

## 第十四章 微服务项目实战 Shop+

一样的在线教育，不一样的教学品质



# 目录Contents

- ◆ 项目需求与规划
- ◆ Nacos 服务发现整合
- ◆ Gateway 服务网关整合
- ◆ OAuth2 安全认证整合
- ◆ Dubbo RPC 服务调用整合
- ◆ Seata 分布式事务整合
- ◆ Sentinel 系统防护整合
- ◆ Skywalking 调用链跟踪整合
- ◆ Springboot Admin 服务监控整合
- ◆ Nacos 配置中心整合
- ◆ Docker 容器化部署

## 小节导学

### 项目整体情况

#### 架构

项目整体  
的架构  
宏观上了解全局

#### 技术

项目中所  
需的技术  
点

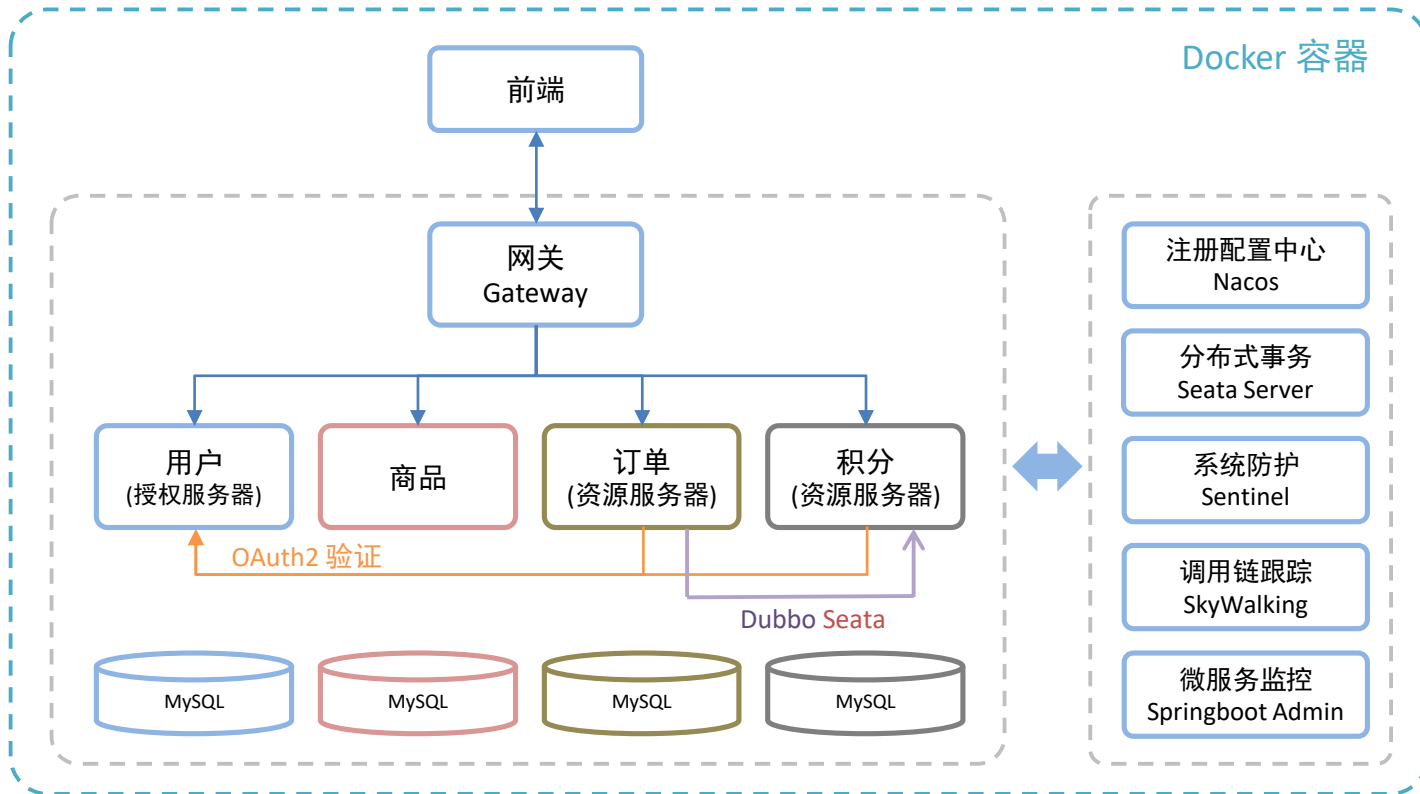
#### 原型

前端界面  
的原型  
了解具体  
功能点

#### 实现

服务划分  
服务中处理  
逻辑梳理

# 1. 项目架构



### 技术列表

- **Nacos** - 服务发现、分布式配置
- **Sentinel** - 系统防护, 限流降级
- **Gateway** - 服务网关
- **Dubbo** - 高性能 RPC
- **Seata** - 分布式事务
- **SkyWalking** - 调用链追踪
- **Spring Security** - Spring 安全框架
- **Oauth2** - 服务安全认证框架
- **SpringBoot Admin** - 微服务监控
- **Docker** - 服务容器化工具
- **Mybatis** - 数据库访问框架
- **MySQL** - 数据库

### 3. 原型界面

## 商品列表页面



## 3. 原型界面

### 商品详情页面

Shop+ 电商平台

登录

商品详情



#### 水果便携电脑

价格: ¥6698

CPU型号: 其他

运行内存: 8GB

机身存储: 256GB

存储卡: 支持MicroSD(TF)

分辨率: 1920 x 1080

屏幕比例: 16:9

立即购买

### 3. 原型界面

#### 商品详情页面





## 3. 原型界面

### 我的订单页面

Shop+ 电商平台

testuser

个人中心

我的订单

我的积分

2020-04-02 15:35:30

订单号：116480227524



水果便携电脑

数量：1

吕 小郭

总金额 ¥ 6698

2020-04-02 15:35:30

订单号：116480227524



水果便携电脑

数量：1

吕 小郭

总金额 ¥ 6698

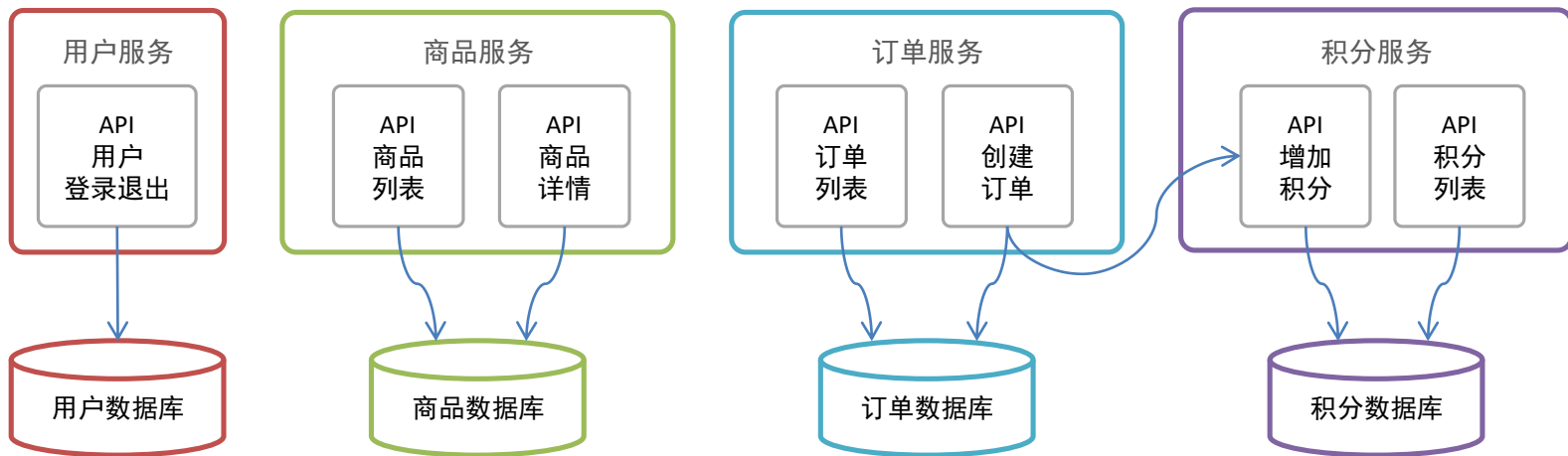
## 3. 原型界面

### 我的积分页面

Shop+ 电商平台			testuser
个人中心	我的订单	我的积分	
总积分: 814			
时间	收入	说明	
2020-04-02 11:23:41	2	购买便携电脑奖励	
2020-03-31 08:45:33	10	购买手机奖励	
2020-03-29 16:57:10	100	购买笔记本奖励	

## 4. 服务规划

### 服务API



登录 **/oauth/token**  
退出 **/logout**

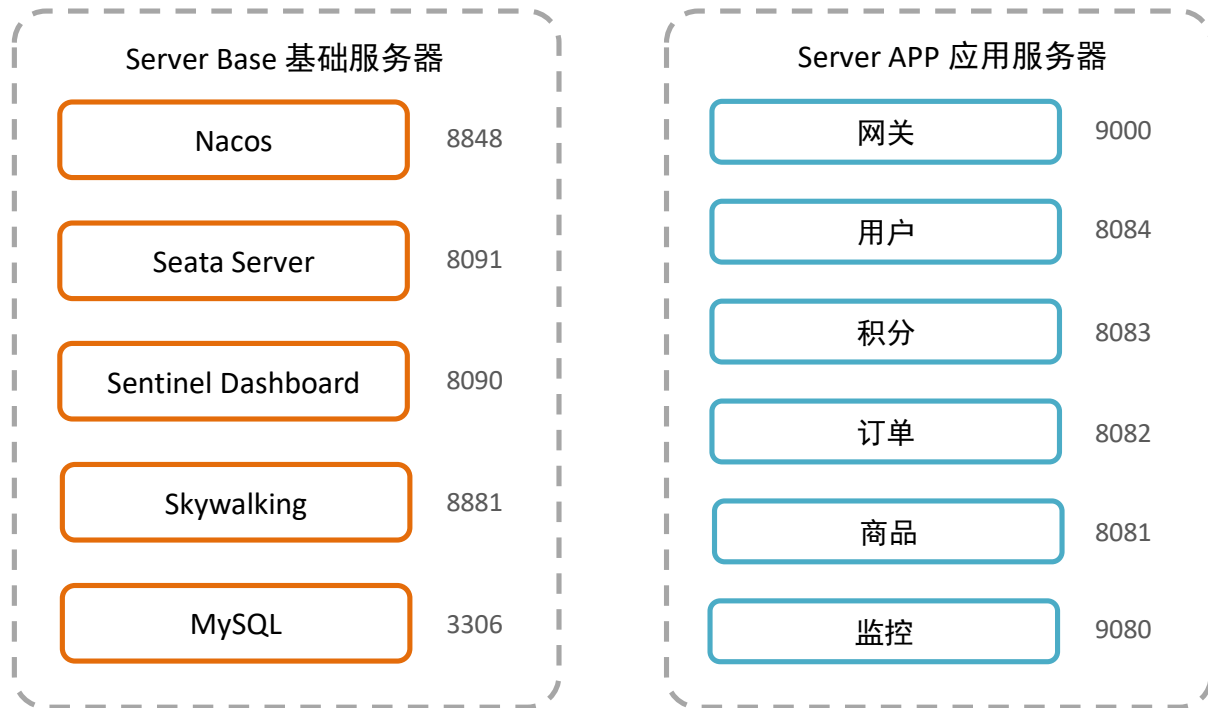
列表 **/products**  
详情 **/product?id=1**

列表 **/orders**  
创建 **/order**

列表 **/score**  
创建 **/addscore**

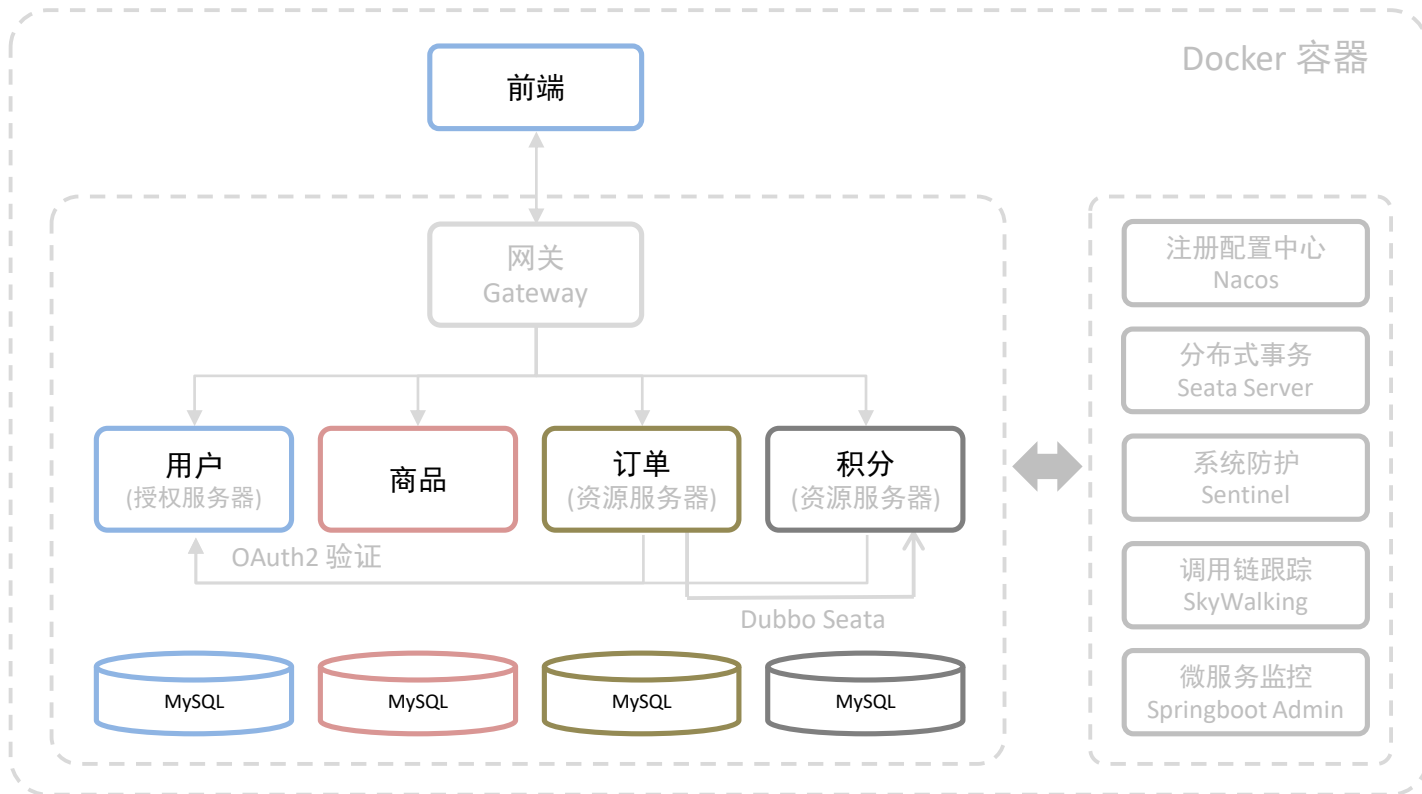
## 4. 服务规划

### 服务部署



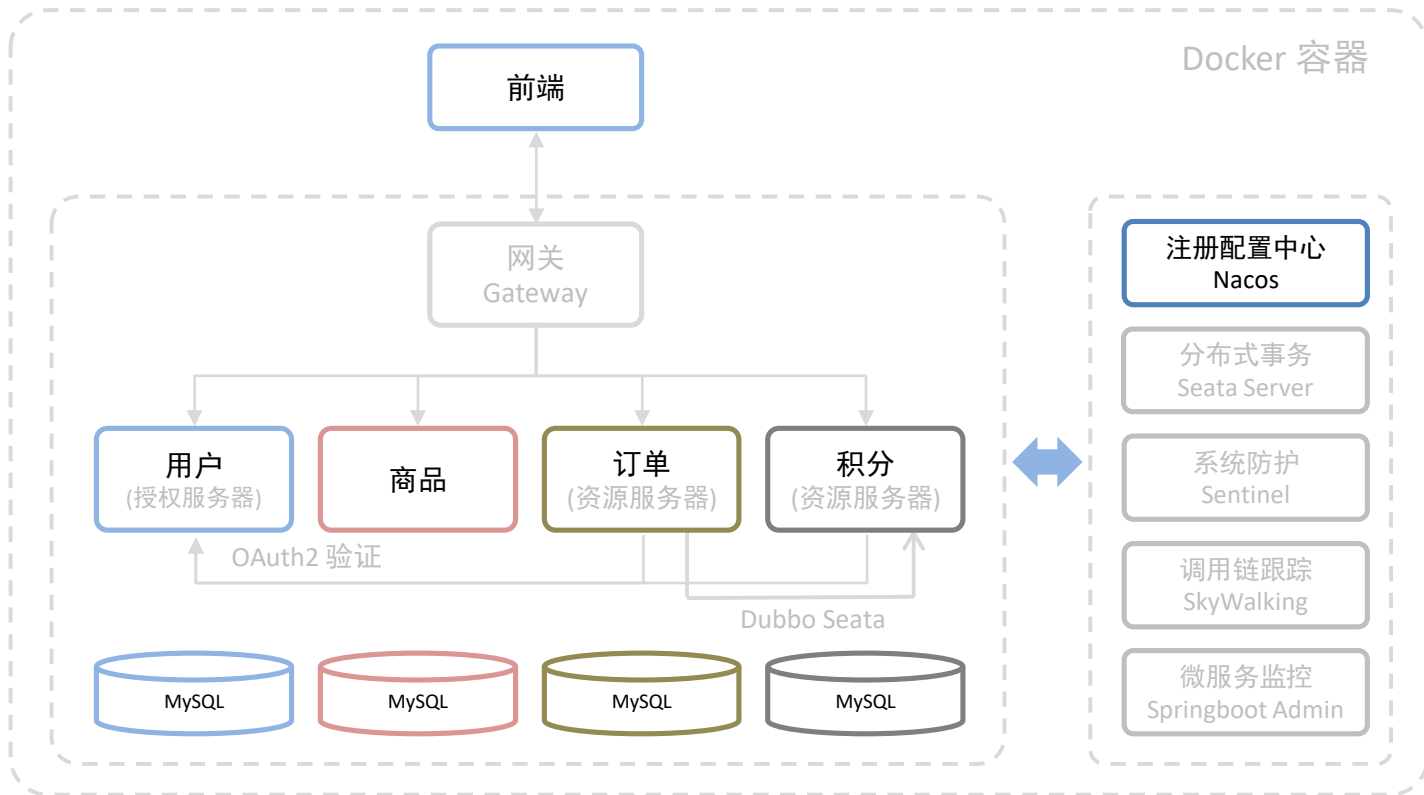
## 4. 服务规划

### 开发步骤 – 基础功能



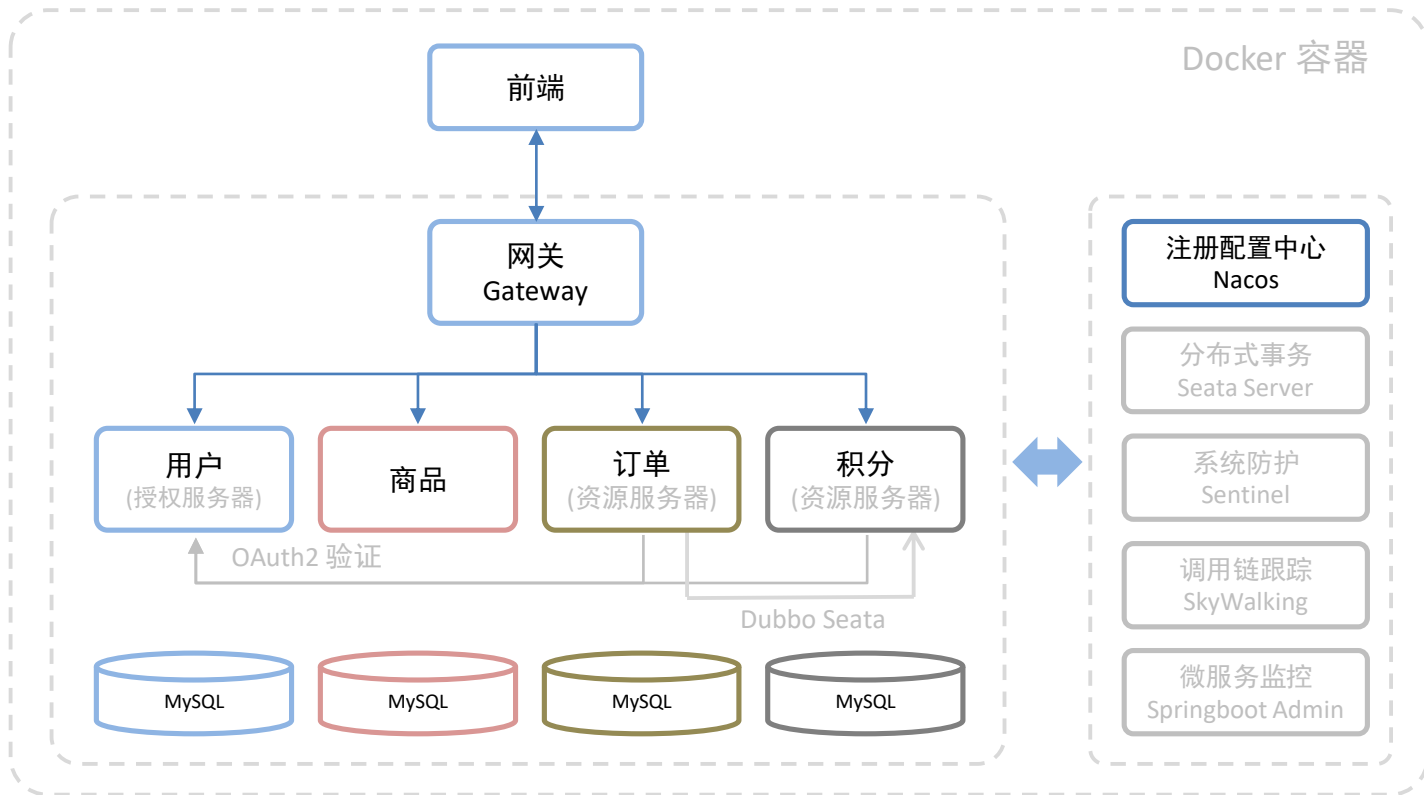
## 4. 服务规划

### 开发步骤 - Nacos



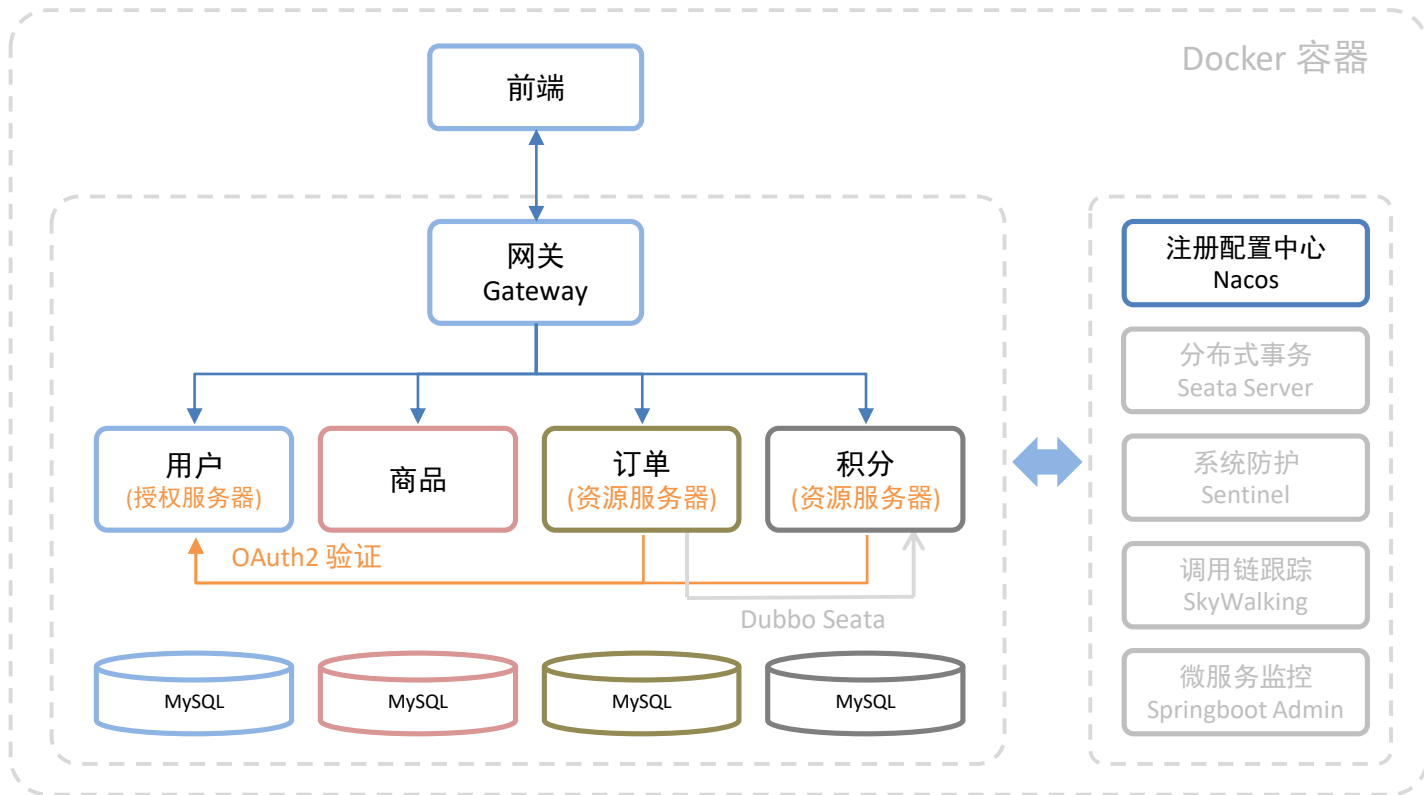
## 4. 服务规划

### 开发步骤 - Gateway



## 4. 服务规划

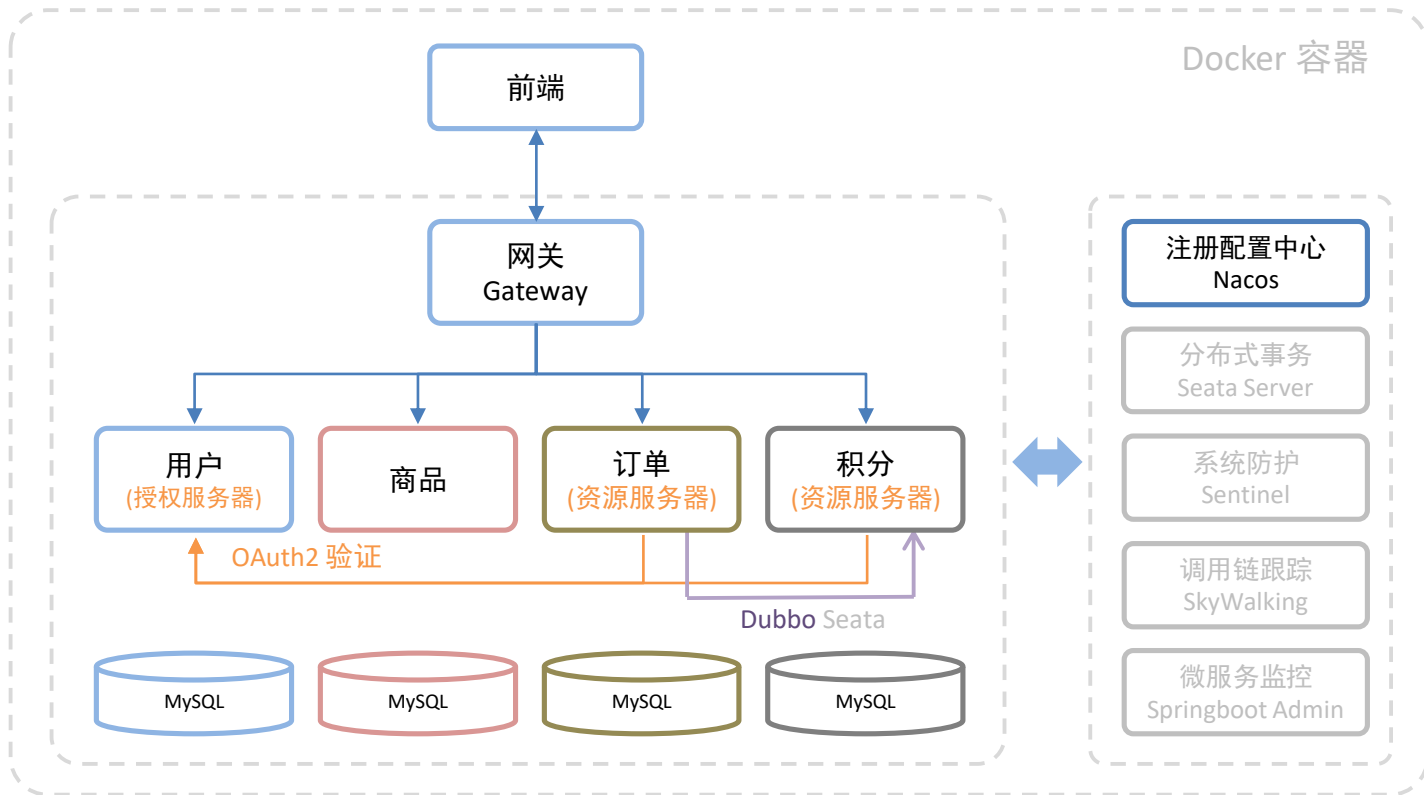
### 开发步骤 – OAuth2





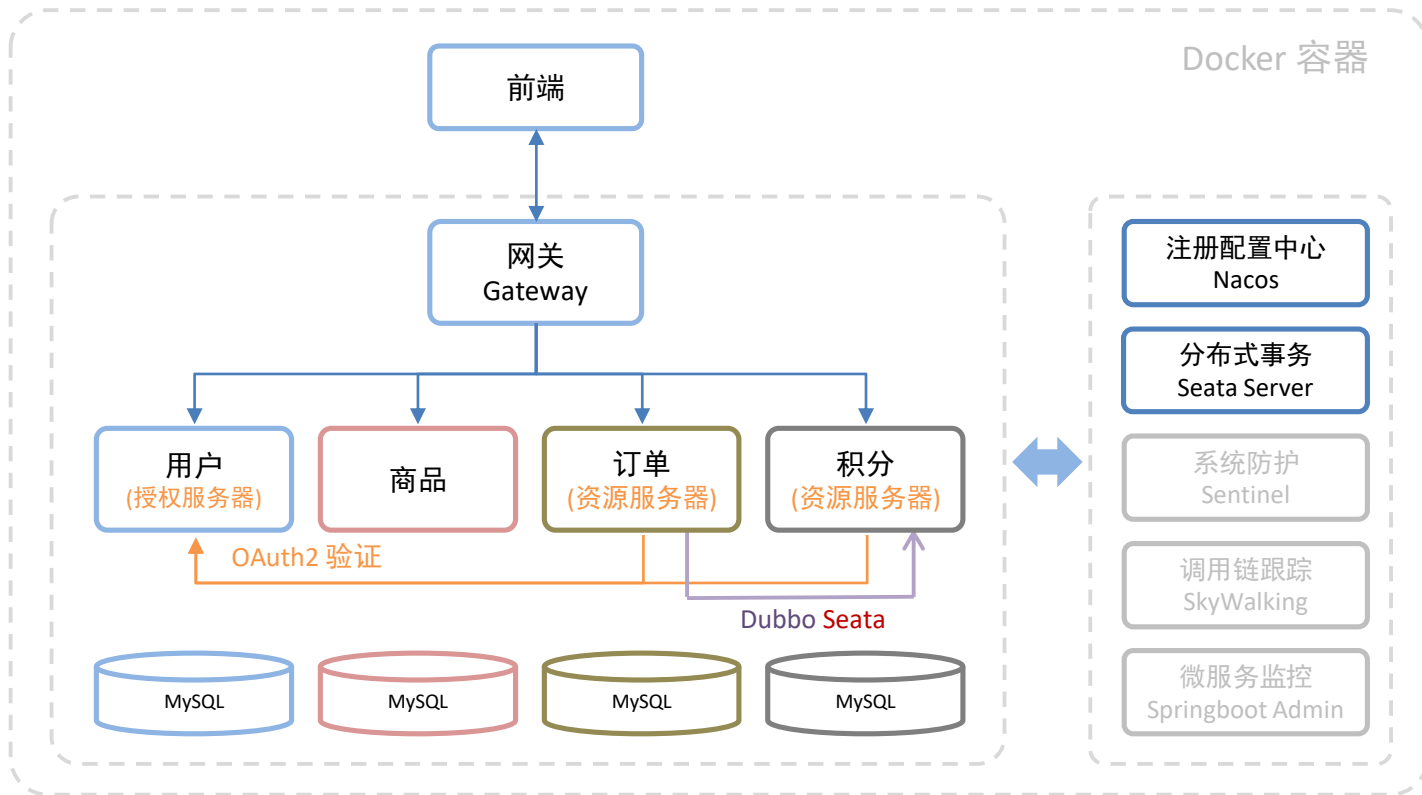
## 4. 服务规划

### 开发步骤 - Dubbo



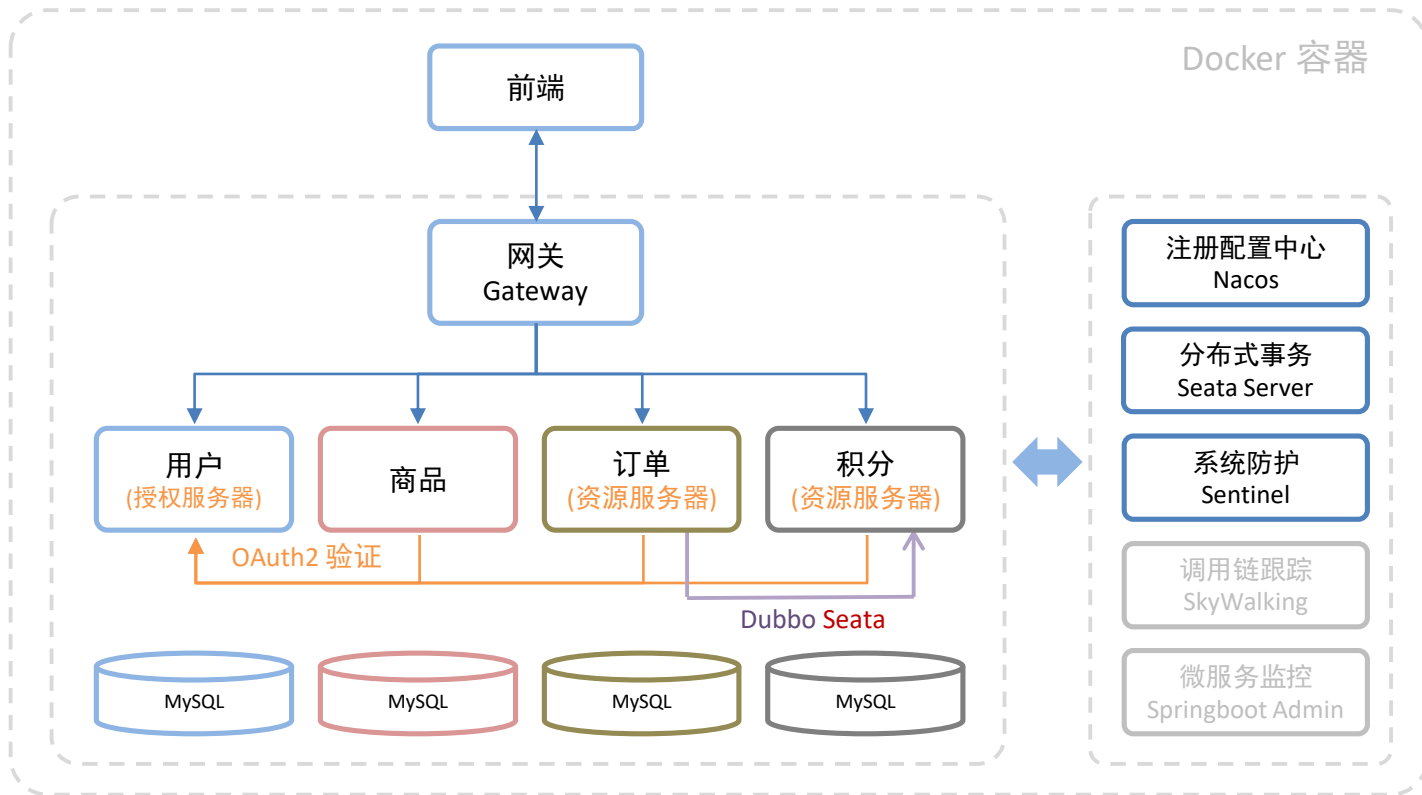
## 4. 服务规划

### 开发步骤 - Seata



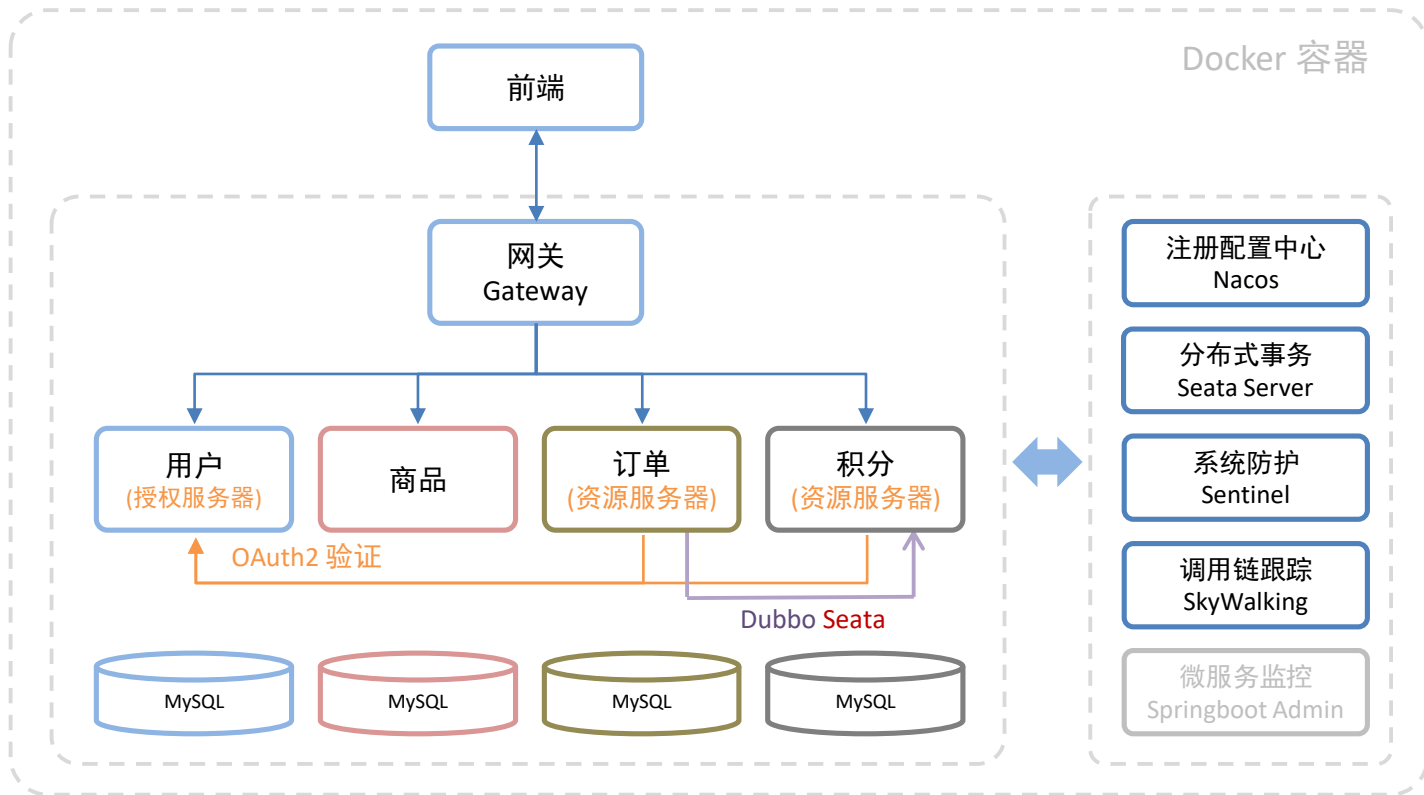
## 4. 服务规划

### 开发步骤 - Sentinel



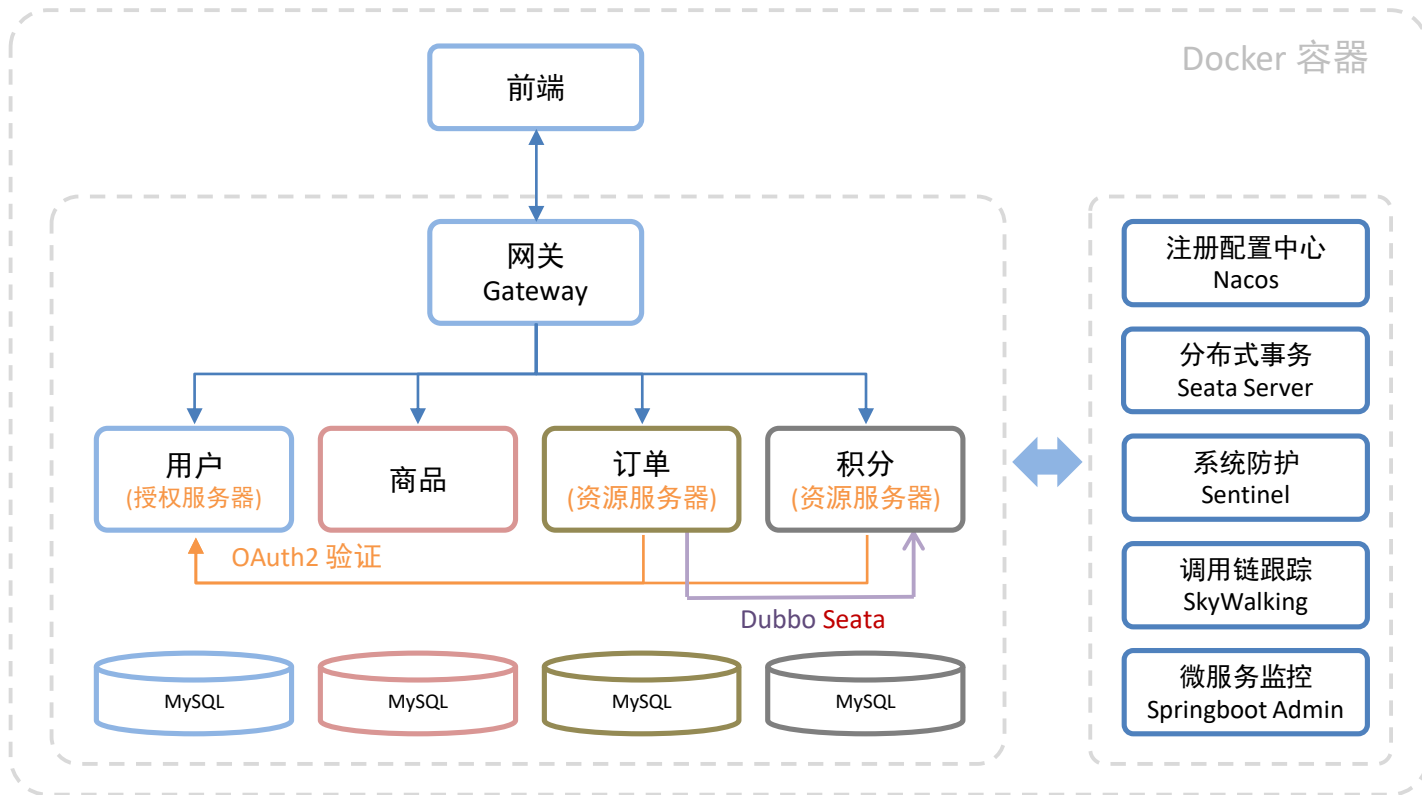
## 4. 服务规划

### 开发步骤 - Skywalking



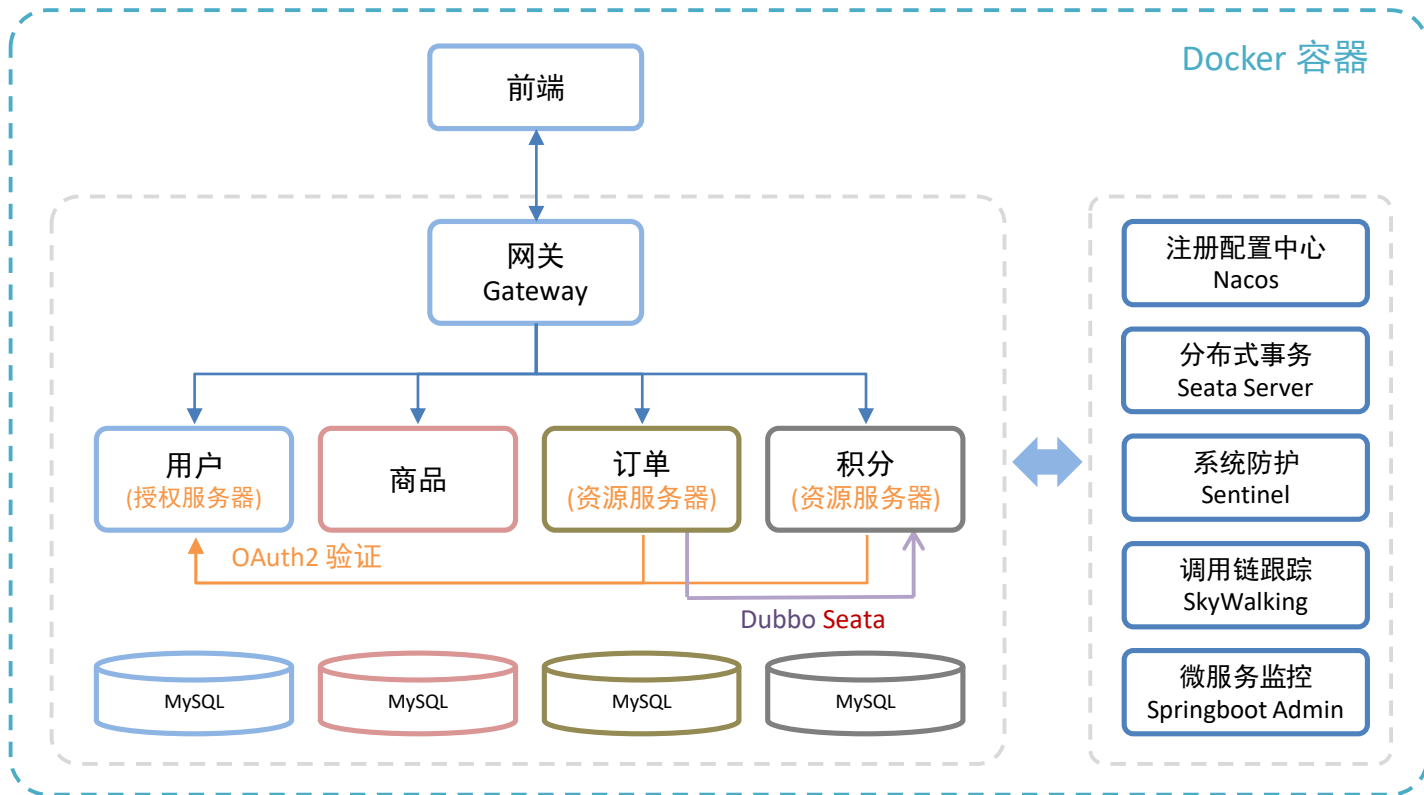
## 4. 服务规划

### 开发步骤 - SpringbootAdmin



## 4. 服务规划

### 开发步骤 - Docker





## 总结

### 重难点

1. 项目整体架构
2. 项目流程
3. 服务规划

### 下节

整合 Nacos 服务注册中心



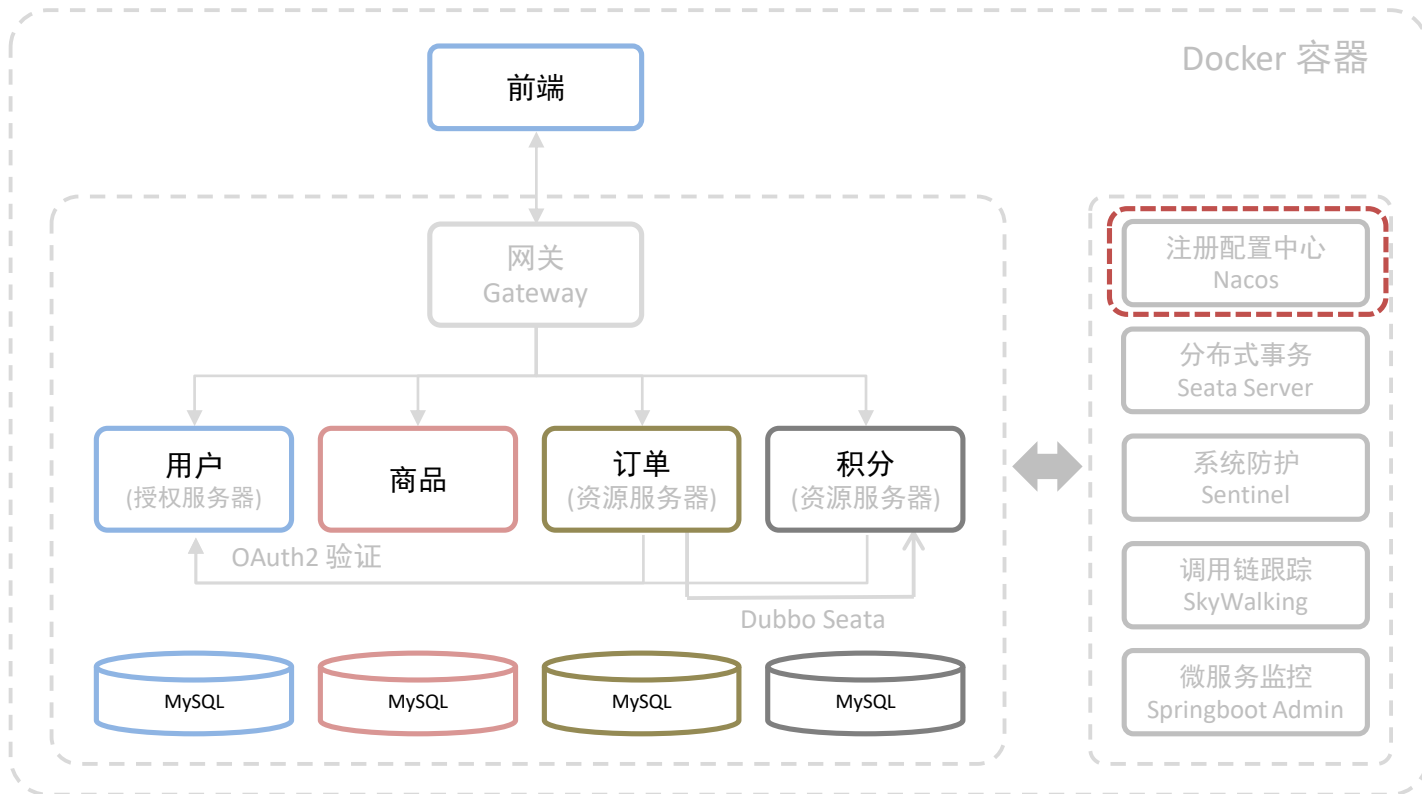
# 目录 Contents

- ◆ 项目需求与规划
- ◆ Nacos 服务发现整合
- ◆ Gateway 服务网关整合
- ◆ OAuth2 安全认证整合
- ◆ Dubbo RPC 服务调用整合
- ◆ Seata 分布式事务整合
- ◆ Sentinel 系统防护整合
- ◆ Skywalking 调用链跟踪整合
- ◆ Springboot Admin 服务监控整合
- ◆ Nacos 配置中心整合
- ◆ Docker 容器化部署



## 4. 服务规划

### 开发步骤 - 基础功能



## 小节导学

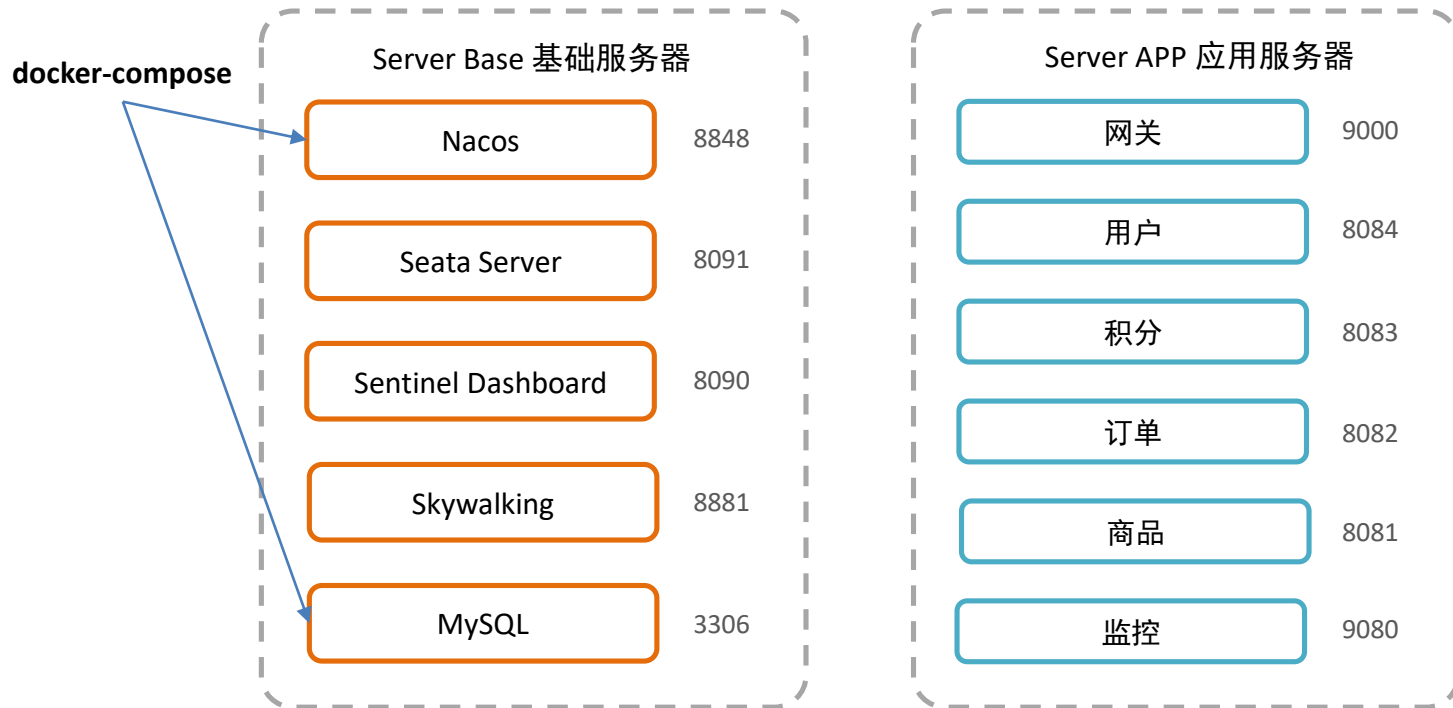
本节我们把各个服务都整合到 Nacos 服务注册中心，以便各个服务都可以发现对方。



## 整合步骤



## Docker 容器部署



## Docker 容器部署

### `docker-compose.yml`

```
version: "2"
services:
  nacos:
    image: nacos/nacos-server:latest
    container_name: nacos
    environment:
      - PREFER_HOST_MODE=hostname
      - MODE=standalone
    ports:
      - "8848:8848"
  mysql:
    image: mysql:5.7
    container_name: mysql
    environment:
      MYSQL_ROOT_PASSWORD: 123456
    ports:
      - 3306:3306
    volumes:
      - ./mysqldata:/var/lib/mysql
```

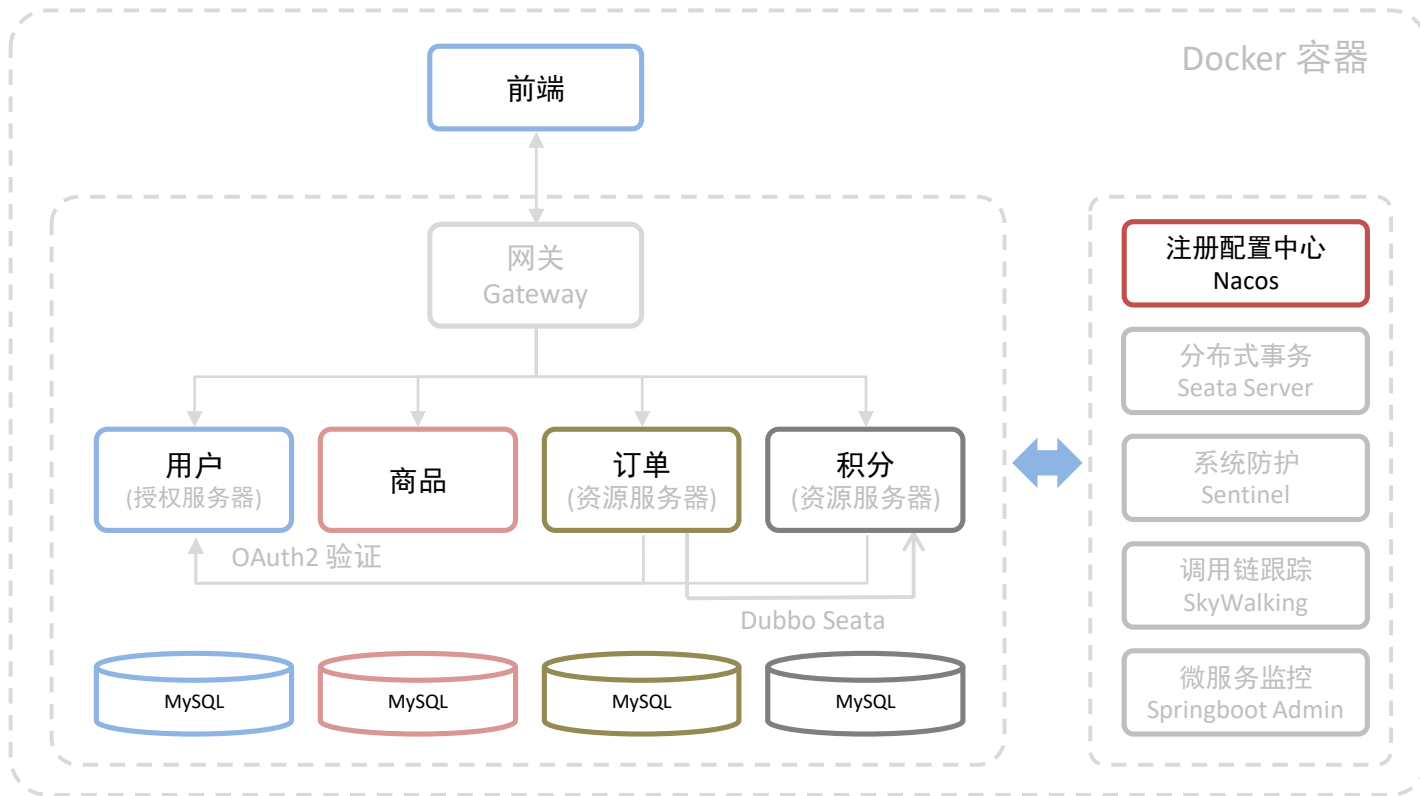


## 总结

### 重难点

1. 服务整合 Nacos 注册中心
2. Nacos Docker 容器运行

# Nacos 服务发现整合 - 总结





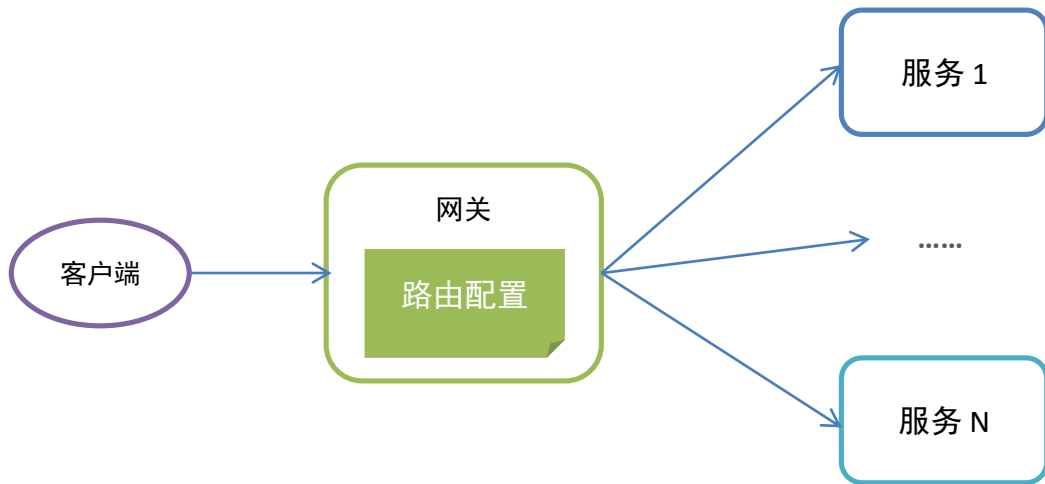
# 目录 Contents

- ◆ 项目需求与规划
- ◆ Nacos 服务发现整合
- ◆ Gateway 服务网关整合
- ◆ OAuth2 安全认证整合
- ◆ Dubbo RPC 服务调用整合
- ◆ Seata 分布式事务整合
- ◆ Sentinel 系统防护整合
- ◆ Skywalking 调用链跟踪整合
- ◆ Springboot Admin 服务监控整合
- ◆ Nacos 配置中心整合
- ◆ Docker 容器化部署



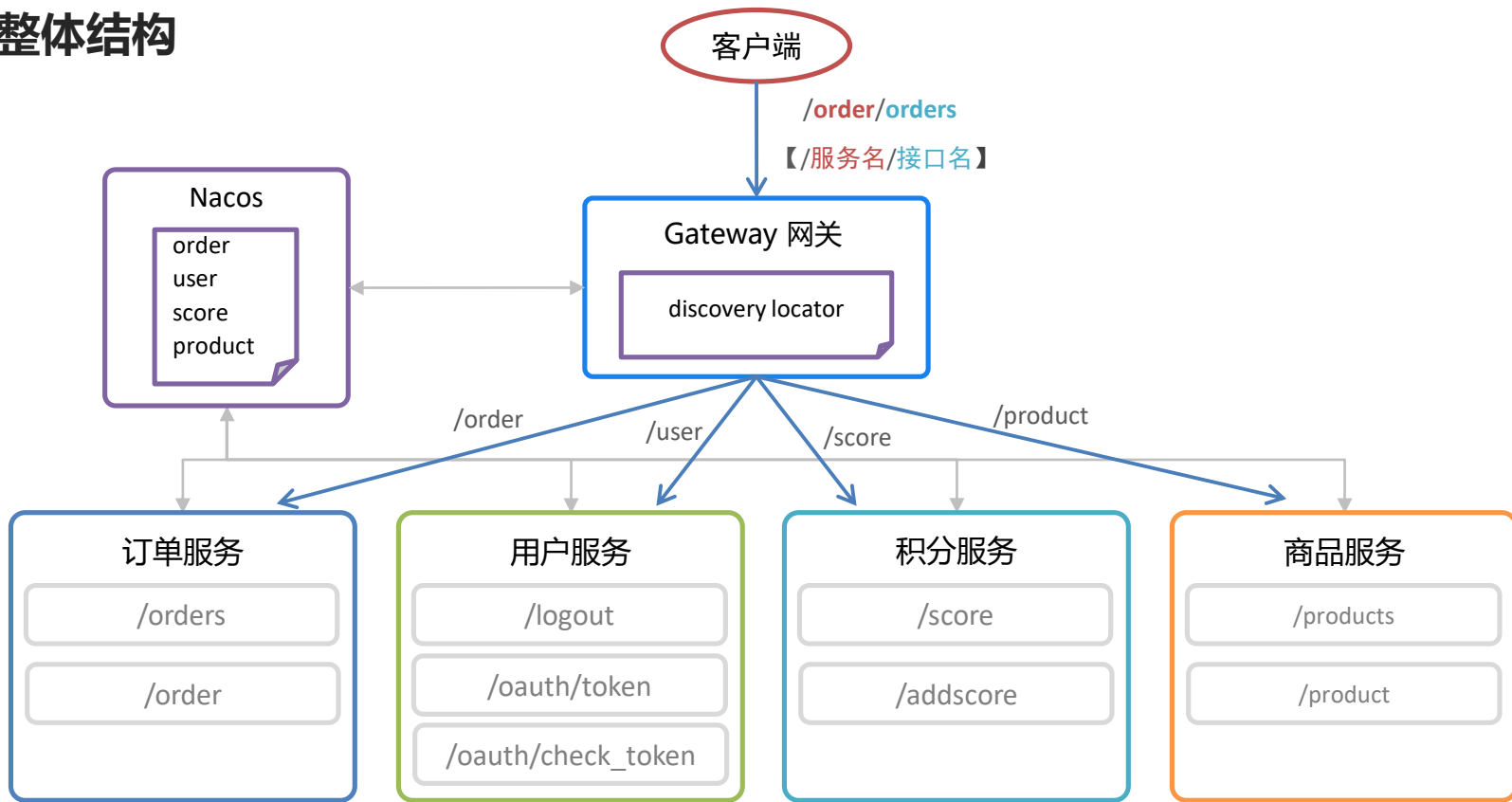
## 小节导学

本节我们整合网关 Gateway。



# Gateway 服务网关整合

## 整体结构



## 整合流程



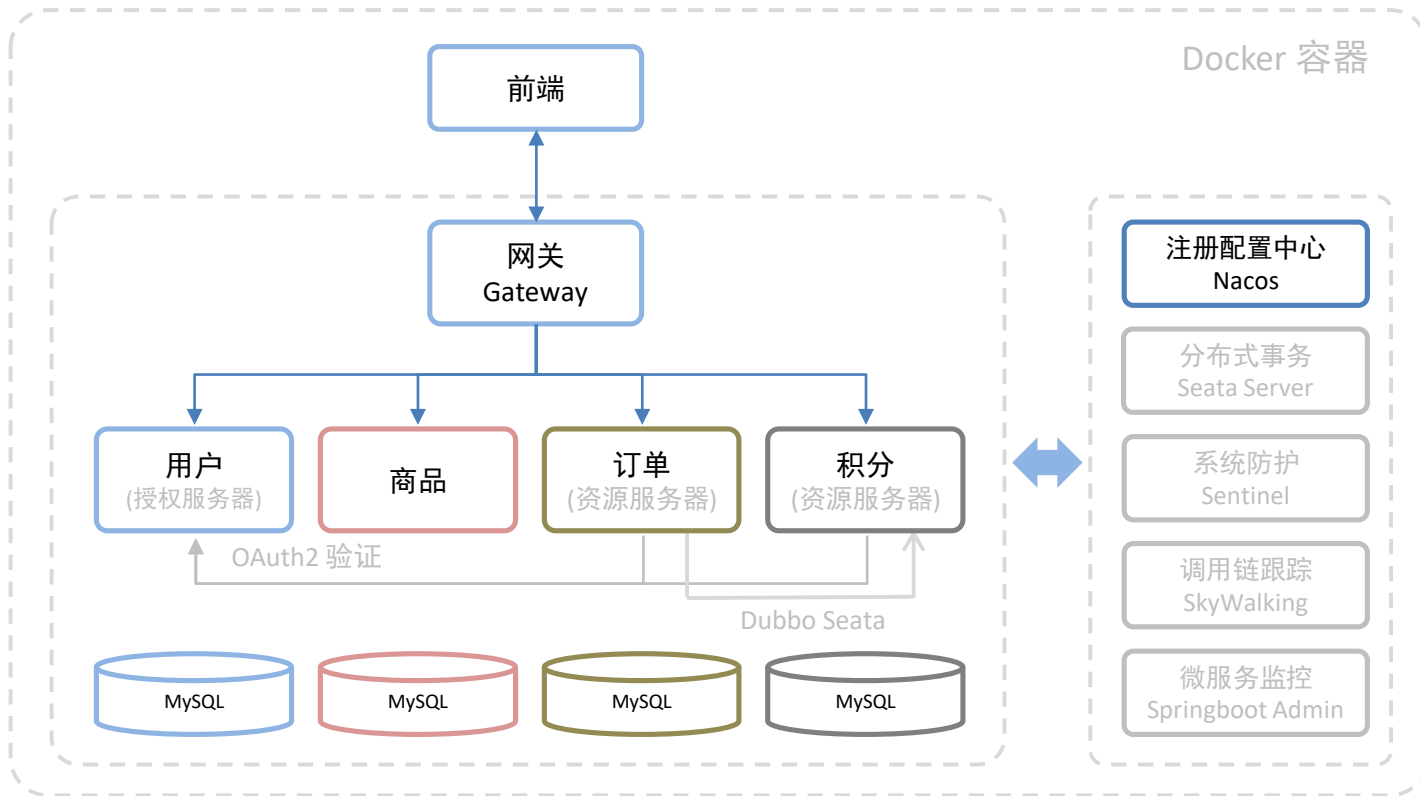


## 总结

### 重难点

1. 服务整合 Gateway 后的流程

# Gateway 服务网关整合 - 总结



# 目录 Contents

- ◆ 项目需求与规划
- ◆ Nacos 服务发现整合
- ◆ Gateway 服务网关整合
- ◆ OAuth2 安全认证整合
- ◆ Dubbo RPC 服务调用整合
- ◆ Seata 分布式事务整合
- ◆ Sentinel 系统防护整合
- ◆ Skywalking 调用链跟踪整合
- ◆ Springboot Admin 服务监控整合
- ◆ Nacos 配置中心整合
- ◆ Docker 容器化部署

## 小节导学

本节整合 OAuth2 安全认证，实现3个目标：

### 登录

前端输入用户名密码  
后可以获得 token

### 请求资源

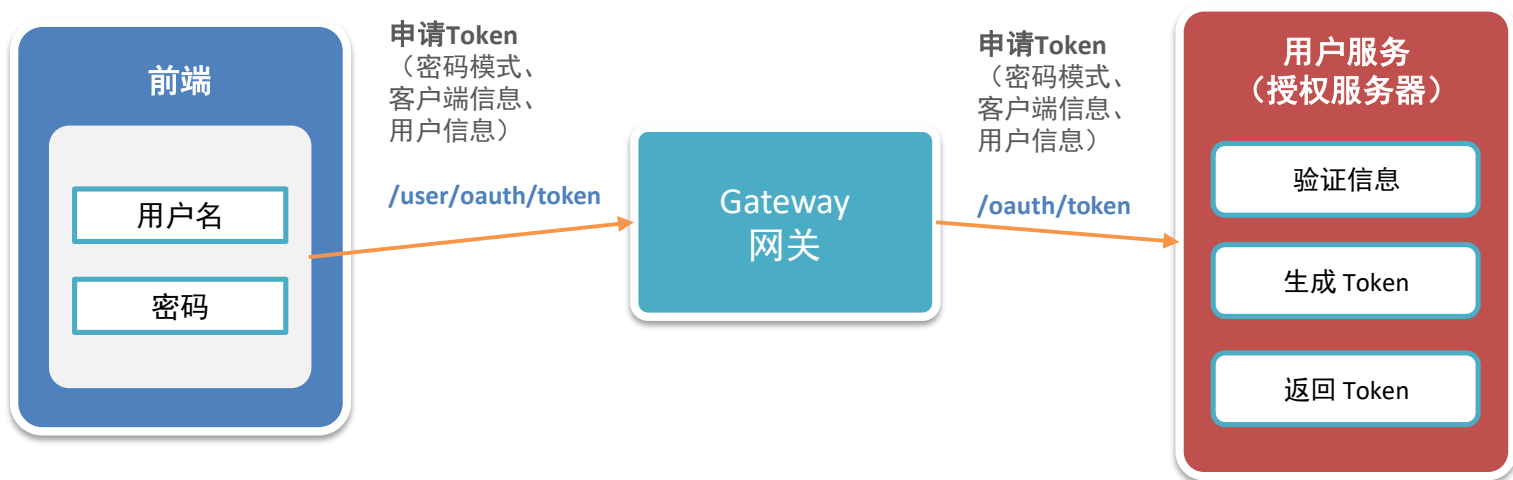
前端页面使用 token 请  
求受保护的资源后可以  
得到正常响应

### 退出

前端可以申请销毁  
token 实现退出

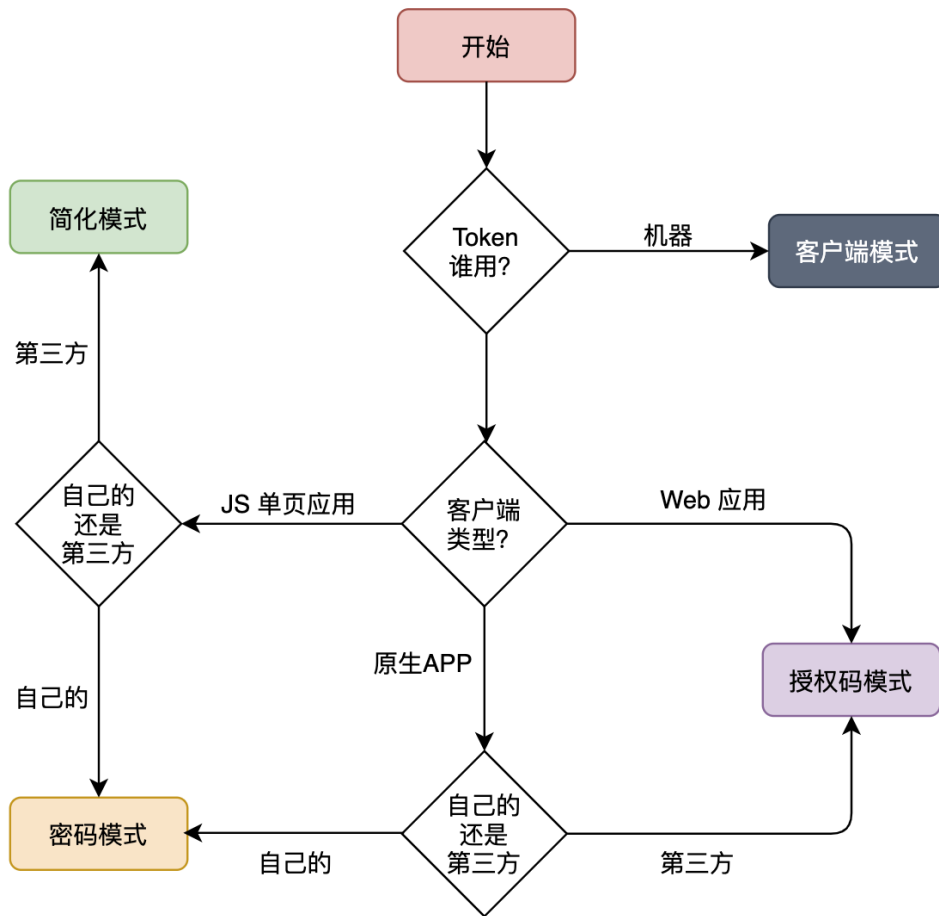
# 1. 登录

## 整体流程





# 回顾 - 模式选择策略





# 1. 登录

## 实践流程

### 前端

提交登录请求

保存 Token

### 服务接口

根据 Token 获取用户  
信息

### 数据库

创建OAuth2、用户表

初始化数据

### 用户服务

添加依赖  
(security、oauth2)

数据库操作  
根据用户名查询用户

实现用户详情服务接口

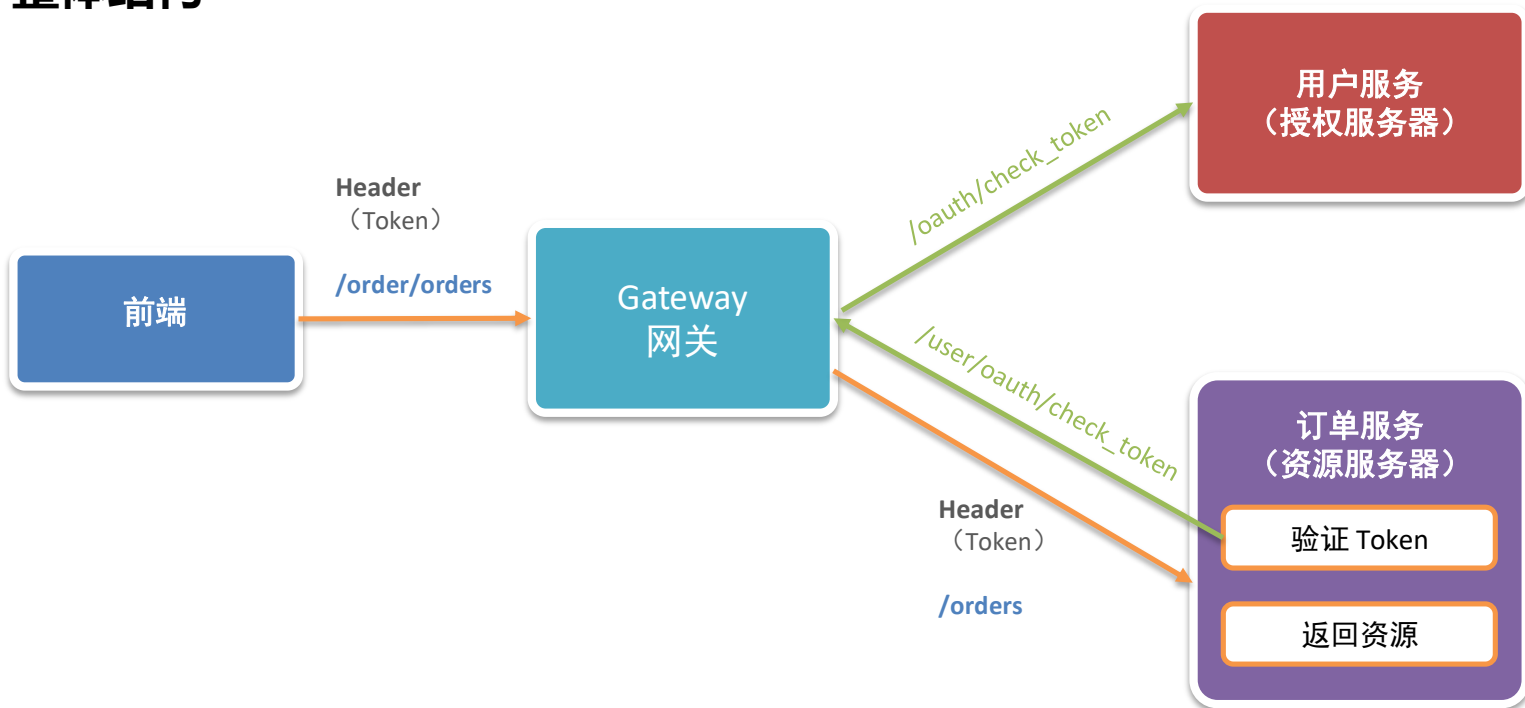
安全相关配置

授权服务器配置

开启授权服务器

## 2. 请求资源

### 整体结构



## ■ 2. 请求资源

### 实践流程

#### 前端

请求 Header 中添加 Token

#### 订单服务

加依赖  
(security、oauth2)

加注解  
(开启资源服务器)

改配置  
(资源服务器信息)

#### 积分服务

加依赖  
(security、oauth2)

加注解  
(开启资源服务器)

改配置  
(资源服务器信息)

### 3. 退出

#### 整体结构



## 3. 退出

### 实践流程

#### 前端

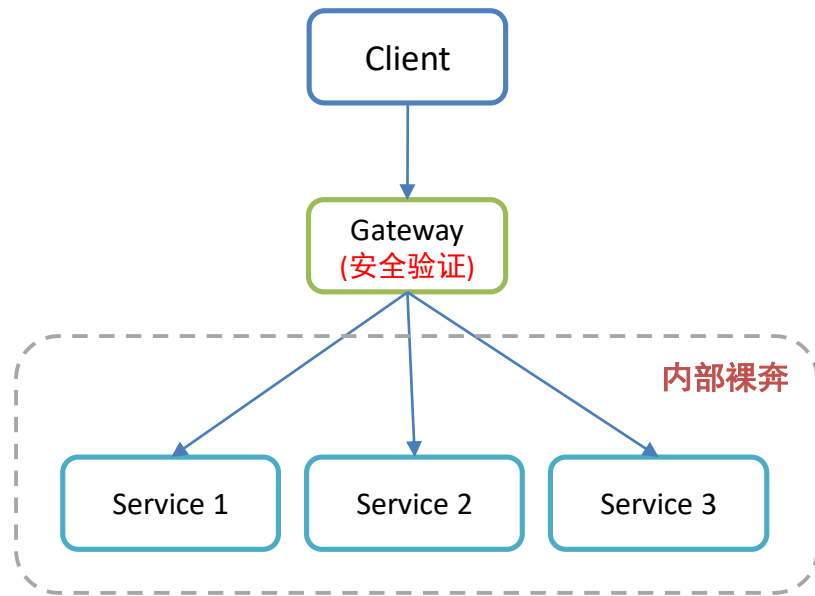
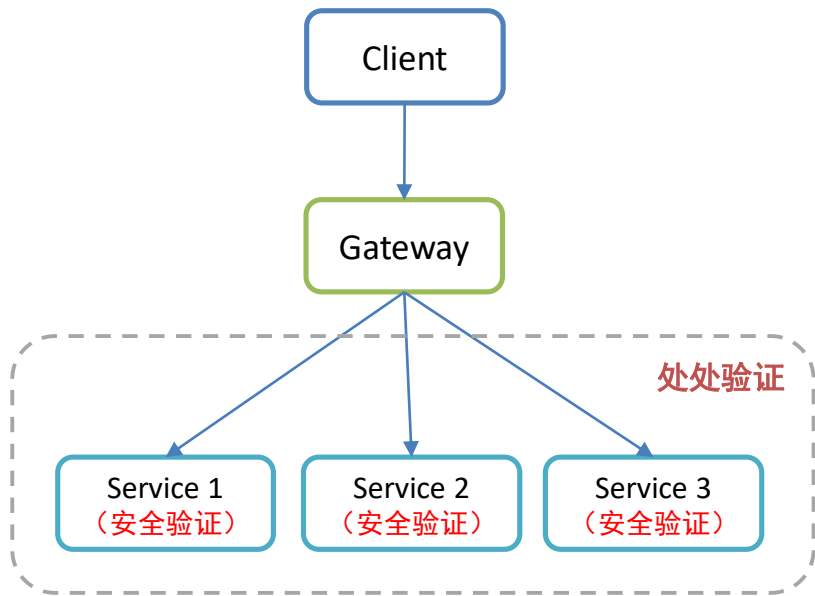
退出 Header 中添加 Token

#### 用户服务

创建退出接口

加注解  
(开启资源服务器)

资源服务器配置





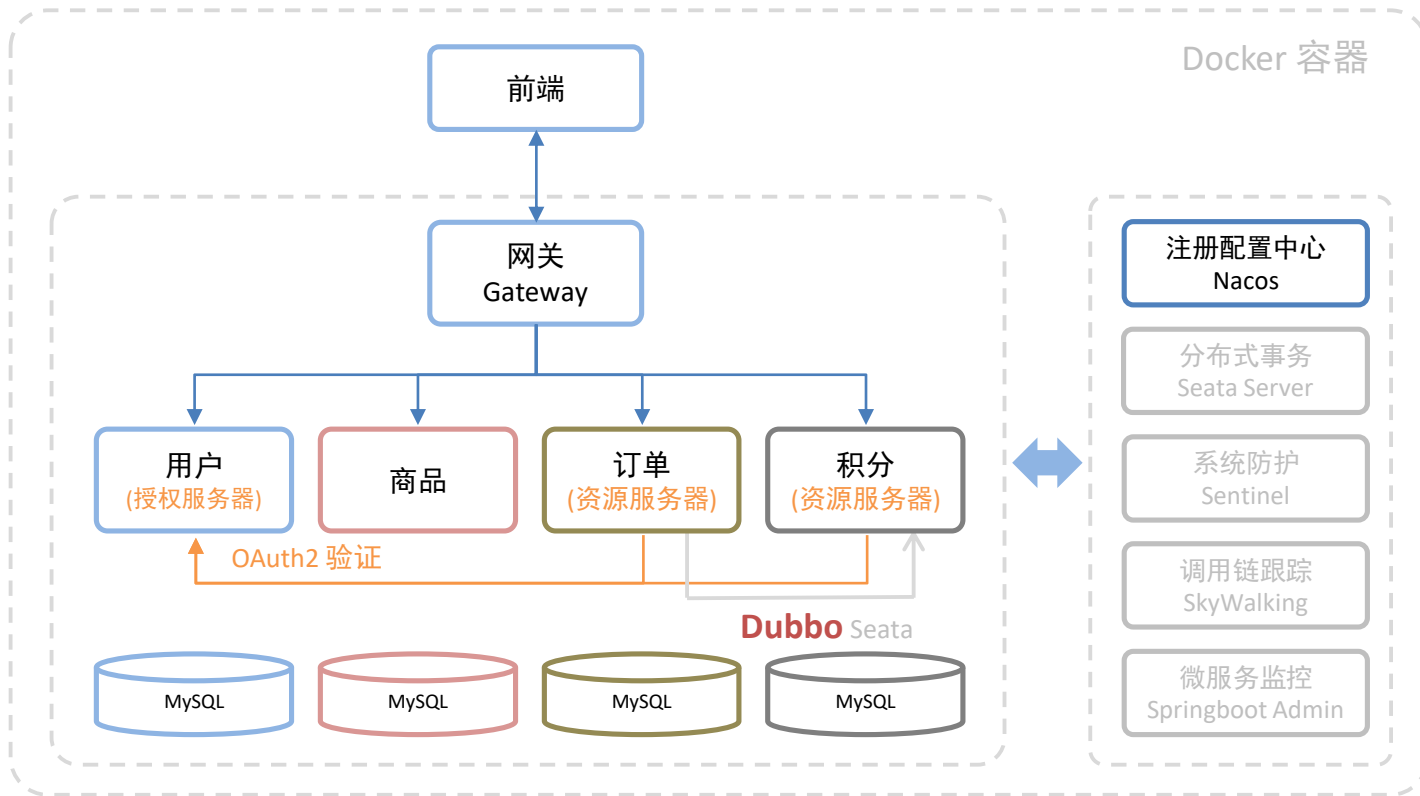
## 总结

### 重难点

1. 前端与后端的安全验证流程
2. 安全认证方案拓展



# OAuth2 安全认证整合 - 总结

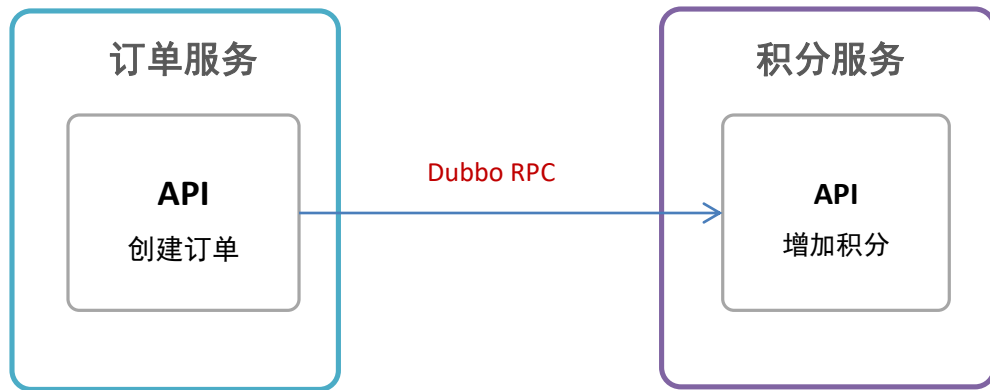


# 目录 Contents

- ◆ 项目需求与规划
- ◆ Nacos 服务发现整合
- ◆ Gateway 服务网关整合
- ◆ OAuth2 安全认证整合
- ◆ Dubbo RPC 服务调用整合
- ◆ Seata 分布式事务整合
- ◆ Sentinel 系统防护整合
- ◆ Skywalking 调用链跟踪整合
- ◆ Springboot Admin 服务监控整合
- ◆ Nacos 配置中心整合
- ◆ Docker 容器化部署

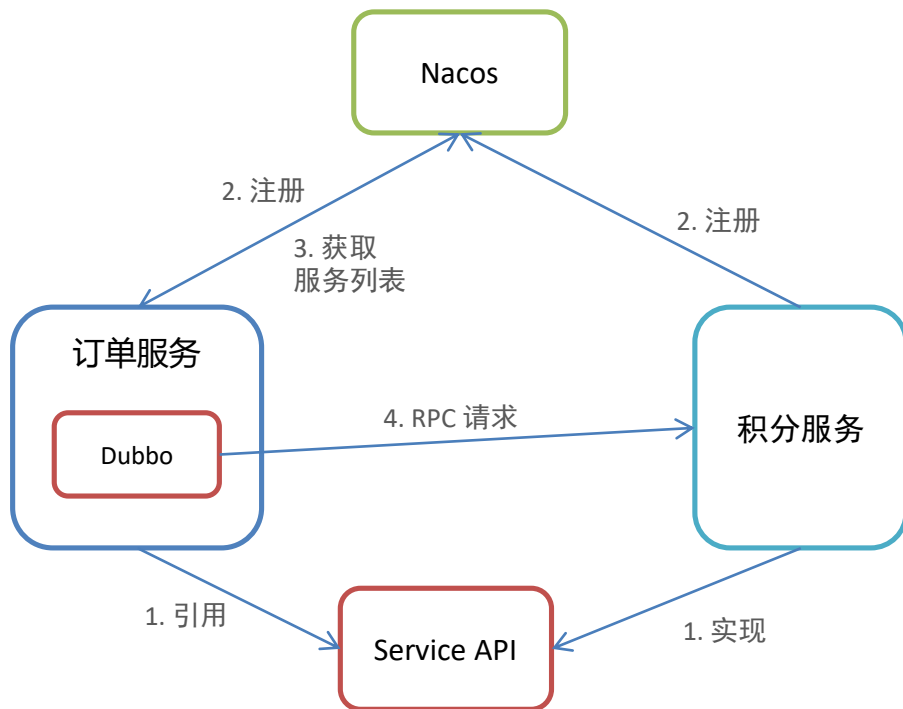
## 小节导学

订单服务创建订单时，需要增加积分，所以，订单服务需要调用积分服务，我们采用 Dubbo RPC 方式调用。



# Dubbo RPC 服务调用整合

## 整体结构



## 实践流程

### 积分服务接口

创建积分服务接口  
(ScoreService)

### 积分服务 (服务提供者)

加依赖  
(dubbo-api、dubbo)

改配置  
(添加 dubbo 配置)

实现 ScoreService 接口

### 订单服务 (服务消费者)

加依赖  
(dubbo-api、dubbo)

改配置  
(添加 dubbo 配置)

引用 ScoreService

调用 ScoreService

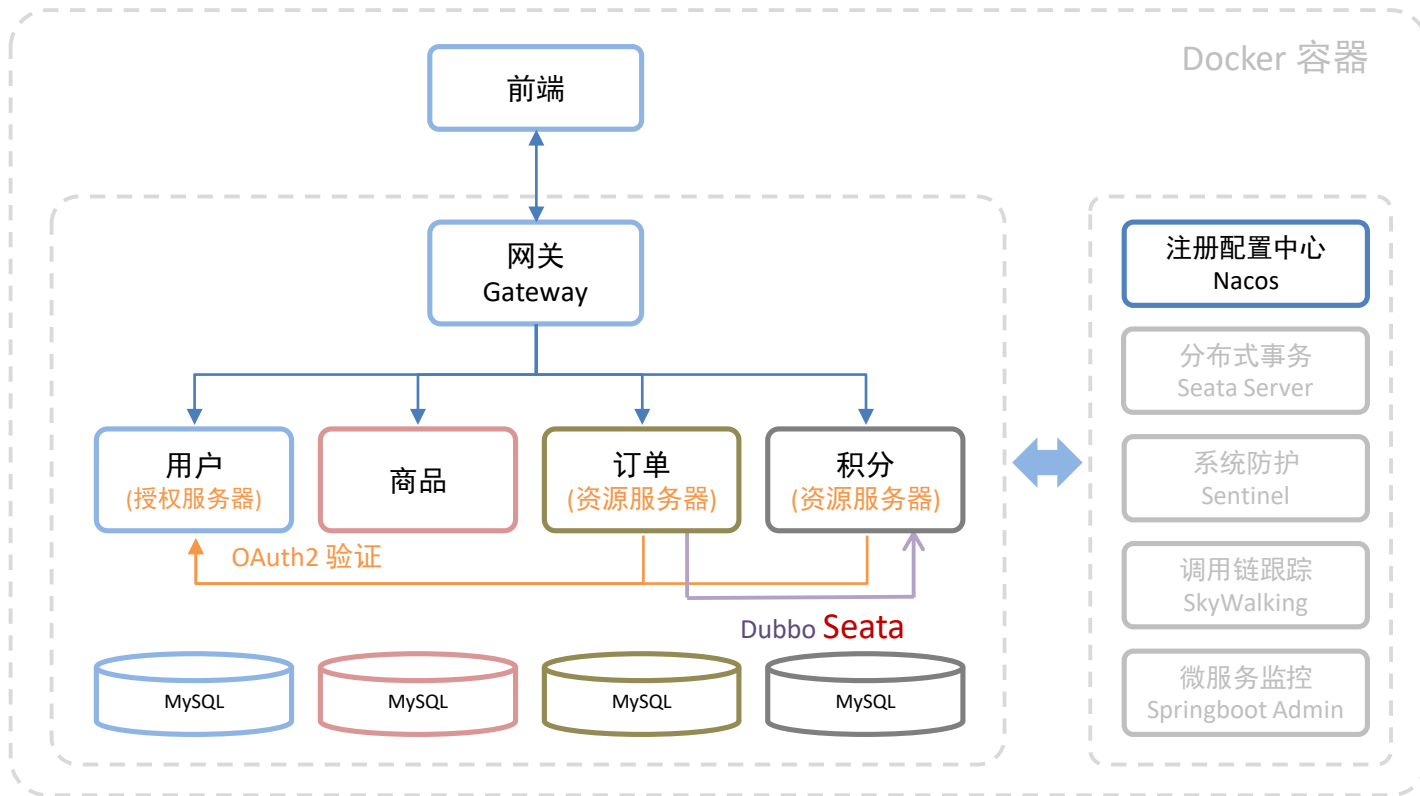


## 总结

### 重难点

1. 服务整合 Dubbo 的流程

# Dubbo RPC 服务调用整合 - 总结

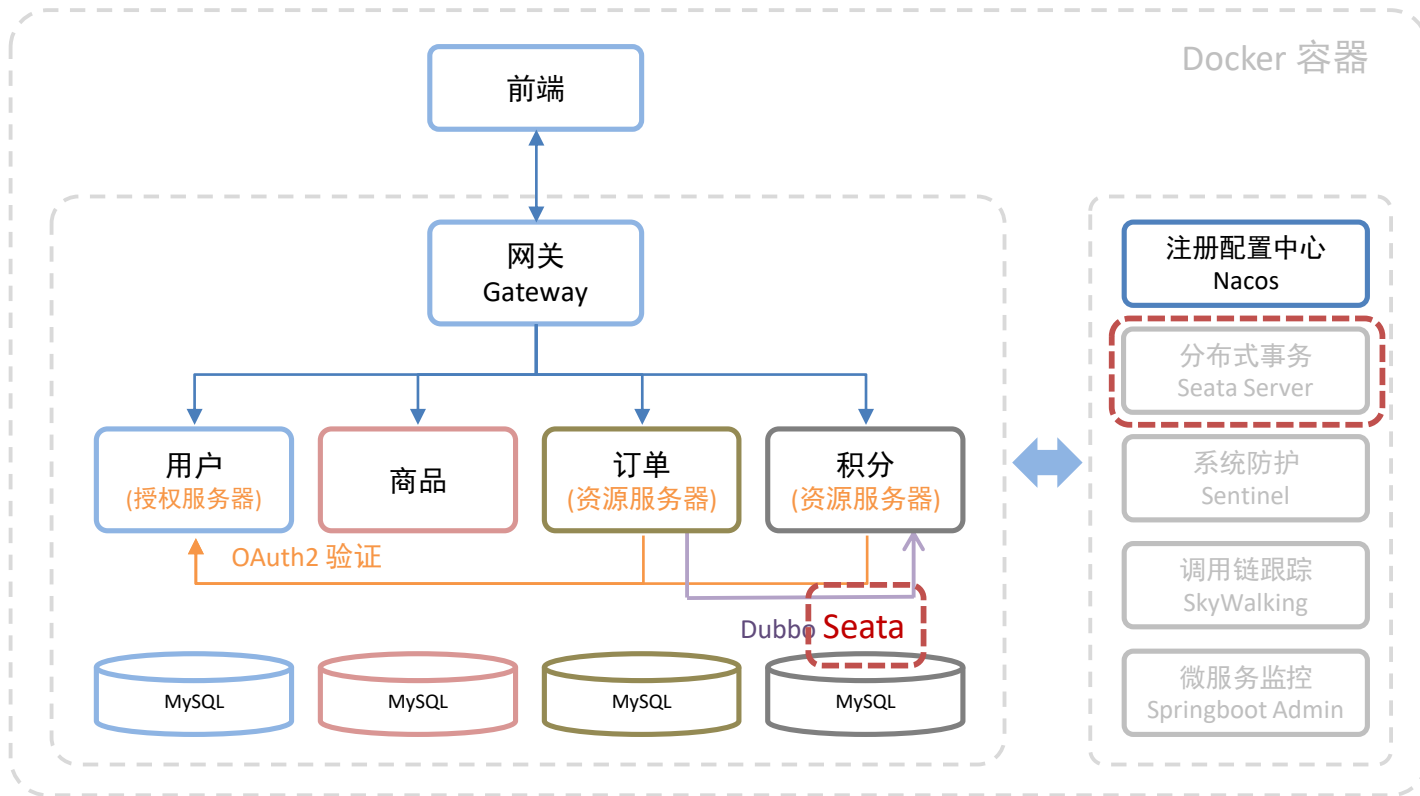


# 目录 Contents

- ◆ 项目需求与规划
- ◆ Nacos 服务发现整合
- ◆ Gateway 服务网关整合
- ◆ OAuth2 安全认证整合
- ◆ Dubbo RPC 服务调用整合
- ◆ Seata 分布式事务整合
- ◆ Sentinel 系统防护整合
- ◆ Skywalking 调用链跟踪整合
- ◆ Springboot Admin 服务监控整合
- ◆ Nacos 配置中心整合
- ◆ Docker 容器化部署

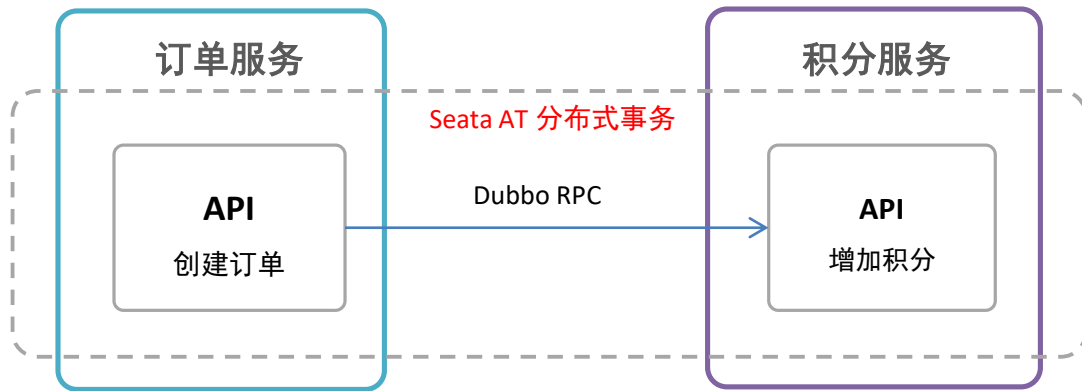


# Seata 分布式事务整合

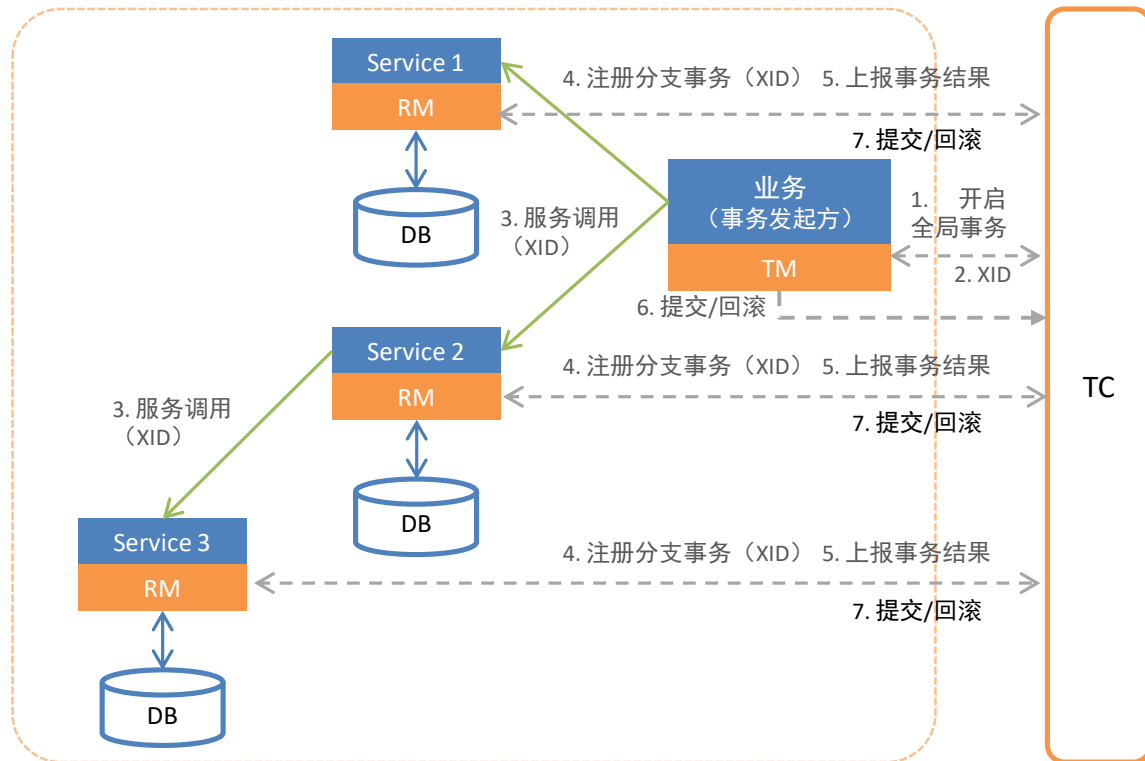


## 小节导学

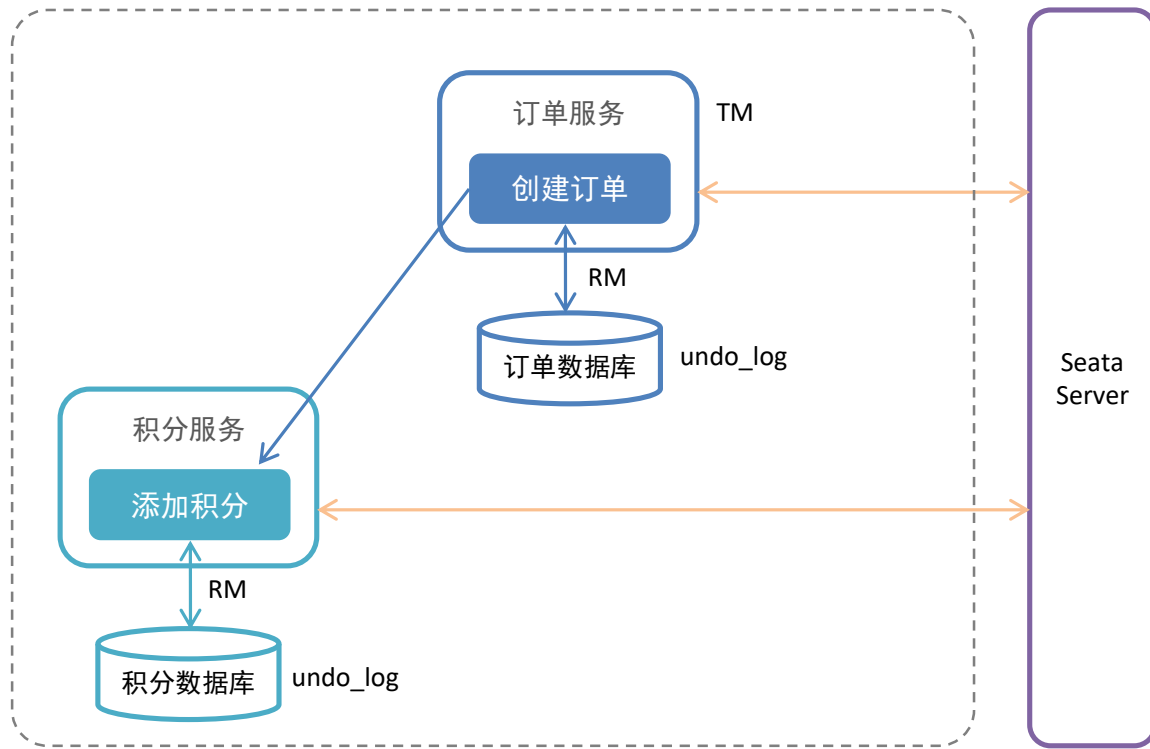
创建订单时需要调用添加积分接口，这个跨服务调用需要在一个分布式事务中，本节我们使用 Seata AT 模式处理。



## AT模式工作机制



## 整体结构



## 实践流程

### 积分服务 (服务提供者)

创建 undo\_log 表

加依赖 (seata)

替换为 seata DataSource

配置 seata

### 订单服务 (服务消费者)

创建 undo\_log 表

加依赖 (seata)

替换为 seata DataSource

配置 seata

开启全局事务

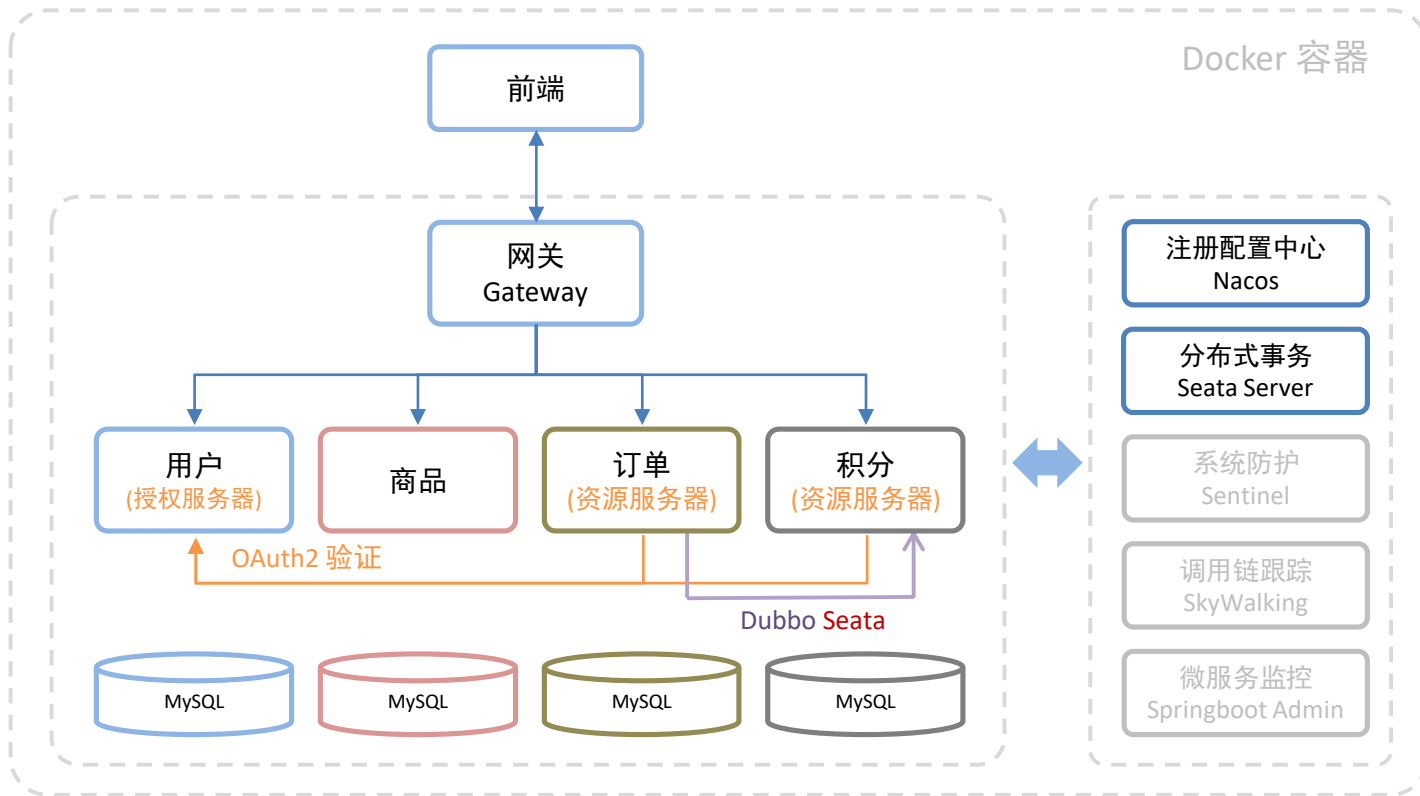


## 总结

### 重难点

1. 服务整合 Seata AT 模式的流程

# 分布式事务整合-总结



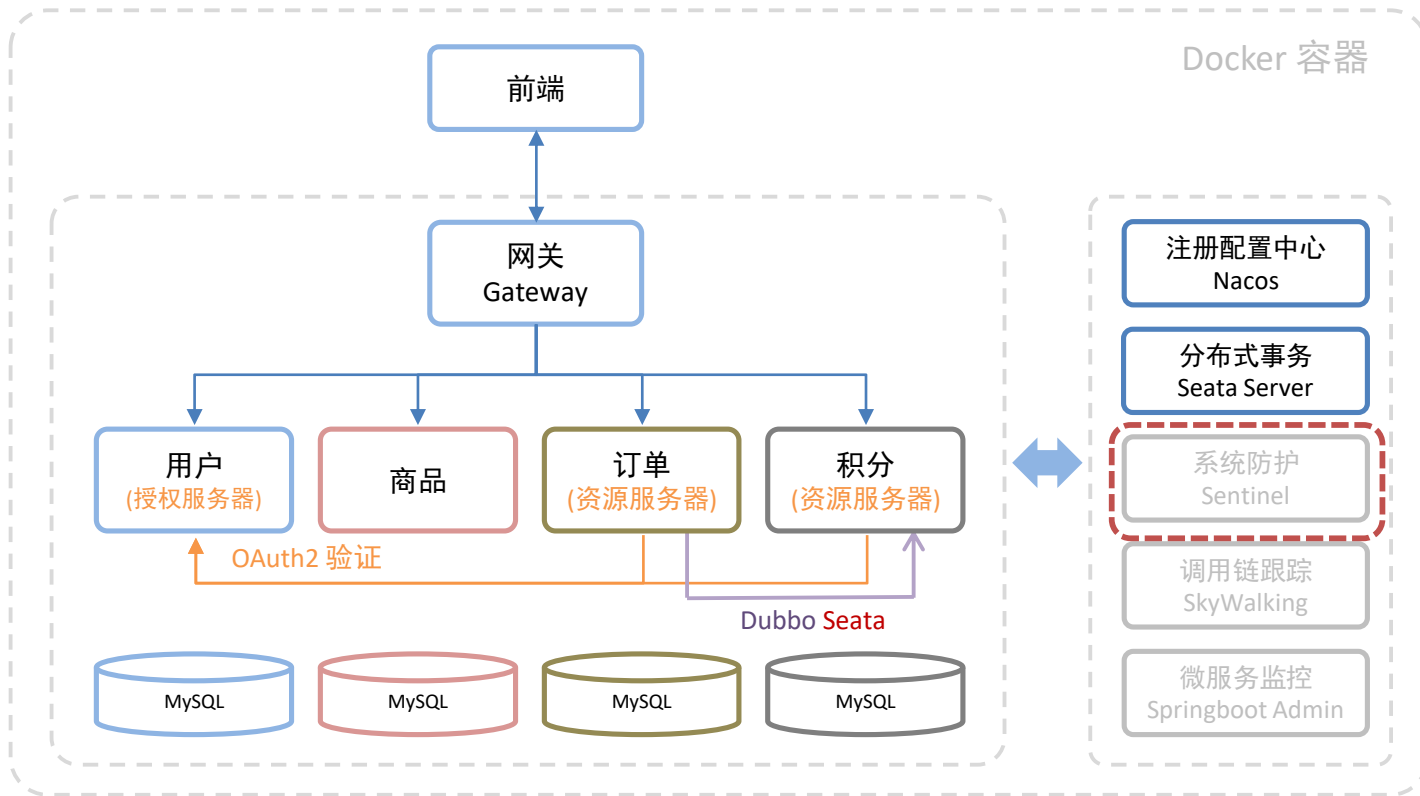


# 目录 Contents

- ◆ 项目需求与规划
- ◆ Nacos 服务发现整合
- ◆ Gateway 服务网关整合
- ◆ OAuth2 安全认证整合
- ◆ Dubbo RPC 服务调用整合
- ◆ Seata 分布式事务整合
- ◆ Sentinel 系统防护整合
- ◆ Skywalking 调用链跟踪整合
- ◆ Springboot Admin 服务监控整合
- ◆ Nacos 配置中心整合
- ◆ Docker 容器化部署

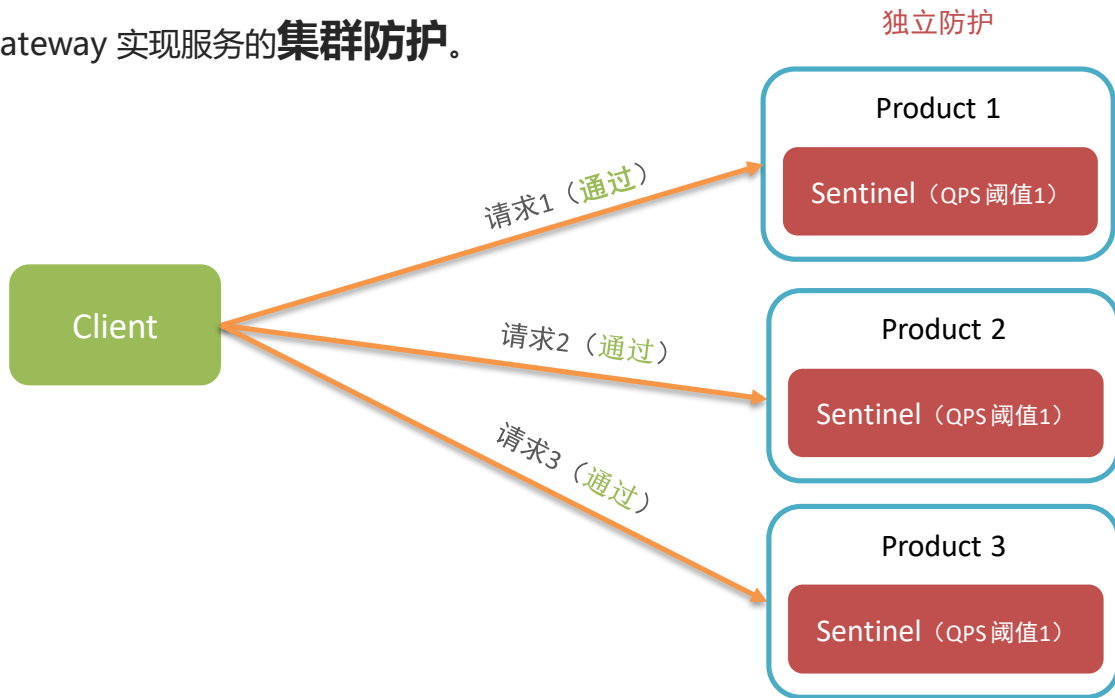


# ■ Sentinel 系统防护整合



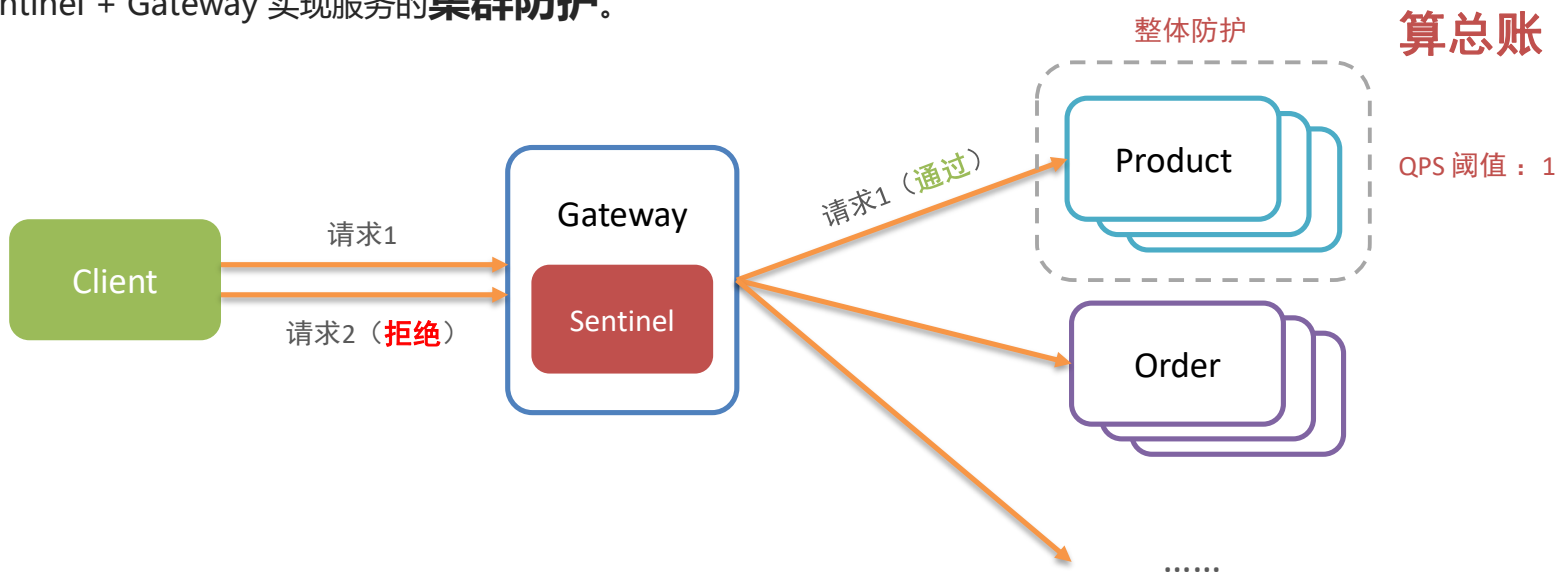
## 小节导学

Sentinel + Gateway 实现服务的**集群防护**。



## 小节导学

Sentinel + Gateway 实现服务的**集群防护**。



## 实践流程



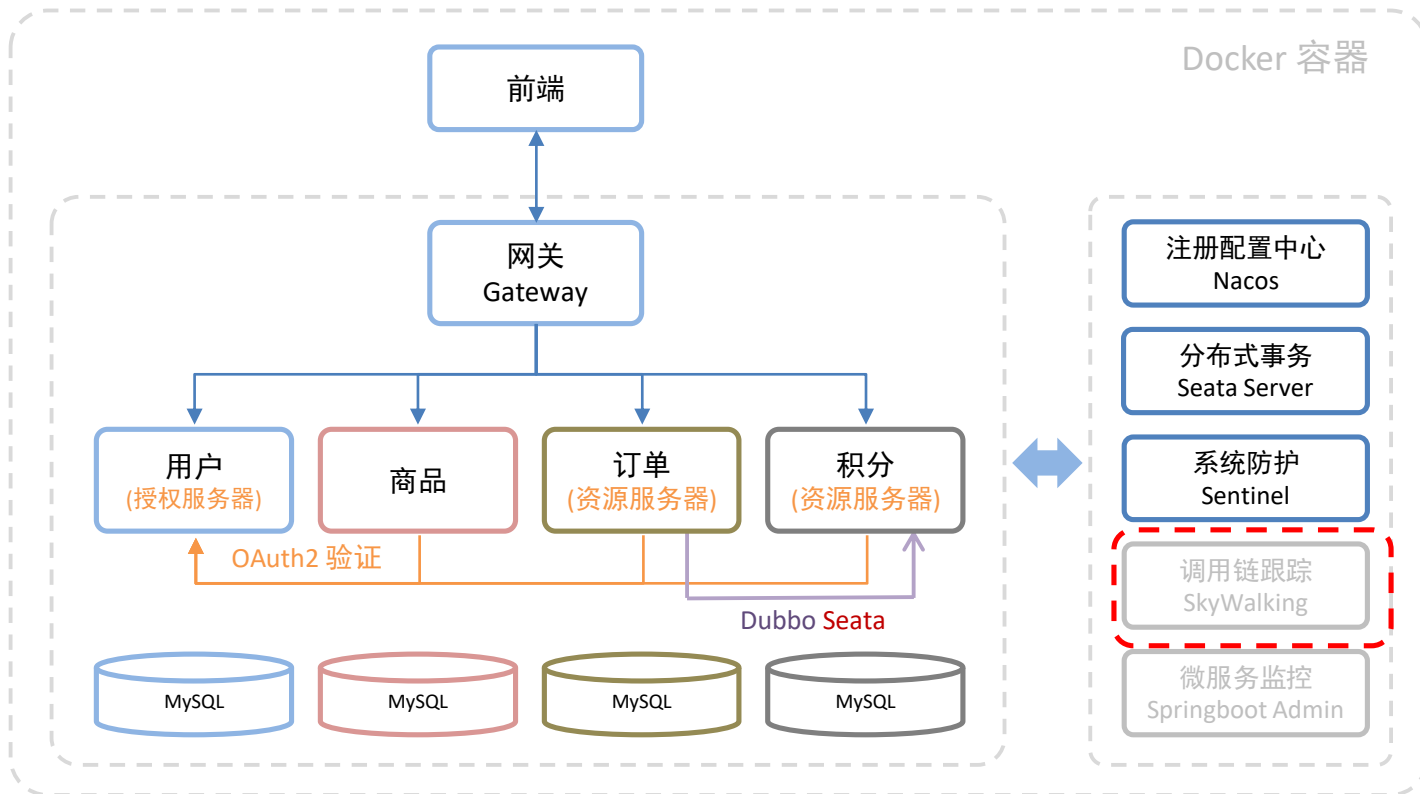


## 总结

### 重难点

1. Sentinel + Gateway 的整合流程

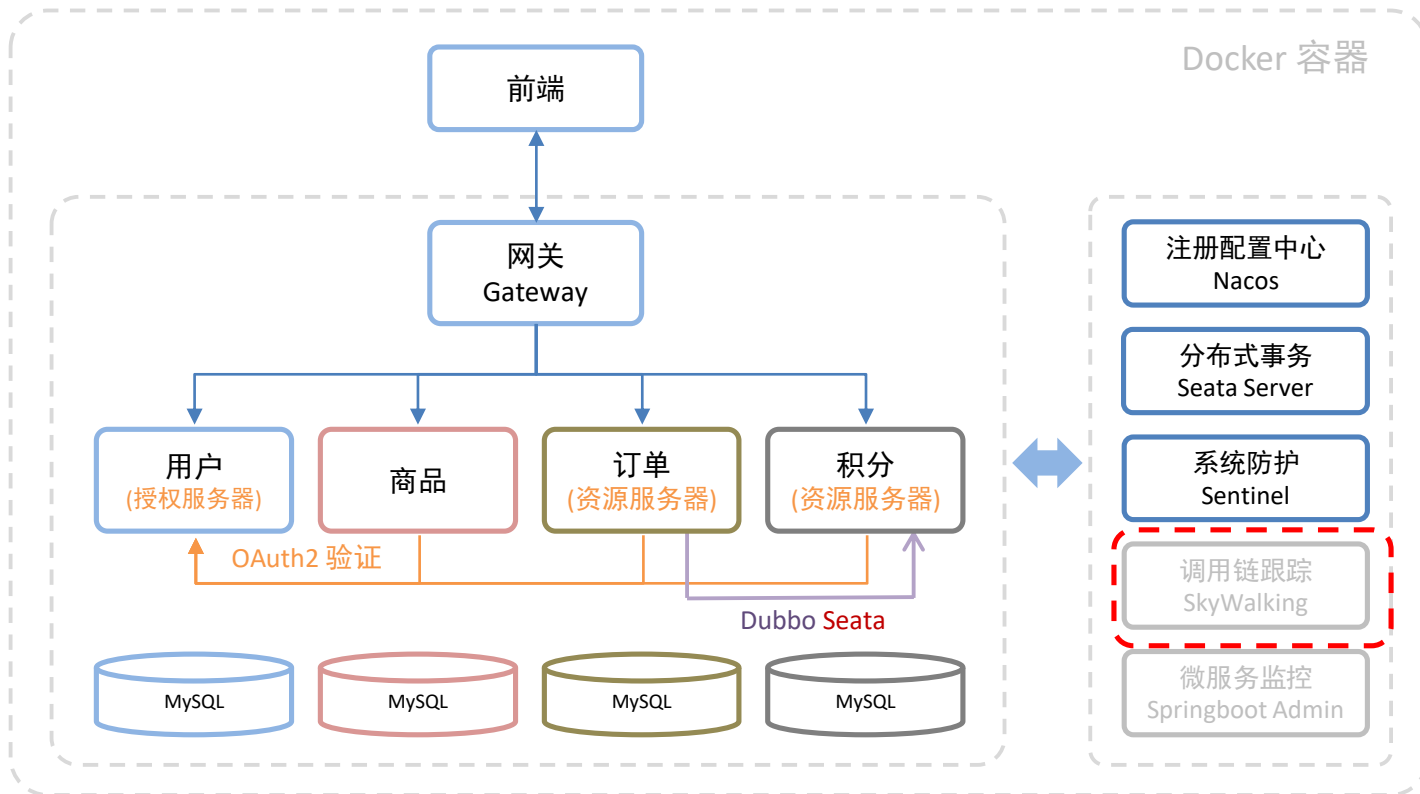
# ■ Sentinel 系统防护整合 - 总结



# 目录 Contents

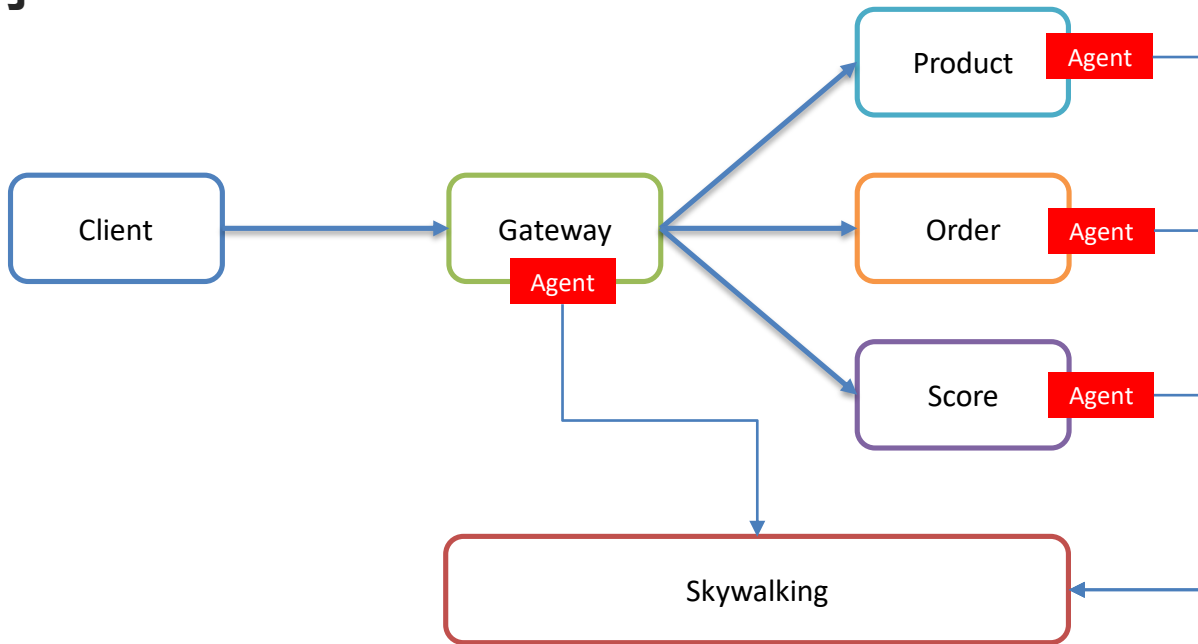
- ◆ 项目需求与规划
- ◆ Nacos 服务发现整合
- ◆ Gateway 服务网关整合
- ◆ OAuth2 安全认证整合
- ◆ Dubbo RPC 服务调用整合
- ◆ Seata 分布式事务整合
- ◆ Sentinel 系统防护整合
- ◆ Skywalking 调用链跟踪整合
- ◆ Springboot Admin 服务监控整合
- ◆ Nacos 配置中心整合
- ◆ Docker 容器化部署

# Skywalking 调用链跟踪整合

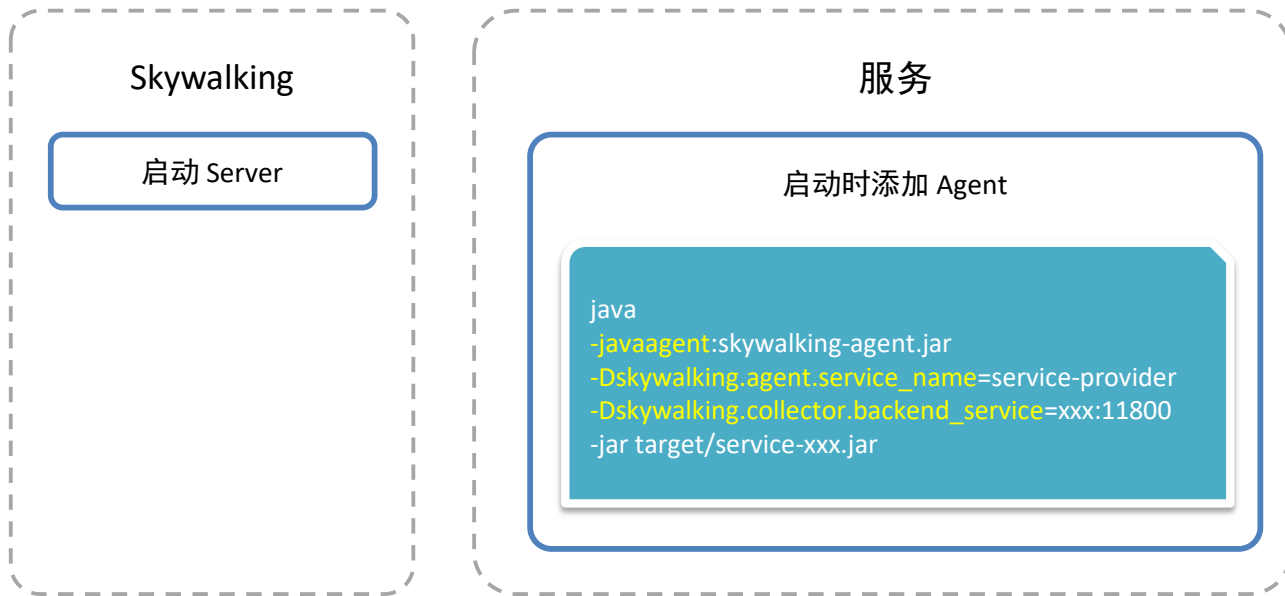




## 小节导学



## 实践流程



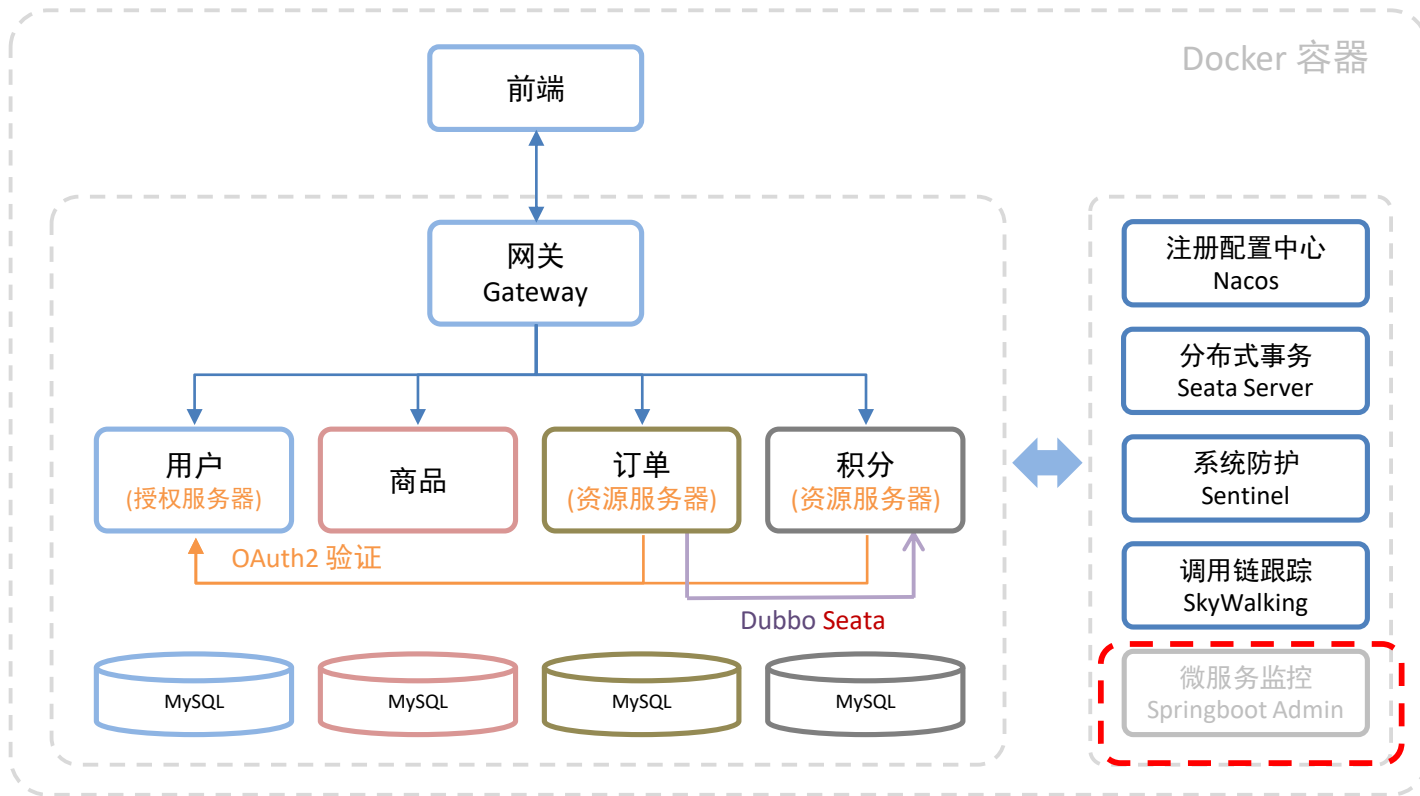


## 总结

### 重难点

1. 服务整合 Skywalking 的流程

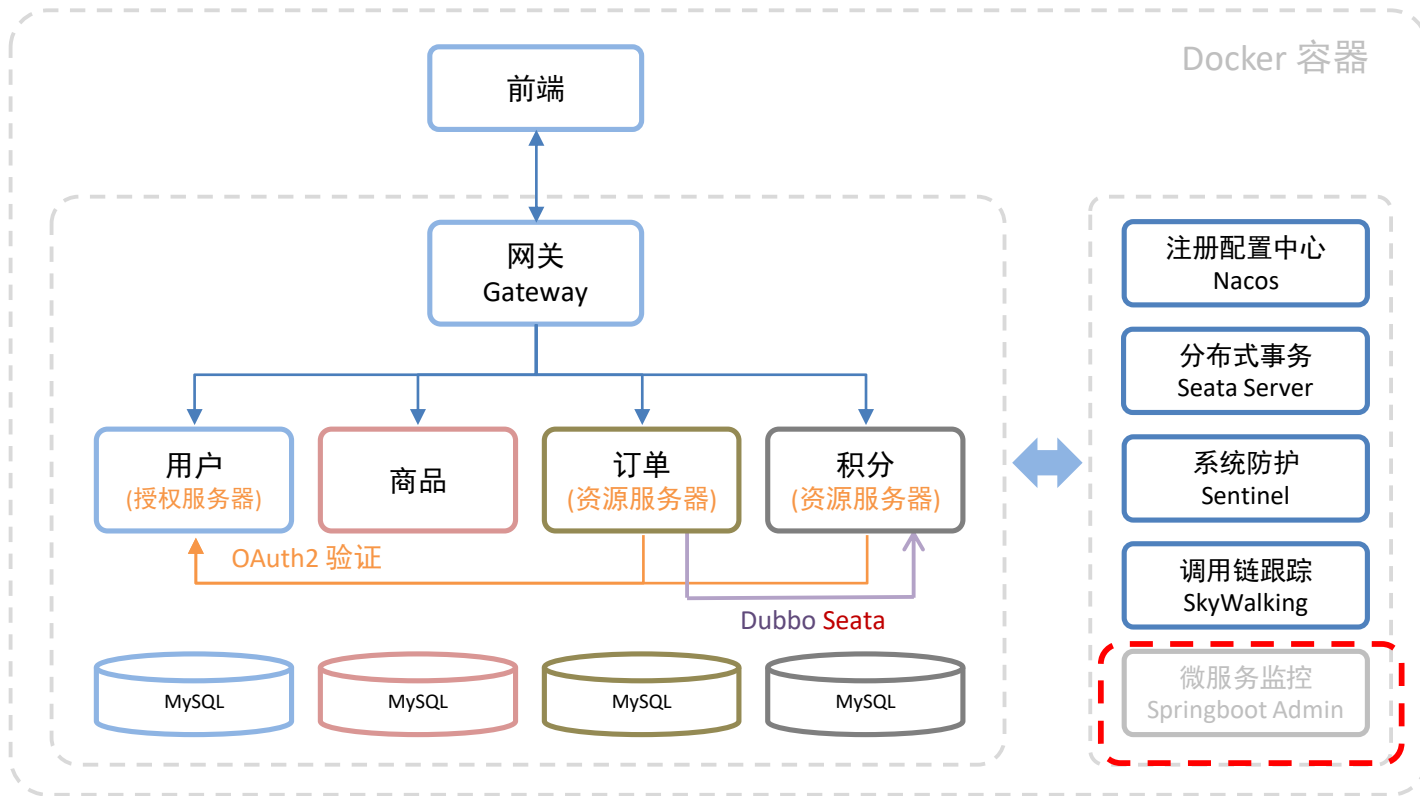
# Skywalking 调用链跟踪整合-总结



# 目录 Contents

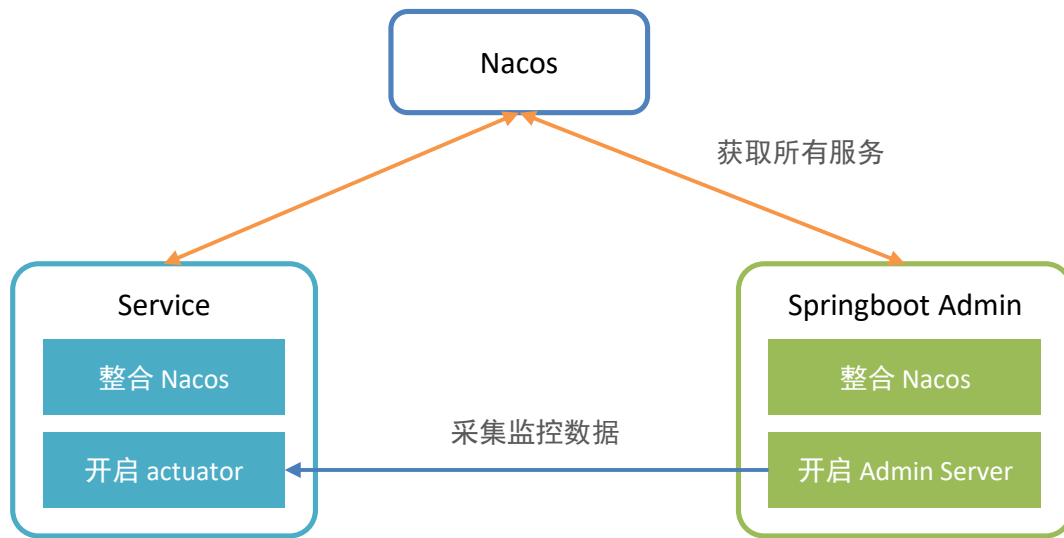
- ◆ 项目需求与规划
- ◆ Nacos 服务发现整合
- ◆ Gateway 服务网关整合
- ◆ OAuth2 安全认证整合
- ◆ Dubbo RPC 服务调用整合
- ◆ Seata 分布式事务整合
- ◆ Sentinel 系统防护整合
- ◆ Skywalking 调用链跟踪整合
- ◆ Springboot Admin 服务监控整合
- ◆ Nacos 配置中心整合
- ◆ Docker 容器化部署

# Springboot Admin 服务监控整合

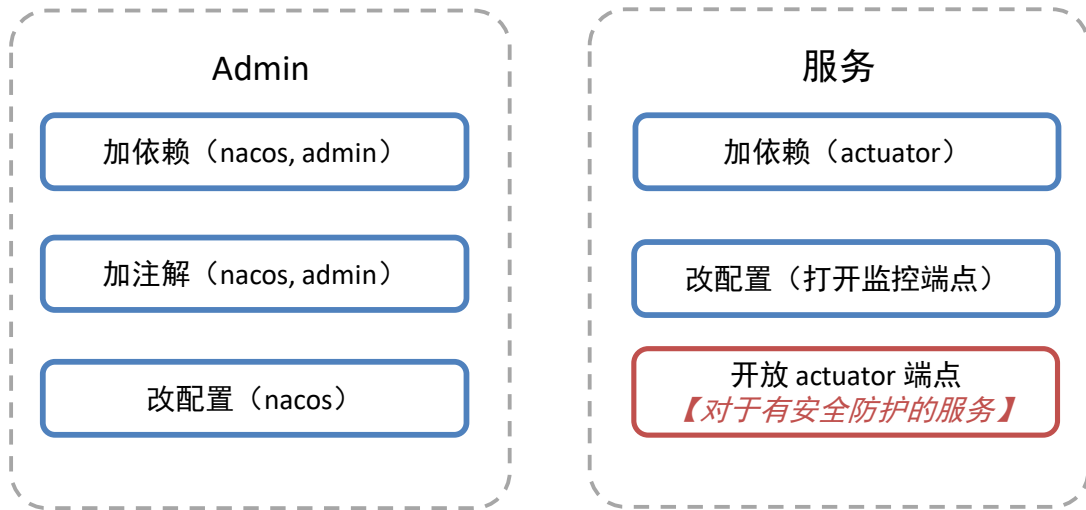


## 小节导学

使用 Springboot Admin 监控服务状态。



## 实践流程





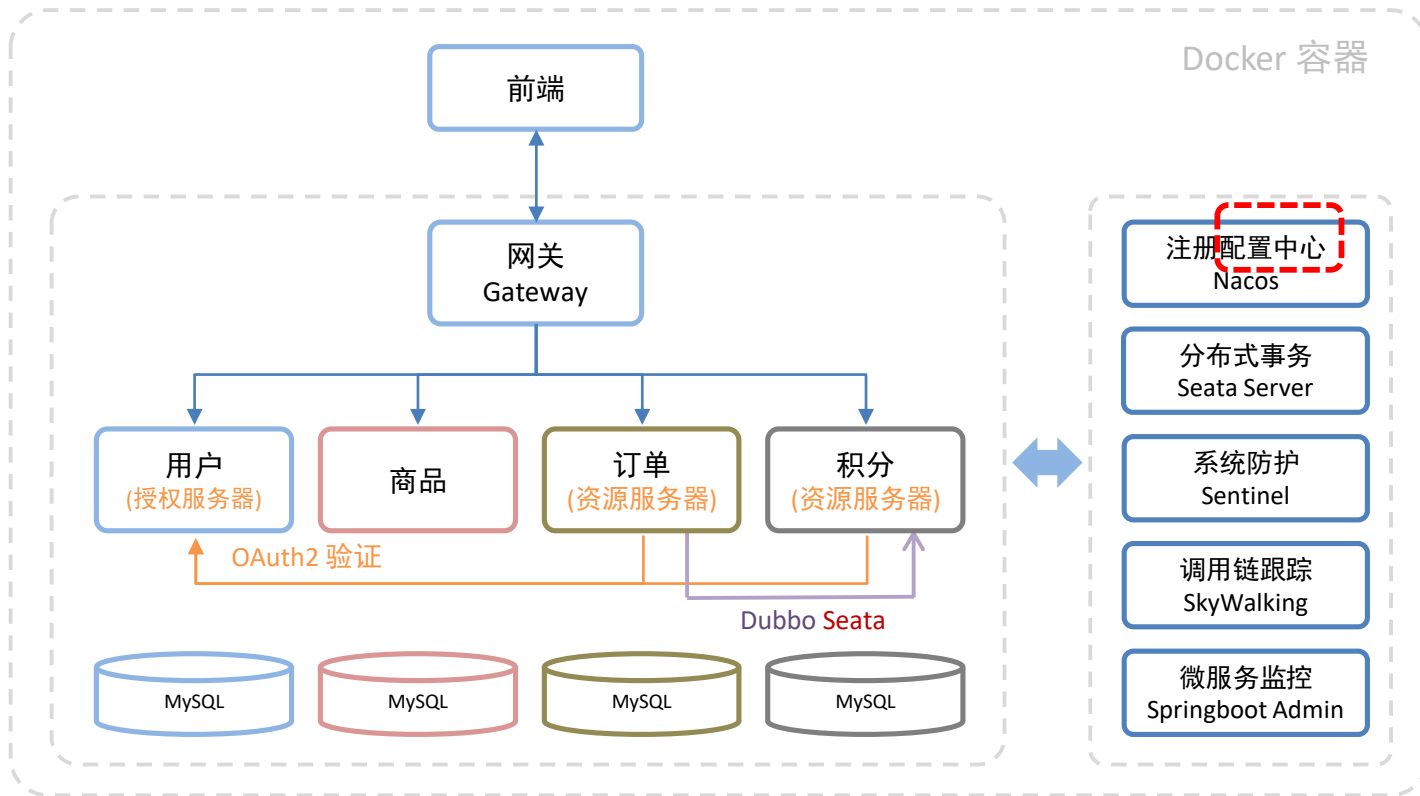


## 总结

### 重难点

1. 服务整合 Springboot Admin 的流程

# Springboot Admin 服务监控整合-总结

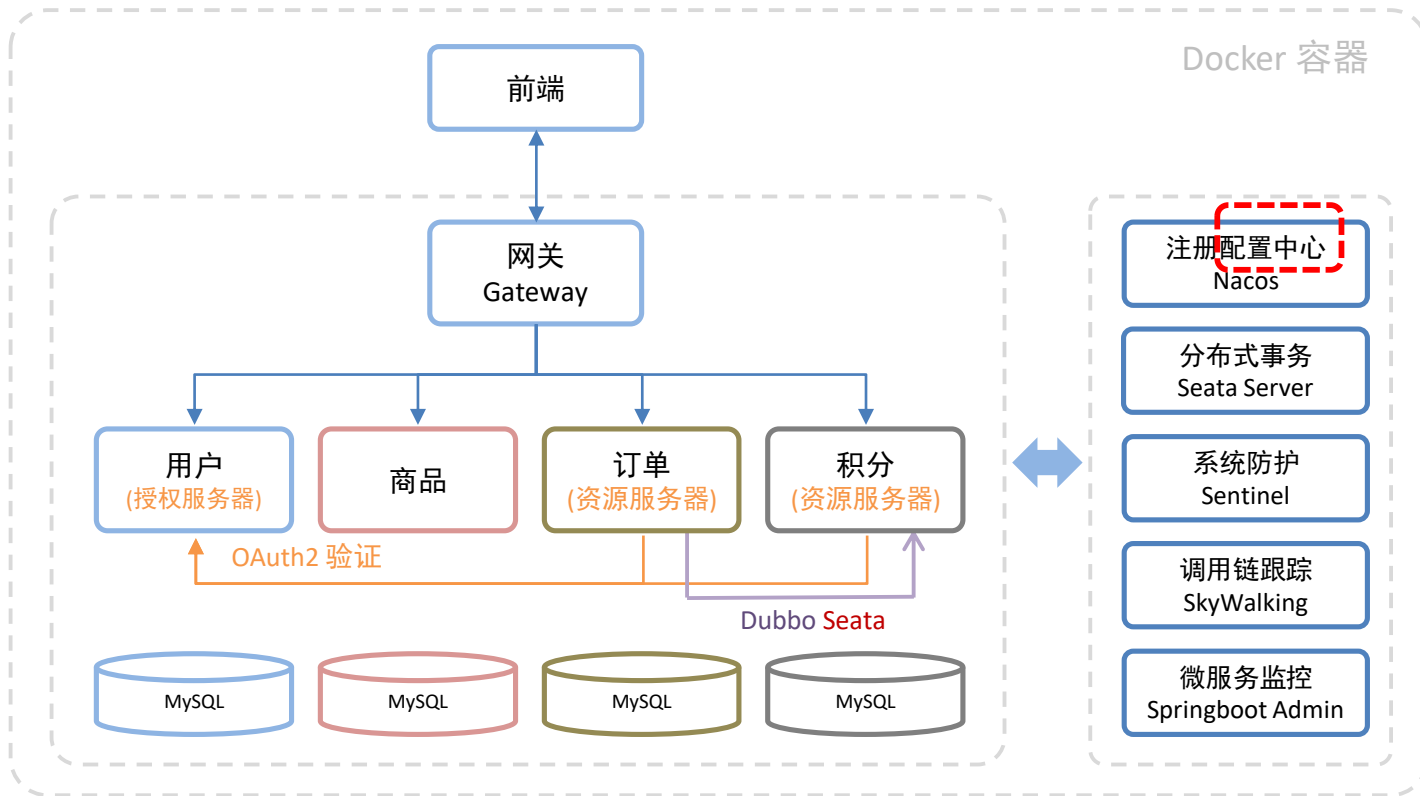




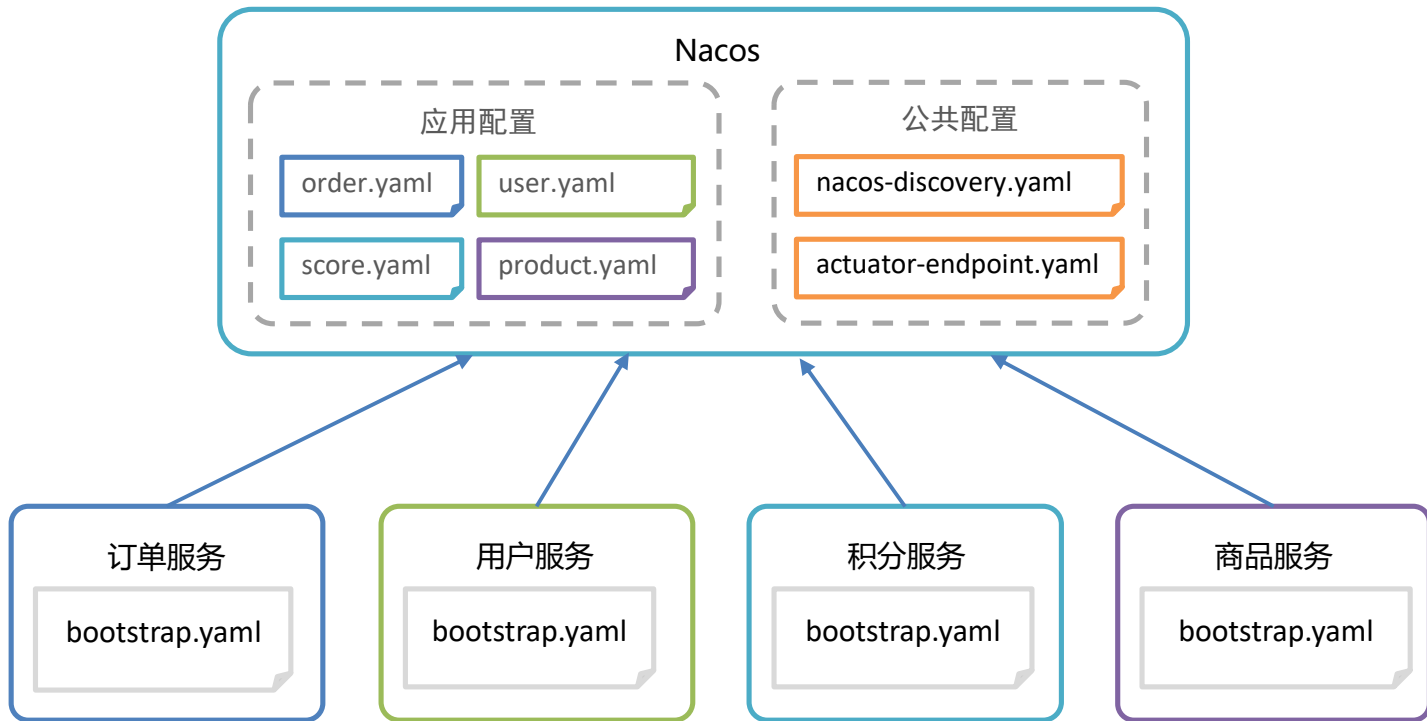
# 目录 Contents

- ◆ 项目需求与规划
- ◆ Nacos 服务发现整合
- ◆ Gateway 服务网关整合
- ◆ OAuth2 安全认证整合
- ◆ Dubbo RPC 服务调用整合
- ◆ Seata 分布式事务整合
- ◆ Sentinel 系统防护整合
- ◆ Skywalking 调用链跟踪整合
- ◆ Springboot Admin 服务监控整合
- ◆ **Nacos 配置中心整合**
- ◆ Docker 容器化部署

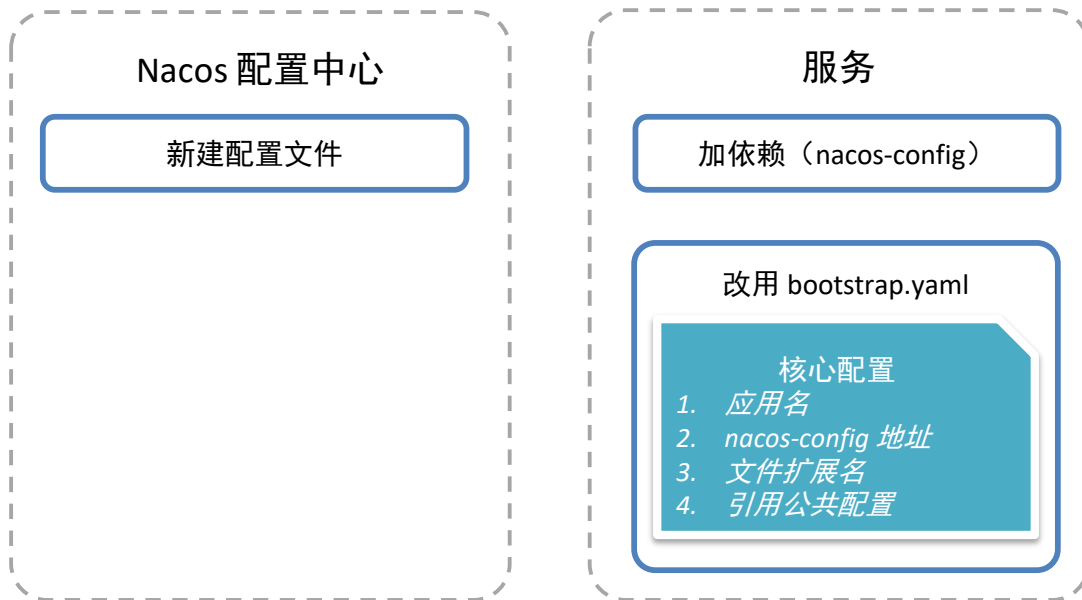
# Nacos 配置中心整合



## 小节导学



## 实践流程



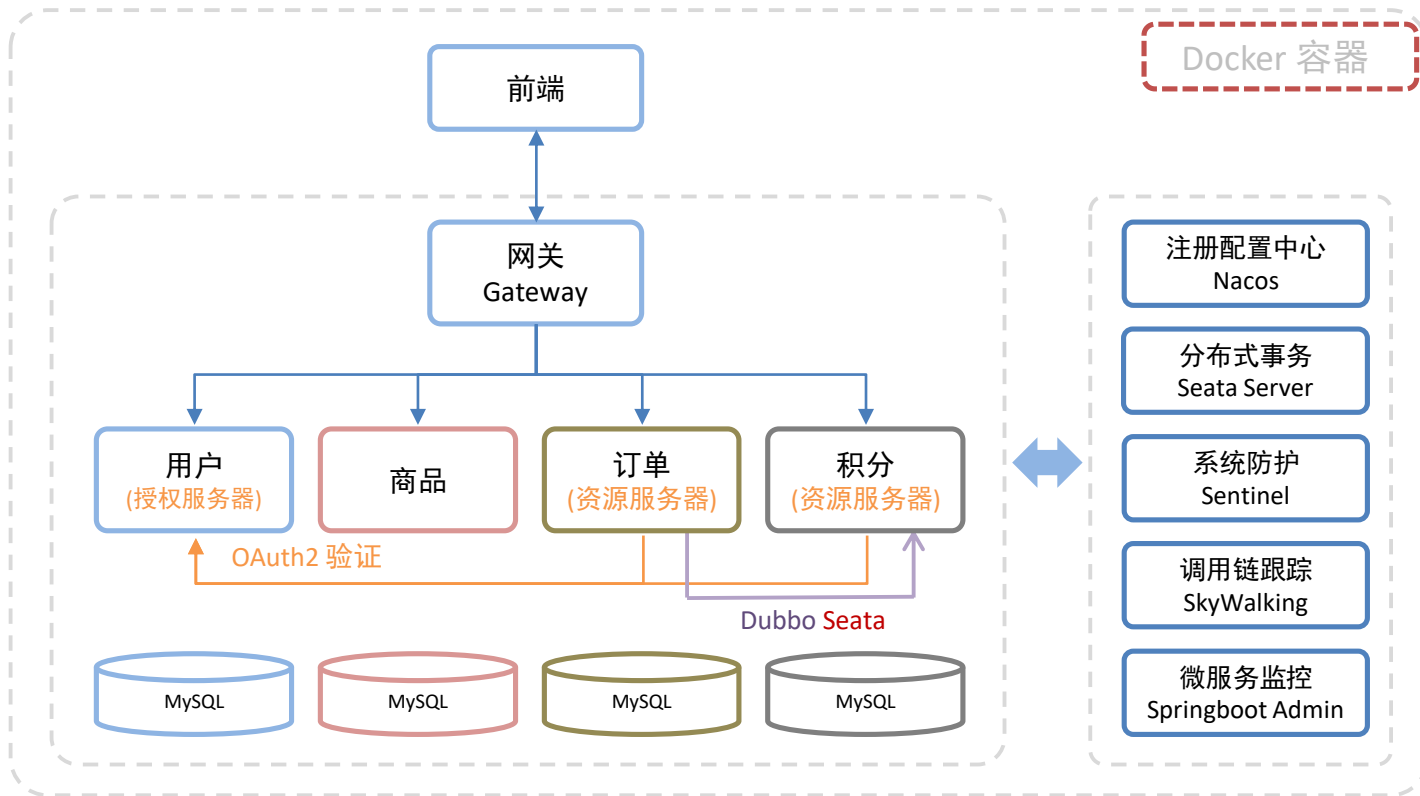


## 总结

### 重难点

1. 服务整合 Nacos 配置中心的流程

# Nacos 配置中心整合 - 总结

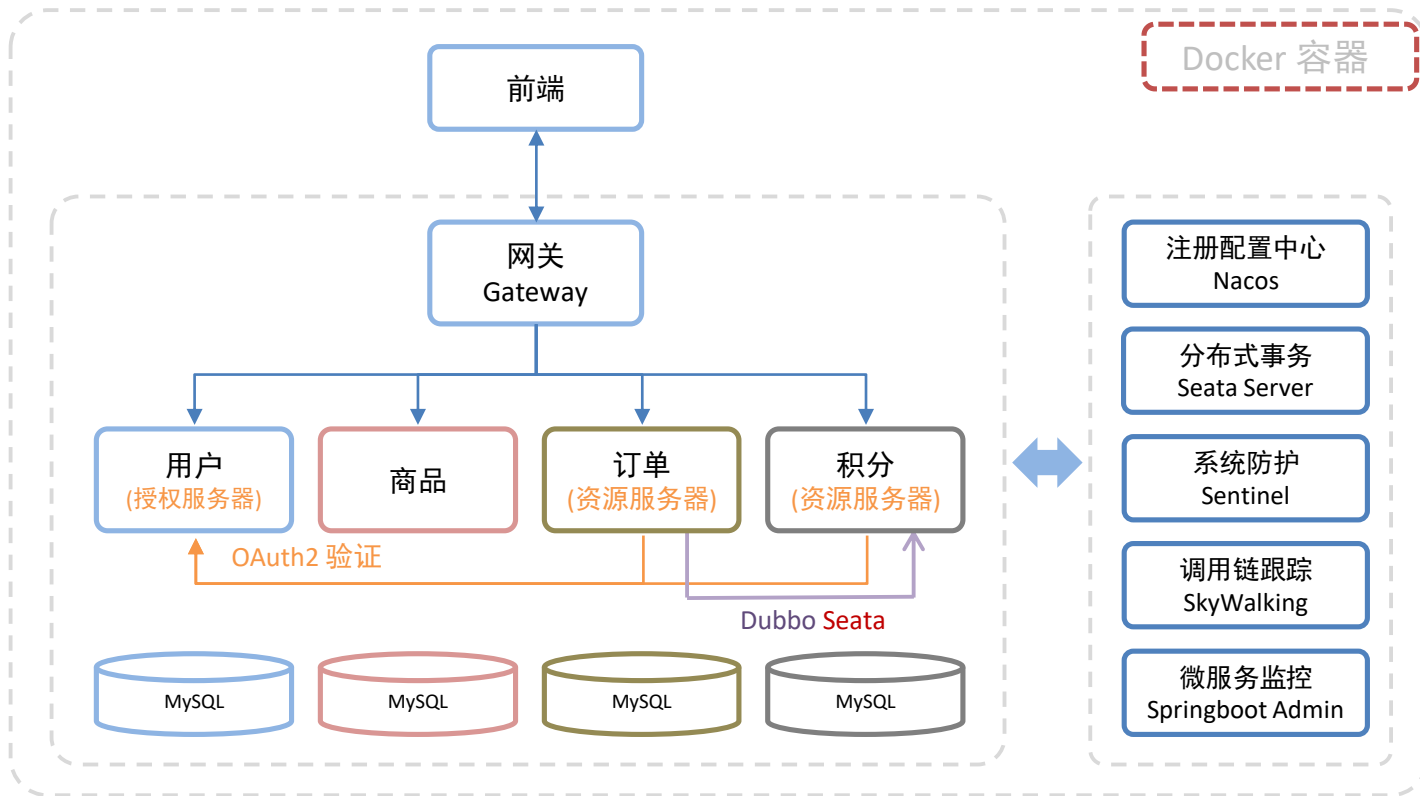




# 目录 Contents

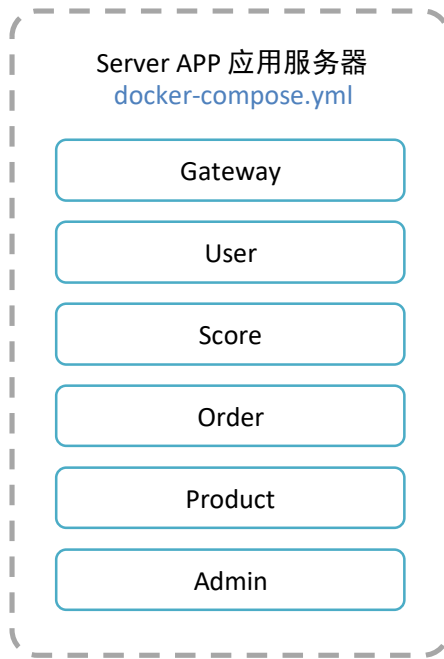
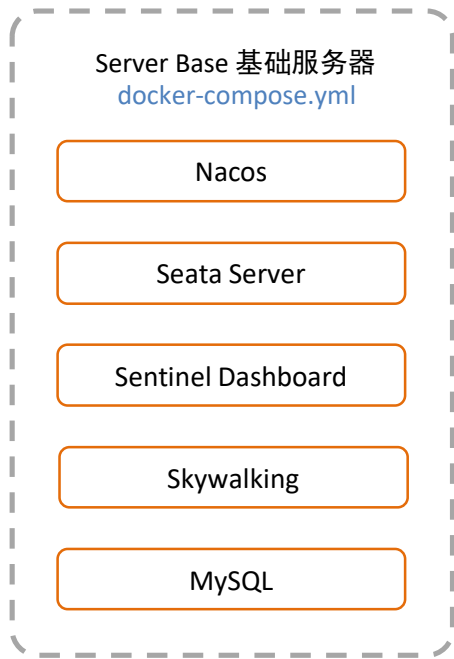
- ◆ 项目需求与规划
- ◆ Nacos 服务发现整合
- ◆ Gateway 服务网关整合
- ◆ OAuth2 安全认证整合
- ◆ Dubbo RPC 服务调用整合
- ◆ Seata 分布式事务整合
- ◆ Sentinel 系统防护整合
- ◆ Skywalking 调用链跟踪整合
- ◆ Springboot Admin 服务监控整合
- ◆ Nacos 配置中心整合
- ◆ Docker 容器化部署

# Nacos 配置中心整合 - 总结



## 小节导学

所有服务都使用容器方式运行，使用 docker compose 一键启动。



## 实践流程

Server Base 基础服务器

docker-compose.yml

Server APP 应用服务器

服务构建镜像

docker-compose.yml

容器启动时整合 Agent

## 服务构建镜像

目标：maven 打包时直接构建出镜像。

步骤：

### 1. Dockerfile

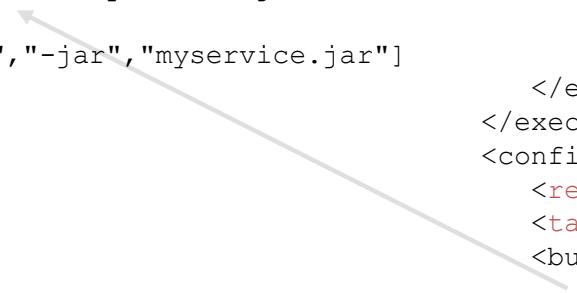
```
FROM java:8

ARG JAR_FILE
ADD target/${JAR_FILE} myservice.jar

ENTRYPOINT ["java","-jar","myservice.jar"]
```

### 2. dockerfile-maven-plugin

```
<plugin>
  <groupId>com.spotify</groupId>
  <artifactId>dockerfile-maven-plugin</artifactId>
  <version>1.4.13</version>
  <executions>
    <execution>
      <id>default</id>
      <goals>
        <goal>build</goal>
      </goals>
    </execution>
  </executions>
  <configuration>
    <repository>${project.name}</repository>
    <tag>${project.version}</tag>
    <buildArgs>
      <JAR_FILE>${project.build.finalName}.jar</JAR_FILE>
    </buildArgs>
  </configuration>
</plugin>
```



## 容器启动时整合 Agent

docker-compose

```
gateway:
  ...
  volumes:
    - ./skywalking:/home
  entrypoint: [
    "java",
    "-javaagent:/home/agent/skywalking-agent.jar",
    "-Dskywalking.agent.service_name=gateway",
    "-Dskywalking.collector.backend_service=server-base:11800",
    "-jar","myservice.jar"
  ]
```

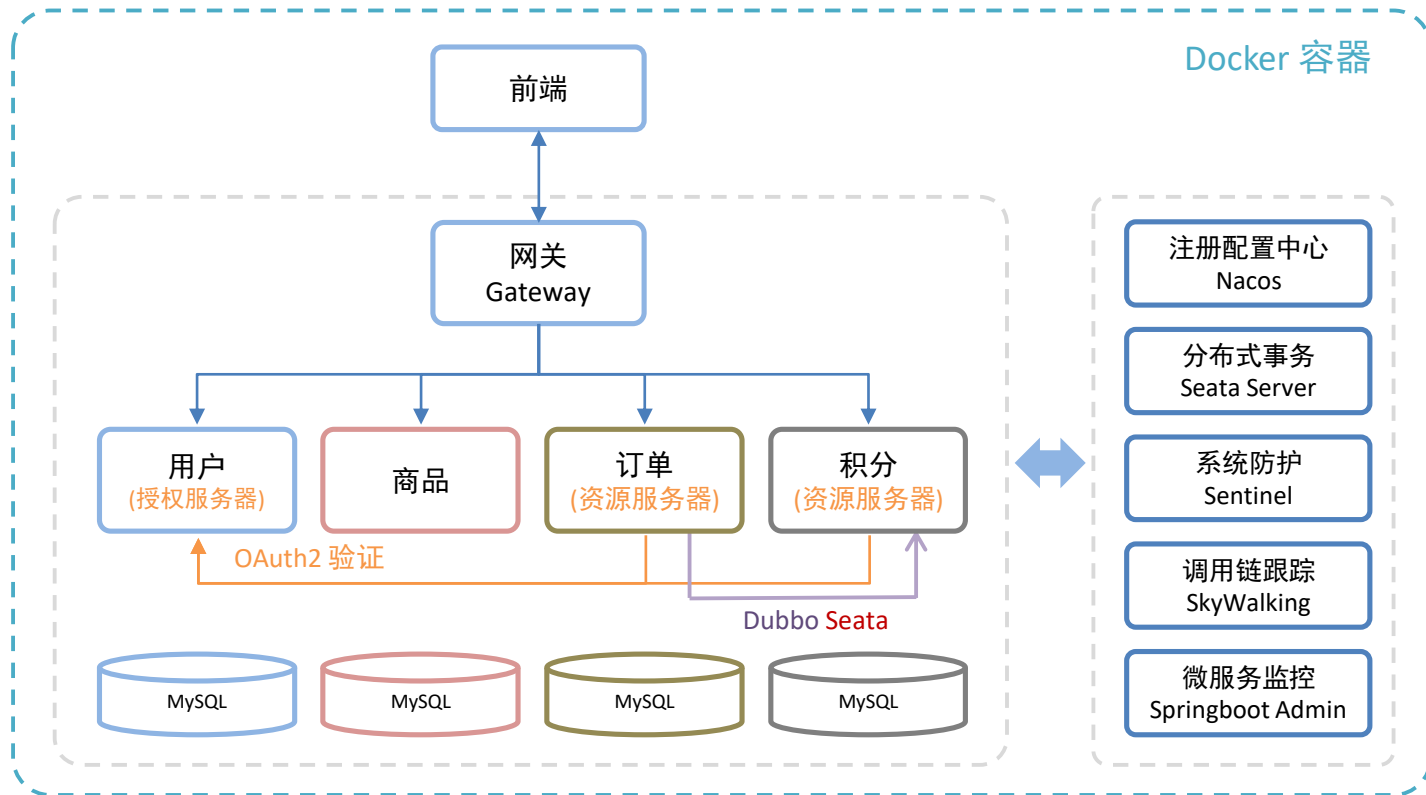


## 总结

### 重难点

1. dockerfile-maven-plugin
2. 容器启动时整合 Agent 的用法

# Docker 容器化部署 - 总结







一样的在线教育，不一样的教学品质