云原生产业大会 2021 CLOUD NATIVE INDUSTRY CONFERENCE
2021/05/26 × 中国 / 北京
原生蓄力 云领未来

新华三云原生微服务Runtime

李科伟 新华三云计算产品经理

CAICT 中国信通院

目录

微服务架构及其特点

服务注册与发现

微服务间如何调用

配置管理

配置的集中管理和分发

API网关

外部应用如何访问内部服务

流量安全

服务之间如何安全访问

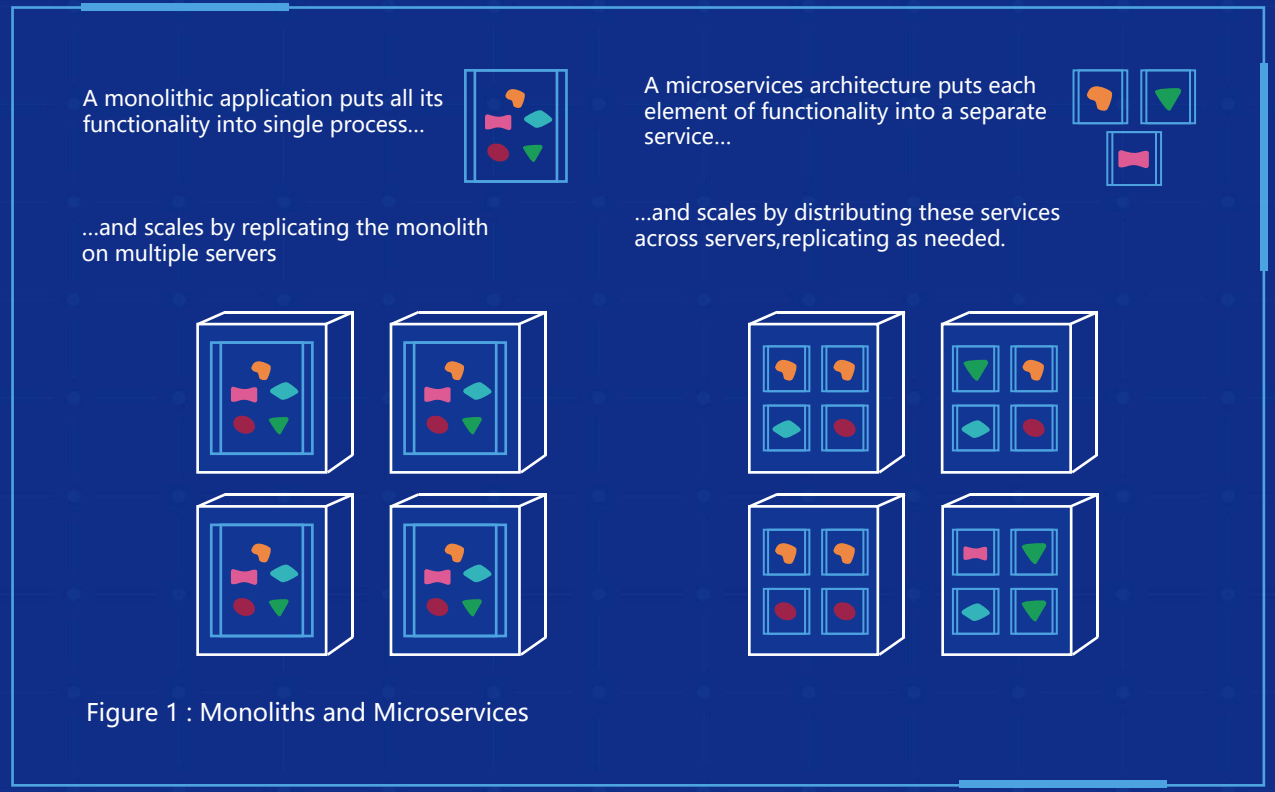


Figure 1 : Monoliths and Microservices

服务管理

部署、伸缩、升级、重启

可观测性

日志、监控、调用链

流量管理

负载均衡、路由、限流

容错

熔断、降级

微服务典型解决方案

Spring Cloud

Kubernetes

Istio

Spring Cloud

服务注册与发现

微服务间如何调用

配置管理

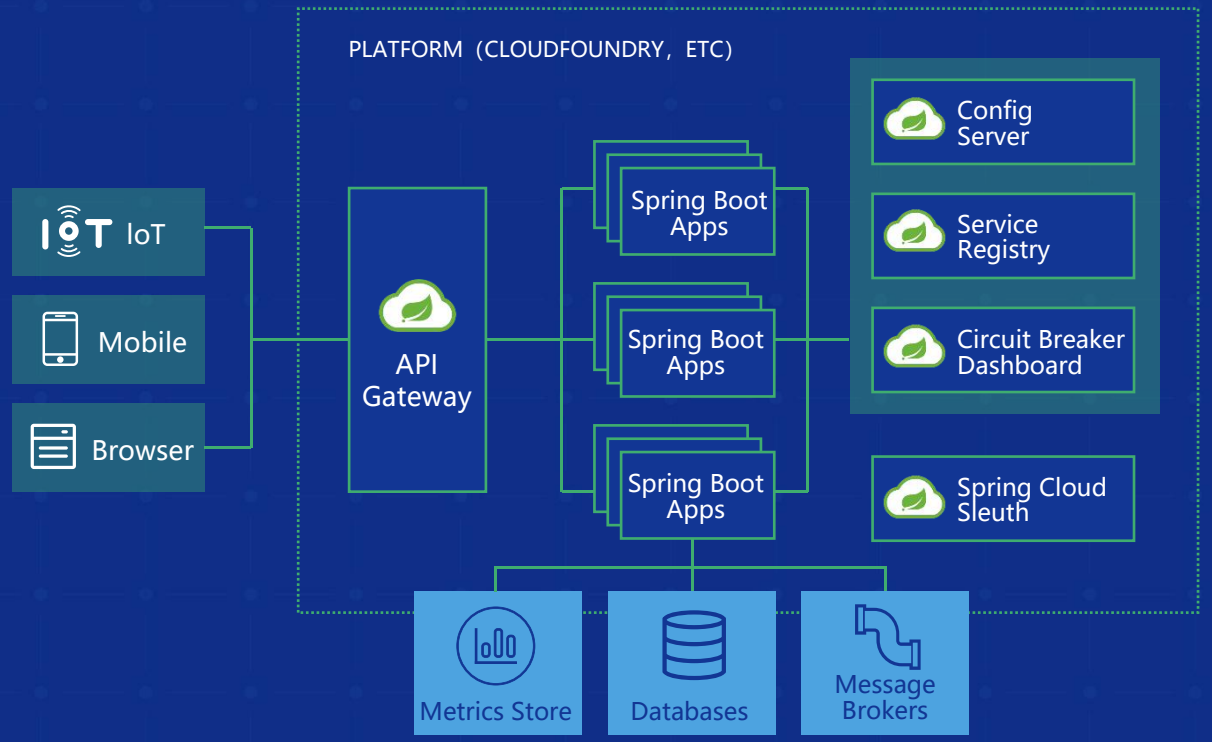
配置的集中管理和分发

API网关

外部应用如何访问内部服务

流量安全

服务之间如何安全访问



服务管理

部署、伸缩、升级、重启

可观测性

日志、监控、调用链

流量管理

负载均衡、路由、限流

容错

熔断、降级

Kubernetes

Kubernetes 是用于自动部署，扩展和管理容器化应用程序的开源系统。

服务注册与发现

微服务间如何调用

配置管理

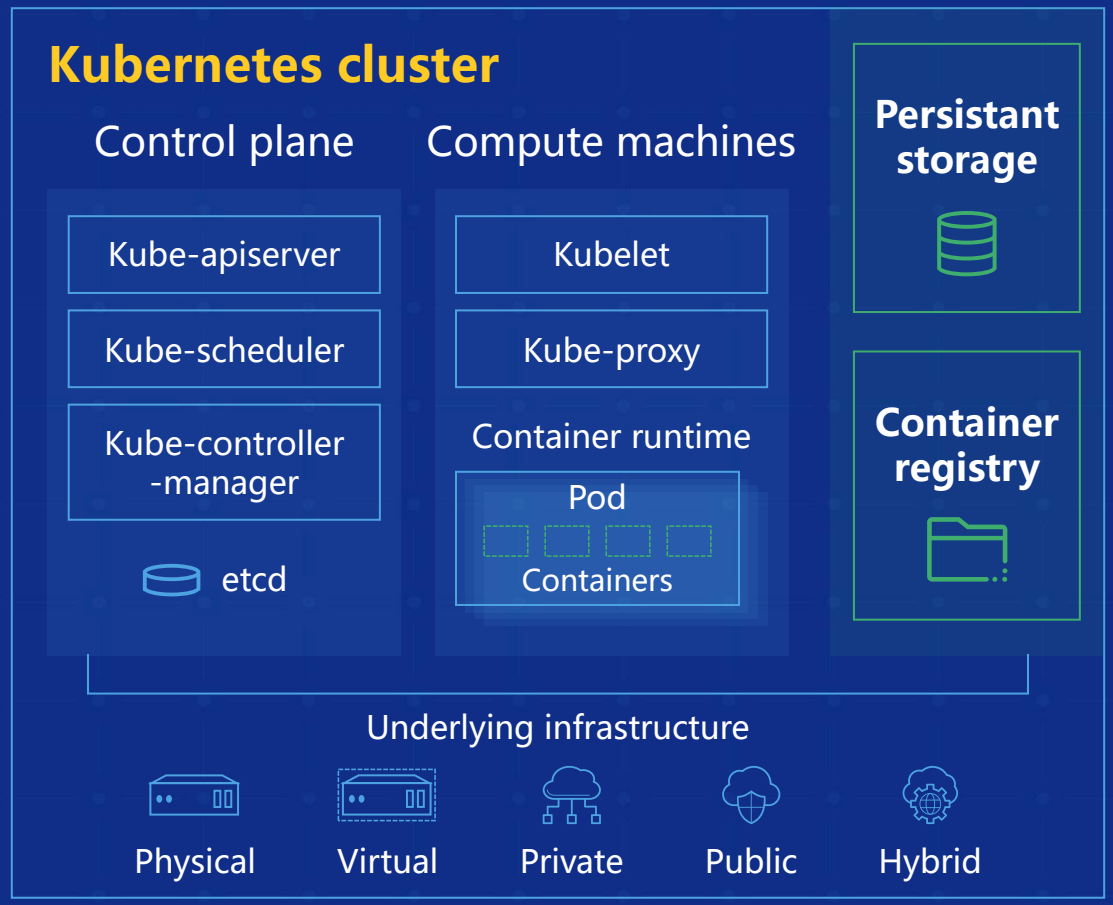
配置的集中管理和分发

API网关

外部应用如何访问内部服务

流量安全

服务之间如何安全访问



服务管理

部署、伸缩、升级、重启

可观测性

日志、监控、调用链

流量管理

负载均衡、路由、限流

容错

熔断、降级

istio

服务注册与发现

微服务间如何调用

配置管理

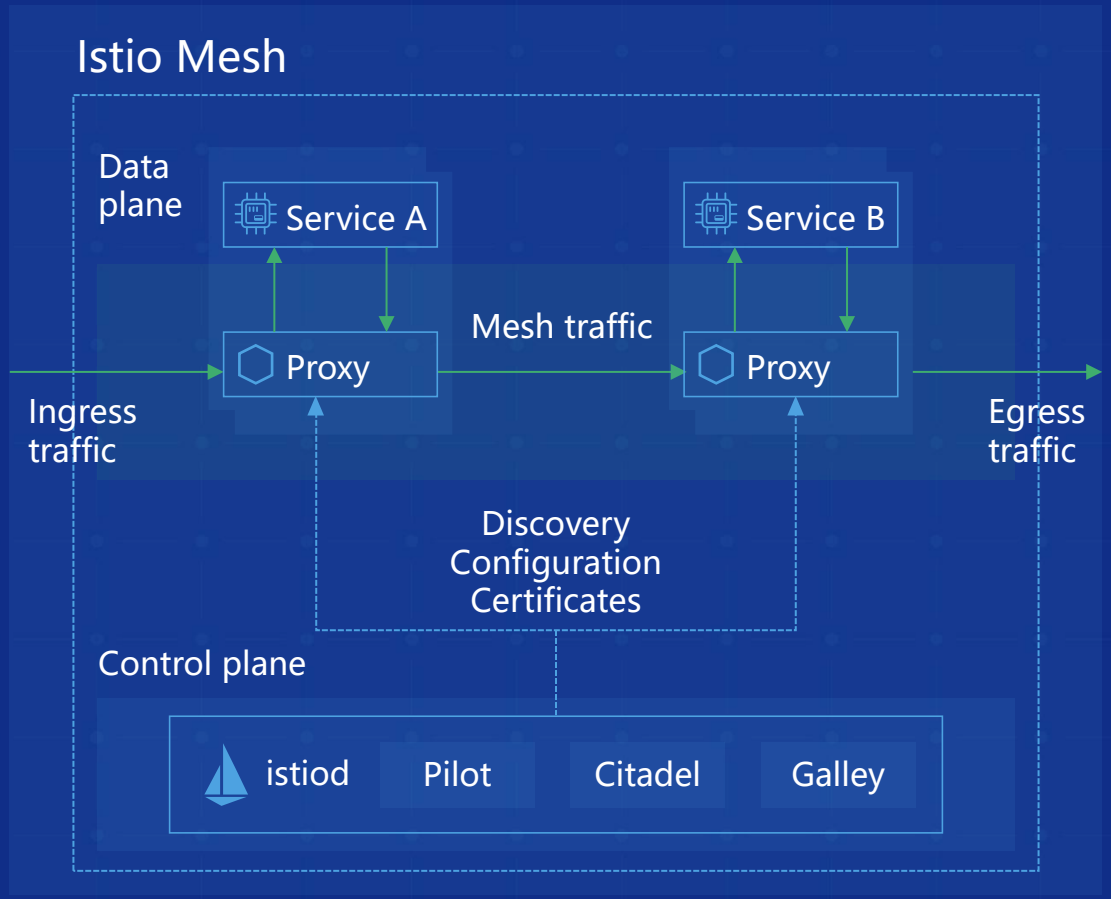
配置的集中管理和分发

API网关

外部应用如何访问内部服务

流量安全

服务之间如何安全访问



服务管理

部署、伸缩、升级、重启

可观测性

日志、监控、调用链

流量管理

负载均衡、路由、限流

容错

熔断、降级

微服务发展历程

微服务架构	描述	代表框架/工具
第一代	应用自己完成业务逻辑与服务通信、容错等问题	无
第二代	通过sdk的方式将服务注册发现、熔断等非业务能力封装并提供给应用使用	Spring Cloud
第三代	使用servicemesh技术进一步将非业务能力放到进程外	Istio
第四代	微服务变为微逻辑(microLogic)，使用多运行时(Multi-Runtime)提供微服务所需的其他能力	Dapr

能力地图


配置管理
配置的集中管理和分发
服务管理
部署、伸缩、升级、重启
服务注册与发现
微服务间如何调用
可观测性
日志、监控、调用链
API网关
外部应用如何访问内部服务
流量管理
负载均衡、路由、限流
流量安全
服务之间如何安全访问
容错
熔断、降级
Spring Cloud
Kubernetes
Istio

可观测性

云原生计算基金会（CNCF）：

云原生技术有利于各组织在公有云、私有云和混合云等新型动态环境中，构建和运行可弹性扩展的应用。
云原生的代表技术包括容器、服务网格、微服务、不可变基础设施和声明式API。
这些技术能够构建容错性好、易于管理和便于观察的松耦合系统。结合可靠的自动化手段，云原生技术使工程师能够轻松地对系统作出频繁和可预测的重大变更。

比较

Spring Cloud
（传统）

VS

Kubernetes + Istio
（云原生）

当一个spring cloud应用部署到kubernetes上的时候会发生什么?

沿用Spring Cloud架构:

1. 容器化注册中心
2. 容器化配置中心
3. 容器化服务网关
4. 客户端负载均衡

应用改造:

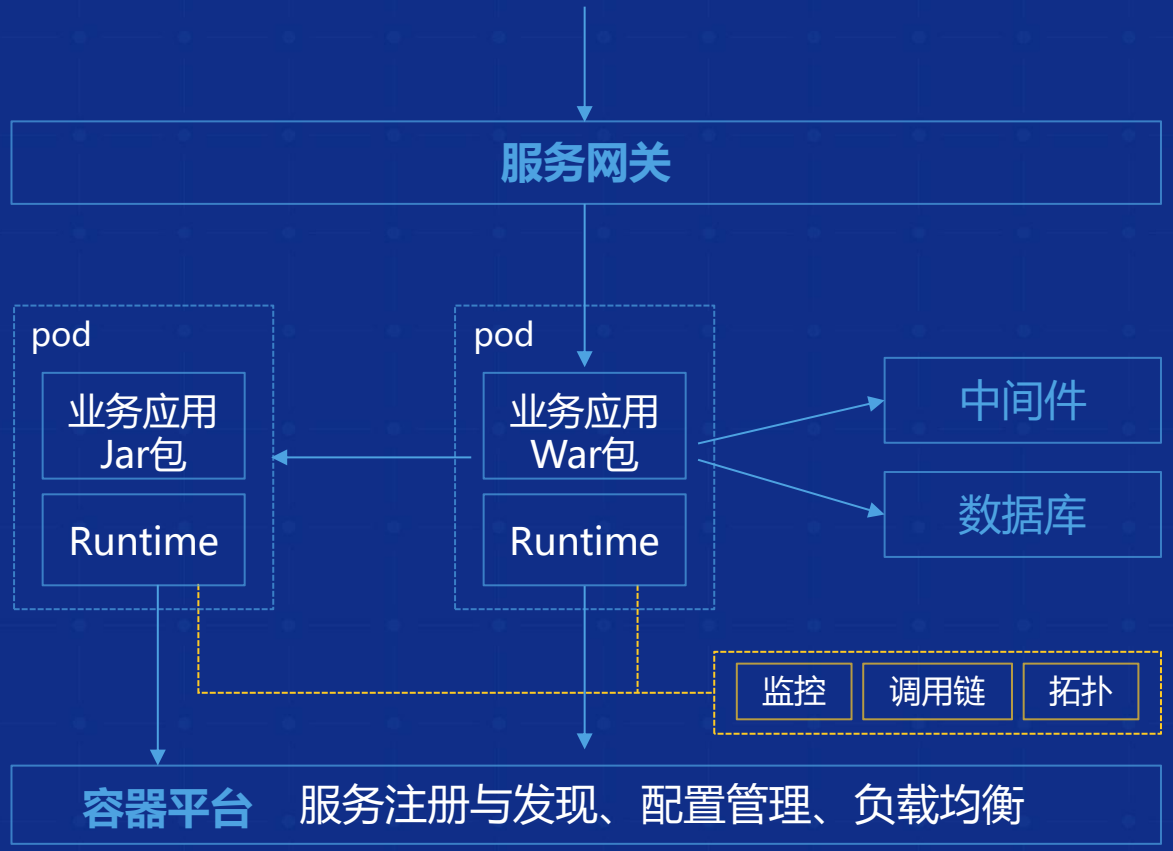
1. 去除原注册中心
2. 去除原配置中心
3. 使用ingress作为网关
4. 使用kubernetes Service

有没有一种方式，针对Spring Cloud应用，不改动代码同时能够使用Kubernetes?

```yaml
spring:
  application:
    name: consumer-service
#eureka:
#  client:
#    service-url:
#      #示例：defaultZone: http://127.0.0.1:8761/eureka
#      defaultZone: ${EUREKA_URL:http://10.125.31.185:31122/eureka}
eureka:
  client:
    enabled: false
ribbon:
  eureka:
    enabled: false
feign:
  hystrix:
    enabled: false
provider:
  ribbon:
    listOfServers: provider-service:10011
```

- 根据代码来修改配置文件
- 监控？调用链?

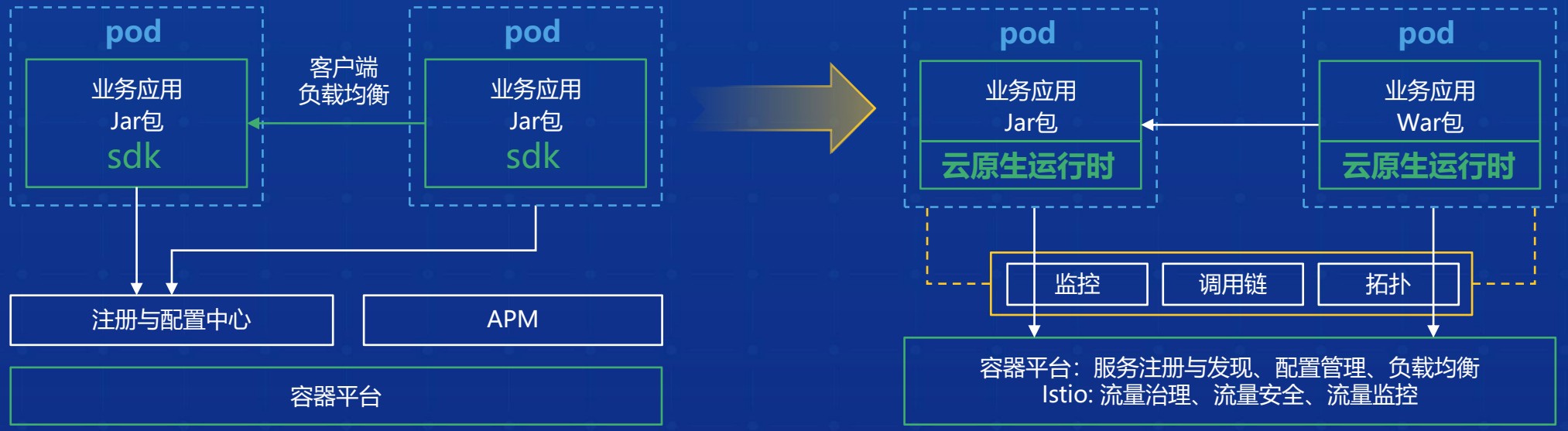
H3C 云原生微服务Runtime



- 无侵入式的调用链、拓扑和监控
- 屏蔽原spring cloud能力
- 使用Kubernetes服务注册与发现
- 可使用istio做服务治理

转变

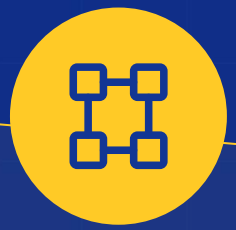
相比Spring Cloud的实现方式，云原生微服务Runtime是更“云原生”的方式，直接使用了云容器平台提供的服务注册与发现、配置管理、负载均衡能力。同时能够使用istio来完成流量的管理。



场景

场景一

Spring Cloud遗留系统迁移到k8s



场景二

新开发的Spring Cloud应用，spring cloud开发，k8s原生方式部署

场景三

k8s原生方式开发，使用runtime完成链路、监控的接入

实现

+

ByteBuddy

+

Spring Cloud

Runtime小结


小王，传统应用怎么上云呢?

什么样的传统应用呢?



一个Spring Cloud应用

可以改造一下代码，变成云原生架构



应用改造太难了，我们也没有经验

可以不改造，使用容器部署就行，只是不是那么“云原生”



有没有又不改造，又比较“云原生”？

如果想不改造又想用云原生架构，可以使用“云原生运行时”


什么是“云原生架构”？

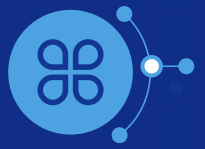
完全根据云原生能力设计的应用架构，充分使用云平台(k8s)提供的注册与发现、配置管理、负载均衡、服务治理、链路追踪等能力的架构。

什么是“云原生微服务运行时”？

一个应用运行的环境，其针对Spring Cloud类型应用，屏蔽了原来的注册与发现、配置管理、负载均衡等能力，并使用云平台(k8s)原生能力进行替代。

什么是“零修改”？

使用“云原生运行时”，应用无需修改自身代码，仅需在部署时遵循一些规范即可。



适用于什么类型应用?

适用于采用标准方式编写的Spring Cloud应用，非标准方式编写的应用需要进行评估。

云原生 产业大会
2021 CLOUD NATIVE INDUSTRY CONFERENCE
2021/05/26 × 中国 / 北京
原生蓄力 云领未来

THANKS !

CAICT 中国信通院