Git使用笔记

查看Git版本

```
git --version
```

Git配置

Git必要配置-用户名与邮箱

```
git config --global user.name <用户名>
git config --global user.email <邮箱>
```

Git配置查看

```
git config --list
或者
git config --list --global          //--排名不分先后
git config --list --local
```

Git配置的作用域

```
git config --local          //当前Git工作区的配置
git config --global         //全局Git工作区的配置
git config --system         //系统Git工作区的配置
```

Git重置配置

```
git config --unset --local  //Git配置重置
```

补充说明

1. `--system` 选项的 `git config` 存放于 `/etc/gitconfig` 文件
2. `--global` 选项的 `git config` 存放于 `~/.gitconfig` 或 `~/.config/git/config` 文件：只针对当前用户
3. 当前使用仓库的 Git 目录中的 `config` 文件（就是 `.git/config`）：针对当前仓库

Git使用

Git项目初始化

```
git init
或者
git init <项目名>
```

Git添加文件到暂存区

```
git add <path/file>     //Git 初始化后 使用 add 来 将file添加到暂存区
git add -u                      //Git 添加 所有tracked文件中被修改过或已删除文件的信息
git add -A                      //Git 添加 所有文件中被修改过或已删除文件的信息 和 未追踪的文件
git add -h                      //Git add 帮助说明
```

Git 提交文件到 版本库

```
git commit -m '版本自定义说明文本' //Git 将暂存区修改 提交到版本库 并给commit标识
git commit -am '版本自定义说明文本'
git commit -h                                          //Git commit 帮助信息
git commit --amend                                     //将最近的修改 附加到 上一次的 commit 上
```

查看 工作区 暂存区文件状态

```
git status
```

查看 commit 历史记录

```
git log
        --oneline               //一条 commit 在一行显示
        --graph                 //显示 路径 线条
```

Git回滚/恢复到指定版本

工作区 回滚到 暂存区

```
git checkout .                                  //还原工作区 <所有> 文件 为暂存区状态暂存区状态
git checkout -- filename        //还原工作区 <指定> 文件 为暂存区状态暂存区状态
```

暂存区 回滚到 指定版本

```
git reset HEAD                          // 恢复暂存区到 HEAD 指针指向的
位置
git reset HEAD^                         // 恢复暂存区到 HEAD 的上一次
commit
git reset HEAD~4                        // 恢复暂存区到 HEAD 的上第4次
commit

git reset <hash>                        // 恢复暂存区到 指定hash
commit处

git reset 缺省值为 mixed                //将HEAD reset 没有修改 工作区
但是修改了 暂存区

git reset --soft                        //将HEAD reset 没有修改 工作区 与
暂存区

git reset --hard                        //将HEAD reset 同时修改工作区与暂
存区
```

工作区 回滚到 指定版本

```
git checkout <hash>
git checkout <指定分支名>
```

Git 分支

Git 分支建立与切换

```
git branch <branch_name>                //建立分支
git checkout <branch_name>          //切换到 branch_name 的分支

等同于
git checkout -b <branch_name>       //新建 并切换到 branch_name 的分支
```

Git 已有分支查看

```
git branch                          //查看分支
git branch -v                       //查看每一个分支的最后一次提交
git branch -a                       //查看所有分支 ， 包含 远端分支
git branch -av                    //查看所有分支 以及 最后一次提交
```

Git分支删除

```
git branch -d <branch_name>         //删除 指定分支
git branch -D <branch_name>         //强制删除 指定分支，删除分支并丢掉所做的修改
```

Git 分支合并

```
git checkout master                     //切换到分支 master
git merge <branch_name>             //将指定分支 的修改 合并到 master 上
```

Git 远程管理

git remote 操作

```
git remote -v                                   //查看远程信息
git remote add <name> <url>     //添加 远程 同时可以指定 branch 以及其他
git remote rename <old> <new>   //修改 远程 名称
git remote remove <name>            //删除 远程 引用
```

git clone 操作

```
git clone [url/path]
//从指定的 URL或路径 克隆仓库到本地
git clone --bare file:///path/……/project_name name.git
//将指定工程的版本库 克隆或者备份到 当前位置，不加--bare原样克隆
```

【案例1】clone一个github项目

```
git clone https://github.com/leobod/GitGuide.git        //克隆当前项目到本地
git remote -v
    shell显示
    origin  git@github.com:leobod/GitGuide.git (fetch)
    origin  git@github.com:leobod/GitGuide.git (push)
    克隆的项目 远程名一般为 origin  远端分支一般叫 origin/master 或者其他
```

【案例2】远程连接github项目

```
git remote add github https://github.com/leobod/GitGuide.git
git remote -v
    shell显示
    github  git@github.com:leobod/GitGuide.git (fetch)
    github  git@github.com:leobod/GitGuide.git (push)
    请类比上文
```

补充说明

对于自己开发的项目或者需要在多个设备上修改的项目，最好用 案例2的方式

好处是当你在控制台生成ssh-keygen并将公钥存放到Github时，每次pull不需要反复验证

Git仓库与本地联动

拉取/推送命令

```
git fetch                          //从指定 远程 拉取文件到版本库
git pull                           //从指定 远程 拉取并合并到 版本库
git push                           //将版本库 推送到 远程仓库去

git push  --set-upstream origin master
git fetch origin master
```

仓库与本地分支关联

```
git branch --set-upstream-to=origin/remote_branch local_branch
也可以使用
git branch -u origin/remote_branch local_branch

撤销分支关联
git branch --unset-upstream local_branch

查看所有分支
git branch -av
查看分支映射关系
git branch -vv
```