

1、初始化

```
1 # 在使用git上传文件之前，需要配置以下信息
2 git config --global user.name "lty"
3 git config --global user.email 771153907@qq.com
4 git config --global credential.helper store
5
6 # 查看信息
7 git config --global --list
```

2、创建仓库

```
1 # 创建一个新的本地仓库（省略project-name则在当前目录创建）
2 git init <project-name>
3
4 # 下载远程目录
5 git clone <url>
6
7 # 克隆某一个仓库分支
8 git clone -b <分支名> <仓库地址>
```

3、添加和提交文件

```
1 git status # 查看仓库状态
2
3 git add # 添加到暂存区
4 可以使用通配符，例如：git add *.txt
5 将当前目录所有文件添加进去，例如：git add .
6
7 git commit # 提交
8 只提交暂存区中的内容，不会提交工作区中的内容
9 提交所有暂存区的文件到仓库 git commit -m "message"
10 提交所有已修改的文件到仓库 git commit -am "message"
11
12 git log # 查看仓库中的提交历史记录
13 可以使用 --oneline 参数来查看简洁的提交记录
```

4、回退到某一版本

```
1 git reset --soft # 保存工作区和暂存区的内容
2 git reset --hard # 丢弃工作区和暂存区的内容
3 git reset --mixed # 保存工作区的内容，丢弃暂存区的内容
4
5 !!! 注意后面要加版本码， 可以使用git log --oneline来查看版本码
```

5、查看差异

```
1 git diff # 默认查看工作区和暂存区的内容
2 git diff HEAD # 比较工作区和版本库之间的差异
3 git diff --cached # 查看暂存区和版本库之间的差异
4
5 git diff HEAD~ HEAD # 比较当前版本与上一个版本之间的差异，HEAD~等价于HEAD~1或者HEAD^
6 git diff HEAD~2 HEAD # 比较当前版本与上两个版本之间的差异
```

6、删除文件

```
1 rm <file>; git add <file> 先从工作区删除文件，然后在暂存区删除内容
2 git rm <file> # 把文件从工作区和暂存区同时删除
3 -add <file> 把文件从暂存区删除，但保留在当前工作区中
4 git rm -r * 递归删除某个目录下的所有子目录和文件
5
6 删除后不要忘记提交
```

7、.gitignore忽略文件

```
1 # 忽略所有的 .a文件
2 *.a
3
4 # 但跟踪所有的 lib.a 文件，即便你在前面忽略了 .a 文件
5 !lib.a
6
7 # 只忽略当前目录下的 TODO 文件，而不忽略 subdir/TODO
8 /TODO
9
10 # 忽略任何目录下名为 build 的文件夹
11 build/
12
13 # 忽略 doc/notes.txt，但不忽略doc/server/arch.txt
14 doc/*.txt
15
16 # 忽略 doc/ 目录及其所有子目录下的 .pdf 文件
17 doc/**/*.pdf
```

8、GitHub首次配置SSH秘钥

注意网络问题，要连guide01，然后开vpn。

```
1 进入 /home目录下的./.ssh文件
```

```
1 ssh-keygen -t rsa -b 4096 # 不是root用户的时候，需要修改./ssh目录的权限
2 私钥文件: id_rsa
3 公钥文件: id_rsa.pub
```

如果之前生成过SSH秘钥，则在.SSH文件中还要配置config文件，如下

```
1 touch config # 新建文件
2
3 在文件中添加如下5行
4 # github
5 Host github.com
6 HostName github.com
7 PreferredAuthentications publickey
8 IdentityFile ~/.ssh/id_rsa # id_rsa代表创建的私钥文件
```

```
1 克隆仓库 git clone <repo-address>
2 推送更新内容 git push <remote> <branch>
3 拉取更新内容 git pull <remote>
```

9、关联本地仓库和远程仓库

首先添加一个远程仓库并取别名为"origin"。如下图

```
1 git长度。。 git remote add origin git@github.com:lty7711/test.git
```

查看当前仓库所对应远程仓库的别名和地址

```
1 git remote -v
```

如果远程仓库的master分支已经有文件，则还需要执行下列命令，现将远程仓库的文件拉取到本地仓库

```
1 git pull --rebase origin master
```

将远程仓库的master分支与本地的master分支关联起来

```
1 git push -u origin master
```

10、保存更改


- 1、暂存更改（Stash）：将未保存的更改存储到一个临时区域，可以在切换到另一个分支后恢复这些更改。
- （1）使用 'git stash' 命令来保存你的当前更改。
- （2）切换分支： 'git checkout <branch-name>'
- （3）要恢复更改，使用 'git stash pop'

11、查看当前有哪些分支

- 1 查看本地git仓库有哪些分支
- 2 git branch
- 3
- 4 查看远程仓库分支
- 5 git branch -a

12、实战训练

12.1 在github上添加SSH秘钥

-  ssh-keygen -t rsa "XXXX" // 使用ssh-keygen 命令生成公钥和私钥，并将公钥按照如下流程上传到github上面，其中"XXXX"这部分内容可省略，它表示一个特殊身份标记，有利于别人识别

步骤如下：



1、登录github

2、进入设置页面

点击右上角头像，选择Settings

3、进入设置页面

在左侧菜单找到“SSH and GPG keys”选项

4、添加新密钥

点击“New SSH key”按钮

5、填写信息

a. **Title**: 给密钥起个名字（比如“我的电脑”）

b. **Key**: 粘贴完整的公钥内容（包括ssh-rsa开头的整行）

6、保存

点击“Add SSH key”

7、连接测试

设置完成后，在终端测试：

```
ssh -T git@github.com
```

如果看到类似“Hi username! You've successfully authenticated...”的消息表示成功。

12.2 实现本地仓库和github仓库相关联



1. 先在GitHub创建仓库

- a. 登录GitHub
- b. 点击右上角"+"→"New repository"
- c. 添加仓库名称
- d. **不要勾选** "Initialize with README" (因为你本地已有文件)
- e. 点击 "Create repository"

2. 在本地文件夹初始化Git

- a. cd 你的文件夹路径
- b. git init

3. 添加文件到暂存区

- a. git add .

4. 提交文件

- a. git commit -m "添加文件"

5. 添加远程仓库地址

- a. git remote add origin git@github.com:你的用户名/仓库名.git

6. 推送到 GitHub

- a. git push -u origin main

完成上述操作后，本地文件夹就会出现在Github上了

12.3 创建新分支



git checkout -b 新分支名称 // 创建新分支

git push -u origin 新分支名 // 将新分支推送到远程仓库

- git push - 推送命令
- -u - 设置上游分支（第一次推送时使用）
- origin - 远程仓库名
- branch2 - 要推送的分支名

git checkout 分支 // 切换分支对应仓库