SERVICE MESH

MAGNUS LARSSON

CADEC 2019.01.24 & 2019.01.30 | CALLISTAENTERPRISE.SE

CALLISTA

— ENTERPRISE —

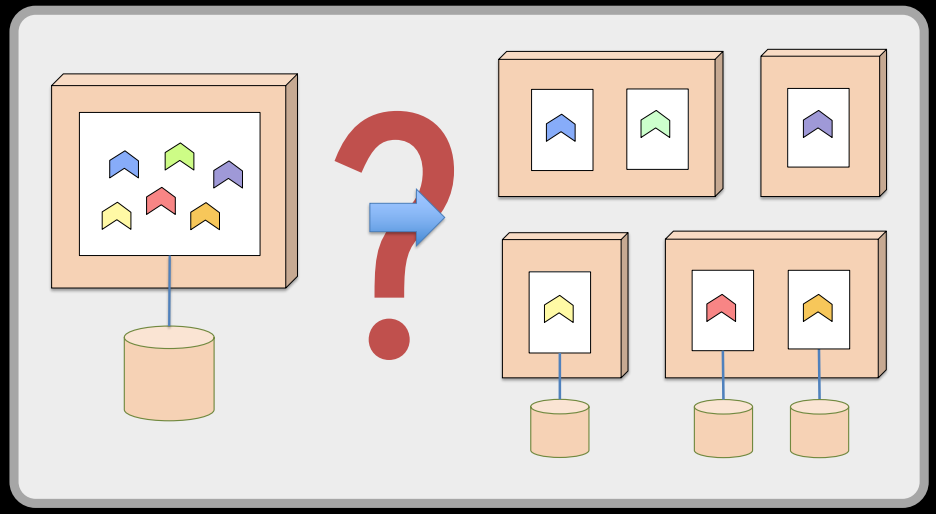
AGENDA

- Problem definition
- Previous solutions
- Service Mesh
  - Architecture
  - Capabilities
  - Products
- DEMO, DEMO, DEMO
- Summary

PROBLEM DEFINITION

DISCOVERY SERVER
WHERE ARE THE SERVICES?
WHICH SERVICE TO CALL?

EDGE SERVER
HOW TO HIDE PRIVATE SERVICES?
HOW TO PROTECT PUBLIC SERVICES?



RESILIENCE
HOW TO HANDLE FAULTS?
- SLOW OR NO RESPONSE
- TEMPORARY FAULTS
- OVERLOAD

DISTRIBUTED TRACING
WHO IS CALLING WHO?

CENTRALIZED CONFIGURATION
WHERE IS MY CONFIGURATION?
ARE ALL SERVICES
CONFIGURATION UP TO DATE?

TRAFFIC MANAGMENT
HOW TO CONTROL ROUTING?
- RATE LIMITING
- CANARY & BLUE/GREEN UPGRADES

SERVICE MANAGEMENT
HOW TO
- DEPLOY SERVICES?
- SCALE SERVICES?
- UPGRADE SERVICES?
- RESTART FAILING SERVICES?

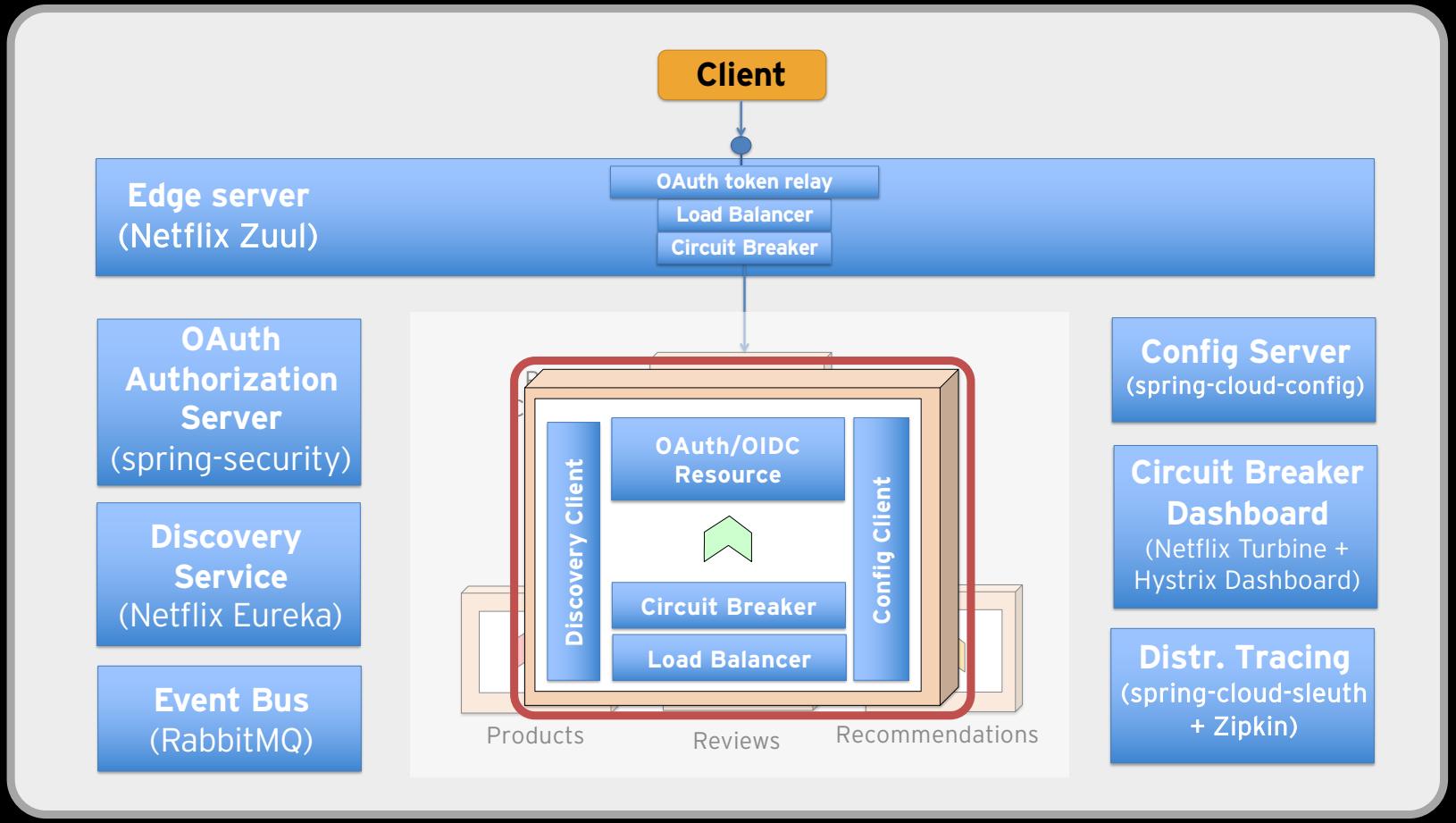
OBSERVABILITY
HOW ARE MY SERVICES PERFORMING?

LOG ANALYSIS
WHERE ARE THE LOGS?
HOW TO CORRELATE LOGS
FROM DIFFERENT SERVICES?

MONITORING
WHAT HARDWARE RESOURCES ARE USED?

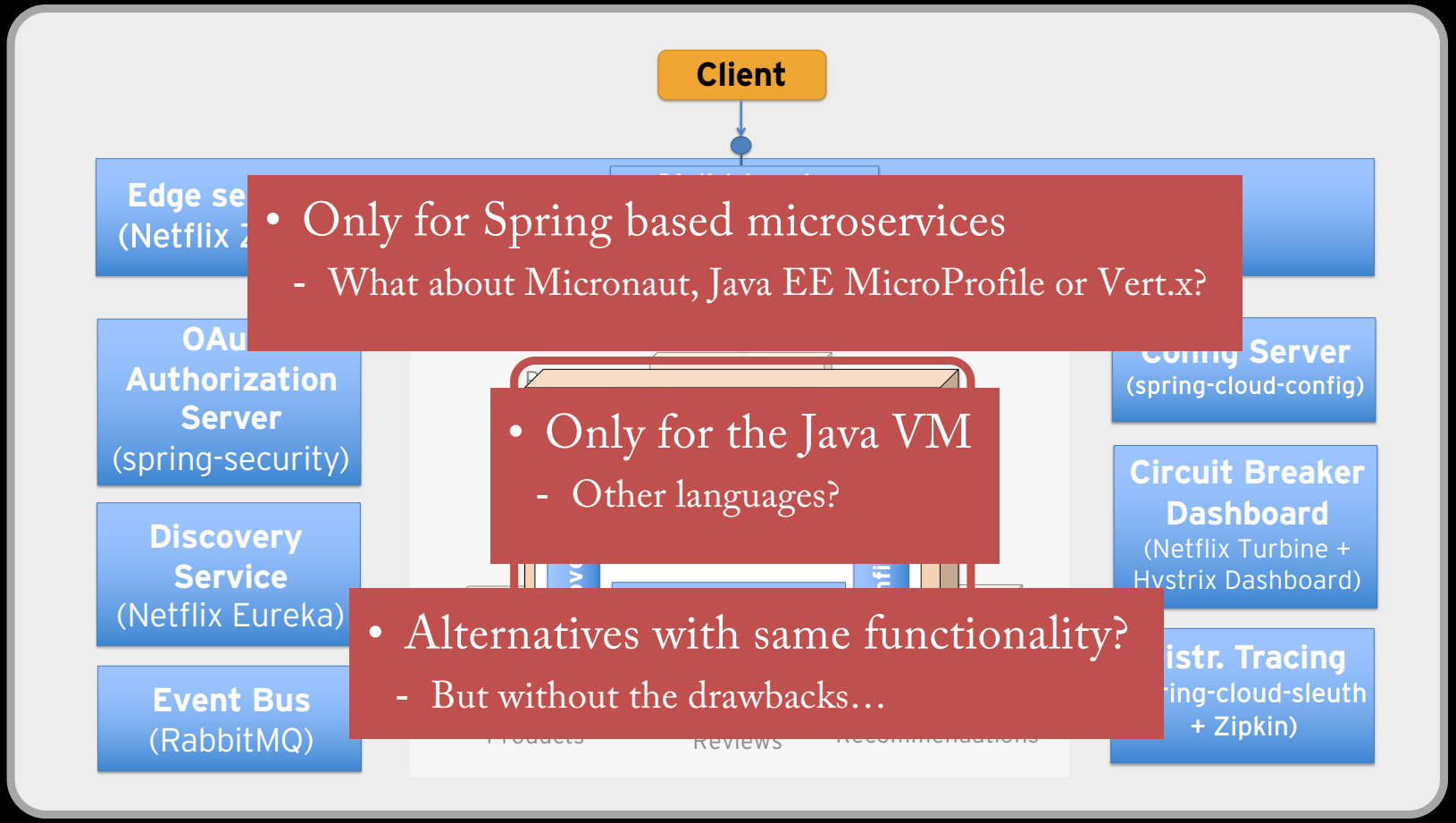
PREVIOUS SOLUTIONS: SPRING CLOUD/NETFLIX OSS

1. Discovery server
2. Edge server
3. Centralized configuration
4. Distributed tracing
5. Resilience



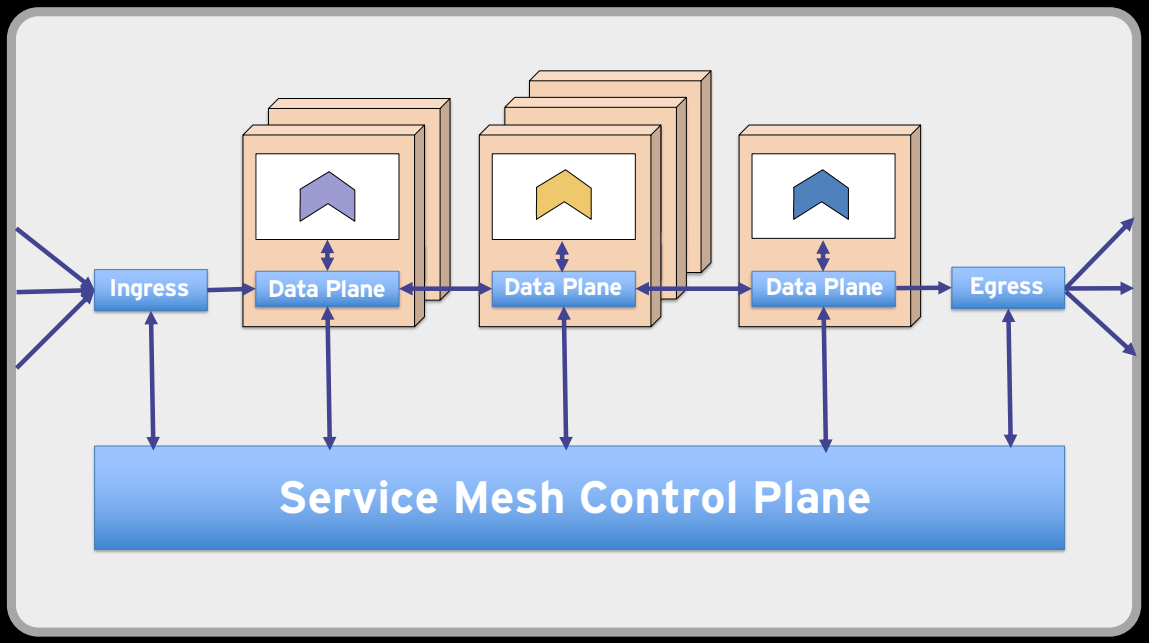
PREVIOUS SOLUTIONS: SPRING CLOUD/NETFLIX OSS

1. Discovery server
2. Edge server
3. Centralized configuration
4. Distributed tracing
5. Resilience



SERVICE MESH - ARCHITECTURE

- Data Plane injected **at Run Time**
  - **Data Plane** acts as a Proxy
  - **Runs as a "sidecar"**
- Ingress and Egress acts a external proxies
- Operators declares a desired state to the Control Plan
- **Control Plane** send commands to the Data Plan
- Data Plan reports metrics to the Control Plane
- No affect on development
  - Trace Ids still need to be managed
- Polyglot



SERVICE MESH - CAPABILITIES

- Traffic Management
- Resilience
- Edge Server
- Observability
- Distributed Tracing
- Monitoring

SERVICE MESH - PRODUCTS

- Linkerd
  - Developed by Buoyant
  - Open Source
  - Written in Scala
  - Launched in February 2016
    - » Based on Twitter Finagle, from 2011
  - Reached one hundred billion production requests in March 2017
  - Also see: How ForeSee processes billions of events with Linkerd per day, Aug 2017
  - Concerns
    - » Heavyweight sidecar…
    - » Upfront complex configuration
- Linkerd 2
  - Launched in September 2018
  - Written in Rust
  - Targeting Kubernetes, highly opinionated
    - » Zero Configration

SERVICE MESH - PRODUCTS

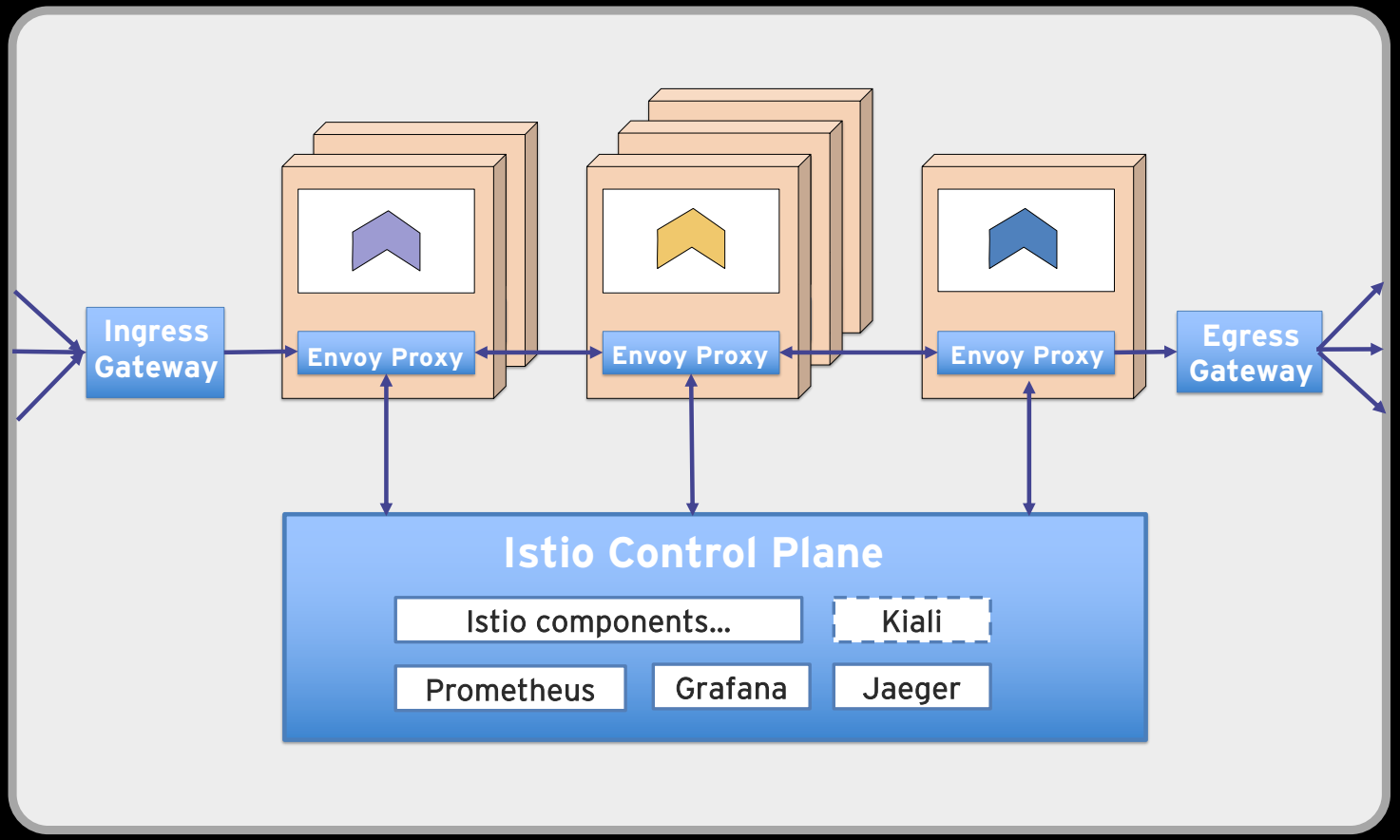
- Istio
  - Developed by Google, IBM and Lyft
  - Open Source
  - Written in Go
  - Data plane based on Lyft's Envoy proxy
    - » Written in C++
  - Launched in May 2017
  - Production ready since July 2018

  - The most functionally rich Service Mesh product as of today

  - Will be used in the **DEMO**!

- AWS App Mesh
  - Proprietary
  - Launched at **re:Invent** in November 2018
  - Based on Envoy proxy
  - Public Preview today

ISTIO - HIGH LEVEL ARCHITECTURE



WHEN IS A SERVICE MESH APPLICABLE?

- Synchronous vs asynchronous communication
  - Istio operates on TCP level, so actually doesn't care…
- Macro-, mini- or micro-services?
  - Or a mix…
  - A service mesh is agnostic to size, but was born in the land of microservices
- In cloud or on premises?
  - A service mesh does not care
- With or without containers?
  - Works without containers, but complex setup and configuration
  - Most used with a container orchestrator, e.g. Kubernetes

CAPABILITY MAPPING

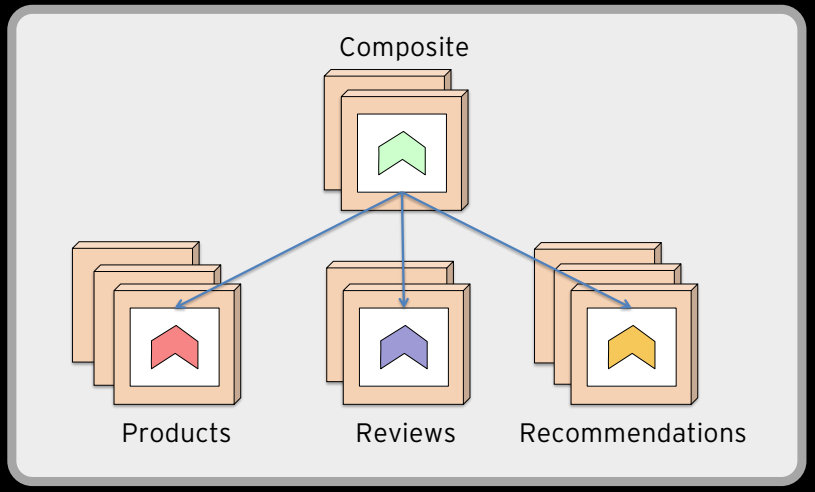
SPRING CLOUD/NETFLIX
KUBERNETES
ISTIO
EFK

DISCOVERY SERVER
WHERE ARE THE SERVICES?
WHICH SERVICE TO CALL?

RESILIENCE
HOW TO HANDLE FAULTS?
- SLOW OR NO RESPONSE
- TEMPORARY FAULTS
- OVERLOAD

EDGE SERVER
HOW TO HIDE PRIVATE SERVICES?
HOW TO PROTECT PUBLIC SERVICES?

DISTRIBUTED TRACING
WHO IS CALLING WHO?



CENTRALIZED CONFIGURATION
WHERE IS MY CONFIGURATION?
ARE ALL SERVICES
CONFIGURATION UP TO DATE?

TRAFFIC MANAGMENT
HOW TO CONTROL ROUTING?
- RATE LIMITING
- CANARY & BLUE/GREEN UPGRADES

SERVICE MANAGEMENT
HOW TO
- DEPLOY SERVICES?
- SCALE SERVICES?
- UPGRADE SERVICES?
- RESTART FAILING SERVICES?

OBSERVABILITY
HOW ARE MY SERVICES PERFORMING?

LOG ANALYSIS
WHERE ARE THE LOGS?
HOW TO CORRELATE LOGS
FROM DIFFERENT SERVICES?

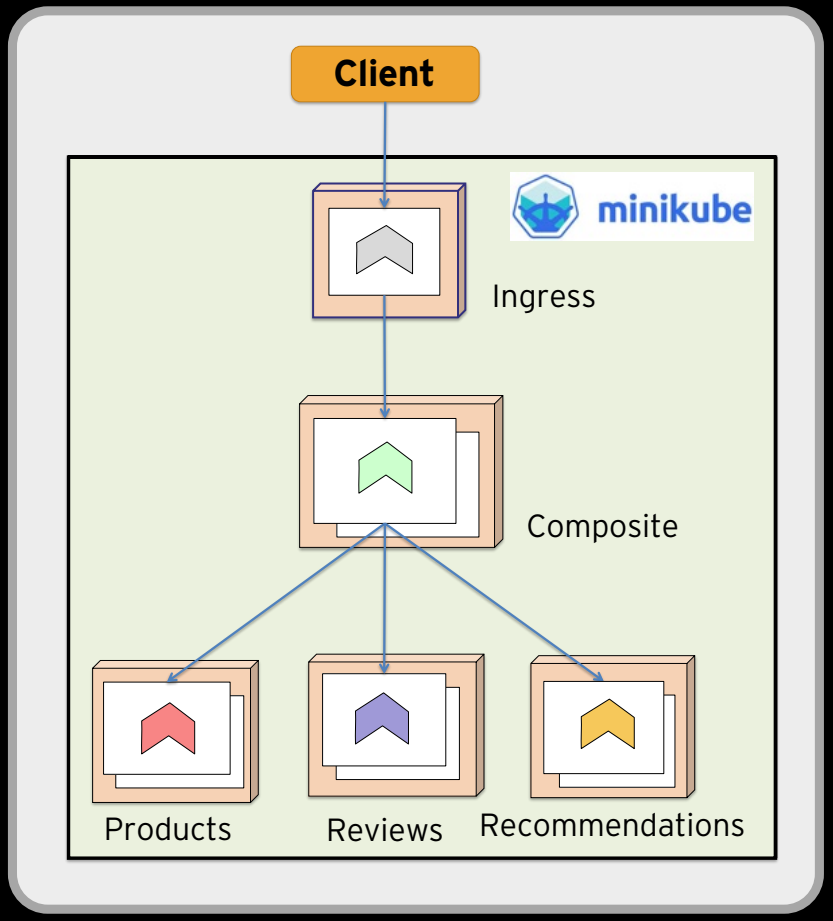
MONITORING
WHAT HARDWARE RESOURCES ARE USED?

DEMO, DEMO, DEMO

1. Observability
   - Kiali, Grafana and Jaeger
2. Resilience
   - Fault injection and retries
3. Rolling Upgrades
   - Canary
   - Blue/Green

DEMO LANDSCAPE

- Use Minikube
- Istio Control plane installed
- Istio Ingress Gateway configured
- V1 services deployed
  - Plain Spring Boot
  - No data storage
  - Istio Data plane injected



DEMO LANDSCAPE

- Prepared V2 services
  - Not Deployed!
  - No changes in API (nor in the databases)
- Log statements
  - Product – Go

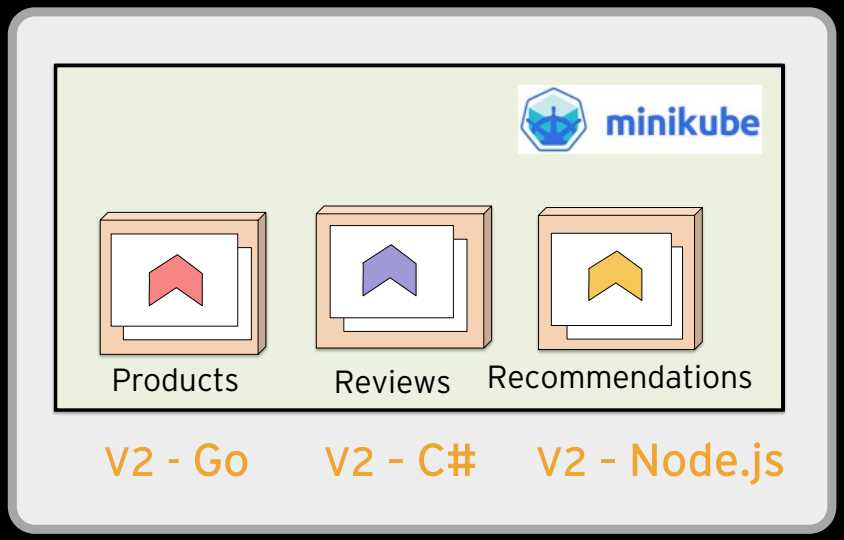
```
log.Printf("GET /product v2 (Go), productId: %v\n", id)
```

  - Recommendation – Node.js

```
console.log("GET /recommendation v2 (Node), productId: " + productId)
```

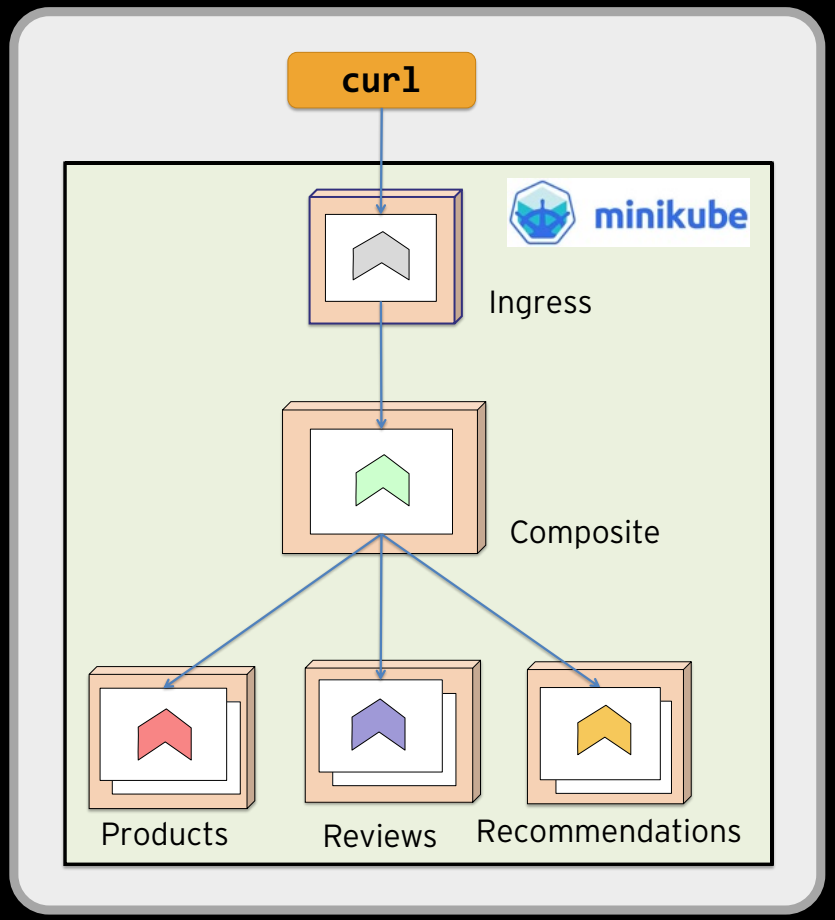
  - Review – .Net Core C#

```
Console.WriteLine( DateTime.Now.ToString("o") + " GET /review v2 (C#), productId: " + productId);
```



DEMO, DEMO, DEMO

1. Observability
   - Kiali, Grafana and Jaeger
2. Resilience
   - Fault injection and retries
3. Rolling Upgrades
   - Canary
   - Blue/Green



SUMMARY - SERVICE MESH

1. Next generation management tools for distributed systems, e.g. microservices
   - Traffic Management
   - Resilience
   - Edge Server
   - Observability
2. Works for
   - Synch and Asynch communication
   - Macro, Mini & Micro-services
   - In Cloud & On Premises
   - Polyglot, any language
3. Only affects runtime
4. Container environment (e.g. Kubernetes) preferred
   - Reduced complexity for installation and configuration