
Decoding the
Service Mesh
Landscape

Paul Bouwer
Software Engineer - Microsoft
@pbouwer

- Why do we need a Service Mesh?
- What does a typical Service Mesh offer?
- Considerations when selecting a Service Mesh
- Istio, Linkerd and Consul
- Look at broader eco-system

Why?

Security

- Firewall is security boundary
- Security managed via ip address

- Zero trust networks
- Strong identity and secure communication
- Ip addresses are dynamic

Delivery

- Known or fixed infrastructure
- Monolithic releases

- Dynamic workloads and versions
- Progressive, A/B and canary releases
- Observability requirements
- Traffic policy requirements across multiple dependent services

Resiliency and Observability

- Observability, policy (retries, timeouts, rate limiting, circuit breaking) managed in code as part of application
- Approved set of libraries for a single language or framework

- Workloads built using different languages and frameworks
- Requirements for consistent observability and policy across all workloads

What?

> “A service mesh is the connective tissue between your services that adds additional capabilities like traffic control, service discovery, load balancing, resilience, observability, security, and so on.
>
> A service mesh allows applications to offload these capabilities from application-level libraries and allow developers to focus on differentiating business logic.”

Introducing Istio Service Mesh for Microservices
by Burr Sutter, Christian Posta

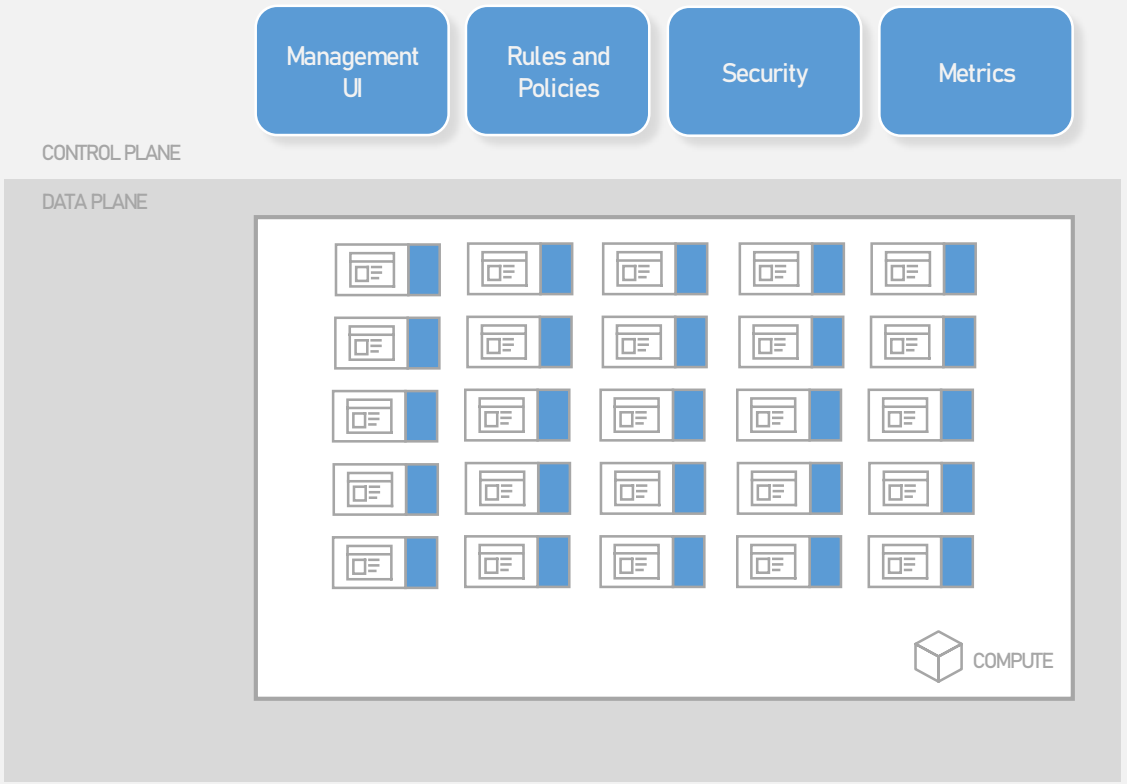
> “Istio decouples operational aspects of the services from the implementation of the services.”
>
> Eric Brewer, VP Infrastructure & Google Fellow, Google Cloud

> “Linkerd moves visibility, reliability, and security constraints down to the infrastructure layer, out of the application layer.”
>
> William Morgan, CEO, Buoyant

Typical Capabilities

- Traffic management
  - Protocol – layer 7, http, grpc
  - Dynamic Routing – conditional, weighting, mirroring
  - Resiliency – timeouts, retries, circuit breakers
  - Policy – access control, rate limits, quotas
  - Testing - fault injection
- Security
  - Encryption – mTLS, certificate management, external CA
  - Strong Identity – SPIFFE or similar
  - Auth – authentication, authorisation
- Observability
  - Metrics – golden metrics, prometheus, grafana
  - Tracing - traces across workloads
  - Traffic – cluster, ingress/egress

Typical Architecture



- Control plane
  - Management UI
  - Rules and policy definitions
  - Security management
  - Metrics collection
- Data plane
  - Sidecar proxies
  - Secure traffic between pods
  - Route and manage traffic
  - Apply policy to traffic
  - Provide traffic metrics
  - Provide tracing info

Considerations

Considerations

- Technical - traffic management, policy, security, observability
- Business - commercial support, foundation (CNCF), OSS license, governance
- Operational – installation/upgrades, resource requirements, performance requirements, integrations, mixed workloads
- Security - certificate management and rotation, external CA

Questions to ask

- Is an Ingress Controller sufficient for my needs?
- Can my workloads and environment tolerate the additional overheads?
- Is this adding additional complexity unnecessarily?
- Can this be adopted in an incremental approach?

Service Meshes


Istio


LINKERD


HashiCorp
Consul


Istio

Full featured, customisable and extensible service mesh

- Announced - May 2017, GA (1.0) in July 2018
- Governance - Google, IBM
- OSS - Apache 2.0
- Commercial support – Aspen Mesh
- Cloud offerings – Google Cloud, IBM Cloud

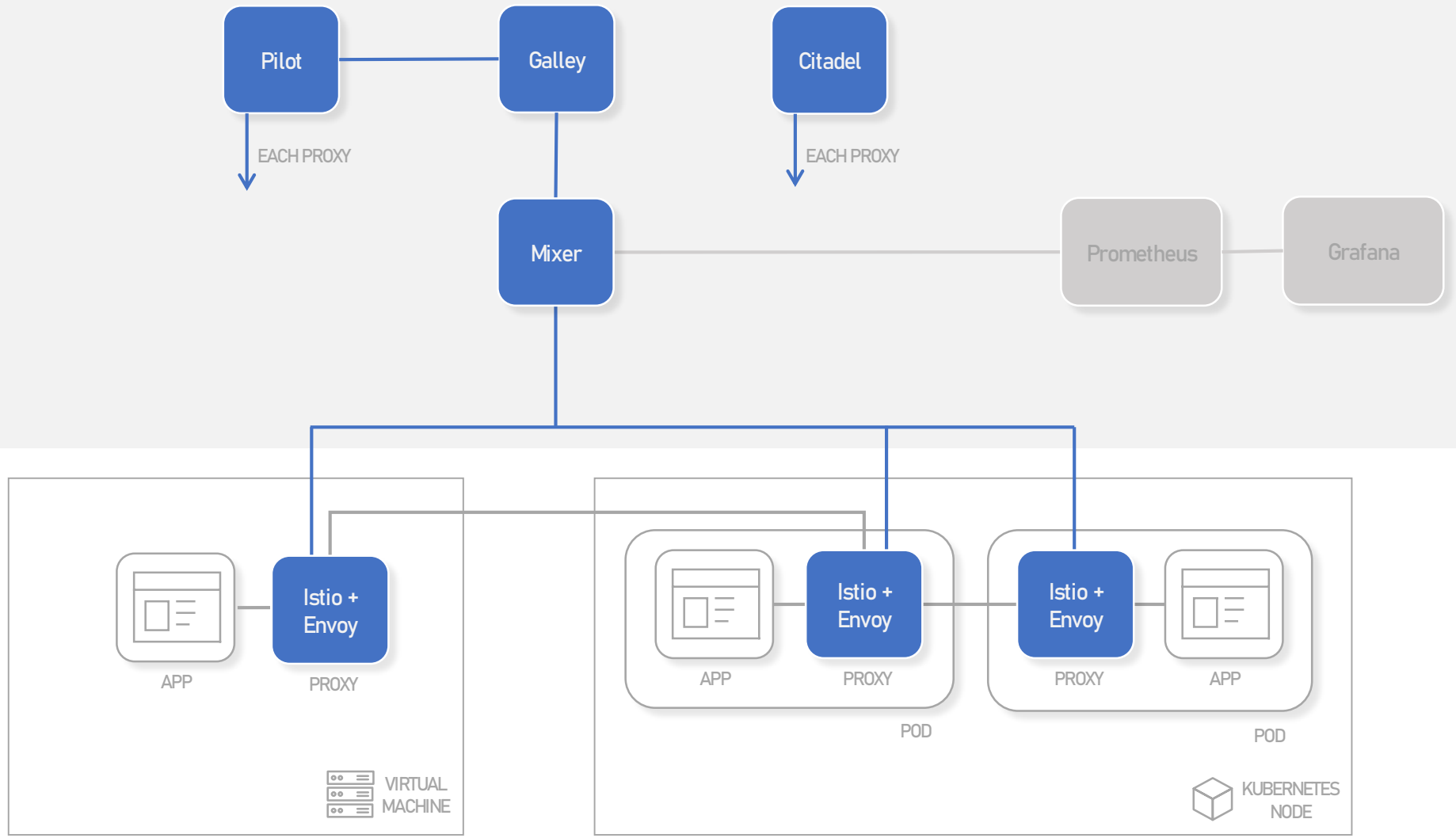
Design Goals

- Maximize Transparency
- Extensibility
- Portability
- Policy Uniformity

Istio

ISTIO TRAFFIC

APPLICATION TRAFFIC



Capabilities

- Mesh – gateways (multi-cluster), vms (expansion)
- Traffic Management – routing, splitting, timeouts, circuit breakers, retries, ingress, egress
- Policy – access control, rate limit, quota, custom policy adapters
- Security – authentication (jwt), authorisation, encryption, external CA (HashiCorp Vault)
- Observability – golden metrics, mirror, tracing, custom adapters, prometheus, grafana

Heads up

- Complex installation (lots of moving parts)
- Resource overhead
- Management of certificates for mTLS

Operational

- Installation via Helm Chart, Operator in the works
- Upgrade via Helm Chart
- Proxy auto injection

Scenarios

- Require extensibility and rich set of capabilities
- Mesh expansion to VM based workloads
- Multi-cluster service mesh


Istio


LINKERD


HashiCorp
Consul


LINKERD

Easy to use and lightweight service mesh

- Announced - December 2017, GA (2.0) in September 2018
- Governance - CNCF
- OSS - Apache 2.0
- Commercial support – Buoyant

Linkerd 1.x announced in February 2016, and went GA (1.0) in April 2017

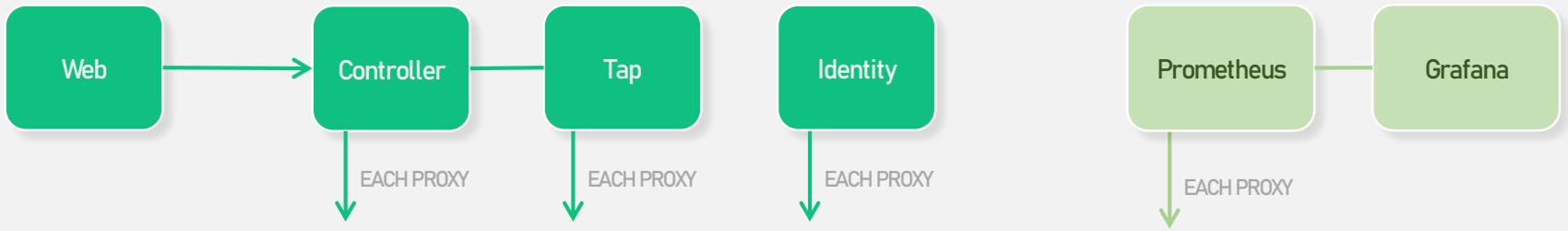
Design Goals

- Keep it simple
- Minimize resource requirements
- Just work

LINKERD

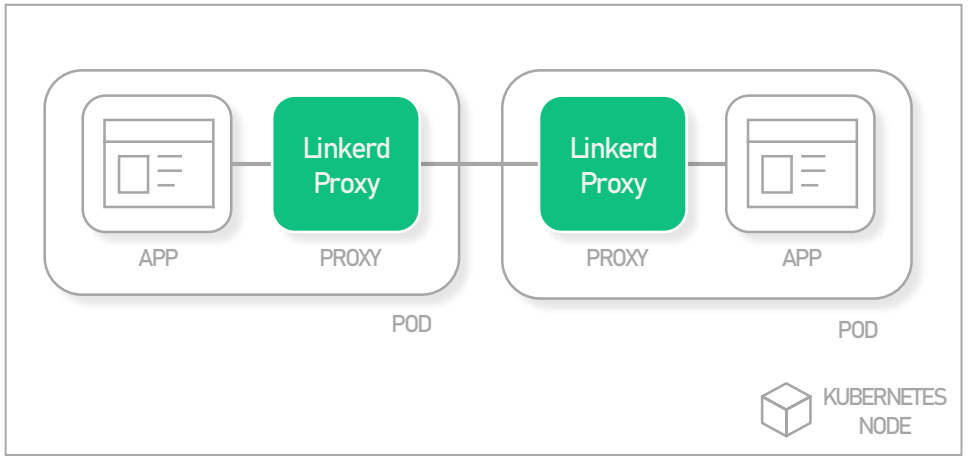
LINKERD TRAFFIC

APPLICATION TRAFFIC



CONTROL PLANE

DATA PLANE



Capabilities

- Mesh – built in debugging option
- Traffic Management – splitting, timeouts, retries, ingress
- Security – encryption, certificates auto-rotated every 24 hours
- Observability – golden metrics, tap, service profiles and per route metrics, web dashboard with topology graphs, prometheus, grafana

Heads up

- Smaller set of features, but growing
- No support for policy (allow/deny) for traffic
- No tracing support

Operational

- Installation via cli (via Helm Chart), allows for pre and post checks too
- Installation can be split for security roles (cluster wide vs namespace components)
- Built in High Availability mode for control plane
- Upgrade built in via cli
- Proxy auto injected

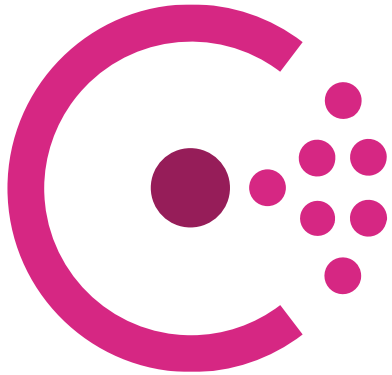
Scenarios

- Simple to use with small set of capability requirements
- Low latency, low overhead, with focus on observability and simple traffic management


Istio


LINKERD


HashiCorp
Consul



HashiCorp

Consul

A multi data centre aware service mesh to connect and secure services across runtime platforms

- Announced - June 2016, GA (1.0) in June 2018
- Governance - HashiCorp
- OSS - Mozilla Public License 2.0
- Commercial support – HashiCorp via Consul Enterprise

Design Principles

- API Driven
- Run and Connect Anywhere
- Extend and Integrate

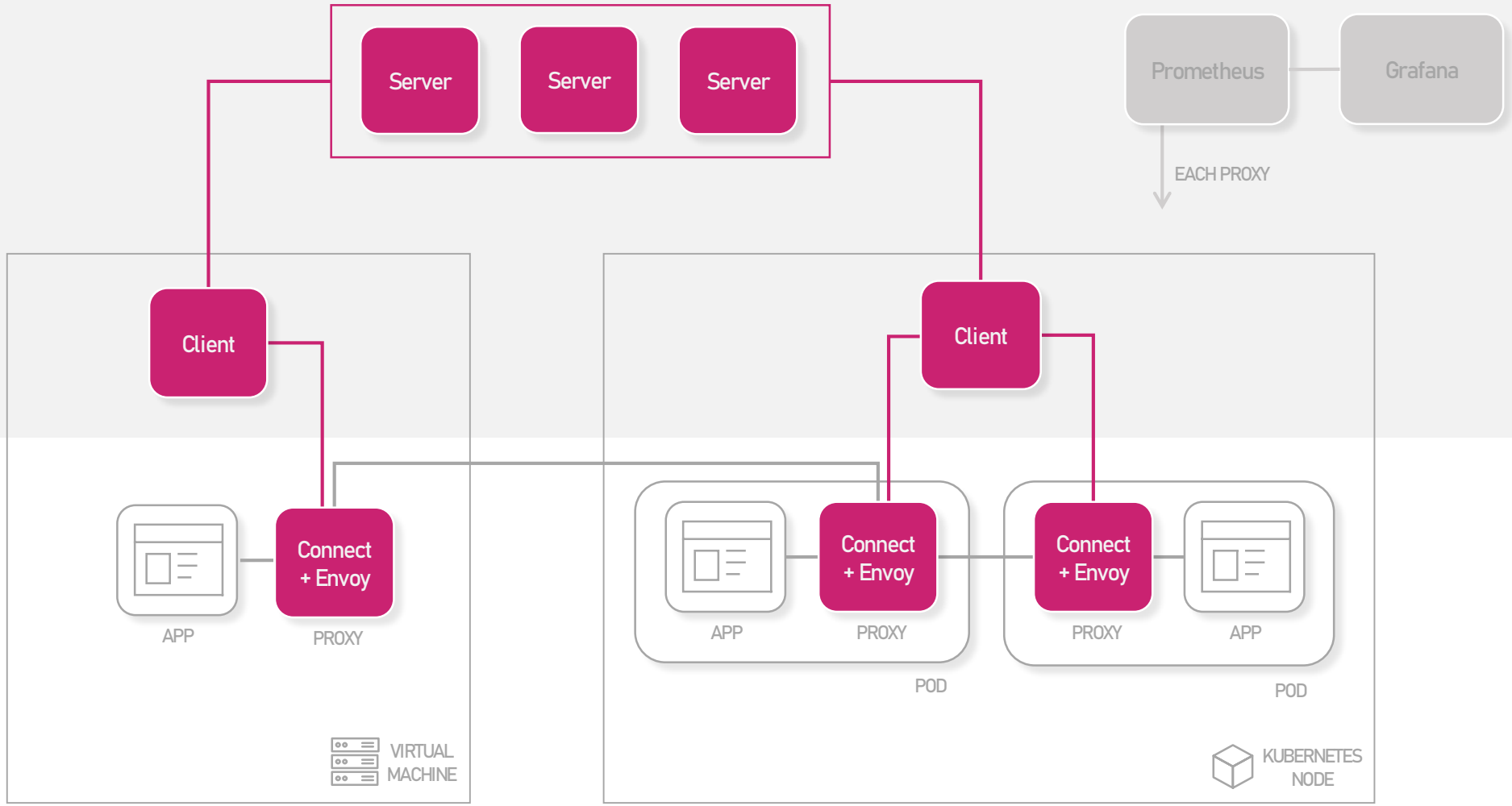
HashiCorp Consul

CONSUL TRAFFIC

APPLICATION TRAFFIC

CONTROL PLANE

DATA PLANE



Capabilities

- Mesh – gateway (multi data centre), vms (out of cluster nodes), service sync, built in debugging
- Proxies – Envoy, built-in proxy, pluggable, l4 proxy available for Windows workloads
- Traffic Management – routing, splitting, resolution
- Policy – intentions, ACLs
- Security – authorisation, authentication, encryption, SPIFFE based identities, external CA (Vault), certificate management and rotation
- Observability – metrics, ui dashboard, prometheus, grafana

Heads up

- Complex installation (lots of Consul specific concepts)
- Need basic understanding of Consul
- No tracing support
- Doesn't use Kubernetes DNS
- Metrics require additional effort to expose
- Proxied services all listen on localhost

Operational

- Installation via Helm Chart
- Upgrade via Helm Chart
- Proxy auto injection

Scenarios

- Extending existing Consul connected workloads
- Compliance requirements around certificate management
- Multi cluster and/or VM based workloads to be included in the service mesh


Istio


LINKERD


HashiCorp
Consul

Benchmarking

Meshery

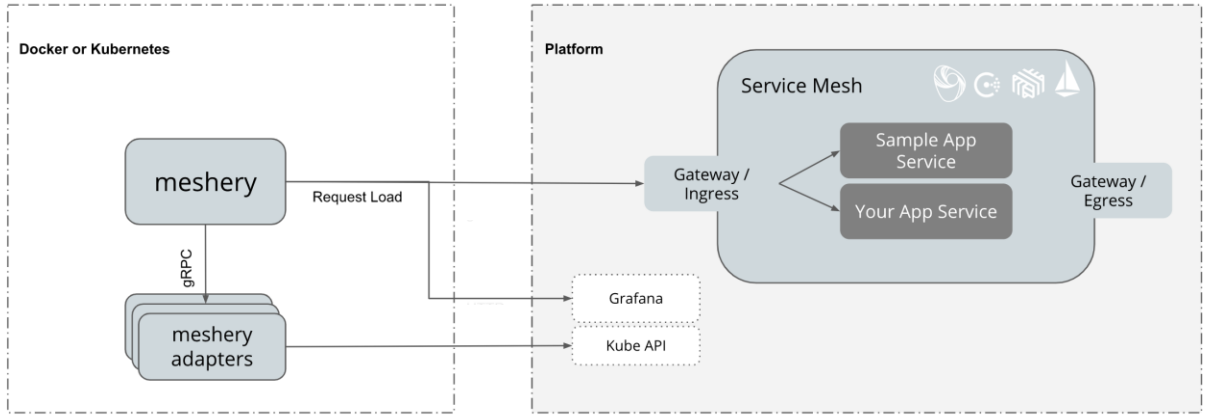
Performance Benchmark

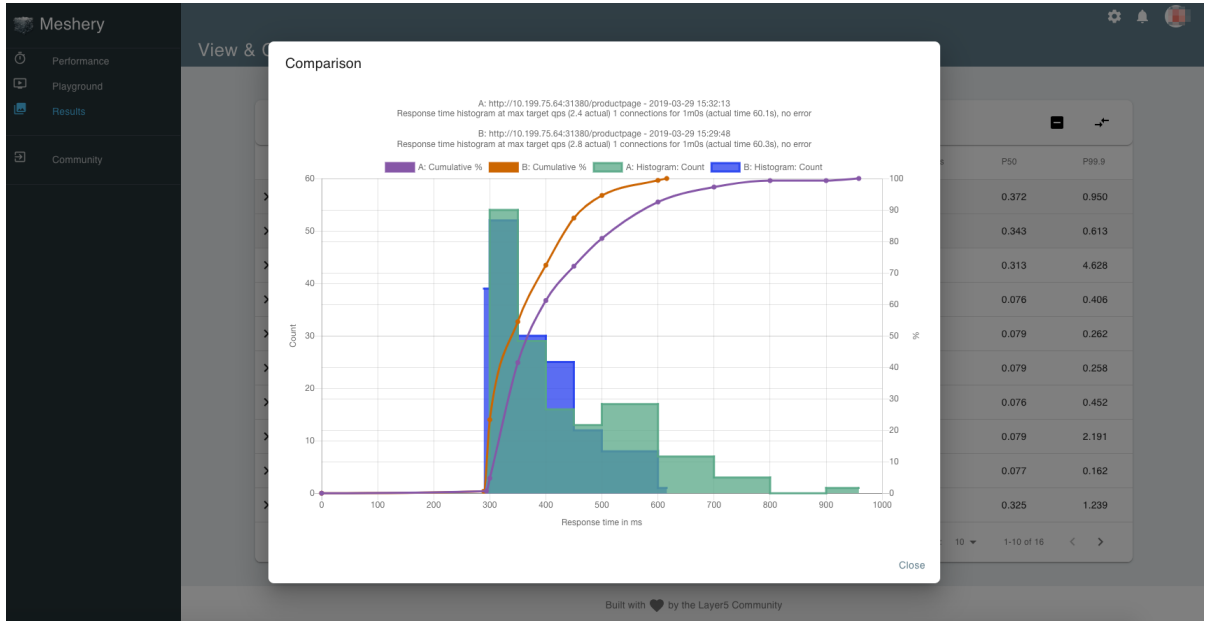
Benchmark the performance of your application across different service meshes and compare their overhead.

Meshery Adapters

- Istio
- Linkerd
- Consul
- Octarine
- Network Service Mesh

https://layer5.io/meshery/





Microsoft

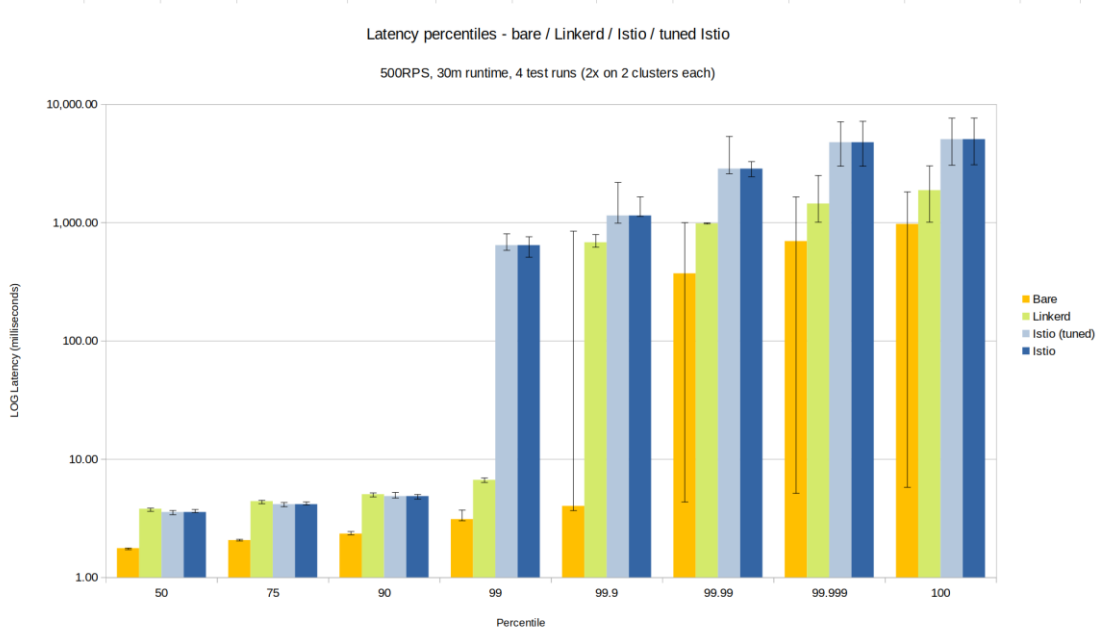
kinvolk.io

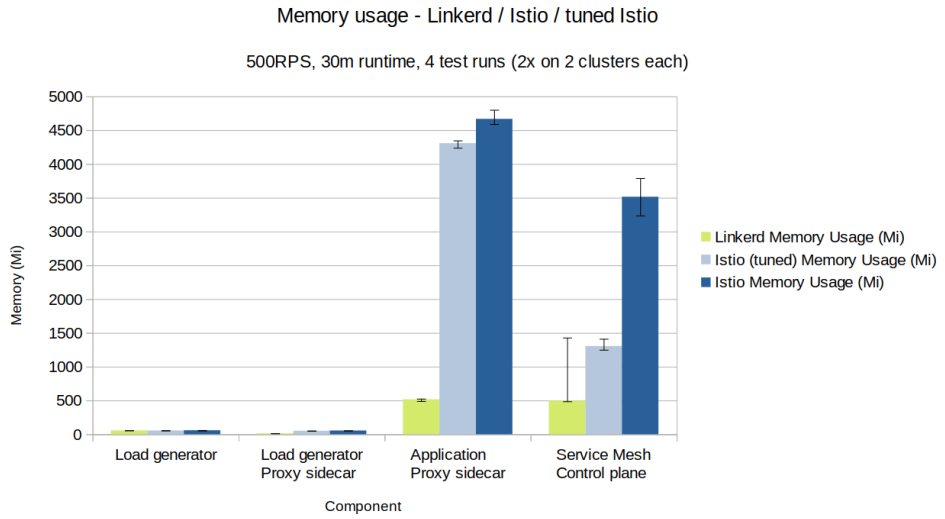
Performance Benchmark Analysis of Istio and Linkerd

Aim was to understand service mesh performance under regular operating conditions of a cluster under load.

Goals

- Provide a reproducible benchmark framework that anyone else can download and use.
- Identify scenarios and metrics that best reflect the operational cost of running a service mesh.
- Evaluate popular service meshes on these metrics by following industry best practices for benchmarking, including controlling for sources of variability, handling coordinated omission, and more.

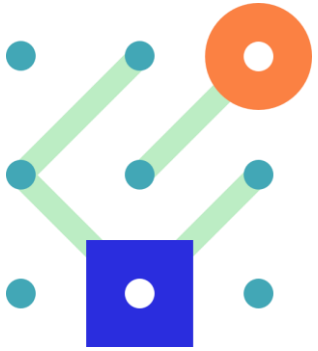




https://kinvolk.io/blog/2019/05/performance-benchmark-analysis-of-istio-and-linkerd/
https://github.com/kinvolk/service-mesh-benchmark

Microsoft

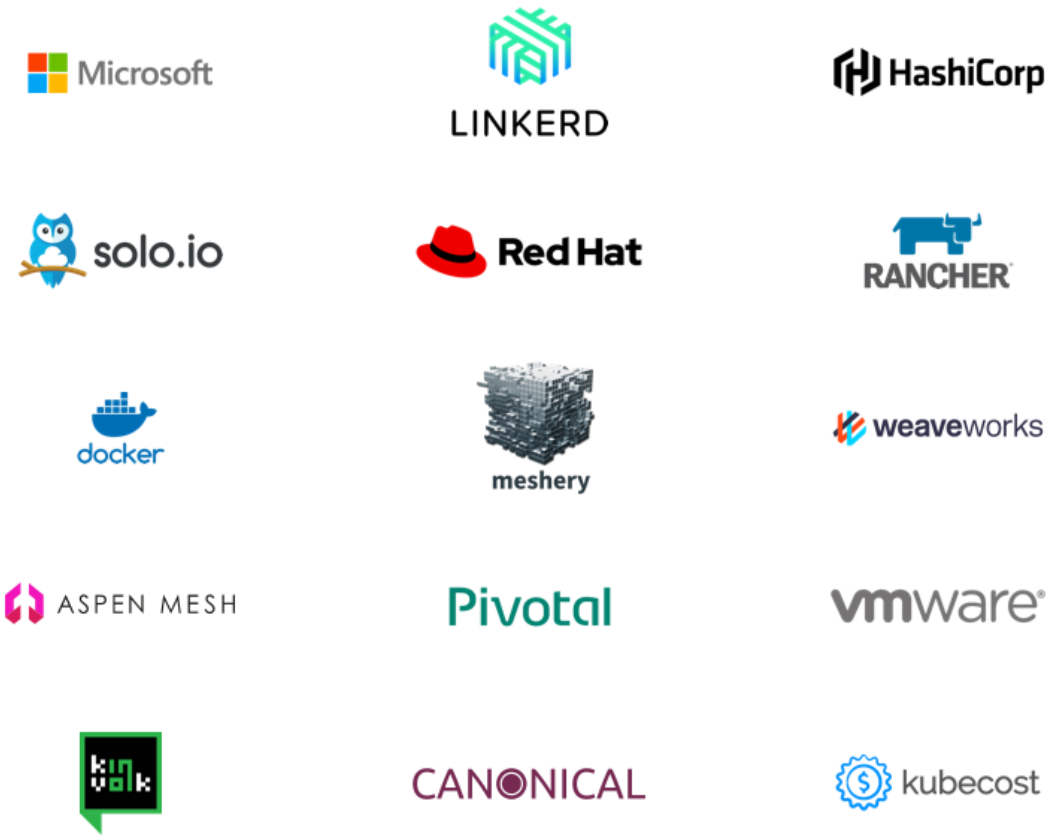
Standardisation



Service Mesh Interface

A standard interface for service meshes on Kubernetes.

- A standard interface for service meshes on Kubernetes
- A basic feature set for the most common service mesh use cases
- Flexibility to support new service mesh capabilities over time
- Space for the ecosystem to innovate with service mesh technology



Service Mesh Interface

Specifications

https://smi-spec.io/

- Traffic Access Control - configure access to specific pods and routes based on the identity of a client for locking down applications to only allowed users and services.
- Traffic Specs - define how traffic looks on a per-protocol basis. These resources work in concert with access control and other types of policy to manage traffic at a protocol level.
- Traffic Split - incrementally direct percentages of traffic between various services to assist in building out canary rollouts.
- Traffic Metrics - expose common traffic metrics for use by tools such as dashboards and autoscalers.

Service Mesh Interface

Early Support

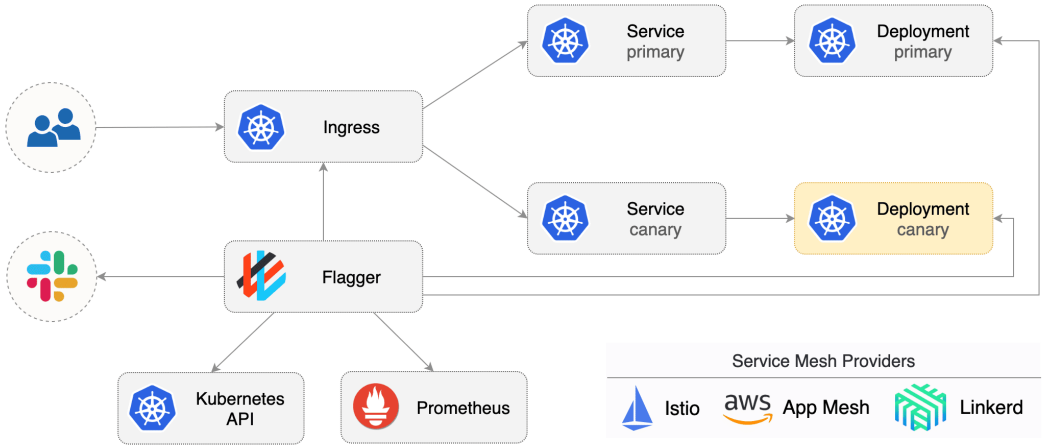
- Istio – Traffic Access Control, Traffic Split
- Linkerd – Traffic Split, Traffic Metrics
- Consul – Traffic Access Control
- Flagger – Traffic Split
- Solo Service Mesh Hub, Kiali, Kubecost

Tooling

weaveworks

Flagger

Automate and manage canary and other advanced deployments with Istio, Linkerd, AWS App Mesh or NGINX for traffic shifting.
Integrated Prometheus metrics control canary deployment success or failure.



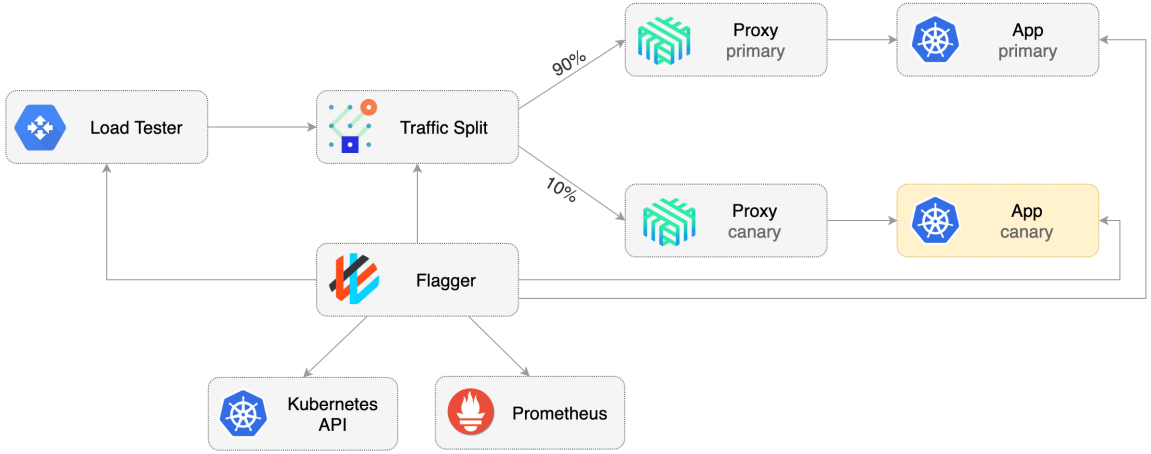
https://www.weave.works/oss/flagger/
https://docs.flagger.app/

Microsoft

weaveworks

Flagger

Automate and manage canary and other advanced deployments with Istio, Linkerd, AWS App Mesh or NGINX for traffic shifting. Integrated Prometheus metrics control canary deployment success or failure.



https://www.weave.works/oss/flagger/
https://docs.flagger.app/

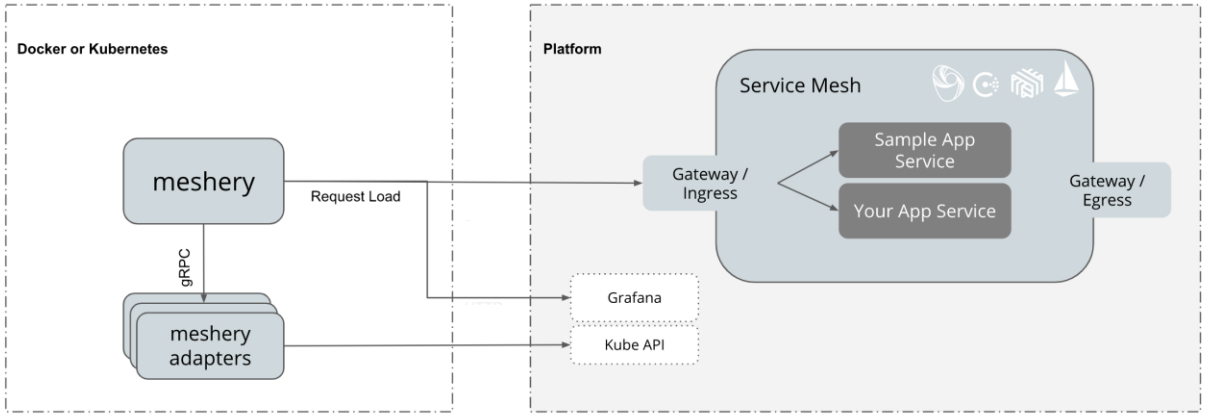
Microsoft

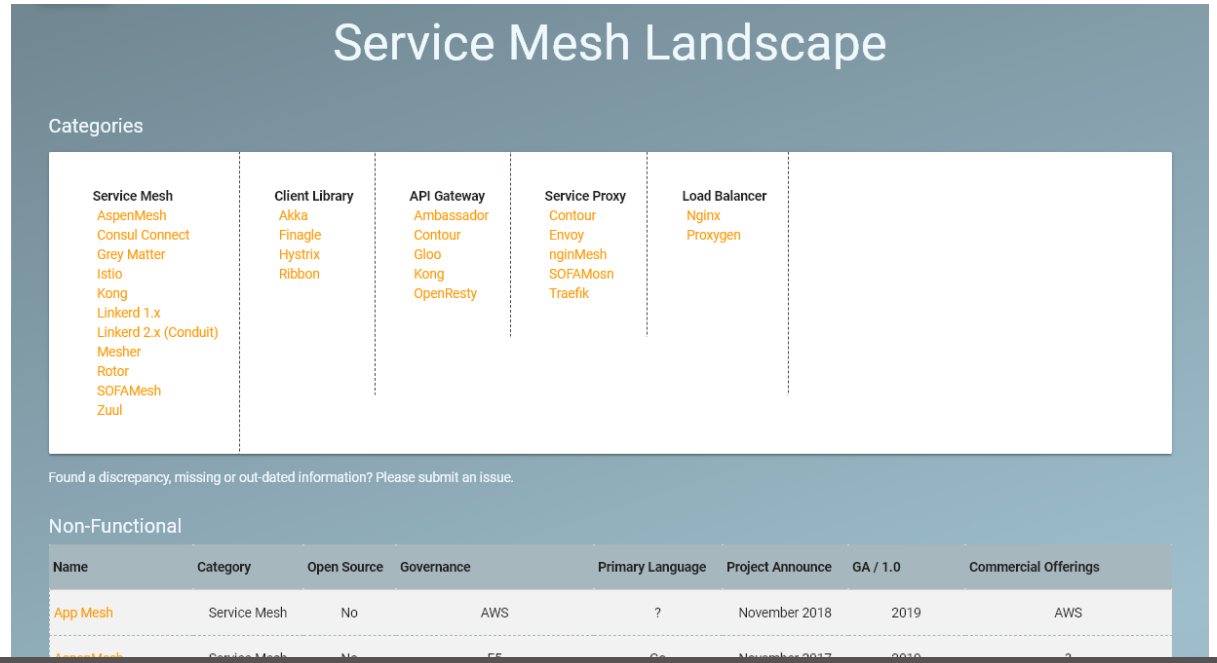
Meshery

Playground

Learn about the functionality of different service meshes and visually manipulate mesh configuration.

https://layer5.io/meshery/
https://layer5.io/landscape/





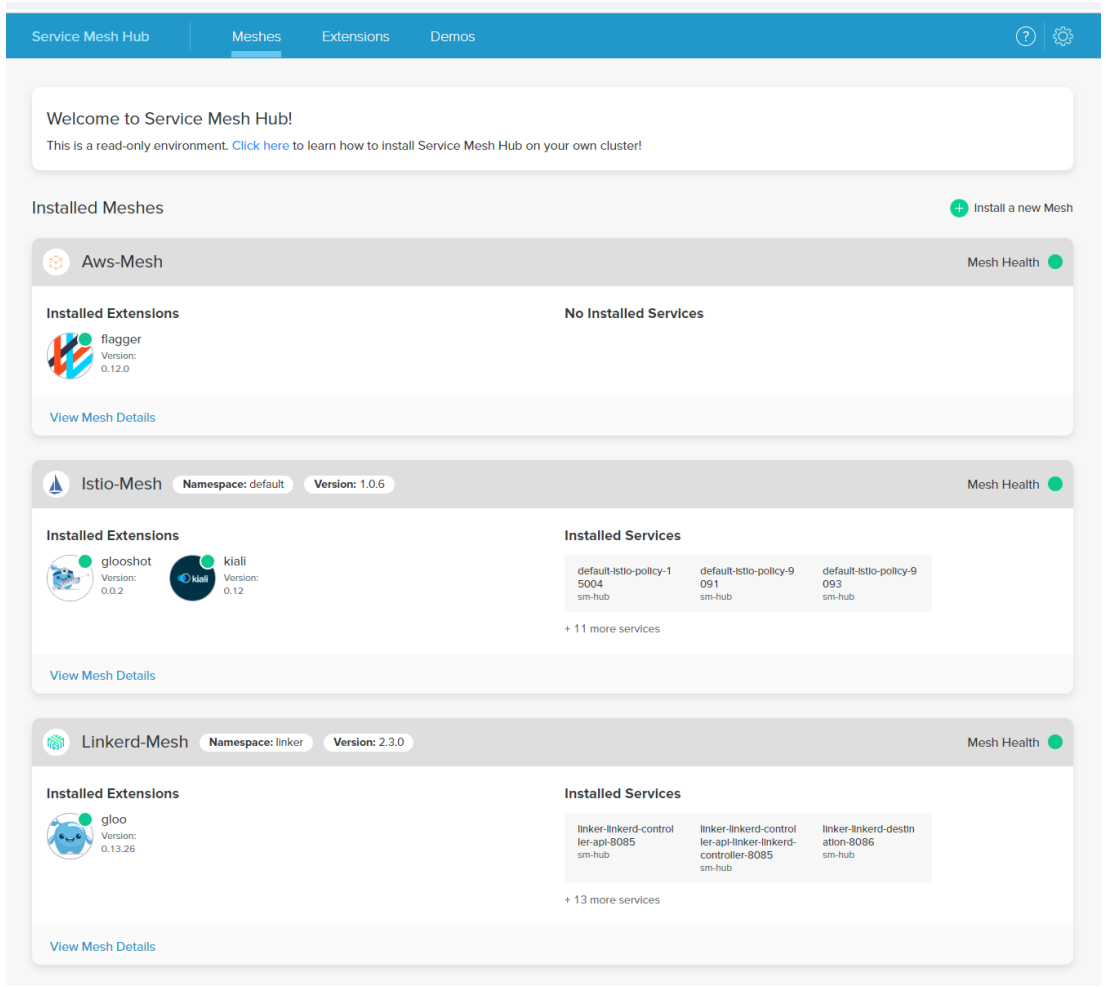
solo.io

Service Mesh Hub

The Service Mesh Hub is an industry hub designed for the community and ecosystem to collaborate on service mesh technology and for organizations to deploy and operate their any service mesh on any cloud.

The Service Mesh Hub simplifies the adoption of service mesh for end users and the ecosystem.

https://www.solo.io/servicemeshhub
https://servicemeshhub.io/meshes/



Takeaways

- Very active space
- Ensure you understand your requirements before selecting a mesh
- Understand the impact of deploying a mesh in your cluster
- Keep an eye on standardisation efforts like Service Mesh Interface (SMI), and tooling like Meshery, Solo Service Mesh Hub, and Flagger