软件工程 hw2 报告文档

2022012110 李堃秀 likx22@mails.tsinghua.edu.cn

目录

一、作业要求

将清软论坛在云服务器上实现容器化部署。

二、实验环境

1、本地环境

Hyper-V 虚拟机，CentOS 7.8.2003+Docker version 26.1.4

2、云服务器

IP：140.143.163.191，Ubuntu 22.04+Docker 24.0.7

三、完成情况清单

	作业要求	结果
1	编写合适的 Dockerfile 以及 docker-compose 配置实现： 清软论坛以 MySQL 为数据库、通过 nginx 以 8000 端口向外提供服务； 通过服务器 ip:8000 可以访问论坛前端并正常操作； 通过服务器 ip:8000/api/v1 可以直接访问后端 API；	完成
2	Python 版本需要恰好为 3.8.x，你的 Nginx 版本为 latest，MySQL 版本恰好为 8.1	完成
3	各个 service 之间的依赖关系应设置合理（后端服务需要在 MySQL 服务初始化完成后再启动，你可以通过 depends_on 和 healthcheck 结合实现）。数据库健康检查的间隔时间应为 10 秒，超时时间为 5 秒，重试次数为 5 次；	完成
4	清软论坛镜像 container 名称为 app，nginx 镜像 container 名称为 nginx，MySQL 镜像的 container 名称为 mysql	完成
5	数据库使用账号 root（默认值），其密码为你的学号，数据库名称为 thss，使用端口 3306（默认值）。请注意 MySQL 默认镜像的时区为 UTC，字符集是 latin1，你也需要进行调整。	完成
6	前端文件应该通过 Nginx 的静态文件服务实现(在目前的项目中是通过单独的 React 服务实现的，你需要将前端通过 npm run build 打包后的产物，放在后端的 build 目录下，将其改为 nginx 静态文件服务实现)。你可以使用 volume 实现也可以构建一个基于 nginx 的镜像实现。同时你需要在 nginx 中实现反向代理，将 /api/v1 的请求转发到后端容器中；	完成
7	nginx 与论坛后端处于一个 network，论坛后端与数据库处于一个 network。也即通过 nginx 所在容器无法访问数据库容器	完成
8	仅 nginx 容器将端口映射给宿主机，端口号为 8000	完成
9	MySQL 镜像需要指定 /home/ubuntu/mysql/ 文件夹为持久化存储 Volume，将镜像内 /var/lib/mysql 目录挂载到宿主机的 /home/ubuntu/mysql/ 目录	完成

10	确保后端服务已正确连接至 MySQL 数据库而非 SQLite 数据库。在 docker-compose.yaml 文件中，必须确保后端容器遵循环境搭建章节中部署部分的指引，进行数据库迁移并启动应用。	完成

三、实现思路及过程

先在本地环境按以下步骤操作，完成调试并测试成功后，在云服务器上进行部署。

1、准备本地及云端服务器环境

使用 SSH 连接到服务器配置环境，安装 Docker，并确保 Docker 保持后台运行。

2、编写 Dockerfile 和 docker-compose.yml

1） Dockerfile

构建应用程序镜像。

```
# TODO: 补充 Dockerfile
FROM python:3.8

# Set working directory
WORKDIR /app

# Install dependencies
COPY requirements.txt requirements.txt
# 网络卡顿,使用清华镜像源
RUN pip install -r requirements.txt -i https://pypi.tuna.tsinghua.edu.cn/simple

# Copy application source code
COPY . .

# Set environment variables
ENV PYTHONUNBUFFERED 1
```

2） docker-compose.yml

配置多服务的编排以及依赖关系。

```
# TODO: 补充 docker compose 配置文件
version: '3.8'

services:
  app:
    build: .
```

```
    container_name: app
    environment:
      MYSQL_HOST: mysql
      MYSQL_DB: thss
      MYSQL_USER: root
      MYSQL_PASSWORD: 2022012110
    command:
      - /bin/bash
      - -c
      - |
        python manage.py migrate --settings=app.settings_prod
        python manage.py init_db --settings=app.settings_prod
        DJANGO_SETTINGS_MODULE=app.settings_prod   gunicorn   -w4   -b
0.0.0.0:8000 --log-level=debug app.wsgi
    depends_on:
      mysql:
        condition: service_healthy
    networks:
      - nginx_net
      - mysql_net

  nginx:
    image: nginx:latest
    container_name: nginx
    ports:
      - "8000:80"
    volumes:
      - ./nginx/app.conf:/etc/nginx/conf.d/default.conf
      - ./build:/usr/share/nginx/html
    depends_on:
      - app
    networks:
      - nginx_net

  mysql:
    image: mysql:8.1
    container_name: mysql
    environment:
      MYSQL_ROOT_PASSWORD: 2022012110
      MYSQL_DATABASE: thss
      TZ: Asia/Shanghai
    command: ['mysqld', '--character-set-server=utf8mb4', '--collation-
server=utf8mb4_unicode_ci']
    volumes:
```

```
      - /home/ubuntu/mysql/:/var/lib/mysql
    healthcheck:
      test: ["CMD", "mysqladmin", "ping", "-h", "localhost"]
      interval: 10s
      timeout: 5s
      retries: 5
    networks:
      - mysql_net

networks:
  nginx_net:
    driver: bridge
  mysql_net:
    driver: bridge
```

3、更新配置文件

1) nginx 配置文件（/backend/nginx/nginx.conf）

```
# TODO: 补充 Nginx 配置文件
server {
    listen 80;
    server_name localhost;

    location / {
        root /usr/share/nginx/html;
        try_files $uri /index.html;
    }
    location /admin {
        proxy_pass http://app:8000/admin;
        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;
    }
    location /api/v1 {
        proxy_pass http://app:8000/api/v1;
        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;
    }
}
```

2) 数据库连接信息及初始化参数(/backend/app/setting_prod.py)

```
DATABASES = {
```

```
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'thss',
        'USER': 'root',
        'PASSWORD': '2022012110',
        'HOST': 'mysql',
        'PORT': '3306',
        'OPTIONS': {
            'charset': 'utf8mb4',
            'init_command': 'SET sql_mode="STRICT_TRANS_TABLES"'
        },
    }
}
```

4、启动服务

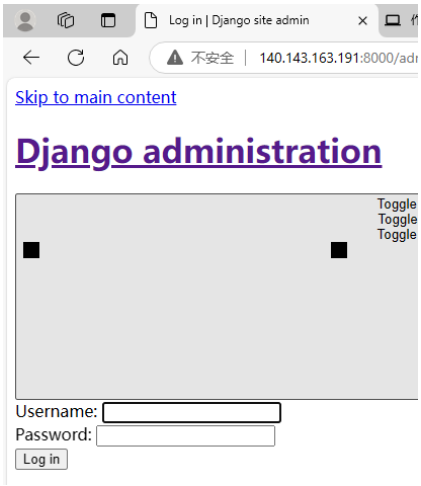
使用 docker-compose up 命令启动并运行服务，确认正常运行,进行各项功能测试及验证。

四、结果验证

1. 完成部署并正常访问

用前述 Dockerfile、docker-compose.yml 及相应配置文件部署完成后，访问云服务器（IP：http://140.143.163.191/）页面截图：

Admin:



API:

2. 版本要求

```
ubuntu@VM-24-17-ubuntu:~$ sudo docker images
REPOSITORY    TAG       IMAGE ID       CREATED          SIZE
backend_app   latest    032088e06163   10 hours ago     1.08GB
python        3.8       3ea6eaad4f17   3 weeks ago      995MB
nginx         latest    9527c0f683c3   7 weeks ago      188MB
mysql         8.1       ae2502152260   14 months ago    574MB
```

3. 依赖关系及数据库健康检查

详见 docker-compose.yml 配置。

4. container 名称要求

```
ubuntu@VM-24-17-ubuntu:~/code/backend$ sudo docker ps --format "table{{.ID}}\t{{.Names}}"
CONTAINER ID   NAMES
b339ea634de4   nginx
4da385a83322   app
73bc5d6a2176   mysql
ubuntu@VM-24-17-ubuntu:~/code/backend$
```

5. 数据库设置

```
mysql:
  image: mysql:8.1
  container_name: mysql
  environment:
    MYSQL_ROOT_PASSWORD: 2022012110
    MYSQL_DATABASE: thss
    TZ: Asia/Shanghai
  command: ['mysqld', '--character-set-server=utf8mb4', '--collation-server=utf8mb4_unicode_ci']
  volumes:
    - /home/ubuntu/mysql/:/var/lib/mysql
```

6. 静态页面及 Nginx 反向代理

详见 nginx 配置文件：

```
ubuntu@VM-24-17-ubuntu:~$ cat code/backend/nginx/app.conf
# TODO: 补充Nginx配置文件
server {
    listen 80;
    server_name localhost;

    location / {
        root /usr/share/nginx/html;
        try_files $uri /index.html;
    }
        location /admin {
            proxy_pass http://app:8000/admin;
        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;
    }
    location /api/v1 {
        proxy_pass http://app:8000/api/v1;
        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;
    }

}ubuntu@VM-24-17-ubuntu:~$
```

7. Network 隔离

配置 nginx_net 及 mysql_net 两个 network,详见 docker-compose.yml 配置：

```
version: '3.8'

services:
  app:
    build: .
    container_name: app
    environment:
      MYSQL_HOST: mysql
      MYSQL_PORT: 8000
      MYSQL_DB: thss
      MYSQL_USER: root
      MYSQL_PASSWORD: 2022012110
    depends_on:
      mysql:
        condition: service_healthy
    networks:
      - nginx_net
      - mysql_net

  nginx:
    image: nginx:latest
    container_name: nginx
    ports:
      - "8000:80"
    volumes:
      - ./nginx/app.conf:/etc/nginx/conf.d/default.conf
      - ./build:/usr/share/nginx/html
    depends_on:
      - app
    networks:
      - nginx_net

  mysql:
    image: mysql:8.1
    container_name: mysql
    environment:
      MYSQL_ROOT_PASSWORD: 2022012110
      MYSQL_DATABASE: thss
      TZ: Asia/Shanghai
    volumes:
      - /home/ubuntu/mysql/:/var/lib/mysql
    healthcheck:
      test: ["CMD", "mysqladmin", "ping", "-h", "localhost"]
      interval: 10s
      timeout: 5s
      retries: 3
    networks:
      - mysql_net

networks:
  nginx_net:
    driver: bridge
  mysql_net:
    driver: bridge
ubuntu@VM-24-17-ubuntu:~/code/backend$
```

启动服务后查询三个容器的 IP 如下：

```
ubuntu@VM-24-17-ubuntu:~/code/backend/nginx$ sudo docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS                    P
8485399c53c5   nginx:latest   "/docker-entrypoint.…"   13 minutes ago   Up 13 minutes             0
37305119b3fd   backend_app    "/bin/bash -c 'pytho…"   13 minutes ago   Up 13 minutes
718aa6b5307f   mysql:8.1      "docker-entrypoint.s…"   13 minutes ago   Up 13 minutes (healthy)   3
ubuntu@VM-24-17-ubuntu:~/code/backend/nginx$ sudo docker inspect 8485399c53c5 | grep "IPAddress"
            "SecondaryIPAddresses": null,
            "IPAddress": "",
                    "IPAddress": "172.30.0.3",nginx
ubuntu@VM-24-17-ubuntu:~/code/backend/nginx$ sudo docker inspect 37305119b3fd | grep "IPAddress"
            "SecondaryIPAddresses": null,
            "IPAddress": "",
                    "IPAddress": "172.29.0.3", app
                    "IPAddress": "172.30.0.2",
ubuntu@VM-24-17-ubuntu:~/code/backend/nginx$ sudo docker inspect 718aa6b5307f | grep "IPAddress"
            "SecondaryIPAddresses": null,
            "IPAddress": "",
                    "IPAddress": "172.29.0.2",mysql
ubuntu@VM-24-17-ubuntu:~/code/backend/nginx$
```

8. Nginx 端口开放

详见 nginx 配置文件。

9. Mysql 数据文件持久化存储

在 docker-compose.yml 中配置

```
  mysql:
    image: mysql:8.1
    container_name: mysql
    environment:
      MYSQL_ROOT_PASSWORD: 2022012110
      MYSQL_DATABASE: thss
      TZ: Asia/Shanghai
    volumes:
      - /home/ubuntu/mysql/:/var/lib/mysql
    healthcheck:
```

ls 宿主机及容器的相关目录文件：

```
ubuntu@VM-24-17-ubuntu:~$ sudo docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS                    PO
RTS                                        NAMES
b339ea634de4   nginx:latest   "/docker-entrypoint.…"   10 hours ago   Up 10 hours               0.
0.0.0:8000->80/tcp, :::8000->80/tcp        nginx
4da385a83322   backend_app    "gunicorn -b 0.0.0.0…"   10 hours ago   Up 10 hours
                                           app
73bc5d6a2176   mysql:8.1      "docker-entrypoint.s…"   10 hours ago   Up 10 hours (healthy)     33
06/tcp, 33060/tcp                          mysql
ubuntu@VM-24-17-ubuntu:~$ sudo docker exec -it 73bc5d6a2176 bash
bash-4.4# ls /var/lib/mysql
 #ib_16384_0.dblwr    binlog.000008   binlog.000020     mysql
'#ib_16384_1.dblwr'   binlog.000009   binlog.000021     mysql.ibd
'#innodb_redo'        binlog.000010   binlog.000022     mysql.sock
'#innodb_temp'        binlog.000011   binlog.000023     performance_schema
 auto.cnf             binlog.000012   binlog.index      private_key.pem
 binlog.000001        binlog.000013   ca-key.pem        public_key.pem
 binlog.000002        binlog.000014   ca.pem            server-cert.pem
 binlog.000003        binlog.000015   client-cert.pem   server-key.pem
 binlog.000004        binlog.000016   client-key.pem    sys
 binlog.000005        binlog.000017   ib_buffer_pool    thss
 binlog.000006        binlog.000018   ibdata1           undo_001
 binlog.000007        binlog.000019   ibtmp1            undo_002
bash-4.4# exit
exit
ubuntu@VM-24-17-ubuntu:~$ ls /home/ubuntu/mysql
 auto.cnf             binlog.000010   binlog.000020      '#ib_16384_1.dblwr'   private_key.pem
 binlog.000001        binlog.000011   binlog.000021       ib_buffer_pool       public_key.pem
 binlog.000002        binlog.000012   binlog.000022       ibdata1              server-cert.pem
 binlog.000003        binlog.000013   binlog.000023       ibtmp1               server-key.pem
 binlog.000004        binlog.000014   binlog.index       '#innodb_redo'        sys
 binlog.000005        binlog.000015   ca-key.pem         '#innodb_temp'        thss
 binlog.000006        binlog.000016   ca.pem              mysql                undo_001
 binlog.000007        binlog.000017   client-cert.pem     mysql.ibd            undo_002
 binlog.000008        binlog.000018   client-key.pem      mysql.sock
 binlog.000009        binlog.000019  '#ib_16384_0.dblwr'  performance_schema
ubuntu@VM-24-17-ubuntu:~$
```

在 thss 数据库中 create table（见下图的 lkx_test_table 表），使用 Docker-compose down 停止服务，并确认容器关闭后重新 docker-compose up，验证数据持久化。

```
ubuntu@VM-24-17-ubuntu:~$ sudo docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS
          NAMES
e682fe07ebf8   nginx:latest   "/docker-entrypoint.…"   14 hours ago   Up 14 hours
->80/tcp   nginx
04249b315a62   backend_app    "gunicorn -b 0.0.0.0…"   14 hours ago   Up 14 hours
          app
d6144d2eccea   mysql:8.1      "docker-entrypoint.s…"   14 hours ago   Up 14 hours (healthy)
          mysql
ubuntu@VM-24-17-ubuntu:~$ sudo docker exec -it d6144d2eccea bash
bash-4.4# mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 5062
Server version: 8.1.0 MySQL Community Server - GPL

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> Use thss;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> SHOW TABLES;
+----------------------------+
| Tables_in_thss             |
+----------------------------+
| auth_group                 |
| auth_group_permissions     |
| auth_permission            |
| auth_user                  |
| auth_user_groups           |
| auth_user_user_permissions |
| django_admin_log           |
| django_content_type        |
| django_migrations          |
| django_session             |
| lkx_test_table             |
| post_post                  |
| post_reply                 |
| user_user                  |
+----------------------------+
14 rows in set (0.00 sec)

mysql>
```

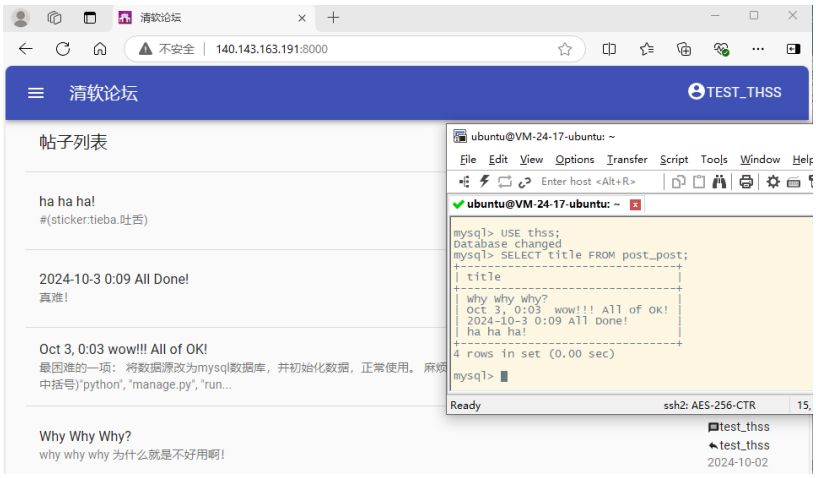
10. 数据库切换并启动应用

配置数据库接口，修改 docker-compose.yml，遵循环境搭建章节中部署部分的指引，进行数据库切换并启动 Gunicorn 服务器。

```
ubuntu@VM-24-17-ubuntu:~/code/backend$ cat docker-compose.yml
# TODO: 补充docker compose配置文件
version: '3.8'

services:
  app:
    build: .
    container_name: app
    environment:
      MYSQL_HOST: mysql
      MYSQL_PORT: 8000
      MYSQL_DB: thss
      MYSQL_USER: root
      MYSQL_PASSWORD: 2022012110
    command:
      - /bin/bash
      - -c
      - |
        python manage.py migrate --settings=app.settings_prod;
        python manage.py init_db;
        DJANGO_SETTINGS_MODULE=app.settings_prod gunicorn -w4 -b 0.0.0.0:8000 --log-level=debug app.wsgi
    depends_on:
      mysql:
        condition: service_healthy
    networks:
      - nginx_net
      - mysql_net
```

通过在前台界面多次编辑，同步在 mysql 容器中对数据库进行查询，确认数据切换无误。



五、结论及体会

通过上述步骤和代码，本次个人任务成功地完成了清软论坛的应用容器化部署任务。通过这次作业，我有以下几点体会：

1、做好计划，稳扎稳打。在面对一项任务时，首先要明确任务要求，在头脑中想清楚每一步要做的事，不管是从环境安装、配置、调试到最后的服务启动，每一步都力求清晰明了，并且在执行过程中要保持足够的冷静，确保每一步都准确无误并记录清晰，这样在遇到问题时，很容易分析原因并加以解决。

2、反复验证，确认无误。在本地环境以及云服务器配置成功后，我在本地重新建了一个空的环境从头开始部署，服务正常启动但始终无法登录，经查询数据库发现没有初始化 test_thss 用户，尽管 user_user 是空表但运行初始代码时始终提示 UNIQUE constraint failed: user_user.username，确认一切操作无误后，发现需要修改 docker-compose.yml 文件，在数据库的初始代码后加上参数，问题得到解决。

3、深刻理解了配置管理的重要性。灵活用好 Docker，可以大大简化环境配置和服务管理，提高部署效率。同时，我在任务中遇到了各服务循环依赖导致程序报错，解决该问题的过程也让我充分理解各个服务间的依赖关系，合理配置健康检查和启动顺序，以确保整体系统的稳定性和可靠性。

4、在完成作业的过程中，我还遇到了诸如端口冲突、VI 操作、mysql 操作等很多小问题，这大大锻炼了我的 Linux 操作技能。

以上这些都让我受益匪浅，不仅锻炼了我的 linux 操作能力，更重要的是深化了我对 Docker 的理解和应用能力，为未来类似的容器化部署提供了宝贵的经

验。

参考文档：

1、软件工程课程项目文档，https://linliulab.github.io/SE-2024/

2、Docker 网站，https://docs.docker.com/

3、CSDN 网站,https://www.csdn.net/