

Git基础

创建版本库

代码提交

时光穿梭

版本回退

撤销修改

删除文件

添加远程库

从远程库克隆

解决冲突

分支

普通分支

Bug分支

Feature分支

多人协作

创建标签

操作标签

忽略文件

Git使用

配置SSH公钥验证(免密码登录)

了解更多

本文记录了在Git学习中纪录的一些要点， 主要涉及Git日常使用的一些基础操作和简单使用。

首先，我们需要熟悉Git的几个操作区域：


Workspace: 工作区

Index / Stage: 暂存区

Repository: 本地仓库

Remote: 远程仓库

Git命令在各区域间的体现

 Git操作在各区域间的体现

Git基础

创建版本库

- 初始化一个Git仓库，使用 `git init` 命令。

- 添加文件到Git仓库，分两步：
 - a. 第一步，使用命令 `git add <file>`，注意，可反复多次使用，添加多个文件,若需添加所有，则可使用 `git add .` 或 `git add -A`；
 - b. 第二步，使用命令 `git commit -m '<message>'`，完成。

代码提交

- 提交暂存区文件到仓库区 `git commit -m 'message'`
- 提交暂存区的指定文件到仓库区 `git commit <file1> <file2> ... -m '<message>'`
- 提交工作区自上次commit之后的变化到仓库区 `git commit -a`
- 提交时显示diff信息 `git commit -v`
- 使用一次新的commit，替代上一次commit（如果无任何改变则改写上一次提交信息） `git commit --amend -m '<message>'`

时光穿梭

- 要随时掌握工作区的状态，使用 `git status` 命令。
- 如果 `git status` 告诉你有文件被修改过，用 `git diff` 可以查看修改内容。

版本回退

- HEAD 指向的版本就是当前版本，因此，Git允许我们在版本的历史之间穿梭，使用命令 `git reset -hard commit_id`。
- 穿梭前，用 `git log` 可以查看提交历史，以便确定要回退到哪个版本。
- 要重返未来，用 `git reflog` 查看命令历史，以便确定要回到未来的哪个版本。

撤销修改

- 场景1：当你改乱了工作区某个文件的内容，想直接丢弃工作区的修改时，用命令 `git checkout --file`。
- 场景2：当你不但改乱了工作区某个文件的内容，还添加到了暂存区时，想丢弃修改，分两步，第一步用命令 `git reset HEAD file`，就回到了场景1，第二步按场景1操作。
- 场景3：已经提交了不合适的修改到版本库时，想要撤销本次提交，参考版本回退一节，不过前提是没有推送到远程库。

删除文件

- 命令 `git rm` 用于删除一个文件。如果一个文件已经被提交到版本库，那么你永远不用担心误删，但是要小心，你只能恢复文件到最新版本，你会丢失最近一次提交后你修改的内容。

添加远程库

- 要关联一个远程库，使用命令 `git remote add origin git@server-name:path/repo-name.git` ；
- 关联后，使用命令 `git push -u origin master` 第一次推送master分支的所有内容；
- 此后，每次本地提交后，只要有必要，就可以使用命令 `git push origin master` 推送最新修改。

从远程库克隆

- 要克隆一个仓库，首先必须知道仓库的地址，然后使用`git clone`命令克隆。
- Git支持多种协议，包括https，但通过ssh支持的原生git协议速度最快。

解决冲突

- 当Git无法自动合并分支时，就必须首先解决冲突。解决冲突后，再提交，合并完成。
- 用 `git log --graph` 命令可以看到分支合并图。

分支

普通分支

- 查看分支： `git branch`
- 查看所有远程分支： `git branch -r`
- 列出所有本地分支和远程分支： `git branch -a`
- 创建分支，但仍然停留在当前分支： `git branch <branch name>`
- 创建分支并切换到新分支： `git checkout -b <branch>`
- 创建分支，与指定的远程分支建立追踪关系： `git branch --track <branch name> <remote-branch>`
- 切换分支，并更新工作区： `git checkout <branch name>`
- 切换到上一个分支： `git checkout -`
- 合并某分支到当前分支： `git merge <branch name>`
- 删除分支： `git branch -d <name>`
- 查看历史日志： `git log --graph --pretty=oneline --abbrev-commit`

Bug分支

- 修复bug时，我们会通过创建新的bug分支进行修复，然后合并，最后删除；
- 当手头工作没有完成时，先把工作现场 `git stash` 一下，然后去修复bug，修复后，再 `git stash pop` ，回到工作现场。

Feature分支

- 开发一个新feature，最好新建一个分支；
- 如果要丢弃一个没有被合并过的分支，可以通过 `git branch -D <name>` 强行删除。

多人协作

- 查看远程库信息，使用`git remote -v`；
- 本地新建的分支如果不推送到远程，对其他人就是不可见的；
- 从本地推送分支，使用`git push origin branch-name`，如果推送失败，先用`git pull`抓取远程的新提交；
- 在本地创建和远程分支对应的分支，使用`git checkout -b branch-name origin/branch-name`，本地和远程分支的名称最好一致；
- 建立本地分支和远程分支的关联，使用`git branch --set-upstream branch-name origin/branch-name`；
- 从远程抓取分支，使用`git pull`，如果有冲突，要先处理冲突。

创建标签

- 命令 `git tag <name>` 用于新建一个标签，默认为 HEAD ， 也可以指定一个commit id；
- `git tag -a <tagname> -m "blablabla..."` 可以指定标签信息；
- `git tag -s <tagname> -m "blablabla..."` 可以用PGP签名标签；
- 命令`git tag`可以查看所有标签。

操作标签

- 命令 `git push origin <tagname>` 可以推送一个本地标签；
- 命令 `git push origin --tags` 可以推送全部未推送过的本地标签；
- 命令 `git tag -d <tagname>` 可以删除一个本地标签；
- 命令 `git push origin :refs/tags/<tagname>` 可以删除一个远程标签。

忽略文件

- 忽略某些文件时，需要编写`.gitignore`；
- `.gitignore`文件本身要放到版本库里，并且可以对`.gitignore`做版本管理！

Git使用

配置SSH公钥验证(免密码登录)

1. 首先进入账户目录下，查看是否有id_rsa.pub 和 id_rsa文件；
2. 如果没有，需要创建公钥和密钥：`ssh-keygen` ,此时需要输入两次密码，如果不需要设置密码，直接回车，默认生成的两个文件是：
 - id_rsa
 - id_rsa.pub
3. 查看并将公钥添加到git源码服务器即可。先查看公钥：`cat id_rsa.pub`，然后将公钥内容添加到服务器的`~/.ssh/authorized_keys` 文件中。

了解更多

- [Git官方文档](#)
- [廖雪峰Git教程](#)