【教程贴】宝塔面板实践jenkins自动化构建Docker镜像(Python项目)

此使用帮助适用宝塔面板9.2.0以上的版本（2024年9月5日后发布）

如何在Jenkins中完成自动化Python项目部署/测试?
下面我们通过宝塔面板的Docker应用服务部署Jenkins，简单高效的实践

应用商店已经支持100+应用，**欢迎更多应用服务商和宝塔面板用户联系我们持续更新，最下方有我们联系方式**

【真实使用场景】

张三是一个公司的技术人员，公司的Python项目部署模式是运行在Docker中的，但是有一个问题一直困扰着他，每次修改代码都需要手动停止并删除容器，然后重新执行构建Docker镜像的命令，再重新运行新的容器，这个重复的事情很影响他的工作效率，现在有一个需求，如何高效并自动化的完成此工作?

李四提出方案：使用Jenkins来添加一个自动化任务，监听git变化，自动拉取最新代码并完成容器构建/部署的所有工作，只需要创建1次任务，即可减少100%的重复工作量

张三听后同意了这个方案，李四开始实施此方案

【场景实现】

准备环境

```
部署Python项目的服务器（同时也是部署Jenkins的服务器，本教程必须安装宝塔面板和Docker）
IP：192.168.1.104
系统：麒麟v10

git使用码云（gitee.com）
项目目录结构：
/www/wwwroot/flask_02/
            |
            ----main.py
            |
            ----Dockerfile

```

```
Python项目代码（main.py）：
from flask import Flask

app = Flask(__name__)

@app.route('/')
def hello_world():
    return "Hello, World!"

@app.route('/new_route')
def new_route():
    return "It's new_route!"

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)

Dockerfile内容:
FROM python:3.7.9-alpine

RUN pip install --no-cache-dir -i http://mirrors.aliyun.com/pypi/simple/ --trusted-host mirrors.aliyun.com Flask

ADD . /app

CMD ["python", "/app/main.py"]
```

【实现目标】

1.使用宝塔面板的Docker应用商店成功安装Jenkins应用

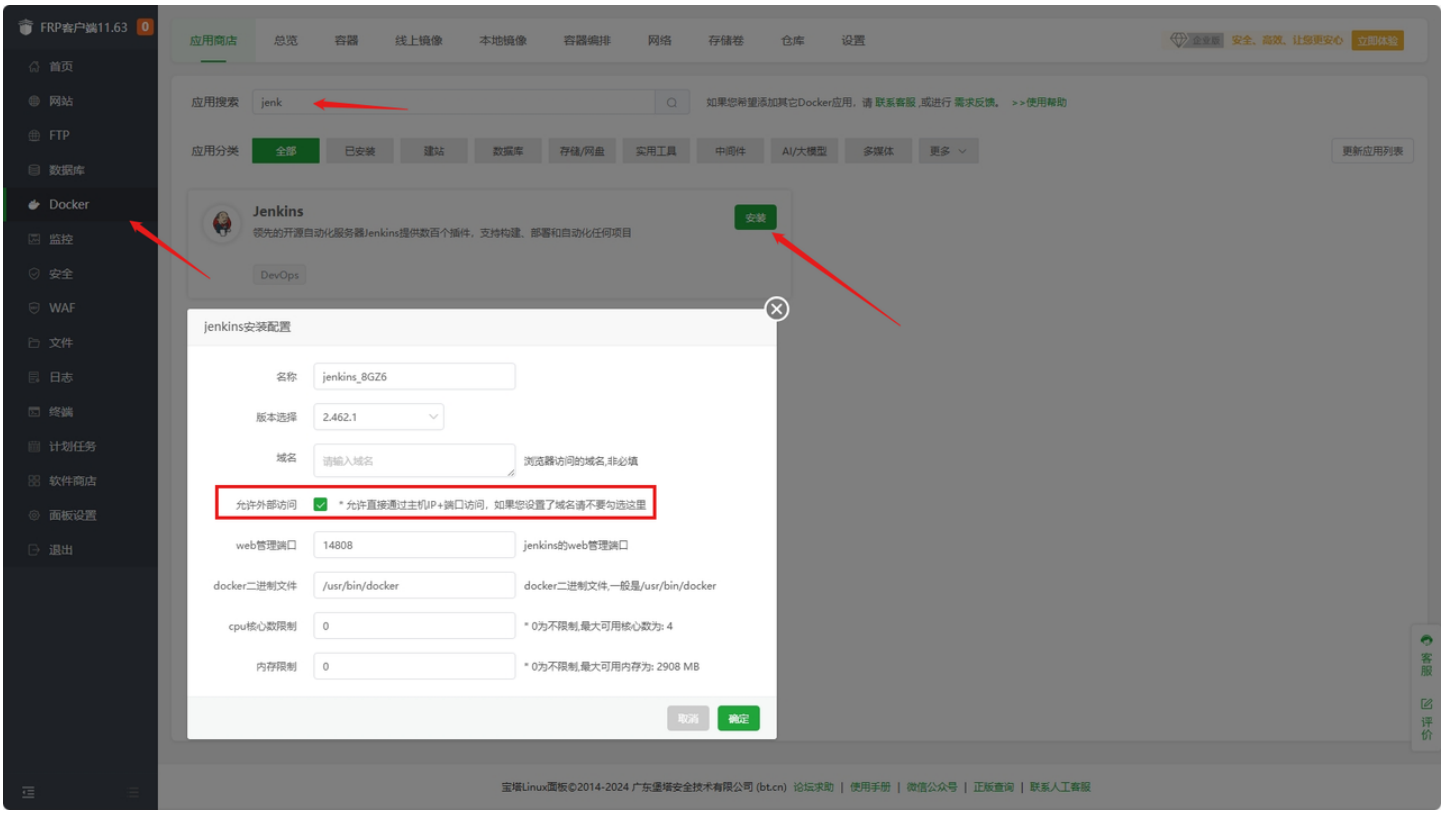
2.使用码云（gitee.com）创建flask_02项目，并在 192.168.1.104 这台服务器初始化它

3.使用Jenkins创建自动化部署的任务，并成功监听提交的git commit完成自动化部署

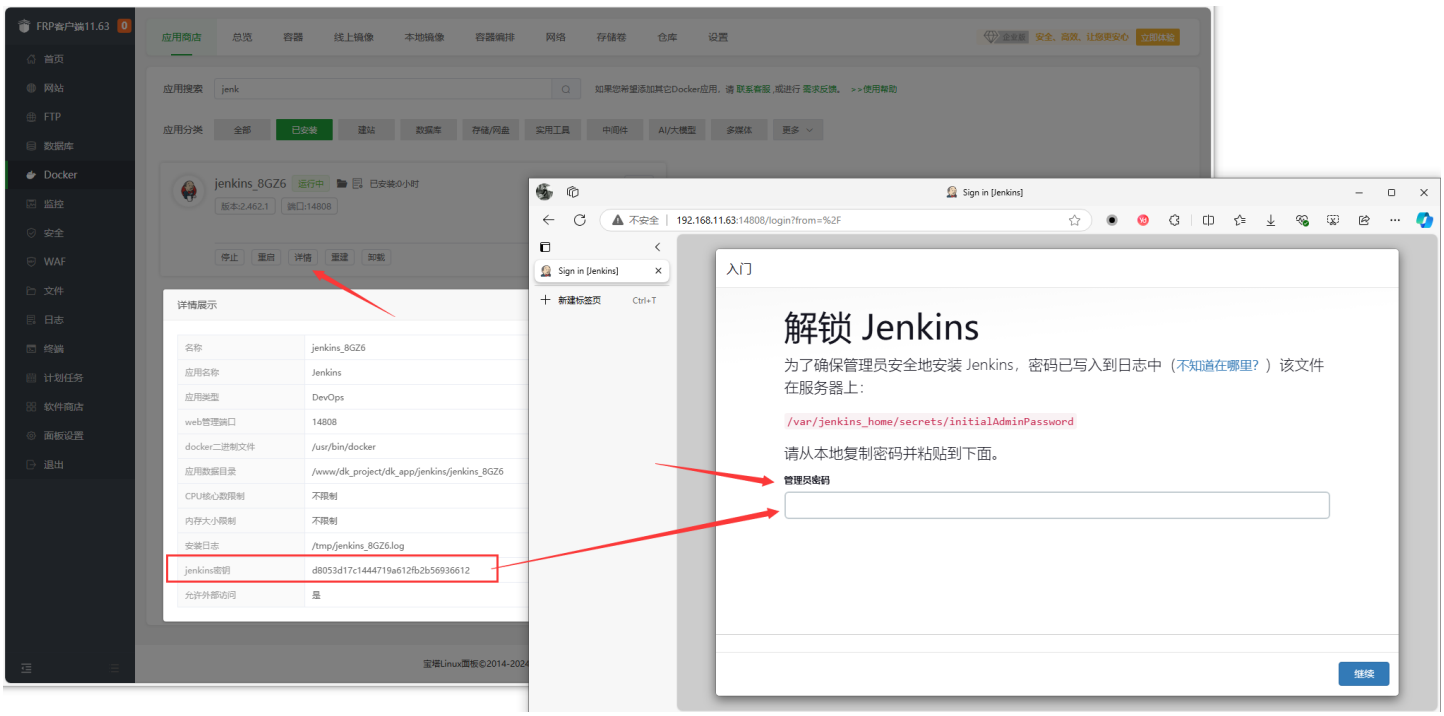
【安装Jenkins】

在 192.168.11.63 这台服务器安装宝塔面板（9.2.0以上的正式版），并安装Docker服务，安装完成后点击 应用商店 并且搜索关键词 【Jenkins】

得到搜索结果后，选择【Jenkins】，点击安装，并且确保【允许外部访问】已经勾选（需要设置域名可填写域名然后不勾选）

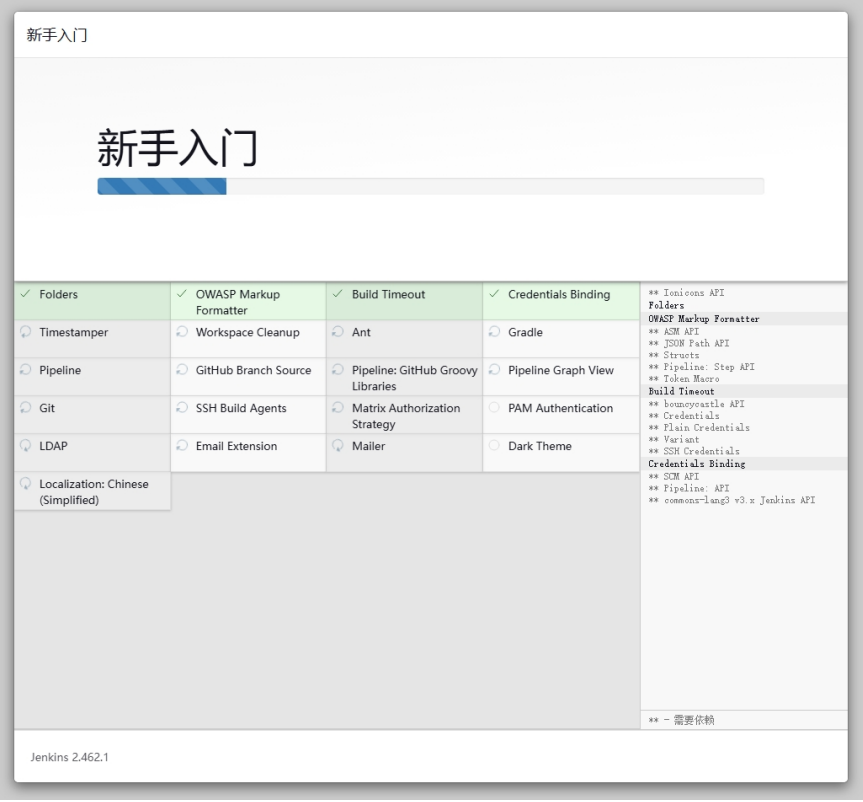


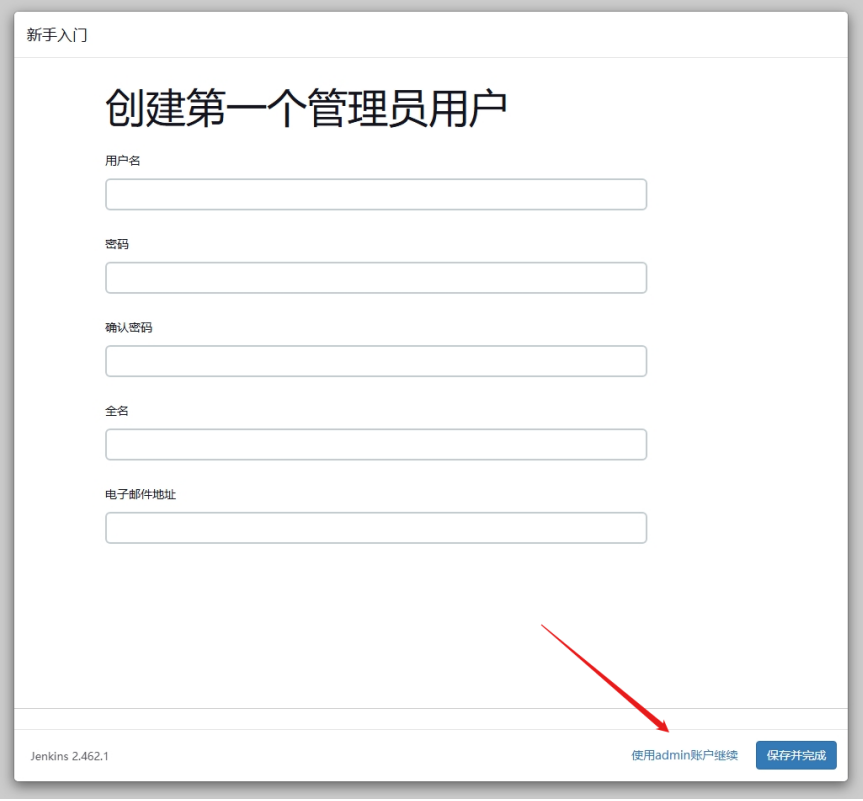
服务安装完成后，浏览器访问 http://192.168.11.63:14808，并点击详情查看初始的管理员密钥（此密钥获取时机是在Jenkins在Docker中初始化完成之后，请耐心等待30秒以上）

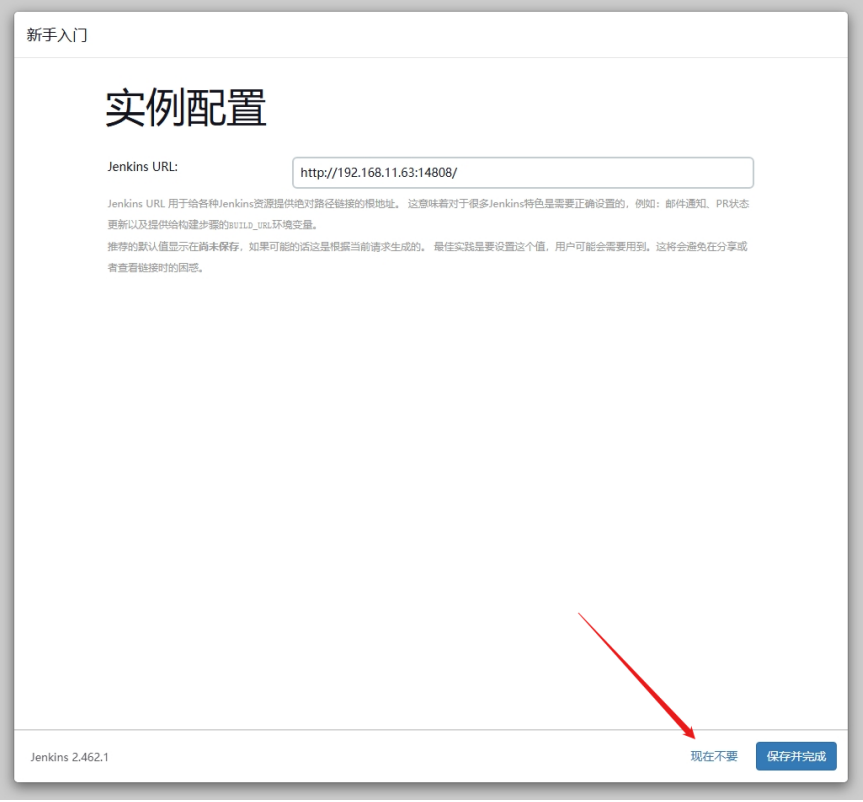


接下来快速进行初始化，教程全部保持默认，请根据下面几张图操作（如果你想自定义则安装自己需求设置即可）


新手入门
自定义Jenkins
插件通过附加特性来扩展Jenkins以满足不同的需求。
安装推荐的插件
安装Jenkins社区推荐的插件。
选择插件来安装
选择并安装最适合的插件。
Jenkins 2.462.1


新手入门
新手入门
Folders
OWASP Markup Formatter
Build Timeout
Credentials Binding
Timestamper
Workspace Cleanup
Ant
Gradle
Pipeline
GitHub Branch Source
Pipeline: GitHub Groovy Libraries
Pipeline Graph View
Git
SSH Build Agents
Matrix Authorization Strategy
PAM Authentication
LDAP
Email Extension
Mailer
Dark Theme
Localization: Chinese (Simplified)
** Ionicons API
Folders
OWASP Markup Formatter
** ASM API
** JSON Path API
** Structs
** Pipeline: Step API
** Token Macro
Build Timeout
** bouncycastle API
** Credentials
** Plain Credentials
** Variant
** SSH Credentials
Credentials Binding
** SCM API
** Pipeline: API
** commons-lang3 v3.x Jenkins API
** - 需要依赖
Jenkins 2.462.1

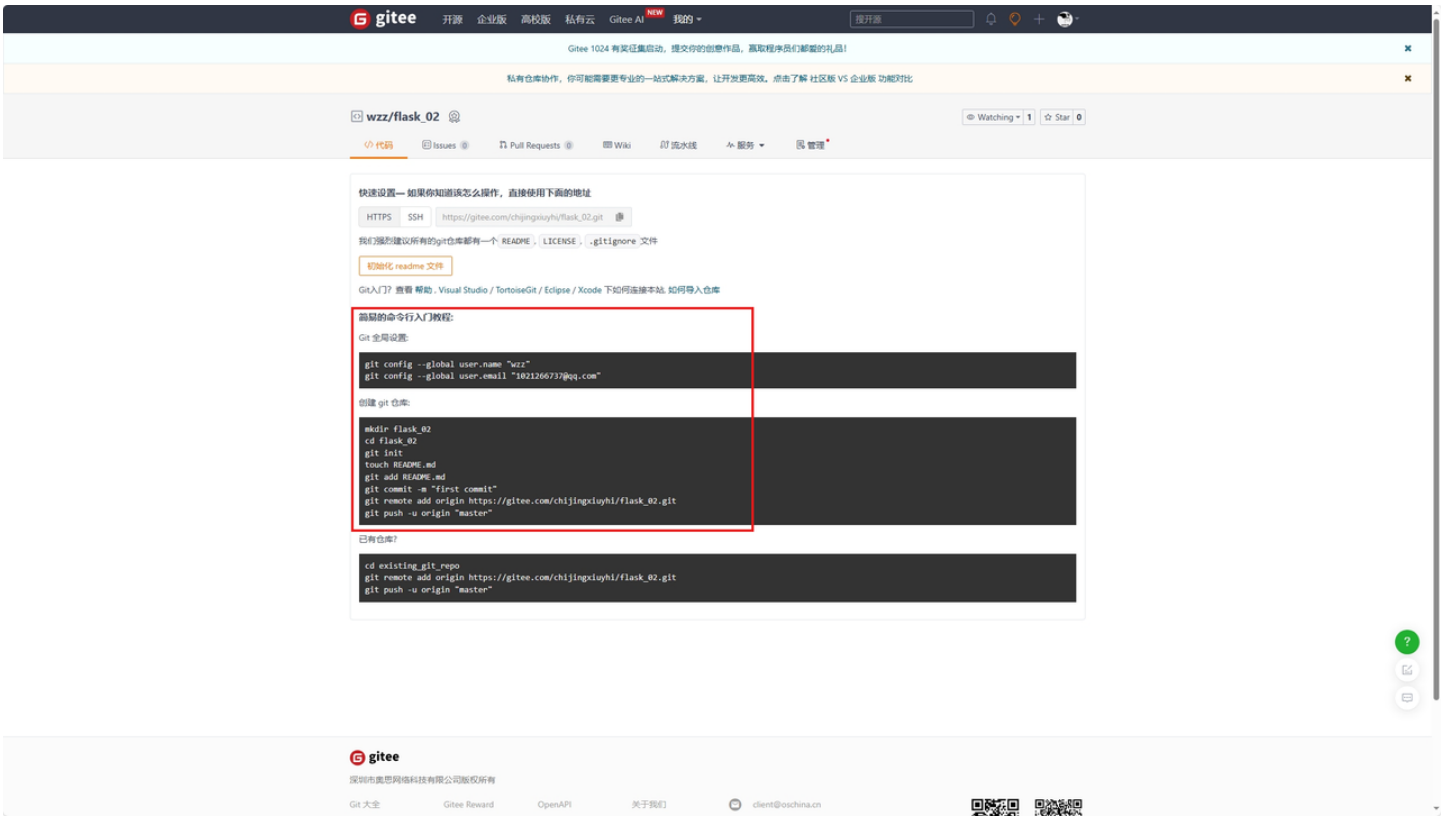

新手入门
创建第一个管理员用户
用户名
密码
确认密码
全名
电子邮件地址
Jenkins 2.462.1
使用admin账户继续
保存并完成


新手入门
实例配置
Jenkins URL:
http://192.168.11.63:14808/
Jenkins URL 用于给各种Jenkins资源提供绝对路径链接的根地址。 这意味着对于很多Jenkins特色是需要正确设置的，例如：邮件通知、PR状态更新以及提供给构建步骤的BUILD_URL环境变量。
推荐的默认值显示在蓝未保存，如果可能的话这是根据当前请求生成的。 最佳实践是要设置这个值，用户可能会需要用到。这将会避免在分享或者查看链接时的困惑。
Jenkins 2.462.1
现在不要
保存并完成

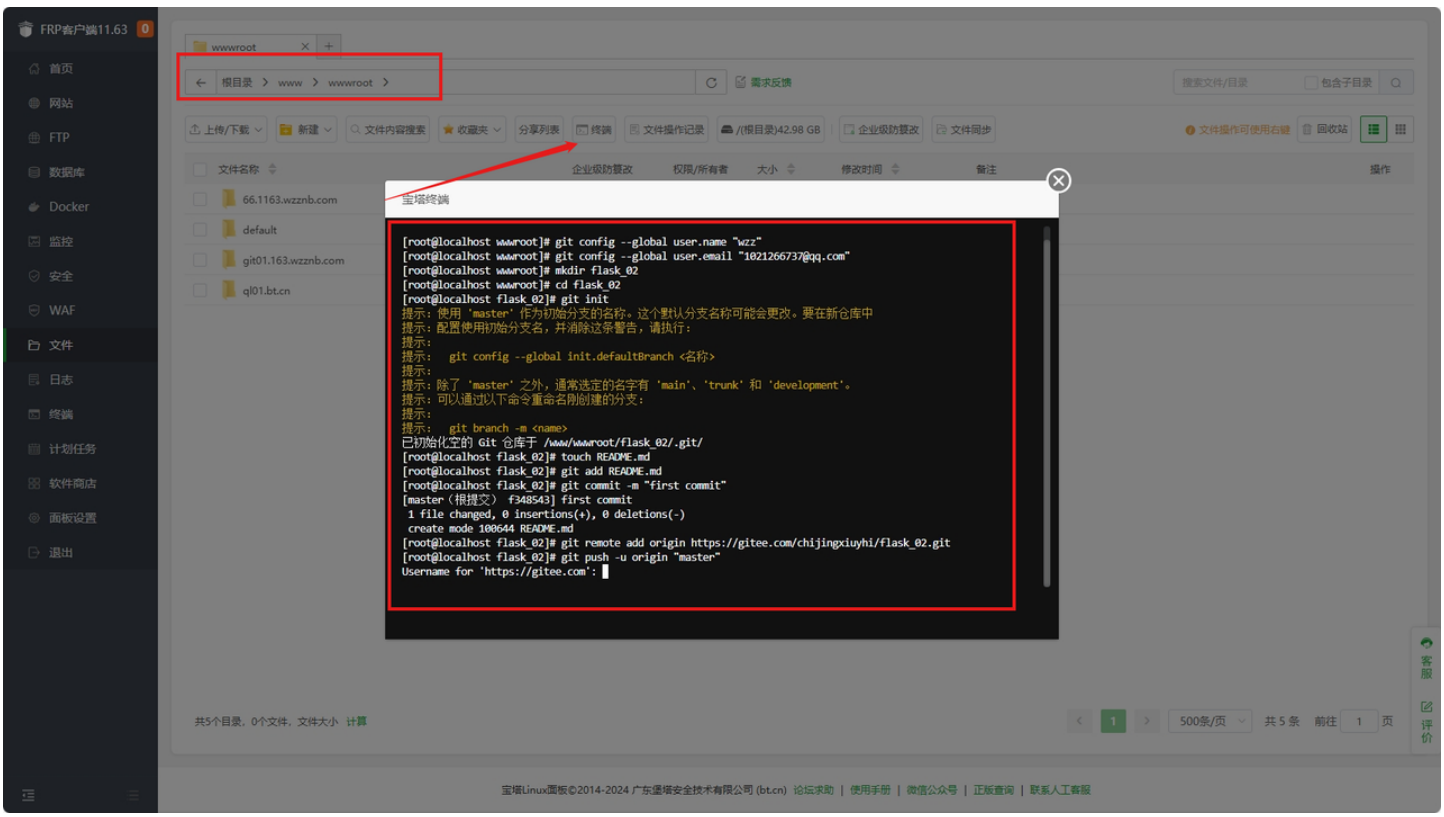
【创建并初始化git仓库】

浏览器访问https://gitee.com/projects/new，并在名称中输入flask_02，其他保持默认，点击创建

创建成功后会提示初始化的页面，需要在 192.168.11.63 服务器中执行红色框的命令

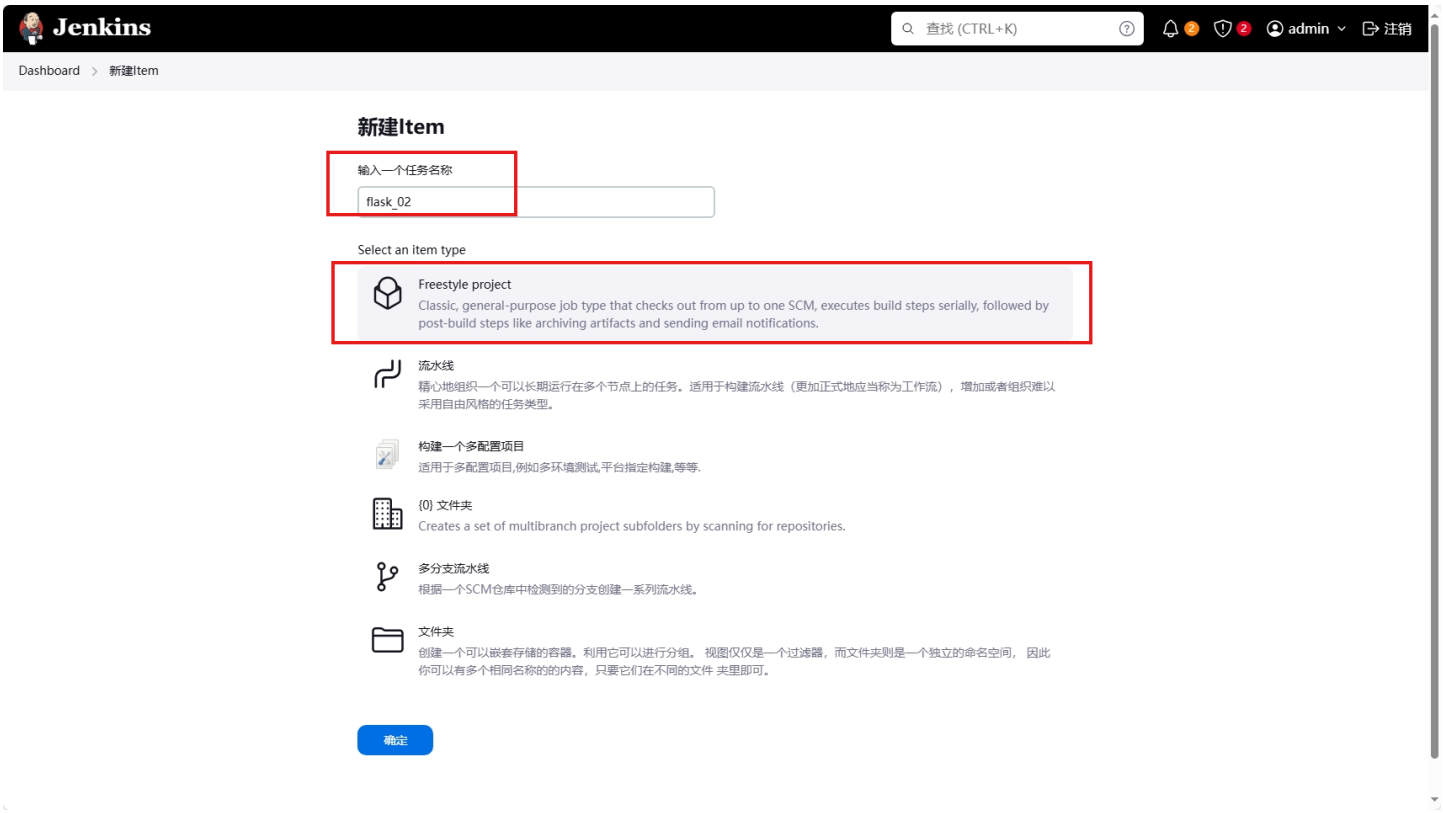


打开 192.168.11.63 的宝塔面板，并前往【文件】页面，进入 /www/wwwroot 目录，并点击上方终端按钮，开始执行命令进行 git仓库 初始化（最后一步会提示输入你的gitee.com账号和密码，根据提示输入即可，如果你选了SSH的方式则需要设置密钥，具体方式请参考gitee官方教程：https://help.gitee.com/base/account/SSH%E5%85%AC%E9%92%A5%E8%AE%BE%E7%BD%AE）

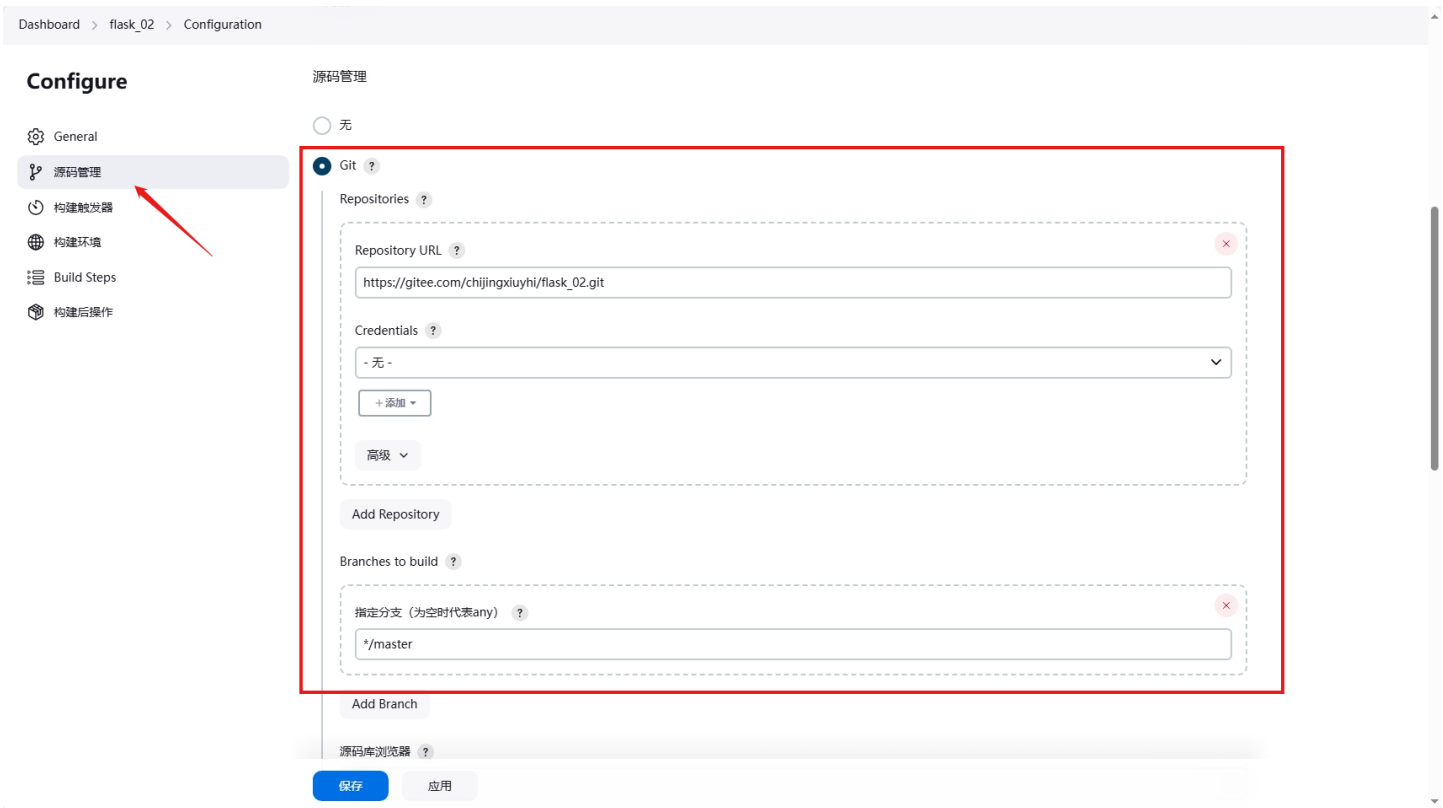


【创建Jenkins自动化任务】

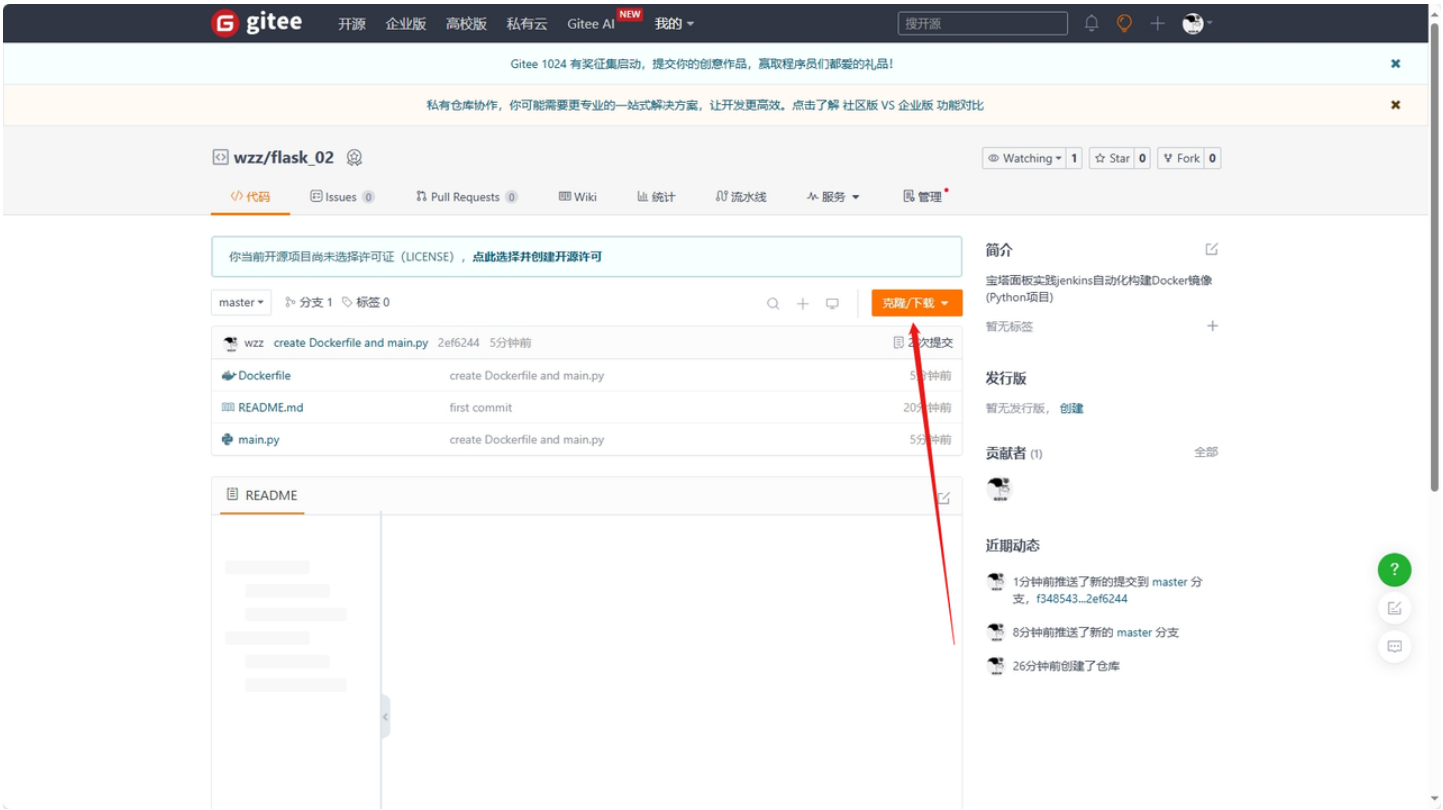
点击Jenkins左侧【新建Item】选项，输入一个任务名称：flask_02，选择 【Freestyle project】，点击 确定 按钮

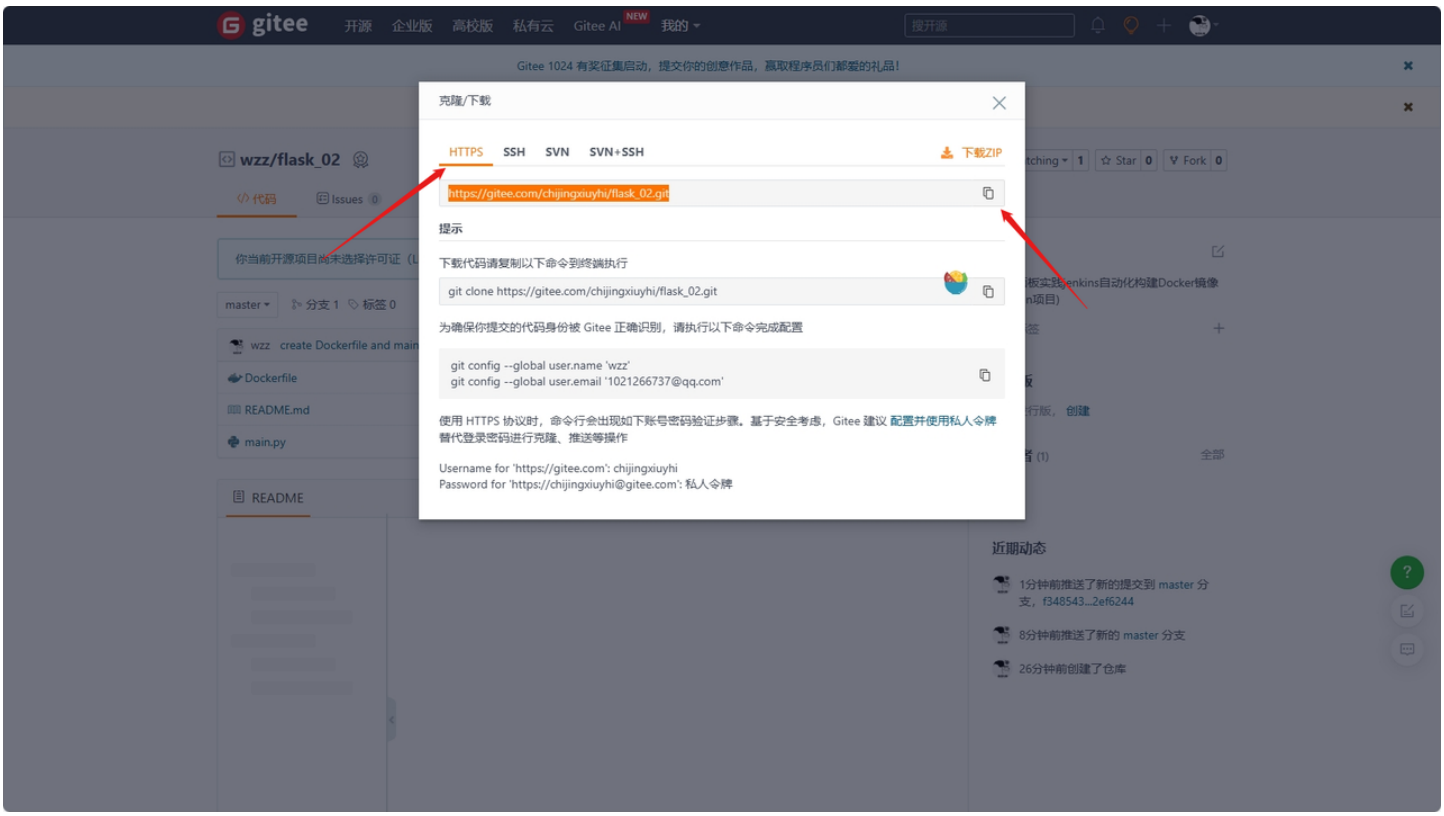


接下来会提示配置此任务，在【源码管理】这一项中，选择Git的方式，随后填写git的地址，填入前面git仓库的地址即可



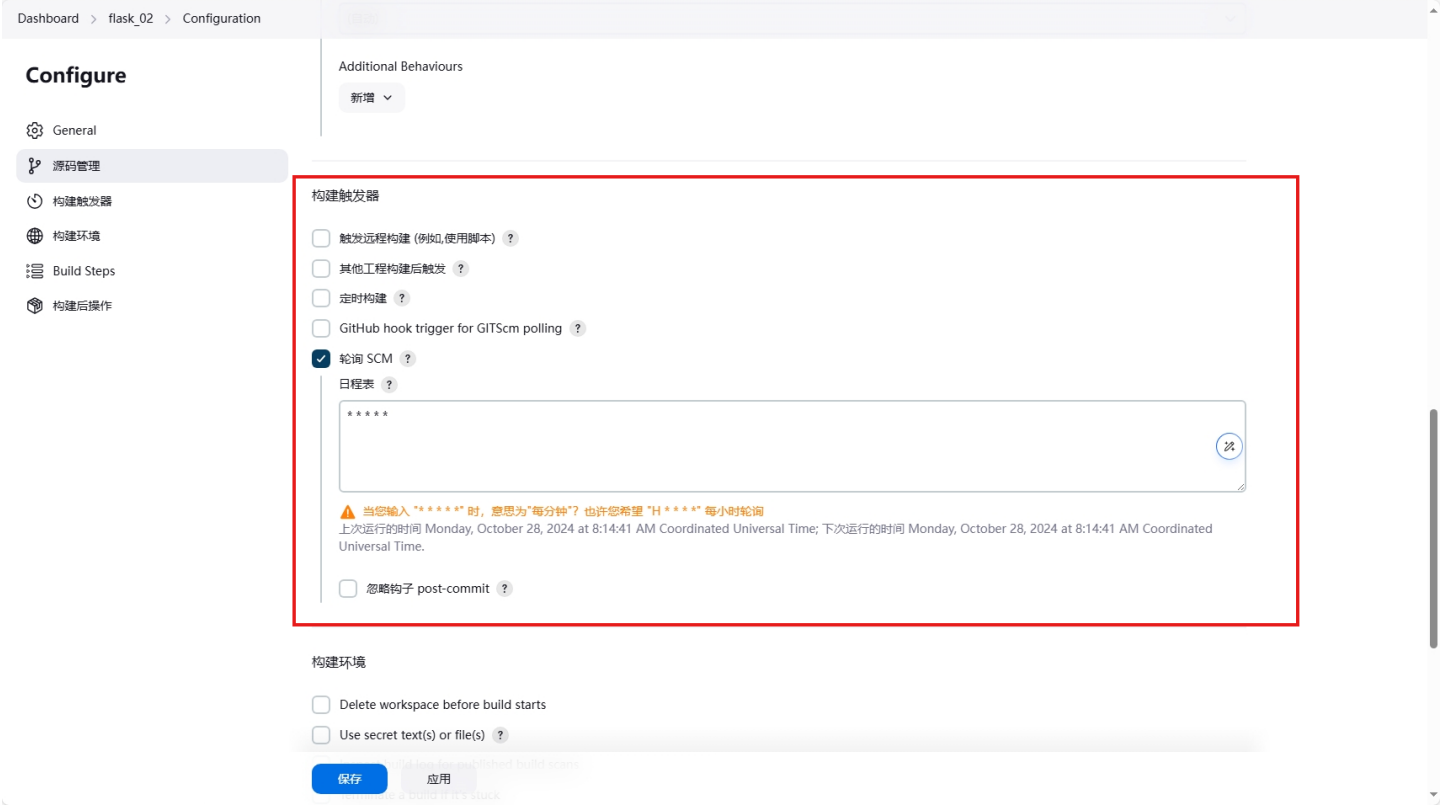
点击【克隆/下载】，即可复制git地址





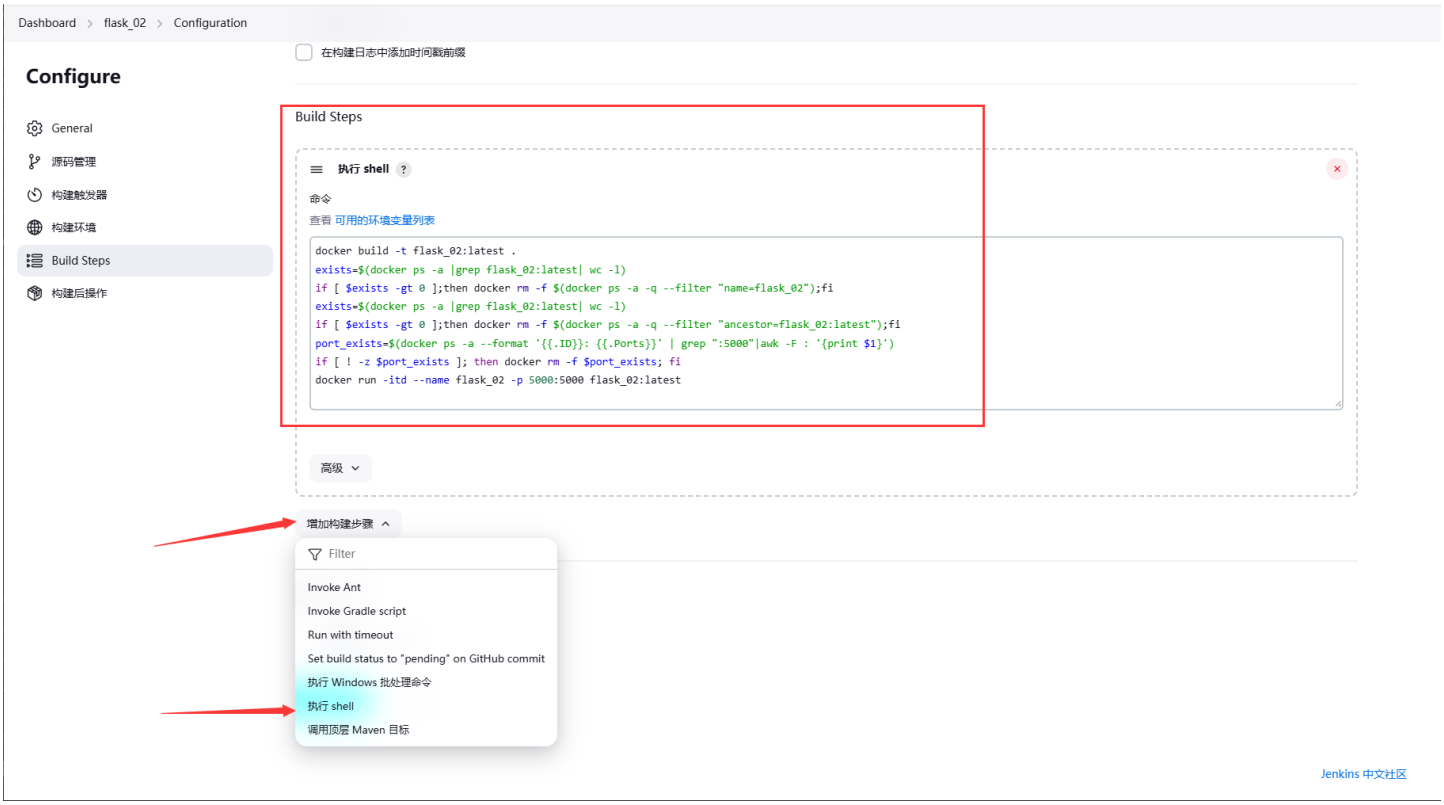
接着下滑配置窗口，【构建触发器】中，选择【轮询SCM】项，并将下面的*填入日程表中（代表每分钟轮询一次git提交记录，如果有新的就会执行此任务）

```
1 * * * * *
```



继续下滑配置窗口，在【Build Steps】中，选择【增加构建步骤】-->【执行shell】，填入下面的shell代码

```
1 docker build -t flask_02:latest .
2 exists=$(docker ps -a |grep flask_02:latest| wc -l)
3 if [ $exists -gt 0 ];then docker rm -f $(docker ps -a -q --filter
  "name=flask_02");fi
4 exists=$(docker ps -a |grep flask_02:latest| wc -l)
5 if [ $exists -gt 0 ];then docker rm -f $(docker ps -a -q --filter
  "ancestor=flask_02:latest");fi
6 port_exists=$(docker ps -a --format '{{.ID}}: {{.Ports}}' | grep ":5000"|awk -
  F : '{print $1}')
7 if [ ! -z $port_exists ]; then docker rm -f $port_exists; fi
8 docker run -itd --name flask_02 -p 5000:5000 flask_02:latest
```



最后点击保存，自动化任务就创建完成了

【提交git测试自动化部署】

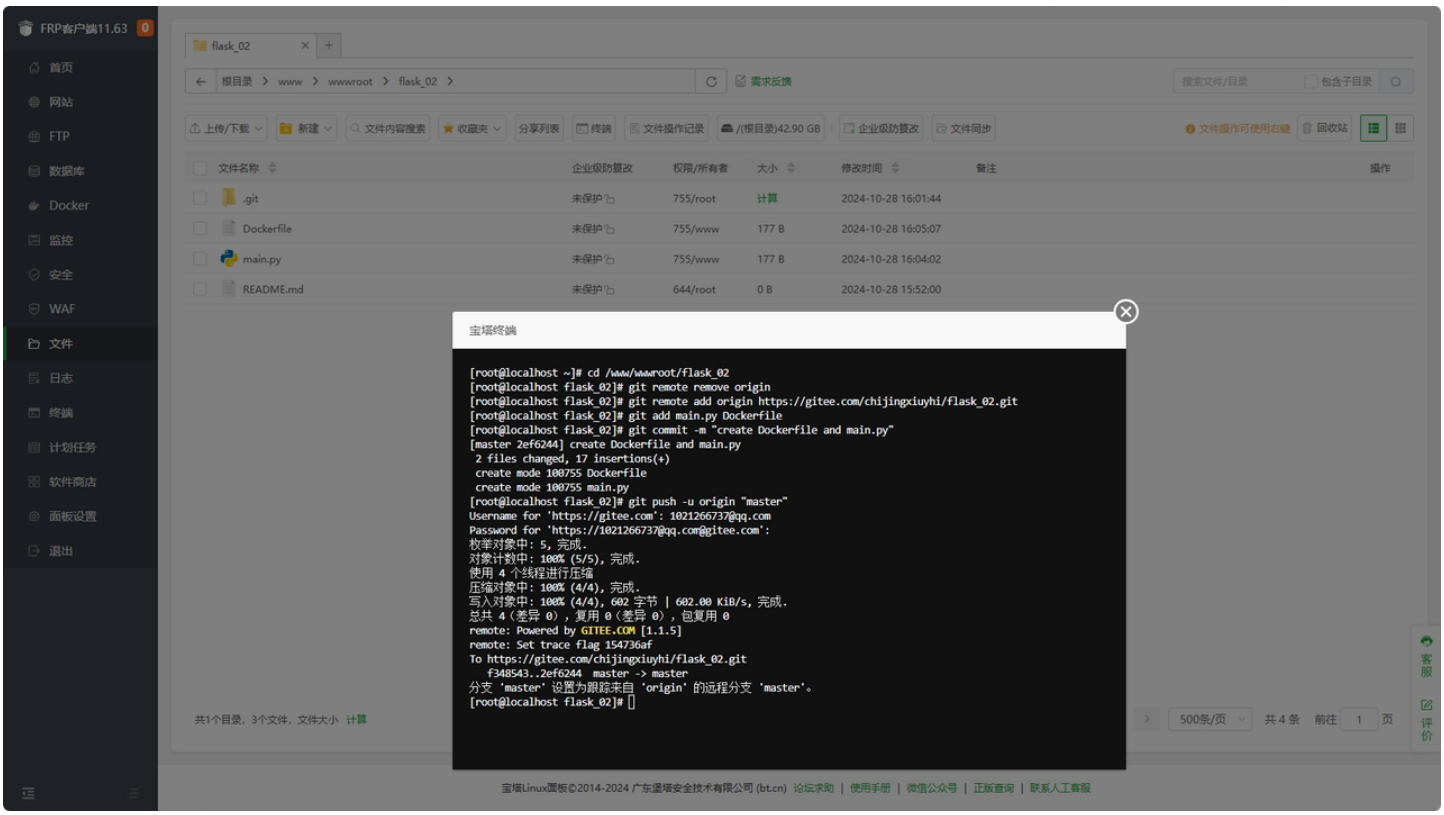
在 /www/wwwroot/flask_02 目录下面新建main.py和Dockerfile两个文件，并填入对应的代码

```python
# main.py文件的内容
from flask import Flask

app = Flask(__name__)

@app.route('/')
def hello_world():
    return "Hello, World!"

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)
```

```
# Dockerfile文件的内容
FROM python:3.7.9-alpine

RUN pip install --no-cache-dir -i http://mirrors.aliyun.com/pypi/simple/ --trusted-host mirrors.aliyun.com Flask

ADD . /app
```

```
7
8 CMD ["python", "/app/main.py"]
```

提交代码更改到码云中



git remote remove origin

git remote add origin https://gitee.com/befunny/flask_02.git

git add main.py Dockerfile

git commit -m "creat main.py and Dockerfile"

git push -u origin "master"

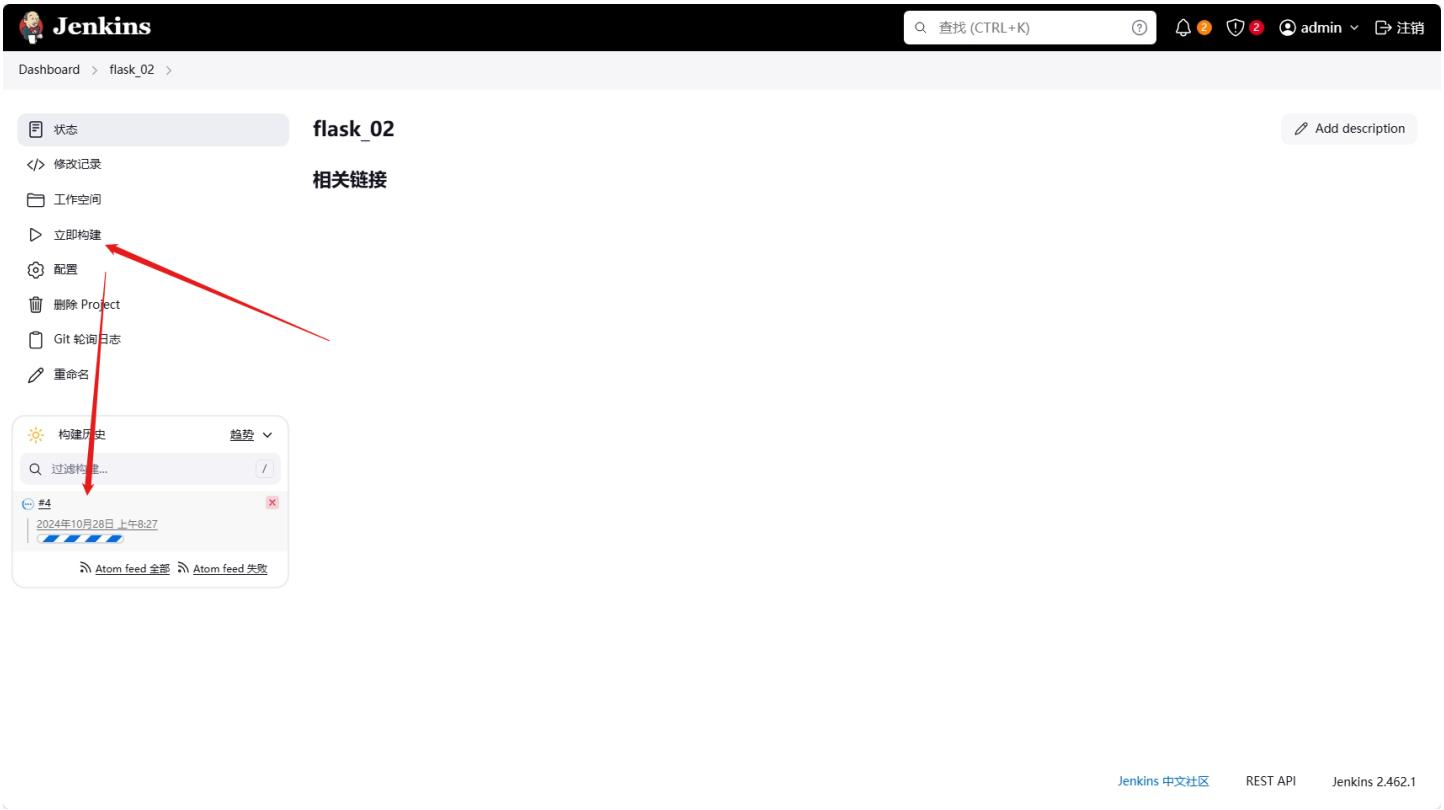
---

git add main.py

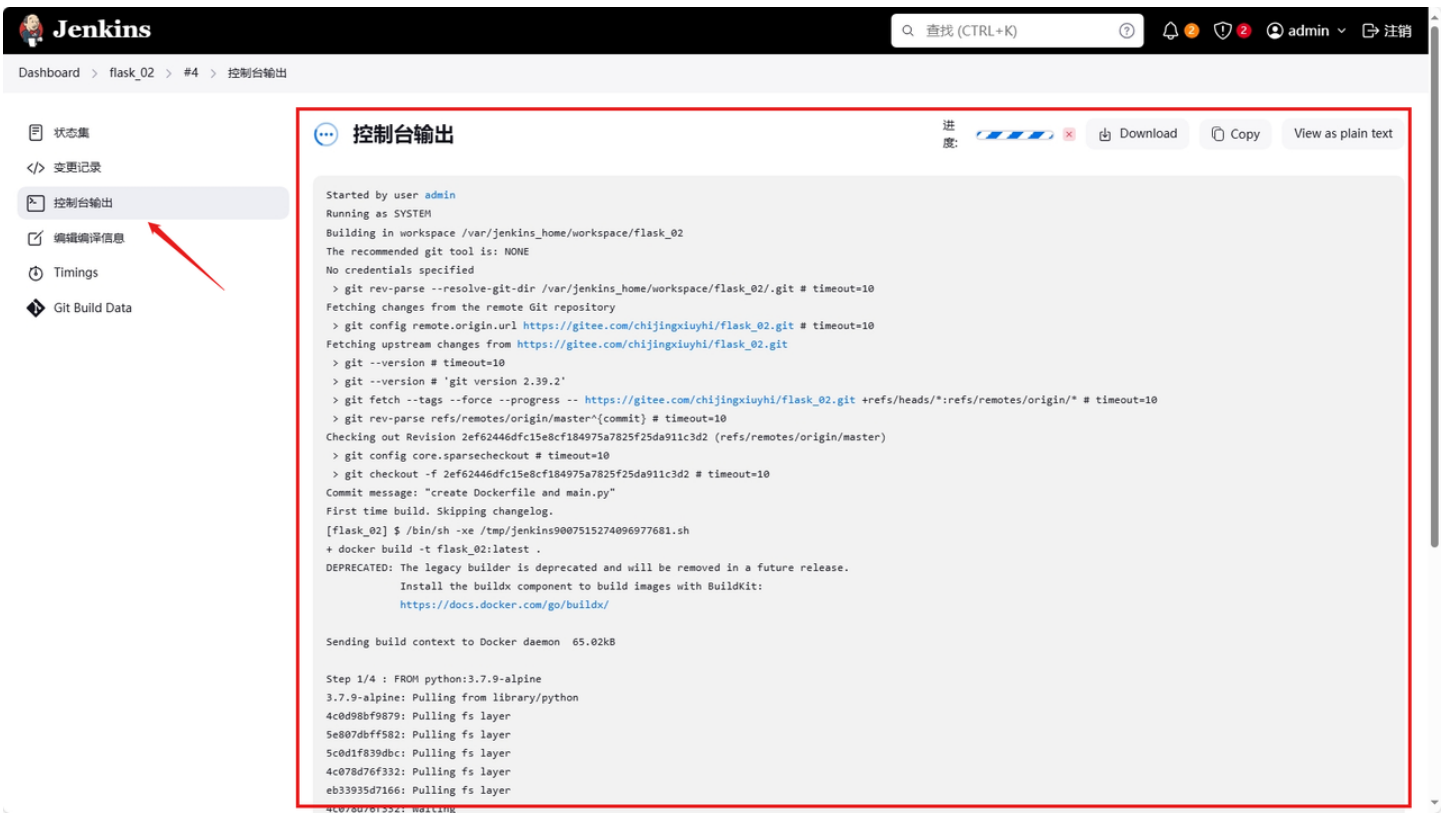
git commit -m "add new route"

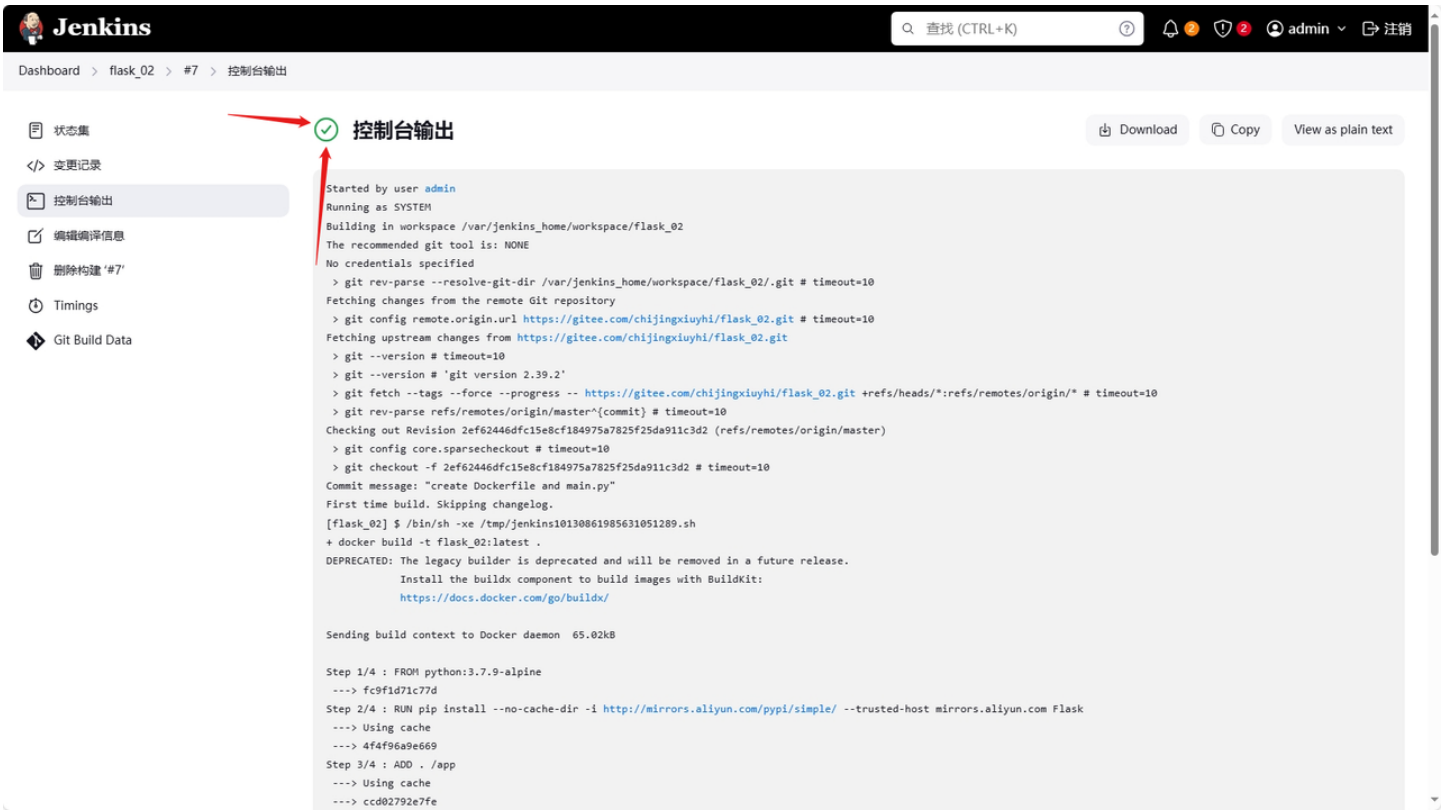
git push -u origin "master"

随后观察Jenkins的【flask_02】任务，是否出现构建变化，如果等不及可以直接点击【立即构建】（建议等待）

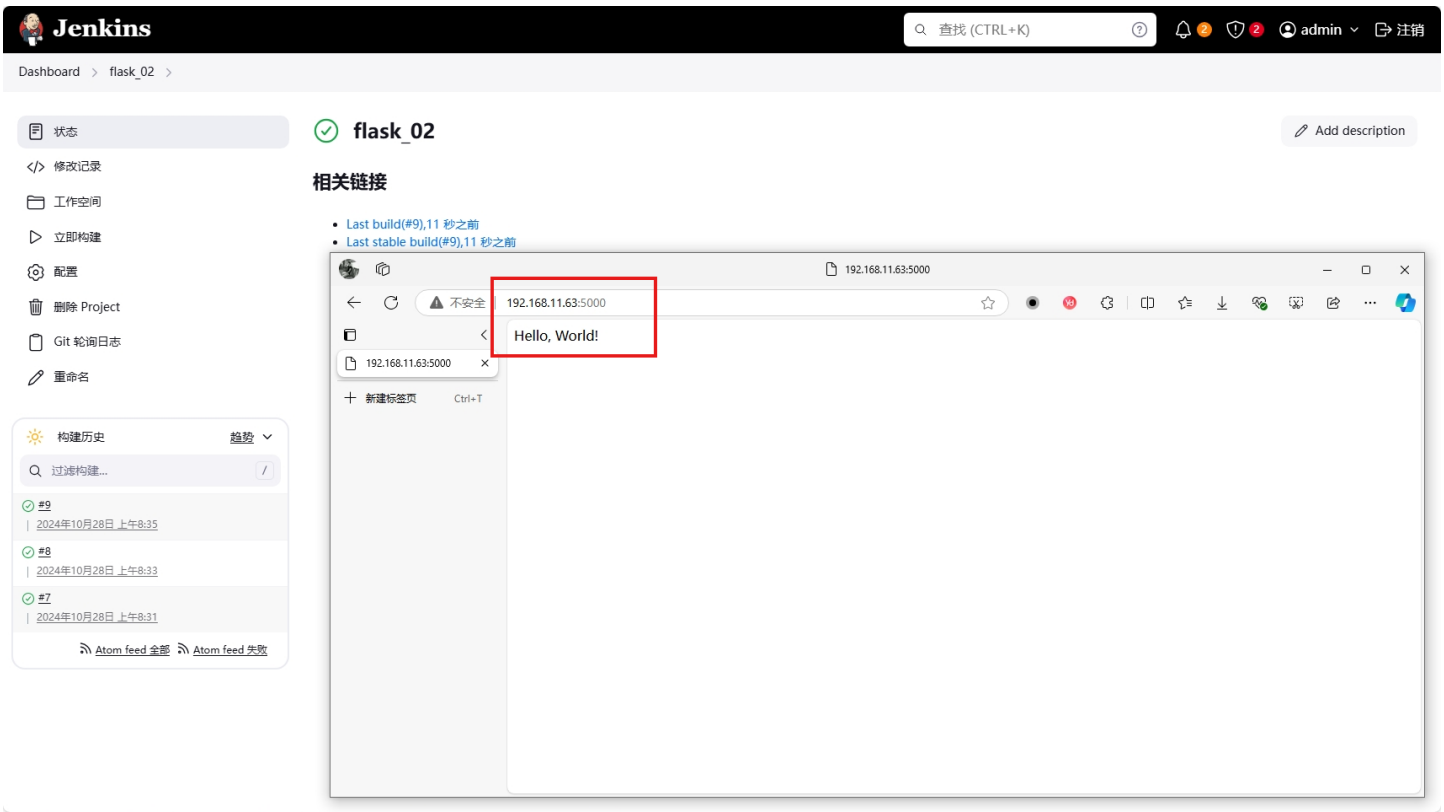


在构建过程中，可以查看对应任务的构建控制台输出





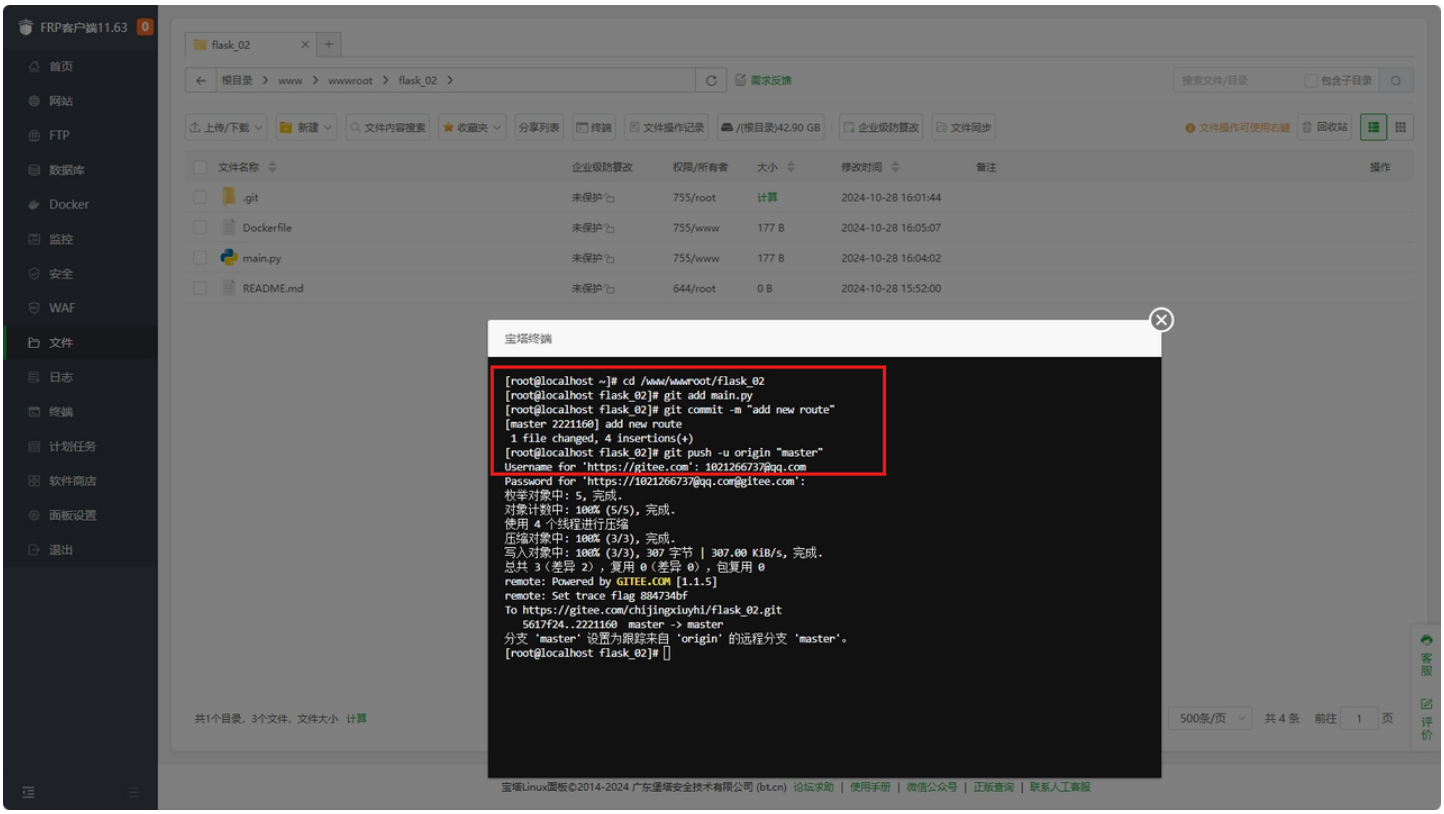
构建完成后，访问我们代码中的5000端口进行测试



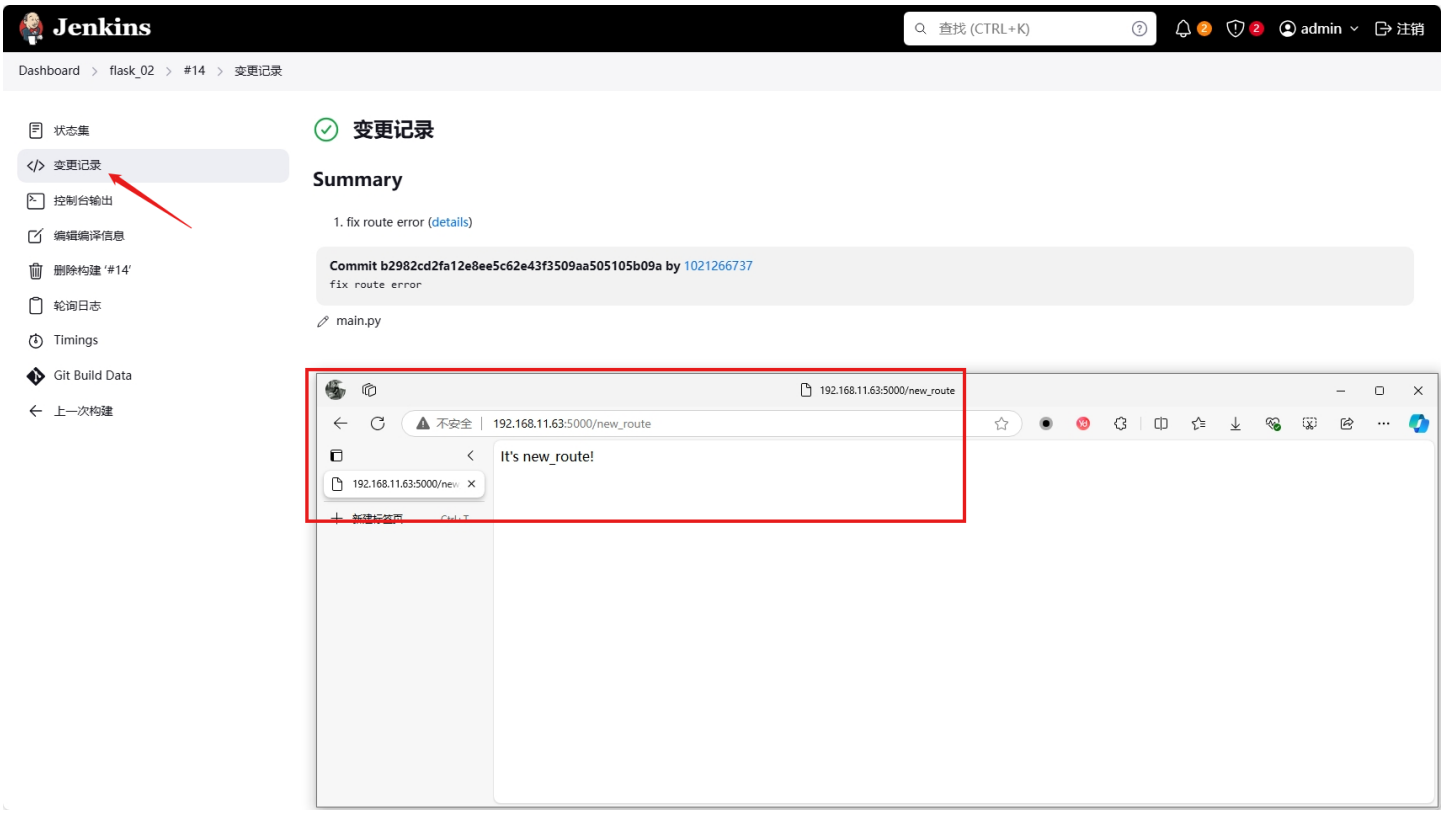
部署成功，接下来新增一个【/new_route】路由，再次提交git试试

```
1  # main.py文件内容
```

```python
from flask import Flask

app = Flask(__name__)

@app.route('/')
def hello_world():
    return "Hello, World!"

# 新增/new_route路由
@app.route('/new_route')
def new_route():
    return "It's new_route!"

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)

```



再次观察Jenkins中的构建动态，会自动新增一个构建记录，构建完成后，访问新的路由



方案完成！

注意：应用内其他端口不经过系统防火墙，勾选【允许外部访问】后是直接允许外部访问，如果您是云服务器，勾选【允许外部访问】后，请到服务器安全组放行此数据库的端口

开放安全组示例(必需)：

阿里云

腾讯云

使用遇到问题欢迎帖子留言，或者加入Docker官方交流QQ群：662047798