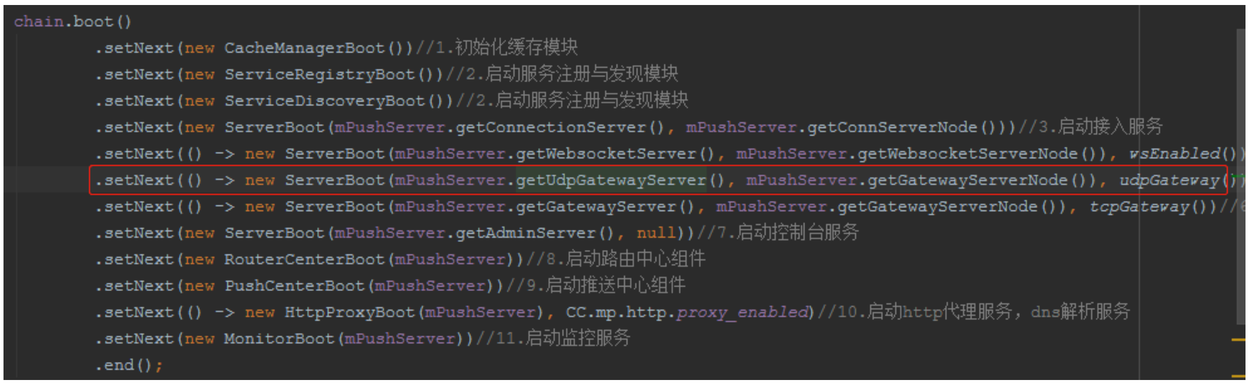
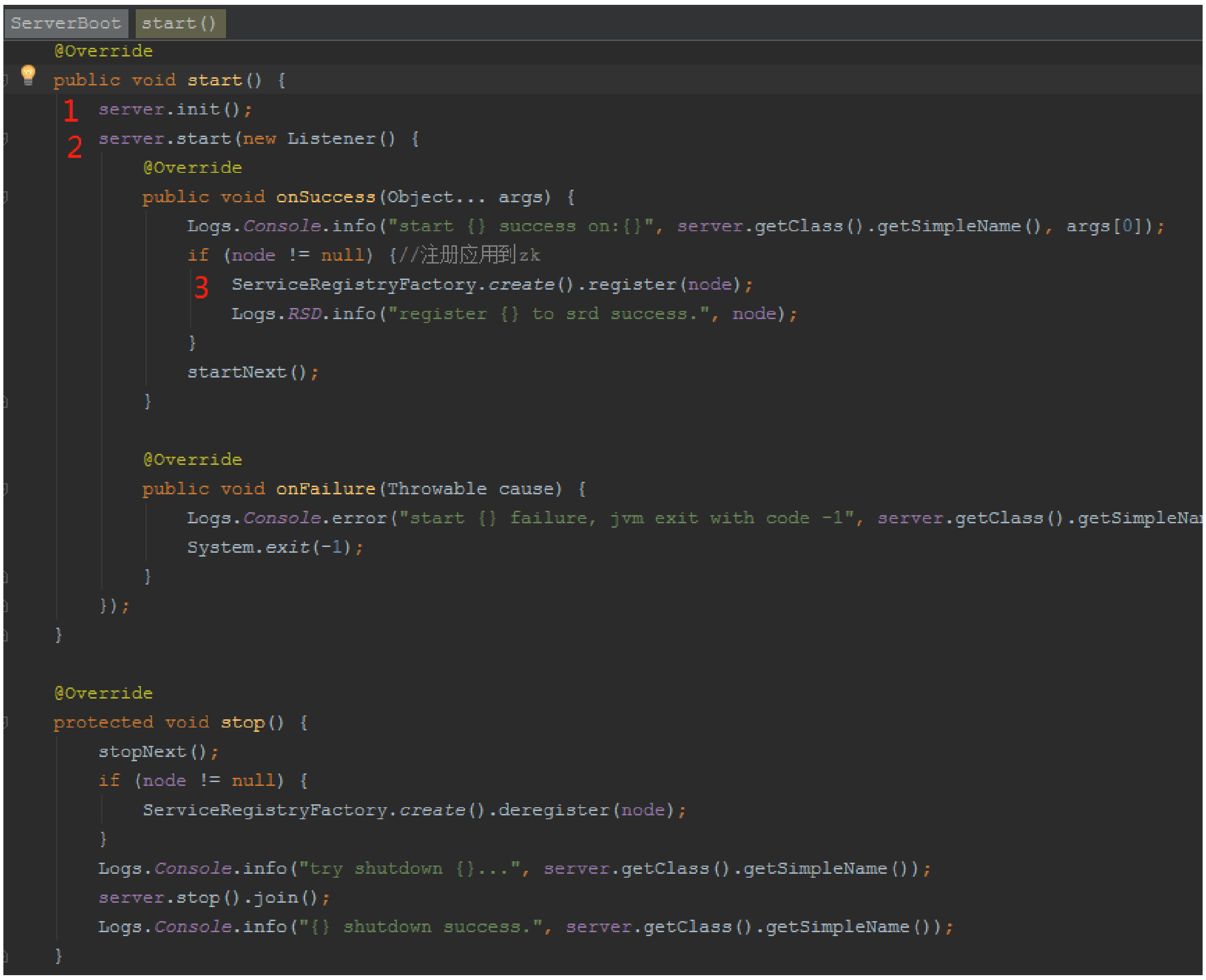
初始化UDP网关服务



```java
chain.boot()
        .setNext(new CacheManagerBoot())//1.初始化缓存模块
        .setNext(new ServiceRegistryBoot())//2.启动服务注册与发现模块
        .setNext(new ServiceDiscoveryBoot())//2.启动服务注册与发现模块
        .setNext(new ServerBoot(mPushServer.getConnectionServer(), mPushServer.getConnServerNode()))//3.启动接入服务
        .setNext(() -> new ServerBoot(mPushServer.getWebsocketServer(), mPushServer.getWebsocketServerNode()), wsEnabled())
        .setNext(() -> new ServerBoot(mPushServer.getUdpGatewayServer(), mPushServer.getGatewayServerNode()), udpGateway()
        .setNext(() -> new ServerBoot(mPushServer.getGatewayServer(), mPushServer.getGatewayServerNode()), tcpGateway())//
        .setNext(new ServerBoot(mPushServer.getAdminServer(), null))//7.启动控制台服务
        .setNext(new RouterCenterBoot(mPushServer))//8.启动路由中心组件
        .setNext(new PushCenterBoot(mPushServer))//9.启动推送中心组件
        .setNext(() -> new HttpProxyBoot(mPushServer), CC.mp.http.proxy_enabled)//10.启动http代理服务, dns解析服务
        .setNext(new MonitorBoot(mPushServer))//11.启动监控服务
        .end();
```

服务启动

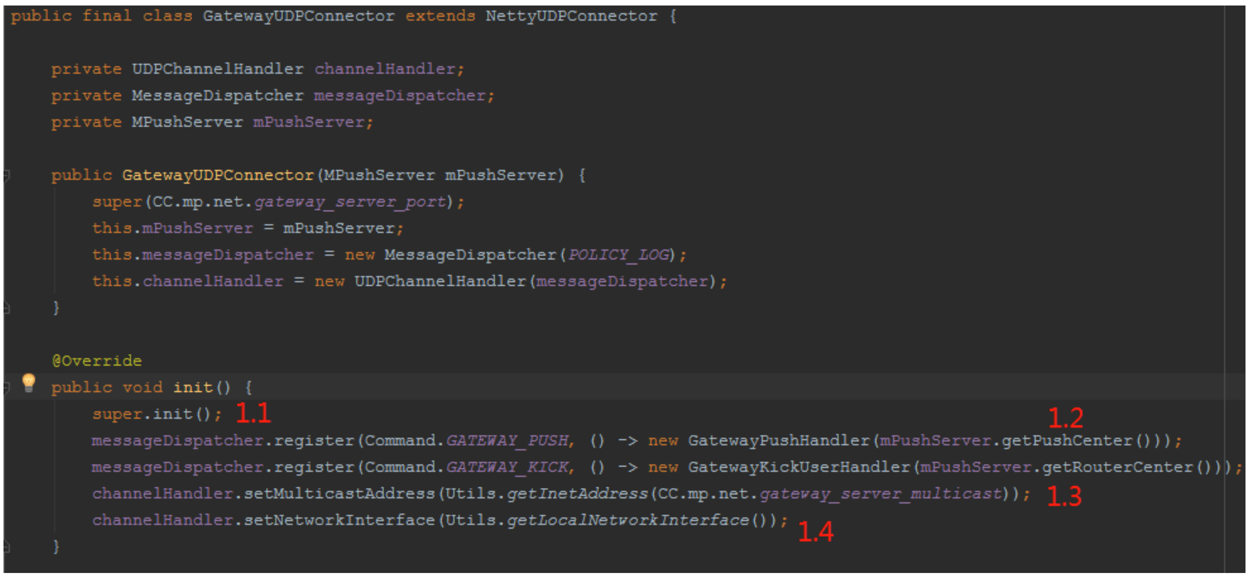


```java
ServerBoot  start()
    @Override
    public void start() {
     1  server.init();
     2  server.start(new Listener() {
            @Override
            public void onSuccess(Object... args) {
                Logs.Console.info("start {} success on:{}", server.getClass().getSimpleName(), args[0]);
                if (node != null) {//注册应用到zk
                 3  ServiceRegistryFactory.create().register(node);
                    Logs.RSD.info("register {} to srd success.", node);
                }
                startNext();
            }

            @Override
            public void onFailure(Throwable cause) {
                Logs.Console.error("start {} failure, jvm exit with code -1", server.getClass().getSimpleNa
                System.exit(-1);
            }
        });
    }

    @Override
    protected void stop() {
        stopNext();
        if (node != null) {
            ServiceRegistryFactory.create().deregister(node);
        }
        Logs.Console.info("try shutdown {}...", server.getClass().getSimpleName());
        server.stop().join();
        Logs.Console.info("{} shutdown success.", server.getClass().getSimpleName());
    }
```

1、调用GatewayUDPConnector#init()

2、调用GatewayUDPConnector的父类NettyTCPServer#start()

3、将GS节点信息注册到Zookeeper



```java
public final class GatewayUDPConnector extends NettyUDPConnector {

    private UDPChannelHandler channelHandler;
    private MessageDispatcher messageDispatcher;
    private MPushServer mPushServer;

    public GatewayUDPConnector(MPushServer mPushServer) {
        super(CC.mp.net.gateway_server_port);
        this.mPushServer = mPushServer;
        this.messageDispatcher = new MessageDispatcher(POLICY_LOG);
        this.channelHandler = new UDPChannelHandler(messageDispatcher);
    }

    @Override
    public void init() {
        super.init();
        messageDispatcher.register(Command.GATEWAY_PUSH, () -> new GatewayPushHandler(mPushServer.getPushCenter()));
        messageDispatcher.register(Command.GATEWAY_KICK, () -> new GatewayKickUserHandler(mPushServer.getRouterCenter()));
        channelHandler.setMulticastAddress(Utils.getInetAddress(CC.mp.net.gateway_server_multicast));
        channelHandler.setNetworkInterface(Utils.getLocalNetworkInterface());
    }
```

1.1 调用NettyUDPConnector#BaseService#init()

　　空方法，没作用；

1.2 注册各种消息的处理类

1.3 设置组播地址

1.4 设置本地网络接口

2 调用NettyUDPConnector#start()方法，创建Bootstrap启动Netty长连接服务；

　　NettyUDPConnector创建netty Bootstrap服务时，会调用其子类GatewayUDPConnector中方法：

　　initOptions(): 设置发送、接收BUF缓冲区大小

　　getChannelHandler(): 设置netty 事件处理类UDPChannelHandler，处理建连、消息、断连、异常事件；