# 为什么要使用代码版本控制软件

作用：

* 代码备份
* 团队协作开发
* 代码回滚（回退）

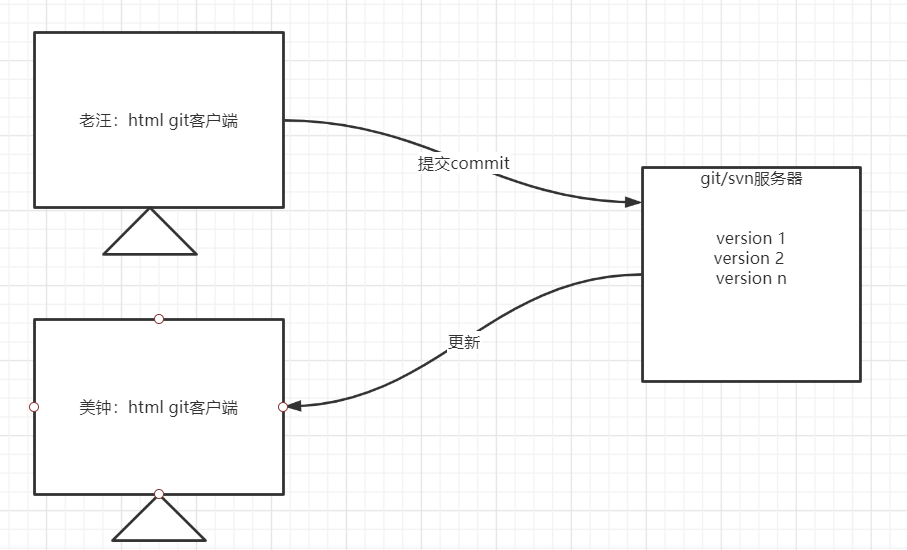
常见的代码版本控制软件有csv（淘汰）、svn（使用简单）、git （功能非常强大）

代码版本工作模式：

Svn/git是c/s架构的软件：c=>client s=>server

C:client客户端，自己需要安装

S:server服务端，可以自己搭建git服务器，或使用第三方的如github或码云，以后在大型公司中，企业一般都会搭建自己私有的代码托管服务器。

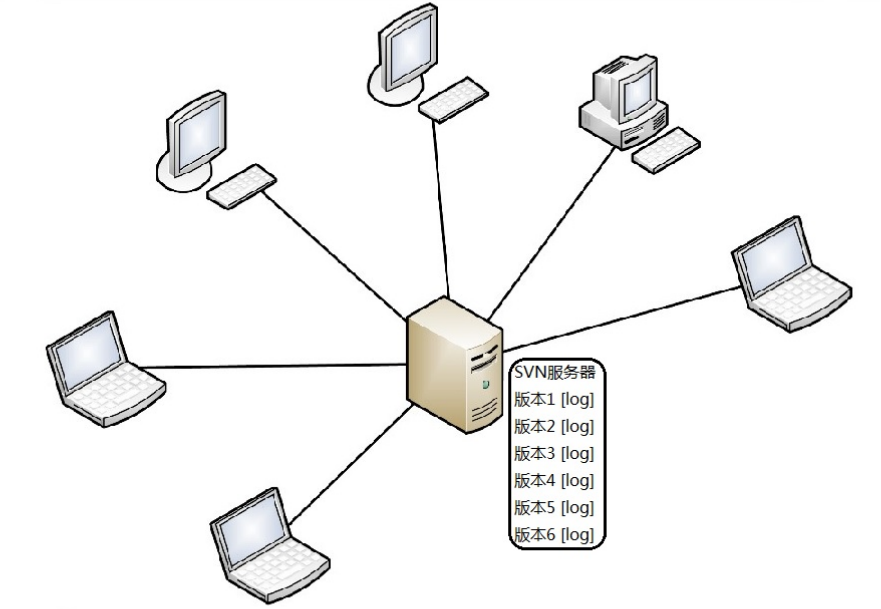


# git和svn的区别

svn是集中式的版本控制系统

git是分布式的版本控制系统

svn集中式：



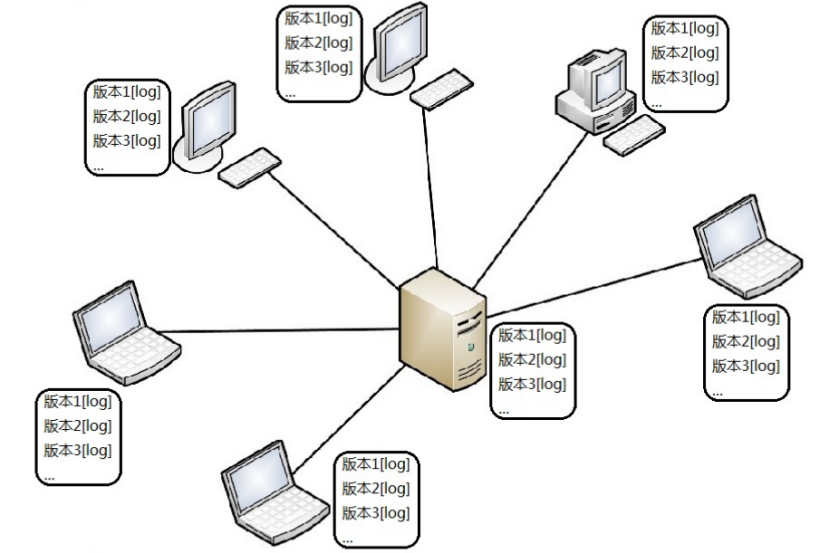
可以看出svn集中式版本控制都是把所有的版本存储在svn服务器中

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！

Git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。

Centos搭建私有git服务器： http://blog.w0824.com/linux/Centos7搭建GIT服务器.html

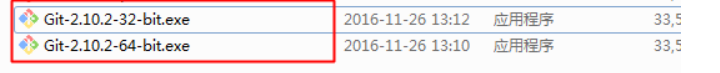
免费第三方托管平台：github、码云

git发明者：是Linux创始人李纳斯

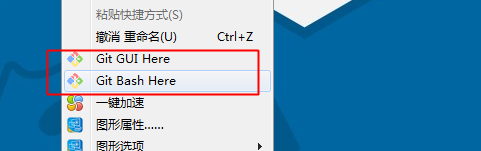
# 安装git客户端工具

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。



# 全局设置用户名和邮箱

* 全局设置（一定要设置）：先要设置提交的用户名和邮箱，不设置则无法提交代码。

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

# 创建git仓库

获取git仓库有两种方式，

1. 在本地目录中执行git init指令，初始化一个空仓库。

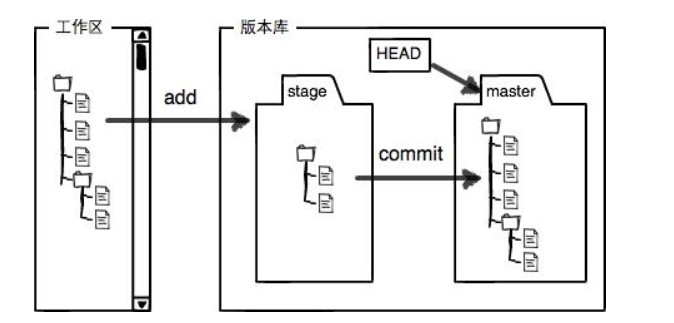
2. 从远程服务器（github、码云）拉取一个仓库。如从github拉取，或是从自己搭建的私有git服务器拉取。

# 4、git工作流

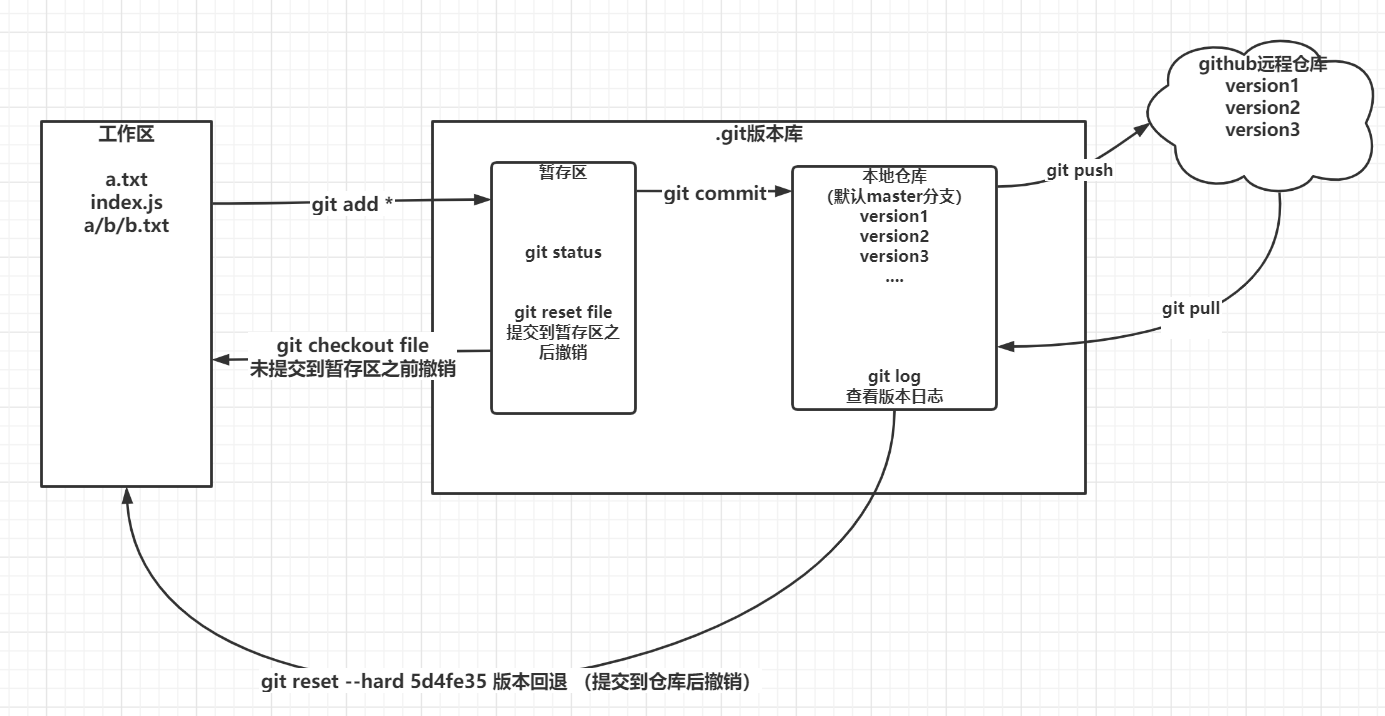
一个git仓库包含以下三个部分：

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。

本地版本库：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， 其中HEAD 指针是指向当前所在的分支。

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

命令具体使用，可以在终端输入,命令 –help,如下：

git commit --help

## git add

作用：添加当前工作目录中的文件添到暂存区中:

git add . #添加当前工作目录中的所有文件到暂存区

git add filename #添加当前工作目录中指定文件到暂存区:

## git status

作用：查看暂存区状态:

git status

## git commit

作用：把暂存区中的文件提交到本地仓库中

git commit -m '备注信息'

对于已经跟踪过的文件（已经提交的），中间可以省略add步骤即已经被之前仓库跟踪过。

git commit -am '备注信息...'

## git log

作用：查看提交备注（日志）的信息

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。

## [git之alias别名操作](https://www.cnblogs.com/zi-wang/p/12350236.html)

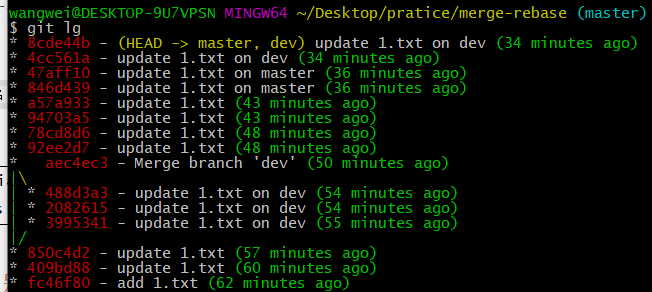
在使用git时，会遇到一些带有较多参数的长命令，可以使用alias别名的操作简化长命令

git config --global alias.别名 "命令全称"

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"

效果如下：



也可以直接在配置文件中修改：

vim ~/.gitconfig

[alias]

co=checkout

ci=commit

st=status

pl=pull

ps=push

dt=difftool

l=log—stat

cp=cherry-pick

ca=commit-a

b=branch

## git log和git reflog的区别

git log可以显示所有提交过的版本信息，不包括已经被删除的commit 记录和reset 的操作

git reflog是显示所有的操作过记录，包括提交，回退的操作。

一般用来找出操作记录中的版本号，进行回退。

参考:

<https://www.atlassian.com/git/tutorials/git-log>

## git checkout 、git reset

checkout和reset命令都可以用于git的撤销。

## reset实现版本回退



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)

## checkout撤销文件修改

## git rm

作用：删除文件 .删除之后记得commit提交并push推送，远程仓库才会一起删除！

git rm filename

## git push推送 / pull拉取

作用： git push命令用于将本地分支的更新，推送到远程主机。而pull是拉取远程分支的更新到本地。两者是相反的。

* 推送:

git push 远程名称 本地分支名:远程分支名

* 拉取：

git pull 远程分支:本地分支

注意，分支推送顺序的写法是<来源地>:<目的地>，所以git pull是<远程分支>:<本地分支>，而git push是<本地分支>:<远程分支>。

git push -u origin master（第一次推送加-u，当本地和远程分支同名时，只需简写一个即可）

上面命令表示，将本地的master分支推送到origin主机的master分支。如果后者不存在，则会被新建。

如果当前分支与多个主机存在追踪关系，那么这个时候-u选项会指定一个默认主机，这样后面就可以不加任何参数直接使用git push。

不带任何参数的git push，默认只推送当前分支，这叫做simple方式。

* 查看和修改本地的远程仓库地址：

git remote -v #查看本地的远程仓库地址

git remote rm origin # 移除本地远程仓库地址

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

* 从远程服务器拉取内容:

git pull orgin master 拉取远程仓库代码并合并到本地master分支

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

* 当远程和本地仓库不相关时，可以强制合并到本地：

git pull origin master --allow-unrelated-histories

## git blame files

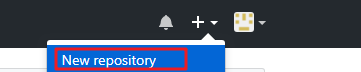
作用: 查看文件的每行代码是谁写的，尤其实现发现了错误代码的情况下，想跑都没门。

git blame files

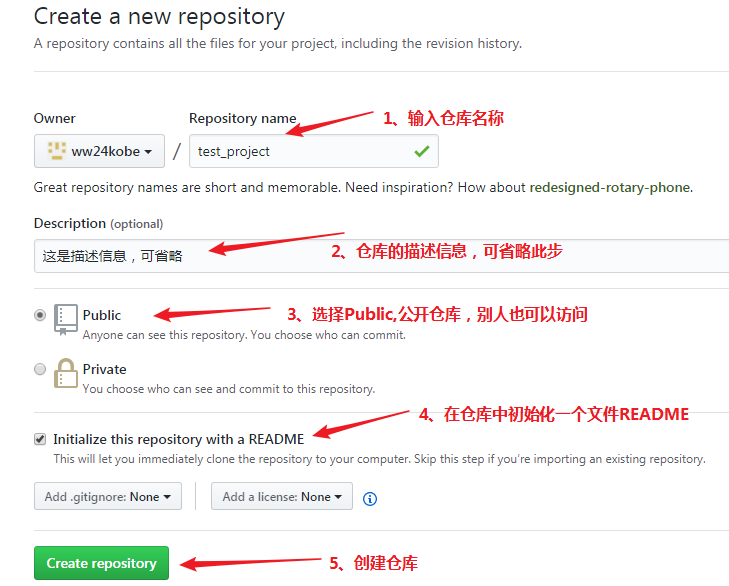
# 在github中创建仓库

创建仓库比较简单，参考下面的图片即可：

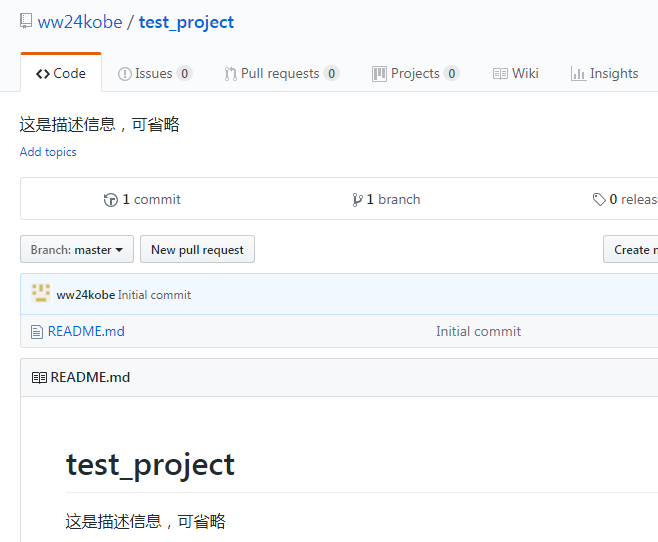
1、点击加号+，选择New repository新建仓库



2、输入创建仓库的信息

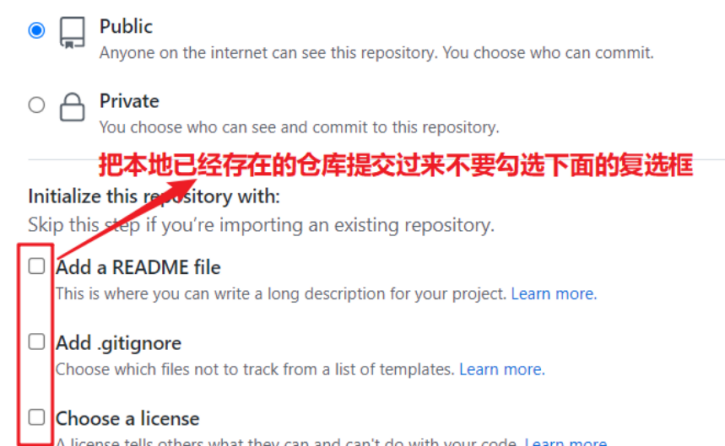


创建好之后，如下所示：



若导入一个本地已经存在的仓库，则不需要勾选以下复选框。

勾选了就是两个不相关的仓库了，后面还需要强制关联起来，比较繁琐！



# 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

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

# 创建.gitignore忽略文件

必须要和.git隐藏文件夹在同一目录

在项目仓库中可以创建此文件，可以指定哪些文件不需要提交

查看各语言规则参考： <https://github.com/github/gitignore>

# commit提交日志规范

Commit message格式：

<type>: <subject>

git commit -m ‘feat:登录功能’

git commit -m ‘fix:修复上传文件的预览’

注意冒号后面有空格。

type：用于说明 commit 的类别，只允许使用下面7个标识。  
subject：是 commit 目的的简短描述，不超过50个字符，且结尾不加句号（.）。

提交类型type有如下常见的值：

* feat：新功能（feature）
* fix：修补bug
* docs：文档（documentation）
* style： 格式（不影响代码运行的变动）
* refactor：重构（即不是新增功能，也不是修改bug的代码变动）
* test：增加测试
* chore：构建过程或辅助工具的变动（家庭杂务，杂活；累活，苦差事）

# 9、获取和拉取

git pull （拉取）= git fetch +git merge   
git fetch（获取） = 拉取之后还要手动合并 git merge origin/master

# 10、创建标签（版本号）

* 查看所有标签

git tag

* 创建标签

git tag v1.0.0 -m '完成注册' -m '完成订单'

* 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。

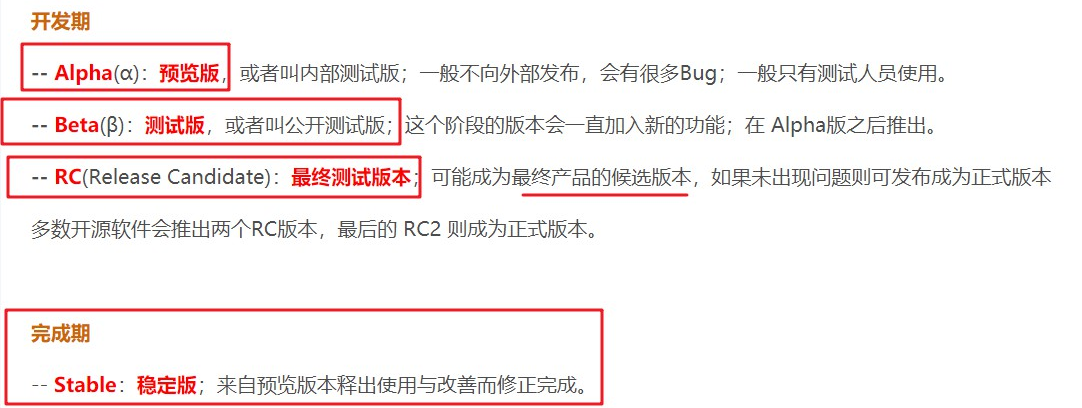
git push --tags

版本控制规范：

版本格式：主版本号.次版本号.修订号，版本号递增规则如下：

* 主版本号：当你做了不兼容的 API 修改，
* 次版本号：当你做了向下兼容的功能性新增，
* 修订号：当你做了向下兼容的问题修正。

开源软件常见的版本说明：



* alpha版 : 内部测试版。此版本表示该软件在此阶段主要是以实现软件功能为主，通常只在软件开发者内部交流，一般而言，该版本软件的Bug较多，需要继续修改。
* beta版: 公开测试版。 该版本相对于alpha版已有了很大的改进，消除了严重的错误，但还是存在着一些缺陷，需要经过多次测试来进一步消除,后续还会不断的增加新功能。如果你是发烧友，可以下载这个版本。
* RC版: 全写：Release Candidate，该版本已经相当成熟了，基本上不存在导致错误的BUG，与即将发行的正式版相差无几。
* Release版: 该版本意味“最终版本”，在前面版本的一系列测试版之后，终归会有一个正式（发行）版本，是最终交付用户使用的一个版本。该版本有时也称为标准版。一般情况下，Release不会以单词形式出现在软件封面上，取而代之的是符号(Ｒ)。
* Stable版：稳定版。 用户可以放心大胆的用了。

如微信小程序开发工具版本：

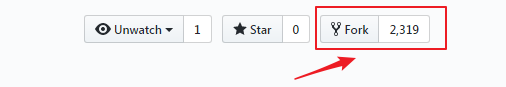


# 11、HEAD指针

简单的说：HEAD是指向前当前分支中最后一次提交的引用。

# 12、fork克隆操作

可以把别人的项目fork（理解为复制）到自己的用户名下面



# 13、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。

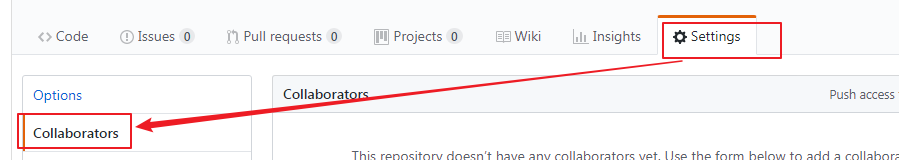
# 14、pull request 即发一个PR请求

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，项目作者会把此请求合并merge到项目中。

合并成功之后，你就是这个开源项目的贡献者了！

# 15、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。



# 16、github代码搜索技巧

参考：

<https://help.github.com/cn/github/searching-for-information-on-github/getting-started-with-searching-on-github>

<https://github.com/search/advanced> ： 高级搜索，可以生成查询条件

如：找出点赞数超过100，且名字含有vue字样的。



如：找出后缀名为.vue的文件，且含有v-for指令的使用。



# 17、设置ssh协议免密提交

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。

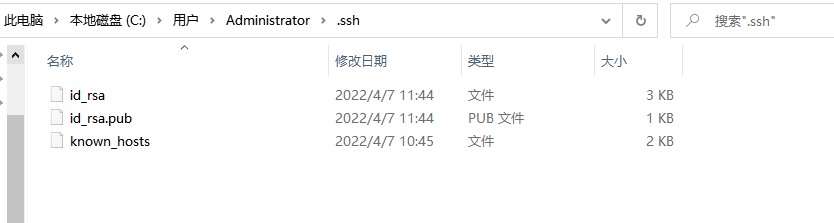
私钥：理解为锁的钥匙，在本地电脑存放着。

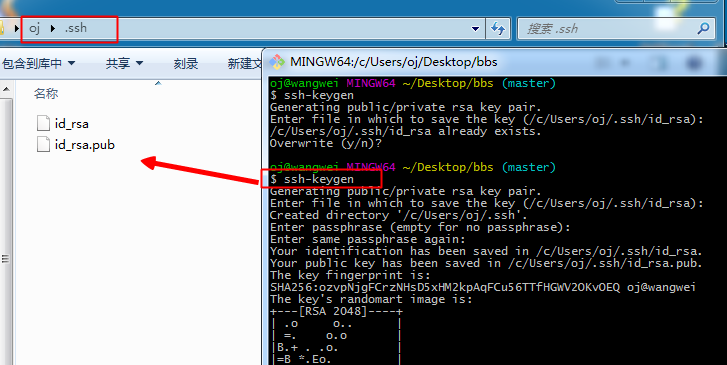
也就是说只有锁的对应钥匙才可以进行提交代码。

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。

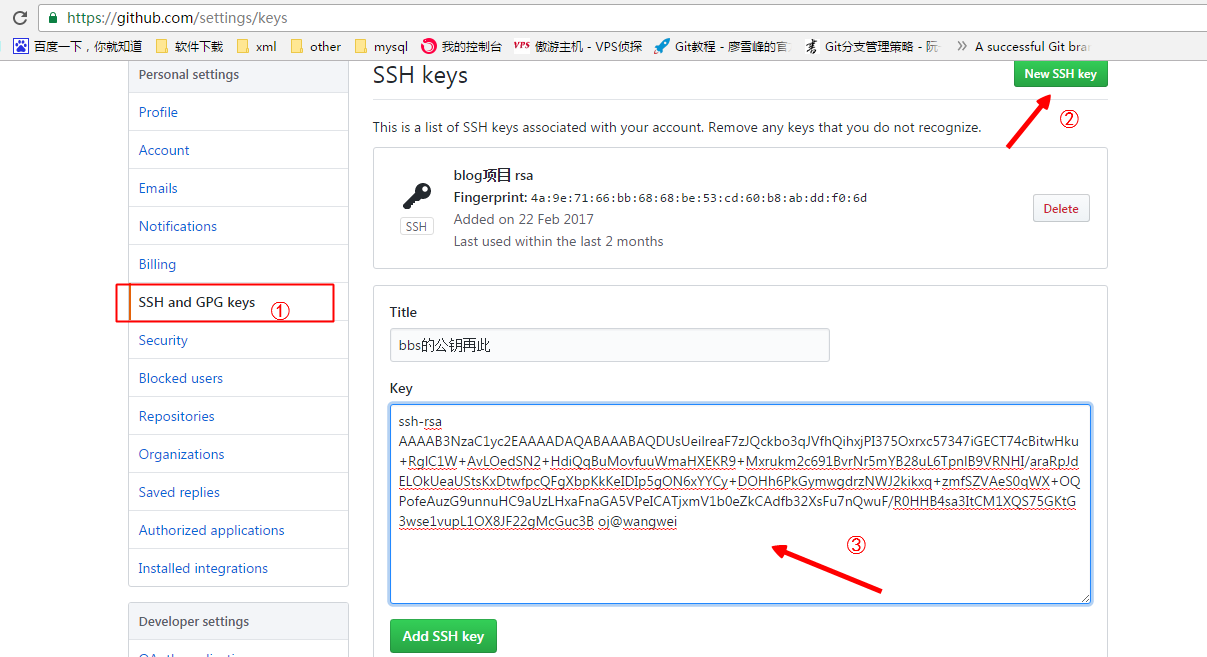
id.rsa:私钥文件

id\_rsa.pub:公钥文件

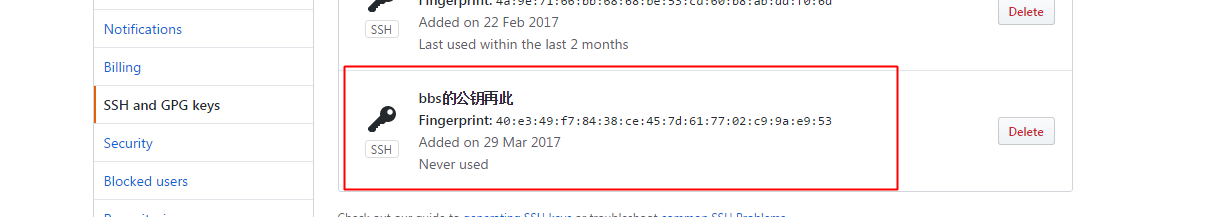
文件目录:



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：



输入以下指令,查看是否配置成功：

ssh -T git@github.com

提示有successfully字样代表成功。

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

上面配置好之后，push推送测试下即可。

# 18、git分支

## git分支

master分支：这是每个仓库默认都有的分支，主要用来发布代码正式版本的。

dev分支：此分支需要自己去创建，平常代码的开发在此分支上进行。

master分支和dev开发分支配合工作：

一般多是在开发分支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

* 创建并切换到dev分支

git checkout -b dev

* 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，这样可以让分支线更加清晰。

* 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

* 拉取远程origin的dev分支

git checkout -b origin/dev

有关git分支的管理策略文档：

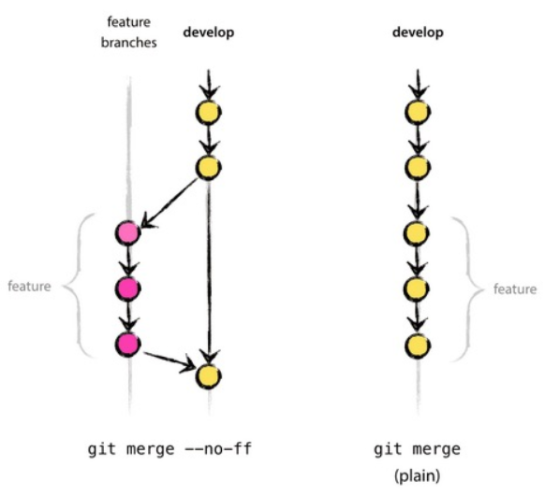


## merge合并分支

git merge --no-f <branch> #合并指定branch分支到当前分支

git merge --no-ff 可以保存你之前的分支历史。能够更好的查看 merge历史，以及branch 状态。git merge 则不会显示 feature，只保留单条分支记录。

如图：

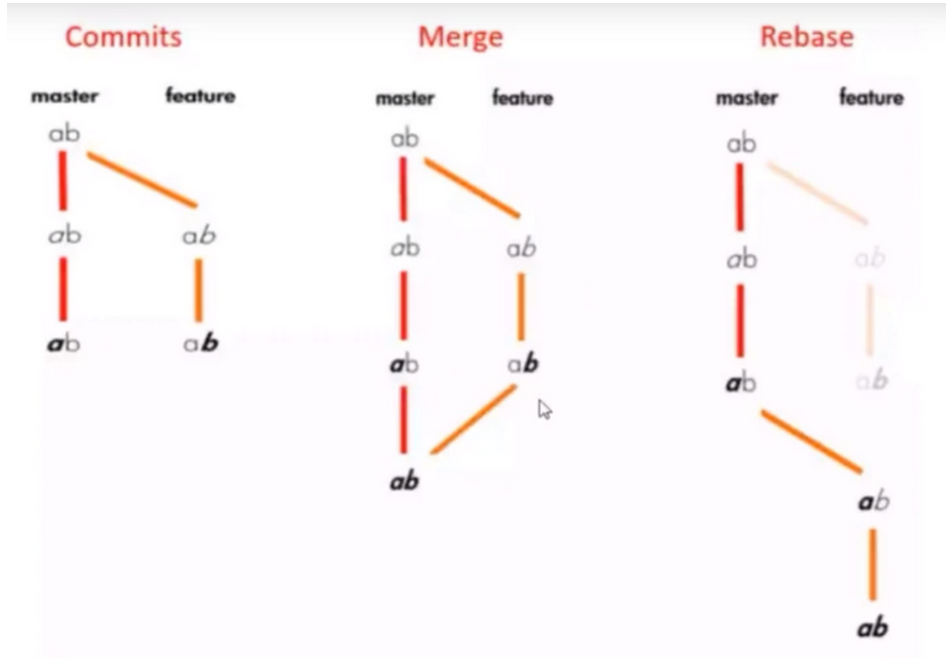


参考：

<https://www.atlassian.com/git/tutorials/merging-vs-rebasing>

## git merge 和 git rebase的区别

rebase: 这就是rebase操作的特点：把分叉的提交历史“整理”成一条直线，看上去更直观。缺点是本地的分叉提交已经被修改过了。



git pull --rebase

合并前：

D---E master

/

A---B---C---F origin/master

使用 merge --no-f 合并后：

D--------E

/ \

A---B---C---F----G master, origin/master

如果是 rebase 的方式，就不会有 G 合并点：

A---B---C---F---D'---E' master, origin/master

# 19、搭建自己git服务器（私有仓库）

参考地址: http://blog.w0824.com/linux/Centos7搭建GIT服务器.html

可以学完linux之后再看。