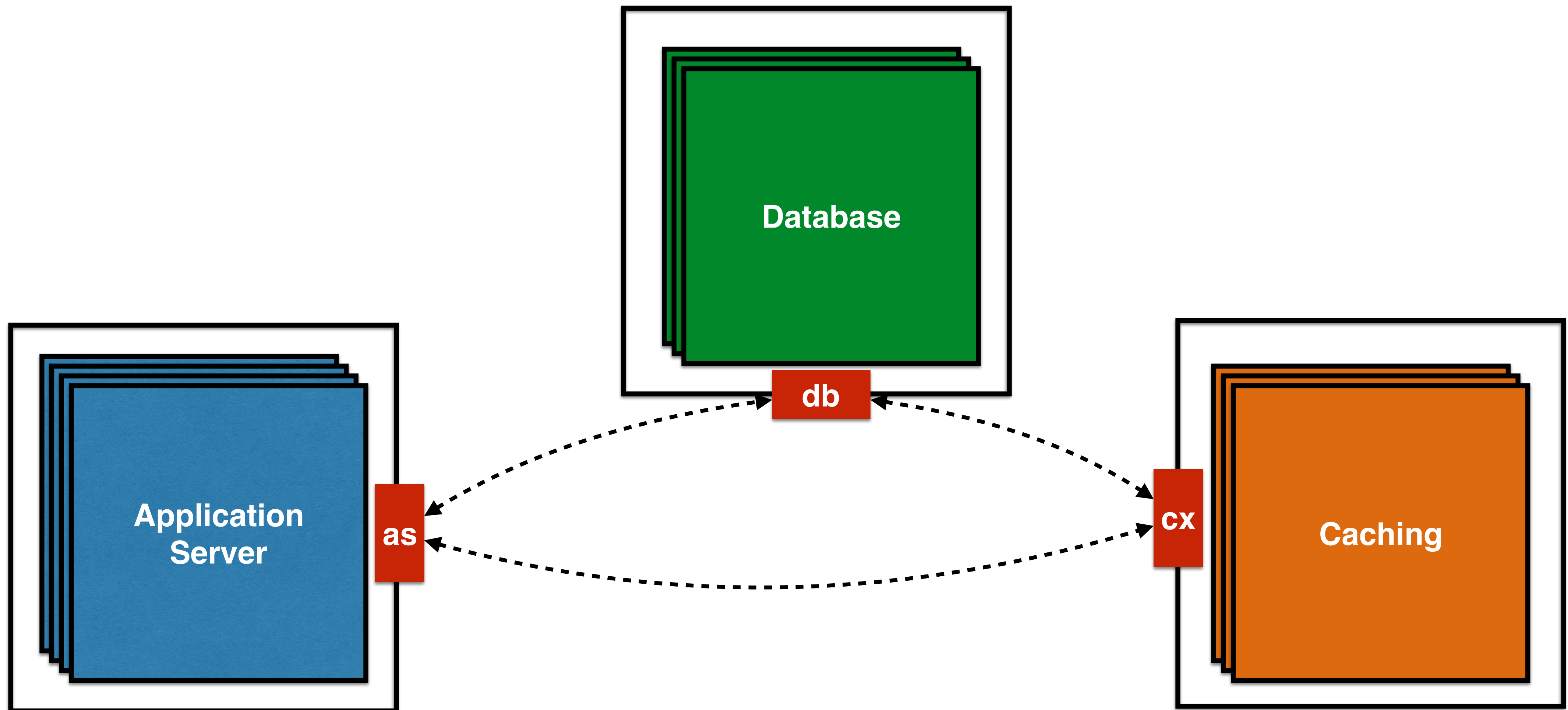
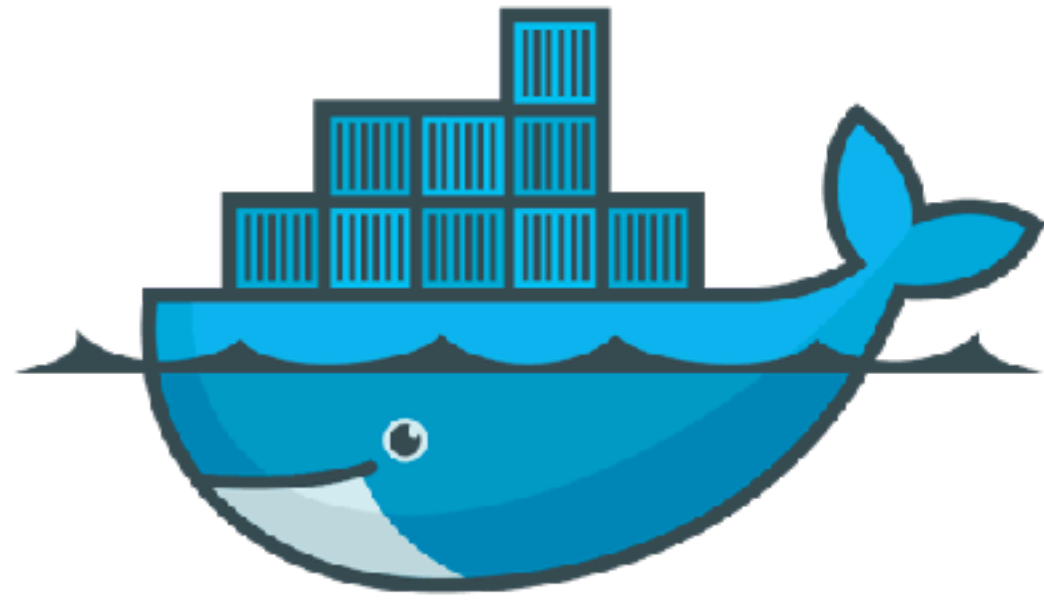


Service Discovery in Container Orchestration

Arun Gupta, @arungupta

Why Service Discovery?

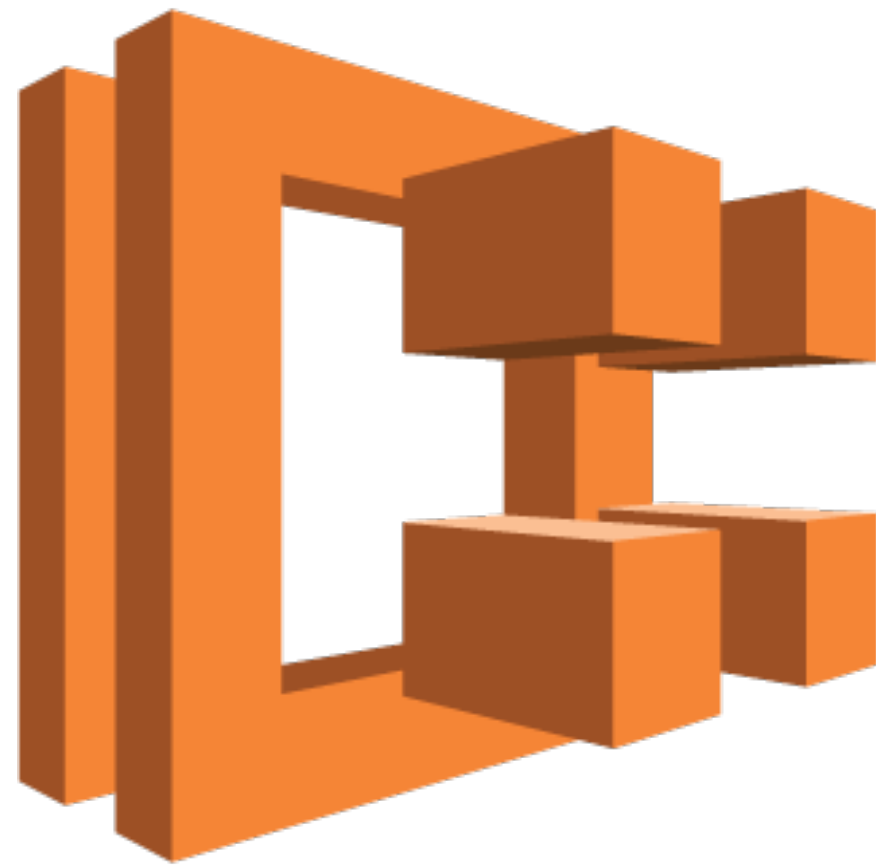




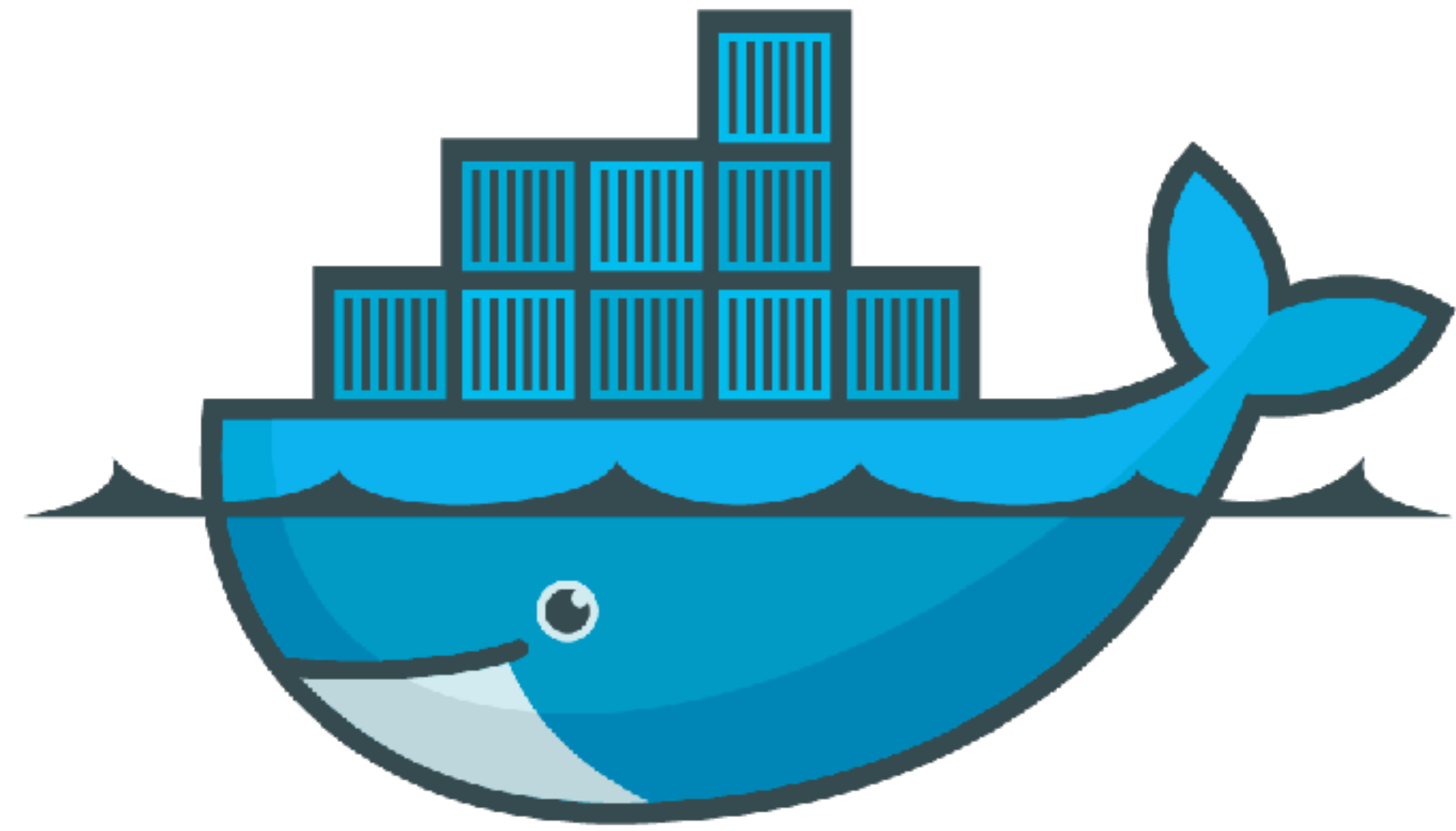
docker



kubernetes

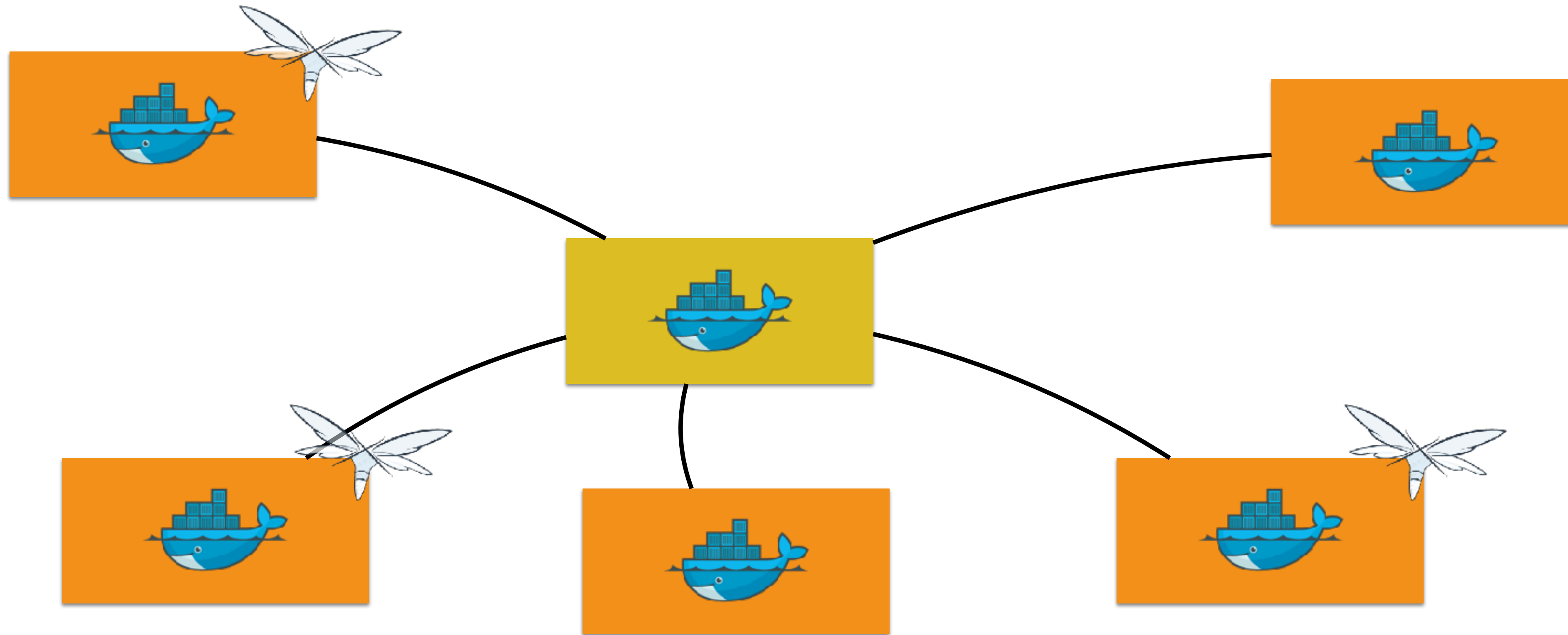


MESOSPHERE



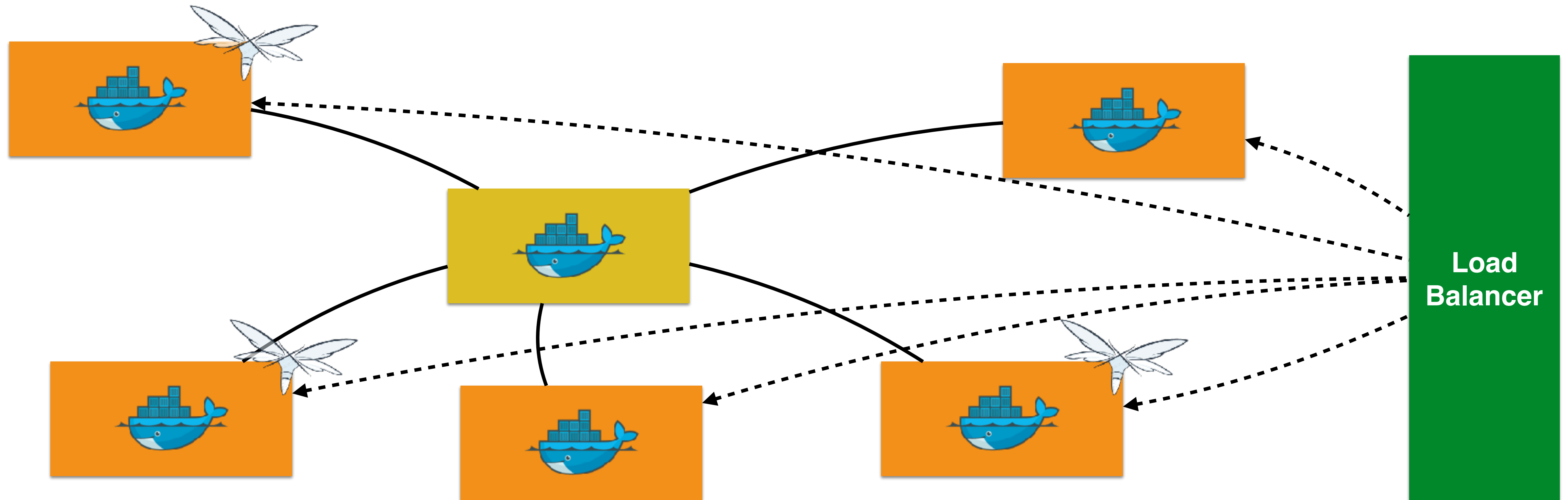
docker

Swarm-mode: Replicated Service



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

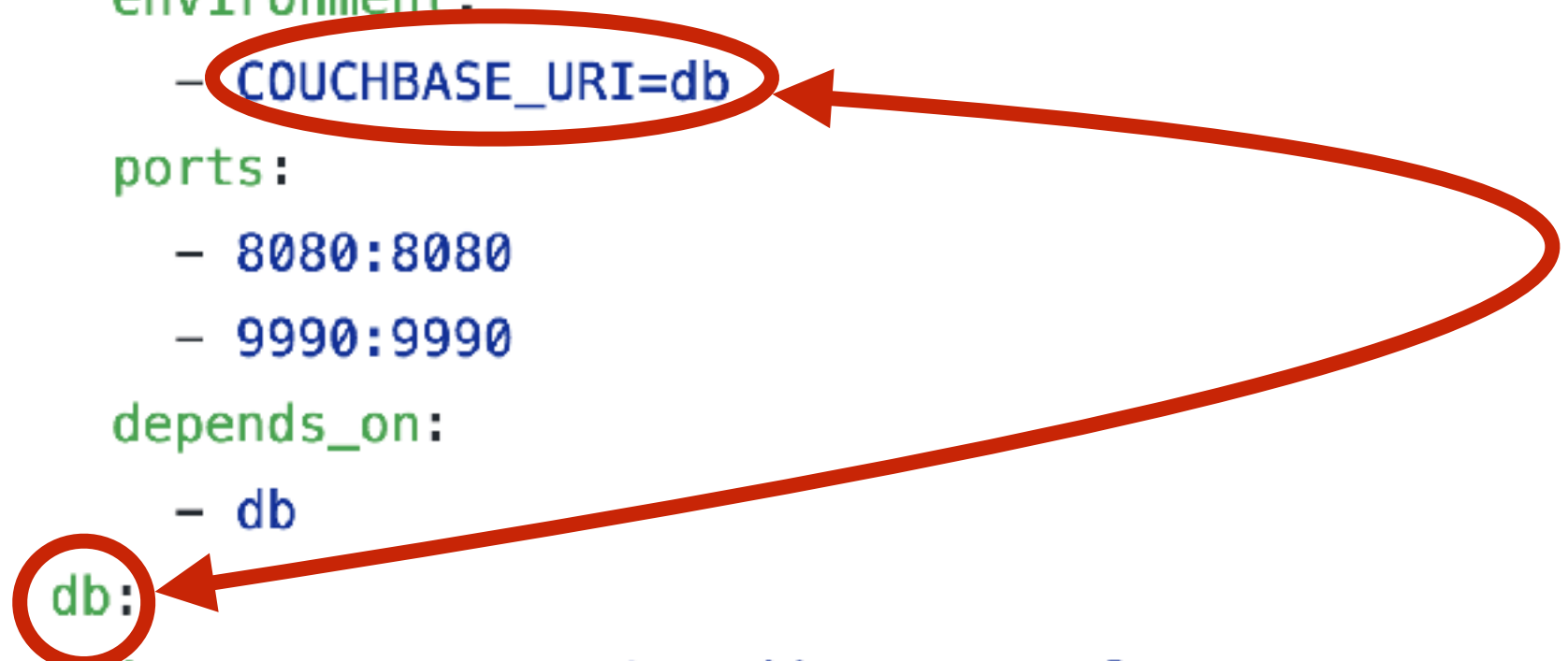
Swarm-mode: Routing Mesh



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


Service Discovery with Docker

```
1  version: '3'
2  services:
3    web:
4      image: arungupta/couchbase-javaee:travel
5      environment:
6        - COUCHBASE_URI=db
7      ports:
8        - 8080:8080
9        - 9990:9990
10     depends_on:
11       - db
12     db:
13       image: arungupta/couchbase:travel
14       ports:
15         - 8091:8091
16         - 8092:8092
17         - 8093:8093
18         - 11210:11210
19
```

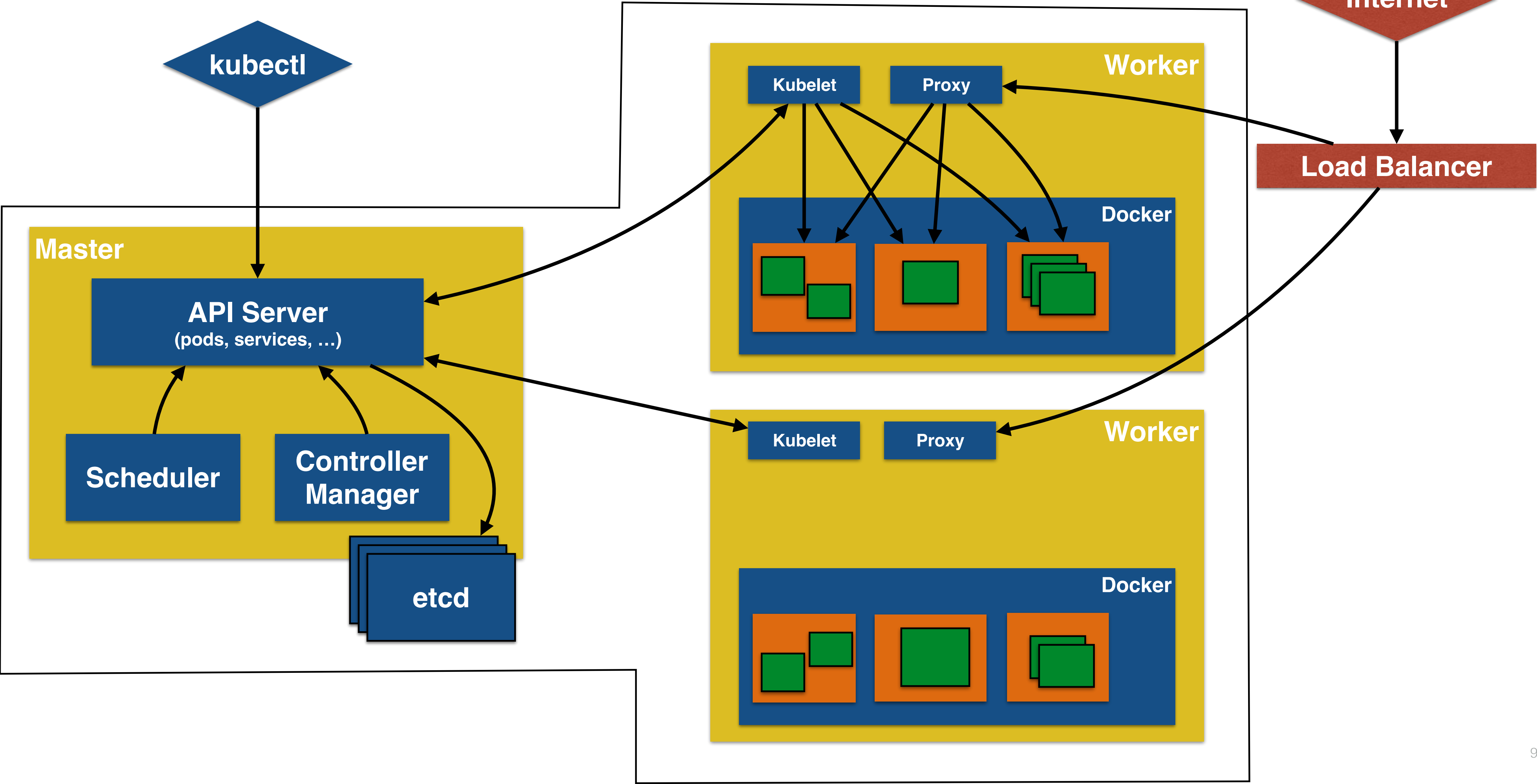


```
docker stack deploy --compose-file=docker-compose.yml webapp
```



kubernetes

Kubernetes Cluster



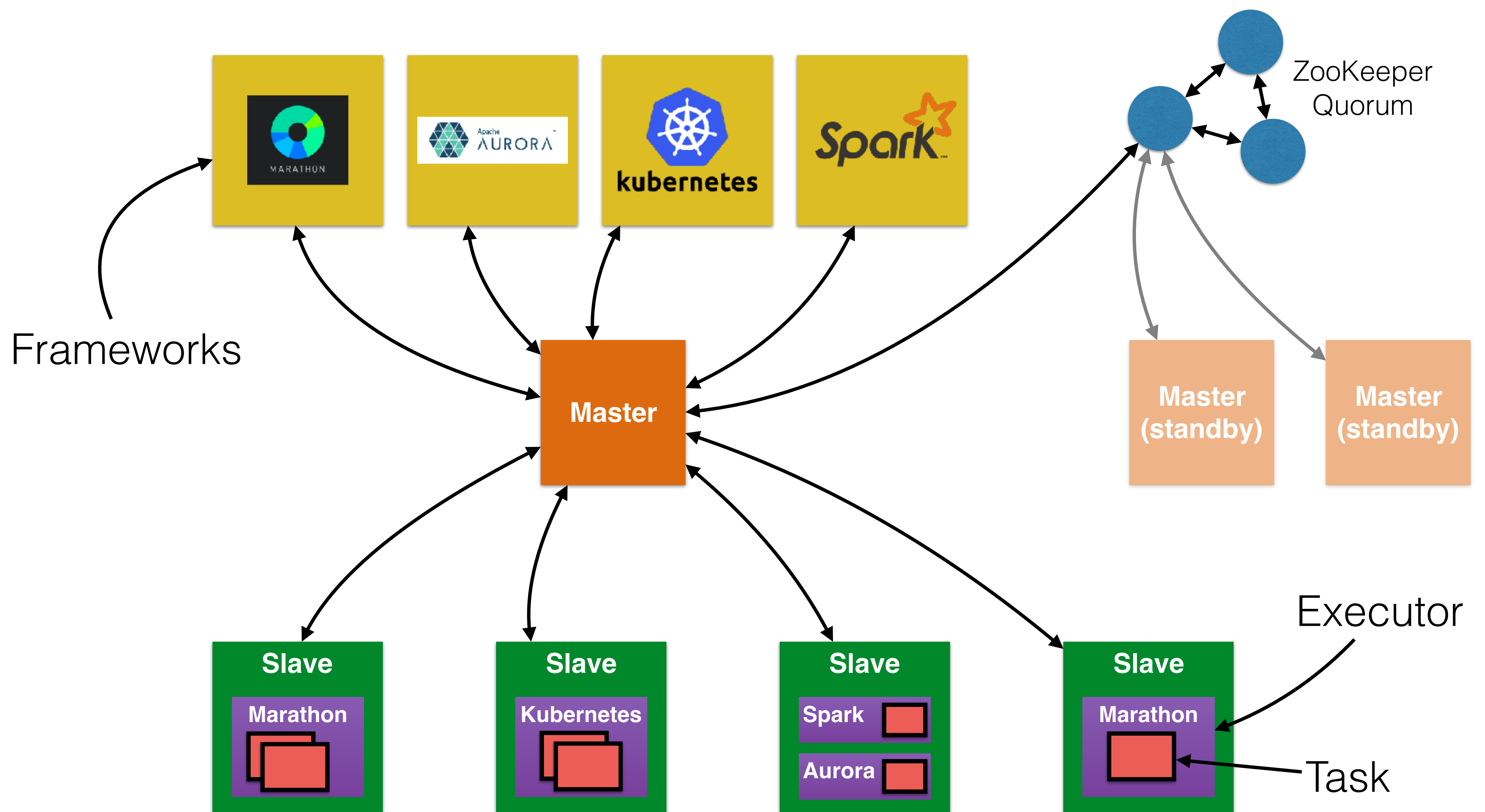
Kubernetes Configuration File

```
1  apiVersion: v1
2  kind: Service
3  metadata:
4  name: couchbase-service
5  spec:
6  selector:
7  app: couchbase-rs-pod
8  ports:
9    - name: admin
10     port: 8091
11    - name: views
12     port: 8092
13    - name: query
14     port: 8093
15    - name: memcached
16     port: 11210
17  ---
18  apiVersion: extensions/v1beta1
19  kind: ReplicaSet
20  metadata:
21  name: couchbase-rs
22  spec:
23  replicas: 1
24  template:
25    metadata:
26    labels:
27    app: couchbase-rs-pod
28    spec:
29    containers:
30    - name: couchbase
31      image: arungupta/couchbase:travel
32      ports:
33        - containerPort: 8091
34        - containerPort: 8092
35        - containerPort: 8093
36        - containerPort: 11210
37  ---
38  apiVersion: extensions/v1beta1
39  kind: ReplicaSet
40  metadata:
41  name: wildfly-rs
42  labels:
43  name: wildfly
44  spec:
45  replicas: 1
46  template:
47    metadata:
48    labels:
49    name: wildfly
50  spec:
51  containers:
52  - name: wildfly-rs-pod
53    image: arungupta/wildfly-couchbase-javaee:travel
54    env:
55    name: COUCHBASE_URI
56    value: couchbase-service
57  ports:
58  - containerPort: 8080
59  ---
```

```
graph LR
    subgraph Service [Service]
        S_name["name: couchbase-service"]
    end
    subgraph Couchbase_RS [Couchbase ReplicaSet]
        C_label["app: couchbase-rs-pod"]
    end
    subgraph Wildfly_RS [Wildfly ReplicaSet]
        W_env["name: COUCHBASE_URI<br/>value: couchbase-service"]
    end
    S_name --> C_label
    S_name --> W_env
    C_label --> W_env
```



MESOSPHERE




```

1  {
2    "id":"/webapp",
3    "apps":[
4      {
5        "id":"database",
6        "cpus":4,
7        "mem":4096,
8        "instances":1,
9        "container":{
10         "type":"DOCKER",
11         "docker":{
12           "image":"arungupta/couchbase:travel",
13           "network":"USER"
14         }
15       },
16       "ipAddress":{
17         "networkName":"dcos"
18       }
19     ],

```

```

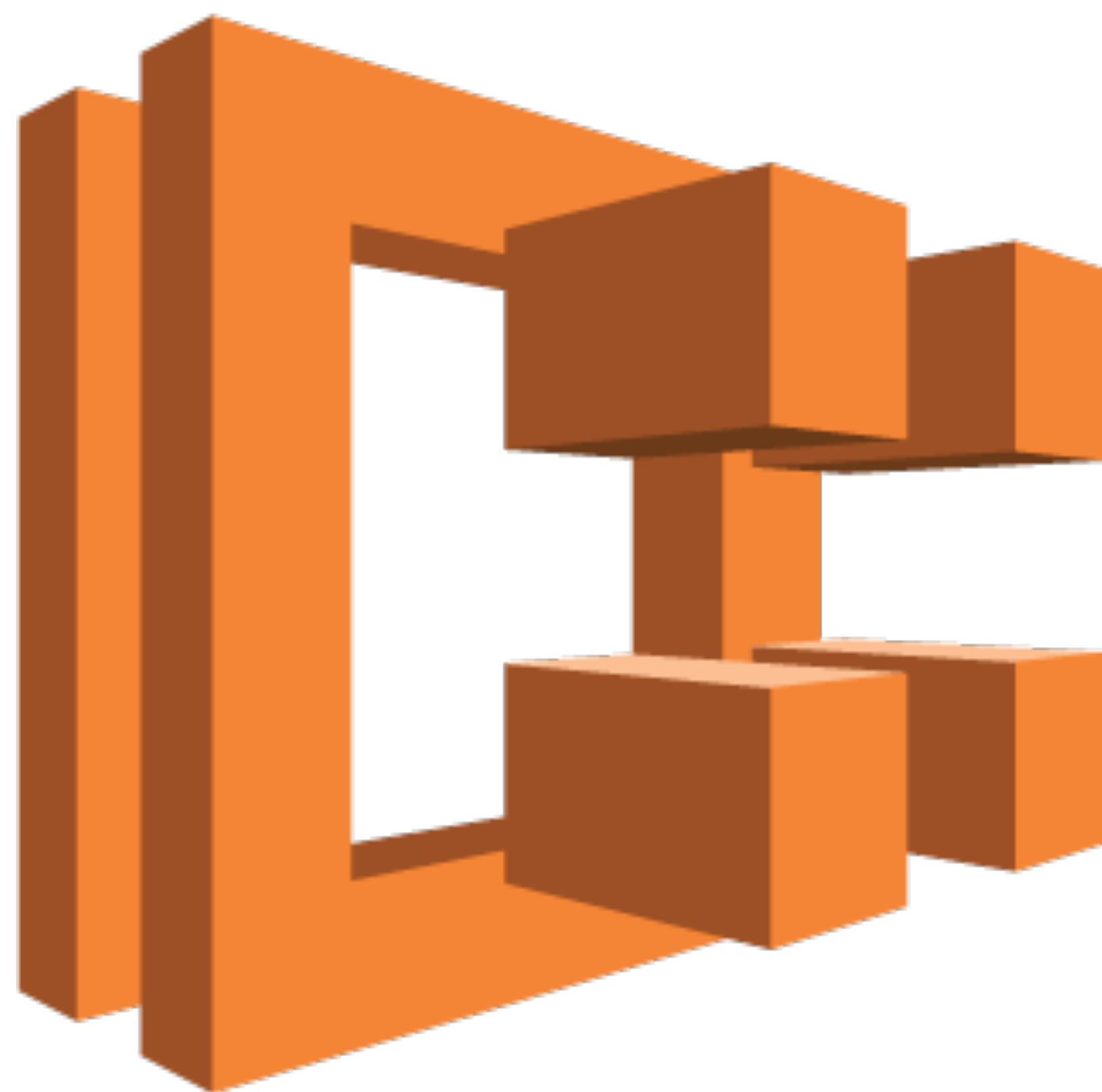
45   "env":{
46     "COUCHBASE_URI":"database-webapp.marathon.containerip.dcos.thisdcos.directory",
47   },
48   "labels":{
49     "HAPROXY_0_VHOST":"DCOS-PublicSlaveLo-18F0F3VUV2GFU-1481429739.us-west-1.elb.amazonaws.com",
50     "HAPROXY_GROUP":"external"
51   }
52 }
53 ]
54 }

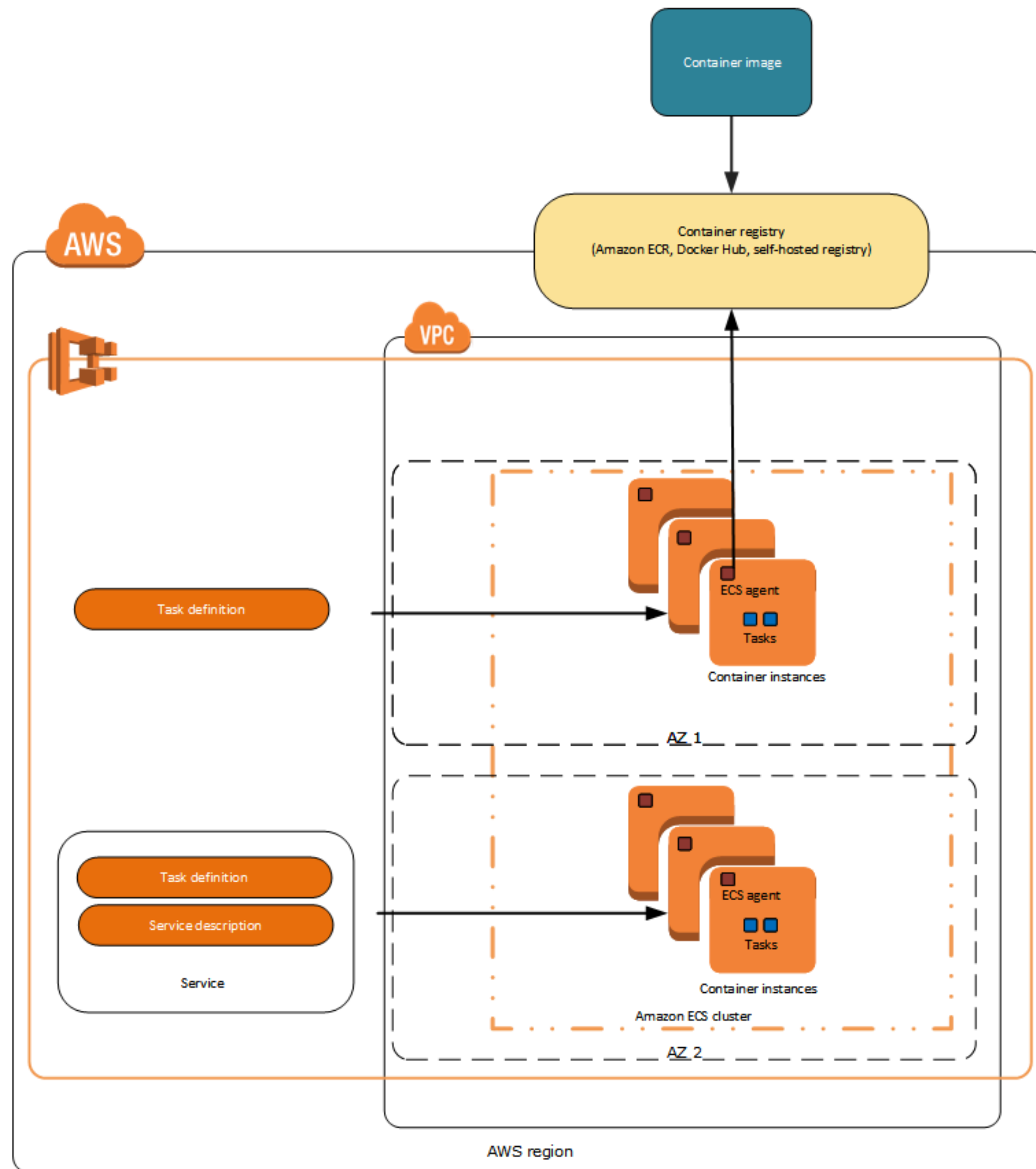
```

```

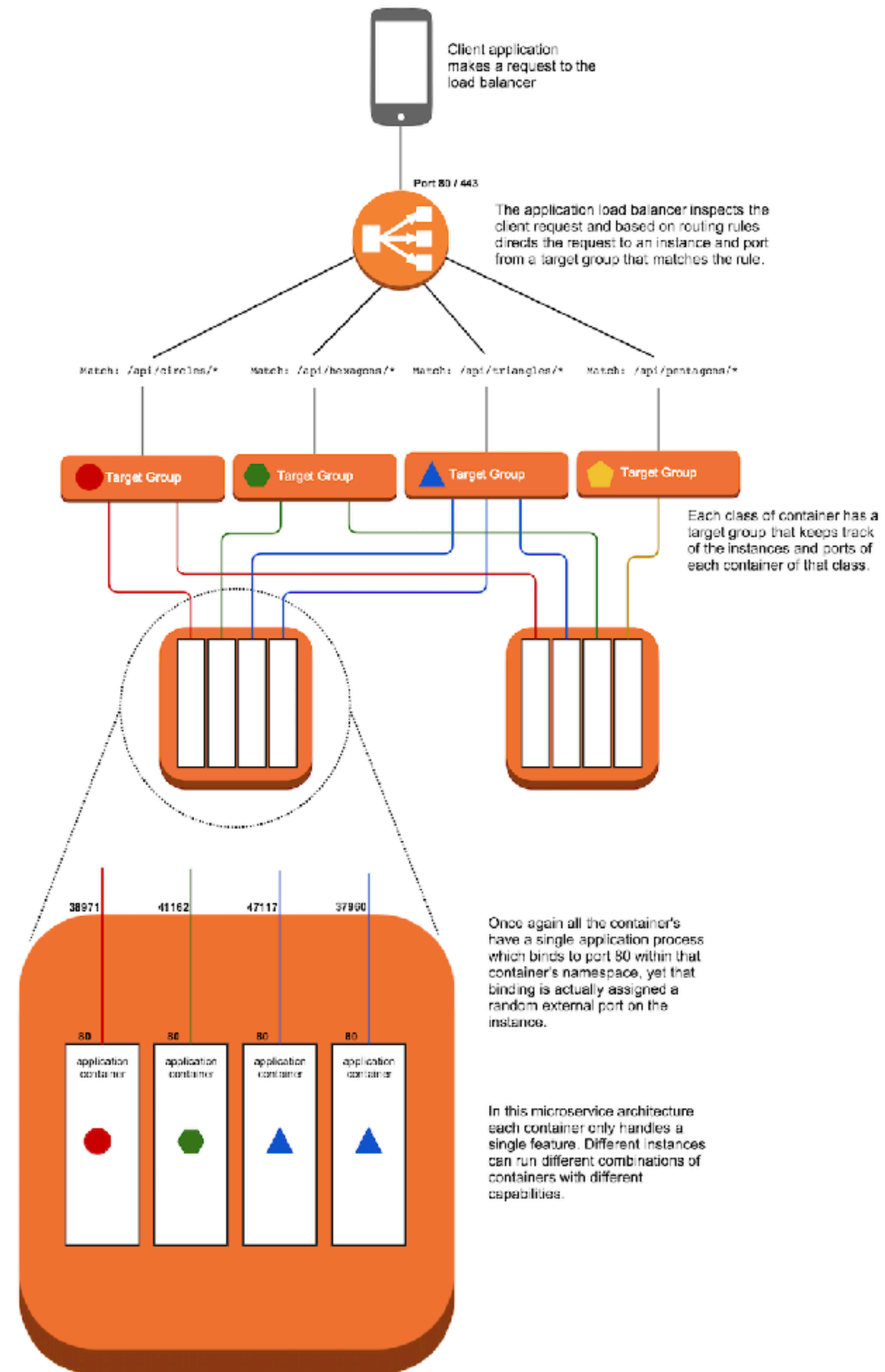
20 {
21   "id":"web",
22   "dependencies":[
23     "/webapp/database"
24   ],
25   "cpus":2,
26   "mem":4096,
27   "instances":1,
28   "container":{
29     "type":"DOCKER",
30     "docker":{
31       "image":"arungupta/wildfly-couchbase-javaee:travel",
32       "network":"USER",
33       "portMappings":[
34         {
35           "hostPort":0,
36           "containerPort":8080,
37           "protocol":"tcp"
38         }
39       ]
40     }
41   },

```





Feature	Classic Load Balancer	Application Load Balancer
Protocols	HTTP, HTTPS, TCP, SSL	HTTP, HTTPS
Platforms	EC2-Classic, EC2-VPC	EC2-VPC
Sticky sessions (cookies)	✓	load balancer generated
Idle connection timeout	✓	✓
Connection draining	✓	✓
Cross-zone load balancing †	✓	Always enabled
Health checks † †	✓	Improved
CloudWatch metrics	✓	Improved
Access logs	✓	Improved
Host-based routing		✓
Path-based routing		✓
Route to multiple ports on a single instance		✓
HTTP/2 support		✓
Websockets support		✓
Load balancer deletion protection		✓



References

- Docker: docker.io
- Kubernetes: kubernetes.io
- DC/OS: dcos.io
- ECS: aws.amazon.com/ecs
- Slides: github.com/arun-gupta/docker-java/tree/master/slides