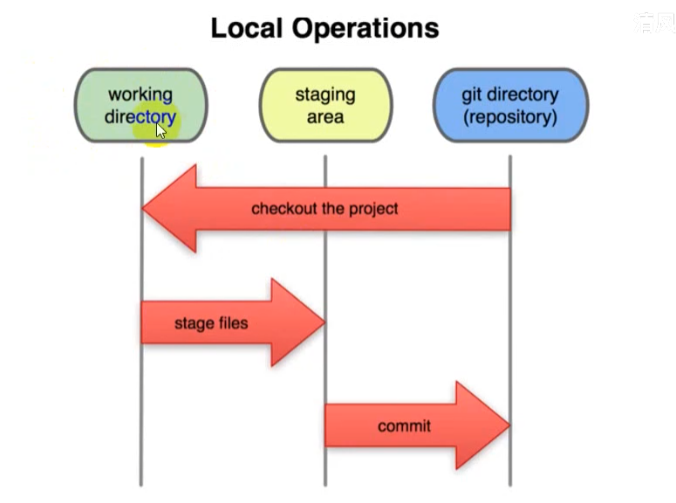
# 一、Git文件的三种状态与工作模式



1、已提交（committed），数据已经安全的保存到本地库中

2、已修改（modified），文件已修改，还未保存到本地库中

3、已暂存（staged），对一个已修改的文件的当前版本做了标记，使之饱汉子啊下次提交的快照中。

# 二、常用命令

## 1、配置

git config --global user.name “用户名”

git config --global user.email “邮箱”

## 2、查看配置

git config --list

## 3、查看版本号

git --version

## 4、初始化一个本地仓库

cd <path>（如果含有空格，空格前加上“\”）

git init

## 5、将修改文件提交到暂存区和从暂存区撤回

（1）git add <path>

path可以是文件也可以是目录，git可以判断出path中修改和新增的文件（不包括删除的），并把它们新增到索引库中。

（2）git reset HEAD

将本次已存到暂存区的文件撤回，变为未追踪的状态。

## 6、查看暂存区待提交的文件

git status

用于显示工作目录和暂存区的状态。可以看到哪些修改被提到暂存区了，哪些没有提到暂存区；以及哪些修改没有被追踪到（tracked）

## 7、提交暂存区里的内容到本地库中

git commit -m [修改描述]

将索引的当前内容与描述更改的用户和日志消息一起存储在新的提交中。

## 8、查看提交日志

git log 用于显示提交日志信息

git log -5 --pretty=oneline 简化输出日志

## 9、清除缓存区文件

git rm --cached <file>

## 10、版本回退

git reset --hard HEAD^

1个“^”表示回退1个版本，2个表示回退2个版本。

git reset --hard HEAD~1

表示回退到1个版本，若数字为n则回退n个版本。

git reset --hard <版本标识符>

输入版本标识符前n位（n≥7，且不与其他版本的标识符前几位有冲突），表示回退指定版本去。

git reflog

显示记录在本地的HEAD和分支引用在过去指向的位置

## 11、本地误删恢复

git checkout -- <file>

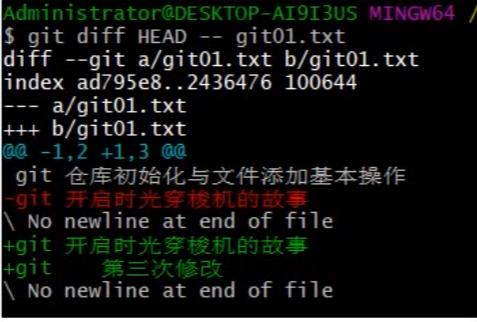
## 12、查看目录

（1）git ls-files 查看本地仓库

（2）ls 看工作区目录

## 13、没有被暂存就直接提交了

先执行：git diff HEAD -- <file>



对比 文件在执行git commit前后的变化

---表示修改之前

+++表示修改之后的

@@ -1,2 +1,3 @@ 这行代码表示：

修改前的文件从第1行开始总共变动了2行

修改后的文件从第1行开始总共变动了3行。

## 14、克隆远程项目到本地

cd <path> (path 为本地存储目录)

git clone <URL>

## 15、将本地库推送到远程仓库

在本地库执行以下操作

git init

git add README.md

git commit -m “Ver.1.0.0”

（1）在github 通过https方式上传

在github上生成URL （例：https://github.com/wangjianhf/socialcode.git）

git remote add origin <URL>

git push -u origin master

（2）通过ssh协议上传

①使用本地git客户端生成SSH公钥与私钥

命令：ssh-keygen -t rsa -C [wangajin\_hf@163.com](mailto:wangajin_hf@163.com)，之后不要输入任何内容，一直回车。

②在github上绑定公钥（点击头像>setting> SSH and GPG keys>new SSH key）

③验证公钥是否绑定

命令：ssh -T [git@github.com](mailto:git@github.com) 然后输入 yes 见到如即成功了



在github上生成URL （例：git@github.com:wangjianhf/socialcode.git）

git remote add origin <URL>

git push -u origin master

# 三、分支操作

## 1、本地分支操作

|  |  |
| --- | --- |
| 命令 | 描述 |
| git checkout <分支名称> | 切换到指定分支 |
| git checkout -b <分支名称> | 新建分支并切换到此分支 |
| git branch -d <分支名称> | 删除指定分支 |
| git branch | 查看所有分支，标“\*”为当前所在分支 |
| git merge <分支名称> | 合并分支 |
| git merge master | 在分支下执行此命令  将master内容同步到分支 |
| git branch -m|-M <旧分支名称> <新分支名称> | 重命名分支，如果newbranch名字分支已经存在，则需要使用-M强制重命名 否则，使用-m进行重命名。 |

## 2、分支的push与pull

|  |  |
| --- | --- |
| 命令 | 描述 |
| git branch -a | 查看本地与远程分支 |
| git push origin <本地分支名称> | 推送本地分支到远程 |
| git push origin :<远程分支名称> | 删除远程分支(本地分支还在保留) |
| git fetch | 读取远程仓库状态 |
| git checkout -b <本地分支名称> origin /<远程分支名称> | 拉取远程指定分支并在本地创建分支 |
| git pull | 更新远程仓库到本地仓库 |

## 3、标签**管理**

|  |  |
| --- | --- |
| 命令 | 描述 |
| git tag | 查看所有标签 |
| git tag <标签名称> | 新建标签默认为HEAD |
| git tag -a <标签名称> -m 'xxx' | 添加标签并指定标签描述信息 |
| git tag -d <标签名称> | 删除一个本地标签 |
| git push origin <标签名称> | 推送本地标签到远程 |
| git push origin --tags | 推送全部未推送过的本地标签到远程 |
| git push origin:refs/tags/<标签名称> | 删除一个远程标签 |