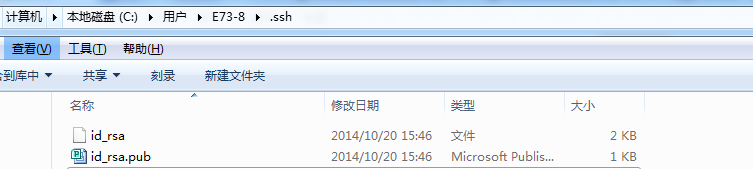
**Github使用教程**

     在了解之前，先注册github账号，由于你的本地Git仓库和github仓库之间的传输是通过SSH加密的，所以需要一点设置：

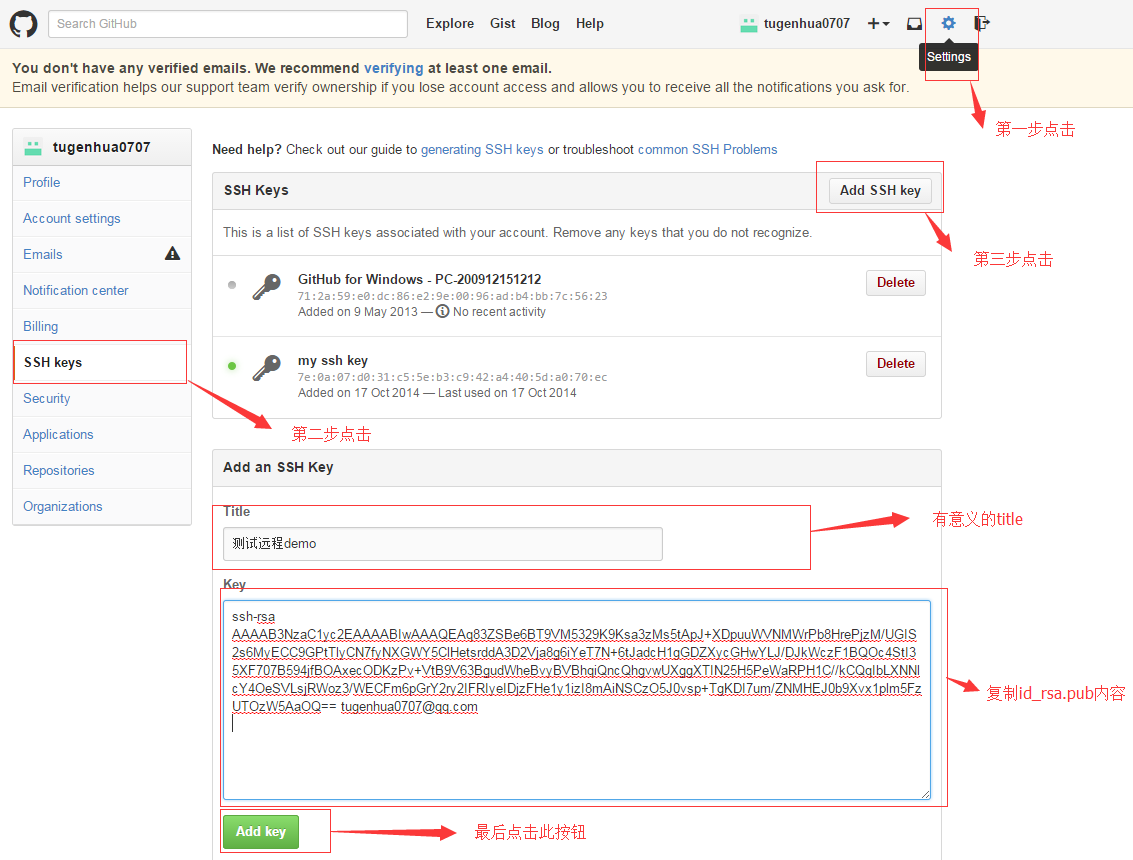
     第一步：创建SSH Key。在用户主目录下，看看有没有.ssh目录，如果有，再看看这个目录下有没有id\_rsa和id\_rsa.pub这两个文件，如果有的话，直接跳过此如下命令，如果没有的话，打开命令行，输入如下命令：

ssh-keygen  -t rsa –C “youremail@example.com”, 由于我本地此前运行过一次，所以本地有，如下所示：

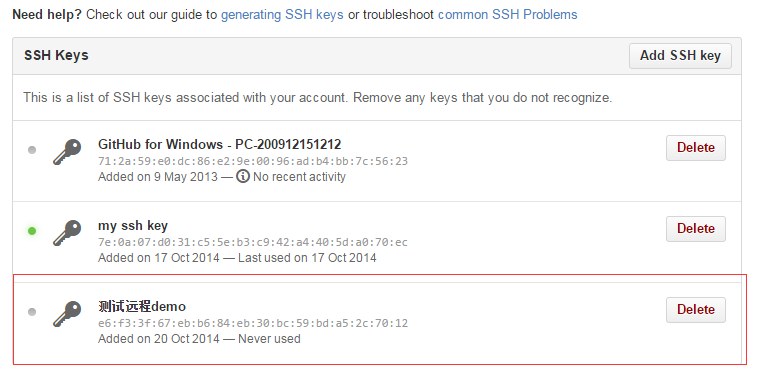


id\_rsa是私钥，不能泄露出去，id\_rsa.pub是公钥，可以放心地告诉任何人。

第二步：登录github,打开” settings”中的SSH Keys页面，然后点击“Add SSH Key”,填上任意title，在Key文本框里黏贴id\_rsa.pub文件的内容。



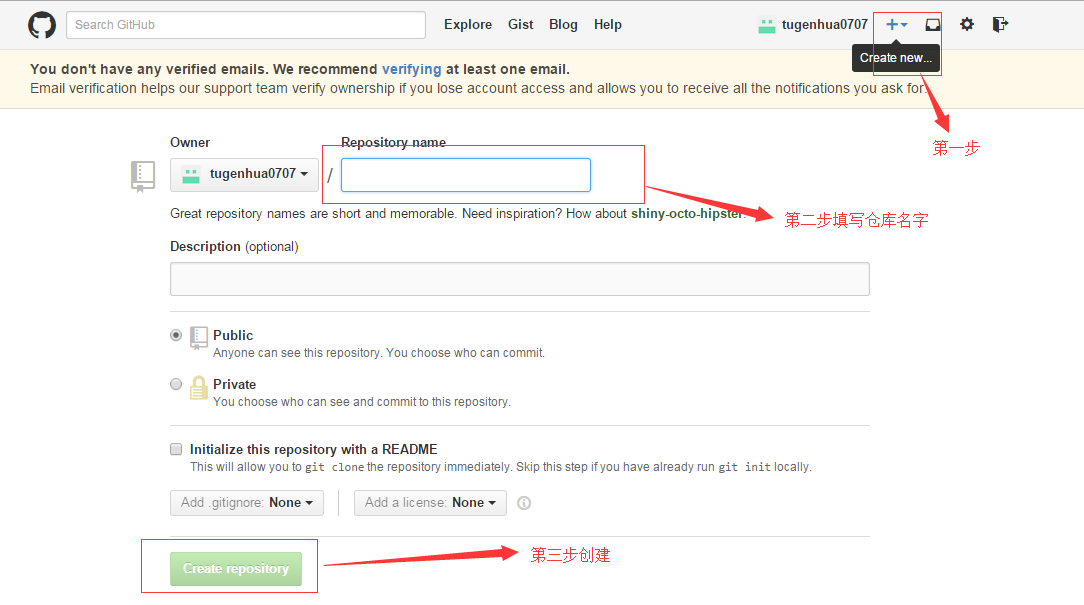
点击 Add Key，你就应该可以看到已经添加的key。



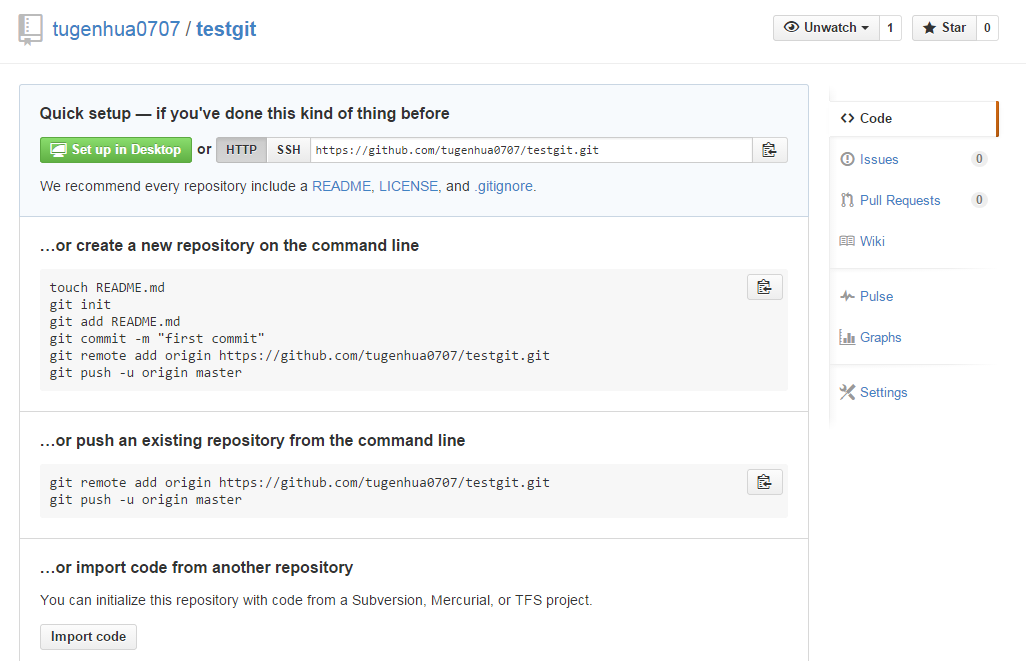
1. 如何添加远程库？

         现在的情景是：我们已经在本地创建了一个Git仓库后，又想在github创建一个Git仓库，并且希望这两个仓库进行远程同步，这样github的仓库可以作为备份，又可以其他人通过该仓库来协作。

   首先，登录github上，然后在右上角找到“create a new repo”创建一个新的仓库。如下：



在Repository name填入testgit，其他保持默认设置，点击“Create repository”按钮，就成功地创建了一个新的Git仓库：

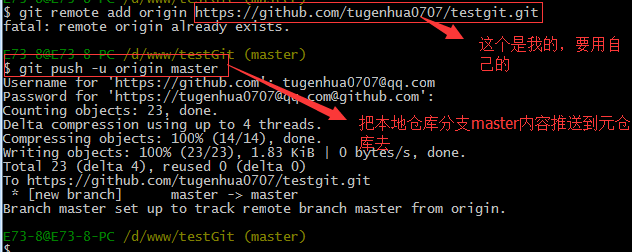


    目前，在GitHub上的这个testgit仓库还是空的，GitHub告诉我们，可以从这个仓库克隆出新的仓库，也可以把一个已有的本地仓库与之关联，然后，把本地仓库的内容推送到GitHub仓库。

现在，我们根据GitHub的提示，在本地的testgit仓库下运行命令：

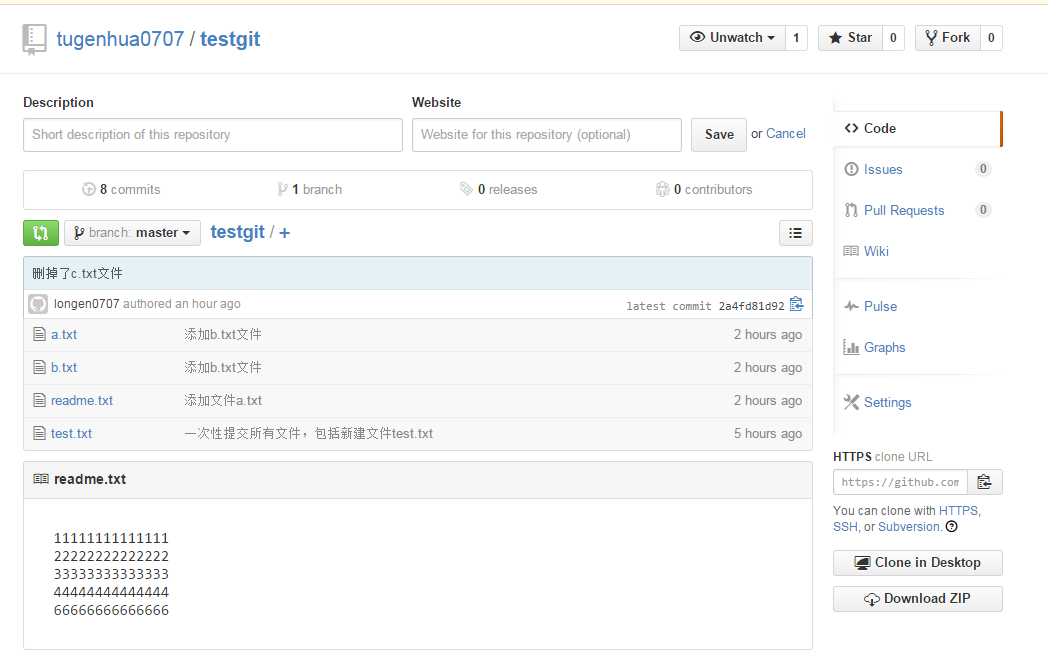
git remote add origin https://github.com/tugenhua0707/testgit.git

所有的如下：



把本地库的内容推送到远程，使用 git push命令，实际上是把当前分支master推送到远程。

由于远程库是空的，我们第一次推送master分支时，加上了 –u参数，Git不但会把本地的master分支内容推送的远程新的master分支，还会把本地的master分支和远程的master分支关联起来，在以后的推送或者拉取时就可以简化命令。推送成功后，可以立刻在github页面中看到远程库的内容已经和本地一模一样了，上面的要输入github的用户名和密码如下所示：



从现在起，只要本地作了提交，就可以通过如下命令：

git push origin master

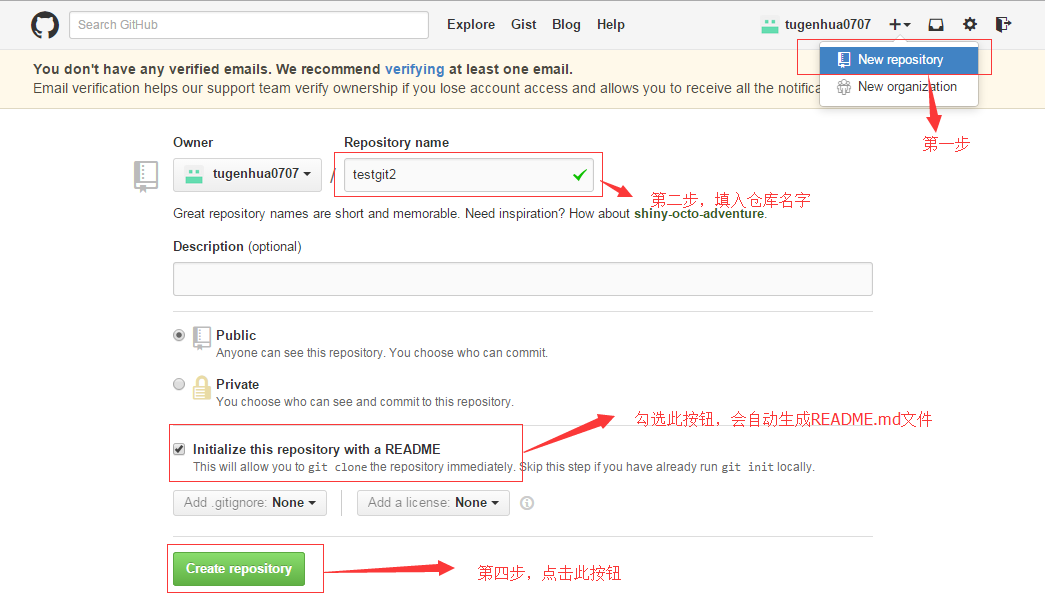
把本地master分支的最新修改推送到github上了，现在你就拥有了真正的分布式版本库了。

2. 如何从远程库克隆？

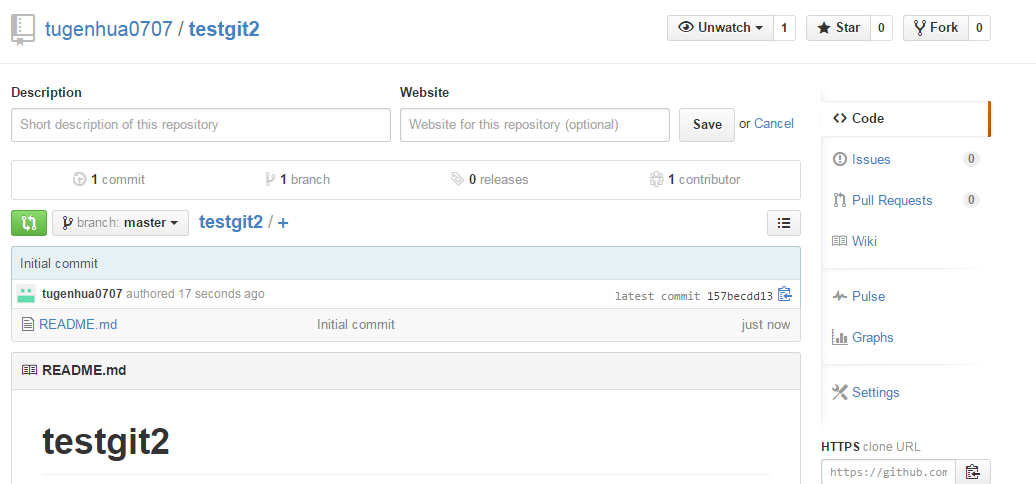
  上面我们了解了先有本地库，后有远程库时候，如何关联远程库。

  现在我们想，假如远程库有新的内容了，我想克隆到本地来 如何克隆呢？

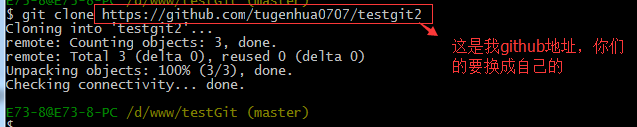
  首先，登录github，创建一个新的仓库，名字叫testgit2.如下：



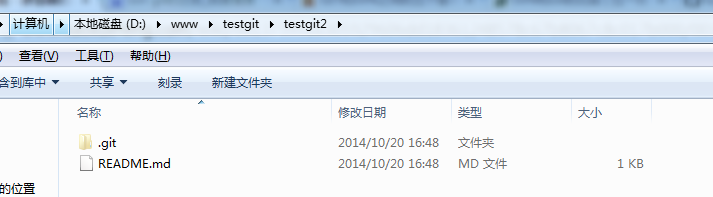
如下，我们看到：



现在，远程库已经准备好了，下一步是使用命令git clone克隆一个本地库了。如下所示：



接着在我本地目录下 生成testgit2目录了，如下所示：



**创建与合并分支。**

    在  版本回填退里，你已经知道，每次提交，Git都把它们串成一条时间线，这条时间线就是一个分支。截止到目前，只有一条时间线，在Git里，这个分支叫主分支，即master分支。HEAD严格来说不是指向提交，而是指向master，master才是指向提交的，所以，HEAD指向的就是当前分支。

首先，我们来创建dev分支，然后切换到dev分支上。如下操作：



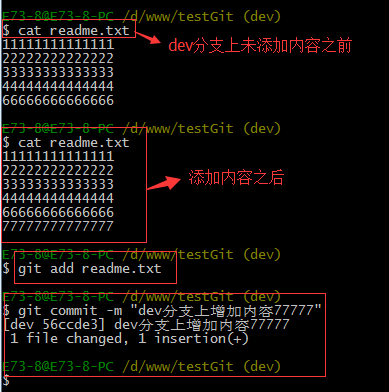
git checkout 命令加上 –b参数表示创建并切换，相当于如下2条命令

git branch dev

git checkout dev

git branch查看分支，会列出所有的分支，当前分支前面会添加一个星号。然后我们在dev分支上继续做demo，比如我们现在在readme.txt再增加一行 7777777777777

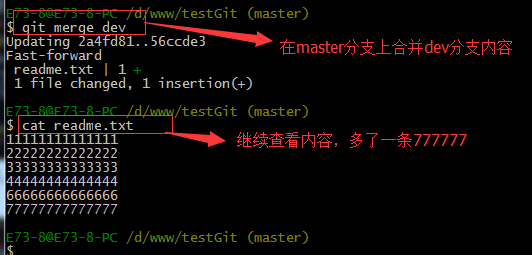
首先我们先来查看下readme.txt内容，接着添加内容77777777，如下：



现在dev分支工作已完成，现在我们切换到主分支master上，继续查看readme.txt内容如下：



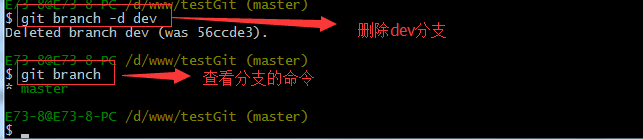
现在我们可以把dev分支上的内容合并到分支master上了，可以在master分支上，使用如下命令 git merge dev 如下所示：



git merge命令用于合并指定分支到当前分支上，合并后，再查看readme.txt内容，可以看到，和dev分支最新提交的是完全一样的。

注意到上面的Fast-forward信息，Git告诉我们，这次合并是“快进模式”，也就是直接把master指向dev的当前提交，所以合并速度非常快。

合并完成后，我们可以接着删除dev分支了，操作如下：



总结创建与合并分支命令如下：

   查看分支：git branch

   创建分支：git branch name

   切换分支：git checkout name

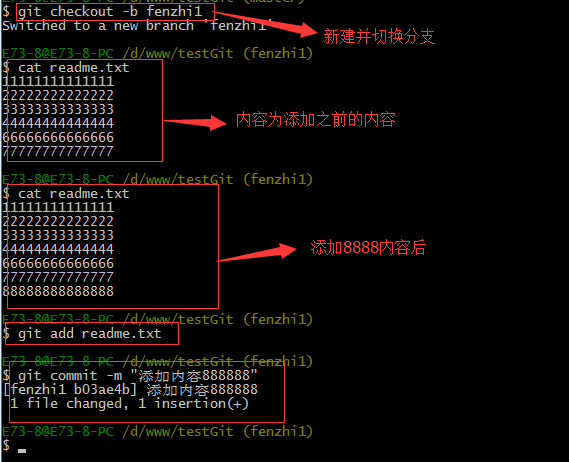
创建+切换分支：git checkout –b name

合并某分支到当前分支：git merge name

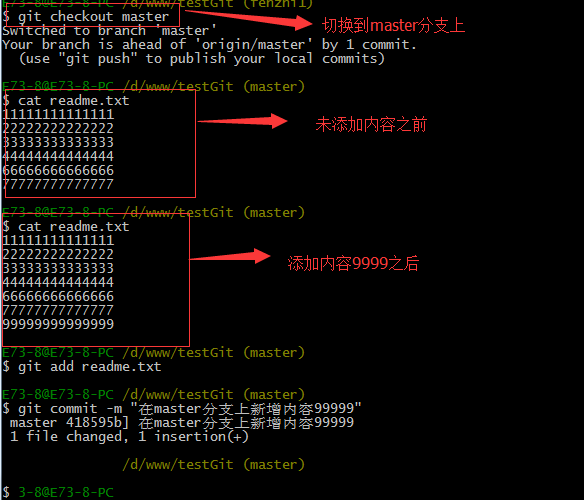
删除分支：git branch –d name

1. 如何解决冲突？

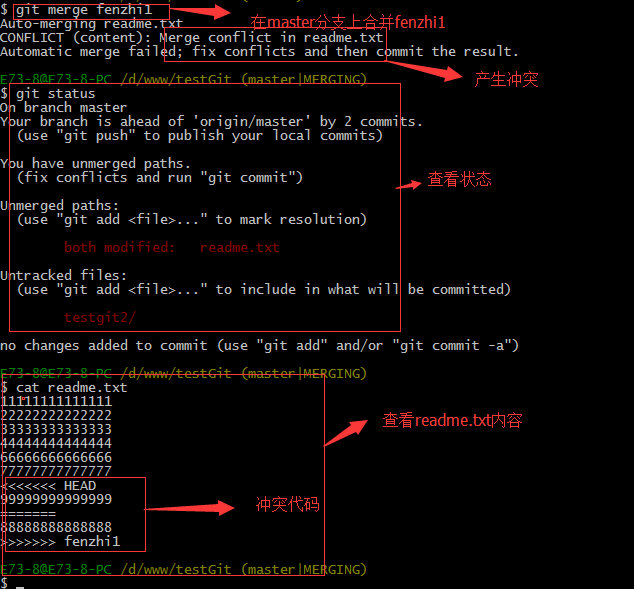
下面我们还是一步一步来，先新建一个新分支，比如名字叫fenzhi1，在readme.txt添加一行内容8888888，然后提交，如下所示：



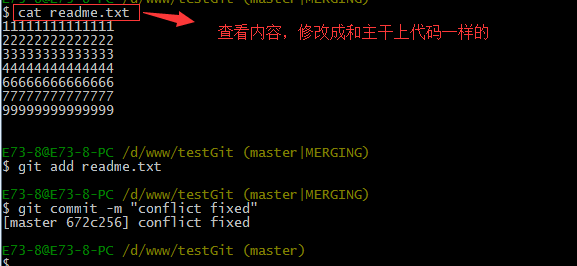
同样，我们现在切换到master分支上来，也在最后一行添加内容，内容为99999999，如下所示：



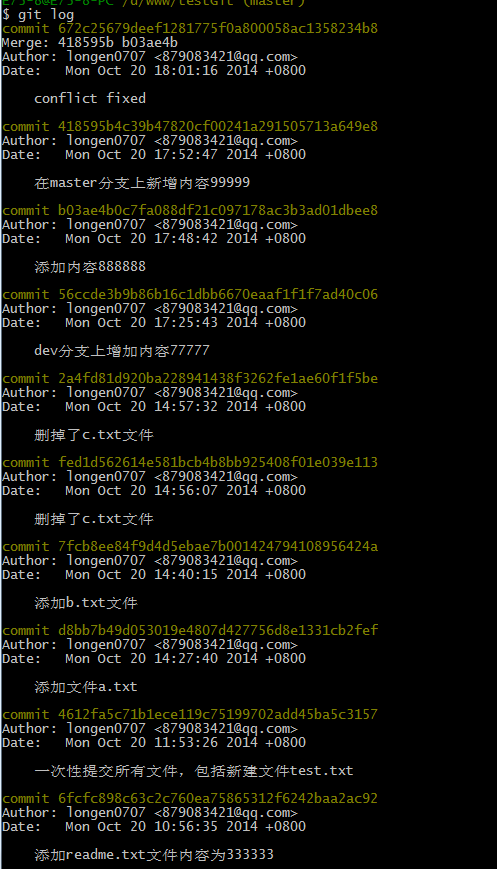
现在我们需要在master分支上来合并fenzhi1，如下操作：



Git用<<<<<<<，=======，>>>>>>>标记出不同分支的内容，其中<<<HEAD是指主分支修改的内容，>>>>>fenzhi1 是指fenzhi1上修改的内容，我们可以修改下如下后保存：



如果我想查看分支合并的情况的话，需要使用命令 git log.命令行演示如下：

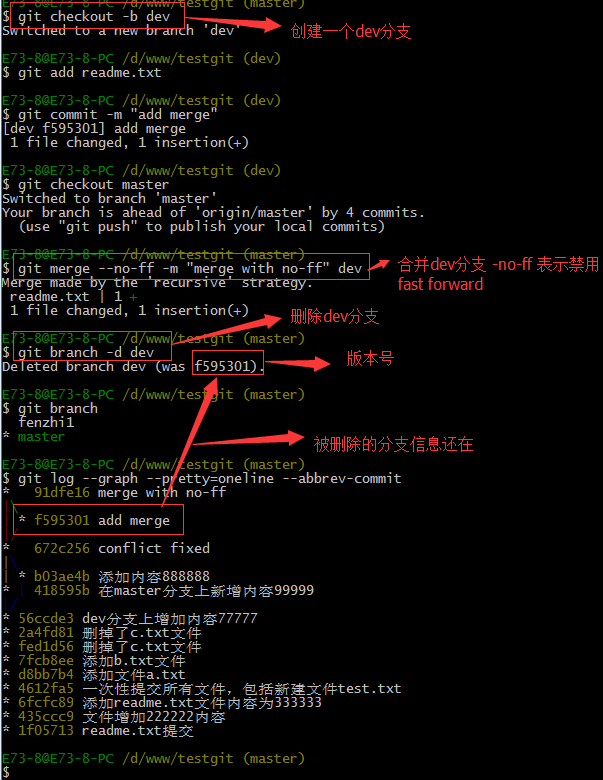


    3.分支管理策略。

      通常合并分支时，git一般使用”Fast forward”模式，在这种模式下，删除分支后，会丢掉分支信息，现在我们来使用带参数 –no-ff来禁用”Fast forward”模式。首先我们来做demo演示下：

1. 创建一个dev分支。
2. 修改readme.txt内容。
3. 添加到暂存区。
4. 切换回主分支(master)。
5. 合并dev分支，使用命令 git merge –no-ff  -m “注释” dev
6. 查看历史记录

截图如下：

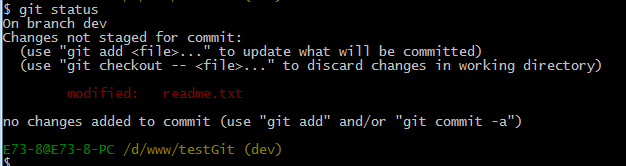


**分支策略：**首先master主分支应该是非常稳定的，也就是用来发布新版本，一般情况下不允许在上面干活，干活一般情况下在新建的dev分支上干活，干完后，比如上要发布，或者说dev分支代码稳定后可以合并到主分支master上来。

**bug分支：**

     在开发中，会经常碰到bug问题，那么有了bug就需要修复，在Git中，分支是很强大的，每个bug都可以通过一个临时分支来修复，修复完成后，合并分支，然后将临时的分支删除掉。

比如我在开发中接到一个404 bug时候，我们可以创建一个404分支来修复它，但是，当前的dev分支上的工作还没有提交。比如如下：

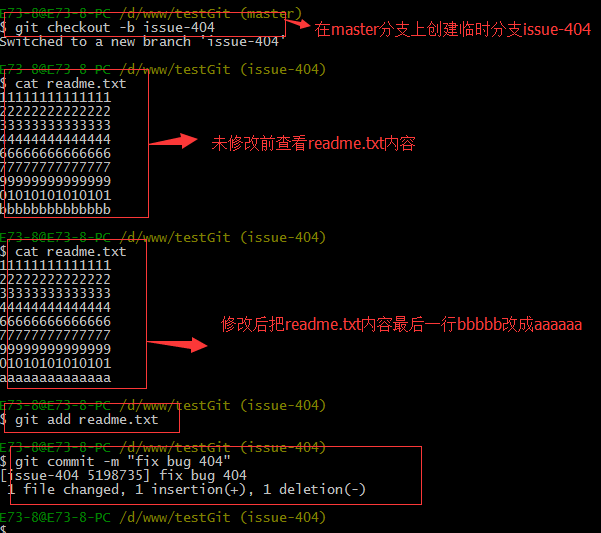


   并不是我不想提交，而是工作进行到一半时候，我们还无法提交，比如我这个分支bug要2天完成，但是我issue-404 bug需要5个小时内完成。怎么办呢？还好，Git还提供了一个stash功能，可以把当前工作现场 ”隐藏起来”，等以后恢复现场后继续工作。如下：

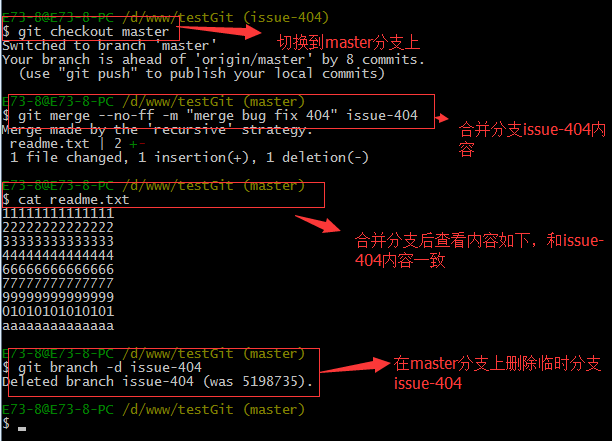


   所以现在我可以通过创建issue-404分支来修复bug了。

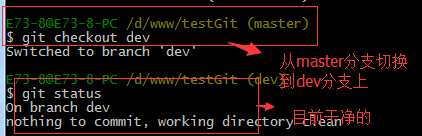
首先我们要确定在那个分支上修复bug，比如我现在是在主分支master上来修复的，现在我要在master分支上创建一个临时分支，演示如下：



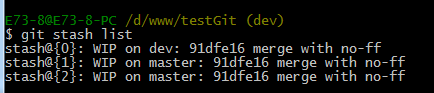
修复完成后，切换到master分支上，并完成合并，最后删除issue-404分支。演示如下：



现在，我们回到dev分支上干活了。



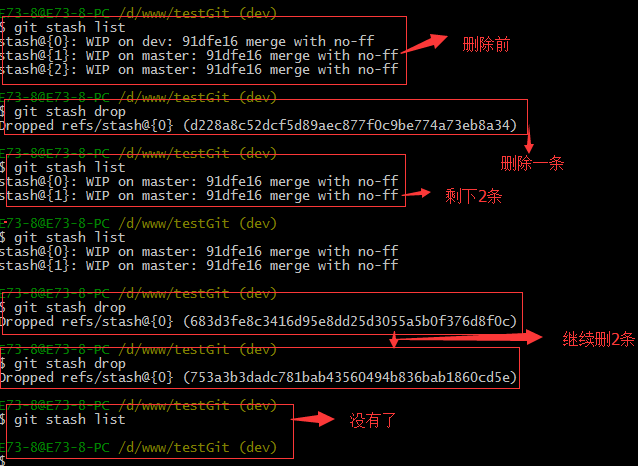
工作区是干净的，那么我们工作现场去哪里呢？我们可以使用命令 git stash list来查看下。如下：



工作现场还在，Git把stash内容存在某个地方了，但是需要恢复一下，可以使用如下2个方法：

1. git stash apply恢复，恢复后，stash内容并不删除，你需要使用命令git stash drop来删除。
2. 另一种方式是使用git stash pop,恢复的同时把stash内容也删除了。

         演示如下

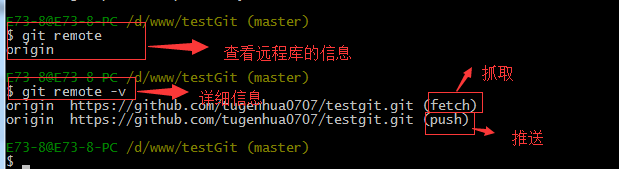


**多人协作。**

当你从远程库克隆时候，实际上Git自动把本地的master分支和远程的master分支对应起来了，并且远程库的默认名称是origin。

1. 要查看远程库的信息 使用 git remote
2. 要查看远程库的详细信息 使用 git remote –v

如下演示：



**一：推送分支：**

      推送分支就是把该分支上所有本地提交到远程库中，推送时，要指定本地分支，这样，Git就会把该分支推送到远程库对应的远程分支上：

      使用命令 git push origin master

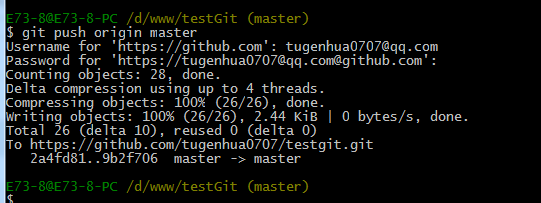
比如我现在的github上的readme.txt代码如下：



本地的readme.txt代码如下：



现在我想把本地更新的readme.txt代码推送到远程库中，使用命令如下：



我们可以看到如上，推送成功，我们可以继续来截图github上的readme.txt内容 如下：



可以看到 推送成功了，如果我们现在要推送到其他分支，比如dev分支上，我们还是那个命令 git push origin dev

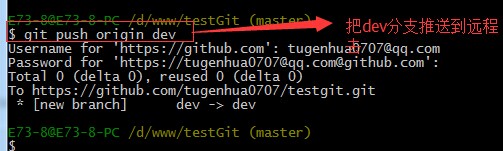
那么一般情况下，那些分支要推送呢？

1. master分支是主分支，因此要时刻与远程同步。
2. 一些修复bug分支不需要推送到远程去，可以先合并到主分支上，然后把主分支master推送到远程去。

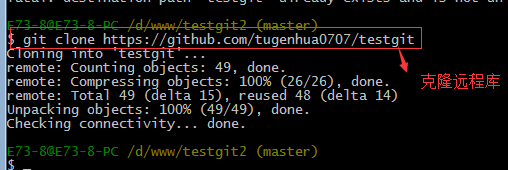
**二：抓取分支：**

多人协作时，大家都会往master分支上推送各自的修改。现在我们可以模拟另外一个同事，可以在另一台电脑上（注意要把SSH key添加到github上）或者同一台电脑上另外一个目录克隆，新建一个目录名字叫testgit2

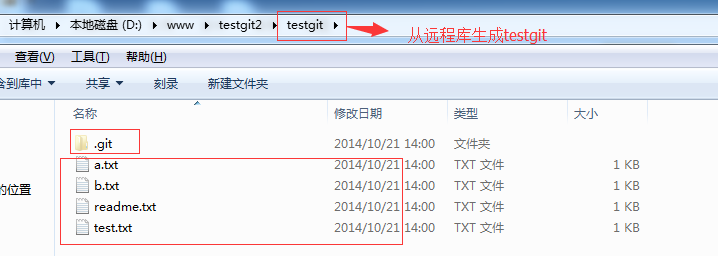
但是我首先要把dev分支也要推送到远程去，如下



接着进入testgit2目录，进行克隆远程的库到本地来，如下：



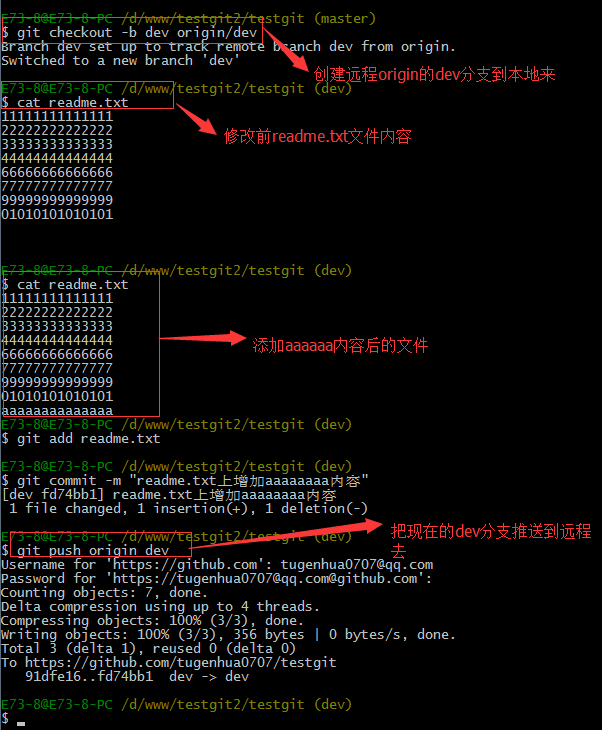
现在目录下生成有如下所示：



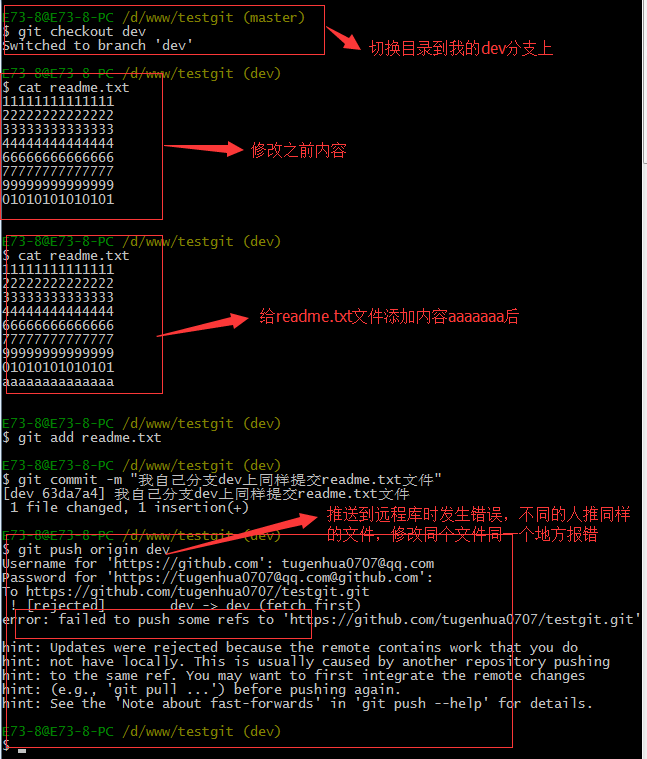
现在我们的小伙伴要在dev分支上做开发，就必须把远程的origin的dev分支到本地来，于是可以使用命令创建本地dev分支：git checkout  –b dev origin/dev

现在小伙伴们就可以在dev分支上做开发了，开发完成后把dev分支推送到远程库时。

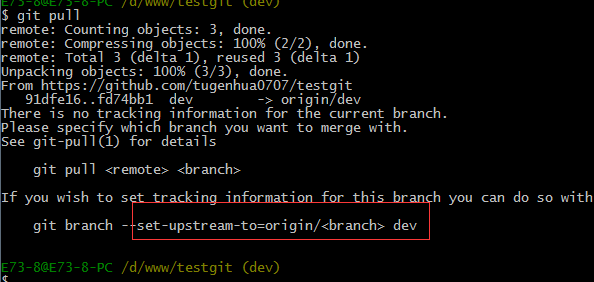
如下：



小伙伴们已经向origin/dev分支上推送了提交，而我在我的目录文件下也对同样的文件同个地方作了修改，也试图推送到远程库时，如下：



由上面可知：推送失败，因为我的小伙伴最新提交的和我试图推送的有冲突，解决的办法也很简单，上面已经提示我们，先用git pull把最新的提交从origin/dev抓下来，然后在本地合并，解决冲突，再推送。

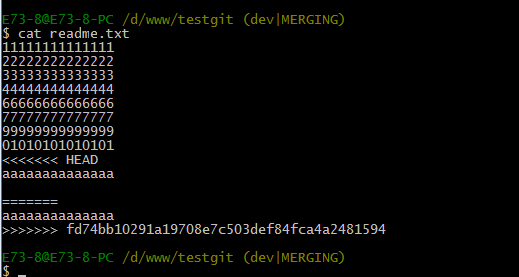


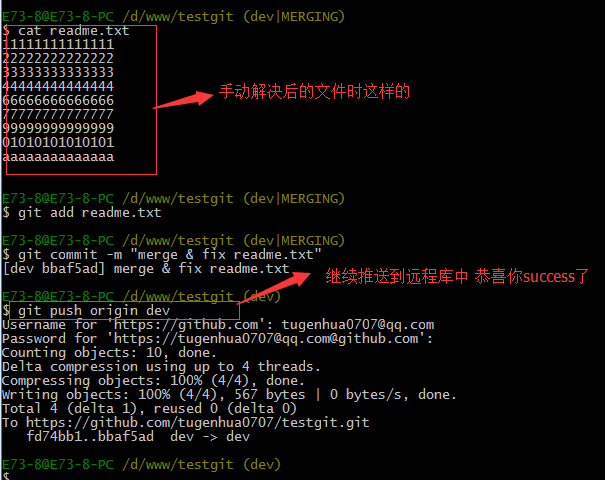
git pull也失败了，原因是没有指定本地dev分支与远程origin/dev分支的链接，根据提示，设置dev和origin/dev的链接：如下：



这回git pull成功，但是合并有冲突，需要手动解决，解决的方法和分支管理中的 解决冲突完全一样。解决后，提交，再push：

我们可以先来看看readme.txt内容了。



现在手动已经解决完了，我接在需要再提交，再push到远程库里面去。如下所示：  


因此：多人协作工作模式一般是这样的：

1. 首先，可以试图用git push origin branch-name推送自己的修改.
2. 如果推送失败，则因为远程分支比你的本地更新早，需要先用git pull试图合并。
3. 如果合并有冲突，则需要解决冲突，并在本地提交。再用git push origin branch-name推送。

**Git基本常用命令如下：**

   mkdir：         XX (创建一个空目录 XX指目录名)

   pwd：          显示当前目录的路径。

   git init          把当前的目录变成可以管理的git仓库，生成隐藏.git文件。

   git add XX       把xx文件添加到暂存区去。

   git commit –m “XX”  提交文件 –m 后面的是注释。

   git status        查看仓库状态

   git diff  XX      查看XX文件修改了那些内容

   git log          查看历史记录

   git reset  --hard HEAD^ 或者 git reset  --hard HEAD~ 回退到上一个版本

                        (如果想回退到100个版本，使用git reset –hard HEAD~100 )

   cat XX         查看XX文件内容

   git reflog       查看历史记录的版本号id

   git checkout -- XX  把XX文件在工作区的修改全部撤销。

   git rm XX          删除XX文件

   git remote add origin <https://github.com/tugenhua0707/testgit> 关联一个远程库

   git push –u(第一次要用-u 以后不需要) origin master 把当前master分支推送到远程库

   git clone <https://github.com/tugenhua0707/testgit>  从远程库中克隆

   git checkout –b dev  创建dev分支 并切换到dev分支上

   git branch  查看当前所有的分支

   git checkout master 切换回master分支

   git merge dev    在当前的分支上合并dev分支

   git branch –d dev 删除dev分支

   git branch name  创建分支

   git stash 把当前的工作隐藏起来 等以后恢复现场后继续工作

   git stash list 查看所有被隐藏的文件列表

   git stash apply 恢复被隐藏的文件，但是内容不删除

   git stash drop 删除文件

   git stash pop 恢复文件的同时 也删除文件

   git remote 查看远程库的信息

   git remote –v 查看远程库的详细信息

   git push origin master  Git会把master分支推送到远程库对应的远程分支上