

实验 14：管理系统发布

一、实验目的

- 1、掌握 Docker 容器化部署的基本流程
- 2、理解前后端分离项目的容器化架构
- 3、能够在 openEuler 系统上部署完整的管理系统

二、实验学时

2 学时

三、实验类型

综合性

四、实验需求

1、硬件

每人配备计算机 1 台，建议优先使用个人计算机开展实验。

2、软件

安装 Visual Studio Code，以及 Edge 浏览器。

3、网络

本地主机能够访问互联网和实验中心网络。

4、工具

无。

五、实验任务

- 1、在 openEuler 上安装 Docker 环境

- 2、构建前端Vue应用的 Docker 镜像
- 3、构建后端APIJSON的 Docker 镜像
- 4、配置 MySQL 数据库容器
- 5、使用 Docker Compose 编排多服务
- 6、测试并验证系统

六、实验内容及步骤

1、openEuler 系统准备

步骤 1：安装 Docker 和 Docker Compose

Shell

```
1 # 1. 卸载旧版本
2 sudo yum remove docker \
3     docker-client \
4     docker-client-latest \
5     docker-common \
6     docker-latest \
7     docker-latest-logrotate \
8     docker-logrotate \
9     docker-engine
10
11 # 2. 安装Docker
12 sudo yum install -y docker
13 sudo systemctl start docker
14 sudo systemctl enable docker
15
16 # 3. 安装Docker Compose
17 sudo curl -L "https://github.com/docker/compose/releases/latest/download/docker-compose-$(uname -s)-$(uname -m)" \
18     -o /usr/local/bin/docker-compose
19 sudo chmod +x /usr/local/bin/docker-compose
20
21 # 4. 验证安装
22 docker --version
23 docker-compose --version
```

步骤 2：创建项目目录结构

Shell

```
1 mkdir -p /opt/tcm-system/{frontend,backend,database,nginx}  
2 cd /opt/tcm-system
```

2、前端容器化部署

步骤 1：准备前端项目

将前端项目代码上传至 /opt/tcm-system/frontend。

步骤 2：编写前端 Dockerfile

Dockerfile

```
1 # Dockerfile.frontend
2 FROM node:18-alpine as builder
3
4 # 设置工作目录
5 WORKDIR /app
6
7 # 复制包管理文件
8 COPY package*.json ./
9
10 # 安装依赖
11 RUN npm install --registry=https://registry.npmmirror.com
12
13 # 复制源代码
14 COPY . .
15
16 # 构建应用
17 RUN npm run build
18
19 # 使用nginx作为生产服务器
20 FROM nginx:alpine
21
22 # 复制nginx配置
23 COPY nginx.conf /etc/nginx/conf.d/default.conf
24
25 # 从构建阶段复制dist文件
26 COPY --from=builder /app/dist /usr/share/nginx/html
27
28 # 暴露端口
29 EXPOSE 80
30
31 # 启动nginx
32 CMD ["nginx", "-g", "daemon off;"]
```

步骤3：编写前端Nginx配置

```
1 # nginx.conf
2 server {
3     listen 80;
4     server_name localhost;
5
6     root /usr/share/nginx/html;
7     index index.html;
8
9     # 开启gzip
10    gzip on;
11    gzip_types text/plain text/css application/json application/javascript
    text/xml application/xml application/xml+rss text/javascript;
12
13    # 处理SPA路由
14    location / {
15        try_files $uri $uri/ /index.html;
16    }
17
18    # API代理
19    location /api/ {
20        proxy_pass http://backend:8080/;
21        proxy_set_header Host $host;
22        proxy_set_header X-Real-IP $remote_addr;
23        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
24        proxy_set_header X-Forwarded-Proto $scheme;
25    }
26 }
```

3、后端容器化部署

步骤 1：准备后端项目

将后端项目代码上传至 /opt/tcm-system/backend。

步骤 2：编写后端 Dockerfile

Dockerfile

```
1 # Dockerfile.backend
2 # 使用openEuler基础镜像
3 FROM openeuler/openeuler:22.03-lts
4
5 # 设置工作目录
6 WORKDIR /app
7
8 # 安装OpenJDK 8
9 RUN yum install -y java-1.8.0-openjdk java-1.8.0-openjdk-devel maven & \
10     yum clean all
11
12 # 复制源代码
13 COPY . .
14
15 # 构建应用
16 RUN mvn clean package -DskipTests
17
18 # 复制构建结果
19 RUN cp target/*.jar app.jar
20
21 # 暴露端口
22 EXPOSE 8080
23
24 # 设置JVM参数
25 ENV JAVA_OPTS="-Xmx512m -Xms256m -Dspring.profiles.active=prod"
26
27 # 启动应用
28 ENTRYPOINT ["sh", "-c", "java $JAVA_OPTS -jar app.jar"]
```

步骤3：修改后端配置文件

YAML

```
1 # application-docker.yml
2 server:
3   port: 8080
4   servlet:
5     context-path: /
6
7 spring:
8   datasource:
9     url: jdbc:mysql://mysql:3306/tcm_db?useUnicode=true&characterEncoding=UTF-8&serverTimezone=Asia/Shanghai
10    username: tcm_user
11    password: Tcm123456
12    driver-class-name: com.mysql.cj.jdbc.Driver
13    hikari:
14      maximum-pool-size: 10
15      minimum-idle: 5
16      connection-timeout: 30000
17
18 logging:
19   level:
20     com.github.apijson: INFO
21   file:
22     name: logs/tcm-api.log
23   pattern:
24     file: "%d{yyyy-MM-dd HH:mm:ss} [%thread] %-5level %logger{36} - %msg%n"
```

4、数据库容器配置

步骤 1：准备数据库初始化脚本

SQL

```
1 -- init.sql
2 CREATE DATABASE IF NOT EXISTS tcm_db;
3 USE tcm_db;
4
5 -- 创建用户
6 CREATE USER 'tcm_user'@'%' IDENTIFIED BY 'Tcm123456';
7 GRANT ALL PRIVILEGES ON tcm_db.* TO 'tcm_user'@'%';
8 FLUSH PRIVILEGES;
```

步骤 2：编写数据库 Dockerfile

Dockerfile

```
1 # Dockerfile.database
2 FROM mysql:8.0
3
4 # 设置字符集
5 ENV LANG=C.UTF-8
6
7 # 复制初始化脚本
8 COPY init.sql /docker-entrypoint-initdb.d/
9
10 # 设置时区
11 ENV TZ=Asia/Shanghai
```

5、Docker Compose 编排

步骤 1：编写 docker-compose.yml

YAML

```
1 version: '3.8'
2
3 services:
4   # MySQL数据库
5   mysql:
6     build:
7       context: ./database
8       dockerfile: Dockerfile.database
9     container_name: tcm-mysql
10    restart: always
11    environment:
12      MYSQL_ROOT_PASSWORD: Root123456
13      MYSQL_DATABASE: tcm_db
14    ports:
15      - "3306:3306"
16    volumes:
17      - mysql_data:/var/lib/mysql
18      - ./database/init.sql:/docker-entrypoint-initdb.d/init.sql
19    networks:
20      - tcm-network
21    healthcheck:
22      test: ["CMD", "mysqladmin", "ping", "-h", "localhost"]
23      timeout: 20s
24      retries: 10
25
26   # APIJSON后端
27   backend:
28     build:
29       context: ./backend
30       dockerfile: Dockerfile.backend
31     container_name: tcm-backend
32     restart: always
33     depends_on:
34       mysql:
35         condition: service_healthy
36     ports:
37       - "8080:8080"
38     environment:
39       - SPRING_PROFILES_ACTIVE=docker
40     volumes:
41       - backend_logs:/app/logs
42     networks:
```

```
43     - tcm-network
44     healthcheck:
45         test: ["CMD", "curl", "-f", "http://localhost:8080/get"]
46         interval: 30s
47         timeout: 10s
48         retries: 3
49
50 # Vue前端
51 frontend:
52     build:
53         context: ./frontend
54         dockerfile: Dockerfile.frontend
55     container_name: tcm-frontend
56     restart: always
57     depends_on:
58         - backend
59     ports:
60         - "80:80"
61     networks:
62         - tcm-network
63     healthcheck:
64         test: ["CMD", "curl", "-f", "http://localhost:80"]
65         interval: 30s
66         timeout: 10s
67         retries: 3
68
69 # Nginx反向代理（可选）
70 nginx:
71     image: nginx:alpine
72     container_name: tcm-nginx
73     restart: always
74     depends_on:
75         - frontend
76         - backend
77     ports:
78         - "8000:80"
79     volumes:
80         - ./nginx/nginx.conf:/etc/nginx/nginx.conf
81         - ./nginx/conf.d:/etc/nginx/conf.d
82     networks:
83         - tcm-network
84
85 networks:
86     tcm-network:
```

```
87     driver: bridge
88
89 volumes:
90     mysql_data:
91     backend_logs:
```

步骤 2：使用 Docker Compose 启动项目

Bash

```
1 # 停止并删除旧容器
2 docker-compose down
3
4 # 构建并启动新容器
5 docker-compose build
6 docker-compose up -d
7
8 # 检查服务状态
9 docker-compose ps
```

6、测试并验证系统

- (1) 访问 <http://<服务器 IP>:80> 查看界面
- (2) 登录系统，测试药材管理功能

七、实验考核

本实验考核采用【实验随堂查】方式开展。

每个实验完成后，在实验课上通过现场演示的方式向实验指导教师进行汇报，并完成现场问答交流。

每个实验考核满分 100 分，其中实验成果汇报 60 分，现场提问交流 40 分。

实验考核流程：

- (1) 学生演示汇报实验内容的完成情况，实验指导老师现场打分。
- (2) 指导老师结合实验内容进行提问，每位学生提问 2-3 个问题，根据回答的情况现场打分。
- (3) 实验考核结束后，进行公布成绩。