
devops
全球敏捷运维峰会
阿里云数据库平台架
构演进之路
演讲人：徐东来

自我介绍

徐东来，花名云智

- **10余年**IT从业，从传统IT企业到互联网
  - **8年**时间，从事安全审计领域产品研发

    从工程师到技术总监，陆续参与并推出如下产品和服务：

    ✓ 面向企业的软硬件一体的“**运维安全**审计产品”和“**数据库安全**审计产品”

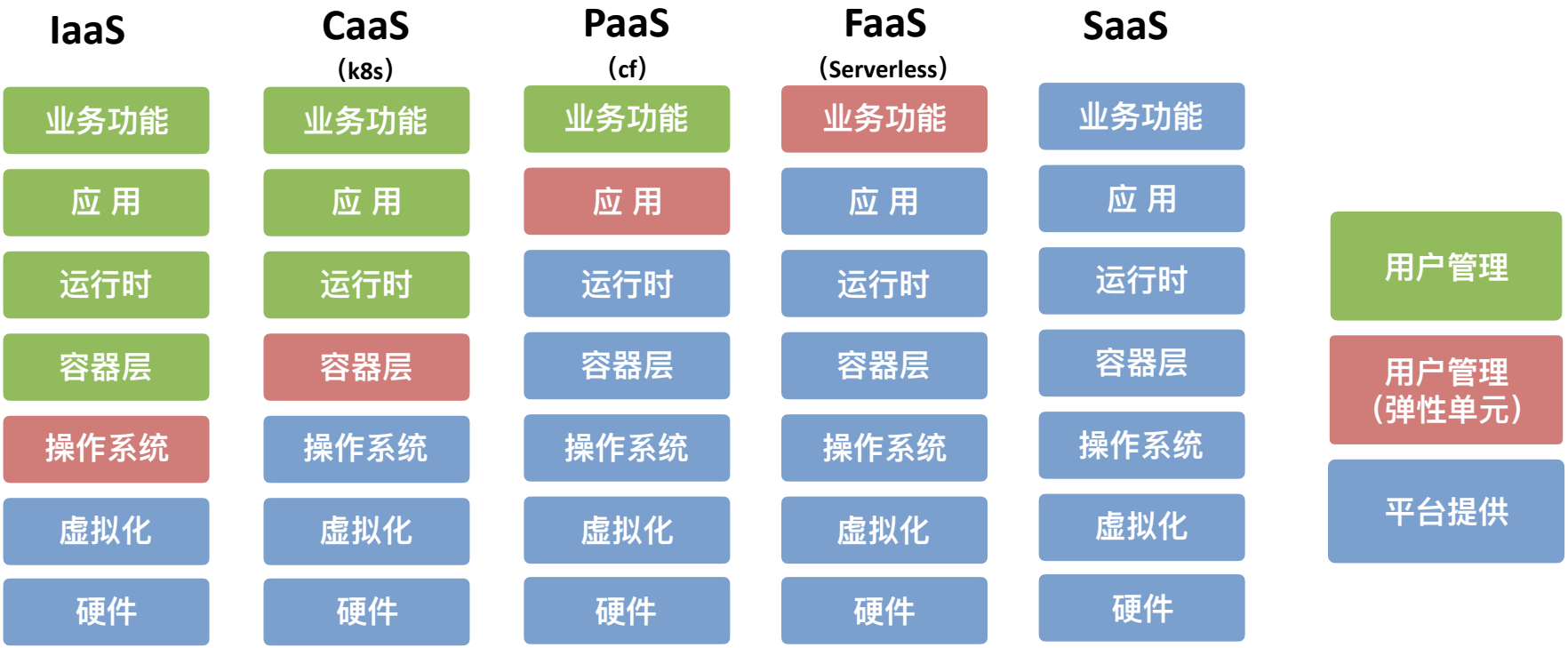
    ✓ 其他系列审计产品：网络安全审计、日志安全审计、恶意代码监控、等保工具箱、基于大数据架构的威胁检测系统等
  - **2年**时间，从事B2C电商的数据分析、搜索和推荐系统的研发；
- **2015年3月追寻云计算的梦想，来到了阿里云云数据库团队。**

  云数据库**平台和安全**相关系统的研发

主要内容

- DBaaS的定位
- 阿里云DBaaS的架构
  - IaaS层的选择
  - 应用拓扑的演进
  - 资源调度与分配
  - 任务工作流系统
  - 可用性检测与故障恢复
  - 弹性扩缩容系统
  - 备份与恢复
  - 性能监控
  - 部署与运维、DevOps实践
- DBaaS -> App as a Service VS. CaaS(k8s)

应用托管云化趋势



- 路径：
  - 完全自建->数据库云化->缓存/消息中间件云化->应用微服务化/Docker化；
  - 从看到主机，到看到容器和应用，到Serverless；

Stateless Service vs. Stateful Service

Stateless Service

- 无需持久化，一般为Web应用
- 容易水平扩展
- 可快速扩缩容
- 故障恢复快
- 只需考虑CPU和内存资源
- 非常合适使用容器编排类技术，如K8S、Swarm等

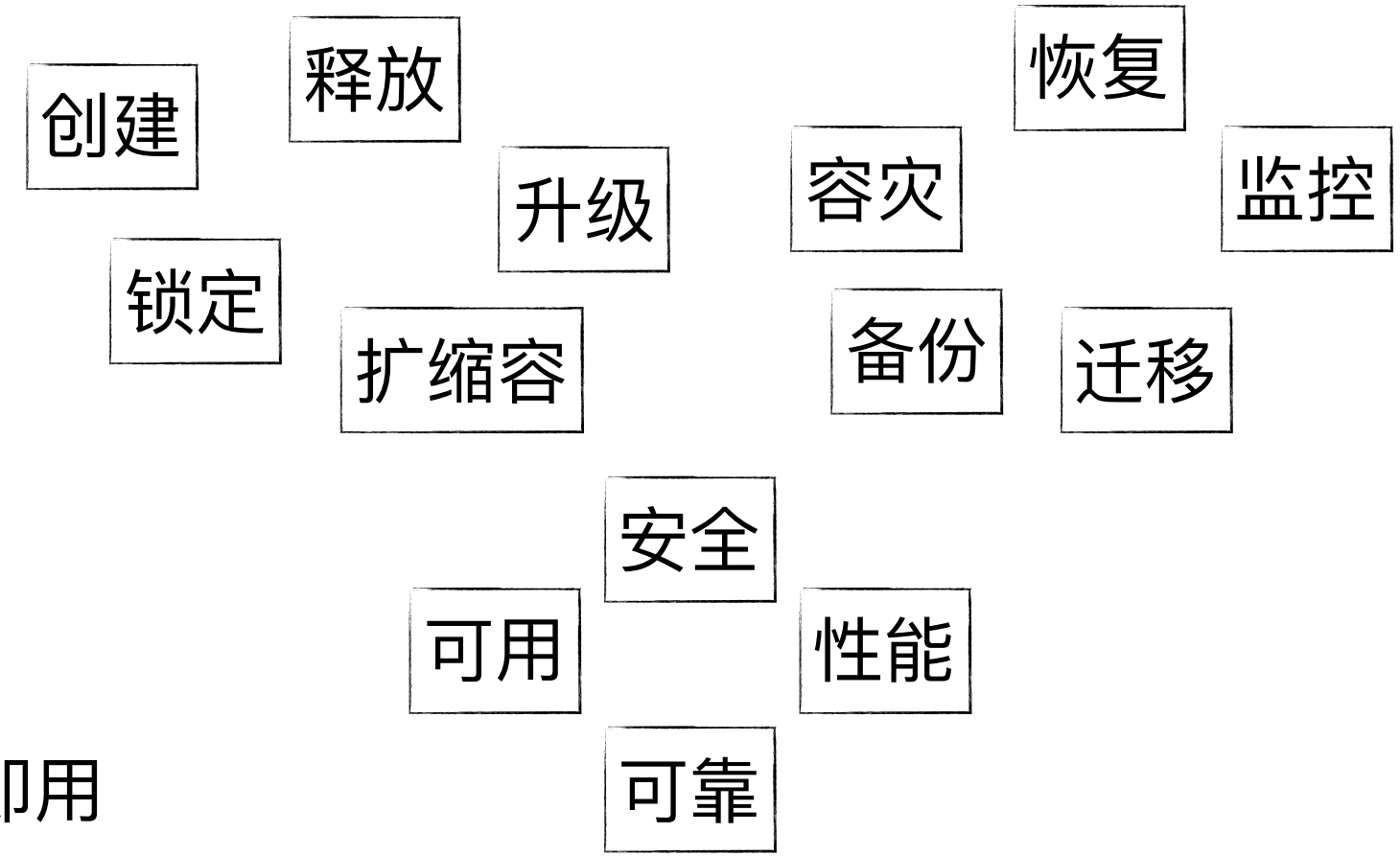
Stateful Service

- 持久化，一般为数据库、缓存、中间件等应用
- 要考虑数据同步、备份还原、数据可靠性、数据安全等
- 不易扩展
- 故障恢复慢
- 综合考虑CPU、内存、IO、磁盘等资源
- 现有的容器平台支持度不好

DBaaS是什么？

全生命周期管理

自动化运维



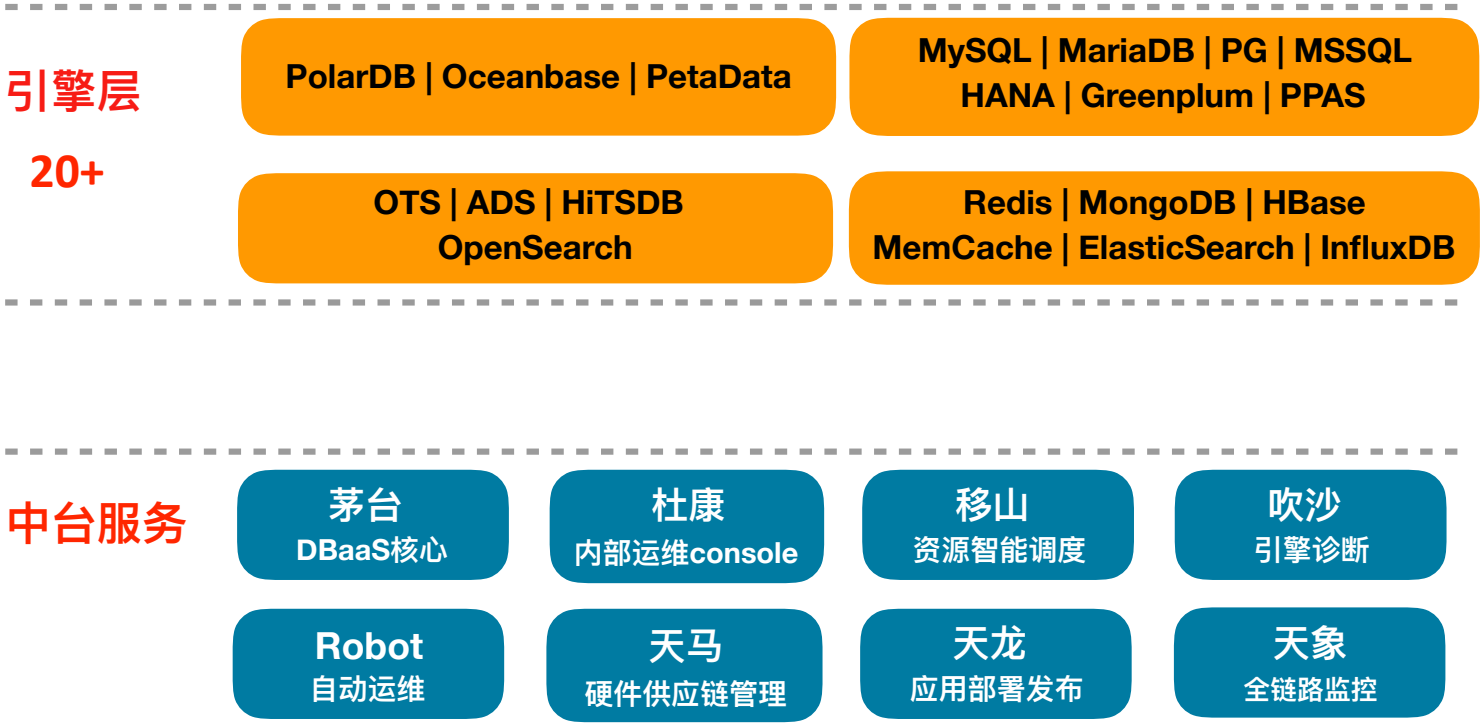
开箱即用

SLA保证

DBaaS与 自建数据库的区别

	DBaaS	自建数据库
服务可用性	99.95%	需自行保障，自行搭建主从复制，RAID等
数据备份	自动备份	自行实现，但需要寻找备份存放空间以及定期验证备份是否可恢复
硬件成本	无托管费用，无软硬件投入，按需付费	数据库服务器成本相对较高；每台服务器每年托管费超过5000元；对于MS SQL Server需支付许可证费用
部署运维	即时开通，快速部署，弹性扩容，按需选购	需硬件采购与，机房托管，部署机器，周期较长
资源利用率	按实际结算，利用率很高	考虑峰值，资源利用率很低

阿里云数据库产品及中台



架构演进的特征

性能提升

软件优化
硬件加速：
RDMA / 25G / FPGA

效率提升

DEVOPS
机器 / 应用交付自动化

稳定性

资源争抢问题
SLA大盘

用户体验

迁移防闪断
热升级
自定义运维时间

成本降低

资源利用率
库存预测

安全体系化

多层防护
事前、事中与事后结合
国内外安全合规法案

平台化通用化

从支撑DB，到支撑通用应用
应用拓扑从简单到复杂
微服务化，专业化分工

数据驱动闭环

业务画像
异常分析与智能诊断
个性化推荐

阿里云DBaaS的演进

DBAAS 1.0
Only for RDS

资源模型为Master / Slave结构；
为数据库自动运维需要设计；
业务与平台紧耦合；
产品：MySQL / SQL Server / PGSQL

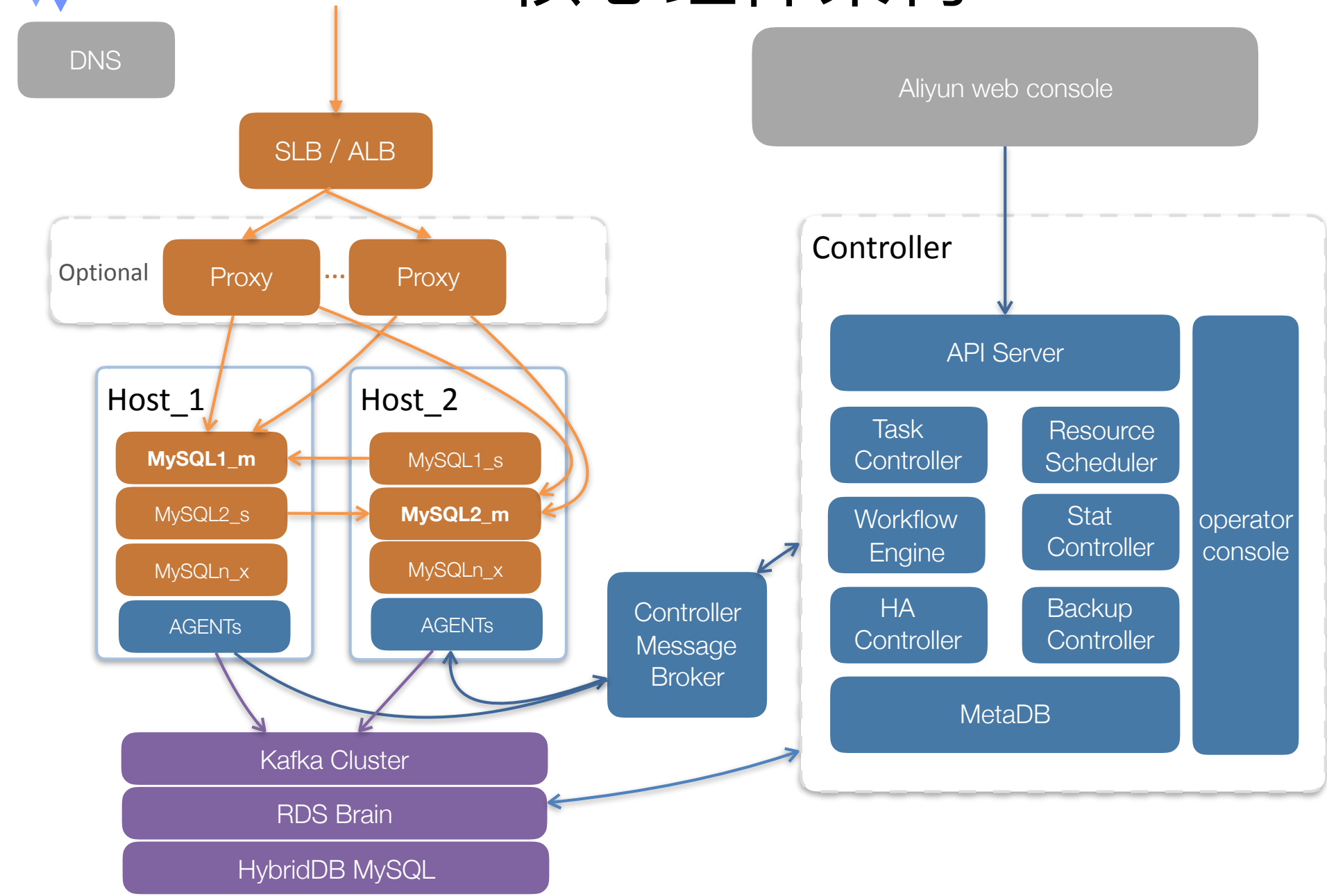
DBAAS 2.0
业务驱动的管控平台化

通用能力沉淀复制；
快速新产品开发模式；
组件通用化、插件化设计；
产品：MongoDB / Redis / Greenpulm等

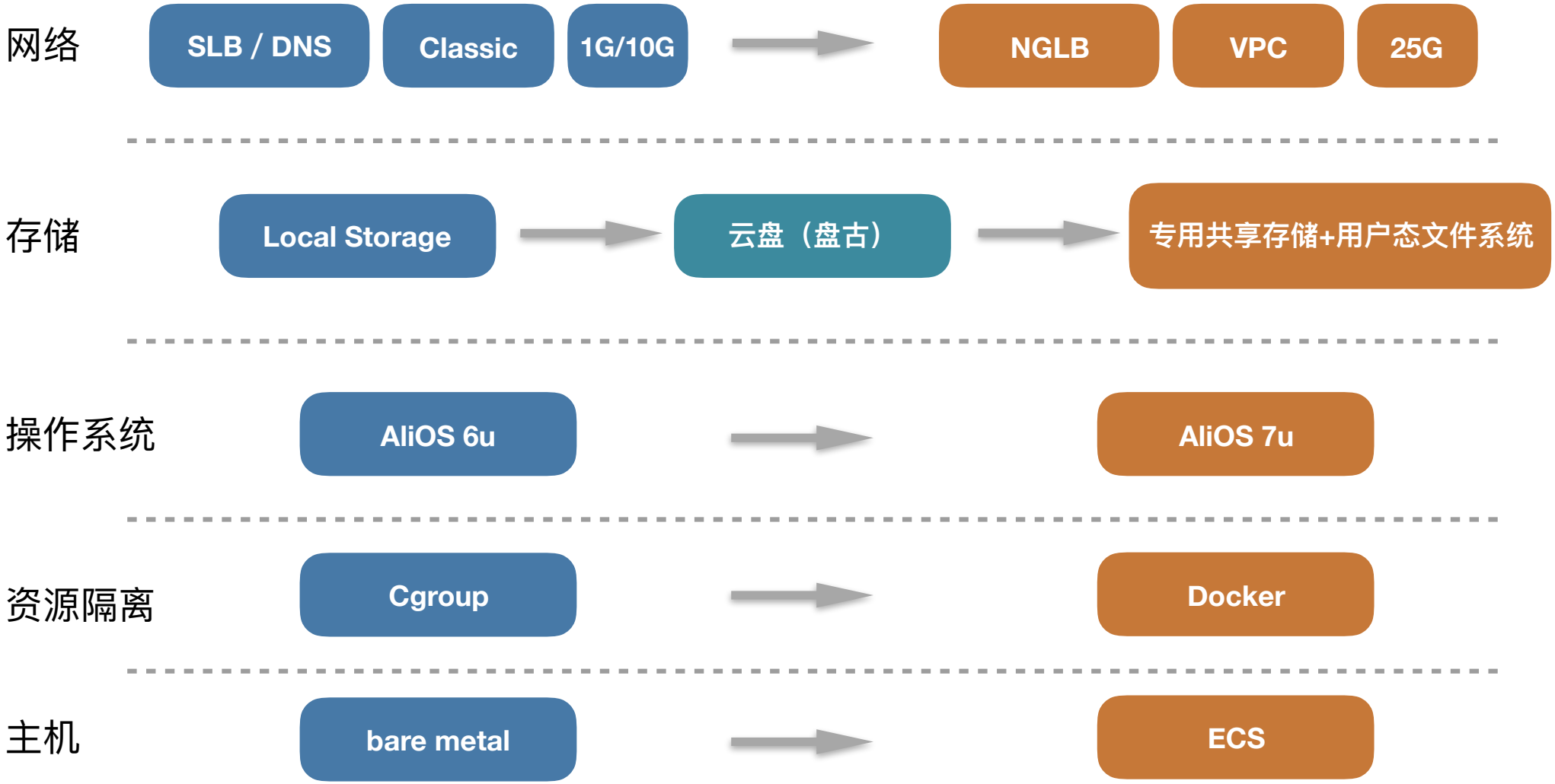
DBAAS 3.0
从业务回归到平台本质

not only DBAAS，do more
基于DSL的集群编排的快速新品开发能力
利用阿里云生态红利（ECS / 存储 / 网络）
工作流 / 备份 / HA / 监控平台化改造
平台和产品独立迭代能力
中台能力输出到更广泛的产品
产品：HBase / HiTSDB / PG10 / 列存 / kafka / Flink / ElasticSearch等

核心组件架构


DNS
SLB / ALB
Aliyun web console
Optional
Proxy
...
Proxy
Controller
API Server
Task Controller
Resource Scheduler
Workflow Engine
Stat Controller
HA Controller
Backup Controller
MetaDB
operator console
Host_1
MySQL1_m
MySQL2_s
MySQLn_x
AGENTs
Host_2
MySQL1_s
MySQL2_m
MySQLn_x
AGENTs
Controller Message Broker
Kafka Cluster
RDS Brain
HybridDB MySQL

IaaS层的选择


网络
SLB / DNS
Classic
1G/10G
NGLB
VPC
25G
存储
Local Storage
云盘（盘古）
专用共享存储+用户态文件系统
操作系统
AliOS 6u
AliOS 7u
资源隔离
Cgroup
Docker
主机
bare metal
ECS

几个概念

用户视角

用户实例

- 不感知应用的拓扑
- 只需要感知到服务地址和端口

规格

- 规格（CPU、内存、磁盘、IOPS、节点数等）

内部视角

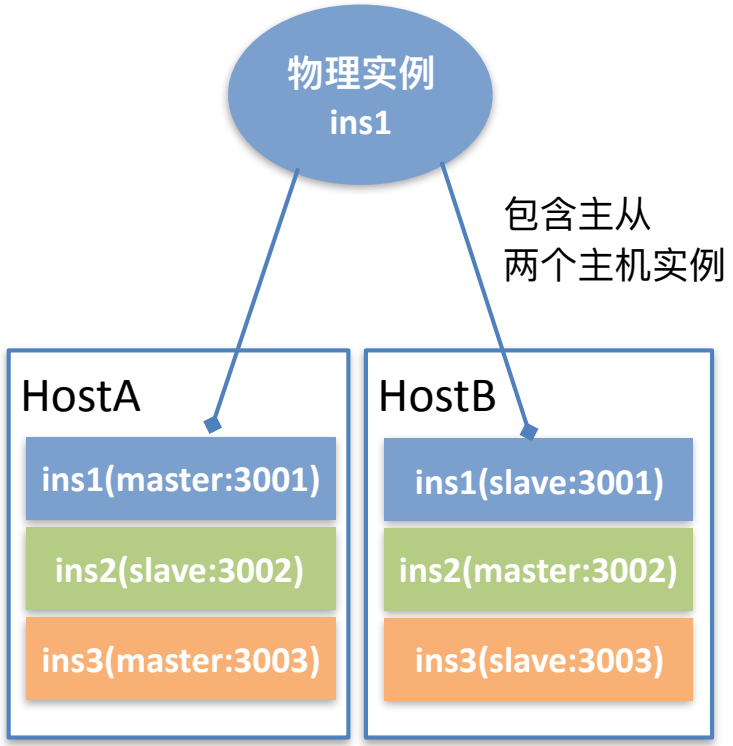
逻辑实例

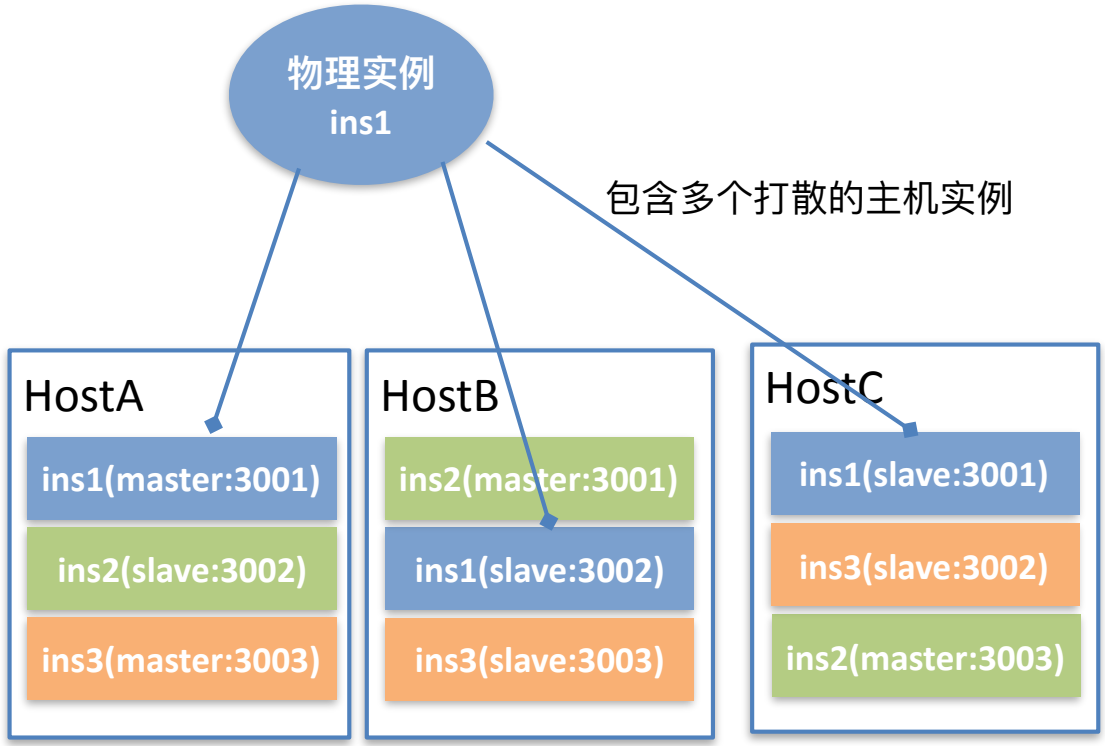
物理实例

主机实例

- 主机实例角色：Master、Slave、Hidden

应用拓扑演进





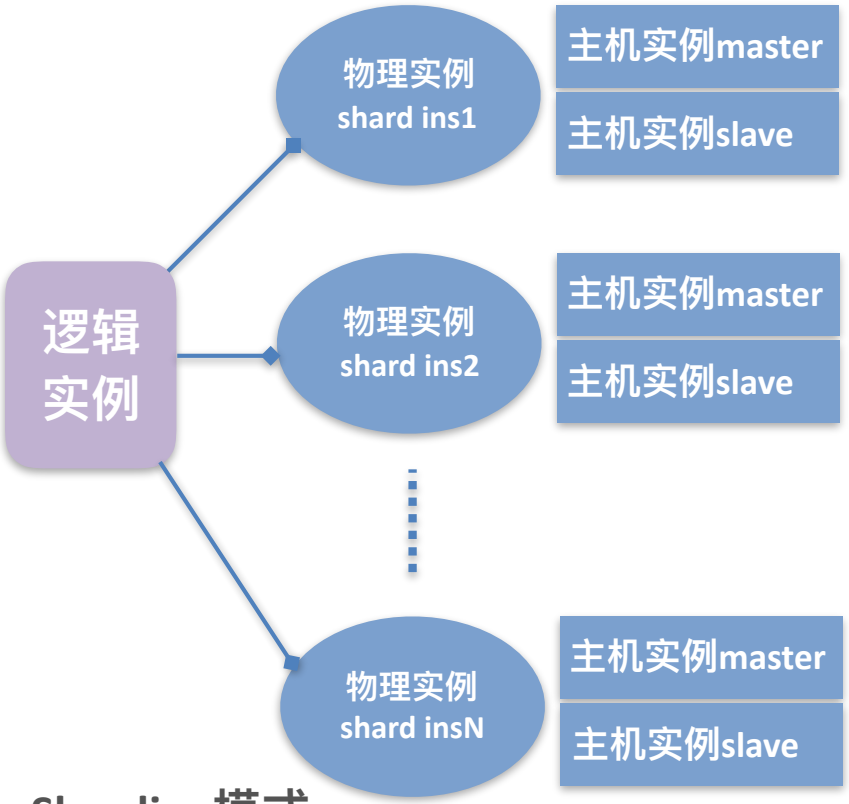
一主一从模式

- 以对机为资源管理单位
- DNS链路，主从实例端口相同，**why?**
- bare metal / localdisk时代，CGroup
- 产品：MySQL / SqlServer / PGSQL / Redis

一主多从模式

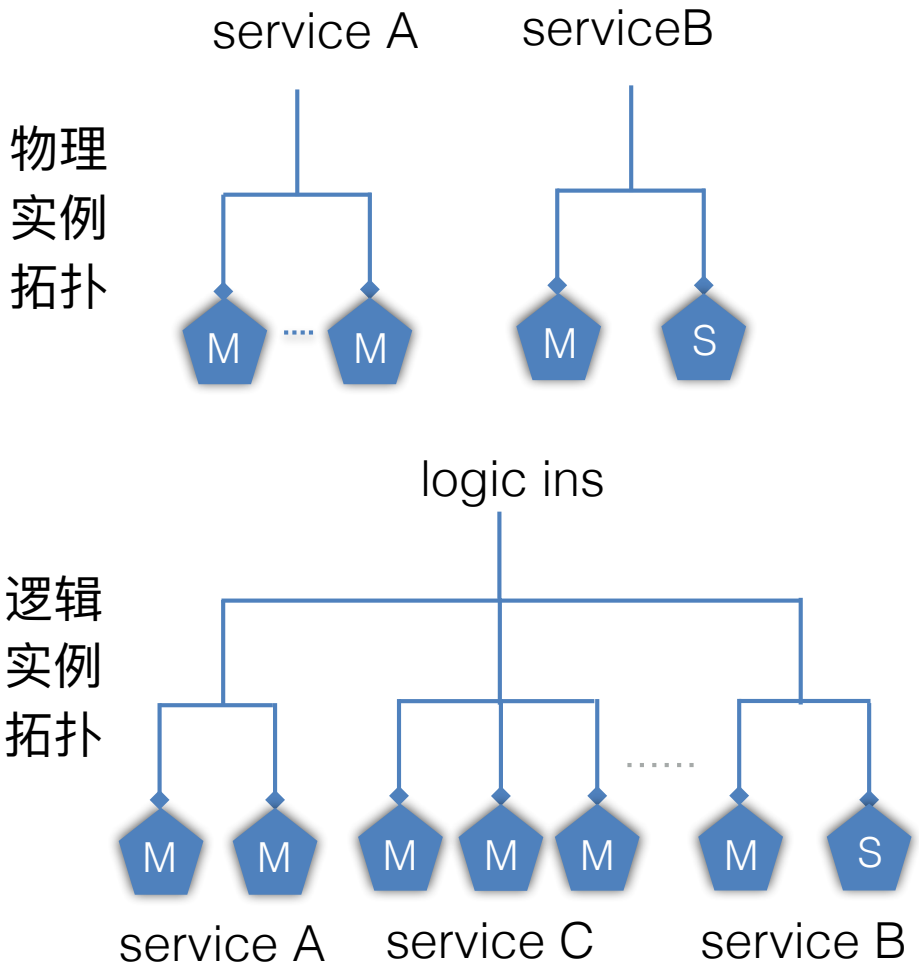
- 不再以对机为资源管理单位，实例全部打散
- 魔方模式，支持异构机型
- SLB链路，主从实例端口可以不同
- bare metal / localdisk时代，CGroup
- 产品：MySQL三节点，MongoDB三节点

应用拓扑演进



Sharding模式

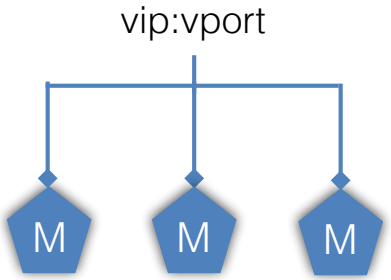
- 引入逻辑实例，三层结构
- 每个分片一个物理实例，同构
- 一个物理实例包含主备主机实例
- 产品：PetaData / Redis Sharding / MongoDB Sharding



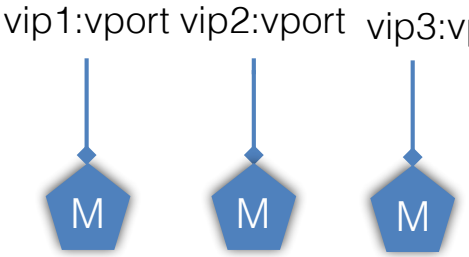
通用集群模式

- 灵活定义，自由组合，异构，覆盖所有场景
- 产品：HBase / HiTSDB / PolarDB / kafka / Flink

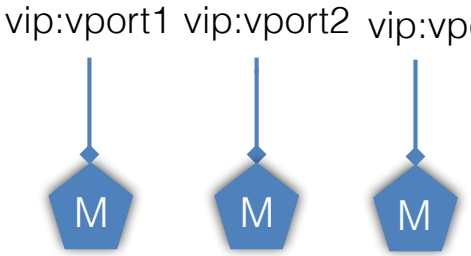
应用拓扑演进-服务发现



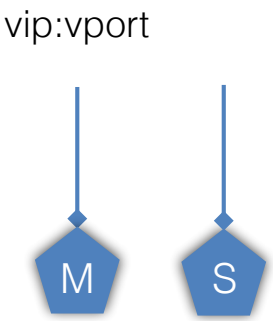
1vip 1vport

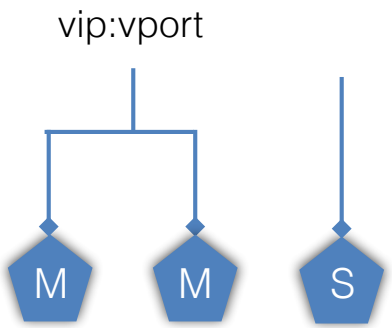


多vip同vport

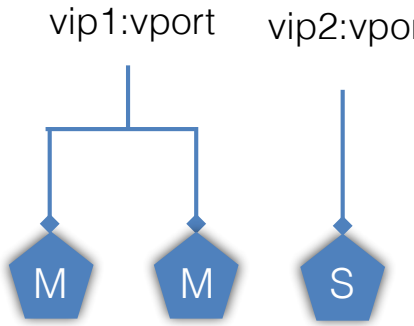


1vip多vport





按角色决定是否绑定vip



按角色绑定vip

覆盖应用
各种vip绑定需求

资源分配与调度

分配维度

- Resource： CPU、Memory、Disk IOPS、Disk Quota、Net IO、QPS、Max Connections； Port、VIP
- Location：单元、可用区、集群、机房、机柜、机器

分配算法

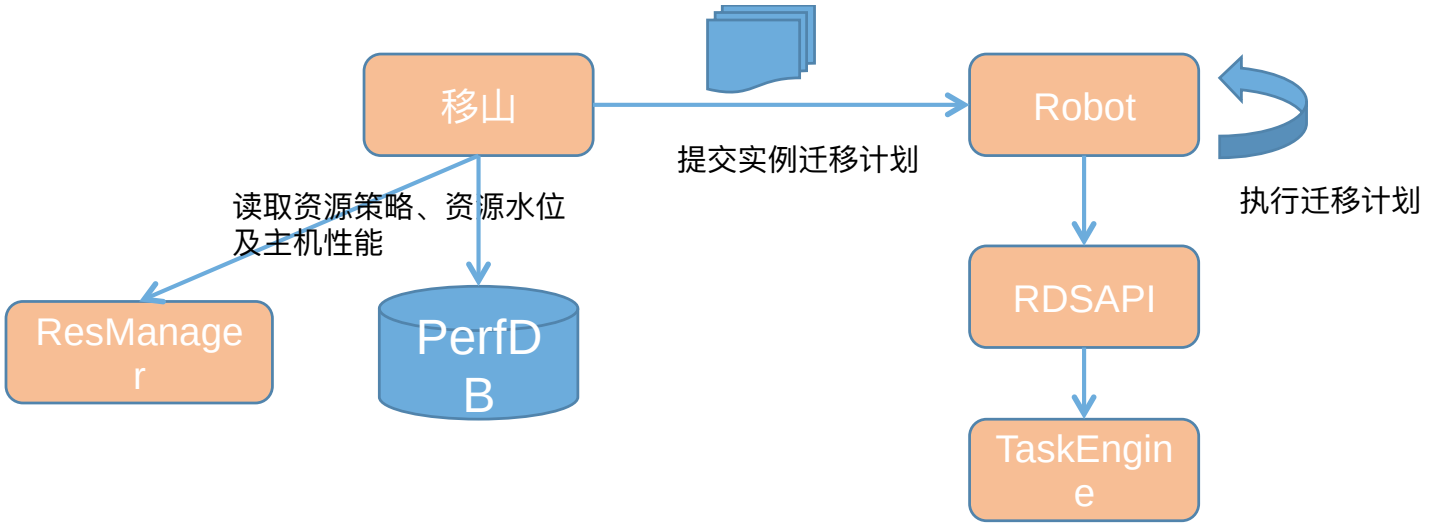
- 堆叠模式
- 分散模式
- 预留分散模式：集群中预留出指定比例空闲资源，分散模式优先从非预留池子中分配，当无法分配时，开始从预留池子中分配。why?
- 伴随模式

- 悲观锁 or 乐观锁?
- 共享 or 独占?

资源分配与调度

动态调度

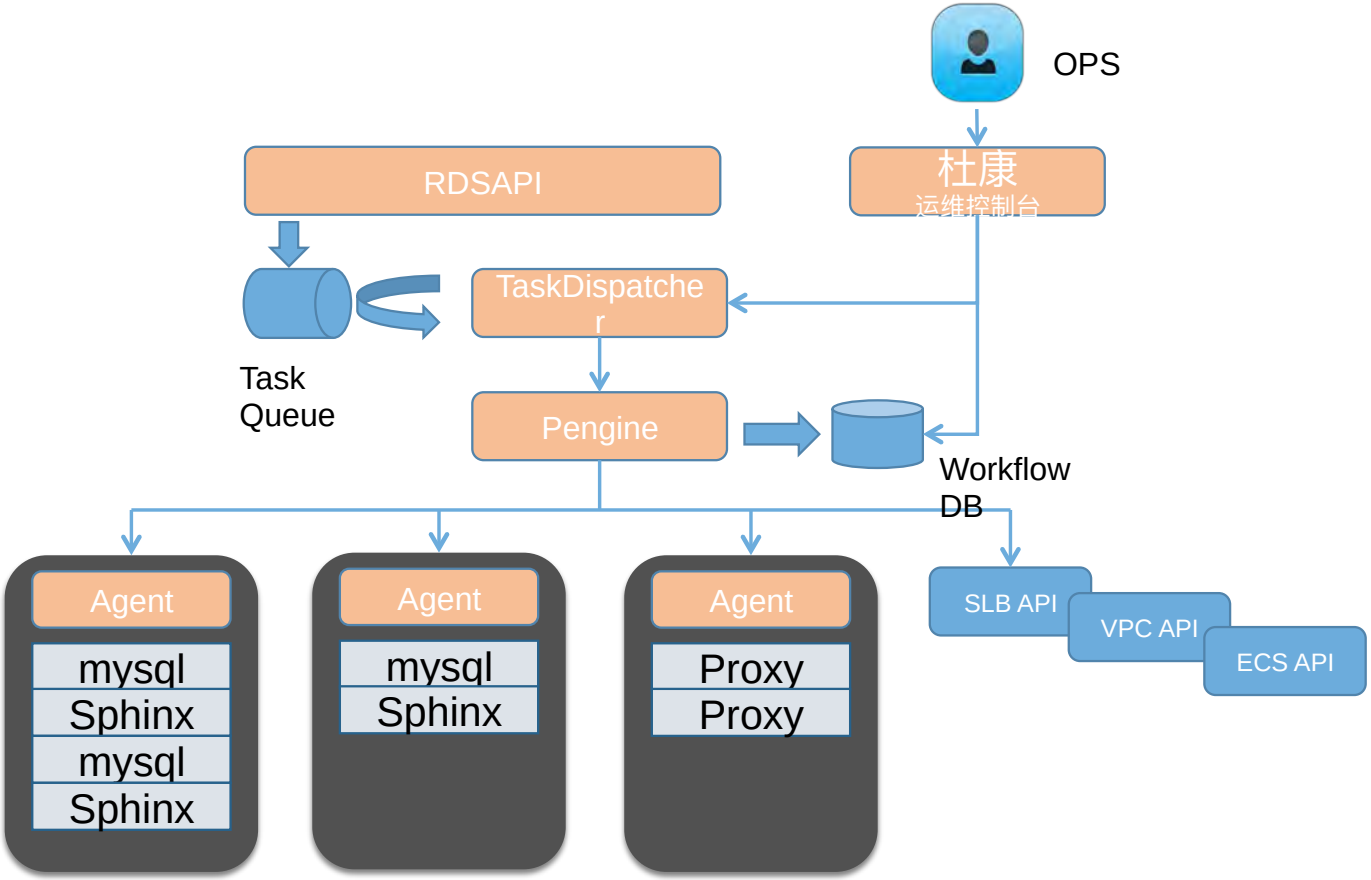
- **必要性**
  - 资源水位、主机负载随着资源的使用动态变化，比如磁盘空间；
  - 资源策略的变化资源要重新调度(如双十一前后)；
  - 资源碎片整合
- 依据大量历史性能数据分析，给出合理的迁移方案



任务系统-基于工作流

Why?

- 大量长时间业务，进展需要可视
- 云环境的复杂性，各种失败和异常是常态，需要重试甚至人工干预



任务系统-基于工作流

- 设计原则：
  - **异步**：用户控制台一切控制操作皆产生一个后端异步任务，一个任务可能涉及到在多个节点上操作，包括安装、配置、启动、变更元数据、变更链路等操作；
  - **可重试**：任务中任意步骤可能失败，失败后保留上下文，可重试
  - **幂等**：保证每个步骤的幂等性，以支持重试
  - **并行**：同一实例任务串行执行，不同实例间任务并发
  - **可暂停 / 回滚 / 取消**
  - **性能**：并行的容量，可横向扩展，**熔断机制**
  - **可运维性**：内部运维控制台
  - **一致性巡检**：元数据与节点出现不一致的报警与修正

可用性检测

- 容器级别 —标准接口
- 链路级别 —标准接口
- 实例级别 —实例容器提供health-check工具镜像

新接入engine无需新开发state job
由应用提供的仅仅是一个health-check镜像

有状态服务的自动恢复

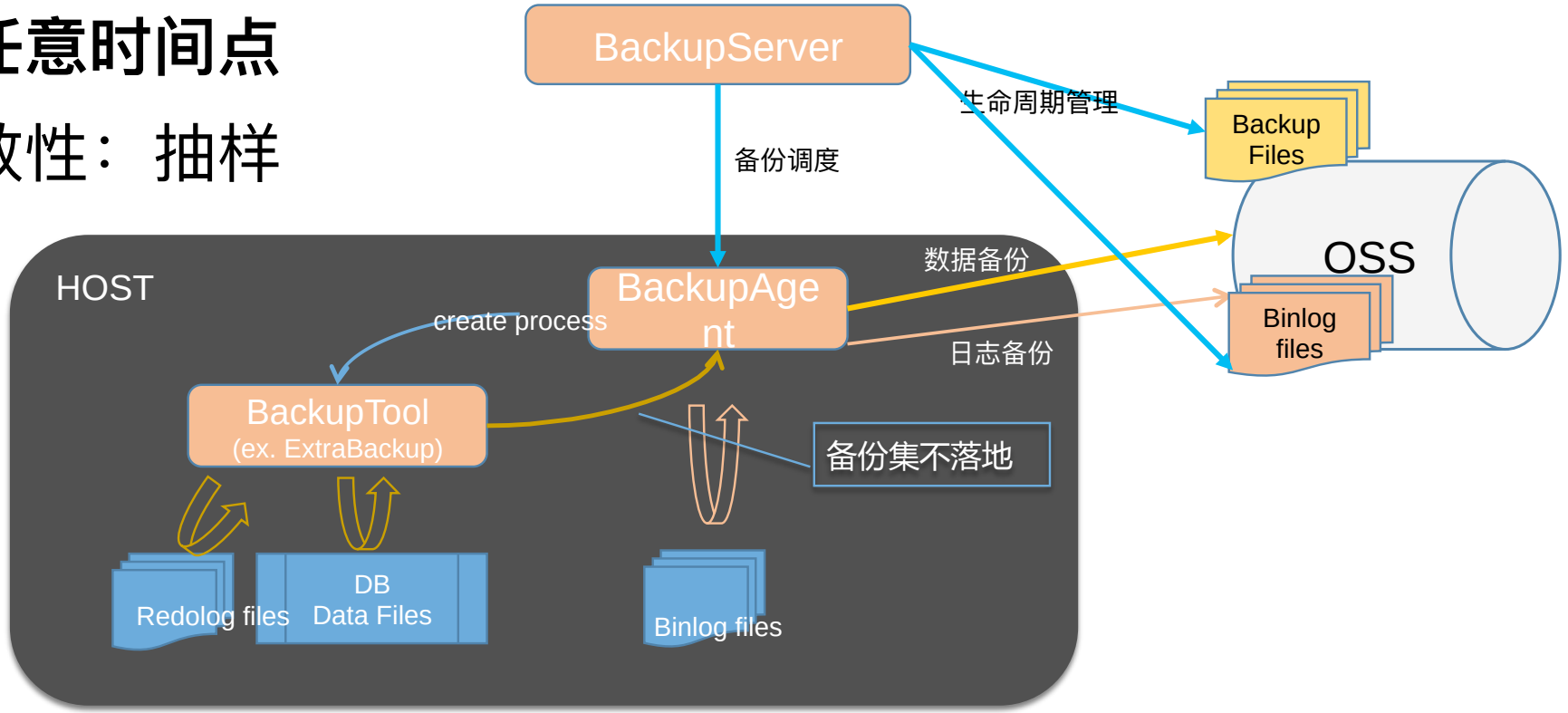
- 故障节点检测：实时检测与深度分析相结合
- 故障节点切换：如何保证数据不丢失?
- 可用性：任何控制操作都要考虑当前的服务是不能中断的；proxy的引入；

弹性扩缩容

- 本地升降级：本地资源充足时，速度快
- 跨机升降级：本地资源不足时，速度慢，涉及到数据搬迁
- 闪断问题：引入proxy，连接保持技术，切换无感知
- 可运维时间：用户自定义

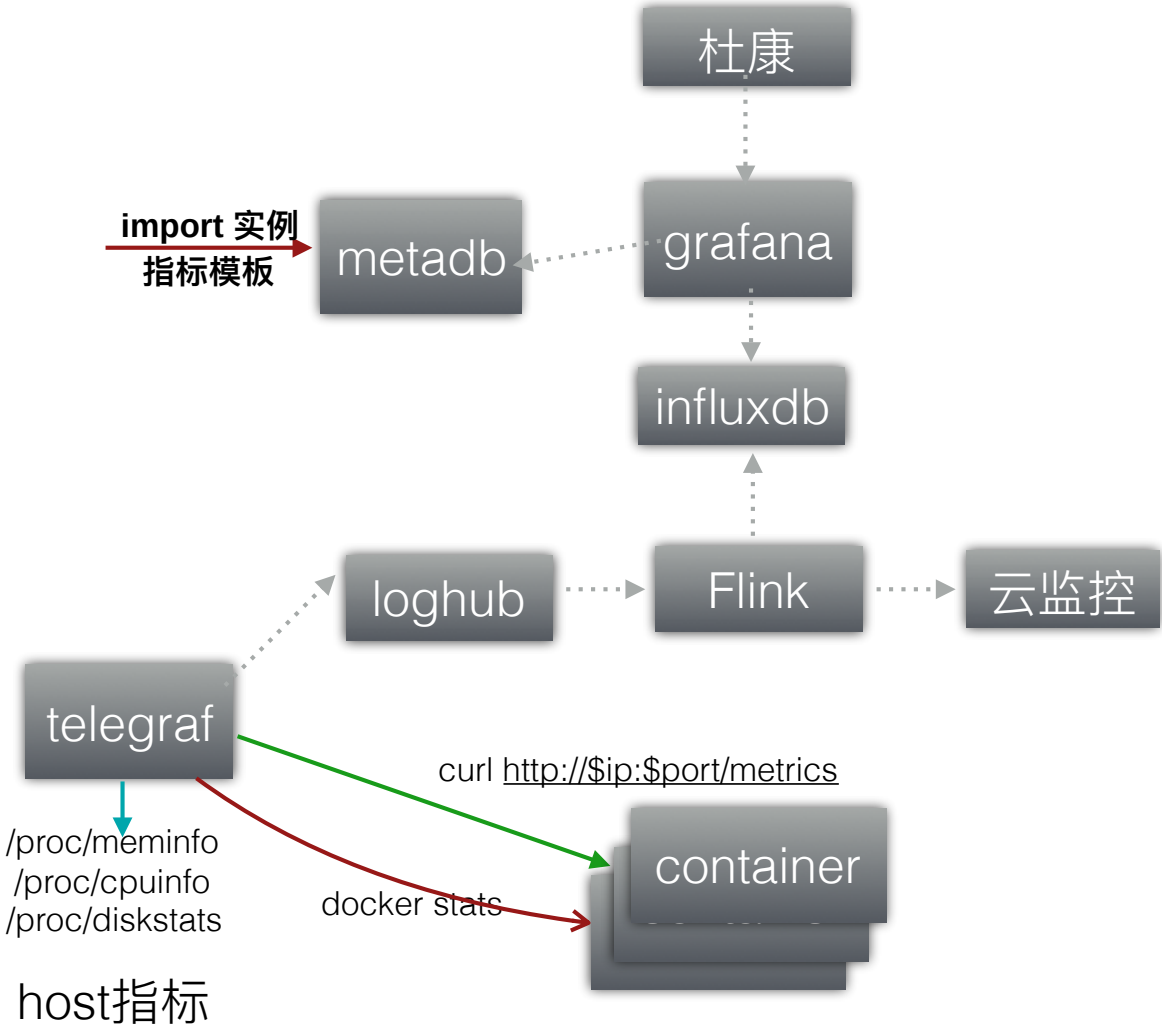
备份和恢复

- 增量和全量备份
- 备份策略和调度
- **还原到任意时间点**
- 备份有效性：抽样



性能监控

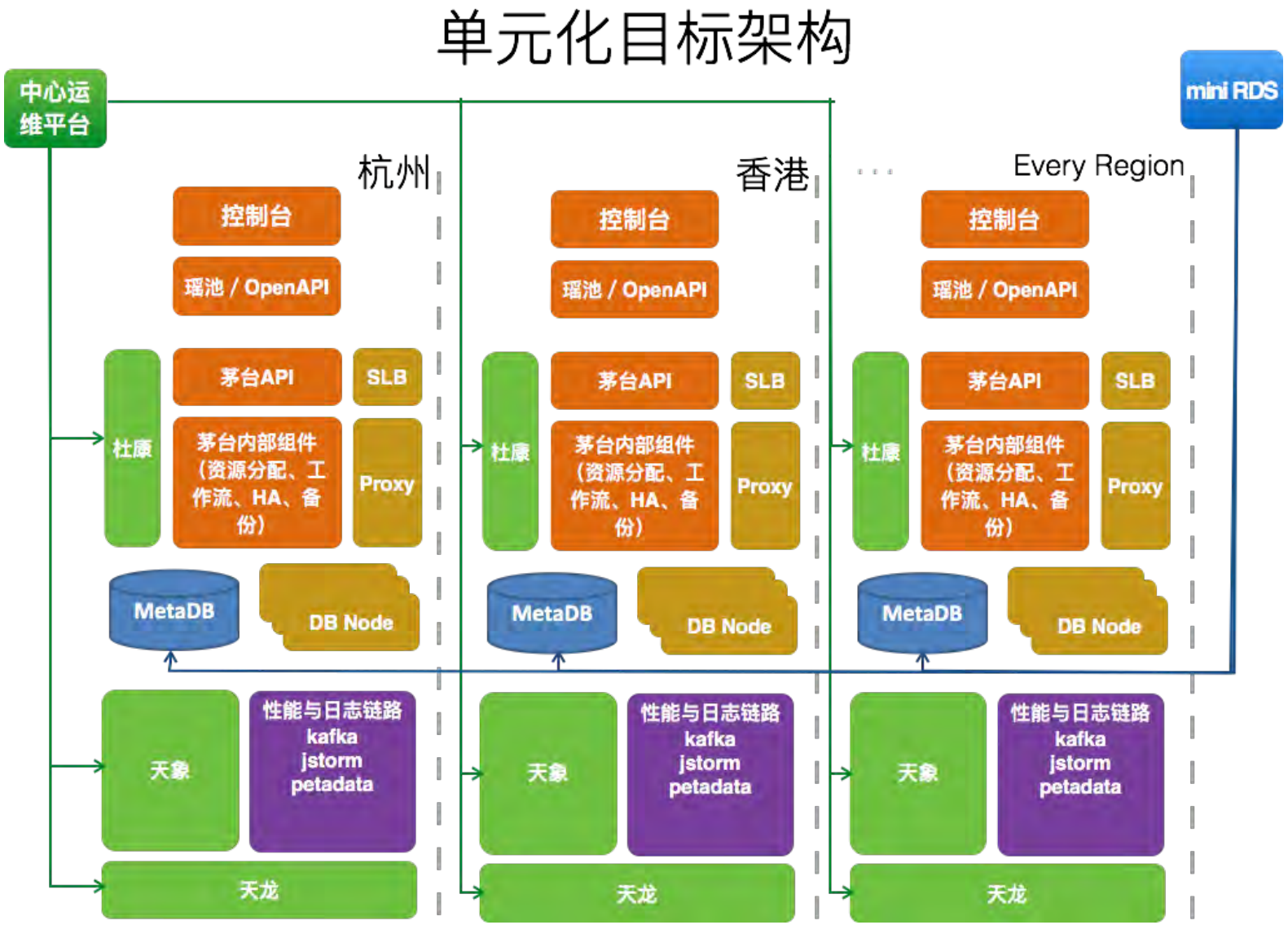
- 支持主机，容器，app指标监控
- 容器label判断是否需要采集，Dockerfile内定义
- 容器元数据由管控在容器上打标（客户实例id，实例名，主机实例id）
- 应用展示模板：开发调试好grafna模板->导出为json->随引擎dsl导入metadb



展示什么指标
展示样式均由应用自定义
平台不涉及元数据库及代码修改

部署：中心化到单元化

- 一个中心，到多个单元
- 全球化，合规
- 规模高速增长，如何降低运维成本?

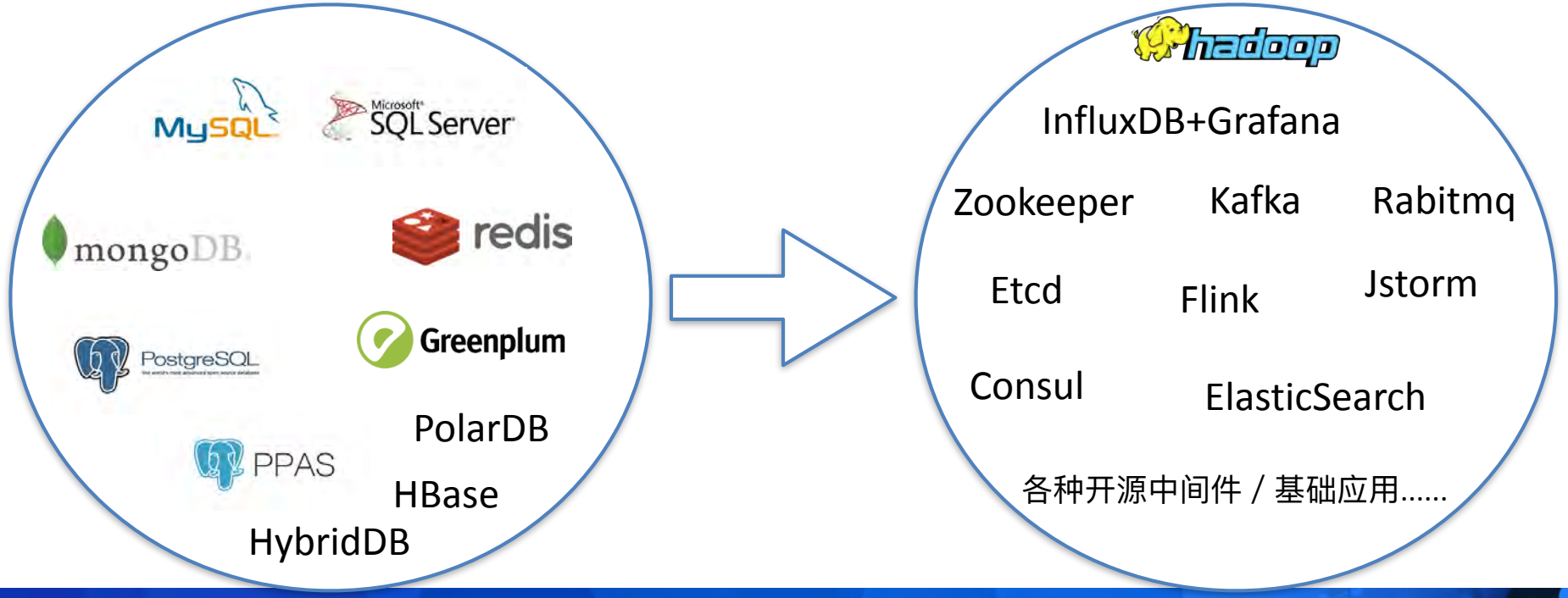


一张图诠释什么是devops

DevOps实践： 标准化->自动化->智能化

从DBaaS到AaaS：CaaS还是FaaS？都不是

- 数据库服务都能做好，那其他应用呢?
- AaaS，App as a Service，是一个应用快速服务化、云化的PaaS平台
- 大规模的实例化能力，多租户快速复制生产能力，复用当前基础设施，支撑快速的商业化
- 一套标准的应用孵化流程，覆盖一个应用集群所有控制层面的能力，DO More！对用户来说真正做到免运维
- 我们的路径是相反的：**从stateful到覆盖stateless，从同构集群到异构集群**



AaaS

引擎类型

数据库服务化 -> 通用应用服务化
涵盖stateful和stateless service

中台红利

工作流、资源调度、基础支撑体系

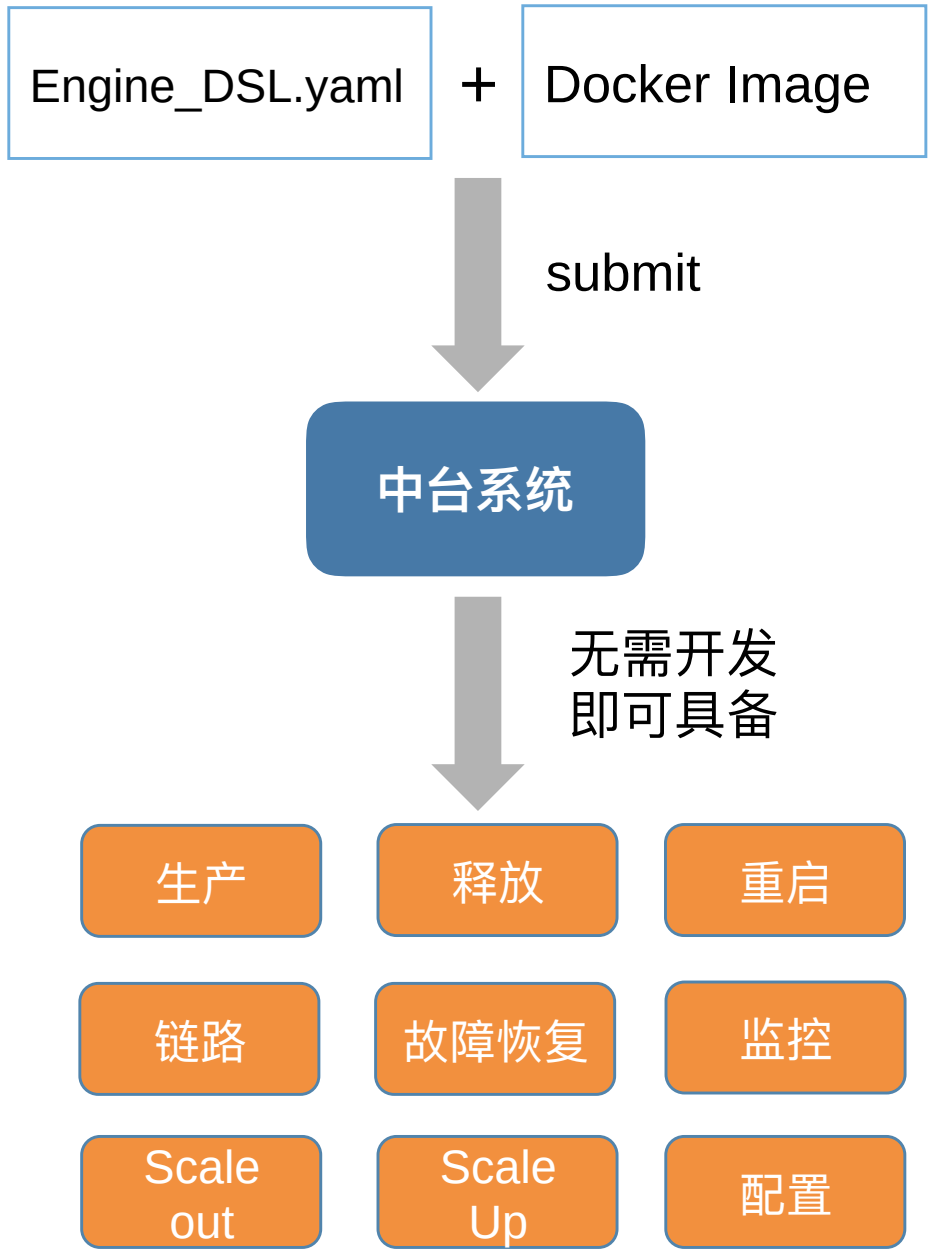
开发标准化

类docker compose语法，更丰富：

- 规格定义：mem/cpu/iops/disk
- 节点拓扑
- 端口分配
- VIP结构
- Volumes挂载
- 多个组件的启动顺序
- 磁盘分配
- 通用运维镜像和操作定义

接入效率

3个月 -> 1-2个星期



AaaS vs. K8S

K8S key features	AaaS
• Automatic binpacking	资源调度有更丰富的维度和策略，并且会依据历史数据进行预测，进行运行时动态调度
• Self-healing	多层次检测；故障恢复，考虑数据可靠性，不丢失
• Horizontal scaling	支持纵向、横向扩容，多维度，按规格付费
• Service discovery and load balancing	基本能力，考虑master / slave模式；专用SLB，带宽保证
• Automated rollouts and rollbacks	滚动升级，多版本共存
• Secret and configuration management	基本能力，支持参数修改，热修改，冷修改
• Storage orchestration	支持localstorage / 云盘 / polarstore，主机磁盘调度
	更多的能力：数据同步、备份与还原；数据库中间件；将数据的调度结合到所有的控制流程中；对数据可靠性追求极致；

邮箱：
alaix.xu@
gmail.com


Gdevops
全球敏捷运维峰会
THANK YOU！