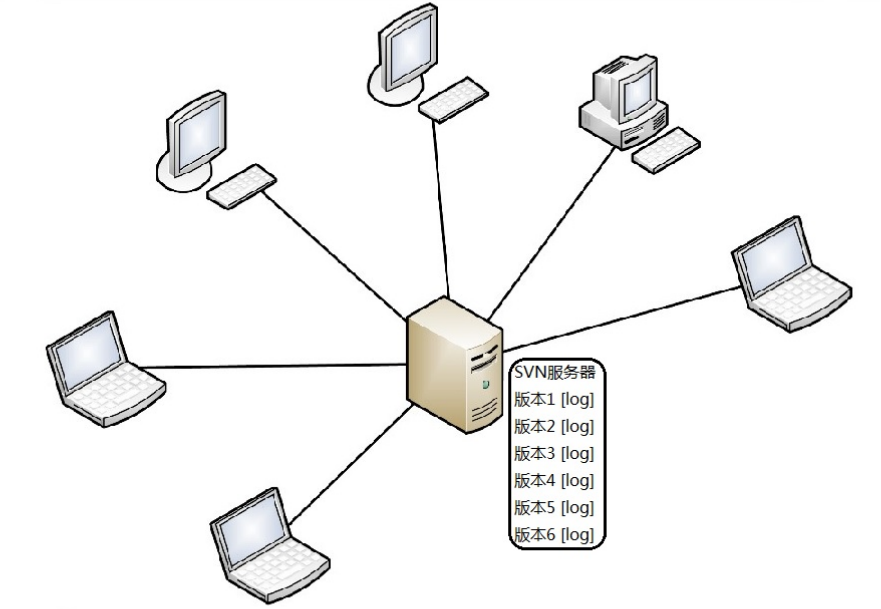
# Git 简易教程

## git和svn的区别

svn是集中式的版本控制系统

git是分布式的版本控制系统

集中式：



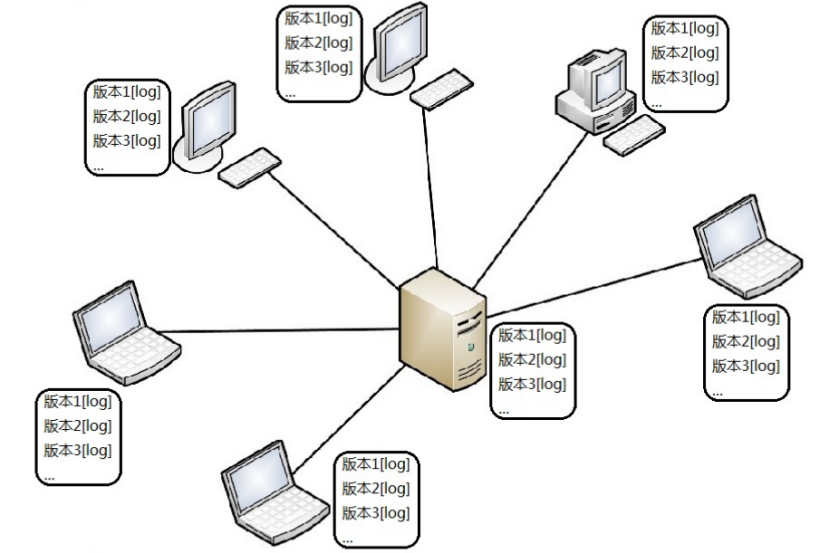
可以看出svn集中式版本控制都是把所有的版本存储在svn服务器中

且必须要联网到svn服务器才可以进行版本的回退、更新等操作。

假如哪天svn服务器坏了，那么所有的版本代码都将丢失，也就无法获取代码和回退版本等操作。天啊，这将是一个悲伤的故事！

如何化悲伤为快乐呢，来看看git如何工作的把！

分布式：



可以看出，使用git,每个电脑都有完整的版本号和日志信息。

且没有网的时候，git照样可以工作，只是把代码提交到本地，待有网在提交到远程git仓库

就算哪天git服务器坏了也没事，因为本地仓库都保存了完整的版本。

搭建github服务器： <http://blog.w0824.com/2018/07/09/Centos%E6%90%AD%E5%BB%BAGIT%E6%9C%8D%E5%8A%A1%E5%99%A8/>

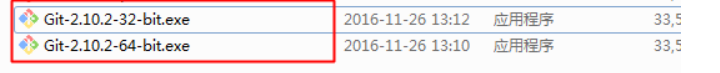
免费第三方平台：github、码云

git发明者：是Linux创始人李纳斯

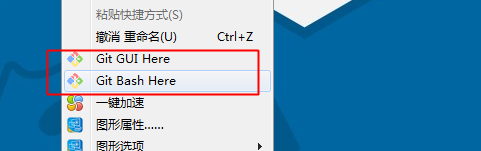
## 安装git工具

git是linux之父李纳斯发明的。

安装注意自己系统的位数。



安装的时候选择安装路径即可，然后一路next即可。安装好后鼠标右键会多出以下两个选项，代表git工具安装完成。



## 获取git仓库

获取git仓库有两种方式，

一是在本地目录中执行git init指令，初始化一个仓库。

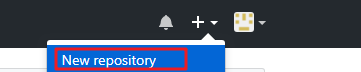
二是从远程服务器拉取一个仓库。如从github拉取，或是从自己搭建的git服务器拉取。

这里重点讲解第二种方式如何在github上面创建仓库，第一种方式较简单。

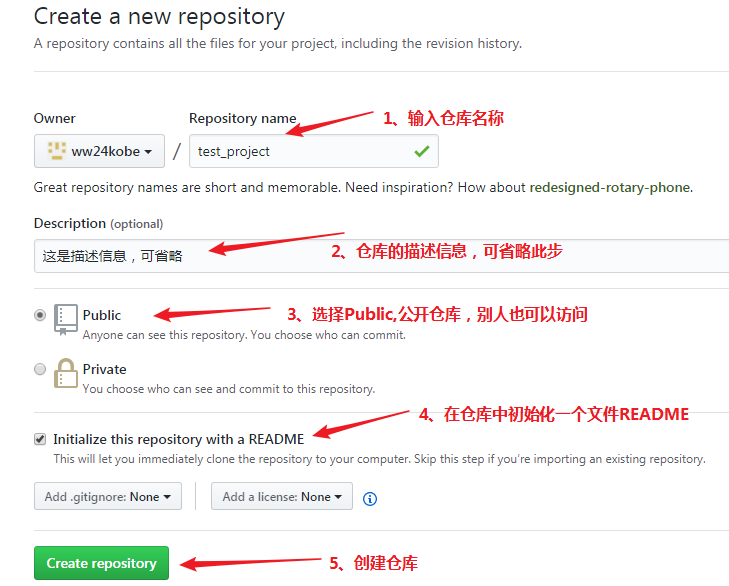
### （1）在github创建仓库

创建仓库比较简单，参考下面的图片即可：

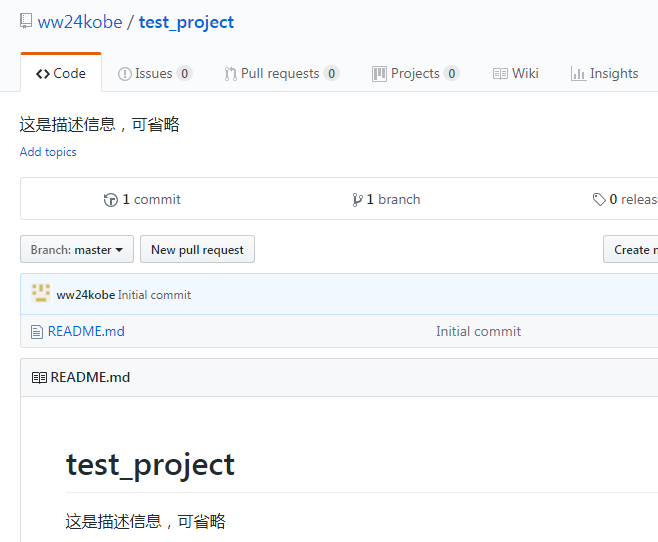
1、点击加号+，选择New repository新建仓库



2、输入创建仓库的信息



创建好之后，如下所示：



## 克隆远程仓库代码到本地

克隆远程的仓库代码到指定目录:

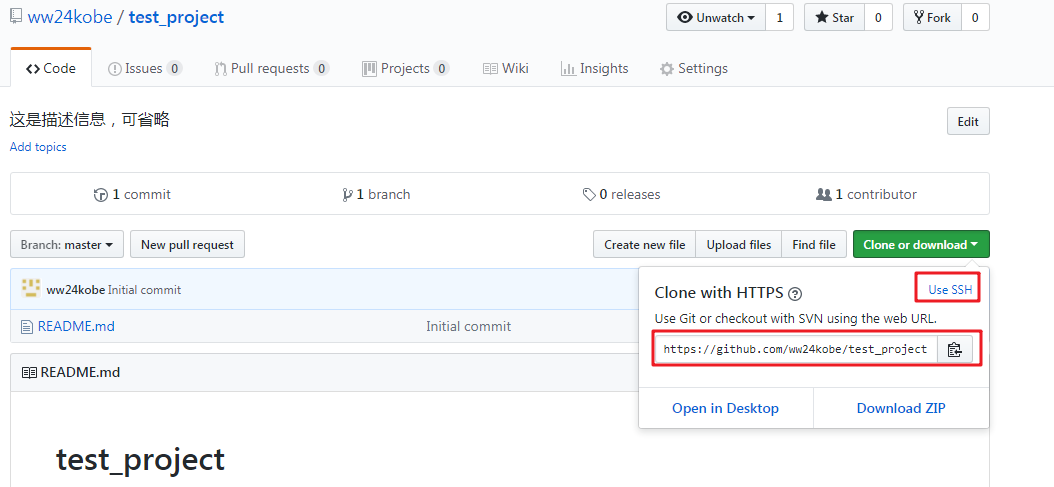
命令：git clone 仓库地址 [目录]

注意：不写目录名称会在当前目录创建一个与github仓库同名的目录

如下面的指令代表把仓库代码检出到当前目录(./)

git clone 仓库地址 ./

仓库地址位于：



仓库地址有两种协议：https、ssh。后面使用ssh协议可以免去每次推送代码输入密码的烦恼。

完整命令：git clone <https://github.com/ww24kobe/test_project.git> ./ ，成功之后，会把目录中多一个.git的隐藏文件夹。

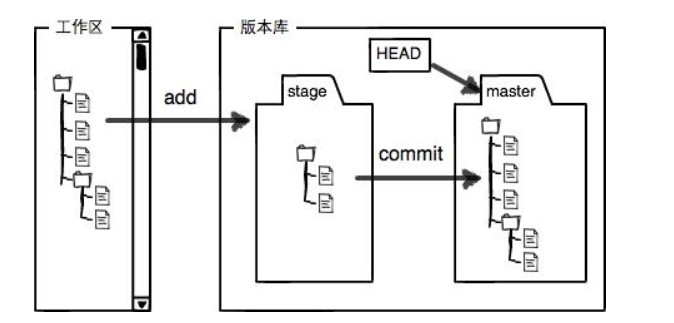
## git工作流

一个git仓库包含以下三个部分：

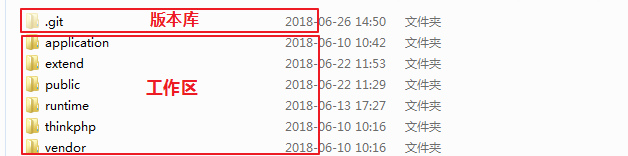
工作区：就是我们电脑里能看到的目录。

暂存区：英文名叫stage,或index。一般存放在“.git目录”的index文件中。

版本库：工作区有一个隐藏目录.git,这个不算工作区，而是git的版本库。且自动创建一个 master 分支以及指向分支的 HEAD 指针



如下：



后面众多的git命令，就是在以上三者中互相交叉使用。

## 开发中git常用的指令

* 全局设置：先要设置提交的用户名和邮箱，不设置则无法提交代码。以后可以找到代码的负责人

git config --global user.name 名字 # 叫啥名字

git config --global user.email 邮箱 # 怎么联系你

去掉--global则只在当前项目中有效

git config user.name 名字 # 叫啥名字

git config user.email 邮箱 # 怎么联系你

* 查看配置信息

git config --list , 查看命令如何使用，如git commit --help

* 在指定目录创建一个git仓库

git init :执行完后会在当前目录生成一个.git的隐藏文件夹

如果后期需要把本地仓库代码推送到远程仓库，则需要设置远程仓库地址。

git remote add origin url # 设置本地的远程仓库地址

* 克隆远程的仓库代码到指定目录:

git clone url [目录]

注意：不写目录名称会在当前目录创建一个与github仓库同名的目录

如下面的指令代表把指定url仓库代码检出到当前目录(./)

git clone url ./

* 添加当前目录的所有文件到暂存区:

git add .

* 查看暂存区状态:

git status

* 提交文件:

git commit -m ‘备注信息’

* 查看提交备注的信息（查看提交记录）

git log 或者 git reflog 或 git log --oneline

更酷的显示方式：git log --oneline --graph

--graph图形化显示，比较直观。

更牛逼的显示：给log命令设置别名,

git config --global alias.lg “log --graph --pretty=format:'%Cred%h%Creset -%C(yellow)%d%Creset %s %Cgreen(%cr)%Creset' --abbrev-commit --date=relative”

* 对于已经在仓库中的文件若是做了修改，可以将add和commit一起执行

git commit -am ‘备注信息...’

* 代码版本回退:



git reset --hard HEAD 回到当前的版本

git reset --hard HEAD^ 回到当前的版本前一个版本

git reset --hard HEAD^^ 回到当前的版本前两个版本

git reset --hard af4542g(使用git log 获取日志的前7位,可以回到指定的版本)

* 删除文件

git rm files

* 撤掉修改

git checkout files

* 推送代码到远程服务器:

git push -u origin master

* 修改本地远程仓库地址：

git remote add origin url # 设置本地的远程仓库地址

git remote rm origin # 移除本地远程仓库地址

* 从远程服务器获取内容:

git pull orgin master 拉取远程仓库代码并合并

git fetch orgin master 拉取远程仓库代码不会合并，需要执行git merge origin/merge进行合并

远程代码强制合并本地代码：

git pull origin master --allow-unrelated-histories

* 仓库地址

git remote -v #查看本地的远程仓库路径

git remote rm origin #移除本地远程仓库地址

git remote add origin git@github.com:用户名/仓库名.git #设置本地的远程仓库地址

* 查看文件的每行代码是谁写的，尤其实现发现了错误代码的情况下，想跑都没门。

git blame files

## 给ssh协议创建私钥和公钥

如果仓库地址使用https的协议，每次提交都会要求输入远程仓库github的用户名和密码，

如果我们使用ssh协议作为仓库地址的话，并且配置好私钥和公钥，每次提交就会免去输入用户名和密码的烦恼。

公钥：理解为锁,上传到github中存放着。

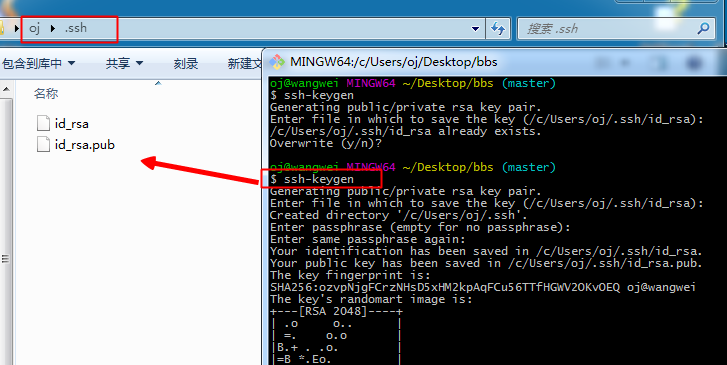
私钥：理解为锁的钥匙，在本地电脑存放着。

也就是说只有锁的对应钥匙才可以进行提交代码。

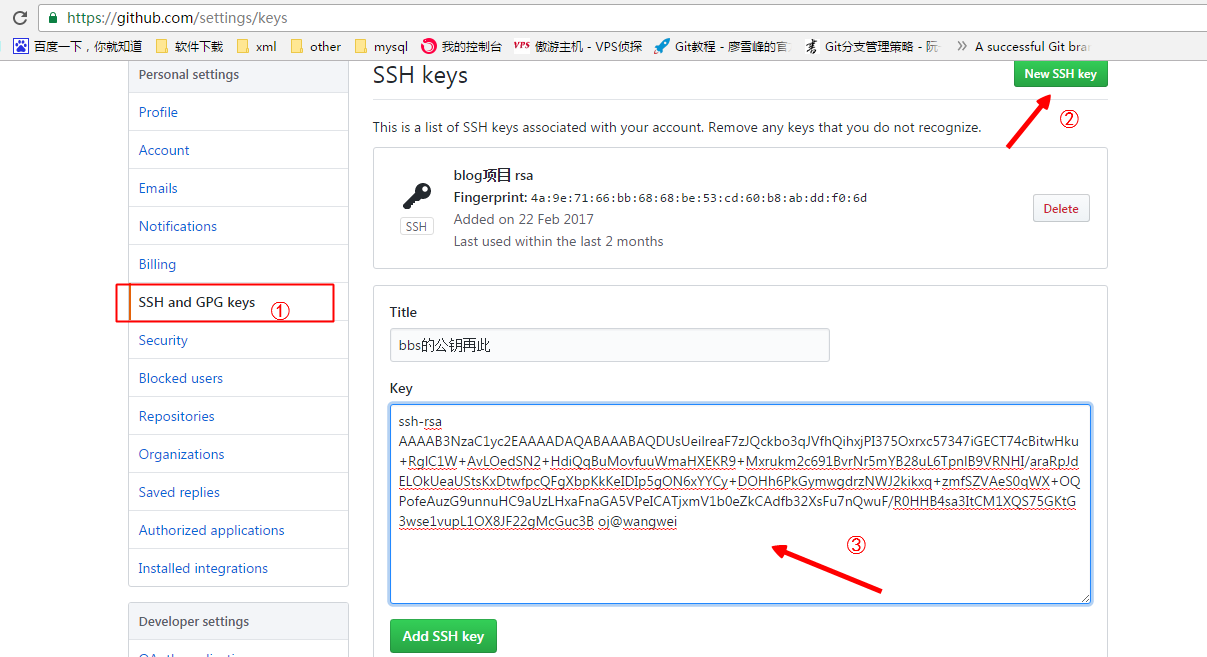
创建ssh私钥和公钥，输入: ssh-keygen -t rsa -C（大写C哦） '邮箱地址' ，然后一路回车即可，成功之后会在当前用户的目录多出如下的两个文件。

id.rsa:私钥文件

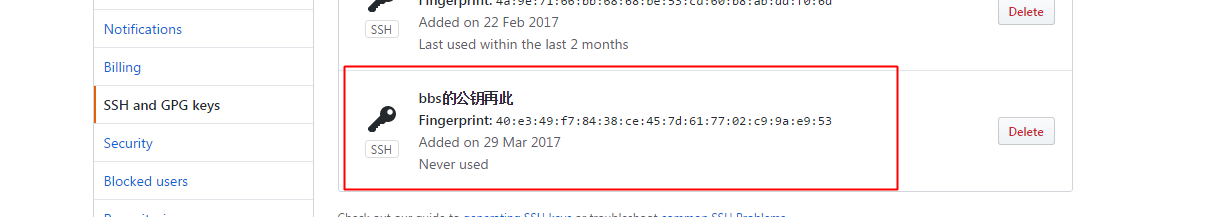
id\_rsa.pub:公钥文件



把id\_rsa.pub的公钥内容复制到github上面去，步骤如下：



添加好后如下所示：



最后修改远程仓库地址为ssh协议即可：

git remote -v #查看本地的远程仓库路径

git remote rm origin #移除本地远程仓库地址

git remote add origin git@github.com:用户名/仓库名.git #设置本地的远程仓库地址

## git分支

master分支：每个仓库默认有的分支，主要用来发布代码新版本。

平常代码的开发在创建的开发分支 dev 上进行。

开发分支dev开发完毕后，把此分支的代码合并到master 分支，最后再把master分支下的代码推送到远程服务器（即远程仓库github）

分支有关的指令：

* 查看仓库所有的分支:

git branch

* 创建dev分支:

git branch dev

* 切换分支(切换到master分支):

git checkout master

* 合并分支dev到master主分支

先切换到要合并的分支，再把dev分支合并到当前分支

git checkout master

git merge --no-f dev -m ‘合并的信息’

注：通过选项--no-f合并也算一次提交

* 删除分支dev:

git branch -d dev

* 如果分支还未合并，可以强制删除:

git branch -D dev

* 提交分支dev到远程

git push origin dev

* 删除远程的dev分支

git push --delete dev

有关git分支的管理策略文档：



## 创建标签（版本号）

* 查看所有标签

git tag

* 创建标签，版本号为1.0， -m 备注信息

git tag v1.0 -m ‘version 1.0’

* 推送本地的所有标签到远程仓库，其他人克隆此仓库或拉取数据同步后，也会看到这些标签。

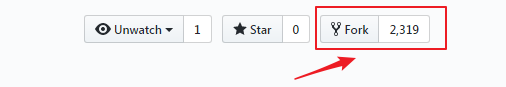
git push origin master --tags

## 搭建git服务器

参考简书地址：<https://www.jianshu.com/p/e79ea05d9b61>

## fork

可以把别人的项目fork（理解为复制）到自己的用户名下面



## issue

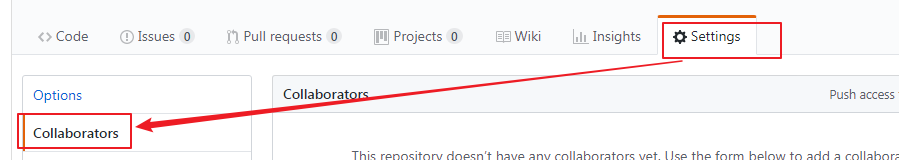
给项目提出意见和建议，包括项目bug，一起把项目做得更好，一起进步学习。

## full request

当你发觉fork下来的项目有bug时，你可以进行修改提交到自己仓库，在发起一个pull request即可，项目的原作者会看到这个pull request请求，如果没问题，会把此请求合并merge到项目中。

## contributor

给自己项目开发增加协助者，共同完成这个项目。点击自己项目的Settings



添加成功，通过contributor可查看。

