JFOKUS 2017 - EXPERIENCES FROM USING DISCOVERY SERVICES IN A MICROSERVICE LANDSCAPE

MAGNUS LARSSON

2017-02-07 | CALLISTAENTERPRISE.SE

CALLISTA
— ENTERPRISE —

USE MICROSERVICES WITH OR WITHOUT CONTAINERS?

MICROSERVICES WITHOUT CONTAINERS?

- Can we start without containers and migrate later on?
  - Without require code changes?

WHAT CONTAINER ORCHESTRATION TOOL TO USE?

- Google Kubernetes, Docker Swarm, Amazon ECS…
  - Can we change orchestration tool, without require code changes?

MY PERSONAL QUEST FOR 2016

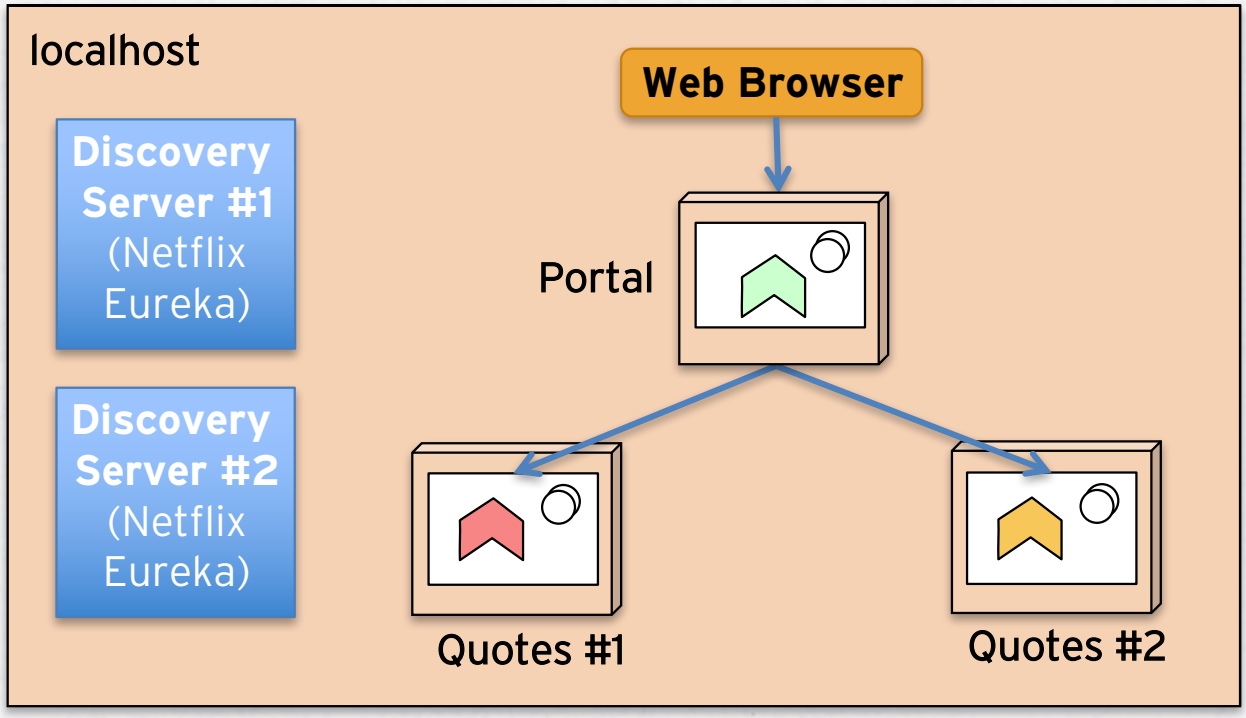
- Write once, deploy "everywhere"?
  - Only require configuration changes!

Main challenge
How to write code that works with the various Service Discovery implementations?

- Selected platforms
  - Without containe ervice Discovery
  - Container orches Discovery):
    - » *Google Kubernetes*
    - » *Docker Swarm mode*
    - » *Amazon ECS – EC2 Container Services*

EXAMPLE #1 - SPRING CLOUD AND NETFLIX OSS (NO CONTAINERS)

Test landscape



SPRING CLOUD

NETFLIX OSS

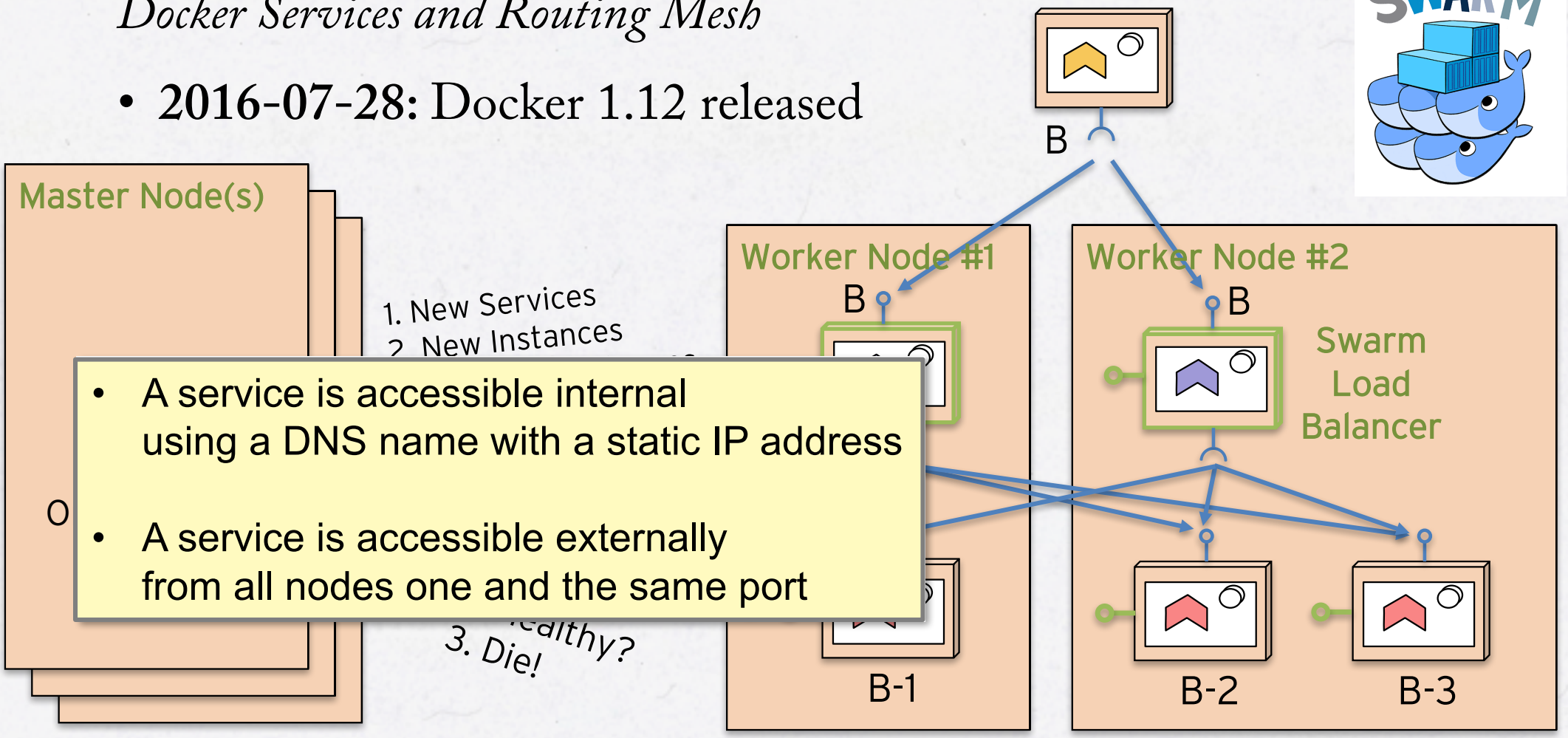
1. Verify load balancing
2. Kill one of the quote service instances…
3. …or a discovery service…
4. Start it again…

CALLISTA — ENTERPRISE —

EXAMPLE #2: DOCKER SWARM MODE

Docker Services and Routing Mesh

- **2016-07-28:** Docker 1.12 released



- A service is accessible internal using a DNS name with a static IP address
- A service is accessible externally from all nodes one and the same port

EXAMPLE #2: DOCKER SWARM MODE

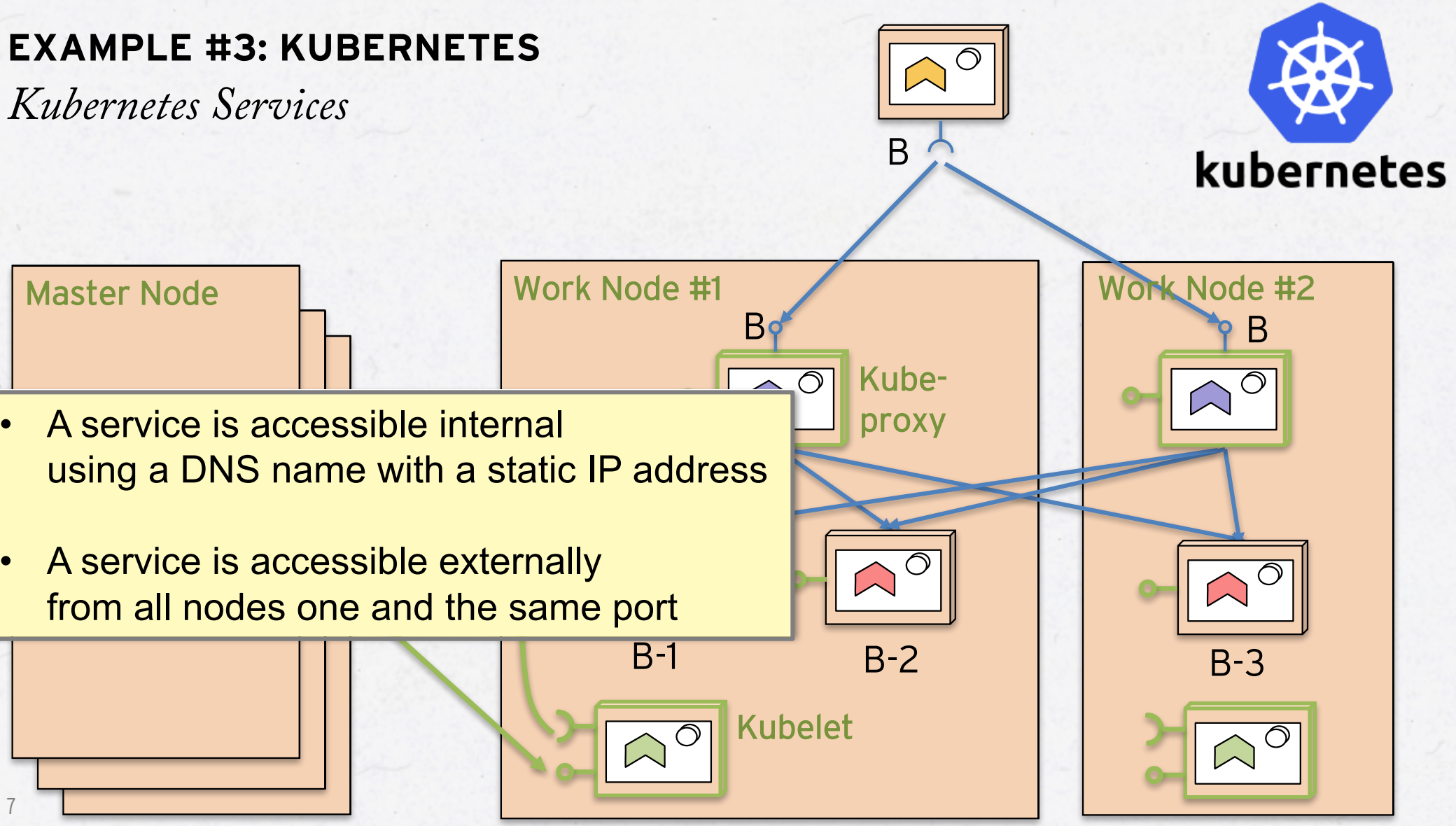
Tests

1. Same tests as for Netflix Eureka

2. Test Resilience
   - Kill quote container
   - Kill node with containers
   - Verify that containers are rescheduled automatically!

EXAMPLE #3: KUBERNETES

Kubernetes Services



- A service is accessible internal using a DNS name with a static IP address
- A service is accessible externally from all nodes one and the same port

EXAMPLE #3: KUBERNETES

Tests

- Setup a Kubernetes Cluster in Google Cloud
  - With auto scaling of both pods (e.g. containers) and nodes
- Same tests as for Docker Swarm
- Test auto scaling!

EXAMPLE #3: KUBERNETES

- Put some load using the portal and wait for a while…
- Any new pods and nodes?

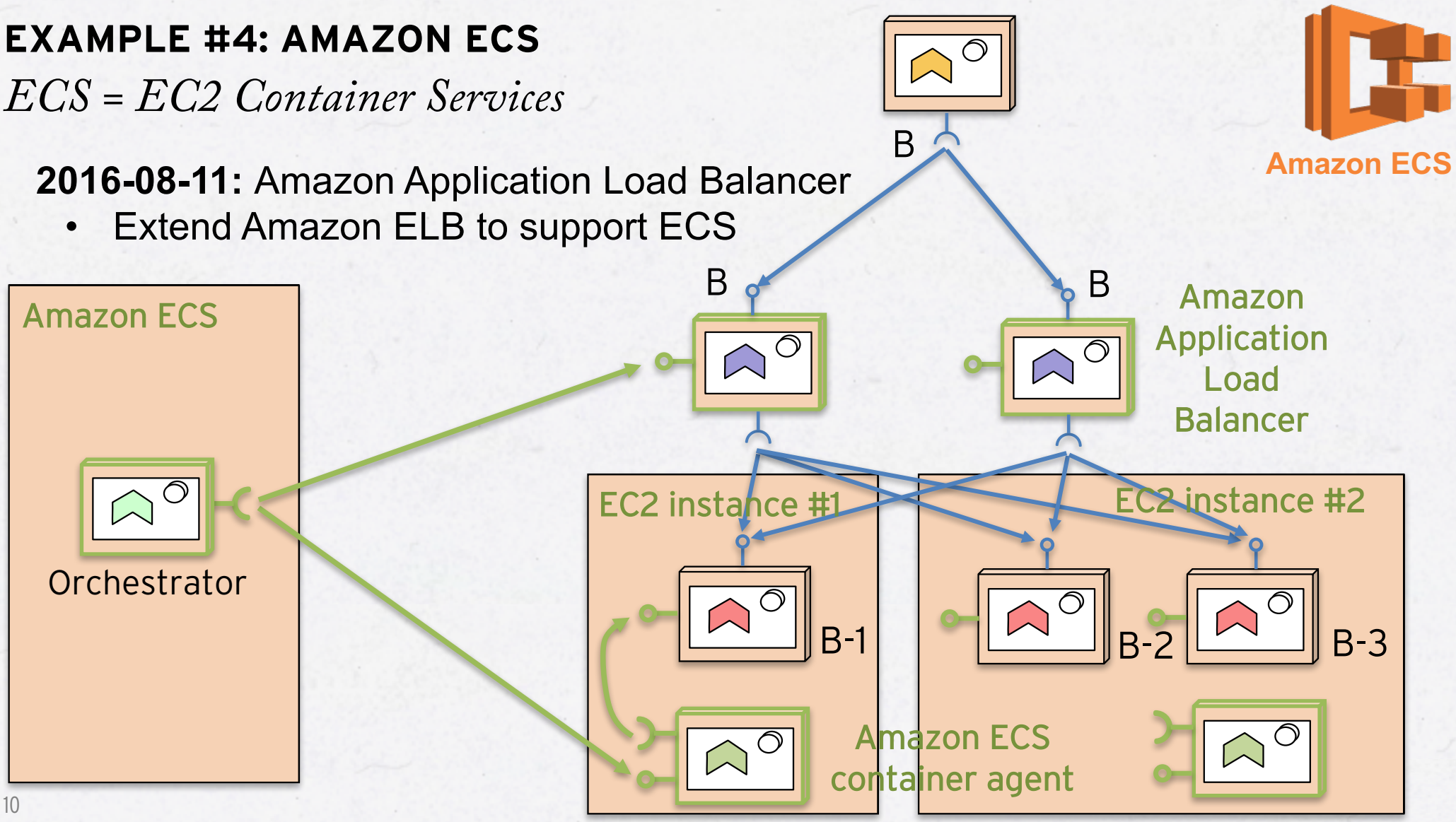
```
$ kubectl get pods
NAME                       READY     STATUS    RESTARTS   AGE
quotes-4029858897-2nsm6    1/1       Running   0          2m
quotes-4029858897-5xn93    1/1       Running   0          17m
quotes-4029858897-82vdc    1/1       Running   0          6m
quotes-4029858897-d5ctp    1/1       Running   0          6m
quotes-4029858897-s6sl4    0/1       Pending   0          2m
quotes-4029858897-w8crj    0/1       Pending   0          2m
```

```
$ kubectl get nodes
NAME                          STATUS                        AGE
kubernetes-master             Ready,SchedulingDisabled      21m
kubernetes-minion-group-4ptj  Ready                         22m
kubernetes-minion-group-l6kv  NotReady                      6s
kubernetes-minion-group-xq6d  Ready                         18m
```

EXAMPLE #4: AMAZON ECS

ECS = EC2 Container Services

- **2016-08-11:** Amazon Application Load Balancer
  - Extend Amazon ELB to support ECS

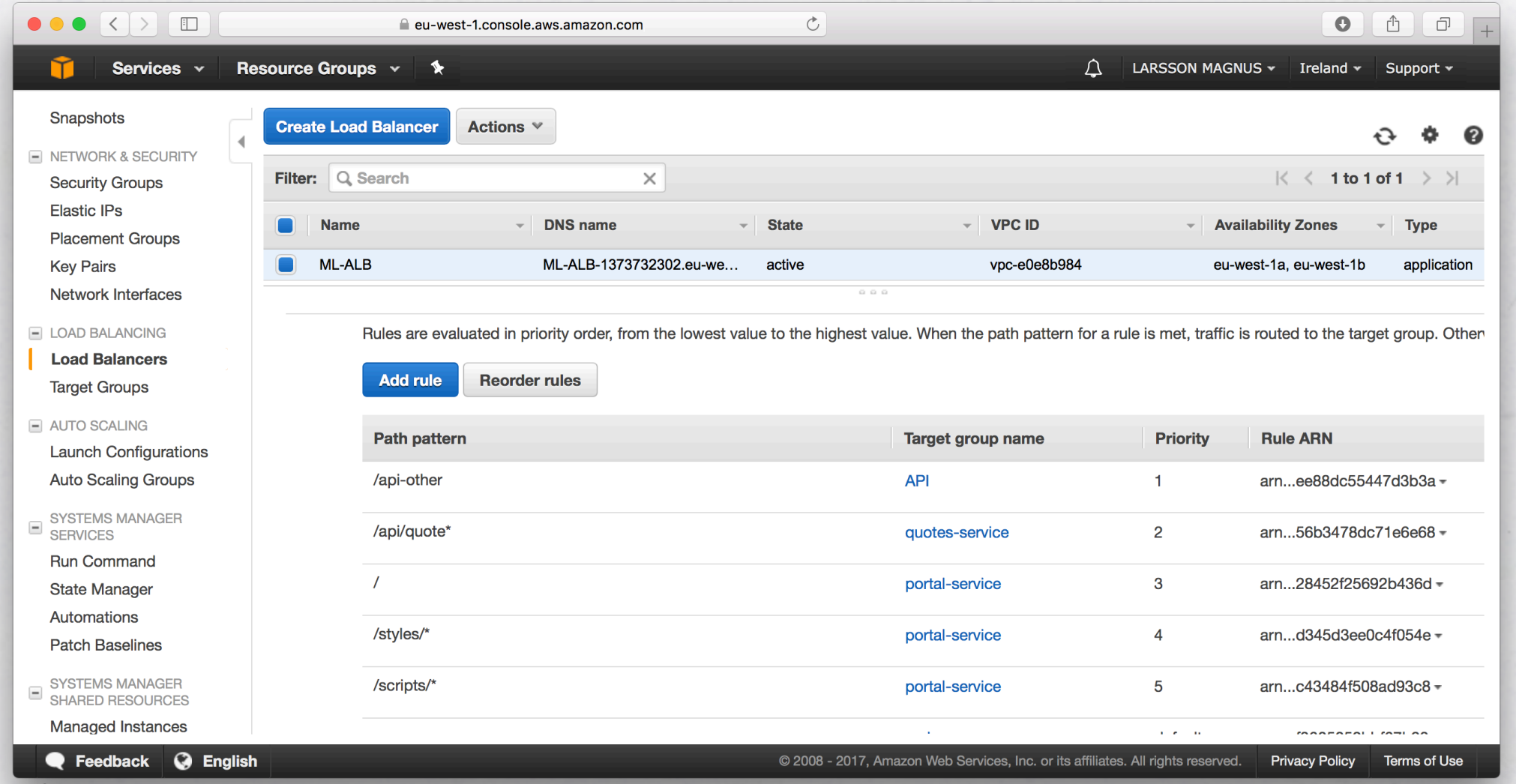


EXAMPLE #4: AMAZON ECS

Tests

- Setup a Amazon ECS Cluster with Application Load Balancer
- Same tests as for Kubernetes in Google Cloud

AMAZON APPLICATION LOAD BALANCER WITH PATH BASED ROUTING



IF YOU WANT TO LEARN MORE…

- Blog series – Building microservices:
  http://callistaenterprise.se/blogg/teknik/2015/05/20/blog-series-building-microservices/

- Workshop in developing microservices
  - Build a set of collaborating microservices from ground up using Spring Boot, Spring Cloud, Netflix OSS and Docker.
  - ~~Jfokus – 2017-02-06~~
  - jDays – 2017-03-09

SUMMARY

- Mission accomplished, worked to write once and deployed on:
  - Without containers using *Netflix Eureka* as Discovery Service
  - Container orchestration tools (with built in Discovery Service):
    - » *Google Kubernetes*
    - » *Docker Swarm mode*
    - » *Amazon ECS*
- Only differs in configuration
- Prevents vendor lock in
- Platform specific features can be used, e.g. auto scaling in the cloud
- Can be accomplished using any language, not only Java