Service Discovery
In
Container Orchestration Frameworks

Arun Gupta, @arungupta

Docker Captain

Java Champion

JavaOne Rock Star (4 years)

NetBeans Dream Team

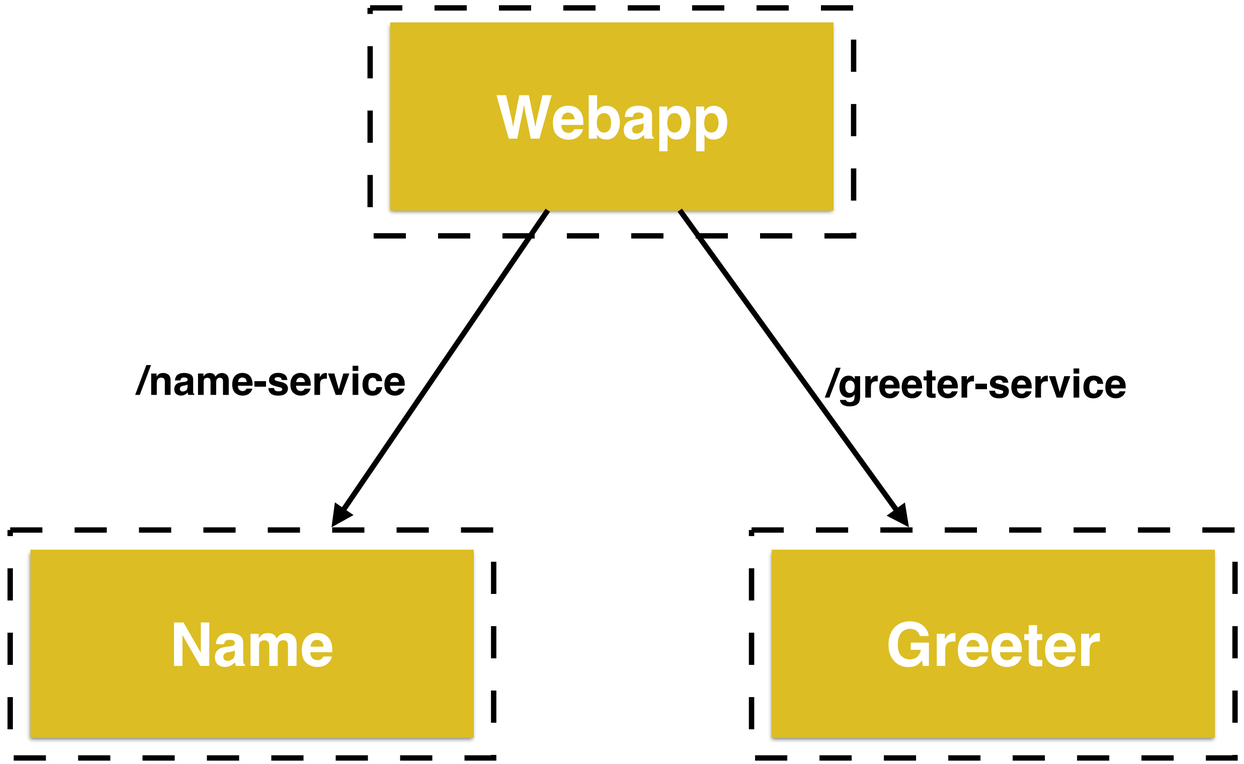
Silicon Valley JUG Leader

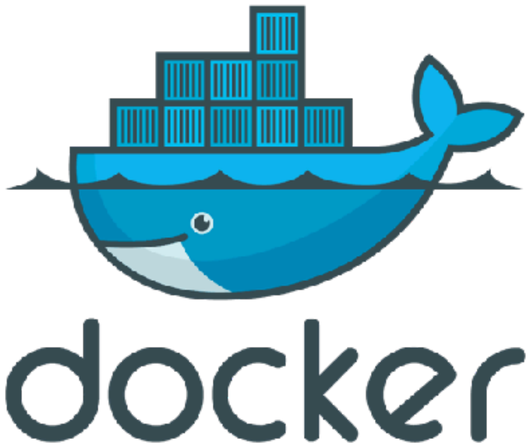
Author

Runner

Lifelong learner

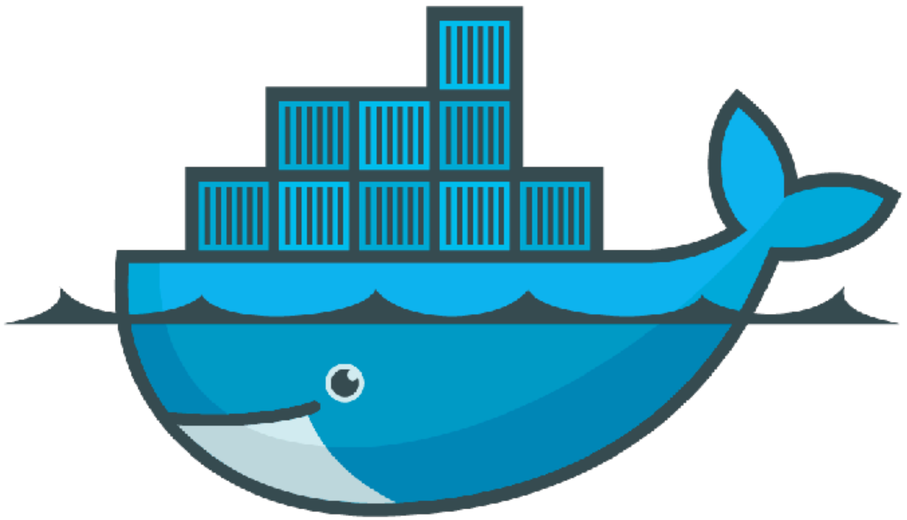
Service Discovery





kubernetes

MESOSPHERE

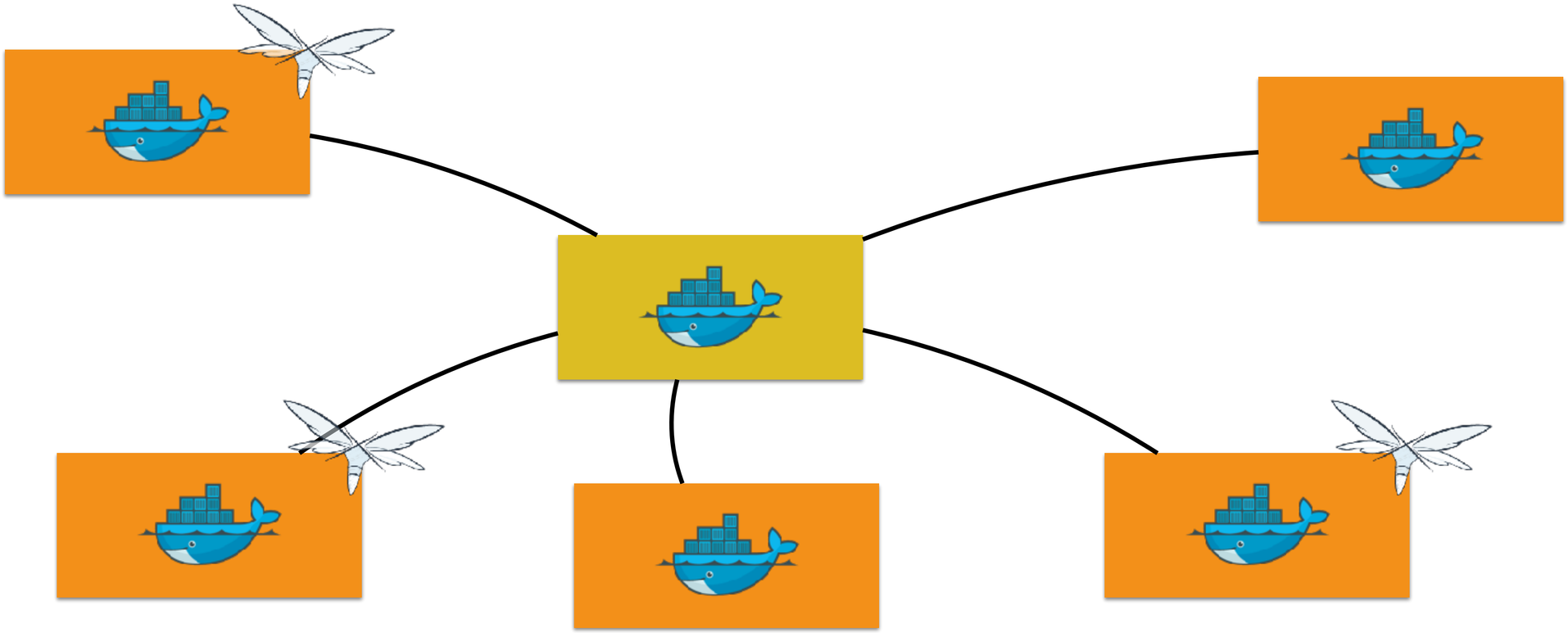


docker

Docker for AWS

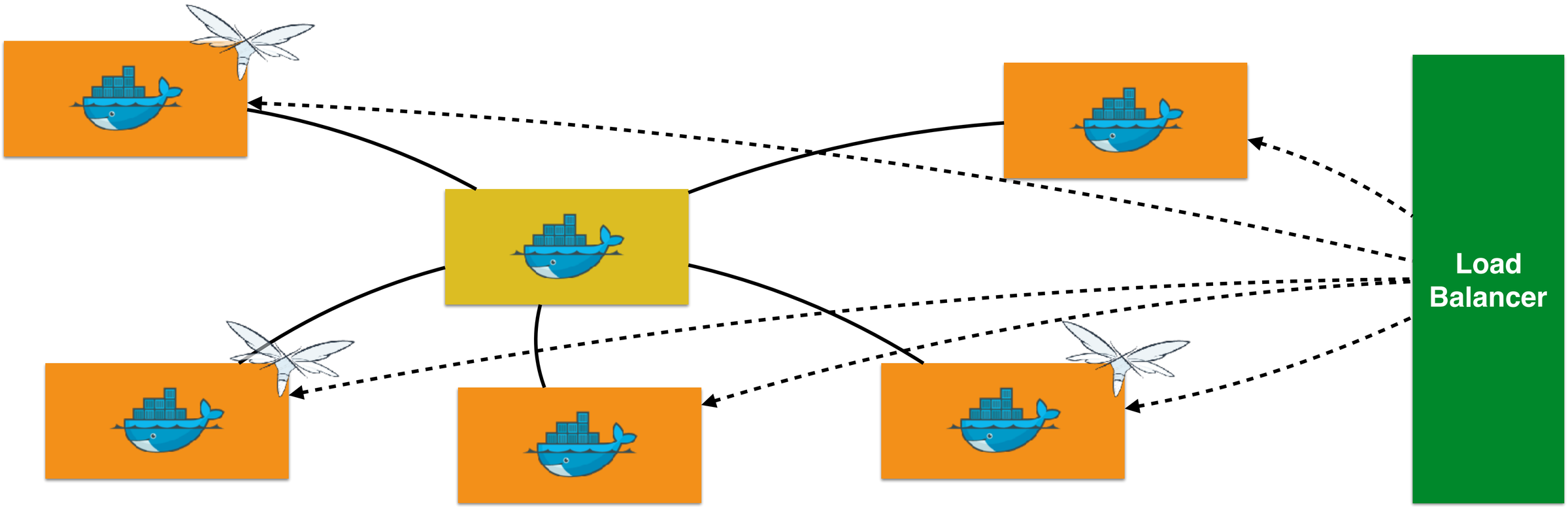
- CloudFormation templates
  - With a new VPC or a pre-existing VPC
- Integrated with
  - Autoscaling Groups - one for manager and worker each
  - Traffic routing using ELB
  - Container logs in CloudWatch
- Available in Docker CE and Docker EE

Swarm-mode: Replicated Service



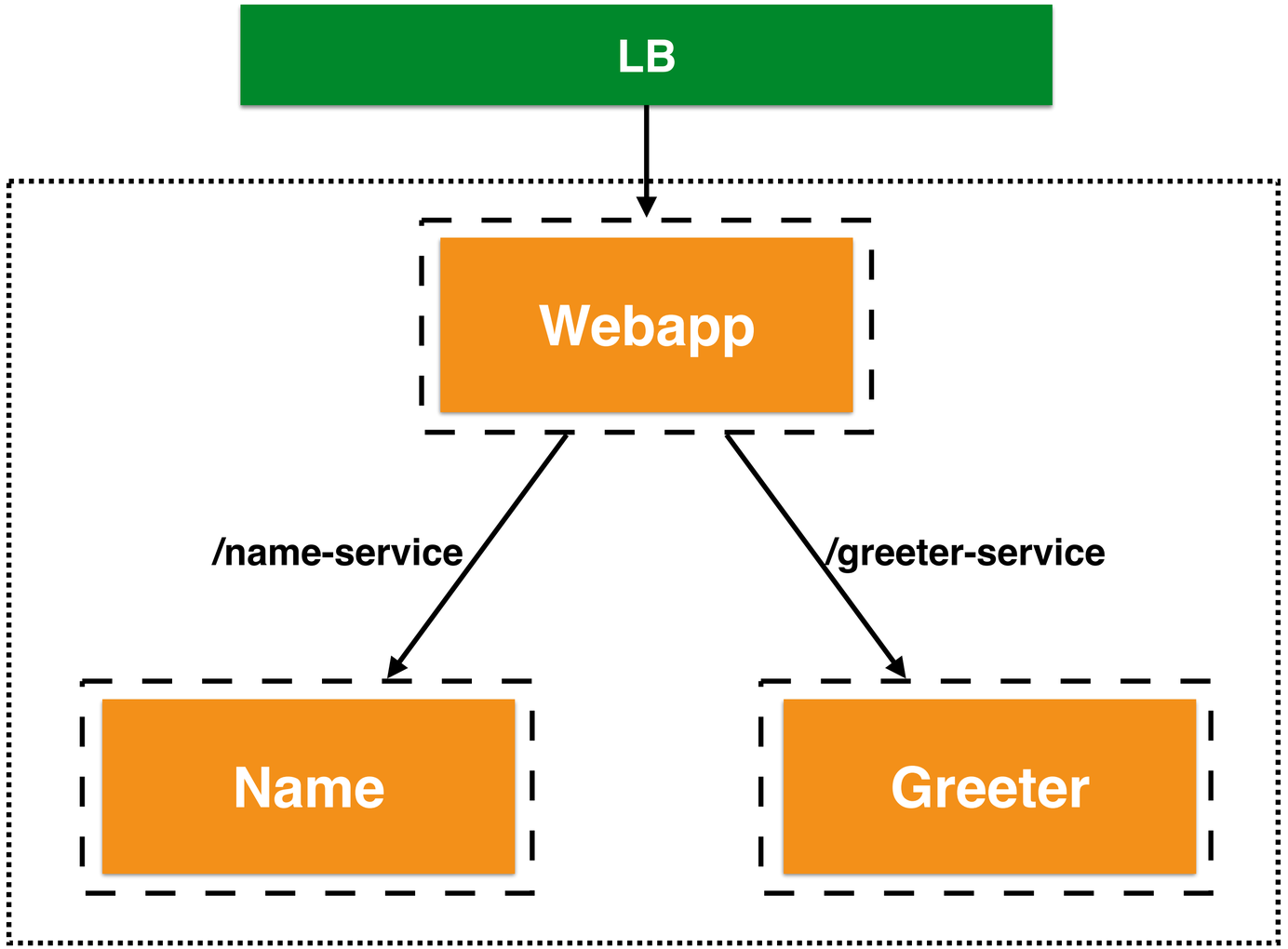
```
docker service create --replicas 3 --name web jboss/wildfly
```

Swarm-mode: Routing Mesh




```
docker service create --replicas 3 --name web -p 8080:8080 jboss/
wildfly
```

Docker

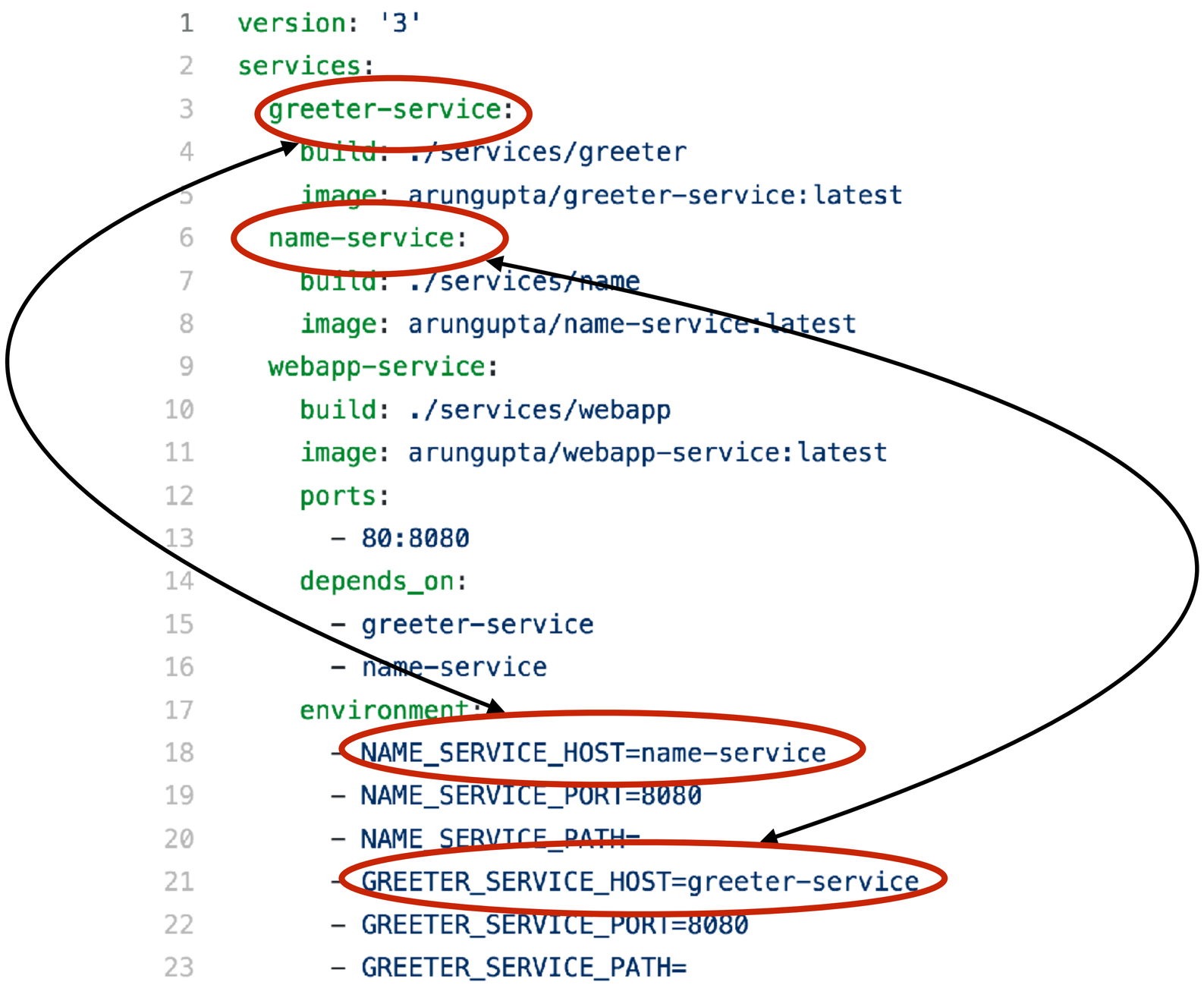


Task

Service

Stack

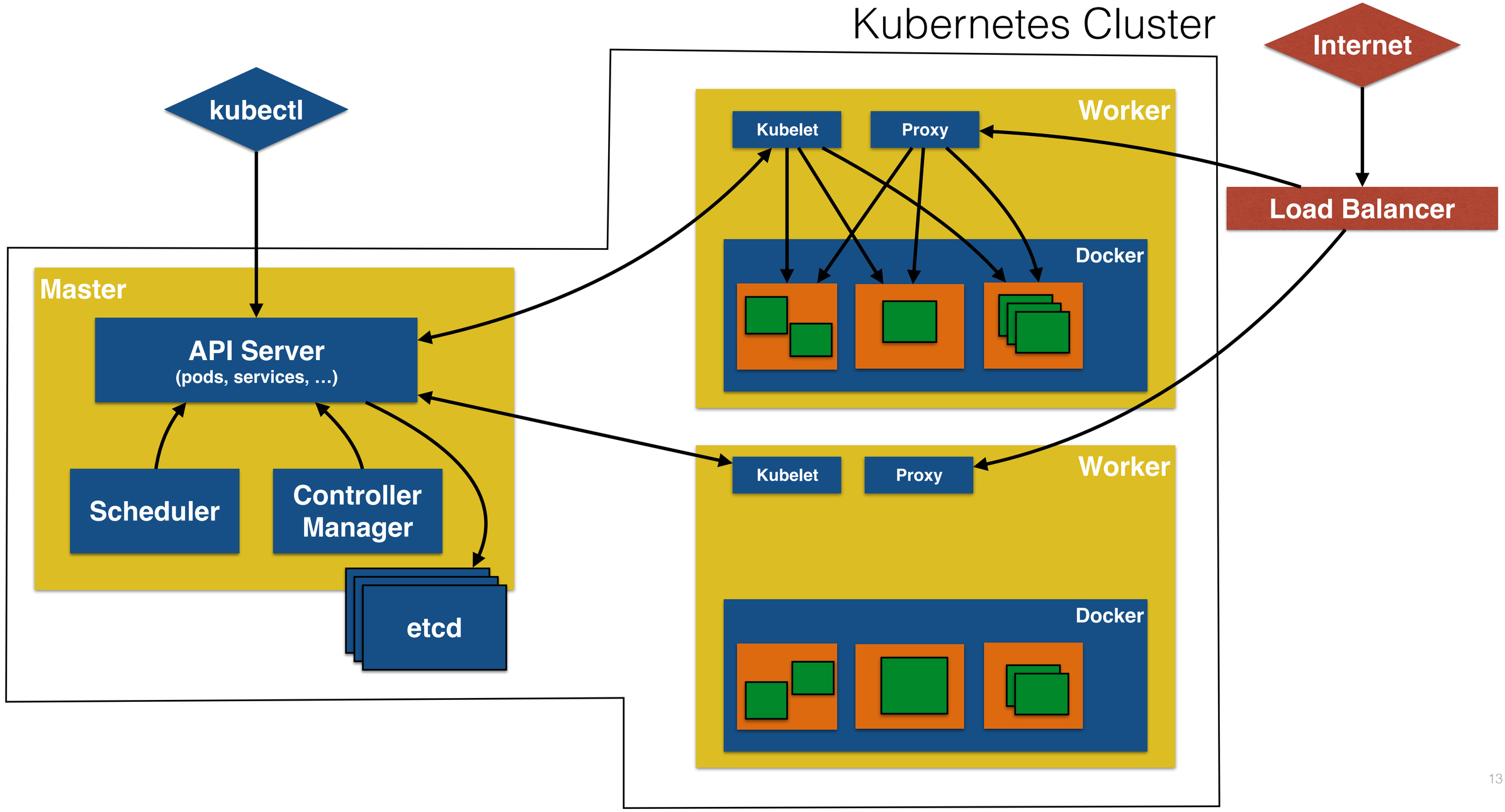
Elastic Load Balancer


version: '3'
services:
greeter-service:
build: ./services/greeter
image: arungupta/greeter-service:latest
name-service:
build: ./services/name
image: arungupta/name-service:latest
webapp-service:
build: ./services/webapp
image: arungupta/webapp-service:latest
ports:
- 80:8080
depends_on:
- greeter-service
- name-service
environment:
- NAME_SERVICE_HOST=name-service
- NAME_SERVICE_PORT=8080
- NAME_SERVICE_PATH=
- GREETER_SERVICE_HOST=greeter-service
- GREETER_SERVICE_PORT=8080
- GREETER_SERVICE_PATH=

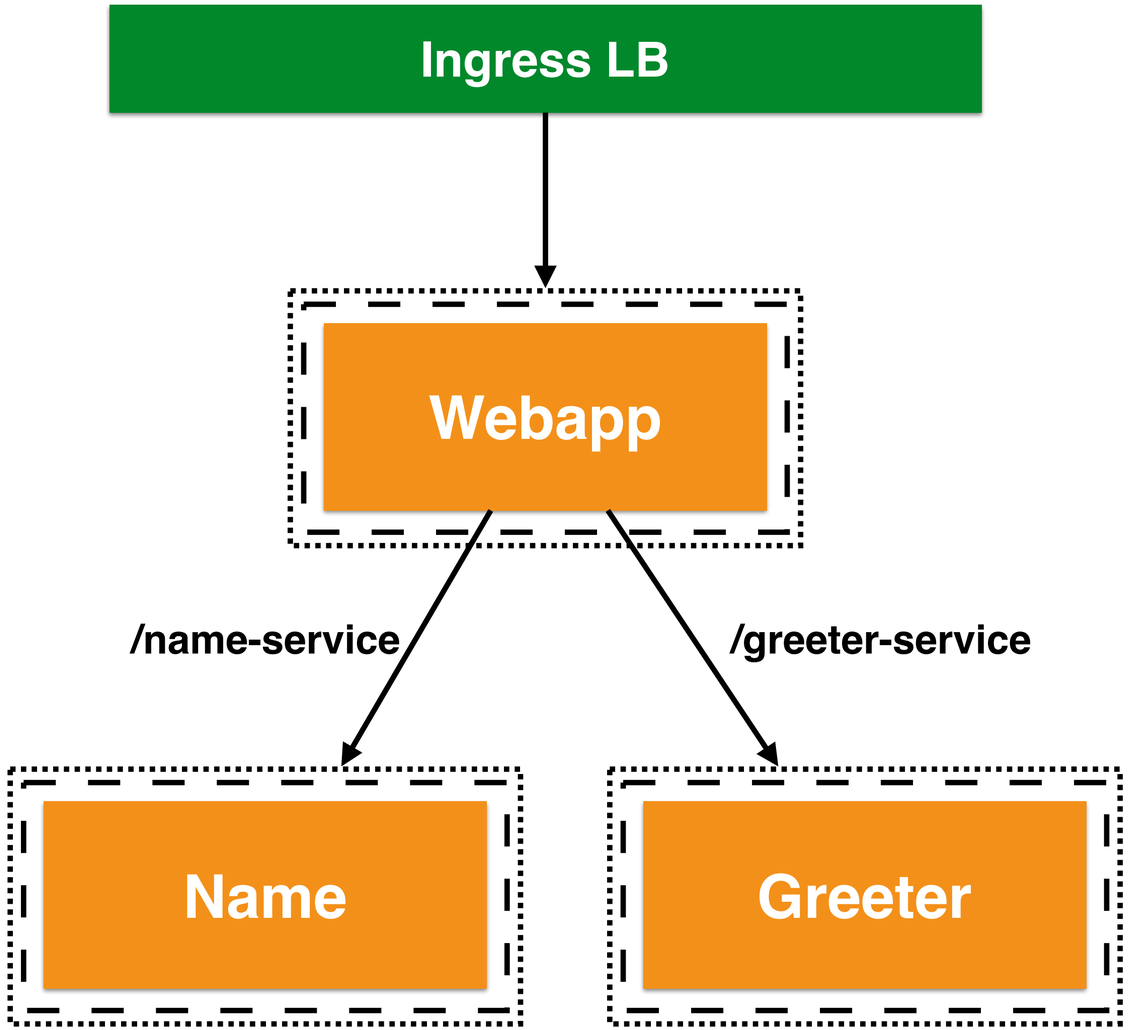
kubernetes

Kubernetes

- Open source orchestration system for containers
  - Docker, rkt, OCI, …
- A CNCF project
- Provide declarative primitives for the “desired state”
  - Self-healing
  - Horizontal scaling
  - Automatic binpacking
  - Service discovery and load balancing


Kubernetes Cluster
Internet
kubectl
Worker
Kubelet
Proxy
Load Balancer
Docker
Master
API Server
(pods, services, …)
Scheduler
Controller
Manager
etcd
Kubelet
Proxy
Worker
Docker

Kubernetes



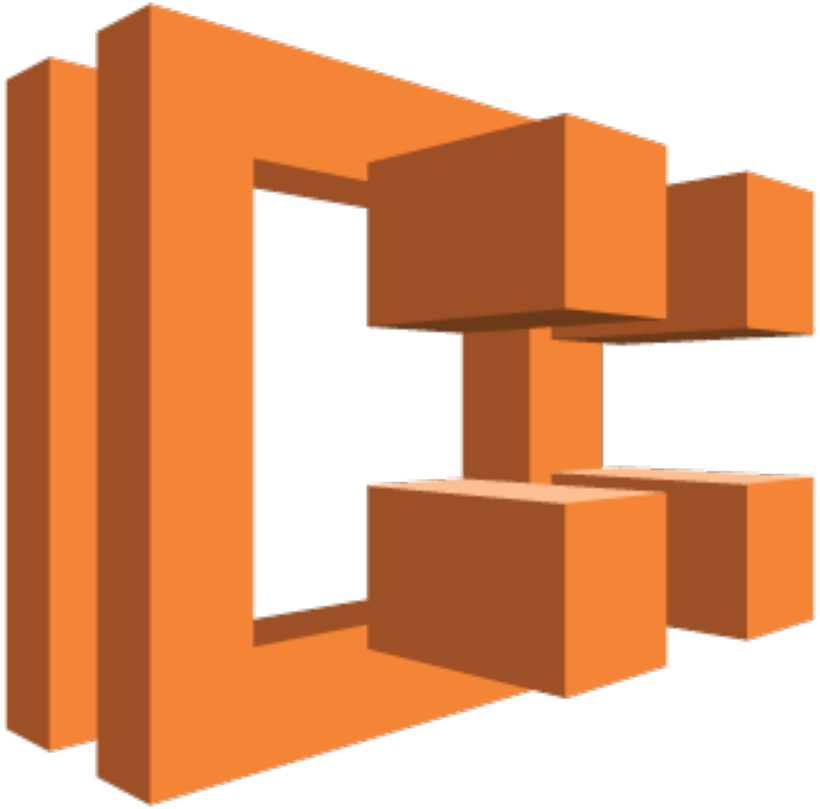
Pod

Replica Set

Service

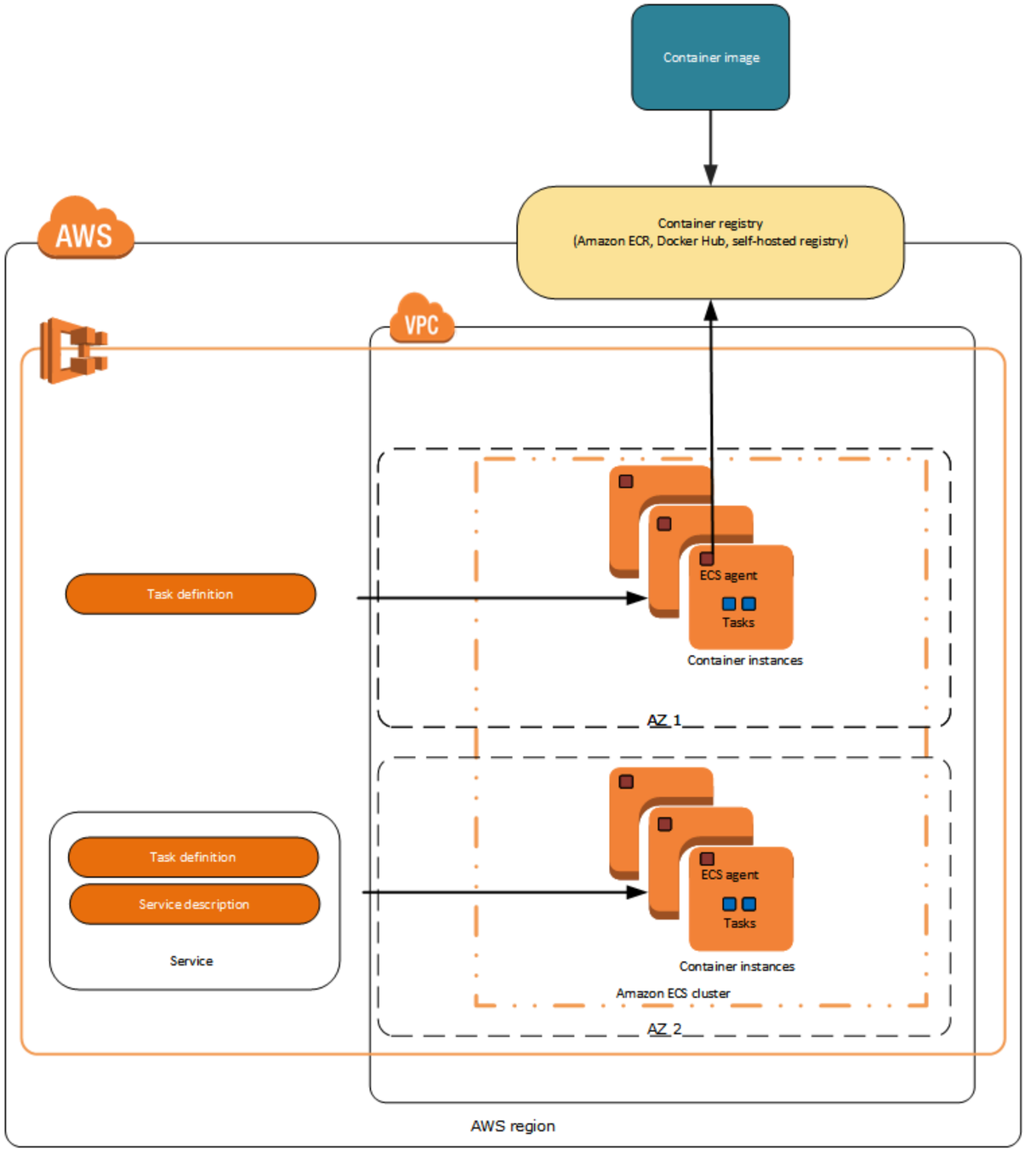
Elastic Load Balancer


apiVersion: v1
kind: Service
metadata:
name: name-service
spec:
selector:
app: name-pod
ports:
- port: 8080
---
apiVersion: extensions/v1beta1
kind: ReplicaSet
metadata:
name: name-rs
spec:
replicas: 1
template:
metadata:
labels:
app: name-pod
spec:
containers:
- name: name
image: arungupta/name-service:latest
ports:
- containerPort: 8080
---
apiVersion: v1
kind: Service
metadata:
name: greeter-service
spec:
selector:
app: greeter-pod
ports:
- port: 8080
---
apiVersion: extensions/v1beta1
kind: ReplicaSet
metadata:
name: greeter-rs
spec:
replicas: 1
template:
metadata:
labels:
app: greeter-pod
spec:
containers:
- name: name
image: arungupta/greeter-service:latest
ports:
- containerPort: 8080
---
apiVersion: v1
kind: Service
metadata:
name: webapp-service
spec:
selector:
app: webapp-pod
ports:
- name: web
port: 80
type: LoadBalancer
---
apiVersion: extensions/v1beta1
kind: ReplicaSet
metadata:
name: webapp-rs
spec:
replicas: 1
template:
metadata:
labels:
app: webapp-pod
spec:
containers:
- name: webapp-pod
image: arungupta/webapp-service:latest
env:
- name: NAME_SERVICE_HOST
value: name-service
- name: NAME_SERVICE_PORT
value: "8080"
- name: NAME_SERVICE_PATH
value: /
- name: GREETER_SERVICE_HOST
value: greeter-service
- name: GREETER_SERVICE_PORT
value: "8080"
- name: GREETER_SERVICE_PATH
value: /
ports:
- containerPort: 8080



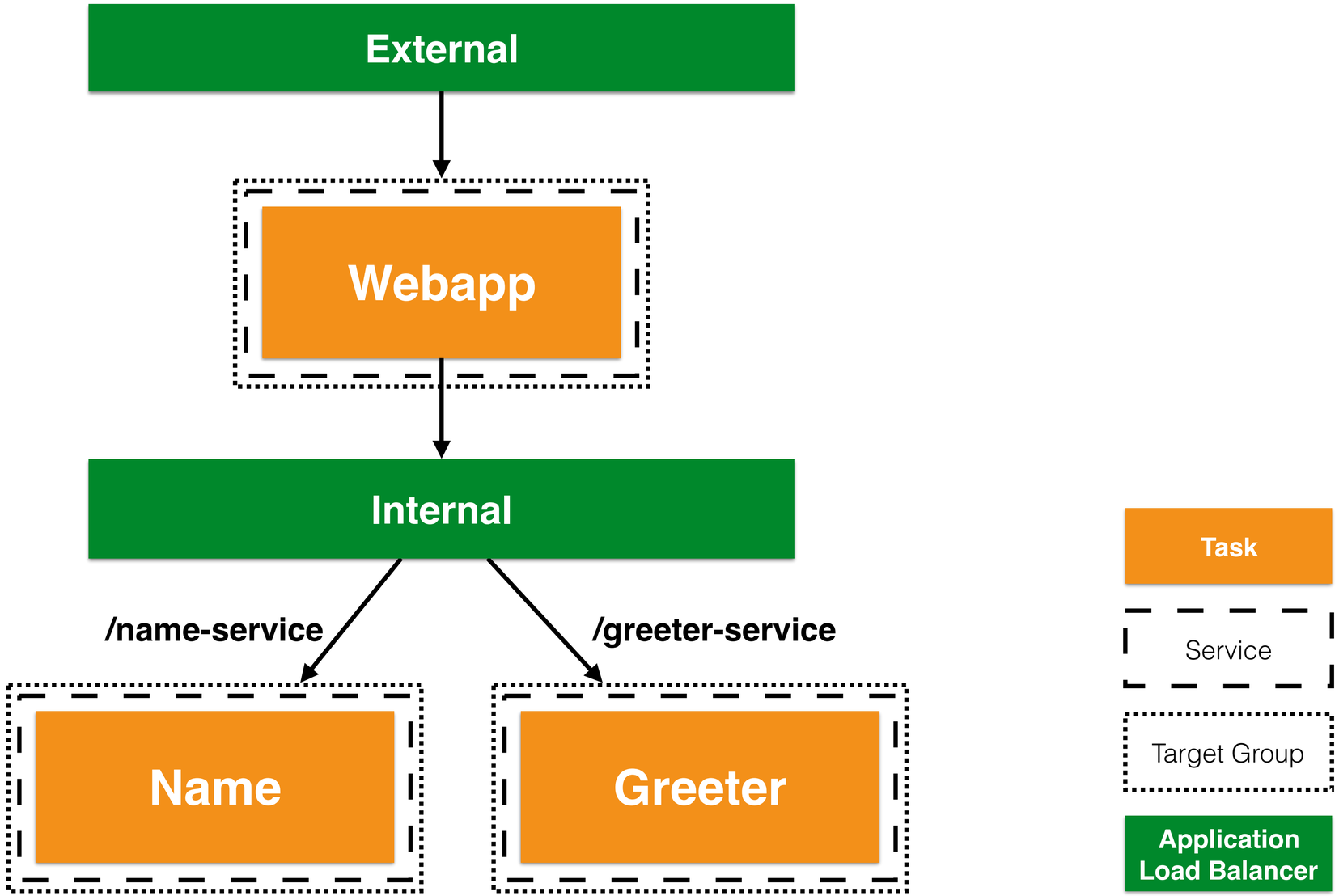
Amazon EC2 Container Service

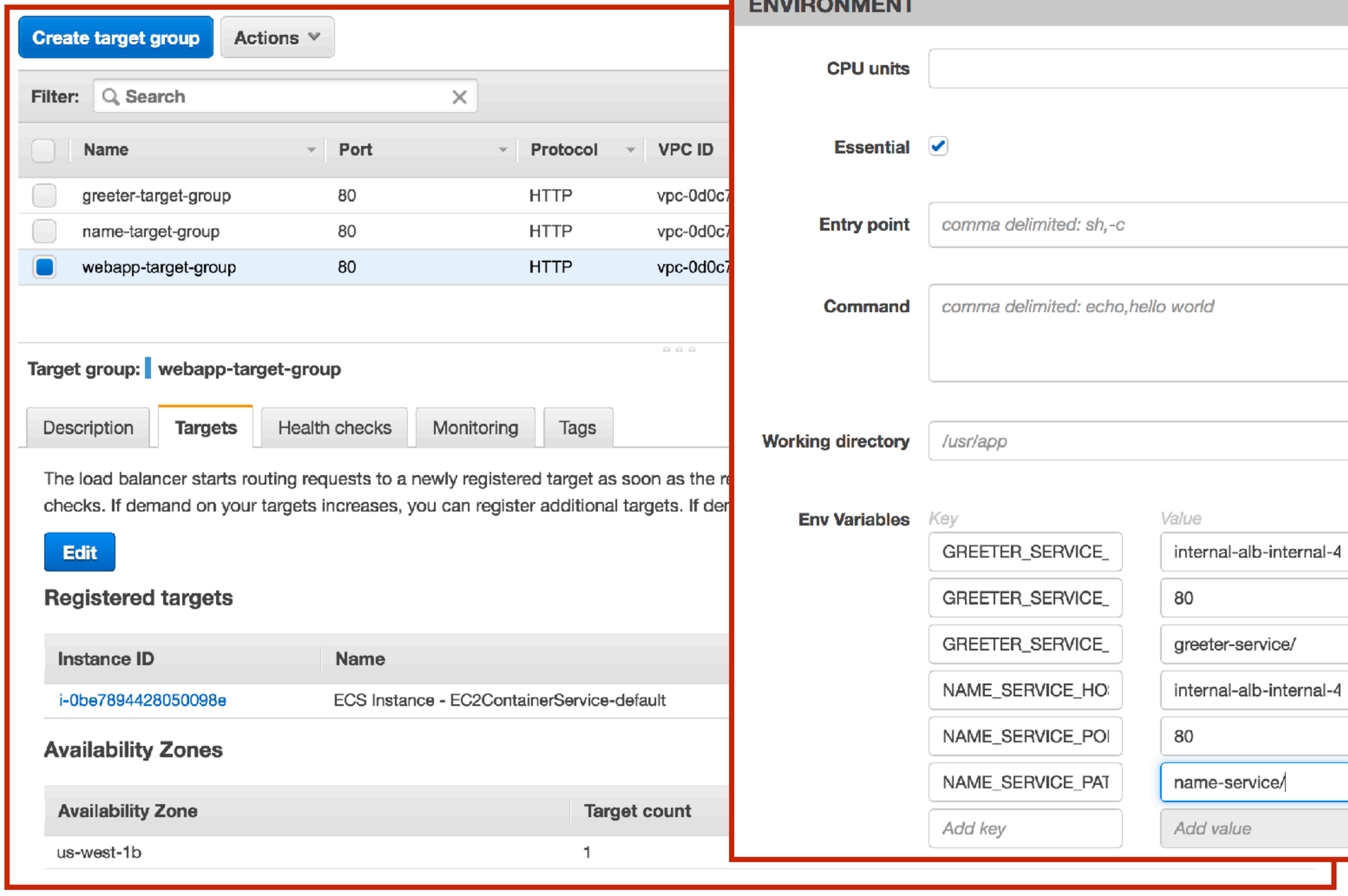
- Container management service on Amazon EC2 instances
- Fully-managed: no need to install, operate and scale your own infrastructure
- Resource management, scheduling, task placement
- Designed for use with other AWS services
  - ELB, VPC, CloudWatch, CloudTrail, IAM, …



http://docs.aws.amazon.com/AmazonECS/latest/developerguide/

Amazon EC2 Container Service



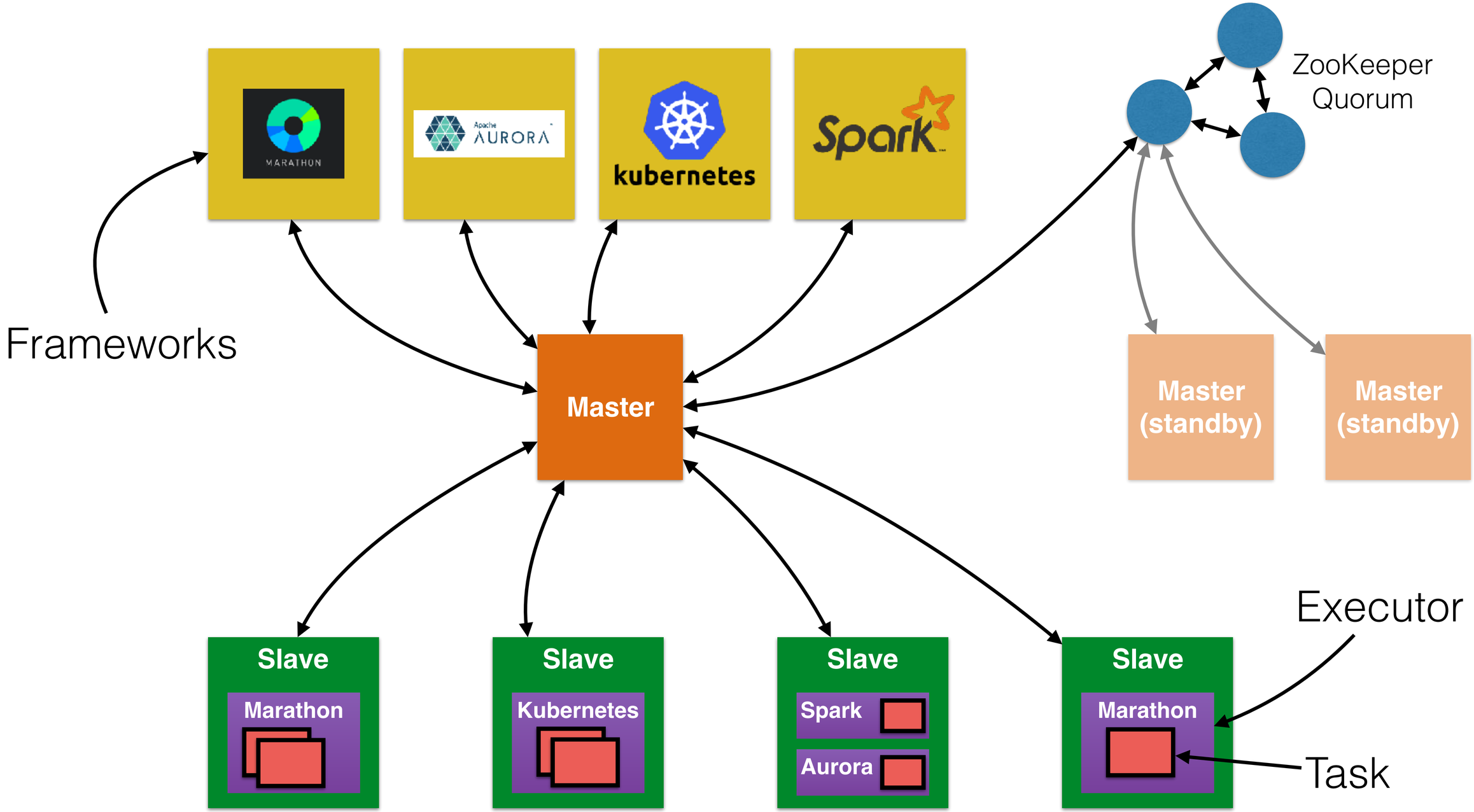

Create target group
Actions
Filter:
Search
Name
Port
Protocol
VPC ID
greeter-target-group
80
HTTP
name-target-group
80
HTTP
webapp-target-group
80
HTTP
Target group: webapp-target-group
Description
Targets
Health checks
Monitoring
Tags
The load balancer starts routing requests to a newly registered target as soon as the
checks. If demand on your targets increases, you can register additional targets. If
Edit
Registered targets
Instance ID
Name
i-0be7894428050098e
ECS Instance - EC2ContainerService-default
Availability Zones
Availability Zone
Target count
us-west-1b
1

ENVIRONMENT
CPU units
Essential
Entry point
comma delimited: sh,-c
Command
comma delimited: echo,hello world
Working directory
/usr/app
Env Variables
Key
Value
GREETER_SERVICE_
internal-alb-internal-4
GREETER_SERVICE_
80
GREETER_SERVICE_
greeter-service/
NAME_SERVICE_HO
internal-alb-internal-4
NAME_SERVICE_PO
80
NAME_SERVICE_PAT
name-service/
Add key
Add value

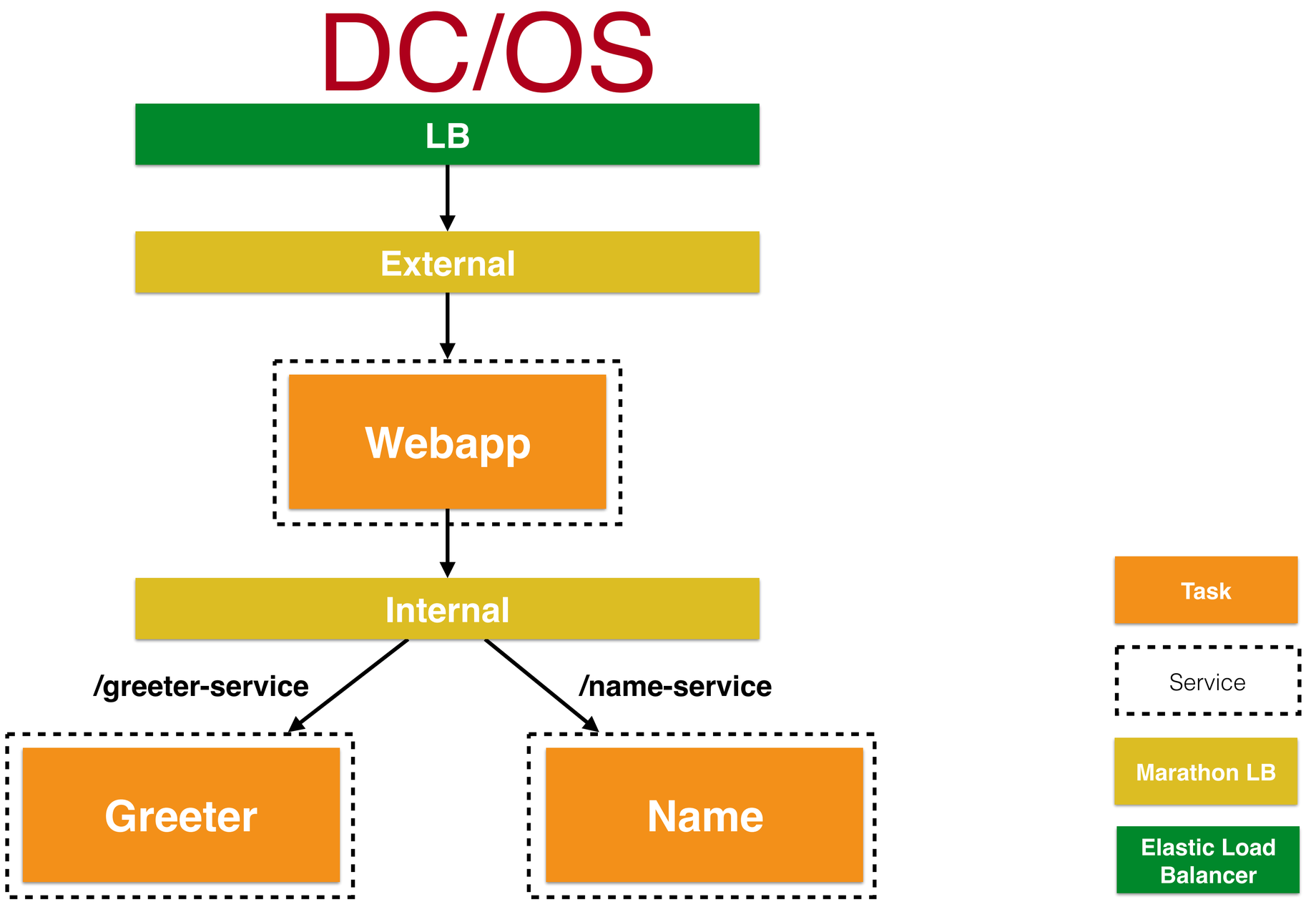
MESOSPHERE

DC/OS

- Open source cluster manager
- Developed at UC Berkeley
- Provides resource isolation and sharing across distributed applications
- Run distributed systems on the same pool of nodes
  - Docker, Hadoop, Spark, Jenkins, …
- Cluster monitoring
- Tasks isolated via Linux containers



http://mesos.apache.org/documentation/latest/architecture/


DC/OS
LB
External
Webapp
Internal
/greeter-service
/name-service
Greeter
Name
Task
Service
Marathon LB
Elastic Load Balancer

```
{
  "marathon-lb":{
    "name":"marathon-lb-internal",
    "haproxy-group":"internal",
    "bind-http-https":false,
    "role":""
  }
}
```

```
{
  "id": "/name",
  "cpus": 1,
  "mem": 1024,
  "instances": 1,
  "container": {
    "type": "DOCKER",
    "docker": {
      "image": "arungupta/name-service:latest"
      "network": "BRIDGE",
      "portMappings": [
        {
          "hostPort": 0,
          "containerPort": 8080,
          "servicePort": 10001
        }
      ]
    }
  },
  "labels": {
    "HAPROXY_GROUP": "internal"
  }
}
```

```
{
  "id": "/greeter",
  "cpus": 1,
  "mem": 1024,
  "instances": 1,
  "container": {
    "type": "DOCKER",
    "docker": {
      "image": "arungupta/greeter-servic
      "network": "BRIDGE",
      "portMappings": [
        {
          "hostPort": 0,
          "containerPort": 8080,
          "servicePort": 10000
        }
      ]
    }
  },
  "labels": {
    "HAPROXY_GROUP": "internal"
  }
}
```

```
{
  "id":"/webapp",
  "cpus":1,
  "mem":1024,
  "instances":1,
  "container":{
    "type":"DOCKER",
    "docker":{
      "image":"arungupta/webapp-service:latest",
      "network":"BRIDGE",
      "portMappings":[
        {
          "hostPort":0,
          "containerPort":8080,
          "servicePort": 10001,
          "protocol":"tcp"
        }
      ]
    }
  },
  "env":{
    "NAME_SERVICE_HOST":"marathon-lb-internal.marathon.mesos",
    "NAME_SERVICE_PORT":"10001",
    "NAME_SERVICE_PATH":"/",
    "GREETER_SERVICE_HOST":"marathon-lb-internal.marathon.mesos",
    "GREETER_SERVICE_PORT":"10000",
    "GREETER_SERVICE_PATH":"/"
  },
  "labels":{
    "HAPROXY_0_VHOST":"dcos-PublicSlaveLo-1B7KS7V0OI0Y6-1504556418.us-west-1.elb.amazo
    "HAPROXY_GROUP":"external"
  }
}
```

References

- Docker: docker.io
- Amazon ECS: aws.amazon.com/ecs
- Kubernetes: kubernetes.io
- DC/OS: dcos.io
- Slides & Code: github.com/arun-gupta/container-service-discovery