

Git 操作命令集合

单人操作:

- 1.sudo apt-get install git ,安装 git ,并创建 git 密码
- 2.git, 查看安装结果,有提示则证明安装成功
- 3.git init , 创建本地仓库
- 4.git config user.name '张三', 配置 git 提交的用户名
- 5.git config user.email '123@qq.com', 配置 git 提交的邮箱
- 6.git status, 查看工作区的文件状态
- 7.git add 文件名, 添加指定文件名到暂存区
- 8.git add . 添加所有的改动文件到暂存区
- 9.git commit -m '版本信息描述', 添加当前版本说明
- 10.git commit -am '版本信息', 直接添加到暂存区并提交到 git 仓库
- 11.git log, 查看详情历史版本
- 12.git reflog, 查看简单历史版本
- 13.git reset --hard HEAD^ 回退到上个版本
- 14.git reset --hard HEAD~1 回到上个版本
- 15.git reset --hard 版本号, 跳转到指定的版本号
- 16.git checkout 文件名, 撤销指定文件的修改,工作区
- 17.git reset HEAD 文件名, 撤销暂存区的代码
- 19.git diff HEAD HEAD^ -- 文件名,对比版本库
- 20.rm 文件名, 删除文件,通过下面 21 行命令撤销
- 21.git checkout -- 文件名, 撤销删除
- 22.git rm 文件名, 删除本地文件, 并提交,通过下面的
- 23 行命令撤销
23. git reset --hard HEAD^,撤销

多人开发:

- 23.git clone 地址, 克隆远程的代码到本地
- 24.git push, 推送到远程仓库

25.git config --global credential.helper cache 十五分钟有效期

26.git config credential.helper 'cache --timeout==3600' 一个小时有效期

27.git config --global credential.helper store 长期有效

28.git pull ,拉取远程代码到本地目录

标签

29.git tag -a 标签名 -m '标签描述 v1.0', 本地标签

30.git push origin 标签名, 将本地标签版本推送到远程端

31.git tag -d 标签名, 删除本地标签

32.git push origin --delete tag 标签名, 删除远端的标签名

分支

33.git branch, 查看当前分支

34.git checkout -b 分支名, 切换到指定分支

35.git push -u origin 分支名, 推送本地分支跟踪远程分支

36.git checkout master/dev 切换到 master 主分支/子分支

37.git merge 分支 A, 合并指定分支 A 到主分支中

1.新建项目



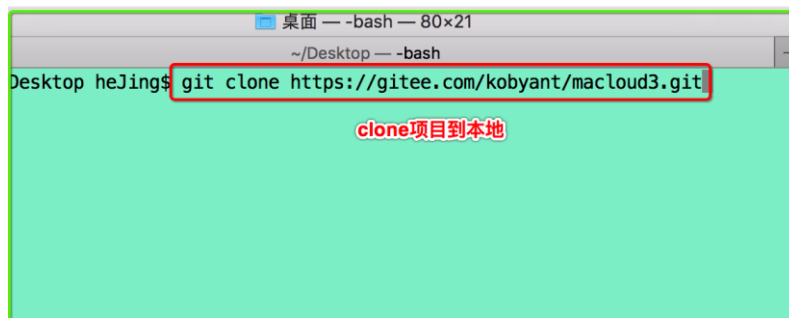
2.填写项目说明信息



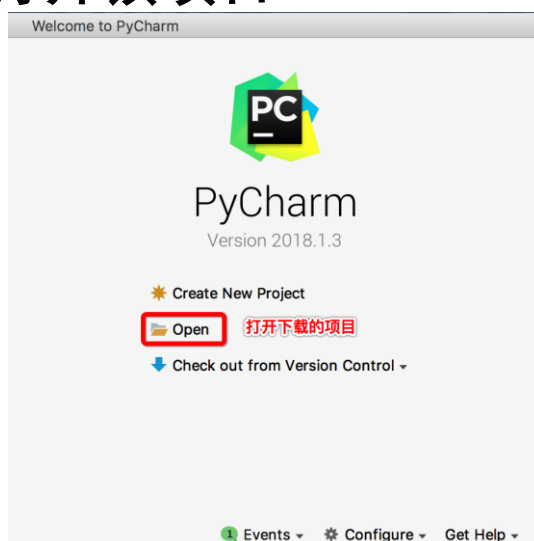
3.复制项目下载地址



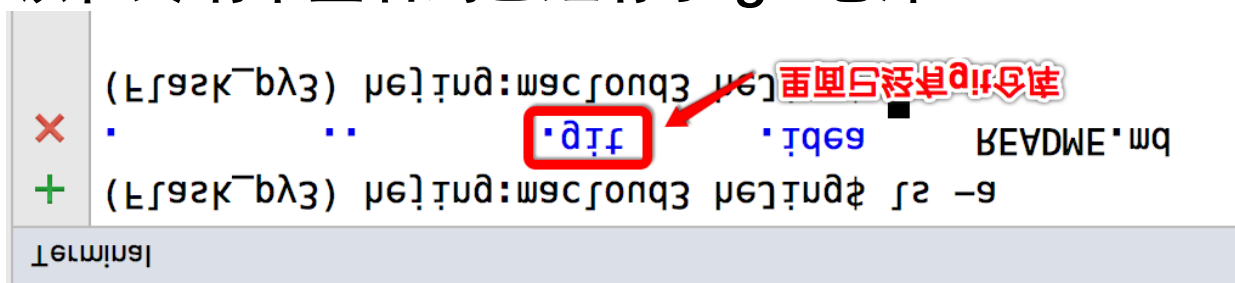
4.clone 项目到本地电脑



5.使用 pycharm 打开该项目



6.可以在终端中查看到已经有了 git 仓库



7.创建.gitignore 文件,用来忽略不需要提交的内容(.idea / *.pyc)

```
(Flask_py3) hejing:macloud3 heJing$ touch .gitignore
(Flask_py3) hejing:macloud3 heJing$ ls -a
.      ..      .git      .gitignore  .idea      README.md
(Flask_py3) hejing:macloud3 heJing$ git status
On branch master
Your branch is up to date with 'origin/master'.

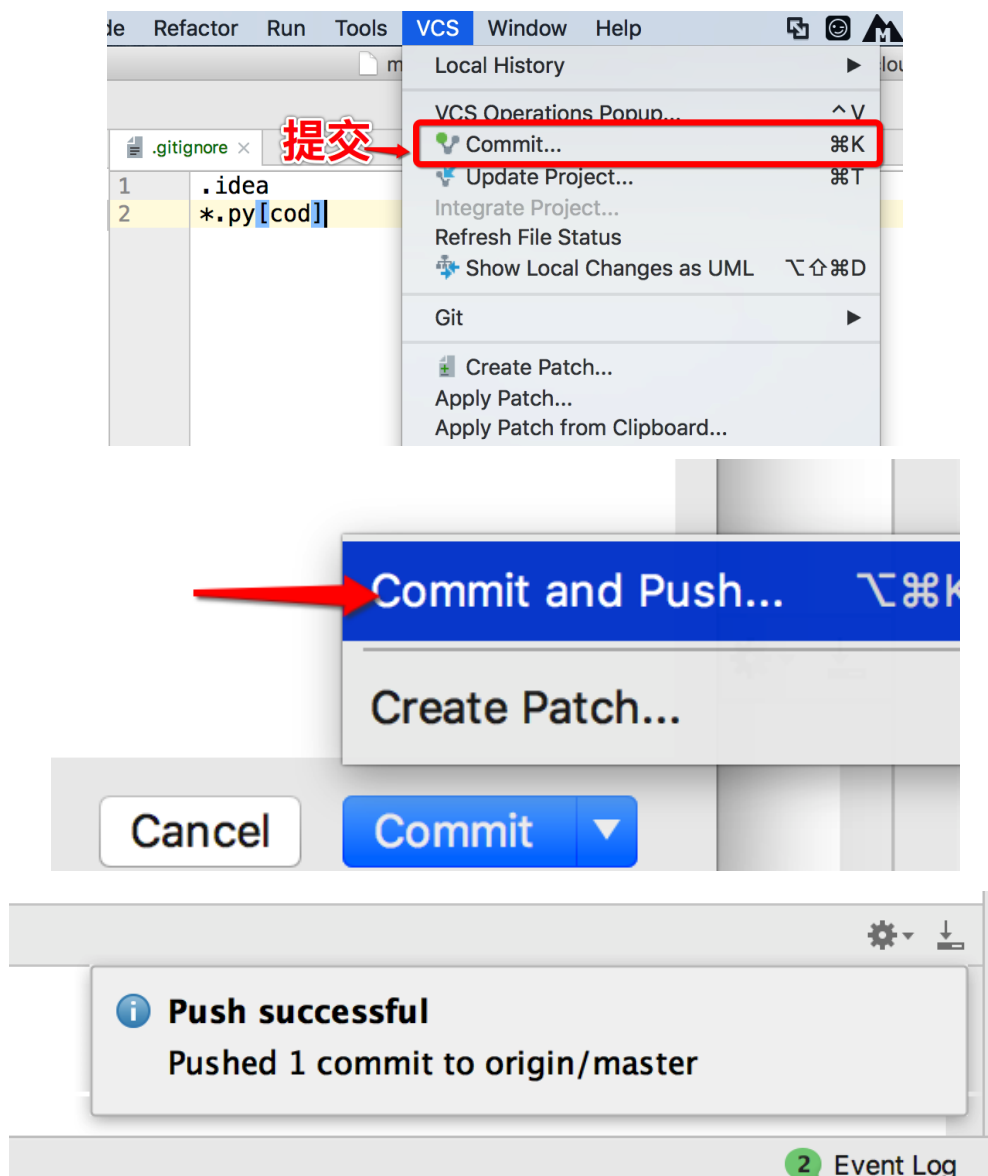
Untracked files:
  (use "git add <file>..." to include in what will be committed)

.gitignore
.idea/

nothing added to commit but untracked files present (use "git add" to track)
(Flask_py3) hejing:macloud3 heJing$
```

需要忽略.idea文件

8.添加,提交,push 到远程仓库中(弹框,则输入码云,账号密码)



8.在码云上可以查看到该项目

日活突破1000000的项目,很好很好 -- 编辑

2 次提交

1 个分支

0 个标签

0 个发行版

2 位贡献者

master

+ Pull Request

+ Issue

文件

克隆/下载

heJing 最后提交于 1分钟前 创建工程,添加ignore

.gitignore

heJing 创建工程,添加ignore

1分钟前

README.md

kobyant Initial commit

19分钟前

本地项目如何上传到 gitlab

平常我们自己写的项目，一直存放在本地总是担心丢失，买服务器搭建一般来说如果仅仅只是为了存档项目也不划算，那么我们可以选择将项目托管到 github 或者 gitlab 上，本文以 gitlab 为例子介绍如何将本地项目上传到 gitlab

工具/原料

git

方法/步骤

1. 首先本地得安装 git，然后切换到需要上传的项目所在路径下，点击鼠标右键

这台电脑 > 工作 (E:) > project

名称

修改日期

类型

大小

xingzhi-boot

2019/7/5 12:08

文件夹

查看(V)

排序方式(O)

分组依据(P)

刷新(E)

自定义文件夹(F)...

粘贴(P)

粘贴快速方式(S)

Git GUI Here

Git Bash Here

授予访问权限(G)

SVN Checkout...

TortoiseSVN

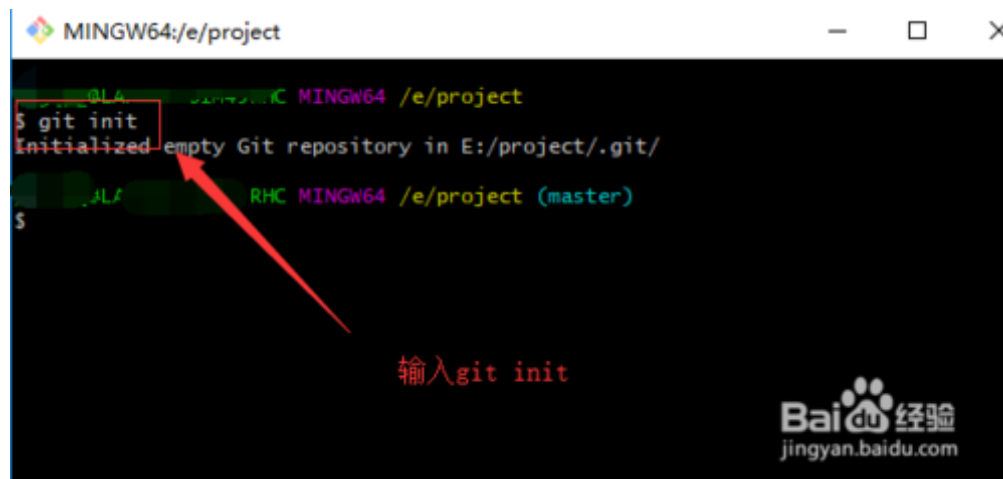
新建(W)

属性(R)

切换至项目所在目录

鼠标右键

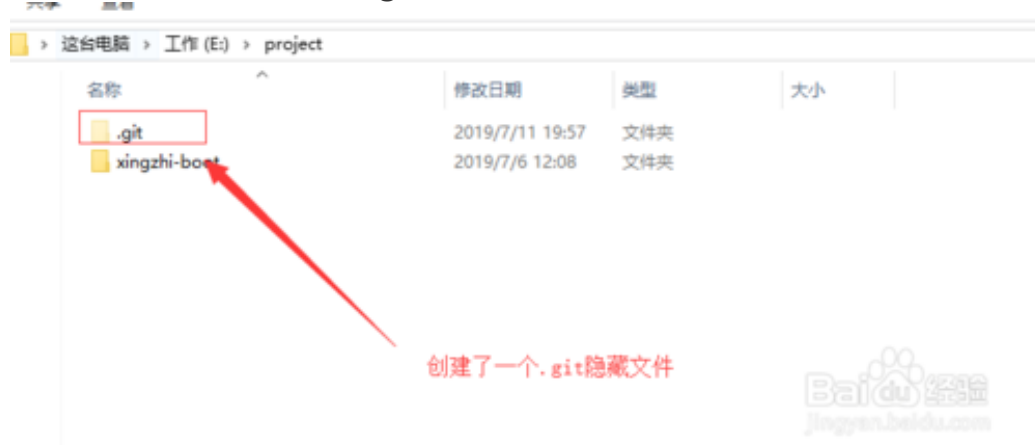
2. 在弹出的框中选择 Git Bash Here, 这时候会弹出一个命令框, 输入命令 `git init`



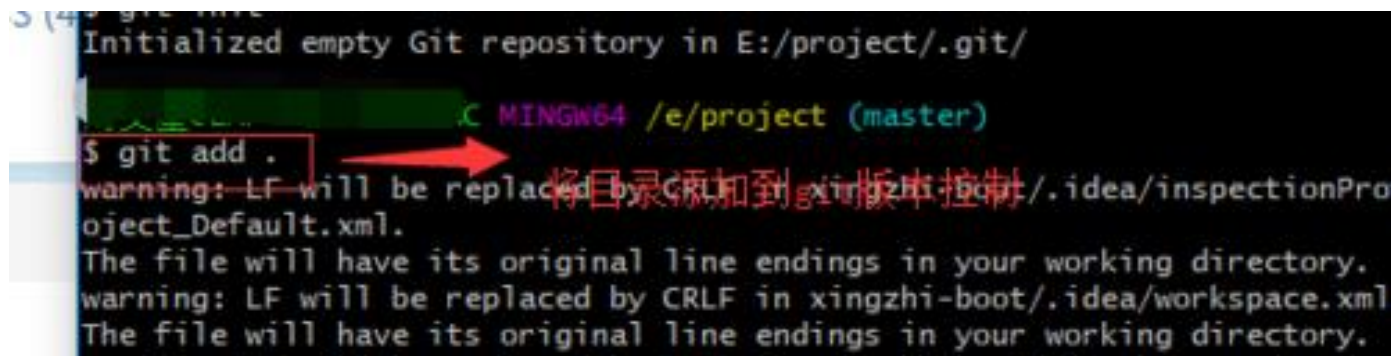
```
MINGW64:/e/project
$ git init
Initialized empty Git repository in E:/project/.git/
$
```

输入git init

3. 这时候发现项目所在目录下创建了一个.git 文件夹



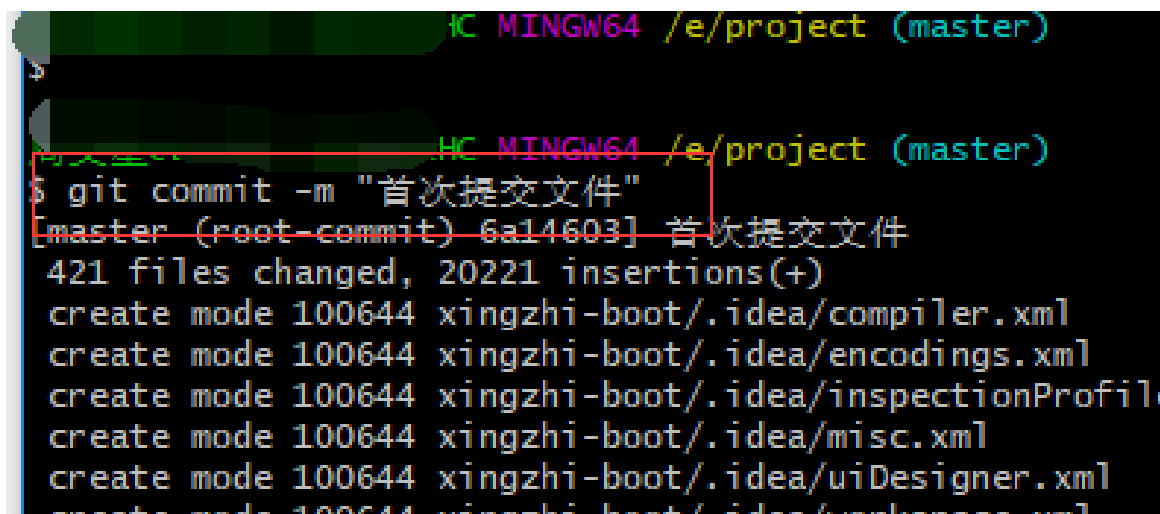
4. 接下来继续输入命令 `git add .` 表示将当前目录下所有文件纳入 git 版本控制, 如果当前目录有多个文件, 可以将.换成文件名指定文件



```
Initialized empty Git repository in E:/project/.git/
$ git add .
warning: LF will be replaced by CRLF in xingzhi-boot/.idea/inspectionPro
ject_Default.xml.
The file will have its original line endings in your working directory.
warning: LF will be replaced by CRLF in xingzhi-boot/.idea/workspace.xml
The file will have its original line endings in your working directory.
```

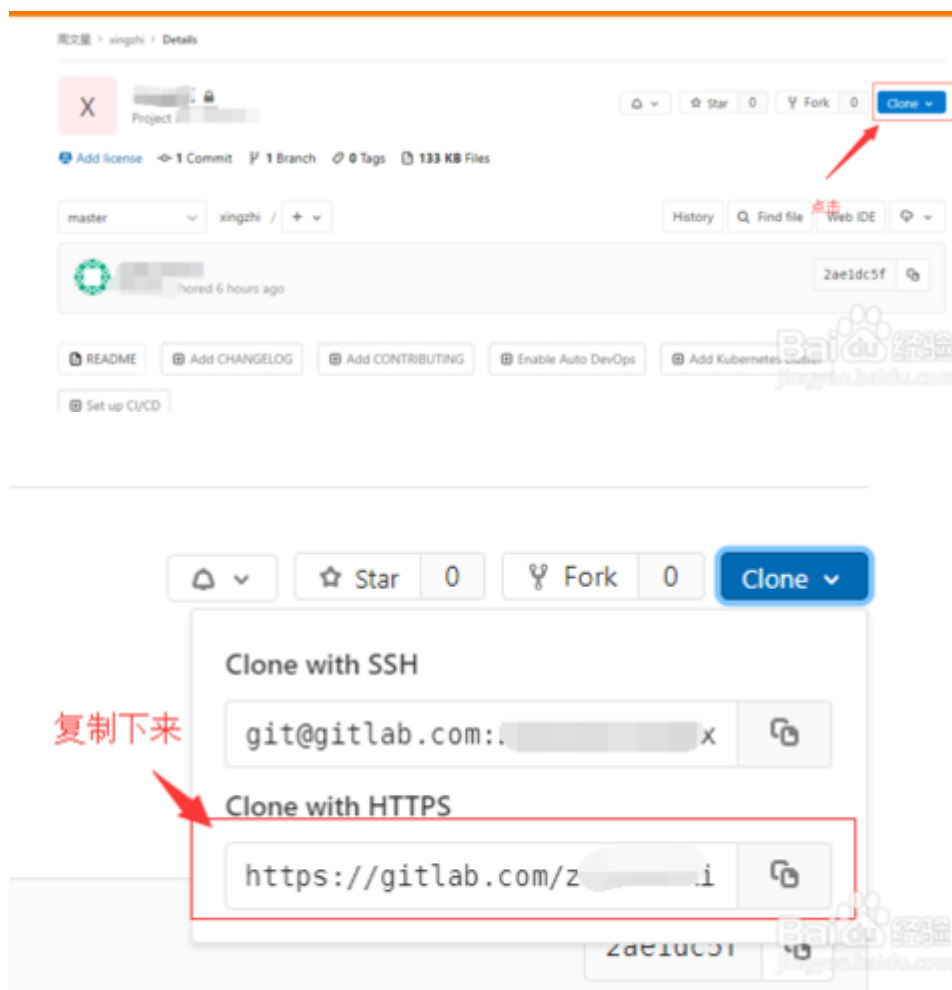
将目录添加到git版本控制

5. 接下来可以执行命令提交: `git commit -m "注释语句"`, 提交到本地仓库



```
MINGW64:/e/project (master)
$ git commit -m "首次提交文件"
[master (root-commit) 6a14603] 首次提交文件
421 files changed, 20221 insertions(+)
create mode 100644 xingzhi-boot/.idea/compiler.xml
create mode 100644 xingzhi-boot/.idea/encodings.xml
create mode 100644 xingzhi-boot/.idea/inspectionProfile
create mode 100644 xingzhi-boot/.idea/misc.xml
create mode 100644 xingzhi-boot/.idea/uiDesigner.xml
create mode 100644 xingzhi-boot/.idea/workspace.xml
```

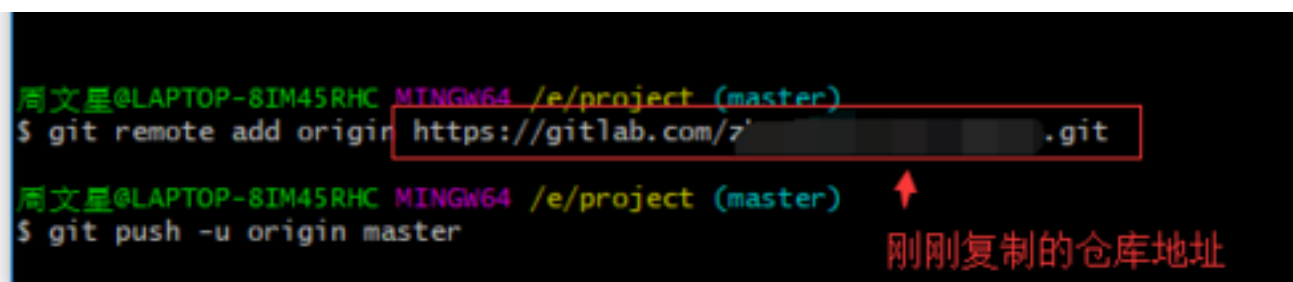
6. 登录自己的 gitlab 账号，创建好仓库，然后进入到如下页面，点击 clone，然后弹出的框中，复制下 http 路径

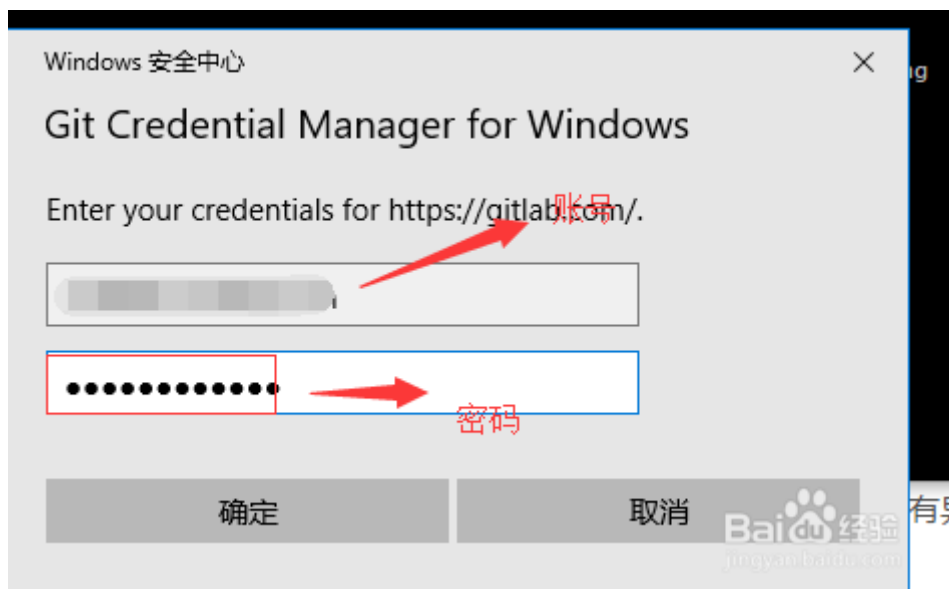


7. 然后先执行命令 `git pull https://gitlab.com/xxxx/xxx.git master --allow-unrelated-histories`，注意--后面表示允许不相干的两个仓库合并，因为我们之前两个仓库是完全独立的，所以直接是没办法 pull 下来，需要加上后面参数才行



8. 然后执行如下命令将本地仓库和远程仓库关联，再 push 上去，push 的时候会提示输入账号名和密码，输入即可成功





END

注意事项

- 如果是要上传到 github 上面，只需要把命令 gitlab 换成 github 即可

经验内容仅供参考，如果您需解决具体问题(尤其法律、医学等领域)，建议您详细咨询相关领域专业人士。

【Git】如何把别人仓库里面的代码更新到自己的远程仓库上来（无冲突的前提下）

很早以前关注了一个项目：

`git@github.com:halo-dev/halo.git`

Halo，一个不错的博客系统。

想学习一下其中的源码，当时就 fork 了一份到自己的账号下面来，然后一直就没有动过了。。。。

fork 到自己仓库里的 git 项目地址：

`git@github.com:VelonicaScofield/halo.git`

今天突然又想起来了，就想拿出来看看，但是已经过了很久了，Halo 作者已经更新了很多新特性了，难道还要守着老代码看？肯定是要最新的嘛。

这个就涉及到一个问题，怎么把别人的库上的代码更新到自己的远程库上去？我在公司使用 git 的时候也在考虑这个问题，但是公司电脑上装了乌龟，用起来太爽了，就没关注过用 git 命令怎么处理。

其实在公司的时候，我根据乌龟上打印的 git 命令猜测了一下，应该差不多，今天终于有机会实践一下了。

首先，在本地代码中，用 `git remote -v` 命令查看当前本地关联的远程分支有哪些，一般只会有两个 origin 远程分支，这里我已经加上了 remote 远程分支。

```
Administrator@PC-201811171556 MINGW64 /g/Code/github/halo (master)
$ git remote -v
origin  git@github.com:VelonicaScofield/halo.git (fetch)
origin  git@github.com:VelonicaScofield/halo.git (push)
remote  git@github.com:halo-dev/halo.git (fetch)
remote  git@github.com:halo-dev/halo.git (push)
```

然后使用：`git remote add <remoteName> <remoteAddress>` 把别人的远程分支加上来，如：

`git remote add remote git@github.com:halo-dev/halo.git`

使用 `git remote -v` 就是这样子了：

```
$ git remote -v
origin  git@github.com:VelonicaScofield/halo.git (fetch)
origin  git@github.com:VelonicaScofield/halo.git (push)
remote  git@github.com:halo-dev/halo.git (fetch)
remote  git@github.com:halo-dev/halo.git (push)
```

然后从这个 remote 远程分支上拉去 master 分支的代码，`git pull <remoteName> <branchName>`，如：
`git pull remote master`

```
Administrator@PC-201811171556 MINGW64 /g/Code/github/halo (master)
$ git pull remote master
remote: Enumerating objects: 1366, done.
remote: Counting objects: 100% (1366/1366), done.
remote: Compressing objects: 100% (174/174), done.
remote: Total 16635 (delta 1181), reused 1304 (delta 1157), pack-reused 15269
Receiving objects: 100% (16635/16635), 5.45 MiB | 6.00 KiB/s, done.
Resolving deltas: 100% (8831/8831), completed with 236 local objects.
From github.com:halo-dev/halo
 * branch            master      -> FETCH_HEAD
 * [new branch]      master      -> remote/master
Checking out files: 100% (1249/1249), done.
Updating 1d23670b..117883fc
```

最后把这个分支的代码推送到你的远程仓库上去，`git push <remoteName> <branchName>`：
`git push origin master`

```
Administrator@PC-201811171556 MINGW64 /g/Code/github/halo (master)
$ git push origin master
Enumerating objects: 16959, done.
Counting objects: 100% (16924/16924), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4951/4951), done.
Writing objects: 100% (16635/16635), 5.48 MiB | 99.00 KiB/s, done.
Total 16635 (delta 8783), reused 16412 (delta 8617)
remote: Resolving deltas: 100% (8783/8783), completed with 138 local objects.
To github.com:VelonicaScofield/halo.git
 1d23670b..117883fc master -> master
```

这样就 OK 了，去自己的仓库上面看看，已经和 Halo 作者的仓库保持一致了：

[Manage topics](#)

🔖 1,388 commits 🌿 2 branches 🕒 14 releases 👤 11 contributors 📄 GPL-3.0

Branch: master ▾ New pull request Create new file Upload files Find File Clone or download ▾

This branch is even with halo-dev:master. Pull request Compare

ruibaby and JohnNiang release 1.0.3 (halo-dev#235) Latest commit 117883f 1 hour ago

| | | |
|-------------------|---|--------------|
| 📁 .github | Merge pull request halo-dev#172 from halo-dev/dev | last month |
| 📁 gradle/wrapper | Add git and github test | 2 months ago |
| 📁 scripts | Fix docker build release bug | last month |
| 📁 src | release 1.0.3 | 1 hour ago |
| 📄 .gitattributes | Refactor .gitignore | 3 months ago |
| 📄 .gitignore | Update admin and comment. | 2 months ago |
| 📄 .travis.yml | Fix issue(halo-dev#153) | 2 months ago |
| 📄 Dockerfile | Change openjdk:8-jdk-alpine to openjdk:8-jre-alpine in Dockerfile | 2 months ago |
| 📄 LICENSE | Create LICENSE | last year |
| 📄 README.md | release 1.0.3 | 1 hour ago |
| 📄 build.gradle | release 1.0.3 | 1 hour ago |
| 📄 gradlew | Change build tool from maven to gradle | 3 months ago |
| 📄 gradlew.bat | Change build tool from maven to gradle | 3 months ago |
| 📄 lombok.config | Change build tool from maven to gradle | 3 months ago |
| 📄 settings.gradle | Change settings.gradle to settings.gradle | 22 days ago |

README.md

其实这里只是解决了最简单的情况，如果你修改了代码，你的仓库中的代码可能就和别人的有冲突了，这时候如何解决呢？可能要用 **merge** 命令解决吧，后面我再试试。

又或者如果别人新增了一个分支，上面的操作都是把别人远程仓库上的一个分支的代码往自己的分支上更新，如果别人新增了一个分支，怎么把这个新的分支更新过来呢？可能涉及创建分支的动者，后面我也试试看。

时间总是悄悄流逝