JDCloud cross-cluster large-scale application management practice

JDCloud

2024

KubeCon CloudNativeCon China 2024

THE LINUX FOUNDATION OPEN SOURCE SUMMIT

AI_dev Open Source GenAI & ML Summit

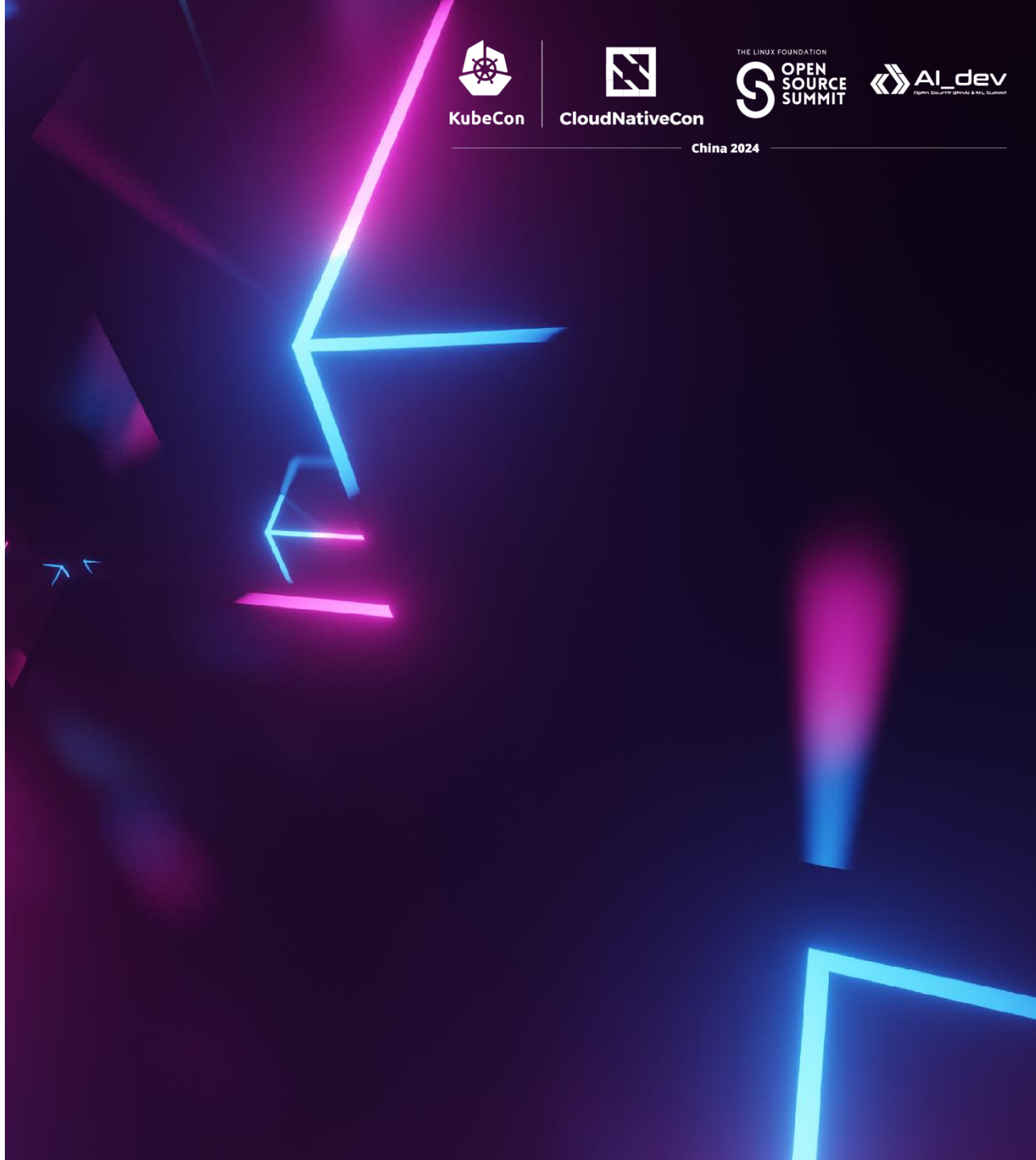
JDCloud

2024

XiaoFei Wang

JDCloud
CloudNativeEngineer
wangxiaofei67@jd.com

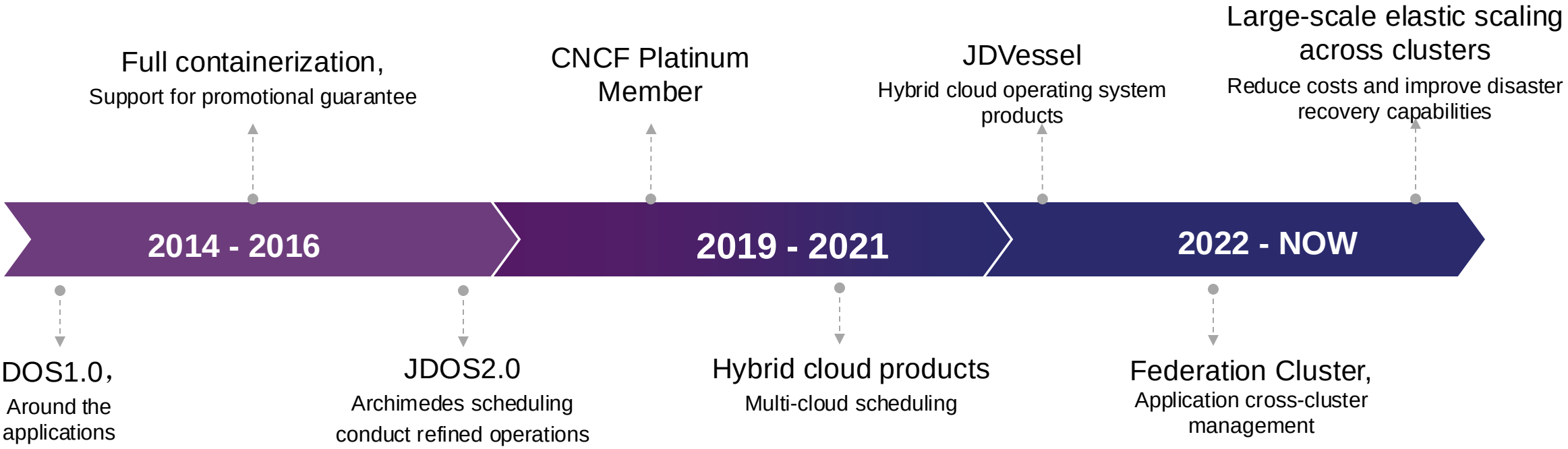
CONTENTS



01

Development Road of JD Cloud Containers

JD Containerization Development History



Earliest implementation
Large-scale hybrid cloud container cluster.

The containerization is the most thorough.
Application middleware, database, big data, mixed deployment, artificial intelligence.

100% coverage
Carrying the business of JD Retail, Logistics, Finance, Health, Industry and Insurance.

Efficiency improvement.
Achieve the optimal solution of efficiency, cost and elasticity in multi-cluster management.

Everything around the application.

Single-Cluster limitation

- Fault explosion radius is too large, affecting the application of SLA.
- The application of cross-cluster auto-scaling is difficult, manual operation and maintenance.

Multi-cluster Scheduling

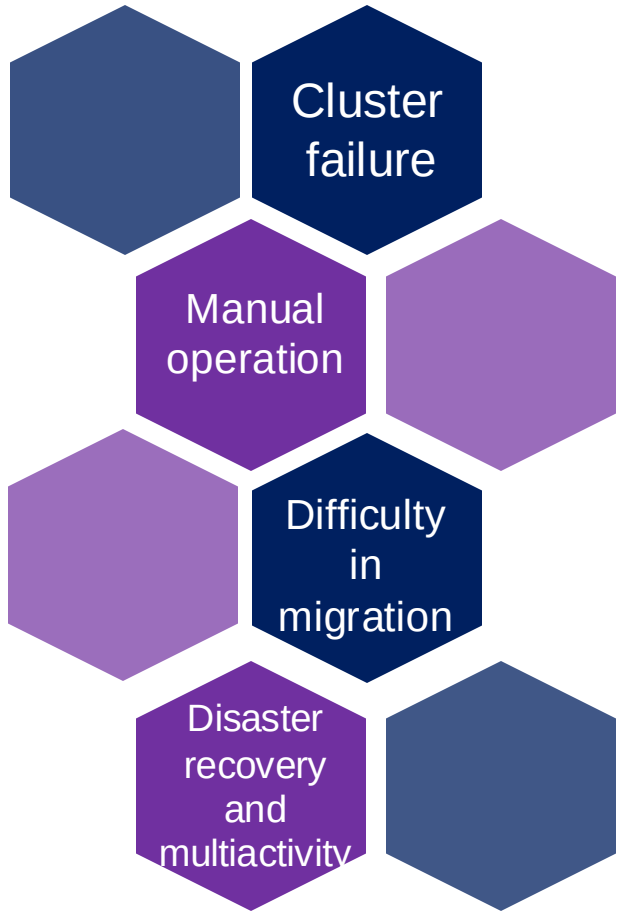
- The Unified resource scheduling
- Multi-group flexible scheduling strategy

Multi-cluster auto-scaling

- Cross-cluster elastic scaling
- One-click migration

Multiple and HA

- Multiple applications
- Cross-cluster network



02

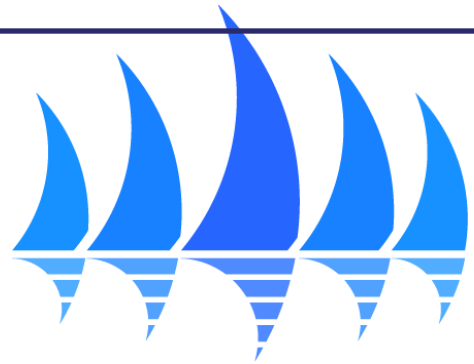
Federation Cluster

Product

京东云

JDCloud·JDVessel

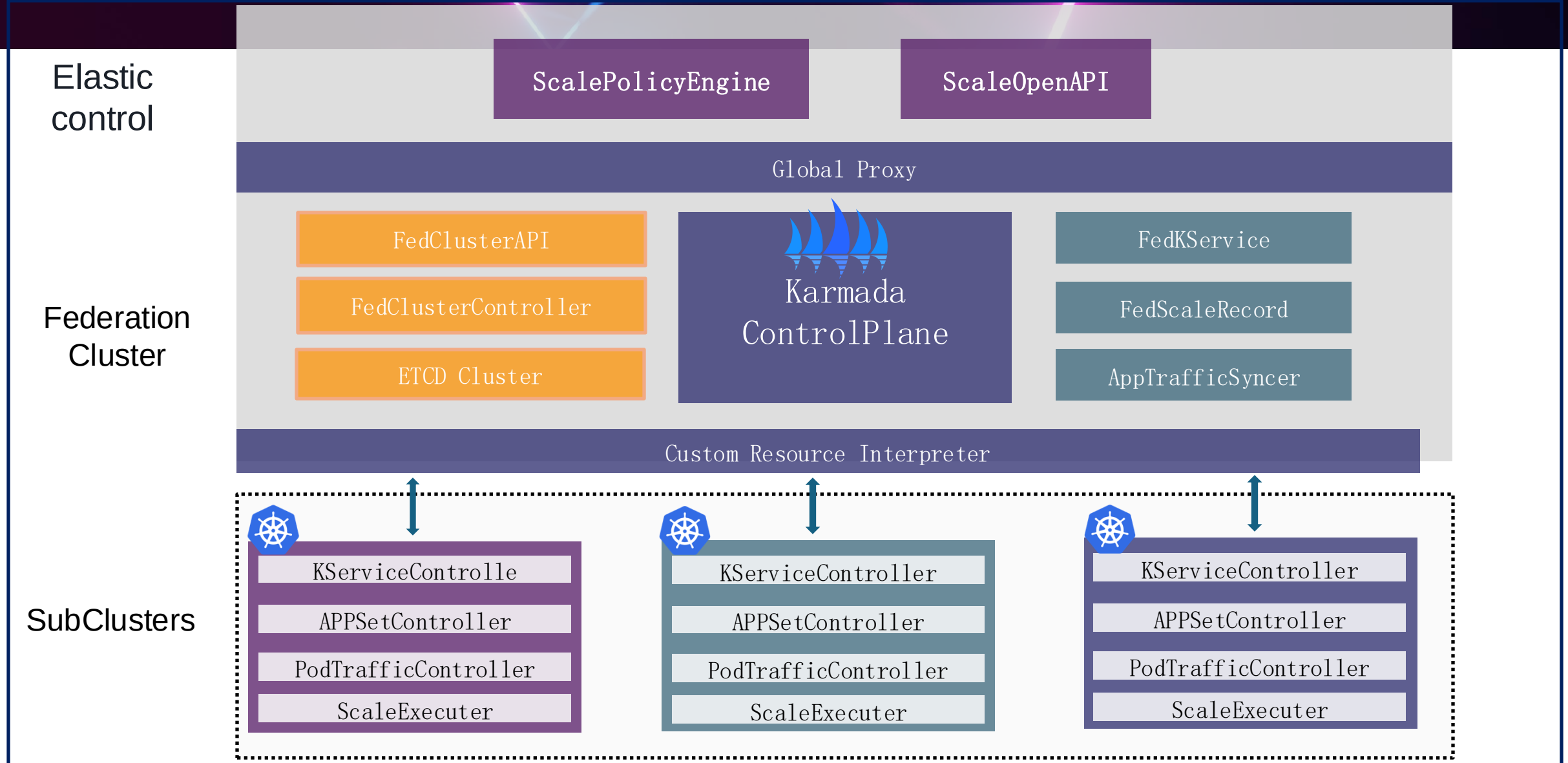
https://www.jdcloud.com/cn/products/yunjian



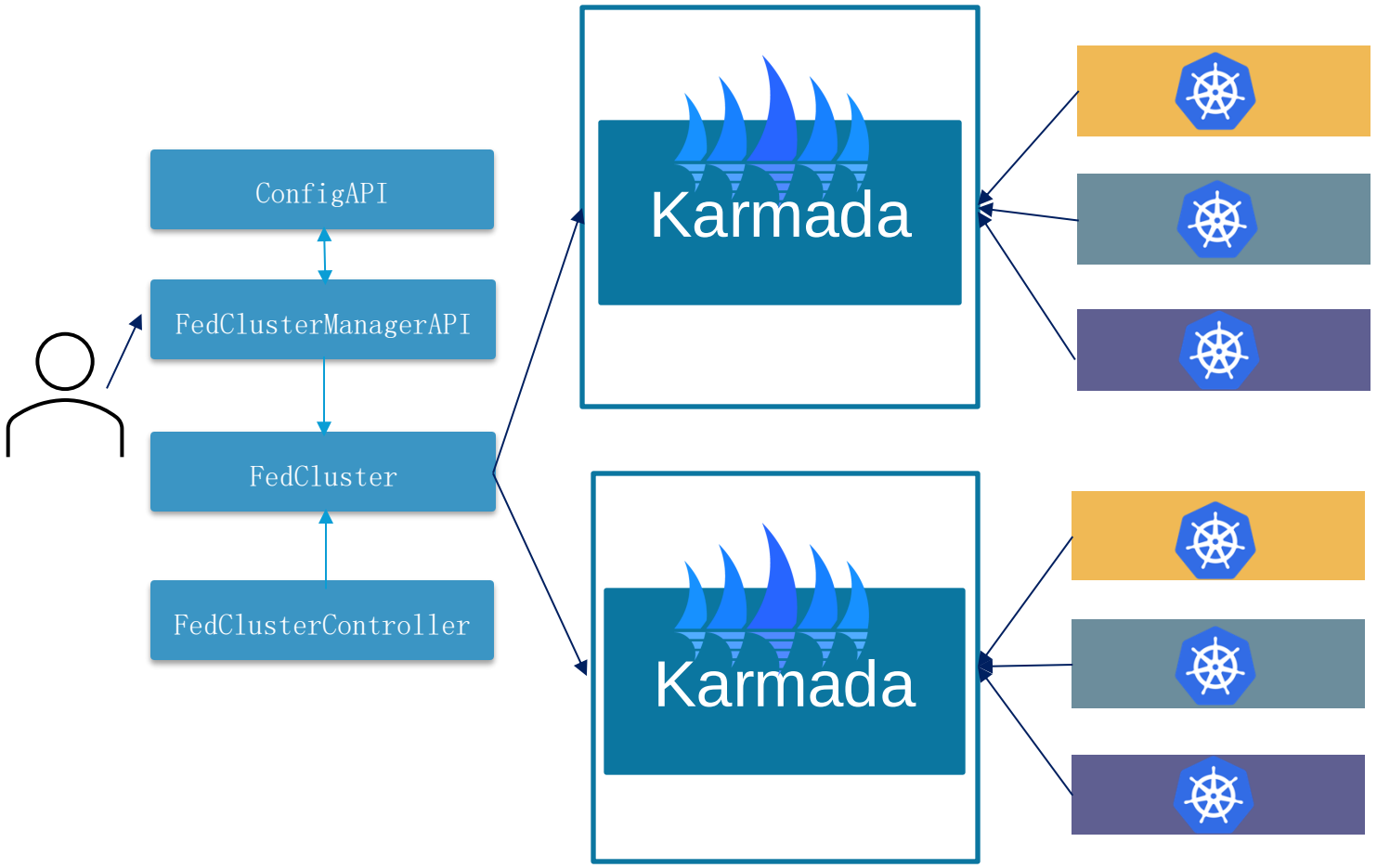
In-depth development and enhancement based on Karmada

https://github.com/karmada-io/karmada

Architecture



Federation cluster deployment



Create

The FedClusterManagerAPI handles user requests to create a Federation cluster and interacts with the ConfigAPI service to obtain the creation context and parameters. Create the FedCluster CR.

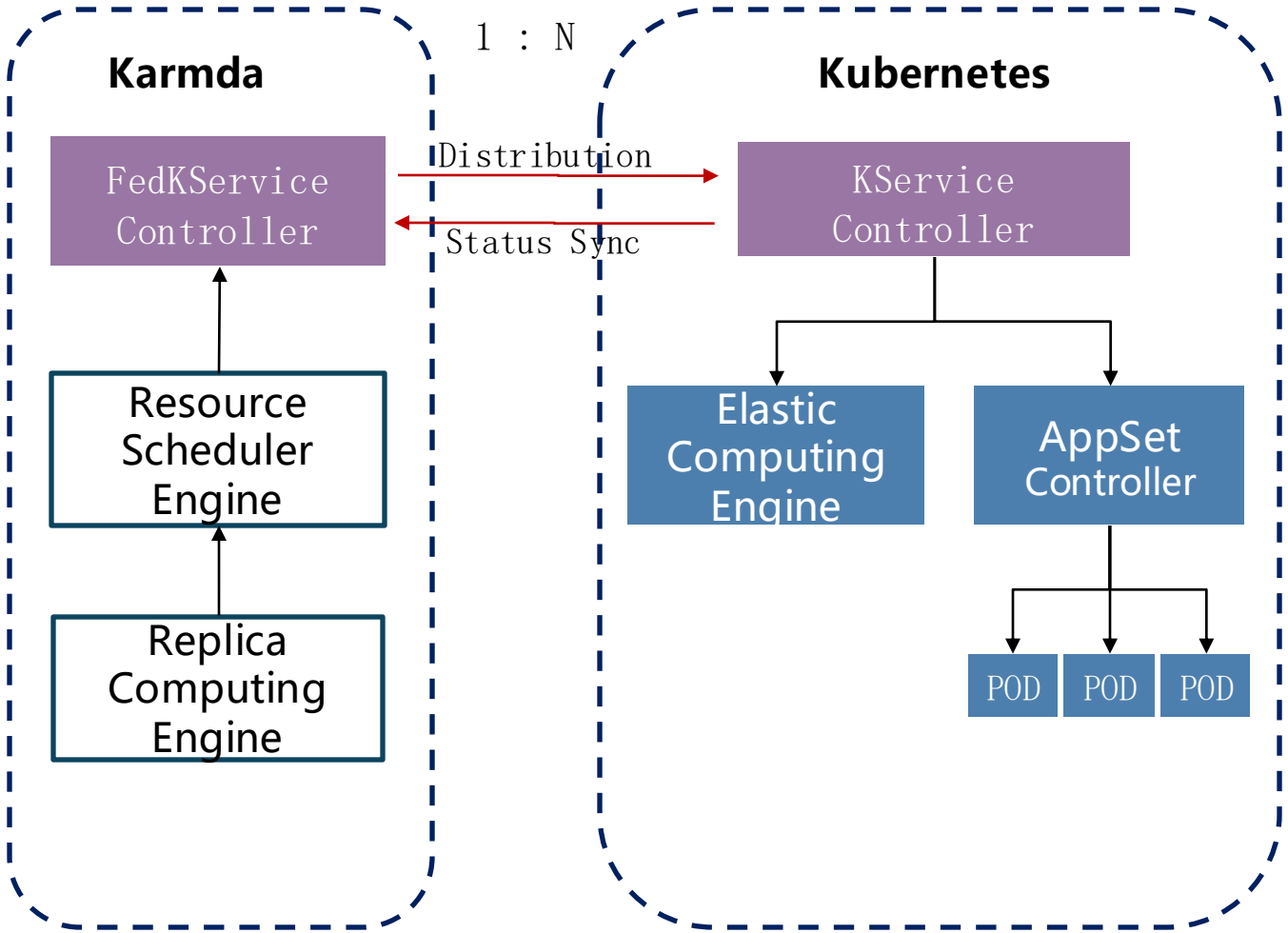
Watch

FedClusterController monitors FedCluster, according to the Spec statement, in the specified Kubernetes cluster creates a federated cluster ControlPlane.

Manage

FedClusterController monitors FedCluster and declares Spec to join the specified federation cluster. The federation cluster begins to manage multi-cloud Kubernetes clusters.

FedKService Controller



Resource Scheduler Engine

Based on the Hippo service, conduct precise calculation of multi-cluster resources to obtain resource portraits.

Replica Computing Engine

According to the resource portrait and scheduling strategy, perform replica multi-cluster splitting.

Elastic Computing Engine

Execute elastic strategy, execute scaling plan.

AppSet Controller

Directly control the workload of pods.

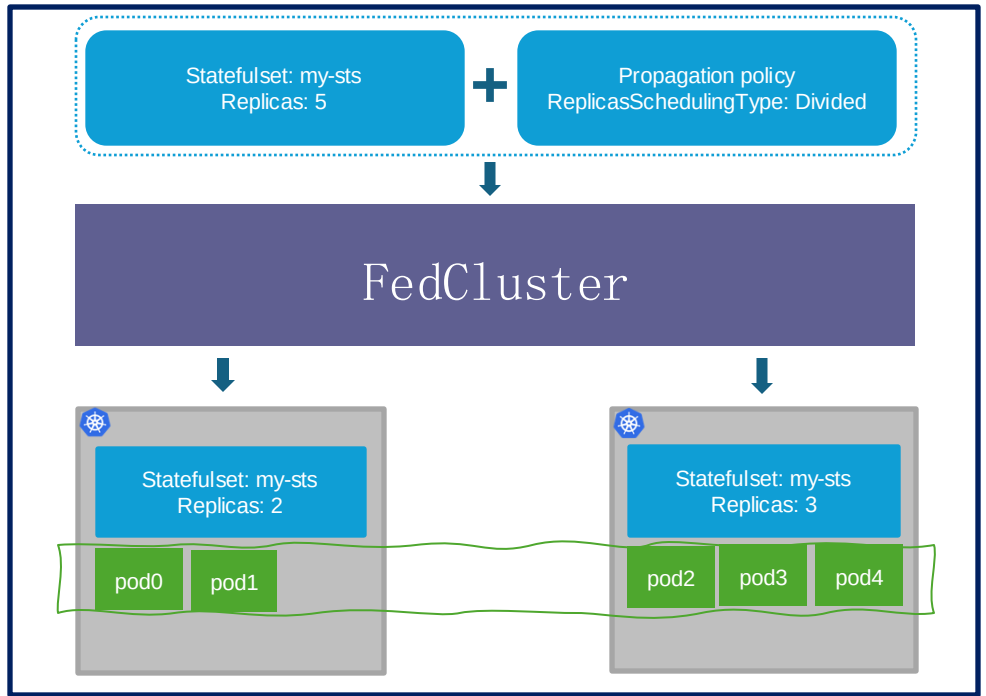
Federation Enhance

KService status sync

Solve the problem of KService status resource synchronization and merging from the sub cluster to the federation cluster.

StatefulSet starting number in multi-cluster control

The behavior of using StatefulSets in the federation cluster and single Kubernetes clusters with sequence number control is the same.

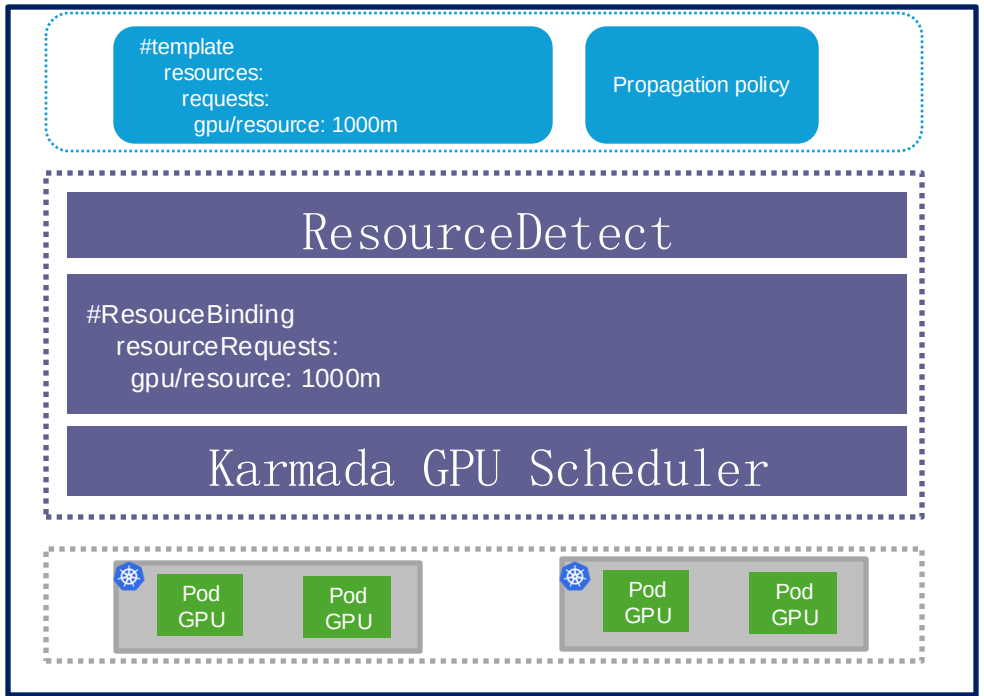


Global proxy performance improvement

1. Use the PB protocol for sub-cluster queries.
2. Modify from separate queries to concurrent queries.
3. Optimize stored procedures to achieve higher data access performance.
4. Performance improvement of 30%.

Karmada scheduler enhance

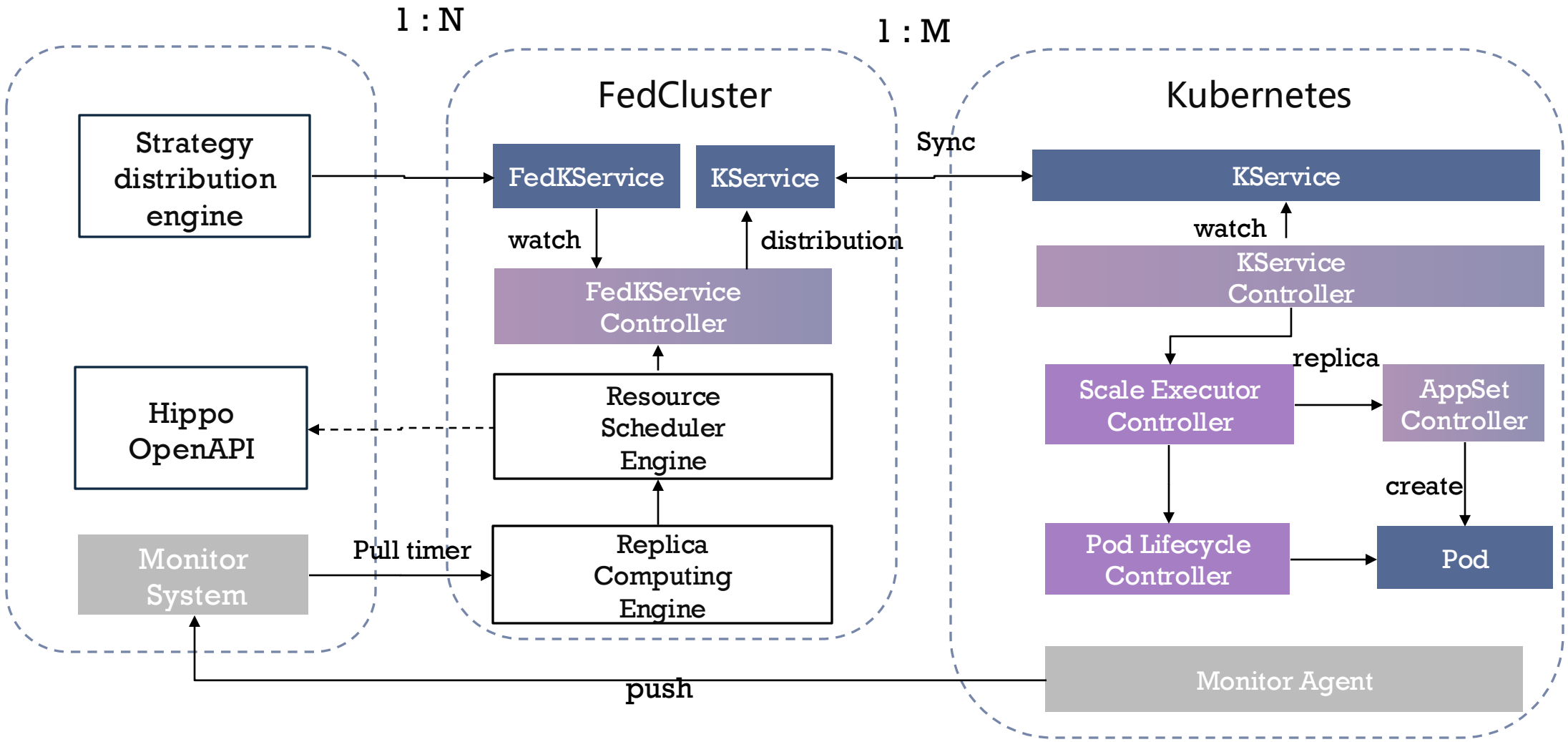
Support GPU cluster scheduling. Support heterogeneous cluster scheduling.



03

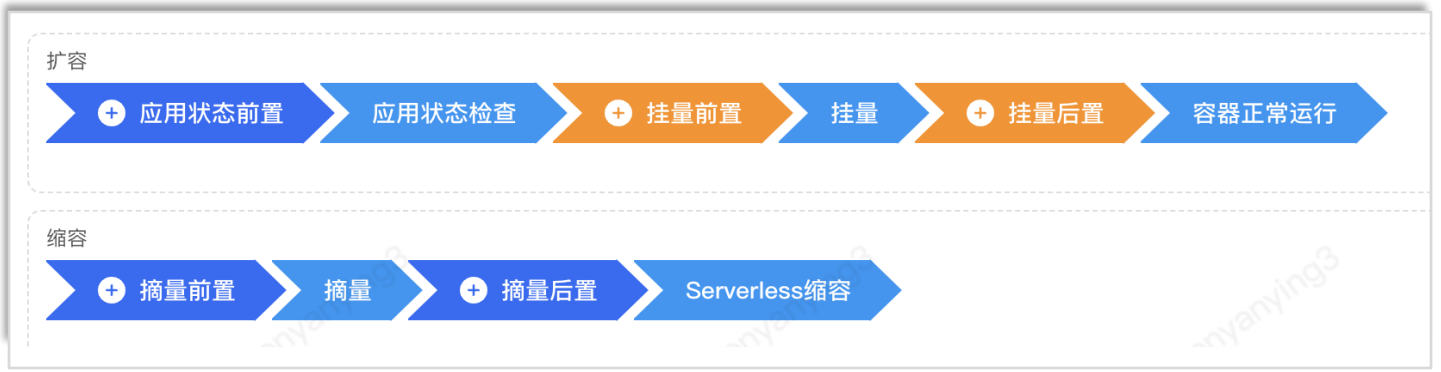
Across-cluster Auto-scaling

Auto-scaling Architecture



Auto-scaling Process

Strategy Calculation	Resource distribution	Resource Sync & Desync	Scaling execution	Pod Manage
Federation	**Federation**	**Federation**	**Cluster**	**Cluster**
Real-time matching auto-scaling strategy, calculating the number of replicas required	Resource estimation and distributing replicas to distinct clusters	Synchronize and desynchronize resource by Karmada	Repeatedly scaling up and down according to the auto-scaling strategy until the num of mounted Pods reaches the expected value	Configurable Pod lifecycle management Abnormal Pod GC mechanism



Auto-scaling Scenario

Automatic failover

Cluster node failover
Cluster offline
Disaster recovery migration across data centers

Big promotion of hot standby of pressure test container

When traffic is low, the hot standby container is compressed to a low specification
When traffic is high, low-specification hot standby containers are restored

Indicator monitoring scaling

Support machine performance and method call performance indicators
Personalized expansion and contraction rules

Scheduled task scaling

Support Cron expressions

Practice： Switching between auto-scaling and manual deployment

Auto-scaling scenarios

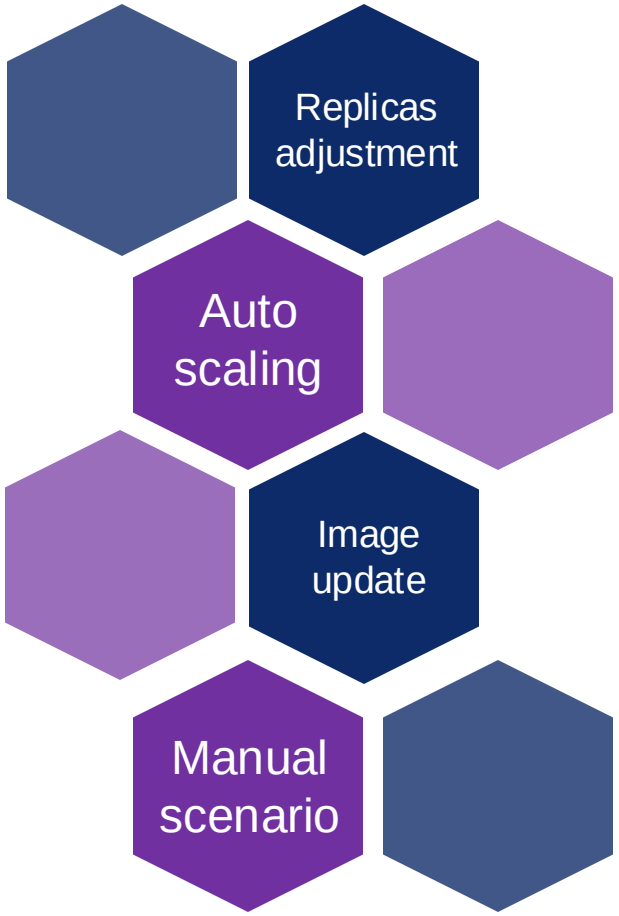
- Enabled by default, auto-scaling
- Only adjust the number of replicas
- Not responsible for updates of image versions, environment variables, configuration files, etc.

Manual scenario

- First online deployment
- Updates to images, configurations, environment variables, etc.
- Specify a single Pod update, online regression verification

Scene contradiction

- When deploying manually, you want the number of replicas to be stable and stop auto-scaling
- During auto-scaling, manual deployment will interfere with the execution effect of scaling



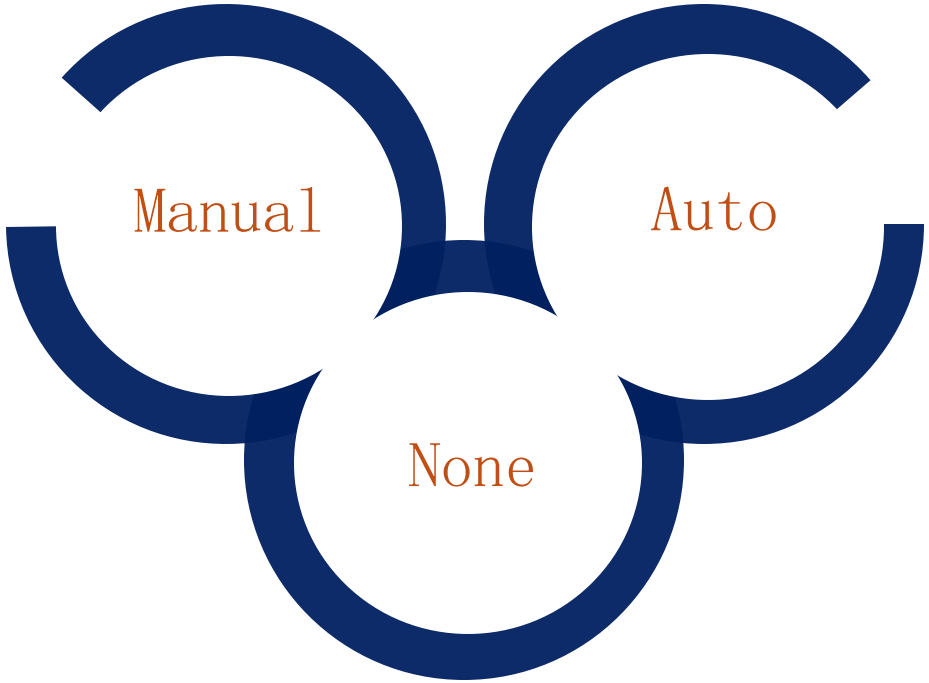
Three modes

Manual

Manual deployment scenario

Allow manual deployment

Shield auto-scaling control



Auto

Auto-scaling scenarios

Allow auto-scaling

Shield manual deployment control

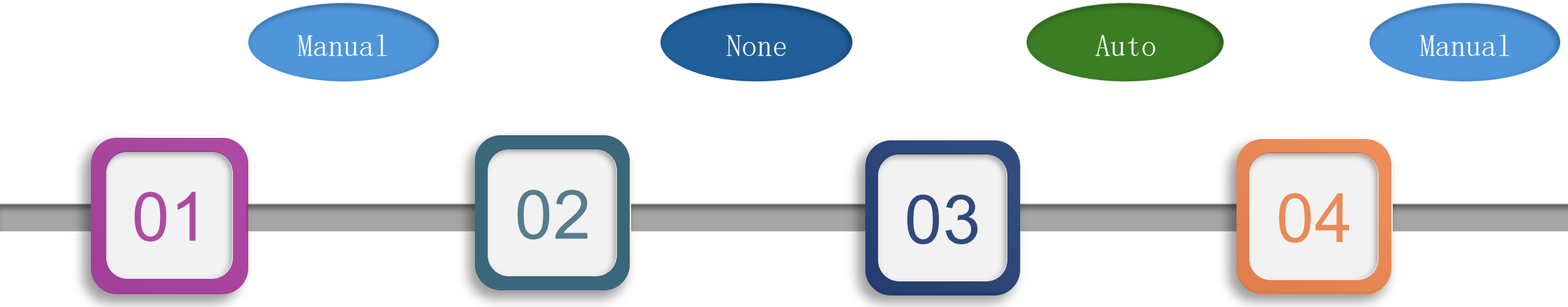
None

Modeless state, isolated scenarios

Shield auto-scaling control

Shield manual deployment control

The whole process



First deploy

The first manual deployment is in Manual mode

Confirm that manual deployment is complete

- Supports manual or automatic confirmation
- Switching can only be done after deployment is dispatched and completed

Mode switching

Switching conditions:

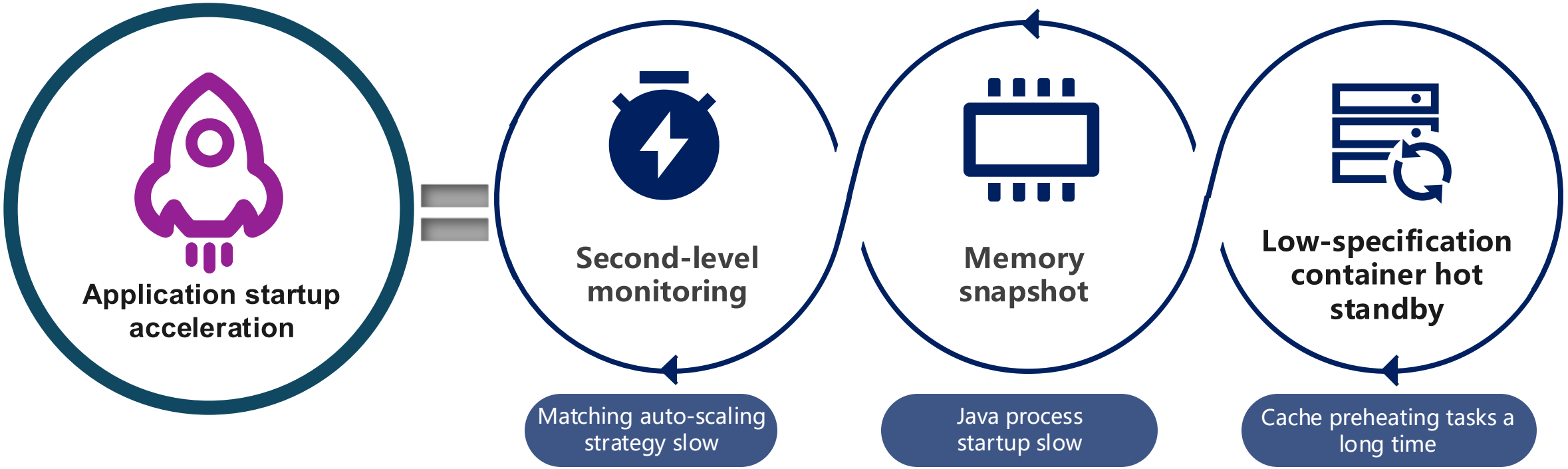
- Configure and enable auto-scaling strategy
- Manual deployment is completed
- Pod version is consistent

Manual deployment preemption

Post-preemption processing:

- Stop auto-scaling related controllers scaling
- Correct the number of replicas

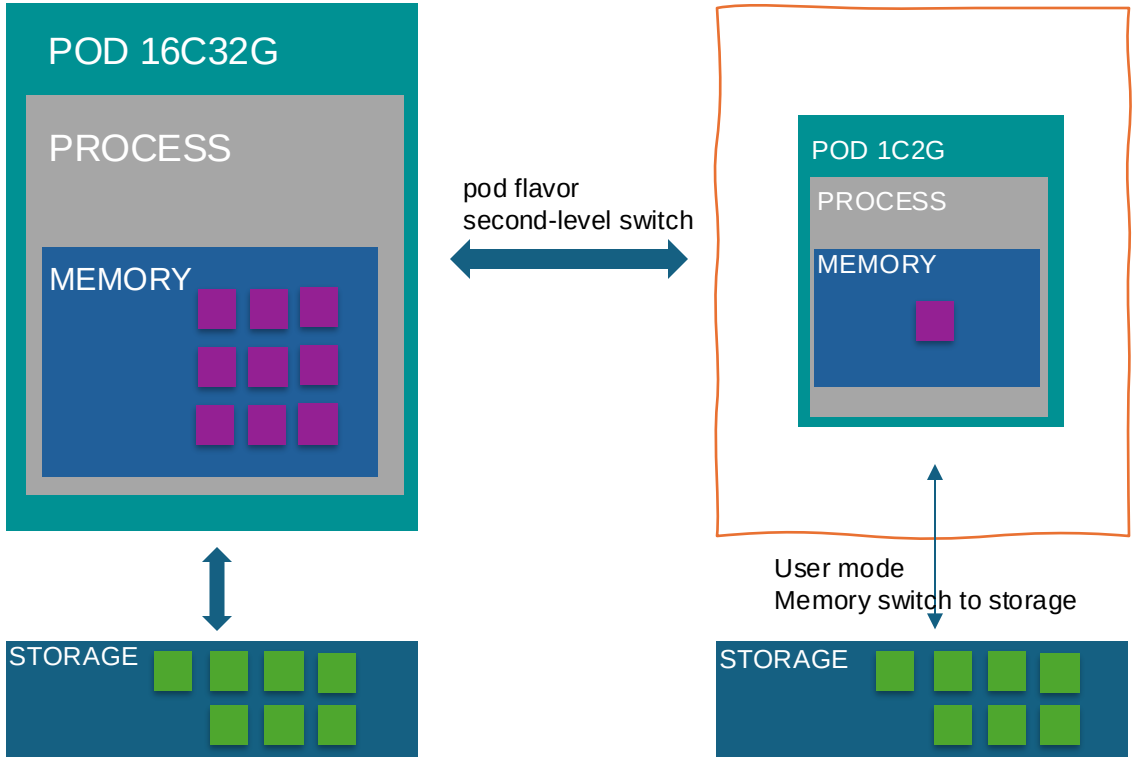
Practice: Application startup acceleration



Low-specification container hot standby

Solve scene problems

Before the promotion stress test, hot standby containers need to be cache-prepared in advance, which takes up a lot of resources.



Advantages

- Lossless, no need to modify applications
- Quick Pod startup, startup time reduced by more than 80%
- Java applications can handle traffic without preheating

Disadvantages

Need to deploy low-specification instances in advance, occupying a small amount of resources

Scenarios

The workload is mainly triggered by traffic, and the CPU utilization is low after traffic is removed.

Joylive

- **Service governance framework**

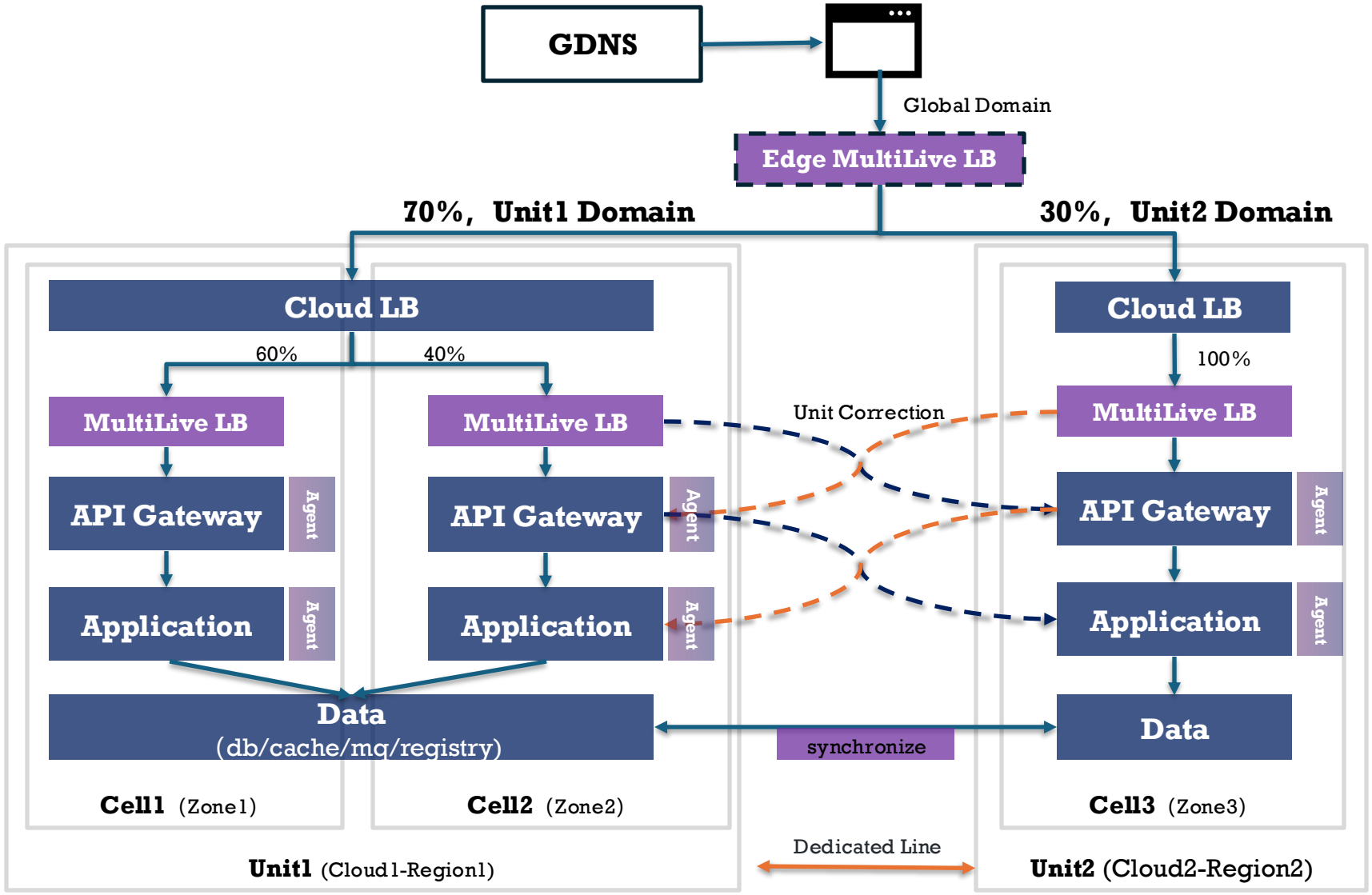
  A microservice traffic governance framework for application multi-active and unit-based based on bytecode enhancement

- **High availability improvement**

  Integrates Joylive to provide unit traffic management and disaster recovery recovery capabilities

- **Github**

  https://github.com/jd-opensource/joylive-agent



Cross-cluster service discovery and cross-cluster networking

Cross-cluster network

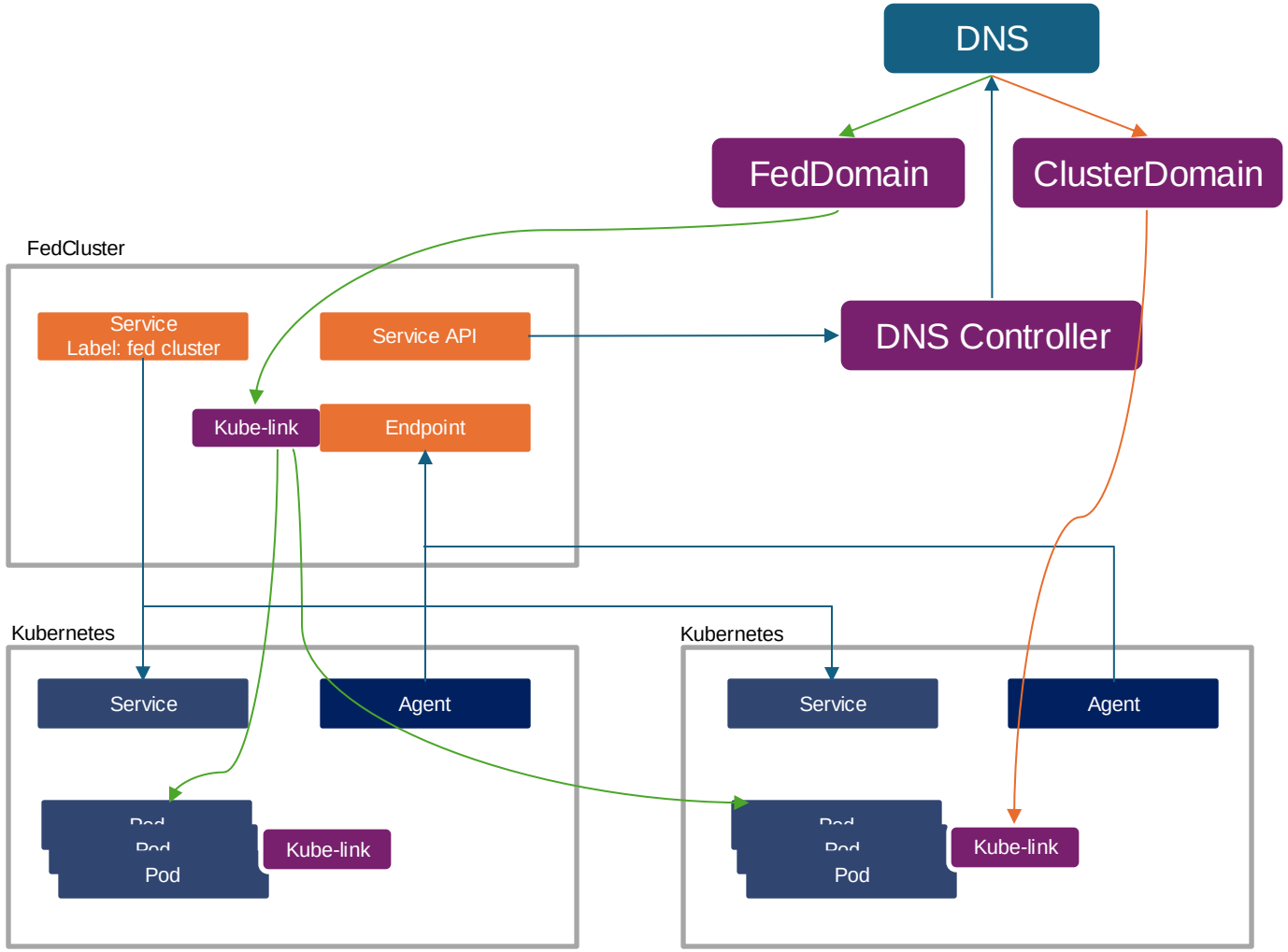
- Cross-cluster service discovery

Four-tier LB and federated domain name registration for multiple clusters

- Cross-cluster network communication

High-performance load balancing based on eBPF

- Prepare to open source



04

Summary

Production use

JD Spring Festival Gala Red envelopes Project

Supported JD Spring Festival Gala Red envelopes
Support hundreds of millions of times to grab red envelopes

E-commerce promotion

Supported multiple JD618 and JD11.11 e-commerce promotions, and supported the scaling of trillion-level transaction volume business

The main way to deploy services in the future

Continuous promotion and it will be one of the main ways to services deployment and automatic migration in the future
Application 200,000 cpu+

KubeCon CloudNativeCon China 2024

THE LINUX FOUNDATION OPEN SOURCE SUMMIT

AI_dev Open Source GenAI & ML Summit

JDCloud 2024

Thank you

Contact us

JDCloud Dev

Tech Group