Scalable architecture at Sendo
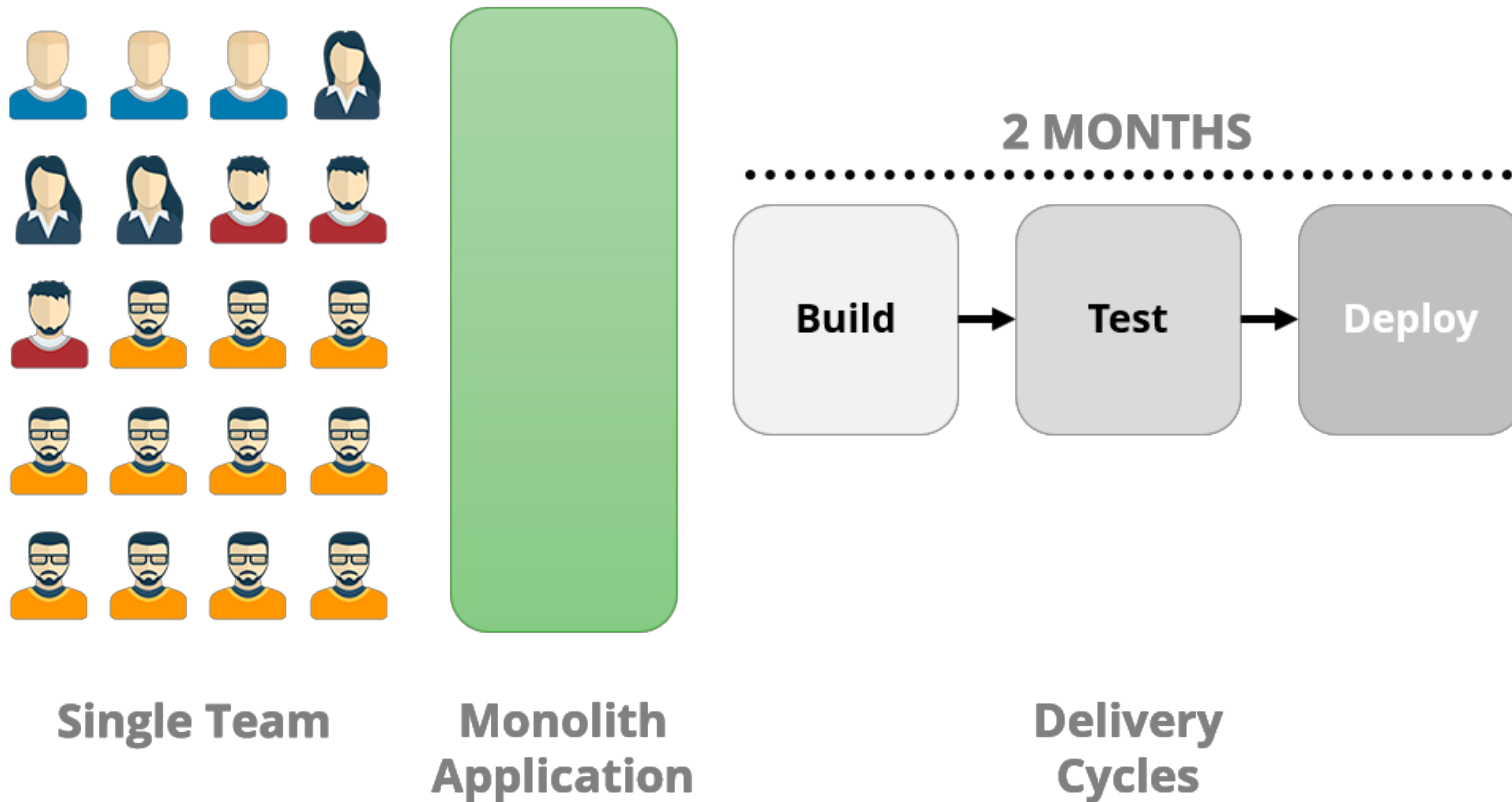
Viet Tran - Software Architect

Agenda

- Brief history of Sendo system

- Sendo microservices with gRPC and Protobuf

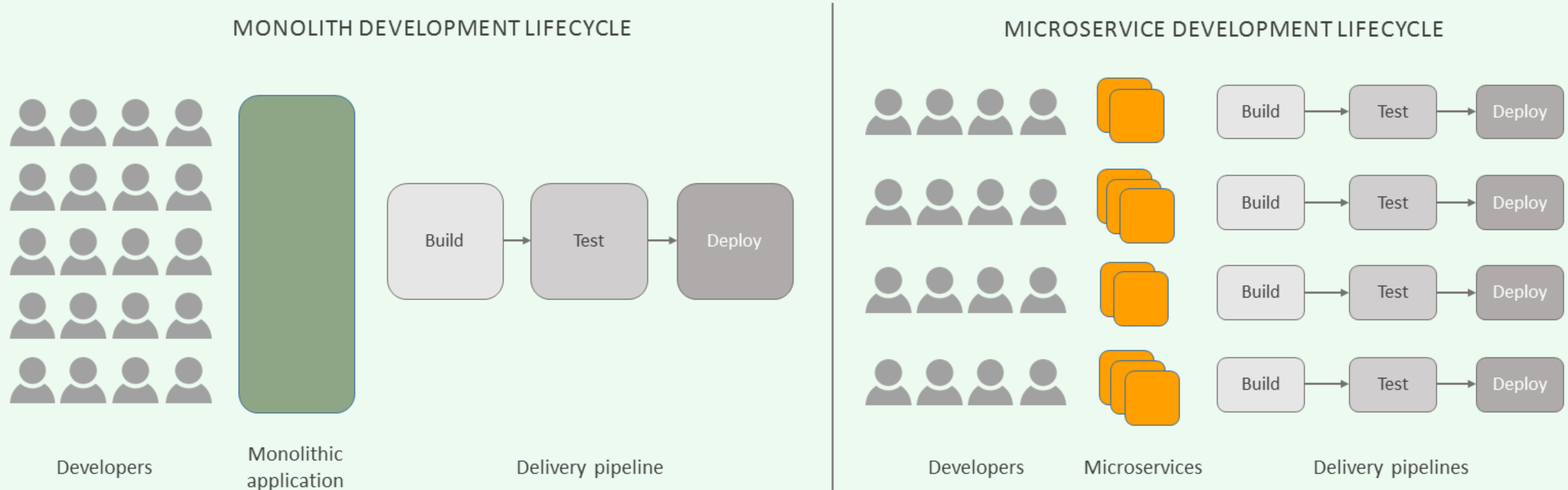- Sendo monitoring and logging

Brief history of Sendo system

In 2012: Sendo system was based on Magento

Brief history of Sendo system (cont)

2 MONTHS

Build → Test → Deploy

Single Team

Monolith Application

Delivery Cycles

Image Source: Beesion Technologies

Brief history of Sendo system (cont)

MONOLITH DEVELOPMENT LIFECYCLE

MICROSERVICE DEVELOPMENT LIFECYCLE

Build → Test → Deploy

Developers   Monolithic application   Delivery pipeline

Developers   Microservices   Delivery pipelines

In 2016: Sendo started to use microservices architecture

Brief history of Sendo system (cont)

We use multiple languages to build the services
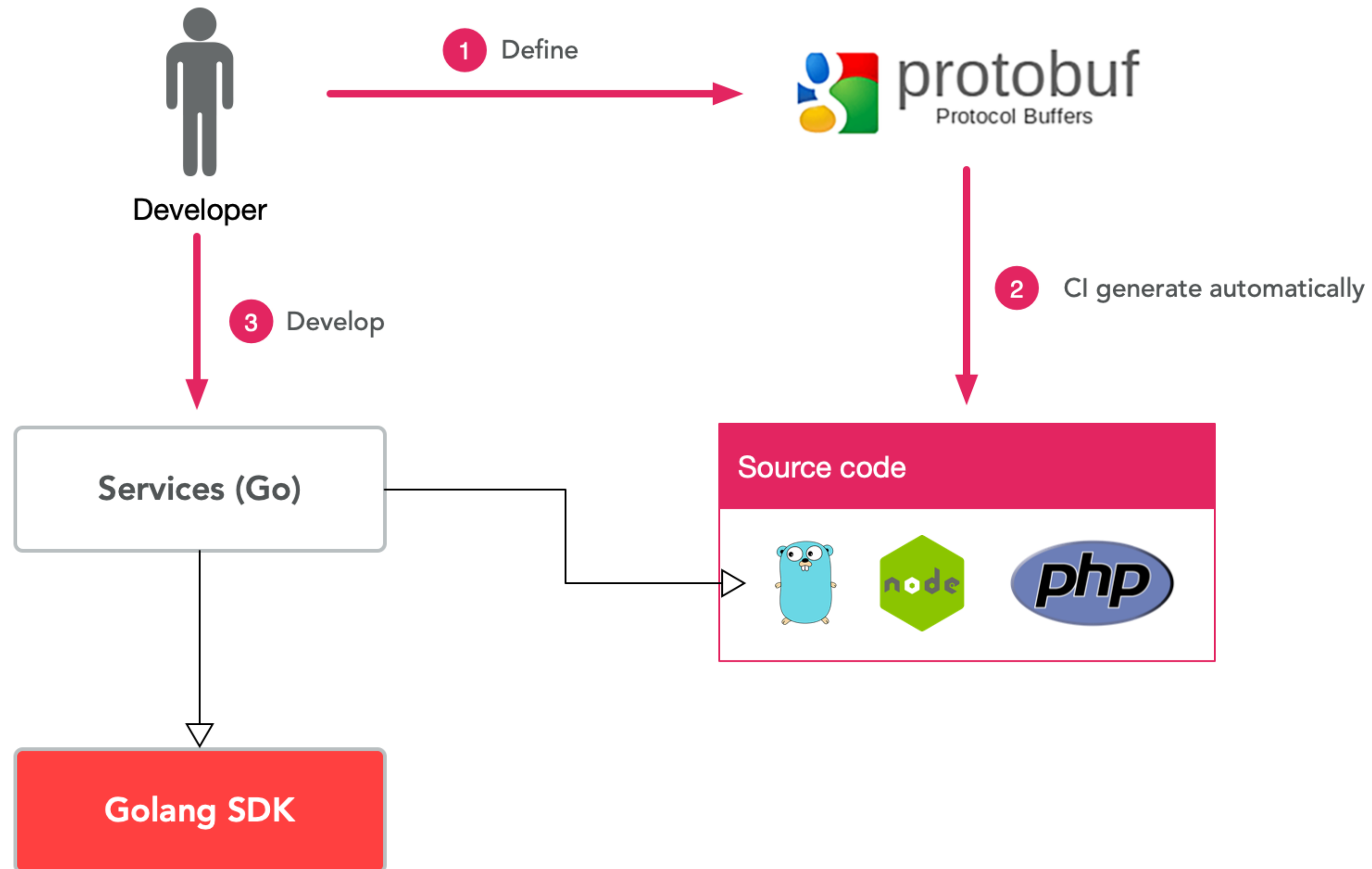
Image Source: Weaveworks

Problems

- High latency and low throughput

- Duplication data structures

- Hard to failure recovery on distributed system

We decided to migrate almost source codes to **GO**

Microservices with gRPC and Protobuf

Developer Workflow

Developer

1 Define

protobuf
Protocol Buffers

3 Develop

2 CI generate automatically

Services (Go)

Source code

Golang SDK
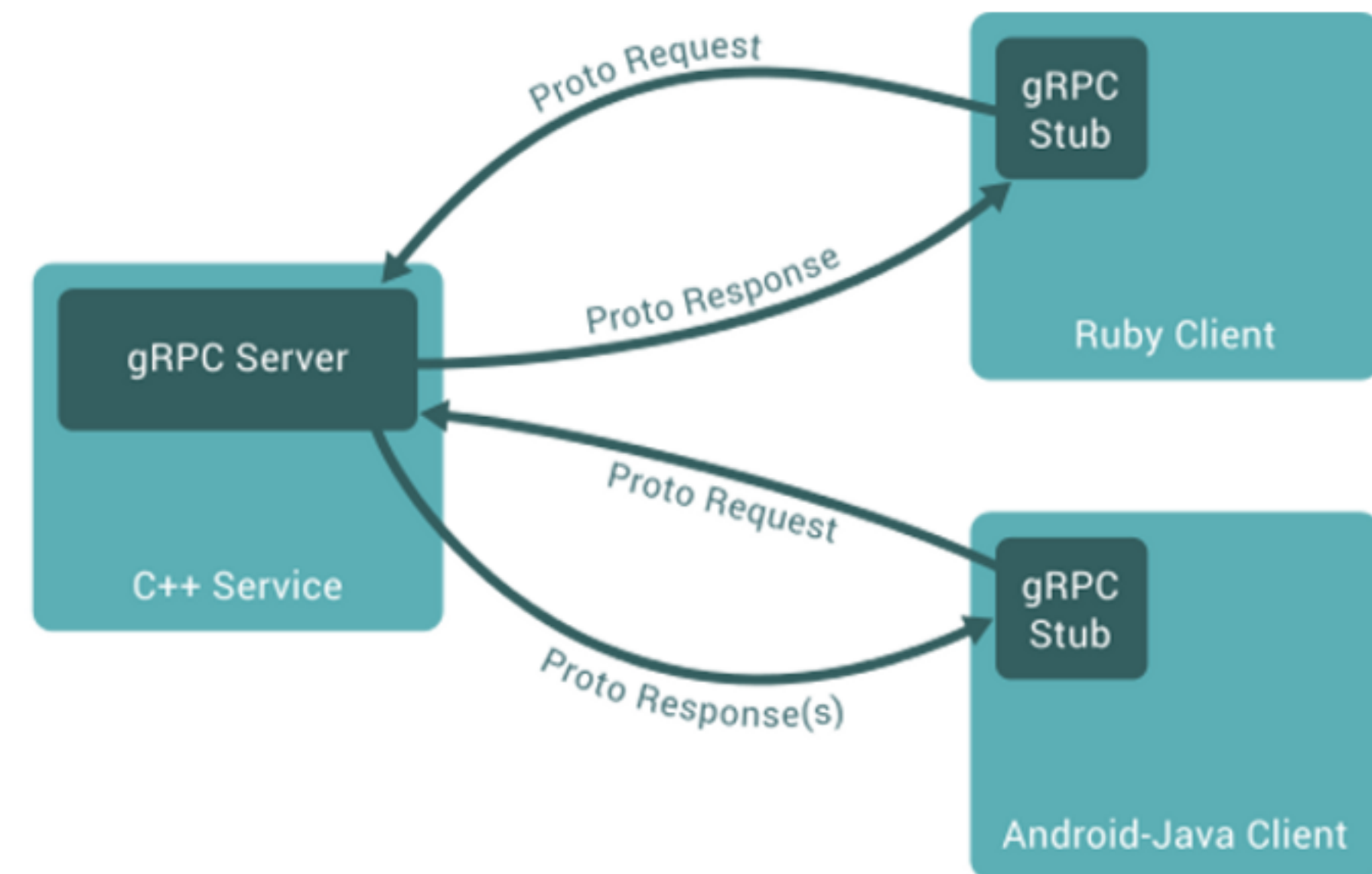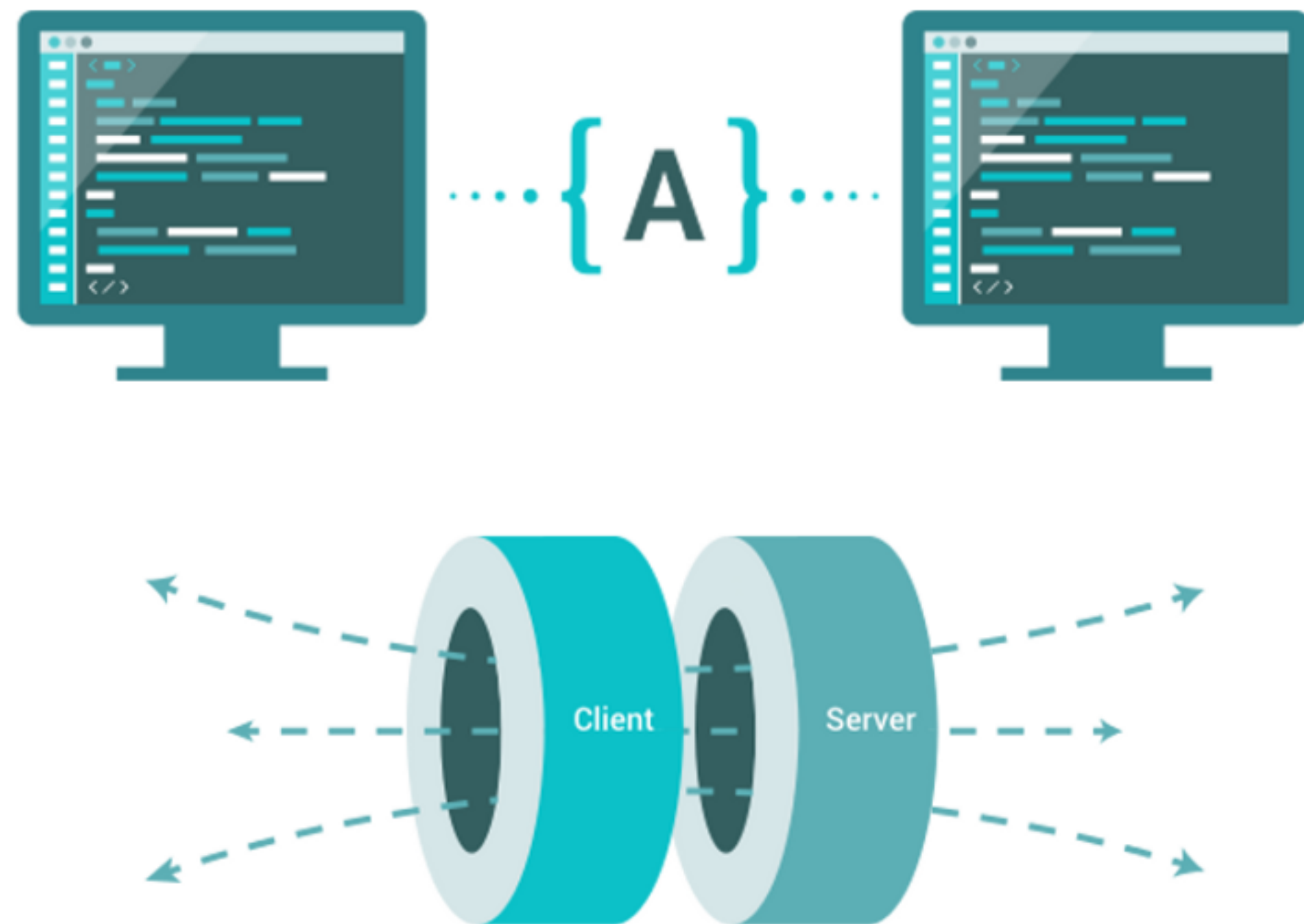
A demo Protobuf file

```protobuf
1   service NoteService {
2     // add new note
3     rpc Add(NoteAddReq) returns (Note) {
4       option (google.api.http) = {
5         post: "/demo/notes"
6         body: "*"
7       };
8     }
9     // list note
10    rpc List(NoteListReq) returns (Notes) {
11      option (google.api.http) = {
12        get: "/demo/notes"
13      };
14    }
15    // ...
16  }
```

# A demo Protobuf file (cont.)

```protobuf
42   message Note {
43     int64 id = 1;
44     string text = 2;
45     google.protobuf.Timestamp created = 3;
46     google.protobuf.Timestamp modified = 4;
47   }
48
49   message Notes {
50     int64 total = 1;
51     repeated Note notes = 2;
52   }
53
54   message NoteListReq {
55     base.Pagination pagination = 1;
56   }
```

Generated files

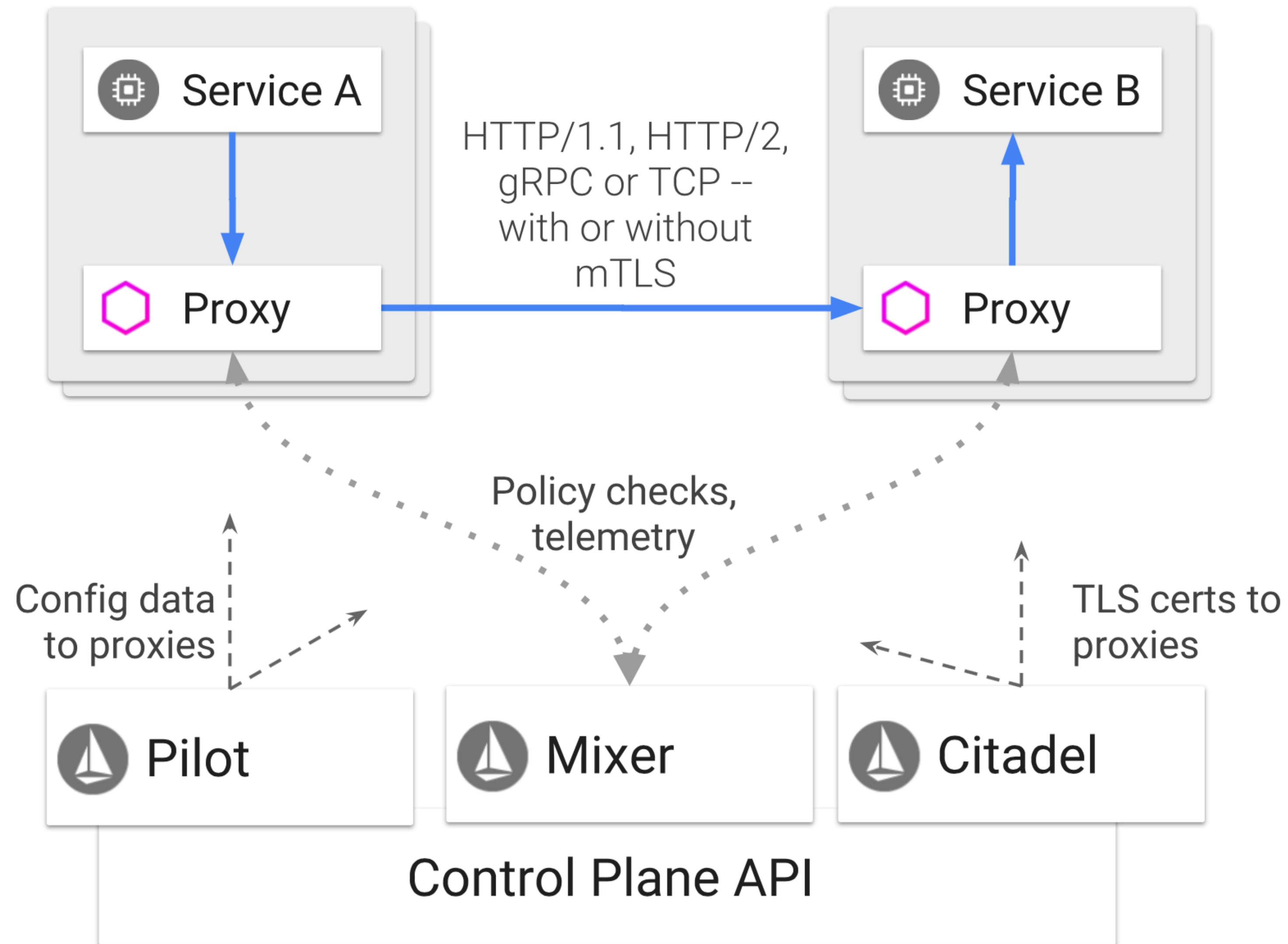| Name	Last commit
..	
📄 demo.pb.go	update build for v1.2.50
📄 demo.pb.gw.go	update build for v1.1.88
📄 demo.pb.validate.go	update build for v1.1.88
📄 demo.sendo.go	update build for v1.2.50

Sendo Microservices

- Service Mess with Envoy

- Service Discovery with Consul

- Load balancing with Nginx and Envoy

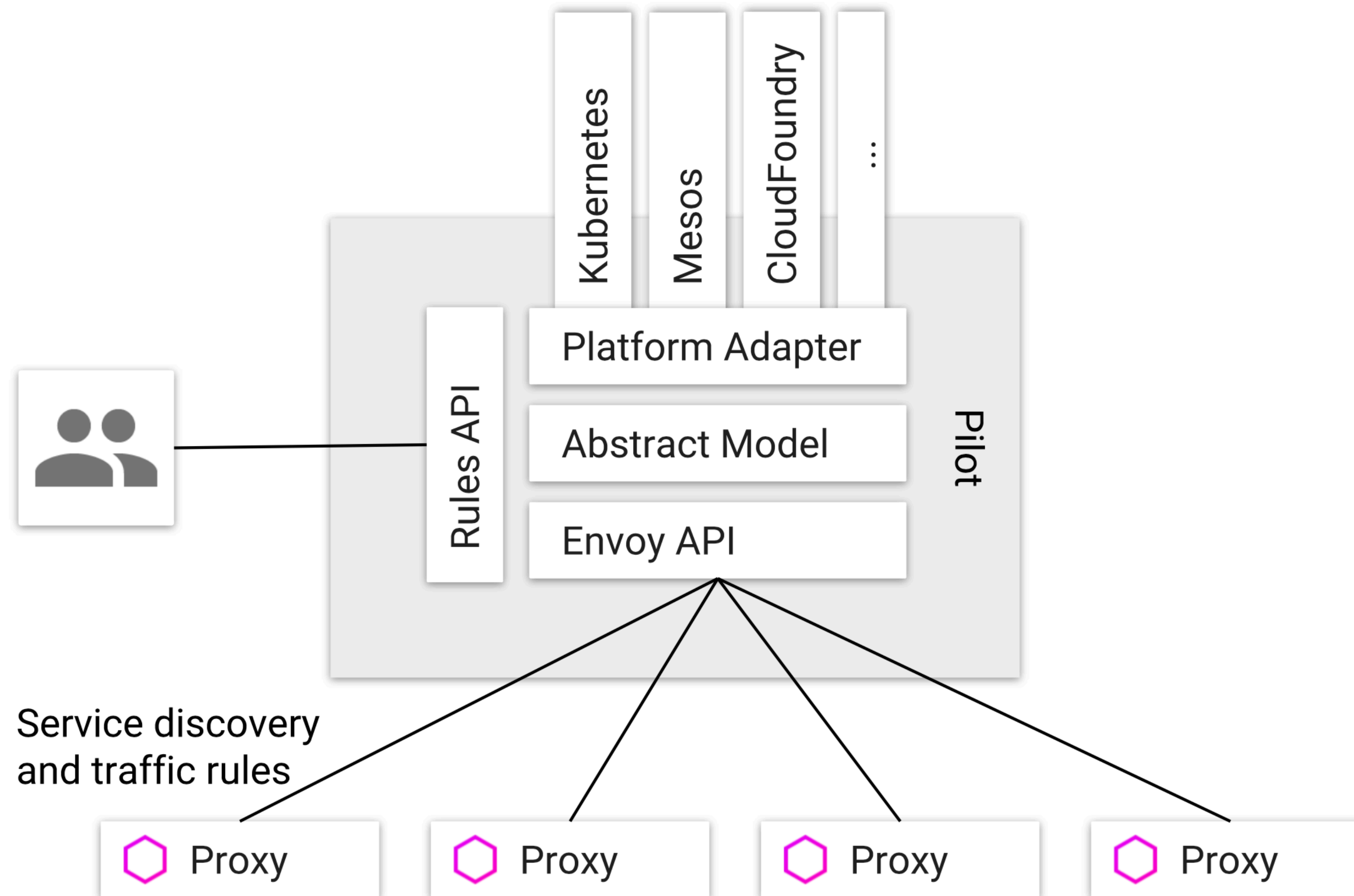- Very high throughput with Protobuf
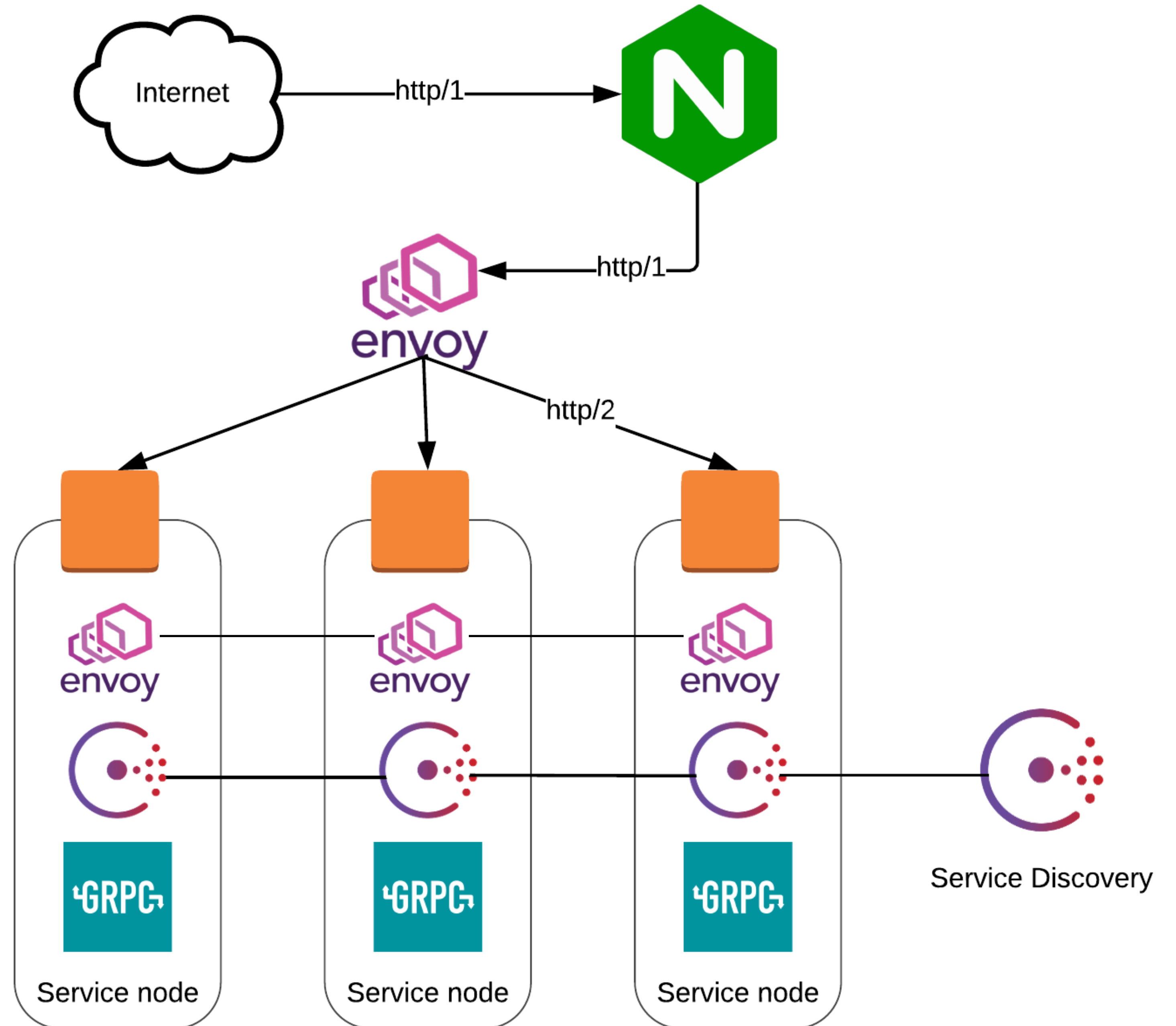
Problems solved so far

- High latency and low throughput
  Use Go and gRPC for Inter-service communication.

- Duplication data structures
  Use protobuf for generating data structures

- Hard to failure recover on distributed system
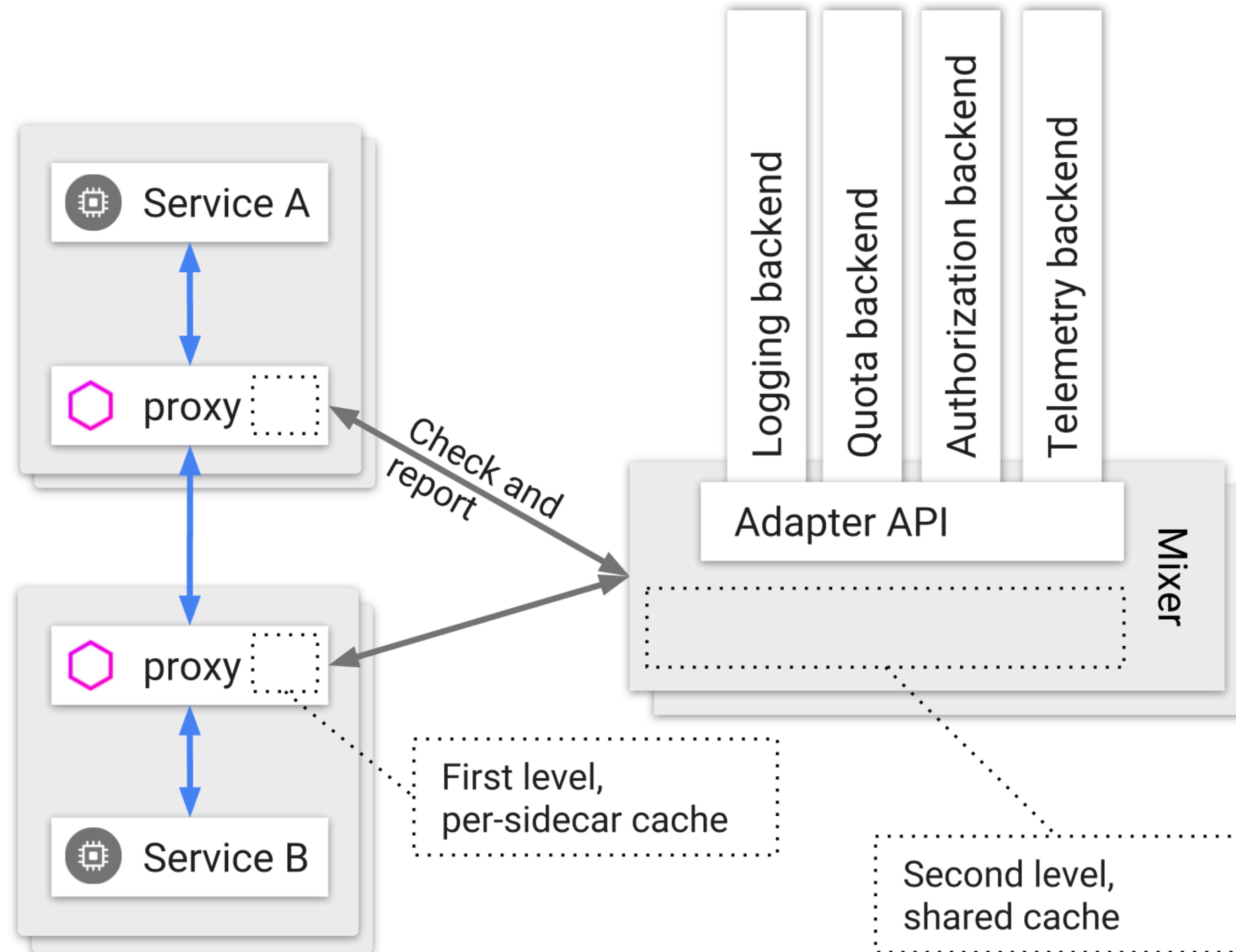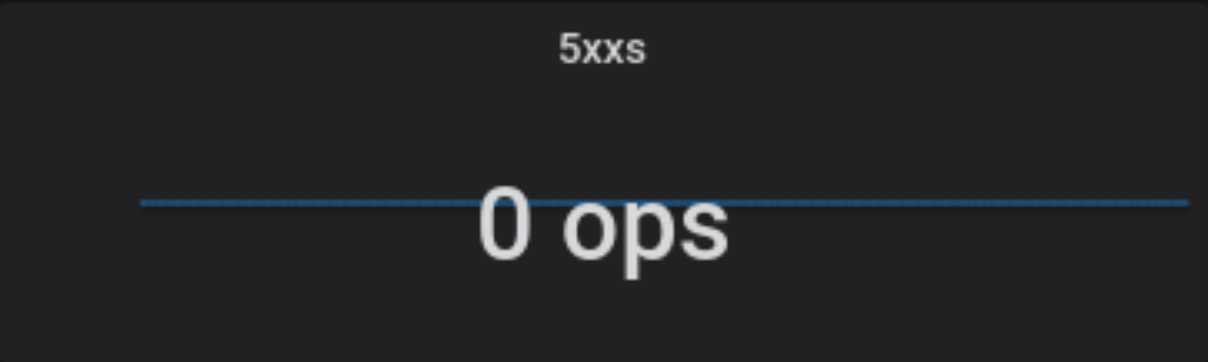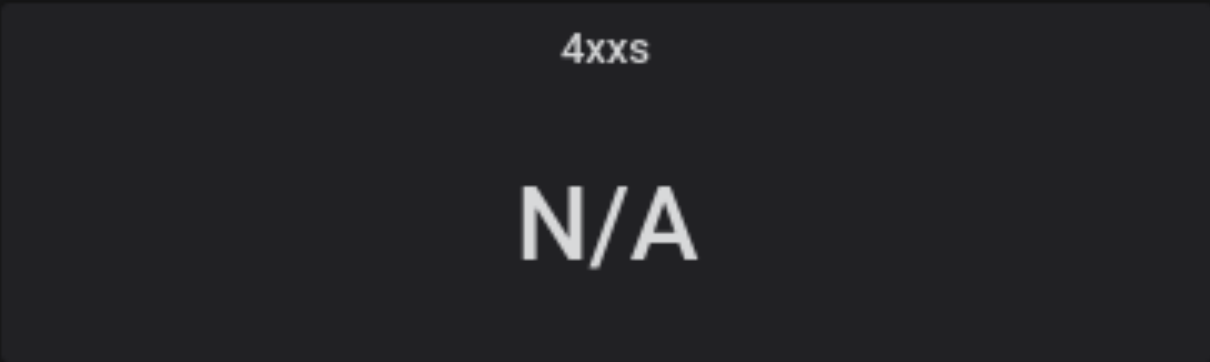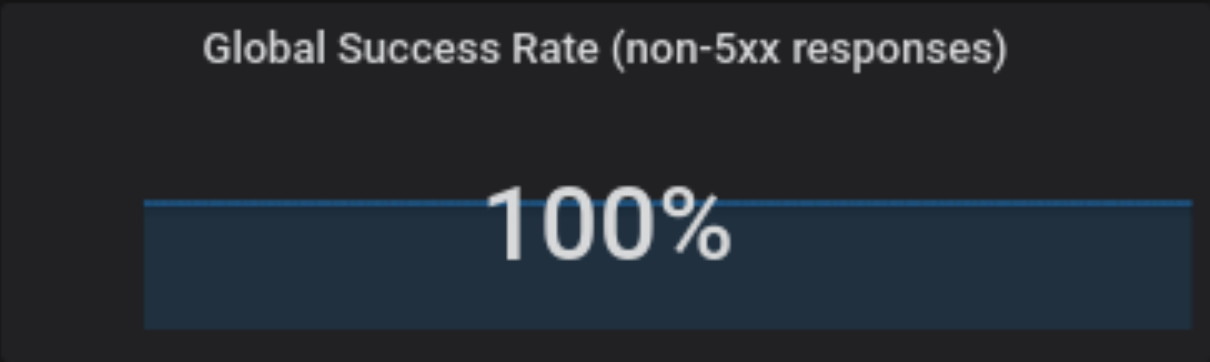 The hardest part. How ?

Control Plane with Istio

Image Source: istio.io

Dynamic Routing

Image Source: istio.io

Istio Mixer

Image Source: istio.io

Istio

Resource Usage: CPU

Resource Usage: Memory
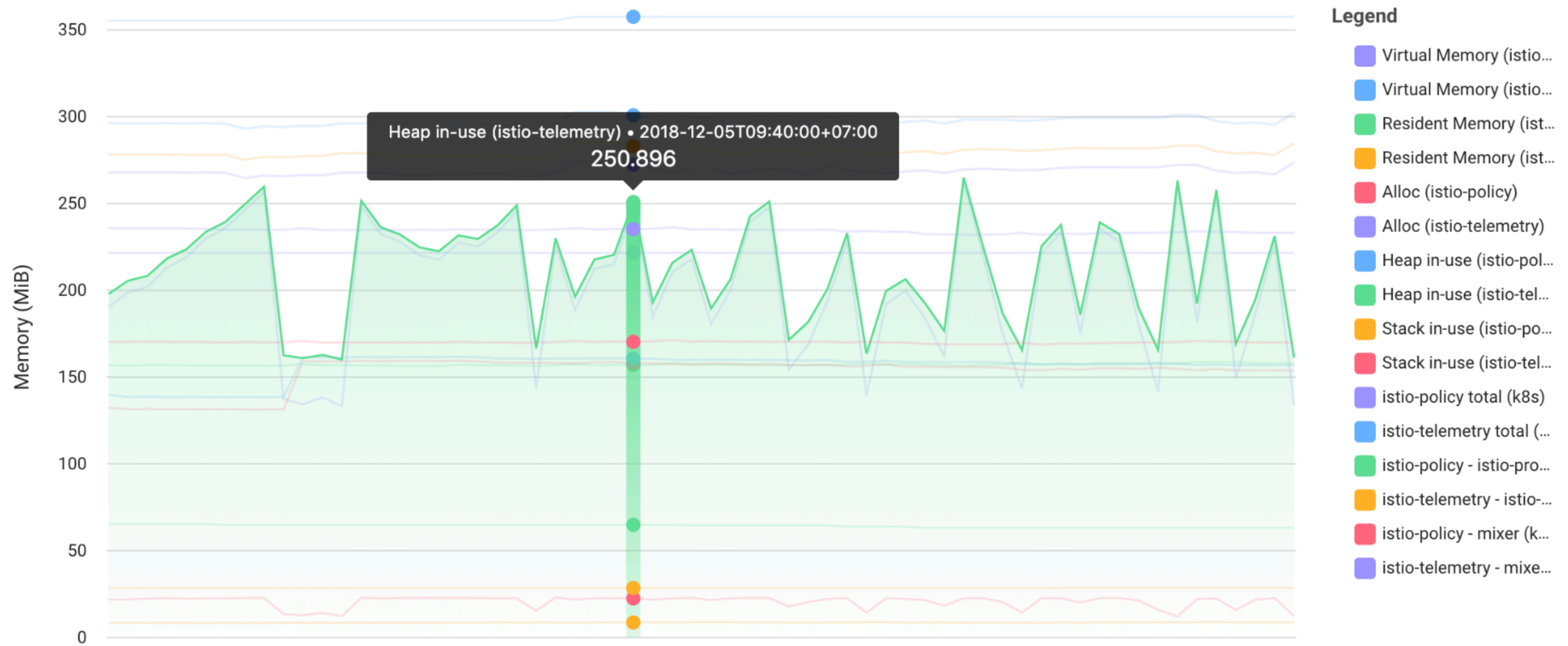
Resource Usage: Disk

Kubernets Summary

▾ Node

Number Of Nodes	Nodes Out of Disk	Nodes Unavailable
3	**0**	**0**

▾ Pods

Pods Running	Pods Pending
18	**0**

Pods Failed	Pods Succeeded	Pods Unknown
0	**0**	**0**

▾ Containers

Containers Running	Containers Waiting	Containers Terminated	Containers Restarts (Last 30 Minutes)
36	**0**	**0**	**0**

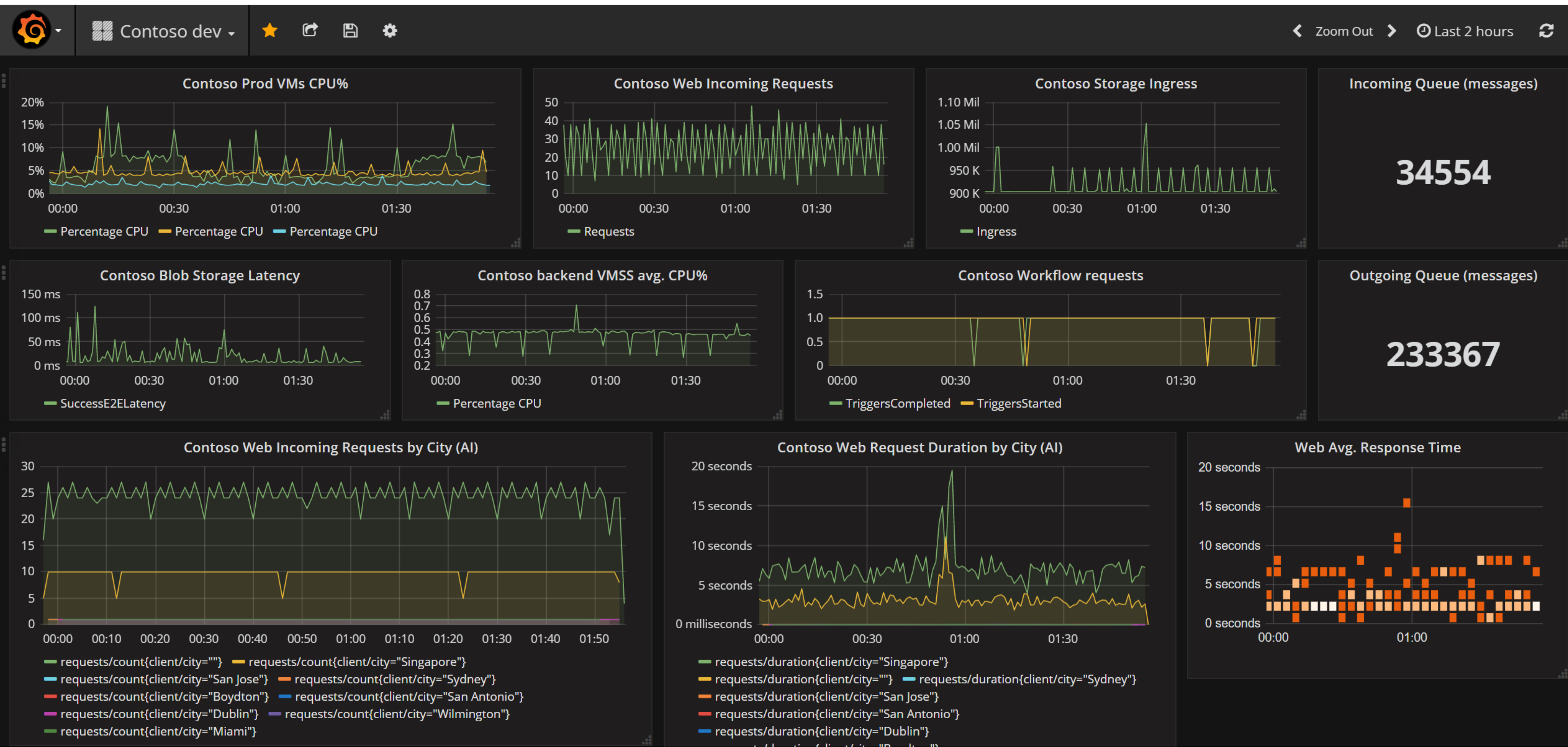| CPU Cores Requested by Containers	Memory Requested By Containers
0.6	**201 MB**

Kubernets Summary (cont)
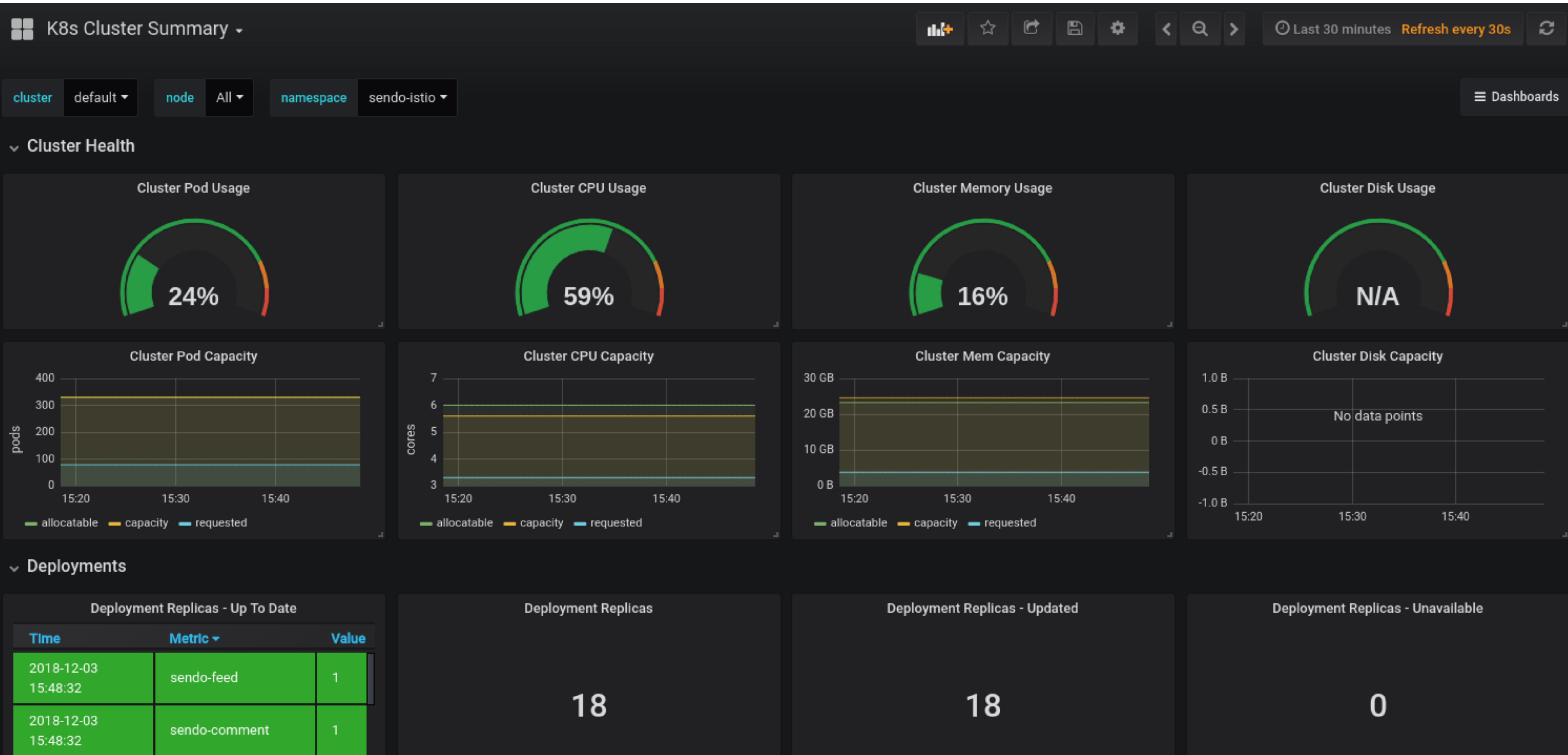
Logging with Fluentd & Elasticsearch
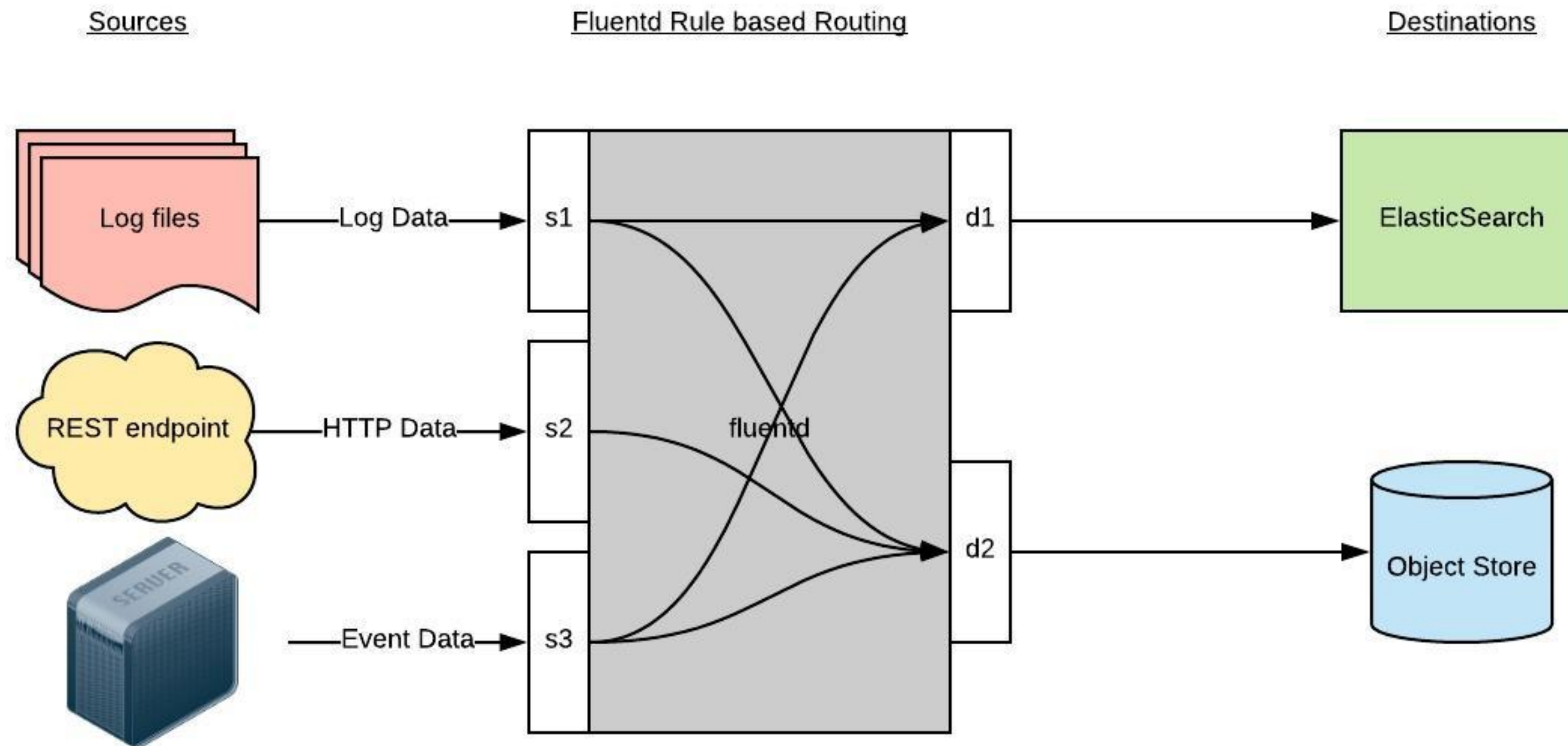
Visualize logging data: Grafana

Problems solved so far

- High latency and low throughput
  Use Go and gRPC for Inter-service communication.

- Duplication data structures
  Use protobuf for generating data structures

- Hard to failure recover on distributed system
  Data Plane and Control Plane: **Istio**, **Kubernetes**
  Logging system: **Fluentd**, **Elasticsearch**, **Grafana**
  Monitoring system: **Netdata**, **Graphite**, **Grafana**

Thank you