

CODING DevOps

代码托管



腾讯云

【版权声明】

©2013–2025 腾讯云版权所有

本文档（含所有文字、数据、图片等内容）完整的著作权归腾讯云计算（北京）有限责任公司单独所有，未经腾讯云事先明确书面许可，任何主体不得以任何形式复制、修改、使用、抄袭、传播本文档全部或部分内容。前述行为构成对腾讯云著作权的侵犯，腾讯云将依法采取措施追究法律责任。

【商标声明】



及其它腾讯云服务相关的商标均为腾讯云计算（北京）有限责任公司及其关联公司所有。本文档涉及的第三方主体的商标，依法由权利人所有。未经腾讯云及有关权利人书面许可，任何主体不得以任何方式对前述商标进行使用、复制、修改、传播、抄录等行为，否则将构成对腾讯云及有关权利人商标权的侵犯，腾讯云将依法采取措施追究法律责任。

【服务声明】

本文档意在向您介绍腾讯云全部或部分产品、服务的当时的相关概况，部分产品、服务的内容可能不时有所调整。您所购买的腾讯云产品、服务的种类、服务标准等应由您与腾讯云之间的商业合同约定，除非双方另有约定，否则，腾讯云对本文档内容不做任何明示或默示的承诺或保证。

【联系我们】

我们致力于为您提供个性化的售前购买咨询服务，及相应的技术售后服务，任何问题请联系 4009100100 或 95716。

文档目录

代码托管

 快速开始

 权限配置

 Git 仓库

 创建仓库

 导入或关联外部仓库

 管理仓库

 代码对比

 SVN 仓库

 创建 SVN 仓库

 访问 SVN 仓库

 管理 SVN 目录权限

 SSH 协议

 配置 SSH 公钥

 密钥指纹鉴权

 通过 SSH 协议推拉代码

 分支管理

 创建分支

 设置默认分支

 设置保护分支

 设置隐藏分支

 设置只读分支

 合并请求与代码评审

 合并请求设置

 发起合并请求

 使用 GPG 签名 commit 记录

 版本与标签

 代码版本

 代码标签

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

- 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。
- 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

- 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。

- homebrew 安装好后，在终端输入 `brew install git` 命令完成 Git 安装。

- 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。

```
/Volumes/CODING-Help/new-file git init      SIGINT(2) ↵ 11074 ↵ 11:28:57
Initialized empty Git repository in /Volumes/CODING-Help/new-file/.git/
/Volumes/CODING-Help/new-file [master] | ✓ 11075 ↵ 11:29:02
```

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：

```
1 git config --global user.name "你的名称"  
2 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：

项目动态

所有 任务 代码 文件 讨论 其他动态

今天 (2017-11-22 周三)

11:43 大黑 推送到了 项目 分支： master
</> 大白 : [b079372] edit learn_git
</> 大白 : [593743f] resolv conflict
</> 大白 : [eea4d4f] edit readme.txt

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。

项目概览 代码仓库 关联仓库

我的星标 已归档 全部仓库 未分组 更多 搜索仓库... 默认排序

仓库名	更新时间	操作
demo	更新于 4 小时前	...
ruby	更新于 1 天前	...
spring	更新于 8 个月前	...

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。

[基本设置](#) [菜单管理](#) [通知设置](#) [成员](#) [权限组](#) [每日工作邮件提醒](#) [分类标签](#)

菜单管理

管理员可根据团队协作场景，自由组织和配置各功能模块的开关。

关闭后，菜单将隐藏功能模块入口，成员无法访问该模块和数据。但已产生的数据或已配置的触发规则不受影响。再次开启，即可恢复至关闭前状态。

菜单名称	菜单描述	开关
项目协同	管理项目信息，包括需求管理、任务分配、缺陷跟踪、状态流转、进度展示等。	<input checked="" type="checkbox"/>
代码仓库	代码仓库、分支、合并请求、代码评审、发布，包括设置 > 代码托管。	<input checked="" type="checkbox"/>
持续集成	完全兼容持续集成软件 Jenkins，支持多任务并发。	<input checked="" type="checkbox"/>
持续部署	基于 Spinnaker，支持 Kubernetes、云服务器等多种场景的部署。	<input checked="" type="checkbox"/>
代码扫描	代码检查、代码度量、重复代码、代码统计。	<input checked="" type="checkbox"/>

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。

The screenshot shows the Cloud Studio interface. On the left, a GitHub repository page for 'ji' is displayed, showing branches v4.0, 2, and 4, and tags 4. A message indicates a file was updated by 'CODING-Support'. Below the repository, there's a terminal window with the command `git clone https://e.coding.net/ji/test-db-ntPgVT.git` entered. To the right of the terminal, a modal window for cloning the repository via HTTPS or SSH is open, with the URL `https://e.coding.net/ji/test-db-ntPgVT.git` and a 'Download ZIP' button. A red box highlights the 'Clone/Download' button in the top right corner of the modal.

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。

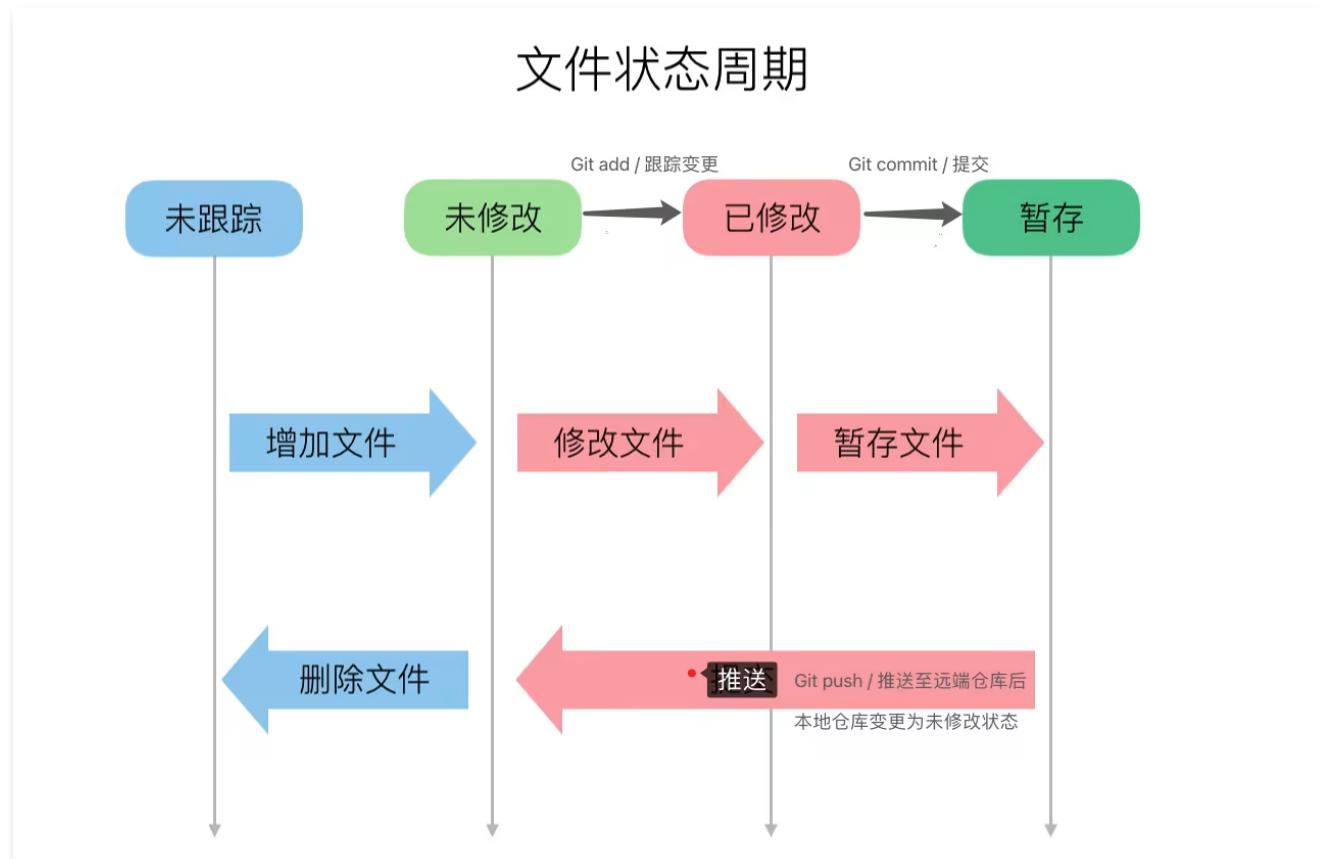
```
/Volumes/CODING-Help/视频制作/代码仓库 > git clone https://e.coding.net/StrayBirds/demo/demo.git
Cloning into 'demo'...
Username for 'https://e.coding.net':
Password for 'https://':
remote: Enumerating objects: 13, done.
remote: Counting objects: 100% (13/13), done.
remote: Compressing objects: 100% (7/7), 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"
```

```
/Volumes/CODING-Help/new-file [master +] git commit -m "wrote a readme and a learn_git file"
[master (root-commit) f382773] wrote a readme and a learn_git file
 2 files changed, 2 insertions(+)
  create mode 100644 learn-git.txt
  create mode 100644 readme.txt
/Volumes/CODING-Help/new-file [master] | 11084 14:08:16
```

`-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。

```
1 // 第一步：安装 yarn
2
3 brew install yarn
4
5 // 第二步：安装插件
6
7 yarn add -D standard-version
8
9 // 第三步：使用插件提交代码
10
11 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。

#630 「常见问题」个人版常见问题补充

1人关注 | ...

编辑描述 上传附件 添加子工作项 引用资源

点此编辑描述

引用资源 +

被引用

38993b Accept Merge Request #642: (mr/master/issue/630 -> master) coding-help-gener...
17776b fix: 常见账号问题 #630 coding-help-gener...

活动日志

全部 只看日志 只看评论

+ [redacted] 创建了任务 2021-03-01 11:05
https://help.coding.net/docs/[redacted].html
2021-03-08 19:27 打开评论 (0)

[redacted] 更新了处理状态为 已完成 2021-03-08 19:27

点此发表评论

状态 已完成
处理人 未指定
迭代 帮助中心三月排期
故事点 -
优先级 中
截止日期 未指定
标签 +
腾讯会议 快速会议 预定会议
创建于 3 个月前

关联事项后，将鼠标移至提交记录的 ID 上可以预览事项。

jy [redacted] demo 切换仓库

代码 合并请求 版本 云原生构建 设置

pro

提交人 全部 提交日期 开始日期 至 结束日期

Accept Merge Request #32 : (master-patch-1 -> pro) ...
提交于 几秒前

更新 readme #1 [redacted]
#1 project2023

到期提醒机制
提交于 1 天前

Initial commit [redacted]
提交于 1 天前

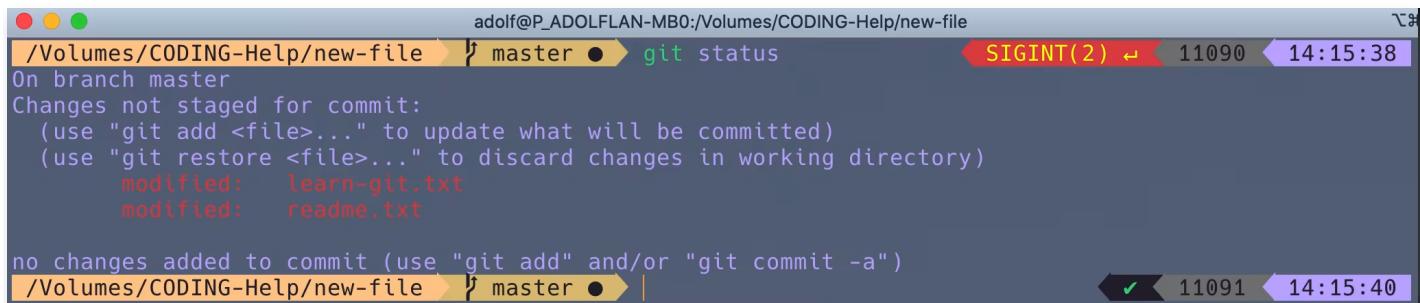
查看文件状态 (git status)

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：

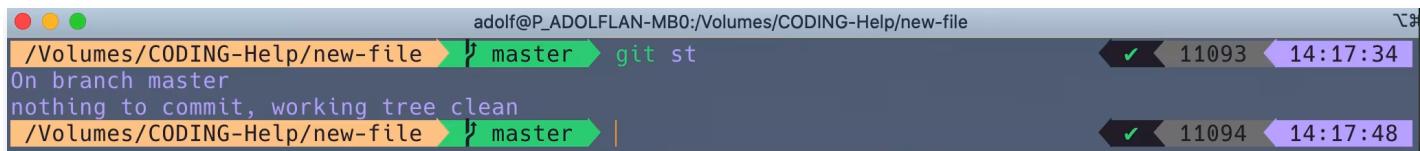


```
adolf@P_ADOLFLAN-MB0:/Volumes/CODING-Help/new-file
/Volumes/CODING-Help/new-file(master) git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified:   learn-git.txt
    modified:   readme.txt

no changes added to commit (use "git add" and/or "git commit -a")
/Volumes/CODING-Help/new-file(master)|
```

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：



```
adolf@P_ADOLFLAN-MB0:/Volumes/CODING-Help/new-file
/Volumes/CODING-Help/new-file(master) git st
On branch master
nothing to commit, working tree clean
/Volumes/CODING-Help/new-file(master)|
```

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”。



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。

仓库设置

基本设置

仓库名称: flash-demo

仓库描述: 请输入仓库描述

仓库图标: F (可更改)

仓库容量管理: 已使用 24.67 MB / 总容量 20.00 GB (修改仓库容量上限)

仓库清理: 仓库清理使用 git gc 优化代码仓库存储空间, 压缩存储库对象, 减少存储库磁盘占用。

危险操作: 归档代码仓库 (归档)

● 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。

团队设置中心

功能设置

仓库设置

- 团队部署公钥
- 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。

全局设置 / 项目权限方案 / 开发

开发 索
具备代码仓库、持续集成、制品仓库的访问与操作权限。

配置权限

分类	功能权限
项目协同	<input checked="" type="checkbox"/> 访问项目协同 <input type="checkbox"/> 编辑迭代 <input type="checkbox"/> 删迭代 <input checked="" type="checkbox"/> 编辑事项
测试管理	<input type="checkbox"/> 访问测试管理 <input type="checkbox"/> 编辑用例 <input type="checkbox"/> 删除用例 <input type="checkbox"/> 编辑自动化用例 <input type="checkbox"/> 归档测试计划 <input type="checkbox"/> 编辑报告 <input type="checkbox"/> 删测试计划 <input type="checkbox"/> 删报告
代码扫描	<input checked="" type="checkbox"/> 访问代码扫描 <input checked="" type="checkbox"/> 扫描任务管理 <input checked="" type="checkbox"/> 扫描方案管理
代码仓库	<input checked="" type="checkbox"/> 访问代码 <input checked="" type="checkbox"/> 创建仓库 <input checked="" type="checkbox"/> 拉取代码 <input checked="" type="checkbox"/> 本地推送代码 <input type="checkbox"/> 删普通分支 <input type="checkbox"/> 删保护分支 <input type="checkbox"/> 删保护标签 <input checked="" type="checkbox"/> 创建合并请求 <input type="checkbox"/> 在线编辑仓库文件 <input type="checkbox"/> 保护分支规则 <input checked="" type="checkbox"/> 创建版本 <input type="checkbox"/> 创标签 <input type="checkbox"/> 编辑版本 <input type="checkbox"/> 部署公钥 <input type="checkbox"/> 解锁锁定文件
持续集成	<input checked="" type="checkbox"/> 访问持续集成 <input checked="" type="checkbox"/> 持续集成管理 <input checked="" type="checkbox"/> 手动触发/停止构建 <input type="checkbox"/> 删构建记录 <input checked="" type="checkbox"/> 创建构建计划 <input type="checkbox"/> 修改构建计划 <input type="checkbox"/> 人工确认
持续部署	<input checked="" type="checkbox"/> 访问持续部署 <input checked="" type="checkbox"/> 持续部署管理 <input type="checkbox"/> 删部署记录 <input type="checkbox"/> 应用发布 <input type="checkbox"/> 应用编排
应用管理	<input checked="" type="checkbox"/> 访问应用管理 <input type="checkbox"/> 数据库变更管理

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。

全局设置 / 项目权限方案

项目权限方案

项目权限组 **项目成员管理** 项目管理

本页显示当前所选项目的成员列表，您可以对特定项目添加/删除成员，或者修改成员的权限组。

项目	权限组	成员名称	类型	项目权限组	操作
1	权限组	用户组	用户组	开发	关联权限组 移除
		用户组	用户组	开发	关联权限组 移除
		用户	用户	开发	关联权限组 移除
		鱼	用户	开发	关联权限组 移除
		用户	用户	开发	关联权限组 移除
		用户	用户	开发	关联权限组 移除
		用户	用户	开发	关联权限组 移除
		用户	用户	开发	关联权限组 移除
		用户	用户	开发	关联权限组 移除

+ 添加成员

Q 搜索

项目管理员
项目经理
开发
测试
产品
运维
默认配置
123-3ee5a38

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：

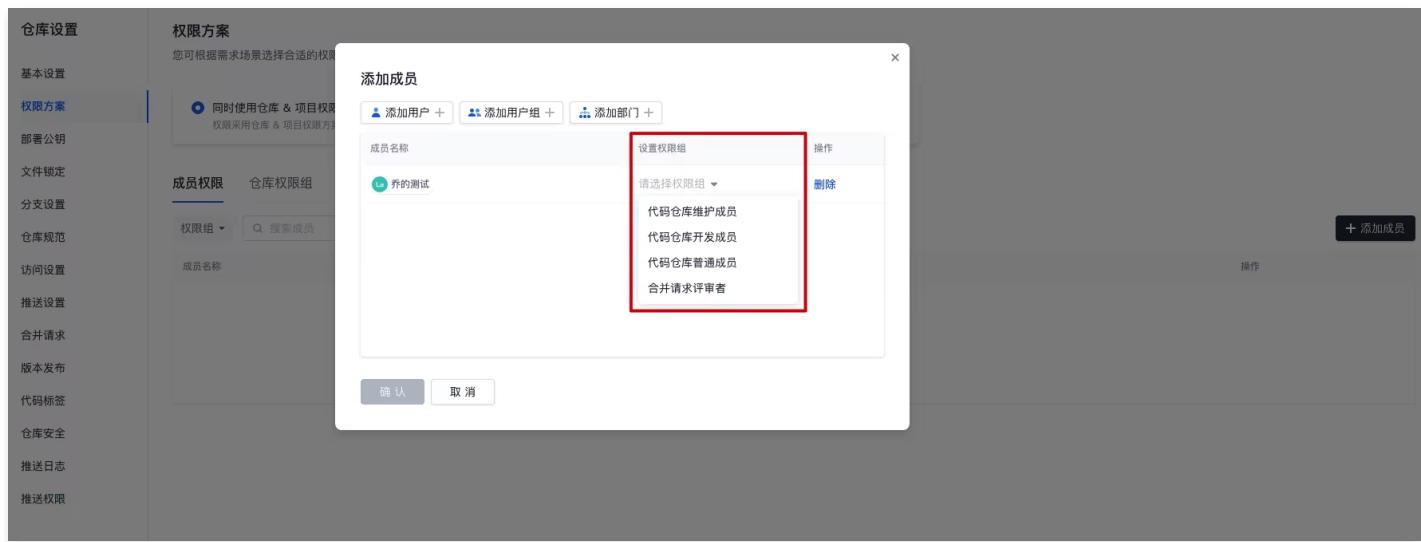
若选择“仅使用仓库权限方案”，仅团队负责人 / 管理员或团队中其他拥有代码权限的成员有权进入仓库页进行操作。

● 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。

The screenshot shows the CODING DevOps platform's repository management interface. On the left, there is a sidebar with various project and repository management options. The main area displays a list of existing repositories, each with a color-coded icon, the repository name, its status (e.g., '未填写仓库描述'), and some basic statistics like pull requests and last update time. At the top right of the main area, there is a prominent red-bordered button labeled '创建代码仓库' (Create Repository). The entire interface is clean and modern, typical of a cloud-based development tool.

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。

所属项目 *

所属项目

搜索

- newone
- 经典项目类型
- 度量测试
- test0001
- 蓝的测试
- 测试用例的测试项目
- 项目协同测试
- 123

添加 .gitignore 文件

添加分支模型 (仓库创建后将根据所选模型创建分支)

是否开源

私有仓库 (仅对仓库成员可见, 仓库成员可访问仓库)

公开仓库 (公开后, 任何人都可以访问代码仓库, 请谨慎考虑!)

仓库规范

完成创建 **取消**

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。

创建代码仓库 | 普通创建 | 模板创建

所属项目 *

仓库名称 *

仓库名称只支持字母、数字、下划线(_)、中划线(–)和点(.)的组合 0/100

仓库描述

请输入仓库描述

选择模板 *

预置模板 | 自定义模板

 spring-demo 基于简单的 Java 网页应用。	 ruby-on-rails-demo 基于简单的 Ruby on Rails 网页应用。	 ruby-sinatra-demo 基于简单的 Ruby Sinatra 网页应用。
 express-demo 基于简单的 Node.js 网页应用。	 android-demo 基于简单的 Android APP 应用。	 flask-demo 基于简单的 Python Flask 网页应用。

是否开源

私有仓库（仅对仓库成员可见，仓库成员可访问仓库）
 公开仓库（公开后，任何人都可以访问代码仓库，请谨慎考虑！）

完成创建 取消

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。



The screenshot shows a modal window titled "代码仓库还未初始化" (Code repository has not been initialized). It contains a message: "当您使用客户端克隆代码仓库时，终端提示的用户名是您在 CODING 个人设置 里填写的「手机」或「邮箱」。" (When you clone the code repository using the client, the terminal prompt username is the one you filled in under CODING Personal Settings, 'Phone' or 'Email'). Below the message is a URL bar showing "HTTPS https://e.coding.net/ [REDACTED]/lans-project/init.git". The main content area lists four initialization methods:

- 导入外部仓库 >
- 快速初始化仓库 >
- 使用命令行创建仓库 >
- 使用命令行推送已存在的仓库 >

导入外部仓库

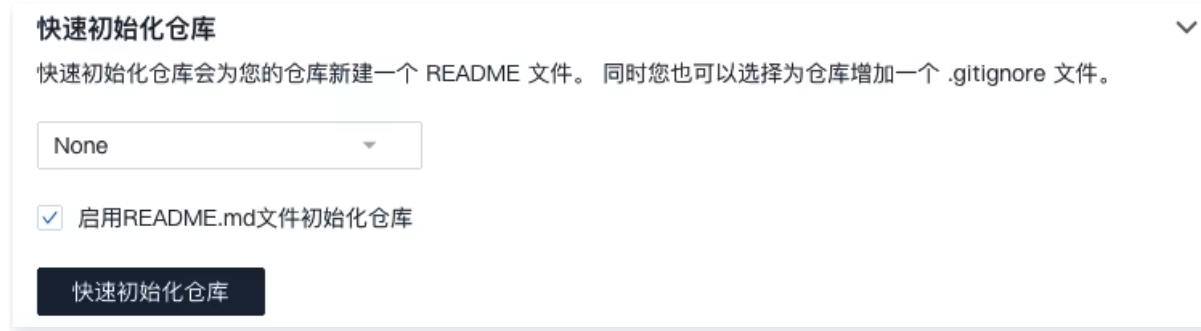
输入 Git 仓库地址后进行导入，完成初始化。



The screenshot shows the "Import External Repository" section. It includes a "Repository Clone Address" input field with placeholder text "请输入仓库克隆地址, 如 https://github.com/Coding/WebIDE.git" and a "Start Import" button.

快速初始化仓库

此方法将通过生成 README.md 文件完成初始化。



The screenshot shows the "Quick Initialize Repository" section. It includes a dropdown menu set to "None", a checked checkbox "Enable README.md file initialization for repository", and a "Quick Initialize Repository" button.

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。

使用命令行创建仓库

如果您还没有任何代码，可以通过命令行工具创建一个全新的 Git 仓库并初始化到本项目仓库中。

```
git clone https://e.coding.net, nit.git  
cd init  
echo "# init" >> README.md  
git add README.md  
git commit -m "first commit"  
git push -u origin master
```

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。

使用命令行推送已存在的仓库

如果您已有一个 Git 仓库，可以通过命令行工具将其直接推送到本仓库中。

```
git remote add origin http://e.staging-corp.coding.i...ting/restore.git  
git push -u origin master
```

导入或关联外部仓库

最近更新时间：2023-10-12 18:53:41

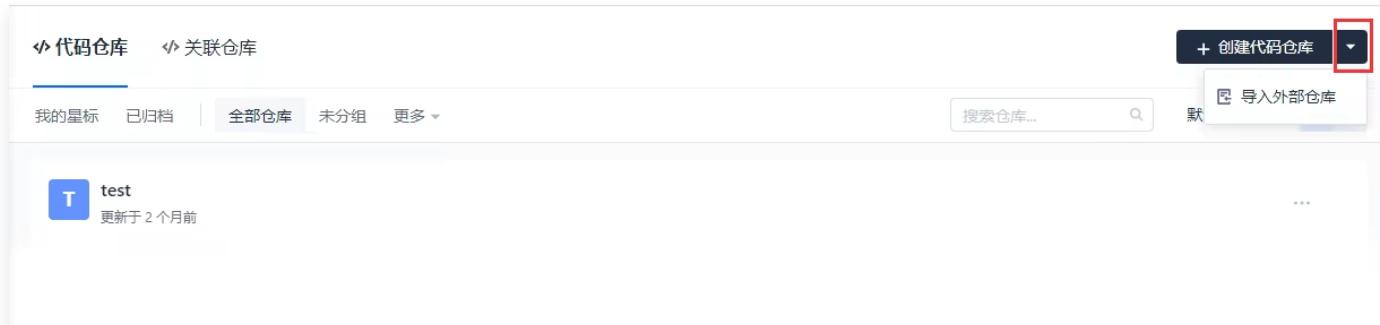
进入项目

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

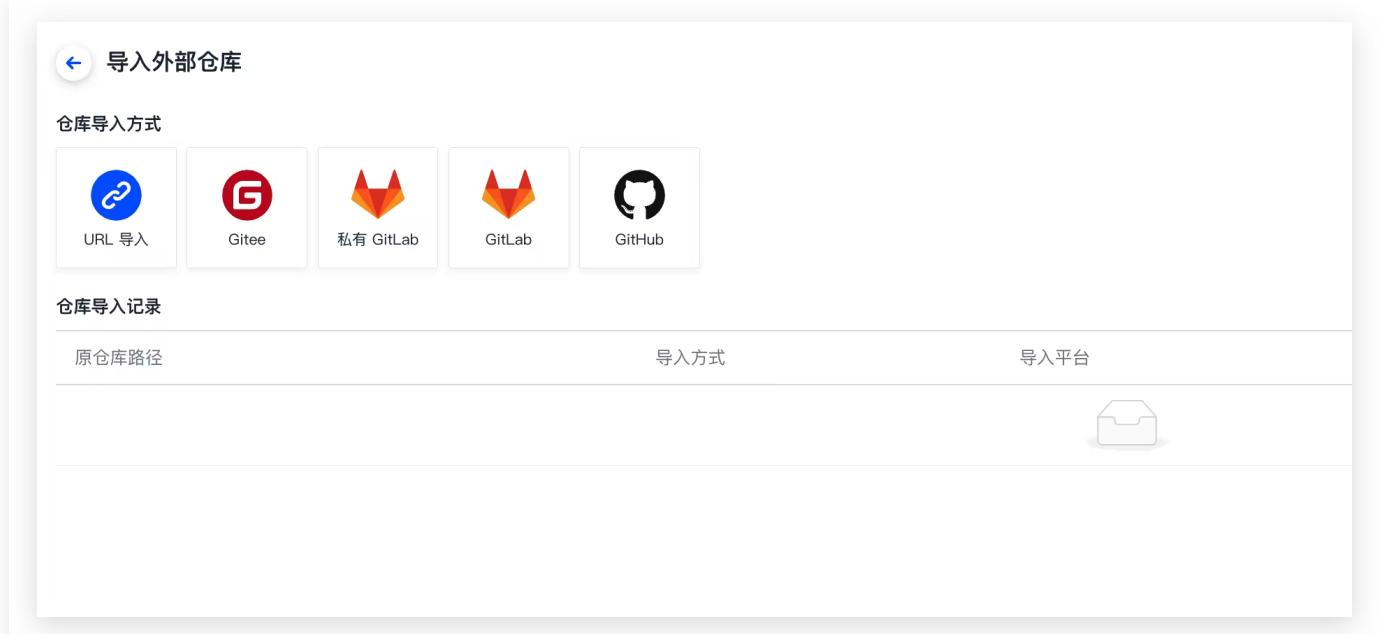
CODING 代码仓库不仅提供了一键导入外部仓库功能，还支持定时同步外部开源仓库。

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**。



The screenshot shows the CODING repository list interface. At the top right, there is a button labeled '+ 创建代码仓库' (Create Repository) and a dropdown menu. Below it is a search bar with the placeholder '搜索仓库...' and a magnifying glass icon. A red box highlights the '导入外部仓库' (Import External Repository) button, which is located in the top right corner of the main content area. The main content area displays a list of repositories, with one repository named 'test' visible.

2. 按照提示选择仓库来源类型。



The screenshot shows the 'Import External Repository' dialog box. At the top left is a back arrow and the title '导入外部仓库'. Below it is a section titled '仓库导入方式' (Repository Import Methods) containing five options: 'URL 导入' (Import via URL), 'Gitee', '私有 GitLab' (Private GitLab), 'GitLab', and 'GitHub'. Below this is a section titled '仓库导入记录' (Repository Import History). It has three columns: '原仓库路径' (Original Repository Path), '导入方式' (Import Method), and '导入平台' (Import Platform). The first row shows an empty path and an empty platform icon. A red box highlights the 'Import Platform' column header.

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。

The screenshot shows the 'Import External Repository / URL Import' page. On the left, there's a form with fields for 'Git 仓库 URL *' (containing 'http://.../zwk-test1230/0907.git'), '仓库名称 *' (containing '0907'), and '是否开源' (with '私有仓库' selected). Below these is a toggle switch for '代码扫描'. A note below the switch says: '开启代码扫描可以发现代码中的安全漏洞、功能缺陷等代码问题，结果将展示在合并请求详情中，辅助您进行代码评审。查看详情'.

A modal dialog titled '这是一个私有仓库' is displayed over the page. It contains a note: '① 需要填写您登录 [REDACTED] 时的用户名或邮箱、密码。为了安全考虑，我们不会记住它们。' It has input fields for '用户名或邮箱' and '密码', and two buttons at the bottom: '确认' (Confirm) and '取消' (Cancel).

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。

The screenshot shows the Tencent Cloud GitHub interface. At the top, there are navigation links for '代码仓库' (Code Repositories) and '关联仓库' (Associated Repositories). A button for '+ 创建代码仓库' (Create Code Repository) is also present. Below this, a search bar and filter options like '我的星标' (My Favorites), '已归档' (Archived), '全部仓库' (All Repositories), '未分组' (Unsorted), and '更多...' (More...) are visible. A search bar with placeholder '搜索仓库...' and a sorting dropdown labeled '默认排序' (Default Sort) are on the right. The main area displays a repository named 'test', updated 2 months ago. To the right of the repository card is a context menu with options like '移动到分组' (Move to Group) and '仓库设置' (Repository Settings), with '仓库设置' highlighted by a red box.

This screenshot shows the 'Repository Settings' page for the 'test' repository. The left sidebar lists various settings categories: 基本设置 (Basic Settings), 部署公钥 (Deployment Keys), 文件锁定 (File Locking), 分支设置 (Branch Settings), 访问设置 (Access Settings), 推送设置 (Push Settings), 合并请求 (Pull Requests), 版本发布 (Version Release), 代码标签 (Code Labels), 保护目录 (Protected Directories), 仓库安全 (Repository Security), and 同步信息 (Sync Information). The '同步信息' (Sync Information) tab is selected and displayed on the right. It includes sections for '自动同步触发时间' (Automatic Sync Trigger Time) set to 19:00 and '开启自动同步' (Enable Automatic Sync) checked, a '保存' (Save) button, and a '最近 30 天同步记录' (Recent 30 Days Sync Record) table. The table has columns for '同步类型' (Sync Type), '状态' (Status), and '上次同步时间' (Last Sync Time). The sync record table is currently empty, showing a placeholder icon.

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。

This screenshot shows the project homepage for 'test 01'. The left sidebar contains project management links: 项目概览 (Project Overview), 项目协同 (Project Collaboration), 代码仓库 (Code Repository), 持续集成 (Continuous Integration), 持续部署 (Continuous Deployment), 代码扫描 (Code Scan) [BETA], 应用管理 (Application Management), 制品管理 (Product Management), 文档管理 (Document Management), and API 管理 (API Management) [BETA]. The '代码仓库' link is currently selected. The main content area shows repository statistics: master branch, 6 branches, 26 tags, 641 commits, 128 pushes, and 128 pull requests. A prominent '强制同步' (Force Sync) button is located at the top right of the repository header. The repository code view shows commit history for 'frontend' and 'backend' branches.

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:ios-go/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。



The screenshot shows the CODING web interface for creating a new repository. At the top, there is a navigation bar with links for 'gitlab', '浏览' (Browse), '提交' (Commit), '分支' (Branch), '合并请求' (Merge Request), '版本' (Version), '对比' (Compare), '设置' (Settings), and a button for '+ 创建代码仓库' (Create Repository). Below the navigation bar, there is a large input field with 'SSH' selected and the URL 'git@gitlab.com:ios-go/ios-go.git'. This URL is highlighted with a red box. Below the input field, there are four options listed: '导入外部仓库 (目前仅支持公开仓库)' (Import External Repository (Currently only supports public repositories)), '快速初始化仓库' (Quick Initialize Repository), '使用命令行创建仓库' (Create Repository via Command Line), and '使用命令行推送已存在的仓库' (Push Existing Repository via Command Line). Each option has a right-pointing arrow next to it.

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。

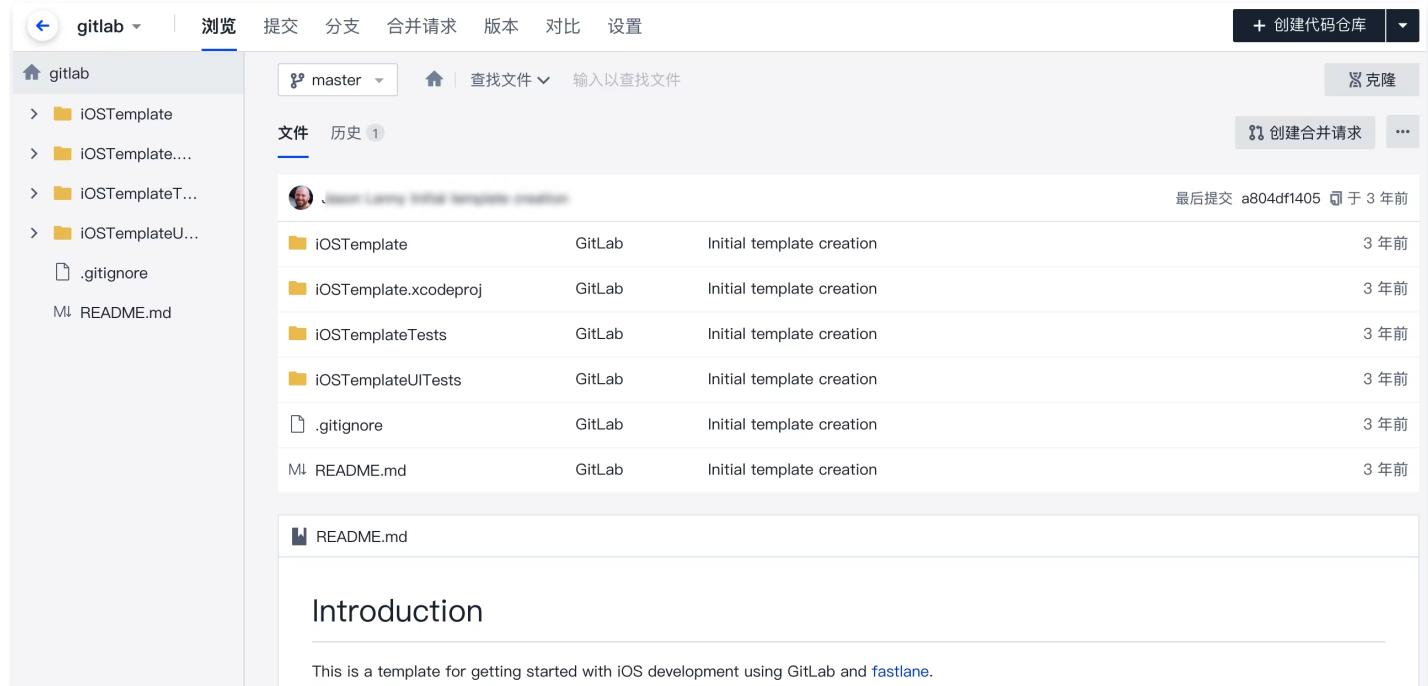
```
git push --mirror 目标仓库地址
```

```
/Volumes/CODING-Help/ios-go.git master git push --mirror git@e.coding.net:StrayBirds/demo/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。



The screenshot shows the GitLab interface for a repository named 'gitlab'. The sidebar on the left lists repository files: iOSTemplate, iOSTemplate...., iOSTemplateT..., iOSTemplateU..., .gitignore, and README.md. The main area displays a commit history for the 'master' branch. The first commit, made by 'GitLab', is titled 'Initial template creation' and was pushed 3 years ago. It created the files iOSTemplate, iOSTemplate.xcodeproj, iOSTemplateTests, iOSTemplateUITests, .gitignore, and README.md.

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。

The screenshot shows the left sidebar with various project management options like Project Overview, Project Cooperation, and Code Scan. The main area is titled 'Associated Repositories' with tabs for 'Code Repository' and 'Associated Repositories'. It includes filters for Repository Source (All), Authentication Method (All), Enabled Project Modules (All), and Contact Persons (All). A search bar and a 'Search Repository...' button are also present. Below the filters, a table lists one repository: 'Repository Name' (redacted), 'Repository Source' (GitHub), 'Authentication Method' (OAuth), 'Enabled Project Modules' (All modules), 'Contact Person' (Primary account), 'Associated Time' (2021-06-11 11:38), and an ellipsis for more actions.

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。

关联代码仓库

仓库来源

认证方式

授权人

主账号, [刷新 GitHub OAuth 认证](#)

代码仓库 *

确认关联 取消

This dialog box allows users to associate a repository from various sources. The 'Repository Source' dropdown is set to 'GitHub'. The 'Authorization Method' dropdown is set to 'OAuth'. The 'Authorized Person' field shows 'Primary account'. The 'Repository' dropdown is empty. At the bottom, there are 'Associate' and 'Cancel' buttons.

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。

关联代码仓库

仓库来源

CODING GitHub GitLab 私有 GitLab Gitee 工蜂 通用 git 仓库

认证方式 OAuth 授权人 您尚未认证 GitLab OAuth, 前往认证

代码仓库 *

确认关联 取消

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。

关联代码仓库

仓库来源

CODING GitHub GitLab 私有 GitLab Gitee 工蜂 通用 git 仓库

认证方式 个人令牌

代码仓库 *

确认关联 取消

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。

The screenshot shows the GitHub 'Applications' settings page under 'Authorized OAuth Apps'. It lists four applications:

- CODING DevOps**: Last used within the last week - Owned by Coding. The 'Revoke' button for this entry is highlighted with a red box.
- iconfont**: Last used within the last 6 months - Owned by iconfont-cn
- Linode**: Last used within the last 3 months - Owned by linode
- Server酱**: Last used within the last 12 months - Owned by easychen

At the bottom, there is a note: **(i) Read more about connecting with third-party applications at [GitHub Help](#).**

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。



The screenshot shows the GitHub repository settings interface. The top navigation bar includes 'ji' (repository name), 'st' (shortcuts), and a '切换仓库' (switch repository) button. Below the navigation are tabs: '代码' (Code), '合并请求' (Pull Requests), '版本' (Versions), '云原生构建' (Cloud Native Builds), and '设置' (Settings), with '设置' being the active tab. On the left, a sidebar lists repository settings categories: '仓库设置' (Repository Settings), '基本设置' (Basic Settings) (which is selected and highlighted in blue), '权限方案' (Access Control), '部署公钥' (Deployment Keys), '文件锁定' (File Locking), '分支设置' (Branch Settings), '仓库规范' (Repository Rules), '访问设置' (Access Settings), '推送设置' (Push Settings), and '合并请求' (Pull Requests). The main content area has sections for '仓库描述' (Repository Description) with a placeholder '请输入仓库描述' (Enter repository description) and a '更改图标' (Change icon) button. Below this is a section for '设为自定义模板仓库' (Set as custom template repository) with a note about creating new repositories from the same structure. A '保存' (Save) button is present. Another section for '仓库容量管理' (Repository Capacity Management) shows usage information: '已使用 40.00 KB / 总容量 5.00 GB' (Used 40.00 KB / Total capacity 5.00 GB) and a link to '修改仓库容量上限' (Modify repository capacity limit). At the bottom of the main content area, a button labeled '仓库清理' (Repository Cleanup) is highlighted with a red box, with a note below it stating '仓库清理使用 git gc 优化代码仓库存储空间, 压缩存储库对象, 减少存储库磁盘占用。' (Repository cleanup uses git gc to optimize code repository storage space, compress repository objects, and reduce disk usage).

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。

代码仓库 关联仓库

我的星标 已归档 全部仓库 未分组 更多

A api-test 未填写仓库描述
0 更新于 1年前

C cd-multi-env-demo 未填写仓库描述
更新于 2年前
...
移动到分组
仓库设置
归档仓库
重置仓库
删除仓库

P public1 公开 未填写仓库描述
0 更新于 1年前

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。

代码仓库 关联仓库

我的星标 已归档 全部仓库 未分组 更多

A api-test 未填写仓库描述
0 更新于 1年前
...
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。

代码仓库 关联仓库

我的星标 已归档 全部仓库 未分组 更多

A api-test 未填写仓库描述
0 更新于 1 年前

C cd-multi-env-demo 未填写仓库描述
0 更新于 2 年前

P public1 公开 未填写仓库描述
0 更新于 1 年前

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。

- 在代码仓库页面，单击右上角回收站图标。

代码仓库 关联仓库

我的星标 已归档 全部仓库 未分组 更多 搜索仓库... 默认排序

A api-test 未填写仓库描述
0 更新于 1 年前

C cd-multi-env-demo 未填写仓库描述
0 更新于 2 年前

C cd-multi-env-deploy-d... 未填写仓库描述
0 更新于 2 年前

J jack-docker-push-demo 未填写仓库描述
0 更新于 2 年前

P public1 公开 未填写仓库描述
0 更新于 1 年前

- 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。

回收站

仓库名称	删除操作人	删除时间	剩余时间	操作
test		2021-10-26 15:02:02	30 天	恢复

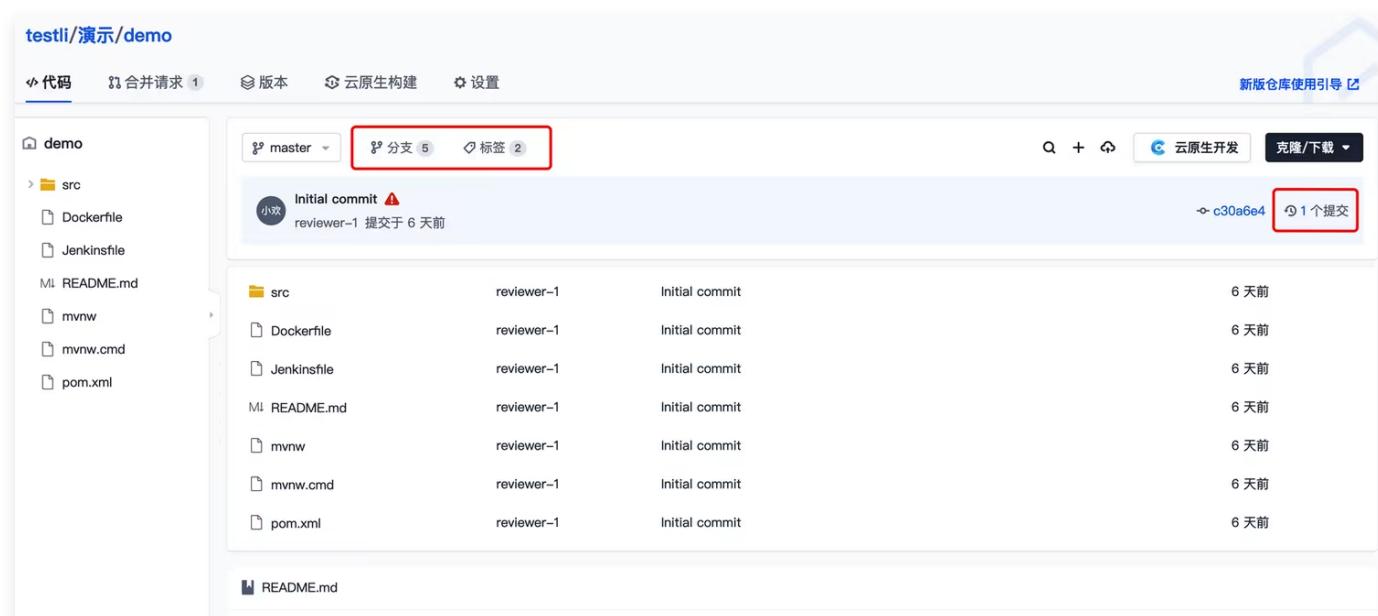
代码对比

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。



The screenshot shows the CODING repository interface for the project 'testli/演示/demo'. On the left, there's a sidebar with a tree view of files and folders under the 'demo' repository. In the center, there's a list of commits. At the top of this list, there's a dropdown menu with options: 'master' (selected), '分支 5' (highlighted with a red box), and '标签 2'. To the right of the dropdown, there's a commit card for 'Initial commit' by 'reviewer-1' submitted 6 days ago. Below the commit card is a list of individual file commits. At the bottom of the commit list, there's a file named 'README.md'. On the far right, there are buttons for '云原生开发' (Cloud Native Development) and '克隆/下载' (Clone/Download). A red box highlights the '分支 5' button.

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**。



The screenshot shows the CODING repository interface for the project 'ji...'. On the left, there's a sidebar with a tree view of files and folders under the 'ji...' repository. In the center, there's a list of branches. At the top of this list, there's a dropdown menu with options: '活跃分支' (Active Branches), '我的分支' (My Branches), '保护分支' (Protected Branches), '只读分支' (Read-only Branches), '所有分支' (All Branches), and '过时分支' (Outdated Branches). To the right of the dropdown, there's a search bar and a '创建合并请求' (Create Merge Request) button. Below the dropdown, there are two branches listed: 'master' (marked as '默认分支' - Default Branch) and 'dev'. Each branch has a small preview card showing the last commit. On the far right, there's a context menu with several options: '创建合并请求' (Create Merge Request), '对比' (Compare) (highlighted with a red box), '设为只读' (Set to Read-only), '编辑备注' (Edit Notes), 'cherry-pick' (Cherry-pick), 'revert' (Revert), and '下载分支' (Download Branch).

- 以标签列表为例，选择指定标签，单击 并选择对比。

The screenshot shows a list of four tags: v4.0, v3.0, v2.0, and v1.0. Each tag entry includes a user profile picture, the tag name, a creation date, and a commit hash. To the right of each tag is a set of buttons: 'Create Version Description', 'Download', 'Compare' (which is highlighted with a red box), and 'Delete'. Below the list is a pagination bar showing '1 - 4个, 共 4个' and a dropdown for '15 条/页'.

- 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。

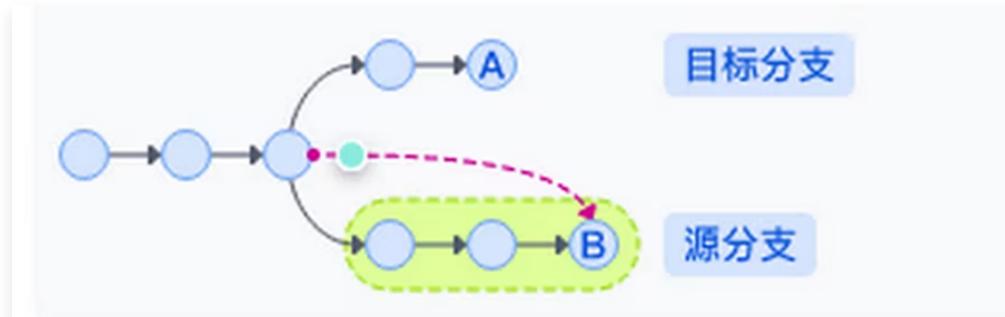
The screenshot shows a code diff interface for a file named '.coding-ci.yml'. It displays 22 changes (22 additions / 0 deletions). A comment section is visible below the code. In the top right corner, there are buttons for 'Browse Code', 'Create Label', and 'Compare' (which is highlighted with a red box). Below these buttons are links for 'Expand All', 'Collapse All', and 'Switch to Plain Mode'.

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。

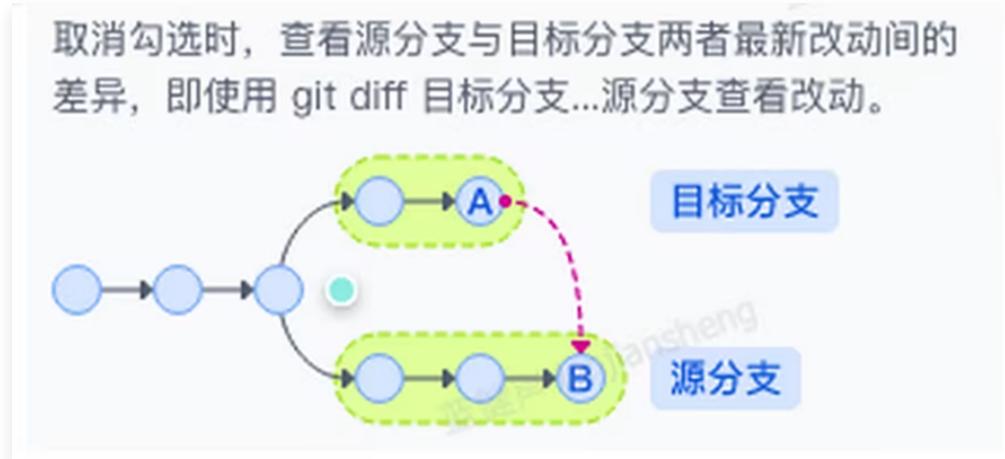
The screenshot shows the 'Compare' page. At the top, it says 'Source Branch: dev' and 'Target Branch: master'. There is a checkbox for 'Compare against common ancestor' and a note about merge conflicts. Below this is a dropdown menu labeled 'Branches, Tags, and Versions' which is currently set to 'Branches'. Underneath, it lists 'dev' and 'master'. The 'Compare Branches' tab is highlighted with a red box. At the bottom right are buttons for 'Expand All', 'Collapse All', 'Switch to Left Mode', and 'Advanced Options'.

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**。



The screenshot shows the 'Create Repository' interface. On the left, there's a sidebar with various project management options like Project Overview, Project Collaboration, and Code Repository (which is currently selected and highlighted with a blue background). The main area has tabs for '普通创建' (Normal Creation) and '模板创建' (Template Creation). A note says 'In other websites already have a repository? [Import](#)'. The 'Repository Type' dropdown is set to 'SVN Repository', and its corresponding input field is also highlighted with a red box. There's another dropdown for 'GIT Repository'. Below these fields is a section for 'Initial Repository' with a checkbox for 'Create SVN repository recommended layout (tags, branches, trunk)' and a link to 'SVN Repository Usage Guide'. At the bottom are '完成创建' (Finish Creation) and '取消' (Cancel) buttons.

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。

This screenshot shows the SVN repository interface for a project named 'svn_test'. On the left, there's a sidebar with branches ('branches', 'trunk', 'tags'). The main area displays a commit history table with columns: 名称 (Name), 提交人 (Committer), 更新时间 (Last Commit), and 提交 (Commit). The commits are:

名称	提交人	更新时间	提交
branches	[redacted]	2023-08...	r1 Initial project layout
trunk	[redacted]	2023-08...	r1 Initial project layout
tags	[redacted]	2023-08...	r1 Initial project layout

5. 单击右上角检出可以看到这个仓库的 SVN 地址。

This screenshot shows the same SVN repository interface as above, but with a red box highlighting the '检出' (Checkout) button in the top right corner of the main content area. This button is used to copy the SVN URL for cloning the repository.

⚠ 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：

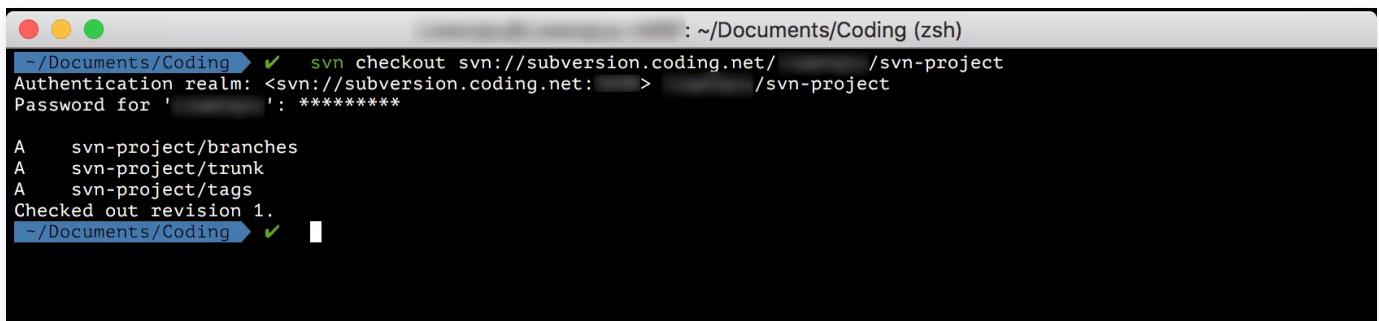
```
svn, version 1.9.7 (r1800392)  
compiled Feb 28 2018, 15:54:50 on x86_64-apple-darwin17.3.0  
Copyright (C) 2017 The Apache Software Foundation.  
This software consists of contributions made by many people;  
see the NOTICE file for more information.  
Subversion is open source software, see http://subversion.apache.org/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

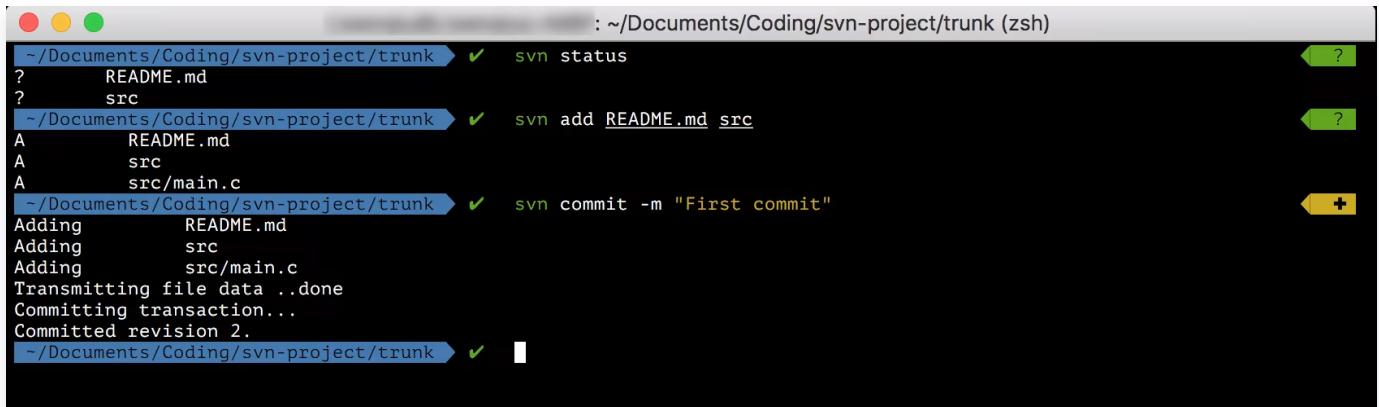
The following authentication credential caches are available:
* Plaintext cache in /Users/Liwengiu/.subversion
* Mac OS X Keychain
```

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:



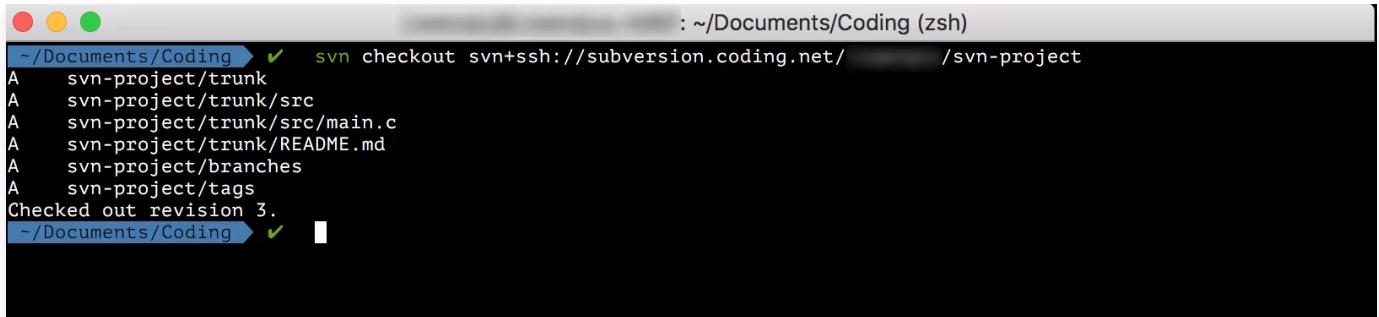
```
: ~/Documents/Coding (zsh)
~/Documents/Coding ➤ ✓ svn checkout svn://subversion.coding.net/          /svn-project
Authentication realm: <svn://subversion.coding.net:      >          /svn-project
Password for '': *****
A   svn-project/branches
A   svn-project/trunk
A   svn-project/tags
Checked out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

5. 接下来可以使用 add、commit 命令往仓库中新添加内容:



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

6. 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：

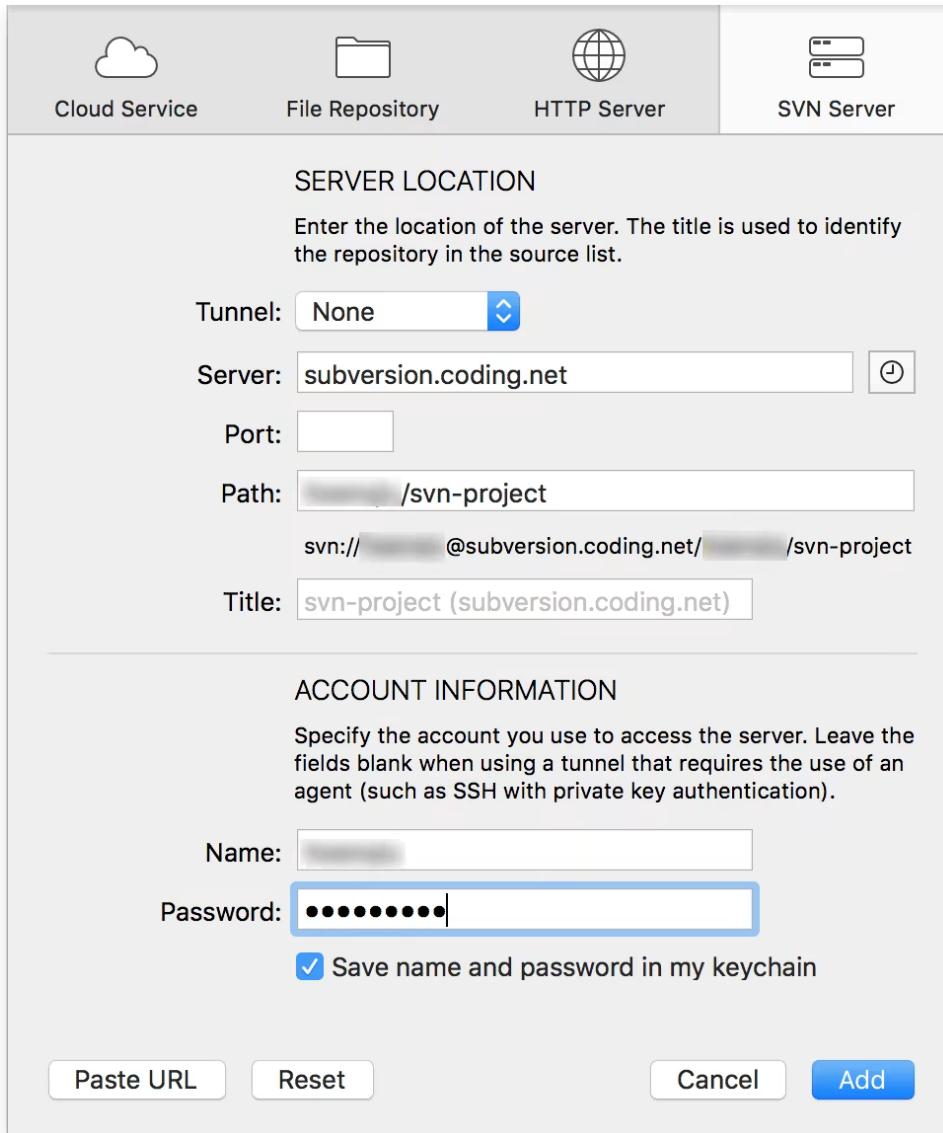


```
: ~/Documents/Coding (zsh)
~/Documents/Coding ➤ ✓ svn checkout svn+ssh://subversion.coding.net/          /svn-project
A   svn-project/trunk
A   svn-project/trunk/src
A   svn-project/trunk/src/main.c
A   svn-project/trunk/README.md
A   svn-project/branches
A   svn-project/tags
Checked out revision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

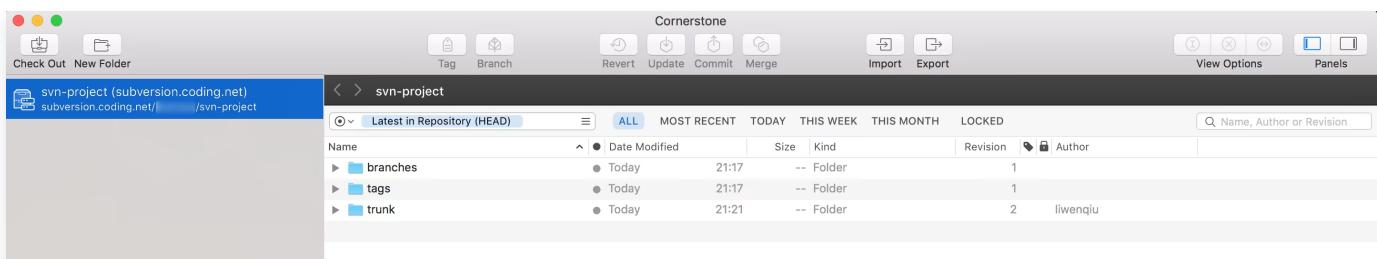
Cornerstone 工具

您可以通过 Cornerstone 来使用 SVN 仓库。

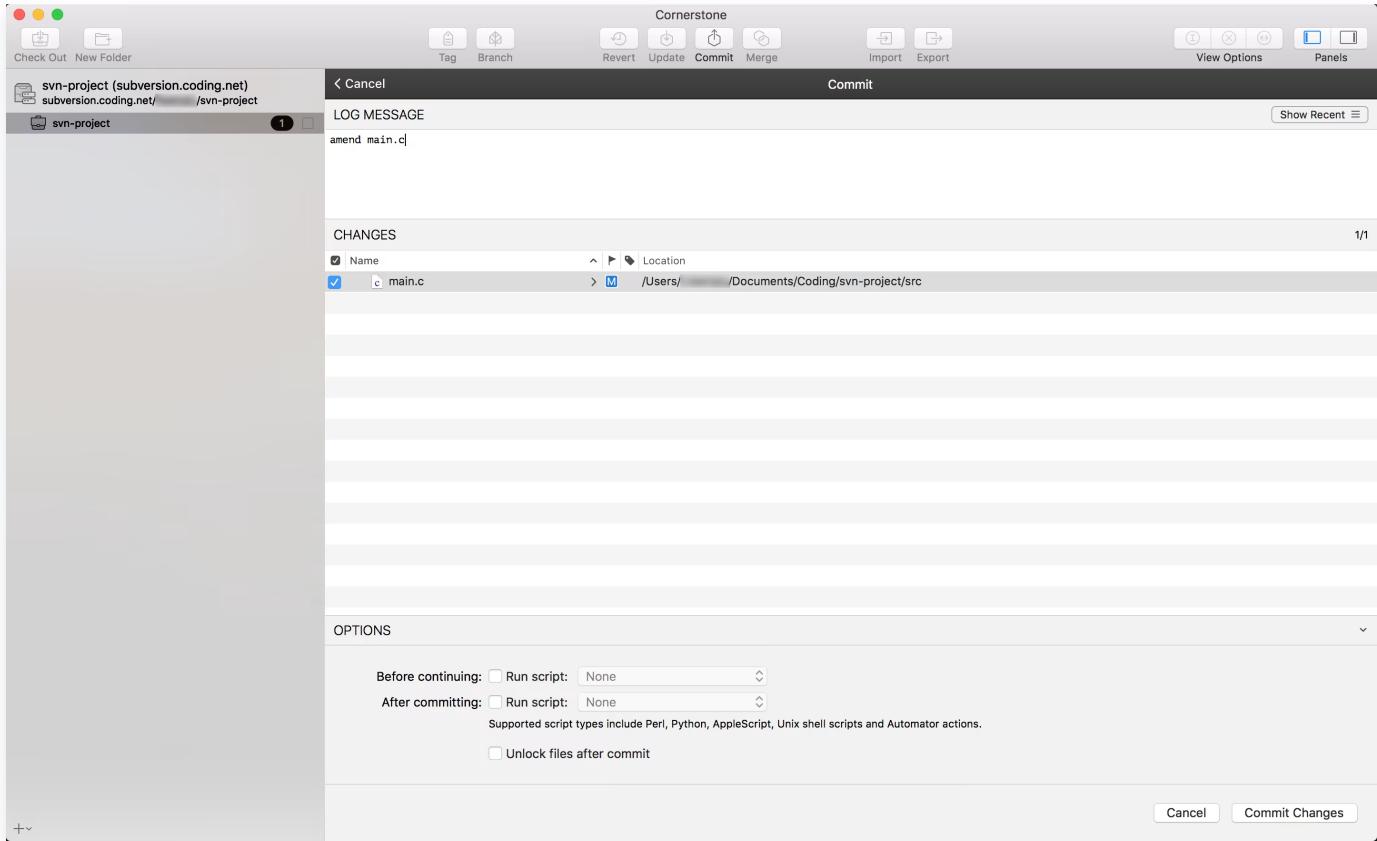
1. 打开 Cornerstone 后，单击 **Add Repository** 来添加 SVN 仓库（请将地址替换为您的 SVN 仓库地址）引用。



然后可以看到仓库的内容如下：



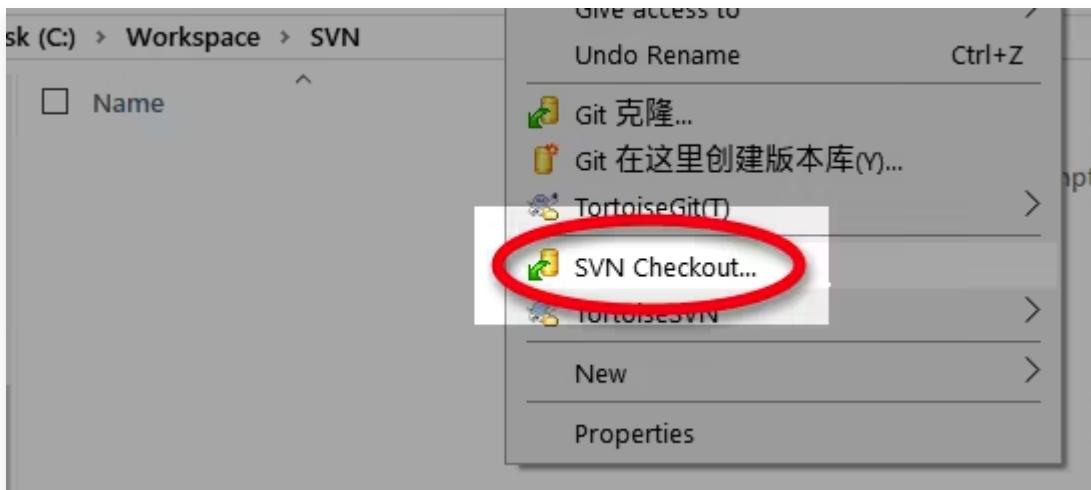
2. 把仓库 checkout 出来，并且编辑文件之后，就可以 commit 进仓库，如下图：



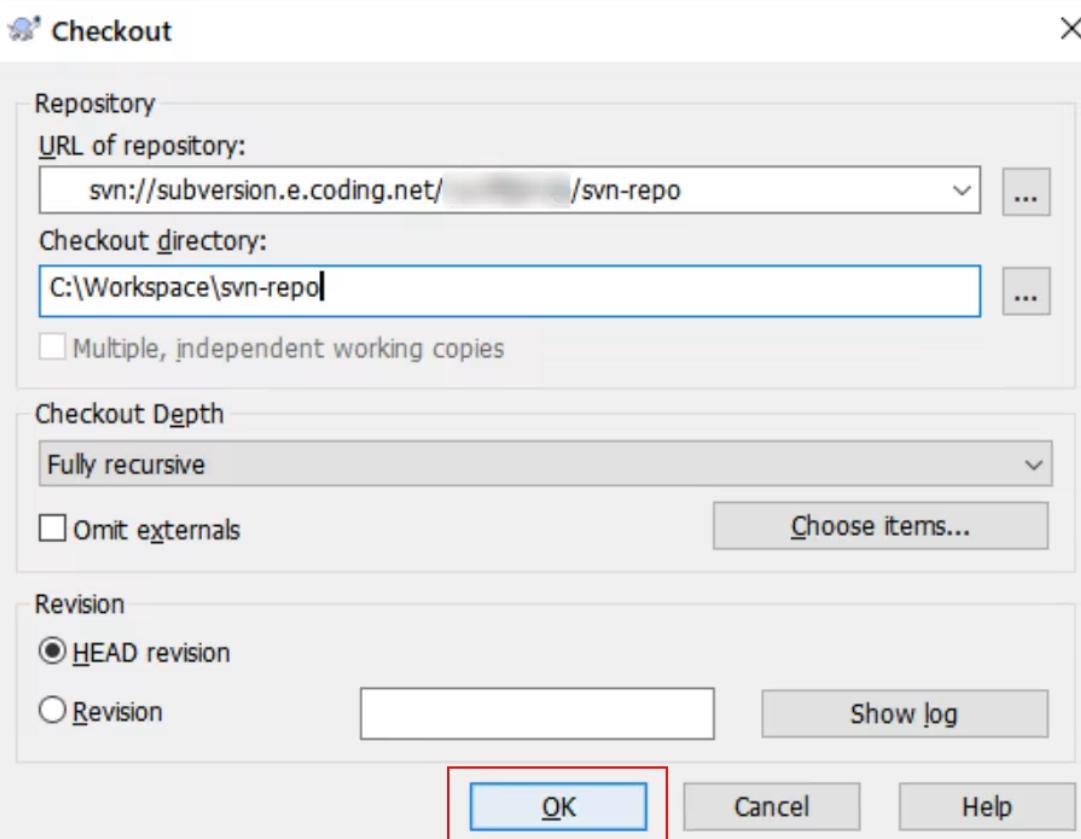
Windows 环境

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。

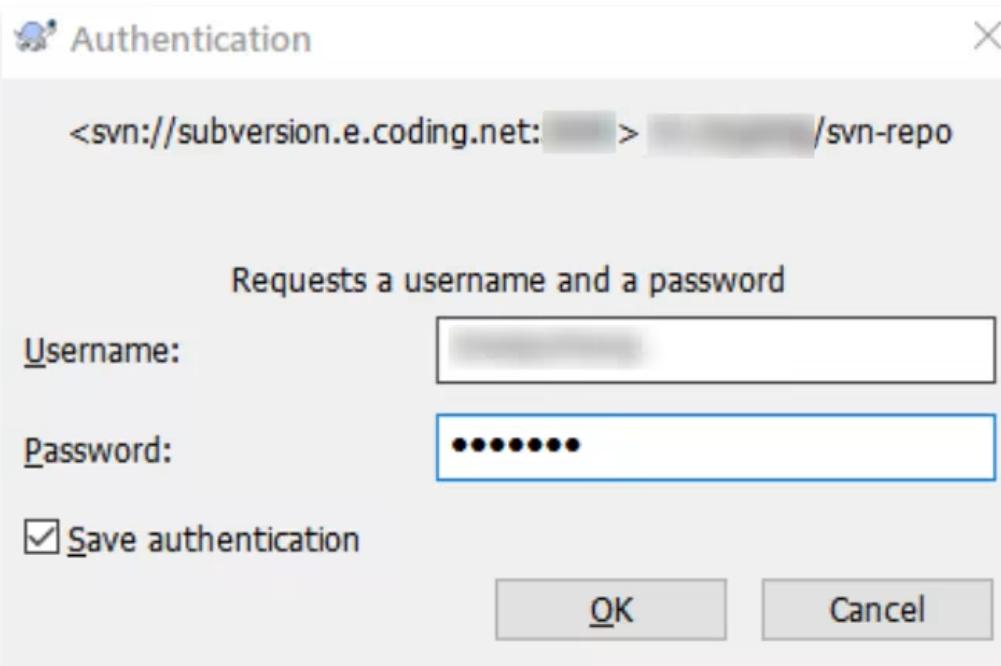


选择 `checkout` 把 SVN 仓库 `checkout` 到本地（请将地址替换为您的 SVN 仓库地址）。

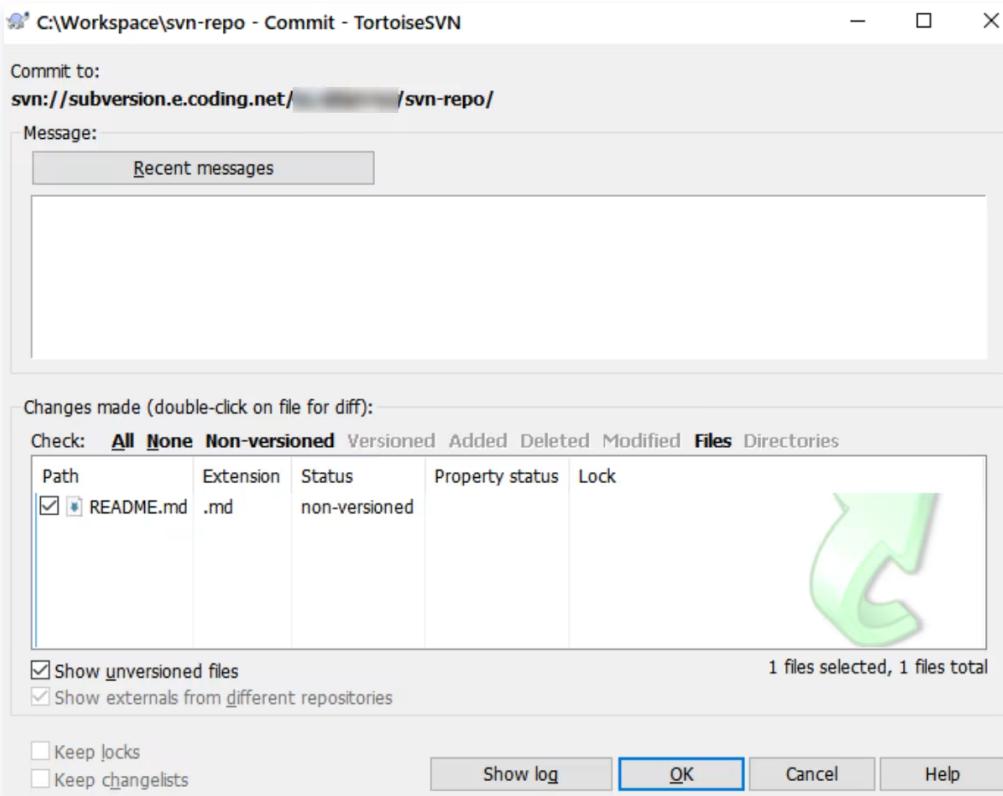


2. 第一次 `checkout` 需要输入用户名和密码。勾选“Save authentication”保存认证信息后，就不需要每次都输入密码。

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。

① 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
```

① 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

- 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。



```
~/ssh cat id_ed25519.pub
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5A
83328@aa.com
```

- 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。



仓库设置

部署公钥

部署公钥用以部署项目，只针对当前代码仓库，可设置拥有只读或者读写权限（默认为只读）。不能跟个人公钥通用，如需要设置个人公钥，[请点击这里](#)。

CODING 的 SSH 公钥指纹
SHA256:jc
MD5:98

仓库已使用公钥 团队内可用公钥

新建部署公钥

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。

团队设置中心 / 功能设置 / 团队部署公钥

创建团队部署公钥

公钥名称：自定义公钥名称，可不填

公钥内容：

```
AAAAB3NzaC1yc2EAAQABAAQADQHil6/Zs8DVJduqR0DH08s5JDT4SpnXS+jv  
LjkAuJl2G3nBYTdtjfvMx616i6Lvx3MMecogYyujhx/k911+8ZgfpJawNTfNuw/JlLaH  
989QLdM7F2NaJi3OHV8484Z6KkVvEXyO99HIG/oCNYDyp/78kd0kzBb9ghlxKG3M8Y  
RHd5udv6VErKS5qQH+i9WTsEaf2VZj/EPAiYLULHxJ7oikuKmwZ8CvlyqhPxCbHG4wd  
KprAtpdopiaxi5mmH/+pt0vzK01RgZ/ibhVkfAAmxOCWxTSNetqAlYd+z01/6880UJIIWqf  
bzMe7FSp4CLezlln8Ulowb8xw+yRoYMI2Lqon coding@MBP
```

有效期至：2024-07-12

修改 | 删除

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/known_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:ox9ko2YsRDgwp4C9im0Tha9FWxAxhfe7H7yLIXhcT5A [ ]:11900 (ECDSA)
256 SHA256:u2Xk2ekDfmrav2FALTnPPnGX9seyiZEk0vsFiGyp/EKo [ ]:28524 (ECDSA)
256 SHA256:za8qm0BYDLKBCx+hG4gT4/0iq06ZR/w00JhMoqJIWtA [ ]:ECDSA
2048 SHA256:jok3FH7q5LJ6qvE7iPNehBgXRw51ErE77S0Dn+Vg/Ik e.coding.net, (RSA)
~ |
```

验证 MD5 算法指纹

查看本地 `.ssh/known_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。

The screenshot shows the CODING repository browser interface. At the top, there are tabs for '代码' (Code), '合并请求' (Merge Requests), '版本' (Version), '云原生构建' (Cloud Native Build), and '设置' (Settings). Below the tabs, there are buttons for 'master' (branch), '分支' (Branches), '标签' (Tags), and a search bar. On the left, there's a file tree with 'public' and 'test' folders. In the center, there's a commit history with one entry from 'coding.net' (陈星) titled 'Initial commit'. To the right, there's a section for cloning the repository. It shows two options: 'HTTPS' and 'SSH', with 'SSH' being selected. Below these options is a note: '克隆代码仓库时，终端提示的用户名是您在 CODING 个人设置 里填写的手机或邮箱'. A red box highlights the 'SSH' button. Further down, there's a text input field containing the SSH URL 'git@e.coding.net:chaylee/typical/node-express-example.git' and a '下载 ZIP' (Download ZIP) button.

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

The screenshot shows a terminal window with the following command and output:

```
/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)+ ?
```

The terminal window has a dark theme with light-colored text. The status bar at the bottom shows the date and time: '11141 11:35:16'.

分支管理

创建分支

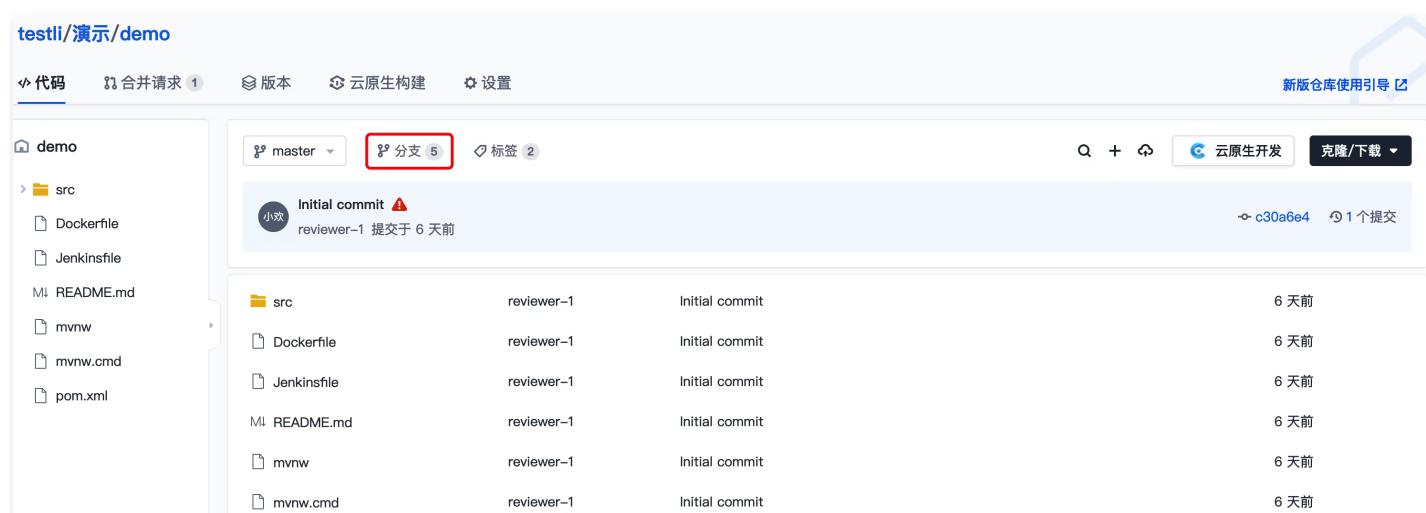
最近更新时间：2024-08-19 11:48:11

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。

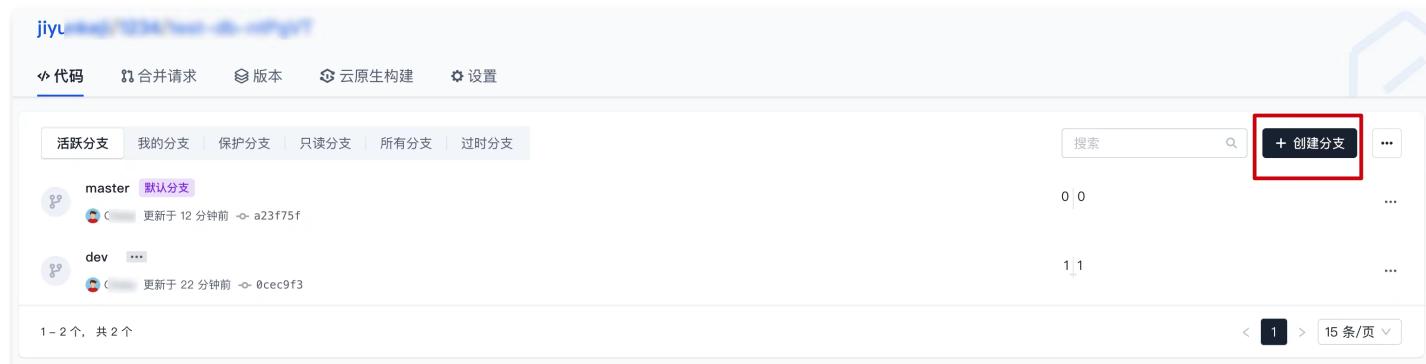


The screenshot shows the 'demo' repository details page. In the top navigation bar, the '分支' (Branches) tab is selected, indicated by a red border. Below the navigation, there's a search bar and a '+' button. On the left, a sidebar lists repository files like README.md, Dockerfile, Jenkinsfile, etc. The main area displays a table of branches with their commit history:

分支	提交者	提交信息	时间
master	reviewer-1	Initial commit	6 天前
src	reviewer-1	Dockerfile	6 天前
Jenkinsfile	reviewer-1	Jenkinsfile	6 天前
README.md	reviewer-1	Initial commit	6 天前
mvnw	reviewer-1	mvnw	6 天前
mvnw.cmd	reviewer-1	mvnw.cmd	6 天前
pom.xml	reviewer-1	pom.xml	6 天前

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。



The screenshot shows the 'demo' repository details page. The '分支' (Branches) tab is selected. A red box highlights the '+ 创建分支' (Create Branch) button in the top right corner of the header. Below the header, there are tabs for '活跃分支' (Active Branches), '我的分支' (My Branches), '保护分支' (Protected Branches), '只读分支' (Read-only Branches), '所有分支' (All Branches), and '过时分支' (Outdated Branches). The 'master' branch is listed as the default branch. The 'dev' branch is also listed. At the bottom, there's a pagination bar showing '1 / 1'.

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。

The screenshot shows the 'Code' tab of a repository interface. It lists two branches: 'master' (marked as the default branch) and 'dev'. The 'master' branch was updated 14 minutes ago with commit 'a23f75f'. The 'dev' branch was updated 24 minutes ago with commit '0cec9f3'. A sidebar on the right provides options like 'Create Merge Request', 'Compare', 'Set Read-Only', 'Edit Note' (which is highlighted with a red box), 'Cherry-Pick', 'Revert', and 'Download Branch'. The main area shows a summary of 1 merge request and 1 pull request.

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。

The screenshot shows the 'Branch Settings' page for a repository named 'android'. The left sidebar lists various settings categories: Basic Settings, Permission Schemes, Deployment Public Keys, File Locking, Branch Settings (which is selected), Repository Rules, Access Settings, Push Settings, Merge Requests, Version Release, and Repository Security. The main content area is titled 'Branch Settings' and contains three sections: 'Default Branch' (set to 'master'), 'Protect Branch Rules' (empty table), and 'Hidden Branch' (empty table). A note states that hidden branch access priority is determined by user groups.

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。

The screenshot shows the 'Configure Permissions' page under the 'Permission Group' section. The left sidebar lists project management, basic settings, menu management, notification settings, member configuration, members, and permission groups (which is selected). The main content area shows a grid of permissions categorized by project, test management, code scanning, and code repository. In the 'Code Repository' section, the 'Delete Protected Branch' and 'Manage Branch Protection Rules' checkboxes are highlighted with a red border.

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。

仓库设置

分支设置

默认分支

master

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

保护分支规则

+ 添加分支规则

保护分支规则	保护分支数量	操作
dev/001	暂无数据	+ 新增用户组/成员 ✎ 删除

说明：隐藏分支访问权限优先级为 用户 > 用户组 > 所有用户，当某个成员属于多个用户组时，分支访问权限以所在的未排除的用户组为依据。

隐藏分支

+ 添加隐藏分支

dev/001	所有人 (2)	项目管理员 (undefined)	允许访问	允许访问
			<input type="radio"/>	<input checked="" type="radio"/>

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。

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

添加保护分支规则

分支规则名称

|

示例: *-stable 或 production/*

添加

禁止强制推送

开启禁止强制推送后, 将不允许使用 git push -f 强制修改分支历史。

开启状态检查

开启状态检查后, 所有合并到此分支的合并请求必须在全部 CI 任务都通过之后才能合并。

[查看设置 CI 触发规则的帮助文档](#)

自动添加分支管理员为评审者

开启后, 所有合并到此分支的合并请求, 将自动添加所有分支管理员为评审者。

开启代码所有者评审

开启代码所有者评审后。所有合并到此分支的合并请求, 如果有修改代码所有者的文件, 则必须经过相应的代码所有者评审之后才允许合并。

合并请求允许合并授权数量

请选择

设置合并请求必须有多少位分支管理员授权之后才允许合并到目标分支。

- 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。

分支管理员 | + 添加分支管理员

分支管理员为可选项。添加分支管理员后，所有合并请求需分支管理员「允许合并」。分支管理员也受保护分支限制，需创建合并请求修改分支，若将其设置为「允许直接 Push」则可以直接推送代码修改分支。

管理员	权限	操作
阿乔	<input checked="" type="checkbox"/> 允许直接推送	移除

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：

```
adolf@P_ADOLFLAN-MB0:/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: 你没有权限推送到保护分支 (refs/heads/api) , 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/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：2023-12-29 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。

- 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。

The screenshot shows the 'Branch Settings' page under the 'Settings' tab of a repository's details. On the left sidebar, 'Branch Settings' is selected. In the main area, there's a 'Default Branch' dropdown set to 'master'. Below it, a note says: '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。' Under 'Protected Branch Rules', there's a note: '开启保护分支后，创建合并请求并邀请其他成员评审代码，其他成员「允许合并」后可自行合并分支。' A table lists 'Protected Branch Rules' and 'Protected Branch Count'. In the 'Hidden Branches' section, there's a note: '说明 1：隐藏分支访问权限优先级为 用户 > 用户组 > 所有用户。当某个成员属于多个用户组时，分支访问权限以所在的未排除的用户组为依据。说明 2：分支规则为前缀匹配，即 分支规则为 test，将会匹配以 test 为前缀的所有分支。' A table lists 'gh-pages' with 'All (3)' and 'Project Administrators (1)' under 'Allow Access'. Buttons for '+ Add Hidden Branch' and '+ Add User Group / Member' are visible.

- 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。

- 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），在这种情况下用户 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。

团队设置中心 / 全局设置 / 资源权限方案

仓库权限方案

代码仓库开发成员 系
具备代码仓库访问、代码仓库写入、版本发布管理权限。

权限组详情

配置权限

分类	功能权限	仓库设置	仓库管理	下载代码
代码仓库	<input checked="" type="checkbox"/> 访问代码	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
	<input checked="" type="checkbox"/> 拉取代码	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/> 在线编辑仓库文件	<input checked="" type="checkbox"/> 创建分支和编辑备注
	<input checked="" type="checkbox"/> 删除普通分支	<input type="checkbox"/> 删除保护分支	<input type="checkbox"/> 保护分支规则	<input checked="" type="checkbox"/> 创建标签
	<input checked="" type="checkbox"/> 删除普通标签	<input type="checkbox"/> 删除保护标签	<input checked="" type="checkbox"/> 创建版本	<input checked="" type="checkbox"/> 编辑版本
	<input checked="" type="checkbox"/> 删除版本	<input checked="" type="checkbox"/> 创建合并请求	<input checked="" type="checkbox"/> 部署公钥	<input checked="" type="checkbox"/> 解锁文件

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。

The screenshot shows the repository structure for 'testli/演示/demo'. On the left, there's a tree view of files: 'demo' (containing 'src', 'Dockerfile', 'Jenkinsfile', 'README.md', 'mvnw', 'mvnw.cmd', 'pom.xml'). In the center, a list of branches is shown: 'master' (selected), '分支 5' (highlighted with a red box), and '标签 2'. Below the branches, a commit history is displayed:

分支	提交者	提交信息	时间		
分支 5	reviewer-1	Initial commit	6 天前		
分支 5	reviewer-1	reviewer-1 提交于 6 天前	c30a6e4		
分支 5	reviewer-1	reviewer-1	Initial commit	6 天前	
分支 5	reviewer-1	Dockerfile	reviewer-1	Initial commit	6 天前
分支 5	reviewer-1	Jenkinsfile	reviewer-1	Initial commit	6 天前
分支 5	reviewer-1	README.md	reviewer-1	Initial commit	6 天前
分支 5	reviewer-1	mvnw	reviewer-1	Initial commit	6 天前
分支 5	reviewer-1	mvnw.cmd	reviewer-1	Initial commit	6 天前

2. 选择指定分支，单击 并选择设为只读。

活跃分支 | 我的分支 | 保护分支 | 只读分支 | 所有分支 | 过时分支

搜索 + 创建分支 ...

master 默认分支
Chasy 更新于 34 分钟前 -o- a23f75f

dev
Chasy 更新于 43 分钟前 -o- 0cec9f3

0 0 1 1

创建合并请求 对比 编辑备注 cherry-pick revert 下载分支

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。

活跃分支 | 我的分支 | 保护分支 | 只读分支 | 所有分支 | 过时分支

搜索 + 创建分支 ...

只读分支
master 默认分支
Chasy 更新于 36 分钟前 -o- a23f75f

dev
Chasy 更新于 1 小时前 -o- 0cec9f3

0 0 1 1

创建合并请求 ...

1 - 2 个, 共 2 个 < 1 > 15 条/页

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）。

The screenshot shows the 'Merge Request' tab selected in the settings menu. On the left sidebar, 'Merge Request' is also highlighted. The main area contains several configuration sections:

- 基础设置**:
 - 是否默认删除源分支**: A toggle switch is off. Description: After enabling, merge requests will default to deleting the source branch, but it's not mandatory.
 - 是否默认以 Fast-Forward 模式合并**: A toggle switch is off. Description: After enabling, merge requests will default to Fast-Forward merging, but it's not mandatory.
 - 合并请求源分支有新提交时自动取消合并授权**: A toggle switch is off. Description: After enabling, if the source branch has new commits, the merge request's merge permission will be automatically revoked.
 - 开启状态检查**: A toggle switch is off. Description: After enabling, all merge requests to this branch must pass all CI tasks before merging.
 - 合并前必须获得所有评审者的允许合并**: A toggle switch is off. Description: All reviewers must approve the merge request for it to be merged.
- 合并方式选择**: Radio buttons for merge strategies:
 - 默认直接合并
 - 默认 Squash 合并
 - 只能 Squash 合并
- 合并请求默认目标分支**: A dropdown menu set to 'master'.

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。

合并此请求

Accept Merge Request #12: (new-br -> master)

Merge Request: A new MR

Created By: @阿蓝

Accepted By: @阿蓝

URL: [https://\[REDACTED\]/p/lans-project/d/android/git/merge/12](https://[REDACTED]/p/lans-project/d/android/git/merge/12)

删除源分支 Fast-Forward 模式合并

合并分支

取消

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。

The screenshot shows a list of status checks for a merge request:

- 合并请求源分支与目标分支无冲突**: 自动合并可以工作
- 合并状态检查: 成功**: 1 个合并检查成功
Android-test — 构建成功
- 代码扫描: 质量门禁**: 未开启
0 个致命问题, 0 个错误问题, 3 个警告问题, 0 个提示问题
- 保护分支: 等待分支管理员授权**: 您作为分支管理员可以给予合并授权，或者直接合并MR

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。

The screenshot shows the merge request details for pull request #36. It includes sections for 'Mergeable' (可合并), 'Reviewers' (评审者), 'Tags' (标签), 'Followers' (关注者), and 'Start Review' (发起代码评审). A red box highlights the 'Reviewers' section.

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。

合并提交消息模版

如果合并时产生一个合并提交，提交消息将默认使用此模版

```
Accept Merge Request ${id} : (${source_branch} -> ${target_branch})  
Merge Request: ${title}
```

Created By: \${creator}
Reviewed By: \${reviewers}
Approved By: \${approvers}
Accepted By: \${merger}
URL: \${url}

0 / 400

压缩提交消息模版

如果合并时产生一个压缩提交，提交消息将默认使用此模版

```
Accept Merge Request ${id} : (${source_branch} -> ${target_branch})  
Merge Request: ${title}
```

Created By: \${creator}
Reviewed By: \${reviewers}
Approved By: \${approvers}
Accepted By: \${merger}
URL: \${url}

0 / 400

保存

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

<code>#{reviewer}</code>	合并请求评审人员	@开发组长
<code>#{approver}</code>	合并请求允许合并人员	@开发组长
<code>#{acceptor}</code>	合并请求合入分支人员	@开发组员
<code>#{creator}</code>	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

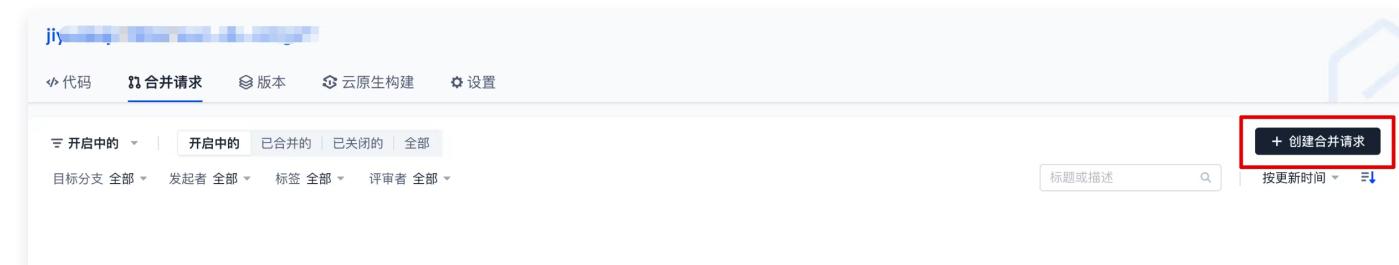
您可以通过命令行或在 Web 端手动创建合并请求。

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。

创建合并请求

请选择合并请求的源分支和目标分支，并查看提交记录和文件改动，合并之后目标分支将会包括源分支的改动内容

源分支: dev 合并到 目标分支: master 交换 △ 存在代码冲突 你仍然可以创建合并请求

概览 提交记录 1 文件改动 1

合并请求标题 *

请输入合并请求标题

描述

编辑 预览

请在此输入合并请求描述内容（支持 Markdown）

关联资源 0 + 添加资源

点击右上角“+”关联项目资源（迭代、任务、合并请求等）及添加外部链接

创建合并请求

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。

创建合并请求

请选择合并请求的源分支和目标分支，并查看提交记录和文件改动，合并之后目标分支将会包括源分支的改动内容

源分支: dev 合并到 目标分支: master 交换 △ 存在代码冲突 你仍然可以创建合并请求

概览 提交记录 1 文件改动 1

提交于 2023-04-14

更新文件 README.md
Chasy 提交于 1小时前 0cec9f3

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。

代码 合并请求 3 版本 云原生构建 设置

文件冲突 更新 readme

#54 陈星 期望将 master-patch-2 合并到 master

概览 2 提交记录 1 文件改动 1

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。

The screenshot shows a merge request for pull request #54, which merges 'master-patch-2' into 'master'. A conflict has occurred in the file 'README.md'. The conflict is highlighted with red boxes around the conflict markers and the '查看冲突内容' (View Conflict Content) button. The code editor shows the conflict markers and the three options for resolution: '使用当前的改动' (Use Current Changes), '使用传入的改动' (Use Incoming Changes), and '使用两者的改动' (Use Both Changes).

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。

The screenshot shows a code editor for the file 'README.md'. A conflict has occurred at line 47. The conflict markers are highlighted with red boxes, and the '使用当前的改动 | 使用传入的改动 | 使用两者的改动' (Use Current Changes | Use Incoming Changes | Use Both Changes) button is also highlighted. The code editor shows the conflict markers and the three options for resolution.

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。

The screenshot shows a merge request page with the following details:

- Repository:** test_protect
- Branches:** #1163 Ruby 期望将 test_protect 合并到 master
- Reviewers:** Ruby (发起人), 未选择 (审阅者)
- Labels:** 未选择 (标签)
- Watchers:** 未选择 (关注者)
- Code Review:** 发起代码评审 (发起)

The "Merge Type" dropdown is open, showing three options:

- 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。

合并此请求

Accept Merge Request #79: (new-feature -> master)

Merge Request: demo

Created By: @主账号

Accepted By: @主账号

URL: <https://straybirds.coding.net/p/coding-demo/d/coding-demo/git/merge/79>

删除源分支 Fast-Forward 模式合并

合并分支

取消

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。

The screenshot shows the CODING account settings page. On the left, there's a sidebar with options like Personal Account Settings, Account Information, Personal Account, Email Settings, etc. The 'GPG Keys' option is selected. A modal window titled 'Add GPG Key' is open, with a red box highlighting the 'Add GPG Key' button at the top right. The main area of the modal contains a text input field with placeholder text: 'Please paste the public key content starting with -----BEGIN PGP PUBLIC KEY BLOCK-----'. Below the input field is a sample key block. At the bottom of the modal are 'Confirm' and 'Cancel' buttons.

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。

The screenshot shows the GPG Keys management page. It displays a single key entry with a blue key icon. The details are: Email Address (redacted), Status: Verified (green box), Key ID (redacted), Subkey (redacted), and Added On: 2021-09-14. There is a 'Delete' button to the right of the key entry.

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。

The screenshot shows a pull request page. On the right side, there is a detailed view of a commit with a red box highlighting the GPG verification status. The status is '已验证' (Verified) with a green checkmark icon. Below it, there is a note: '当前提交已通过验证，且提交签名密钥的邮箱属于提交者。' (The current submission has passed verification, and the email address of the signing key belongs to the submitter.) To the right of this note is a 'GPG 公钥' (GPG Public Key) section, which includes the owner's name ('CODING 官方'), public key ID ('A'), and a link to learn how to sign submissions ('如何对提交进行签名'). At the bottom of the commit view, there are two buttons: '未验证' (Not Verified) and '4f4542a'.

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。

The screenshot shows the 'Personal Account Settings' page. On the left sidebar, under 'GPG Public Key' (highlighted in grey), there is a list of items: 'Account Information', 'Personal Account', 'Email Settings', 'Personal Settings', 'Template Settings', 'SSH Public Key', and 'GPG Public Key beta'. The 'GPG Public Key beta' item is currently selected. On the right, there is a 'GPG Public Key' section with a sub-section titled 'In addition to adding a GPG public key, you can add GPG signatures to your commits.' It shows a public key entry with fields: 'Email Address' (显示为已验证), 'Public Key ID' (显示为未验证), and 'Subkey' (显示为未验证). A red box highlights the 'Delete' button next to the public key entry. Below this, it says 'Added on 2021-09-14'.

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。

The screenshot shows a GitHub repository named 'node-express-example'. The 'master' branch is selected. A sidebar on the left lists various project management sections like '代码仓库', '持续集成', and '持续部署'. The main area displays a list of commits. One commit, '953a011', has a red box drawn around its status badge, which is labeled '未验证' (Unverified). Other commits in the list are labeled '已验证' (Verified) with green boxes. Each commit entry includes the commit hash, author, date, and a 'View' button.

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。

PGP 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。

The screenshot shows the 'Code Versions' management page. At the top, there are tabs for 'Code', 'Merge Requests (3)', 'Versions' (which is selected), 'Cloud Native Build', and 'Settings'. On the right, there is a link to 'New Repository Usage Guide'. Below the tabs are filters for 'Status', 'Assignee', 'Label', 'Create Date', and search fields for 'Label, Description or Tag Version' and a 'Create Version' button. The main area displays two code versions: '预先测试' (Pre-test) and '预发布内容' (Pre-release content). Each entry includes a user icon, name (Chen Xing), creation date (2 years ago), merge request ID (release-03 or release01), commit hash (8541b4ff), and a status indicator ('已发布' - Published for the first, '预发布' - Pre-release for the second). At the bottom, it shows '1 - 2个, 共2个' (1 - 2 pages, total 2 pages) and a page navigation bar.

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）。

创建代码版本发布

标签版本 *

创建或复用已有标签

创建来源 *

@  master ▾

版本发布标题

请输入版本发布标题

描述

编辑 预览

H B I ⌂ “ </ 困 ■ 三 三 □ @ # ✎ ⏺

请在此输入描述内容（支持 Markdown）

 点击或将文件拖拽至此上传！

支持所有文件格式，单个文件不超过100M。

关联资源 0 | + 添加资源

点击“+”关联项目资源（迭代、任务、合并请求等）及添加外部链接

这是一个预发布
我们会为此 Release 显示预发布标识。

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。

The screenshot shows the Tencent Cloud CODING DevOps interface. On the left is a dark sidebar with navigation links: '代码仓库' (Code Repository), '仓库设置' (Repository Settings) which is currently selected, '团队部署公钥' (Team Deployment Public Key), and '仓库规范' (Repository Standards). The main content area has a light background. At the top, it says '代码仓库 / 仓库设置' (Code Repository / Repository Settings) and '仓库设置' (Repository Settings) with the sub-instruction '统一设置您的团队中所有仓库的部分设置' (Uniformly set part of the repository settings for all repositories in your team). Below this is a '推送设置' (Push Settings) section with two checkboxes: one for requiring committer and author email verification, and another for limiting file size to 0 MB (Git LFS files excluded). A '保存' (Save) button is present. The next section is '版本创建页摘要' (Version Creation Page Summary) with the instruction '自定义版本创建页面右侧摘要内容' (Customize the summary content on the right side of the version creation page). It contains a text input field with placeholder '版本格式：请遵循相应的版本规范。' (Version format: Please follow the corresponding versioning standard.) and a character count indicator '17 / 400'. Another '保存' (Save) button is shown. The final section is '标签创建页摘要' (Tag Creation Page Summary) with the same customization instructions and a text input field for '标签版本格式' (Label Version Format). A character count indicator '6 / 400' is also present here, along with a '保存' (Save) button.

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。

代码 合并请求 3 版本 云原生构建 设置

#20 预先测试

已发布 发布于 2021-08-16 | release-03 · 8541b4ff

描述 | 编辑

编辑 预览

请在此输入版本发布描述内容（支持 Markdown）

保存 取消

引用 0

点击“+”关联项目资源（迭代、任务、合并请求等）及添加外部链接

下载

源代码 (zip) 源代码 (tar.gz)

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。

The screenshot shows a detailed view of a version release page. At the top, there's a header with a cloud icon, the text '#20', and a '删除版本' (Delete Version) button. Below the header, there's a status bar with '已发布' (Published), a user profile, and release details: '发布于 2021-08-16', 'release-03', and '8541b4ff'. To the right of the status bar is a red-bordered '回' (Return) button. The main area has tabs for '描述' (Description) and '编辑' (Edit). A rich text editor is present with a toolbar above it containing icons for H, B, I, etc. Below the editor is a placeholder text '请在此输入版本发布描述内容 (支持 Markdown)'. At the bottom left are '保存' (Save) and '取消' (Cancel) buttons. On the right side, there are sections for '引用 0' (Cited 0) and '+ 添加资源' (Add Resource), with a note to '点击“+”关联项目资源 (迭代、任务、合并请求等) 及添加外部链接'. Below these are download and upload sections. Under '下载' (Download), there are two options: '源代码 (zip)' and '源代码 (tar.gz)'. Under '上传文件' (Upload File), there is a '+ 上传文件' button.

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

版本发布默认分支

新建版本发布时，目标分支会默认选择此分支

保存

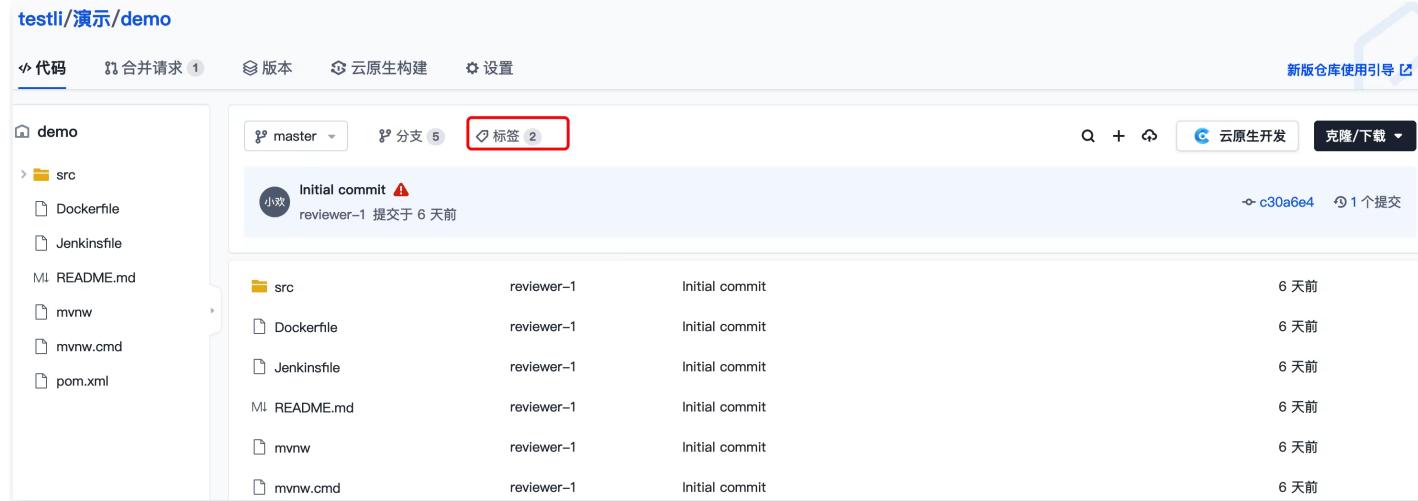
代码标签 仓库安全 推送权限

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。

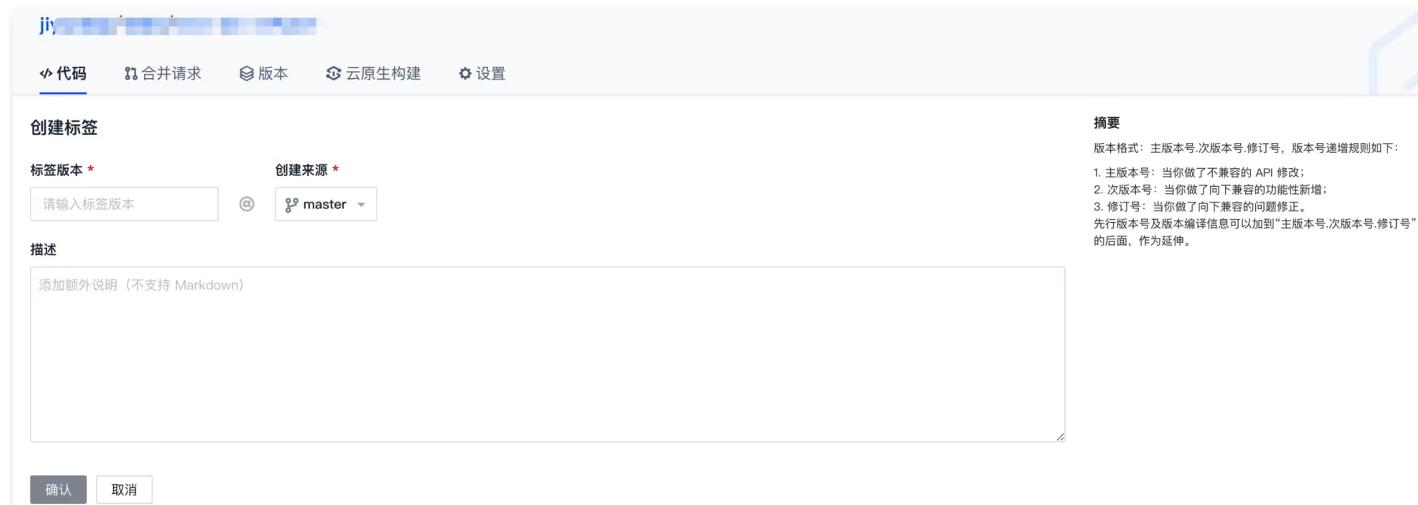


The screenshot shows the repository details page for 'demo'. On the left, there's a sidebar with file navigation. In the center, there are tabs for '代码' (Code), '合并请求' (Pull Requests), '版本' (Versions), '云原生构建' (Cloud Native Build), and '设置' (Settings). The '版本' tab is selected. Below it, there are dropdown menus for '分支' (Branch) set to 'master' and '标签' (Tags) set to '2'. A red box highlights the '标签' button. To the right, a summary card displays 'Initial commit' by 'reviewer-1' submitted 6 days ago. Below this, a table lists 8 commits, all from 'reviewer-1' and labeled 'Initial commit', each 6 days ago. The commits are: src, Dockerfile, Jenkinsfile, README.md, mvnw, mvnw.cmd, pom.xml.

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。



The screenshot shows the 'Create Tag' dialog box. It has tabs for '代码' (Code), '合并请求' (Pull Requests), '版本' (Versions), '云原生构建' (Cloud Native Build), and '设置' (Settings). The '版本' tab is selected. The main area has sections for '创建标签' (Create Tag), '标签版本 *' (Tag Version *), '请输入标签版本' (Enter tag version), '创建来源 *' (Create Source *), and a dropdown for 'master'. To the right, there's a '摘要' (Summary) section with instructions about tag versioning rules:

版本格式：主版本号.次版本号.修订号，版本号递增规则如下：
1. 主版本号：当你做了不兼容的 API 修改；
2. 次版本号：当你做了向下兼容的功能性新增；
3. 修订号：当你做了向下兼容的问题修正。
先行版本号及版本编译信息可以加到“主版本号.次版本号.修订号”的后面，作为延伸。

At the bottom, there are '确认' (Confirm) and '取消' (Cancel) buttons.

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。

团队设置中心 / 功能设置 / 仓库设置

仓库设置

团队部署公钥

仓库规范

代码仓库 / 仓库设置

仓库设置

统一设置您的团队中所有仓库的部分设置

推送设置

开启后将会强制覆盖团队中所有仓库的相同设置点

 检查 Git 提交的提交者 (Committer) 和提交作者 (Author) 必须是已验证的邮箱。 单次提交的文件总大小不能超过 MB, Git LFS 文件除外。

保存

版本创建页摘要

自定义版本创建页面右侧摘要内容

版本格式：主版本号.次版本号.修订号，版本号递增规则如下：

1. 主版本号：当你做了不兼容的 API 修改；
2. 次版本号：当你做了向下兼容的功能性新增；
3. 修订号：当你做了向下兼容的问题修正。

先行版本号及版本编译信息可以加到“主版本号.次版本号.修订号”的后面，作为延伸。

0 / 400

保存

标签创建页摘要

自定义标签创建页面右侧摘要内容

版本格式：主版本号.次版本号.修订号，版本号递增规则如下：

1. 主版本号：当你做了不兼容的 API 修改；
2. 次版本号：当你做了向下兼容的功能性新增；
3. 修订号：当你做了向下兼容的问题修正。

先行版本号及版本编译信息可以加到“主版本号.次版本号.修订号”的后面，作为延伸。

0 / 400

保存

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。

代码 合并请求 3 版本 云原生构建 设置 新版仓库使用引导

仓库设置

代码标签

允许删除标签

保护标签规则

+ 添加标签规则

操作

暂无数据

基本设置 权限方案 部署公钥 文件锁定 分支设置 库存规范 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

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。

代码 合并请求 版本 云原生构建 设置

共 4 条标签

标签名	操作
v4.0	... 创建于 2 分钟前 ~ 0cec9f3
v3.0	... 创建于 1 小时前 ~ 0cec9f3
v2.0	... 创建于 2 小时前 ~ d2d652c
v1.0	... 创建于 2 小时前 ~ d2d652c

创建版本描述 下载 对比 删除

版本描述

版本描述

版本描述

1 - 4 个, 共 4 个 < 1 > 15 条/页

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。

The screenshot shows the 'Code Labels' section of the repository settings. On the left sidebar, 'Code Labels' is selected. In the main area, there's a checkbox for 'Allow deleting labels' which is checked. Below it, a note says: '未勾选时，任何人都无法删除标签，包括无法通过 git push -f 来强制删除标签。'. A button '+ Add label rule' is visible. A table titled '保护标签规则' (Label Protection Rules) shows '暂无数据' (No data). Other sections like 'Basic Settings', 'Permissions', and 'Branches' are also listed in the sidebar.

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/yyyy-release).
remote: 你没有权限推送到保护标签 (refs/tags/yyyy-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/yyyy-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。

The dialog box has two fields: 'Label Version *' containing 'prod-1' and 'Create Source *' showing 'master'. Below the fields, a red message box displays the error: '没有权限操作保护标签' (No permission to operate protected label).

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。

ji... 项目名

代码 合并请求 版本 云原生构建 设置

共 4 条标签

版本	操作
v4.0	... 创建于 2 分钟前 -> 0cec9f3
v3.0	... 创建于 1 小时前 -> 0cec9f3
v2.0	... 创建于 2 小时前 -> d2d652c
v1.0	... 创建于 2 小时前 -> d2d652c

1 - 4 个, 共 4 个

版本描述 下载 ...

版本描述

版本描述

版本描述

版本描述

1 / 15 条/页

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。

ji... 项目名

代码 合并请求 版本 云原生构建 设置

共 4 条标签

版本	操作
v4.0	... Chasy 创建于 2 分钟前 -> 0cec9f3
v3.0	... Chasy 创建于 1 小时前 -> 0cec9f3
v2.0	... Chasy 创建于 2 小时前 -> d2d652c
v1.0	... Chasy 创建于 2 小时前 -> d2d652c

1 - 4 个, 共 4 个

创建版本描述 下载 ...

版本描述

版本描述

版本描述

版本描述

1 / 15 条/页