KubeCon CloudNativeCon

THE LINUX FOUNDATION

OPEN SOURCE SUMMIT

AI_dev

Open Source GenAI & ML Summit

China 2024

KubeCon

CloudNativeCon

THE LINUX FOUNDATION
OPEN SOURCE SUMMIT

AI_dev
Open Source GenAI & ML Summit

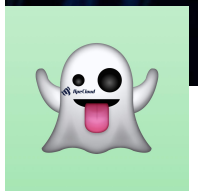
China 2024

快手的 100% 资源利用率提升：从裸机迁移 100K Redis 到 Kubernetes

刘裕惺 快手 高级软件工程师
吴学强 云猿生 研发总监 | KubeBlocks 维护者

关于我

- 前阿里云 PolarDB-X 团队技术 TL
- 开源版 PolarDB-X Maintainer
- 阿里巴巴 2021 年度开源先锋人物
- 目前是云猿生的研发总监、KubeBlocks Maintainer
- 对操作系统、分布式系统、数据库等多个领域感兴趣



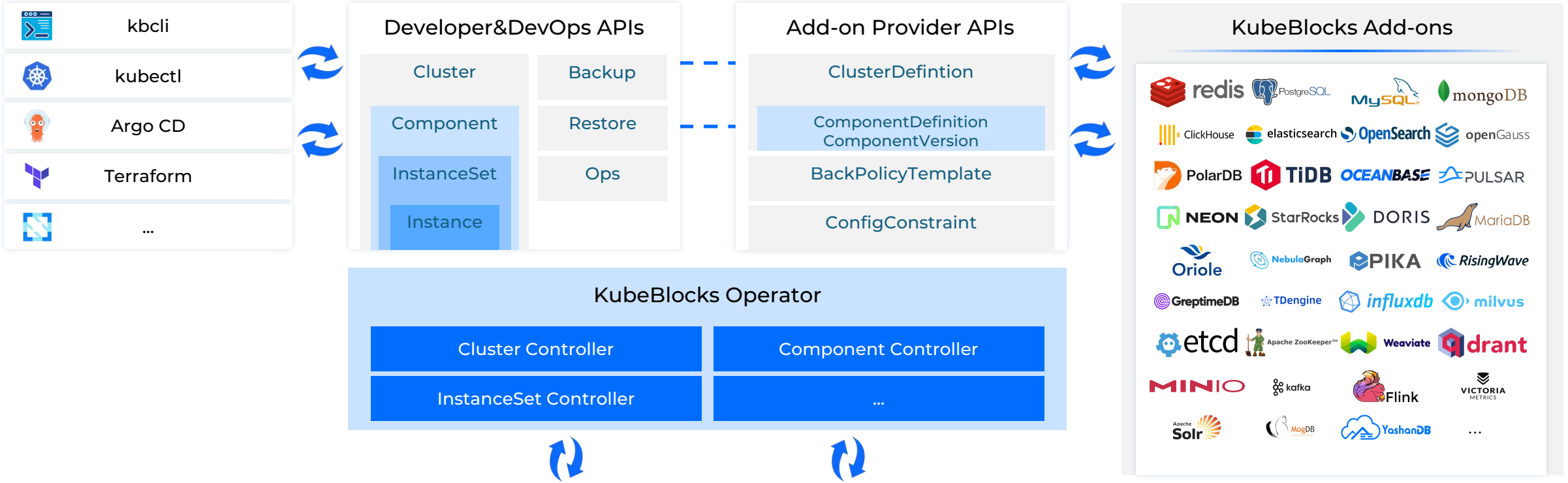
吴学强
GitHub ID: free6om

大纲

- 什么是 KubeBlocks
- KubeBlocks 解决了什么问题（单个 Redis 集群视角）
- 多 Redis 集群和多 K8s 集群（大规模视角）
- Q&A

简介：什么是 KubeBlocks？

什么是 KubeBlocks



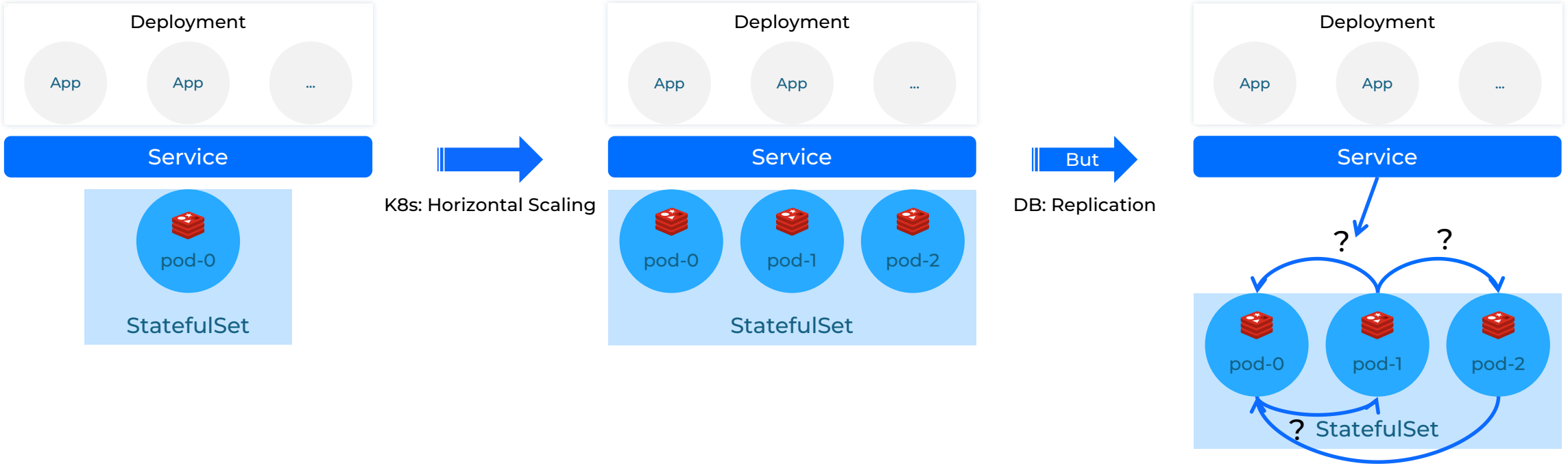
一个专为将数据库运行在 K8s 设计的 Operator

- 支持 35 种数据库
- 可通过 add-on APIs 扩展更多
- 统一的 开发者&DevOps APIs
- 支持数据库生命周期管理
- 支持备份和恢复（包括PITR）
- 已与 Prometheus 和 Grafana 集成
- 更多特性

单个 Redis 集群视角：KubeBlocks 解决了哪些问题从而让数据库更好的运行在 K8s 上？

如何处理数据复制（Replication）

在 K8s 上跑数据库是有非常挑战的



- 吞吐瓶颈
- 单点故障
- 数据丢失风险

- 如何找到具备读写能力的 Pod
- 如何知道并搭建正确的复制关系
- 更新时如何尽量降低对服务的影响
- 。。。。。。

如何处理数据复制（Replication）

角色（Role）：在有状态之上增加的一层新的抽象

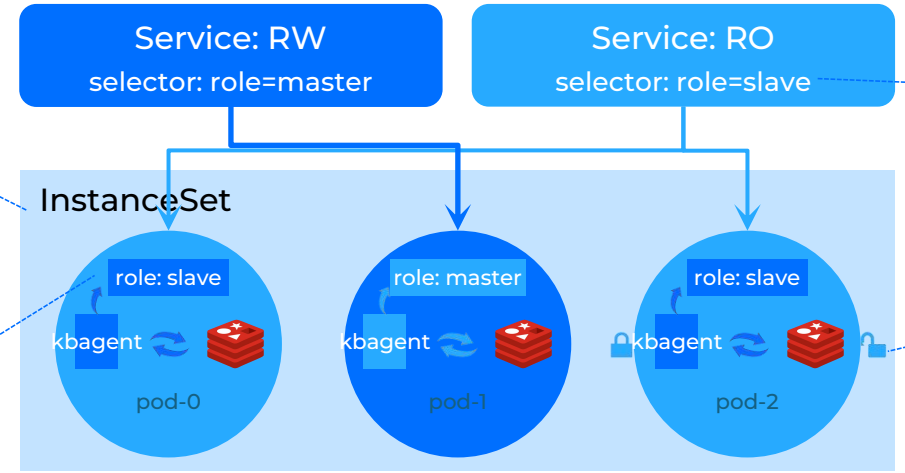
1. InstanceSet API

- 一个专为数据库设计的 Workload API
- 与 StatefulSet类似，每个 Pod 都有固定的网络标识
- 在这之上，增加一层角色抽象

2. 每个 Pod 都有一个角色标签

① 角色周期性探测
② 触发角色事件
③ 观测角色事件
④ 更新 Pod 角色 Label

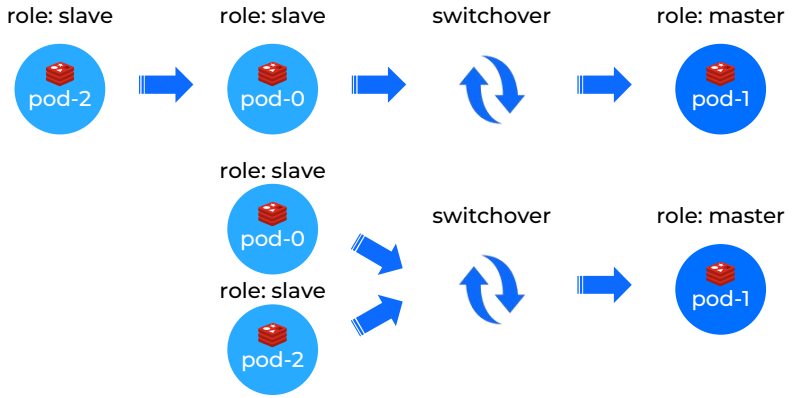
- 一个通用的角色探测框架
- 多种事件发送机制
- 如何处理网络分区
- 如何处理事件延迟



3. 基于角色的 Service Selector

4. 基于角色的成员管理

- 成员家人
- 成员离开
- switchover



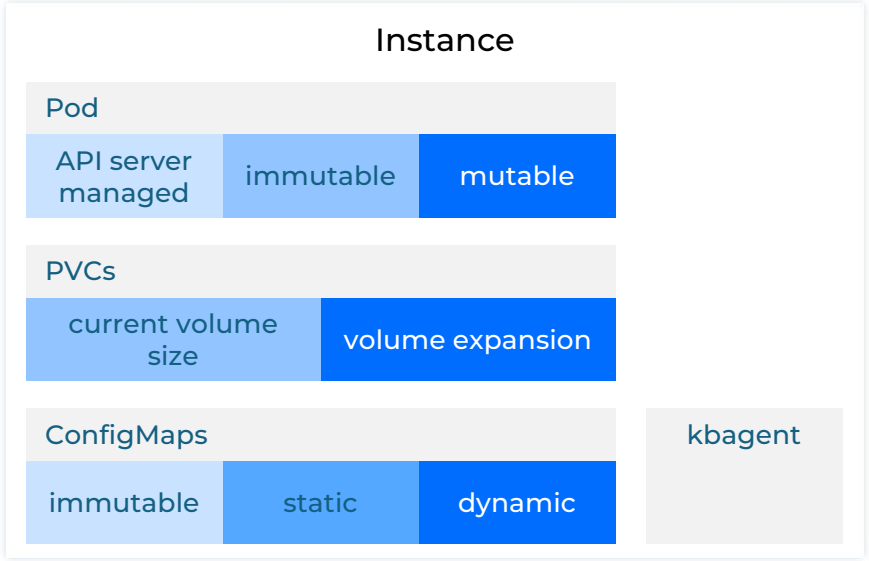
5. 基于角色的更新策略

- Serial, Parallel, BestEffortParallel
- 计划内 switchover

RTO 从分钟级降到秒级

如何实现高可用（HA）

实例原地更新（Instance In-place Update）

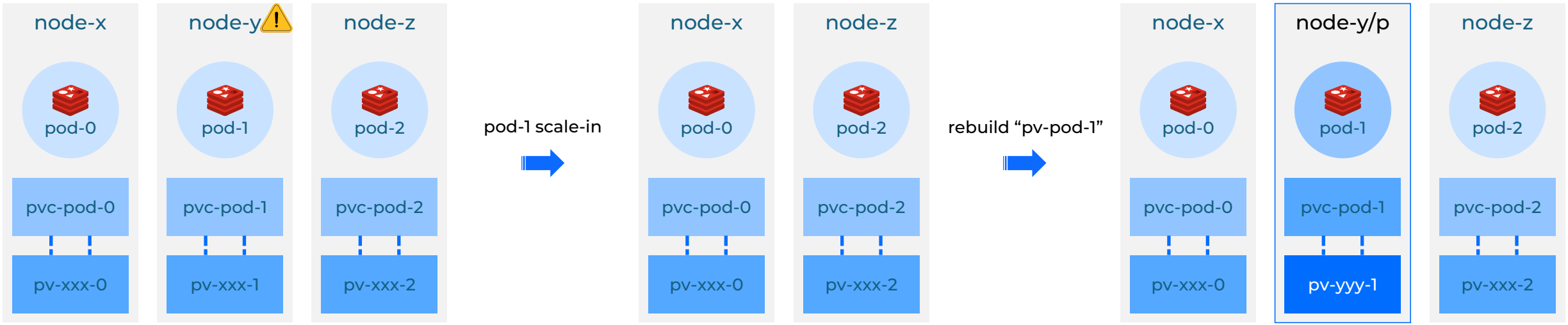


- Pod 原地更新
- Volume 扩缩容
- 配置动态加载

可用性（Uptime） & MTBF 增加
RTO 从秒级降到接近于 0

如何实现高可用（HA）

实例重搭（Instance Rebuilding）



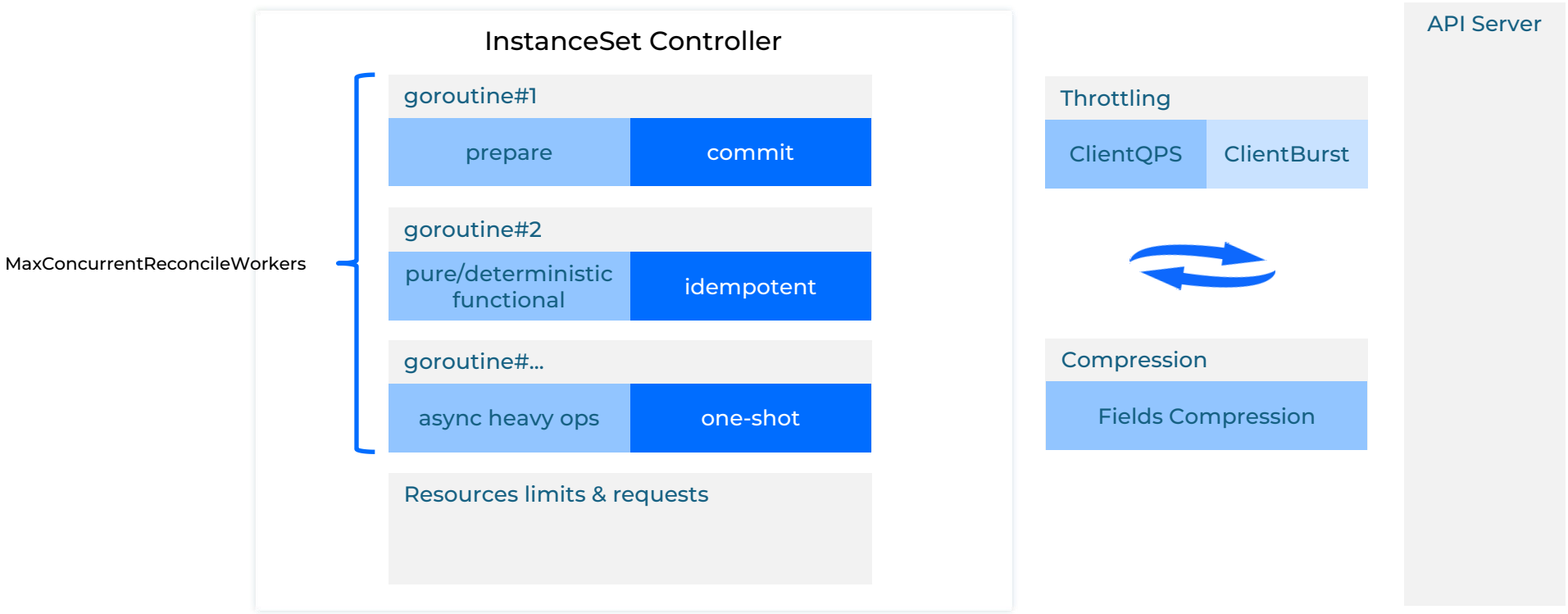
Node node-y 磁盘故障。

node-y 下线，使用指定实例缩容（specified instance scale-in）功能将 pod-1 和 pvc-pod-1 删除。

在新的名字为 node-p 的 Node 上，使用实例重搭（instance rebuilding）功能创建跟老的 PV 包含相同数据的新 PV，即 pv-yyy-1。

RTO 从几小时降到几分钟

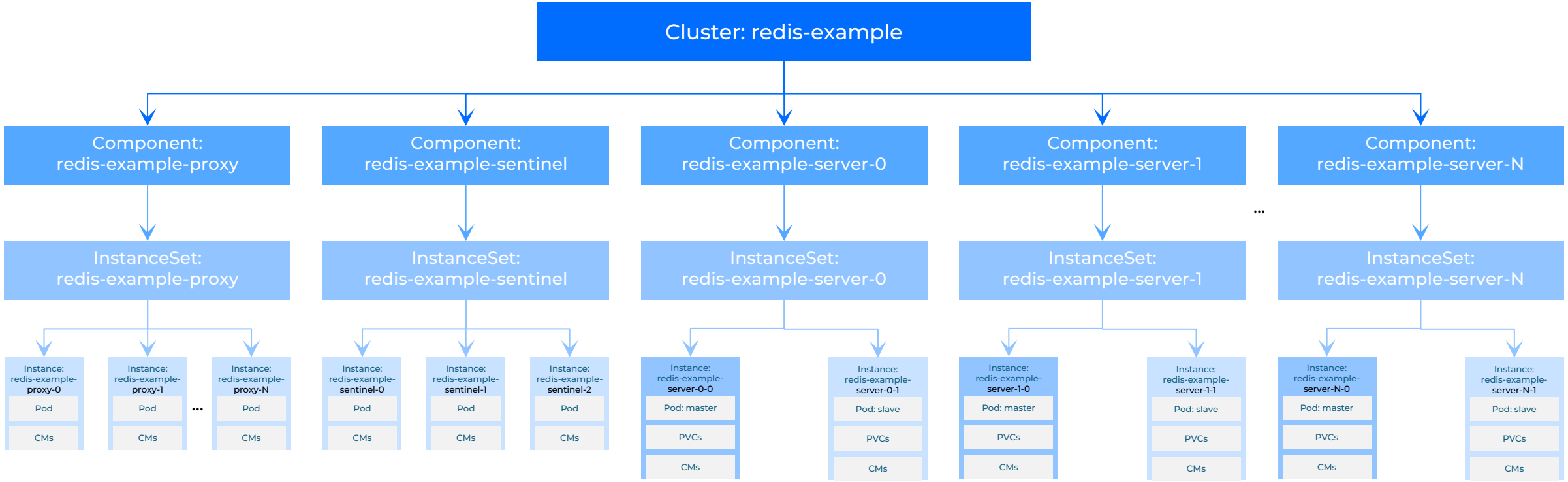
Operator P10K 问题



定义：
与 C10K 问题类似，P10K 问题是指通过优化 Operator 使其能够调谐包含大量 Pod的单个 CR。

场景：
一个 Redis 集群接近 10000 个 Pods

小结



- 角色（Role）
  - 基于角色的 Service Selector
  - 基于角色的成员管理
  - 基于角色的更新策略
- 高可用（HA）
  - Pod 原地更新
  - Volume 扩缩容
  - 配置动态更新
  - 实例重搭（Instance rebuilding）

- Operator P10K 问题
  - 流控
  - 对象压缩
  - 资源限制
  - MaxConcurrenctReconcileWorkers
  - 两阶段调谐
  - 重（heavy）操作异步化

大规模视角：
快手如何借助 KubeBlocks 来将大规模 Redis 实例运行在多 Kubernetes 集群上？

关于我 - 刘裕惺

- 曾就职于阿里云，负责云原生应用分发产品商业化
- 开源&CNCF 项目 Dragonfly 和 Sealer 的 Maintainer
- 目前专注于快手数据库业务的云原生化
- Github ID: starnop

快手 - Redis 介绍

有哪些特点？ - 极端的大规模：

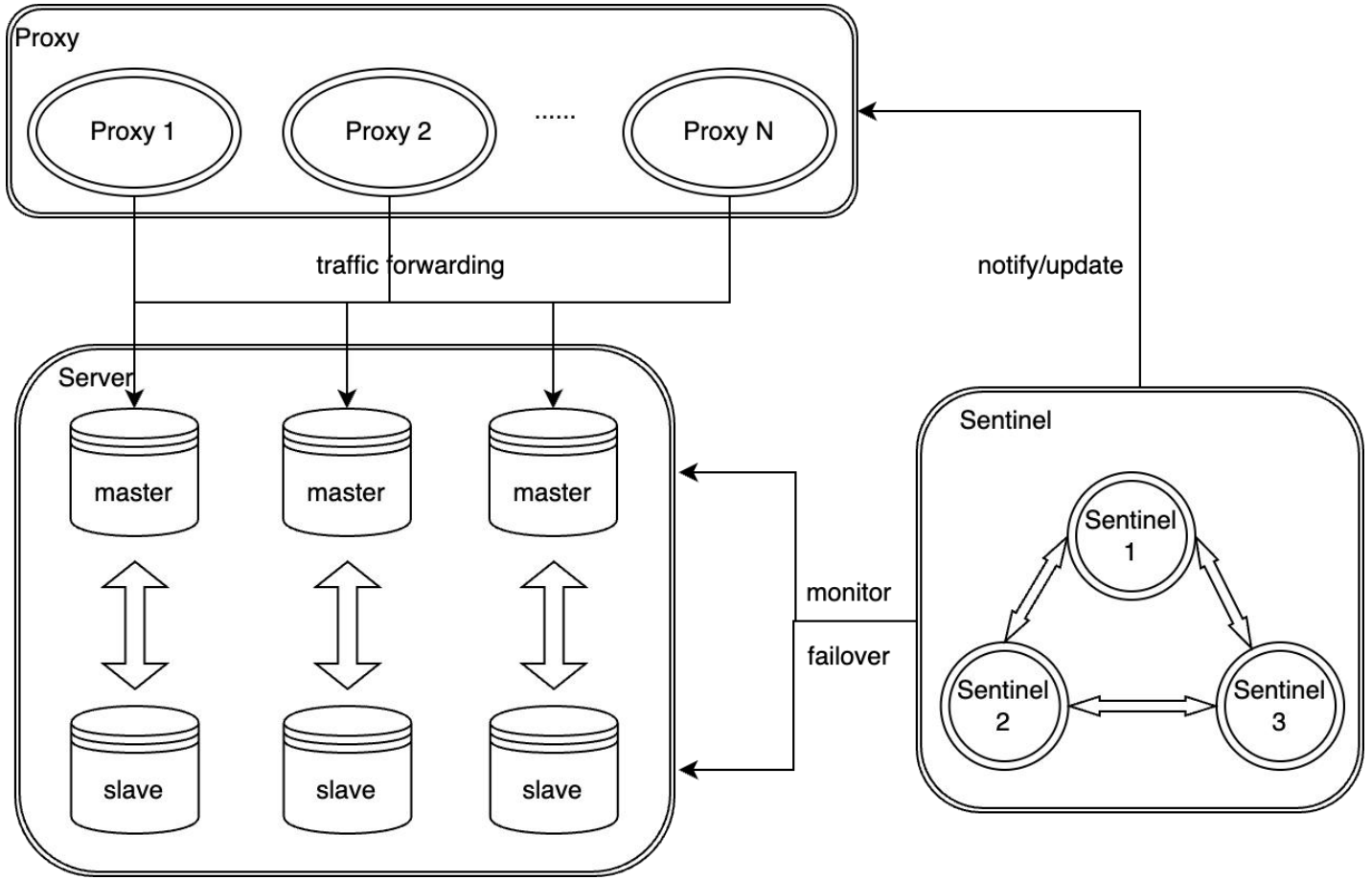
1. 超大规模总实例数
2. 单个 Redis 集群接近 10000 实例

为什么要云原生化？

1. 提高资源利用率并降低资源成本
2. 解耦基础设施与业务从而提高业务迭代敏捷性

组织结构？

1. Redis DBA 团队
2. 云原生团队



为什么选择 KubeBlocks

KubeBlocks - 面向有状态服务的 API

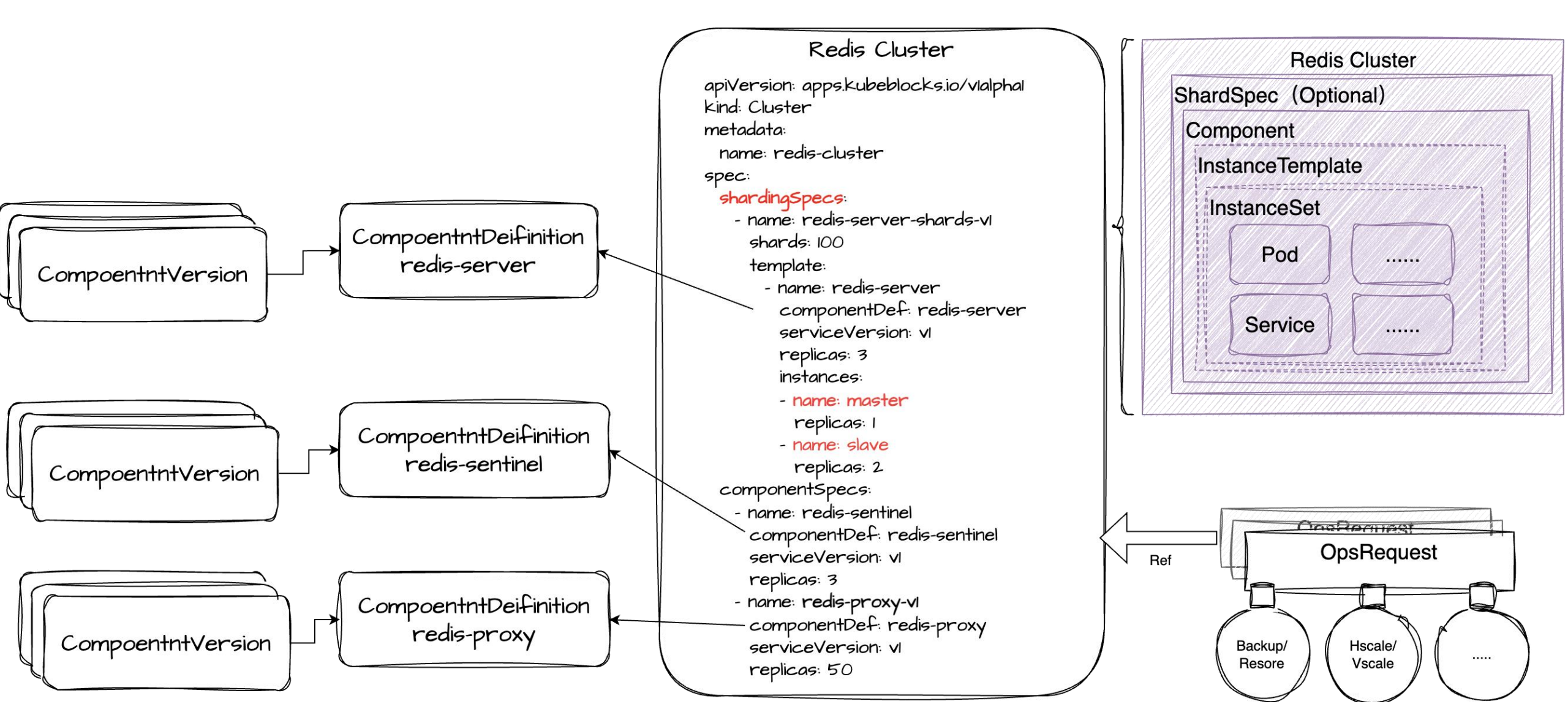
1.强大的功能： 基于角色的管理能力等

2.技术可复用： 支持多种数据库类型

3.云原生化成本低：

   1. 实例生命周期与运维管理分离
   2. 通过 OpsRequest 提供面向过程的 API

Redis 集群编排定义

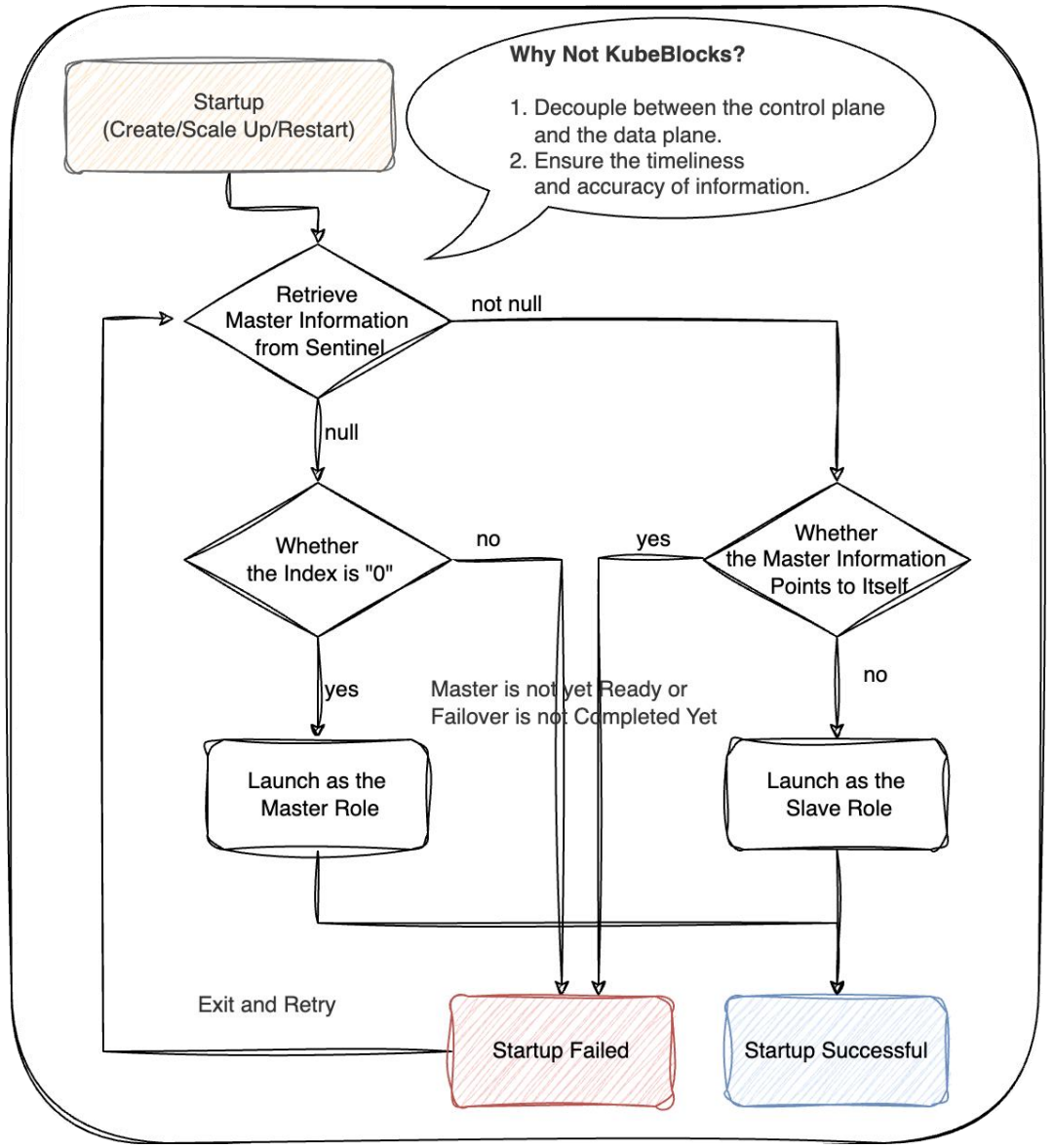


角色（Role）管理

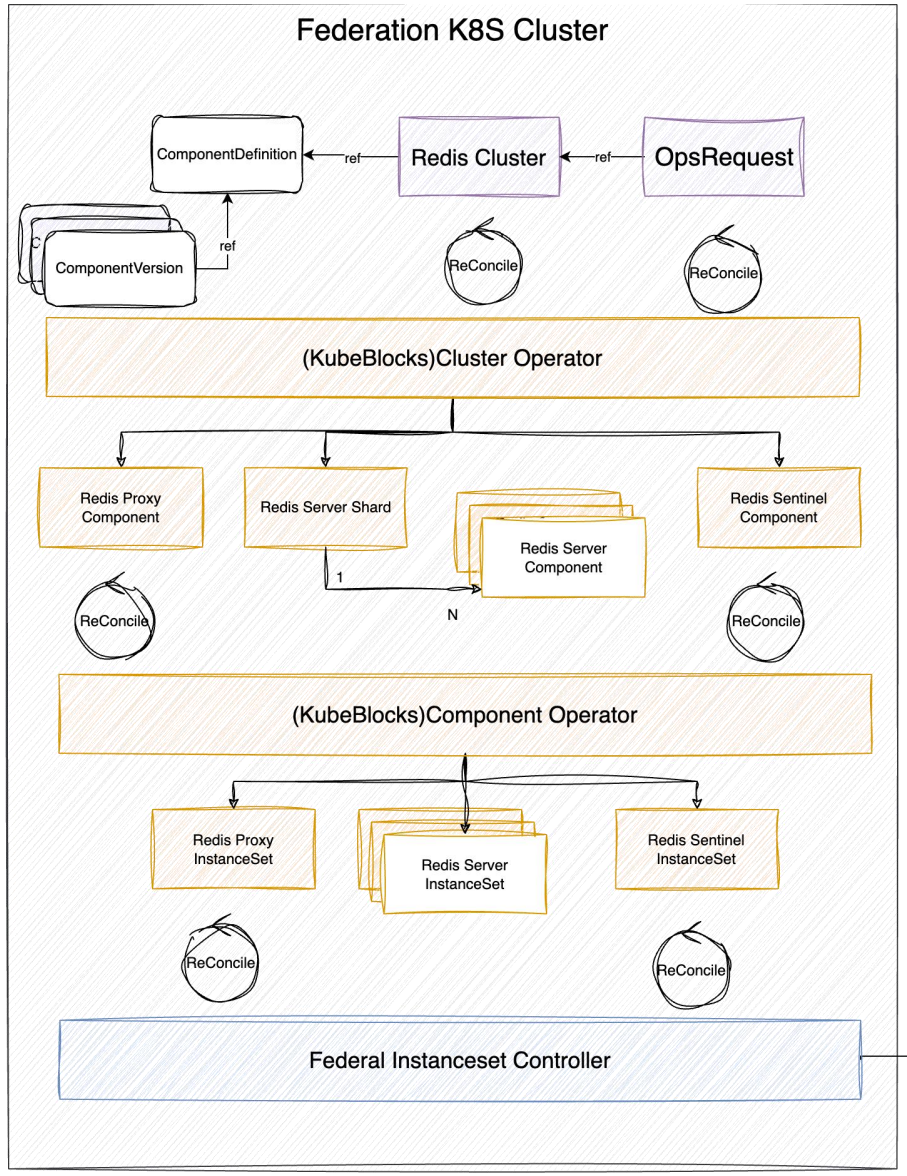
1. How to maintain the correct role relationships?

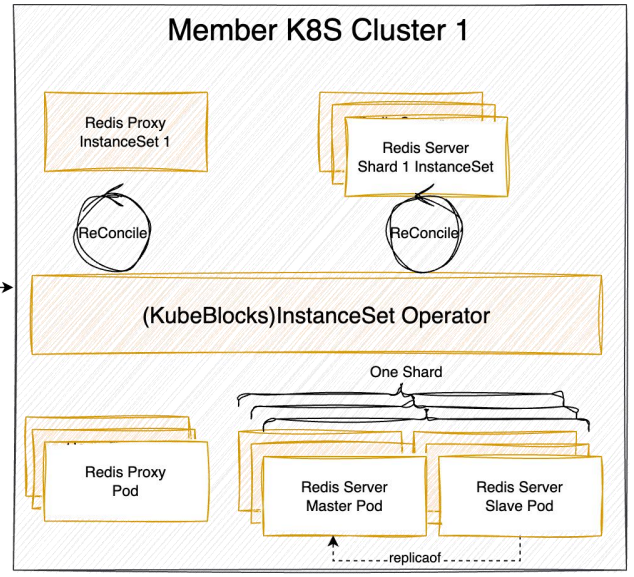
2. How to implement Role-Based O&M Management?

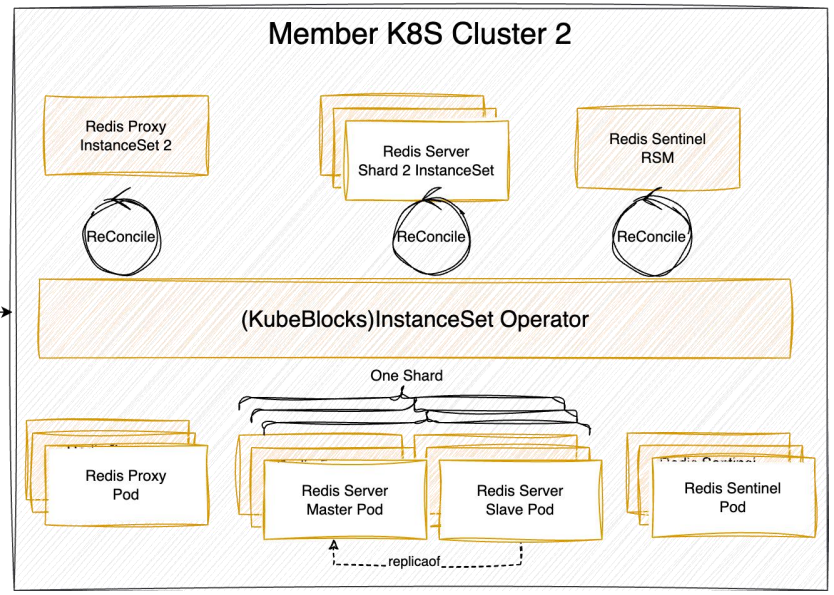
KubeBlocks has implemented it 👍



部署架构

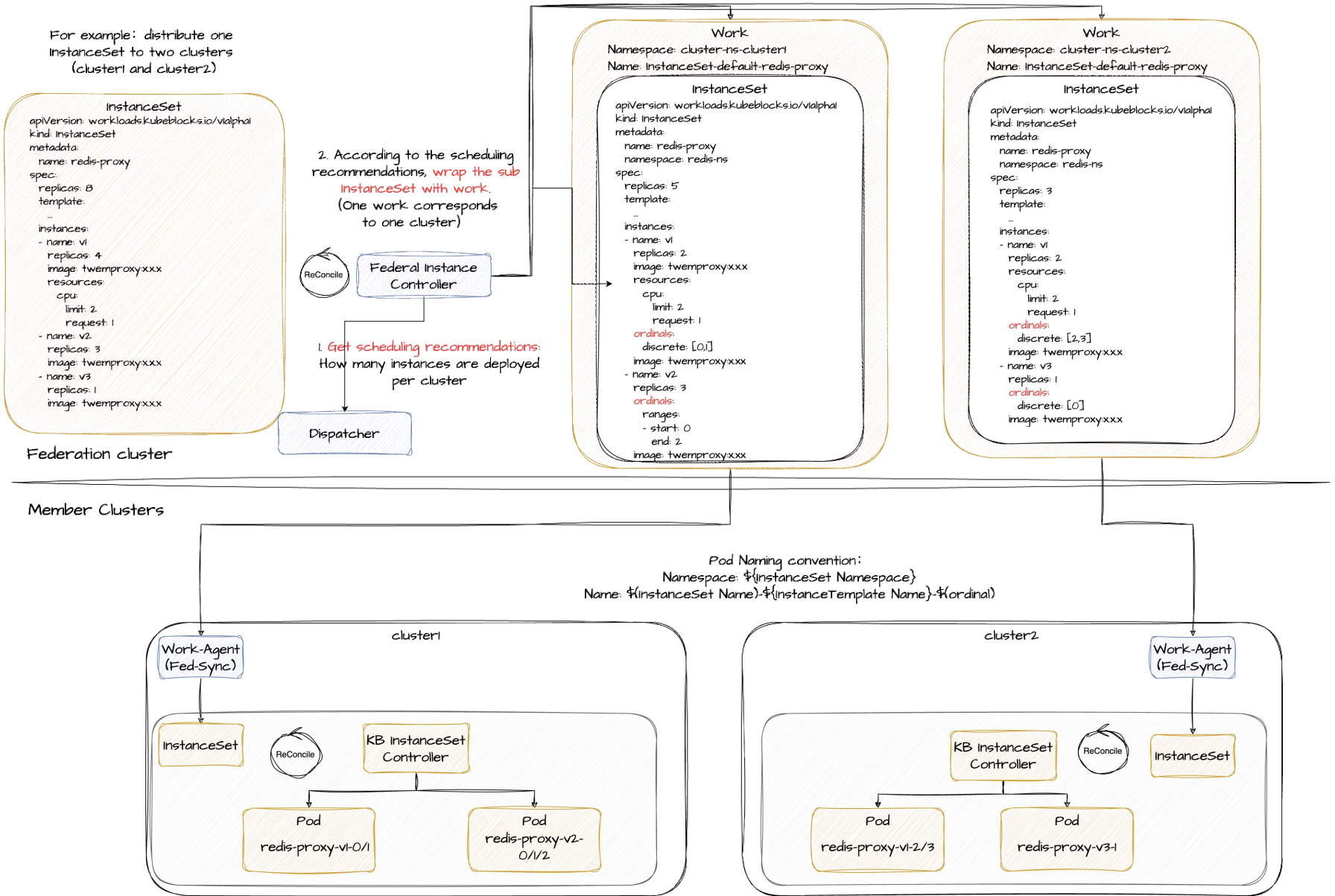




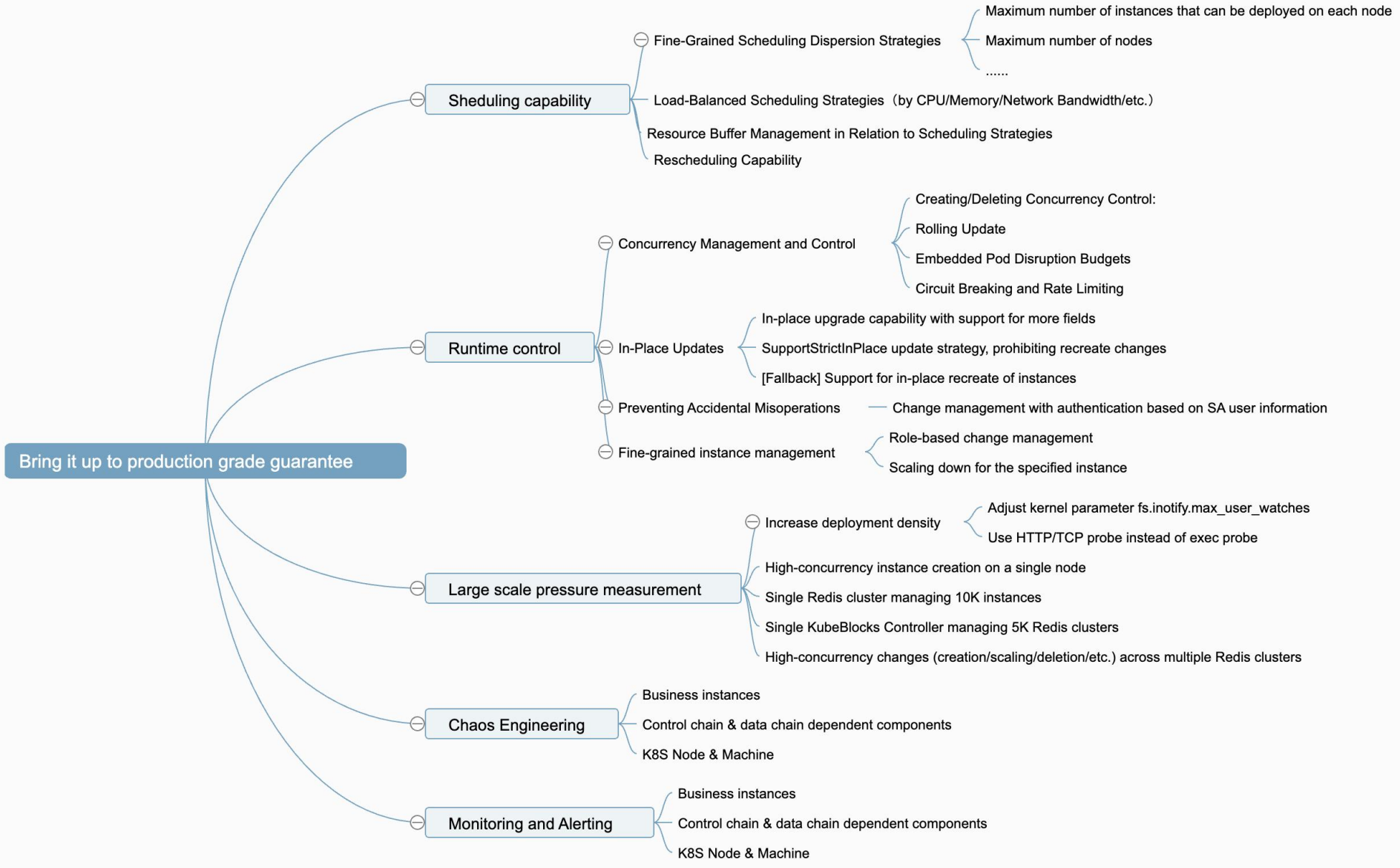


- Cross-Cluster Instance Distribution and Management
- Cross-Cluster Dependency Resource Distribution and Management
- Status Aggregation
- Concurrency Control
- .....

多集群分发管理


For example: distribute one InstanceSet to two clusters (cluster1 and cluster2)
InstanceSet
apiVersion: workloads.kubeblocks.io/v1alpha1
kind: InstanceSet
metadata:
name: redis-proxy
spec:
replicas: 8
template:
...
instances:
- name: v1
replicas: 4
image: twemproxy:xxx
resources:
cpu:
limit: 2
request: 1
- name: v2
replicas: 3
image: twemproxy:xxx
- name: v3
replicas: 1
image: twemproxy:xxx
Federation cluster
2. According to the scheduling recommendations, wrap the sub InstanceSet with work. (One work corresponds to one cluster)
ReConcile
Federal Instance Controller
1. Get scheduling recommendations: How many instances are deployed per cluster
Dispatcher
Work
Namespace: cluster-ns-cluster1
Name: InstanceSet-default-redis-proxy
InstanceSet
apiVersion: workloads.kubeblocks.io/v1alpha1
kind: InstanceSet
metadata:
name: redis-proxy
namespace: redis-ns
spec:
replicas: 5
template:
...
instances:
- name: v1
replicas: 2
image: twemproxy:xxx
resources:
cpu:
limit: 2
request: 1
ordinals:
discrete: [0,1]
image: twemproxy:xxx
- name: v2
replicas: 3
ordinals:
ranges:
- start: 0
end: 2
image: twemproxy:xxx
Work
Namespace: cluster-ns-cluster2
Name: InstanceSet-default-redis-proxy
InstanceSet
apiVersion: workloads.kubeblocks.io/v1alpha1
kind: InstanceSet
metadata:
name: redis-proxy
namespace: redis-ns
spec:
replicas: 3
template:
...
instances:
- name: v1
replicas: 2
resources:
cpu:
limit: 2
request: 1
ordinals:
discrete: [2,3]
image: twemproxy:xxx
- name: v3
replicas: 1
ordinals:
discrete: [0]
image: twemproxy:xxx
Member Clusters
Pod Naming convention:
Namespace: ${InstanceSet Namespace}
Name: ${InstanceSet Name}-${InstanceTemplate Name}-${ordinal}
cluster1
Work-Agent (Fed-Sync)
InstanceSet
ReConcile
KB InstanceSet Controller
Pod
redis-proxy-v1-0/1
Pod
redis-proxy-v2-0/1/2
cluster2
Work-Agent (Fed-Sync)
KB InstanceSet Controller
ReConcile
InstanceSet
Pod
redis-proxy-v1-2/3
Pod
redis-proxy-v3-1

稳定性保障


Bring it up to production grade guarantee
Sheduling capability
Fine-Grained Scheduling Dispersion Strategies
Maximum number of instances that can be deployed on each node
Maximum number of nodes
.....
Load-Balanced Scheduling Strategies（by CPU/Memory/Network Bandwidth/etc.）
Resource Buffer Management in Relation to Scheduling Strategies
Rescheduling Capability
Runtime control
Concurrency Management and Control
Creating/Deleting Concurrency Control:
Rolling Update
Embedded Pod Disruption Budgets
Circuit Breaking and Rate Limiting
In-Place Updates
In-place upgrade capability with support for more fields
SupportStrictInPlace update strategy, prohibiting recreate changes
[Fallback] Support for in-place recreate of instances
Preventing Accidental Misoperations
Change management with authentication based on SA user information
Fine-grained instance management
Role-based change management
Scaling down for the specified instance
Large scale pressure measurement
Increase deployment density
Adjust kernel parameter fs.inotify.max_user_watches
Use HTTP/TCP probe instead of exec probe
High-concurrency instance creation on a single node
Single Redis cluster managing 10K instances
Single KubeBlocks Controller managing 5K Redis clusters
High-concurrency changes (creation/scaling/deletion/etc.) across multiple Redis clusters
Chaos Engineering
Business instances
Control chain & data chain dependent components
K8S Node & Machine
Monitoring and Alerting
Business instances
Control chain & data chain dependent components
K8S Node & Machine

关于 KubeBlocks 的一些思考

The control plane for your cloud-native data infrastructure

Install, create, connect, and you have it all.

与 KubeBlocks 合作

- InstanceSet 直管 Pod 和 PVC
- 实例模板 Instance Template(former heterogeneous pod)
- 与联邦集群集成
- 并发控制策略（Parallel concurrency policy）
- 严格更新策略（Restrict update policy）
- In-place update 和 in-place vertical scaling
- 指定实例下线
- 大规模/高并发场景性能优化
- ...

微信

Slack

加入 KubeBlocks 社区!

https://github.com/apecloud/kubeblocks

快手微信公众号

关注“快手技术”公众号
获取更多技术干货