SAMSUNG SDS
Realize your vision

Foresee

Techtonic 2019

Disrupt

Partner

2019.11.14 • SAMSUNG SDS Tower B1F
Magellan Hall / Pascal Hall

Track 4 | **Cloud**

Service Mesh For All

Michael Venezia

AGENDA

1. Microservices and Service Meshes
2. Environment
3. Use Cases
4. Lessons Learned

1

Microservices and Service Meshes

Service Meshes only for Microservices?

- We often find service meshes being discussed alongside microservices. So much so that one could reasonably feel that they are synonymous with each other.

- As a result, some may presume that unless you're deploying microservices you have no need for service meshes and this may be an improper conclusion.

Why Do We Feel This Way?

- Let's see what led up to this conclusion.
- Let's start with our understanding of Microservices

Microservices Defined

Microservices are somewhat ambiguously defined. Wikipedia lists a set of properties that are commonly found in a microservice:

- Communicate over the network (often leveraging the http protocol)
- Independently deployable
- Organized around capabilities
- Written in languages or techniques that best fit the problem being solved
- “Small” in scope

Microservices Are Just Services

Arguably most services fall into this microservice category if you are looking from a 30,000 foot view, as everything at that perspective is “small”

But we generally accept that most legacy services are not microservices - so there must be more to it.

There is a certain je ne sais quoi to identifying a microservice, but you'll find that they often have some gaps:

- In order to stay small, Microservices tend to focus only at the task at hand, deferring boring things like security, rate limiting, identity, etc. to something else
- Microservices tend to have to connect to other microservices, thus concepts of service discovery become important

Service Meshes Solve Microservices’ Problems

- Why do we see service meshes together with microservices?
  - They often solve these boring problems for microservices!
- Quite possibly, service meshes can solve these boring problems for their bigger friends, generic applications/services too!

So what is a service mesh?

Good News

343 Million results in google for "service mesh“

Bad News

2 sentences in Wikipedia

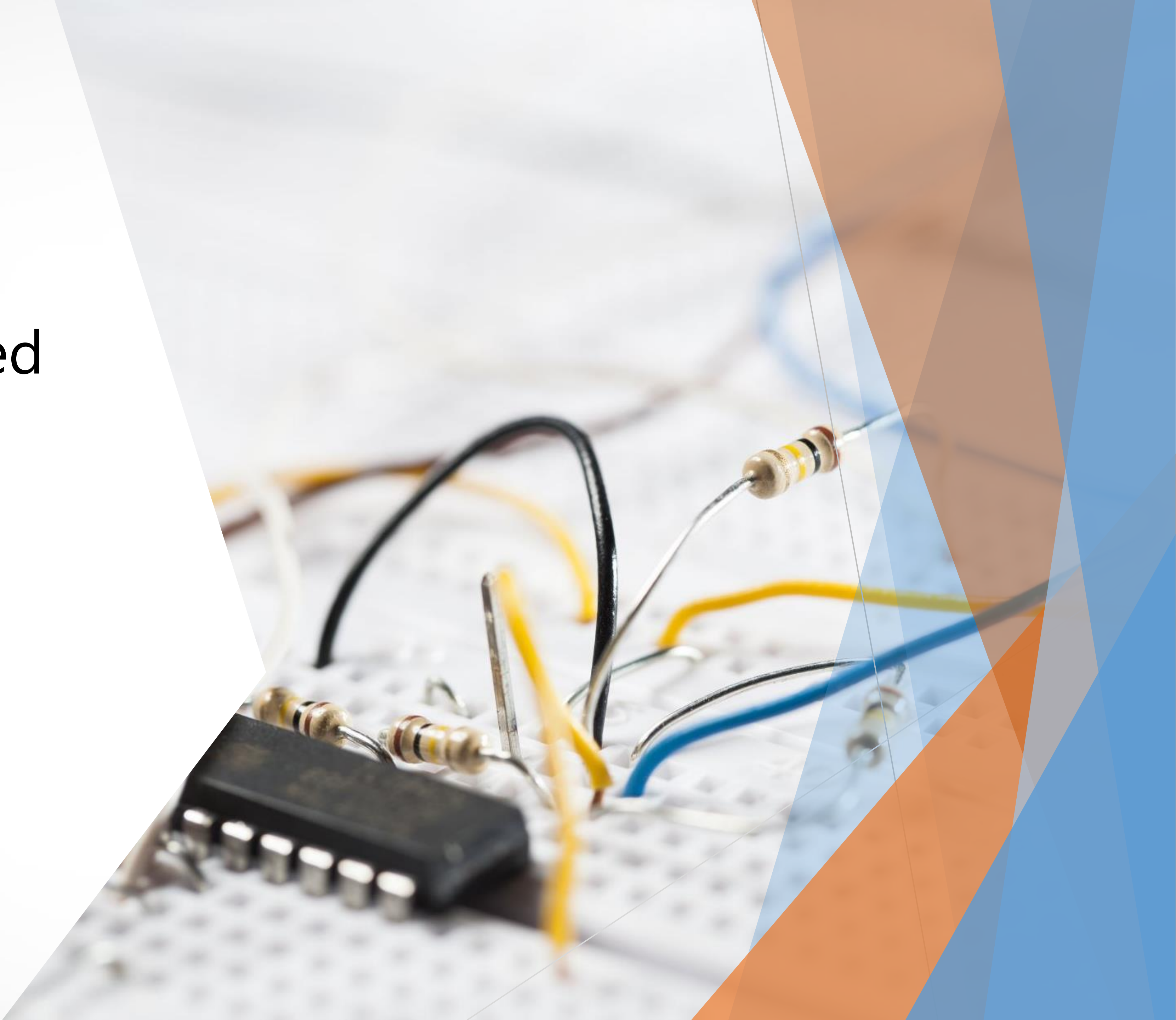
Result

Service Mesh is not very well defined but a focus of attention for many individuals and organizations

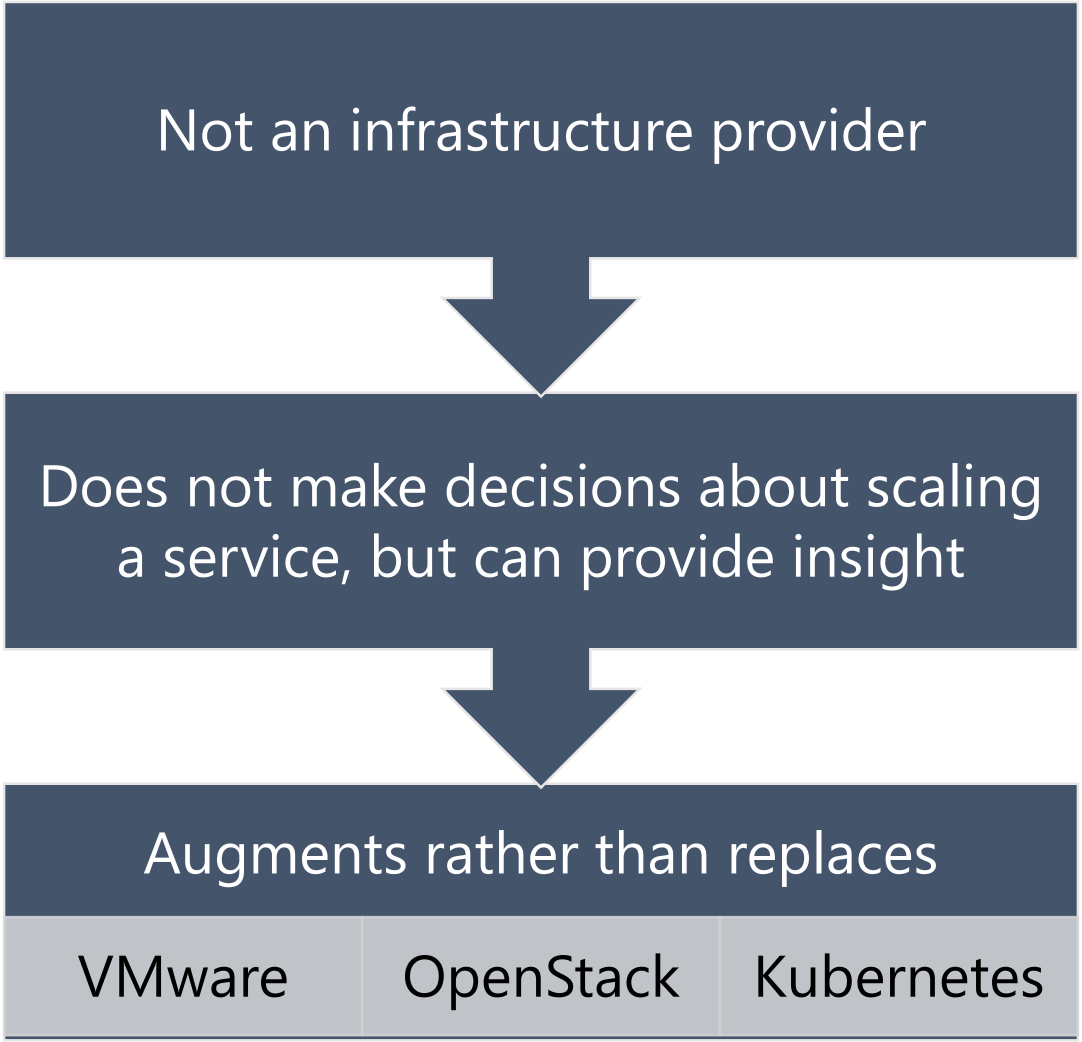
Mike’s Spin on Service Meshes

Service meshes are principally concerned about how to consume services

- End-User to Service
- Service to Service



Service Meshes are not about how to run a service



Not an infrastructure provider

Does not make decisions about scaling a service, but can provide insight

Augments rather than replaces

VMware

OpenStack

Kubernetes

Provide Network connectivity services

Rate Limiting

Coordination of load balancing

Layer 7 routing decisions (for HTTP/HTTPS)

Augment Authentication and Authorization Infrastructure

Service to Service Identity

Original Requester (Principal) to Service Identity

Integrate into existing Identity providers (consume JWTs, TLS certs, etc.)

Provide Security Controls

Permission to access a service or component of a service

Ingress/Egress identification and control

TLS on every level

Network Normalization

Service Discovery

Multiple Data Centers

Multiple Clusters

Hybrid Environments

Observation Services

Logs

Distributed Tracing

Usage Metrics

Chaos Engineering Support

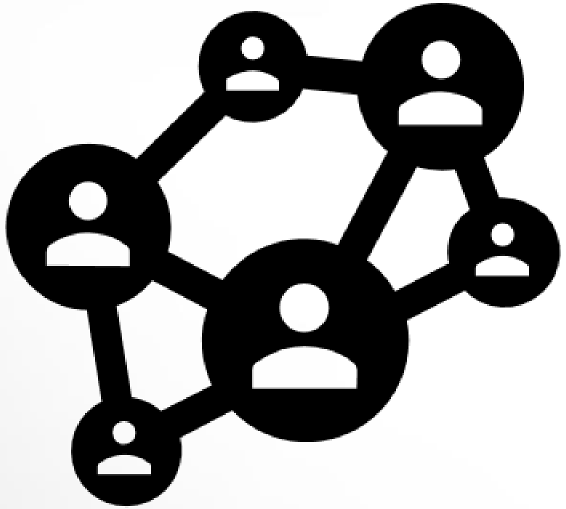
Simulate Failures

Simulate Delays

Lots of Features, But Consistent Themes

- Service meshes are providing "boring" services that some applications may or may not have
  - If they have, they may not be done using best practices
- Help services be consumed by other services or end users
- Not concerned how something is run:
  - Bare Metal vs VMs
  - Containerized deployments vs traditional deployments

Service Meshes are Useful to (Micro)Services



- We can easily see this helps streamline microservice development lifecycle.
  - It allows microservices to focus on their core business logic.
- But these same services can augment legacy or non-microservice services as well

Several use cases have been identified and tested by the Cloud Native Compute Team

RETROFIT LEGACY APPLICATIONS TO BEST PRACTICES

ALLOW SERVICES AND MICROSERVICES TO EXIST TOGETHER AS FIRST CLASS MEMBERS

ALLOW FOR A SINGLE VECTOR FOR SECURITY AND OPERATIONS MANAGEMENT

HELP "NORMALIZE" A NETWORK

PROVIDE A SIMPLER HA SOLUTION

2

Environment

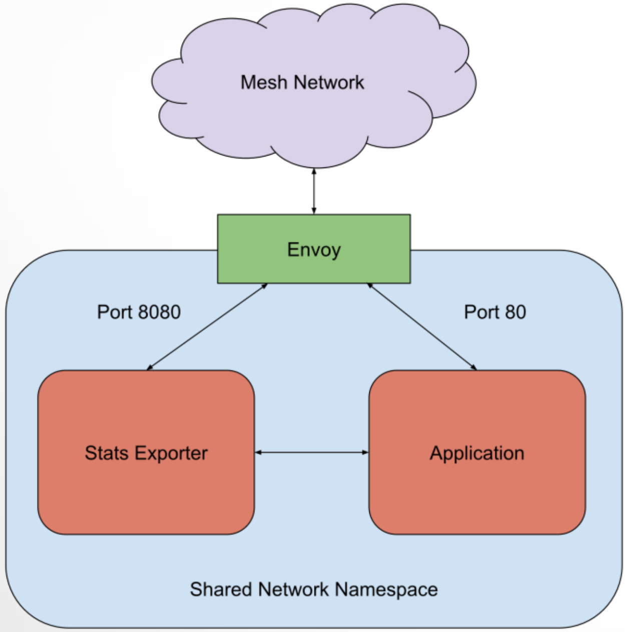
Service Mesh Used: Istio

- Community effort principally from Google, IBM, Lyft, Cisco and VMWare
- Used as a basis for many ML projects, including Kubeflow
- Also a foundation of KNative, a serverless project
- Heavily reliant on Envoy

Envoy

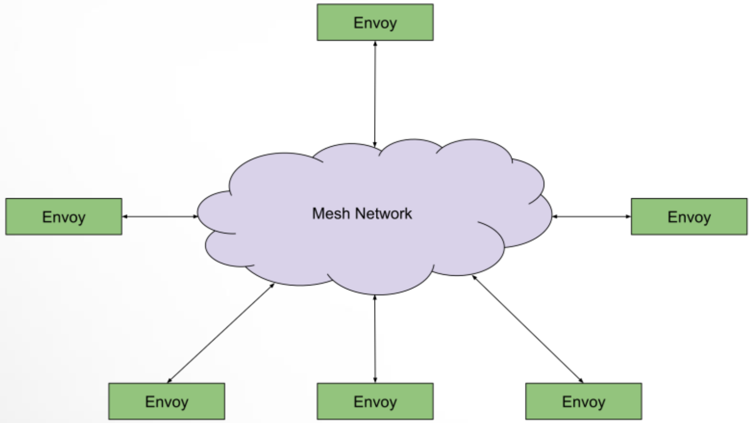
- Community effort, originally created by Lyft
- CNCF graduated project
- Network proxy
- Designed for high usage and performance

Envoy Acts a Gatekeeper Between the World and the Application



- Effect is transparent
- Traffic within namespace does not pass through Envoy

A Full Featured Istio Service Mesh Can Be Thought of as a Collection of Envoys



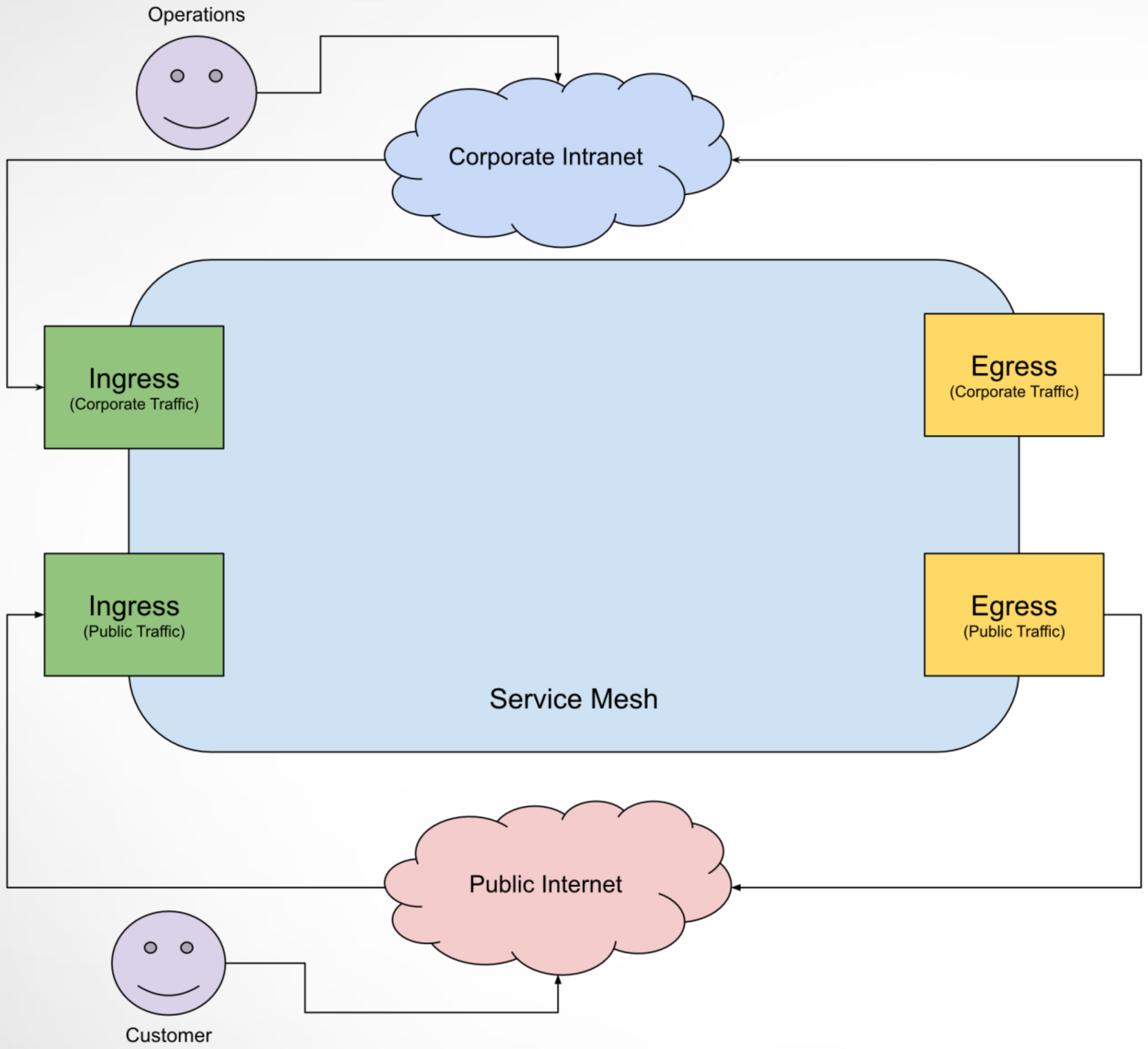
- In Istio, we can think that all traffic is communication between envoy processes
- Monitoring Envoy is a proxy of monitoring an application
- Configuration network is essentially the configuration of Envoy instances

Istio Has Other Components

- A proxy by itself is just a proxy, there must be a control plane that configures these proxies
- While somewhat straightforward; the biggest takeaway is that envoy proxies are always being used as gatekeepers

Tested Both Bare Metal and Cloud Environments

- To simulate different environments, different infrastructure providers were used
  - Bare Metal machines and Bare Metal Kubernetes clusters
  - AWS VMs and AWS-powered Kubernetes clusters



Realistic Network Topology

- Differentiated ingress/egress from Corporate vs Public networks
- Customers from public network
- Operations from corporate network

3

Use Cases

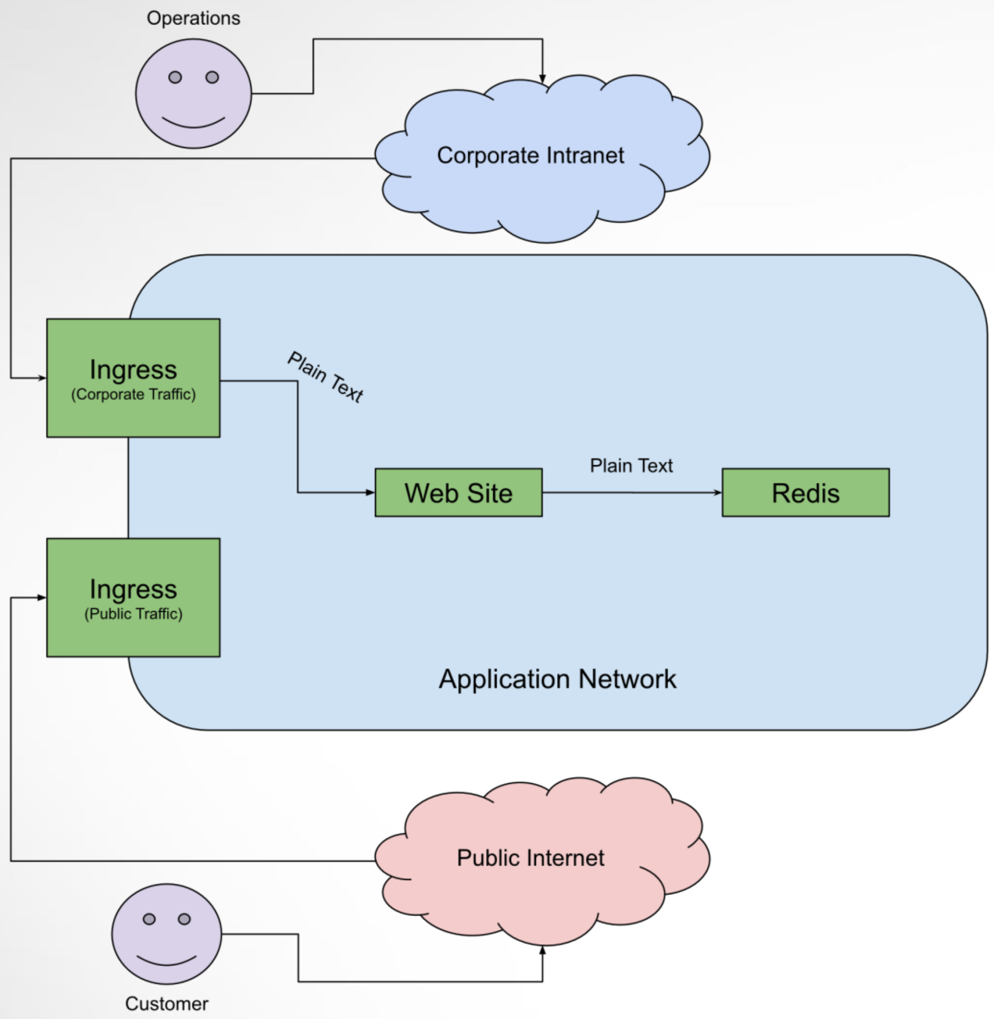
Use Case 1

Retrofit Legacy Applications to Best Practices

REDIS - NORMALLY RUNNING WITHOUT ENCRYPTION

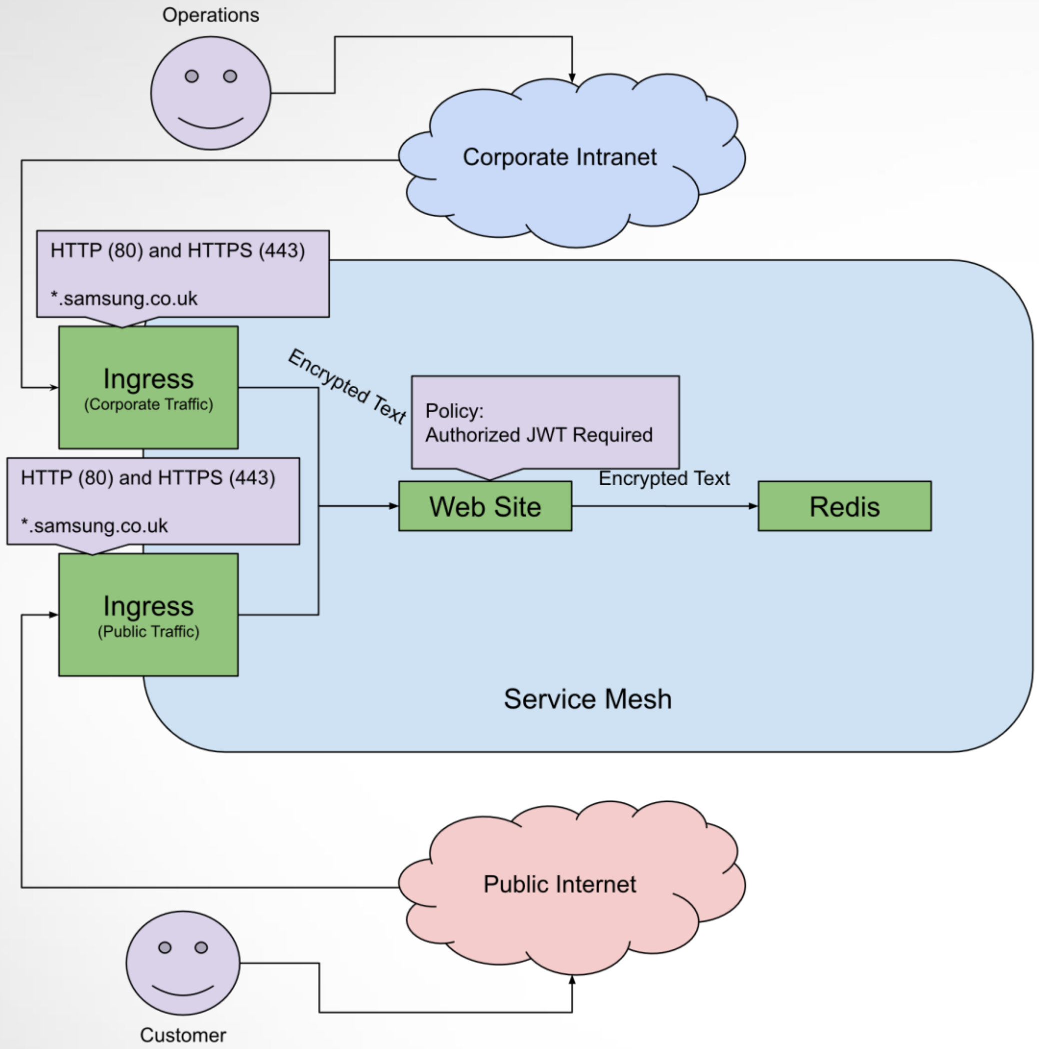
WEB SERVICE WITHOUT TLS OR AUTHENTICATION/AUTHORIZATION

Problem Description



- Web service originally relied on firewall rules only for protection
- Traffic to web service was done over HTTP
- Traffic between redis and web service was not encrypted

Typical Problems



- Enable TLS to web service handled by service mesh
- Have service mesh process authentication and authorization through JWT
- Have transparent mTLS encryption enabled between web service and redis

Improvements Made

Use Case 2

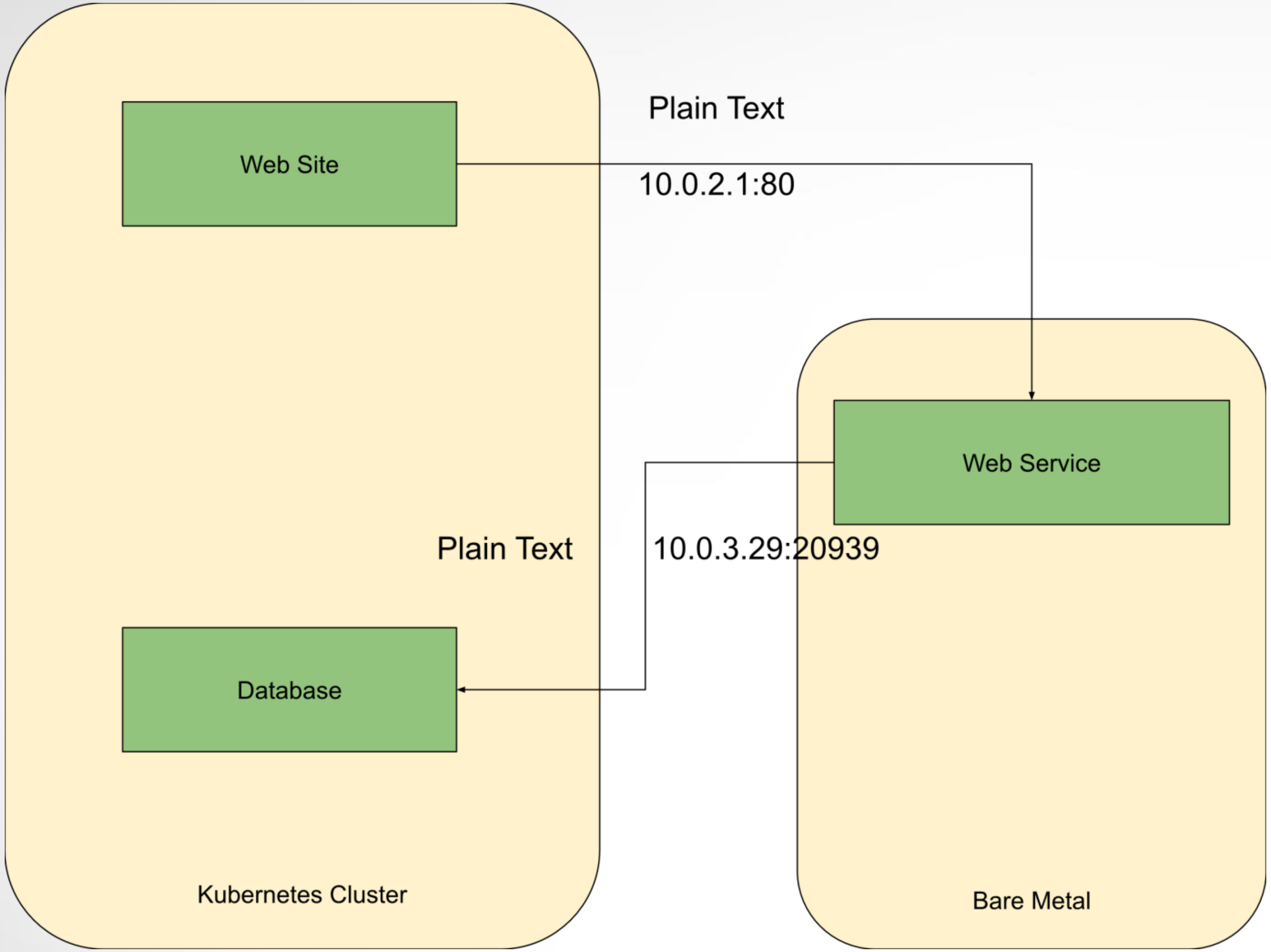
Allow Services and Microservices to Exist Together as First Class Members

Problem Description

MODERN WEB SERVICE RUNNING IN KUBERNETES

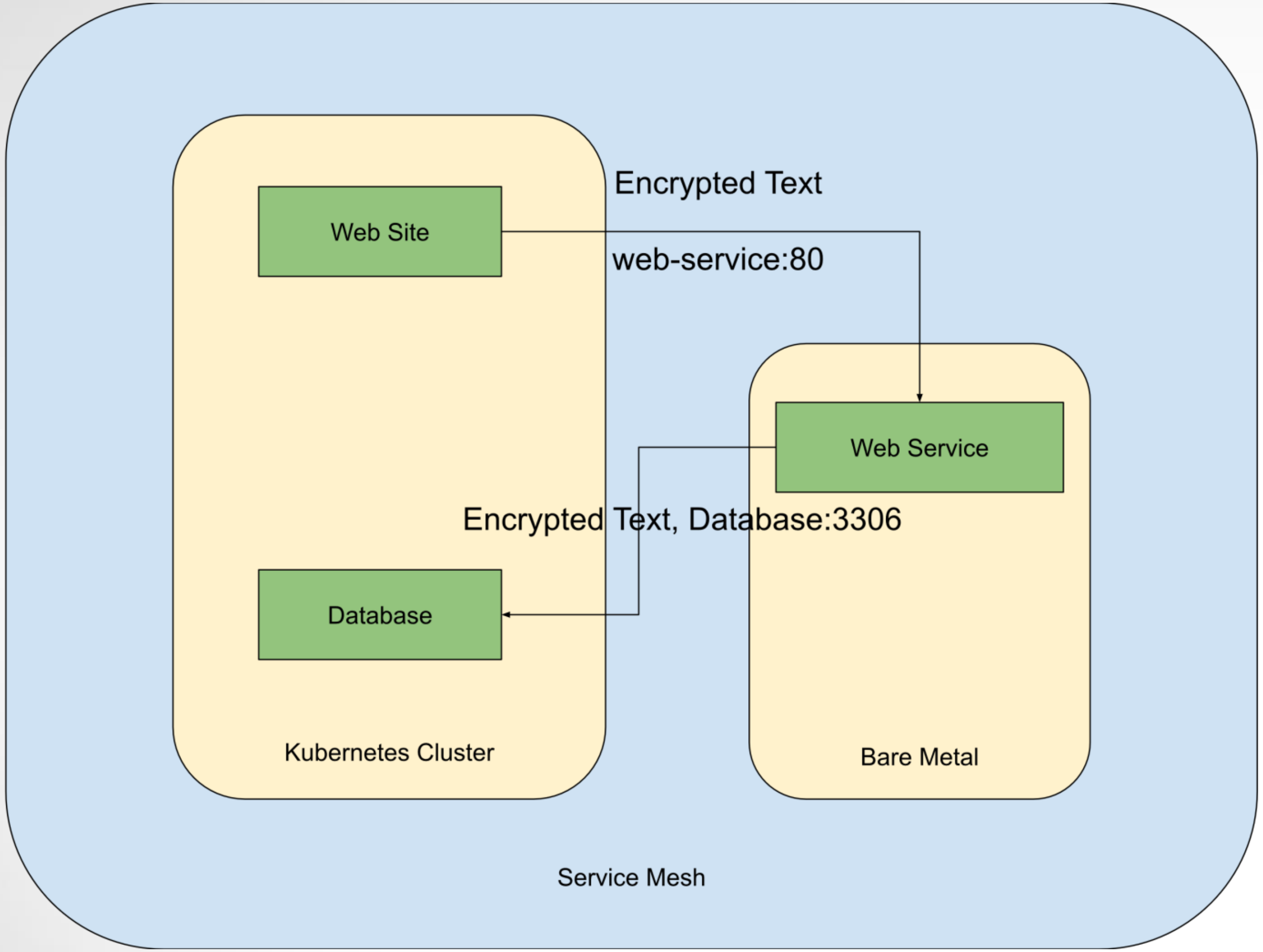
LEGACY WEB SERVICE RUNNING ON BARE METAL MACHINE

MYSQL DATABASE RUNNING IN KUBERNETES



- Hard coded IPs and ports for legacy service to talk to mysql service
- No transparent encryption enabled between services
- Hard to differentiate traffic going between bare metal machine and kubernetes cluster

Typical Problems



- Have bare metal machine join service mesh through mesh expansion
- Service discovery managed through service mesh - no hard coded service IPs and ports
- Traffic control managed and auditable through the service mesh control plane
- Enable transparent mTLS encryption across service mesh

Improvements Made

Use Case 3

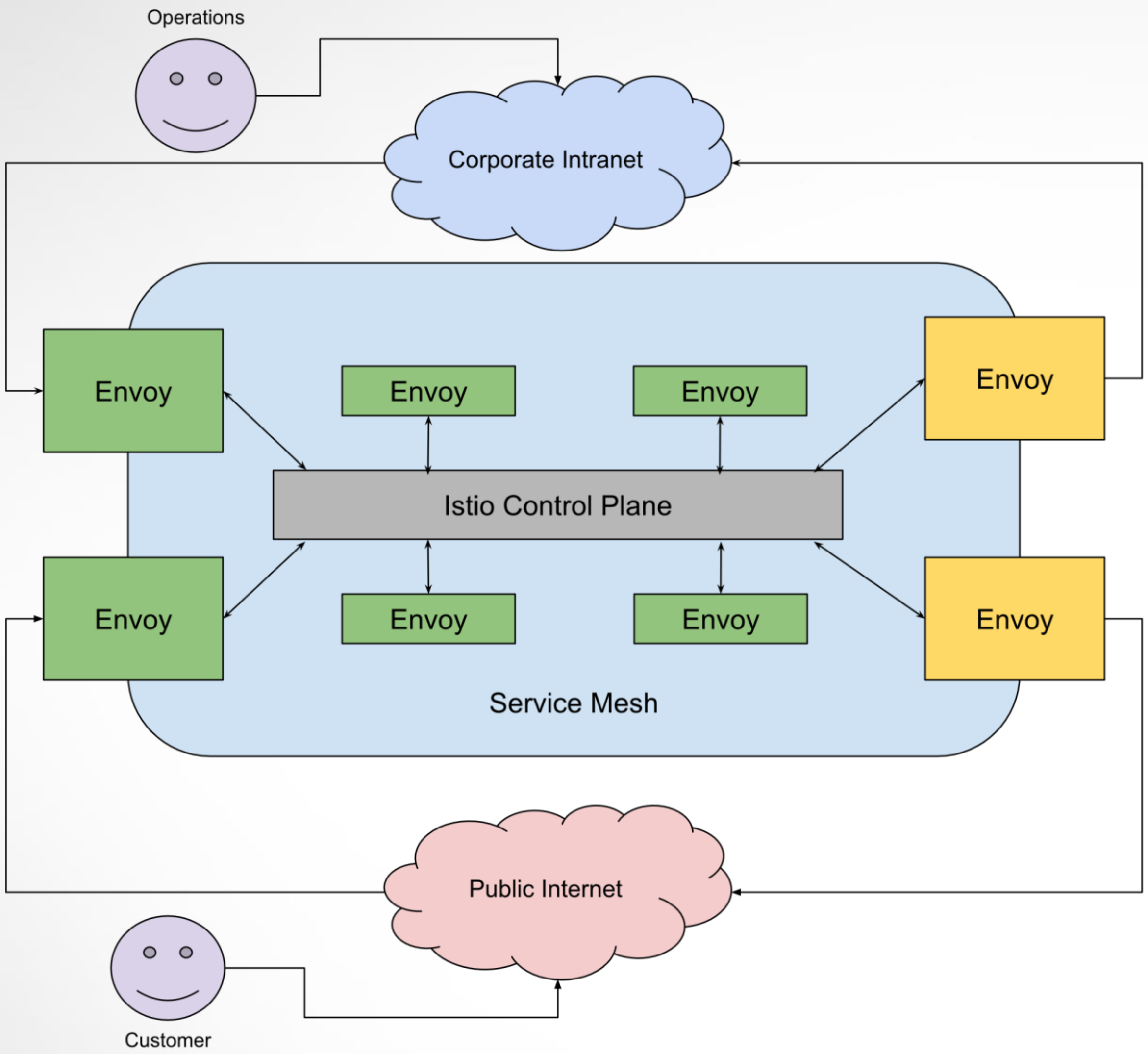
Allow For a Single Vector For Security and Operations Management

EACH SERVICE HAVING THEIR OWN WAY OF DELIVERING TLS CERTIFICATES

EACH SERVICE HAVING THEIR OWN LOGGING SYSTEM

FIREWALL RULES SOMETIMES DONE WITHIN KUBERNETES, SOMETIMES DONE BY NETWORKING TEAM

Problem Description



- Firewall rules implemeneted mesh-wide
- TLS certificates managed by service mesh
- Logging handled through common vector

Leverage Istio Control Plane

Use Case 4

Help “normalize” a Network

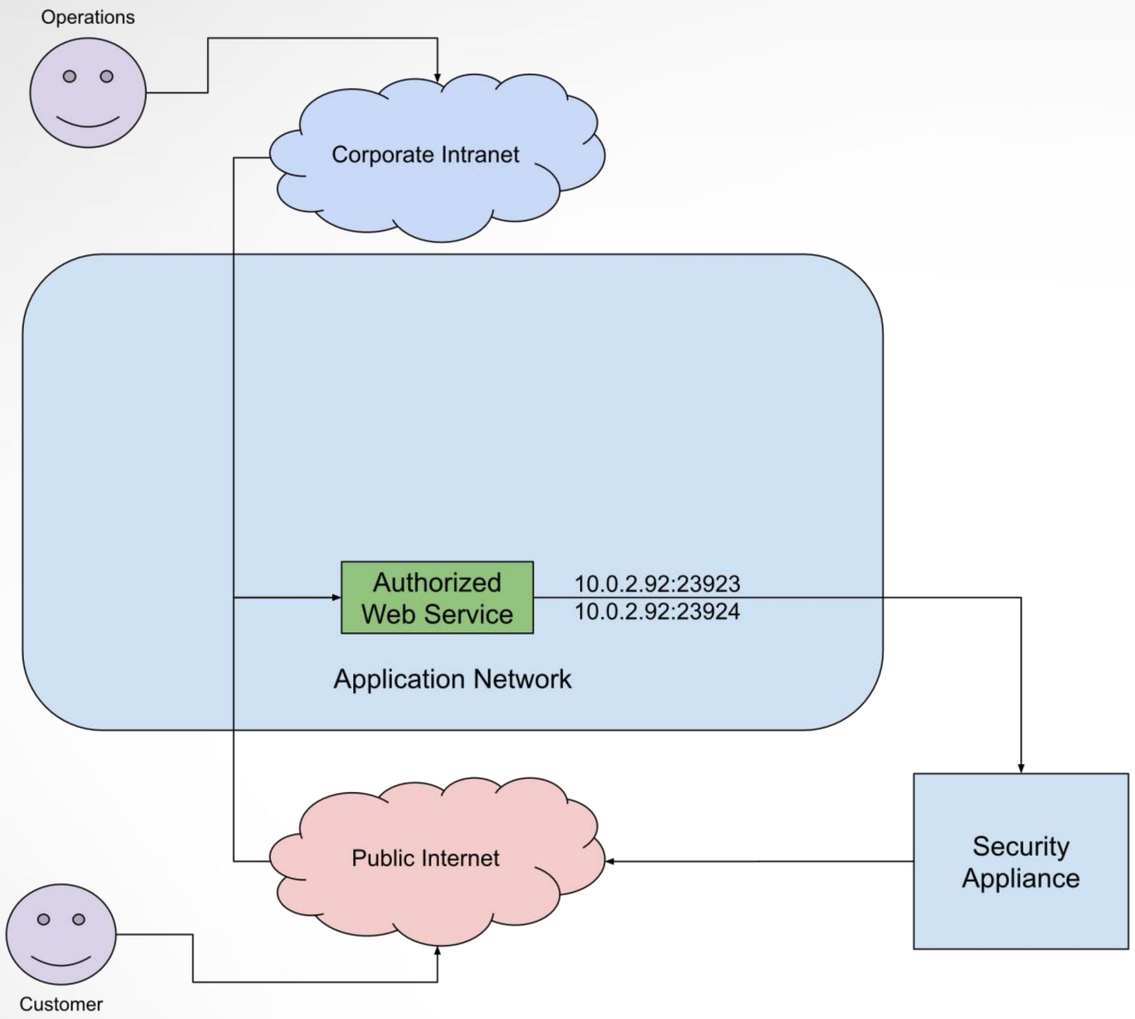
SERVICE ACCESS TWO PUBLIC EXTERNAL APIS, TWITTER AND FACEBOOK

NEED TO INSPECT TRAFFIC TO THESE PUBLIC SERVICES - PROXY TLS

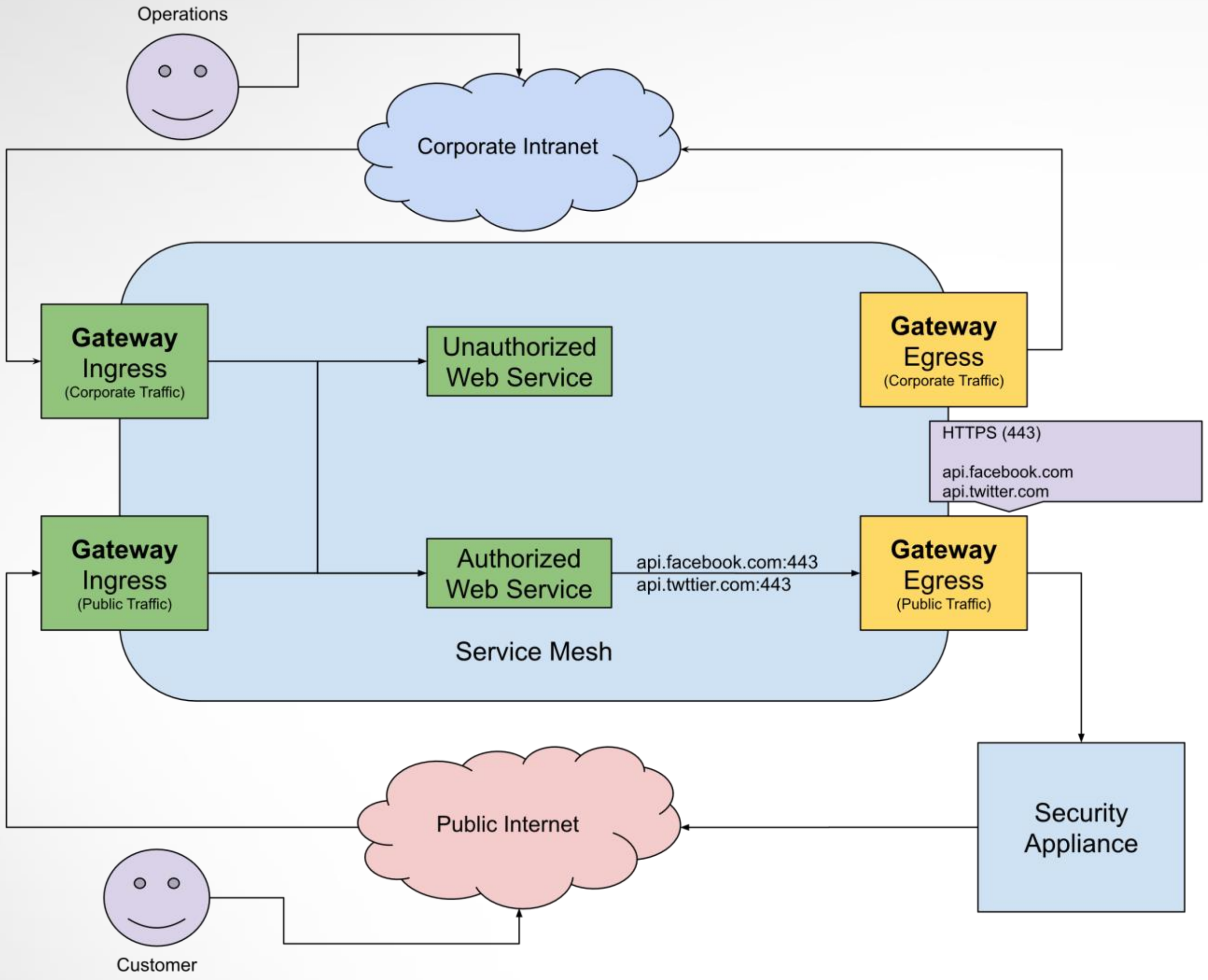
NEED TO ENSURE ONLY THIS SERVICE TALKS TO THE EXTERNAL APIS

Problem Description

Typical Problems



- Because of proxy TLS, service code works differently on developer's laptop vs production
- Often not handled with TLS enabled
- Hard to audit to ensure only whitelisted customers can talk to whitelisted endpoints



- Egress with TLS enabled in service mesh
- Egress configured to talk to specific proxy service
- Traffic route created for whitelisted services to talk public APIs
- Service Mesh certificate authority included in service app build
- From service code perspective, service is talking "naturally" to public APIs
- Security Compliance is kept

Improvements Made

Use Case 5

Simpler HA Solution

Problem Description

FIRST WEB SERVICE IS ACCESSED BY END USER

SECOND WEB SERVICE ACCESSED BY FIRST WEB SERVICE

SECOND SERVICE EXPERIENCES UNEXPECTED DOWNTIME IN SAME DATA CENTER

IDENTICAL CLUSTER EXISTS IN ANOTHER DATA CENTER, HAS EXCESS CAPACITY

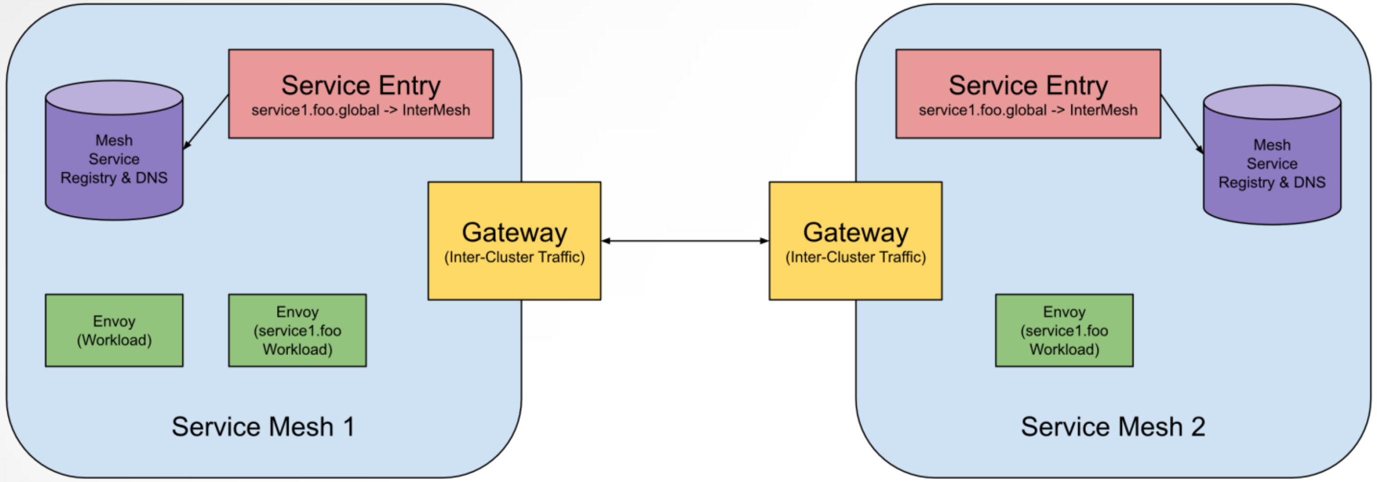
Typical Problems

1

Handling failover to another data center is typically hard

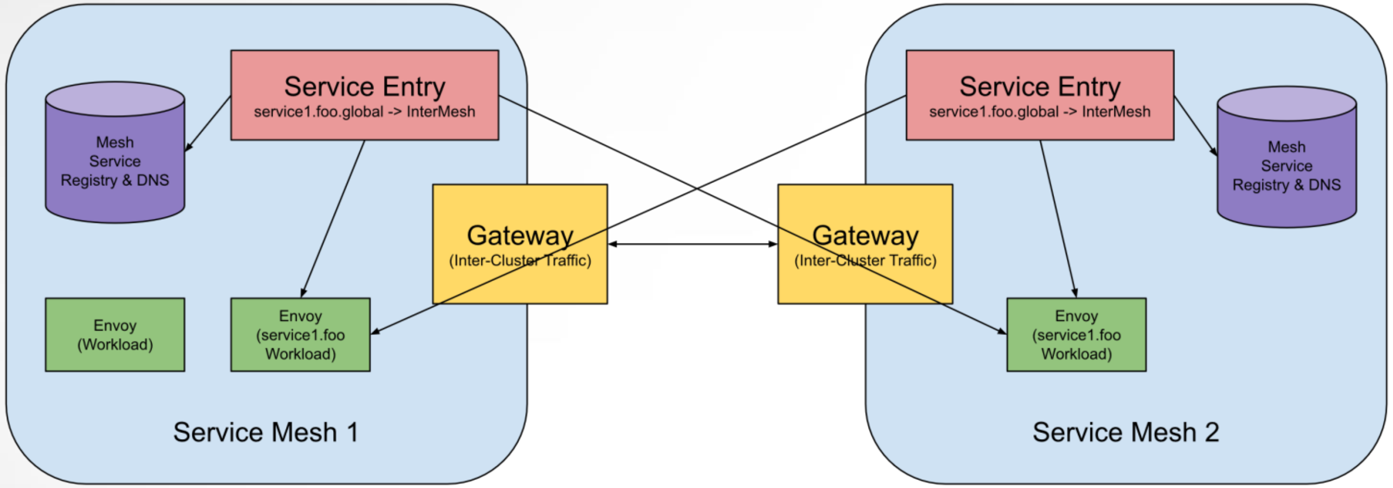
2

Handling encryption of the traffic across data centers can be even more difficult



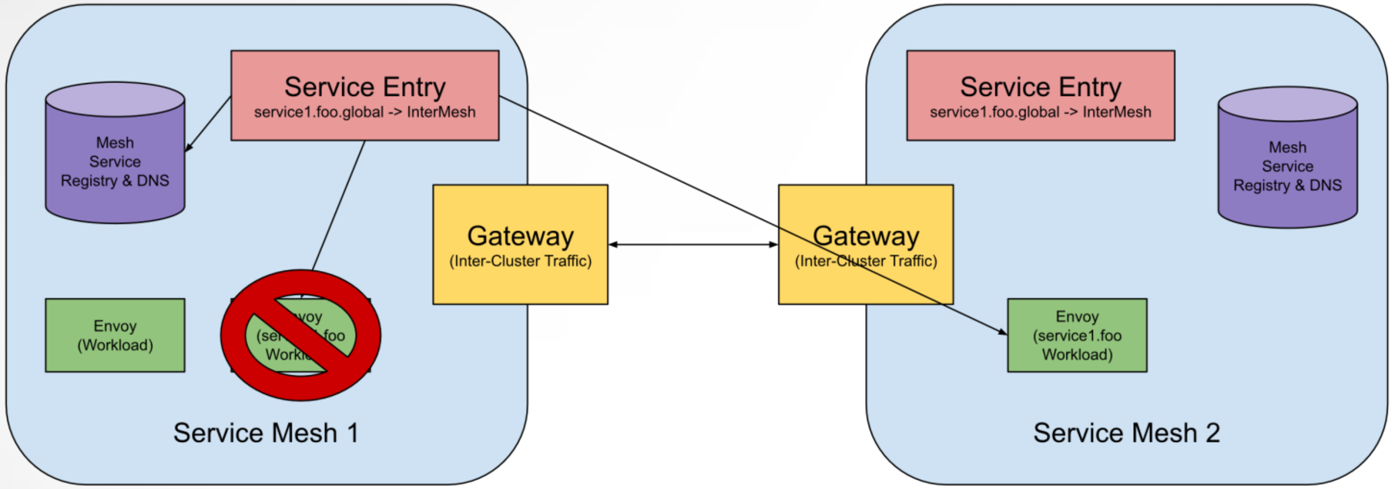
- Two Clusters
- Two Control Planes
- Workload on Cluster 1 wants to talk to a service that is normally available on both networks

Service Mesh Setup


Service Entry
service1.foo.global -> InterMesh
Mesh Service Registry & DNS
Gateway
(Inter-Cluster Traffic)
Envoy (Workload)
Envoy (service1.foo Workload)
Service Mesh 1
Service Entry
service1.foo.global -> InterMesh
Mesh Service Registry & DNS
Gateway
(Inter-Cluster Traffic)
Envoy (service1.foo Workload)
Service Mesh 2

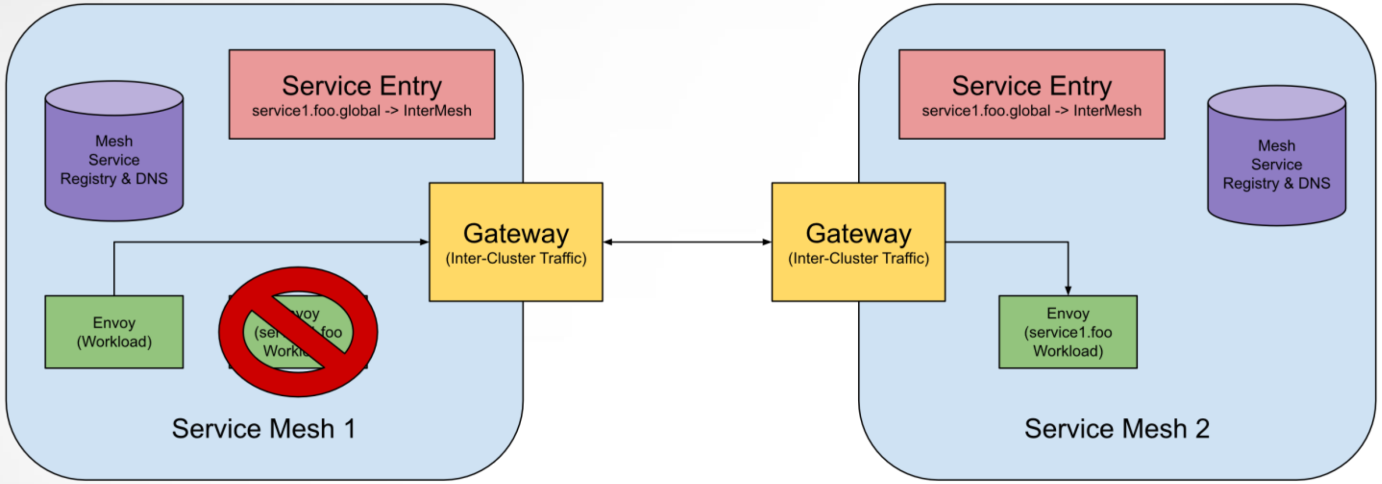
- Workloads on either cluster can potentially connect to service1.foo workload if they ask for service1.foo.global

Global Service Registry



- service1.foo on cluster 1 is currently down
- service1.foo.global does also point to service1.foo on service mesh 2

Workload is Offline



- Traffic from Service Mesh 1 is routed over to Service Mesh 2
- No downtime for customer

Service is available

4

Lessons Learned

Better for HTTP services

Service Meshes are far more useful for HTTP-based Services

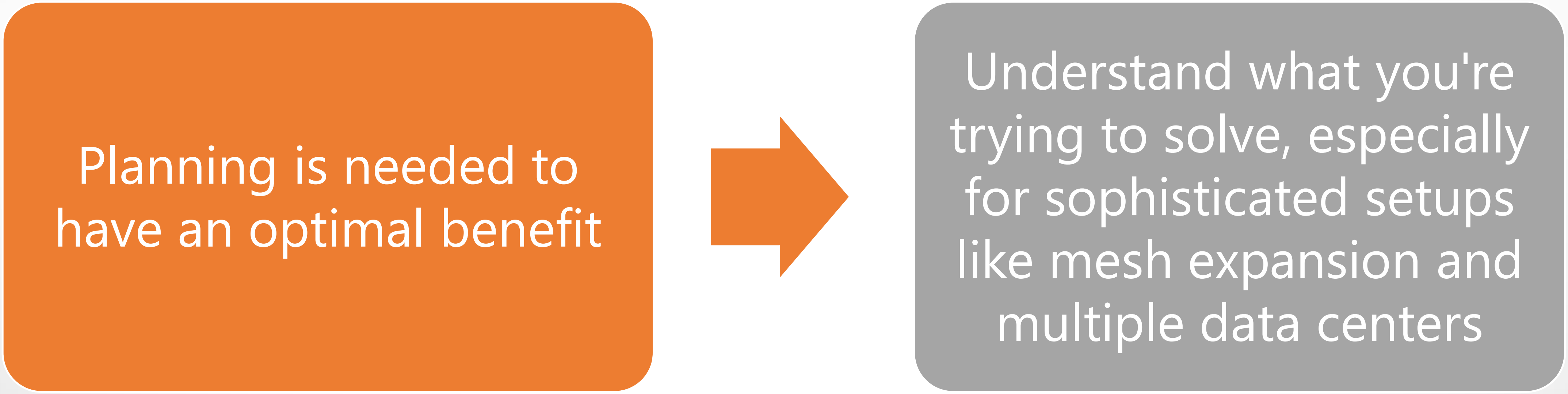
REST, gRPC, Mongo, etc. Because of well known layer 7 capabilities

Matching, Mutating on HTTP parameters

Still useful to non HTTP-based services

Less options because raw TCP is much more free-form

Come In With a Plan



Planning is needed to have an optimal benefit

Understand what you're trying to solve, especially for sophisticated setups like mesh expansion and multiple data centers

Great Solution For Legacy Services

Ability to apply without any application changes

Over-the-Wire Encryption

Authentication

Rate Limiting

Helps comply with modern security recommendations

Great Solution for Fault Tolerance

Configuration driven fault tolerance is easy to understand and quickly implement

Versatility in Multi Cluster Service Mesh deployment models means almost every cluster can be securely linked

Helps pin data storage to specific customer regions (EU data stored on EU infrastructure)

Service Meshes are Almost a Must Have For Microservices

But the benefits extend to regular services as well

Consider how a service mesh can benefit your current environment - it may be a great fit to your current situation

Thank You

Q & A

Partner Disrupt Foresee