package com.yingchong.kafka;

import java.util.Date;

import java.util.Properties;

import kafka.javaapi.producer.Producer;

import kafka.producer.KeyedMessage;

import kafka.producer.ProducerConfig;

public class ProducerDemo {

public static void main(String[] args) {

int events=100;

// 设置配置属性

Properties props = new Properties();

props.put("metadata.broker.list","centos-node6:19092,centos-node7:19092,centos-node8:19092");

props.put("serializer.class", "kafka.serializer.StringEncoder");

// key.serializer.class默认为serializer.class

props.put("key.serializer.class", "kafka.serializer.StringEncoder");

// 可选配置，如果不配置，则使用默认的partitioner

props.put("partitioner.class", "com.jxl.cloub.kafka.PartitionerDemo");

// 触发acknowledgement机制，否则是fire and forget，可能会引起数据丢失

// 值为0,1,-1,可以参考

props.put("request.required.acks", "1");

ProducerConfig config = new ProducerConfig(props);

// 创建producer

Producer<String, String> producer = new Producer<String, String>(config);

// 产生并发送消息

long start=System.currentTimeMillis();

for (long i = 0; i < events; i++) {

long runtime = new Date().getTime();

String ip = "10.1.6." + i;//rnd.nextInt(255);

String msg = runtime + ",www.example.com," + ip;

//如果topic不存在，则会自动创建，默认replication-factor为1，partitions为0

KeyedMessage<String, String> data = new KeyedMessage<String, String>(

"test", ip, msg);

producer.send(data);

}

System.out.println("耗时:" + (System.currentTimeMillis() - start));

// 关闭producer

producer.close();

}

}

package com.yingchong.kafka;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

import java.util.Properties;

import java.util.concurrent.ExecutorService;

import java.util.concurrent.Executors;

import kafka.consumer.Consumer;

import kafka.consumer.ConsumerConfig;

import kafka.consumer.KafkaStream;

import kafka.javaapi.consumer.ConsumerConnector;

public class ConsumerDemo {

private final ConsumerConnector consumer;

private final String topic;

private ExecutorService executor;

public ConsumerDemo(String a\_zookeeper, String a\_groupId, String a\_topic) {

consumer = Consumer.createJavaConsumerConnector(createConsumerConfig(a\_zookeeper,a\_groupId));

this.topic = a\_topic;

}

public void shutdown() {

if (consumer != null)

consumer.shutdown();

if (executor != null)

executor.shutdown();

}

public void run(int numThreads) {

Map<String, Integer> topicCountMap = new HashMap<String, Integer>();

topicCountMap.put(topic, new Integer(numThreads));

Map<String, List<KafkaStream<byte[], byte[]>>> consumerMap = consumer

.createMessageStreams(topicCountMap);

List<KafkaStream<byte[], byte[]>> streams = consumerMap.get(topic);

// now launch all the threads

executor = Executors.newFixedThreadPool(numThreads);

// now create an object to consume the messages

//

int threadNumber = 0;

for (final KafkaStream stream : streams) {

executor.submit(new ConsumerMsgTask(stream, threadNumber));

threadNumber++;

}

}

private static ConsumerConfig createConsumerConfig(String a\_zookeeper,

String a\_groupId) {

Properties props = new Properties();

props.put("zookeeper.connect", a\_zookeeper);

props.put("group.id", a\_groupId);

props.put("zookeeper.session.timeout.ms", "400");

props.put("zookeeper.sync.time.ms", "200");

props.put("auto.commit.interval.ms", "1000");

return new ConsumerConfig(props);

}

public static void main(String[] arg) {

String[] args = { "centos-node7:2181", "0", "test", "12" };

String zooKeeper = args[0];

String groupId = args[1];

String topic = args[2];

int threads = Integer.parseInt(args[3]);

ConsumerDemo demo = new ConsumerDemo(zooKeeper, groupId, topic);

demo.run(threads);

try {

Thread.sleep(10000);

} catch (InterruptedException ie) {

}

demo.shutdown();

}

}

package com.yingchong.service.yccloud.network.service;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.Date;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

import javax.annotation.Resource;

import com.yingchong.common.yccloud.commonapi.util.CloudProperty;

import org.apache.commons.lang.StringUtils;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.amqp.AmqpException;

import org.springframework.amqp.rabbit.core.RabbitTemplate;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import com.alibaba.fastjson.JSON;

import com.alibaba.fastjson.JSONArray;

import com.alibaba.fastjson.JSONObject;

import com.github.pagehelper.PageHelper;

import com.github.pagehelper.PageInfo;

import com.yingchong.common.yccloud.commonapi.bean.NoticeBean;

import com.yingchong.common.yccloud.commonapi.bean.RabbitmqParam;

import com.yingchong.common.yccloud.commonapi.constants.Constants;

import com.yingchong.common.yccloud.commonapi.constants.exception\_code.ErrorCode;

import com.yingchong.common.yccloud.commonapi.mapper.AppGroupMapper;

import com.yingchong.common.yccloud.commonapi.mapper.CloudContainerAssetsMapper;

import com.yingchong.common.yccloud.commonapi.mapper.CloudNetworkDeviceMapper;

import com.yingchong.common.yccloud.commonapi.mapper.ClusterMapper;

import com.yingchong.common.yccloud.commonapi.mapper.InstanceMapper;

import com.yingchong.common.yccloud.commonapi.mapper.IntrIpParaMapper;

import com.yingchong.common.yccloud.commonapi.mapper.IpMapper;

import com.yingchong.common.yccloud.commonapi.mapper.NetworkContainerMacMapper;

import com.yingchong.common.yccloud.commonapi.mapper.NetworkSecurityGroupMapper;

import com.yingchong.common.yccloud.commonapi.mapper.NetworkSecurityGroupTagMapper;

import com.yingchong.common.yccloud.commonapi.mapper.NetworkSecurityTagRelationMapper;

import com.yingchong.common.yccloud.commonapi.mapper.NetworkSecuritygroupIpDetailMapper;

import com.yingchong.common.yccloud.commonapi.mapper.PublicIpMapper;

import com.yingchong.common.yccloud.commonapi.mapper.VlanMapper;

import com.yingchong.common.yccloud.commonapi.model.AppGroup;

import com.yingchong.common.yccloud.commonapi.model.AppGroupExample;

import com.yingchong.common.yccloud.commonapi.model.CloudContainerAssets;

import com.yingchong.common.yccloud.commonapi.model.CloudContainerAssetsExample;

import com.yingchong.common.yccloud.commonapi.model.CloudNetworkDevice;

import com.yingchong.common.yccloud.commonapi.model.CloudNetworkDeviceExample;

import com.yingchong.common.yccloud.commonapi.model.Cluster;

import com.yingchong.common.yccloud.commonapi.model.ClusterExample;

import com.yingchong.common.yccloud.commonapi.model.Instance;

import com.yingchong.common.yccloud.commonapi.model.InstanceExample;

import com.yingchong.common.yccloud.commonapi.model.IntrIpPara;

import com.yingchong.common.yccloud.commonapi.model.Ip;

import com.yingchong.common.yccloud.commonapi.model.NetworkContainerMac;

import com.yingchong.common.yccloud.commonapi.model.NetworkSecurityGroup;

import com.yingchong.common.yccloud.commonapi.model.NetworkSecurityGroupExample;

import com.yingchong.common.yccloud.commonapi.model.NetworkSecurityGroupTag;

import com.yingchong.common.yccloud.commonapi.model.NetworkSecurityTagRelation;

import com.yingchong.common.yccloud.commonapi.model.NetworkSecurityTagRelationExample;

import com.yingchong.common.yccloud.commonapi.model.NetworkSecuritygroupIpDetail;

import com.yingchong.common.yccloud.commonapi.model.NetworkSecuritygroupIpDetailExample;

import com.yingchong.common.yccloud.commonapi.model.PublicIp;

import com.yingchong.common.yccloud.commonapi.model.Vlan;

import com.yingchong.common.yccloud.commonapi.util.MD5;

import com.yingchong.common.yccloud.commonapi.util.RedisUtils;

import com.yingchong.common.yccloud.remote.bean.ResponseBean;

import com.yingchong.service.yccloud.network.mapper.SecurityGroupBusinessMapper;

import com.yingchong.service.yccloud.network.util.IpUtils;

@Service

public class SecurityGroupBusinessService {

private static final Logger logger = LoggerFactory.getLogger(SecurityGroupBusinessService.class);

private static final String SECURITY\_COUNT = "SECURITY\_COUNT\_";

@Autowired

private CloudProperty property;

@Autowired

private SecurityGroupBusinessMapper securityGroupBusinessMapper;

@Autowired

private NetworkSecurityGroupTagMapper networkSecurityGroupTagMapper;

@Autowired

private NetworkSecurityGroupMapper networkSecurityGroupMapper;

@Autowired

private CloudNetworkDeviceMapper cloudNetworkDeviceMapper;

@Resource(name="bridgeIn")

private String bridgeIn;

@Resource(name="bridgeOut")

private String bridgeOut;

@Autowired

private IpMapper ipMapper;

@Autowired

private PublicIpMapper publicIpMapper;

@Autowired

private NetworkContainerMacMapper networkContainerMacMapper;

@Autowired

private IntrIpParaMapper intrIpParaMapper;

@Autowired

private VlanMapper vlanMapper;

@Autowired

private RedisUtils redisUtils;

@Autowired

private NetworkSecuritygroupIpDetailMapper networkSecuritygroupIpDetailMapper;

@Autowired

private CloudContainerAssetsMapper cloudContainerAssetsMapper;

@Autowired

private RabbitTemplate rabbitTemplate;

@Autowired

private InstanceMapper instanceMapper;

@Autowired

private NetworkSecurityTagRelationMapper networkSecurityTagRelationMapper;

@Autowired

private ClusterMapper clusterMapper;

@Autowired

private AppGroupMapper appGroupMapper;

public ResponseBean<Void> addSecurityGroup(JSONObject obj, String userId){

ResponseBean<Void> result = new ResponseBean<>();

Date now = new Date();

JSONObject tagJson = obj.getJSONObject("tag");

NetworkSecurityGroupTag tag = JSON.toJavaObject(tagJson, NetworkSecurityGroupTag.class);

tag.setUserId(userId);

tag.setCreateTime(now);

this.networkSecurityGroupTagMapper.insert(tag);

JSONArray groupArray = obj.getJSONArray("group");

for (int i=0; i < groupArray.size(); i++){

JSONObject groupJson = groupArray.getJSONObject(i);

NetworkSecurityGroup group = JSON.toJavaObject(groupJson, NetworkSecurityGroup.class);

group.setGroupTagId(tag.getId());

group.setUserId(userId);

group.setCreateTime(now);

this.networkSecurityGroupMapper.insert(group);

}

result.setRetCode(ErrorCode.SUCCESS.getCode());

result.setRetMsg(ErrorCode.SUCCESS.getMsg());

return result;

}

\* }

\* @param obj

\* @return

\*/

public ResponseBean<Void> addGroupRole(JSONObject obj){

ResponseBean<Void> result = new ResponseBean<>();

String tagId = obj.getString("tagId");

JSONArray groupArray = obj.getJSONArray("group");

NetworkSecuritygroupIpDetailExample example = new NetworkSecuritygroupIpDetailExample();

example.createCriteria().andTagIdEqualTo(tagId);

List<NetworkSecuritygroupIpDetail> list = this.networkSecuritygroupIpDetailMapper.selectByExample(example);

for (int i=0; i < groupArray.size(); i++){

JSONObject groupJson = groupArray.getJSONObject(i);

NetworkSecurityGroup group = JSON.toJavaObject(groupJson, NetworkSecurityGroup.class);

group.setGroupTagId(tagId);

group.setCreateTime(new Date());

this.networkSecurityGroupMapper.insert(group);

try {

for (NetworkSecuritygroupIpDetail s : list){

CloudNetworkDeviceExample deviceExample = new CloudNetworkDeviceExample();

deviceExample.createCriteria().andStatusEqualTo(0).andTypeEqualTo(s.getType()).andIpIdEqualTo(s.getIpId());

List<CloudNetworkDevice> deviceList = this.cloudNetworkDeviceMapper.selectByExample(deviceExample);

CloudNetworkDevice device = deviceList.get(0);

NetworkContainerMac mac = this.networkContainerMacMapper.selectByPrimaryKey(device.getInsideMacId());

CloudContainerAssets container = this.cloudContainerAssetsMapper.selectByPrimaryKey(device.getCloudAssetsId());

String ip = null;

String vlan = null;

if (s.getType() == 0){

Ip nip = this.ipMapper.selectByPrimaryKey(s.getIpId());

IntrIpPara para = this.intrIpParaMapper.selectByPrimaryKey(nip.getIpParaId());

Vlan vlanBean = this.vlanMapper.selectByPrimaryKey(para.getVlanId());

ip = nip.getIp();

vlan = vlanBean.getVlanName();

}else{

PublicIp ipInfo = this.publicIpMapper.selectByPrimaryKey(s.getIpId());

ip = ipInfo.getIp();

}

Map<String, Object> paramMap = new HashMap<>();

if (s.getType() == 0){

paramMap.put("bridge\_name", property.getProperty("pub", "bridgeIn"));

}else{

paramMap.put("bridge\_name", property.getProperty("pub", "bridgeOut"));

}

paramMap.put("actions", getAcrion(group.getBehavior()));

paramMap.put("dl\_type", getDlType(group.getProtocolType()));

paramMap.put("priority", group.getPriority());

//拆分ip

List<String> ips = IpUtils.splitIp(group.getAimsIp());

paramMap.put("operator\_ips", ips);

paramMap.put("bind\_ip", ip);

paramMap.put("bind\_mac", mac.getMac());

// paramMap.put("direction", getDirection(group.getDirection()));

paramMap.put("start\_port", group.getStartPort());

paramMap.put("end\_port", group.getEndPort());

if(s.getType() == 0){

paramMap.put("vlan", vlan);

}

Map<String, Object> baseMap = new HashMap<>();

baseMap.put("serverId", container.getServerAssetsId());

RabbitmqParam rp = new RabbitmqParam();

rp.setActiveType(0);//不需回调

rp.setBaseParam(baseMap);

rp.setMethodType(Constants.ADD\_INTERFLOW\_LIMIT);

rp.setMethodParam(paramMap);

logger.info("==============");

logger.info("添加安全组参数：【{}】", rp.toJson());

logger.info("==============");

rabbitTemplate.convertAndSend(Constants.TASK\_QUEUE + container.getServerAssetsId(), rp.toJson());

}

} catch (AmqpException e) {

logger.error("关联出错", e);

}

}

result.setRetCode(ErrorCode.SUCCESS.getCode());

result.setRetMsg(ErrorCode.SUCCESS.getMsg());

return result;

}

/\*\*

\* 编辑安全组规则

\* {

\* "tagId": "tagId",

\* "group": {

\* "id": "id"

\* "securityGroupName": "规则名",

\* "protocolType": "协议类型:0 tcp/1 udp/2 icmp",

\* "priority": "优先级",

\* "behavior": "行为:0 接受/1 拒绝",

\* "startPort": "起始端口",

\* "endPort": "结束端口",

\* "aimsIp": "目标ip"

\* }

\* }

\* @param obj

\* @return

\*/

public ResponseBean<Void> editGroupRole(JSONObject obj){

String tagId = obj.getString("tagId");

JSONObject group = obj.getJSONObject("group");

//删除原来规则

JSONArray groupIds = new JSONArray();

groupIds.add(group.getString("id"));

JSONObject deleteParam = new JSONObject();

deleteParam.put("tagId", tagId);

deleteParam.put("groupIds", groupIds);

ResponseBean<Void> deleteResult = deleteGroupRole(deleteParam);

if (!ErrorCode.SUCCESS.getCode().equals(deleteResult.getRetCode())) {

return deleteResult;

}

//加新规则

JSONObject addParam = new JSONObject();

addParam.put("tagId", tagId);

JSONArray groupList = new JSONArray();

groupList.add(group);

addParam.put("group", groupList);

return addGroupRole(addParam);

}

/\*\*

\* 安全组删除规则

\* {

\* "tagId": "安全组id",

\* "groupIds": [1,2] 安全组规则id

\* }

\* @Description

\* han.zhifeng

\* @param obj

\* @return

\*/

public ResponseBean<Void> deleteGroupRole(JSONObject obj){

ResponseBean<Void> result = new ResponseBean<>();

String tagId = obj.getString("tagId");

JSONArray groupArray = obj.getJSONArray("groupIds");

List<String> ids = new ArrayList<>();

for (int i=0; i < groupArray.size(); i++){

ids.add(groupArray.getString(i));

}

NetworkSecurityGroupExample example = new NetworkSecurityGroupExample();

example.createCriteria().andGroupTagIdEqualTo(tagId).andIdIn(ids);

this.networkSecurityGroupMapper.deleteByExample(example);

NetworkSecuritygroupIpDetailExample detailExample = new NetworkSecuritygroupIpDetailExample();

detailExample.createCriteria().andSecurityGroupIdIn(ids);

List<NetworkSecuritygroupIpDetail> detailList = this.networkSecuritygroupIpDetailMapper.selectByExample(detailExample);

for (NetworkSecuritygroupIpDetail detail : detailList){

Map<String, Object> paramMap = new HashMap<>();

paramMap.put("bridge\_name", getBridgeName(detail.getType()));

paramMap.put("cookie", detail.getCookie());

Map<String, Object> baseMap = new HashMap<>();

baseMap.put("id", detail.getId());

RabbitmqParam rp = new RabbitmqParam();

rp.setActiveType(Constants.ACTIVE\_TYPE\_UNBIND\_SECURITY\_GROUP);

rp.setBaseParam(baseMap);

rp.setMethodType(Constants.DELETE\_INTERFLOW\_LIMIT);

rp.setMethodParam(paramMap);

logger.info("==============");

logger.info("删除ip绑定的安全组：【{}】", rp.toJson());

logger.info("==============");

rabbitTemplate.convertAndSend(Constants.TASK\_QUEUE + detail.getServerId(), rp.toJson());

}

result.setRetCode(ErrorCode.SUCCESS.getCode());

result.setRetMsg(ErrorCode.SUCCESS.getMsg());

return result;

}

/\*\*

\*

\* @Description

\* {

\* "groupId": "安全组的id",

\* "ipId": [{"id": "1", "type": "0内网 1外网"},{"id": "2", "type": "0内网 1外网"}]//安全组id数组

\* }

\* han.zhifeng

\* @param obj

\* @return

\*/

public ResponseBean<Void> groupBindIp(JSONObject obj){

ResponseBean<Void> result = new ResponseBean<>();

String groupTagId = obj.getString("groupId");

JSONArray ipIds = obj.getJSONArray("ipId");

// String type = obj.getString("type");//0内网 1外网

NetworkSecurityGroupExample groupExample = new NetworkSecurityGroupExample();

groupExample.createCriteria().andGroupTagIdEqualTo(groupTagId);

List<NetworkSecurityGroup> list = this.networkSecurityGroupMapper.selectByExample(groupExample);

Integer groupCount = list.size();

Integer ipCount = ipIds.size();

Integer count = groupCount \* ipCount;

String countKey = MD5.uuid();

for (NetworkSecurityGroup group : list){

for(int i = 0; i < ipIds.size(); i++){

JSONObject ipJson = ipIds.getJSONObject(i);

CloudNetworkDeviceExample deviceExample = new CloudNetworkDeviceExample();

deviceExample.createCriteria().andStatusEqualTo(0).andTypeEqualTo(ipJson.getInteger("type")).andIpIdEqualTo(ipJson.getString("id"));

List<CloudNetworkDevice> deviceList = this.cloudNetworkDeviceMapper.selectByExample(deviceExample);

CloudNetworkDevice device = deviceList.get(0);

NetworkContainerMac mac = this.networkContainerMacMapper.selectByPrimaryKey(device.getInsideMacId());

CloudContainerAssets container = this.cloudContainerAssetsMapper.selectByPrimaryKey(device.getCloudAssetsId());

String ip = null;

String vlan = null;

if ("0".equals(ipJson.getString("type"))){//内网

Ip nip = this.ipMapper.selectByPrimaryKey(ipIds.getString(i));

IntrIpPara para = this.intrIpParaMapper.selectByPrimaryKey(nip.getIpParaId());

Vlan vlanBean = this.vlanMapper.selectByPrimaryKey(para.getVlanId());

ip = nip.getIp();

vlan = vlanBean.getVlanName();

}else{//公网

PublicIp ipInfo = this.publicIpMapper.selectByPrimaryKey(ipIds.getString(i));

ip = ipInfo.getIp();

}

Map<String, Object> paramMap = new HashMap<>();

if ("0".equals(ipJson.getString("type"))){//内网

paramMap.put("bridge\_name", property.getProperty("pub", "bridgeIn"));

paramMap.put("vlan", vlan);

}else{

paramMap.put("bridge\_name", property.getProperty("pub", "bridgeOut"));

}

paramMap.put("actions", getAcrion(group.getBehavior()));

paramMap.put("dl\_type", getDlType(group.getProtocolType()));

paramMap.put("priority", group.getPriority());

//拆分ip

List<String> ips = IpUtils.splitIp(group.getAimsIp());

paramMap.put("operator\_ips", ips);

paramMap.put("bind\_ip", ip);

paramMap.put("bind\_mac", mac.getMac());

// paramMap.put("direction", getDirection(group.getDirection()));

paramMap.put("start\_port", group.getStartPort());

paramMap.put("end\_port", group.getEndPort());

Map<String, Object> baseMap = new HashMap<>();

baseMap.put("count", count);

baseMap.put("ipId", ipIds.getInteger(i));

baseMap.put("ipIds", ipIds);

baseMap.put("type", ipJson.getString("type"));

baseMap.put("groupTag", group.getGroupTagId());

baseMap.put("groupId", group.getId());

baseMap.put("countKey", countKey);

baseMap.put("userId", group.getUserId());

baseMap.put("serverId", container.getServerAssetsId());

RabbitmqParam rp = new RabbitmqParam();

rp.setActiveType(Constants.ACTIVE\_TYPE\_BIND\_SECURITY\_GROUP);

rp.setBaseParam(baseMap);

rp.setMethodType(Constants.ADD\_INTERFLOW\_LIMIT);

rp.setMethodParam(paramMap);

logger.info("==============");

logger.info("添加安全组参数：【{}】", rp.toJson());

logger.info("==============");

rabbitTemplate.convertAndSend(Constants.TASK\_QUEUE + container.getServerAssetsId(), rp.toJson());

}

}

result.setRetCode(ErrorCode.SUCCESS.getCode());

result.setRetMsg(ErrorCode.SUCCESS.getMsg());

return result;

}

/\*\*

\* ip关联安全组回调

\* @Description

\* han.zhifeng

\* @param result

\*/

public void bindIpForGroupActive(JSONObject result){

logger.info("ip关联安全组回调：【{}】", result);

JSONObject r = result.getJSONObject("result");

JSONObject baseParam = result.getJSONObject("baseParam");

Integer activeType = result.getInteger("activeType");

long c = this.redisUtils.incr(SECURITY\_COUNT + baseParam.getString("countKey"));

logger.info("次数：【{}】", c);

if (Constants.SUCCESS.equals(r.getString("status"))){

JSONObject data = r.getJSONObject("data");

NetworkSecuritygroupIpDetail detail = new NetworkSecuritygroupIpDetail();

detail.setSecurityGroupId(baseParam.getString("groupId"));

detail.setTagId(baseParam.getString("groupTag"));

detail.setIpId(baseParam.getString("ipId"));

detail.setType(baseParam.getInteger("type"));

detail.setCookie(data.getString("cookie"));

detail.setServerId(baseParam.getString("serverId"));

detail.setRelationId(baseParam.getString("relationId"));

this.networkSecuritygroupIpDetailMapper.insert(detail);

if (c == baseParam.getLongValue("count")){

logger.info("关联成功，通知前端");

NoticeBean nb = new NoticeBean();

nb.setUserId(baseParam.getString("userId"));

nb.setActiveType(activeType);

nb.setRetCode(ErrorCode.SUCCESS.getCode());

nb.setRetMessage(ErrorCode.SUCCESS.getMsg());

rabbitTemplate.convertAndSend(Constants.NOTICE\_WEB\_QUEUE,"", nb.toJson());

this.redisUtils.del(SECURITY\_COUNT + baseParam.getString("countKey"));

}

}else{

//下发出错的话cookie为空

NetworkSecuritygroupIpDetail detail = new NetworkSecuritygroupIpDetail();

detail.setSecurityGroupId(baseParam.getString("groupId"));

detail.setTagId(baseParam.getString("groupTag"));

detail.setIpId(baseParam.getString("ipId"));

detail.setType(baseParam.getInteger("type"));

// detail.setCookie(data.getString("cookie"));

detail.setServerId(baseParam.getString("serverId"));

this.networkSecuritygroupIpDetailMapper.insert(detail);

JSONObject error = r.getJSONObject("error");

logger.error("ip关联安全组出错：【{}】 参数【{}】", error, baseParam);

NoticeBean nb = new NoticeBean();

nb.setUserId(baseParam.getString("userId"));

nb.setActiveType(activeType);

nb.setRetCode(ErrorCode.CODE\_030102.getCode());

nb.setRetMessage(ErrorCode.CODE\_030102.getMsg() + "安全组：" + baseParam.getString("groupId"));

rabbitTemplate.convertAndSend(Constants.NOTICE\_WEB\_QUEUE,"", nb.toJson());

}

}

/\*\*

\* 解绑ip的安全组

\* @Description

\* {

"groupId": "安全组id",

"type": "1:分组、2:集群、3：容器",

"ids": ["1","2"]//id集合

}

\* han.zhifeng

\* @param obj

\* @return

\*/

public ResponseBean<Void> unBindIpGroup(JSONObject obj){

logger.info("解绑ip绑定的安全组：【{}】", obj);

ResponseBean<Void> result = new ResponseBean<>();

String groupId = obj.getString("groupId");

JSONArray ipId = obj.getJSONArray("ipId");

for (int i = 0; i < ipId.size(); i++){

NetworkSecuritygroupIpDetailExample example = new NetworkSecuritygroupIpDetailExample();

example.createCriteria().andTagIdEqualTo(groupId).andIpIdEqualTo(ipId.getString(i));

this.networkSecuritygroupIpDetailMapper.deleteByExample(example);

NetworkSecuritygroupIpDetailExample detailExample = new NetworkSecuritygroupIpDetailExample();

detailExample.createCriteria().andTagIdEqualTo(groupId).andIpIdEqualTo(ipId.getString(i));

List<NetworkSecuritygroupIpDetail> detailList = this.networkSecuritygroupIpDetailMapper.selectByExample(detailExample);

for (NetworkSecuritygroupIpDetail detail : detailList){

Map<String, Object> paramMap = new HashMap<>();

paramMap.put("bridge\_name", getBridgeName(detail.getType()));

paramMap.put("cookie", detail.getCookie());

Map<String, Object> baseMap = new HashMap<>();

baseMap.put("id", detail.getId());

RabbitmqParam rp = new RabbitmqParam();

rp.setActiveType(Constants.ACTIVE\_TYPE\_UNBIND\_SECURITY\_GROUP);

rp.setBaseParam(baseMap);

rp.setMethodType(Constants.DELETE\_INTERFLOW\_LIMIT);

rp.setMethodParam(paramMap);

logger.info("==============");

logger.info("删除ip绑定的安全组：【{}】", rp.toJson());

logger.info("==============");

rabbitTemplate.convertAndSend(Constants.TASK\_QUEUE + detail.getServerId(), rp.toJson());

}

}

result.setRetCode(ErrorCode.SUCCESS.getCode());

result.setRetMsg(ErrorCode.SUCCESS.getMsg());

return result;

}

/\*\*

\* 解绑ip的安全组

\* @Description

\* {

"groupId": "安全组id",

"deleteArr": [{"type": 1, "id": "2"}, {"type": 1, "id": "2"}]//"type": "1:分组、2:集群、3：容器" id:对应id

}

\* han.zhifeng

\* @param obj

\* @return

\*/

public ResponseBean<Void> unBindSecurityGroup(JSONObject obj){

logger.info("解绑安全组：【{}】", obj);

ResponseBean<Void> result = new ResponseBean<>();

String groupId = obj.getString("groupId");

JSONArray arr = obj.getJSONArray("deleteArr");

for (int i=0; i < arr.size(); i++) {

JSONObject json = arr.getJSONObject(i);

NetworkSecurityTagRelationExample example = new NetworkSecurityTagRelationExample();

example.createCriteria().andTagIdEqualTo(groupId).andTargetIdEqualTo(json.getString("id")).andTypeEqualTo(json.getInteger("type"));

List<NetworkSecurityTagRelation> list = this.networkSecurityTagRelationMapper.selectByExample(example);

for (NetworkSecurityTagRelation relation : list) {

NetworkSecuritygroupIpDetailExample detailExample = new NetworkSecuritygroupIpDetailExample();

detailExample.createCriteria().andTagIdEqualTo(groupId).andRelationIdEqualTo(relation.getId());

List<NetworkSecuritygroupIpDetail> detailList = this.networkSecuritygroupIpDetailMapper.selectByExample(detailExample);

for (NetworkSecuritygroupIpDetail detail : detailList){

Map<String, Object> paramMap = new HashMap<>();

paramMap.put("bridge\_name", getBridgeName(detail.getType()));

paramMap.put("cookie", detail.getCookie());

Map<String, Object> baseMap = new HashMap<>();

baseMap.put("id", detail.getId());

RabbitmqParam rp = new RabbitmqParam();

rp.setActiveType(Constants.ACTIVE\_TYPE\_UNBIND\_SECURITY\_GROUP);

rp.setBaseParam(baseMap);

rp.setMethodType(Constants.DELETE\_INTERFLOW\_LIMIT);

rp.setMethodParam(paramMap);

logger.info("==============");

logger.info("删除ip绑定的安全组：【{}】", rp.toJson());

logger.info("==============");

rabbitTemplate.convertAndSend(Constants.TASK\_QUEUE + detail.getServerId(), rp.toJson());

}

this.networkSecurityTagRelationMapper.deleteByPrimaryKey(relation.getId());

}

}

result.setRetCode(ErrorCode.SUCCESS.getCode());

result.setRetMsg(ErrorCode.SUCCESS.getMsg());

return result;

}

/\*\*

\* 删除完全组回调

\* @Description

\* @param result

\*/

public void unBindIpGroupActive(JSONObject result){

logger.info("ip删除关联安全组回调：【{}】", result);

JSONObject r = result.getJSONObject("result");

JSONObject baseParam = result.getJSONObject("baseParam");

String id = baseParam.getString("id");

if (Constants.SUCCESS.equals(r.getString("status"))){

logger.info("删除安全组成功 network\_securitygroup\_ip\_detail 【{}】", id);

this.networkSecuritygroupIpDetailMapper.deleteByPrimaryKey(id);

}else{

logger.error("删除安全组失败 network\_securitygroup\_ip\_detail 【{}】", id);

}

}

/\*\*

\* 获取用户可用的优先级

\* @Description

\* @param userId

\* @return

\*/

public ResponseBean<List<Integer>> getUserPriority(String userId){

ResponseBean<List<Integer>> result = new ResponseBean<>();

List<Integer> list = this.securityGroupBusinessMapper.getUserPriority(userId);

List<Integer> reList = new ArrayList<>();

for (int i=1; i<=100; i++){

if (!list.contains(i)){

reList.add(i);

}

}

result.setRetCode(ErrorCode.SUCCESS.getCode());

result.setRetMsg(ErrorCode.SUCCESS.getMsg());

result.setData(reList);

return result;

}

/\*\*

\* 查询安全组规则

\* @Description

\* @param tagId

\* @return

\*/

public ResponseBean<PageInfo<NetworkSecurityGroup>> getSecurityGroup(Integer page, Integer pageNum, String tagId){

ResponseBean<PageInfo<NetworkSecurityGroup>> result = new ResponseBean<>();

NetworkSecurityGroupExample example = new NetworkSecurityGroupExample();

example.createCriteria().andGroupTagIdEqualTo(tagId);

PageHelper.startPage(page, pageNum);

List<NetworkSecurityGroup> list = this.networkSecurityGroupMapper.selectByExample(example);

PageInfo<NetworkSecurityGroup> info = new PageInfo<>(list);

result.setRetCode(ErrorCode.SUCCESS.getCode());

result.setRetMsg(ErrorCode.SUCCESS.getMsg());

result.setData(info);

return result;

}

/\*\*

\* 查询内网安全组

\* @Description

\* @param page

\* @param pageNum

\* @param startTime

\* @param endTime

\* @param keyword

\* @param userId

\* @return

\*/

public ResponseBean<PageInfo<Map<String, Object>>> getInGroup(

Integer page, Integer pageNum, String startTime, String endTime, String keyword, String userId, String mobile

){

ResponseBean<PageInfo<Map<String, Object>>> result = new ResponseBean<>();

Map<String, Object> param = new HashMap<>();

if (startTime != null){

startTime = startTime + " 00:00:00";

param.put("startTime", startTime);

}

if(endTime != null){

endTime = endTime + " 23:59:59";

param.put("endTime", endTime);

}

param.put("keyword", keyword);

param.put("userId", userId);

param.put("mobile", mobile);

PageHelper.startPage(page, pageNum);

List<Map<String, Object>> list = this.securityGroupBusinessMapper.getInGroup(param);

PageInfo<Map<String, Object>> info = new PageInfo<>(list);

result.setRetCode(ErrorCode.SUCCESS.getCode());

result.setRetMsg(ErrorCode.SUCCESS.getMsg());

result.setData(info);

return result;

}

/\*\*

\* 查询安全组详情

\* @Description

\* @param id

\* @return

\*/

public ResponseBean<NetworkSecurityGroupTag> getSecurityGroup(String id){

ResponseBean<NetworkSecurityGroupTag> result = new ResponseBean<>();

NetworkSecurityGroupTag group = this.networkSecurityGroupTagMapper.selectByPrimaryKey(id);

result.setRetCode(ErrorCode.SUCCESS.getCode());

result.setRetMsg(ErrorCode.SUCCESS.getMsg());

result.setData(group);

return result;

}

/\*\*

\* 编辑安全组

\* @Description

\* @param param

\* @return

\*/

public ResponseBean<Void> updateSecurityGroup(NetworkSecurityGroupTag param){

ResponseBean<Void> result = new ResponseBean<>();

this.networkSecurityGroupTagMapper.updateByPrimaryKeySelective(param);

result.setRetCode(ErrorCode.SUCCESS.getCode());

result.setRetMsg(ErrorCode.SUCCESS.getMsg());

return result;

}

public ResponseBean<Void> deleteSecurityGroup(String id){

ResponseBean<Void> result = new ResponseBean<>();

// NetworkSecuritygroupIpDetailExample example = new NetworkSecuritygroupIpDetailExample();

// example.createCriteria().andTagIdEqualTo(id);

NetworkSecurityTagRelationExample example = new NetworkSecurityTagRelationExample();

example.createCriteria().andTagIdEqualTo(id);

long count = this.networkSecurityTagRelationMapper.countByExample(example);

if (count > 0){

result.setRetCode(ErrorCode.CODE\_030103.getCode());

result.setRetMsg(ErrorCode.CODE\_030103.getMsg());

return result;

}

this.networkSecurityGroupTagMapper.deleteByPrimaryKey(id);

NetworkSecurityGroupExample deleteExample = new NetworkSecurityGroupExample();

deleteExample.createCriteria().andGroupTagIdEqualTo(id);

this.networkSecurityGroupMapper.deleteByExample(deleteExample);

result.setRetCode(ErrorCode.SUCCESS.getCode());

result.setRetMsg(ErrorCode.SUCCESS.getMsg());

return result;

}

/\*\*

\* 批量删除安全组

\* @Description

\* @param obj

\* @return

\*/

public ResponseBean<Void> deleteSecurityGroup(JSONObject obj){

ResponseBean<Void> result = new ResponseBean<>();

JSONArray array = obj.getJSONArray("id");

List<String> ids = new ArrayList<>();

for(int i=0; i < array.size(); i++){

ids.add(array.getString(i));

}

NetworkSecuritygroupIpDetailExample example = new NetworkSecuritygroupIpDetailExample();

example.createCriteria().andTagIdIn(ids);

long count = this.networkSecuritygroupIpDetailMapper.countByExample(example);

if (count > 0){

result.setRetCode(ErrorCode.CODE\_030103.getCode());

result.setRetMsg(ErrorCode.CODE\_030103.getMsg());

return result;

}

for(String id : ids){

this.networkSecurityGroupTagMapper.deleteByPrimaryKey(id);

NetworkSecurityGroupExample deleteExample = new NetworkSecurityGroupExample();

deleteExample.createCriteria().andGroupTagIdEqualTo(id);

this.networkSecurityGroupMapper.deleteByExample(deleteExample);

}

result.setRetCode(ErrorCode.SUCCESS.getCode());

result.setRetMsg(ErrorCode.SUCCESS.getMsg());

return result;

}

private String getBridgeName(int type){

switch (type) {

case 0:

return property.getProperty("pub", "bridgeIn");

case 1:

return property.getProperty("pub", "bridgeOut");

default:

return null;

}

}

private String getAcrion(int action){

switch (action) {

case 0:

return "normal";

case 1:

return "drop";

default:

return null;

}

}

private String getDlType(int action){

switch (action) {

case 0:

return "tcp";

case 1:

return "udp";

case 2:

return "icmp";

default:

return null;

}

}

/\*\*

\* 深度递归查询应用分组下的容器

\* @param appId

\* @param idList

\*/

private void getContainerIdByAppId(String appId, List<String> idList) {

List<Map<String, Object>> slotList = this.securityGroupBusinessMapper.getAppSlot(appId);

for (Map<String, Object> slot : slotList) {

if (slot.get("id\_type") == null || slot.get("p\_id") == null) {

logger.error("有异常slot：【{}】", JSON.toJSONString(slot));

continue;

}

String type = slot.get("id\_type").toString();

String id = slot.get("p\_id").toString();

if ("3".equals(type)) {

Instance instance = this.instanceMapper.selectByPrimaryKey(id);

idList.add(instance.getContainerId());

}else if ("2".equals(type)) {

InstanceExample example = new InstanceExample();

example.createCriteria().andClusterIdEqualTo(id);

List<Instance> cluster = this.instanceMapper.selectByExample(example);

for (Instance instance : cluster) {

idList.add(instance.getContainerId());

}

}else if ("1".equals(type)){

getContainerIdByAppId(id, idList);

}

}

}

/\*\*

\* 查询集群下的容器

\* @param clusterId

\* @param idList

\*/

private void getContainerIdByCluster(String clusterId, List<String> idList) {

InstanceExample example = new InstanceExample();

example.createCriteria().andClusterIdEqualTo(clusterId);

List<Instance> cluster = this.instanceMapper.selectByExample(example);

for (Instance instance : cluster) {

idList.add(instance.getContainerId());

}

}

private void containerBindSecurityGroup(String tagId, List<String> containerIds, String relationId) {

CloudNetworkDeviceExample example = new CloudNetworkDeviceExample();

example.createCriteria().andStatusEqualTo(0).andTypeIn(Arrays.asList(0,1)).andCloudAssetsIdIn(containerIds);

List<CloudNetworkDevice> networkList = this.cloudNetworkDeviceMapper.selectByExample(example);

NetworkSecurityGroupExample groupExample = new NetworkSecurityGroupExample();

groupExample.createCriteria().andGroupTagIdEqualTo(tagId);

List<NetworkSecurityGroup> list = this.networkSecurityGroupMapper.selectByExample(groupExample);

Integer groupCount = list.size();

Integer ipCount = networkList.size();

Integer count = groupCount \* ipCount;

String countKey = MD5.uuid();

for (NetworkSecurityGroup group : list){

for(CloudNetworkDevice device : networkList){

NetworkContainerMac mac = this.networkContainerMacMapper.selectByPrimaryKey(device.getInsideMacId());

CloudContainerAssets container = this.cloudContainerAssetsMapper.selectByPrimaryKey(device.getCloudAssetsId());

String ip = null;

String vlan = null;

if (device.getType() == 0){//内网

Ip nip = this.ipMapper.selectByPrimaryKey(device.getIpId());

IntrIpPara para = this.intrIpParaMapper.selectByPrimaryKey(nip.getIpParaId());

Vlan vlanBean = this.vlanMapper.selectByPrimaryKey(para.getVlanId());

ip = nip.getIp();

vlan = vlanBean.getVlanName();

}else{//公网

PublicIp ipInfo = this.publicIpMapper.selectByPrimaryKey(device.getIpId());

ip = ipInfo.getIp();

}

Map<String, Object> paramMap = new HashMap<>();

if (device.getType() == 0){//内网

paramMap.put("bridge\_name", property.getProperty("pub", "bridgeIn"));

paramMap.put("vlan", vlan);

}else{

paramMap.put("bridge\_name", property.getProperty("pub", "bridgeOut"));

}

paramMap.put("actions", getAcrion(group.getBehavior()));

paramMap.put("dl\_type", getDlType(group.getProtocolType()));

paramMap.put("priority", group.getPriority());

//拆分ip

List<String> ips = IpUtils.splitIp(group.getAimsIp());

paramMap.put("operator\_ips", ips);

paramMap.put("bind\_ip", ip);

paramMap.put("bind\_mac", mac.getMac());

paramMap.put("start\_port", group.getStartPort());

paramMap.put("end\_port", group.getEndPort());

Map<String, Object> baseMap = new HashMap<>();

baseMap.put("count", count);

baseMap.put("ipId", device.getIpId());

baseMap.put("type", device.getType());

baseMap.put("groupTag", group.getGroupTagId());

baseMap.put("groupId", group.getId());

baseMap.put("countKey", countKey);

baseMap.put("userId", group.getUserId());

baseMap.put("serverId", container.getServerAssetsId());

baseMap.put("relationId", relationId);

RabbitmqParam rp = new RabbitmqParam();

rp.setActiveType(Constants.ACTIVE\_TYPE\_BIND\_SECURITY\_GROUP);

rp.setBaseParam(baseMap);

rp.setMethodType(Constants.ADD\_INTERFLOW\_LIMIT\_BATCH);

rp.setMethodParam(paramMap);

logger.info("==============");

logger.info("添加安全组参数：【{}】", rp.toJson());

logger.info("==============");

rabbitTemplate.convertAndSend(Constants.TASK\_QUEUE + container.getServerAssetsId(), rp.toJson());

}

}

}

/\*\*

\* 绑定安全组

\* {

"groupId": "安全组id",

"type": "1:分组、2:集群、3：容器",

"ids": ["1","2"]//id集合

}

\* @param obj

\* @return

\*/

public ResponseBean<Void> bindSecurityGroup(JSONObject obj){

ResponseBean<Void> result = new ResponseBean<>();

JSONArray ids = obj.getJSONArray("ids");

String groupId = obj.getString("groupId");

Integer type = obj.getInteger("type");

NetworkSecurityTagRelation relation = new NetworkSecurityTagRelation();

relation.setTagId(groupId);

relation.setType(type);

for (int i = 0; i < ids.size(); i++) {

List<String> containerIds = new ArrayList<>();

if (type == 3) {

containerIds.add(ids.getString(i));

}else {

List<String> idList = new ArrayList<>();

if (type == 1) {

getContainerIdByAppId(ids.getString(i), idList);

}else if (type == 2) {

getContainerIdByCluster(ids.getString(i), idList);

}

containerIds.addAll(idList);

}

relation.setTargetId(ids.getString(i));

this.networkSecurityTagRelationMapper.insert(relation);

if (containerIds != null && !containerIds.isEmpty()) {

containerBindSecurityGroup(groupId, containerIds, relation.getId());

}

}

result.setRetCode(ErrorCode.SUCCESS.getCode());

result.setRetMsg(ErrorCode.SUCCESS.getMsg());

return result;

}

/\*\*

\* 查询用户所有主机选择ip

\* @param serverCenterId

\* @return

\*/

public ResponseBean<List<CloudContainerAssets>> selectContainerIp(String serverCenterId, String userId, String name){

ResponseBean<List<CloudContainerAssets>> result = new ResponseBean<>();

CloudContainerAssetsExample example = new CloudContainerAssetsExample();

CloudContainerAssetsExample.Criteria c = example.createCriteria().andIsGatewayEqualTo(0).andStatusIn(Arrays.asList(1,2)).andServerCenterIdEqualTo(serverCenterId).andUserIdEqualTo(userId).andIsDeleteEqualTo(0).andIsDisplayEqualTo(1);

if (name != null) {

c.andWebnameLike("%" + name + "%");

}

List<CloudContainerAssets> list = this.cloudContainerAssetsMapper.selectByExample(example);

result.setRetCode(ErrorCode.SUCCESS.getCode());

result.setRetMsg(ErrorCode.SUCCESS.getMsg());

result.setData(list);

return result;

}

/\*\*

\* 查询用户所有集群选择ip

\* @param serverCenterId

\* @return

\*/

public ResponseBean<List<Cluster>> selectClusterIp(String serverCenterId, String userId, String name){

ResponseBean<List<Cluster>> result = new ResponseBean<>();

ClusterExample example = new ClusterExample();

ClusterExample.Criteria c = example.createCriteria().andServerCenterIdEqualTo(serverCenterId).andUserIdEqualTo(userId).andStatusIn(Arrays.asList(1,2)).andIsDeleteEqualTo(0);

if (name != null) {

c.andClusterNameLike("%" + name + "%");

}

List<Cluster> list = this.clusterMapper.selectByExample(example);

result.setRetCode(ErrorCode.SUCCESS.getCode());

result.setRetMsg(ErrorCode.SUCCESS.getMsg());

result.setData(list);

return result;

}

/\*\*

\* 查询用户的应用分组选择ip

\* @param serverCenterId

\* @return

\*/

public ResponseBean<List<AppGroup>> selectAppGroupIp(String serverCenterId, String userId, String name){

ResponseBean<List<AppGroup>> result = new ResponseBean<>();

AppGroupExample example = new AppGroupExample();

AppGroupExample.Criteria c = example.createCriteria().andServerCenterIdEqualTo(serverCenterId).andUserIdEqualTo(userId).andIsDeleteEqualTo(0);

if (name != null) {

c.andAppGroupNameLike("%" + name + "%");

}

List<AppGroup> list = this.appGroupMapper.selectByExample(example);

result.setRetCode(ErrorCode.SUCCESS.getCode());

result.setRetMsg(ErrorCode.SUCCESS.getMsg());

result.setData(list);

return result;

}

/\*\*

\* 查询ip选取的容器、集群、应用分组的ip

\* {

"type": "1:分组、2:集群、3：容器",

"ids": ["1","2"]//id集合

}

\* @param obj

\* @return

\*/

public ResponseBean<List<String>> getIp(JSONObject obj){

ResponseBean<List<String>> result = new ResponseBean<>();

List<String> containerIds = new ArrayList<>();

JSONArray ids = obj.getJSONArray("ids");

Integer type = obj.getInteger("type");

for (int i = 0; i < ids.size(); i++) {

if (type == 3) {

containerIds.add(ids.getString(i));

}else {

List<String> idList = new ArrayList<>();

if (type == 1) {

getContainerIdByAppId(ids.getString(i), idList);

}else if (type == 2) {

getContainerIdByCluster(ids.getString(i), idList);

}

containerIds.addAll(idList);

}

}

if (!containerIds.isEmpty()) {

List<String> list = getContaienrIp(containerIds);

result.setData(list);

}

result.setRetCode(ErrorCode.SUCCESS.getCode());

result.setRetMsg(ErrorCode.SUCCESS.getMsg());

return result;

}

public ResponseBean<List<String>> getIpType(JSONObject obj){

ResponseBean<List<String>> result = new ResponseBean<>();

List<String> containerIds = new ArrayList<>();

JSONArray ids = obj.getJSONArray("ids");

Integer type = obj.getInteger("type");

for (int i = 0; i < ids.size(); i++) {

if (type == 3) {

containerIds.add(ids.getString(i));

}else {

List<String> idList = new ArrayList<>();

if (type == 1) {

getContainerIdByAppId(ids.getString(i), idList);

}else if (type == 2) {

getContainerIdByCluster(ids.getString(i), idList);

}

containerIds.addAll(idList);

}

}

List<String> list = getContaienrIpType(containerIds);

result.setRetCode(ErrorCode.SUCCESS.getCode());

result.setRetMsg(ErrorCode.SUCCESS.getMsg());

result.setData(list);

return result;

}

private List<String> getContaienrIp(List<String> containerIds){

List<String> ips = new ArrayList<>();

CloudNetworkDeviceExample example = new CloudNetworkDeviceExample();

example.createCriteria().andStatusEqualTo(0).andTypeIn(Arrays.asList(0,1)).andCloudAssetsIdIn(containerIds);

List<CloudNetworkDevice> networkList = this.cloudNetworkDeviceMapper.selectByExample(example);

for(CloudNetworkDevice device : networkList){

if (device.getType() == 0){//内网

Ip nip = this.ipMapper.selectByPrimaryKey(device.getIpId());

ips.add(nip.getIp());

}else{//公网

PublicIp ipInfo = this.publicIpMapper.selectByPrimaryKey(device.getIpId());

ips.add(ipInfo.getIp());

}

}

return ips;

}

private List<String> getContaienrIpType(List<String> containerIds){

if (containerIds.isEmpty()) {

return null;

}

List<String> ips = new ArrayList<>();

CloudNetworkDeviceExample example = new CloudNetworkDeviceExample();

example.createCriteria().andStatusEqualTo(0).andTypeIn(Arrays.asList(0,1)).andCloudAssetsIdIn(containerIds);

List<CloudNetworkDevice> networkList = this.cloudNetworkDeviceMapper.selectByExample(example);

for(CloudNetworkDevice device : networkList){

if (device.getType() == 0){//内网

Ip nip = this.ipMapper.selectByPrimaryKey(device.getIpId());

ips.add(nip.getIp() + "(内)");

}else{//公网

PublicIp ipInfo = this.publicIpMapper.selectByPrimaryKey(device.getIpId());

ips.add(ipInfo.getIp() + "(公)");

}

}

return ips;

}

/\*\*

\* 绑定安全组选择容器

\* @param serverCenterId

\* @param userId

\* @return

\*/

public ResponseBean<List<Map<String, Object>>> getUserContainerBindSecurity(String serverCenterId, String userId, String tagId, String name){

ResponseBean<List<Map<String, Object>>> result = new ResponseBean<>();

Map<String, Object> param = new HashMap<>();

param.put("serverCenterId", serverCenterId);

param.put("userId", userId);

param.put("tagId", tagId);

param.put("name", name);

List<Map<String, Object>> list = this.securityGroupBusinessMapper.getUserContainerBindSecurity(param);

result.setRetCode(ErrorCode.SUCCESS.getCode());

result.setRetMsg(ErrorCode.SUCCESS.getMsg());

result.setData(list);

return result;

}

/\*\*

\* 绑定安全组选择集群

\* @param serverCenterId

\* @param userId

\* @return

\*/

public ResponseBean<List<Map<String, Object>>> getUserClusterBindSecurity(String serverCenterId, String userId, String tagId, String name){

ResponseBean<List<Map<String, Object>>> result = new ResponseBean<>();

Map<String, Object> param = new HashMap<>();

param.put("serverCenterId", serverCenterId);

param.put("userId", userId);

param.put("tagId", tagId);

param.put("name", name);

List<Map<String, Object>> list = this.securityGroupBusinessMapper.getUserClusterBindSecurity(param);

result.setRetCode(ErrorCode.SUCCESS.getCode());

result.setRetMsg(ErrorCode.SUCCESS.getMsg());

result.setData(list);

return result;

}

/\*\*

\* 绑定安全组选择应用分组

\* @param serverCenterId

\* @param userId

\* @return

\*/

public ResponseBean<List<Map<String, Object>>> getUserAppGroupBindSecurity(String serverCenterId, String userId, String tagId, String name){

ResponseBean<List<Map<String, Object>>> result = new ResponseBean<>();

Map<String, Object> param = new HashMap<>();

param.put("serverCenterId", serverCenterId);

param.put("userId", userId);

param.put("tagId", tagId);

param.put("name", name);

List<Map<String, Object>> list = this.securityGroupBusinessMapper.getUserAppGroupBindSecurity(param);

result.setRetCode(ErrorCode.SUCCESS.getCode());

result.setRetMsg(ErrorCode.SUCCESS.getMsg());

result.setData(list);

return result;

}

/\*\*

\* 查询安全组绑定的成员带分页

\* @param page

\* @param pageNum

\* @param tagId

\* @return

\*/

public ResponseBean<PageInfo<Map<String, Object>>> securityGroupBindList(Integer page, Integer pageNum, String tagId){

ResponseBean<PageInfo<Map<String, Object>>> result = new ResponseBean<>();

PageHelper.startPage(page, pageNum);

List<Map<String, Object>> list = this.securityGroupBusinessMapper.securityGroupBindList(tagId);

for (Map<String, Object> map : list) {

JSONObject obj = new JSONObject();

obj.put("type", map.get("type"));

List<Object> ids = new ArrayList<>();

ids.add(map.get("target\_id"));

obj.put("ids", ids);

List<String> ipList = getIpType(obj).getData();

if (ipList != null) {

String ip = StringUtils.join(ipList.toArray(), ",");

map.put("ip", ip);

}

}

PageInfo<Map<String, Object>> info = new PageInfo<>(list);

result.setRetCode(ErrorCode.SUCCESS.getCode());

result.setRetMsg(ErrorCode.SUCCESS.getMsg());

result.setData(info);

return result;

}

}

package com.yingchong.service.yccloud.network.service;

import com.alibaba.fastjson.JSON;

import com.alibaba.fastjson.JSONObject;

import com.yingchong.common.yccloud.commonapi.constants.Constants;

import com.yingchong.common.yccloud.commonapi.mapper.\*;

import com.yingchong.common.yccloud.commonapi.model.\*;

import com.yingchong.common.yccloud.commonapi.util.CloudProperty;

import com.yingchong.common.yccloud.commonapi.util.HttpUtils;

import com.yingchong.common.yccloud.remote.bean.ResponseBean;

import com.yingchong.common.yccloud.remote.bean.network.BatchNetworkParams;

import com.yingchong.common.yccloud.commonapi.constants.exception\_code.ErrorCode;

import com.yingchong.service.yccloud.network.mapper.IpBusinessMapper;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import javax.annotation.Resource;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

@Service

public class NetworkBusinessService {

private static final Logger logger = LoggerFactory.getLogger(NetworkBusinessService.class);

@Autowired

private CloudProperty property;

//@Value("${create.network.bridge.in}")

// @Resource(name = "bridgeIn")

// private String bridgeIn;

//@Value("${create.network.bridge.out}")

// @Resource(name = "bridgeOut")

// private String bridgeOut;

// @Resource(name="userLocation")

// private String userLocation;

// @Resource(name = "webConsoleSshdaddr")

// private String webConsoleSshdaddr;

// @Resource(name = "webconsoleMaxNum")

// private String webconsoleMaxNum;

@Autowired

private CloudNetworkDeviceMapper cloudNetworkDeviceMapper;

public static final String NETWORK\_TYPE\_IN = "0";

public static final String NETWORK\_TYPE\_OUT = "1";

public static final String GATEWAY\_TYPE = "2";

@Autowired

private IpMapper ipMapper;

@Autowired

private PublicIpMapper publicIpMapper;

@Autowired

private NetworkContainerMacMapper networkContainerMacMapper;

@Autowired

private IpBusinessMapper ipBusinessMapper;

@Autowired

private UserInfoMapper userInfoMapper;

@Autowired

private CloudContainerAssetsMapper cloudContainerAssetsMapper;

public ResponseBean<String> getShellBoxUrl(String containerId) {

ResponseBean<String> responseBean = new ResponseBean<>();

CloudNetworkDeviceExample example = new CloudNetworkDeviceExample();

example.createCriteria().andCloudAssetsIdEqualTo(containerId).andStatusNotEqualTo(Constants.DELETED);

List<CloudNetworkDevice> cloudNetworkDevices = cloudNetworkDeviceMapper.selectByExample(example);

for (CloudNetworkDevice cloudNetworkDevice : cloudNetworkDevices) {

if (cloudNetworkDevice.getType() == 1) {//外网

PublicIp publicIp = publicIpMapper.selectByPrimaryKey(cloudNetworkDevice.getIpId());

responseBean.setData(publicIp.getIp()+":4200");

responseBean.setCodeAndMsg(ErrorCode.SUCCESS\_code.getCode(),ErrorCode.SUCCESS\_code.getMsg());

return responseBean;

} else if (cloudNetworkDevice.getType() == 0) {

//查内网ip,获取内网网段

Ip ip = ipMapper.selectByPrimaryKey(cloudNetworkDevice.getIpId());

CloudNetworkDeviceExample example1 = new CloudNetworkDeviceExample();

example1.createCriteria().andIpIdEqualTo(ip.getIpParaId()).andStatusNotEqualTo(Constants.DELETED);

//用内网网段查出网关

List<CloudNetworkDevice> cloudNetworkDevices1 = cloudNetworkDeviceMapper.selectByExample(example1);

CloudNetworkDevice cloudNetworkDevice1 = cloudNetworkDevices1.get(0);

CloudNetworkDeviceExample example2 = new CloudNetworkDeviceExample();

example2.createCriteria().andCloudAssetsIdEqualTo(cloudNetworkDevice1.getCloudAssetsId())

.andTypeEqualTo(1);

//查出网关外网

List<CloudNetworkDevice> cloudNetworkDevices2 = cloudNetworkDeviceMapper.selectByExample(example2);

//查询外网信息

PublicIp publicIp = publicIpMapper.selectByPrimaryKey(cloudNetworkDevices2.get(0).getIpId());

responseBean.setData(publicIp.getIp()+"/"+remDot(ip.getIp()));

responseBean.setCodeAndMsg(ErrorCode.SUCCESS\_code.getCode(),ErrorCode.SUCCESS\_code.getMsg());

return responseBean;

}

}

responseBean.setData("null");

responseBean.setCodeAndMsg(ErrorCode.code\_030106.getCode(),ErrorCode.code\_030106.getMsg());

return responseBean;

}

public ResponseBean<String> getWebconsuleUrl(String containerId) {

ResponseBean<String> responseBean = new ResponseBean<>();

CloudNetworkDeviceExample example = new CloudNetworkDeviceExample();

example.createCriteria().andCloudAssetsIdEqualTo(containerId).andStatusNotEqualTo(Constants.DELETED);

List<CloudNetworkDevice> cloudNetworkDevices = cloudNetworkDeviceMapper.selectByExample(example);

for (CloudNetworkDevice cloudNetworkDevice : cloudNetworkDevices) {

if (cloudNetworkDevice.getType().equals("1")) {//外网

PublicIp publicIp = publicIpMapper.selectByPrimaryKey(cloudNetworkDevice.getIpId());

responseBean.setData(publicIp.getIp()+":22");

responseBean.setCodeAndMsg(ErrorCode.SUCCESS\_code.getCode(),ErrorCode.SUCCESS\_code.getMsg());

return responseBean;

} else if (cloudNetworkDevice.getType().equals("0")) {

Ip ip = ipMapper.selectByPrimaryKey(cloudNetworkDevice.getIpId());

responseBean.setData(ip.getIp()+":22");

responseBean.setCodeAndMsg(ErrorCode.SUCCESS\_code.getCode(),ErrorCode.SUCCESS\_code.getMsg());

return responseBean;

}

}

responseBean.setData("null");

responseBean.setCodeAndMsg(ErrorCode.code\_030106.getCode(),ErrorCode.code\_030106.getMsg());

return responseBean;

}

private String remDot(String ip) {

return ip.replaceAll("\\.","");

}

/\*\*\*

\* 组装参数

\* @param containerId

\* @return

\*/

public List<BatchNetworkParams> getPatchNetworkParams(String containerId) {

List<BatchNetworkParams> paramsList = new ArrayList<>();

//CloudContainerAssets cloudContainerAssets = cloudContainerAssetsMapper.selectByPrimaryKey(containerId);

CloudNetworkDeviceExample example = new CloudNetworkDeviceExample();

example.createCriteria().

andCloudAssetsIdEqualTo(containerId).

andStatusNotEqualTo(Constants.DELETED);

List<CloudNetworkDevice> cloudNetworkDevices = cloudNetworkDeviceMapper.selectByExample(example);

for (CloudNetworkDevice cloudNetworkDevice : cloudNetworkDevices) {

NetworkContainerMac networkContainerMac = networkContainerMacMapper.selectByPrimaryKey(cloudNetworkDevice.getInsideMacId());

if (networkContainerMac == null) {

return null;

}

BatchNetworkParams params = new BatchNetworkParams();

int type = cloudNetworkDevice.getType();

String deviceId = cloudNetworkDevice.getId();

params.setDeviceId(deviceId);

if (type == 1) {//baseNetwork

//PublicIp publicIp = publicIpMapper.selectByPrimaryKey(cloudNetworkDevice.getIpId());

List<Map<String, Object>> pubIpInfoNotEndTime = ipBusinessMapper.findPubIpInfoNotEndTime(cloudNetworkDevice.getIpId());

if (pubIpInfoNotEndTime != null && pubIpInfoNotEndTime.size() > 0) {

Map<String, Object> map = pubIpInfoNotEndTime.get(0);

params.setBridge(property.getProperty("pub", "bridgeOut"));

params.setGateway(map.get("gateway").toString());

params.setIp(map.get("ip").toString());

params.setIp\_public("1");

params.setMac(networkContainerMac.getMac());

params.setVlan("");//baseNetwork vlan is null

params.setMask("24");

String[] route = new String[1];

route[0] = map.get("gateway").toString()+"/24";

params.setRoute(route);

paramsList.add(params);

}

}else {

Map<String, Object> ipParam = new HashMap<>();

ipParam.put("ipId",cloudNetworkDevice.getIpId());

List<Map<String, Object>> ipInfo = ipBusinessMapper.findIpInfo(ipParam);

if (ipInfo == null || ipInfo.isEmpty()) {

logger.error("未找到ipId为{}的内网ip,请查询数据",cloudNetworkDevice.getIpId());

throw new RuntimeException("未找到ipId为"+cloudNetworkDevice.getIpId()+"的内网ip,请查询数据");

}

Map<String, Object> ip = ipInfo.get(0);

params.setBridge(property.getProperty("pub", "bridgeIn"));

params.setGateway(String.valueOf(ip.get("gateway")));

params.setIp(String.valueOf(ip.get("ip")));

params.setIp\_public("0");

params.setMac(networkContainerMac.getMac());

params.setVlan(String.valueOf(ip.get("vlan\_name")));

params.setMask("24");

String[] route = new String[1];

route[0] = String.valueOf(ip.get("gateway"))+"/24";

params.setRoute(route);

paramsList.add(params);

}

}

return paramsList;

}

public ResponseBean<String> getEnAddr(String vmAddr,String userId) {

UserInfoExample example = new UserInfoExample();

example.createCriteria().andUserIdEqualTo(userId);

List<UserInfo> userInfos = userInfoMapper.selectByExample(example);

UserInfo userInfo = userInfos.get(0);

//String id = userInfo.getId();

Integer hashNum = 0;//todo 以后修改此数值

int index = hashNum/Integer.parseInt(property.getProperty("network", "webconsoleMaxNum"))+1; //发送到webconsole上的序号

//http://10.10.25.39:8080/console/chksshdaddr?vm\_addr=10.10.25.45:22

//{"ok":true,"msg":"","data":{"en\_addr":"t-npNFuEf5boMzO8lyA=","sshd\_addr":"10.10.25.82:22"}

String url = property.getProperty("network", "webConsoleSshdaddr") + "ws"+index + "/console/chksshdaddr";

String result = HttpUtils.sendGet(url, "vm\_addr="+vmAddr);

logger.info("result = {},url={},vm\_addr={}",result,url,vmAddr);

JSONObject jsonObject = JSON.parseObject(result);

String en\_addr = jsonObject.getJSONObject("data").getString("en\_addr");

//http://10.10.25.39/ws01/console/login/t-npNFuEf5boMzO8lyA=

String gotoUrl = property.getProperty("network", "webConsoleSshdaddr") + "ws"+index +"/console/login/"+en\_addr;

ResponseBean<String> responseBean = new ResponseBean<>();

responseBean.setData(gotoUrl);

return responseBean;

}

/\*\*

\* 查询网关

\* @param userId

\* @param serverCenterId

\* @return

\*/

public ResponseBean<List<CloudContainerAssets>> getUserGateway(String userId, String serverCenterId){

ResponseBean<List<CloudContainerAssets>> result = new ResponseBean<>();

CloudContainerAssetsExample example = new CloudContainerAssetsExample();

example.createCriteria().andUserIdEqualTo(userId).andServerCenterIdEqualTo(serverCenterId).andIsGatewayEqualTo(1).andStatusIn(Arrays.asList(1, 2));

List<CloudContainerAssets> list = this.cloudContainerAssetsMapper.selectByExample(example);

result.setData(list);

result.setCodeAndMsg(ErrorCode.SUCCESS.getCode(), ErrorCode.SUCCESS.getMsg());

return result;

}

}

package com.yingchong.service.yccloud.network.service;

import com.alibaba.fastjson.JSON;

import com.alibaba.fastjson.JSONObject;

import com.yingchong.common.yccloud.commonapi.constants.Constants;

import com.yingchong.common.yccloud.commonapi.mapper.\*;

import com.yingchong.common.yccloud.commonapi.model.\*;

import com.yingchong.common.yccloud.commonapi.util.CloudProperty;

import com.yingchong.common.yccloud.commonapi.util.HttpUtils;

import com.yingchong.common.yccloud.remote.bean.ResponseBean;

import com.yingchong.common.yccloud.remote.bean.network.BatchNetworkParams;

import com.yingchong.common.yccloud.commonapi.constants.exception\_code.ErrorCode;

import com.yingchong.service.yccloud.network.mapper.IpBusinessMapper;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import javax.annotation.Resource;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

@Service

public class NetworkBusinessService {

private static final Logger logger = LoggerFactory.getLogger(NetworkBusinessService.class);

@Autowired

private CloudProperty property;

//@Value("${create.network.bridge.in}")

// @Resource(name = "bridgeIn")

// private String bridgeIn;

//@Value("${create.network.bridge.out}")

// @Resource(name = "bridgeOut")

// private String bridgeOut;

// @Resource(name="userLocation")

// private String userLocation;

// @Resource(name = "webConsoleSshdaddr")

// private String webConsoleSshdaddr;

// @Resource(name = "webconsoleMaxNum")

// private String webconsoleMaxNum;

@Autowired

private CloudNetworkDeviceMapper cloudNetworkDeviceMapper;

public static final String NETWORK\_TYPE\_IN = "0";

public static final String NETWORK\_TYPE\_OUT = "1";

public static final String GATEWAY\_TYPE = "2";

@Autowired

private IpMapper ipMapper;

@Autowired

private PublicIpMapper publicIpMapper;

@Autowired

private NetworkContainerMacMapper networkContainerMacMapper;

@Autowired

private IpBusinessMapper ipBusinessMapper;

@Autowired

private UserInfoMapper userInfoMapper;

@Autowired

private CloudContainerAssetsMapper cloudContainerAssetsMapper;

public ResponseBean<String> getShellBoxUrl(String containerId) {

ResponseBean<String> responseBean = new ResponseBean<>();

CloudNetworkDeviceExample example = new CloudNetworkDeviceExample();

example.createCriteria().andCloudAssetsIdEqualTo(containerId).andStatusNotEqualTo(Constants.DELETED);

List<CloudNetworkDevice> cloudNetworkDevices = cloudNetworkDeviceMapper.selectByExample(example);

for (CloudNetworkDevice cloudNetworkDevice : cloudNetworkDevices) {

if (cloudNetworkDevice.getType() == 1) {//外网

PublicIp publicIp = publicIpMapper.selectByPrimaryKey(cloudNetworkDevice.getIpId());

responseBean.setData(publicIp.getIp()+":4200");

responseBean.setCodeAndMsg(ErrorCode.SUCCESS\_code.getCode(),ErrorCode.SUCCESS\_code.getMsg());

return responseBean;

} else if (cloudNetworkDevice.getType() == 0) {

//查内网ip,获取内网网段

Ip ip = ipMapper.selectByPrimaryKey(cloudNetworkDevice.getIpId());

CloudNetworkDeviceExample example1 = new CloudNetworkDeviceExample();

example1.createCriteria().andIpIdEqualTo(ip.getIpParaId()).andStatusNotEqualTo(Constants.DELETED);

//用内网网段查出网关

List<CloudNetworkDevice> cloudNetworkDevices1 = cloudNetworkDeviceMapper.selectByExample(example1);

CloudNetworkDevice cloudNetworkDevice1 = cloudNetworkDevices1.get(0);

CloudNetworkDeviceExample example2 = new CloudNetworkDeviceExample();

example2.createCriteria().andCloudAssetsIdEqualTo(cloudNetworkDevice1.getCloudAssetsId())

.andTypeEqualTo(1);

//查出网关外网

List<CloudNetworkDevice> cloudNetworkDevices2 = cloudNetworkDeviceMapper.selectByExample(example2);

//查询外网信息

PublicIp publicIp = publicIpMapper.selectByPrimaryKey(cloudNetworkDevices2.get(0).getIpId());

responseBean.setData(publicIp.getIp()+"/"+remDot(ip.getIp()));

responseBean.setCodeAndMsg(ErrorCode.SUCCESS\_code.getCode(),ErrorCode.SUCCESS\_code.getMsg());

return responseBean;

}

}

responseBean.setData("null");

responseBean.setCodeAndMsg(ErrorCode.code\_030106.getCode(),ErrorCode.code\_030106.getMsg());

return responseBean;

}

public ResponseBean<String> getWebconsuleUrl(String containerId) {

ResponseBean<String> responseBean = new ResponseBean<>();

CloudNetworkDeviceExample example = new CloudNetworkDeviceExample();

example.createCriteria().andCloudAssetsIdEqualTo(containerId).andStatusNotEqualTo(Constants.DELETED);

List<CloudNetworkDevice> cloudNetworkDevices = cloudNetworkDeviceMapper.selectByExample(example);

for (CloudNetworkDevice cloudNetworkDevice : cloudNetworkDevices) {

if (cloudNetworkDevice.getType().equals("1")) {//外网

PublicIp publicIp = publicIpMapper.selectByPrimaryKey(cloudNetworkDevice.getIpId());

responseBean.setData(publicIp.getIp()+":22");

responseBean.setCodeAndMsg(ErrorCode.SUCCESS\_code.getCode(),ErrorCode.SUCCESS\_code.getMsg());

return responseBean;

} else if (cloudNetworkDevice.getType().equals("0")) {

Ip ip = ipMapper.selectByPrimaryKey(cloudNetworkDevice.getIpId());

responseBean.setData(ip.getIp()+":22");

responseBean.setCodeAndMsg(ErrorCode.SUCCESS\_code.getCode(),ErrorCode.SUCCESS\_code.getMsg());

return responseBean;

}

}

responseBean.setData("null");

responseBean.setCodeAndMsg(ErrorCode.code\_030106.getCode(),ErrorCode.code\_030106.getMsg());

return responseBean;

}

private String remDot(String ip) {

return ip.replaceAll("\\.","");

}

/\*\*\*

\* 组装参数

\* @param containerId

\* @return

\*/

public List<BatchNetworkParams> getPatchNetworkParams(String containerId) {

List<BatchNetworkParams> paramsList = new ArrayList<>();

//CloudContainerAssets cloudContainerAssets = cloudContainerAssetsMapper.selectByPrimaryKey(containerId);

CloudNetworkDeviceExample example = new CloudNetworkDeviceExample();

example.createCriteria().

andCloudAssetsIdEqualTo(containerId).

andStatusNotEqualTo(Constants.DELETED);

List<CloudNetworkDevice> cloudNetworkDevices = cloudNetworkDeviceMapper.selectByExample(example);

for (CloudNetworkDevice cloudNetworkDevice : cloudNetworkDevices) {

NetworkContainerMac networkContainerMac = networkContainerMacMapper.selectByPrimaryKey(cloudNetworkDevice.getInsideMacId());

if (networkContainerMac == null) {

return null;

}

BatchNetworkParams params = new BatchNetworkParams();

int type = cloudNetworkDevice.getType();

String deviceId = cloudNetworkDevice.getId();

params.setDeviceId(deviceId);

if (type == 1) {//baseNetwork

//PublicIp publicIp = publicIpMapper.selectByPrimaryKey(cloudNetworkDevice.getIpId());

List<Map<String, Object>> pubIpInfoNotEndTime = ipBusinessMapper.findPubIpInfoNotEndTime(cloudNetworkDevice.getIpId());

if (pubIpInfoNotEndTime != null && pubIpInfoNotEndTime.size() > 0) {

Map<String, Object> map = pubIpInfoNotEndTime.get(0);

params.setBridge(property.getProperty("pub", "bridgeOut"));

params.setGateway(map.get("gateway").toString());

params.setIp(map.get("ip").toString());

params.setIp\_public("1");

params.setMac(networkContainerMac.getMac());

params.setVlan("");//baseNetwork vlan is null

params.setMask("24");

String[] route = new String[1];

route[0] = map.get("gateway").toString()+"/24";

params.setRoute(route);

paramsList.add(params);

}

}else {

Map<String, Object> ipParam = new HashMap<>();

ipParam.put("ipId",cloudNetworkDevice.getIpId());

List<Map<String, Object>> ipInfo = ipBusinessMapper.findIpInfo(ipParam);

if (ipInfo == null || ipInfo.isEmpty()) {

logger.error("未找到ipId为{}的内网ip,请查询数据",cloudNetworkDevice.getIpId());

throw new RuntimeException("未找到ipId为"+cloudNetworkDevice.getIpId()+"的内网ip,请查询数据");

}

Map<String, Object> ip = ipInfo.get(0);

params.setBridge(property.getProperty("pub", "bridgeIn"));

params.setGateway(String.valueOf(ip.get("gateway")));

params.setIp(String.valueOf(ip.get("ip")));

params.setIp\_public("0");

params.setMac(networkContainerMac.getMac());

params.setVlan(String.valueOf(ip.get("vlan\_name")));

params.setMask("24");

String[] route = new String[1];

route[0] = String.valueOf(ip.get("gateway"))+"/24";

params.setRoute(route);

paramsList.add(params);

}

}

return paramsList;

}

public ResponseBean<String> getEnAddr(String vmAddr,String userId) {

UserInfoExample example = new UserInfoExample();

example.createCriteria().andUserIdEqualTo(userId);

List<UserInfo> userInfos = userInfoMapper.selectByExample(example);

UserInfo userInfo = userInfos.get(0);

//String id = userInfo.getId();

Integer hashNum = 0;//todo 以后修改此数值

int index = hashNum/Integer.parseInt(property.getProperty("network", "webconsoleMaxNum"))+1; //发送到webconsole上的序号

//http://10.10.25.39:8080/console/chksshdaddr?vm\_addr=10.10.25.45:22

//{"ok":true,"msg":"","data":{"en\_addr":"t-npNFuEf5boMzO8lyA=","sshd\_addr":"10.10.25.82:22"}

String url = property.getProperty("network", "webConsoleSshdaddr") + "ws"+index + "/console/chksshdaddr";

String result = HttpUtils.sendGet(url, "vm\_addr="+vmAddr);

logger.info("result = {},url={},vm\_addr={}",result,url,vmAddr);

JSONObject jsonObject = JSON.parseObject(result);

String en\_addr = jsonObject.getJSONObject("data").getString("en\_addr");

//http://10.10.25.39/ws01/console/login/t-npNFuEf5boMzO8lyA=

String gotoUrl = property.getProperty("network", "webConsoleSshdaddr") + "ws"+index +"/console/login/"+en\_addr;

ResponseBean<String> responseBean = new ResponseBean<>();

responseBean.setData(gotoUrl);

return responseBean;

}

/\*\*

\* 查询网关

\* @param userId

\* @param serverCenterId

\* @return

\*/

public ResponseBean<List<CloudContainerAssets>> getUserGateway(String userId, String serverCenterId){

ResponseBean<List<CloudContainerAssets>> result = new ResponseBean<>();

CloudContainerAssetsExample example = new CloudContainerAssetsExample();

example.createCriteria().andUserIdEqualTo(userId).andServerCenterIdEqualTo(serverCenterId).andIsGatewayEqualTo(1).andStatusIn(Arrays.asList(1, 2));

List<CloudContainerAssets> list = this.cloudContainerAssetsMapper.selectByExample(example);

result.setData(list);

result.setCodeAndMsg(ErrorCode.SUCCESS.getCode(), ErrorCode.SUCCESS.getMsg());

return result;

}

}

package com.yingchong.service.yccloud.network.service;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.Date;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

import javax.annotation.Resource;

import com.yingchong.common.yccloud.commonapi.util.CloudProperty;

import org.apache.commons.lang.StringUtils;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.amqp.AmqpException;

import org.springframework.amqp.rabbit.core.RabbitTemplate;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import com.alibaba.fastjson.JSON;

import com.alibaba.fastjson.JSONArray;

import com.alibaba.fastjson.JSONObject;

import com.github.pagehelper.PageHelper;

import com.github.pagehelper.PageInfo;

import com.yingchong.common.yccloud.commonapi.bean.NoticeBean;

import com.yingchong.common.yccloud.commonapi.bean.RabbitmqParam;

import com.yingchong.common.yccloud.commonapi.constants.Constants;

import com.yingchong.common.yccloud.commonapi.constants.exception\_code.ErrorCode;

import com.yingchong.common.yccloud.commonapi.mapper.AppGroupMapper;

import com.yingchong.common.yccloud.commonapi.mapper.CloudContainerAssetsMapper;

import com.yingchong.common.yccloud.commonapi.mapper.CloudNetworkDeviceMapper;

import com.yingchong.common.yccloud.commonapi.mapper.ClusterMapper;

import com.yingchong.common.yccloud.commonapi.mapper.InstanceMapper;

import com.yingchong.common.yccloud.commonapi.mapper.IntrIpParaMapper;

import com.yingchong.common.yccloud.commonapi.mapper.IpMapper;

import com.yingchong.common.yccloud.commonapi.mapper.NetworkContainerMacMapper;

import com.yingchong.common.yccloud.commonapi.mapper.NetworkSecurityGroupMapper;

import com.yingchong.common.yccloud.commonapi.mapper.NetworkSecurityGroupTagMapper;

import com.yingchong.common.yccloud.commonapi.mapper.NetworkSecurityTagRelationMapper;

import com.yingchong.common.yccloud.commonapi.mapper.NetworkSecuritygroupIpDetailMapper;

import com.yingchong.common.yccloud.commonapi.mapper.PublicIpMapper;

import com.yingchong.common.yccloud.commonapi.mapper.VlanMapper;

import com.yingchong.common.yccloud.commonapi.model.AppGroup;

import com.yingchong.common.yccloud.commonapi.model.AppGroupExample;

import com.yingchong.common.yccloud.commonapi.model.CloudContainerAssets;

import com.yingchong.common.yccloud.commonapi.model.CloudContainerAssetsExample;

import com.yingchong.common.yccloud.commonapi.model.CloudNetworkDevice;

import com.yingchong.common.yccloud.commonapi.model.CloudNetworkDeviceExample;

import com.yingchong.common.yccloud.commonapi.model.Cluster;

import com.yingchong.common.yccloud.commonapi.model.ClusterExample;

import com.yingchong.common.yccloud.commonapi.model.Instance;

import com.yingchong.common.yccloud.commonapi.model.InstanceExample;

import com.yingchong.common.yccloud.commonapi.model.IntrIpPara;

import com.yingchong.common.yccloud.commonapi.model.Ip;

import com.yingchong.common.yccloud.commonapi.model.NetworkContainerMac;

import com.yingchong.common.yccloud.commonapi.model.NetworkSecurityGroup;

import com.yingchong.common.yccloud.commonapi.model.NetworkSecurityGroupExample;

import com.yingchong.common.yccloud.commonapi.model.NetworkSecurityGroupTag;

import com.yingchong.common.yccloud.commonapi.model.NetworkSecurityTagRelation;

import com.yingchong.common.yccloud.commonapi.model.NetworkSecurityTagRelationExample;

import com.yingchong.common.yccloud.commonapi.model.NetworkSecuritygroupIpDetail;

import com.yingchong.common.yccloud.commonapi.model.NetworkSecuritygroupIpDetailExample;

import com.yingchong.common.yccloud.commonapi.model.PublicIp;

import com.yingchong.common.yccloud.commonapi.model.Vlan;

import com.yingchong.common.yccloud.commonapi.util.MD5;

import com.yingchong.common.yccloud.commonapi.util.RedisUtils;

import com.yingchong.common.yccloud.remote.bean.ResponseBean;

import com.yingchong.service.yccloud.network.mapper.SecurityGroupBusinessMapper;

import com.yingchong.service.yccloud.network.util.IpUtils;

/\*\*

\*

\* =======================================================

\*

\* @Version ：0.0.1

\* @Description ：安全组业务service

\* ========================================================

\*/

@Service

public class SecurityGroupBusinessService {

private static final Logger logger = LoggerFactory.getLogger(SecurityGroupBusinessService.class);

private static final String SECURITY\_COUNT = "SECURITY\_COUNT\_";

@Autowired

private CloudProperty property;

@Autowired

private SecurityGroupBusinessMapper securityGroupBusinessMapper;

@Autowired

private NetworkSecurityGroupTagMapper networkSecurityGroupTagMapper;

@Autowired

private NetworkSecurityGroupMapper networkSecurityGroupMapper;

@Autowired

private CloudNetworkDeviceMapper cloudNetworkDeviceMapper;

// @Resource(name="bridgeIn")

// private String bridgeIn;

// @Resource(name="bridgeOut")

// private String bridgeOut;

@Autowired

private IpMapper ipMapper;

@Autowired

private PublicIpMapper publicIpMapper;

@Autowired

private NetworkContainerMacMapper networkContainerMacMapper;

@Autowired

private IntrIpParaMapper intrIpParaMapper;

@Autowired

private VlanMapper vlanMapper;

@Autowired

private RedisUtils redisUtils;

@Autowired

private NetworkSecuritygroupIpDetailMapper networkSecuritygroupIpDetailMapper;

@Autowired

private CloudContainerAssetsMapper cloudContainerAssetsMapper;

@Autowired

private RabbitTemplate rabbitTemplate;

@Autowired

private InstanceMapper instanceMapper;

@Autowired

private NetworkSecurityTagRelationMapper networkSecurityTagRelationMapper;

@Autowired

private ClusterMapper clusterMapper;

@Autowired

private AppGroupMapper appGroupMapper;

/\*\*

\* 新增安全组

\* @Description

\* {

\* "tag": {

\* "tagname" : "安全组名称",

\* "description": "描述"

\* },

\* "group": [{

\* "securityGroupName": "规则名",

\* "protocolType": "协议类型:0 tcp/1 udp/2 icmp",

\* "priority": "优先级",

\* "behavior": "行为:0 接受/1 拒绝",

\* "startPort": "起始端口",

\* "endPort": "结束端口",

\* "aimsIp": "目标ip"

\* }]

\* }

\* @param obj

\* @return

\*/

public ResponseBean<Void> addSecurityGroup(JSONObject obj, String userId){

ResponseBean<Void> result = new ResponseBean<>();

Date now = new Date();

JSONObject tagJson = obj.getJSONObject("tag");

NetworkSecurityGroupTag tag = JSON.toJavaObject(tagJson, NetworkSecurityGroupTag.class);

tag.setUserId(userId);

tag.setCreateTime(now);

this.networkSecurityGroupTagMapper.insert(tag);

JSONArray groupArray = obj.getJSONArray("group");

for (int i=0; i < groupArray.size(); i++){

JSONObject groupJson = groupArray.getJSONObject(i);

NetworkSecurityGroup group = JSON.toJavaObject(groupJson, NetworkSecurityGroup.class);

group.setGroupTagId(tag.getId());

group.setUserId(userId);

group.setCreateTime(now);

this.networkSecurityGroupMapper.insert(group);

}

result.setRetCode(ErrorCode.SUCCESS.getCode());

result.setRetMsg(ErrorCode.SUCCESS.getMsg());

return result;

}

/\*\*

\* 给安全组添加规则

\* @Description

\* {

\* "tagId": "tagId",

\* "group": [{

\* "securityGroupName": "规则名",

\* "protocolType": "协议类型:0 tcp/1 udp/2 icmp",

\* "priority": "优先级",

\* "behavior": "行为:0 接受/1 拒绝",

\* "startPort": "起始端口",

\* "endPort": "结束端口",

\* "aimsIp": "目标ip"

\* }]

\* }

\* han.zhifeng

\* @param obj

\* @return

\*/

public ResponseBean<Void> addGroupRole(JSONObject obj){

ResponseBean<Void> result = new ResponseBean<>();

String tagId = obj.getString("tagId");

JSONArray groupArray = obj.getJSONArray("group");

NetworkSecuritygroupIpDetailExample example = new NetworkSecuritygroupIpDetailExample();

example.createCriteria().andTagIdEqualTo(tagId);

List<NetworkSecuritygroupIpDetail> list = this.networkSecuritygroupIpDetailMapper.selectByExample(example);

for (int i=0; i < groupArray.size(); i++){

JSONObject groupJson = groupArray.getJSONObject(i);

NetworkSecurityGroup group = JSON.toJavaObject(groupJson, NetworkSecurityGroup.class);

group.setGroupTagId(tagId);

group.setCreateTime(new Date());

this.networkSecurityGroupMapper.insert(group);

try {

for (NetworkSecuritygroupIpDetail s : list){

CloudNetworkDeviceExample deviceExample = new CloudNetworkDeviceExample();

deviceExample.createCriteria().andStatusEqualTo(0).andTypeEqualTo(s.getType()).andIpIdEqualTo(s.getIpId());

List<CloudNetworkDevice> deviceList = this.cloudNetworkDeviceMapper.selectByExample(deviceExample);

CloudNetworkDevice device = deviceList.get(0);

NetworkContainerMac mac = this.networkContainerMacMapper.selectByPrimaryKey(device.getInsideMacId());

CloudContainerAssets container = this.cloudContainerAssetsMapper.selectByPrimaryKey(device.getCloudAssetsId());

String ip = null;

String vlan = null;

if (s.getType() == 0){

Ip nip = this.ipMapper.selectByPrimaryKey(s.getIpId());

IntrIpPara para = this.intrIpParaMapper.selectByPrimaryKey(nip.getIpParaId());

Vlan vlanBean = this.vlanMapper.selectByPrimaryKey(para.getVlanId());

ip = nip.getIp();

vlan = vlanBean.getVlanName();

}else{

PublicIp ipInfo = this.publicIpMapper.selectByPrimaryKey(s.getIpId());

ip = ipInfo.getIp();

}

Map<String, Object> paramMap = new HashMap<>();

if (s.getType() == 0){

paramMap.put("bridge\_name", property.getProperty("pub", "bridgeIn"));

}else{

paramMap.put("bridge\_name", property.getProperty("pub", "bridgeOut"));

}

paramMap.put("actions", getAcrion(group.getBehavior()));

paramMap.put("dl\_type", getDlType(group.getProtocolType()));

paramMap.put("priority", group.getPriority());

//拆分ip

List<String> ips = IpUtils.splitIp(group.getAimsIp());

paramMap.put("operator\_ips", ips);

paramMap.put("bind\_ip", ip);

paramMap.put("bind\_mac", mac.getMac());

// paramMap.put("direction", getDirection(group.getDirection()));

paramMap.put("start\_port", group.getStartPort());

paramMap.put("end\_port", group.getEndPort());

if(s.getType() == 0){

paramMap.put("vlan", vlan);

}

Map<String, Object> baseMap = new HashMap<>();

baseMap.put("serverId", container.getServerAssetsId());

RabbitmqParam rp = new RabbitmqParam();

rp.setActiveType(0);//不需回调

rp.setBaseParam(baseMap);

rp.setMethodType(Constants.ADD\_INTERFLOW\_LIMIT);

rp.setMethodParam(paramMap);

logger.info("==============");

logger.info("添加安全组参数：【{}】", rp.toJson());

logger.info("==============");

rabbitTemplate.convertAndSend(Constants.TASK\_QUEUE + container.getServerAssetsId(), rp.toJson());

}

} catch (AmqpException e) {

logger.error("关联出错", e);

}

}

result.setRetCode(ErrorCode.SUCCESS.getCode());

result.setRetMsg(ErrorCode.SUCCESS.getMsg());

return result;

}

/\*\*

\* 编辑安全组规则

\* {

\* "tagId": "tagId",

\* "group": {

\* "id": "id"

\* "securityGroupName": "规则名",

\* "protocolType": "协议类型:0 tcp/1 udp/2 icmp",

\* "priority": "优先级",

\* "behavior": "行为:0 接受/1 拒绝",

\* "startPort": "起始端口",

\* "endPort": "结束端口",

\* "aimsIp": "目标ip"

\* }

\* }

\* @param obj

\* @return

\*/

public ResponseBean<Void> editGroupRole(JSONObject obj){

String tagId = obj.getString("tagId");

JSONObject group = obj.getJSONObject("group");

//删除原来规则

JSONArray groupIds = new JSONArray();

groupIds.add(group.getString("id"));

JSONObject deleteParam = new JSONObject();

deleteParam.put("tagId", tagId);

deleteParam.put("groupIds", groupIds);

ResponseBean<Void> deleteResult = deleteGroupRole(deleteParam);

if (!ErrorCode.SUCCESS.getCode().equals(deleteResult.getRetCode())) {

return deleteResult;

}

//加新规则

JSONObject addParam = new JSONObject();

addParam.put("tagId", tagId);

JSONArray groupList = new JSONArray();

groupList.add(group);

addParam.put("group", groupList);

return addGroupRole(addParam);

}

/\*\*

\* 安全组删除规则

\* {

\* "tagId": "安全组id",

\* "groupIds": [1,2] 安全组规则id

\* }

\* @Description

\* @param obj

\* @return

\*/

public ResponseBean<Void> deleteGroupRole(JSONObject obj){

ResponseBean<Void> result = new ResponseBean<>();

String tagId = obj.getString("tagId");

JSONArray groupArray = obj.getJSONArray("groupIds");

List<String> ids = new ArrayList<>();

for (int i=0; i < groupArray.size(); i++){

ids.add(groupArray.getString(i));

}

NetworkSecurityGroupExample example = new NetworkSecurityGroupExample();

example.createCriteria().andGroupTagIdEqualTo(tagId).andIdIn(ids);

this.networkSecurityGroupMapper.deleteByExample(example);

NetworkSecuritygroupIpDetailExample detailExample = new NetworkSecuritygroupIpDetailExample();

detailExample.createCriteria().andSecurityGroupIdIn(ids);

List<NetworkSecuritygroupIpDetail> detailList = this.networkSecuritygroupIpDetailMapper.selectByExample(detailExample);

for (NetworkSecuritygroupIpDetail detail : detailList){

Map<String, Object> paramMap = new HashMap<>();

paramMap.put("bridge\_name", getBridgeName(detail.getType()));

paramMap.put("cookie", detail.getCookie());

Map<String, Object> baseMap = new HashMap<>();

baseMap.put("id", detail.getId());

RabbitmqParam rp = new RabbitmqParam();

rp.setActiveType(Constants.ACTIVE\_TYPE\_UNBIND\_SECURITY\_GROUP);

rp.setBaseParam(baseMap);

rp.setMethodType(Constants.DELETE\_INTERFLOW\_LIMIT);

rp.setMethodParam(paramMap);

logger.info("==============");

logger.info("删除ip绑定的安全组：【{}】", rp.toJson());

logger.info("==============");

rabbitTemplate.convertAndSend(Constants.TASK\_QUEUE + detail.getServerId(), rp.toJson());

}

result.setRetCode(ErrorCode.SUCCESS.getCode());

result.setRetMsg(ErrorCode.SUCCESS.getMsg());

return result;

}

/\*\*

\*

\* @Description

\* {

\* "groupId": "安全组的id",

\* "ipId": [{"id": "1", "type": "0内网 1外网"},{"id": "2", "type": "0内网 1外网"}]//安全组id数组

\* }

\* han.zhifeng

\* @param obj

\* @return

\*/

public ResponseBean<Void> groupBindIp(JSONObject obj){

ResponseBean<Void> result = new ResponseBean<>();

String groupTagId = obj.getString("groupId");

JSONArray ipIds = obj.getJSONArray("ipId");

// String type = obj.getString("type");//0内网 1外网

NetworkSecurityGroupExample groupExample = new NetworkSecurityGroupExample();

groupExample.createCriteria().andGroupTagIdEqualTo(groupTagId);

List<NetworkSecurityGroup> list = this.networkSecurityGroupMapper.selectByExample(groupExample);

Integer groupCount = list.size();

Integer ipCount = ipIds.size();

Integer count = groupCount \* ipCount;

String countKey = MD5.uuid();

for (NetworkSecurityGroup group : list){

for(int i = 0; i < ipIds.size(); i++){

JSONObject ipJson = ipIds.getJSONObject(i);

CloudNetworkDeviceExample deviceExample = new CloudNetworkDeviceExample();

deviceExample.createCriteria().andStatusEqualTo(0).andTypeEqualTo(ipJson.getInteger("type")).andIpIdEqualTo(ipJson.getString("id"));

List<CloudNetworkDevice> deviceList = this.cloudNetworkDeviceMapper.selectByExample(deviceExample);

CloudNetworkDevice device = deviceList.get(0);

NetworkContainerMac mac = this.networkContainerMacMapper.selectByPrimaryKey(device.getInsideMacId());

CloudContainerAssets container = this.cloudContainerAssetsMapper.selectByPrimaryKey(device.getCloudAssetsId());

String ip = null;

String vlan = null;

if ("0".equals(ipJson.getString("type"))){//内网

Ip nip = this.ipMapper.selectByPrimaryKey(ipIds.getString(i));

IntrIpPara para = this.intrIpParaMapper.selectByPrimaryKey(nip.getIpParaId());

Vlan vlanBean = this.vlanMapper.selectByPrimaryKey(para.getVlanId());

ip = nip.getIp();

vlan = vlanBean.getVlanName();

}else{//公网

PublicIp ipInfo = this.publicIpMapper.selectByPrimaryKey(ipIds.getString(i));

ip = ipInfo.getIp();

}

Map<String, Object> paramMap = new HashMap<>();

if ("0".equals(ipJson.getString("type"))){//内网

paramMap.put("bridge\_name", property.getProperty("pub", "bridgeIn"));

paramMap.put("vlan", vlan);

}else{

paramMap.put("bridge\_name", property.getProperty("pub", "bridgeOut"));

}

paramMap.put("actions", getAcrion(group.getBehavior()));

paramMap.put("dl\_type", getDlType(group.getProtocolType()));

paramMap.put("priority", group.getPriority());

//拆分ip

List<String> ips = IpUtils.splitIp(group.getAimsIp());

paramMap.put("operator\_ips", ips);

paramMap.put("bind\_ip", ip);

paramMap.put("bind\_mac", mac.getMac());

// paramMap.put("direction", getDirection(group.getDirection()));

paramMap.put("start\_port", group.getStartPort());

paramMap.put("end\_port", group.getEndPort());

Map<String, Object> baseMap = new HashMap<>();

baseMap.put("count", count);

baseMap.put("ipId", ipIds.getInteger(i));

baseMap.put("ipIds", ipIds);

baseMap.put("type", ipJson.getString("type"));

baseMap.put("groupTag", group.getGroupTagId());

baseMap.put("groupId", group.getId());

baseMap.put("countKey", countKey);

baseMap.put("userId", group.getUserId());

baseMap.put("serverId", container.getServerAssetsId());

RabbitmqParam rp = new RabbitmqParam();

rp.setActiveType(Constants.ACTIVE\_TYPE\_BIND\_SECURITY\_GROUP);

rp.setBaseParam(baseMap);

rp.setMethodType(Constants.ADD\_INTERFLOW\_LIMIT);

rp.setMethodParam(paramMap);

logger.info("==============");

logger.info("添加安全组参数：【{}】", rp.toJson());

logger.info("==============");

rabbitTemplate.convertAndSend(Constants.TASK\_QUEUE + container.getServerAssetsId(), rp.toJson());

}

}

result.setRetCode(ErrorCode.SUCCESS.getCode());

result.setRetMsg(ErrorCode.SUCCESS.getMsg());

return result;

}

/\*\*

\* ip关联安全组回调

\* @Description

\* han.zhifeng

\* @param result

\*/

public void bindIpForGroupActive(JSONObject result){

logger.info("ip关联安全组回调：【{}】", result);

JSONObject r = result.getJSONObject("result");

JSONObject baseParam = result.getJSONObject("baseParam");

Integer activeType = result.getInteger("activeType");

long c = this.redisUtils.incr(SECURITY\_COUNT + baseParam.getString("countKey"));

logger.info("次数：【{}】", c);

if (Constants.SUCCESS.equals(r.getString("status"))){

JSONObject data = r.getJSONObject("data");

NetworkSecuritygroupIpDetail detail = new NetworkSecuritygroupIpDetail();

detail.setSecurityGroupId(baseParam.getString("groupId"));

detail.setTagId(baseParam.getString("groupTag"));

detail.setIpId(baseParam.getString("ipId"));

detail.setType(baseParam.getInteger("type"));

detail.setCookie(data.getString("cookie"));

detail.setServerId(baseParam.getString("serverId"));

detail.setRelationId(baseParam.getString("relationId"));

this.networkSecuritygroupIpDetailMapper.insert(detail);

if (c == baseParam.getLongValue("count")){

logger.info("关联成功，通知前端");

NoticeBean nb = new NoticeBean();

nb.setUserId(baseParam.getString("userId"));

nb.setActiveType(activeType);

nb.setRetCode(ErrorCode.SUCCESS.getCode());

nb.setRetMessage(ErrorCode.SUCCESS.getMsg());

rabbitTemplate.convertAndSend(Constants.NOTICE\_WEB\_QUEUE,"", nb.toJson());

this.redisUtils.del(SECURITY\_COUNT + baseParam.getString("countKey"));

}

}else{

//下发出错的话cookie为空

NetworkSecuritygroupIpDetail detail = new NetworkSecuritygroupIpDetail();

detail.setSecurityGroupId(baseParam.getString("groupId"));

detail.setTagId(baseParam.getString("groupTag"));

detail.setIpId(baseParam.getString("ipId"));

detail.setType(baseParam.getInteger("type"));

// detail.setCookie(data.getString("cookie"));

detail.setServerId(baseParam.getString("serverId"));

this.networkSecuritygroupIpDetailMapper.insert(detail);

JSONObject error = r.getJSONObject("error");

logger.error("ip关联安全组出错：【{}】 参数【{}】", error, baseParam);

NoticeBean nb = new NoticeBean();

nb.setUserId(baseParam.getString("userId"));

nb.setActiveType(activeType);

nb.setRetCode(ErrorCode.CODE\_030102.getCode());

nb.setRetMessage(ErrorCode.CODE\_030102.getMsg() + "安全组：" + baseParam.getString("groupId"));

rabbitTemplate.convertAndSend(Constants.NOTICE\_WEB\_QUEUE,"", nb.toJson());

}

}

/\*\*

\* 解绑ip的安全组

\* @Description

\* {

"groupId": "安全组id",

"type": "1:分组、2:集群、3：容器",

"ids": ["1","2"]//id集合

}

\* han.zhifeng

\* @param obj

\* @return

\*/

public ResponseBean<Void> unBindIpGroup(JSONObject obj){

logger.info("解绑ip绑定的安全组：【{}】", obj);

ResponseBean<Void> result = new ResponseBean<>();

String groupId = obj.getString("groupId");

JSONArray ipId = obj.getJSONArray("ipId");

for (int i = 0; i < ipId.size(); i++){

NetworkSecuritygroupIpDetailExample example = new NetworkSecuritygroupIpDetailExample();

example.createCriteria().andTagIdEqualTo(groupId).andIpIdEqualTo(ipId.getString(i));

this.networkSecuritygroupIpDetailMapper.deleteByExample(example);

NetworkSecuritygroupIpDetailExample detailExample = new NetworkSecuritygroupIpDetailExample();

detailExample.createCriteria().andTagIdEqualTo(groupId).andIpIdEqualTo(ipId.getString(i));

List<NetworkSecuritygroupIpDetail> detailList = this.networkSecuritygroupIpDetailMapper.selectByExample(detailExample);

for (NetworkSecuritygroupIpDetail detail : detailList){

Map<String, Object> paramMap = new HashMap<>();

paramMap.put("bridge\_name", getBridgeName(detail.getType()));

paramMap.put("cookie", detail.getCookie());

Map<String, Object> baseMap = new HashMap<>();

baseMap.put("id", detail.getId());

RabbitmqParam rp = new RabbitmqParam();

rp.setActiveType(Constants.ACTIVE\_TYPE\_UNBIND\_SECURITY\_GROUP);

rp.setBaseParam(baseMap);

rp.setMethodType(Constants.DELETE\_INTERFLOW\_LIMIT);

rp.setMethodParam(paramMap);

logger.info("==============");

logger.info("删除ip绑定的安全组：【{}】", rp.toJson());

logger.info("==============");

rabbitTemplate.convertAndSend(Constants.TASK\_QUEUE + detail.getServerId(), rp.toJson());

}

}

result.setRetCode(ErrorCode.SUCCESS.getCode());

result.setRetMsg(ErrorCode.SUCCESS.getMsg());

return result;

}

/\*\*

\* 解绑ip的安全组

\* @Description

\* {

"groupId": "安全组id",

"deleteArr": [{"type": 1, "id": "2"}, {"type": 1, "id": "2"}]//"type": "1:分组、2:集群、3：容器" id:对应id

}

\* @param obj

\* @return

\*/

public ResponseBean<Void> unBindSecurityGroup(JSONObject obj){

logger.info("解绑安全组：【{}】", obj);

ResponseBean<Void> result = new ResponseBean<>();

String groupId = obj.getString("groupId");

JSONArray arr = obj.getJSONArray("deleteArr");

for (int i=0; i < arr.size(); i++) {

JSONObject json = arr.getJSONObject(i);

NetworkSecurityTagRelationExample example = new NetworkSecurityTagRelationExample();

example.createCriteria().andTagIdEqualTo(groupId).andTargetIdEqualTo(json.getString("id")).andTypeEqualTo(json.getInteger("type"));

List<NetworkSecurityTagRelation> list = this.networkSecurityTagRelationMapper.selectByExample(example);

for (NetworkSecurityTagRelation relation : list) {

NetworkSecuritygroupIpDetailExample detailExample = new NetworkSecuritygroupIpDetailExample();

detailExample.createCriteria().andTagIdEqualTo(groupId).andRelationIdEqualTo(relation.getId());

List<NetworkSecuritygroupIpDetail> detailList = this.networkSecuritygroupIpDetailMapper.selectByExample(detailExample);

for (NetworkSecuritygroupIpDetail detail : detailList){

Map<String, Object> paramMap = new HashMap<>();

paramMap.put("bridge\_name", getBridgeName(detail.getType()));

paramMap.put("cookie", detail.getCookie());

Map<String, Object> baseMap = new HashMap<>();

baseMap.put("id", detail.getId());

RabbitmqParam rp = new RabbitmqParam();

rp.setActiveType(Constants.ACTIVE\_TYPE\_UNBIND\_SECURITY\_GROUP);

rp.setBaseParam(baseMap);

rp.setMethodType(Constants.DELETE\_INTERFLOW\_LIMIT);

rp.setMethodParam(paramMap);

logger.info("==============");

logger.info("删除ip绑定的安全组：【{}】", rp.toJson());

logger.info("==============");

rabbitTemplate.convertAndSend(Constants.TASK\_QUEUE + detail.getServerId(), rp.toJson());

}

this.networkSecurityTagRelationMapper.deleteByPrimaryKey(relation.getId());

}

}

result.setRetCode(ErrorCode.SUCCESS.getCode());

result.setRetMsg(ErrorCode.SUCCESS.getMsg());

return result;

}

/\*\*

\* 删除完全组回调

\* @Description

\* han.zhifeng

\* @param result

\*/

public void unBindIpGroupActive(JSONObject result){

logger.info("ip删除关联安全组回调：【{}】", result);

JSONObject r = result.getJSONObject("result");

JSONObject baseParam = result.getJSONObject("baseParam");

String id = baseParam.getString("id");

if (Constants.SUCCESS.equals(r.getString("status"))){

logger.info("删除安全组成功 network\_securitygroup\_ip\_detail 【{}】", id);

this.networkSecuritygroupIpDetailMapper.deleteByPrimaryKey(id);

}else{

logger.error("删除安全组失败 network\_securitygroup\_ip\_detail 【{}】", id);

}

}

/\*\*

\* 获取用户可用的优先级

\* @Description

\* @param userId

\* @return

\*/

public ResponseBean<List<Integer>> getUserPriority(String userId){

ResponseBean<List<Integer>> result = new ResponseBean<>();

List<Integer> list = this.securityGroupBusinessMapper.getUserPriority(userId);

List<Integer> reList = new ArrayList<>();

for (int i=1; i<=100; i++){

if (!list.contains(i)){

reList.add(i);

}

}

result.setRetCode(ErrorCode.SUCCESS.getCode());

result.setRetMsg(ErrorCode.SUCCESS.getMsg());

result.setData(reList);

return result;

}

/\*\*

\* 查询安全组规则

\* @Description

\* @param tagId

\* @return

\*/

public ResponseBean<PageInfo<NetworkSecurityGroup>> getSecurityGroup(Integer page, Integer pageNum, String tagId){

ResponseBean<PageInfo<NetworkSecurityGroup>> result = new ResponseBean<>();

NetworkSecurityGroupExample example = new NetworkSecurityGroupExample();

example.createCriteria().andGroupTagIdEqualTo(tagId);

PageHelper.startPage(page, pageNum);

List<NetworkSecurityGroup> list = this.networkSecurityGroupMapper.selectByExample(example);

PageInfo<NetworkSecurityGroup> info = new PageInfo<>(list);

result.setRetCode(ErrorCode.SUCCESS.getCode());

result.setRetMsg(ErrorCode.SUCCESS.getMsg());

result.setData(info);

return result;

}

/\*\*

\* 查询内网安全组

\* @Description

\* @param page

\* @param pageNum

\* @param startTime

\* @param endTime

\* @param keyword

\* @param userId

\* @return

\*/

public ResponseBean<PageInfo<Map<String, Object>>> getInGroup(

Integer page, Integer pageNum, String startTime, String endTime, String keyword, String userId, String mobile

){

ResponseBean<PageInfo<Map<String, Object>>> result = new ResponseBean<>();

Map<String, Object> param = new HashMap<>();

if (startTime != null){

startTime = startTime + " 00:00:00";

param.put("startTime", startTime);

}

if(endTime != null){

endTime = endTime + " 23:59:59";

param.put("endTime", endTime);

}

param.put("keyword", keyword);

param.put("userId", userId);

param.put("mobile", mobile);

PageHelper.startPage(page, pageNum);

List<Map<String, Object>> list = this.securityGroupBusinessMapper.getInGroup(param);

PageInfo<Map<String, Object>> info = new PageInfo<>(list);

result.setRetCode(ErrorCode.SUCCESS.getCode());

result.setRetMsg(ErrorCode.SUCCESS.getMsg());

result.setData(info);

return result;

}

/\*\*

\* 查询安全组详情

\* @Description

\* @param id

\* @return

\*/

public ResponseBean<NetworkSecurityGroupTag> getSecurityGroup(String id){

ResponseBean<NetworkSecurityGroupTag> result = new ResponseBean<>();

NetworkSecurityGroupTag group = this.networkSecurityGroupTagMapper.selectByPrimaryKey(id);

result.setRetCode(ErrorCode.SUCCESS.getCode());

result.setRetMsg(ErrorCode.SUCCESS.getMsg());

result.setData(group);

return result;

}

/\*\*

\* 编辑安全组

\* @Description

\* @param param

\* @return

\*/

public ResponseBean<Void> updateSecurityGroup(NetworkSecurityGroupTag param){

ResponseBean<Void> result = new ResponseBean<>();

this.networkSecurityGroupTagMapper.updateByPrimaryKeySelective(param);

result.setRetCode(ErrorCode.SUCCESS.getCode());

result.setRetMsg(ErrorCode.SUCCESS.getMsg());

return result;

}

public ResponseBean<Void> deleteSecurityGroup(String id){

ResponseBean<Void> result = new ResponseBean<>();

// NetworkSecuritygroupIpDetailExample example = new NetworkSecuritygroupIpDetailExample();

// example.createCriteria().andTagIdEqualTo(id);

NetworkSecurityTagRelationExample example = new NetworkSecurityTagRelationExample();

example.createCriteria().andTagIdEqualTo(id);

long count = this.networkSecurityTagRelationMapper.countByExample(example);

if (count > 0){

result.setRetCode(ErrorCode.CODE\_030103.getCode());

result.setRetMsg(ErrorCode.CODE\_030103.getMsg());

return result;

}

this.networkSecurityGroupTagMapper.deleteByPrimaryKey(id);

NetworkSecurityGroupExample deleteExample = new NetworkSecurityGroupExample();

deleteExample.createCriteria().andGroupTagIdEqualTo(id);

this.networkSecurityGroupMapper.deleteByExample(deleteExample);

result.setRetCode(ErrorCode.SUCCESS.getCode());

result.setRetMsg(ErrorCode.SUCCESS.getMsg());

return result;

}

/\*\*

\* 批量删除安全组

\* @Description

\* @param obj

\* @return

\*/

public ResponseBean<Void> deleteSecurityGroup(JSONObject obj){

ResponseBean<Void> result = new ResponseBean<>();

JSONArray array = obj.getJSONArray("id");

List<String> ids = new ArrayList<>();

for(int i=0; i < array.size(); i++){

ids.add(array.getString(i));

}

NetworkSecuritygroupIpDetailExample example = new NetworkSecuritygroupIpDetailExample();

example.createCriteria().andTagIdIn(ids);

long count = this.networkSecuritygroupIpDetailMapper.countByExample(example);

if (count > 0){

result.setRetCode(ErrorCode.CODE\_030103.getCode());

result.setRetMsg(ErrorCode.CODE\_030103.getMsg());

return result;

}

for(String id : ids){

this.networkSecurityGroupTagMapper.deleteByPrimaryKey(id);

NetworkSecurityGroupExample deleteExample = new NetworkSecurityGroupExample();

deleteExample.createCriteria().andGroupTagIdEqualTo(id);

this.networkSecurityGroupMapper.deleteByExample(deleteExample);

}

result.setRetCode(ErrorCode.SUCCESS.getCode());

result.setRetMsg(ErrorCode.SUCCESS.getMsg());

return result;

}

private String getBridgeName(int type){

switch (type) {

case 0:

return property.getProperty("pub", "bridgeIn");

case 1:

return property.getProperty("pub", "bridgeOut");

default:

return null;

}

}

private String getAcrion(int action){

switch (action) {

case 0:

return "normal";

case 1:

return "drop";

default:

return null;

}

}

private String getDlType(int action){

switch (action) {

case 0:

return "tcp";

case 1:

return "udp";

case 2:

return "icmp";

default:

return null;

}

}

/\*\*

\* 深度递归查询应用分组下的容器

\* @param appId

\* @param idList

\*/

private void getContainerIdByAppId(String appId, List<String> idList) {

List<Map<String, Object>> slotList = this.securityGroupBusinessMapper.getAppSlot(appId);

for (Map<String, Object> slot : slotList) {

if (slot.get("id\_type") == null || slot.get("p\_id") == null) {

logger.error("有异常slot：【{}】", JSON.toJSONString(slot));

continue;

}

String type = slot.get("id\_type").toString();

String id = slot.get("p\_id").toString();

if ("3".equals(type)) {

Instance instance = this.instanceMapper.selectByPrimaryKey(id);

idList.add(instance.getContainerId());

}else if ("2".equals(type)) {

InstanceExample example = new InstanceExample();

example.createCriteria().andClusterIdEqualTo(id);

List<Instance> cluster = this.instanceMapper.selectByExample(example);

for (Instance instance : cluster) {

idList.add(instance.getContainerId());

}

}else if ("1".equals(type)){

getContainerIdByAppId(id, idList);

}

}

}

/\*\*

\* 查询集群下的容器

\* @param clusterId

\* @param idList

\*/

private void getContainerIdByCluster(String clusterId, List<String> idList) {

InstanceExample example = new InstanceExample();

example.createCriteria().andClusterIdEqualTo(clusterId);

List<Instance> cluster = this.instanceMapper.selectByExample(example);

for (Instance instance : cluster) {

idList.add(instance.getContainerId());

}

}

private void containerBindSecurityGroup(String tagId, List<String> containerIds, String relationId) {

CloudNetworkDeviceExample example = new CloudNetworkDeviceExample();

example.createCriteria().andStatusEqualTo(0).andTypeIn(Arrays.asList(0,1)).andCloudAssetsIdIn(containerIds);

List<CloudNetworkDevice> networkList = this.cloudNetworkDeviceMapper.selectByExample(example);

NetworkSecurityGroupExample groupExample = new NetworkSecurityGroupExample();

groupExample.createCriteria().andGroupTagIdEqualTo(tagId);

List<NetworkSecurityGroup> list = this.networkSecurityGroupMapper.selectByExample(groupExample);

Integer groupCount = list.size();

Integer ipCount = networkList.size();

Integer count = groupCount \* ipCount;

String countKey = MD5.uuid();

for (NetworkSecurityGroup group : list){

for(CloudNetworkDevice device : networkList){

NetworkContainerMac mac = this.networkContainerMacMapper.selectByPrimaryKey(device.getInsideMacId());

CloudContainerAssets container = this.cloudContainerAssetsMapper.selectByPrimaryKey(device.getCloudAssetsId());

String ip = null;

String vlan = null;

if (device.getType() == 0){//内网

Ip nip = this.ipMapper.selectByPrimaryKey(device.getIpId());

IntrIpPara para = this.intrIpParaMapper.selectByPrimaryKey(nip.getIpParaId());

Vlan vlanBean = this.vlanMapper.selectByPrimaryKey(para.getVlanId());

ip = nip.getIp();

vlan = vlanBean.getVlanName();

}else{//公网

PublicIp ipInfo = this.publicIpMapper.selectByPrimaryKey(device.getIpId());

ip = ipInfo.getIp();

}

Map<String, Object> paramMap = new HashMap<>();

if (device.getType() == 0){//内网

paramMap.put("bridge\_name", property.getProperty("pub", "bridgeIn"));

paramMap.put("vlan", vlan);

}else{

paramMap.put("bridge\_name", property.getProperty("pub", "bridgeOut"));

}

paramMap.put("actions", getAcrion(group.getBehavior()));

paramMap.put("dl\_type", getDlType(group.getProtocolType()));

paramMap.put("priority", group.getPriority());

//拆分ip

List<String> ips = IpUtils.splitIp(group.getAimsIp());

paramMap.put("operator\_ips", ips);

paramMap.put("bind\_ip", ip);

paramMap.put("bind\_mac", mac.getMac());

paramMap.put("start\_port", group.getStartPort());

paramMap.put("end\_port", group.getEndPort());

Map<String, Object> baseMap = new HashMap<>();

baseMap.put("count", count);

baseMap.put("ipId", device.getIpId());

baseMap.put("type", device.getType());

baseMap.put("groupTag", group.getGroupTagId());

baseMap.put("groupId", group.getId());

baseMap.put("countKey", countKey);

baseMap.put("userId", group.getUserId());

baseMap.put("serverId", container.getServerAssetsId());

baseMap.put("relationId", relationId);

RabbitmqParam rp = new RabbitmqParam();

rp.setActiveType(Constants.ACTIVE\_TYPE\_BIND\_SECURITY\_GROUP);

rp.setBaseParam(baseMap);

rp.setMethodType(Constants.ADD\_INTERFLOW\_LIMIT\_BATCH);

rp.setMethodParam(paramMap);

logger.info("==============");

logger.info("添加安全组参数：【{}】", rp.toJson());

logger.info("==============");

rabbitTemplate.convertAndSend(Constants.TASK\_QUEUE + container.getServerAssetsId(), rp.toJson());

}

}

}

/\*\*

\* 绑定安全组

\* {

"groupId": "安全组id",

"type": "1:分组、2:集群、3：容器",

"ids": ["1","2"]//id集合

}

\* @param obj

\* @return

\*/

public ResponseBean<Void> bindSecurityGroup(JSONObject obj){

ResponseBean<Void> result = new ResponseBean<>();

JSONArray ids = obj.getJSONArray("ids");

String groupId = obj.getString("groupId");

Integer type = obj.getInteger("type");

NetworkSecurityTagRelation relation = new NetworkSecurityTagRelation();

relation.setTagId(groupId);

relation.setType(type);

for (int i = 0; i < ids.size(); i++) {

List<String> containerIds = new ArrayList<>();

if (type == 3) {

containerIds.add(ids.getString(i));

}else {

List<String> idList = new ArrayList<>();

if (type == 1) {

getContainerIdByAppId(ids.getString(i), idList);

}else if (type == 2) {

getContainerIdByCluster(ids.getString(i), idList);

}

containerIds.addAll(idList);

}

relation.setTargetId(ids.getString(i));

this.networkSecurityTagRelationMapper.insert(relation);

if (containerIds != null && !containerIds.isEmpty()) {

containerBindSecurityGroup(groupId, containerIds, relation.getId());

}

}

result.setRetCode(ErrorCode.SUCCESS.getCode());

result.setRetMsg(ErrorCode.SUCCESS.getMsg());

return result;

}

/\*\*

\* 查询用户所有主机选择ip

\* @param serverCenterId

\* @return

\*/

public ResponseBean<List<CloudContainerAssets>> selectContainerIp(String serverCenterId, String userId, String name){

ResponseBean<List<CloudContainerAssets>> result = new ResponseBean<>();

CloudContainerAssetsExample example = new CloudContainerAssetsExample();

CloudContainerAssetsExample.Criteria c = example.createCriteria().andIsGatewayEqualTo(0).andStatusIn(Arrays.asList(1,2)).andServerCenterIdEqualTo(serverCenterId).andUserIdEqualTo(userId).andIsDeleteEqualTo(0).andIsDisplayEqualTo(1);

if (name != null) {

c.andWebnameLike("%" + name + "%");

}

List<CloudContainerAssets> list = this.cloudContainerAssetsMapper.selectByExample(example);

result.setRetCode(ErrorCode.SUCCESS.getCode());

result.setRetMsg(ErrorCode.SUCCESS.getMsg());

result.setData(list);

return result;

}

/\*\*

\* 查询用户所有集群选择ip

\* @param serverCenterId

\* @return

\*/

public ResponseBean<List<Cluster>> selectClusterIp(String serverCenterId, String userId, String name){

ResponseBean<List<Cluster>> result = new ResponseBean<>();

ClusterExample example = new ClusterExample();

ClusterExample.Criteria c = example.createCriteria().andServerCenterIdEqualTo(serverCenterId).andUserIdEqualTo(userId).andStatusIn(Arrays.asList(1,2)).andIsDeleteEqualTo(0);

if (name != null) {

c.andClusterNameLike("%" + name + "%");

}

List<Cluster> list = this.clusterMapper.selectByExample(example);

result.setRetCode(ErrorCode.SUCCESS.getCode());

result.setRetMsg(ErrorCode.SUCCESS.getMsg());

result.setData(list);

return result;

}

/\*\*

\* 查询用户的应用分组选择ip

\* @param serverCenterId

\* @return

\*/

public ResponseBean<List<AppGroup>> selectAppGroupIp(String serverCenterId, String userId, String name){

ResponseBean<List<AppGroup>> result = new ResponseBean<>();

AppGroupExample example = new AppGroupExample();

AppGroupExample.Criteria c = example.createCriteria().andServerCenterIdEqualTo(serverCenterId).andUserIdEqualTo(userId).andIsDeleteEqualTo(0);

if (name != null) {

c.andAppGroupNameLike("%" + name + "%");

}

List<AppGroup> list = this.appGroupMapper.selectByExample(example);

result.setRetCode(ErrorCode.SUCCESS.getCode());

result.setRetMsg(ErrorCode.SUCCESS.getMsg());

result.setData(list);

return result;

}

/\*\*

\* 查询ip选取的容器、集群、应用分组的ip

\* {

"type": "1:分组、2:集群、3：容器",

"ids": ["1","2"]//id集合

}

\* @param obj

\* @return

\*/

public ResponseBean<List<String>> getIp(JSONObject obj){

ResponseBean<List<String>> result = new ResponseBean<>();

List<String> containerIds = new ArrayList<>();

JSONArray ids = obj.getJSONArray("ids");

Integer type = obj.getInteger("type");

for (int i = 0; i < ids.size(); i++) {

if (type == 3) {

containerIds.add(ids.getString(i));

}else {

List<String> idList = new ArrayList<>();

if (type == 1) {

getContainerIdByAppId(ids.getString(i), idList);

}else if (type == 2) {

getContainerIdByCluster(ids.getString(i), idList);

}

containerIds.addAll(idList);

}

}

if (!containerIds.isEmpty()) {

List<String> list = getContaienrIp(containerIds);

result.setData(list);

}

result.setRetCode(ErrorCode.SUCCESS.getCode());

result.setRetMsg(ErrorCode.SUCCESS.getMsg());

return result;

}

public ResponseBean<List<String>> getIpType(JSONObject obj){

ResponseBean<List<String>> result = new ResponseBean<>();

List<String> containerIds = new ArrayList<>();

JSONArray ids = obj.getJSONArray("ids");

Integer type = obj.getInteger("type");

for (int i = 0; i < ids.size(); i++) {

if (type == 3) {

containerIds.add(ids.getString(i));

}else {

List<String> idList = new ArrayList<>();

if (type == 1) {

getContainerIdByAppId(ids.getString(i), idList);

}else if (type == 2) {

getContainerIdByCluster(ids.getString(i), idList);

}

containerIds.addAll(idList);

}

}

List<String> list = getContaienrIpType(containerIds);

result.setRetCode(ErrorCode.SUCCESS.getCode());

result.setRetMsg(ErrorCode.SUCCESS.getMsg());

result.setData(list);

return result;

}

private List<String> getContaienrIp(List<String> containerIds){

List<String> ips = new ArrayList<>();

CloudNetworkDeviceExample example = new CloudNetworkDeviceExample();

example.createCriteria().andStatusEqualTo(0).andTypeIn(Arrays.asList(0,1)).andCloudAssetsIdIn(containerIds);

List<CloudNetworkDevice> networkList = this.cloudNetworkDeviceMapper.selectByExample(example);

for(CloudNetworkDevice device : networkList){

if (device.getType() == 0){//内网

Ip nip = this.ipMapper.selectByPrimaryKey(device.getIpId());

ips.add(nip.getIp());

}else{//公网

PublicIp ipInfo = this.publicIpMapper.selectByPrimaryKey(device.getIpId());

ips.add(ipInfo.getIp());

}

}

return ips;

}

private List<String> getContaienrIpType(List<String> containerIds){

if (containerIds.isEmpty()) {

return null;

}

List<String> ips = new ArrayList<>();

CloudNetworkDeviceExample example = new CloudNetworkDeviceExample();

example.createCriteria().andStatusEqualTo(0).andTypeIn(Arrays.asList(0,1)).andCloudAssetsIdIn(containerIds);

List<CloudNetworkDevice> networkList = this.cloudNetworkDeviceMapper.selectByExample(example);

for(CloudNetworkDevice device : networkList){

if (device.getType() == 0){//内网

Ip nip = this.ipMapper.selectByPrimaryKey(device.getIpId());

ips.add(nip.getIp() + "(内)");

}else{//公网

PublicIp ipInfo = this.publicIpMapper.selectByPrimaryKey(device.getIpId());

ips.add(ipInfo.getIp() + "(公)");

}

}

return ips;

}

/\*\*

\* 绑定安全组选择容器

\* @param serverCenterId

\* @param userId

\* @return

\*/

public ResponseBean<List<Map<String, Object>>> getUserContainerBindSecurity(String serverCenterId, String userId, String tagId, String name){

ResponseBean<List<Map<String, Object>>> result = new ResponseBean<>();

Map<String, Object> param = new HashMap<>();

param.put("serverCenterId", serverCenterId);

param.put("userId", userId);

param.put("tagId", tagId);

param.put("name", name);

List<Map<String, Object>> list = this.securityGroupBusinessMapper.getUserContainerBindSecurity(param);

result.setRetCode(ErrorCode.SUCCESS.getCode());

result.setRetMsg(ErrorCode.SUCCESS.getMsg());

result.setData(list);

return result;

}

/\*\*

\* 绑定安全组选择集群

\* @param serverCenterId

\* @param userId

\* @return

\*/

public ResponseBean<List<Map<String, Object>>> getUserClusterBindSecurity(String serverCenterId, String userId, String tagId, String name){

ResponseBean<List<Map<String, Object>>> result = new ResponseBean<>();

Map<String, Object> param = new HashMap<>();

param.put("serverCenterId", serverCenterId);

param.put("userId", userId);

param.put("tagId", tagId);

param.put("name", name);

List<Map<String, Object>> list = this.securityGroupBusinessMapper.getUserClusterBindSecurity(param);

result.setRetCode(ErrorCode.SUCCESS.getCode());

result.setRetMsg(ErrorCode.SUCCESS.getMsg());

result.setData(list);

return result;

}

/\*\*

\* 绑定安全组选择应用分组

\* @param serverCenterId

\* @param userId

\* @return

\*/

public ResponseBean<List<Map<String, Object>>> getUserAppGroupBindSecurity(String serverCenterId, String userId, String tagId, String name){

ResponseBean<List<Map<String, Object>>> result = new ResponseBean<>();

Map<String, Object> param = new HashMap<>();

param.put("serverCenterId", serverCenterId);

param.put("userId", userId);

param.put("tagId", tagId);

param.put("name", name);

List<Map<String, Object>> list = this.securityGroupBusinessMapper.getUserAppGroupBindSecurity(param);

result.setRetCode(ErrorCode.SUCCESS.getCode());

result.setRetMsg(ErrorCode.SUCCESS.getMsg());

result.setData(list);

return result;

}

/\*\*

\* 查询安全组绑定的成员带分页

\* @param page

\* @param pageNum

\* @param tagId

\* @return

\*/

public ResponseBean<PageInfo<Map<String, Object>>> securityGroupBindList(Integer page, Integer pageNum, String tagId){

ResponseBean<PageInfo<Map<String, Object>>> result = new ResponseBean<>();

PageHelper.startPage(page, pageNum);

List<Map<String, Object>> list = this.securityGroupBusinessMapper.securityGroupBindList(tagId);

for (Map<String, Object> map : list) {

JSONObject obj = new JSONObject();

obj.put("type", map.get("type"));

List<Object> ids = new ArrayList<>();

ids.add(map.get("target\_id"));

obj.put("ids", ids);

List<String> ipList = getIpType(obj).getData();

if (ipList != null) {

String ip = StringUtils.join(ipList.toArray(), ",");

map.put("ip", ip);

}

}

PageInfo<Map<String, Object>> info = new PageInfo<>(list);

result.setRetCode(ErrorCode.SUCCESS.getCode());

result.setRetMsg(ErrorCode.SUCCESS.getMsg());

result.setData(info);

return result;

}

}