# 中间件部署平台

# 1、代码常量

## 1.1 构建者模式：

### 1.1.1构建者类：

package com.example.middlewaredeploy.build;   
import com.example.middlewaredeploy.entity.to.ShellFile;   
public abstract class Builder {   
 protected ShellFile file = new ShellFile();   
 public abstract void builderShell();   
 public abstract ShellFile createShell();   
}

### 1.1.2 指挥者类

package com.example.middlewaredeploy.build;   
import com.example.middlewaredeploy.entity.to.ShellFile;   
public class Director {   
 //声明 builder类型的变量   
 public Builder builder;   
 public Director(Builder builder){   
 this.builder = builder;   
 }   
 //组装文件   
 public ShellFile construct(){   
 builder.builderShell();   
 return builder.createShell();   
 }   
}

### 1.1.3 组件类

package com.example.middlewaredeploy.build.shellBuilder;   
import com.example.middlewaredeploy.build.Builder;   
import com.example.middlewaredeploy.constant.DockerShell;   
import com.example.middlewaredeploy.entity.to.ShellFile;   
import java.util.ArrayList;   
import java.util.Arrays;   
public class DockerBuilder extends Builder {   
 @Override   
 public void builderShell() {   
 ArrayList arrayList = new ArrayList<>(Arrays.asList(DockerShell.values()));   
 file.setAdapterList(arrayList);   
 }   
 @Override   
 public ShellFile createShell() {   
 return file;   
 }   
}

## 1.2 装饰者模式：

### 1.2.1 中间件配置策略接口

package com.example.middlewaredeploy.build;   
import java.net.URISyntaxException;   
/\*\*   
 \* @project middlewareDeploy   
 \* @description 中间件配置策略接口   
 \* @author capture or new   
 \* @date 2023/7/1 09:04:15   
 \* @version 1.0   
 \*/   
public interface MiddlewareConfigStrategy {   
 void generateConfig(Object middleware) throws URISyntaxException;   
}

### 1.2.2 中间件配置策略工厂

package com.example.middlewaredeploy.build;   
import java.net.URISyntaxException;   
/\*\*   
 \* @project middlewareDeploy   
 \* @description 中间件配置策略工厂   
 \* @author capture or new   
 \* @date 2023/7/1 09:07:24   
 \* @version 1.0   
 \*/   
public class MiddlewareConfigStrategyFactory {   
 // 聚合策略类对象   
 private MiddlewareConfigStrategy strategy;   
 public MiddlewareConfigStrategyFactory(MiddlewareConfigStrategy strategy) {   
 this.strategy = strategy;   
 }   
 public void getStrategy(Object middleware) throws URISyntaxException {   
 strategy.generateConfig(middleware);   
 }   
}

### 1.2.3 minio配置类

package com.example.middlewaredeploy.build.shellBuilder;   
import com.example.middlewaredeploy.build.MiddlewareConfigStrategy;   
import com.example.middlewaredeploy.constant.MinioShell;   
import com.example.middlewaredeploy.constant.NacosShell;   
import com.example.middlewaredeploy.entity.dto.MiddlewareDto;   
import com.example.middlewaredeploy.utils.WriteYamlUtils;   
import static com.example.middlewaredeploy.constant.docker.DockerFilePath.DOCKER\_COMPOSE\_PATH;   
/\*\*   
 \* @project middlewareDeploy   
 \* @description minio配置类   
 \* @author capture or new   
 \* @date 2023/7/1 09:05:25   
 \* @version 1.0   
 \*/   
public class NacosConfigStrategy implements MiddlewareConfigStrategy {   
 @Override   
 public void generateConfig(Object middleware) {   
 // 转为对应的对象   
 MiddlewareDto middlewareDto = (MiddlewareDto) middleware;   
// 格式化   
 String nacosShell = String.format(NacosShell.NACOS\_SHELL.getCommand(),   
 middlewareDto.getMysqlServiceHost(),   
 middlewareDto.getNacosDatabase(),   
 middlewareDto.getUserName(),   
 middlewareDto.getPassword());   
 // 写入 docker compose中   
 WriteYamlUtils.writeAppend(DOCKER\_COMPOSE\_PATH.getFilePath(),nacosShell);   
 }   
}

## 1.3 以下常量等：

### 1.3.1 Dcokerfile 部署脚本

package com.example.middlewaredeploy.constant;   
/\*\*   
 \* @project middlewareDeploy   
 \* @description Dcokerfile 部署脚本   
 \* @author capture or new   
 \* @date 2023/7/24 16:09:58   
 \* @version 1.0   
 \*/   
public enum DockerfileShell implements Adapter {   
 DOCKERFILE\_SHELL("#基础镜像使用\n" +   
 "FROM java:8\n" +   
 "#作者\n" +   
 "MAINTAINER %s\n" +   
 "#VOLUME 指定临时文件目录为/tmp，在主机本目录之下创建一个临时文件并连接到容器的/tmp\n" +   
 "VOLUME /tmp\n" +   
 "#将jar包添加到容器中并更名为 \n" +   
 "ADD %s %s\n" +   
 "#运行jar包\n" +   
 "RUN bash -c 'touch /%s'\n" +   
 "ENTRYPOINT [\"java\",\"-jar\",\"/%s\"]\n" +   
 "#暴露端口 和springboot工程中的yml文件中的端口进行一致 注意：记得开放安全组\n" +   
 "EXPOSE %s"),   
 DOCKERFILE\_BUILDER\_SHELL("docker build -t %s %s"),   
 DOCKERFILE\_RUN\_SHELL("docker run -d -p %s:%s %s"),   
 ;   
 private final String command;   
 DockerfileShell(String command) {   
 this.command = command;   
 }   
 @Override   
 public String getCommand() {   
 return command;   
 }   
}

### 1.3.2 docker 安装脚本

package com.example.middlewaredeploy.constant;   
public enum DockerShell implements Adapter{   
 DELETE\_DOCKER("sudo yum -y remove docker\*"),   
 YUM\_UTILS("sudo yum install -y yum-utils"),   
 ADD\_REPO("sudo yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo"),   
 DOCKER\_CE("sudo yum install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin"),   
 ENABLE\_DOCKER("sudo systemctl enable docker --now"),   
 DISPOSITION\_DOCKER("sudo mkdir -p /etc/docker"),   
 DOCKER\_MIRRORS("sudo tee /etc/docker/daemon.json <<-'EOF'\n" +   
 "{\n" +   
 " \"registry-mirrors\": [\"https://82m9ar63.mirror.aliyuncs.com\"],\n" +   
 " \"exec-opts\": [\"native.cgroupdriver=systemd\"],\n" +   
 " \"log-driver\": \"json-file\",\n" +   
 " \"log-opts\": {\n" +   
 " \"max-size\": \"100m\"\n" +   
 " },\n" +   
 " \"storage-driver\": \"overlay2\"\n" +   
 "}\n" +   
 "EOF"),   
 DAEMON\_RELOAD("sudo systemctl daemon-reload"),   
 RESTART\_DOCKER("sudo systemctl restart docker"),   
 DOCKER\_PS("docker ps"),   
 DOCKER\_COMPOSE("docker compose")   
 ;   
 private final String command;   
 DockerShell(String command) {   
 this.command = command;   
 }   
 @Override   
 public String getCommand() {   
 return command;   
 }   
}

### 1.3.3 minio 脚本

package com.example.middlewaredeploy.constant;   
/\*\*   
 \* @project middlewareDeploy   
 \* @description minio脚本   
 \* @author capture or new   
 \* @date 2023/7/24 16:09:58   
 \* @version 1.0   
 \*/   
public enum MinioShell implements Adapter {   
 MINIO\_SHELL(" minio:\n" +   
 " image: minio/minio:latest\n" +   
 " container\_name: minio\n" +   
 " command: server /data --console-address \":9090\" --address \":9000\" \n" +   
 " volumes:\n" +   
 " - ./minio-data:/data # 将 MinIO 数据目录映射到宿主机的 ./minio-data 目录\n" +   
 " ports:\n" +   
 " - \"9000:9000\"\n" +   
 " environment:\n" +   
 " - MINIO\_ROOT\_USER=%s \n" +   
 " - MINIO\_ROOT\_PASSWORD=%s"+   
 "\n"),   
 ;   
 private final String command;   
 MinioShell(String command) {   
 this.command = command;   
 }   
 @Override   
 public String getCommand() {   
 return command;   
 }   
}

### 1.3.4 mysql脚本

package com.example.middlewaredeploy.constant;   
public enum MysqlShell implements Adapter{   
 MYSQL\_SHELL(" mysql:\n" +   
 " image: mysql:%s\n" +   
 " restart: always\n" +   
 " environment:\n" +   
 " MYSQL\_ROOT\_PASSWORD: %s\n" +   
 " volumes:\n" +   
 " - ./mysql-data:/var/lib/mysql\n" +   
 " ports:\n" +   
 " - 3306:3306" +   
 "\n"),   
 VOLUMES(" mysqldata:"),   
 ;   
 private final String command;   
 MysqlShell(String command) {   
 this.command = command;   
 }   
 @Override   
 public String getCommand() {   
 return command;   
 }   
}

### 1.3.5 nacos脚本

package com.example.middlewaredeploy.constant;   
/\*\*   
 \* @project middlewareDeploy   
 \* @description minio脚本   
 \* @author capture or new   
 \* @date 2023/7/24 16:09:58   
 \* @version 1.0   
 \*/   
public enum NacosShell implements Adapter {   
 NACOS\_SHELL(" mall4cloud-nacos:\n" +   
 " image: nacos/nacos-server:v2.2.0-slim\n" +   
 " container\_name: mall4cloud-nacos\n" +   
 " restart: always\n" +   
 " ports:\n" +   
 " - 8848:8848\n" +   
 " - 9848:9848\n" +   
 " - 9849:9849\n" +   
 " environment:\n" +   
 " - JVM\_XMS=256m\n" +   
 " - JVM\_XMX=256m\n" +   
 " - MODE=standalone\n" +   
 " - PREFER\_HOST\_MODE=hostname\n" +   
 " - SPRING\_DATASOURCE\_PLATFORM=mysql\n" +   
 " - MYSQL\_SERVICE\_HOST=%s\n" +   
 " - MYSQL\_SERVICE\_DB\_NAME=%s\n" +   
 " - MYSQL\_SERVICE\_USER=%s\n" +   
 " - MYSQL\_SERVICE\_PASSWORD=%s\n" +   
 " volumes:\n" +   
 " - ./nacos/logs:/home/nacos/logs"),   
 ;   
 private final String command;   
 NacosShell(String command) {   
 this.command = command;   
 }   
 @Override   
 public String getCommand() {   
 return command;   
 }   
}

### 1.3.6 redis脚本

package com.example.middlewaredeploy.constant;   
/\*\*   
 \* @project middlewareDeploy   
 \* @description redis Shell脚本   
 \* @author capture or new   
 \* @date 2023/7/1 10:26:07   
 \* @version 1.0   
 \*/   
public enum RedisShell implements Adapter{   
   
 REDIS\_SHELL(" redis:\n" +   
 " image: redis:%s\n" +   
 " restart: always\n" +   
 " volumes:\n" +   
 " - ./redis-data:/data\n" +   
 " ports:\n" +   
 " - 6379:6379" +   
 "\n"),   
 ;   
 private final String command;   
 RedisShell(String command) {   
 this.command = command;   
 }   
 @Override   
 public String getCommand() {   
 return command;   
 }   
}

### 1.3.7 rocket 脚本

package com.example.middlewaredeploy.constant;   
public enum RocketmqShell implements Adapter{   
 ROCKETMQ\_SHELL(" mall4cloud-rocketmq-namesrv:\n" +   
 " image: apache/rocketmq:4.9.4\n" +   
 " container\_name: mall4cloud-rocketmq-namesrv\n" +   
 " restart: always\n" +   
 " ports:\n" +   
 " - 9876:9876\n" +   
 " volumes:\n" +   
 " - ./rocketmq/namesrv/logs:/home/rocketmq/logs\n" +   
 " - ./rocketmq/namesrv/store:/home/rocketmq/store\n" +   
 " environment:\n" +   
 " JAVA\_OPT\_EXT: \"-Duser.home=/home/rocketmq -Xms512M -Xmx512M -Xmn128M\"\n" +   
 " command: [\"sh\",\"mqnamesrv\"]\n" +   
 "\n" +   
 " mall4cloud-rocketmq-broker:\n" +   
 " image: apache/rocketmq:4.9.4\n" +   
 " container\_name: mall4cloud-rocketmq-broker\n" +   
 " restart: always\n" +   
 " ports:\n" +   
 " - 10909:10909\n" +   
 " - 10911:10911\n" +   
 " volumes:\n" +   
 " - ./rocketmq/broker/logs:/home/rocketmq/logs\n" +   
 " - ./rocketmq/broker/store:/home/rocketmq/store\n" +   
 " - ./rocketmq/broker/conf/broker.conf:/etc/rocketmq/broker.conf\n" +   
 " environment:\n" +   
 " JAVA\_OPT\_EXT: \"-Duser.home=/home/rocketmq -Xms512M -Xmx512M -Xmn128M -XX:-AssumeMP\"\n" +   
 " command: [\"sh\",\"mqbroker\",\"-c\",\"/etc/rocketmq/broker.conf\",\"-n\",\"mall4cloud-rocketmq-namesrv:9876\",\"autoCreateTopicEnable=true\"]\n" +   
 " depends\_on:\n" +   
 " - mall4cloud-rocketmq-namesrv\n" +   
 "\n" +   
 " mall4cloud-rocketmq-dashboard:\n" +   
 " image: apacherocketmq/rocketmq-dashboard:1.0.0\n" +   
 " container\_name: mall4cloud-rocketmq-dashboard\n" +   
 " restart: always\n" +   
 " ports:\n" +   
 " - 8180:8080\n" +   
 " environment:\n" +   
 " JAVA\_OPTS: \"-Drocketmq.namesrv.addr=mall4cloud-rocketmq-namesrv:9876 -Dcom.rocketmq.sendMessageWithVIPChannel=false\"\n" +   
 " depends\_on:\n" +   
 " - mall4cloud-rocketmq-namesrv"+   
 "\n"),   
 ;   
 private final String command;   
 RocketmqShell(String command) {   
 this.command = command;   
 }   
 @Override   
 public String getCommand() {   
 return command;   
 }   
}

### 1.3.8 elasticsearch脚本

package com.example.middlewaredeploy.constant;   
/\*\*   
 \* @project middlewareDeploy   
 \* @description elasticsearch脚本   
 \* @author capture or new   
 \* @date 2023/7/24 16:09:58   
 \* @version 1.0   
 \*/   
public enum ElasticsearchShell implements Adapter {   
 ELASTICSEARCH\_SHELL(" elasticsearch:\n" +   
 " image: docker.elastic.co/elasticsearch/elasticsearch:7.14.0\n" +   
 " container\_name: elasticsearch\n" +   
 " environment:\n" +   
 " - discovery.type=single-node\n" +   
 " - \"ES\_JAVA\_OPTS=-Xms512m -Xmx512m\" # 设置Elasticsearch的内存限制，根据需要进行调整\n" +   
 " user: \"0:0\"\n"+   
 " ports:\n" +   
 " - \"9200:9200\"\n" +   
 " - \"9300:9300\"\n" +   
 " volumes:\n" +   
 " - ./es-data:/usr/share/elasticsearch/data "+   
 "\n"),   
   
   
 KIBANA\_SHELL(" kibana:\n" +   
 " image: docker.elastic.co/kibana/kibana:7.14.0\n" +   
 " container\_name: kibana\n" +   
 " ports:\n" +   
 " - \"5601:5601\"\n" +   
 " environment:\n" +   
 " - ELASTICSEARCH\_HOSTS=http://elasticsearch:9200"+   
 "\n"),   
 ;   
 private final String command;   
 ElasticsearchShell(String command) {   
 this.command = command;   
 }   
 @Override   
 public String getCommand() {   
 return command;   
 }   
}

### 1.3.8 netty信息编码

package com.example.middlewaredeploy.constant.Command;   
import lombok.AllArgsConstructor;   
import lombok.Data;   
import lombok.NoArgsConstructor;   
/\*\*   
 \* @author beiming   
 \*/   
@AllArgsConstructor   
@NoArgsConstructor   
@Data   
public class Command {   
 /\*\*   
 \* 连接信息编码   
 \*/   
 private Integer code;   
 /\*\*   
 \* 昵称   
 \*/   
 private String nickname;   
   
}

### 1.3.9 信息类型

package com.example.middlewaredeploy.constant.Command;   
import lombok.AllArgsConstructor;   
import lombok.Getter;   
/\*\*   
 \* @author beiming   
 \*/   
@Getter   
@AllArgsConstructor   
public enum CommandType {   
 /\*\*   
 \* 建立连接   
 \*/   
 CONNECTION(10001),   
 ERROR(-1),   
 ;   
 private final Integer code;   
 public static CommandType match(Integer code){   
 for (CommandType value : CommandType.values()) {   
 if (value.getCode().equals(code)){   
 return value;   
 }   
 }   
 return ERROR;   
 }   
}

### 1.3.10 全局文档

package com.example.middlewaredeploy.constant.Command;  
import io.netty.channel.ChannelHandlerContext;   
/\*\*   
 \* @project middlewareDeploy   
 \* @description   
 \* @author capture or new   
 \* @date 2023/7/18 10:31:58   
 \* @version 1.0   
 \*/   
public class GlobalContext {   
 private static ChannelHandlerContext ctx;   
 public static synchronized void setContext(ChannelHandlerContext context) {   
 ctx = context;   
 }   
 public static synchronized ChannelHandlerContext getContext() {   
 return ctx;   
 }   
}

## 1.4 配置类：

### 1.4.1 dockerfile配置类

package com.example.middlewaredeploy.build.shellBuilder;   
import com.example.middlewaredeploy.build.MiddlewareConfigStrategy;   
import com.example.middlewaredeploy.constant.DockerfileShell;   
import com.example.middlewaredeploy.constant.MinioShell;   
import com.example.middlewaredeploy.entity.dto.MiddlewareDto;   
import com.example.middlewaredeploy.utils.WriteYamlUtils;   
import static com.example.middlewaredeploy.constant.docker.DockerFilePath.\*;   
/\*\*   
 \* @project middlewareDeploy   
 \* @description dockerfile配置类   
 \* @author capture or new   
 \* @date 2023/7/1 09:05:25   
 \* @version 1.0   
 \*/   
public class DockerfileConfigStrategy implements MiddlewareConfigStrategy {   
 @Override   
 public void generateConfig(Object middleware) {   
 // 转为对应的对象   
 MiddlewareDto middlewareDto = (MiddlewareDto) middleware;   
 String[] split = middlewareDto.getJarPath().split("\\\\");   
 String jarName = split[split.length-1];   
 System.out.println(split);   
 System.out.println(jarName);   
// 格式化   
 String dockerfile = String.format(DockerfileShell.DOCKERFILE\_SHELL.getCommand(),   
 middlewareDto.getAuthor(),jarName,jarName.toLowerCase(),jarName.toLowerCase(),jarName.toLowerCase(),middlewareDto.getExpose());   
 // 写入 docker compose中   
WriteYamlUtils.writeAppend(DOCKERFILE\_PATH.getFilePath(),dockerfile);   
 }   
}

### 1.4.2 Elasticsearch配置类

package com.example.middlewaredeploy.build.shellBuilder;   
import com.example.middlewaredeploy.build.MiddlewareConfigStrategy;   
import com.example.middlewaredeploy.constant.ElasticsearchShell;   
import com.example.middlewaredeploy.constant.MysqlShell;   
import com.example.middlewaredeploy.entity.dto.MiddlewareDto;   
import com.example.middlewaredeploy.utils.WriteYamlUtils;   
import static com.example.middlewaredeploy.constant.docker.DockerFilePath.DOCKER\_COMPOSE\_PATH;   
   
/\*\*   
 \* @project middlewareDeploy   
 \* @description Elasticsearch配置类   
 \* @author capture or new   
 \* @date 2023/7/1 09:05:25   
 \* @version 1.0   
 \*/   
public class ElasticsearchConfigStrategy implements MiddlewareConfigStrategy {   
 @Override   
 public void generateConfig(Object middleware) {   
 // 转为对应的对象   
 MiddlewareDto middlewareDto = (MiddlewareDto) middleware;   
 for (ElasticsearchShell value : ElasticsearchShell.values()) {   
 // 写入 docker compose中   
 WriteYamlUtils.writeAppend(DOCKER\_COMPOSE\_PATH.getFilePath(),value.getCommand());   
 }   
 }   
}

### 1.4.3 Nacos配置类

package com.example.middlewaredeploy.build.shellBuilder;   
import com.example.middlewaredeploy.build.MiddlewareConfigStrategy;   
import com.example.middlewaredeploy.constant.MinioShell;   
import com.example.middlewaredeploy.constant.NacosShell;   
import com.example.middlewaredeploy.entity.dto.MiddlewareDto;   
importcom.example.middlewaredeploy.utils.WriteYamlUtils;   
import static com.example.middlewaredeploy.constant.docker.DockerFilePath.DOCKER\_COMPOSE\_PATH;   
/\*\*   
 \* @project middlewareDeploy   
 \* @description Nacos配置类   
 \* @author capture or new   
 \* @date 2023/7/1 09:05:25   
 \* @version 1.0   
 \*/   
public class NacosConfigStrategy implements MiddlewareConfigStrategy {   
 @Override   
 public void generateConfig(Object middleware) {   
   
 // 转为对应的对象   
 MiddlewareDto middlewareDto = (MiddlewareDto) middleware;   
// 格式化   
 String nacosShell = String.format(NacosShell.NACOS\_SHELL.getCommand(),   
 middlewareDto.getMysqlServiceHost(),   
 middlewareDto.getNacosDatabase(),   
 middlewareDto.getUserName(),   
 middlewareDto.getPassword());   
 // 写入 docker compose中   
 WriteYamlUtils.writeAppend(DOCKER\_COMPOSE\_PATH.getFilePath(),nacosShell);   
 }   
}

### 1.4.4 redis配置类

package com.example.middlewaredeploy.build.shellBuilder;   
import com.example.middlewaredeploy.build.MiddlewareConfigStrategy;   
import com.example.middlewaredeploy.constant.RedisShell;   
import com.example.middlewaredeploy.entity.dto.MiddlewareDto;   
import com.example.middlewaredeploy.entity.vo.Redis;   
import com.example.middlewaredeploy.utils.WriteYamlUtils;   
import java.net.URISyntaxException;   
import static com.example.middlewaredeploy.constant.docker.DockerFilePath.DOCKER\_COMPOSE\_PATH;   
/\*\*   
 \* @project middlewareDeploy   
 \* @description redis配置类   
 \* @author capture or new   
 \* @date 2023/7/1 09:06:03   
 \* @version 1.0   
 \*/   
public class RedisConfigStrategy implements MiddlewareConfigStrategy {   
 @Override   
 public void generateConfig(Object middleware){   
 // 转为对应的对象   
// Redis redis = (Redis) middleware;   
 MiddlewareDto middlewareDto = (MiddlewareDto) middleware;   
// 格式化   
 String redisShell = String.format(RedisShell.REDIS\_SHELL.getCommand(), middlewareDto.getVersion());   
 // 写入 docker compose中   
WriteYamlUtils.writeAppend(DOCKER\_COMPOSE\_PATH.getFilePath(),redisShell);   
 }   
}

### 1.4.5 Rocketmq配置类

package com.example.middlewaredeploy.build.shellBuilder;   
import com.example.middlewaredeploy.build.MiddlewareConfigStrategy;   
import com.example.middlewaredeploy.constant.MinioShell;   
import com.example.middlewaredeploy.constant.RocketmqShell;   
import com.example.middlewaredeploy.entity.dto.MiddlewareDto;   
import com.example.middlewaredeploy.utils.WriteYamlUtils;   
import static com.example.middlewaredeploy.constant.docker.DockerFilePath.DOCKER\_COMPOSE\_PATH;   
/\*\*   
 \* @project middlewareDeploy   
 \* @description Rocketmq配置类   
 \* @author capture or new   
 \* @date 2023/7/1 09:05:25   
 \* @version 1.0   
 \*/   
public class RocketmqConfigStrategy implements MiddlewareConfigStrategy {   
 @Override   
 public void generateConfig(Object middleware) {   
 // 转为对应的对象   
 MiddlewareDto middlewareDto = (MiddlewareDto) middleware;   
 // 写入 docker compose中   
 WriteYamlUtils.writeAppend(DOCKER\_COMPOSE\_PATH.getFilePath(), RocketmqShell.ROCKETMQ\_SHELL.getCommand());   
 }   
}

# 2、实体类：

## 2.1 中间件

package com.example.middlewaredeploy.entity.dto;   
import lombok.AllArgsConstructor;   
import lombok.Data;   
import lombok.NoArgsConstructor;   
import lombok.ToString;   
/\*\*   
 \* @project middlewareDeploy   
 \* @description   
 \* @author capture or new   
 \* @date 2023/7/17 11:45:33   
 \* @version 1.0   
 \*/   
@Data   
@NoArgsConstructor   
@AllArgsConstructor   
@ToString   
public class MiddlewareDto {   
 private Integer id;   
 private Integer parentId;   
 private String name;   
 private String version;   
 private String userName;   
 private String password;   
 private String description;   
 private String imgUrl;   
 private Boolean isSelect;   
 // jar 包路径   
 private String jarPath;   
 //容器暴露端口   
 private String expose;   
 //作者   
 private String author;   
 // mysqlServiceHost   
 private String mysqlServiceHost;   
 private String nacosDatabase;   
}

## 2.2 分类：

package com.example.middlewaredeploy.entity.dto;   
import lombok.AllArgsConstructor;   
import lombok.Data;   
import lombok.NoArgsConstructor;   
import lombok.ToString;   
/\*\*   
 \* @project middlewareDeploy   
 \* @description 分类类   
 \* @author capture or new   
 \* @date 2023/7/17 09:56:36   
 \* @version 1.0   
 \*/   
@Data   
@AllArgsConstructor   
@NoArgsConstructor   
@ToString   
public class CategoryDto {   
 private Integer id;   
 private String timestamp;   
 private String title;   
 private String description;   
}

## 2.3 中间件mysql：

package com.example.middlewaredeploy.entity.vo;   
import com.example.middlewaredeploy.entity.model.BaseEntity;   
import lombok.AllArgsConstructor;   
import lombok.Data;   
import lombok.NoArgsConstructor;   
import lombok.ToString;   
@Data   
@NoArgsConstructor   
@AllArgsConstructor   
@ToString(callSuper = true)   
public class MySQL extends BaseEntity {   
 private String name = "MySQL";   
 private Integer isUserName = 0;   
 private Integer isPassword = 1;   
}

## 2.4 服务器

package com.example.middlewaredeploy.entity.vo;   
import lombok.AllArgsConstructor;   
import lombok.Data;   
import lombok.NoArgsConstructor;   
import lombok.ToString;   
/\*\*   
 \* @project middlewareDeploy   
 \* @description   
 \* @author capture or new   
 \* @date 2023/7/1 13:40:41   
 \* @version 1.0   
 \*/   
@AllArgsConstructor   
@Data   
@NoArgsConstructor   
@ToString   
public class ServerDispositionVo {   
 private String host;   
 private Integer port;   
 private String userName;   
 private String password;   
}

## 2.5 分类类

package com.example.middlewaredeploy.entity.dto;   
import lombok.AllArgsConstructor;   
import lombok.Data;   
import lombok.NoArgsConstructor;   
import lombok.ToString;   
/\*\*   
 \* @project middlewareDeploy   
 \* @description 分类类   
 \* @author capture or new   
 \* @date 2023/7/17 09:56:36   
 \* @version 1.0   
 \*/   
@Data   
@AllArgsConstructor   
@NoArgsConstructor   
@ToString   
public class CategoryDto {   
 private Integer id;   
 private String timestamp;   
 private String title;   
 private String description;   
}

## 2.6 基类

package com.example.middlewaredeploy.entity.model;   
import lombok.AllArgsConstructor;   
import lombok.Data;   
import lombok.NoArgsConstructor;   
import lombok.ToString;   
/\*\*   
 \* @project middlewareDeploy   
 \* @description 基类   
 \* @author capture or new   
 \* @date 2023/7/1 09:51:31   
 \* @version 1.0   
 \*/   
@Data   
@NoArgsConstructor   
@AllArgsConstructor   
public class BaseEntity {   
 private String userName;   
 private String password;   
 private String version;   
 private boolean isSelect;   
}

# 3、异常：

## 3.1 自定义异常

package com.example.middlewaredeploy.exception;  
import com.example.middlewaredeploy.result.ResultCodeEnum;   
import lombok.Data;   
@Data   
public class DeployException extends RuntimeException {   
 private Integer code;//状态码   
 private String msg;//描述信息   
 public DeployException(Integer code, String msg) {   
 super(msg);   
 this.code = code;   
 this.msg = msg;   
 }   
 /\*\*   
 \* 接收枚举类型对象   
 \* @param resultCodeEnum   
 \*/   
 public DeployException(ResultCodeEnum resultCodeEnum) {   
 super(resultCodeEnum.getMessage());   
 this.code = resultCodeEnum.getCode();   
 this.msg = resultCodeEnum.getMessage();   
 }   
 @Override   
 public String toString() {   
 return "ERPException{" +   
 "code=" + code +   
 ", msg='" + msg + '\'' +   
 '}';   
 }   
}

## 3.2 全局异常

package com.example.middlewaredeploy.exception;   
import com.example.middlewaredeploy.result.Result;   
import com.example.middlewaredeploy.result.ResultCodeEnum;   
import org.springframework.validation.BindingResult;   
import org.springframework.web.bind.MethodArgumentNotValidException;   
import org.springframework.web.bind.annotation.ExceptionHandler;   
import org.springframework.web.bind.annotation.RestControllerAdvice;   
import java.util.HashMap;   
@RestControllerAdvice   
public class GlobalExceptionHandler {   
   
 //全局异常处理，执行的方法   
 @ExceptionHandler(Exception.class)   
 public Result error(Exception e) {   
 e.printStackTrace();   
 return Result.fail().message("执行全局异常处理...");   
 }   
 //特定异常处理   
 @ExceptionHandler(ArithmeticException.class)   
 public Result error(ArithmeticException e) {   
 e.printStackTrace();   
 return Result.fail().message("执行特定异常处理...");   
 }   
 //自定义异常处理   
 @ExceptionHandler(DeployException.class)   
 public Result error(DeployException e) {   
 e.printStackTrace();   
 return Result.fail().code(e.getCode()).message(e.getMsg());   
 }   
 /\*\*   
 \* 执行参数校验异常   
 \* @param e   
 \* @return   
 \*/   
 @ExceptionHandler(value = MethodArgumentNotValidException.class) //使用Exception 表示所有异常都能处理   
 public Result handleVaildException(MethodArgumentNotValidException e)   
 {   
// 获取所有的异常   
 BindingResult bindingResult = e.getBindingResult();   
 HashMap<String, String> map = new HashMap<>();   
 bindingResult.getFieldErrors().forEach((fieldError)->   
 {   
 map.put(fieldError.getField(),fieldError.getDefaultMessage());   
 });   
 return Result.fail(map, ResultCodeEnum.VAILD\_EXCEPTION);   
 }   
}

# 4、结果集

## 4.1 结果类

package com.example.middlewaredeploy.result;   
import lombok.Data;   
@Data   
public class Result<T> {   
   
 private Integer code;//状态码   
 private String message;//返回信息   
 private T data;//数据   
 //私有化   
 private Result() {}   
 //封装返回是数据   
 public static <T> Result<T> build(T body, ResultCodeEnum resultCodeEnum) {   
 Result<T> result = new Result<>();   
 //封装数据   
 if(body != null) {   
 result.setData(body);   
 }   
 //状态码   
 result.setCode(resultCodeEnum.getCode());   
 //返回信息   
 result.setMessage(resultCodeEnum.getMessage());   
 return result;   
 }   
 //成功   
 public static<T> Result<T> ok() {   
 return build(null,ResultCodeEnum.SUCCESS);   
 }   
 public static<T> Result<T> ok(T data) {   
 return build(data,ResultCodeEnum.SUCCESS);   
 }   
 //失败   
 public static<T> Result<T> fail() {   
 return build(null,ResultCodeEnum.FAIL);   
 }   
 public static<T> Result<T> fail(T data) {   
 return build(data,ResultCodeEnum.FAIL);   
 }   
 public static<T> Result<T> fail(T data,ResultCodeEnum resultCodeEnum) {   
 return build(data,resultCodeEnum);   
 }   
 public Result<T> message(String msg){   
 this.setMessage(msg);   
 return this;   
 }   
 public Result<T> code(Integer code){   
 this.setCode(code);   
 return this;   
 }   
}

## 4.2 结果Code

package com.example.middlewaredeploy.result;   
import lombok.Getter;   
@Getter   
public enum ResultCodeEnum {   
 SUCCESS(200,"成功"),   
 FAIL(201, "失败"),   
 LOGIN\_ERROR(208,"认证失败"),   
 VAILD\_EXCEPTION(10001,"参数格式校验失败"),   
 ;   
 private Integer code;   
 private String message;   
 ivate ResultCodeEnum(Integer code, String message) {   
 this.code = code;   
 this.message = message;   
 }   
}

## 4.3 结果类型

package com.example.middlewaredeploy.result;   
import lombok.AllArgsConstructor;   
import lombok.Getter;   
@Getter   
@AllArgsConstructor   
public enum ResultType {   
 BASIC\_ERROR(90005,"服务端未兼容异常"),   
 NOT\_SUPPORTED\_CODE(90001,"不支持的消息CODE"),   
 NOT\_FIND\_TARGET(90002,"消息发送失败，发送消息前请指定接收对象"),  
 NOT\_REPEAT\_LOGIN(90003,"该用户已上线，请更换昵称后再换~"),   
 MESSAGE\_FORMAT\_ERROR(90004,"发送消息格式错误，请确认后在试"),   
 SERVICE\_CONNECTED(80001,"服务连接建立成功"),   
 GET\_ONLINE\_USERS(80002,"获取用户列表"),   
 JOIN\_GROUP\_SUCCESS(80003,"加入系统默认群聊成功~"),   
 CHAT\_MESSAGE(80004,"聊天消息"),   
 JOIN\_USER(80005,"新用户上线")   
 ;  
 private Integer code;   
 private String message;   
}

# 5、工具类

## 5.1 执行命令

package com.example.middlewaredeploy.utils;   
import com.example.middlewaredeploy.constant.Command.GlobalContext;   
import com.example.middlewaredeploy.nettyServer.MiddlewareServer;   
import com.jcraft.jsch.ChannelExec;   
import com.jcraft.jsch.JSch;   
import com.jcraft.jsch.Session;   
import io.netty.channel.ChannelHandlerContext;   
import io.netty.handler.codec.http.websocketx.TextWebSocketFrame;   
impot java.io.BufferedReader;   
import java.io.IOException;   
import java.io.InputStream;   
import java.io.InputStreamReader;   
public class CentosOperate {   
 /\*\*   
 \* @author CaptureOrNew   
 \* @description 执行命令   
 \* @date 15:02:42 2023/7/16   
 \* @param session   
 \* @param command   
 \* @return java.lang.String   
 \*\*/   
 public static String executeCommand(Session session, String command) throws Exception {   
 io.netty.channel.Channel channel1 = MiddlewareServer.channels.get("connection");   
 ChannelHandlerContext ctx = GlobalContext.getContext();   
 ChannelExec channel = (ChannelExec) session.openChannel("exec");   
 channel.setCommand(command);   
 channel.setErrStream(System.err); // 转入错误输出流   
 InputStream inputStream = channel.getInputStream();   
 channel.connect();   
 BufferedReader reader = new BufferedReader(new InputStreamReader(inputStream));   
 StringBuilder output = new StringBuilder();   
 String line;   
 while ((line = reader.readLine()) != null) {   
 System.out.println(line);   
 // 写入到 netty 管道中   
 channel1.writeAndFlush(new TextWebSocketFrame(line));   
 output.append(line).append(System.lineSeparator());   
 System.out.flush();   
 }   
 channel.disconnect();   
 return output.toString();   
 }   
 public static boolean isCommandAvailable(Session session, String command) throws Exception {   
 // 创建执行通道   
 ChannelExec channel = (ChannelExec) session.openChannel("exec");   
 // 设置执行的命令   
 channel.setCommand("command -v " + command + " >/dev/null 2>&1 && echo \"Found\" || echo \"Not Found\"");   
 // 获取命令执行的输出流   
 InputStream inputStream = channel.getInputStream();   
 // 连接通道   
 channel.connect();   
 // 读取命令执行的输出   
 BufferedReader reader = new BufferedReader(new InputStreamReader(inputStream));   
 String line = reader.readLine();   
 // 关闭通道   
 channel.disconnect();   
 // 判断命令是否可用   
 return line.equals("Found");   
 }   
}

## 5.2 部署中间件工具类

package com.example.middlewaredeploy.utils;   
import com.example.middlewaredeploy.constant.Command.GlobalContext;   
import com.example.middlewaredeploy.handler.WebSocketHandler;   
import com.example.middlewaredeploy.nettyServer.MiddlewareServer;   
import com.example.middlewaredeploy.result.Result;   
import com.jcraft.jsch.Channel;   
import com.jcraft.jsch.ChannelExec;   
import com.jcraft.jsch.JSchException;   
import com.jcraft.jsch.Session;   
import io.netty.channel.ChannelHandlerContext;   
import io.netty.handler.codec.http.websocketx.TextWebSocketFrame;   
import java.io.BufferedReader;   
import java.io.IOException;   
import java.io.InputStream;   
import java.io.InputStreamReader;   
/\*\*   
 \* @project middlewareDeploy   
 \* @description 部署中间件   
 \* @author capture or new   
 \* @date 2023/7/14 19:51:01   
 \* @version 1.0   
 \*/   
public class deployMiddleware {   
 public static void deploy(Session session, String command) throws IOException {   
 io.netty.channel.Channel channel1 = MiddlewareServer.channels.get("connection");   
// 开始执行   
 InputStream inputStream = null;   
 InputStream errorStream = null;   
 Channel channel = null;   
 try {   
 // 打开通道   
 channel = session.openChannel("exec");   
 // 设置命令   
 ((ChannelExec) channel).setCommand(command);   
 // 获取命令执行的输出流   
 inputStream = channel.getInputStream();   
// BufferedReader inputReader = new BufferedReader(new InputStreamReader(inputStream));   
 // 连接通道   
 channel.connect();   
// 获取错误输出流   
 errorStream = ((ChannelExec) channel).getErrStream();   
 BufferedReader errorReader = new BufferedReader(new InputStreamReader(errorStream));   
// 读取命令执行的输出和错误输出   
 byte[] buffer = new byte[1024];   
 StringBuilder output = new StringBuilder();   
 while (true) {   
 while (inputStream.available() > 0) {   
 int bytesRead = inputStream.read(buffer, 0, 1024);   
 if (bytesRead < 0) {   
 break;   
 }   
 String line = new String(buffer, 0, bytesRead);   
 System.out.print(line); // 输出到控制台   
// 写入到 netty 管道中   
 channel1.writeAndFlush(new TextWebSocketFrame(line));   
 output.append(line); // 保存到变量中   
 }   
 // 逐行读取错误输出流   
 String errorLine;   
 while ((errorLine = errorReader.readLine()) != null) {   
 System.out.println(errorLine); // 输出到标准输出流   
 // 写入到 netty 管道中   
 channel1.writeAndFlush(new TextWebSocketFrame(errorLine));   
 output.append(errorLine); // 保存到变量中   
 }   
 if (channel.isClosed()) {   
 if (inputStream.available() > 0 || errorStream.available() > 0) {   
 continue;   
 }   
 break;   
 }   
 }   
 } catch (JSchException | IOException e) {   
 e.printStackTrace();   
 } finally {   
 // 关闭输入流、错误输出流、通道和会话   
 inputStream.close();   
 errorStream.close();   
 channel.disconnect();   
// session.disconnect();   
 }   
 }   
}

## 5.3 部署docker和compose 工具类

package com.example.middlewaredeploy.utils;   
import com.example.middlewaredeploy.constant.DockerShell;   
import com.jcraft.jsch.JSch;   
import com.jcraft.jsch.Session;   
import java.util.ArrayList;   
import java.util.Arrays;   
import static com.example.middlewaredeploy.utils.CentosOperate.executeCommand;   
/\*\*   
 \* @project middlewareDeploy   
 \* @description 部署docker和compose   
 \* @author capture or new   
 \* @date 2023/7/14 19:41:10   
 \* @version 1.0   
 \*/   
public class DocerAndComposeDeploy {   
 public static void deployDockerCompose(Session session){   
 try {   
 // 执行远程命令   
 ArrayList<DockerShell> dockerShells = new ArrayList<>(Arrays.asList(DockerShell.values()));   
 for (DockerShell dockerShell : dockerShells) {   
 executeCommand(session, dockerShell.getCommand());   
 }   
 } catch (Exception e) {   
 e.printStackTrace();   
 }   
 }   
}

## 5.4 判断指定文件夹是否存在

package com.example.middlewaredeploy.utils;   
import com.jcraft.jsch.Channel;   
import com.jcraft.jsch.ChannelExec;   
import com.jcraft.jsch.JSchException;   
import com.jcraft.jsch.Session;   
import java.io.IOException;   
import java.io.InputStream;   
/\*\*   
 \* @project middlewareDeploy   
 \* @description 判断指定文件夹是否存在   
 \* @author capture or new   
 \* @date 2023/7/14 20:27:28   
 \* @version 1.0   
 \*/   
public class FilesIfexist {   
 public static boolean IsNotExist(Session session,String folderPath){   
 annelExec channel = null;   
 try {   
 // 检查文件夹是否存在   
// String folderPath = "/opt/dockerCompose"; // 要检查的文件夹路径   
 String checkCommand = "test -d " + folderPath + " && echo exists || echo doesnotexist";   
 channel = (ChannelExec) session.openChannel("exec");   
 channel.setCommand(checkCommand);   
 channel.setErrStream(System.err);   
 channel.connect();   
 InputStream inputStream = channel.getInputStream();   
 byte[] buffer = new byte[1024];   
 int bytesRead;   
 StringBuilder output = new StringBuilder();   
 while ((bytesRead = inputStream.read(buffer)) != -1) {   
 output.append(new String(buffer, 0, bytesRead));   
 }   
 channel.disconnect();   
 // 解析输出并判断文件夹是否存在   
 boolean folderExists = output.toString().trim().equals("exists");   
 System.out.println("文件夹存在: " + folderExists);   
 return folderExists;   
 } catch (JSchException | IOException e) {   
 e.printStackTrace();   
 }   
 return false;   
 }   
}

## 5.5 获取resources目录下的文件路径 也就是target 下的

package com.example.middlewaredeploy.utils;   
import com.example.middlewaredeploy.demo1;   
import java.io.InputStream;   
import java.net.URL;   
/\*\*   
 \* @project middlewareDeploy   
 \* @description 获取resources目录下的文件路径 也就是target 下的   
 \* @author capture or new   
 \* @date 2023/7/19 13:13:14   
 \* @version 1.0   
 \*/   
public class GetResourcesFilePath {   
 public static String getPath(String resourcesPath){   
 // 使用ClassLoader加载资源文件   
 URL resourceURL = demo1.class.getClassLoader().getResource(resourcesPath);   
 if (resourceURL == null) {   
 return "Resource file not found: " + "docker/docker-compose.yaml";   
 }   
 return resourceURL.getPath();   
 }   
}

## 5.6 读取Json中的数据

package com.example.middlewaredeploy.utils;   
import com.alibaba.fastjson2.JSON;   
import com.alibaba.fastjson2.TypeReference;   
import java.io.BufferedReader;   
import java.io.IOException;   
import java.io.InputStream;   
import java.io.InputStreamReader;   
import java.nio.charset.StandardCharsets;   
import java.util.List;   
/\*\*   
 \* @project middlewareDeploy   
 \* @description 读取Json中的数据   
 \* @author capture or new   
 \* @date 2023/7/17 10:35:08   
 \* @version 1.0   
 \*/   
public class JsonFileReader {   
 public static <T> List<T> readJsonFile(String filePath, TypeReference<List<T>> typeReference) {   
 try {   
 InputStream inputStream = JsonFileReader.class.getClassLoader().getResourceAsStream(filePath);   
 BufferedReader reader = new BufferedReader(new InputStreamReader(inputStream, StandardCharsets.UTF\_8));   
 StringBuilder content = new StringBuilder();   
 String line;   
 while ((line = reader.readLine()) != null) {   
 content.append(line);   
 }   
 String jsonContent = content.toString();   
 return JSON.parseObject(jsonContent, typeReference);   
 } catch (IOException e) {   
 e.printStackTrace();   
 }  
 return null;   
 }   
}

## 5.7 获取session

package com.example.middlewaredeploy.utils;   
import com.example.middlewaredeploy.constant.ServerDisposition;   
import com.example.middlewaredeploy.entity.vo.ServerDispositionVo;   
import com.example.middlewaredeploy.exception.DeployException;   
import com.jcraft.jsch.\*;   
import lombok.Data;   
/\*\*   
 \* @project middlewareDeploy   
 \* @description 获取session   
 \* @author capture or new   
 \* @date 2023/7/27 15:35:56   
 \* @version 1.0   
 \*/   
@Data   
public class SessionUtils {   
 private Session session = null;   
 public SessionUtils(){   
 ServerDispositionVo serverDispositionVo = ServerDisposition.serverDispositionVo;   
 JSch jsch = new JSch();   
 try {   
 session = jsch.getSession(serverDispositionVo.getUserName(), serverDispositionVo.getHost(), serverDispositionVo.getPort());   
 session.setPassword( serverDispositionVo.getPassword());   
 session.setConfig("StrictHostKeyChecking", "no");   
 session.connect();   
 } catch (JSchException e) {   
 throw new DeployException(201,e.getMessage());   
 } catch (Exception e) {   
 throw new DeployException(201,e.getMessage());   
 }   
 }   
// //定义一个静态内部类   
// private static class SingletonHolder{   
// //在内部类中声明并初始化外部类的对象   
// private static SessionUtils sessionUtils = new SessionUtils();   
// }   
 //提供公共的访问方式   
// public static SessionUtils getInstance(){   
// return new SessionUtils();   
// }   
}

## 5.8 数据读写yaml

package com.example.middlewaredeploy.utils;   
import com.example.middlewaredeploy.constant.MysqlShell;   
import com.example.middlewaredeploy.demo1;   
import org.yaml.snakeyaml.DumperOptions;   
import org.yaml.snakeyaml.Yaml;   
import java.io.\*;   
import java.net.URISyntaxException;   
import java.net.URL;   
import java.nio.file.Paths;   
import java.util.Arrays;   
import java.util.HashMap;   
import java.util.LinkedHashMap;   
import java.util.Map;   
/\*\*   
 \* @project middlewareDeploy   
 \* @description 数据读写yaml   
 \* @author capture or new   
 \* @date 2023/7/1 10:53:50   
 \* @version 1.0   
 \*/   
public class WriteYamlUtils {   
 /\*\*   
 \* @author CaptureOrNew   
 \* @description 重构yaml文件 加入新的配置   
 \* @param path yaml文件位置   
 \* @param map 配置   
 \* @param level 配置的层级 存放位置   
 \* @param dispositionName 配置名称   
 \*\*/   
 public static void reconstitution(String path, Map<String, Object> map, String level, String dispositionName) {   
 // 从YAML文件中加载数据到Java对象中   
 InputStream input = null;   
 try {   
 input = new FileInputStream(path);   
 Yaml yaml = new Yaml();   
 Map<String, Object> obj = yaml.load(input);   
 // 获取现有的services部分   
 Map<String, Object> services = (Map<String, Object>) obj.get("services");   
 if (services == null) {   
 services = new LinkedHashMap<>();   
 obj.put("services", services);   
 }   
 // 获取现有的volumes部分   
 Map<String, Object> volumes1 = (Map<String, Object>) obj.get("volumes");   
 if (volumes1 == null) {   
 volumes1 = new LinkedHashMap<>();   
 obj.put("volumes", volumes1);   
 }   
 Map<String, Object> yamlmap = null;   
 Map<String, Object> yamlmap2 = null;   
 yamlmap = obj;   
 System.out.println(yamlmap + "======");   
 yamlmap2 = obj;   
 if (level != null) {   
 String[] keys = level.split("\\.");   
 if (keys.length == 1 && dispositionName == null) {   
 // 在没有key的时候就用原来的key   
 dispositionName = keys[0];   
 } else {   
 for (String key : keys) {   
 yamlmap2 = (Map<String, Object>) yamlmap2.get(key);   
 }   
 }   
 }   
 yamlmap2.put(dispositionName, map);   
 input.close();   
 System.out.println(yamlmap);   
 File file = new File(path);   
 boolean delete = file.delete();   
 System.out.println(delete + "yaml已经被删除");   
 if (delete) {   
 // 将修改后的Java对象写回YAML格式   
 write(path, yamlmap);   
 }   
 } catch (Exception e) {   
 throw new RuntimeException(e);   
 }   
 }  
 public static void write(String filePath, String shell) {   
 // 将配置写入到文件中   
 try (FileWriter writer = new FileWriter(filePath)) {   
 writer.write(shell);   
 } catch (IOException e) {   
 e.printStackTrace();   
 }   
 }  
 public static void writeAppend(String filePath, String shell) {   
 // 将配置写入到文件中   
 try (FileWriter writer = new FileWriter(filePath, true)) {   
 writer.write(shell);   
 } catch (IOException e) {   
 e.printStackTrace();   
 }   
 } public static void write(String path, Map<String, Object> yamlData) {   
 final DumperOptions options = new DumperOptions();   
 options.setDefaultFlowStyle(DumperOptions.FlowStyle.BLOCK);   
 options.setDefaultScalarStyle(DumperOptions.ScalarStyle.PLAIN);   
 // 将Java对象转换为YAML格式并写入文件： true 表示是否以追加的形式进行写入   
 try (FileWriter writer = new FileWriter(path, true)) {   
 Yaml yaml = new Yaml(options);   
 yaml.dump(yamlData, writer);   
 } catch (IOException e) {   
 e.printStackTrace();   
 }   
 }   
}

## 5.9 文件上传

ackage com.example.middlewaredeploy.utils;   
import cm.example.middlewaredeploy.constant.ServerDisposition;   
import com.example.middlewaredeploy.entity.vo.ServerDispositionVo;   
import com.example.middlewaredeploy.exception.DeployException;   
import com.jcraft.jsch.\*;   
import java.io.File;   
   
import static com.example.middlewaredeploy.constant.docker.DockerFilePath.DOCKER\_COMPOSE\_PATH;   
import static com.example.middlewaredeploy.utils.CentosOperate.executeCommand;   
import static com.example.middlewaredeploy.utils.CentosOperate.isCommandAvailable;   
/\*\*   
 \* @project middlewareDeploy   
 \* @description 上传文件   
 \* @author capture or new   
 \* @date 2023/7/1 13:44:38   
 \* @version 1.0   
 \*/   
public class UploadFileUtils {   
// public static boolean uploadFile(String localFilePath,String remoteFilePath) throws JSchException, SftpException {   
// System.out.println(ServerDisposition.serverDispositionVo);   
// ServerDispositionVo serverDispositionVo = ServerDisposition.serverDispositionVo;   
//   
// boolean uploadSuccess = upload(localFilePath, remoteFilePath, serverDispositionVo.getHost(), serverDispositionVo.getPort(),   
// serverDispositionVo.getUserName(), serverDispositionVo.getPassword());   
//   
//   
// if (uploadSuccess) {   
// return true;   
// } else {   
// return false;   
// }   
// }   
//   
//   
// public static boolean upload(String localFilePath, String remoteFilePath, String host, int port, String username, String password) throws JSchException, SftpException {   
//   
// JSch jsch = new JSch();   
// Session session = null;   
// ChannelSftp channel = null;   
// try {   
// session = jsch.getSession(username, host, port);   
// session.setPassword(password);   
// session.setConfig("StrictHostKeyChecking", "no");   
// session.connect();   
//   
// channel = (ChannelSftp) session.openChannel("sftp");   
// channel.connect();   
//   
// System.out.println("mkdir "+remoteFilePath);   
//   
//// 判断文件夹是否存在   
// if(!FilesIfexist.IsNotExist(session)){   
// System.out.println("文件夹不存在");   
// // 创建文件夹   
// CentosOperate.executeCommand(session,"mkdir "+remoteFilePath);   
// }   
//   
// // 上传compose 文件   
// channel.put(localFilePath, remoteFilePath);   
//   
// channel.disconnect();   
//   
//// 判断docker 和 docker compose是否安装 false 重新安装 true直接部署   
// if(!isCommandAvailable(session,"docker")){   
// DockerAndComposeDeploy.deployDockerCompose(session);   
// }   
//   
// if(!isCommandAvailable(session,"docker compose")){   
// executeCommand(session, "sudo yum install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin");   
// }   
//   
// //执行docker compose 部署命令   
// deployMiddleware.deploy(session,"docker compose -f "+remoteFilePath+"/docker-compose.yaml up -d");   
//   
// return true;   
// } catch (JSchException | SftpException e) {   
// throw new DeployException(201,e.getMessage());   
// } catch (Exception e) {   
// throw new DeployException(201,e.getMessage());   
// }   
// }   
   
   
 public static void uploadFileV2(String localFilePath, String remoteFilePath){   
// SessionUtils instance = SessionUtils.getInstance();   
 SessionUtils sessionUtils = new SessionUtils();   
 ChannelSftp channel = null;   
 try{   
 channel = (ChannelSftp) sessionUtils.getSession().openChannel("sftp");   
 channel.connect();   
 System.out.println("mkdir "+remoteFilePath);   
 // 判断文件夹是否存在   
 if(!FilesIfexist.IsNotExist(sessionUtils.getSession(),remoteFilePath)){   
 System.out.println("文件夹不存在");   
 // 创建文件夹   
 CentosOperate.executeCommand(sessionUtils.getSession(),"mkdir "+remoteFilePath);   
 }   
 // 上传compose 文件   
 channel.put(localFilePath, remoteFilePath);   
 System.out.println("上传成功 ");   
 channel.disconnect();   
 }catch (Exception e){   
 throw new DeployException(201,e.getMessage());   
 }finally {   
 sessionUtils.getSession().disconnect();   
 }   
   
 }   
 // 递归上传文件夹   
 public static void uploadFolder(Session session, ChannelSftp channel, String localFolderPath, String remoteFolderPath) throws SftpException {   
 File localFolder = new File(localFolderPath);   
 if (localFolder.isDirectory()) {   
   
 // 判断文件夹是否存在   
 if (!FilesIfexist.IsNotExist(session,remoteFolderPath)) {   
 System.out.println("文件夹不存在");   
 // 创建文件夹   
 // 在远程服务器上创建相应的文件夹   
 channel.mkdir(remoteFolderPath);   
 }   
 // 遍历本地文件夹中的文件和子文件夹   
 File[] files = localFolder.listFiles();   
 if (files != null) {   
 for (File file : files) {   
 if (file.isFile()) {   
 // 上传文件   
 channel.put(file.getAbsolutePath(), remoteFolderPath + "/" + file.getName(), ChannelSftp.OVERWRITE);   
 } else if (file.isDirectory()) {   
 // 递归上传子文件夹   
 uploadFolder(session,channel, file.getAbsolutePath(), remoteFolderPath + "/" + file.getName());   
 }   
 }   
 }   
 }else{   
 // 判断文件夹是否存在   
 if (!FilesIfexist.IsNotExist(session,remoteFolderPath)) {   
 System.out.println("文件夹不存在");   
 // 创建文件夹   
 // 在远程服务器上创建相应的文件夹   
 channel.mkdir(remoteFolderPath);   
 }   
 channel.put(localFolderPath, remoteFolderPath);   
 }   
 }   
}

# 6、service

## 6.1 部署service 接口

package com.example.middlewaredeploy.service;   
import com.example.middlewaredeploy.entity.dto.MiddlewareDto;   
import com.example.middlewaredeploy.entity.vo.AlwaysResponse;   
import com.example.middlewaredeploy.entity.vo.ServerDispositionVo;   
import com.example.middlewaredeploy.result.Result;   
import com.jcraft.jsch.JSchException;   
import com.jcraft.jsch.SftpException;   
import java.util.List;   
import java.util.Map;

public interface DeployService {   
 void deploy(AlwaysResponse alwaysResponse) throws IllegalAccessException, JSchException, SftpException;   
 String connect();   
 void deployV2(List<MiddlewareDto> middlewares) throws Exception;   
 void deployService(List<MiddlewareDto> middlewares) throws Exception;   
}

## 6.2 中间件service

package com.example.middlewaredeploy.service;   
import com.example.middlewaredeploy.entity.dto.CategoryDto;   
import com.example.middlewaredeploy.entity.dto.MiddlewareDto;   
import java.util.List;   
/\*\*   
 \* @project middlewareDeploy   
 \* @description 中间件service   
 \* @author capture or new   
 \* @date 2023/7/17 11:42:22   
 \* @version 1.0   
 \*/   
public interface MiddlewareService {   
 List<CategoryDto> getCategory();   
 List<MiddlewareDto> getMiddlewares(Integer id);   
// List<MiddlewareDto> getListById(Integer id);   
}

# 7、实现类

## 7.1 部署实现类

package com.example.middlewaredeploy.service.impl;   
import com.example.middlewaredeploy.build.MiddlewareConfigStrategy;   
import com.example.middlewaredeploy.build.MiddlewareConfigStrategyFactory;   
import com.example.middlewaredeploy.constant.DockerfileShell;   
import com.example.middlewaredeploy.entity.dto.MiddlewareDto;   
import com.example.middlewaredeploy.entity.vo.AlwaysResponse;   
import com.example.middlewaredeploy.entity.vo.ServerDispositionVo;   
import com.example.middlewaredeploy.exception.DeployException;   
import com.example.middlewaredeploy.service.DeployService;   
import com.example.middlewaredeploy.utils.\*;   
import com.jcraft.jsch.JSchException;   
import com.jcraft.jsch.SftpException;   
import org.springframework.stereotype.Service;   
import java.lang.reflect.Field;   
import java.util.List;   
import java.util.Map;   
import static com.example.middlewaredeploy.constant.docker.DockerFilePath.\*;   
import static com.example.middlewaredeploy.utils.CentosOperate.executeCommand;   
import static com.example.middlewaredeploy.utils.CentosOperate.isCommandAvailable;   
@Service   
public class DeployServiceImpl implements DeployService {   
 @Override   
 public void deploy(AlwaysResponse alwaysResponse) throws JSchException, SftpException {   
 Field[] declaredFields = alwaysResponse.getClass().getDeclaredFields();   
 for (Field declaredField : declaredFields) {   
 declaredField.setAccessible(true); // 设置字段可访问（包括私有字段）   
 // 判断其中的字段是否为空   
 Object value = null;   
 try {   
 value = declaredField.get(alwaysResponse);   
 if (value != null) {   
// TODO 通过指定的路径（bug 并且实体类的名称和 Builder策略类的 前缀要相同） 使用反射机制 创建对象   
 System.out.println(declaredField.getType().getSimpleName() + "ConfigStrategy");   
 Class<?> mysqlEntityClass = Class.forName("com.example.middlewaredeploy.build.shellBuilder." + declaredField.getType().getSimpleName() + "ConfigStrategy");   
 Object mysqlEntityInstance = mysqlEntityClass.newInstance();   
// 通过工厂传递对象   
 MiddlewareConfigStrategyFactory middlewareConfigStrategyFactory = new MiddlewareConfigStrategyFactory((MiddlewareConfigStrategy) mysqlEntityInstance);   
 middlewareConfigStrategyFactory.getStrategy(value);   
 }  
 } catch (Exception e) {   
 throw new DeployException(201, e.getMessage());   
 }   
 }   
 }   
 @Override   
 public void deployV2(List<MiddlewareDto> middlewares) throws Exception {   
 // 重置docker compose 文件   
 WriteYamlUtils.write(DOCKER\_COMPOSE\_PATH.getFilePath(), "version: '3.9'\n" +   
 "services:\n");   
 //写入文件   
 writeFile(middlewares);   
 //上传文件   
 UploadFileUtils.uploadFileV2(DOCKER\_COMPOSE\_PATH.getFilePath(),   
 "/opt/dockerCompose");   
// 获取session   
 SessionUtils sessionUtils = new SessionUtils();   
 // 判断docker 和 docker compose是否安装 false 重新安装 true直接部署   
 if (!isCommandAvailable(sessionUtils.getSession(), "docker")) {   
 DockerAndComposeDeploy.deployDockerCompose(sessionUtils.getSession());   
 }   
   
 if (!isCommandAvailable(sessionUtils.getSession(), "docker compose")) {   
 executeCommand(sessionUtils.getSession(), "sudo yum install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin");   
 }  
 //执行docker compose 部署命令   
 deployMiddleware.deploy(sessionUtils.getSession(), "docker compose -f /opt/dockerCompose/docker-compose.yaml up -d");   
 }   
 @Oveide   
 public void deployService(List<MiddlewareDto> middlewares) throws Exception {   
 // 重置dockerfile 文件   
 WriteYamlUtils.write(DOCKERFILE\_PATH.getFilePath(), "");   
 writeFile(middlewares); //写入文件   
 // 2、上传文件   
 //上传文件   
 UploadFileUtils.uploadFileV2(DOCKERFILE\_PATH.getFilePath(),   
 DOCKERFILE\_REMOTE\_PATH.getFilePath());   
 String jarName = "";   
 String expose = "";   
 //上传文件   
 for (MiddlewareDto middleware : middlewares) {   
 UploadFileUtils.uploadFileV2(middleware.getJarPath(),   
 DOCKERFILE\_REMOTE\_PATH.getFilePath());   
 String[] split = middleware.getJarPath().split("\\\\");   
 jarName = split[split.length-1];   
 expose = middleware.getExpose();   
 }   
 / 3、部署   
 // 获取session   
 SessionUtils sessionUtils = new SessionUtils();   
 // 判断docker 和 docker compose是否安装 false 重新安装 true直接部署   
 if (!isCommandAvailable(sessionUtils.getSession(), "docker")) {   
 DockerAndComposeDeploy.deployDockerCompose(sessionUtils.getSession());   
 }   
   
   
 //执行docker compose 部署命令   
 deployMiddleware.deploy(sessionUtils.getSession(),   
 String.format(DockerfileShell.DOCKERFILE\_BUILDER\_SHELL.getCommand(),   
 jarName.toLowerCase().replace(".jar", ""),   
 DOCKERFILE\_REMOTE\_PATH.getFilePath()));   
   
 deployMiddleware.deploy(sessionUtils.getSession(),   
 String.format(DockerfileShell.DOCKERFILE\_RUN\_SHELL.getCommand(),expose,expose,   
 jarName.toLowerCase().replace(".jar","")));   
 }   
 public void writeFile(List<MiddlewareDto> middlewares) {   
 if (middlewares != null) {   
 for (MiddlewareDto middleware : middlewares) {   
 try {   
 System.out.println(middleware.getIsSelect());   
 if (middleware.getIsSelect()) {   
 // 创建对象   
 Class<?> objectEntityClass = Class.forName("com.example.middlewaredeploy.build.shellBuilder." + middleware.getName() + "ConfigStrategy");   
 Object objectEntityInstance = objectEntityClass.newInstance();   
// 通过工厂传递对象   
 MiddlewareConfigStrategyFactory middlewareConfigStrategyFactory = new MiddlewareConfigStrategyFactory((MiddlewareConfigStrategy) objectEntityInstance);   
 middlewareConfigStrategyFactory.getStrategy(middleware);   
 }   
 } catch (Exception e) {   
 throw new RuntimeException(e);   
 }   
 }   
 }   
 }   
 @Override   
 public String connect() {   
 String connectMsg = ConnectTest.connect();   
 return connectMsg;   
 }  
}

## 7.2 中间件实现类

package com.example.middlewaredeploy.service.impl;   
import com.alibaba.fastjson2.TypeReference;   
import com.example.middlewaredeploy.entity.dto.CategoryDto;   
import com.example.middlewaredeploy.entity.dto.MiddlewareDto;   
import com.example.middlewaredeploy.service.MiddlewareService;   
import com.example.middlewaredeploy.utils.JsonFileReader;   
import org.springframework.stereotype.Service;   
import org.springframework.util.StringUtils;   
import java.util.List;   
import java.util.stream.Collectors;   
import static com.example.middlewaredeploy.constant.json.JsonFilePath.\*;   
/\*\*   
 \* @project middlewareDeploy   
 \* @description 实现类   
 \* @author capture or new   
 \* @date 2023/7/17 11:42:49   
 \* @version 1.0   
 \*/   
@Service   
public class MiddlewareServiceImpl implements MiddlewareService {   
 /\*\*   
 \* @author CaptureOrNew   
 \* @description 获取所有的分类   
 \* @date 15:19:22 2023/7/19   
 \* @return java.util.List<com.example.middlewaredeploy.entity.dto.CategoryDto>   
 \*\*/   
 @Override   
 public List<CategoryDto> getCategory() {   
 List<CategoryDto> categorys = JsonFileReader.readJsonFile(CATEGORY\_JSONFILE\_PATH.getFilePath(), new TypeReference<List<CategoryDto>>() {   
 });   
 return categorys;   
 }   
 /\*\*   
 \* @author CaptureOrNew   
 \* @description 返回对应类型的中间件   
 \* @date 15:18:18 2023/7/19   
 \* @param id   
 \* @return java.util.List<com.example.middlewaredeploy.entity.dto.MiddlewareDto>   
 \*\*/   
 @Override   
 public List<MiddlewareDto> getMiddlewares(Integer id) {   
 List<MiddlewareDto> middlewares = JsonFileReader.readJsonFile(MIDDLEWARE\_JSONFILE\_PATH.getFilePath(), new TypeReference<List<MiddlewareDto>>() {   
 });   
 // 过滤出 中间件类型的父Id 相同 也就是判别类型   
 middlewares = middlewares.stream().filter(middlewareDto -> {   
 return middlewareDto.getParentId() == id;   
 }).collect(Collectors.toList());   
 return middlewares;   
 }   
}

# 8、 controller控制层

package com.example.middlewaredeploy.web;   
import com.alibaba.fastjson2.JSON;   
import com.example.middlewaredeploy.constant.ServerDisposition;   
import com.example.middlewaredeploy.entity.dto.CategoryDto;   
import com.example.middlewaredeploy.entity.dto.MiddlewareDto;   
import com.example.middlewaredeploy.entity.vo.ServerDispositionVo;   
import com.example.middlewaredeploy.result.Result;   
import com.example.middlewaredeploy.service.DeployService;   
import com.example.middlewaredeploy.service.MiddlewareService;   
import com.example.middlewaredeploy.utils.CentosOperate;   
import org.springframework.beans.factory.annotation.Autowired;   
import org.springframework.stereotype.Controller;   
import org.springframework.ui.Model;   
import org.springframework.web.bind.annotation.\*;   
import java.io.\*;   
import java.util.List;   
import static com.example.middlewaredeploy.constant.json.JsonFilePath.\*;   
/\*\*   
 \* @project middlewareDeploy   
 \* @description 登录页面   
 \* @author capture or new   
 \* @date 2023/7/14 21:21:08   
 \* @version 1.0   
 \*/   
@Controller   
@RequestMapping("/admin")   
public class LoginMiddware {   
 @Autowired   
 DeployService deployService;   
 @Autowired   
 MiddlewareService middlewareService;   
 /\*\*   
 \* @author CaptureOrNew   
 \* @description 登录页面   
 \* @date 15:17:01 2023/7/19   
 \* @return java.lang.String   
 \*\*/   
 @GetMapping("/login")   
 public String login(){   
 return "login";   
 }   
 /\*\*\*   
 \* @author CaptureOrNew   
 \* @description 中间件页面   
 \* @date 15:16:44 2023/7/19   
 \* @param model   
 \* @return java.lang.String   
 \*\*/   
 @GetMapping("/middleware")   
 public String middleware(Model model) throws IOException {   
 // TODO 校验 是否连接   
 List<CategoryDto> categorys = middlewareService.getCategory();   
 model.addAttribute("categorys",JSON.toJSONString(categorys));   
 // 进入页面就获取初始数据   
 List<MiddlewareDto> middlewares = middlewareService.getMiddlewares(0);   
 model.addAttribute("middlewares",JSON.toJSONString(middlewares));   
 return "middleware";   
 }   
 /\*\*   
 \* @author CaptureOrNew   
 \* @description 点击连接就会触发 判断是否连接成功 成功就跳跳转到中间件页面中   
 \* @date 15:15:53 2023/7/19   
 \* @param serverDisposition   
 \* @return com.example.middlewaredeploy.result.Result   
 \*\*/   
 @PostMapping("/serverDisposition")   
 @ResponseBody   
 public Result serverDisposition(@RequestBody ServerDispositionVo serverDisposition) throws IOException {   
 ServerDisposition.serverDispositionVo = serverDisposition;   
 String connect = deployService.connect();   
 return Result.ok(connect);   
 }   
}

# 9、 netty服务端

## 9.1 netty服务端

package com.example.middlewaredeploy.nettyServer;   
import com.example.middlewaredeploy.handler.WebSocketHandler;   
import io.netty.bootstrap.ServerBootstrap;   
import io.netty.channel.Channel;   
import io.netty.channel.ChannelInitializer;   
import io.netty.channel.ChannelPipeline;   
import io.netty.channel.EventLoopGroup;   
import io.netty.channel.nio.NioEventLoopGroup;   
import io.netty.channel.socket.SocketChannel;   
import io.netty.channel.socket.nio.NioServerSocketChannel;   
import io.netty.handler.codec.http.HttpObjectAggregator;   
import io.netty.handler.codec.http.HttpServerCodec;   
import io.netty.handler.codec.http.websocketx.WebSocketServerProtocolHandler;   
import io.netty.handler.stream.ChunkedWriteHandler;   
import java.util.Map;   
import java.util.concurrent.ConcurrentHashMap;   
/\*\*   
 \* @project middlewareDeploy   
 \* @description netty服务端   
 \* @author capture or new   
 \* @date 2023/7/18 09:31:31   
 \* @version 1.0   
 \*/   
public class MiddlewareServer {   
 public static final Map<String, Channel> channels = new ConcurrentHashMap<>(1024);   
 public static void start() throws Exception {   
// UserHandler.execute();   
 EventLoopGroup boss = new NioEventLoopGroup();   
 EventLoopGroup worker = new NioEventLoopGroup();   
 // 绑定端口   
 ServerBootstrap bootstrap = new ServerBootstrap();   
 bootstrap.group(boss,worker)   
 .channel(NioServerSocketChannel.class)   
 .childHandler(new ChannelInitializer<SocketChannel>() {   
 @Override   
 protected void initChannel(SocketChannel socketChannel) throws Exception {   
 ChannelPipeline pipeline = socketChannel.pipeline();   
 // 添加http编码解码器   
 pipeline.addLast(new HttpServerCodec())   
 // 支持大数据流   
 .addLast(new ChunkedWriteHandler())   
 // 对http消息做聚合操作，FullHttpRequest、FullHttpResponse   
 .addLast(new HttpObjectAggregator(1024 \* 64))   
 // websocket   
 .addLast(new WebSocketServerProtocolHandler("/"))   
 // 自定义的handler   
 .addLast(new WebSocketHandler());   
 }   
 });   
 bootstrap.bind(9998).sync();   
 }   
}

## 9.2 连接

package com.example.middlewaredeploy.handler;   
import com.alibaba.fastjson2.JSON;   
import com.example.middlewaredeploy.constant.Command.Command;   
import com.example.middlewaredeploy.nettyServer.MiddlewareServer;   
import com.example.middlewaredeploy.result.Result;   
import io.netty.channel.ChannelHandlerContext;   
import io.netty.handler.codec.http.websocketx.TextWebSocketFrame;   
import io.netty.util.internal.StringUtil;   
import java.time.LocalDateTime;   
/\*\*   
 \* @author beiming   
 \*/   
public class ConnectionHandler {   
 public static void execute(ChannelHandlerContext ctx, Command command){   
 System.out.println("保存通道成功");   
 MiddlewareServer.channels.put("connection",ctx.channel());   
 MiddlewareServer.channels.get("connection").writeAndFlush(new TextWebSocketFrame("保存成功了"));   
 }   
}

## 9.3 分发

package com.example.middlewaredeploy.handler;   
import com.alibaba.fastjson2.JSON;   
import com.example.middlewaredeploy.constant.Command.Command;   
import com.example.middlewaredeploy.constant.Command.CommandType;   
import com.example.middlewaredeploy.constant.Command.GlobalContext;   
import com.example.middlewaredeploy.nettyServer.MiddlewareServer;   
import com.example.middlewaredeploy.result.Result;   
import com.example.middlewaredeploy.result.ResultType;   
import io.netty.channel.ChannelHandlerContext;   
import io.netty.channel.SimpleChannelInboundHandler;   
import io.netty.handler.codec.http.websocketx.TextWebSocketFrame;   
public class WebSocketHandler extends SimpleChannelInboundHandler<TextWebSocketFrame> {   
// public static ChannelHandlerContext ctxP = null;   
 @Override   
 protected void channelRead0(ChannelHandlerContext ctx, TextWebSocketFrame frame) throws Exception {   
   
 try{   
 System.out.println(frame.text());   
 Command command = JSON.parseObject(frame.text(), Command.class);   
 switch (CommandType.match(command.getCode())){   
 case CONNECTION -> ConnectionHandler.execute(ctx,command);   
 default -> ctx.channel().writeAndFlush(Result.fail(ResultType.NOT\_SUPPORTED\_CODE.getMessage()));   
 }   
 }catch (Exception e){   
// System.out.println(e.getMessage());   
 ctx.channel().writeAndFlush(Result.fail(ResultType.BASIC\_ERROR.getMessage()));   
 }   
 }   
}

# 11、前端：

## 11.1 登录

<!DOCTYPE html>   
<html>   
<head>   
 <!-- Import style -->   
 <link   
 rel="stylesheet"   
 href="../static/css/cdn.jsdelivr.net\_npm\_element-plus\_dist\_index.css"   
 />   
 <!-- Import Vue 3 -->   
 <script src="../static/js/cdn.jsdelivr.net\_npm\_vue@3"></script>   
 <!-- Import component library -->   
 <script src="../static/js/cdn.jsdelivr.net\_npm\_element-plus"></script>   
<#-- <script src="//cdn.jsdelivr.net/npm/element-plus@latest/dist/index.full.js"></script>-->   
 <#-- 引入 axios-->   
 <script src="../static/js/cdn.jsdelivr.net\_npm\_axios\_dist\_axios.min.js"></script>   
 <script src="../static/js/cdnjs.cloudflare.com\_ajax\_libs\_three.js\_r134\_three.min.js"></script>   
 <script src="../static/js/cdn.jsdelivr.net\_npm\_vanta\_dist\_vanta.clouds.min.js"></script>   
   
</head>   
<body style=" height: 100% ; width: 100%; margin: 0;padding: 0">   
<div id="app2" style=" width: 100vw; height: 100vh;">   
 <div id="app">   
 <span id="title">{{title}}</span>   
   
 <div id="form">   
 <el-form :model="form" label-width="60px">   
 <el-form-item label="host">   
 <el-input v-model="form.host" />   
 </el-form-item>   
 <el-form-item label="port">   
 <el-input-number   
 v-model="form.port"   
 class="mx-4"   
 :min="1"   
 :max="64500"   
 controls-position="right"   
 />   
 </el-form-item>   
 <el-form-item label="userName">   
 <el-input v-model="form.userName" />   
 </el-form-item>   
 <el-form-item label="password">   
 <el-input v-model="form.password"/>   
 </el-form-item>   
   
 <el-form-item>   
 <el-button type="primary" @click="onSubmit">connect</el-button>   
 </el-form-item>   
 </el-form>   
 </div>   
 </div>   
</div>   
<script>   
 VANTA.CLOUDS({   
 el: "#app2",   
 mouseControls: true,   
 touchControls: true,   
 gyroControls: false,   
 minHeight: 200.00,   
 minWidth: 200.00   
 })   
 const app = Vue.createApp({   
 setup() {   
 const title = Vue.ref("MiddleWareDeploy");   
 const form = Vue.reactive({   
 host:'',   
 port:22,   
 userName:'',   
 password:'',   
 })   
 const onSubmit = () => {   
 console.log("开始网络请求")   
 console.log(form)   
 axios.post('http://localhost:9999/admin/serverDisposition', form)   
 .then( ({data}) =>{   
 console.log(data);   
 // 处理成功响应   
 if(data.code === 200){   
 // ElMessage({   
 // message: data.data,   
 // type: 'success',   
 // })   
 // 进行页面跳转   
 window.location.href = 'http://localhost:9999/admin/middleware';   
 }   
 })   
 .catch(function (error) {   
 // 处理错误响应   
 // ElMessage.error(error)   
 });   
 }   
 return{   
 title,   
 form,   
 onSubmit   
 }   
 }   
 });   
   
 app.use(ElementPlus)   
 app.mount('#app');   
</script>   
<style>   
 html, body{   
 height: 100%;   
 width: 100%;   
 }   
 #title{   
 font-size: 48px;   
 font-family: Algerian;   
 position: fixed;   
 top: 40%;   
 left: 13%;   
 }   
 #form{   
 height: 500px;   
 width: 500px;   
 position: fixed;   
 top: 50%;   
 left: 13%;   
 }   
</style>   
   
</body>   
</html>

## 11.2 部署页面

<!DOCTYPE html>   
<html lang="en">   
<head>   
 <meta charset="UTF-8">   
 <title>Title</title>   
 <!-- Import style -->   
 <link   
 rel="stylesheet"   
 href="../static/css/cdn.jsdelivr.net\_npm\_element-plus\_dist\_index.css"   
 />   
 <!-- Import Vue 3 -->   
 <script src="../static/js/cdn.jsdelivr.net\_npm\_vue@3"></script>   
 <!-- Import component library -->   
 <script src="../static/js/cdn.jsdelivr.net\_npm\_element-plus"></script>   
 <#-- <script src="//cdn.jsdelivr.net/npm/element-plus@latest/dist/index.full.js"></script>-->   
 <#-- 引入 axios-->   
 <script src="../static/js/cdn.jsdelivr.net\_npm\_axios\_dist\_axios.min.js"></script>   
 <script src="../static/js/cdn.jsdelivr.net\_npm\_@element-plus\_icons-vue"></script>   
<#-- <script src="https://unpkg.com/vue-typed-js@2.0.0/dist/vue-typed-js.umd.min.js"></script>-->   
 <script src="../static/js/cdnjs.cloudflare.com\_ajax\_libs\_three.js\_r134\_three.min.js"></script>   
 <script src="../static/js/cdn.jsdelivr.net\_npm\_vanta\_dist\_vanta.clouds.min.js"></script>

</head>   
<body>   
<div id="app2" style=" width: 100vw; height: 100vh;">   
 <div id="app" style="width: 100vw; height: 100vh;">   
 <div class="common-layout">   
 <el-container style="width: 100vw; height: 100vh;">   
 <el-header height="100px">   
 <span id="title">{{title}}</span>   
 </el-header>   
   
 <el-container style="width: 100vw; height: 100vh;">   
 <el-aside>   
 <el-timeline>   
 <el-timeline-item   
 v-for="(item, index) in timelineItems"   
 :key="item.id"   
 :timestamp="item.timestamp"   
 placement="top"   
 >   
 <el-card   
 :class="{ 'selected': selectedItem === item.id }"   
 @click="selectItem(item.id)"   
 class="custom-card"   
 >   
 <h4>{{ item.title }}</h4>   
 <p>{{ item.description }}</p>   
 </el-card>   
 </el-timeline-item>   
 </el-timeline>   
 </el-aside>   
   
 <el-main>   
 <el-page-header :icon="null">   
 <template #content>   
 <div class="flex items-center">   
 <el-avatar   
 :size="100"   
 class="mr-3"   
 src="https://cube.elemecdn.com/0/88/03b0d39583f48206768a7534e55bcpng.png"   
 />   
 <!-- 正确使用方式 -->   
 </div>   
 </template>   
 <template #extra>   
 <div class="flex items-center">   
 <el-badge is-dot class="item" style="padding-right: 20px">   
 <el-button @click="dialogFormVisible = true" class="share-button" :icon="Share" type="primary">   
 日志   
 </el-button>   
 </el-badge>   
 <el-button>Print</el-button>   
 <el-button v-if="deployButton" type="primary" class="ml-2" @click="deploy">一键部署</el-button>   
 </div>   
 </template>   
 </el-page-header>   
   
 <el-row wrap>   
 <el-col v-for="(item, index) in middlewares" :key="item.id" :span="6">   
 <el-card id="card" class="card-item" style="height: 80%">   
 <div style="height: 100px; width: 100%" class="bottom">   
 <img   
 :src=item.imgUrl   
 class="image"   
 style="height: 100px; width: 40%"   
 />   
 <div style="height: 100px; width: 60%; margin-left: 5%">   
 <span>{{item.name}}   
 <el-tag   
 class="mx-1"   
 effect="light"   
 round   
 >   
 {{ item.version }}   
 </el-tag>   
 </span>   
 <div style="margin-top: 10px">   
 <span v-if="item.hasOwnProperty('userName') && item.userName !== null">   
 <label>name：<el-input   
 v-model="item.userName" style="width: 50%"   
 placeholder="Please Input"   
 >   
 </el-input></label>   
 </span>   
 <span v-if="item.hasOwnProperty('password') && item.password !== null">   
 <label>pwd：<el-input v-model="item.password" style="width: 50%"   
 placeholder="Please Input"   
 >   
 </el-input>   
 </label>   
 </span>   
 <span v-if="item.hasOwnProperty('jarPath') && item.jarPath !== null">   
 <label>jarPath：<el-input v-model="item.jarPath" style="width: 50%"   
 placeholder="Please Input"   
 >

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Title</title>

<!-- Import style -->

<link

rel="stylesheet"

href="../static/css/cdn.jsdelivr.net\_npm\_element-plus\_dist\_index.css"

/>

<!-- Import Vue 3 -->

<script src="../static/js/cdn.jsdelivr.net\_npm\_vue@3"></script>

<!-- Import component library -->

<script src="../static/js/cdn.jsdelivr.net\_npm\_element-plus"></script>

<#-- <script src="//cdn.jsdelivr.net/npm/element-plus@latest/dist/index.full.js"></script>-->

<#-- 引入 axios-->

<script src="../static/js/cdn.jsdelivr.net\_npm\_axios\_dist\_axios.min.js"></script>

<script src="../static/js/cdn.jsdelivr.net\_npm\_@element-plus\_icons-vue"></script>

<#-- <script src="https://unpkg.com/vue-typed-js@2.0.0/dist/vue-typed-js.umd.min.js"></script>-->

<script src="../static/js/cdnjs.cloudflare.com\_ajax\_libs\_three.js\_r134\_three.min.js"></script>

<script src="../static/js/cdn.jsdelivr.net\_npm\_vanta\_dist\_vanta.clouds.min.js"></script>

</head>

<body>

<div id="app2" style=" width: 100vw; height: 100vh;">

<div id="app" style="width: 100vw; height: 100vh;">

<div class="common-layout">

<el-container style="width: 100vw; height: 100vh;">

<el-header height="100px">

<span id="title">{{title}}</span>

</el-header>

<el-container style="width: 100vw; height: 100vh;">

<el-aside>

<el-timeline>

<el-timeline-item

v-for="(item, index) in timelineItems"

:key="item.id"

:timestamp="item.timestamp"

placement="top"

>

<el-card

:class="{ 'selected': selectedItem === item.id }"

@click="selectItem(item.id)"

class="custom-card"

>

<h4>{{ item.title }}</h4>

<p>{{ item.description }}</p>

</el-card>

</el-timeline-item>

</el-timeline>

</el-aside>

<el-main>

<el-page-header :icon="null">

<template #content>

<div class="flex items-center">

<el-avatar

:size="100"

class="mr-3"

src="https://cube.elemecdn.com/0/88/03b0d39583f48206768a7534e55bcpng.png"

/>

<!-- 正确使用方式 -->

</div>

</template>

<template #extra>

<div class="flex items-center">

<el-badge is-dot class="item" style="padding-right: 20px">

<el-button @click="dialogFormVisible = true" class="share-button" :icon="Share" type="primary">

日志

</el-button>

</el-badge>

<el-button>Print</el-button>

<el-button v-if="deployButton" type="primary" class="ml-2" @click="deploy">一键部署</el-button>

</div>

</template>

</el-page-header>

<el-row wrap>

<el-col v-for="(item, index) in middlewares" :key="item.id" :span="6">

<el-card id="card" class="card-item" style="height: 80%">

<div style="height: 100px; width: 100%" class="bottom">

<img

:src=item.imgUrl

class="image"

style="height: 100px; width: 40%"

/>

<div style="height: 100px; width: 60%; margin-left: 5%">

<span>{{item.name}}

<el-tag

class="mx-1"

effect="light"

round

>

{{ item.version }}

</el-tag>

</span>

<div style="margin-top: 10px">

<span v-if="item.hasOwnProperty('userName') && item.userName !== null">

<label>name：<el-input

v-model="item.userName" style="width: 50%"

placeholder="Please Input"

>

</el-input></label>

</span>

<span v-if="item.hasOwnProperty('password') && item.password !== null">

<label>pwd：<el-input v-model="item.password" style="width: 50%"

placeholder="Please Input"

>

</el-input>

</label>

</span>

<span v-if="item.hasOwnProperty('jarPath') && item.jarPath !== null">

<label>jarPath：<el-input v-model="item.jarPath" style="width: 50%"

placeholder="Please Input"

>

</el-input>

</label>

</span>

<span v-if="item.hasOwnProperty('expose') && item.expose !== null">

<label>expose：<el-input v-model="item.expose" style="width: 50%"

placeholder="Please Input"

>

</el-input>

</label>

</span>

<span v-if="item.hasOwnProperty('author') && item.author !== null">

<label>author：<el-input v-model="item.author" style="width: 50%"

placeholder="Please Input"

>

</el-input>

</label>

</span>

<span v-if="item.hasOwnProperty('mysqlServiceHost') && item.mysqlServiceHost !== null">

<label>mysqlServiceHost：<el-input v-model="item.mysqlServiceHost" style="width: 50%"

placeholder="Please Input"

>

</el-input>

</label>

</span>

<span v-if="item.hasOwnProperty('nacosDatabase') && item.nacosDatabase !== null">

<label>nacosDatabase：<el-input v-model="item.nacosDatabase" style="width: 50%"

placeholder="Please Input"

>

</el-input>

</label>

</span>

</div>

</div>

</div>

<div style="padding: 14px">

<span>{{item.description}}</span>

<div class="bottom">

<time class="time">{{ currentDate }}</time>

<el-button v-if="!deployButton" type="primary" class="ml-2" @click="deployService(item.id)">部署</el-button>

<el-switch v-if="deployButton"

v-model="item.isSelect"

class="ml-2"

inline-prompt

style="--el-switch-on-color: #13ce66; --el-switch-off-color: #ff4949"

active-text="选择"

inactive-text="取消

@change="isSelect"

/>

</div>

</div>

</el-card>

</el-col>

</el-row>

<el-dialog v-model="dialogFormVisible" title="Shipping address" style="background-color: black;">

<div ref="scrollContainer" style="height: 50vh; overflow-y: scroll;" class="scroll-container">

<p v-for="(item, index) in ListMessage" :key="item" style="color: white">{{ item }}</p