GIT USAGE

WHAT'S GIT

- 分布式代码管理工具
- 所有修改历史都保存在本地
- 保证存储的完整性
- 所有修改都是递增的

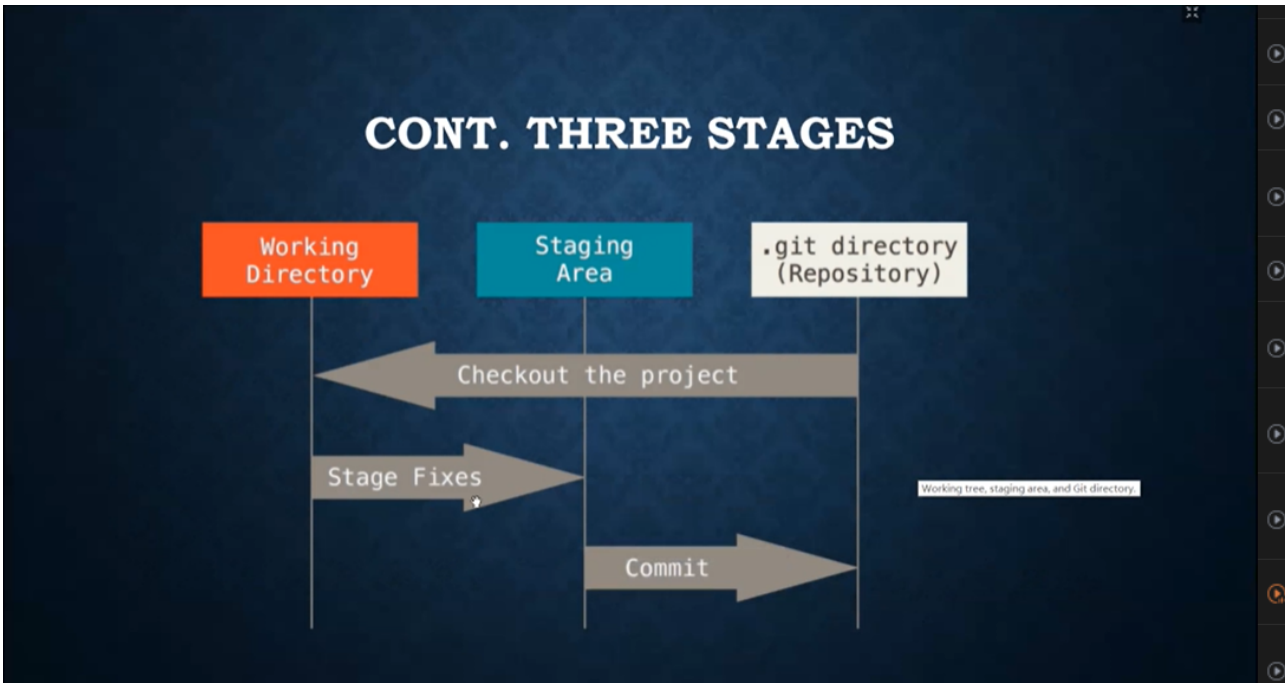
  （没有回退的概念，所有改动都是有迹可循，包括错误的的改动）；

  二分查找法查找问题的所在，可以切换到不同的版本进性测试，查找错误；
- 支持多个远端的源输入

  git 可以放到很多server上，可以在很多源放在同一个目录，对不同的源进行同步；
- 支持多个分支
- 支持标签

  快照
- 支持不同分支之间的各种同步操作

GIT 的三个阶段



check out : 从远端或者本地 checkout 一个branch

stage fixes： 做的一些修改

commit： 把改动commit上来

```
git status //获取GIT的一些状态，文件的增删改查
```

git tools

- GIT:
    - GIT Bash
- Tortoisegit GUI GIT:

```
[pull]
rebase = true // 先将远端的改动加载到本地的分支，然后再将本地的改动加载到最后面；把自己的改动加载到历史上的最顶端
```

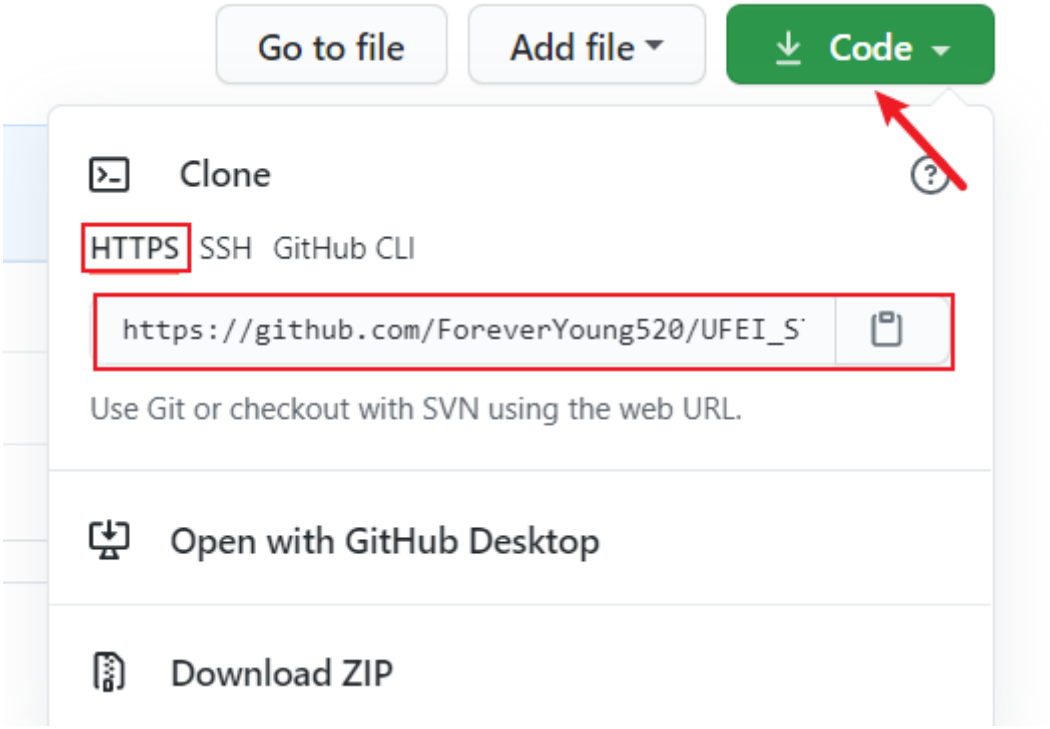
CONT(.SSH/CONFIG)

两种方式：

- HTTP:简单，但每次要输入用户名和密码；（gitlab）
- SSH:通过公钥和密钥这种方式，公钥上传到GITHUB,本地要配置ssh/Config，不需要用户名和密码；

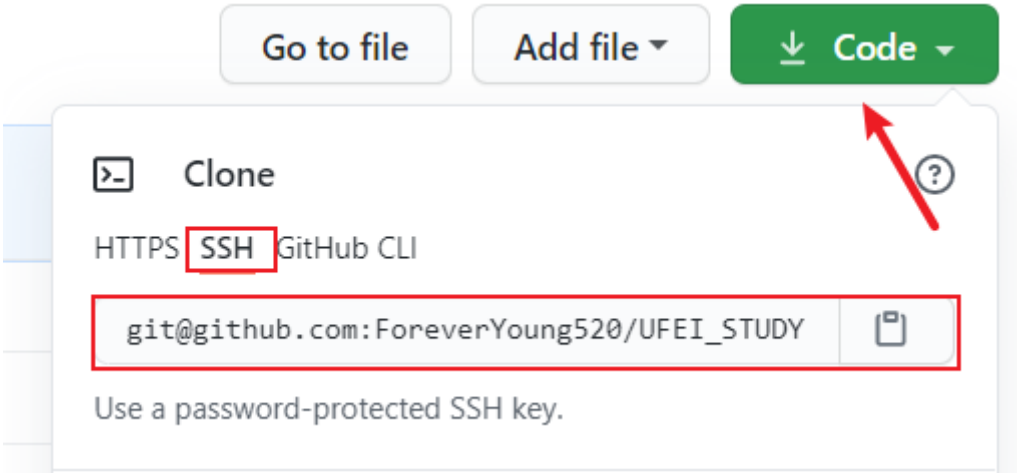
CLONE

- HTTP :




```
git clone  (+url地址)
```

- SSH:




```
git clone  (+ssh地址) -b master //可以选择克隆的分支
cd directory     //
git submodule update -init    //目录内容来自另外一个GIT project ,作用引用别的project 的内容;
```

REMOTE

```
add remote
    git remote add remote_name  https://链接远程仓库
show Remote
    git  remote -v  //server端有多少个
remove remote
    git remote remove remote_name  //删除源
```

BRANCH

- ADD local branch

```
git checkout origin/master -b branch  //添加一个分支
```

- Delete local branch

```
git delete -D test(分支名) //删除分支
```

- Switch local branch

```
git checkout test (分支名)  // 选择分支
```

- push remote branch

```
git push origin local_branch :remote _branch
```

- Delete remote branch

```
git push origin --delete remote_branch
```

UPDATE

- Fetch remote update

```
git fetch origin  //把远端的改动下载到本地，但是不会修改本地的代码，
只是把修改历史下载到.git目录下，本地的代码并没有更新；
//使用命令需要联网
```

- Update local code

```
git rebase origin/master //不需要联网 从local  origin/master 把相应的改动应用到自己的local源码上

git reset origin/master --hard   //其他的改动都不需要了，把本地的代码和远端的代码一致

git pull origin //->相当于git fetch origin 和git rebase origin/master命令的集合
```

- Push local commit to remote

```
git push origin local_branch :remote_branch
```

PATCH

- Add new commits

```
git status //产看有哪些改动
git add (changed _files)// 添加改动文件
git commit -m "commit message//添加注释
```

- Revert the commits

```
git reset HEAD ~ n  //保留改动，不保留commit;
git reset HEAD ~ n  --hard//不保留commit，不保留改动;

    //用处：当不想要不必要的改动时，用到这两个命令
```

- Create git patches

```
git format-patch -n
git send-email *.patch
```

- Apply git patches

```
git  am  --3way  --ignore-space-change --keep-cr *.patch
git cherry-pickcommit_hash_value
```