CODING DevOps

代码托管

腾讯云

文档目录

代码托管
快速开始

最近更新时间：2024-09-05 17:38:41

初始化阶段

本地安装 Git

Windows

1. 从 Git 官网 下载 Git 客户端并按提示完成安装，推荐使用默认选项完成安装。
2. 完成安装后即可使用鼠标右键调出 Git Bash 终端，开始您的 Git 之旅。

Linux

在 Linux 上可直接通过系统提供的包管理工具安装预编译好的 Git 二进制安装包。

- 在 Fedora/Centos 上用 yum 安装：`yum install git-core`
- 在 Ubuntu/Debian 上可以用 apt-get 安装：`apt-get install git`

macOS

1. 运行以下命令安装包管理工具 homebrew 。

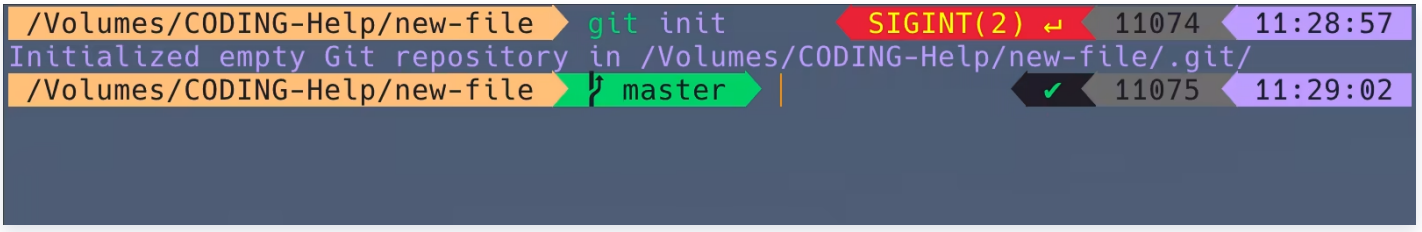
   `/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"`

   > **说明：**
   > 若出现错误提示，请参考此 开源仓库，切换至国内源进行下载。

2. homebrew 安装好后，在终端输入 `brew install git` 命令完成 Git 安装。
3. 安装 Git 之后，可运行 `git --version` 查看当前 Git 版本。

本地环境初始化

完成安装后，新建文件夹并在终端中进入该文件夹，输入 git init 命令完成本地环境初始化。



设置用户信息

安装完 Git 后应该及时设置提交者名称与邮箱地址。此后的每次提交都会使用这些信息作为记录。设置用户信息的命令如下：

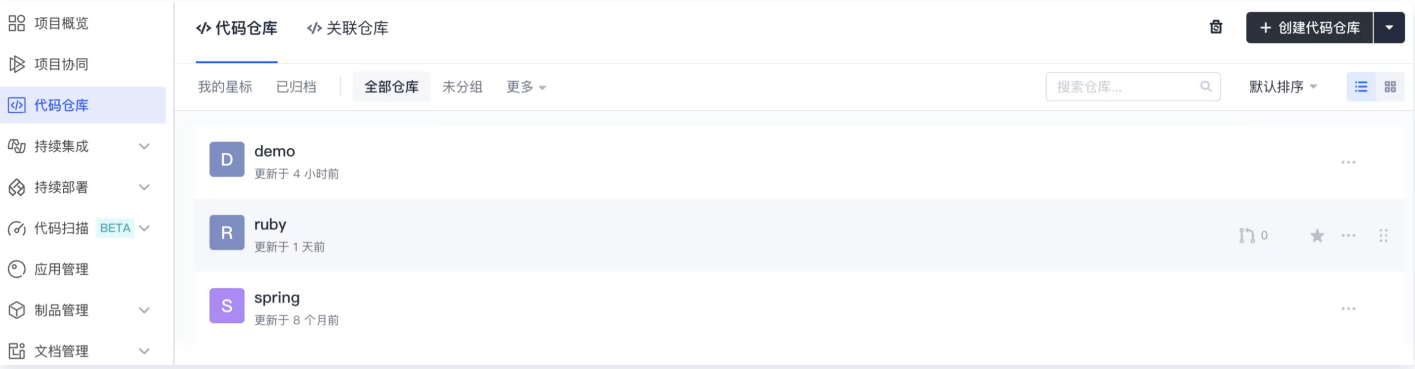
```
git config --global user.name "你的名称"
git config --global user.email "你的邮箱"
```

例如，您的 CODING 账户昵称叫“大黑”，Git 中的用户信息配置为：`user.name 大白` 、`user.email dabai@coding.net` 。当您推送代码到 CODING 仓库后，动态显示如下图：

在 Git 中配置的用户信息是可自定义的，建议您直接填写为 CODING 用户名与注册邮箱，以便更好协作。

创建 CODING 代码仓库

进入任意项目，单击左侧导航栏的**代码仓库**进入代码仓库管理页面，新建或进入任一仓库。



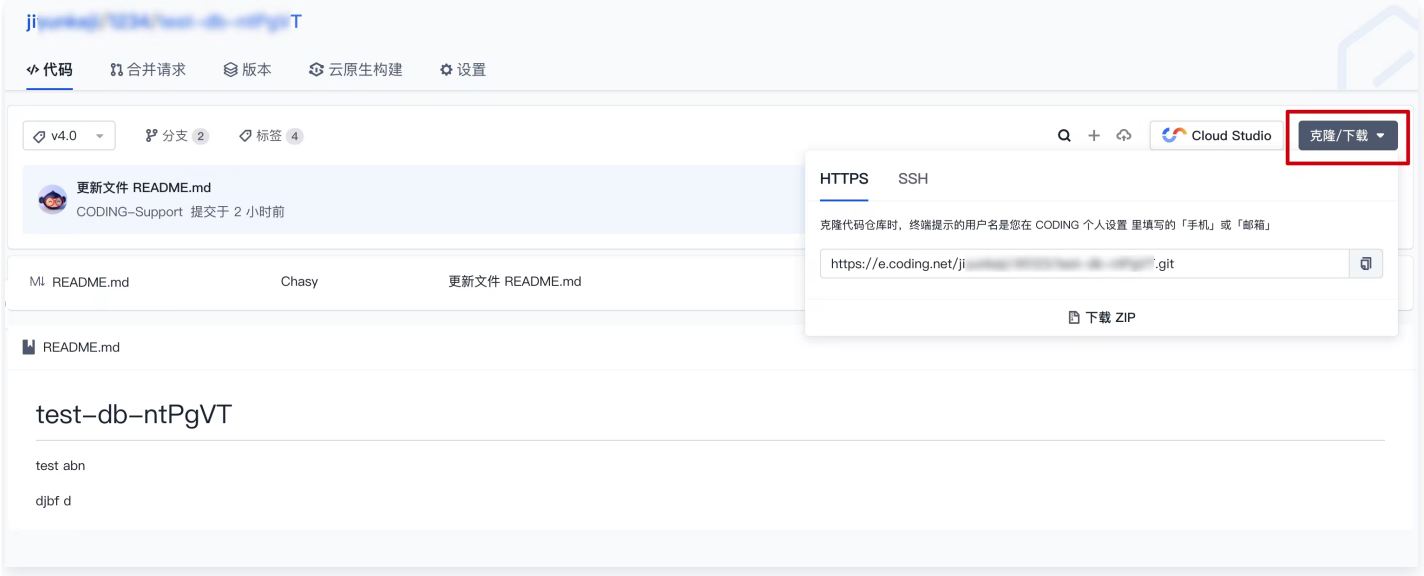
若代码仓库入口未显示，项目管理员需进入该项目，单击左下角的**项目设置**，然后在**项目与成员 > 菜单管理**中打开代码仓库功能。

推拉代码

在此操作中将会演示如何从远端仓库拉取代码 / 上传本地代码至远端仓库，助力您的代码上云之旅。

拉取远程仓库代码

在本地完成代码仓库的初始化后，在文件夹中调出终端输入 `git clone + 仓库地址` 命令拉取代码。



首次拉取后会提示填写凭据，此处填写在注册 CODING 时所使用的邮箱与密码即可。您也可以前往左下角的个人账户设置处修改凭证信息。
命令操作提示成功之后，您可以在本地代码仓库中进行代码修改。

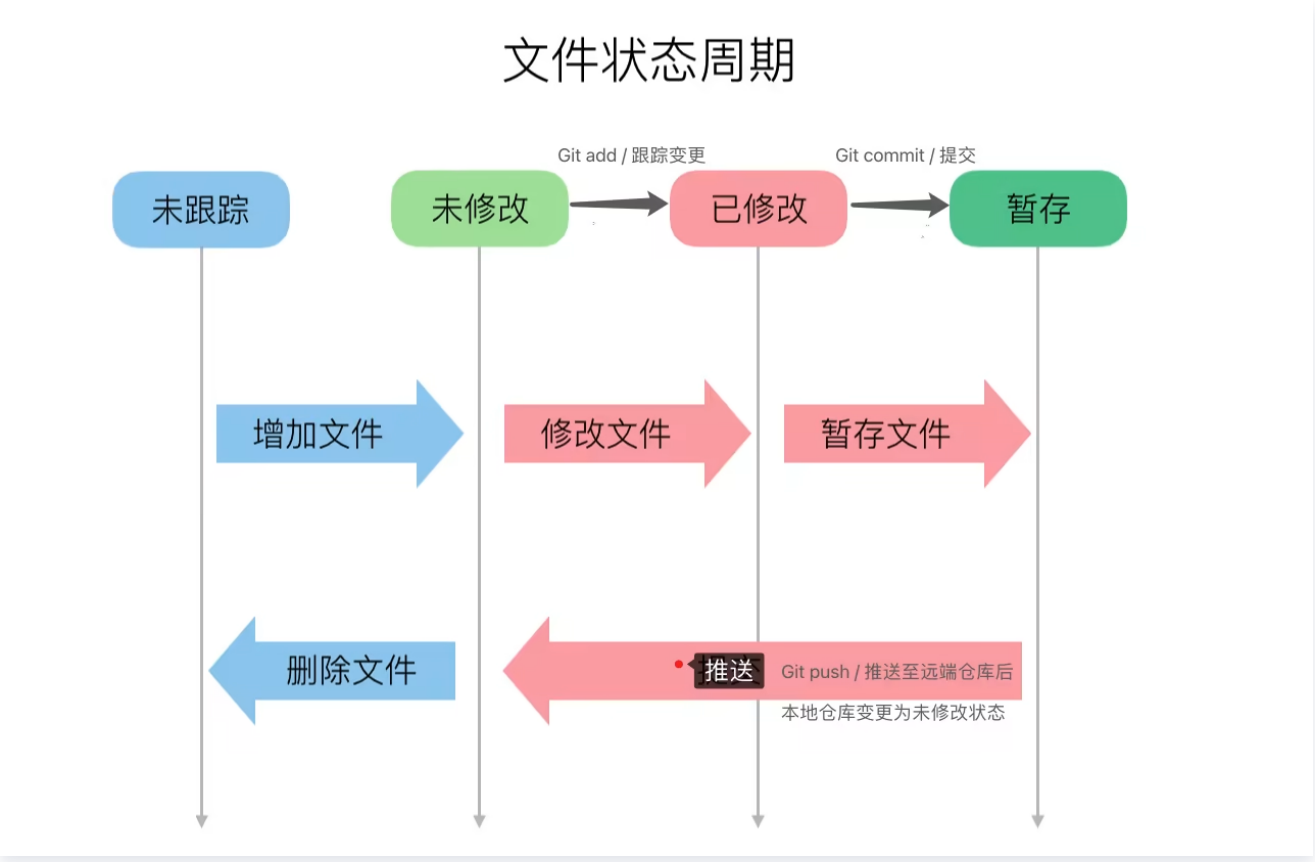
```
/Volumes/CODING-Help/视频制作/代码仓库 git clone https://e.coding.net/StrayBirds/demo/demo.git
Cloning into 'demo'...
Username for 'https://e.coding.net':
Password for 'https://                                    :
remote: Enumerating objects: 13, done.
remote: Counting objects: 100% (13/13), done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 13 (delta 4), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (13/13), done.
/Volumes/CODING-Help/视频制作/代码仓库 cd demo
/Volumes/CODING-Help/视频制作/代码仓库/demo  master
```

编辑文件

在文件夹中新建 `readme.txt` 和 `learn-git.txt` 文件，在其中一个文件中写入 `I'm learning git.`（写入内容您可以按需填写）这句话并保存。

流程示意图

在正式进行代码提交前，您可以参考此流程示意图了解 Git 在追踪文件时所需经历的状态周期。



跟踪文件 (git add)

创建或修改文件后调用 `git add` 命令，将变更的文件添加至本地 Git 仓库的暂存区（Index Stage）。

追踪特定文件时的命令：

`git add readme.txt`

添加多个文件的命令：

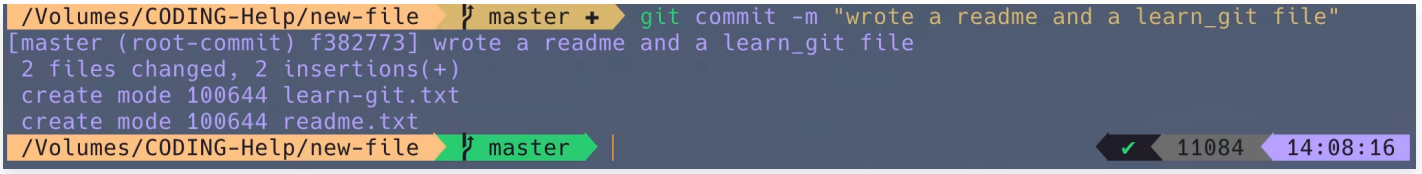
`git add readme.txt learn_git.txt`

如果您想一次性**跟踪所有文件**，则可以直接在终端输入 `git add .`

提交文件 (git commit)

将拟提交的文件纳入暂存区后，运行 git commit 命令即可将文件正式提交至本地仓库，此命令将会一次性会提交暂存区中的所有文件：

`git commit -m "wrote a readme and a learn_git file"`



`-m` 后引号中的内容是您的提交说明，下面几行是终端的返回结果。每次提交文件时附上变更说明，可以清楚地把控提交了什么样的修改。

除了 `git commit` 命令，您还可以使用标准化插件规范仓库中的提交信息，方便后期回溯。

```
// 第一步：安装 yarn

brew install yarn

// 第二步：安装插件

yarn add -D standard-version

// 第三步：使用插件提交代码

git cz
```

自动关联事项

提交代码时在提交信息中附上 `# 事项 ID` 还可以直接与项目内的事项相关联，在事项内即可查看相关代码提交记录。

例如：事项（类型需求）ID 为 630，在 commit message 中添加 `#630` 即可自动与该需求关联；添加 `project-1#630` 可以关联 `project-1` 项目中的 630 事项。

关联事项后，将鼠标移至提交记录的 ID 上可以预览事项。

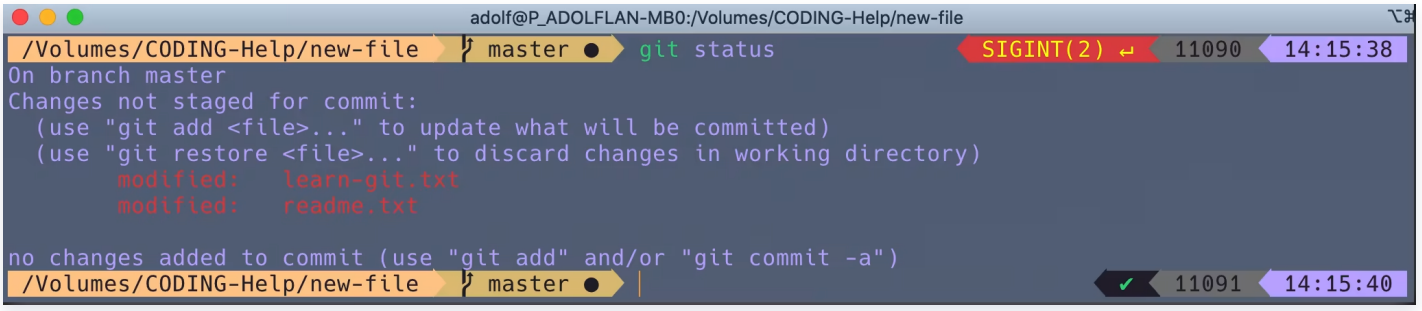
查看文件状态 (git status)

使用 `git status` 命令查看文件状态。通过此命令您可以确认 Git 是否精准的追踪了修改过的文件及文件处于哪种状态。

当前仓库里任何文件都没有被跟踪时返回结果如下：

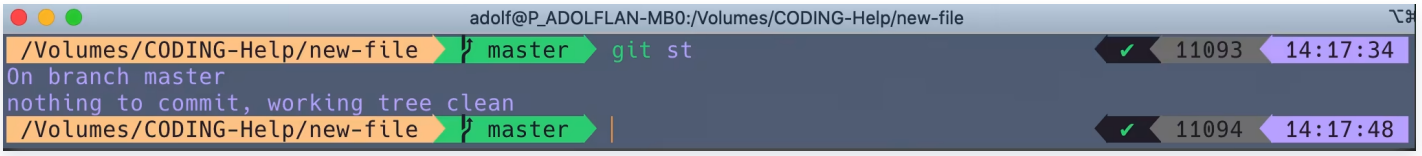
```
$ git status
On branch master
Your branch is up-to-date with 'origin/master'
nothing to commit, working directory clean
```

当文件有变更，但没有被跟踪时返回结果如下：



此时运行 `git add` 命令即可跟踪文件，跟踪完成后字体颜色由红色变为绿色。

当文件已跟踪且已经提交到仓库时，返回结果如下：



推送文件到远程仓库 (git push)

在终端运行命令：

`git push`

若希望在提交时自动创建合并请求并关联项目，可以使用以下命令：

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

`git push` 是推送命令，实际上是把本地的分支推送到了远程仓库，在远程有了一个备份。前往 CODING 代码仓库即可查看推送上来的文件。若多人共同协作维护此远程代码仓库，待他人提交代码后，您只需要在本地运行 `git pull` 命令即可使本地与远端保持代码同步。

查看或编辑远程仓库

文件被推送至 CODING 代码仓库后，您可以在网页上进行代码编辑、保存提交等操作。

以 `README.md` 文件为例，完成编辑并提交后，可以简短描述此次修改内容。若不填写，默认的提交说明是“更新文件 xxx”。


提交说明
描述
更新文件 README.md
添加额外的说明
直接提交到 master 分支
为这次提交创建一个新分支，并发起一个合并请求
确认
取消

权限配置

最近更新时间：2024-08-19 11:48:11

根据代码仓库的使用场景与功能定位，权限类型包含“团队权限方案”、“项目权限组”与“单仓库权限”三种类型。

团队权限方案

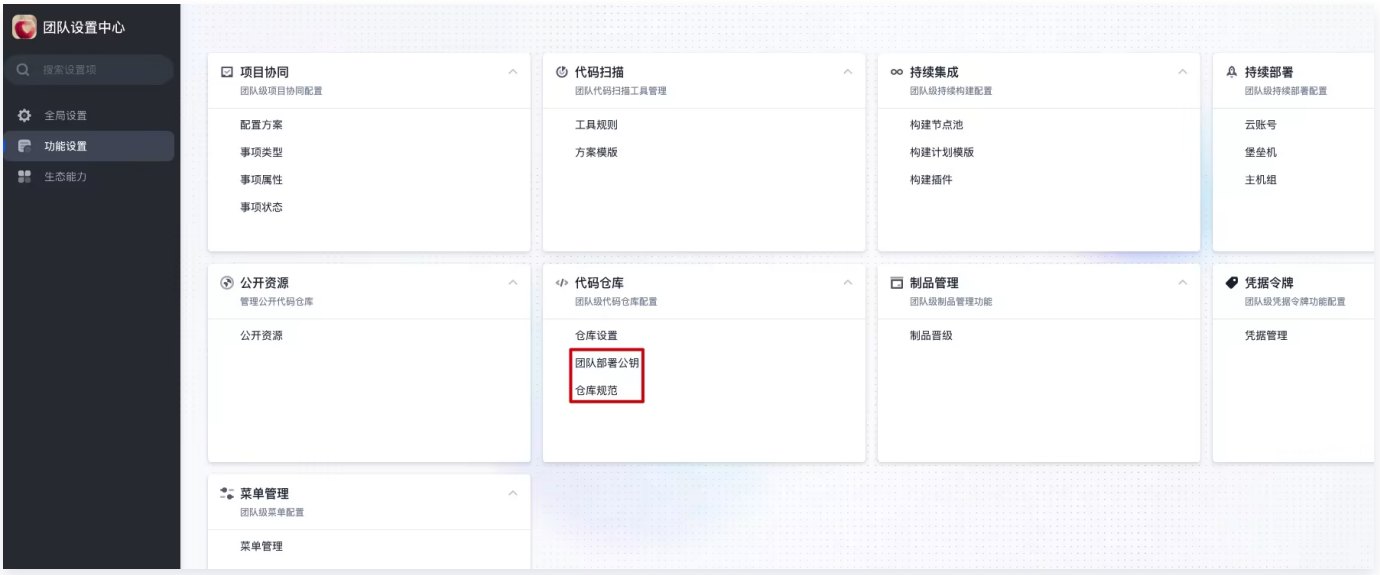
仓库设置、团队部署公钥、团队仓库规范归属于团队级权限管理。由团队所有者 / 管理员前往团队设置中心的“团队权限方案”进行调整。

- 仓库设置

  位于具备此权限点分组中的成员可以访问项目中所有仓库中的**设置**页。

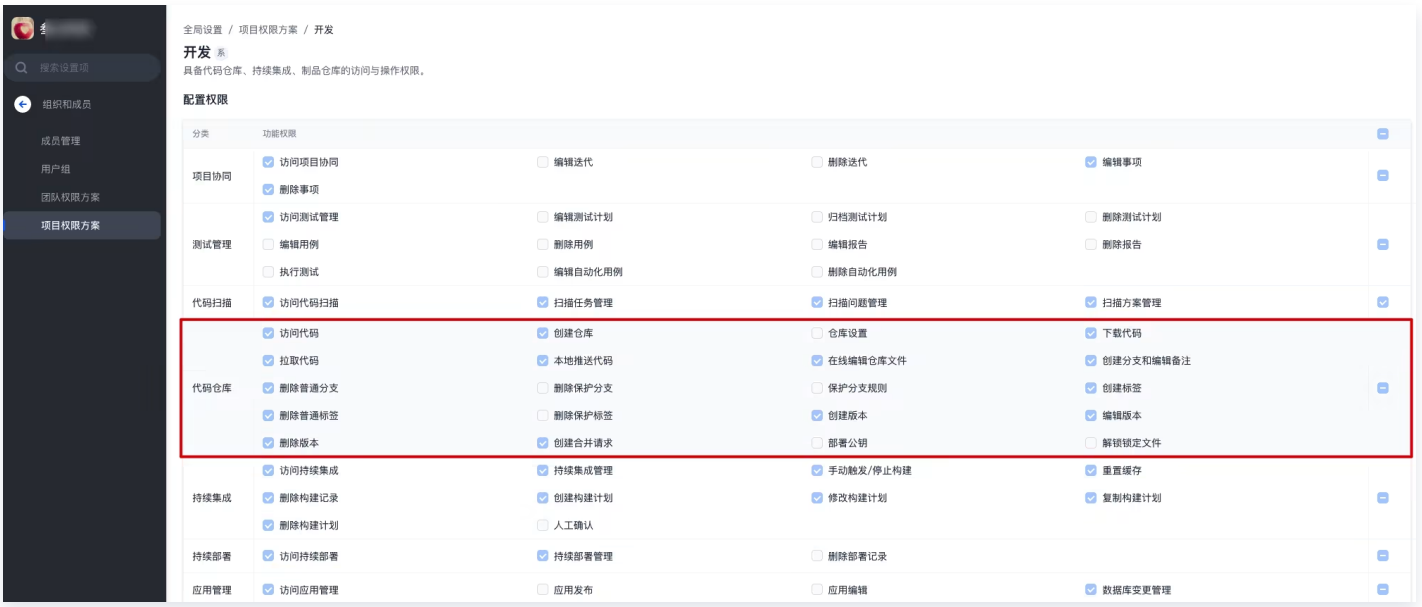
- **团队部署公钥与仓库规范**

  位于具备此权限点分组中的成员可以访问功能设置中的团队部署公钥与仓库规范。

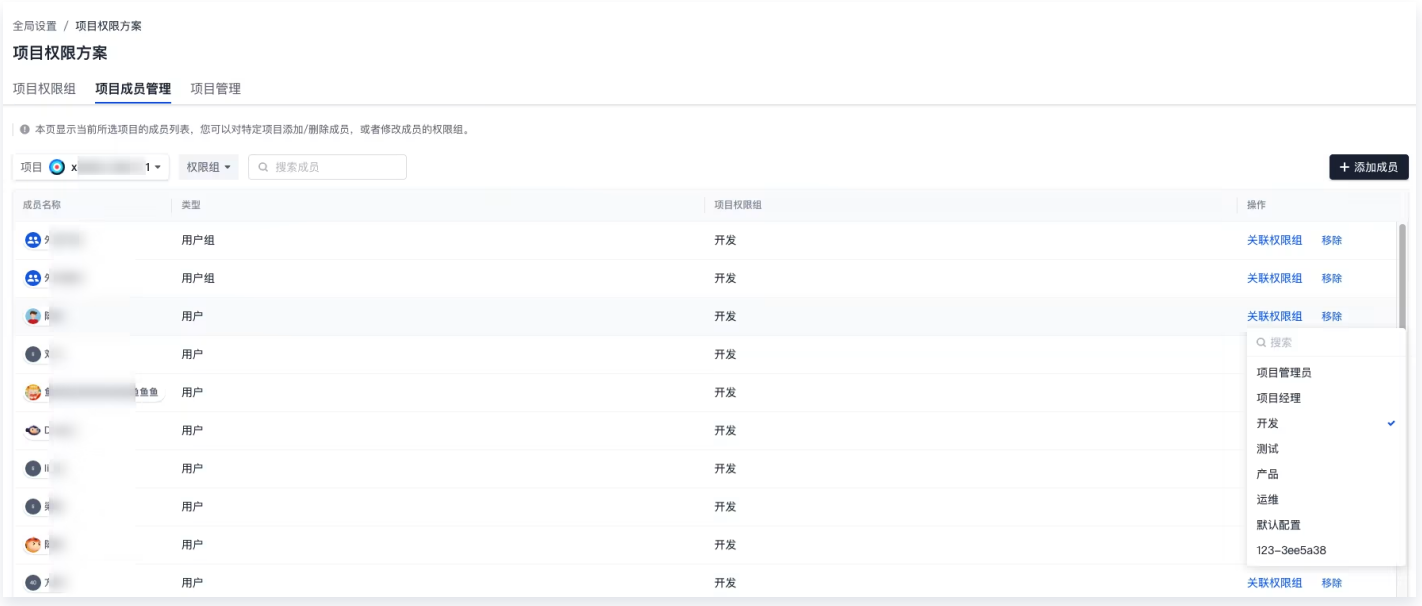


项目权限方案

项目权限组用于控制项目成员使用代码仓库协作的各项功能，包含保护分支、标签与版本创建等各项权限点。

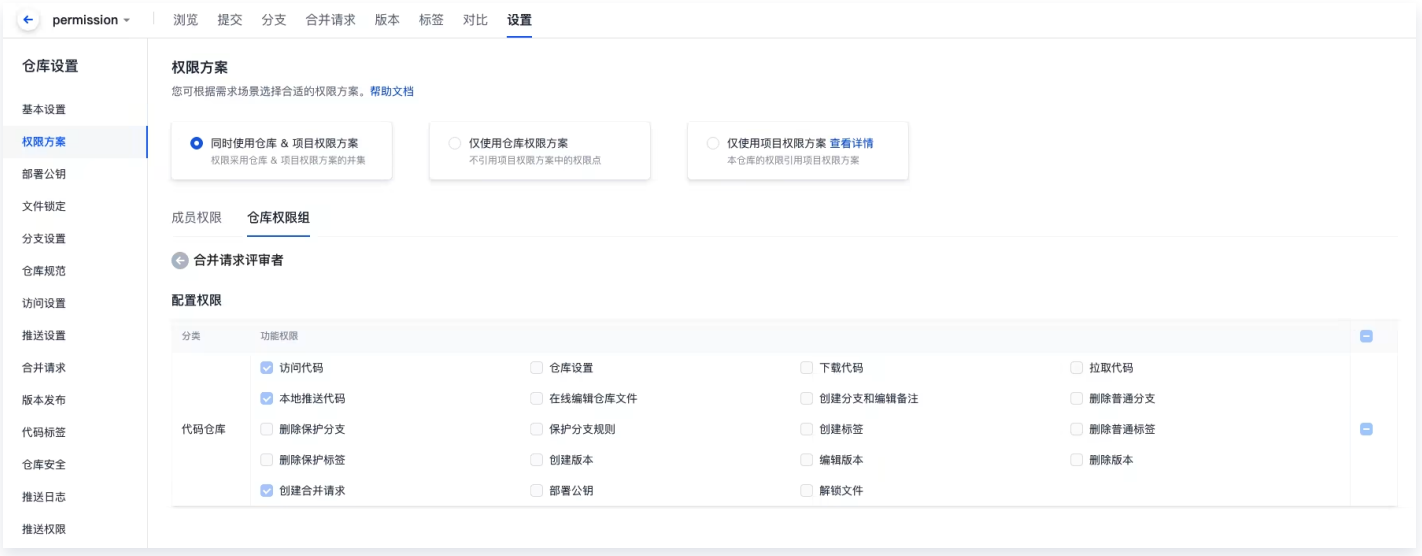


团队负责人 / 管理员完成项目权限组配置后，即可前往**全局设置 > 项目权限方案 > 项目成员管理**页中为项目成员分配合适的权限方案。



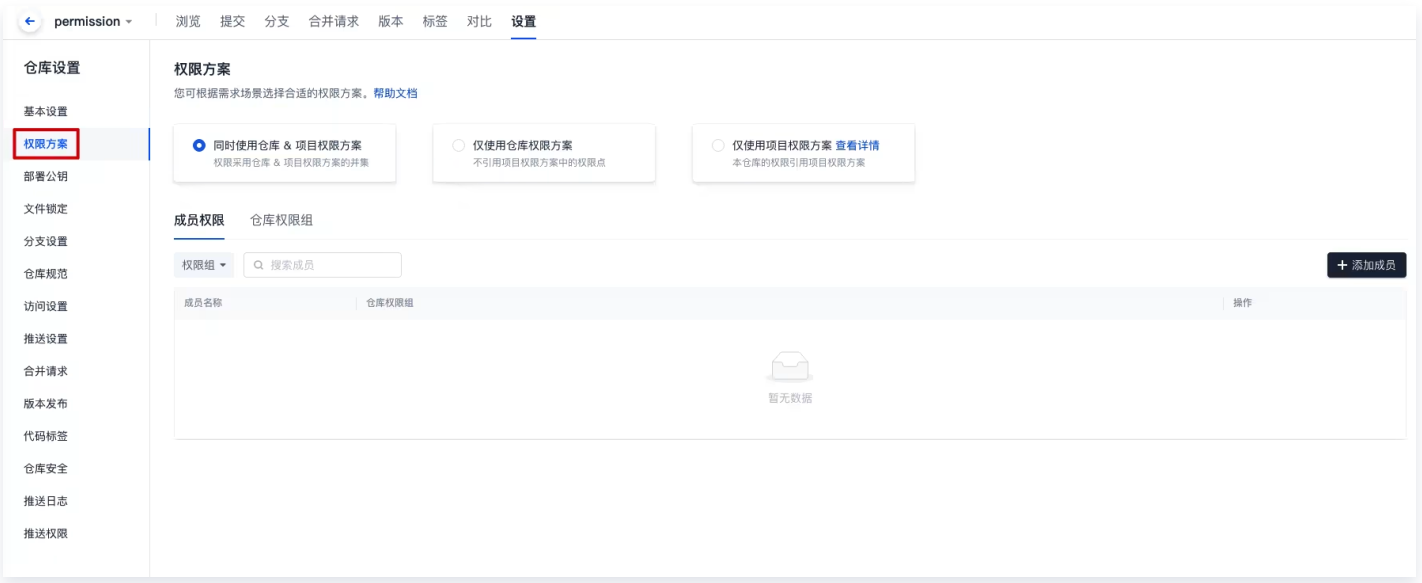
仓库权限方案

除了采用项目权限方案，还支持在单个代码仓库内配置仓库权限方案，满足同一个项目下不同代码仓库间的开发流程安全性要求，详细配置说明请参见 创建资源权限方案组 。



指定权限方案类型

代码仓库权限组创建完成后，仓库管理员前往仓库设置中调整“权限方案”。



> **说明：**
> 仓库管理员指的是用户所在项目权限方案具备“仓库设置”权限点。

为了区分所应用的权限方案，仓库管理员在添加成员前需根据需求场景选择合适的权限方案。

- **同时使用项目权限 & 仓库方案**

  兼顾仓库成员的所属权限方案与仓库权限方案。例如 A 成员所属的项目权限方案具备“删除普通分支”权限，但代码仓库权限不具备“删除普通分支”权限，此时 A 成员具备“删除普通分支”权限。

- **仅使用仓库权限方案**

  放弃项目权限方案，以仓库成员所属的仓库权限方案为准。例如 A 成员所属的项目权限方案具备“删除普通分支”权限，但代码仓库权限不具备“删除普通分支”权限，此时 A 成员不具备“删除普通分支”权限。

> **注意：**
> 若选择“仅使用仓库权限方案”，仅团队负责人 / 管理员或团队中其他拥有代码权限的成员有权进入仓库页进行操作。

- **仅使用项目权限方案**

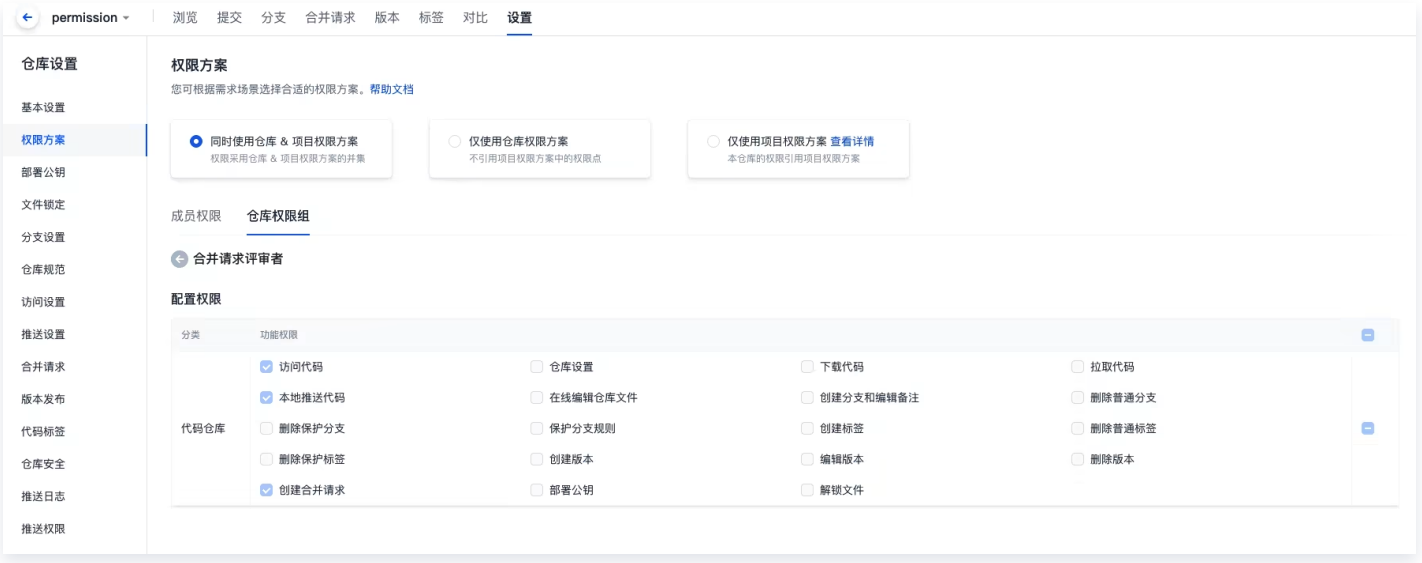
  放弃仓库权限方案，以仓库成员所属的项目权限方案为准。例如 A 成员所属的项目权限方案具备“删除普通分支”权限，但代码仓库权限不具备“删除普通分支”权限，此时 A 成员具备“删除普通分支”权限。

分配代码仓库权限

添加成员后为其分配代码仓库权限组。

查看代码仓库权限组

切换至**仓库权限组**页签，您可以看到当前权限组中包含的权限点。



若所属的权限分组缺少您需要的权限点，请联系团队管理员参见 配置资源权限方案 来添加相应的权限点。

Git 仓库
创建仓库

最近更新时间：2023-09-11 16:05:17

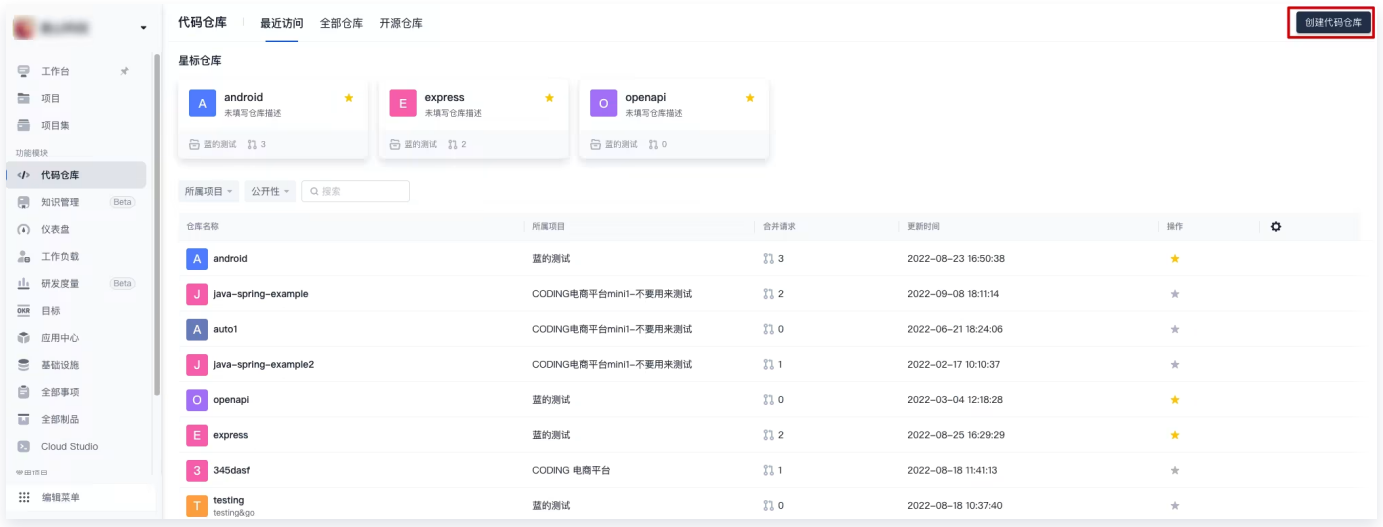
本文为您详细介绍如何创建 Git 代码仓库。

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. 单击团队首页左侧的**项目**，进入项目列表页，选择目标项目。
3. 选择左侧菜单**代码仓库**，进入代码仓库首页。

手动创建仓库

1. 登录团队后，单击首页左侧的**代码仓库**中的**创建代码仓库**进行创建。



2. 代码仓库无法独立存在，需从属于某个项目。两者的对应关系为一个项目对应多个代码仓库。

3. 仓库类型选择 Git，输入符合条件的仓库名称。

4. 创建代码仓库时还可以选择是否启用仓库规范，用以快速制订团队开发工作流。

模板创建仓库

CODING 提供数个预置开发框架的代码仓库模块，助您快速体验代码仓库时如何与持续集成或构建产物进行关联。

导入外部仓库

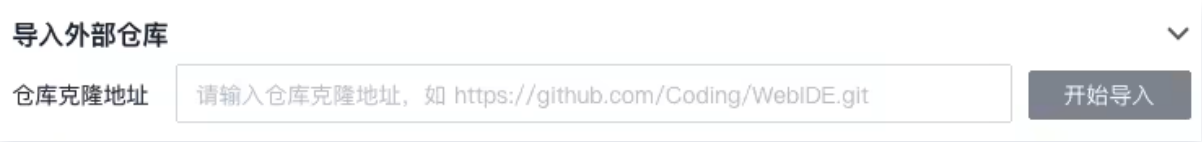
您可以将开源 Git 仓库或其他平台的仓库快速迁移至 CODING DevOps 平台，详情请参见 导入或关联外部仓库 。

初始化 Git 仓库

创建仓库后可以通过四种方式进行初始化。

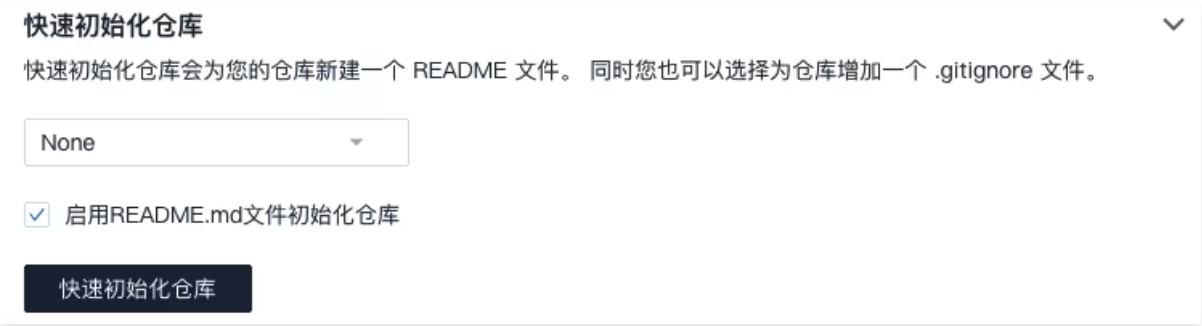
导入外部仓库

输入 Git 仓库地址后进行导入，完成初始化。



快速初始化仓库

此方法将通过生成 README.md 文件完成初始化。



使用命令行创建仓库

此方法本质上是在本地仓库中生成一个 README.md 文件后上传至远端仓库完成初始化。

使用命令行推送本地仓库

您也可以直接将本地已初始化的仓库上传至远端仓库中。

导入或关联外部仓库

最近更新时间：2023-10-12 18:53:41

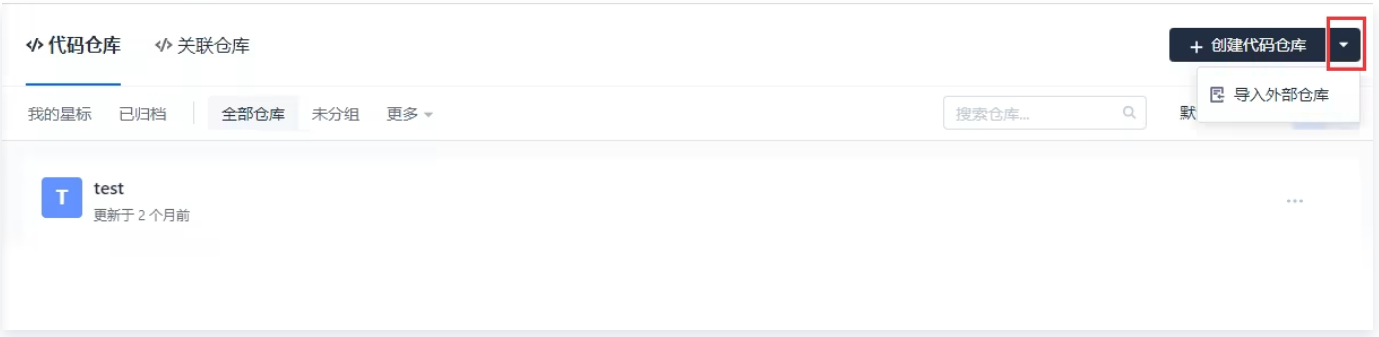
进入项目

1. 登录 CODING 控制台，单击团队域名进入 CODING 使用页面。
2. 单击团队首页左侧的**项目**，进入项目列表页，选择目标项目。
3. 选择左侧菜单**代码仓库**，进入代码仓库首页。

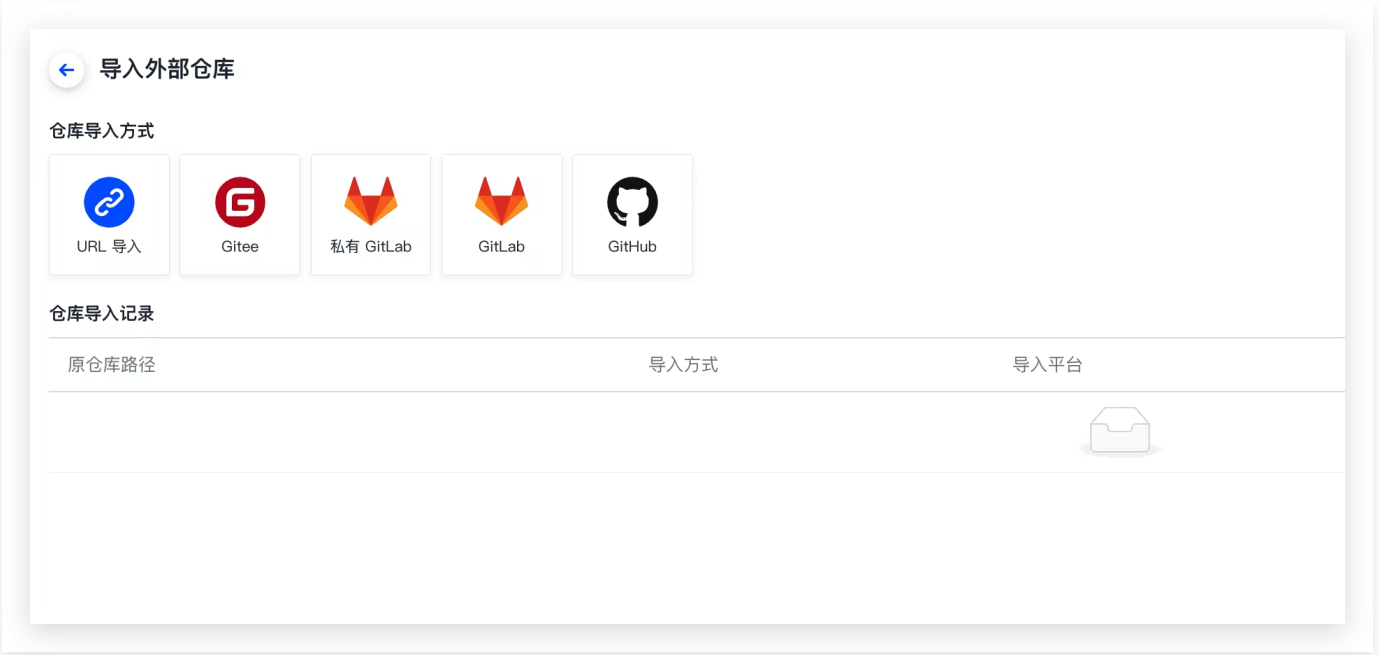
导入仓库

CODING 代码仓库不仅提供了一键导入外部仓库功能，还支持定时同步外部开源仓库。

1. 前往任意项目，单击左侧栏的**代码仓库**，单击页面右上角的**导入外部仓库**。



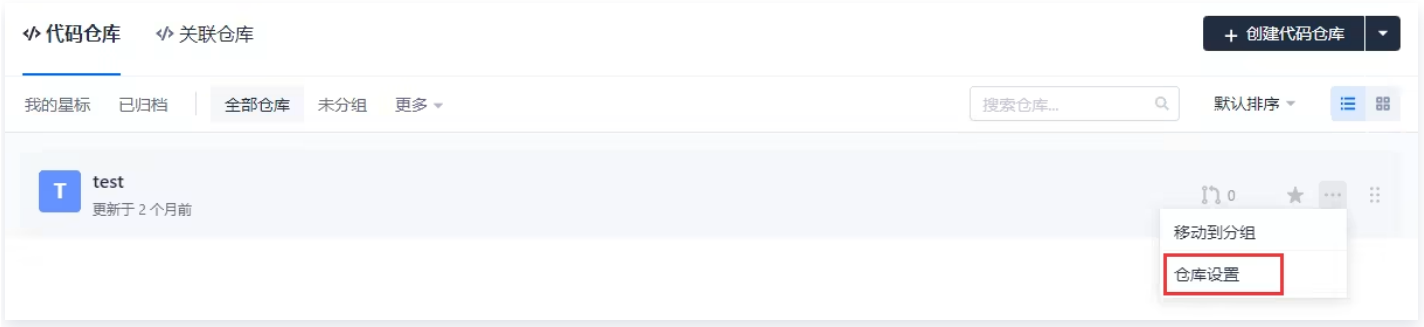
2. 按照提示选择仓库来源类型。



3. 您可以选择通过 URL 导入私有或开源仓库。若源仓库类型为私有，那么需按照页面提示填写用户名与密码。

同步仓库

同步功能仅对开源仓库开放。这意味着与源仓库保持一致，将覆盖导入 CODING 仓库后做出的变更。单击**仓库设置**可以修改同步频率或关闭自动同步功能。



在已导入仓库的首页中还可以通过单击导入按钮手动强制同步代码仓库。

导入镜像仓库

若出于安全原因考虑，您也可以选择导入外部私有（非开源）的 Git 镜像仓库，参见以下操作步骤。

步骤1：将私有仓库拉取至本地

```
git clone --mirror 私有仓库地址
```

```
/Volumes/CODING-Help git clone --mirror git@gitlab.com:[redacted]ios-go.git
Cloning into bare repository 'ios-go.git'...
remote: Enumerating objects: 53, done.
remote: Total 53 (delta 0), reused 0 (delta 0), pack-reused 53
Receiving objects: 100% (53/53), 2.31 MiB | 1.68 MiB/s, done.
Resolving deltas: 100% (9/9), done.
```

> **说明：**
> 拉取至本地的镜像仓库文件夹通常带有 `.git` 后缀。

步骤2：获取目标 CODING 代码仓库地址

在 Web 端进入 CODING 项目，获取目标仓库的地址。

步骤3：推送至目标 CODING 仓库

在终端中输入命令进入私有仓库。

```
cd 私有仓库
```

使用推送命令将私有仓库推送至目标 CODING 仓库。

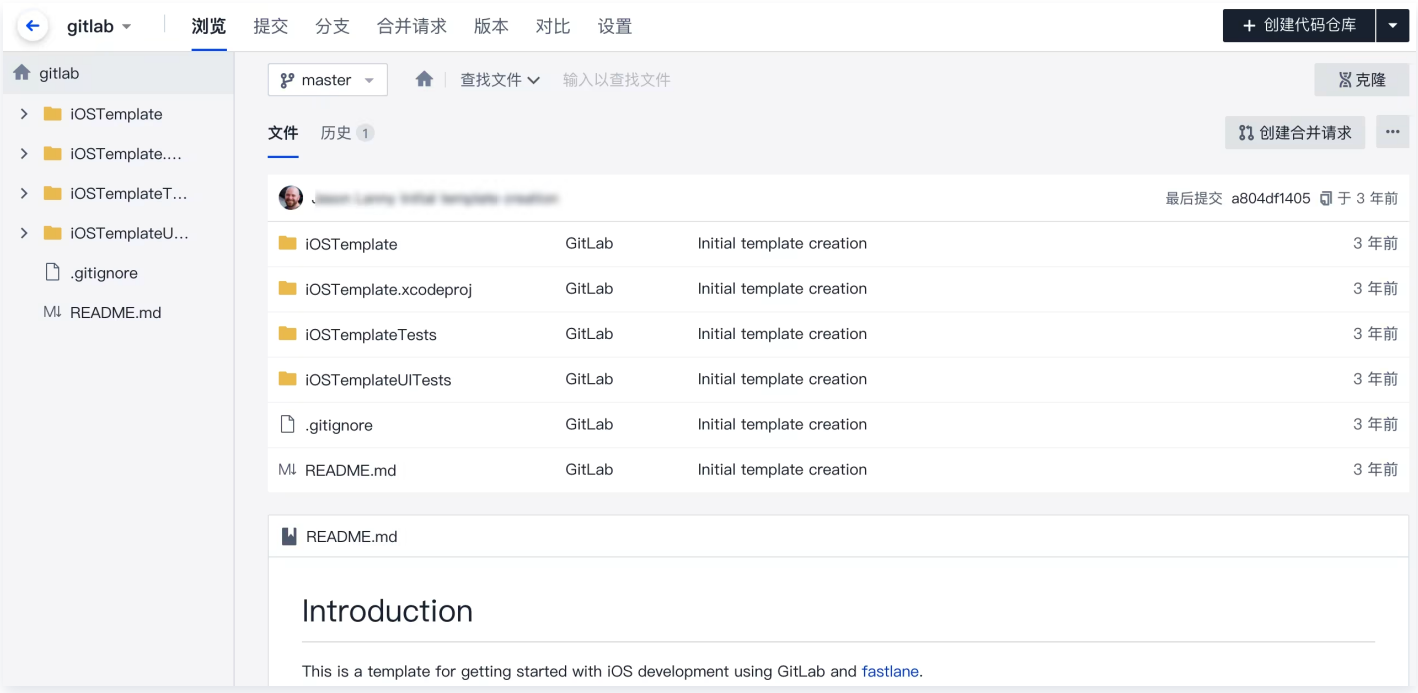
```
git push --mirror 目标仓库地址
```

```
/Volumes/CODING-Help/ios-go.git  master  git push --mirror gi                    s/d
emo/gitlab.git
Enumerating objects: 53, done.
Counting objects: 100% (53/53), done.
Delta compression using up to 12 threads
Compressing objects: 100% (43/43), done.
Writing objects: 100% (53/53), 2.31 MiB | 2.23 MiB/s, done.
Total 53 (delta 9), reused 53 (delta 9)
To e.coding.net:StrayBirds/demo/gitlab.git
 * [new branch]      master -> master
```

若出现 `refs/pull` 报错，可以使用以下命令避免报错：

```
git push URL "+refs/heads/*:refs/heads/*" "+refs/tags/*:refs/tags/*"
```

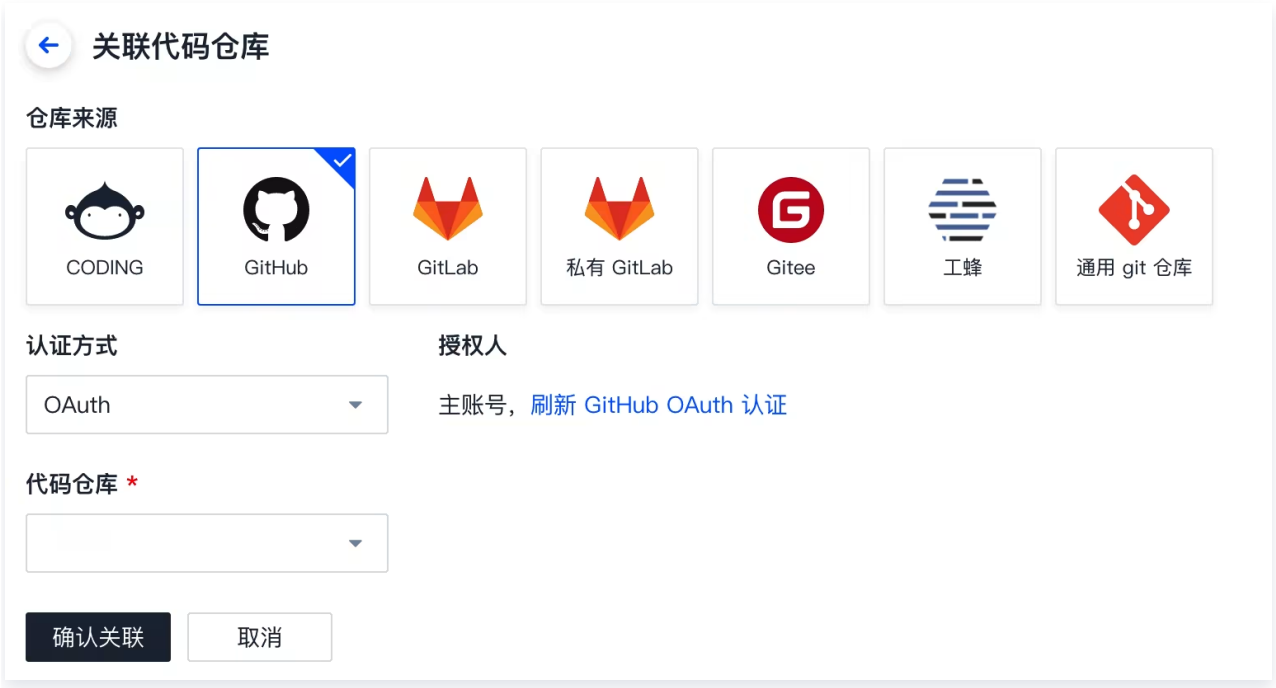
推送成功后，可以看到的私有仓库已上传至目标仓库中。



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。

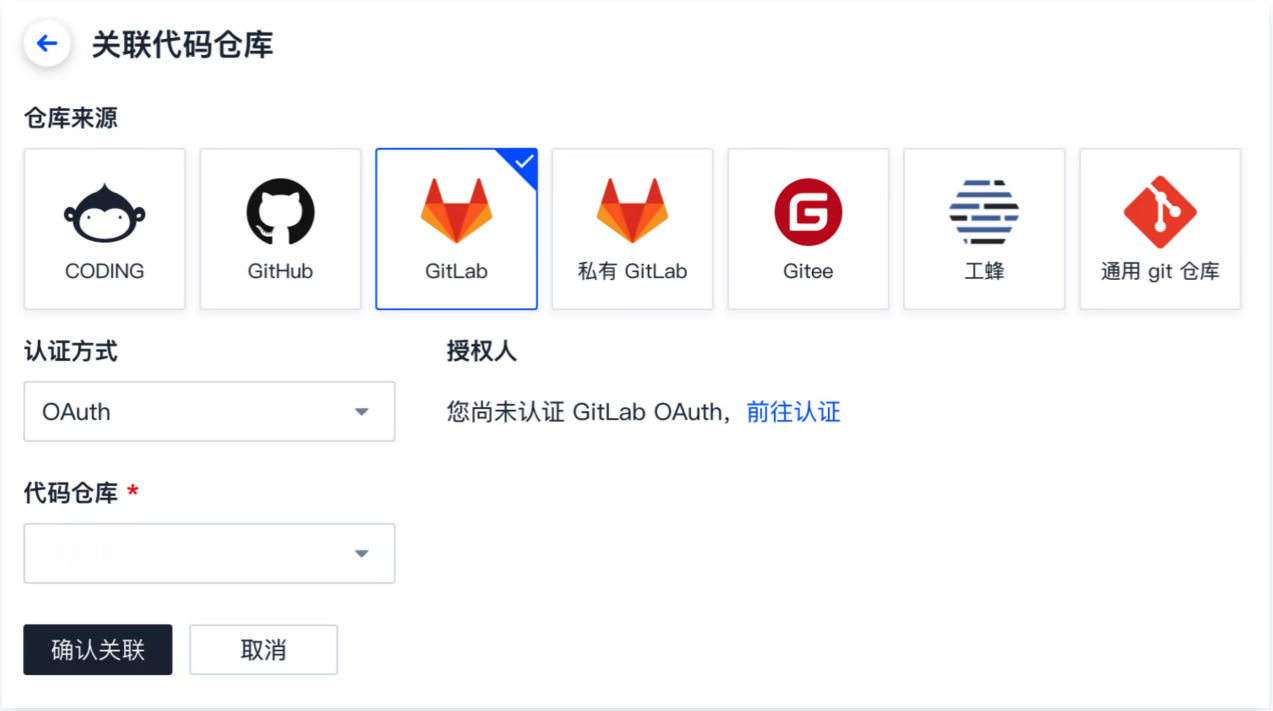


关联 GitLab 私有仓库

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。

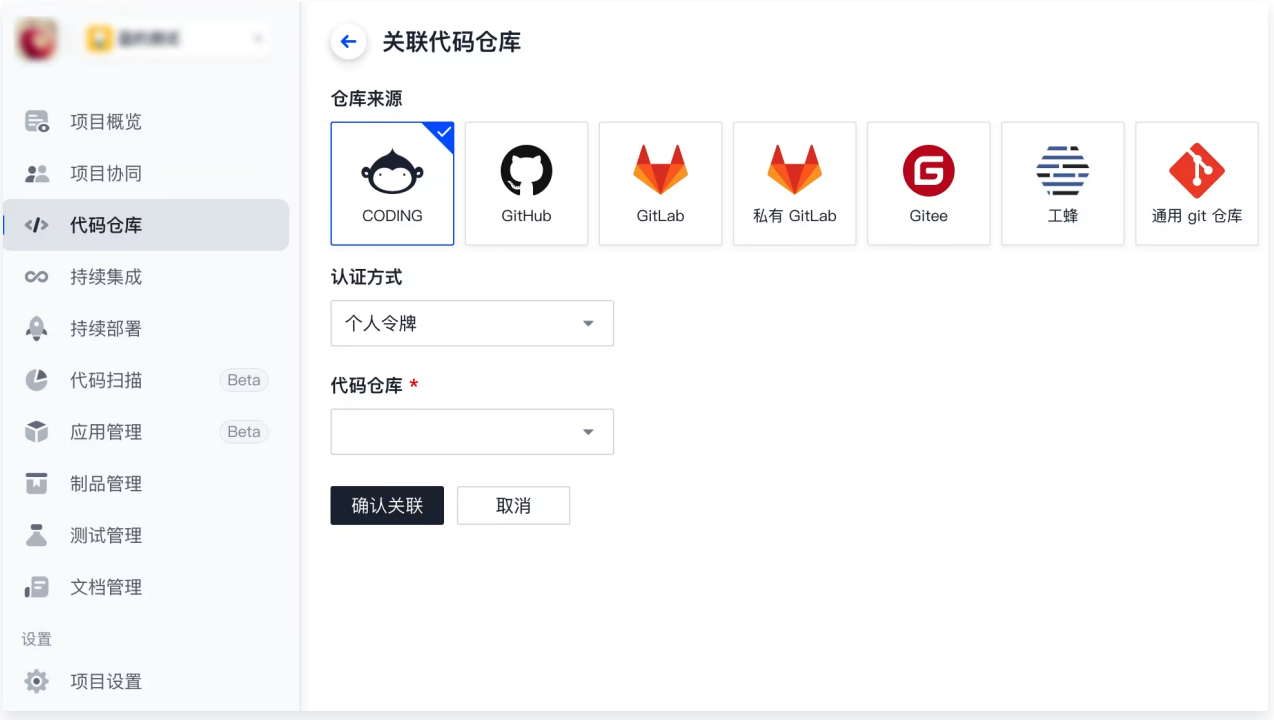


关联 GitHub 仓库

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。

关联其他项目的代码仓库

若希望关联其他 CODING 项目中本人名下的代码仓库，需前往**关联代码仓库**页面，仓库来源选择 CODING，勾选目标仓库后进行关联。关联后可用作持续集成或持续部署中的代码源。

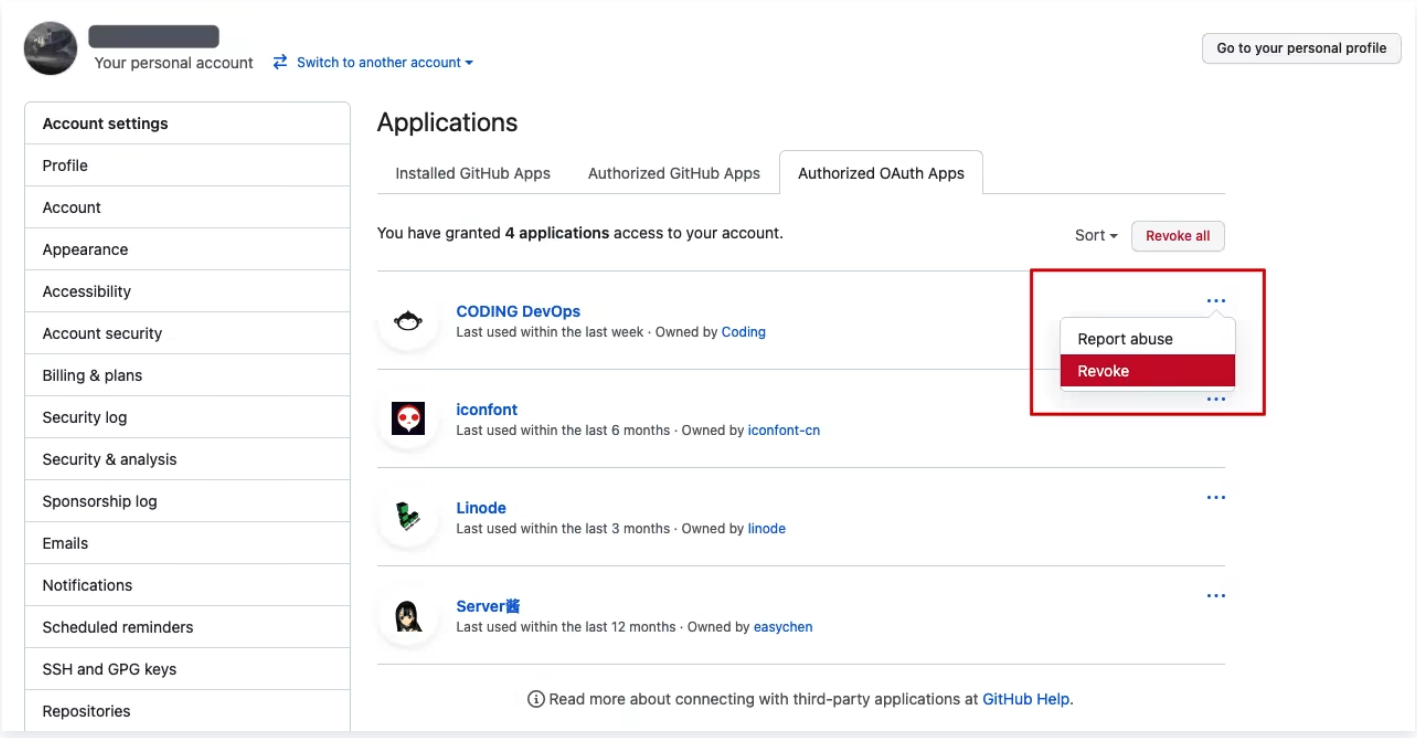


若希望将仓库直接导入至项目中进行修改，请前往目标仓库的**访问设置**并将其设置为开源状态，然后参见 导入开源仓库 导入目标代码。

修改关联仓库

本章节以 GitHub 为例：

登录 GitHub 后单击右上角头像 **Settings** > **Applications** > **Authorized OAuth Apps** 取消关联。



切换为其他的 GitHub 账号重新进行 OAuth 认证完成 关联 GitHub 仓库 。

管理仓库

最近更新时间：2023-09-11 16:05:17

本文为您详细介绍如何在仓库中执行部分操作。

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. 单击团队首页左侧的**项目**，进入项目列表页，选择目标项目。
3. 选择左侧菜单**代码仓库**，进入代码仓库首页。

清理仓库

远程仓库在使用的过程中会产生缓存文件，您可以单击**仓库清理**减少缓存文件对仓库容量的占用。

归档仓库

如需归档仓库，在代码仓库列表单击 ··· 并按照提示确认即可。

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。

删除与重置

如需重置或删除仓库，在代码仓库列表单击 ··· 并按照提示确认即可。

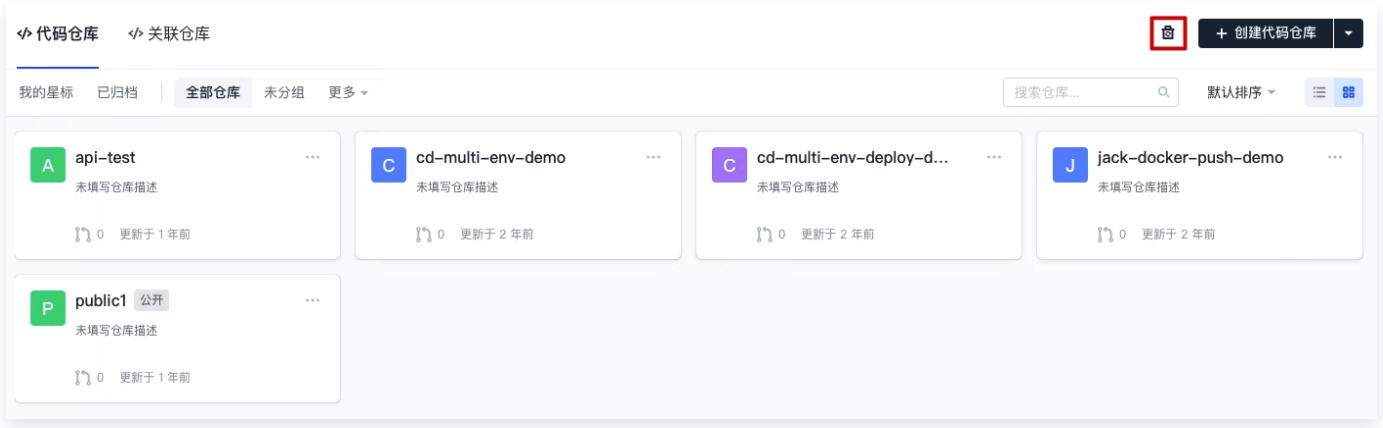
删除后的代码仓库将移入回收站保留 30 天，项目管理员可在到期前进入回收站恢复。逾期将会彻底删除当前代码仓库内的所有代码，包括代码分支、合并请求、代码版本，数据无法恢复。

单击重置仓库后，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，并且仓库也会被重置为空仓库。

恢复已删除仓库

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，单击右上角回收站图标。



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。

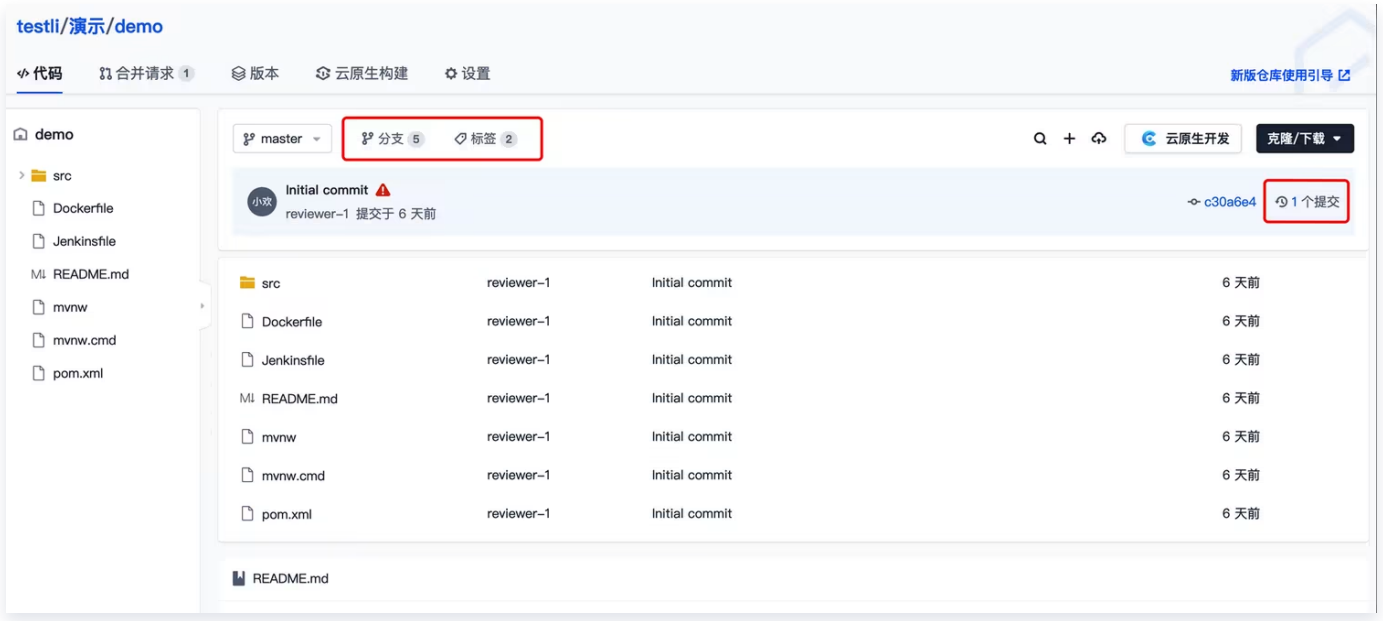
代码对比

最近更新时间：2023-09-11 16:05:17

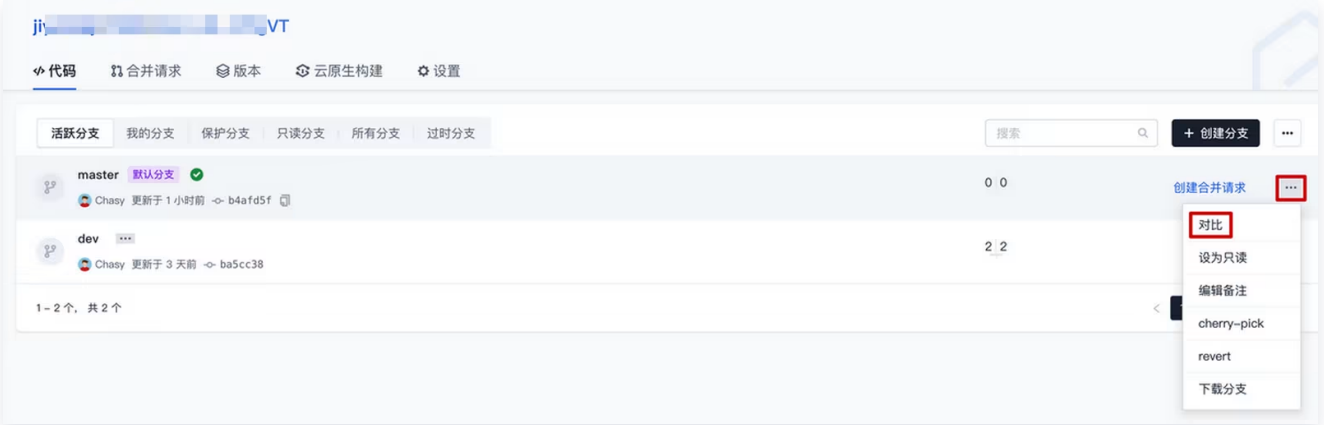
本文为您详细介绍如何使用代码对比功能。

操作步骤

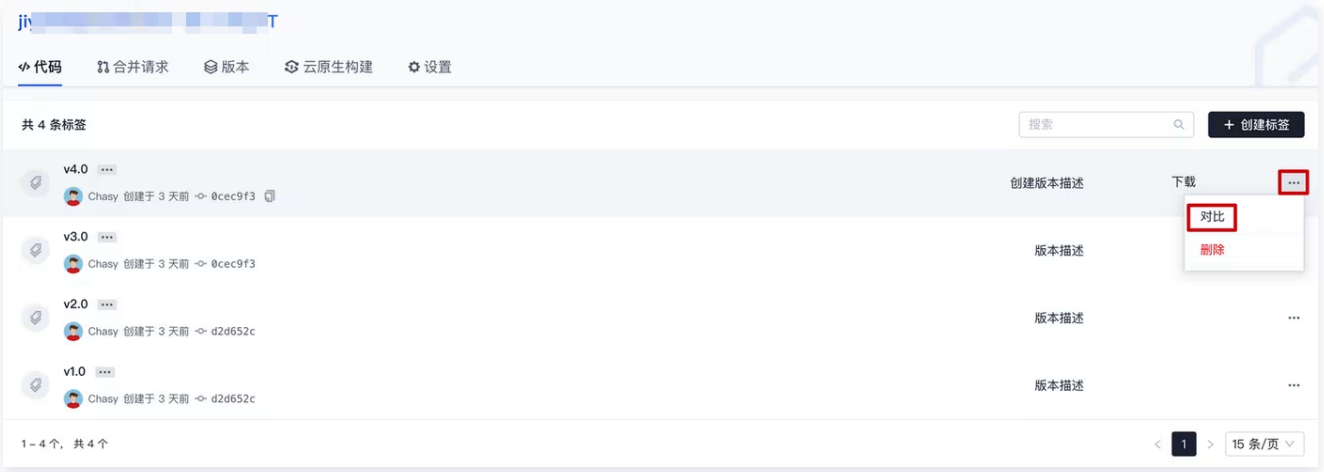
1. 登录 CODING 控制台，单击团队域名进入 CODING 使用页面。
2. 单击团队首页左侧的**项目**，进入项目列表页，选择目标项目。
3. 选择左侧菜单**代码仓库**，进入代码仓库首页。
4. 进入指定代码仓库，可单击**分支**、**标签**或修订版本（哈希值）进入对应页面。



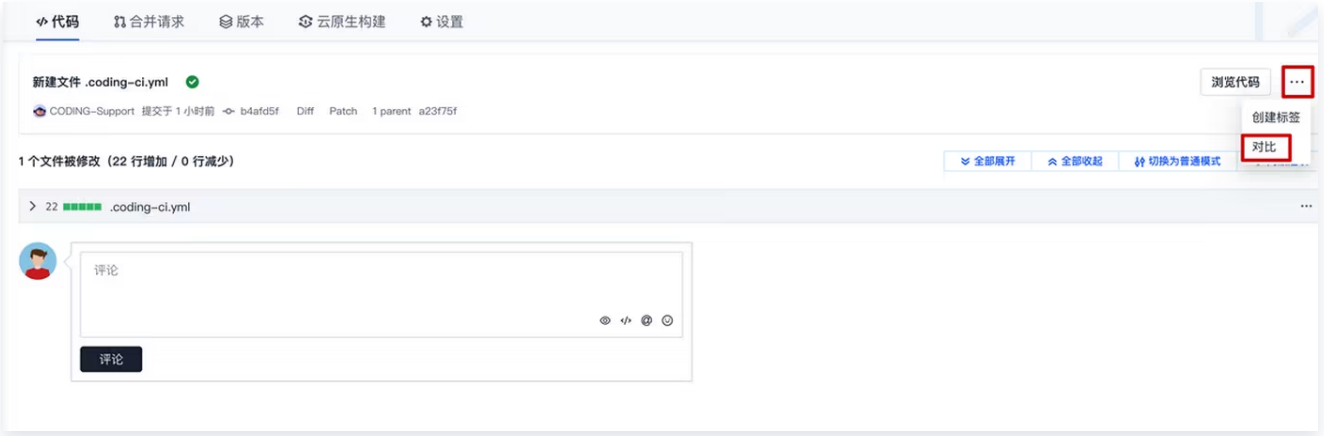
5. 在对应页面中单击 ··· 并选择**对比**。
   - 以分支列表为例，选择指定分支，单击 ··· 并选择**对比**。



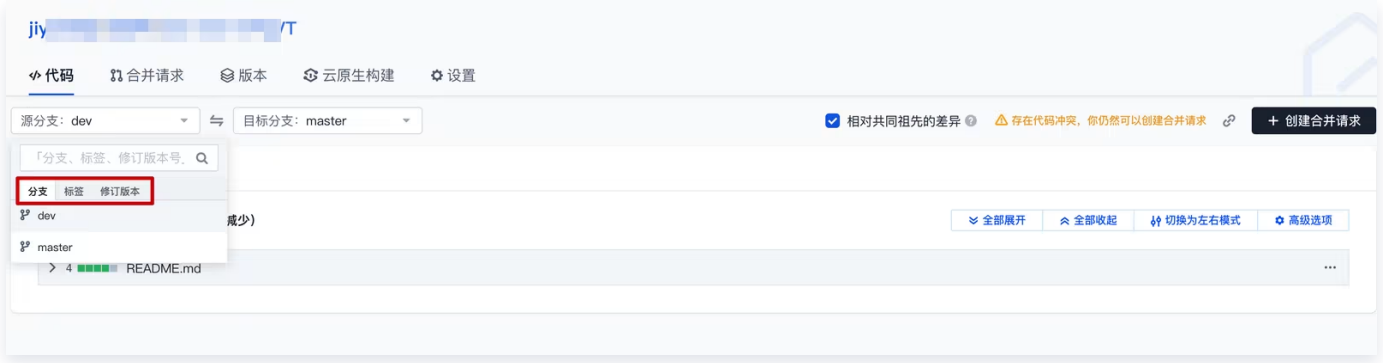
- 以标签列表为例，选择指定标签，单击 ··· 并选择**对比**。



- 如果进入了指定的修订版本（哈希值），在右上角单击 ··· 并选择**对比**。

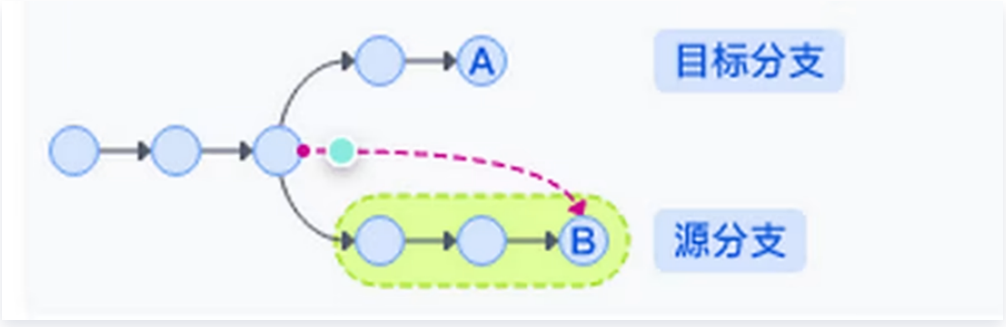


6. 在代码对比页面，可选择对应分支、标签或修订版本进行对比。



功能说明

- 为保证网页响应速度，若超过 20 个文件存在差异将无法使用**全部展开**功能。
- 代码对比功能本质上是 `git diff` 命令的在线图形化表达。若勾选**相对共同祖先的差异**选项，那么将展示源分支与共同祖先间提交的差异。



- 不勾选**相对共同祖先的差异**选项将直接展示源分支与目标分支间的对比。

SVN 仓库
创建 SVN 仓库

最近更新时间：2023-09-11 16:05:17

本文为您详细介绍如何创建 SVN 仓库。

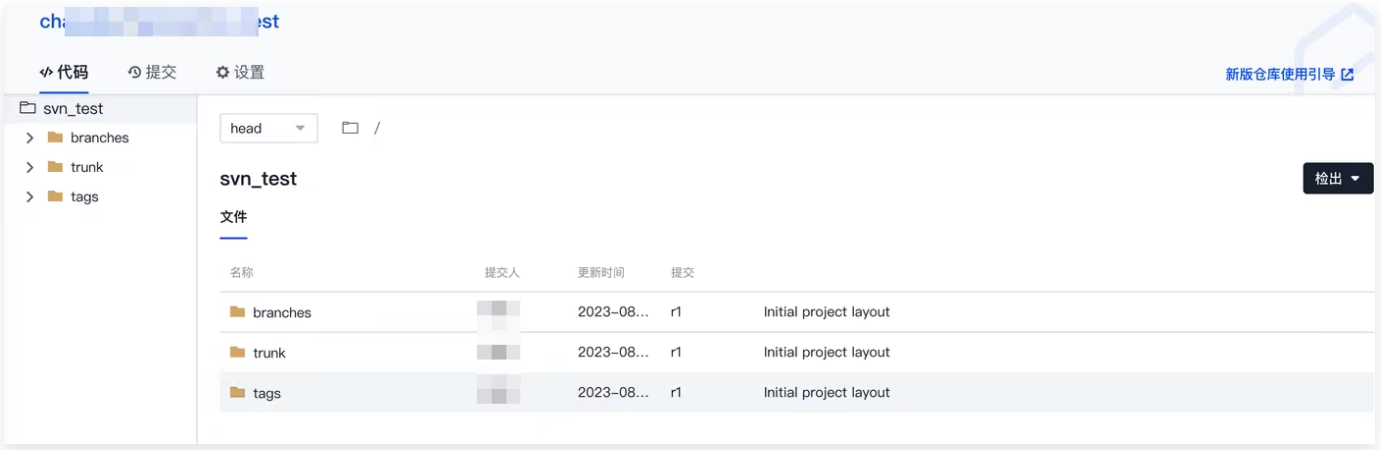
进入项目

1. 登录 CODING 控制台，单击团队域名进入 CODING 使用页面。
2. 单击团队首页左侧的**项目**，进入项目列表页，选择目标项目。
3. 选择左侧菜单**代码仓库**，进入代码仓库首页。

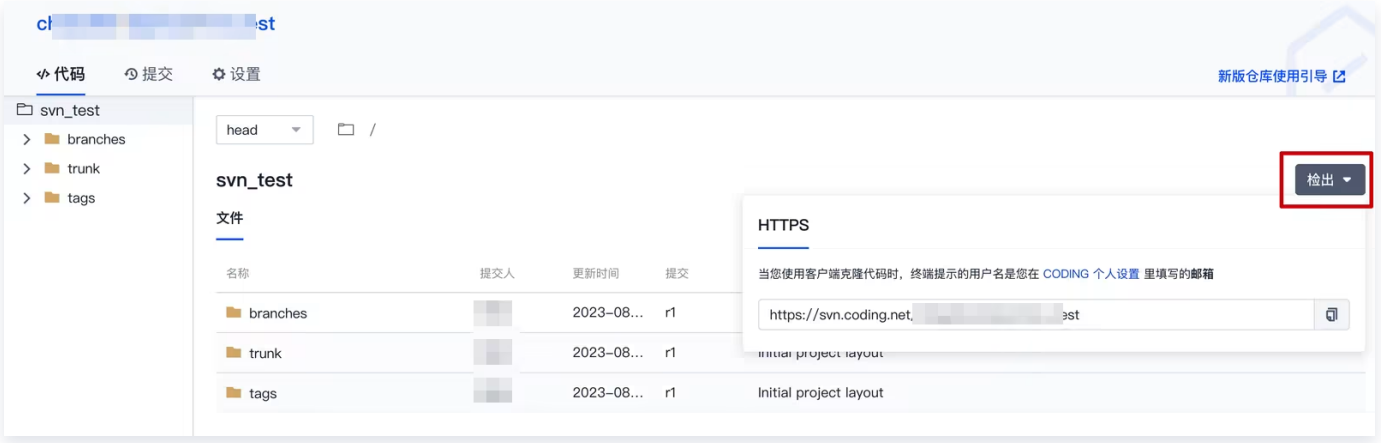
操作步骤

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、trunk 三个目录。这是多数 SVN 仓库的推荐目录布局。
4. 仓库初始化完成之后，即可在**代码**界面看到 SVN 仓库的内容。



5. 单击右上角**检出**可以看到这个仓库的 SVN 地址。



> **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：2024-08-19 11:48:11

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. 单击团队首页左侧的**项目**，进入项目列表页，选择目标项目。
3. 选择左侧菜单**代码仓库**，进入代码仓库首页。

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/master/install)"
```

2. Homebrew 安装好之后，在终端输入以下命令完成 SVN 安装：

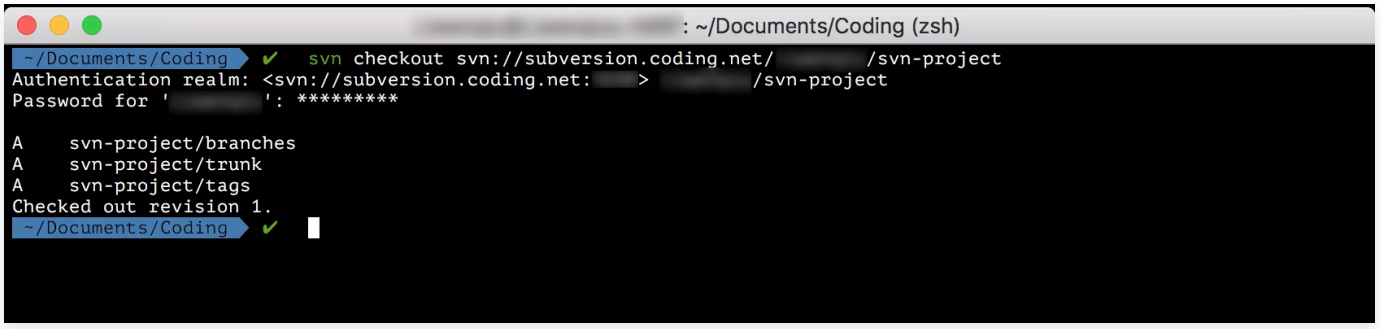
```
brew install subversion
```

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

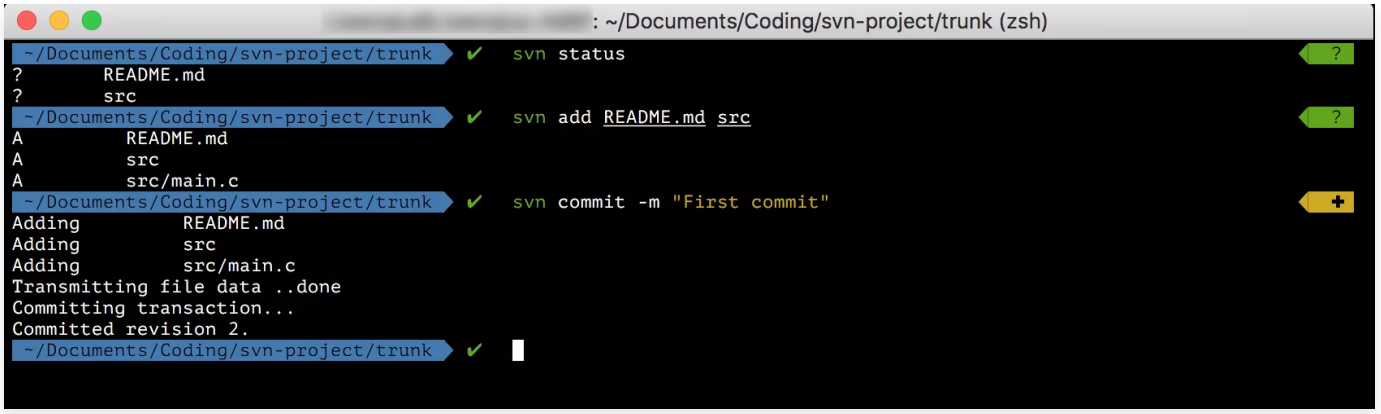
```
svn, version 1.9.7 (r1800392)
compiled Feb 28 2018, 15:54:50 on x86_64-apple-darwin17.3.0
Copyright (C) 2017 The Apache Software Foundation.
This software consists of contributions made by many people;
see the NOTICE file for more information.
Subversion is open source software, see http://subversion.apache.org/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

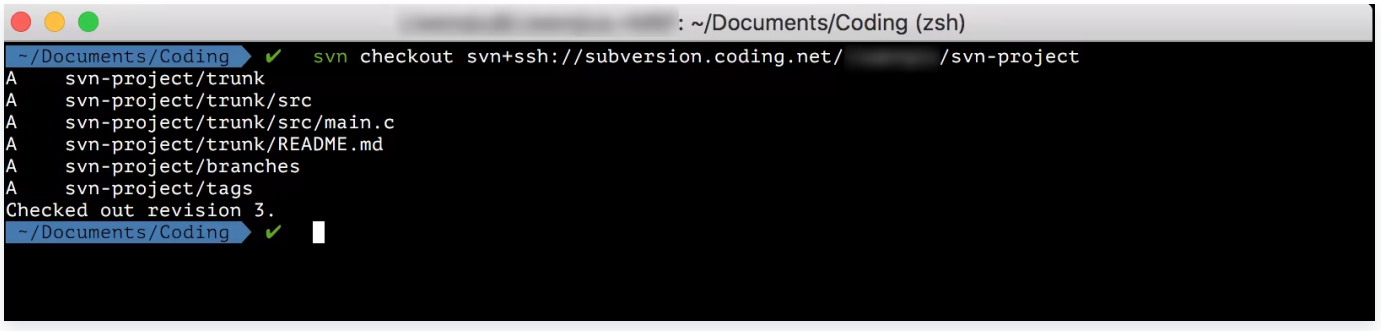
4. 使用命令 `svn checkout svn://subversion.e.coding.net/example/example-project`（请将地址替换为您的 SVN 仓库地址）来检出 SVN 仓库：



5. 接下来可以使用 add、commit 命令往仓库中新添加内容：



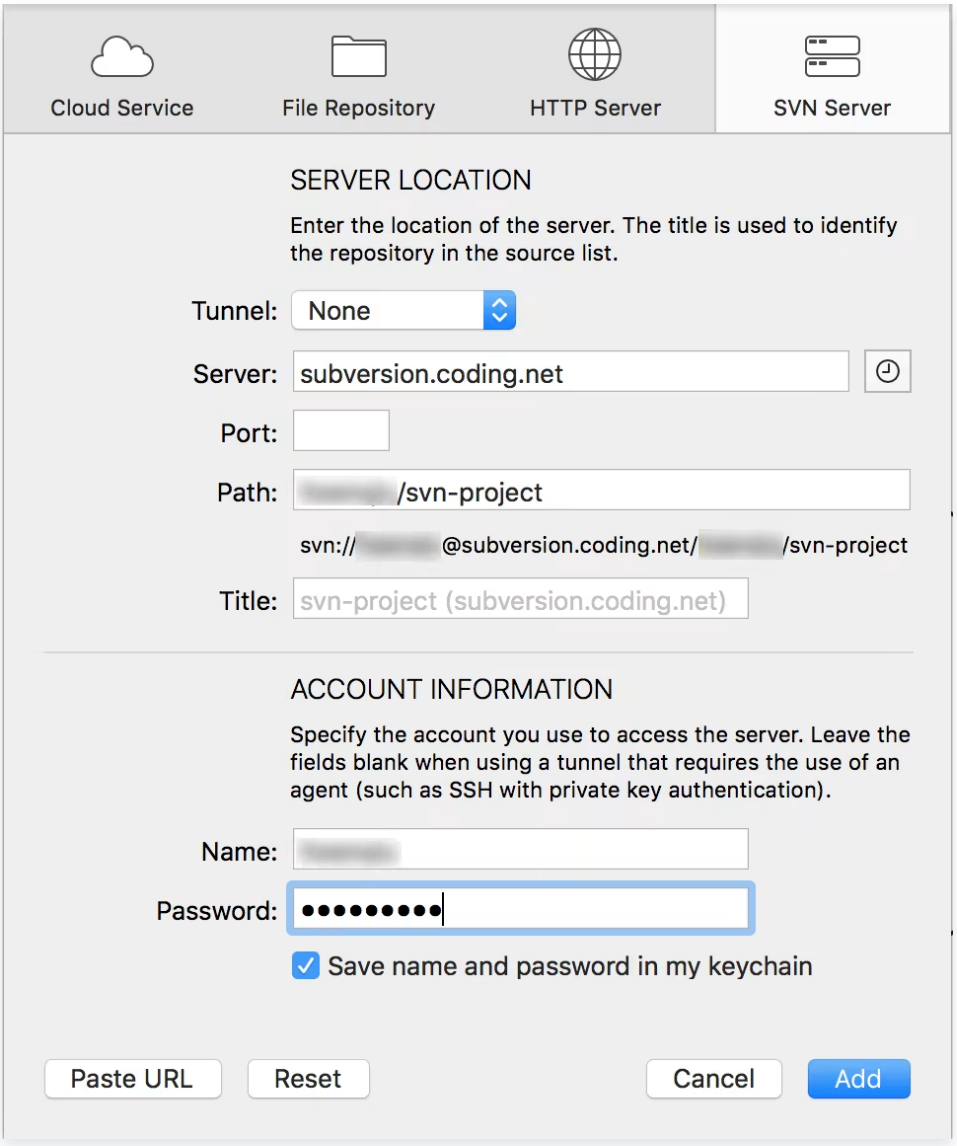
6. 除了使用 SVN 协议之外，还可以使用 `svn+ssh` 协议来访问仓库，如下图所示：



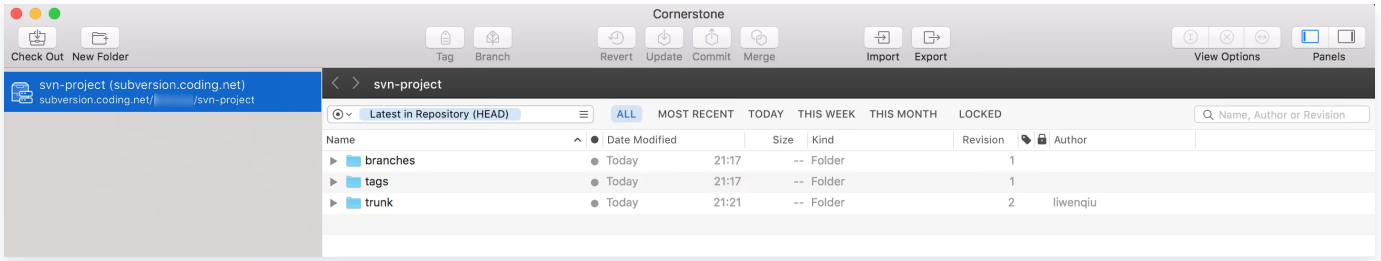
Cornerstone 工具

您可以通过 Cornerstone 来使用 SVN 仓库。

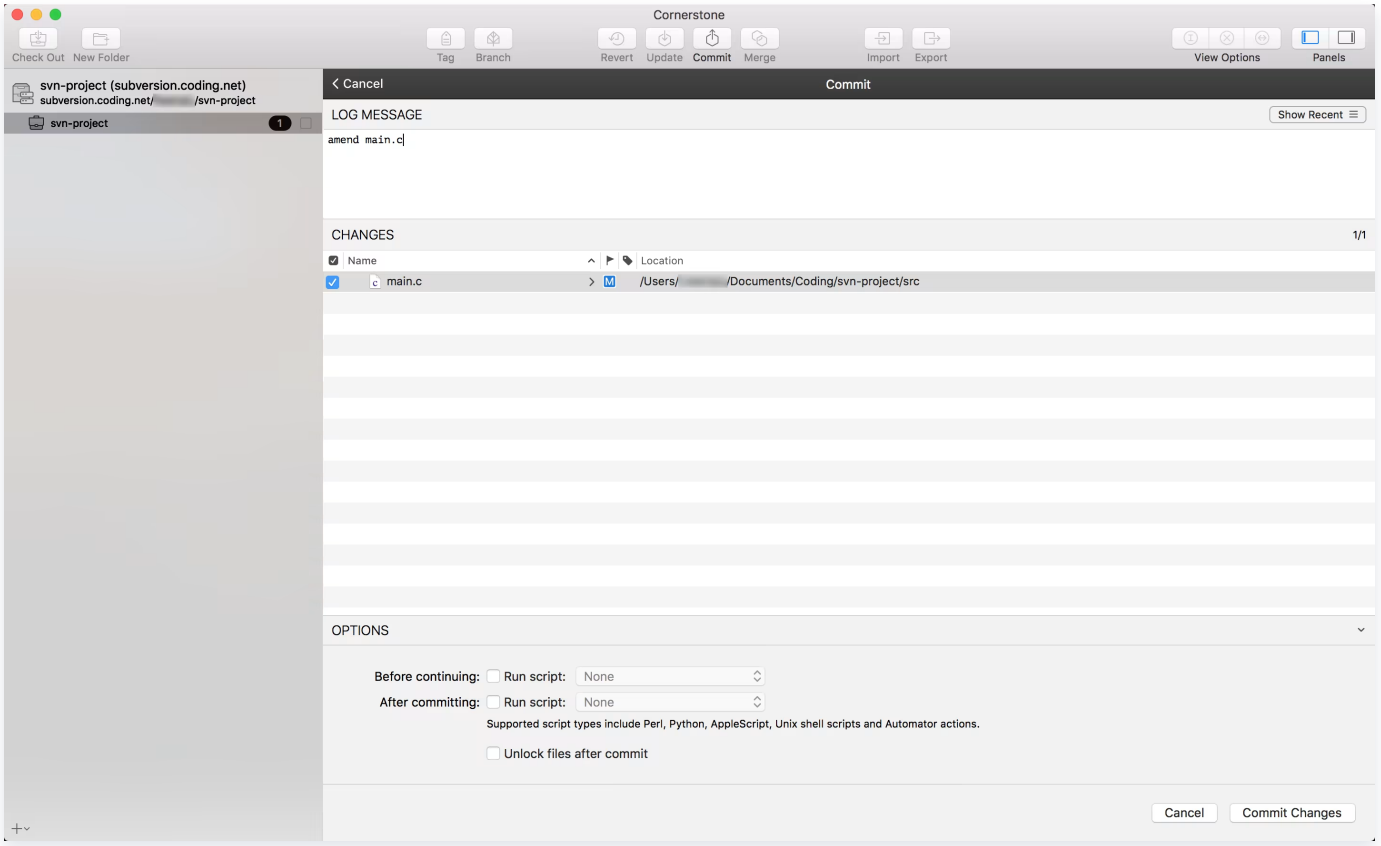
1. 打开 Cornerstone 后，单击 **Add Repository** 来添加 SVN 仓库（请将地址替换为您的 SVN 仓库地址）引用。



然后可以看到仓库的内容如下：



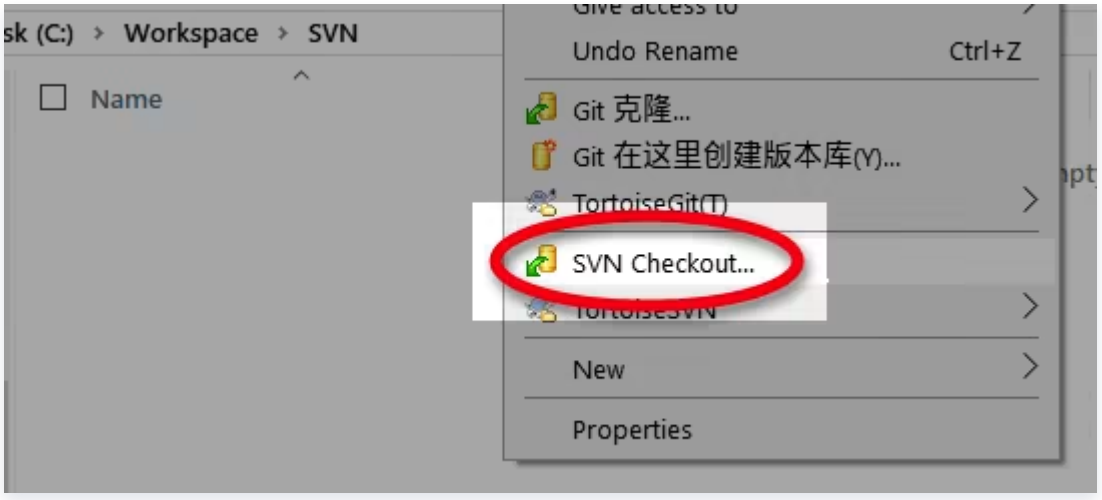
2. 把仓库 checkout 出来，并且编辑文件之后，就可以 commit 进仓库，如下图：



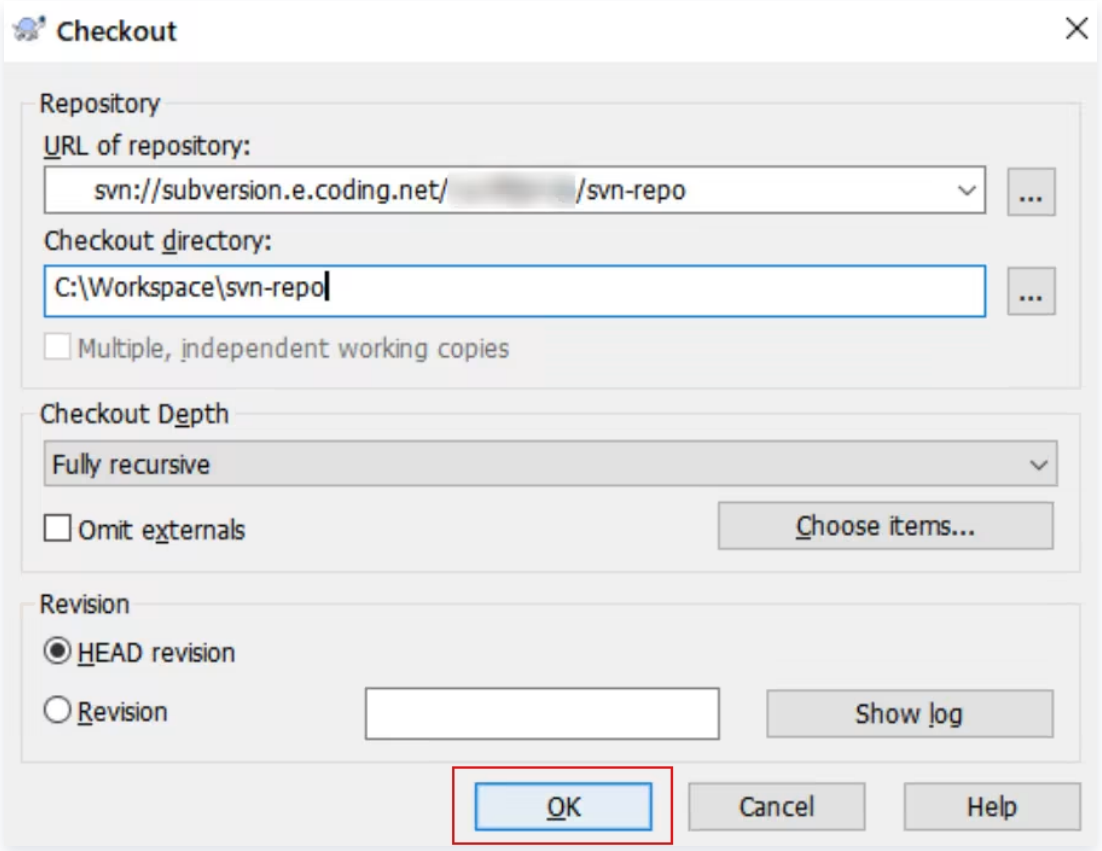
Windows 环境

在 Windows 平台，推荐使用 TortoiseSVN。

1. 下载 安装完成之后，在任意文件目录单击鼠标右键。

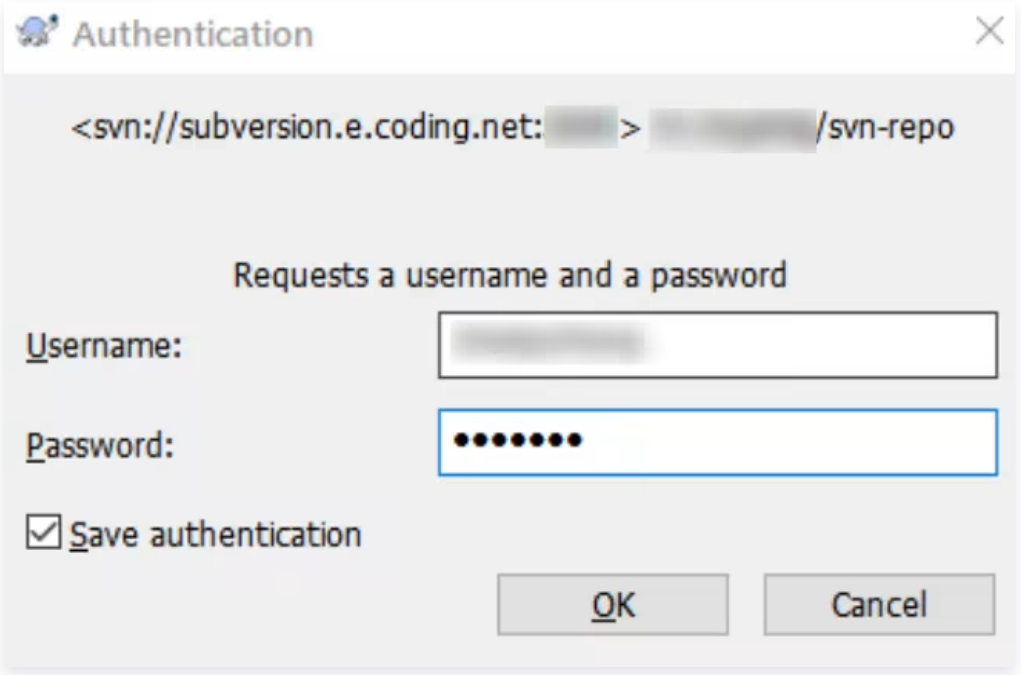


选择 `checkout` 把 SVN 仓库 `checkout` 到本地（请将地址替换为您的 SVN 仓库地址）。

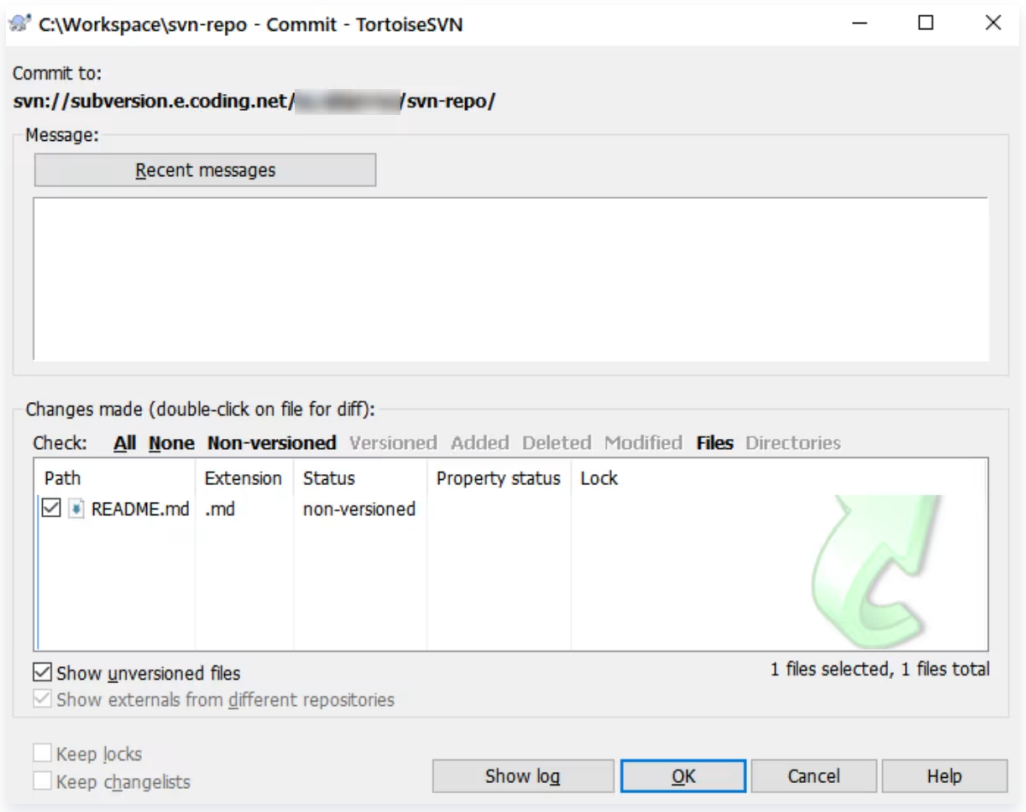


2. 第一次 `checkout` 需要输入用户名和密码。勾选“Save authentication”保存认证信息后，就不需要每次都输入密码。

   其中输入的用户名是 CODING 账号绑定的邮箱。



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：2024-08-19 11:48:11

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. 单击团队首页左侧的**项目**，进入项目列表页，选择目标项目。
3. 选择左侧菜单**代码仓库**，进入代码仓库首页。

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三种权限：

- **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

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 ··· ，选择**权限**。

在弹出的权限设置页面，您可以为该目录添加单独的用户/用户组与相应权限。

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：
  - 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。
  - 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。
  - 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
配置 SSH 公钥

最近更新时间：2024-12-02 15:34:32

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. 单击团队首页左侧的**项目**，进入项目列表页，选择目标项目。
3. 选择左侧菜单**代码仓库**，进入代码仓库首页。

功能介绍

SSH 的全称为 Secure Shell 即安全外壳协议，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，支持新增读写权限；如果添加至团队设置中心，则称为**团队部署公钥**，仅拥有只读权限；如果添加至个人账户，称为**账户 SSH 公钥**，配置后拥有账户内所有代码仓库的读写权限。同一个 SSH 公钥无法既作为部署公钥，又作为个人账户 SSH 公钥。

> **说明：**
> 若 SSH 公钥未用作账户 SSH 公钥，但添加为部署公钥时依然提示错误，那么有可能是此公钥已被其他人用作账户 SSH 公钥。若您的名下有多个团队，也可能是在其他团队中的个人账户设置中添加了此 SSH 公钥。

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 ed25519 -C "your.email@example.com"  // 创建新的 SSH 私钥与公钥密钥对，输入您的邮箱作为标签
Enter file in which to save the key (/Users/you/.ssh/id_rsa): [Press enter]  // 推荐使用默认地址
Enter passphrase (empty for no passphrase):  // 此处直接回车即可；若设置密码，则每次使用 SSH 方式推送代码时都会要求输入密码
```

> **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 百度百科 。
- 若您的系统不支持 Ed25519 算法，请使用命令 `ssh-keygen -m PEM -t rsa -b 4096 -C "your.email@example.com"` 。

成功之后显示如下信息：

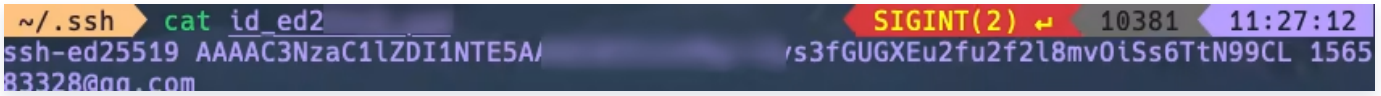
```
Your identification has been saved in /Users/you/.ssh/id_ed25519.
# Your public key has been saved in /Users/you/.ssh/id_ed25519.pub.
# The key fingerprint is:
# 01:0f:f4:3b:ca:85:d6:17:a1:7d:f0:68:9d:f0:a2:db your.email@example.com
```

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。

添加至部署公钥

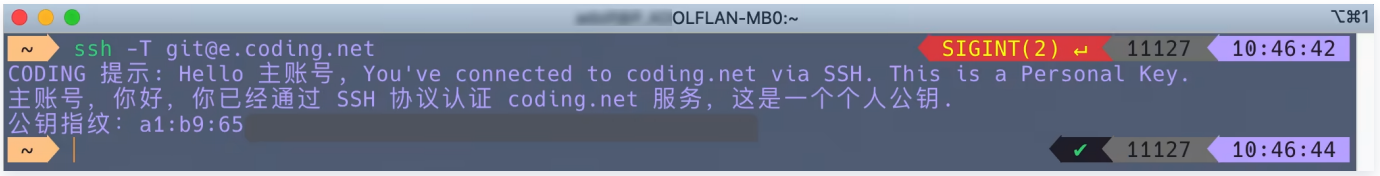
1. 打开上文中生成的密钥对的地址（通常为 `~/.ssh/` 文件夹）找到后缀为 pub 的公钥文件，使用 cat 命令输出所有内容并复制。



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，末尾邮箱将自动填充为公钥名称。

> **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` 。



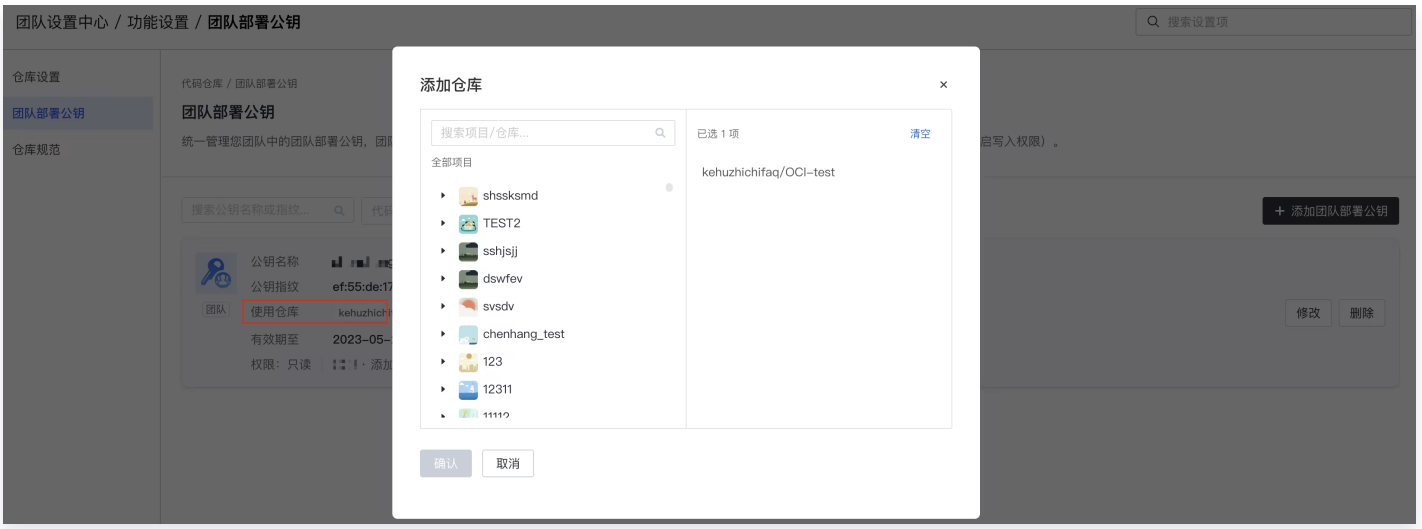
添加至团队部署公钥

团队部署公钥仅具备代码拉取权限，主要用于便捷地拉取不同项目下的代码仓库。团队管理员录入某个 SSH 公钥作为团队部署公钥后，可以将相对应的私钥分享给团队成员。无需开源代码仓库、无需担心个人私钥泄露，仅需通过私钥便能够拉取代码仓库。

在持续交付的场景下，提供私钥便能够让相应的构建机或主机组检出代码。无需繁杂的二次关联与验证，让构建过程更加便捷。

团队所有者 / 管理员点击首页导航左下角的设置按钮，前往**团队设置中心 > 功能设置 > 代码仓库 > 团队部署公钥**中录入 SSH 公钥。

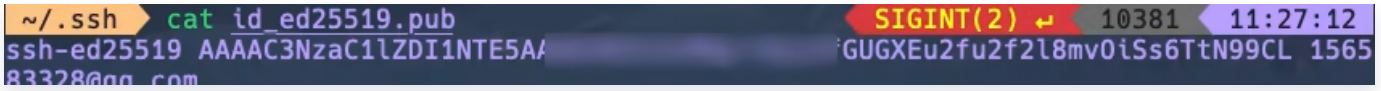
录入后添加至相应的目标仓库。



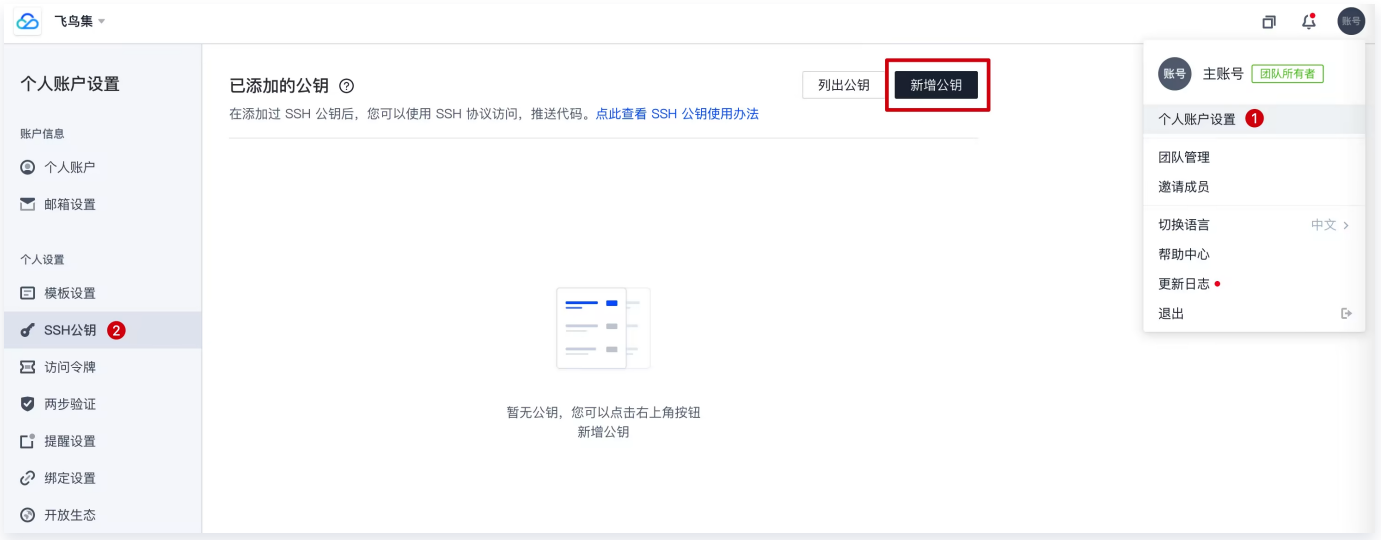
添加完成后，通过私钥便能够使用 SSH 协议拉取代码仓库。

添加至个人账户 SSH 公钥

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：2023-09-11 16:05:18

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. 单击团队首页左侧的**项目**，进入项目列表页，选择目标项目。
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
```

```
~ 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:ox9ko2YsRDgwp4C9imOTha9FWxAXhfe7H7yLIXhcT5A [             ]:11900 (ECDSA)
256 SHA256:u2Xk2ekDfmrav2FAlTPPnGX9seyiZEkOvsFiGyp/EKo               ]:28524 (ECDSA)
256 SHA256:za8qmOBYDLKBCx+hG4gT4/OiqO6ZR/wO0JhMoqJIWtA              ECDSA)
2048 SHA256:jok3FH7q5LJ6qvE7iPNehBgXRw51ErE77S0Dn+Vg/Ik e.coding.net,              (RSA)
~
```

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
```

通过 SSH 协议推拉代码

最近更新时间：2023-09-11 16:05:18

本文为您详细介绍如何通过 SSH 协议进行代码推拉。

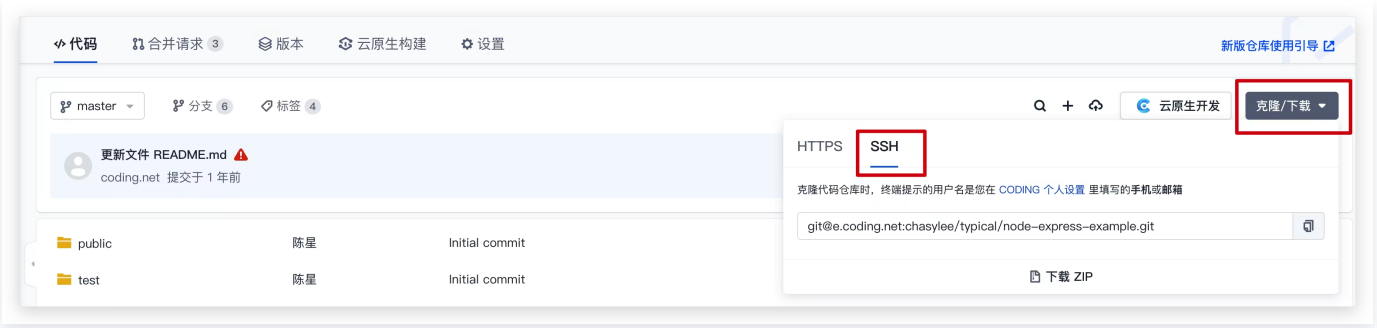
进入项目

1. 登录 CODING 控制台，单击团队域名进入 CODING 使用页面。
2. 单击团队首页左侧的**项目**，进入项目列表页，选择目标项目。
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。

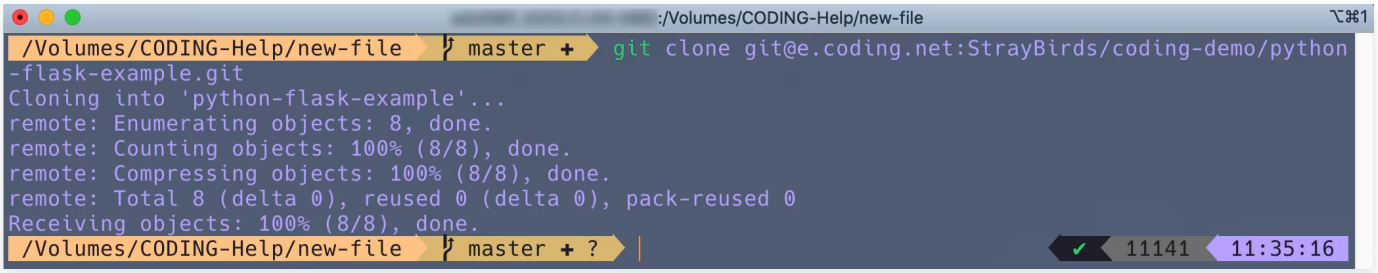
操作步骤

CODING 支持使用 SSH 协议推拉代码。

1. 参见 配置 SSH 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。



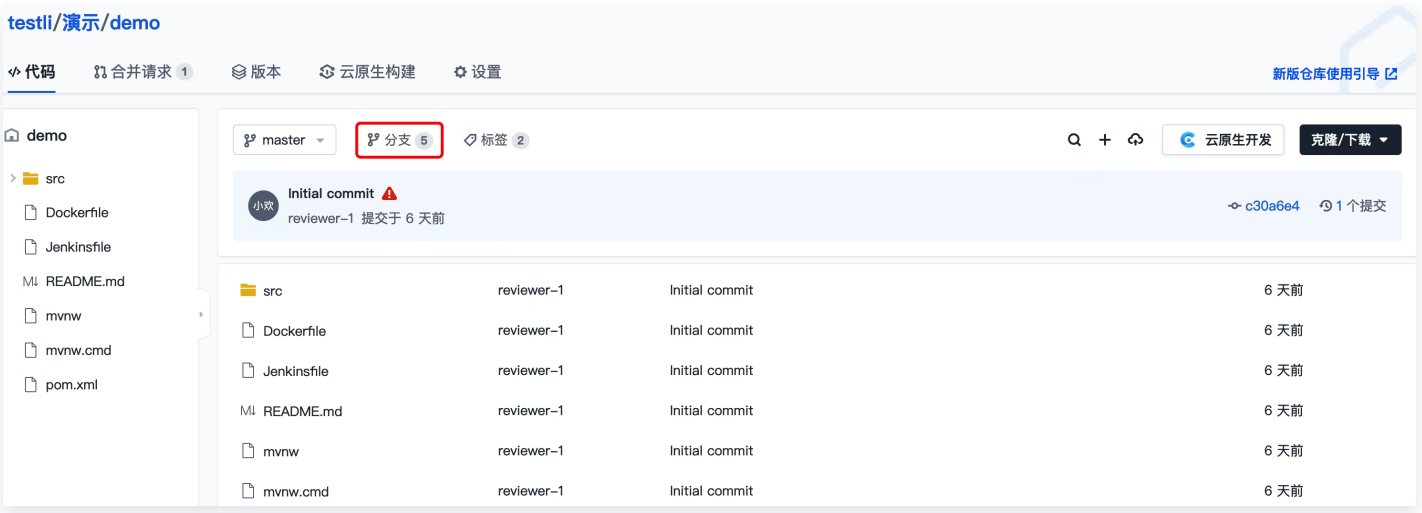
分支管理
创建分支

最近更新时间：2024-08-19 11:48:11

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。
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 常用命令速查表 。

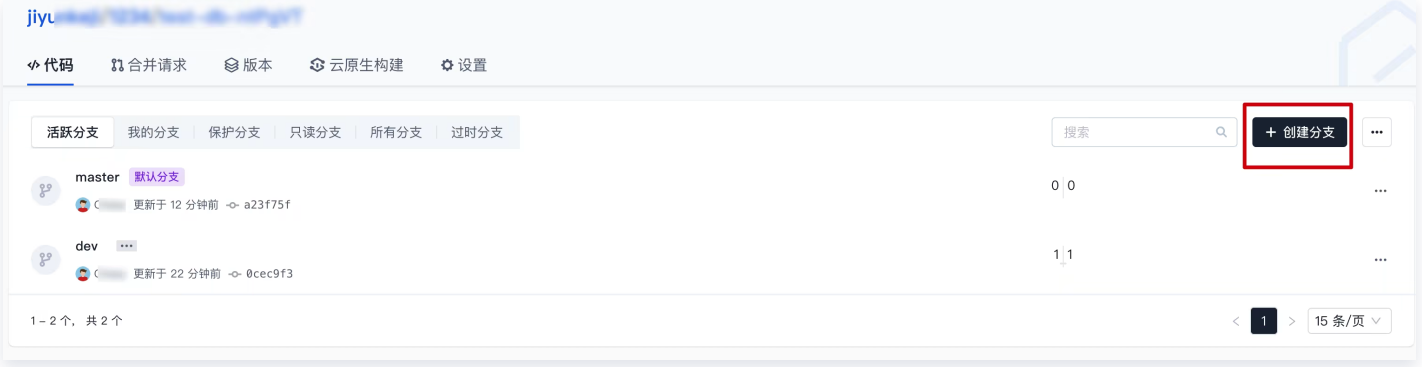
查看分支

进入代码仓库的详情页面之后，单击**分支**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。



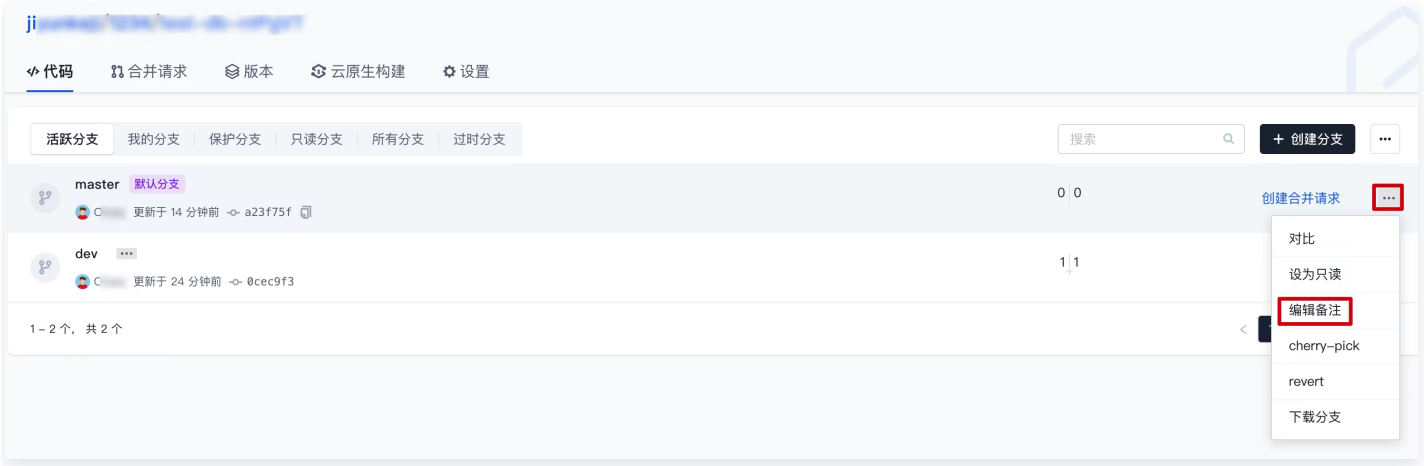
新建分支

单击右上角**创建分支**，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。新旧分支内容一致，相当于在原分支的基础上衍生出新分支。



添加备注

当分支名称不能完整描述分支用途时，可以添加一个简单的描述信息用以说明分支用途。



设置默认分支

最近更新时间：2024-08-16 15:04:21

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。

当用户所在的权限组具备保护分支权限时，将具备调整默认分支的权限。

设置保护分支

最近更新时间：2023-10-23 15:14:52

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**中的分支规则名称中输入需要被保护的分支，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，查看 触发规则 了解更多信息。
- **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，都会自动将当前分支的全部管理员设置为评审者。
- **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。一般而言不同的代码路径决定了应用中的某项功能，这些功能的维护者天然不希望代码被其他成员无意间改动。
- **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”后，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：

设置隐藏分支

最近更新时间：2023-12-29 17:59:41

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。

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。

设置只读分支

最近更新时间：2023-10-23 15:14:52

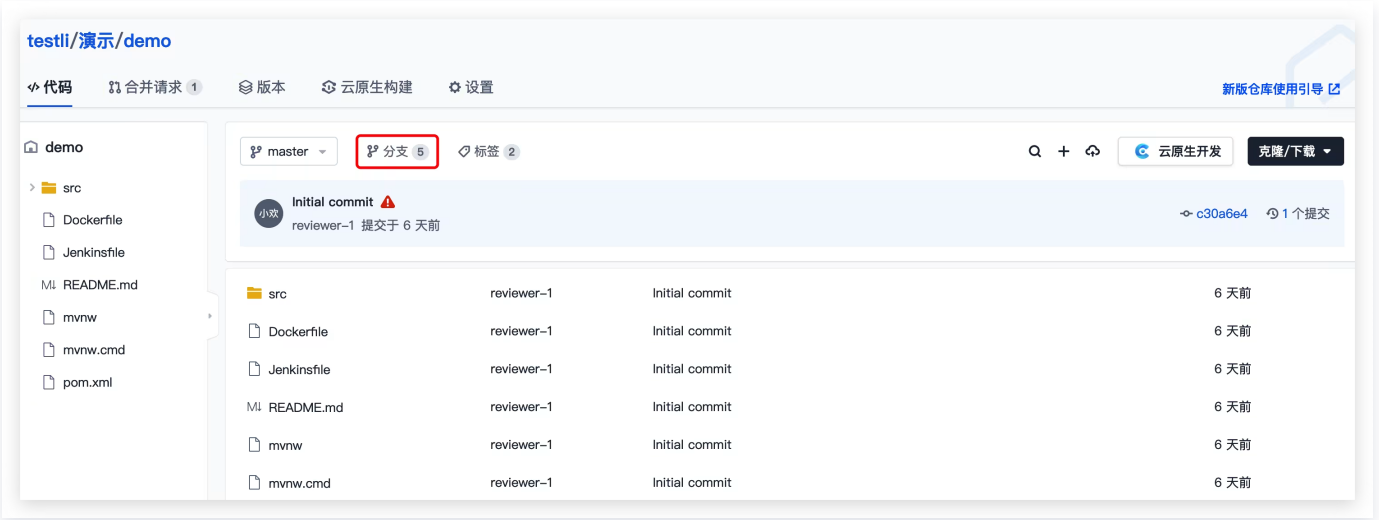
设置只读分支后，该分支将无法被团队中的任何成员写入或提交合并请求，仅允许拉取。

权限配置

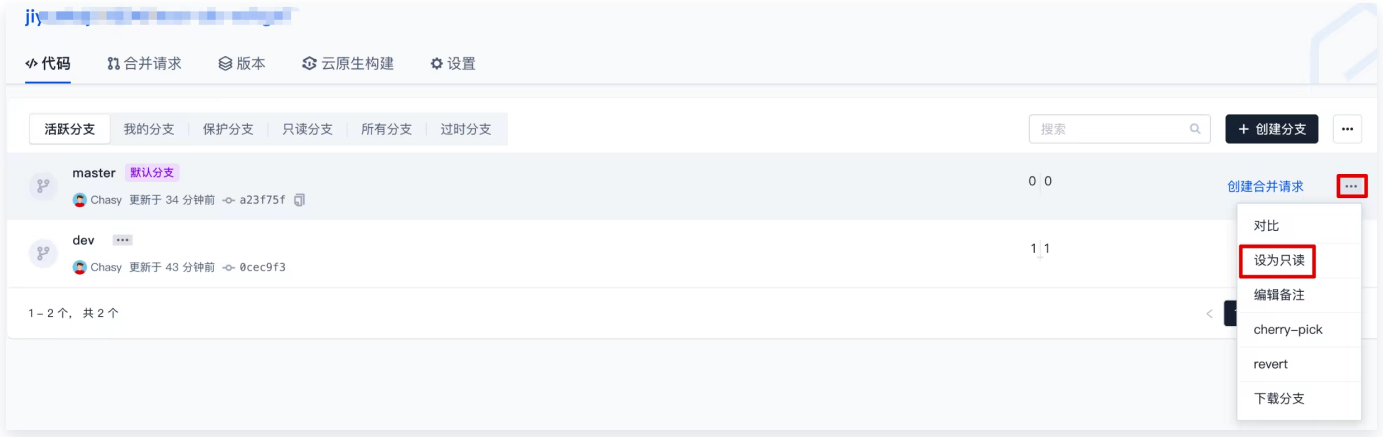
在**团队设置中心 > 组织和成员 > 资源权限方案**的**仓库权限方案**页签中单击权限列表中的权限组名称，设置仓库权限。当仓库权限包含**仓库设置**时，管理员可以设置或取消只读分支。

设置只读分支

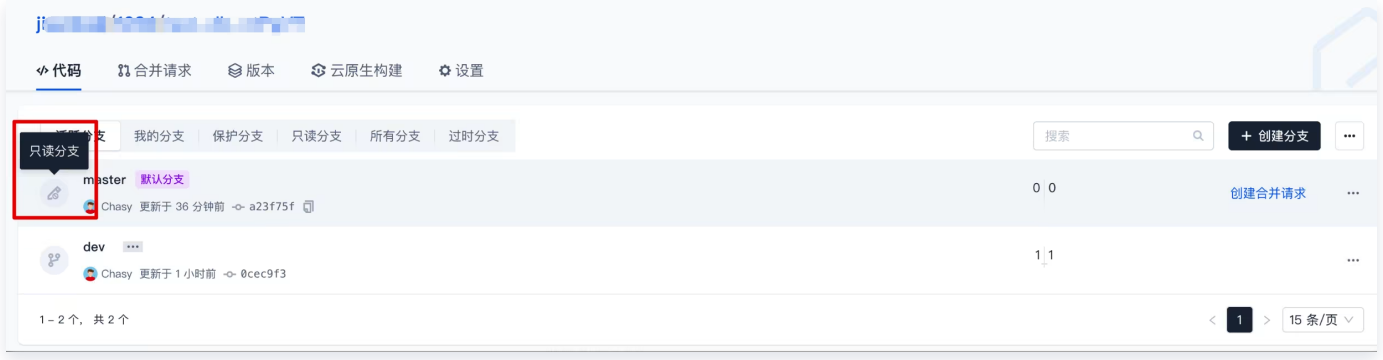
1. 进入任一代码仓库后，单击**分支**进入分支列表。



2. 选择指定分支，单击 ··· 并选择**设为只读**。



开启只读模式后，该分支出现只读标识。如需取消只读，单击 ··· 并选择**取消只读**即可。



合并请求与代码评审
合并请求设置

最近更新时间：2024-08-16 15:04:21

合并请求的全称为 Merge Request，下文将简称为 MR。
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、调整默认合并方式、修改 MR 默认目标分支与设置合并提交信息模板（Accept Merge Request）。

基础设置

是否默认删除源分支

删除源分支有助于保持仓库整洁，打开此开关后，MR 请求将自动勾选图中的开关。如不需要删除源分支，合并者取消勾选此按钮即可。

是否默认以 Fast-Forward 模式合并

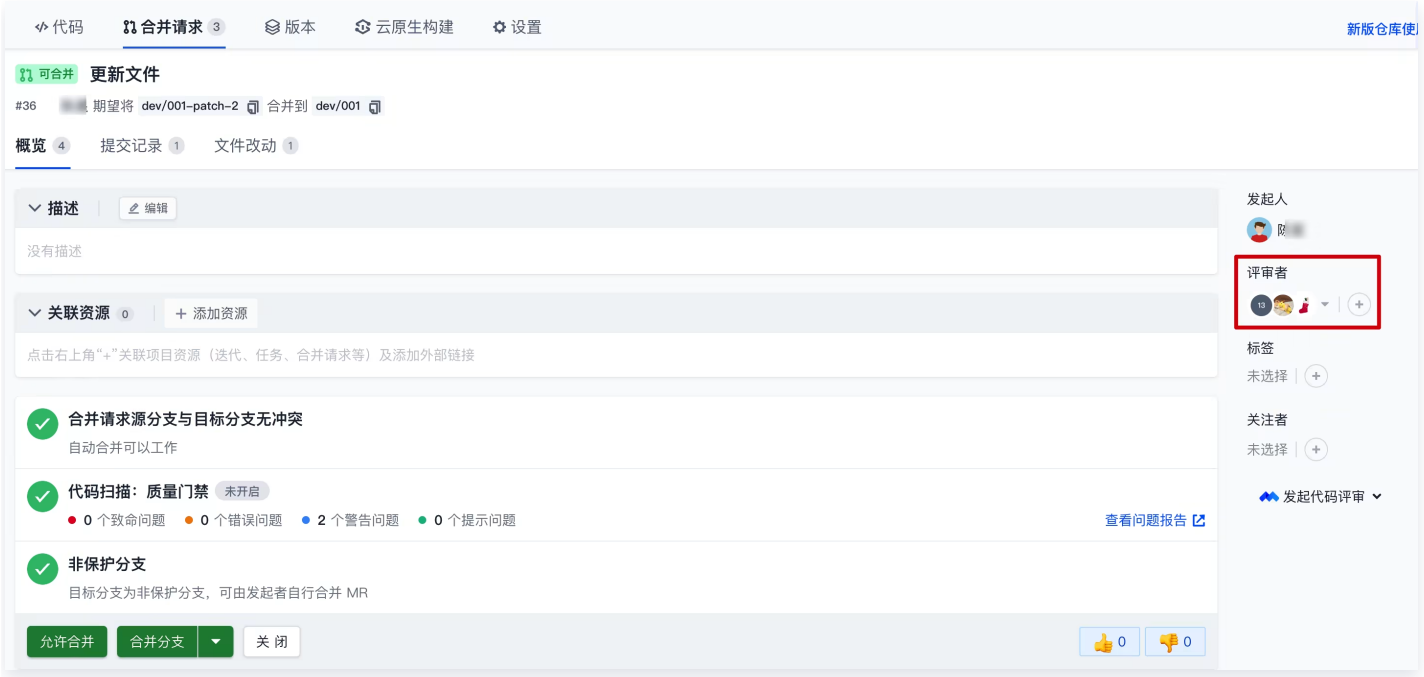
此功能主要用于维护主干分支的整洁。这种模式的合并过程将不会在目标分支中留下曾与源分支进行合并的历史信息。源分支的提交信息将直接与目标分支融合，而不会产生一个两者曾经发生过合并行为的提交记录，相当于使用 git merge 命令时添加 -ff 参数。

状态检查

开启此功能后，所有的 MR 将在源分支自动触发持续集成任务，通过后才能允许合并至目标分支。

合并前必须获得所有评审者的允许合并

此功能主要用于确保所有的 MR 经评审者全体同意后才能够被允许合并，无论目标分支是否为保护分支。



合并方式

当源分支有多个提交的时候，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设置该分支为目标分支。建议使用主干分支为合并请求的默认目标分支。

合并提交消息模板

所有的合并请求都会在目标分支中留下一个提交消息，例如 Git 中的默认消息为 Accept Merge Request #xxx，您可以在此处修改这条消息模板，更加清晰的记录合并结果。

提交消息变量

变量名	变量描述	变量示例
${source_branch}	源分支	feature/demo
${target_branch}	目标分支	main
${title}	合并请求标题	这是一个合并请求的标题
${id}	合并请求的引用事项号	#1
${url}	合并请求访问链接	https://team-name.coding.net/p/project-name/d/depot-name/git/merge/21

${reviewer}	合并请求评审人员	@开发组长
${approver}	合并请求允许合并人员	@开发组长
${acceptor}	合并请求合入分支人员	@开发组员
${creator}	合并请求创建人	@开发组员

发起合并请求

最近更新时间：2023-10-23 15:14:52

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。

发起合并请求

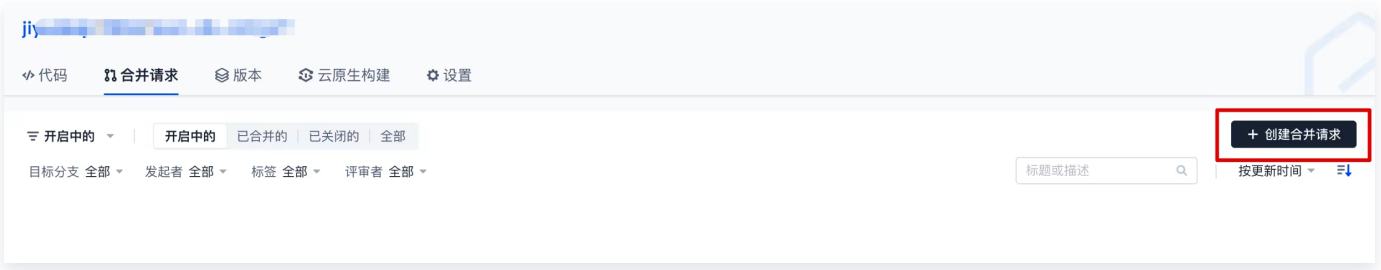
您可以通过命令行或在 Web 端手动创建合并请求。

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**创建合并请求**。



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。

3. 创建合并请求后还可以在**提交记录**页分别查看本次请求中的所有提交历史与单次推送中包含的所有提交记录。

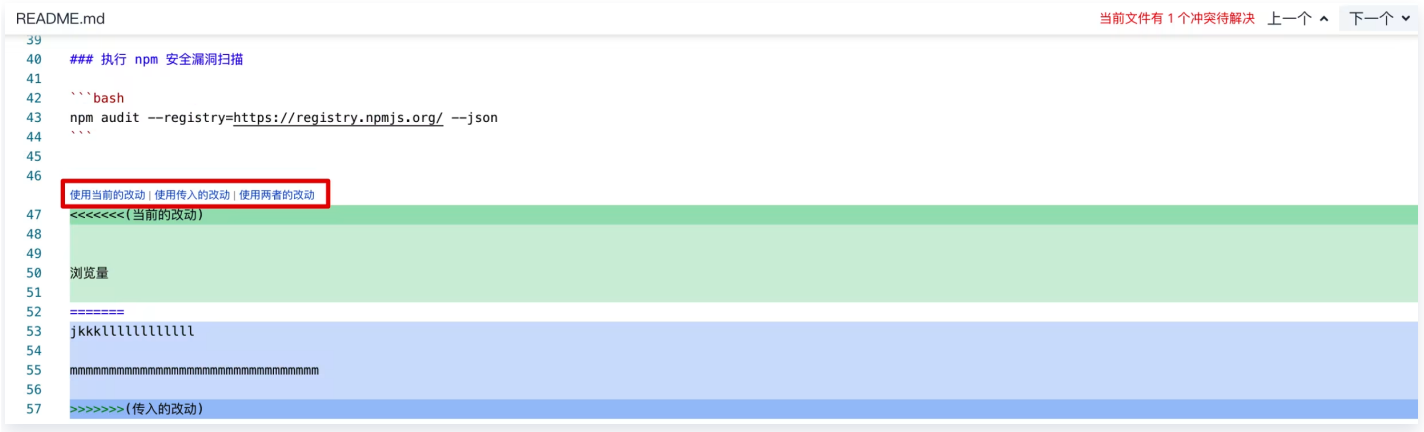
解决合并冲突

造成合并冲突的原因是源分支与目标分支对相同的代码做出了不一致的变更。发起合并请求时将自动比对源分支与目标分支的内容差异，若发现存在冲突内容，系统将提示源分支和目标分支不可自动合并，您可以通过在线或本地两种方式解决冲突后继续发起合并请求。

在线解决冲突

遇到代码冲突的情况往往需要在本地终端中反复拉取并解决，然后再推送至远端仓库中。在线解决冲突功能让繁琐的步骤简化为简单的鼠标操作，单击网页中的**查看冲突内容**便能够在线预览相冲突的内容。

冲突内容将会被高亮展示，此时选择需要保留的内容。单击右上角的切换按钮快速切换至其他冲突的所在行，解决所有冲突后提交改动，此时将在代码仓库中产生一个新的提交记录。



本地解决冲突

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。

确认合并请求

- **合并的目标分支为保护分支**

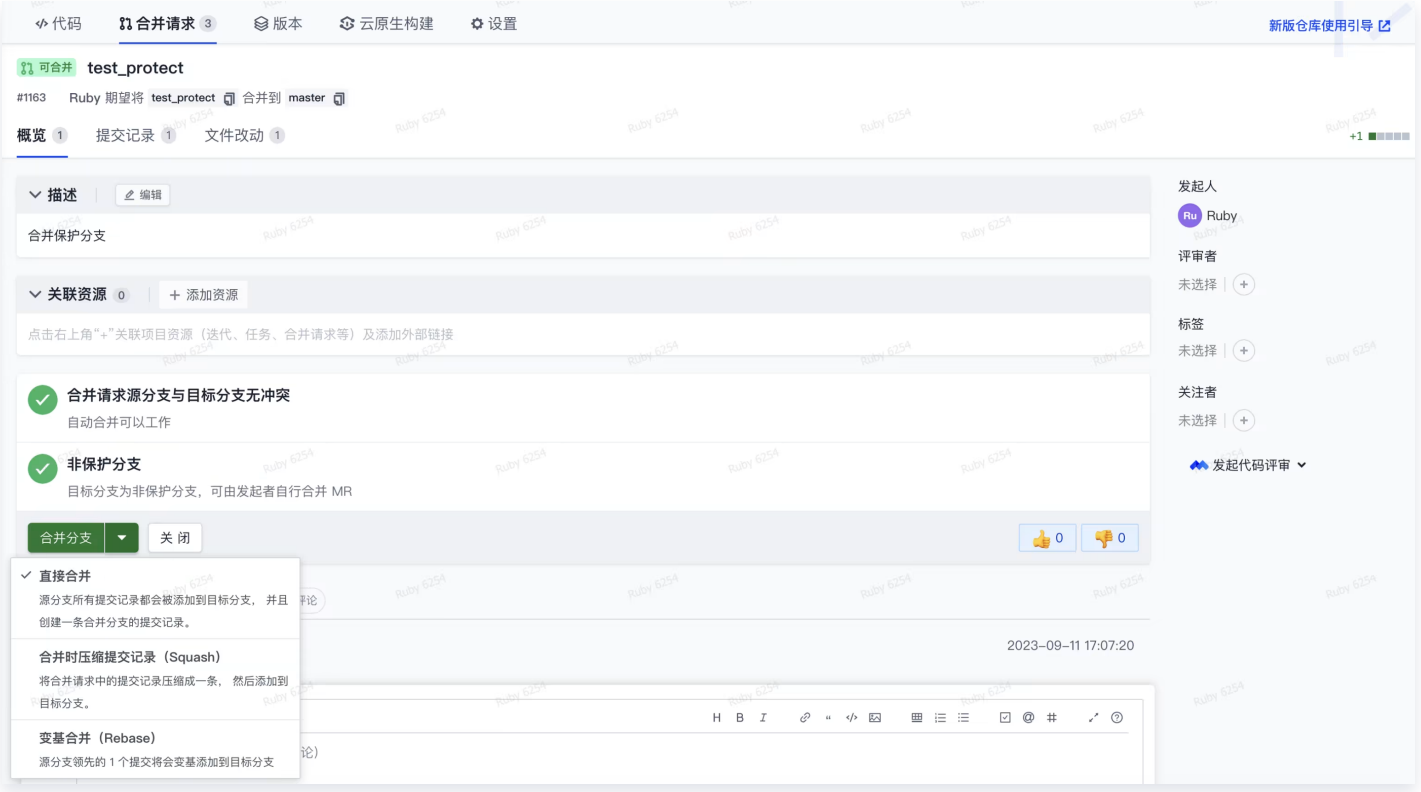
  若合并请求的发起者为分支管理员，那么可以自行合并。若发起者为普通成员，那么需要通过分支管理员的评审才能完成合并。
- **合并的目标分支为非保护分支**

  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。

选择合并类型

在**合并请求**页面单击**合并分支**，在下拉列表中选择合并类型，系统将根据选择的合并类型进行相应的合并处理。

- **直接合并**：源分支所有提交记录都会被添加到目标分支， 并且创建一条合并分支的提交记录。
- **合并时压缩提交记录（Squash）**：将合并请求中的提交记录压缩成一条， 然后添加到目标分支。
- **变基合并（Rebase）**：源分支领先的 1 个提交将会变基添加到目标分支。



删除源分支

发起合并请求时，勾选删除源分支，可在发起合并请求时删除源分支。

Fast-Forward 模式合并

Fast-Forward 模式下的合并过程不会在目标分支中留下曾与源分支进行合并的历史信息。源分支的提交信息将直接与目标分支融合，而不会产生一个两者曾经发生过合并行为的提交记录。

若勾选了以 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。
勾选此选项相当于使用 git merge 时添加 -ff 参数。

> **说明：**
> 在 合并请求设置 可以设置默认删除源分支及默认以 Fast-Forward 模式合并。

使用 GPG 签名 commit 记录

最近更新时间：2024-08-16 15:04:21

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。

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
```

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2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
```

生成公钥之后，可将其添加至您的 CODING 账户。

添加公钥至个人账户设置

1. 登录 CODING 之后，单击页面左下角中的**个人账户设置**选项。
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。

> **注意：**
> 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。

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。

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。

验证签名

将签了名的提交推送至 CODING 代码仓库后，您可以在代码仓库的**提交**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

如果参考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
```

版本与标签
代码版本

最近更新时间：2023-09-11 16:05:19

代码版本可以理解为代码仓库某一时刻的快照。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，再单击版本进入管理发布页。

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，单击右上角的创建版本。**
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. 摘要内容支持自定义修改，可作为团队公告板告知如何填写合乎规范的版本号。团队负责人/管理员可以前往**团队设置中心 > 功能设置 > 仓库设置**中自定义摘要中的内容。

4. 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。

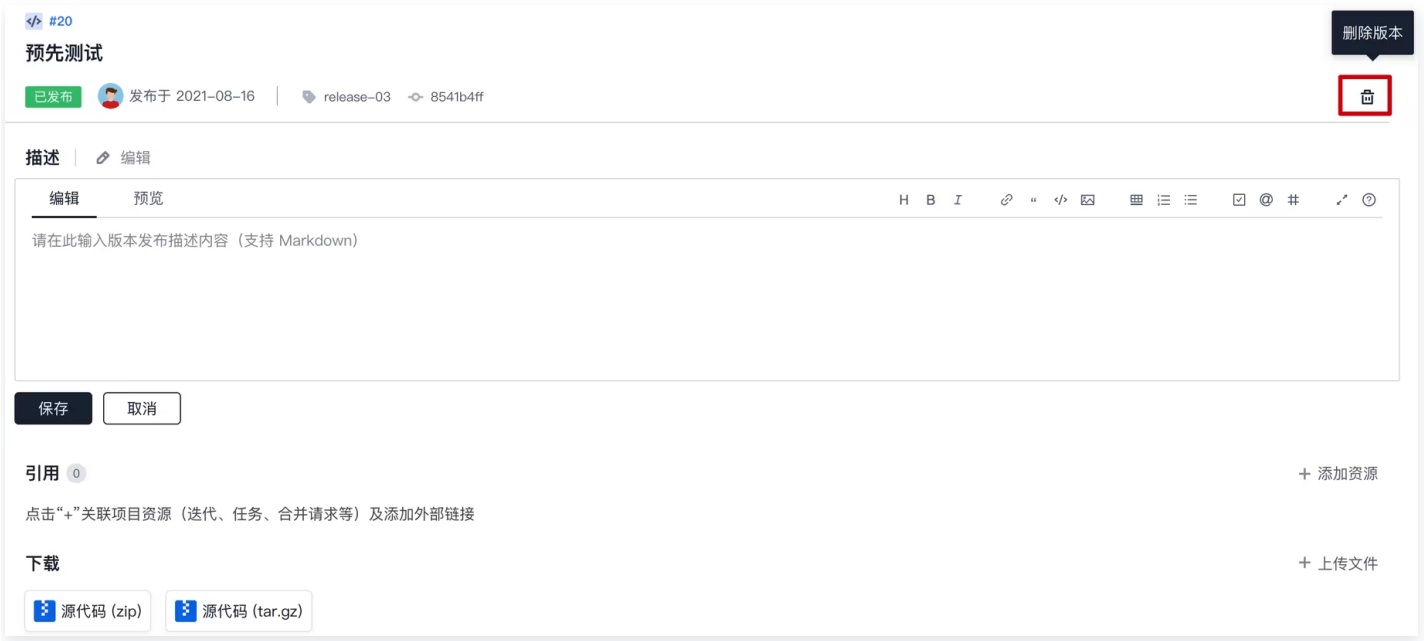
编辑代码版本

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**，即可编辑版本信息。

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

前往代码仓库的设置 > 版本发布页面，项目管理员可以指定版本发布时所选择的默认分支。

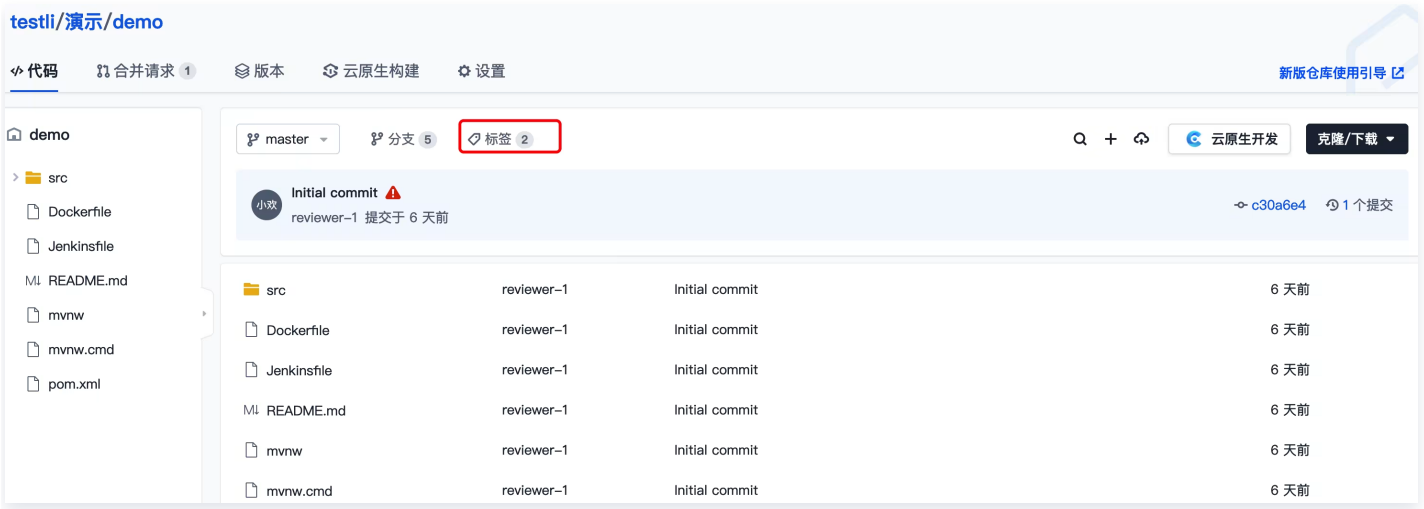

代码
合并请求 3
版本
云原生构建
设置
仓库设置
基本设置
权限方案
部署公钥
文件锁定
分支设置
仓库规范
访问设置
推送设置
合并请求
版本发布
代码标签
仓库安全
推送权限
版本发布
版本发布默认分支
保存
新建版本发布时，目标分支会默认选择此分支

代码标签

最近更新时间：2024-09-04 14:25:11

概述

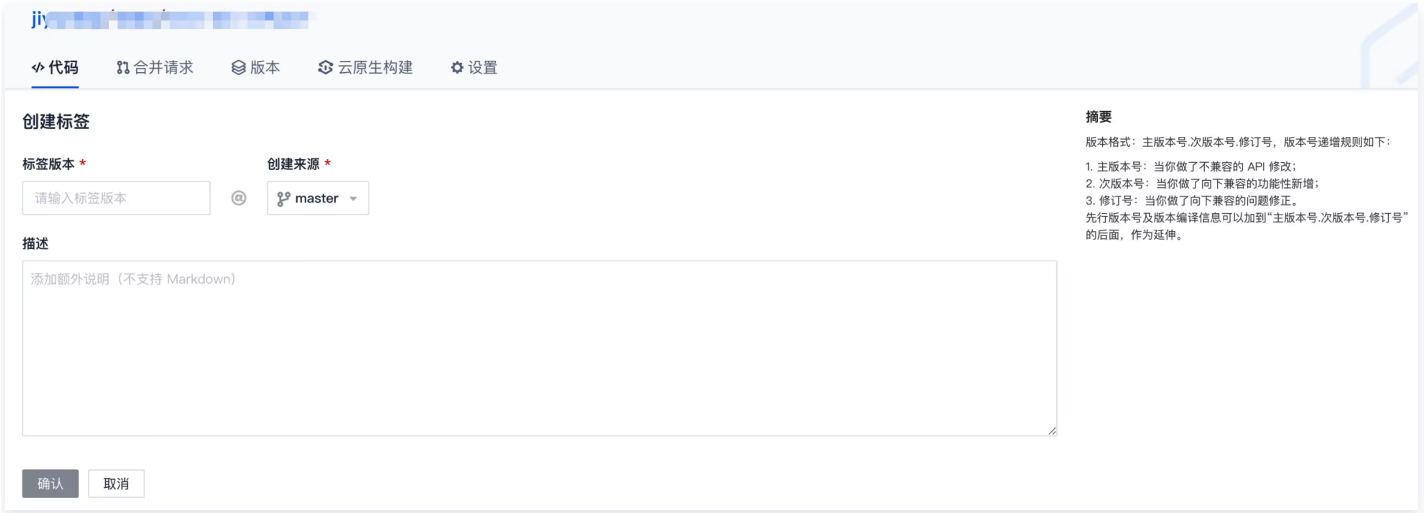
登录 CODING DevOps 控制台 > 代码仓库，单击**立即使用**后，在代码仓库管理列表，继续单击目标代码仓库进入其详情页面。再单击**标签**进入代码标签管理列表。



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。单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）即可创建新标签。页面右侧展示的“摘要”可作为公告板，告知团队成员如何书写统一格式的代码标签。

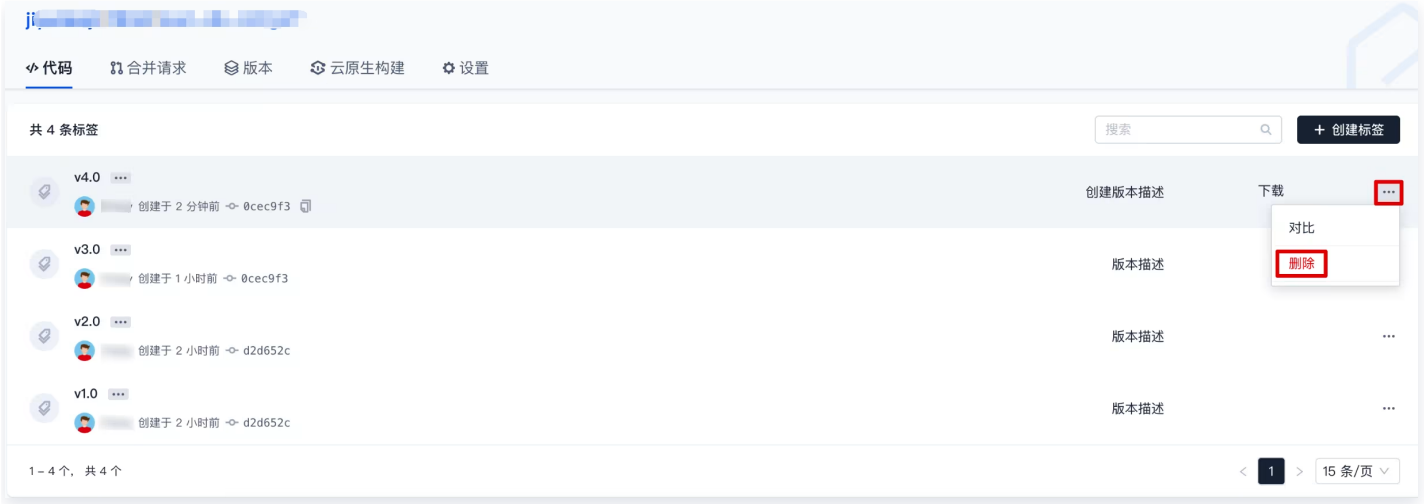


团队负责人/管理员可以前往**团队设置中心 > 功能设置 > 代码仓库 > 仓库设置**中自定义摘要中的内容。

若希望代码标签严格遵守团队规范，那么可以在代码仓库 > 设置 > 代码标签页面 设置保护标签 ，以限制成员创建或修改特定格式的代码标签。

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关联的代码标签。



若任一标签与版本发布关联，只能在**版本**列表页删除对应版本的同时将标签删除。若不希望标签被删除，项目管理员可以前往**代码仓库 > 设置 > 代码标签**页面，勾选是否允许删除标签。若不允许删除标签，那么所有项目成员将无法通过命令行删除标签，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：

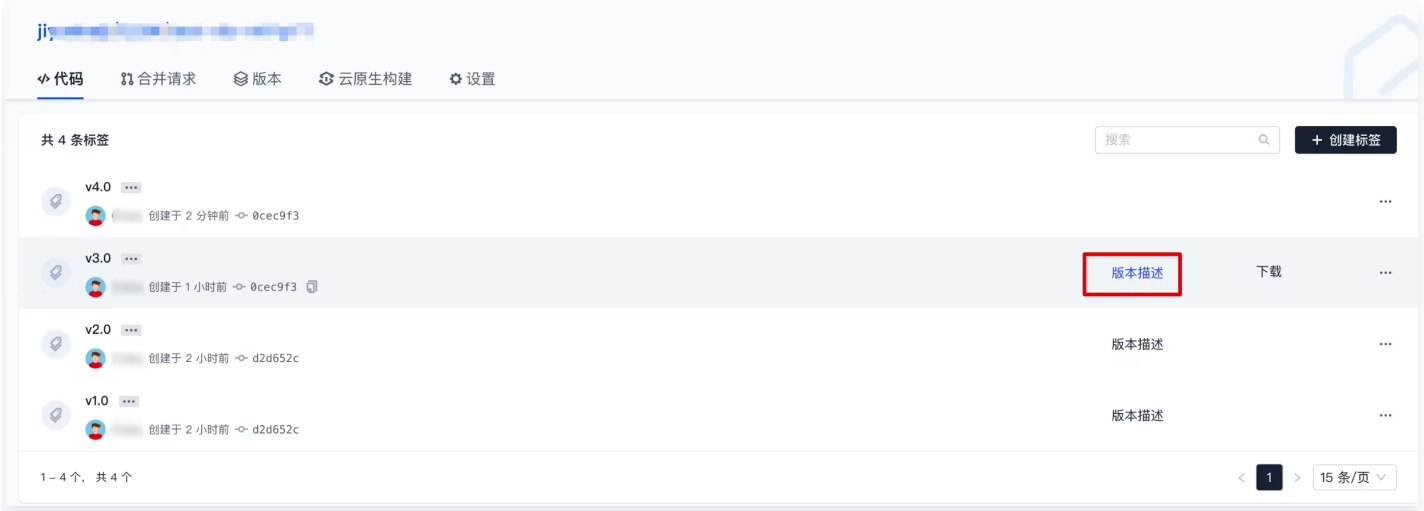
```
git push --tag origin xxxx-release
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
remote: [err=32] You have no permission to update protected tag (refs/tags/xxxx-release).
remote: 你没有权限推送到保护标签（refs/tags/xxxx-release），查看更多信息：https://coding.net/help/doc/git/git-branch.html#保护分支
remote: error: hook declined to update refs/tags/xxxx-release
To https://e.coding.net/vcs-test/coding-demo/coding-demo.git
 ! [remote rejected] xxxx-release -> xxxx-release (hook declined)
error: failed to push some refs to 'https://e.coding.net/vcs-test/coding-demo/coding-demo.git'
```

在 Web 端创建标签或新建版本时也同样会失败。

例如，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。

查看版本信息

代码标签可作为某一项代码版本，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**即可进行编辑。



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。