从Servicemesh到云原生

Dapr v1.0展望

作者：敖小剑 @ 阿里云

内容

1 回顾：Servicemesh原理和方向

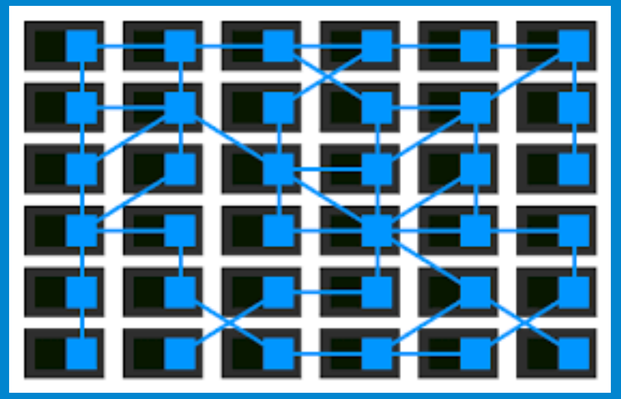
2 演进：云原生分布式应用运行时

3 介绍：分布式应用运行时Dapr

4 展望：应用和中间件的未来形态

1 回顾：Servicemesh原理和方向

Service Mesh的定义



“Service Mesh是一个基础设施层，用于处理服务间通讯。现代云原生应用有着复杂的服务拓扑，服务网格负责在这些拓扑中实现请求的可靠传递。”

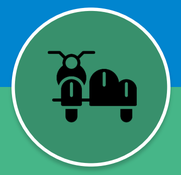
“在实践中，服务网格通常实现为一组轻量级网络代理，它们与应用程序部署在一起，而对应用程序透明。”

定位

基础设施层

功能

服务间通讯



部署

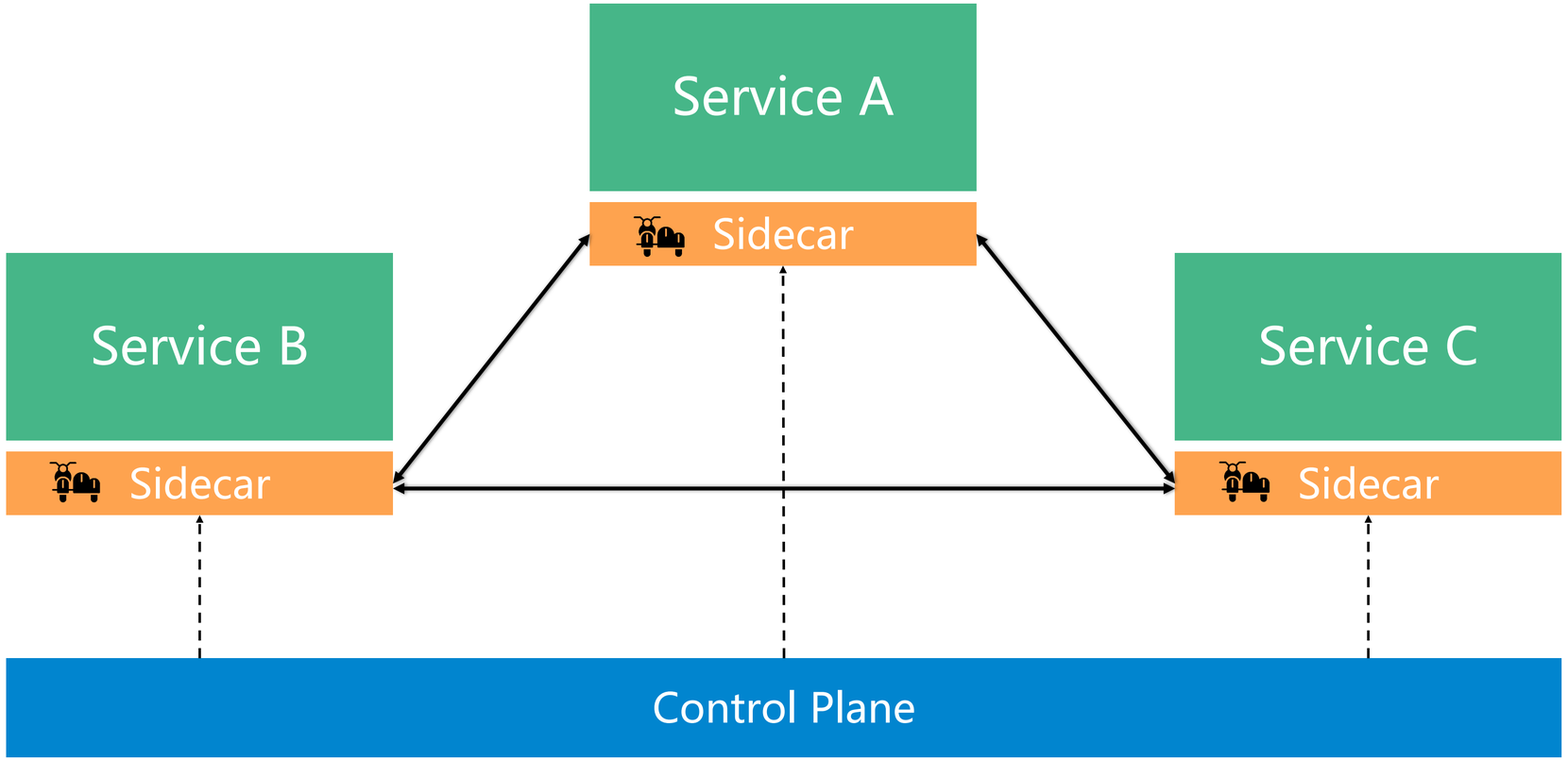
轻量级网络代理
（sidecar）

零侵入

对应用透明

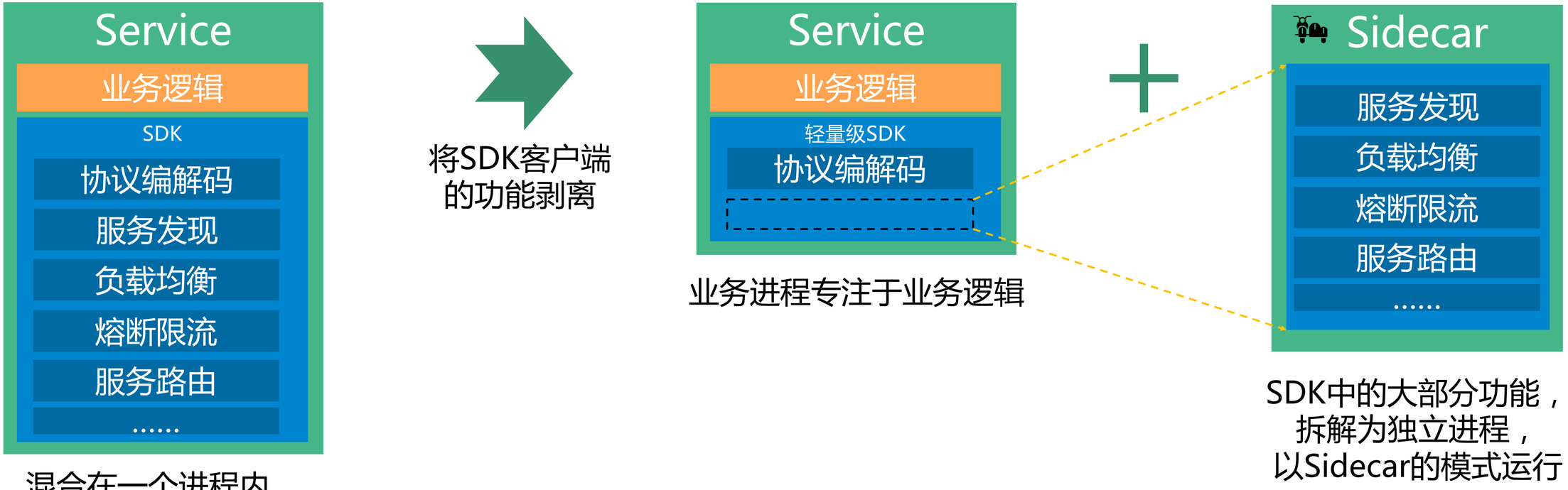
ServiceMesh的创新：Sidecar 模式

Alibaba



Servicemesh的基本思路：关注点分离 + 独立维护

Alibaba



混合在一个进程内，
应用既有业务逻辑，
也有各种非业务的功能

业务进程专注于业务逻辑

SDK中的大部分功能，
拆解为独立进程，
以Sidecar的模式运行

ServiceMesh发展趋势：以Istio为代表

Alibaba

协议支持

Istio只支持HTTP和gRPC，社区在提供更多协议支持，包括 Dubbo、Thrift、Redis，如Aeraki 项目

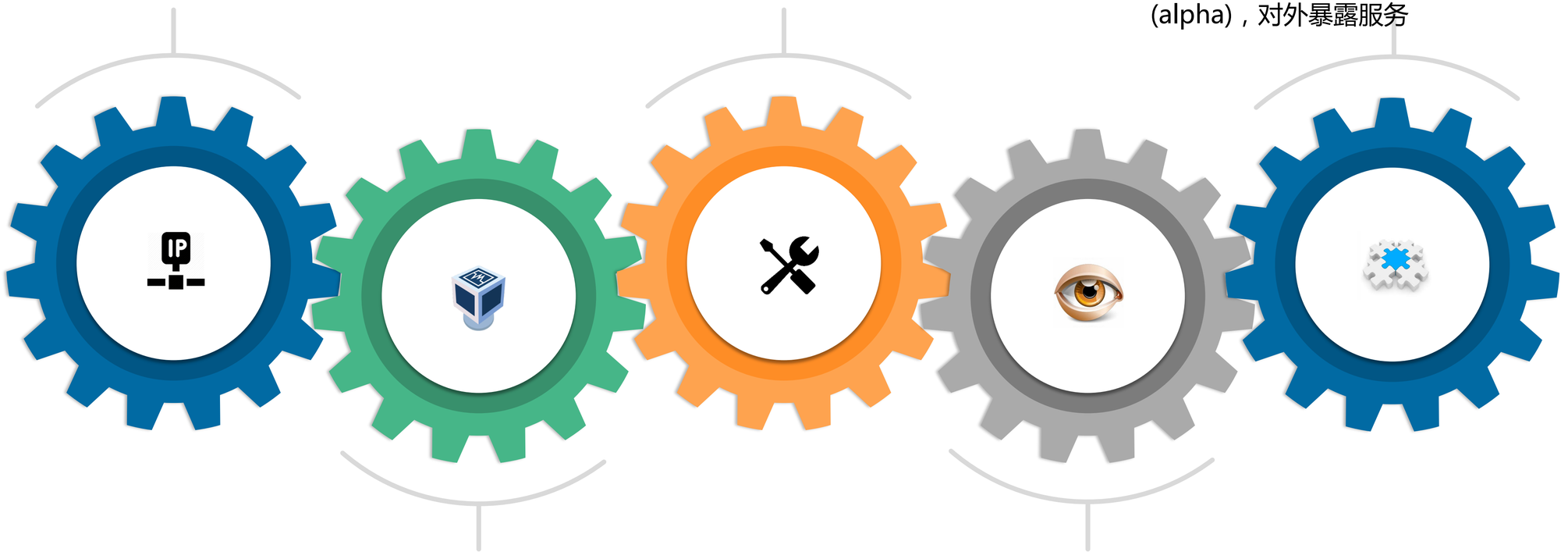
易用性

- Istio 1.5：控制平面单体化，合并多个组件为istiod
- Istio 1.7：主推 Operator安装方式，增强istioctl工具，支持在sidecar启动之后再启动应用容器
- Istio 1.8：改善升级和安装,引入istioctl bug-report

外部集成

和非 service mesh 体系相互访问，实现平滑迁移。

- Istio曾计划通过MCP协议提供统一的解决方案
- Istio 1.7：MCP协议被废弃，改为 mcp over xds
- Istio 1.9：Kubernetes Service API支持(alpha)，对外暴露服务



虚拟机支持

- Istio 0.2：Mesh Expansion
- Istio 1.1：ServiceEntry
- Istio 1.6：WorkloadEntry
- Istio 1.8：WorkloadGroup 和智能DNS代理
- Istio 1.9：虚拟机集成

可观测性

- Istio 1.8：正式移除Mixer，在Envoy基于wasm重新实现Mixer功能
- Istio 1.9：远程获取和加载wasm模块

ServiceMesh发展趋势：以Istio为代表

Alibaba

协议支持

Istio只支持HTTP和gRPC，社区在提供更多协议支持，包括 Dubbo、Thrift、Redis，如Aeraki 项目

易用性

- Istio 1.5：控制平面单体化，合并多个组件为istiod
- Istio 1.7：主推 Operator安装方式，增强istioctl工具，支持在sidecar启动之后再启动应用容器
- Istio 1.8：改善升级和安装,引入istioctl bug-report

外部集成

和非 service mesh 体系相互访问，实现平滑迁移。

- Istio曾计划通过MCP协议提供统一的解决方案
- Istio 1.7：MCP协议被废弃，改为 mcp over xds
- Istio 1.9：Kubernetes Service API支持(alpha)，对外暴露服务

目标：Make Istio Product Ready (Again...)

虚拟机支持

- Istio 0.2：Mesh Expansion
- Istio 1.1：ServiceEntry
- Istio 1.6：WorkloadEntry
- Istio 1.8：WorkloadGroup 和智能DNS代理
- Istio 1.9：虚拟机集成

可观测性

- Istio 1.8：正式移除Mixer，在Envoy基于wasm重新实现Mixer功能
- Istio 1.9：远程获取和加载wasm模块

回顾总结：ServiceMesh 六年（2016 - 2021）

Alibaba

蓬勃发展，但核心元素未变

定位

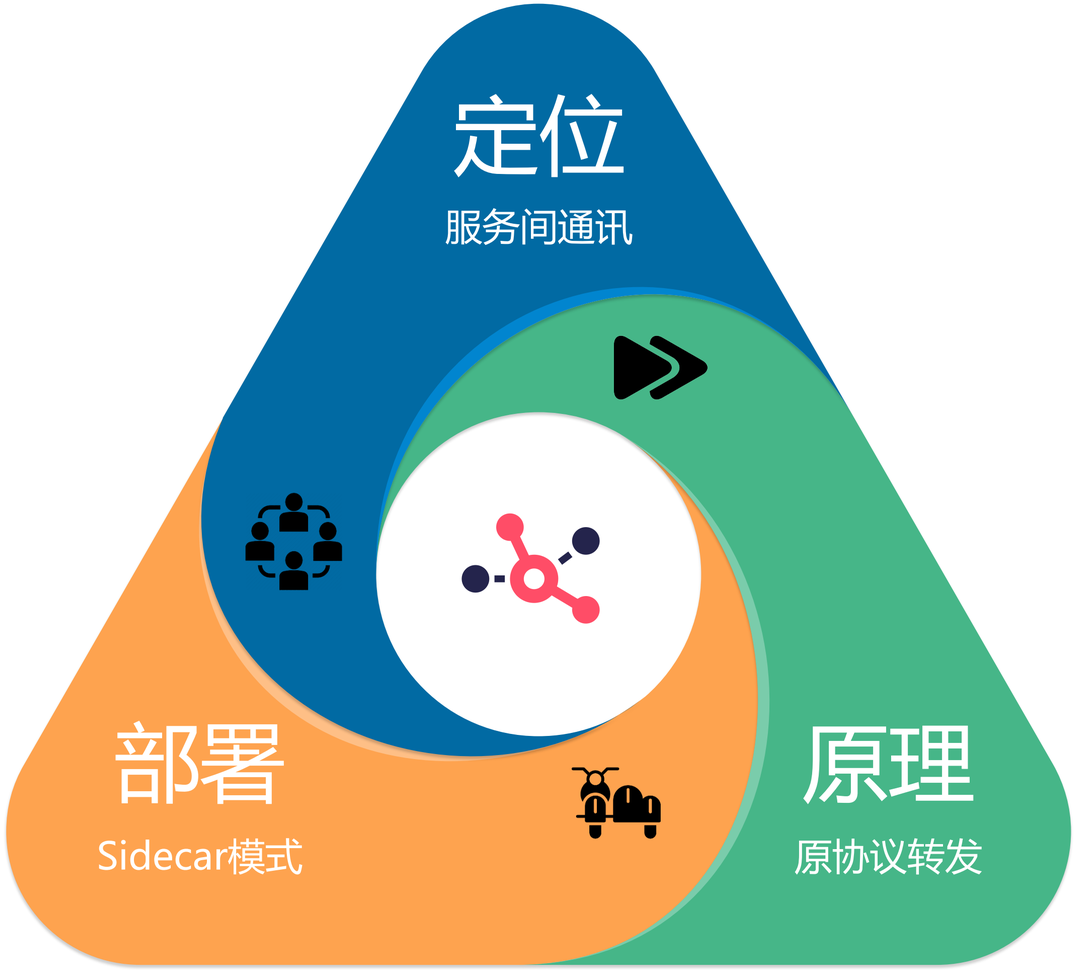
Servicemesh的定位是提供 **服务间通讯** 的基础设施层，范围包括HTTP和RPC：支持HTTP1.1/REST，支持HTTP2/gRPC，支持TCP协议。也有一些小的尝试如对redis 、 kafka的支持。

部署

ServiceMesh 支持 Kubernetes 和虚拟机，但都是采用 **Sidecar 模式**部署，没有采用其他方式如 Node 部署、中心化部署。

原理

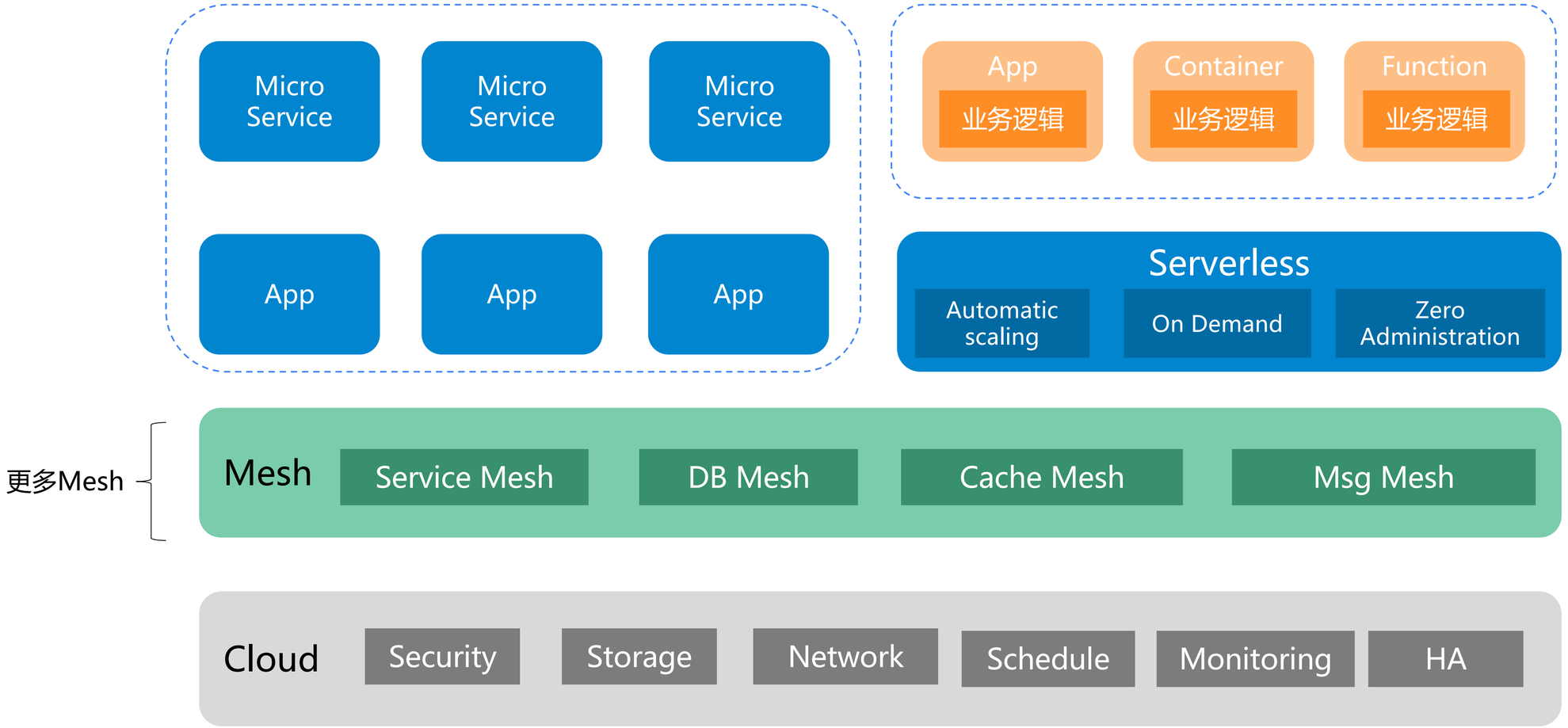
Servicemesh的工作原理是**原协议转发**，原则上不改变协议内容（通常只是header有些小改动）。为了达到零侵入的目标，还引入了iptables等流量劫持技术



2 演进：云原生分布式应用运行时

更多实践：业界对 Sidecar 模式的推广，更多Mesh形态出现

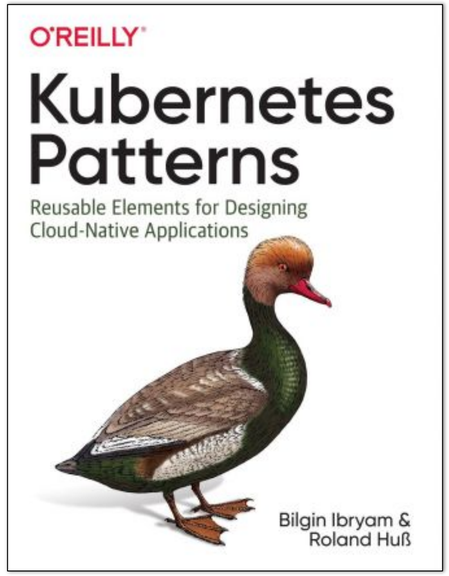
Alibaba



参考：蚂蚁金服从2019年开始的各种实践案例

理论升华： Bilgin Ibryam 提出 Multi-Runtime 的理念

Alibaba



Author Kubernetes Patterns

Bilgin Ibryam

Architect @ RedHat

Committer @ Apache Camel

Multi-Runtime的理论出发点：分布式应用存在四大类需求

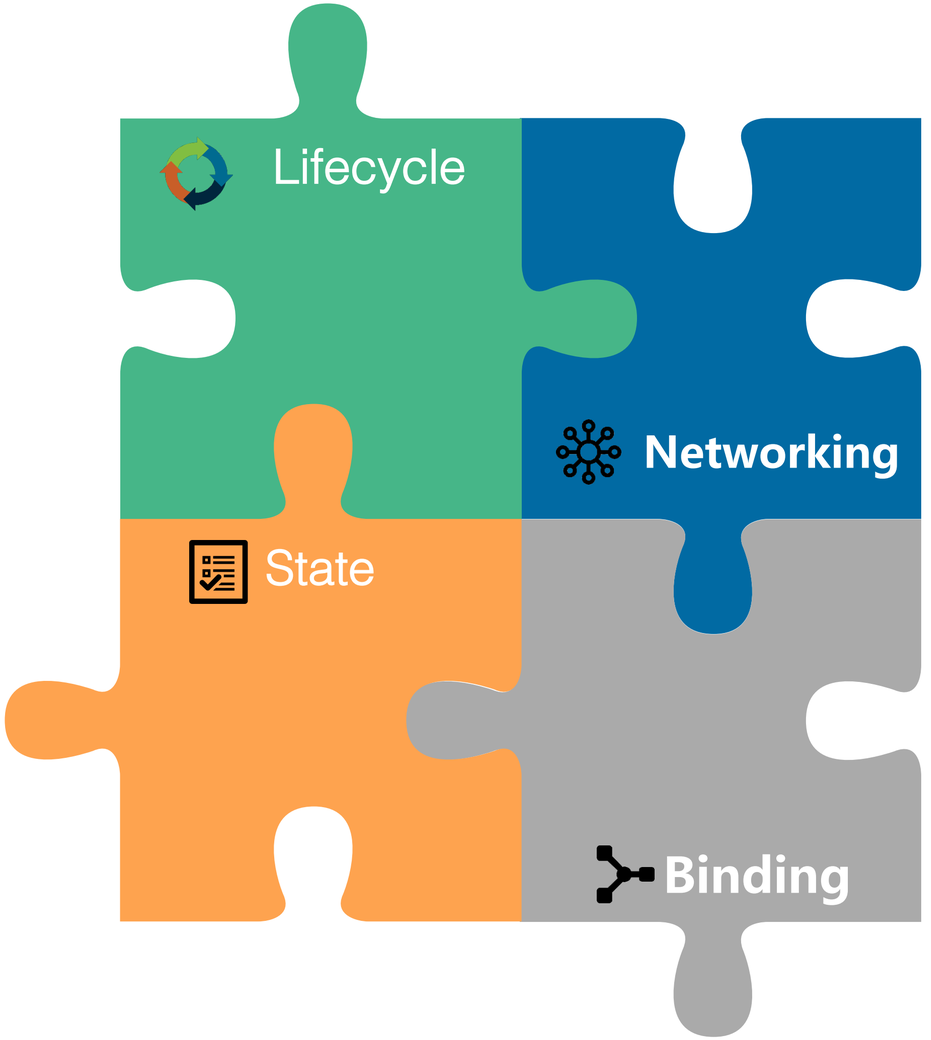
Alibaba

生命周期

- Package
- Health check
- Deployment
- Scaling
- Configuration

状态

- Workflow mgmt.
- Idempotency
- Temporal scheduling
- Caching
- Application state



网络

- Service discovery
- A/B testing, canary rollouts
- Retry, timeout, circuit breaker
- **Point-to-Point**, pub/sub
- Security, observability

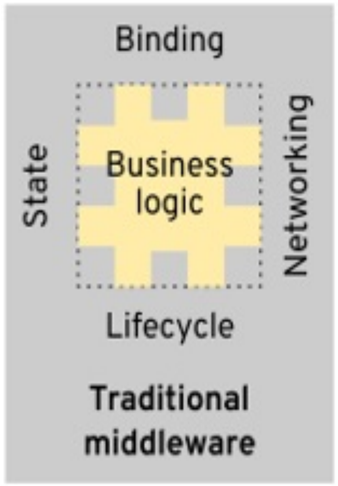
绑定

- Connectors
- Protocol conversion
- Message transformation
- Message routing
- Transnationality

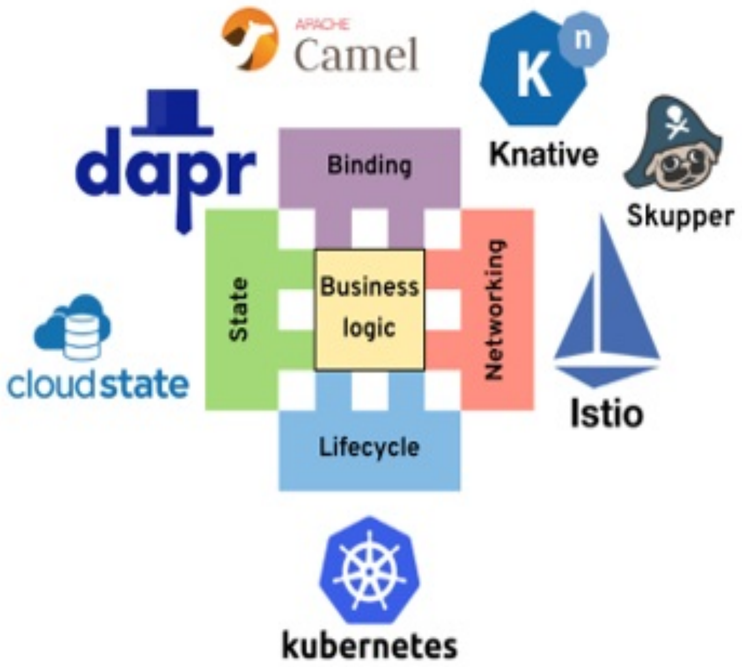
Multi-Runtime的理论推导：效仿Servicemesh，能力外移

Alibaba

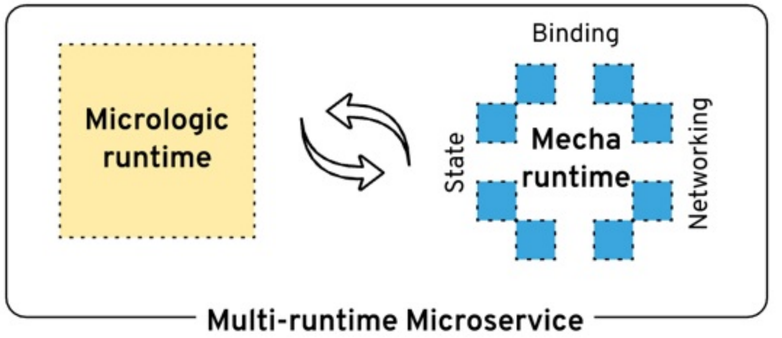
传统中间件模式



步骤1：能力外移到各种 Runtime



步骤2：多个runtime**整合**



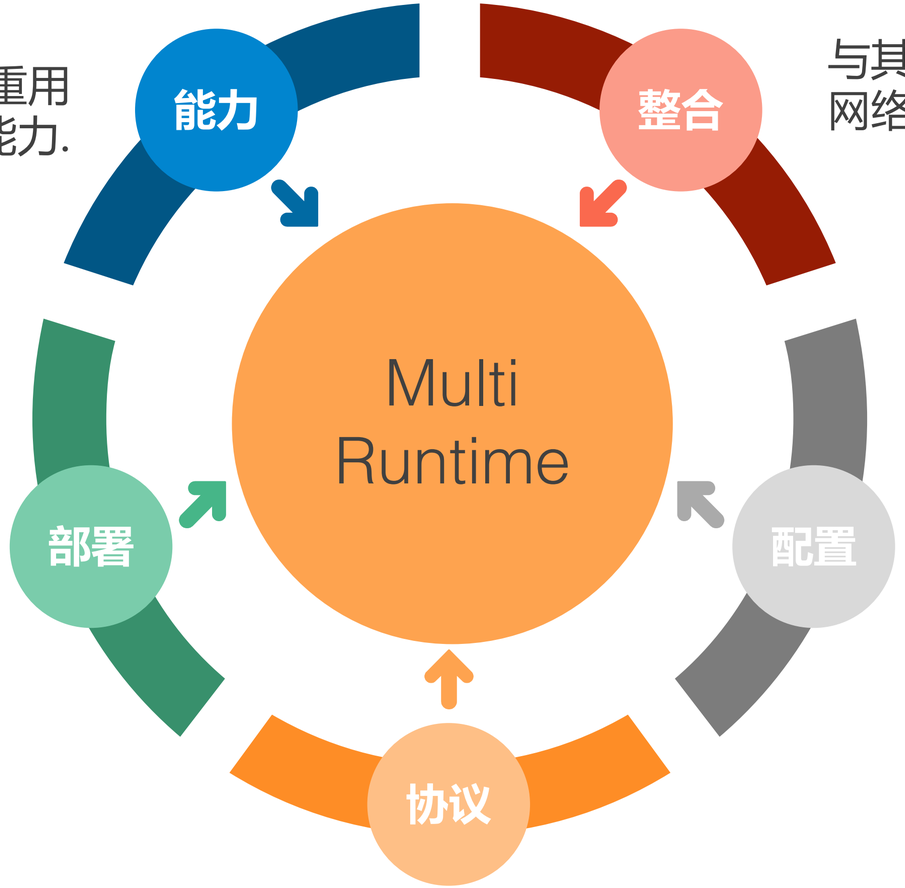
Micrologic	Mecha
• Developed in-house	• Off-the-shelf mechanincs
• Custom business logic	• Configurable capabilities
• Higher-level language	• Declarative (YAML, JSON)
• HTTP/gRPC, CloudEvents	• OpenAPI, AsyncAPI, SQL

Mecha Runtime **和应用** Runtime **共同组成微服务**

引入 Multi-Runtime / Mecha 到云原生分布式应用

Alibaba

Mecha 是通用的，高度可配置的，可重用的组件，提供分布式原语作为现成的能力.



与其依靠多个代理来实现不同的目的（例如网络代理，缓存代理，绑定代理），不如使用一个Mecha提供所有这些能力。

Mecha 可以与单个Micrologic组件一起部署(Sidecar模式)，也可以部署为多个共享(如Node模式)。

Mecha以简单的文本格式（例如YAML，JSON）声明式地配置，指示要启用的功能以及如何将其绑定到Micrologic端点。

Mecha不对 Micrologic 运行时做任何假设。它与使用开放协议和格式（如HTTP/gRPC，JSON，Protobuf，CloudEvents）的多语言微服务甚至单体一起使用。

Multi-Runtime的特点

Alibaba

提供能力的方式和范围

Multi-Runtime提供的是分布式能力，体现为应用需要的各种分布式原语，并不局限于单纯的服务间点对点通讯的网络代理

Runtime部署的方式

Multi-Runtime的部署模型，不局限于Sidecar模式，Node模式在某些场景下（如Edge/IoT，Serverless FaaS）可能会是更好的选择。

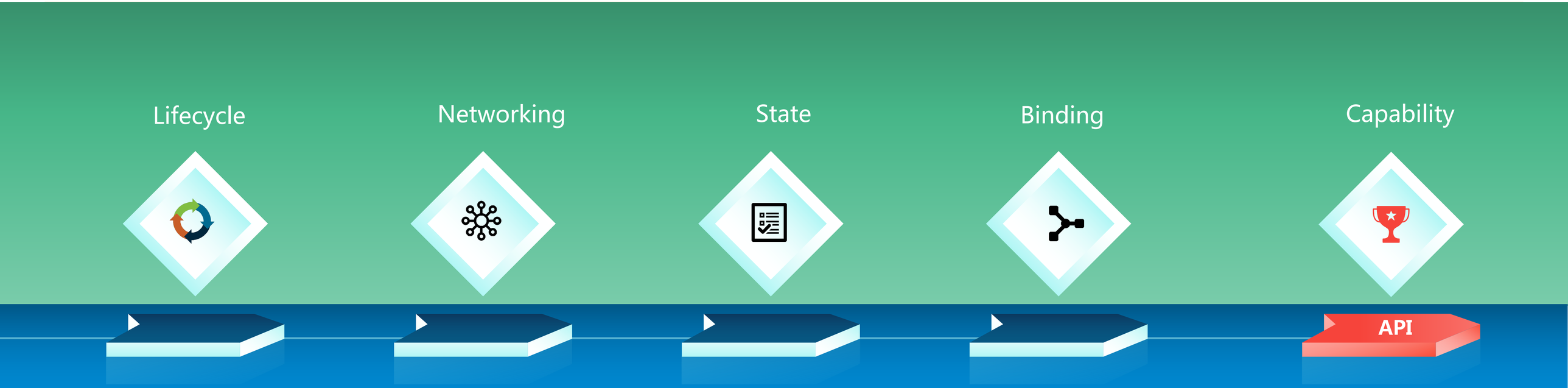
和App的交互方式

Multi-Runtime 和应用之间的交互是开放而有 API 标准的，Runtime 和 Micrologic 之间的“协议”体现在API上，而不是原生的TCP通讯协议。

Servicemesh	服务间通讯	Sidecar 模式	原协议转发
差异	功能	部署	工作原理
Multi-Runtime	各种分布式能力 （包括服务间通讯）	Sidecar 模式 Node 模式	标准API + 语言SDK + 提供各种能力的Runtime

Multi-Runtime的本质：面向云原生应用的分布式能力抽象层

Alibaba



生命周期

- Package
- Health check
- Deployment
- Scaling
- Configuration

网络

- Service discovery
- A/B testing, canary rollouts
- Retry, timeout, circuit breaker
- Point-to-Point, pub/sub
- Security, observability

状态

- Workflow mgmt.
- Idempotency
- Temporal scheduling
- Caching
- Application state

绑定

- Connectors
- Protocol conversion
- Message transformation
- Message routing
- Transnationality

抽象

- 将能力抽象为API
- 为每种能力提供多种实现
- 开发时：面对能力编程
- 运行时：通过配置选择实现

3 介绍：分布式应用运行时Dapr

什么是 Dapr？

Dapr is a portable, event-driven runtime that makes it easy for any developer to build resilient, stateless and stateful applications that run on the cloud and edge and embraces the diversity of languages and developer frameworks.

Dapr是一个可移植的、事件驱动的运行时，它使任何开发者都能轻松地构建运行在云和边缘的弹性、无状态和有状态的应用程序，并拥抱语言和开发者框架的多样性。

定位

运行时

功能

简化应用开发

- 弹性
- 有状态
- 无状态
- 事件驱动

多样性

多语言

- 任何开发者
- 开发者框架

可移植性

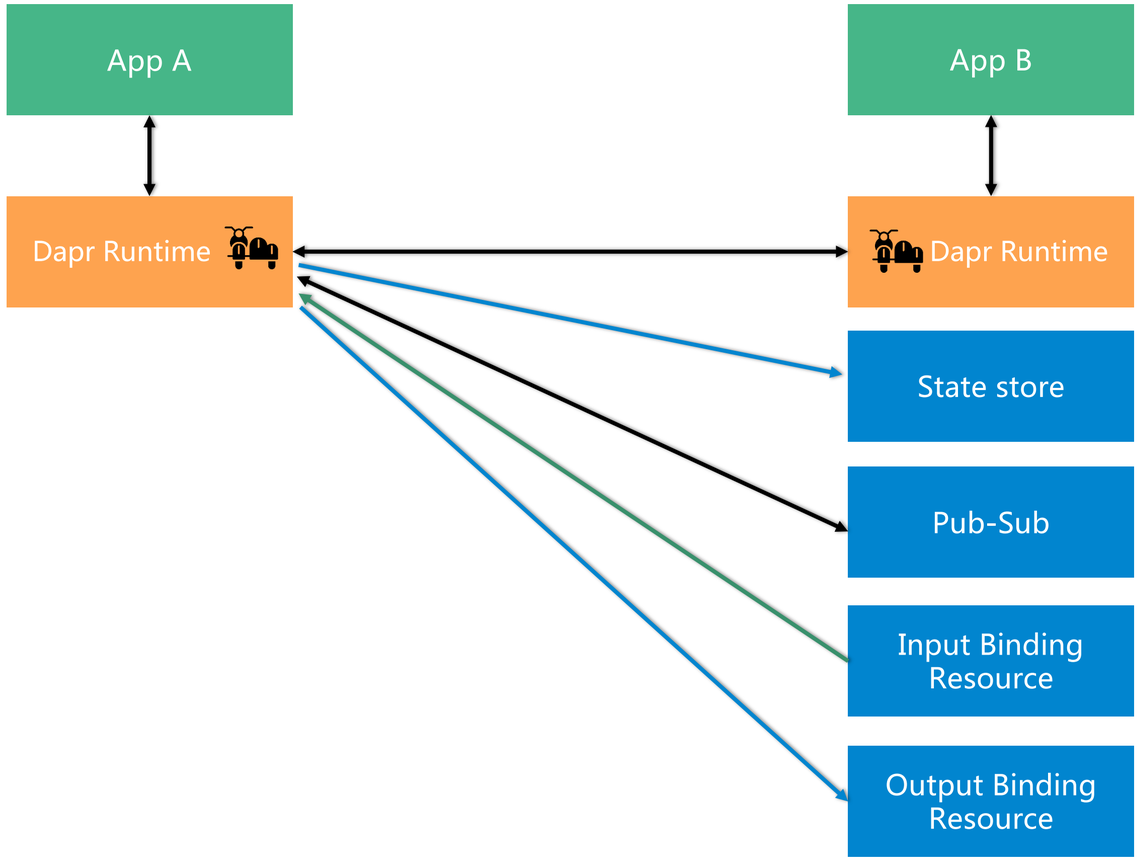
云 & 边缘

分布式应用运行时

Distributed Application Runtime

Dapr 同样基于 Sidecar 模式：但场景比 ServiceMesh 要复杂

Alibaba



HTTP1.1 / gRPC

Redis / consul / cassandra
Memcached / mongodb / zookeeper
Gcp.firestore / aws dynamodb

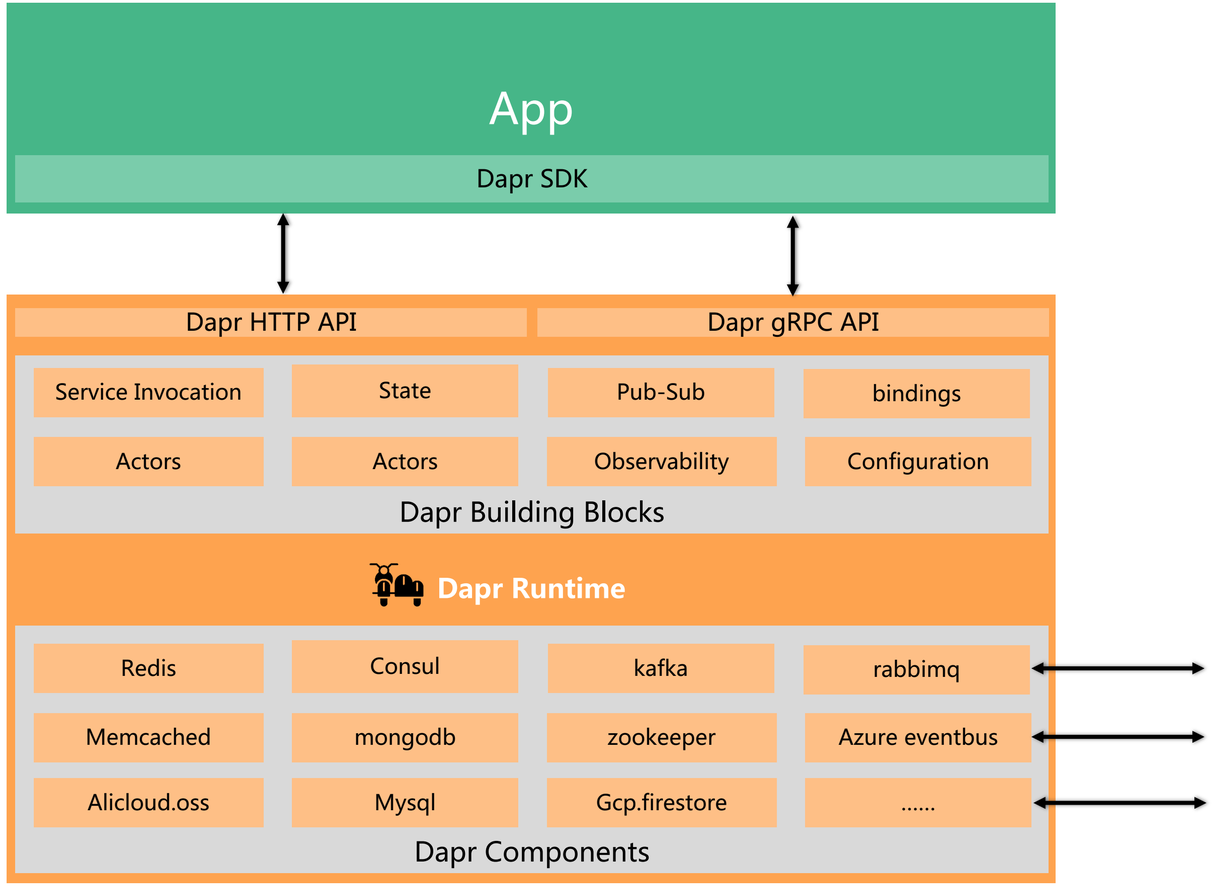
Redis / kafka / rabbimq
pulsar / mqtt / hazelcast
Gcp.pubsub / azure servicebus

kafka / rabbimq
Azure eventbus, aws kinesis

Alicloud.oss / aws sqs
Gcp.bucket/ aws.s3
Mysql / postgres / twitter
http / smtp

Dapr Runtime 的架构: API / Building Blocks / Components

Alibaba



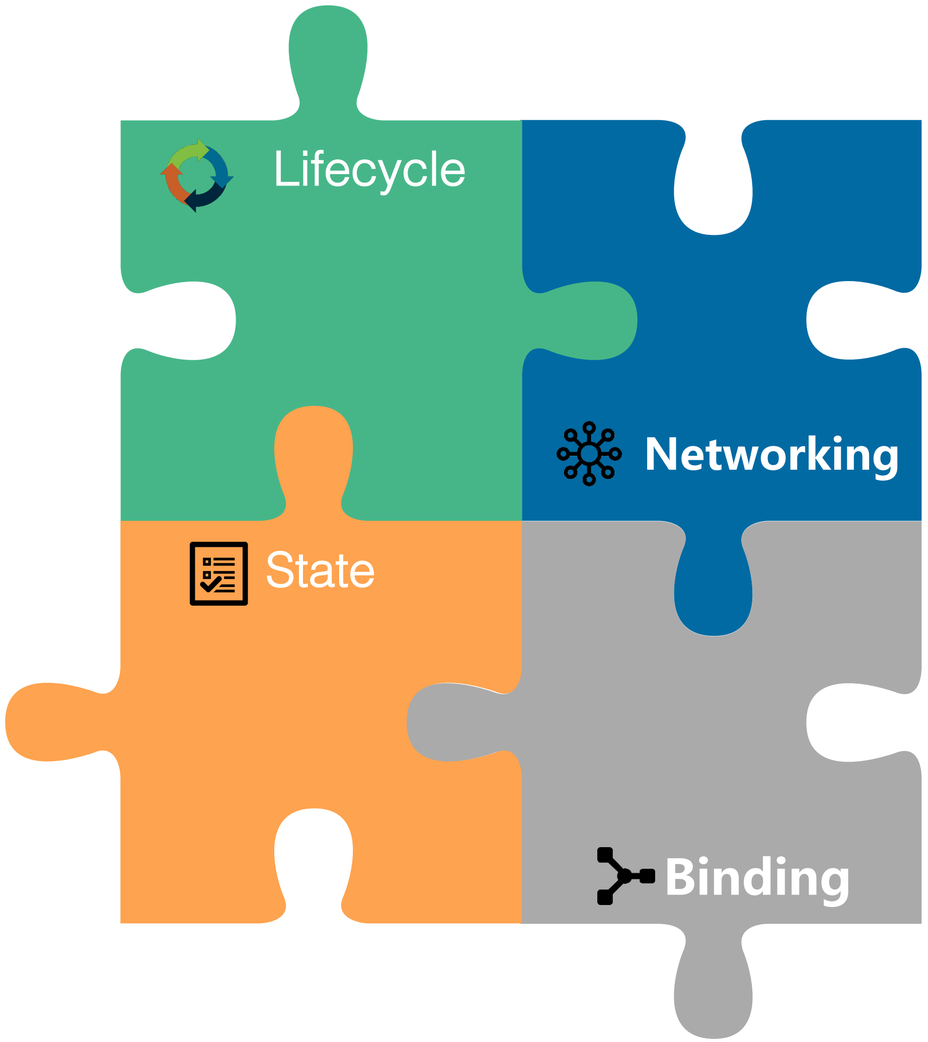
Multi-Runtime 的本质：面向云原生应用的分布式能力抽象层

生命周期

- Package
- Health check
- Deployment
- Scaling
- Configuration

状态

- Workflow mgmt.
- Idempotency
- Temporal scheduling
- Caching
- Application state



网络

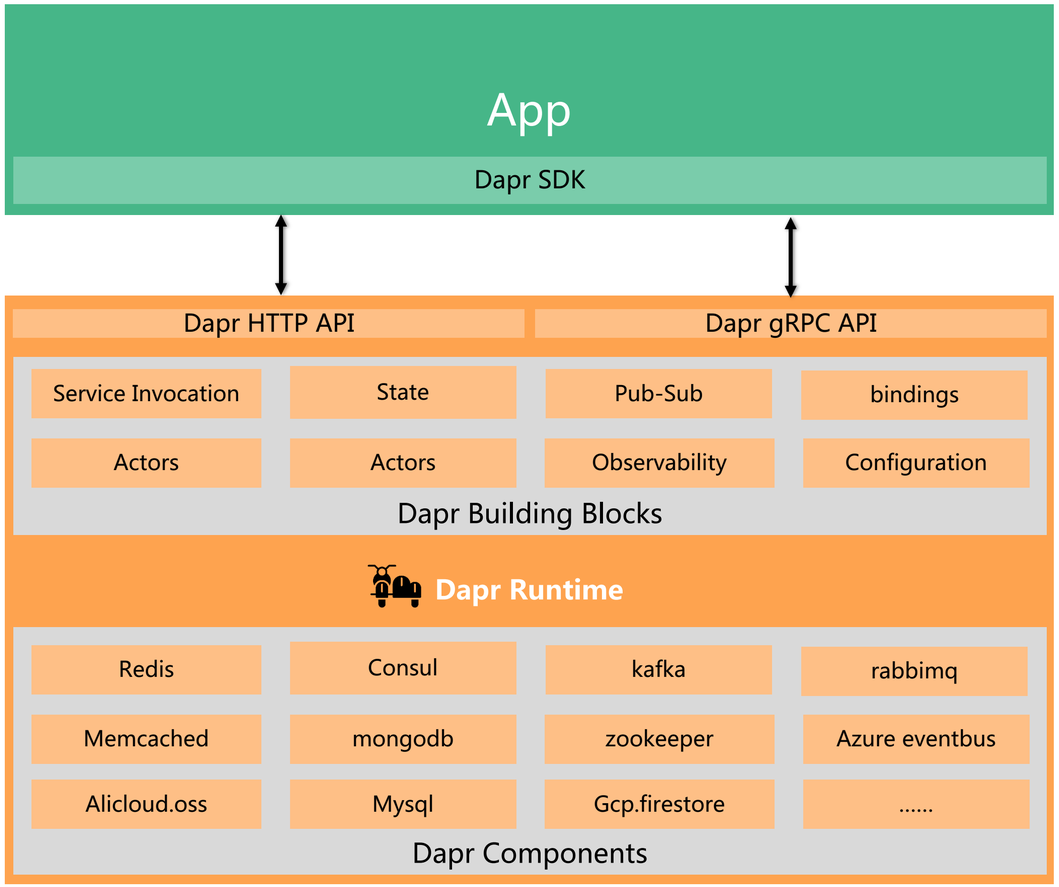
- Service discovery
- A/B testing, canary rollouts
- Retry, timeout, circuit breaker
- **Point-to-Point**, pub/sub
- Security, observability

绑定

- Connectors
- Protocol conversion
- Message transformation
- Message routing
- Transnationality

Multi-Runtime 理念在 Dapr Runtime 架构上的体现

Alibaba



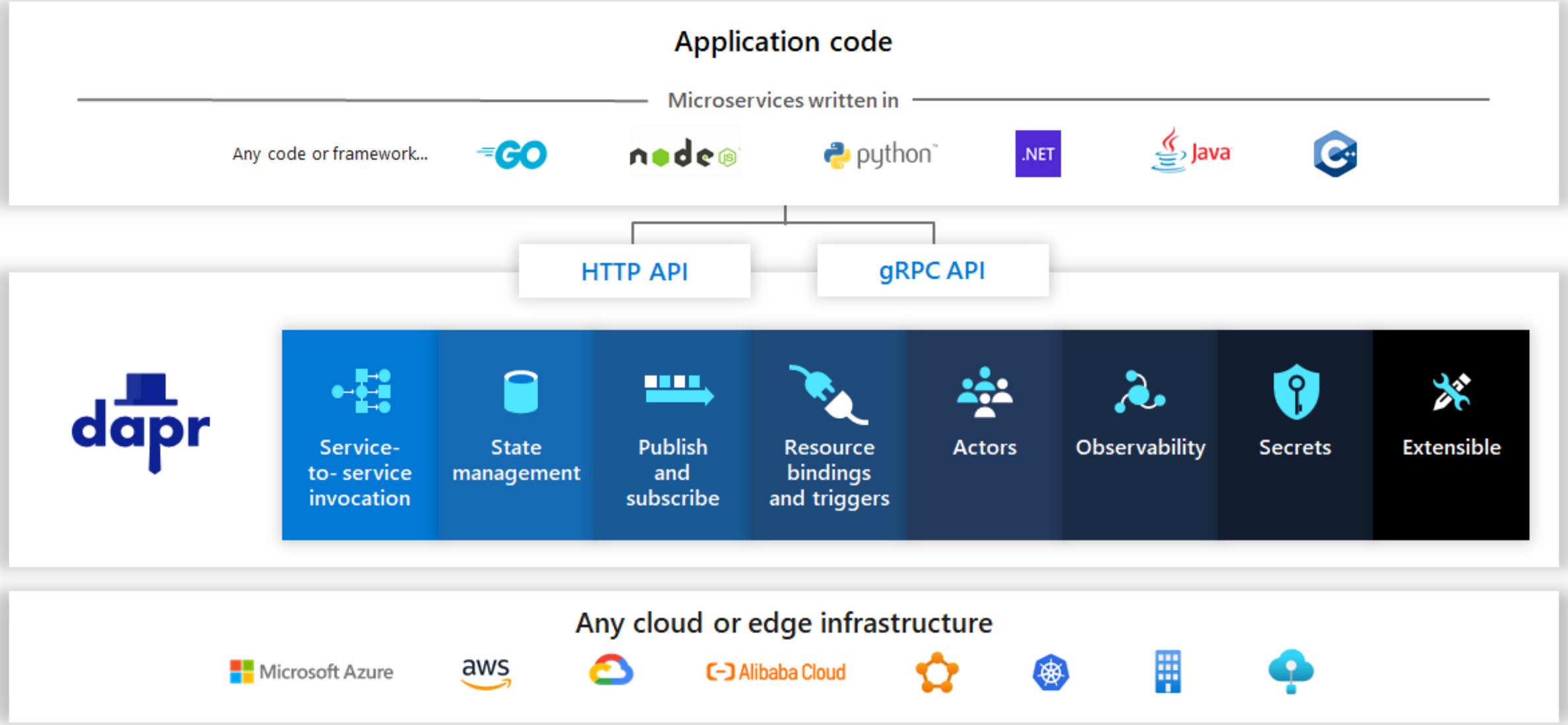
API ---------- 抽象

Building Blocks ---------- 能力

Components ---------- 实现

Dapr 愿景： Any language, any framework, anywhere

Alibaba



洞察跨组件和服务的调用

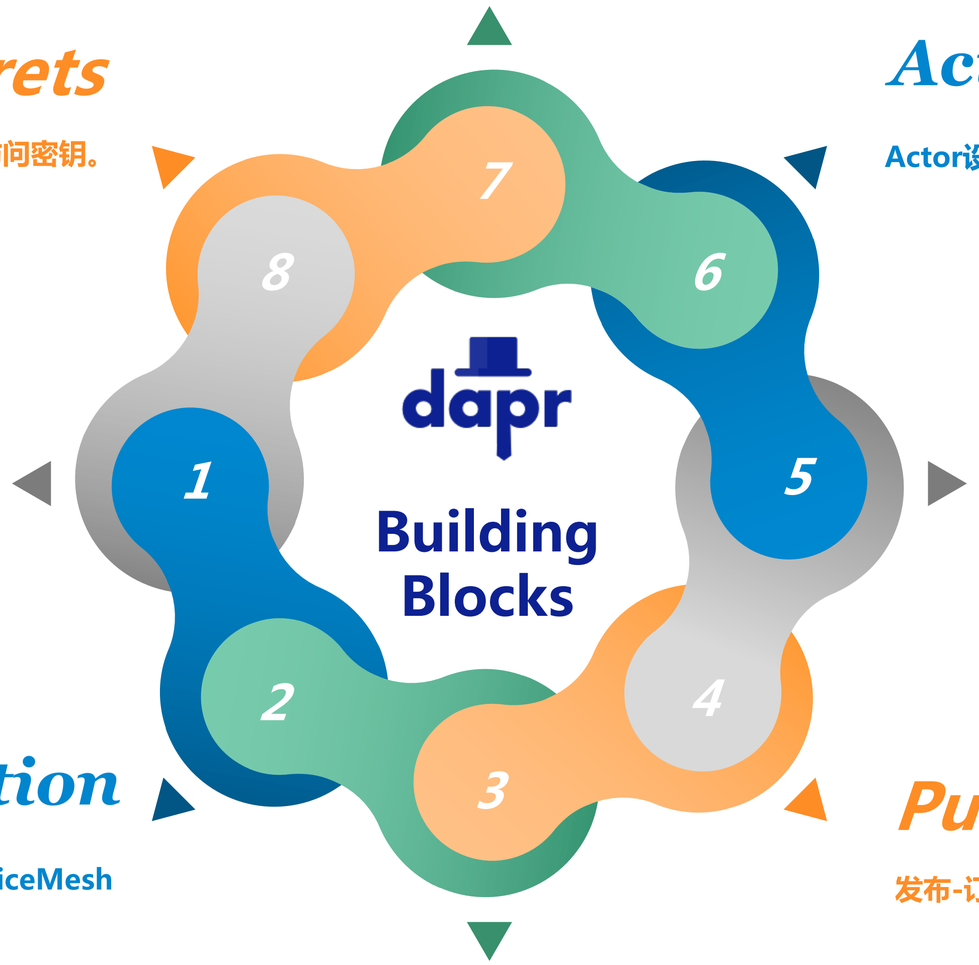
Observability

Secrets

让应用可以安全的访问密钥。

Actors

Actor设计模型，以可重用的actor对象的方式封装代码和数据



Configuration

获取应用的动态配置，进行中……

Resource Binding

资源绑定

- Input Binding: 外部输入源通过事件触发代码
- Output Binding: 绑定到外部资源如数据、消息队列等

Service Invocation

服务调用，服务间点对点通讯，类ServiceMesh

Pub-Sub

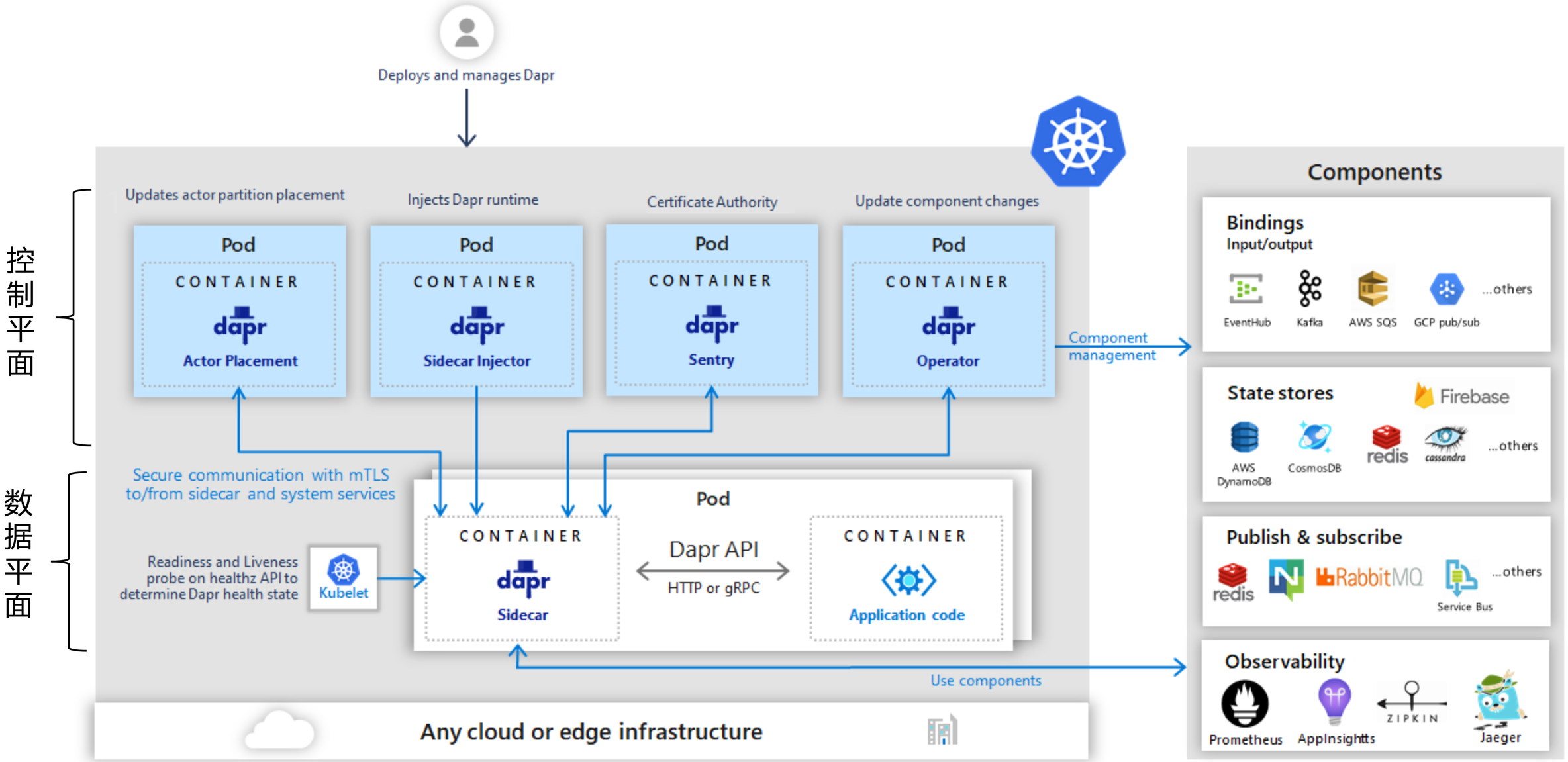
发布-订阅。服务间的消息通讯，异步，1对多

State Management

管理应用的状态，简化有状态服务的开发

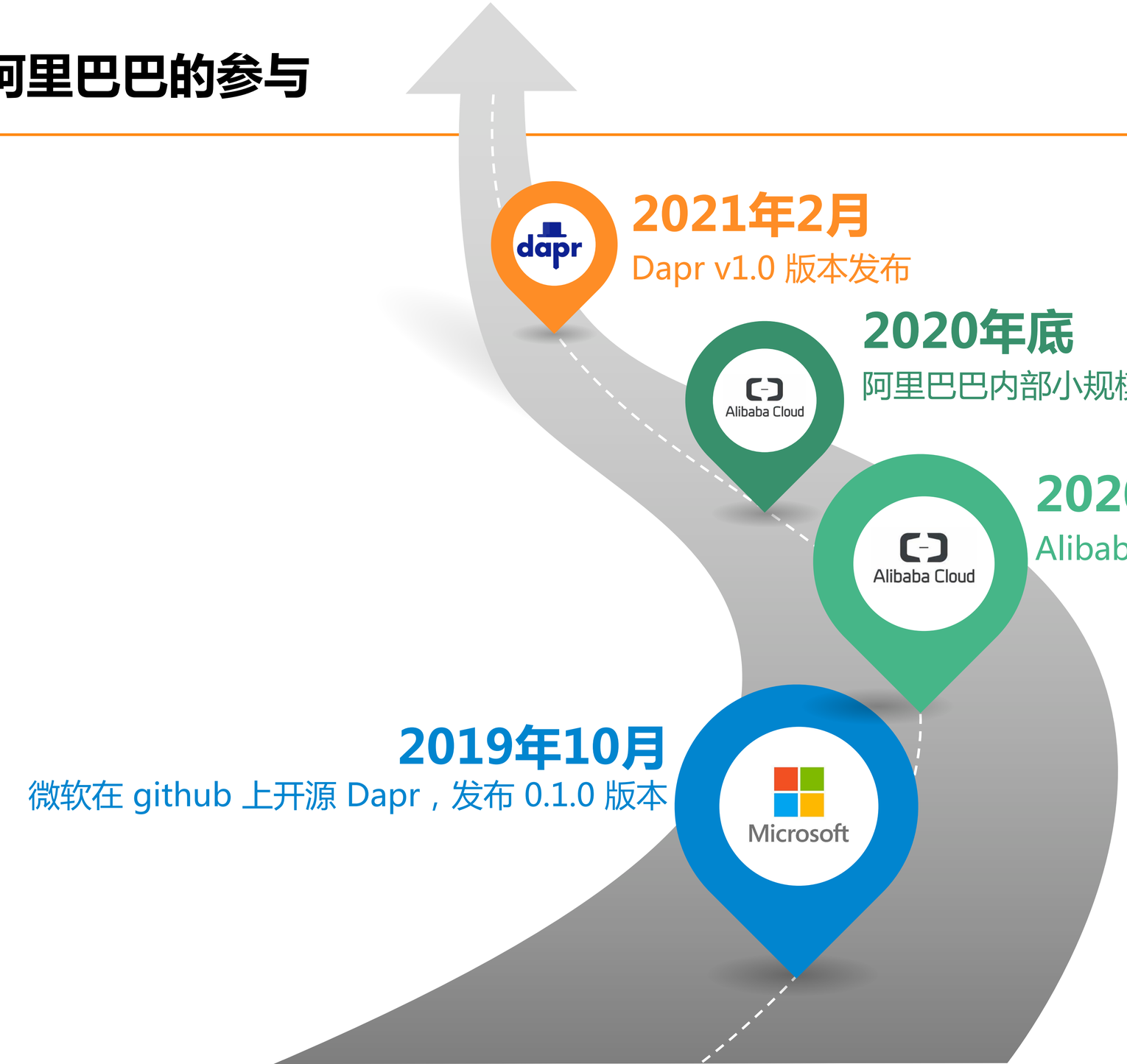
Dapr的架构：控制平面、数据平面、组件

Alibaba



Dapr的发展和阿里巴巴的参与

Alibaba



Dapr 入门教程

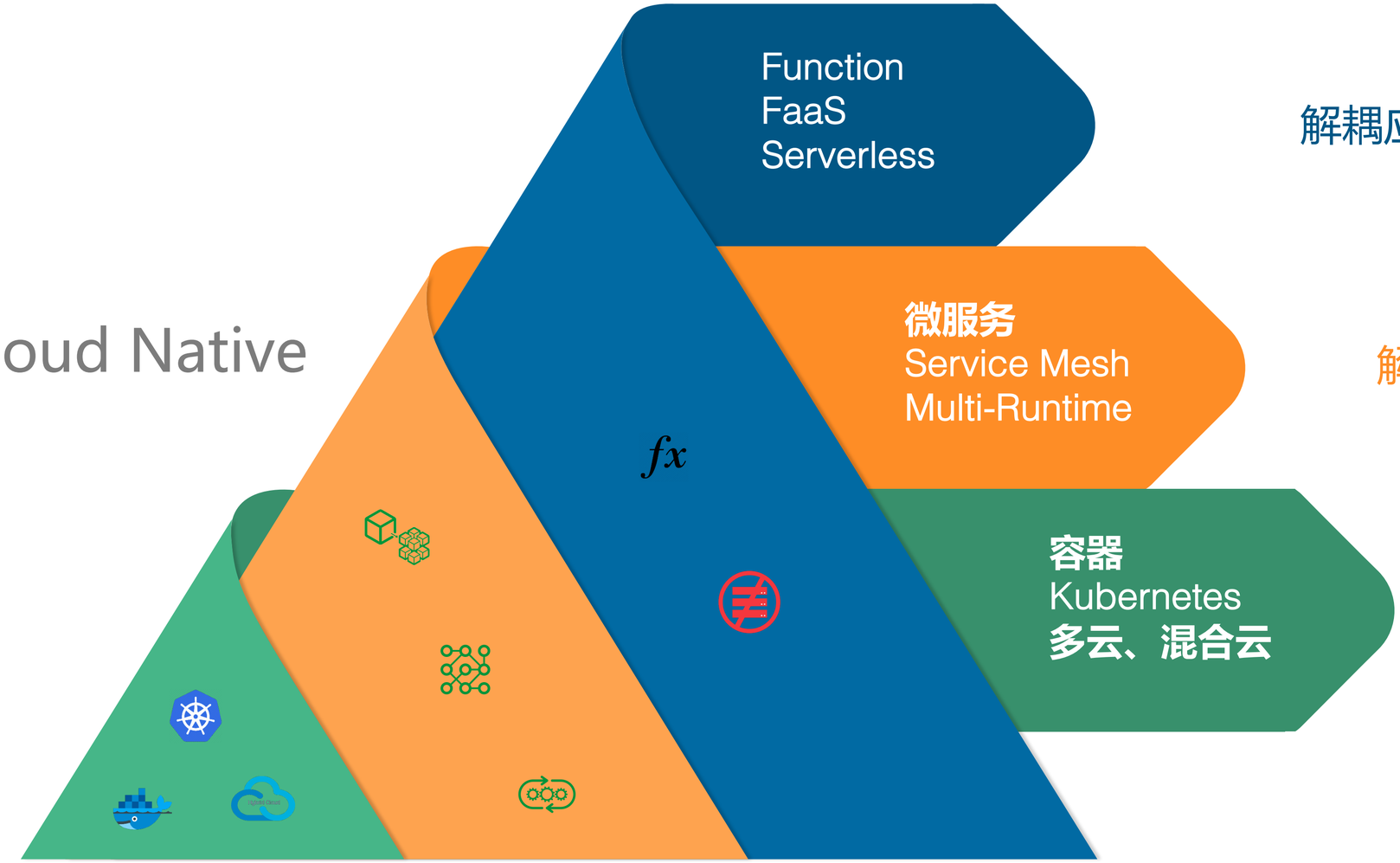
https://start.aliyun.com/course?id=gImrX5Aj

十分钟快速领略开源分布式运行时 Dapr 应用的开发、部署过程

4 展望：应用和中间件的未来形态

时代背景：云原生的历史大潮

Alibaba



解耦应用与服务器

解耦应用与分布式能力

云原生时代的操作系统

应用的期望

云原生背景下的甜美梦想

应用可以使用任意喜爱而适合的语言编写，可以快速开发和快速迭代。应用需要的能力都可以通过标准的API提供，无需关心底层具体实现。应用可以部署到任意的云端，不管是公有云、私有云还是混合云，没有平台和厂商限制，无需代码改造。应用可以根据流量弹性伸缩，顶住波峰的压力，也能在空闲时释放资源……

Serverless 会是云上应用的理想形态和主流发展方向。

实现真正的语言自由
为每个语言提供平等的分布式能力

跨平台的多云、混合云部署
无厂商锁定，可以自由迁移

应用轻量化：快速开发，快速迭代
按需分配，按使用付费

中间件的方向

在美好目标推动下摸索前进

完善的多语言支持和应用轻量化的需求推动中间件将更多的能力从应用中分离出来，Sidecar 模式会推广到更大的领域，越来越多的中间件产品会开始 Mesh 化，整合到 Runtime 。

对厂商锁定的天然厌恶和规避，会加剧对可移植性的追求，从而进一步促使为下沉到 Runtime 的中分布式能力提供标准而业界通用的 API。

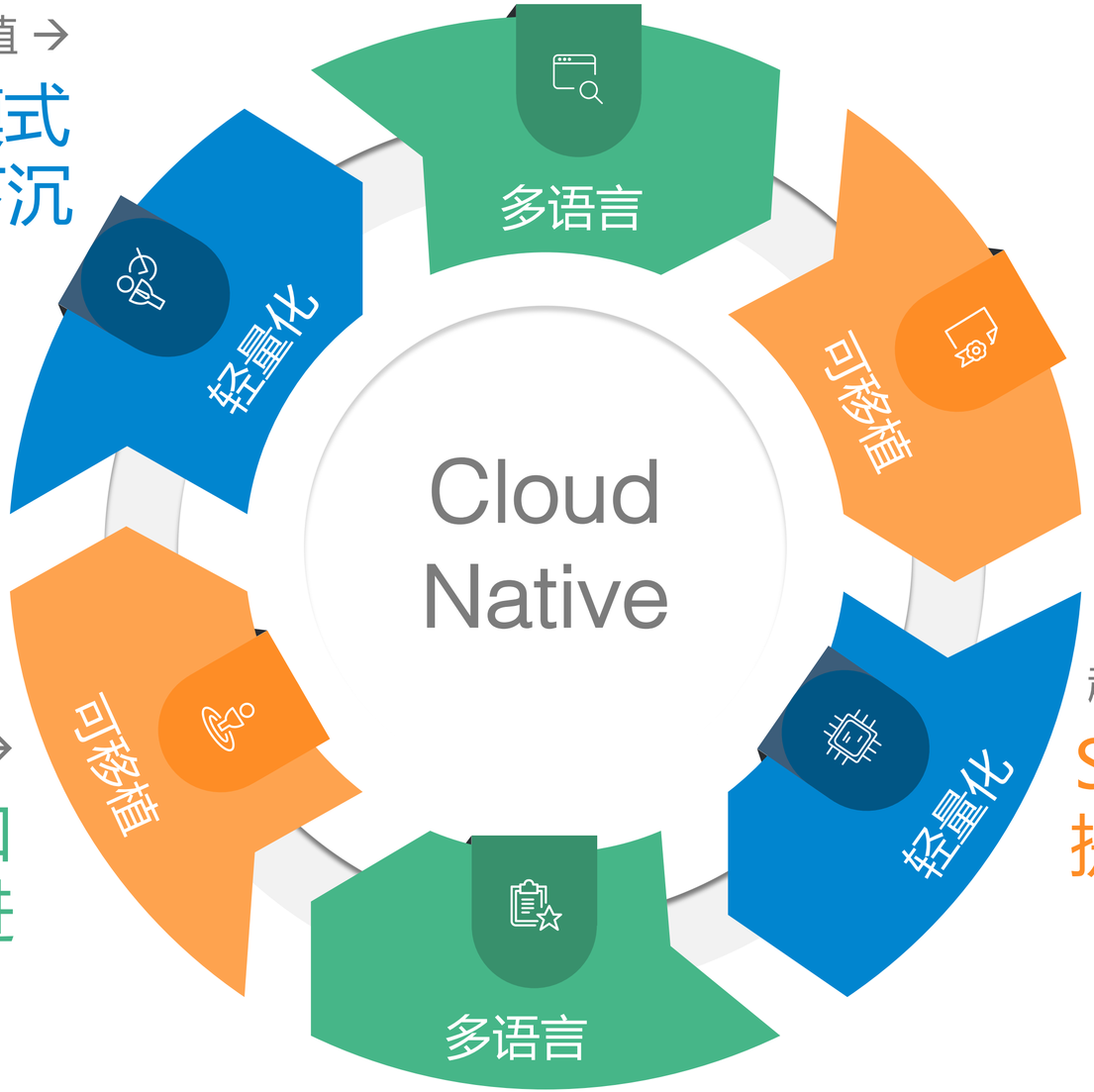
API 的标准化和社区认可，将成为 Runtime 普及的最大挑战，但同时也将推动各种中间件产品改进自身实现，实现中间价产品和社区标准API之间的磨合与完善。

云原生的飞轮滚滚向前

Alibaba

多语言 + 轻量化 + 可移植 →

采用 Sidecar 模式
分布式能力下沉



越来越多的 Sidecar →

Sidecar 整合为 Runtime
提供越来越完善的能力

更高的可移植性追求 →

推动 API 标准化和
中间件产品改进

感谢观看

作者：敖小剑 @ 阿里云