代码托管

快速入门

产品文档

腾讯云

快速入门

最近更新时间：2023-12-25 17:08:18

操作场景

CODING 代码托管基于开源的版本控制系统 Git。借助 Git，您的本地电脑与 CODING 会各有一个完整的代码仓库，两者可以进行分布式的版本管理。为了在您的本地电脑上管理代码，您需要先配置好 Git，包括下载安装、设置等操作。如果您不需要在本地电脑上管理代码，CODING 支持在浏览器中实现大部分的代码仓库操作。本文档主要介绍如何快速使用 CODING 代码托管。

前提条件

使用 CODING 代码托管的前提是，您的腾讯云账号需要开通 CODING DevOps 服务，并按照页面弹框的提示前往**个人设置**完善服务密码、邮箱、手机，您在后续操作（克隆代码、推送代码等）会需要使用到上述信息。

本地环境初始化

安装 Git

Windows 版本

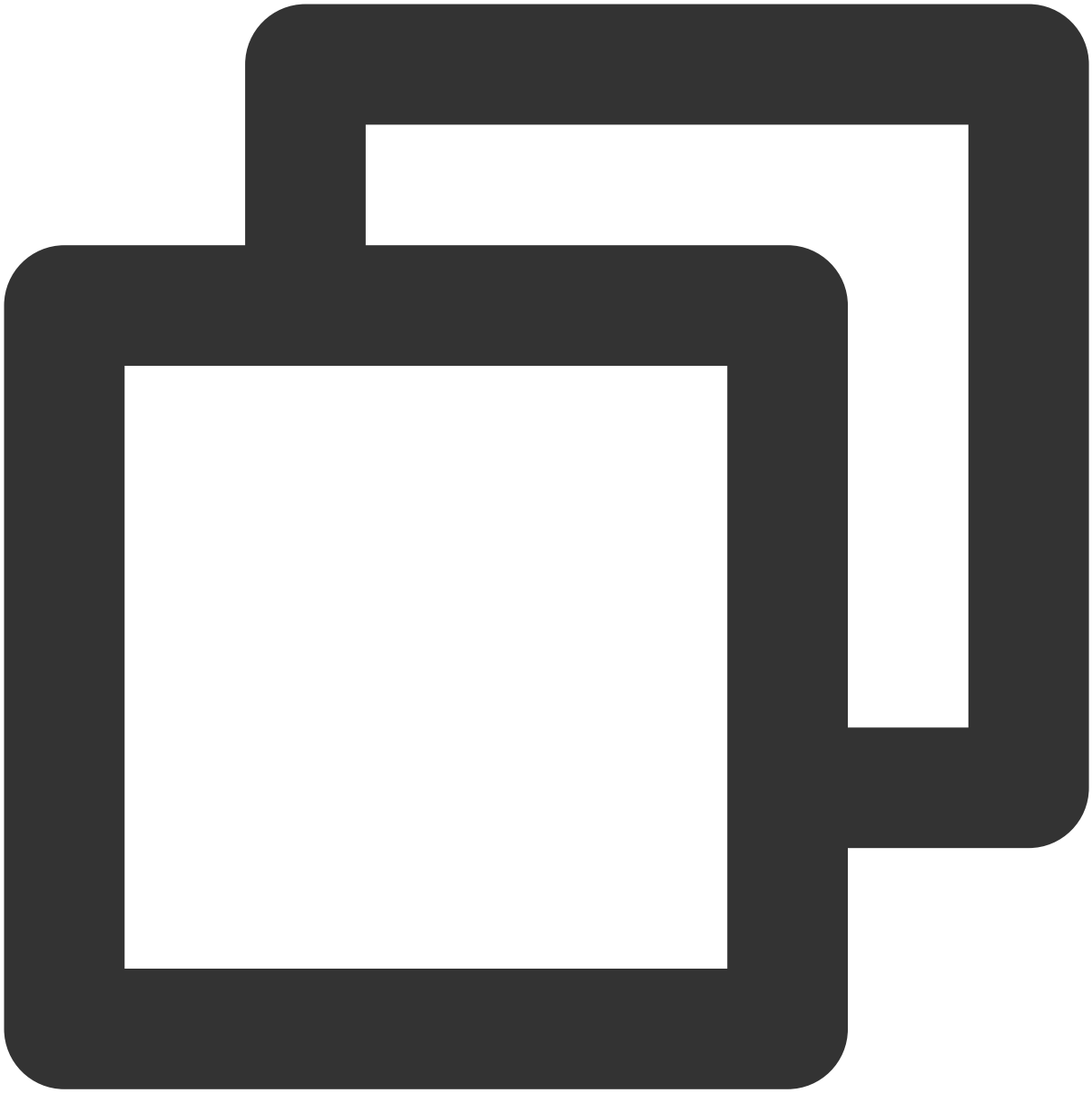
1. 从 Git 官网 下载 Git 客户端并按提示完成安装，推荐使用默认选项完成安装。
2. 完成安装后即可使用鼠标右键调出 Git Bash 终端，开始您的 Git 之旅。

Linux 版本

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 版本

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/
```

说明：

若出现错误提示，请参见 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。

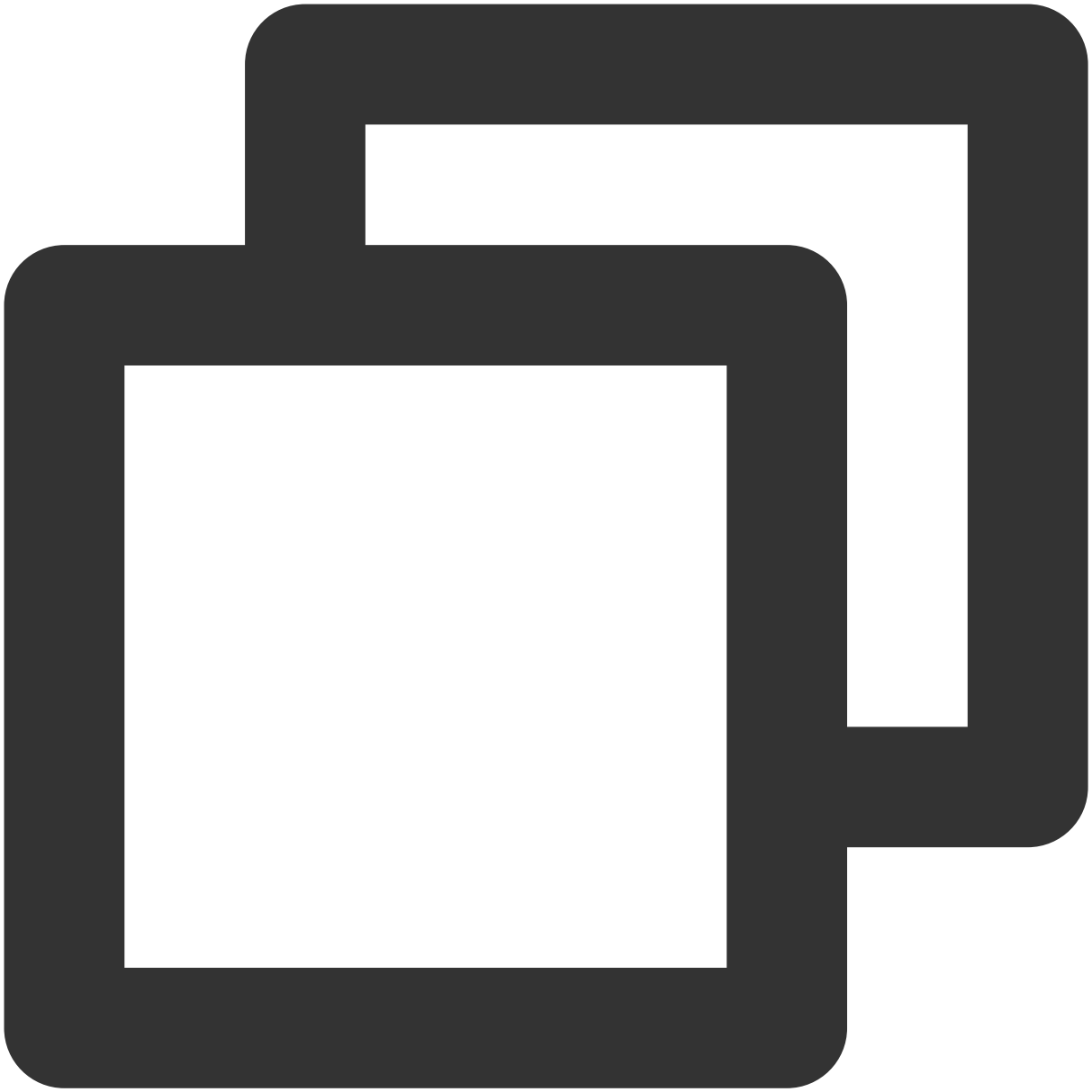
3. 安装 Git 之后，可运行 `git --version` 查看当前 Git 版本。

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。

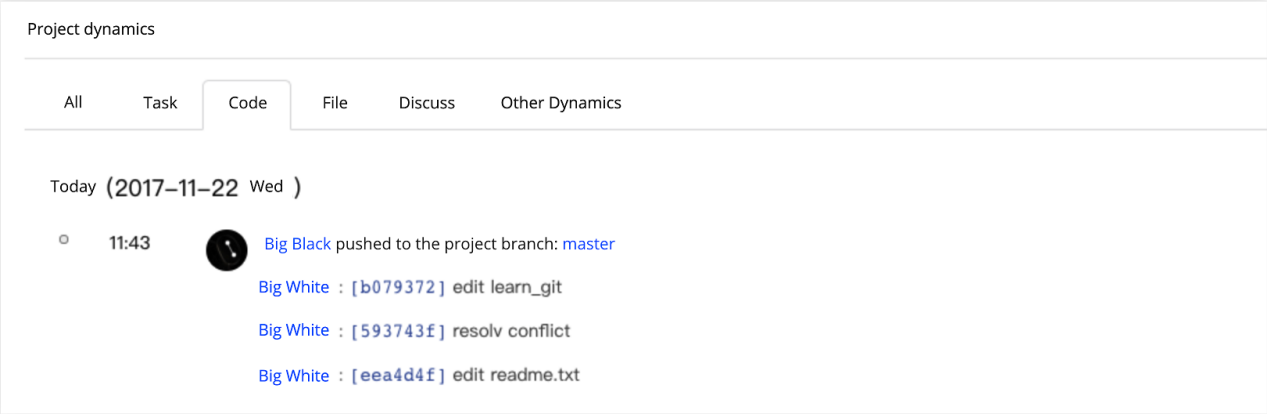
设置用户信息

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。使用以下命令设置用户信息。



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

例如您的 CODING 账户昵称叫**大黑**，Git 中的用户信息配置为：**名称 – 大白**，**邮箱 –** dabai@coding.net，当您推送代码到 CODING 仓库后，动态显示如下图：

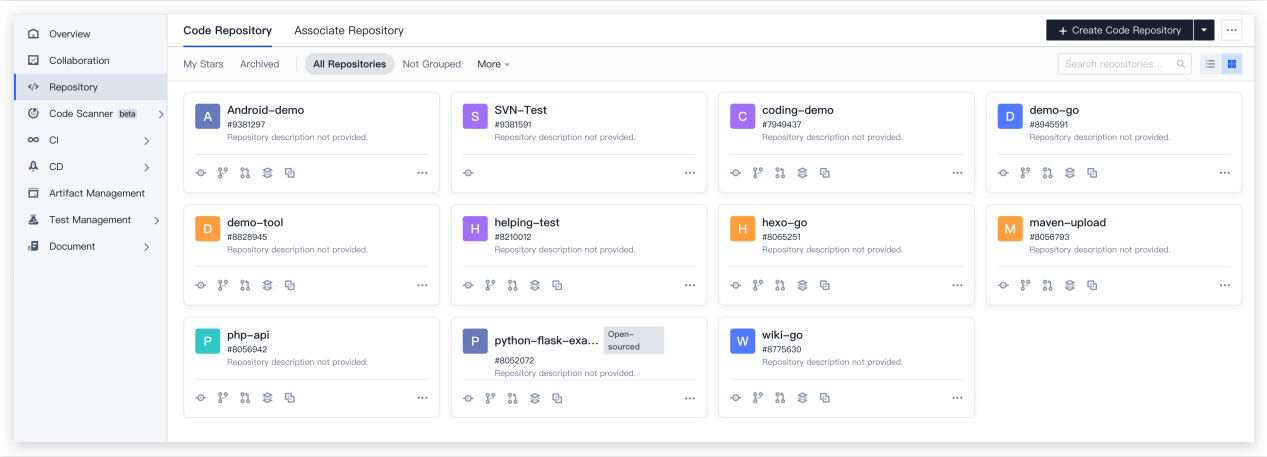


说明：

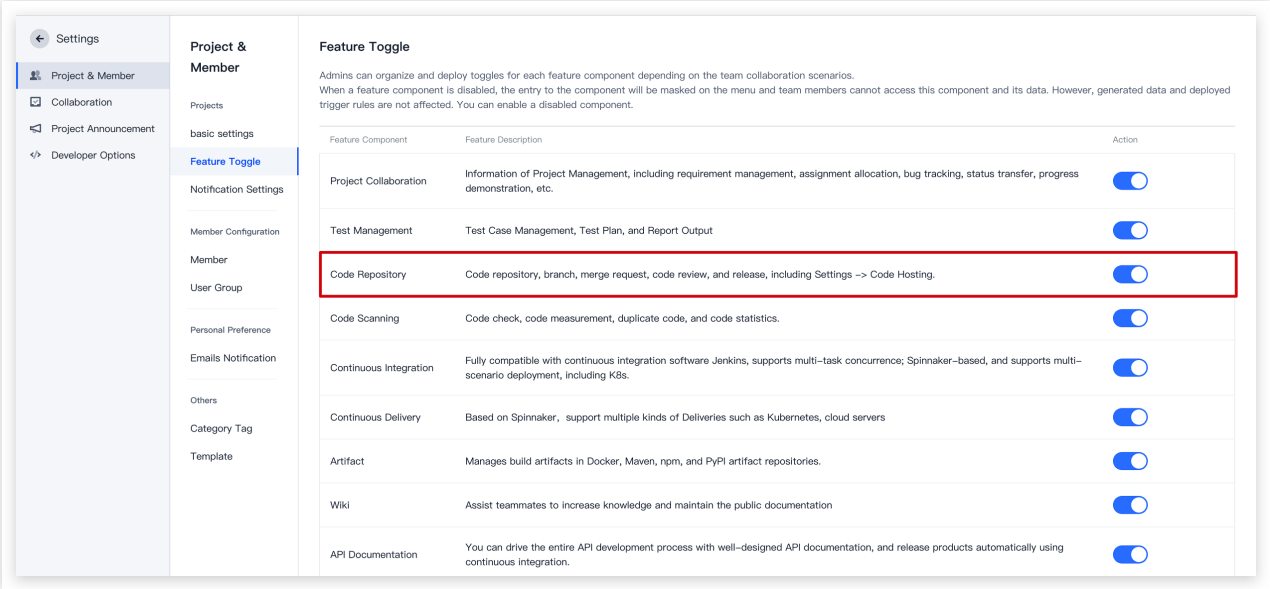
在 Git 中配置的用户信息是可自定义的，建议您直接填写为 CODING 用户名与注册邮箱，以便更好协作。

创建 CODING 代码仓库

1. 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。



2. 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 > 功能开关**中打开代码仓库功能。

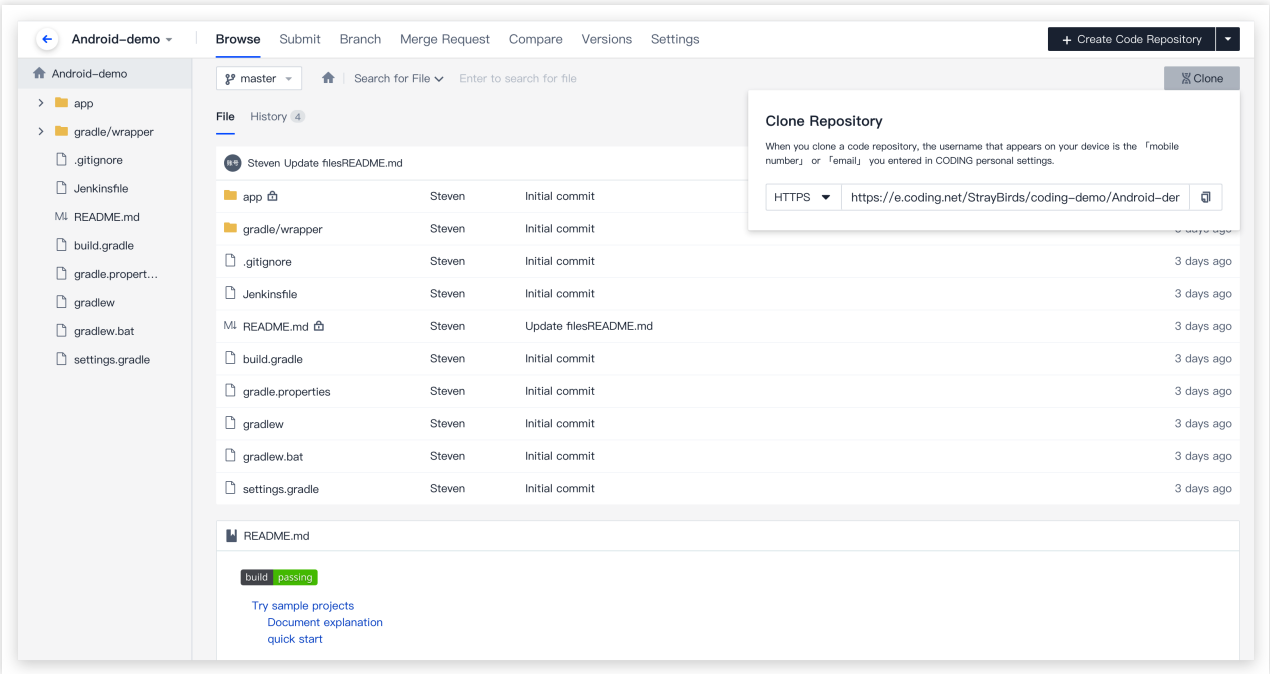


推拉代码

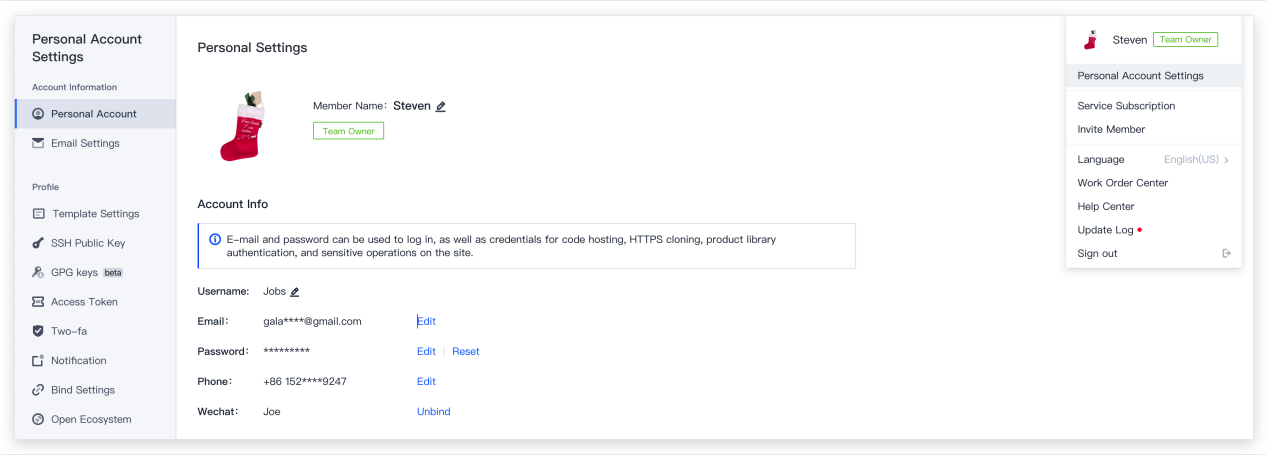
在此操作中将会演示如何从远端仓库拉取代码 / 上传本地代码至远端仓库，助力您的代码上云之旅。

拉取远程仓库代码

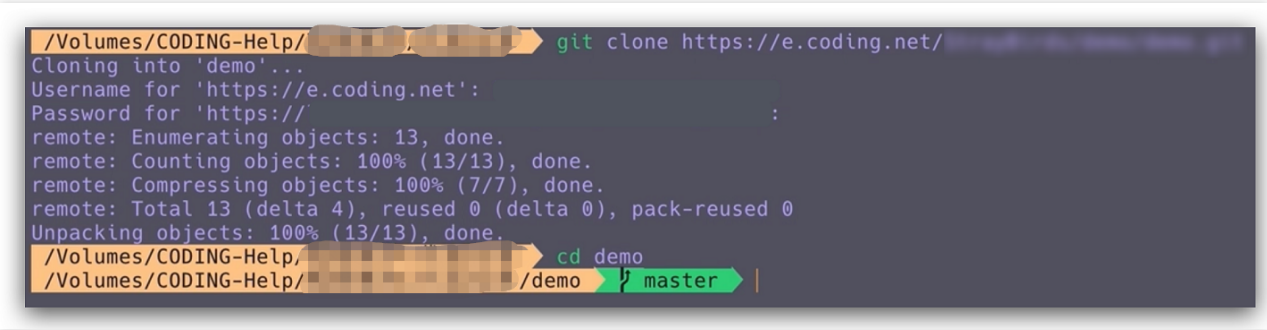
在本地完成代码仓库的初始化后，在文件夹中调出终端输入 `git clone + 仓库地址` 命令拉取代码。



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，您也可以单击右上角下拉菜单，前往**个人账户设置**中修改凭证信息。



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。

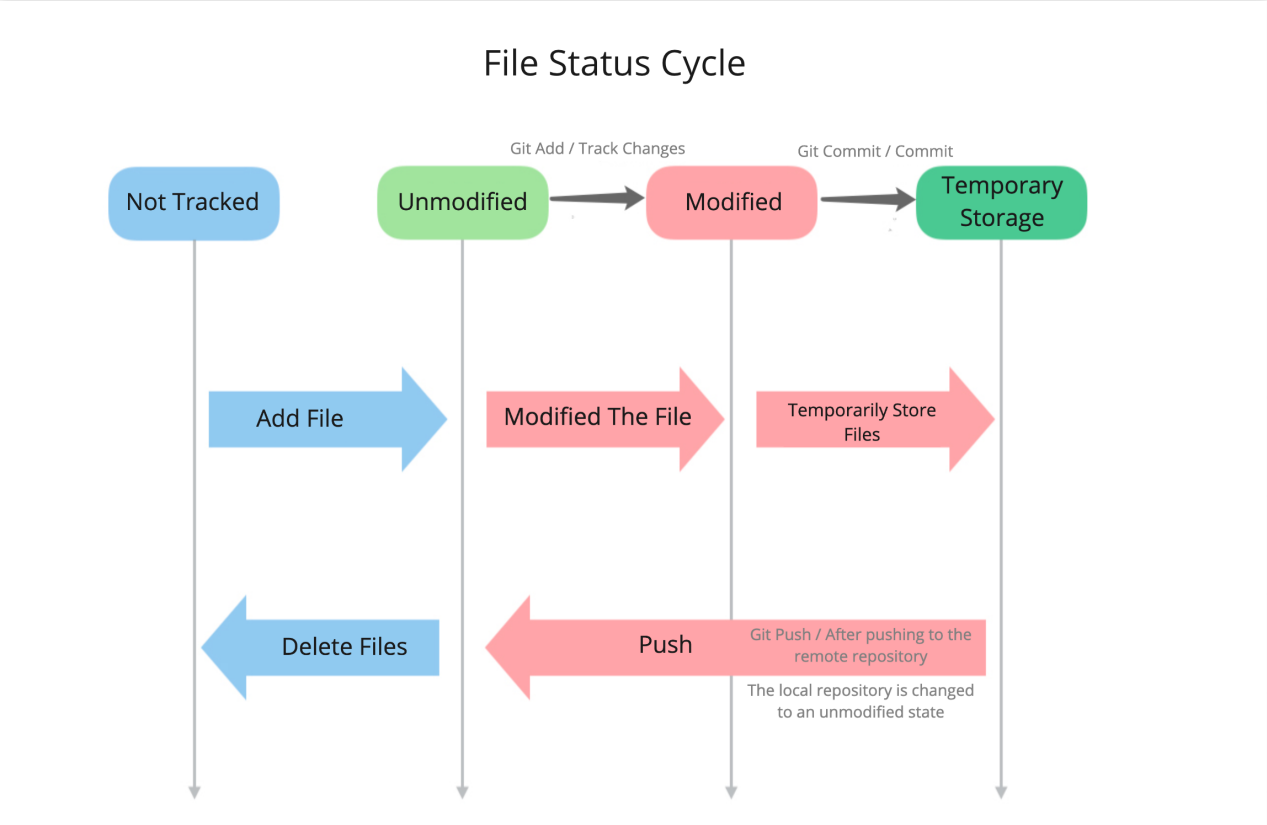


编辑文件

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.` （当然您也可以自由发挥）这句话并保存。

流程示意图

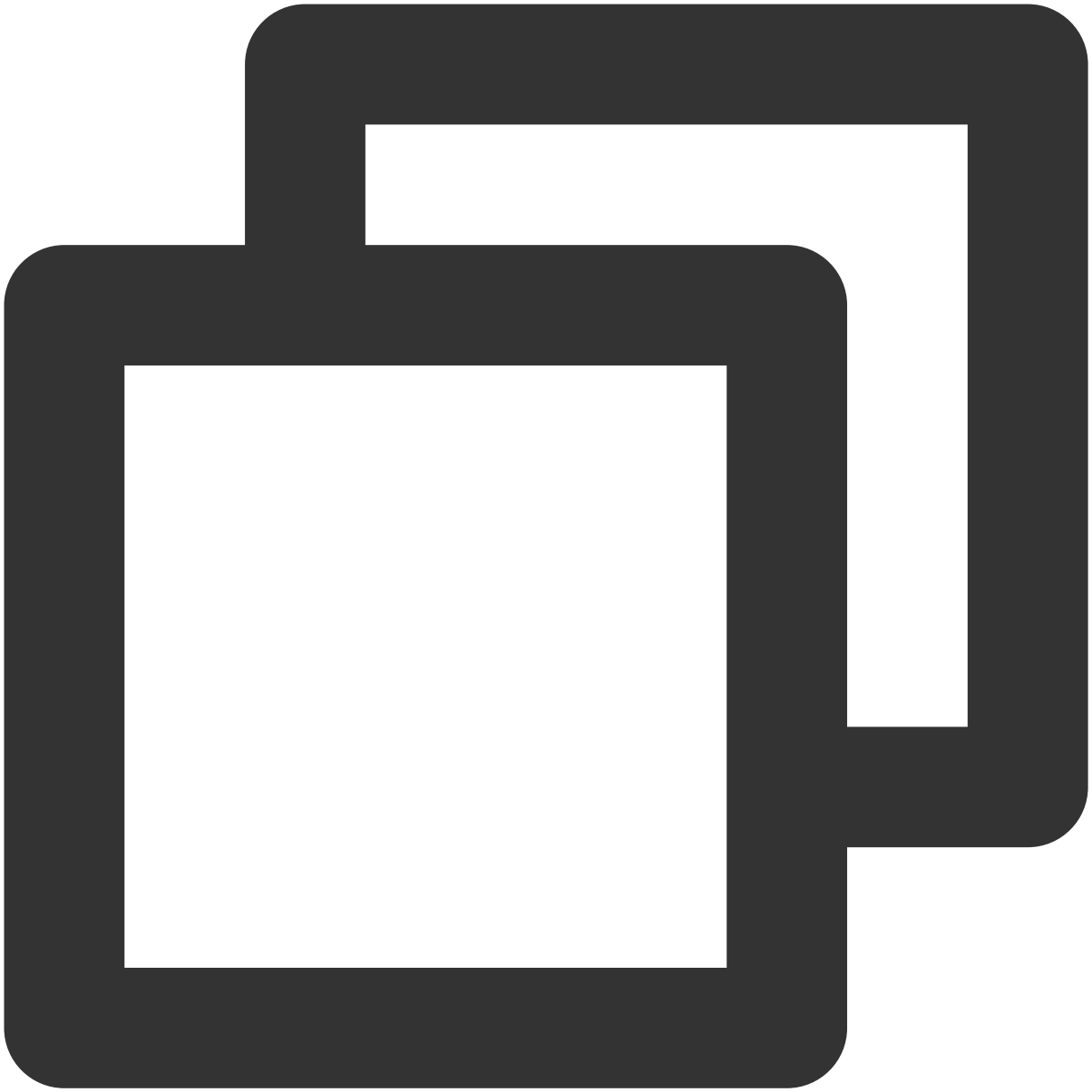
在正式进行代码提交前，您可以参见此流程示意图了解 Git 在追踪文件时所需经历的状态周期。



跟踪文件 (git add)

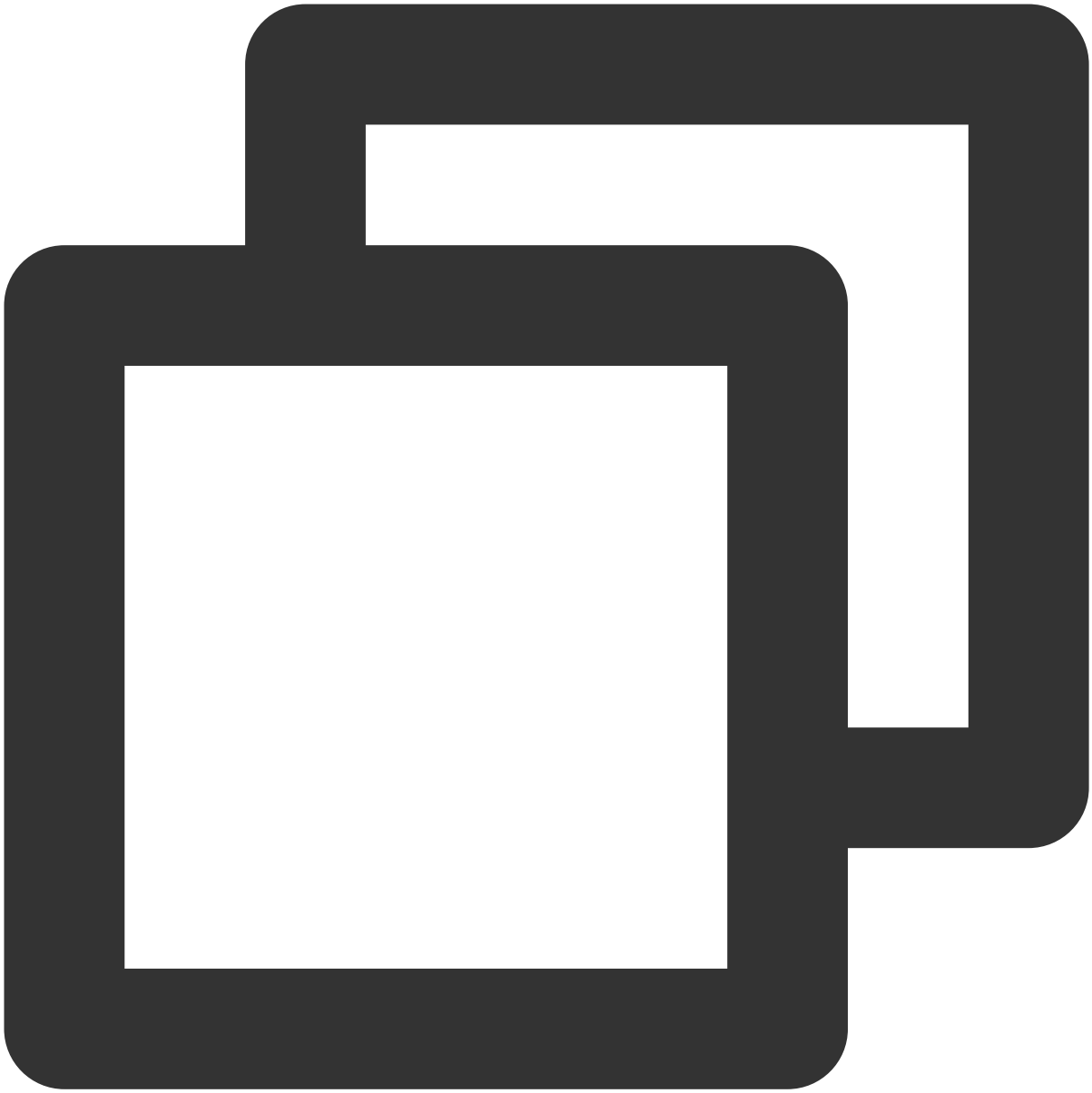
创建或修改文件后调用 `git add` 命令，将变更的文件添加至本地 Git 仓库的暂存区（Index Stage）。

追踪特定文件时的命令：



```
git add readme.txt
```

添加多个文件的命令：

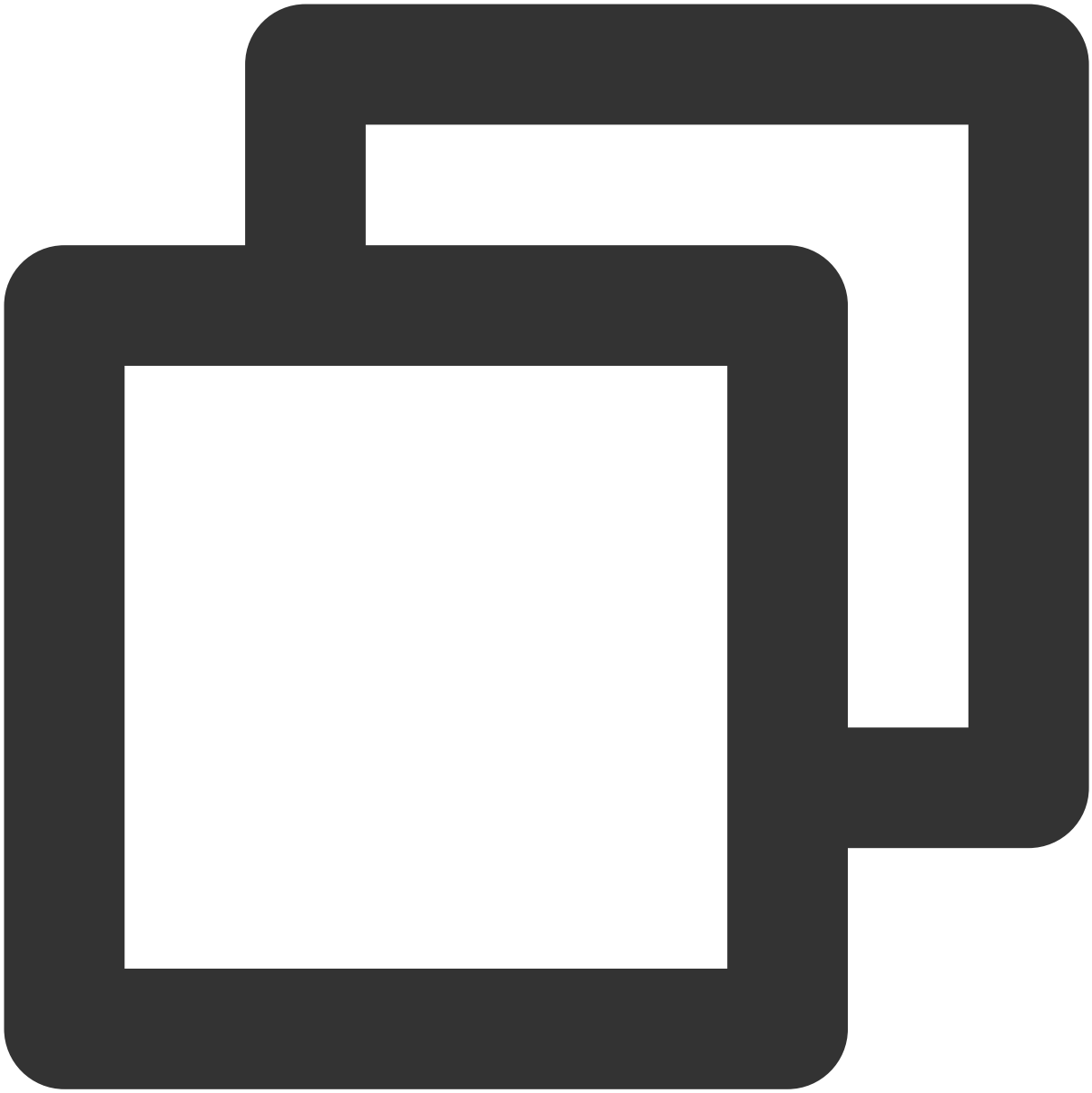


```
git add readme.txt learn_git.txt
```

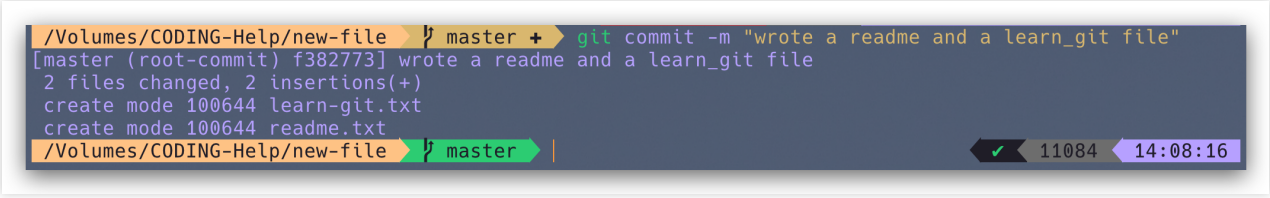
如果您想一次性跟踪**所有文件**，则可以直接在终端输入 `git add` 。

提交文件 (git commit)

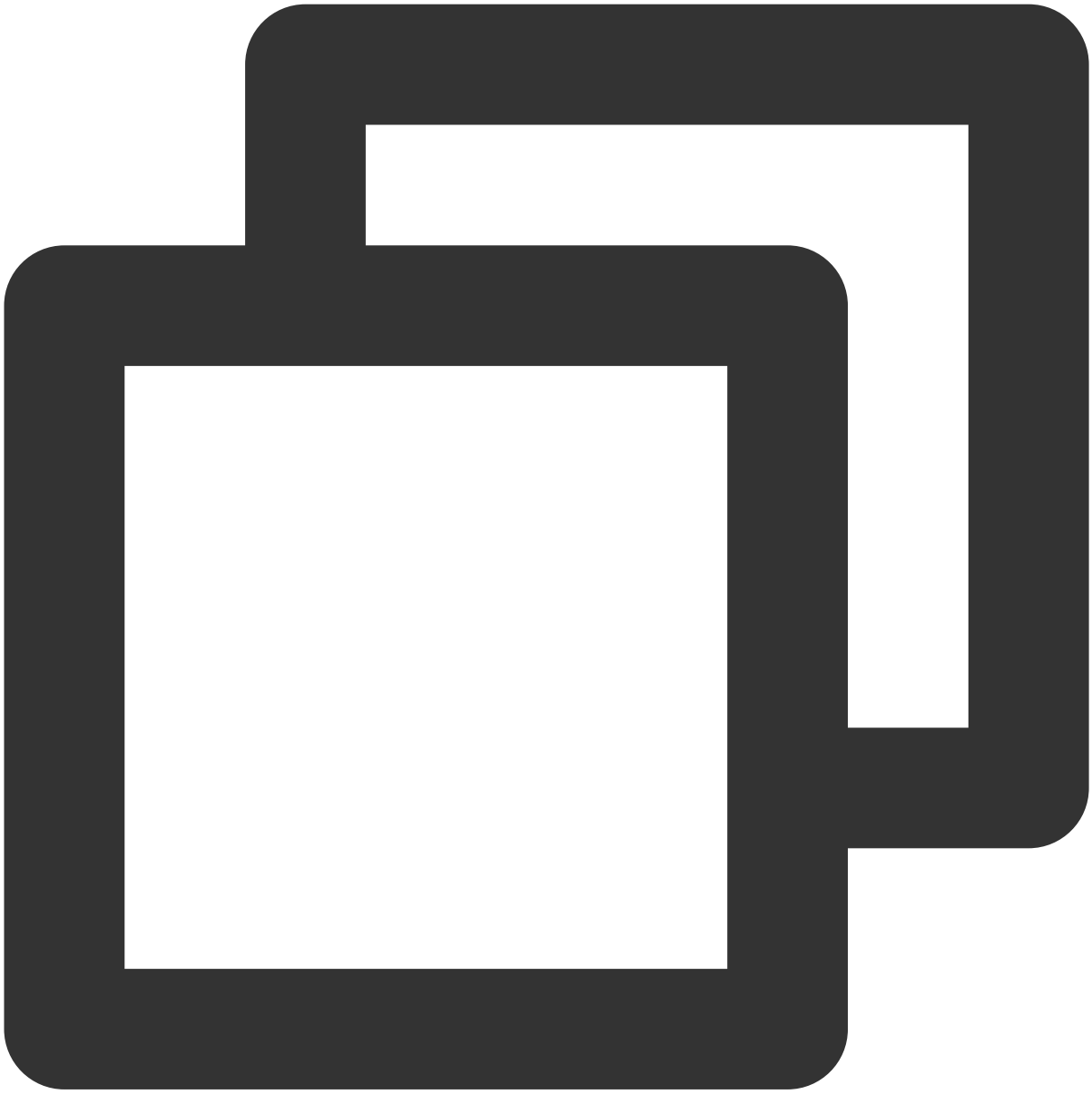
将拟提交的文件纳入暂存区后，运行 `git commit` 命令即可将文件正式提交至本地仓库，此命令将会一次性会提交暂存区中的所有文件：



```
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。养成每次提交文件时附上变更说明的习惯，以便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。

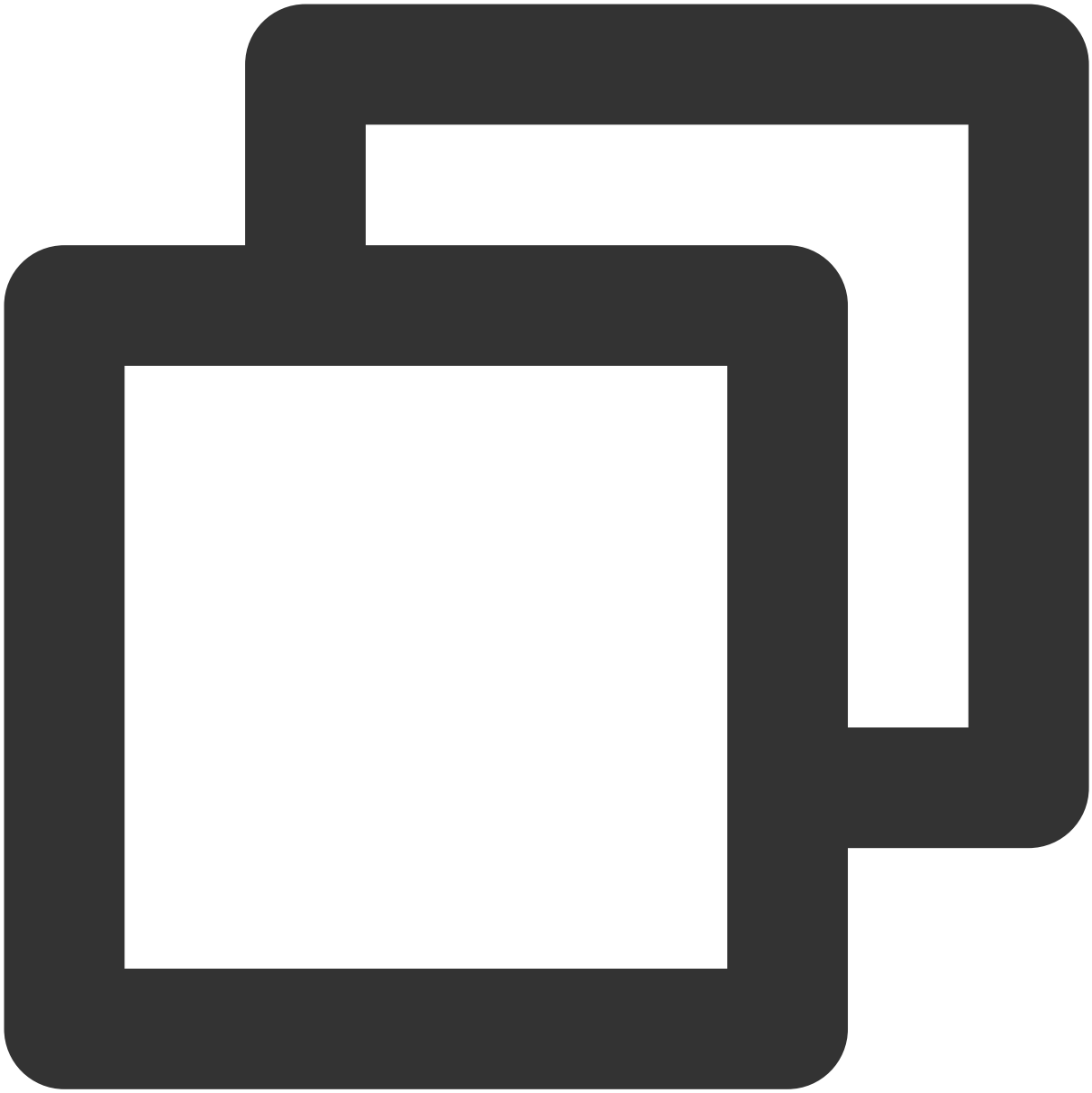
若该代码仓库开启了 研发规范 功能并在 分支规范 中指定了提交规范（即检验关联事项），确保在提交信息中关联了对应类型的事项。否则该分支将被检测为违规分支。

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。

查看文件状态 (git status)

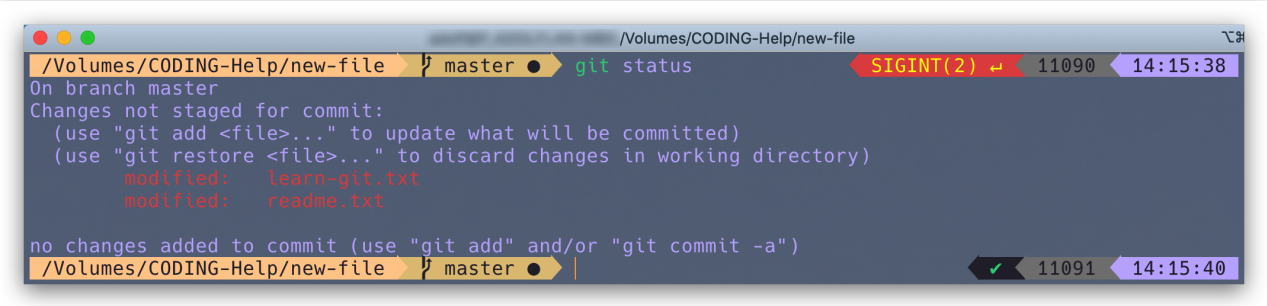
不确定 Git 是否精准的追踪了修改过的文件？想再次确认文件处于哪种状态？使用 `git status` 命令查看文件状态。

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：

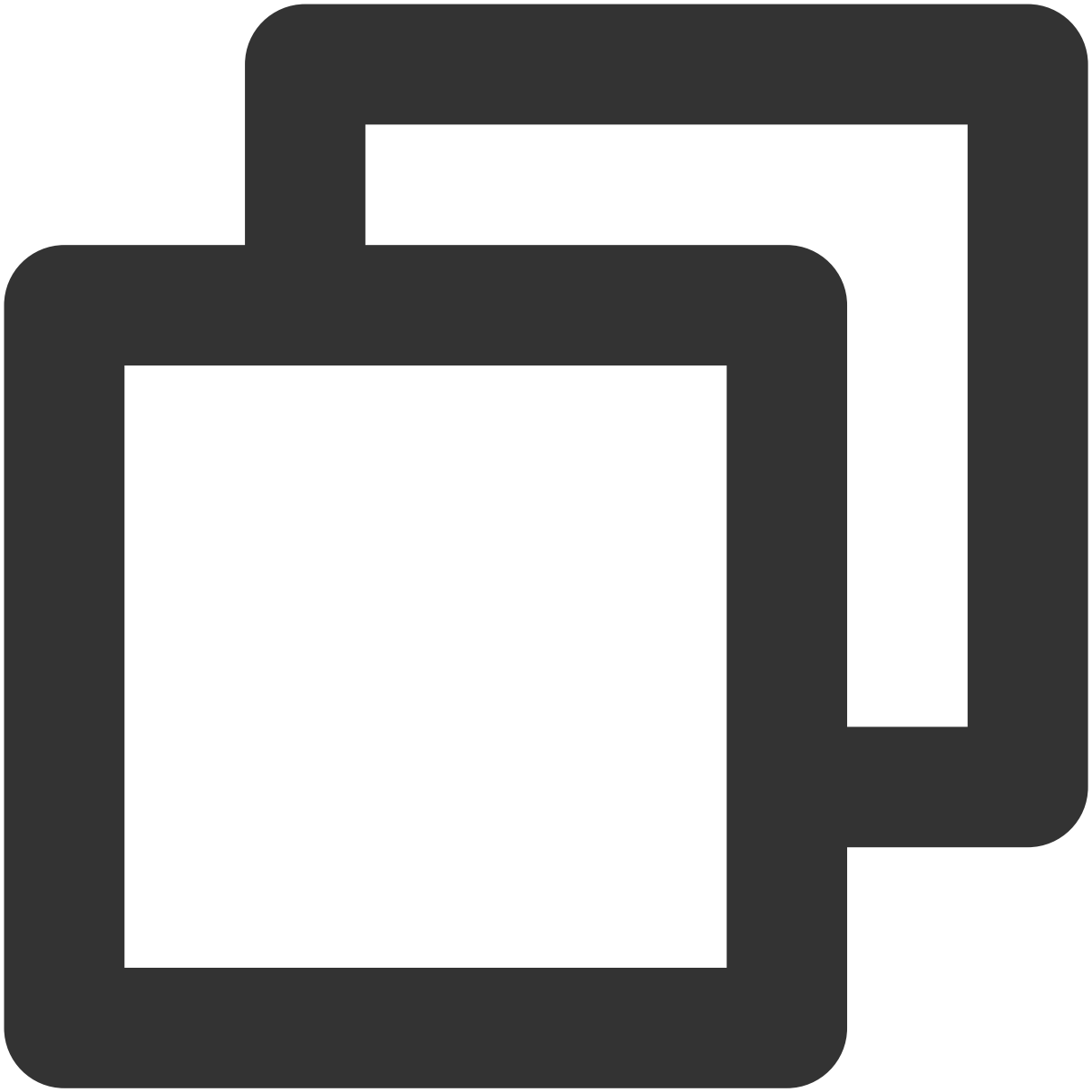


此时运行 `git add` 命令即可跟踪文件，跟踪完成后字体颜色由红色变为绿色。

当文件已跟踪且已经提交到仓库时,返回结果如下：

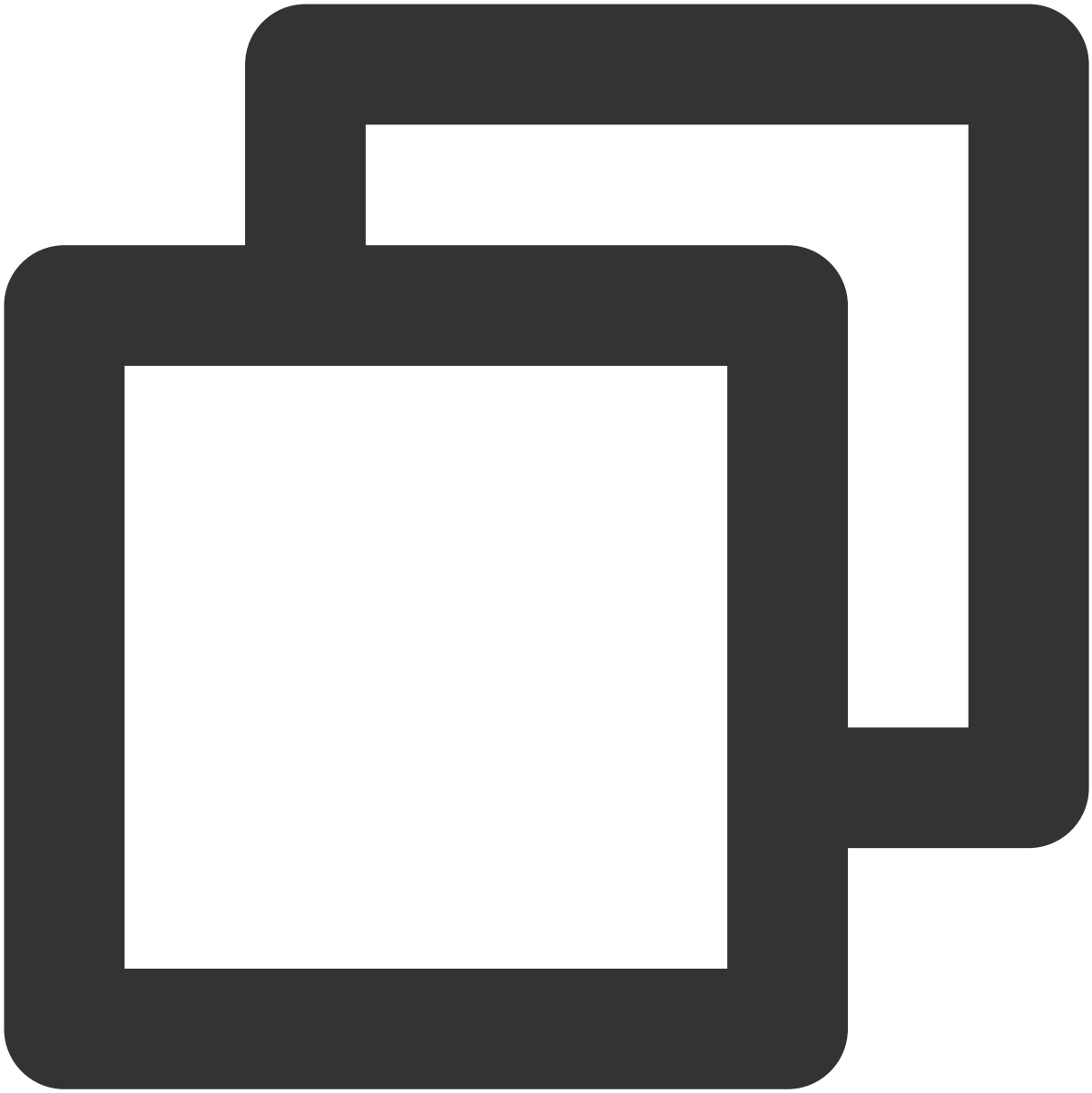
推送文件到远程仓库 (git push)

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，相当于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。

查看和编辑远程仓库

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”。