

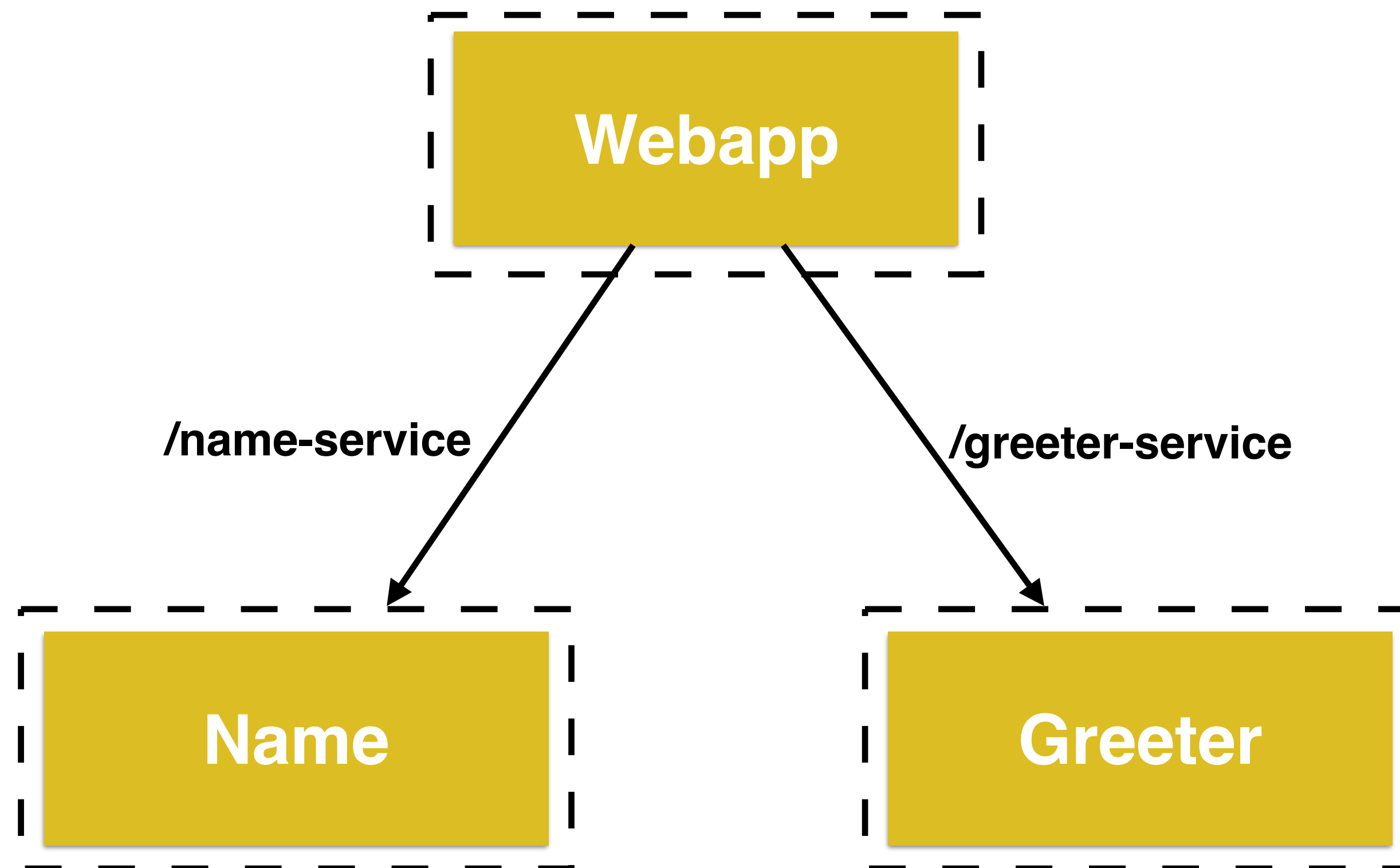
# Service Discovery In Container Orchestration Frameworks

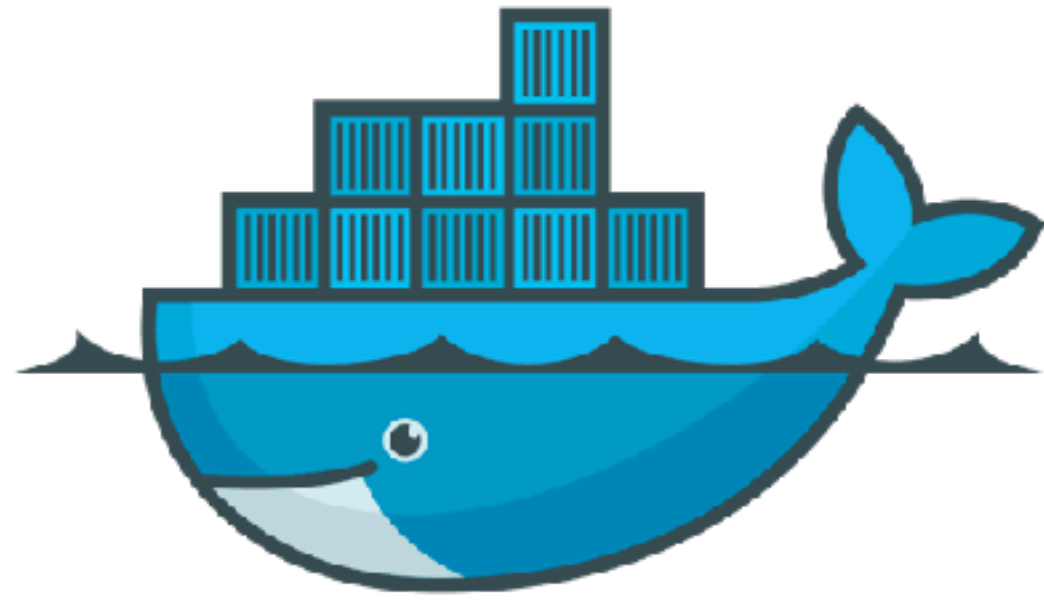
Arun Gupta, @arungupta  
Amazon Web Services

Docker Captain  
Java Champion  
JavaOne Rock Star (4 years)  
NetBeans Dream Team  
Silicon Valley JUG Leader  
Author  
Runner  
Lifelong learner



# Service Discovery

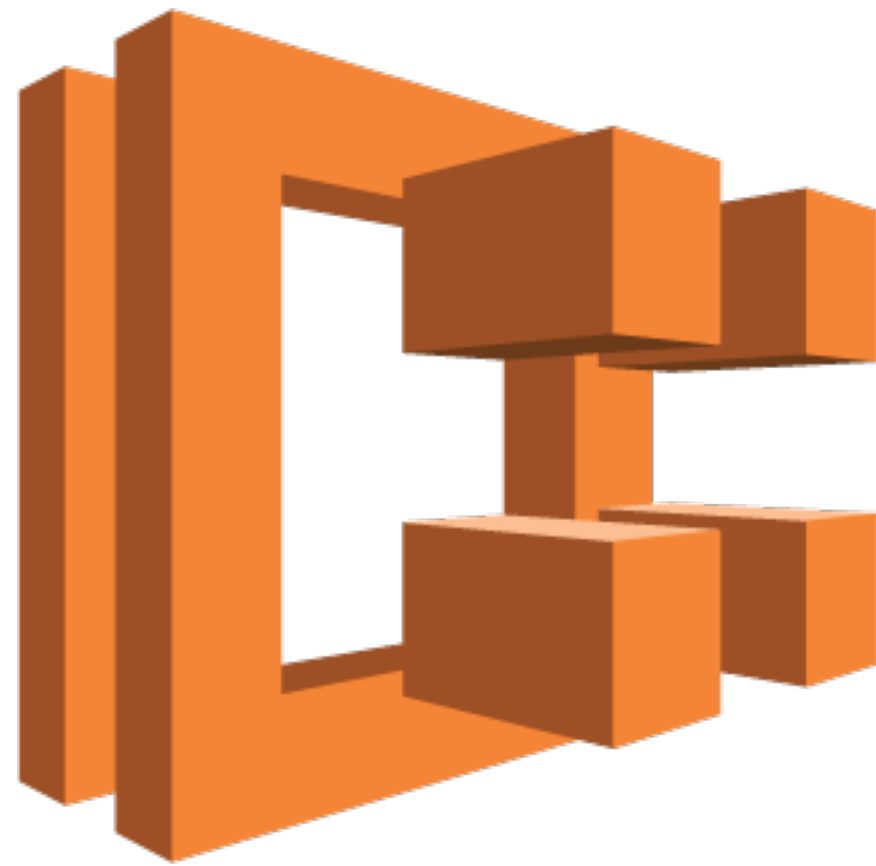




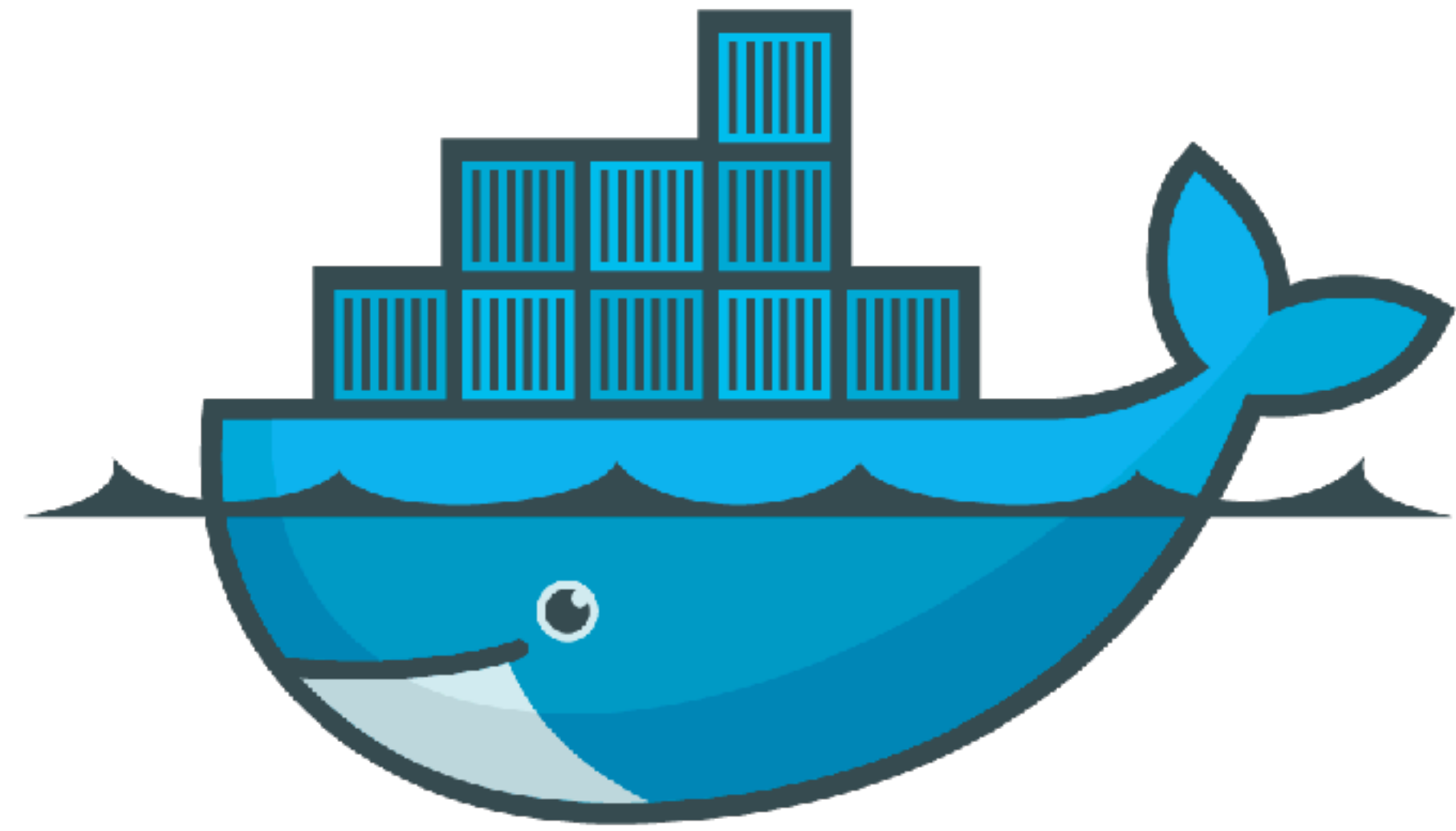
docker



kubernetes



MESOSPHERE

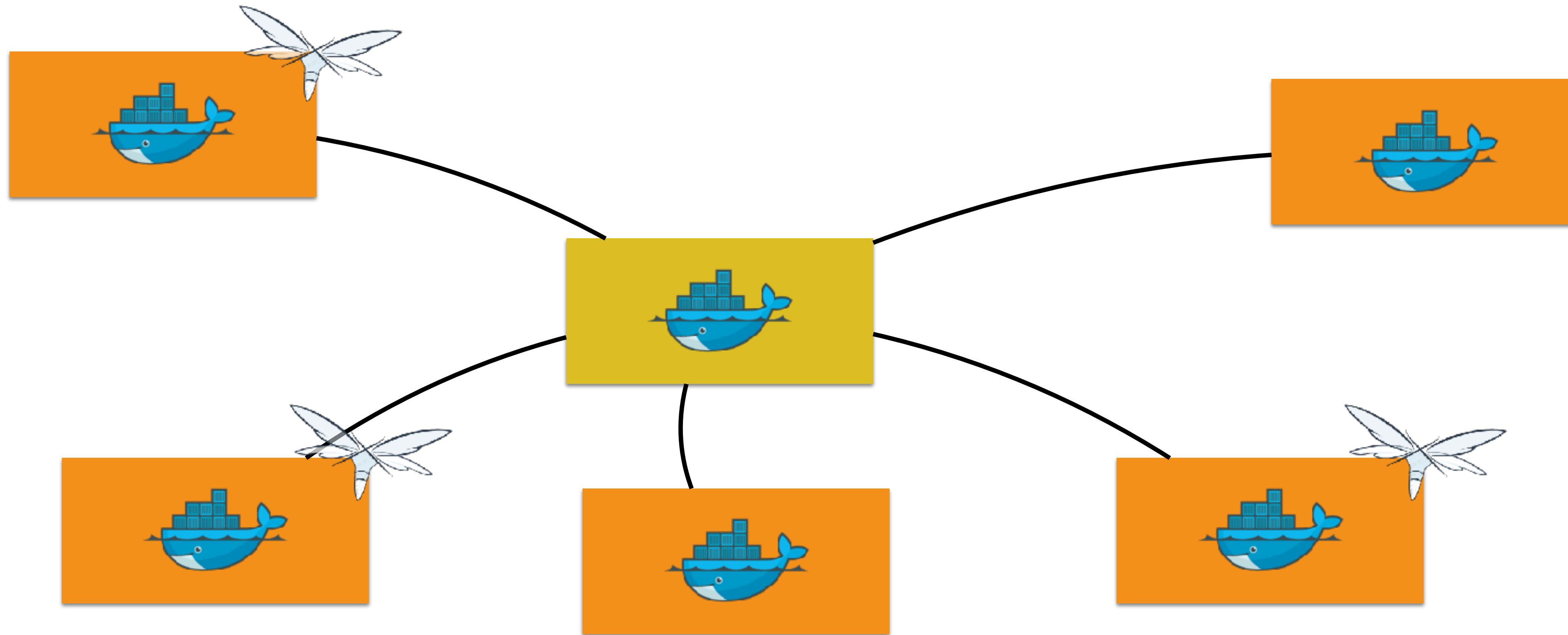


docker

# 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

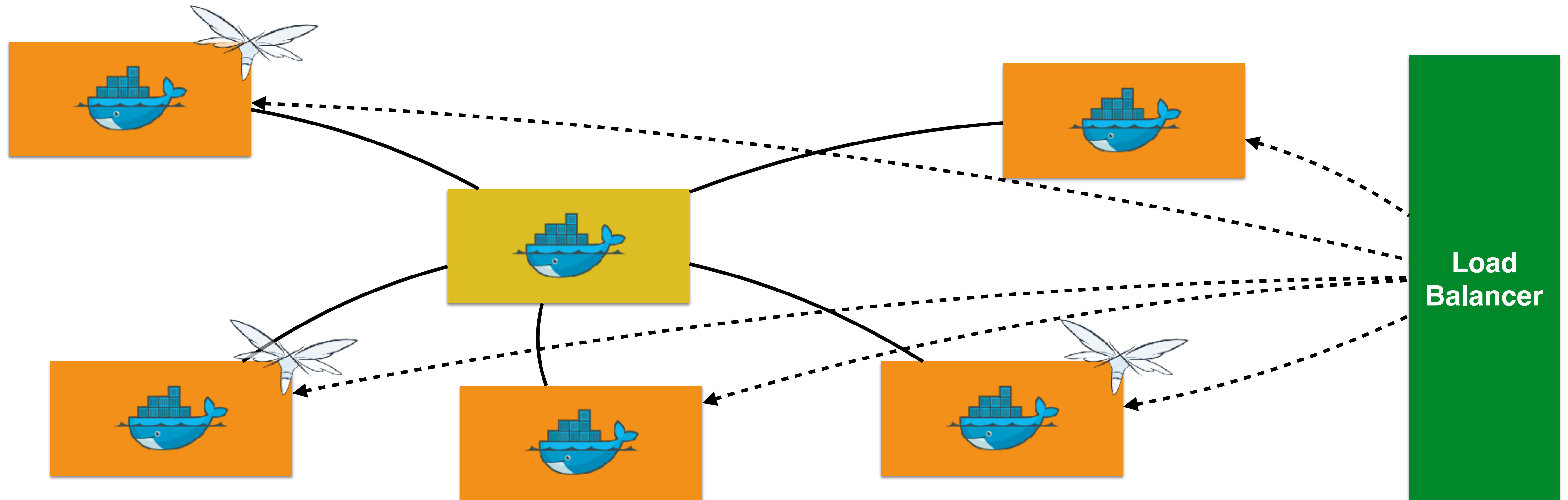
# Swarm-mode: Replicated Service



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



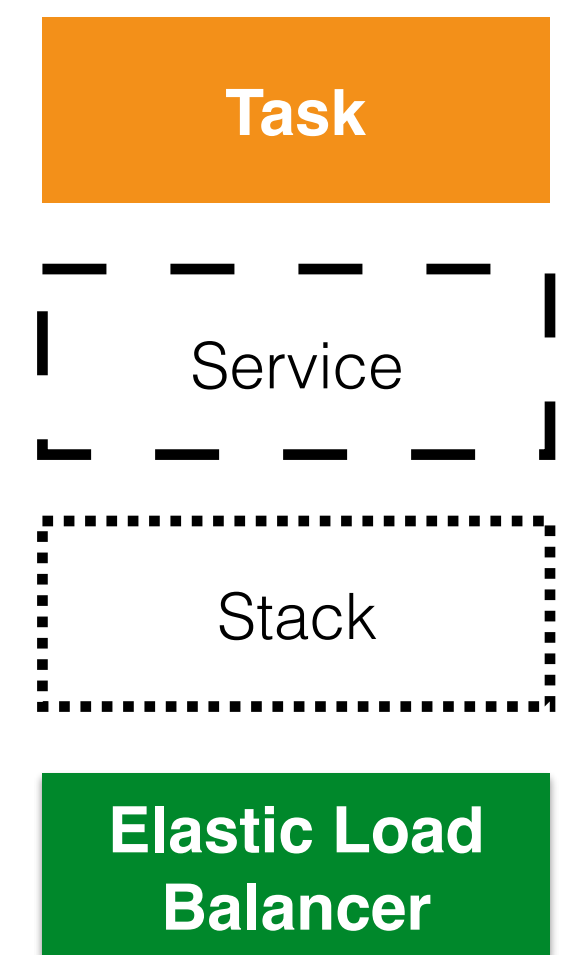
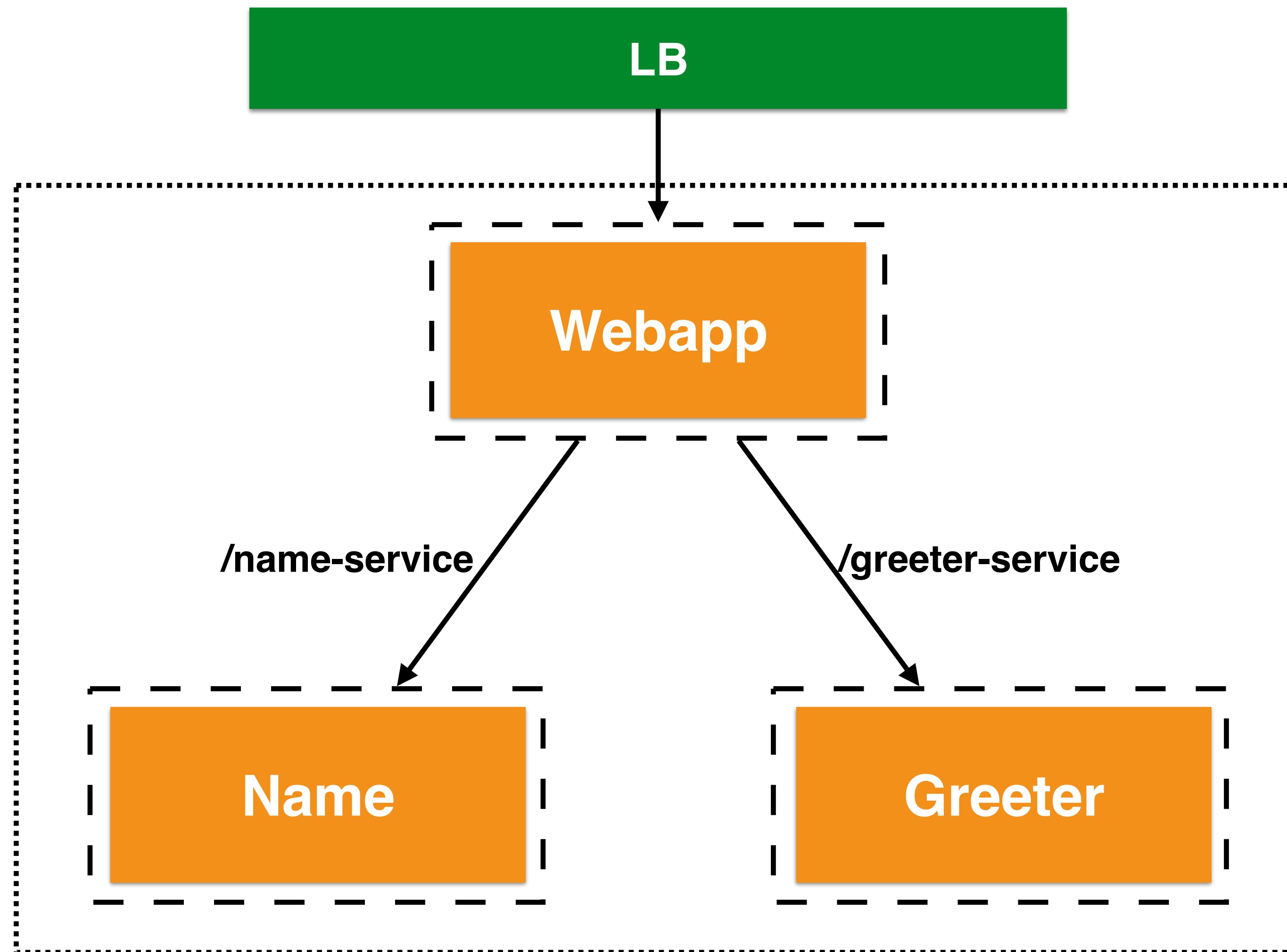
# Swarm-mode: Routing Mesh



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



# Docker



```
1  version: '3'
2  services:
3    greeter-service:
4      build: ./services/greeter
5      image: arungupta/greeter-service:latest
6    name-service:
7      build: ./services/name
8      image: arungupta/name-service:latest
9    webapp-service:
10     build: ./services/webapp
11     image: arungupta/webapp-service:latest
12     ports:
13       - 80:8080
14     depends_on:
15       - greeter-service
16       - name-service
17     environment:
18       - NAME_SERVICE_HOST=name-service
19       - NAME_SERVICE_PORT=8080
20       - NAME_SERVICE_PATH=
21       - GREETER_SERVICE_HOST=greeter-service
22       - GREETER_SERVICE_PORT=8080
23       - GREETER_SERVICE_PATH=
```



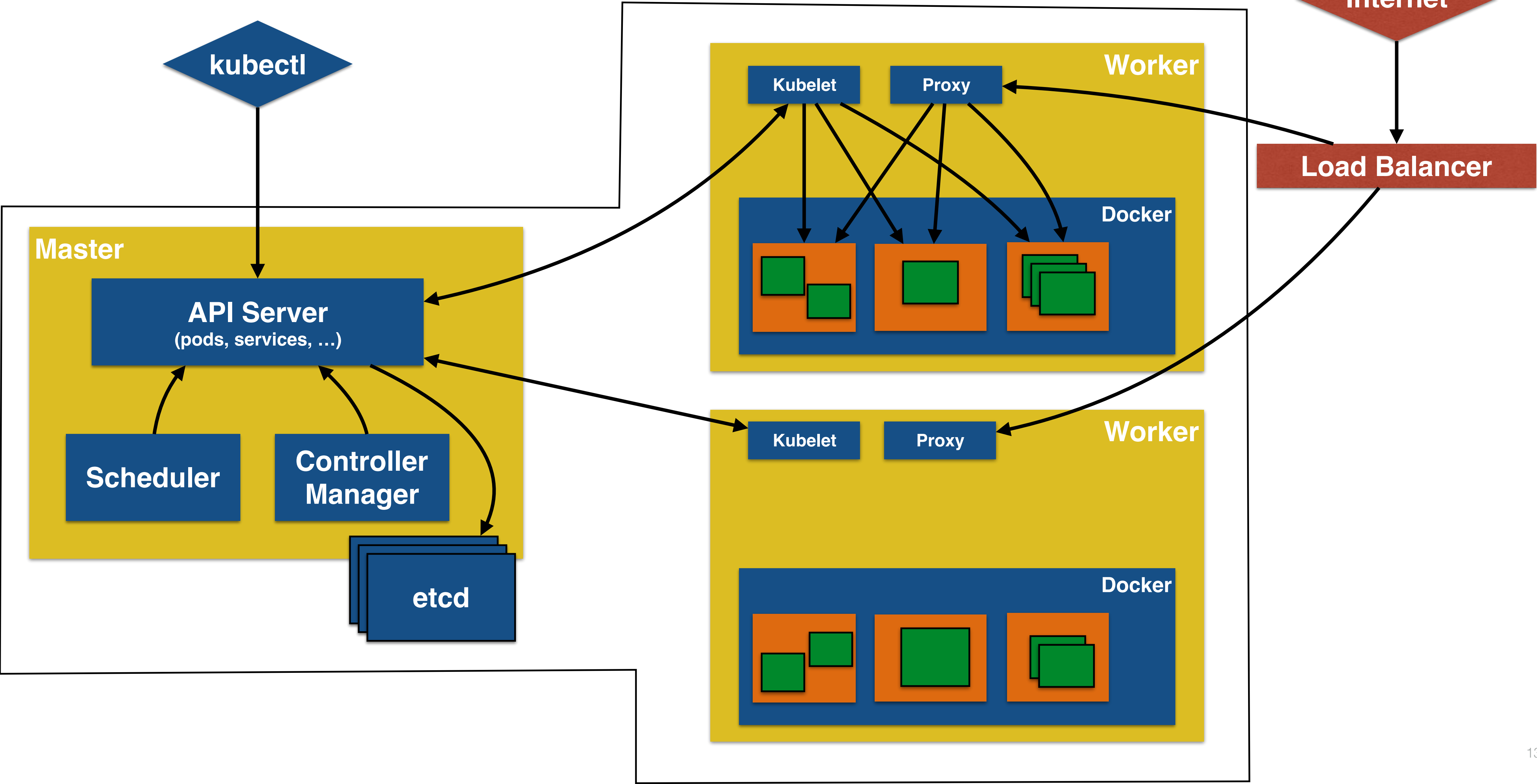


**kubernetes**

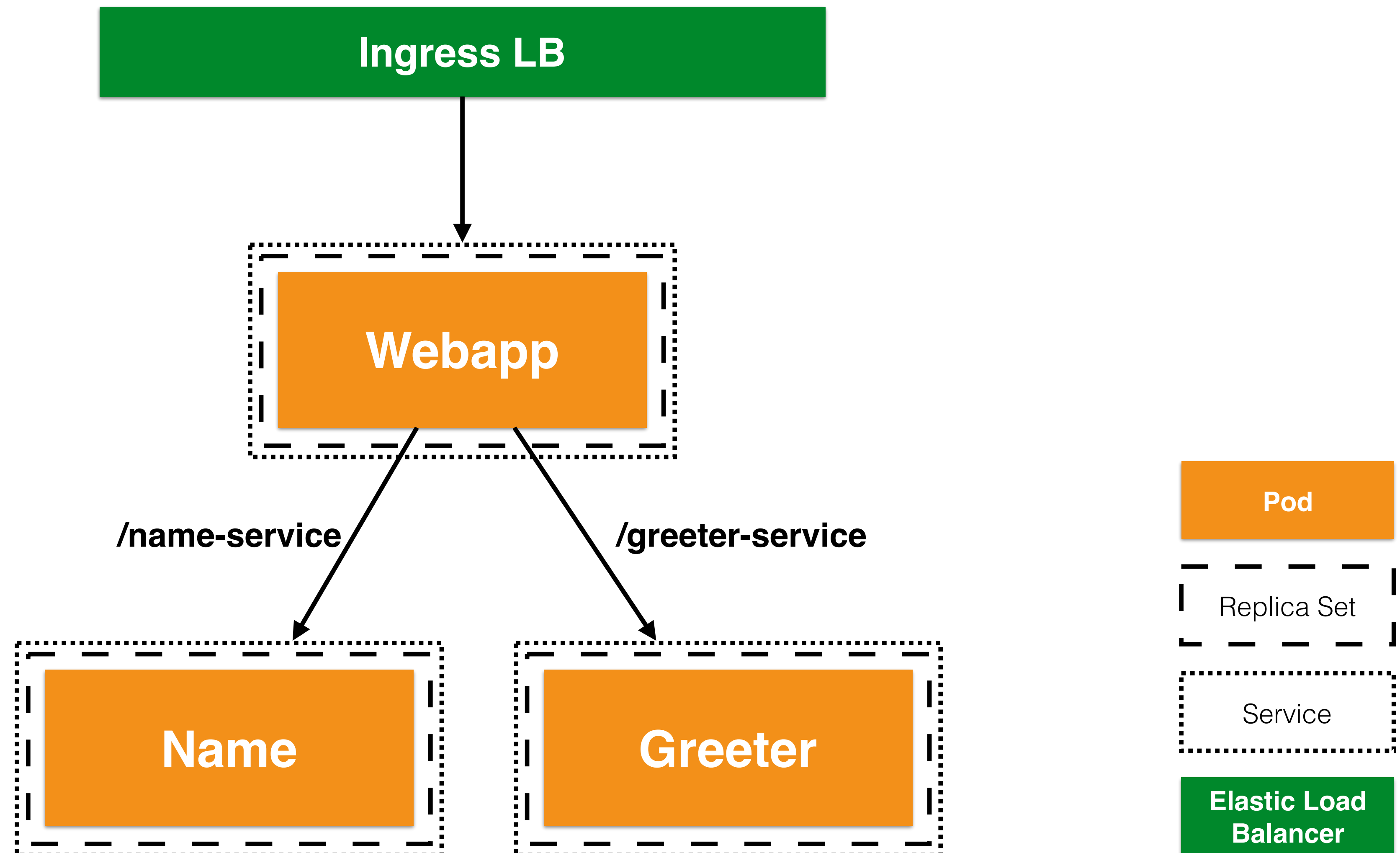
# Kubernetes

- 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



# Kubernetes





```
1  apiVersion: v1
2  kind: Service
3  metadata:
4    name: name-service
5  spec:
6    selector:
7      app: name-pod
8    ports:
9      - port: 8080
10 ---
11 apiVersion: extensions/v1beta1
12 kind: ReplicaSet
13 metadata:
14   name: name-rs
15 spec:
16   replicas: 1
17   template:
18     metadata:
19       labels:
20         app: name-pod
21     spec:
22       containers:
23         - name: name
24           image: arungupta/name-service:latest
25           ports:
26             - containerPort: 8080
27 ---
```

```
28 apiVersion: v1
29 kind: Service
30 metadata:
31   name: greeter-service
32 spec:
33   selector:
34     app: greeter-pod
35   ports:
36     - port: 8080
37 ---
38 apiVersion: extensions/v1beta1
39 kind: ReplicaSet
40 metadata:
41   name: greeter-rs
42 spec:
43   replicas: 1
44   template:
45     metadata:
46       labels:
47         app: greeter-pod
48     spec:
49       containers:
50         - name: name
51           image: arungupta/greeter-service:latest
52           ports:
53             - containerPort: 8080
54 ---
```

```
55 apiVersion: v1
56 kind: Service
57 metadata:
58   name: webapp-service
59 spec:
60   selector:
61     app: webapp-pod
62   ports:
63     - name: web
64       port: 80
65   type: LoadBalancer
66 ---
67 apiVersion: extensions/v1beta1
68 kind: ReplicaSet
69 metadata:
70   name: webapp-rs
71 spec:
72   replicas: 1
73   template:
74     metadata:
75       labels:
76         app: webapp-pod
77     spec:
78       containers:
79         - name: webapp-pod
80           image: arungupta/webapp-service:latest
81           env:
82             - name: NAME_SERVICE_HOST
83               value: name-service
84             - name: NAME_SERVICE_PORT
85               value: "8080"
86             - name: NAME_SERVICE_PATH
87               value: /
88             - name: GREETER_SERVICE_HOST
89               value: greeter-service
90             - name: GREETER_SERVICE_PORT
91               value: "8080"
92             - name: GREETER_SERVICE_PATH
93               value: /
94           ports:
95             - containerPort: 8080
```

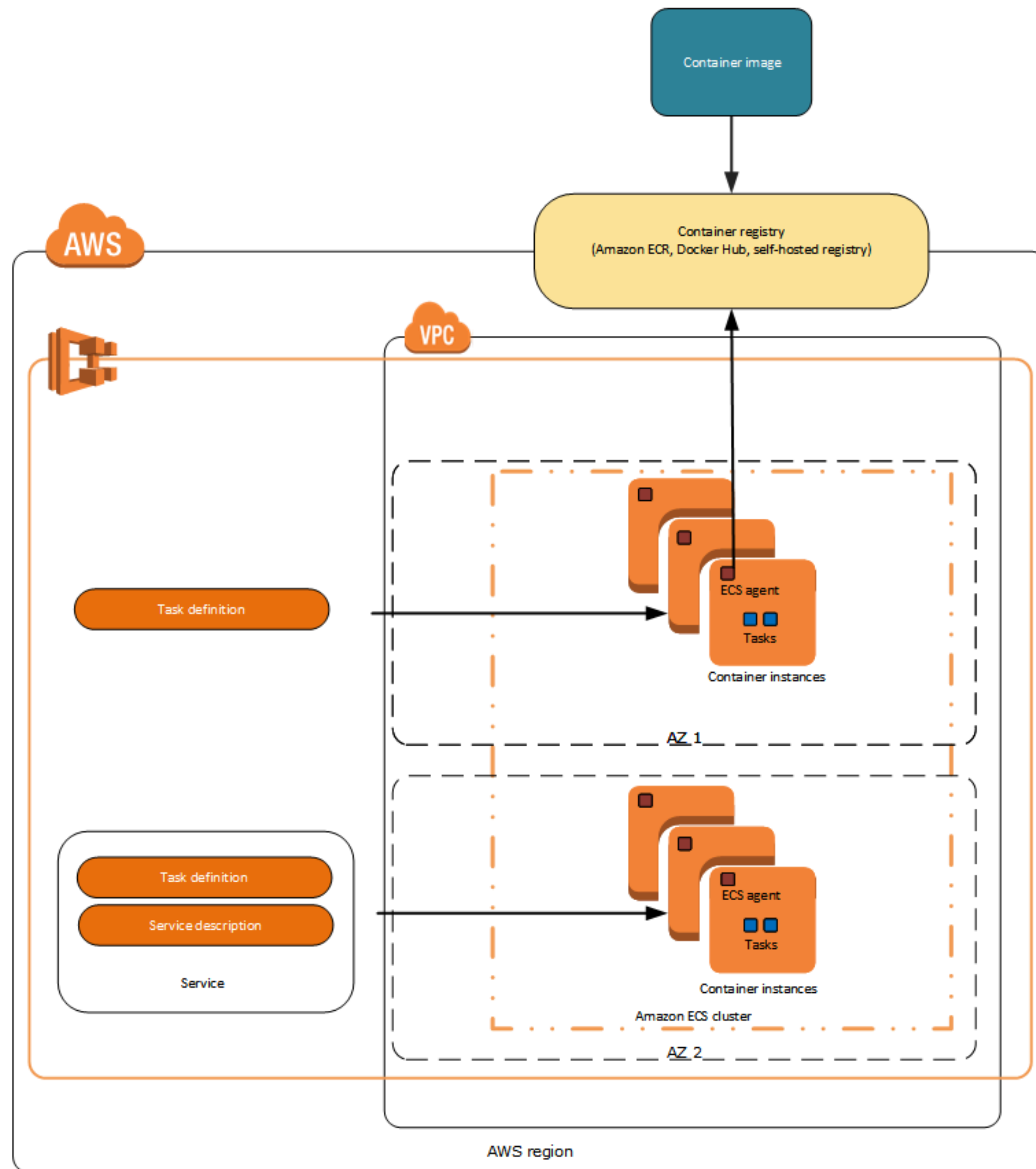




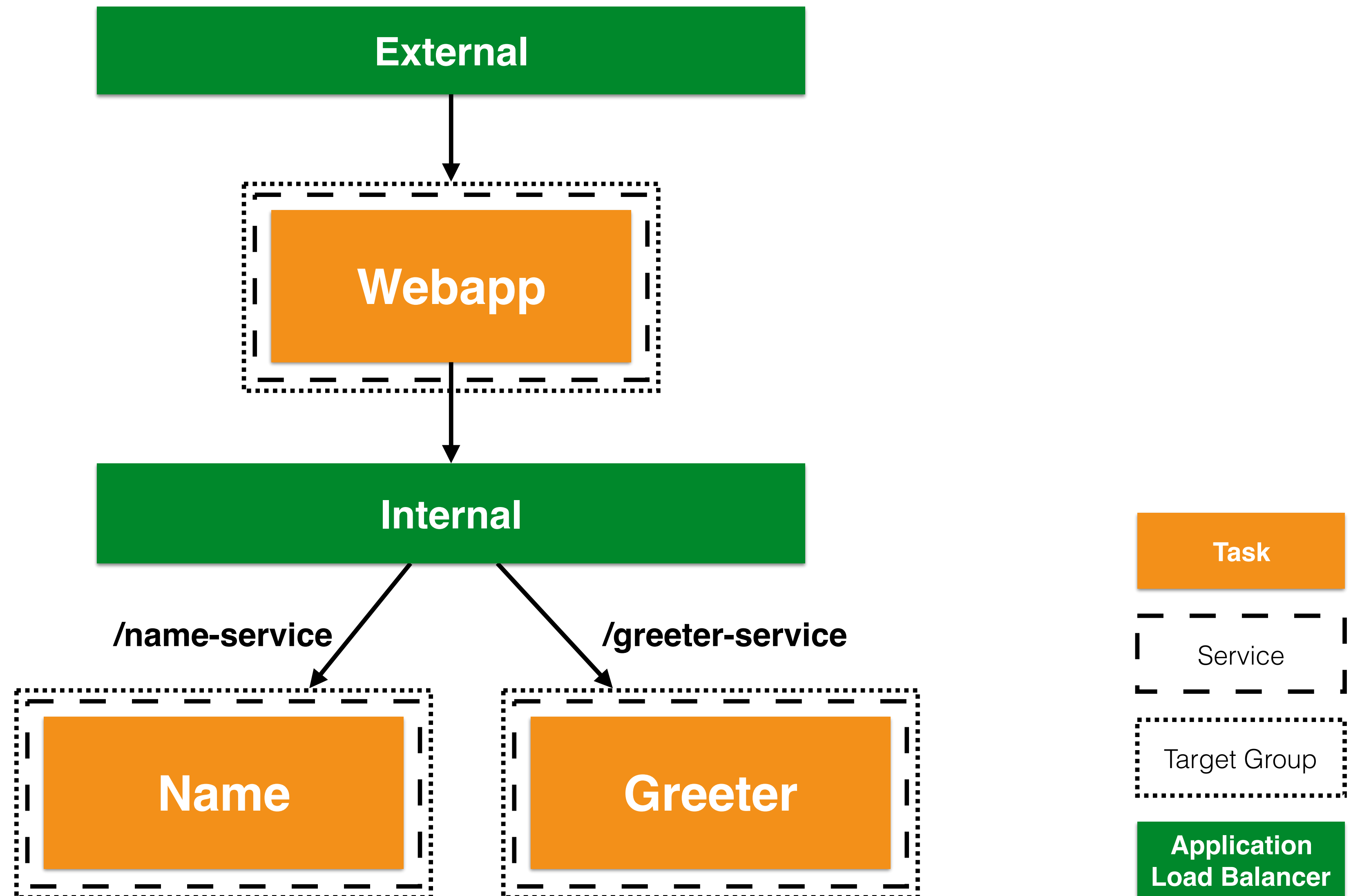


# Amazon EC2 Container Service

- 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, ...



# Amazon EC2 Container Service



Create target group

Actions

Filter: Search

	Name	Port	Protocol	VPC ID
<input type="checkbox"/>	greeter-target-group	80	HTTP	vpc-0d0c7
<input type="checkbox"/>	name-target-group	80	HTTP	vpc-0d0c7
<input checked="" type="checkbox"/>	webapp-target-group	80	HTTP	vpc-0d0c7

Target group: webapp-target-group

DescriptionTargetsHealth checksMonitoringTags

The load balancer starts routing requests to a newly registered target as soon as the re checks. If demand on your targets increases, you can register additional targets. If der

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

i

Essential

☒

i

Entry point

comma delimited: sh,-c

i

Command

comma delimited: echo,hello world

i

Working directory

/usr/app

i

Env Variables

Key	Value	
GREETER_SERVICE_	internal-alb-internal-4	x
GREETER_SERVICE_	80	x
GREETER_SERVICE_	greeter-service/	x
NAME_SERVICE_HO	internal-alb-internal-4	x
NAME_SERVICE_POI	80	x
NAME_SERVICE_PAT	name-service/	x
Add key	Add value	

i

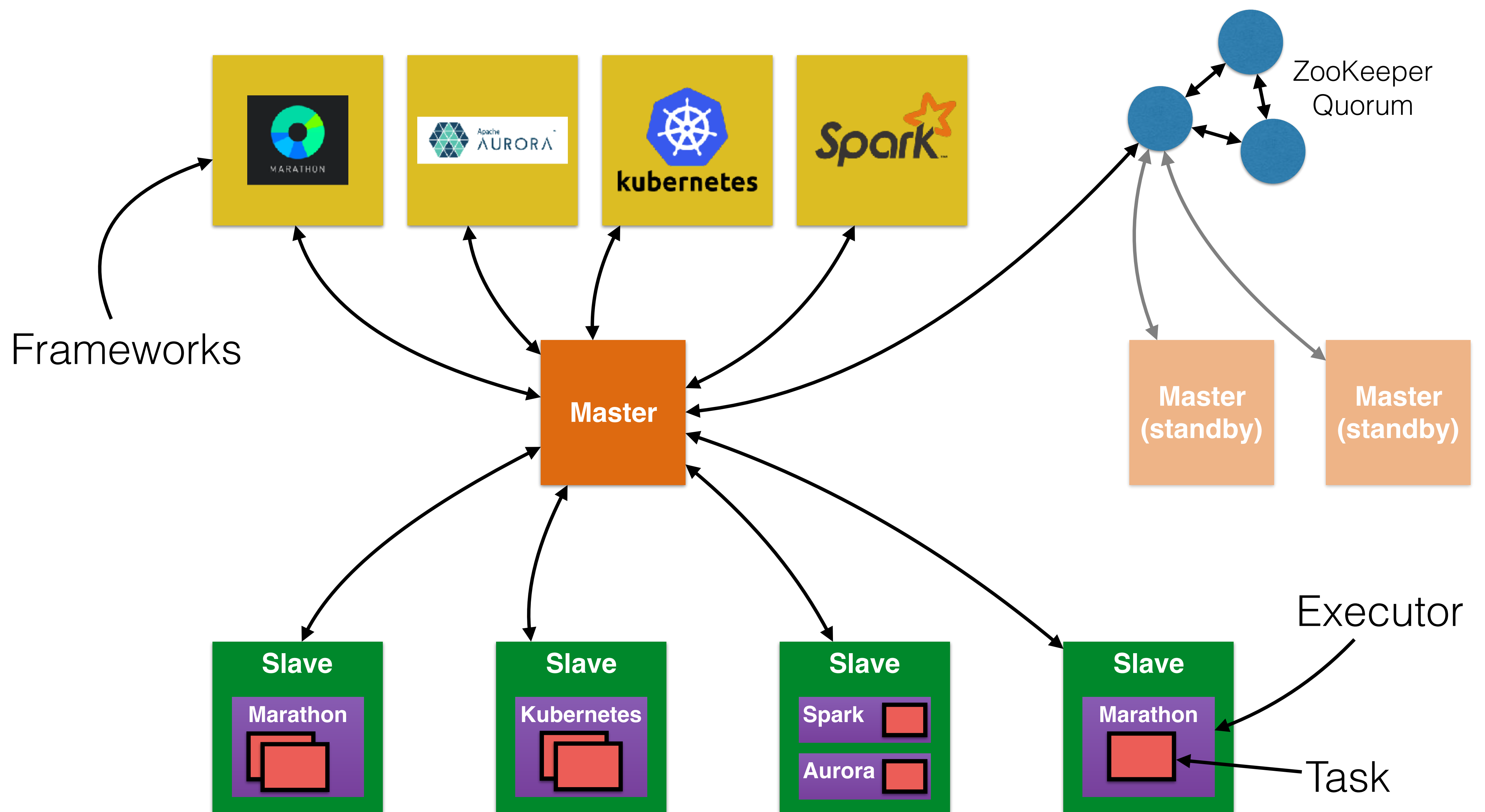


MESOSPHERE

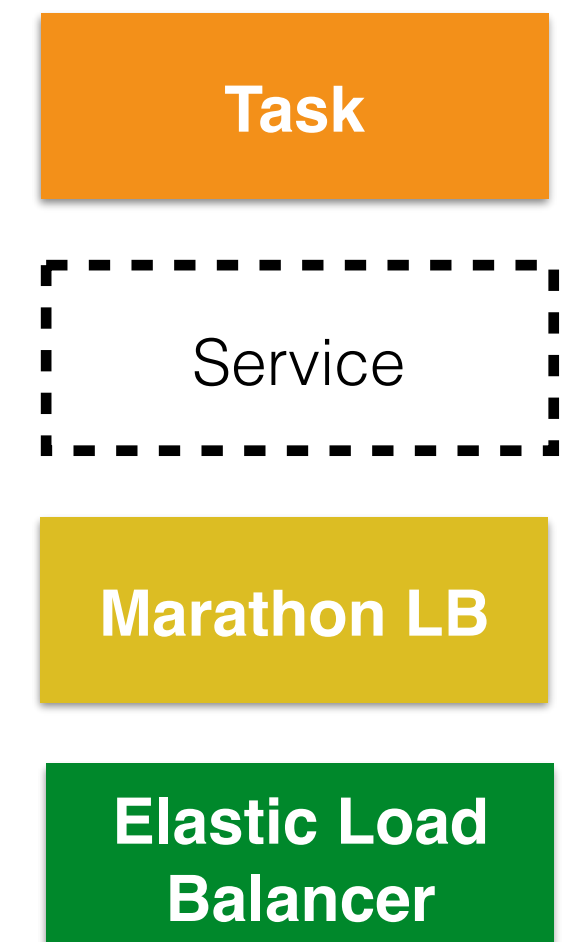
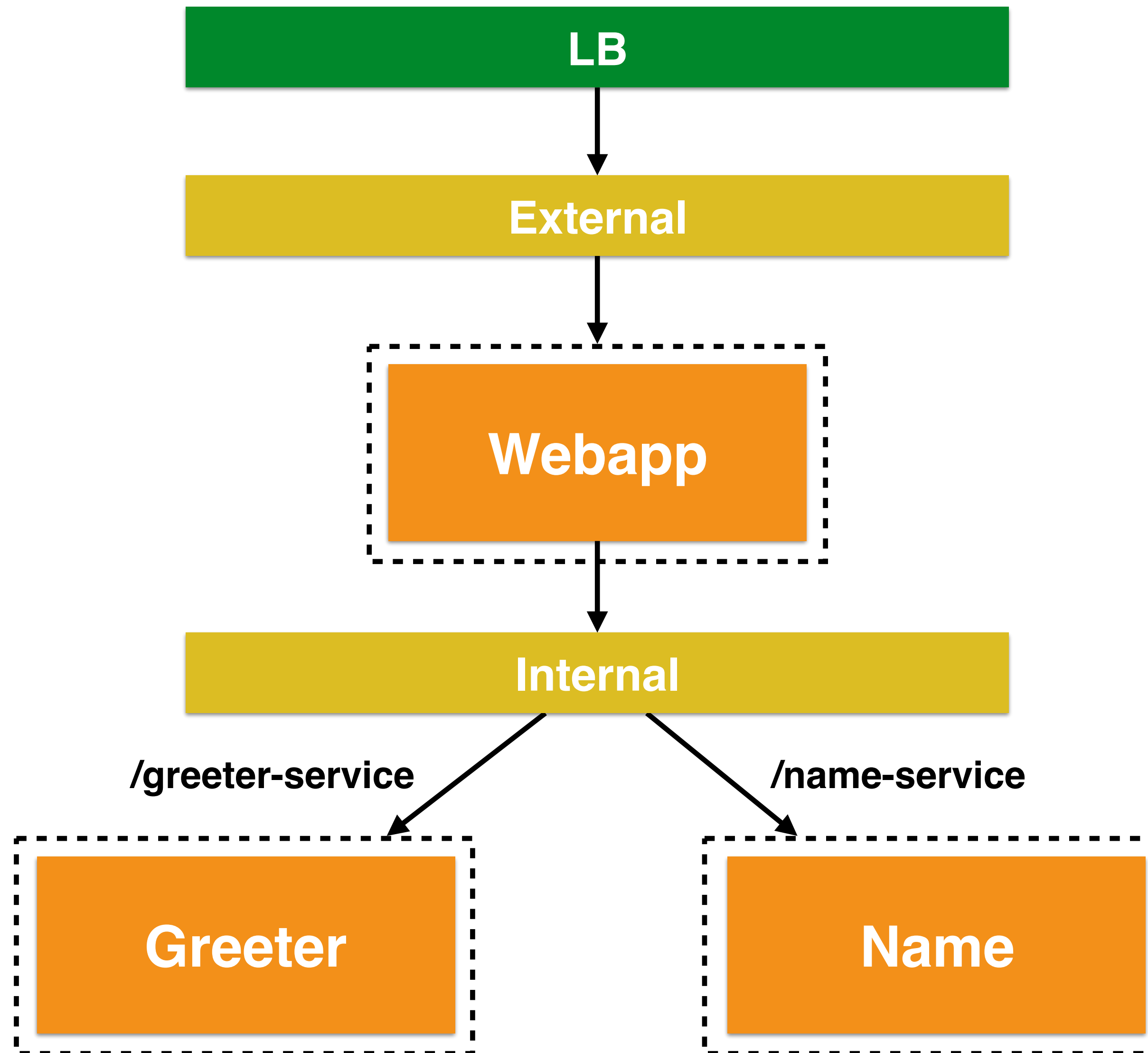
# DC/OS

- 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





# DC/OS



```
1 {
2   "marathon-lb": {
3     "name": "marathon-lb-internal",
4     "haproxy-group": "internal",
5     "bind-http-https": false,
6     "role": ""
7   }
8 }

1 {
2   "id": "/name",
3   "cpus": 1,
4   "mem": 1024,
5   "instances": 1,
6   "container": {
7     "type": "DOCKER",
8     "docker": {
9       "image": "arungupta/name-service:latest",
10      "network": "BRIDGE",
11      "portMappings": [
12        {
13          "hostPort": 0,
14          "containerPort": 8080,
15          "servicePort": 10001
16        }
17      ]
18    }
19  },
20  "labels": {
21    "HAPROXY_GROUP": "internal"
22  }
23 }

1 {
2   "id": "/greeter",
3   "cpus": 1,
4   "mem": 1024,
5   "instances": 1,
6   "container": {
7     "type": "DOCKER",
8     "docker": {
9       "image": "arungupta/greeter-service:latest",
10      "network": "BRIDGE",
11      "portMappings": [
12        {
13          "hostPort": 0,
14          "containerPort": 8080,
15          "servicePort": 10000
16        }
17      ]
18    }
19  },
20  "labels": {
21    "HAPROXY_GROUP": "internal"
22  }
23 }

1 {
2   "id": "/webapp",
3   "cpus": 1,
4   "mem": 1024,
5   "instances": 1,
6   "container": {
7     "type": "DOCKER",
8     "docker": {
9       "image": "arungupta/webapp-service:latest",
10      "network": "BRIDGE",
11      "portMappings": [
12        {
13          "hostPort": 0,
14          "containerPort": 8080,
15          "servicePort": 10001,
16          "protocol": "tcp"
17        }
18      ]
19    }
20  },
21  "env": {
22    "NAME_SERVICE_HOST": "marathon-lb-internal.marathon.mesos",
23    "NAME_SERVICE_PORT": "10001",
24    "NAME_SERVICE_PATH": "/",
25    "GREETER_SERVICE_HOST": "marathon-lb-internal.marathon.mesos",
26    "GREETER_SERVICE_PORT": "10000",
27    "GREETER_SERVICE_PATH": "/"
28  },
29  "labels": {
30    "HAPROXY_0_VHOST": "dcos-PublicSlaveLo-1B7KS7V00I0Y6-1504556418.us-west-1.elb.amazonaws.com",
31    "HAPROXY_GROUP": "external"
32  }
33 }
```

The diagram illustrates the configuration dependencies between four services in a Mesos environment:

- marathon-lb-internal** (leftmost configuration) defines the load balancer configuration, including the `name` (`marathon-lb-internal`), `haproxy-group` (`internal`), and `role` (`internal`).
- name** (second configuration) is a service that depends on `marathon-lb-internal`. It uses the `marathon-lb-internal` configuration to set the `NAME_SERVICE_HOST` and `NAME_SERVICE_PORT` in its `env` section.
- greeter** (third configuration) is a service that depends on `marathon-lb-internal`. It uses the `marathon-lb-internal` configuration to set the `GREETER_SERVICE_HOST` and `GREETER_SERVICE_PORT` in its `env` section.
- webapp** (rightmost configuration) is a service that depends on `marathon-lb-internal`. It uses the `marathon-lb-internal` configuration to set the `NAME_SERVICE_HOST` and `NAME_SERVICE_PORT` in its `env` section.

Red circles highlight the specific configuration values used in the `env` section of the services:

- `marathon-lb-internal.marathon.mesos` (used for `NAME_SERVICE_HOST` and `GREETER_SERVICE_HOST`)
- `10001` (used for `NAME_SERVICE_PORT` and `webapp` `servicePort`)
- `10000` (used for `GREETER_SERVICE_PORT`)

# References

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