# Service Discovery and Scalability using Docker Engine and Kubernetes

Arun Gupta, @arungupta

github.com/javaee-samples/docker-java/tree/master/slides

Couchbase

# Docker for Mac/Windows

- Native application and UI
- Auto update capability
- No additional software required, e.g. VirtualBox
  - OSX: xhyve VM using Hypervisor.framework
  - Windows: Hyper-V VM
- Better networking and filesystem mounting/notification
- Download: docker.com/getdocker
- Requires Yosemite 10.10+ or Windows 10 64-bit

# Docker for AWS/Azure


- **Amazon**: CloudFormation template, Integrated with Autoscaling, ELB, and EBS
- **Azure**: Integrated with VM Scale Sets for autoscaling, Azure Load Balancer, Azure Storage
- Sign up at beta.docker.com (restricted availability)

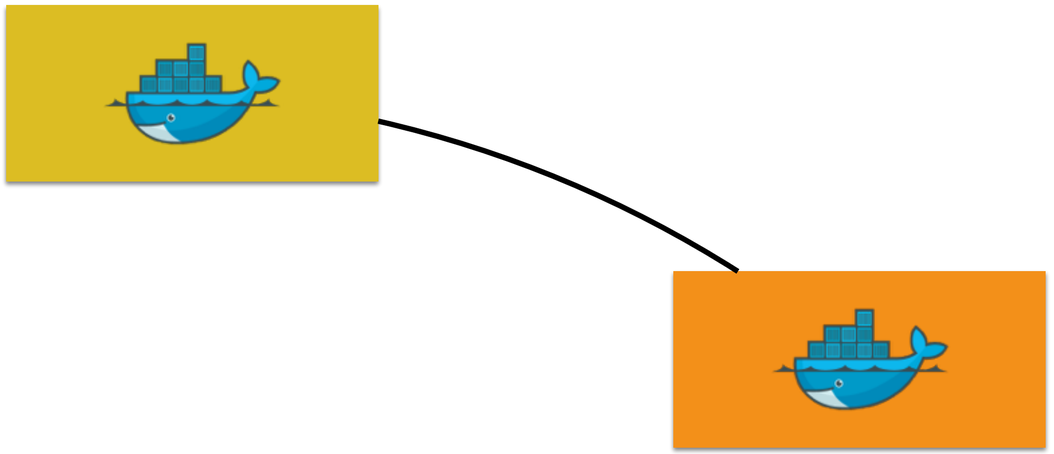
# Swarm Mode Overview

- New in 1.12
- Natively managing a cluster of Docker Engines called a Swarm
- Use Docker CLI to create a swarm, deploy apps, and manage swarm
- Reconciles “desired” vs “actual” state
- Uses SwarmKit
- Backwards compatible

# Swarm Mode: Initialize

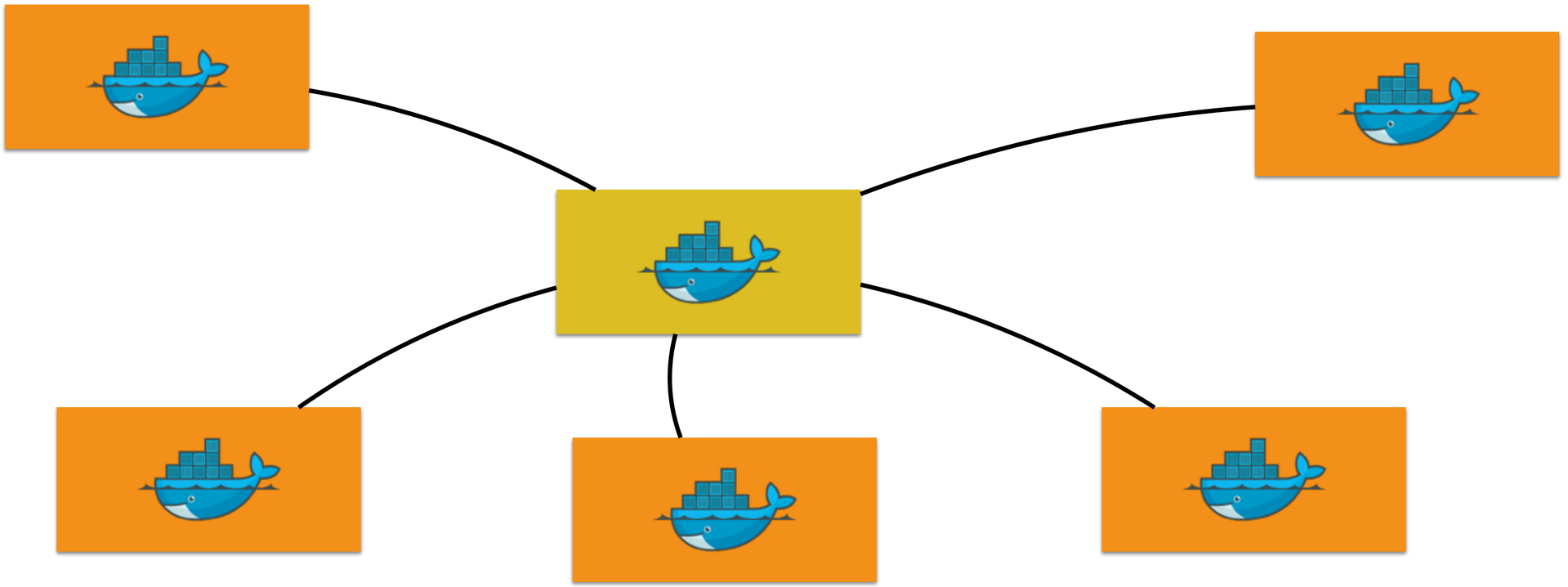
```
docker swarm init --listen-addr <ip>:2377 --secret <SECRET>
```

# Swarm Mode: Add Worker



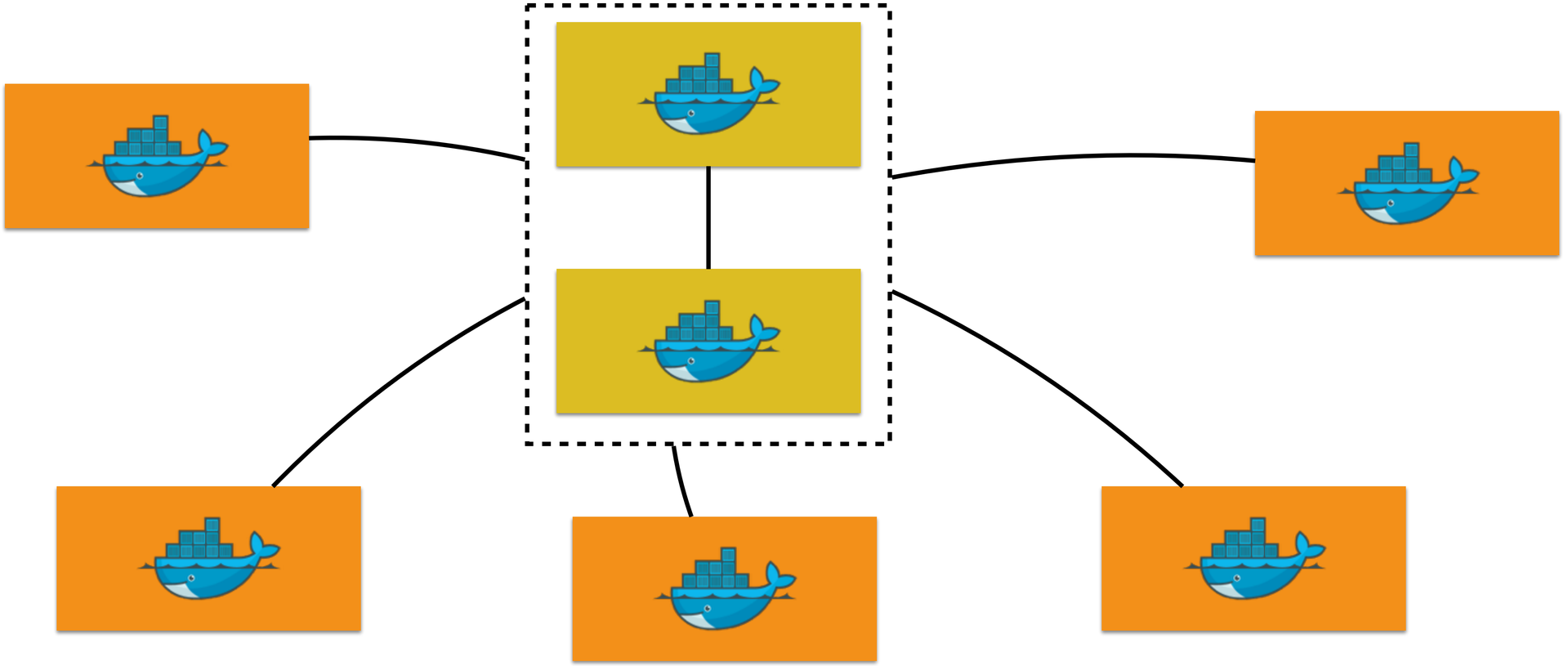
```
docker swarm join --secret <SECRET> <manager>:2377
```

# Swarm Mode: Add More Workers



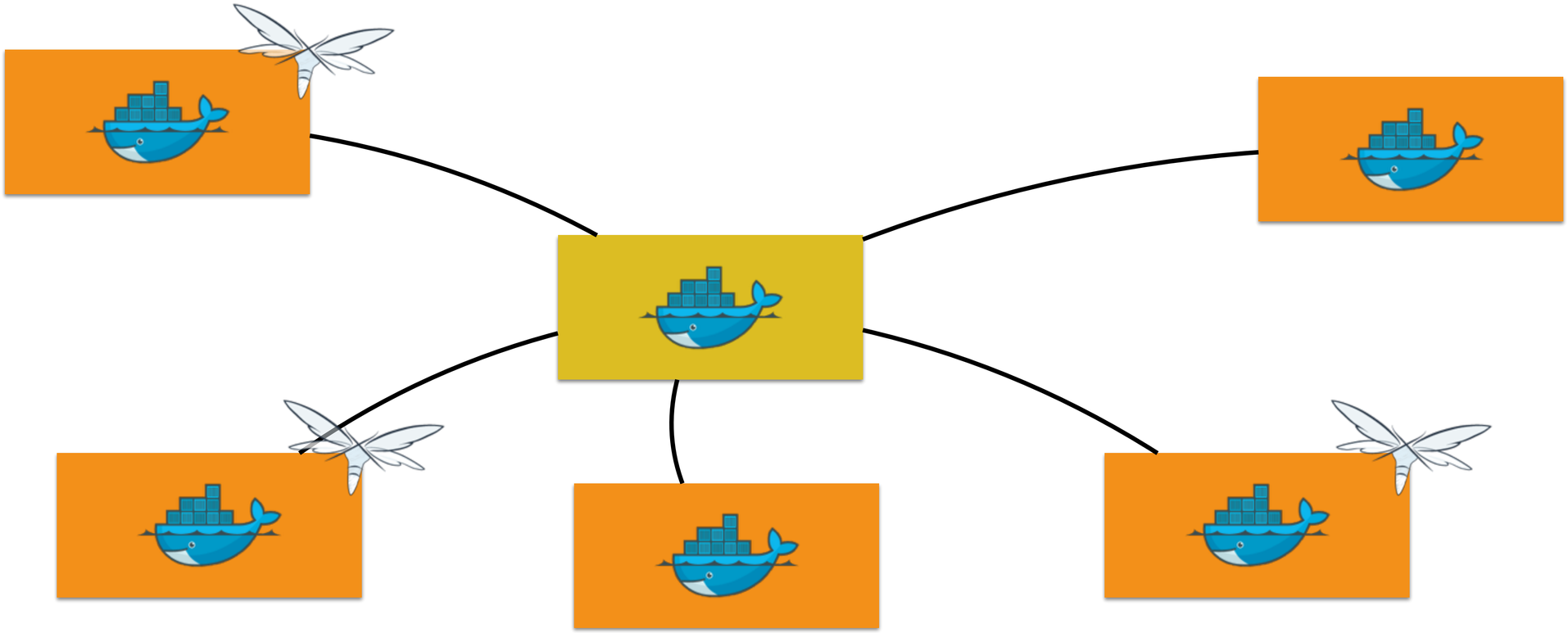
```
docker swarm join --secret <SECRET> <manager>:2377
```

# Swarm Mode: Primary/Secondary Master



```
docker swarm join --manager --secret <SECRET> --listen-addr
<master2>:2377 <master1>:2377
```

# Swarm Mode: Replicated Service

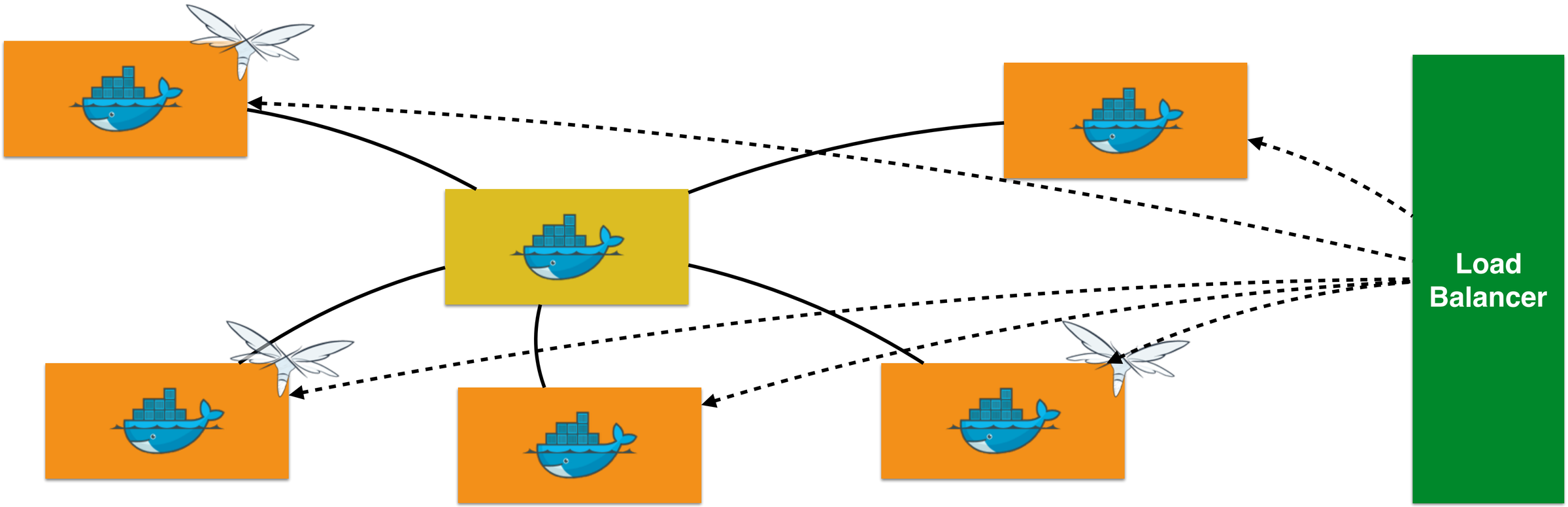


```
docker service create --replicas 3 --name web jboss/wildfly
```

# Swarm Mode - Routing Mesh

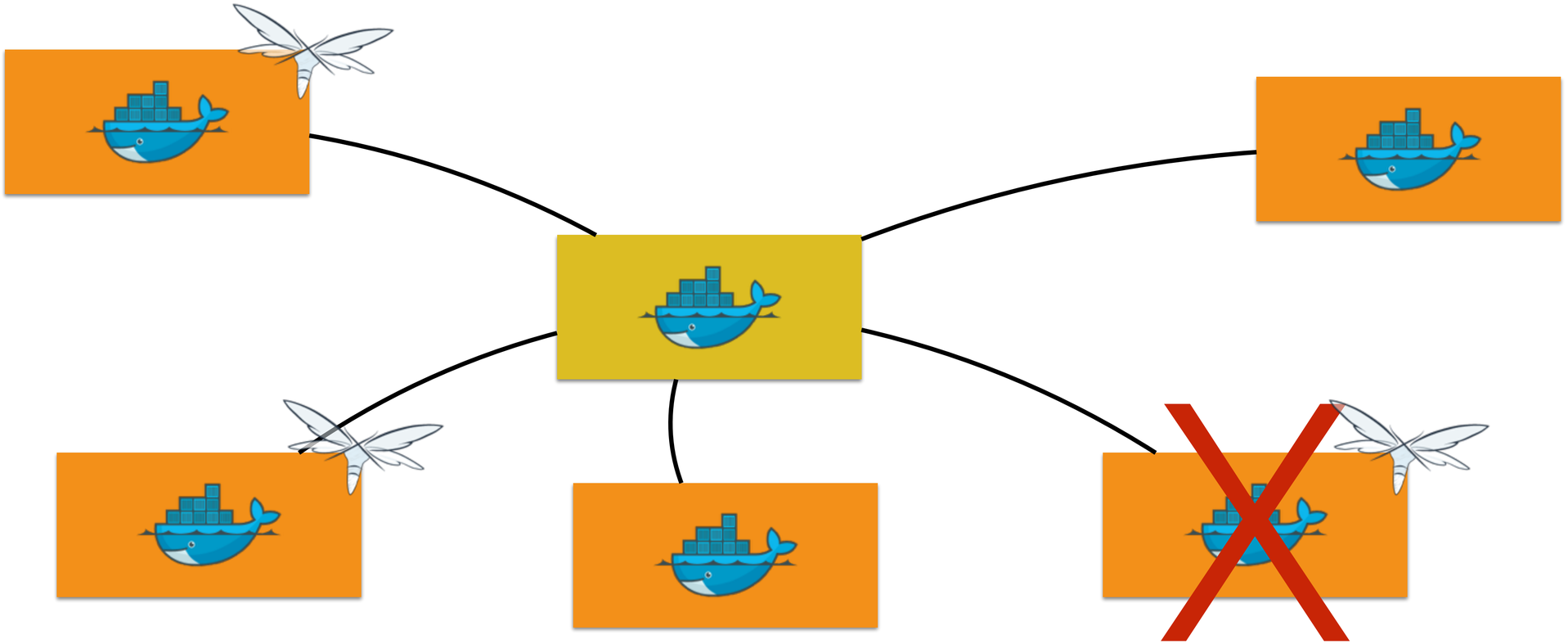
- Load balancers are host-aware, not container-aware
- Swarm mode introduces container-aware routing mesh
- Reroutes traffic from any host to a container
  - Reserves a Swarm-wide ingress port
  - Uses DNS-based service discovery

# Swarm Mode: Routing Mesh

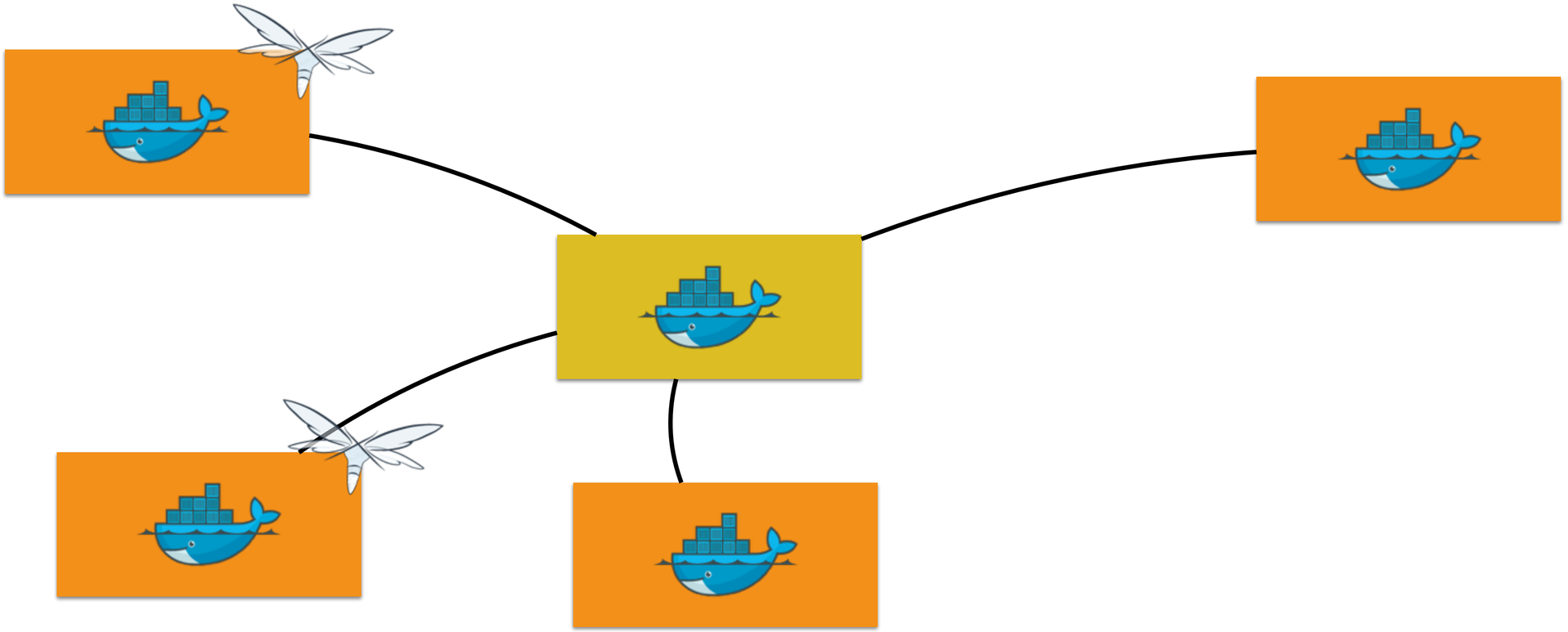



```
docker service create --replicas 3 --name web -p 8080:8080 jboss/
wildfly
```

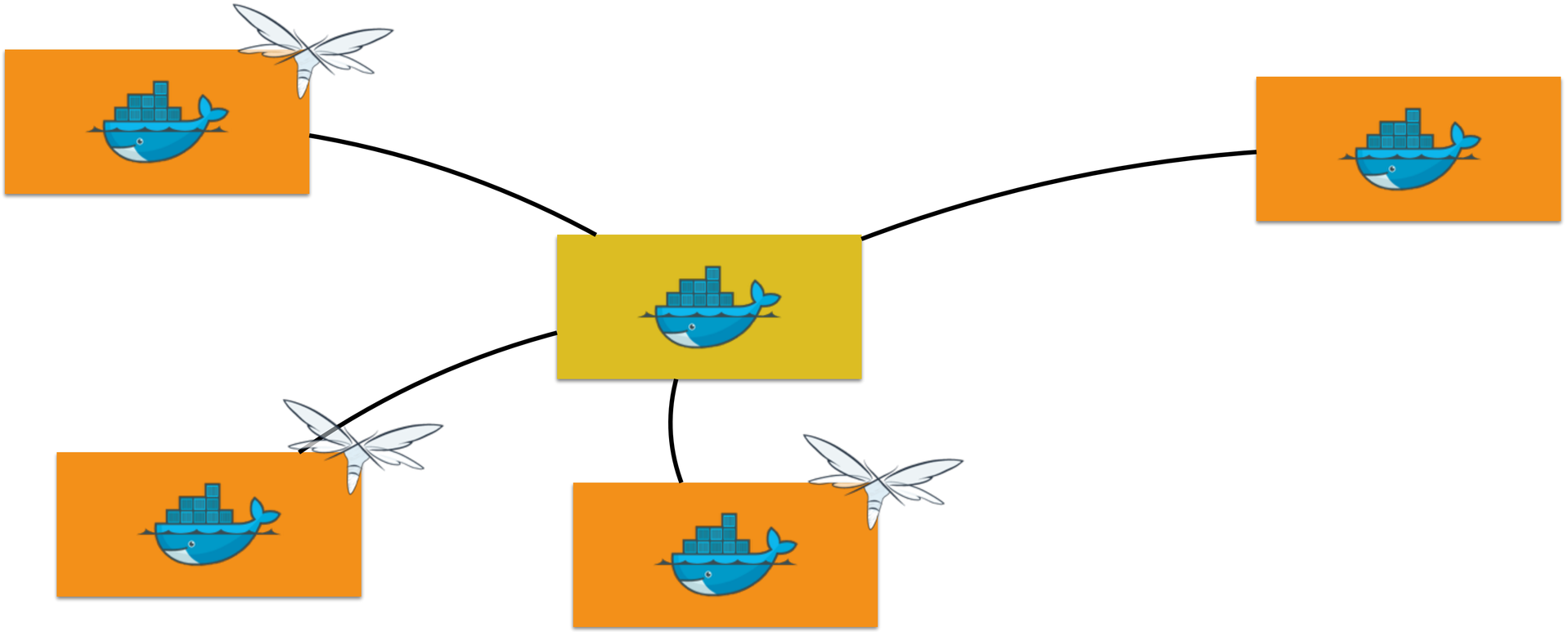
# Swarm Mode: Node Failure



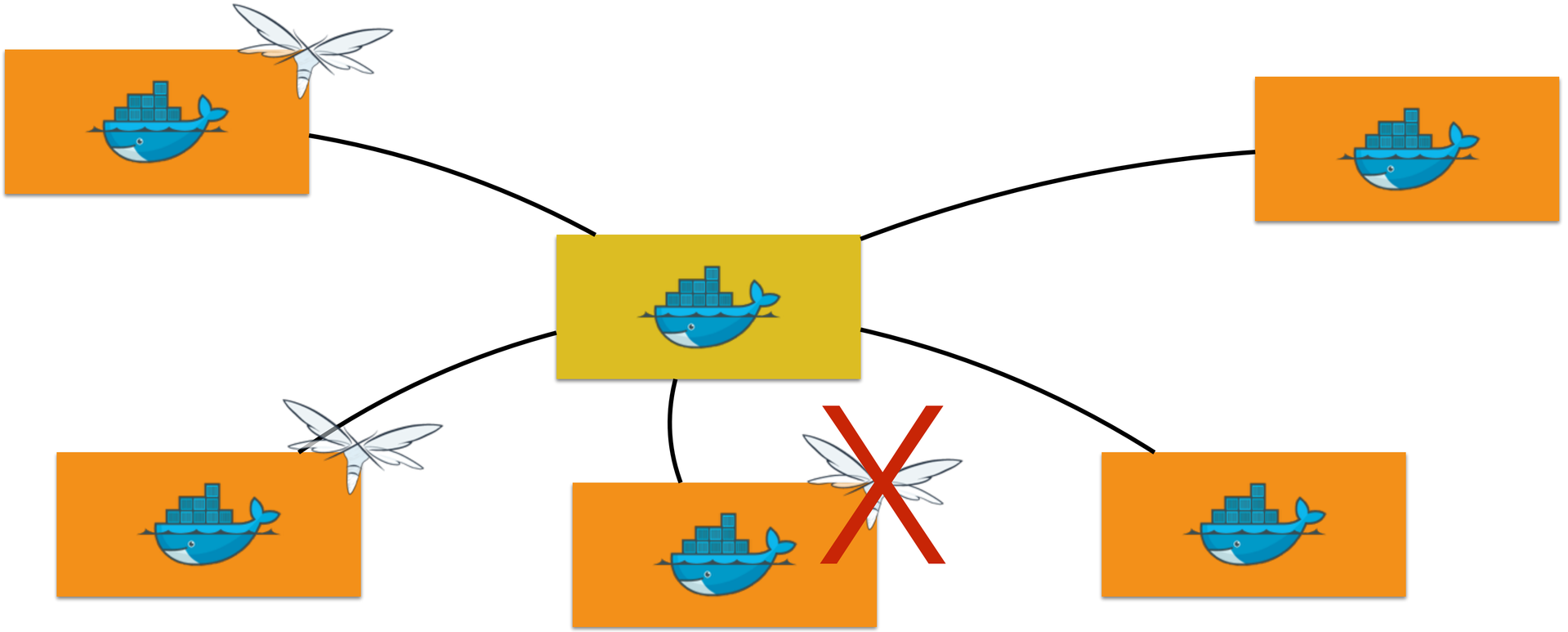
# Swarm Mode: Desired != Actual



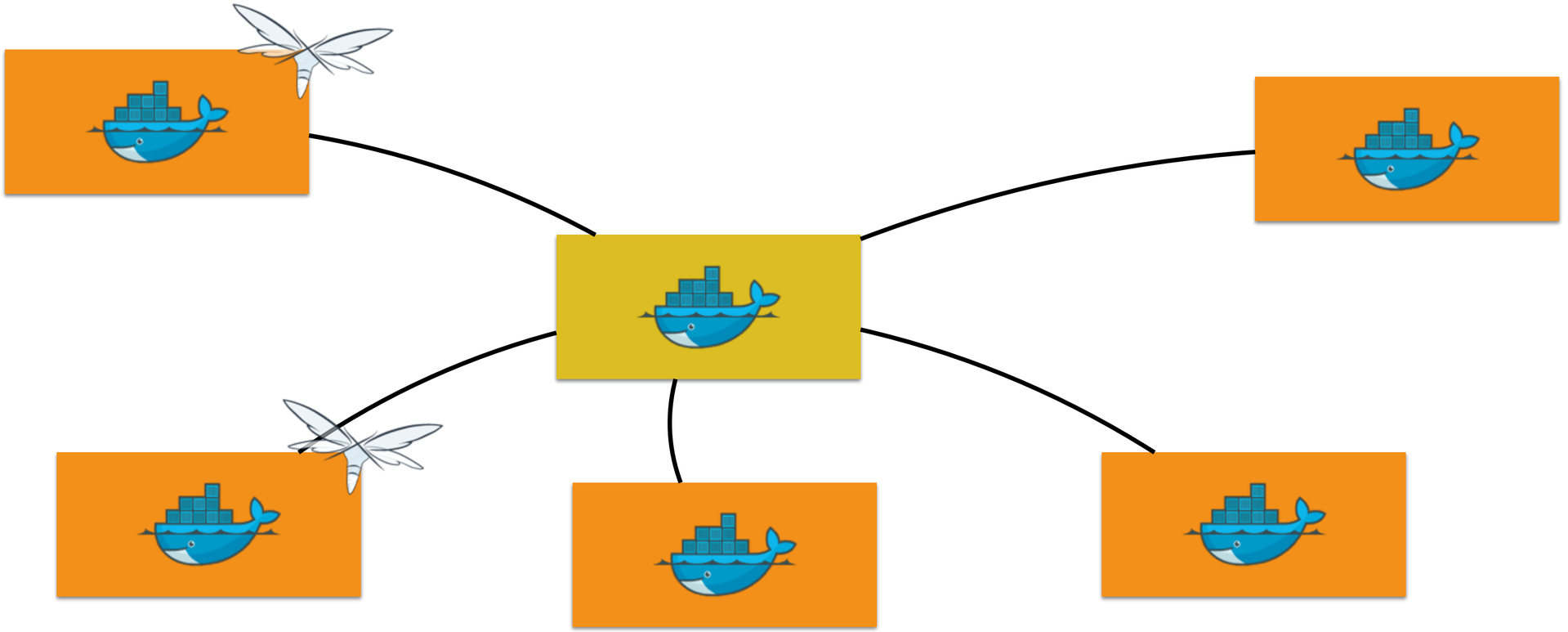
# Swarm Mode: Reconcile



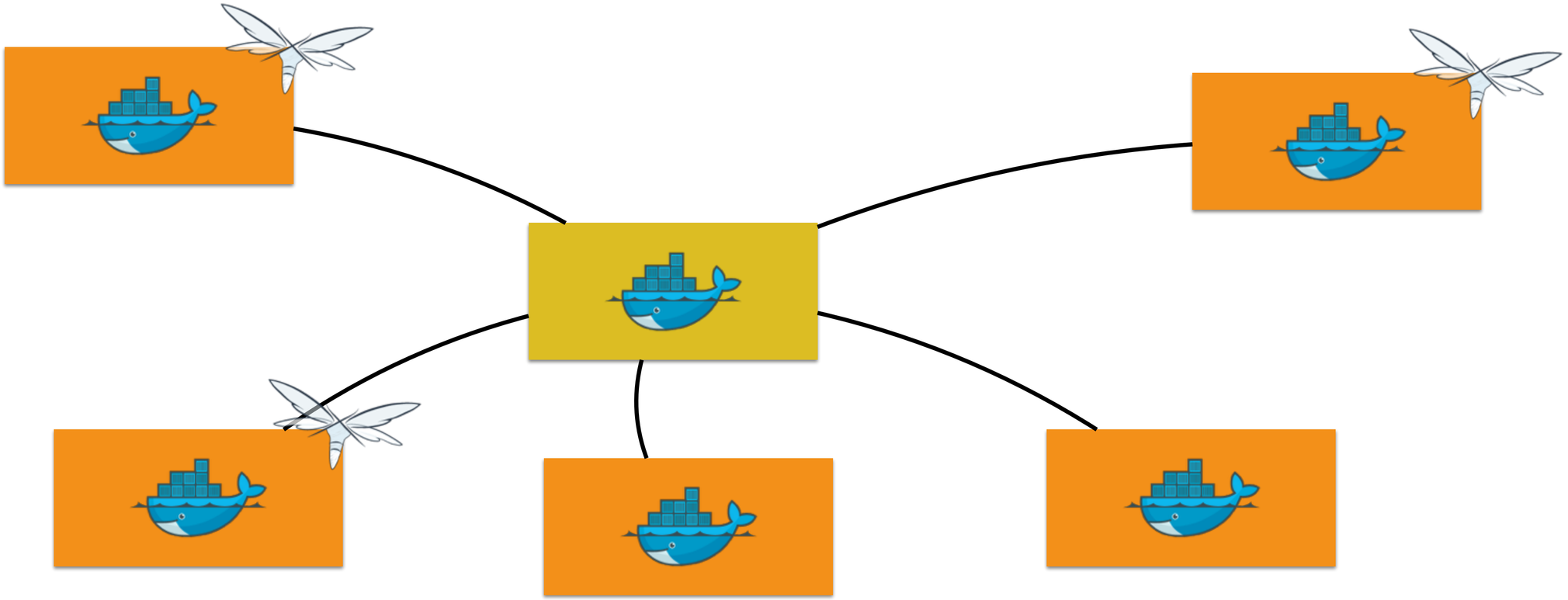
# Swarm Mode: Container Failure



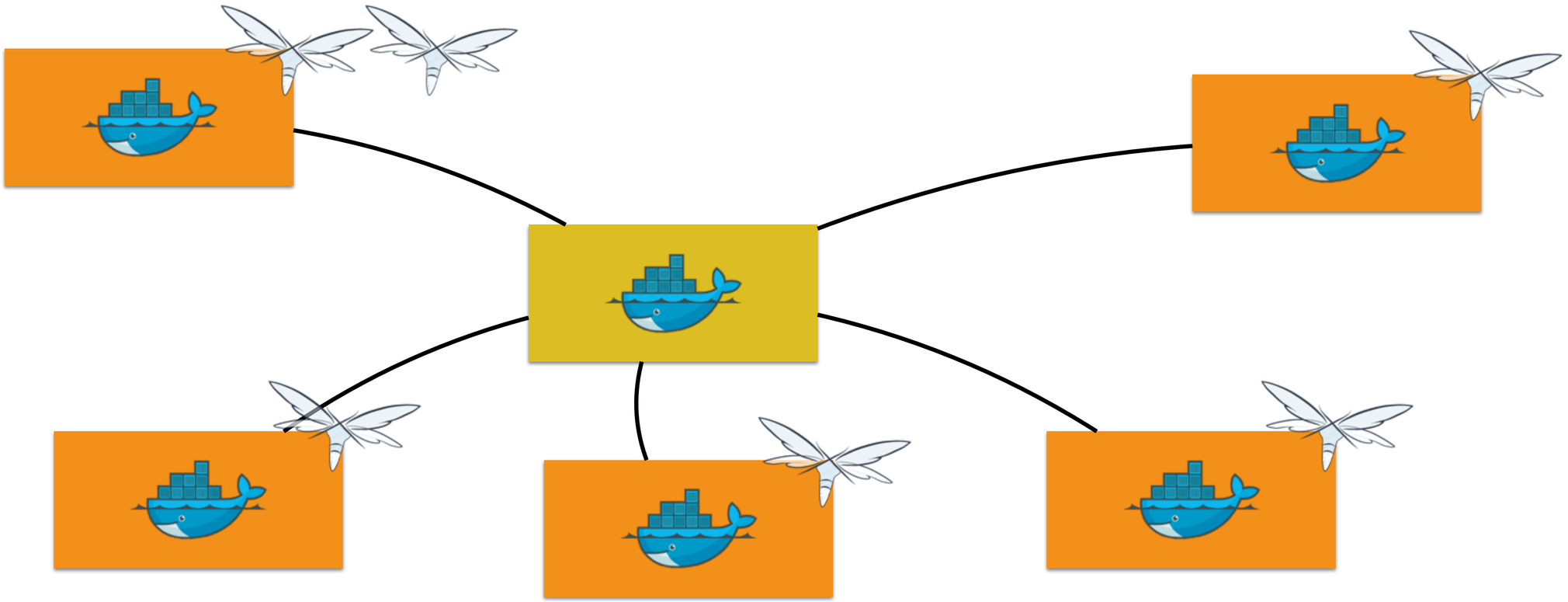
# Swarm Mode: Desired != Actual



# Swarm Mode: Reconcile



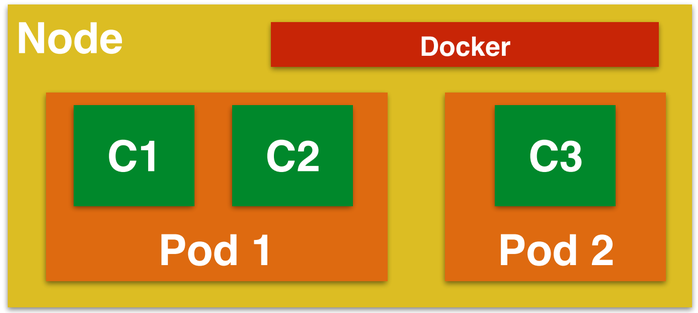
# Swarm Mode: Scale

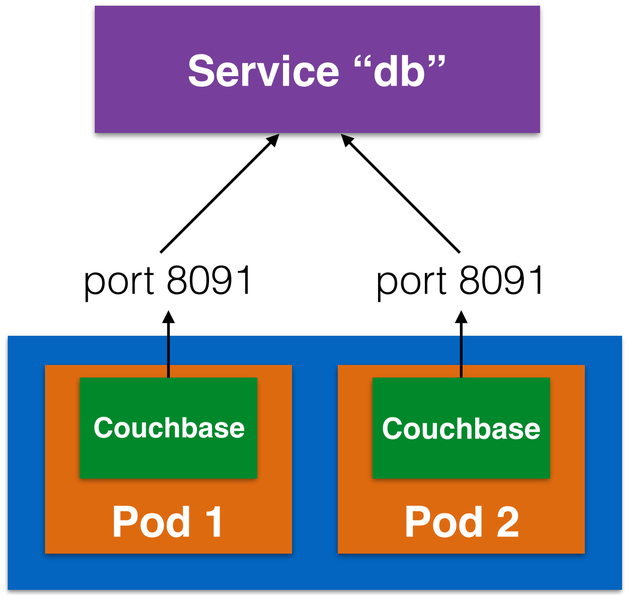


```
docker service scale web=6
```

# Kubernetes Concepts

- **Pods**: collocated group of Docker containers that share an IP and storage volume
- **Service**: Single, stable name for a set of pods, also acts as LB
- **Label**: used to organize and select group of objects
- **Replication Controller**: manages the lifecycle of pods and ensures specified number are running



# kubectl

- Controls the Kubernetes cluster manager
- `kubectl get pods` or `minions`
- `kubectl create -f <filename>`
- `kubectl update` or `delete`
- `kubectl resize –replicas=3 replicationcontrollers <name>`

# Kubernetes Config

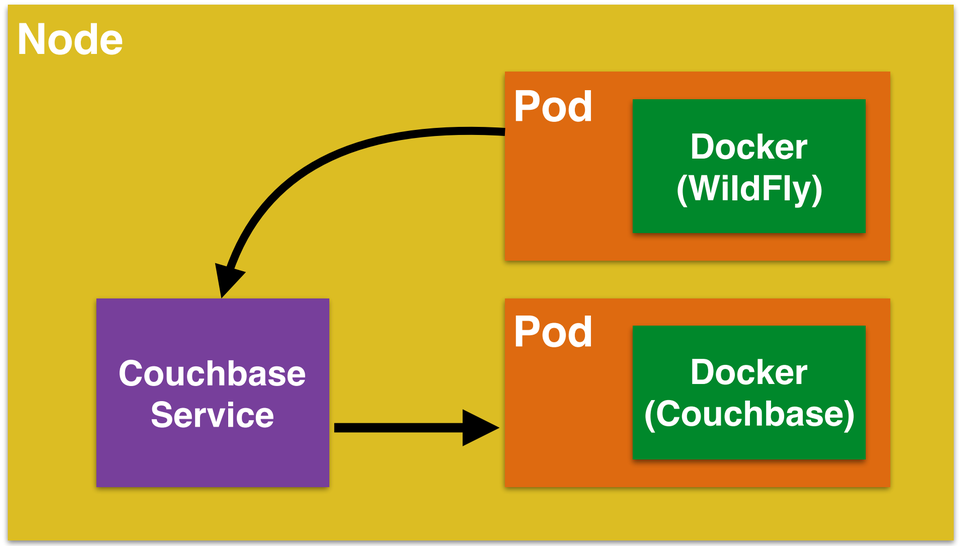
```yaml
apiVersion: v1
kind: Pod
metadata:
  name: wildfly-pod
  labels:
    name: wildfly
spec:
  containers:
    - image: jboss/wildfly
      name: wildfly-pod
      ports:
        - containerPort: 8080
```

```yaml
apiVersion: v1
kind: ReplicationController
metadata:
  name: wildfly-rc
  labels:
    name: wildfly
spec:
  replicas: 2
  template:
    metadata:
      labels:
        name: wildfly
    spec:
      containers:
      - name: wildfly-rc-pod
        image: jboss/wildfly
        ports:
        - containerPort: 8080
```

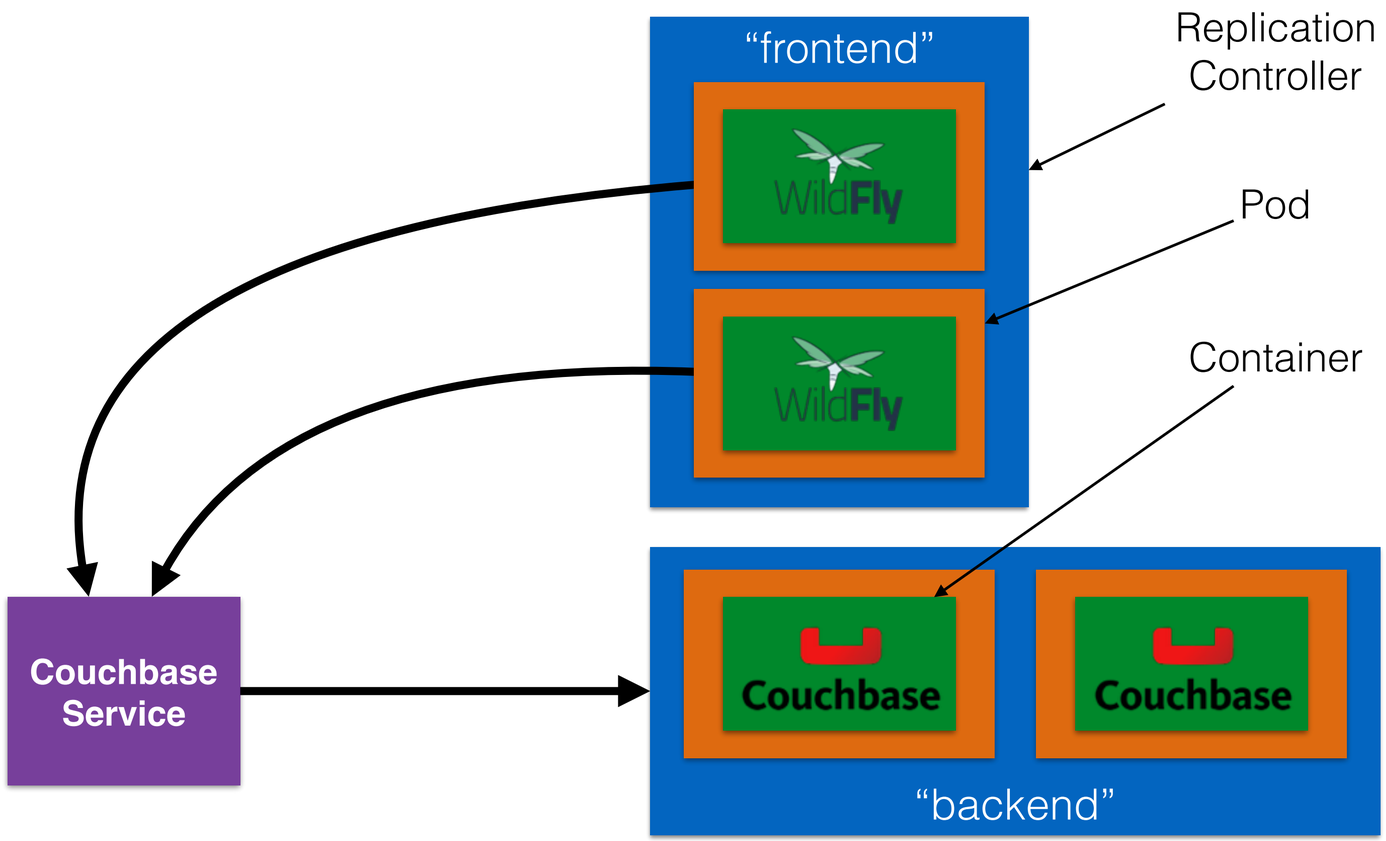
# Services

- Abstract a set of pods as a single IP and port
  - Simple TCP/UDP load balancing
- Creates environment variables in other pods
- Stable endpoint for pods to reference
  - Allows list of pods to change dynamically

# Services



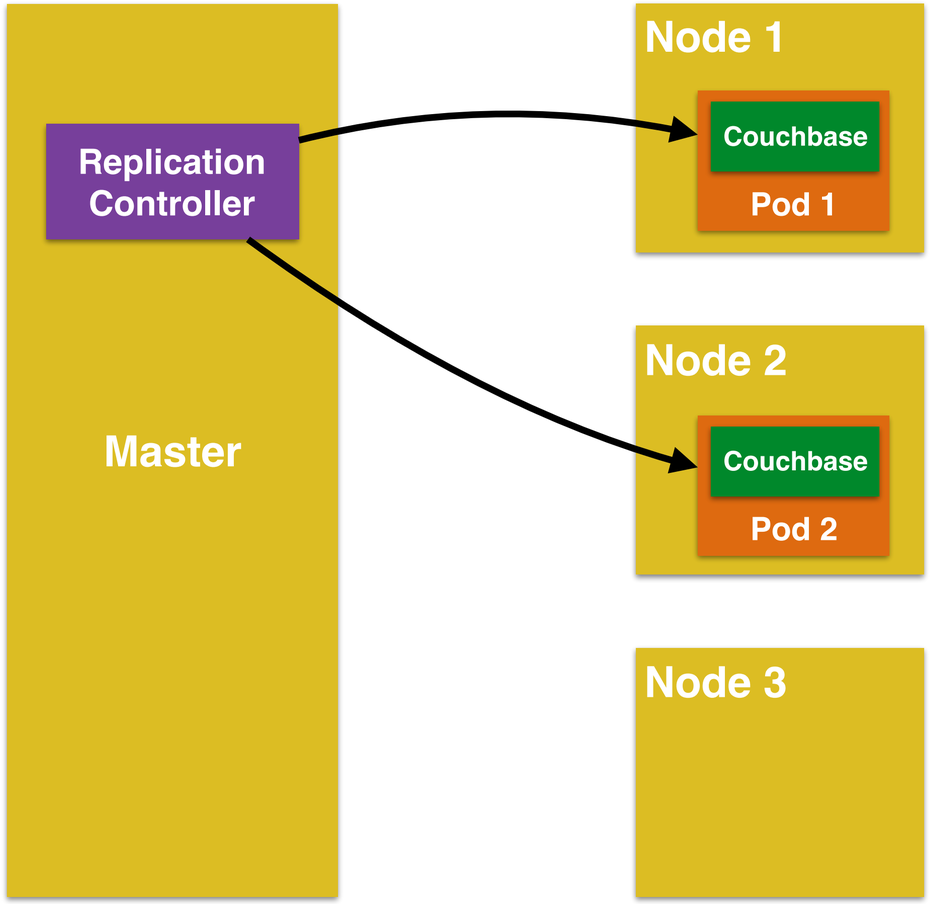
# Services



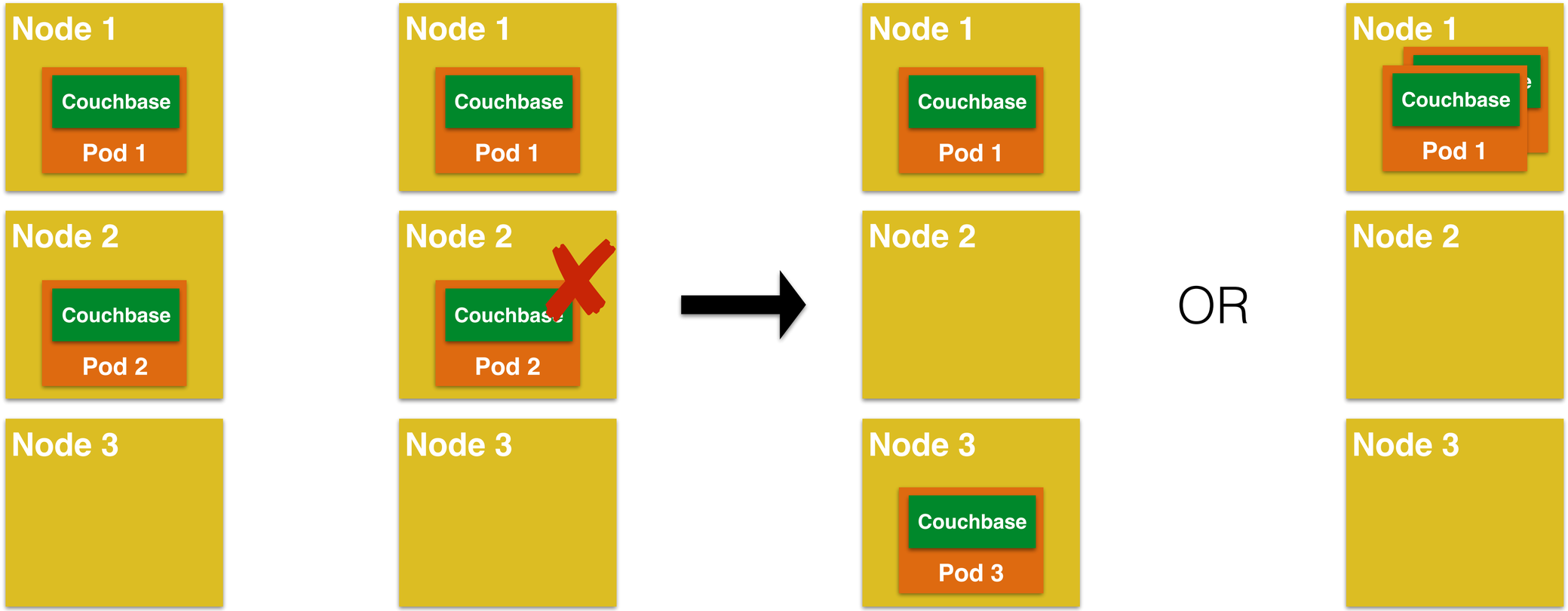
# Replication Controller Configuration

```yaml
apiVersion: v1
kind: ReplicationController
metadata:
  name: wildfly-rc
  labels:
    name: wildfly
    context: docker-k8s-lab
spec:
  replicas: 1
  template:
    metadata:
      labels:
        name: wildfly
    spec:
      containers:
      - name: wildfly-rc-pod
        image: arungupta/wildfly-mysql-javaee7:k8s
        ports:
        - containerPort: 8080
```

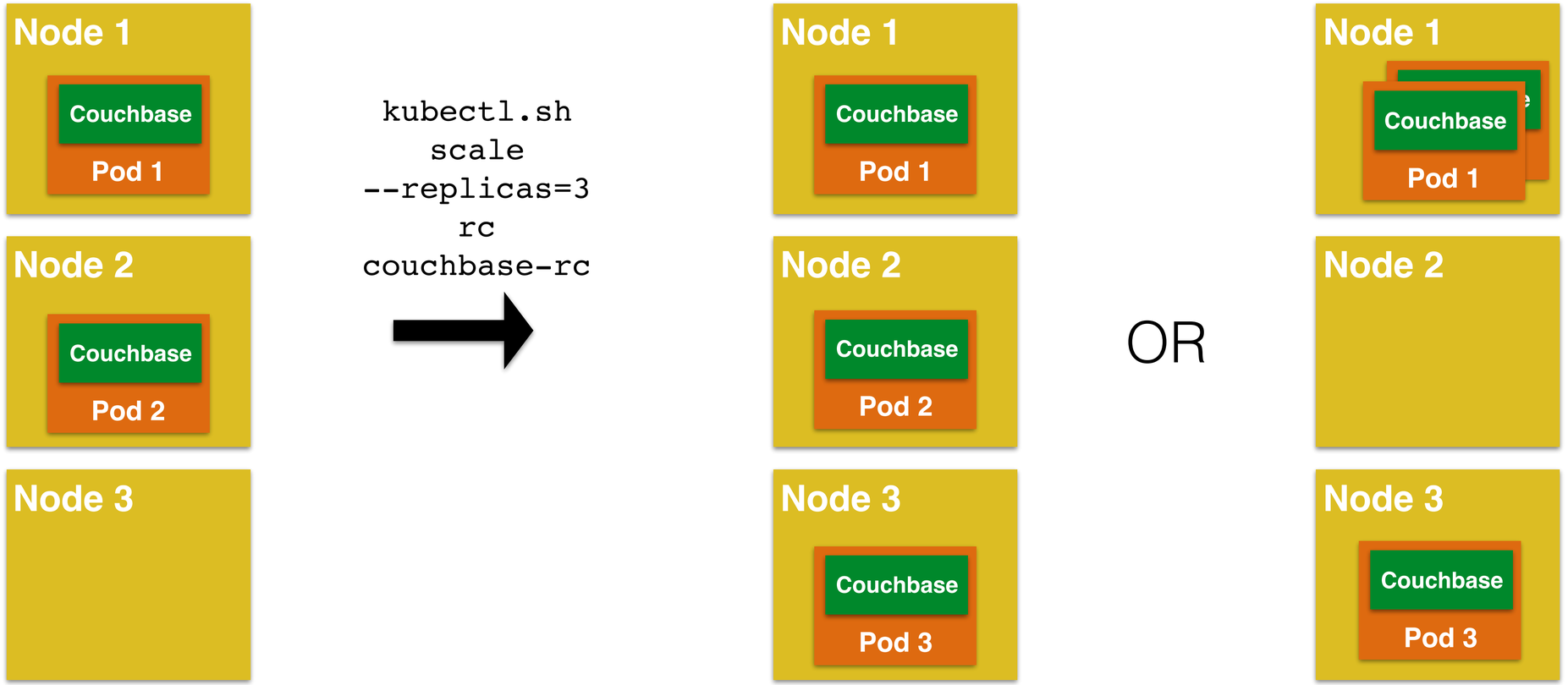
# Replication Controller



# Replication Controller: Automatic Rescheduling



# Replication Controller: Scaling



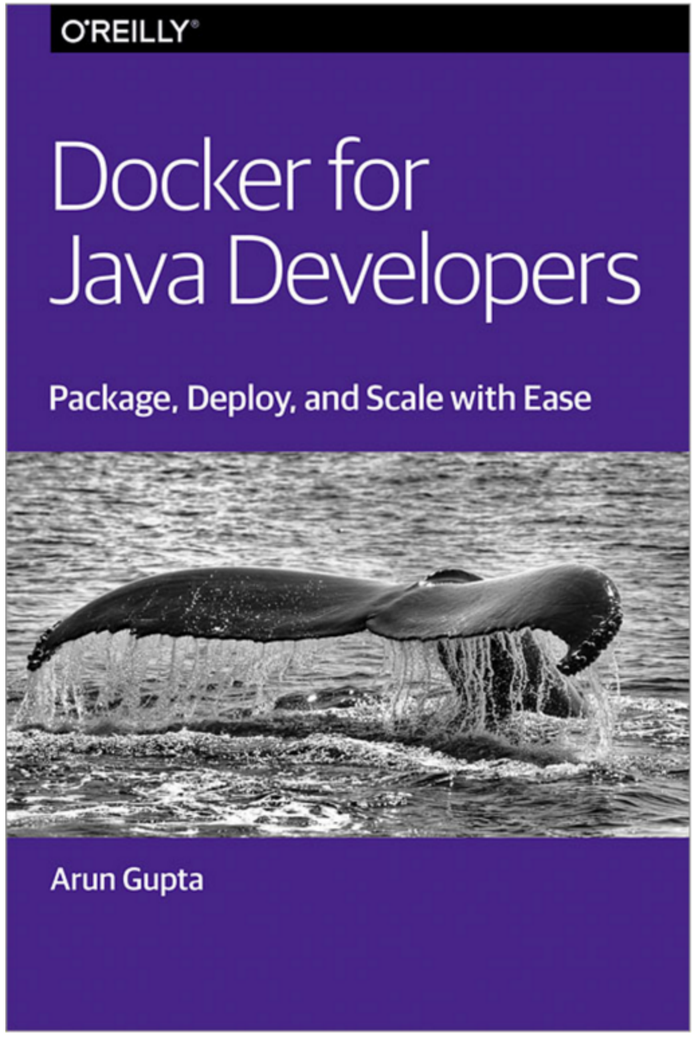
ONLINE COURSE

Arun Gupta

# KUBERNETES FOR JAVA DEVELOPERS

September 23 | 10AM-12PM

bit.ly/kubejava




bit.ly/dockerjava

# Thanks!

Arun Gupta, @arungupta

github.com/javaee-samples/docker-java/tree/master/slides