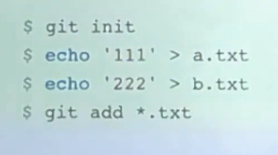
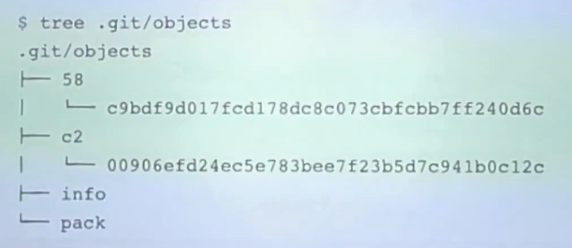
**git 学习笔记**

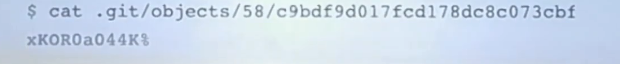
# 0.git原理

https://www.bilibili.com/video/BV11z4y1X79p?from=search&seid=15274598805648294925

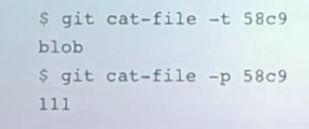
https://www.bilibili.com/video/BV1RJ411X7kh?from=search&seid=15274598805648294925

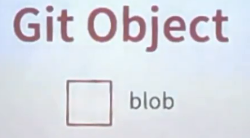




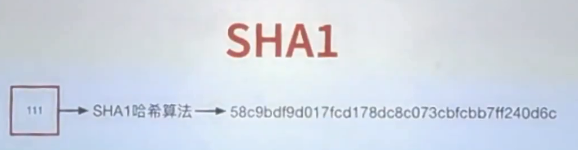
(二进制压缩乱码)

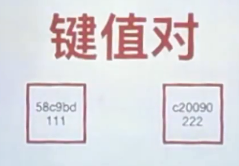




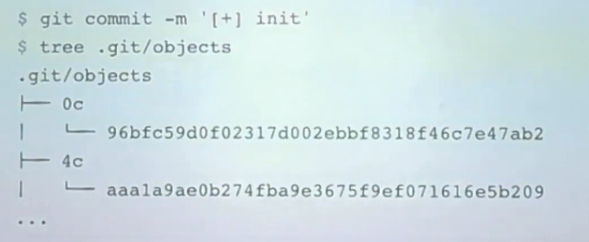


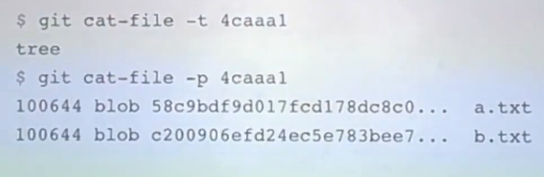
git object：储存文件的最小单元，blob是储存文件的具体内容。

  
git使用键值对数据库，当我们使用git.add后，git object中会多出响应的内容，sha1值指向响应的blob

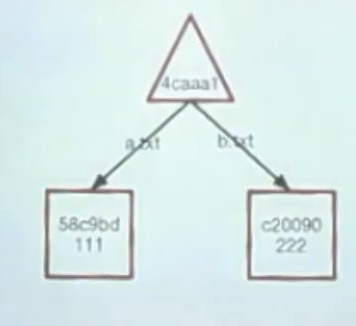


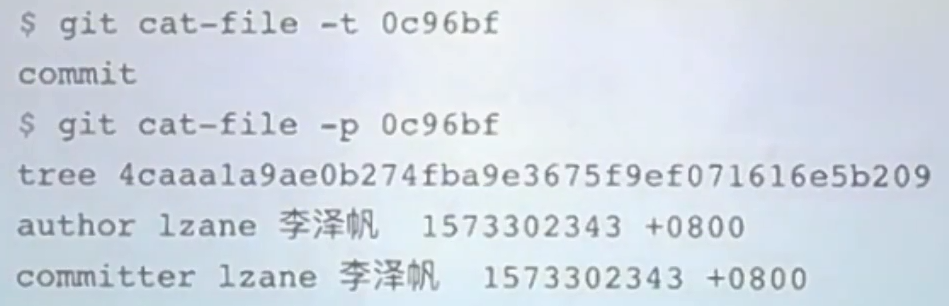
接着执行 git commit后，object中多出两个文件

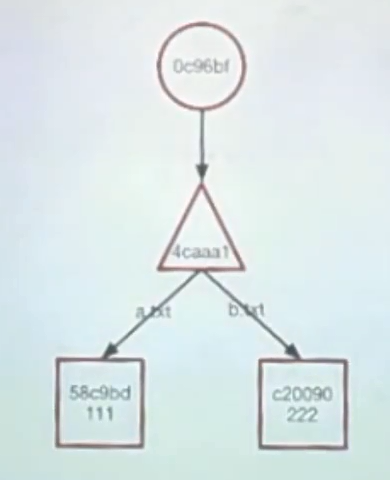




一个tree object指向blobobject



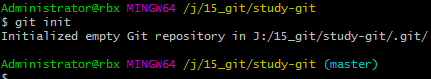




# 1.创建仓库

1.使用本地仓库

Git init命令可以使用一个已经存在的目录作为git仓库



在执行完成 **git init** 命令后，Git 仓库会生成一个 .git 目录，该目录包含了资源的所有元数据，其他的项目目录保持不变。

2.克隆仓库

git clone git://github.com/schacon/grit.git

如果遇到以下错误：

ssh: connect to host github.com port 22: Connection timed out

fatal: Could not read from remote repository.

Please make sure you have the correct access rights

and the repository exists.

解决办法为：

C:\Users\Administrator\.ssh

git config --global user.name "knight"

git config --global user.email "1952933329@qq.com"

ssh-keygen -t rsa -C "1952933329@qq.com"

然后一直回车，就会在.ssh文件夹下生成公私钥对，把公钥添加到响应的githug即可



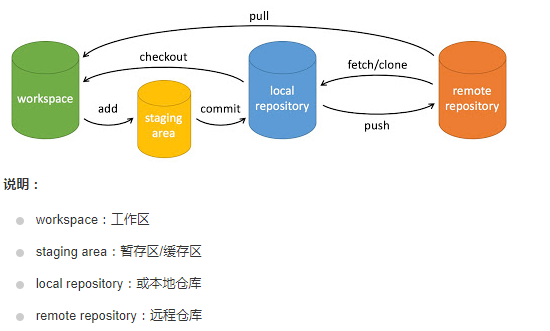
### 经测试，git协议比ssh更简单，但是没有直接可以copy的路径，需要修改。

### git://github.com/helloworld729/Spider.git

git push -u git://github.com/helloworld729/40\_torch-self-learning.git master



# 2.基本操作



## 2.1 add

git add 将文件添加到暂存区，git atatus -s 查看各个文件的状态，加上-s是简单版的信息。

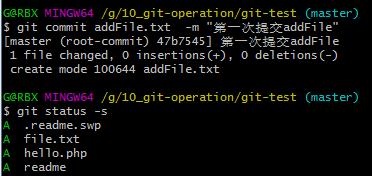
git add filename 可以将工作区的文件添加到暂存区。也可以多个文件用空格间隔，或者直接用 add . 表示添加所有的文件。

## 2.2 commit

git commit [filename] -m "备注信息"

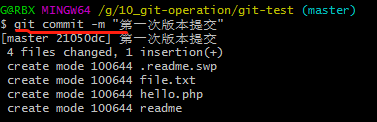
git commnit [filename] -am "备注信息"

前面我们用add将数据添加到暂存区，现在用commit可以添加到本地仓库



commit之后，暂存区就没有相应的文件了。 -m之后添加一些备注信息

如果git commit之后没有加具体的文件，就会把所有暂存区的文件提交到本地仓库



git commnit [filename] -am "备注信息"



该命令会将本地工作区中修改后，还未使用git add . 命令添加到暂存区中的文件也一并提交上去。相当于git add -u 与git commit –m “本次提交描述”两句操作合并为一句进行使用。

push

**git push** 命用于从将本地的分支版本上传到远程并合并。

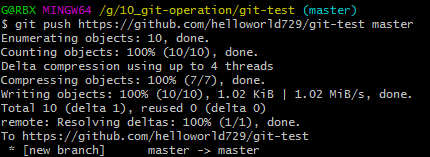
**git pull** 其实就是 **git fetch** 和 **git merge FETCH\_HEAD** 的简写。 命令格式如下：

git push <远程主机名> <本地分支名>:<远程分支名>

如果本地分支名与远程分支名相同，则可以省略冒号：

git push <远程主机名> <本地分支名>

https://github.com/helloworld729/git-test



push的时候添加-u参数

## 2.3 push

git push 远程主机名 本地分支名[：远程分支名]

如果本地版本与远程版本有差异，但又要强制推送可以使用 --force 参数：

git push --force origin master

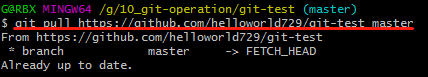
删除主机但分支可以使用 --delete 参数，以下命令表示删除 origin 主机的 master 分支：

git push origin --delete master

## 2.4 pull

git pull 远程主机名 远程分支名:本地分支名

如果远程分支是与当前分支合并，则冒号后面的部分可以省略。



强制覆盖：

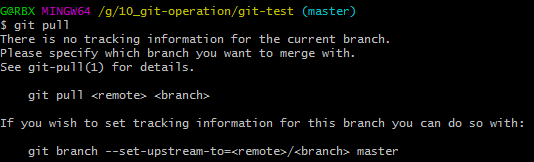
    git fetch --all

     git reset --hard origin/master

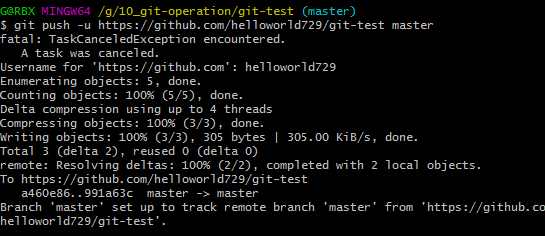
     git pull

## 2.5分支追踪

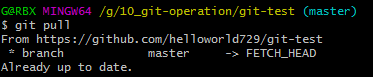
一般情况下，如果我们直接使用简化的pull命令，会报错如下：



为此，我们可以在push的时候添加 -u 参数，然后可能需要填写 账号和密码



加上-u参数，这样push时，本地指明分支就和远程主机的同名分支建立追踪关系，在以后的推送或拉取时就可以简化命令。

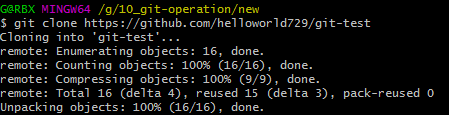


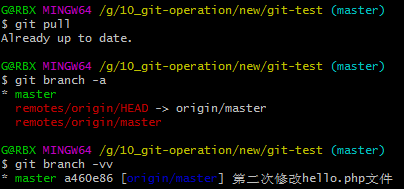
其他建立追踪的方式参见：https://blog.csdn.net/qq\_42780289/article/details/97762596

## 2.6 clone

git clone url

假如我们在本地拷贝远程的工作目录，建议使用git clone 而不是git pull，虽然都可以下载，但是clone下载的项目自动包含分支追踪，可以用git pull 直接拉去项目，不用配置分支。





默认情况下，Git 会按照你提供的 URL 所指向的项目的名称创建你的本地项目目录。 通常就是该 URL 最后一个 / 之后的项目名称。如果你想要一个不一样的名字， 你可以在该命令后添加空格并加上你想要的名称。



详细参考：https://www.runoob.com/git/git-basic-operations.html

## 其他实用操作：

**1、每次push都需要输入用户名+密码(http协议克隆)**

在git push时每次都要输入用户名和密码是一件很痛苦的事，其实解决方法很简单步骤如下：

在项目文件夹（包含隐藏文件夹.git文件夹的）内右键 git bash here

在git bash交互环境输入命令

git config credential.helper store

再次运行git push 就不用输入用户名和密码了，因为在用户主目录文件夹多了一个文件git-credentials，这个就是用来存储用户名和密码的

若想把项目的用户名和密码删掉，输入下列命令

git credential-manager uninstall

这样每次git push时又需要输入用户名和密码了。

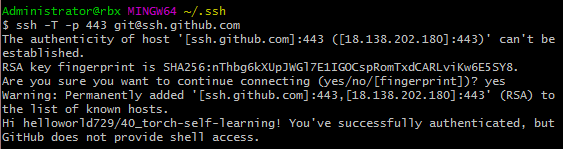
**2、如何使用ssh协议clone项目，以及简化push操作(避免每次都要验证用户名和密码)**

前面的实验表明，如果克隆的时候用http协议，那么每次push的时候都会要求填github的用户名和密码，网上说可以用ssh协议解决这个问题。

1. 首先用前面的方法生成公私钥对
2. 测试与远程服务器的链接



但是直接通过网页是可以访问的，所以可能是默认端口配置问题：用443测试



确定访问后，发现knownhost文件里面增加了本次连接的私钥。

为了解决上述端口问题，需要手动写一份配置文件

在.ssh文件夹下面touch一份config文件后，按照key空格value格式写入：

Host github.com

User "1952933329@qq.com"

Hostname ssh.github.com

PreferredAuthentications publickey

IdentityFile ~/.ssh/id\_rsa

Port 443

再次测试连接成功：



之后就可以直接clone和push了。

**Ubuntu配置远程访问：**

一下操作不需要进入python解释器(除了第二种生成密码方式)，而是在conda环境下

1.生成配置文件

1 jupyter notebook --generate-config

2.生成连接密钥

这里有两种方法

先说第一种：使用password指令生成密钥，生成的密钥在一个文件里

jupyter notebook password

第二种方法：在python环境中生成密钥

python

from notebook.auth import passwd

passwd()

会生成一组密钥sha1:\*\*\*\*\*\*\*\*\*:\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*这种格式，也有可能是argon2格式的。

复制下来

3.修改配置文件

vim ~/.jupyter/jupyter\_notebook\_config.py

按以下规则修改

c.NotebookApp.ip='\*'　　　　 # 允许所有ip访问

c.NotebookApp.password = u'sha:ce...' # 第2步复制的密钥(注意第一个单引号前加小写u)

c.NotebookApp.open\_browser = False　　# 可改可不改 默认打开浏览器

c.NotebookApp.port =8888 　　　　　　 # 外部连接的端口

c.NotebookApp.allow\_remote\_access = True　　# 不改就报错，被坑了一下午

保存退出就可以了

在windows上输入ip地址:端口号，输入密码进入bingxianren

后台执行jupyter：

screen -S jupyter 新建名为jupyter的窗口并进入，开启jupyter服务

按ctrl+A+D分离窗口，使得jupyter notebook在另一命令窗口保持运行；新窗口用于其他任务；