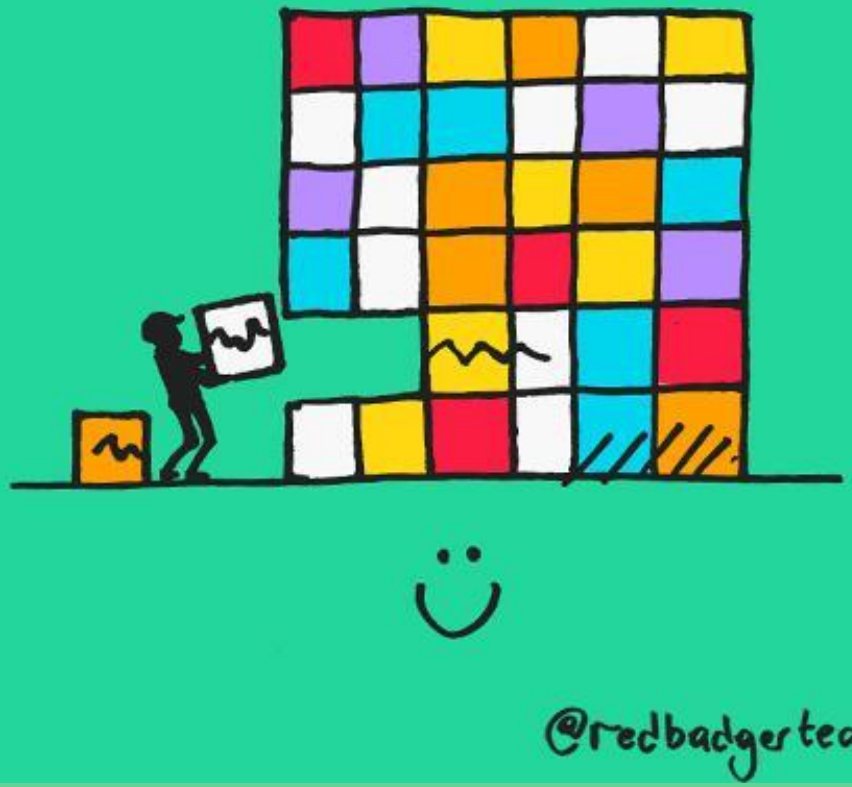
ServiceMesh

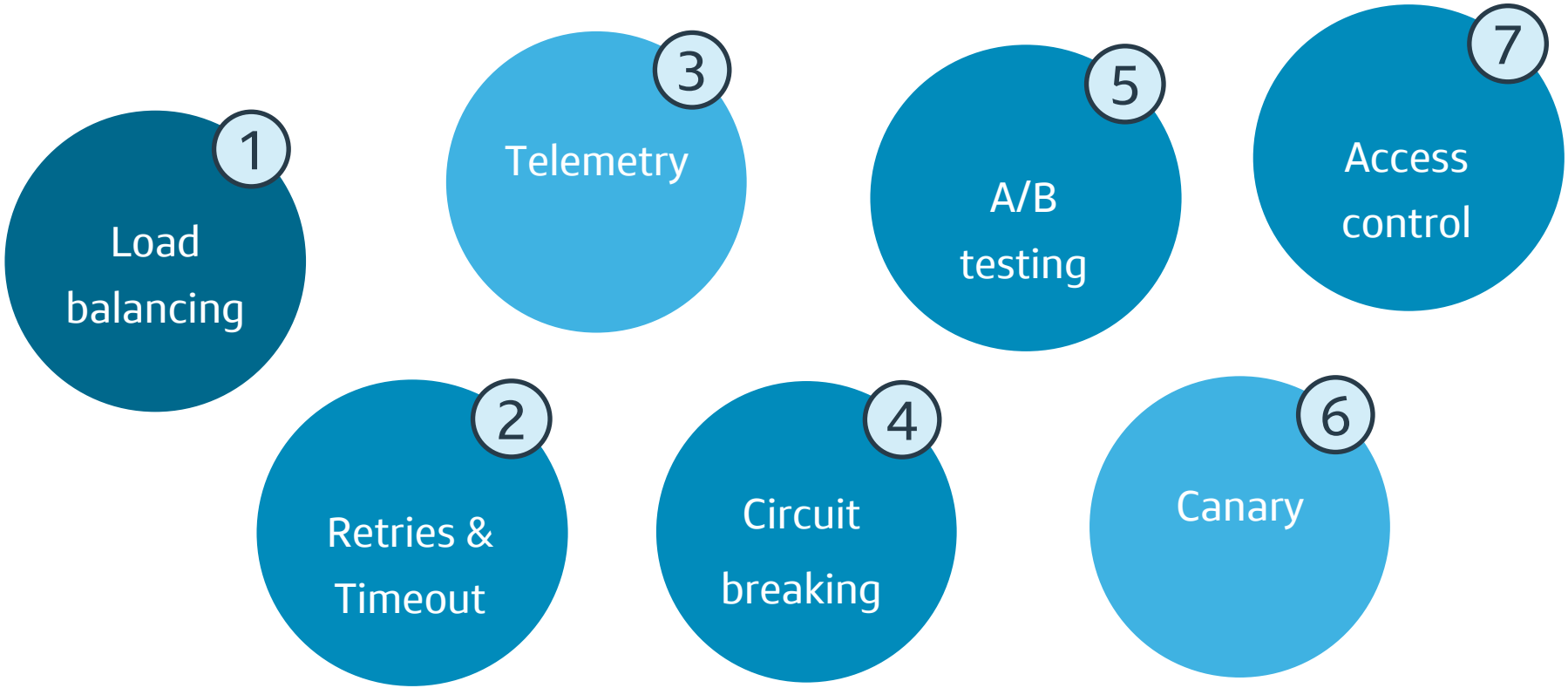
The endgame of Microservice implementation nightmare

Mahesh Veerabathiran

Monolith vs Microservices



Triats of Microservices



Microservices at play

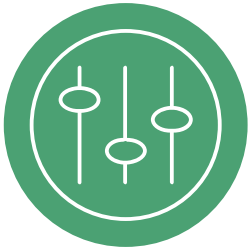
Challenges in cloud-native microservice implementation

- Lack of libraries/utilities for all programming languages. #Polyglot
- Lack of consistency across service implementations.
- Any cross non-functional change requires massive coordination.
- Lack of visibility on end-end performance of services.
- Services are bloated with non-functional features and impacts performance.

Service Mesh *abstracts cross cutting cloud-native features from individual services and make those features available at dedicated infrastructure layer shared by all.*

It provides knobs & levers to secure, control and monitor services with consistency

ServiceMesh - Main players



Control plane

An interface to configure traffic rules, policies and observe behaviors

Data plane

Sidecar proxy intercepts requests/response, enforce configured rules and emit events

Data plane

Envoy is known as Universal data plane

Envoy is used in many control planes such as Istio, AWS AppMesh, Hashicorp consul connect e.t.c.

Data plane is typically a **lightweight proxy** that gets deployed beside the service and assume responsibilities of the following.

- Dynamic service discovery
- Load balancing
- HTTP/2 & GRPC support
- Circuit breaker
- Health checks
- Canary releases
- Staged rollout
- Fault injection e.t.c.

Control plane

- Receives the configuration in human readable format, transform and inject rules in a data plane aware format
- Provides an interface to monitor and manipulate traffic flow through the services.
- Though control plane is integral part of ServiceMesh, it sits outside of the critical path.

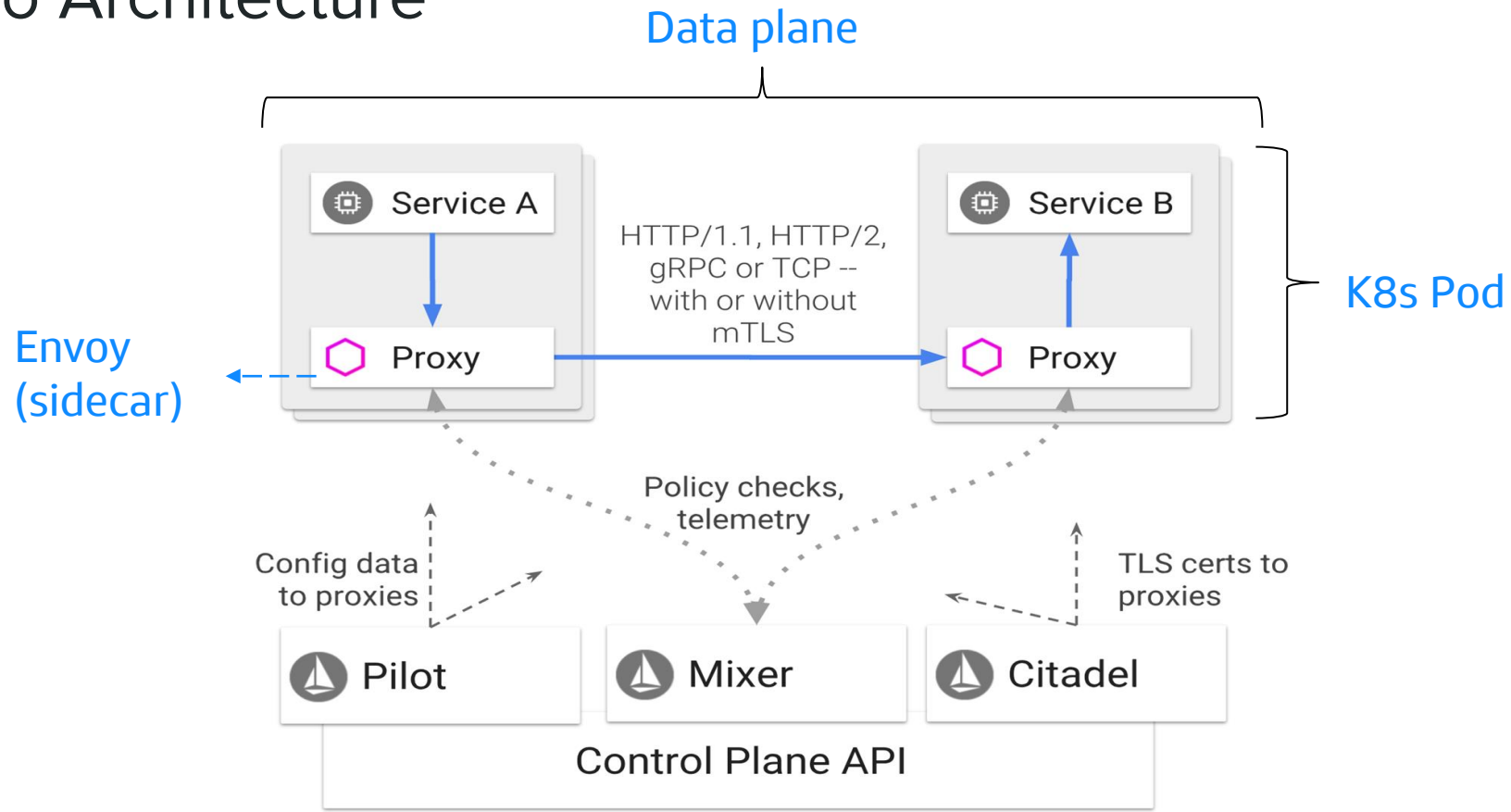
Top Providers:

Istio

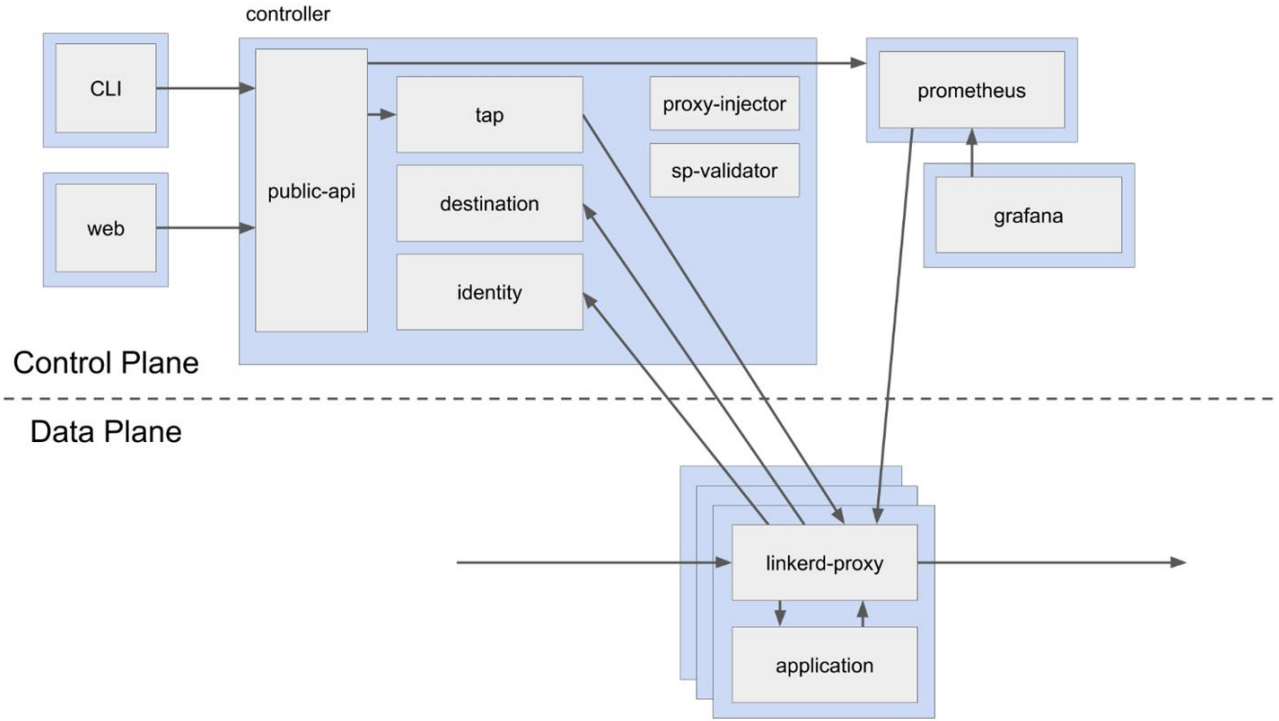
Linkerd

Istio is the leading control plane provider of servicemesh

Istio Architecture



Linkerd Architecture



Architecture

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

The future of ServiceMesh looks bright!

Service Mesh Interface *(SMI) is a specification that covers most common servicemesh capabilities: traffic policy, traffic telemetry & traffic management.*

This is a huge step towards enabling SM provider agnostic. Allows one or more SM from different implementations can together form a nested mesh for the greater enterprise needs

*The current SMI spec is native to Kubernetes (K8s), built on the foundational primitives of K8s.

Thank you!

Twitter: @maheshvra
https://www.linkedin.com/in/aboutmahesh