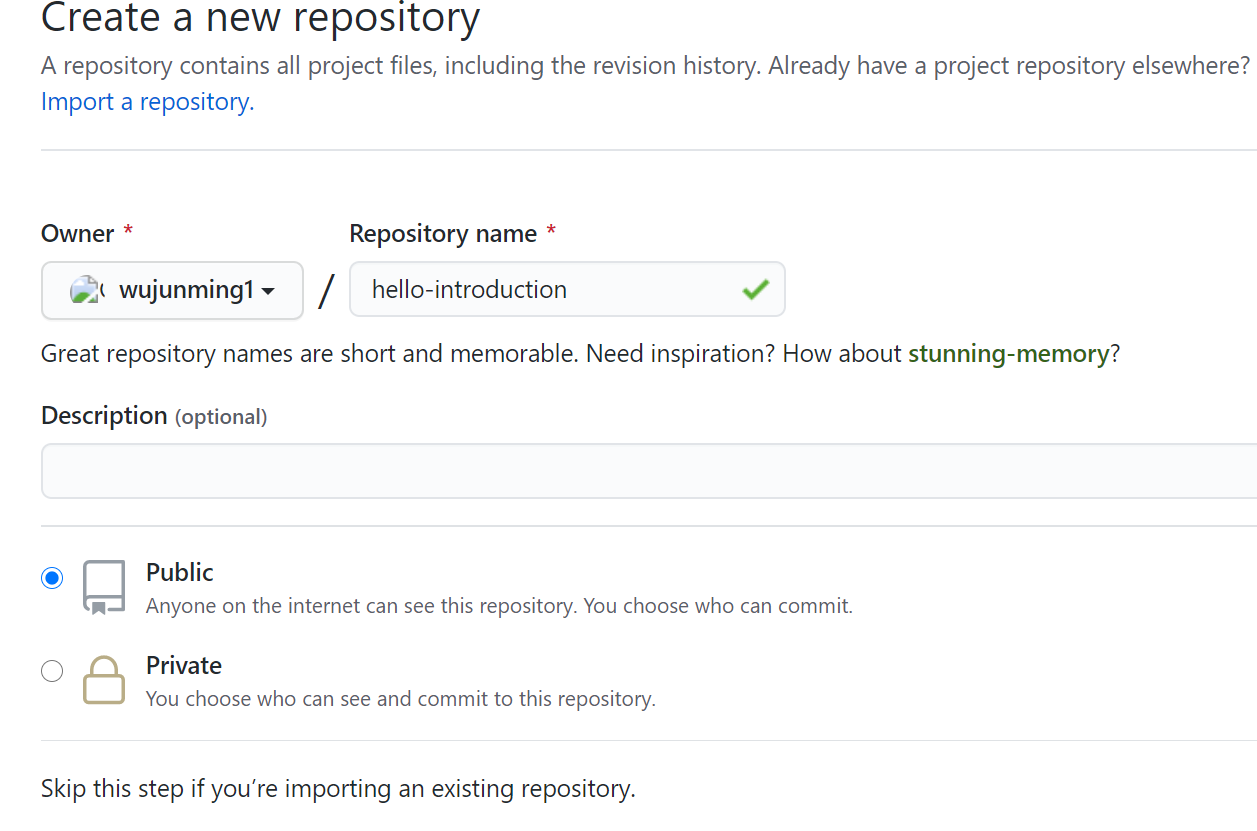
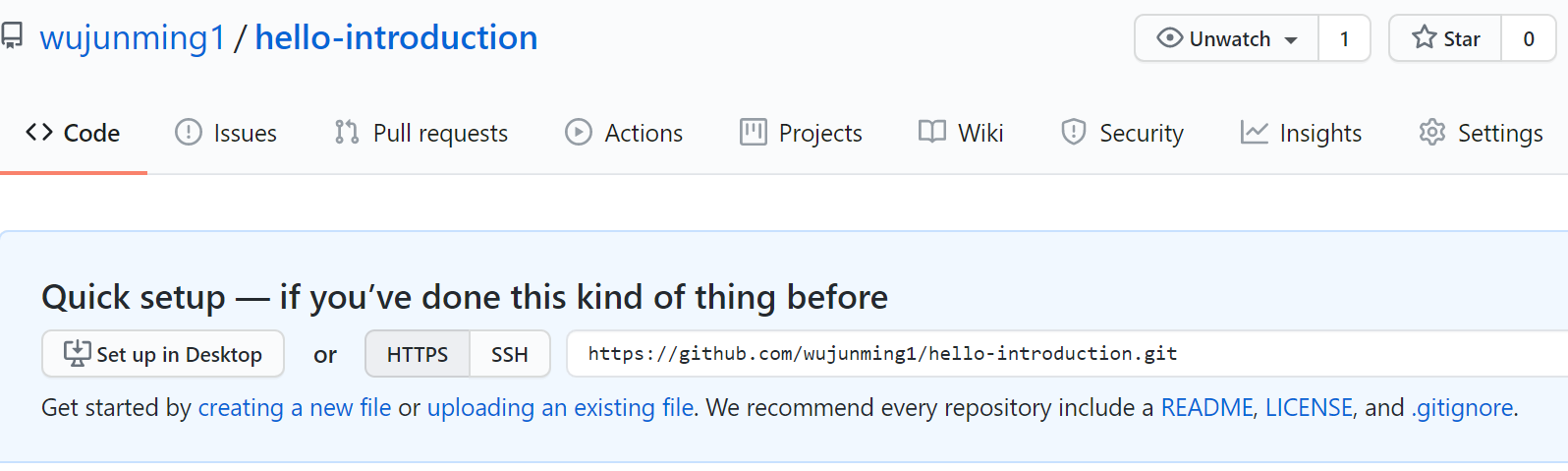
1. **Git下载和安装：**
2. 从git官网上<https://git-scm.com/downloads>下载和安装对于系统（windows、Linux和Mac）的git，默认安装即可，但由于官网上的git的镜像源在国外，下载速度非常慢，甚至可能下载失败，因此推荐第二种方式；
3. <https://npm.taobao.org/mirrors/git-for-windows/>，可通过该镜像源（该镜像源在国内，下载速度非常快）下载安装相应版本的git；

git下载安装完成后，接下来可以使用git指令将本地仓库代码上传到GitHub或GitLab远程仓库

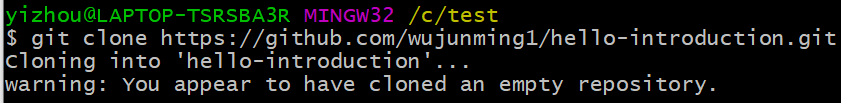
1. **git上传代码或文件至GitHub远程仓库**
2. 在 GitHub 或者 GitLab 上手动创建新的repository，本文基于 GitHub 操作，创建如下图所示：



（2）创建好之后，然后复制一下远程仓库的https地址，如下所示：



（3）在你的本地创建一个空的目录，用来与GitHub的远程仓库相关联（我这里创建时test文件夹），使用git终端切换到test目录下，通过git clone https://github.com/wujunming1/hello-introduction.git命令将GitHub远程仓库上的代码和文件克隆下来，此时本地test目录下就出现了一个hello-introduction文件夹（其实就是GitHub上创建的仓库名），这个文件夹已经不是简单普通的文件夹，它现在完全是由git管理，包括文件的增加、修改和删除都是由git进行记录和跟踪，这样一个文件夹也称为git本地仓库。而且，通过这一指令之后本地仓库与GitHub远程仓库上相关联了，也就是我们通过在本地使用git指令对远程仓库GitHub上的文件更新和删除了。



我们接着可以尝试使用git branch -vv查看一下本地仓库的master分支与远程仓库GitHub上的master是否已经关联。可以看到左边是本地仓库的master分支，右边的是远程仓库master分支，二者已经进行了关联。换句话说，本地仓库已经和远程仓库已经建立好联系了。注：还可通过以下命令行的方式把本地普通文件夹变成git管理的本地仓库并并与远程仓库相互关联（GitHub远程创建了空的“123”仓库）**。**

**echo "# 123" >> README.md**

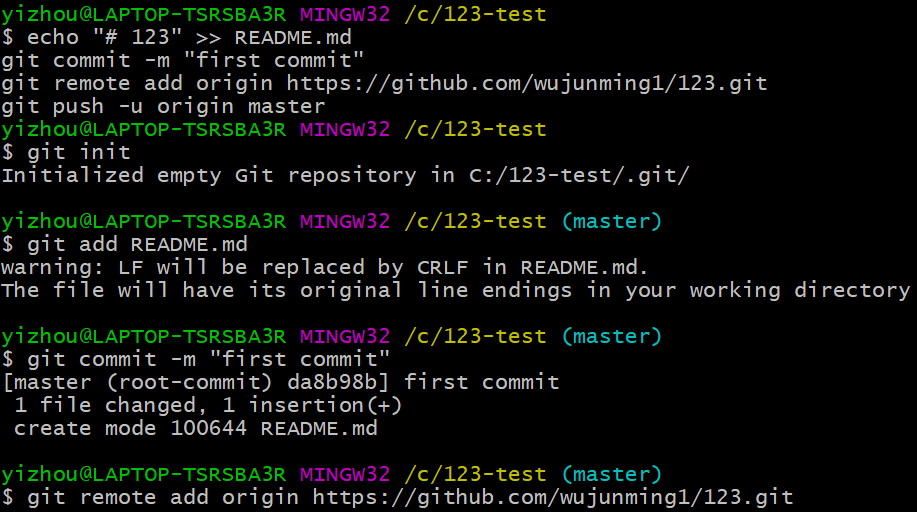
**git init**

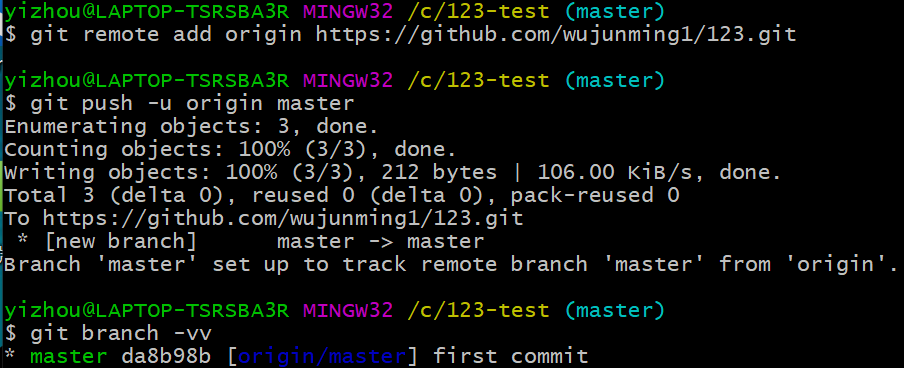
**git add README.md**

**git commit -m "first commit"**

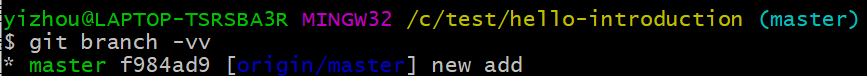
**git remote add origin https://github.com/wujunming1/123.git**

**git push -u origin master**

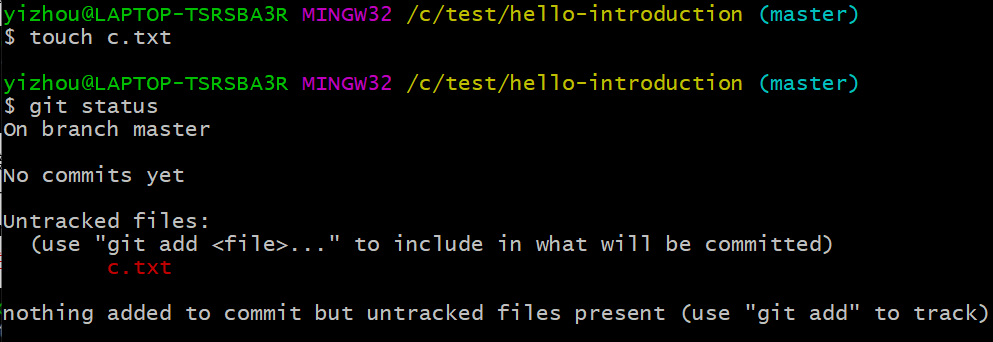


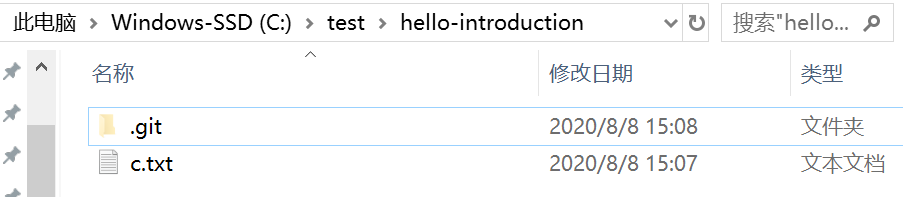


注意：分支的概念，创建、删除、如何切换分支等在之后进行概述。

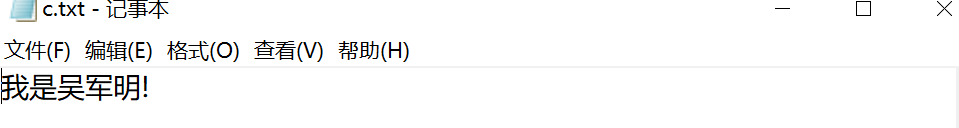


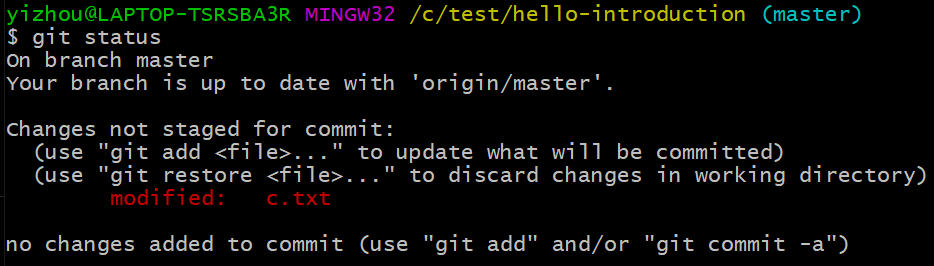
（4）在本地仓库上，我们先假设创建一个c.txt文件，这时我们可以使用git status查看本地仓库的变化。可以看到git提示我们新创建一个文件，是否要使用git add指令提交文件。



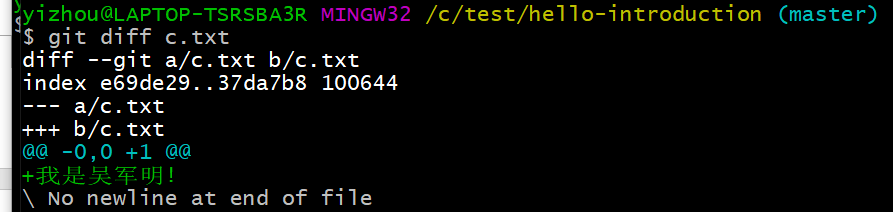


我们接着再对c.txt进行修改操作，添加了“我是吴军明!”这段话，我们再用git status查看一下变化，可以看到git已经提示我们c.txt文件进行了修改，是否提交使用git add 或git commit提交新的修改。





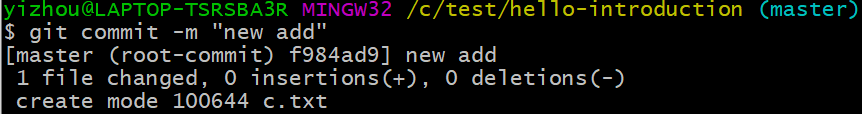
我们还可以使用git diff c.txt指令查看增加和删除了哪些内容。甚至可以使用git checkout – c.txt撤销你所做的修改，回到最初提交之前的版本。这些指令在之后会详细讲解。



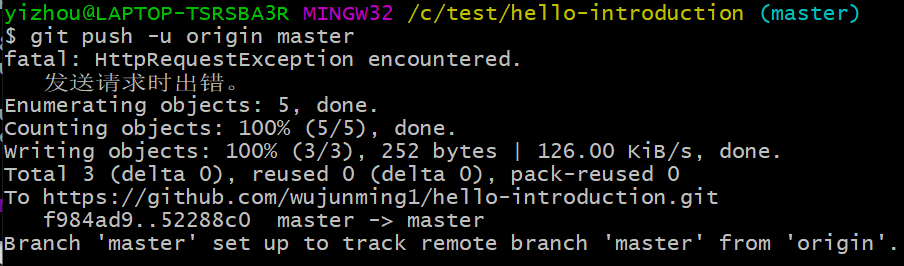
（5）接下来就是提交文件的流程了，我们使用git add c.txt指令（git add . 是提交新创建/修改的所有文件）提交新创建的文件c.txt，add指令是将本地工作区的文件内容提交到暂存区（缓存区）。若没报错，说明提交成功。



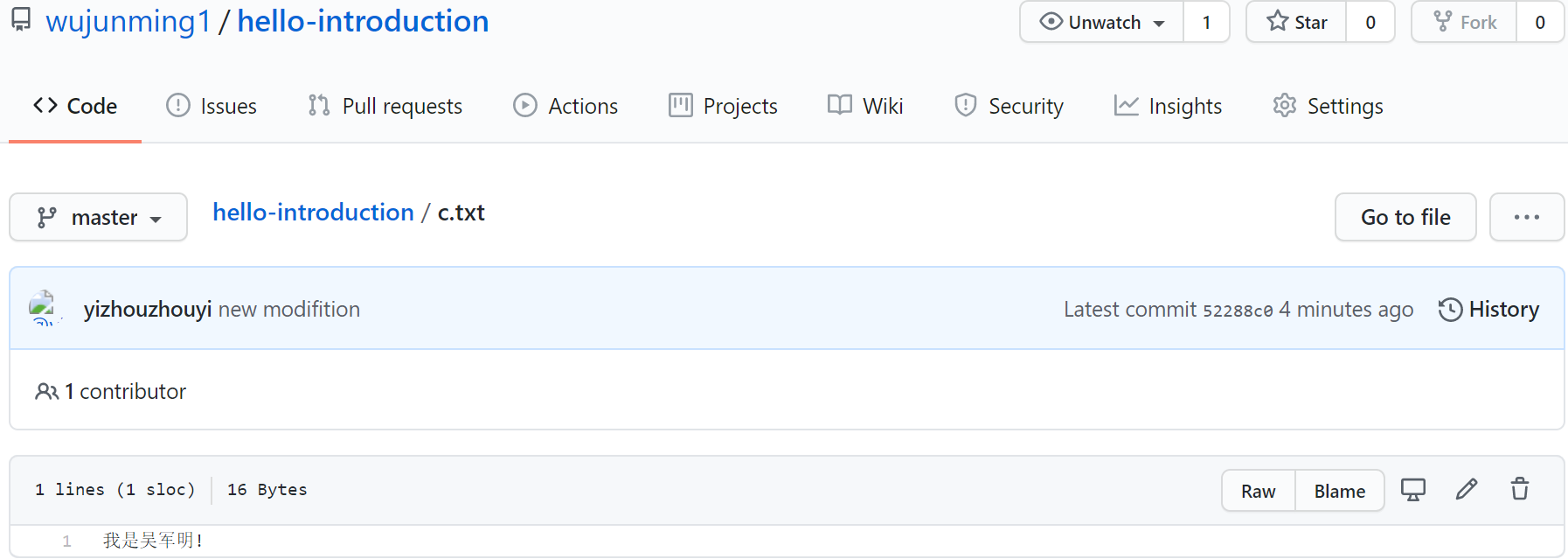
（6）接着再使用git commit -m “文件提交说明”指令提交文件到本地仓库。



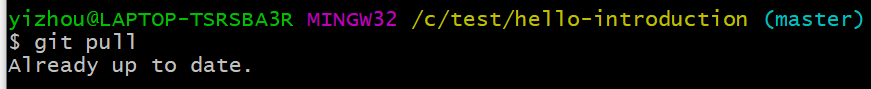
（7）最后关键的一步是，如何将本地仓库的master分支下的代码或文件（这里是c.txt）推送到远程仓库GitHub上。使用git push -u origin master指令即可完成此次操作。很明显git push是推动的意思，-u就是用户的意思，origin有源头的意思，也就是远程仓库的意思，最后master就是分支名。每次使用该指令时需要输入GitHub的用户名和密码，至此我们把修改好的c.txt文件推到了GitHub远程仓库。



&&\*\*&&：可能你会有疑问，git是如何进行版本管理、协同开发的呢，看上面一些指令操作不就是把文件从本地仓库推送到GitHub远程仓库上吗，怎么就能进行协同开发了？最开始的时候我们就说了，使用git clone指令把代码从远端仓库克隆到本地的时候，本地git仓库就已经和GitHub仓库关联了。**现在假设有三个人协同开发一个代码，A负责开发A功能，B负责开发B功能，C负责开发C功能。如果没有git管理的话，三个人修改后的代码只能通过U盘拷贝的方式合并，而有了git这一牛逼的东西之后，合并代码就变得异常方便和简单。**



具体来说就是，A修改完A功能的代码后直接可以把修改后的代码提交到远端仓库，这个时候B、C可能还在用以前的代码开发，这样就可能出现问题。所以B和C在开发之前，一定要先使用**git pull**指令，把远端仓库的代码或文件给拉下来，确保是最新的代码，最后修改各自功能后提交代码即可。可以看出来，有了git指令，省去代码合并时磨嘴皮的功夫，一个指令就能全部搞定！！！



使用git pull指令后，提示已经是最新的代码。如果协同开发时，有人提交和修改了代码和文件，那么它会把修改的代码和文件从远端拉下来。

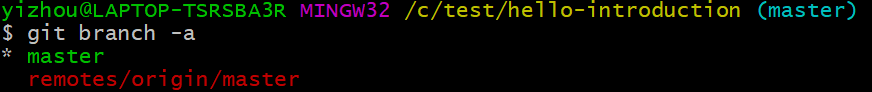
（8）总结来说，git status、git add、git commit、git push和git pull这些基础指令就足够使用了。

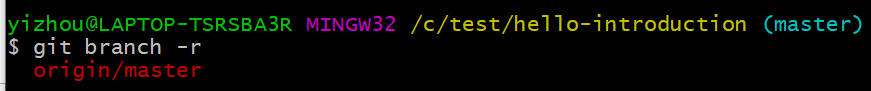
1. **Git分支概念、查看、创建、删除、切换分支**

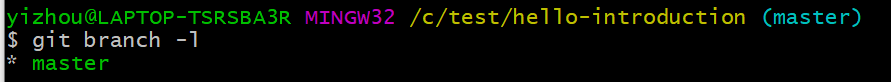
什么是git分支呢？可以理解为树的节点，每个节点有很多子节点。Git仓库有一个最大的master分支，称为主分支。之后创建的分支都是子分支。为啥要有分支这一概念，一个master分支不就够了，为啥还要子分支？。其实你很容易想到，当项目很大时，需要大量的人协同开发，那么只有一个主分支的时候，开发效率很容易受到影响，开发模式还是属于串行的。而且只有一个分支时很容易出现代码合并冲突。有了子分支之后，大家可以在自己创建的子分支上参与开发自己负责的功能，最后每个人在各自分支上编写的代码统一在master主分支合并。这样一种开发模式是不是就变成并行的呢，互不影响，大大提升了开发的效率。其实分支并不是什么高级的东西，当项目很小的时候，一个master主分支就足够了。

接下来介绍一下git分支创建、删除和切换分支的一些指令。

（1）查看分支：git branch -a 查看所有分支，包括本地和远程分支；git branch -r查看远程分支；git branch -r；git branch -l查看本地分支；



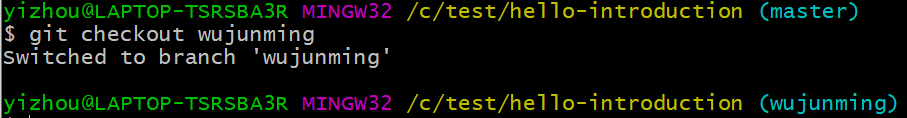




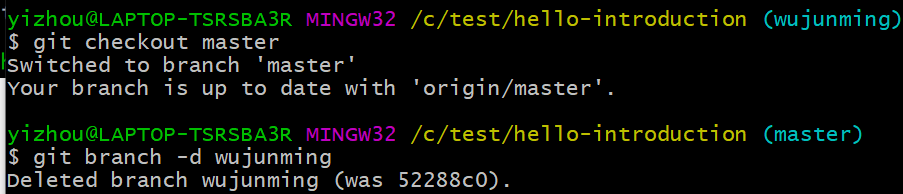
1. 创建分支：git branch “分支名”，没报错，说明创建分支成功了！



（3）切换分支：使用git checkout wujunming指令可以从master分支切换创建好的wujunming分支上；

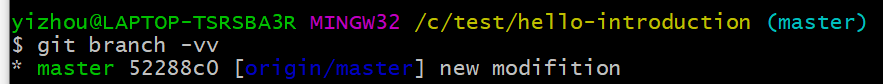


（4）删除分支：git branch -d wujunming指令删除创建的分支；

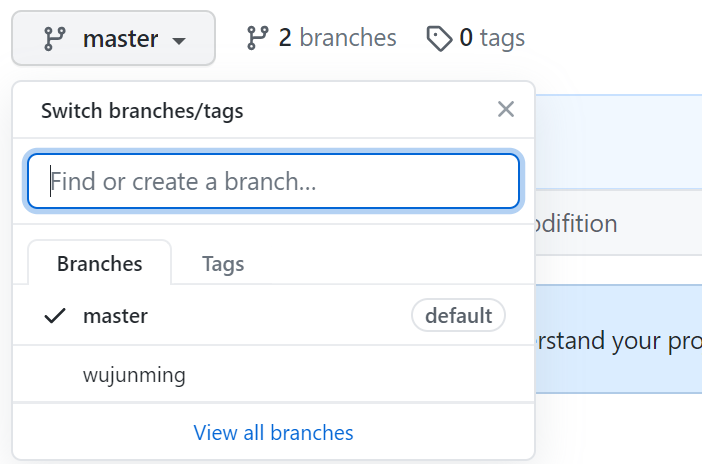


创建的wujunming分支已经被删除了！

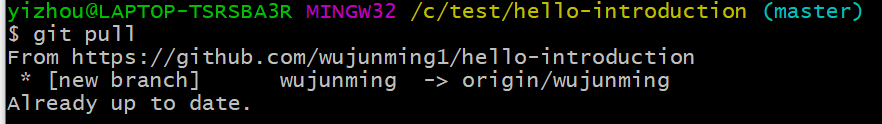
（5）查看本地分支是否与远程分支相关联。git branch -vv，可以看到本地master分支已经和远程分支相互联系。



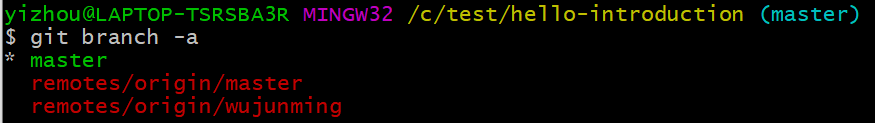
（6）还有一条非常很重要的指令就是如何创建本地分支并且与远程分支关联。假设远程仓库创建好了一个wujunming分支，那么我们如何在本地创建一个wujunming分支并且与远程wujunming关联起来呢？



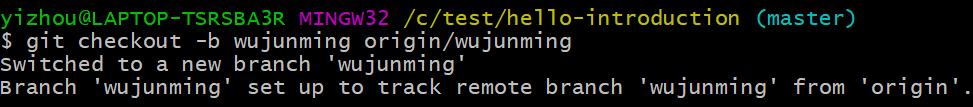
远程仓库创建了新的分支，也就是远端代码或文件发生了变化，此时该怎么做？是不是应该先用git pull指令把最新的代码拉下来。

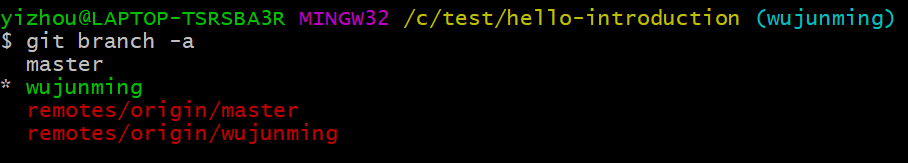


提示我们在远程仓库创建了一个新的分支wujunming。我们再用git branch -a查看一下现在的分支情况。可以看到，远端仓库多了一个wujunming分支。

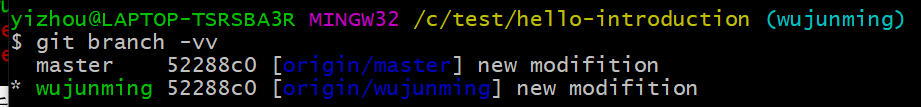


那么，最后我们如何在本地创建一个wujunming分支并且与远程wujunming分支关联呢？使用git checkout -b wujunming origin/wujunming指令，出现以下提示时说明关联成功了！我们再使用git branch -a查看所有分支情况。两个本地分支master、wujunming，两个远程分支origin/master，origin/wujunming。



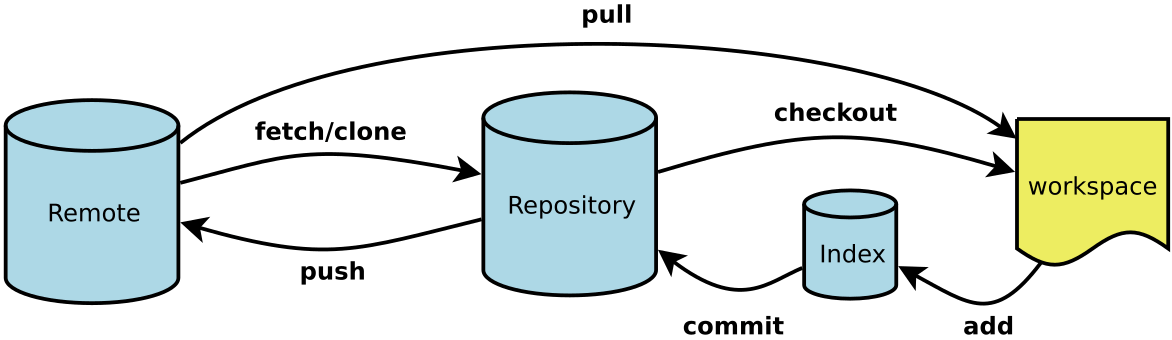


最后再使用git branch -vv查看分支关联情况。本地wujunming分支已经和远程wujunming分支关联起来了！这样本地wujunming分支所做的修改可以同步到wujunming上了！至此分支相关的一些操作都介绍完毕了！



1. **Git其他一些指令（撤销修改和查看提交（git commit）日志）**

**基础知识：**Git本地有四个工作区域：工作目录（Working Directory）、暂存区(Stage/Index)、资源库(Repository或Git Directory)、git仓库(Remote Directory)。文件在这四个区域之间的转换关系如下：



**Workspace**： 工作区，就是你平时存放项目代码的地方；

**Index / Stage**： 暂存区，用于临时存放你的改动，事实上它只是一个文件，保存即将提交到文件列表信息

**Repository**： 仓库区（或版本库），就是安全存放数据的位置，这里面有你提交到所有版本的数据。其中HEAD指向最新放入仓库的版本

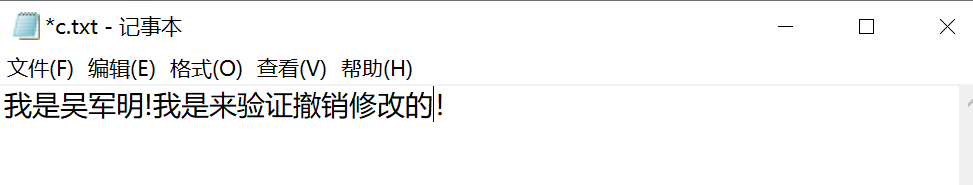
**Remote**： 远程仓库，托管代码的服务器，可以简单的认为是你项目组中的一台电脑用于远程数据交换。

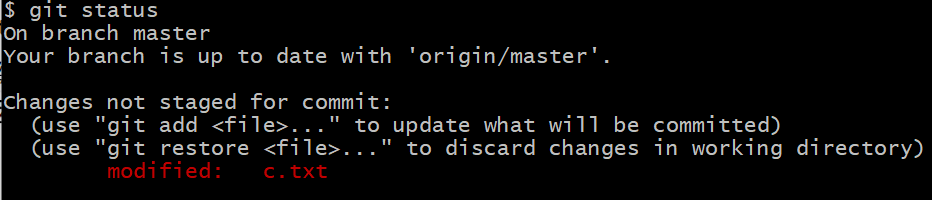
如果在使用“git commit”提交到本地仓库区后，再对文件进行修改时发现修改错误时，要使用git放弃本地的修改时，分为以下三种情况。

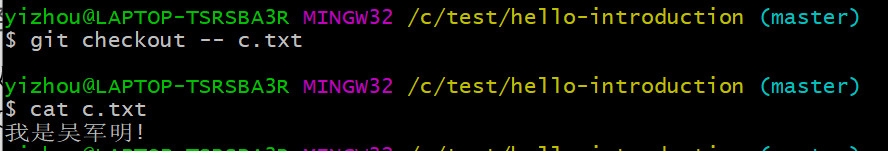
1. **未使用 git add 缓存修改时**

可以使用 git checkout -- filepathname (比如：git checkout -- c.txt ，不要忘记中间的 “--” ，不写就成了切换分支了！！)。放弃所有的文件修改可以使用 git checkout . 命令。此命令用来放弃掉所有还没有加入到缓存区（就是 git add 命令）的修改：内容修改与整个文件删除。但是此命令不会删掉新建的文件。

我们之前在本地仓库创建的c.txt文件，使用“git commit”提交代码到了本地仓库区，之后我们发现还需要对该文件做一些修改，并且修改了此文件，见下图。这时我们还没使用git add缓存代码，这时可以使用git checkout -- c.txt撤销本地刚刚所做的修改。







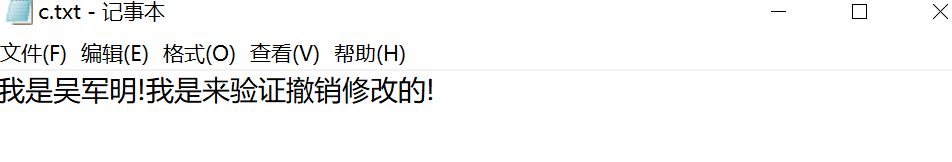
可以看到，我们使用git checkout – c.txt后，c.txt所做的修改已经被撤销了，由“我是吴军明！我是来验证撤销的！”变成了“我是吴军明！”。

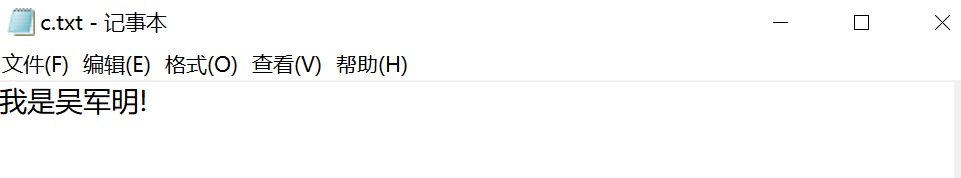
1. **已经使用了  git add 缓存了代码**

可以使用  git reset HEAD filepathname （比如： git reset HEAD readme.md）来放弃指定文件的缓存，放弃所有的缓存可以使用 git reset HEAD . 命令。

此命令用来清除 git  对于文件修改的缓存。相当于撤销 git add 命令所在的工作，即清空缓存区/暂存区。在使用本命令后，本地的修改并不会消失，而是回到了a）的状态。继续用a）中的操作，就可以放弃本地的修改。







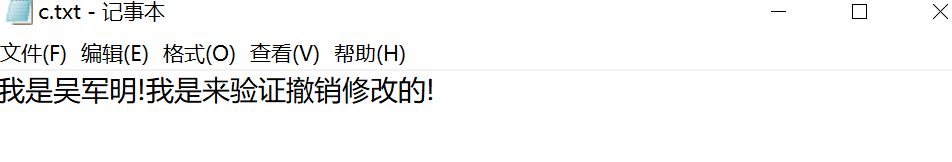
**注意：git commit提交到代码到仓库区时，使用git checkout – “文件名”撤销本地修改都是指的是对工作区的文件进行了回退操作。而仓库区该文件的内容未发生变化，当使用git push指令把代码推送到远端时，推的还是修改之后的文件内容。要推送回退后的文件时，还得使用再次使用git add和git commit指令再次提交。**

1. **已经用 git commit  提交了代码**

可以使用 git reset --hard HEAD^ 来回退到上一次commit的状态。此命令可以用来回退到任意版本：git reset --hard commitid(提交的ID号)

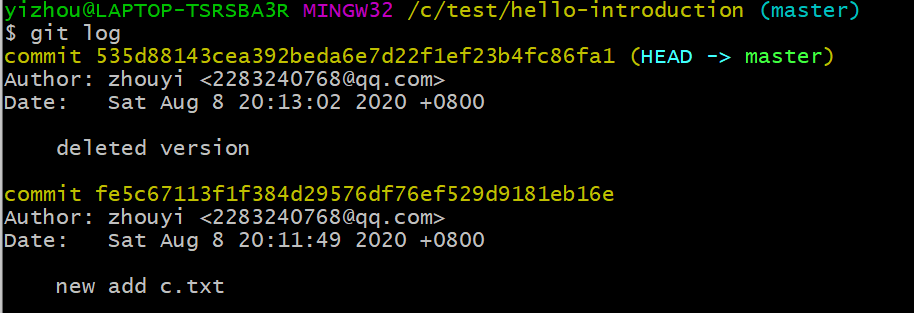
你可以使用 **git log** 命令来查看git的提交历史。git log 的输出如下，这里可以看到的第一行就是最近的一次commit记录。红色圈起来的就是commitid。

**1）第一次提交的文件内容**

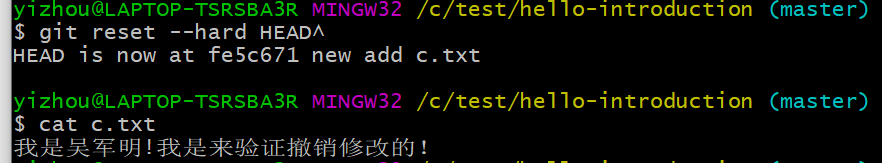


**2）第二次修改并提交的文件内容**

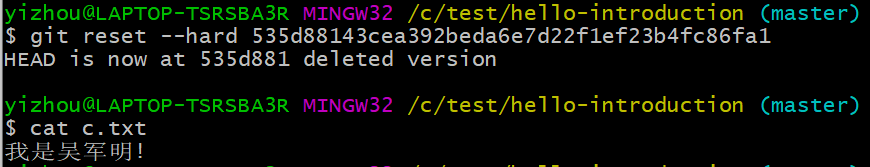




那么我们可以通过git reset –hard HEAD^回退到修改之前的版本。



还可以通过git reset –hard 535d88143cea392beda6e7d22f1ef23b4fc86fa1(提交版本号)回退到你想要的任何一个版本上。



读到这里，你已经掌握了90%的git基础知识，这些经常用到指令得常常练习，熟能生巧。另外还有其他的一些指令可去官网查询。