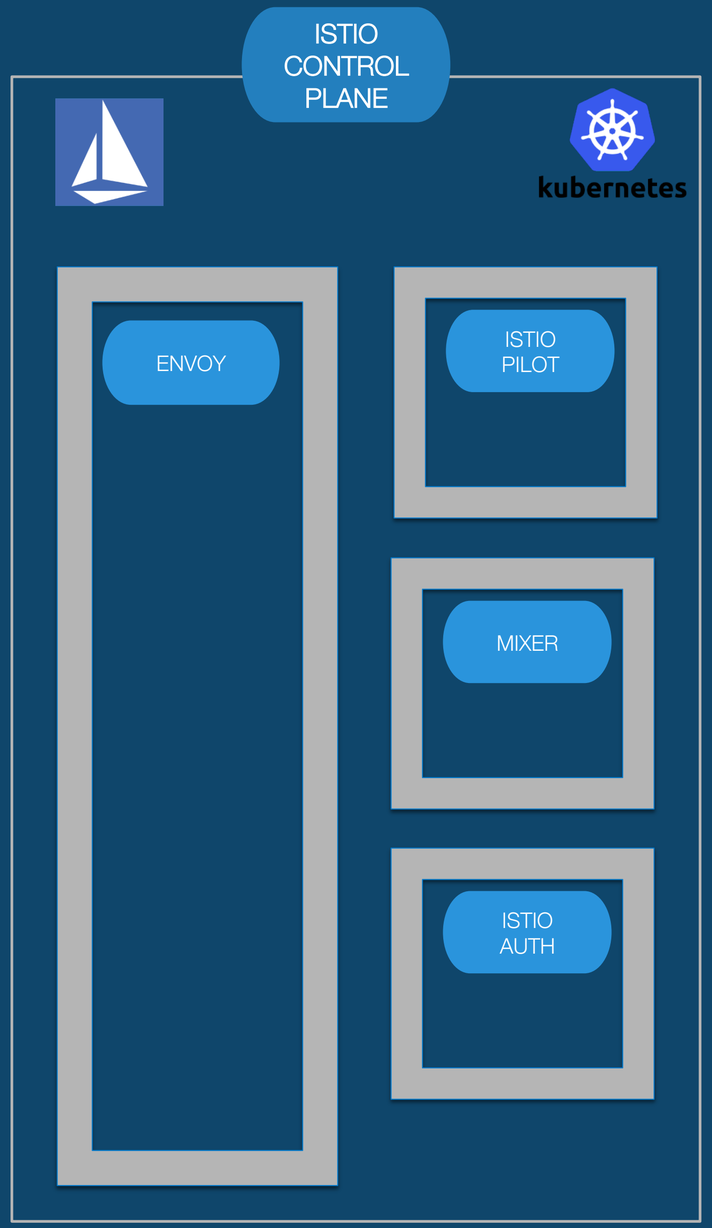
Microservices, Kubernetes & Istio - A great fit!

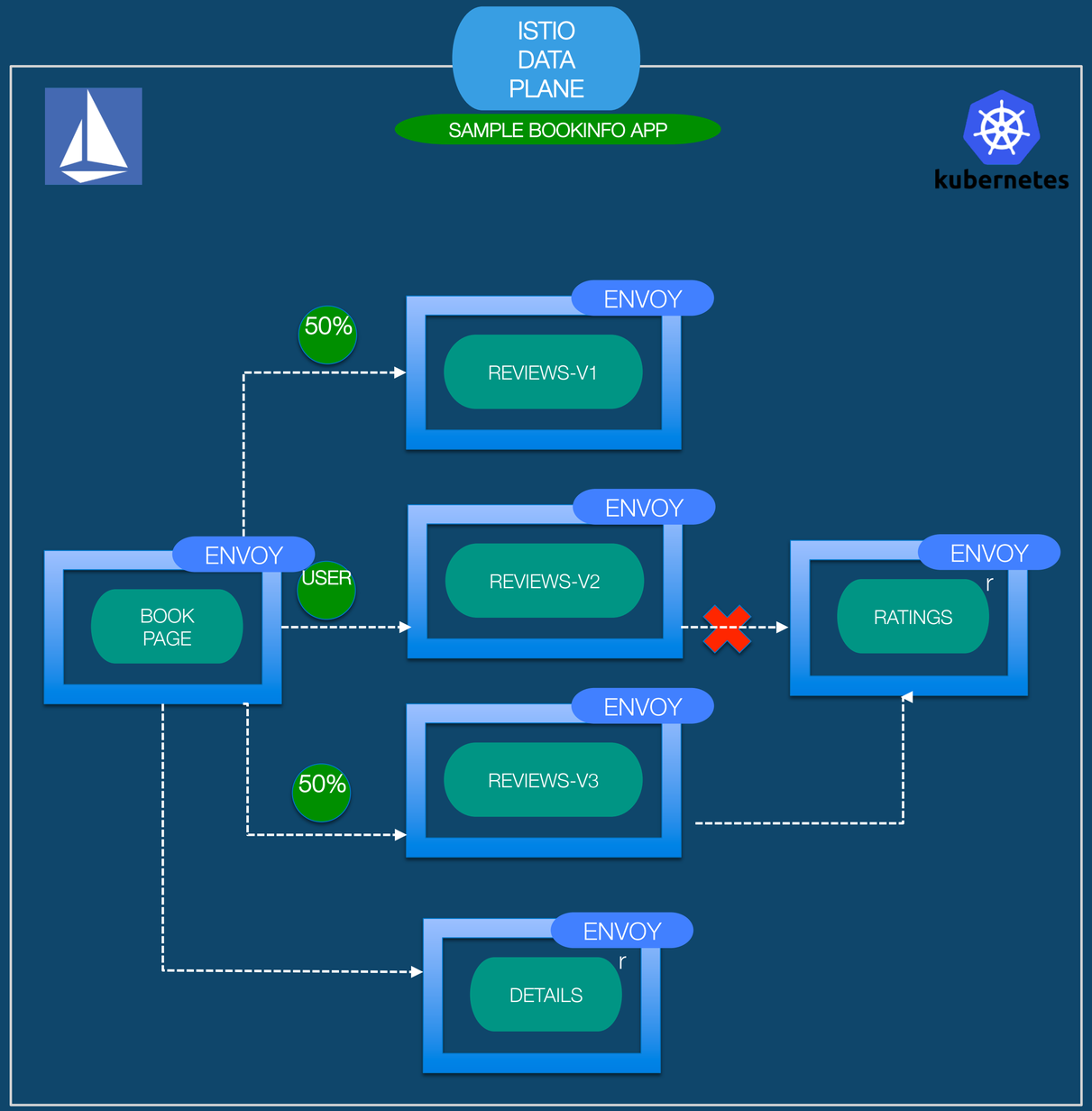
Presenters:

Animesh Singh
Anthony Amanse
Tommy Li

Panelists:

Chris Rosen
Nilesh Patel





Microservices, Kubernetes & Istio - A great fit!

Presenters:

Animesh Singh
Anthony Amanse
Tommy Li

Panelists:

Chris Rosen
Nilesh Patel

Slides : http://ibm.biz/kube-istio

Developer Journeys: http://ibm.biz/kube-journeys

Reach out to us at:

@AnimeshSingh

@AnthonyAmanse

@Tomipli

@ChrisRosen188

@nileshandweb

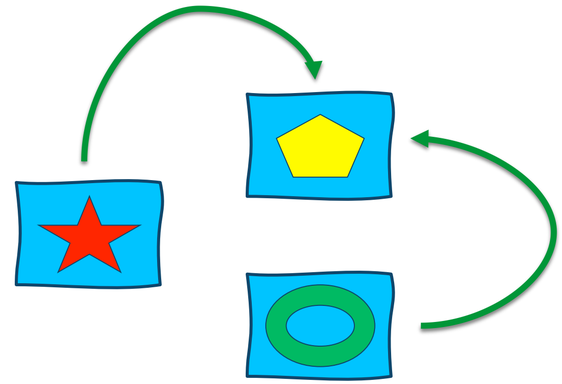
Evolution of Microservices

Microservices

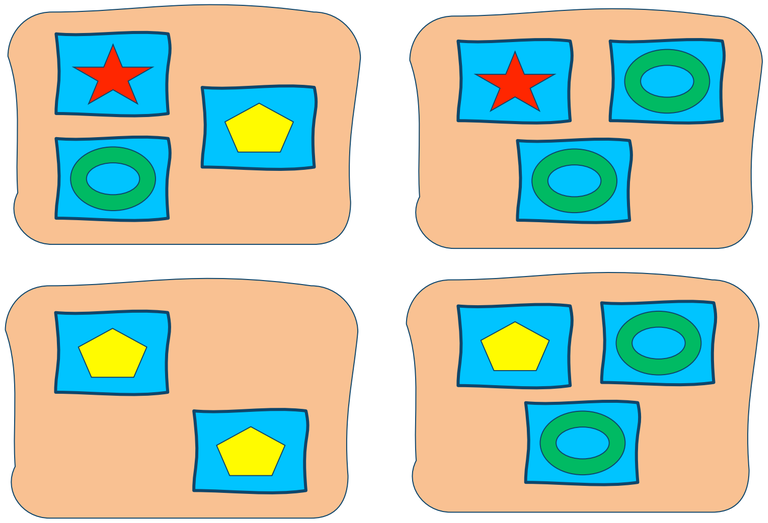
An engineering approach focused on decomposing an application into single-function modules with well defined interfaces which are independently deployed and operated by a small team who owns the entire lifecycle of the service.

Microservices accelerate delivery by minimizing communication and coordination between people while reducing the scope and risk of change.

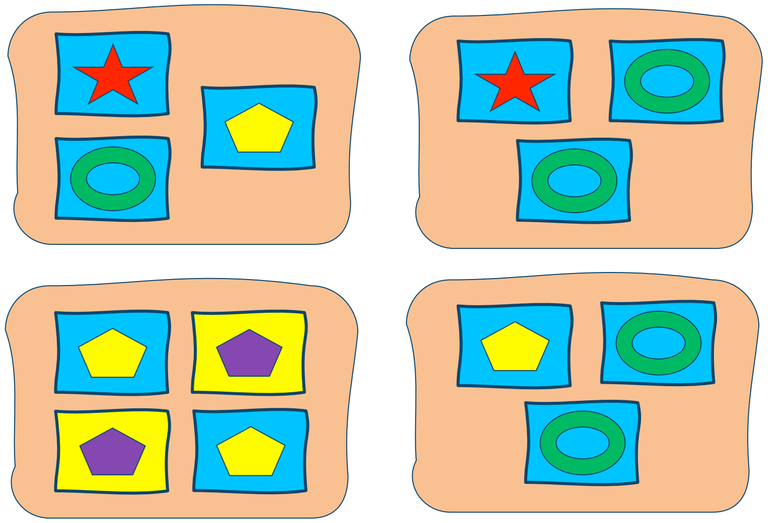
Microservices Application Interactions



Microservices Application Scaling



Microservices Application Update



Microservices, Containers and Container Orchestrator

Typically microservices are encapsulated inside containers…

One:One relationship between a microservice and a container

Everyone’s container journey starts with one container….

At first the growth is easy to handle….

But soon it is overwhelming…we need container and microservices management

Enter Container Orchestrator




Container Orchestration =

Scheduling, Cluster Mgmt,

and Discovery

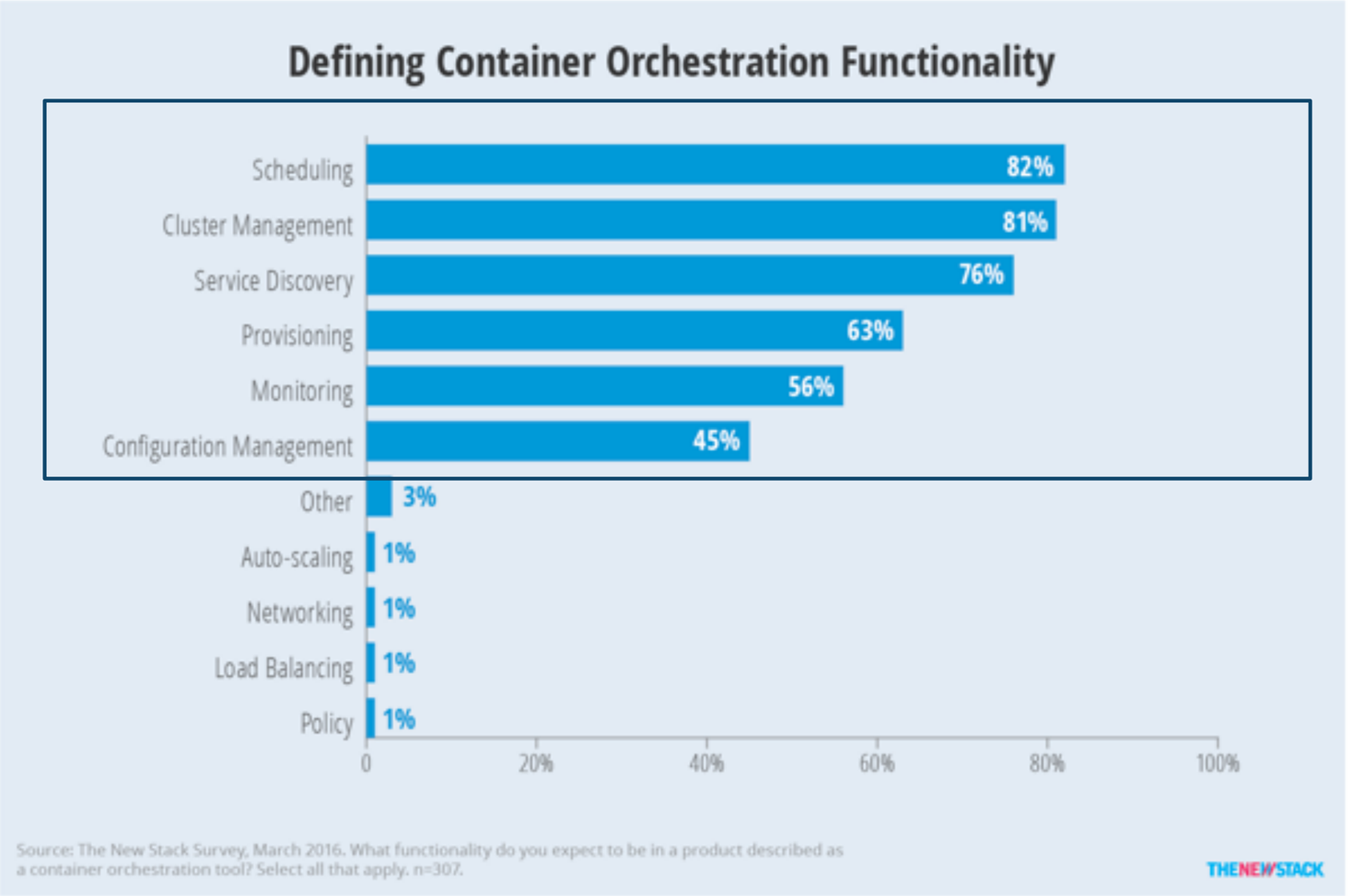
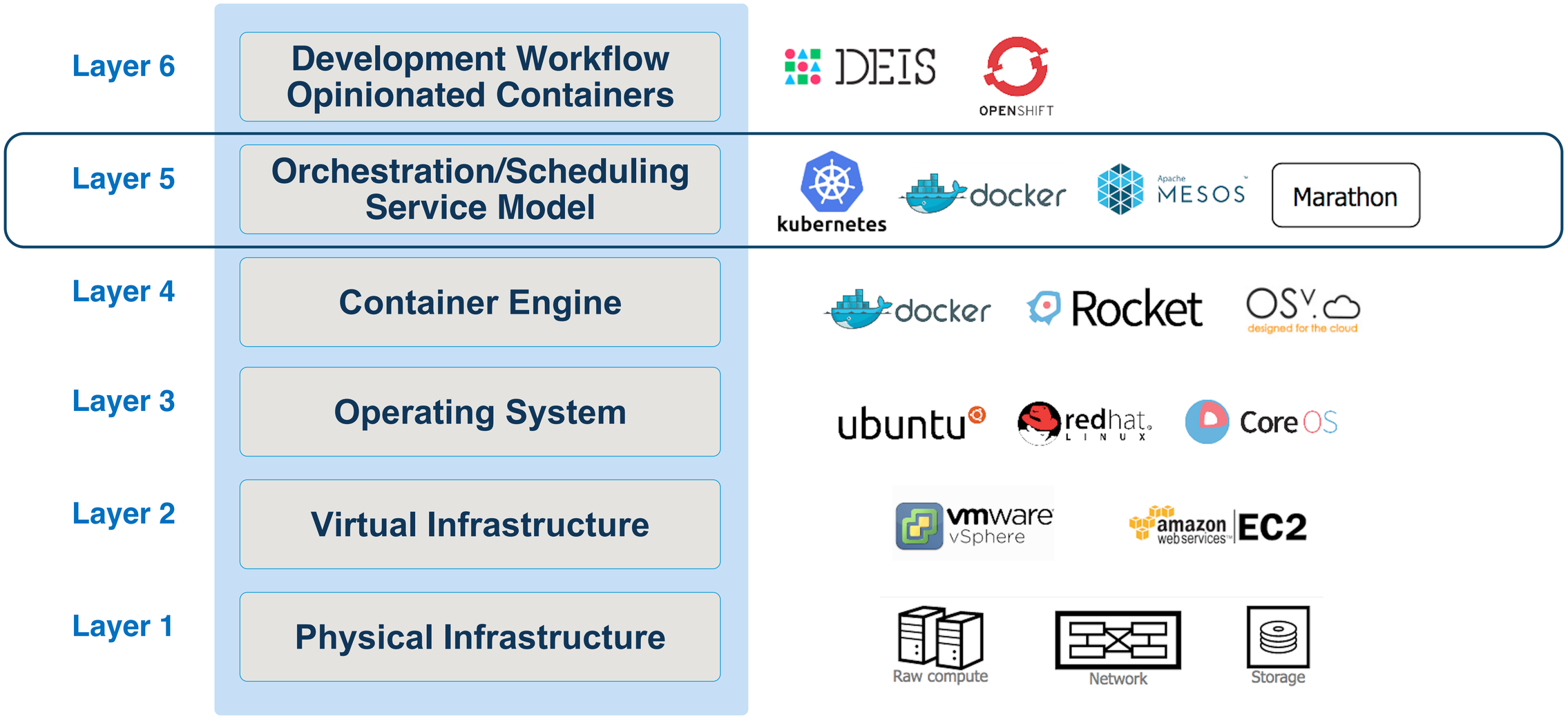
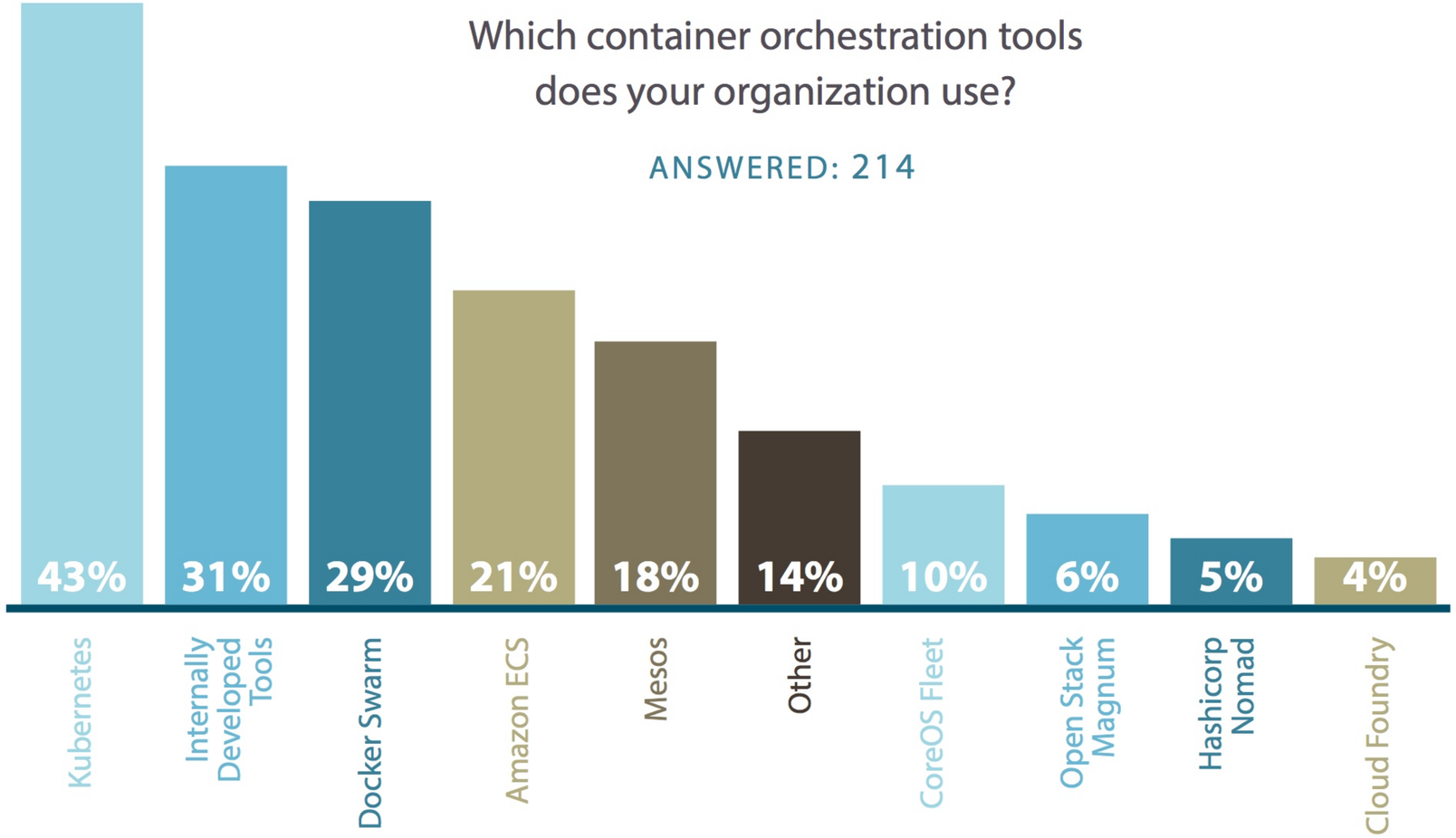


Figure 3: Only 45 percent of respondents consider configuration management to be part of a container orchestration product.

Container Stack



Container Orchestration



Source: devops.com

Kubernetes

What is Kubernetes?

- Container orchestrator
- Runs and manages containers
- Supports multiple cloud and bare-metal environments
- Inspired and informed by Google's experiences and internal systems
- 100% Open source, written in Go
- Manage applications, not machines
- Rich ecosystem of plug-ins for scheduling, storage, networking

IBM Bluemix Container Service

Intelligent Scheduling

Self-healing

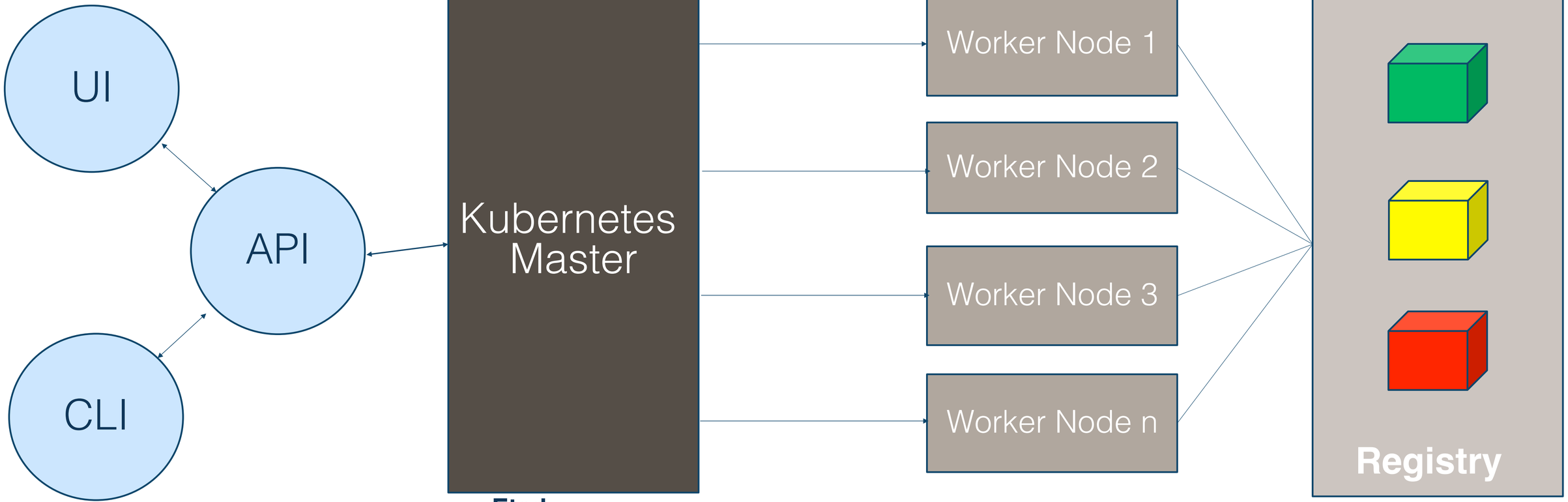
Horizontal scaling

Service discovery & load balancing

Automated rollouts and rollbacks

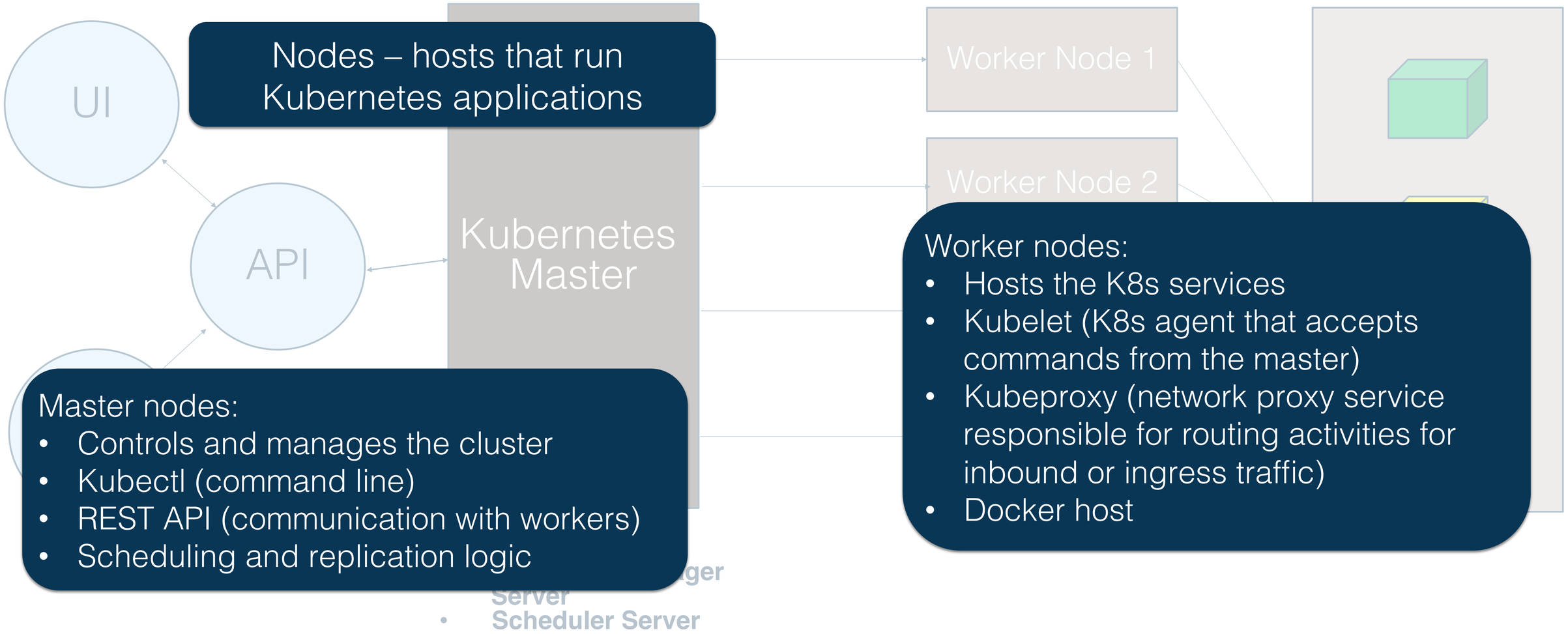
Secret and configuration management

Kubernetes Architecture



- **Etcd**
- **API Server**
- **Controller Manager Server**
- **Scheduler Server**

Kubernetes Architecture



Kubernetes Architecture

UI

Pods:
- Smallest deployment unit in K8s
- Collection of containers that run on a worker node
- Each has its own IP
- Pod shares a PID namespace, network, and hostname

Labels:
- Metadata assigned to K8s resources
- Key-value pairs for identification
- Critical to K8s as it relies on querying the cluster for resources that have certain labels

Worker Node 3

Replication controller:
- Ensures availability and scalability
- Maintains the number of pods as requested by user
- Uses a template that describes specifically what each pod should contain

Service:
- Collections of pods exposed as an endpoint
- Information stored in the K8s cluster state and networking info propagated to all worker nodes

IBM Bluemix Container Service

- Single tenant k8s master
- Running in IBM managed account
- Key managed to maintain account separation

API

CLI

Kubernetes Master

- **Etcd**
- **API Server**
- **Controller Manager Server**
- **Scheduler Server**

Worker Node 1

Worker Node 2

Worker Node 3

- Single tenant worker nodes
- Dedicated nodes are single tenant top to bottom (worker nodes, hypervisor, hardware)
- No core or RAM over subscription to ensure allocated resources and ensure no noisy neighbors

(https://www.ibm.com/cloud-computing/bluemix/virtual-servers)

Registry

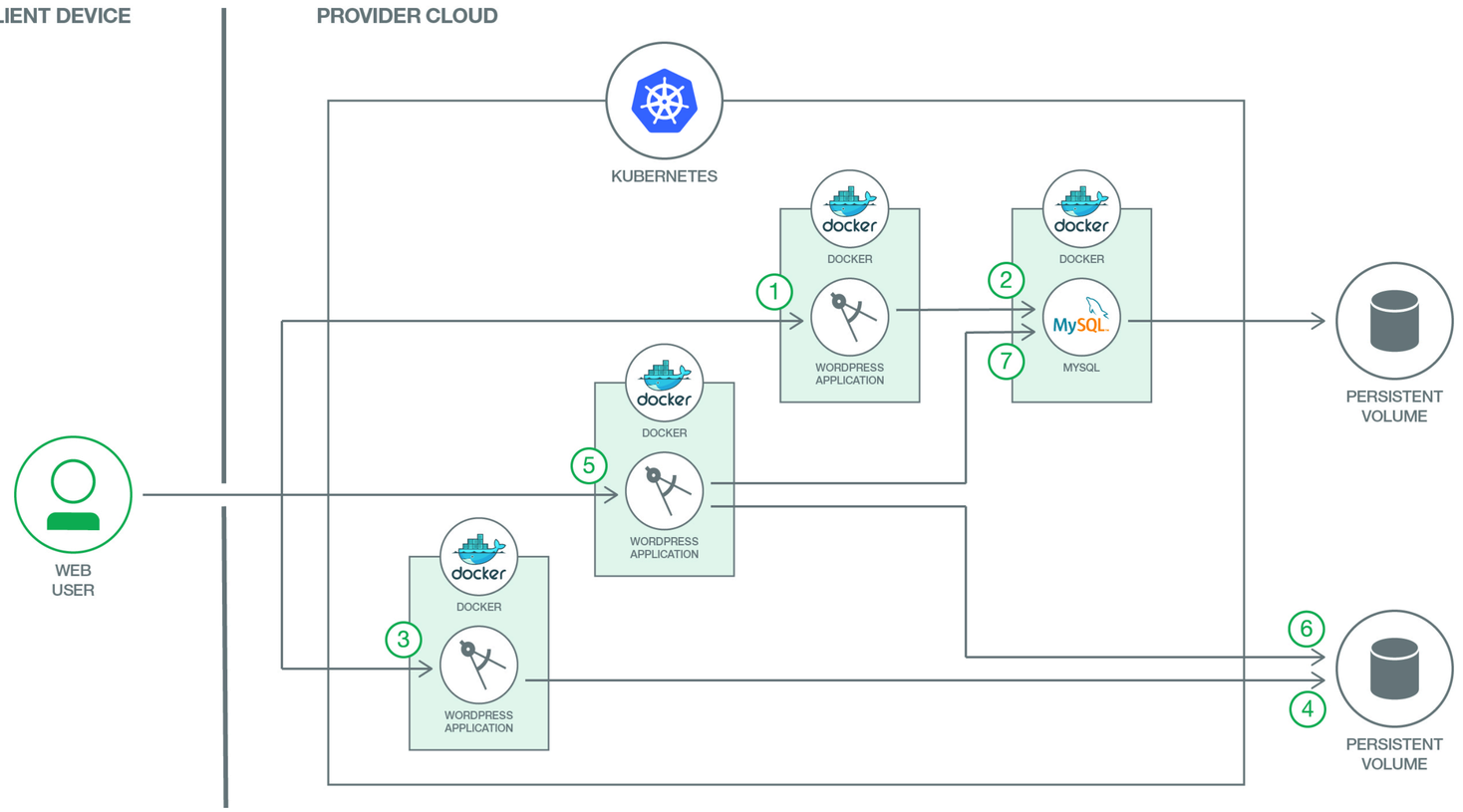
Multi-tenant services with per tenant isolation

Kubernetes Developer Journeys

Developer Journeys:
Scalable Wordpress on Kubernetes

In addition to running MySQL inside a container, we also show advanced capabilities like Bluemix service binding by leveraging Compose for MySQL service

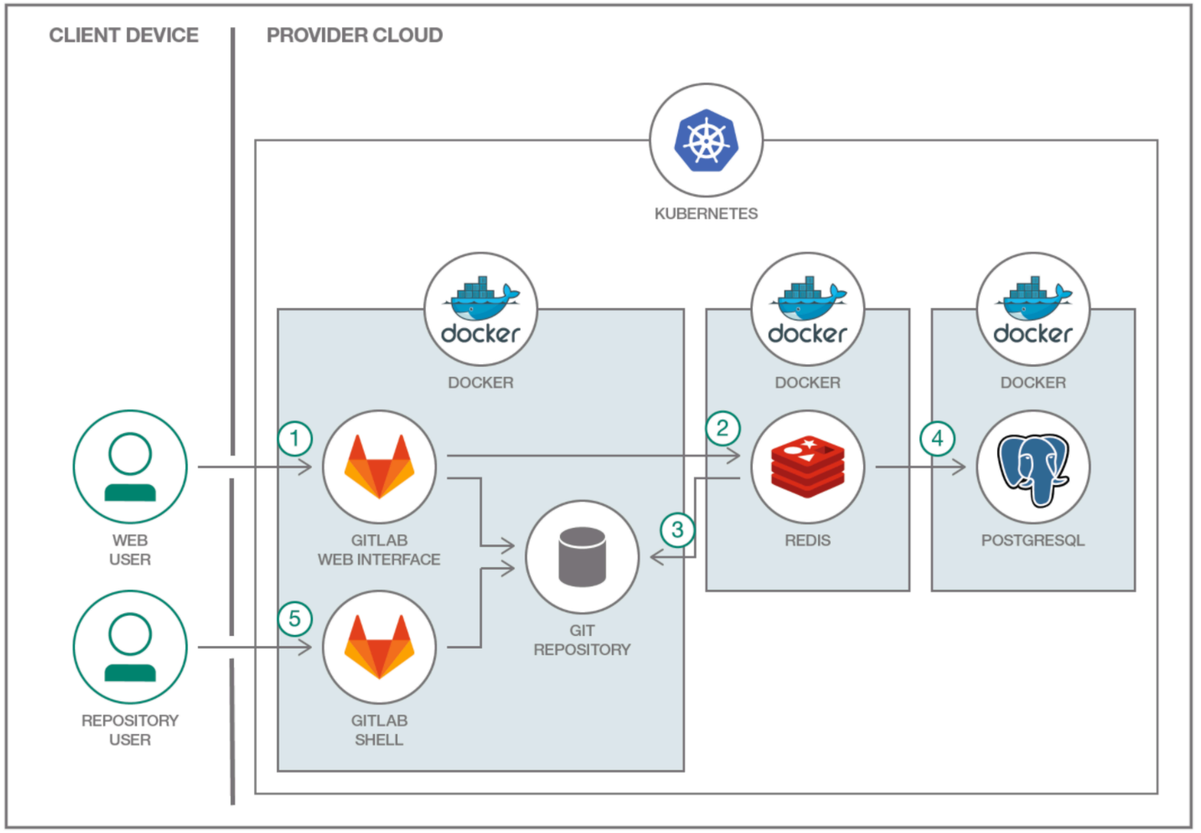
Developer Works Code: https://developer.ibm.com/code/journey/scalable-wordpress-on-kubernetes
Github: https://github.com/IBM/scalable-wordpress-deployment-on-kubernetes



Developer Journeys:
Deploy a Distributed GitLab on Kubernetes

This project shows how a common multi-component application can be deployed. GitLab represents a typical multi-tier app and each component will have their own container(s).

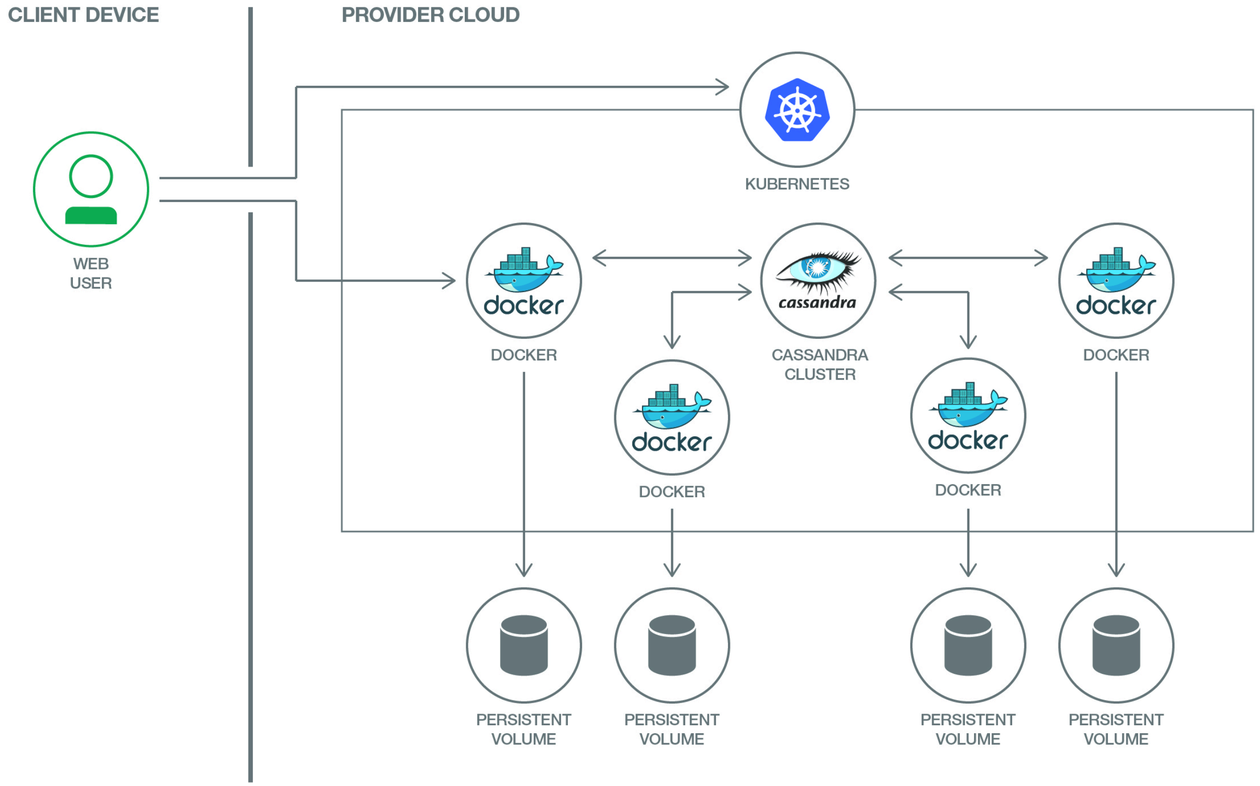
Developer Works Code: https://developer.ibm.com/code/journey/run-gitlab-kubernetes/
Github: https://github.com/IBM/kubernetes-container-service-gitlab-sample



Developer Journeys:
Scalable Apache Cassandra on Kubernetes

Leverages Kubernetes Pods, Service, Replication Controller, StatefulSets

Developer Works Code: https://developer.ibm.com/code/journey/deploy-a-scalable-apache-cassandra-database-on-kubernetes
Github: https://github.com/IBM/scalable-cassandra-deployment-on-kubernetes



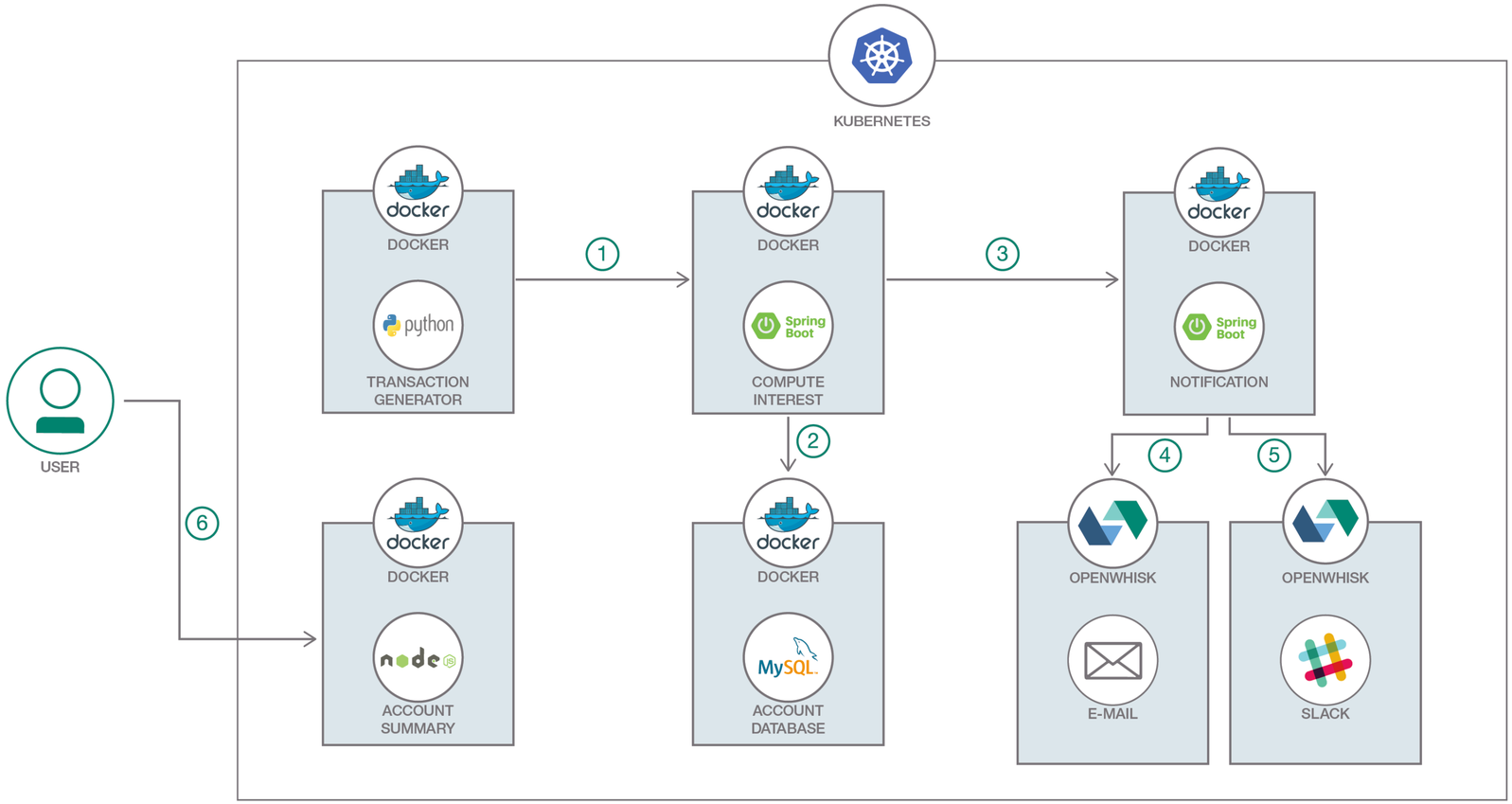
Kubernetes & Microservices

Developer Journeys

Developer Journeys:
Spring Boot Microservices on Kubernetes

This journey shows you how to create and deploy Spring Boot microservices within a polyglot application and then deploy the app to a Kubernetes cluster.

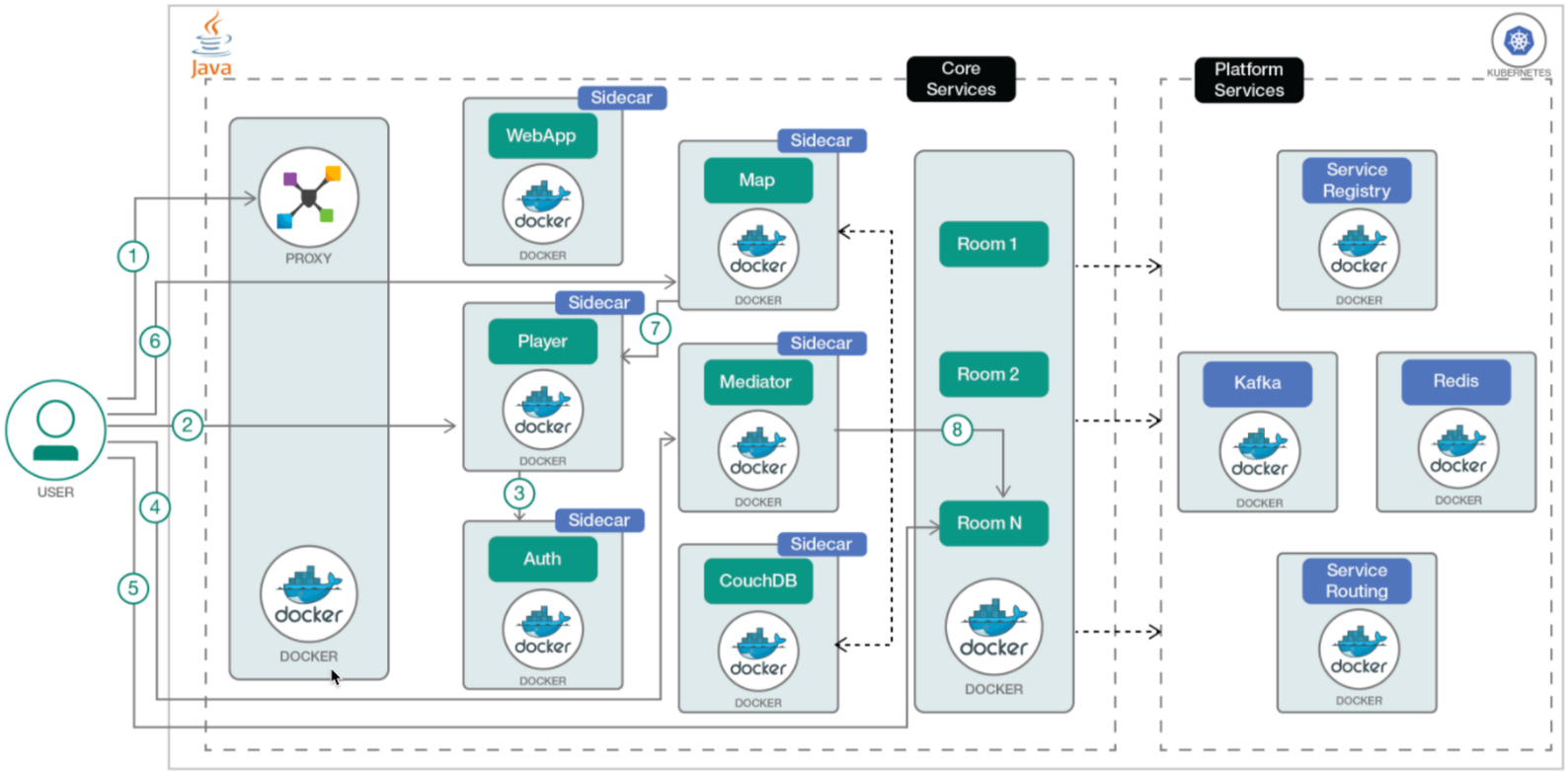
Developer Works Code: https://developer.ibm.com/code/journey/deploy-spring-boot-microservices-on-kubernetes/
Github: https://github.com/IBM/spring-boot-microservices-on-kubernetes



Developer Journeys:
Deploy Java microservices on Kubernetes within a polyglot ecosystem

With current application architectures, microservices need to co-exist in polyglot environments. In this developer journey, you'll learn how to deploy a Java microservices application that runs alongside other polyglot microservices, leveraging service discovery, registration, and routing.

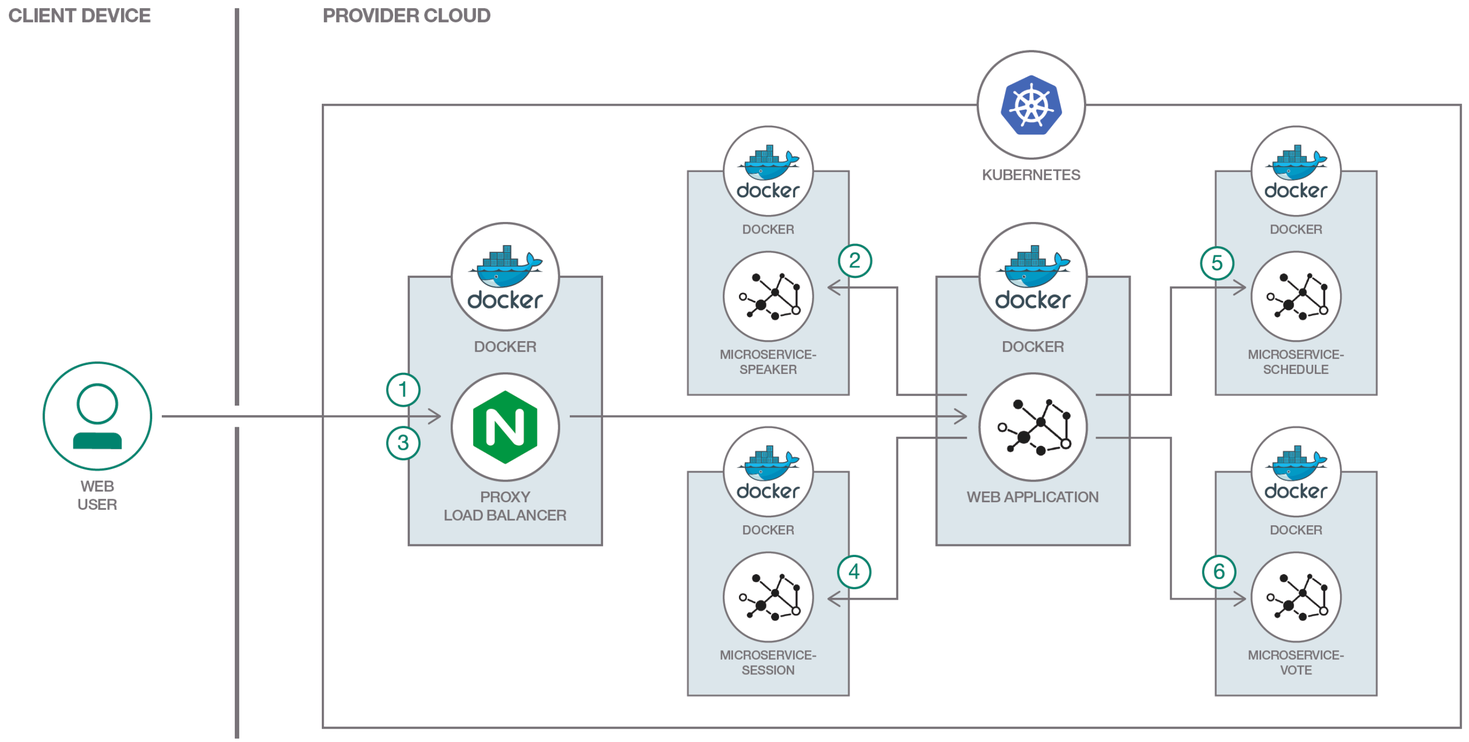
Developer Works Code: https://developer.ibm.com/code/journeys/deploy-java-microservices-on-kubernetes-with-polyglot-support/
Github: https://github.com/IBM/GameOn-Java-Microservices-on-Kubernetes



Developer Journeys:
Java MicroProfile Microservices on Kubernetes

Java based Microservices application using MicroProfile (baseline for Java Microservices architecture) and Microservices Builder on Kubernetes

Developer Works Code: https://developer.ibm.com/code/journey/deploy-microprofile-java-microservices-on-kubernetes
Github: https://github.com/IBM/java-microprofile-on-kubernetes



Developer Journeys:
Java MicroProfile Microservices on Kubernetes

Java based Microservices application using MicroProfile (baseline for Java Microservices architecture) and Microservices Builder on Kubernetes

Developer Works Code: https://developer.ibm.com/code/journey/deploy-microprofile-java-microservices-on-kubernetes
Github: https://github.com/IBM/java-microprofile-on-kubernetes

DEMO

Kubernetes is great for Microservices…

Why do we need a Service mesh and what is it?

Container Orchestration =

Scheduling, Cluster Mgmt,

and Discovery

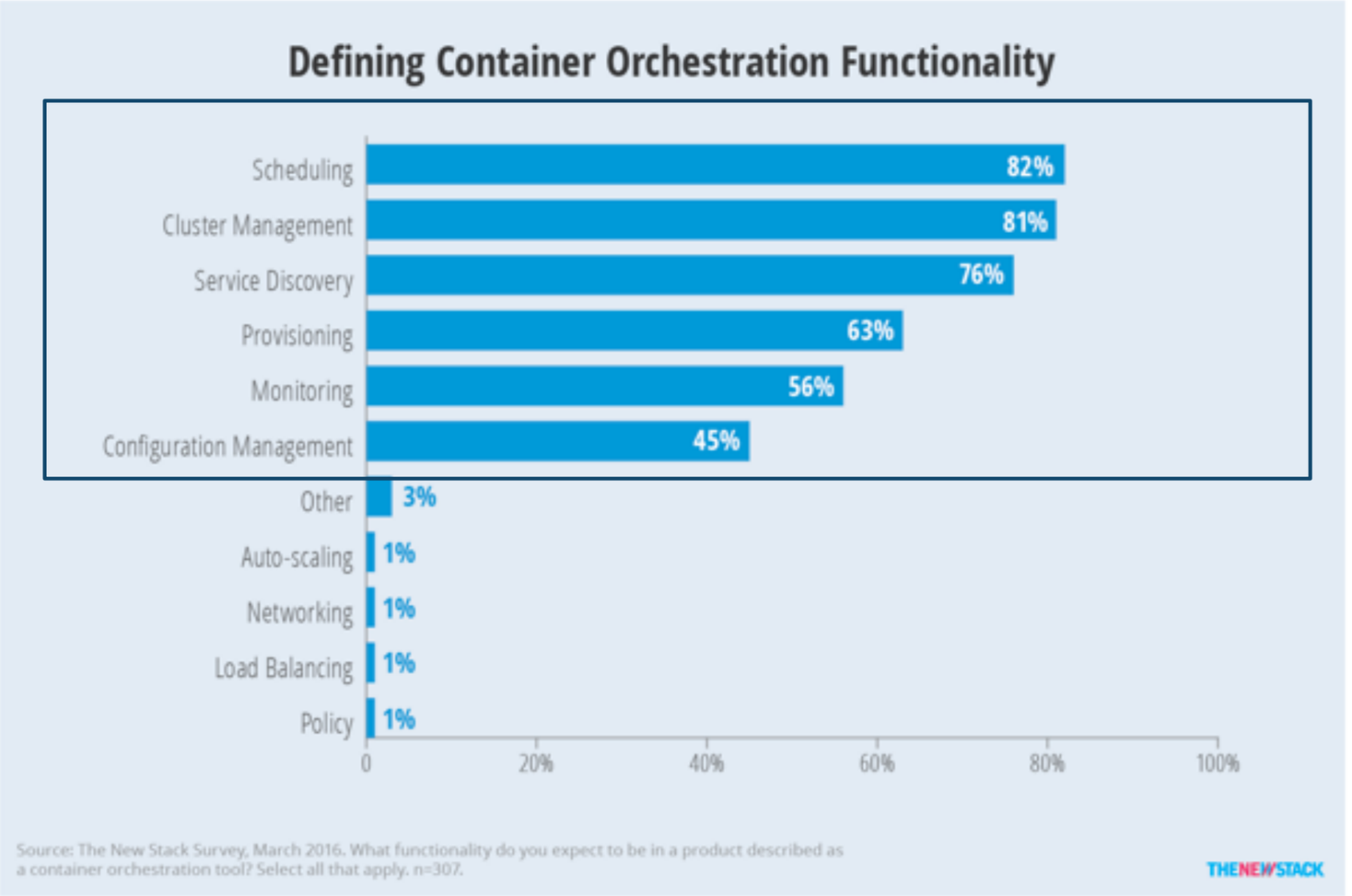
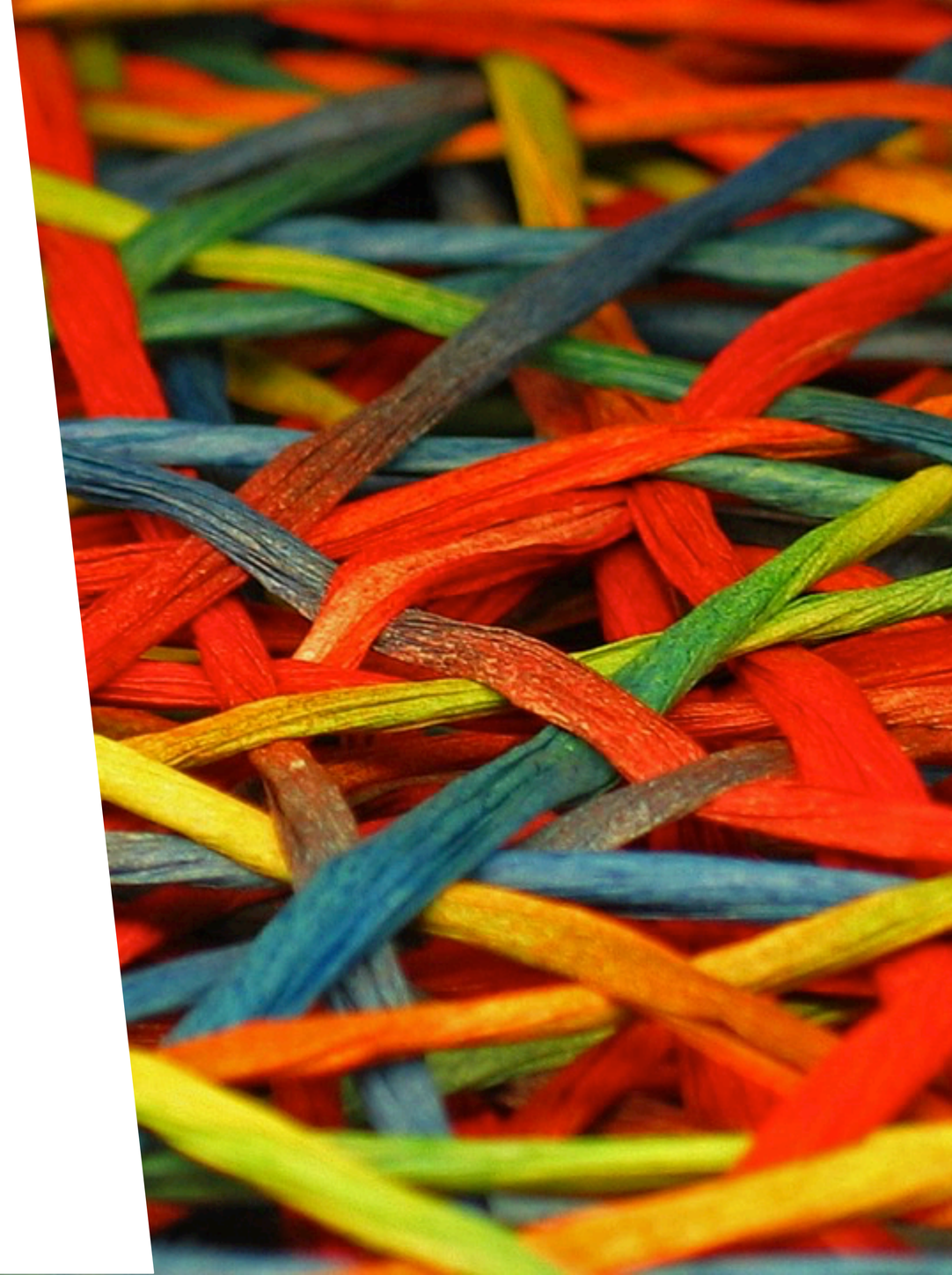


Figure 3: Only 45 percent of respondents consider configuration management to be part of a container orchestration product.

What else do we need for Microservices?

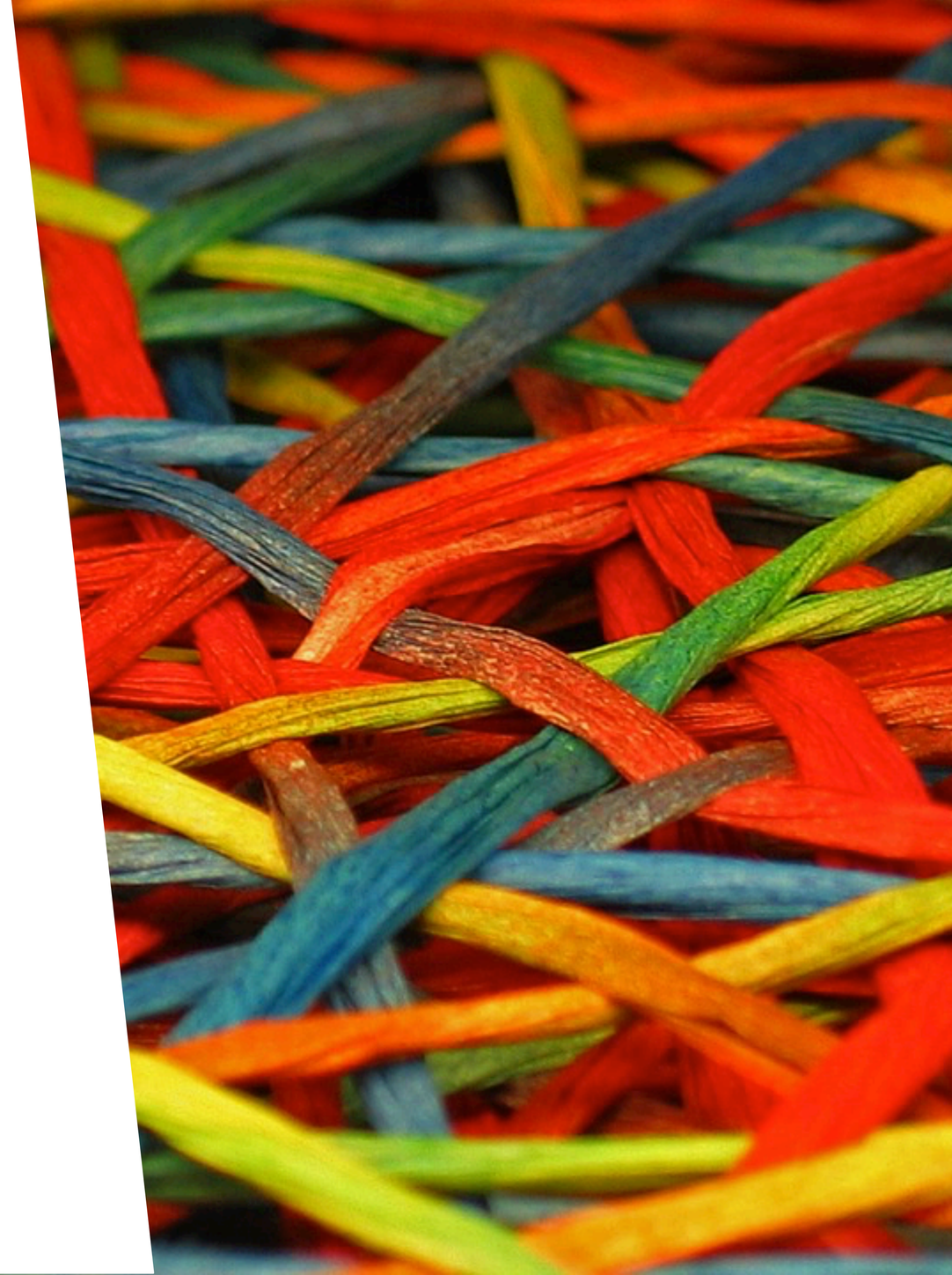
- Visibility
- Resiliency & Efficiency
- Traffic Control
- Security
- Policy Enforcement

Enter Service Mesh



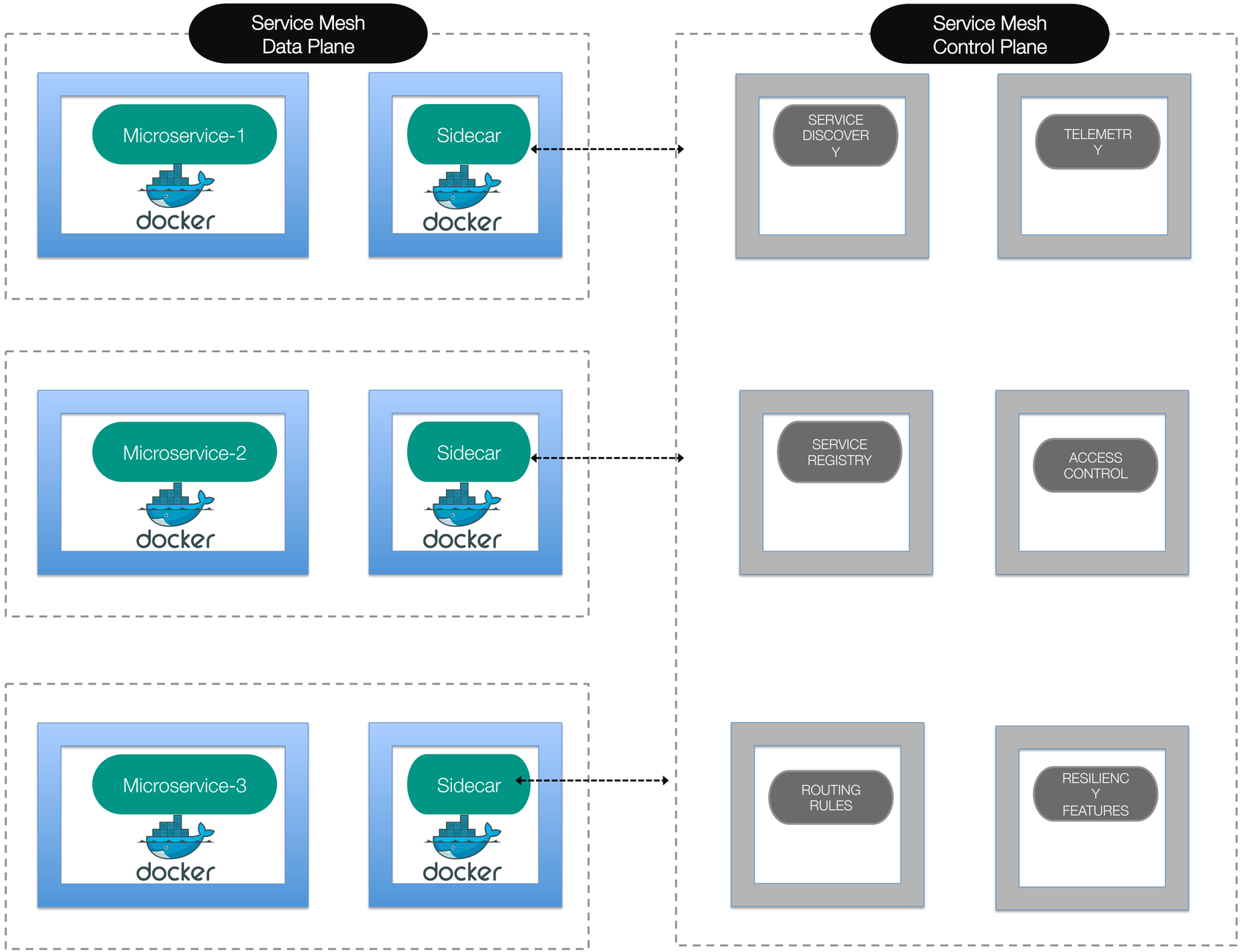
What is a ‘Service Mesh’ ?

- A network for services, not bytes
  - Visibility
  - Resiliency & Efficiency
    - Traffic Control
  - Security
  - Policy Enforcement



How to build a ‘Service Mesh’ ?

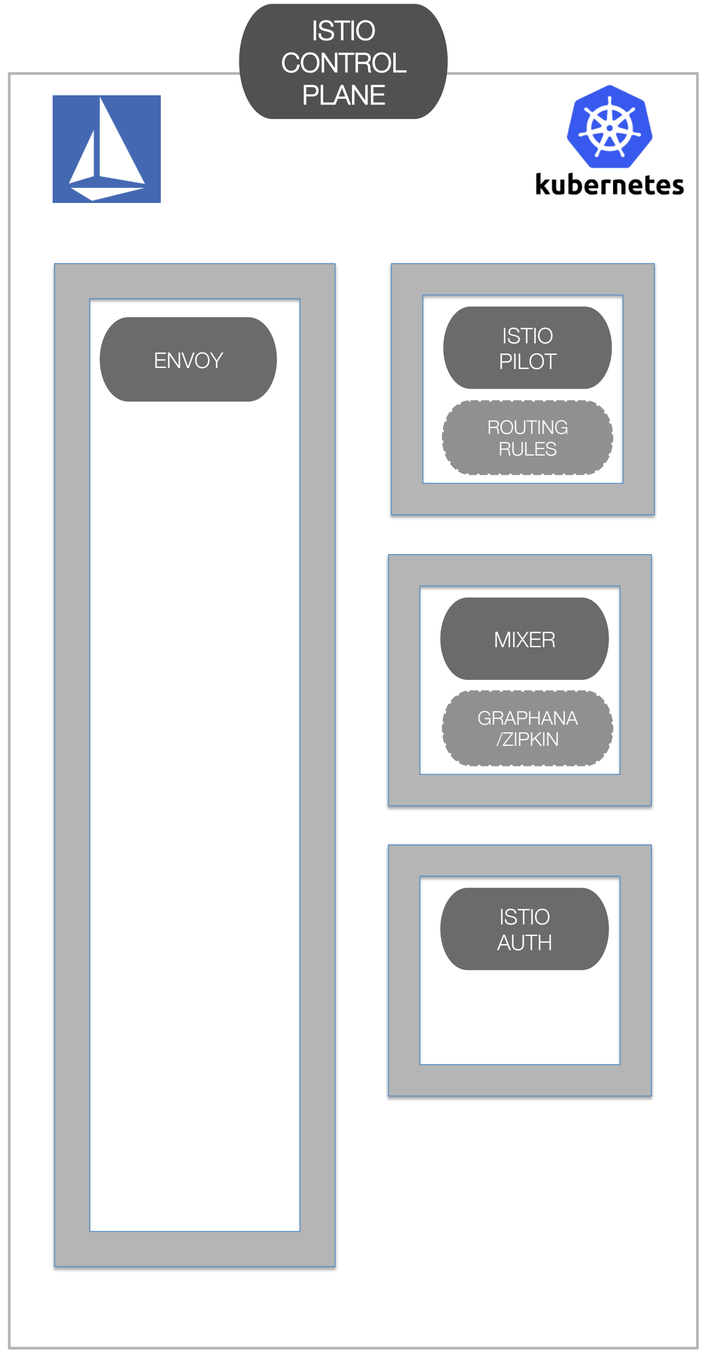
- Lightweight sidecars to manage traffic between services
- Sidecars can do ***much more*** than just load balancing!

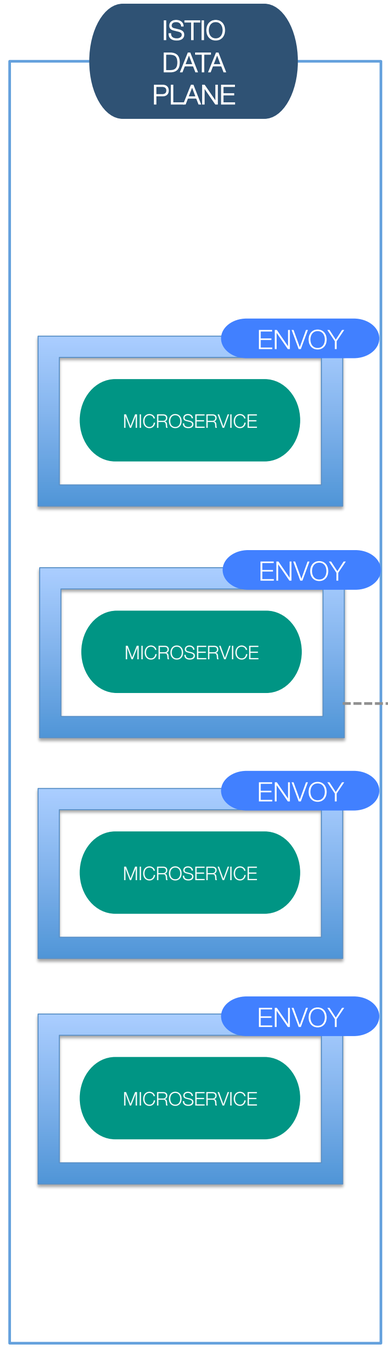


Istio

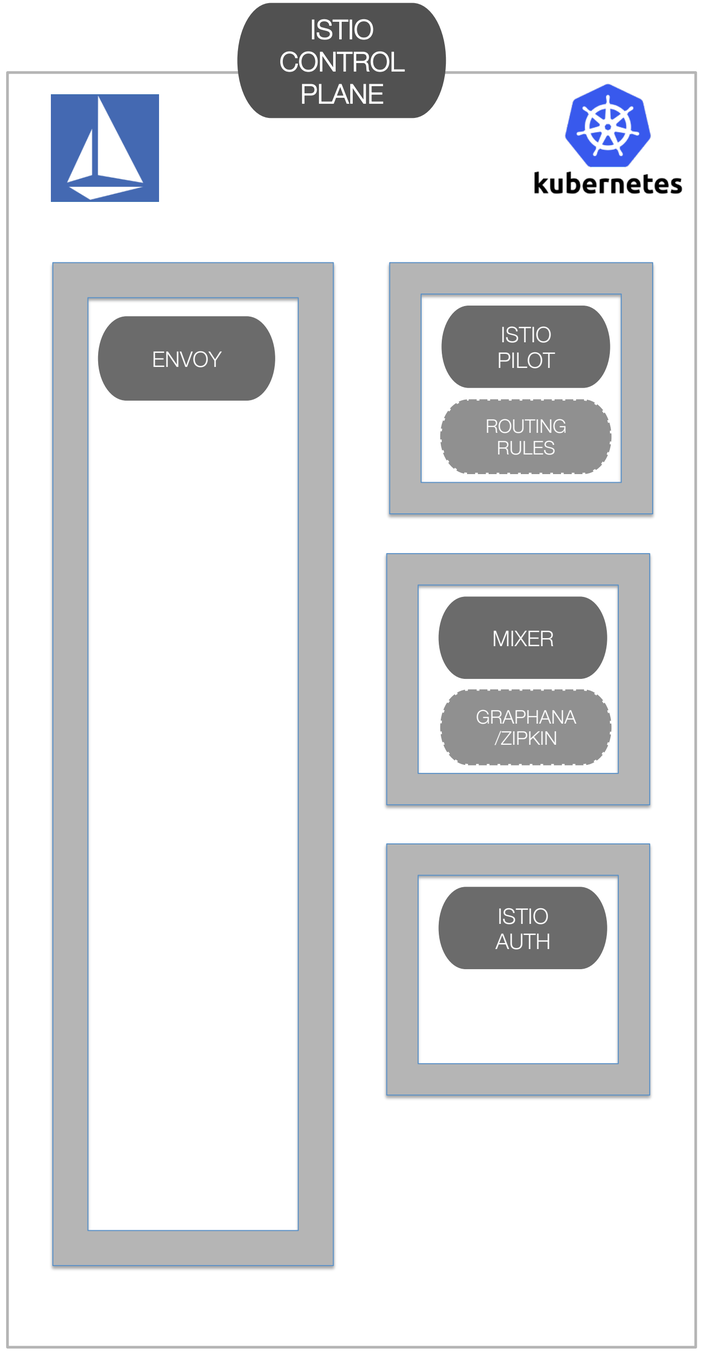
Istio Concepts

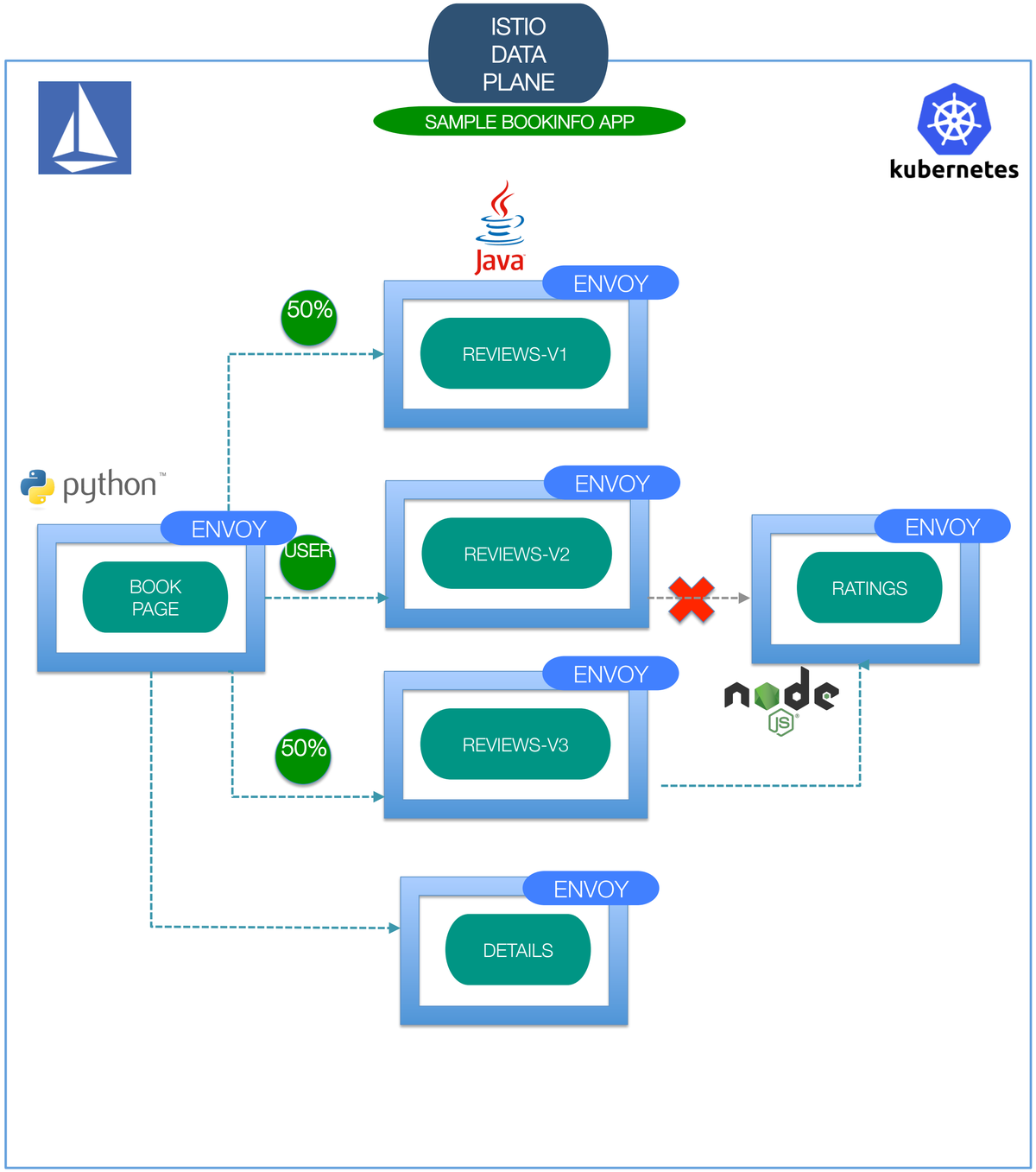
- **Pilot - Configures Istio deployments and propagate configuration to the other components of the system. Routing and resiliency rules go here**
- **Mixer - Responsible for policy decisions and aggregating telemetry data from the other components in the system using a flexible plugin architecture**
- **Proxy – Based on Envoy, mediates inbound and outbound traffic for all Istio-managed services. It enforces access control and usage policies, and provides rich routing, load balancing, and protocol conversion.**



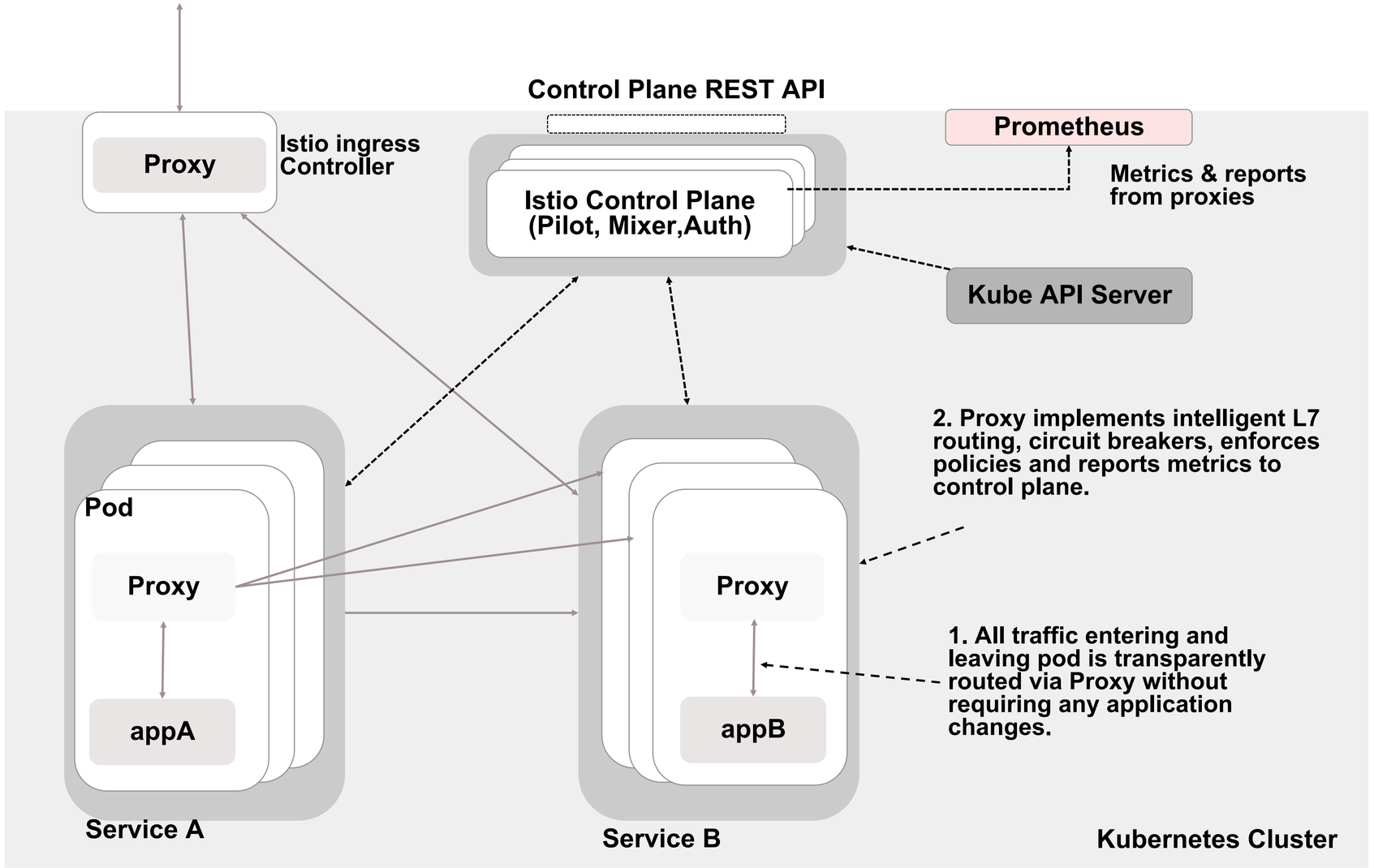


Istio Concepts





Istio Architecture



Proxy. Based on Envoy, a high performance L7 proxy from Lyft, currently being used at large scale in production. https://github.com/lyft/envoy

Istio control plane traffic. Request routing rules, resilience configuration (circuit breakers, timeouts, retries), policies (ACLs, rate limits, auth), and metrics/reports from proxies.

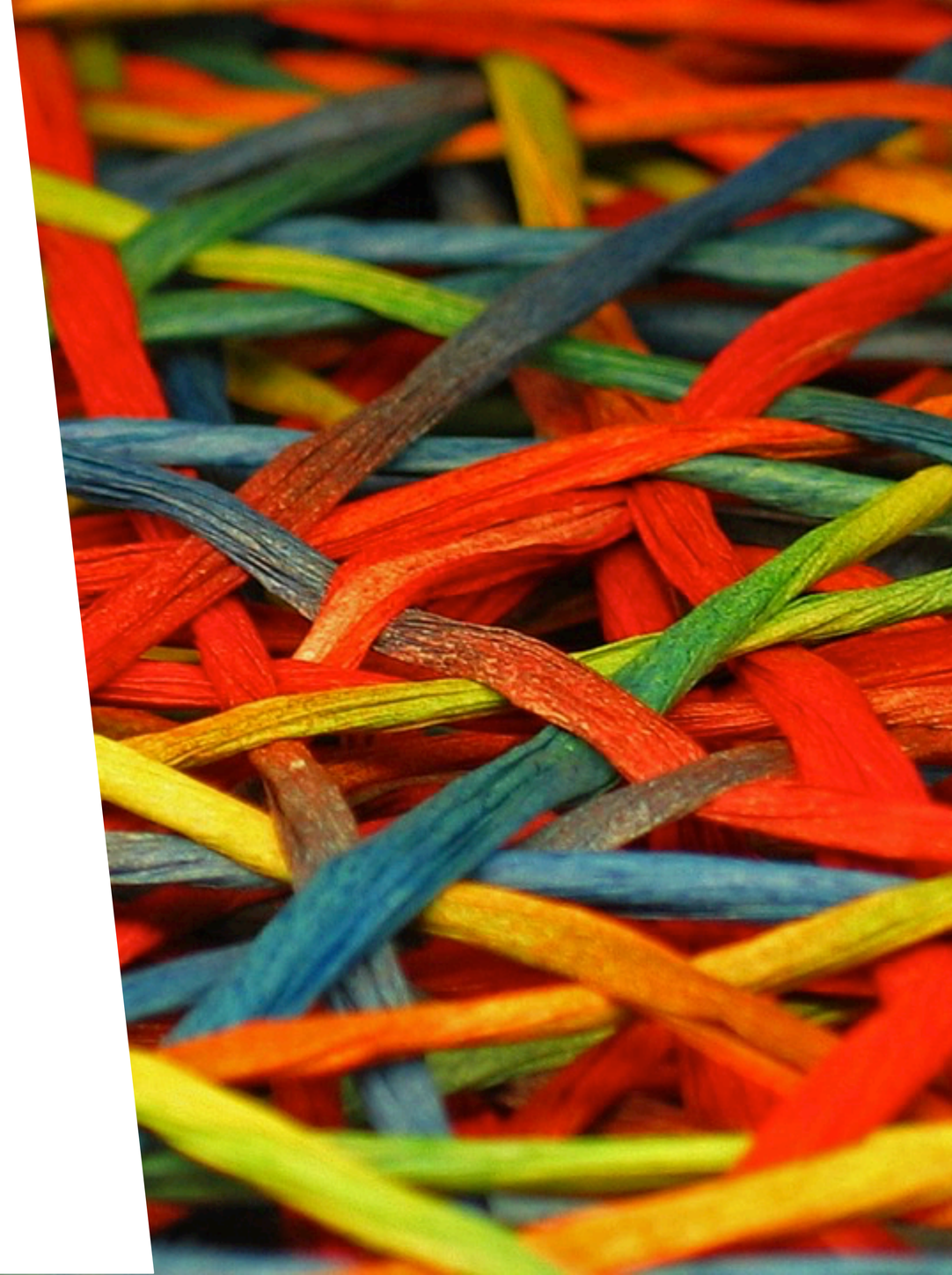
User/application traffic. HTTP/1.1, HTTP/2, gRPC, TCP with or without TLS

Kubernetes, Microservices and Istio Developer Journeys

What is a ‘Service Mesh’ ?

A network for services, not bytes

- **Resiliency & Efficiency**
- Traffic Control
- Visibility
- Security
- Policy Enforcement



Resiliency

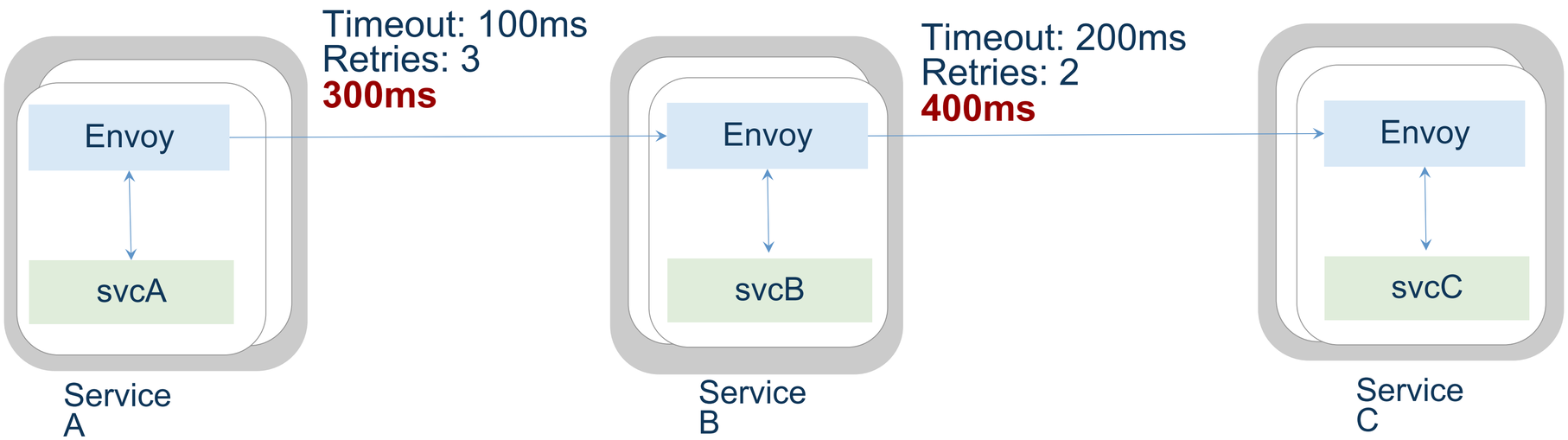
- Istio adds fault tolerance to your application without any changes to code

```
// Circuit breakers
destination: serviceB.example.cluster.local
policy:
- tags:
    version: v1
  circuitBreaker:
    simpleCb:
      maxConnections: 100
      httpMaxRequests: 1000
      httpMaxRequestsPerConnection: 10
      httpConsecutiveErrors: 7
      sleepWindow: 15m
      httpDetectionInterval: 5m
```

- **Resilience features**
  - ❖ Timeouts
  - ❖ Retries with timeout budget
  - ❖ Circuit breakers
  - ❖ Health checks
  - ❖ AZ-aware load balancing w/ automatic failover
  - ❖ Control connection pool size and request load
  - ❖ Systematic fault injection

Resiliency Testing

- Systematic fault injection to identify weaknesses in failure recovery policies
  - HTTP/gRPC error codes
  - Delay injection

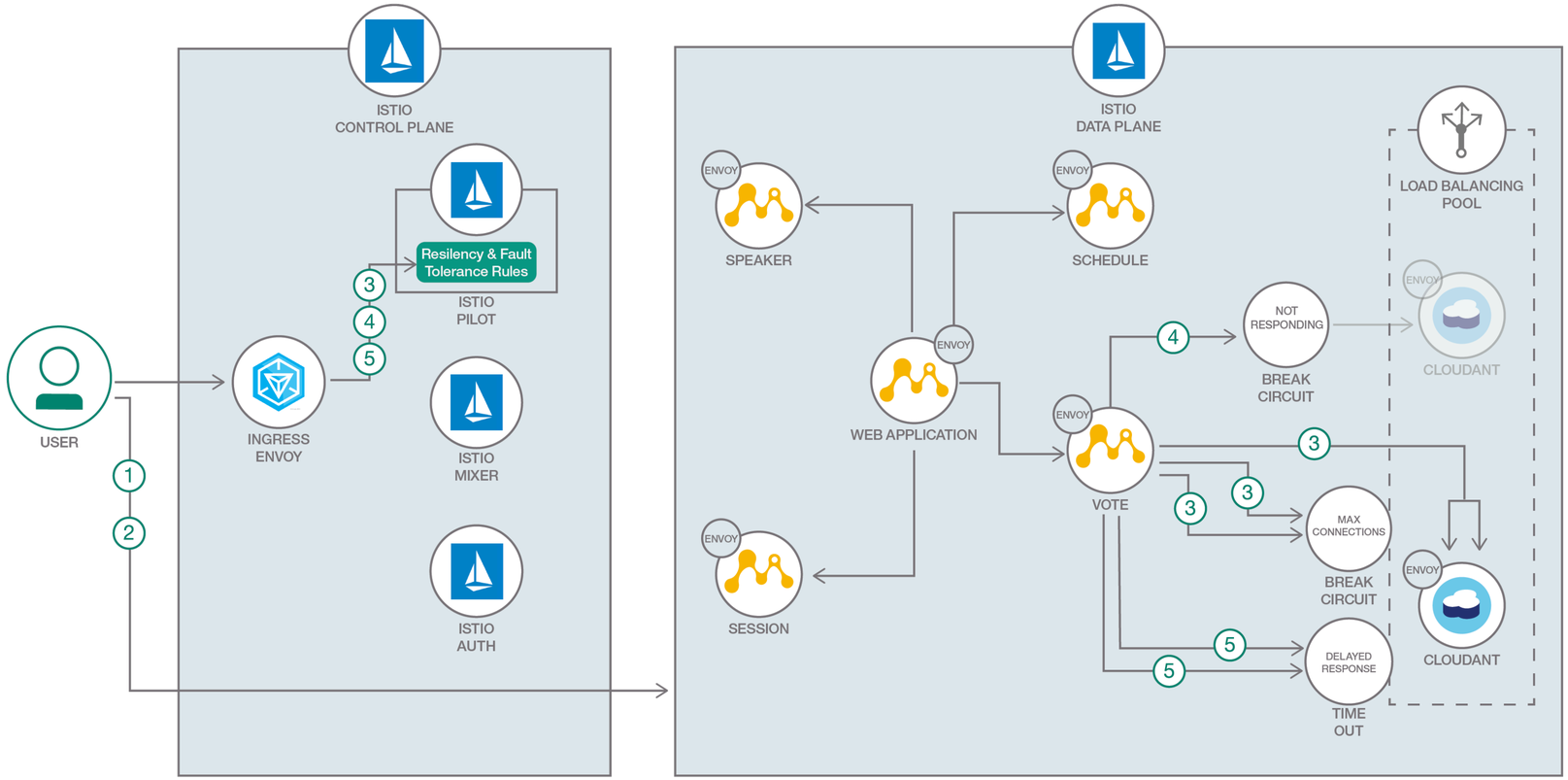


Developer Journey:

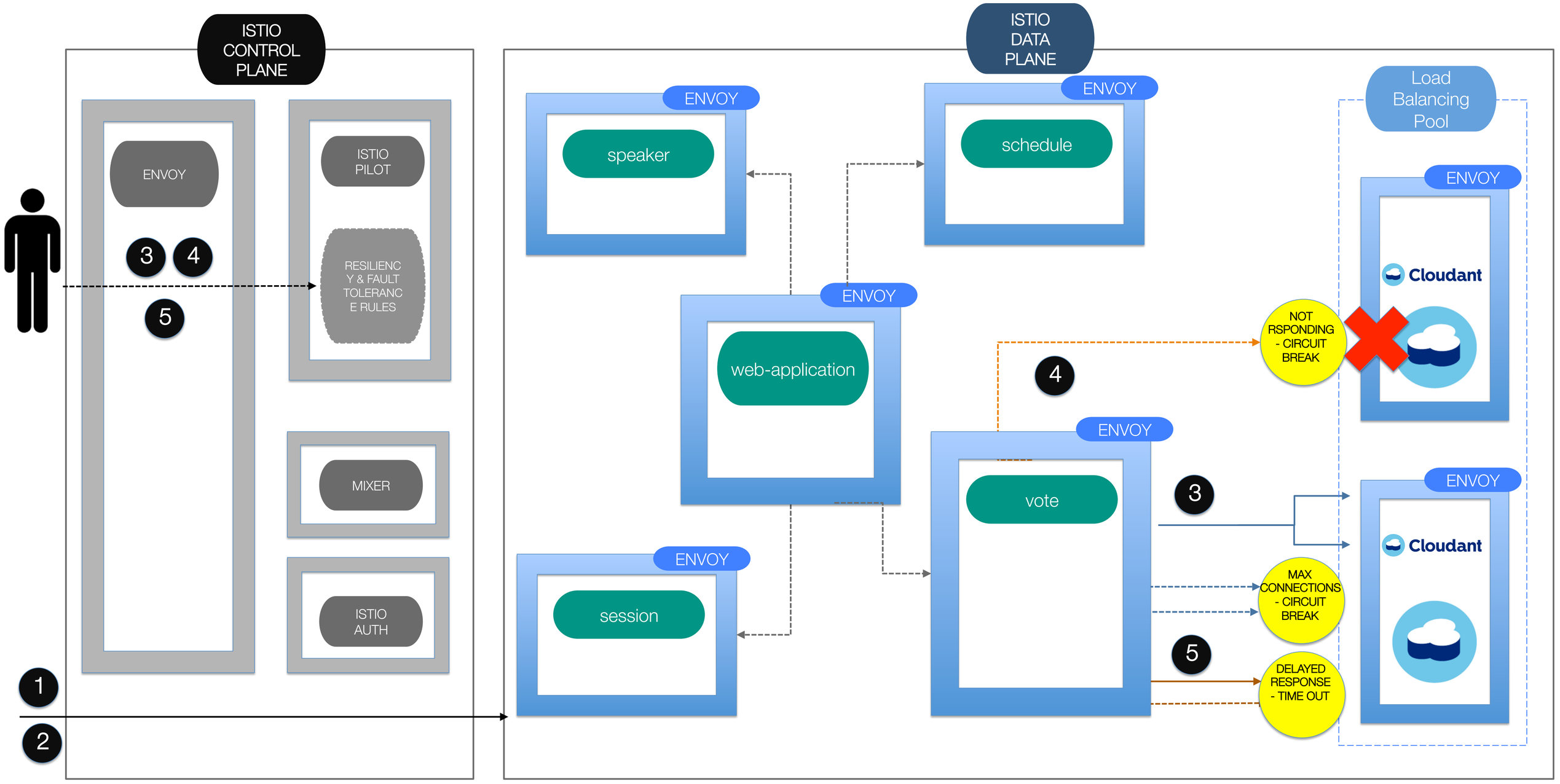
Leverage Istio to create resilient and fault tolerant Microservices

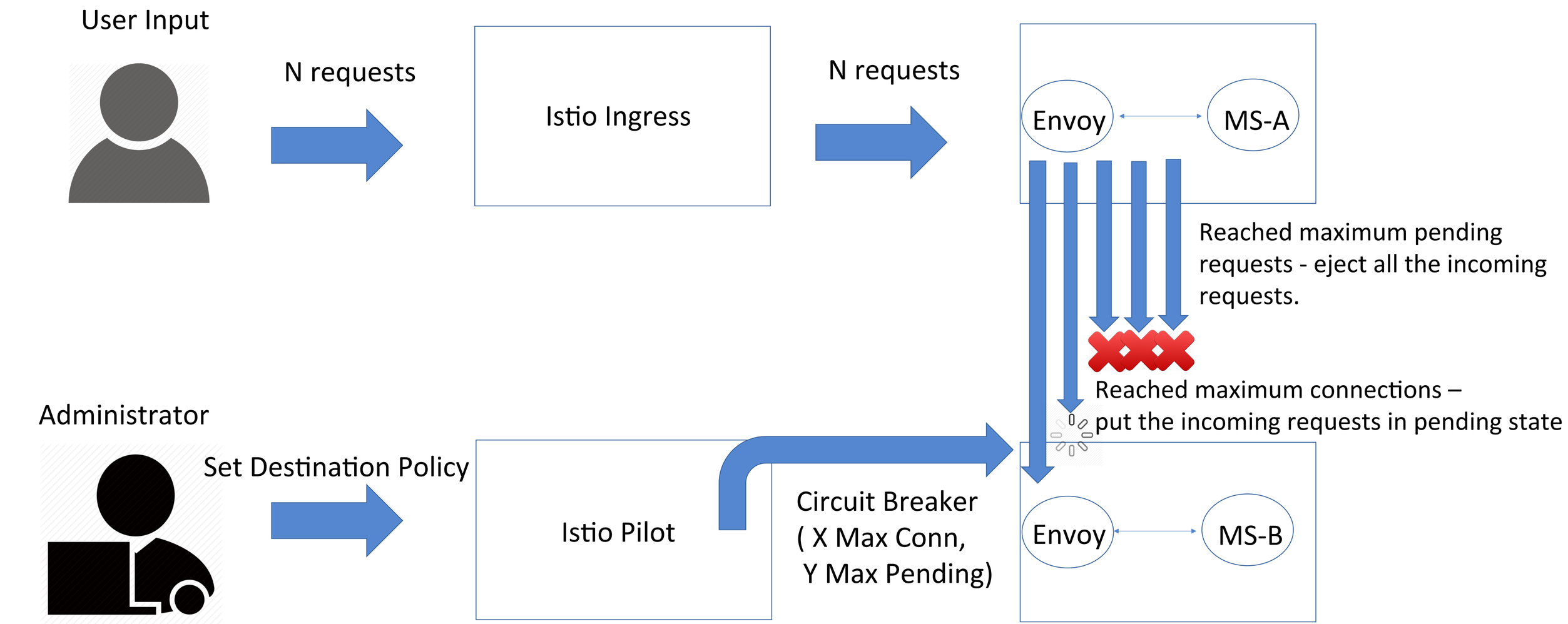
Twelve-factor apps make a strong case for designing and implementing your microservices for failure. What that means is with the proliferation of microservices, failure is inevitable, and applications should be fault-tolerant. Istio, a service mesh, can help make your microservices resilient without changing application code.

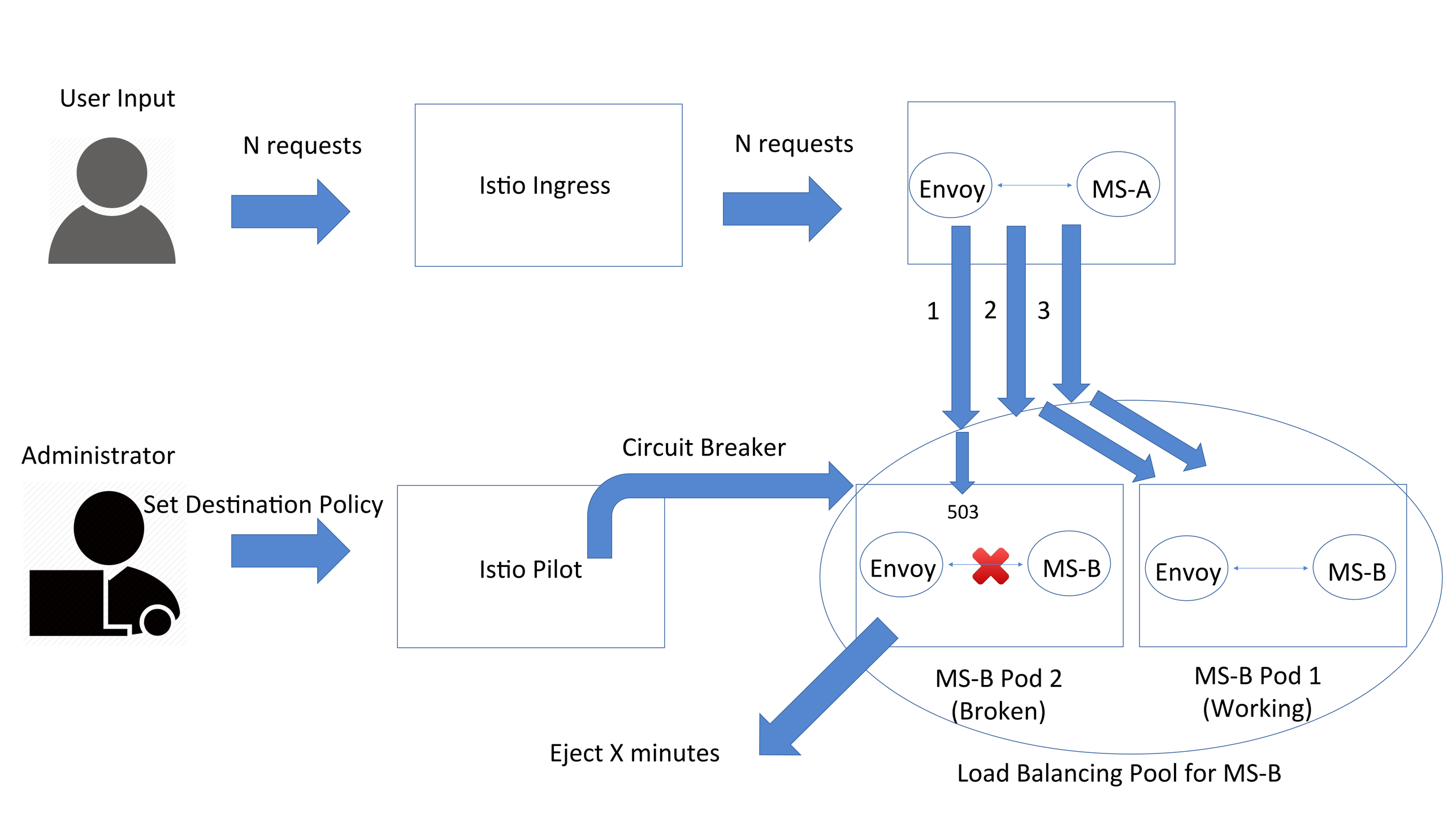
Developer Works Code: https://developer.ibm.com/code/journey/make-java-microservices-resilient-with-istio/
Github: https://github.com/IBM/resilient-java-microservices-with-istio

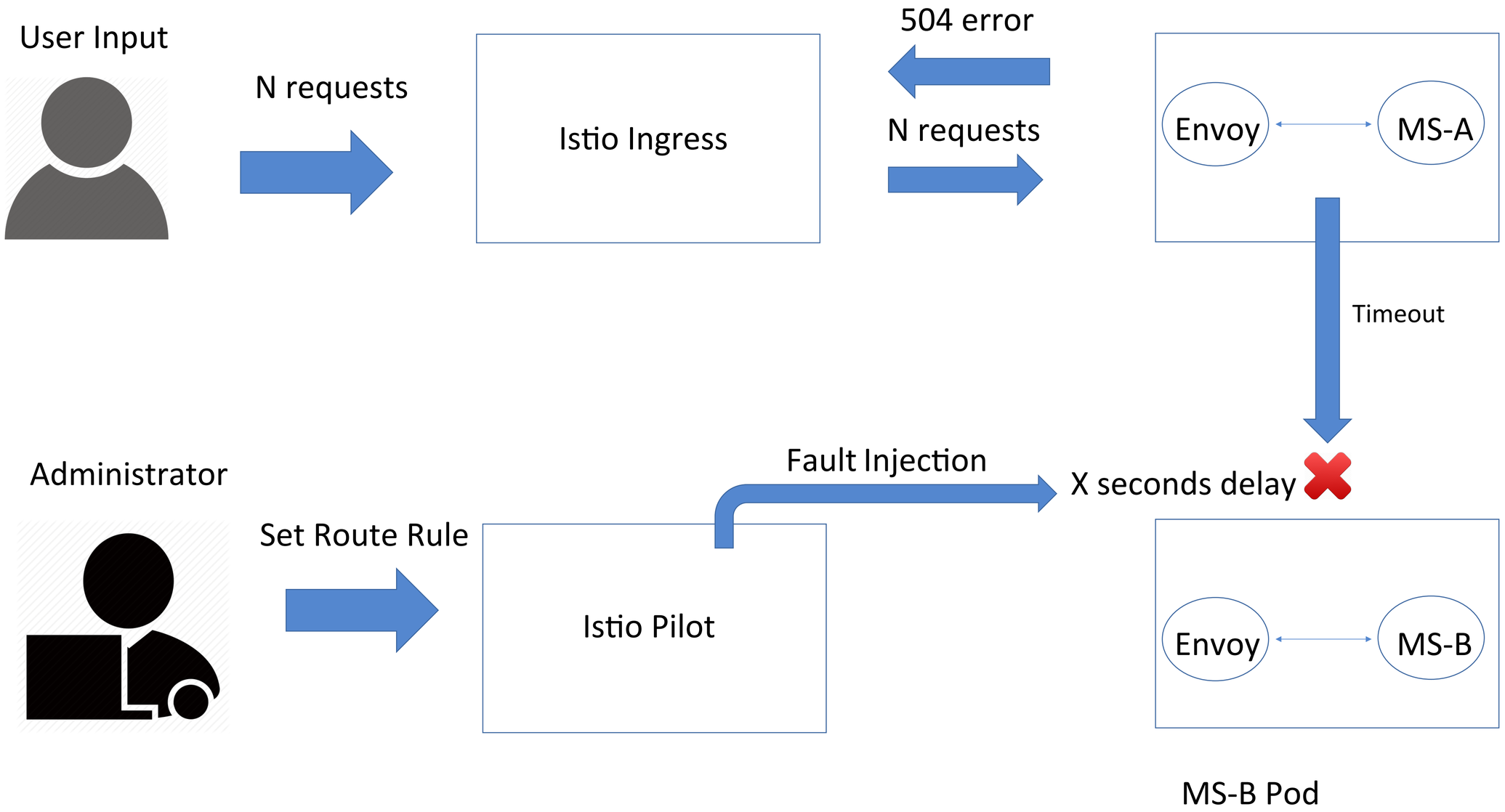


Resilient Microservices with Istio




User Input
N requests
Istio Ingress
N requests
Envoy
MS-A
Reached maximum pending requests - eject all the incoming requests.
Reached maximum connections – put the incoming requests in pending state
Administrator
Set Destination Policy
Istio Pilot
Circuit Breaker
(X Max Conn,
Y Max Pending)
Envoy
MS-B

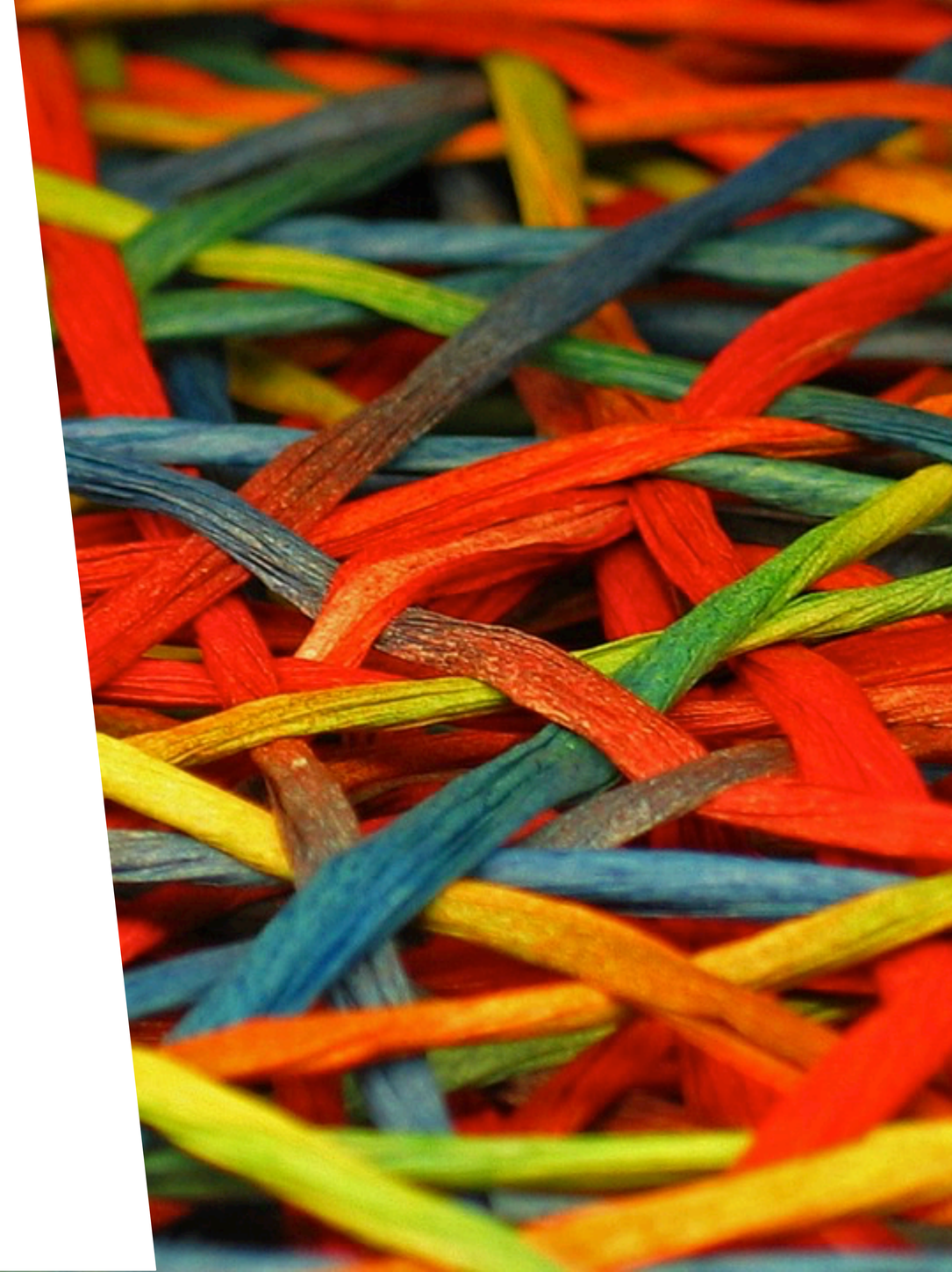

User Input
N requests
Istio Ingress
N requests
Envoy
MS-A
1
2
3
Administrator
Set Destination Policy
Circuit Breaker
Istio Pilot
503
Envoy
MS-B
Envoy
MS-B
MS-B Pod 2
(Broken)
MS-B Pod 1
(Working)
Eject X minutes
Load Balancing Pool for MS-B


User Input
N requests
Istio Ingress
504 error
N requests
Envoy
MS-A
Timeout
Administrator
Set Route Rule
Istio Pilot
Fault Injection
X seconds delay
Envoy
MS-B
MS-B Pod

What is a ‘Service Mesh’ ?

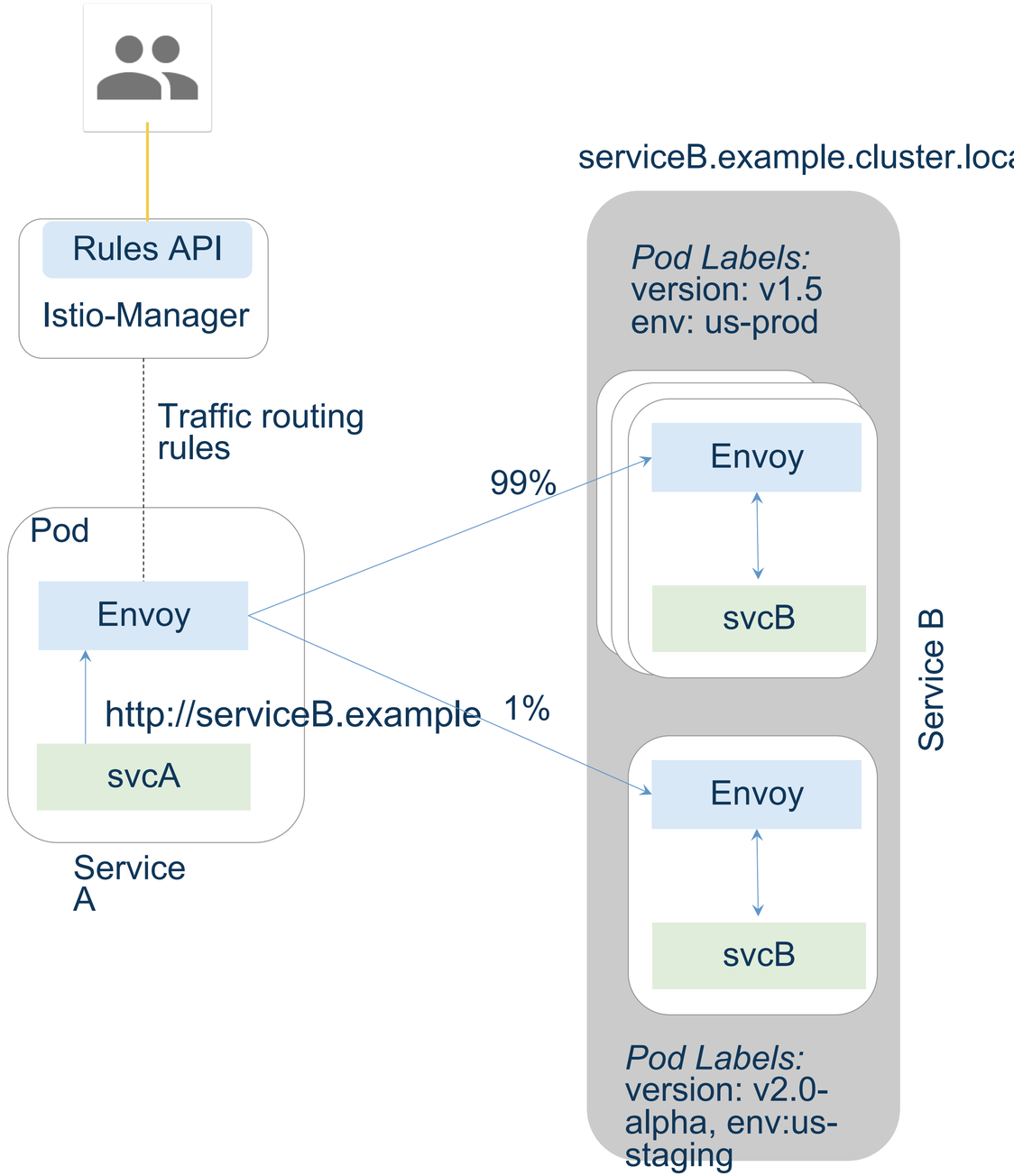
A network for services, not bytes

- Resiliency & Efficiency
- **Traffic Control**
- **Visibility**
- Security
- Policy Enforcement



Traffic Splitting

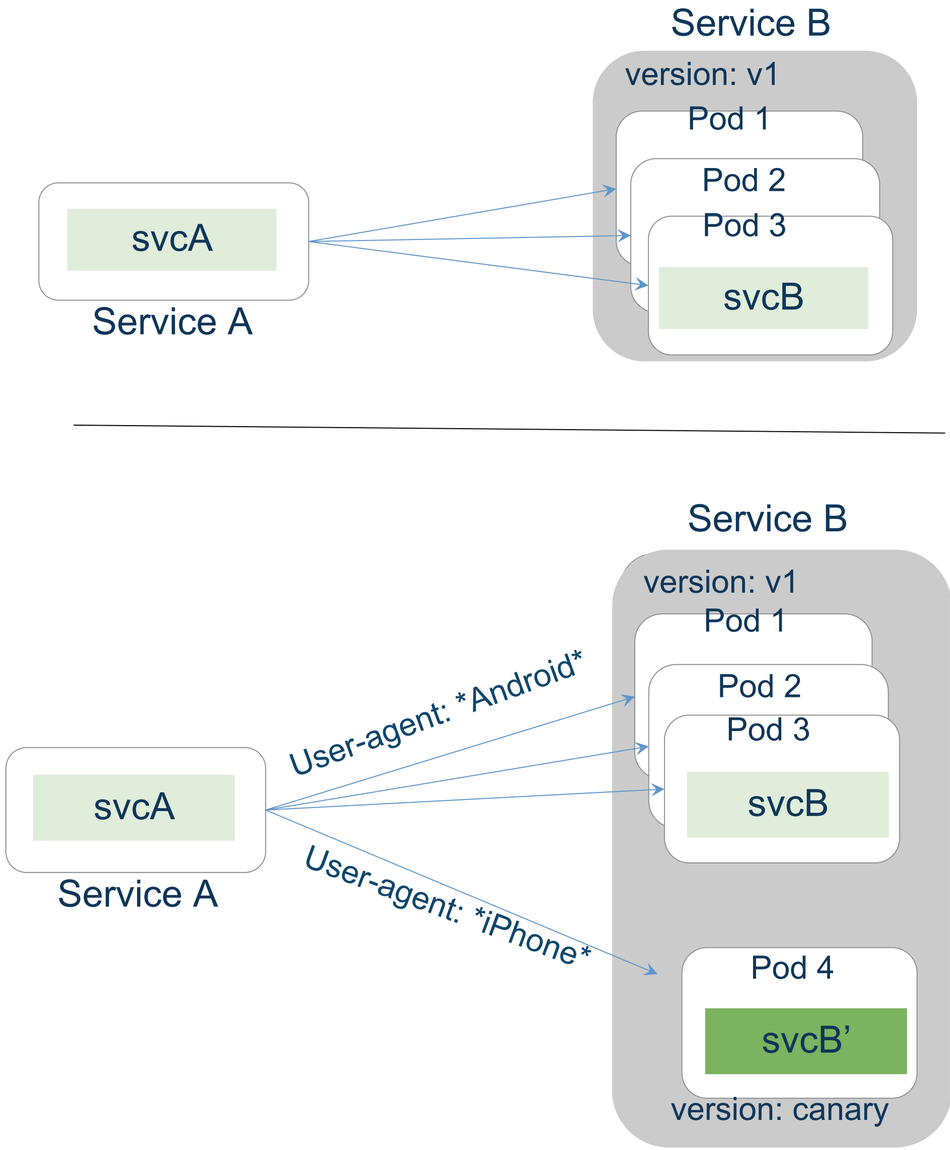
```
// A simple traffic splitting rule
destination:
  serviceB.example.cluster.local
match:
    source:
  serviceA.example.cluster.local
  route:
  - tags:
      version: v1.5
      env: us-prod
    weight: 99
  - tags:
      version: v2.0-alpha
      env: us-staging
    weight: 1
```



Traffic Steering

- // Content-based traffic steering rule

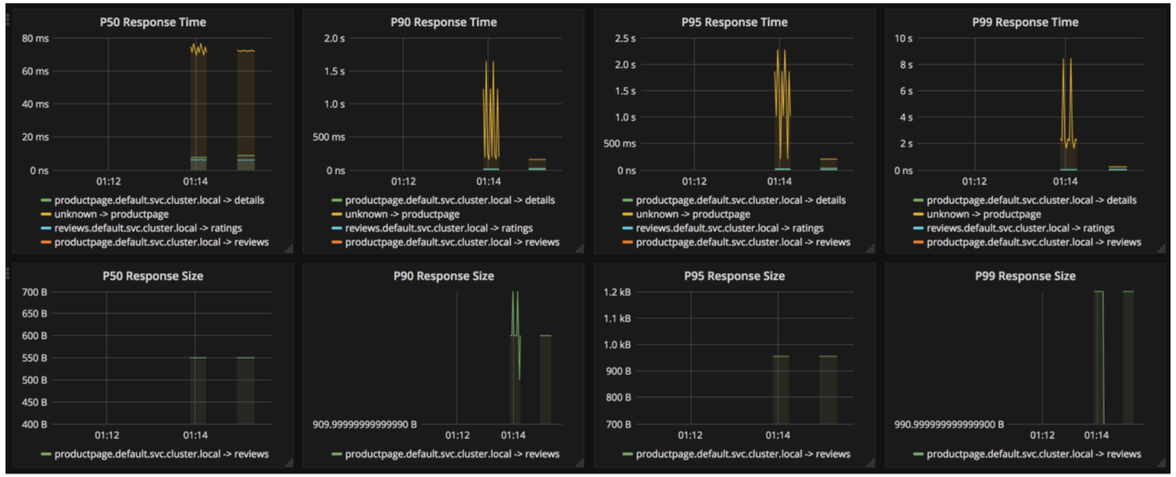
- **destination**: serviceB.example.cluster.local
  match:
    httpHeaders:
      user-agent:
        regex: ^(.*?;)?(iPhone)(;.*)?$
  precedence: 2
  route:
  - tags:
      version: canary



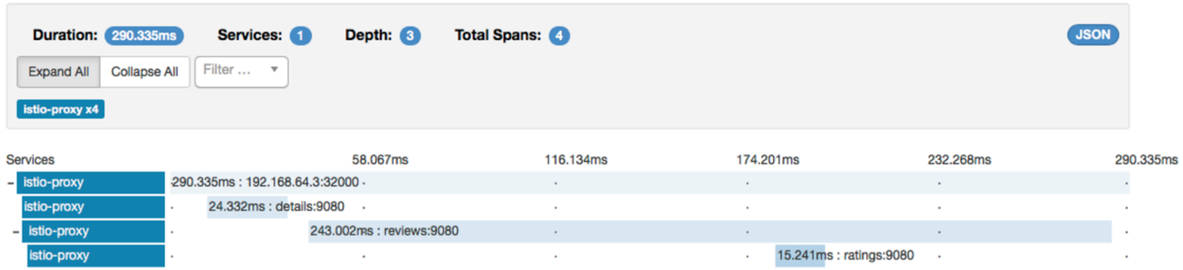
Content-based traffic steering

Visibility

- *Monitoring & tracing should not be an afterthought in the infrastructure*
- Goals
  - Metrics without instrumenting apps
  - Consistent metrics across fleet
  - Trace flow of requests across services
  - Portable across metric backend providers



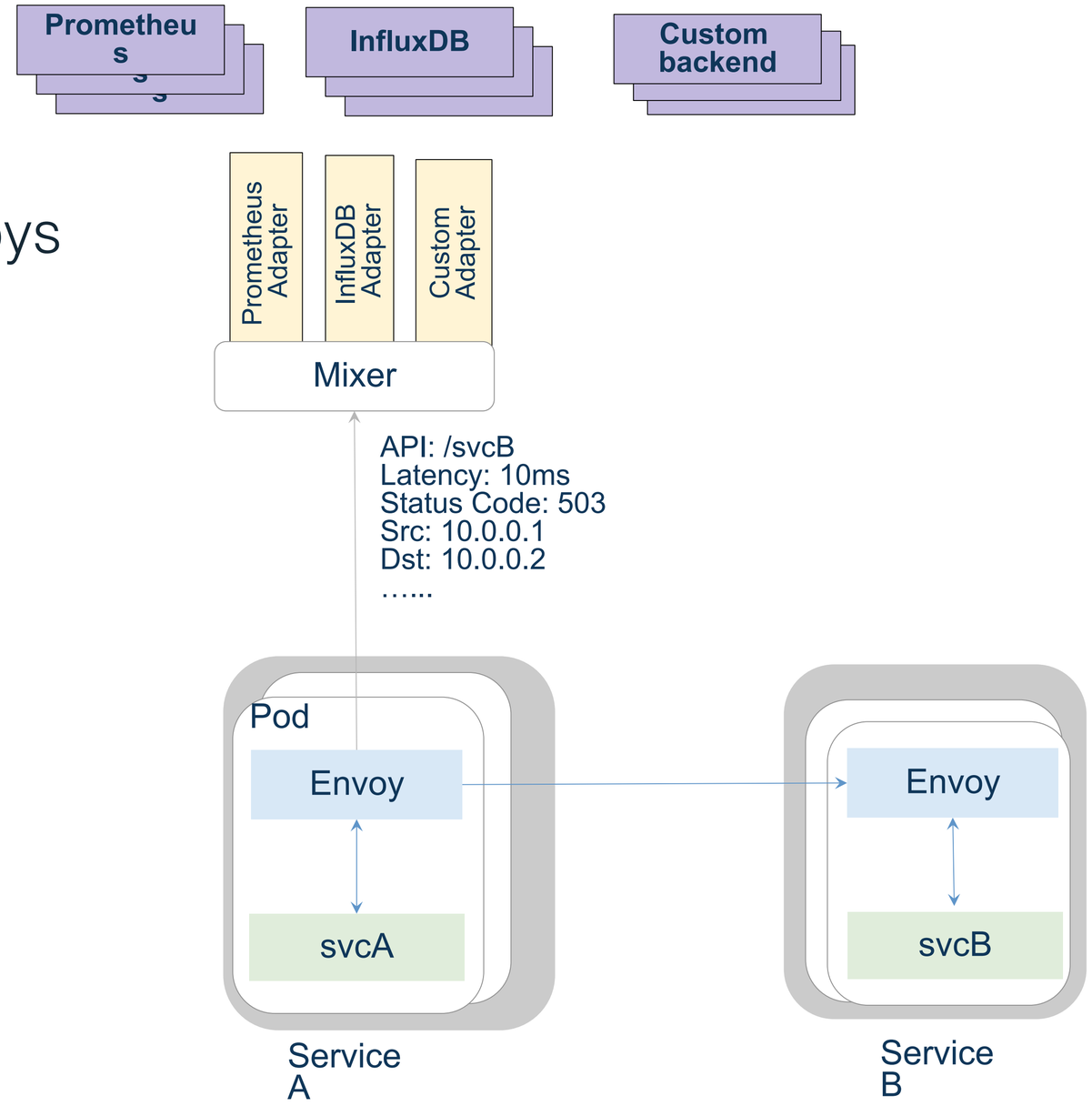
Istio - Grafana dashboard w/ Prometheus backend



Istio Zipkin tracing dashboard

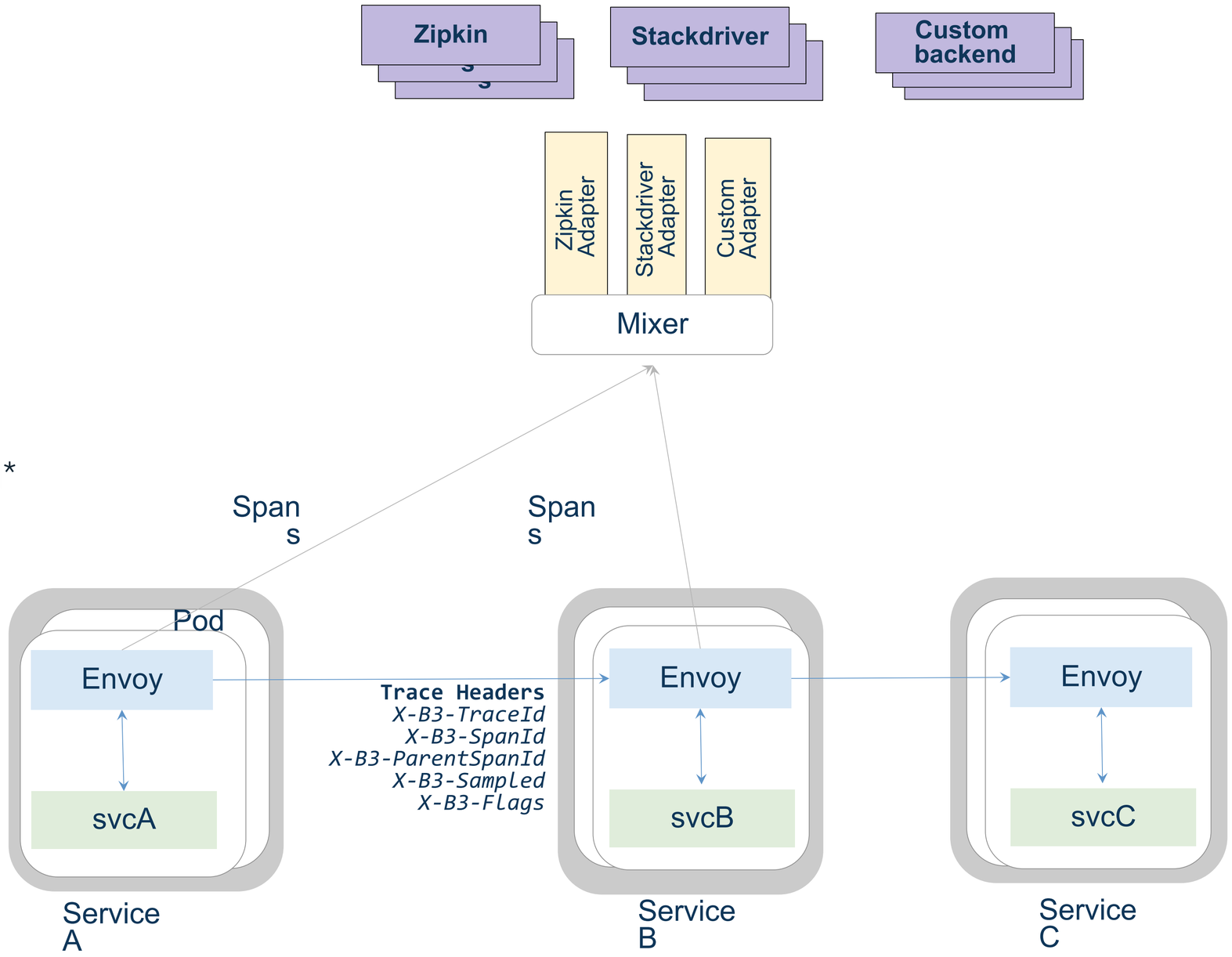
Metric Flow

- Mixer collects metrics emitted by Envoys
- Adapters in the Mixer normalize and forward to monitoring backends
- Metrics backend can be swapped at runtime



Visibility : Tracing

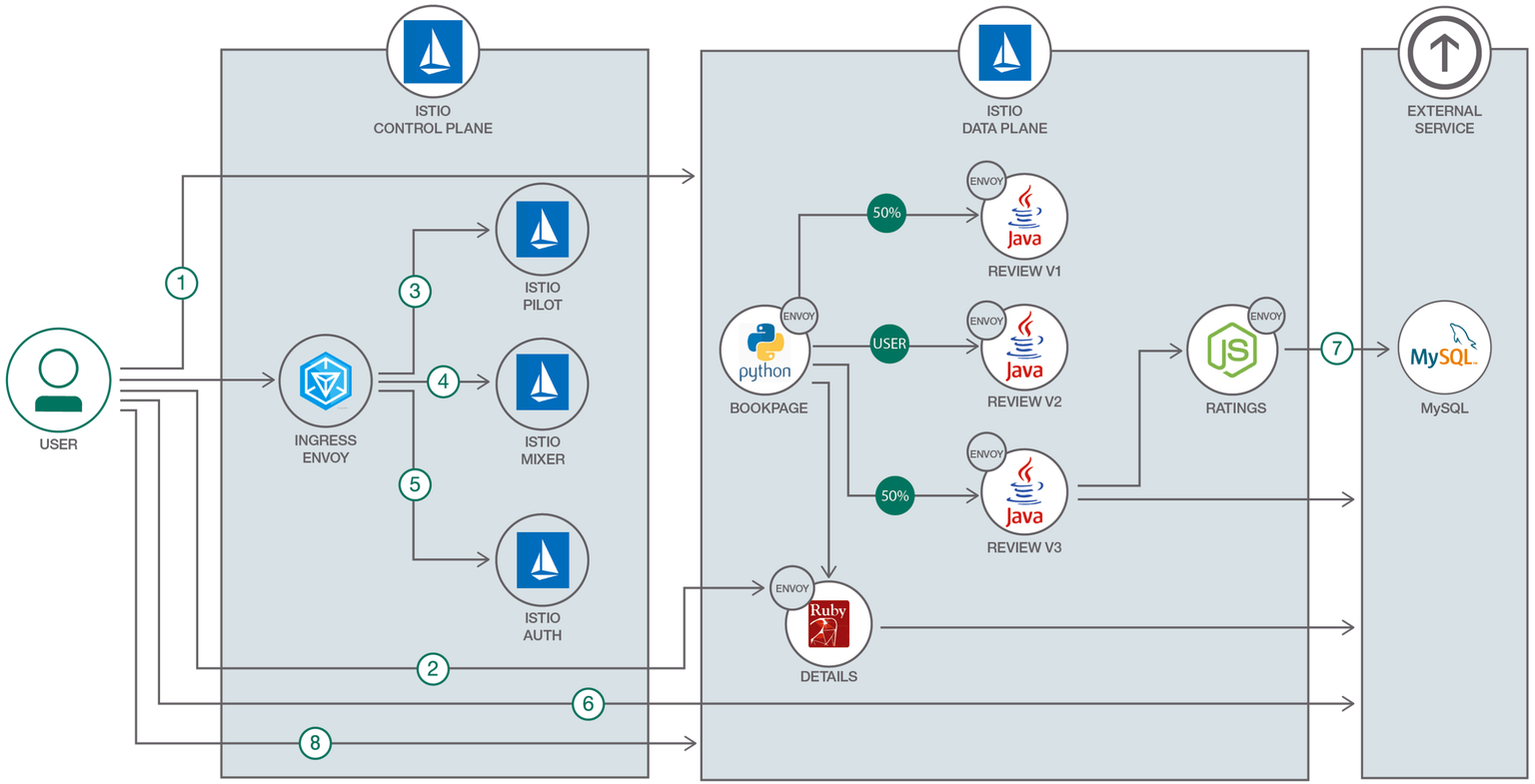
- Application do not have to deal with generating spans or correlating causality
- Envoys generate spans
  - Applications need to *forward* context headers on outbound calls
- Envoys send traces to Mixer
- Adapters at Mixer send traces to respective backends

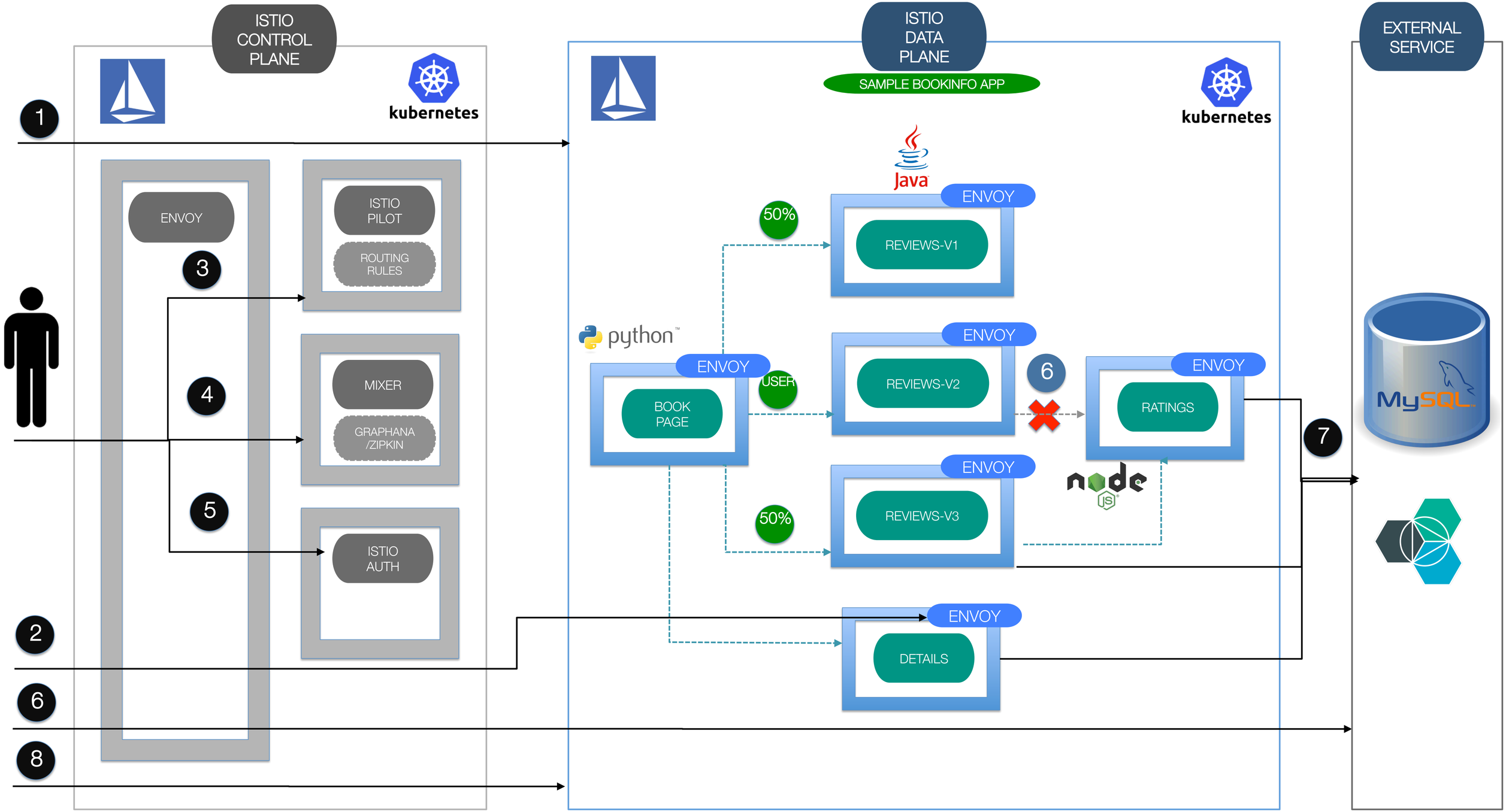


Developer Journey:
Manage micro services traffic using Istio on Kubernetes

Microservices and containers have changed application design and deployment patterns. They have also introduced new challenges, such as service discovery, routing, failure handling, and visibility to microservices. Kubernetes can handle multiple container-based workloads, including microservices, but when it comes to more sophisticated features like traffic management, failure handling, and resiliency, a microservices mesh like Istio is required.

Developer Works Code: https://developer.ibm.com/code/journey/manage-microservices-traffic-using-istio/
Github: https://github.com/IBM/microservices-traffic-management-using-istio




ISTIO CONTROL PLANE
kubernetes
ENVOY
ISTIO PILOT
ROUTING RULES
MIXER
GRAPHANA /ZIPKIN
ISTIO AUTH
ISTIO DATA PLANE
SAMPLE BOOKINFO APP
kubernetes
Java
ENVOY
REVIEWS-V1
50%
python
ENVOY
BOOK PAGE
USER
ENVOY
REVIEWS-V2
6
ENVOY
RATINGS
node
ENVOY
REVIEWS-V3
50%
ENVOY
DETAILS
EXTERNAL SERVICE
MySQL
1
2
3
4
5
6
7
6
8

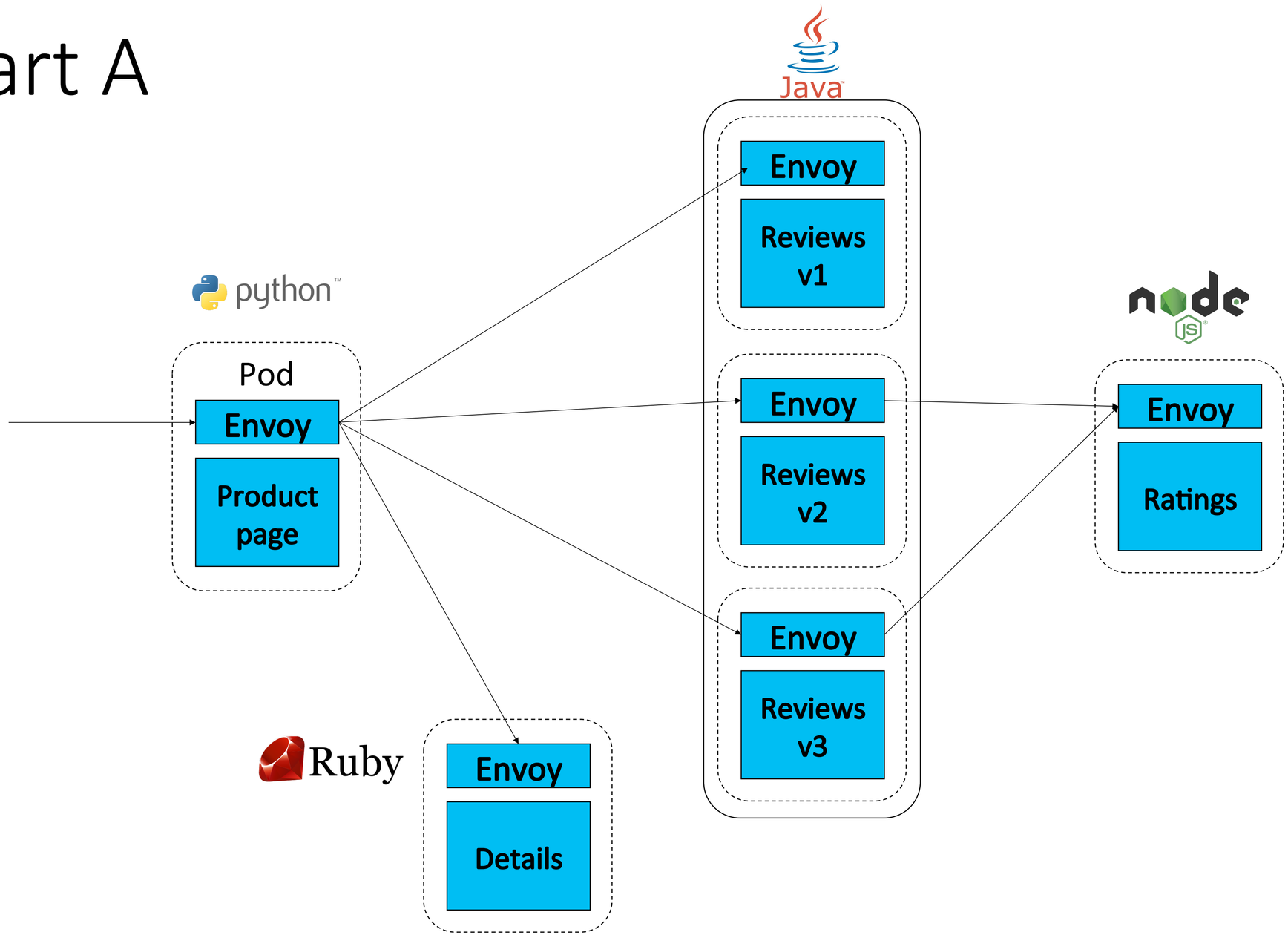
Developer Journey:
Manage micro services traffic using Istio on Kubernetes

Microservices and containers have changed application design and deployment patterns. They have also introduced new challenges, such as service discovery, routing, failure handling, and visibility to microservices. Kubernetes can handle multiple container-based workloads, including microservices, but when it comes to more sophisticated features like traffic management, failure handling, and resiliency, a microservices mesh like Istio is required.

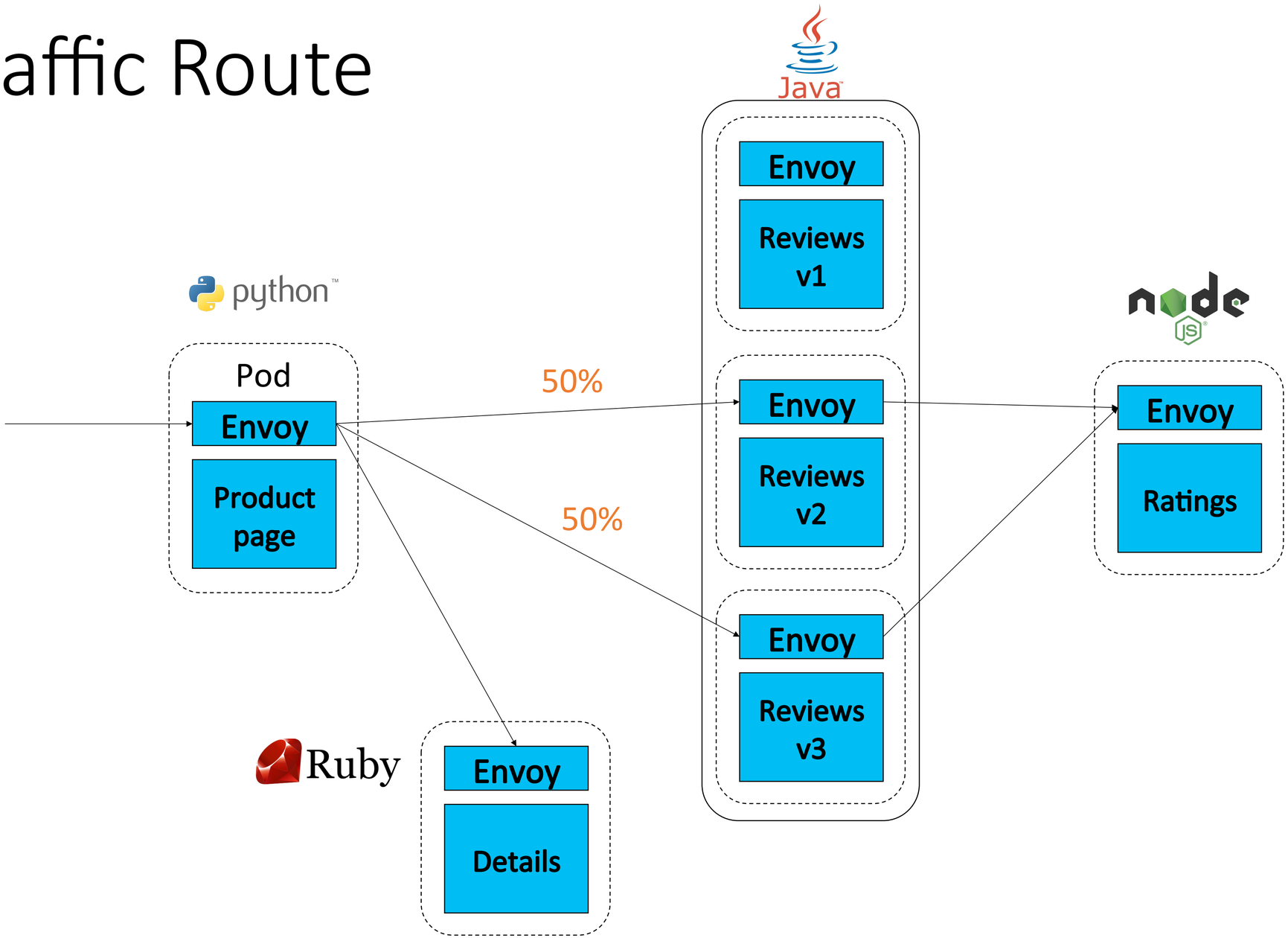
Developer Works Code: https://developer.ibm.com/code/journey/manage-microservices-traffic-using-istio/
Github: https://github.com/IBM/microservices-traffic-management-using-istio

DEMO

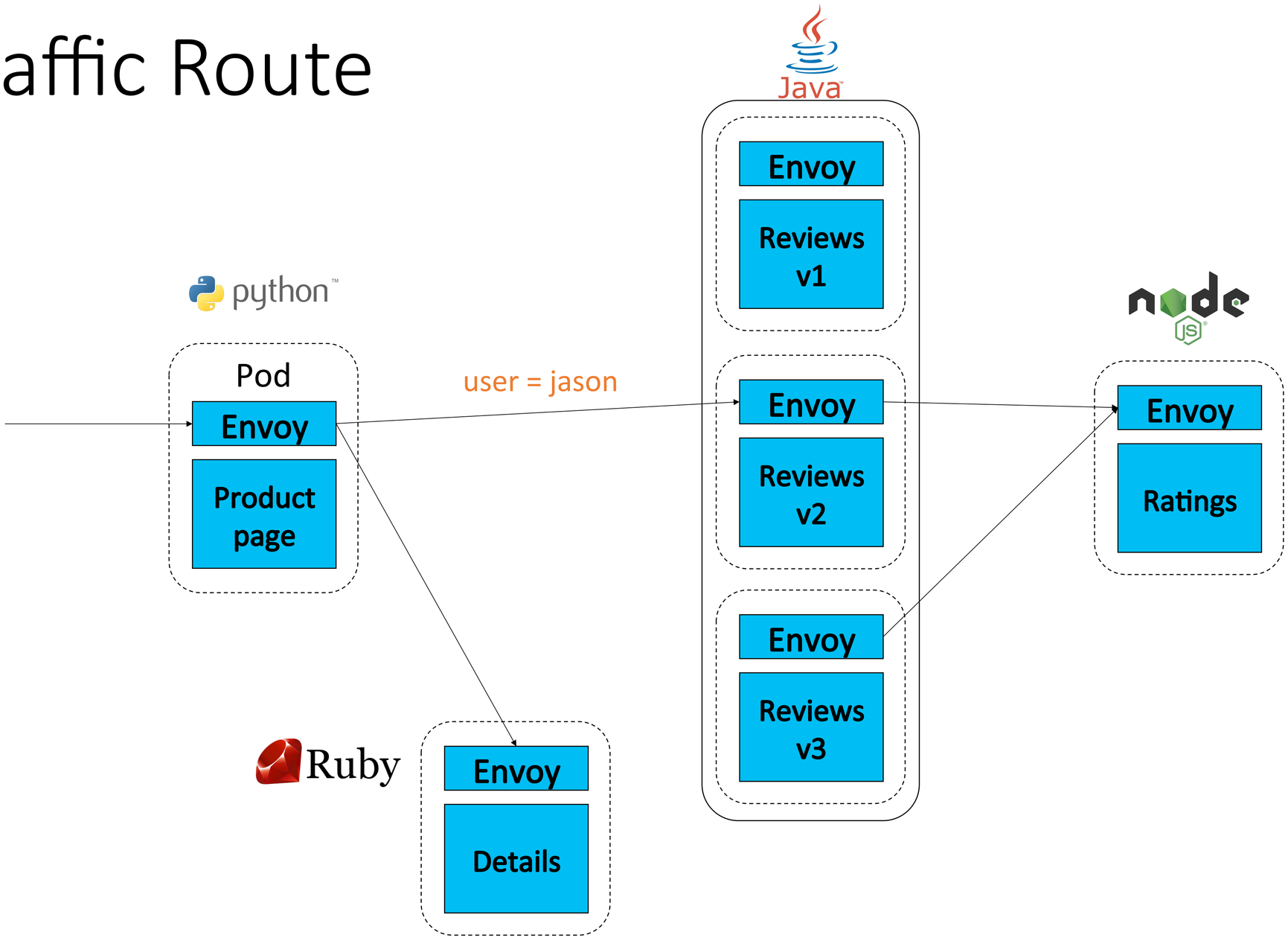
Part A


Java
python
Pod
Envoy
Product page
Envoy
Reviews v1
Envoy
Reviews v2
Envoy
Reviews v3
node
Envoy
Ratings
Ruby
Envoy
Details

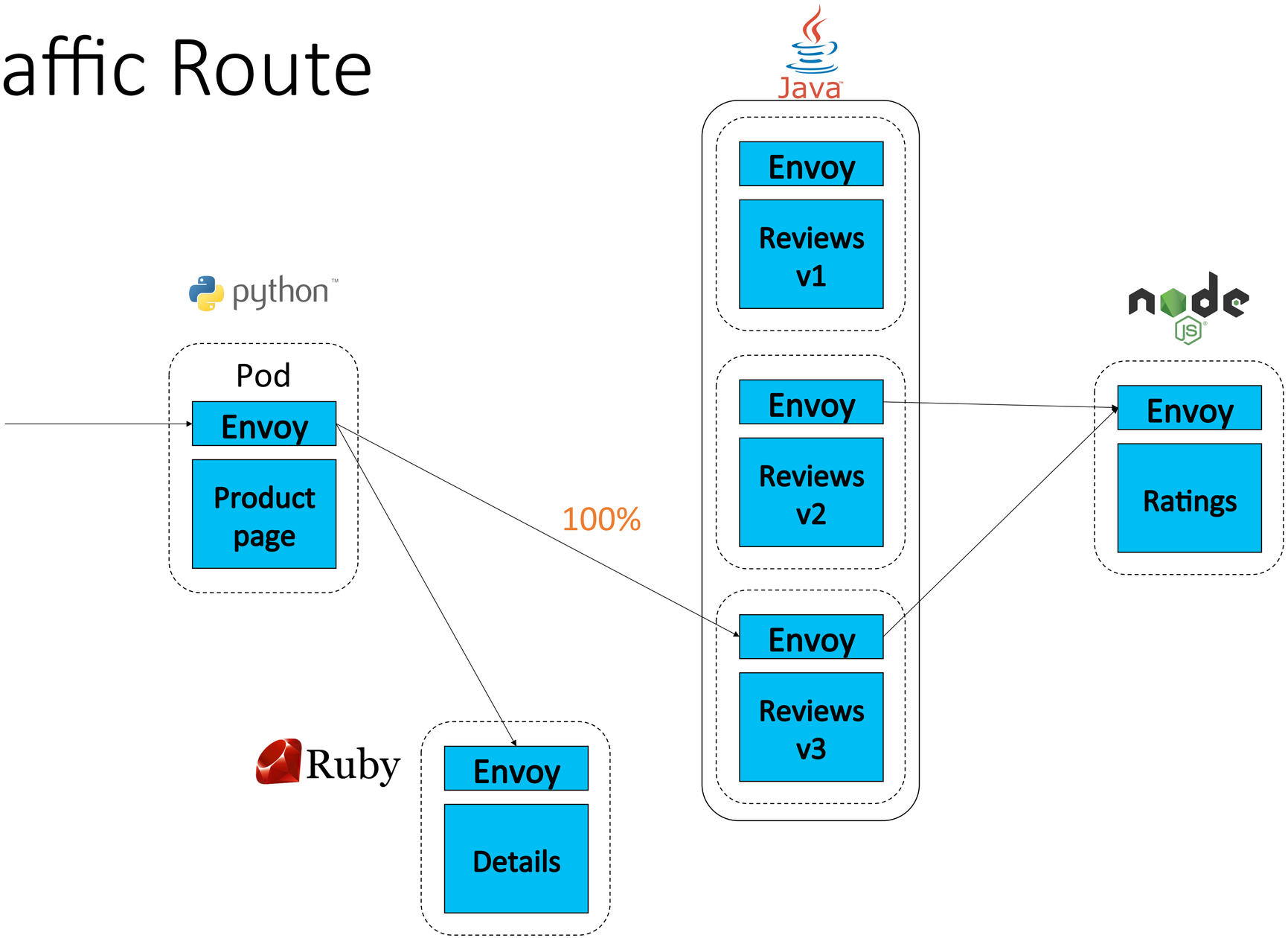
Traffic Route



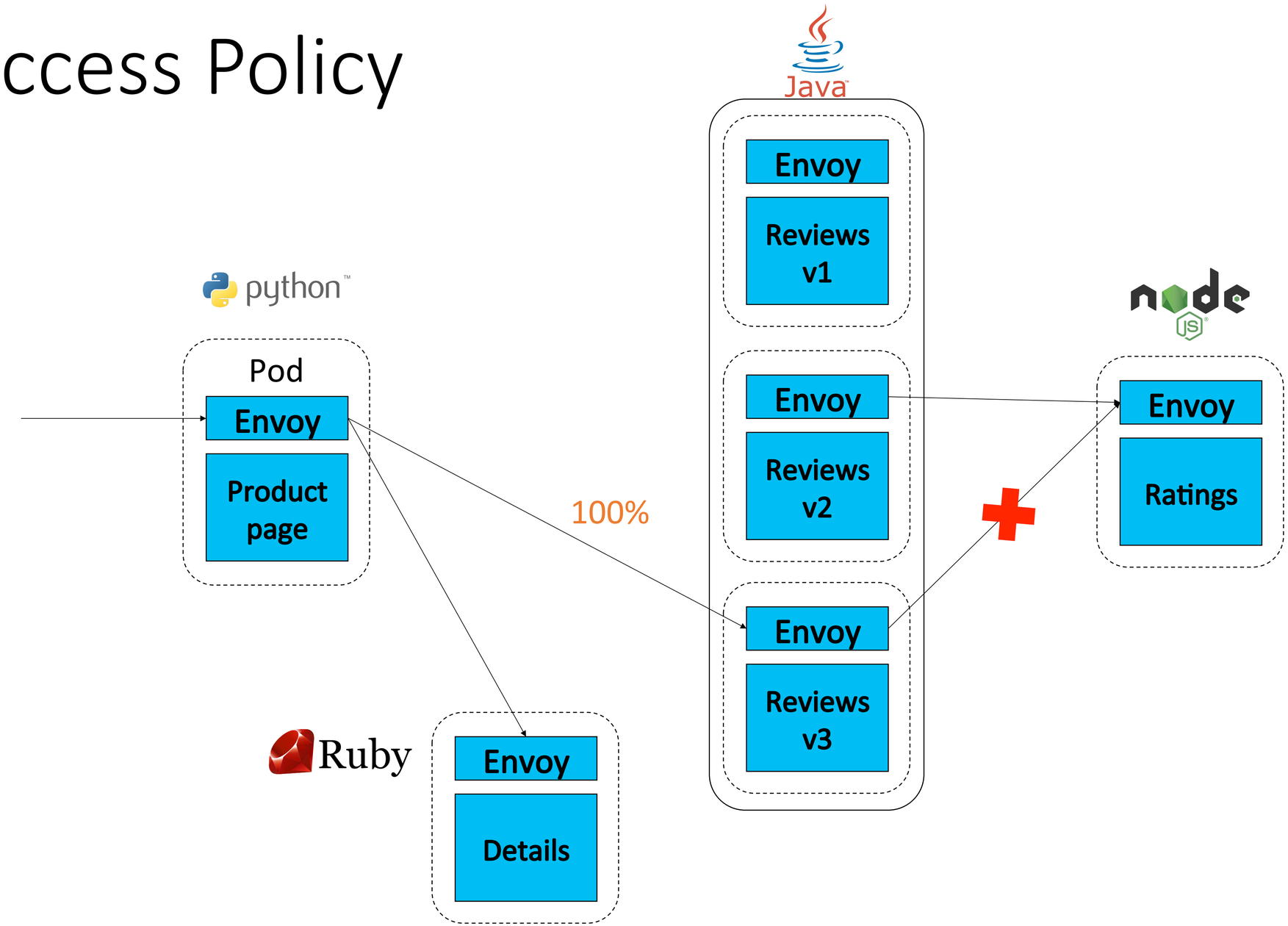
Traffic Route



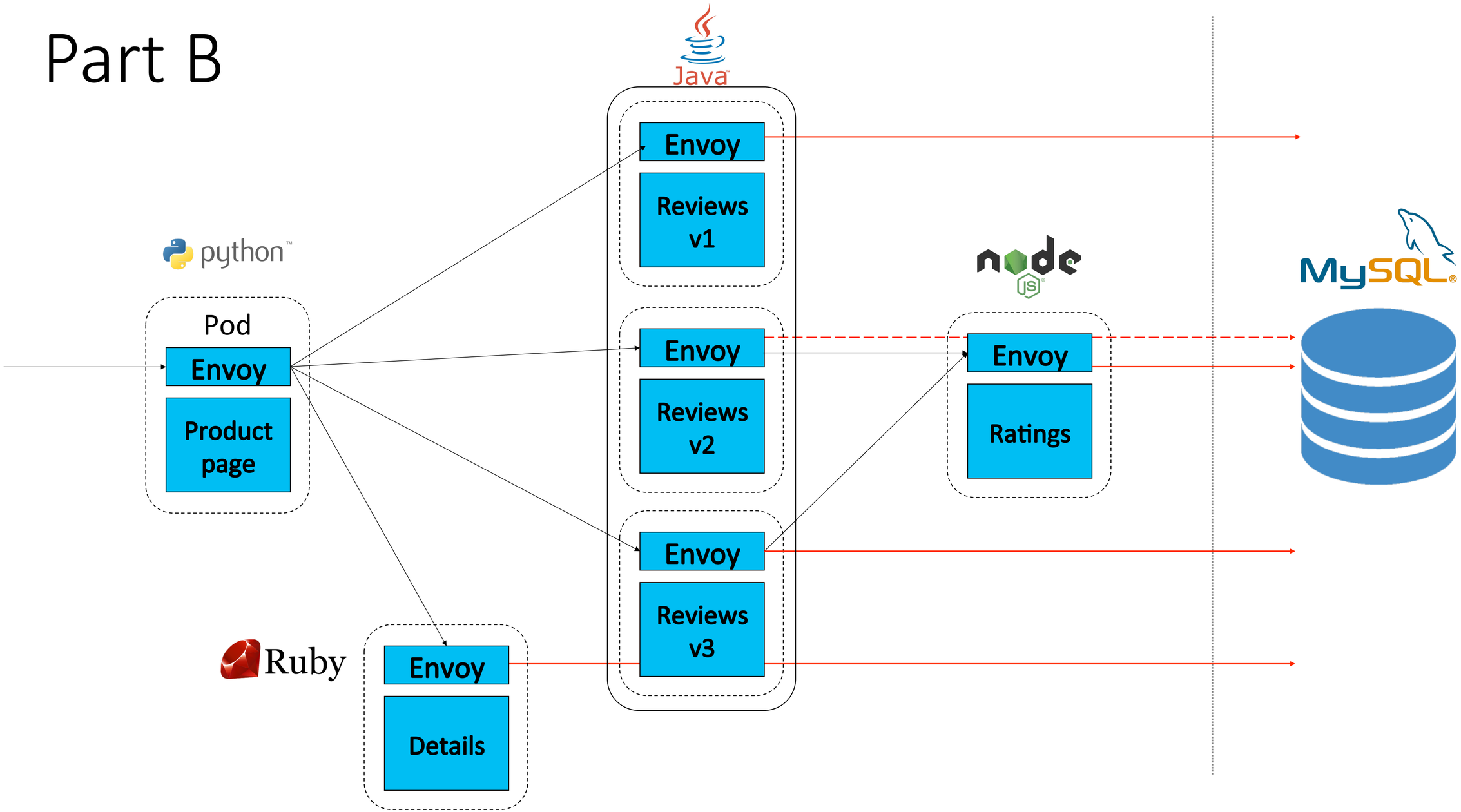
Traffic Route



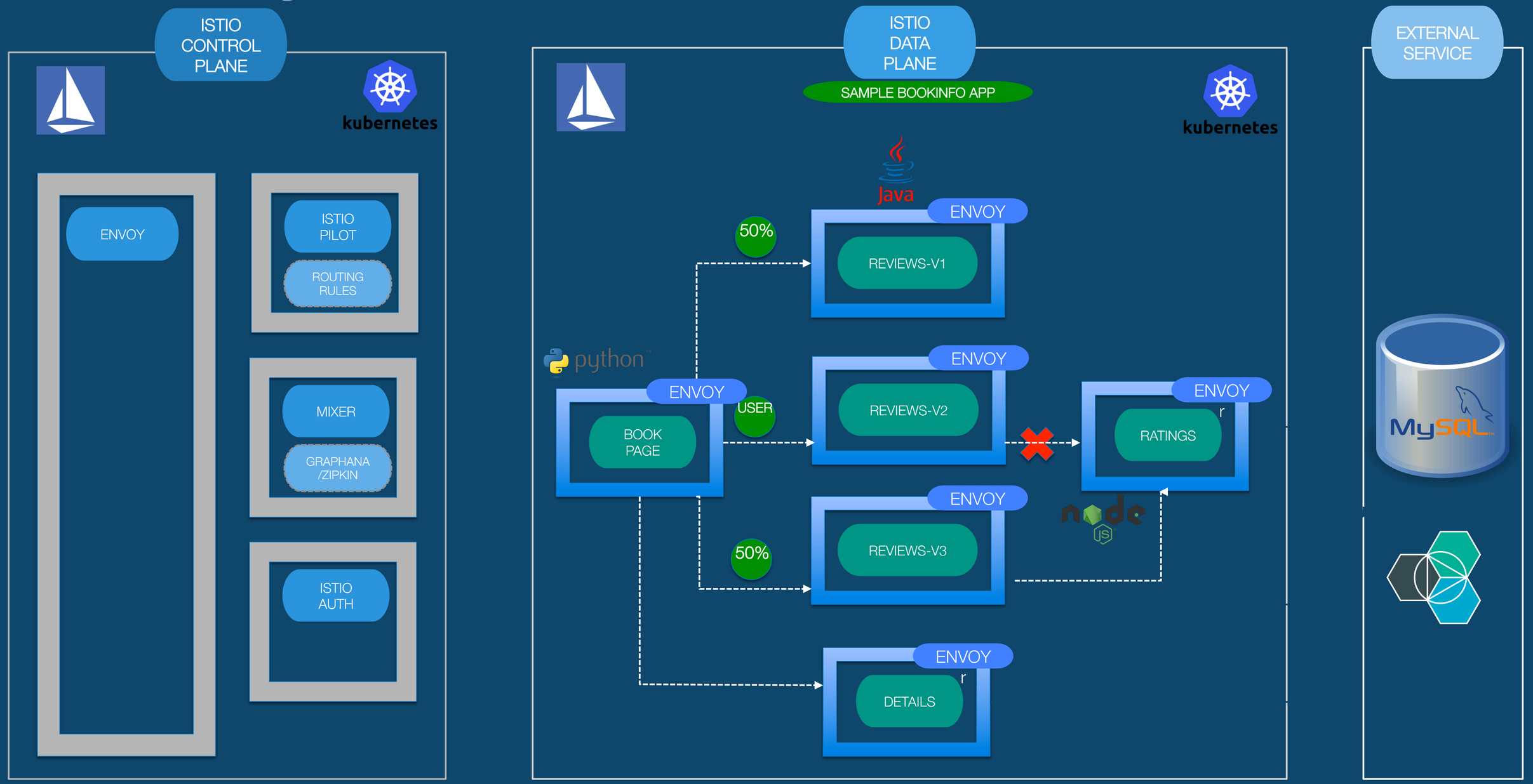
Access Policy



Part B


Java
Envoy
Reviews v1
Envoy
Reviews v2
Envoy
Reviews v3
python
Pod
Envoy
Product page
node
Envoy
Ratings
MySQL
Ruby
Envoy
Details

Thank you!


ISTIO
CONTROL
PLANE
kubernetes
ENVOY
ISTIO
PILOT
ROUTING
RULES
MIXER
GRAPHANA
/ZIPKIN
ISTIO
AUTH
ISTIO
DATA
PLANE
SAMPLE BOOKINFO APP
kubernetes
Java
50%
ENVOY
REVIEWS-V1
python
ENVOY
BOOK
PAGE
USER
ENVOY
REVIEWS-V2
ENVOY
RATINGS
ENVOY
REVIEWS-V3
50%
node
JS
ENVOY
DETAILS
EXTERNAL
SERVICE
MySQL