# A Crash Course in Service Mesh Solutions



A beginner's guide

@melissajmckay


JFrog

London Microservices

http://bit.ly/AirPodsLDNMicroservices

slides → jfrog.com/shownotes

# Melissa McKay

- Mom
- Developer
- Developer Advocate @ JFrog
- Self appointed **UN**Conference Advocate and Promoter:

***JCrete*** (http://www.jcrete.org/)

***JOnsen*** (http://jonsen.jp/)

***JSpirit*** (https://jspirit.org)

***JAlba*** (https://jalba.scot)

***LavaOne/UnVoxxed Hawaii*** (https://voxxeddays.com/hawaii/)

# Simple Rules

To maximize what you can get out of the unconference, simple rules apply. Wikipedia summarizes them nicely:

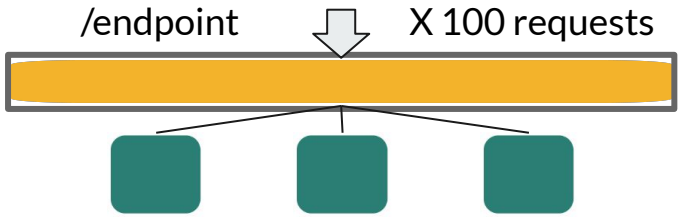
1. **Whoever shows up are the right people**
   …reminds participants that they don't need the CEO and 100 people to get something done, you need people who *care.* And, absent the direction or control exerted in a traditional meeting, that's who shows up in the various breakout sessions of an Open Space meeting.
2. **Whenever it starts is the right time**
   …reminds participants that spirit and creativity do not run on the clock.
3. **Whatever happens is the only thing that could have**
   …reminds participants that once something has happened, it's done—and no amount of fretting, complaining or otherwise rehashing can change that. Move on.
4. **Wherever it happens is the right place**
   …reminds participants that space is opening everywhere all the time. Please be conscious and aware.
5. **When it's over, it's over**
   …reminds participants that we never know how long it will take to resolve an issue, once raised, but that whenever the issue or work or conversation is finished, move on to the next thing. Don't keep rehashing just because there's 30 minutes left in the session. Do the work, not the time.

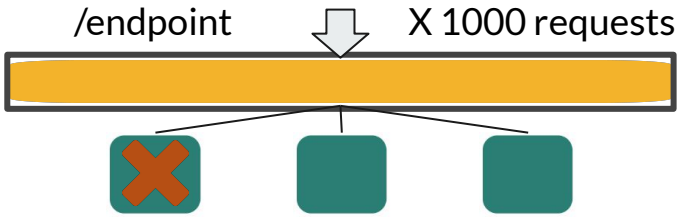
http://www.jcrete.org/what-is-an-unconference_/

# What am I going to get out of this?

- **Understand the concepts** behind a service mesh
- **Learn key differentiators** between solutions (Linkerd, Istio)
- **Develop an educated opinion**

# How did I get here?

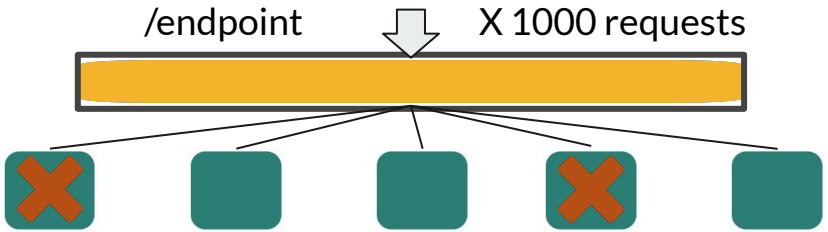



Some history...

- A mis-behaving service
- Missing SLAs

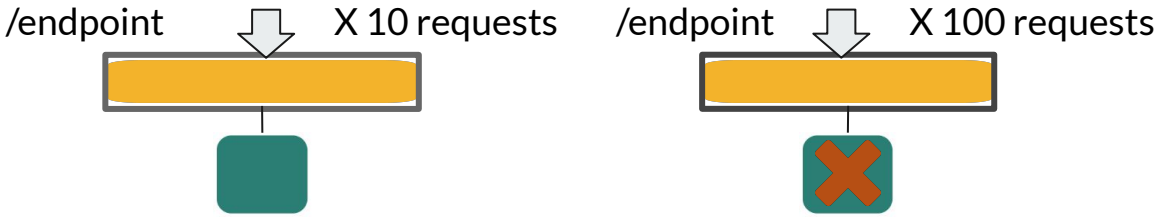
# How did I get here?




Some history...

- A mis-behaving service
- Missing SLAs
- Replicating the service under a load balancer did **NOT** solve the problem!

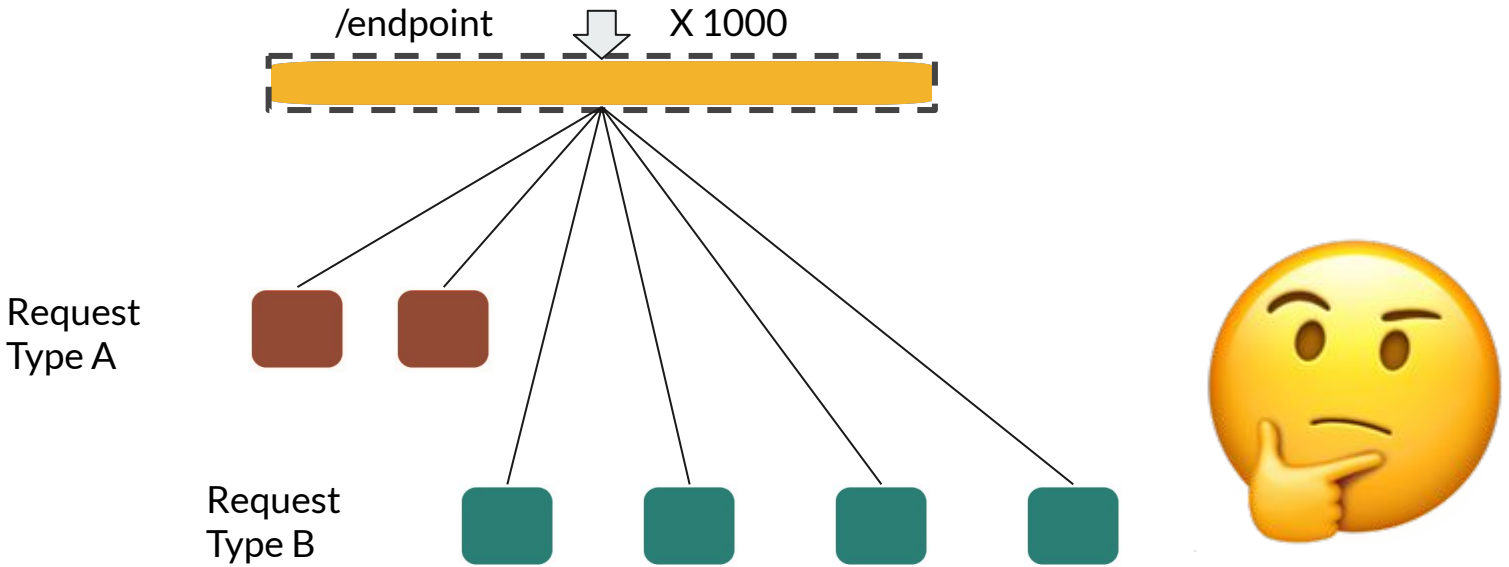
# How did I get here?



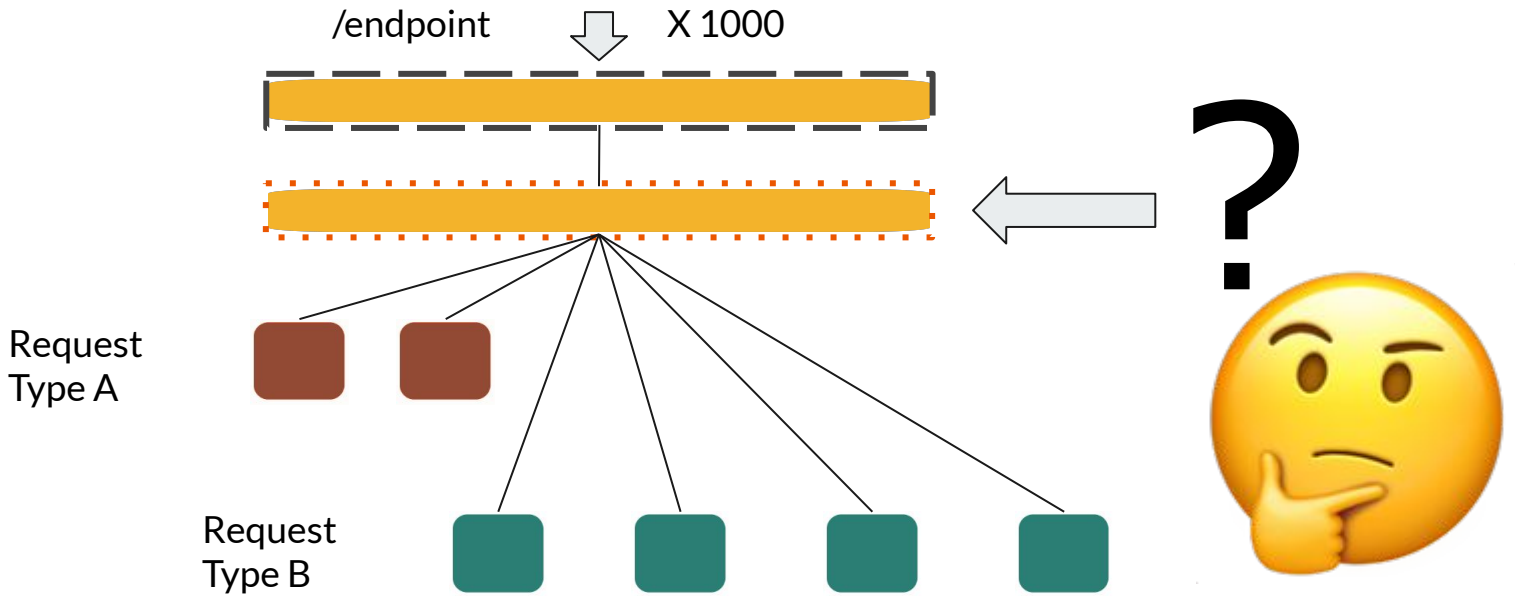

Some history...

- A mis-behaving service
- Missing SLAs
- Replicating the service under a load balancer did **NOT** solve the problem!
- Analyzed, determined issue & did some research on best way to solve...

# Architectural Solution



# Architectural Solution

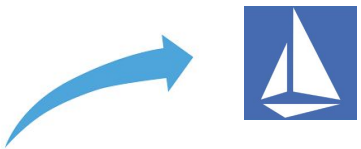


# An API Gateway?

- Filter options for routing
- Enabled rolling upgrades
- Enabled traffic shifting & rate limiting
- Ability to canary nodes
- Blue-green deployment

# An API Gateway?

- Filter options for routing
- Enabled rolling upgrades
- Enabled traffic shifting & rate limiting
- Ability to canary nodes
- Blue-green deployment



*But what's this?*

Tyk.io

Kong

express gateway

Amazon API Gateway

# Istio… I've heard of that!

14
FEB

Wednesday, February 14, 2018

## Ray Tsang: Making Microservices Micro with Istio Service Mesh & Kubernetes

Hosted by Denver Java Users Group

Public group ?

# How is it different?

by Zach Jory MVB · Sep. 19, 18 · Microservices Zone · Opinion

| API Gateway | Service Mesh |
|---|---|
| Handles north-south traffic | Handles east-west traffic |
| Exposes external services to make them easily consumable | Manages and controls service inside your network |
| Maps external traffic to internal resources | Focuses on brokering internal resources |
| Exposes APIs or edge services to serve a specific business function | Sits between the network and application, no real business notion of your solution. |

"They can both handle service discovery, request routing, authentication, rate limiting, and monitoring, but there are differences in architectures and intentions. *A service mesh's primary purpose is to manage internal service-to-service communication*, while an API Gateway is primarily meant for external client-to-service communication."

https://dzone.com/articles/api-gateway-vs-service-mesh

# How is it different?

by Zach Jory MVB · Sep. 19, 18 · Microservices Zone · Opinion

What **IS** this?

| API Gateway | Service Mesh |
|---|---|
| Handles north-south traffic | Handles east-west traffic |
| Exposes external services to make them easily consumable | Manages and controls service inside your network |
| Maps external traffic to internal resources | Focuses on brokering internal resources |
| Exposes APIs or edge services to serve a specific business function | Sits between the network and application, no real business notion of your solution. |

"They can both handle service discovery, request routing, authentication, rate limiting, and monitoring, but there are differences in architectures and intentions. *A service mesh's primary purpose is to manage internal service-to-service communication*, while an API Gateway is primarily meant for external client-to-service communication."

https://dzone.com/articles/api-gateway-vs-service-mesh

**A service mesh is a dedicated infrastructure layer that controls service-to-service communication over a network.**

https://searchitoperations.techtarget.com/definition/service-mesh - Margaret Rouse, Alex Gillis, WhatIs.com (January, 2019)

**A service mesh is a configurable, low-latency infrastructure layer designed to handle a high volume of network-based interprocess communication among application infrastructure services using application programming interfaces (APIs).**

https://www.nginx.com/blog/what-is-a-service-mesh/ - Floyd Smith & Owen Garrett, NGINX (April, 2018)

**tl;dr: A service mesh is a dedicated infrastructure layer for making service-to-service communication safe, fast, and reliable.**

https://buoyant.io/2017/04/25/whats-a-service-mesh-and-why-do-i-need-one/ - William Morgan, Buoyant (April, 2017)

# A service mesh brings security, resiliency, and visibility to service communications, so developers don't have to

https://www.infoworld.com/article/3402260/what-is-a-service-mesh-service-mesh-explained.html - Josh Fruhlinger, InfoWorld (July, 2019)

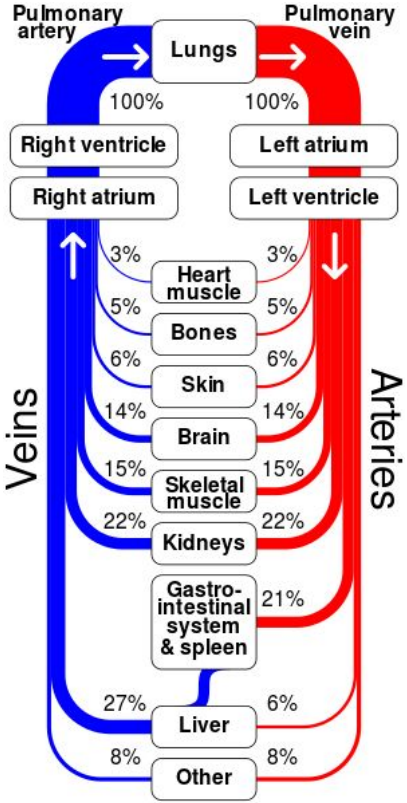
**The term service mesh is used to describe the network of microservices that make up such applications and the interactions between them.**

https://istio.io/docs/concepts/what-is-istio/ - Istio (accessed September, 2019)

# So what is a service mesh, really?

A service mesh is a separately managed distributed system that handles common functions required and normally implemented by services that do not concern the business logic of the service itself.

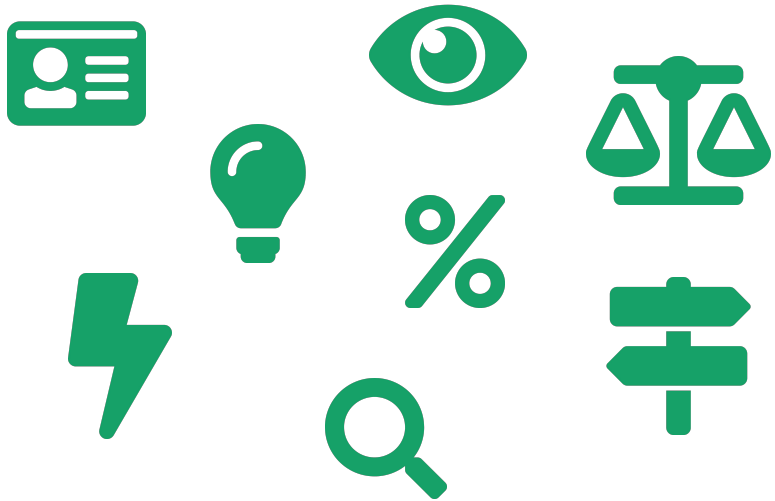
A real-life metaphor…. human circulatory system?




https://en.wikipedia.org/wiki/Circulatory_system

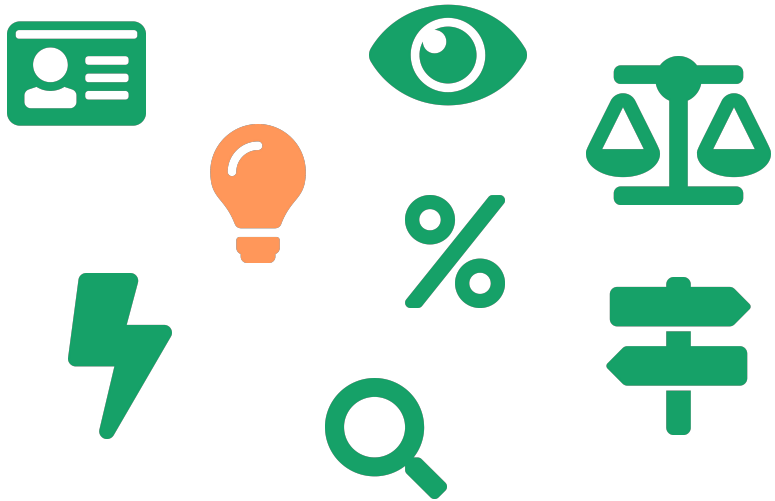
# Service Mesh Capabilities... and more coming!

- Service Discovery
- Observability
- Rate Limiting
- Circuit Breaking
- Traffic Shifting
- Load Balancing
- Authorization/Authentication
- Distributed Tracing



# Service Mesh Capabilities… and more coming!

- **Service Discovery**
- Observability
- Rate Limiting
- Circuit Breaking
- Traffic Shifting
- Load Balancing
- Authorization/Authentication
- Distributed Tracing

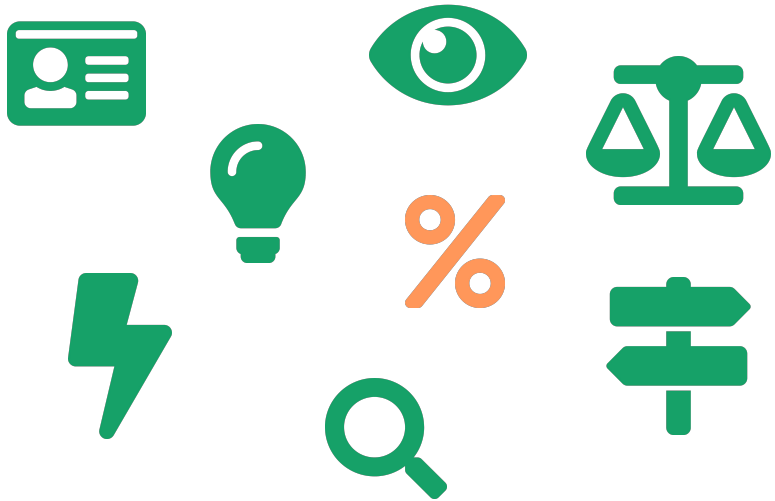


# Service Mesh Capabilities... and more coming!

- Service Discovery
- **Observability**
- Rate Limiting
- Circuit Breaking
- Traffic Shifting
- Load Balancing
- Authorization/Authentication
- Distributed Tracing

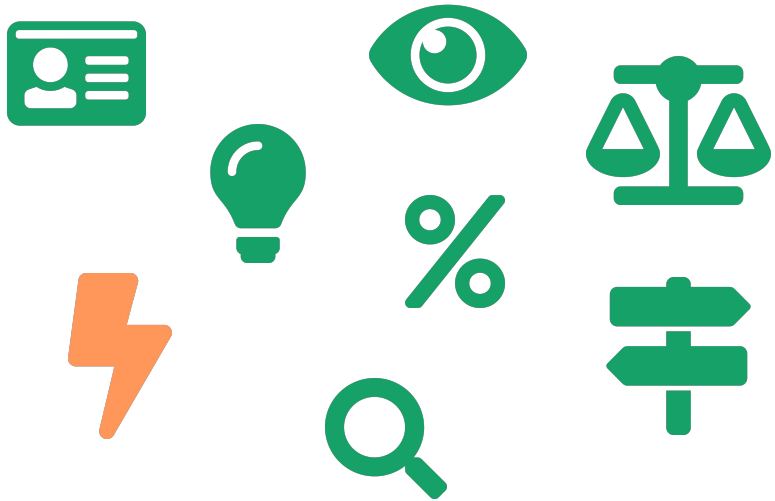
# Service Mesh Capabilities… and more coming!

- Service Discovery
- Observability
- **Rate Limiting**
- Circuit Breaking
- Traffic Shifting
- Load Balancing
- Authorization/Authentication
- Distributed Tracing



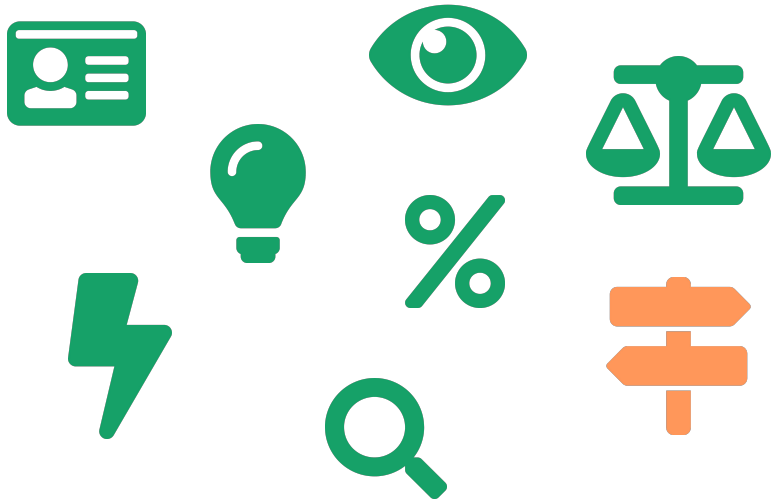
# Service Mesh Capabilities… and more coming!

- Service Discovery
- Observability
- Rate Limiting
- **Circuit Breaking**
- Traffic Shifting
- Load Balancing
- Authorization/Authentication
- Distributed Tracing



# Service Mesh Capabilities… and more coming!

- Service Discovery
- Observability
- Rate Limiting
- Circuit Breaking
- **Traffic Shifting**
- Load Balancing
- Authorization/Authentication
- Distributed Tracing



# Service Mesh Capabilities... and more coming!

- Service Discovery
- Observability
- Rate Limiting
- Circuit Breaking
- Traffic Shifting
- **Load Balancing**
- Authorization/Authentication
- Distributed Tracing

# Service Mesh Capabilities… and more coming!

- Service Discovery
- Observability
- Rate Limiting
- Circuit Breaking
- Traffic Shifting
- Load Balancing
- **Authorization/Authentication**
- Distributed Tracing

# Service Mesh Capabilities... and more coming!

- Service Discovery
- Observability
- Rate Limiting
- Circuit Breaking
- Traffic Shifting
- Load Balancing
- Authorization/Authentication
- **Distributed Tracing**

# Service Mesh Capabilities… and more coming!

- Service Discovery
- Observability
- Rate Limiting
- Circuit Breaking
- Traffic Shifting
- Load Balancing
- Authorization/Authentication
- Distributed Tracing

**… legitimate solutions to all kinds of problems!**

# Why isn't everyone using a service mesh???

"If you're wondering about service mesh, you don't need one. Period. If you've reached the scale and microservice maturity level that requires a service mesh, you will be actively — perhaps desperately — searching for a solution and it will be abundantly obvious that a service mesh is necessary." https://thenewstack.io/primer-the-who-what-and-why-of-service-mesh/ - *Emily Omier (May, 2019)*

"I think we have a tendency to chase the shiny object, in the sense that X company does Y, therefore I must do Y, even though I don't have any of X company's problems." - *Matt Klein*

# Service Mesh Capabilities… and more coming!

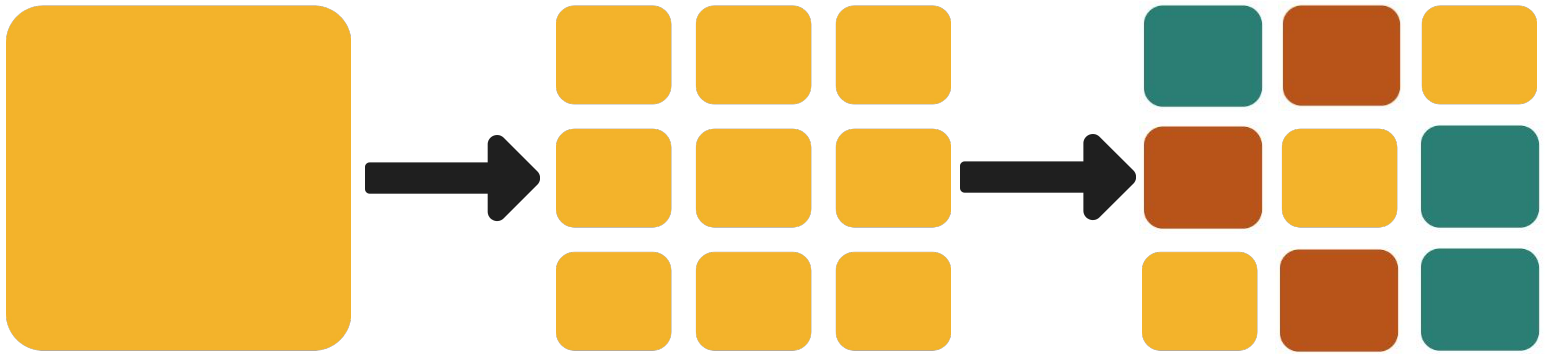
- Service Discovery
- Observability
- Rate Limiting
- Circuit Breaking
- Traffic Shifting
- Load Balancing
- Authorization/Authentication
- Distributed Tracing

**… legitimate solutions to all kinds of problems!**

# Back up… what problem are we trying to solve?

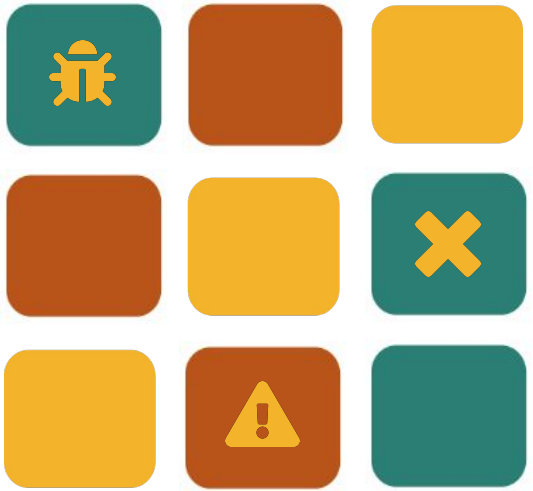
Matt Klein - Lyft engineer who started Envoy Proxy

Podcast:  https://softwareengineeringdaily.com/2017/02/14/service-proxying-with-matt-klein/



# Biggest Problems?:  I vote for Observability & Reliability

- Large numbers of services
- Diverse/Polyglot
- Different Communication Protocols
- Service/language specific libraries
- No standardization on logging or stats



# How is a service mesh implemented?

DATA PLANE

This is the part that touches every request in the system.

Sidecar proxies.

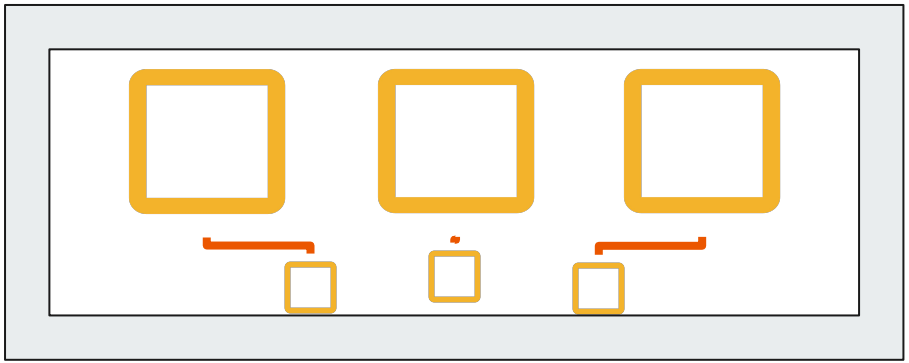
- All service communication (ingress & egress) routed through proxies
- The proxy acts as a gateway to the service

# How is a service mesh implemented?

CONTROL PLANE

This is the part that manages the data planes - providing them with the data and configuration needed by the system.

UI, CLI or some other interface where an operator can set configuration settings.



# I'm ready to try it!  Where do I start?

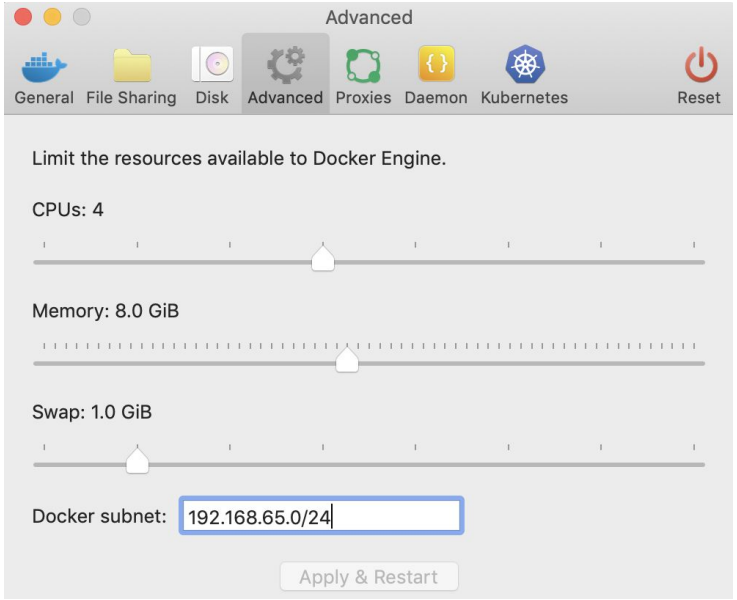
ENVOY: ~~https://www.envoyproxy.io/learn/~~ https://www.envoyproxy.io/docs/envoy/latest/start/start

LINKERD: https://linkerd.io/2/getting-started/

ISTIO: https://istio.io/docs/setup/install/kubernetes/

# Prereqs to play

- Docker
- Docker Compose
- Kubernetes basics
- Helm charts/templates
- Get used to YAML if you aren't already




Pay attention to versioning, of course!
Definitely up your CPUs to 4 and memory to 8 GiB if you use Docker Desktop.

# 50 days from zero to hero with Kubernetes

Microsoft Azure

Click the bubbles to access that resource

**Day 1**
**A Kubernetes Story: Phippy Goes to the Zoo**

Tag along with Phippy and Zee as Phippy explains the basics of Kubernetes. *6 minutes.*

**Day 6-15**
**Azure Kubernetes Service core concepts**

Learn about the core concepts of AKS, like infrastructure components and access, in six short articles. *41 minutes.*

**Day 21-25**
**Microservices architecture design**

Learn about the infrastructure and DevOps considerations involved in running a microservices architecture on Azure Kubernetes Service via this article. *21 minutes.*

**Day 28-35**
**Distributed system patterns and practices**

Download this complimentary e-book and learn how to develop reliable distributed systems more easily and efficiently. *Free e-book.*

Zero

Hero

**Day 2-5**
**Kubernetes basics: video series with Brendan Burns**

Join Kubernetes co-founder Brendan Burns for six short videos that answer your general Kubernetes questions. *40 minutes.*

**Day 16-20**
**Explore Kubernetes with Katacoda**

Explore essential tools and capabilities of Kubernetes in these six interactive Katacoda training modules. *60-85 minutes.*

**Day 26-27**
**Azure Kubernetes Service workshop**

Go through a step-by-step workshop to learn how to create a Kubernetes cluster, set up a CI/CD pipeline, and more. *1 workshop.*


**Day 36-50**
**Operational best practices for Kubernetes**

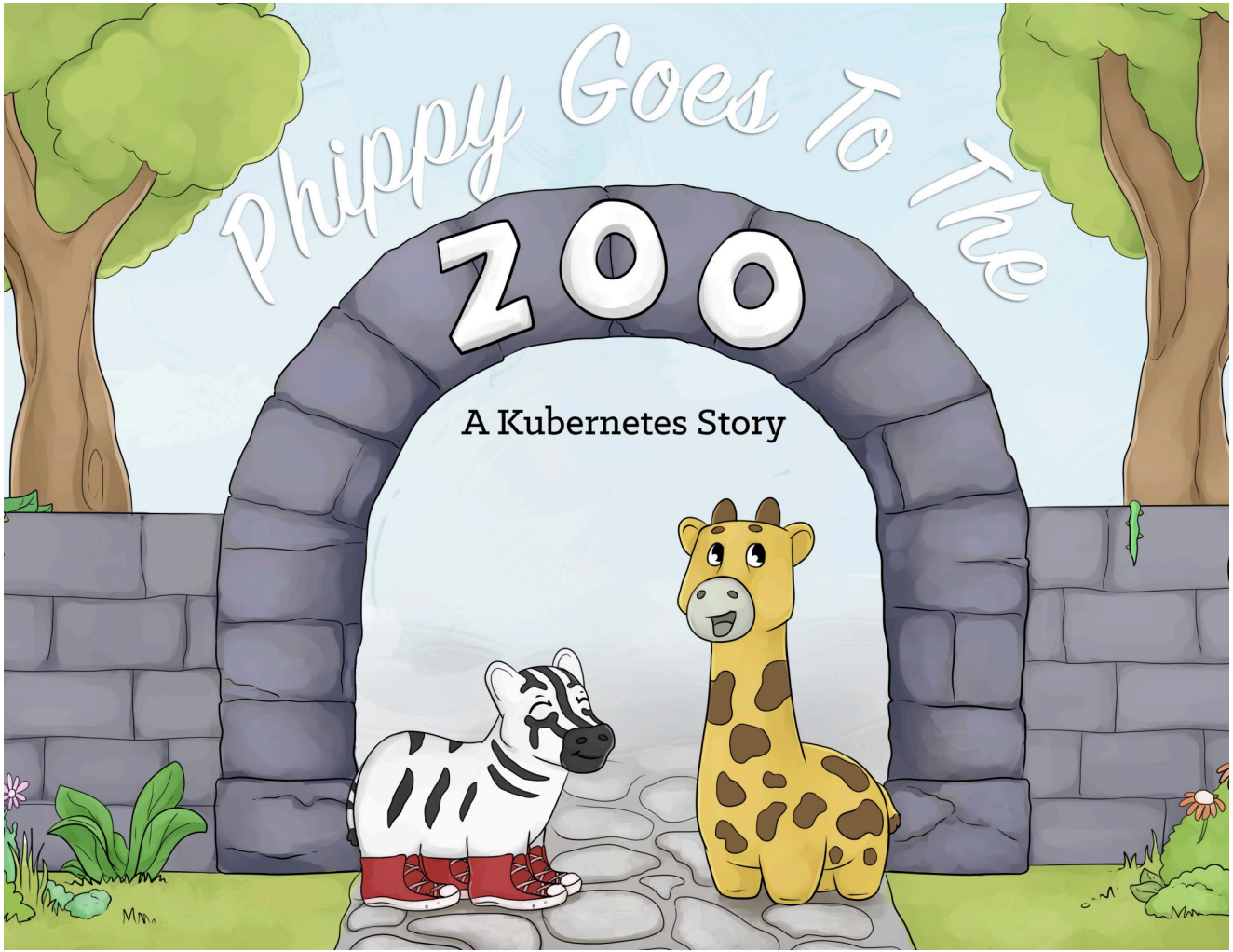
With a series of 11 articles and one webinar developed by subject matter experts, go beyond Day 1 with best practices for Day 2 operations. *61 minutes to read; 64 minutes to watch.*

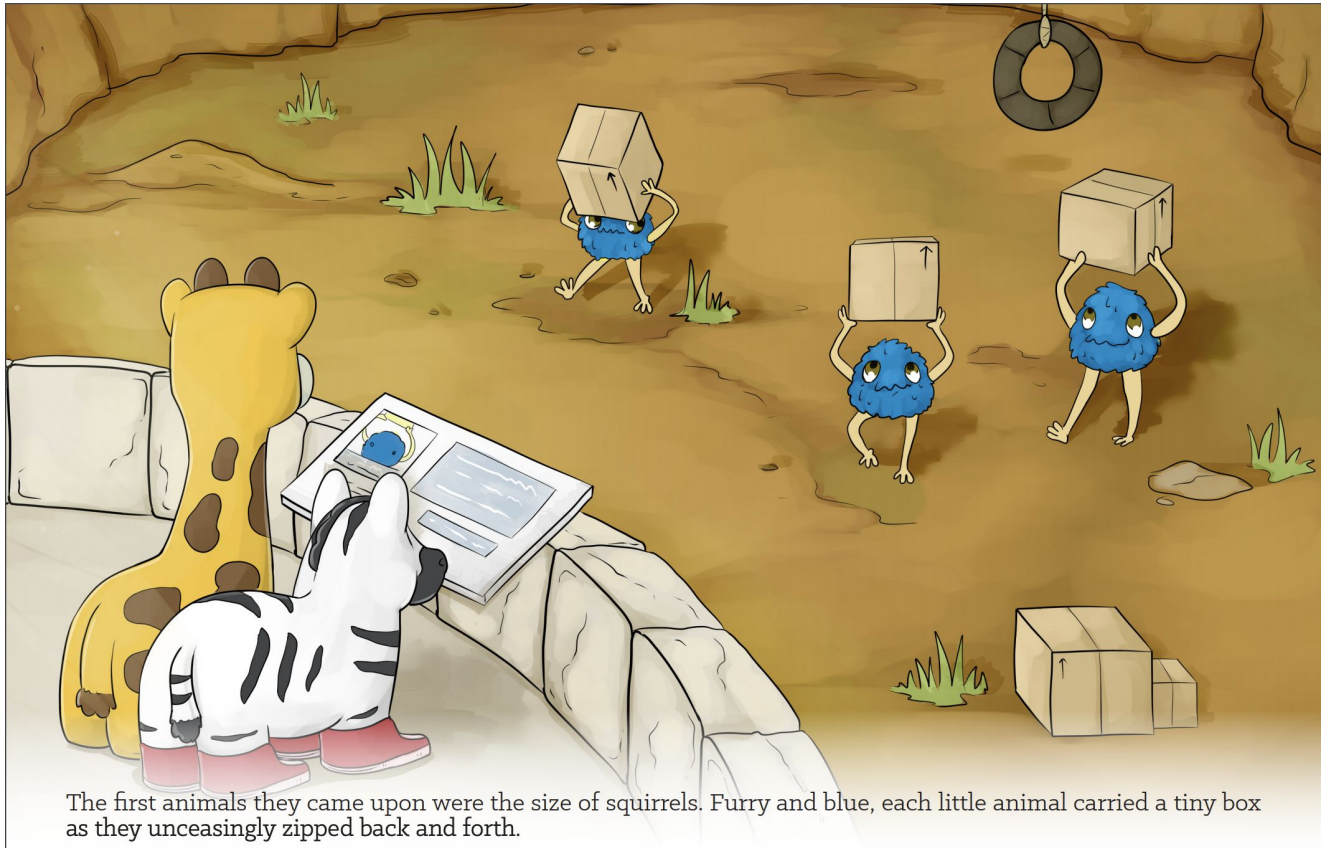
https://azure.microsoft.com/en-us/resources/kubernetes-learning-path/

Brought to you by...


and


Written by: Matt Butcher & Karen Chu
Illustrated by: Bailey Beougher
Designed by: Karen Chu


Phippy Goes To The
ZOO
A Kubernetes Story

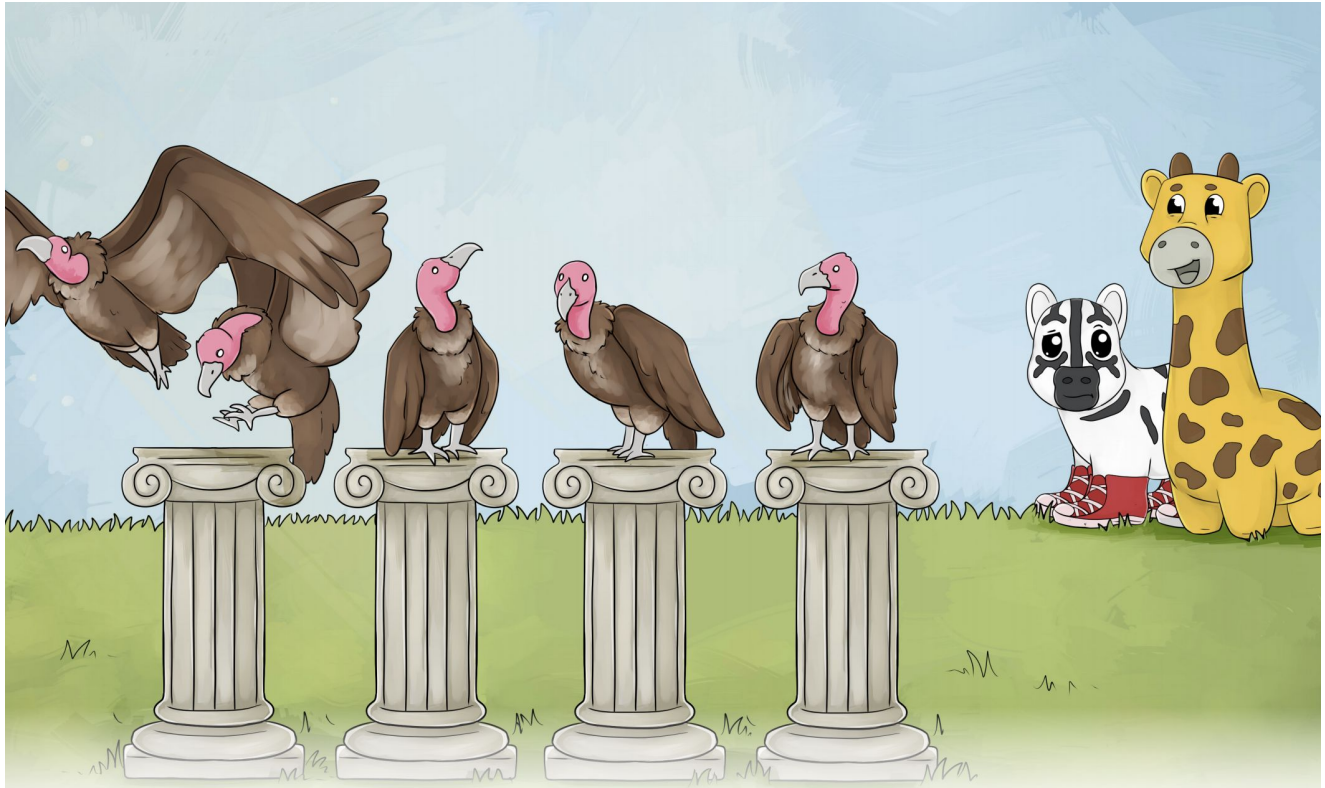


The first animals they came upon were the size of squirrels. Furry and blue, each little animal carried a tiny box as they unceasingly zipped back and forth.

“Those,” said Phippy, “are Pods. All day and all night, they run back and forth carrying their little containers.”

“Is that all they do, Aunt Phippy?”

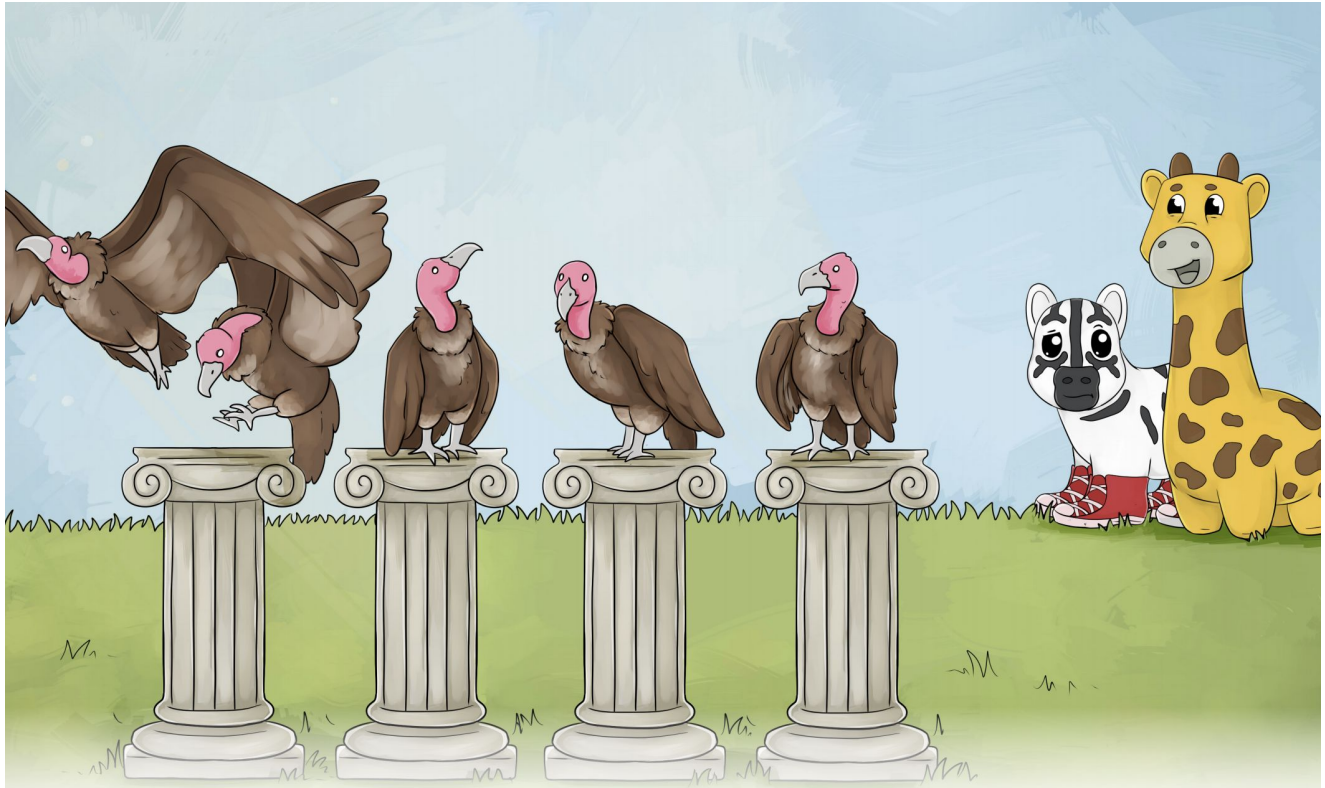
“Yup, Zee. For their entire lives, that’s all the Pods do. They run.”



Several stone pillars arose from a grassy knoll and at the top of each sat a vulture. As Zee and Phippy watched, one vulture spread its wings and flapped off into the distance. No sooner had one left than another took its place. Zee asked, "What are they doing?"

"Those are DaemonSets," said Phippy, "They make sure to occupy every pillar, rain or shine, day or night."

"I bet that if we added a new pillar, a new bird would land on it faster than you could say **kubectl** chuckled Phippy.



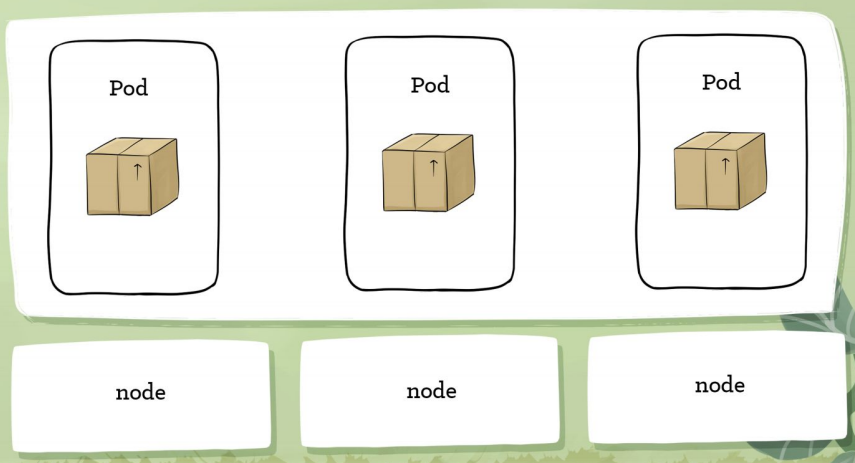
Several stone pillars arose from a grassy knoll and at the top of each sat a vulture. As Zee and Phippy watched, one vulture spread its wings and flapped off into the distance. No sooner had one left than another took its place. Zee asked, "What are they doing?"

"Those are DaemonSets," said Phippy, "They make sure to occupy every pillar, rain or shine, day or night."

"I bet that if we added a new pillar, a new bird would land on it faster than you could say 'cube cuddle,'" chuckled Phippy.

# DaemonSets

- DaemonSets have many uses – one frequent pattern is to use a DaemonSet to install or configure software on each host node




DaemonSets provide a way to ensure that a copy of a Pod is running on every node in the cluster. As a cluster grows and shrinks, the DaemonSet spreads these specially labeled Pods across all of the nodes.

ISTIO: https://istio.io/docs/setup/install/kubernetes/

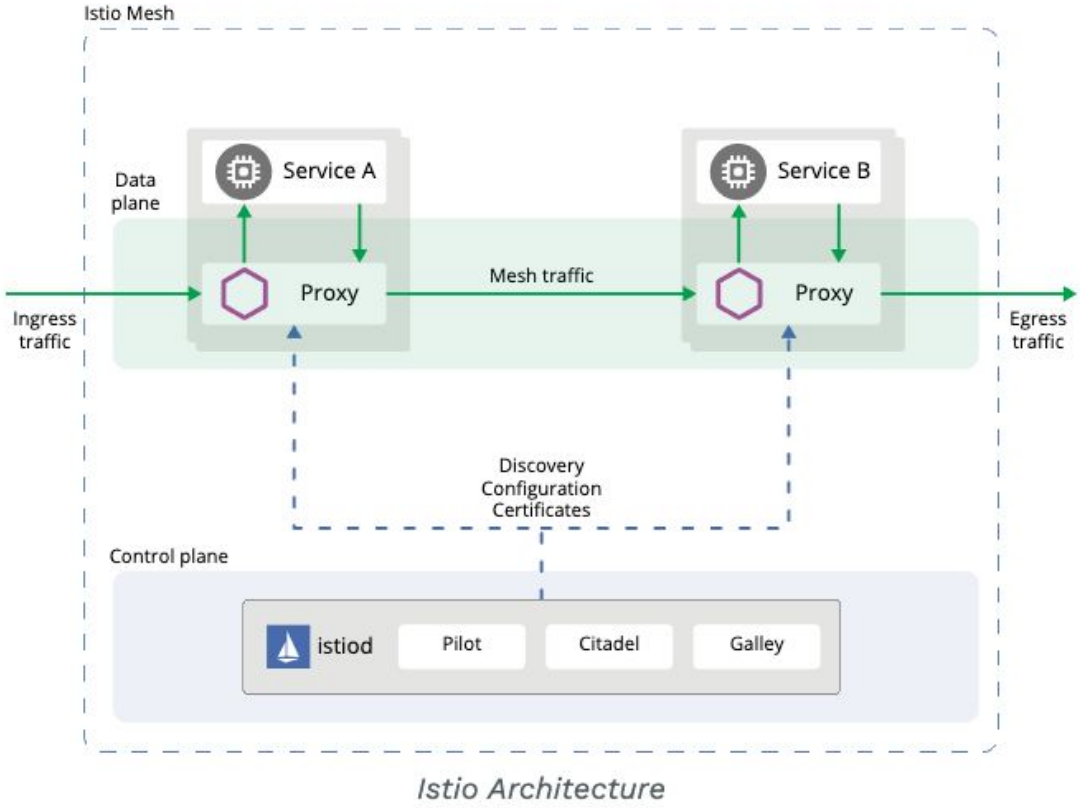
# Istio

- Go
- Apache 2.0 license
- Designed for extensibility, but might come at the cost of complexity
- Modular, pluggable
- Supports HTTP 1.1, HTTP2, gRPC, and TCP
- Support for Kubernetes, VMs

- Backed by Google, RedHat & IBM
- The quick install was easy, but choosing another type of install or configuration felt a little like choose your own adventure.
- Great documentation
- There are a TON of online tutorials, etc

**OVER 40 EXAMPLES** available to play with different features!!!

# Istio



*Istio Architecture*

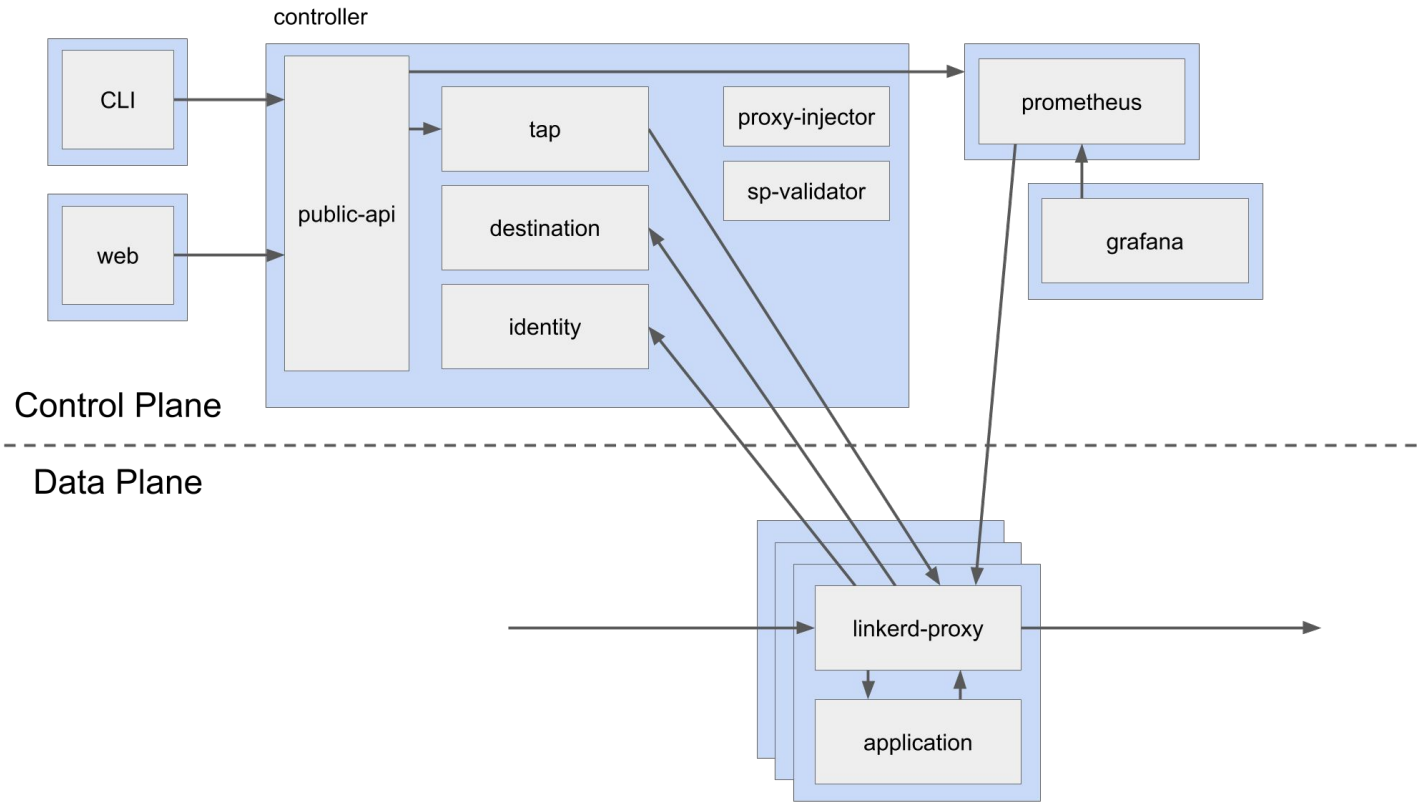
https://istio.io/docs/concepts/what-is-istio/

LINKERD: https://linkerd.io/2/getting-started/

# Linkerd2

- Go/Rust
- Apache 2.0 license
- Supported by Cloud-Native Computing Foundation
- Data & Control Plane tightly integrated (less modular, but smooth)
- VERY easy to install and get it up and running
- Several examples are available to try out key features
- Intended for Kubernetes
- Supports HTTP 1.1, HTTP2, gRPC, and TCP
- Not as feature rich as Istio, but is a very active project (weekly edge releases, 6-8 week stable releases)
- Latest updates (2.9): Distributed tracing, traffic shifting (blue/green, canaries), telemetry, retries, timeouts, proxy auto-injection, mTLS on by default for all TCP
- Excellent documentation

# Linkerd2

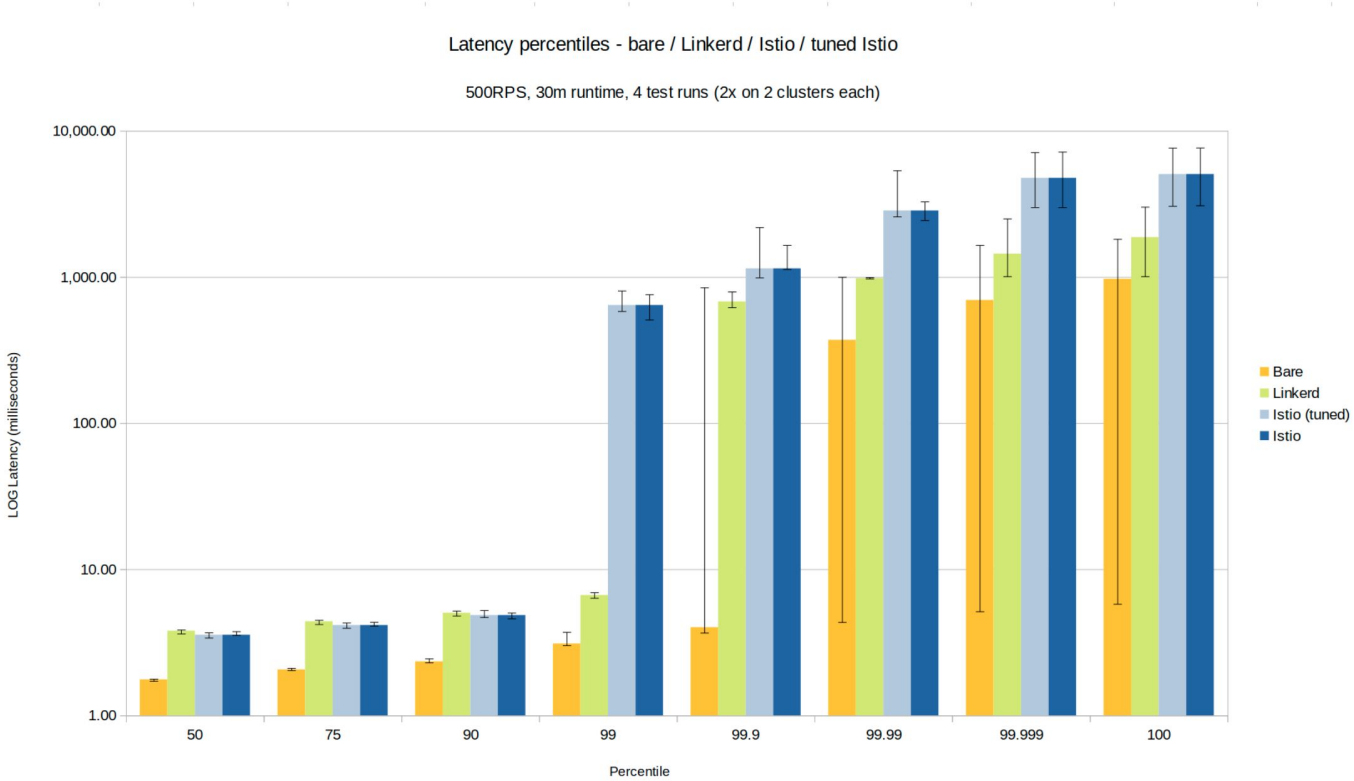



https://linkerd.io/2/reference/architecture/index.html

## Benchmark #1 - 500RPS over 30 minutes

This benchmark was run over 30 minutes, with a constant load of 500 requests per second.

## Latency percentiles




https://kinvolk.io/blog/2019/05/performance-benchmark-analysis-of-istio-and-linkerd/
https://github.com/kinvolk/service-mesh-benchmark/issues/5

# Quick Compare - Istio AND Linkerd2

- Supports Kubernetes
- Apache 2.0 license
- Side Car Pattern Deployment
- Control Plane written in Go
- Supported Protocols - HTTP1.1, HTTP2, gRPC, TCP
- Similar traffic control & monitoring features
- Helm Chart support
- mTLS support

# Quick Compare - Differences

## ISTIO

- Data plane: Envoy (C++), or others (Nginx)
- Higher performance overhead
- Pluggable/Modular

Generally more Complex Setup

## LINKERD2

- Data plane: Native (Rust)
- Lower performance overhead
- Opinionated/Tightly Coupled

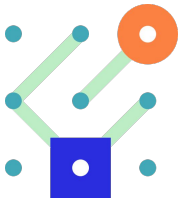
Generally Simple Setup

Comparison Chart: https://dzone.com/articles/service-mesh-comparison-istio-vs-linkerd

# What next?

EXPLORE OTHERS!

This space is growing fast and getting a lot of attention - one might presume this means there is a definite need in the market, so it's definitely worth checking out.



**Service Mesh Interface (SMI)**:
A standard interface for service meshes on Kubernetes.
https://smi-spec.io/

# Conclusion

Choose a solution that addresses REAL problems you need to solve for your system.

Consider your developers.

Consider your codebase.

Consider the performance cost.

Evaluate MULTIPLE solutions - don’t simply jump on a bandwagon.

The whole idea of a service mesh is pretty cool!

London Microservices

# THANK YOU!

JFrog

## Q & A

Don't forget!

http://bit.ly/AirPodsLDNMicroservices