# 1、本地库初始化

**git init**

# 2、设置签名

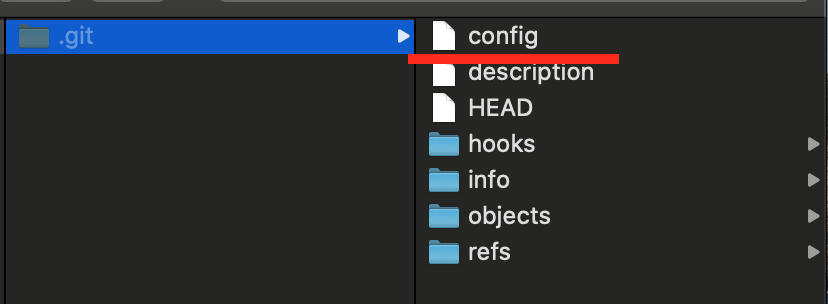
作用：区分不同开发人员的身份

辨析：这里设置的签名和登录远程库（代码托管中心）的账号、密码没有任何关系

2.1、项目级别/仓库级别：仅在当前本地库范围内有

**git config user.name RyuuShihin**

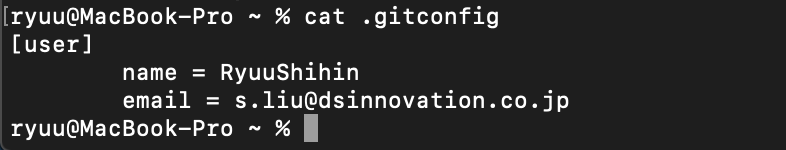
**git config user.email lsben1014＠gmail.com**



2.2、系统用户级别：登录当前操作系统的用户范围

**git config --global user.name RyuuShihin**

**git config --global user.email lsben1014＠gmail.com**



2.3、级别优先级

就近原则：项目级别优先于系统用户级别，二者都有时采用项目级别的签名

如果只有系统用户级别的签名，就以系统用户级别的签名为准

二者都没有不允许

# 3、状态查看

**git status**

# 4、将工作区的"新建/修改"添加到暂存区

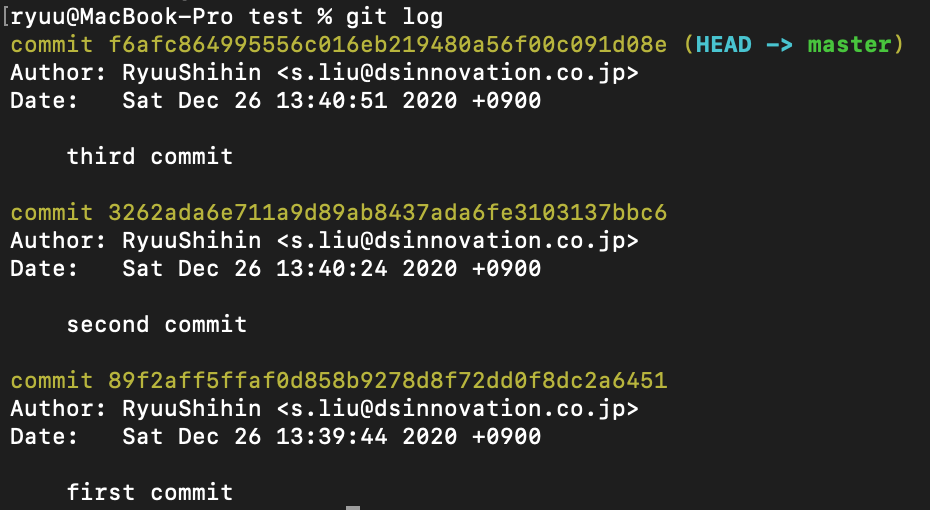
**git add 文件名**

# 5、将暂存区的内容提交导本地库

**git commit -m "提交日志文件" 文件名**

# 6、查看历史记录

6.1、**git log**



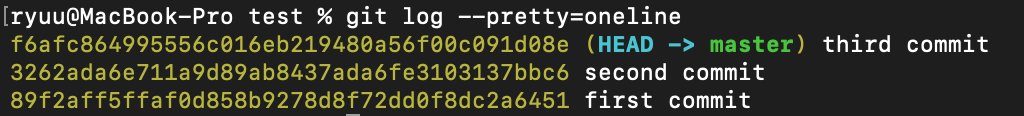
多屏显示控制方式：

空格：向下翻页

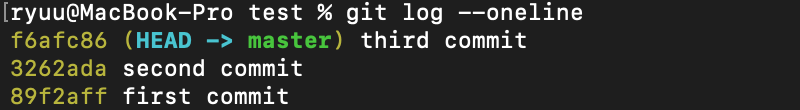
b：向上翻页

q：退出

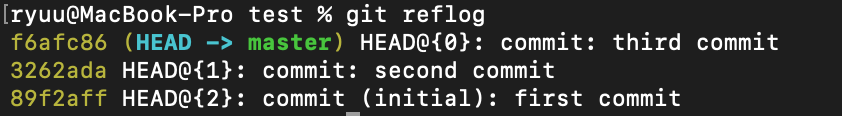
6.2、**git log --pretty=oneline（两个-）**



6.3、**git log --oneline**



6.4、**git reflog**



HEAD@{移动到当前版本需要多少步}

# 7、前进后退

7.1、基于索引值操作

**git reset --hard 版本号**

7.2、使用^符号：只能后退

**git reset --hard HEAD^**

注：一个^表示后退一步，n 个表示后退 n 步

7.3、使用~符号：只能后退

**git reset --hard HEAD~n**

注：表示后退 n 步

# 8、reset命令的三个参数对比

8.1、仅仅在本地库移动HEAD指针

**git reset --soft 版本号**

8.2、在本地库移动HEAD指针

重置暂存区

**git reset --mixed 版本号**

8.3、在本地库移动HEAD指针

重置暂存区

重置工作区

**git reset --hard 版本号**

# 9、删除文件并找回

前提：删除前，文件存在时的状态提交到了本地库

操作：git reset -hard 版本号

9.1、删除操作已经提交到本地库：指针位置指向历史记录

9.2、删除操作尚未提交到本地库：指针位置使用HEAD

# 10、比较文件差异

10.1、将工作区中的文件和暂存区进行比较

**git diff 文件名**

10.2、将工作区中的文件和本地库历史记录比较

**git diff 本地库中历史版本 文件名**

10.3、不带文件名比较多个文件

**git diff**

# 11、分支操作

11.1、创建分支

**git branch 分支名**

11.2、查看所有分支

**git branch -v**

11.3、切换分支

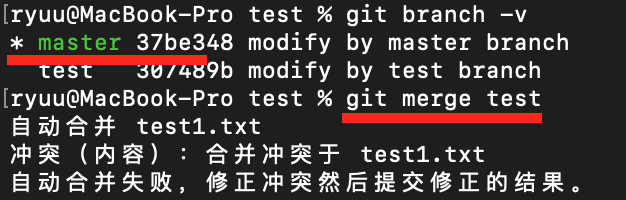
**git checkout 分支名**

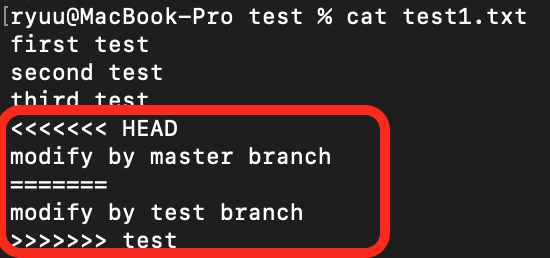
11.4、合并分支

**git checkout 被合并分支名**

**git merge 有新内容分支名**

11.5、解决冲突





a、 编辑文件，删除特殊符号

b、 把文件修改到满意的程度，保存退出

c、 git add 文件名

d、 **git commit -m "日志信息"**

**注：此时commit一定不能带具体文件名**

# 12、GitHub

12.1、查看当前所有远程地址别名

**git remote -v**

12.2、创建远程库地址别名

git remote add 别名 远程地址

例：**git remote add origin 地址**

12.3、推送

**git push origin master**

12.4、克隆

**git clone 地址**

**注：clone是本地没有repository时，将远程repository整个下载过来**

12.6、拉取

a、 pull=fetch+merge

b、 git fetch 远程库地址别名 远程分支名

git merge 远程库地址别名 远程分支名

git pull 远程库地址别名 远程分支名

注：pull是本地有repository时，将远程repository里新的commit数据（如 果有的话）下载过来，并且与本地代码merge

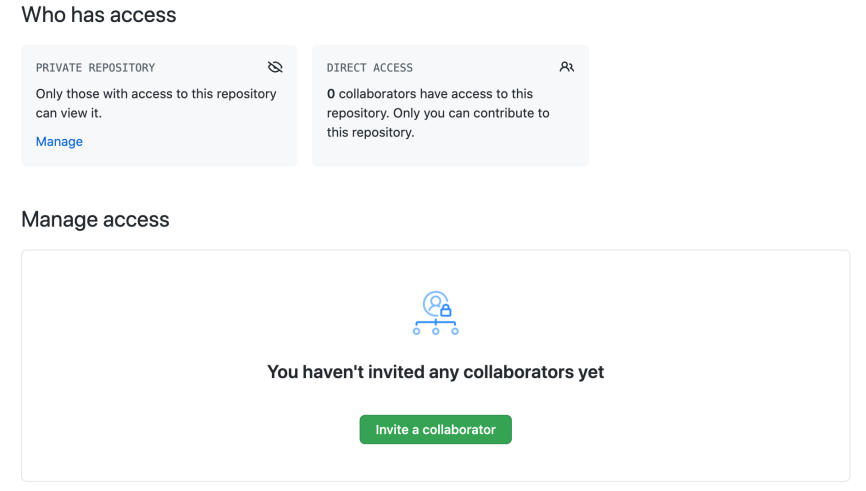
12.7、解决冲突

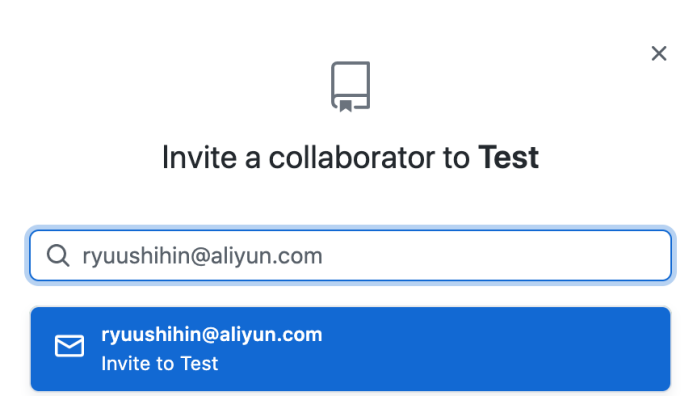
a、 如果不是基于GitHub远程库的最新版所做的修改，不能推送，必须先拉取

b、 拉取下来后如果进入冲突状态，参照【11.5、解决冲突】操作

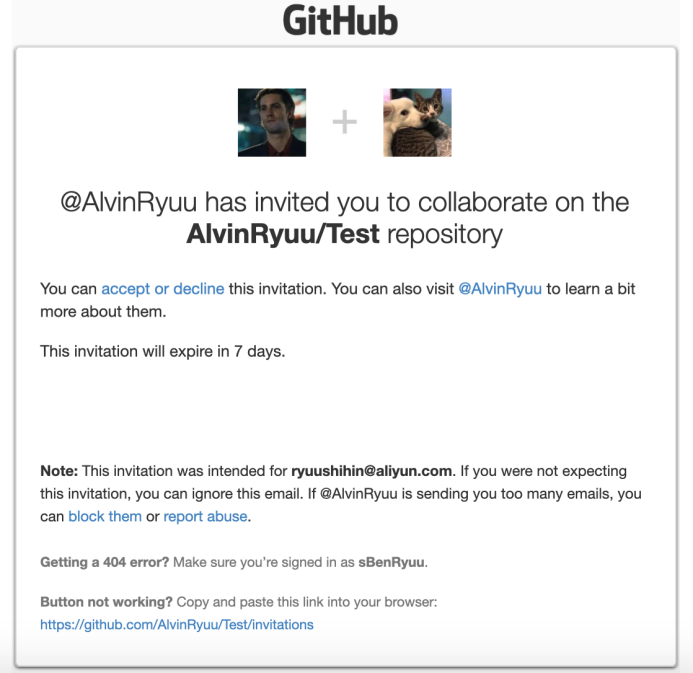
12.8、团队成员邀请

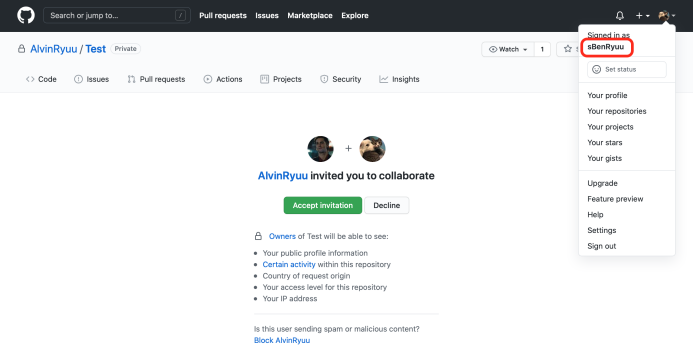
a、 AlvinRyuu邀请sBenRyuu





b、 sBenRyuu收到邮件接收邀请



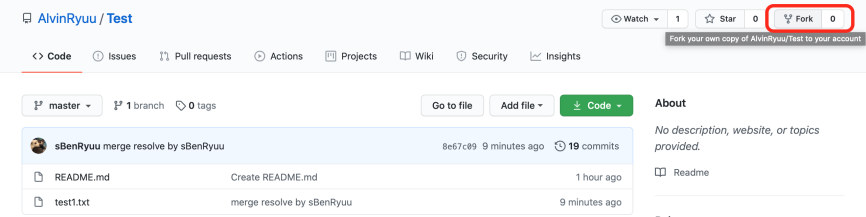


12.9、跨团队协作

a、 Fork

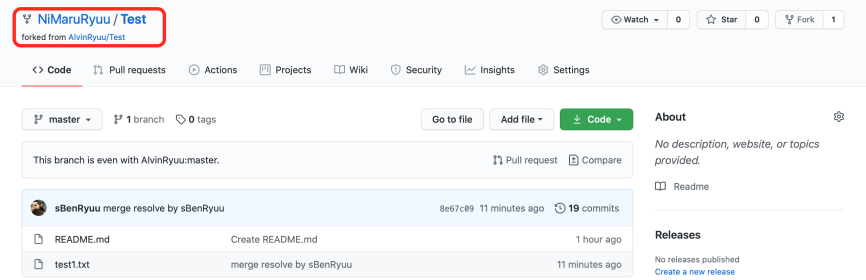
Fork的前提：AlvinRyuu的Test仓库必须是public

以NiMaruRyuu的身份找到AlvinRyuu的Test仓库，然后点击Fork



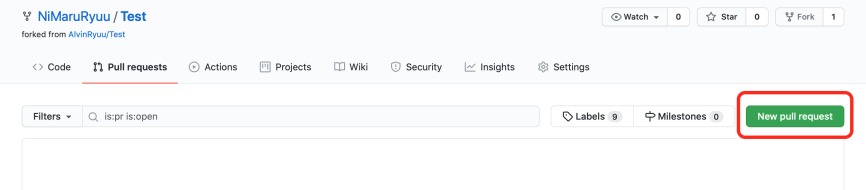
在点击Fork以后，NiMaruRyuu本地就会增加一个新的仓库Test，但是署名 是NiMaruRyuu的仓库

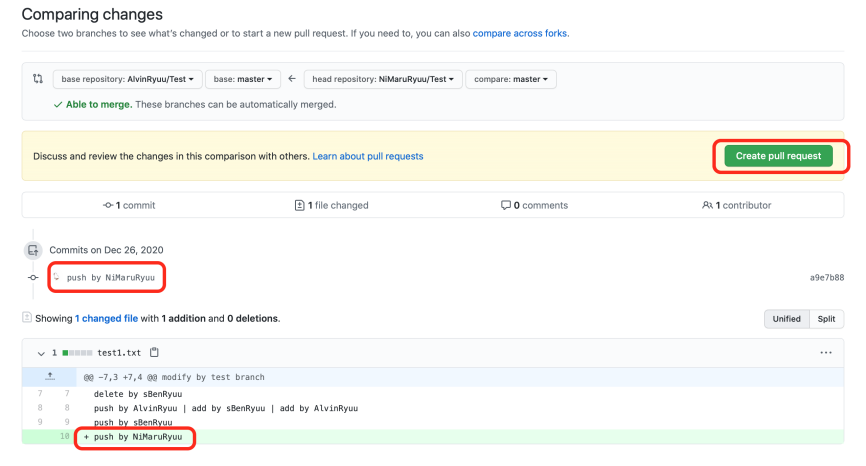
也就是相当于NiMaruRyuu拷贝一份AlvinRyuu的Test仓库放进自己的账户

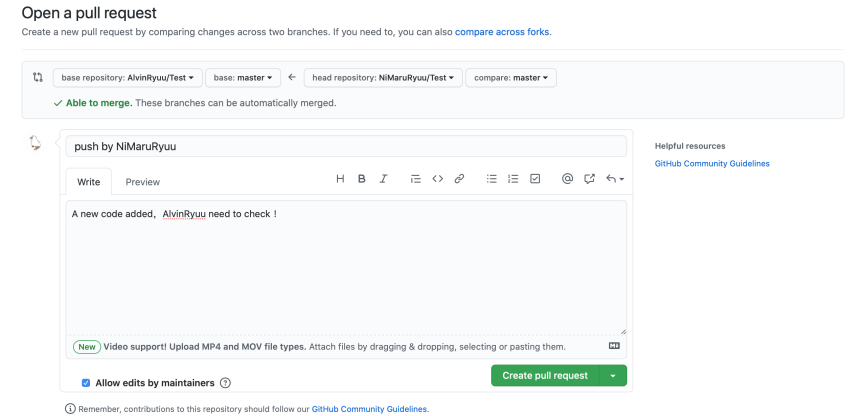


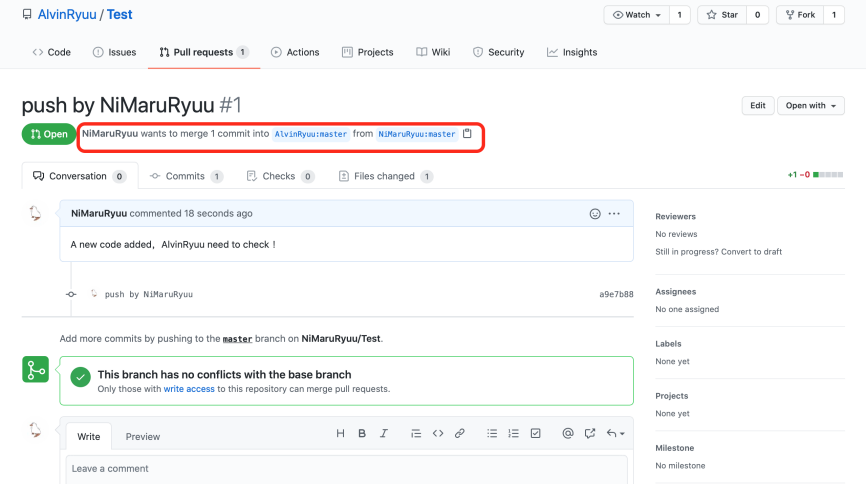
b、 pull request

在NiMaruRyuu的账户界面发起pull request

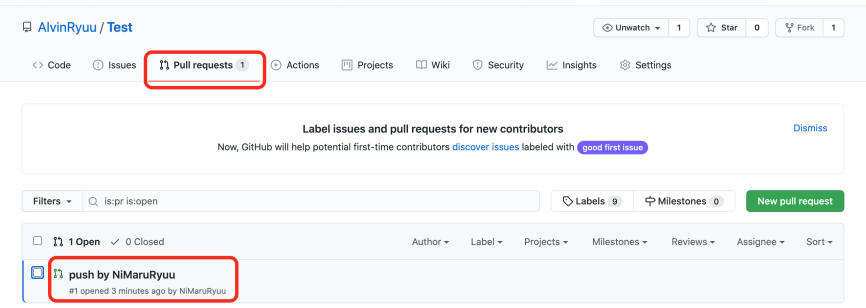




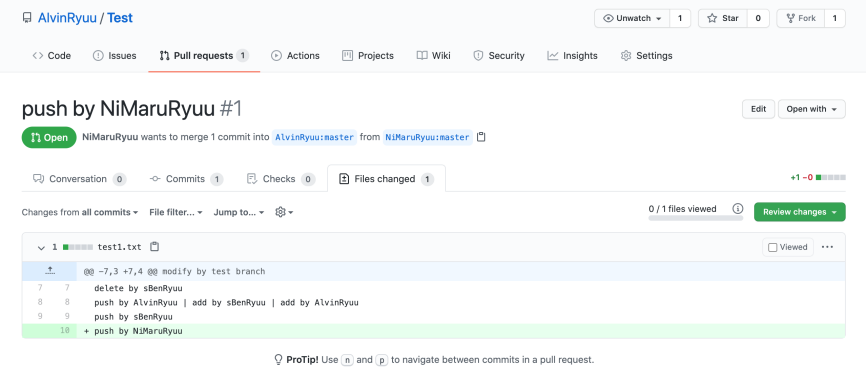




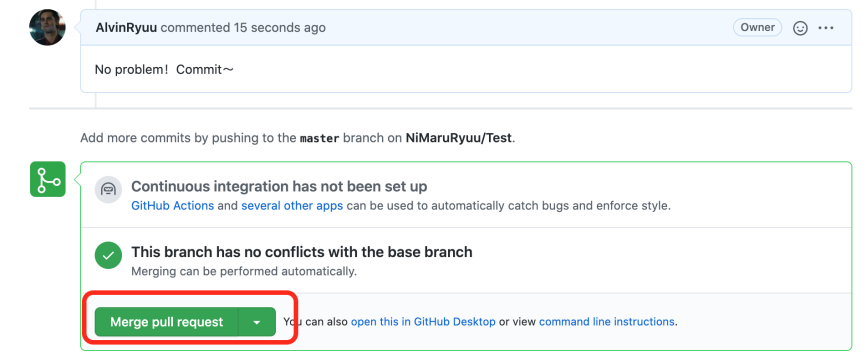
c、 回到AlvinRyuu的界面，就会发现有一个新的pull request



AlvinRyuu可以Review代码



确认没有问题后Merge



# 13、SSH操作

13.1、进入当前用户的家目录

**cd ~**

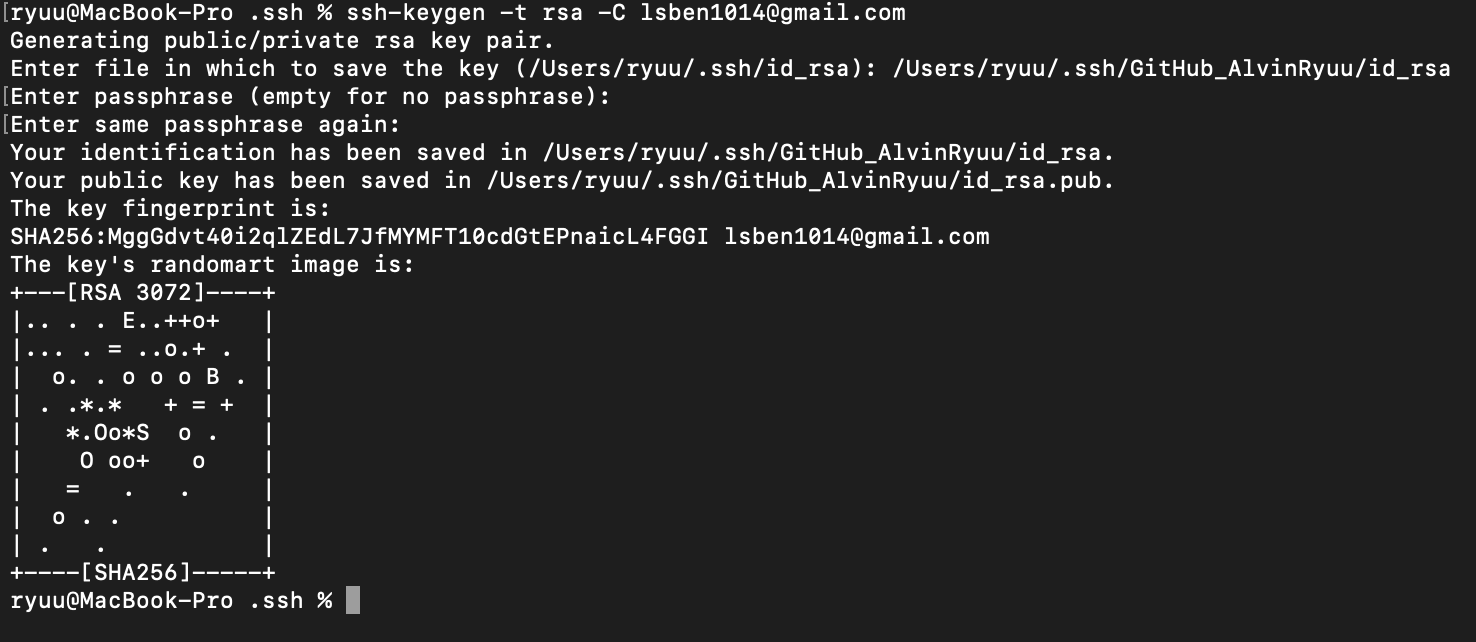
13.2、删除.ssh 目录

**rm -rvf .ssh**

13.3、运行命令生成.ssh 密钥目录

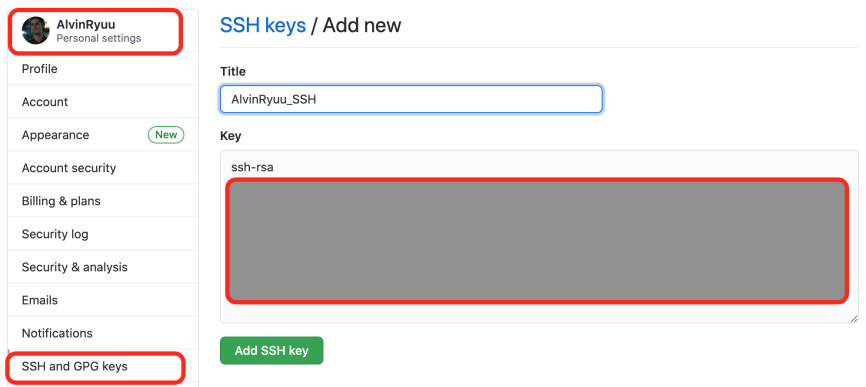
**ssh-keygen -t rsa -C lsben1014@gmail.com**

13.4、指定文件夹



13.5、复制id\_rsa.pub的内容

13.6、



13.7、在 ~/.ssh 目录下新建config文件



13.8、配置系统ssh\_config文件（可忽略）

**sudo vim /etc/ssh/ssh\_config**



13.9、将私钥添加到钥匙串中（键入密码）

**eval `ssh-agent -s`**

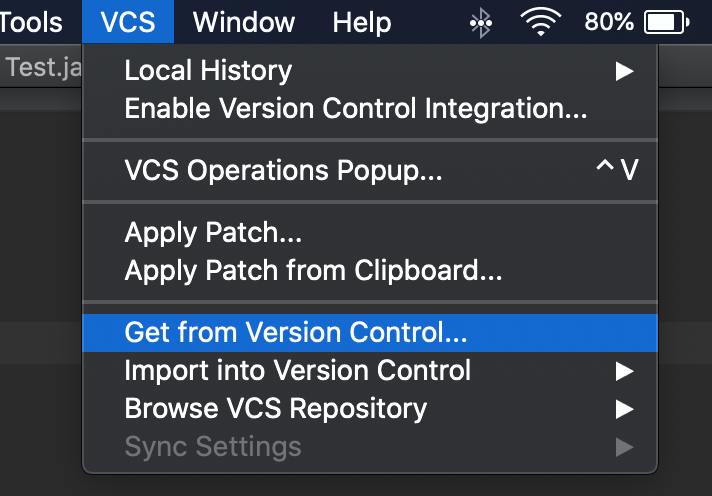
**ssh-add ~/.ssh/GitHub\_AlvinRyuu/id\_rsa**

13.10、在设置remote的时候，必须加上~/.ssh/config文件中的host

**git remote add origin git@AlvinRyuu.github.com:AlvinRyuu/Test.git**

# 14、IDEA+GitHub

14.1、



14.2、

