Advanced Application Management Using Red Hat OpenShift Service Mesh

Introduction

Module Topics

- What Is a Service Mesh?
- Complexity of Distributed Applications
- How to Address the Complexity
- Evolution of Microservices
- Service Mesh Landscape
- Data Plane and Control Plane
- Istio Control Plane Components
- Red Hat® OpenShift® Service Mesh
- OpenShift Service Mesh and API Management
- OpenShift Service Mesh 2.0

What is a Service Mesh?

- Dedicated, policy-based network for service-to-service communication
- Connective tissue between services in distributed application
- Adds additional capabilities to network:
  - Traffic control, load balancing, resilience, observability, security, etc.
- Offloads capabilities from application-level libraries to infrastructure
  - Frees developers from needing to build network concerns into application
  - Empowers operators to declaratively define network behavior
- Enhances capabilities of underlying OpenShift/Kubernetes platform

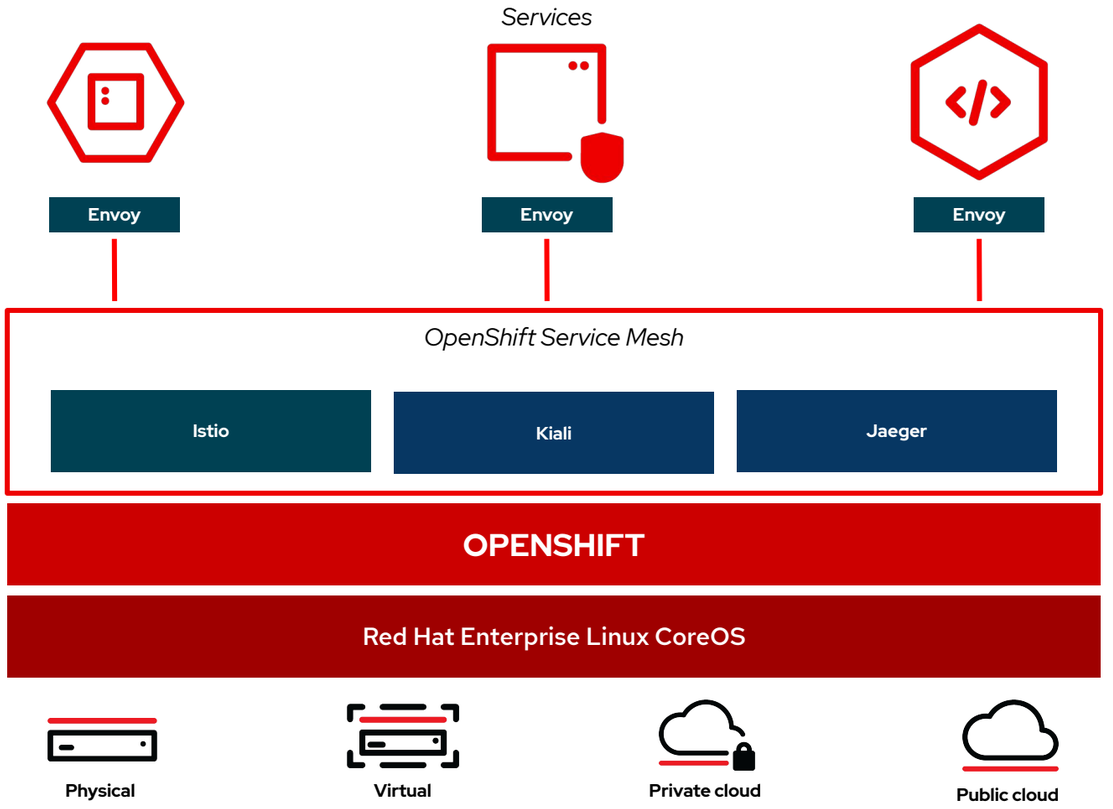
What is a Service Mesh?

Use Cases

- Connect
- Secure
- Control
- Observe

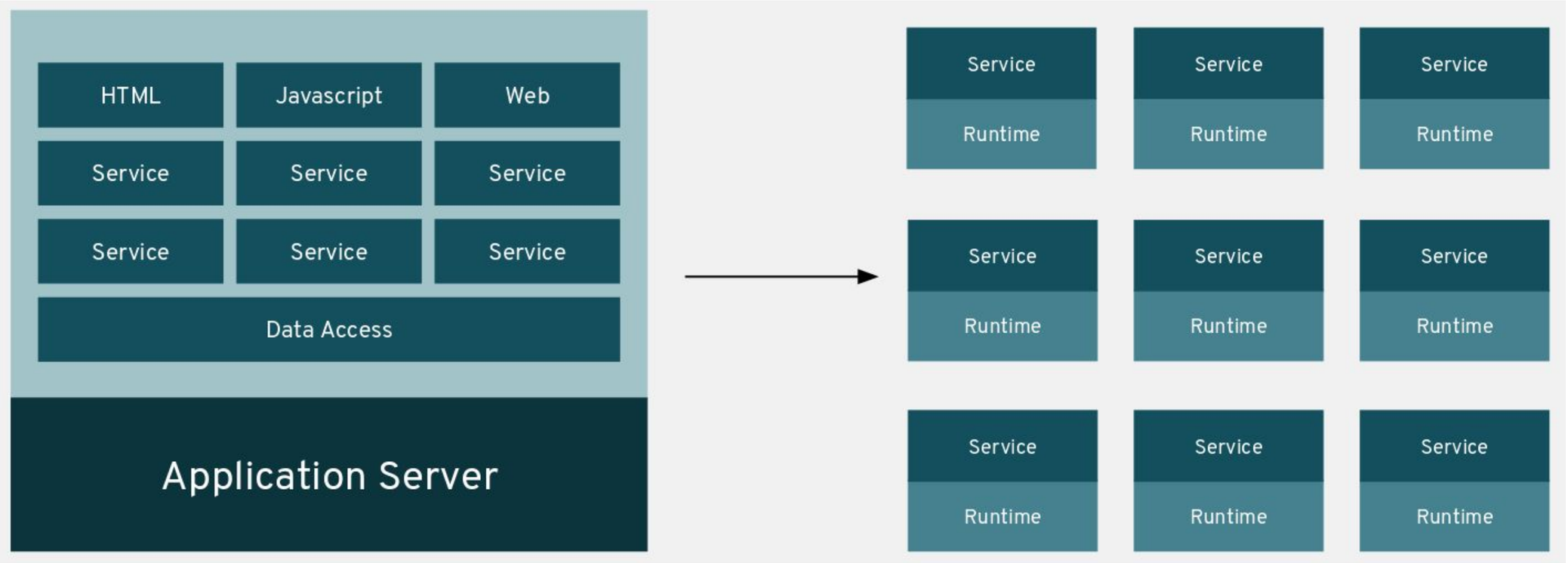
What is a Service Mesh?

- Connect services securely with zero-trust network policies.
- Automatically secure your services with managed authentication, authorization and encryption.
- Control traffic to safely manage deployments, A/B testing, chaos engineering and more.
- See what's happening with out of the box distributed tracing, metrics and logging.
- Manage OpenShift Service Mesh with the Kiali web console.



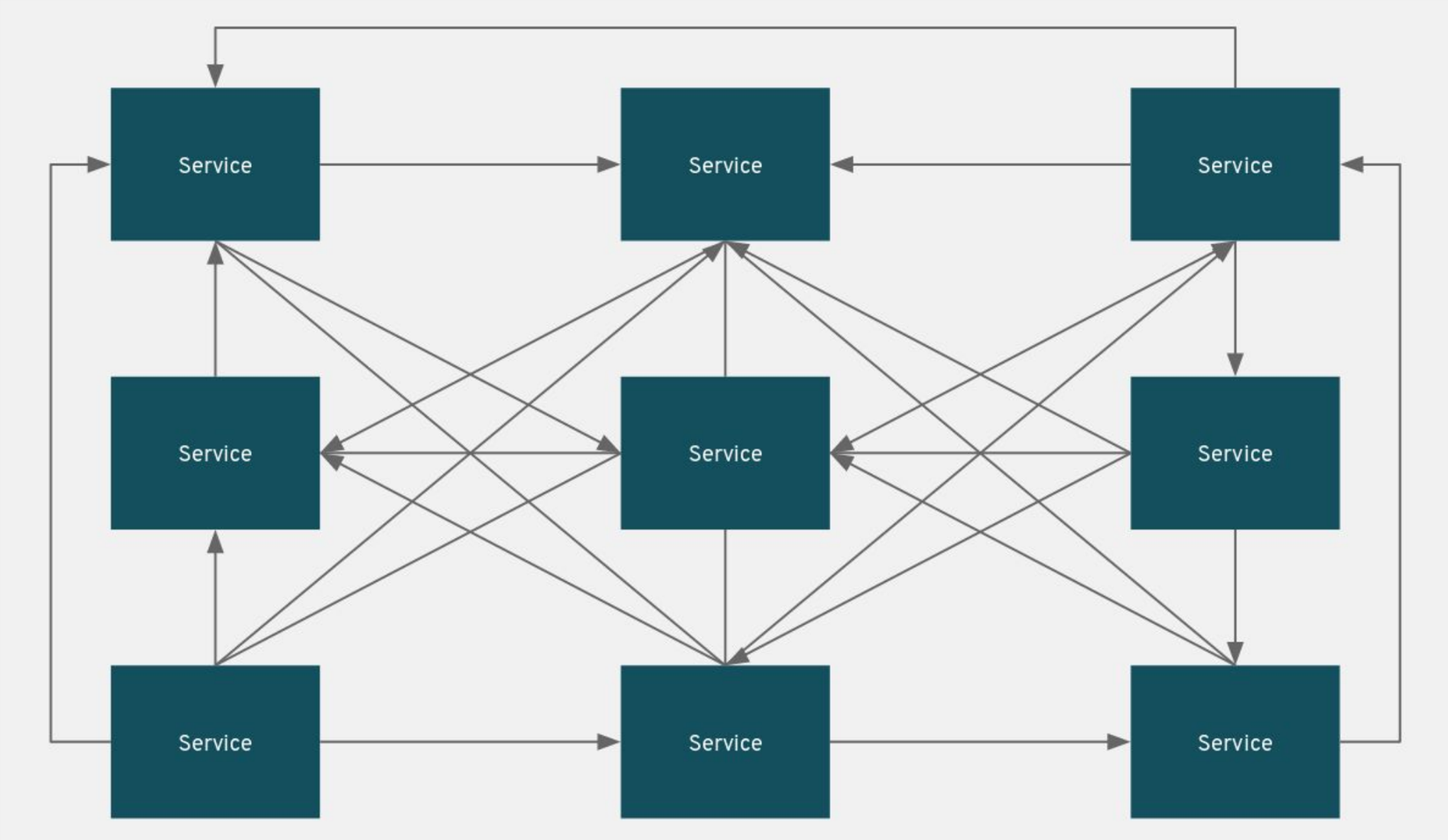
Complexity of Distributed Applications

Distributed Architecture



Complexity of Distributed Applications

Distributed Architecture: Service Connections



Complexity of Distributed Applications

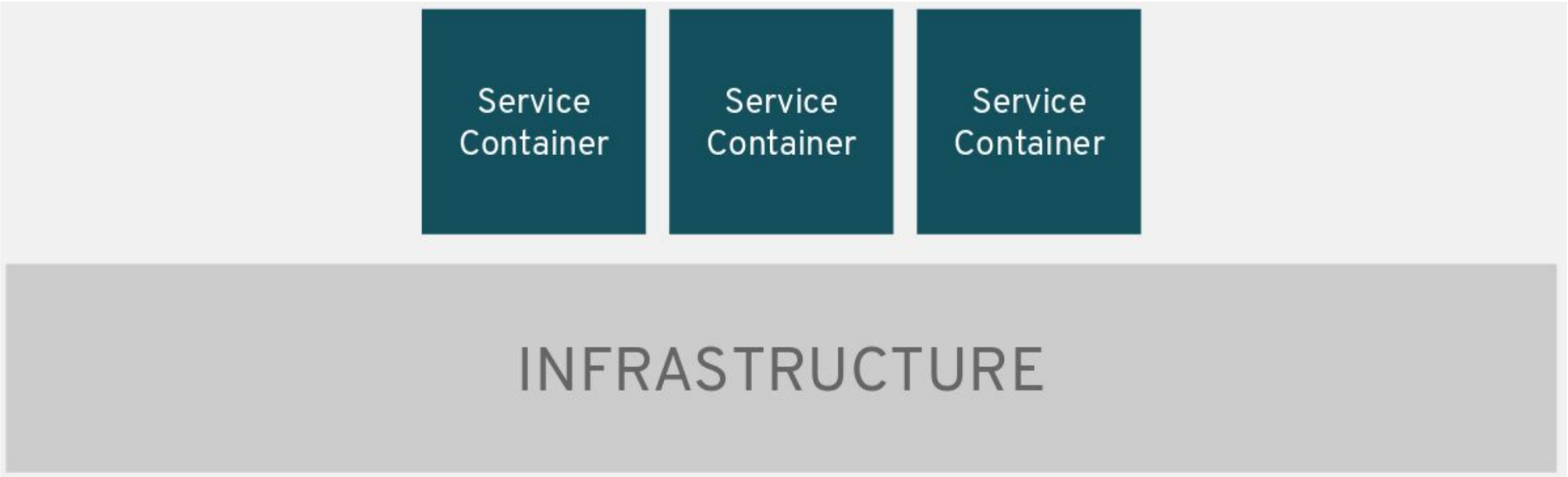
Fallacies of Distributed Applications

- Network is reliable
- Latency is zero
- Bandwidth is infinite
- Network is secure
- Topology does not change
- There is one administrator
- Transport cost is zero
- Network is homogeneous

Source: Wikipedia: Fallacies of distributed computing

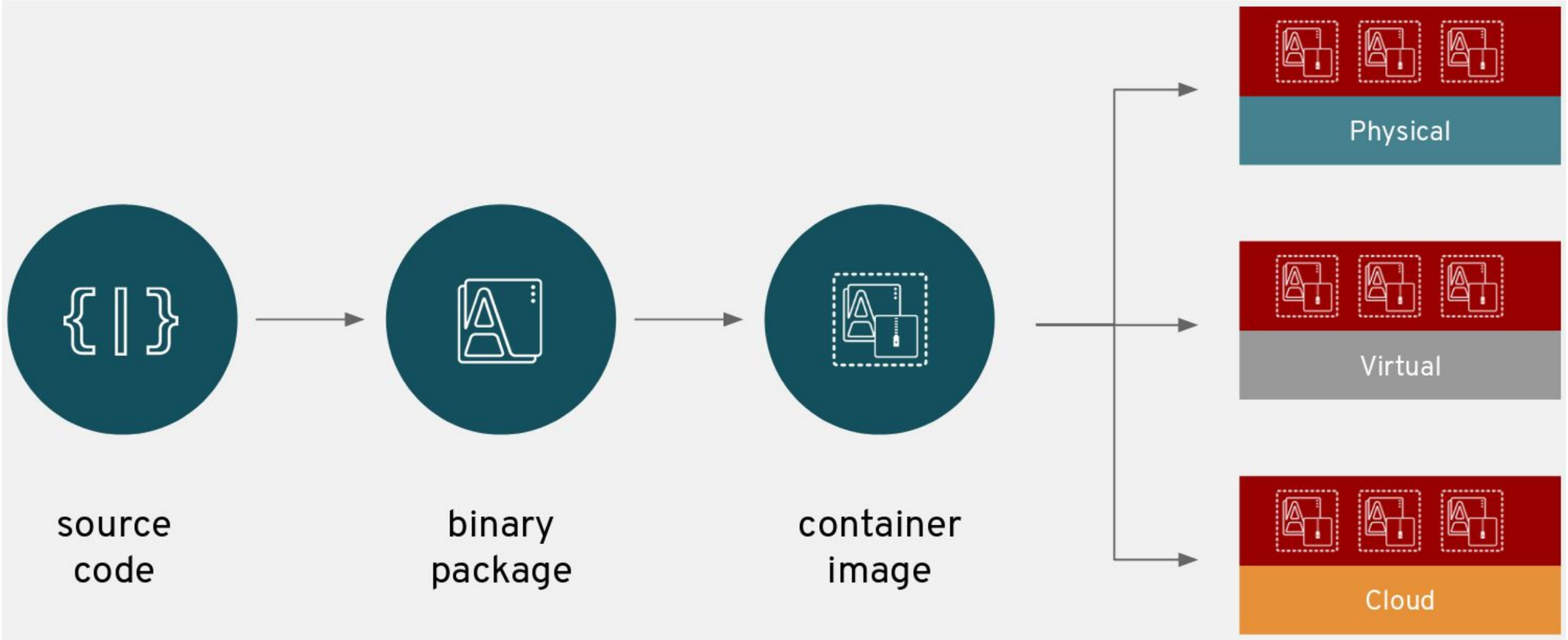
How to Address the Complexity

Deployment: Containers



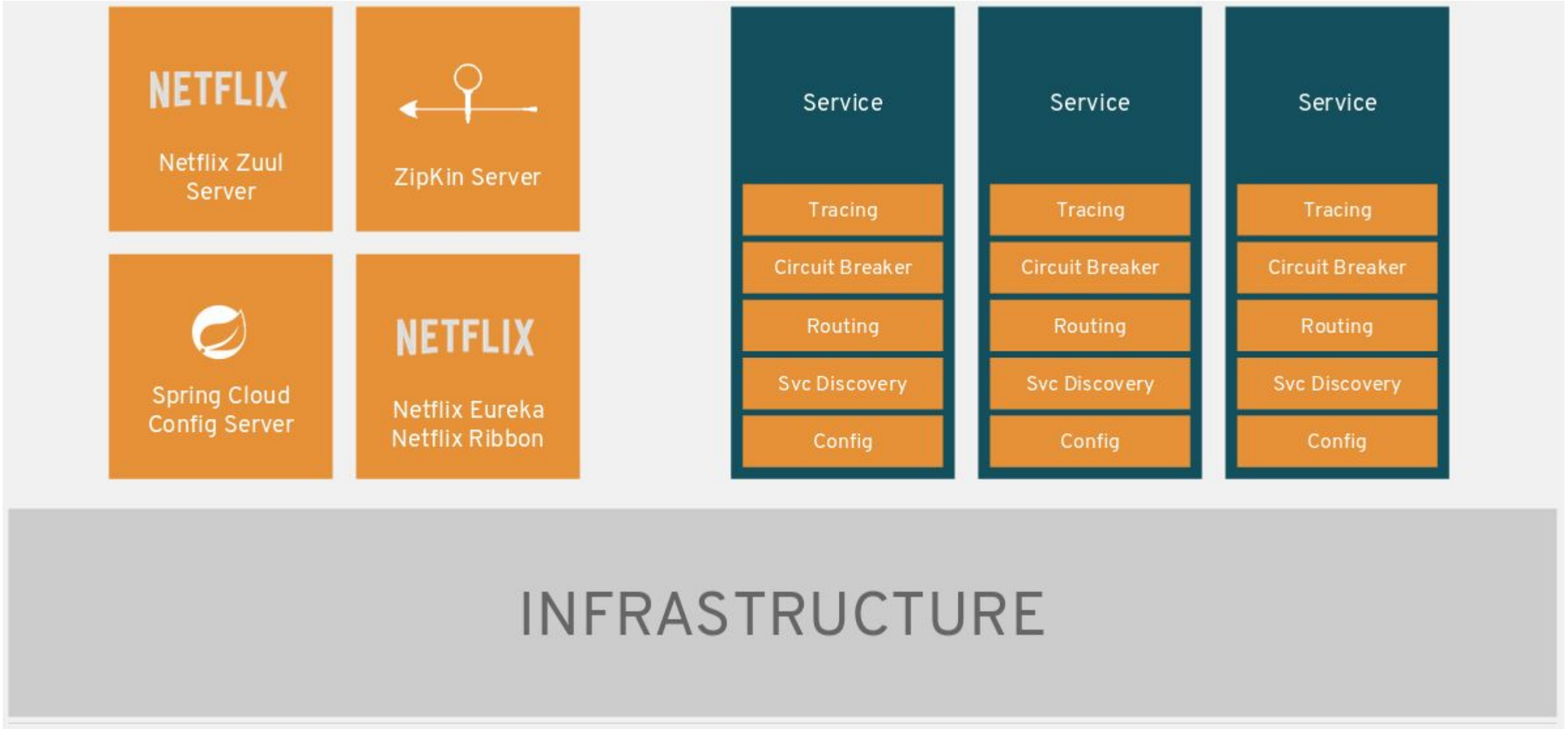
How to Address the Complexity

Application Portability



How to Address the Complexity

Configuration, Service Discovery, Dynamic Routing, Resilience, Observability



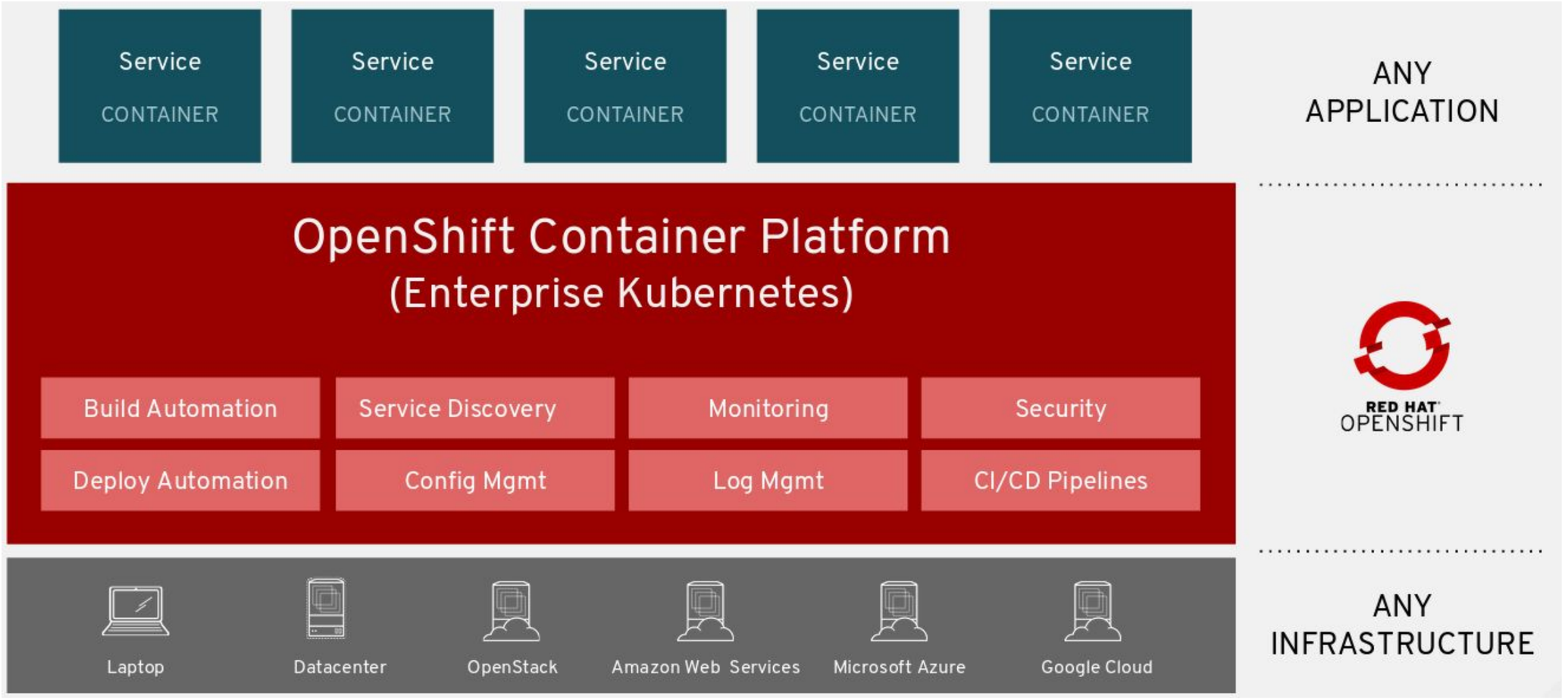
How to Address the Complexity?

Other concerns

- Polyglot applications
- Existing applications

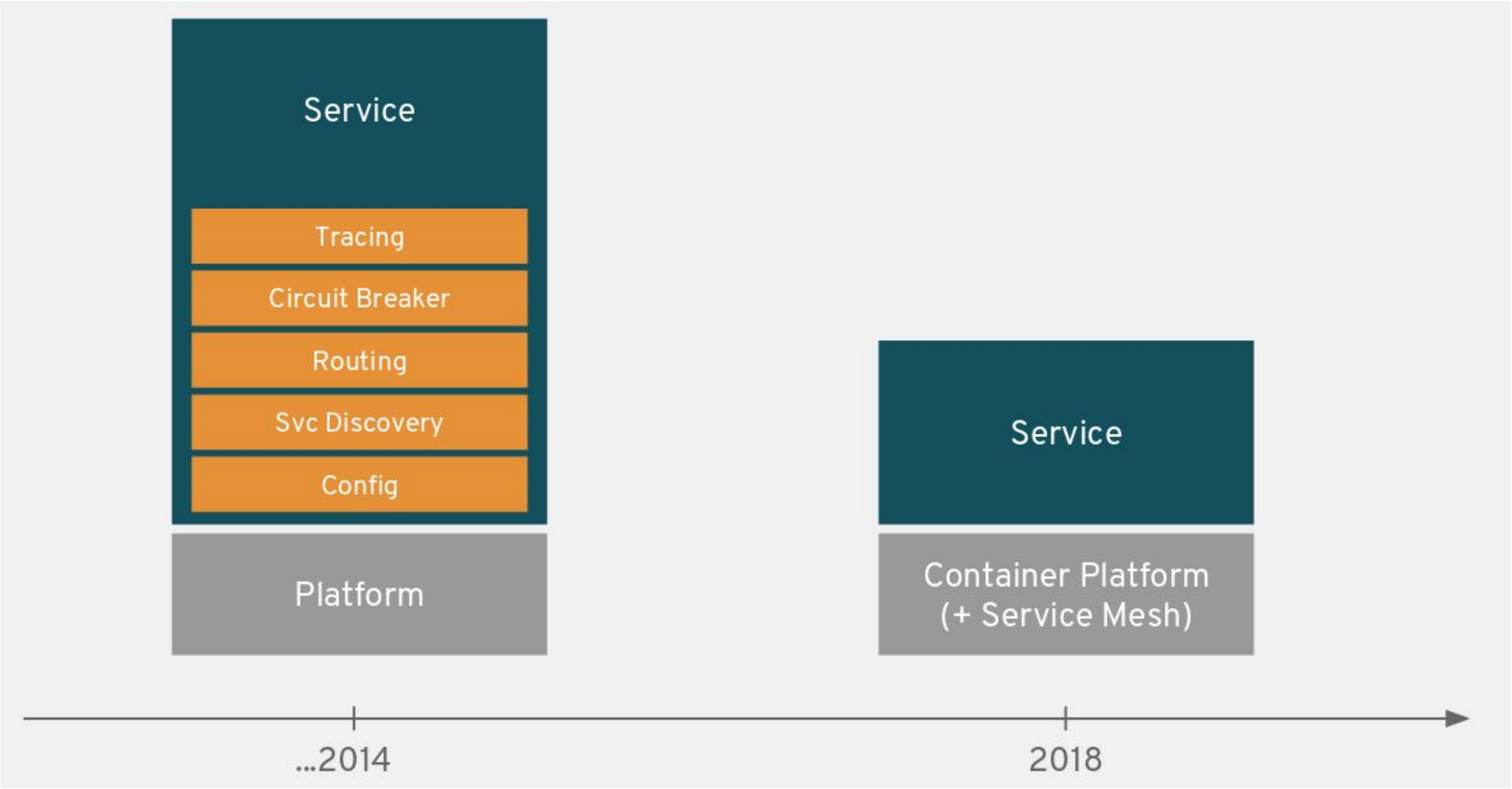
How to Address the Complexity

Infrastructure: Container Orchestration Platform



How to Address the Complexity

Applications: push concerns to infrastructure components



Service Mesh Landscape

Open Source Projects	Vendors

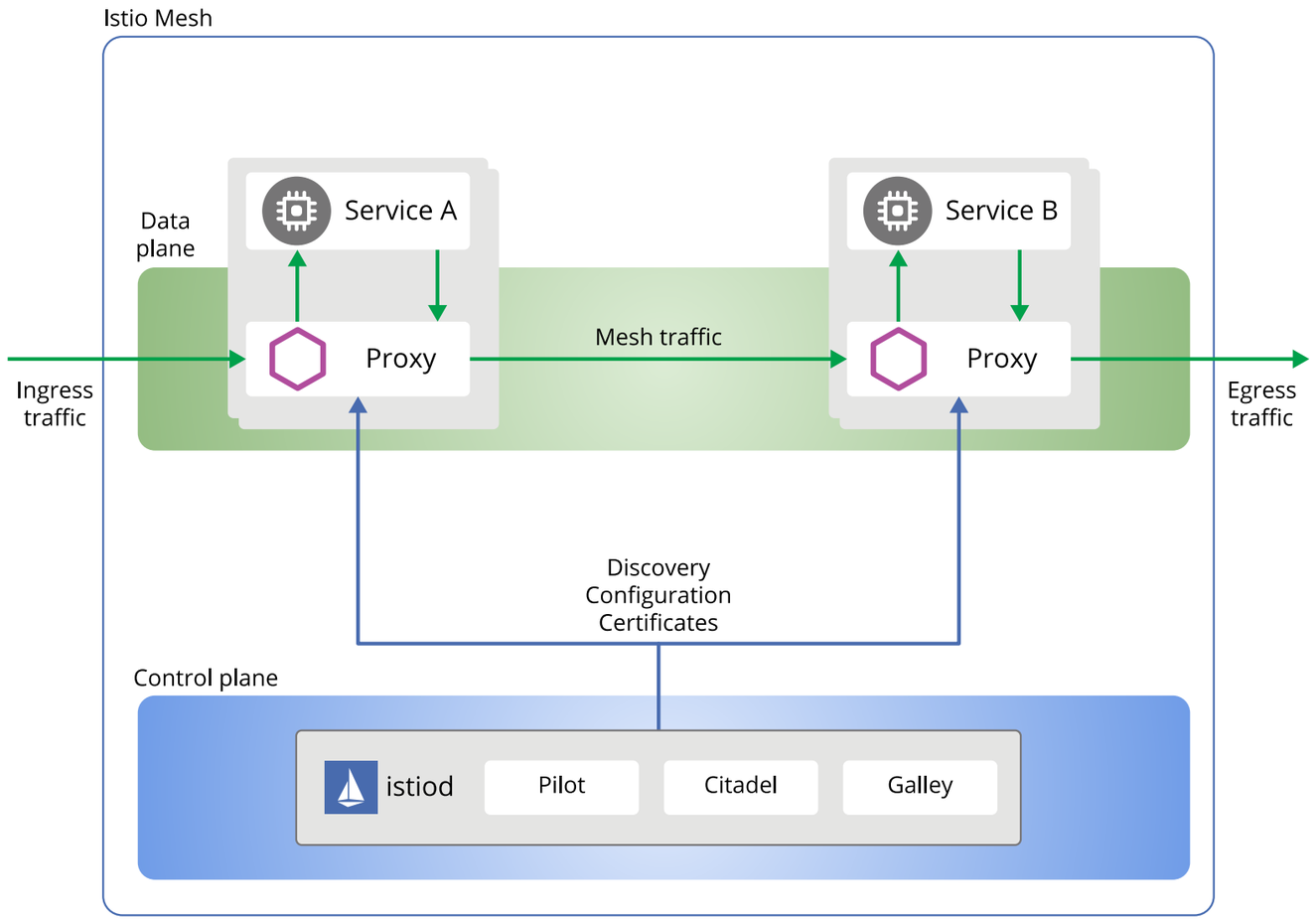
• Linkerd: Control and data plane
• Istio: Control plane
  ○ First created by Google, IBM, Lyft
  ○ Red Hat is major contributor
• Envoy: Data plane
  ○ Created by Lyft
• NGINX: Data plane
• Consul: Control and data plane
• SuperGloo: Multimesh
	• Red Hat
	• Banzai Cloud
	• Aspen Mesh: F5 networks
	• Buoyant: Linkerd
	• Solo.io: GlooMesh
	• Cloud provider-specific mesh offerings
	  — e.g., Azure Service Fabric Mesh

Red Hat OpenShift Service Mesh

- Current release: 2.0.x
- Based on upstream Istio 1.6
- Supported on Red Hat OpenShift Container Platform 4.x
- Main differences from upstream Istio:
  - Operator based installation
  - Multi-tenancy support
  - No cluster-admin privileges required for service mesh operator and application owner
- Kiali: Service Mesh Management Console

Istio Architecture

Control Plane & Data Plane



Data Plane

- Collection of service proxies in service mesh
- Intercepts all inbound and outbound traffic
- Implements policies defined in and managed by control plane
- Injected as sidecar in services that are part of service mesh
- Default service proxy for Istio: Envoy
    - Highly performant, written in C++
    - First-class support for HTTP/2 and gRPC
    - L7 observability and traffic control

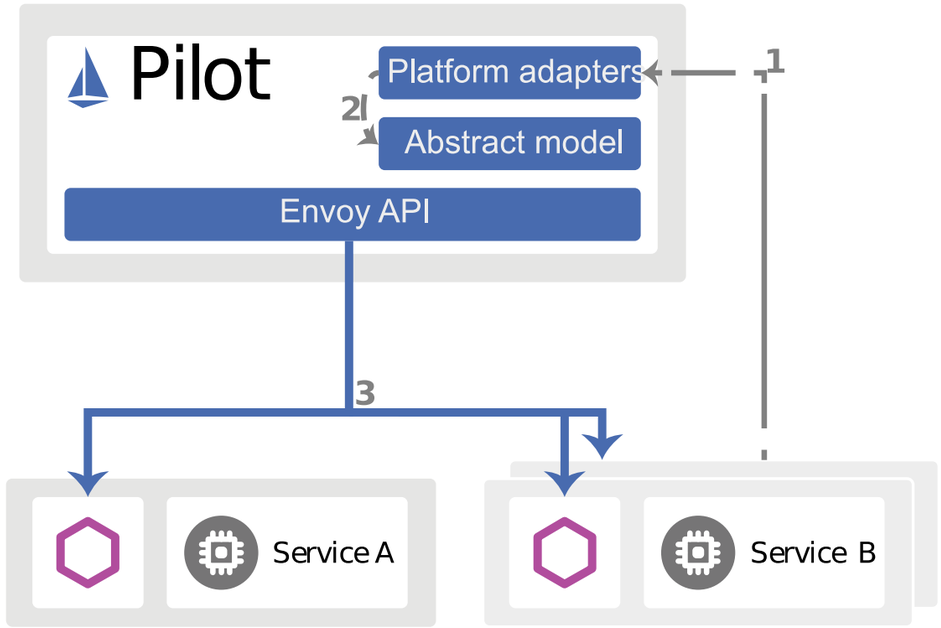
Control Plane

- Single point of administration for service proxies
  - Service proxies need programmatic configuration to be efficiently managed
  - Service proxies must have their configuration updated in real-time as services are rescheduled across environment
- Provides policy and configuration for services in mesh
- Combines set of isolated stateless sidecar proxies into service mesh
- Provides security via certificate issuance and rotation
- Unifies telemetry collection

Control Plane Components

Istiod Pilot

- Produces configuration for data plane service proxies
- Provides:
  - Service discovery for Envoy sidecars
  - Traffic management capabilities for intelligent routing
  - Resiliency
- Converts high-level routing rules that control traffic behavior into Envoy-specific configurations
  - Propagates configurations to sidecars at runtime



Control Plane Components

Istiod Citadel

- Enables strong service-to-service and end-user authentication with built-in identity and credential management.
- Policies enforced based on service identity rather than on relatively unstable network identifiers
- Provides access control to services using authorization policies
- Acts as a Certificate Authority (CA) and generates certificates to allow secure mTLS communication in the data plane.

OpenShift Service Mesh Multi-Tenancy

- Support for soft multi-tenancy
  - Several instances of system can run side-by-side in isolated manner
  - Each instance serves a tenant
- Several Istio control planes can run on single Kubernetes cluster, forming multiple meshes
- Cluster administrator has visibility over all meshes
- Tenant's service mesh administrator has control only over specific Istio instance
- Isolation between meshes provided by Kubernetes namespaces and RBAC
  - Namespace can belong to only one mesh
- Specific CRDs:
  - ServiceMeshControlPlane
  - ServiceMeshMemberRoll

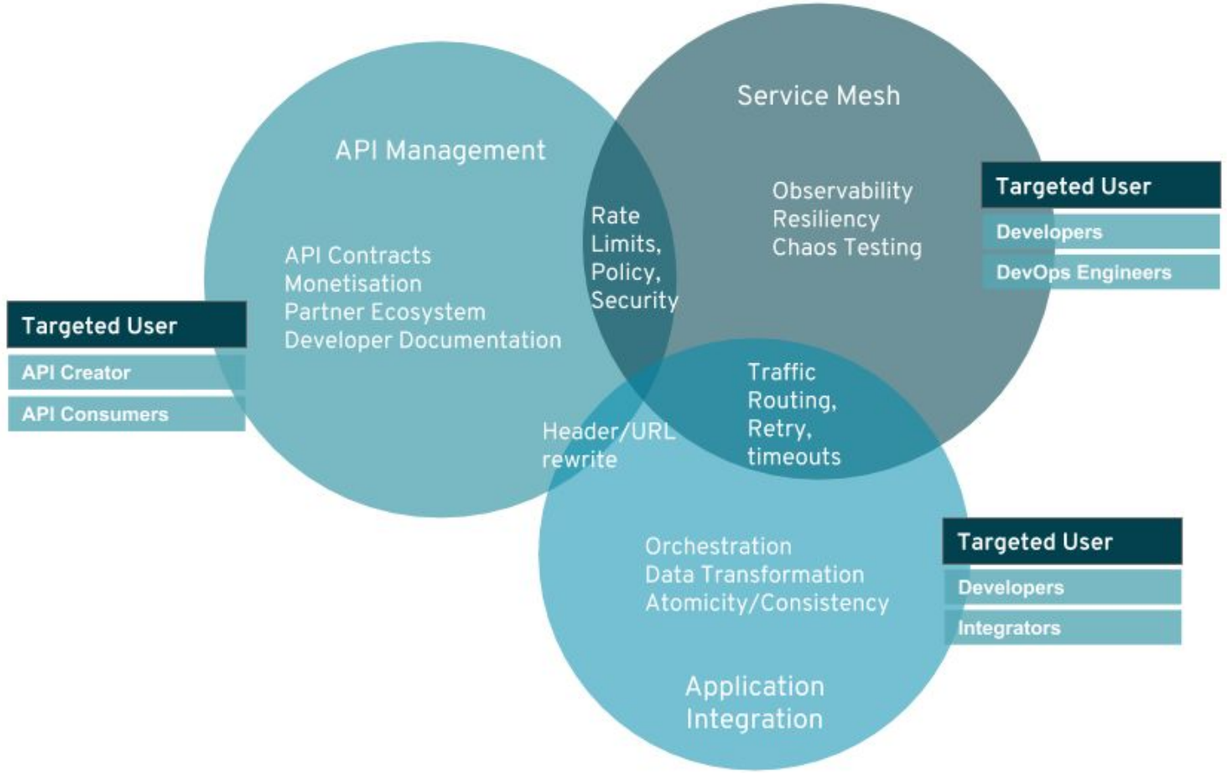
OpenShift Service Mesh Multi-Tenancy

Namespace Isolation

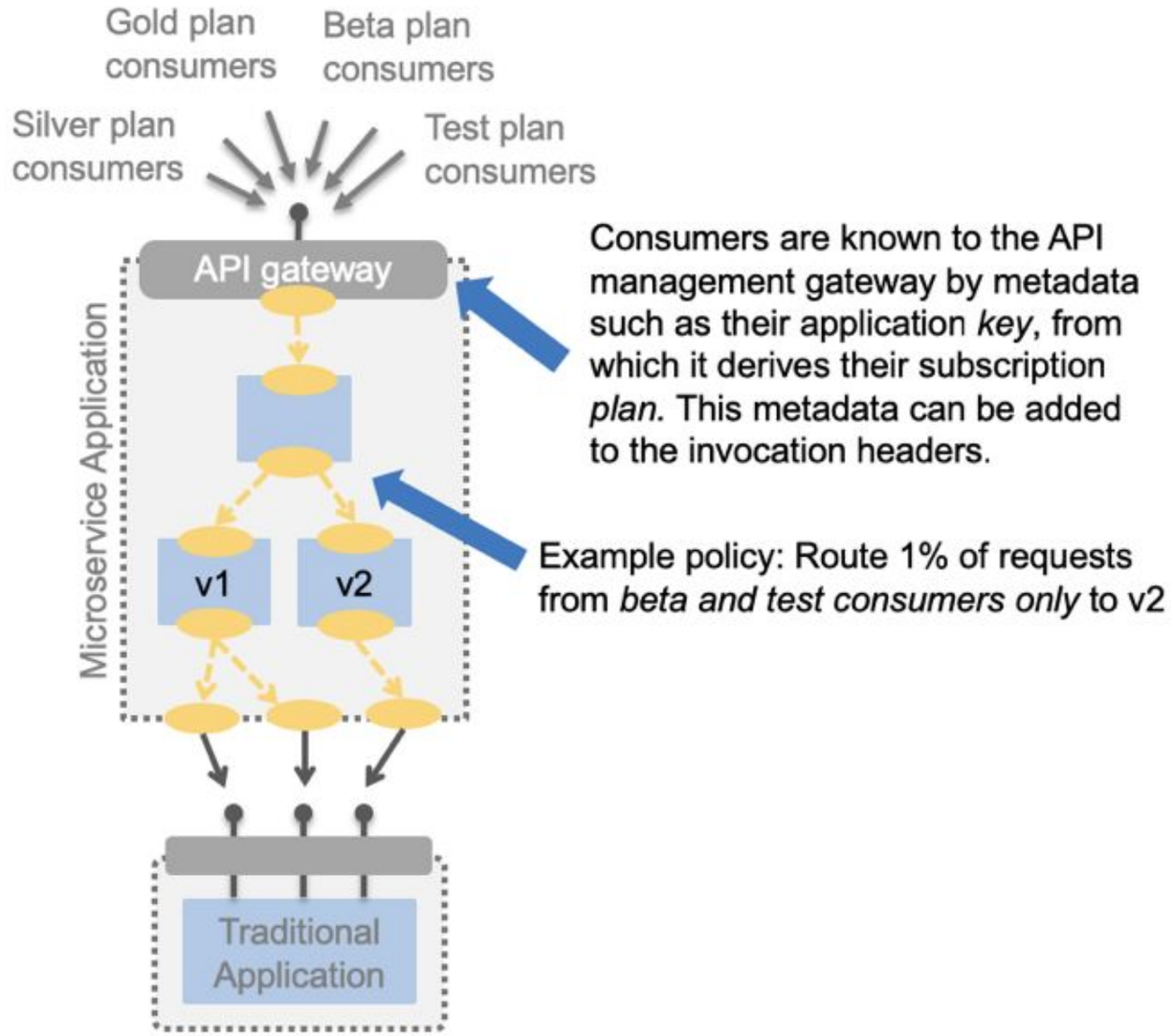
- NetworkPolicy: Default SDN isolation mode for pod network
  - Default: All pods in project are accessible from other pods and network endpoints
  - NetworkPolicy resources used to isolate pods or projects from each other
- OpenShift Service Mesh creates NetworkPolicy resource in each member namespace of control plane
- Each NetworkPolicy resource:
  - Allows ingress to all pods from other members and control plane
  - Denies traffic from all other projects
- Multi-tenant isolation mode: All namespaces isolated from each other
  - Namespace networks must be joined together explicitly
  - Command: oc adm pod-network join-projects
  - OpenShift Service Mesh joins network for each member project to network of control plane project

Service Mesh and API Management

- Complementary technologies
- API management layer mostly concerned with *north-south* traffic
- OpenShift Service Mesh mostly concerned with *east-west* traffic
- Each provides distinct capabilities, with some overlap



Service Mesh and API Management

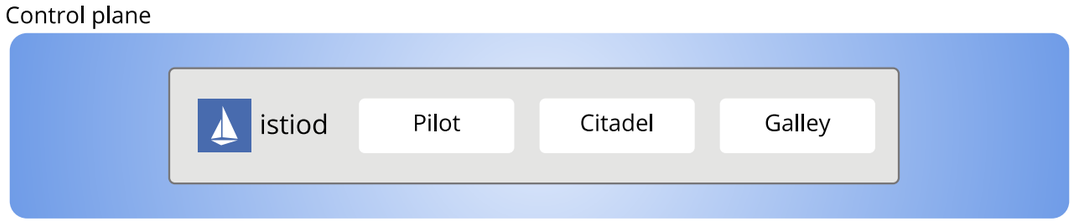


source: https://developer.ibm.com/apiconnect/2018/11/13/service-mesh-vs-api-management/

OpenShift Service Mesh 2.0

Control Plane Architecture

- Consolidates the Istio control plane components (Pilot, Galley, Citadel) into a single binary known as **istiod**.
- Benefits:
  - Simplifies installation, upgrades and management of the Control Plane.
  - Reduces the Control Plane's resource usage and startup time.
  - Improves Control Plane performance due to a reduction in inter-control plane communication over networking.



OpenShift Service Mesh 2.0

Secret Discovery Service

- Secret Discovery Service (SDS) is a more secure and performant mechanism for delivering secrets to Envoy side car proxies.
  - Removes the need to use Kubernetes Secrets, which have well known security risks.
  - Improves performance during certificate rotation, as proxies no longer require a restart.

OpenShift Service Mesh 2.0

Telemetry V2

- The new Telemetry architecture (v2) is built using WebAssembly extensions.
- Provides a significant performance improvement - reducing metrics collection latency.
    - Previously the central Mixer component could become a bottleneck.

Module Summary

- What Is a Service Mesh?
- Complexity of Distributed Applications
- How to Address the Complexity
- Evolution of Microservices
- Service Mesh Landscape
- Data Plane and Control Plane
- Istio Control Plane Components
- Red Hat® OpenShift® Service Mesh
- OpenShift Service Mesh and API Management
- OpenShift Service Mesh 2.0