Managing microservices with Istio Service Mesh

Rafik Harabi, INNOVSQUARE

SRECon EMEA 2019

Quick survey before we start

- Who are already using kubernetes?
- Who are developing microservices ?
- Who are using client library approach to implement microservices common concerns (Circuit Breaker, Retry...) ?
- Who are using Istio or any other service mesh technology?

Moving to microservices network challenges

- Network Reliability
- Fault tolerance and resiliency
- Monitoring and Observability

Challenges deep-dive

Network Reliability

Service have to handle the network facts:

- Network latency / bandwidth
- Transport cost
- Topology and administration

Fault Tolerance

Service have to be able to handle outright failure and timeouts:

- Avoid cascading failure
- Retries
- Circuit breaking

Monitoring

We have to:

- monitor the delivered microservices and their interactions
- Trace requests and identify potential hotspots

The evolution of microservices frameworks: from NetFlix OSS to Istio

NetFlix OSS

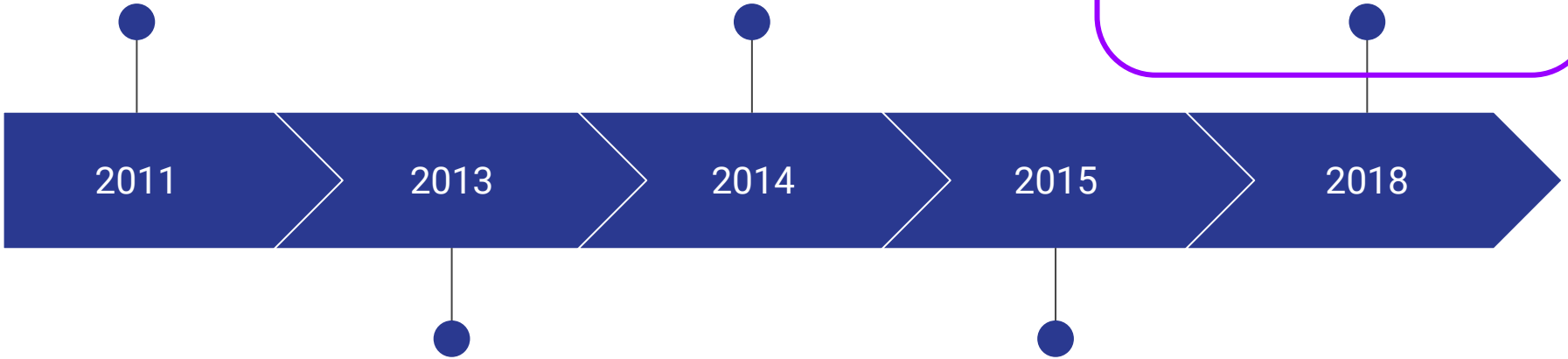
first microservices patterns and libraries open-sourced

Kubernetes

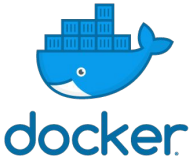
Workload orchestration

Istio

Service mesh



Docker



Containerization

Spring Cloud

Enterprise microservice framework for Java

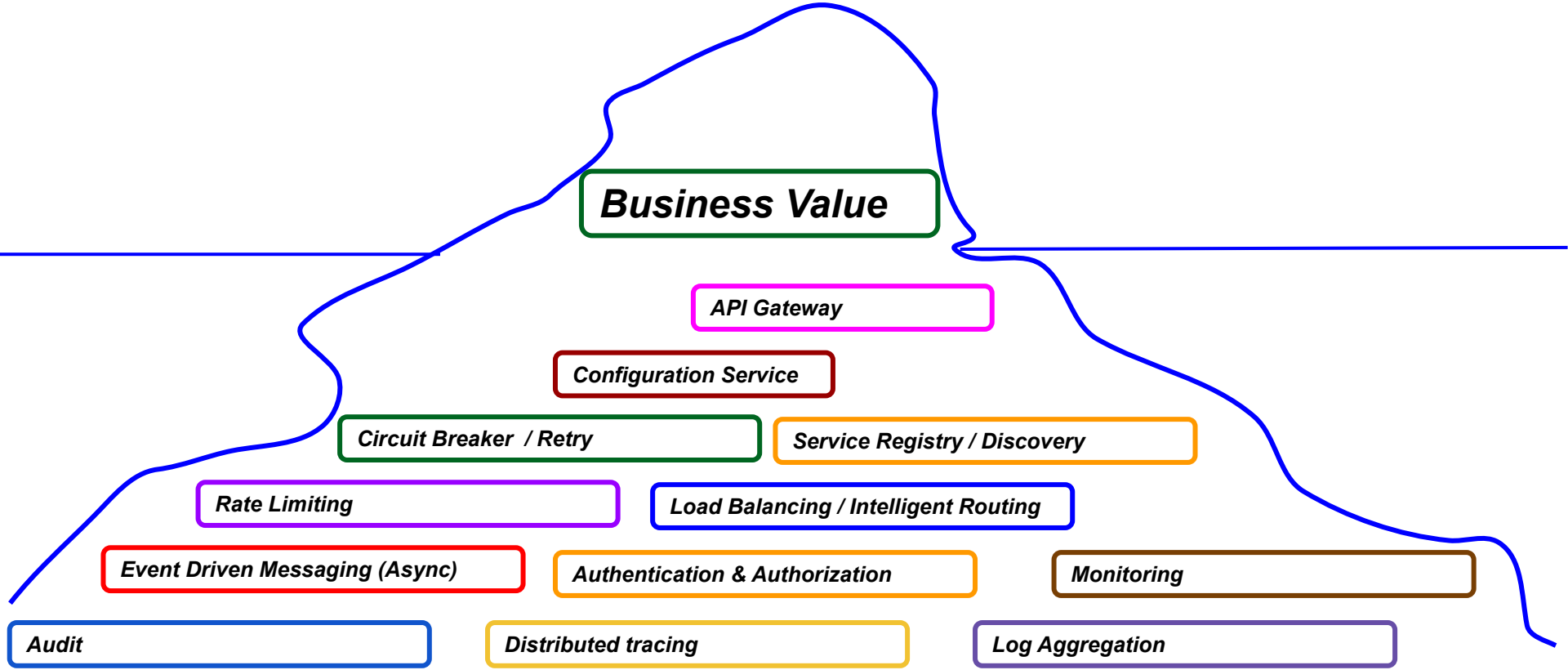
Microservices challenges

- N to N communications.
- Distributed software interconnection and troubleshooting is hard.
- Containers should stay thin and platform agnostic.
- Upgrade of polyglot microservices is hard at scale.

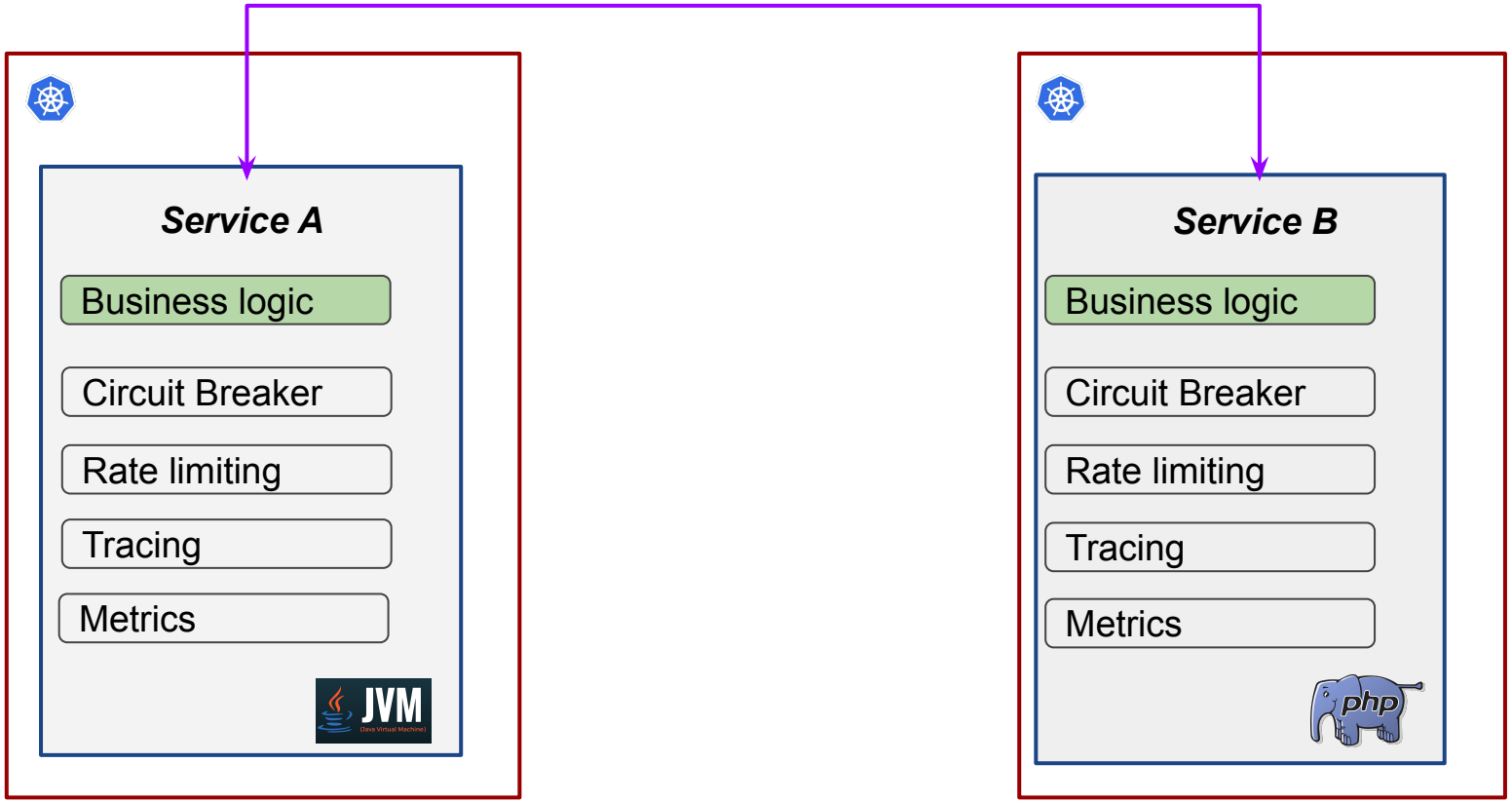
Microservices building blocks

- *Configuration Service*
- *Load Balancing / Intelligent Routing*
- *Service Registry / Discovery*
- *API Gateway*
- *Circuit Breaker / Retry*
- *Authentication & Authorization*
- *Rate Limiting*
- *Monitoring*
- *Event Driven Messaging (Async)*
- *Distributed tracing*
- *Audit*
- *Log Aggregation*

Microservices building blocks



Code oriented frameworks



Code oriented pattern

Business Values

- *Business Service*
- *Load Balancing / Intelligent Routing*
- *Authentication & Authorization*
- *Circuit Breaker/Retry*
- *Rate Limiting*

Dev Scope

Foundation

- *Configuration Service*
- *Service Registry / Discovery*
- *API Gateway*

Platform Managed

Communication

- *Event Driven Messaging (Async)*

Platform Managed

Monitoring and Observability

Platform Managed

- *Monitoring*
- *Distributed tracing*
- *Log Aggregation*
- *Audit*

Code oriented solutions limits

- Language oriented.
- Error prone (implementation).
- Hard to upgrade each microservice when system grow.
- Add technical challenges and duties to development teams.
- Different teams in the same organization may have different implementations.
- Each team should maintien his implementation.

Microservices challenges need to be solved uniformly

Desired state

- Keep microservice concerns separate from the business logic.
- The network should be transparent to applications.
- Developers should focus on delivering business capabilities and not implementing microservices common concerns.
- Microservices interconnection should be language agnostic.
- Easy to upgrade solution.

Service Mesh

Definition

A service mesh is a dedicated infrastructure layer for handling service-to-service communication. It’s responsible for the reliable delivery of requests through the complex topology of services that comprise a modern, cloud native application.

buoyant.io

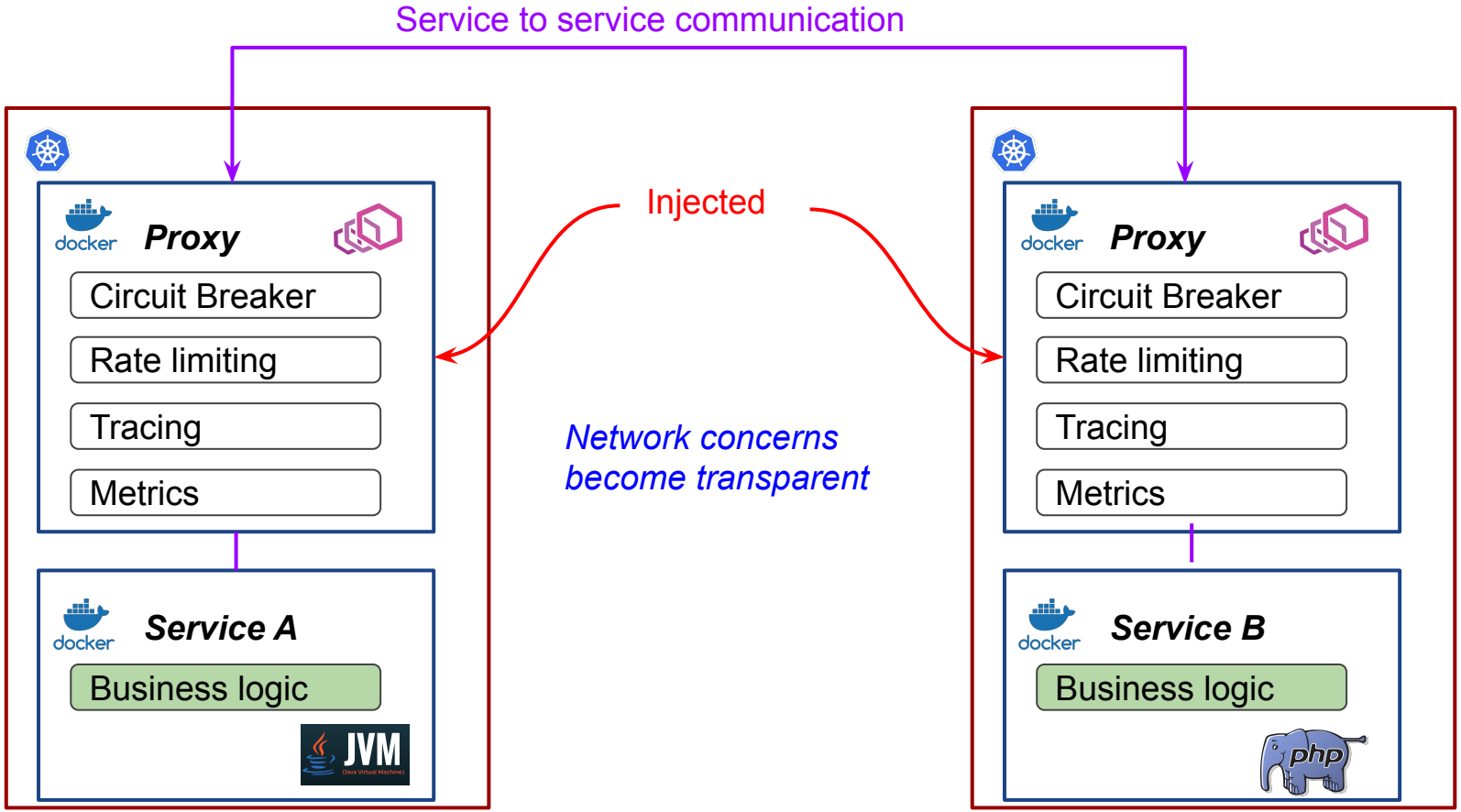
Service Mesh

The design

Each service will have its own proxy service and all these proxy services together form the “Service Mesh”. All the requests to and from each service will go through the mesh proxies.

Proxies are also known as sidecars.

Sidecar pattern



History of Istio

- Envoy proxy (Istio data plane) created by Lyft and open-sourced in 2016.
- IBM and Google launch the project in May 2017.
- First major version released in July 2018.
- Current version: **1.3**

Istio goal

Develop an open technology that provides a uniform way to connect, secure, manage and monitor a network of microservices regardless of the platform source or vendor.

Solution

Istio Promises

- Focus on business logic and spent less time with common concerns.
- No change in the service code.
- Central configuration management.
- Easy to upgrade
- Security

Istio does:

- Service discovery
- Load Balancing & Intelligent Routing
- Resiliency: Circuit Breaker & Retry
- Rate Limiting
- Authentication and Authorization
- Service to Service mTLS
- Policy enforcement
- Observability
- Monitoring metrics
- Distributed tracing

Istio does not:

- Event Driven Asynchronous communication
- Service Orchestration

Sidecar pattern

Business Values

- Business Service
- Business Service
- Business Service

Foundation

- Configuration Service
- Service Registry / Discovery
- API Gateway
- Load Balancing / Intelligent Routing
- Authentication & Authorization
- Circuit Breaker/Retry
- Rate Limiting

Dev Scope

Platform Managed

Communication

Platform Managed

Event Driven Messaging (Async)

Platform Managed

Monitoring and Observability

- Monitoring
- Distributed tracing
- Log Aggregation
- Audit

Service Discovery

Kubernetes provide service discovery, why do I need an extra one 🤔 ?

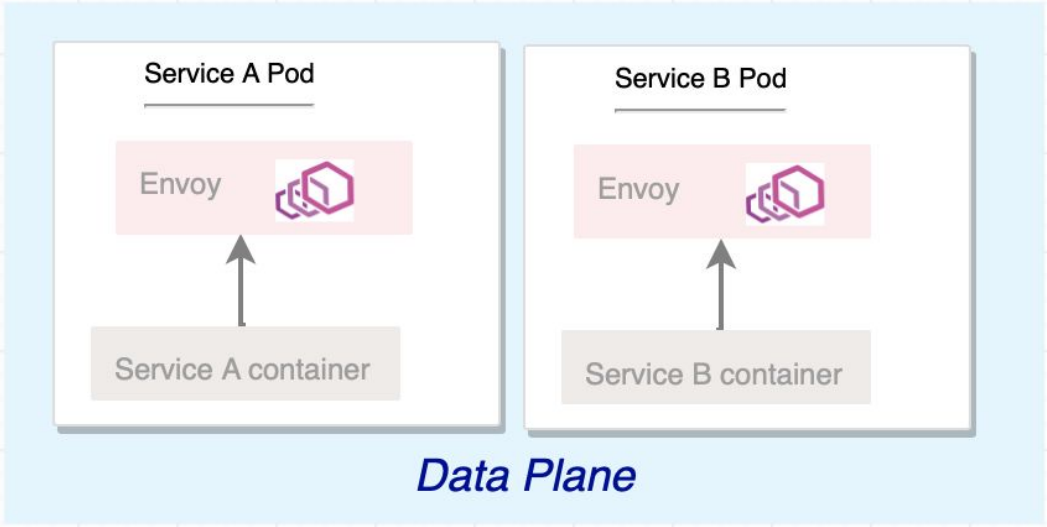
Istio supports:

- HTTP L7 filter
- HTTP L7 routing (based on http headers and cookies)
- First class HTTP/2
- gRPC support
- Fine-grained traffic splitting

Architecture


Pilot
Configuration
Citadel
mTLS
Auth
Galley
Config Ingestion
Mixer
Telemetry
Policy check
Ingress Controller
Egress Controller
Control Plane


Dashboards
Monitoring
Tracing
Observability

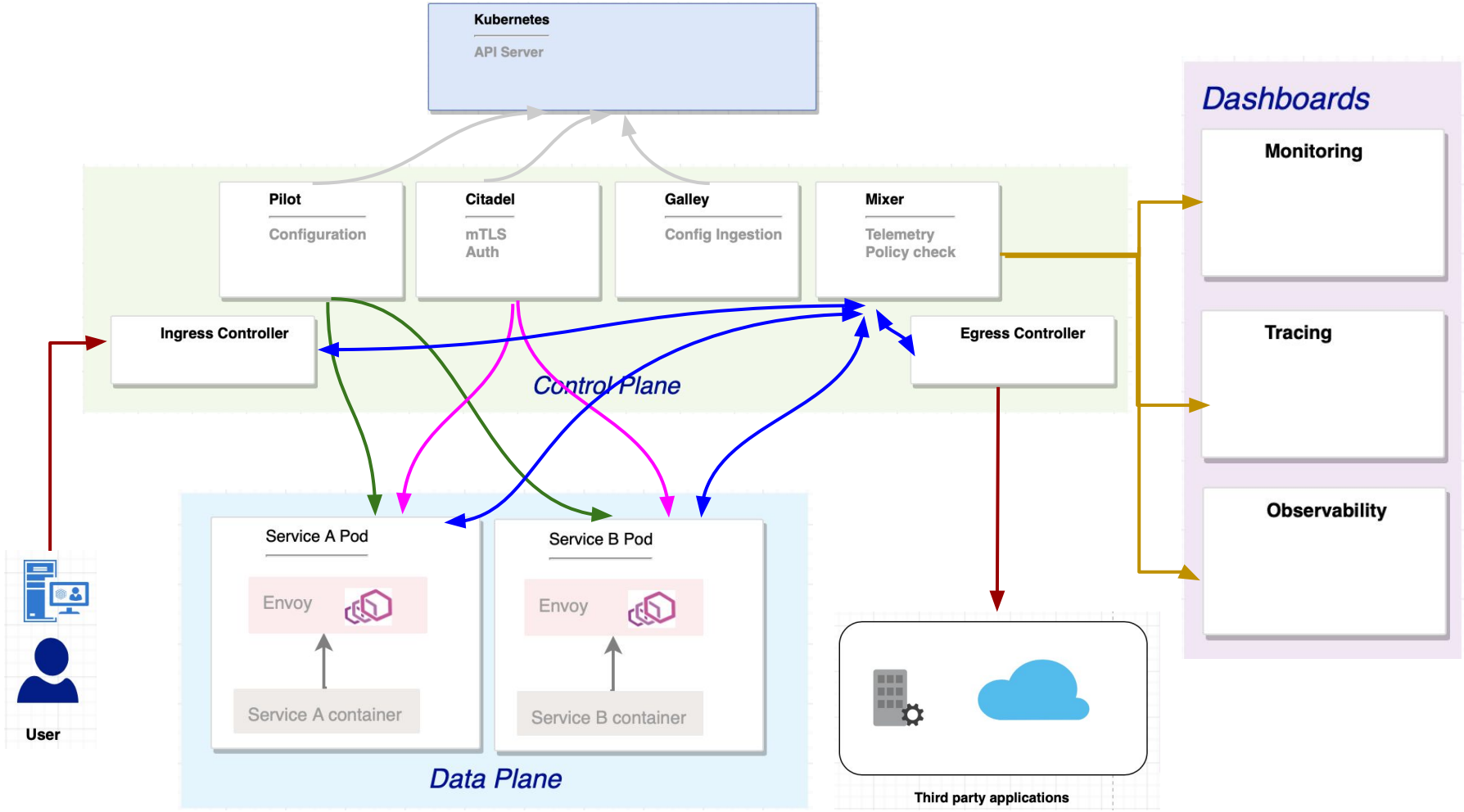

Service A Pod
Envoy
Service A container
Service B Pod
Envoy
Service B container
Data Plane

Istio building blocks 1/2

Component	Description
Pilot	Responsible for service discovery and for configuring the Envoy sidecar proxies
Citadel	Automated key and certificate management
Mixer	Istio-Policy: policy enforcement Istio-Telemetry: gather telemetry data
Galley	Configuration ingestion for istio components
Ingress Gateway	manage inbound connection to the service mesh
Egress Gateway	manage outbound connection from the service mesh
Sidecar injector	Inside sidecar for enabled pods/namespaces

Istio building blocks 1/2

Component	Description
Prometheus	Metrics collection
Grafana	Monitoring dashboard
Jaeger	Distributed tracing
Kiali	Observability dashboard


Kubernetes
API Server
Pilot
Configuration
Citadel
mTLS
Auth
Galley
Config Ingestion
Mixer
Telemetry
Policy check
Ingress Controller
Egress Controller
Control Plane
Dashboards
Monitoring
Tracing
Observability
Service A Pod
Envoy
Service A container
Service B Pod
Envoy
Service B container
Data Plane
User
Third party applications

https://www.istioworkshop.io/