# Developing Microservices with Istio Service Mesh

Kevin Conner Engineering Manager Istio Engineering Red Hat



#### MICROSERVICES ARCHITECTURE



#### MICROSERVICES ARCHITECTURE DISTRIBUTED





#### DISTRIBUTED ARCHITECTURE





- 1. The network is reliable
- 2. Latency is zero
- 3. Bandwidth is infinite
- 4. The network is secure
- 5. Topology doesn't change
- 6. There is one administrator
- 7. Transport cost is zero
- 8. The network is homogeneous

# HOW TO DEAL WITH THE COMPLEXITY?

Photo by <u>Clint Adair</u> on <u>Unsplash</u>



Difficulty of deploying microservices

- Service discovery
- Retries
- Timeouts
- Circuit breaking
- Rate limiting
- Load balancing



Difficulty of deploying on cloud infrastructure

- Edge / DMZ routing (ingress/egress)
- Surgical / fine / per-request routing
- A/B testing
- Traffic shaping
- Internal release / dark launch
- Request shadowing
- Fault injection



Difficulty of deploying on cloud infrastructure

- Adaptive, zone-aware load balancing
- Health checking (active/passive)
- Stats/metrics/telemetry
- Logging
- Distributed tracing



# Security!!



## WHAT DO DISTRIBUTED SYSTEMS NEED?



## AN OPERATION NIGHTMARE!

# Infra capabilities are tightly coupled with applications and services

Incompatible across languages and frameworks
 Existing apps require refactoring
 Upgrades needs tight coordination - libraries





# THERE SHOULD BE A BETTER WAY



## DISTRIBUTED SERVICES WITH KUBERNETES





#### DISTRIBUTED SERVICES WITH OPENSHIFT





## DISTRIBUTED SERVICES WITH ISTIO SERVICE MESH





## USE A SERVICE PROXY

- Envoy Proxy (used in Istio as Istio proxy)
- L3/4 network filter, out of the box L7 filters (HTTP, HTTP2, gRPC)
- Service discovery, load balancing, circuit breaking, metrics collection, timeouts, retries, rate limiting, distributed tracing, et al
- Written in C++
- Dynamic configuration





## SIDECAR PATTERN

- A utility container in the same pod to enhance the main container's functionality
- Share the same network and lifecycle
- Istio uses an Istio Proxy sidecar to proxy all network traffic between apps





WHAT DOES ISTIO GIVE ME?

- Service resiliency
- Observability
- Traffic control
- Security
- Policy Enforcement
- Chaos Testing



#### Istio Service Mesh



\_\_\_\_\_





# **Istio In Action**

# **OBSERVABILITY**



## KIALI - SERVICE MESH OBSERVABILITY



Endpoints

172.17.0.19 : recommendation-v1-b87789c58-zfmnr

172.17.0.22 : recommendation-v2-6/64/9c5b-1949v

Info

Matrice Traces

@ recommendation

Istic Sidecar Deployed Created at Invalid Date Resource Version Pods (2) Deployme

app reco

Created at: 2018-04-16 10:37:34 Created by: recommodition-v1-8677/180:08 (Regicia.Set) Istile Init containens: isto-init (gar:lostio-release/proxy, init:1.0.0) Istile containens: isto-proxy (gar:lostio-release/proxy/2:1.0.0) recommendation-v2-6f84f9c5b-1949v (1 reolica)

app recommenda

Created at: 2018-08-16 11:13:54 Created by: recommendation-v2-8/64/9c5b (ReplicaSet) Istio init containers: isto-init (gcr.iofsto-release/proxy\_init:1.0.0) Istio containers: isto-prox(gcr.iofsto-release/proxy-2:1.0.0)

Ports

recommendation-v1-b87789c58-zfmnr (1 replica)

Deployments (2) Source Services (3) Virtual Services (1) Destination Rules (1)

on pod-template-hash 843345714 version v1

ion pod-template-hash 292065716 version v2

#### FEATURES:

C

Health

 $\odot$ 

- Service graph representation
- Distributed tracing (via Jaeger)
- Metrics collection and graphs (from Prometheus)
- Configuration validation
- Health computation/display
- Service discovery

Namespace × Filter by Namespace × Namespace × I 2 Rate Interval: Last 10 minutes ×			
ctive Filters: Namespace: tutorial × Clear All Filter	S		
Customer	Health: <u>/</u>	Error Rate: 0.10%	
Preference	Health: 📐	Error Rate: 0.10%	
recommendation	Health: 🔗	Error Rate: 0.00%	

# TRACING



Jarger Ul Lookup by Trace ID... Search Dependencies ~ customer: customer:tutorial.svc.cluster.local:8080/\*

Trace Start: August 16, 2018 3:05 PM Duration: 7.41ms Services: 3 Depth: 9 Total Spans: 9

Service & Operat

customer gebute
 customer (et)

preference -



# TRAFFIC CONTROL



## CANARY DEPLOYMENT WITH ISTIO





## WEIGHTED ROUTING WITH ISTIO





## DARK LAUNCHES WITH ISTIO



# SERVICE SECURITY

# SECURE COMMU

## SECURE COMMUNICATION WITH ISTIO



# mutual TLS authentication, transparent to the services



## CONTROL SERVICE ACCESS WITH MUTUAL TLS



# control the service access flow, transparent to the services

# CONTROL SERVICE ACCESS WITH END USER AUTHENTICATION



# control the service access flow, transparent to the services

# FAULT TOLERANCE



## **CIRCUIT BREAKERS WITH ISTIO**



# transparent to the services



## **CIRCUIT BREAKERS WITH ISTIO**



# transparent to the services



## TIMEOUTS AND RETRIES WITH ISTIO



# configure timeouts and retries, transparent to the services



## **RATE LIMITING WITH ISTIO**



# limit invocation rates, transparent to the services

# **CHAOS ENGINEERING**



## CHAOS ENGINEERING WITH ISTIO



# inject delays, transparent to the services



## CHAOS ENGINEERING WITH ISTIO



# inject protocol-specific errors, transparent to the services

# RED HAT OPENSHIFT SERVICE MESH

## **OPENSHIFT SERVICE MESH**

- Supported distribution of
  - Istio
  - Jaeger
  - Kiali
  - Prometheus
  - Grafana
- Upstream project called Maistra
  <u>https://github.com/Maistra</u>
- Integrated with Red Hat OpenShift Application Runtimes (RHOAR)
- Integrated with Red Hat 3scale API Management

spec	ification that describes what your application will do.
1	- And the second stream of the constraints of the second stream of th
0	Istio - Circuit Breaker
	The Istio Circuit Breaker mission demonstrates limiting access to More
0	Istio - Security
	Ths Istio Security mission demonstrates how Istio secures communi More
0	Istio - Distributed Tracing
	The Istio Distributed Tracing mission demonstrates how simple dis More

# RESOURCES



#### O'REILLY'

# Microservices

A Hands-On Introduction to Frameworks & Containers



Christian Posta



<u>bit.ly/</u>istio-book

Build and Deploy Resilient, Fault-Tolerant Cloud-Native Applications



Christian Posta & Burr Sutter

bit.ly/reactivemicroservi cesbook



Asynchronous and Event-Based Application Design



**Clement Escoffier** 

#### https://www.manning.co m/books/istio-in-action



Demo: <u>bit.ly/msa-instructions</u> Slides: <u>bit.ly/microservicesdeepdive</u> Video Training: <u>bit.ly/microservicesvideo</u>

## bit.ly/mono2microdb

Kubernetes for Java Developers



## WORKSHOPS

# Self-paced Labs learn.openshift.com/servicemesh

# GitHub

# bit.ly/istio-tutorial

https://github.com/thoraxe/istio-lab-summit-2019



# **Questions?**