
KubeCon
CloudNativeCon
OPEN SOURCE SUMMIT
China 2023

The Service Mesh Exploration of China Mobile Cloud

Haiwen Zhang
China Mobile

CONTENTS


KubeCon
CloudNativeCon
OPEN SOURCE SUMMIT
China 2023
Application Practice

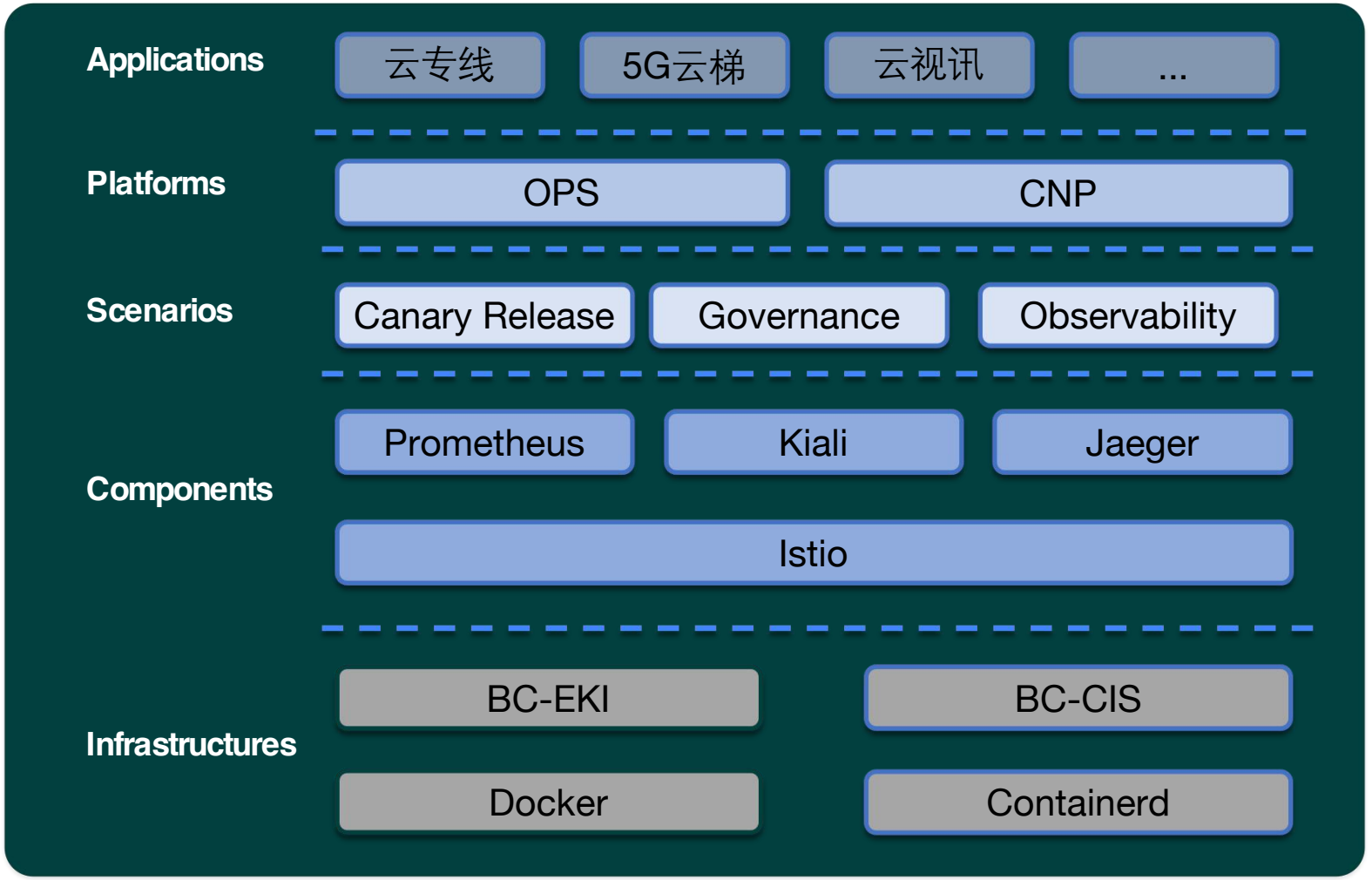
Overview

China Mobile Cloud

- China Mobile's cloud computing platform,TOP5 in public cloud market and TOP3 in the IaaS subdomain market in China.

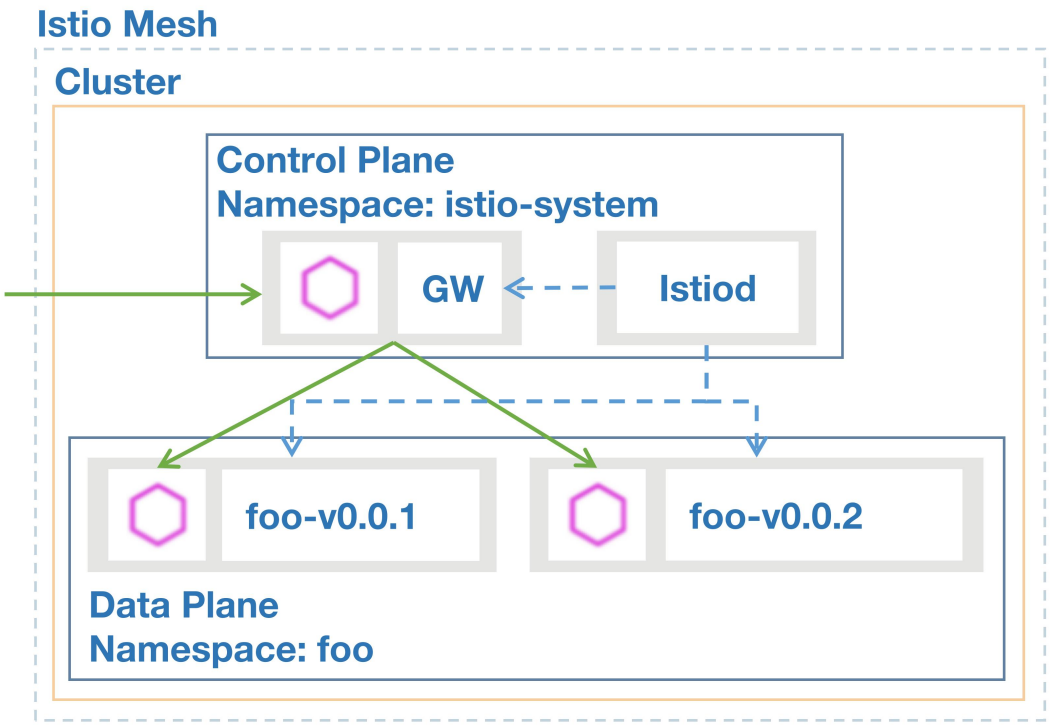
Scenarios

- Nearly 50 products are using service mesh for canary releases, traffic management, and observability.
- By OPS and CNP, we provide service mesh capabilities to both internal products and external customers.

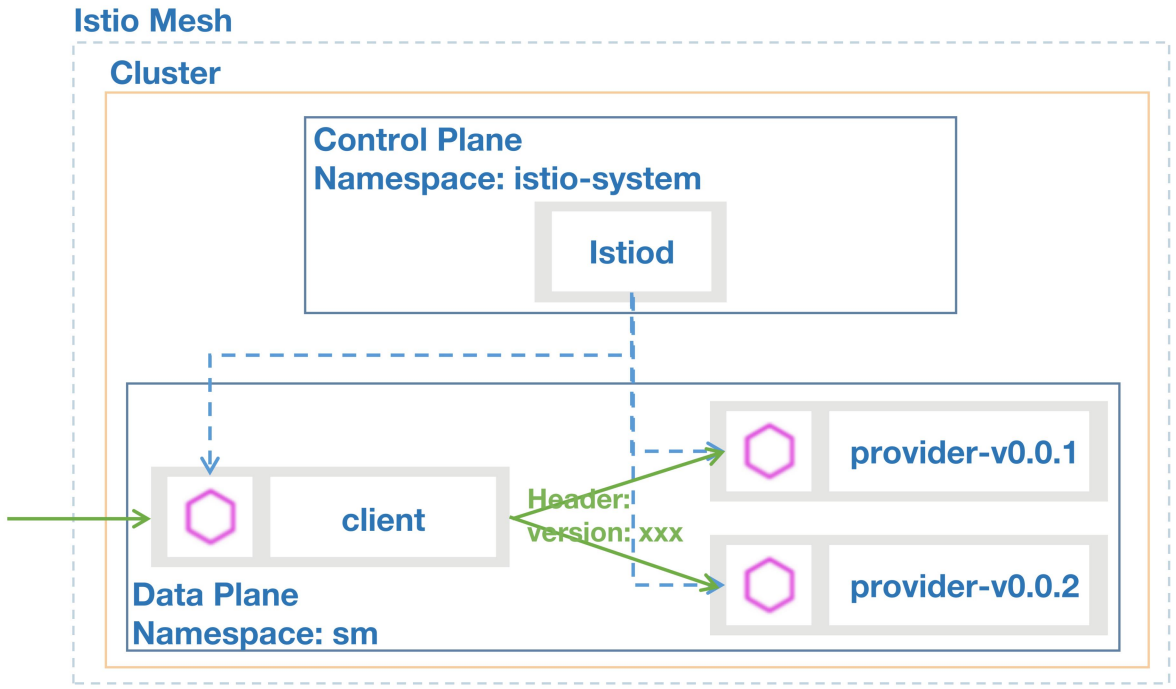


Canary Release

Many scenarios for canary release

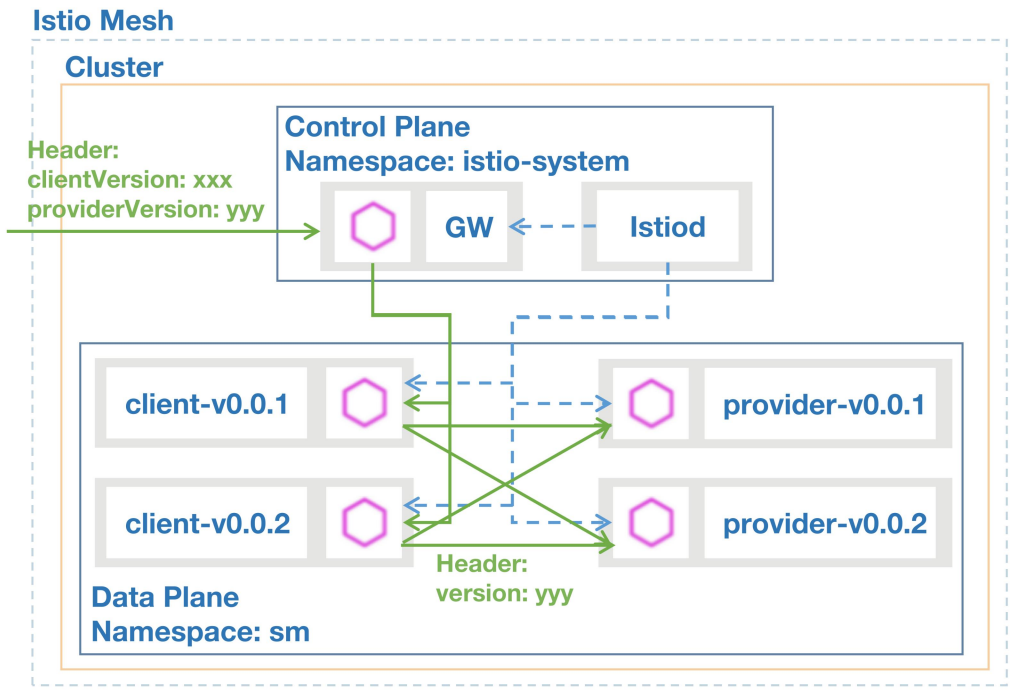


Canary Release for the Entry Microservice

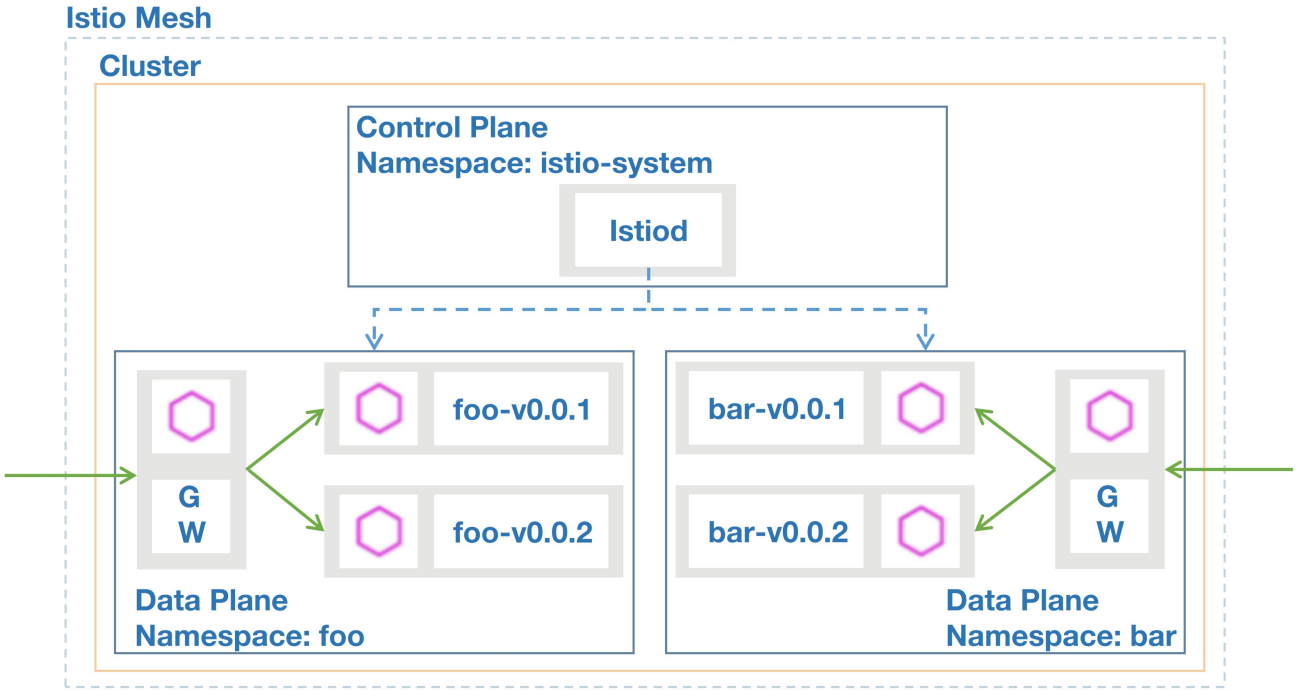


Canary Release for Internal Microservices

Canary Release



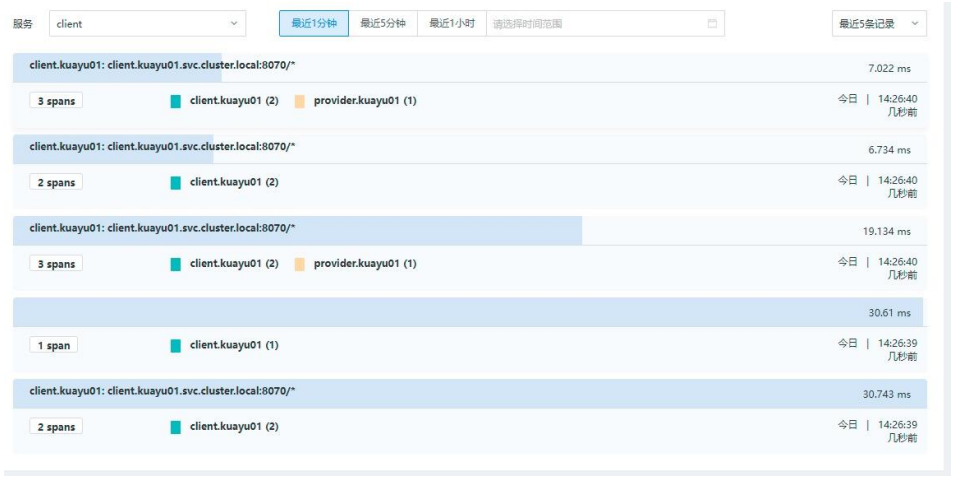
Canary Release for End-to-End Microservices



Canary Release for Multi-Tenant Microservices

Observability

Service Topology

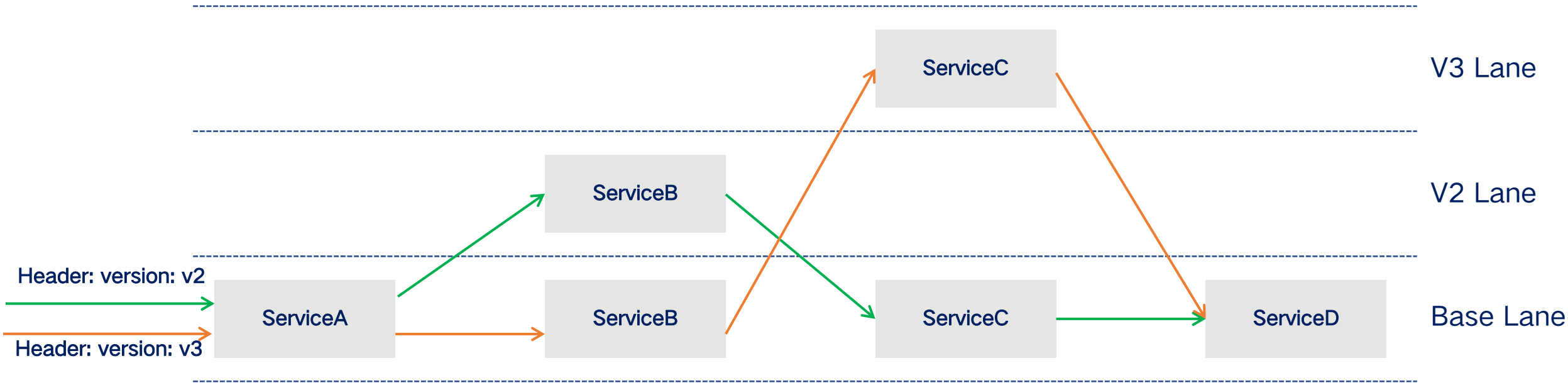


Tracing

Traffic Lane

Background: Dozens of microservices; Multiple concurrent development iterations; Challenge to Deploy the environment for each iteration; Sharing the same environment leads to mutual interference.

Solution: Building end-to-end traffic lanes based on Istio+OpenTelemetry to achieve iteration environment isolation.



Traffic Lane

Computing Network-Native

East-West Computing Project

Aim to enable low-cost computing resources in the western regions to support data processing in the eastern regions.

Computing Network

A deep integration of computation and networking, to meet the demands of East-West Computing Project.

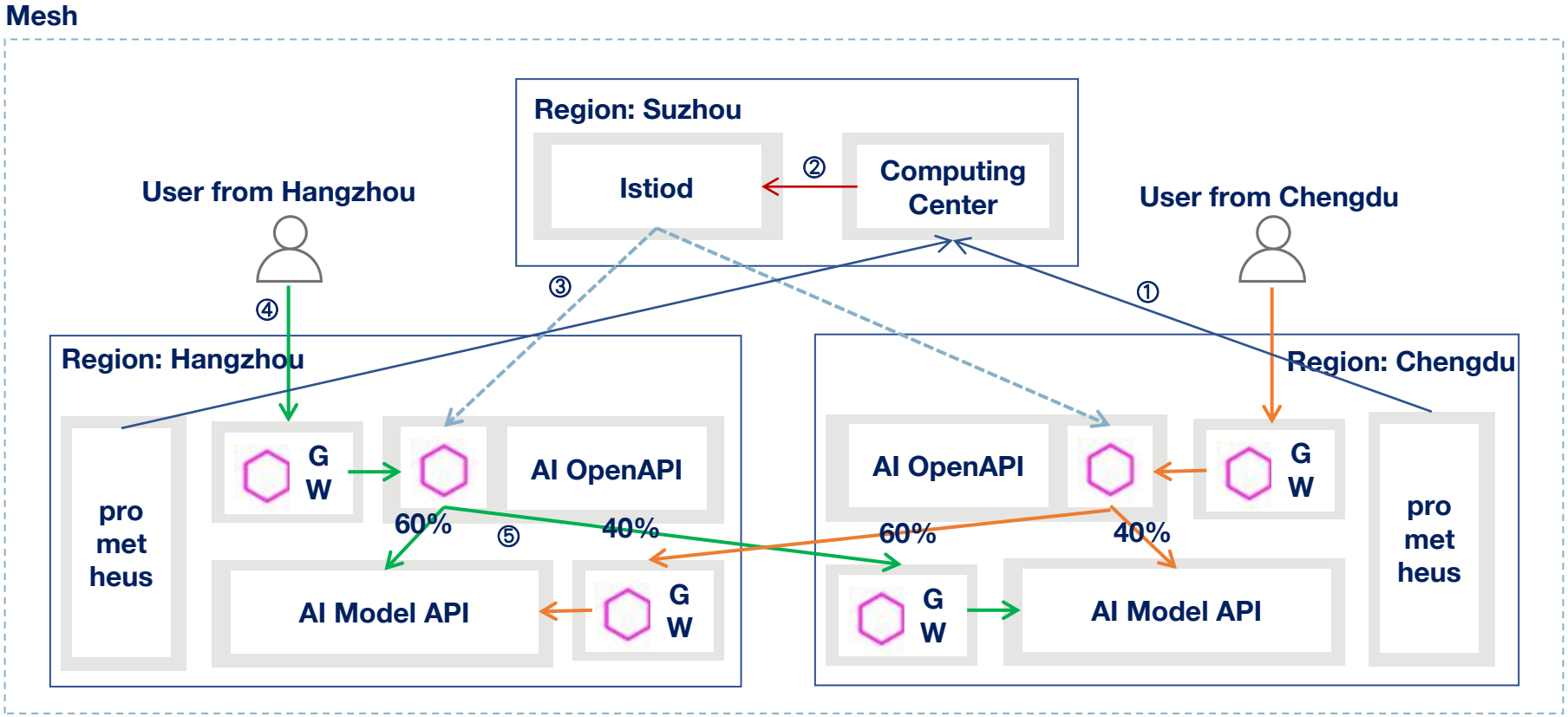
Computing Network-Native

The computing network presents new challenges to cloud-native.China Mobile Cloud has proposed the concept of 'Computing Network-Native'.

Computing Network-Native

Demo: Users access the facial recognition service from nearby locations. The Computing Network-Native dynamically adjusts traffic and allocates computational resources based on resource availability.

Technology: Istio Single Mesh Multi-Cluster Multi-Network Model.

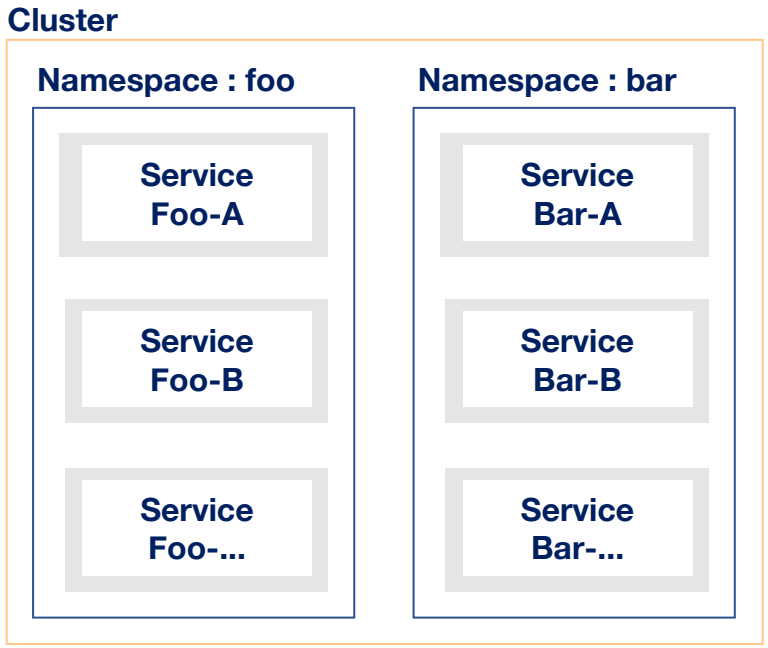


Computing Network-Native Demo

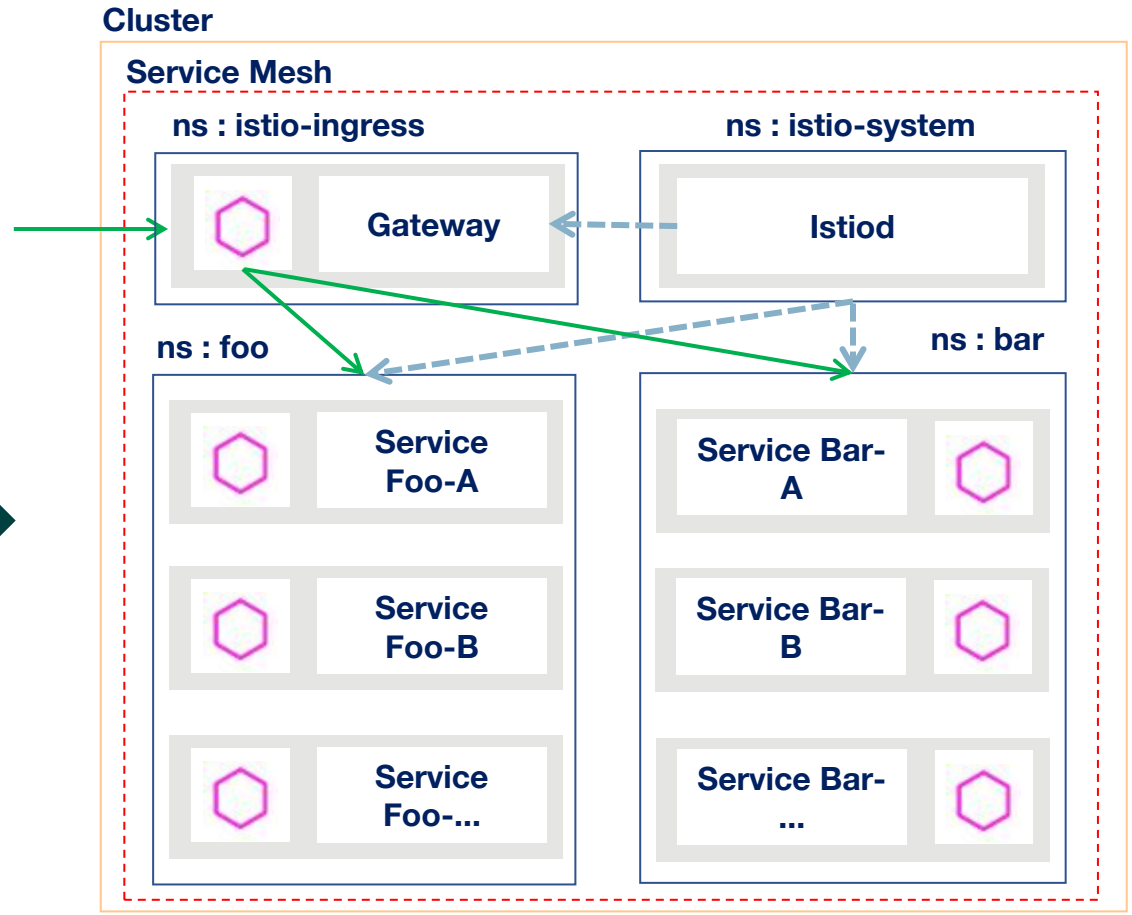

KubeCon
CloudNativeCon
OPEN SOURCE SUMMIT
China 2023
Technical Exploration

Multi-Tenant Practice

Background: Multiple products are deployed within a single Kubernetes cluster, segmented by namespaces.
Issue: If following the Istio classic deployment model, performance and fault isolation issues may arise.
Requirement: A multi-tenant solution at the Namespace level for Istio.



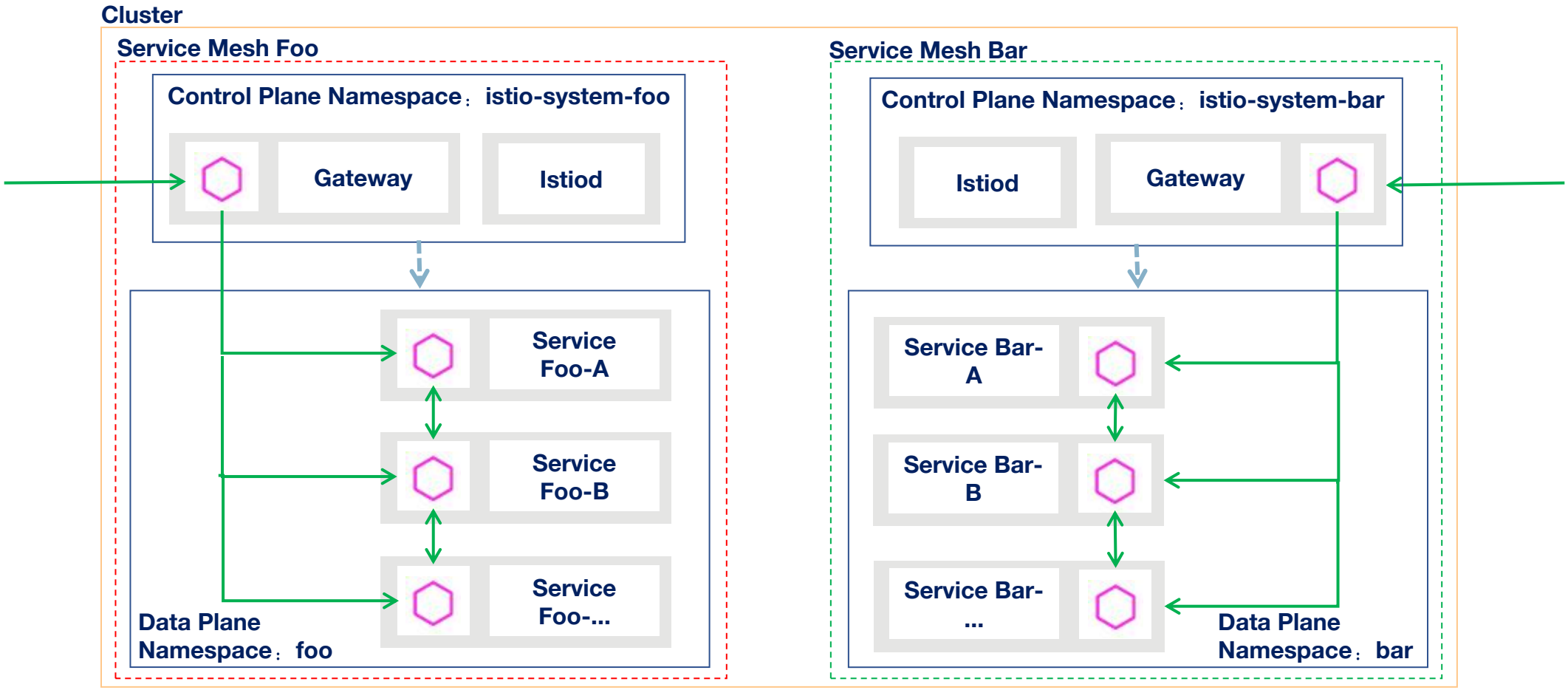
Application Deployment Model



Istio Classic Deployment Model

Multi-Tenant Practice

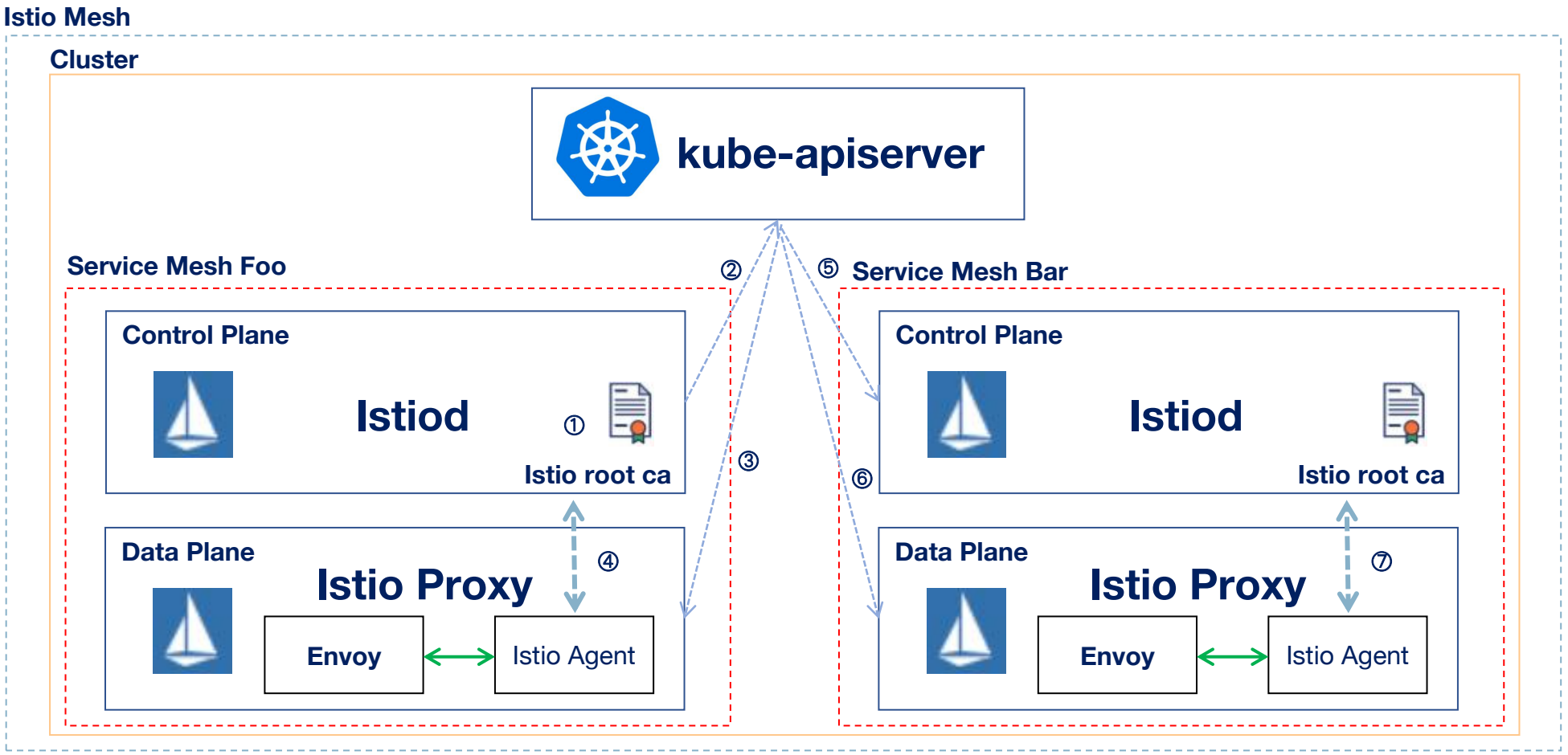
Ideal Solution: Tenant isolation for both the control plane and data plane.



Tenant isolation Model for both the control plane and data plane

Multi-Tenant Practice

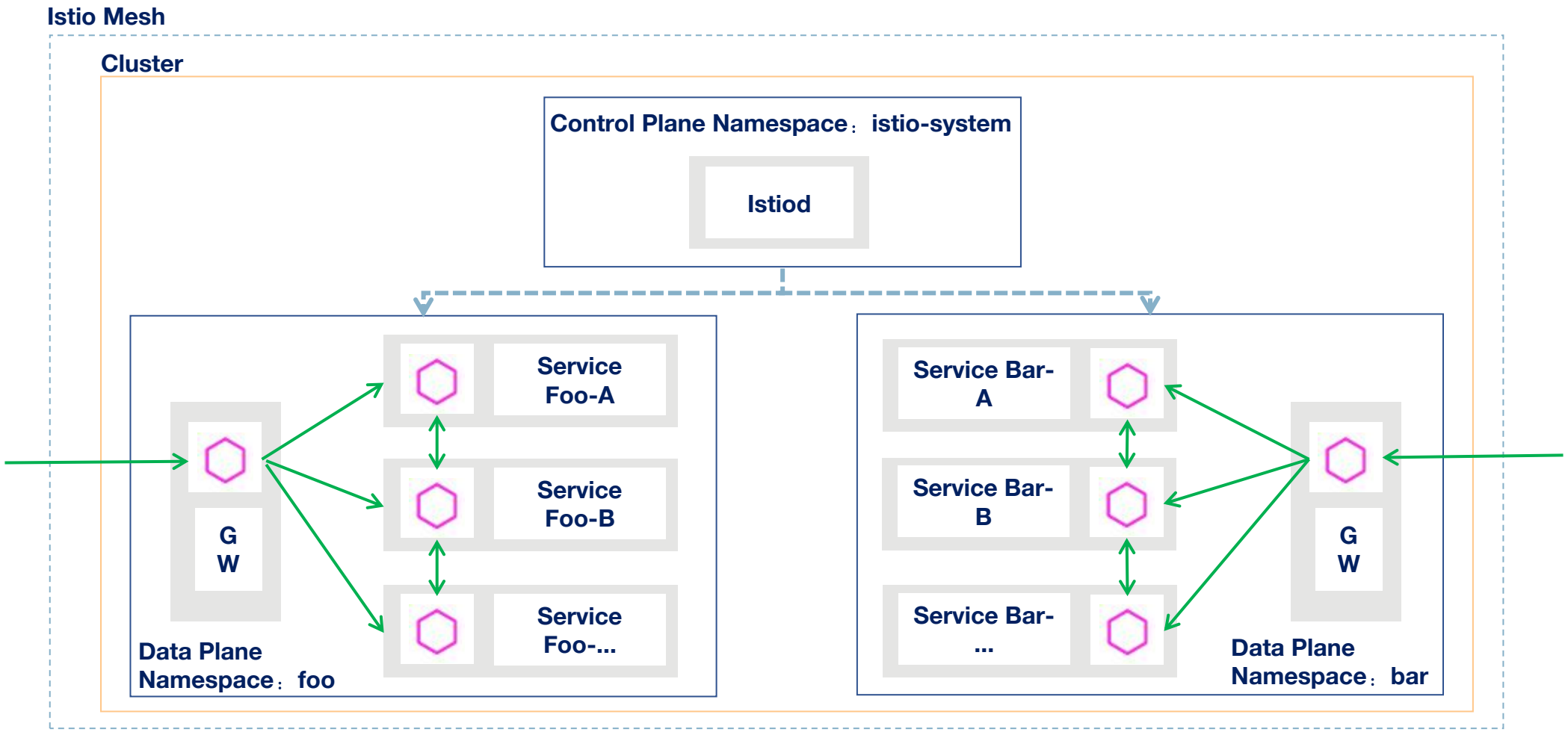
The solution is feasible, but we decide to abandon it. Because it requires code modifications, leading to high development and maintenance costs.



Service Mesh authentication process

Multi-Tenant Practice

Middle ground Model: Partial isolation, shared control plane, isolated data plane, no code modification, and no development or maintenance costs. We have adopted this model.



Partial isolation Model

Multi-Tenant Practice

Key Point: Istiod's environment variable PILOT_SCOPE_GATEWAY_TO_NAMESPACE

```yaml
apiVersion: install.istio.io/v1alpha1
kind: IstioOperator
metadata:
  namespace: istio-system
  name: istio-control-plane
spec:
  namespace: istio-system
  profile: default
  components:
    pilot:
      k8s:
        env:
          - name: PILOT_SCOPE_GATEWAY_TO_NAMESPACE
            value: "true"
    ingressGateways:
      - name: istio-ingressgateway
        enabled: false
```

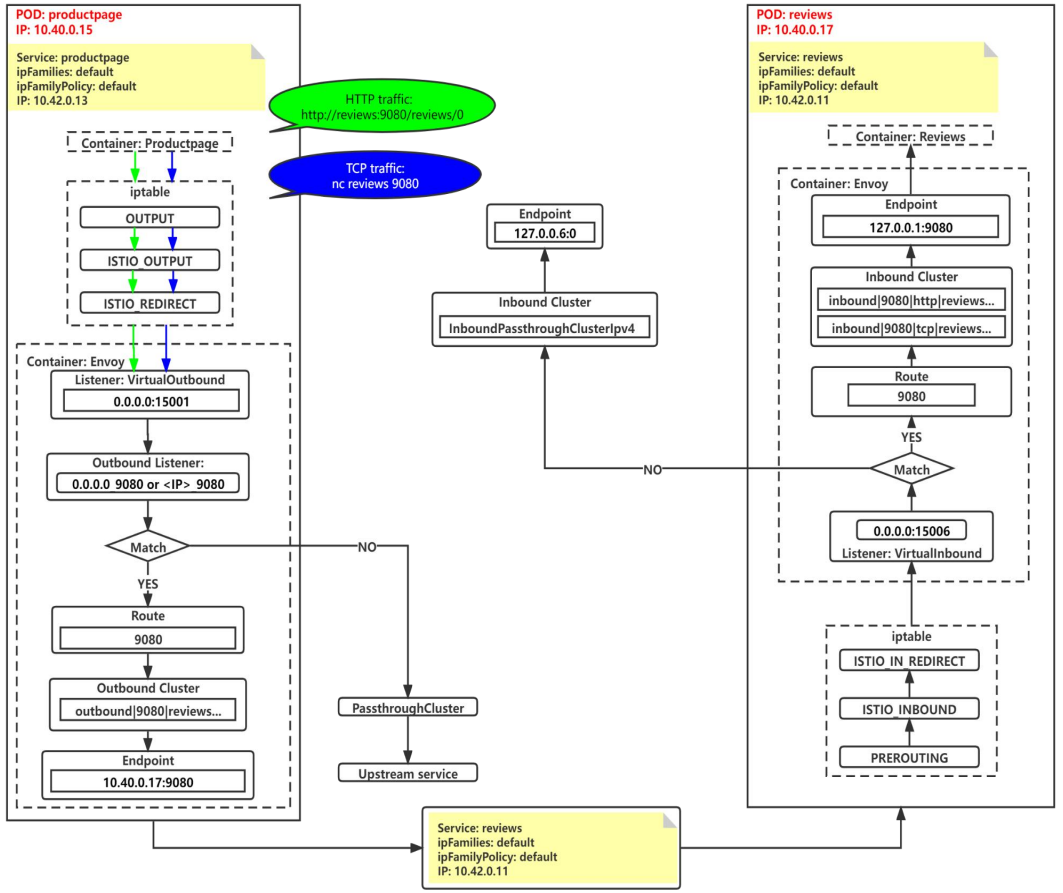
IstioOperator for control plane

```yaml
apiVersion: install.istio.io/v1alpha1
kind: IstioOperator
metadata:
  namespace: istio-system
  name: foo-ingress-gateway
spec:
  profile: empty
  components:
    ingressGateways:
      - name: istio-ingress-gateway
        namespace: foo
        enabled: true
```

IstioOperator for ingress gateway

Dual-Stack Practice

With the increasing trend towards IPv6 adoption, In 2022, China Mobile Cloud planned to enable all products to be accessible via both IPv4 and IPv6, implementing a dual-stack approach. However, Istio did not support dual-stack network environments in the early part of 2022.

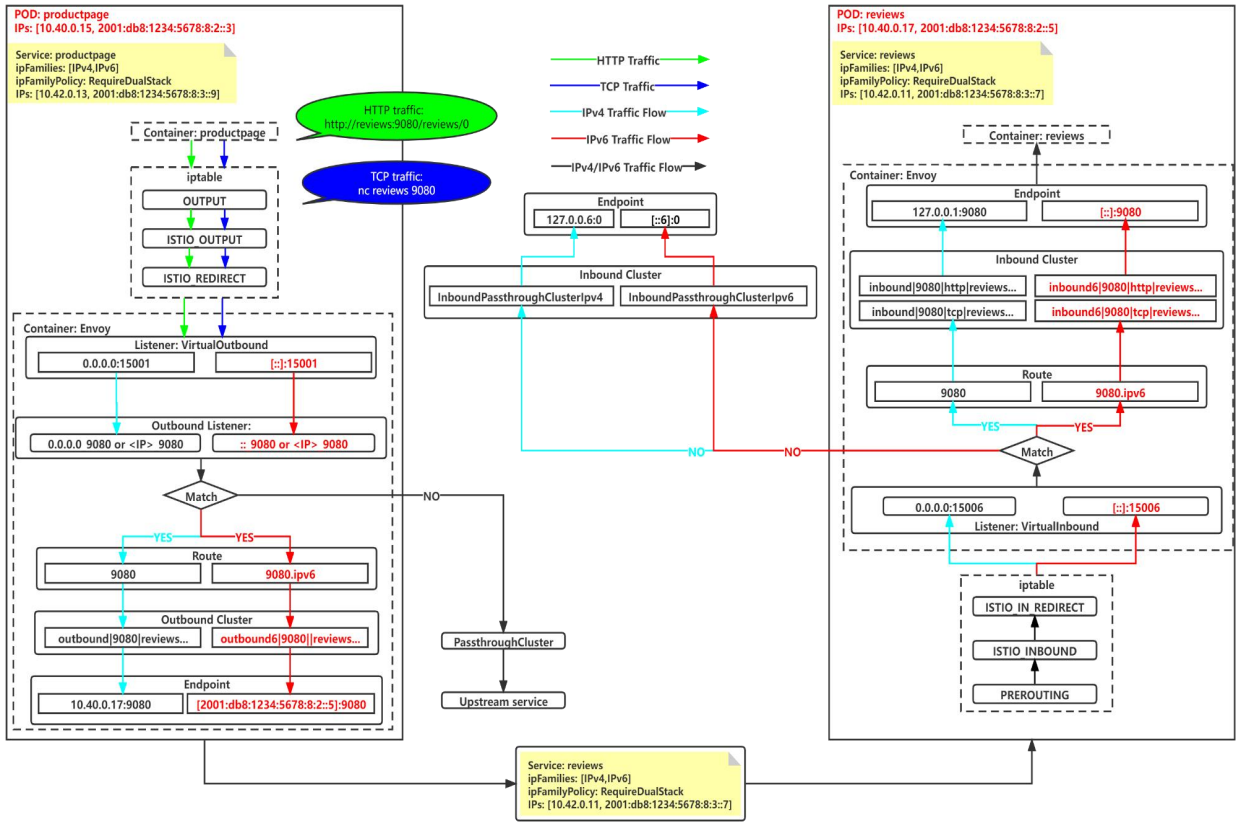


Not supporting dual-stack, only single-stack.

Dual-Stack Practice

Based on the solution involving the generation of duplicate sets of xDS configurations, We collaborated with Intel to explore and implement dual-stack technology in Istio, contributed exploration findings back to the Istio Community （Istio experimental-dual-stack branch）.

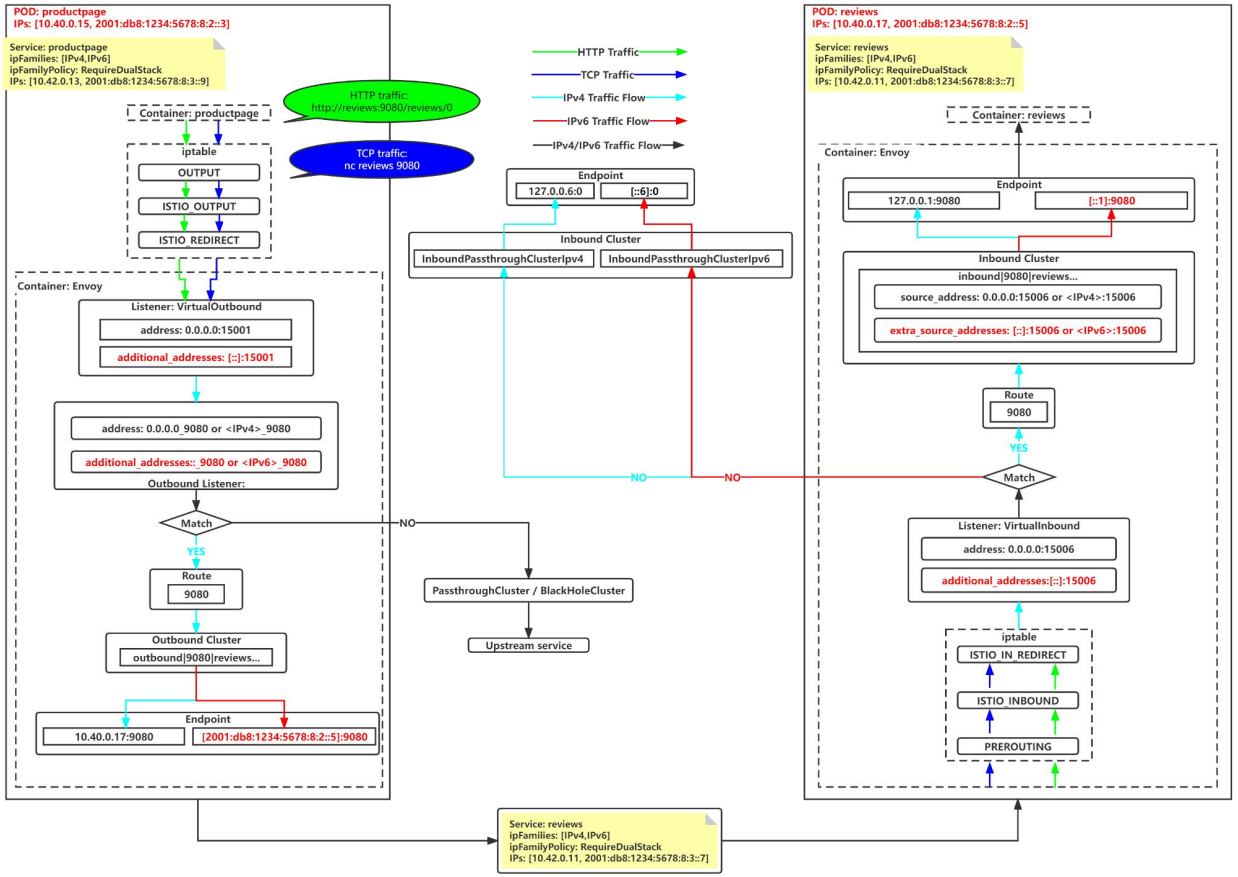
It has been widely adopted within China Mobile Cloud and China Mobile, becoming the first successful deployment of dual-stack Istio Community solution in China.



Duplicate sets of xDS configurations solution

Dual-Stack Practice

Currently, the Istio and Envoy communities are optimizing and enhancing the dual-stack solution. They have achieved a dual-stack solution based on a single set of xDS configurations to reduce resource consumption and improve performance. This enhancement was released in version 1.17.
China Mobile Cloud is currently experimenting with this solution.



Single set of xDS configurations solution


KubeCon
CloudNativeCon
OPEN SOURCE SUMMIT
China 2023
Product Development

Evolution Process

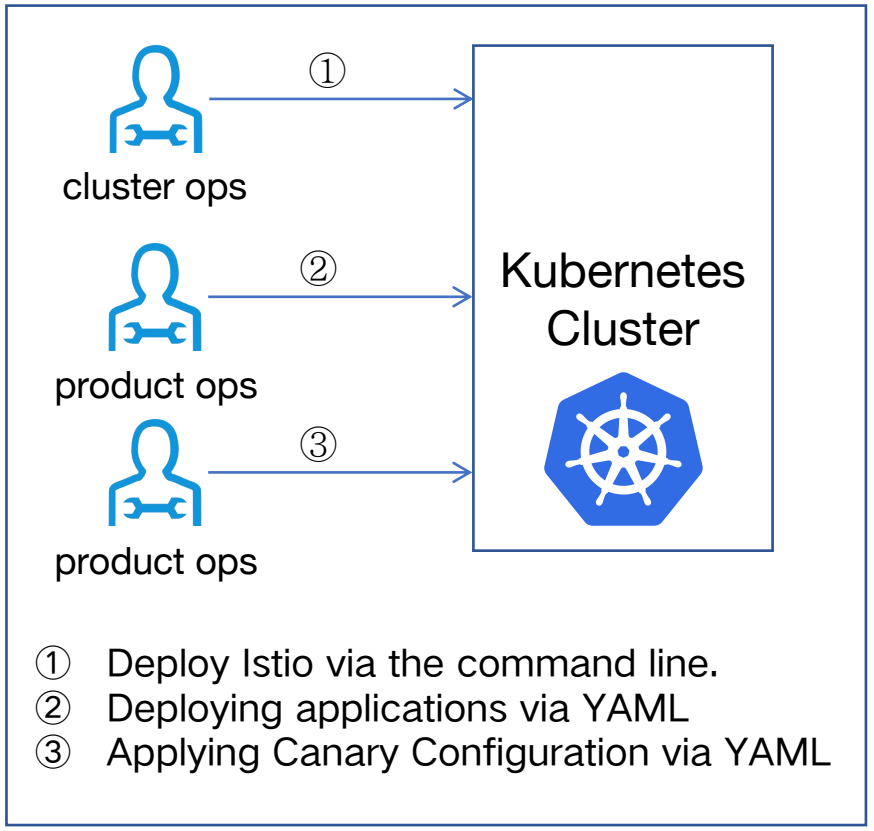
The exploration of China Mobile Cloud's service mesh has undergone an evolutionary process

Usage in component-level

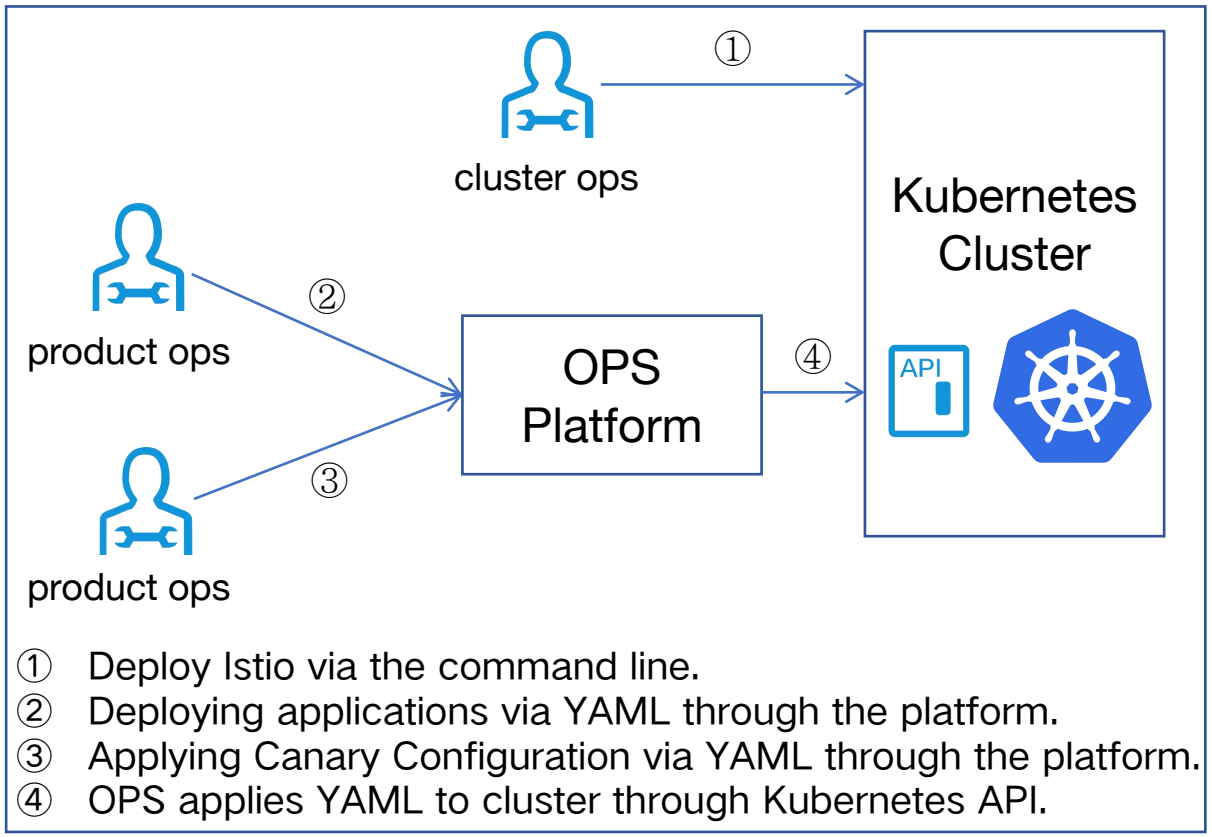
Integration with OPS platforms

Combining with OAM

Evolution Process



Usage in component-level



Integration with OPS platforms

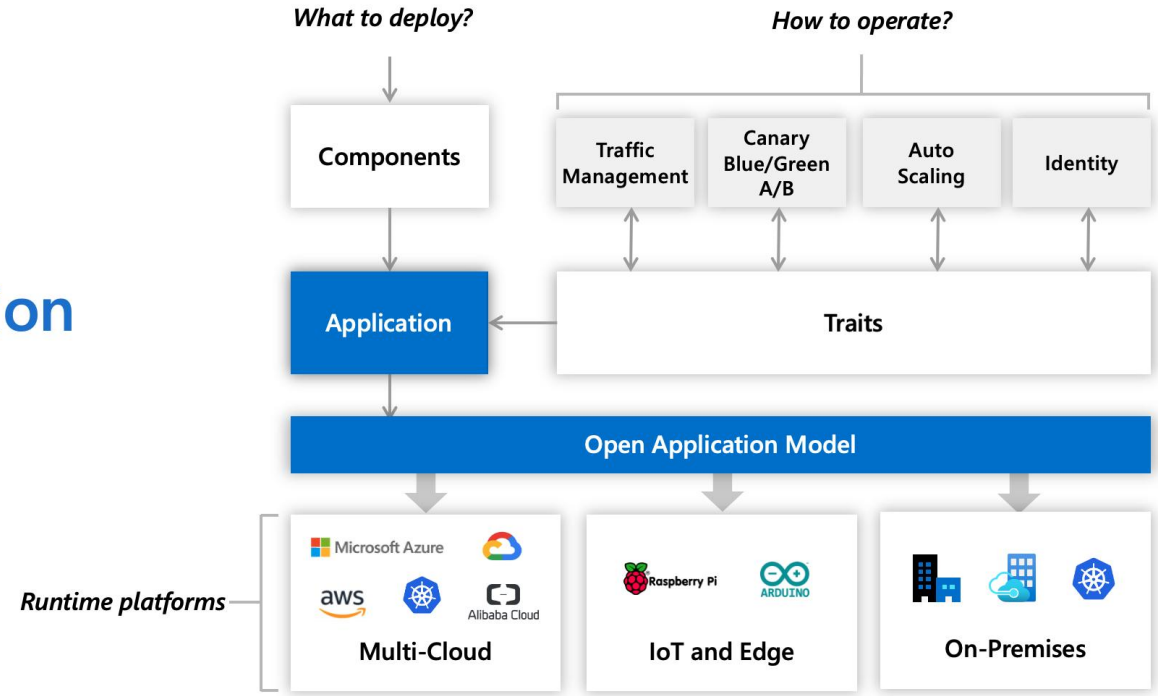
Whether at the component level or through integration with operational platforms, the product teams have been compelled to become Kubernetes experts, Istio experts, and YAML engineers.

OAM

We are gradually gaining a clear understanding of the cloud-native application platform, and We need to provide a platform that adheres to the following principles:

- Abstracting the underlying infrastructure.
- User-centric with applications at the core.
- Excellent scalability.

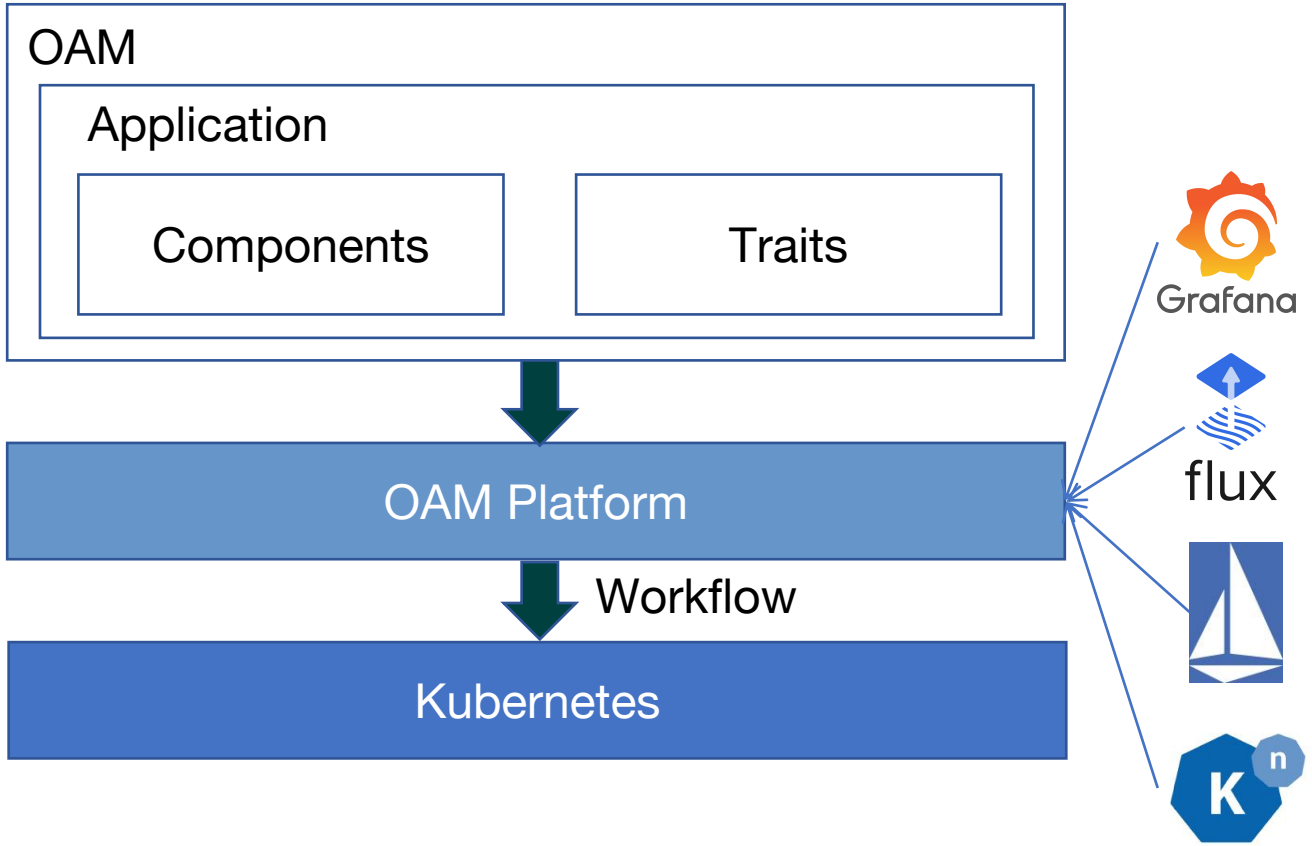
The Open Application Model (OAM) perfectly meets our needs.



OAM+OAM Platform

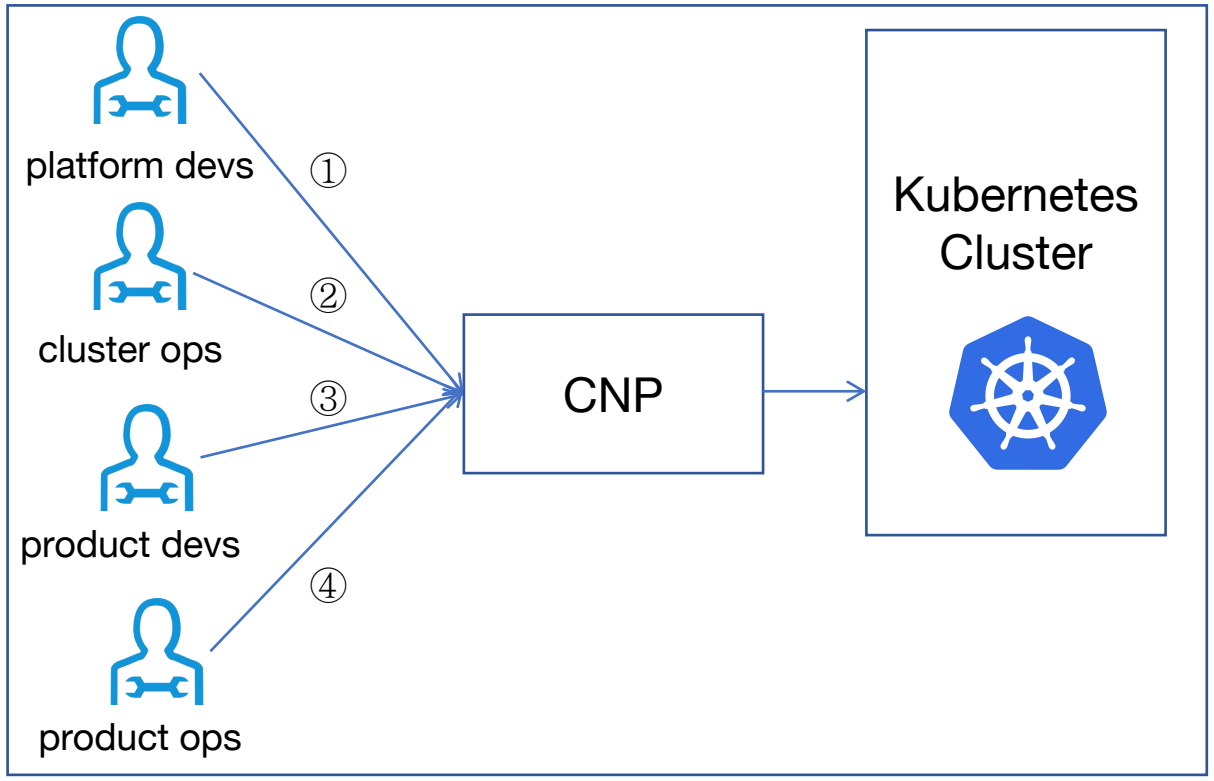
Basic Concepts:

- Component: The fundamental building block for defining an application.
- Trait: Operational properties defined on components.
- Application: Combining Components and Traits together to represent the complete configuration of an application.
- OAM Platform: The platform for implementing the OAM model like KubeVela.
- Plugin: The OAM platform introduces some ecosystem components in the form of plugins.
- Workflow: Part of the application deployment plan, assisting users in customizing the steps within the application deployment plan.



Evolution Process

We have explored and built our own cloud-native platform (CNP) based on the OAM and OAM Platform (KubeVela) to serve both internal products within China Mobile Cloud and external customers. We are also actively contributing our exploration findings back to the KubeVela community.



Combining with OAM

① Develop Istio plugin, canary release traits, canary release workflows, and integrate them into CNP.
② Deploy Istio through the plugin-based approach.
③ Create Application Components.
④ Creating product-specific canary traits and workflow, deploy canary versions of components and apply grey release configurations through the workflow.

Technical Implementation

As platform developers, our task is only to develop Istio plugins, canary release traits, canary release workflows, and integrate them into the platform.

```
package main

import "strings"

_base:    *"" | string
registry: *"" | string

deployGroup: *[] | [...]

if parameter.registry != _|_ {
    registry: parameter.registry
}
if parameter.deployGroup != _|_ {
    deployGroup: parameter.deployGroup
}

if registry != "" && !strings.HasSuffix(registry, "/") {
    _base: registry + "/"
}
if registry == "" || strings.HasSuffix(registry, "/") {
    _base: registry
}

let _deployNamePrefix = "deploy-"
let _overrideNamePrefix = "override-"

// produce deploy policy acccording to deployGroup parameter
deployPolicy: [ for k, v in deployGroup {
    type: "topology"
    name: _deployNamePrefix + "\(k)"
    properties: {
        namespace: "istio-system"
        if v.clusters != _|_ {
            clusters:  v.clusters
```

Istio plguin template

```
# Code generated by KubeVela templates. DO NOT EDIT. Please edit
#Definition source cue file: vela-templates/definitions/interna
apiVersion: core.oam.dev/v1beta1
kind: TraitDefinition
metadata:
  annotations:
    definition.oam.dev/description: use istio to manage traffic
  name: canary-traffic
  namespace: vela-system
spec:
  appliesToWorkloads:
    - deployments.apps
  podDisruptive: true
  schematic:
    cue:
      template: |

        outputs: virtualService: {
            apiVersion: "networking.istio.io/v1alpha3"
            kind: "VirtualService"
            metadata: {
                name:      parameter.originName
                namespace: context.namespace
            }
            spec: {
               hosts: [parameter.originName]
               http: [{route: [
               {destination: {
                  host: parameter.originName
                  subset: "withoutCanary"
               }}]}]
            }
        }
```

canary release trait

```
apiVersion: core.oam.dev/v1beta1
kind: WorkflowStepDefinition
metadata:
  name: canary-rollout-content
  namespace: vela-system
spec:
  schematic:
    cue:
      template: |-
        import ("vela/op")
        import "strings"

        parameter: {
                target: {
                    cluster: string
                    namespace: string
                }
                component: string
                versions: {
                   oldVersion: string
                   newVersion: string
                }
                header?: {
                    matchRule: *"exact" | "prefix" | "regex"
                    key: string
                    value: string
                }
                uri?: {
                    matchRule: *"exact" | "prefix" | "regex"
                    path: string
                }
        }
```

canary release workflow based on request contents

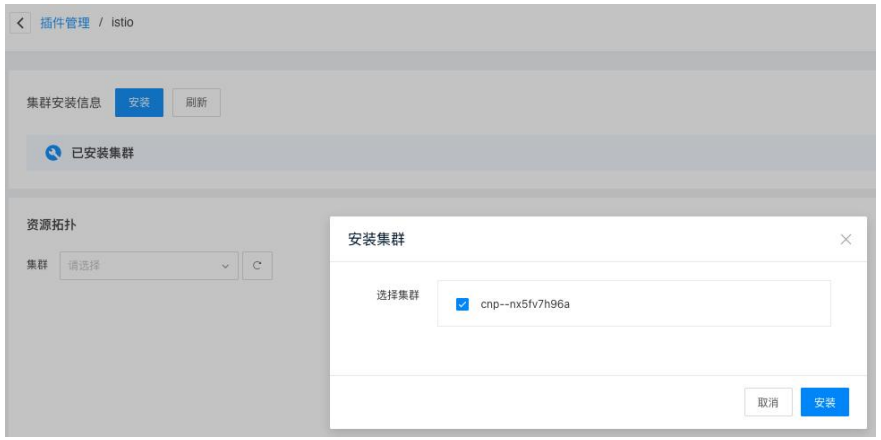
```
apiVersion: core.oam.dev/v1beta1
kind: WorkflowStepDefinition
metadata:
  name: canary-rollout-weight
  namespace: vela-system
spec:
  schematic:
    cue:
      template: |-
        import ("vela/op")
        import "strings"

        parameter: {
                target: {
                        cluster: string
                        namespace: string
                }
                component: string
                traffic: [...{
                        revision: string
                        weight:   int
                }]
        }
```

canary release workflow based on traffic weights

Product Form

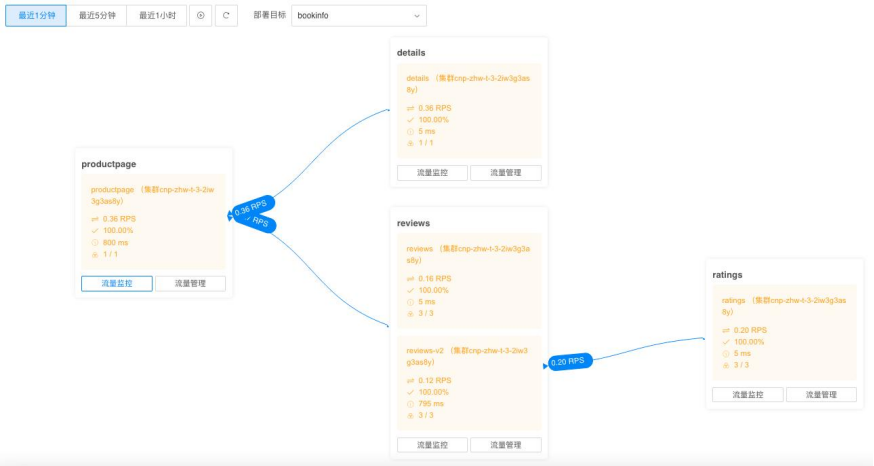
For the product team, there is no need to have in-depth knowledge of Kubernetes and Istio's underlying details. They can easily access service mesh capabilities through the platform.



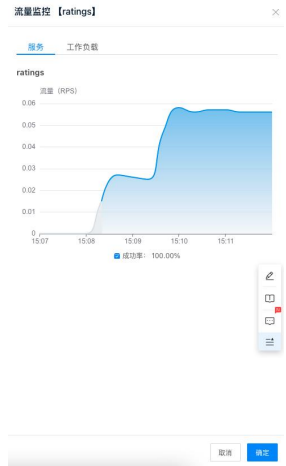
Deploy Istio via plugin

Create canary Component

Define canary workflow



Canary release service topology



Traffic Observability

Futrue Prospects

Future Prospects

- Piloting the dual-stack solution based on a single set of xDS configurations.
- Exploring Ambient Mesh
- Serverless Exploration in the Computing Network


KubeCon
CloudNativeCon
OPEN SOURCE SUMMIT
China 2023