一、简介

- Git 是一个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。

下载配置 Git

配置用户名和邮箱

```
# 配置用户名
$ git config --global user.name <用户名>

# 配置用户邮箱(不需真实存在)
$ git config --global user.email <用户邮箱>
```

查看配置信息

```
# 查已有的配置信息
$ git config --list
```

git 使用

- 使用Git前，需要先建立一个仓库(repository)。可以使用一个已经存在的目录作为Git仓库或创建一个空目录。
- 对 Git 仓库进行初始化

```
# 对当前目录初始化
git init

# 初始化指定 Git 仓库
git init <仓库名>
```

- 在仓库中添加新文件

```
git add <filename>

# 把当前文件夹内所有文件和非空文件夹添加到index
$ git add .
```

  - 可以使用add... 继续添加任务文件
- 提交版本文件到仓库

```
git commit -m "Adding files"
```

  - 不使用-m，会出现编辑器来让你写自己的注释信息（window双引号，linux单引号）

- 当修改了很多文件，而不想每一个都add，想commit自动来提交本地修改，我们可以使用-a标识

```
git commit -a -m "Changed some files"
```

  - git commit 命令的-a选项可将所有**被修改或者已删除的且已经被git管理的文档**提交到仓库中
  - -a不会造成新文件被提交，只能修改。

- 发布版本

```
# 从服务器克隆一个库并上传
git clone ssh://example.com/~/www/project.git

# 修改之后可以进行推送到服务器
git push ssh://example.com/~/www/project.git

```

- 取回更新

```
# 当前分支自动与唯一一个追踪分支进行合并
git pull

# 从非默认位置更新到指定的url
git pull http://git.example.com/project.git
```

- 删除

```
# 从资源库中删除文件
git rm file
```

- 分支和合并

```
# 创建一个新的分支
git branch test

# branch命令不会将我们带入分支
# 使用checkout命令来更改分支
git checkout test

# 使用checkout命令来更改到主分支（master）
git checkout master

# 对其他分支的更改不会反映在主分支上。
# 如果想将更改提交到主分支，则需切换回master分支，然后使用合并。
git checkout master
git merge test

# 删除分支
git branch -d test
```

- git checkout 命令主要有两大用途——切换分支和恢复文件。

Github https://www.runoob.com/w3cnote/git-guide.html

1 - 简介

- github是一个基于git的代码托管平台，付费用户可以建私人仓库，我们一般的免费用户只能使用公共仓库（代码公开）

2 - Github使用

2.1 - 注册 github 账号

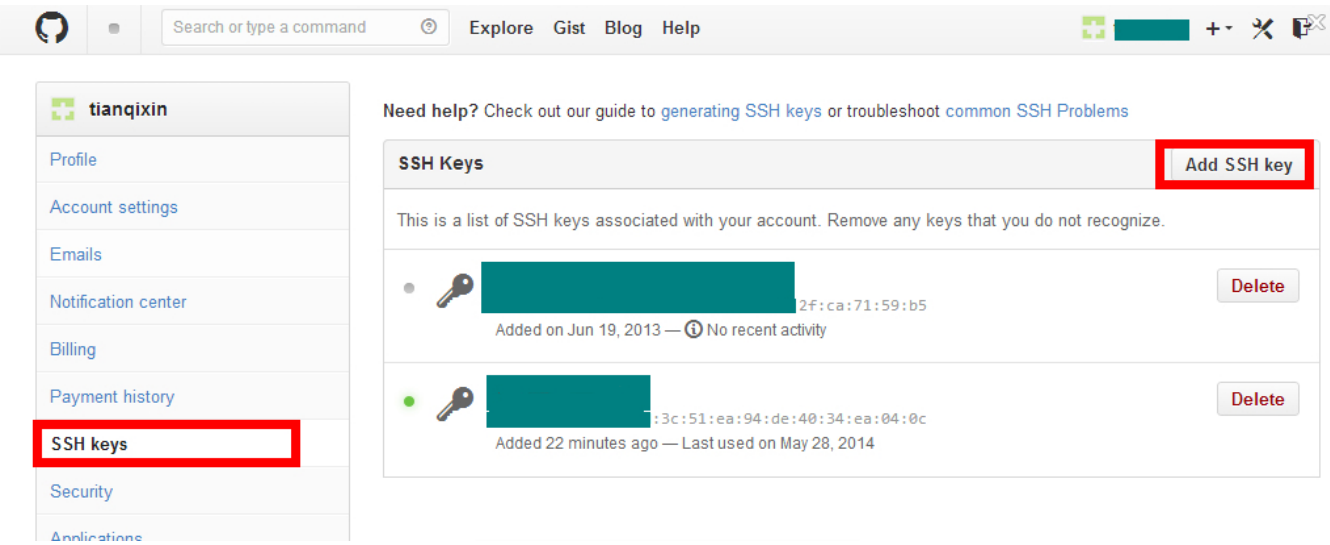
2.2 - 创建仓库 Create a New Repository

2.3 - 配置 Git

- 本地创建 ssh key (使用默认的一路回车)

```
# your_email@youremail.com为 github 上注册的邮箱
$ ssh-keygen -t rsa -C "your_email@youremail.com"
```

- 进入 `~/` 下生成 `.ssh` 文件夹，打开 `id_rsa.pub`，复制里面的 `key`
- 返回 github上，进入 Account Settings（账户配置），左边选择SSH Keys，Add SSH Key,title随便填，粘贴在你电脑上生成的key



- 验证是否成功，git bash 输入

```
$ ssh -T git@github.com
```

  - 第一次提示是否 continue，输入 yes，看见 `You've successfully authenticated, but GitHub does not provide shell access` 表示成功连上 github
- 上传本地仓库至 github （需设置 username 和 email）

```
# 配置用户名
$ git config --global user.name <用户名>

# 配置用户邮箱(不需真实存在)
$ git config --global user.email <用户邮箱>
```

- 进入要上传的仓库， git bash 中输入

```
# 添加远程地址 ourName和yourRepo表示你再github的用户名和刚才新建的仓库
$ git remote add origin git@github.com:yourName/yourRepo.git
```

- 加完之后进入.git，打开config，这里会多出一个remote "origin"内容，这就是刚才添加的远程地址，也可以直接修改config来配置远程地址。
- 创建新文件夹，打开，然后执行`git init`以创建新的 git 仓库。

2.4 - 检出仓库

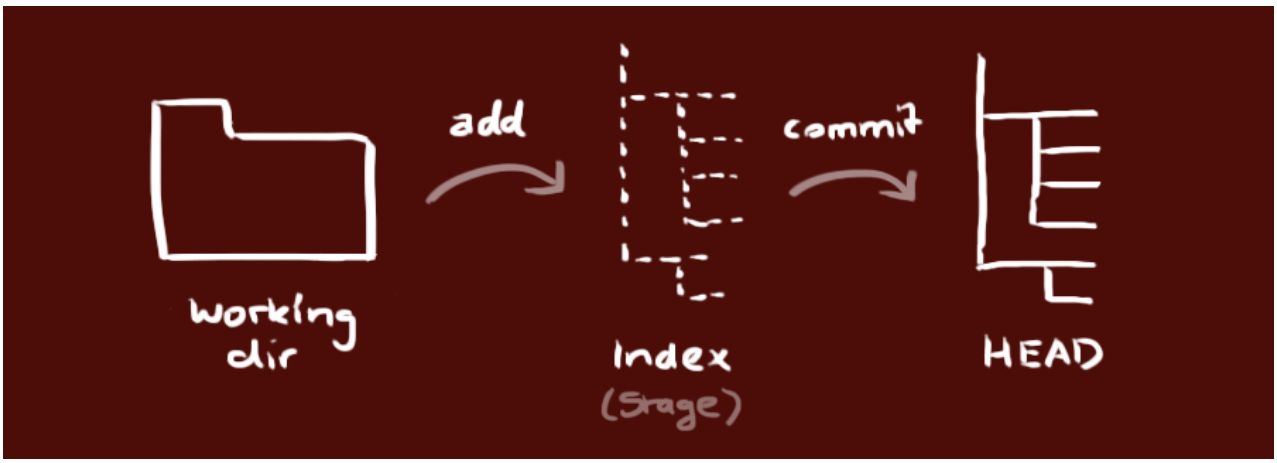
```
# 创建一个本地仓库的克隆版本
$ git clone /path/to/repository

# 创建远端服务器上的仓库的克隆版本
$ git clone username@host:/path/to/repository
```

2.5 - 工作流

你的本地仓库由 git 维护的三棵"树"组成。

- 工作目录
- 暂存区（Index）—— 像个缓存区域，临时保存你的改动
- HEAD —— 指向"current branch"(当下的分支)你最后一次提交的结果



step1: 提出更改，将文件添加到暂存区

```
$ git add <filename>
$ git add .
```

step2: 实际提交改动

```
# 将改动提交到了 HEAD （但还没到你的远端仓库）
$ git commit -m "代码提交信息"
```

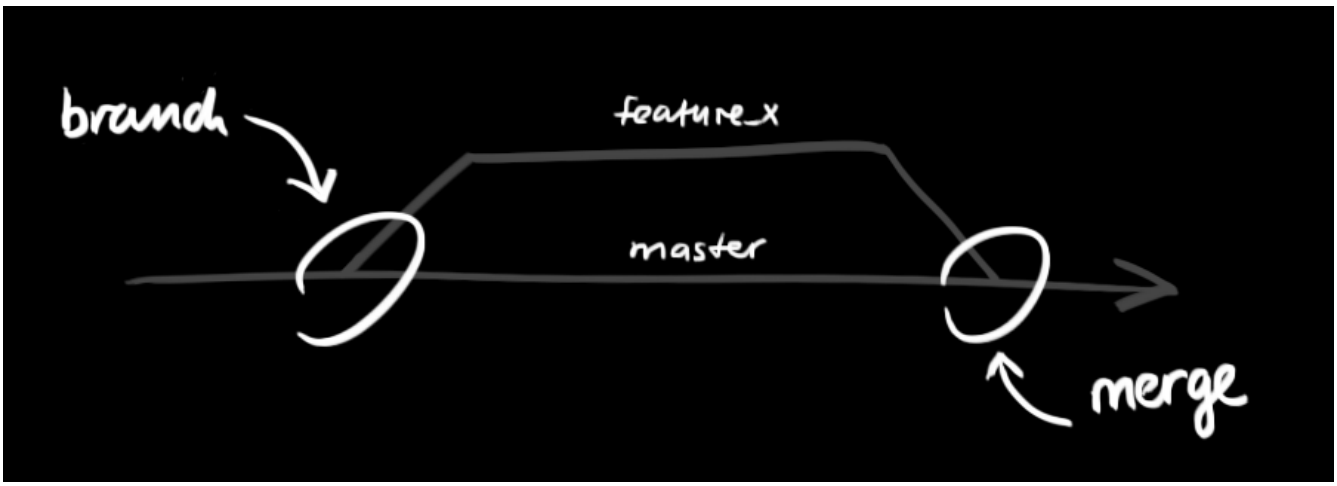
step3: 将改动提交到远端仓库

```
# 可以把 master 换成你想要推送的任何分支
$ git push origin master

# 将改动推送到所添加的服务器上
$ git remote add origin <server>
```

2.6 - 分支

- 分支是用来将特性开发绝缘开来的。
- 在你创建仓库的时候，*master* 是"默认的"分支。
- 在其他分支上进行开发，完成后再将它们合并到主分支上。




```
# 创建新分支，并切换至该分支
$ git checkout -b feature_x

# 删除分支
$ git branch -d feature_x

# 切换回主分支
$ git checkout master

# 将分支推送到远端仓库（公开代码，为他人可见）
git push origin <branch>
```

2.7 - 更新与合并

```
# 更新你的本地仓库至最新改动,在你的工作目录中 获取（fetch） 并 合并（merge） 远端的改动
$ git pull

# 合并其他分支到你的当前分支（例如 master）
$ git merge <branch>
```

- 在这两种情况下，git 都会尝试去自动合并改动。遗憾的是，这可能并非每次都成功，并可能出现*冲突（conflicts）*。
- 出线冲突需要你修改这些文件来手动合并这些*冲突（conflicts）*
- 改完之后，需要执行如下命令以将它们标记为合并成功

```
$ git add <filename>

# 预览差异
$ git diff <source_branch> <target_branch>
```

2.8 - 标签

```
# 为软件发布创建标签
# 1b2e1d63ff 是你想要标记的提交 ID 的前 10 位字符
$ git tag 1.0.0 1b2e1d63ff

#获取提交id
$ git log
```

2.9 - 替换本地改动

```
$ git checkout -- <filename>
```

- 操作失误时可使用
- 使用 HEAD 中的最新内容替换掉你的工作目录中的文件
- 已添加到暂存区的改动以及新文件都不会受到影响。

```
# 丢弃你在本地的所有改动与提交，可以到服务器上获取最新的版本历史，并将你本地主分支指向它
$ git fetch origin
$ git reset --hard origin/master
```

2.10 - 其他使用命令

```
# 内建的图形化 git
$ gitk

# 彩色的 git 输出
$ git config color.ui true

# 显示历史记录时，每个提交的信息只显示一行
$ git config format.pretty oneline

# 交互式添加文件到暂存区
$ git add -i
```

常用命令

git 使用 linux 命令

```
# 新建文件夹
$ mkdir <fileName>

# 新建文件
$ touch <fileName>
```