1. 设置本地git与github

目的：项目的运行环境为redhat7，而想在windows环境下编辑项目。因此，在windows和redhat7都部署git环境。windows环境下每次修改之后同步到github上，然后在redhat下运行。（使用github是为了在不同的环境下，只要有网络都可以编辑该项目。如果项目有隐私要求，则可以自己搭建git服务器，而不使用github托管）。

* 1. 基本设置

首先分别在redhat和windows环境下安装git。(redhat可以yum安装，windows可以去git官网下载windows版本的git)。然后在redhat和windows下分别进行如下设置：

1. ssh-keygen -t rsa -f github -C “coolkilling@163.com”

此命令会在当前目录下生成github和github.pub文件，其中github为生成的私钥，github.pub为生成的公钥。

-t rsa表示用rsa加密方式

-f github表示生成的密钥名字为github

-C “coolkilling@163.com”会在公钥中加入comment

注：如果之前有用ssh-keygen生成过密钥，则当前生成的密钥可能不会生效。需要删除旧的密钥。cd ~/.ssh/就能看到旧的密钥。可以使用旧的密钥，而不用新生成密钥。

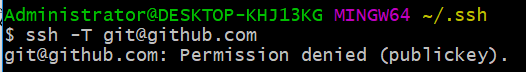
1. github->settings->New SSH key

在key一栏，填入more github.pub得到的公钥信息。

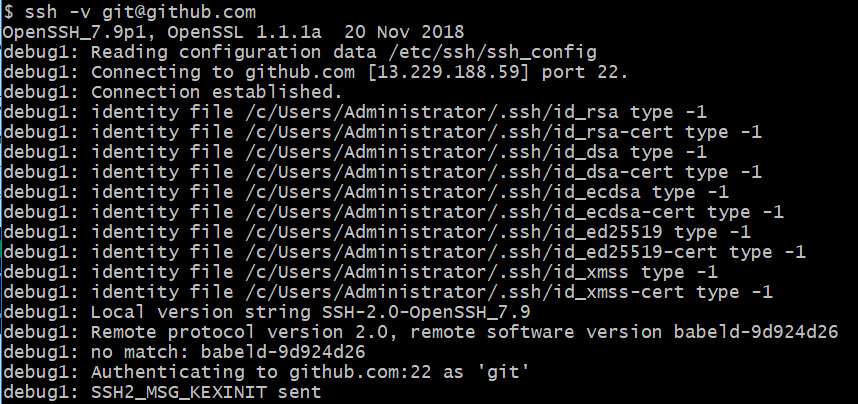
使用ssh -T git@github.com，如果出现以下信息则表示设置ssh登陆成功：



可能出现如下的错误：



通过ssh -v git@github.com可以看到debug信息：



错误的原因是：

ssh命令会去ROOT-PATH/.ssh目录下找名为id\_rsa（等）的文件。由于ssh-keygen生成的密钥（私钥github和公钥github.pub）在运行命令时的路径下，且名字分别为github和github.pub。因此解决办法为：复制当前路径下的github和github.pub文件到ROOT-PATH/.ssh目录下，并分别命名为id\_rsa和id\_rsa.pub即可。

1. 设置git用户信息

git config --global user.name “xxx”

git config --global user.email xxx@gmail.com

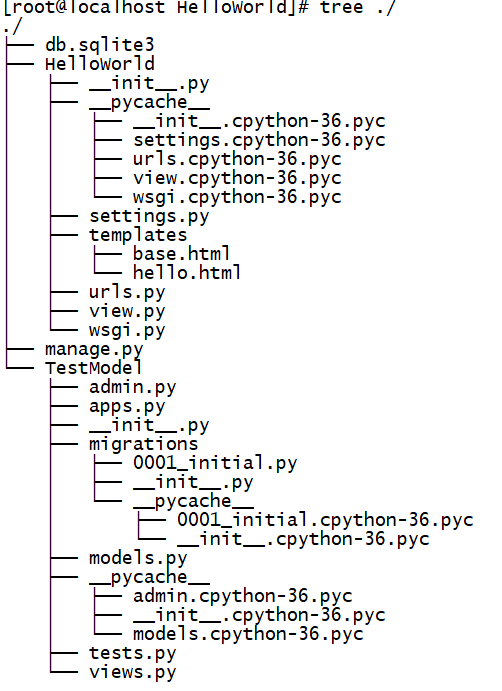
两条命令分别设置git的用户名字和邮箱，每次git提交的时候会显示这个用户名，用以区别到底是哪个用户提交的。

可以通过git config --get user.name来查询git用户名。

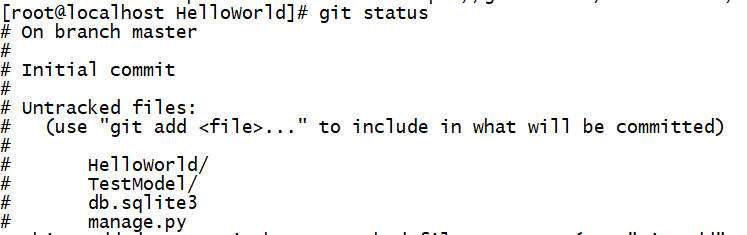
* 1. git与github
     1. 创建本地git库

1. cd命令进入想要建库的文件夹

这里以django的一个最简单的模板项目HelloWorld为例子。项目的所有文件都在…./HelloWorld目录下，tree ./显示的目录如下：



1. git init
2. git status显示



提示要将HelloWorld文件夹、TestModel文件夹、db.sqlite3文件和manage.py文件git add。

依次执行git add HelloWorld/等。

1. git commit -m “init add”
   * 1. 将本地库与github库关联
        1. 推送本地库到github--push命令
2. 登陆github创建新的repository
3. 本地输入命令：

remote设置有两种方式，ssh和http，在1.1中我们已经设置了ssh免密登陆。http方式每次都要输入账号和密码，因此这里我们选择ssh方式。

ssh方式：git remote add origin git@github.com:ToBeBetterM/django-HelloWorld.git

http方式 ：git remote add origin https://github.com/ToBeBetterM/django-HelloWorld.git

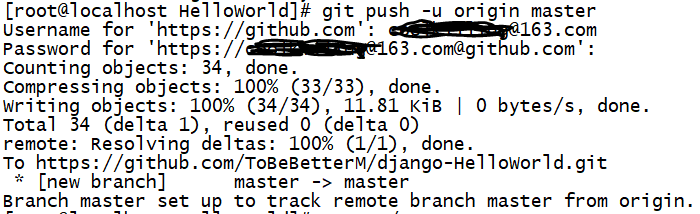
git remote -v可以查看当前的remote库情况。

将本地库关联到github的新建库（django-HelloWorld），远程（github）库在本地显示名为origin。

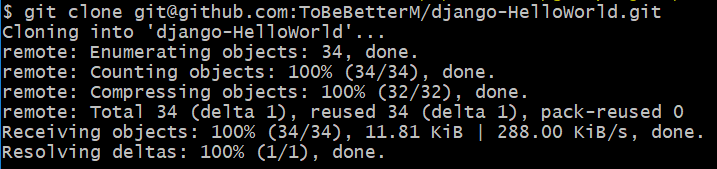
git push -u origin master

将本地的master库推送到远程库。（如果报错可以在末尾加--force）

成功执行上述两条命令后，显示如下图：

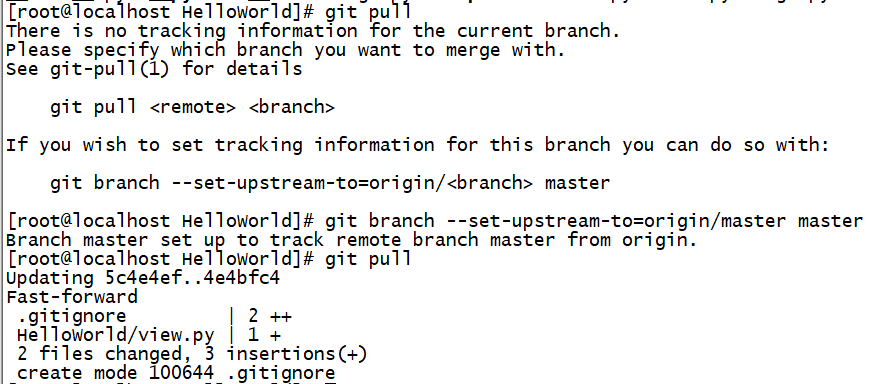


1. 如果本地做了修改，则commit之后通过以下命令可以本地最新的master修改推送到github：git push origin master
   * + 1. 从github获取库--clone命令
2. git clone git@github.com:ToBeBetterM/django-HelloWorld.git



* + - 1. 从github获取更新

git pull命令



1. git管理
   1. 创建和管理主分支
      1. 初体验

步骤：

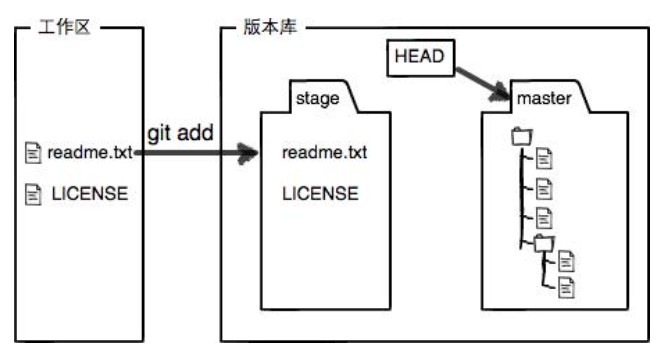
1）创建该project使用的目录，进入该目录

2）初始化git init

3）假设当前目录下有一个git.docx文件，添加该文件到git，git add git.docx

4）提交修改并为该次修改添加注释git commit -m “add git.docx”

* + 1. git重要概念--工作区和暂存区



git分支所在的目录(即你可见的部分)为工作区（working directory）。工作区的根目录有一个隐藏的目录.git，这个不属于工作区，称之为版本库（repository）。版本库包含一个暂存区(stage或者index)和 分支，每次git add将会将工作区做的修改提交到暂存区，git commit将暂存区提交的所有修改写入到分支。

注意：git commit只会提交最近一次git add时的状态。考虑以下的情况：

第一次修改🡪第一次git add🡪第二次修改🡪git commit

假设修改文件为test.txt，则git commit之后，分支的状态为第一次修改后的状态，工作区的状态为第二次的修改后的状态，git status可以看到test.txt为modified状态。

注意：git add提交的修改只会提交文件最近一次保存的状态。即如果在工作区修改了文件但是没有保存，则git add所提交的状态不会是当前编辑的状态，而是上次保存的状态。

* + 1. git常用操作
* 查看git状态

git status

* 查看历史提交

git log或者git log --pretty=oneline

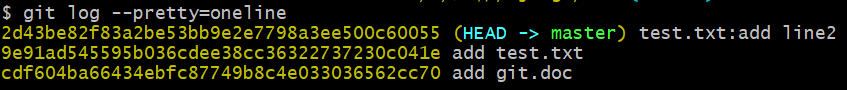
* 版本回退

HEAD表示当前版本，HEAD^表示上个版本，HEAD^^表示上上个版本，HEAD~100表示往上100个版本。

回退到上个版本：git reset --hard HEAD^

* 跳到某个版本

git log --pretty=oneline查看当前的提交历史



假如我要跳到 add test.txt这个版本，则

git reset --hard 9e91

(只需要输入部分该commit对应的UUID即可)

* 查看当前分支和工作区某文件的不同

git diff HEAD -- test.txt

* 丢弃修改：将丢弃工作区的修改，将工作区状态回退到最近一次add或者commit状态

git checkout -- test.txt

* 1. 创建与合并分支
     1. 分支常用操作
* 创建分支

git checkout -b dev 创建并切换到分支

以上语句为创建并切换到分支dev。可以分解为以下两条语句：

git branch dev 创建分支

git checkout dev 切换到分支

* 查看当前分支

git branch

* 合并分支

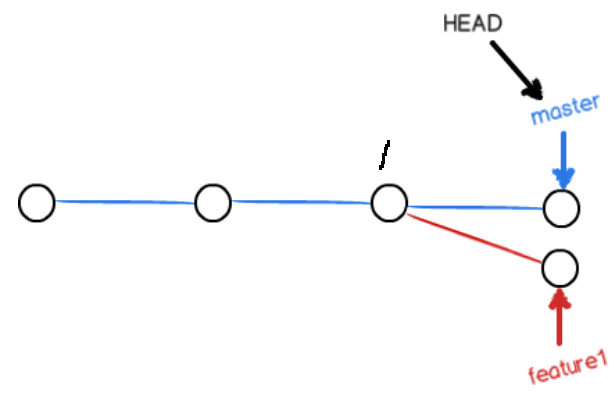
现有分支master和分支dev，打算将分支dev合并到master

首先切换到master：git checkout master

然后合并dev分支：git merge dev

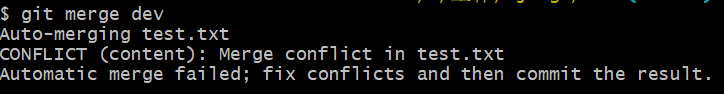
* + 1. 解决冲突

合并分支时，会产生冲突，如下图所示：



假设上述过程进行了如下的操作：

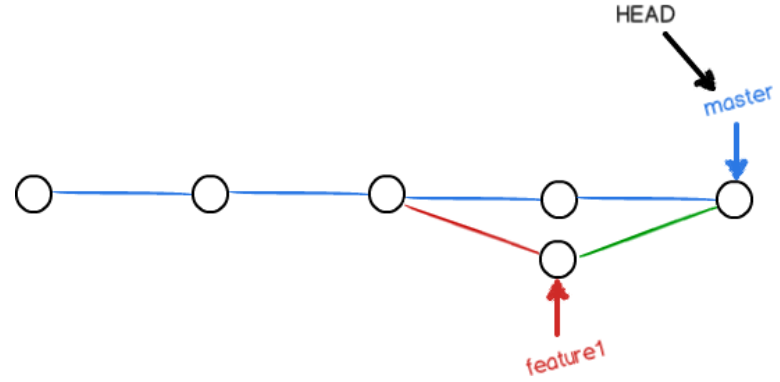
1. vi test.txt，创建test.txt，并添加一行 a new line
2. git checkout -b dev创建并切换到dev分支
3. vi test.txt，修改该行dev new line
4. git add,git commit提交dev分支的修改
5. git checkout master切换到master分支
6. vi test.txt，看到的是a new line。将该行改为master new line
7. git add,git commit提交master分支的修改
8. git merge dev合并dev分支，报错如下：



说明合并过程产生了冲突，需要修改test.txt，且此时master分支的状态为master|merging



1. vi test.txt，将那一行改为A new line
2. git add,git commit提交修改，则冲突解决，此时分支的状态如下图所示



1. 参考

[1]https://www.liaoxuefeng.com/wiki/0013739516305929606dd18361248578c67b8067c8c017b000