**西 安 邮 电 大 学**

（计算机学院）

软件项目管理课内实验报告

**实验名称：**  **使用Git进行版本控制**

**专业名称： 软件工程**

**班 级： 软件1903**

**学生姓名：黄宇尘 任基伟 史龙文 李忠帅 马利合 邹夏冰**

**指导教师： 高巍然**

**实验日期：** 2022年5月11日

1. **实验目的**

掌握版本控制工具Git的主要用法，理解版本控制的作用和以项目组形式进行软件开发的方法。

**二、 实验内容**

**（1）安装Git**

下面以Ubuntu20.04.3 系统上的Git为例介绍Git的安装和使用，Windows和macOS上的Git安装、使用方法与Ubuntu上的Git类似，网上参考资料很多。

可以从网址<https://git-scm.com/downloads>下载最新版的Git安装包并解压安装。

这里我们选择直接使用命令行进行一键下载安装：

Terminal中敲出如下命令：

sudo apt-get install git

下载完成后查看git版本：



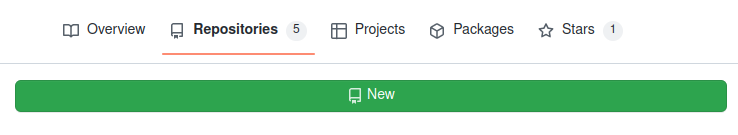
**（2）使用github作为远程仓库**

为了实现多人协同开发，需要有一个Git服务器存放共享的项目代码，本次实验我们选择使用github作为远程仓库进行项目托管。

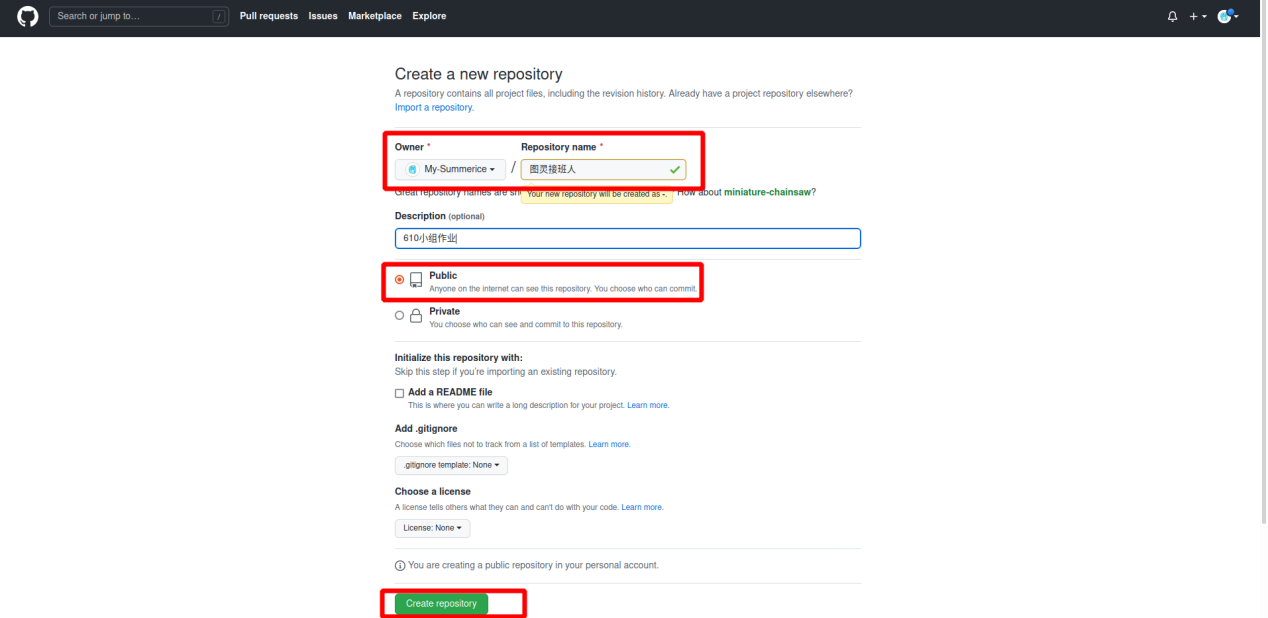
打开Github首页点击Sign up进行账号注册，注册、登录完成后进行下一步。

**（3）创建远程代码仓库**

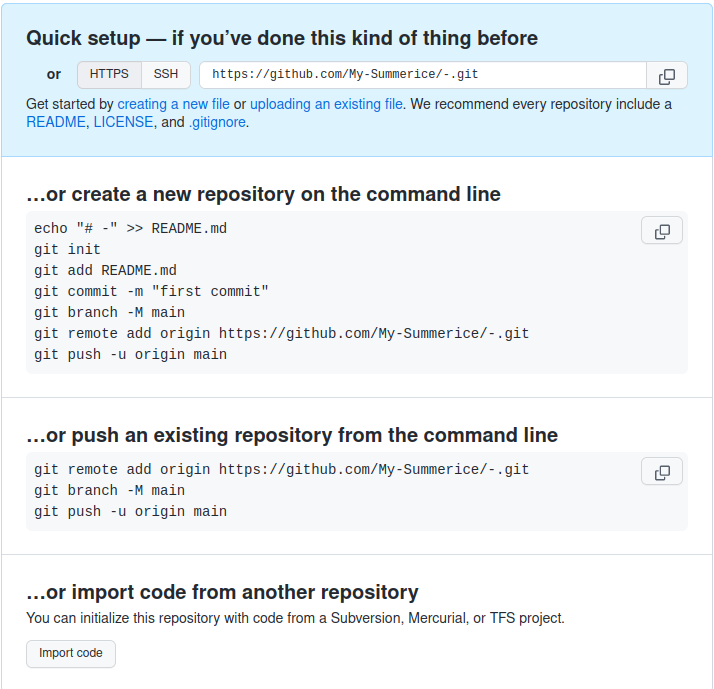
1.新建一个远程仓库，点击Repositories并New一个，如下图所示：



2.设置该仓库的基础信息：



3.创建成功后显示：

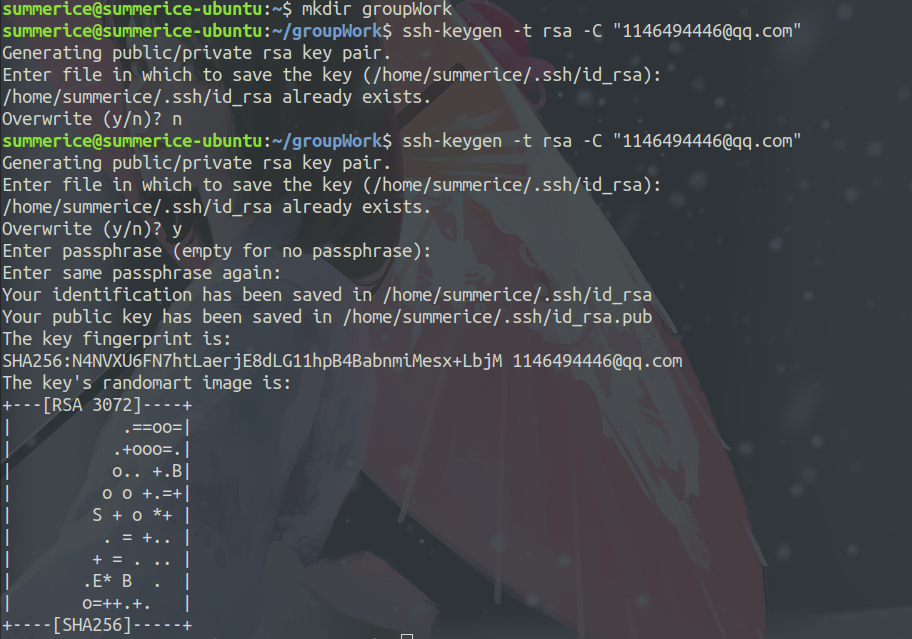


**（4）本机连接到github**

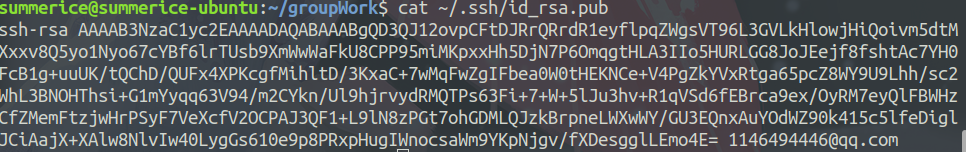
1. 生成ssh key:

输入 $ ssh-keygen -t rsa -C ["youremail@example.com"](mailto:\"youremail@example.com\")

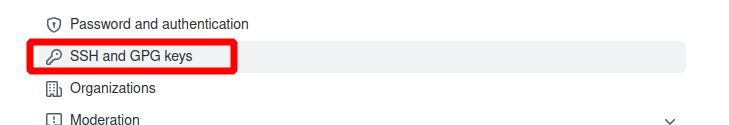
后面的 your\_email@youremail.com 改为你在 Github 上注册的邮箱，之后会要求确认 路径和输入密码，我们这使用默认的，一路回车就行。

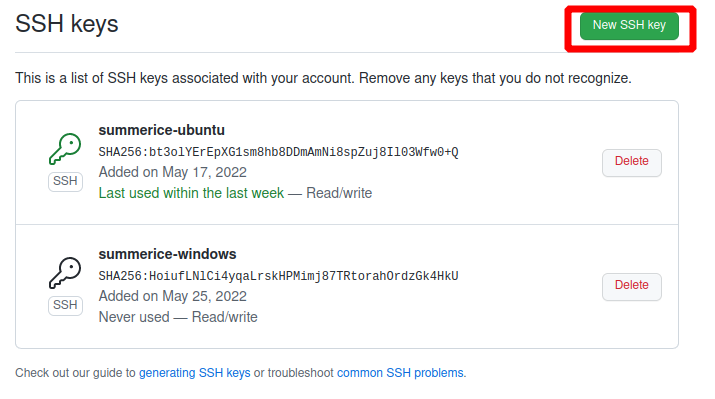


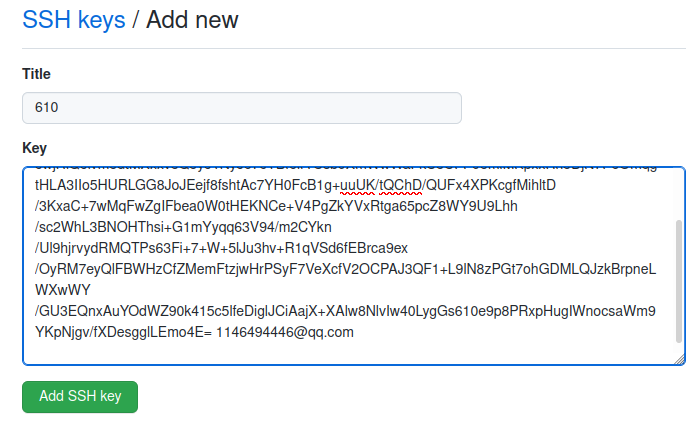
2.成功的话会在 ~/ 下生成 .ssh 文件夹，进去，打开 id\_rsa.pub，复制里面的 key。



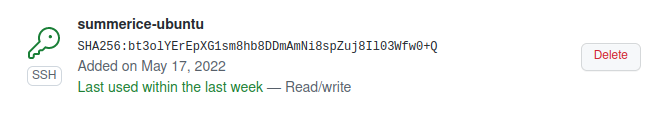
3.回到github ，进入Account => Settings。



 4.将刚才的ssh key粘贴到此处：



成功后显示：

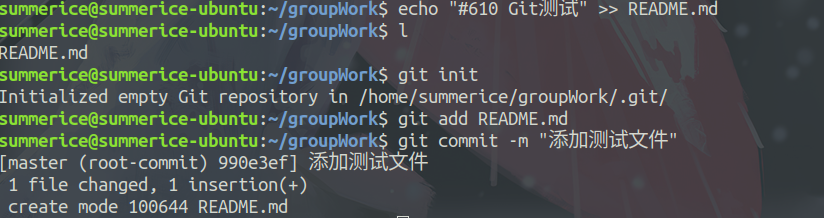


5.命令行再次验证已经连接上了Github：

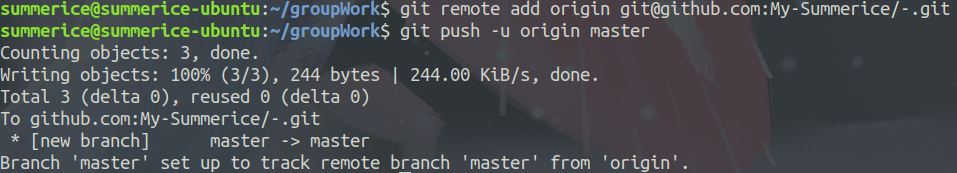


1. **初始化本地Git仓库并创建测试文件提交到远程仓库**

1.根据github给的提示信息进行初始化本地仓库。



1. 添加刚在github创建的远程仓库的SSH链接，并将本地仓库内容推送到远程仓库：



1. 在github中查看提交的内容：



1. **提取、推送、删除远程仓库**

**Git 有两个命令用来提取远程仓库的更新。**

1、从远程仓库下载新分支与数据：

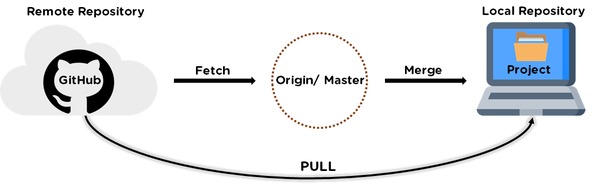
git fetch

该命令执行完后需要执行 git merge 远程分支到你所在的分支。

2、从远端仓库提取数据并尝试合并到当前分支：

git merge

该命令就是在执行 git fetch 之后紧接着执行 git merge 远程分支到你所在的任意分支。



假设你配置好了一个远程仓库，并且你想要提取更新的数据，你可以首先执行 git fetch [alias] 告诉 Git 去获取它有你没有的数据，然后你可以执行 git merge [alias]/[branch] 以将服务器上的任何更新（假设有人这时候推送到服务器了）合并到你的当前分支。

**推送你的新分支与数据到某个远端仓库命令:**

git push [alias] [branch]

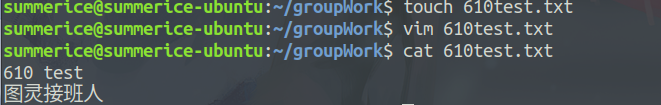
以上命令将你的 [branch] 分支推送成为 [alias] 远程仓库上的 [branch] 分支

**删除远程仓库你可以使用命令：**

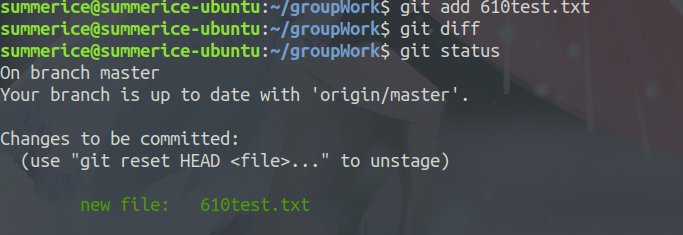
git remote rm [别名]

**（7）把文件提交到本地仓库**

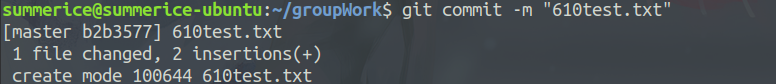
1.小组的各成员在自己本地的工作区文件夹下创建一个文本文件（下面以610test.txt为例），并输入内容：



2.然后将该文件添加到暂存区：



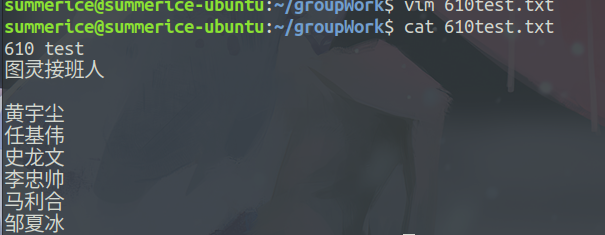
3.再进行commit：



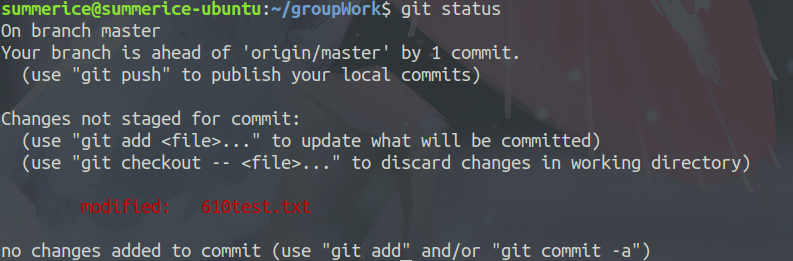
执行完以上两条命令后，610test.txt文件就被提交到了本地仓库中。

**（8）修改仓库中的文件**

1.修改上一步提交的文件610test.txt：

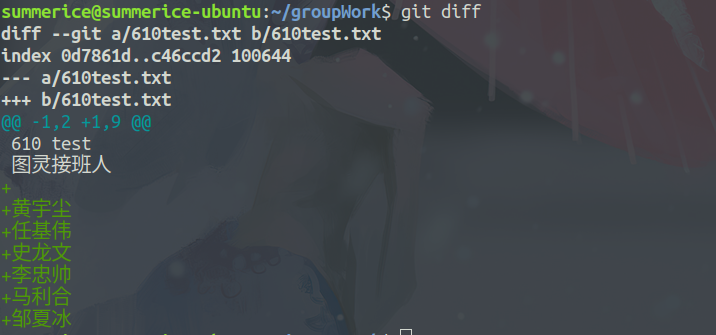


2.使用status参数观察仓库当前状态：



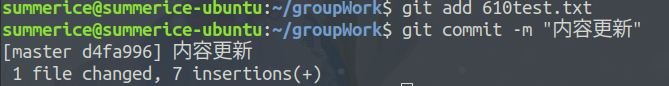
可以观察到工作区与本地仓库的文件不同，Git提示我们工作区的文件发生了变动但未提交，建议我们git commit all。

3.使用diff参数观察工作区与暂存区的差异：



可以看到Git提示a/610test.txt是本地仓库的文件版本，而b/610test.txt是工作区的文件版本，并且底下文件内容的白色部分是共同之处，而绿色的内容则是b版本新加入的内容。

1. 添加并提交改动的文件：



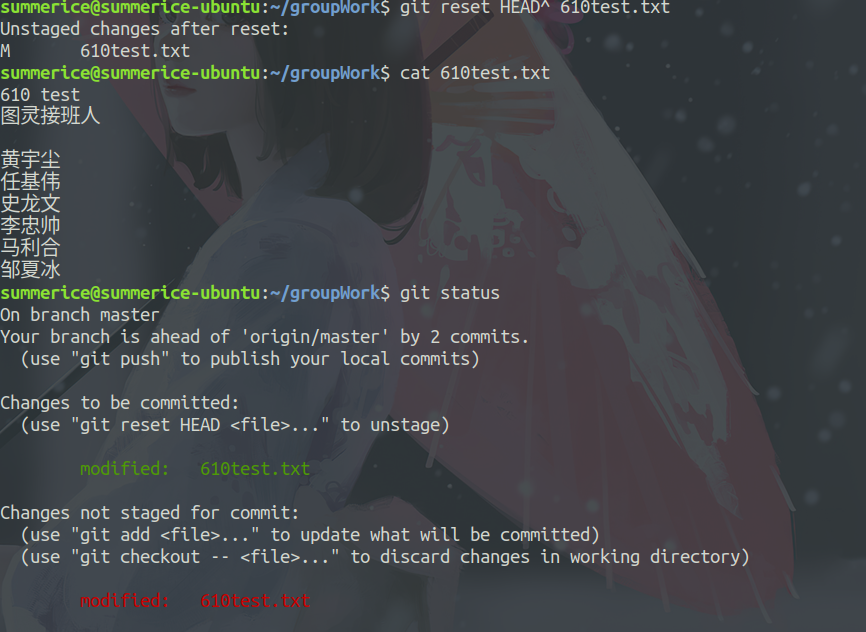
**（9）回退版本**

git reset 命令用于回退版本，可以指定退回某一次提交的版本：

$ git reset HEAD^ # 回退所有内容到上一个版本

$ git reset HEAD^ hello.php # 回退 hello.php 文件的版本到上一个版本

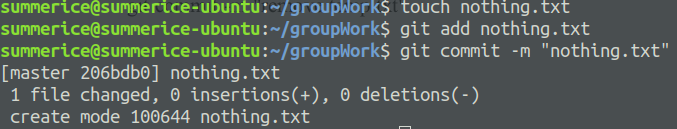
$ git reset 052e # 回退到指定版本

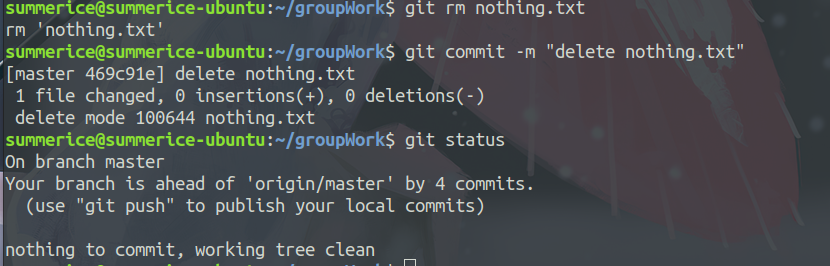


可以看到回退版本回退的是本地仓库的文件而不是工作区的文件。

**（10）删除文件**

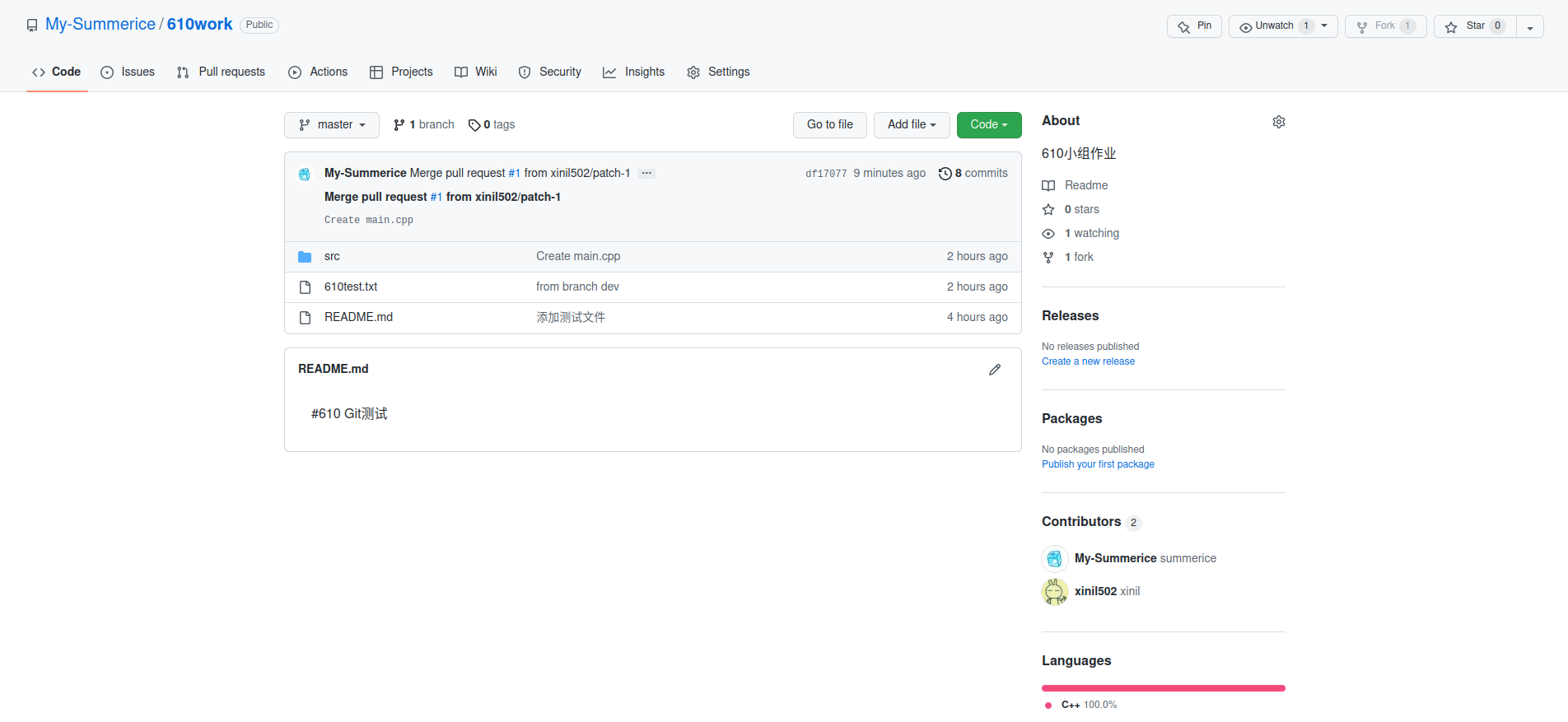
1.在工作区中新建一个文本文件nothing.txt，然后执行以下命令添加和提交该文件：

 2.再执行以下命令从暂存区和工作区中删除文件nothing.txt：



**（11）多人向远程仓库推送不同的文件**

小组成员分别在自己的工作区中创建一个文件名不同的文件，然后各自添加、提交、推送到远程仓库：



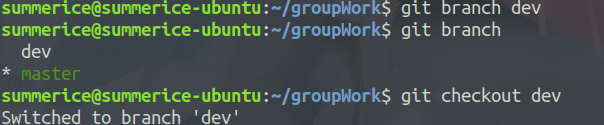
**（12）多人修改和推送同一文件**

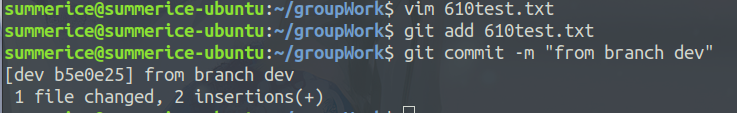
在多人协作开发的过程中，常出现不同成员对同一文件的修改产生冲突的问题。以下操作步骤演示这一问题的出现和解决。

小组中多个成员分别在自己的仓库文件夹中修改同一个文件，然后成员1先执行一次提交和推送命令，把这个文件推送到远程仓库；然后成员2也执行以上这两条命令，此时Git会提示不能推送，因为远程仓库中的这个文件包含另一个成员添加的新内容。所以成员2要先执行push将该文件拉取到本地并与本地的文件合并；之后成员2打开该文件，可以看到Git自动合并后的内容（在符号<<<<<和>>>>>>之间），对文件内容进行修改后保存，然后添加、提交、推送到远程仓库。（以此类推，每个人的操作都相同）

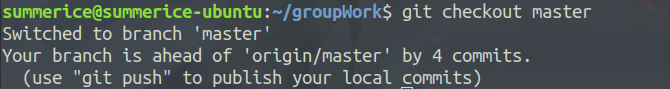
**（13）创建与合并分支**

1.小组各成员在自己的仓库文件夹中执行以下命令创建一个名称为dev的分支：

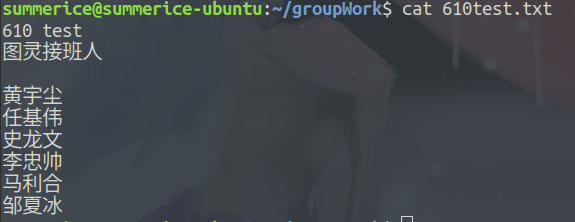
 2.接下来修改610test.txt文件，加上一行文字：from branch dev。然后添加并提交：



3.切换回master分支：

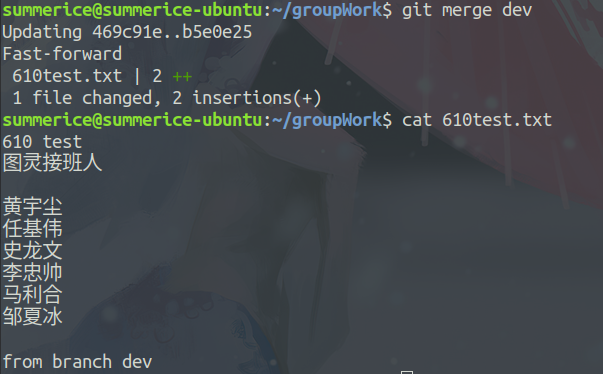


4.查看610test.txt文件，可以看到新加的一行内容不见了：



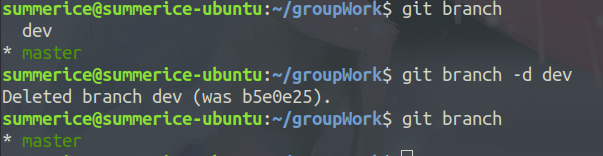
这说明分支之间的内容在合并之前是互不影响的。

5.执行以下命令，把dev分支的工作成果合并到master分支上：



可以看到在dev分支中更新的内容已经被合并到master分支中了。

6.执行以下命令删除dev分支：



可以看到dev分支已经被删除掉了。

1. 注意：

1.以上所有操作都可以在commit之后push到远程仓库。

2.在合并分支时若出现冲突，则需要手动修改文件内容并添加提交。

3.git reset慎用，回退版本之后再强行推送到远程仓库可能会导致回退版本前的所有提交记录都消失。（非常棘手。。。）

**三、实验总结**

通过本次实验我们了解到了Git 是一个免费的、开源的分布式版本控制系统，可以快速高效地处理从小型到大型的各种项目，并且它有着独一无二的分支模型，深刻地体会到了它的强大之处。并且Git 易于学习，指令多而不杂，很容易理解，我们已经使用Git将我们重要的项目代码托管到了github上，多人协作时也十分便捷，再也不用像以前一样每个人写完自己的代码，最终再由一个人进行汇总。给我们带来了极大的便利，由衷地希望每个开发者都能来学习使用Git。