过程就像安装应用程式一样简单。目前 CODING 代码仓库已支持在多仓库中使用 **go get** 功能，以下是快速上手指南。

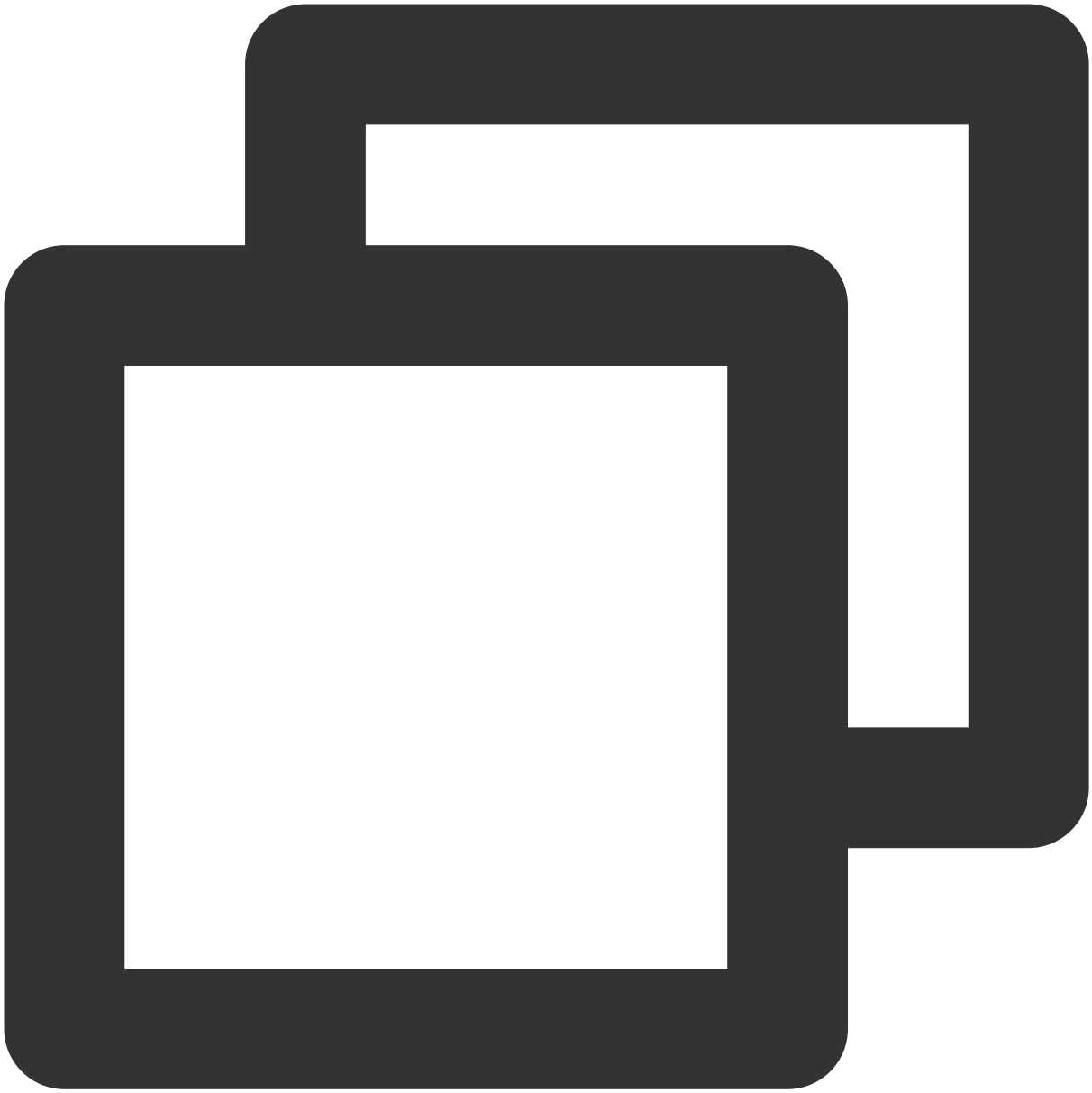
快速开始

假如用户拥有 A 仓库，其代码仓库的 Git HTTPS 地址是：`https://e.coding.net/{team}/{project}/{repo}.git`（花括号中是变量，以实际情况为准），以该仓库 `https://e.coding.net/baulk/jackson/mux.git` 为例：
用户可以通过以下命令设置好模块名。



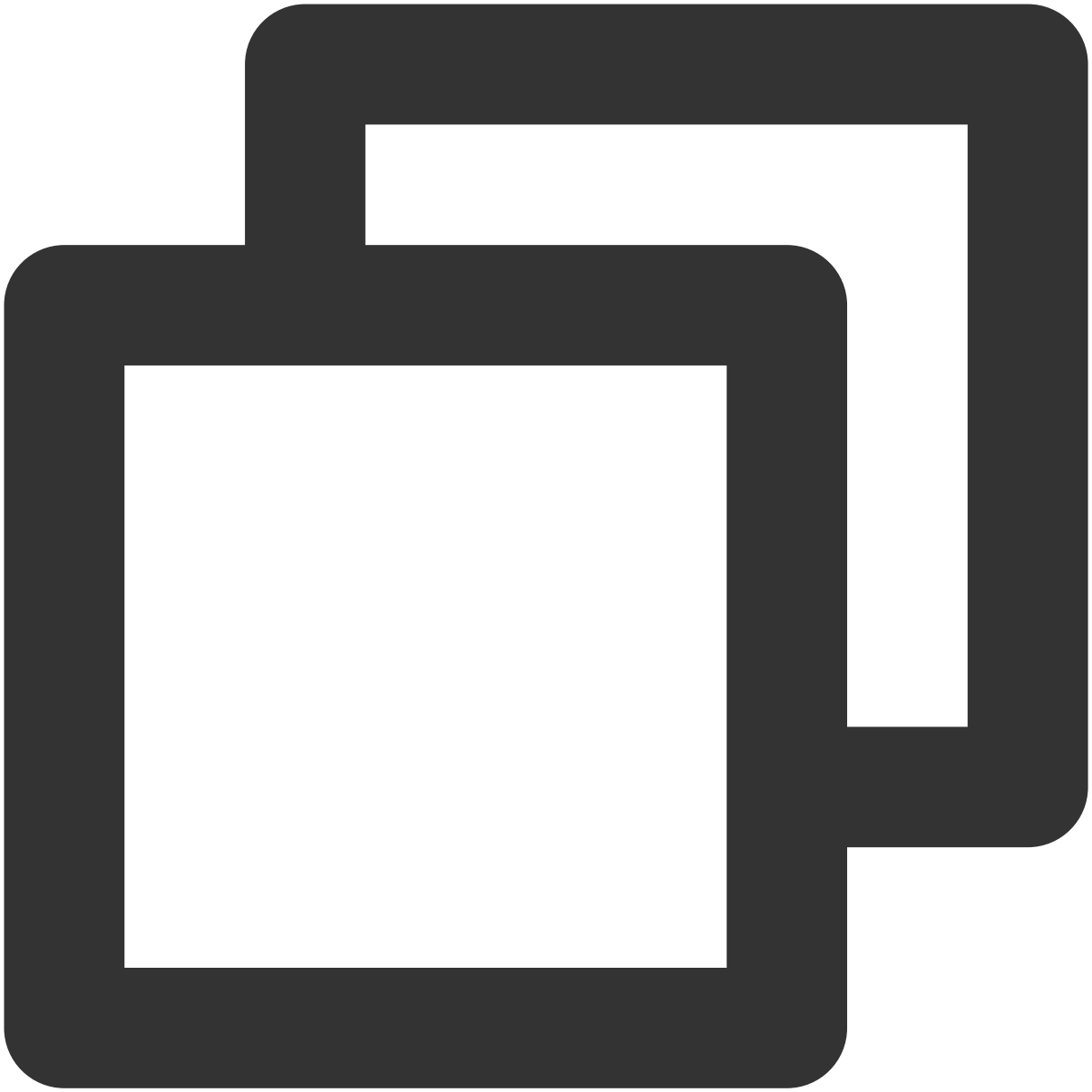
```
go mod init e.coding.net/baulk/jackson/mux
```

如果要通过 go get 获得模块，可以运行如下命令：



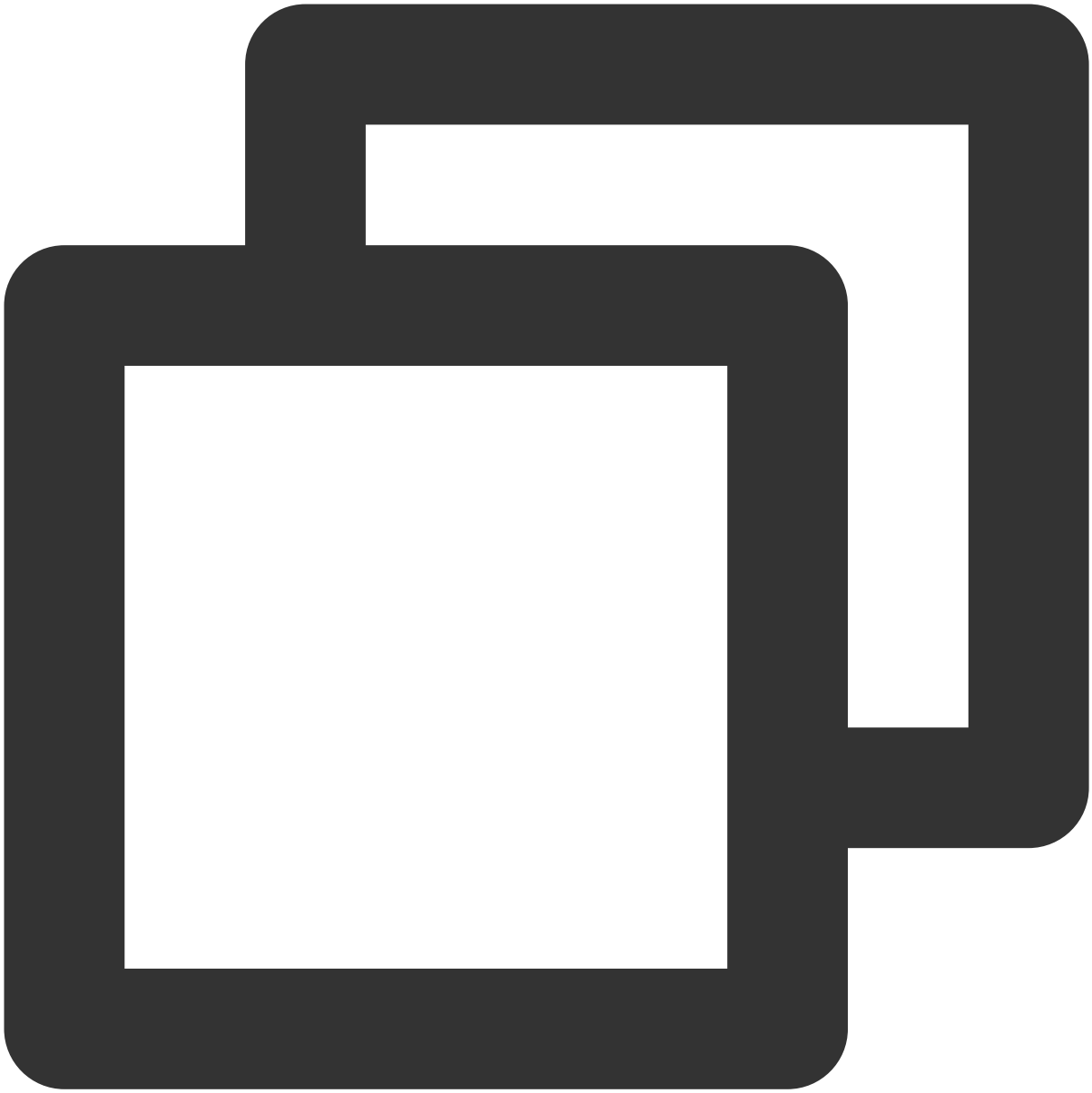
```
go get e.coding.net/baulk/jackson/mux
```

另外多仓库也支持获取子仓库的模块：



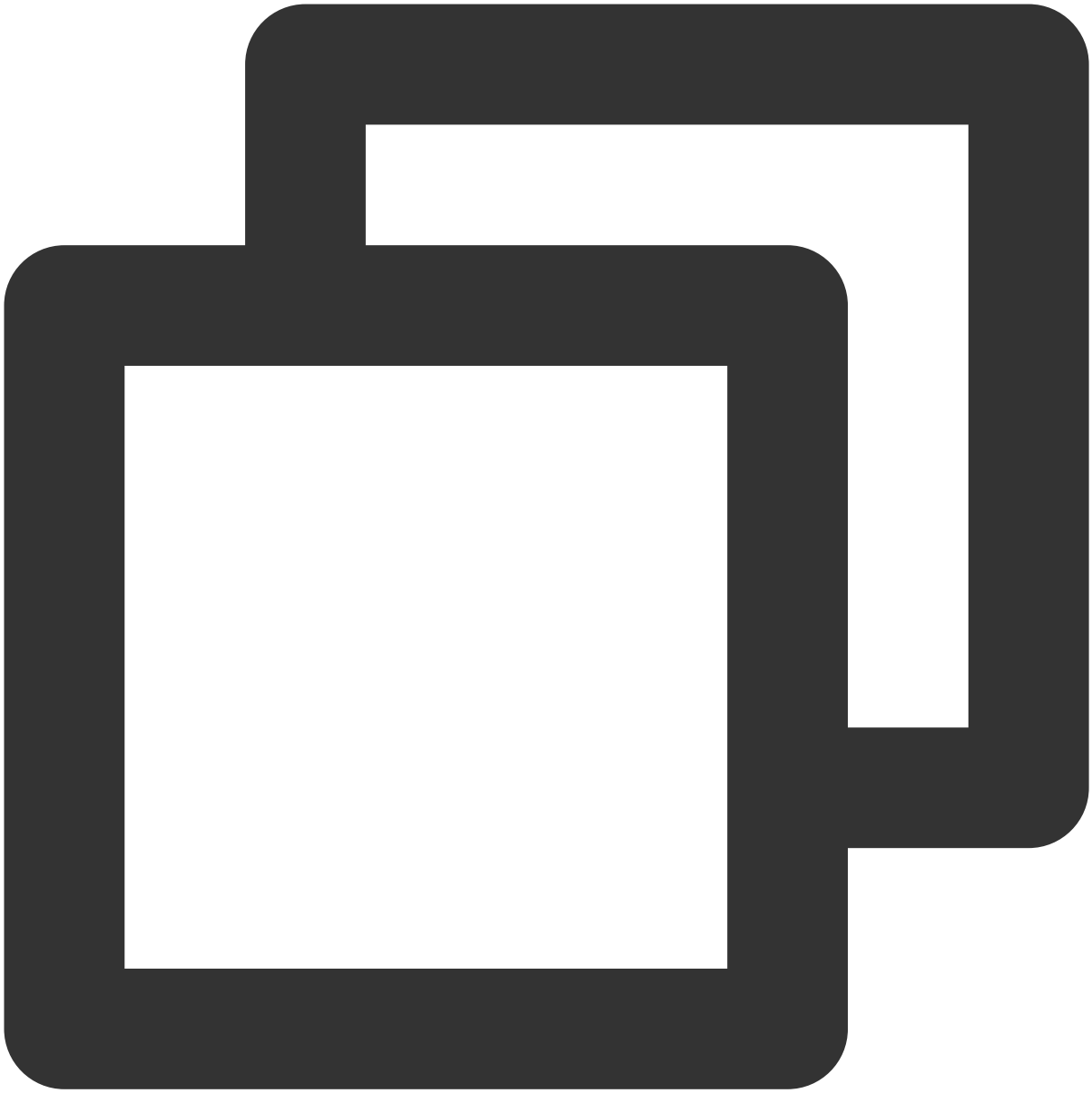
```
go get e.coding.net/baulk/jackson/mux/dev
```

有一些存储库的 Git HTTPS 克隆地址是: `https://e.coding.net/{team}/{project}.git` ，用户可以通过如下命令设置好模块名：



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

如果要通过 go get 获得模块，可以运行如下命令：



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

注意：

此类存储库不完全支持直接获取子模块，您应该使用 `e.coding.net/team/project/project` 作为模块名、获得模块及其子模块。

个人设置
访问令牌

最近更新时间：2023-12-25 17:08:18

个人访问令牌类似某些系统中的应用专用密码，生成后可根据设置的权限访问特定的 API， 可以用于创建自己的程序或者脚本。

操作步骤

1. 登录 CODING 控制台，单击**立即使用**进入 CODING 使用页面。
2. 鼠标移动至右上角的头像图标处，单击浮现出来的**个人设置**，进入个人设置页面。
3. 单击左侧菜单栏的**访问令牌**，进入访问令牌页面。

创建访问令牌

1. 单击**新建令牌**，填写令牌描述，以及选择令牌的访问权限。
2. 单击**创建令牌**，提交时需要输入服务密码进行身份验证。
3. 单击**确定**，即可创建访问令牌。
4. 提交后，页面将跳回访问令牌列表页面，并显示新建令牌的内容。

注意：

令牌的内容只会显示一次，为了安全，请将其复制粘贴至您需要使用的程序或者脚本中，并不要保存副本。如果您是需要调试，请在调试后，选择重新生成令牌，再将其粘贴到最终使用程序或者脚本中。

编辑访问令牌

1. 单击对应令牌右侧的**编辑**，进入编辑页面。
2. 可以编辑令牌的描述和重新选择权限。
3. 单击**更新令牌**，输入服务密码进行身份验证。
4. 单击**确定**，即可完成编辑。

说明：

如果遗失或遗忘了该令牌，在编辑页面可以单击**重新生成**，更新令牌。

删除访问令牌

1. 单击对应令牌右侧的**删除**，删除特定令牌，也可单击**移除所有**，一键移除所有访问令牌。
2. 输入服务密码进行身份验证，单击**确定**，即可删除令牌。

说明：

如果不希望令牌再被继续使用了，为了避免任何有可能的泄露，建议将其删除。

账户 SSH 公钥与项目令牌

最近更新时间：2023-12-25 17:08:18

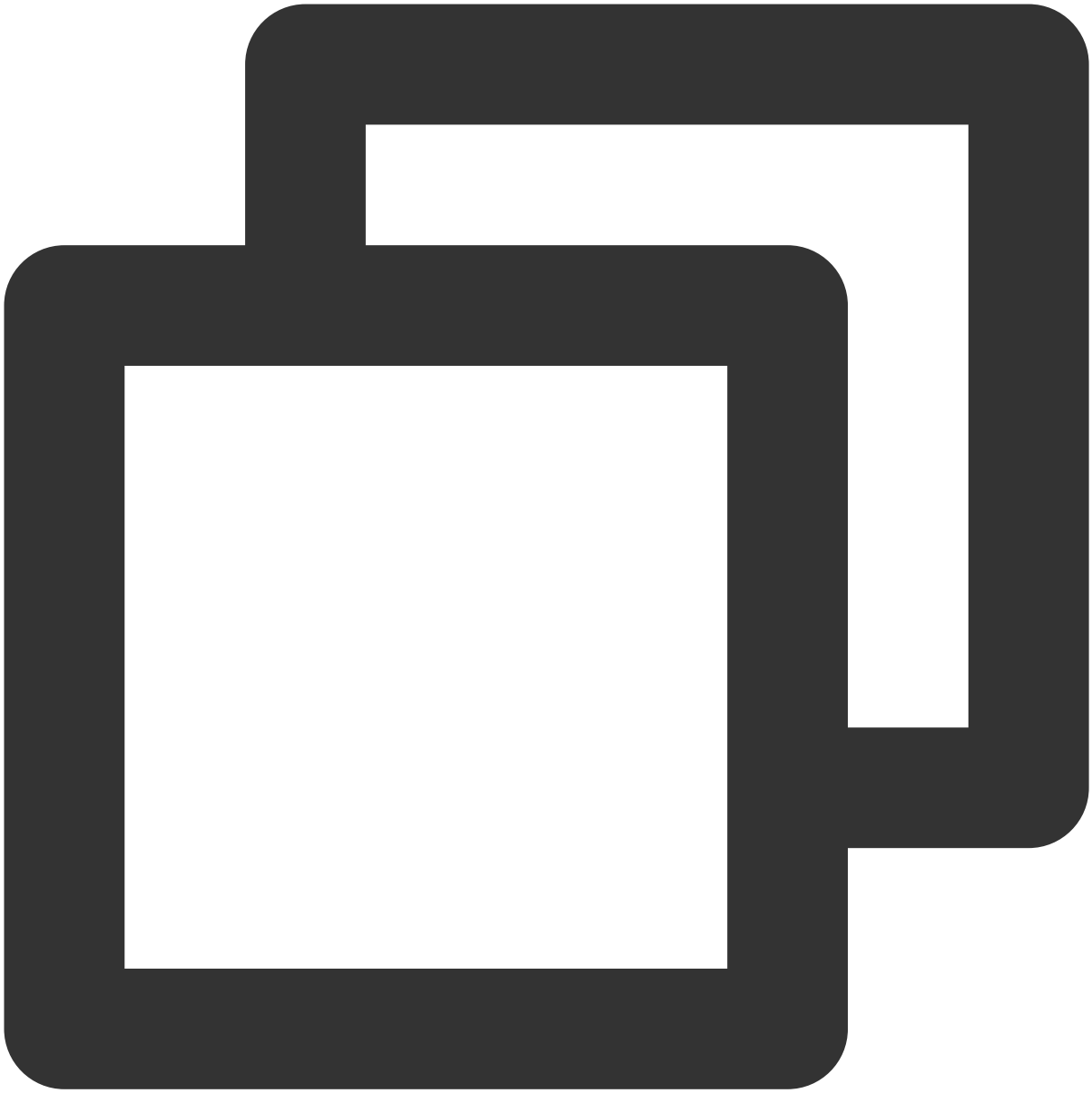
SSH 公钥在 CODING 中会因使用场景的差异而有不同的权限范围。本文为您介绍账户 SSH 公钥与项目令牌的区别。

功能介绍

SSH 公钥文件与 CODING 账户关联，便称为**账户 SSH 公钥**，配置后拥有账户下所有项目的读写权限；如果和某一个项目关联，则称为**项目令牌**，配置后默认拥有该项目的只读权限。

生成公钥

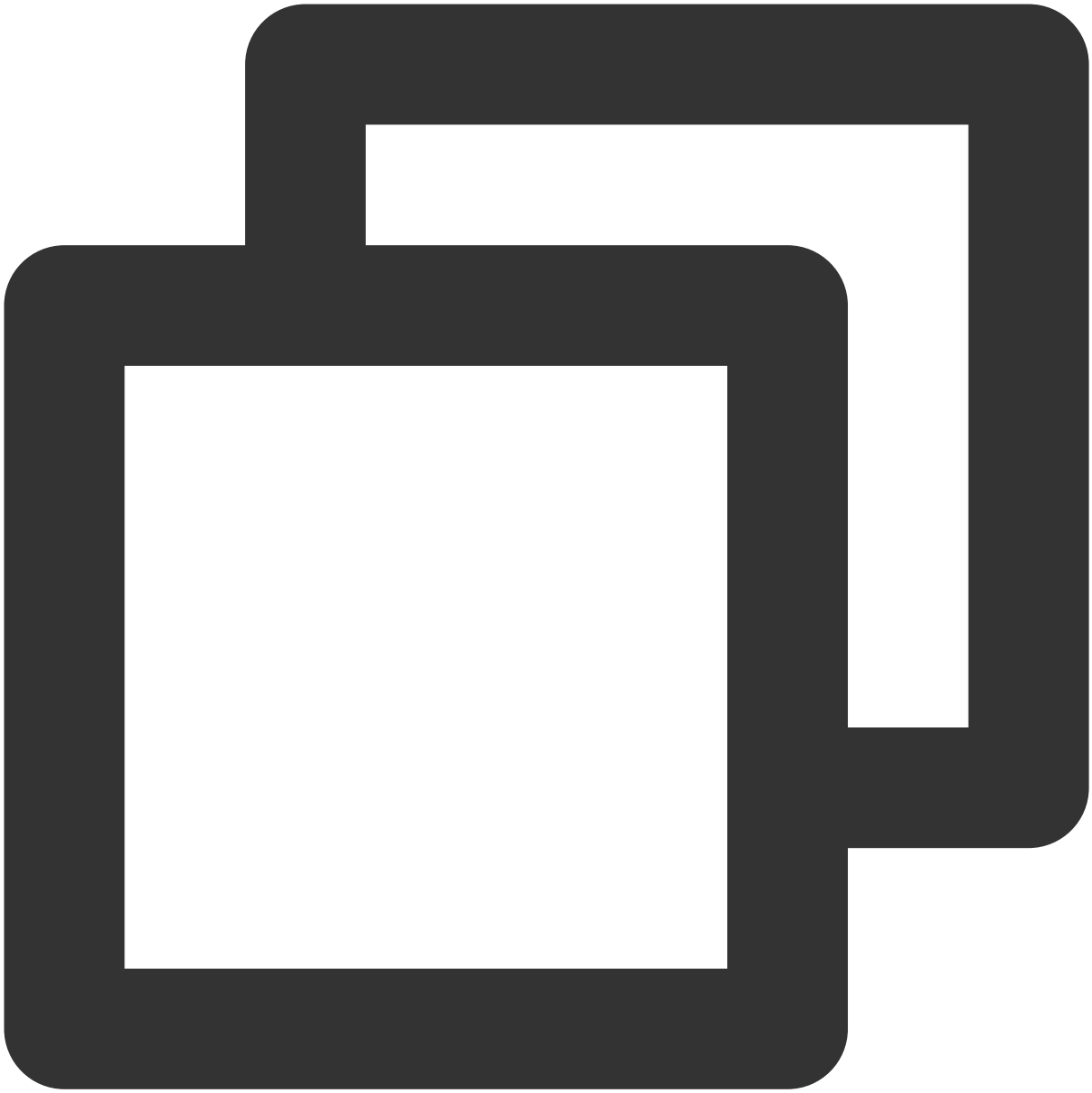
执行命令：



```
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]  // 推荐
Enter passphrase (empty for no passphrase):   // 此处不填写，回车即可；如果填写密码，则每次
```

若需要使用多个 SSH 密钥对（您可能同时在多个代码托管平台工作），在提示“Enter file in which to save the key” 时输入一个新的文件名称就不会覆盖默认的密钥对。

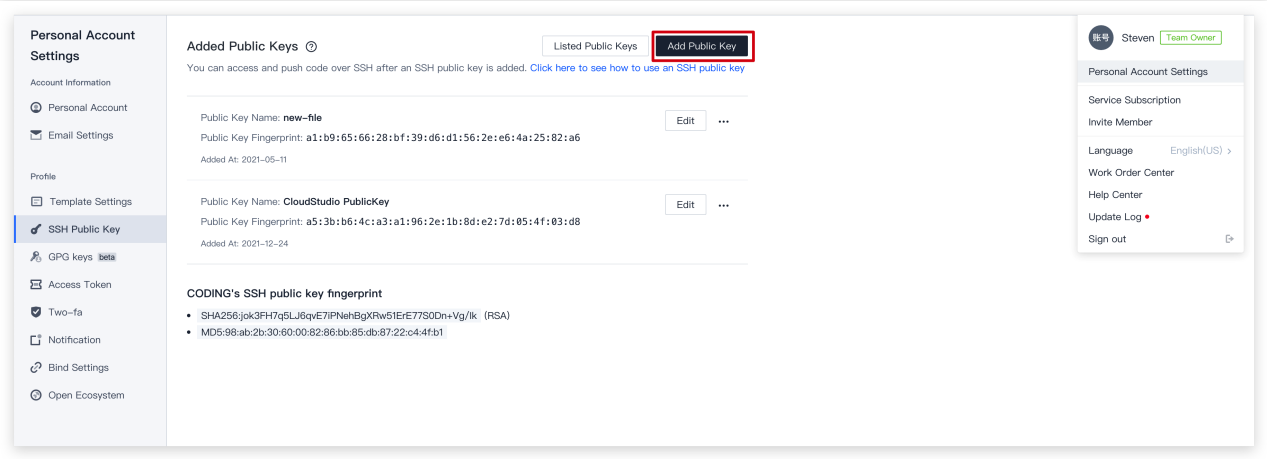
成功之后显示如下信息：



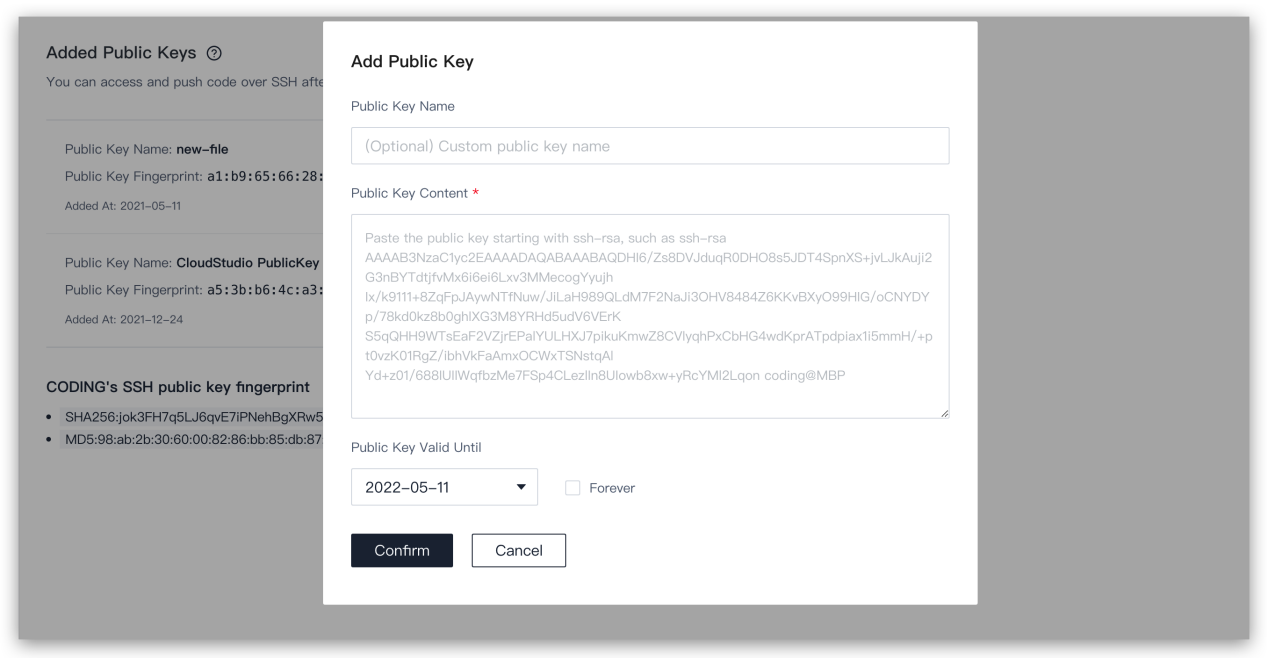
```
Your identification has been saved in /Users/you/.ssh/id_rsa.
# Your public key has been saved in /Users/you/.ssh/id_rsa.pub.
# The key fingerprint is:
# 01:0f:f4:3b:ca:85:d6:17:a1:7d:f0:68:9d:f0:a2:db your.email@example.com
```

添加账户 SSH 公钥

1. 在终端输入 `open ~/.ssh` ，用文本编辑器打开 `id_rsa.pub` 文件（此处是生成公钥的默认名称，如果生成公钥时采用了其他名称，打开相对应的文件即可），复制全部内容。
2. 单击页面右上角个人头像，选择**个人账户设置**。进入**个人账户设置** > **个人设置** > **SSH 公钥**页面。



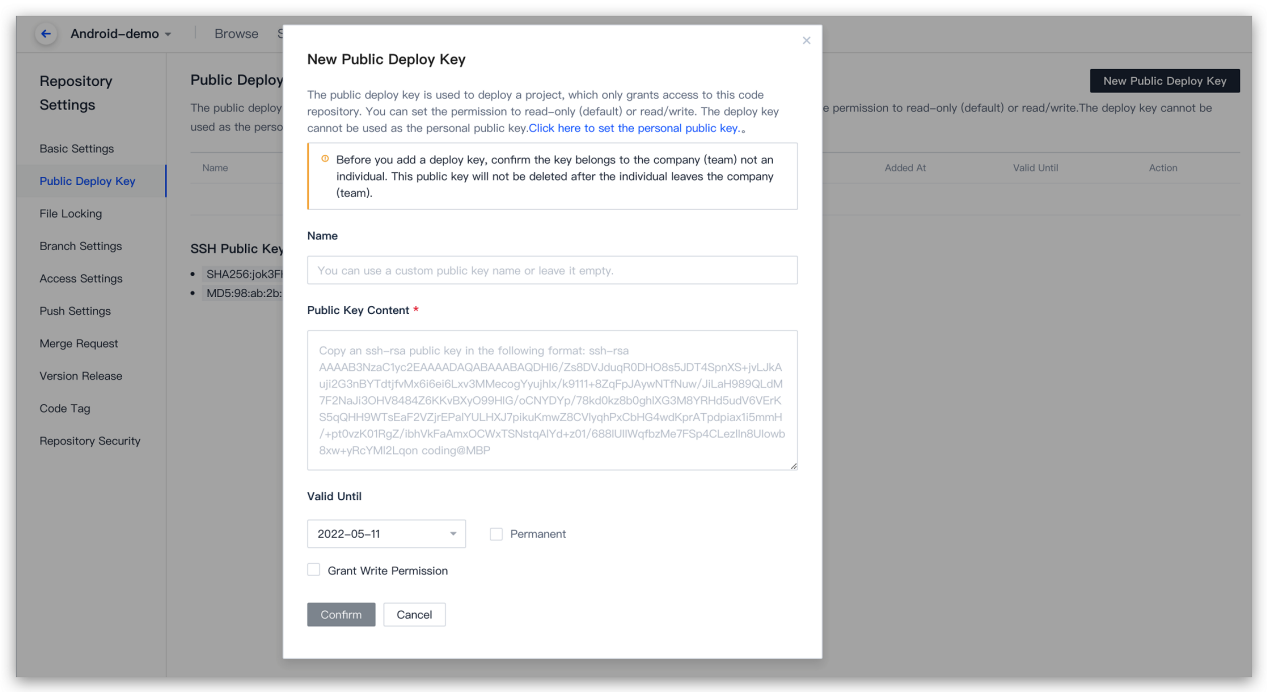
3. 将第一步中复制的内容填写到**公钥内容**一栏，公钥名称按需填写即可。
4. 设定公钥有效期，可选择具体日期或设置永久有效。



5. 单击**添加**，然后输入账户密码即可成功添加公钥。
6. 完成后在命令行测试，首次建立链接会要求信任主机。命令 `ssh -T git@e.coding.net` 。您也可以通过密钥指纹鉴权验证是否与真正的 CODING 远程仓库所连接。

添加部署公钥

1. 在终端输入 `open ~/.ssh` ，用文本编辑器打开 `id_deploy.pub` 文件（此处部署公钥名称为 `id_deploy.pub` ，您在生成部署公钥的时候可以自定义名称），复制全部内容。
2. 进入目标项目内的代码仓库 > **部署公钥**页面，单击**新建部署公钥**。



3. 将第一步中复制的内容填写到**公钥内容**一栏，公钥名称自定义。
4. 单击**新建**，然后输入账户密码即可成功添加部署公钥。

说明：

部署公钥默认拥有该项目的只读权限，如果需要获取推送权限，请勾选**授予推送权限**。