# 项目容器化部署

## 一：技术准备工作

容器技术：Docker

多容器管理技术：Docker Compose

之前分享、整理的docker学习资料（欢迎补充）：

<https://app.gitbook.com/@loosegoose/s/docker/>

## 二：panel的member-site部分容器化部署案例

在对docker和docker compose有一定了解后，

开始实践，讲解下本次对docker compose技术的使用成果。

本次实践容器化的是panel的member-site项目代码

**第一步：**

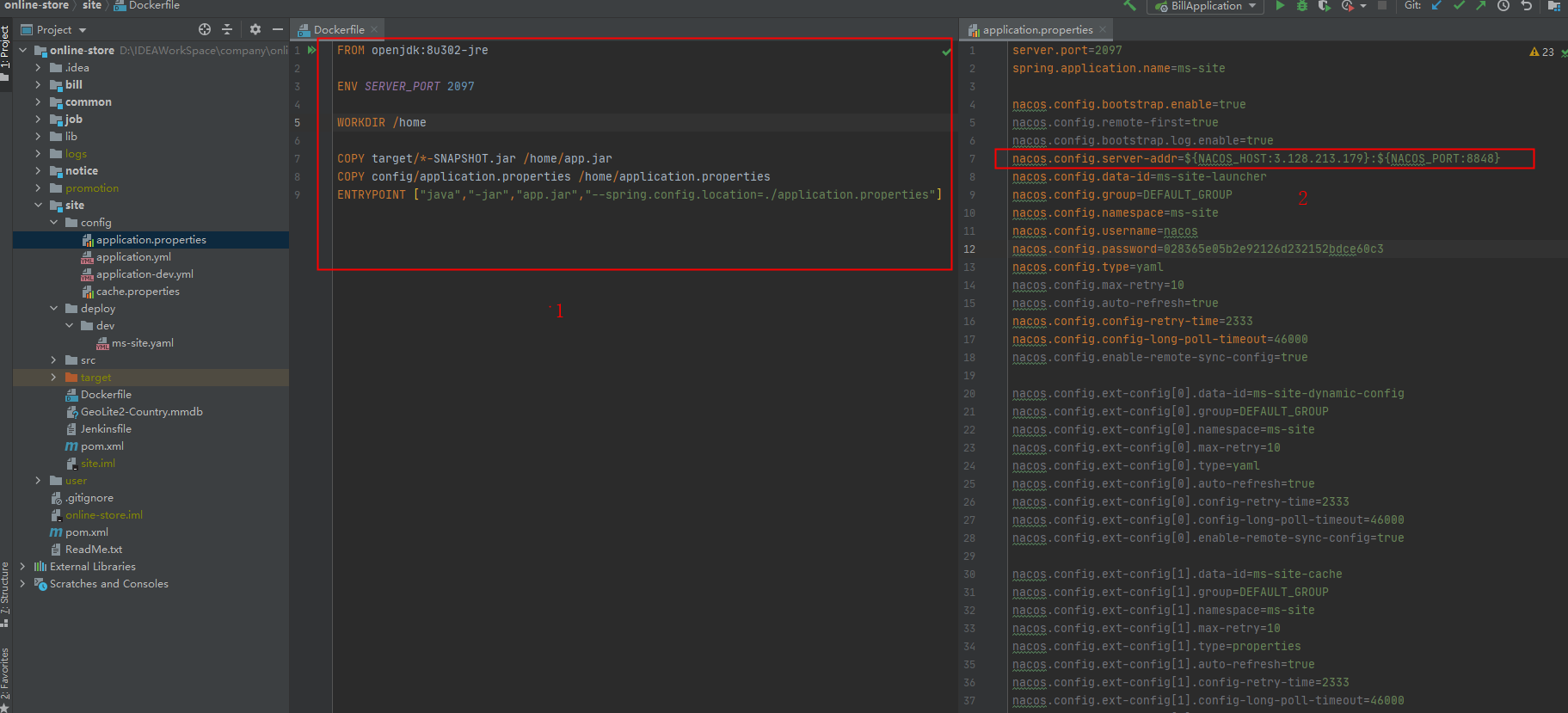
对member-site各个模块的代码先编写Dockerfile打包成docker镜像，上次到搭建、部署的docker镜像私服——Harbor（一个开源的企业级的Docker Registry管理项目。之所以没用到公共的镜像仓库dockerhub，还是出于安全和效率等方面的考虑）。

Harbor地址：

http://69.234.235.70:30002/harbor/projects

打包镜像时每个模块的Dockerfile文件编写，举例说明：

site模块:



1:编写的Dockerfile文件

#构建镜像的基础镜像

FROM openjdk:8u302-jre  
#设置环境变量 服务端口  
ENV *SERVER\_PORT* 2097  
#设置工作目录  
WORKDIR /home  
#拷贝该模块的jar包到/home/app.jar  
COPY target/\*-SNAPSHOT.jar /home/app.jar

#拷贝配置文件  
COPY config/application.properties /home/application.properties  
#执行java命令运行jar文件

ENTRYPOINT *[*"java","-jar","app.jar","--spring.config.location=./application.properties"*]*

1. 容器化的外部配置：

容器化后，环境的配置支持动态替换

**第二步：**

有了各个模块的镜像文件之后，开始编写docker-compose.yml文件，一键启动所以容器，运行起整个项目。

首先，docker连接Harbor仓库地址，使用docker login命令，输入用户名与密码，方便下面docker构建容器是能拉取到镜像文件。

docker-compose.yml文件：

#docker-compose版本，官网可查

version: '3.8'

#各种服务

services:

#mysql服务

mysql:

#使用到的镜像 mysql:5.7.30

image: mysql:5.7.30

restart: always

#构建后的容器名称

container\_name: Valor-mysql5.7.30

#docker端口映射，把docker容器内部端口映射到宿主机端口，格式：宿主机端口:容器内端口

ports:

- 3306:3306

#docker数据卷挂载，做数据持久化

volumes:

- ./data/mysql/logs:/var/log/mysql

- ./data/mysql/data:/var/lib/mysql

#环境变量：mysql的root登录密码

environment:

- MYSQL\_ROOT\_PASSWORD:Valor123

#命令设置mysql数据库的字符集

command: [

'--character-set-server=utf8mb4',

'--collation-server=utf8mb4\_unicode\_ci'

]

#nacos服务

nacos:

image: nacos/nacos-server:1.4.2

ports:

- 8848:8848

container\_name: Valor-nacos1.4.2

volumes:

- ./nacos/standalone-logs/:/home/nacos/logs

#环境变量，配置的上面那个mysql容器服务，url那里直接写服务名称，不用写ip。（如果你连的数据库是上面新建的mysql容器）

environment:

- db.url.0=jdbc:mysql://mysql:3306/nacos?Unicode=true&characterEncoding=UTF-8

- spring.datasource.platform=mysql

- MYSQL\_SERVICE\_USER=root

- MYSQL\_SERVICE\_PASSWORD=Valor123

# nacos的单机模式

- MODE=standalone

# depends\_on:nacos服务启动依赖与上面的mysql服务，写服务名

depends\_on:

- mysql

restart: always

#redis服务

redis:

image: redis:6.0.9-alpine

container\_name: Valor-redis6.0.9-alpine

ports:

- 6379:6379

restart: always

volumes:

- ./data/redis.conf:/usr/local/etc/redis/redis.conf:rw

- ./data/redis:/data:rw

#member-site的容器化内容， ms-site模块的容器服务

ms-site:

#镜像就是打包后推到Harbor仓库的镜像

image: 69.234.235.70:30002/member-site/ms-site:latest

#暴露服务端口到宿主机

ports:

- 2097:2097

#环境变量，就是上面说的环境的配置支持动态替换，这里填写的是nacos，就是代表为上面的nacos服务。也就是说ms-site容器启动后访问的地址是nacos服务地址，读取nacos服务配置信息。

environment:

- NACOS\_HOST=nacos

depends\_on:

- mysql

- nacos

#ms-bill模块服务

ms-bill:

image: 69.234.235.70:30002/member-site/ms-bill:latest

ports:

- 2095:2095

environment:

- NACOS\_HOST=nacos

depends\_on:

- mysql

- nacos

#ms-job模块服务

ms-job:

image: 69.234.235.70:30002/member-site/ms-job:latest

ports:

- 2091:2091

environment:

- NACOS\_HOST=nacos

depends\_on:

- mysql

- nacos

#ms-notice模块服务

ms-notice:

image: 69.234.235.70:30002/member-site/ms-notice:latest

ports:

- 2099:2099

environment:

- NACOS\_HOST=nacos

depends\_on:

- mysql

- nacos

#ms前端页面模块服务

web-ui:

image: 69.234.235.70:30002/member-site/web-ui:latest

#环境变量：前端访问的后端服务接口地址和端口，对应上面配置的那几个后端服务的地址和端口。

environment:

- NACOS\_HOST=nacos

- USER\_API\_HOST=192.168.80.131

- USER\_API\_PORT=2097

- SITE\_API\_HOST=192.168.80.131

- SITE\_API\_PORT=2097

- JOB\_API\_HOST=192.168.80.131

- JOB\_API\_PORT=2091

- BILL\_API\_HOST=192.168.80.131

- BILL\_API\_PORT=2095

- PAY\_API\_HOST=192.168.80.131

- PAY\_API\_PORT=2054

depends\_on:

- ms-site

- ms-bill

- ms-job

#暴露端口，容器启动，服务IP+31080即可访问前端页面

ports:

- 31080:80

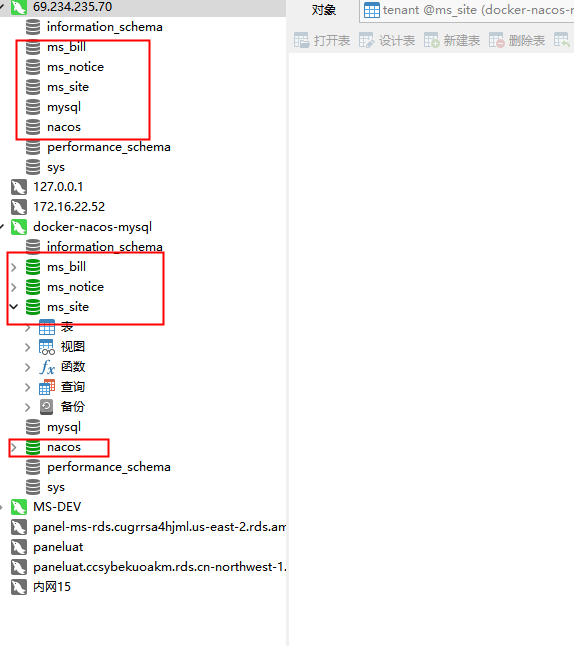
注意：书写格式严重按照yaml文件格式。

**第三步：**

在存放docker-compose.yml文件的目录下，使用docker-compose up命令，开始使用容器的构建，此时还不能访问项目，因为此时的member-site模块的所以容器都打包失败，因为在nacos上面还没有配置构建之后的mysql和redis连接地址。

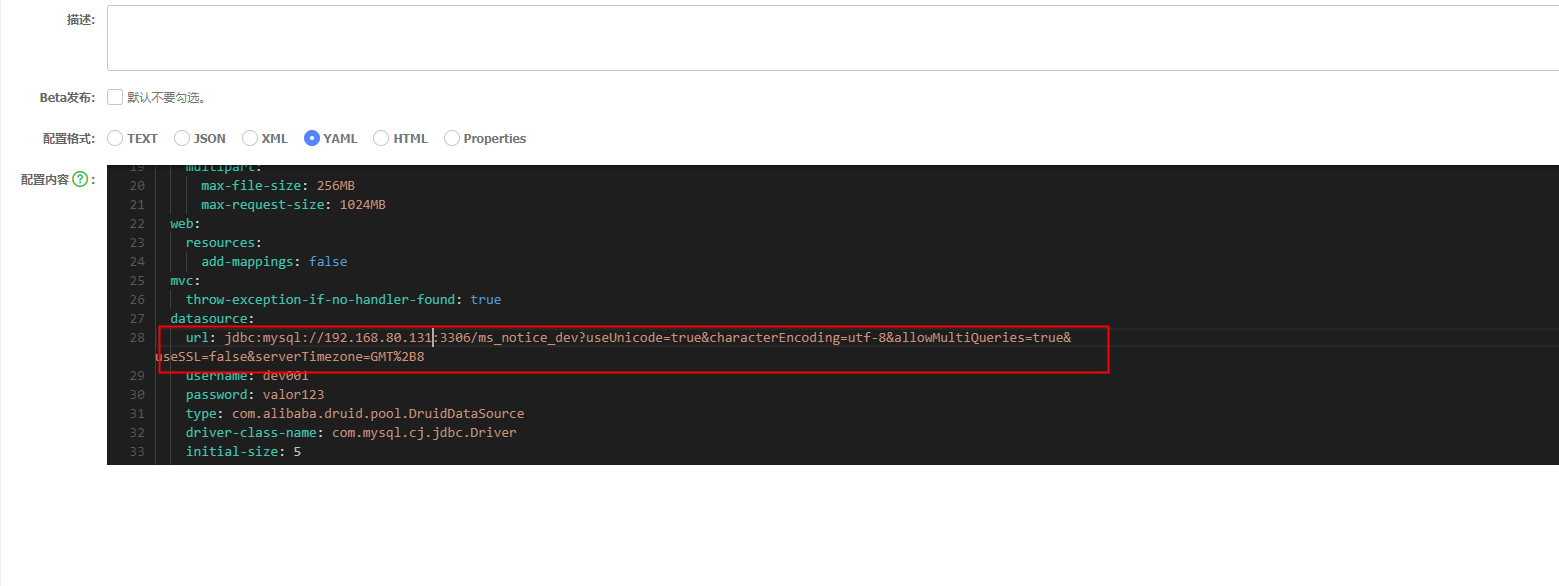
但是本次构建nacos、mysql和redis都能容器化成功，所以只需要进行配置的修改和mysql数据库的创建，再次重启构建即可。

1. 使用navicat连接构建成功的mysql容器，因为做了端口暴露，所以直接连接宿主机IP+映射port。在navicat中把需要用到的数据库拷贝到该mysql容器中。

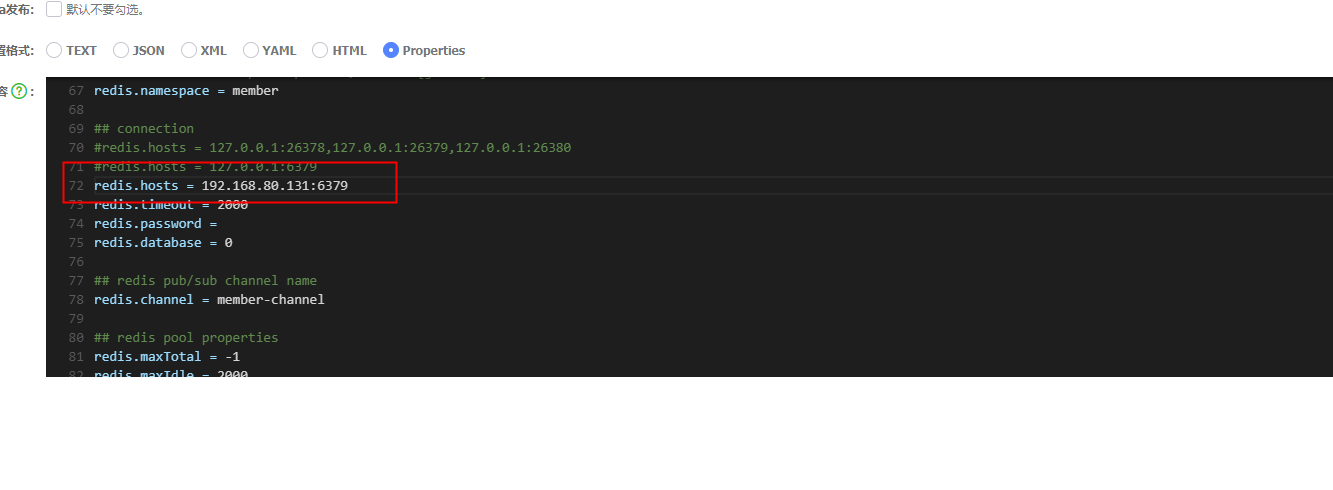


1. 浏览器登录构建成功后的nacos地址，宿主机IP+映射port。

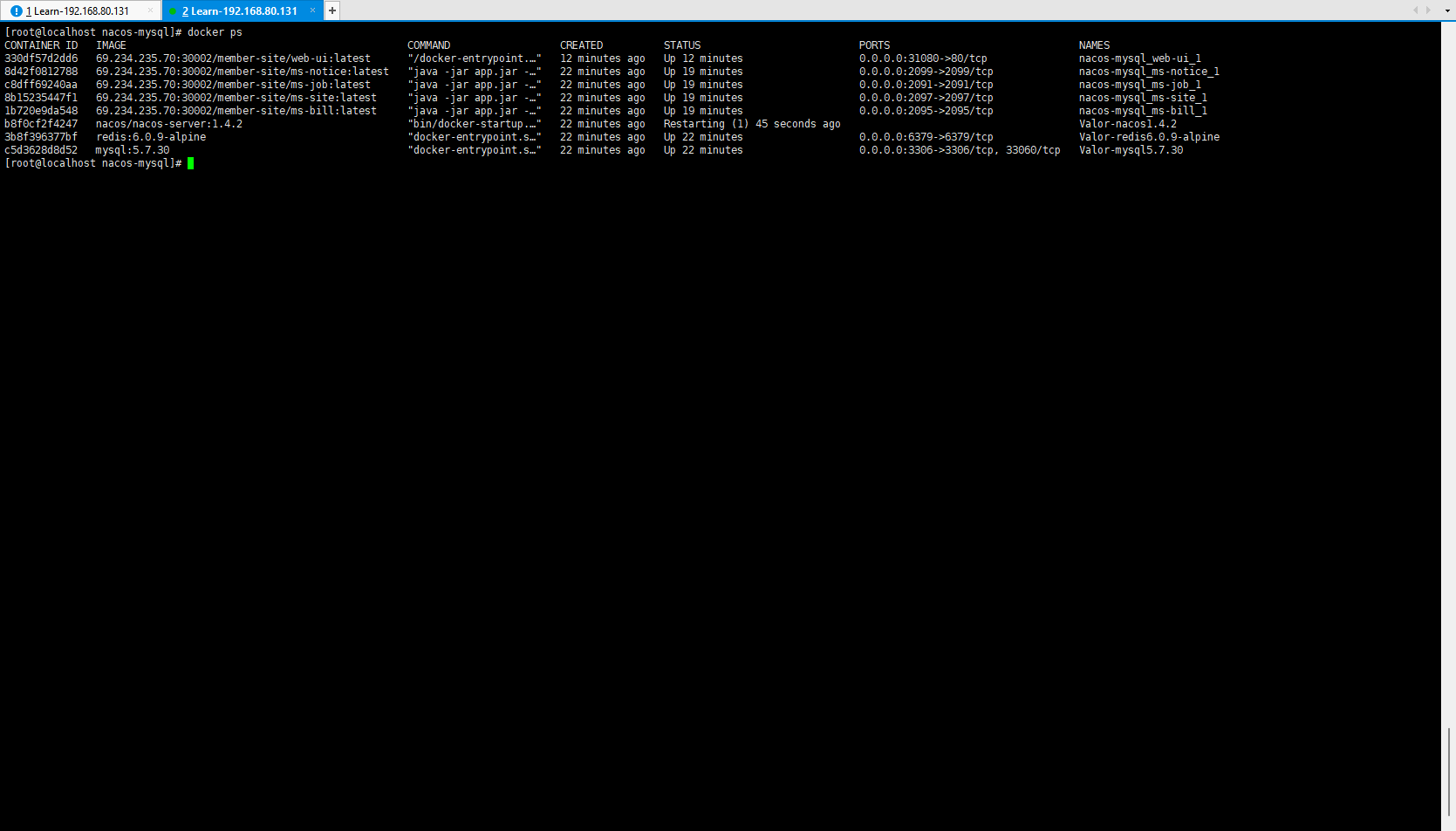
修改member-site要使用到的数据库

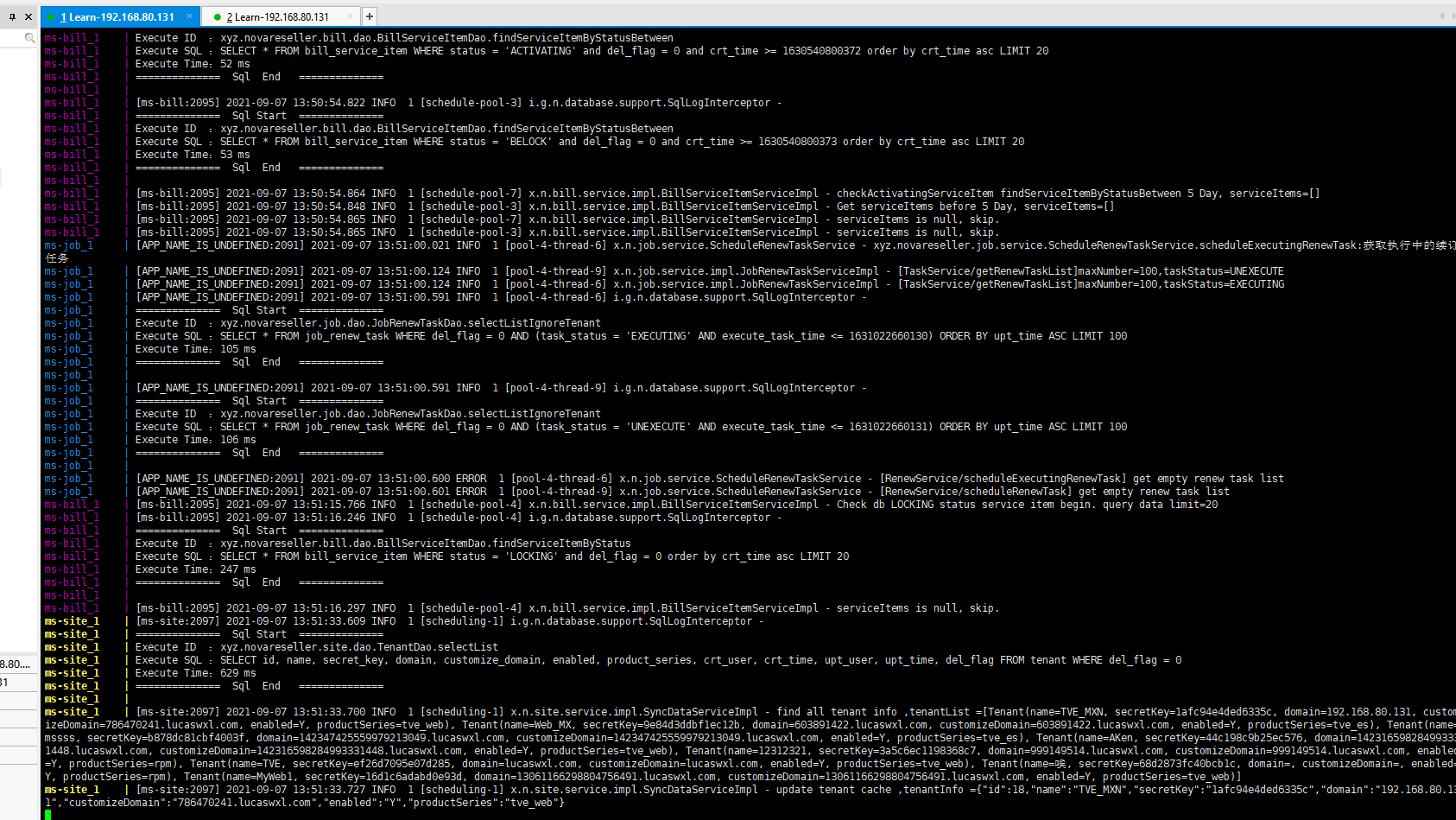


修改member-site要使用到的redis

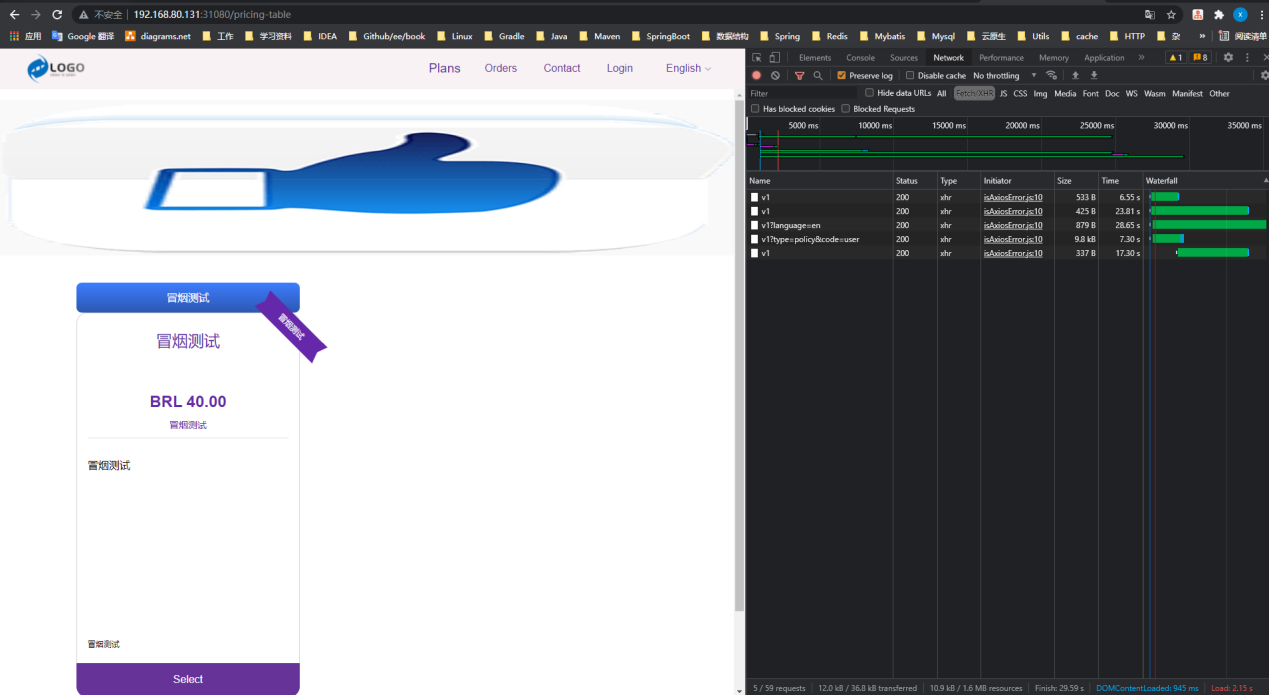


这些工作做完后，再次执行命令docker-compose up，这一次所有容器的构建成功。使用docker ps 即可查看容器运行情况。





最后一步：浏览器访问前端页面，192.168.80.131：31080发现整个项目都可以正常访问了。



注意：因为本次实践使用的虚拟机搭建的CentOS,分配的资源为:2核2G。资源偏小，至于访问页面，资源请求加载速度过慢。所以提醒大家对于服务器的资源要多分配。

本次panle会员网站的容器化部署实践工作完成。

至此！撒花，完结！

实现人员：Simple

完成时间：2021、9、7

版权编号：1008611