**Docker学习笔记**

目录

[Docker安装 3](#_Toc497128312)

[在Docker中部署运行SpringBoot项目 4](#_Toc497128313)

[Docker File解析 6](#_Toc497128314)

[使用Docker的数据卷，来实现读取项目日志 7](#_Toc497128315)

[Docker上实现jenkins自动部署 10](#_Toc497128316)

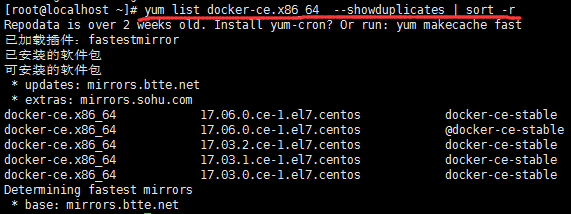
[Docker-compose插件 24](#_Toc497128317)

## Docker安装

Docker安装方法：

列出docker的最近版本命令：

yum list docker-ce.x86\_64 --showduplicates | sort -r



安装需要的软件包。

sudo yum install -y yum-utils device-mapper-persistent-data lvm2

使用以下命令设置稳定存储库。

sudo yum-config-manager \

--add-repo \

<https://download.docker.com/linux/centos/docker-ce.repo>

更新yum包索引

$ sudo yum makecache fast

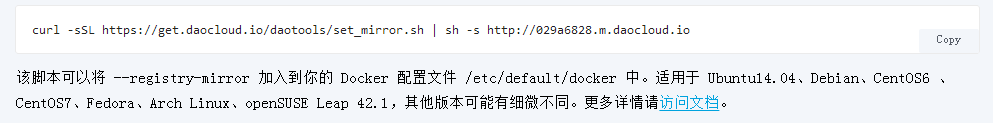
下载安装docker

$ sudo yum install docker-ce

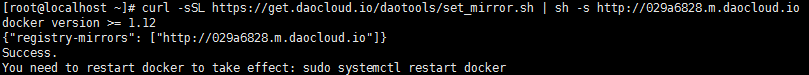
安装完成，我们启动docker。

systemctl start docker

配置国内镜像源，由于docker hub在中国没有CND代理，所以使用官方镜像库下载资源速度是非常慢的，所以我们需要将下载库指定到国内的镜像源，国内镜像源很多，这里我们使用的是daoclond。



curl -sSL https://get.daocloud.io/daotools/set\_mirror.sh | sh -s <http://029a6828.m.daocloud.io>

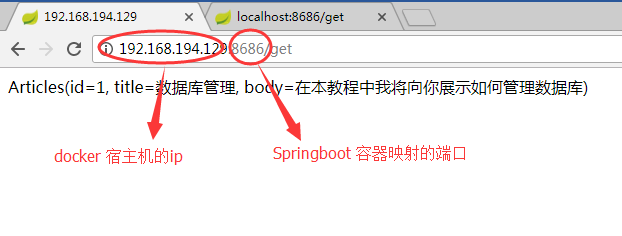
至此配置完成。

Docker进入容器的方法：

docker exec -it [容器id/容器名称] /bin/bash

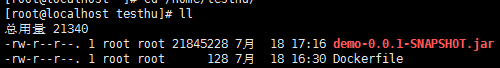
## 在Docker中部署运行SpringBoot项目

研究了docker的项目部署，总结几点为：安装java镜像、mysql镜像，写了一个简单的Dockerfile将上周日写的Springboot示例便以为一个镜像并运行，然后学习了docker之间的容器互联，web项目容器与mysql容器连接的这个问题研究了比较久，不过最后还是搞定了，Springboot项目完美的运行了起来，开心！

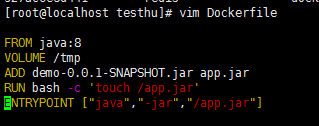


步骤：

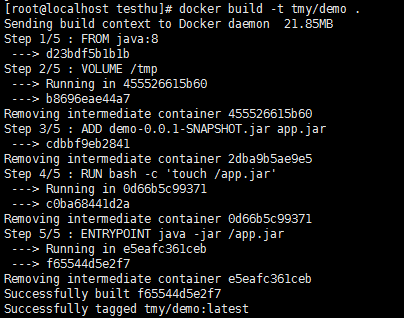
1.将项目打包并上传到linux服务器



2.编辑Dockerfile文件



3.根据Dockerfile创建Springboot项目镜像



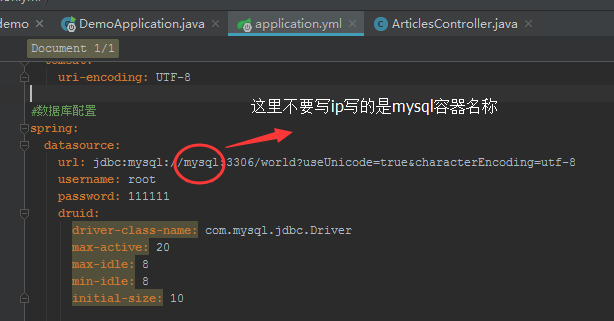
4.启动mysql镜像，创建一个容器，并设置root密码为111111



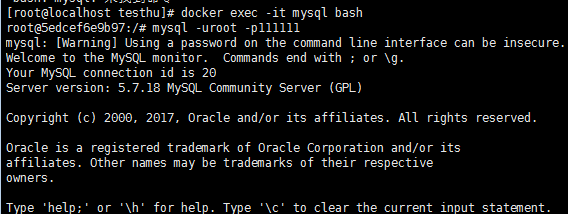
5.启动java项目镜像，并与mysql容器连接，映射宿主机8686端口；



注意：项目的数据库连接配置要注意，看下图。



引申：docker进入mysql容器进行操作的命令



---------------------------------注意-----------------------------------

mysql容器启动第一次启动的时候必须要包含root密码设置，否则启动失败

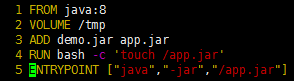
docker run -p 3306:3306 --name mysql2 -d -e MYSQL\_ROOT\_PASSWORD=111111 mysql

就是上面命令中的 -e MYSQL\_ROOT\_PASSWORD=111111 将root密码设置为111111

-----------------------------------------------------------

## Docker File解析

简单的解析一下Docker



第一行，表示新创建的镜像，基于哪一个镜像

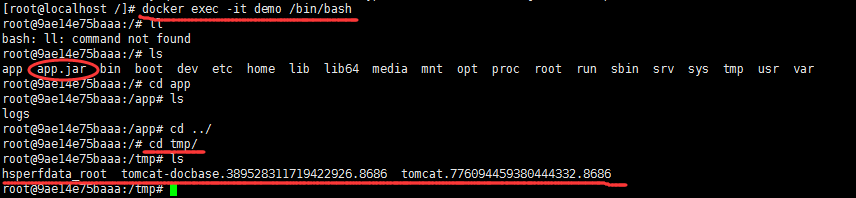
第二行，设置新镜像的数据在容器中的挂载目录

第三行，将宿主机中的文件复制到容器中，并重新命名（相当于mv命令）

第四行，更新容器中文件的时间戳

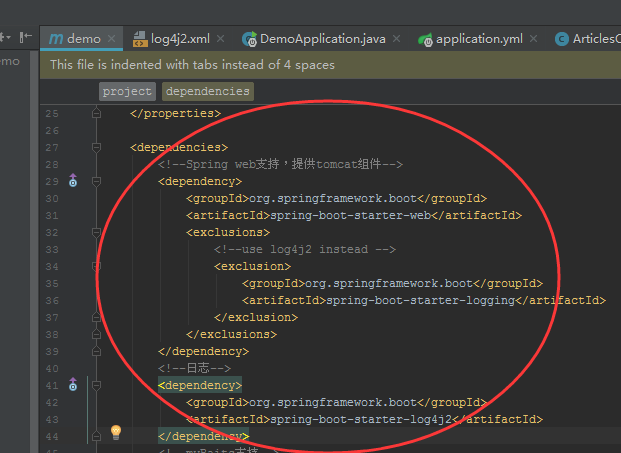
第五行，设定容器运行的时候，容器中运行的命令

我们进入到容器中看下，我们的容器根目录中有app.jar，tmp目录下，有我们的挂载数据。

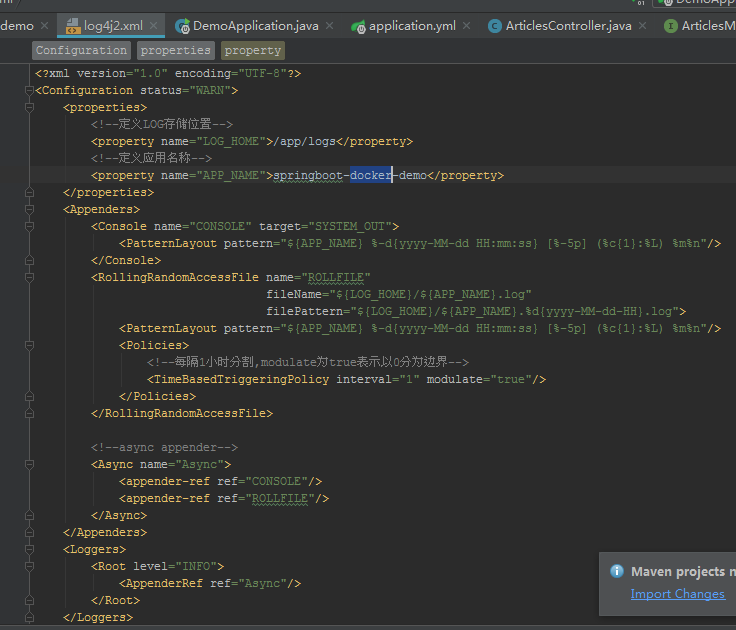


## 使用Docker的数据卷，来实现读取项目日志

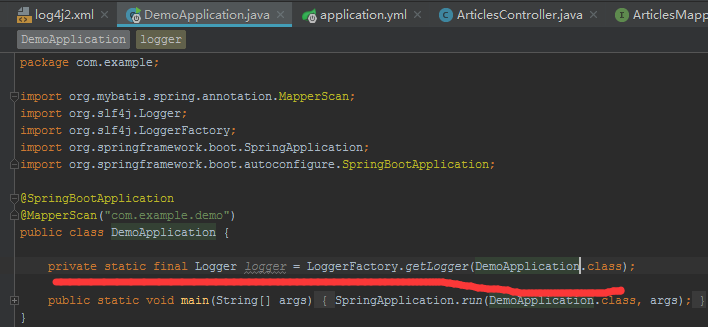
步骤：首先我们在我们的Springboot项目中添加日志支持包



然后我们写一个log4j2的xml配置文件



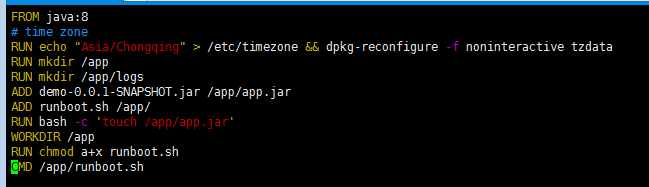
在我们需要查看日志的类中添加一行代码



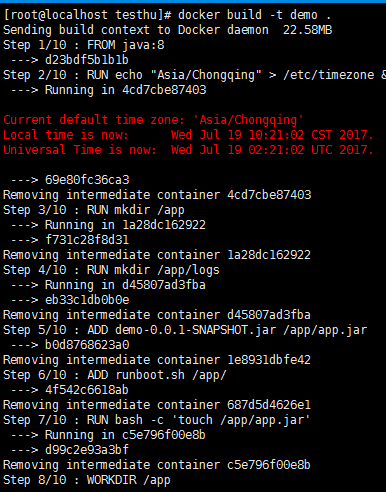
编写一个启动脚本文件runboot.sh



编写修改我们之前的Dockerfile文件



之后通过Dockerfile文件生成我们的镜像



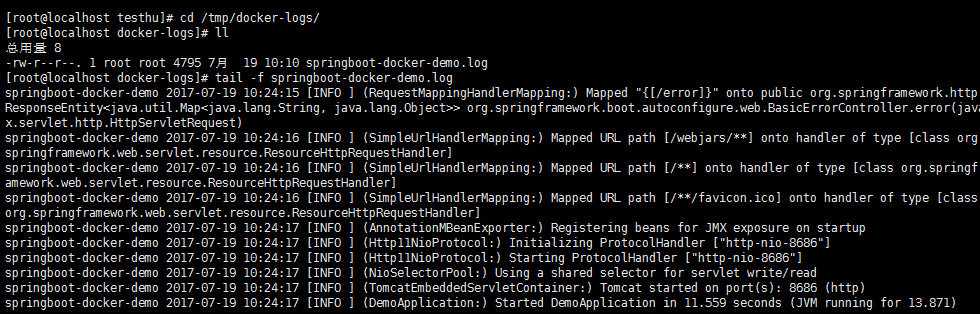
首先启动mysql容器然后运行我们的工程镜像





说明：-v参数就是配置log的参数

最后我们进入日志文件夹查看日志，成功！



## Docker上实现jenkins自动部署

我成功了idea上更新到github的代码成功通过jenkins部署到我的虚拟机服务器上！

接下来我要将整个流程记录下来。

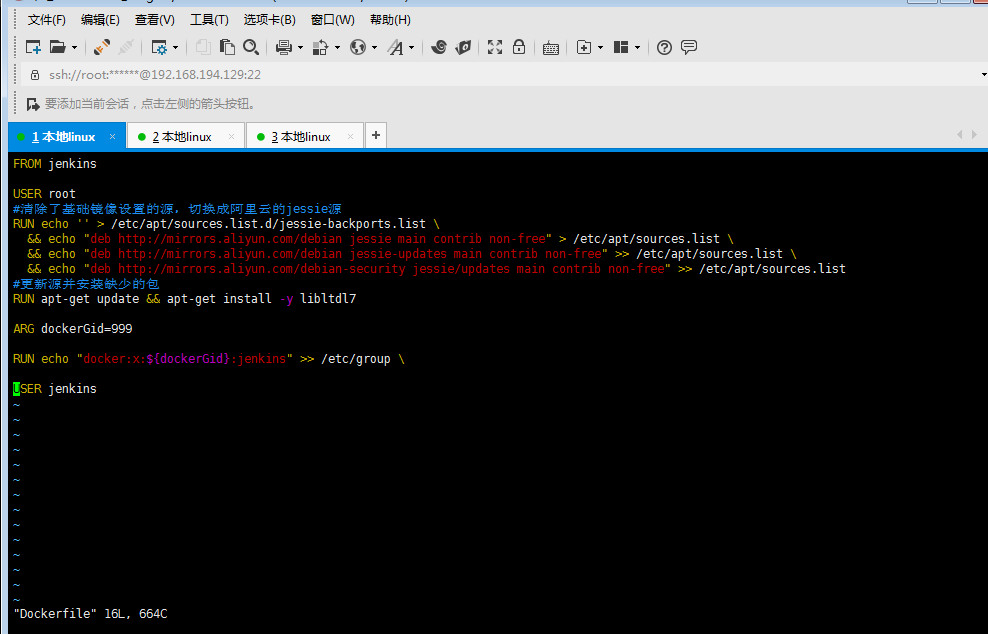
首先我们在docker 中 拉取 Jenkins的镜像。

docker pull jenkins

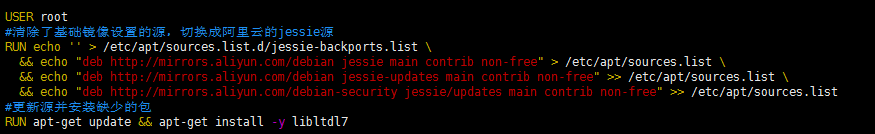
然后，我们开启一个gitHub的账号，并通过gitHub的客户端将我们的本地代码上传到gitHub，也可以直接用idea去提交代码到gitHub。我是先通过客户端上传的我的本地代码，然后用idea拉取的gitHub的代码。

最重要的部分，是配置jenkins。

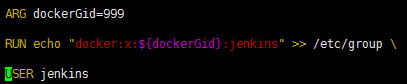
首先，在使用jenkins之前 我们需要将下载的jenkins镜像包装一下，生成一个我们自己的镜像，



官方的jenkins镜像是基于debian jessie的，dockerfile最好把源切换成国内的，就是这里害苦了我，导致没有权限执行命令，以及找不到各种资源。



这一部分就是转换源 ，使得我们执行命令的时候不会因为找不到各种文件报错。



这一部分，是将jenkins在linux构建中使用的用户划分到docker用户组，使其能够执行docker命令。

然后就是构建自己的jenkins镜像。

docker build -t myjenk .

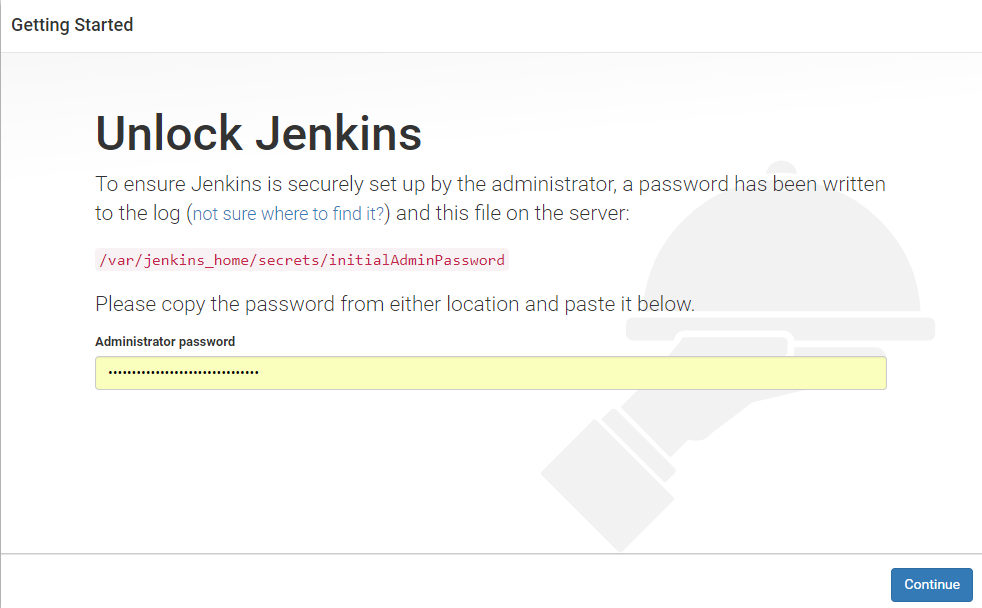
最后的 . 表示使用当前目录下的Dockerfile来构建镜像。

构建成功后，执行镜像，

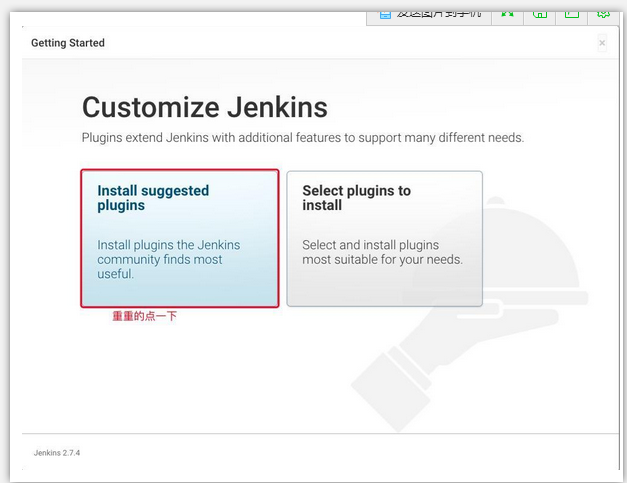
docker run -d -p 8083:8080 -p 8086:8083 -p 50000:50000 -v /var/jenkins\_home:/var/jenkins\_home -v /var/run/docker.sock:/var/run/docker.sock -v $(which docker):/usr/bin/docker --name myjenk myjenk

这里做了端口映射，将jenkins的默认端口8080 映射到了8083

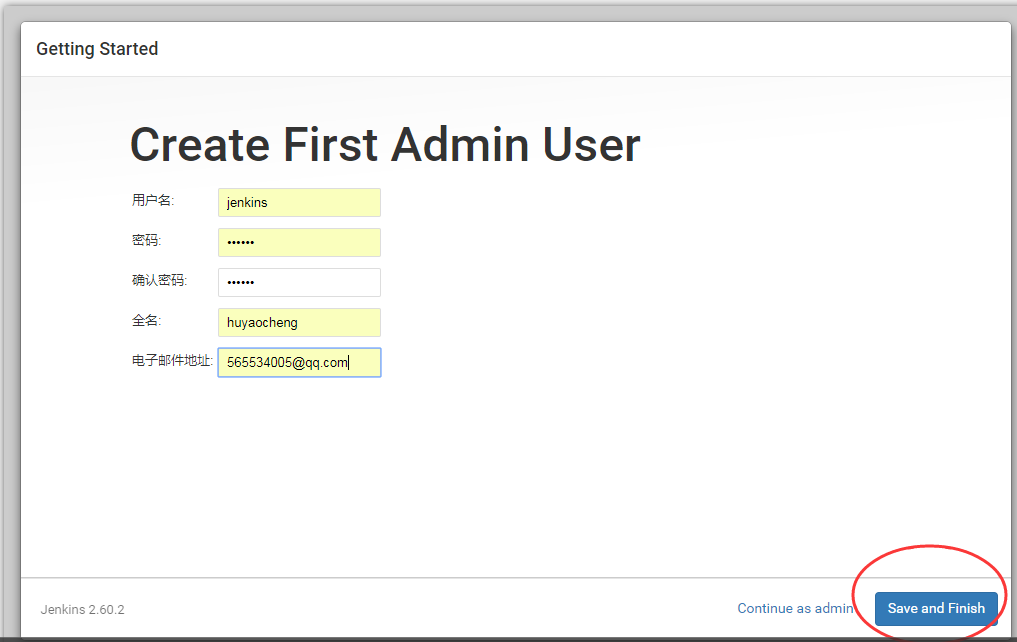
然后 我们访问虚拟机IP + 端口号8083 就能够访问到jenkins的主页面了

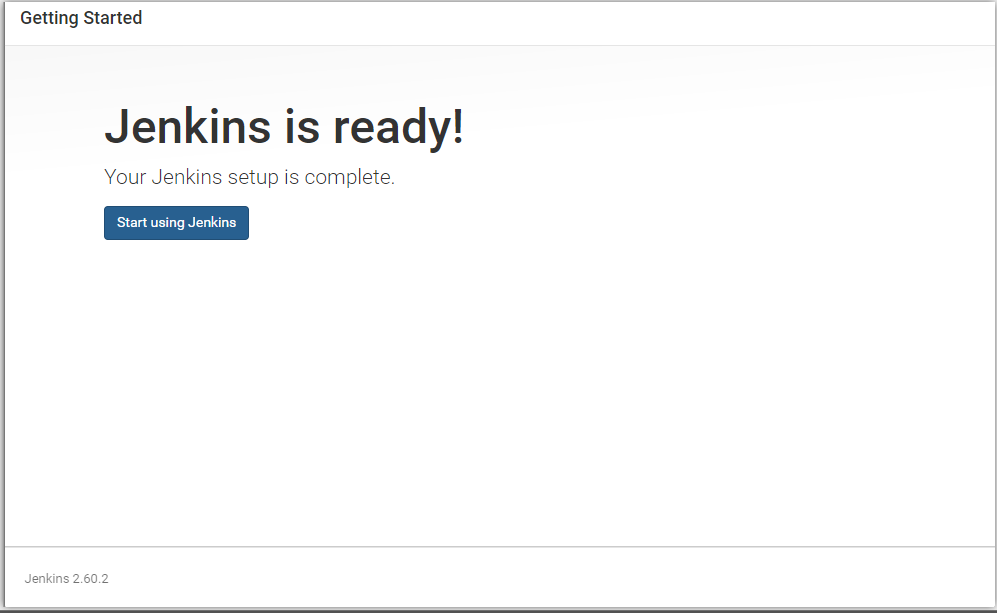


第一次访问需要去jenkins的工作目录下去取密码。上面已经给了密码所在的地方，打开文件就能开到密码，然后登陆。



这里我们选择安装下建议安装的组件，等待安装完成后 我们需要建立一个用户。

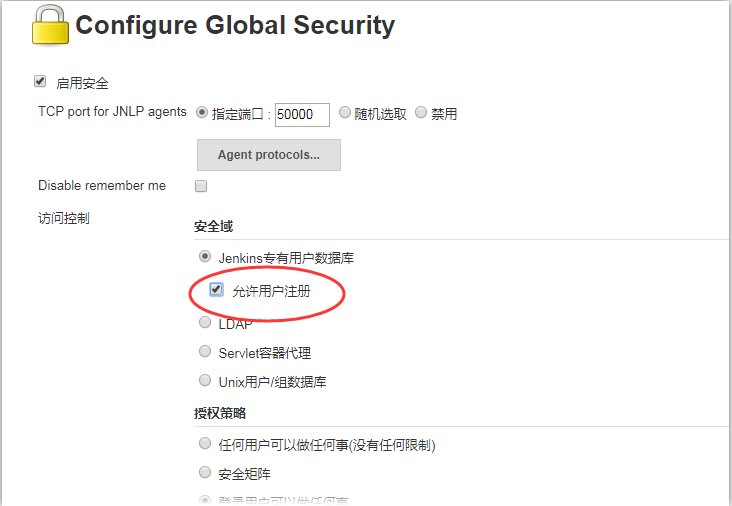




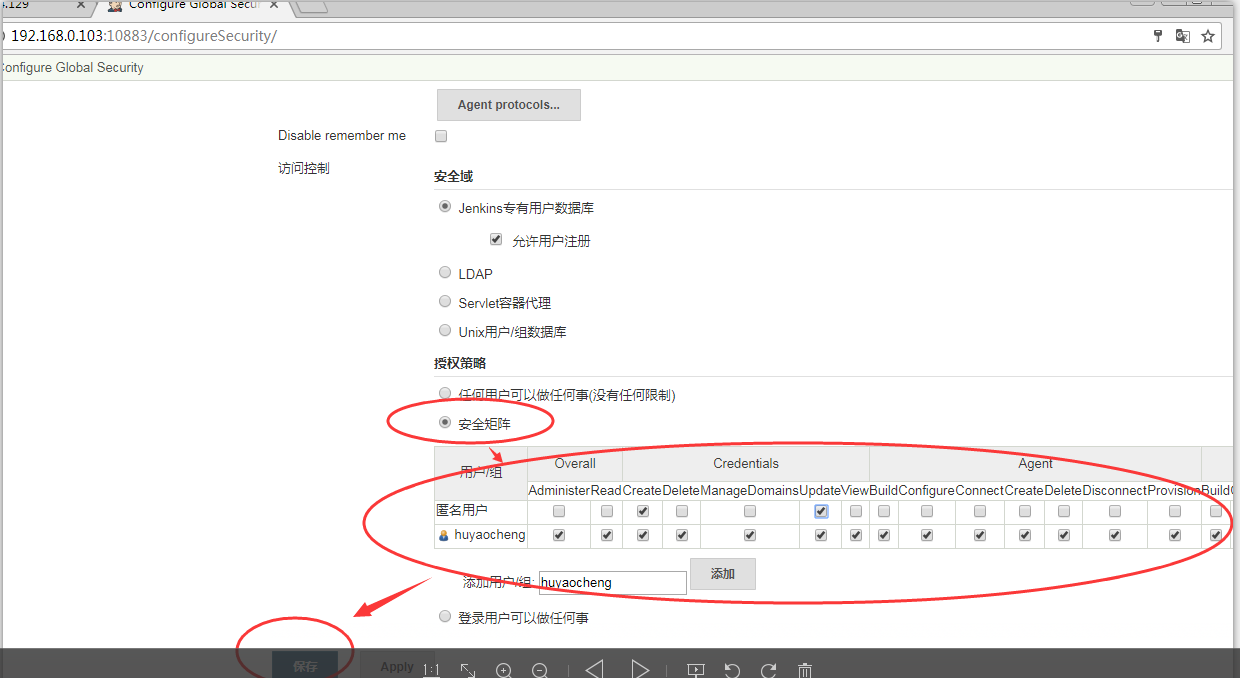
创建完成 接下来开始配置jenkins

首先我们需要先配置用户的权限。





这里可以选择允许注册，可以开启几个账号，然后分配权限。也可以不允许注册。但是如果没有选择允许注册，需要下面将自己的用户分配权限，不然会导致登录不了jenkins。



这里将自己的用户添加进去，否则登录不了jenkins。

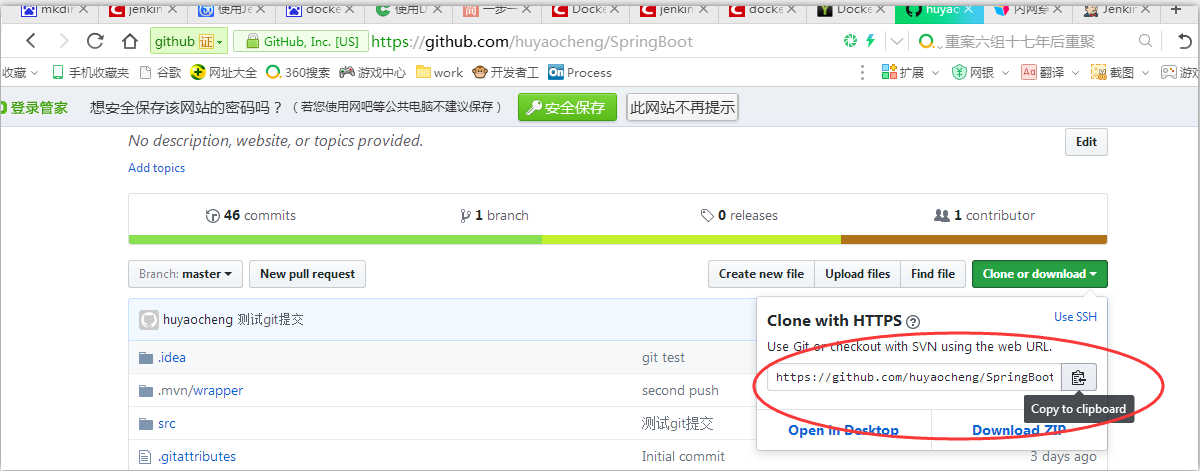
然后，我们新建一个工程项目

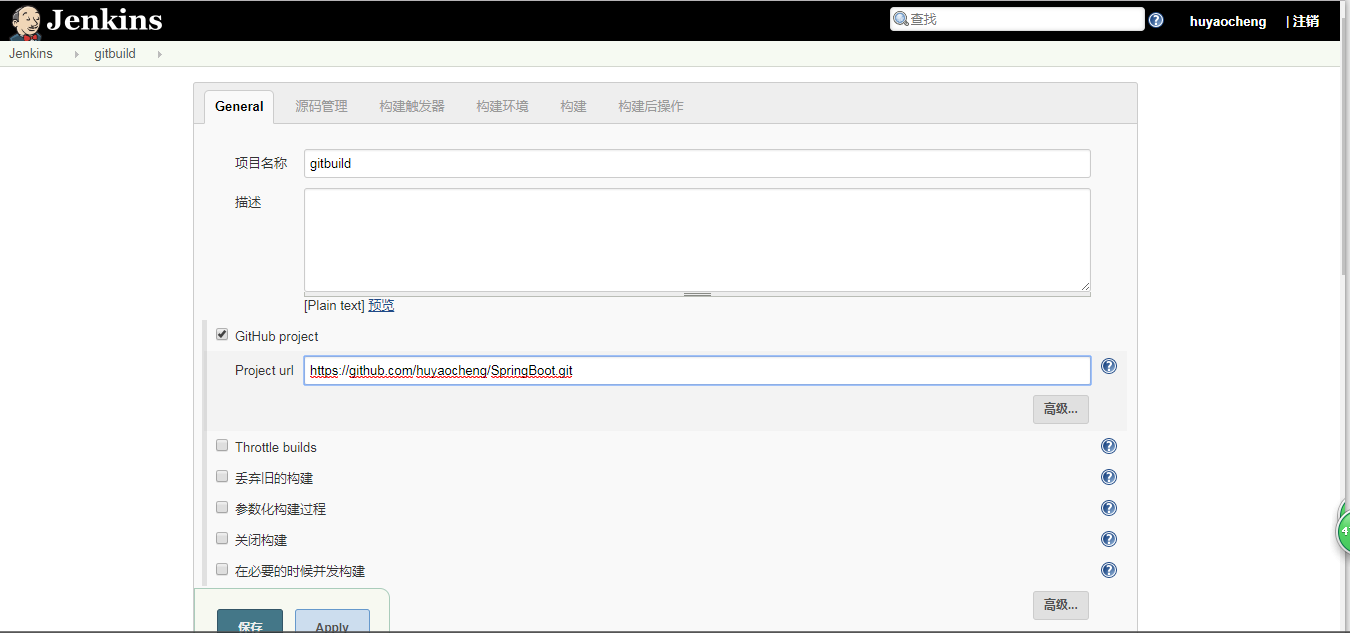


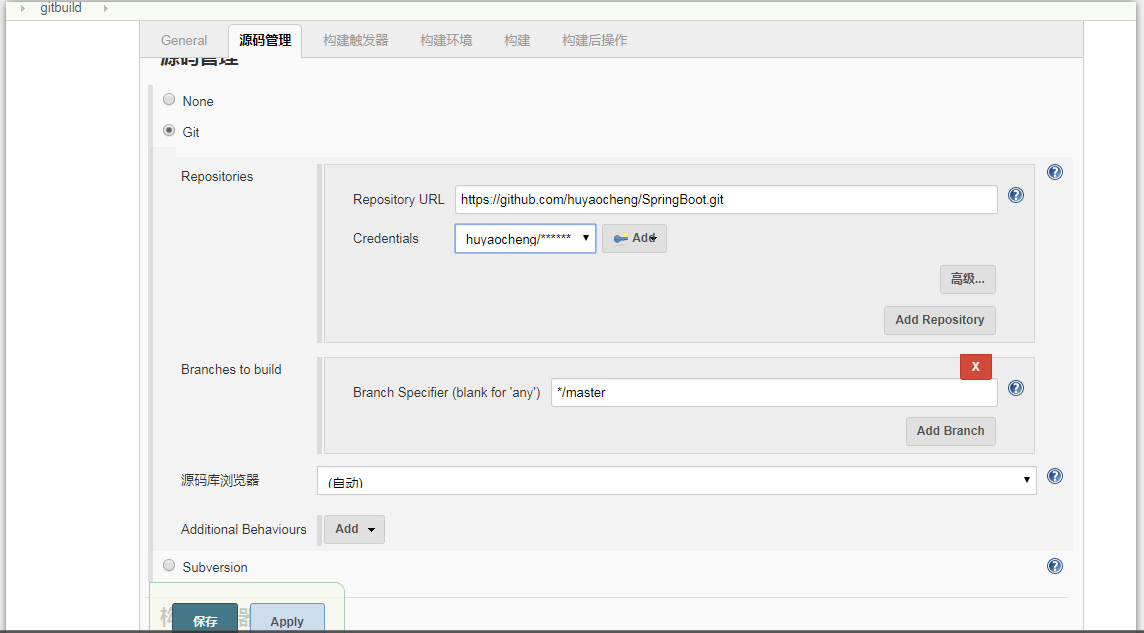
选择构建一个自由风格而的软件项目



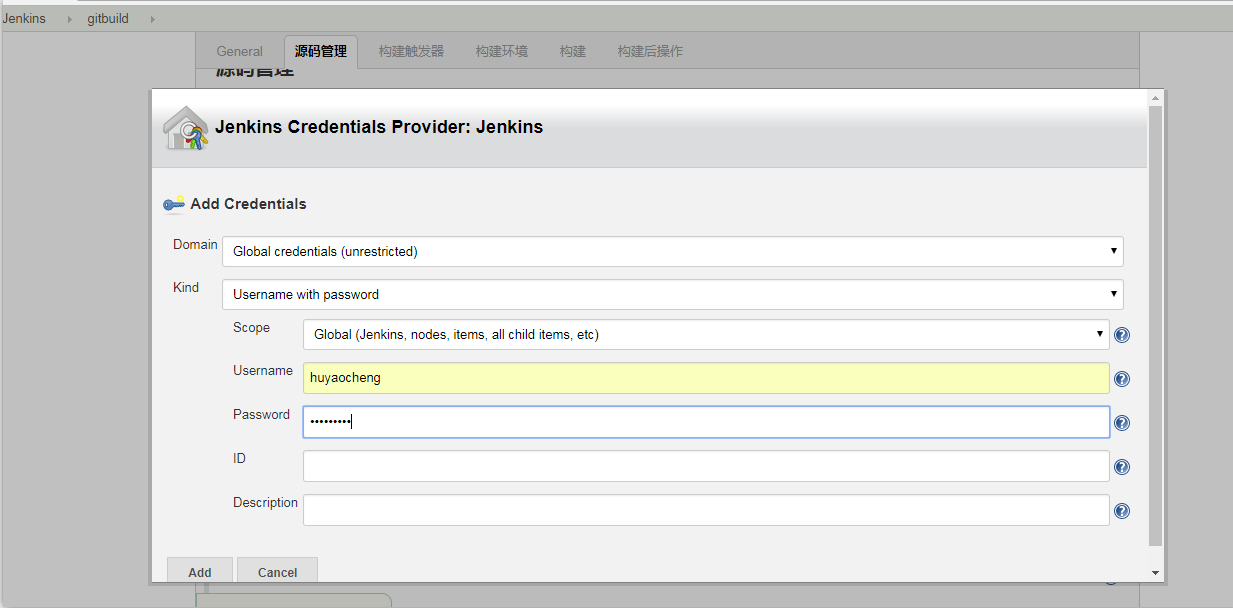
然后我们开始配置工程，首先我们需要到github上将我们的项目地址拷贝下。

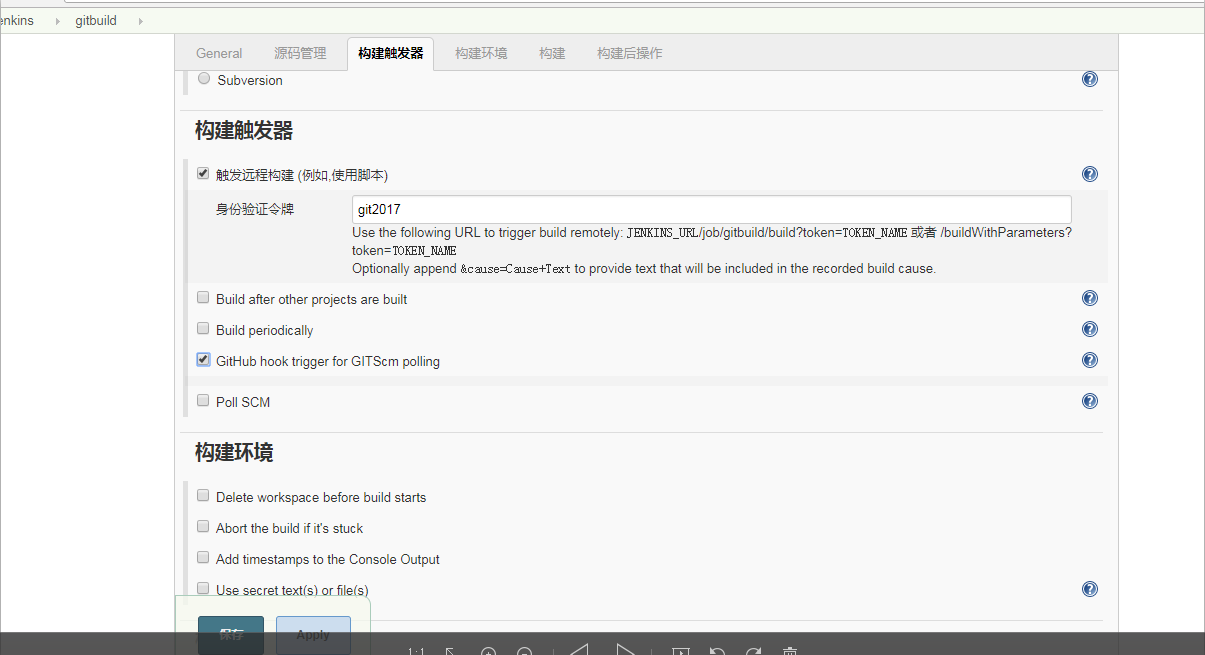






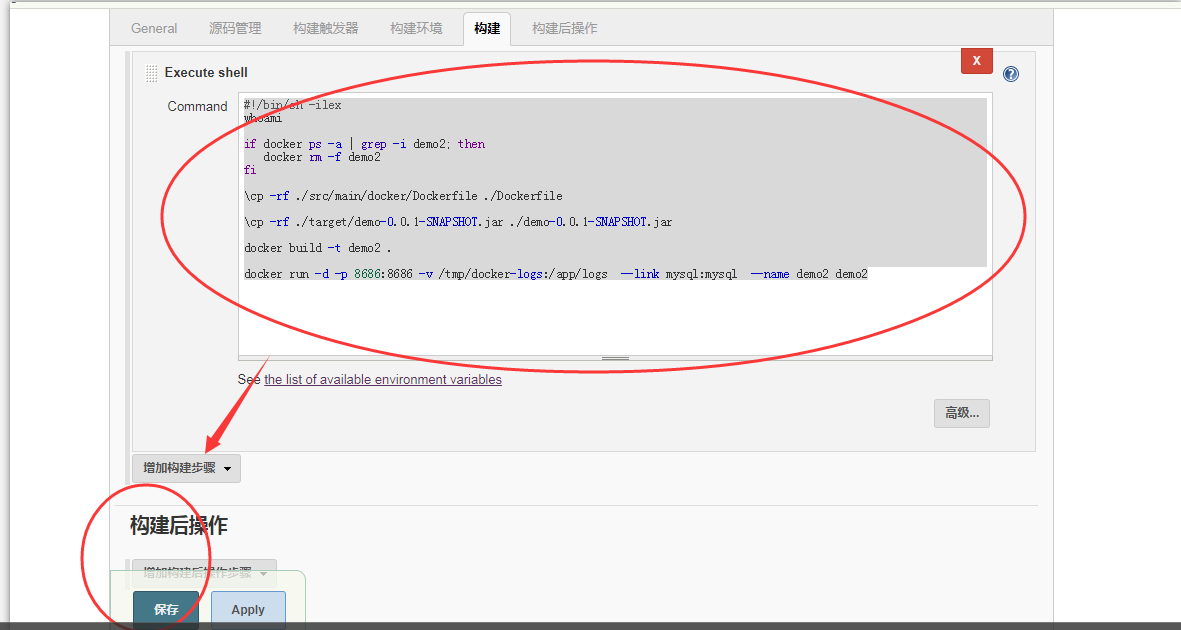
添加你的github账号





使用public 的github的话，这里可以不用配置触发远程构建。

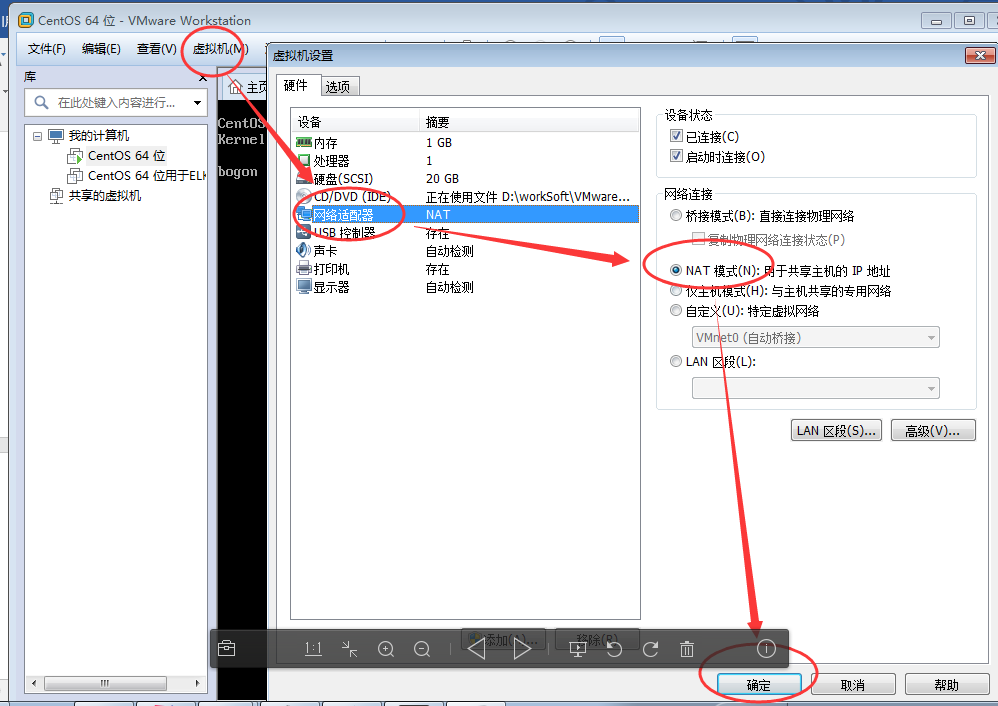
写好构建的shell脚本

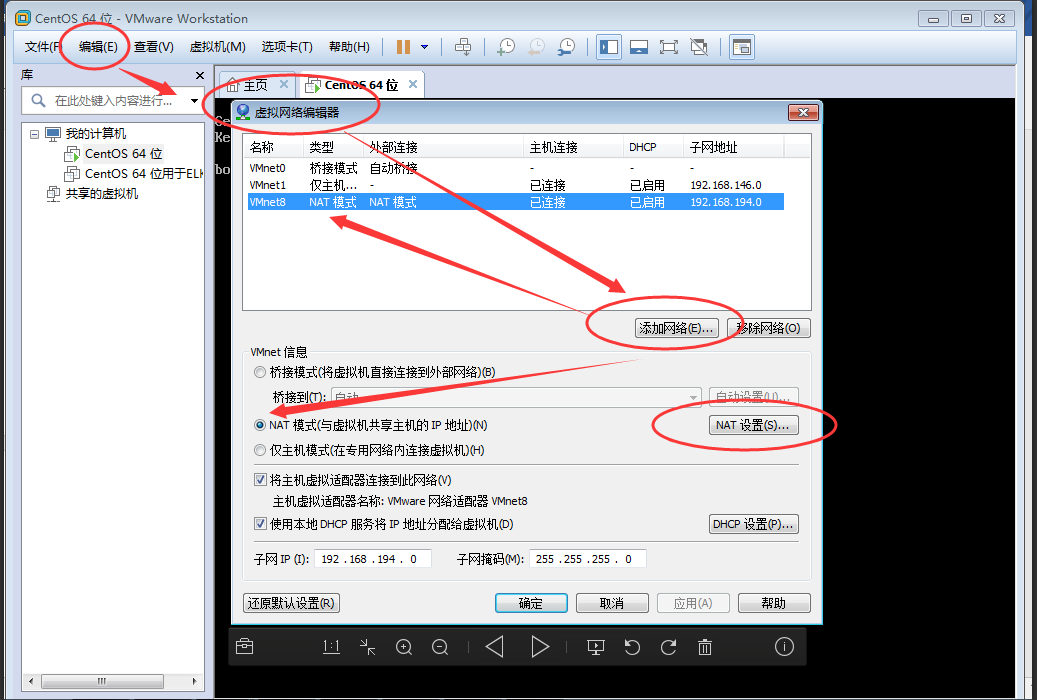


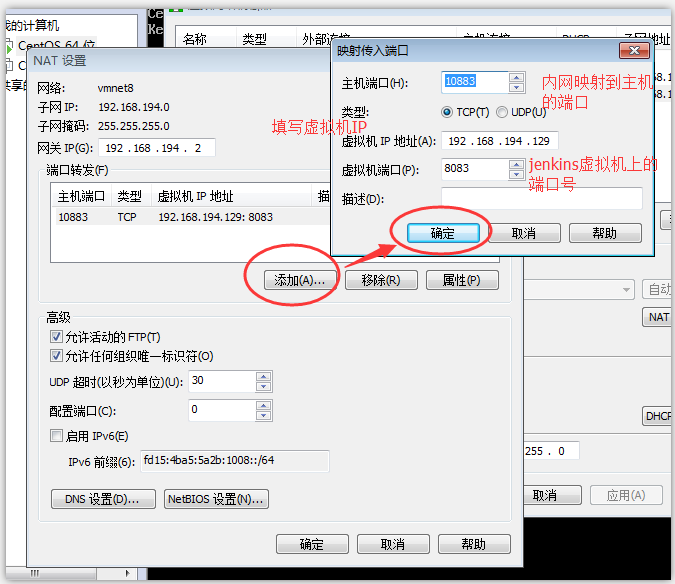
保存退出。

接下来我们需要另一个重要的步骤。

将我们的虚拟机网络映射到本机的本地局域网，然后再将局域网映射到外网，使github上更新的推送能够通知到我们虚拟机中的jenkins。

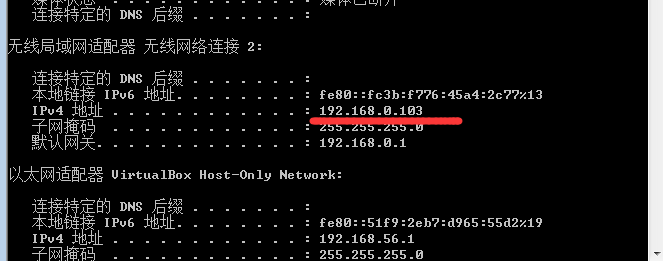






此时我们已经将虚拟机的ip端口映射到了主机的ip和端口

此时cmd下ipconfig查看下主机端口



Ip为192.168.0.103 加上映射端口10883 访问



正常登陆 映射成功

登陆花生壳，然后注册账号 得到了一个免费域名 下载客户端 办理内网穿透。



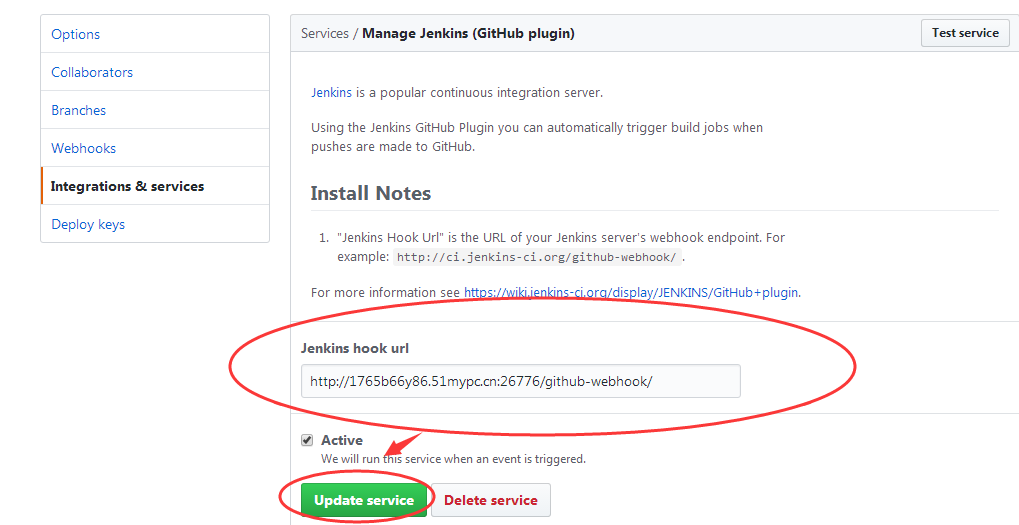
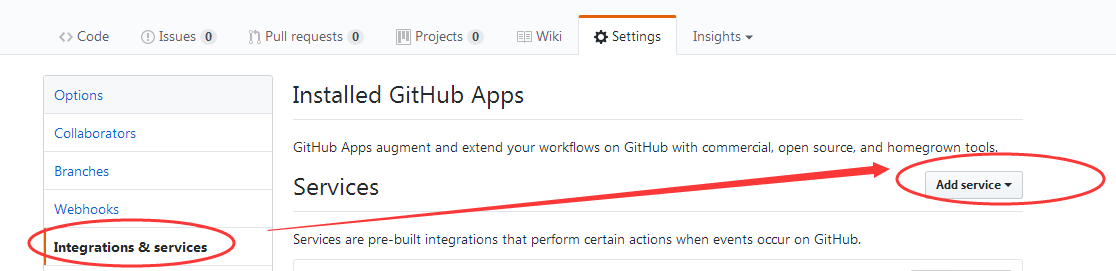
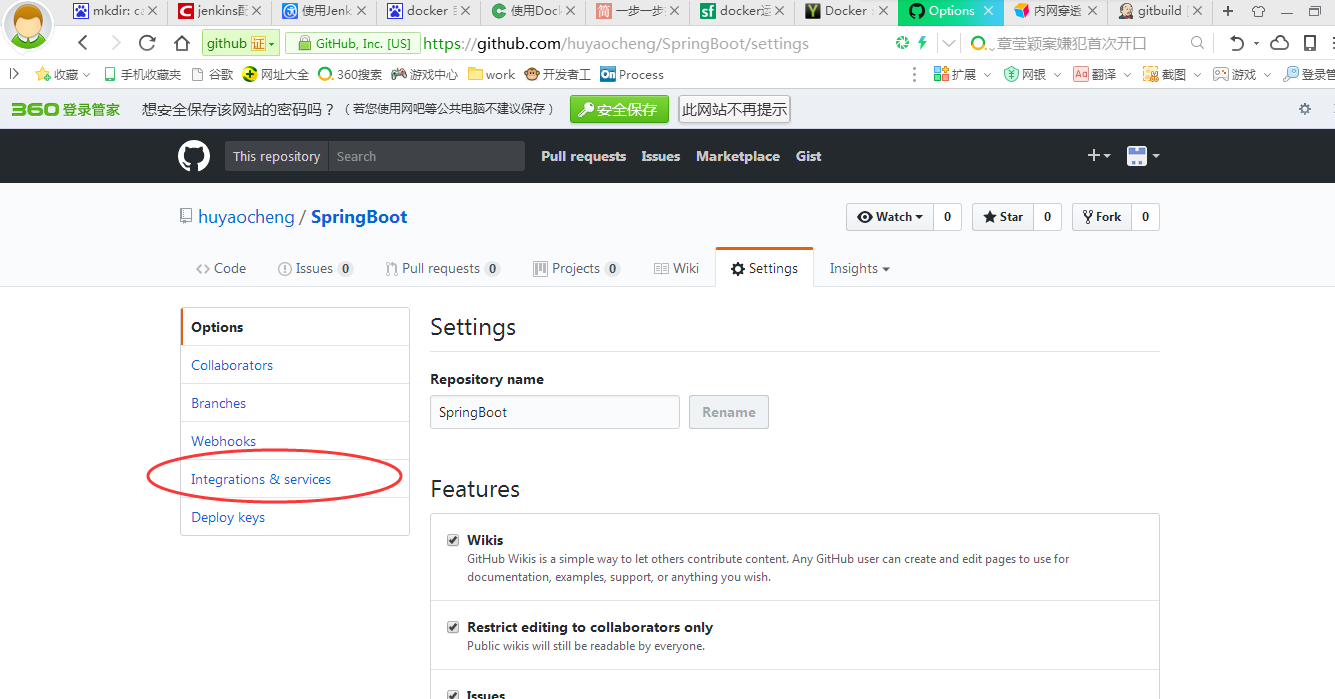
添加映射，将自己的主机ip和端口映射到自己的免费域名下。

然后用映射的外网域名访问jenkins

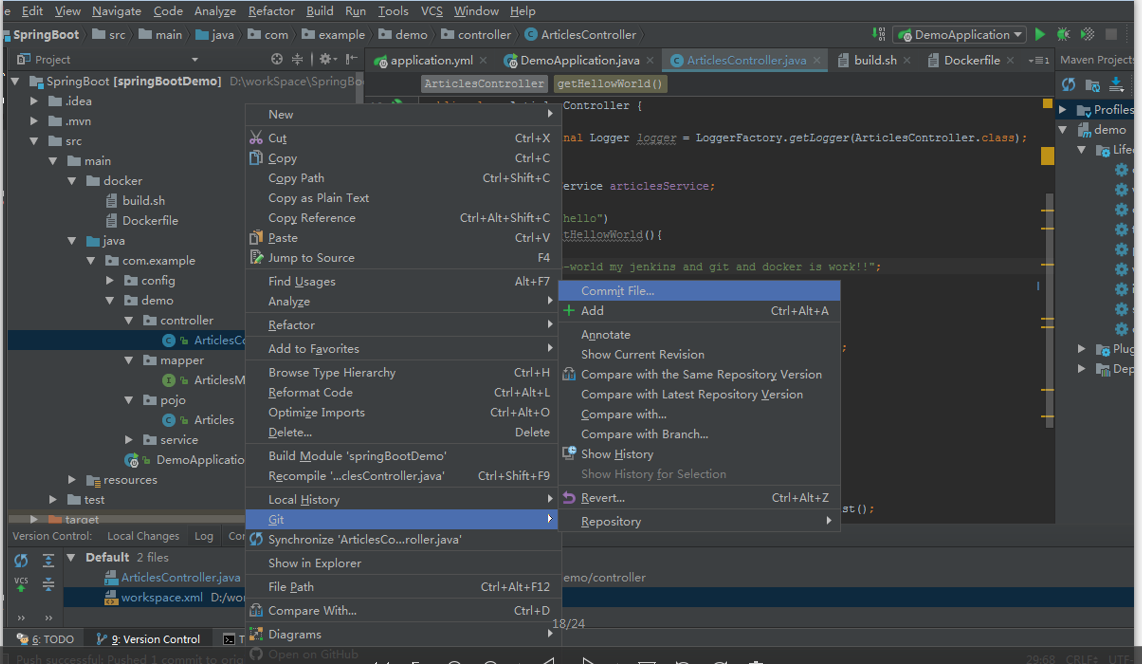


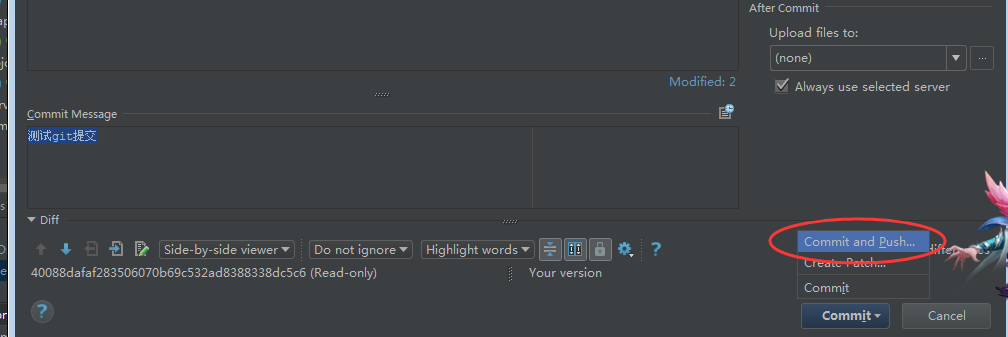
正常，说明映射成功。

然后回到github我们去配置jenkins接受github请求的地址。



添加自己的花生壳上映射的域名后面加上/github-webhook/后缀点击修改服务。

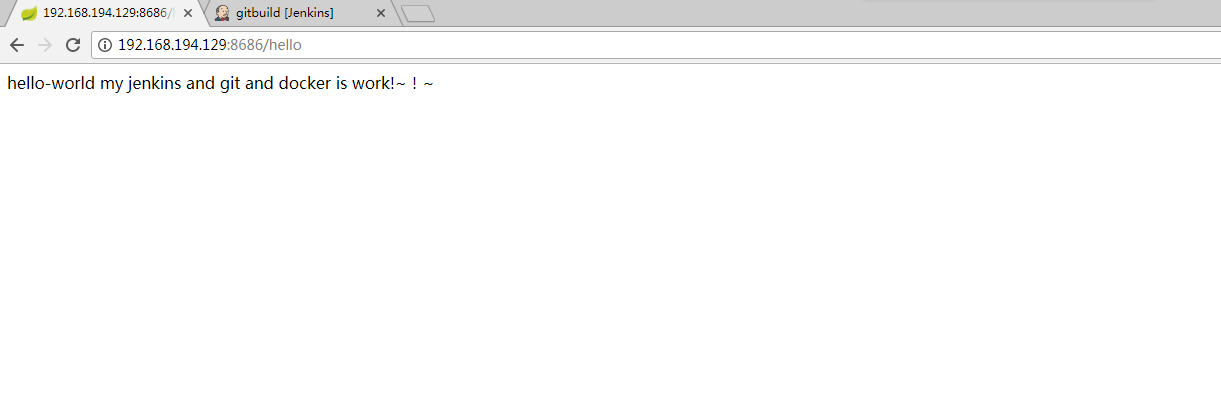
然后我们回到idea中 修改我们的代码然后提交到github。



回到jenkins 然后等待。。。

构建成功！！！！





给自己的辛苦加油和努力点赞！！

写一个目标，优化整个流程，目前的问题：

1.使用docker 命令进行打包 每次要重新下载所有资源包，这个要改进。

2.整个流程还是不够优雅，比如maven打包的方式，是使用docker maven插件 还是在Dockerfile中打包 两者那个更好。

3.整个流程需要优化，知道那些事必须要配置的那些不需要配置

即将要学习的，Dockerfile文件的编写， 将整个项目的所有容器打包成一个镜像。

深入了解docker 使用以及原理，学习linux系统。

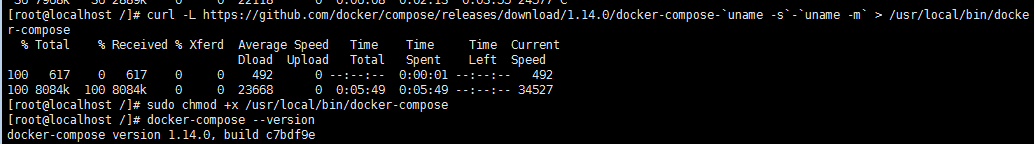
模拟完成一个docker项目的整个迁移，试验下docker的优点。

Jenkins+github+docker的分布式自动部署。

## Docker-compose插件

今天抽空了解了一下 docker-compose插件 部署docker环境，并简单试验了一下。

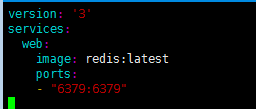
安装docker-compose



下载：curl -L https://github.com/docker/compose/releases/download/1.14.0/docker-compose-`uname -s`-`uname -m` > /usr/local/bin/docker-compose

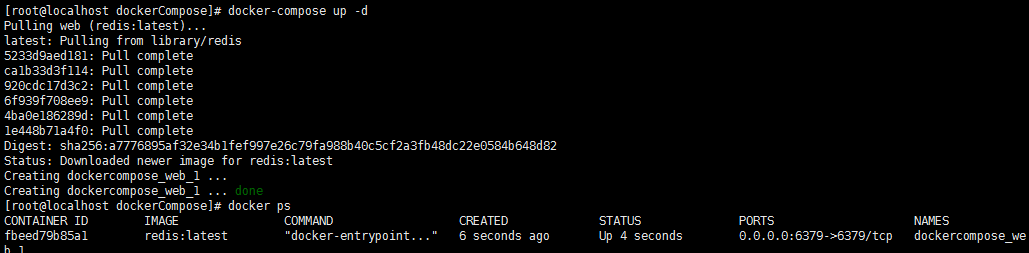
赋予其执行权限：sudo chmod +x /usr/local/bin/docker-compose

创建docker-compose的可执行文件docker-compose.yml



文件意思是拉取最新的redis镜像，并将6379端口映射到宿主机6379端口访问。

通过docker-compose启动docker镜像，可以看到镜像自动下载，并且在后台运行了。



通过这个功能可以了解到可以使用docker-compose来瞬间搭建一个可用的docker开发环境，在也不用花费好久时间在新机器上搭建各种开发软件了。只需要一个docker，一个docker-compose就可以完成开发环境的搭建了。

在下载nginx镜像的时候需要注意，就是讲宿主机的nginx.conf配置文件挂载到docker的nginx容器的时候需要准备一个默认的可以用的配置文件来挂载，不然在挂载的时候宿主机的配置文件会覆盖掉nginx容器的配置文件，导致nginx无法正常工作。

Nginx与其他镜像不太相同，挂载的文件如果没有不会从镜像中去自助拷贝需要预先准备好（典型nginx.conf,其他的还没有这个问题），并且会覆盖掉容器中的配置，所以不建议用目录去挂载（如果用一个目录去挂载nginx的配置目录，如果没有准备好相应的配置相当于将nginx的配置目录关铺覆盖干掉了），用到那个文件，就去挂载那个文件。所以，建议用Dockerfile去按照自己的需求去配置自己的nginx镜像，这样不会有太多的坑。

全部研究没问题了在贴上方法。

今天找到了为什么docker-compose中配置的nginx无法访问,原因是我所挂载的目录到了nginx容器中没有执行的权限，这个是因为Centos7的安全模块selinux把权限禁止了。目前试了网上的一些方法。在docker run 中添加 --privileged=true 这个参数是可以的，或者在执行docker-compose.yml之前将nginx要挂载的目录添加到selinux的白名单之中：

chcon -Rt svirt\_sandbox\_file\_t  /dockervolumes/nginx/html

chcon -Rt svirt\_sandbox\_file\_t  /dockervolumes/nginx/conf.d

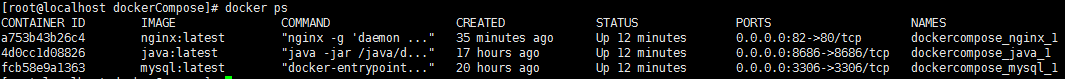
privileged参数简单介绍：可以使容器内的root拥有真正的root权限，否则容器内的root只是宿主机上的一个普通权限。Privileged启动容器，可以看到很多host上的设备，并且可以执行mount，甚至允许你在docker容器中启动docker容器。

然后执行docker-compose.yml文件 文件详情如下：



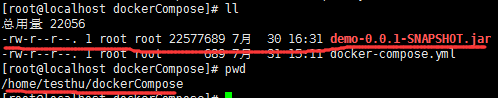
执行 docker-compose up 试运行，如果没有报错 执行docker-compose start

下载完镜像之后，我们发现容器已经启动：

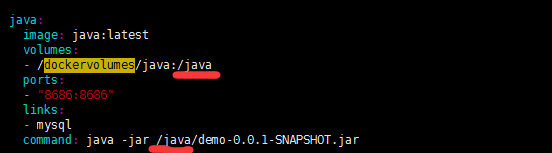


这里列出nginx的配置文件，只是一些简单的配置，配置了一个请求转发，为了试验nginx与web项目的连接，以及排查了docker中使用nginx的一系列问题，主要注意几点：

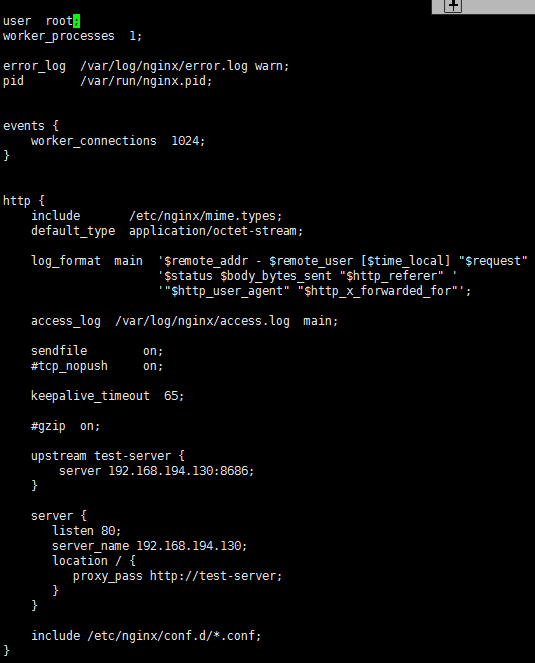
1. nginx.conf在挂在之前一定要准备好，或者不用官方原镜像，采用Dockerfile来定制自己的nginx镜像（推荐）
2. Centos7下挂载nginx的两个目录（conf.d 还有 html）的时候，挂载的目录到了nginx的容器中是没有权限执行的，原因是因为Centos7的安全模块selinux把权限禁止了。亲测了两种可以解决的方法，具体看前面。
3. java容器的挂载目录后，要把你的可执行jar包，放到宿主机下的（注意是宿主机）挂载目录，然后运行时，jar包的根目录要写docker容器的挂载目录,比如：



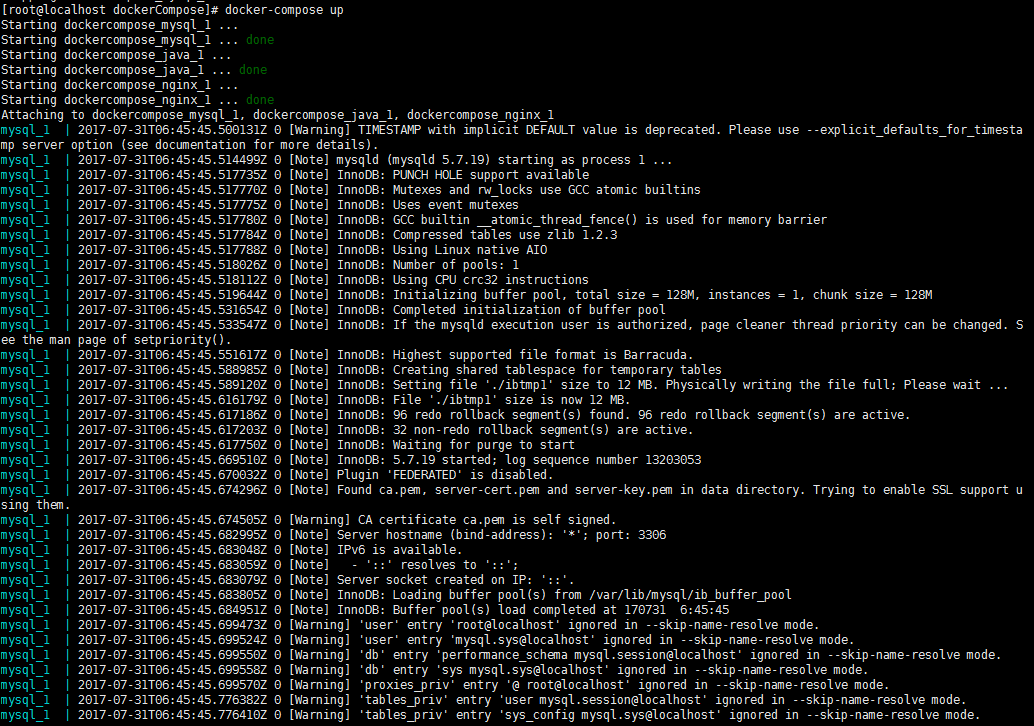
可以看到jar包上传到了宿主机下的挂载目录。

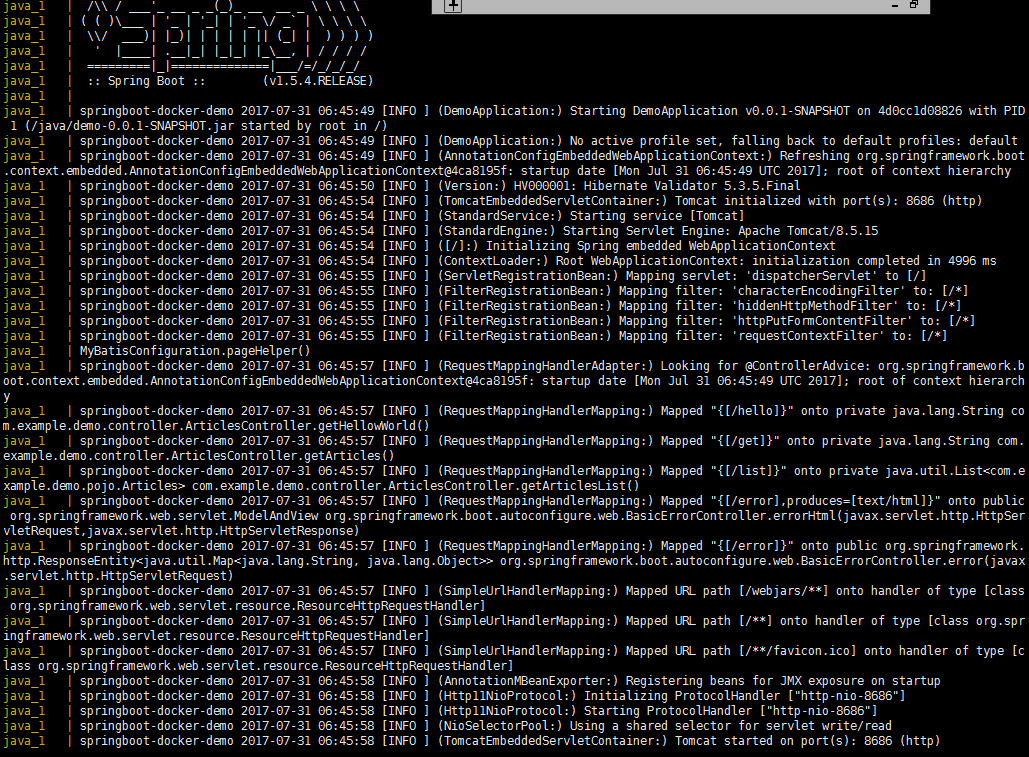


docker-compose.yml中jar包的根路径写的是docker容器的挂载目录

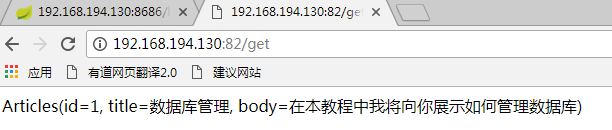


启动docker-compose.yml





访问nginx服务器



成功访问！