一．创建虚拟机

1. 安装vmware

2. 下载ubuntu ISO镜像

[www.ubuntu.com/download下载 ubuntu](http://www.ubuntu.com/download下载ubuntu) server的LTS版本

1. 通过镜像创建虚拟机（必要时克隆新版本）

参考此处进行创建：[在Vmware虚拟机上安装Ubuntu.iso镜像的完整指南\_ubuntu iso-CSDN博客](https://blog.csdn.net/m0_64986186/article/details/132680365)

注意：

使用iso文件，iso.torrent文件不可以

初始化设置需要逐步阅读

建好虚拟机后开机，使用sudo passwd root 为root用户生成密码

1. 安装linux远程控制管理xshell

xshell必须用普通用户登录，登录后输入su，输入root密码，以root账户登录。

修改root密码的方法: sudo passwd root

1. 更新数据源（将 Ubuntu 的数据源修改为国内数据源）

# 查看系统版本

lsb\_release -a

若输出结果为

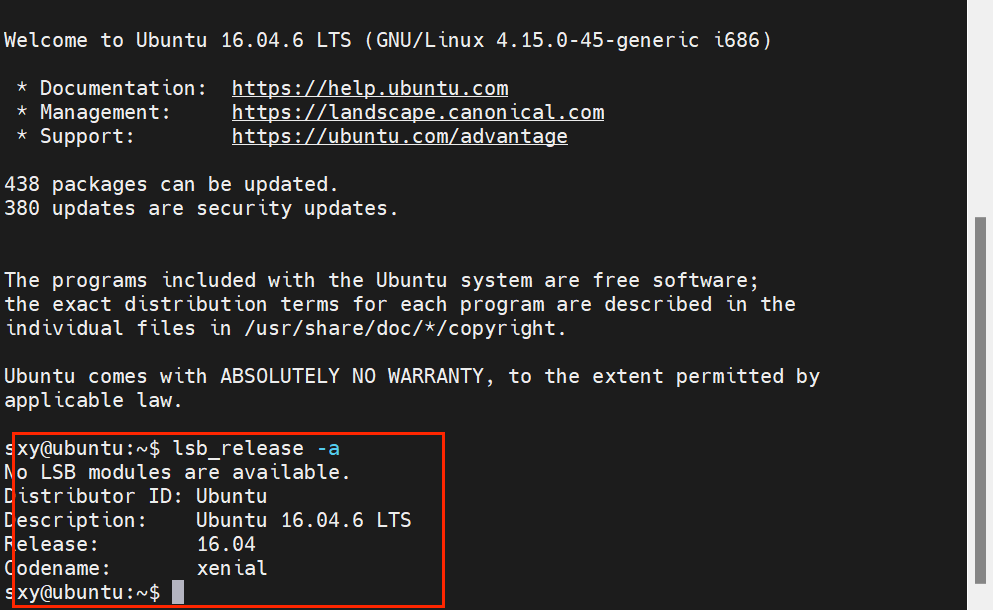
No LSB modules are available.

Distributor ID: Ubuntu

Description: Ubuntu 22.04.3 LTS

Release: 22.04

Codename: jammy



\*\*注意：\*\* Codename 为 `jammy`，该名称为我们 Ubuntu 系统的名称，修改数据源需要用到该名称

# 编辑数据源

vi /etc/apt/sources.list

ubuntu24版本的 sudo vim /etc/apt/sources.list.d/ubuntu.sources

deb http://mirrors.aliyun.com/ubuntu/xenial main restricted universe multiverse

deb http://mirrors.aliyun.com/ubuntu/ xenial-security main restricted universe multiverse

deb http://mirrors.aliyun.com/ubuntu/ xenial -updates main restricted universe multiverse

deb http://mirrors.aliyun.com/ubuntu/ xenial -backports main restricted universe multiverse

删除全部内容并修改为

deb http://mirrors.aliyun.com/ubuntu/ jammy main restricted universe multiverse

deb <http://mirrors.aliyun.com/ubuntu/> jammy-security main restricted universe multiverse

deb http://mirrors.aliyun.com/ubuntu/ jammy-updates main restricted universe multiverse

deb <http://mirrors.aliyun.com/ubuntu/> jammy-backports main restricted universe multiverse

# 更新数据源

apt-get update

二．安装docker

1. 更新数据源

apt-get update

1. 安装docker所需依赖

apt-get -y install apt-transport-https ca-certificates curl software-properties-common

1. 安装GPG证书

curl -fsSL <http://mirrors.aliyun.com/docker-ce/linux/ubuntu/gpg> | apt-key add -

add-apt-repository "deb [arch=amd64] http://mirrors.aliyun.com/docker-ce/linux/ubuntu $(lsb\_release -cs) stable"（增加阿里云数据源）

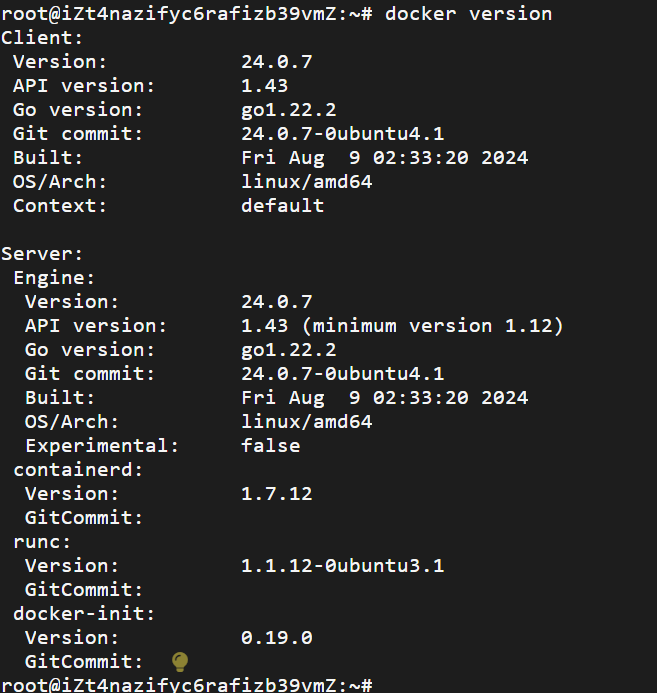
1. 更新并安装Docker CE

apt-get update && apt-get install -y docker-ce

1. 查看版本

docker version

应包含client 和server



1. 配置加速器

由于网络原因，我们在 pull Image 的时候，从 `Docker Hub` 上下载会很慢，所以，国内的 Docker 爱好者们就添加了一些国内的镜像 (mirror),方便大家使用。

可以参考此处选择可用的国内镜像加速器地址

[国内无法拉取Docker镜像了？这些方法拯救你的Docker-腾讯云开发者社区-腾讯云 (tencent.com)](https://cloud.tencent.com/developer/article/2434428)

#选择镜像加速器，复制加速器地址（个人专属加速器地址）

[https://docker.m.daocloud.io](https://cloud.tencent.com/developer/tools/blog-entry?target=https%3A%2F%2Fdocker.m.daocloud.io%2F&source=article&objectId=2434428" \t "_blank)（使用自己的地址）

#创建/etc/docker/daemon.json文件

vi /etc/docker/daemon.json

写入以下内容：

{

"registry-mirrors":["[https://docker.m.daocloud.io](https://cloud.tencent.com/developer/tools/blog-entry?target=https%3A%2F%2Fdocker.m.daocloud.io%2F&source=article&objectId=2434428" \t "_blank)"]

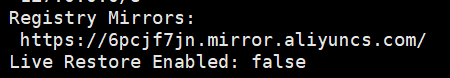
}

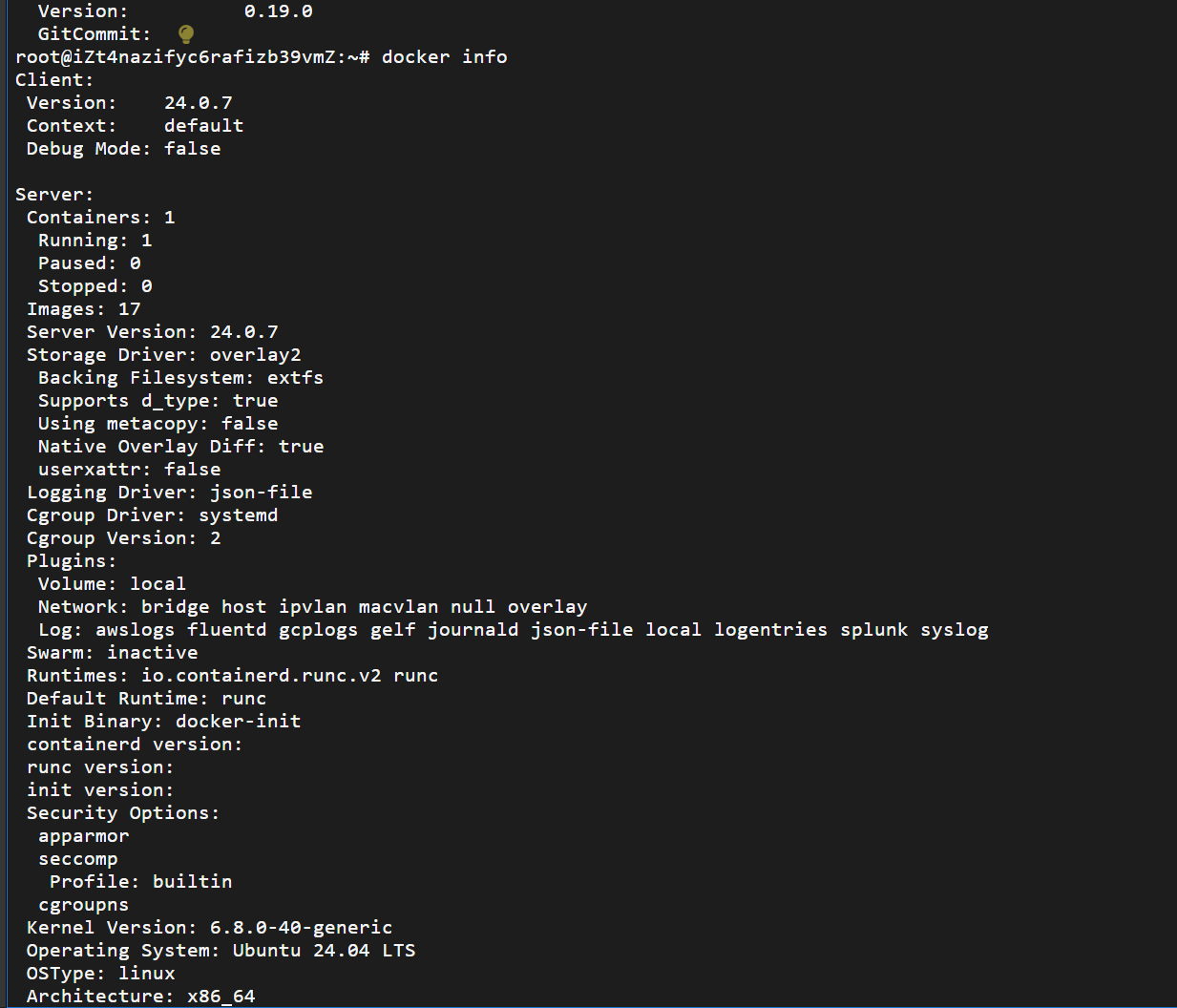
#重启docker

systemctl restart docker

#验证配置是否成功

docker info



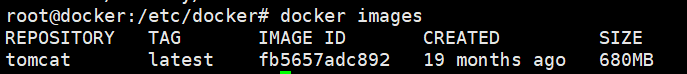


三．安装 Tomcat

#下载tomcat镜像

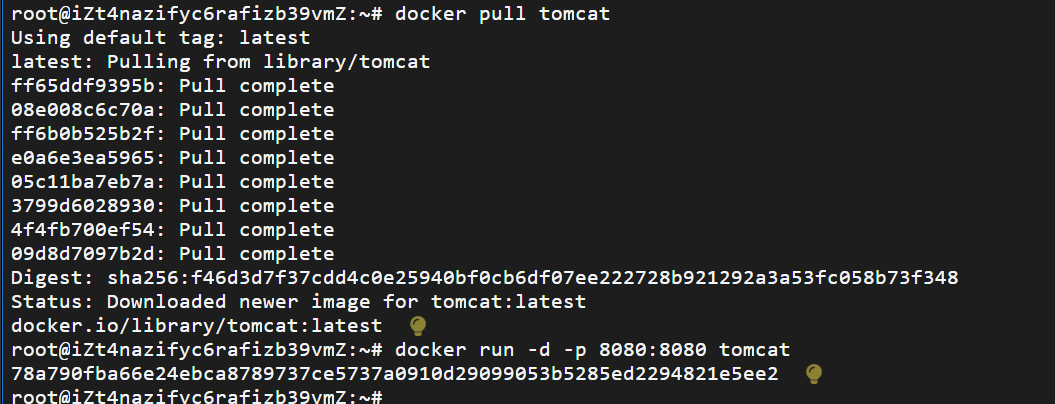
docker pull tomcat

docker images（查看镜像）



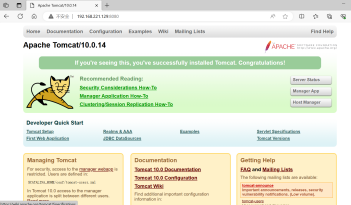
#运行tomcat

docker run -d -p 8080:8080 tomcat



#测试tomcat

浏览器中输入ip：8080，是否进入tomcat登录界面



若Tomcat:HTTP状态 404，有可能是Tomcat版本太新（Tomcat/10.0.），解决方法如下：

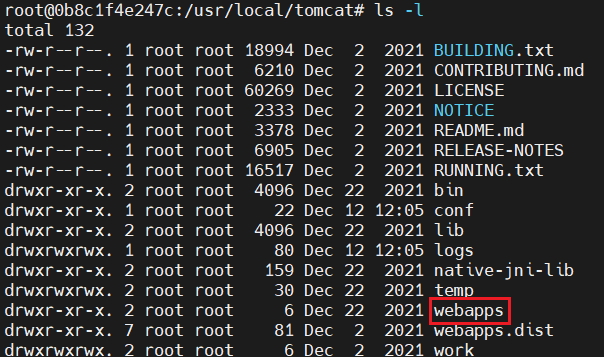
1）首先以前台交互式进入 Tomcat 10.0 镜像的容器 t1 中（t1为容器名，容器名可通过命令查询：docker ps --format '{{.Names}}'）

docker exec -it t1 /bin/bash

IMG_256

查看当前目录。

ls -l



所有网页资源都应该在 webapps 目录下，进入查看。

$ cd webapps

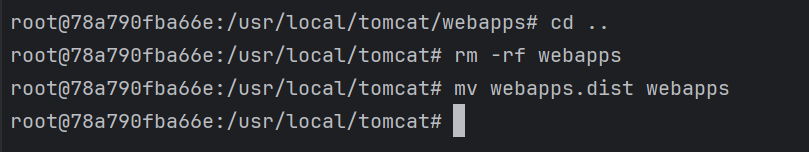
$ ls -l

IMG_256

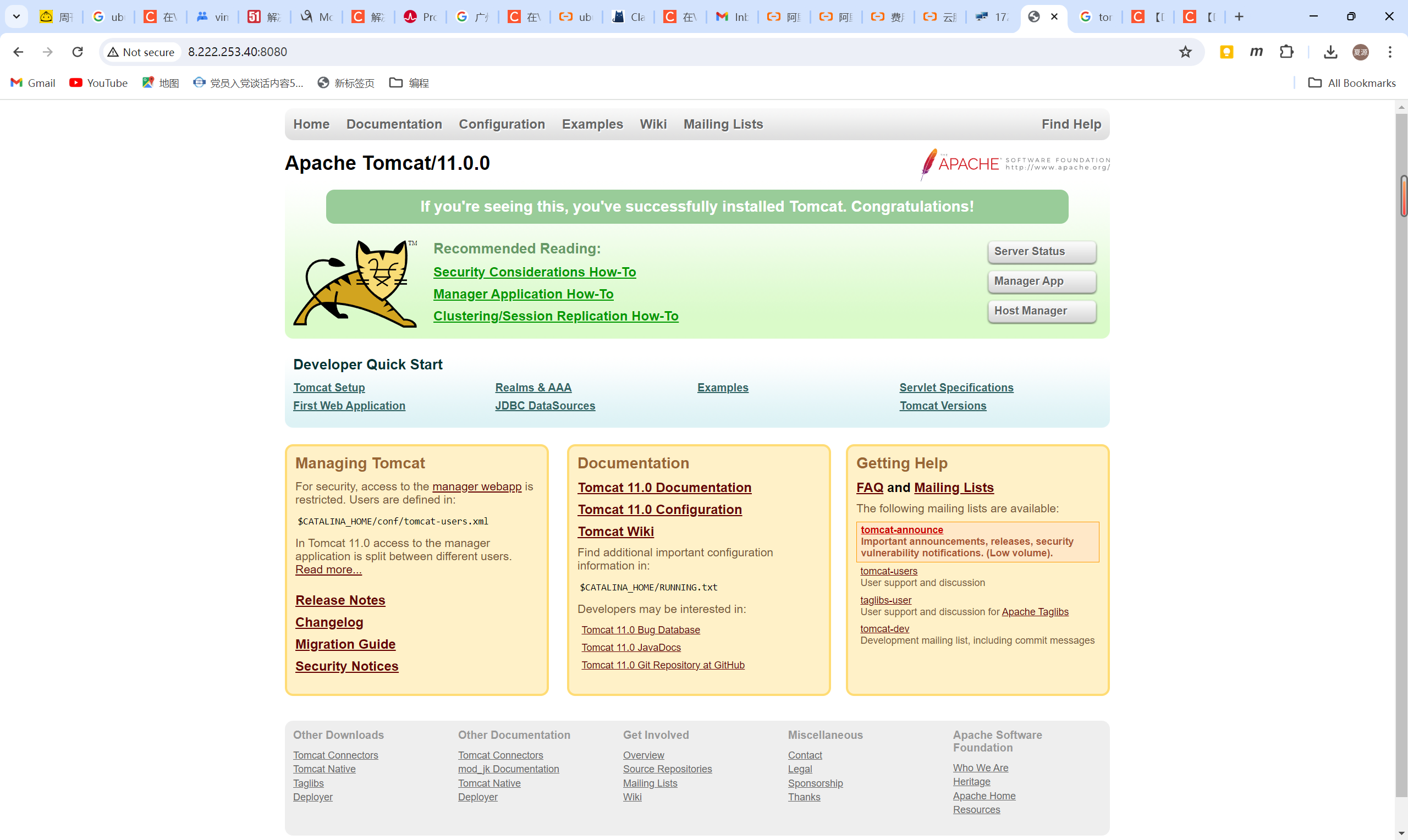
发现是空的，这就解释了为什么没办法访问 Tomcat 首页了。新版的 Tomcat 的猫首页放在了 webapps.dist 里，我们只需要把 webapps 目录删除，再把 webapps.dist 重命名为 webapps 即可。

$ rm -rf webapps

$ mv webapps.dist webapps



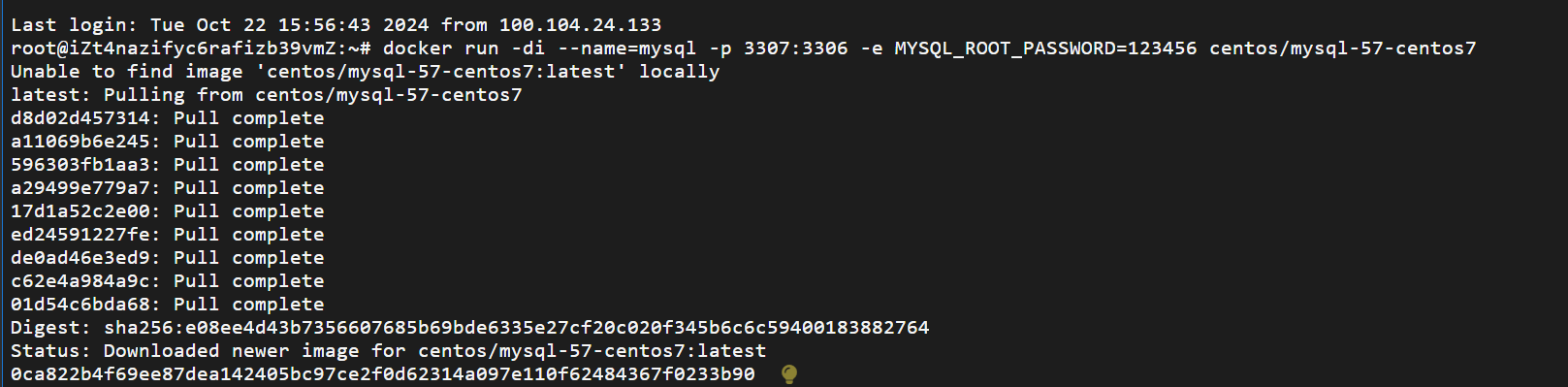
再次在浏览器中访问tomcat，看是否能成功访问 Tomcat 首页



1. 安装mysql

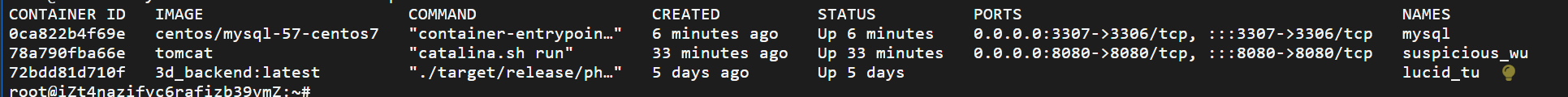
#安装mysql

docker run -di --name=mysql -p 3306:3306 -e MYSQL\_ROOT\_PASSWORD=123456 centos/mysql-57-centos7



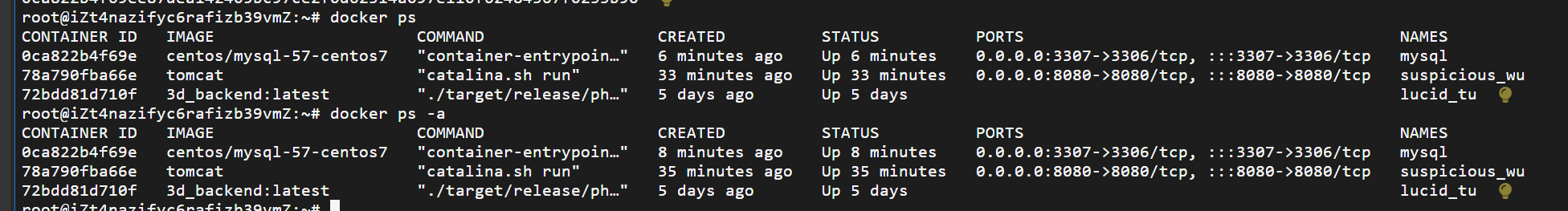
#查看运行的容器的命令：

docker ps



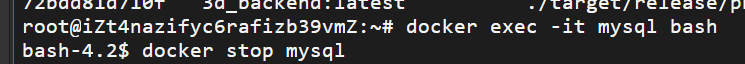
#查看曾经运行过的容器命令：

docker ps -a



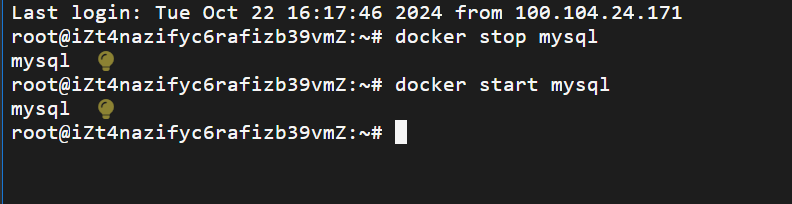
#进入运行中的MySQL容器：

docker exec -it mysql bash



#停止MySQL服务器: docker stop mysql

#重启mysql：docker start mysql



1. 使用dockerfile构建镜像

构建一个自己的tomcat镜像，使登陆界面替换成显示“hello world”。需要替换tomcat的index.jsp。

#准备index.jsp

1. 创建/usr/local/docker目录

cd /usr/local

mkdir docker

cd docker

mkdir myshop

cd myshop

1. myshop目录下创建index.jsp

vi index.jsp

里面写一句“hello world”

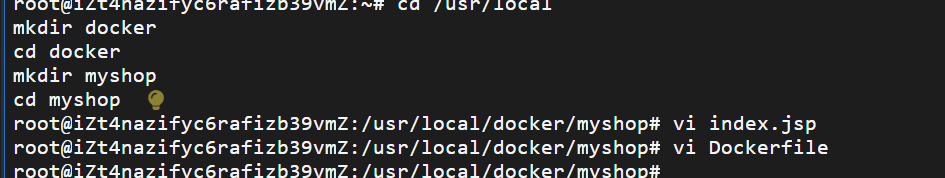
#创建dockerfile文件

myshop目录下，vi Dockerfile

#编写dockerfile文件内容为：

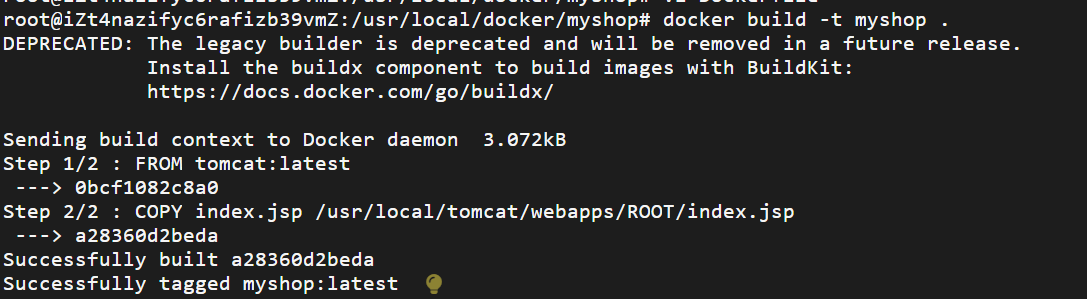
FROM tomcat:latest

COPY index.jsp /usr/local/tomcat/webapps/ROOT/index.jsp



#构建名为myshop的镜像

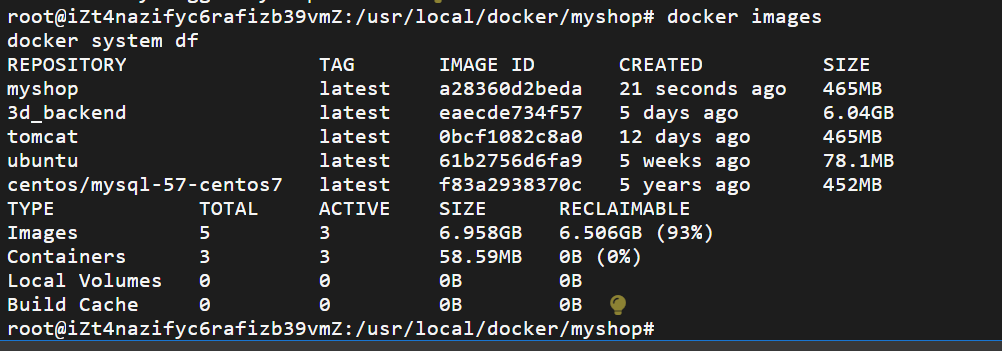
docker build -t myshop .



#查看创建好的镜像

docker images

docker system df



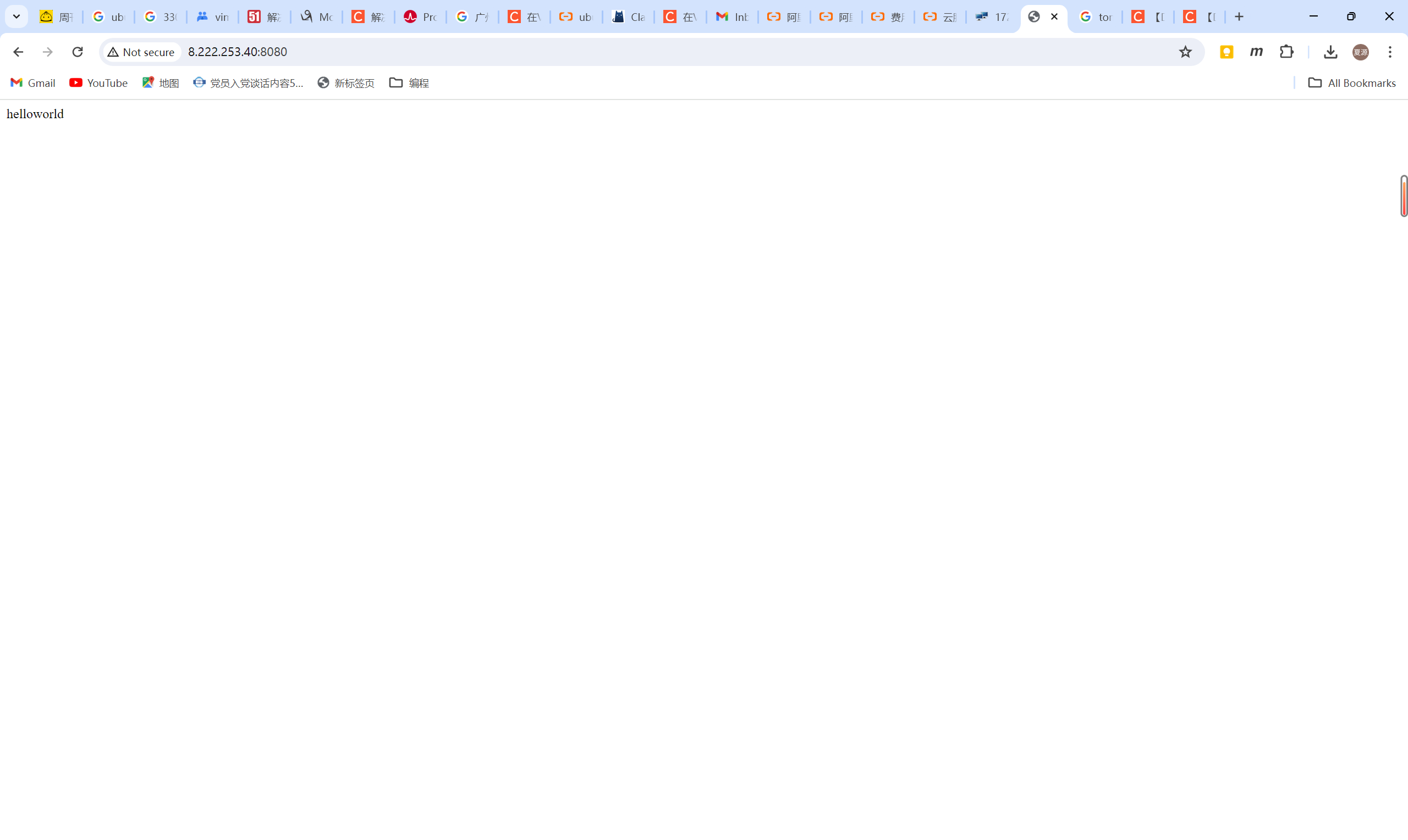
#基于新建的镜像运行容器

docker run -d -p 8080:8080 myshop

#查看容器

docker ps

#测试：通过浏览器访问TOMCAT主页，应显示“hello world”页面，截图



1. Docker compose容器编排

# Docker Compose 安装

1. Ubuntu 系统中安装 Docker Compose 的命令：

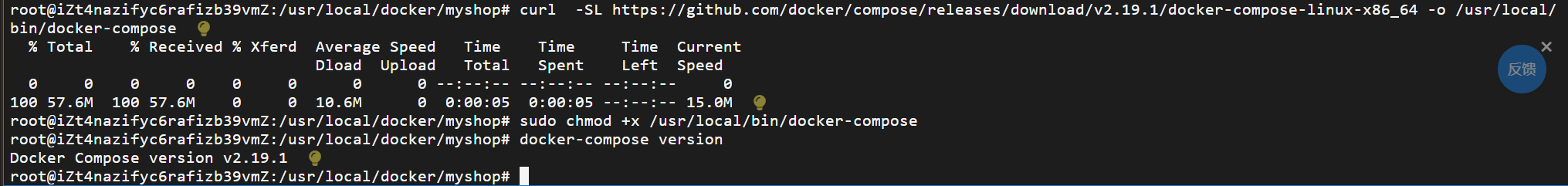
curl -SL https://github.com/docker/compose/releases/download/v2.19.1/docker-compose-linux-x86\_64 -o /usr/local/bin/docker-compose

1. 给 `docker-compose` 添加执行的权限：

sudo chmod +x /usr/local/bin/docker-compose

1. 查看 `docker-compose` 的版本

docker-compose version



# Docker Compose 使用前置

1. 修改IP

编辑 vi /etc/netplan/00-installer-config.yaml 配置文件，修改内容如下（192.168.221.136.xxx改为自己地址）：

network:

ethernets:

ens33:

addresses: [192.168.221.136/24]

routes:

- to: default

via: 192.168.221.2

nameservers:

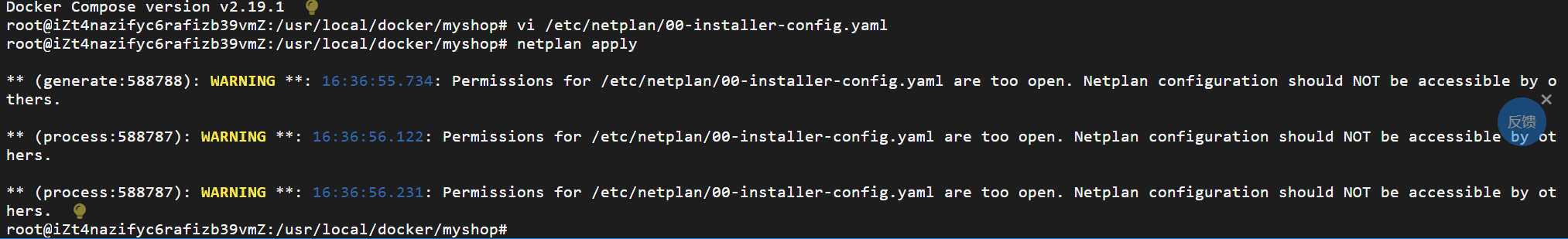
addresses: [192.168.221.2]

dhcp4: no

version: 2

使网络生效：

netplan apply

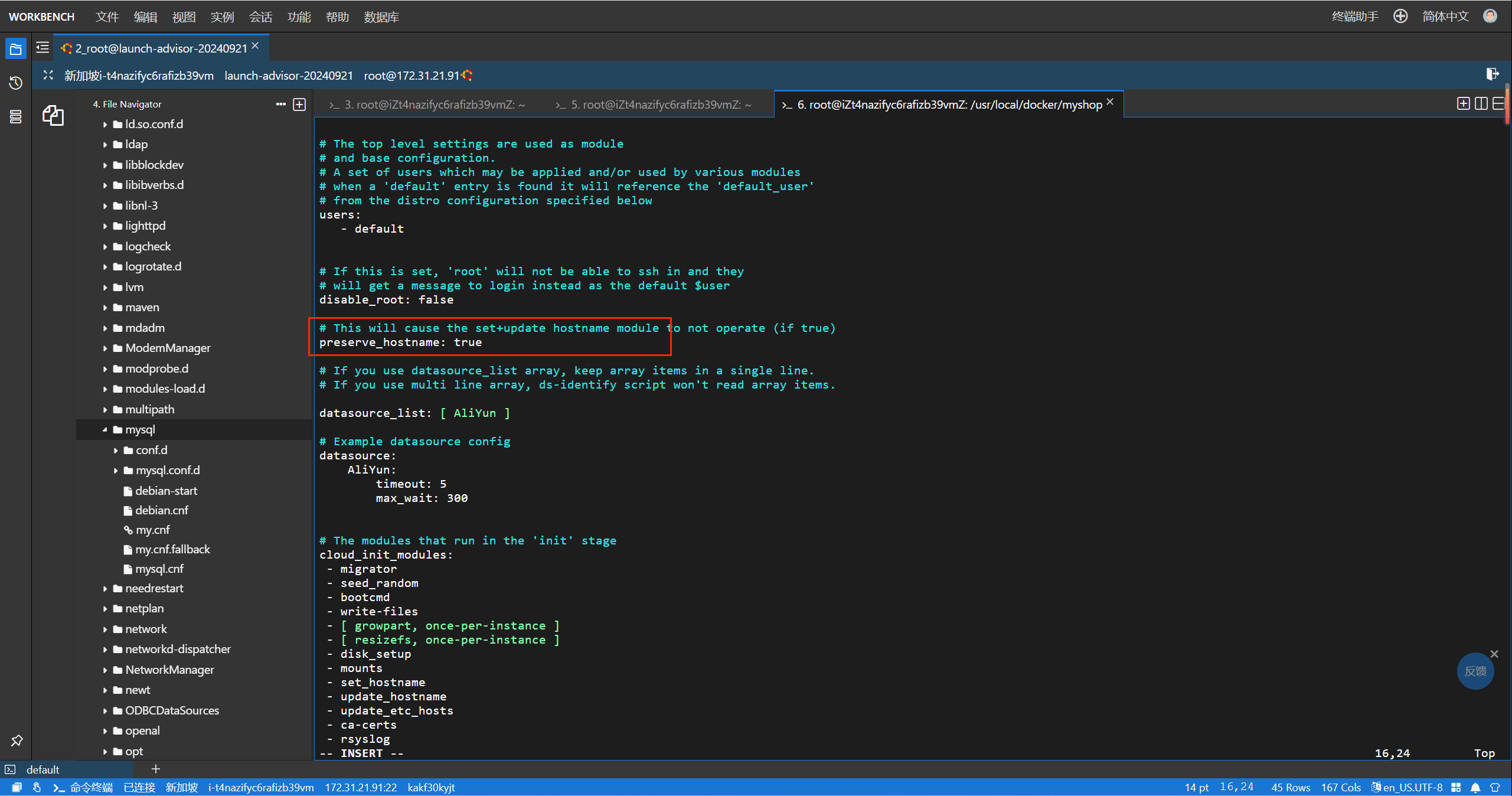


2）修改主机名：

编辑 vi /etc/cloud/cloud.cfg ，防止重启后主机名还原

//该配置默认为false，修改为true即可

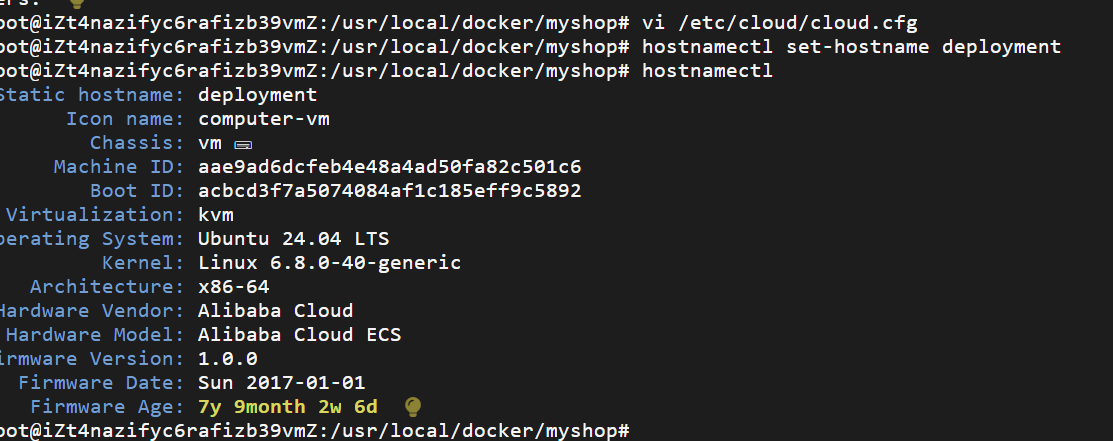
preserve\_hostname: true



//修改主机名（deployment可自定义）

hostnamectl set-hostname deployment

hostnamectl可查看主机名

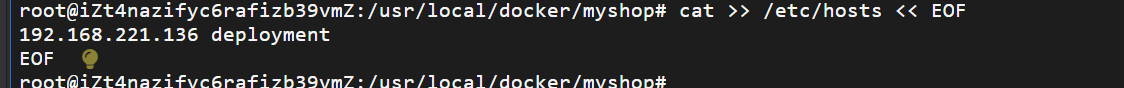


//配置hosts

cat >> /etc/hosts << EOF

192.168.221.136 deployment

EOF

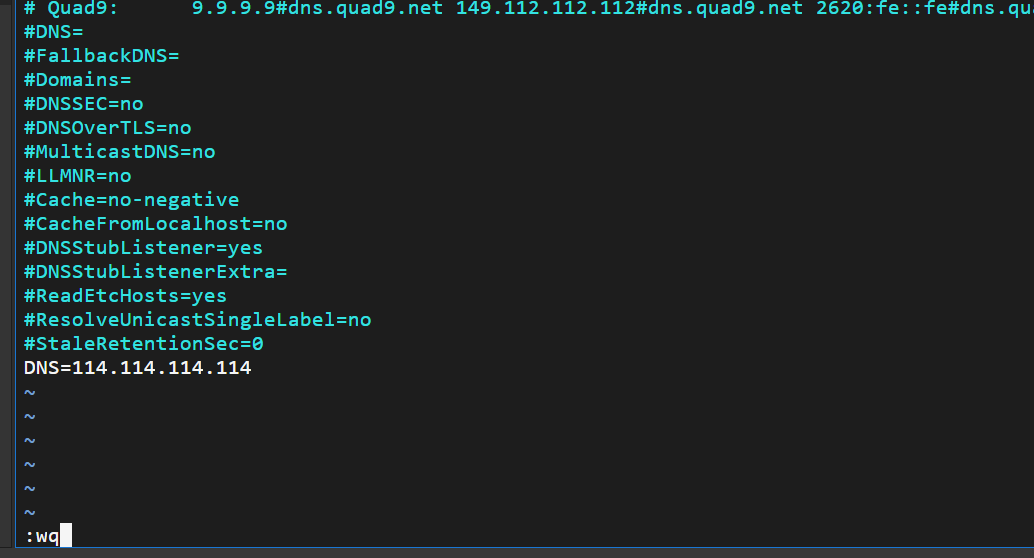


#修改DNS

取消DNS行注释，并增加DNS，配置如下，修改后重启计算机

vi /etc/systemd/resolved.conf

DNS=114.114.114.114



#修改完DNS,需要重启  
reboot

#Docker Compose 的使用--启动tomcat容器

使用DockerCompose运行tomcat容器

1）进入/usr/local/docker目录

所有手动安装都需在此目录

cd /usr/local/docker

2）创建tomcat目录

mkdir tomcat

3）在 tomcat目录下创建docker-compose.yml配置文件，内容如下：

version: '3.1'

services:

tomcat:

restart: always

image: tomcat

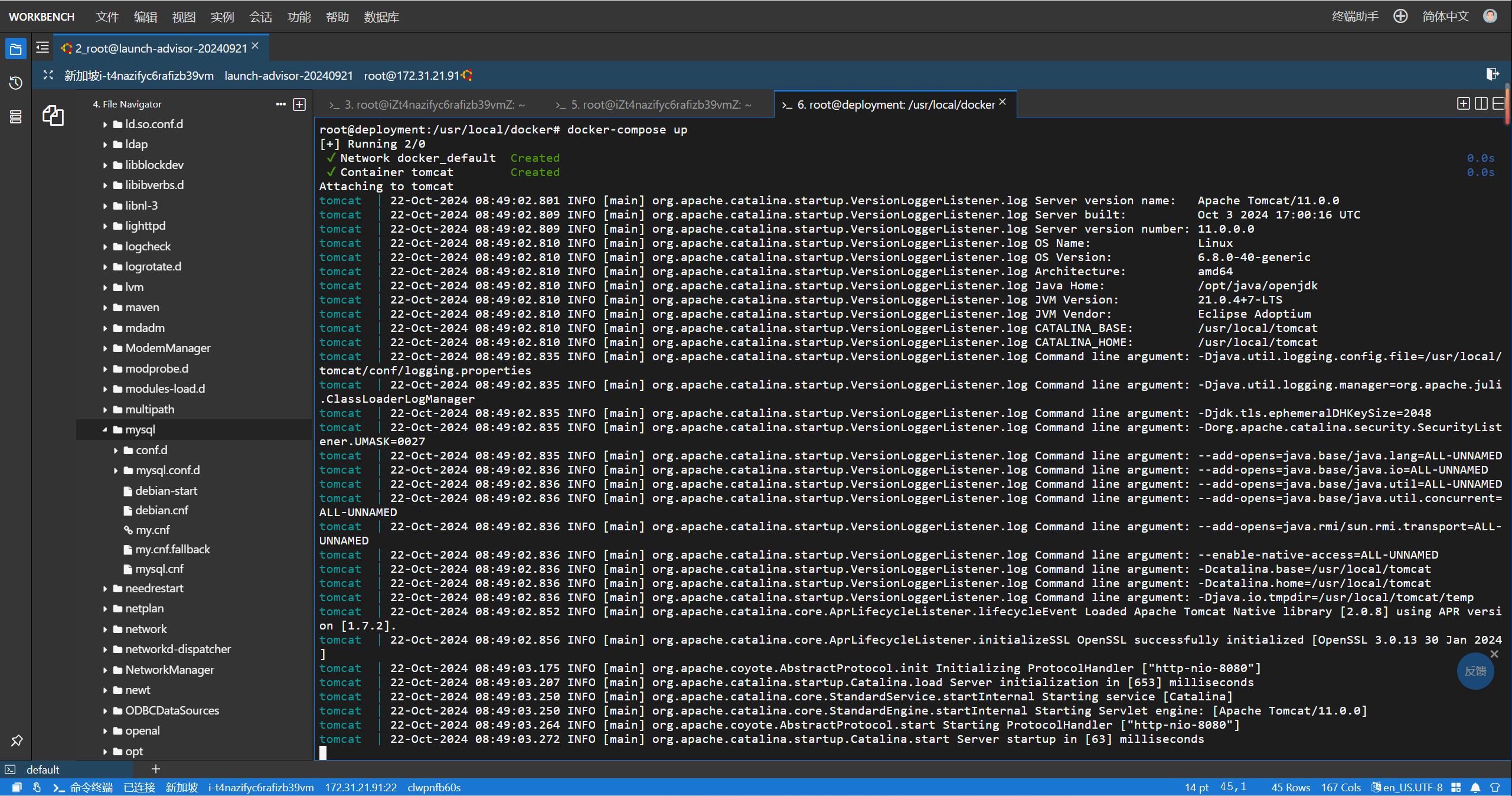
container\_name: tomcat

ports:

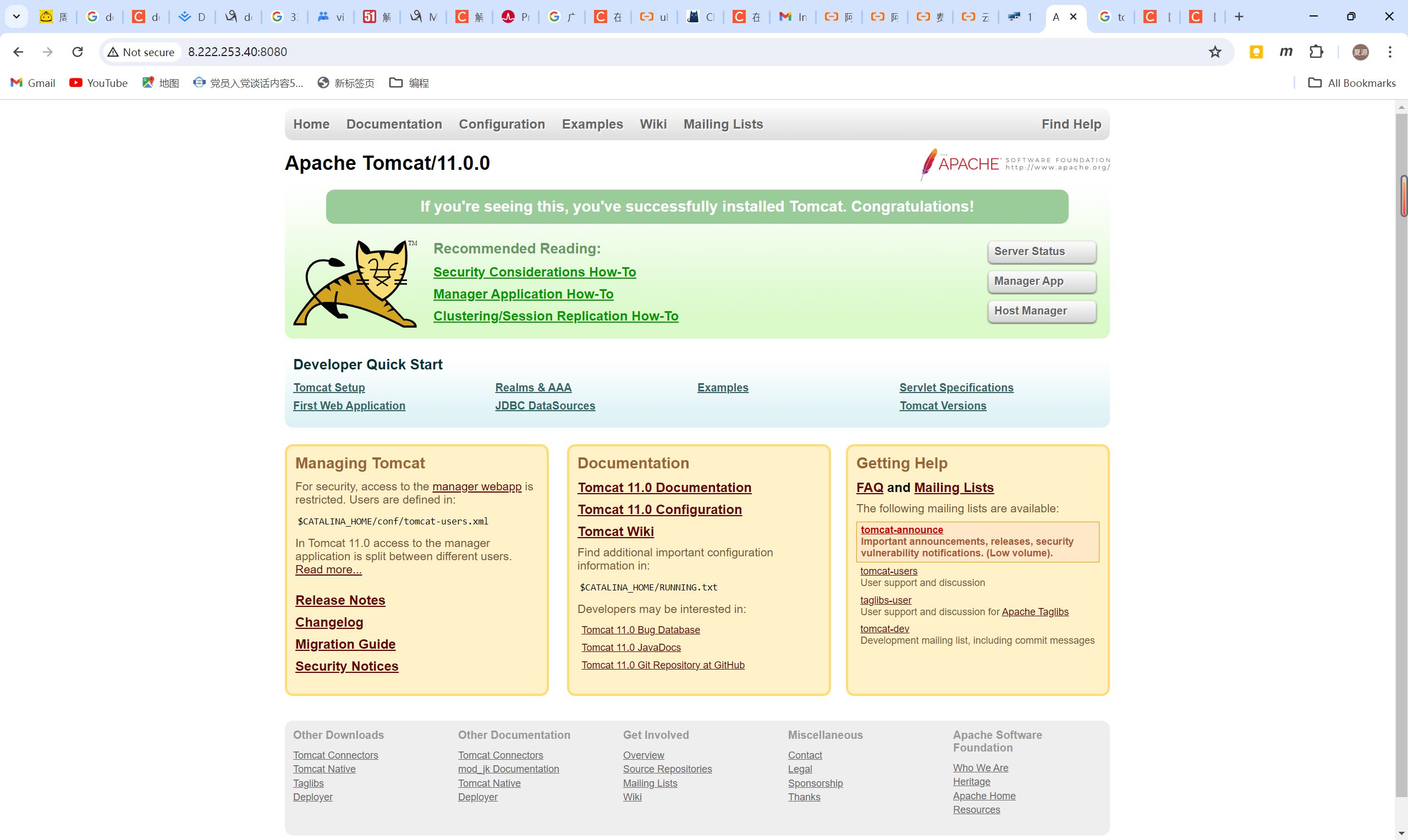
- 8080:8080

4）启动容器，测试tomcat界面

docker-compose up

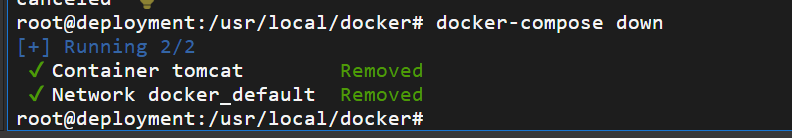


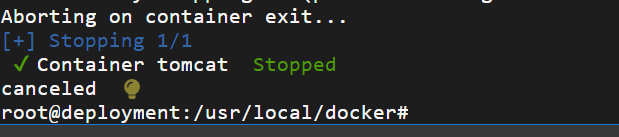
按“三．安装 Tomcat”的步骤修改tomcat的webapps目录，使tomcat界面正常显示。

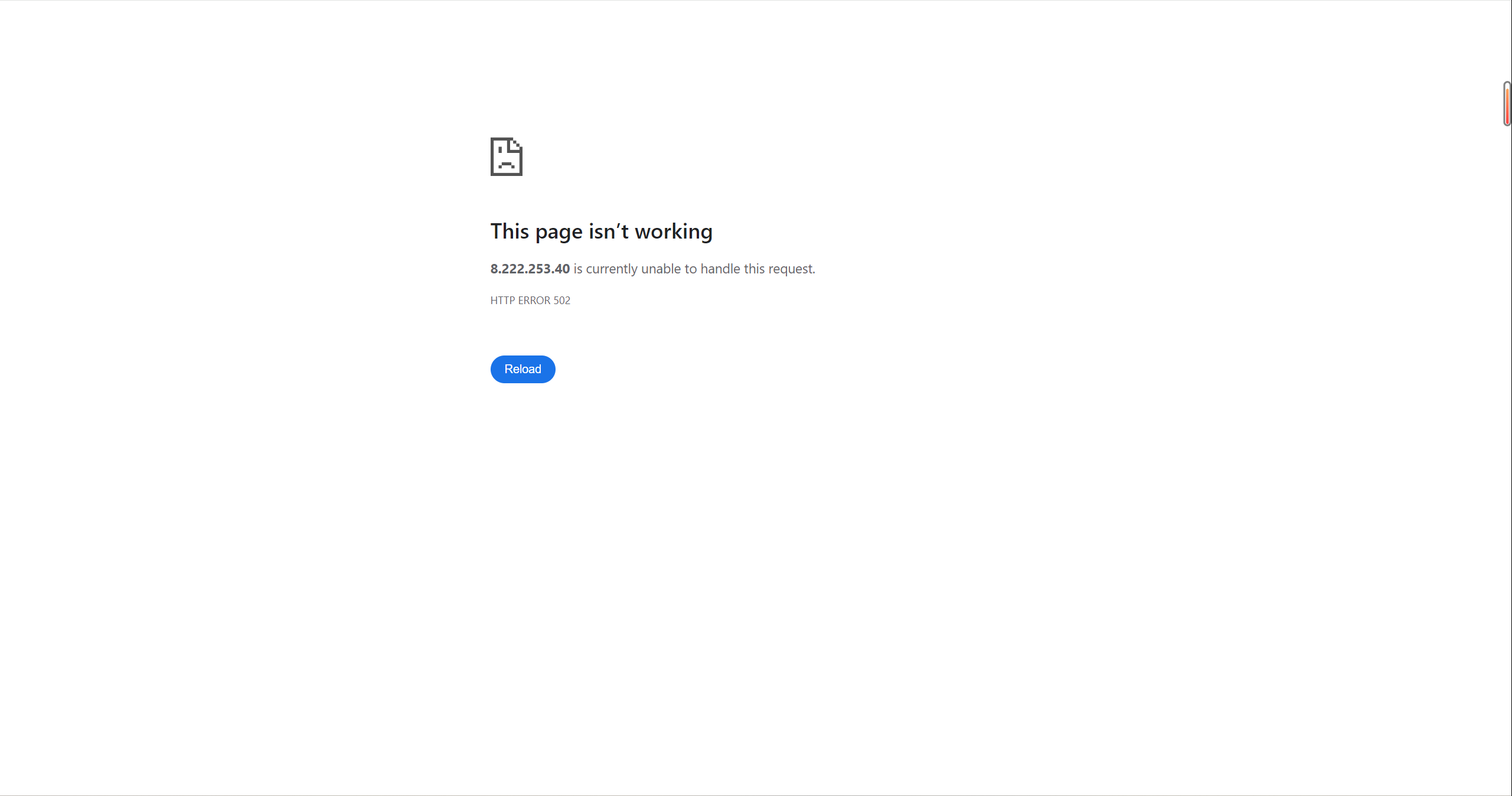


5）停止并移除容器，测试tomcat界面

docker-compose down

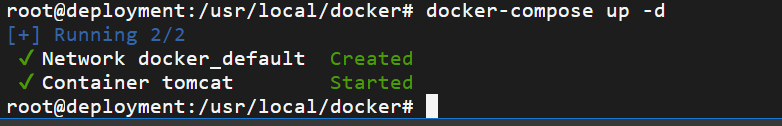


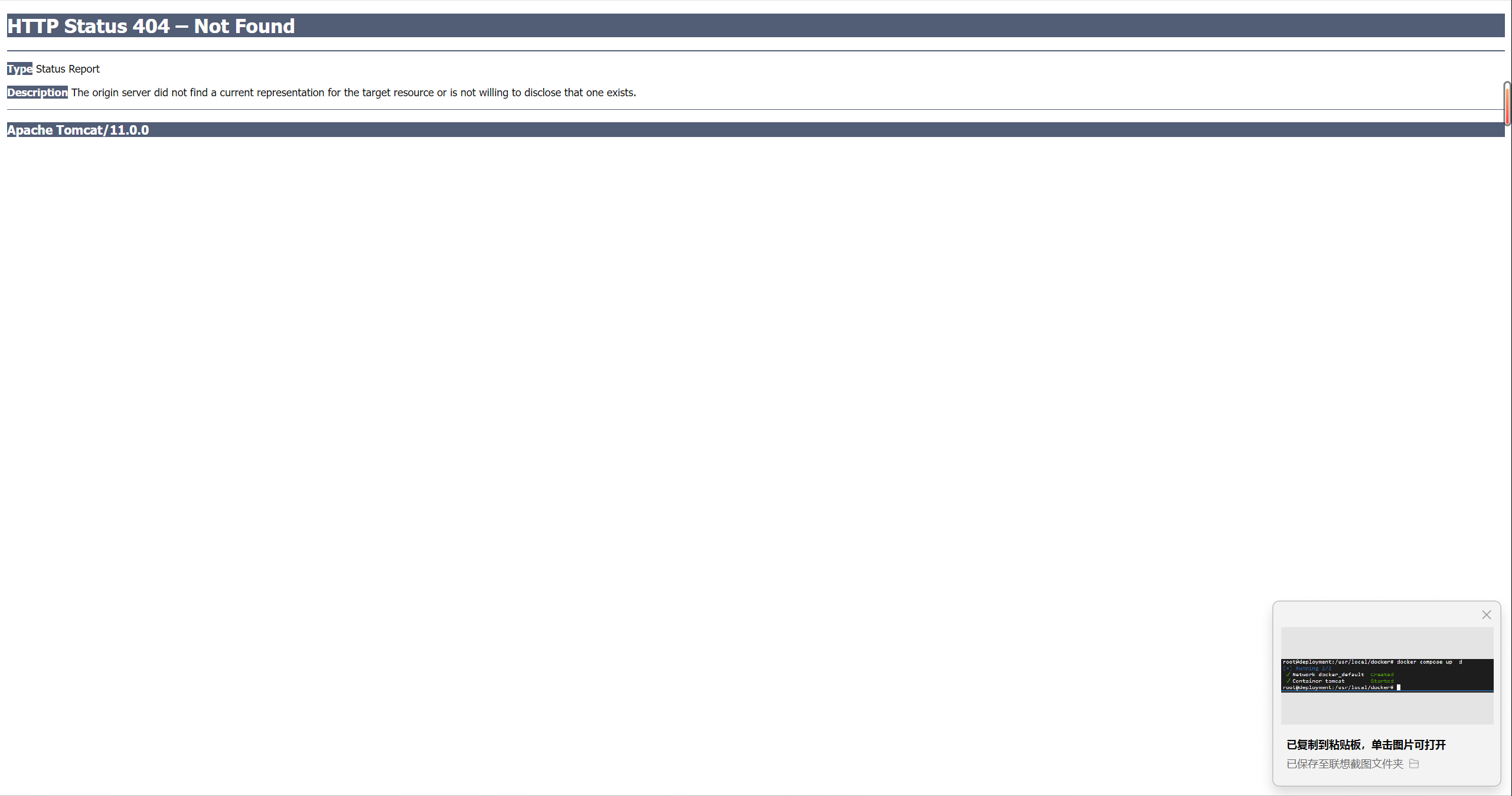




1. 守护态运行容器docker-compose up -d, 测试tomcat界面

从以上步骤发现，tomcat修改webapps，重启容器后修改失效，为了使修改的数据持久化，需要增加数据卷。





1. 增加数据卷，持久化应用中的数据

在/usr/local/docker下创建目录/tomcat/webapps/ROOT，并在ROOT目录下创建文档index.jsp，编辑index.jsp内容为“hello docker-compose！”

8）在文件docker-compose.yml中添加volumes。

vi docker-compose.yml

version: '3.1'

services:

tomcat:

restart: always

image: tomcat

container\_name: tomcat

ports:

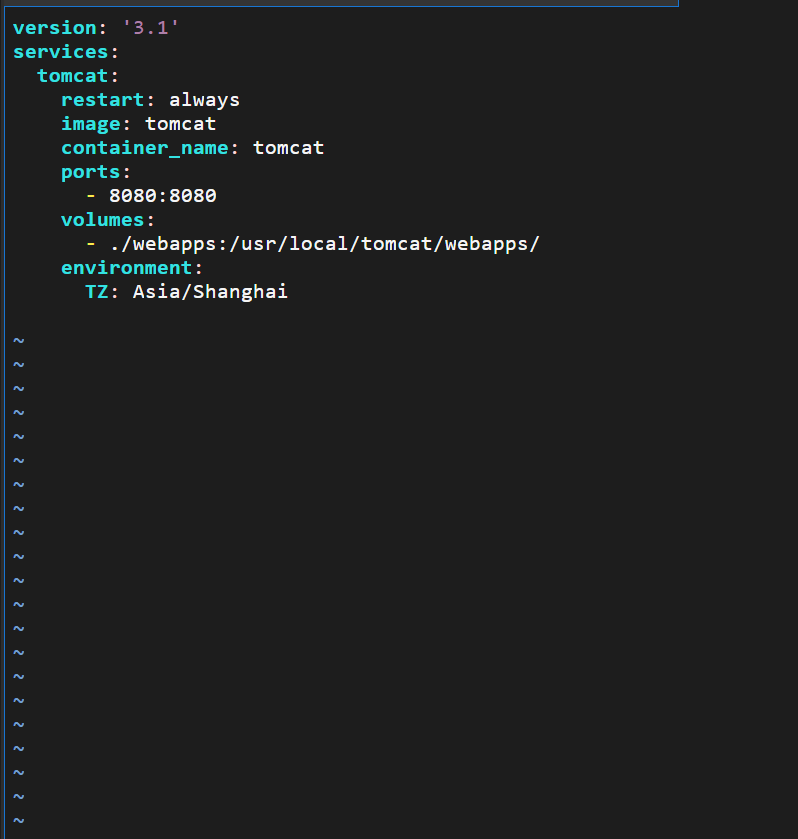
- 8080:8080

volumes:

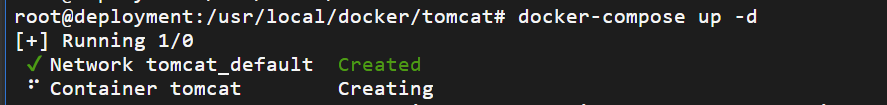
- ./webapps:/usr/local/tomcat/webapps/

environment:

TZ: Asia/Shanghai



9）守护态运行容器docker-compose up -d, 观察并记录此时tomcat界面







#Docker Compose 的使用--启动mysql容器

cd /usr/local/docker

1）创建mysql目录

mkdir mysql

1. 创建docker-compose.yml配置文件，内容如下：

vi /usr/local/docker/mysql/docker-compose.yml

version: '3.1'

services:

db:

restart: always

image: mysql

container\_name: mysql

ports:

- 3306:3306

environment:

MYSQL\_ROOT\_PASSWORD: 123456

command:

--default-authentication-plugin=mysql\_native\_password

--character-set-server=utf8mb4

--collation-server=utf8mb4\_general\_ci

--explicit\_defaults\_for\_timestamp=true

--lower\_case\_table\_names=1

volumes:

- ./data:/var/lib/mysql

#mysql的web客户端

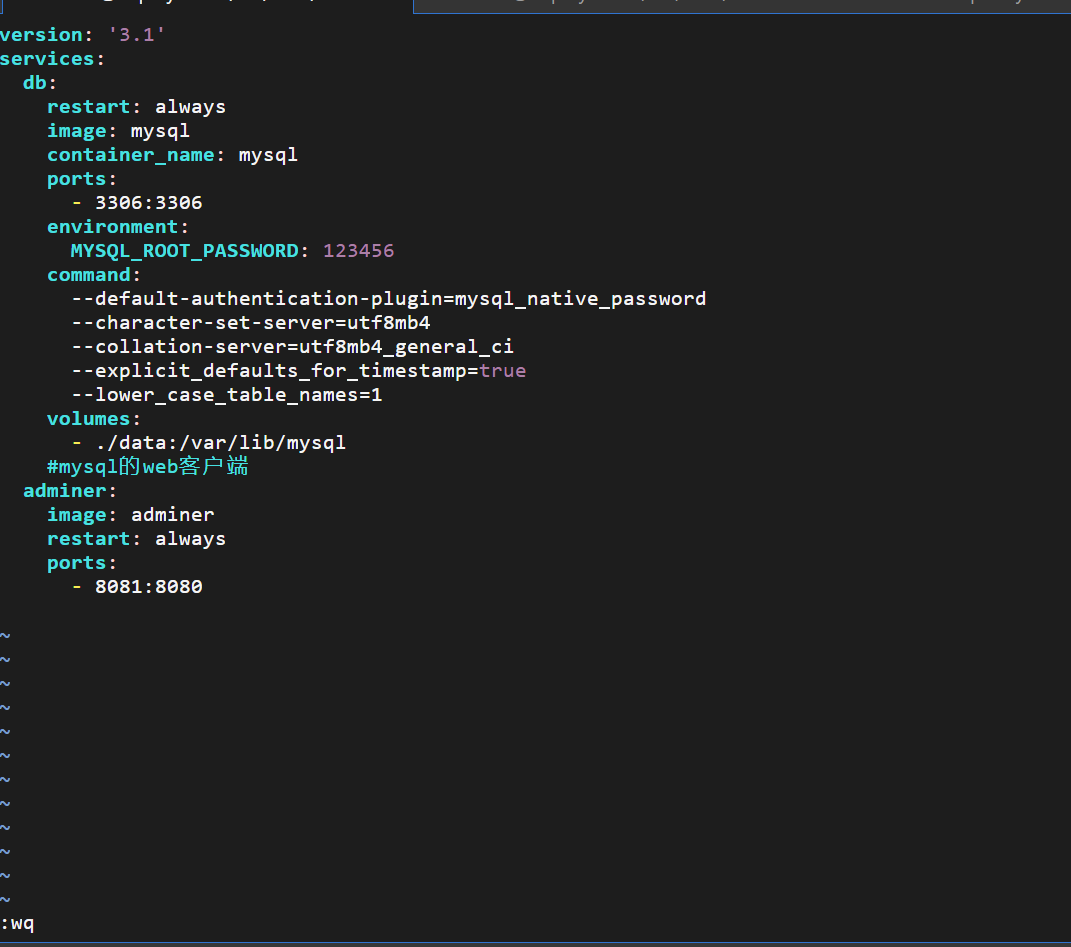
adminer:

image: adminer

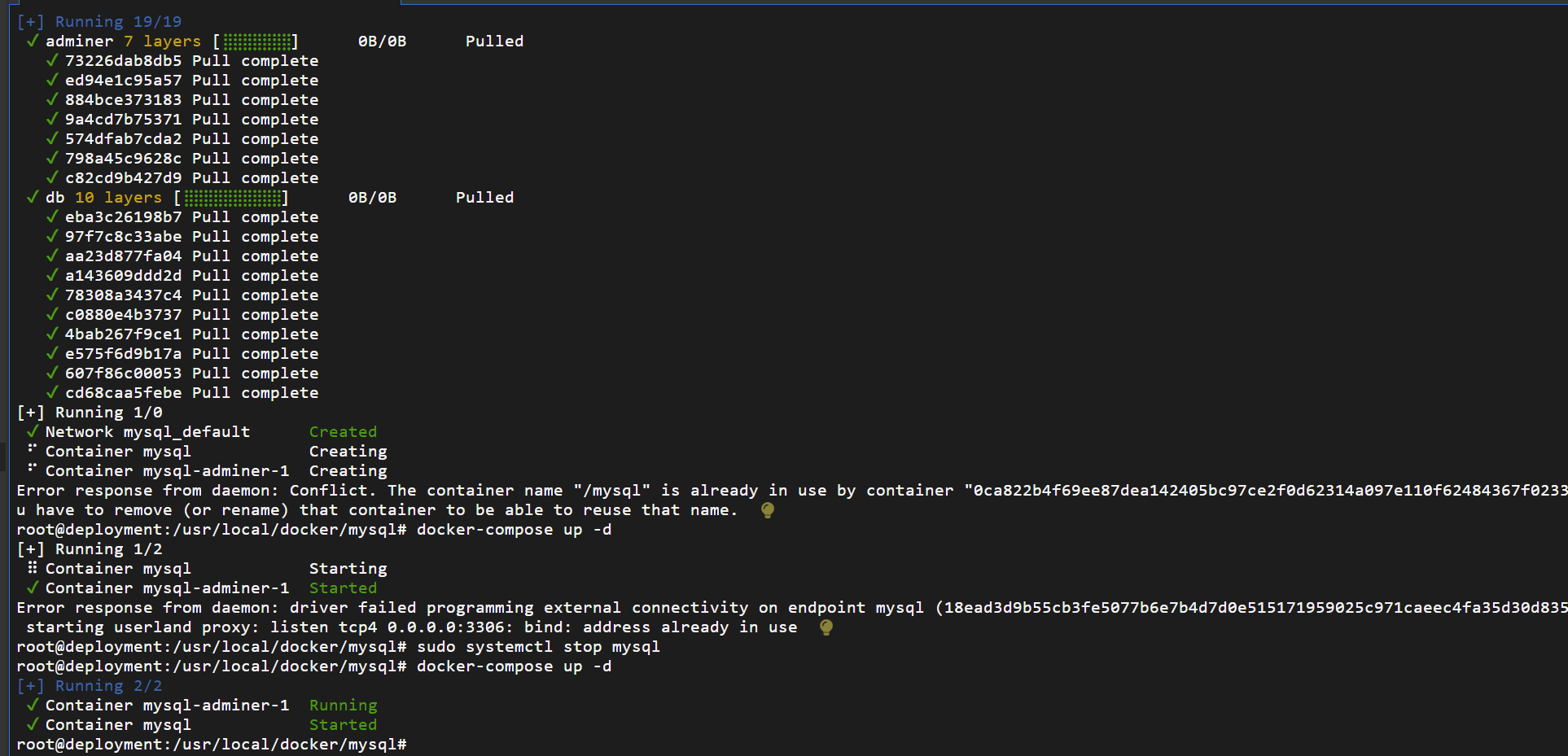
restart: always

ports:

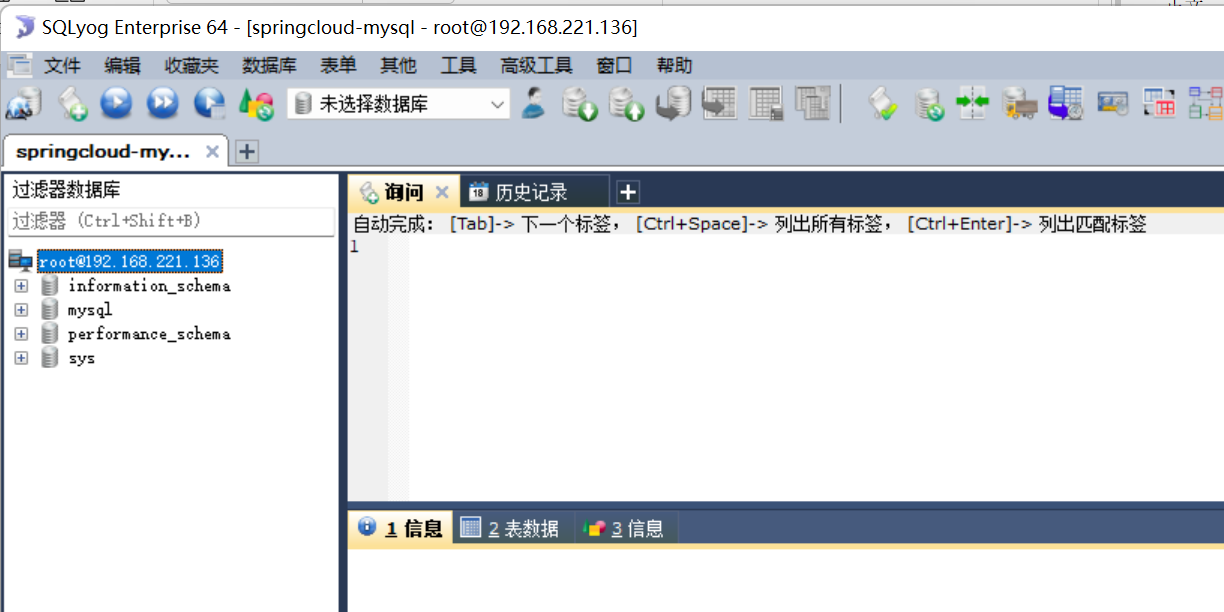
- 8081:8080

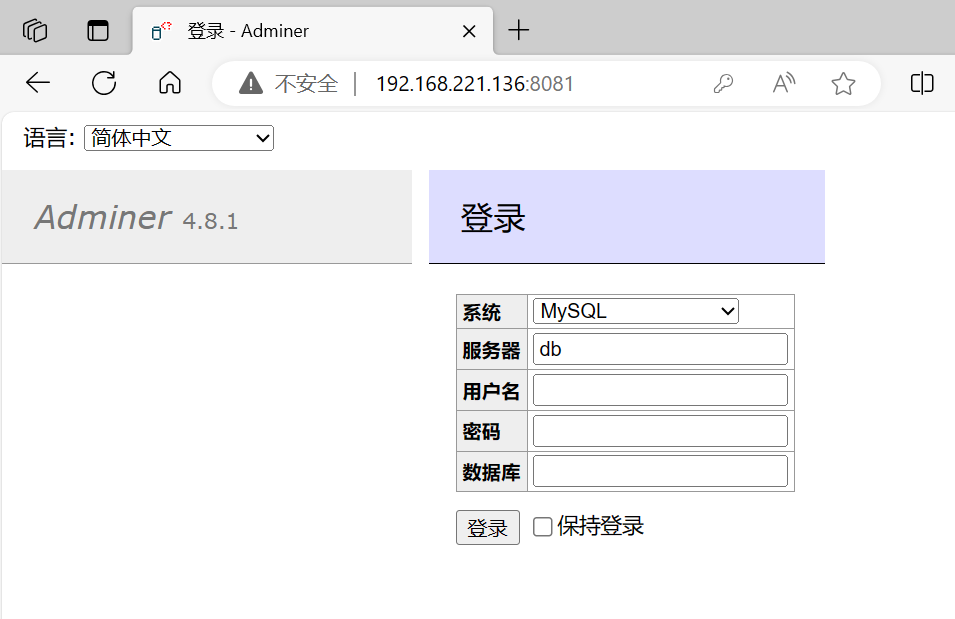


3）守护态运行容器docker-compose up -d



4）客户端连接数据库







1. Docker Compose部署应用程序
2. IDEA中创建一个简单项目“hello-spring”，该项目为父工程，在pom.xml中添加以下依赖：

<!-- SpringBoot项目必须导入的parent依赖-->

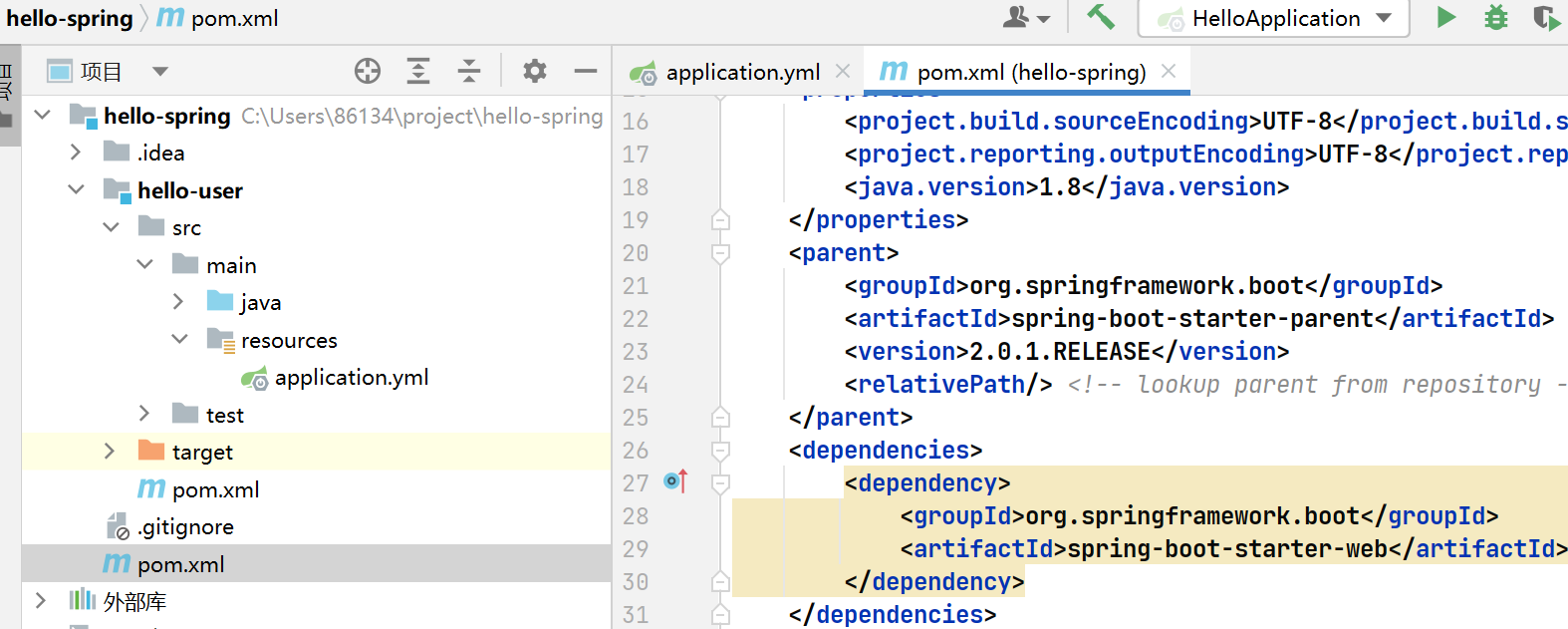
<parent>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-parent</artifactId>  
 <version>2.0.1.RELEASE</version>  
 <relativePath/> <!-- lookup parent from repository -->  
</parent>  
<!-- 定义常用的属性-->

<properties>  
 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  
 <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>  
 <java.version>1.8</java.version>  
</properties>  
<dependencies>

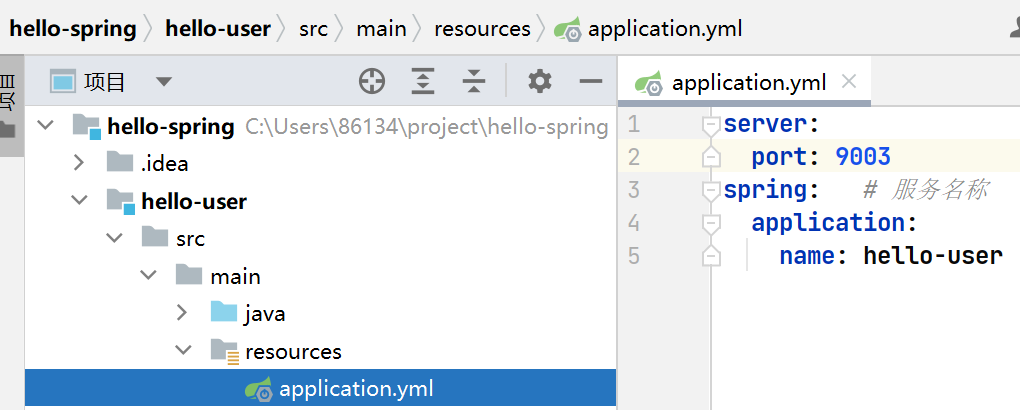
<!-- SpringBoot 对web支持， SpringMVC相关功能，json转换的功能等等-->  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-web</artifactId>  
 </dependency>  
</dependencies>

<!-- maven构建-->

<build>  
 <plugins>  
 <plugin>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-maven-plugin</artifactId>  
 </plugin>  
 </plugins>  
</build>

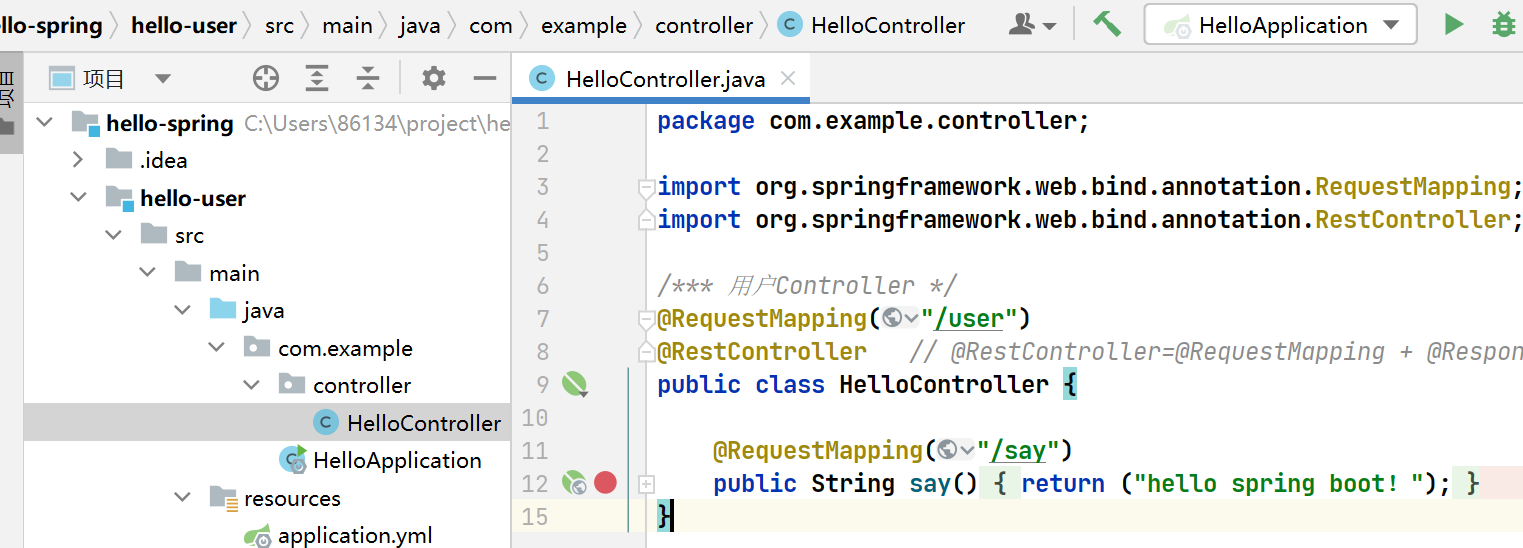


1. 创建名为 ”hello-user” 的子工程，编写配置文件application.yml （resources中）



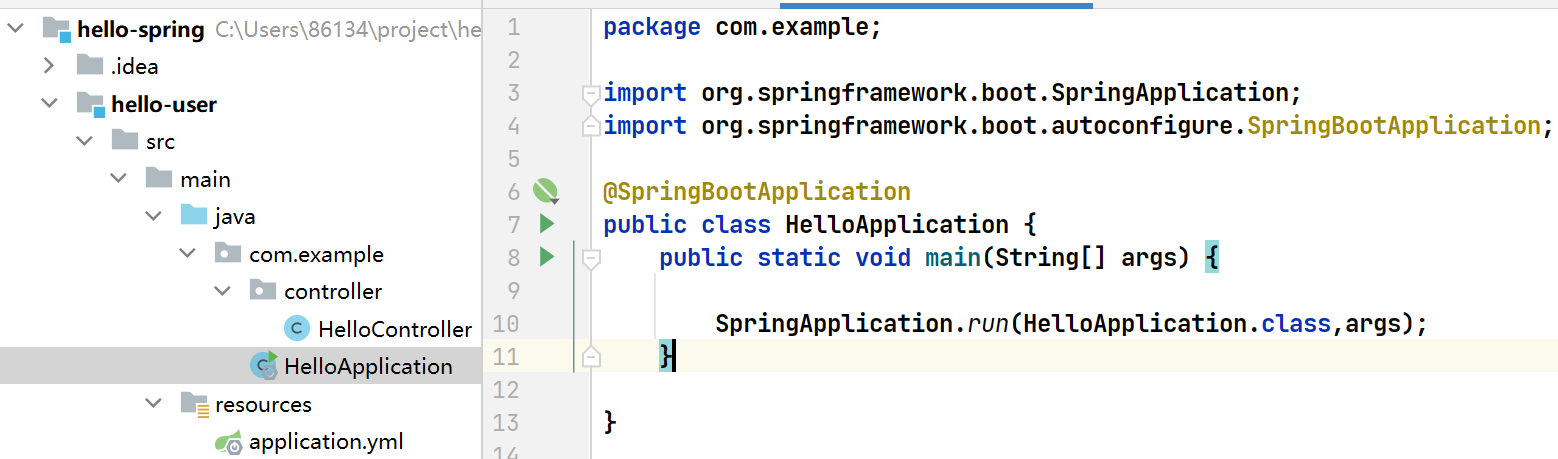
server:  
 port: 9003  
spring: # 服务名称  
 application:  
 name: hello-user

1. 编写HelloController类（xxx.controller包中），代码如下：

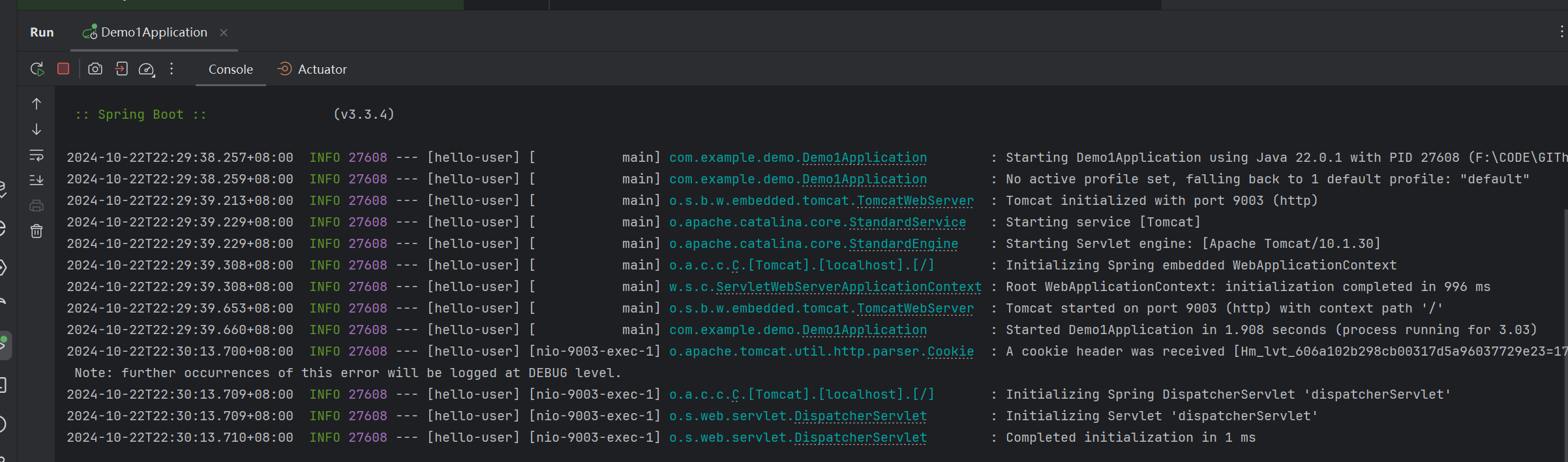


/\*\*\* 用户Controller \*/  
@RequestMapping("/user")  
@RestController // @RestController=@RequestMapping + @ResponseBody  
public class HelloController {  
 @RequestMapping("/say")  
 public String say(){  
 return ("hello spring boot！");  
 }  
}

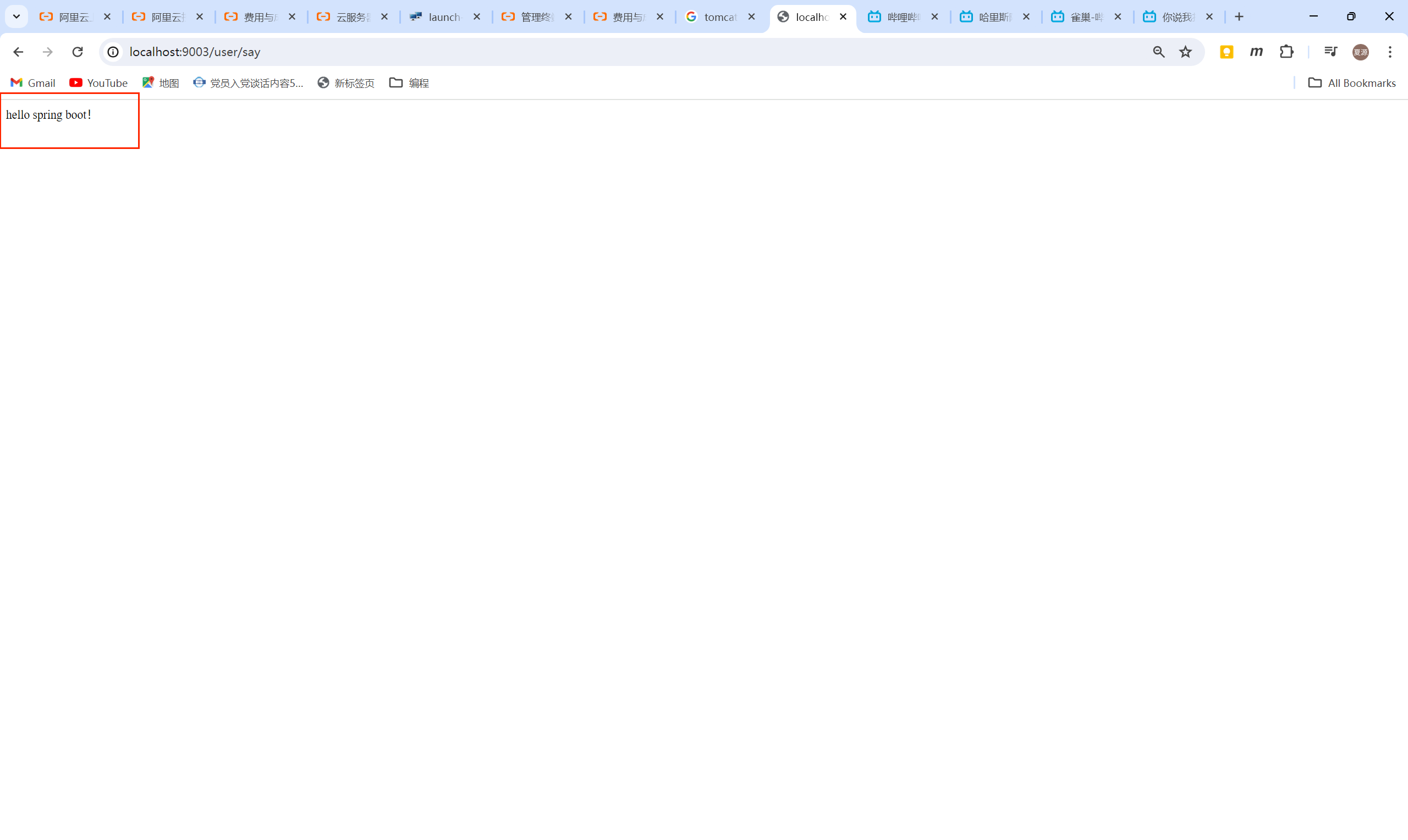
1. 编写启动类



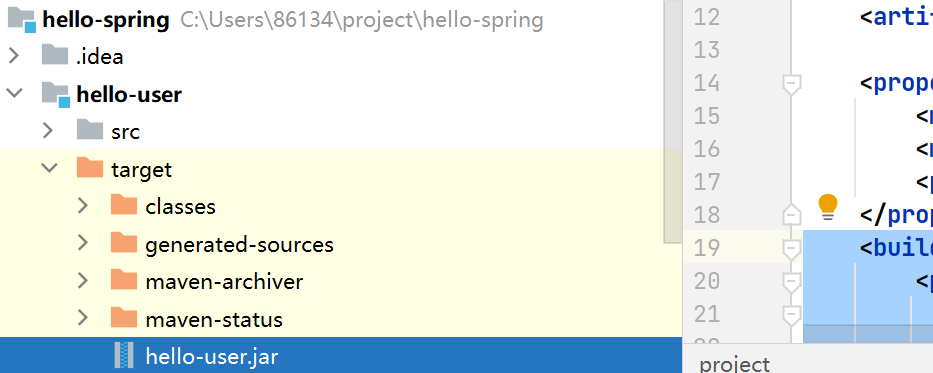
注意：启动类应为微服务中其它类的上一级目录



启动应用程序并测试，浏览器界面截图



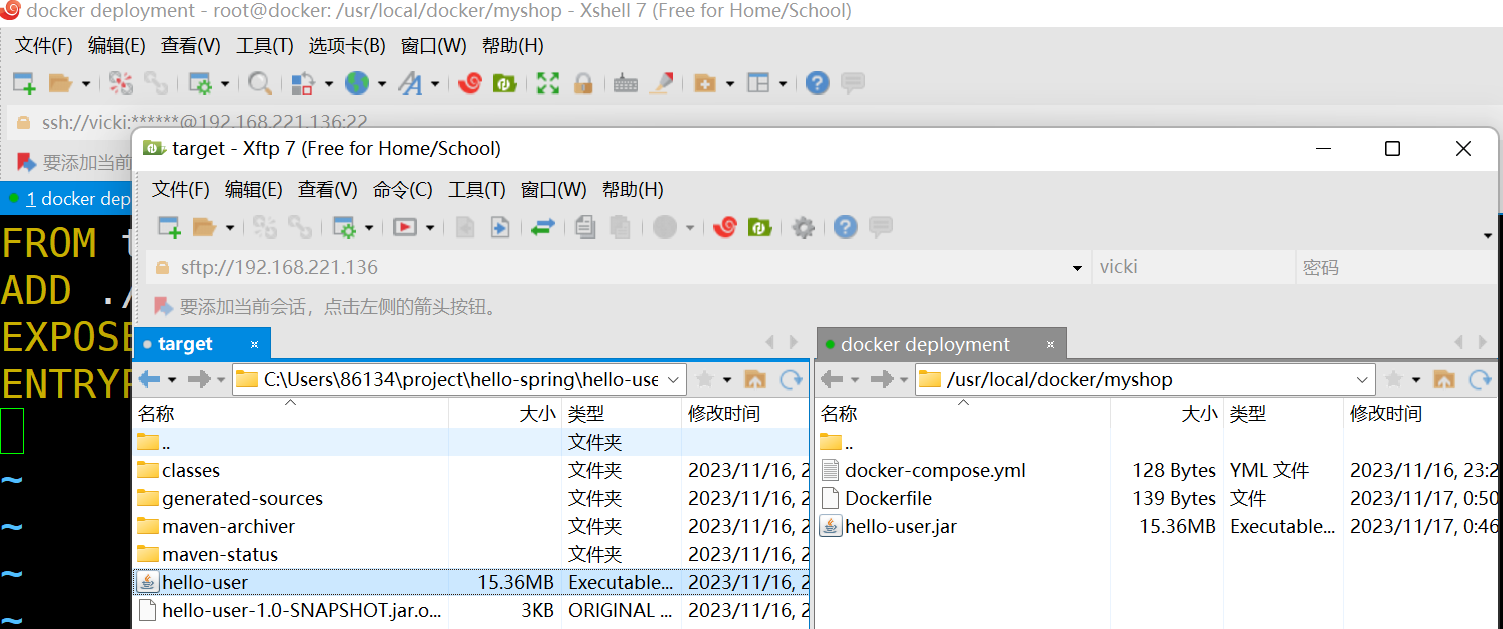
1. IDEA中项目打jar包



1. 将hello-user.jar（jar包名删掉SNAPSHOT）上传至服务器的/usr/local/docker/myshop目录

使用Xshell进行文件传输,如果无法上传，可能是权限受限，需对myshop目录授权：

chmod 777 myshop



上传之后，需要在虚拟机上修改hello-user.jar的权限

chmod 777 hello-user.jar



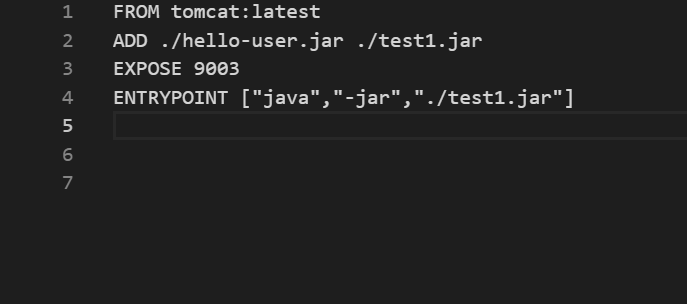
1. 编写Dockerfile（myshop 目录中）

FROM tomcat:latest

ADD ./hello-user.jar ./test1.jar

EXPOSE 9003

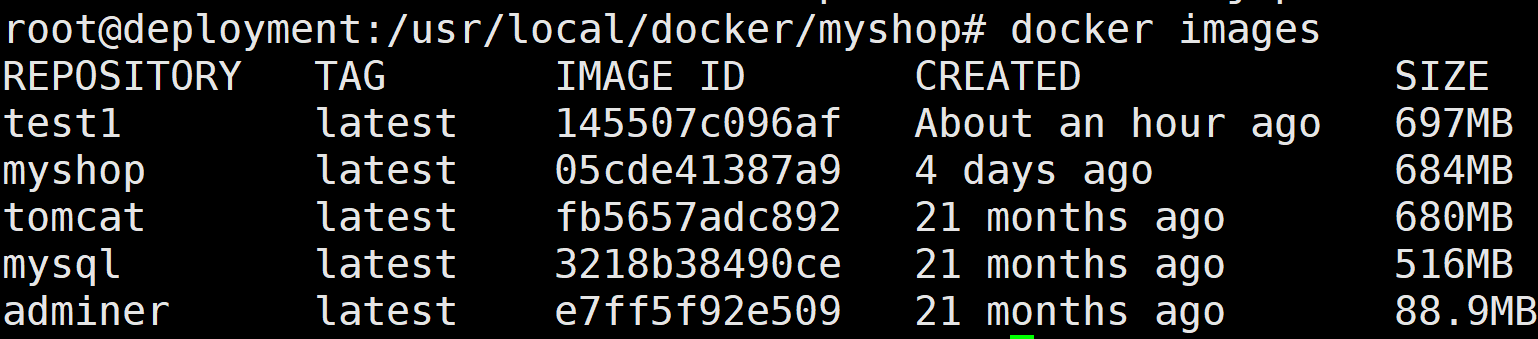
ENTRYPOINT ["java","-jar","./test1.jar"]

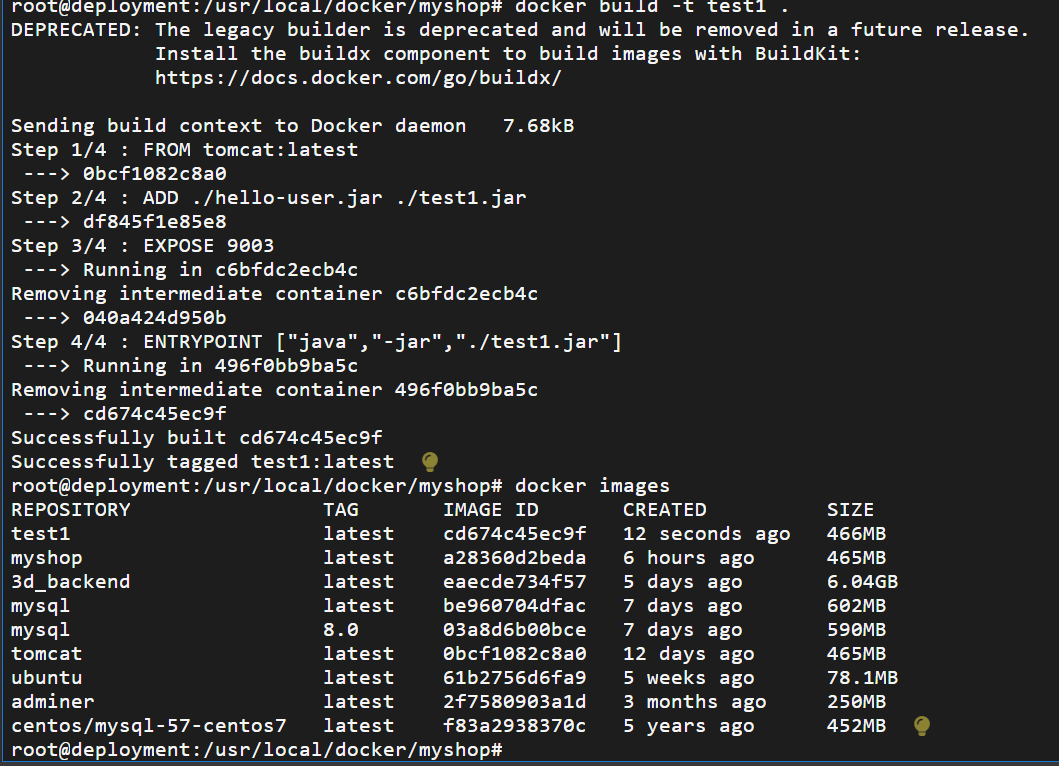


1. 按照dockerfile 构建镜像test1

docker build -t test1 .

用docker images命令查看镜像是否构建成功





1. 编写docker-compose.yml

version: '3.1'

services:

test1:

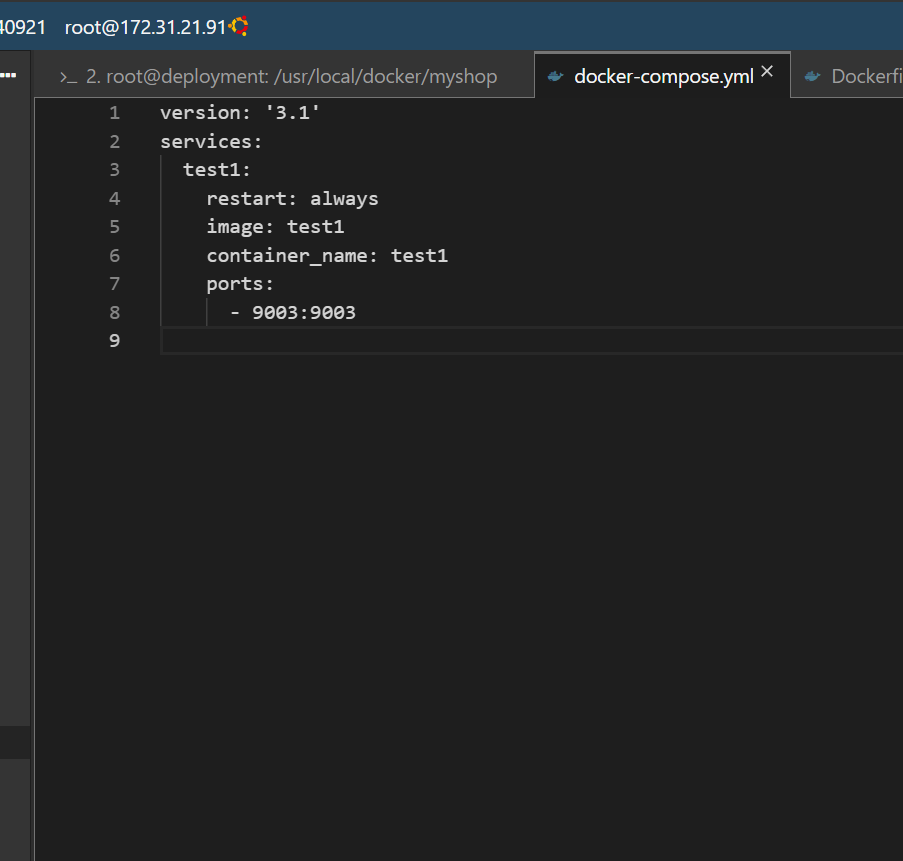
restart: always

image: test1

container\_name: test1

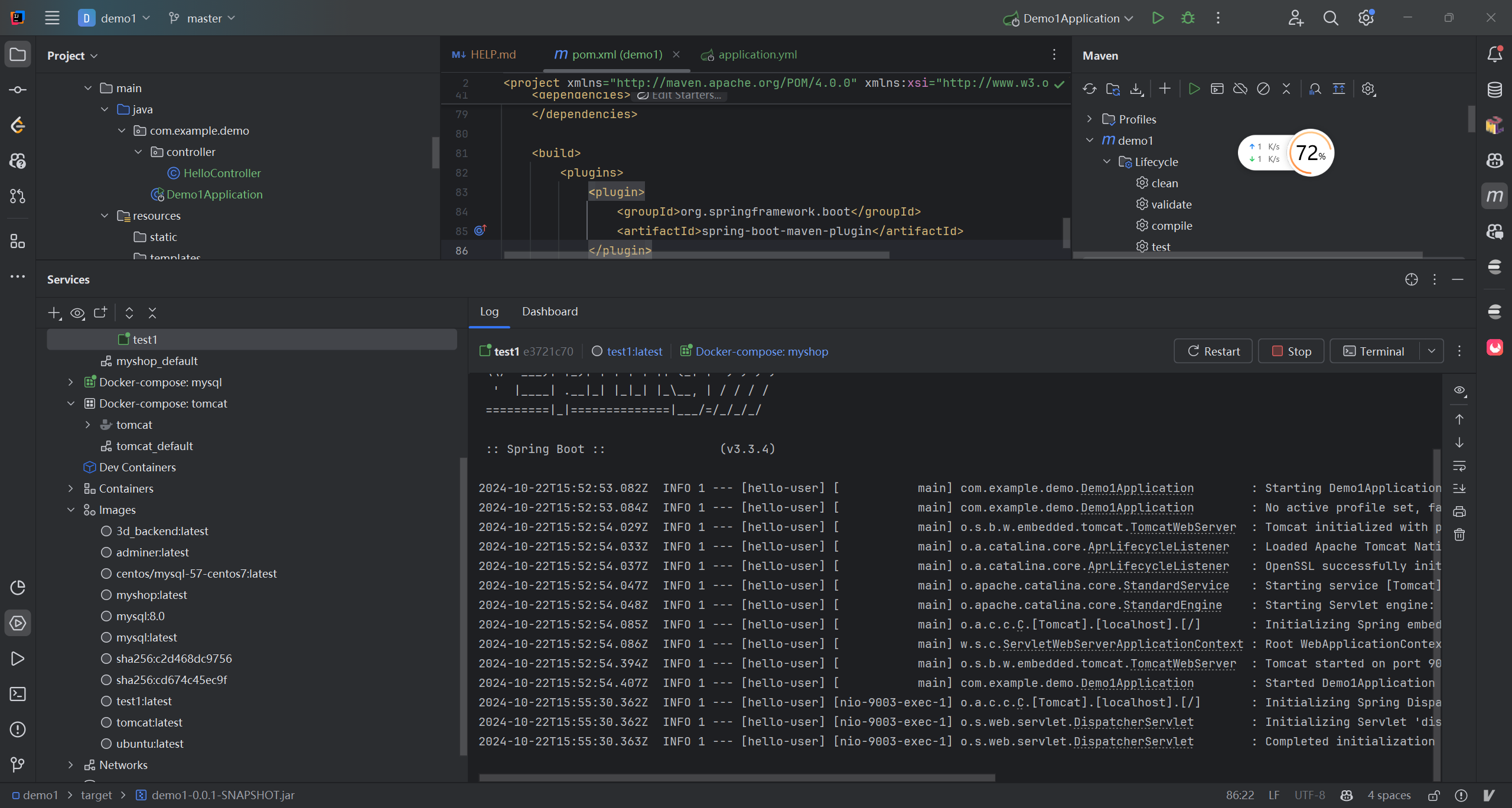
ports:

- 9003:9003



1. 启动容器

docker-compose up -d



在浏览器上测试应用程序是否运行成功



curl -SL https://pan.baidu.com/s/1W7FNYdMA2oSEu7vqvVpn4A?pwd=b6eo -o /usr/local/bin/docker-compose

