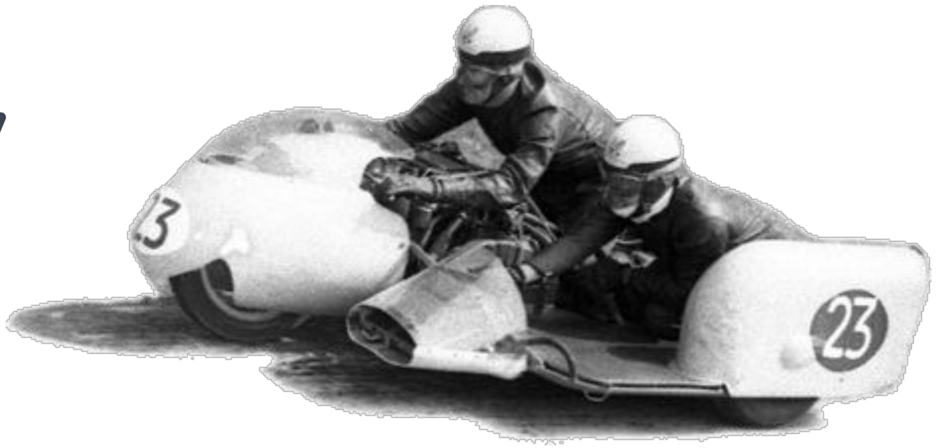
Envoy



GO

Mihai Todor
28.05.2020

Integrating the Envoy gRPC API into a Dynamic Service Discovery Platform

About me

- Senior Software Engineer at Cogito

- Focused on highly-scalable distributed systems and Go

- Find me on:
    - Linkedin: https://www.linkedin.com/in/mtodor
    - Twitter: @MihaiTodor

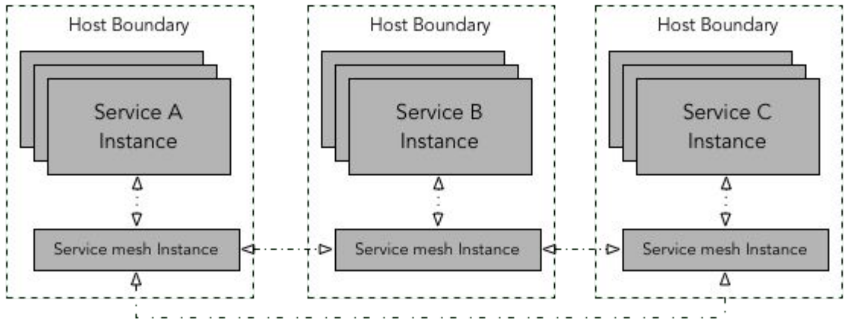
Service meshes

- Infrastructure layer of an enterprise service cluster
- Handle service-to-service communication
  - Reliability
  - Security
  - Observability
  - Management
- Examples:
  - Istio https://istio.io/
  - Consul https://www.consul.io/
  - Linkerd https://linkerd.io/
  - Kuma https://kuma.io/
  - Maesh https://containo.us/maesh/

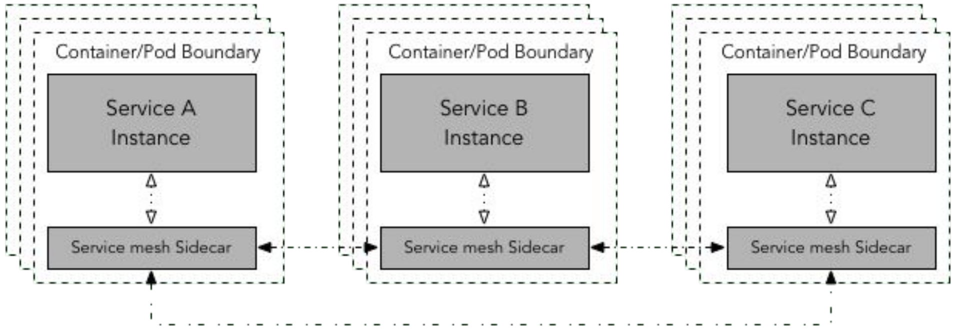
HashiCorp Consul

æ maesh

Service mesh deployment models



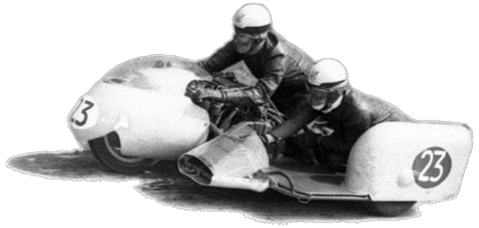
Per-host proxy deployment



Sidecar proxy deployment

Via: https://www.abhishek-tiwari.com/a-sidecar-for-your-service-mesh/

Sidecar https://github.com/Nitro/sidecar

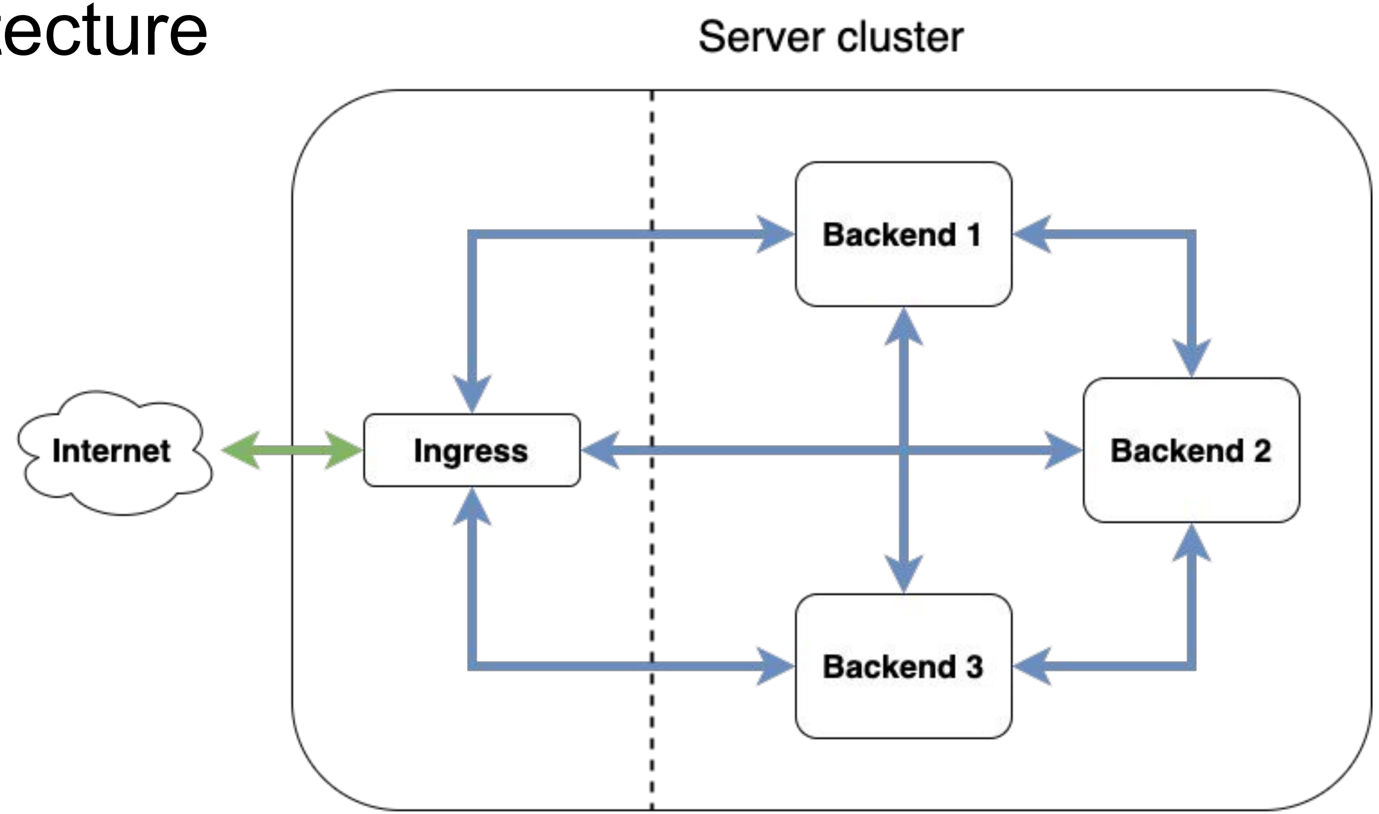


- Dynamic service discovery platform
- Per-host proxy via either HAProxy (through haproxy-api) or Envoy
- Docker native
- Gossip-based communication between hosts via Memberlist
- Health checks (HTTP or external)
- Has been used in production on Apache Mesos clusters

Sidecar history

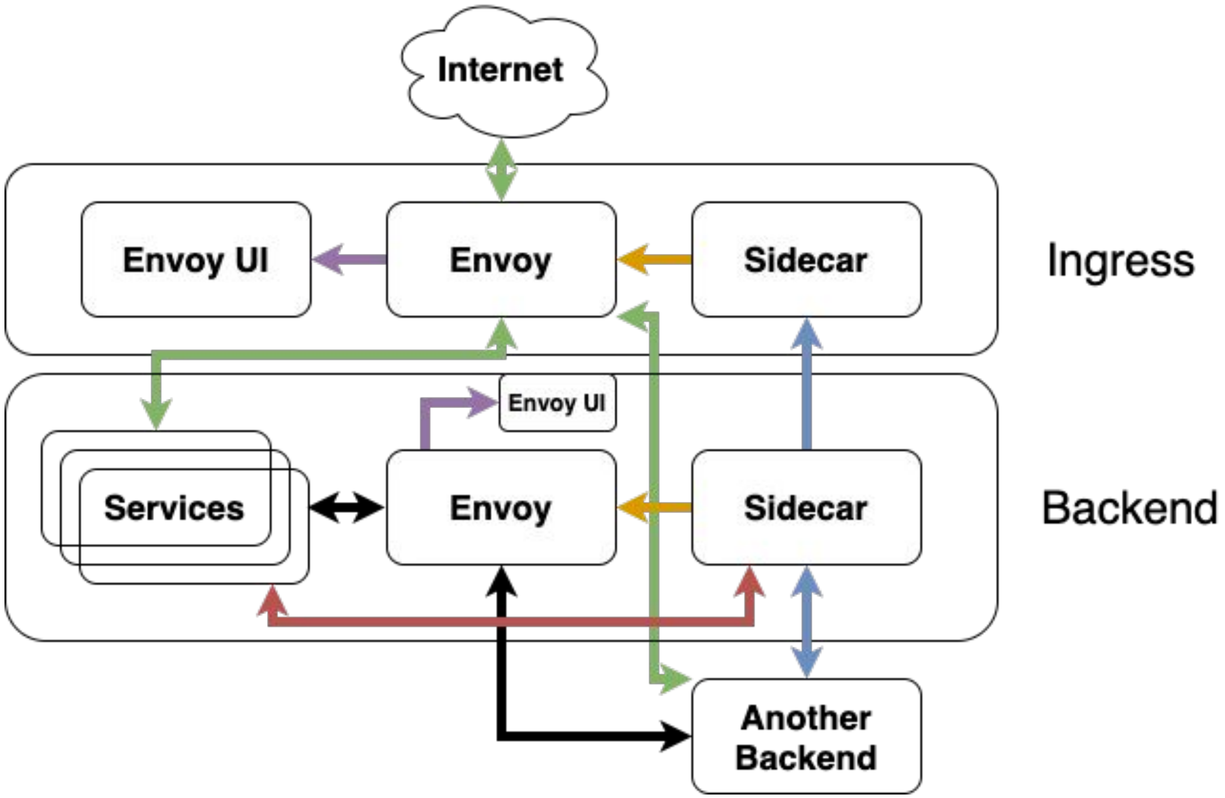
- Karl Matthias started developing it at New Relic back in 2015
    - I'll have to ask him what "bosun" stands for :)
- He deployed a fully-working version in production at Nitro in 2016 on top of a Mesos cluster, initially using HAProxy as a sidecar proxy
    - Envoy support was added afterwards via the now deprecated Envoy REST API
- He continues to use it in production at Community and maintain it
- I recently integrated the Envoy go-control-plane to enable support for the Envoy gRPC API
    - I used the Aggregated Discovery Service, which is part of the xDS gRPC-based V2 API

Architecture



Detail

- Ingress traffic
- Inter-service traffic
- Sidecar gossip
- Sidecar Docker discovery
- Sidecar health checks
- Sidecar -> Envoy updates
- Envoy UI updates



Demo

- Simulated cluster on local laptop using Docker-in-Docker aka DinD
- One ingress container running
  - Sidecar in listener mode
  - Envoy with static listeners and ADS
  - Envoy UI
- Three backend containers (or more) running
  - Sidecar
  - Envoy with ADS
  - Envoy UI
  - Three WhoAmI containers
    - two HTTP services
    - one TCP service

Envoy APIs

- v1 REST-JSON xDS API (deprecated and no longer supported)
- v2 xDS API (deprecated, end-of-life EOY 2020)
- Beyond...

Push vs Pull based APIs

- The Sidecar and Envoy states are designed to be eventually consistent
- For the V1 API, Envoy was configured to pull the whole state from Sidecar every 4 seconds
- For the V2 API, Sidecar checks its internal state for updates every second and pushes the updated state to Envoy if needed
    - Alternatively, it could send an update on each state update event, but this has potential issues:
        - Spurious updates
        - Potentially missed updates

State updates

- Can be expensive to send the whole state on each update in a large cluster

- We can leverage the Envoy Incremental xDS API to send partial updates
    - Eventual consistency considerations
    - Still work in progress in the Envoy go-control-plane

Envoy go-control-plane ingredients

- xDS server: github.com/envoyproxy/go-control-plane/pkg/server
- gRPC server: google.golang.org/grpc
- github.com/envoyproxy/go-control-plane/envoy/service/discovery/v2
  - RegisterAggregatedDiscoveryServiceServer to connect the gRPC and xDS servers
- Resource cache: github.com/envoyproxy/go-control-plane/pkg/cache
  - SetSnapshot instructs Envoy that it needs to fetch the updated resources
    - Sets Envoy listeners, clusters and other resources
    - Requires a new version
    - The ***node*** parameter needs to match the value passed via `--service-node` to Envoy
    - Envoy validates clusters by default for the HTTPConnectionManager-based listeners
      - If enabled, clusters need to be added before adding listeners that depend on them; See the eventual consistency considerations
      - Can be disabled via the validate_clusters parameter of the RouteConfiguration

Testing strategies

- Mock the Envoy go-control-plane… Uh-oh

- Mock Envoy itself by creating a dummy gRPC client
  - Nonces need to be passed around correctly

- Go makes it trivial to inherit all the data members and methods of a struct and override desired methods as needed
  - Enable various assertions during concurrent workflow using a channel-based blocking state machine

Thank you!

Please let me know if you have any questions ☺