



基于Dubbo的微服务架构 在电信运营商体系下的演进

分享人：仵文辉

亚信科技 互联网创新与服务中心

演讲主旨



微服务架构技术选型

微服务架构下的技术选择

包括：服务治理、服务监控、配置中心、日志收集、服务网关等技术选型



在使用dubbo时遇到的问题与解决之道

整个运营上IT架构体系与网络环境下如何解决

包括：安全问题、集成问题、容器化等挑战及使用规范



成功案例与感悟

- 基于上述架构的成功案例
- 一些感悟

演进历程

新时代:抛弃沉重的内部框架，全面拥抱开源，轻装上阵.

服务化

2009年开始基于服务化历程改造：采用自研发轻量级实现服务治理、协议适配、服务编排。

2009年

Docker化

容器化改造，实现中间件、微服务的容器化（主要为Dubbo容器化、运维工具容器化），实现开箱即用，用完即走，动态扩缩

2017年

微服务化

引入微服务框架与思想，实现服务的拆分、微服务治理、链路跟踪。

2015年

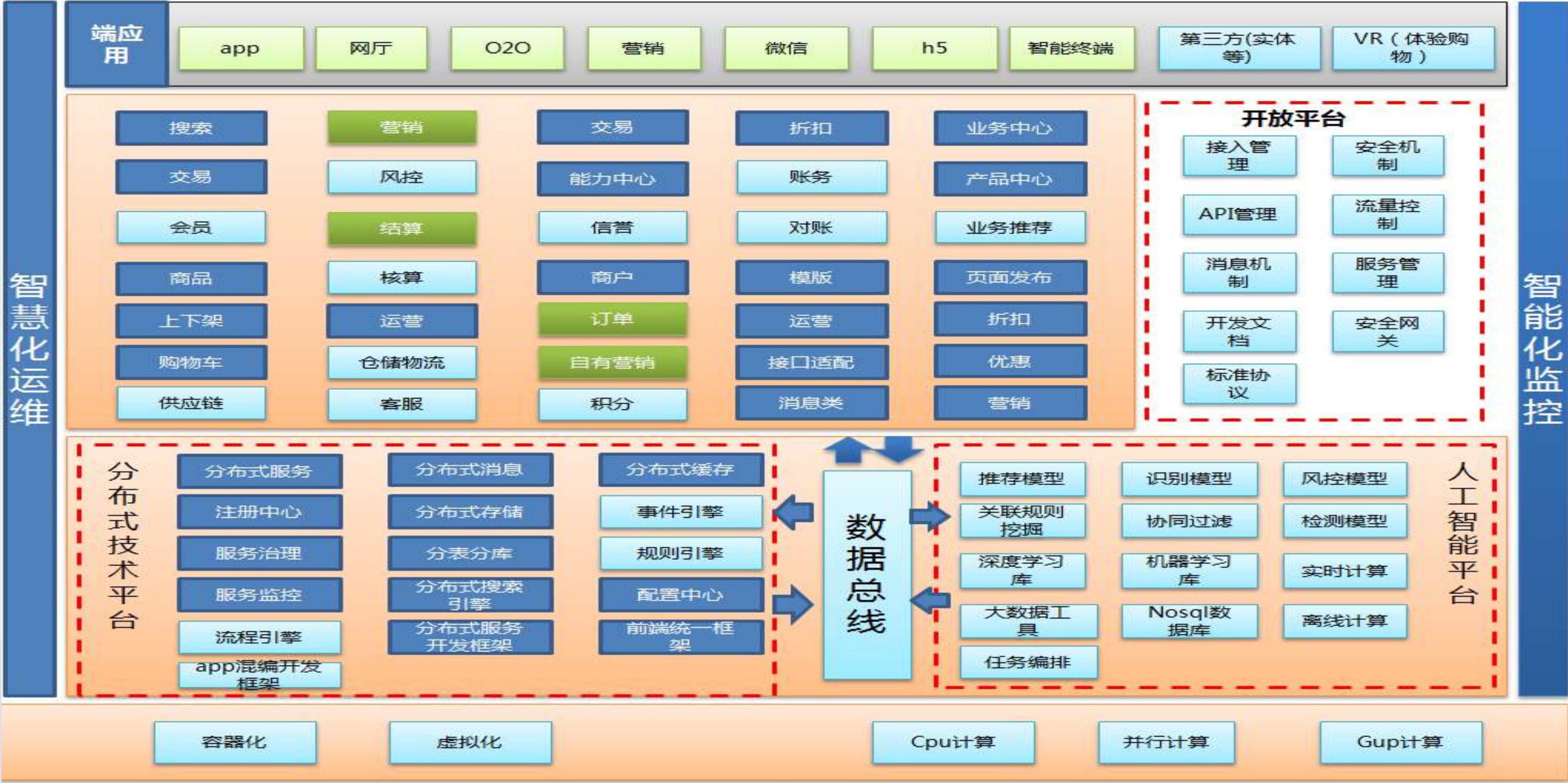
智慧化

基于AI技术全面实现服务治理、链路监控、配置中心、服务监控、智慧化运维等，形成一个有温度、有思想、自学习的架构。

2018年 +

总体架构

- 指导思想：端应用、大中台、平台化、智能化
- 实现：多端应用 + 统一大中台 + 技术平台化 + 运维智能化的“有思想有温度”的架构



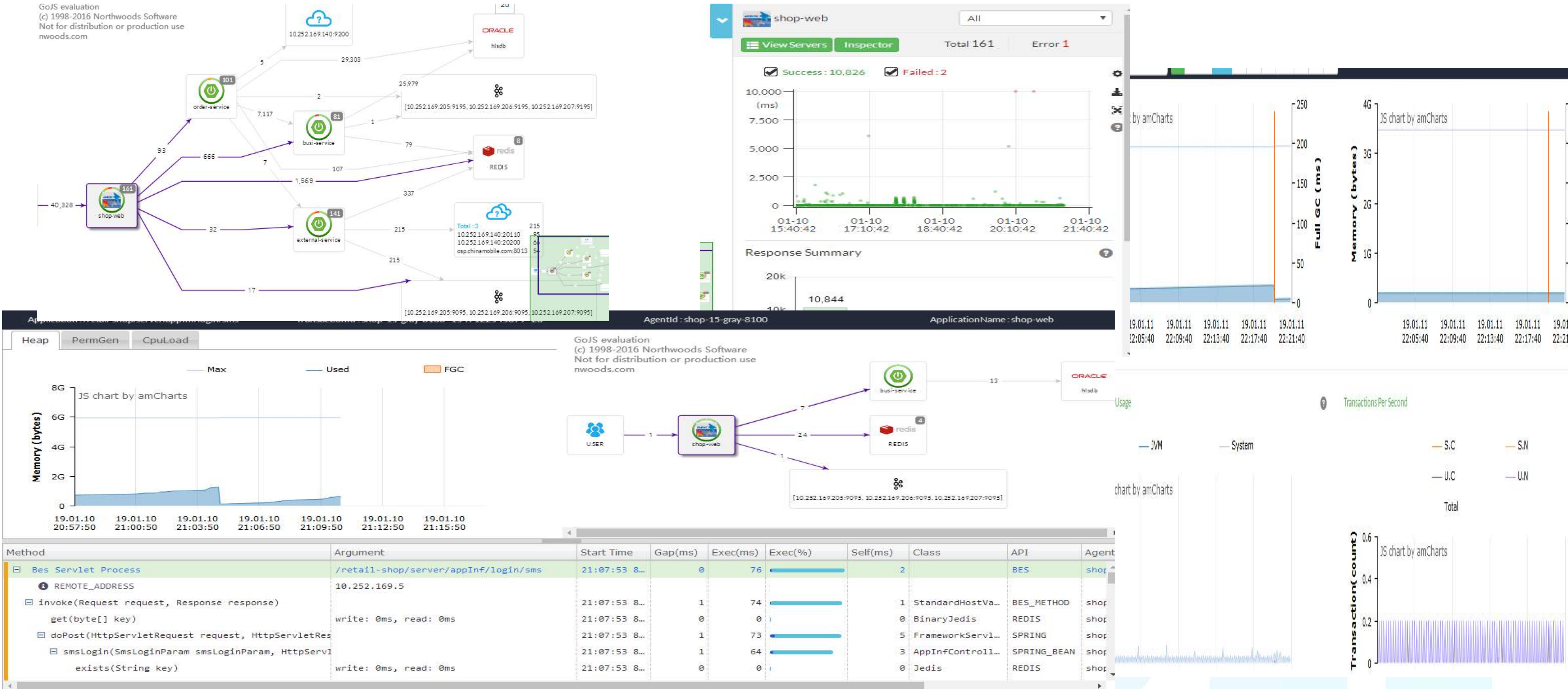
微服务-服务治理

微服务技术选型：
Dubbo官方版本+自研发组件+开源组件（Dubbo开源套件）（官网开始维护后）

治理项	开源技术选型	扩展	Dubbo自带
服务熔断	Dubbo-Sentinel（开源后替换自研发模块）	扩展Dubbo自动熔断（基于Hystrix与redis两种方式） Hystrix会破坏Dubbo线程池	Dubbo-Admin只能手动熔断
服务降级	Dubbo-Sentinel（开源后替换自研发模块）	同上	Dubbo-Admin手动降级
服务路由			消费端负载均衡
服务注册		Curatorx客户端	ZooKeeper
服务编排		扩展服务编排器在消费端编排	
服务熔断	Dubbo-Sentinel（开源后替换自研发模块）	扩展Dubbo自动熔断（基于Hystrix与Redis两种方式） Hystrix会破坏Dubbo线程池	Dubbo-Admin只能手动熔断

微服务-服务监控

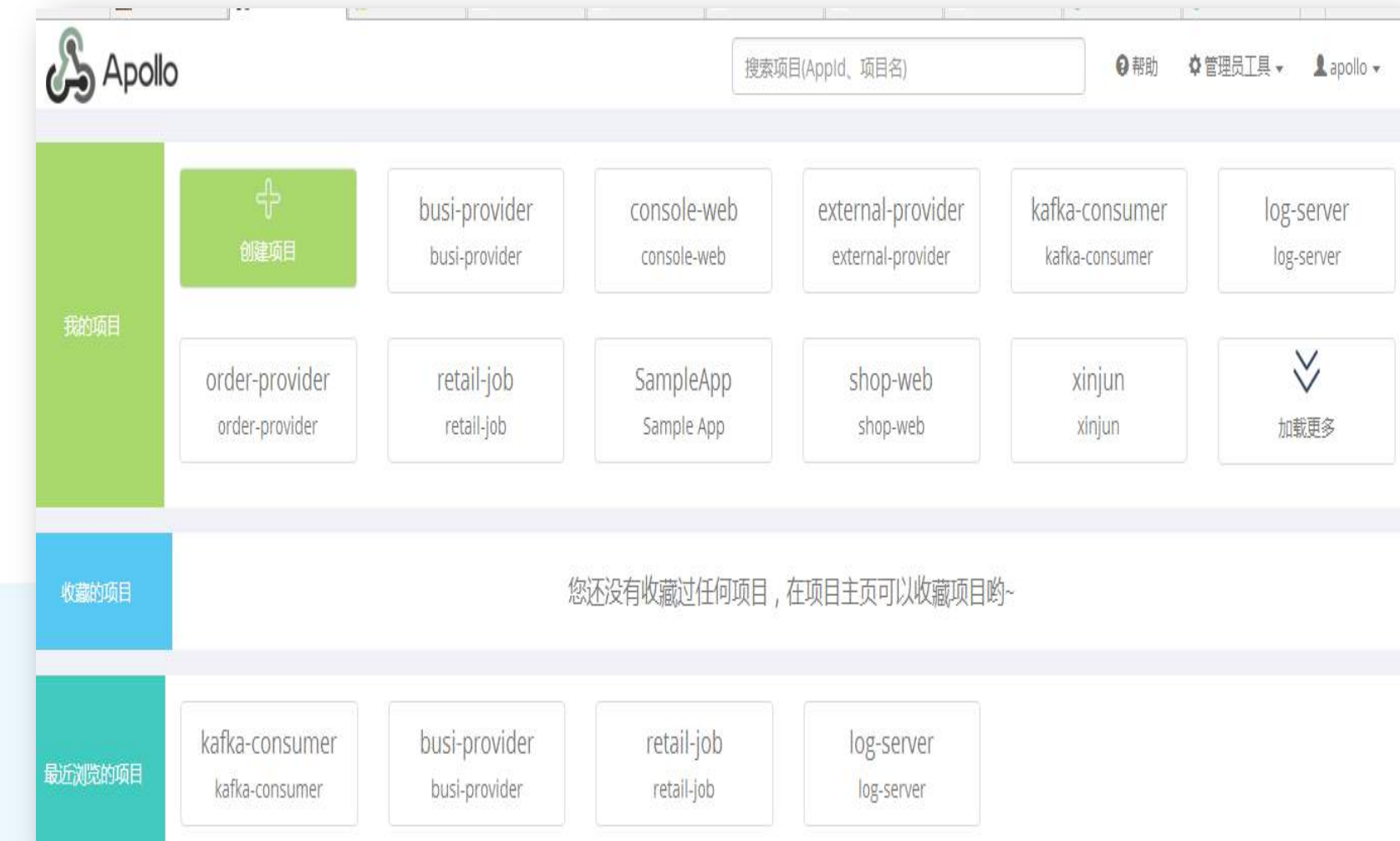
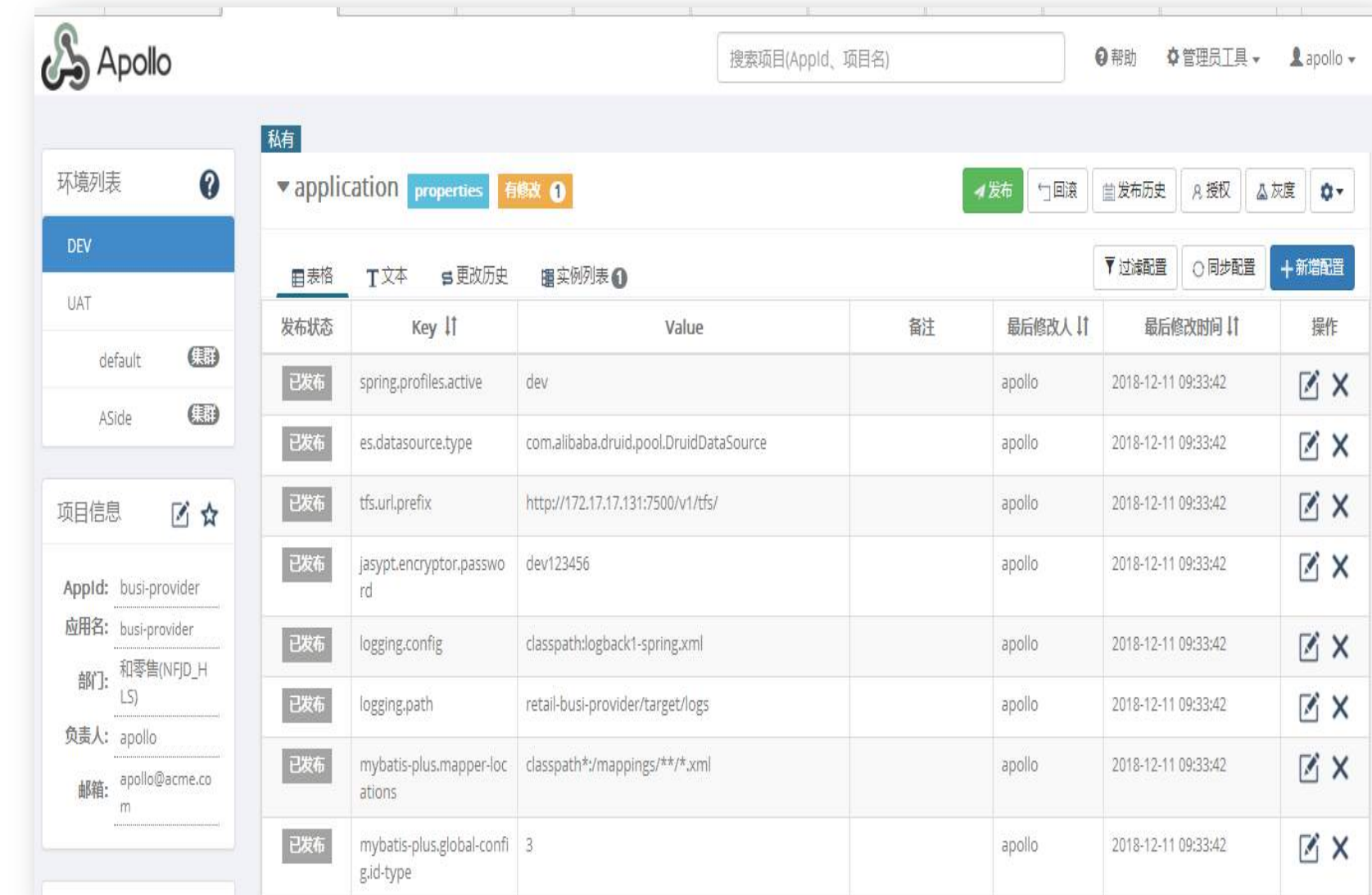
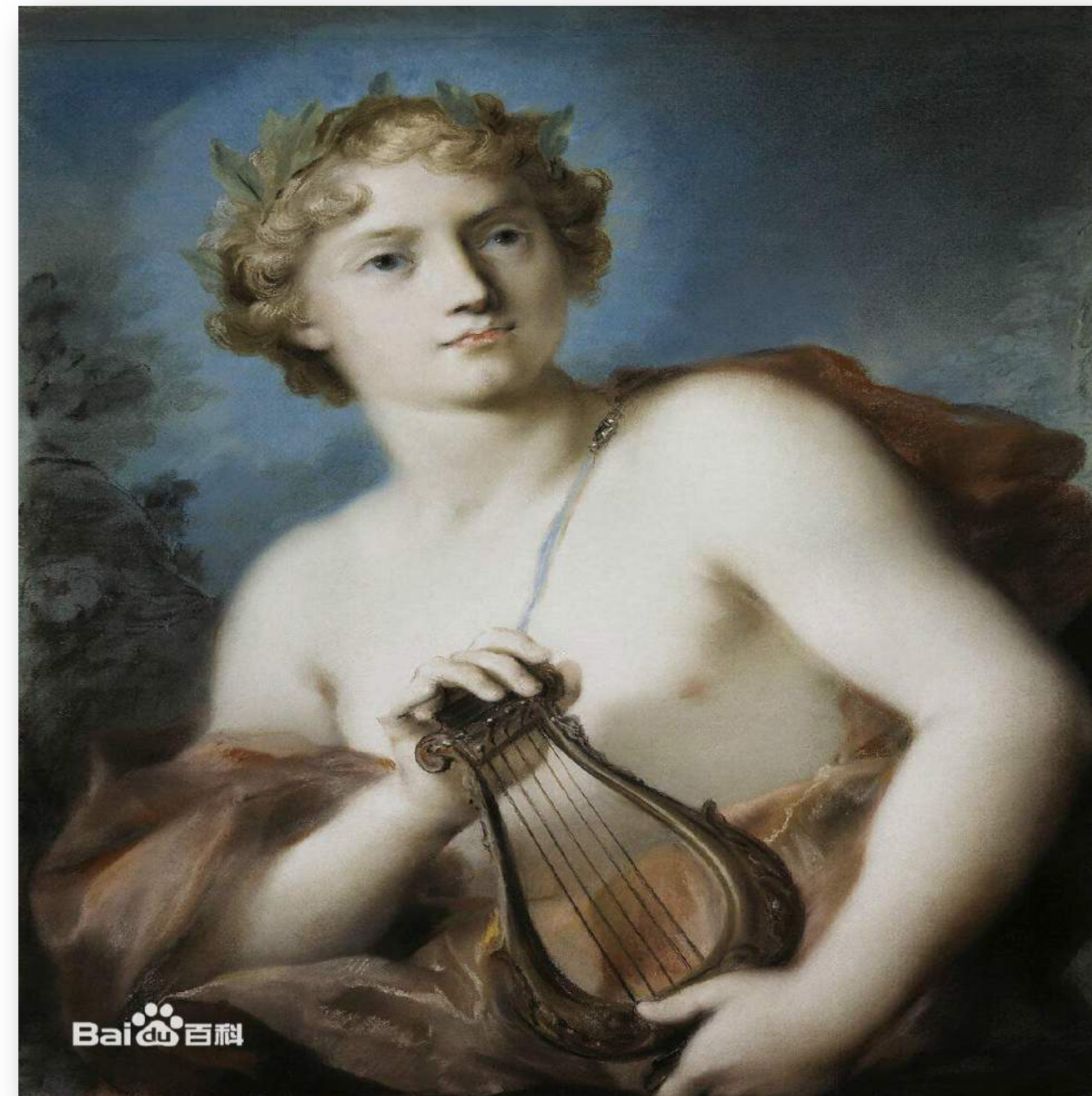
- 没有监控的微服务如同没有任何仪表盘的汽车，随时抛锚。
- 我们采用开源+扩展，研发了基于调用链的监控利器：**Dubbo-Monitor**，实现了微服务调用链的跟踪与应用节点的监控、关键数据的采集和监控。



微服务-统一配置

微服务拆分后有众多节点，且docker化后的配置信息管理繁琐，怎么办？

Apollo



万能的开源社区

额滴神：
保护神、光明之神、预言之神、迁徙和航海者的保护神、医神以及消灾弥难之神

微服务-日志收集



日志怎么看？问题怎么查？



Logstash(Filebeat)：日志收集



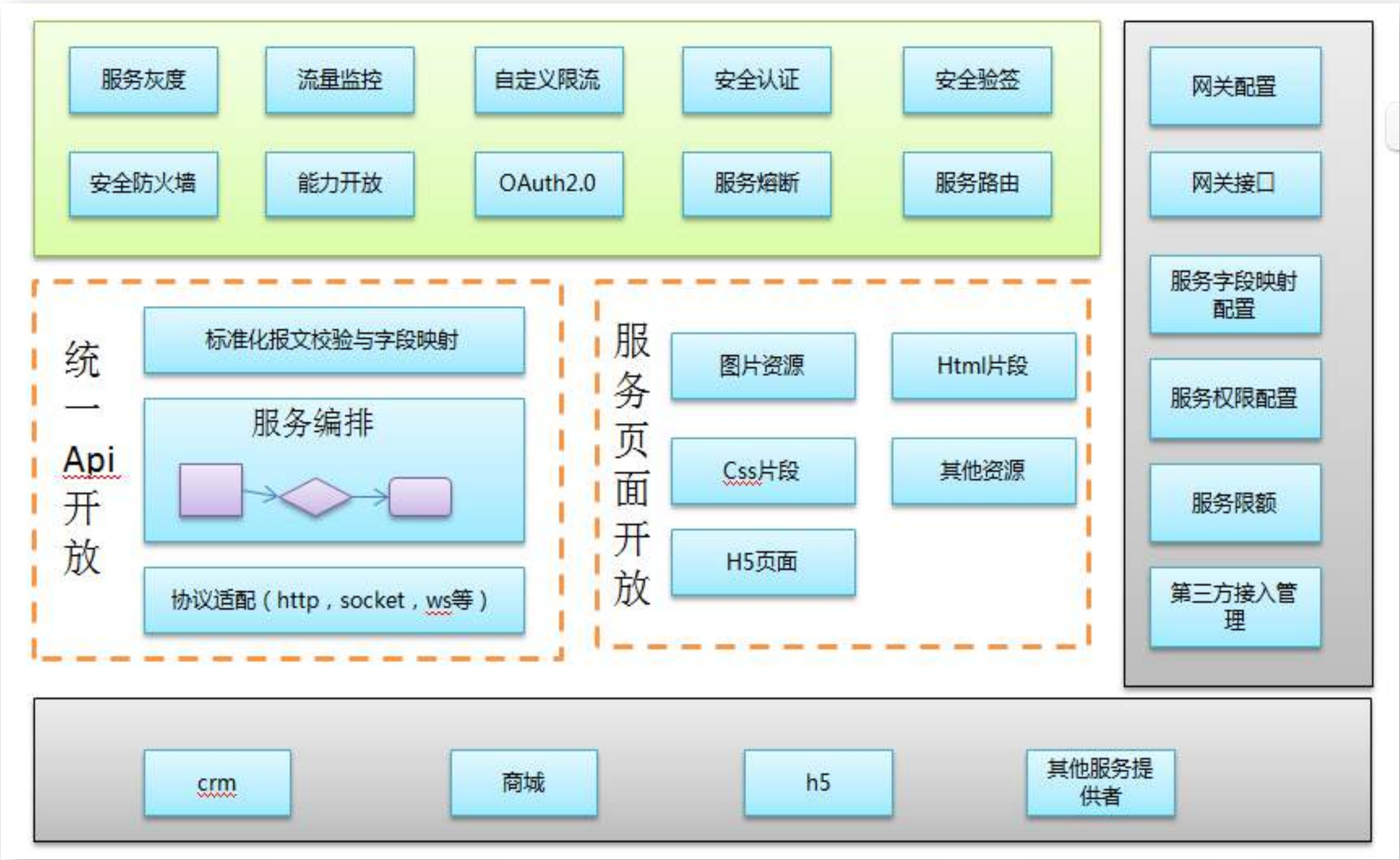
Elasticsearch：日志搜索



Kibana：结果展示

微服务-服务网关

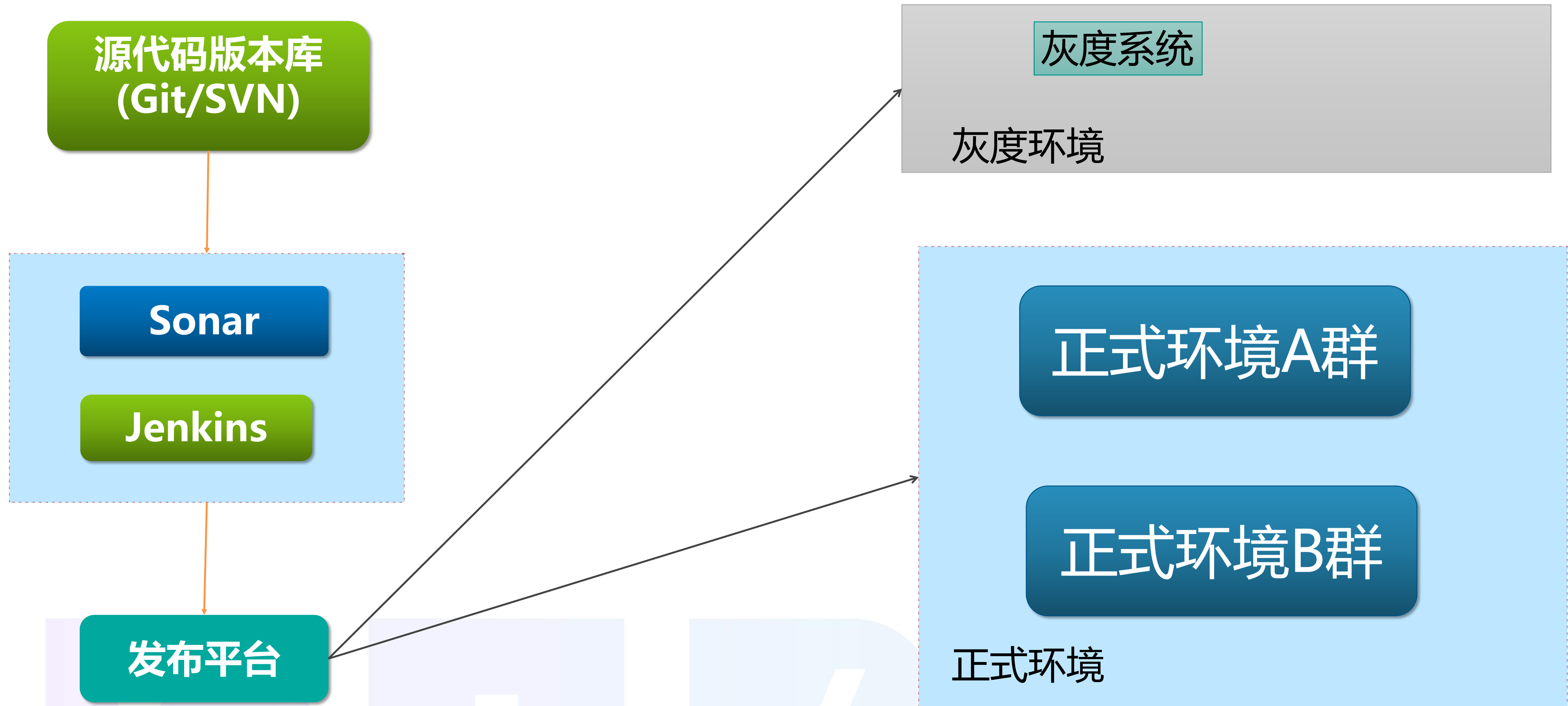
通过网关将内部dubbo服务开放出标准服务，并进行服务路由、流量控制、安全机制、监控、配额等管控。



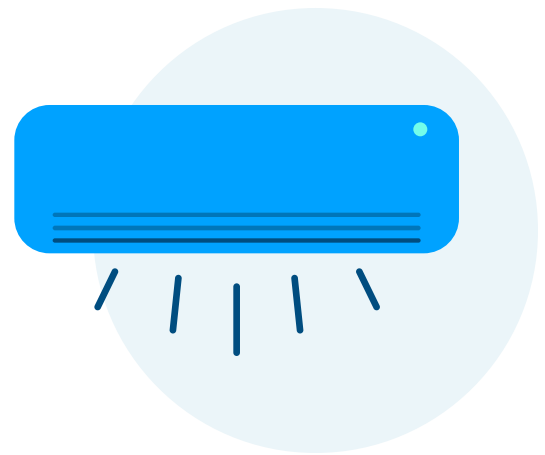
- 1 通过网关对外提供标准化协议
- 2 网关就是dubbo消费端
- 3 流量监控
- 4 安全认证
- 5 流量控制
- 6 调用配额与“软防火墙”
- 7

微服务-自动化部署

通过持续集成工具与灰度发布发布系统 实现众多服务节点的自动部署，一键式发布。

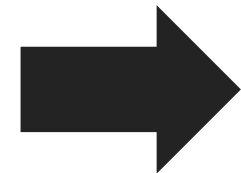


Dubbo使用中遇到的问题-网络篇



Docker化问题

Docker 网络策略导致dubbo服务绑定ip地址为docker内部地址



原因：

由于dubbo获取本地ip时通过 `InetAddress.getLocalHost().getHostAddress()` 。会走 `nameService`通过域名获取ip，所以可以通过配置hosts文件来传递真实ip。

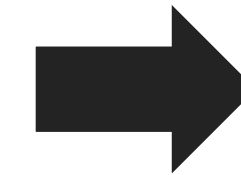
解决：

docker镜像被拉起时，动态设置hosts，将hostname与外部ip地址绑定，问题解决。



IPv6不支持

dubbo 在配置注册服务的时ipv6地址形式时启动报错

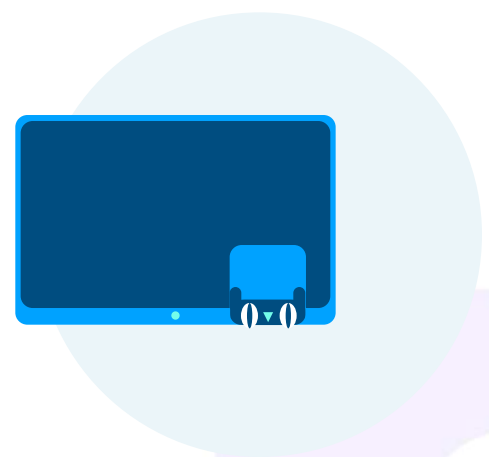


原因：

Dubbo在生成注册服务，链接注册中心时，在获取注册中心配置，解析端口时，按ipv4的方式。解析的即：后面去端口，导致启动报错。

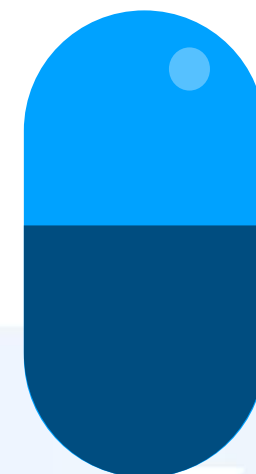
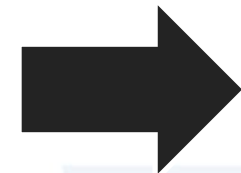
解决：

更新补丁（修改解析处理网络方式）或者升级版本



主机一些参数

系统参数不对导致一些连接不响应超时错误。如：主机时钟一致与Tcp参数：
`tet.ipv4.tcp_tw_recycle`



原因：

时钟不同步，导致连接被认为无响应而被操作系统连接不可用

解决：

调整主机参数

Dubbo使用中遇到问题-安全篇

问题

注册中心安全加固或升级版本
ZK未授权连接获取注册信息

解决办法

- ✓ Zk节点添加ACL：给zk节点上添加ACL安全认证
- ✓ 扩展Dubbo Registry 中的客户端：Curatorx

```
<dubbo:registry address="${dubbo.registry.address}"  
username="${dubbo.username}"  
password="${dubbo.password}" client="${dubbo.client}"/>
```

```
dubbo.username=ecsite  
dubbo.password=5fdd4e2dfb2d64b60c9e687c48f73abb  
dubbo.client=curatorx
```

问题

服务提供者未授权调用

解决办法

- ✓ 扩展IP白名单过滤模块
- ✓ 安全netty transport扩展模块
- ✓ 防火墙策略
- ✓ 用户名，密码 校验Filter验证

Dubbo使用的规范

超时时间合理设置

如果你的服务调用第三方接口，Dubbo服务的超时时间与第三方服务的接口的超时时间：

Dubbo服务超时时间 > 第三方接口超时时间

重试次数

服务消费端重试次数：

服务幂等性设计，或者重试次数为0设置

并发调优设置

如线程池设置，连接数设置等调优

开发规范

尽可能的将业务逻辑后移，下沉至服务端。消费端不要做过多的业务逻辑避免分布式事务。

Dubbo与开发框架集成

非Spring系统集成

需要了解内部机制

通过硬编码的方式创建不同的对象。
发布服务，注册服务，生成代理Invoker等。
ServiceConfig
RegistryConfig
ProviderConfig
ProtocolConfig
.....

使用不多

Spring集成

无缝集成直接使用

SpringContainer 方式

发布状态	Key	Value	备注	最后修改人	最后修改时间	操作
已发布	zookeeper.cluster.nodes	zookeeper://10.252.169.205:2183?backup=10.252.169.206:2183,10.252.169.207:2183		apollo	2019-01-04 22:09:43	
已发布	zookeeper.username	zookeeper		apollo	2019-01-04 22:03:46	
已发布	zookeeper.password	Zookeeper#2018		apollo	2019-01-04 22:03:46	

使用中

SpringBoot集成

@ImportResource 注解集成

```
@SpringBootApplication
@ComponentScan(basePackages = {"com.ai.iisc.es", "com.ai.ecs.retail.order.api", "com.ai.ecs
@ImportResource("classpath:retail-shop-consumer.xml")
@EnableEncryptableProperties
public class RetailShopApplication extends SpringBootServletInitializer {
    private static final Logger log = LoggerFactory.getLogger(RetailShopApplication.class);
    org.apache.kafka.clients.producer.KafkaProducer<String, String> producer;

    <!-- 注册中心服务地址 -->
    <dubbo:registry protocol="zookeeper" address="${zookeeper.cluster.nodes}" username="${zookeeper.username}" password="${zookeeper.pa
    <!-- 客户端超时 -->
    <dubbo:consumer timeout="${dubbo.timeout}" retries="0"/>

    <dubbo:reference id="broadBandBusiService" interface="com.ai.ecs.retail.external.api.province.BroadBandBusiService" check="false" pro
    <dubbo:reference id="channelService" interface="com.ai.ecs.retail.external.api.province.ChannelService" check="false" protocol="dubbo
    <dubbo:reference id="userSMSService" interface="com.ai.ecs.retail.external.api.nation.UserSMSService" check="false" protocol="dubbo"/
    <dubbo:reference id="serverNumberService" interface="com.ai.ecs.retail.external.api.province.ServerNumberService" check="false" proto
    <dubbo:reference id="baseBusiService" interface="com.ai.ecs.retail.external.api.province.BaseBusiService" check="false" protocol="d
    <dubbo:reference id="userInfoService" interface="com.ai.ecs.retail.external.api.nation.UserInfoService" check="false" protocol="dub
    <dubbo:reference id="o2oService" interface="com.ai.ecs.retail.external.api.province.O2OService" check="false" protocol="dubbo"/>
    <dubbo:reference id="validService" interface="com.ai.ecs.retail.external.api.common.ValidService" check="false" protocol="dubbo"/>
    <dubbo:reference id="marketService" interface="com.ai.ecs.retail.external.api.province.MarketService" check="false" protocol="dubbo"/
    <dubbo:reference id="businessLogService" interface="com.ai.ecs.retail.busi.api.log.IBusinessLogService" check="false" protocol="dubbo
    <dubbo:reference id="iQueryService" interface="com.ai.ecs.retail.external.api.common.IQueryService" check="false" protocol="dubbo"/>

    <!-- 服务列表 -->
    <dubbo:reference id="userService" interface="com.ai.ecs.retail.busi.api.system.IUserService" check="false" protocol="dubbo"/>
    <dubbo:reference id="menuService" interface="com.ai.ecs.retail.busi.api.system.IMenuService" check="false" protocol="dubbo"/>
    <dubbo:reference id="roleService" interface="com.ai.ecs.retail.busi.api.system.IRoleService" check="false" protocol="dubbo"/>
    <dubbo:reference id="rolePermissionService" interface="com.ai.ecs.retail.busi.api.system.IRolePermissionService" check="false" protoc
    <dubbo:reference id="plansService" interface="com.ai.ecs.retail.busi.api.plan.IPlansService" check="false" protocol="dubbo"/>
    <dubbo:reference id="appVersionService" interface="com.ai.ecs.retail.busi.api.system.IAppVersionService" check="false" protocol="dubb
```

使用中

Dubbo-移动运营商项目案例

- 1 中国移动多个省级（广东，上海，重庆，湖南，新疆，陕西）
互联网类项目群（电商，电子渠道，O2O，H5，App，网厅等）
- 2 中国移动在线公司某项目
- 3 中国移动南方基地某项目
- 4



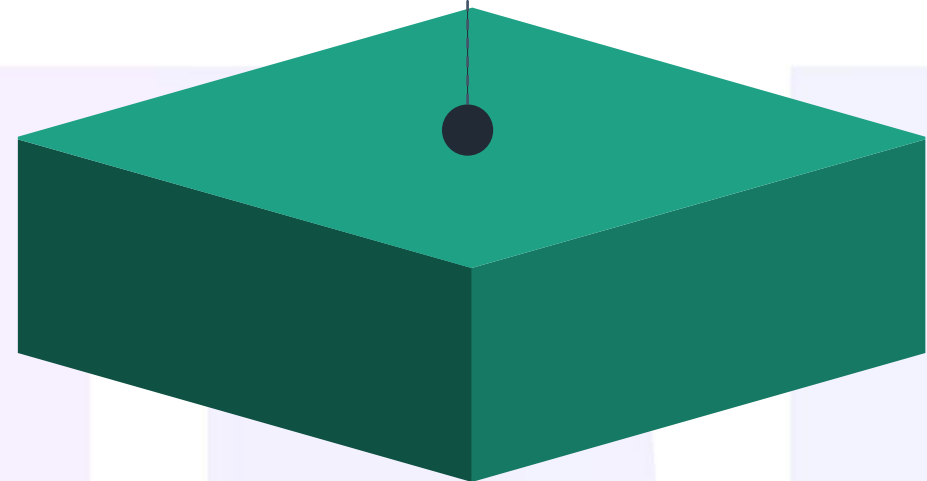
演进方向

- 更好用的框架、更完善的生态、更深度的优化
- 持续关注微服务未来发展：关注社区发展、关注Dubbo生态建设、关注Service Mesh



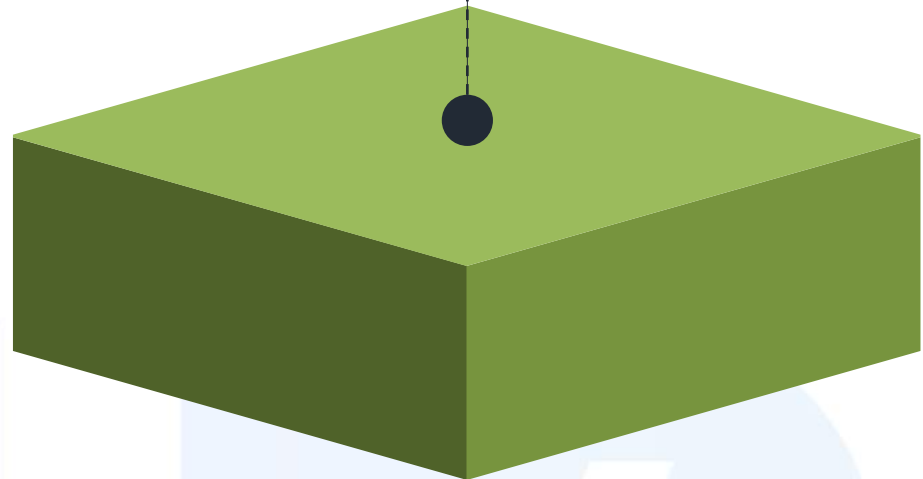
智慧化监控做到有思想

强化监控，做依托机器学习实现智慧化监控，与参数自动调整



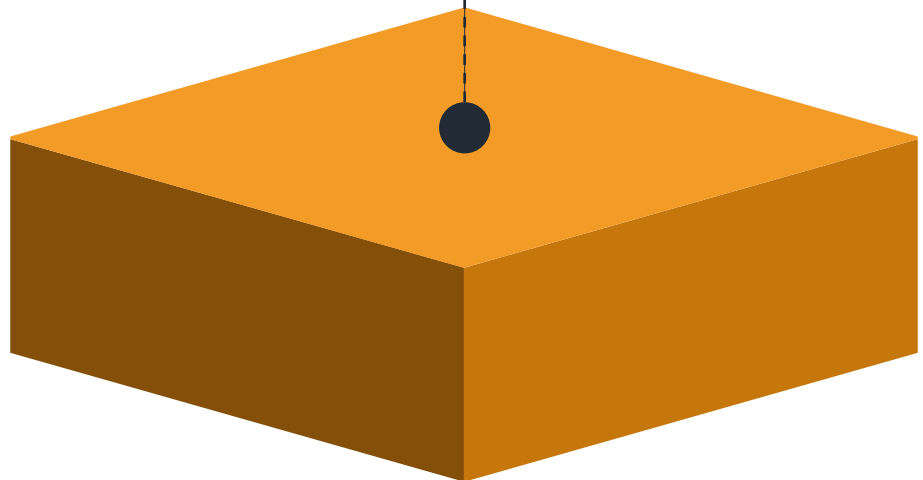
分布式事务更完善

引入分布式事务组件，完善微服务体系



强化服务网关与熔断机制

强化熔断机制，做到自动熔断+手动熔断相结合



总结与感慨

- 1 系统设计要求：服务如何拆分、模型如何设计、是否分布式事务等，是一个综合考虑的事情。
- 2 自动化工具，自动化运维。
- 3 不要试图一步到位，慢慢演进是一个进化的过程。
- 4 适合自己的就是好的。
- 5 开源是一把双刃剑。
- 6 微服务是一种思想，一种风格。
- 7 和一群志同道合的人做对的事情。





谢谢聆听