

本文整理了一些常用的 Git 操作，老司机可以温故知新，新手可以点赞收藏。文末提供了入门教程及学习资源，请自行下滑~

## 配置操作

---

### 全局配置

```
git config --global user.name '你的名字'
git config --global user.email '你的邮箱'
```

### 当前仓库配置

```
git config --local user.name '你的名字'
git config --local user.email '你的邮箱'
```

### 查看 global 配置

```
git config --global --list
```

### 查看当前仓库配置

```
git config --local --list
```

### 删除 global 配置

```
git config --unset --global 要删除的配置项
```

### 删除当前仓库配置

```
git config --unset --local 要删除的配置项
```

# 本地操作

---

## 查看变更情况

```
git status
```

## 将当前目录及其子目录下所有变更都加入到暂存区

```
git add .
```

## 将仓库内所有变更都加入到暂存区

```
git add -A
```

## 将指定文件添加到暂存区

```
git add 文件1 文件2 文件3
```

## 比较工作区和暂存区的所有差异

```
git diff
```

## 比较某文件工作区和暂存区的差异

```
git diff 文件
```

## 比较暂存区和 HEAD 的所有差异

```
git diff --cached
```

## 比较某文件暂存区和 HEAD 的差异

```
git diff --cached 文件
```

## 比较某文件工作区和 HEAD 的差异

```
git diff HEAD 文件
```

## 创建 commit

```
git commit
```

## 将工作区指定文件恢复成和暂存区一致

```
git checkout 文件1 文件2 文件3
```

## 将暂存区指定文件恢复成和 HEAD 一致

```
git reset 文件1 文件2 文件3
```

## 将暂存区和工作区所有文件恢复成和 HEAD 一样

```
git reset --hard
```

## 用 difftool 比较任意两个 commit 的差异

```
git difftool 提交1 提交2
```

## 查看哪些文件没被 Git 管控

```
git ls-files --others
```

## 将未处理完的变更先保存到 stash 中

```
git stash
```

## 临时任务处理完后继续之前的工作

- pop 不保留 stash
- apply 保留 stash

```
git stash pop
```

```
git stash apply
```

## 查看所有 stash

```
git stash list
```

## 取回某次 stash 的变更

```
git stash pop stash@{数字n}
```

## 优雅修改最后一次 commit

```
git add.
```

```
git commit --amend
```

## 分支操作

---

### 查看当前工作分支及本地分支

```
git branch -v
```

### 查看本地和远端分支

```
git branch -av
```

### 查看远端分支

```
git branch -rv
```

### 切换到指定分支

```
git checkout 指定分支
```

### 基于当前分支创建新分支

```
git branch 新分支
```

### 基于指定分支创建新分支

```
git branch 新分支 指定分支
```

## 基于某个 commit 创建分支

```
git branch 新分支 某个 commit 的 id
```

## 创建并切换到该分支

```
git checkout -b 新分支
```

## 安全删除本地某分支

```
git branch -d 要删除的分支
```

## 强行删除本地某分支

```
git branch -D 要删除的分支
```

## 删除已合并到 master 分支的所有本地分支

```
git branch --merged master | grep -v '^*\| master' | xargs -n 1 git branch -d
```

## 删除远端 origin 已不存在的所有本地分支

```
git remote prune origin
```

## 将 A 分支合入到当前分支中且为 merge 创建 commit

```
git merge A分支
```

## 将 A 分支合入到 B 分支中且为 merge 创建 commit

```
git merge A分支 B分支
```

## 将当前分支基于 B 分支做 rebase，以便将B分支合入到当前分支

```
git rebase B分支
```

## 将 A 分支基于 B 分支做 rebase，以便将 B 分支合入到 A 分支

```
git rebase B分支 A分支
```

## 变更历史

---

### 当前分支各个 commit 用一行显示

```
git log --oneline
```

### 显示就近的 n 个 commit

```
git log -n
```

### 用图示显示所有分支的历史

```
git log --oneline --graph --all
```

## 查看涉及到某文件变更的所有 commit

```
git log 文件
```

## 某文件各行最后修改对应的 commit 以及作者

```
git blame 文件
```

## 标签操作

---

### 查看已有标签

```
git tag
```

### 新建标签

```
git tag v1.0
```

### 新建带备注标签

```
git tag -a v1.0 -m '前端食堂'
```

### 给指定的 commit 打标签

```
git tag v1.0 commitid
```

### 推送一个本地标签

```
git push origin v1.0
```

### 推送全部未推送过的本地标签

```
git push origin --tags
```

### 删除一个本地标签

```
git tag -d v1.0
```

### 删除一个远端标签

```
git push origin :refs/tags/v1.0
```

## 远端交互

---

### 查看所有远端仓库

```
git remote -v
```

### 添加远端仓库

```
git remote add url
```

### 删除远端仓库

```
git remote remove remote的名称
```

## 重命名远端仓库

```
git remote rename 旧名称 新名称
```

## 将远端所有分支和标签的变更都拉到本地

```
git fetch remote
```

## 把远端分支的变更拉到本地，且 merge 到本地分支

```
git pull origin 分支名
```

## 将本地分支 push 到远端

```
git push origin 分支名
```

## 删除远端分支

```
git push remote --delete 远端分支名
```

```
git push remote :远端分支名
```