Git与GitHub的故事

简介

```
1. Git是目前世界上最先进的分布式版本控制系统(没有之一)
2. Git免费、开源，可有效、高效的处理或小或大的项目版本管理
3. 版本控制系统：是指能够监控文件的改变做出历史记录，并能进行相关操作的系统
4. 分布式开发，每个独立的个体都可以成为服务器，允许离线工作
```

Git的下载与安装

1. Git的官网 Git走起
2. 下载对应系统，对应版本的Git安装包
3. 傻瓜式安装，无需动脑

Git的基本使用

创建版本库

> 概念：版本库其实就是一个目录，工作的目录，该目录下所有文件被Git管理

创建仓库与初始化

```
// 创建目录
mkdir   gitdemo

// 切换目录
cd gitdemo

// 初始化git创库,初始化完毕后会产生一个.git目录(版本库，该目录用于记录追踪信息)
git init
```

添加文件到版本库

```
// 创建README.md文件
vim README.md(介绍文档)

// 添加文件到暂存区
git add README.md

// 将暂存区的内容提交给版本库
git commit -m "这是提交的注释"
```

> 注：当前若未设置git的用户和邮箱，则要进行设置

```
git config --global user.name "huxiaoshuai"
git config --global user.email "justbecoder@aliyun.com"
```

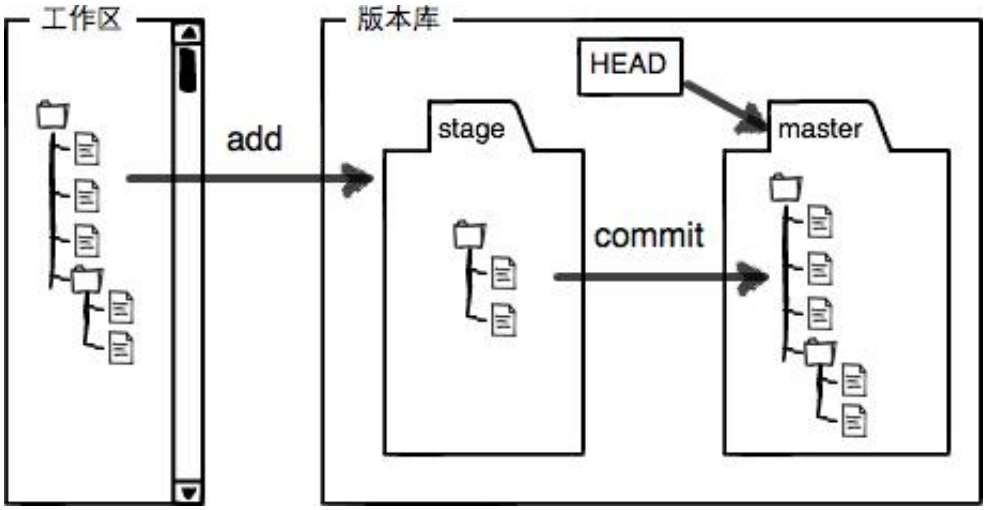
版本穿梭

```
// 查看当前仓库的状态
git status

// 查看当前文件修改
git diff README.md
```

版本回退

```
// 查看记录 --pretty=oneline 格式化美化查询
git log

// 查看历史记录命令
git reflog

// HEAD 指向当前版本

// 版本回退
git reset --hard HEAD^1(上一个版本)
git reset --hard HEAD~n (上n个版本)

// 跳转到指定版本
git reset --hard commit_id(指定版本产生的comiit_id)
```

工作区和暂存区



管理修改

```
1. 任何一个修改只有add暂存区，才能被commit到版本库
2. git diff HEAD -- README.md 查看当前工作区与版本库的文件区别
```

撤销修改

```
1. git checkout -- file 丢弃文件在工作区的修改，从版本库还原
    a. 文本被修改，没有添加到暂存区
    b. 文件添加到暂存区后再次被修改

2. gti reset HEAD file 将暂存区清除退回到工作区
```

删除文件

```
误删：
    rm file 在工作区删除文件
    使用 git checkout -- file 从版本库恢复文件到工作区
真删：
    使用 git rm file 删除工作区和暂存区的文件
    提交数据：git commit 将文件从版本库删除
```

远程仓库

> 1. GitHub 一个神奇的网站,提供Git仓库托管服务 走起
> 2. 自行注册GitHub账户

SSH加密传输

> 本地Git仓库与远程GitHub仓库使用SSH加密传输

创建本地SSH Key

```
在用户主目录生成.ssh目录 id_rsa 私钥 id_rsa.pub 公钥
ssh-keygen -t rsa -C "youremail@example.com"
```

添加SSH Key到GitHub账户(用于识别你就是你)

```
Account Settings --> SSH Keys --> Add SSH Key
```

创建仓库

```
在GitHub上手动创建repository
```

推送项目

```
// 与远程仓库创建连接
git remote add origin git@github.com:justbecoder/memeda.git

// 查看连接
git remote -v

// 删除连接
git remote rm origin

// 推送项目 -- 将当前分支推送到仓库的master分支
git push origin master
```

拉取项目与克隆项目

```
    // 当与仓库建立连接后,拉取master分支
    git pull origin master

    // 从指定的仓库克隆项目
    git clone git@github.com:justbecoder/memeda.git
```

分支管理

创建与合并分支

```
// 创建分支
git branch dev

// 切换分支
git checkout dev

// 快速创建、切换分支
git checkout -b dev

// 合并分支，切换到对应的分支
git merge dev  将dev分支与当前分支合并
```

解决冲突

```
// 查看冲突
git diff file

// 查看历史操作 --pretty=oneline 格式化查询
git log

// 图形化显示历史操作，分支信息
git log --graph
```

删除分支

```
git branch -d dev
```

标签管理

> 目的：当发布版本时，打上标签，方便操作

添加标签

```
// 给当前版本添加标签
git tag v1.0

// 给历史版本添加标签
git tag v0.1 commit_id
```

管理标签

```
// 推送本地标签到远程仓库
git push origin v.10

// 一次性推送全部标签
git push origin --tags

删除标签
git tag -d v0.1

// 远程删除标签
git push origin :refs/tags/v0.9
```

Git配置

忽略指定文件

```
// 在仓库目录下创建.gitignore文件
vim .gitignore

// .gitignore书写规则
1. 将不需要提交给git版本库的目录文件写入
2. 使用换行进行区分
```

配置别名

```
// 设置别名 方便操作
git config --global alias.st status
```