Service Mesh Sharing

Thang Chung – May 2018 – v0.0.1

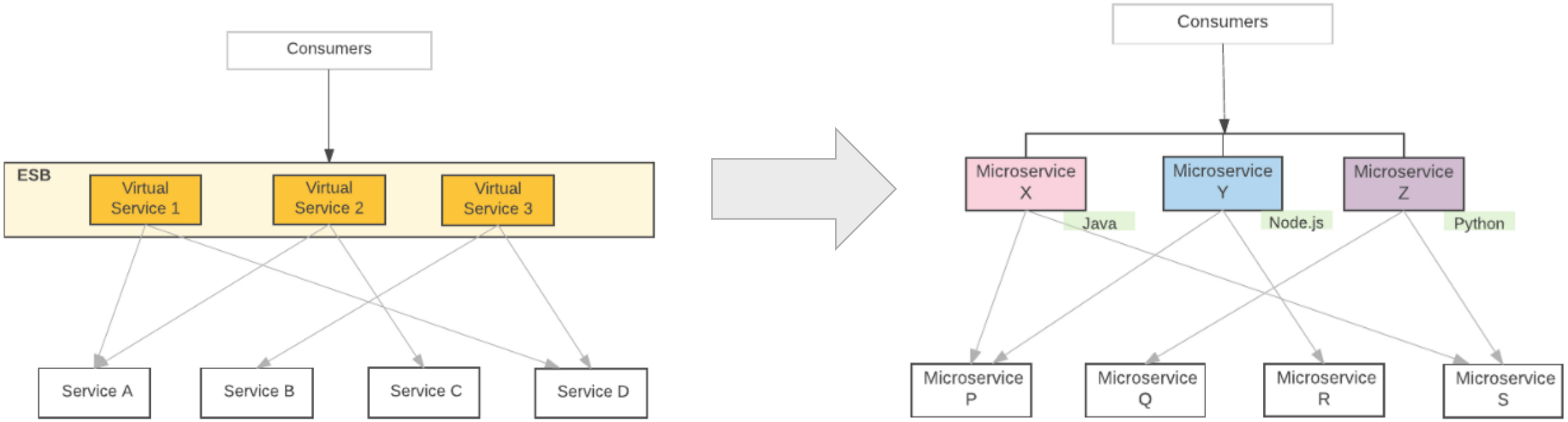
NASH TECH
The Power to Innovate

Agenda

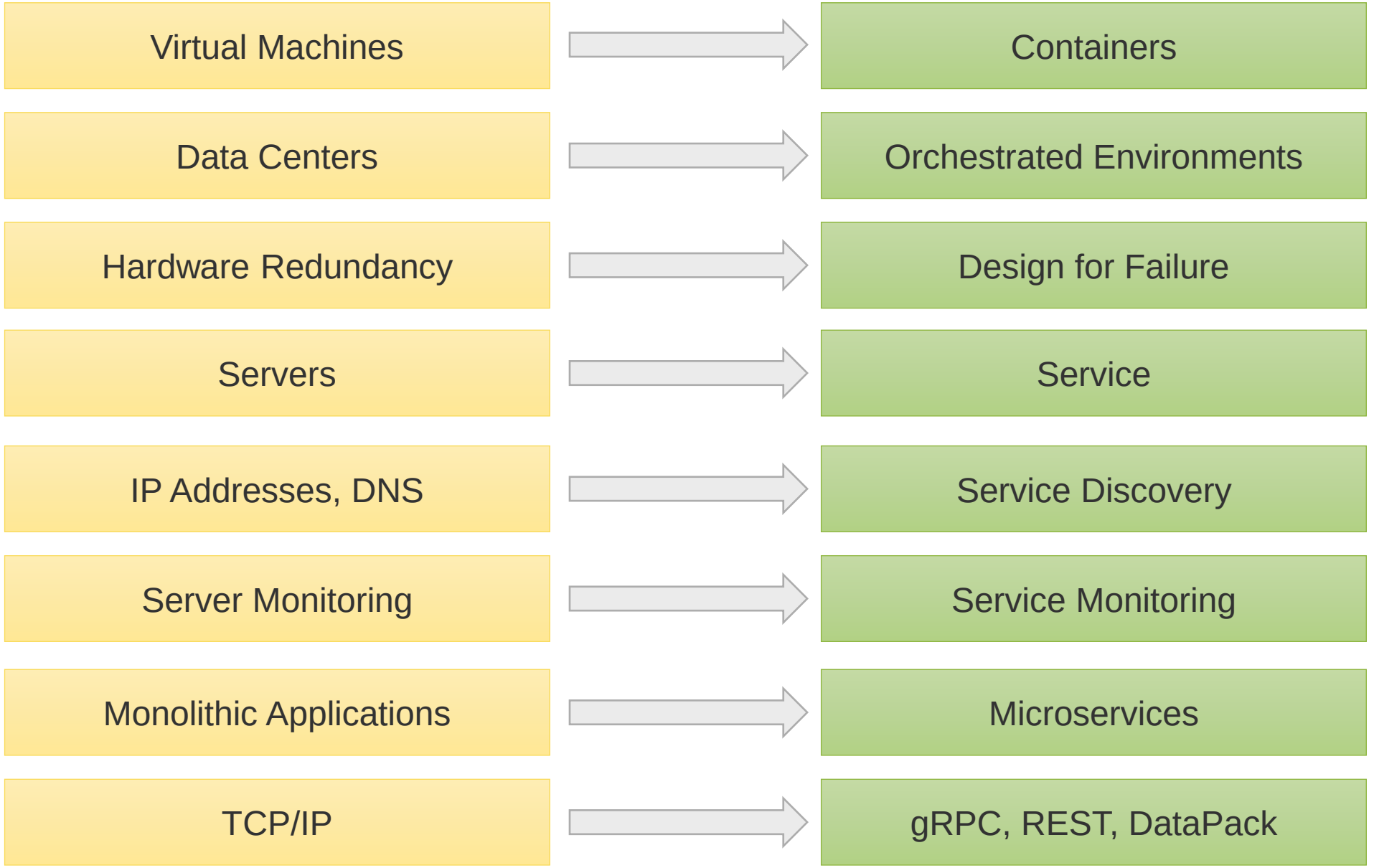
- Before Service Mesh Era
- Service Mesh
- Building Service Mesh for Microservices
- Demo
- Q&A

Before Service Mesh Era

From SOA to MSA pattern



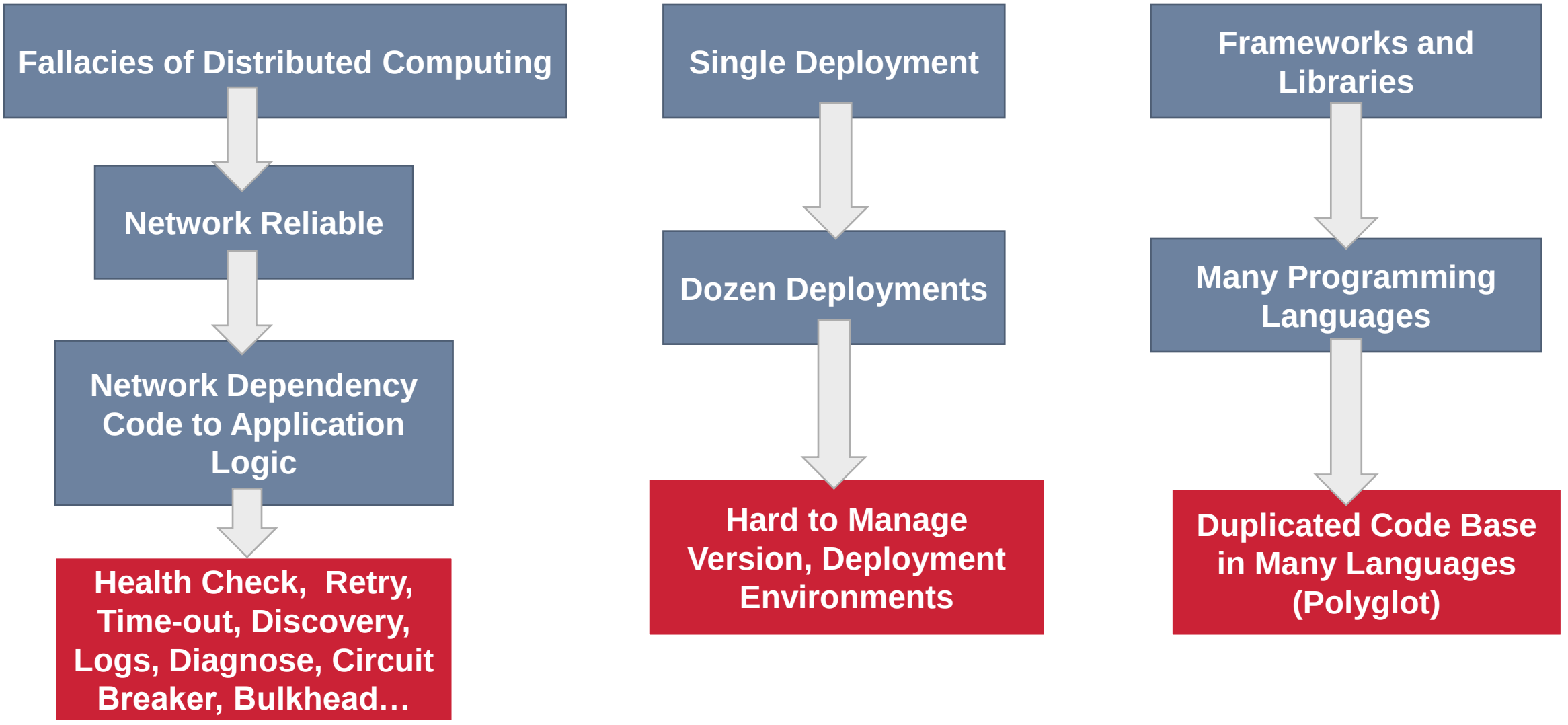
From Server to Service/Container Abstraction



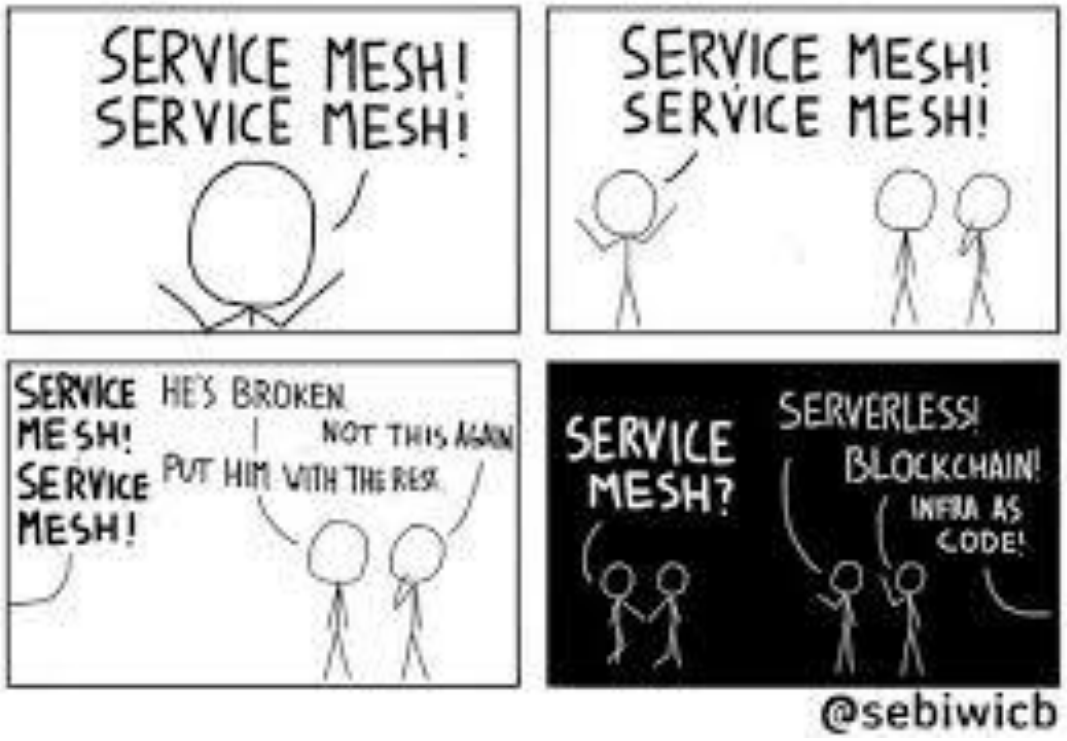
Standard Requirements for MSA

- **Load Balancing**
- **Routing**: path based routing – L7 intelligent proxy
- **Auto Service Discovery**
- **Resiliency for inter-service communications**: circuit-breaking, bulkhead, retries and timeouts, fault injection, fault handling, load balancing and failover, and Rate Limiting.
- **Observability**: metrics, monitoring, distributed logging, and distributed tracing
- **Security**: mTLS and key management
- **Inter-service communication protocols**: HTTP1.x, HTTP2, gRPC, or DataPack protocols
- **Configuration information**
- **Deployment**: native support for containers (Docker) and orchestration layer (Kubernetes, Docker Swarm…)
- **100% Uptime**

Challenge of Going Faster

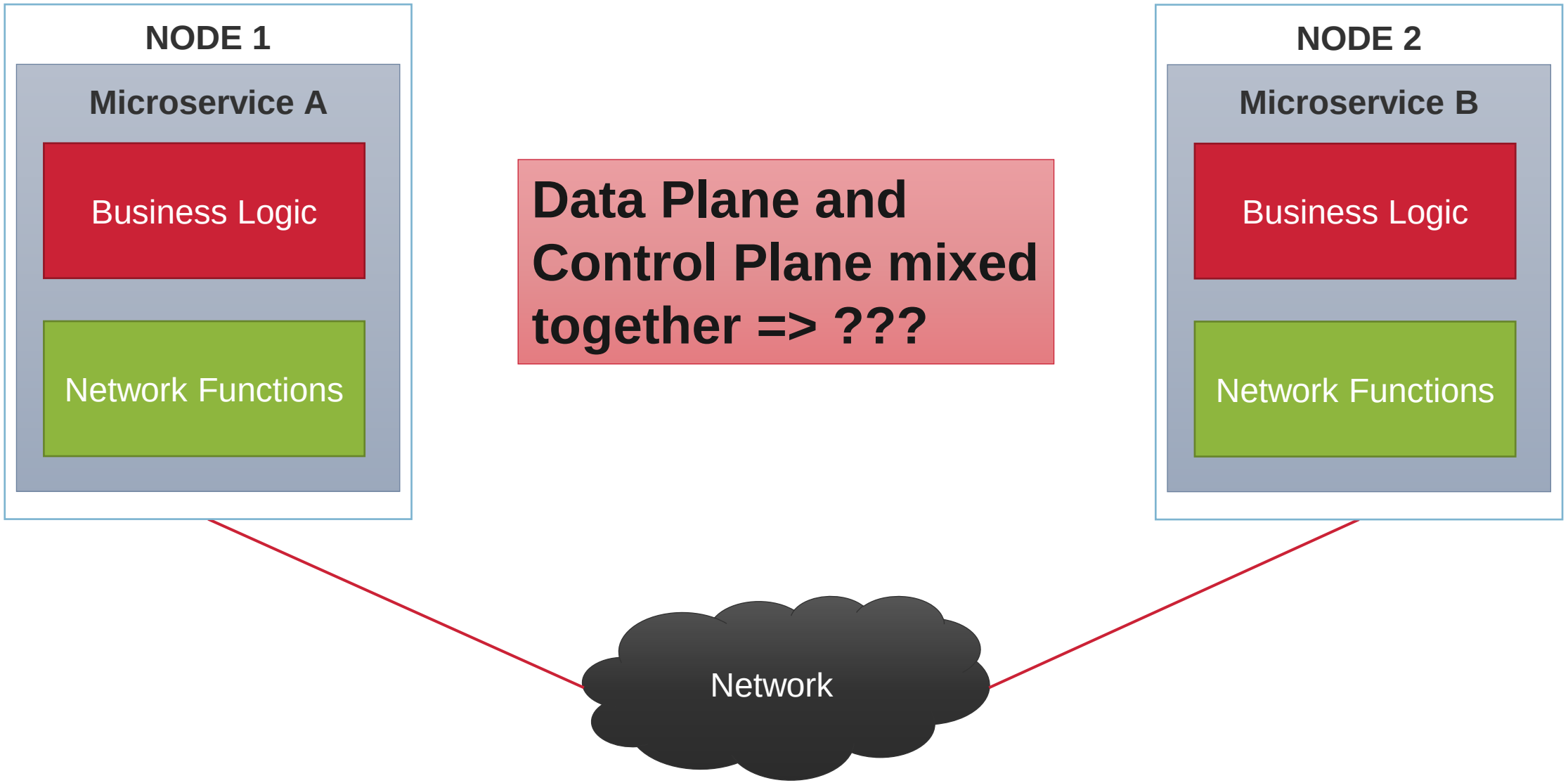


Then what kind of software architecture can help us to sort it out?

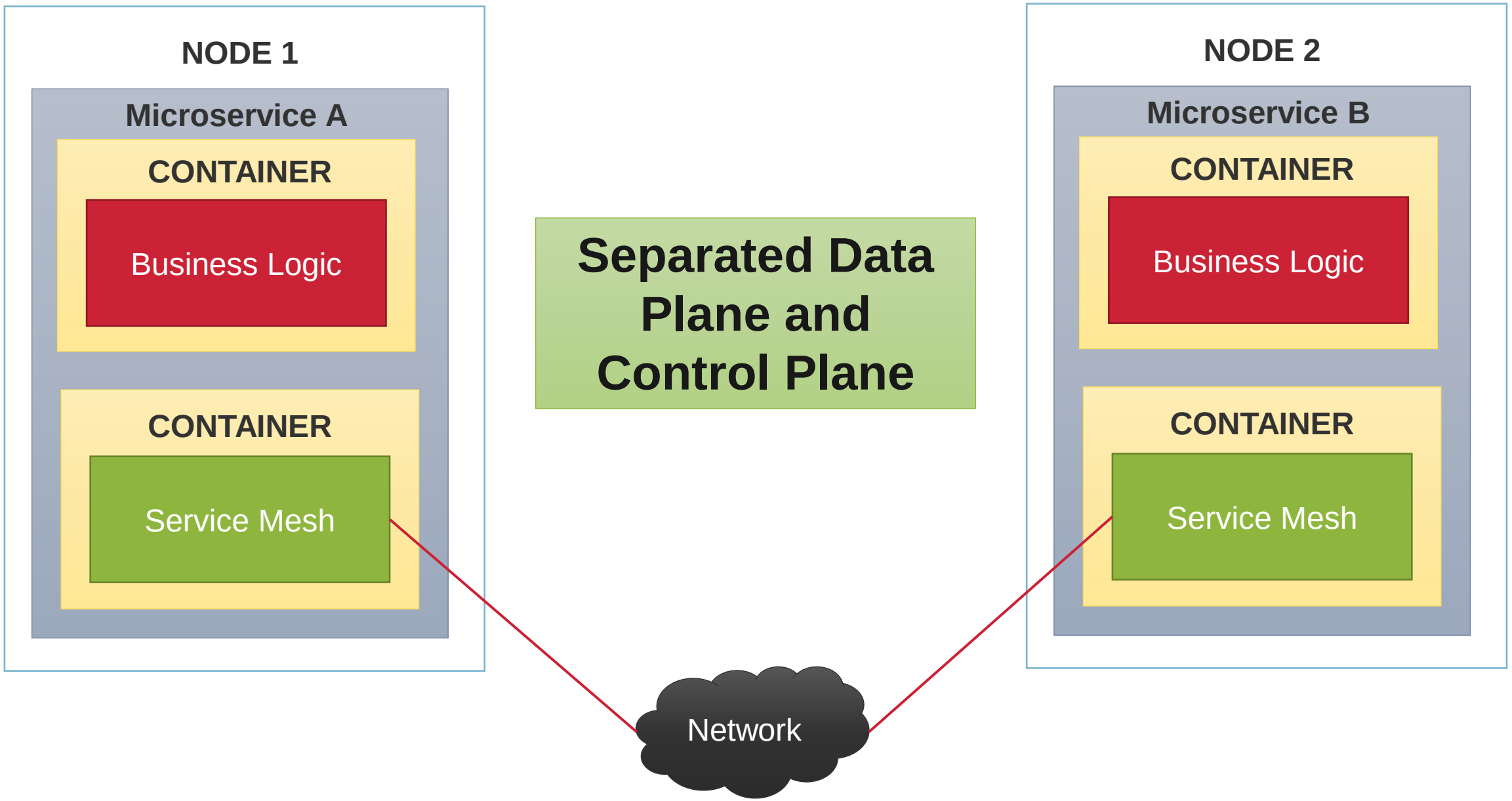


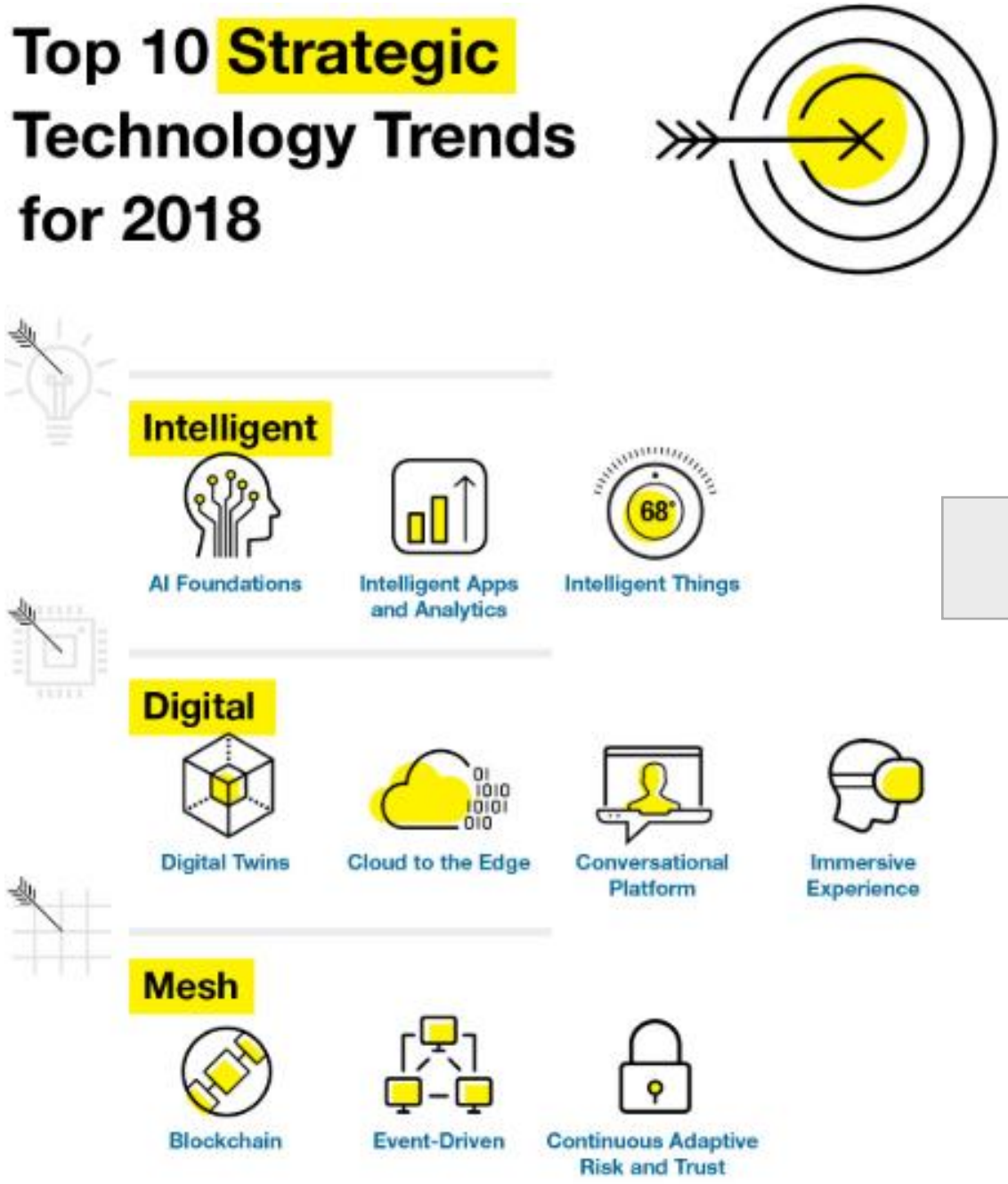
Service Mesh

MSA to Service Mesh



MSA to Service Mesh (cont.)





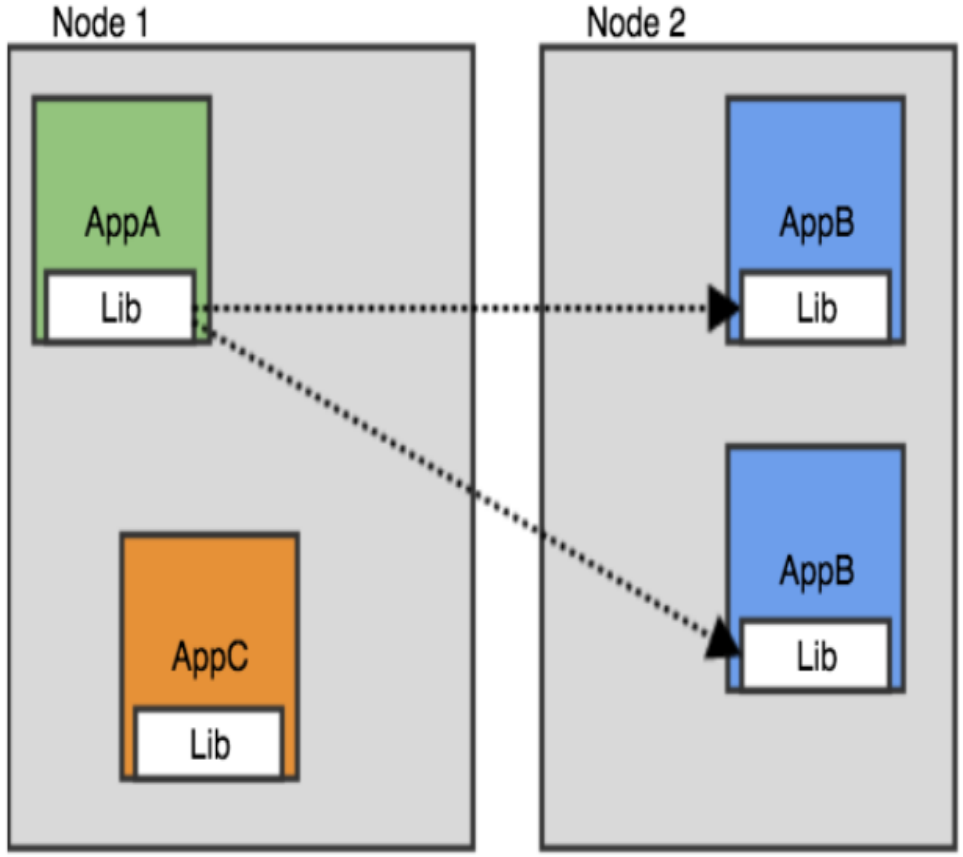
A **SERVICE MESH** offers **consistent discovery**, **security**, **tracing**, **monitoring** and **failure handling** without the need for a shared asset such as an API gateway or ESB.

https://www.thoughtworks.com/radar/techniques/service-mesh

Service Mesh Types

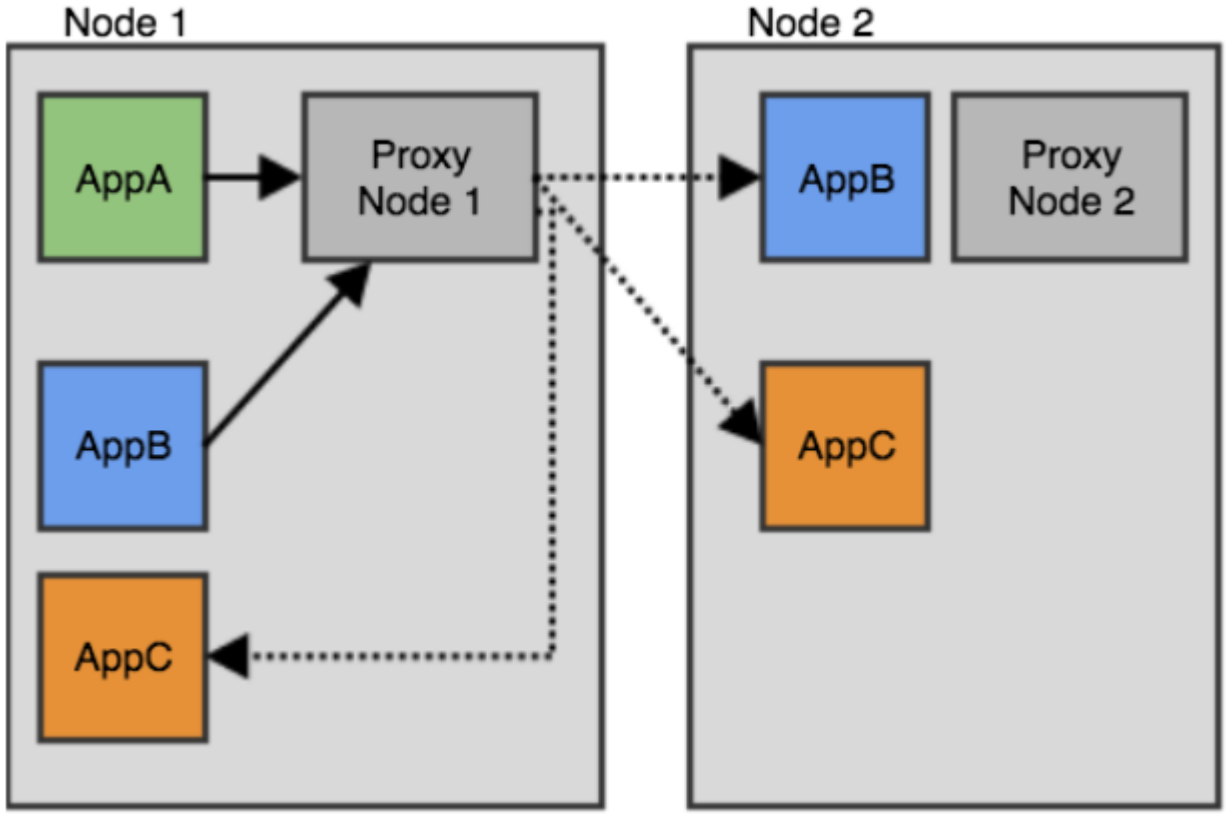
- Library
- Node Agent
- Sidecar

Service Mesh Types - Library



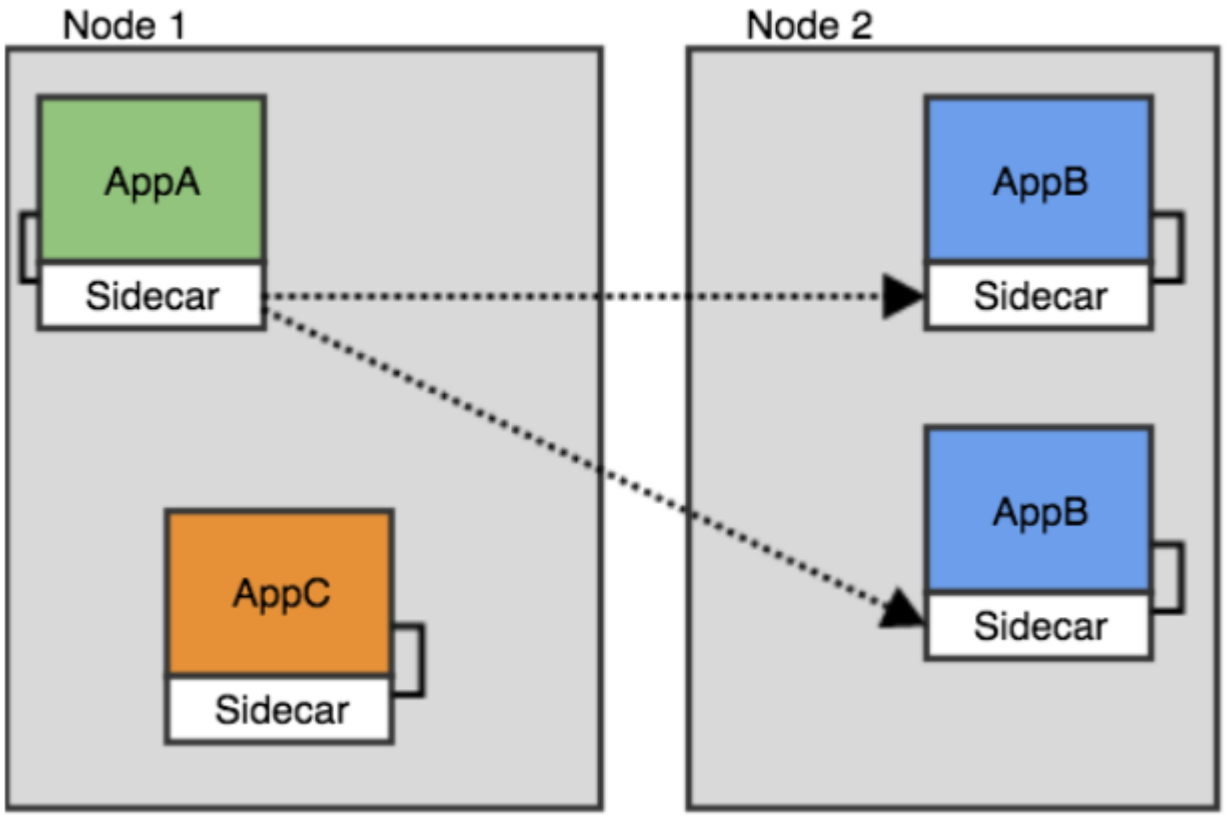
- **Netflix OSS (Hystrix, Ribbon)**, **Twitter Finagle**, **Google Stubby**
- Pros:
  - Simple and straight-forward
  - Work well with one programming language technical stack
  - Don’t care much about co-operate with underlying infrastructure (e.g. Kubernetes, Docker Swarm)
- Cons:
  - Become messy and consume efforts when using many different programming languages (polyglot)
  - Cannot leverage the power of containerized application (Resource Isolation and Dependency Management), orchestration layer (Kubernetes, Docker Swarm)

Service Mesh Types – Node Agent



- **Linkerd** + **Kubernetes**
- Pros:
  - Benefits from sharing is configuration information
  - Containerized Microservices on top of Node Agent or something topologically equivalent
  - Manages the powerful techniques: circuit-breaking, latency-aware load balancing, eventually consistent, service discovery, retries, and deadlines
- Cons:
  - Requires some cooperation from the infrastructure

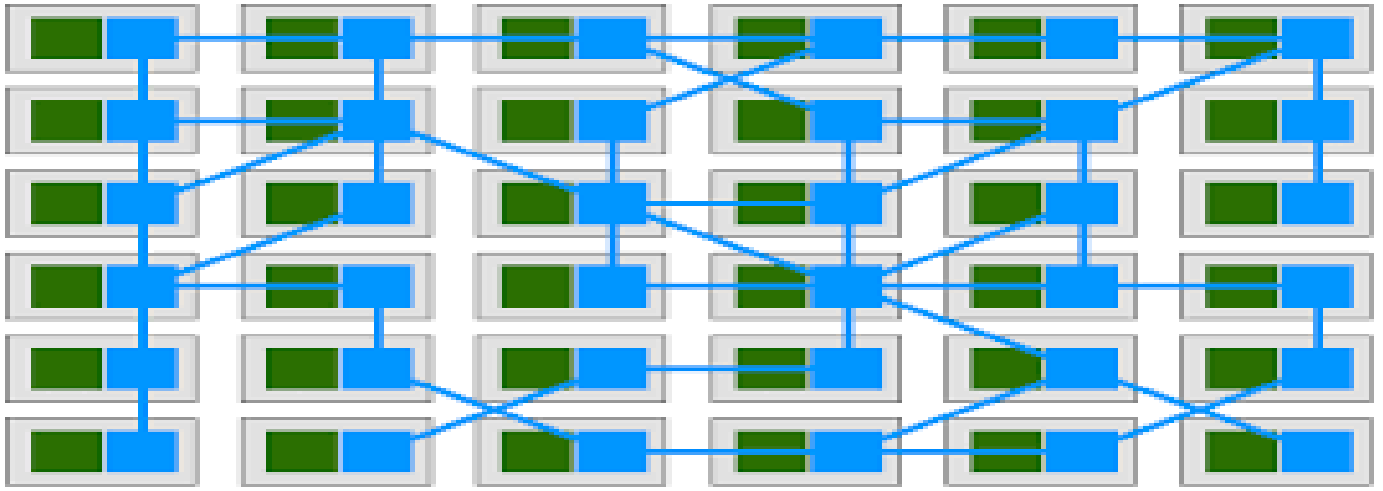
Service Mesh Types – Sidecar



- **Istio + Envoy + Kubernetes, Conduit**
- Pros:
  - Don’t need infrastructure-wide cooperation to deploy that shared agent
  - Security-related aspects - Principle of Least Privilege, e.g. authentication keys, memory and network capabilities
- Cons:
  - Run multiple copies of an identical sidecar
  - Spend more effort computing that reduced configuration for each sidecar.
  - Quite new to the community

Service Mesh - Pros and Cons

- Pros:
  - Commodity features are implemented outside microservice code and be able to reusable.
  - Solves most of the problems in MSA (which we used to have ad-hoc solutions) like distributed tracing, logging, security, access control etc.
  - Freedom when it comes to selecting a microservices implementation language. E.g. PHP, Java, .NET Core, NodeJS, Golang,…
- Cons:
  - Complexity
  - Adding extra hops
  - Immature



Control Plane and Data Plane Competitors

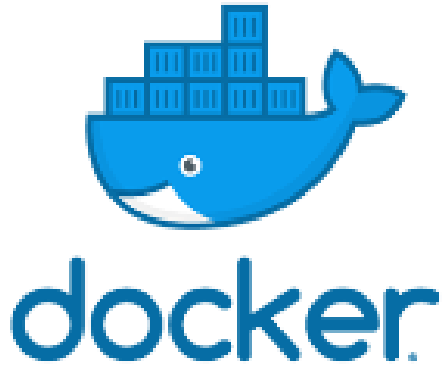
CONTROL PLANE

DATA PLANE

NASH TECH
The Power to Innovate

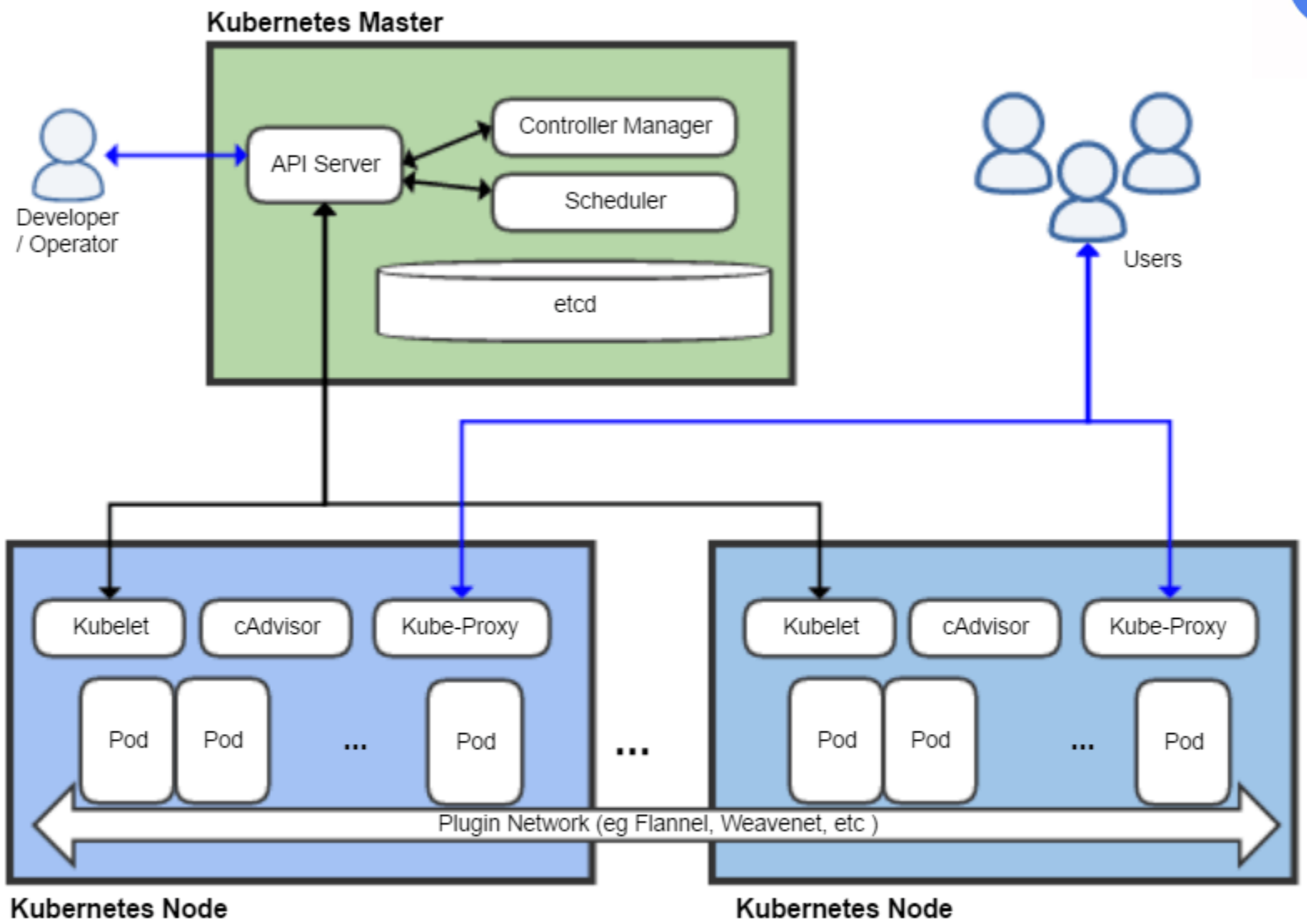
Building Service Mesh for Microservices

Orchestration Layer

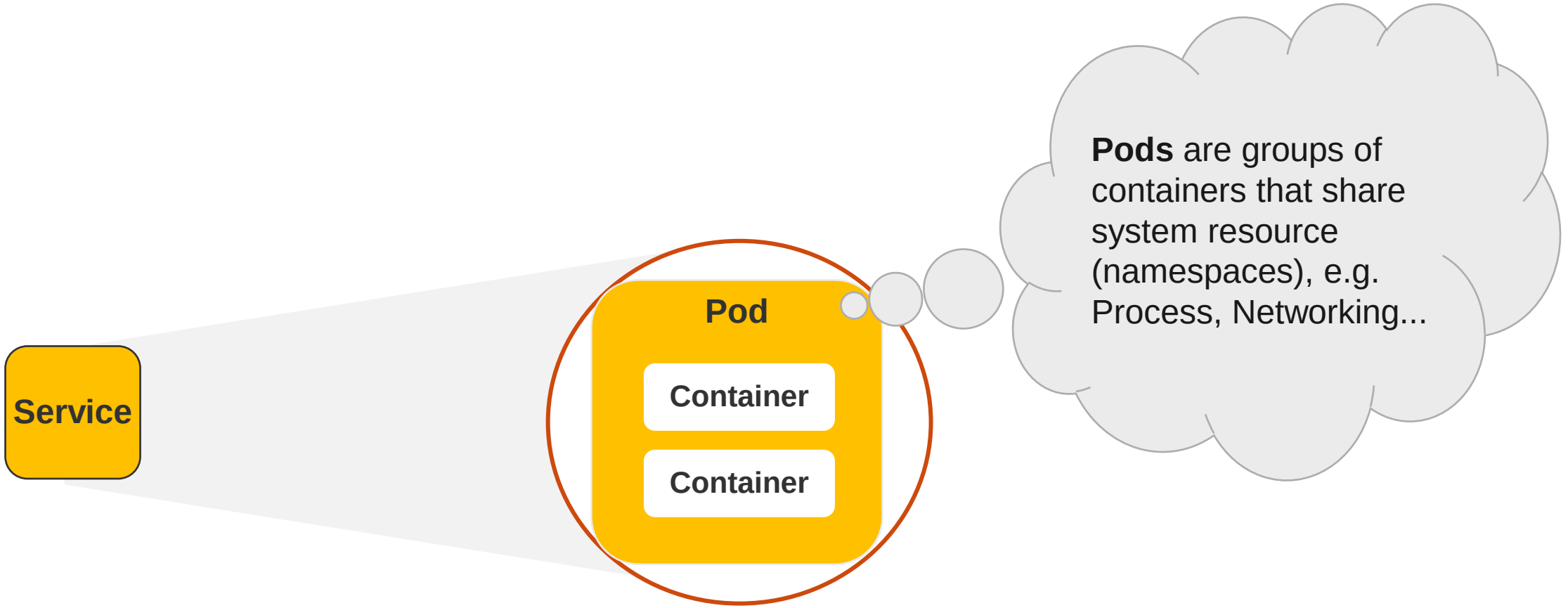


Ambassador

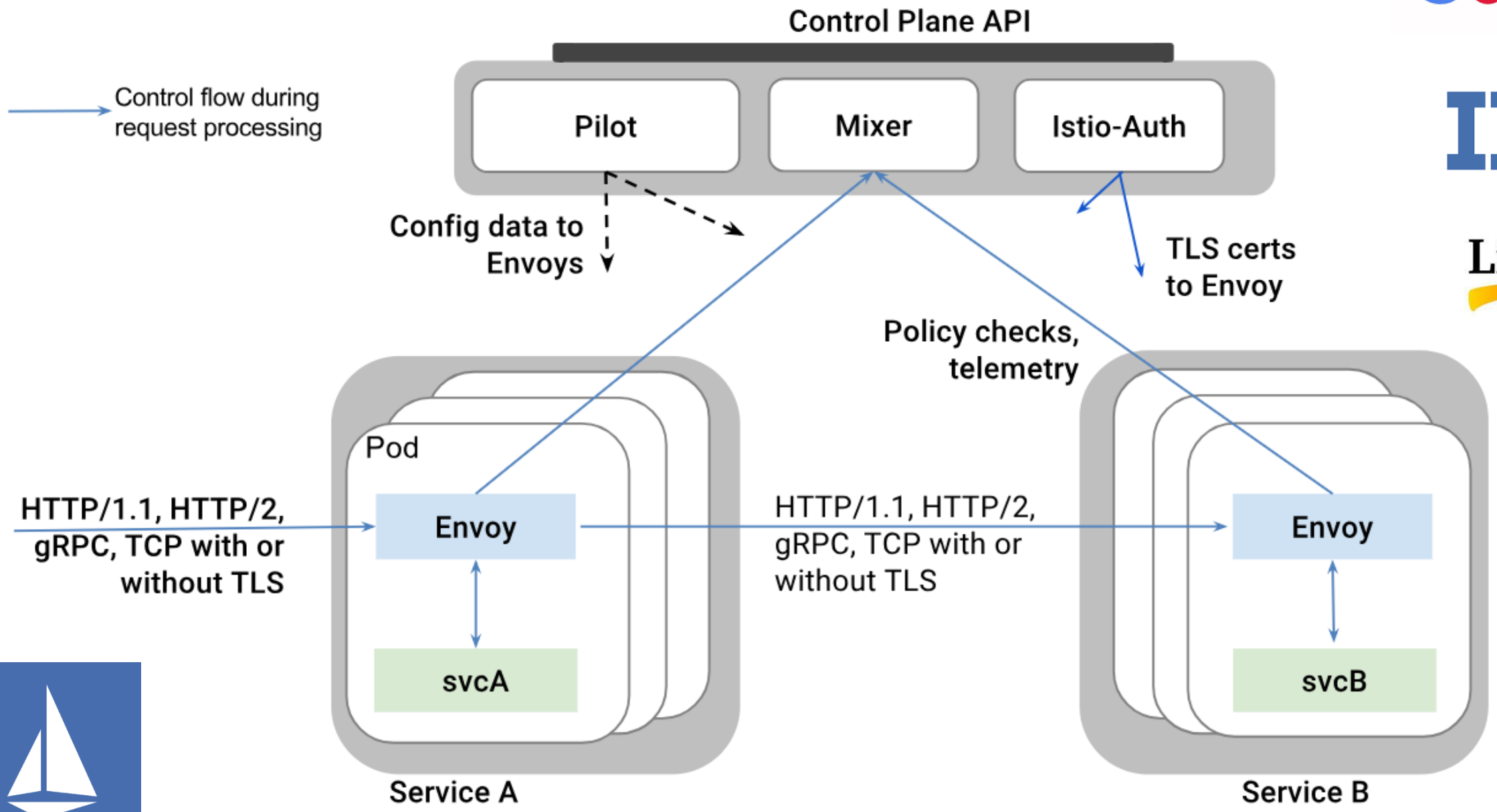
Kubernetes


Kubernetes Master
API Server
Controller Manager
Scheduler
etcd
Developer
/ Operator
Users
Kubelet
cAdvisor
Kube-Proxy
Pod
Pod
...
Pod
...
Kubelet
cAdvisor
Kube-Proxy
Pod
Pod
...
Pod
Plugin Network (eg Flannel, Weavenet, etc)
Kubernetes Node
Kubernetes Node

Kubernetes (cont.)



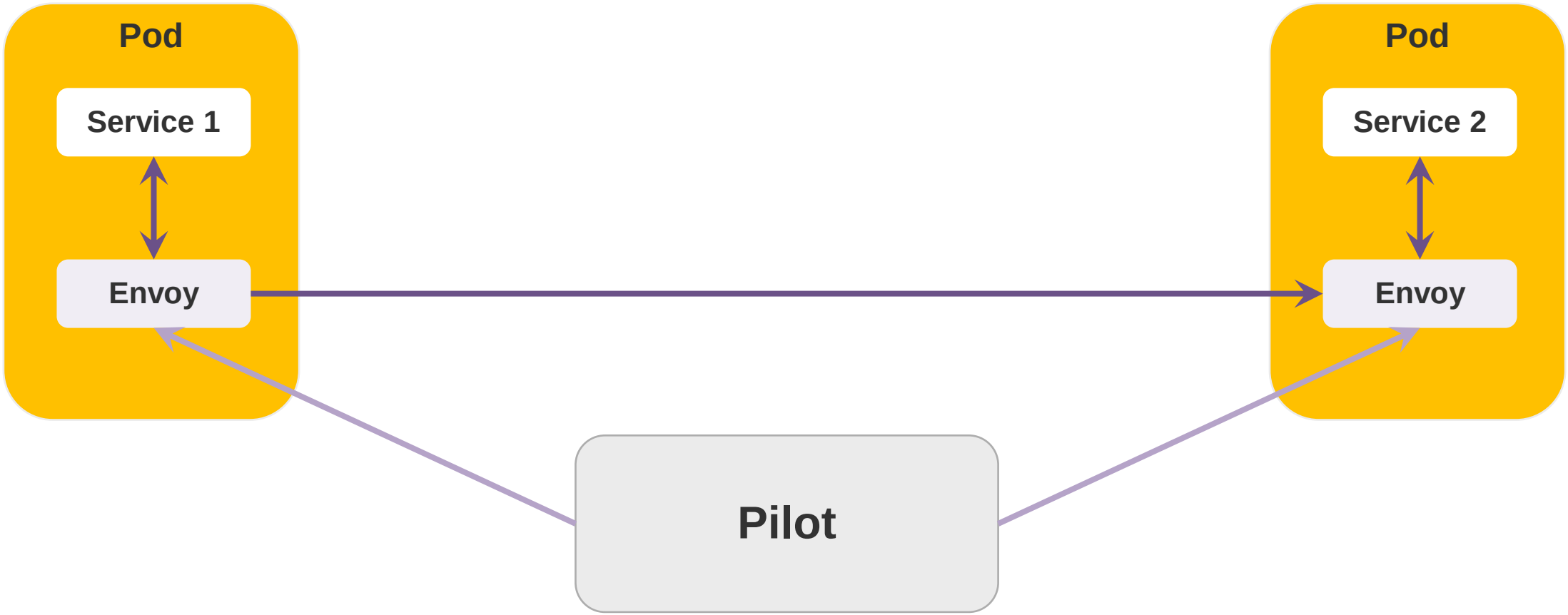
Istio


Control Plane API
Control flow during request processing
Pilot
Mixer
Istio-Auth
Config data to Envoys
TLS certs to Envoy
Policy checks, telemetry
Pod
HTTP/1.1, HTTP/2, gRPC, TCP with or without TLS
Envoy
svcA
HTTP/1.1, HTTP/2, gRPC, TCP with or without TLS
Envoy
svcB
Service A
Service B

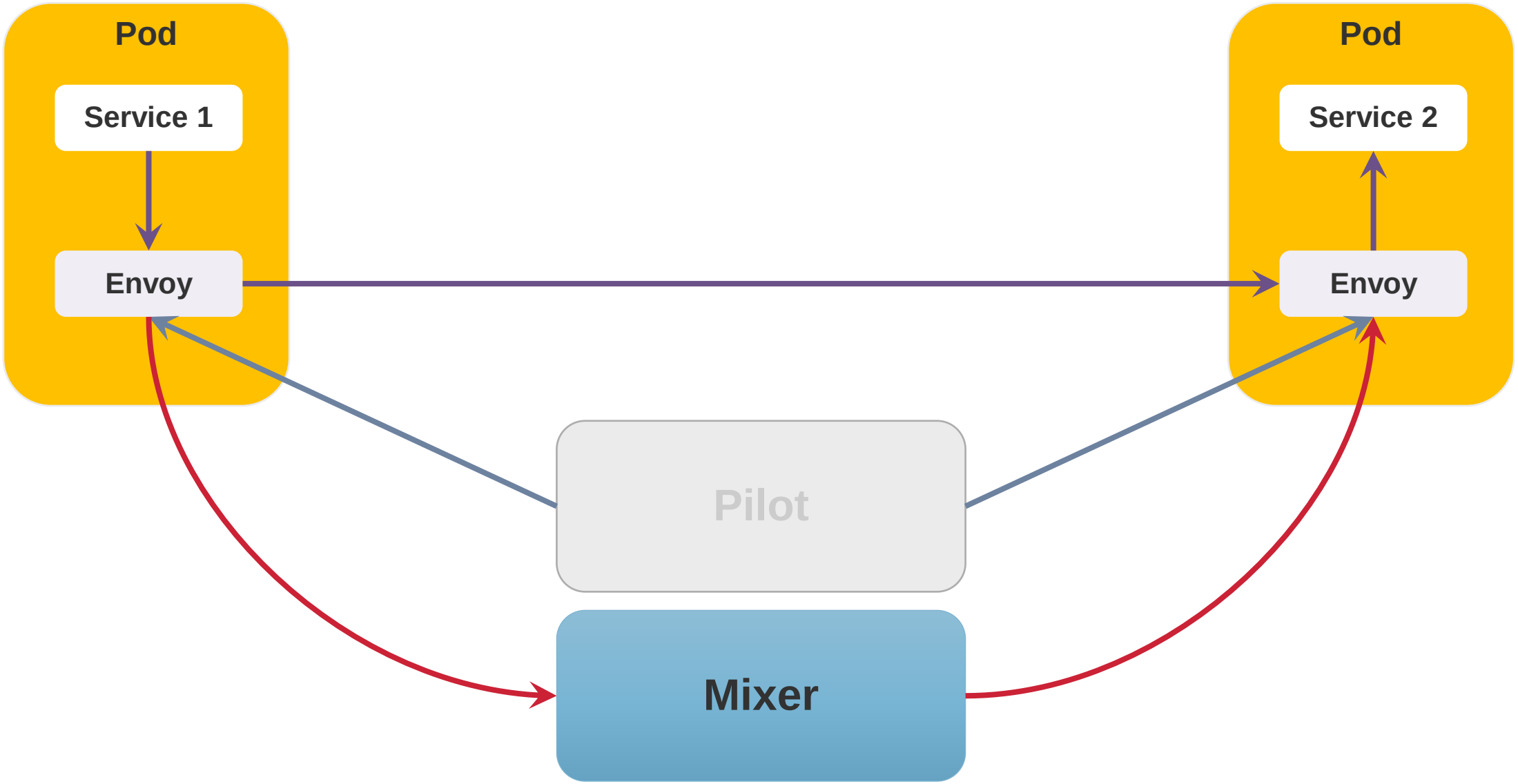
Istio - Sidecar

- **A secondary container in a Pod**
- **Intercept & manages network traffic**
- **Security/Identity**
- **Pluggability**
- **Language Agnostic**

Istio - Pilot



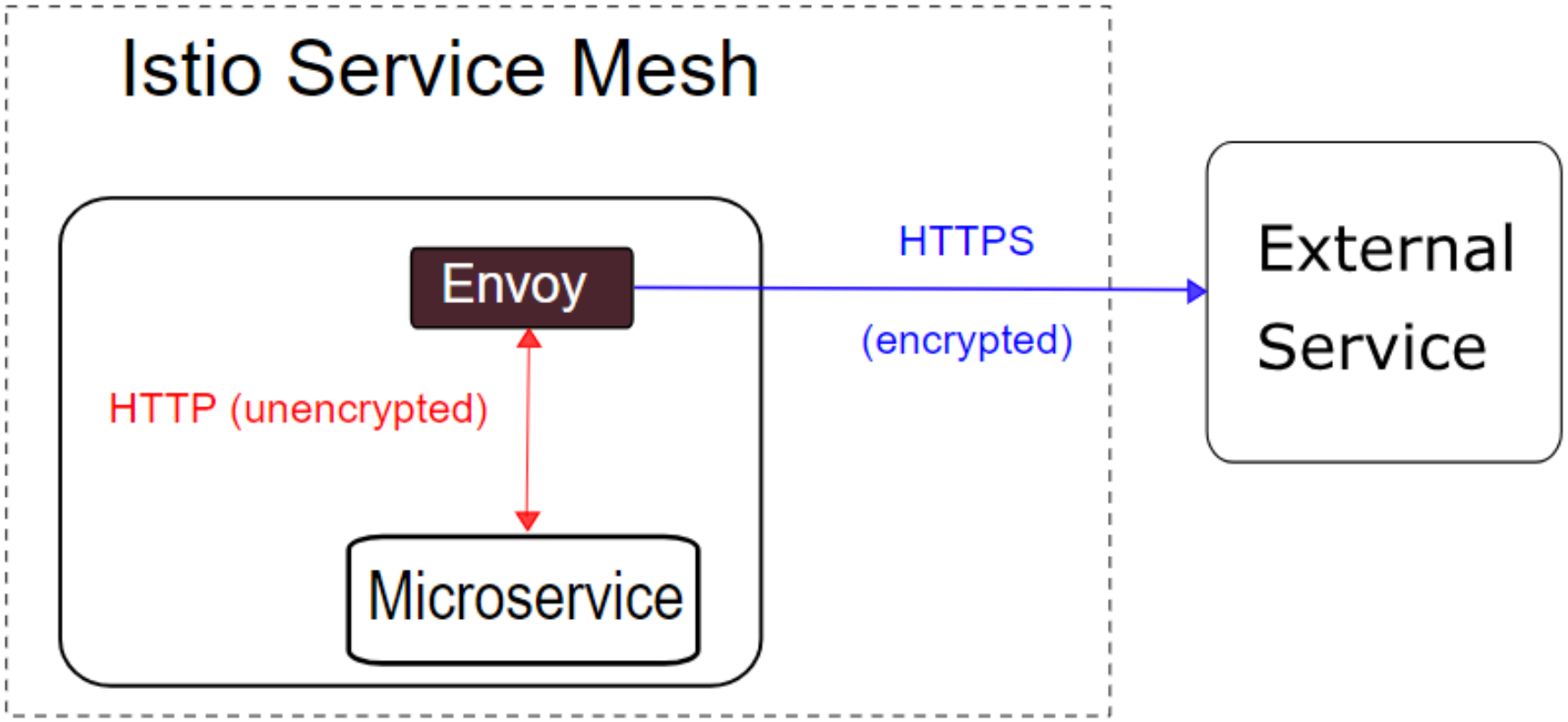
Istio - Mixer



Istio - Auth


Service 1
Service 2
mTLS & Secure Naming
Issue &
Mount Keys
Istio Certificate
Authority
Issue &
Mount Keys

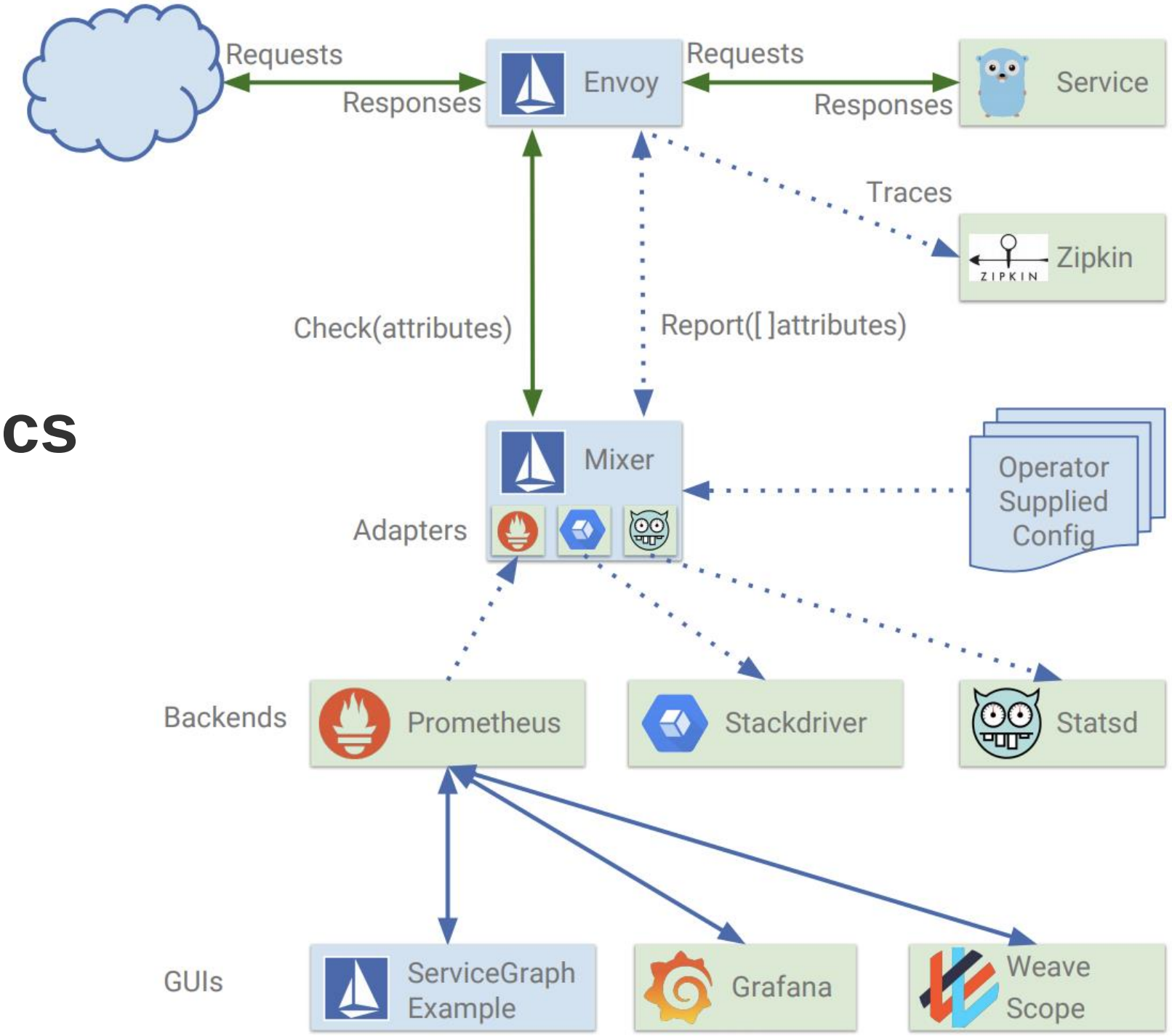
Istio – Auth (cont.)



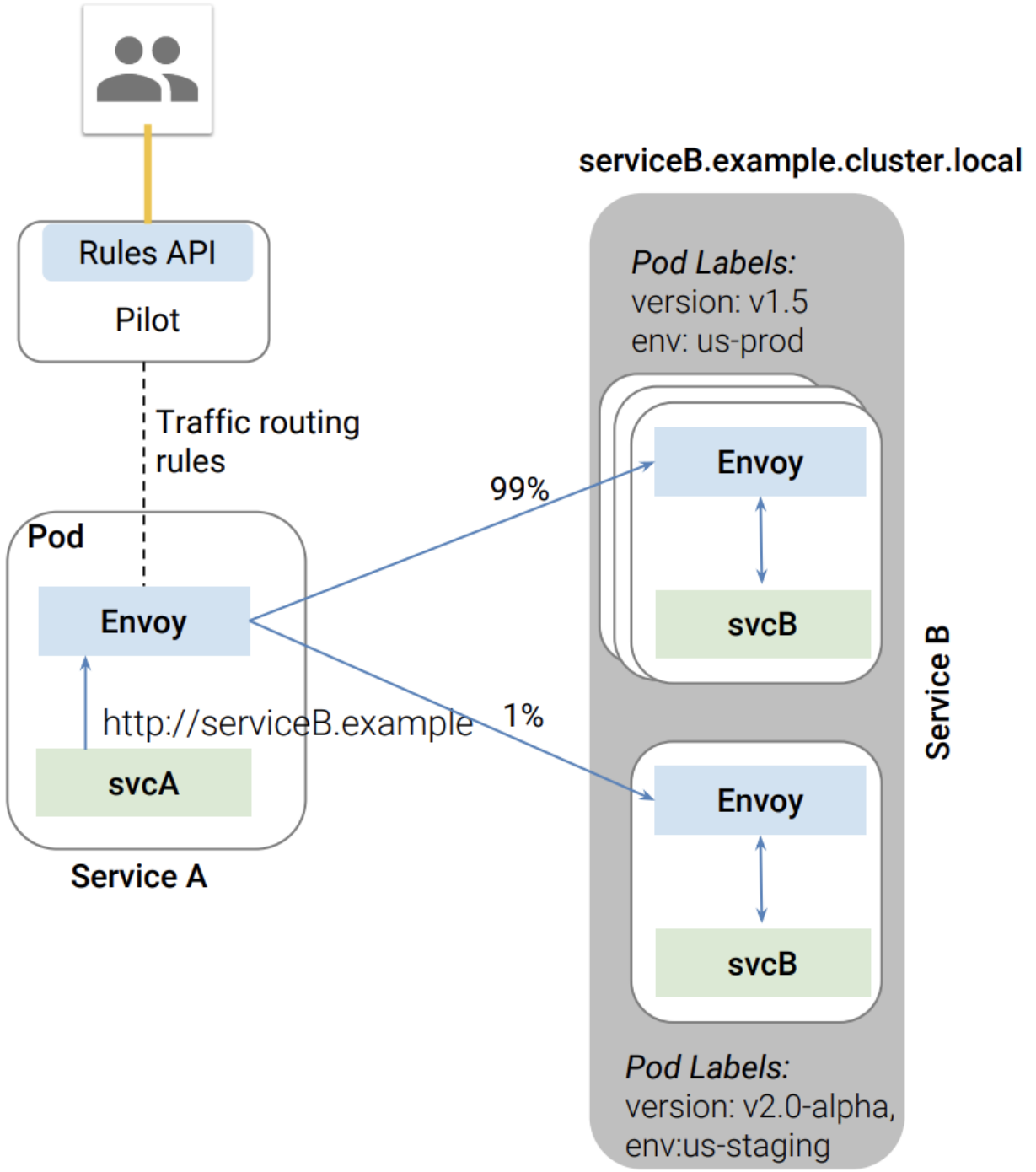
Istio – Key features

- **Automatic Protocol Metrics Collection & Tracing**
- **Mutual TLS Authentication**
- **Circuit Breaking**
- **Failure Injection**
- **Traffic Splitting**

Automatic Protocol Metrics Collection & Tracing

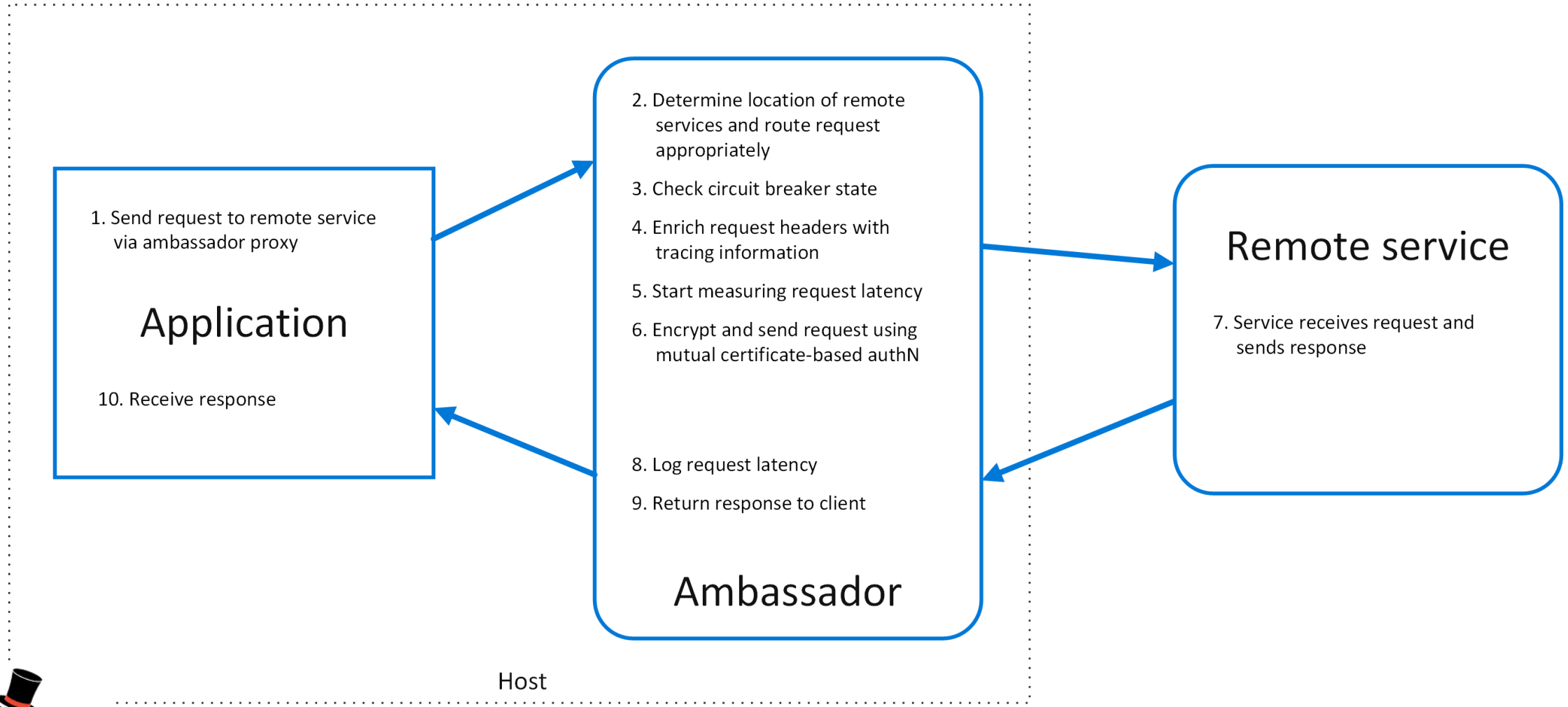


- **Circuit Breaking**
- **Failure Injection**
- **Traffic Splitting**



Ambassador

DATAWIRE



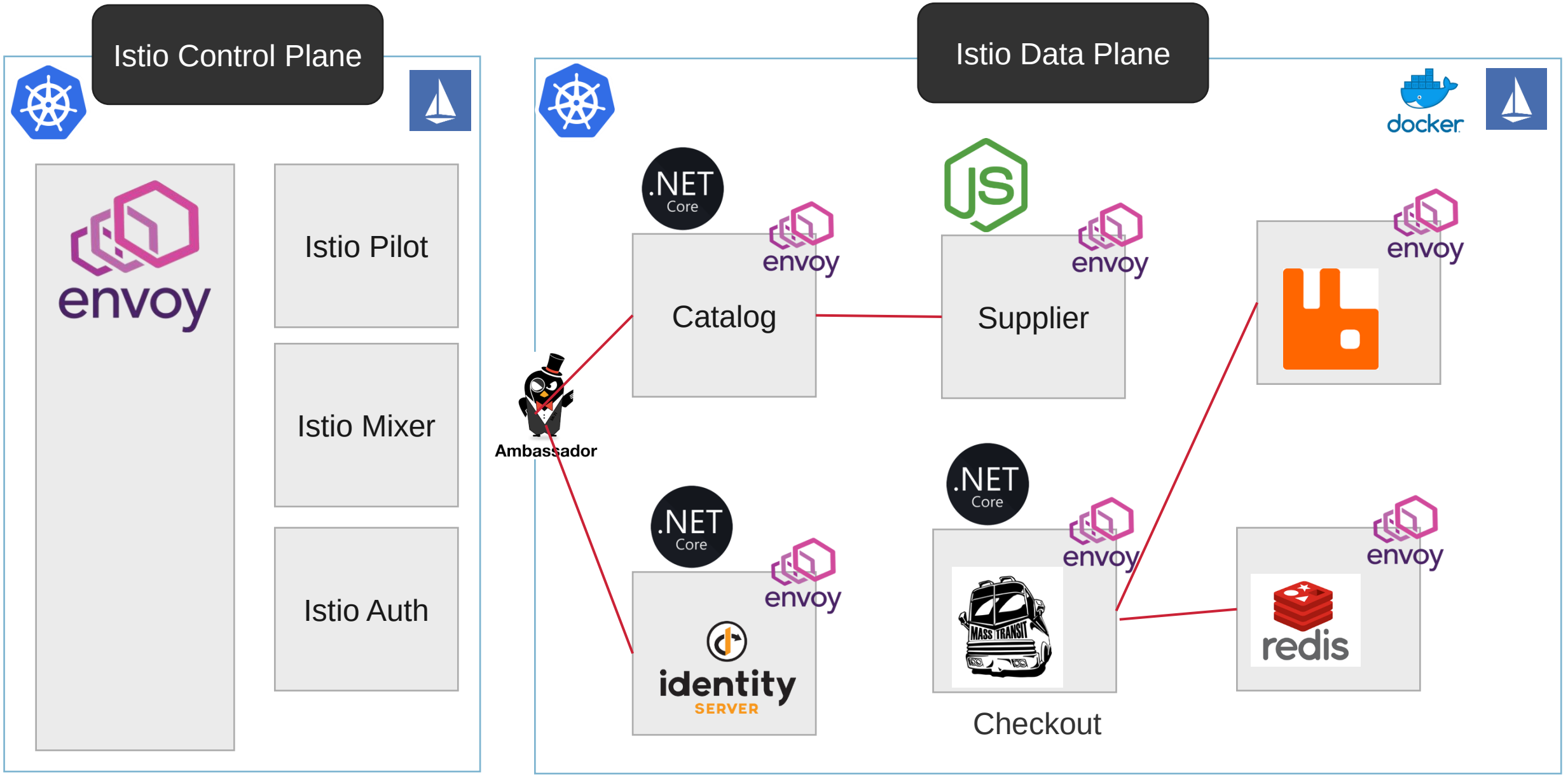
Ambassador

NASH TECH
The Power to Innovate

Demo Time

https://github.com/thangchung/shopping-cart-k8s

Shopping Cart Demo



NASH TECH
The Power to Innovate

Q&A

THANK YOU

www.nashtechglobal.com