# Git的使用

## 一、Git介绍

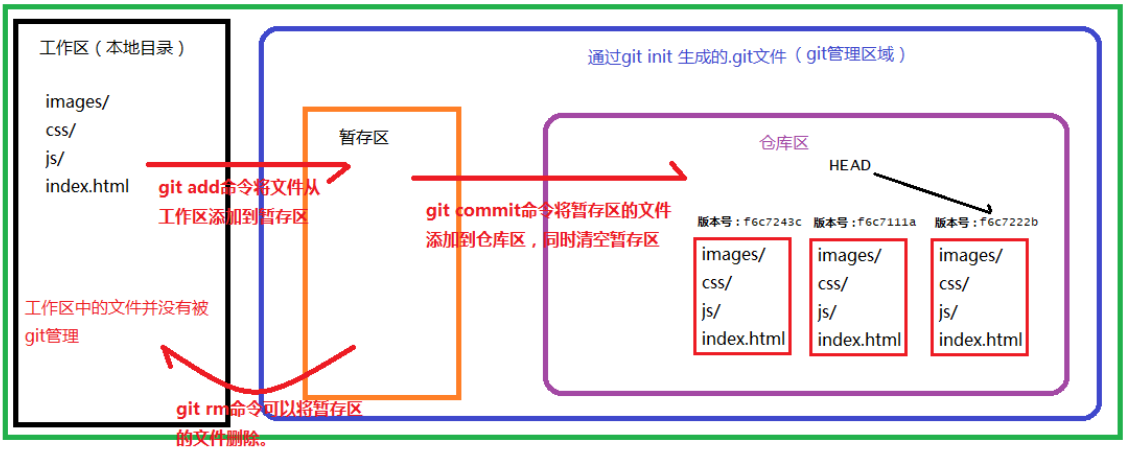

Git是一款免费、开源的分布式版本控制系统， 用于敏捷高效地处理任何或小或大的项目。

- 1. 需要一台服务器作为代码仓库
- 2. 每个用户电脑都是一个服务器（代码仓库），并且和代码仓库是镜像的，用户修改和获取代码都是 提交到自己的服务器当中。
- 3. 不需要网络就可以进行工作。
- 4. 当连接网络时，用户可以选择将自己的服务器与代码仓库进行同步。

## 二、Git 基本操作指南

| 命令 | 解释 |
| --- | --- |
| `git config --global user.name "Your Name"` | 配置全局用户名 |
| `git config --global user.email "your_email@example.com"` | 配置全局用户邮箱 |
| `git init` | 在当前目录创建新的 Git 仓库 |
| `git clone <repository_url>` | 克隆现有仓库 |
| `git add <file_name>` | 添加文件到暂存区 |
| `git add .` | 添加所有文件到暂存区 |
| `git commit -m "描述信息"` | 提交暂存区的更改 |
| `git status` | 查看仓库当前状态 |
| `git log` | 查看提交记录 |
| `git push origin <branch_name>` | 推送本地提交到远程仓库 |
| `git pull origin <branch_name>` | 从远程仓库拉取并合并更改 |
| `git checkout -b <branch_name>` | 创建新分支并切换到新分支 |
| `git checkout <branch_name>` | 切换到已有分支 |
| `git merge <branch_name>` | 将指定分支合并到当前分支 |
| `git branch -d <branch_name>` | 删除本地分支 |
| `git push origin --delete <branch_name>` | 删除远程分支 |
| `git remote add origin <repository_url>` | 添加远程仓库 |
| `git remote -v` | 查看远程仓库 |

## 2.1 Git的工作原理



## 2.2 git命令详解

### 2.2.1. git add(重点)

- 作用：将文件由 工作区 添加到 暂存区，暂存文件
- 命令：
  - git git add 文件名
  - 例如： git add index.html
- git add --all 或者 git add -A 或者git add .（简写） 添加所有文件
- git add a.txt b.txt 同时添加两个文件
- git add *.js 添加当前目录下的所有js文件

### 2.1.2. git checkout 文件名

- 作用：暂存区的内容恢复到工作区。
- git checkout 1.txt 将暂存区中1.txt文件恢复到工作区

### 2.1.3. git commit（重点）

- 作用：将文件由 暂存区 添加到 仓库区
- git commit -m "提交说明"

### 2.1.4. git status

- 作用：查看文件的状态
- 命令：git status
- 命令：git stauts -s 简化日志输出格式

### 2.1.5. git log

- 作用：查看提交日志
- git log 只能查看当前head以及以前的日志
- git log --oneline 简洁的日志信息
- git reflog 查看所有的提交变更日志

### 2.1.6. git reset

- 作用：版本回退，将代码恢复到已经提交的某一个版本中。
- git reset --hard 版本号 将代码回退到某个指定的版本(版本号只要有前几位即可)
- git reset --hard head~1
- 将版本回退到上一次提交
  - ~1:上一次提交
  - ~2:上上次提交
  - ~0:当前提交

## 2.3 git忽视文件

> 在仓库中，有些文件是不想被git管理的，比如数据的配置密码、写代码的一些思路等。git可以通过配置从而达到忽视掉一些文件，这样这些文件就可以不用提交了。

- 在仓库的根目录创建一个.gitignore的文件，文件名是固定的。
- 将不需要被git管理的文件路径添加到.gitignore中

```
```

```
# 忽视idea.txt文件
idea.txt

# 忽视.gitignore文件
.gitignore

# 忽视css下的index.js文件
css/index.js

# 忽视css下的所有的js文件
css/*.js

# 忽视css下的所有文件
css/*.*
# 忽视css文件夹
css
```

# 三、 Git分支操作

## 3.1. 为什么要有分支?

- 如果你要开发一个新的功能，需要2周时间，第一周你只能写50%代码，如果此时立即提交，代码没写完，不完整的代码会影响到别人无法工作。如果等代码写完再提交，代码很容易丢失，风险很大。
- 有了分支，你就可以创建一个属于自己的分支，别人看不到，也不影响别人，你在自己的分支上工作，提交到自己的分支上，等到功能开发完毕，一次性的合并到原来的分支。这样既安全，又不影响他人工作。
- 在工作过程中，经常会碰到多任务并行开发 的情况，使用分支就能很好的避免任务之间的影响。

## 3.2. 分支操作的命令

### 3.2.1. 创建分支

- git branch 分支名称创建分支，分支中的代码，在创建时与当前分支的内容完全相同。
- git在第一次提交时，就有了一个叫master的主分支。

### 3.2.2. 查看分支

- git branch可以查看所有的分支，
- 在当前分支的前面会有一个*

### 3.2.3. 切换分支

- git checkout 分支名称切换分支
- 在当前分支的任何操作，都不会影响到其他的分支，除非进行了分支合并。
- 切换分支之前，必须保证代码已经提交了

### 3.2.4. 创建并切换分支

- git checkout -b 分支名称 创建并切换分支

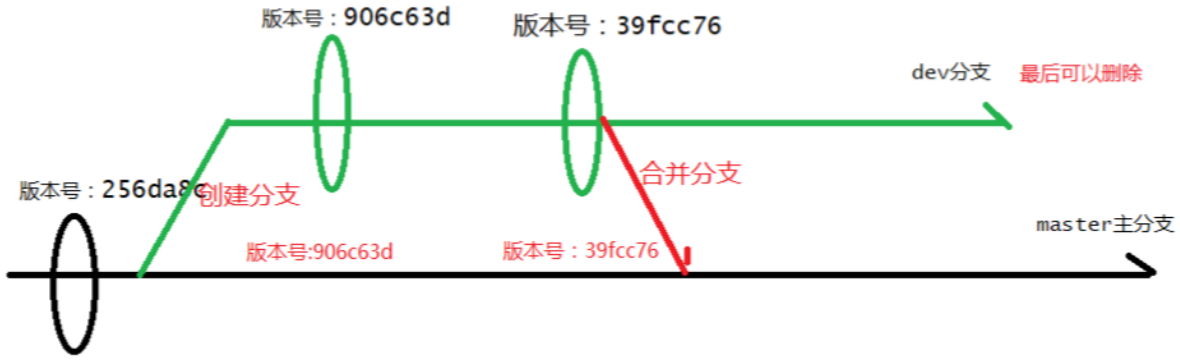
### 3.2.5. 删除分支

- git branch -d 分支名称 可以删除分支
- 注意：不能在当前分支删除当前分支，需要切换到其他分支才能删除。
- 注意：master分支是可以删除的，但是不推荐那么做。

### 3.2.6. 合并分支

- git merge 分支名称 将其他分支的内容合并到当前分支。
- 在master分支中执行git merge dev 将dev分支中的代码合并到master分支

## 3.3. git分支的工作原理




## 3.4. git合并冲突

- 对于同一个文件，如果有多个分支需要合并时，容易出现冲突。
- 合并分支时，如果出现冲突，只能手动处理，再次提交，一般的作法，把自己的代码放到冲突代码的后面即可。

# 四、远程仓库

> 所有的程序员都可以通过远程仓库来进行版本的共享，达到所有人的代码一致的效果。

## 4.1. 远程仓库相关的命令

### 4.1.1. git push

- 作用：将本地代码提交到远程仓库
- git push 仓库地址 master 在代码提交到远程仓库，注意master分支必须写，不能省略
- 例子：git push git@github.com:hucongcong/test.git master 如果第一次使用，需要填写github的用户名和密码

### 4.1.2. git pull

- 作用：将远程的代码下载到本地
- git pull 代码地址 将远程的代码中master分支下载到本地
- 通常在push前，需要先pull一次。

### 4.1.3. git clone

- 作用：克隆远程仓库的代码到本地
- git clone 仓库地址 自定义本地仓库名 将整个仓库克隆到本地

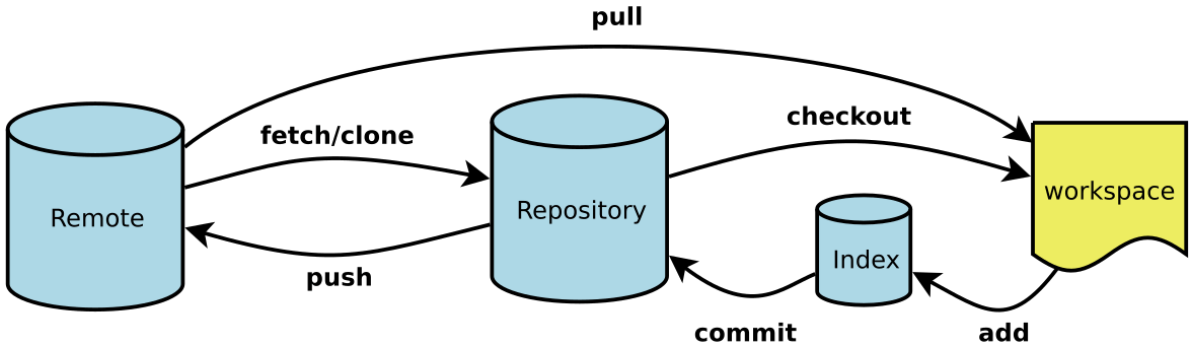
### 4.1.4. git remote

- 每次push和pull操作都需要带上远程仓库的地址，非常的麻烦，我们可以给仓库地址设置一个别名
- git remote add 仓库别名 仓库地址
- 使用仓库别名替代仓库地址。仓库别名相当于一个变量，仓库地址就是对应的值。
- git remote add hucc git@github.com:hucongcong/test.git 设置了一个hucc的仓库别名，以后push和pull都可以不用仓库地址，而用hucc
- git remote remove hucc 删除hucc这个仓库别名。
- git remote 查看所有的仓库别名
- 如果使用了git clone命令从远程仓库获取下来的，那么这个本地仓库会自动添加一个 origin的远程地址，指向的就是克隆的远程地址。

## 4.2. github

- git与github没有直接的关系。
- git是一个版本控制工具。
- github是一个代码托管平台，是git的一个远程代码仓库。
- 将来工作时，公司会有自己的代码仓库。

# 五、附件：命令大全




- Workspace：工作区
- Index / Stage：暂存区
- Repository：仓库区（或本地仓库）
- Remote：远程仓库

## 1、仓库

```
# 在当前目录新建一个Git代码库
$ git init

# 新建一个目录，将其初始化为Git代码库
$ git init [project-name]

# 下载一个项目和它的整个代码历史
$ git clone [url]
```

## 2、配置

```
# 显示当前的Git配置
$ git config --list

# 编辑Git配置文件
$ git config -e [--global]

# 设置提交代码时的用户信息
$ git config [--global] user.name "[name]"
$ git config [--global] user.email "[email address]"
```

## 3、增加/删除文件

```
# 添加指定文件到暂存区
$ git add [file1] [file2] ...

# 添加指定目录到暂存区，包括子目录
$ git add [dir]

# 添加当前目录的所有文件到暂存区
$ git add .

# 添加每个变化前，都会要求确认
# 对于同一个文件的多处变化，可以实现分次提交
$ git add -p

# 删除工作区文件，并且将这次删除放入暂存区
$ git rm [file1] [file2] ...

# 停止追踪指定文件，但该文件会保留在工作区
$ git rm --cached [file]

# 改名文件，并且将这个改名放入暂存区
$ git mv [file-original] [file-renamed]
```

## 4、代码提交

```
# 提交暂存区到仓库区
$ git commit -m [message]

# 提交暂存区的指定文件到仓库区
$ git commit [file1] [file2] ... -m [message]

# 提交工作区自上次commit之后的变化，直接到仓库区
$ git commit -a

# 提交时显示所有diff信息
$ git commit -v

# 使用一次新的commit，替代上一次提交
# 如果代码没有任何新变化，则用来改写上一次commit的提交信息
$ git commit --amend -m [message]

# 重做上一次commit，并包括指定文件的新变化
$ git commit --amend [file1] [file2] ...
```

## 5、分支

```
# 列出所有本地分支
$ git branch

# 列出所有远程分支
$ git branch -r

# 列出所有本地分支和远程分支
$ git branch -a

# 新建一个分支，但依然停留在当前分支
$ git branch [branch-name]

# 新建一个分支，并切换到该分支
$ git checkout -b [branch]

# 新建一个分支，指向指定commit
$ git branch [branch] [commit]

# 新建一个分支，与指定的远程分支建立追踪关系
$ git branch --track [branch] [remote-branch]

# 切换到指定分支，并更新工作区
$ git checkout [branch-name]

# 切换到上一个分支
$ git checkout -

# 建立追踪关系，在现有分支与指定的远程分支之间
$ git branch --set-upstream [branch] [remote-branch]

# 合并指定分支到当前分支
$ git merge [branch]

# 选择一个commit，合并进当前分支
$ git cherry-pick [commit]

# 删除分支
$ git branch -d [branch-name]

# 删除远程分支
$ git push origin --delete [branch-name]
$ git branch -dr [remote/branch]
```

## 6、标签

```
# 列出所有tag
$ git tag

# 新建一个tag在当前commit
$ git tag [tag]

# 新建一个tag在指定commit
$ git tag [tag] [commit]
```

```
# 删除本地tag
$ git tag -d [tag]

# 删除远程tag
$ git push origin :refs/tags/[tagName]

# 查看tag信息
$ git show [tag]

# 提交指定tag
$ git push [remote] [tag]

# 提交所有tag
$ git push [remote] --tags

# 新建一个分支，指向某个tag
$ git checkout -b [branch] [tag]
```

## 7、查看信息

```
# 显示有变更的文件
$ git status

# 显示当前分支的版本历史
$ git log

# 显示commit历史，以及每次commit发生变更的文件
$ git log --stat

# 搜索提交历史，根据关键词
$ git log -S [keyword]

# 显示某个commit之后的所有变动，每个commit占据一行
$ git log [tag] HEAD --pretty=format:%s

# 显示某个commit之后的所有变动，其"提交说明"必须符合搜索条件
$ git log [tag] HEAD --grep feature

# 显示某个文件的版本历史，包括文件改名
$ git log --follow [file]
$ git whatchanged [file]

# 显示指定文件相关的每一次diff
$ git log -p [file]

# 显示过去5次提交
$ git log -5 --pretty --oneline

# 显示所有提交过的用户，按提交次数排序
$ git shortlog -sn

# 显示指定文件是什么人在什么时间修改过
$ git blame [file]

# 显示暂存区和工作区的差异
$ git diff
```

```
# 显示暂存区和上一个commit的差异
$ git diff --cached [file]

# 显示工作区与当前分支最新commit之间的差异
$ git diff HEAD

# 显示两次提交之间的差异
$ git diff [first-branch]...[second-branch]

# 显示今天你写了多少行代码
$ git diff --shortstat "@{0 day ago}"

# 显示某次提交的元数据和内容变化
$ git show [commit]

# 显示某次提交发生变化的文件
$ git show --name-only [commit]

# 显示某次提交时，某个文件的内容
$ git show [commit]:[filename]

# 显示当前分支的最近几次提交
$ git reflog
```

## 8、远程同步

```
# 下载远程仓库的所有变动
$ git fetch [remote]

# 显示所有远程仓库
$ git remote -v

# 显示某个远程仓库的信息
$ git remote show [remote]

# 增加一个新的远程仓库，并命名
$ git remote add [shortname] [url]

# 取回远程仓库的变化，并与本地分支合并
$ git pull [remote] [branch]

# 上传本地指定分支到远程仓库
$ git push [remote] [branch]

# 强行推送当前分支到远程仓库，即使有冲突
$ git push [remote] --force

# 推送所有分支到远程仓库
$ git push [remote] --all
```

## 9、撤销

```
# 恢复暂存区的指定文件到工作区
$ git checkout [file]
```

```
# 恢复某个commit的指定文件到暂存区和工作区
$ git checkout [commit] [file]

# 恢复暂存区的所有文件到工作区
$ git checkout .

# 重置暂存区的指定文件，与上一次commit保持一致，但工作区不变
$ git reset [file]

# 重置暂存区与工作区，与上一次commit保持一致
$ git reset --hard

# 重置当前分支的指针为指定commit，同时重置暂存区，但工作区不变
$ git reset [commit]

# 重置当前分支的HEAD为指定commit，同时重置暂存区和工作区，与指定commit一致
$ git reset --hard [commit]

# 重置当前HEAD为指定commit，但保持暂存区和工作区不变
$ git reset --keep [commit]

# 新建一个commit，用来撤销指定commit
# 后者的所有变化都将被前者抵消，并且应用到当前分支
$ git revert [commit]

# 暂时将未提交的变化移除，稍后再移入
$ git stash
$ git stash pop
```

## 10、其他

```
# 生成一个可供发布的压缩包
$ git archive
```

# 六、参考网址

参考网址

- git大全
  - https://gitee.com/all-about-git
- 深入浅出git教程
  - https://www.cnblogs.com/syp172654682/p/7689328.html
- 阮一峰git教程
  - https://www.liaoxuefeng.com/wiki/896043488029600