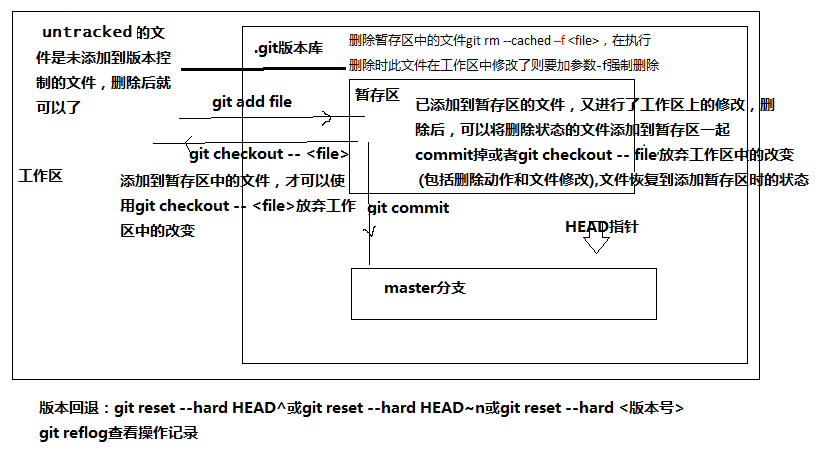
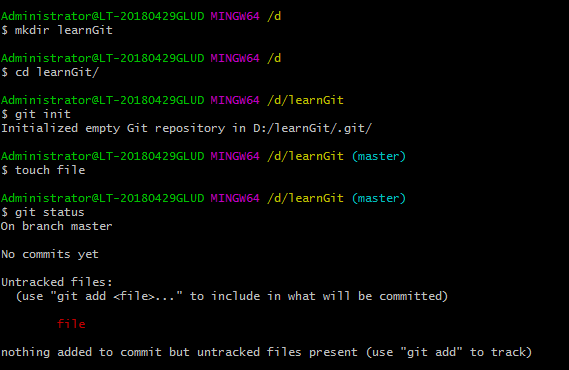
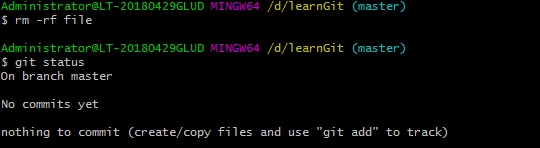
# Git整体结构图



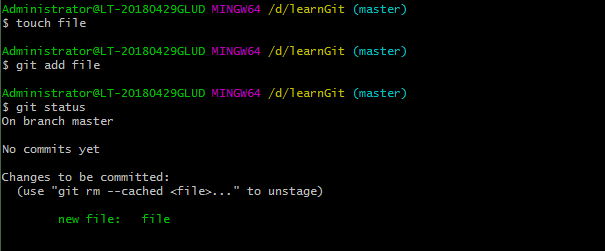
1.git status命令，红色的文件是未添加到版本控制的文件，删除后就可以了



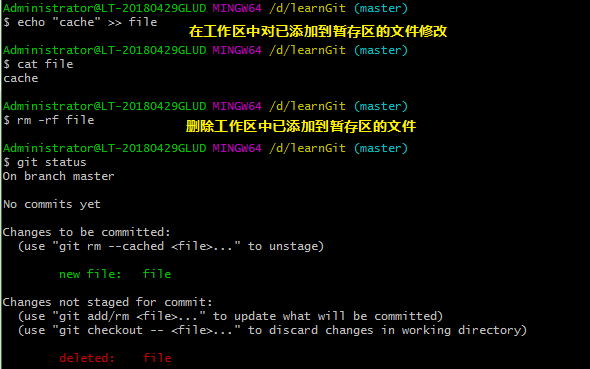
删除未添加到版本控制的文件file后，执行git status如下



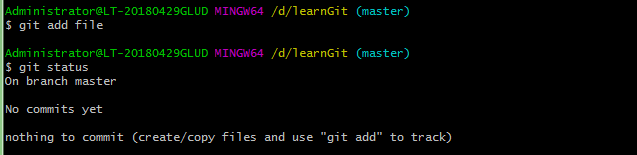
2.☞新建文件file,执行git add file添加到暂存区



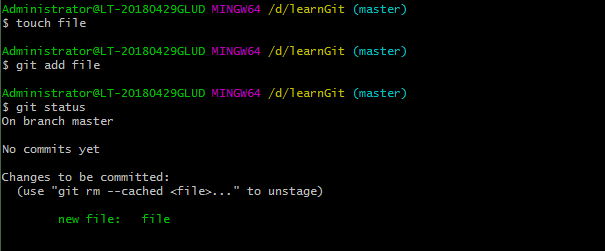
☞在工作区中对file文件进行修改，然后执行删除操作



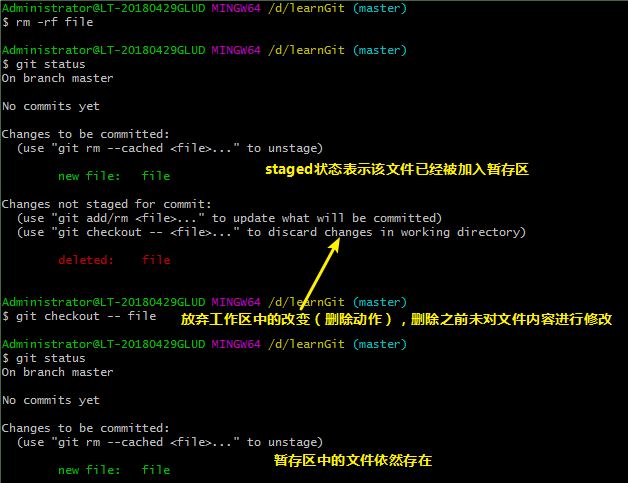
☞将删除状态的文件添加到暂存区，显示无提交，暂存区的文件也没有了。



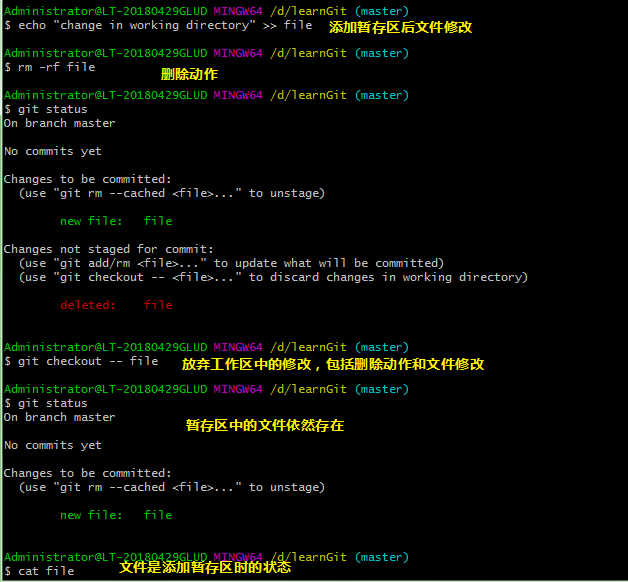
3.☞新建文件file,执行git add file添加到暂存区



☞执行删除操作后使用git checkout -- <file>撤销删除操作，工作区中显示没有要添加到暂存区的文件，但是暂存区的文件依然存在。

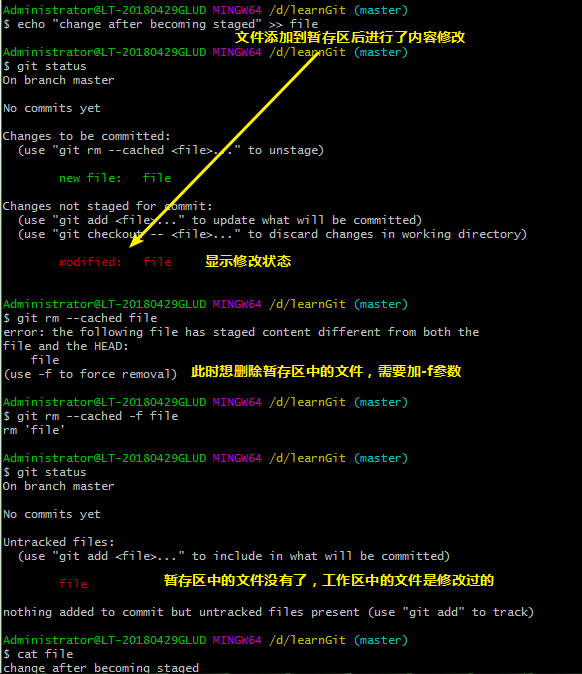


☞添加暂存区后进行了文件的修改并且把文件删除了，执行git checkout -- <file>放弃了工作区中的改变（包括文件修改和删除动作），文件恢复到提交暂存区时的状态



☞文件添加到暂存区后进行了修改，此时想删除添加到暂存区的文件，要加-f参数强制删除git rm --cached –f <file>，命令执行后暂存区中的文件被删除，工作区中的文件是修改后的;

如果文件添加到暂存区后没有进行修改，此时想删除添加到暂存区的文件，那么可以执行git rm --cached file 将暂存区中的文件删除



任何的推送（push）远程仓库的动作都需要在github上有一个存在的仓库

github上添加SSH key,本地电脑设置好相关信息git config --list 可以查看配置信息，如用户名、github注册邮箱

ssh-keygen -t rsa -b 4096 -C "your\_email@example.com" ,然后将C:\Users\Administrator\.ssh\id\_rsa.pub中的内容添加到GitHub上

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

git config --global user.name "Your Name"

## 一、本地创建项目并提交至远程git仓库

1.建一个目录hello

2.cd hello

3.git init

4.echo "hello" >hello.txt

5.git add . 或git add -A 添加所有文件到暂存区

git add命令是个多功能命令，根据目标文件的状态不同，此命令的效果也不同：可以用它开始跟踪新文件，或者把已跟踪的文件放到暂存区，还能用于合并时把有冲突的文件标记为已解决状态。我们需要用git add命令来跟踪新文件，但如果使用git commit -am可以省略使用git add命令将已跟踪文件放到暂存区的功能

6.git commit -m "测试提交"

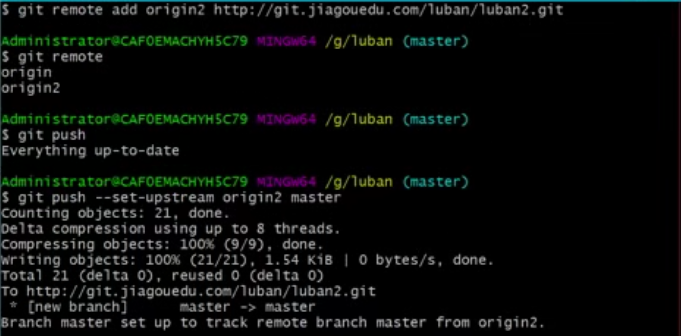
7.在github上创建新的repository,例如hello-world

8.将本地的代码关联到github上，也就是说要在本地添加一个远程仓库地址的别名，具体添加信息保存在本地仓库hello下的.git目录下的config文件中

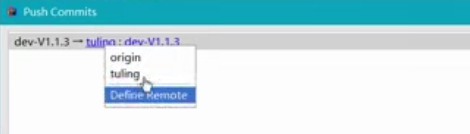
git remote add remoteUrlAlias git@github.com:jikejiangbei/hello-world.git

如果出错了，可以git remote rm/remove remoteUrlAlias删掉重新输入，

如果git remote rm remoteUrlAlias出错，找到.git下的config文件，将里面的[remote "remoteUrlAlias"]及相关信息删除



可以添加多个远程仓库，idea中push时要选择



git remote -v 查看本地配置了哪些远程仓库的地址或git branch --remote -v （刚添加后输入命令可能看不到，pull后可以查看）



git branch 查看本地有哪些分支



git branch -r 查看远程分支名



git branch -a 是查看所有本地分支和配置的远程分支



git branch test 创建本地test分支

git push remoteUrlAlias test 将本地test分支推送到远程服务器

git push --set-upstream remoteUrlAlias test（远程仓库的特定分支） 将提交推送到特定的分支上或git push -u remoteUrlAlias test即使远程没有你要关联的分支，它也会自动创建一个出来，以实现关联

\*\* git push remoteUrlAlias :test 删除远程的分支 （相当于用本地空分支，即冒号前的分支，替换远程的test分支）或git push remoteUrlAlias --delete test

git clone <远程仓库的地址> [指定与远程仓库工程名不同的文件夹名]

## 二、git pull 失败

提示：fatal: refusing to merge unrelated histories因为两个根本不相干的 git 库， 一个是本地库， 一个是远端库，远端觉得这个本地库跟自己不相干， 所以告知无法合并

第一种方法： 从远端库拉下来代码，本地要加入的代码放到远端库的副本（即下载到本地的库）中，然后提交上去，因为这样的话，你基于的库就是远端的库，这是一次update

第二种方法：使用这个强制的方法，会有一次merge的提交

git pull <远程仓库地址别名> <远程master> --allow-unrelated-histories

## 三、git push 命令的使用

git push <远程主机别名> <本地分支名>:<远程分支名>

这个是push的完整写法，将本地分支上传到远程分支，例如：

git push remoteUrlAlias dev:dev

git push <远程主机名> <本地分支名>

如果省略了<远程分支名> 即：

git push remoteUrlAlias dev

则git会push到远程分支的同名本地分支，即和

git push remoteUrlAlias dev:dev等价（红色部分为远程分支）。如果远程分支dev不存在则会创建dev分支。

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

git push <远程主机名> :<远程分支名>

如果省略<本地分支名> 即

git push :dev

则git会删除远程主机上的dev分支，即用一个空分支更新dev分支，相当于删除dev分支，和

git push remoteUrlAlias --delete dev 等价。

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

git push <远程主机名>

如果当前分支和远程分支存在追踪关系，则本地分支和远程分支都可以省略。

git push

如果当前分支只有一个远程分支，那么远程主机名也可以省略，可以使用

git push -u

如果当前分支和多个主机之间存在追踪关系，可以使用这个命令来设置一个默认的主机，例如：

git push -u origin master

设置后可以使用git push来推送到默认的主机上，如果要推送到其他远程主机上则要写全。

git push –all <远程主机>

将本地的所有分支都推送到远程主机。

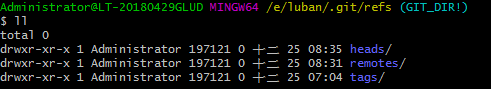
git push –force <远程主机>

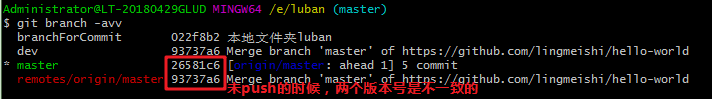
当服务器上版本和本地版本不一致时，push会发生冲突，git会提示你先使用pull来解决冲突，如果你一定要推送，可以使用这个命令，除非你一定要这样，否则应该避免使用force。

## 四、分支的操作

本地的.git文件夹里面对应存储了git本地仓库master分支的commit ID和跟踪的远程分支orign/master的commit ID（可以有多个远程仓库）,没push之前这两类分支的版本可能是不一样的。heads就是本地分支,remotes是跟踪的

远程分支





$ git branch 不加参数查看本地分支

$ git branch –av 查看所有分支

$ git branch –avv 查看所有分支详细信息，包括本地分支关联远程分支的信息



$ git checkout <origin远程仓库别名>/<master跟踪的远程分支> 切到跟踪的远程分支

$ git checkout –b test 创建并切换分支，相当于两条命令git branch test和git checkout test

$ git merge test 在当前分支（如master）上合并指定分支test

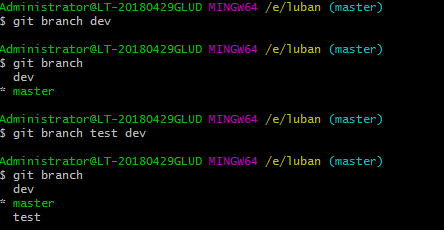
Fast-forward表明这次合并是‘快进模式’，直接把master指向test的当前提交，合并完后可以删除test分支了，

git branch –d test

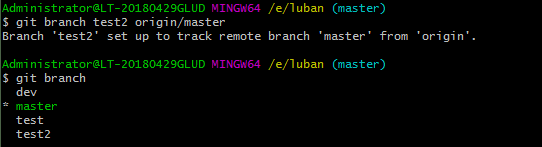
$ git branch -D test 删除本地分支，如果我们要删除一个已经合并了的分支，则参数D要改成小写的d

### 创建分支的四种方式

#### 1.基于本地分支 创建一个分支



#### 2.基于远程分支 创建一个分支

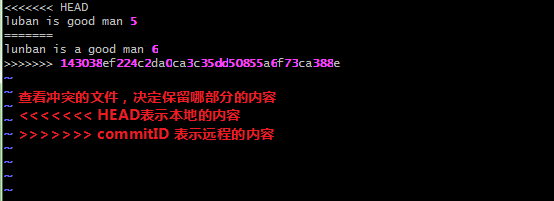


#### 3.基于一个提交 创建一个分支

#### 

## 五：如何解决冲突





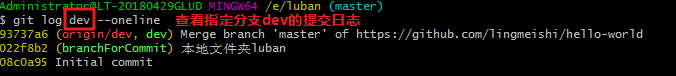


在当前分支（如master）上合并指定分支test

## 六、日志管理

$ git log

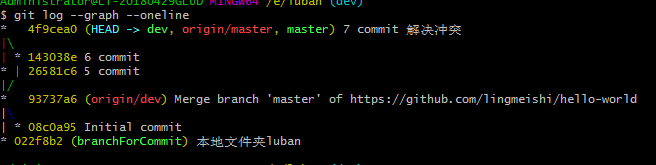








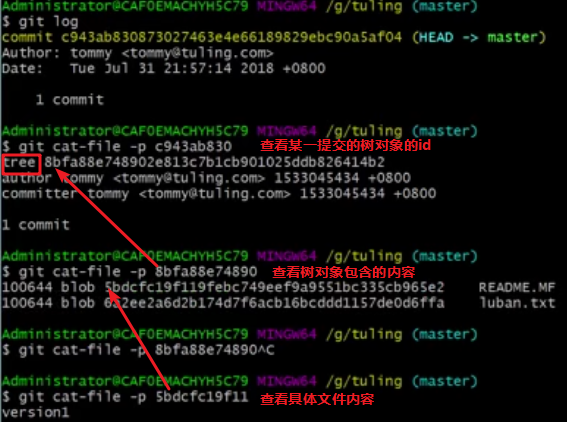
查看项目的提交网络



# Git底层原理

查看某一对象的命令

git cat-file –p <id>





* GIT存储对像
* GIT树对像
* GIT提交对像
* GIT引用

1、GIT存储对像(hashMap)

Git 是一个内容寻址文件系统，其核心部分是一个简单的键值对数据库（key-value data store），你可以向数据库中插入任意内容，它会返回一个用于取回该值的hash 键。

# git 键值库中插入数据

echo 'luban is good man' | git **hash-object** -w --stdin

79362d07cf264f8078b489a47132afbc73f87b9a

#基于键获取指定内容

git cat-file -p 79362d07cf264f8078b489a47132afbc73f87b9a

Git基于该功能 把每个文件的版本中内容都保存在数据库中，当要进行版本回滚的时候就通过其中一个键将其取回并替换。

* 模拟演示git 版写入与回滚过程

# 查找所有的git 对像

  find .git/objects/ -type f

# 写入版本1

echo 'version1' > README.MF; git hash-object -w README.MF;

# 写入版本2

echo 'version2' > README.MF; git hash-object -w README.MF;

# 写入版本3

echo 'version3' > README.MF; git hash-object -w README.MF;

# 回滚指定版本

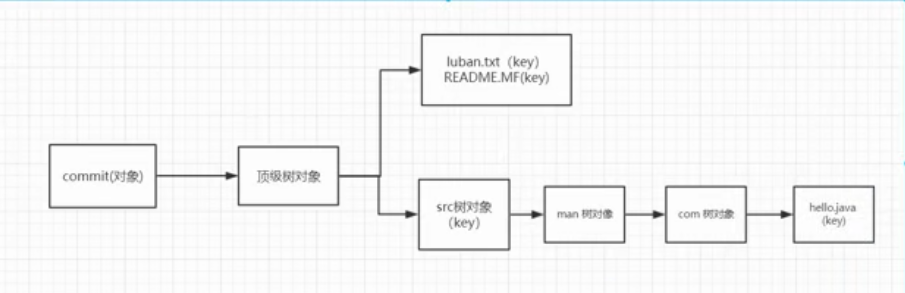
git cat-file -p c11e96db44f7f3bc4c608aa7d7cd9ba4ab25066e > README.MF

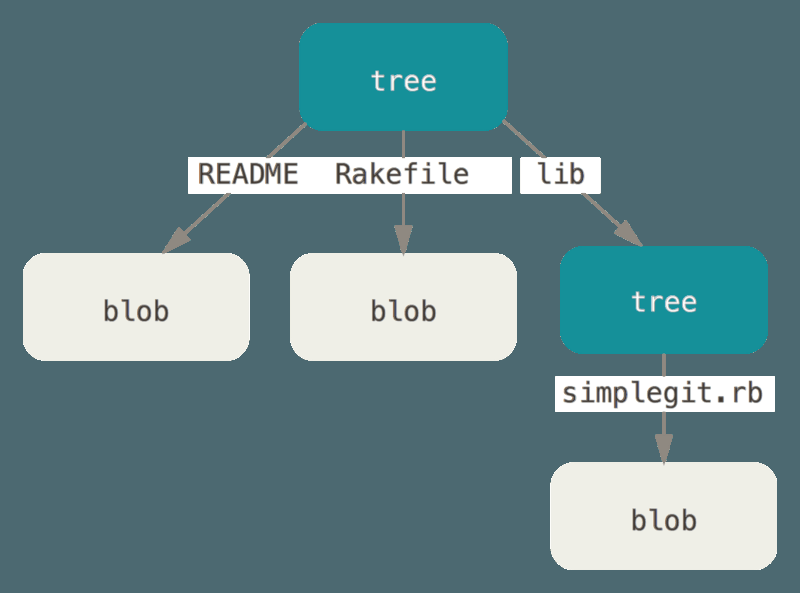
所以我们平常用的 git add 其实就是把修改之后的内容 插入到键值库（即硬盘上的.git/objects/）中。当我们执行 git add README.MF 等同于执行了 git hash-object -w README.MF 把文件写到数据库中。

我们解决了存储的问题，但其只能存储内容同并没有存储文件名，如果要进行回滚 怎么知道哪个内容对应哪个文件呢？接下要讲的就是树对象，它解决了文件名存储的问题 。

2、GIT树对像

树对象解决了文件名的问题，它的目的将多个文件名组织在一起，其包含多个文件名称与其对应的Key和其它树对象的用引用，可以理解成操作系统当中的文件夹，一个文件夹包含多个文件和多个其它文件夹。





每一个分支当中都关联了一个树对像，他存储了当前分支下所有的文件名及对应的 key.

通过以下命令即可查看

#查看分支树

git cat-file -p master^{tree}

3、git提交对象

一次提交即为当前版本的一个快照，该快照就是通过提交对像保存，其存储的内容为：一个顶级树对象、上一次提交的对像啥希、提交者用户名及邮箱、提交时间戳、提交评论。

$ git cat-file -p b2395925b5f1c12bf8cb9602f05fc8d580311836

tree 002adb8152f7cd49f400a0480ef2d4c09b060c07

parent 8be903f5e1046b851117a21cdc3c80bdcaf97570

author tommy <tommy@tuling.com> 1532959457 +0800

committer tommy <tommy@tuling.com> 1532959457 +0800

通过上面的知识，我们可以推测出从修改一个文件到提交的过程总共生成了三个对像：

一个内容对象 ==> 存储了文件内容

一个树对像 ==> 存储了文件名及内容对像的key

一个提交对像 ==> 存储了树对像的key 及提交评论。

* 演示文件提交过程

4、GIT引用

当我们执行 git branch {branchName} 时创建了一个分支，其本质就是在git 基于指定提交创建了一个引用文件，保存在 .git\refs\heads\ 下。

* 演示分支的创建

 git branch dev

cat.git\refs\heads\dev

git 总共 有三种类型的引用：

1. 分支引用
2. 远程分支引用
3. 标签引用

#查询比较两个版本

git log master..experiment

#版本提交历史网络

git log --pretty=format:'%h %s' --graph

#查看分支树

git cat-file -p master^{tree}

# $ git push -u origin master 推送到远程库出错

1、错误描述：

出错场景描述：

github上创建远程仓库的时候选择添加README.md文件，

git bash连接远程库，然后执行push操作，出现下面的问题

$ git push -u origin master

To github.com:XXXX/DesignPatterns.git

 ! [rejected]        master -> master (fetch first)

error: failed to push some refs to 'git@github.com:XXXX/DesignPatterns.git'

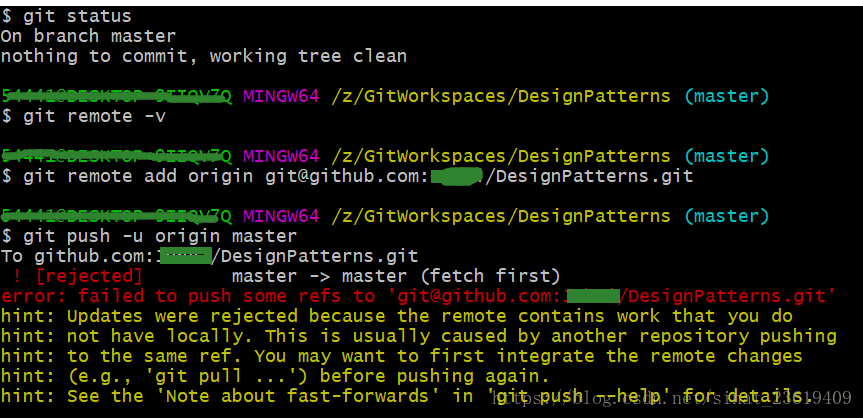
hint: Updates were rejected because the remote contains work that you do

hint: not have locally. This is usually caused by another repository pushing

hint: to the same ref. You may want to first integrate the remote changes

hint: (e.g., 'git pull ...') before pushing again.

hint: See the 'Note about fast-forwards' in 'git push --help' for details.



2、问题分析：

出现错误的主要原因是github中的README.md文件不在本地代码目录中。

3、解决方法：

（1）可以通过如下命令进行代码合并【注：pull=fetch+merge】

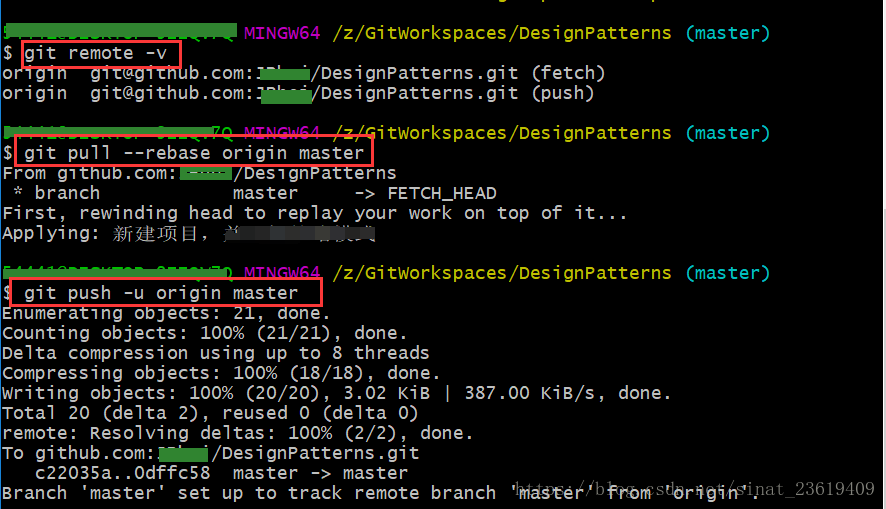
            git pull --rebase origin master

（2）再执行语句：

         git push -u origin master

       即可完成代码上传到github。

去github上即可看到上传的项目



# 其他右键-》Options

# 