Git常用命令

目录

[Git常用命令 1](#_Toc462337022)

[一、初始化 2](#_Toc462337023)

[二、查看状态和改动 2](#_Toc462337024)

[三、追踪、解除、移动 2](#_Toc462337025)

[四、提交 3](#_Toc462337026)

[五、撤销 3](#_Toc462337027)

[六、日志 4](#_Toc462337028)

[七、分支 4](#_Toc462337029)

[八、标签 6](#_Toc462337030)

[九、储藏 6](#_Toc462337031)

[十、清理 7](#_Toc462337032)

[十一、远程仓库 8](#_Toc462337033)

# 一、初始化

**git init**

初始化仓库。

**git clone [url]**

克隆一个仓库。

# 二、查看状态和改动

**git status**

检查当前文件状态

**git status -s**或**git status --short**

检查当前文件状态，更紧凑的输出：

**[?][?]**，新添加未追踪；

**[A][]**，新添加；

**[M][]**，被修改**并**放入暂存区；

**[][M]**，被修改**没**放入暂存区；

**[M][M]**，被修改放入暂存区，又被修改。

**git diff**

查看尚未暂存的文件更新了哪些部分。

**git diff --staged**

查看已暂存，将要添加到下次提交的内容。

# 三、追踪、解除、移动

**git add <file>**

追踪一个文件；将已追踪文件放到暂存区；把有冲突文件标记为已解决。

**git rm <file>**

解除追踪，并且从工作目录删除。

**git rm -f <file>**

解除追踪，并且从工作目录删除，强制执行，用于删除修改过并放入暂存区的文件。

**git rm --cached <file>**

解除追踪，保留文件在工作目录。

**git mv <file\_from> <file\_to>**

用于文件移动和重命名。

# 四、提交

**git commit**

将暂存区的文件提交，会进入编辑器，编辑此次提交的注释。

**git commit -v**

将暂存区的文件提交，会进入编辑器，编辑此次提交的注释，显示本次提交的修改。

**git commit -m “提交信息”**

将暂存区的文件提交，不会进入编辑提交信息，将提交信息附在-m命令后。

**git commit -a**

跳过git add，将已追踪的文件暂存起来一并提交。

**git commit --amend**

尝试重新提交，覆盖前一次提交的信息。

# 五、撤销

**git reset --soft [version-name]**

重置HEAD，不会重置暂存区和工作目录。

例如：执行**git reset --soft HEAD^**或**git reset --soft HEAD~1**，重置到上一个版本。

**git reset --mixed [version-name]**

重置HEAD、暂存区，不会重置工作目录。如果不加这种参数，默认就**--mixed**。

例如：执行**git reset --mixed HEAD^**或**git reset --mixed HEAD~1**，重置到上一个版本。

**git reset --hard [version-name]**

重置HEAD、暂存区、工作目录。

例如：执行**git reset --hard HEAD^**或**git reset --hard HEAD~1**，重置到上一个版本。

# 六、日志

**git log**

按照提交时间，列出所有更新，最近的更新排在最上面。

**git log -p**

显示每一次提交的内容差异。

**git log -[number]**

显示最近的几次提交，以括号内的数字为准。

**git log --stat**

查看每次提交的简略统计信息。

**git log --pretty=oneline**

将每次提交放在一行显示。

**git log --graph**

展示分支、合并历史。

# 七、分支

**git branch**

查看本地的分支列表。

**git branch -a**

查看所有分支，包括远程仓库。

**git branch -v**

查看每一个分支的最后一次提交。

**git branch [branch-name]**

创建一个分支。

**git branch -d [branch-name]**

删除一个分支。

**git branch --merged**

查看已经合并到当前分支的分支。

**git branch --no-merged**

查看未合并到当前分支的分支。

**git branch -u [remote-name]/[branch-name]**

设置已有的本地分支，跟踪一个远程分支。

**git branch -vv**

查看设置的所有跟踪分支。

**git checkout [branch-name]**

检出一个分支。

**git checkout -b [branch-name]**

创建并检出一个分支。

**git checkout -b [branch-name] [remote-name]/[branch-name]**

创建并检出一个分支，并追踪到远程分支。

**git checkout --track [remote-name]/[branch-name]**

创建并检出一个分支，并追踪到远程分支，git v1.6.2版本以上使用。

**git checkout -b [branch-name] [tag-name]**

从特定标签检出分支。

**git merge [branch-name]**

合并到当前分支。

**git rebase [branch-name]**

变基操作，将当前分支的提交变基到命令指定的分支。

例如：当前分支在**dev**，执行**git rebase master**，将**dev**的提交（指的是和**master**从共同祖先分叉后的提交）变基到**master**上。

**git rebase [branch-name] [branch-name]**

变基操作，将一个分支的提交变基到另一个分支上。

例如：执行**git rebase master dev**，将**dev**分支的提交，变基到**master**上。

**git rebase --onto [branch-name] [branch-name] [branch-name]**

变基操作，选择在一个分支不在另一个分支上的提交，变基到第三个分支上。

例如：执行**git rebase --onto master dev1 dev2**，选择在**dev2**不在**dev1**上的提交变基到**master**上。

# 八、标签

**git tag**

查看标签。

**git tag -l ‘tag-name’**

查看某一个系列的标签。

**git tag -a [tag-name] -m ‘标签信息’**

打附注标签。

**git show [tag-name]**

显示标签的信息。

**git tag [tag-name]**

打轻量标签。

**git tag -a [tag-name] 部分校验和**

给过去的提交打标签。

**git push origin [tag-name]**

将标签推送到远程仓库。

**git push origin --tags**

将所有标签推送到远程仓库。

# 九、储藏

**git stash**或**git stash save**

储藏操作，想要切换分支，但是不想提交修改，使用储藏就可以切换了。

**git stash --keep-index**

储藏操作，不会储藏在暂存区的文件。

应用场景：做了几个改动，只想提交其中一部分，过一会再处理剩余的改动。

**git stash --include-untracked**或**git stash -u**

储藏操作，会储藏未被追踪的文件。

**git stash list**

查看储藏栈上所有的储藏。

**git stash apply**

应用栈顶的储藏。

**git stash apply [stash-name]**

应用指定的储藏。

例如：执行**git stash apply stash@{1}**。

**git stash apply --index**

应用栈顶的储藏，如果储藏前有文件在暂存区，使用**--index**参数后，文件还会在暂存区。不使用的话，储藏前文件在暂存区，应用储藏后文件不会在暂存区。

**git stash pop**

应用栈顶的储藏，用后删除。

**git stash drop [stash-name]**

删除指定的储藏。

例如：执行**git stash drop stash@{1}**。

**git stash clear**

清除整个储藏栈。

**git stash branch [branch-name]**

检出储藏时所在的提交，并创建一个新分支应用储藏。

应用场景：如果在储藏后又提交了很多次，害怕应用储藏有冲突，就可以用这个命令。

# 十、清理

**git clean**

清除工作目录下没有忽略的未被追踪的文件。

**git clean -d**

清除工作目录下没有忽略的未被追踪的文件和空的子目录。

**git clean -f**

强制清除工作目录下没有忽略的未被追踪的文件。

**git clean -n**

查看会被清除的文件。

**git clean -x**

清除工作目录下未被追踪的文件，包括被忽略的文件。

# 十一、远程仓库

**git remote**

查看所有远程仓库。

**git remote -v**

查看远程仓库的的简写和URL。

**git remote show [remote-name]**

查看远程仓库的信息。

**git remote rename [remote-name-from] [remote-name-to]**

重命名远程分支的简写。

**git remote rm [remote-name]**

移除一个远程仓库。

**git fetch [remote-name] [remote-branch]:[local branch]**

访问远程仓库，拉取数据。

**git pull [remote-name] [remote-branch]:[local-branch]**

访问远程仓库，拉取数据，并做**merge**，相当于**git fetch**后**git merge**。

**git pull --rebase [remote-name] [remote-branch]:[local-branch]**

访问远程仓库，拉取数据，并做rebase，相当于**git fetch**后**git rebase**。

**git push [remote-name] [local-branch]:[remote-branch]**

推送到远程仓库。

**git push origin --delete [branch-name]**

删除一个远程分支。