KubeCon | CloudNativeCon

Europe 2018

Kubervisor: Pod Anomaly Detection

David Benque, Amadeus S.A.S

Cedric Lamoriniere, Amadeus S.A.S

Who are we?

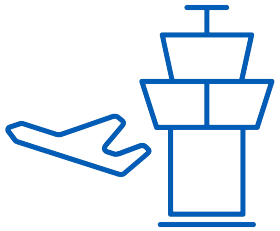
David Benque
@BenqueDavid
Software Engineer at Amadeus

CNCF Meetup Organiser
Distributed systems
PaaS
Automation
Gopher

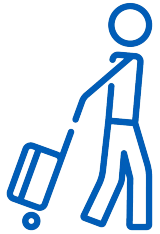
Cedric Lamoriniere
@cedriclam
Software Engineer at Amadeus

CNCF Meetup Organiser
Distributed systems
Gopher

Amadeus



~600 million total bookings processed in 2016



1.3 billion passengers boarded in 2016

~450 000 queries per second (600 000 at peak)

Agenda


KubeCon | CloudNativeCon
Europe 2018

1. Distributed Systems and Stability

2. Solutions to increase reliability
   - ➔ Kubernetes integrated solution
   - ➔ Addons solution

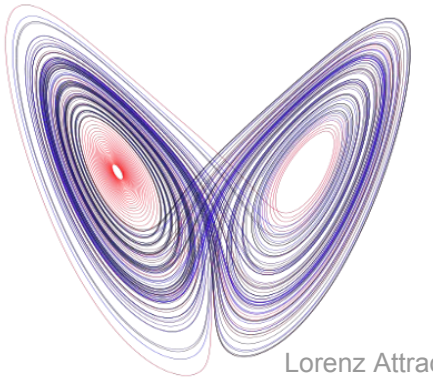
3. Kubervisor: a pod anomaly detection solution
   - ➔ Architecture
   - ➔ Demo

1

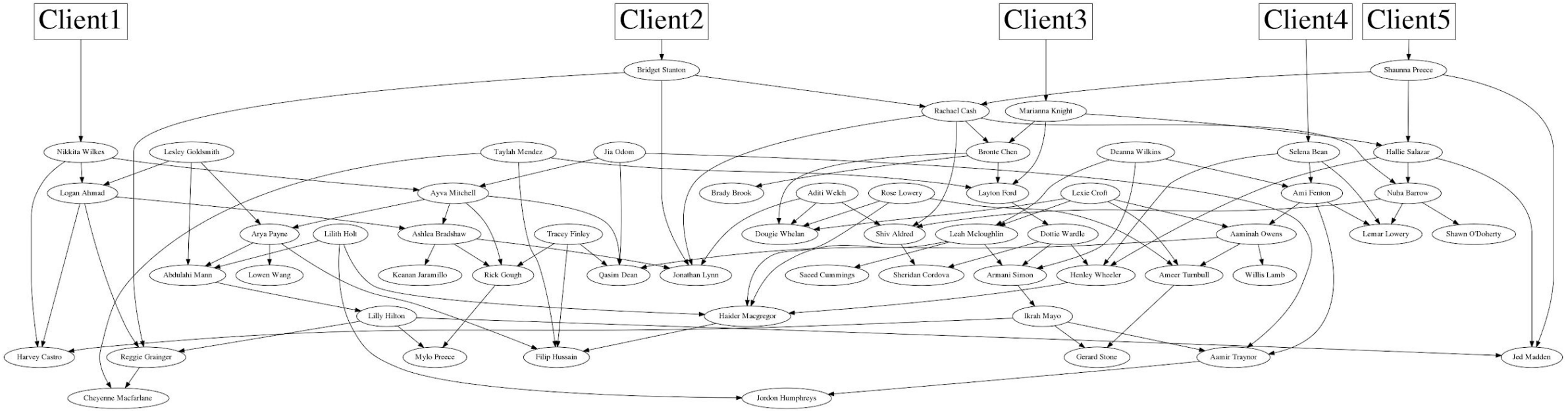
Distributed Systems and Stability

Distributed systems and stability

Butterfly effect in distributed systems

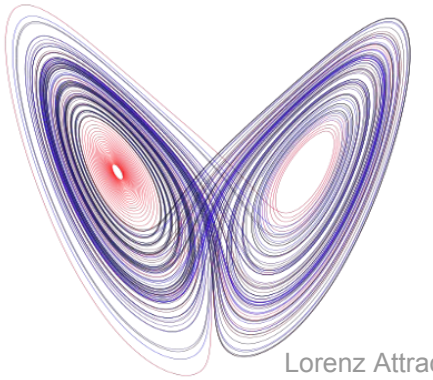


Lorenz Attractor

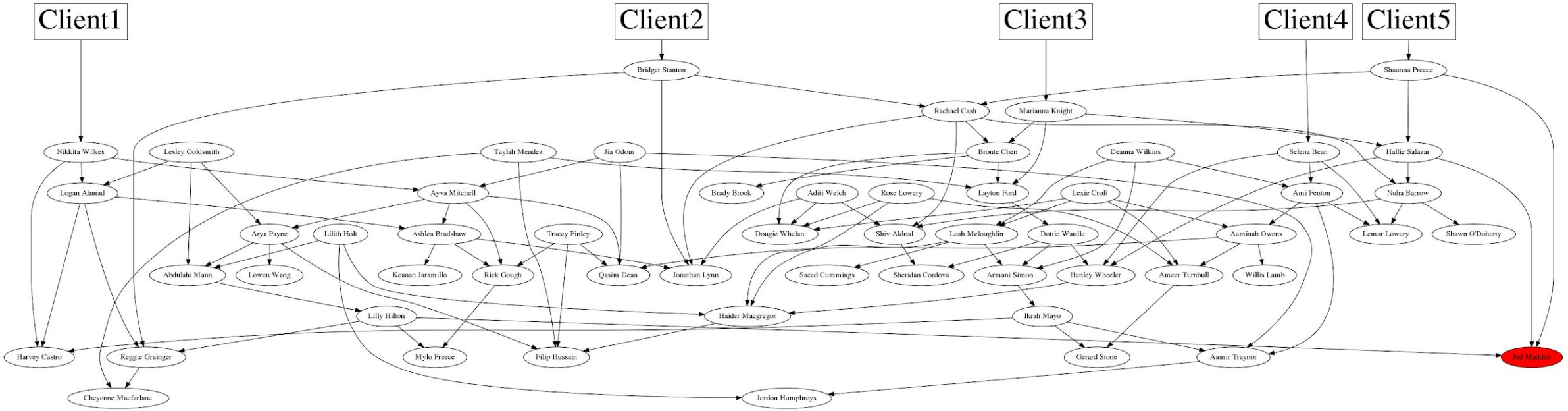


Distributed systems and stability

Butterfly effect in distributed systems

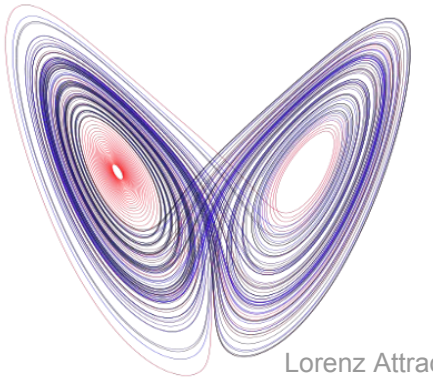


Lorenz Attractor

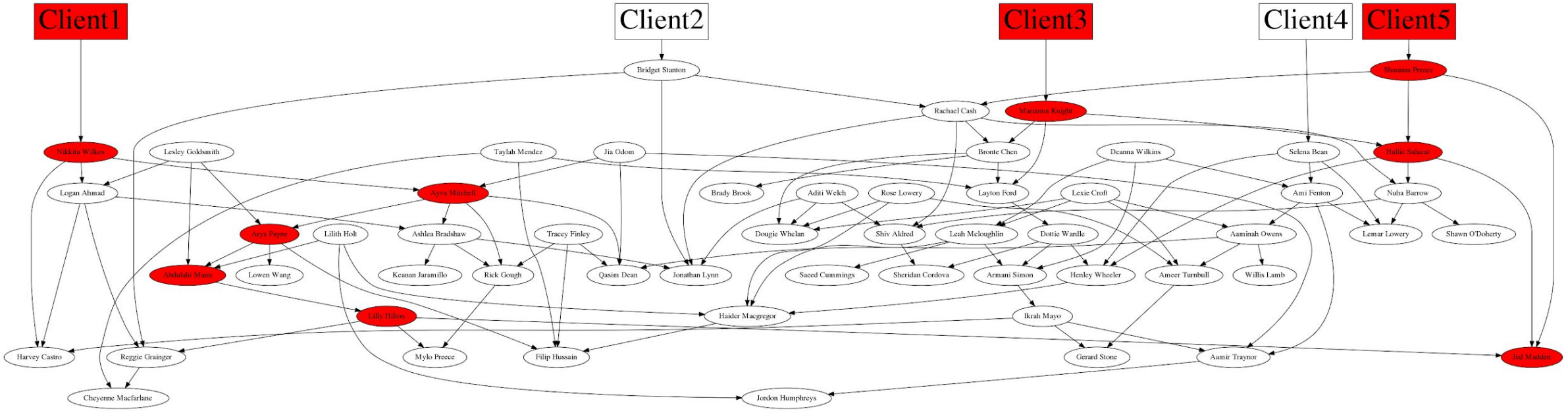


Distributed systems and stability

Butterfly effect in distributed systems

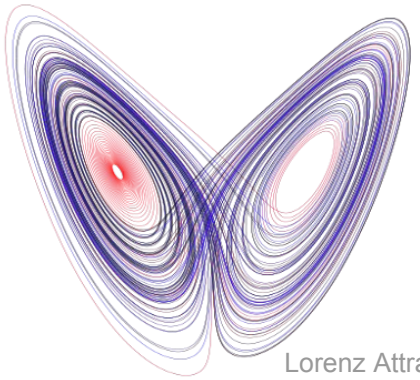


Lorenz Attractor



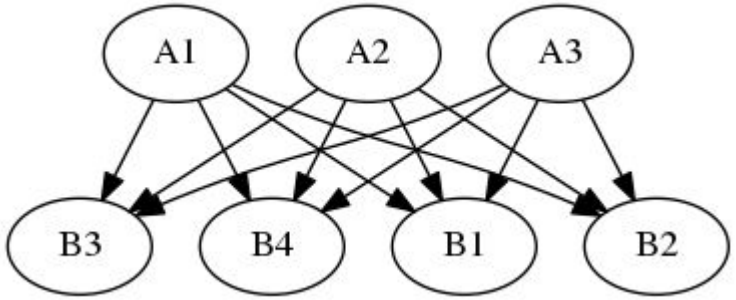
Distributed systems and stability

Load Balancing and multiple requests to a failing dependency could even make things worse

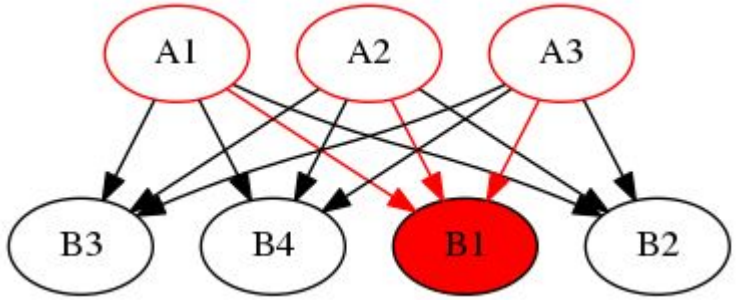


Lorenz Attractor

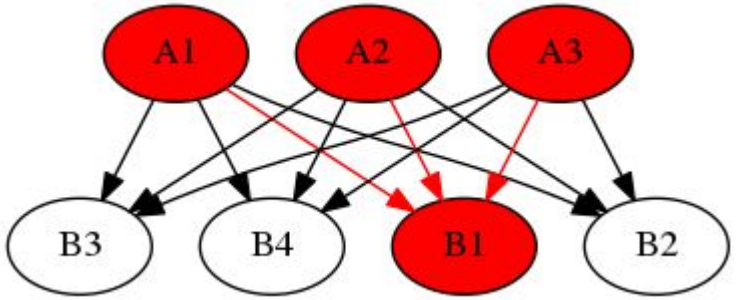
- ➔ All Fine



- ➔ B1 Instance fails
- ➔ Ai instances failure rate 25%



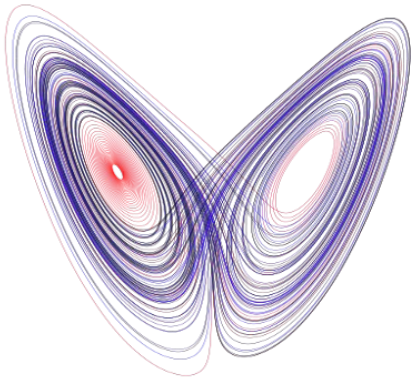
- ➔ B1 Instance fails
- ➔ Ai makes parallel calls to Bj
- ➔ Ai instances failure rate >25%



Distributed systems and stability

“A distributed system is one that prevents you from working
because of the failure of a machine that you had never heard of.”

Leslie Lamport



Attractor
Edward Lorenz

“We live in a rainbow of chaos.”

Paul Cezanne

2

Solutions to increase reliability

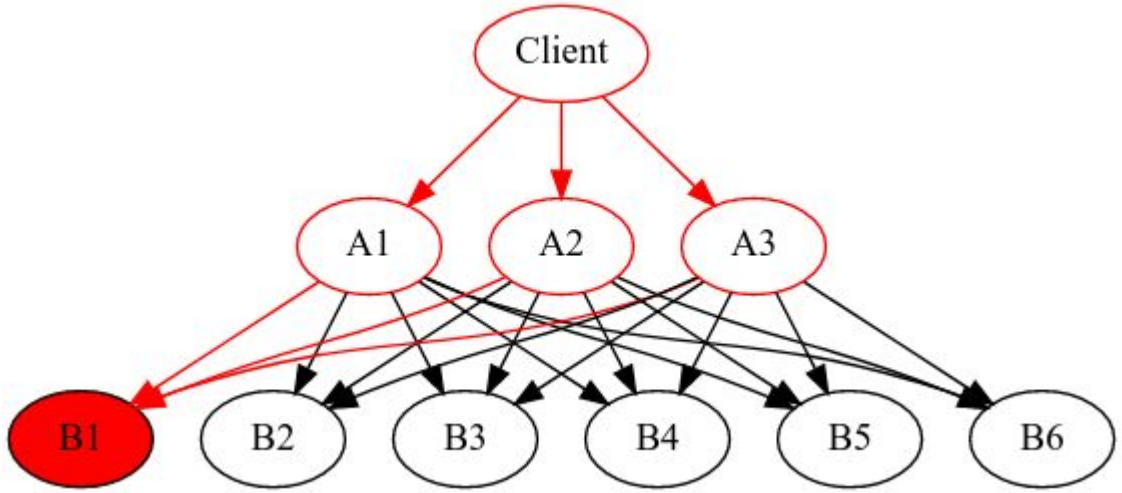
Solutions to increase reliability

Proximity-based Load Balancing

Assumption: loadbalancing using round robin

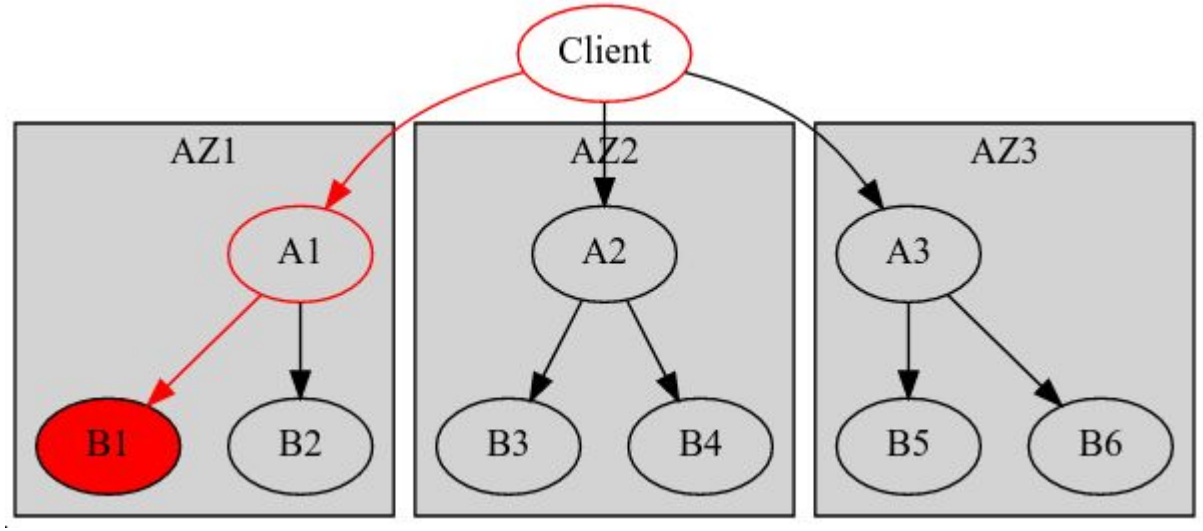
Client->A->B: failure rate ⅙

Client->A-(x6)->B: failure rate 1



Client->A->B: failure rate ⅙

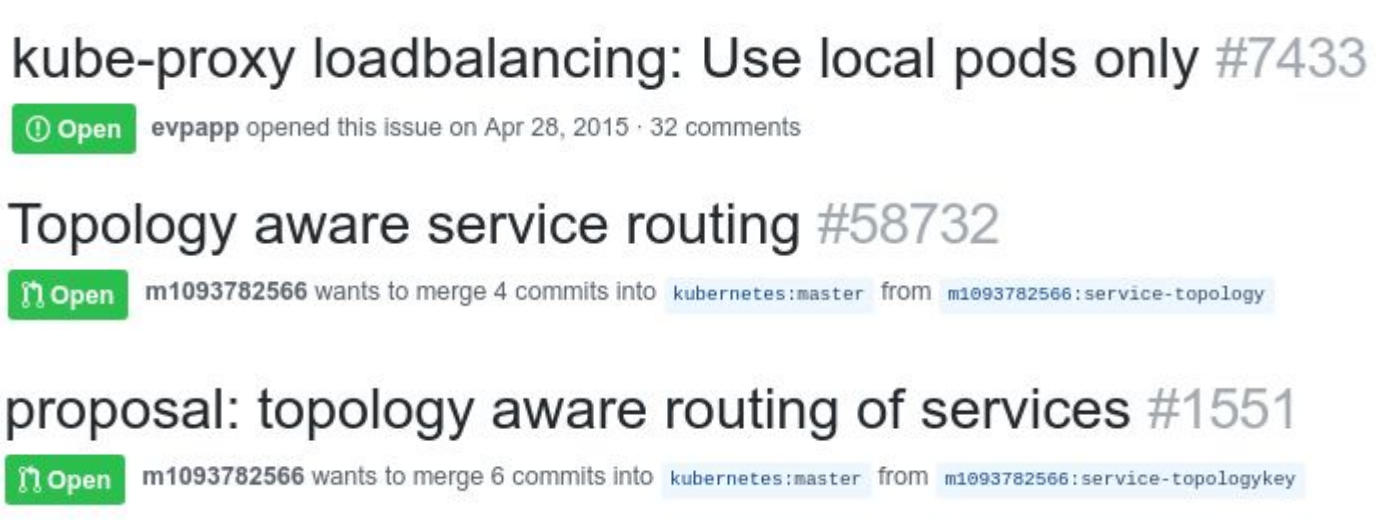
Client->A-(x6)->B: failure rate ⅓



- Ease root cause analysis
- Safeguards overall success rate by constraining fan out

Solutions to increase reliability

Proximity-based Load Balancing



kubernetes/kubernetes

kubernetes/community

Istio

pilot-agent proxy (envoy) is Availability Zone aware:

`--availabilityZone <string> :` The availability zone where this Envoy instance is running. When running Envoy as a sidecar in Kubernetes, this flag must be one of the availability zones assigned to a node using failure-domain.beta.kubernetes.io/zone annotation.

Solutions to increase reliability

Container Termination

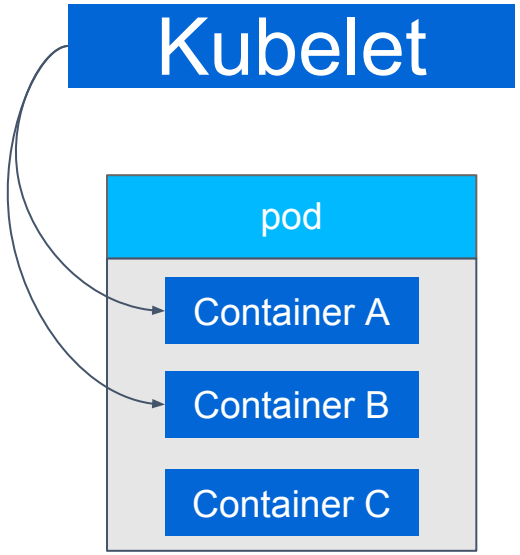
“ A Container can exceed its memory request if the Node has memory available. But a Container is **not allowed to use more than its memory limit.** If a Container allocates more memory than its limit, the Container becomes a candidate for termination.”

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: memory-demo-2
  namespace: mem-example
spec:
  containers:
  - name: memory-demo-2-ctr
    image: memtest
    resources:
      requests:
        memory: "50Mi"
      limits:
        memory: "100Mi"
```

https://kubernetes.io/docs/concepts/configuration/manage-compute-resources-container/

Solutions to increase reliability

Probes



Probe Implementation:

- tcpSocket
- httpGet
- exec

```yaml
apiVersion: v1
kind: Pod
…...
spec:
  containers:
  - name: A
    … ...
    readinessProbe:
      tcpSocket:
        port: 8080
      initialDelaySeconds: 5
    livenessProbe:
      httpGet:
        path: /healthz
        port: 8080
      periodSeconds: 20
  - name: B
    … ...
    readinessProbe:
      exec:
        command:
        - myscript
    … ...
```

Liveness → kill container
Readiness → mark endpoint as not ready

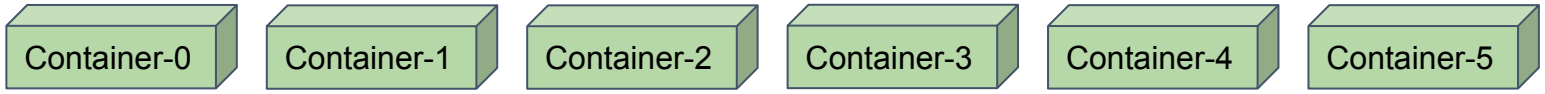
```
// Number of seconds after the container has started before
liveness probes are initiated.
InitialDelaySeconds int32

// Number of seconds after which the probe times out. (default 1)
TimeoutSeconds int32

// How often (in seconds) to perform the probe. (default 10)
PeriodSeconds int32

// Minimum consecutive successes for the probe to be considered
successful after having failed.(default 1)
SuccessThreshold int32

// Minimum consecutive failures for the probe to be considered
failed after having succeeded. (default 3)
FailureThreshold int32
```

https://kubernetes.io/docs/tasks/configure-pod-container/configure-liveness-readiness-probes/

Solutions to increase reliability

Probes

- Keep it simple.

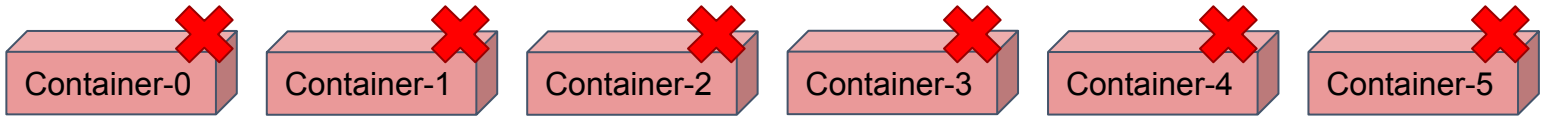


  Complexity → Bugs

Solutions to increase reliability

Probes

- Keep it simple.



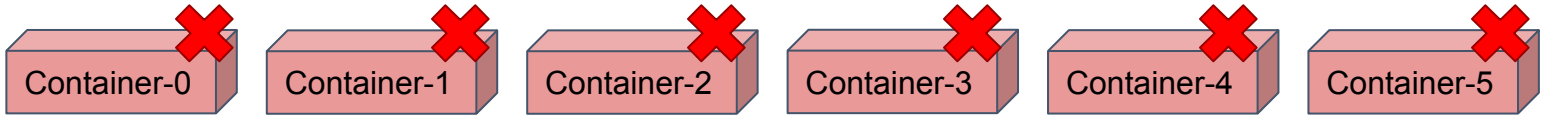
Complexity → Bugs → Containers collective suicide

→ Service Outage

Solutions to increase reliability

Probes

- Keep it simple.



Complexity → Bugs → Containers collective suicide

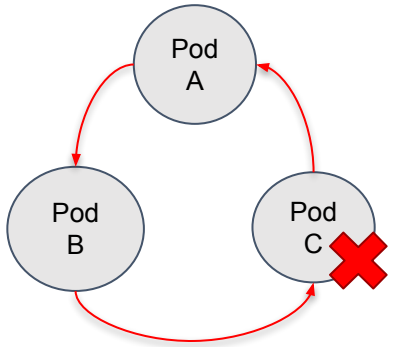
→ Service Outage

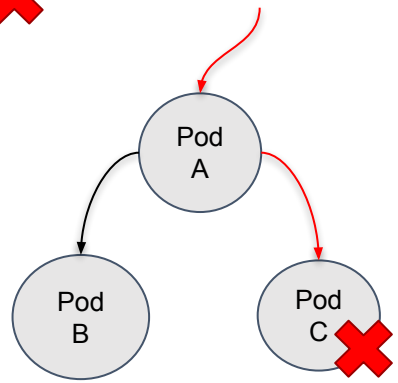
- Don't check external dependency chain

→ May not be able to restart from scratch

→ Difficult to find problem root cause

→ Failure propagation cutting valid branches



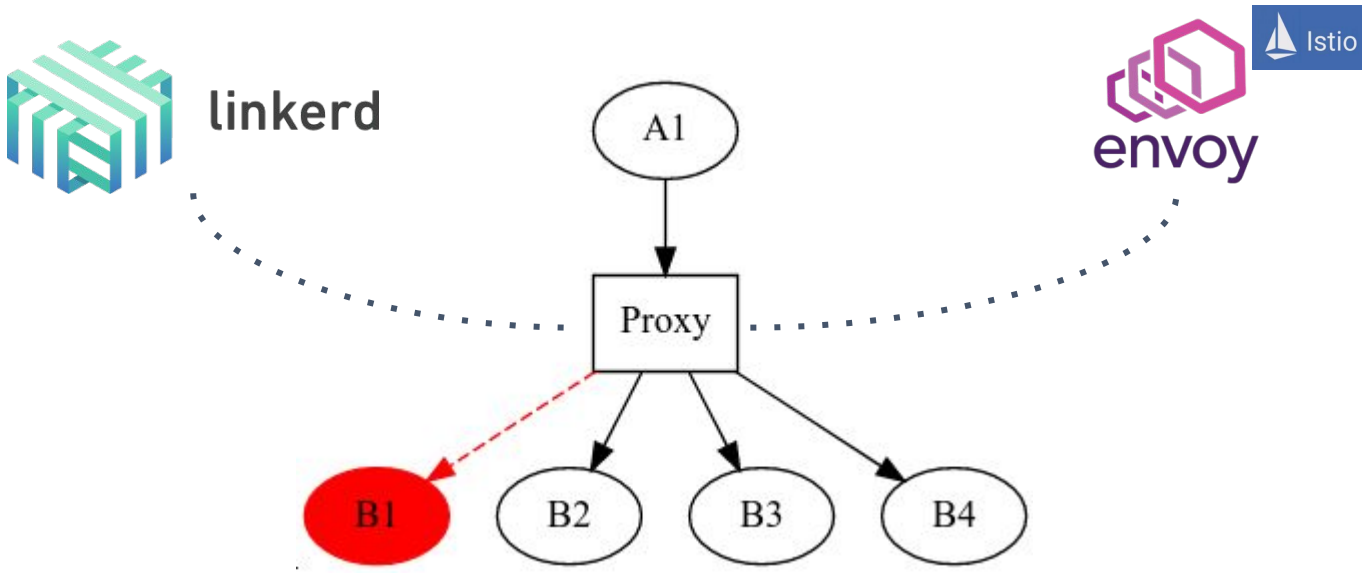


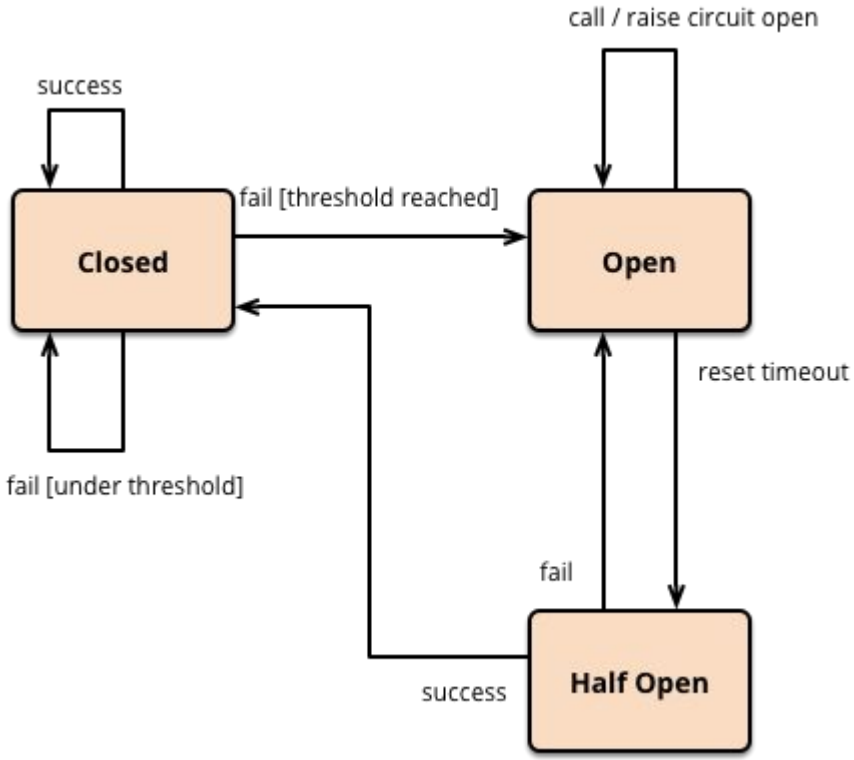
Solutions to increase reliability

Circuit Breaker

https://martinfowler.com/bliki/CircuitBreaker.html

- The service mesh proxy implements the circuit breaker

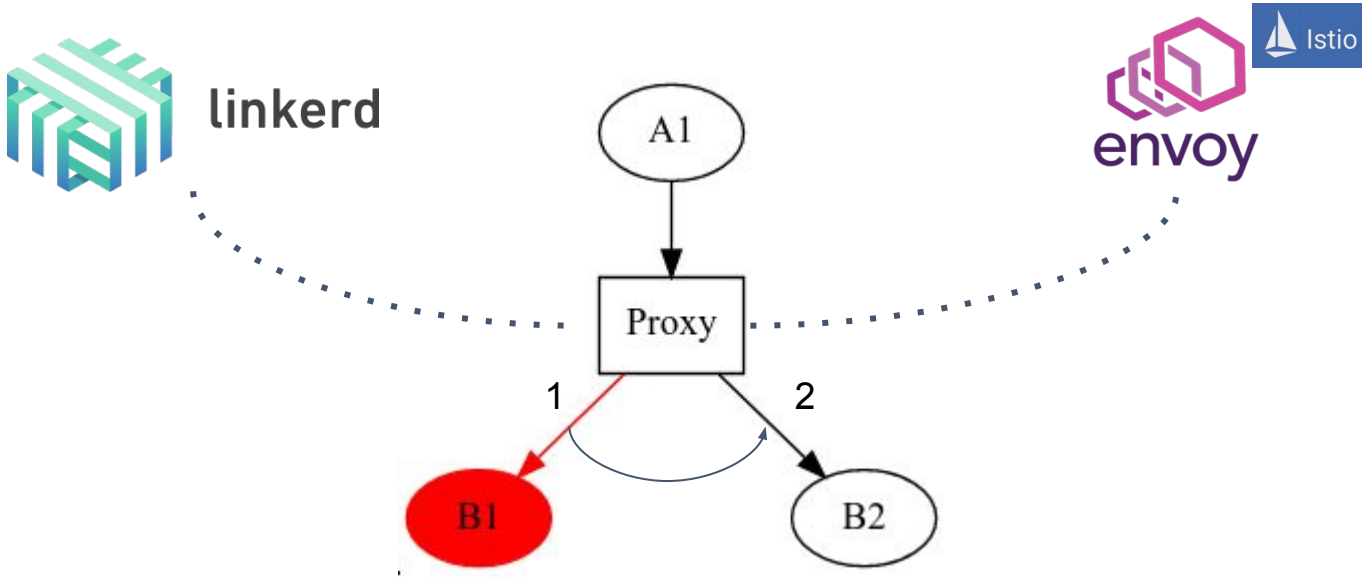


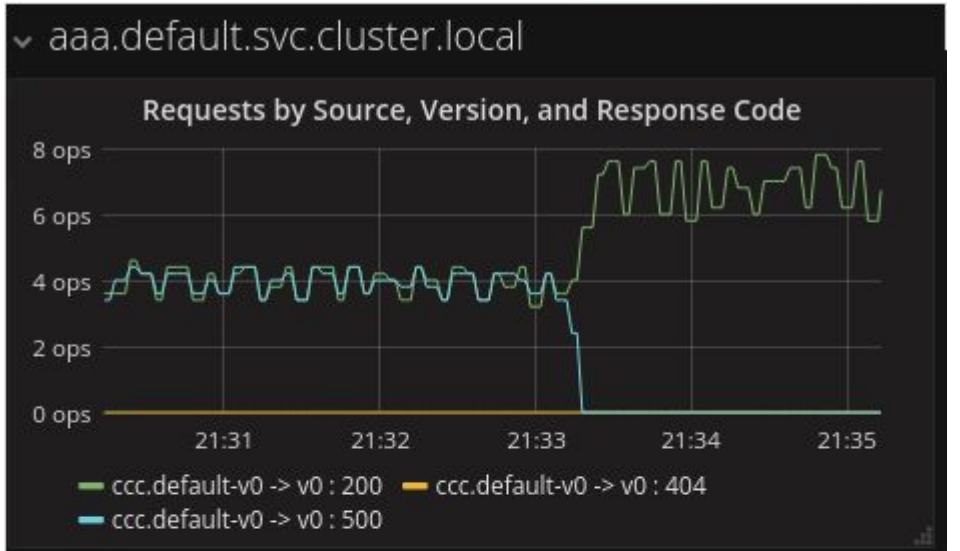


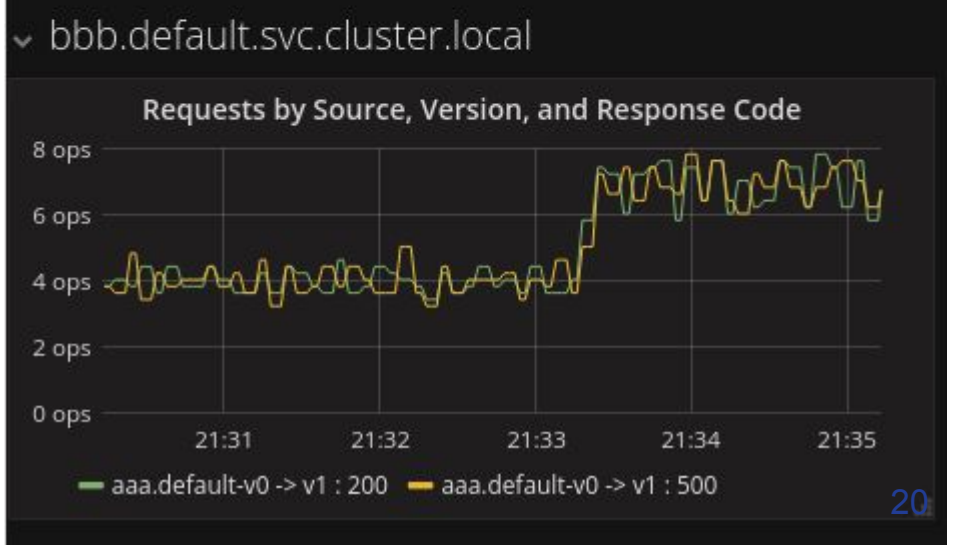
Solutions to increase reliability

Retries

- The service mesh proxy implements retry policy







Solutions to increase reliability

Limitations

- Local decision
  - Probes: in each container
  - Service Mesh: in each proxy
- Based on technical signals only
  - Memory consumption
  - Connection
  - Response time
  - Return Code
  - Circuit Breaker working with HTTP only

3

Kubervisor: a pod anomaly detection solution

Kubervisor

A pod anomaly detection solution

- Orchestrated decision
  - decision at service level
  - avoid collective suicide, service outage
- Decision based on technical or/and business information
  - based on metrics
  - currently supports Prometheus with PromQL.
- Kubervisor controller:
  - divided in 2 components: Breaker, Activator
  - differents strategies possible for the Breaker and Activator
  - configuration in dedicated CRD: KubervisorService

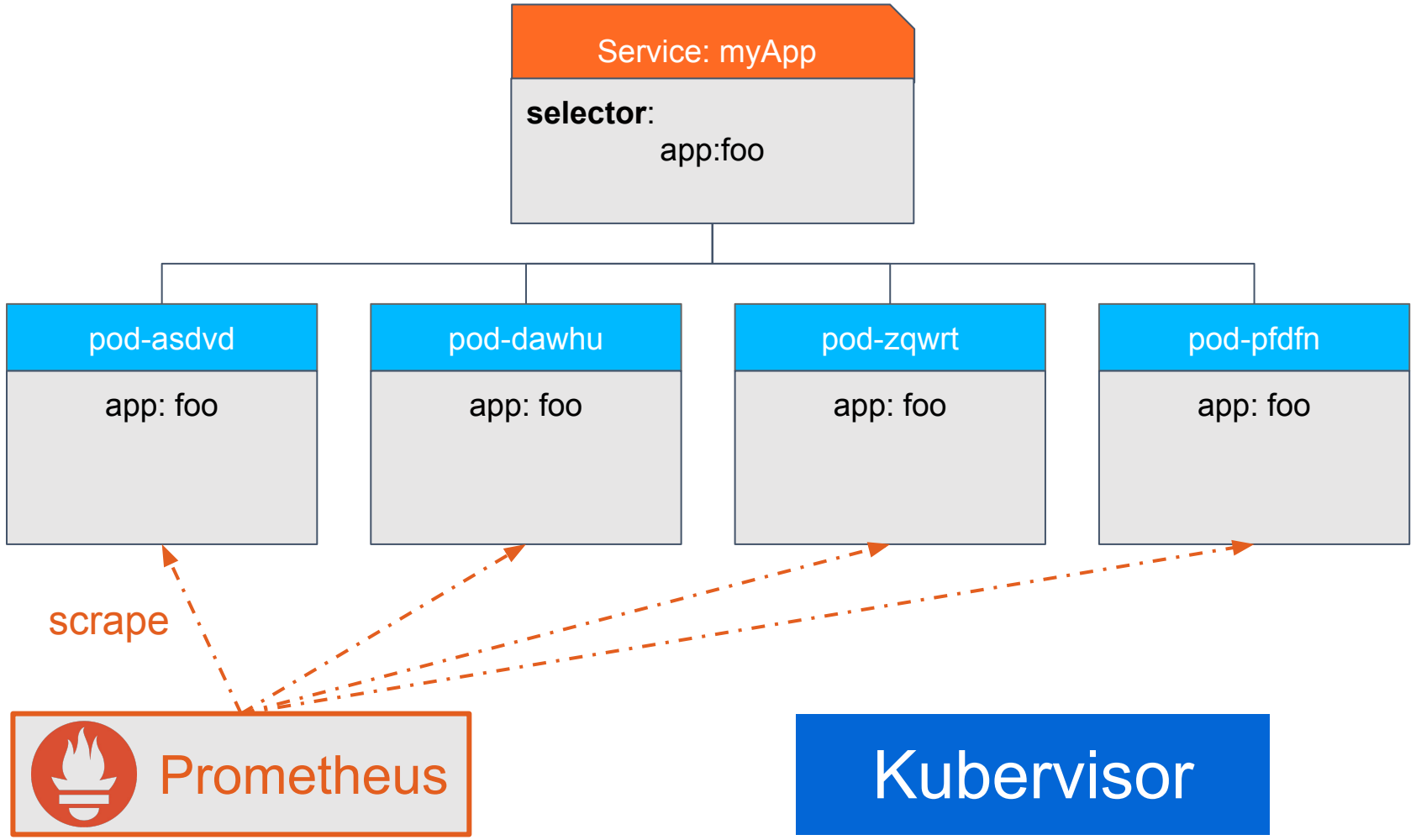
Kubervisor

A pod anomaly detection solution

- KubervisorService CRD
- spec
  - service: application service to watch
  - breaker: breaker configuration
    - global settings
    - differents implementations
  - activator: activator configuration
    - mode: periodic, retryAndPause, retryAndKill
    - period: pause duration

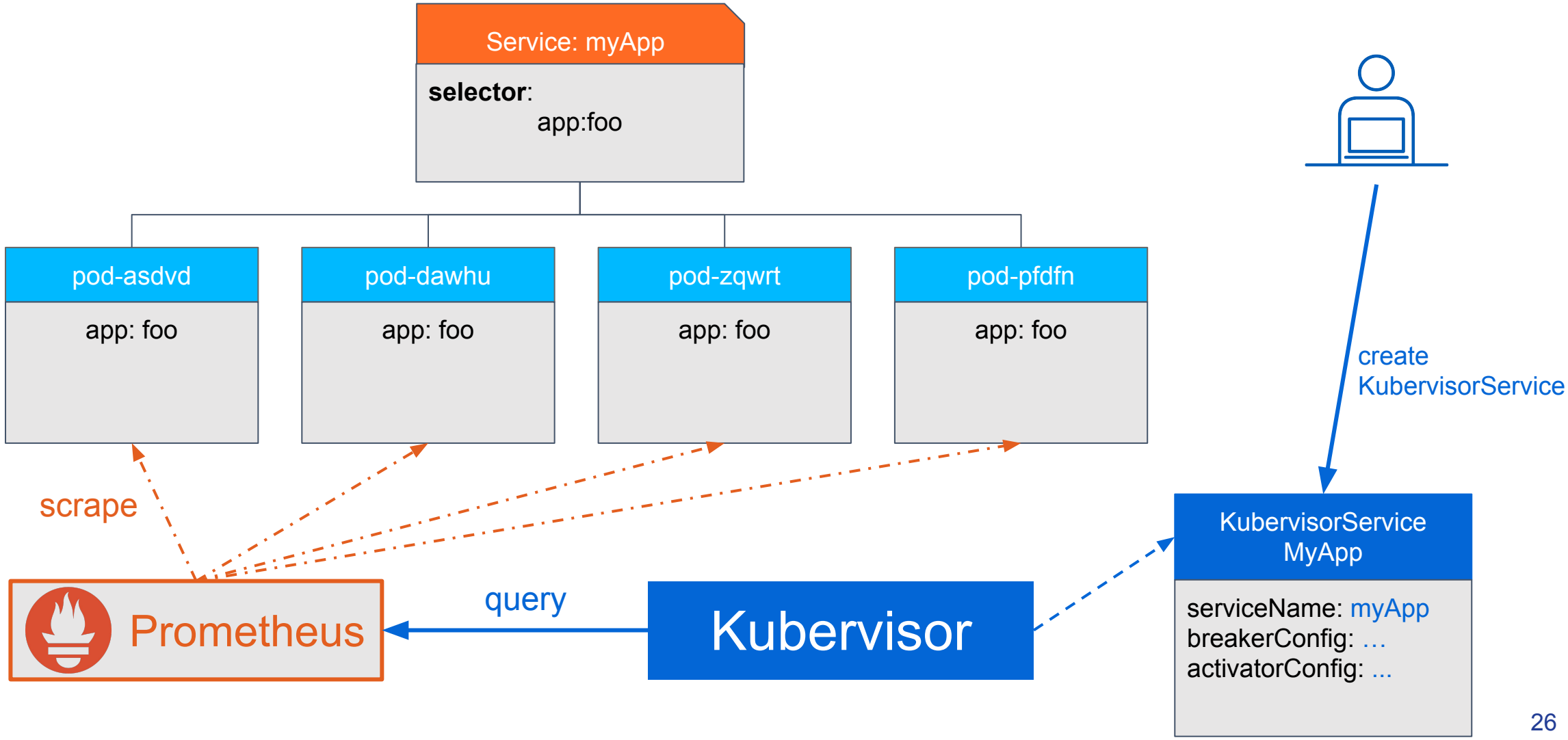
Kubervisor workflow

how to use it



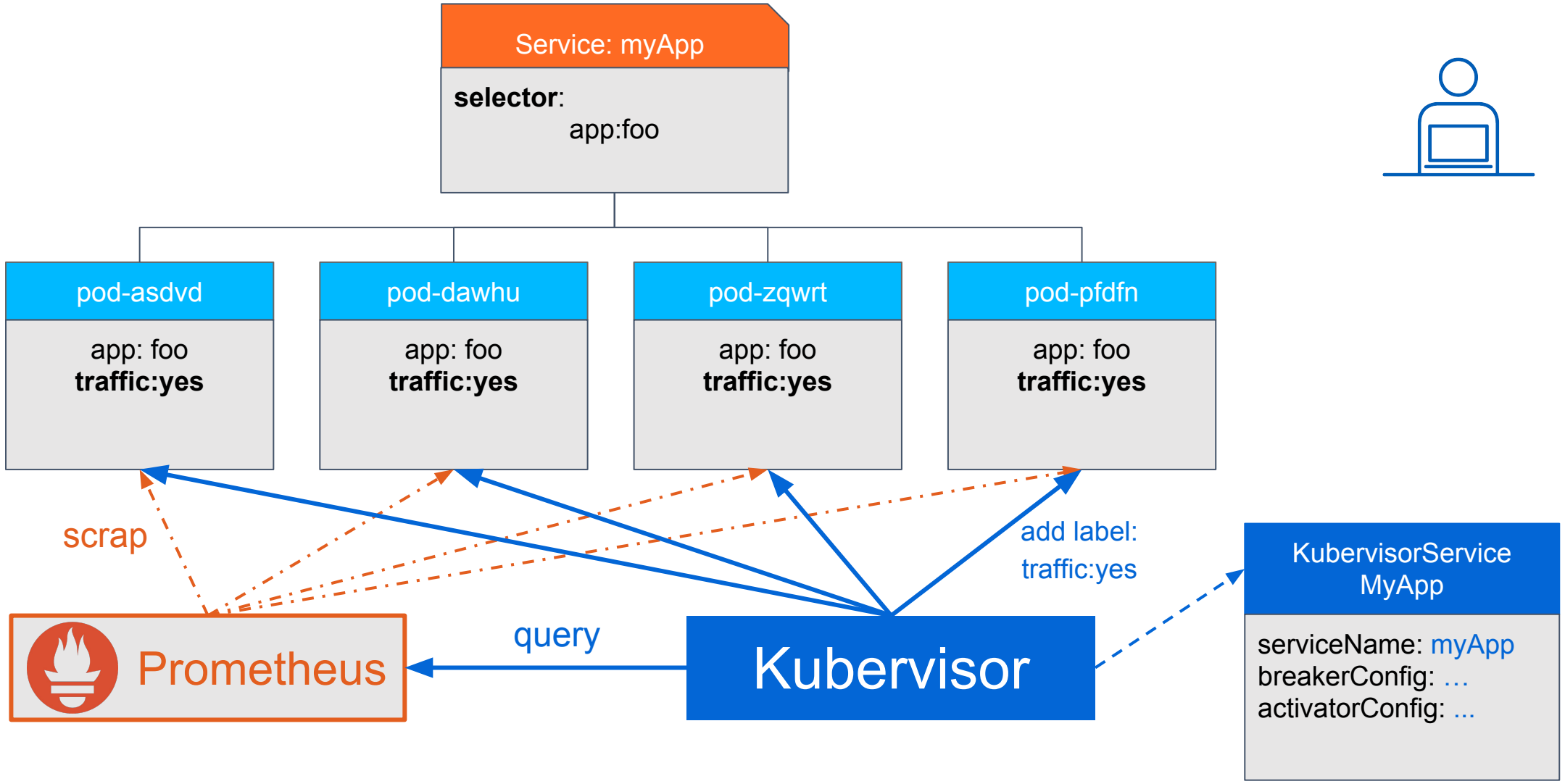
Kubervisor workflow

Initialisation



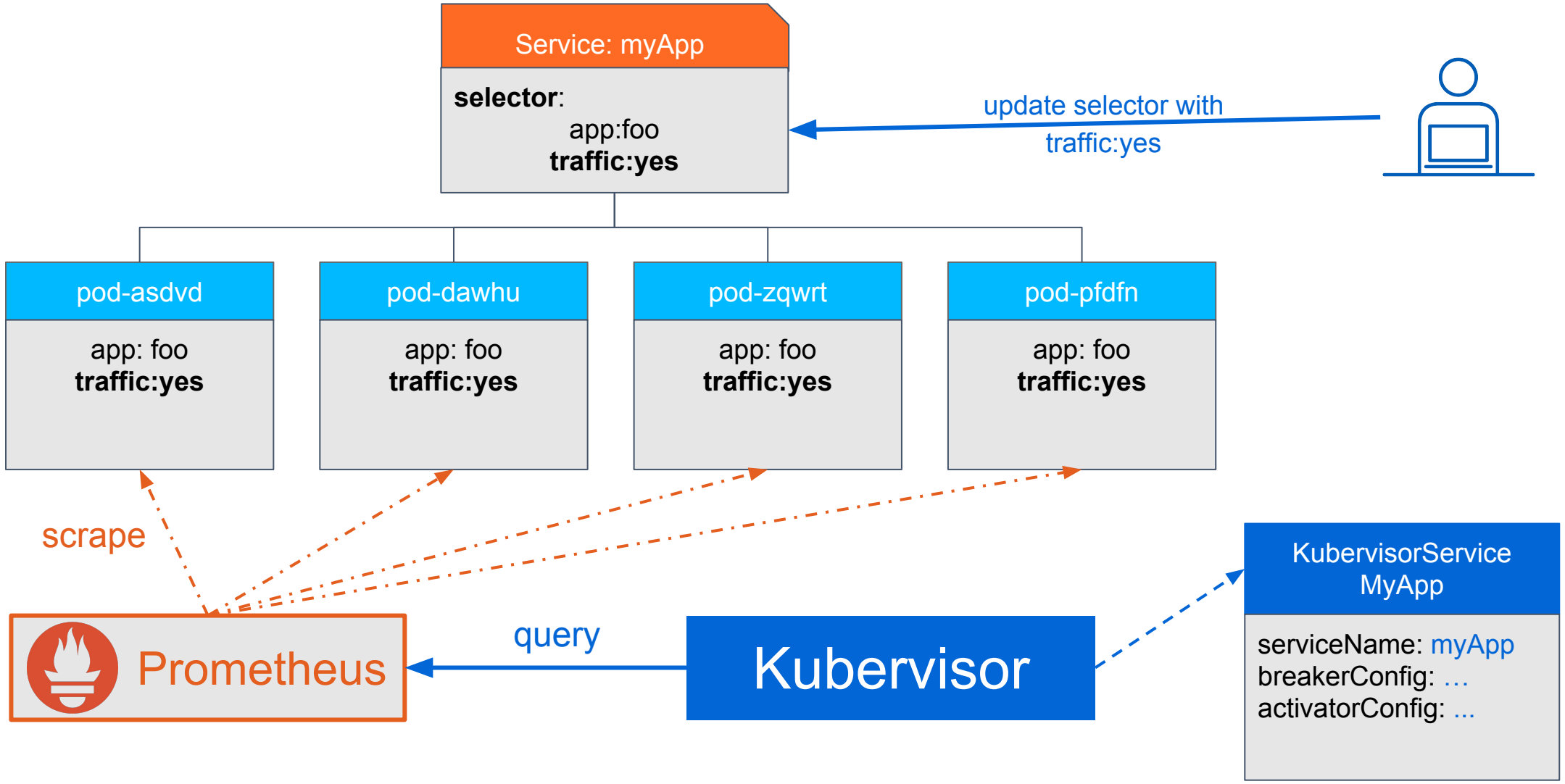
Kubervisor workflow

Initialisation



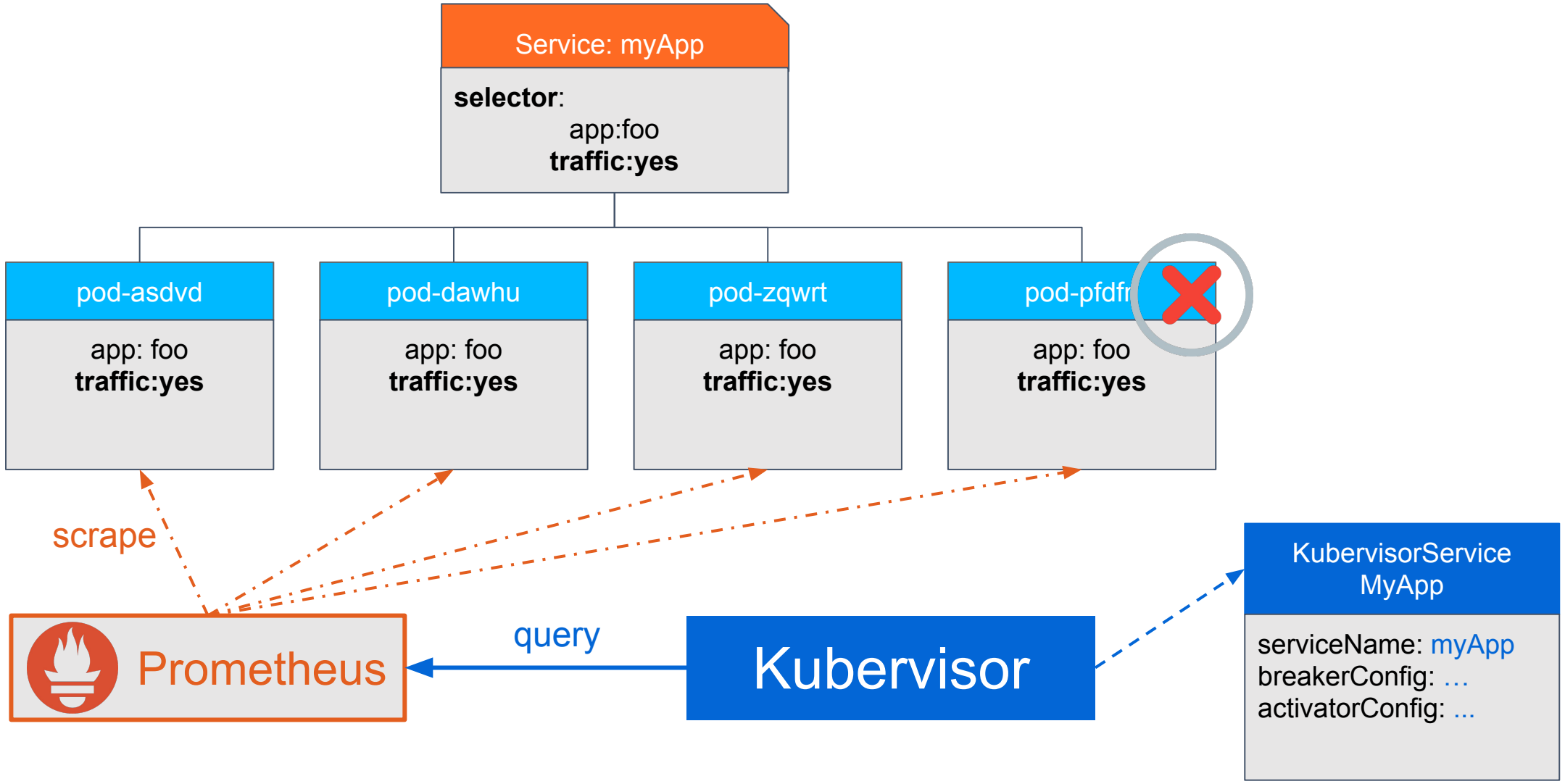
Kubervisor workflow

Initialisation



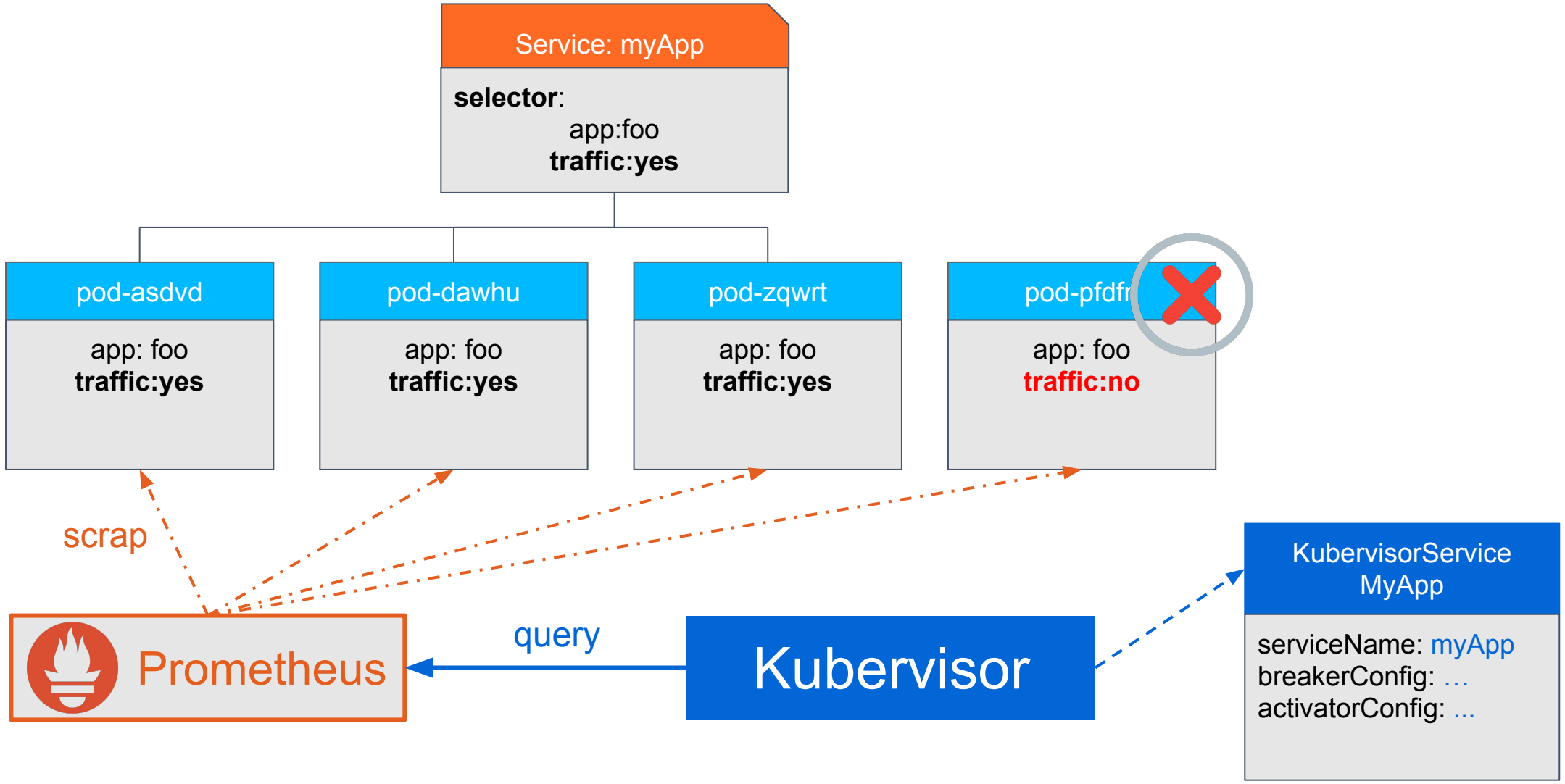
Kubervisor workflow

Anomaly detection



Kubervisor workflow

Breaker activation



Kubervisor

A pod anomaly detection solution

KubeCon

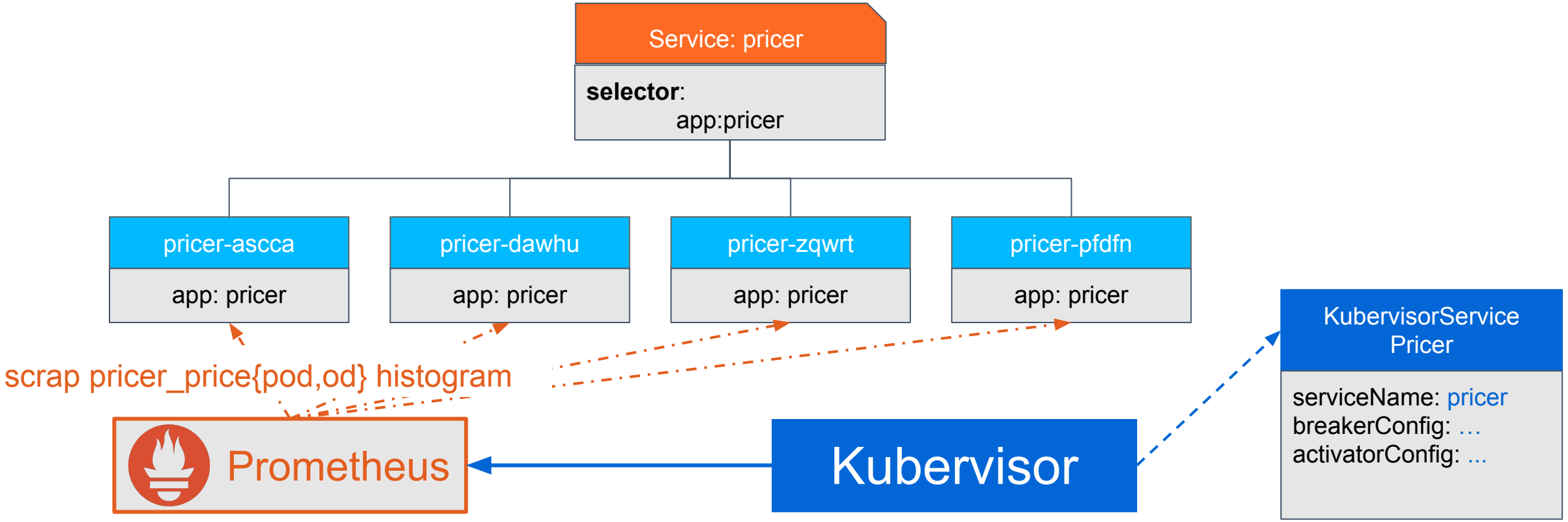
CloudNativeCon

Europe 2018

Demo time!

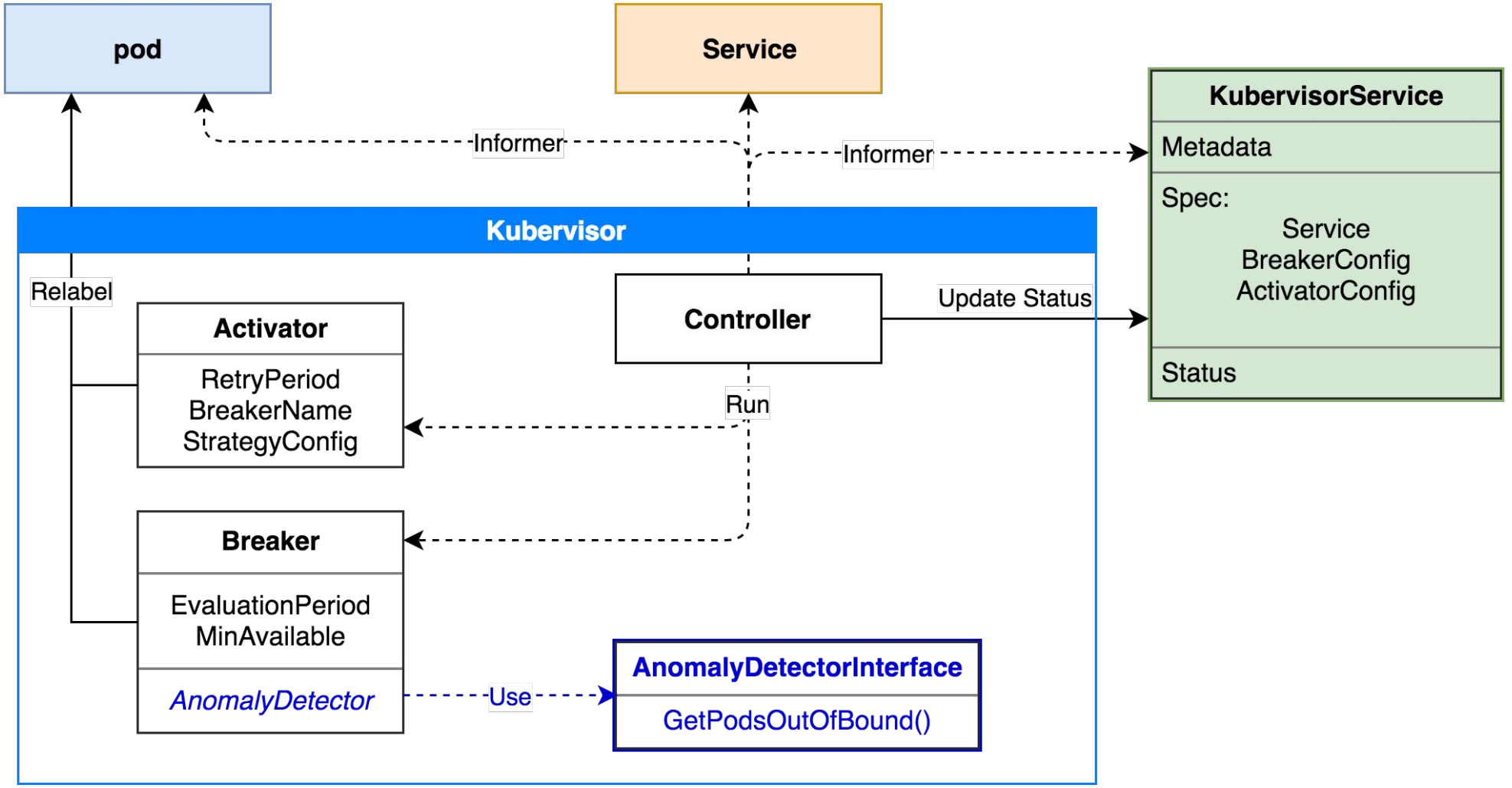
Demo time !

- **Flight price search:** return 3 best prices for an "origin" and "Destination" city on the current day.



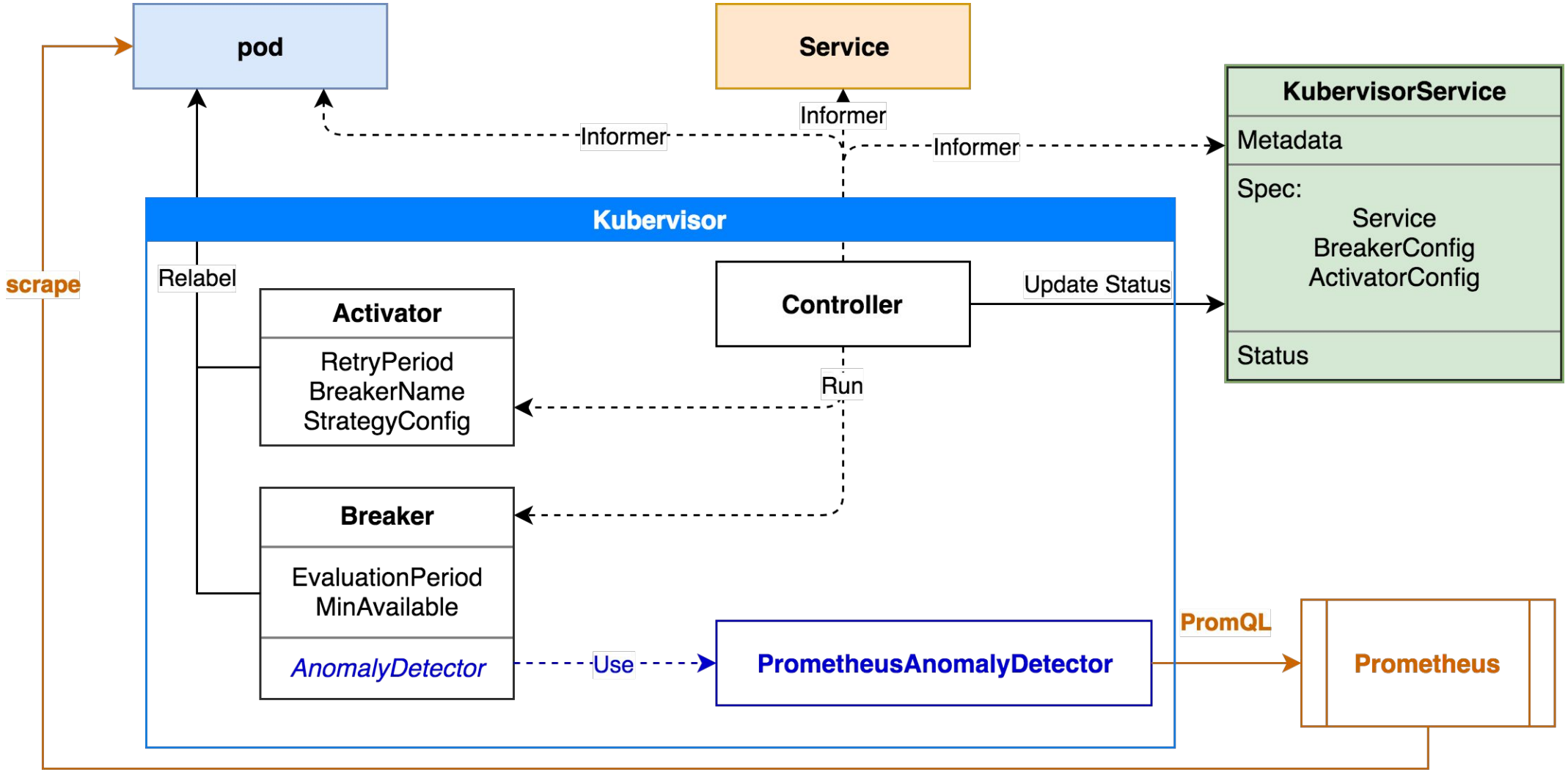
Kubervisor

Internal Architecture



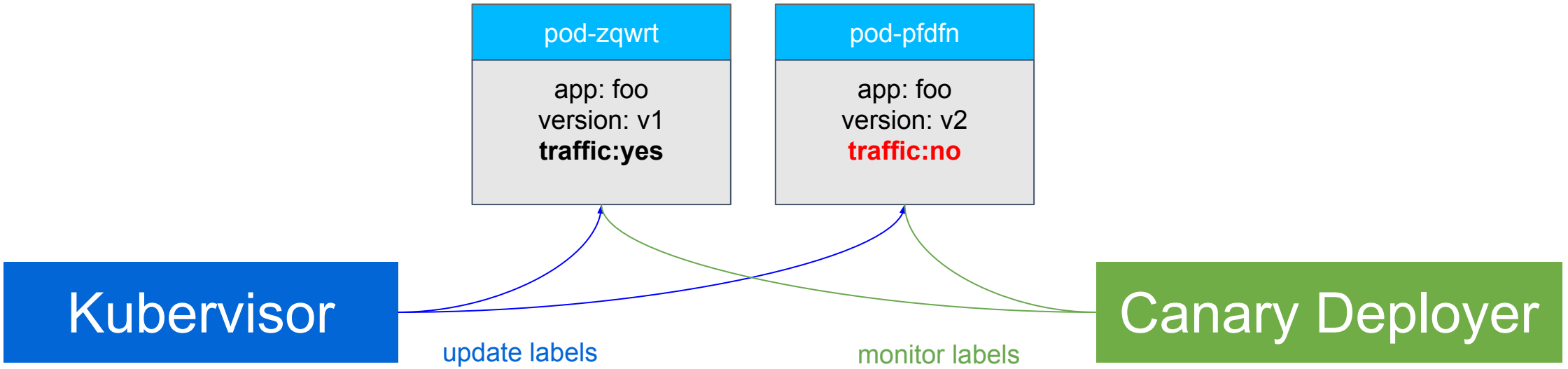
Kubervisor

Internal Architecture



Possible Extension

- Kubervisor labels can be used by other controller
- Example Canary testing deployment



Key takeaways

- Additional solution for improving your services reliability
- Based on standard: Controller with CRD, PromQL (Prometheus)
- Extendable: plugable Breaker and Activator implementation.
- Open source: test it, break it and open us issues

KubeCon

CloudNativeCon

Europe 2018

- github.com/amadeusitgroup/kubervisor
- @cediclam
- @BenqueDavid

Questions?