# 一：创建版本库

mkdir learngit 创建目录

cd learngit 进入目录

ls 目录查看列表

pwd 打印当前目录

git init 当前目录设置为 git 仓库

git add readme.txt 文件添加到 git 仓库

git commit –m ‘write a new file’ 文件提交到仓库，-m 后跟提交说明

为什么要将提交分为两步： 添加可以添加不同文件，commit 可以一次提交很多文件，git add a.txt b.txt; git add c.txt; git comit –m ‘commit three files’

# 二：时光穿梭机

git log 查看提交版本历史，head 表示当前版本

git log --pretty = oneline 使用一行展示

C:\Users\Administrator\AppData\Local\Temp\1547554857(1).png

git reset --hard head~1, head 后跟数字表示回退到倒数第几个版本

git reset --hard 版本号前几个字母，回到指定的版本

git reflog 查看之前的版本回退命令，可以让你来回回退

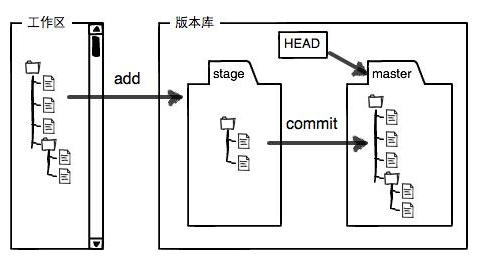
工作区：创建的 git 目录，提交的文件放在工作区

版本库： 隐藏目录 .git 就是版本库，stage 暂存区，master 第一个分支，master 的一个指针 head

git add readme.txt 命令是将文件添加到 stage 区

git commit –m ‘\*\*\*’命令是将 stage 内容提交到 master

对于文件的每次修改都需要先 git add 添加到 stage 中,最后可以统一 commit 提交。可以使用 git status 查看当前工作区的状态



git status 查看当前工作区的状态，如下结果表示当前工作区没有需要添加的文件也没有需要提交的文件

C:\Users\Administrator\AppData\Local\Temp\1547557959(1).png

git checkout --readme.txt 撤销操作，回退到最近一次 git add 或者 git commit 之前的操作

git rm cmdInfo.docx 从当前工作区删除文件，git commit

# 三：远程仓库

远程添加git仓库：

1：本地打开 shell (windows 打开 git bash) 创建 ssh key

ssh-keygen -t rsa -C "2237222914@qq.com"

生成公钥和私钥，用来标识是谁上传

C:\Users\Administrator\AppData\Roaming\Tencent\Users\2237222914\TIM\WinTemp\RichOle\~PNF]]7BRVJ[}DC1HHYGI22.png

2：登录 github -> settings -> new ssh key

title 随便输入

key 本地生成的公钥

3：githup 账户 –> new repository

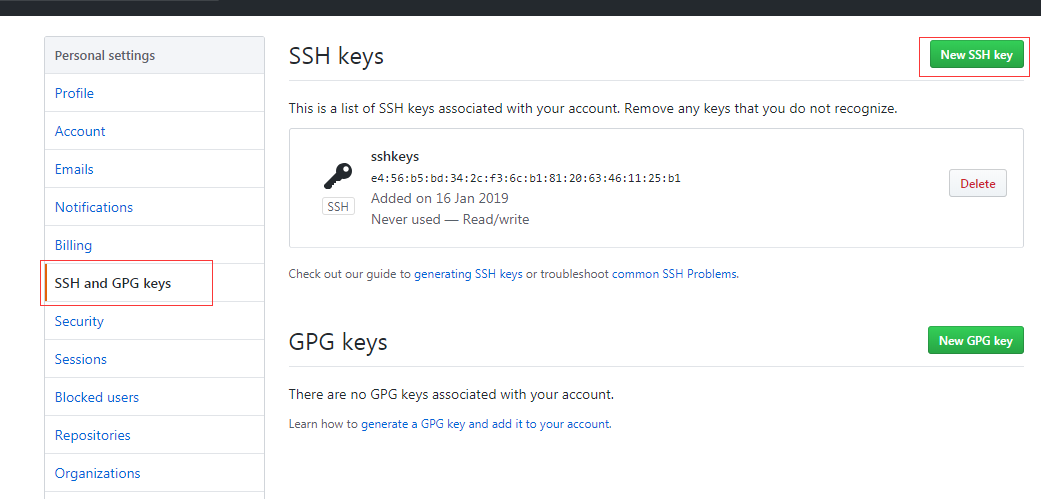
4：由于本地已经有git 项目, origin 远程仓库名

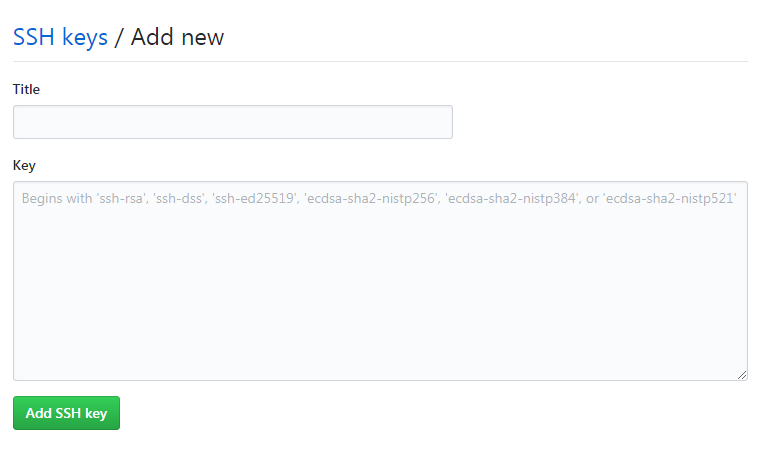
git remote add origin https://github.com/javacode123/gitlearn.git

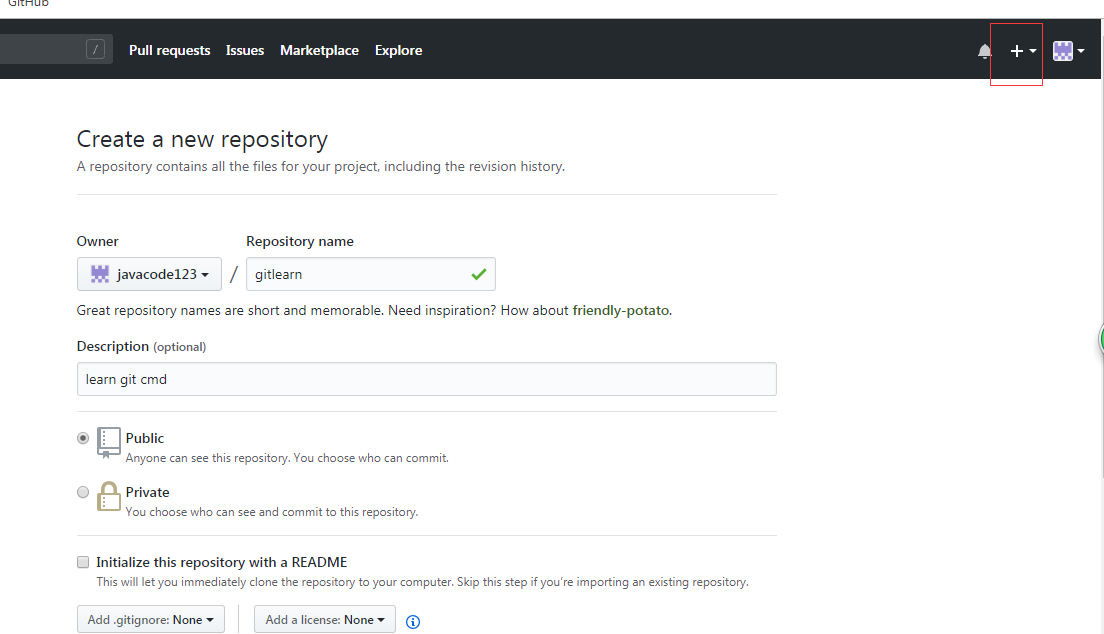
本地内容推送到 githup

git push –u origin master : origin 仓库名，master 分支名

若报错 git config --global http.sslVerify false



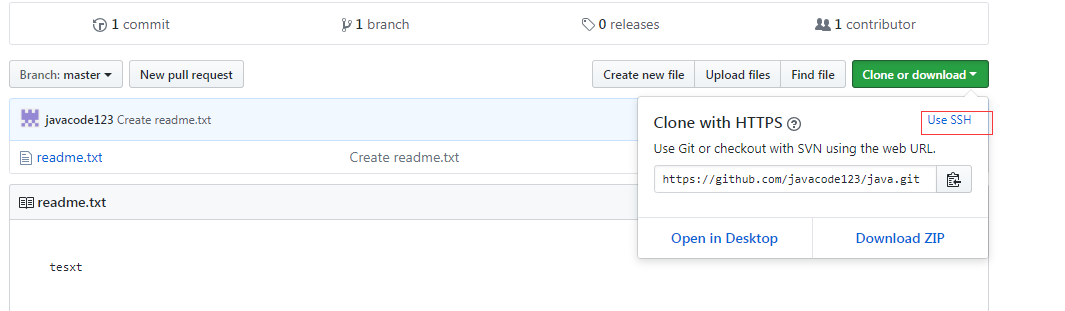




之后使用 git push origin master 提交

从远程库克隆：

1：githup 官网创建 repository



2：本地执行 git clone [git@github.com:javacode123/java.git](mailto:git@github.com:javacode123/java.git)

# 四：分支管理

git checkout –b dev 创建 dev 分支，并切换到 dev 分支，相当于以下命令：

git branch dev : 创建 dev 分支

git checkout dev : 切换 dev 分支, 例如在 dev 分支做出了修改并提交，切换到 master 之后并看不到 dev 的修改

git brach : 查看当前分支

git merge dev : 将 dev 和 master 合并

git branch –d dev : 合并之后可以删除 dev 分支

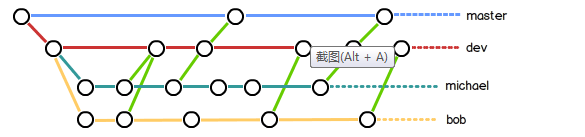
冲突处理：

当 master 分支修改了工作区并提交，dev 分支修改了工作区并提交，合并的时候会有提示发生冲突，使用 git status 查看冲突文件，打开冲突文件我们会看到 <<<<<<<，=======，>>>>>>> 来标记冲突内容, git branch –d dev 删除分支

git log –-graph : 查看分支视图

git merge –-no-ff –m ‘合并注释’dev : 使用 –no-ff 合并分支，并为这次合并添加注释

在实际开发中我们使用以下策略进行版本控制：master 用来表示最新版本，dev 表示测试版本，所有参加开发的开发人员提交到 dev 分支进行测试运行，例如，使用 feature-add 分支新开发一个功能，在没有提交到 dev 之前需要删除：git branch –D feature-add



多人协作：

1：a: git remote –v 查看远程信息

b: 远程克隆文件 git clone [git@github.com:javacode123/gitlearn.git](mailto:git@github.com:javacode123/gitlearn.git)

2：b: 在 dev 分支上开发，‘origin/dev’表示用来 push

git checkout –b origin/dev : 创建分支

git add readme.txt : 添加

git commit –m ‘\*’: 提交

git push origin origin/dev: dev ‘origin/dev’:本地分支名字，‘dev’:服务器分支名字

3：a: 也在 dev 上开发，并添加，提交

git push origin origin/dev: dev ‘origin/dev’:本地分支名字，‘dev’:服务器分支名字，发生冲突。

git pull origin dev 下拉文件，解决冲突

git push origin origin/dev:dev 上传服务器

因此，多人协作中，首先要 pull 取内容，然后 push

# 五：标签管理

对提交的某个记录打一个标签，便于识别，经常用于版本控制。

git tag v1.0 : 对当前提交添加标签

git tag : 查看所有标签

git tag 版本号首字母几位 : 对某次提交历史添加标签

git show v1.0 : 查看版本信息

git tag –d v0.0 : 删除指定标签

git push origin v1.0 : 推送某个标签到服务器

git push origin --tags : 推送全部标签到服务器