ISTIO
An open platform to connect, manage and secure (micro)services

Andreas Neeb
aneeb@redhat.com

whoami

Andreas Neeb,

Chief Architect Financial Services

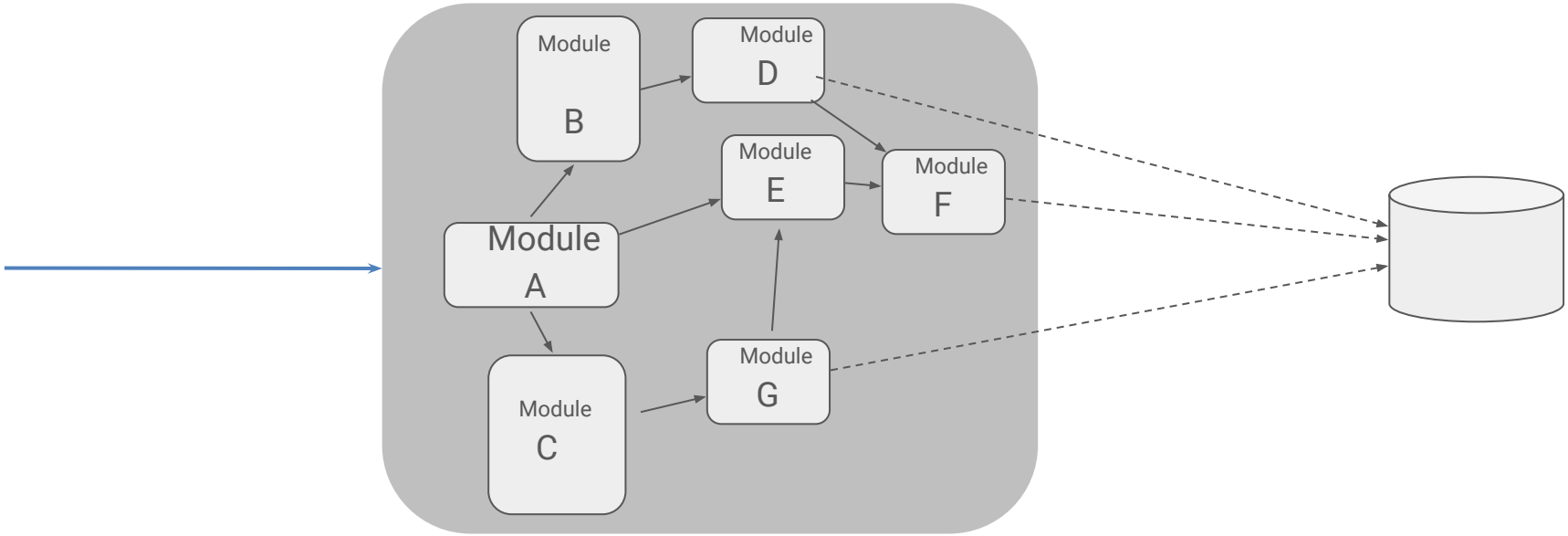
Red Hat GmbH, aneeb@redhat.com

- Strategist
- Container Guy
- Father & Husband
- Nerd
- Football Fan, Scuba Diver, Skier

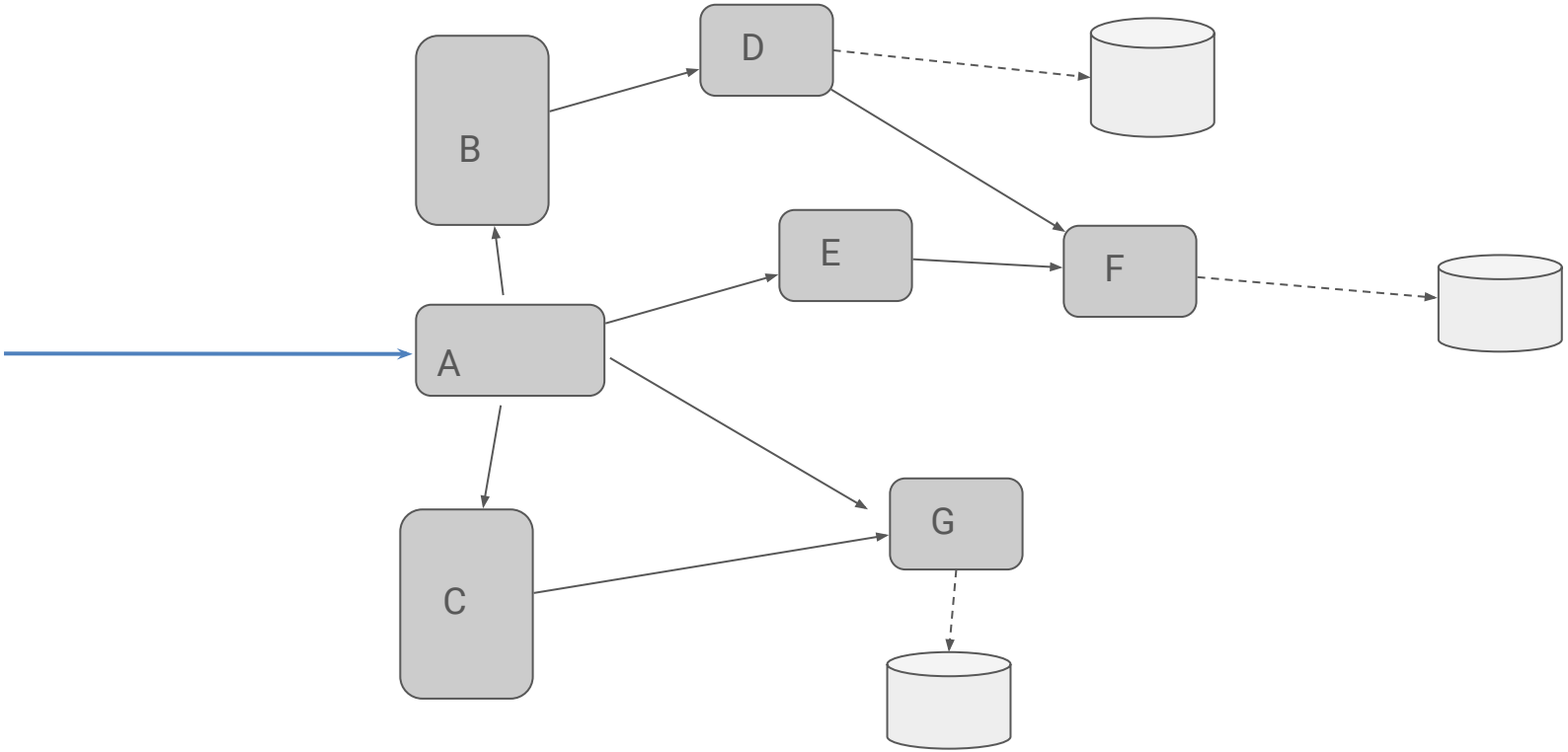
Not necessarily always in that order


A
B

Once Upon a Time ... the Monolith



Microservices!



Microservices!

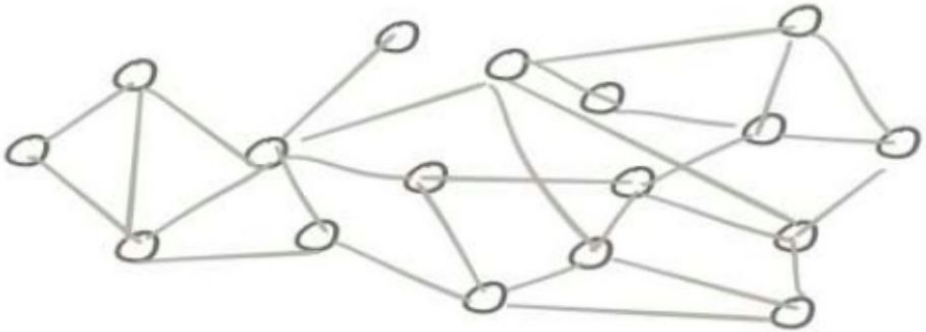
- ★ Agility
- ★ Abstraction
- ★ Scalability

Problem solved!

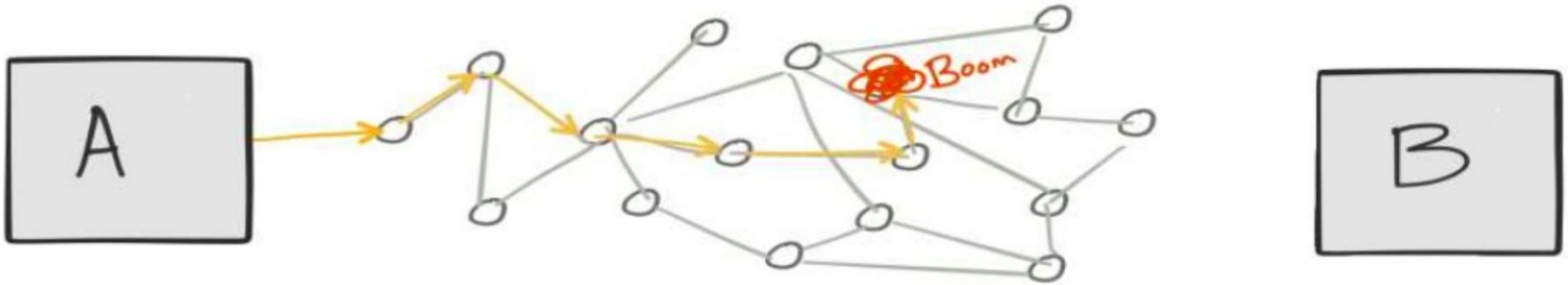

As we move to services architectures, we push the complexity to the space between our services

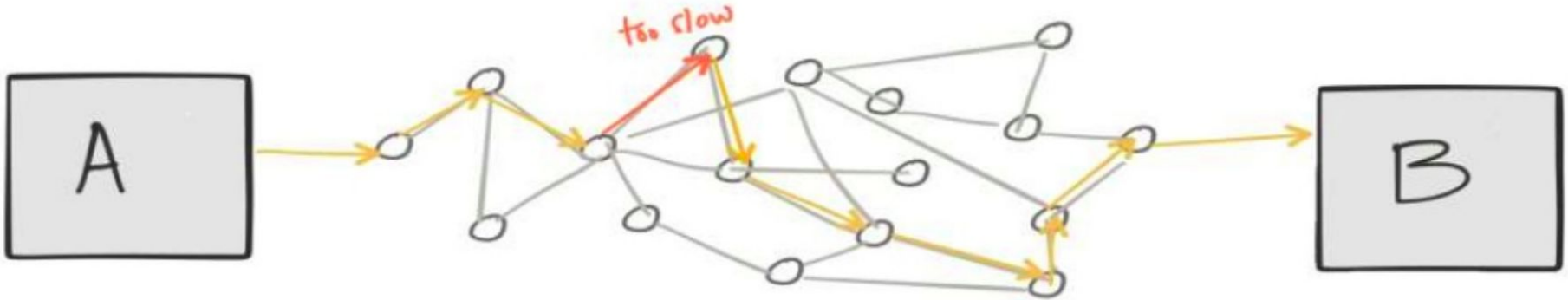

A
B

A



B


A
Boom
B


too slow
A
B

Challenges in a cloudy services world ...

Service discovery

Load balancing

Retries

Timeouts

Circuit breaking

Rate limiting

Thread bulk heading

...

Routing (adaptive, zone-aware)

Deadlines

Back pressure

Outlier detection

Health checking

Traffic shaping

Request shadowing

...

Surgical / fine / per-request routing

A/B rollout

Internal releases / dark launches

Fault injection

Stats, metric, collection

Logging

Tracing

kubernetes

Kubernetes is a deployment platform

It does not adequately address service communication

Entire frameworks have been created to address those concerns

Netflix Hystrix (circuit breaking / bulk heading)

Netflix Zuul (edge router)

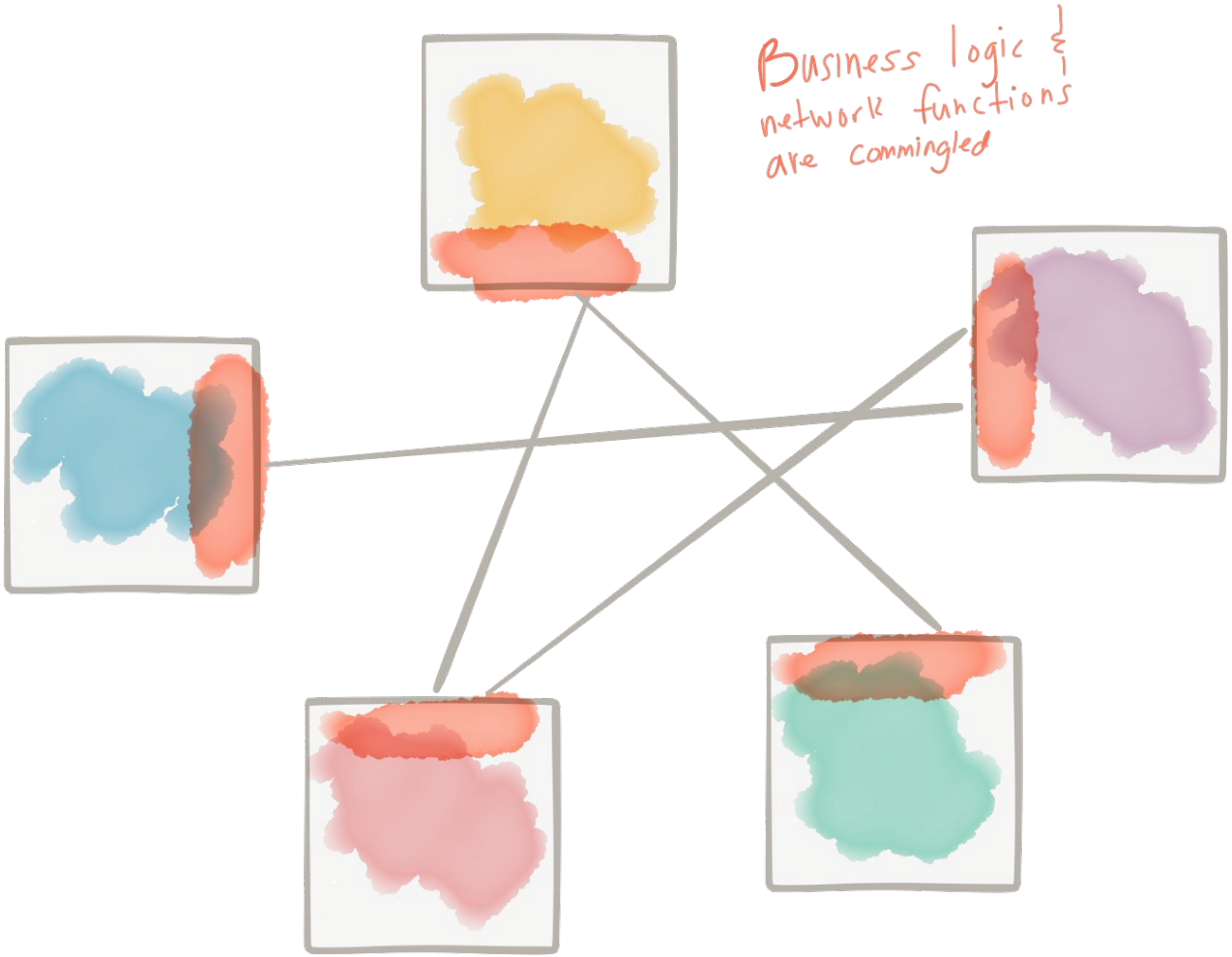
Netflix Ribbon (client-side service discovery / load balance)

Netflix Eureka (service discovery registry)

Brave / Zipkin (tracing)

Netflix spectator / atlas (metrics)

NETFLIX
OSS


Business logic &
network functions
are commingled

But I am using ...

Spring

Vert.x

NodeJS

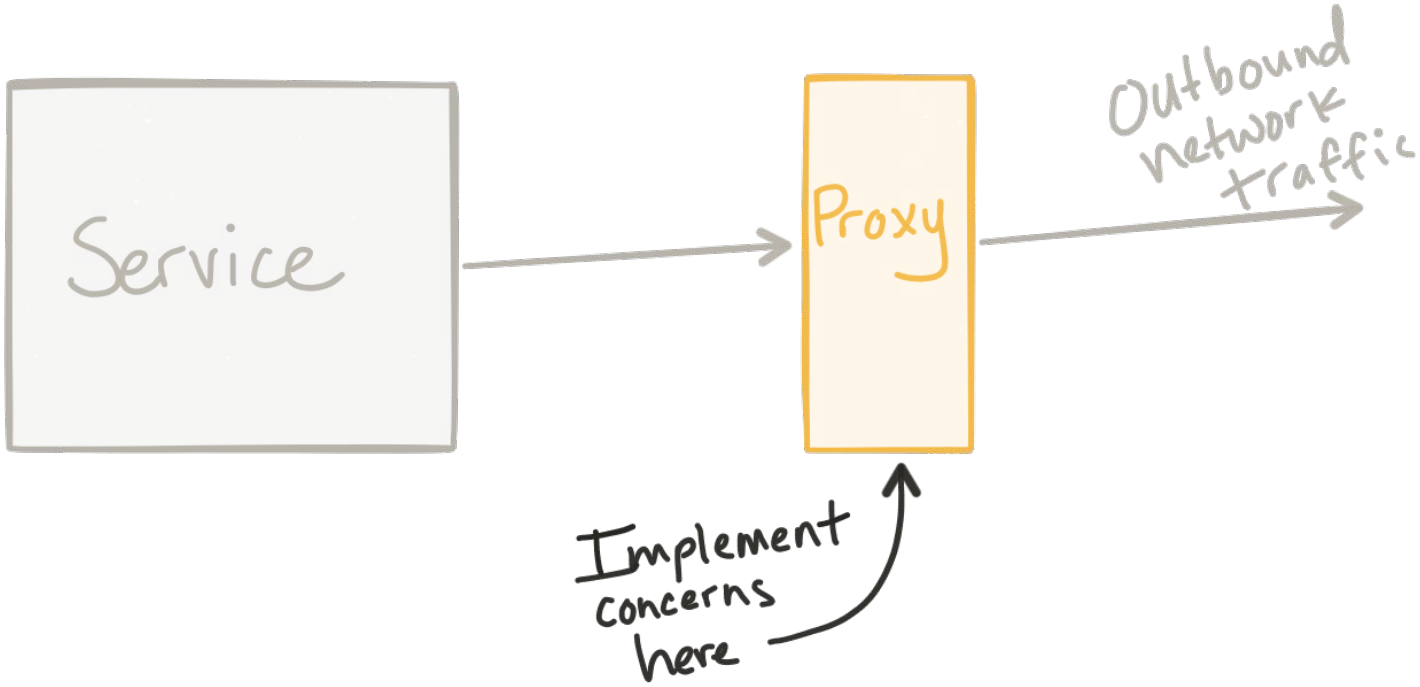
Go

Python

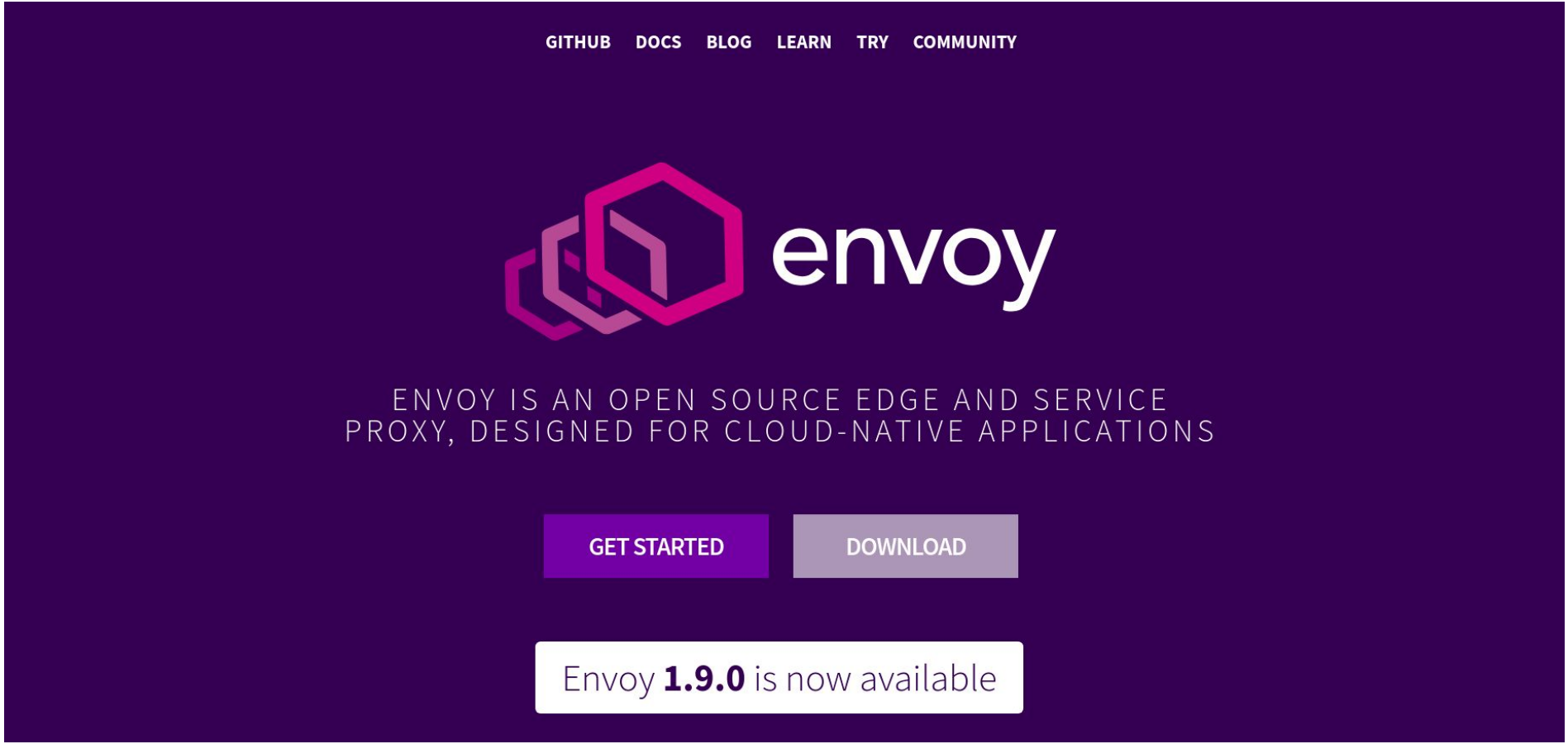
Ruby

C#

How about this?



Hello Envoy!


GITHUB DOCS BLOG LEARN TRY COMMUNITY
envoy
ENVOY IS AN OPEN SOURCE EDGE AND SERVICE PROXY, DESIGNED FOR CLOUD-NATIVE APPLICATIONS
GET STARTED
DOWNLOAD
Envoy 1.9.0 is now available

What is Envoy?

service proxy

C++, highly parallel, non-blocking

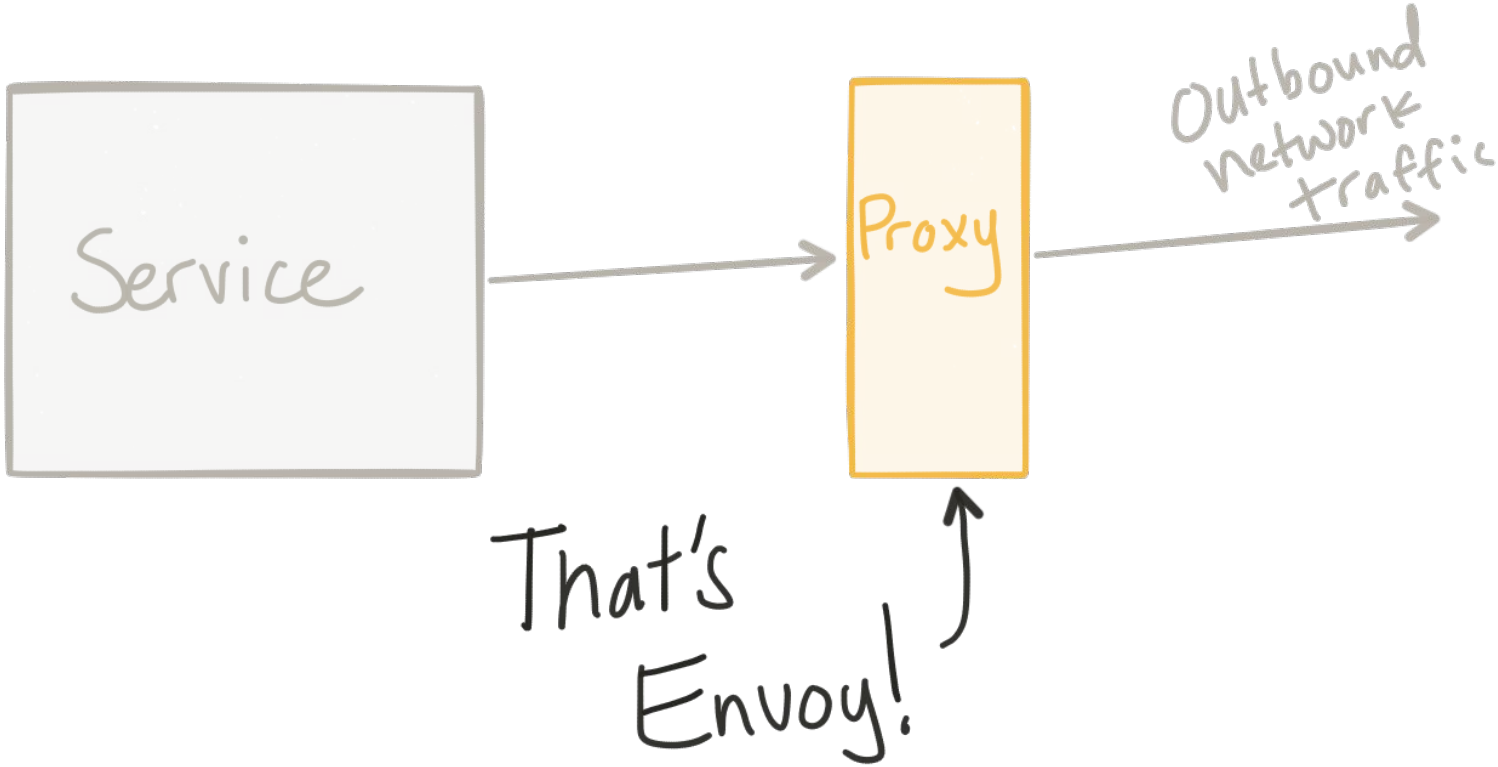
L3/4 network filter, out of the box L7 filters

HTTP 2, including gRPC

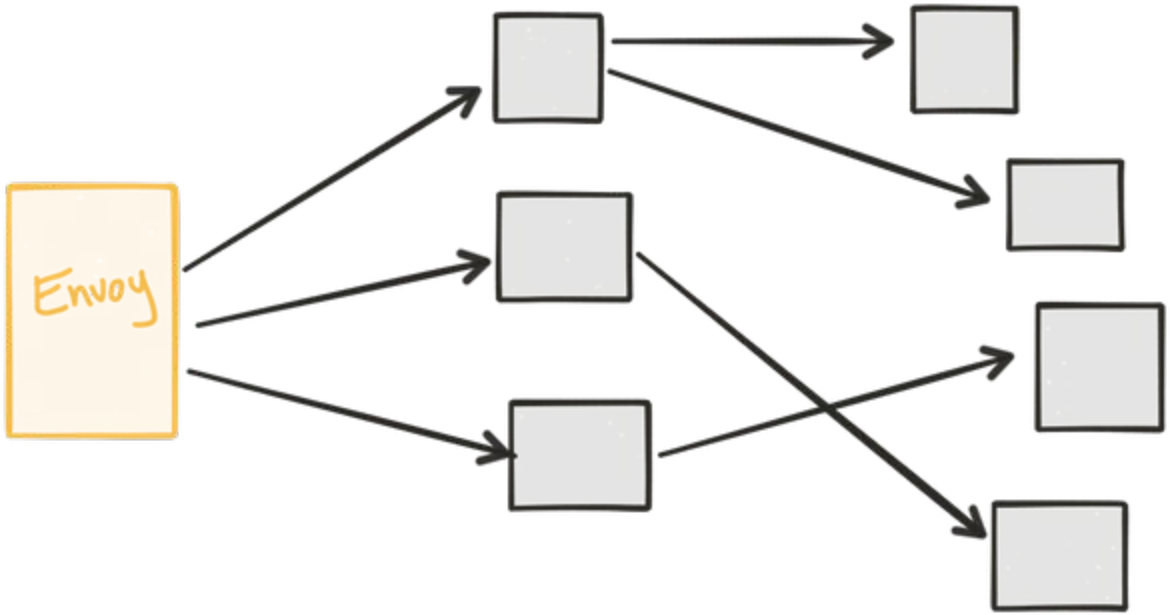
service discovery/health checking

advanced load balancing

stats, metrics, tracing

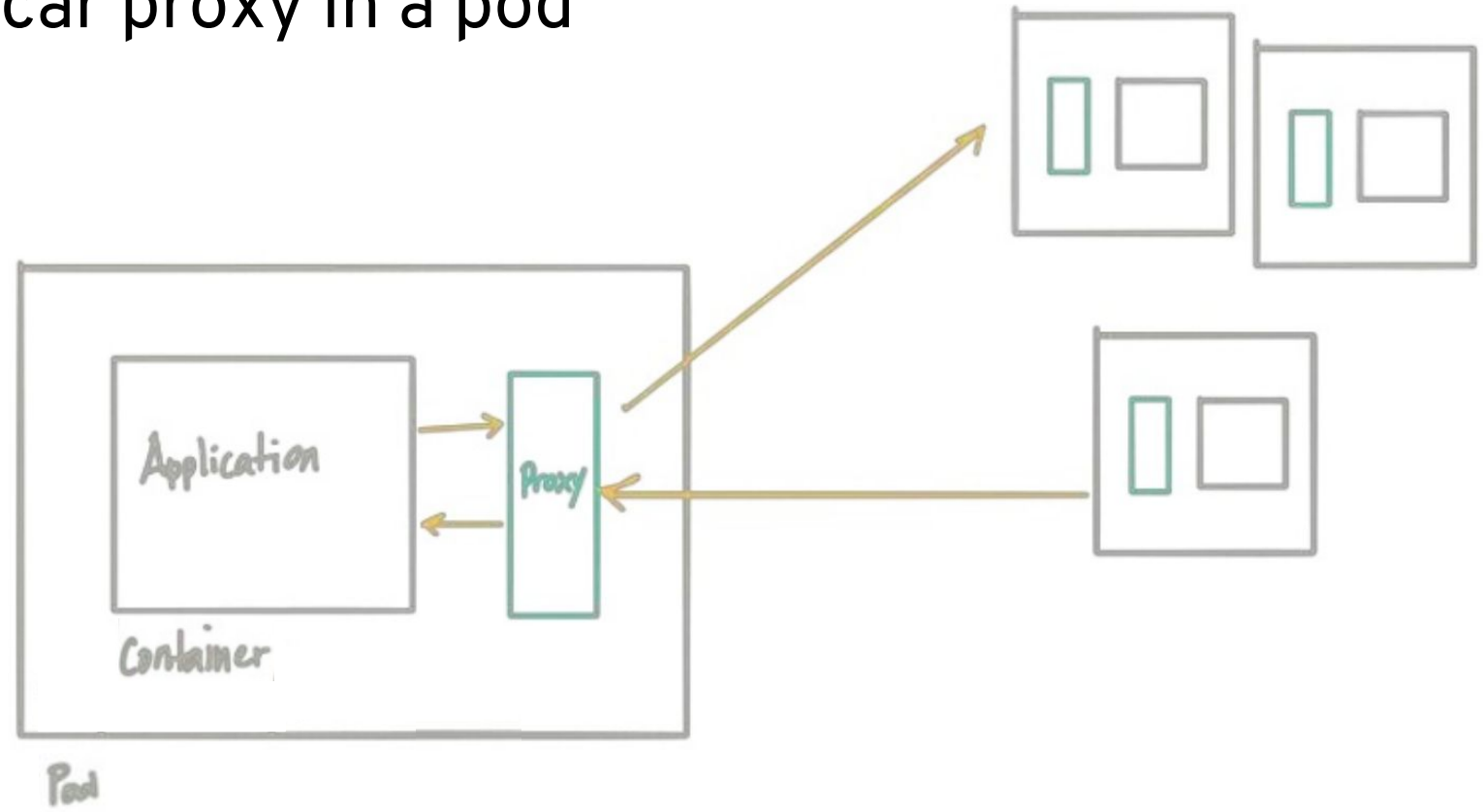

Service
Proxy
Outbound network traffic
That's Envoy!

Envoy as edge proxy

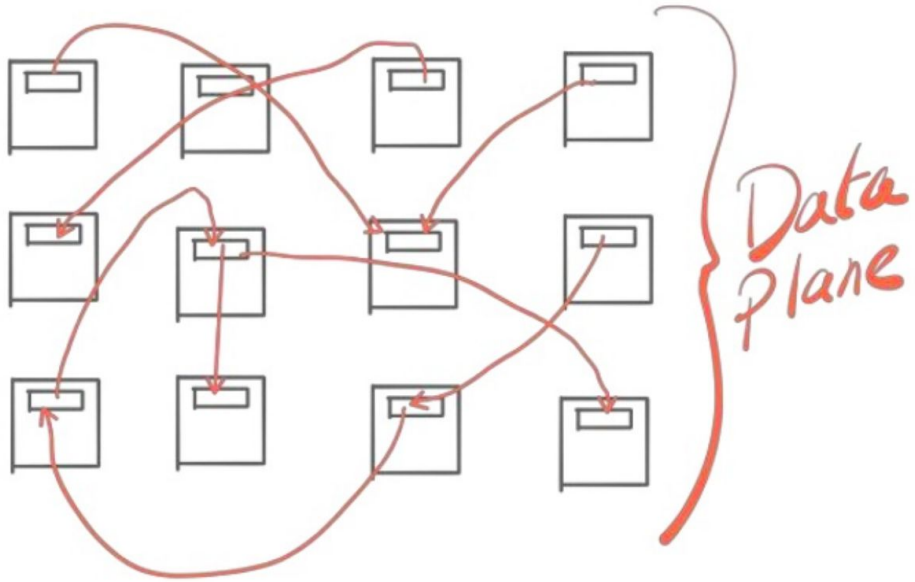


Envoy as sidecar proxy

Sidecar proxy in a pod



All traffic between our applications flows through these proxies. This makes up the “data plane”



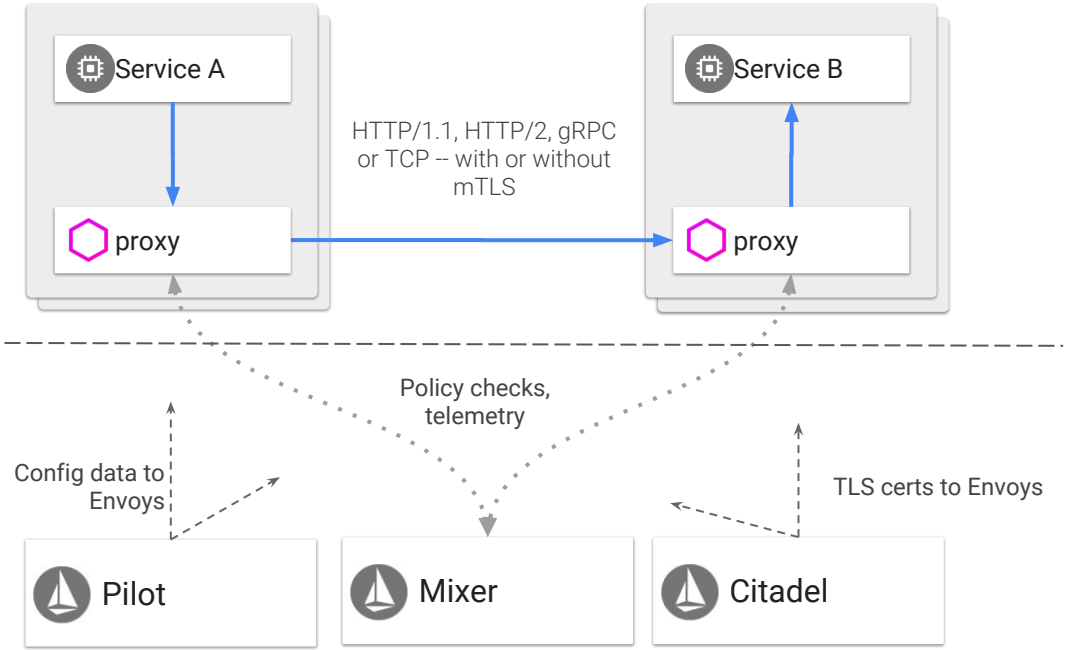
How do we reason about a fleet of Envoys in a large cluster?


istio.io
service mesh

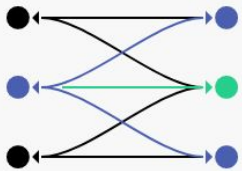
Istio Architecture

Data Plane

Control Plane



Istio Service Mesh -- Connect

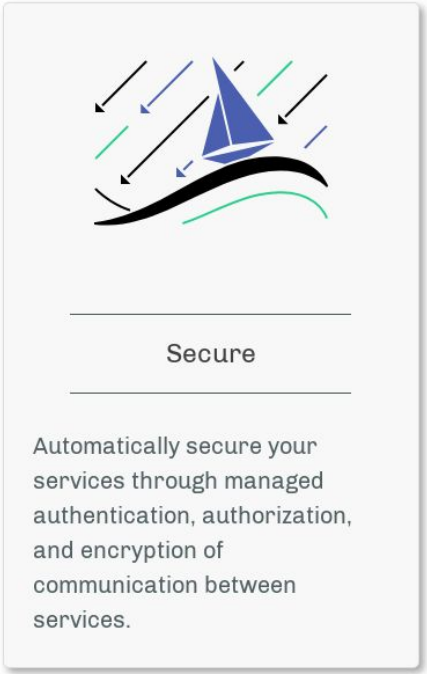


Connect

Intelligently control the flow of traffic and API calls between services, conduct a range of tests, and upgrade gradually with red/black deployments.

- Request routing
- Timeouts
- Retries
- Circuit breaking
- Fault injection
- Traffic shifting
- Mirroring

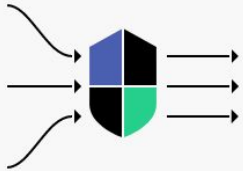
Istio Service Mesh -- Secure



- Identity
- PKI
- Mutual TLS authentication
- Authorization

Istio Service Mesh -- Control

- Rate limits
- Denials
- White & blacklisting

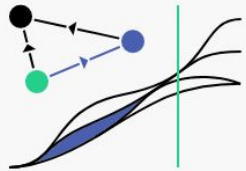


Control

Apply policies and ensure that they're enforced, and that resources are fairly distributed among consumers.

Istio Service Mesh -- Observe

- Service graph
- Metrics
- Logging
- Distributed tracing



Observe

See what's happening with rich automatic tracing, monitoring, and logging of all your services.

live demo

Setup

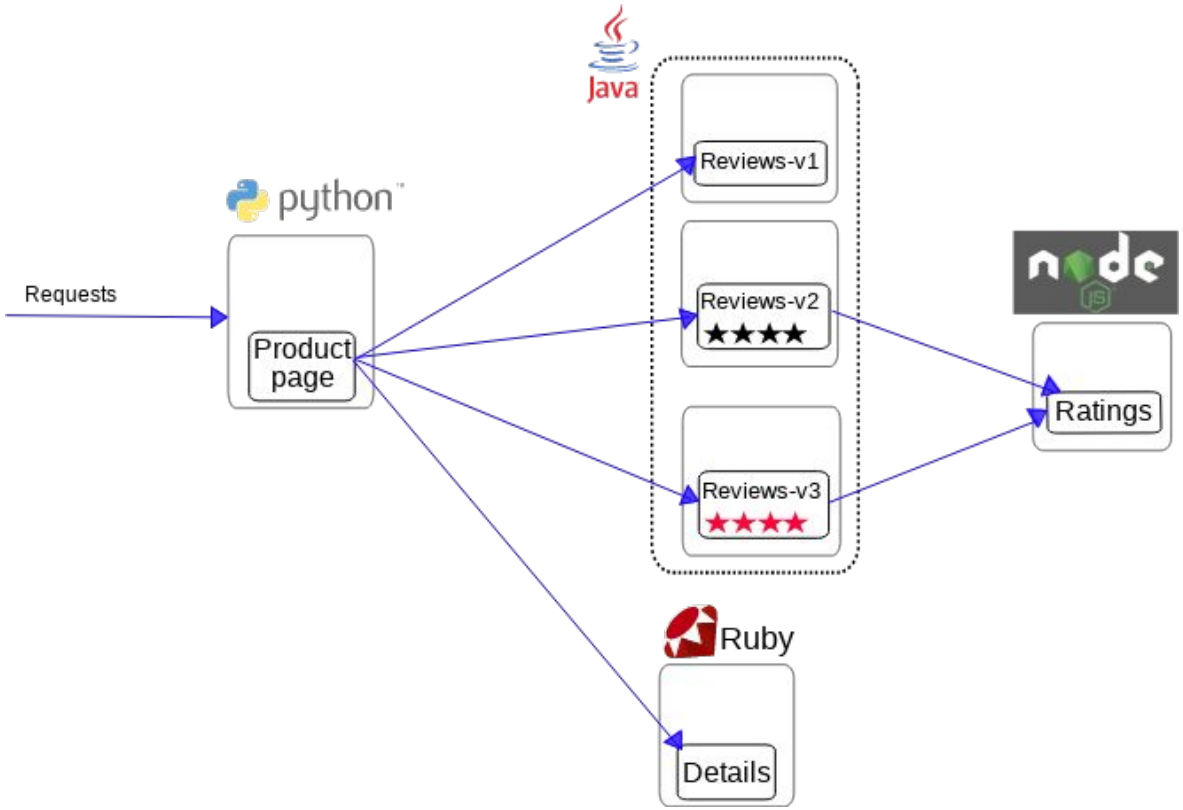
- OpenShift 3.11 (Kubernetes 1.11)
- OpenShift Service Mesh 0.6 (Istio + Kiali + Prometheus + Grafana + Jaeger)
- Bookinfo

Wanna try?

https://learn.openshift.com

https://github.com/redhat-developer-demos/istio-tutorial

bookinfo


Java
python
Requests
Product page
Reviews-v1
Reviews-v2
★★★★
Reviews-v3
★★★★
Ratings
Ruby
Details

Use Cases

Request routing

(all) -> reviews v1
(user: jason) -> reviews v2

Fault injection

(user: jason) -> reviews v2 - (delay) -> ratings

Traffic Shifting

(all) - (canary) -> reviews v3

USE CASE 1 - Request routing

Use Case 1

Request routing

(all) -> reviews v1
(user: jason) -> reviews v2

Send all traffic to v1

oc apply -f virtual-service-all-v1.yaml

oc get virtualservice reviews -o yaml

```
metadata:
  name: reviews

spec:
  hosts:
  - reviews
  http:
  - route:
    - destination:
        host: reviews
        subset: v1
```

oc get destinationrule reviews -o yaml

```
metadata:
  name: reviews

spec:
  host: reviews
  subsets:
  - labels:
      version: v1
    name: v1
  - labels:
      version: v2
    name: v2
  - labels:
      version: v3
    name: v3
  trafficPolicy:
    tls:
      mode: ISTIO_MUTUAL
```

```
oc get pods --show-labels | grep reviews

reviews-v1-8568fdcf99-28hw9     2/2     Running     0       49m     version=v1
reviews-v2-8596f84c5-6bhmz      2/2     Running     0       49m     version=v2
reviews-v3-56568b7595-6hhpg     2/2     Running     0       49m     version=v3
```

Send only user ‘Jason’ to v2

oc apply -f virtual-service-reviews-test-v2.yaml

oc get virtualservice reviews -o yaml

```
metadata:
  name: reviews

spec:
  hosts:
  - reviews
  http:
  - match:
    - headers:
        end-user:
          exact: jason
    route:
    - destination:
        host: reviews
        subset: v2
  - route:
    - destination:
        host: reviews
        subset: v1
```

USE CASE 2 - Fault injection

Use Case 2

Fault injection

(user: jason) -> reviews v2 - (delay) -> ratings

Inject delay for user ‘Jason’

oc apply -f virtual-service-ratings-test-delay.yaml

oc get virtualservice ratings -o yaml

```
metadata:
  name: ratings

spec:
  hosts:
  - ratings
  http:
  - fault:
      delay:
        fixedDelay: 7s
        percent: 100
    match:
    - headers:
        end-user:
          exact: jason
    route:
    - destination:
        host: ratings
        subset: v1

...
```

Understand what happend

The timeout between the productpage and the reviews service is 6 seconds - coded as 3s + 1 retry for 6s total. The timeout between the reviews and ratings service is hard-coded at 10 seconds. Because of the delay we introduced, the /productpage times out prematurely and throws the error.

USE CASE 3 - Traffic shifting

Use Case 3

Traffic Shifting

(all) - (canary) -> reviews v3

Cleanup: all to v1

oc apply -f virtual-service-all-v1.yaml

Transfer 50% from reviews:v1 to reviews:v3

oc apply -f virtual-service-reviews-50-v3.yaml

oc get virtualservice reviews -o yaml

```
metadata:
  name: reviews

spec:
  hosts:
  - reviews
  http:
  - route:
    - destination:
        host: reviews
        subset: v1
      weight: 50
    - destination:
        host: reviews
        subset: v3
      weight: 50
```

Route 100% to reviews:v3

oc apply -f virtual-service-reviews-v3.yaml

redhat.

THANK YOU

plus.google.com/+RedHat

facebook.com/redhatinc

linkedin.com/company/red-hat

twitter.com/RedHat

youtube.com/user/RedHatVideos