## GitHub初始化

在window下面使用的时候，下载window下相应的git软件就可以了

在相应文件夹下，右键选中Git GUI或者Git Bash

1. git初始化

git init

1. 设置用户名和电子邮箱，提交的时候会在repository中进行相应的记录

git config - -global user.name “username”

git config - -global user.email “emailname@email”

1. 显示当前文件夹中文件的修改状态

git ststus

1. 将特定文件或者所有文件添加到本地库中

git add dir/files或者- -a

1. 对当前文件进行提交到本地库中

git commit –m “log”

1. 将当前文件夹与远程的GitHub进行关联

git remote add origin URL

1. 上传代码到远程库中

git pull origin master/brachname #将远程库拉回到本地库

git push origin master/branchname#将本地库添加到远程库

## GitHub分支创建和管理

1、查看本地分支

git branch

2、查看远程分支

git branch –r

3、查看所有分支

git branch –a

4、创建本地分支

git branch branchname

5、切换到新的分支

git checkout branchname

6、将新创建的分支push到github中

git push origin branchname

7、删除本地分支

note:删除分支的时候，必须切换出该分支

git check master #这里切换到master分支

git branch –d branchname

8、删除github上面的远程分支

git push origin :branchname #origin和:之间有空格

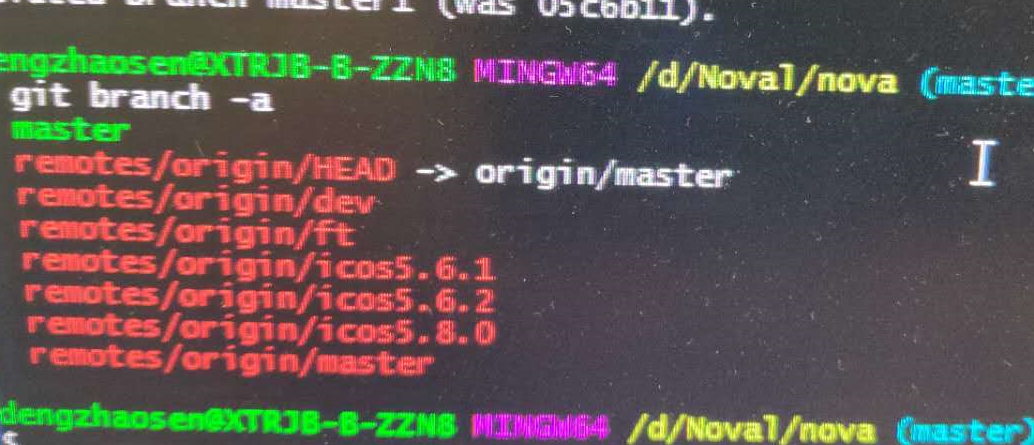
#这个命令和将分支上传到github repository的指令是类似的，只不过这里多一个冒号。

上面的应用场景是，远程仓库中没有分支，我可以在本地建立分支，然后提交到远程仓库中，但是如果远程仓库中存在分支，本地却没有分支，那么本地应该如何建立分支呢？

现在的场景：

远程代码仓库中已经存在master以及其他几个代码分支

现在只是把master分支拉下来了



现在想切换到其他分支上面，并为这个分支起一个分支别名

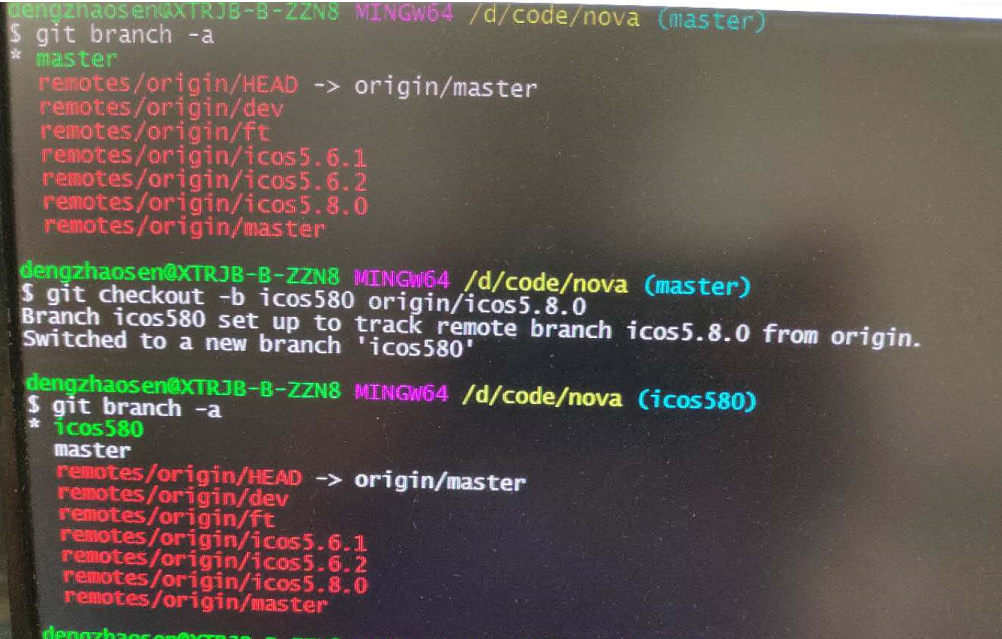
remotes/origin/HEAD ->origin/master的含义是，远程HEAD指向的是master分支

用远程分支创建本地分支的方法是：

git checkout –b newbranchname remotebranchname

创建过程如下图所示：

然后用git branch –vv命令查看本地分支对应的远程分支



## 问题

1. 向远程推送代码时出现“fatal：unable to access ‘https://github.com…’”的现象。

这种情况应检查一下是否可以联网。

不能联网的时候，会报这种错误。

2、执行git commit 的时候出现nothing added to commit but untracked files present的错误。

执行代码 git push -u origin master

第一次将代码提交到远端

## Create a new repository on the command line

通过命令行来创建repository

echo "# GitHubDocument" >> README.md

git init

git add README.md

git commit -m "first commit"

git remote add origin https://github.com/JorhsonDeng/GitHubDocument.git

git push -u origin master

## push an existing repository from the command line

git remote add origin

<https://github.com/JorhsonDeng/GitHubDocument.git>

git push -u origin master

## 用git生成patch

<https://blog.csdn.net/qq_33970713/article/details/106659396>

应用场景：

只想将当前文件夹中改动的文件打包成一个patch，并不想提交相应的代码

执行命令：

git diff nova/api/openstack/compute/servers.py nova/cmd/api.py nova/compute/manager.py nova/conductor/manager.py nova/scheduler/manager.py nova/virt/libvirt/driver.py > debug.log

当再次用这个patch的时候，执行命令

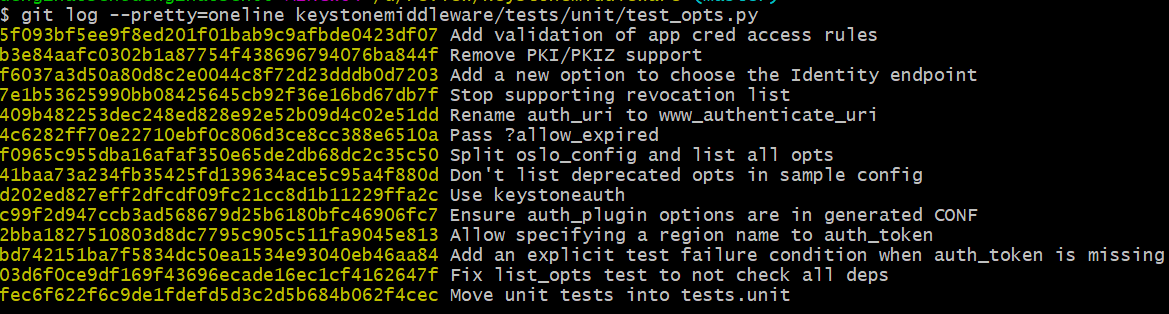
git apply debug.log即可

## git查看历史提交记录

查看单个文件的：

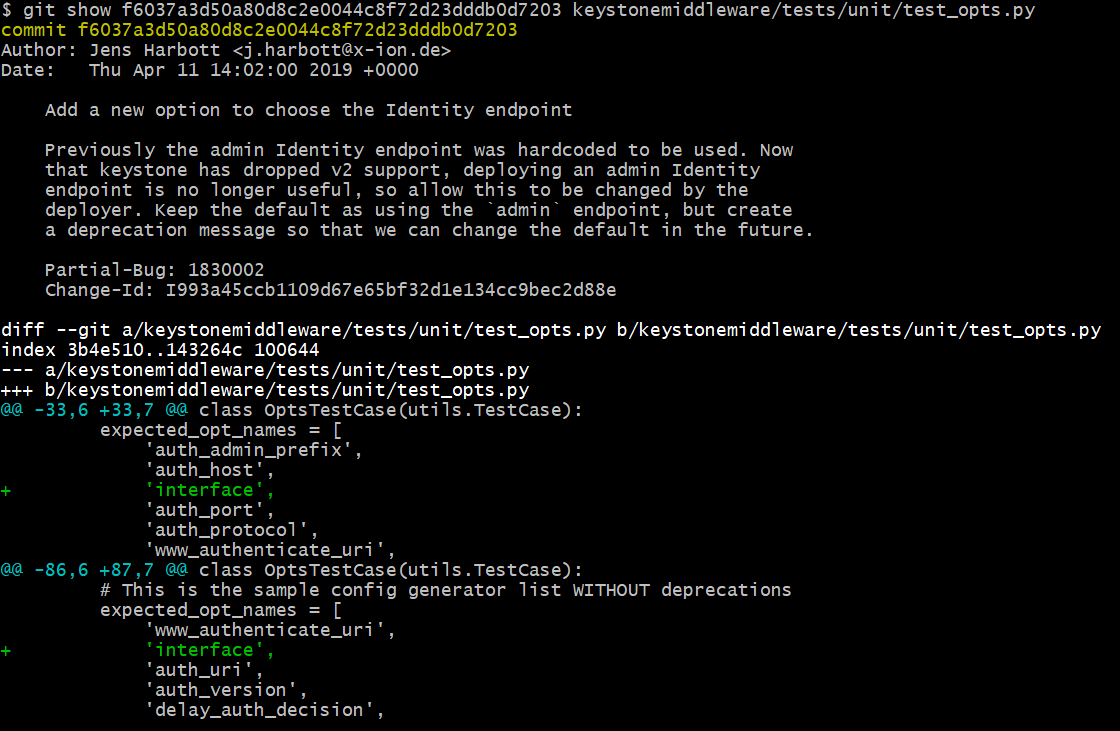
git log --pretty=oneline 文件名

文件名是文件路径+文件名，输入完整



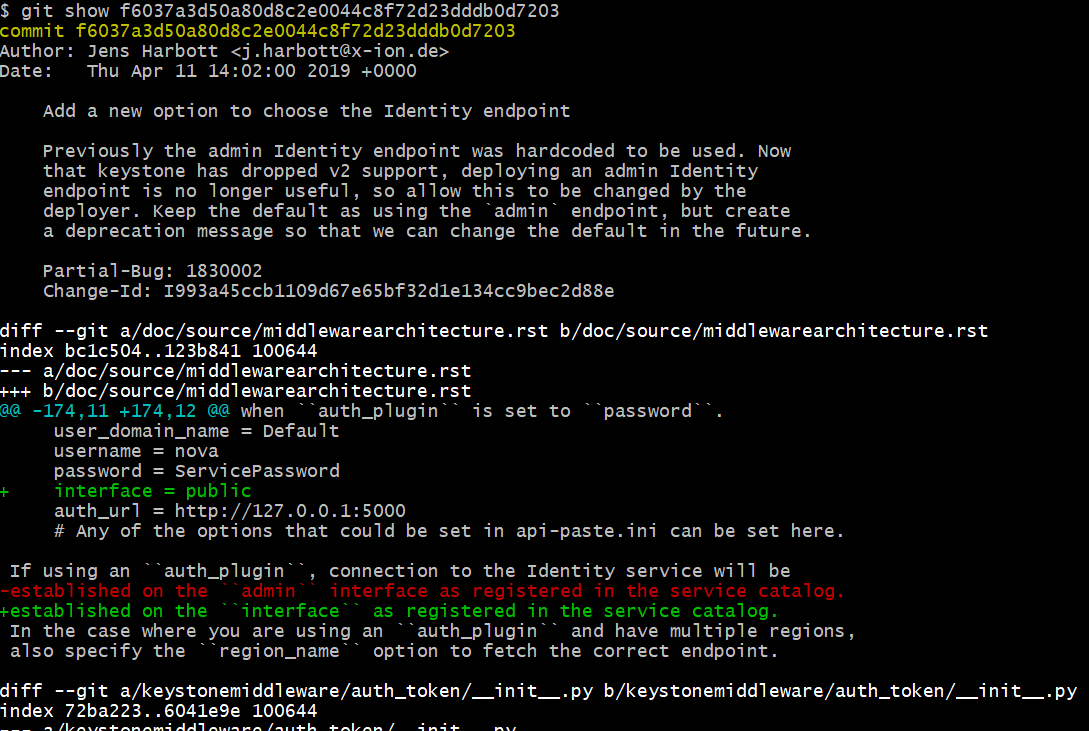
查看某次提交中，文件的改变内容：

git show <git提交版本号> <文件名>



查看这次提交中，所有文件的改变记录：

git show <git提交版本号>



## git cherry-pick解决冲突

应用场景：

对于多分支的代码库，将代码从一个分子转移到另一个分子；这时分为两种情况。一种情况是，你需要另一个分子的所有代码改动，那么就采用合并（git merge）。另一种情况是，只需要改动部分代码改动（某几个分支），这个时候可以采用cherry-pick.

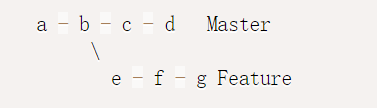
### 基本用法：

git cherry-pick命令的作用，就是将指定的提交（commit）应用于其他分支。

git cherry-pick <commitHash>

上面明ing就会将指定的提交 commitHash，应用于当前分支。这会在当前分支产生一个新的提交，当然他们的hash值是不一样的。

举个例子来说，代码仓库中有master和feature两个分支。



现在将f应用到master分支。

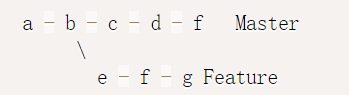
# 切换到master分支

git checkout master

# cherry pick操作

git cherry-pick f

上面操作完成之后，代码库变成下面的样子：



从上面可以看到，master分支的末尾增加了一个提交f。

git cherry-pick命令的参数，不一定是提交的hash值，分支名也是可以的，表示转移该分支的最新提交。

git cherry-pick feature

### 转移多个提交

cherry pick支持一次转移多个提交。

git cherry-pick <HashA> <HashB>

上面命令就是将A和B两个提交应用到当前分支。这会在当前分支生成两个对应的新提交。

如果想要转移一些列的连续提交，可以使用下面的简便语法。

git cherry-pick A..B

上面的命令可以转移A到B的所有提交。他们必须按照正确的顺序放置：提交A必须早于提交B，否则命令将失败，但不会报错。

注意，使用尚明的命令，提交A将不会包含在Cherry pick中。如果要包含提交A，可以使用下面的语法。

git cherry-pick A^..B

### 配置项

git cherry-pick命令常用的配置项如下：

1. –e, --edit

打开外部编辑器，编辑提交信息

1. –n, --no-commit

只更新工作区和暂存区，不产生新的提交。

1. –x

在提交信息的末尾追加一行(cherry picked from commit …), 方便以后查找这个提交是如何产生的。

1. –s, --signoff

在提交信息的末尾追加一行操作者的签名，表示是谁进行了这个操作。

1. –m parent-number, --mainline parent-number

如果原始提交是一个合并节点，来源于两个分支的合并，那么cherry pick默认将失败，因为它不知道应该采用哪个分支的代码改动。

-m配置项告诉git，应该采用哪个分支的变动。它的参数parent-number是一个从1开始的证书，代码原始提交的父分支编号。

git cherry-pick –m 1 <commitHash>

一般来说，1号分支是接受变动的分支(the branch being merged into)，2父号分支是作为变动来源的分支(the branch being merged from)

### 代码冲突

如果操作过程中发生代码冲突，cherry pick会停下来，让用户决定如何继续操作

1. –continue

用户解决代码冲突之后，第一步将修改的文件重新加入暂存区(git add .)

第二步使用下面的命令，让cherry pick过程继续执行

git cherry-pick --continue

1. –abort

发生代码冲突后，放弃合并，回到操作前的样子。

1. –quit

发生代码冲突之后，退出cherry pick，但不是回到操作前的样子。

### 转移到另一个代码库

cherry pick也支持转移到另一个代码库的提交，方法是先将该库添加为远程仓库。

git remote add target git://gitUrl

上面命令添加了一个远程仓库target

然后，将远程代码抓取到本地。

git fetch target

上面命令将远程仓库代码抓取到本地。

接着，检查一下要从远程仓库转移的提交，获取它的Hash值

git log target/master

最后，使用git cherry-pick命令转移提交

git cherry-pick <commitHash>