git

一、什么是**Git(Git**简介**)**

- 是一个源代码管理工具

- 在一个项目中凡是由开发人员写的代码都叫作源代码

- 源代码有必要管理起来吗？？？？

- 人为的维护比较麻烦

- Git是linux之父当年为了维护linux---linus

- Git之前很多使用SVN vss tfs hs 等等

- Git 是版本管理工具，gitHub是个网站

二、**git**常用指令

1.自报家门

- 配置用户名：

```
git config --global user.name  "你账号名"
```

- 配置邮箱 ：

```
git config --global user.email  "你的邮箱"
```

- 查看配置:

```
git config --list
```

2.初始化一个仓储/仓库

- 通过

```
git init
```

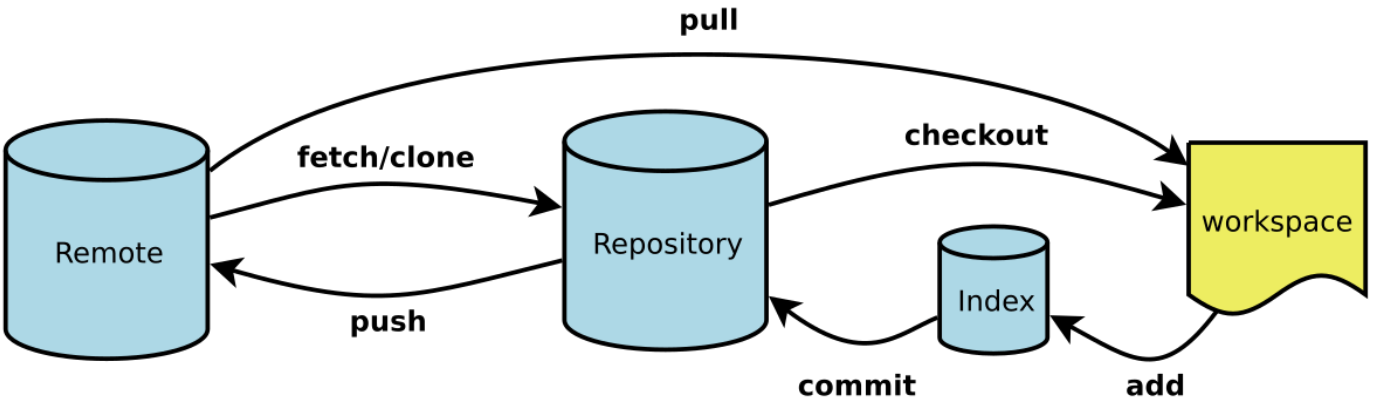
创建一个仓库/仓储

+ 本质创建了一个名为.git的隐藏目录

3.查看当前目录的状态。

4.把大象放到冰箱要几步(小插曲)

- 我们把一个文件保存到仓库时只需要两步



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

5.添加一个文件

```
git add 文件名
```

- 可以使用命令

```
git add .
```

一次性把所有修改过或新添加的文件添加的仓库门口

+ 这里的.表示当前路径

6.提交一个文件

```
git commit -m "我写了一行代码"
```

7.合并操作(不推荐,自己管理自己代码时可以这么做)(了解)

```
git commit -a -m "提交的信息"
```

注意：需要配置user.name 与user.email

8.对比与文件变化

- git diff 用于项目中的代码与add后的代码的区别

9.版本回退

- 命令: git reset --hard Head

+ Head 表示得到最近的一次提交的代码

+ Head~100 最近的第100次提交

- 命令： git reset --hard 提交的版本号

+ 可以回退到指定的版本(根据每次提交的版本的号)

- 命令: git reflog 可以查看每一次的提交的版本号

10.忽略文件

- 新建 .gitingore文件

+ 在这个文件里写一些想要被git忽略的文件或者文件夹

11.创建Git分支

- 1)功能写到一半，相要保存，又不想影响别人的代码,这时我们创建自己的分支去备份代码

+ 命令:

```
git branch 分支名
```

+ branch 还可以列出所有分支：

```
git branch
```

- 在列出所以分支中，当前分支前会有个*号

- 2)创建完分支之后，我们需要切换到新创建的分支中进行代码提交备份

+ 命令:

```
git checkout 分支名
```

+ 做一些提交操作

- 3)当我们在新创建的分支中把功能完成之后，需要把功能给别人用，就需要把功能拿到主分支上去

+ a,切换到主分支:

```
git checkout master
```

+ b,合并分支,把新创建的分支的提交合并到主分支中:

```
git merge 分支名
```

+ 把 指定分支名的分支合并到当前分支

- 4)最后，合并分支之后，我们可以清除之前临时使用的分支

+ 命令:

```
 git branch -d 分支名 // 删除指定的分支
```

- 可以合并1、2步的操作:

+ 创建并切换分支

```
git checkout -b dev
```

12合并Git分支

1)解决冲突

- 我们在不同的分支中对同一文件，同一行代码(只要是使用后面的往发生改变),git就无法

解决这种类型的冲突，因为git只是个工具，它不能够判断对我们来说，哪一部分代码是最有效.

- 这时就需要我们手动的去解决冲突(其实就是在文件中有选择的进行删减);

2)工作区与暂存区

- 工作区指的是我们写代码的目录(不包括.git隐藏目录)
- 暂存区是我们执行`git add xxx` 之后文件存放的地方

13GitHub

Git与GitHub的关系

- Git版本管理工具,
- GitHub就是一个网站

14推送(push)与拉取(pull)

Git 常用命令速查表

```
master :默认开发分支        Head  :默认开发分支
origin :默认远程版本库      Head^ :Head 的父提交
```

创建版本库

```
$ git clone <url>              #克隆远程版本库
$ git init                     #初始化本地版本库
```

修改和提交

```
$ git status                   #查看状态
$ git diff                     #查看变更内容
$ git add .                    #跟踪所有改动过的文件
$ git add <file>               #跟踪指定的文件
$ git mv <old> <new>           #文件改名
$ git rm <file>                #删除文件
$ git rm --cached <file>       #停止跟踪文件但不删除
$ git commit -m “commit message”
                               #提交所有更新过的文件
$ git commit --amend           #修改最后一次提交
```

查看提交历史

```
$ git log                      #查看提交历史
$ git log -p <file>            #查看指定文件的提交历史
$ git blame <file>             #以列表方式查看指定文件
                               的提交历史
```

撤消

```
$ git reset --hard HEAD        #撤消工作目录中所有未提交
                               文件的修改内容
$ git checkout HEAD <file>     #撤消指定的未提交文件的修
                               改内容
$ git revert <commit>          #撤消指定的提交
```

分支与标签

```
$ git branch                   #显示所有本地分支
$ git checkout <branch/tag>    #切换到指定分支或标签
$ git branch <new-branch>      #创建新分支
$ git branch -d <branch>       #删除本地分支
$ git tag                      #列出所有本地标签
$ git tag <tagname>            #基于最新提交创建标签
$ git tag -d <tagname>         #删除标签
```

合并与衍合

```
$ git merge <branch>           #合并指定分支到当前分支
$ git rebase <branch>          #衍合指定分支到当前分支
```

远程操作

```
$ git remote -v                #查看远程版本库信息
$ git remote show <remote>     #查看指定远程版本库信息
$ git remote add <remote> <url>
                               #添加远程版本库
$ git fetch <remote>           #从远程库获取代码
$ git pull <remote> <branch>   #下载代码及快速合并
$ git push <remote> <branch>   #上传代码及快速合并
$ git push <remote> :<branch/tag-name>
                               #删除远程分支或标签
$ git push --tags              #上传所有标签
```