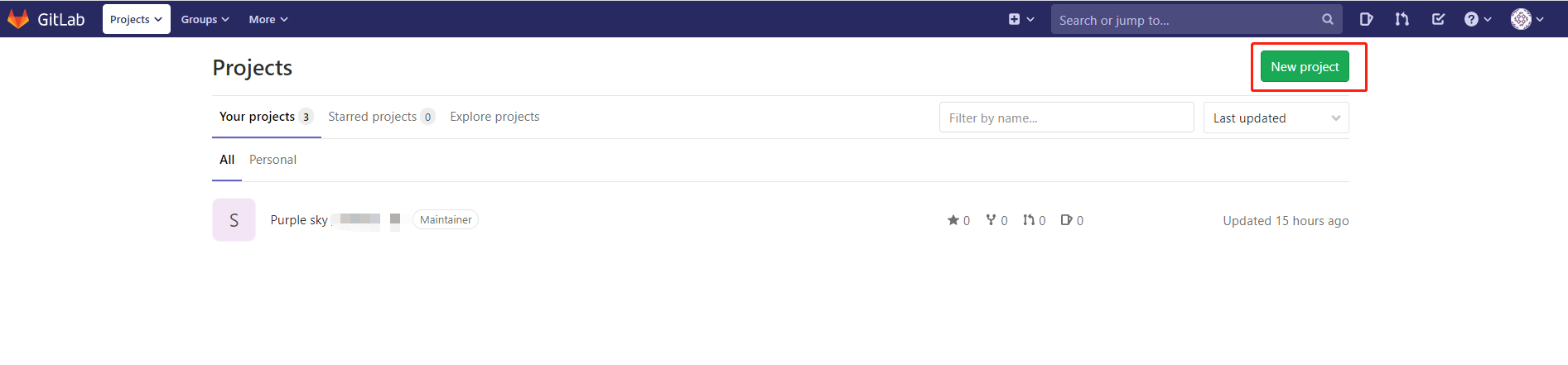
# GitLab使用说明

在本文档中会介绍GitLab的两种使用方式，一种采用git命令行的方式，另一种采用Git的图形化界面软件来进行操作。至于使用那种操作方式用户则可以根据情况而定，各有各的优缺点。至于GitLab账号的注册就不再进行一一介绍了，从仓库的创建开始吧。

### GitLab仓库创建与删除

#### 仓库创建：

进入GitLab的主页面点击NEW Project来创建一个仓库。

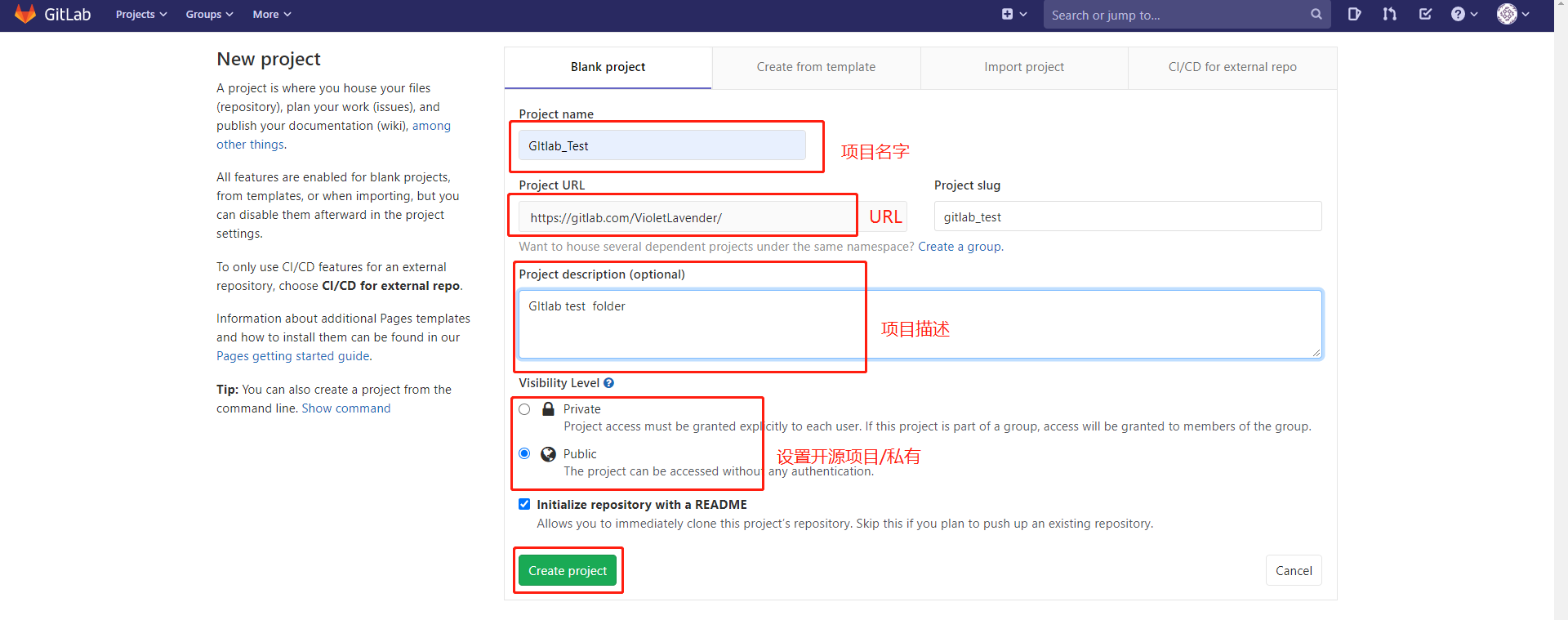


进入以下的界面来配置仓库，配置完成以后点击Creat project，工程创建完成。

URL是根据Project name自动生成的，一般不需要进行操作。

Project decription为项目添加描述。

Visibility Level选项配置仓库为开源或者私有。



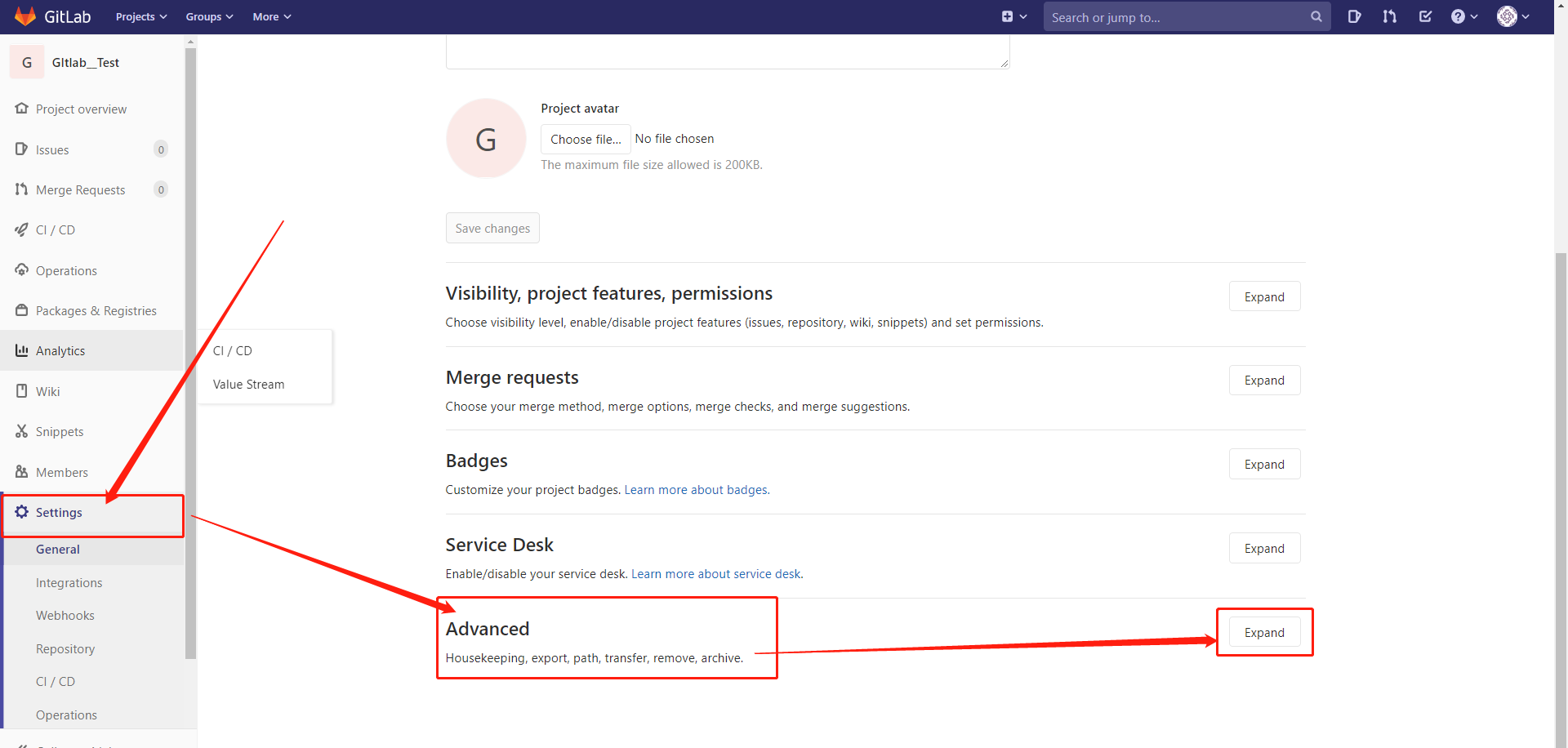
#### 仓库删除

仓库的删除方法如下：

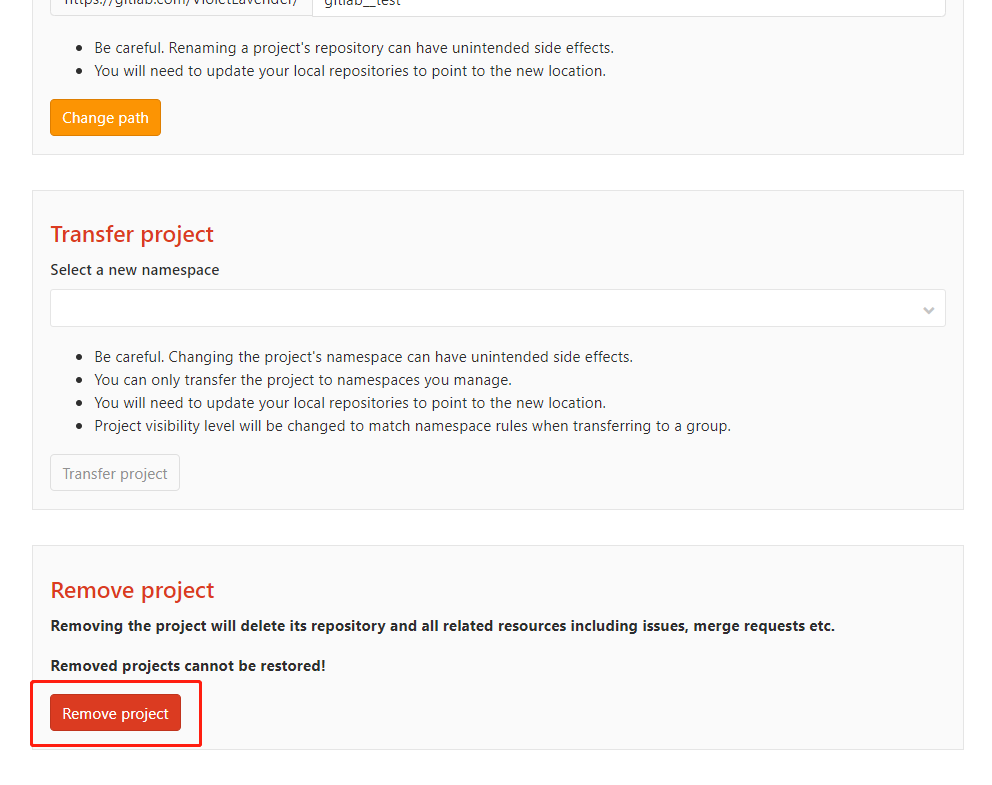
1.进入需要删除的仓库。



2.点击左侧的Setting选项卡，找到Advanced选项，点击Expand来展开相关的操作选项卡。



3.在Expand选项卡的最下面找到并点击Remove Project按钮，工程即可移除。（已经移除的项目与新建的项目的名字不能重复，否则会报错。*删除仓库请慎重！！！*）

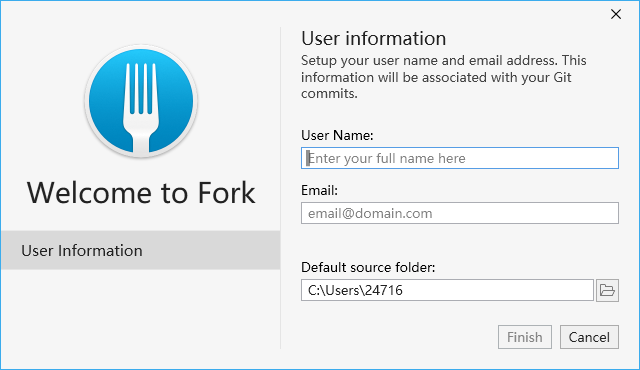


以上就是关于仓库创建与删除的操作。

# Fork使用说明：

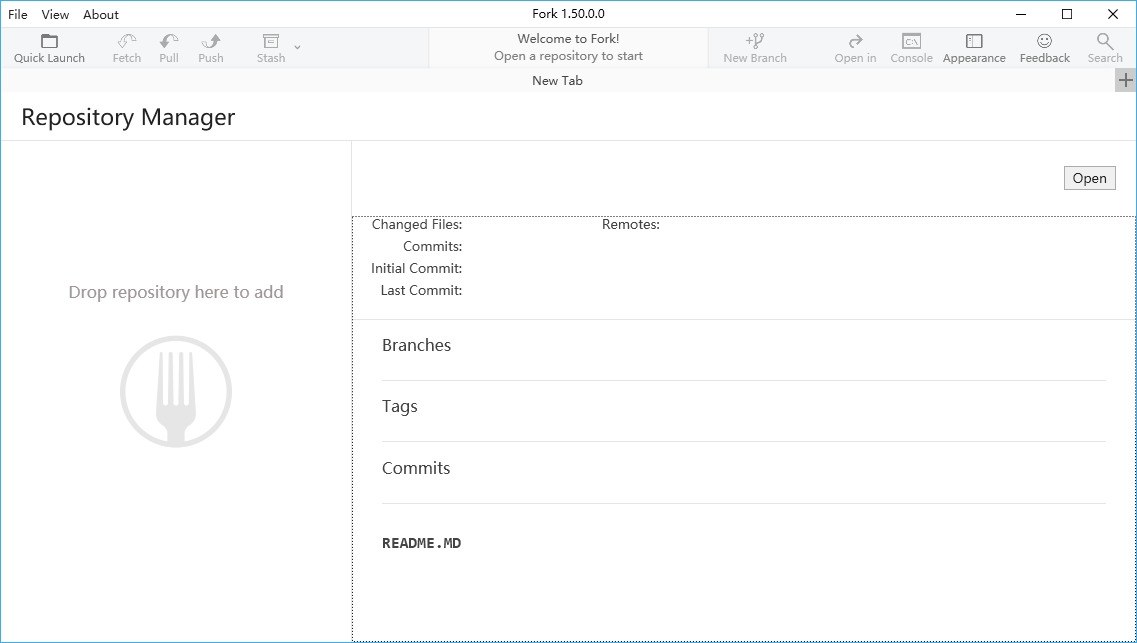
在Fork客户端操作Git是非常方便简单的，不会命令行的用户可以使用fork来进行代码的管理，具体的操作流程就从FORK的安装开始吧。

## Fork安装：

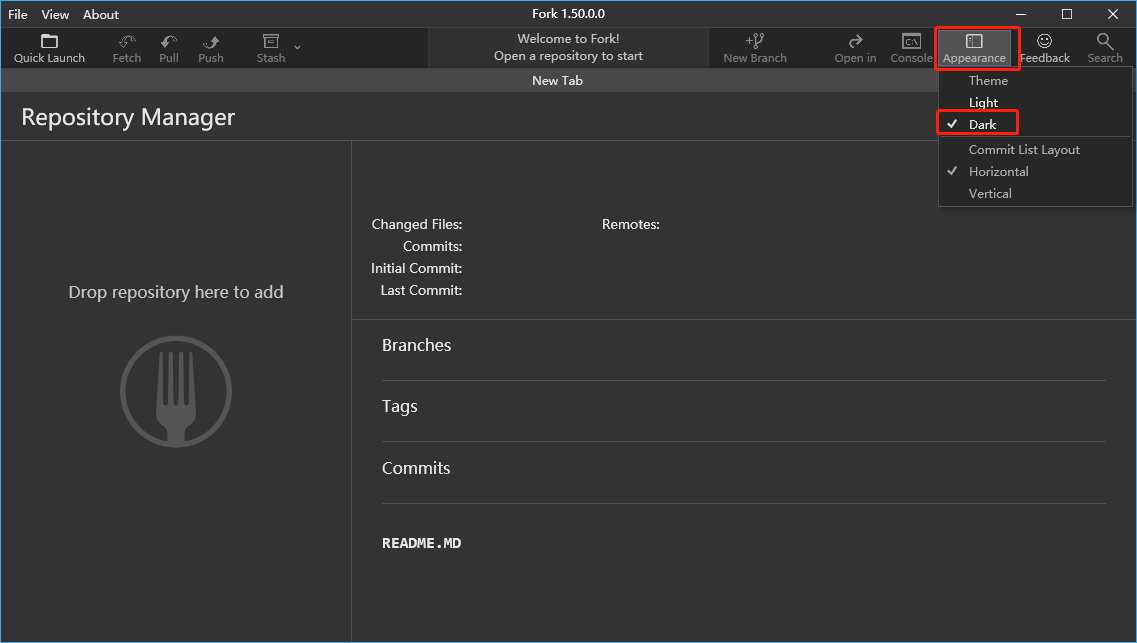


配置Fork的名字，邮箱与文件存放路径。然后点击Finish,等待Fork安装完成。

打开Fork客户端：



Fork 的纯白色的主题感觉不好看，接下来我们来设置背景颜色。

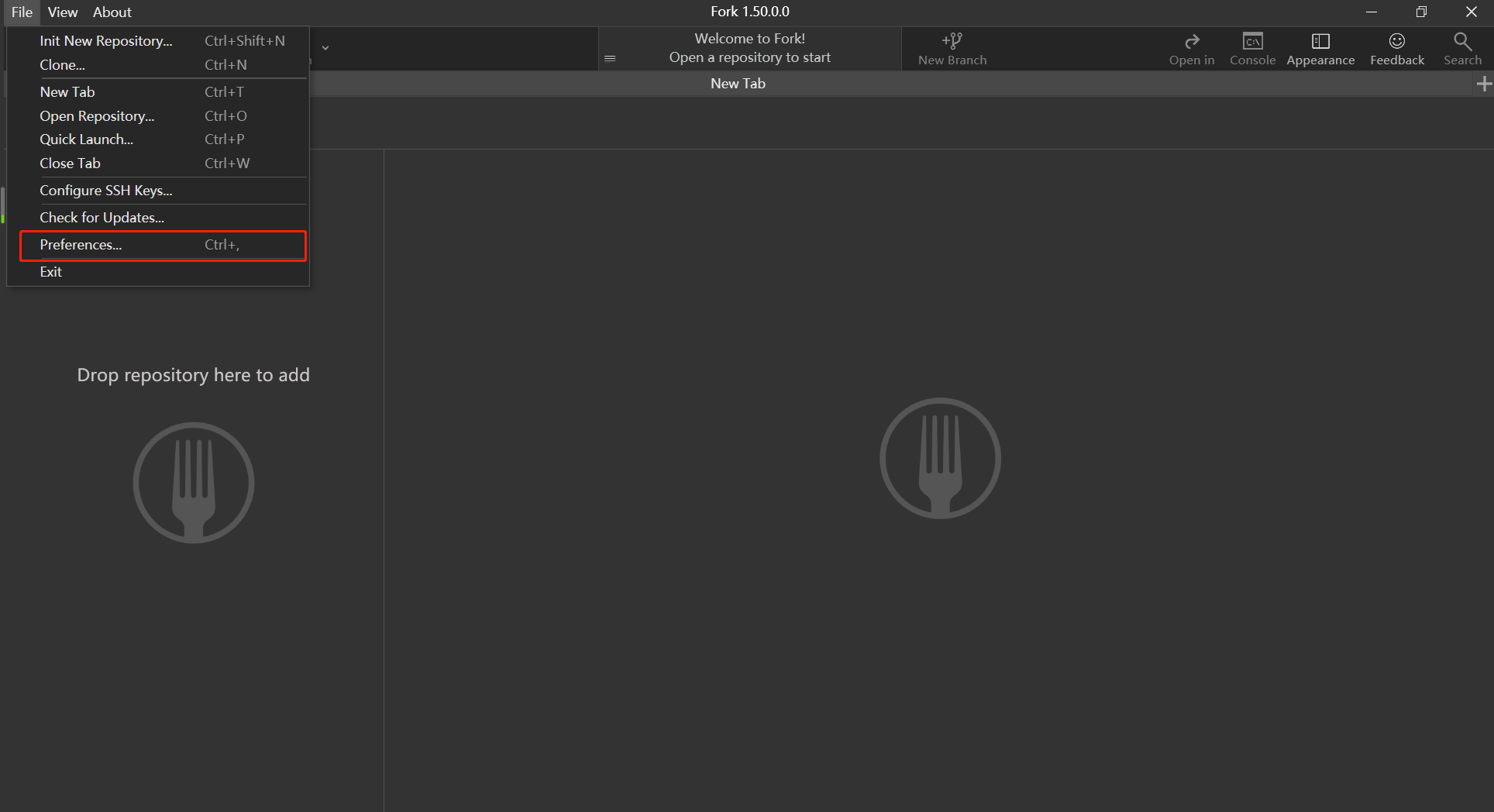


## Fork账户配置：

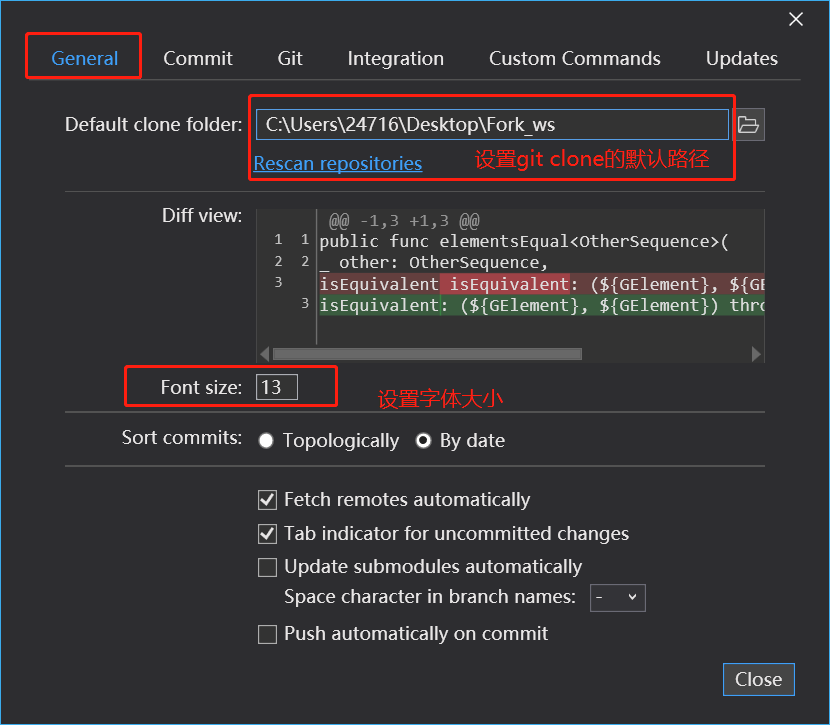
原来配置完Git的用户在这里可以跳过此步骤，如果原来没有配置Git账号的用户可以在fork上对Git进行配置，具体的配置如下：

#### 配置Git的本地账户：

在file选项卡中找到并进入Preferences选项卡。

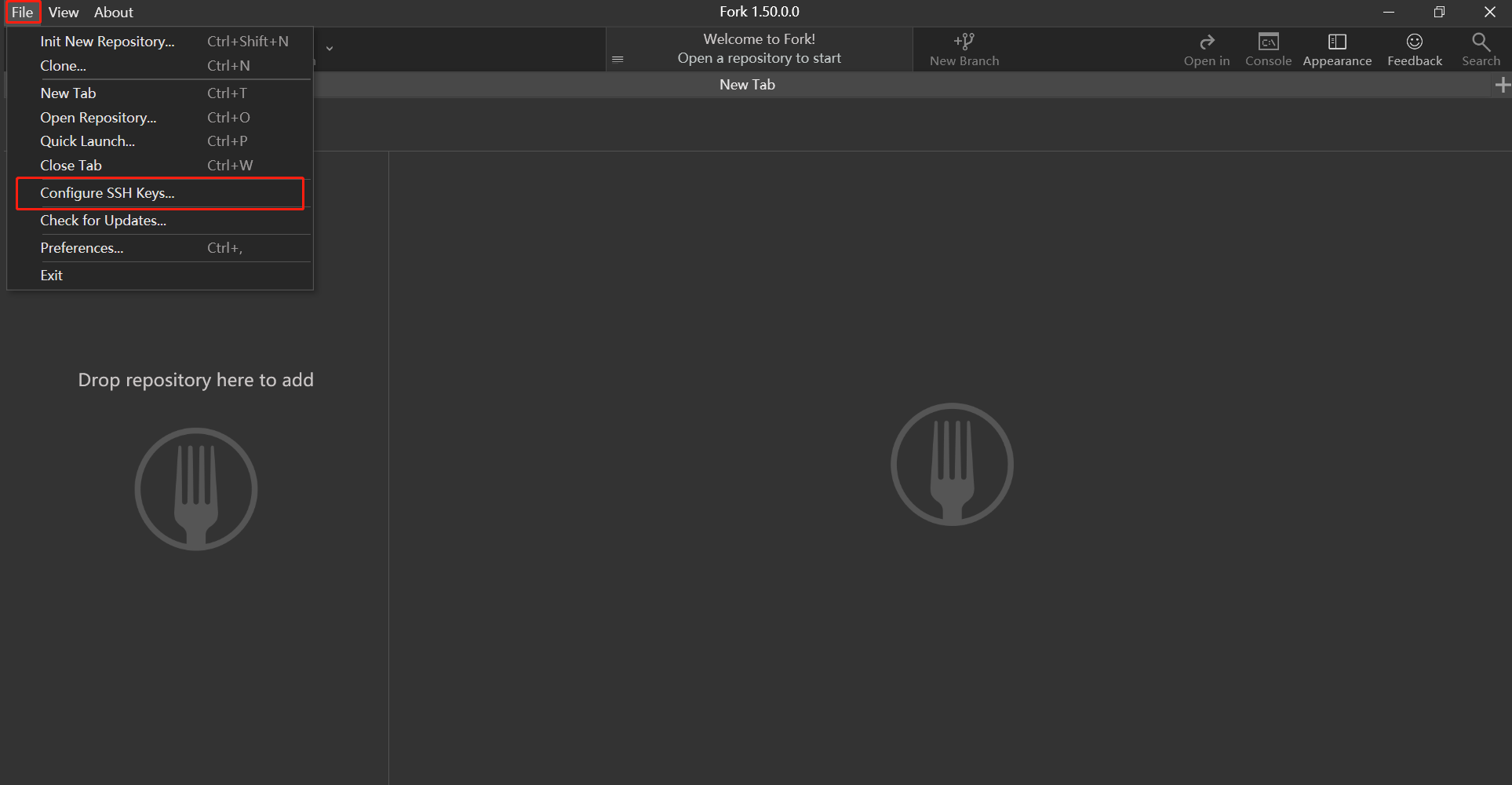


Git clone路径进行设置以及字体大小设置。

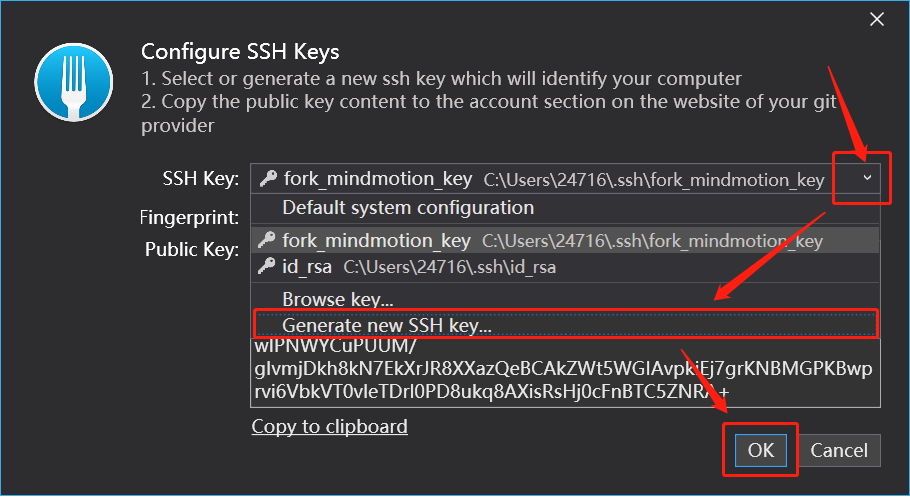


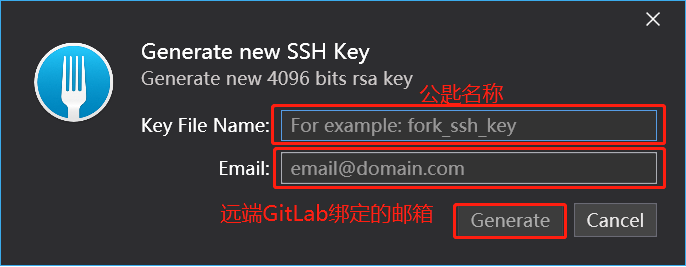
#### 配置SSH

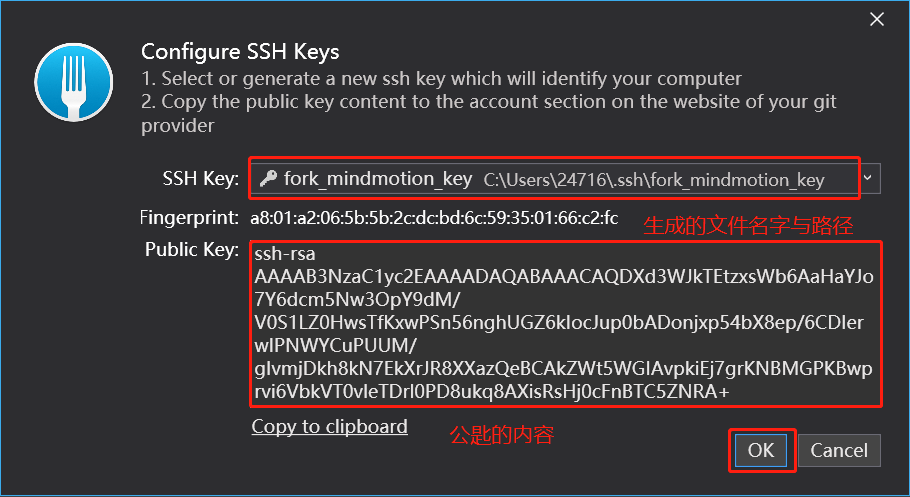
配置SSH 将本地的Git与Gitlab进行关联，进入File->Configure SSH Keys选项卡。



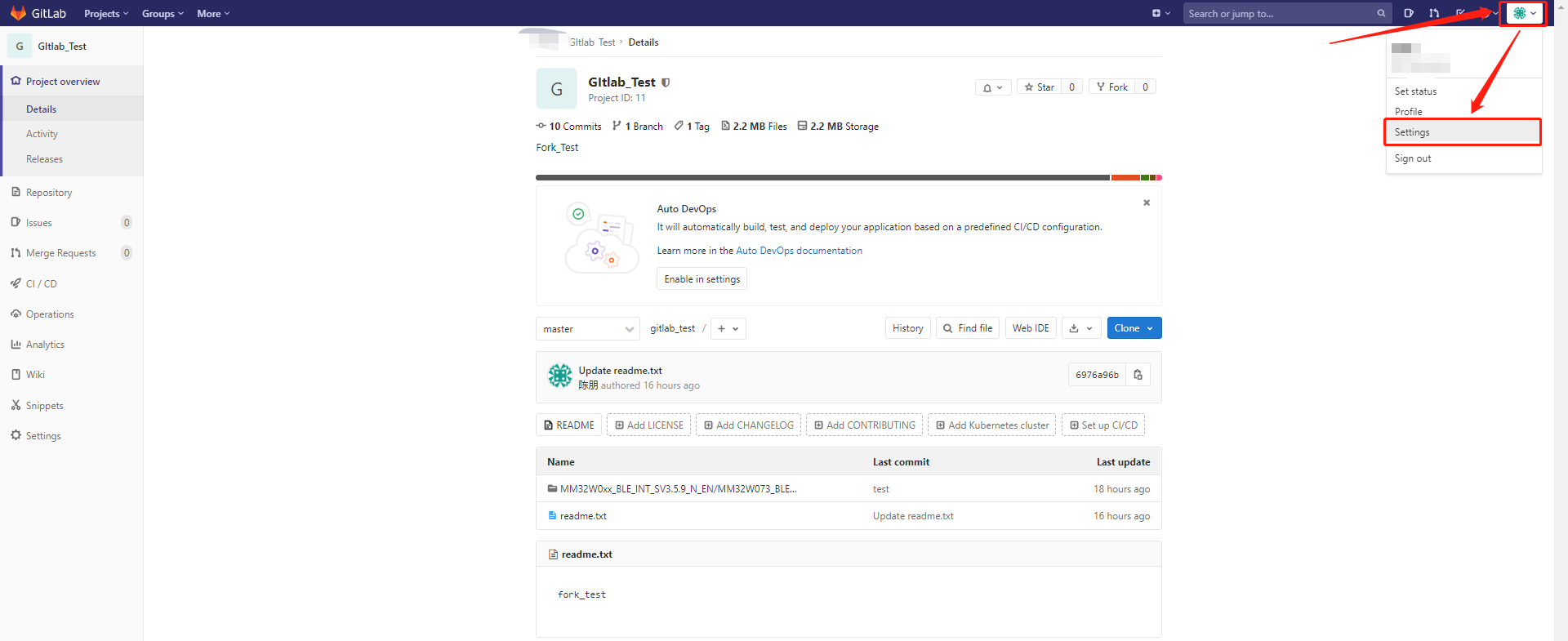
配置并生成新的公匙

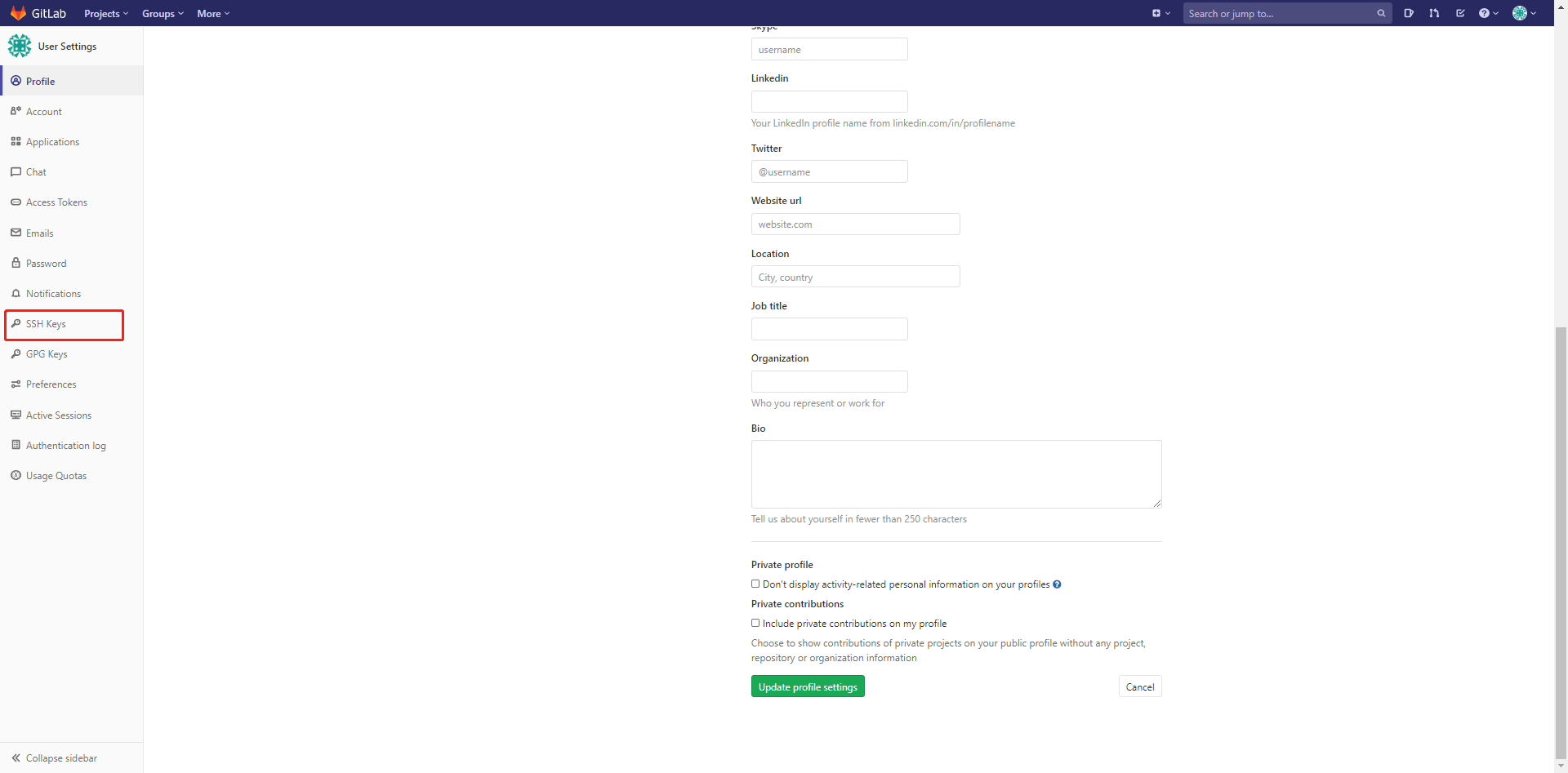


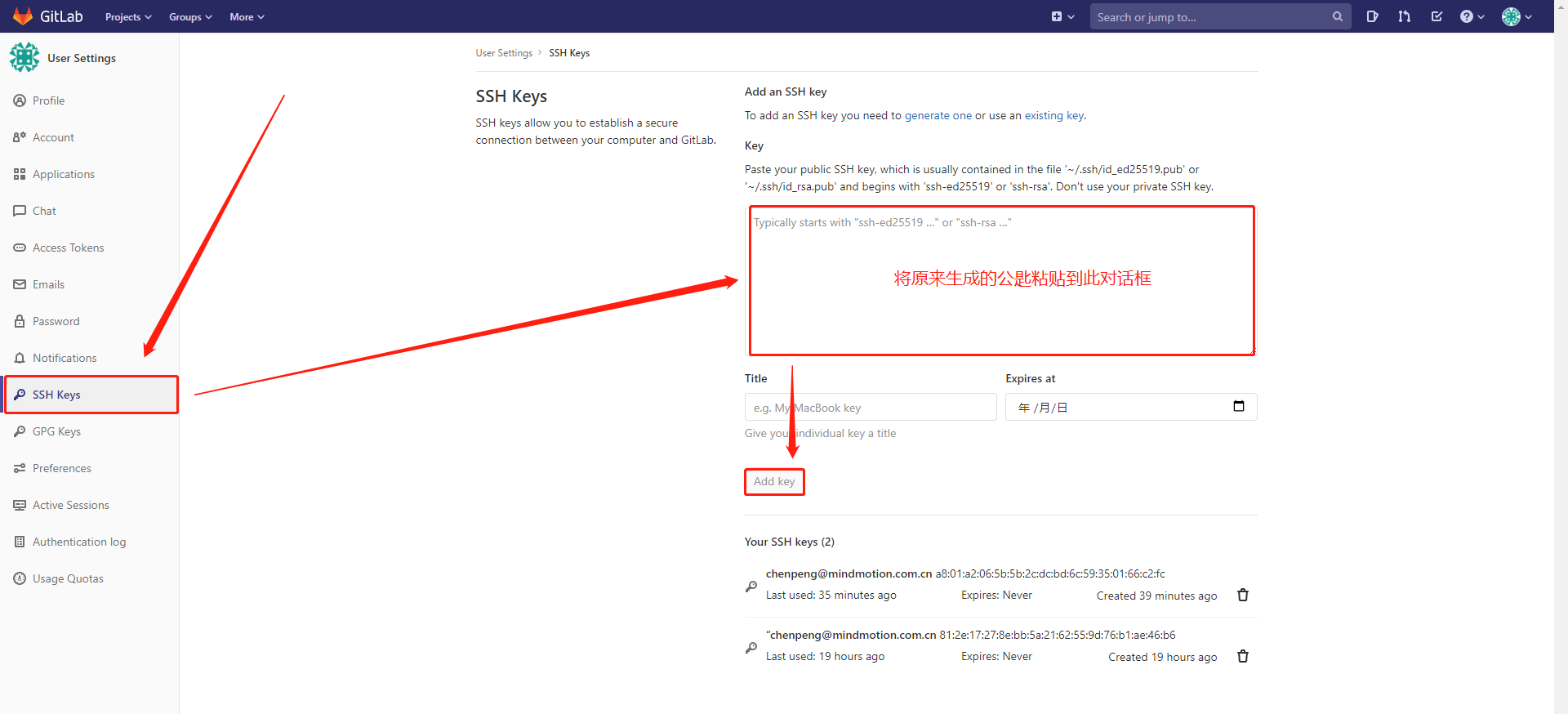




将公匙的内容复制到gitlab网页端的ssh key中具体的操作如下：





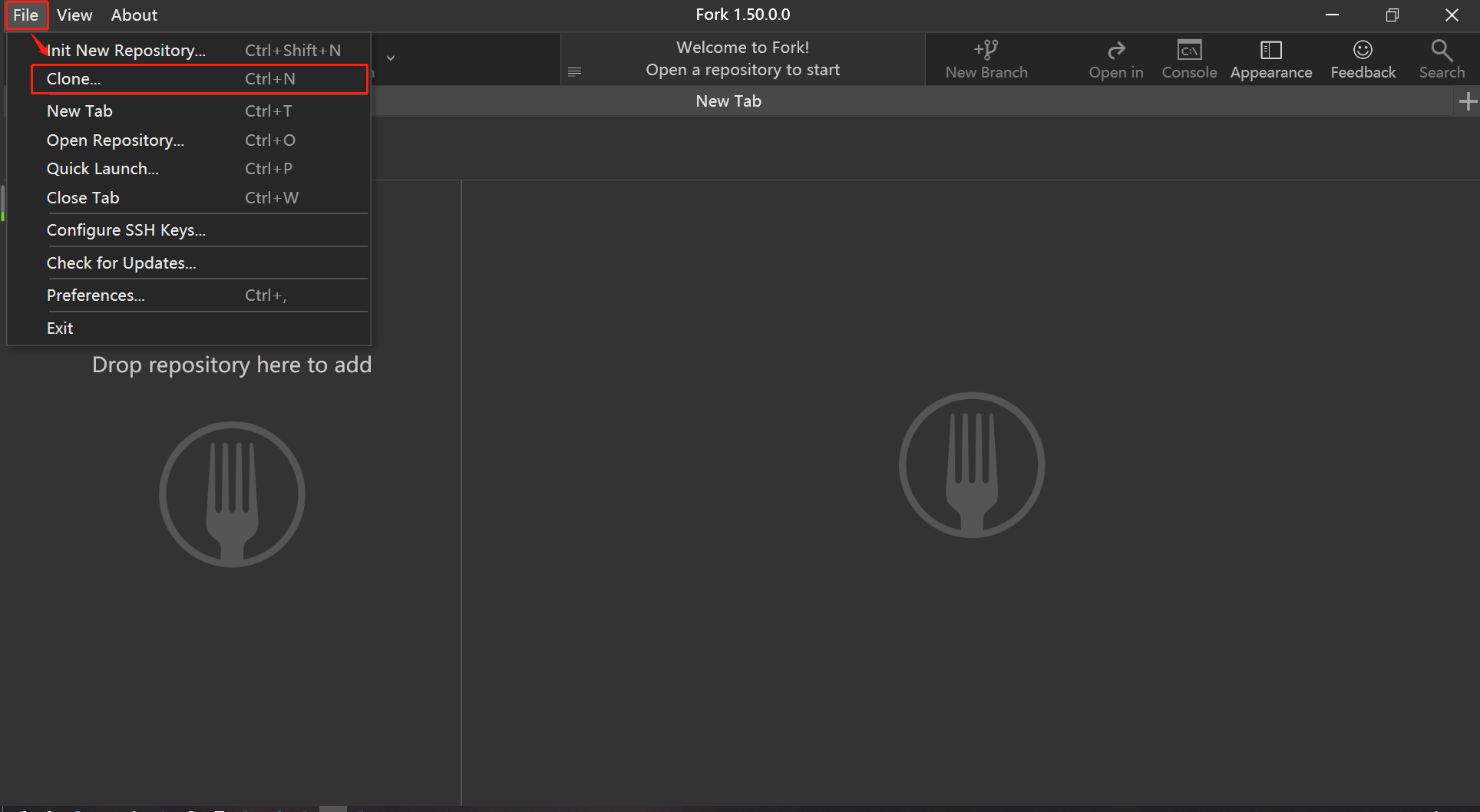


以上我们的fork客户端的GIT与远端的Gitlab就进行了连接，后面我们就可以正常的使用了。

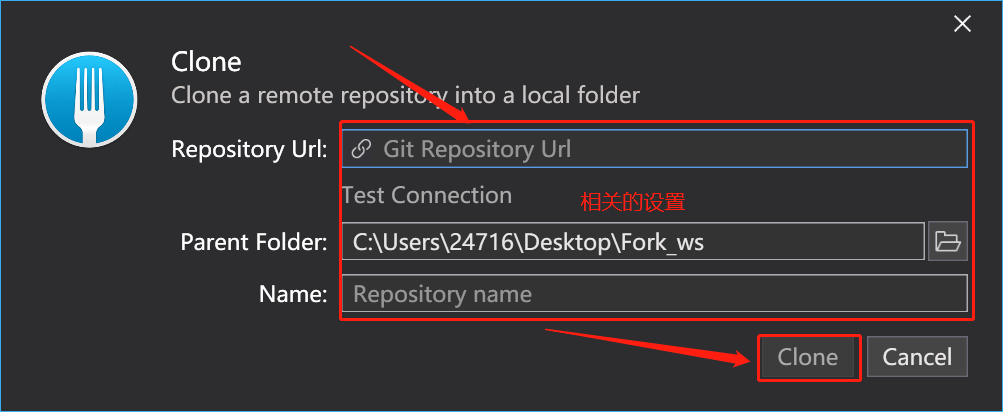
## Fork文件下载与上传

### Fork文件下载

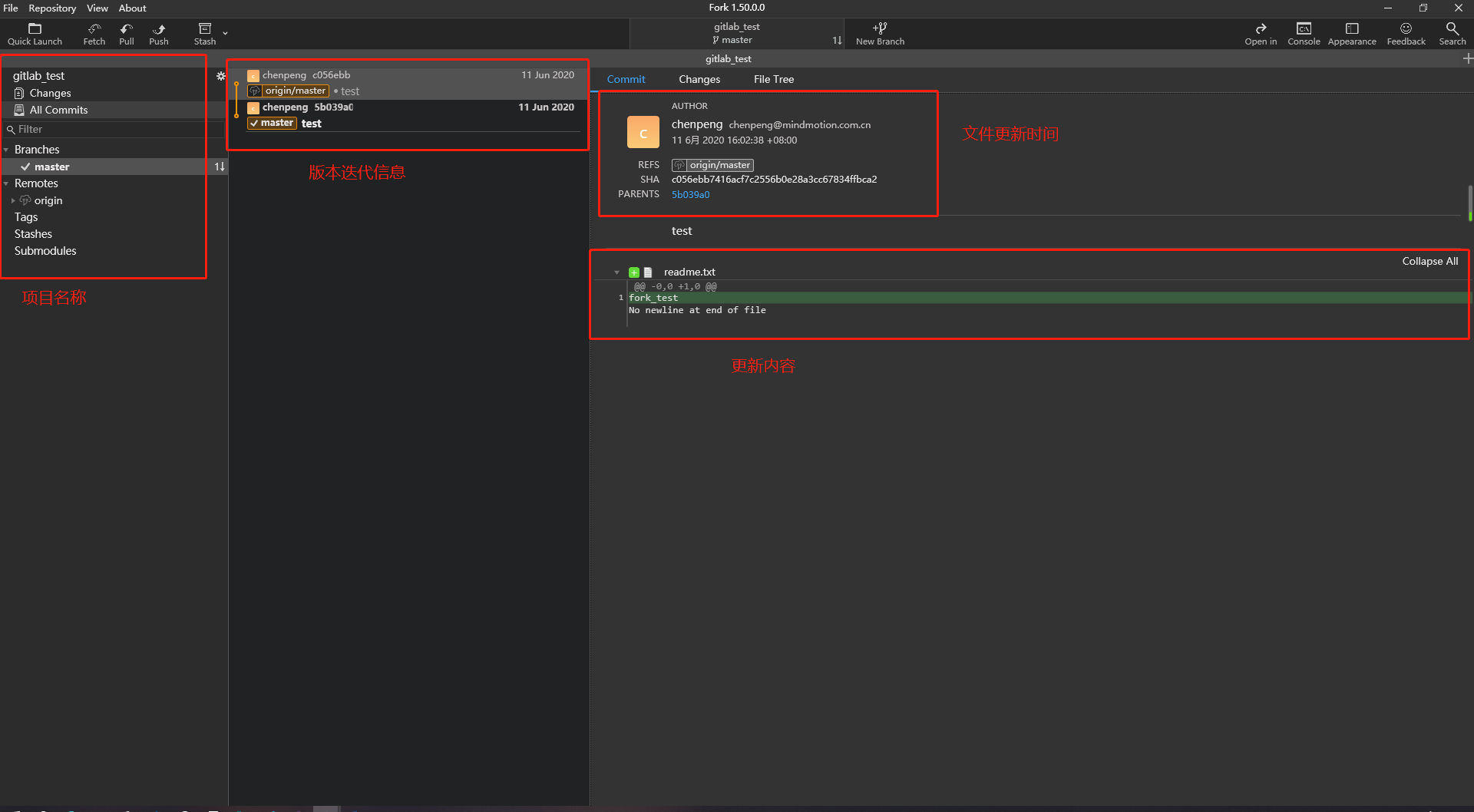
在file选项卡中找到clone选项。



设置文件下载的URL与存放的路径。其中URL就是我们从网上下载的项目的网址，Parent Folder就是克隆到本地的目录，在桌面建立一个Fork\_ws的目录，然后下载项目到此文件夹下面，不用输入URL，直接点击GitLab选择需要下载的Repository URL克隆即可。

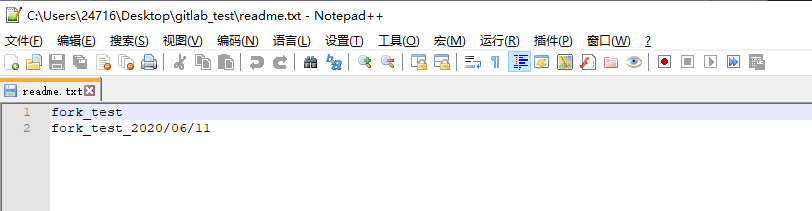


当文件克隆好了以后在左边就会有我们的项目的相关的信息，我们可以在客户端主界面查看项目的分支，版本迭代信息，各前版本的内容。在更新内容中显示的是当前的版本与上一版本的文件内容变更。

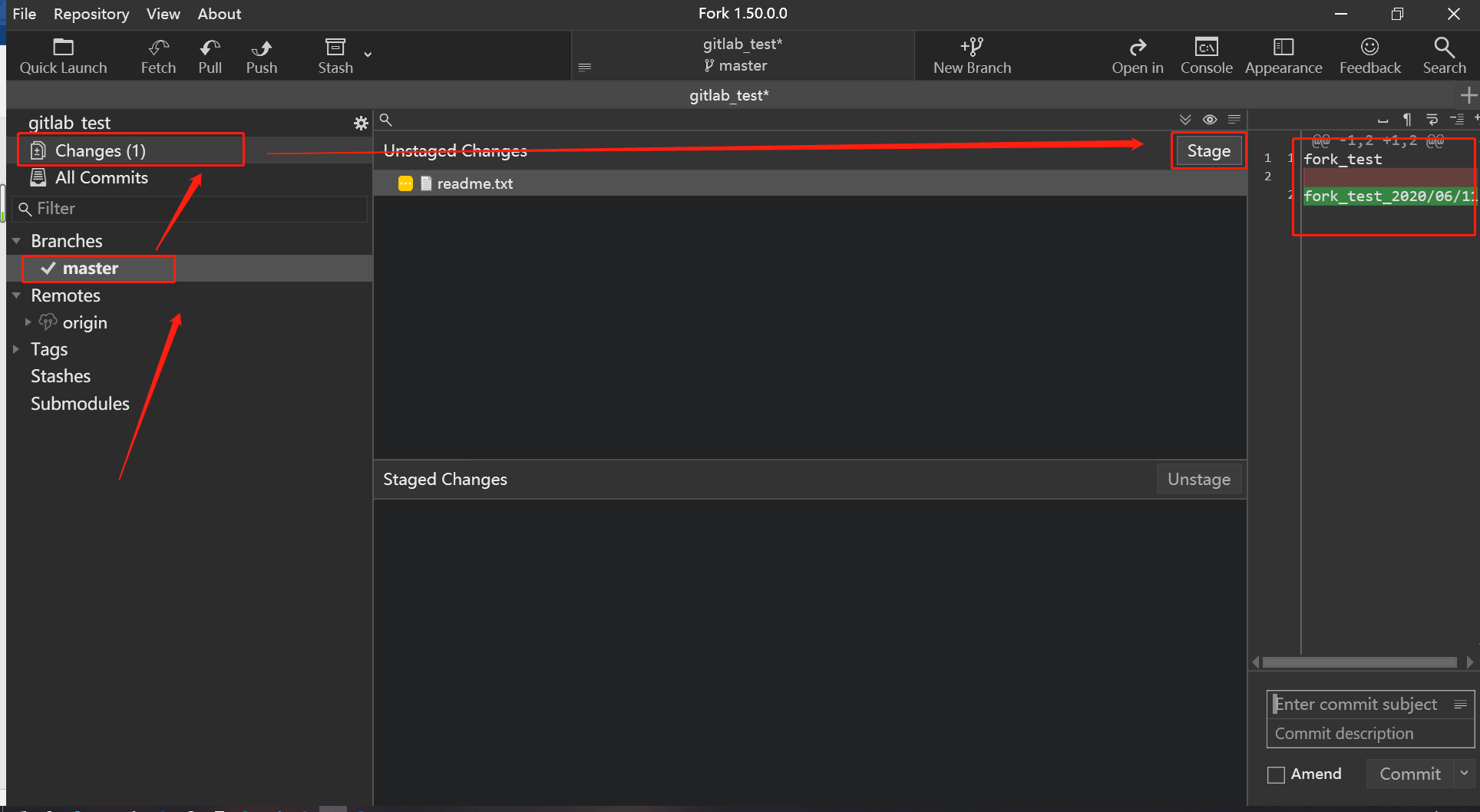


### Fork文件上传

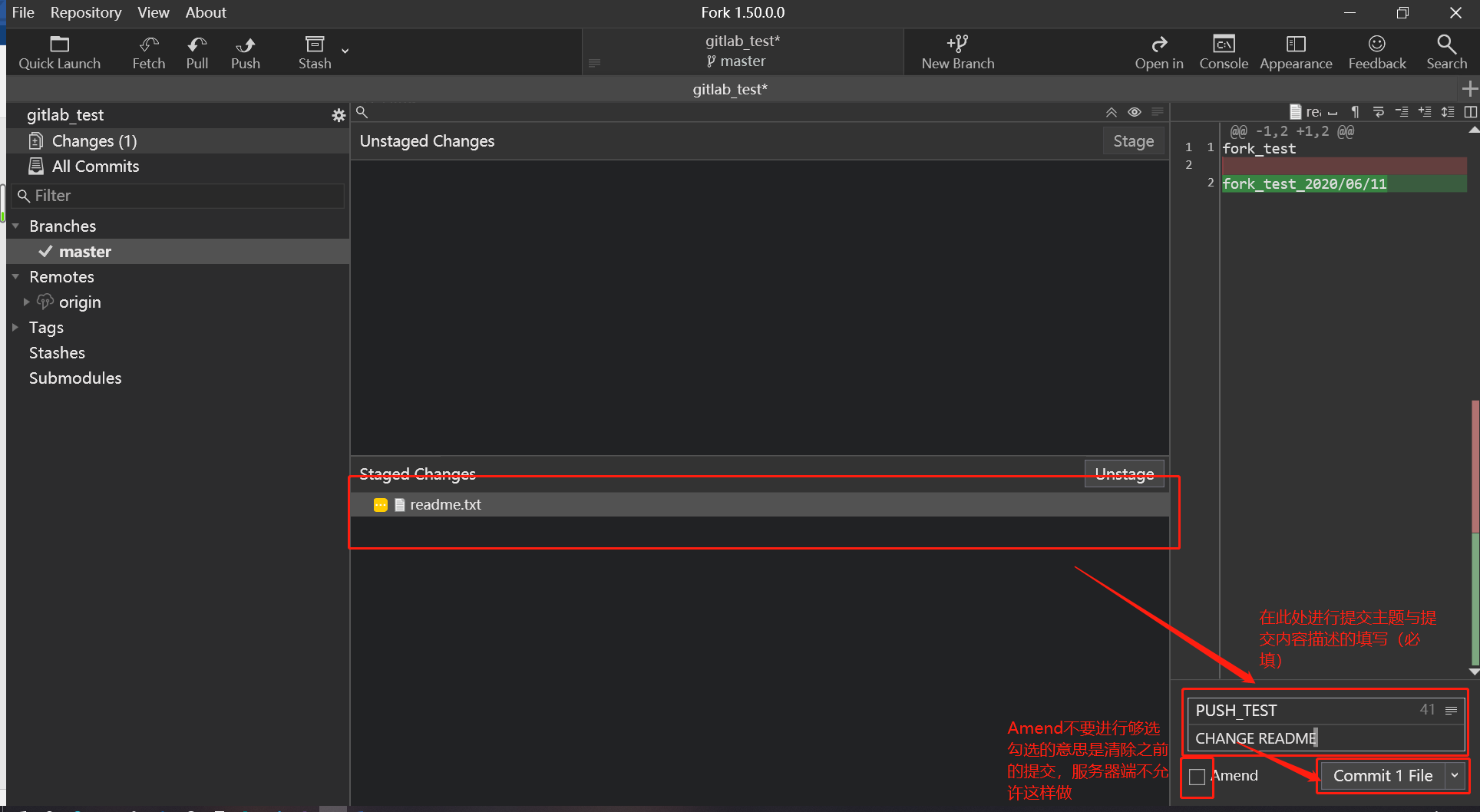
1. 修改我们使用Fork从gitlab仓库下载的readme.txt文件，具体的修改如下：



然后我们便可以在fork的changes选项卡上看到选项卡已经改变。

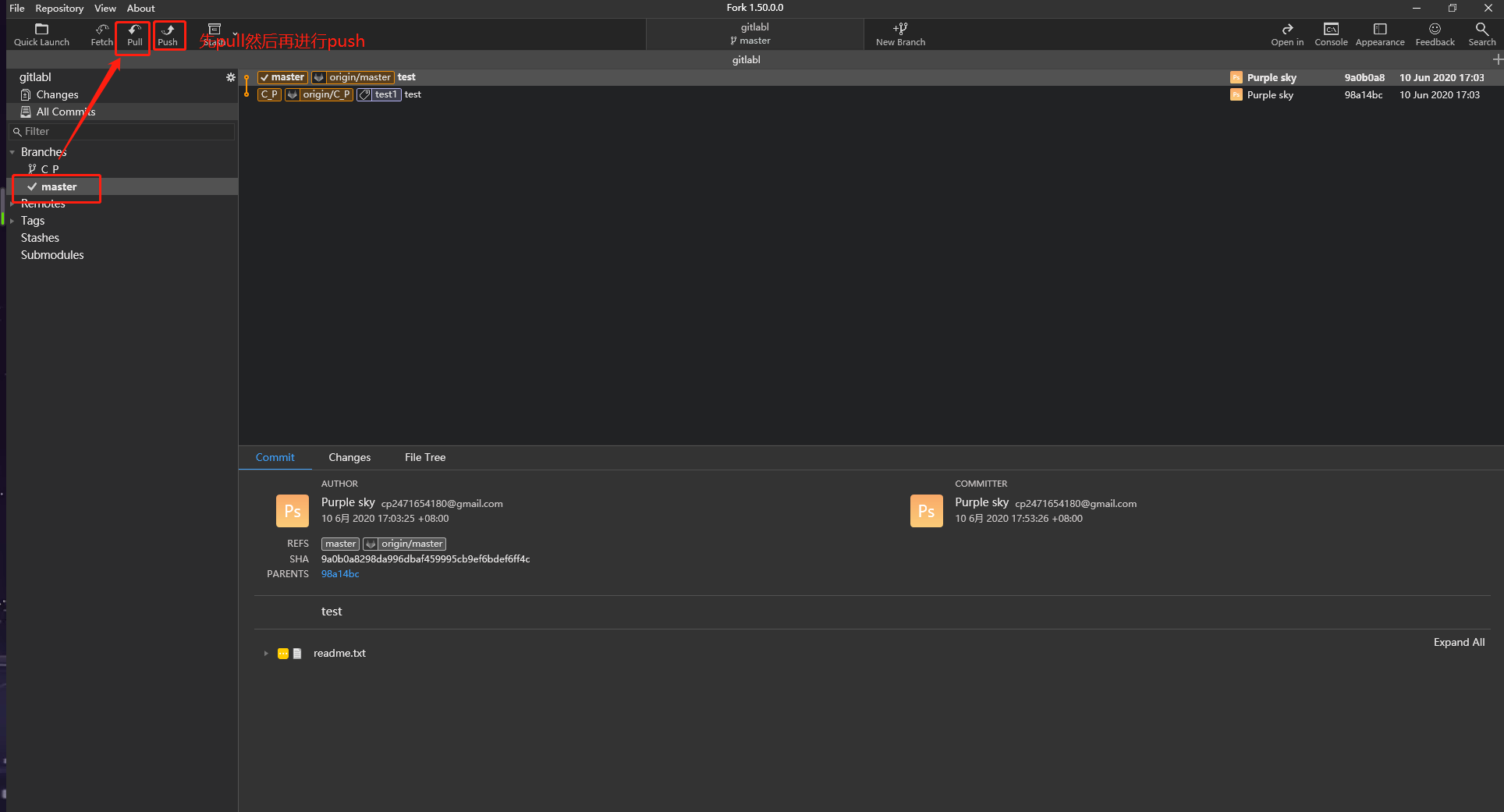


提交到FORK的本地仓库，选中changes，点击stage声明变化，填写文件变更的相关描述。

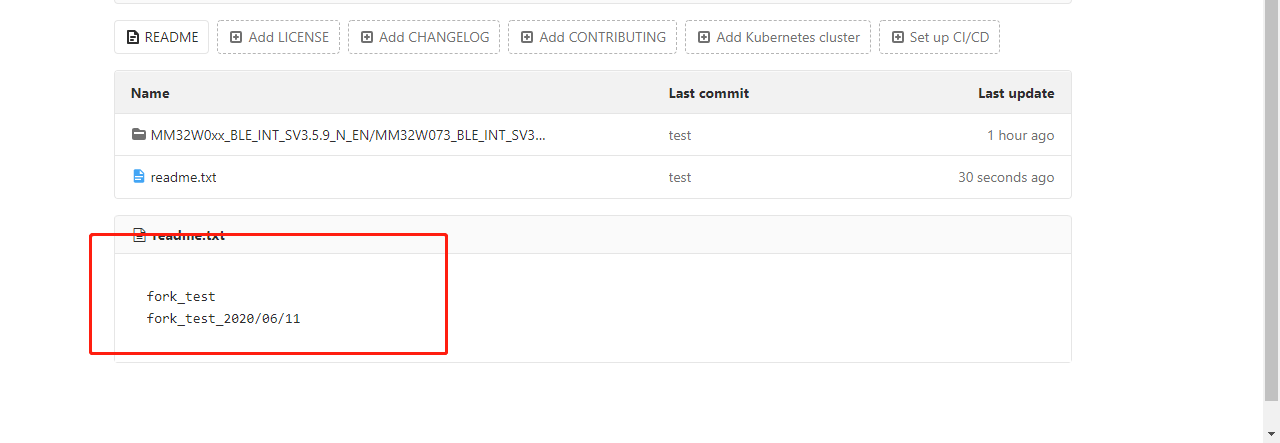


点击amend last commit然后提交到本地仓库。

上传文件到gitlab，我们在将文件同步到本地的git仓库以后，然后将我们修改以后的文件上传到gitlab。



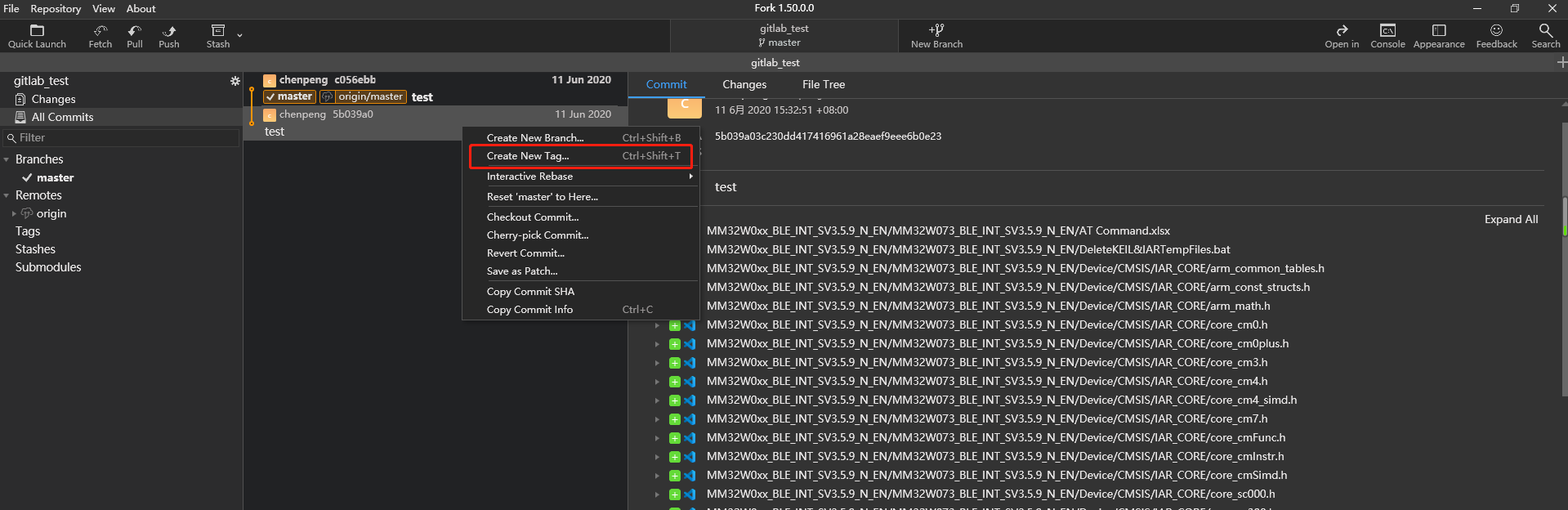
，然后我们直接点击上面的PUSH就可以将代码上传我们的GitLab，当然我们也可以进行先进行pull，然后才能进行push，先将我们本地的代码更新到最新然后再push，这样可以防止出现冲突，如果有冲突在本地解决，然后上传，最后登录网页端然后进行数据的对比。



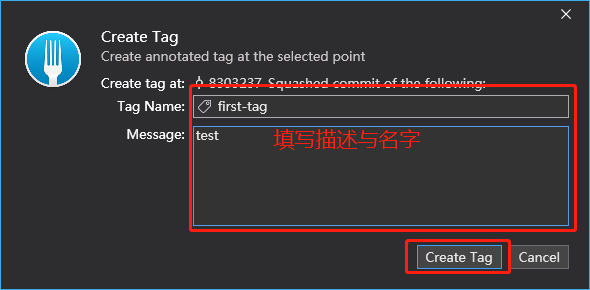
我们可以看到master的文件内容我们push的文件内容是一致的，说明我们操作是正确的。上述操作的都是master进行的push，这样是不需要进行合并分支的，如果是分支上传文件，我们需要在网页端进行分支合并操作。

## 建立一个标签：

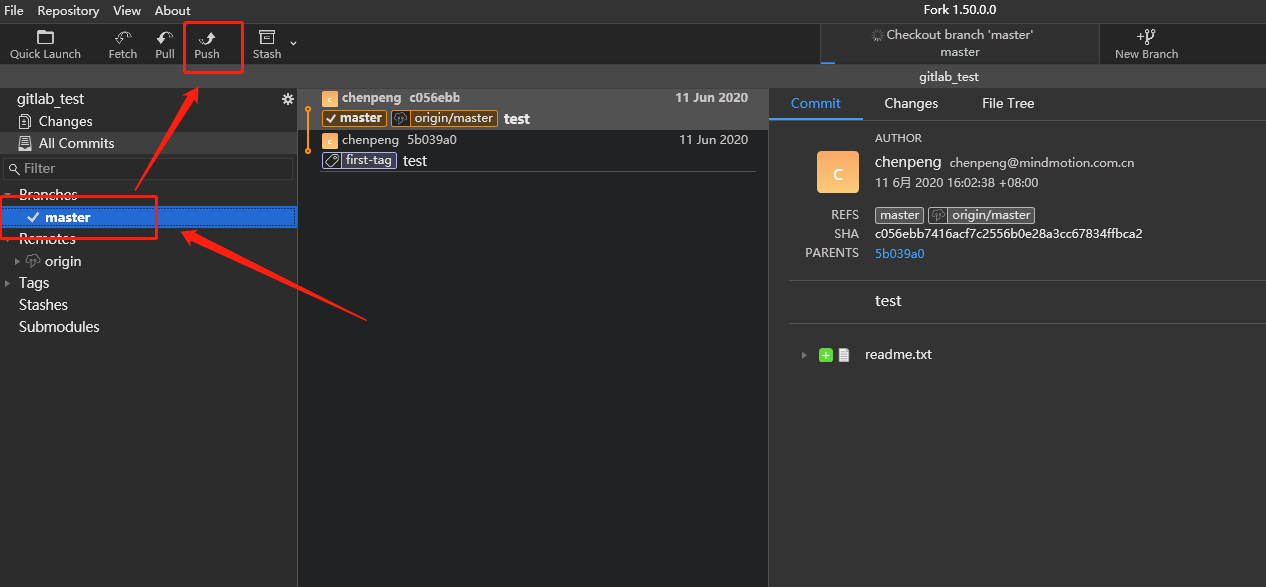
选择需要建立标签的文件版本，右击鼠标打开功能选项卡，选择Create Tag…选项，



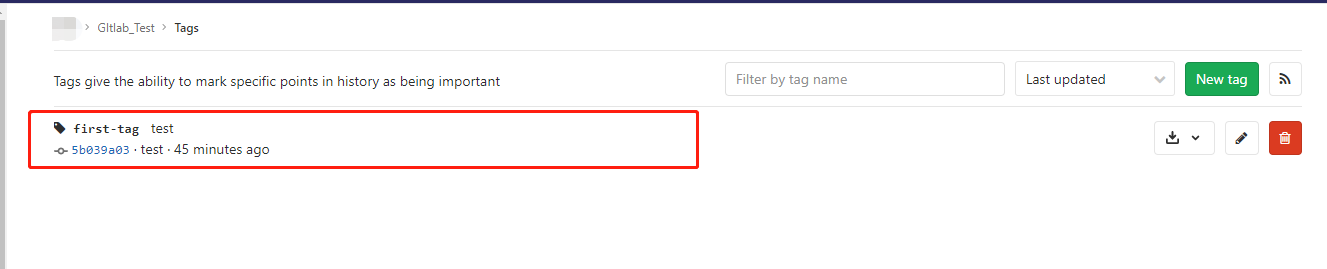
进行有关标签的相关内容进行填写。



更新内容gitlab。



然后去GITlab查看标签是否创建成功



出现以上的内容则说明我们的标签创建成功。

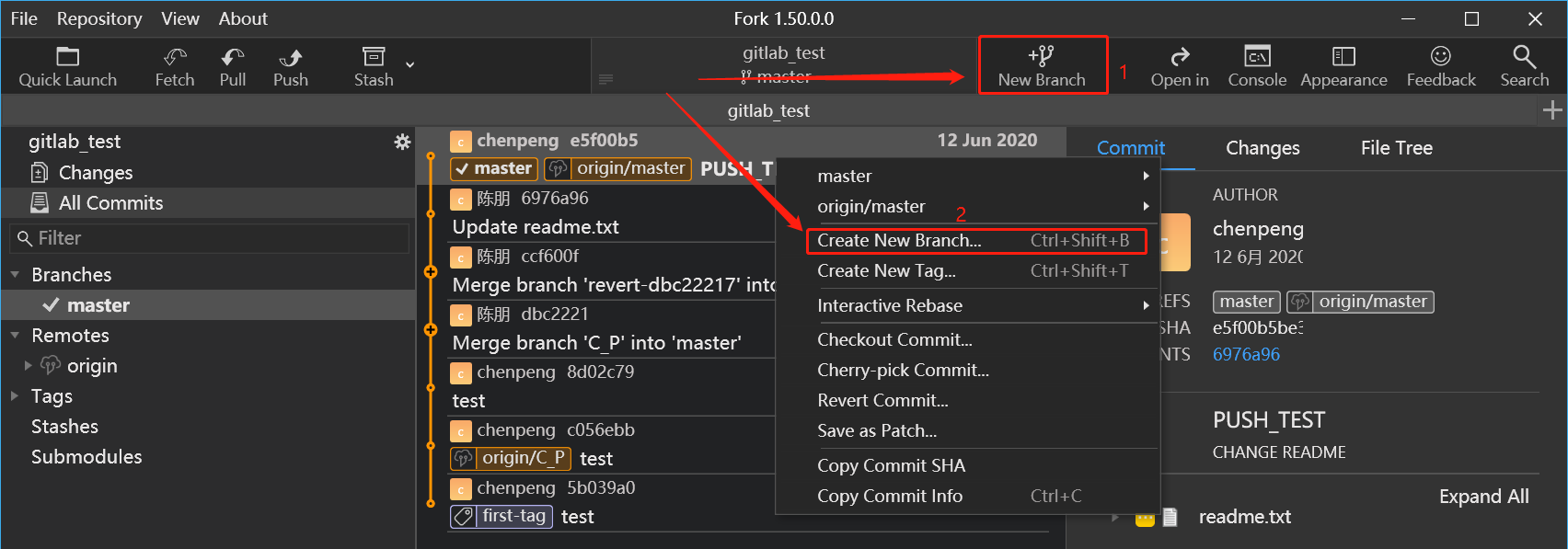
## Branches创建与合并

### 创建Branches

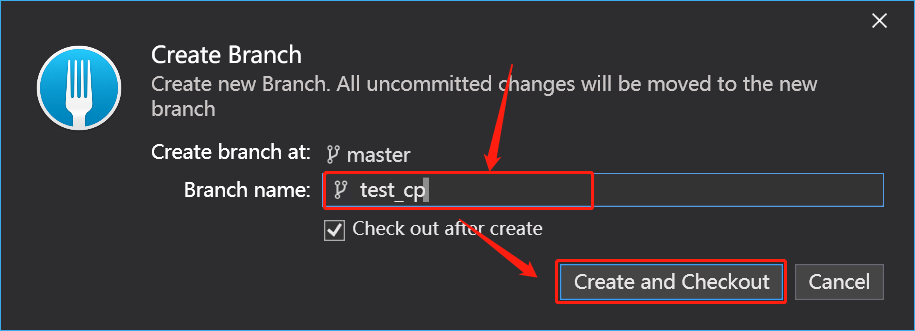
创建分支有Creat New Branche与New Breache两种方式：

第一种方式始在master下面打分支，不能打分支到其他的地方。

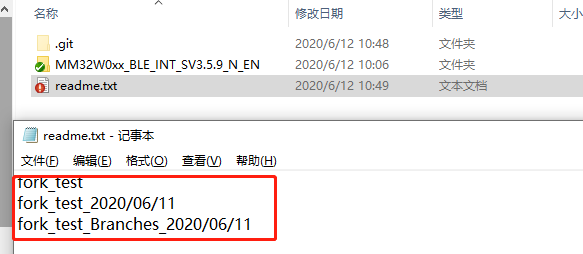
第二种方式可以在任何版本下打分支。



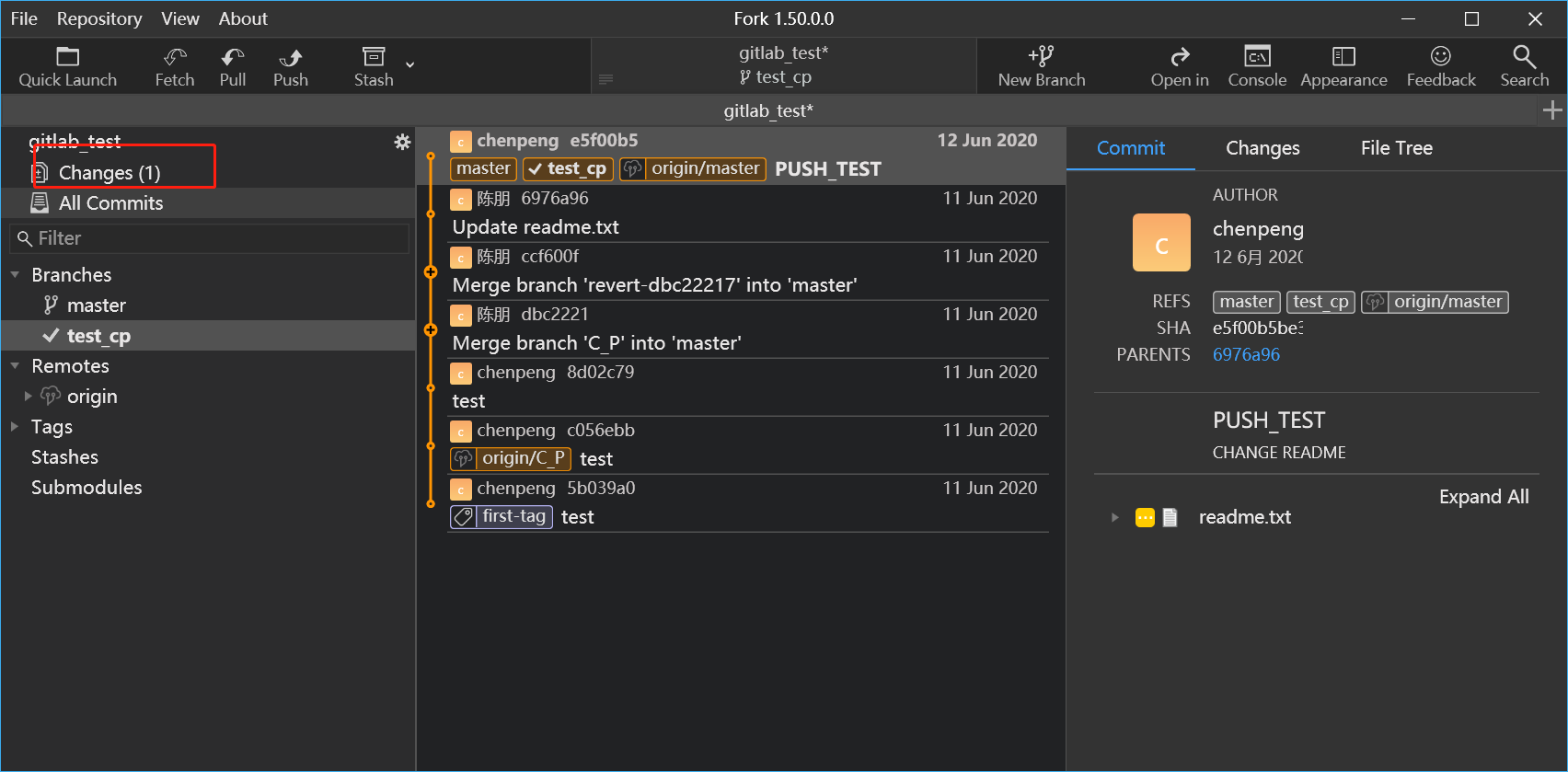
我们就在主目录下面打分支。

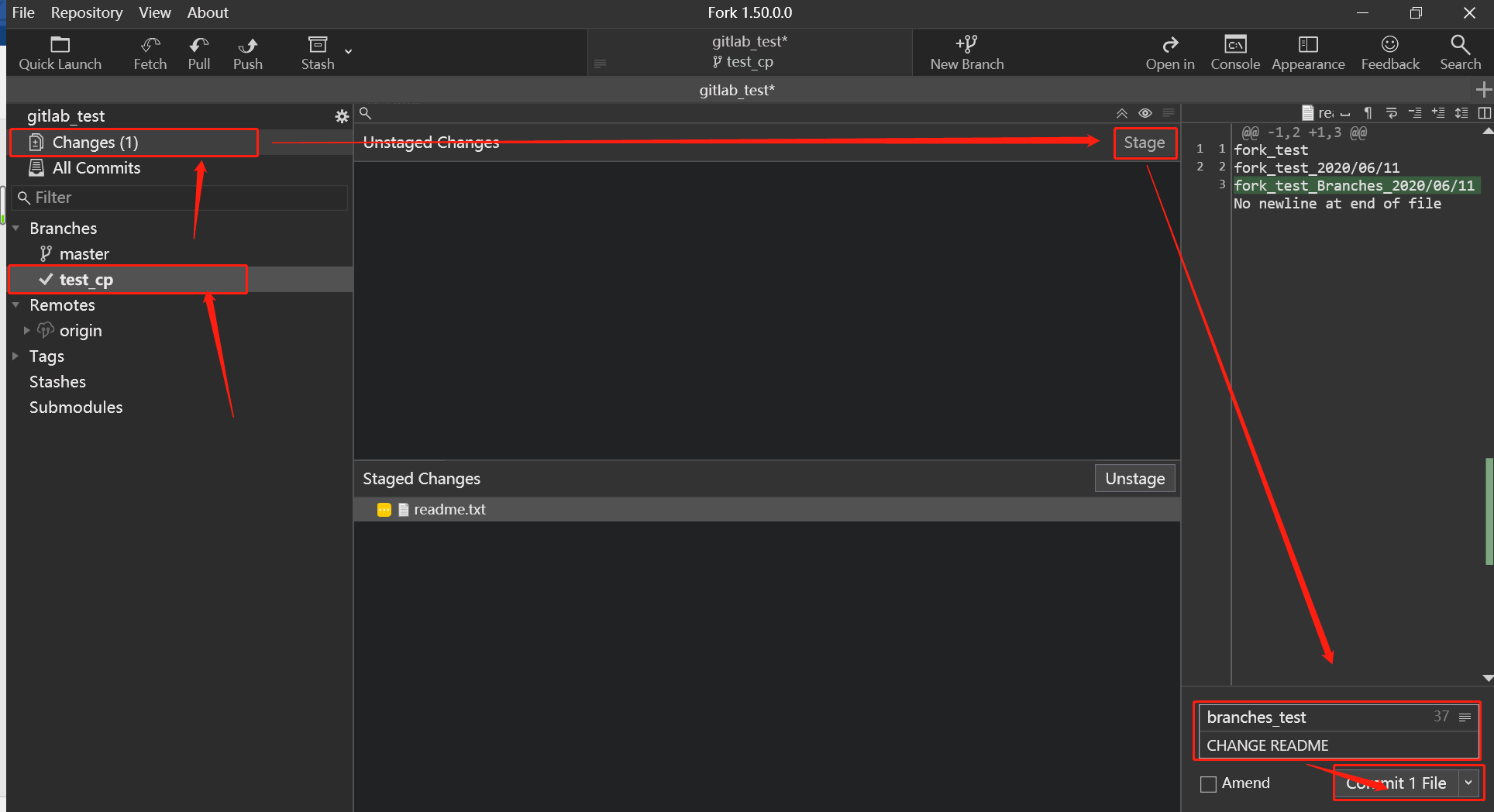


然后修改文件

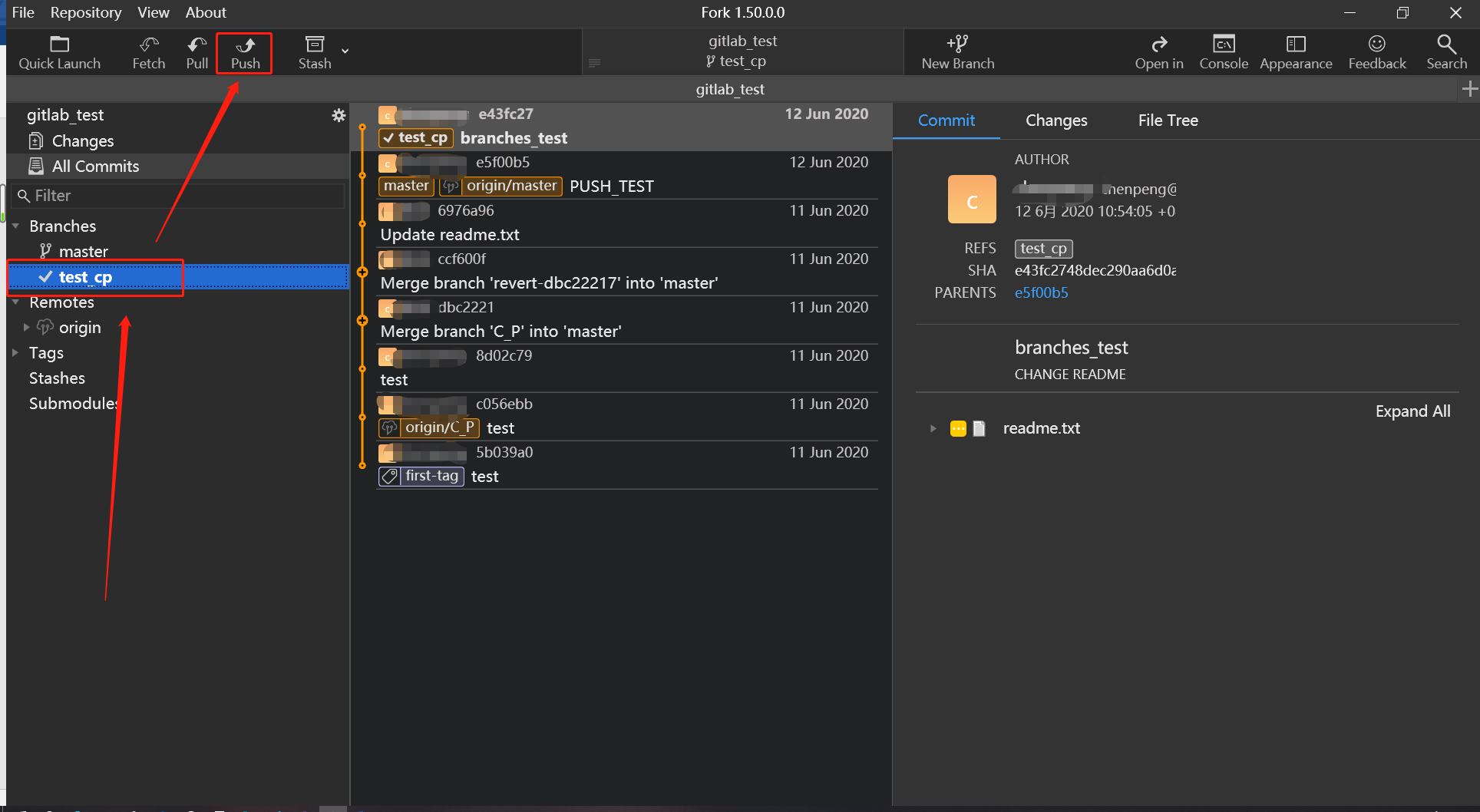


### Branches上传文件





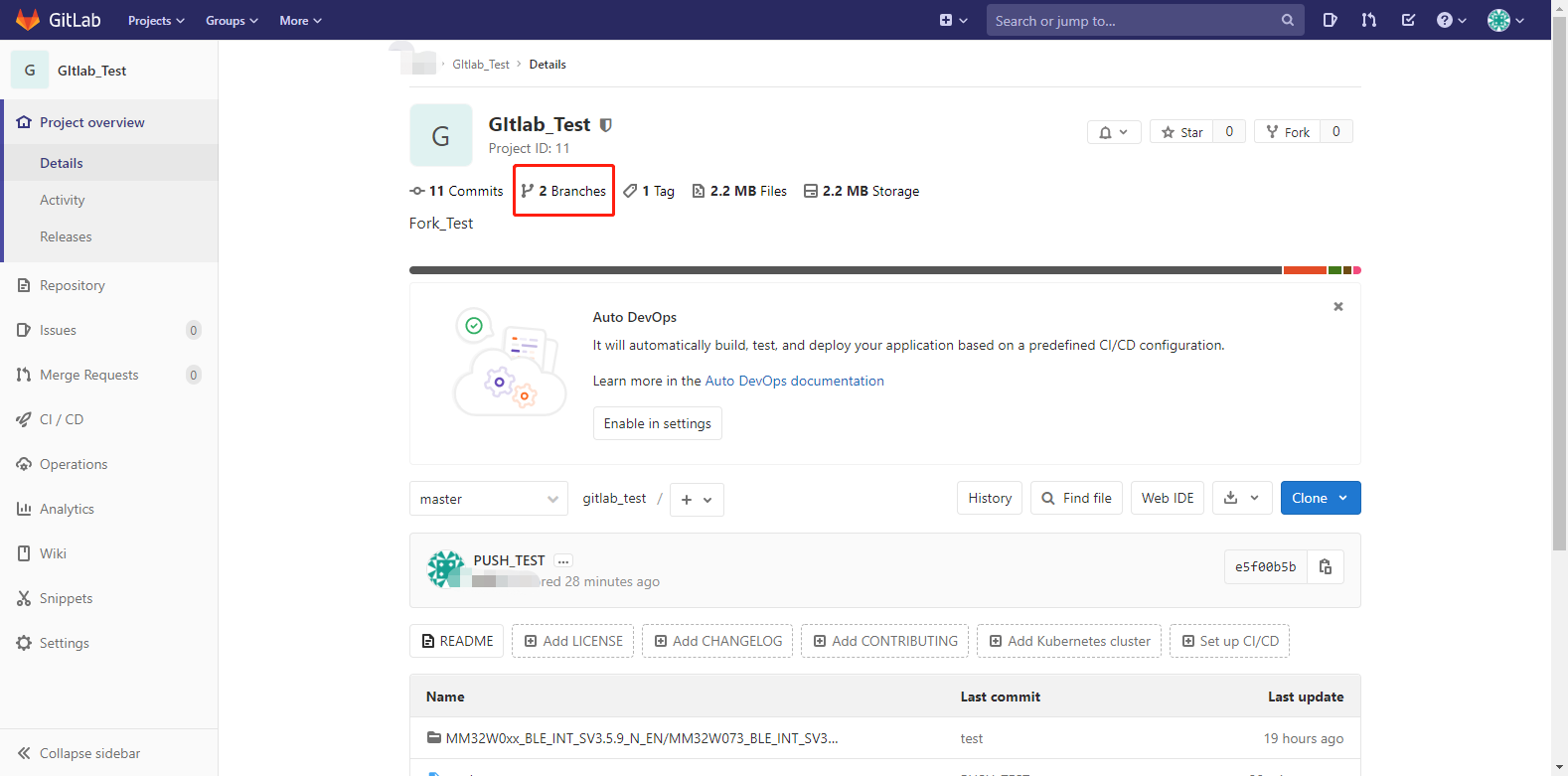
采用同样的方式上传到GitLab。



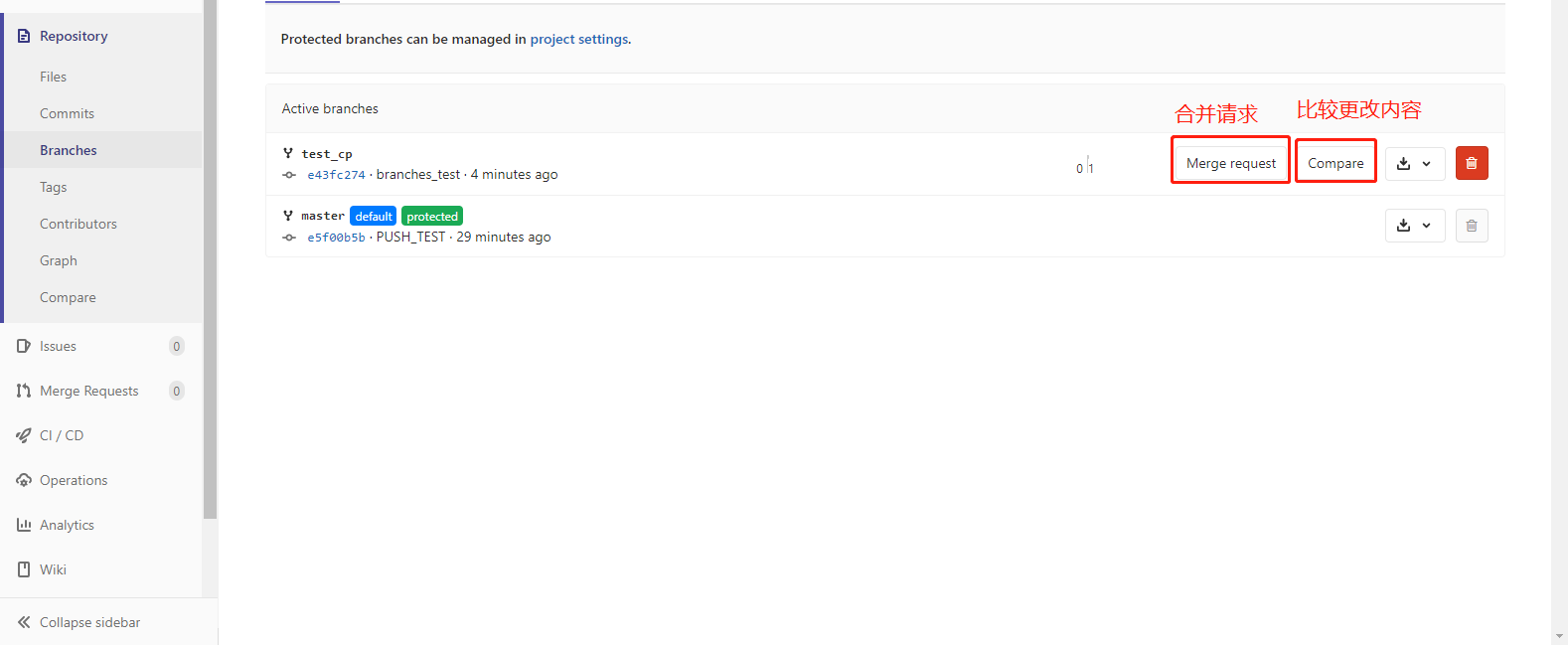
我们修改一个文件然后使用分支上传到GITLAB。

### Branches合并

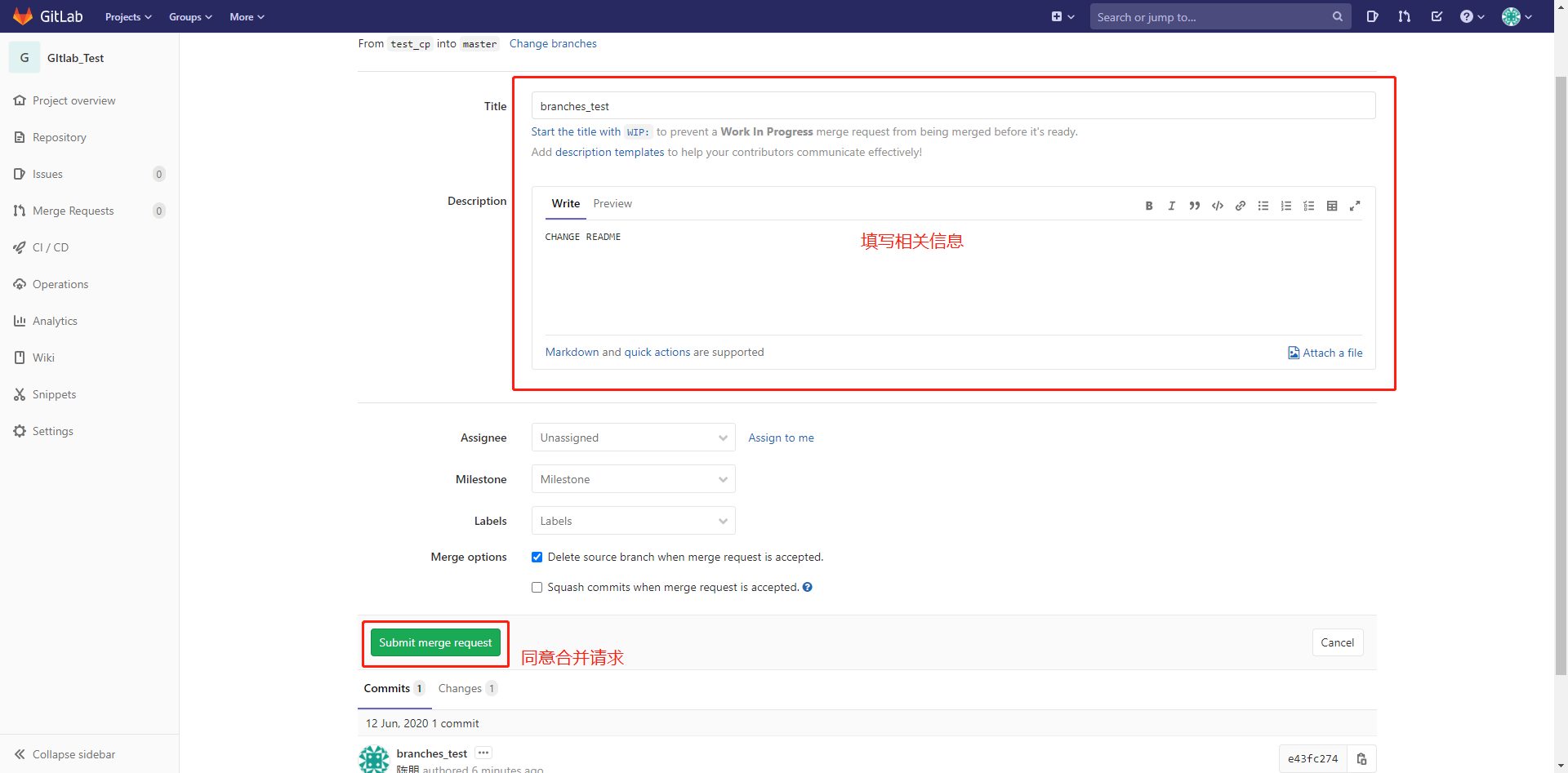
进入我们的刚才上传文件的仓库，找到并进入Branches



进入Branches我们便可以看到test\_cp上传的文件，我们可以对其进行合并操作或者比较操作。

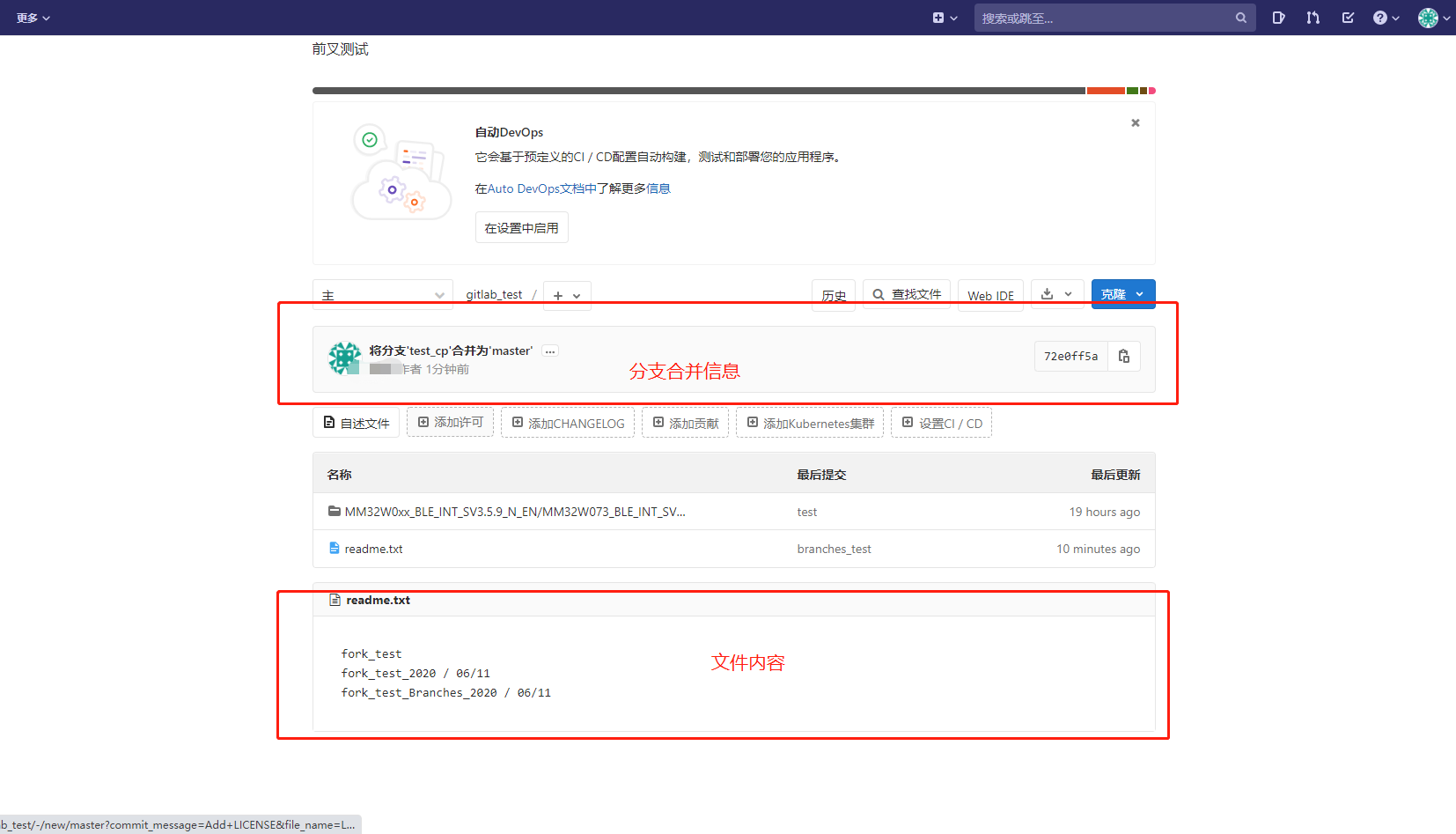


点击merge request进行分支的合并。



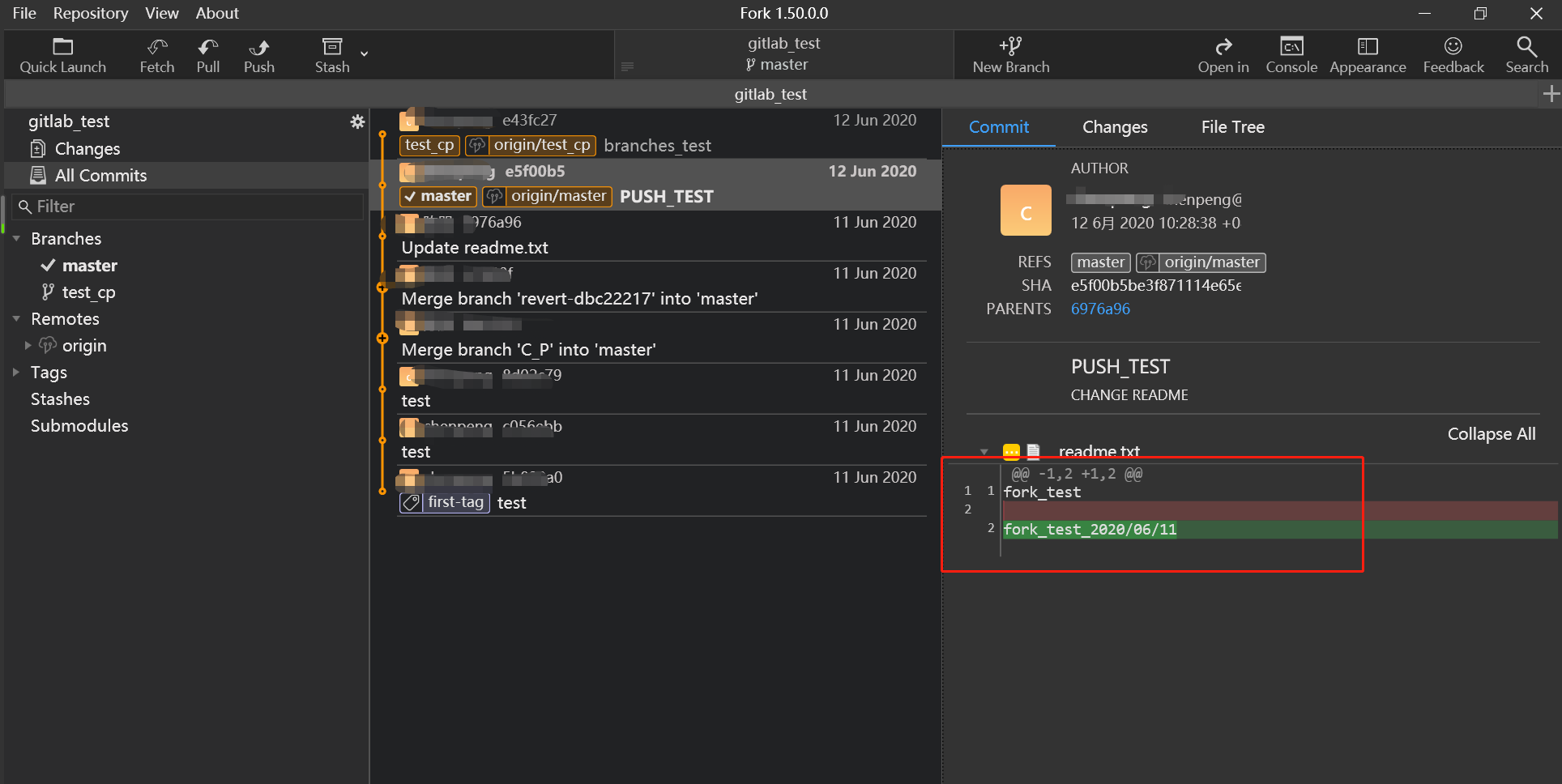


然后去主分支查看我们的文件是否进行了更新：

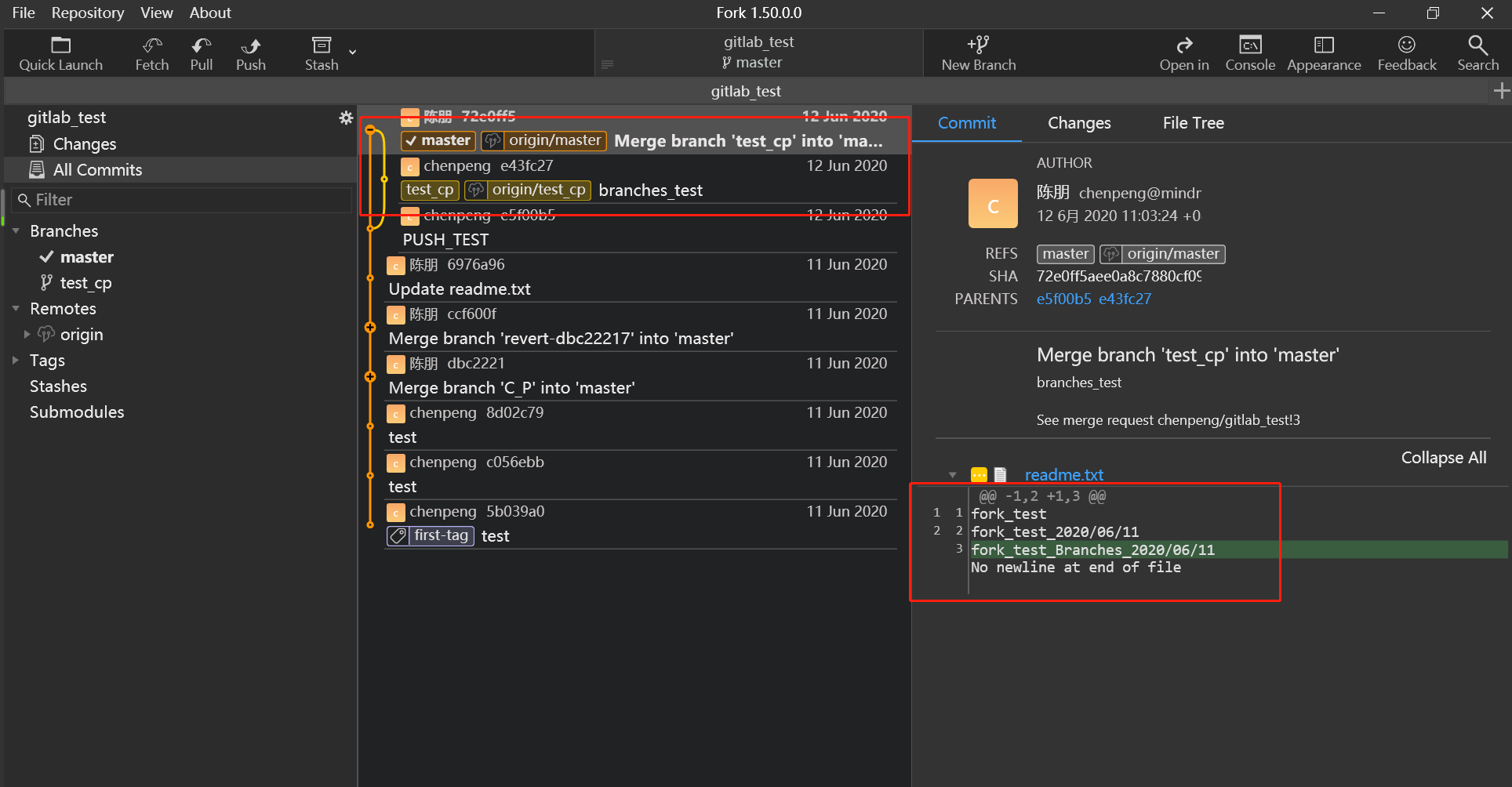


以上我们可以看到我们的master的文件进行了变更。说明我们分支上传文件ok。

下面我们查看以下本地仓库的文件，我们发现我们本地方库的文件并没有变更，是因为我们本地仓库的文件没有进行更新。



我们pull一下就可以进行版本信息的更新。



我们可以看到我们将test\_cp的分支合并到了master，而且master的内容也更新到了最新，这就是pull的作用。

## Fork快捷键：

* F2 -重命名存储库
* Delete -删除存储库
* Enter -开放资料库
* F5 -刷新
* Ctrl+Shift+N -初始化新的存储库
* Ctrl+N -克隆新的存储库
* Ctrl+O -开放资料库
* Ctrl+Shift+F -获取
* Ctrl+Alt+Shift+F（Ctrl+单击按钮）-快速提取
* Ctrl+Shift+L - 拉
* Ctrl+Alt+Shift+L（Ctrl+单击按钮）-快速拉
* Ctrl+Shift+P - 推
* Ctrl+Alt+Shift+P（Ctrl+单击按钮）-快速推送
* Ctrl+Shift+B -新分支
* Ctrl+Shift+T -新标签
* Ctrl+Shift+H -创建藏匿处
* Ctrl+Alt+O -在文件资源管理器中打开
* Ctrl+Alt+T -在终端中打开
* Ctrl+Enter -提交
* Ctrl+Shift+Enter -提交并推送
* Ctrl+1 -焦点提交消息字段
* Ctrl+F -过滤器
* Ctrl+Alt+Shift+S -暂存/取消暂存所有文件
* Ctrl+O -打开所选文件
* Ctrl+D -在外部差异工具中打开选定的文件
* Ctrl+C -复制所选文件的完整路径
* Ctrl+1 -显示更改视图（第二次按下将聚焦提交字段）
* Ctrl+2 -显示所有提交视图（第二次按下将跳至HEAD）
* Ctrl+0 -显示头
* Ctrl+P -显示快速启动窗口
* Ctrl+Tab -选择下一个标签
* Ctrl+Shift+Tab -选择上一个标签
* Ctrl+T -打开新标签
* Ctrl+W -关闭当前标签
* Ctrl=/ Ctrl-放大/缩小

# it与GitLab使用说明

## git与Gitlab建立连接上传与下载文件

### ****1.生成ssh key。****

一般情况下，原来使用过git的用户.ssh文件夹都是默认存在的。执行生成ssh key的命令，会生成有关git的密匙与公匙。具体的命令为ssh-keygen -t rsa -C “自己的邮箱地址”，注意将此处的邮箱地址换成自己的，然后按三次回车。这时候在.ssh文件夹下就得到了id\_rsa和id\_rsa.pub文件。其中id\_rsa中的内容便是私钥，id\_rsa.pub中的内容便是公钥，可以在C:\Users\用户目录\.ssh目录下找到这两个文件。

mkdir ~/.ssh

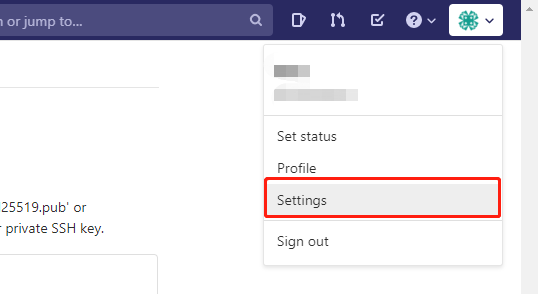
cd ~/.ssh

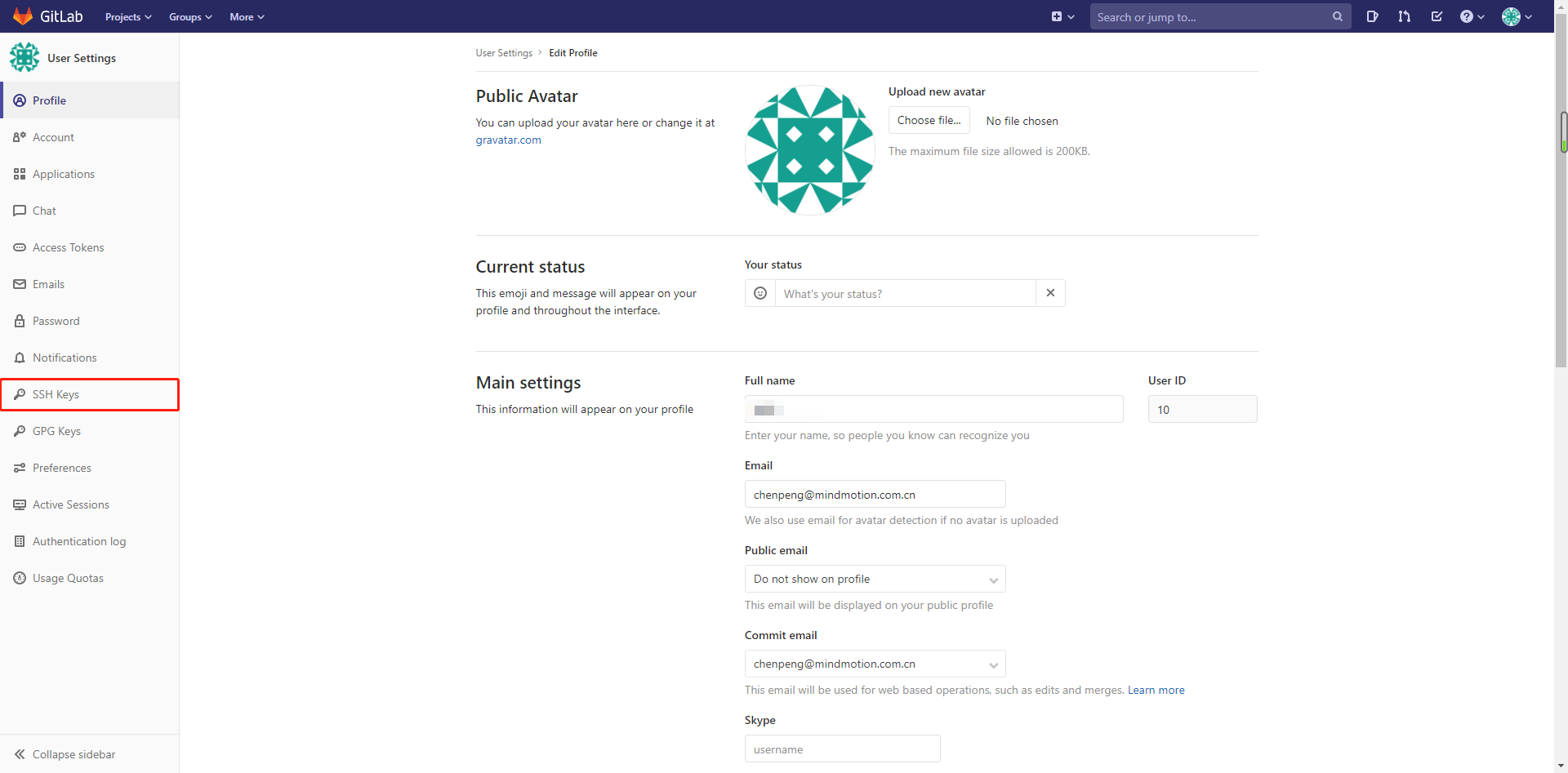
ssh-keygen -t rsa -C “自己的邮箱地址”

### ****2.添加ssh key。****

在gitlab远程仓库中添加ssh key。首先你要注册一个gitlab账号，登陆之后按如下步骤操作：

1. 点击Settings
2. 找到在左侧菜单中找到ssh key。



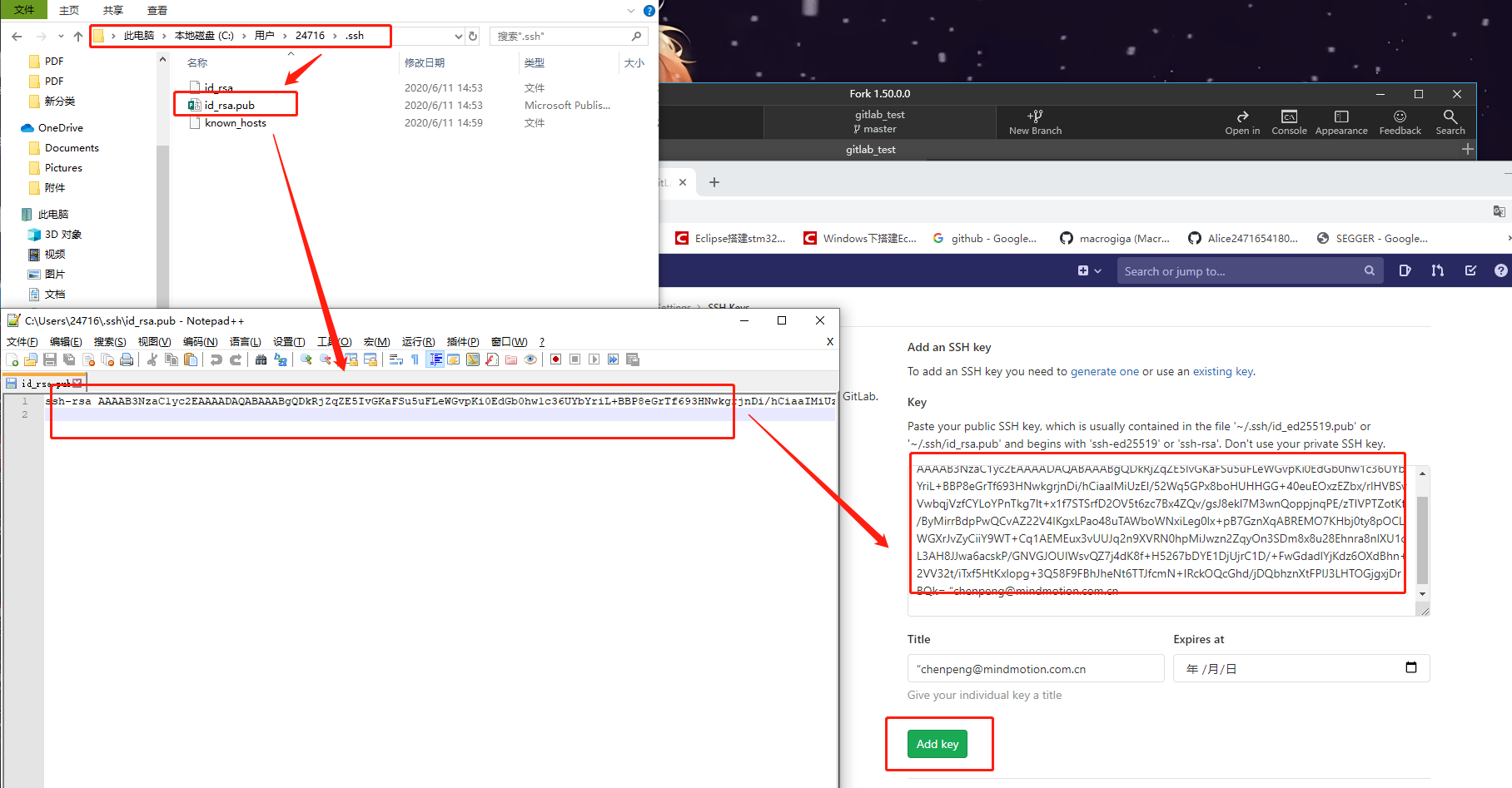


1. 为GitLab添加SSH Key具体的操作如下：

在用户路径下面找到.SSH文件夹。

打开id\_rsa.pub文件。

将内容复制到SSH Keys页面的Key对话框。（源文件为1行，此处为了好看改成了多行）



通过以上的配置，我们的git与Gitlab就建立连接了。

## GIT 查看/修改用户名和邮箱地址

### 用户名和邮箱地址的作用

用户名和邮箱地址是本地git客户端的一个变量，不随git库而改变。

每次commit都会用用户名和邮箱纪录。

### 查看用户名和邮箱地址：

git config user.name

git config user.email

### 修改用户名和邮箱地址：

git config --global user.name "username"

git config --global user.email "email"

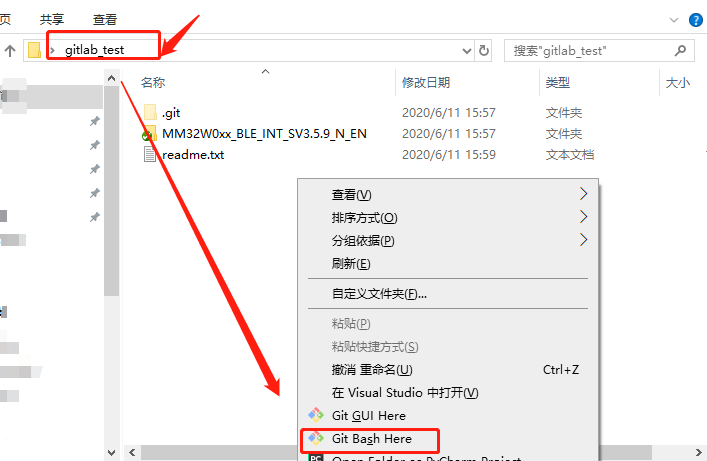
## GitLab文件上传与下载

#### 仓库文件上传

我们先在桌面把我们创建的工程Clone下来，进入到clone的文件夹，并编写的一个readme.txt文件，编程的文本可以随意编写。



在GitLab\_Test（clone下来的文件夹）文件夹下面Bash。

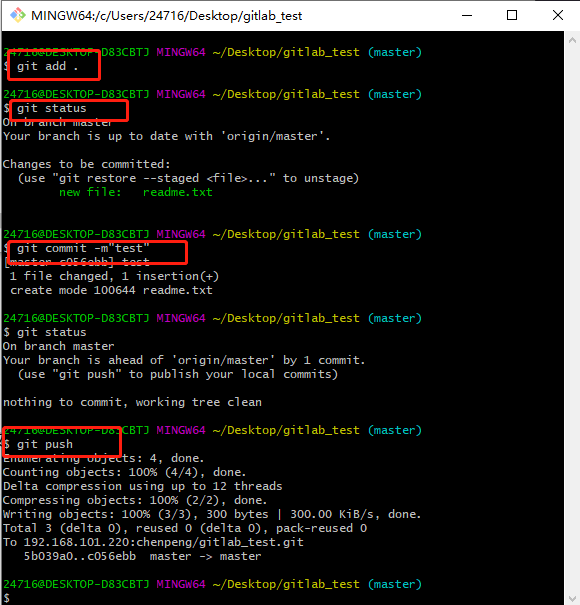


使用命令行进行代码的上传，具体的操作如下：

git add .//需要向上提交的文件（.代表当前文件夹下面的所有文件，也可以使用文件名字来代替.代//表需要上传的文件）

git commit -m"GitLab test file"//添加自己的描述

git push -u origin master//上传到文件夹。



注：

在进行文件上传的时候我们需要注意权限的问题，如果没有权限是不能进行上传操作的会出现权限不足的错误。上传命令发出后，若出现：  failed to push some refs to git 错误，主要原因是gitlab中的README.md文件不在本地代码目录中，因此，可通过如下命令进行代码合并（pull=fetch+merge）:

git pull --rebase origin master

执行完此代码后可以看到本地代码库多了README.MD文件，此时，再执行语句：

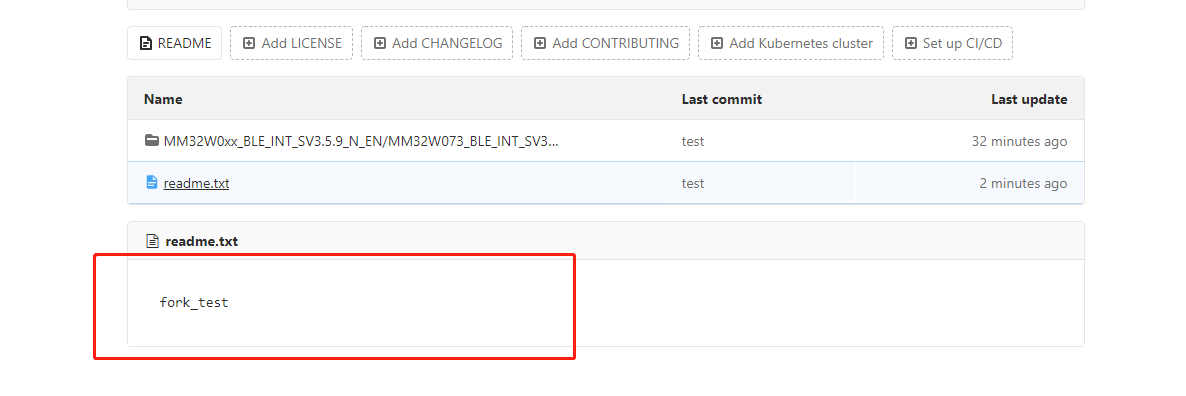
git push -u origin master

即可完成

如果当面的文件夹的其它级目录存在隐藏问价夹会在执行git add .时候出现报错<filename>' does not have a commit checked out fatal: adding files failed

需要将隐藏的文件夹删掉。

打开gitlab的主页查看我们上传的文件：



文件正确，说明我们操作成功了。

#### 文件下载：

执行，命令

git clone git@192.168.101.220:xxxxxxxx/gitlab\_test.git

git clone +需要下载的文件的URL, git@192.168.101.220:xxxxxxxx/gitlab\_test.git是我自己的URL，可以替换成你们的仓库的URL。

