

Lab1-group-3实验报告

小组分工

陈可：Dockerfile、docker-compose.yml, GitHub CI
















































张文擘：投稿功能

邹懿：同下

陈沛仪：在队友完成coding工作后，自己也独立完成了一遍三个任务

- Docker Hub用户名：zouyi0925
 - 投稿服务的仓库名：ase_contribute_service
- 注：最新tag为yys-lab1

1 部署成功截图

<input type="checkbox"/>	>		mysql-docker	-	Running (1/2)	48 seconds ago			
<input type="checkbox"/>	>		redis-docker	-	Running (1/1)	45 seconds ago			
<input type="checkbox"/>	>		nacos-docker	-	Running (2/2)	35 seconds ago			
<input type="checkbox"/>	>		paper_submission_review_system_0319	-	Running (7/7)	16 seconds ago			
<input type="checkbox"/>			ase-notification-service 477b76253814 	huajuan6848/ase_notification_service:yys	Running	19 seconds ago			
<input type="checkbox"/>			ase-file-service 3c7bad79130b 	huajuan6848/ase_file_service:yys-lab1	Running	19 seconds ago			
<input type="checkbox"/>			ase-gateway 1bfccac084ea 	huajuan6848/ase_gateway:yys-lab1	Running	18 seconds ago			
<input type="checkbox"/>			ase-user-service e377af2f3712 	huajuan6848/ase_user_service:yys-lab1	Running	18 seconds ago			
<input type="checkbox"/>			ase-frontend 7b15e97952ca 	huajuan6848/ase_frontend:yys-lab1	Running 80:80 	18 seconds ago			
<input type="checkbox"/>			ase-conference-service b29cf8ad9e26 	huajuan6848/ase_conference_service:yys	Running	19 seconds ago			

Showing 19 items

2 实验过程记录

2.1 任务一

- `contribute`

本函数需要实现用户投稿功能。

- 调用会议服务，获取要投稿的会议信息（基于Dubbo的RPC调用）
- 检查会议是否存在、是否正在投稿中、投稿时间是否已过，若出现异常则抛出 `RuntimeException`
- 以上检查都通过，说明用户可以正常投稿，则调用用户服务，将该用户在此会议中的身份设为作者（基于Dubbo的RPC调用）
- 创建 `Contribution` 对象存储投稿信息，并持久化到数据库中

```
public String contribute(ContributeRequest.In in) {  
    ConferenceDTO conferenceInfo =  
conferenceService.getConferenceInfoByName(in.getConferenceName());
```

```

        if (conferenceInfo == null) {
            throw new RuntimeException("Conference not found");
        }
        if (!conferenceInfo.getConferenceStatus().equals("投稿中")) {
            throw new RuntimeException("Conference not in submission phase");
        }
        if
(conferenceInfo.getSubmissionDeadline().isBefore(LocalDate.now())) {
            throw new RuntimeException("Submission deadline has passed");
        }
        iUserConferenceRoleService.addRoleToUserInConference(in.getUsername(),
in.getConferenceName(), ConferenceRole.AUTHOR);
        Contribution contribution = new Contribution(in.getUsername(),
            in.getRealName(), in.getConferenceName(), in.getTitle(),
            in.getAbstractContent(), in.getEssayId(), sdf.format(new
Date()));
        contributeRepository.save(contribution);

        return null;
    }

```

- `listContributionsByUsername`

本函数需要实现查找某位作者的所有投稿信息。

由于 `ContributeRepository` 中已经定义了接口: `List<Contribution>`

`findAllByAuthor(String author)` , 且JPA框架会自动解析函数名并实现函数, 无需我们显式实现。

注意到上述接口的返回数据类型是 `List<Contribution>` , 而本函数要求的返回数据类型是 `List<ListContribution>` , 查看这两个类定义, 发现 `ListContribution` 是 `Contribution` 类信息的子集, 因此我们只需要对查询到的 `Contribution` 对象进行转换即可:

```

public List<ListContribution> listContributionsByUsername(String author) {
    return contributeRepository.findAllByAuthor(author).stream()
        .map(contribution -> new ListContribution(contribution.getId(),
            contribution.getConferenceName(),
            contribution.getContributeTime(),
            contribution.getTitle(), contribution.getStatus()))
        .toList();
}

```

- `listContributionsByConferenceName`

本函数需要实现查找某会议的所有投稿信息。

同前一个函数一样, 只需要使用 `ContributeRepository` 中定义好的接口 `List<Contribution>` `findAllByConferenceName(String conferenceName)` , 同样将 `Contribution` 对象转换为 `ListContribution` 对象即可:

```

public List<ListContribution> listContributionsByConferenceName(String name)
{
    return contributeRepository.findAllByConferenceName(name).stream()
        .map(contribution -> new ListContribution(contribution.getId(),
            contribution.getConferenceName(),
            contribution.getContributeTime(),
            contribution.getTitle(), contribution.getStatus()))
        .toList();
}

```

- detailById

本函数需要实现根据投稿id查找投稿的详细信息。

同样我们使用 `ContributeRepository` 中定义好的接口 `Contribution` `findContributionById(long id)`，注意将 `String` 类型的参数转换为 `long` 型即可：

```

public Contribution detailById(String idStr) {
    long id = Long.parseLong(idStr);
    return contributeRepository.findContributionById(id);
}

```

2.2 任务二

- 编写Dockerfile文件

```

### 构建阶段

# 使用基础镜像
FROM openjdk:17-jdk as build

# 设置工作目录
WORKDIR /app

# 将当前目录下的所有文件复制到工作目录中
COPY . .

# 赋予 Maven wrapper 可执行权限
RUN chmod +x ./mvnw

# 使用 Maven 编译项目，并打包成可执行的 .jar 文件（跳过测试）
RUN ./mvnw clean package -pl ase_contribute_service -am -DskipTests

### 运行阶段

# 使用基础镜像
FROM openjdk:17-jdk

# 设置工作目录
WORKDIR /app

# 从构建阶段复制编译好的 .jar 文件到运行阶段的工作目录中
COPY --from=build /app/ase_contribute_service/target/*.jar /app/app.jar

```

```
# 从构建阶段复制生产环境的配置文件到运行阶段的工作目录中
COPY --from=build
/app/ase_contribute_service/src/main/resources/application-prod.yml
/app/application.yml

# 暴露容器的 8082 端口，以便外部可以访问应用程序
EXPOSE 8082

# 指定容器启动时执行的命令，即运行 .jar 文件
CMD ["java", "-jar", "app.jar", "--server.error.include-message=always"]
```

- **多架构镜像推送到docker hub上**

使用docker buildx工具，通过如下步骤

1. **创建 Buildx 构建实例**（如果尚未创建）：

```
docker buildx create --name mybuilder --use
```

2. **启动构建实例：**

```
docker buildx inspect --bootstrap
```

3. **构建并推送镜像：**

```
docker login
docker buildx build --platform linux/amd64,linux/arm64 -t
zouyi0925/ase_contribute_service:yys-lab1 -f
ase_contribute_service/Dockerfile . --push
```

最后在docker hub中可得到多种架构的镜像



- **拉取和运行镜像**

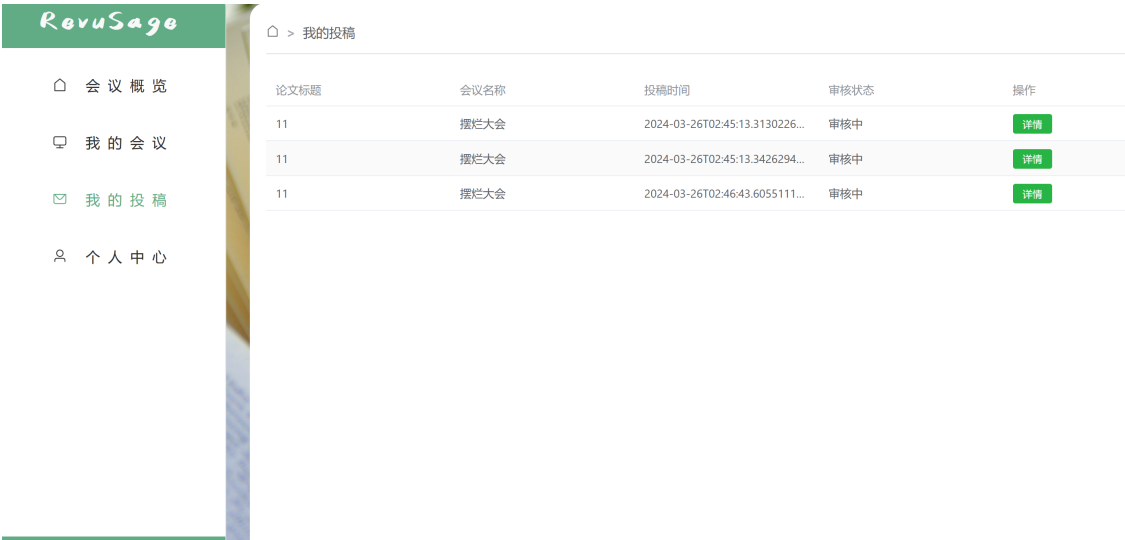
1. **从docker hub仓库中拉取镜像**

```
docker pull zouyi0925/ase_contribute_service:yys-lab1
```

1. **运行下面的命令，创造名为ase_contribute_test的容器**

```
docker run -d --network ase-network --name ase-contribute-test -e env=prod -
e NACOS_HOST=host.docker.internal -e MYSQL_HOST=host.docker.internal -e
REDIS_HOST=host.docker.internal ase_contribute_service
```

两步运行成功后，会看到前端投稿页面正常显示如下：



2.3 任务三

编写 `docker-compose.yml` 文件:

```
version: '3'

networks:
  ase-network:
    driver: bridge
    name: ase-network

volumes:
  file-data:

services:
  ase-contribute-service:
    image: zouyi0925/ase_contribute_service:yys-lab1
    environment:
      - env=prod
      - NACOS_HOST=host.docker.internal
      - NACOS_PORT=8848
      - MYSQL_HOST=host.docker.internal
      - MYSQL_PORT=13306
      - REDIS_HOST=host.docker.internal
    networks:
      - ase-network
    depends_on:
      - ase-file-service
    container_name: ase-contribute-service

  ase-conference-service:
    image: huaJuan6848/ase_conference_service:yys-lab1
    environment:
      - env=prod
      - NACOS_HOST=host.docker.internal
      - NACOS_PORT=8848
      - MYSQL_HOST=host.docker.internal
      - MYSQL_PORT=13306
      - REDIS_HOST=host.docker.internal
    networks:
```

- ase-network

container_name: ase-conference-service

ase-notification-service:

image: huajuan6848/ase_notification_service:yyz-lab1

environment:

- env=prod
- NACOS_HOST=host.docker.internal
- MYSQL_HOST=host.docker.internal
- REDIS_HOST=host.docker.internal

networks:

- ase-network

container_name: ase-notification-service

ase-user-service:

image: huajuan6848/ase_user_service:yyz-lab1

environment:

- env=prod
- NACOS_HOST=host.docker.internal
- MYSQL_HOST=host.docker.internal
- REDIS_HOST=host.docker.internal

networks:

- ase-network

container_name: ase-user-service

ase-file-service:

image: huajuan6848/ase_file_service:yyz-lab1

environment:

- env=prod
- NACOS_HOST=host.docker.internal
- MYSQL_HOST=host.docker.internal
- REDIS_HOST=host.docker.internal

networks:

- ase-network

container_name: ase-file-service

volumes:

- file-data:/root/upload

ase-gateway:

image: huajuan6848/ase_gateway:yyz-lab1

environment:

- env=prod
- NACOS_HOST=host.docker.internal
- MYSQL_HOST=host.docker.internal
- REDIS_HOST=host.docker.internal

networks:

- ase-network

container_name: ase-gateway

ase-frontend:

image: huajuan6848/ase_frontend:yyz-lab1

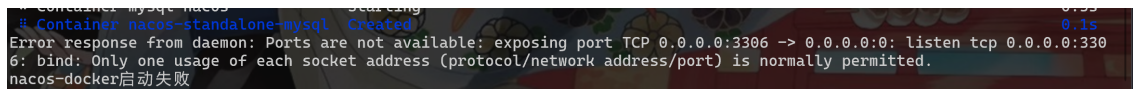
environment:

- env=prod
- NACOS_HOST=host.docker.internal
- MYSQL_HOST=host.docker.internal

```
- REDIS_HOST=host.docker.internal
networks:
- ase-network
ports:
- "80:80"
container_name: ase-frontend
```

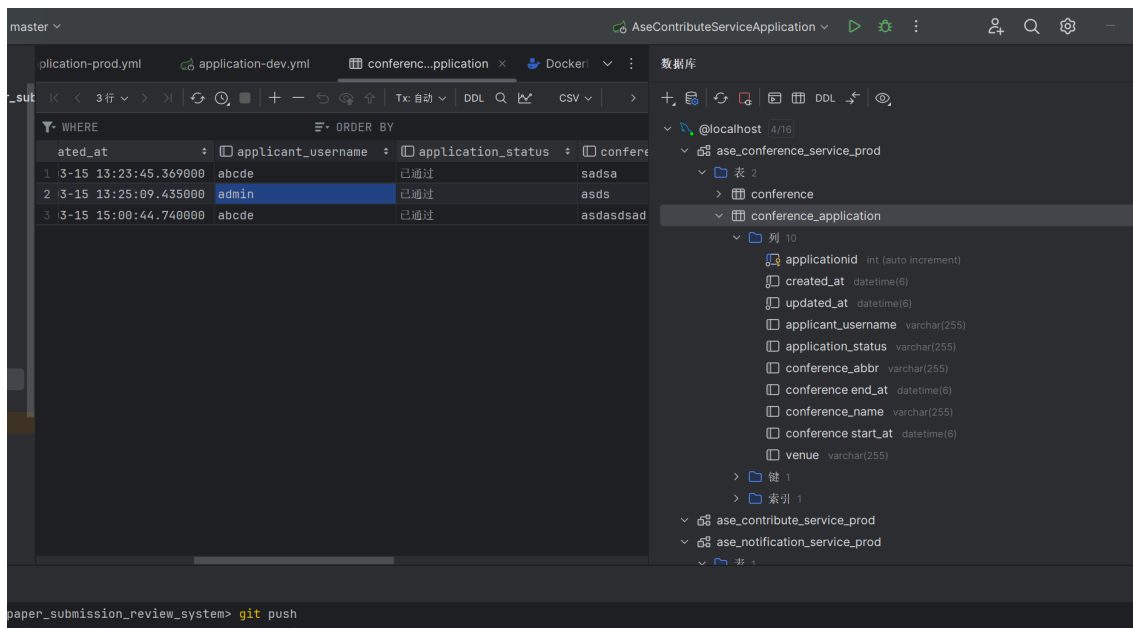
2.4 遇到的问题及解决方法

- 端口占用，中间件无法启动



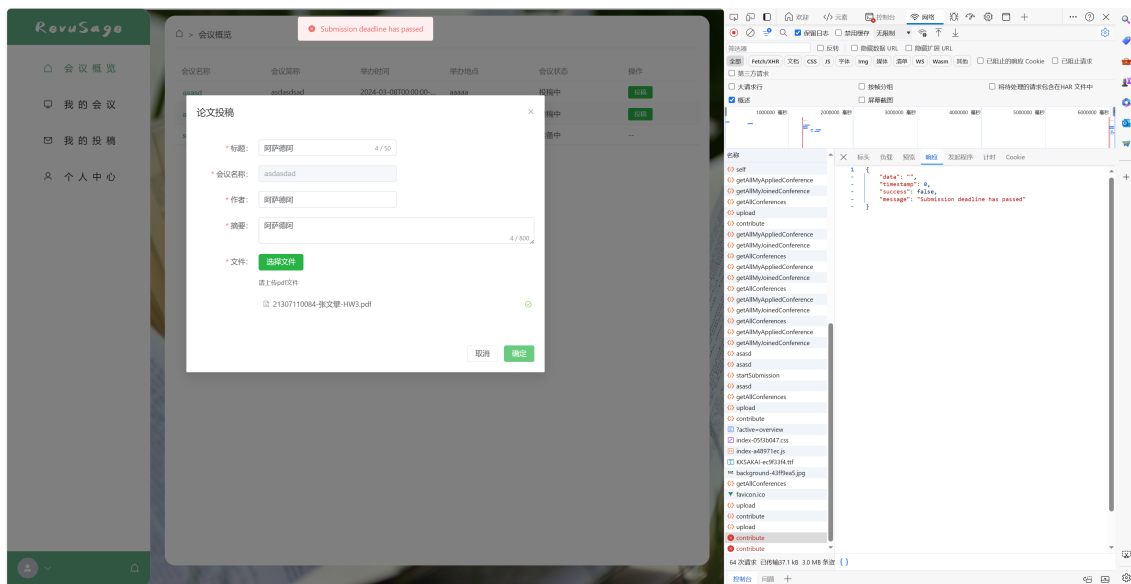
通过cmd执行 `netstat -ano|findstr "3306"`，查看端口运行进程对应的PID，在任务管理器结束该进程；若没有对应的进程，则重启一下winnet服务就好了，管理员打开终端输入 `net stop winnet` 然后再输入 `net start winnet`

- `SimpleDateFormat` 无法对 `LocalDateTime` 类型的时间进行格式化，需要改成 `Date` 类型
// p.s. 关于时间，不用考虑主机时间（CST）和容器时间（UTC）不一致的问题
- admin创建了会议



数据库中会议创建人为admin，与需求不符。经检查原因可能是退出时token未切换

- 加入会议过程中多次出现500错误
起初以为是服务启动失败，排查后发现是截止日期早于当前时间，框架返回的状态码具有迷惑性。
对controller稍作修改（仅为返回错误码，不考虑其他逻辑），成功看到错误信息



```
public ResponseWithData<String> contribute(@RequestBody @Validated
ContributerRequest.In in, HttpServletResponse response) {
    try {
        return new ResponseWithData(service.contribute(in));
    } catch (Exception e) {
        response.setStatus(HttpServletResponse.SC_FORBIDDEN);
        return new ResponseWithData(false, "", e.getMessage());
    }
}
```

- 其他服务：文件上传大小与后端限制不符

Nginx中文件传输大小为默认1M

file_service中prod和dev环境中的文件大小限制不一致

3 实验总结

陈可

本次实验使用 dockerfile 和 docker-compose 构建和部署 spring 微服务项目。spring 的类 monorepo 仓库是非常适合微服务开发的架构设计，dubbo/nacos 的 rpc 框架为微服务之间的通信提供了良好封装，是适合团体协作开发的项目。docker 的部署打破了传统服务部署的繁琐配置，使用 docker-compose 工具可以灵活直观地配置和启动项目，方便使用和调试。GitHub CI 的实现是本实验的一大亮点，在提交到代码仓库后，基于 GitHub Action 的功能可以直接执行一系列命令，打包构建上传镜像到 DockerHub，这避免了开发中停下来构建镜像的烦恼，极大地提高了敏捷开发的效率。

张文攀

本次lab进行了spring项目的构建和部署，学习到了dockerfile和docker-compose书写，尝试了简单的容器部署和githubCI功能，认识到持续集成部署对项目的重要性，感受到docker的强大功能和便捷特性。

邹懿

之前软工课上接触docker只是本地打包好jar然后单阶段直接部署，但是这次lab中，我在完成任务二、三发现了docker容器开发的更多作用，同时也感慨微服务开发的便捷。（除了就是lab中docker一开电脑cici响，没啥不好的）最后感谢我们组的ck和zwb同学，提前熟悉好整个项目，让我和py对项目更加了解，方便后续我们个人去独立完成所有任务！

陈沛仪

Lab的设置非常caring，带领小白从零上手docker，体验了一套完整的coding-构建镜像-组合系统-运行的流程，对微服务及云原生环境有了更深入的了解。感谢伟大的队友飞速写完了代码工作，让我和zy可以没有后顾之忧地慢慢探索。第一次体验这种新型小组合作模式（概括为四线并行，每个人都独立完成了一遍所有任务），感觉可以更自由地安排自己的时间，非常nice。同时也感谢助教不辞辛劳地答疑！