


Git版本控制器的使用

Git是一个开源的分布式版本控制系统。

 分布式:每个人的本地都有一个独立的版本库，开发者自己管理版本库。

版本控制:一种记录一个或多个文件内容变化，查看特定版本情况的系统。

✧ 下载安装

官网: [Git \(git-scm.com\)](https://git-scm.com)

下载对应版本安装即可

✧ 配置作者信息

配置用户名

```
1 git config --global user.name xxxx
2 或 git config --global user.name '中文名字'
```

配置邮箱

```
1 git config --global user.email xxxx@qq.com
```

✧ 常用命令

查看版本信息

```
1 git --version
```

查看用户配置信息

```
1 git config --list
```

初始化本地仓库

```
1 git init
```

查看文件当前状态

```
1 git status
```

跟踪文件(添加文件到暂存区)

```
1 git add filename 单个文件
2 git add . 添加全部文件
```

提交到仓库

```
1 git commit -m '提交信息'
```

增补提交(修改最新的提交描述，覆盖旧的)

```
1 git commit --amend
```

直接提交到本地仓库

-a把已经跟踪的文件提交

```
1 git commit filename - m '信息'
2 git commit -a -m '信息'
```

查看提交记录日志

```
1 git log
2 git log -p 查看文件变动信息
3 git log -p -1 查看最近一次提交信息
4 git log --oneline 每次提交信息显示在一行
5 git log --name-only 查看文件的变化
6 git log --name-status 查看文件类型的变化
```

删除

```
1 git rm filename 版本库与工作区同时删除
2 git rm --cached filename 删除暂存区、版本库文件
```

文件从暂存区恢复到工作区

```
1 git restore file
2 git checkout -- file
```

本地仓库恢复到暂存区

```
1 git restore --staged file
2 git reset -- file
```

修改文件名称

```
1 git mv oldname nowname
```

最新的提交覆盖工作区、暂存区

```
1 git checkout HEAD -- filename
```

版本回退

```
1 git reset id 回退版本提交的前四位id
2 git reset --hard HEAD 将工作区、暂存区回退到上一个版本
3 git reset HEAD 撤销最新的提交,并生成新的提交
```



版本对比

```
1 git diff 对比工作区、暂存区的变化、
2 git diff --cached 对比仓库最新一次提交与暂存区变化
3 git diff --staged
4 git diff id id 对比两次提交的版本差异
5 git diff HEAD 对比仓库与工作区差异
```

分支

```
1 git branch 查看分支
2 git branch -a 查看远程仓库分支
3 git branch 分支名字 在当前分支上创建新分支
4 git checkout 分支名 切换分支
5 git switch 分支名
6 git checkout -b 分支名 创建并且切换分支
7 git branch -v 查看分支最后一次提交信息
8 git branch -d 分支名 删除分支
```

合并分支

- 1 `git merge 分支名` 合并分支到主分支(先切换到主分支)
- 2 `git branch --merged` 查看已合并的分支
- 3 `git branch --no-merged` 查看没有合并的分支
- 4 `git branch -D 分支名` 删除没有合并的分支

合并优化

- 1 `git rebase master` 将子分支提交点移动到主分支的最新点

储藏

- 1 `git stash` 储藏当前内容
- 2 `git stash apply` 恢复储藏
- 3 `git stash list` 储藏区
- 4 `git stash drop 储藏标识` 删除储藏区
- 5 `git stash pop` 恢复并且删除储藏区

标签

- 1 `git tag` 查看标签
- 2 `git tag -a 标签名 -m '说明信息'` 新建标签
- 3 `git show 标签名` 查看标签版本信息
- 4 `git tag -a 标签名 提交id -m '信息'` 给某一次提交追加标签
- 5 `git tag -d 标签名` 删除标签
- 6 `git push 远程仓库名字 标签名` 推送到远程仓库
- 7 `git checkout -b 本地分支名 标签名` 通过标签获取对应版本

生成zip压缩代码发布压缩包

- 1 `git archive 分支名字 --prefix='生成文件夹的名字/' --format=zip`
> 压缩包名字.zip

连接远程仓库

- 1 `git remote add origin ssh链接` origin是远程仓库别名
- 2 `git remote -v` 查看远程仓库
- 3 `git remote rm origin` 删除与远程仓库的关联
- 4 `git push -u origin "master"` 推送到远程仓库的master分支
- 5 `git push --set-upstream origin 分支名` 向远程仓库分支关联
- 6 `git pull origin 远程分支名字:本地分支名字` 获取远程仓库分支代码
- 7 `git push origin --delete 分支名字` 删除远程仓库分支

多库提交

```
1 # 在增加一个github远程库
2 git remote add github 仓库地址
```

* 快捷指令配置

通过创建命令别名可以减少命令输入量，有几种方式进行设置

配置文件定义

修改配置文件 `~/.gitconfig` 并添加以下命令别名配置段

```
1 [alias]
2     a = add .
3     c = commit
4     s = status
5     l = log
6     b = branch
```

现在可以使用 `git a` 实现 `git add .` 一样的效果了。

系统配置定义

window 用户可以修改 `~/.bashrc` 或 `~/.bash_profile` 文件。

mac/linux 修改 `~/.zshrc` 文件中定义常用的别名指令，需要首先安装 `zsh` 命令行扩展

```
1 alias gs="git status"
2 alias gc="git commit -m "
3 alias gl="git log --graph --pretty=format:'%Cred%H%Creset -
  %C(yellow)%d%Creset %s %Cgreen(%cr) %C(bold blue)<%an>%Creset' --
  abbrev-commit"
4 alias gb="git branch"
5 alias ga="git add -A"
6 alias go="git checkout"
7 alias gp="git push;git push github"
8 alias gp="git push & git push github"
```

命令行直接使用 `gs` 即可以实现 `git status` 一样的效果了。



window 系统需要使用 git for window 中的 `Git Base` 软件

