实验 15：发布到云平台

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 IntelliJ IDEA Community。

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 Java 的 Docker 镜像

4、构建 MySQL 数据库的 Docker 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，示例代码如下。

Shell

```
# 1. 卸载旧版本
sudo yum remove docker \
    docker-client \
    docker-client-latest \
    docker-common \
    docker-latest \
    docker-latest-logrotate \
    docker-logrotate \
    docker-engine

# 2. 安装Docker
sudo yum install -y docker
sudo systemctl start docker
sudo systemctl enable docker

# 3. 安装Docker Compose
sudo curl -L "https://github.com/docker/compose/releases/latest/download/docker-compose-$(uname -s)-$(uname -m)" \
    -o /usr/local/bin/docker-compose
sudo chmod +x /usr/local/bin/docker-compose

# 4. 验证安装
docker --version
docker-compose --version
```

步骤 2：创建项目目录结构，示例代码如下。

```shell
mkdir -p /opt/tcm-system/{frontend,backend,database,nginx}
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，示例代码如下。

Dockerfile

```
# Dockerfile.frontend
FROM node:18-alpine as builder

# 设置工作目录
WORKDIR /app

# 复制包管理文件
COPY package*.json ./

# 安装依赖
RUN npm install --registry=https://registry.npmmirror.com

# 复制源代码
COPY . .

# 构建应用
RUN npm run build

# 使用nginx作为生产服务器
FROM nginx:alpine

# 复制nginx配置
COPY nginx.conf /etc/nginx/conf.d/default.conf

# 从构建阶段复制dist文件
COPY --from=builder /app/dist /usr/share/nginx/html

# 暴露端口
EXPOSE 80

# 启动nginx
CMD ["nginx", "-g", "daemon off;"]
```

步骤 3：编写前端 Nginx 配置，示例代码如下。

Plain Text

```
# nginx.conf
server {
    listen 80;
    server_name localhost;

    root /usr/share/nginx/html;
    index index.html;

    # 开启gzip
    gzip on;
    gzip_types text/plain text/css application/json application/javascript text/xml application/xml application/xml+rss text/javascript;

    # 处理SPA路由
    location / {
        try_files $uri $uri/ /index.html;
    }

    # API代理
    location /api/ {
        proxy_pass http://backend:8080/;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}
```

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，示例代码如下。

Dockerfile

```
# Dockerfile.backend
# 使用openEuler基础镜像
FROM openeuler/openeuler:22.03-lts

# 设置工作目录
WORKDIR /app

# 安装OpenJDK 8
RUN yum install -y java-1.8.0-openjdk java-1.8.0-openjdk-devel maven && \
    yum clean all

# 复制源代码
COPY . .

# 构建应用
RUN mvn clean package -DskipTests

# 复制构建结果
RUN cp target/*.jar app.jar

# 暴露端口
EXPOSE 8080

# 设置JVM参数
ENV JAVA_OPTS="-Xmx512m -Xms256m -Dspring.profiles.active=prod"

# 启动应用
ENTRYPOINT ["sh", "-c", "java $JAVA_OPTS -jar app.jar"]
```

步骤 3：修改后端配置文件，示例代码如下。

YAML

```yaml
# application-docker.yml
server:
  port: 8080
  servlet:
    context-path: /

spring:
  datasource:
    url: jdbc:mysql://mysql:3306/tcm_db?useUnicode=true&characterEncoding=UTF-8&serverTimezone=Asia/Shanghai
    username: tcm_user
    password: Tcm123456
    driver-class-name: com.mysql.cj.jdbc.Driver
    hikari:
      maximum-pool-size: 10
      minimum-idle: 5
      connection-timeout: 30000
```

4、数据库容器配置

步骤 1：准备数据库初始化脚本，示例代码如下。

SQL

```sql
-- init.sql
CREATE DATABASE IF NOT EXISTS tcm_db;
USE tcm_db;

-- 创建用户
CREATE USER 'tcm_user'@'%' IDENTIFIED BY 'Tcm123456';
GRANT ALL PRIVILEGES ON tcm_db.* TO 'tcm_user'@'%';
FLUSH PRIVILEGES;
```

步骤 2：编写数据库 Dockerfile，示例代码如下。

Dockerfile

```dockerfile
# Dockerfile.database
FROM mysql:8.0

# 设置字符集
ENV LANG=C.UTF-8

# 复制初始化脚本
COPY init.sql /docker-entrypoint-initdb.d/

# 设置时区
ENV TZ=Asia/Shanghai
```

5、Docker Compose 编排

步骤 1：编写 docker-compose.yml，示例代码如下。

YAML

```yaml
version: '3.8'

services:
  # MySQL数据库
  mysql:
    build:
      context: ./database
      dockerfile: Dockerfile.database
    container_name: tcm-mysql
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: Root123456
      MYSQL_DATABASE: tcm_db
    ports:
      - "3306:3306"
    volumes:
      - mysql_data:/var/lib/mysql
      - ./database/init.sql:/docker-entrypoint-initdb.d/init.sql
    networks:
      - tcm-network
    healthcheck:
      test: ["CMD", "mysqladmin", "ping", "-h", "localhost"]
      timeout: 20s
      retries: 10

  # Java后端
  backend:
    build:
      context: ./backend
      dockerfile: Dockerfile.backend
    container_name: tcm-backend
    restart: always
    depends_on:
      mysql:
        condition: service_healthy
    ports:
      - "8080:8080"
    environment:
      - SPRING_PROFILES_ACTIVE=docker
    volumes:
      - backend_logs:/app/logs
    networks:
```

```yaml
      - tcm-network
    healthcheck:
      test: ["CMD", "curl", "-f", "http://localhost:8080/get"]
      interval: 30s
      timeout: 10s
      retries: 3

  # Vue前端
  frontend:
    build:
      context: ./frontend
      dockerfile: Dockerfile.frontend
    container_name: tcm-frontend
    restart: always
    depends_on:
      - backend
    ports:
      - "80:80"
    networks:
      - tcm-network
    healthcheck:
      test: ["CMD", "curl", "-f", "http://localhost:80"]
      interval: 30s
      timeout: 10s
      retries: 3

  # Nginx反向代理（可选）
  nginx:
    image: nginx:alpine
    container_name: tcm-nginx
    restart: always
    depends_on:
      - frontend
      - backend
    ports:
      - "8000:80"
    volumes:
      - ./nginx/nginx.conf:/etc/nginx/nginx.conf
      - ./nginx/conf.d:/etc/nginx/conf.d
    networks:
      - tcm-network

networks:
  tcm-network:
```

```
    driver: bridge

volumes:
  mysql_data:
  backend_logs:
```

步骤 2：使用 Docker Compose 启动项目，示例代码如下。

Bash

```
# 停止并删除旧容器
docker-compose down

# 构建并启动新容器
docker-compose build
docker-compose up -d

# 检查服务状态
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

1、本课程实验考核方案

本课程实验考核采用【实验智能评】【实验随堂查】方式开展，根据不同的实验内容选择不同的考核方式。

【实验智能评】：实验完成后提交 GitLab，通过自动化代码评审工具进行评分。

【实验随堂查】：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。

2、本实验考核要求

本实验考核方式：实验智能评

实验 13-15 作为本课程第 4 次实验考核。

考核要求：

（1）学生汇报本课程实验的最终成果，由评审组进行评分。

（2）学生汇报要脱离自身的开发环境，基于教学信创云计算平台进行汇报。

（3）学生汇报通过浏览器访问业务和展示部署成果，不使用多媒体课件和视频演示。