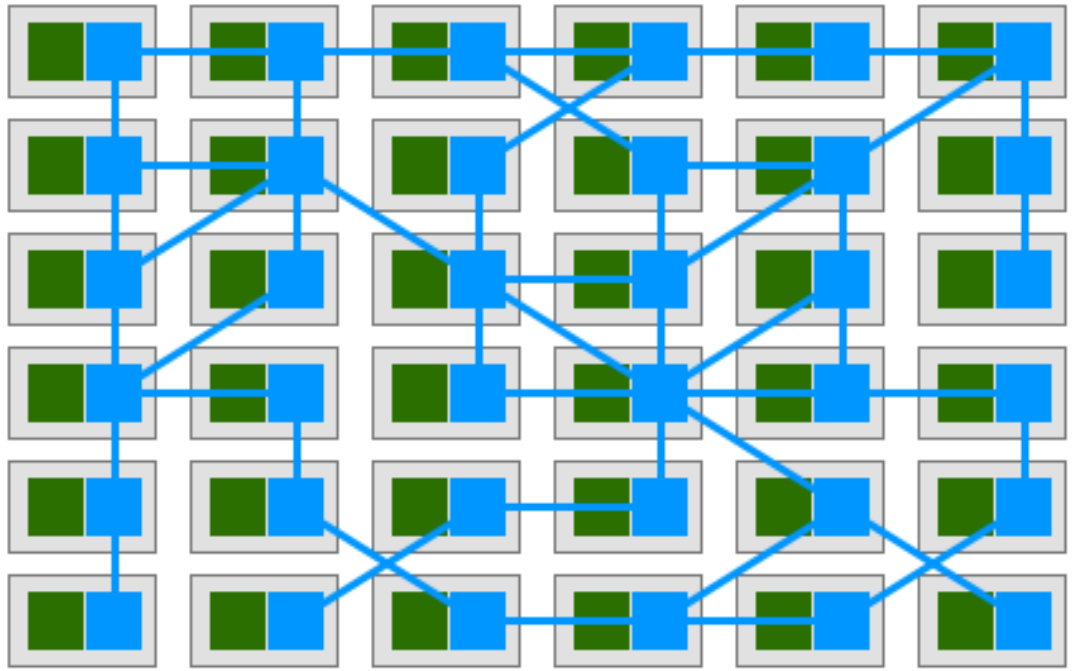
Basics of Service Mesh



Who am I?

Head of Cloud Native Competence Center @ adesso Hungary

- Consulting, training, implementing
- Cloud Native, Kubernetes, Microservices, DevOps
- Previously co-founder @ LeanNet

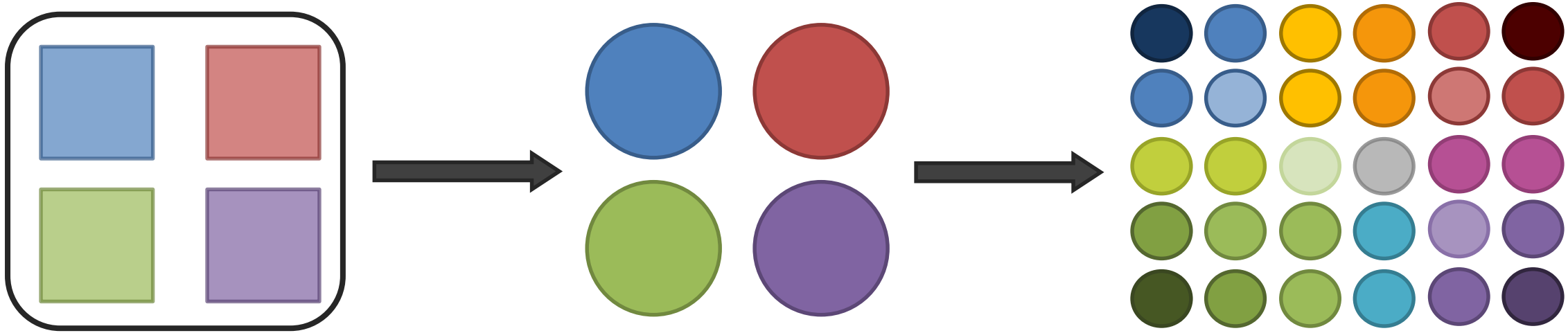
peter.megyesi@adesso.eu

twitter.com/M3gy0

linkedin.com/in/M3gy0

What are Microservices?

Microservices architecture is software development form that structures an application as a collection of **loosely coupled services** having **bounded context**, which implement business capabilities. Microservices enable the **continuous delivery/deployment** of large, complex applications.



Monolithic software

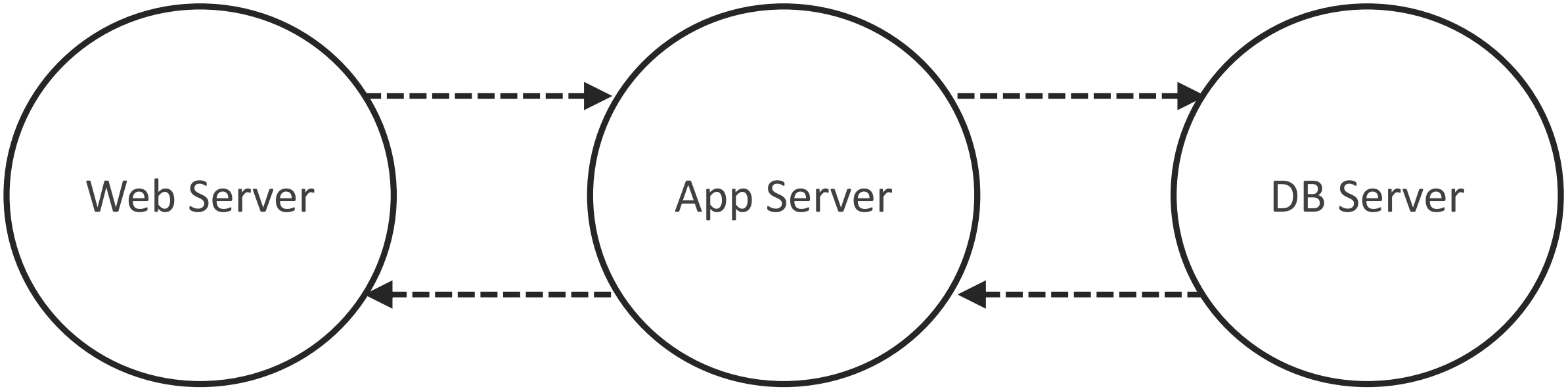
- Vertically scalable
- Hard to maintain and evolve
- Very long build / test / release cycles
- Always fixing bugs
- Lack of innovation

Service Oriented Architecture

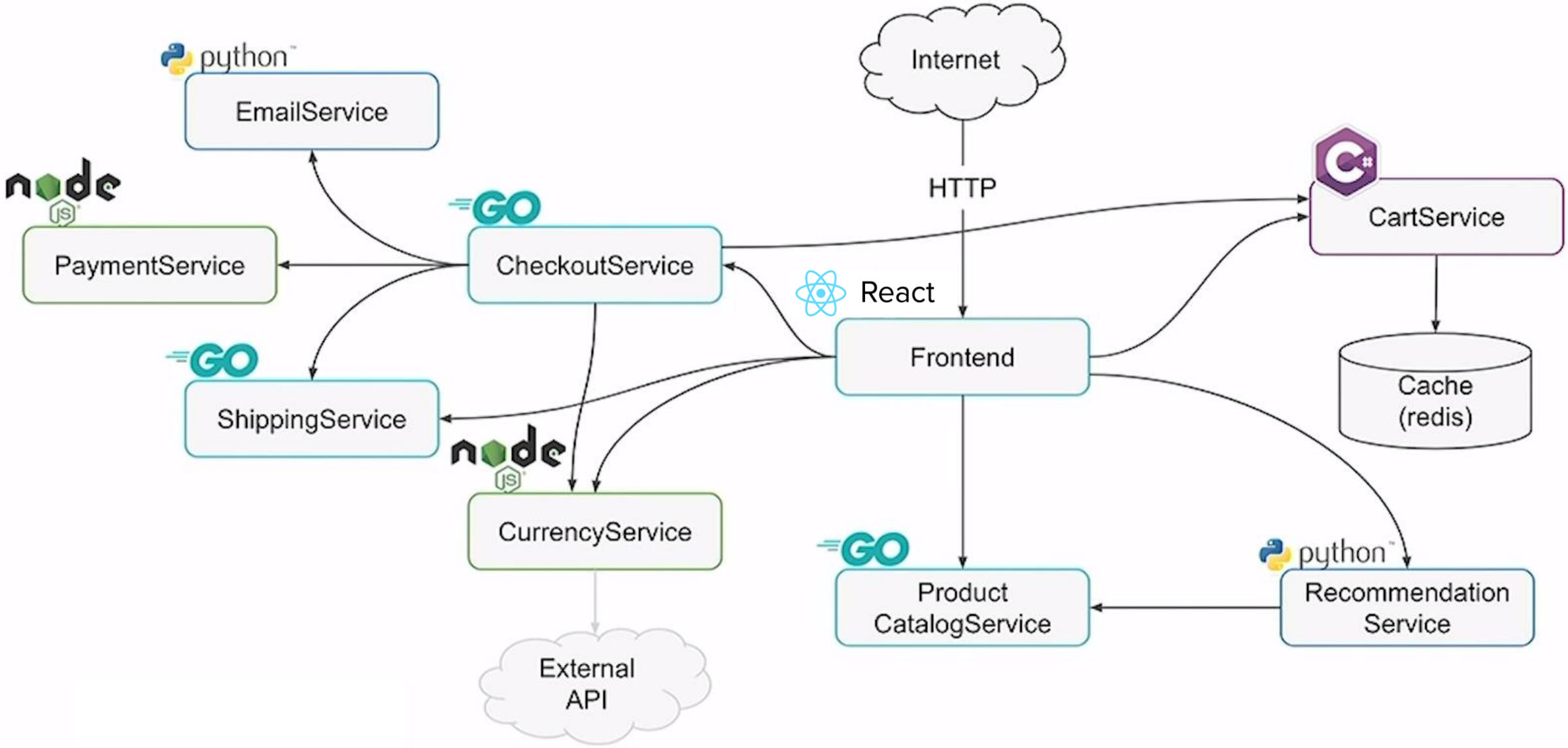
Microservices

- Horizontally scalable
- Services are easy to maintain
- Very short build / test / release cycles
- Easy to innovate

This is not a Microservice Architecture!

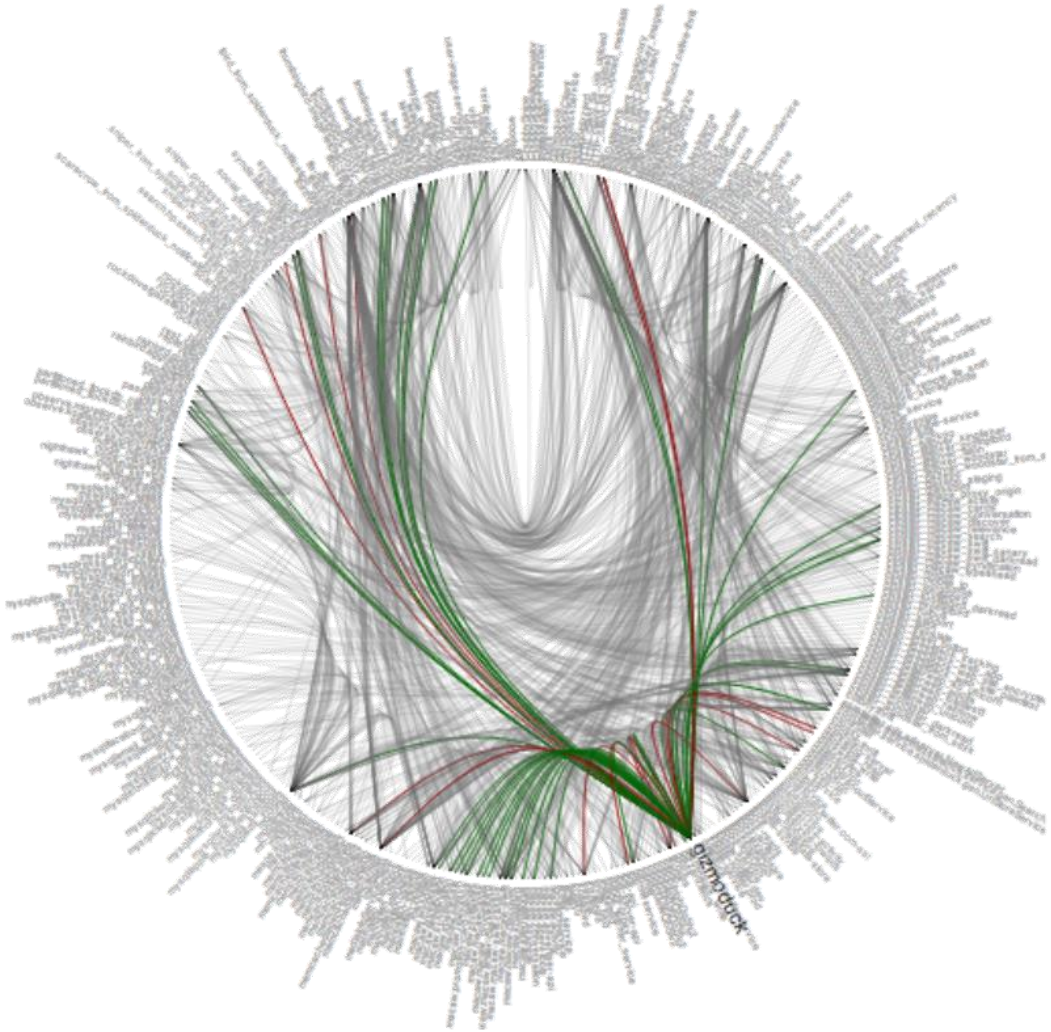


This is Getting There….

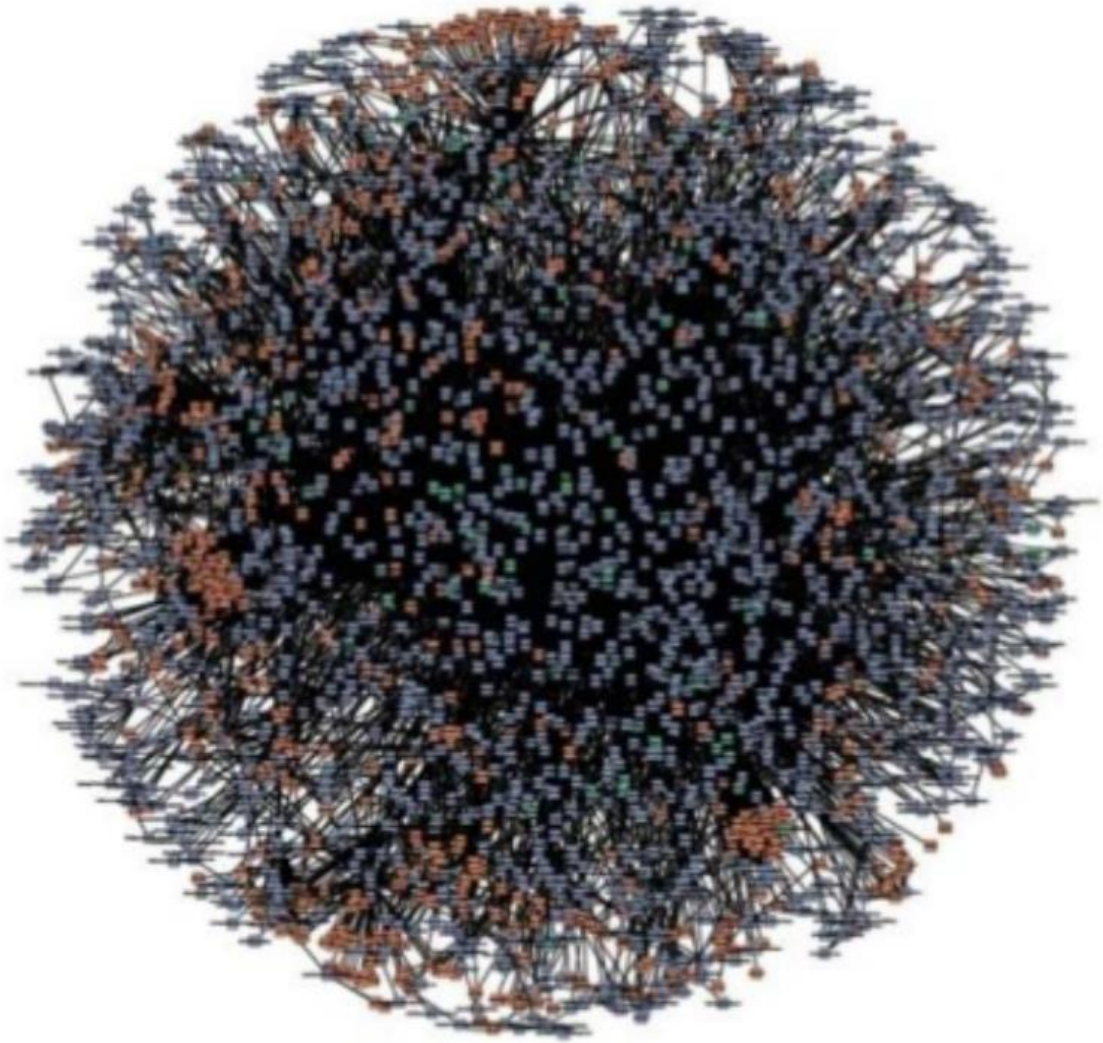


But These are True Microservice Architectures!

Twitter

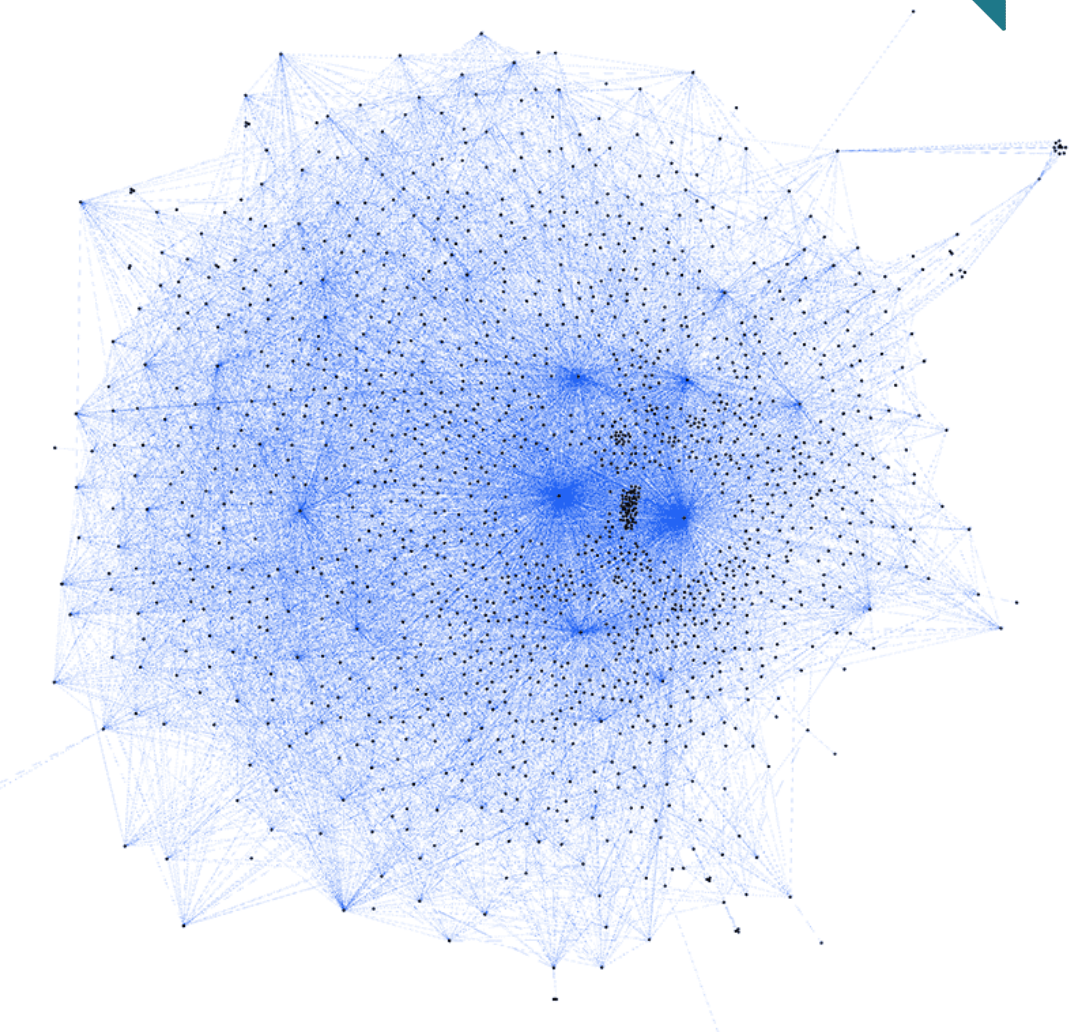


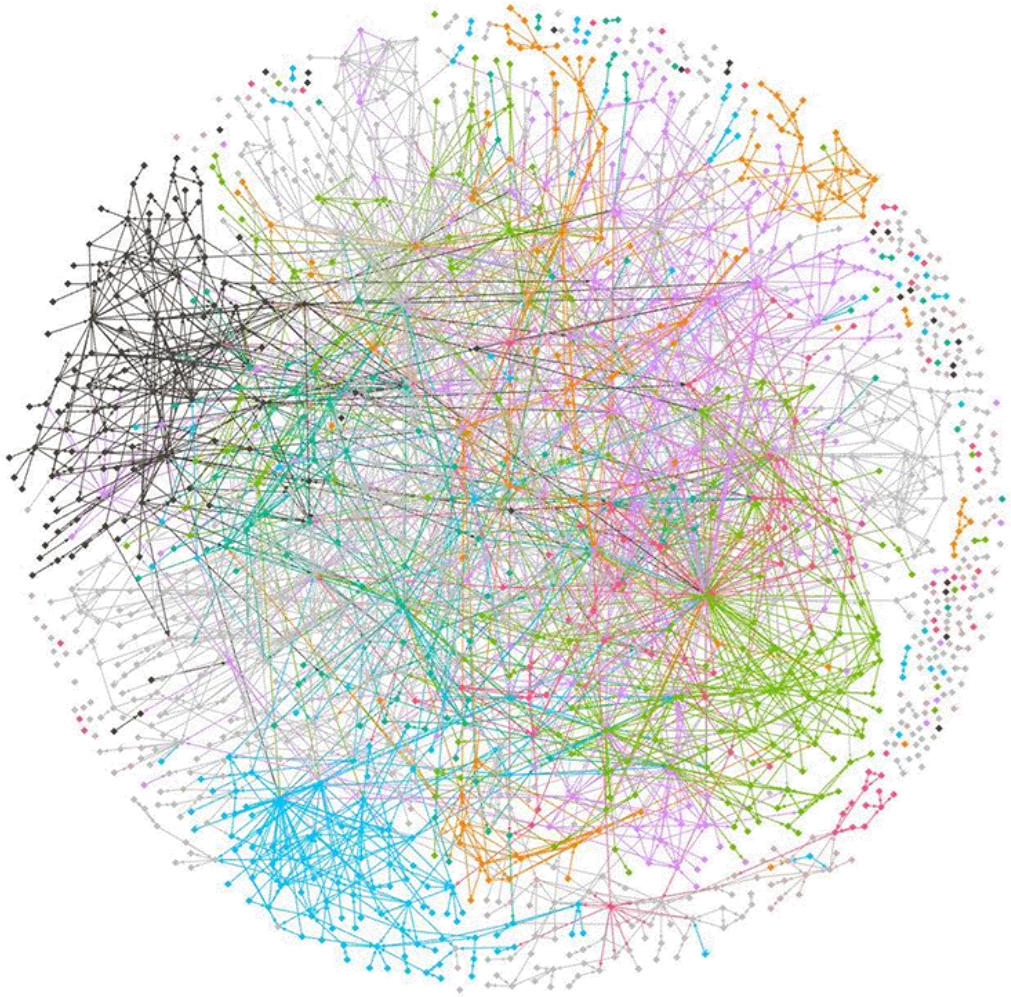
Amazon Web Service



But These are True Microservice Architectures!

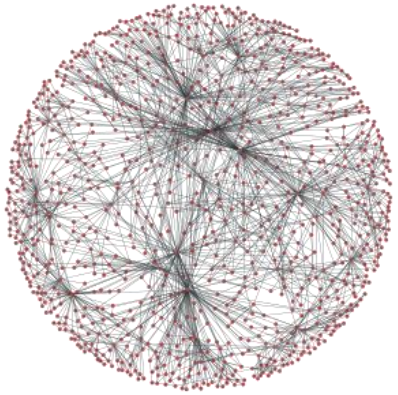
monzo





Monolith vs. Microservices?

Complex application
Easy networking



Easy applications (services)
Complex networking

The fallacies of distributed computing:

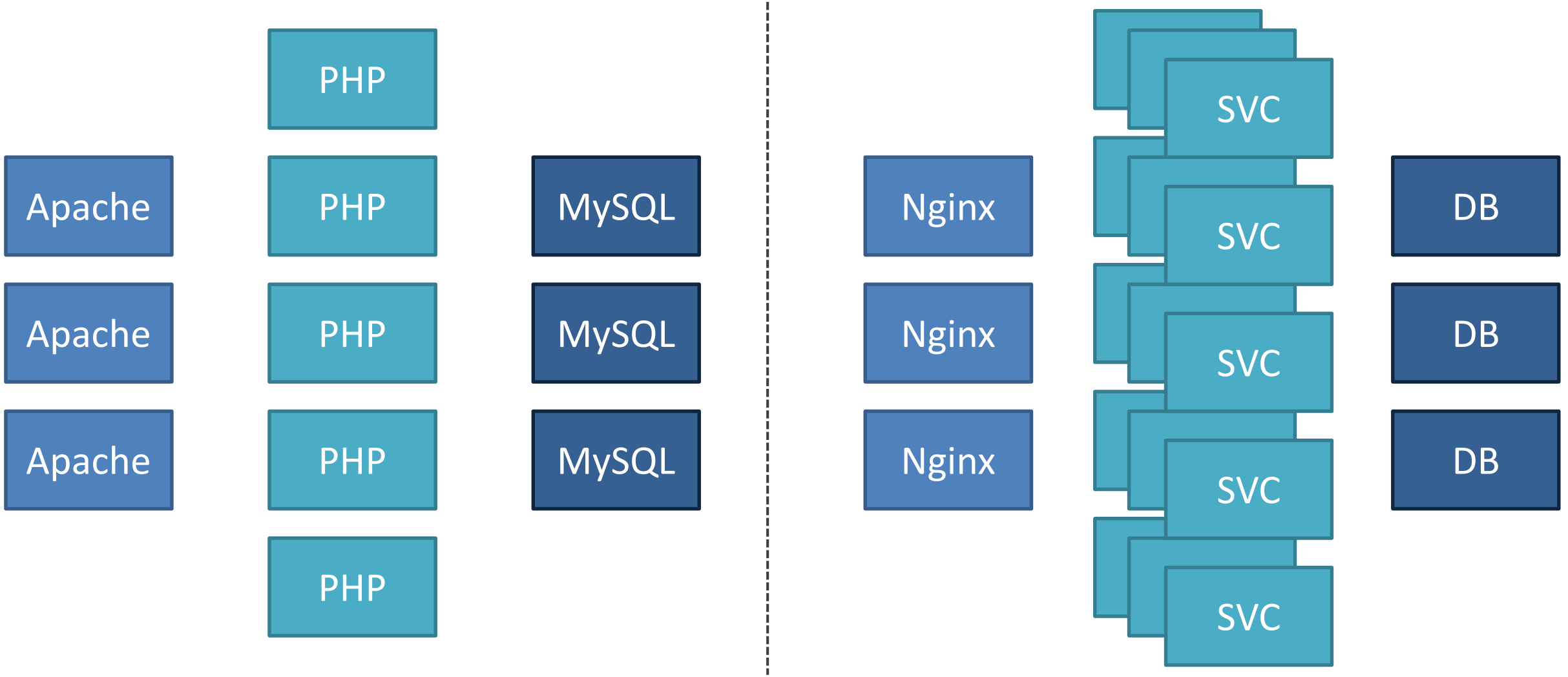
- The network is reliable
- Latency is zero
- Bandwidth is infinite
- The network is secure
- Topology doesn't change
- There is one administrator
- Transport cost is zero
- The network is homogeneous

https://en.wikipedia.org/wiki/Fallacies_of_distributed_computing

What is a Service Mesh?

Service Mesh is **dedicated infrastructure layer** in a **microservices environment** to consistently **manage, monitor** and **control** the **communication** between services across the entire application

Evolution: LAMP to Web Scale



Evolution: Common Features in DevSecOps

- Dynamic service discovery
- Load balancing
- Health checks
- Timeouts
- Retries
- Circuit breakers

- Traffic encryption (mTLS)
- Fine-grained access control
- Access auditing
- Rate limiting
- Rewrites and redirects

- Consistent metrics
- Access logs
- Distributed tracing
- Fault injection

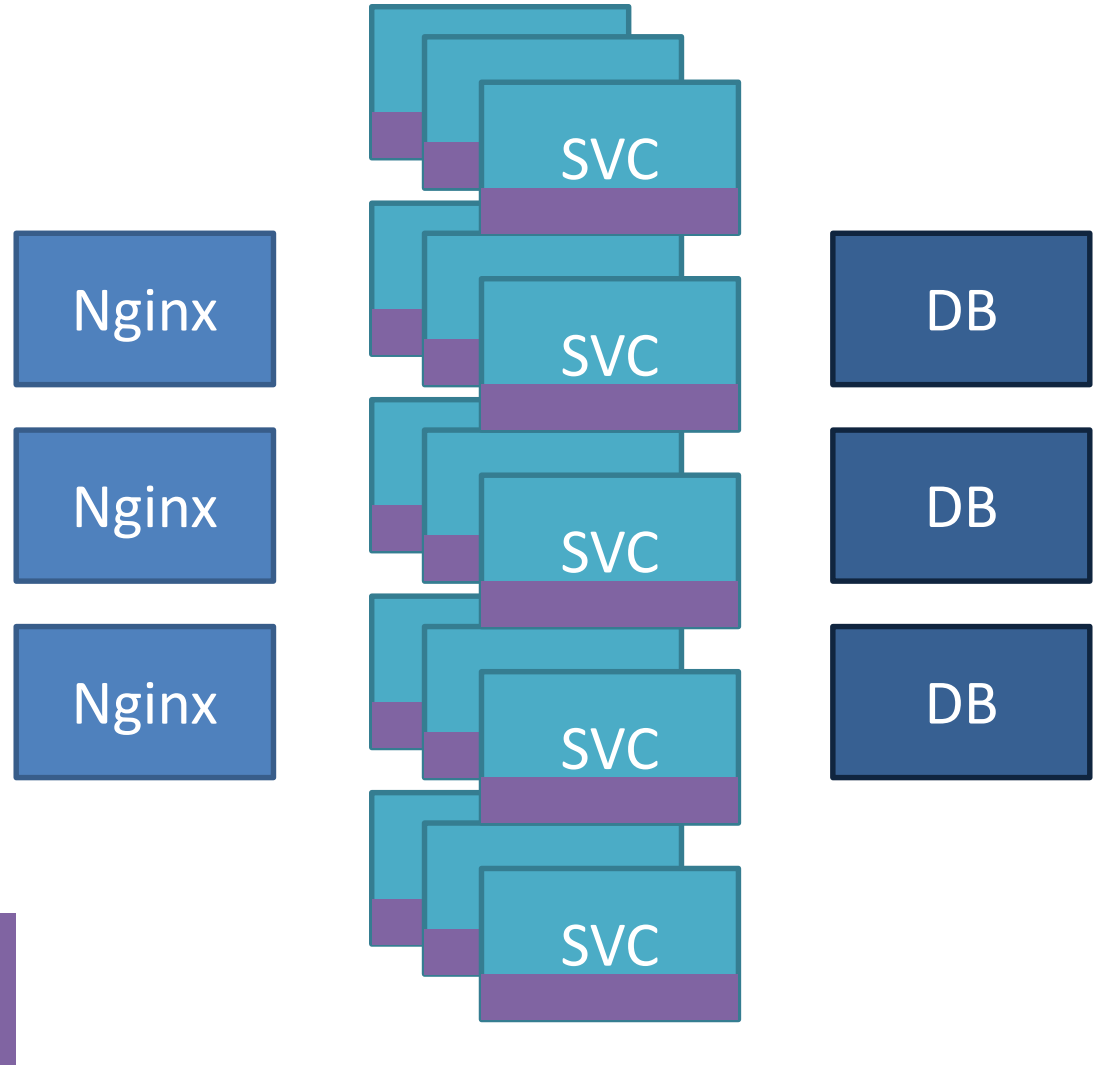
Evolution: Shared Libraries to Service Mesh

Examples for such fat libraries:

- Hystrix @ Netflix
- Stubby @ Google
- Finagle @ Twitter

Disadvantages of shared libraries:

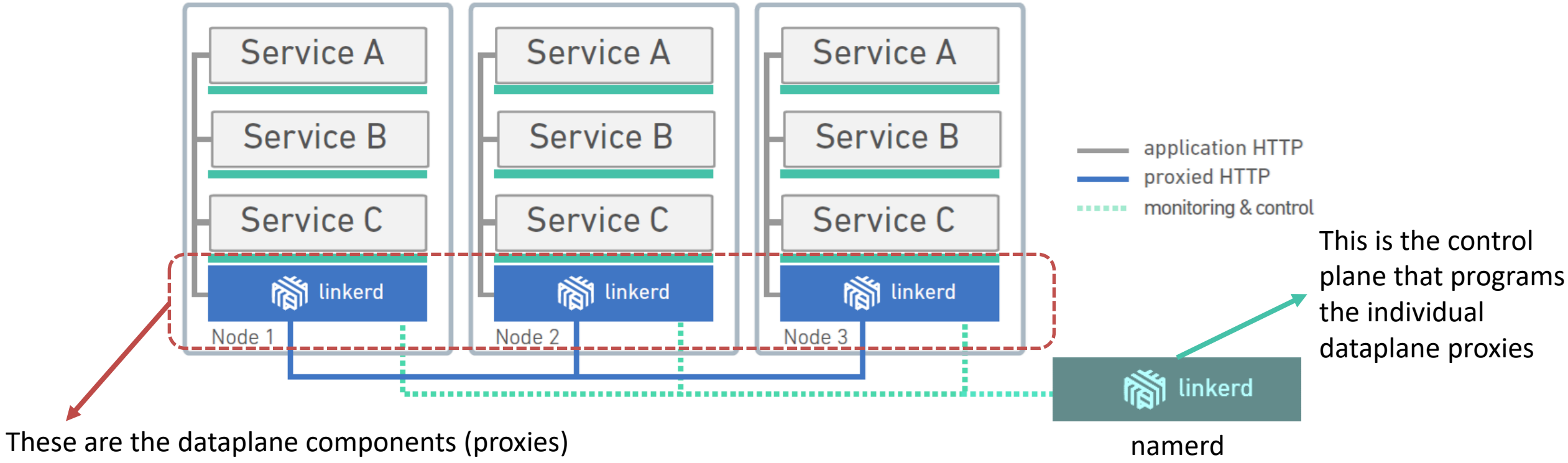
- Have to be implemented in multiple languages
- If the library changes the entire service has to be redeployed
- Too tight involvement of dev teams



Linkerd

A service mesh that adds reliability, security, and visibility to cloud native applications

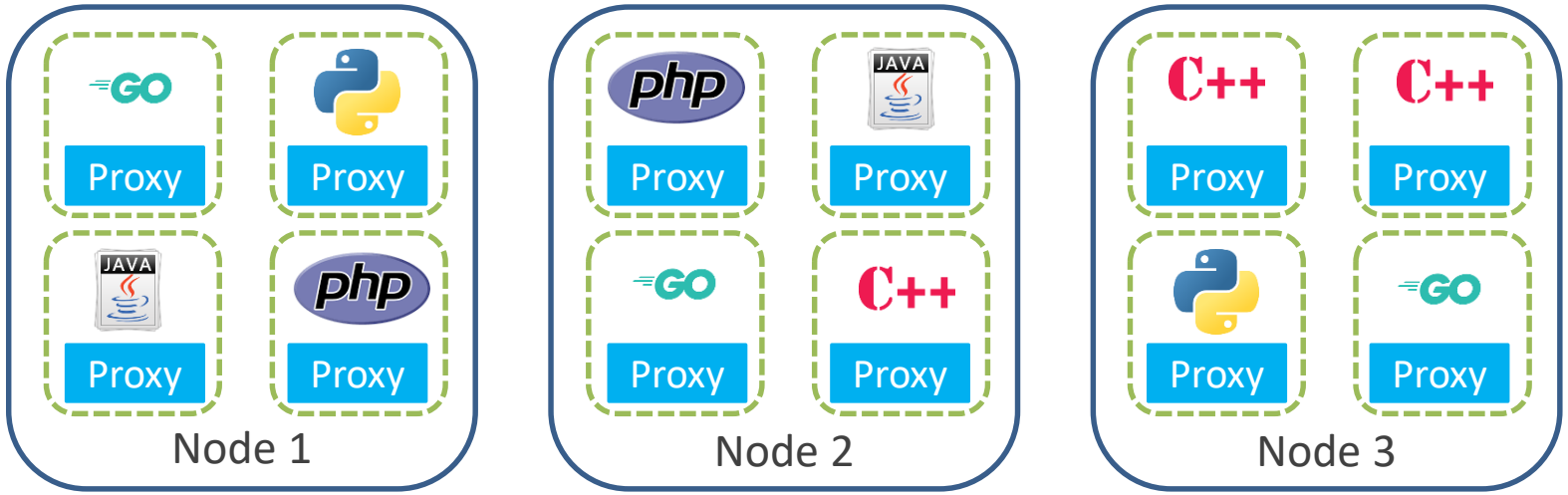
- Official CNCF Project
- Originally created by Buoyant Inc. based on Finagle
- Written in JAVA



Sidecar Model in Container Environments

Disadvantages of per-node model

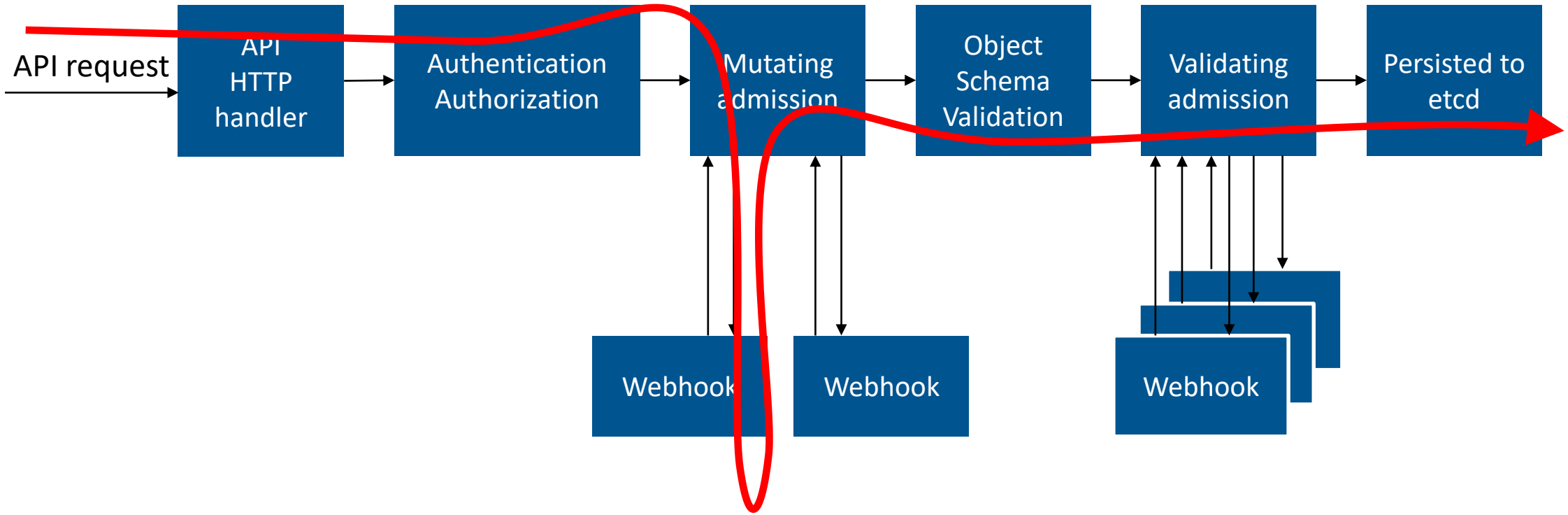
- Raises security concerns in multi-tenant environments (shared TLS secrets, common authentication, etc.)
- Can only be scaled vertically, not horizontally (give it more memory and CPU and it will handle more connection)
- Not optimized for container workloads



The sidecar model

- Put a proxy next to every container
- This is supported by the POD abstraction in Kubernetes
- Linkerd is considered to be too heavy for such environment!

Sidecar Unjection: Using the Mutating Webhook of the API Server



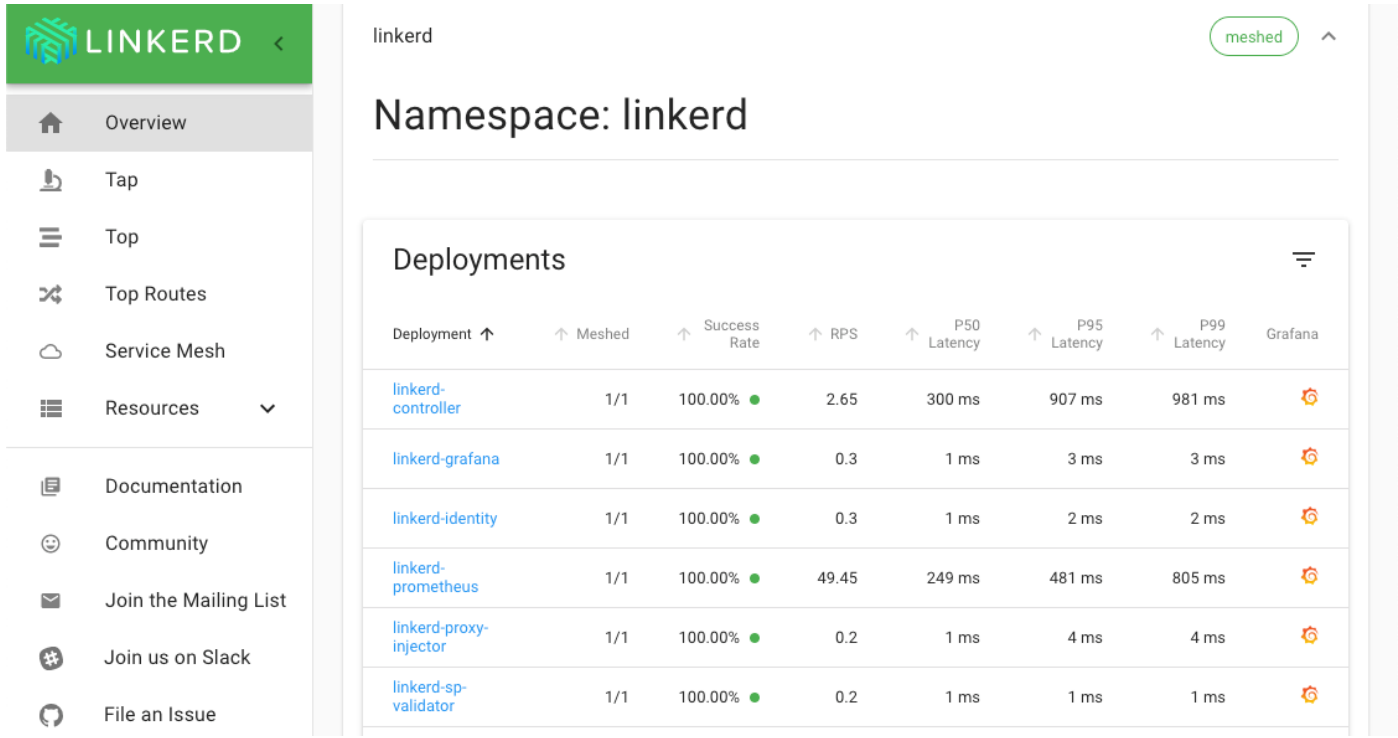
Happens on pod level if annotation is present

- Namespace
- Pod controller
  - Deployment, ReplicaSet, DaemonSet, StatefulSet
- Pod itself

Linkerd2

A novel service mesh that was specifically designed to work in Kubernetes

- Created by Buoyant, the same team who created Linkerd
- Data plane is written in Rust to be very fast and lightweight to sidecar operations (~5MB container size)
- Control plane is written in Go to work well in Kubernetes
- Can be deployed service-by-service (it's not an all-or-nothing choice…)



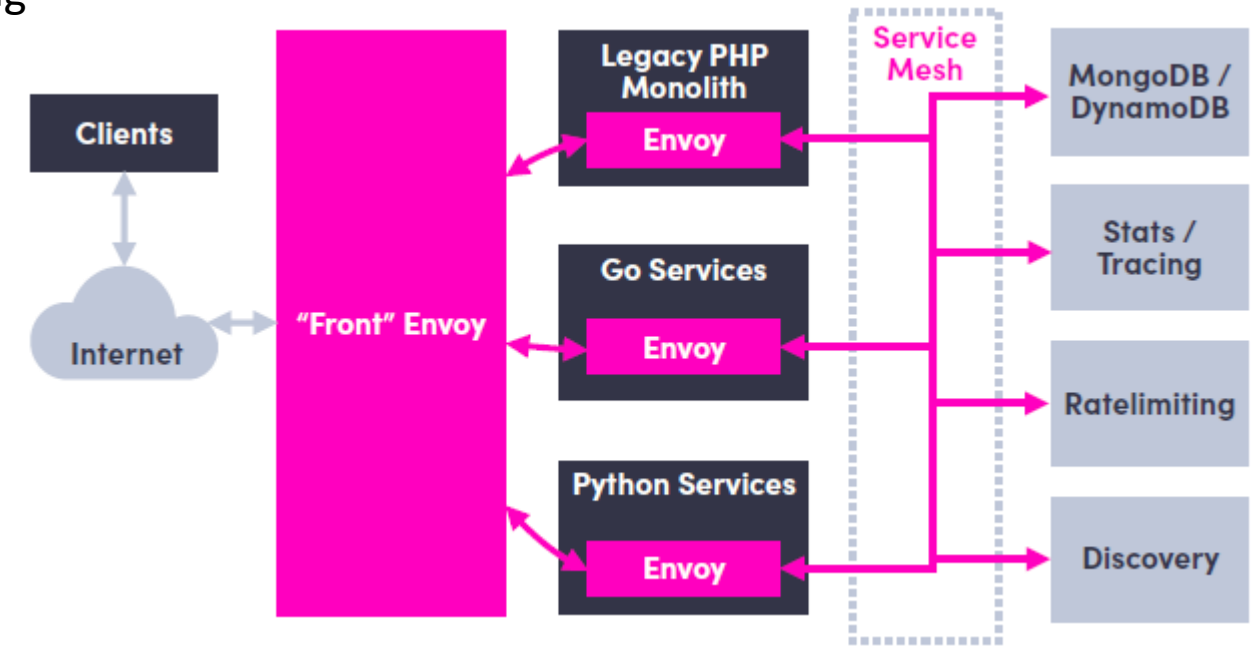
Envoy

Envoy is an open source edge and service proxy, designed for cloud-native applications

- Official CNCF Project
- Originally created by Lyft
- Written in C++
- Out of process architecture with advanced threading
- Best in class observability
- Rich APIs called xDS

Features include

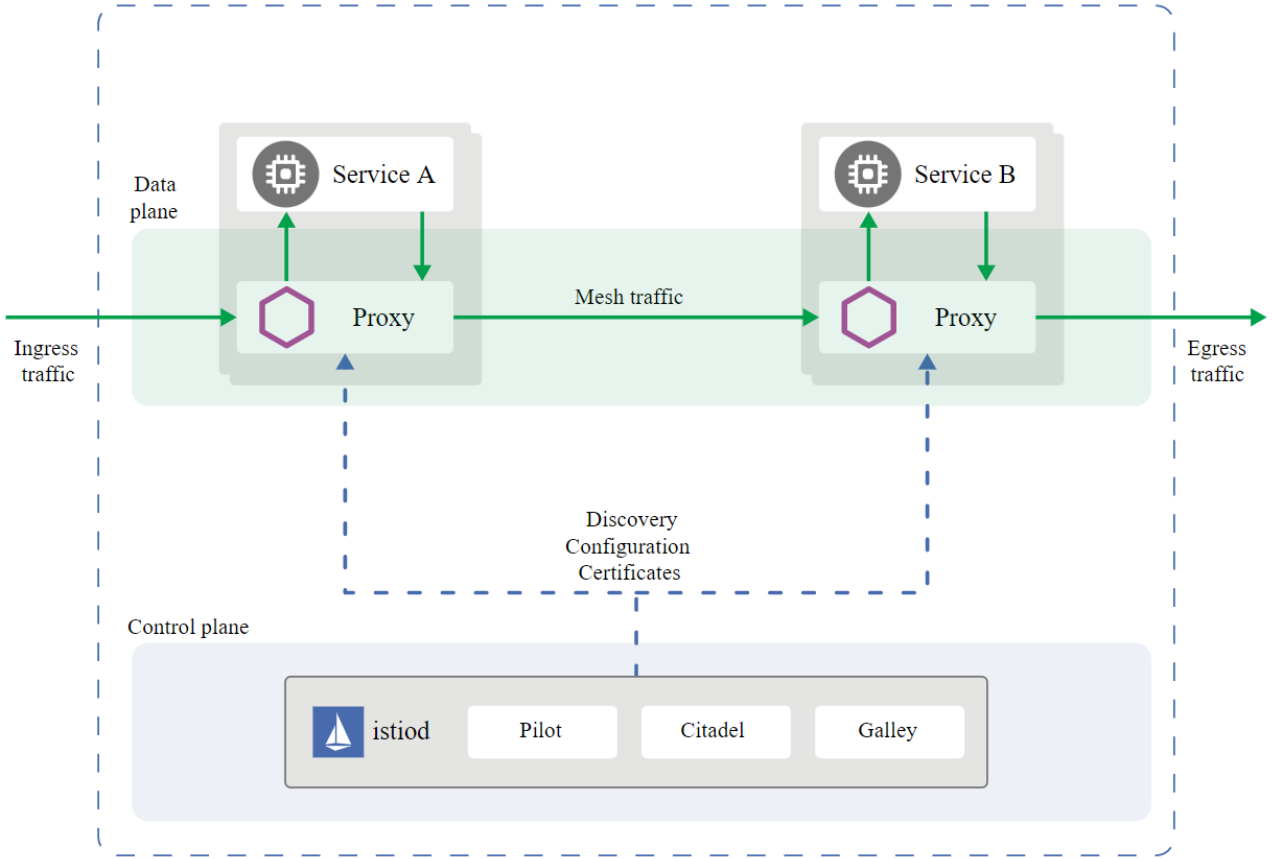
- L3/L4 filter and routing architecture
- HTTP L7 filter architecture
- First class HTTP/2 support
- gRPC support
- MongoDB and DynamoDB L7 support



Istio

A service mesh control plane which uses Envoy as data plane

- Originally created by Google and IBM
- Written in Go
- Provides core features for:
  - Traffic management
  - Observability
  - Security



Has four main components

- Pilot for managing and configuring the Envoy proxies
- Citadel for service-to-service and end-user authentication
- Galley for configuration validation, ingestion, processing and distribution

The Buzz Around Istio

The Buzz Around Istio

openusagecommons

Free and Fair to Use.

The Open Usage Commons gives open source project users peace of mind that projects are free and fair to use.

About the Open Usage Commons

The Open Usage Commons helps projects protect their project identity through programs such as trademark management and usage guidelines. We are guided by a dedication to open source, a passion for open use, and a commitment to being an organization created in service to open source projects.

Open Usage Commons FAQ

Trademark Guidelines

Projects

Istio
An open platform to connect, manage, and secure microservices

Angular
A web application framework for mobile, desktop, and web

Gerrit Code Review
A free, web-based team code collaboration tool.

The Open Usage Commons currently uses the existing trademark policies for the Open Usage Commons project trademarks. Existing

KUBERNETES / SECURITY

THENEWSTACK

How the U.S. Air Force Deployed Kubernetes and Istio on an F-16 in 45 days

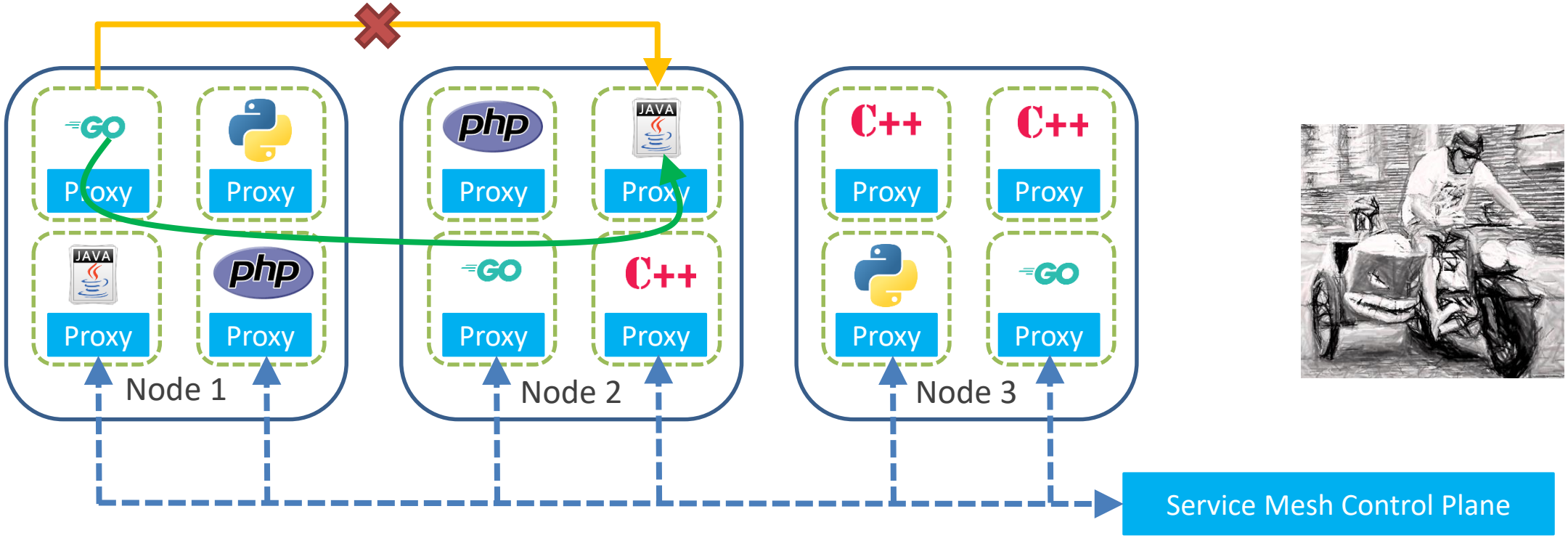
Kubernetes, Istio, knative and an internally developed specification for "hardening" containers are now the default software development platform across the military.

Dec 24th, 2019 8:19am by Tom Krazit

Feature image via Pixabay.

As hybrid cloud strategies go, the U.S. military certainly is taking a unique approach.

The Sidecar Problem



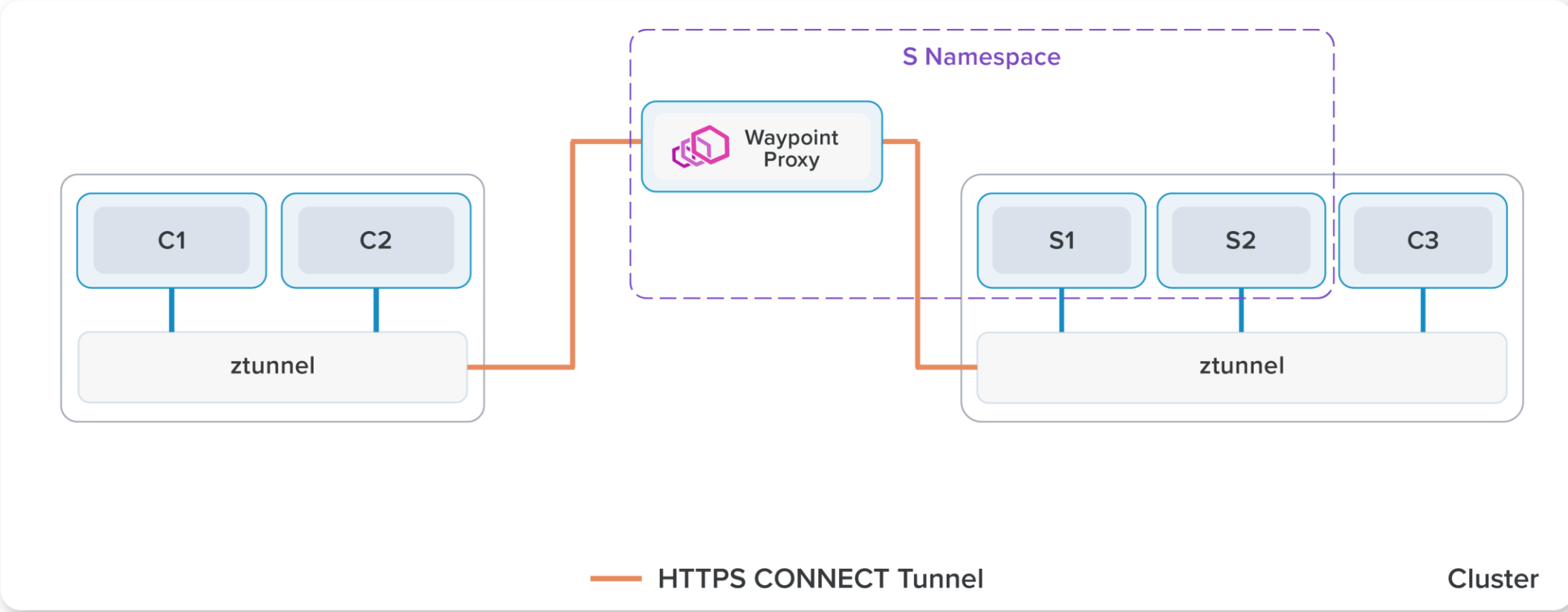
- Dynamic service discovery
- Load balancing
- Health checks
- Timeouts
- Retries
- Circuit breakers

- Traffic encryption (mTLS)
- Fine-grained access control
- Access auditing
- Rate limiting
- Rewrites and redirects

- Consistent metrics
- Access logs
- Distributed tracing
- Fault injection

- Lots of resources
- Added latency
- Operational overhead
- HTTP focused

The Buzz Around Istio: Ambient Mesh



The Two Big Players

Istio

- Somewhat easy to install
- Hard to operate
- Heavy control plane
- Advanced data plane (Envoy)
- Has a built in ingress (and egress) controller (Envoy)
- Kiali as dashboard
- Monitoring:
  - Service graph
  - S2S latency
  - Response codes
  - Jaeger Tracing
- Advanced HTTP routing:
  - Blue/Green
  - Canary
  - Dark launch, shadow

LINKERD

- Very easy to install
- Easy to operate
- Light control plane
- Very light data plane (in Rust)
- No ingress included
- Has a dashboard
- Monitoring:
  - Service graph
  - S2S latency
  - Response codes
  - No tracing
- Very basic HTTP routing:
  - Blue/Green
  - Canary
  - No dark launch

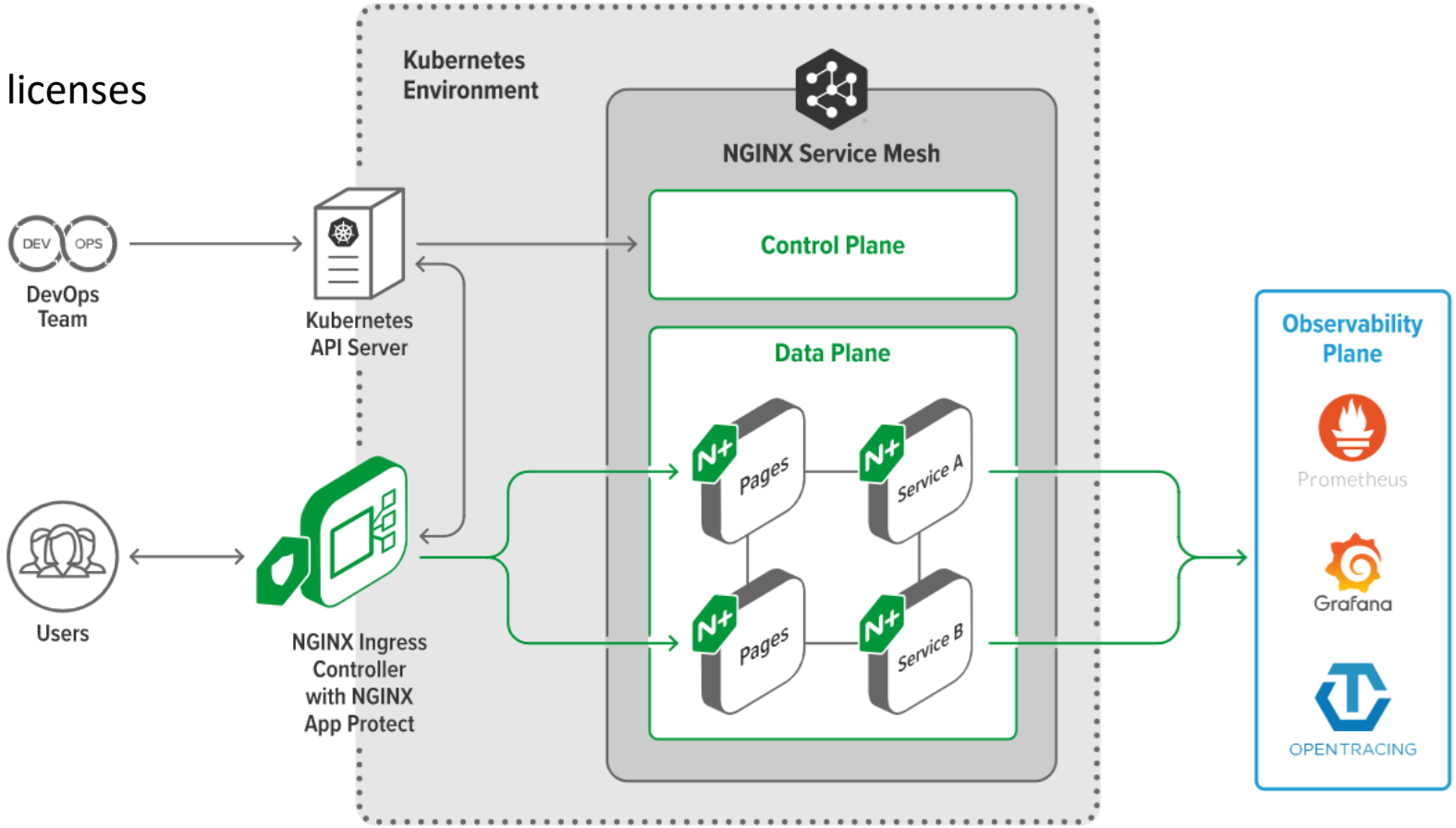
Other Players 1: Nginx Service Mesh

Lightweight and turnkey solution using Nginx Plus

- Created by the **F5/Nginx team**
- **Free**, but can require Nginx Plus Ingress licenses
- Enterprise **support** available

Features include

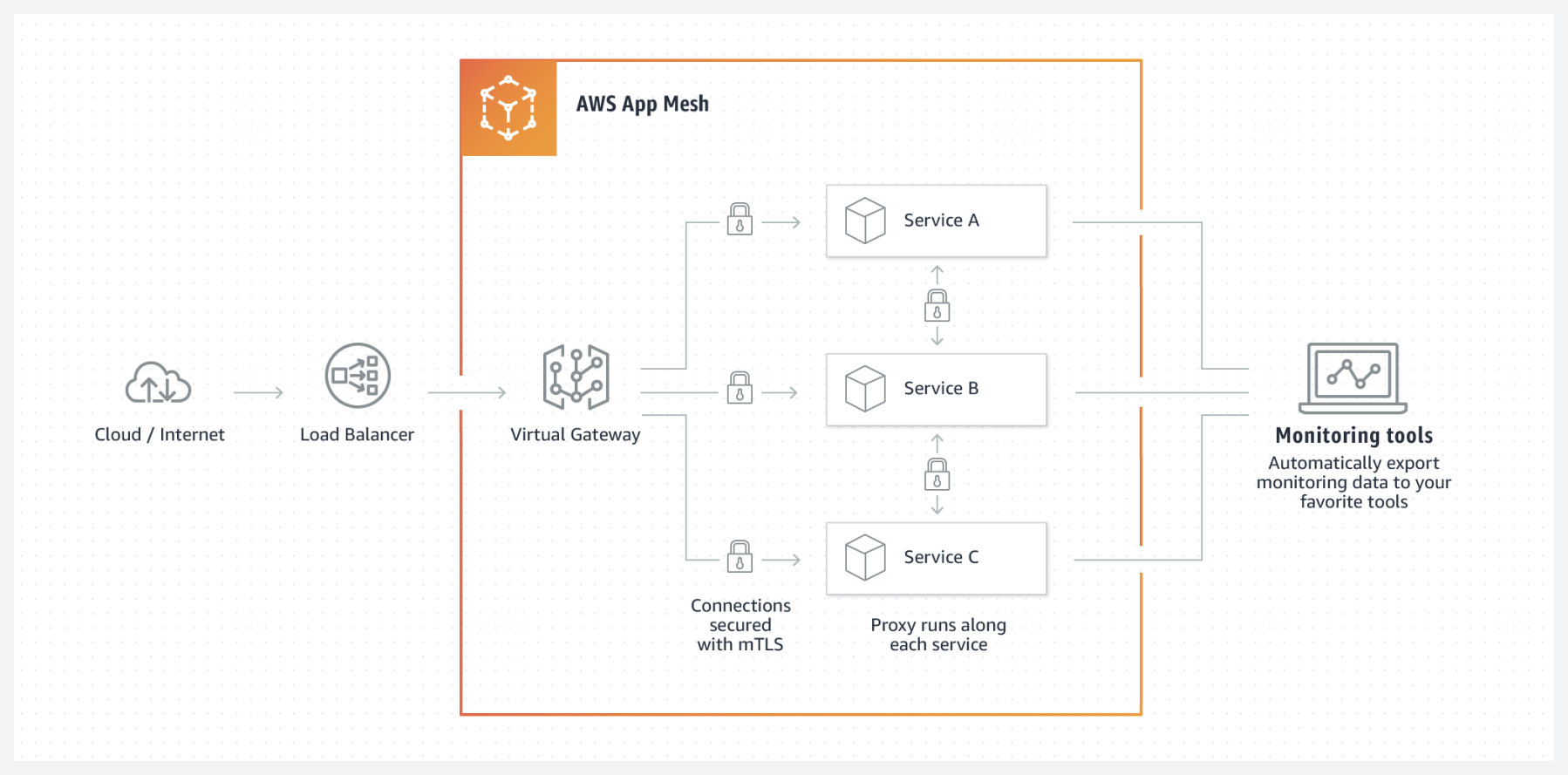
- Rate shaping, quality of service (QoS)
- Blue-green, A/B, Canary deployments
- Circuit breaking
- API gateway features
- Service identity, Zero trust
- mTLS enforcement
- Certificate lifecycle management
- Allowlist support for ingress and egress
- Per-service access control for east-west traffic



Other Players 2: AWS App Mesh

AWS specific service mesh that makes it easy to monitor and control services

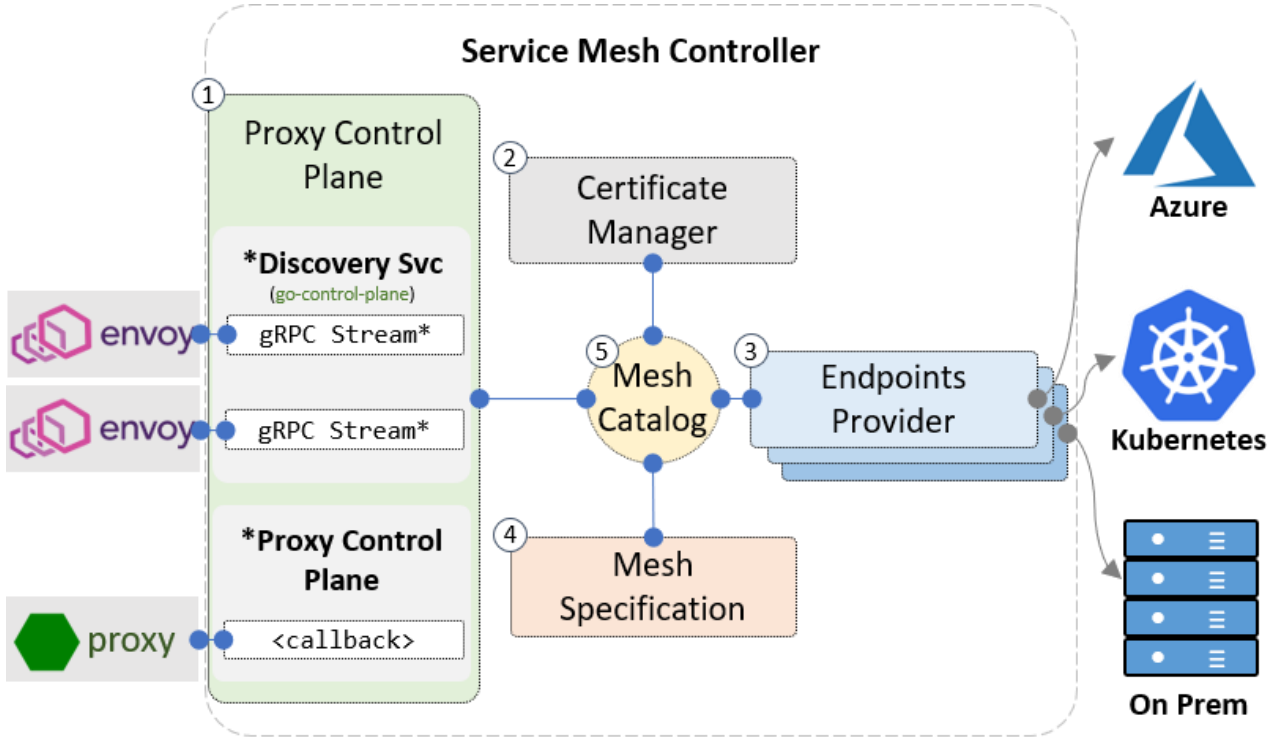
- Works with Amazon EC2, Amazon ECS, Amazon EKS, and AWS Fargate
- **Envoy** as dataplane



Other Players 3: Open Service Mesh

OSM is a lightweight and extensible cloud native service mesh

- Originally created by **Microsoft** (supported by AKS)
- CNCF sandbox project
- **Envoy** as dataplane, control plane written in **Go**
- Configure via **Service Mesh Interface** (SMI)



Features include

- Easily and transparently configure traffic shifting
- Secure end-to-end communication by enabling mTLS
- Fine grained access control policies for services
- Observability and insights into application metrics for debugging and monitoring services
- Integrate with external certificate management services/solutions with a pluggable interface
- Onboard applications onto the mesh by enabling automatic sidecar injection of Envoy proxy
- Flexible enough to handle both simple and complex scenarios through SMI and Envoy XDS APIs

Other Players 4: Service Mesh Interface

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

	Latest Release
Core Specification:	
SMI Specification	v0.6.0
Specification Components	
Traffic Access Control	v1alpha3
Traffic Metrics	v1alpha1
Traffic Specs	v1alpha4
Traffic Split	v1alpha4

Ecosystem

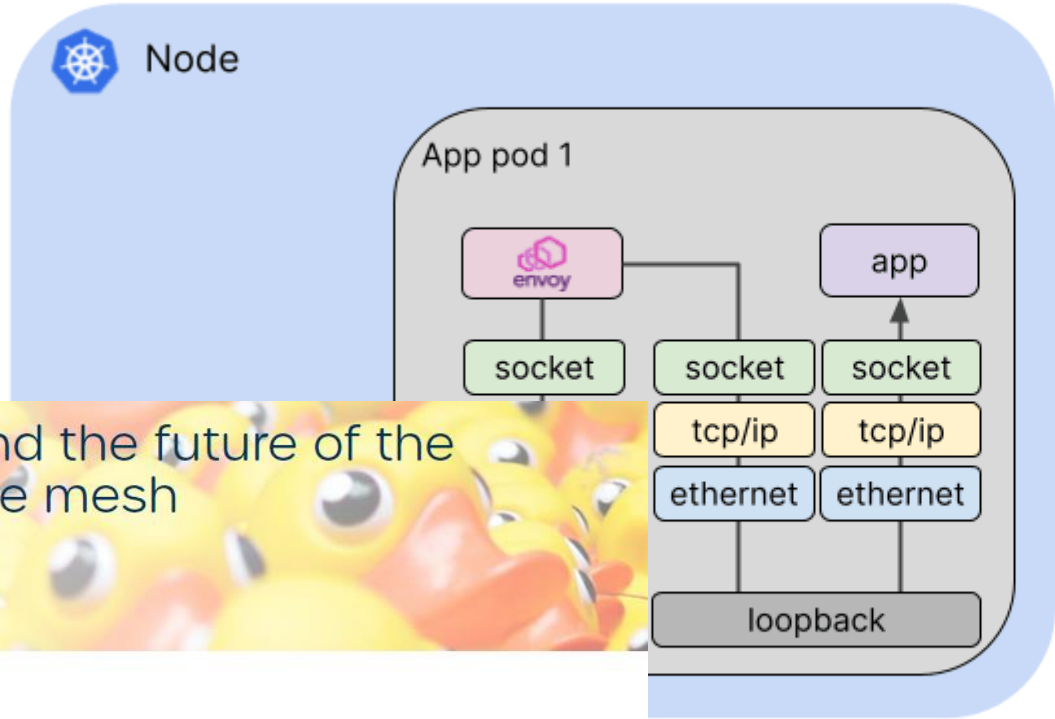
- **Linkerd**: ultralight service mesh
- **Nginx Service Mesh:** turnkey solution using Nginx Plus
- **Open Service Mesh**: lightweight and extensible cloud native service mesh
- **Traefik Mesh**: simpler service mesh
- **Gloo Mesh**: Multi-cluster service mesh management plane
- **Meshery**: the service mesh management plane
- **Flagger**: progressive delivery operator
- **Argo Rollouts**: advanced deployment & progressive delivery controller
- **Istio***: connect, secure, control, observe
- **Consul Connect***: service segmentation

* via adaptor

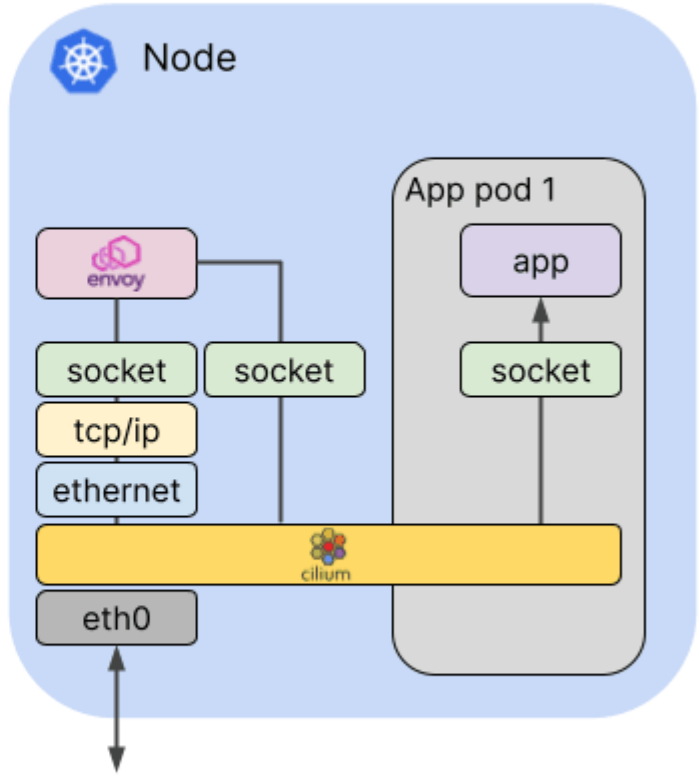
Other Players 5: Even More Tools ☺

Other Players 6: Cilium Service Mesh with eBPF

Service mesh with traditional networking



Sidecarless model, eBPF acceleration



William Morgan
June 7, 2022

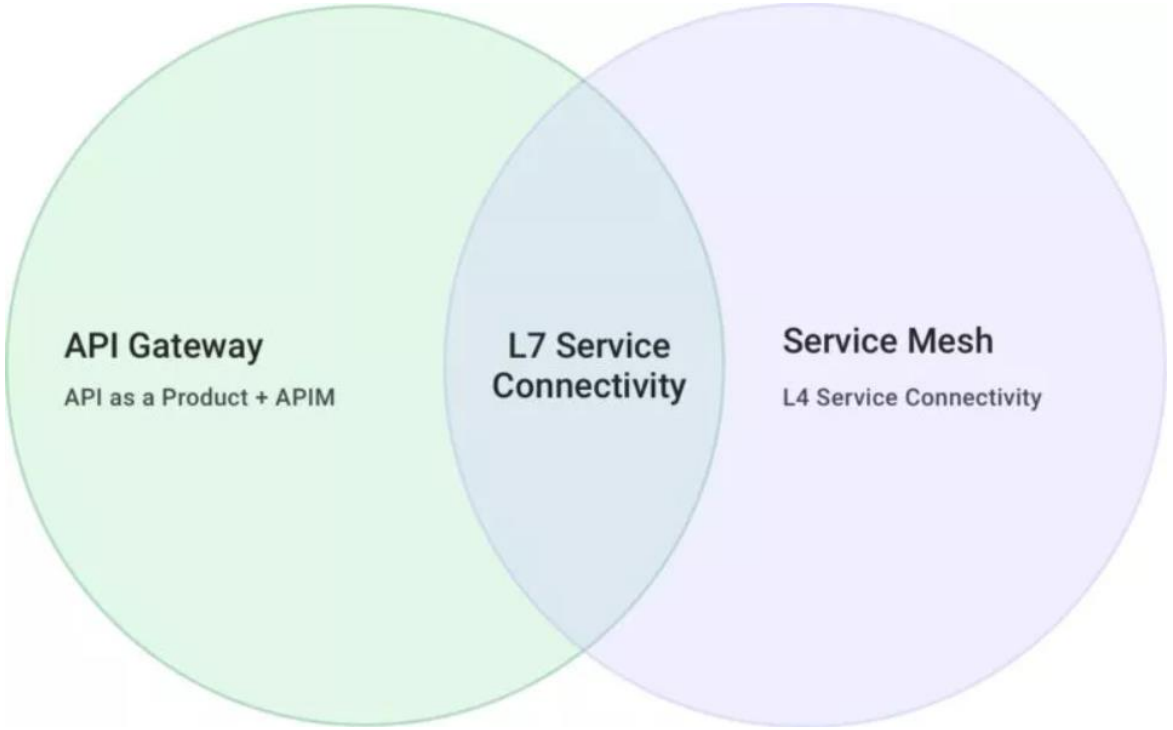
Blog > 101

eBPF is cool technology with a lot to offer the cloud native world. It's been a popular choice for the CNI layer of Kubernetes clusters thanks to projects like Cilium. Service meshes like Linkerd are often deployed with CNI layers like Cilium, combining Linkerd's powerful L7 processing with Cilium's super-fast L3/4 handling.

Other Players 7: Service Mesh vs. API Gateways

Many features of the Service Mesh and API Gateways seems overlapping

- Telemetry collection
- Distributed tracing
- Service discovery
- Load balancing
- TLS termination/origination
- JWT validation
- Request routing
- Traffic splitting
- Canary releasing
- Traffic shadowing
- Rate limiting



Control the edge Control the cluster

Good to read:
https://blog.christianposta.com/microservices/do-i-need-an-api-gateway-if-i-have-a-service-mesh/
https://konghq.com/blog/the-difference-between-api-gateways-and-service-mesh

Other Players 8: Kubernetes Gateway API

The Kubernetes Ingress resource seems to be limited for API Gateway use-cases

- Very similar CRDs appeared for ingress controllers (including service mesh ingresses)
- Ingress annotations are overloaded, and very vendor specific

Gateway API (currently in *v1beta1* status):

- Standardize underlying route matching, traffic management, and service exposure
- Represent L4/L7 routing and traffic management through common core API resources
- Provide extensibility for more complex capabilities in a way that does not sacrifice the user experience of the core API

Other Players 9: Service Mesh Managers

Management plane that enables the adoption, operation, and management of service mesh

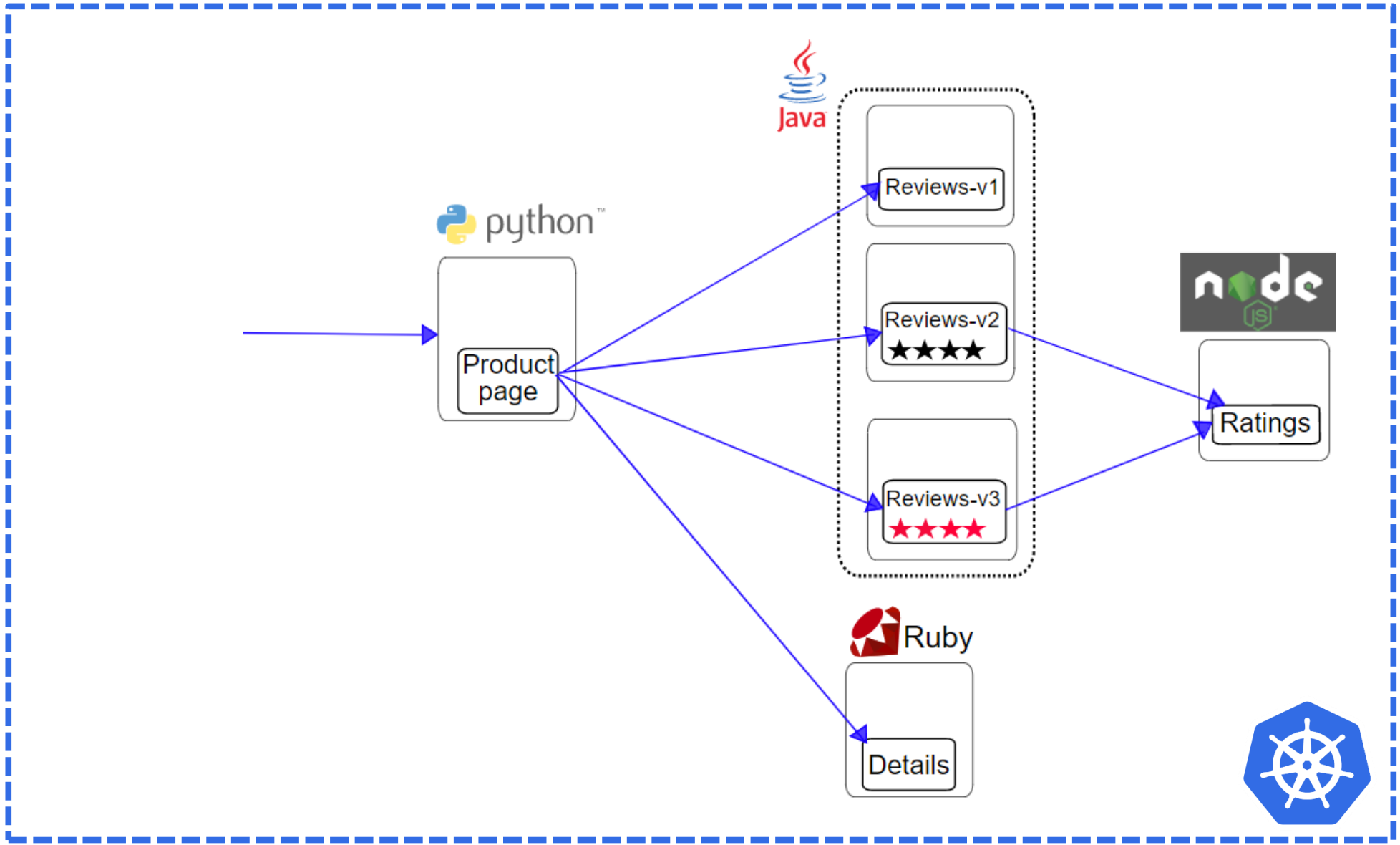
- Typical features:
  - Lifecycle management (e.g. version upgrades)
  - Multi-cluster management and hybrid (non-Kubernetes) deployments
  - Multi-tenancy and self-service
  - Enhanced security (e.g. access control, cert management)

- Created by **Solo.io**
- Supports only Istio

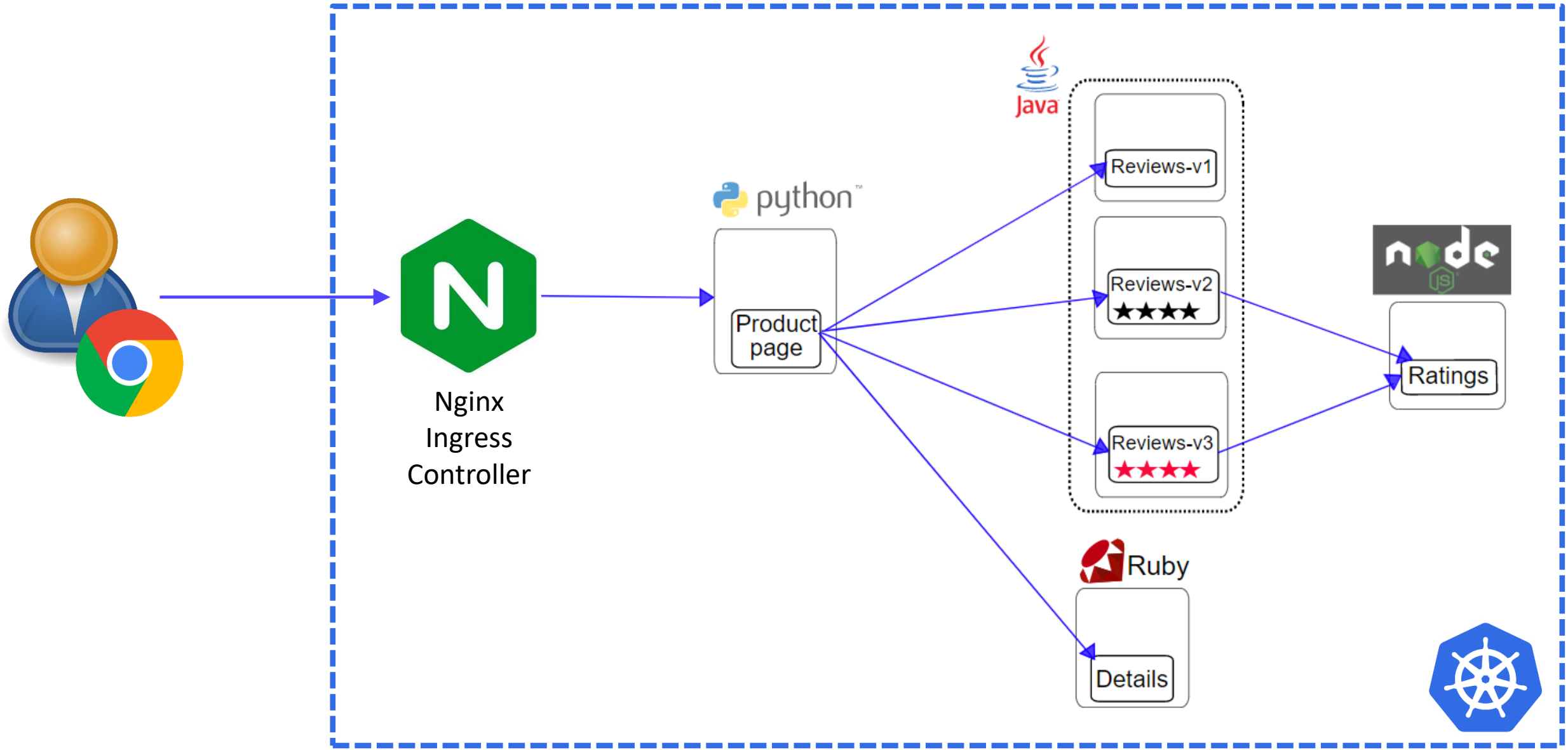
- Created by **Layer5.io**
- Many supported service mesh solutions


Demo time

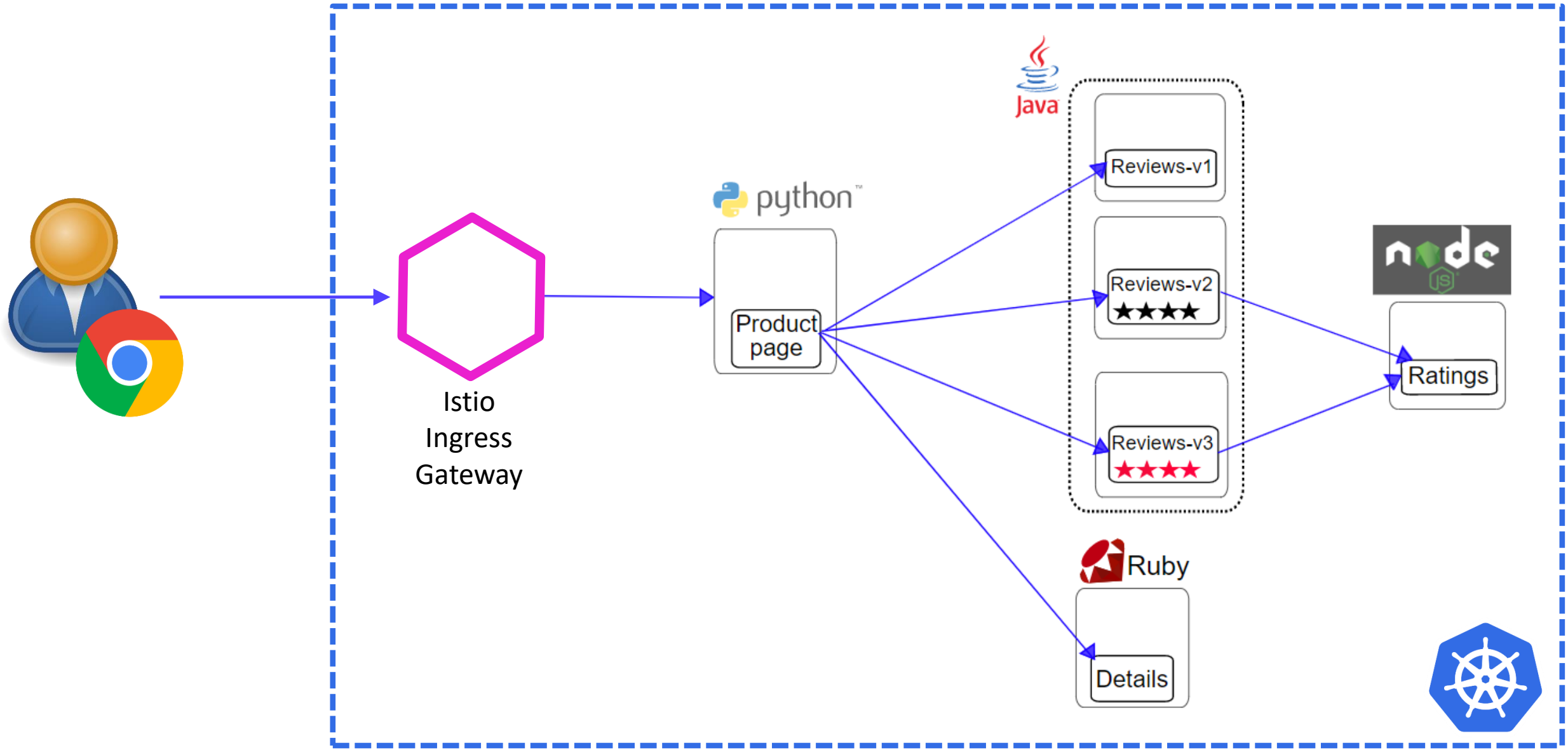
Bookinfo Application



Bookinfo Application



Bookinfo Application



Okay, But Should I Use a Service Mesh???

It depends…

Not a good reason:

- I've heard it's cool since Google made this!
- Want to be cloud native so I must use this new stuff
- My boss told me…

But definitely use it if:

- You're already mastering the basic Kubernetes abstractions, but need to move forward
- You have micro(ish)services environment, written in multiple languages, and you need consistent telemetry
- You need mTLS for every service-to-service communication
- You have a microservices environment where some service are failing, and you can't figure out which one of them
- You need to apply more advance deployment patterns (e.g. canary, blue/green), since your current one is too slow