

Multi-cluster Made Reasonable Envoy Service Mesh Control Plane

2020-08-19

Ashley Kasim & Paul Fisher @ Lyft





Ashley Kasim

Staff Software Engineer
Compute Platform @ Lyft


 akasim@lyft.com



Paul Fisher

Tech Lead on Compute Platform

Willing Kubernetes into existence at Lyft

 pfisher@lyft.com

 @paulnivin



Agenda

- 1 Lyft Overview
- 2 Lyft Envoy Environment
- 3 Multi-Cluster / Dyplomat
- 4 Dyplomat Demo



Lyft Overview

Lyft's Scale



- Rideshare network in all 50 US states, Toronto, Ottawa, and Vancouver
- Scooter and Bikeshare networks in US
- Transit Partnerships in 11 markets
- Autonomous Vehicle Partnerships in two cities, Las Vegas (Aptiv) and Phoenix (Waymo)

Lyft Kubernetes' Scale



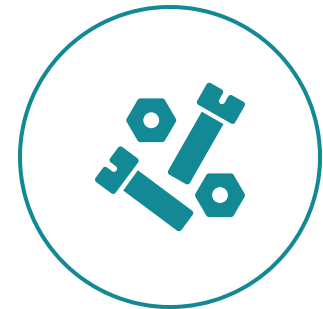
Machine Learning

- Training Jobs
- Jupyter Notebooks
- GPU Workloads
- 5K+ Pods
- 10K+ Cores



Rideshare

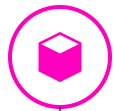
- 600+ Stateless Micro Services
- Redundant Clusters per AZ
- 1 Production Envoy Mesh
- 30K+ Pods (autoscaling)
- 300K+ Containers (sidecars)
- 80K+ Cores



Flyte

- Distributed Workflow Orchestration
- Executors for Spark, Hive, AWS Batch
- 10K+ Pods
- 5K+ cores

Lyft Kubernetes Timeline



December 2015, Lyft starts internal container project for dev/CI stack



December 2017, Lyft open sources VPC CNI stack



2019, Lyft stateless service migration to Kubernetes



May 2017, Lyft investigates options for running Kubernetes on AWS



2018, Lyft batch and ML workloads migrated to Kubernetes



2020, Lyft stateful service migration to Kubernetes

Lyft Kubernetes Environment



- **Kubernetes 1.16**

Moving to 1.18

- **Fedora (n-1) with cri-o**

Moving to Fedora CoreOS

Mainline kernels

Minimal OS

- **Ubuntu User Space**

Lyft Developers like Ubuntu

- **Immutable Infrastructure**

Packer (Fedora), Ignition (Fedora CoreOS)

Terraform Orchestration

- **AWS**

Lots and lots of EC2, EBS, and S3
us-east-1 and us-west-2 build outs

- **Redundant Per-AZ Clusters**

Sets of clusters for staging and production
Staggered roll-outs with limited blast radius

- **Lyft CNI Stack**

VPC native

Low latency

High throughput

Pods are directly part of the Envoy Mesh

Keep it Simple



- No overlay networks
- No NAT
- No Ingress
- No kube-proxy
- Pods can communicate with Pods in any cluster
- Envoy for service to service comms



VPC Native Network

- Pods receive VPC IP addresses
- Full connectivity within VPC
- Native network performance
- 2 main CNI plugin options for AWS

AWS - [amazon-vpc-cni-k8s](#)

Lyft - [cni-ipvlan-vpc-k8s](#)



Lyft Envoy Environment

A close-up photograph of a network switch rack. Several teal-colored Ethernet cables are plugged into the ports, with some cables having yellow labels. The ports are numbered, and the background is slightly blurred, showing more of the rack and other equipment.

Lyft Envoy Overview

- **Two Envoy meshes**
 - One staging, one production
 - Moving to production per-AZ split mesh in the future
- **One main Envoy front-proxy (lyft.com)**
- **Multiple Kubernetes clusters**
- **One VPC IP per Kubernetes Pod**
 - Pods on any cluster can communicate with Pods on all other clusters
- **Lyft EnvoyManager control plane**

Lyft VPC CNI plugin



- **Minimalist design**

No DaemonSets

No Pods

No Runtimes

Stateless go binaries

- **Tested w/ cri-o & containerd**

cri-o @ Lyft

containerd @ Datadog

- **No overlay network**

- **IPvlan VPC interface**

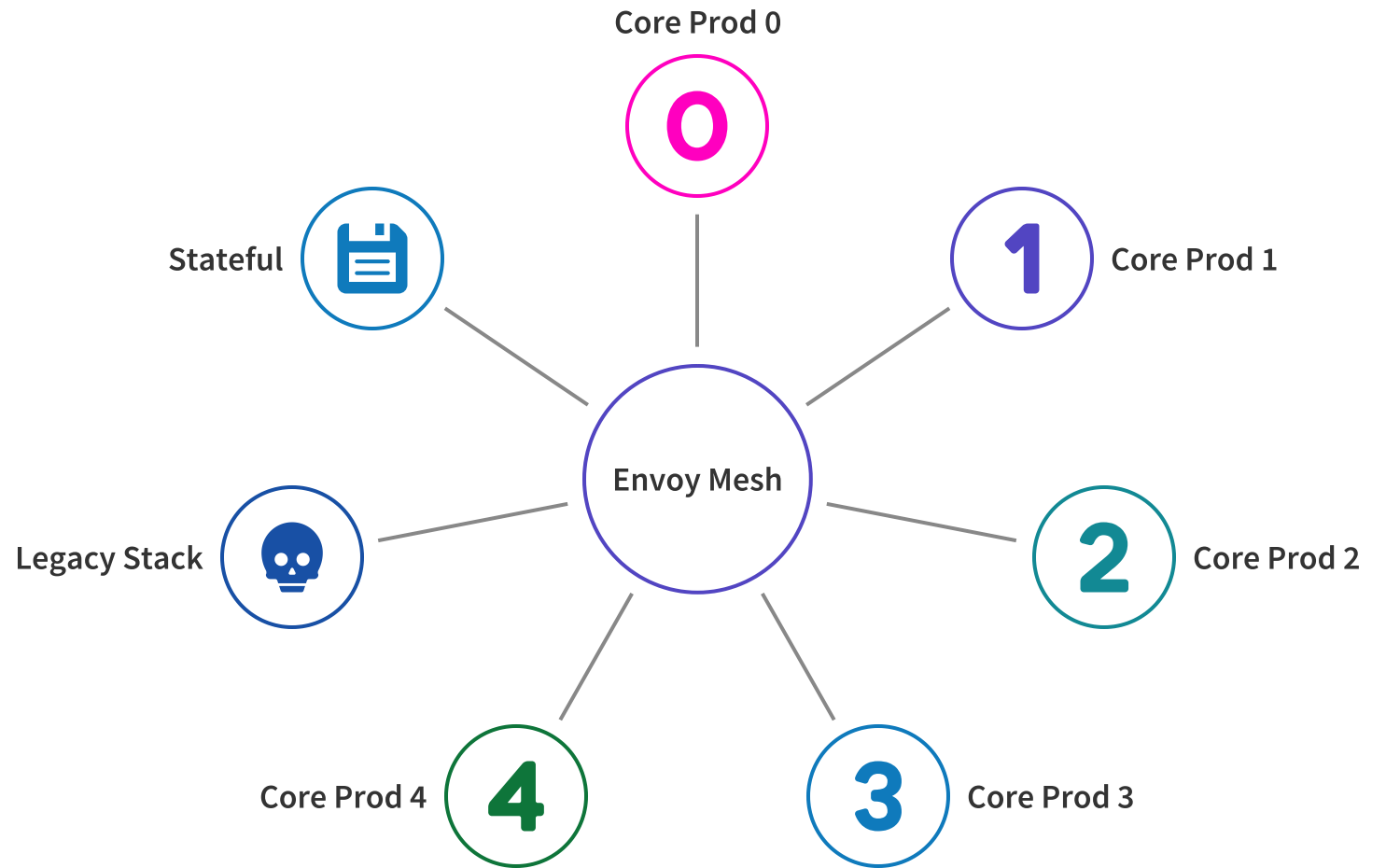
- **No asymmetric routing**

- **No VPC routing table changes**

- **Feature complete**

Running in production for 3 years

Lyft Production Envoy Mesh





Multi-Cluster

Terminology

- **Kubernetes Cluster**

API node w/ at least one worker node running Pods

- **Envoy Cluster**

Collection of Envoy endpoints comprising a “service”

- **Envoy Endpoint**

Envoy cluster member (IP/Port)

- **xDS**

Collection of Envoy discovery services and APIs

Cluster Discovery Service (CDS)

Endpoint Discovery Service (EDS)

- **go-control-plane**

Reference open source go-based implementation of xDS, useful for building custom Envoy control planes

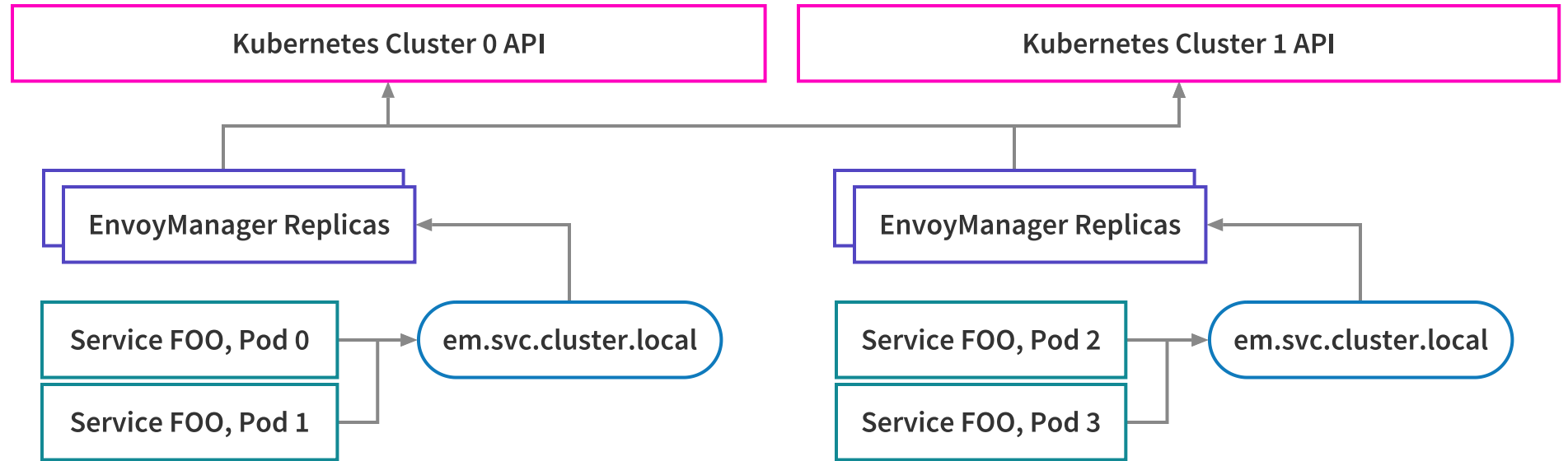


Lyft's Envoy Control Plane

EnvoyManager (EM)

- go-control-plane based
- Informers determine Pod status and bridge cluster together
- Replicas communicate with all clusters
- Provides xDS to Pods on start up
- Lots of Lyft specific & legacy functionality

EnvoyManager Deployment



- **Envoy sidecar uses DNS to find EM**
Cluster-local headless service
- **Service Pods exist on multiple clusters**
Losing a cluster is not catastrophic

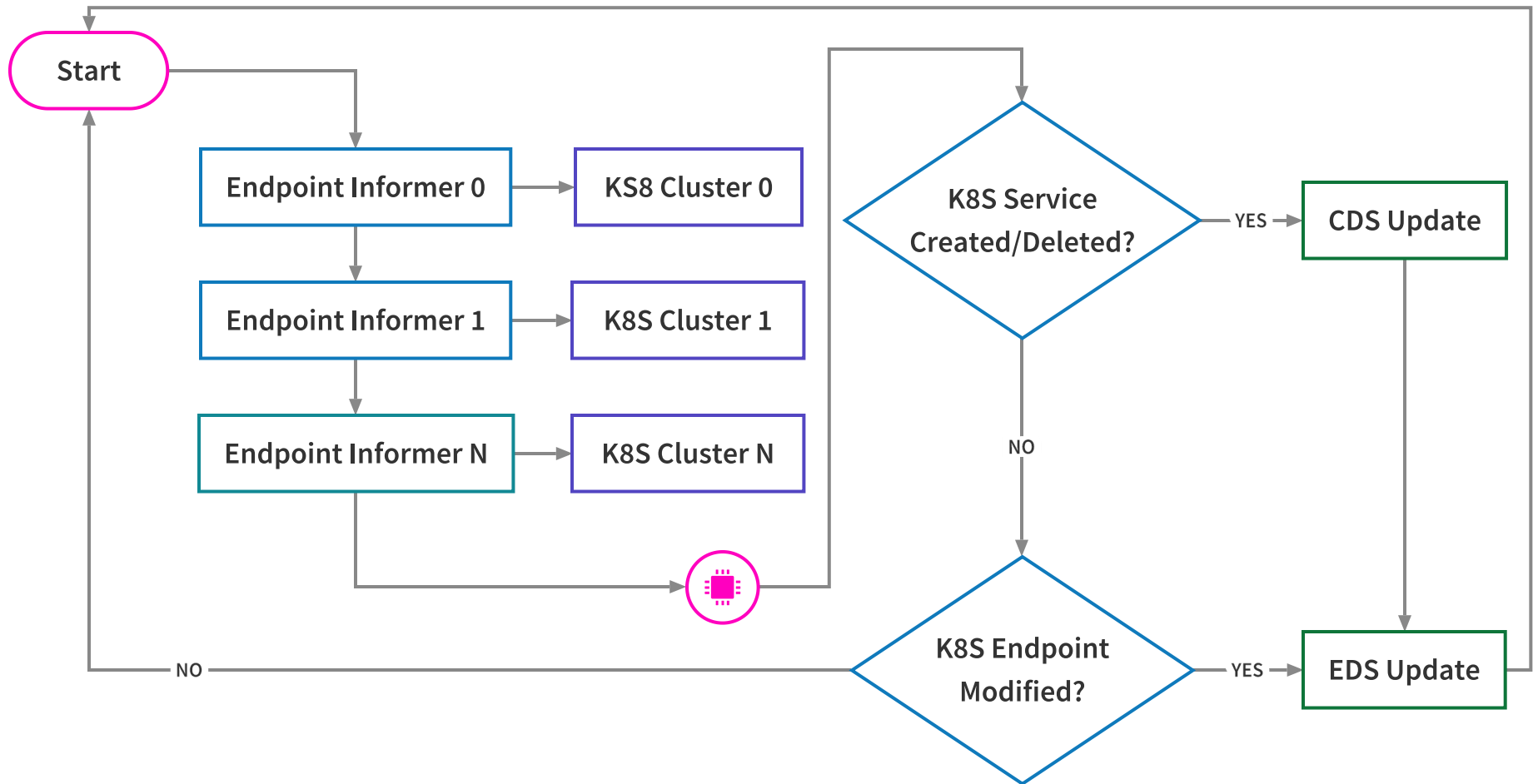
- **Multiple independent EM replicas**
Fault tolerant
- **EM replicas communicate with all clusters**
All service Pods are part of the mesh



Dyplomat

- go-control-plane based example implementation for Kubernetes
- Multi-cluster support
- Open source
- Simplified EDS control loop used by EnvoyManager
- IAM auth support on AWS

Dyplomat CDS/EDS Control Loop



Future Work

- **EndpointSlices**

beta in Kubernetes 1.17

- **Pod vs Endpoint Informer**

See “[Service Mesh in Kubernetes: It’s Not That Easy](#)” from EnvoyCon 2019, Lita Cho & Tom Wanielista, Lyft

- **Allow for immediate host removal**

<https://github.com/envoyproxy/envoy/issues/9246>



Dyplomat Demo

Thanks!

- **Dyplomat is available as part of the upstream go-control-plane repo**

<https://github.com/envoyproxy/go-control-plane/>

- **We're hiring!**

<https://lyft.com/jobs>

<https://github.com/envoyproxy/go-control-plane/>

