Introduction to Istio

- Avadhut (kodtodya.github.io)

$ whoami

- I'm Avadhut!
- Open-source enthusiast & Ex-redhatter ;)
- Active community member for couple of open-source projects
- Working as a Senior Consultant, Integration Lead & Architect for different clients
- Worked on Middleware Integration using Fuse, Camel, Karaf, Kafka and messaging platforms for quite a bit
- Did full production deployments, architecture review and performance tuning for couple of employers and lot of Red Hat customers

** You can find me on:

https://kodtodya.github.io

Pre-Requisite for Istio Course

- Ability to use command line
- Mandatory knowledge of Docker, Kubernetes/Openshift
- Knowledge of Spring framework and Spring Boot is mandatory
- Linux(Any flavor) and Mac are strongly preferred, avoid Windows if possible (I am going to demonstrate this session on linux; if windows runs into problems, I may not deal with it)
- Knowledge of Webservice, JMS and database is also required
- Willingness to learn an awesome technology... ☺

Who is this course for?

- **Developer**: who would like to learn how to write and run an application that uses Istio as service mesh

- **Architects**: who want to understand the role of Istio in the enterprise integration with/using micro-services

- **DevOps**: who want to understand how Istio works on container orachastration plaforms and its infrastructural setup

Exceptions

What I am not going to cover

- Openshift clustering
- Your UAT and production setup and issues
- Istio/Service Mesh Migrations to latest version
- Your enterprise application issues
- CI/CD jobs creation using Jenkins & nexus setup
- Hotfixes and patches to any environment
- Your enterprise project architecture & migration consultation
- Third party license software & tool usage or consultation about solution with it.

If you have any of the above use-case,

Please reach out to me for best pricing of middleware solutions at earliest:

https://kodtodya.github.io/talks

Reference & Special thanks to

Burr Sutter

Istio on Kubernetes Sessions

This entire presentation is copied from his publicly available presentation with small cosmetic changes.

A very warm welcome to

Introduction to Istio

course..

Course Structure

- **Part – 1 : Introduction to Istio**
  - Why Service Mesh
  - Observability
  - Istio Architecture & Introduction
  - Traffic Control
  - Service Resiliency & Circuit Breaking
  - Chaos Testing
  - Egress
  - Security
- **Part – 2 : Istio in Action**
  - Requirements
  - Setup & Scripting
  - Micro-Services Deployment
  - Observability
  - Metrices
  - Chaos

Exercise Setup

CodeReady Containers
https://github.com/kodtodya/fuse-7-istio-examples

Testing/Demo/CodeReady Container & CRC Scripts
https://github.com/kodtodya/fuse-7-istio-examples/tree/main/greeting-service-istio-example

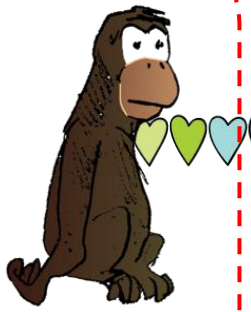
Your Journey to Awesomeness

Re-Org to DevOps

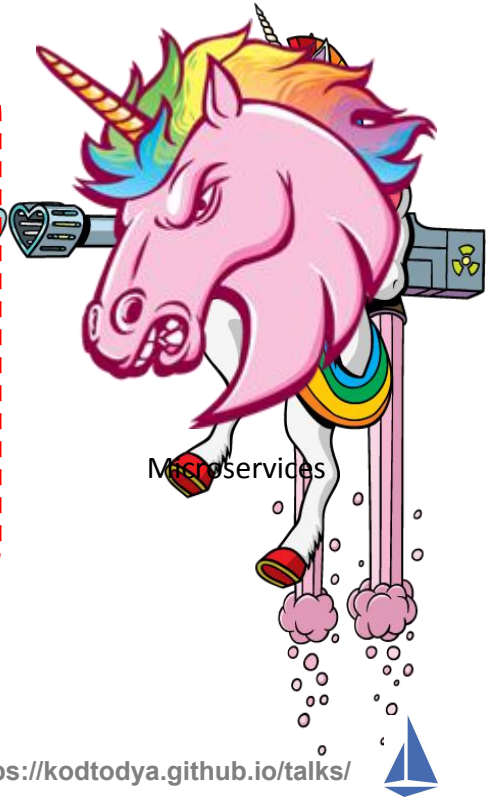
Self-Service, On-Demand, Elastic Infrastructure

Automation

CI & CD Deployment Pipeline



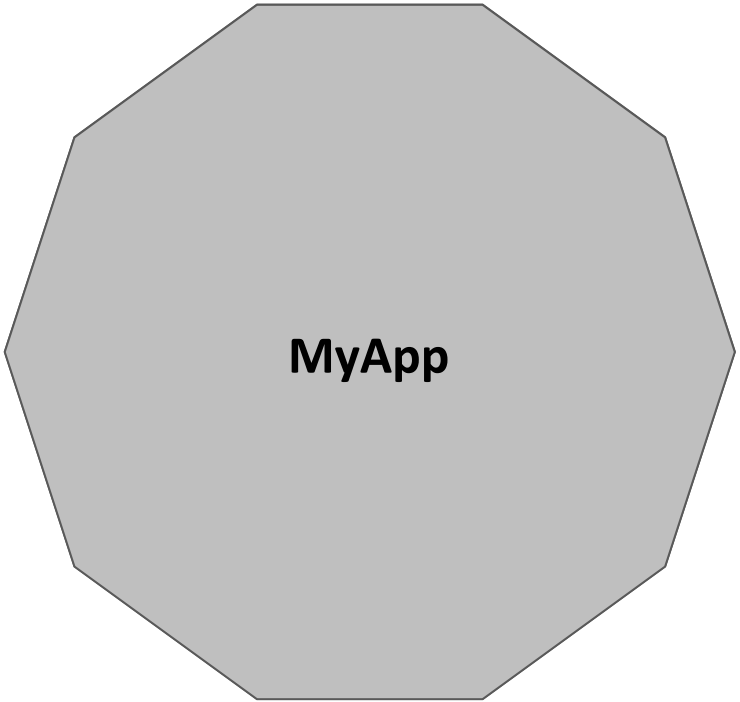
Advanced Deployment Techniques



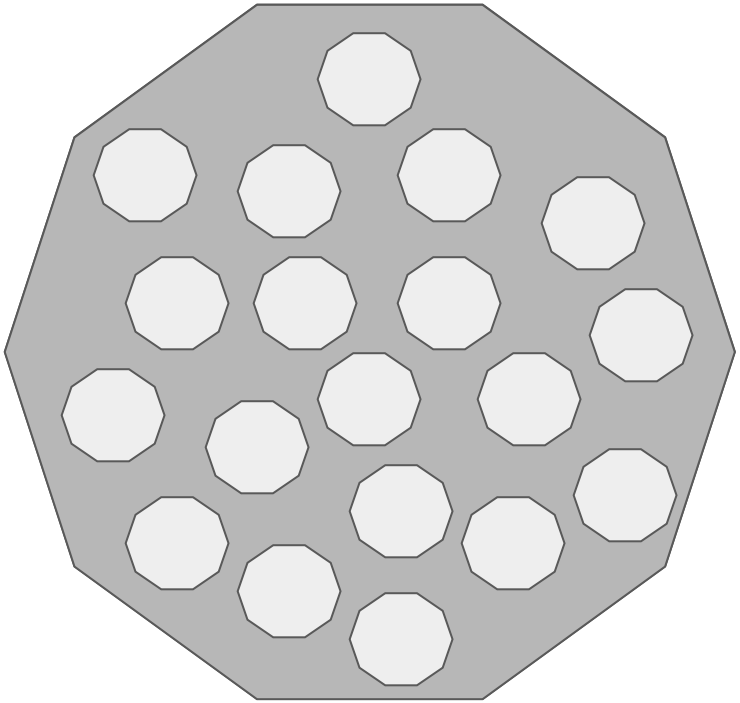
Microservices

https://kodtodya.github.io/talks/

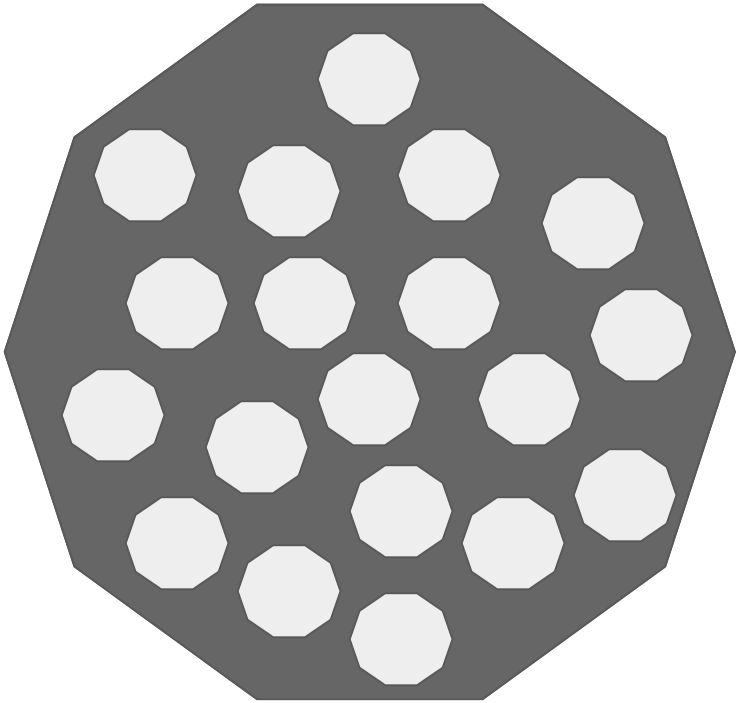
Monolith



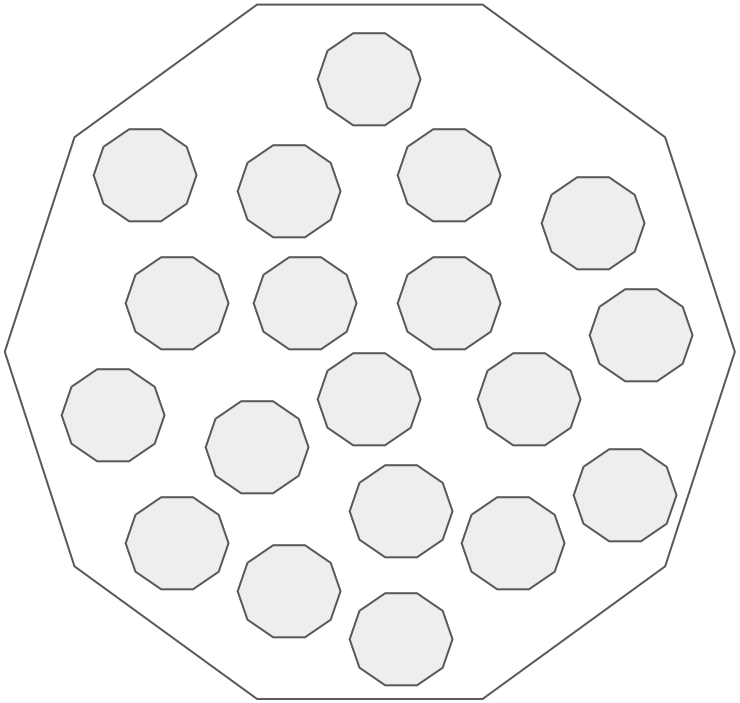
The Application



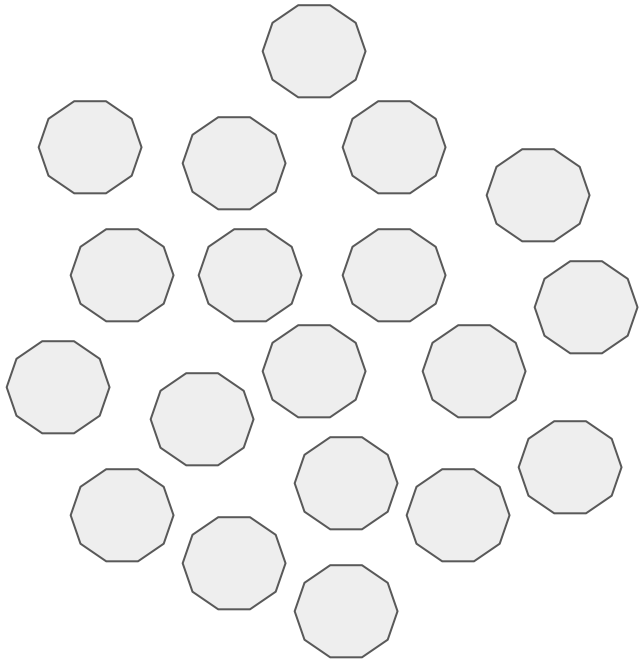
Modules



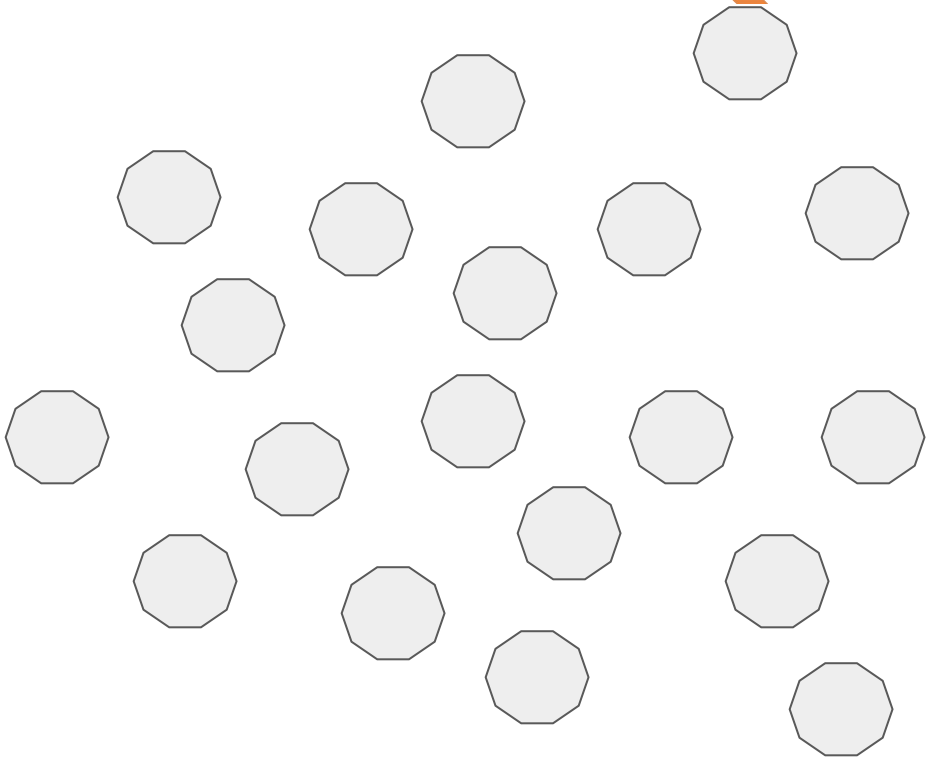
Microservices



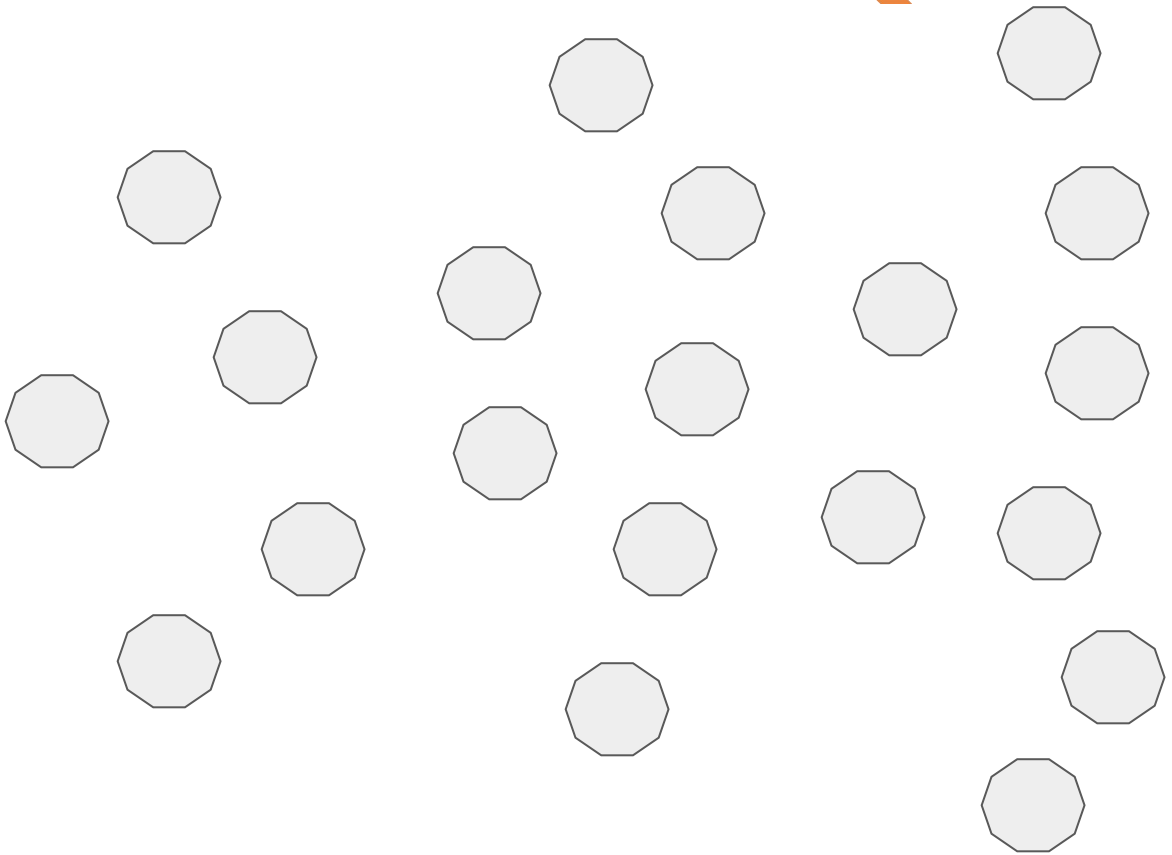
Microservices



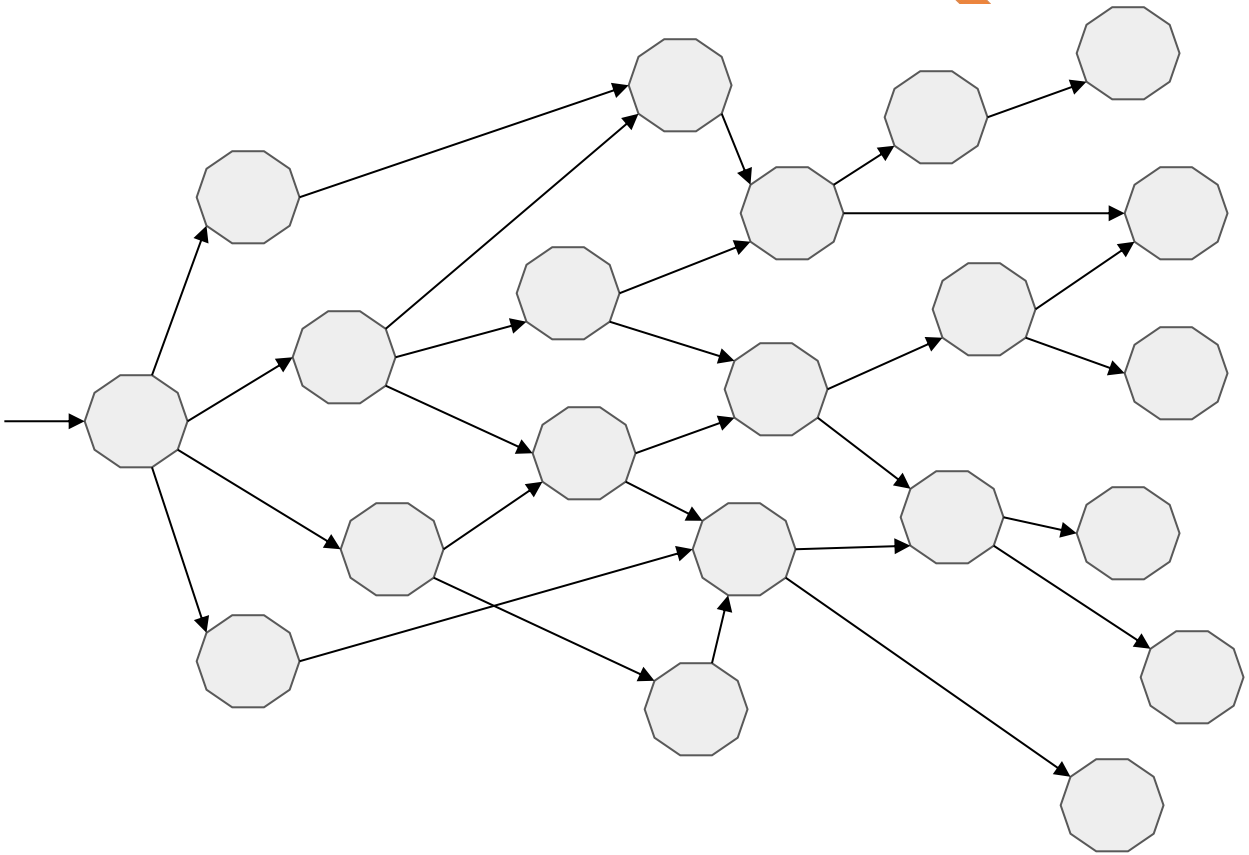
Microservices



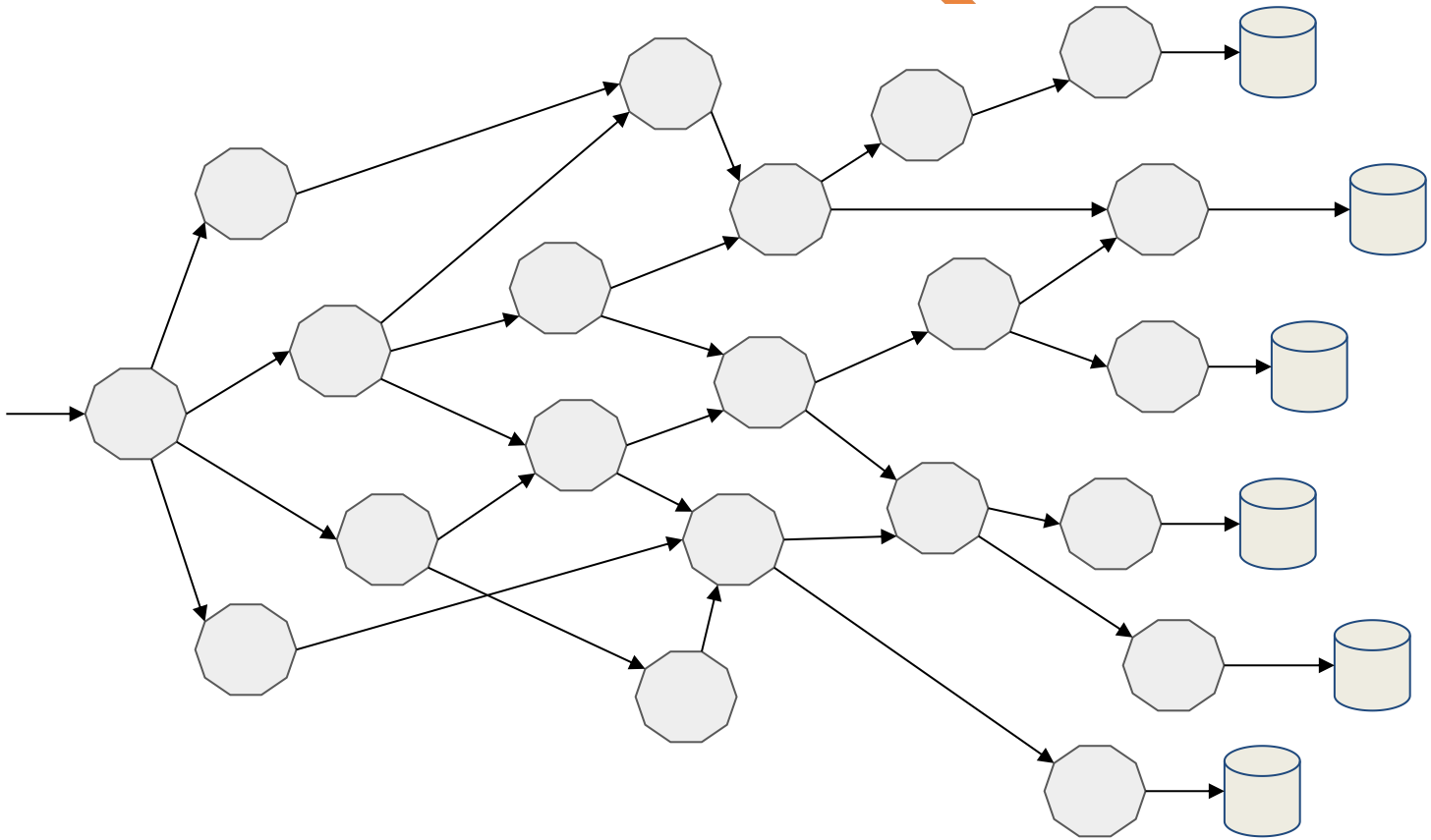
Microservices



Network of Services

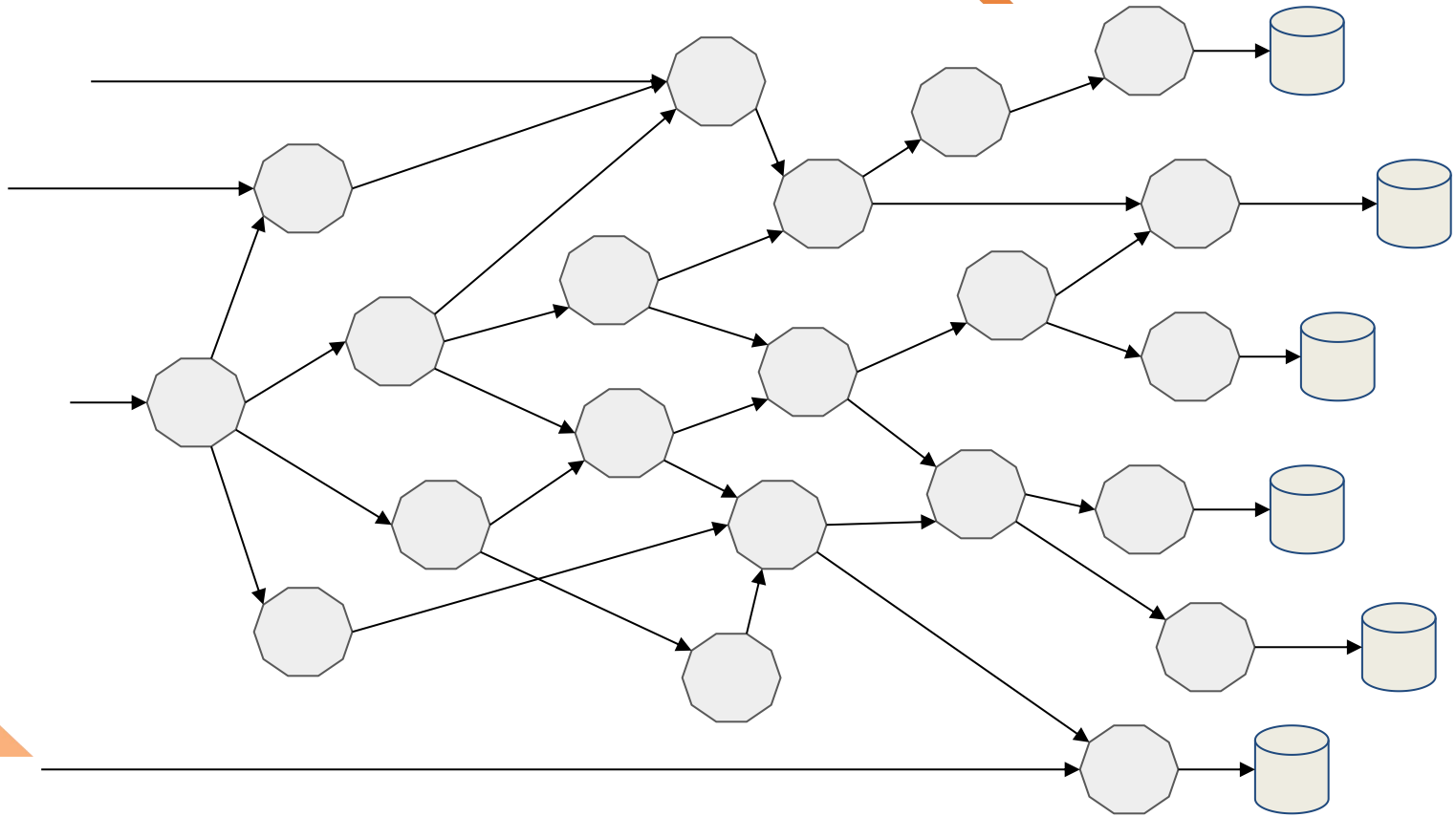


Microservices own their Data

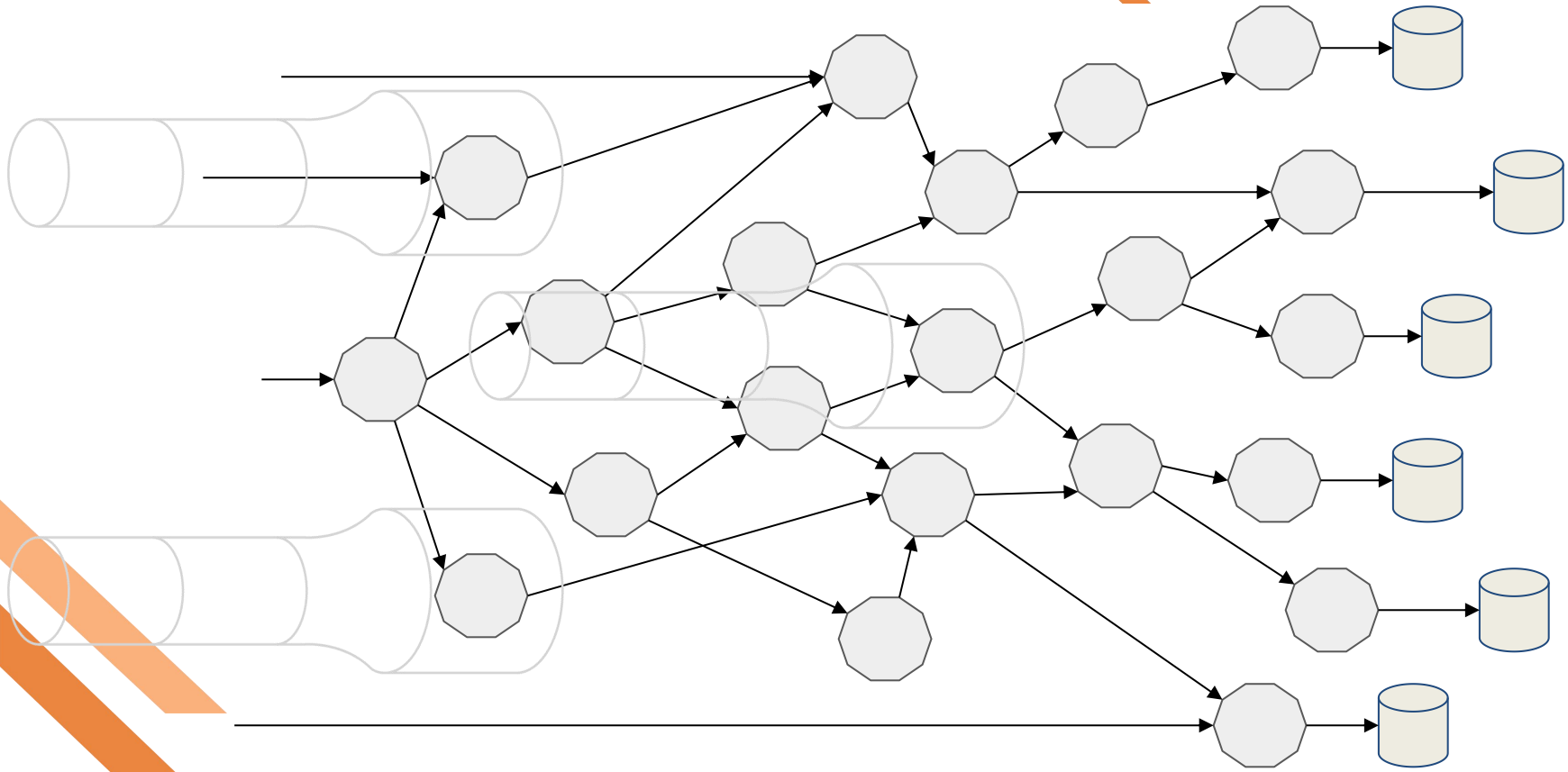


https://kodtodya.github.io/talks/

Multiple Points of Entry

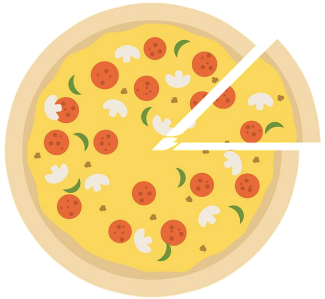


Multiple Teams, Multiple Pipelines



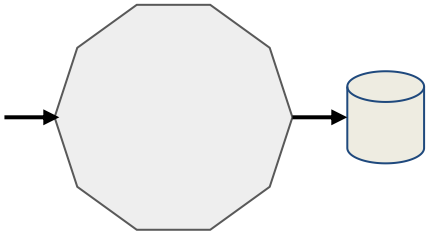
Microservices Principles

1. Deployment **Independence** - updates to an individual microservice have no negative impact to any other component of the system. Optimized for **Replacement**
2. Organized around **business** capabilities
3. **Products** not Projects
4. **API** Focused
5. **Smart** endpoints and dumb pipes
6. Decentralized Governance
7. Decentralized Data Management
8. Infrastructure Automation (infrastructure as code)
9. Design for failure
10. Evolutionary Design

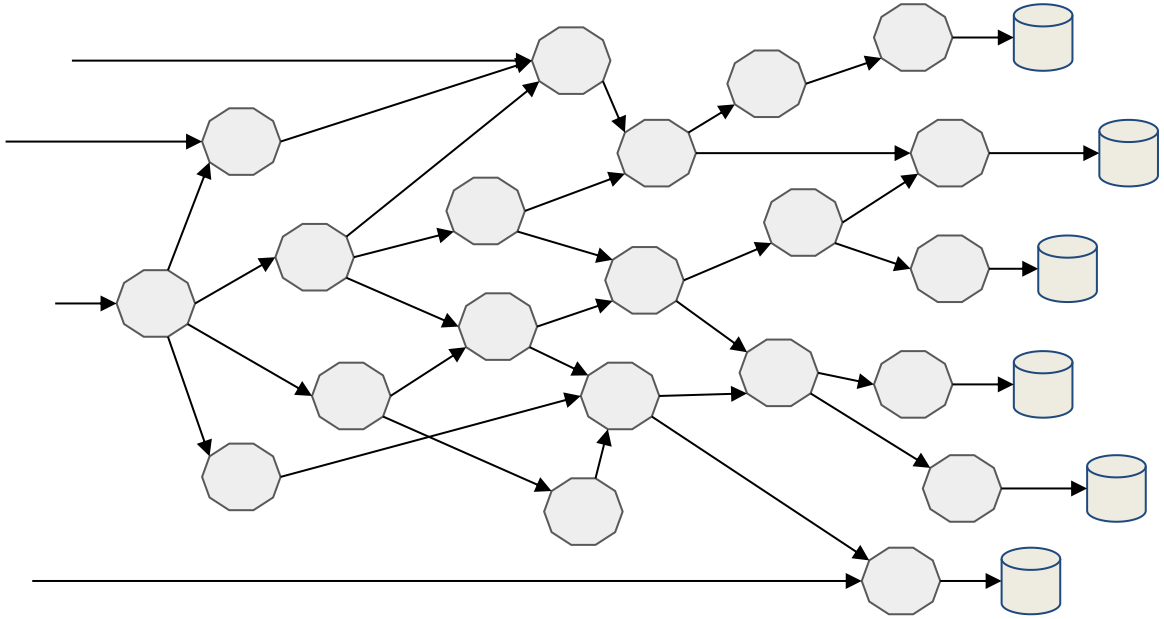


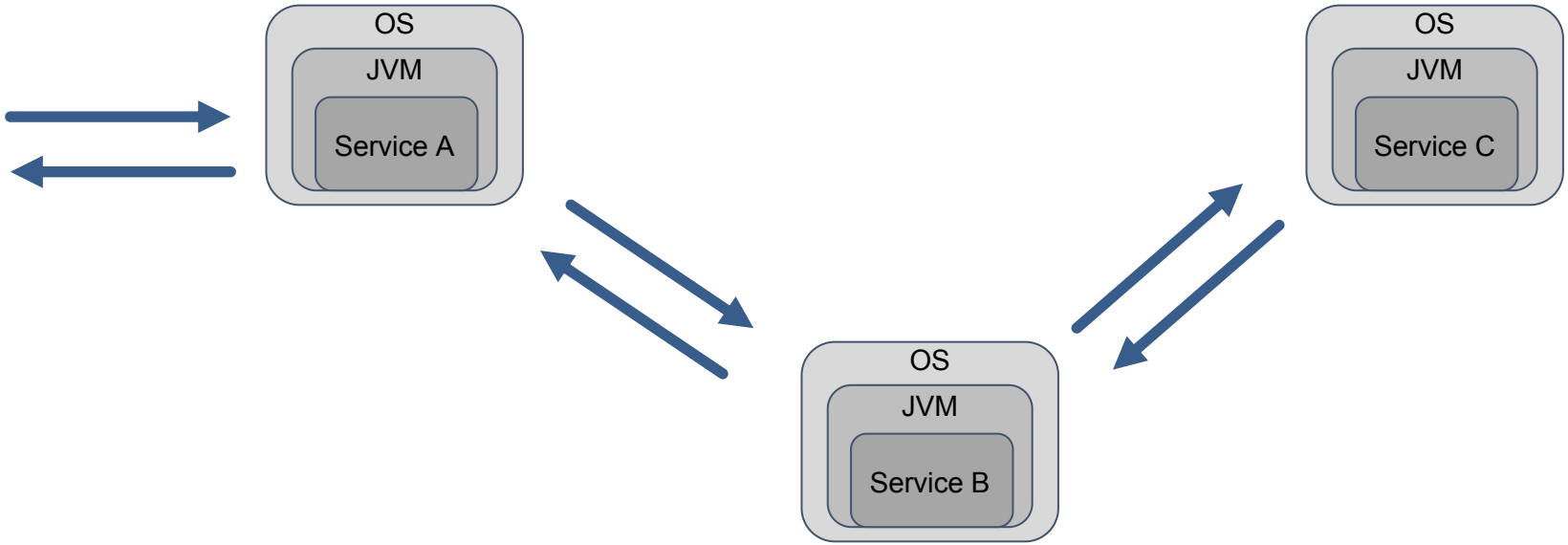
2 Pizza Team

Old vs New School



Love Thy Mono





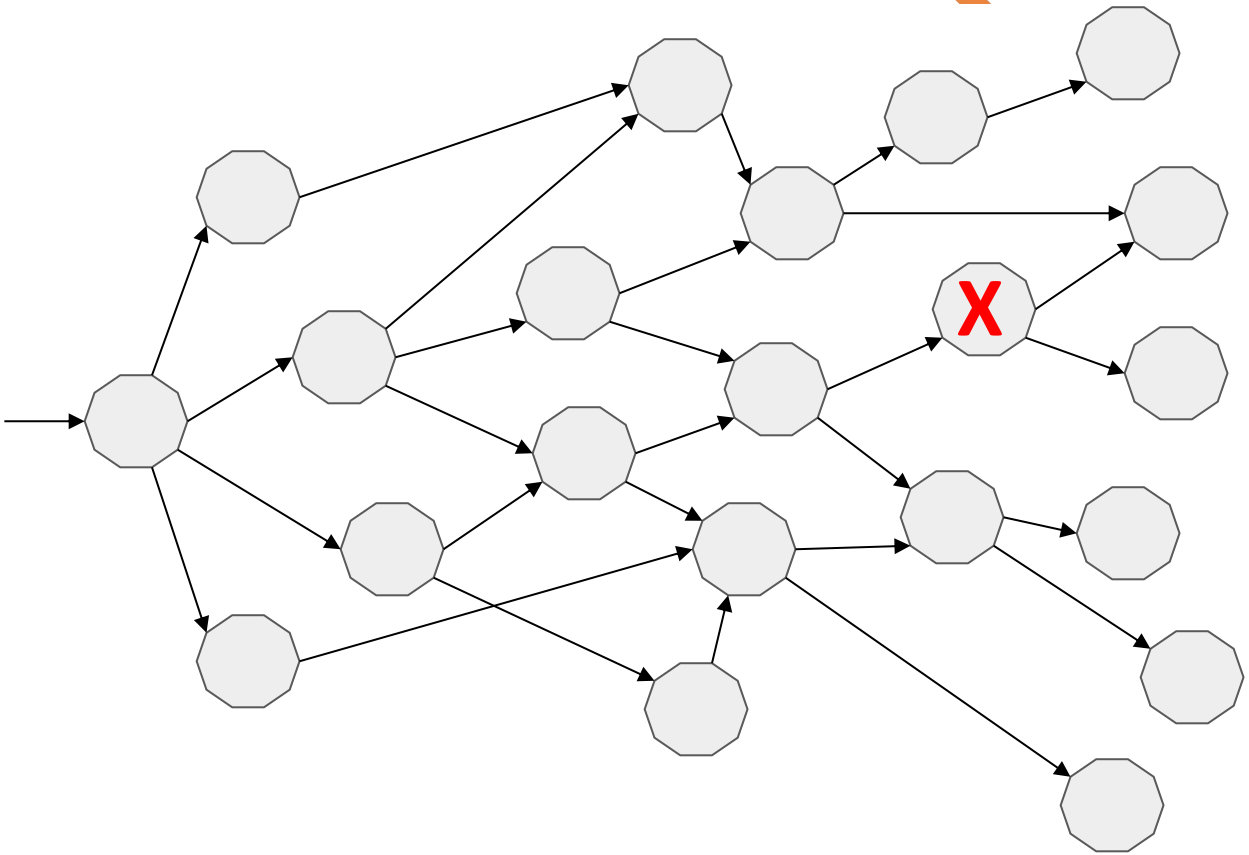
Microservices == Distributed Computing

Fallacies of Distributed Computing

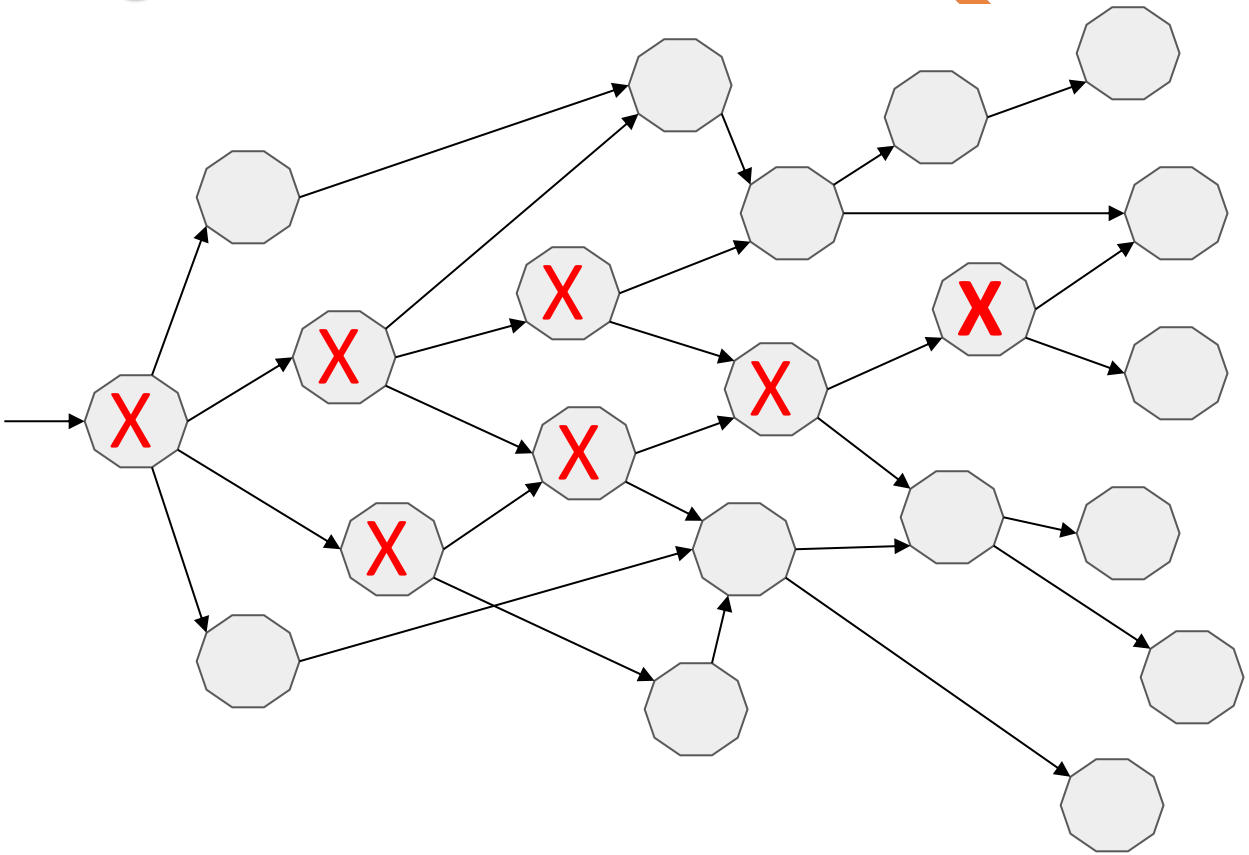
- The Network is Reliable
- Latency is zero
- Bandwidth is infinite
- Topology does not change
- There is one administrator
- Transport cost is zero
- The network is homogeneous

Ref : https://en.wikipedia.org/wiki/Fallacies_of_distributed_computing

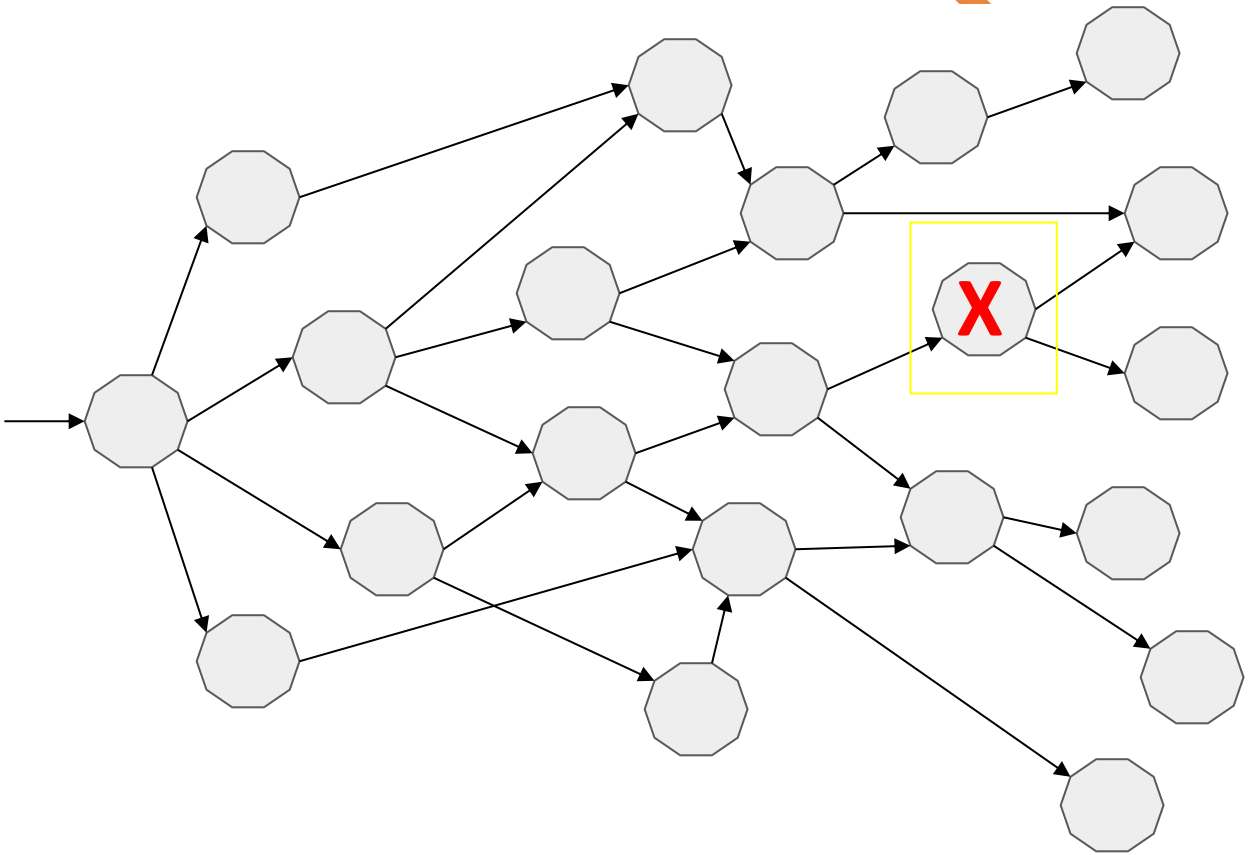
Failure of a Service



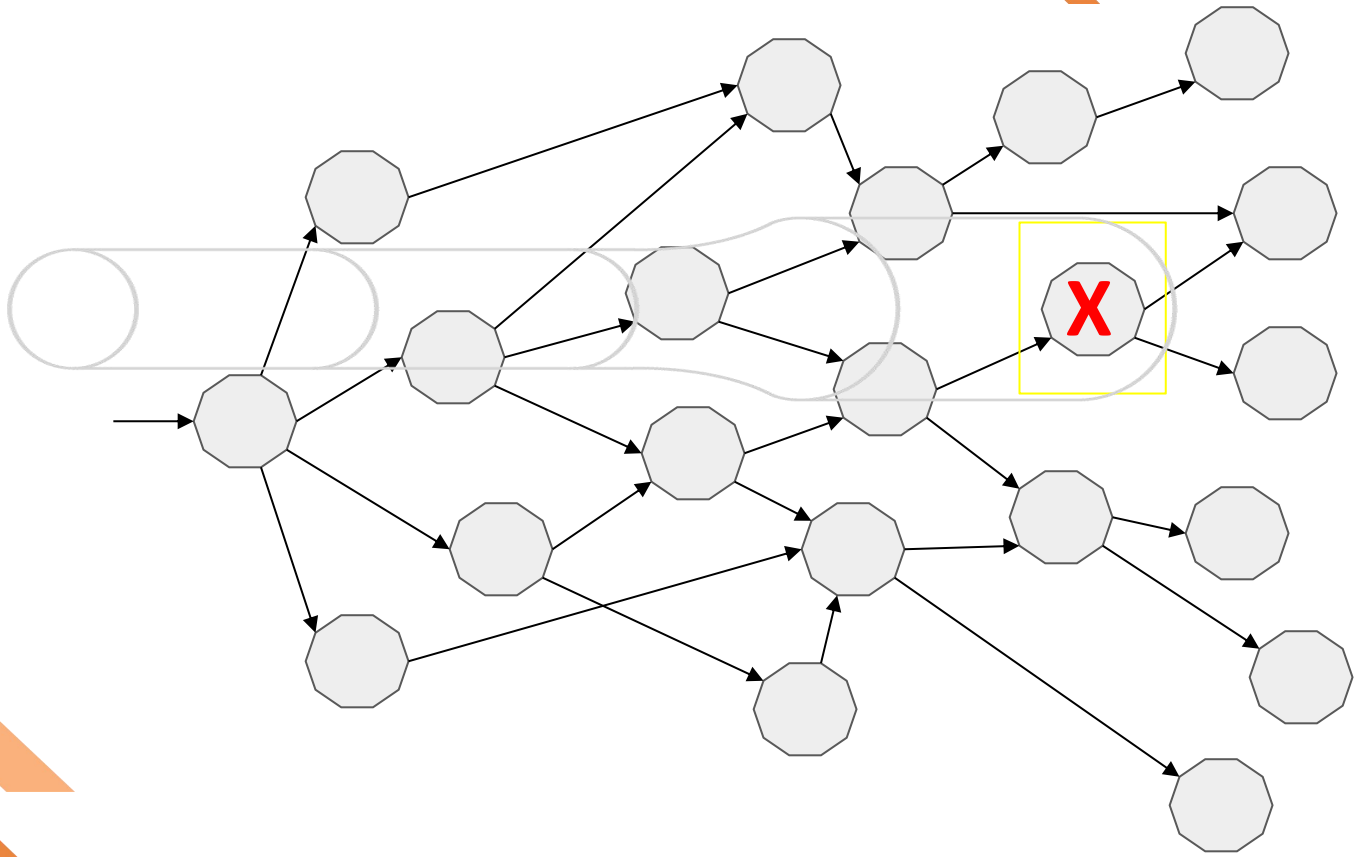
Cascading Failure



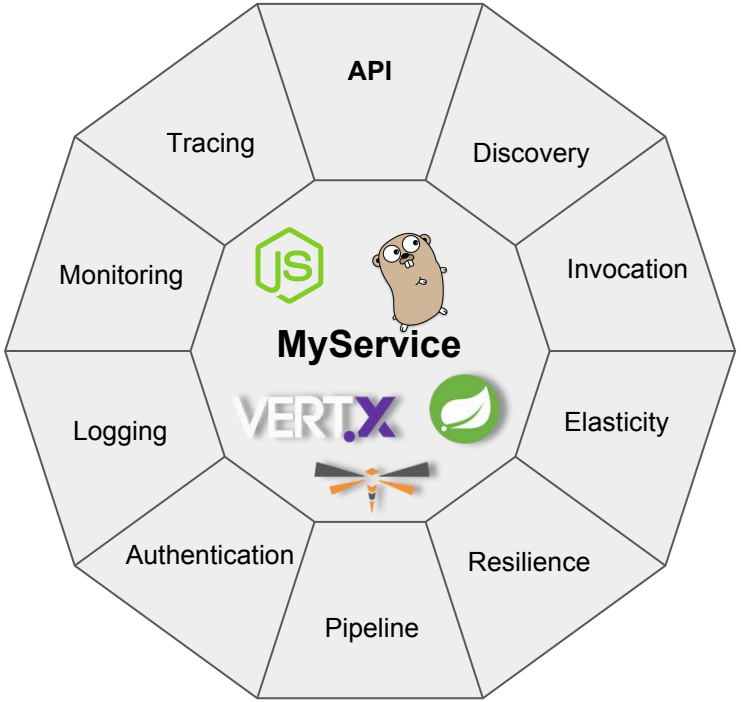
Isolate Failure



A Temporary Glitch

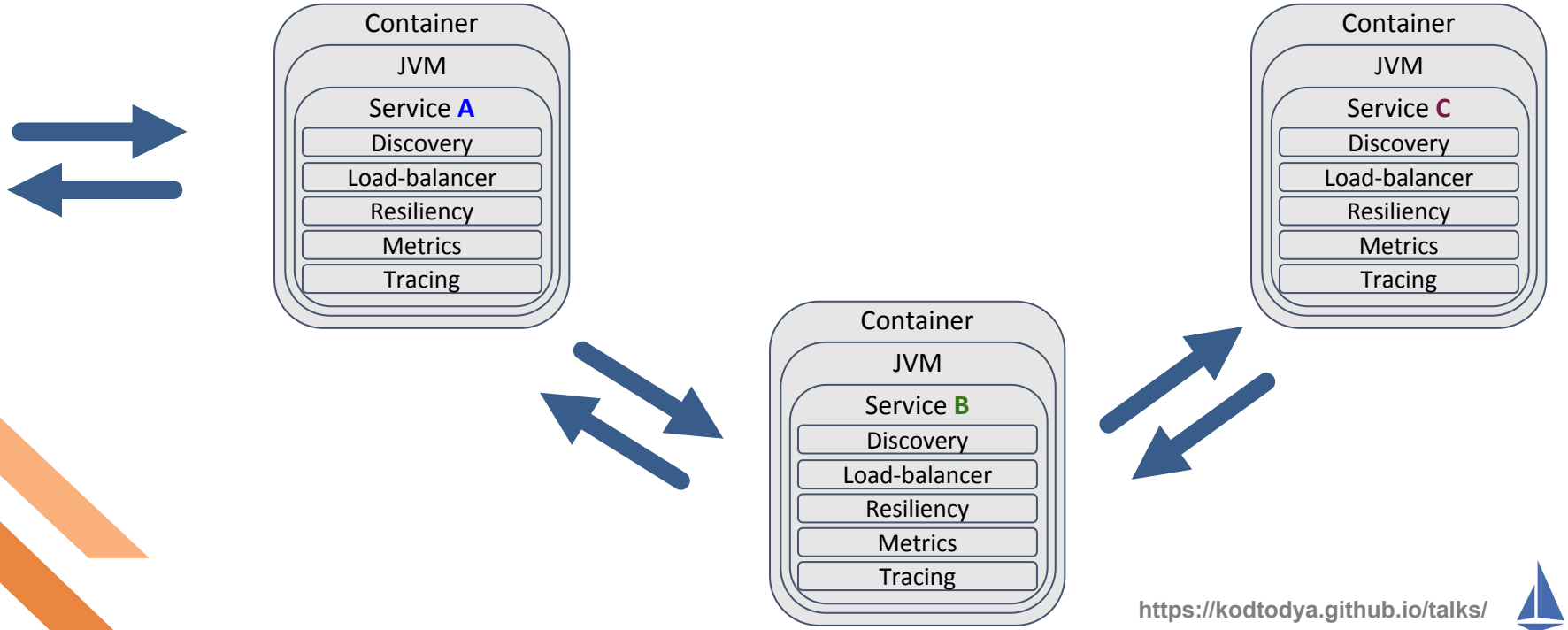


Microservices'ilities



History of Microservices

Microservices embedding Capabilities

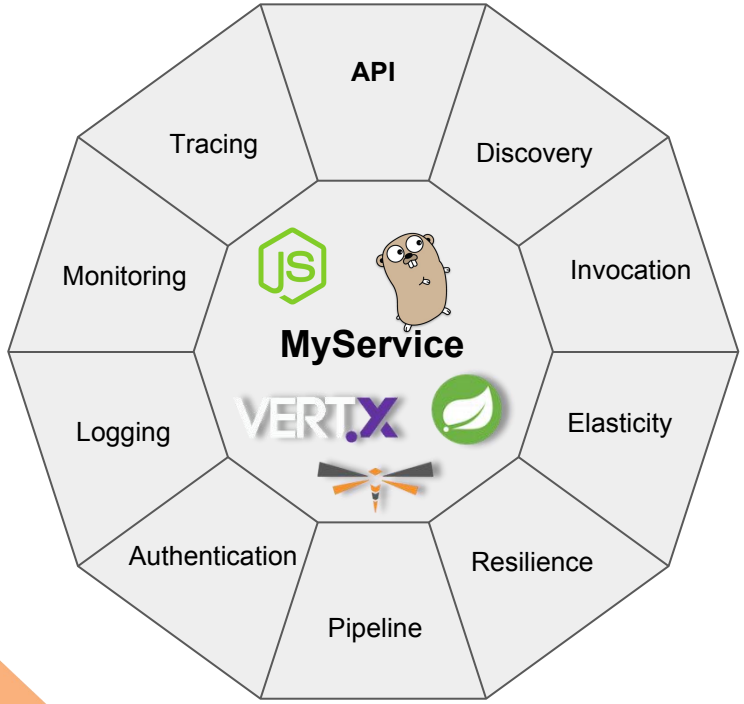


What's Wrong with Netflix OSS?

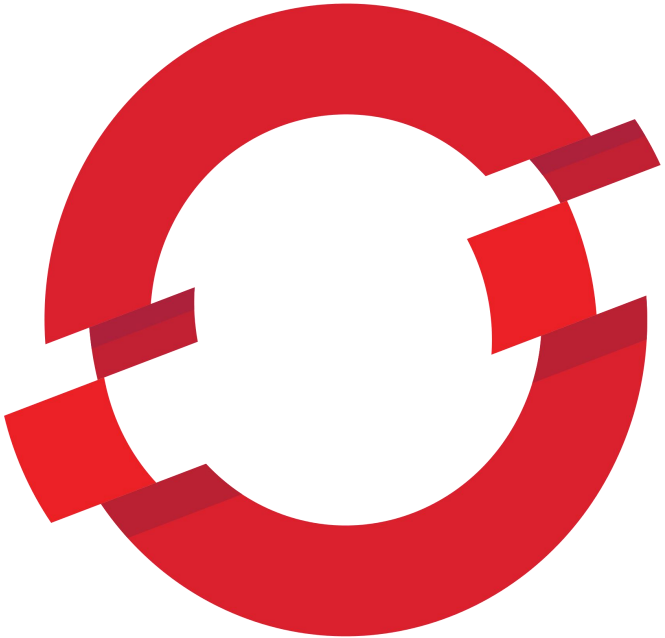
Java Only

Adds a lot of libraries to YOUR code

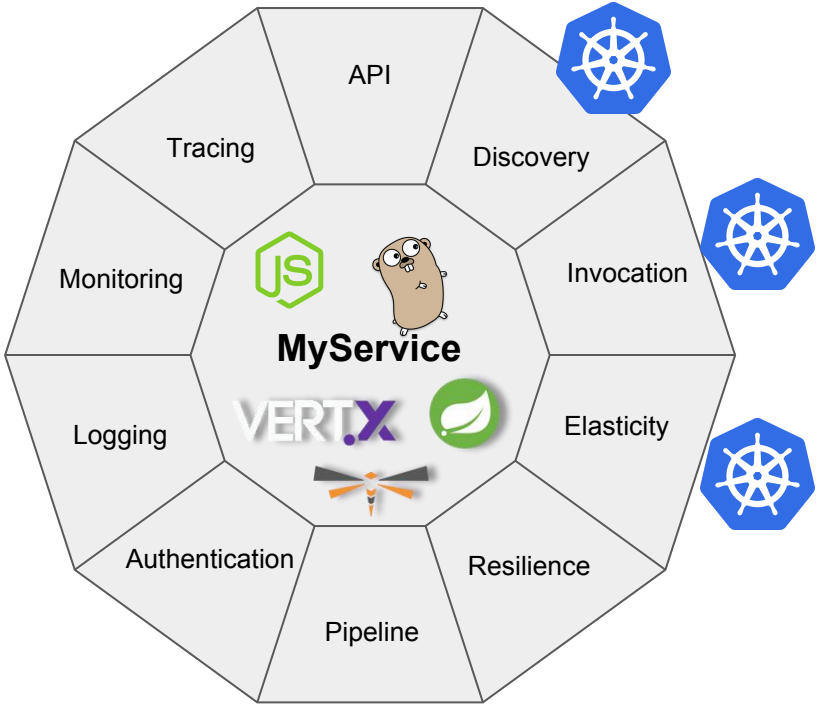
NETFLIX | OSS



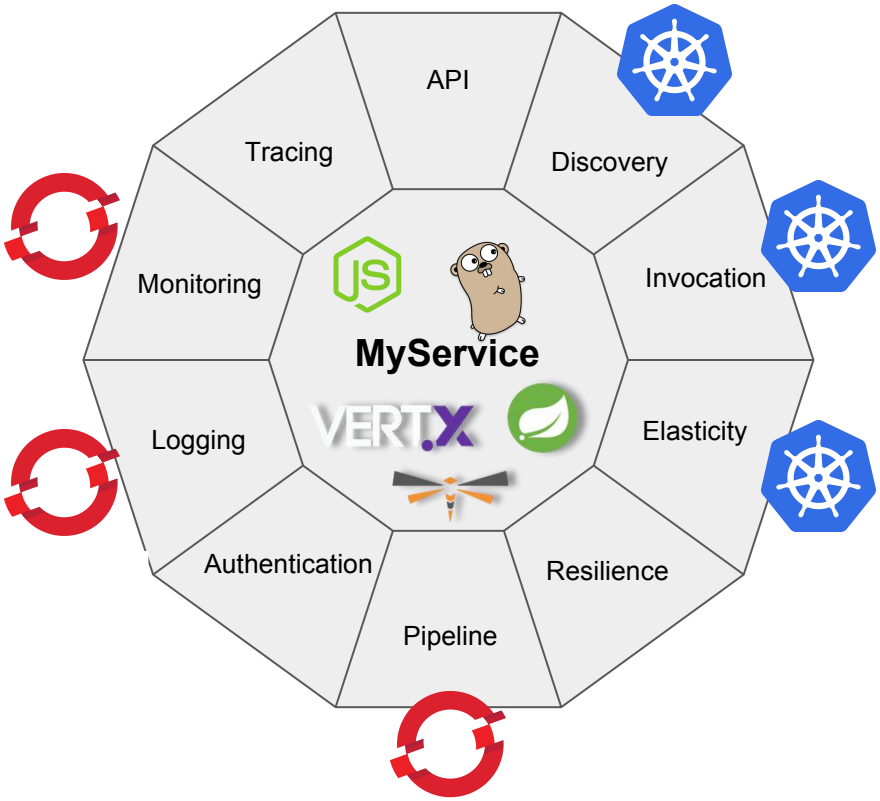
Microservices'ilities



Microservices'ilities + Kubernetes



Microservices'ilities + OpenShift



Kubernetes Re-cap

1. mvn package
2. docker build
3. docker push
4. kubectl apply -f deploy.yml

Istio - Sail

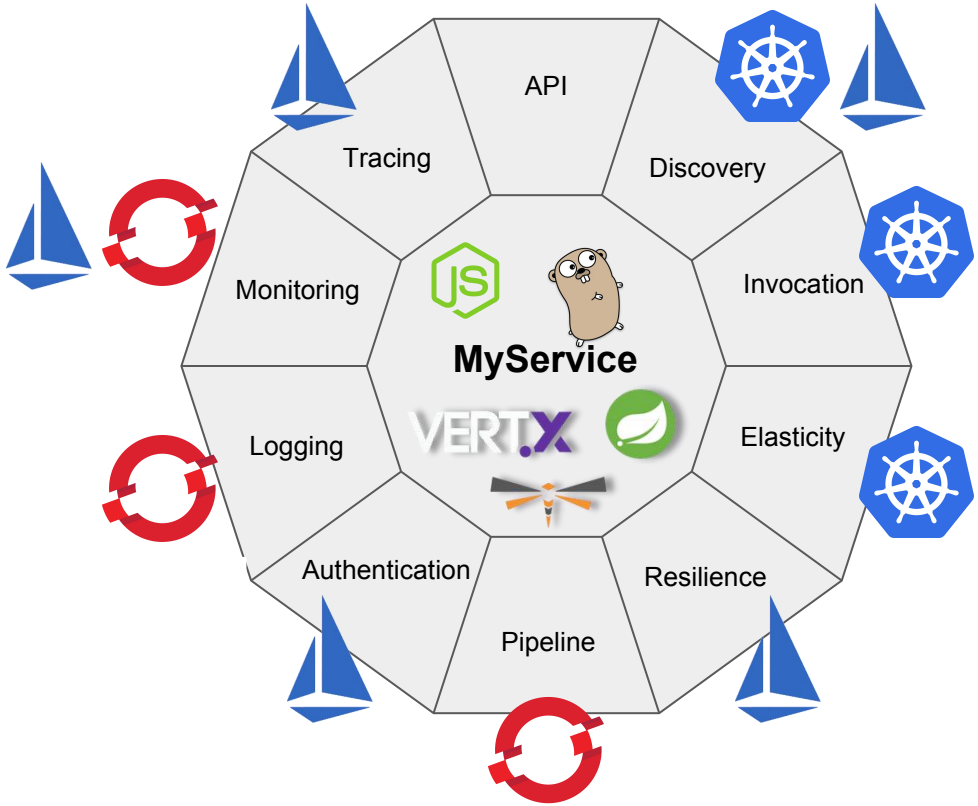
(Kubernetes - Helmsman or ship’s pilot)

Service Mesh Defined

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. In practice, the service mesh is typically implemented as an array of lightweight network proxies that are deployed alongside application code, without the application needing to be aware.

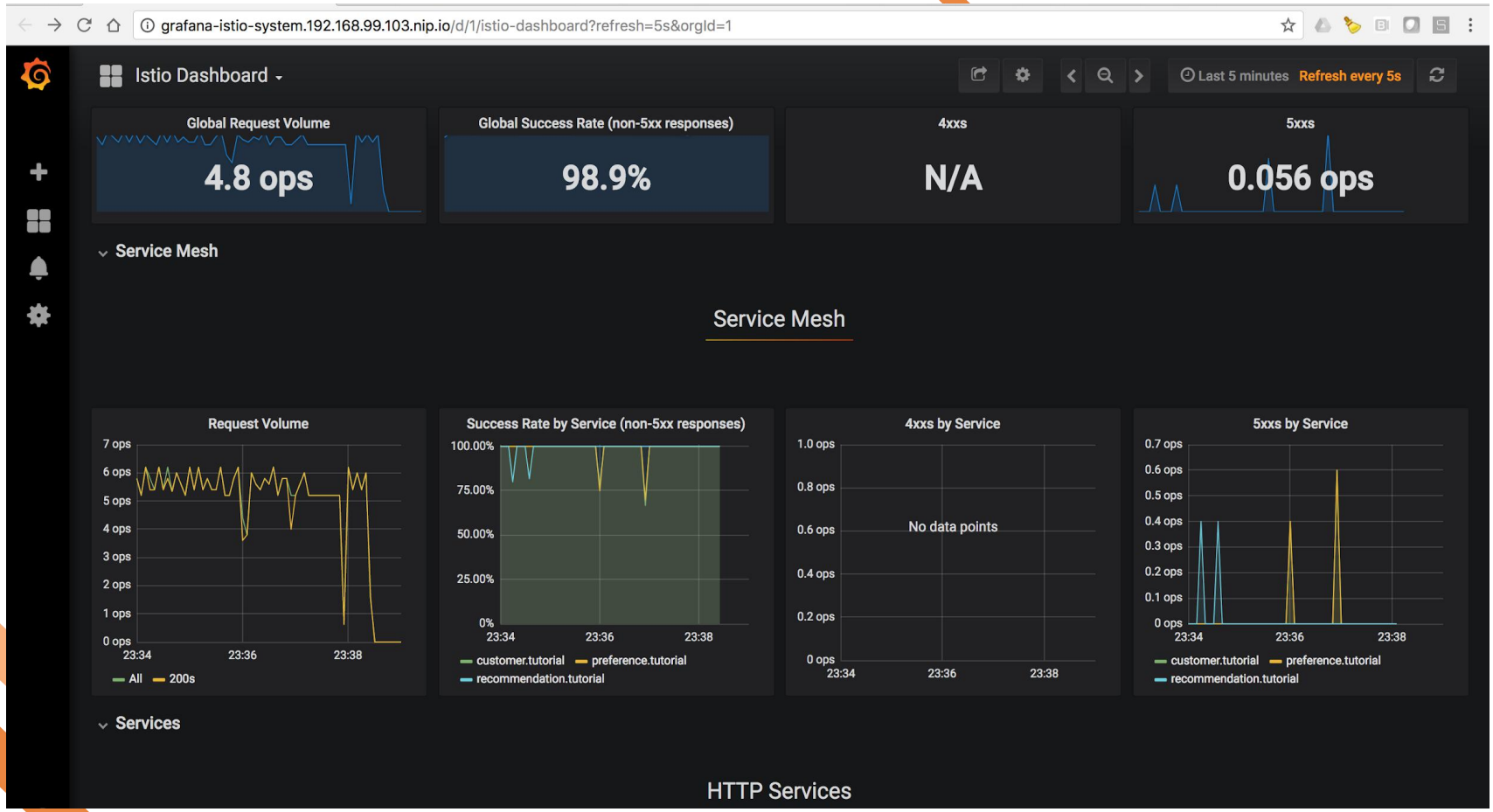
https://buoyant.io/2017/04/25/whats-a-service-mesh-and-why-do-i-need-one/

Microservices'ilities + Istio



Observability

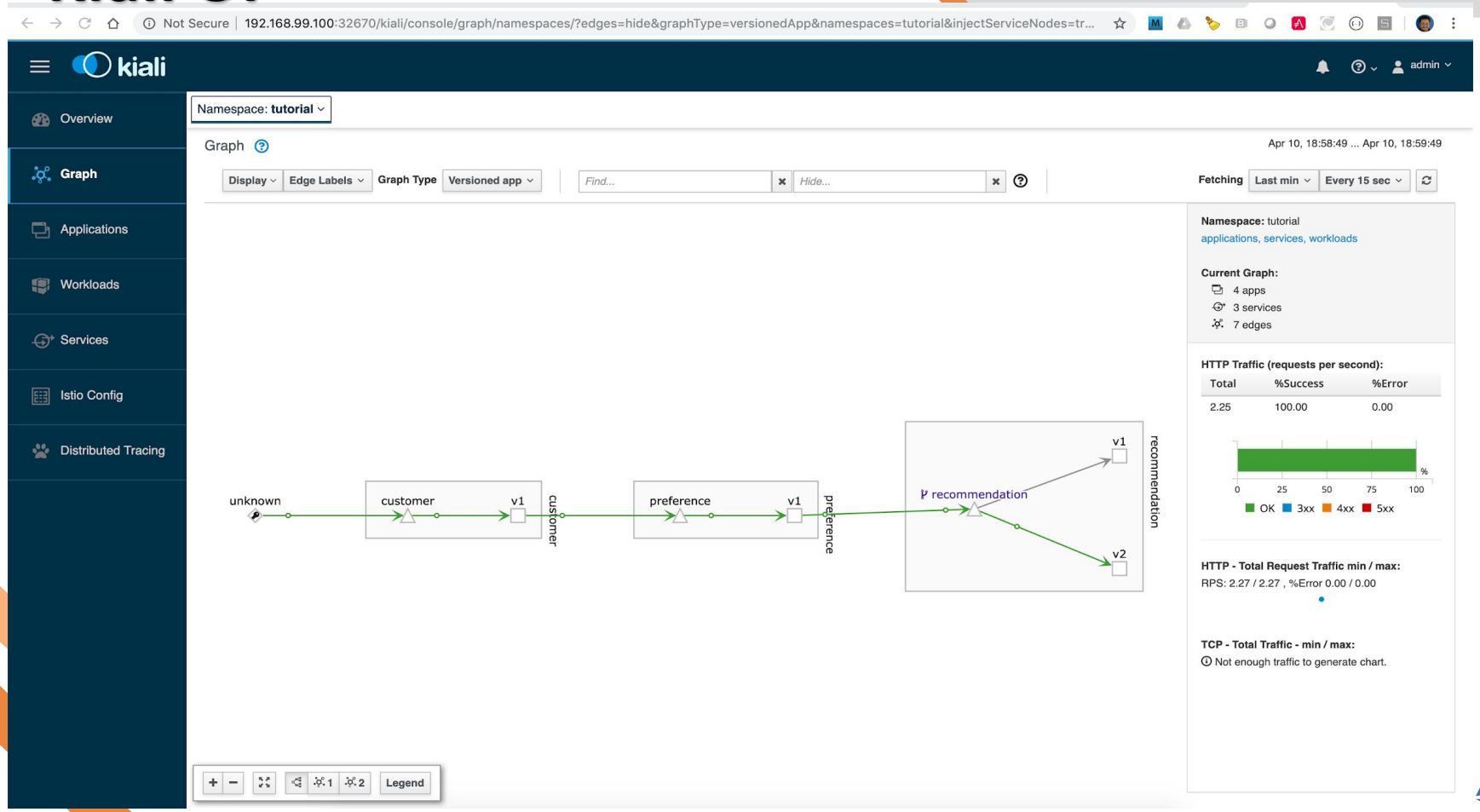
Grafana UI


grafana-istio-system.192.168.99.103.nip.io/d/1/istio-dashboard?refresh=5s&orgId=1
Istio Dashboard
Last 5 minutes
Refresh every 5s
Global Request Volume
4.8 ops
Global Success Rate (non-5xx responses)
98.9%
4xxs
N/A
5xxs
0.056 ops
Service Mesh
Service Mesh
Request Volume
7 ops
6 ops
5 ops
4 ops
3 ops
2 ops
1 ops
0 ops
23:34
23:36
23:38
All
200s
Success Rate by Service (non-5xx responses)
100.00%
75.00%
50.00%
25.00%
0%
customer.tutorial
preference.tutorial
recommendation.tutorial
4xxs by Service
1.0 ops
0.8 ops
0.6 ops
0.4 ops
0.2 ops
0 ops
No data points
5xxs by Service
0.7 ops
0.6 ops
0.5 ops
0.4 ops
0.3 ops
0.2 ops
0.1 ops
Services
HTTP Services

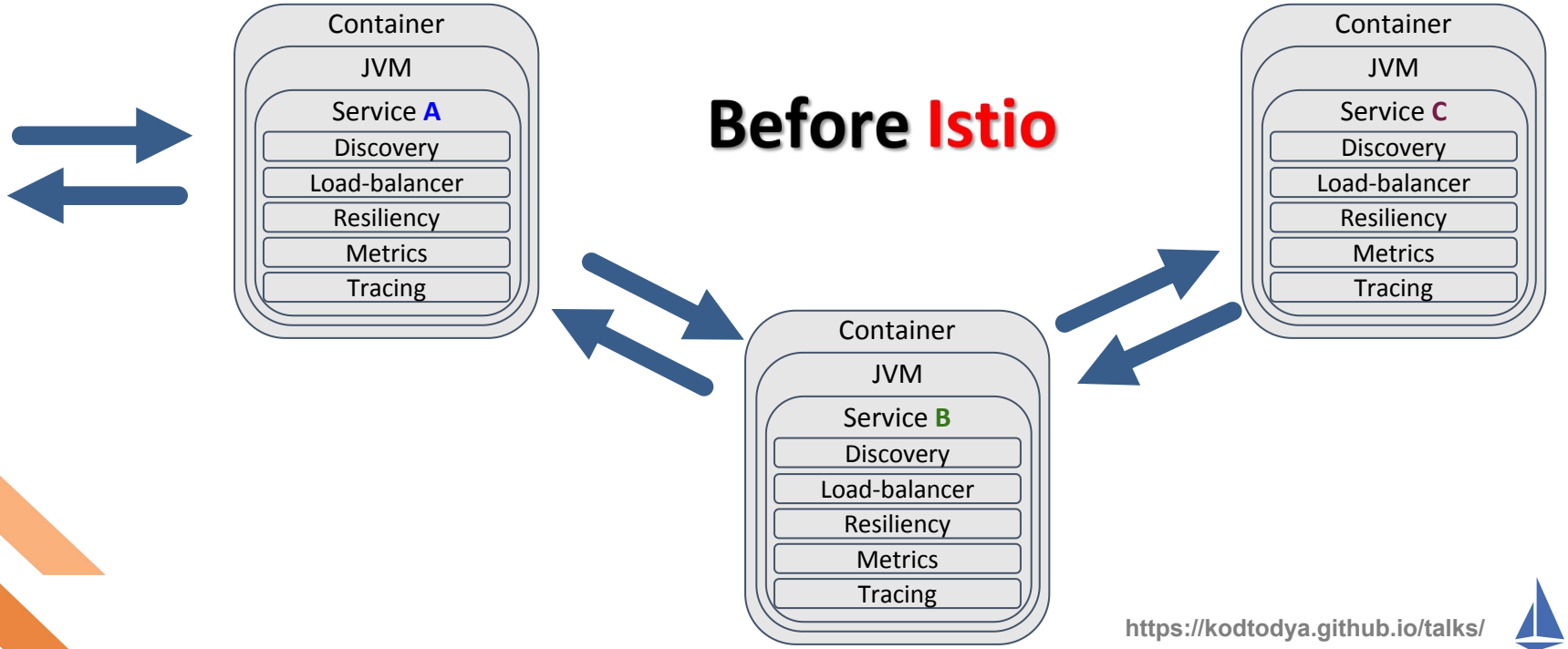
Jaeger UI


Jaeger UI
Lookup by Trace ID...
Search
Dependencies
About Jaeger
customer: default-route
View Options
Search...
Trace Start: March 22, 2018 11:38 PM
Duration: 9.01ms
Services: 4
Depth: 6
Total Spans: 7
0ms
2.25ms
4.51ms
6.76ms
9.01ms
Service & Operation
customer default-route
customer default-route
6.14ms
preference default-route
5.74ms
preferences getPreferences
4.83ms
preferences GET
3.76ms
preference default-route
2.22ms
recommendation default-route
1.5ms

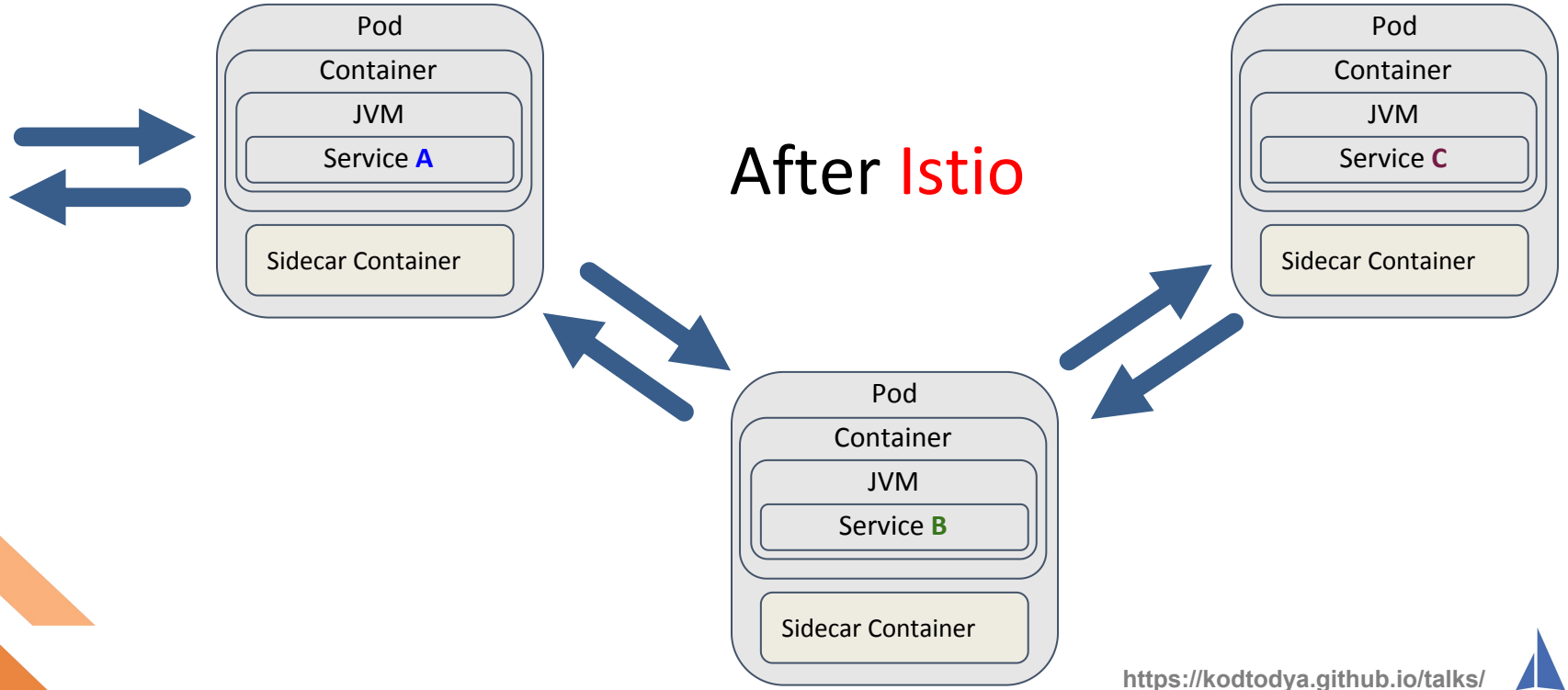
Kiali UI



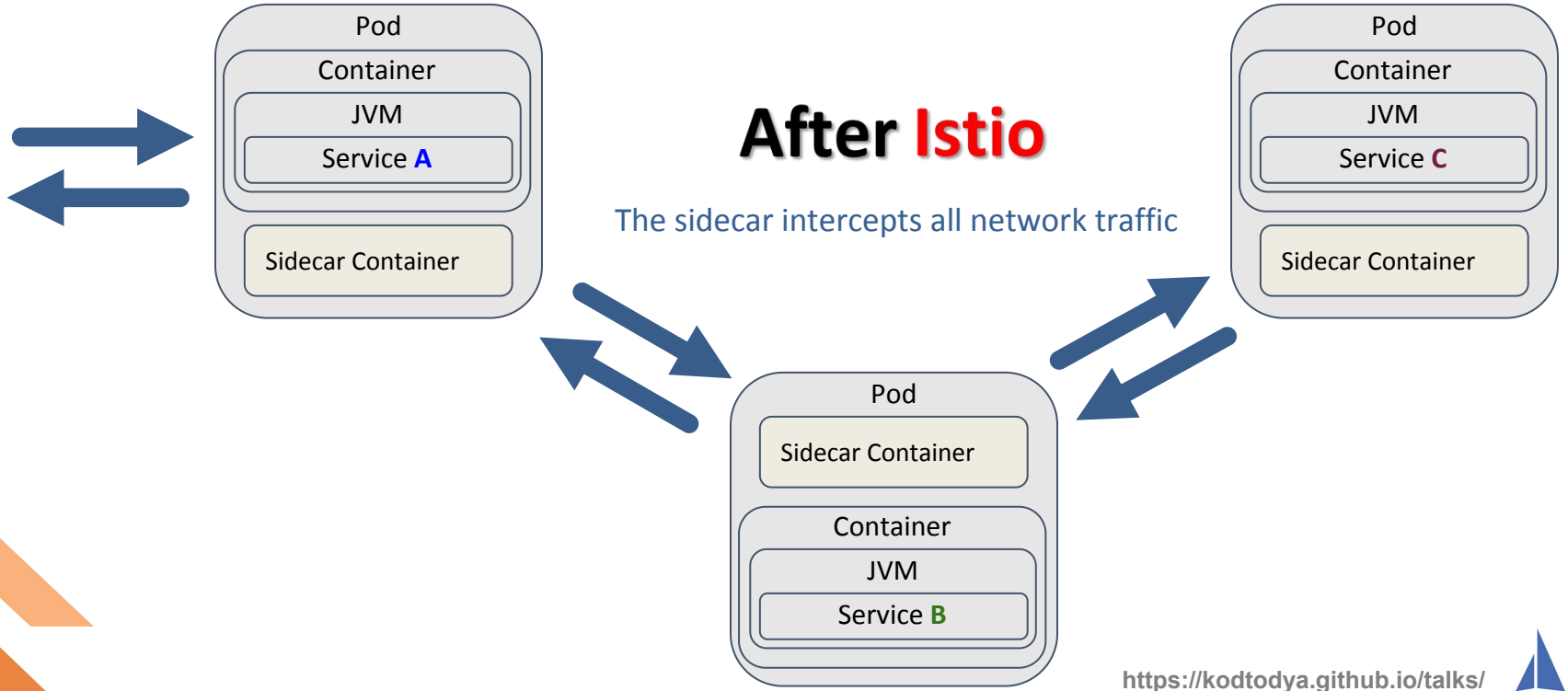
Microservices embedding Capabilities

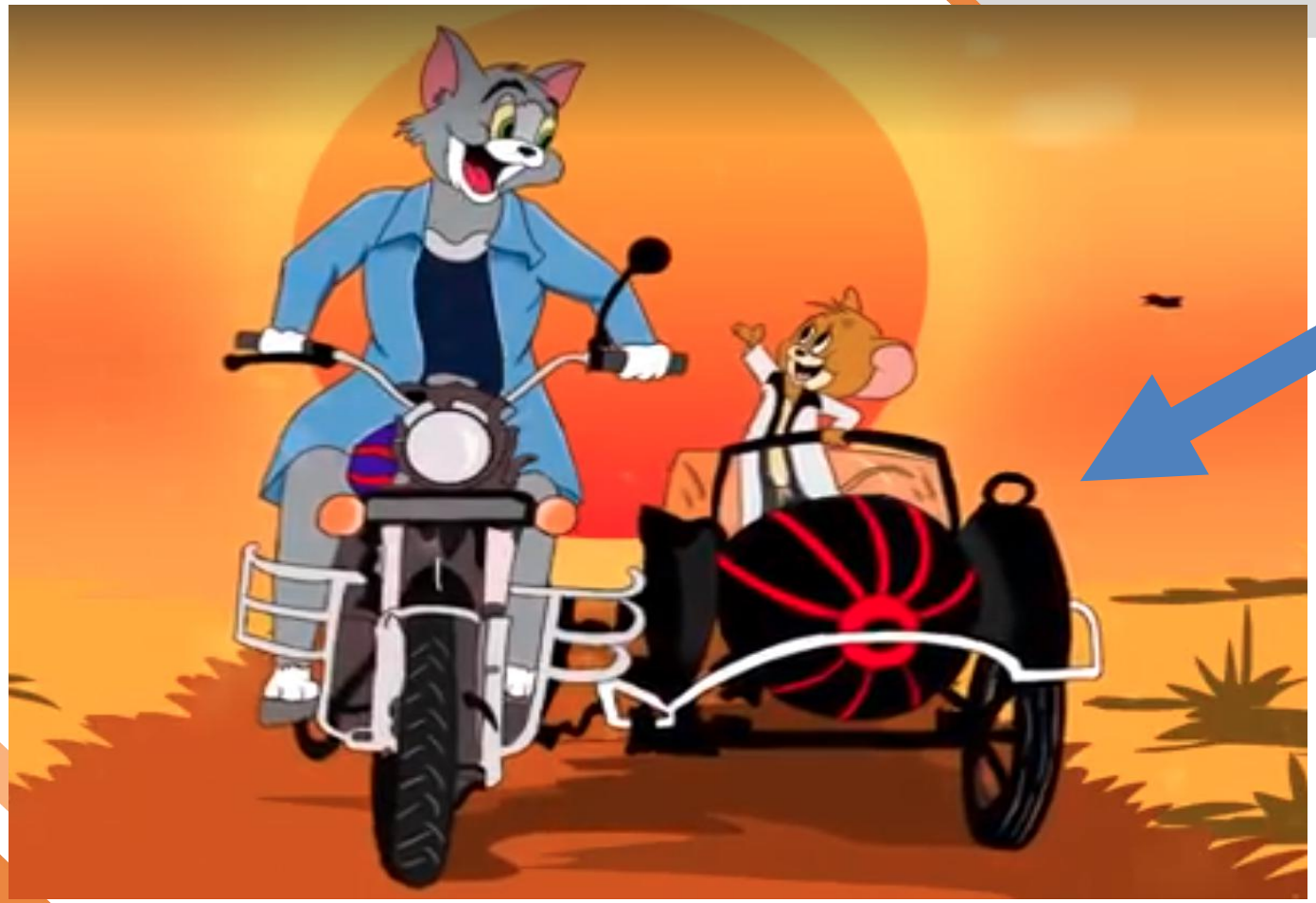


Microservices externalizing Capabilities



Microservices externalizing Capabilities





https://kodtodya.github.io/talks/

How to add an Istio-Proxy (sidecar)?

istioctl kube-inject -f NormalDeployment.yaml

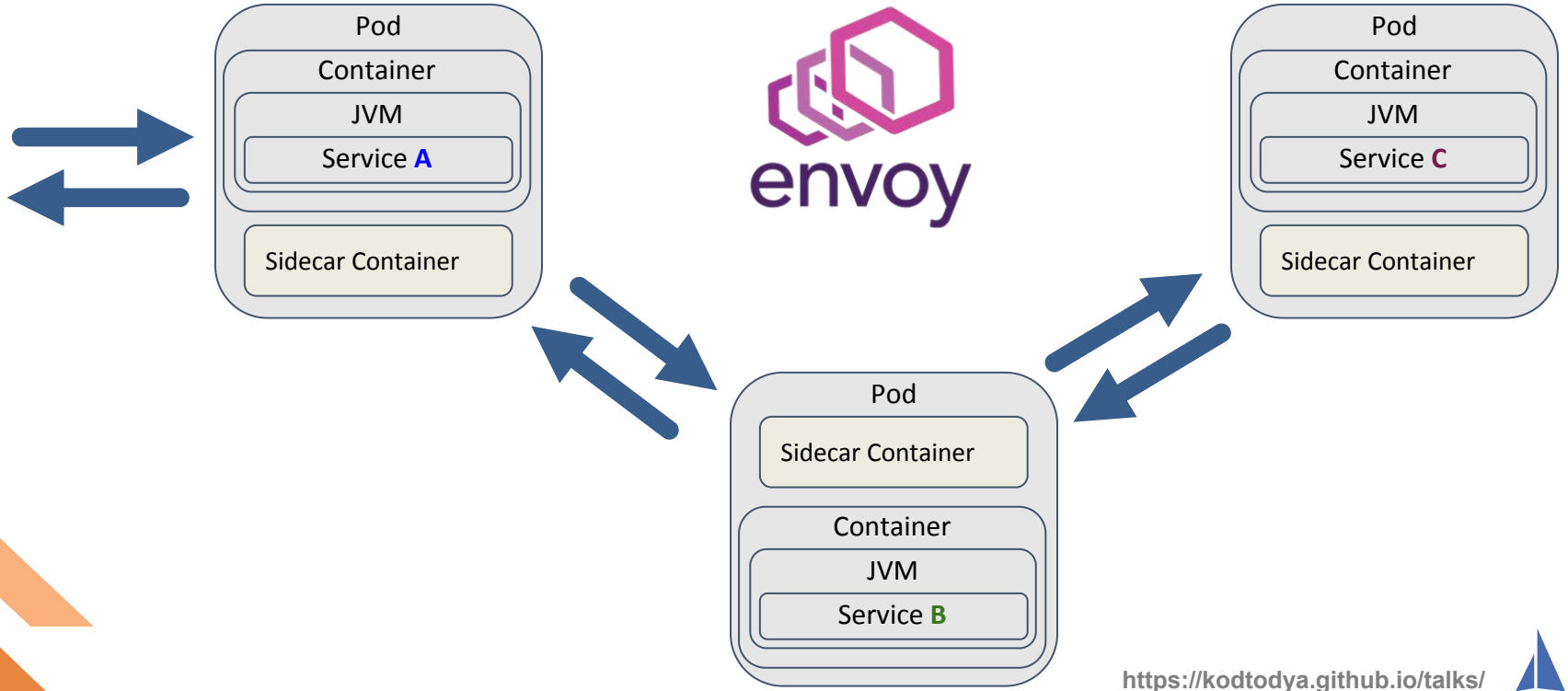
OR

kubectl label namespace tutorial istio-injection=enabled

To "see" the sidecar:

kubectl describe deployment customer

Envoy is the current sidecar

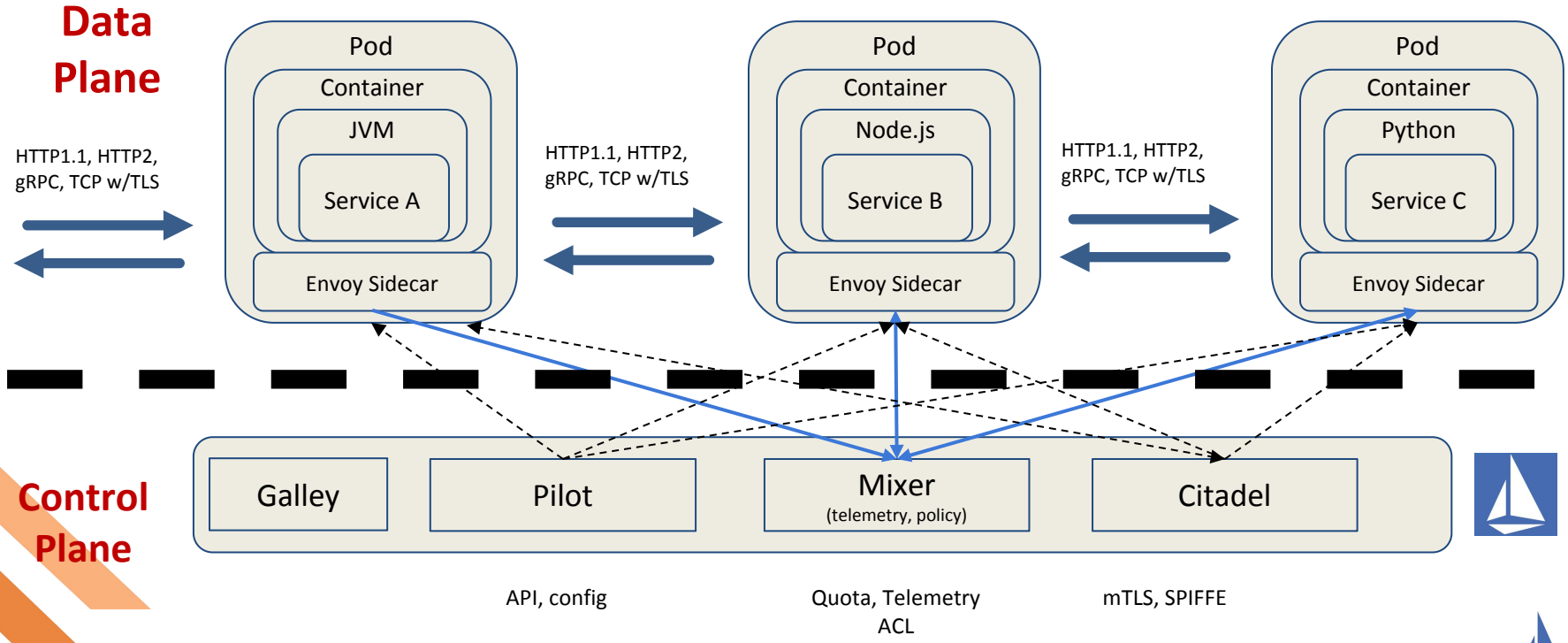


Next Generation Microservices - Service Mesh

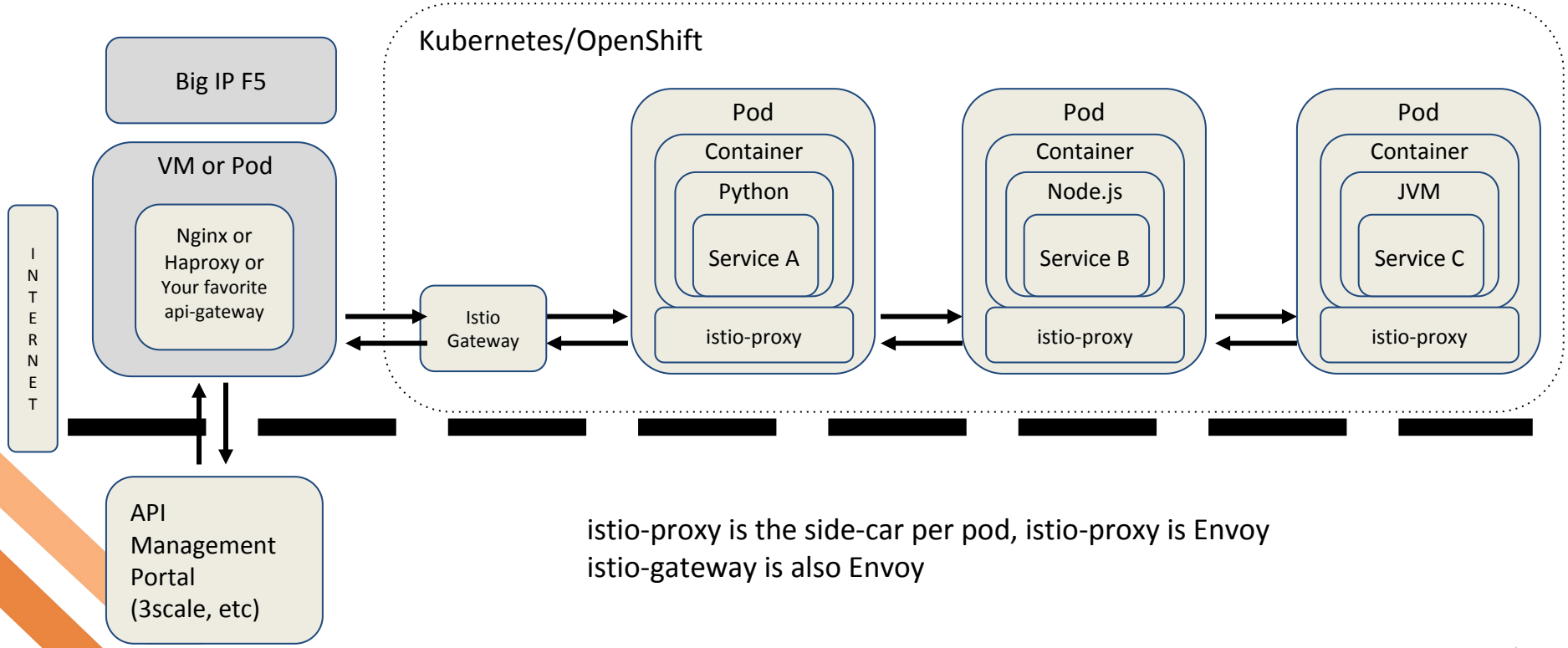
Code Independent (Polyglot)

- Intelligent Routing and Load-Balancing
  - Smarter Canary Releases
  - Dark Launch
- Chaos: Fault Injection
- Resilience: Circuit Breakers
- Observability & Telemetry: Metrics and Tracing
- Security: Encryption & Authorization
- Fleet wide policy enforcement

Istio Data Plane vs Control Plane



API Gateways vs Service Mesh



istio-proxy is the side-car per pod, istio-proxy is Envoy
istio-gateway is also Envoy

kubectl get crds

Adapters.config.istio.io
Apikeys.config.istio.io
Attributemanifests.config.istio.io
Authorizations.config.istio.io
Bypasses.config.istio.io
Checknothings.config.istio.io
Circonuses.config.istio.io
Cloudwatches.config.istio.io
Deniers.config.istio.io
Destinationrules.networking.istio.io
Dogstatsds.config.istio.io
Edges.config.istio.io
Envoyfilters.networking.istio.io
Fluentds.config.istio.io
Gateways.networking.istio.io
Handlers.config.istio.io
Httpapispecbindings.config.istio.io
Httpapispecs.config.istio.io
Instances.config.istio.io
Kubernetesenvs.config.istio.io
Kuberneteses.config.istio.io
Listcheckers.config.istio.io
Listentries.config.istio.io
Logentries.config.istio.io
Memquotas.config.istio.io
Meshpolicies.authentication.istio.io
Metrics.config.istio.io
Metrics.config.istio.io
Noops.config.istio.io
Opas.config.istio.io
Policies.authentication.istio.io
Prometheuses.config.istio.io
Quotas.config.istio.io
Quotaspecbindings.config.istio.io
Quotaspecs.config.istio.io
Rbacconfigs.rbac.istio.io
Rbacs.config.istio.io
Redisquotas.config.istio.io
Reportnothings.config.istio.io
Rules.config.istio.io

CustomResourceDefinitions of Istio 1.0.x

kubectl api-resources | grep istio

Servicecontrolreports.config.istio.io
Servicecontrols.config.istio.io
Serviceentries.networking.istio.io
Servicerolebindings.rbac.istio.io
Serviceroles.rbac.istio.io
Signalfxs.config.istio.io
Solarwindses.config.istio.io
Stackdrivers.config.istio.io
Statsds.config.istio.io
Stdios.config.istio.io
Templates.config.istio.io
Tracespans.config.istio.io
Virtualservices.networking.istio.io

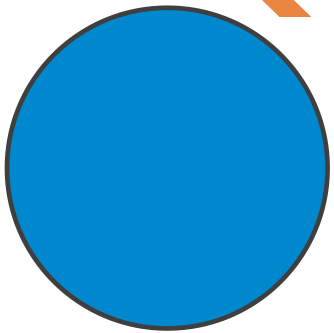
Main Istio Resources (API Objects based on CRDs)

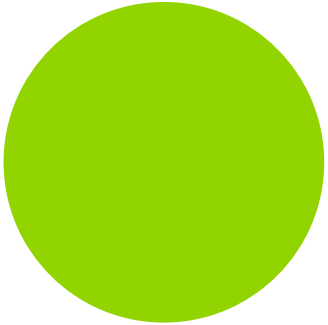
- VirtualService
  - defines the rules that control how requests for a service are routed within an Istio service mesh
  - routing logic, load weighting, chaos injection
- DestinationRule
  - configures the set of policies to be applied to a request after VirtualService routing has occurred
  - load-balancer, outlier, circuit breaker
- **ServiceEntry** - egress enablement
- **Gateway** - making a service external to cluster - Ingres
- **Policy** - enable mTLS
- **ServiceRole** - roles for RBAC
- **ServiceRoleBinding** - "users" for the ServiceRole

Traffic Control

Traffic Control

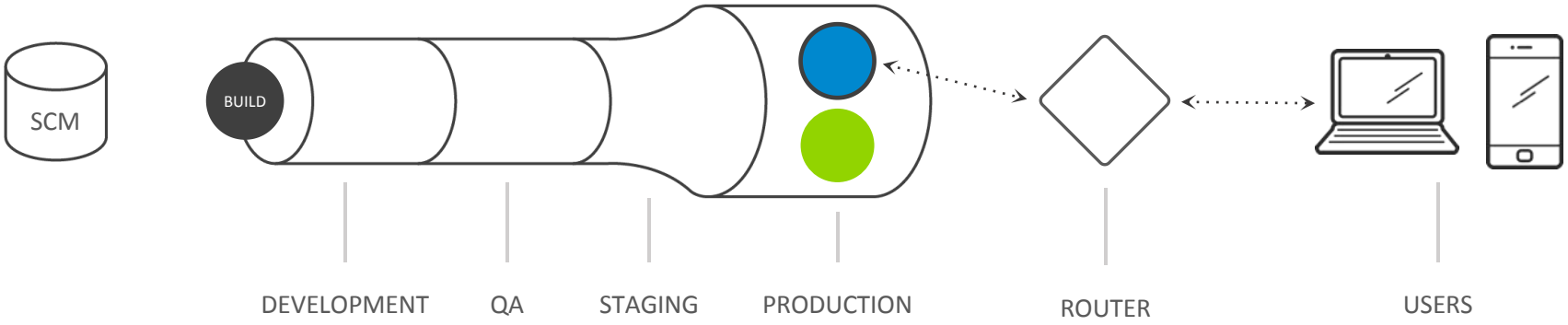
- Blue/Green part of base Kubernetes/OpenShift
- Percentages not based on pod count - Canary Deployment
- Smart Canaries
- Dark Launch



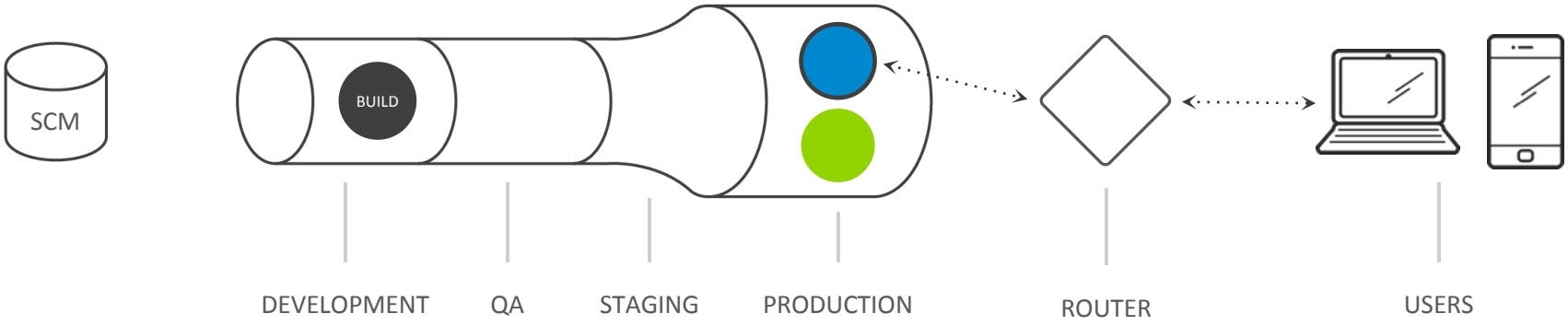


Blue/Green Deployment

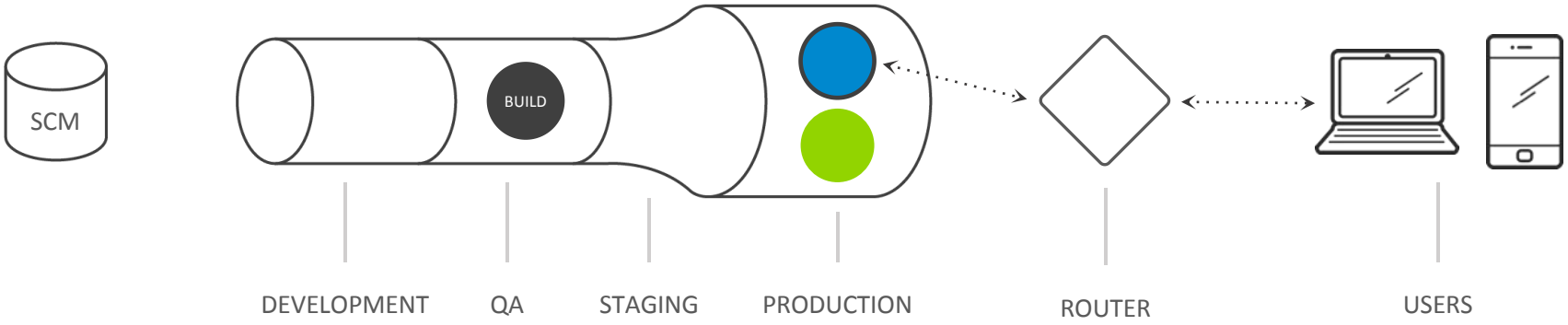
Blue/Green Deployment



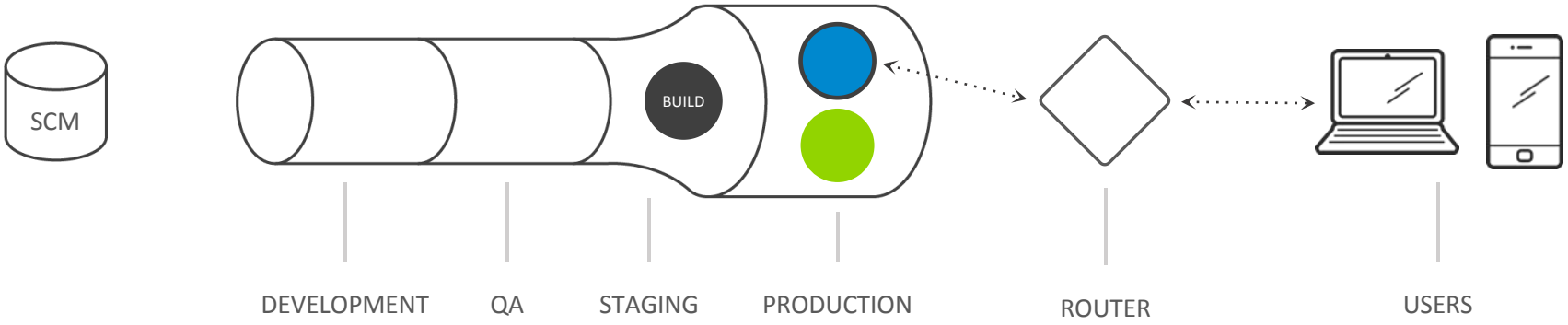
Blue/Green Deployment



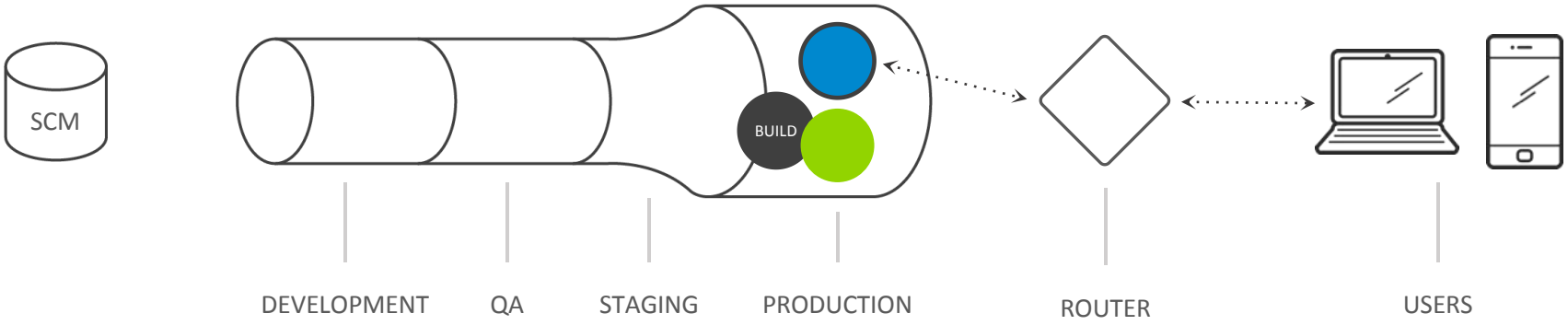
Blue/Green Deployment



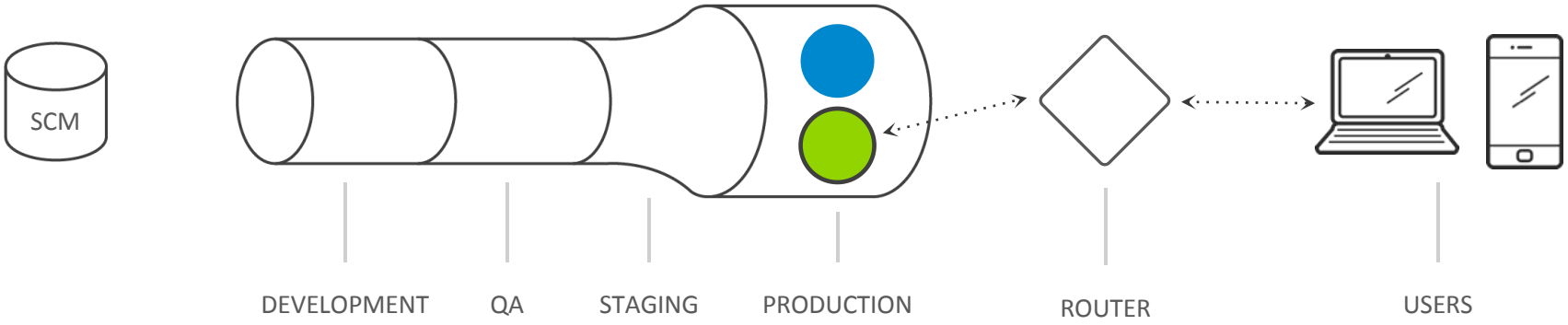
Blue/Green Deployment



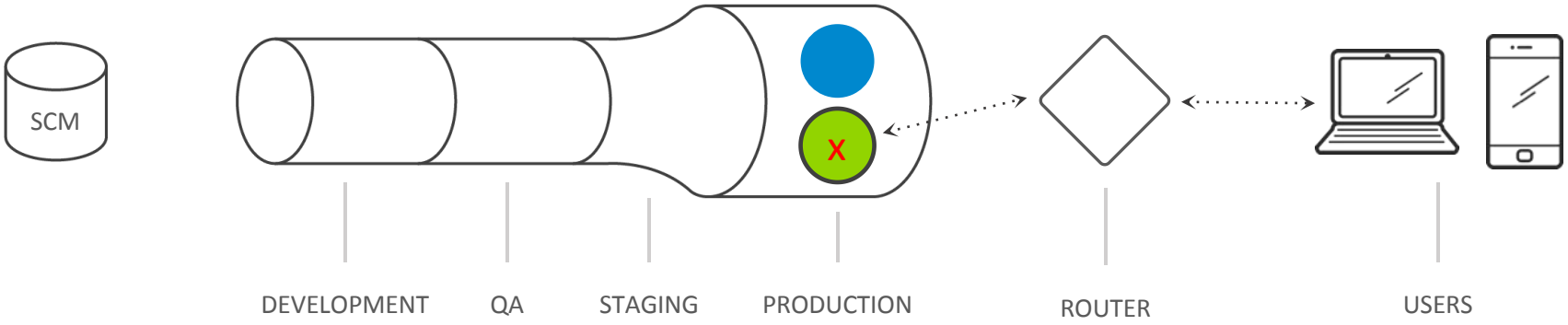
Blue/Green Deployment



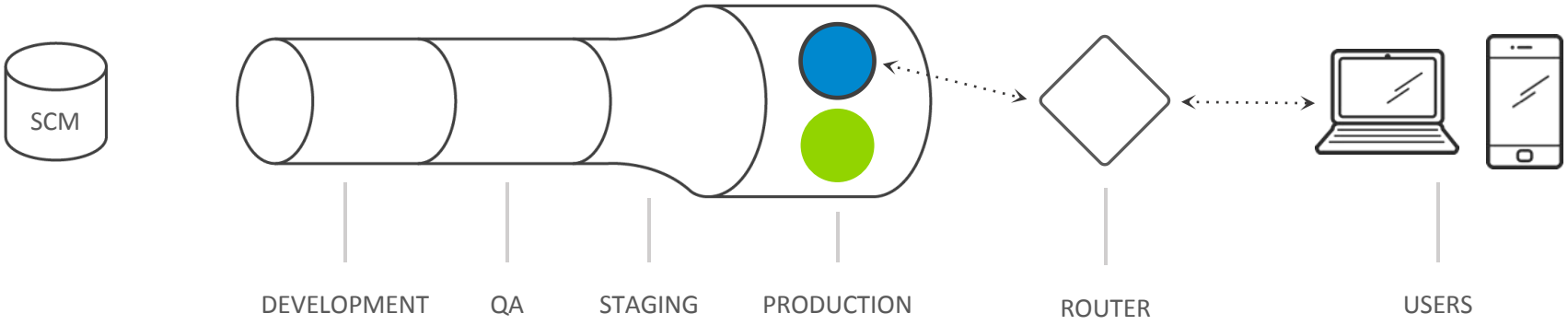
Blue/Green Deployment



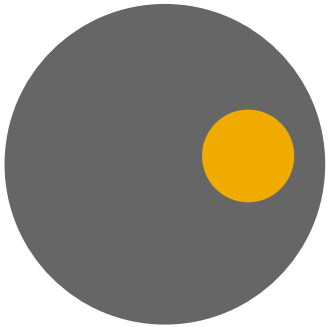
Blue/Green Deployment



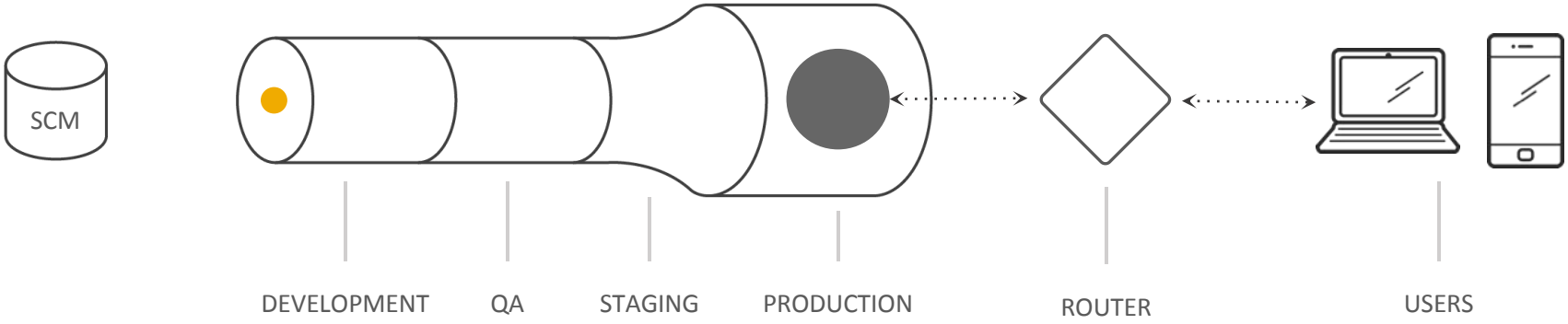
Blue/Green Deployment



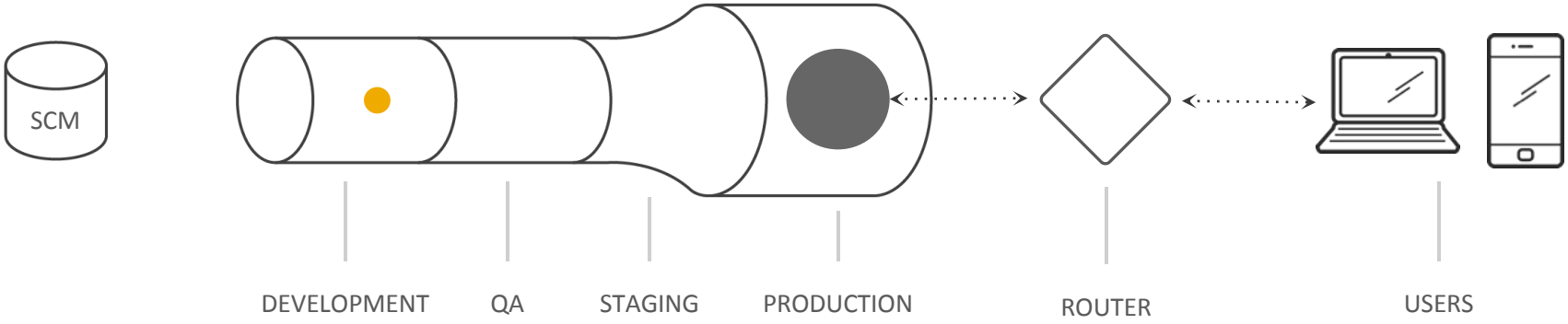
Canary Deployment



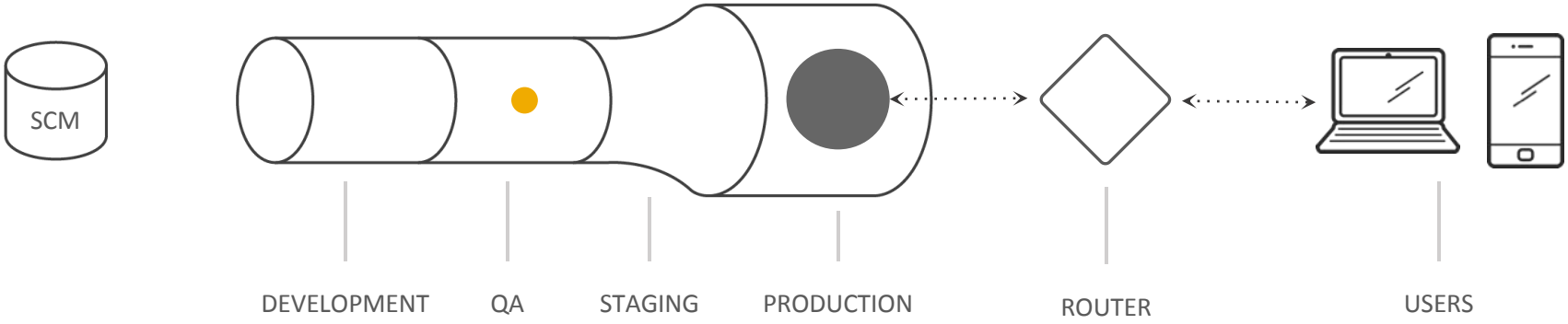
Canary Deployment



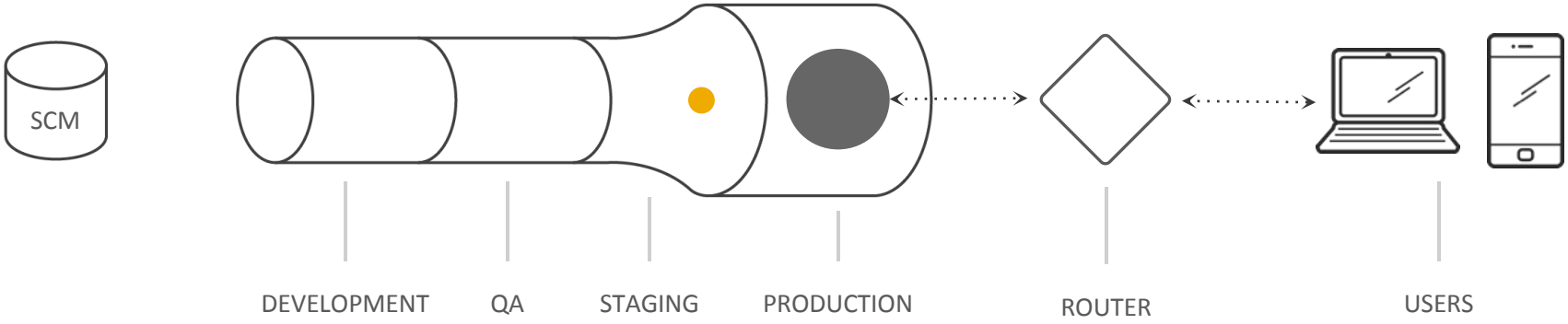
Canary Deployment



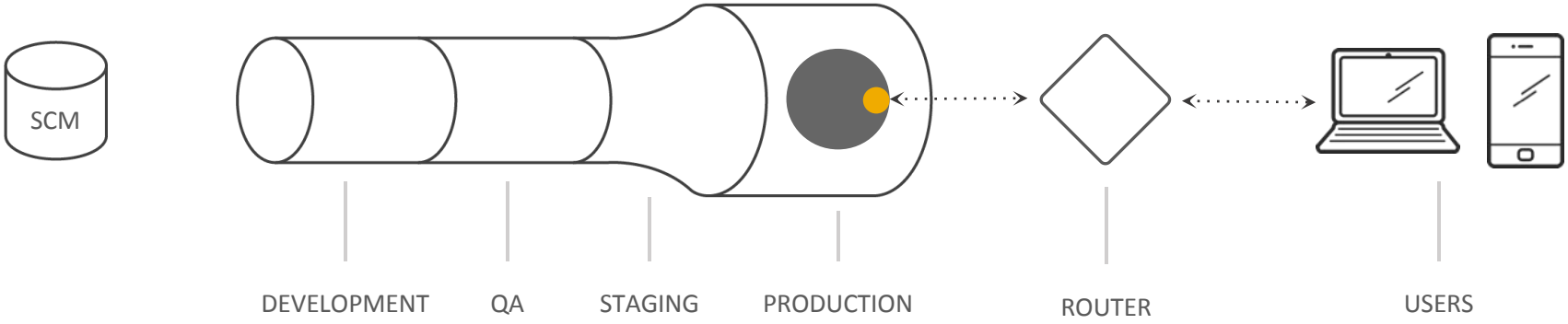
Canary Deployment



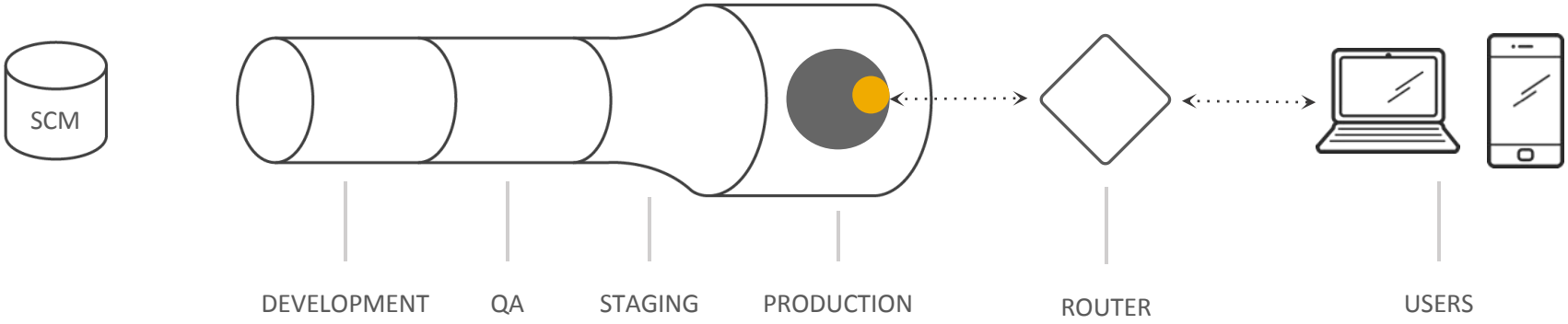
Canary Deployment



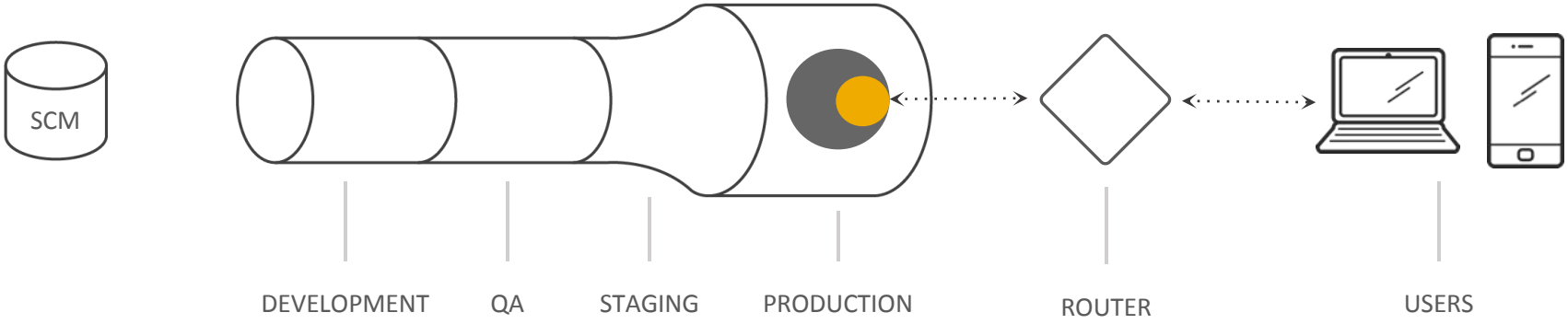
Canary Deployment



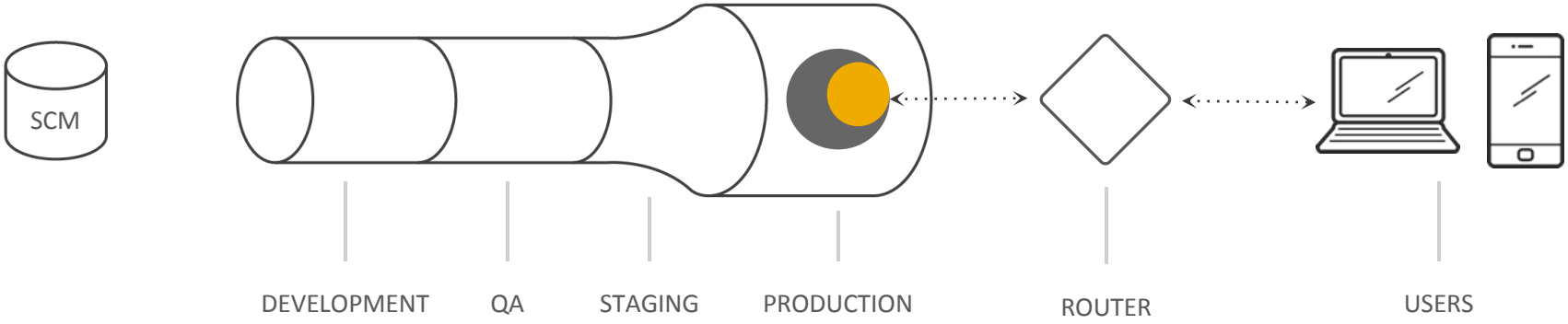
Canary Deployment



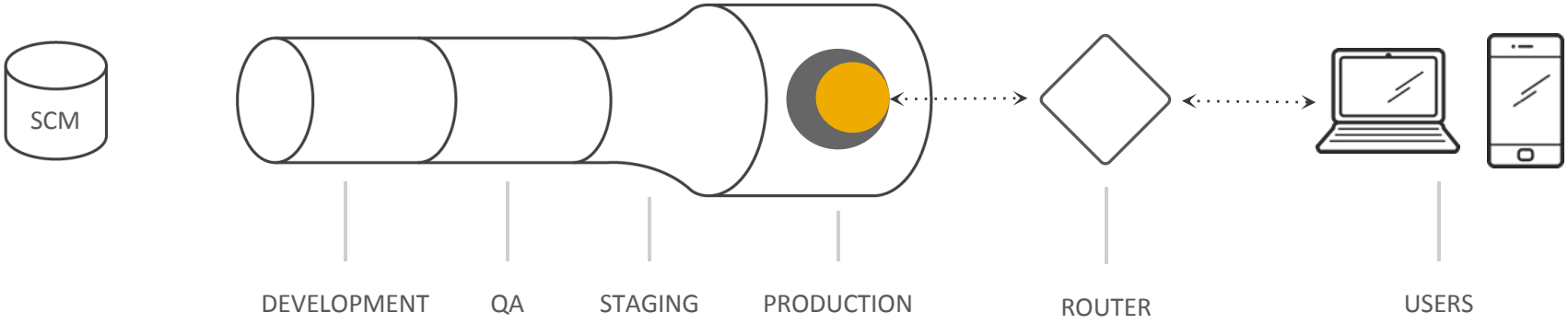
Canary Deployment



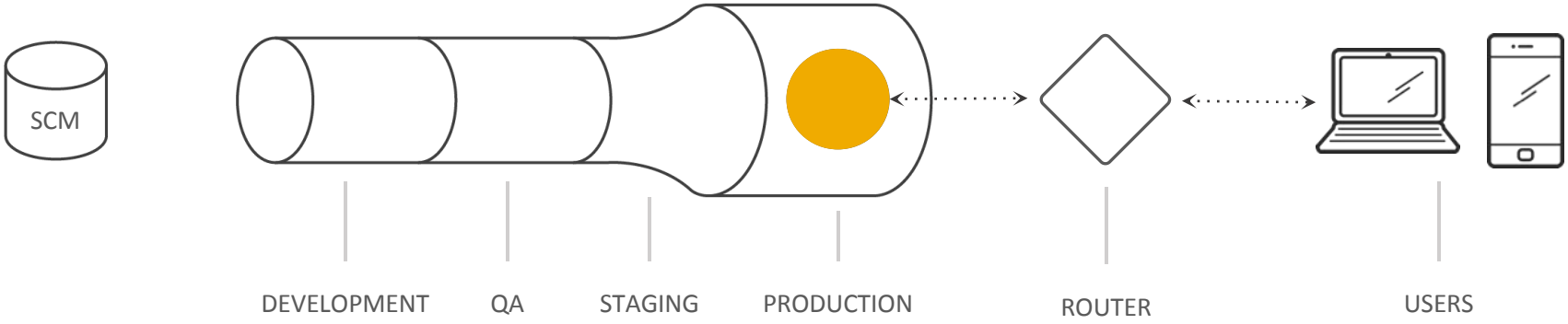
Canary Deployment



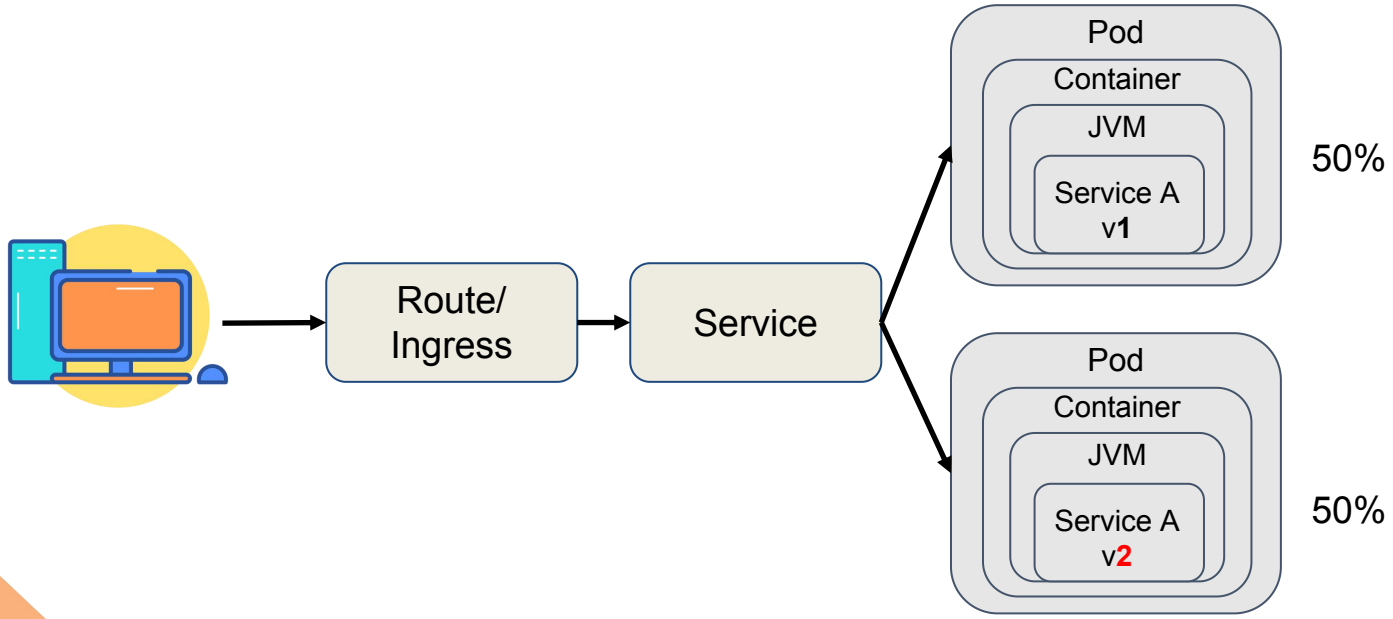
Canary Deployment



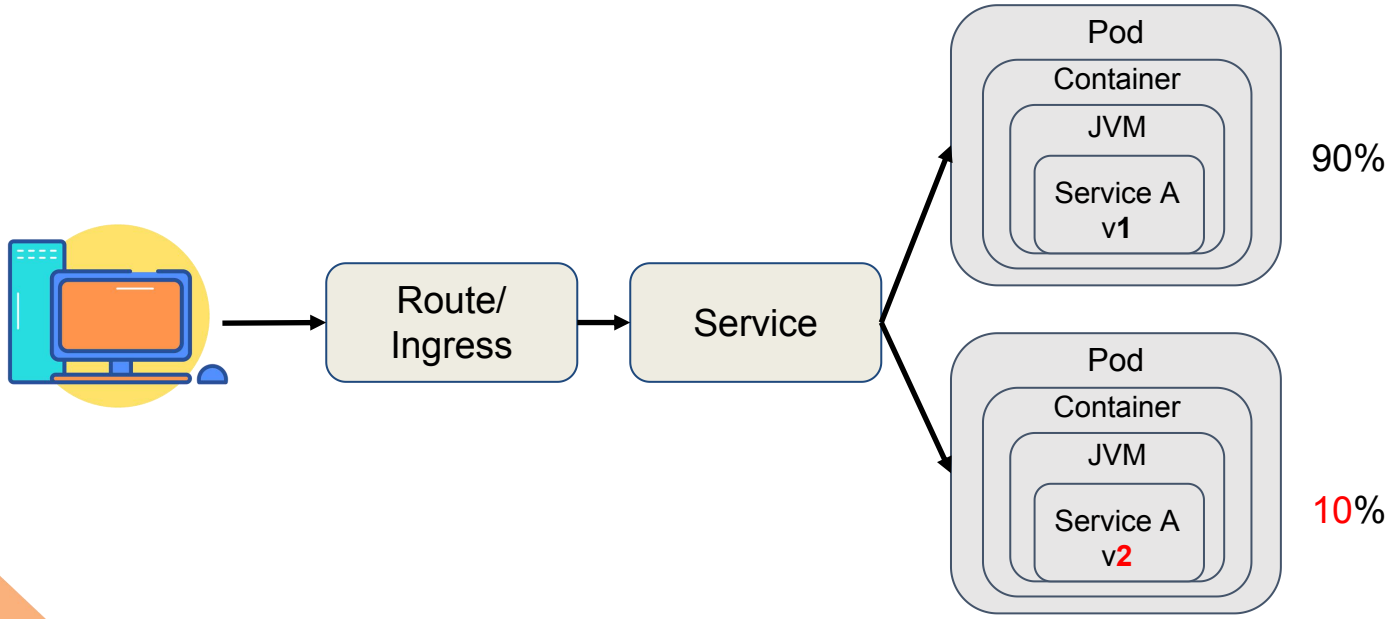
Canary Deployment



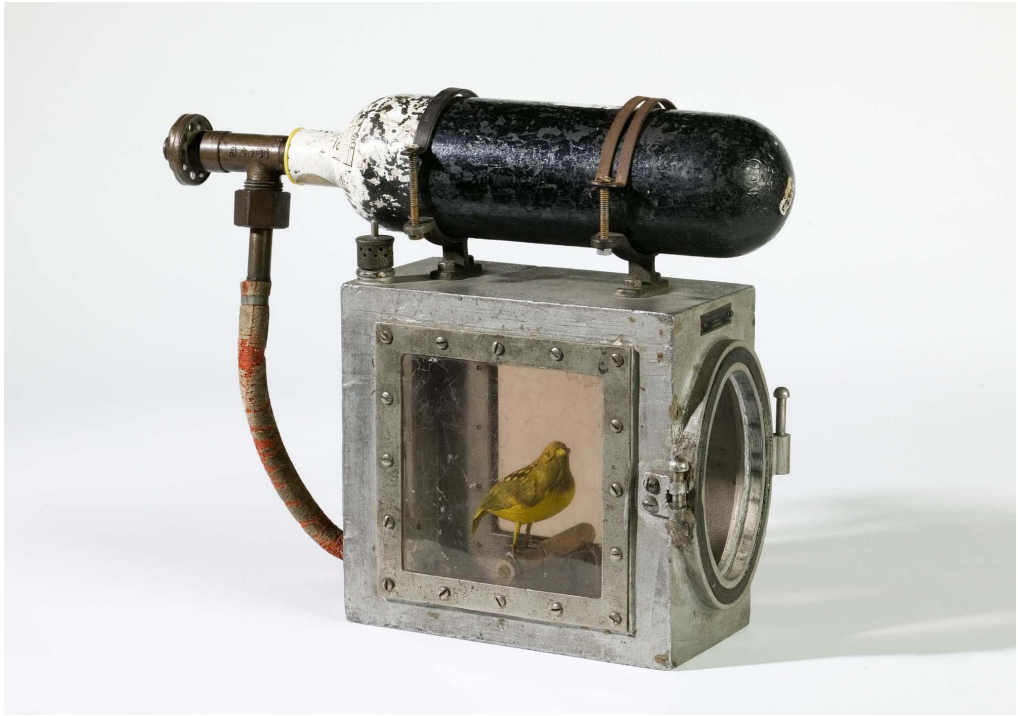
Canaries with Kubernetes



Canaries with Istio



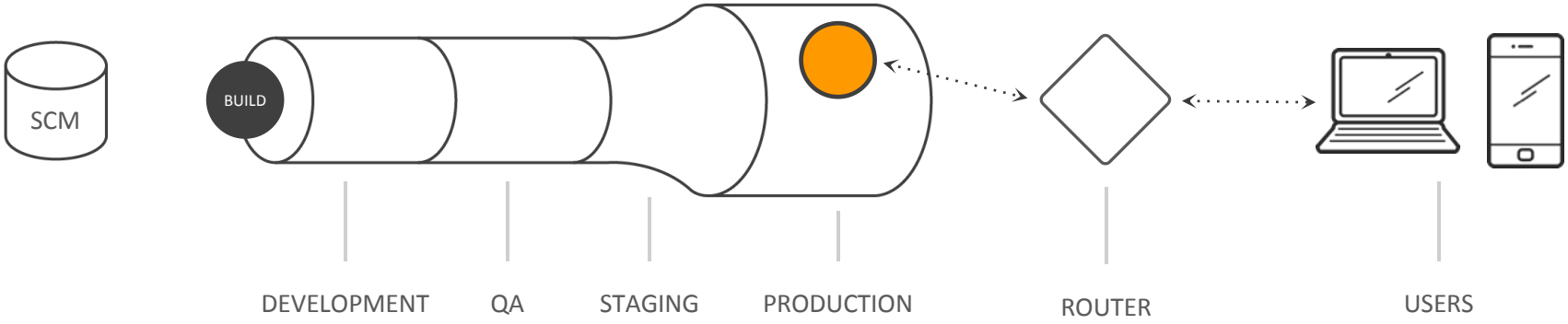
Canary Resuscitator



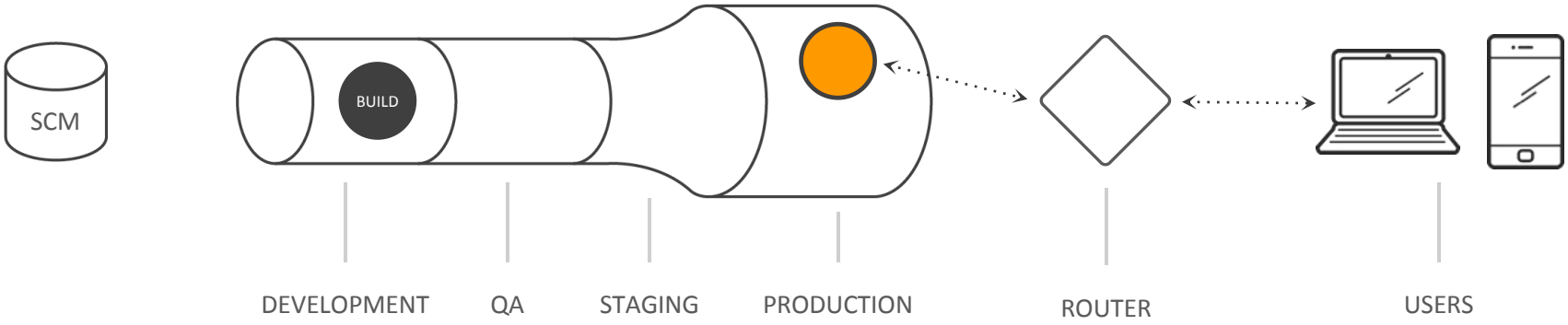
Dark Launch

Dark

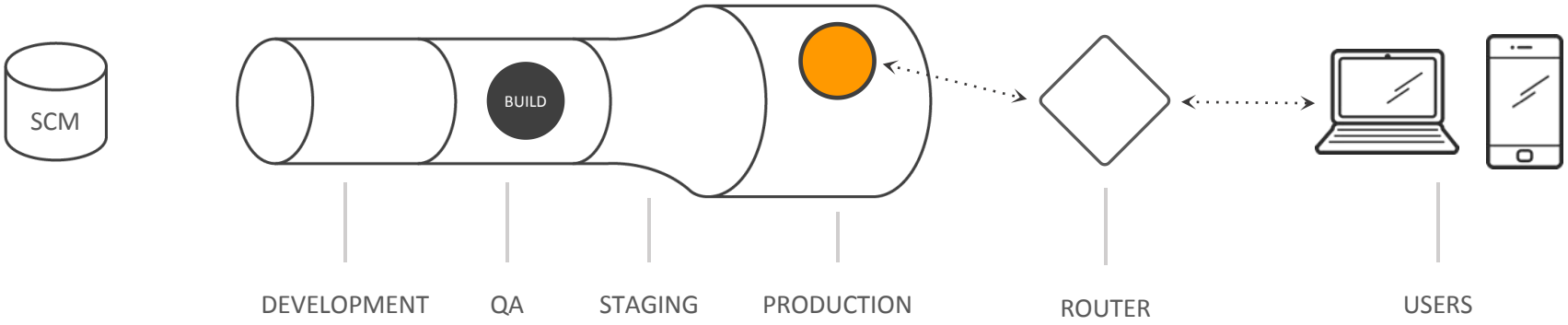
Dark Launch



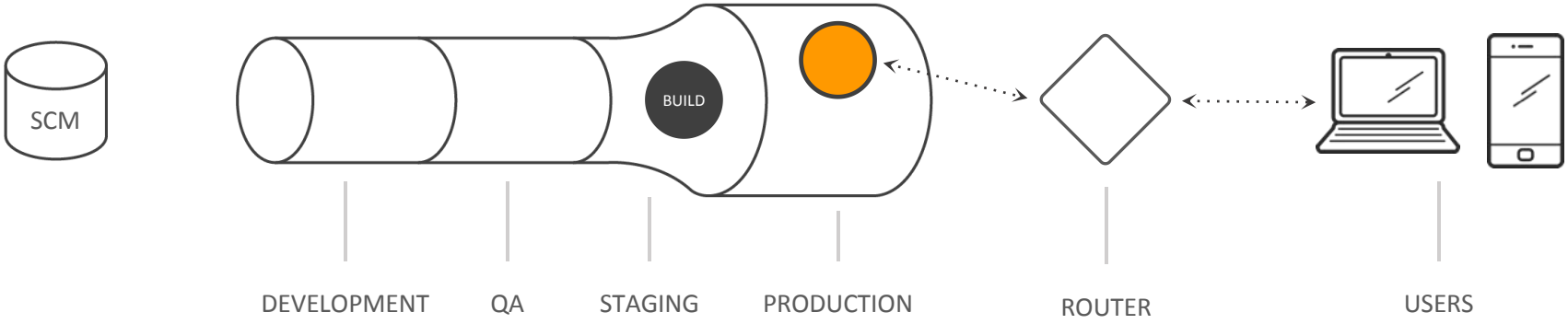
Dark Launch



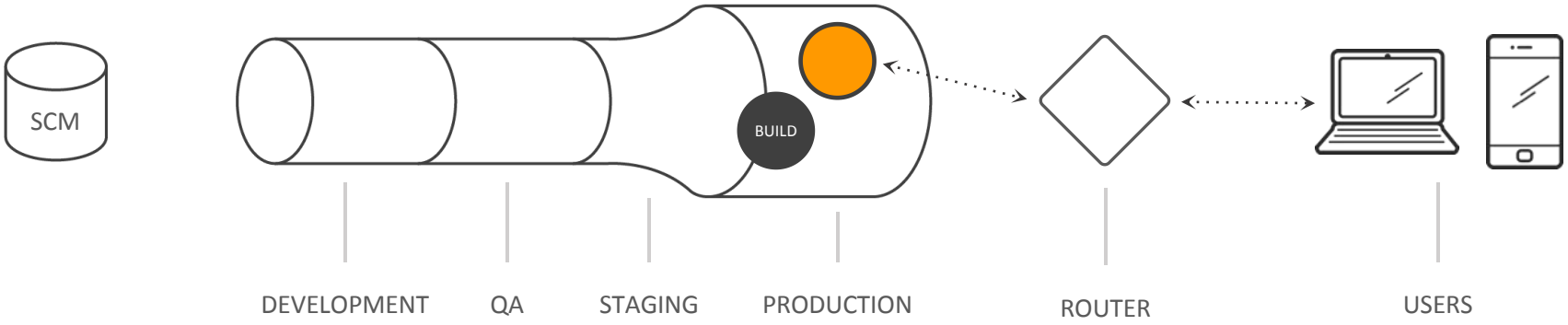
Dark Launch



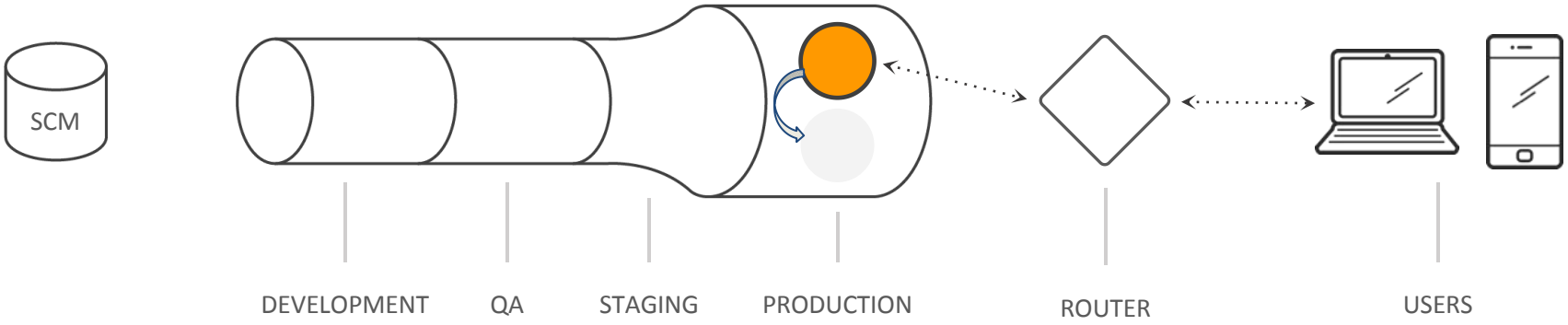
Dark Launch



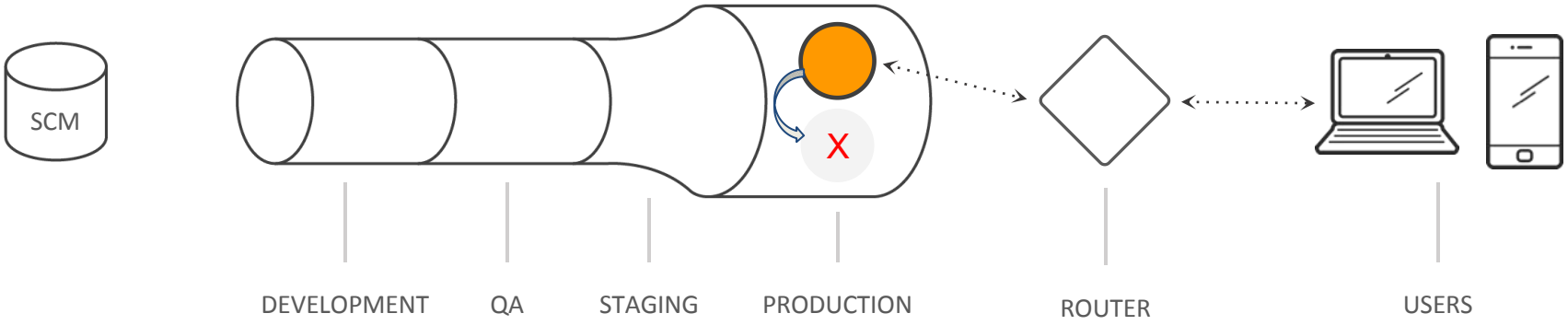
Dark Launch



Dark Launch - Mirroring

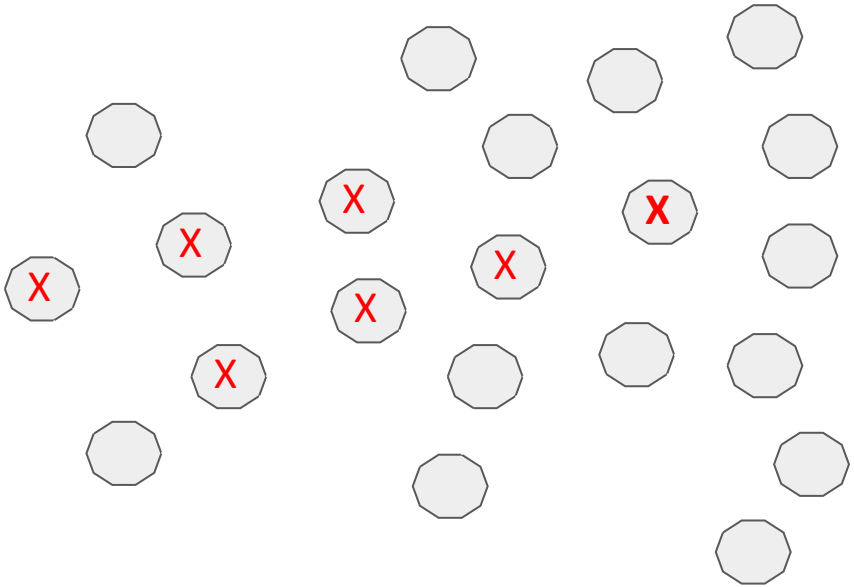


Dark Launch - Monitoring the Mirroring



Service Resiliency

- Fail Fast: Latency Circuit Breaker



Chaos Testing

https://principlesofchaos.org/

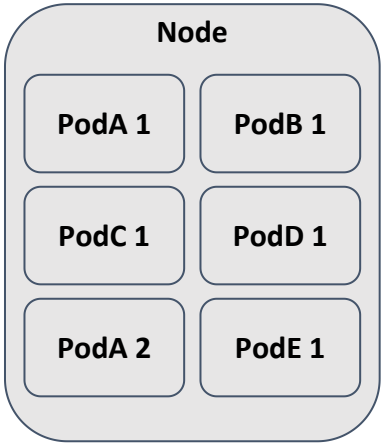
Security

Why Security?

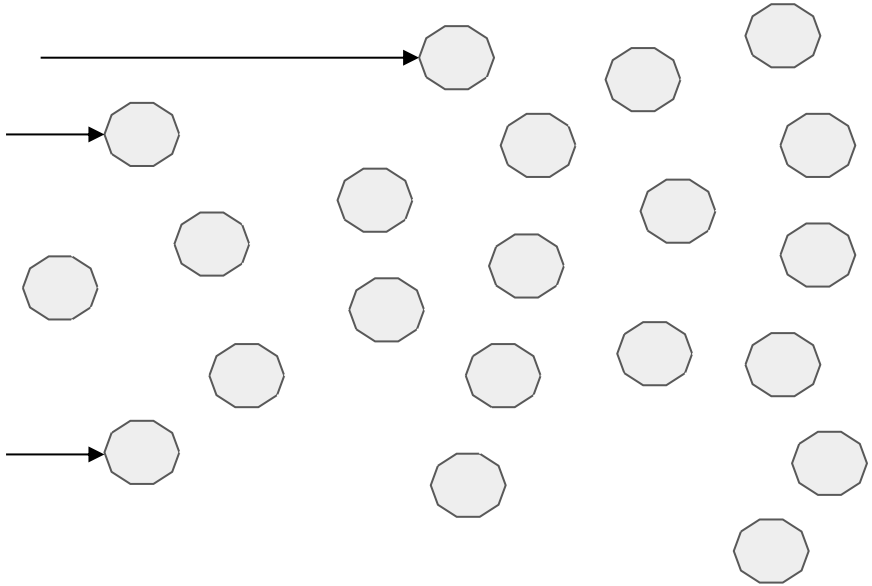
Our Teams:

A) Customer Success Engineering Team
B) Human Resources Engineering Team
C) Marketing Engineering Team
D) Manufacturing Engineering Team
E) Big Money Customer Engineering Team

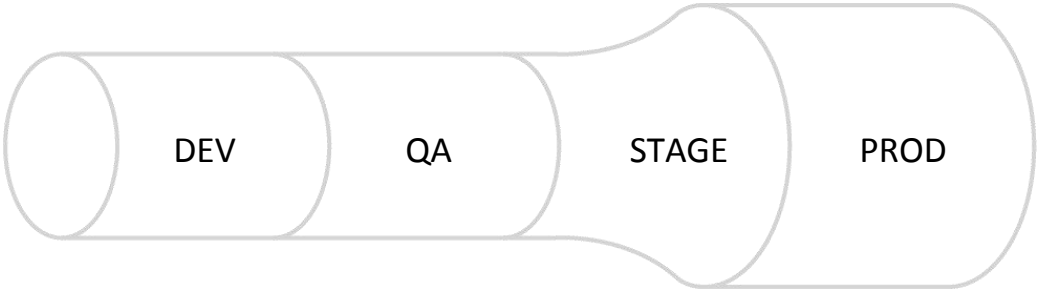
Shared Resources



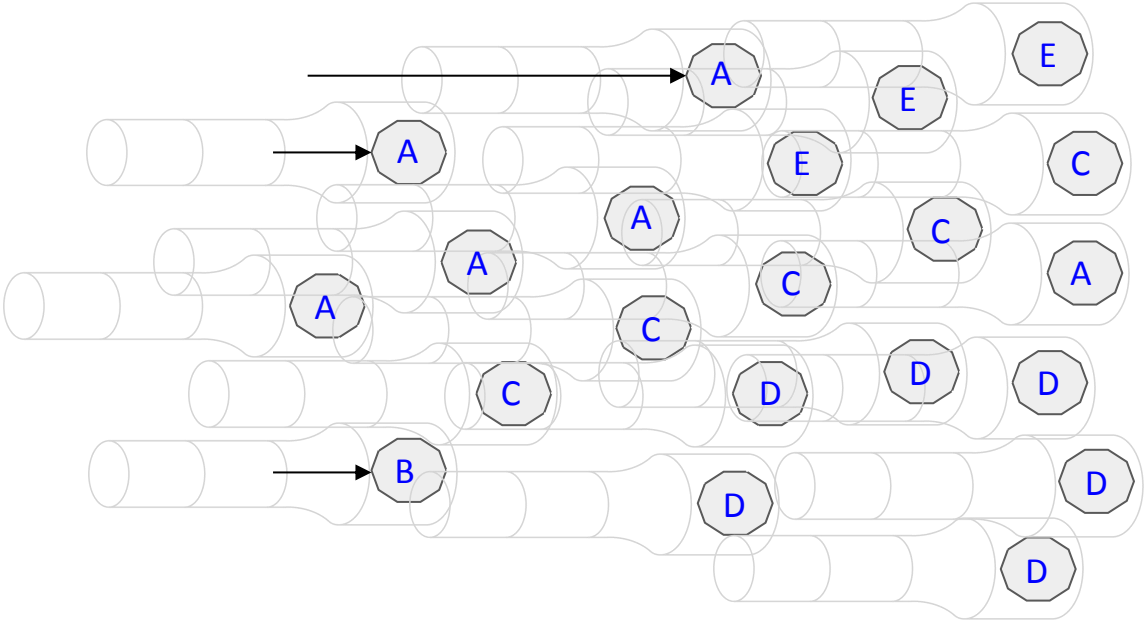
Our Service Mess



Our Pipelines



Our Services, Our Pipelines, Our Teams



Customer Success Engineering Team A
Human Resources Engineering Team B
Marketing Engineering Team C

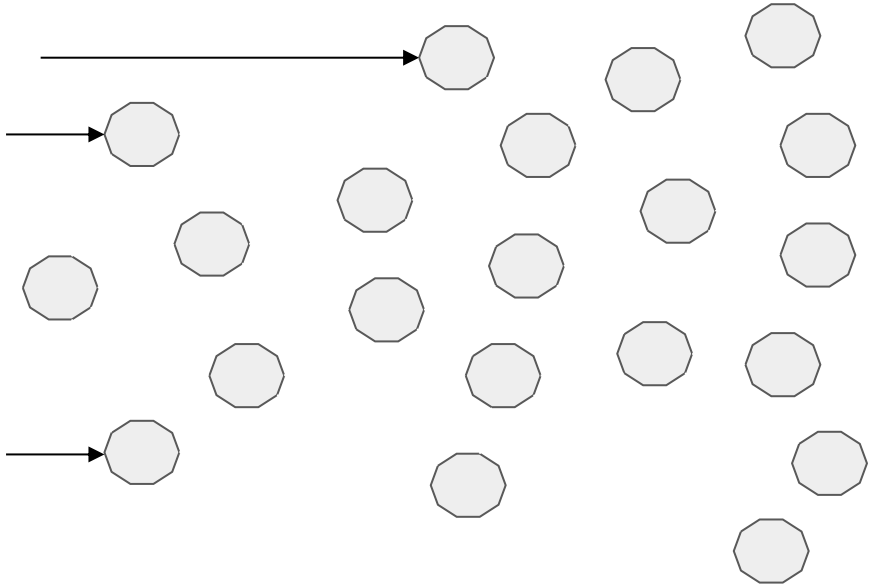
Manufacturing Engineering Team D
Big Money Customer Engineering Team E

Istio Security Capabilities

- Egress Blocking
- mTLS - Encryption
- Access Control
- JSON Web Token (JWT) Authentication
- Role-based Access Control (RBAC) Authorization

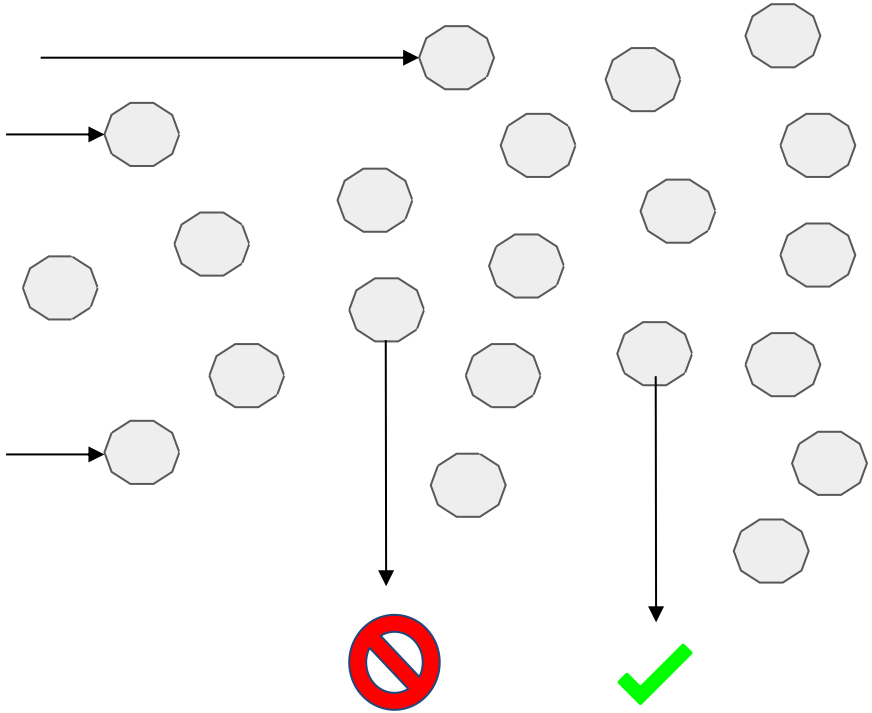
Istio Security Capabilities

- ❑ Egress Blocking
- ❑ mTLS - Encryption
- ❑ Access Control
- ❑ JSON Web Token (JWT) Authentication
- ❑ Role-based Access Control (RBAC) Authorization

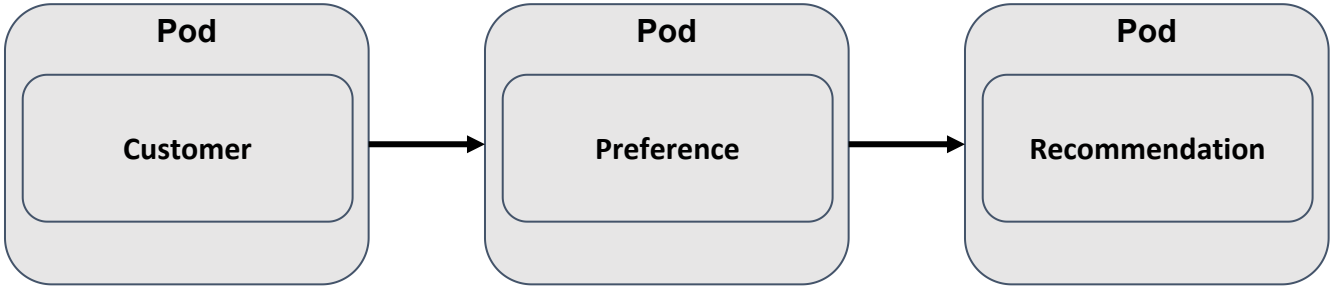


Most Communication Inbound & Internal

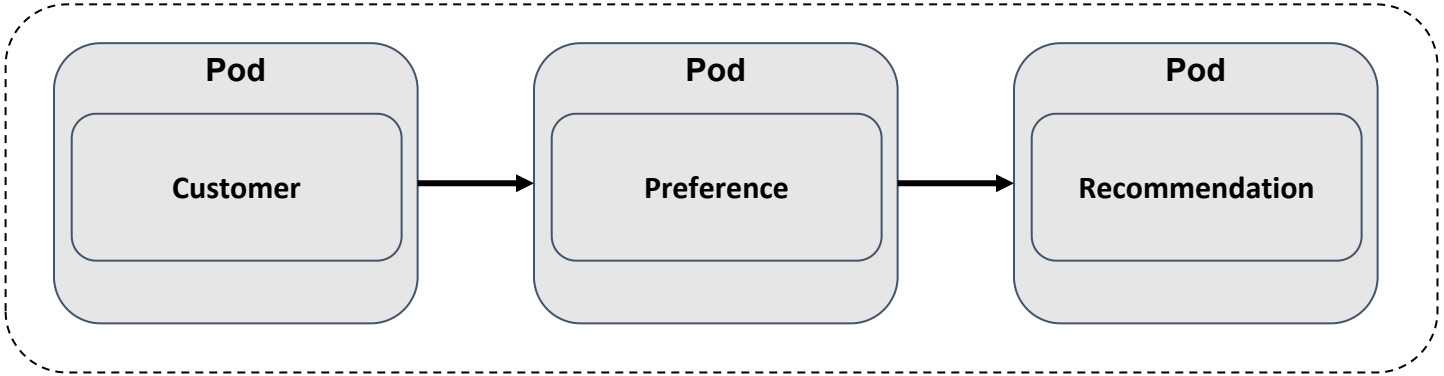
Egress Blocking



Why Encryption?



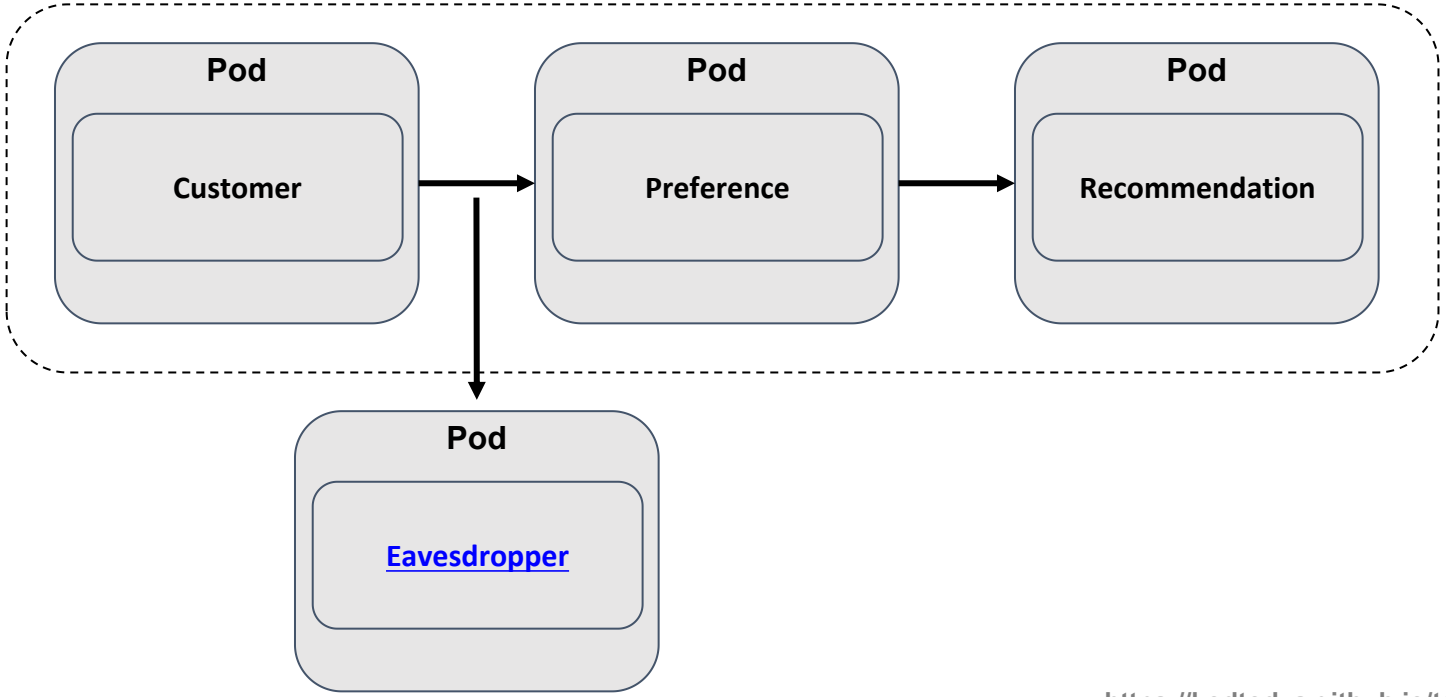
Why Encryption?



Big Money Customer Engineering Team

Why Encryption?

Big Money Customer Engineering Team



istio-proxy@preference-v1-5485dc6f49-fspbb: /

istio | Shell | ...e-v1-5485dc6f49-fspbb: /

istio-proxy@preference-v1-5485dc6f49-fspbb: /

```
p,nop,TS val 9122945 ecr 9122943], length 172: HTTP: HTTP/1.1 200 OK
E.....@.@.1.....>..5.*E......Y......
..4...4.HTTP/1.1 200 OK
content-length: 47
x-envoy-upstream-service-time: 0
date: Mon, 24 Dec 2018 17:26:01 GMT
server: envoy

recommendation v1 from '66b7c9779c-75fpl': 347
```

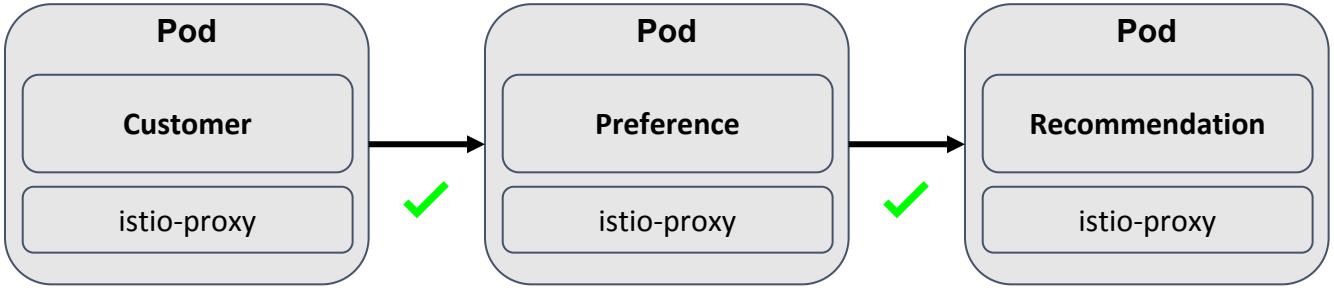
@preference-v1-5485dc6f49-fspbb:~

```
[jboss@preference-v1-5485dc6f49-fspbb ~]$ curl recommendation:8080
recommendation v2 from '7cbd9f9c79-nd69g': 341
[jboss@preference-v1-5485dc6f49-fspbb ~]$ curl recommendation:8080
recommendation v1 from '66b7c9779c-75fpl': 347
[jboss@preference-v1-5485dc6f49-fspbb ~]$
```

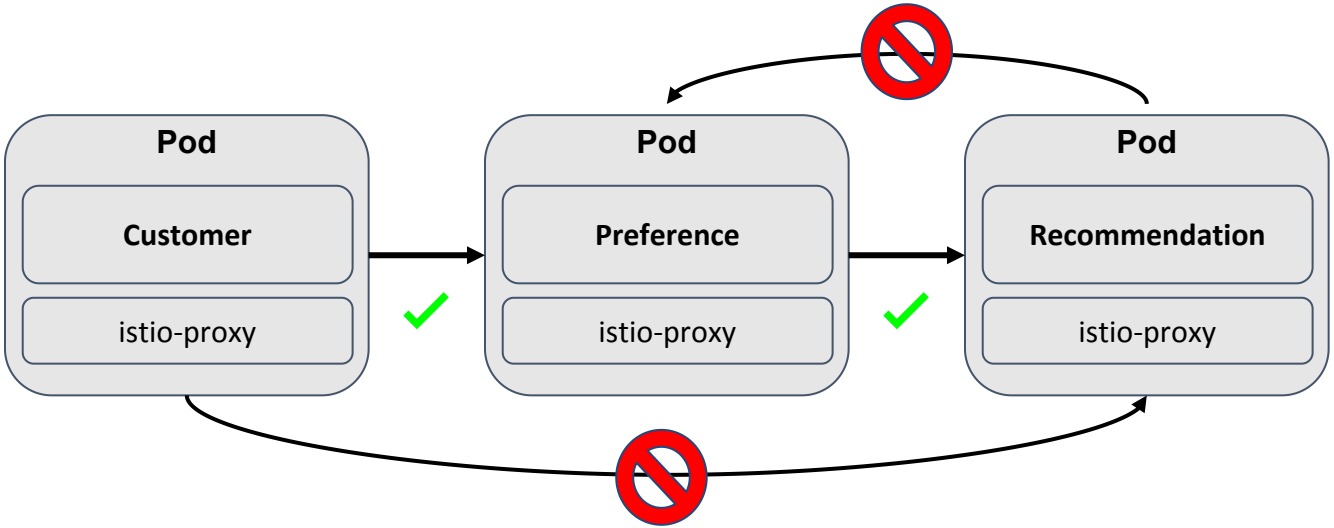
istio

```
~/minishift_1.27.0/istio-tutorial $
```

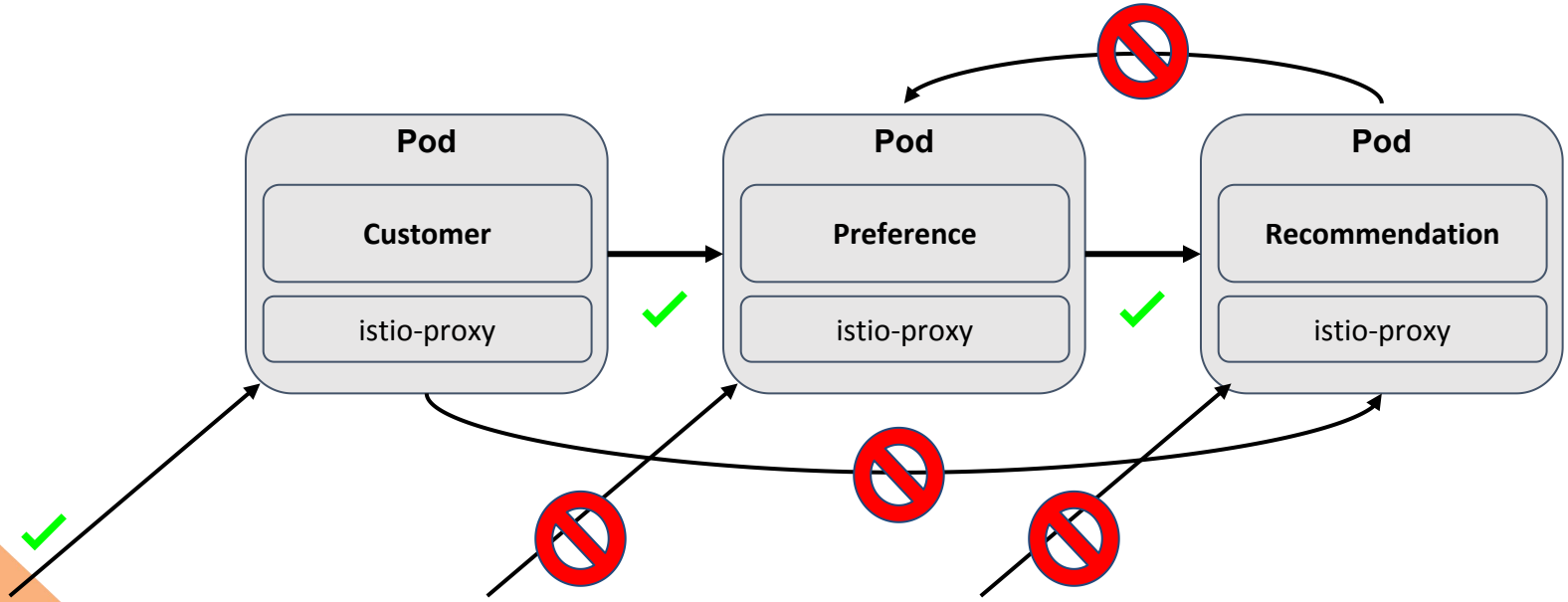
Access Control



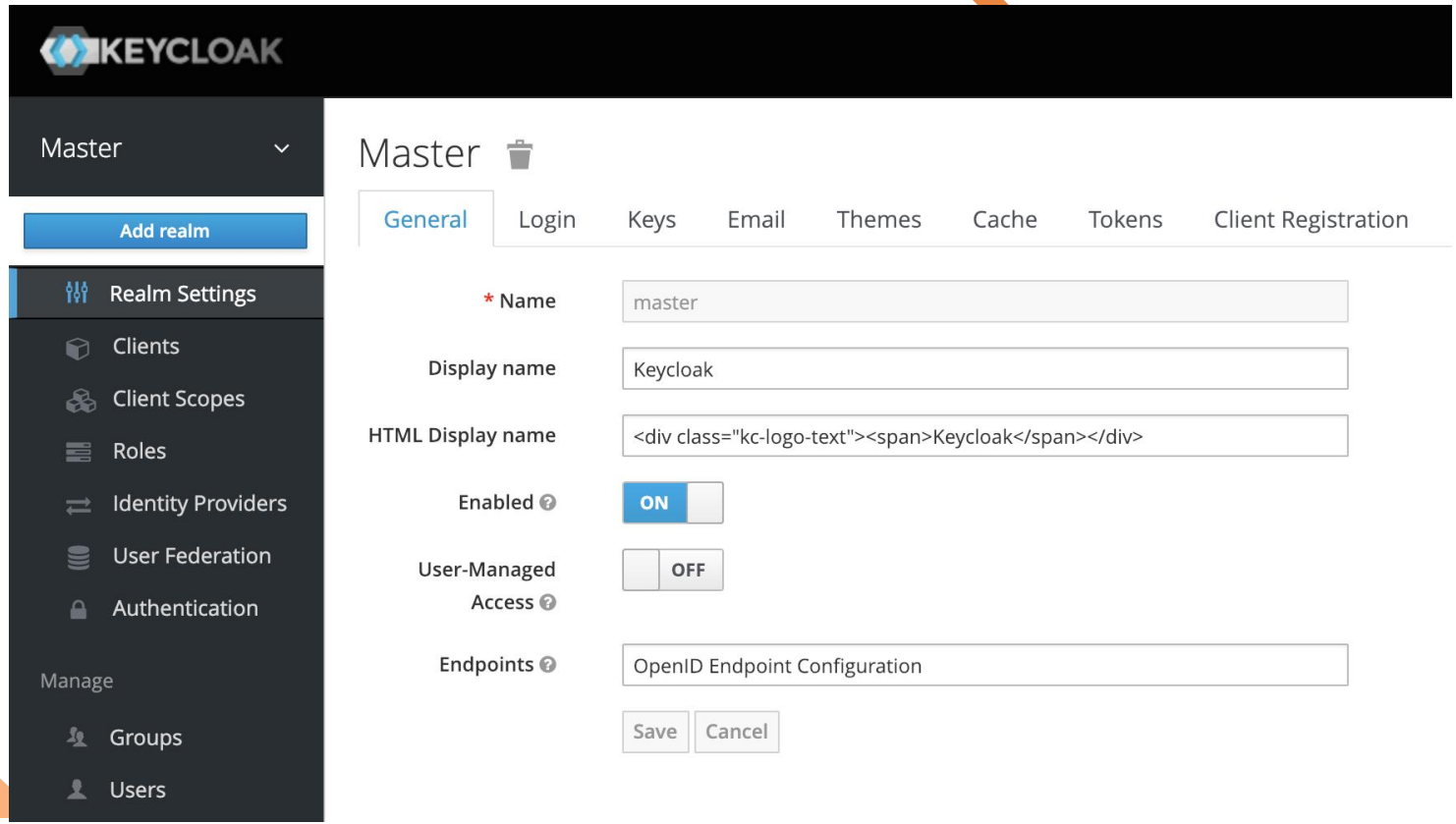
Access Control



Access Control



JWT Issuer


KEYCLOAK
Master
Add realm
Realm Settings
Clients
Client Scopes
Roles
Identity Providers
User Federation
Authentication
Manage
Groups
Users
Master
General
Login
Keys
Email
Themes
Cache
Tokens
Client Registration
* Name
master
Display name
Keycloak
HTML Display name
Keycloak
Enabled
ON
User-Managed Access
OFF
Endpoints
OpenID Endpoint Configuration
Save
Cancel

Questions?

Thank you..!!!

LinkedIn, GitHub, GitLab, Twitter: @kodtodya

https://kodtodya.github.io/talks/