目录

[checkout 2](#_Toc207219594)

[push 2](#_Toc207219595)

[branch 3](#_Toc207219596)

[stash 4](#_Toc207219597)

[pull 6](#_Toc207219598)

[reset 6](#_Toc207219599)

[remote 6](#_Toc207219600)

[fetch 7](#_Toc207219601)

[git add 7](#_Toc207219602)

[git commit 7](#_Toc207219603)

[git merge 8](#_Toc207219604)

[git status 8](#_Toc207219605)

## checkout

切换到已有的分支：

git checkout branch\_name

基于当前分支创建并切换到新的分支：

git checkout -b new\_branch\_name

## push

**将本地的指定分支推送到远程仓库：**

git push remote\_name branch\_name

例如，将本地的main分支推送到远程仓库origin：

git push origin main

**推送所有分支**

将所有本地分支推送到远程仓库：

git push --all remote\_name

例如，将所有分支推送到远程仓库origin：

git push --all origin

**强制推送**

如果在推送时遇到拒绝（例如，远程仓库有新的提交而本地没有更新），可以使用--force选项强制推送：

git push --force remote\_name branch\_name

请谨慎使用强制推送，因为可能会覆盖远程仓库的提交。

**设置上游分支**

在第一次推送时，可以使用-u选项设置上游分支，使以后可以直接使用git push进行推送而不需要指定远程和分支：

git push -u remote\_name branch\_name

例如，将本地的main分支推送到远程仓库origin并设置上游分支：

git push -u origin main

## branch

查看所有本地分支：

git branch

查看所有远程分支：

git branch -r

查看所有分支（包括本地和远程）：

git branch -a

查看当前所在的分支：

git status

删除本地分支feature：

git branch -d feature

强制删除本地分支feature：

git branch -D feature

删除远程分支：

git push <远程仓库名> --delete <branchname>

重命名当前分支为new\_name：

git branch -m new\_name

重命名指定的分支old\_name为new\_name：

git branch -m old\_name new\_name

查看所有分支及其最近提交：

git branch -vv（两个v）

## stash

git stash 命令可以临时保存当前工作目录的更改，以便切换到其他分支或处理其他任务

保存当前工作进度：

git stash

带有描述信息的 git stash

git stash save "your stash message"

查看存储列表：

git stash list

应用最近一次存储的进度：

git stash apply

应用并删除最近一次存储的进度：

git stash pop

应用指定的存储：

git stash apply stash@{n}

注意：该命令不会删除指定的存储

删除特定存储：

git stash drop stash@{n}

清空所有存储：

git stash clear

## pull

git pull 用于从远程获取代码并合并本地的版本。

git pull 是 git fetch 和 git merge 的简写，先从远程仓库获取最新的提交记录，然后将这些提交记录合并到你当前的分支中。

git pull [远程仓库名] [远程分支名]

例：

将远程主机 origin 的 master 分支拉取过来，与本地的 brantest 分支合并

git pull origin master:brantest

如果远程分支是与当前分支合并，则冒号后面的部分可以省略

git pull origin master

## reset

git reset --hard <分支名>或<提交>或<标签>

将当前分支强制重置为 指定分支或提交 的状态，丢弃本地所有未提交的更改和提交后的内容，更新工作目录文件

## remote

git remote add <自定义名称> <远程仓库SSH地址>

添加一个远程仓库，可用自定义名称快捷访问该仓库

## fetch

git fetch --prune

同步远程仓库的分支信息，并在本地删除已不存在的远程分支

## git add

将文件更改从工作目录添加到暂存区

**添加单个文件：**

git add <文件路径>

**添加多个文件：**

git add <文件1> <文件2> ...

**添加所有修改过的文件和新文件：**

git add .

使用一个点 . 表示当前目录及其下的所有子目录中的修改都将被添加到暂存区。这包括新文件（未跟踪的文件）和已修改但尚未暂存的文件。

**交互式添加：**

git add -i

打开一个交互式的界面，允许你有选择性地暂存特定文件的部分更改，而不是全部更改。这对于只需要部分提交的情况特别有用。

**补丁模式：**

git add -p <文件路径>

类似于交互模式，但是它让你以更细粒度的方式选择要添加到暂存区的具体代码块，而不是整个文件。

## git commit

将暂存区中的更改永久保存到本地仓库

git commit -m "提交信息"

这里的 -m 参数允许指定提交信息。提交信息应当清晰地描述此次提交所包含的更改内容，以便日后查阅和理解。

**提交流程**

修改文件：首先，在你的工作目录中对文件进行添加、删除或修改。

暂存更改：使用 git add 将需要提交的文件添加到暂存区。

提交更改：运行 git commit 来创建一个提交对象，其中包含了暂存区的所有更改和一条描述性的提交信息。

## git merge

把一个分支的历史合并到当前所在的分支上

git merge origin/main

**说明：**

将远程分支 origin/main 合并到当前所在本地分支

## git status

显示工作目录和暂存区的状态

**详细：**

当前所在的分支

修改尚未加入提交（Changes not staged for commit）：列出已经修改但还没有通过 git add 添加到暂存区的文件。这些更改不会被包含在下次提交中，除非你先将它们添加到暂存区。

要提交的变更（Changes to be committed）：列出已经被 git add 添加到暂存区的更改。这些更改会在下一次执行 git commit 时被保存。

未跟踪的文件（Untracked files）：列出 Git 中从未被跟踪过的文件，即新文件，Git 不知道该如何处理这些文件。

没有变化的分支提示：如果工作目录是干净的，没有任何未提交的更改或未跟踪的文件，它会提示“nothing to commit, working tree clean”。