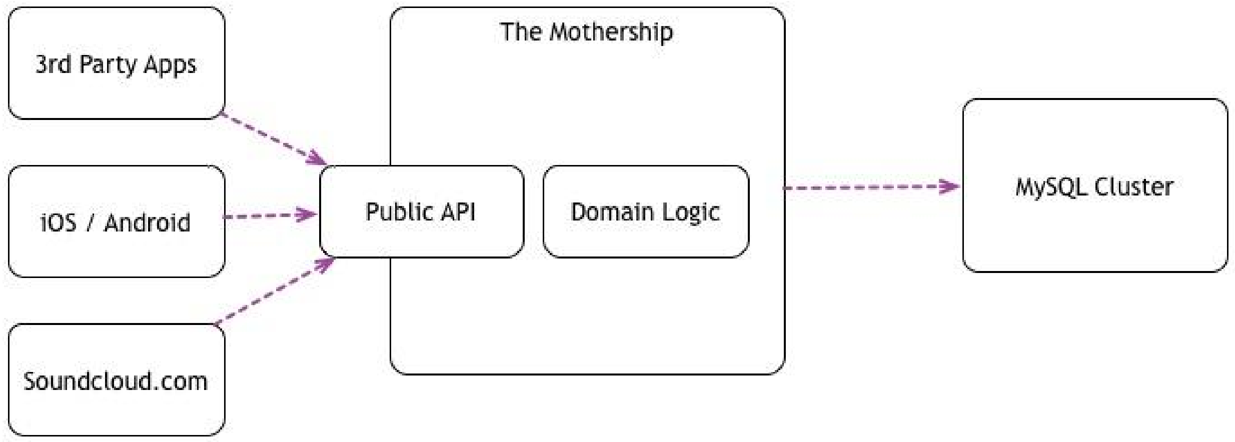
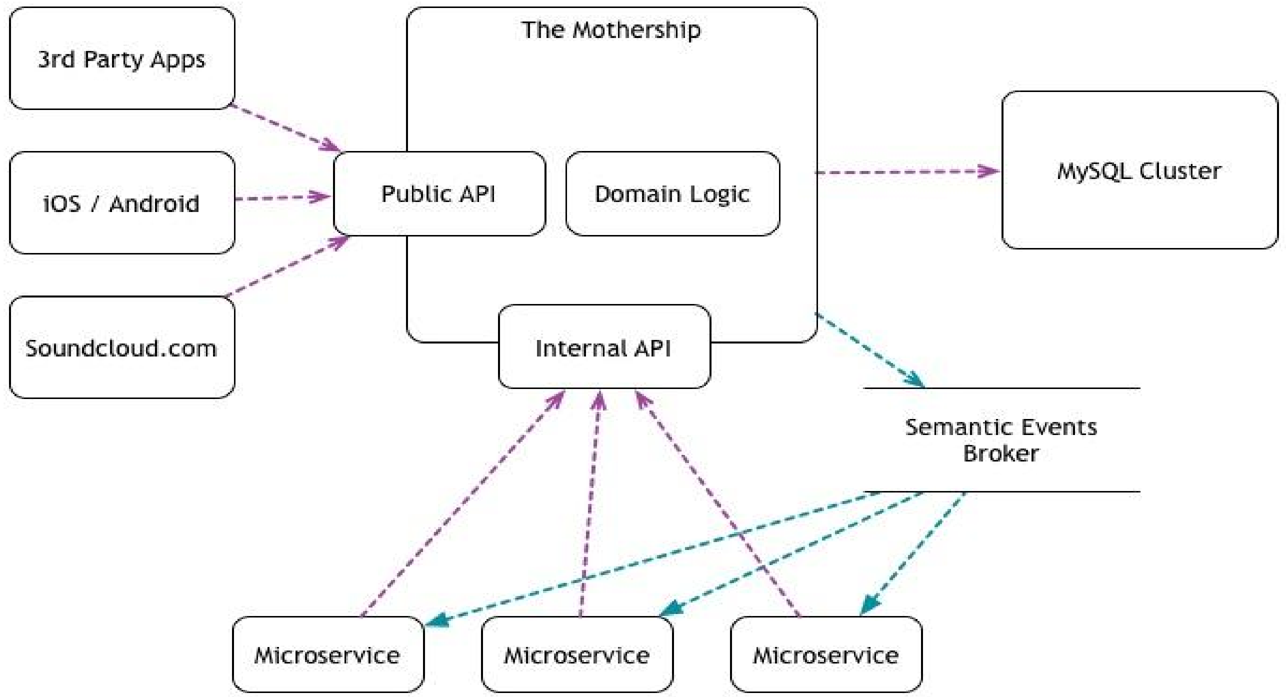
Monitoring a **Kubernetes-backed** microservice architecture with **Prometheus**

Björn “Beorn” Rabenstein — *SoundCloud*
Fabian Reinartz — *CoreOS*

SoundCloud 2012 – from monolith …



… to microservices



Orchestration needed

THE TWELVE-FACTOR APP

heroku

Run containers in a cluster…

In-house innovation: *Bazooka* – PaaS, Heroko style.

Problems:

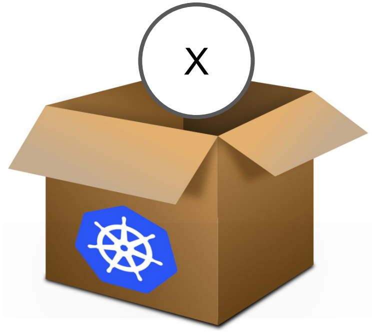
- Only 12-factor apps (stateless etc.).
- Limited resource isolation.
- No sidecars.
- Maturity.

Meanwhile, the open-source world has evolved…

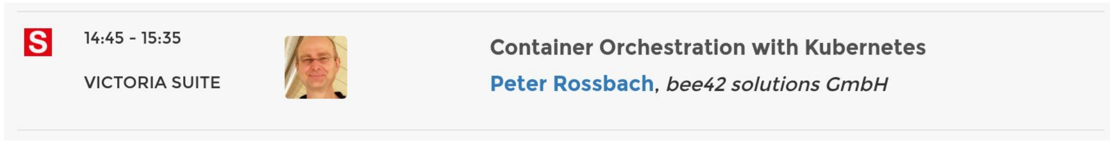
KUBERNETES

Kubernetes

- inspired by Google's Borg
- not Borg



Today:

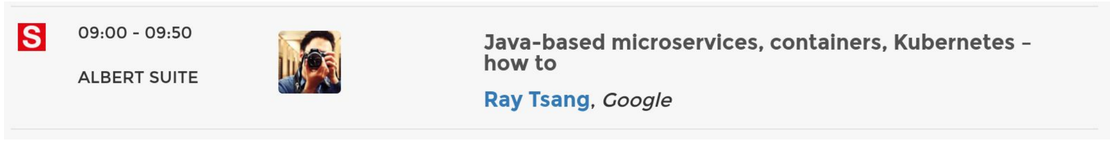


S 14:45 - 15:35 VICTORIA SUITE

Container Orchestration with Kubernetes

Peter Rossbach, *bee42 solutions GmbH*

Tomorrow:

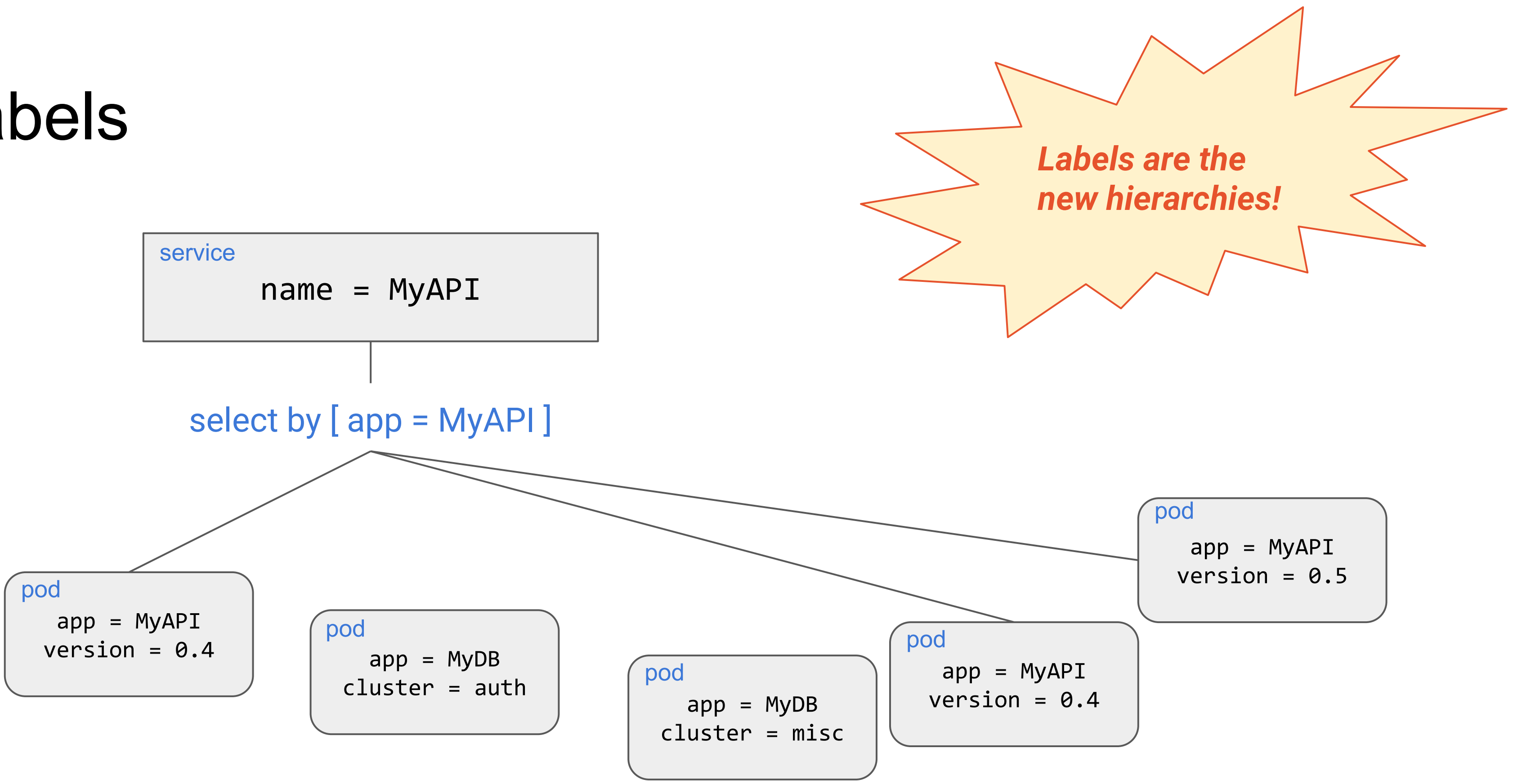


S 09:00 - 09:50 ALBERT SUITE

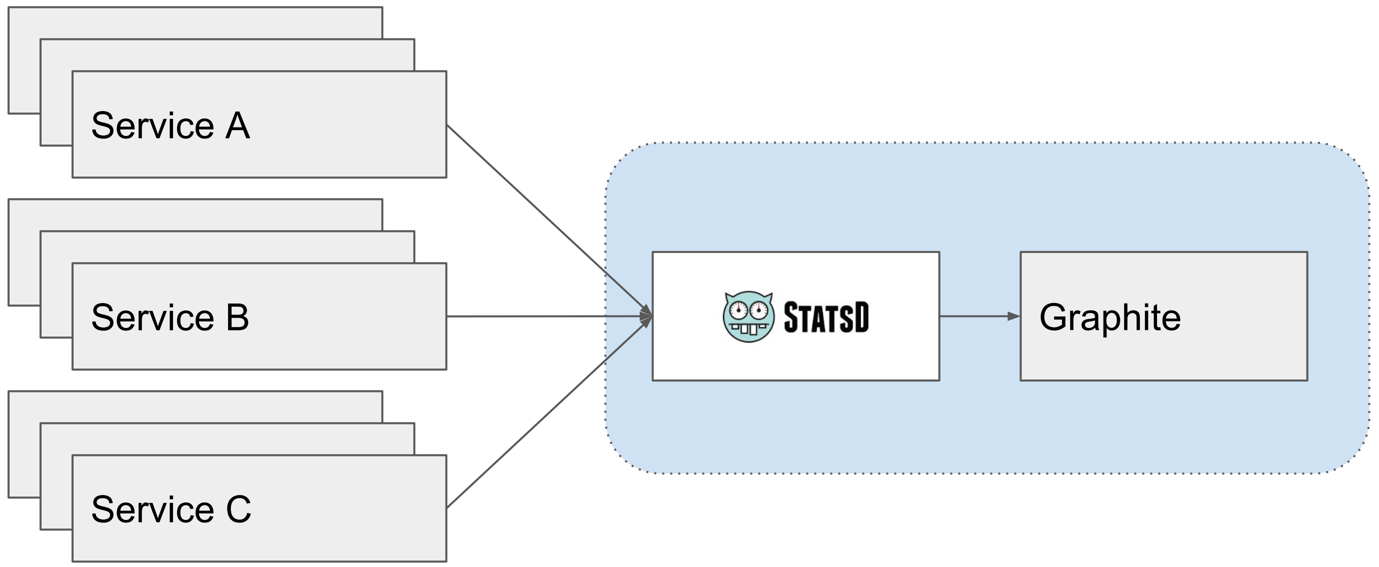
Java-based microservices, containers, Kubernetes – how to

Ray Tsang, *Google*

Labels


Labels are the new hierarchies!
service
name = MyAPI
select by [app = MyAPI]
pod
app = MyAPI
version = 0.4
pod
app = MyDB
cluster = auth
pod
app = MyDB
cluster = misc
pod
app = MyAPI
version = 0.4
pod
app = MyAPI
version = 0.5

Monitoring at SC 2012



Monitoring challenges

- A lot of traffic to monitor
  - Monitoring traffic should not be proportional to user traffic
- Way more targets to monitor
  - One host can run many containers
- And they constantly change
  - Deploys, scaling, rescheduling unhealthy instances …
- Need a fleet-wide view.
  - What’s my overall 99th percentile latency?
- Still need to be able to drill down for troubleshooting
  - Which instance/endpoint/version/... causes those errors I’m seeing?
- Meaningful alerting
  - Symptom-based alerting for pages, cause-based alerting for warnings
  - See Rob Ewaschuk’s *“My philosophy on alerting"* https://goo.gl/2vrpSO

Monitor everything, all levels, with the same system

Level	**What to monitor (examples)**	**What exposes metrics (example)**
Network	Routers, switches	SNMP exporter
Host (OS, hardware)	Hardware failure, provisioning, host resources	Node exporter
Container	Resource usage, performance characteristics	cAdvisor
Application	Latency, errors, QPS, internal state	Your own code
Orchestration	Cluster resources, scheduling	Kubernetes components

“Obviously the solution to all our problems with everything forever, right?”

Benjamin Staffin, Fitbit Site Operations

PROMETHEUS

Prometheus

- inspired by Google’s Borgmon
- not Borgmon

- initially developed at SoundCloud, open-source from the beginning
- public announcement early 2015

- collects metrics at scale via HTTP (think: yet another client of your microservice)
- thousands of targets, millions of time series, 800k samples/s, no dependencies
- easy to scale

Features – multi-dimensional data m

Labels are the new hierarchies!

```
http_requests_total{instance="web-1", path="/index", status= 00", method="GET"}
http_requests_total{instance="web-1", path="/index", status="404", method="POST"}
http_requests_total{instance="web-3", path="/index", status="200", method="GET"}
```

```
#metrics x #values(instance) x #values(path) x #values(status) x #values(method)
```

- millions of time series

Features – powerful query language

The questions to ask are often not known beforehand.

The 3 path-method combinations with the highest number of failing requests?

```
topk(3, sum by(path, method) (
    rate(http_requests_total{status=~"5.."}[5m])
))
```

The 99th percentile request latency by request path?

```
histogram_quantile(0.99, sum by(le, path) (
    rate(http_requests_duration_seconds_bucket[5m])
))
```

Features – powerful query language

```
topk(3, sum by(path, method) (
    rate(http_requests_total{status=~"5.."}[5m])
))
```

```
{path="/api/comments", method="POST"}            105.4
{path="/api/user/:id", method="GET"}             34.122
{path="/api/comment/:id/edit", method="POST"}    29.31
```

Features – easy instrumentation

```
from prometheus_client import start_http_server, Histogram


# Create a metric to track time spent and requests made.
REQUEST_TIME = Histogram('request_processing_seconds', 'Time spent processing request')

# Decorate function with metric.
@REQUEST_TIME.time()
def process_request(t):
    # do work ...
    return


start_http_server(8000)
```

Integrations (selection)

amazon web services™ collectd CONSUL Grafana InfluxDB

Nagios® OPENTSDB HA PROXY node JS

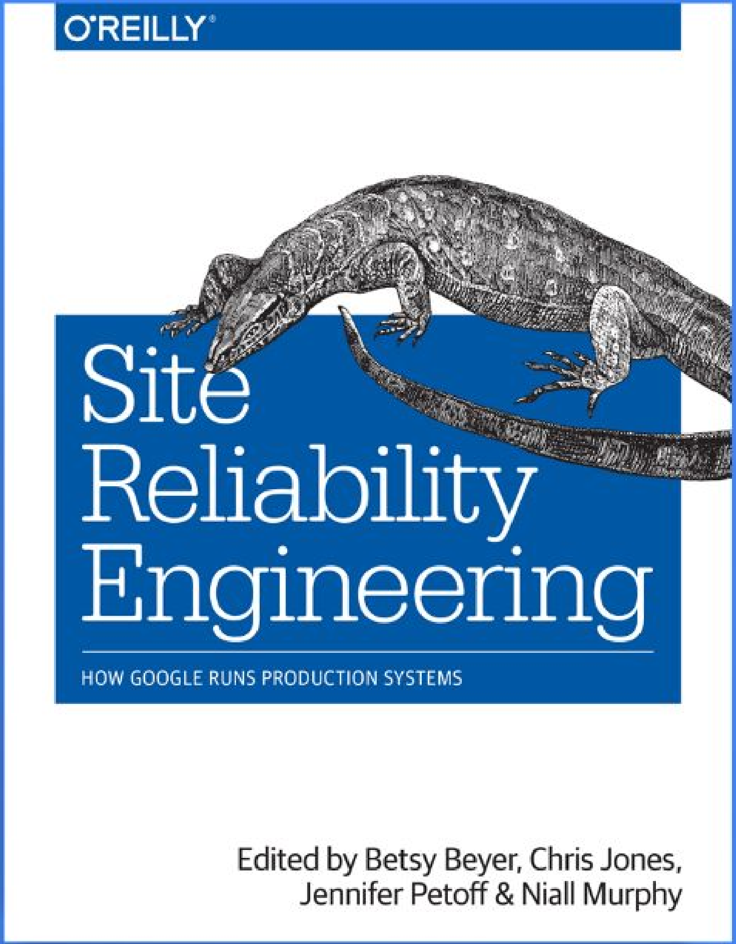
MESOS MySQL® STATSD cAdvisor Java

mongoDB redis MINECRAFT CouchDB

docker RabbitMQ™ Microsoft® .NET

#! etcd New Relic. MUNIN PostgreSQL RethinkDB

KUBERNETES
PROMETHEUS



O'REILLY

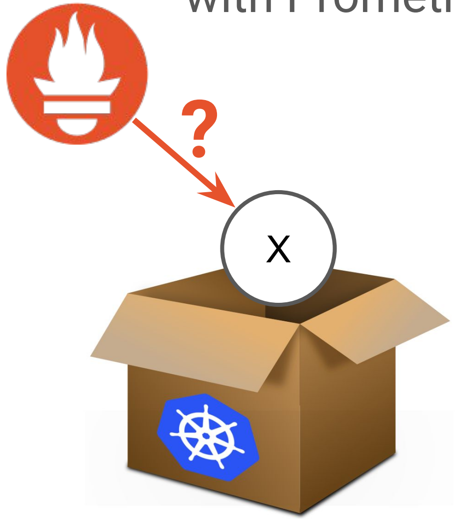
Site Reliability Engineering

HOW GOOGLE RUNS PRODUCTION SYSTEMS

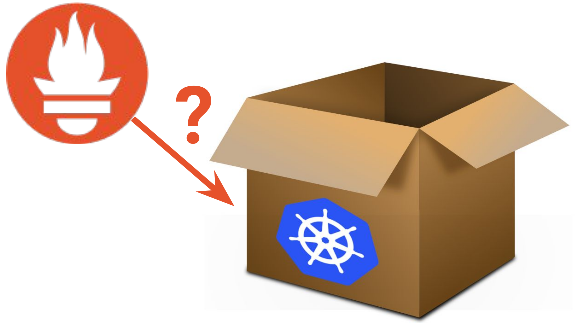
Edited by Betsy Beyer, Chris Jones, Jennifer Petoff & Niall Murphy

Three questions

How to monitor services running on Kubernetes with Prometheus?

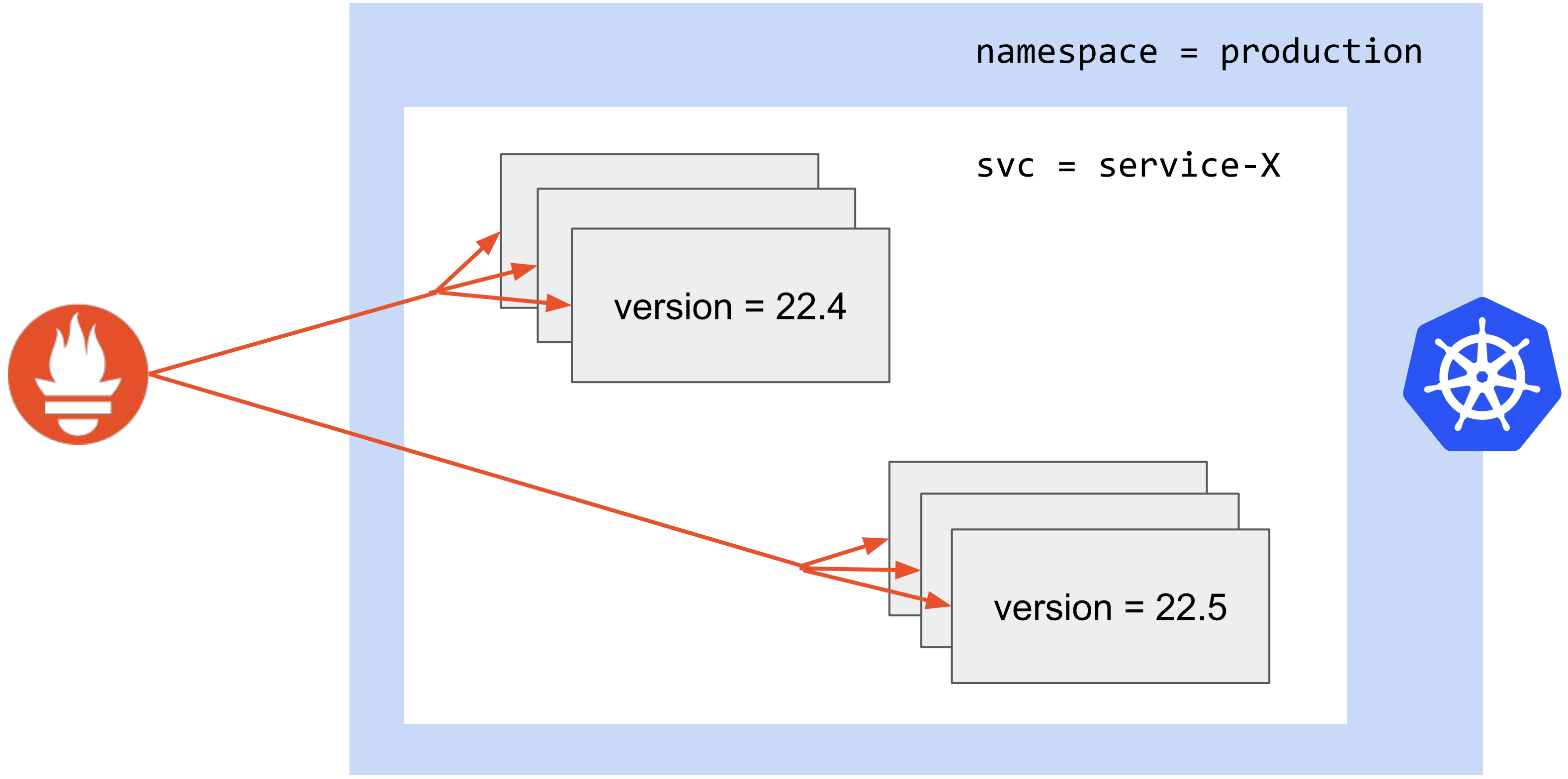


How to monitor Kubernetes and containers with Prometheus?

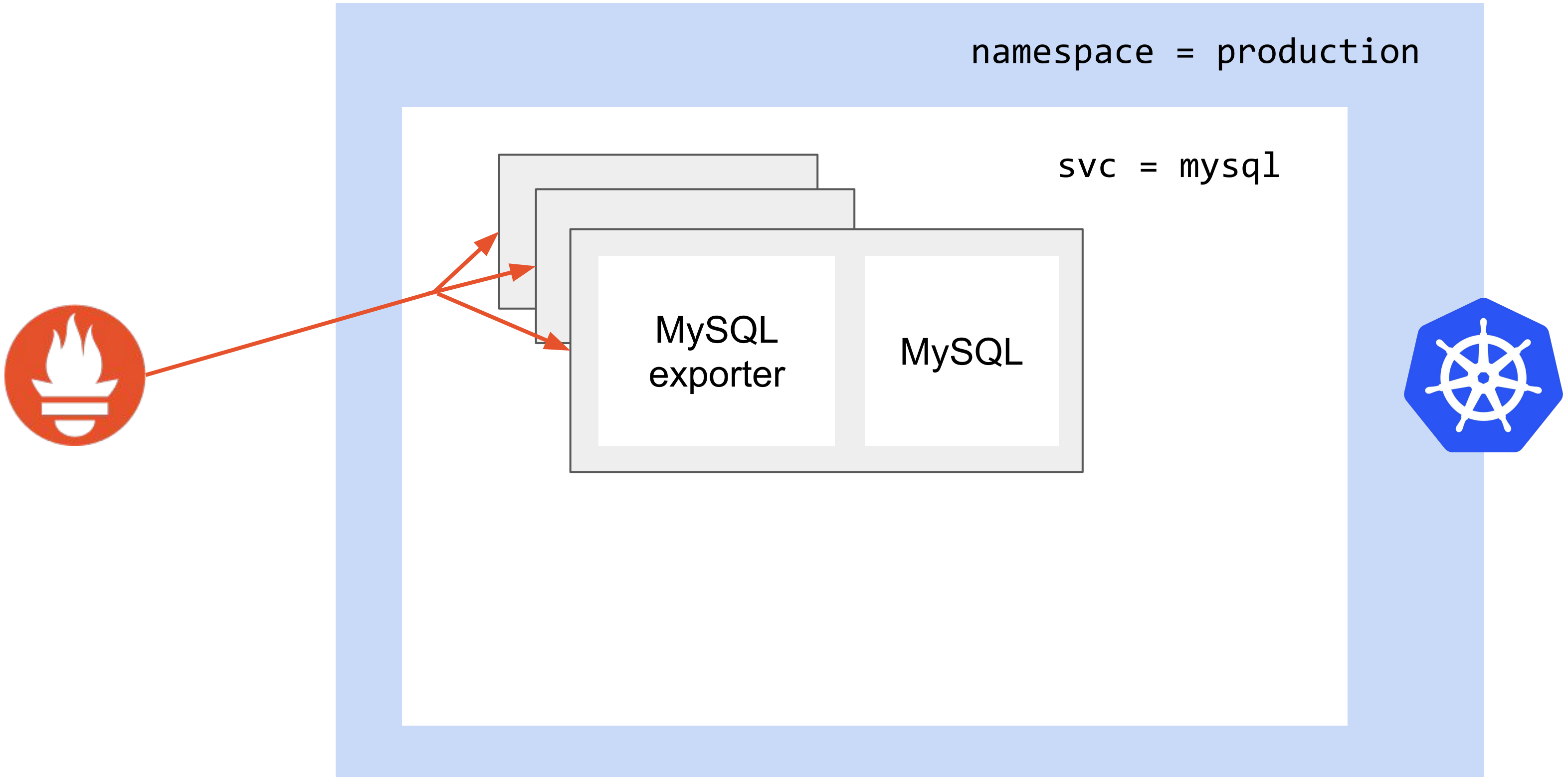


How to run Prometheus on Kubernetes?

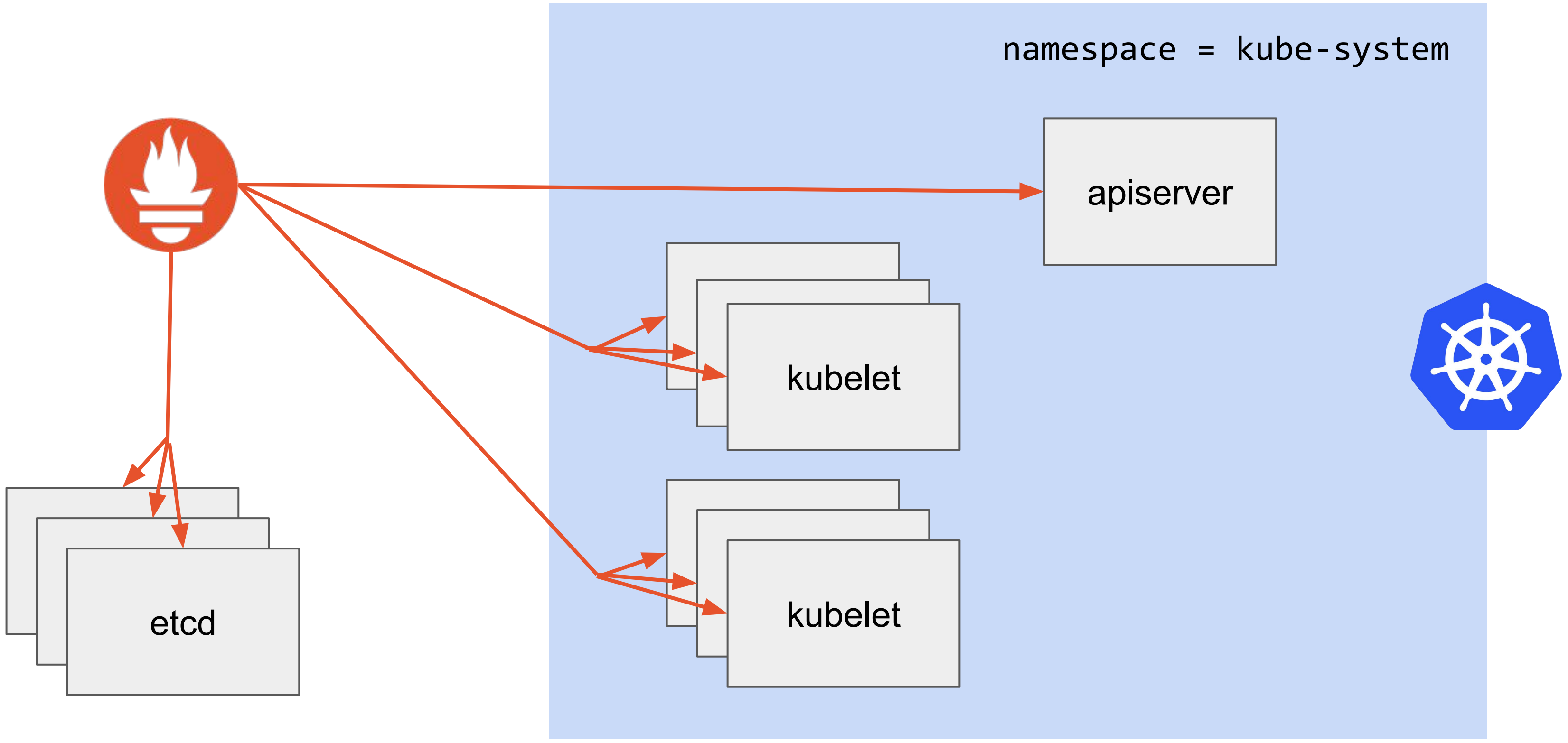
Monitoring Services



Monitoring Services via Exporters



Monitoring Kubernetes



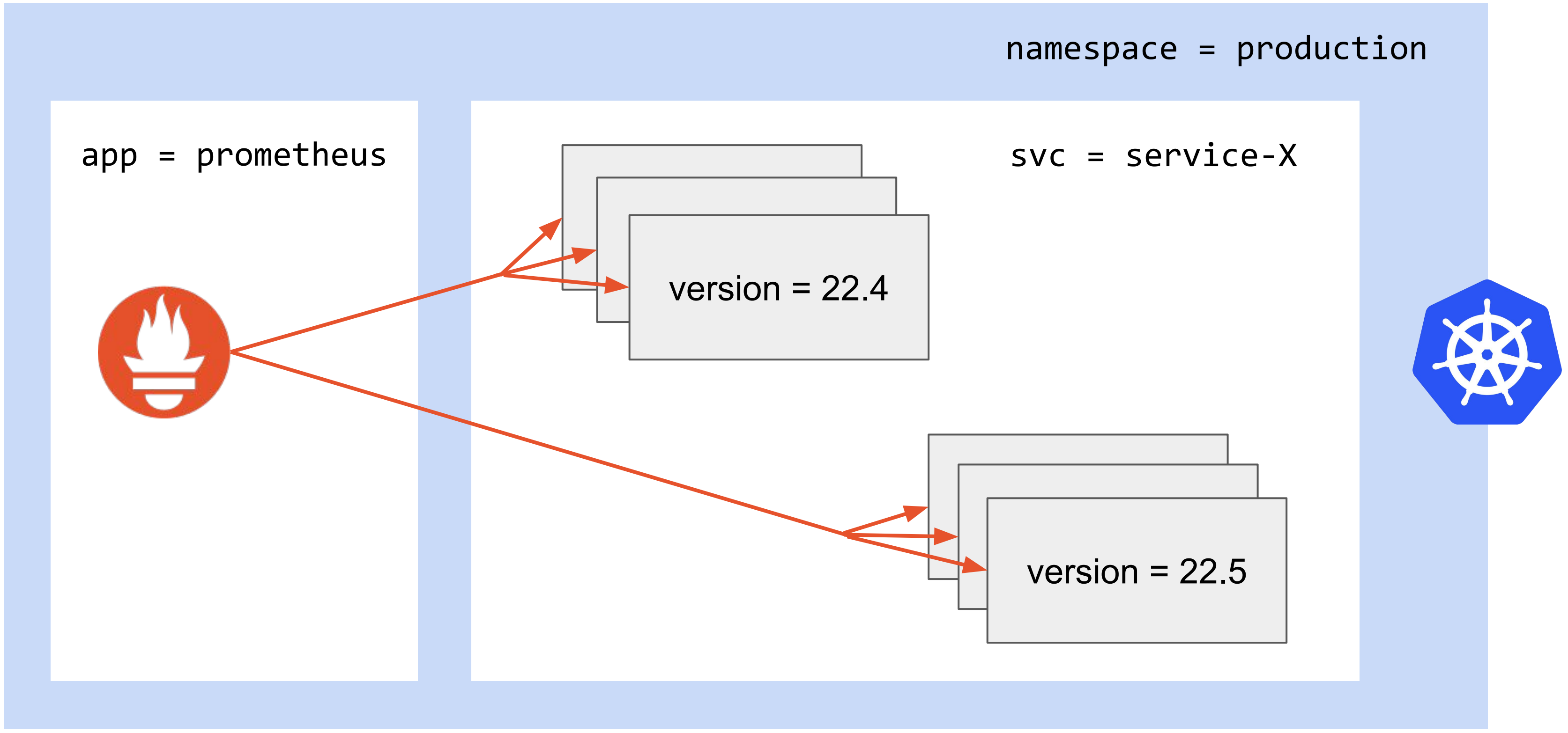
Running Prometheus on Kubernetes

- So far: Prometheus ran outside of cluster
  - Pod IPs must be routable
  - Conventional deployment (Chef, Puppet, …)
  - Service discovery needs authentication

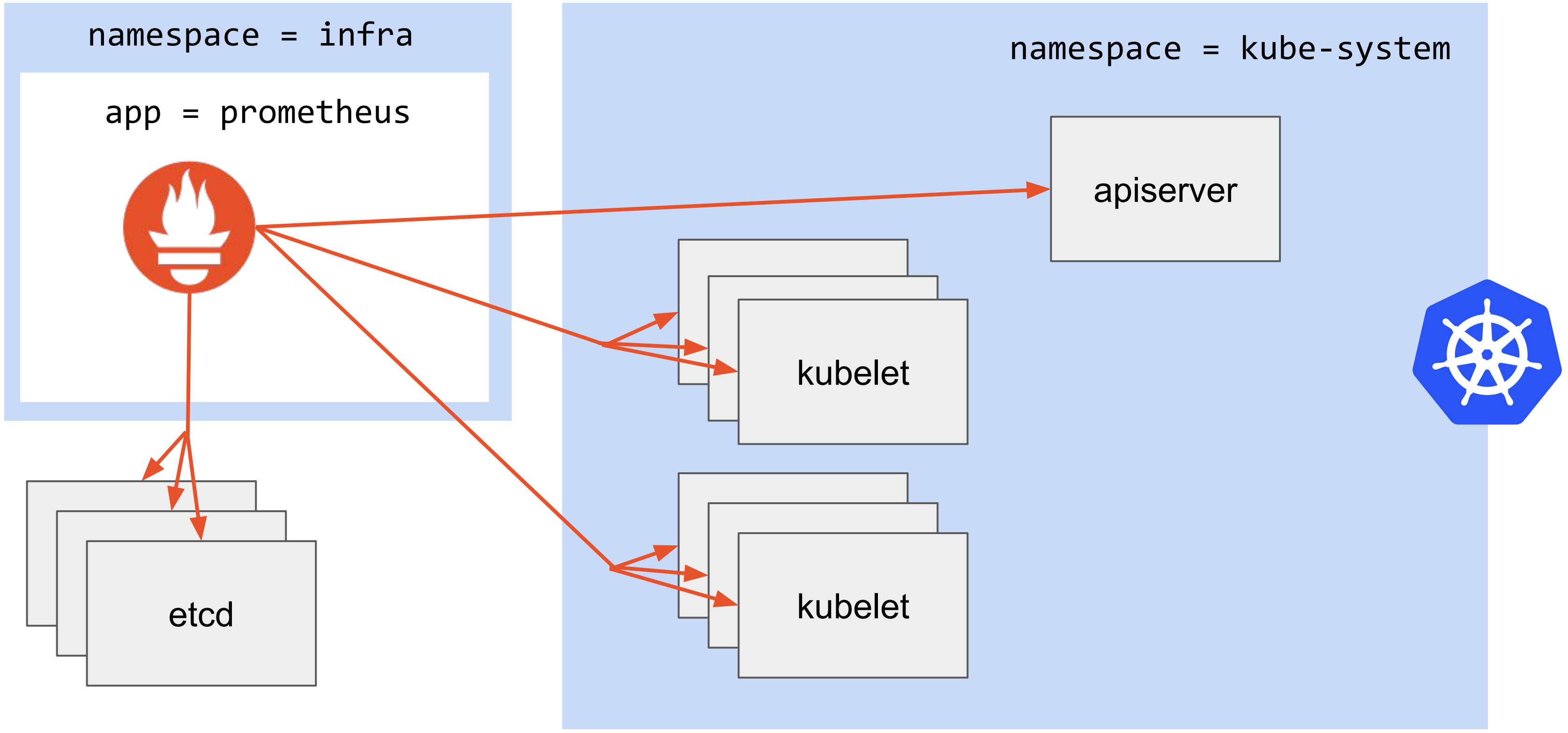
- To run Prometheus inside of cluster:

```
kubectl run --image="quay.io/prometheus/prometheus:0.18.0" prometheus
```

Monitoring Services



Monitoring Kubernetes



What about storage?

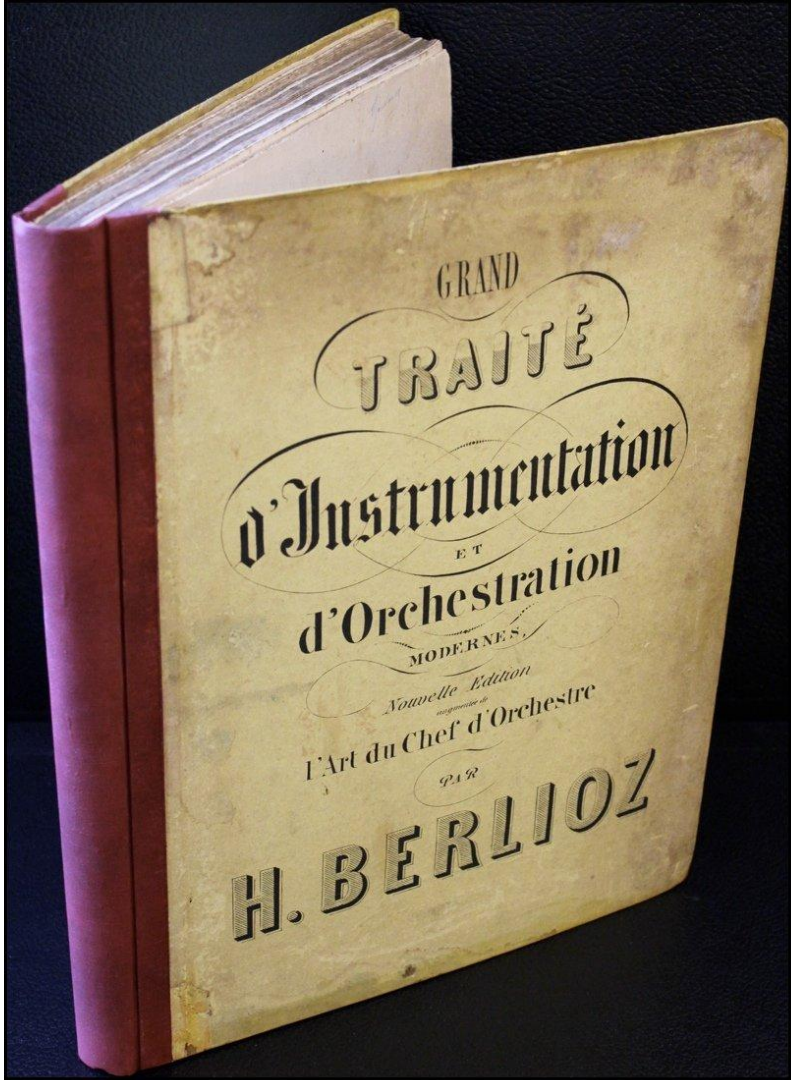
A) None

B) Network/Cloud volumes

C) Host volumes

DEMO

The end


GRAND
TRAITÉ
d'Instrumentation
ET
d'Orchestration
MODERNES,
Nouvelle Edition
augmentée de
l'Art du Chef d'Orchestre
PAR
H. BERLIOZ