

用户文档：面向云原生的下一代微服务

集群监测机制，涵盖Kubernetes、Nacos等

参赛队伍：星轨

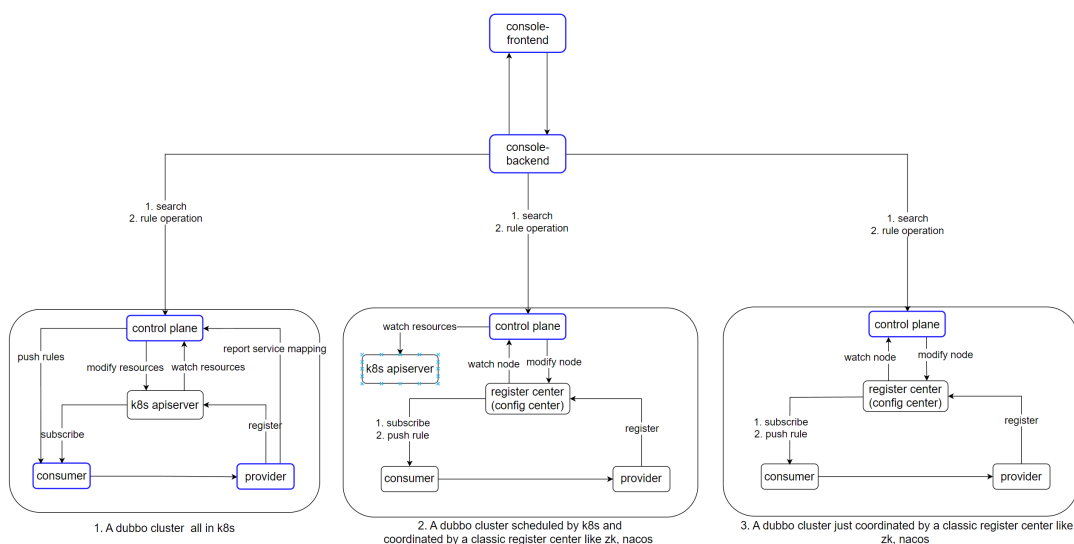
队伍成员：陈才，蔡建悻，程兴源

概述

功能介绍

Dubbo Console定位为Dubbo服务集群的控制台，旨在无侵入的前提下，以统一的视角提供对服务集群的观测和流量治理，为VM，K8s环境下的运行的Dubbo服务集群提供一个用户友好，功能完备的治理平台。

如下图所示，主流的Dubbo服务集群的运行形态可以分为三种：



1. 第一种形态即“全托管k8s”形态，应用的服务发现与部署调度都依赖于k8s和 control plane，不再依赖Zookeeper，Nacos等中间件。
2. 第二种形态为“半托管k8s”形态，应用的服务发现依赖于传统注册中心，而部署与调度依赖于k8s。

3. 第三种形态为传统形态，应用的服务发现依赖于传统注册中心，在非k8s平台上部署

得益于control plane屏蔽掉了底层的实现，Console对于k8s环境和非k8s环境都做到了兼容。在此基础上，Console提供了用户友好的界面，集成了可观测领域的主流组件，便于用户查看集群数据，排查问题，管控集群流量。

术语解释

在更详细地介绍如何使用Console之前，需要对后文中出现的术语提前解释：（部分术语解释摘自 <https://opensergo.io/zh-cn/docs/what-is-opensergo/concepts/>）

1. 应用 (Application)：用于表示一个微服务，通常以独立形态部署，可能提供一个或多个服务供提供者调用。
2. 服务 (Service)：带有明确业务语义的一组接口的集合，供消费者进行调用，通常包含一个或多个接口。
3. 接口 (Interface)：用于表示一个确定的接口，通常有明确的接口描述，调用的参数定义，及返回值定义。
4. 实例 (Instance)：应用的一个运行单位，可以是一个pod，也可以是一台虚拟机或容器，也可以是一个进程
4. 工作负载 (Workload)：表示某个部署集合，如 Kubernetes Deployment, Statefulset 或者一组 pod，或某个业务进程，甚至是一组 DB 实例
5. 标签 (Tag)：一个应用可以有多个节点组成，同一个应用下的节点可以按照功能划分成不同分组，通过标签来筛选出满足一定条件的节点集合。
6. 地址子集(Subsets)：按照标签分组形成的集合，通常作为路由分发的目的地
7. 匹配条件 (Match)：一组逻辑表达式，可分为key, relation, value, 指定输入，返回的是是否满足匹配的结果
8. 作用范围(Scope)：路由的第一个阶段，规则生效的范围，一般包括若干匹配条件，
9. 请求匹配(Request Match)：路由的第二个阶段，一般包括若干匹配条件，满足条件的才会进入路由分发
10. 路由分发(Route distribution)：路由第三个阶段，满足匹配条件的请求按照一定策略分发到不同的地址子集

快速入门

启动

- 默认您已经具备以下集群环境：
 - CLASSIC CLUSTER：传统虚拟机环境，以 nacos、zookeeper 等为注册中心的 Dubbo 集群
 - ALL IN K8S：k8s 集群环境
 - MIX：融合传统集群环境和 k8s 环境

- 安装Dubbo Console Plane

- CLASSIC CLUSTER:

```
1 # 本地启动一个zookeeper, 127.0.0.1:2181 即可
2 make run/dubbo-cp
```

- ALL IN K8S

```
1 kubectl apply -f test/control-plane/crds
2 make run/dubbo-cp
```

- MIX

```
1 #本地启动一个zookeeper, 127.0.0.1:2181 即可
2 kubectl apply -f test/control-plane/crds
3 make run/dubbo-cp
```

- 访问 Dubbo Console 前端
 - <http://localhost:38080/admin>

功能说明

集群概览

在首页可以看到当前集群环境中比较重要的指标和元数据信息：

1. 部署资源数量
2. 集群详情和注册中心详情

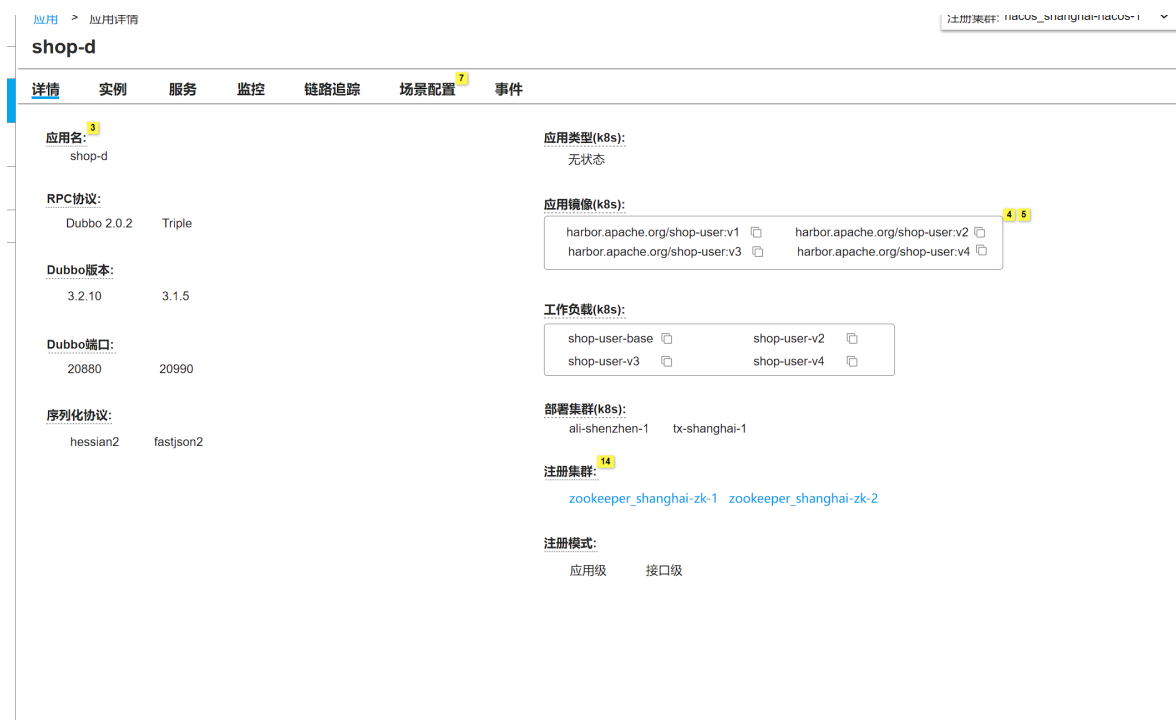


资源详情

资源详情中包括领域模型中应用、实例和服务三个对象，展示不同维度的运行时数据，监控和链路追踪数据，展现

应用：

应用包括应用的查找，应用的详细信息查看，所属实例的查看和流转，提供/依赖的服务信息的查看和流转。在此基础上，在监控这一栏还嵌入grafan面板来查看应用维度的监控数据以及链路追踪数据。此外，还有应用的场景化配置以及事件的时间线展示。



实例：

实例包括实例的查找，实例的详细信息查看。同时也和应用一样，提供grafana面板来查看实例维度的监控数据以及链路追踪数据，实例的事件信息等。

实例

> 实例详情

注册集群: nacos_shanghai-nacos-1

shop-user-base-7ad5gh

详情

监控

链路追踪

场景配置6

事件

实例名称:

shop-user-base-7ad5gh

部署状态:

Running

注册状态:

Registered

实例IP:

192.168.0.1

Dubbo端口:

20880

实例IP:

192.168.0.1

所属应用:

shop-d

所属工作负载(k8s):

shop-user-base

实例标签:

app=shop-user

version=v1

region=shenzhen

创建时间(k8s):

2023/12/19 22:09:34

启动时间(k8s):

2023/12/19 22:12:34

注册时间:

2023/12/19 22:16:56

部署集群:

tx-shanghai-1

注册集群:

zookeeper_shanghai-zk-2 zookeeper_shanghai-zk-1

节点:

30.33.0.1

镜像(k8s):

apache/org.apache.dubbo.samples.shop-user:v1

健康检查(k8s):

启动探针 (StartupProbe) : 关闭

就绪探针 (ReadinessProbe) : 开启

存活探针 (LivenessProbe) : 开启

类型: Http

端口: 22222

服务：

服务包括服务的查找，服务的详细信息查看，服务调试，服务分布（上下游）。同时和应用一样，提供grafana面板来查看服务维度的监控和链路追踪数据，提供一些场景化的配置来应对常用的流量治理需求。

生产者详情

shop-user

超时时间: 3000 ms

重试次数: 2次

是否过时: false

RPC协议: triple

延迟注册时间: 3000 ms

消费者详情

shop-fontend

超时时间: 3000 ms

重试次数: 2次

是否过时: false

RPC协议: triple

延迟注册时间: 3000 ms

流量管控

Dubbo Admin的原有的能力基础上，我们将流量规则的下发流程进一步可视化，只要是Dubbo原生支持的规则，用户就能按照提示填写相应规则的表单即可完成下发过程，同时我们也保留了Yaml编辑，并提升了Yaml编辑的体验。

新增条件路由

选择视图: 表单视图 Yaml视图

基本信息

规则粒度: 应用

作用对象:

容错保护: 关闭 开启

立即启用: 否 是

运行时生效: 关闭 开启

路由列表

路由【1】 当请求方法等于login，且方法参数第1个等于dubbo，就导向标签version=v1的地址子集

请求匹配

匹配条件类型

路由分发

匹配条件类型

增加路由

确认

取消

流量管控 > [标签路由](#) > 修改标签路由

修改shop-user.tag-router

选择视图：[表单视图](#) Yami视图

字段说明 <<

基础信息

规则粒度：应用

作用对象：shop-user

容错保护：⁸关闭 ☒ 开启

立即启用：⁸否 ☒ 是

运行时生效：⁷关闭 ☒ 开启

标签列表

标签【1】 将满足version=v1的实例打上标签dubbo.tag=gray⁹

标签名：

固定范围：

固定条件类型 ☒ labels ☐ addresses

labels

键	关系	值	操作
<input type="text" value="label key"/>	exact <input type="text" value="v"/>	<input type="text" value="label value"/>	<div>- +</div>

增加标签

确认

取消

流量管控 > [动态配置](#) > 表单视图

org.apache.dubbo.samples.service.UserService::configurators

操作

[表单视图](#) [YAML视图](#) [事件](#)

版本记录 <<

基础信息

规则名：org.apache.dubbo.samples.UserService::configurators

规则粒度：服务

作用对象：org.apache.dubbo.samples.UserService

创建时间：2024/2/19/ 11:38:21

启用状态：启用

配置列表

配置【1】 对于org.apache.dubbo.samples.UserService的提供者，将满足地址address等于10.255.10.11的实例，配置重试次数retrie¹等于2

启用状态：启用

作用端：provider

作用范围：

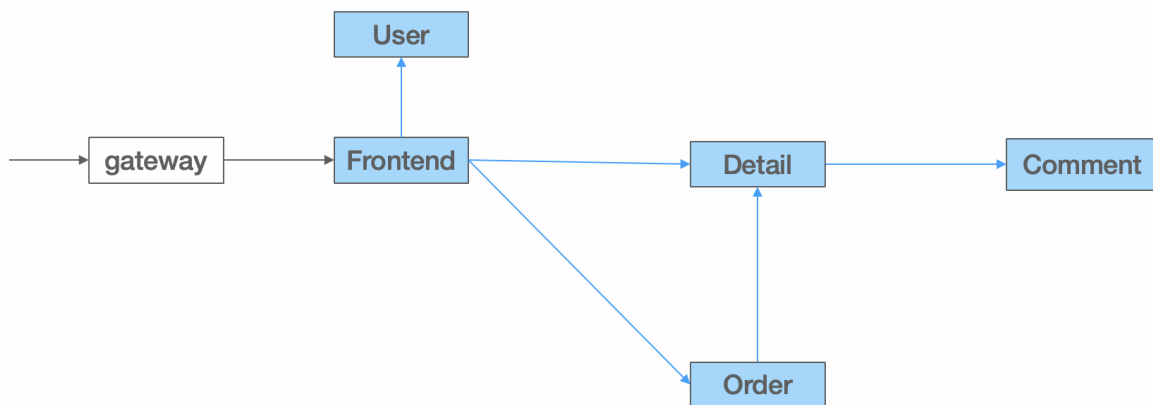
配置项：

示例

示例部署

本示例提供一个简单的线上商城系统来演示资源信息查看，监控查看，链路追踪查看，流量规则下发等功能。

线上商城的架构图如下：



系统由 5 个微服务应用组成：

- **Frontend 商城主页**，作为与用户交互的 web 界面，通过调用 **User**、**Detail**、**Order** 等提供用户登录、商品展示和订单管理等服务。
- **User 用户服务**，负责用户数据管理、身份校验等。
- **Order 订单服务**，提供订订单创建、订单查询等服务，依赖 **Detail** 服务校验商品库存等信息。
- **Detail 商品详情服务**，展示商品详情信息，调用 **Comment** 服务展示用户对商品的评论记录。
- **Comment 评论服务**，管理用户对商品的评论数据。

为方便起见，我们将微服务环境部署在k8s集群，执行以下命令即可完成项目部署：

```
1 kubectl apply -f
  https://raw.githubusercontent.com/robocanic/dubbo-
  samples/shop-metric-trace/10-task/dubbo-samples-
  shop/deploy/All.yml
```

查看资源信息

以应用为例，点击左侧菜单栏“应用”，在列表中选中shop-user，我们即可看到应用运行时信息，包括所属该应用的实例信息，以及该应用提供和依赖的服务信息。

应用名:shop-user

应用类型:无状态

序列化协议:fastjson2

RPC 协议:dubbo 2.0.2

Dubbo 端口:20880

Dubbo 版本:Dubbo 3.2.10

应用镜像:

harbor.apache.org/dubbo-samples-shop-user.v1.0

harbor.apache.org/dubbo-samples-shop-user.v1.1

harbor.apache.org/dubbo-samples-shop-user.v1.2

工作负载:

dubbo-samples-shop-user-base

dubbo-samples-shop-user-gray

dubbo-samples-shop-user-gray

dubbo-samples-shop-user-gray

部署集群:ali-shanghai-1tx-shanghai-2

注册集群:nacos-cluster-1nacos-cluster-2

注册模式:应用级接口级

详情实例服务监控链路追踪配置事件

提供者43

消费者4

cpu56c

memory108.2GB

IP地址

请输入

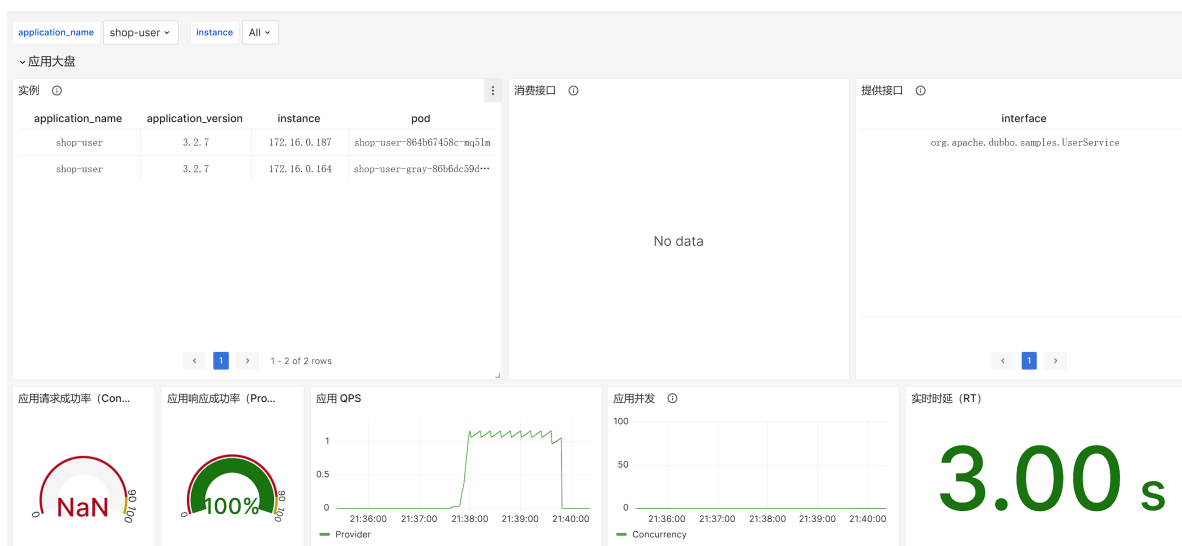
点击升序

IP	实例名称	部署状态	部署集群	注册状态	注册集群	CPU	标签
121.90.211.162	shop-user	Crashing	tx-shanghai-1	Registered	ali-hangzhou-1 ali-hangzhou-2	1.2c	beijingv1
121.90.211.162	shop-user	Terminating	tx-shanghai-1	Registered	ali-hangzhou-1 ali-hangzhou-2	1.2c	beijingv1
121.90.211.162	shop-user	Terminating	tx-shanghai-1	Registered	ali-hangzhou-1 ali-hangzhou-2	1.2c	beijingv1
121.90.211.162	shop-user	Pending	tx-shanghai-1	Registered	ali-hangzhou-1 ali-hangzhou-2	1.2c	beijingv1
121.90.211.162	shop-user	Pending	tx-shanghai-1	Registered	ali-hangzhou-1 ali-hangzhou-2	1.2c	beijingv1
					ali-hangzhou-1		

提供者 + 123		消费者 + 105				
提供者		消费者	服务名: <input type="text" value="请输入"/>	<input type="button" value="搜索"/>		<input type="button" value="菜单"/>
序号	服务	接口数	近 1min QPS	近 1min RT	近 1min 请求总量	
1	org.apache.dubbo.samples.UserService	4	6	194ms	200	
2	org.apache.dubbo.samples.OrderService	12	13	189ms	164	
3	org.apache.dubbo.samples.DetailService	14	0.5	268ms	1324	
4	org.apache.dubbo.samples.PayService	8	9	346ms	189	
5	org.apache.dubbo.samples.CommentService	9	8	936ms	200	

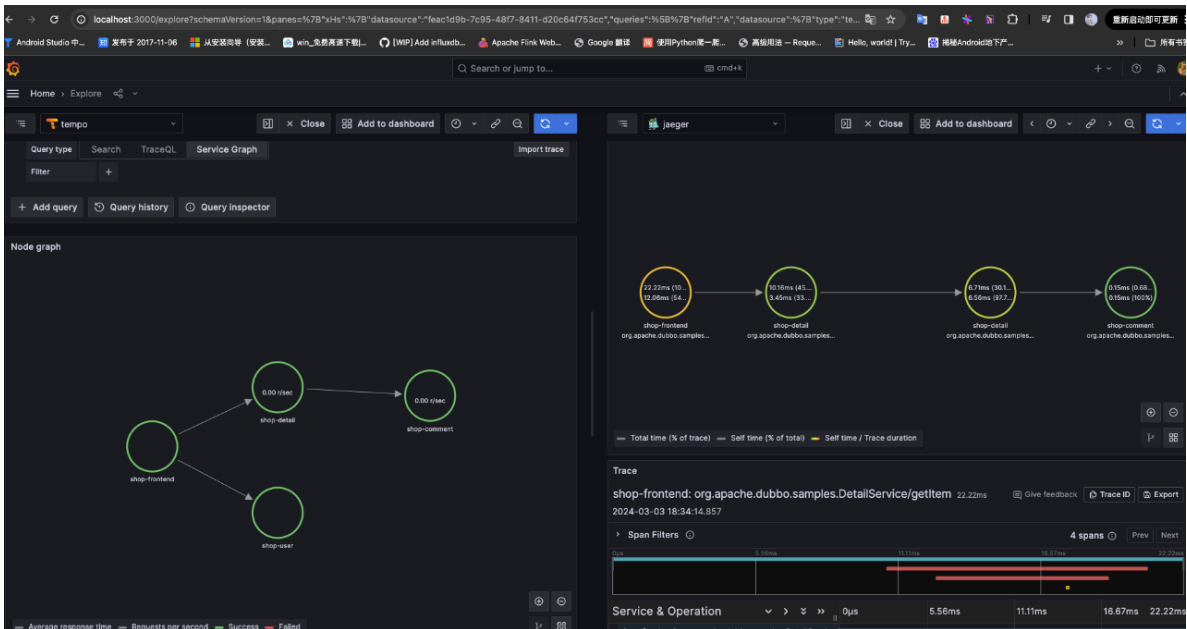
查看监控

点击监控页面，可以看到该应用的QPS，RT，RequestTotal等监控数据：



查看链路追踪

点击链路追踪，可以看到该应用的链路追踪数据：



下发流量规则

以条件路由为例，点击左侧菜单栏进入条件路由页面，选择新增条件路由，进入到新增页面。我们来下发一条这样的规则，对于 `org.apache.dubbo.samples.UserService`，当请求方法等于 `login`，且该方法的第一个参数等于 `"dubbo"`，就导向带有标签 `version=v1` 的地址子集。

流量管控 > 条件路由 > 新增条件路由

新增条件路由

选择视图: 表单视图 Yaml视图

路由列表

路由【1】 1 当请求方法等于login，且方法参数第1个等于dubbo，就导向标签version=v1的地址子集

请求匹配

匹配条件类型? method arguments

method = login

arguments

参数索引	关系	值	操作
1	=	dubbo	- +

• 路由分发

匹配条件类型? 其他

其他

键	关系	值	操作
key	=	value	- +

增加路由

确认 取消

如上图所示，在填写完请求匹配与路由分发两部分后，点击确认就完成了路由规则的下发。现在看shop-user

的v2版本实例，可以看到所有请求都被导向该地址子集。此外，console还提供了Yaml视图，便于高阶用户使用。

新增条件路由

选择视图： 表单视图 Yami视图

key: org.apache.dubbo.demo.DemoService
scope: service
force: true
runtime: false
enabled: true
priority: 1
conditions:
- method=sayHello =>
- method=routeMethod1 => address=30.5.120.16:20880

yaml

字段说明 >>

- key: 作用对象
可能的值: Dubbo应用名或者服务

- scope: 规则粒度
可能的值: application, service

- force: 容错保护
可能的值: true, false
描述: 如果为true, 则路由筛选后若没有可用的地址则会直接报异常; 如果为false, 则会从可用地址中选择完成RPC调用

- runtime: 运行时生效
可能的值: true, false
描述: 如果为true, 则该rule下的所有路由将会实时生效; 若为false, 则只有在启动时才会生效

-