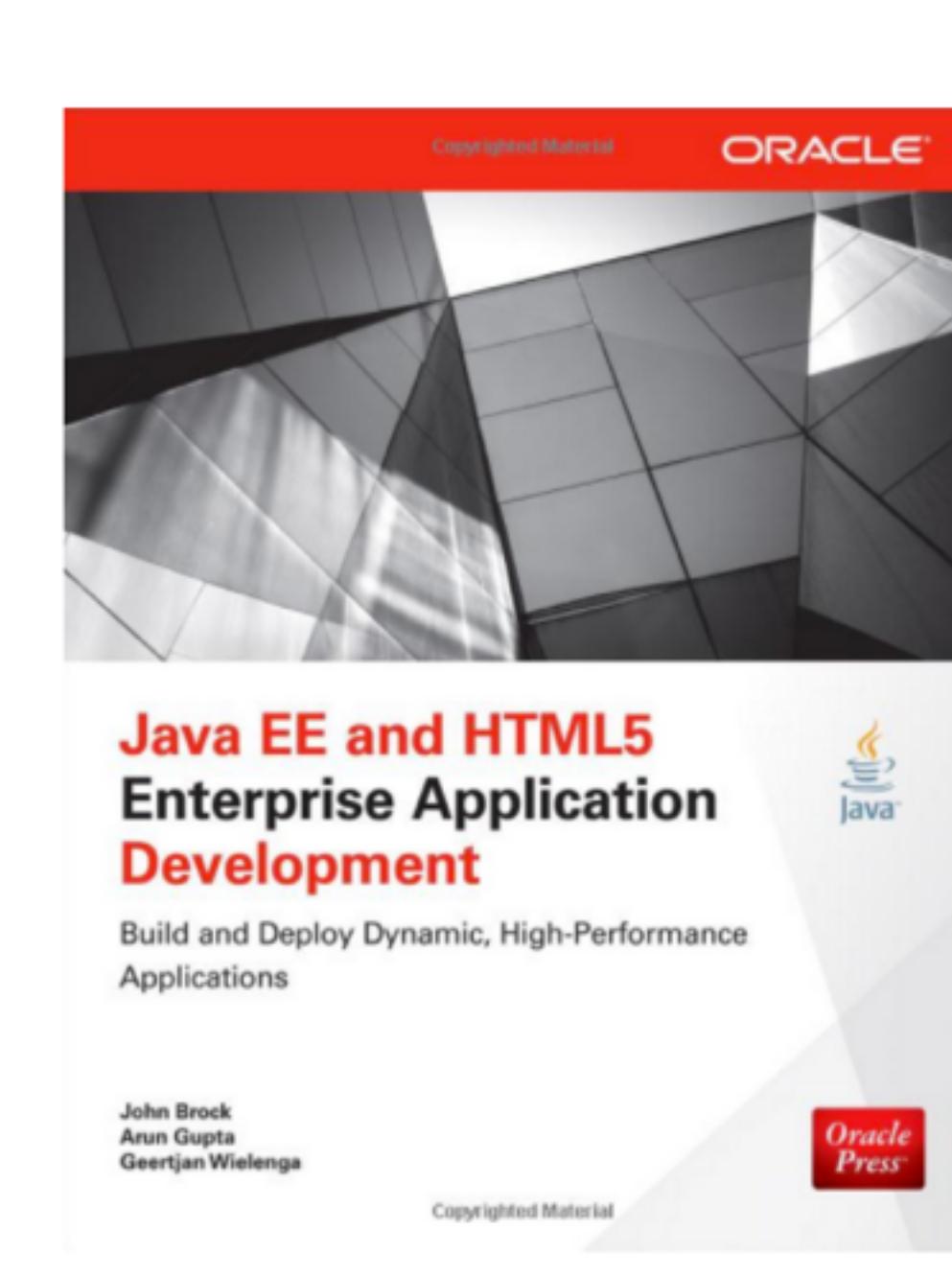
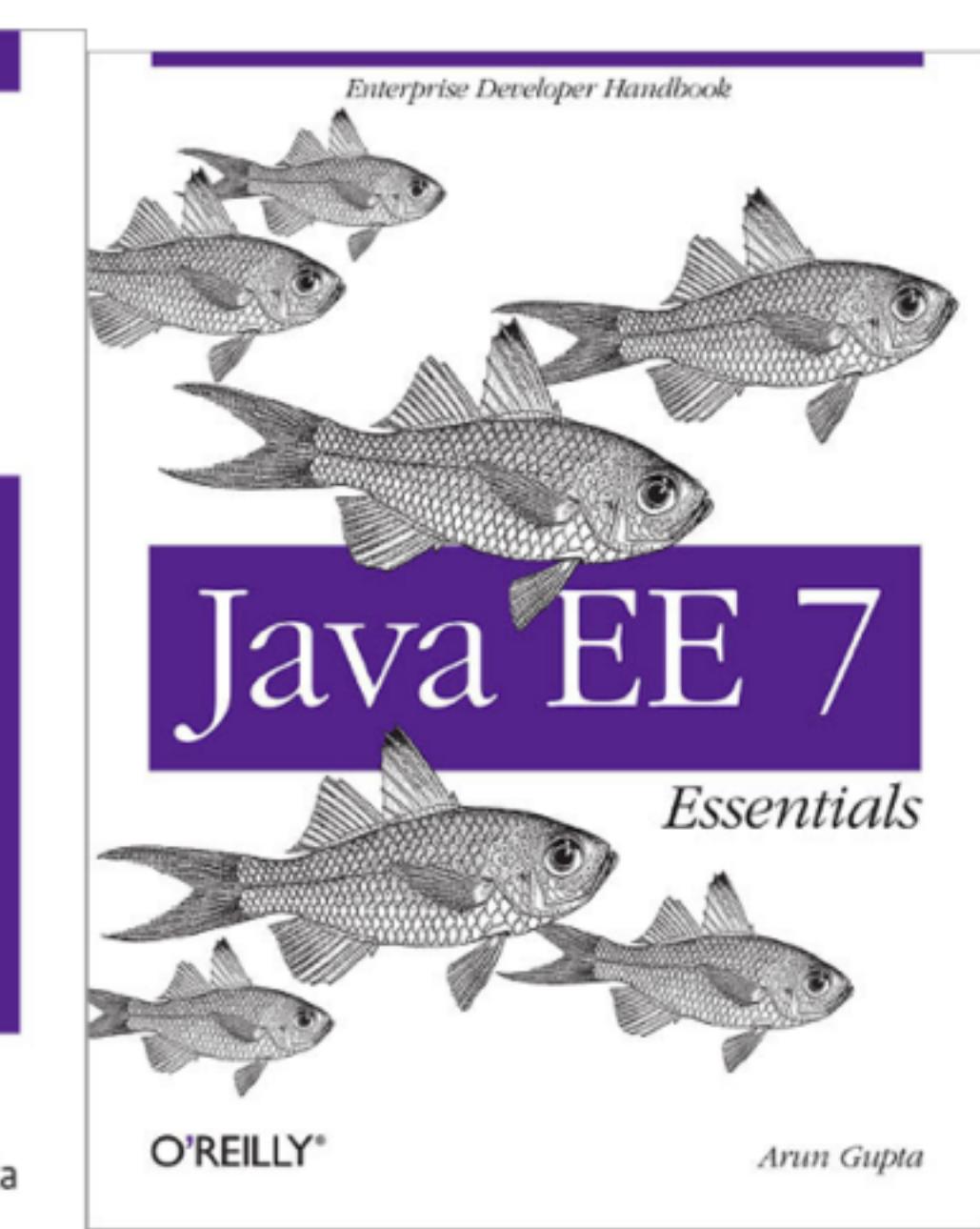
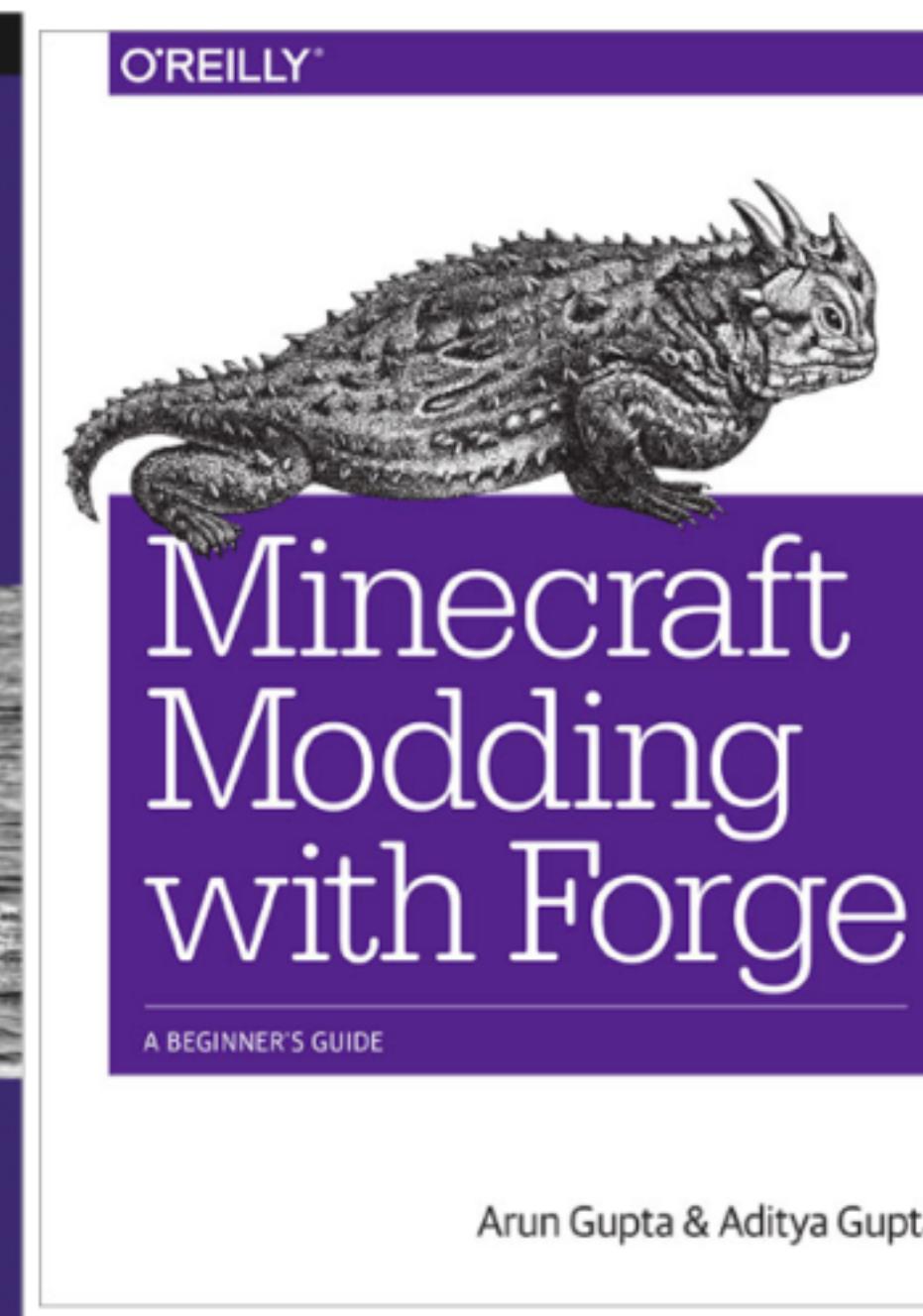
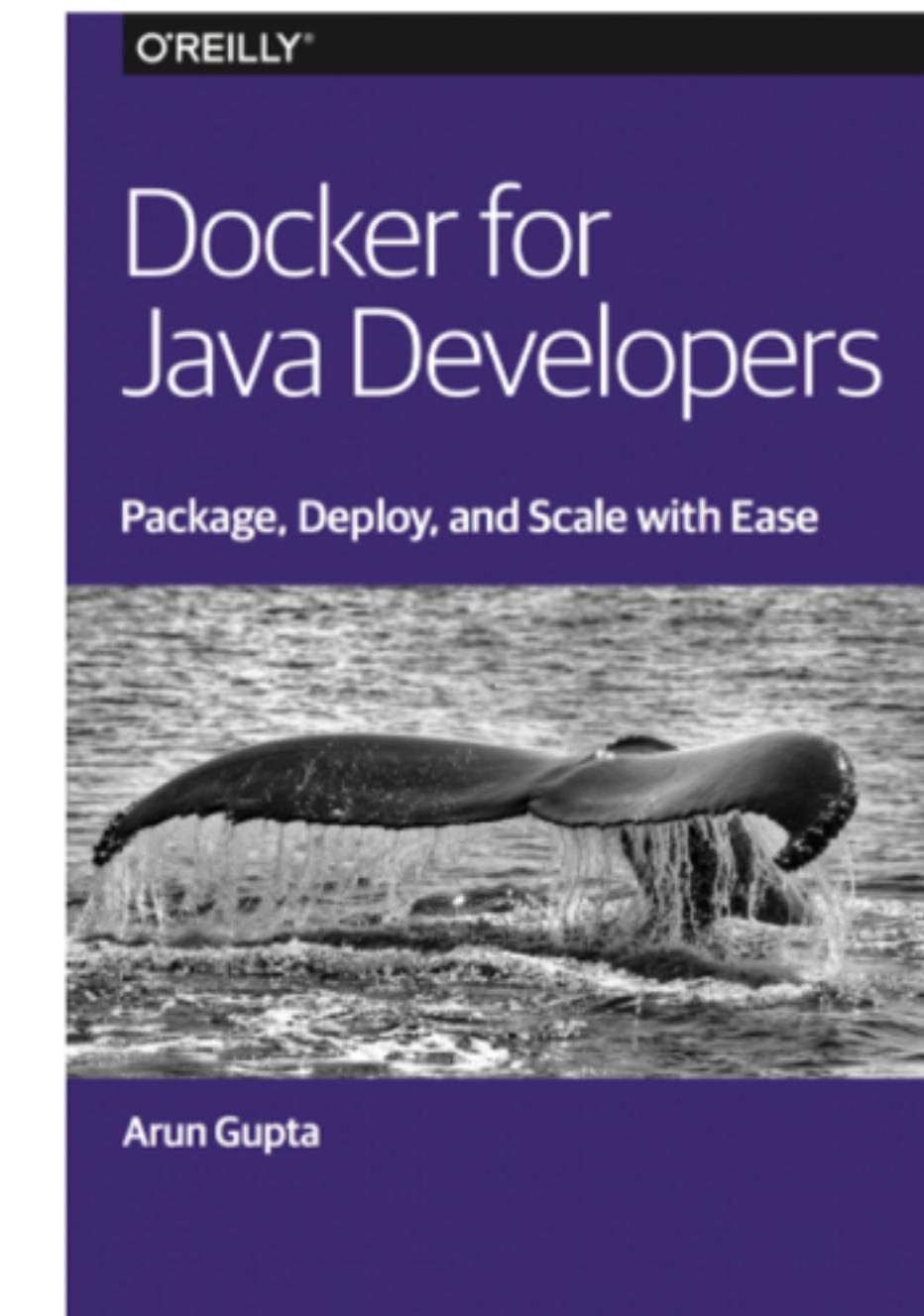




Kubernetes for Java Developers

Arun Gupta, @arungupta
Docker Captain, Java Champion

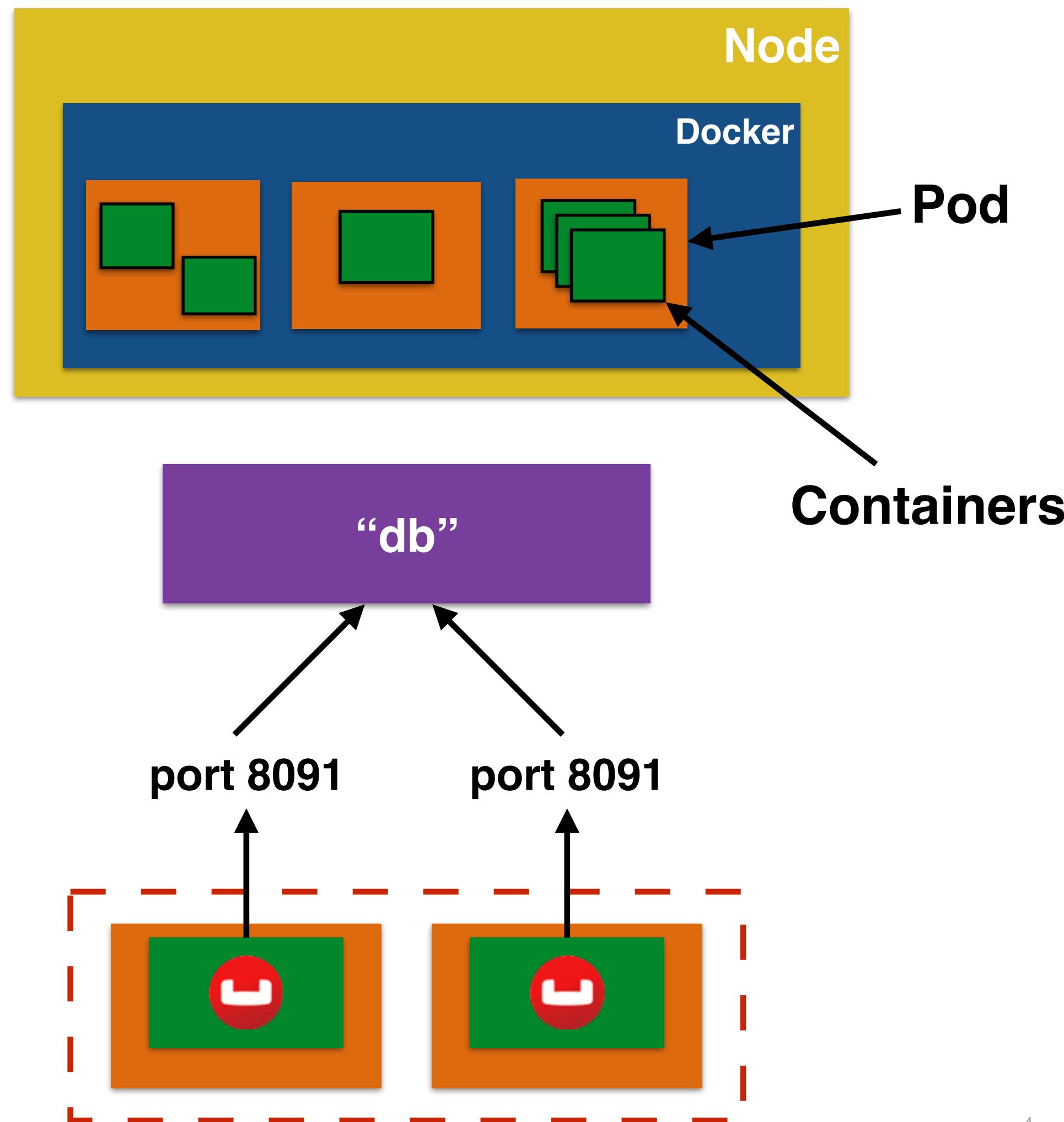


Kubernetes

- Open source orchestration system for containers
 - Docker, rkt, OCI, ...
- Provide declarative primitives for the “desired state”
 - Self-healing
 - Auto-restarting
 - Schedule across hosts
 - Replicating

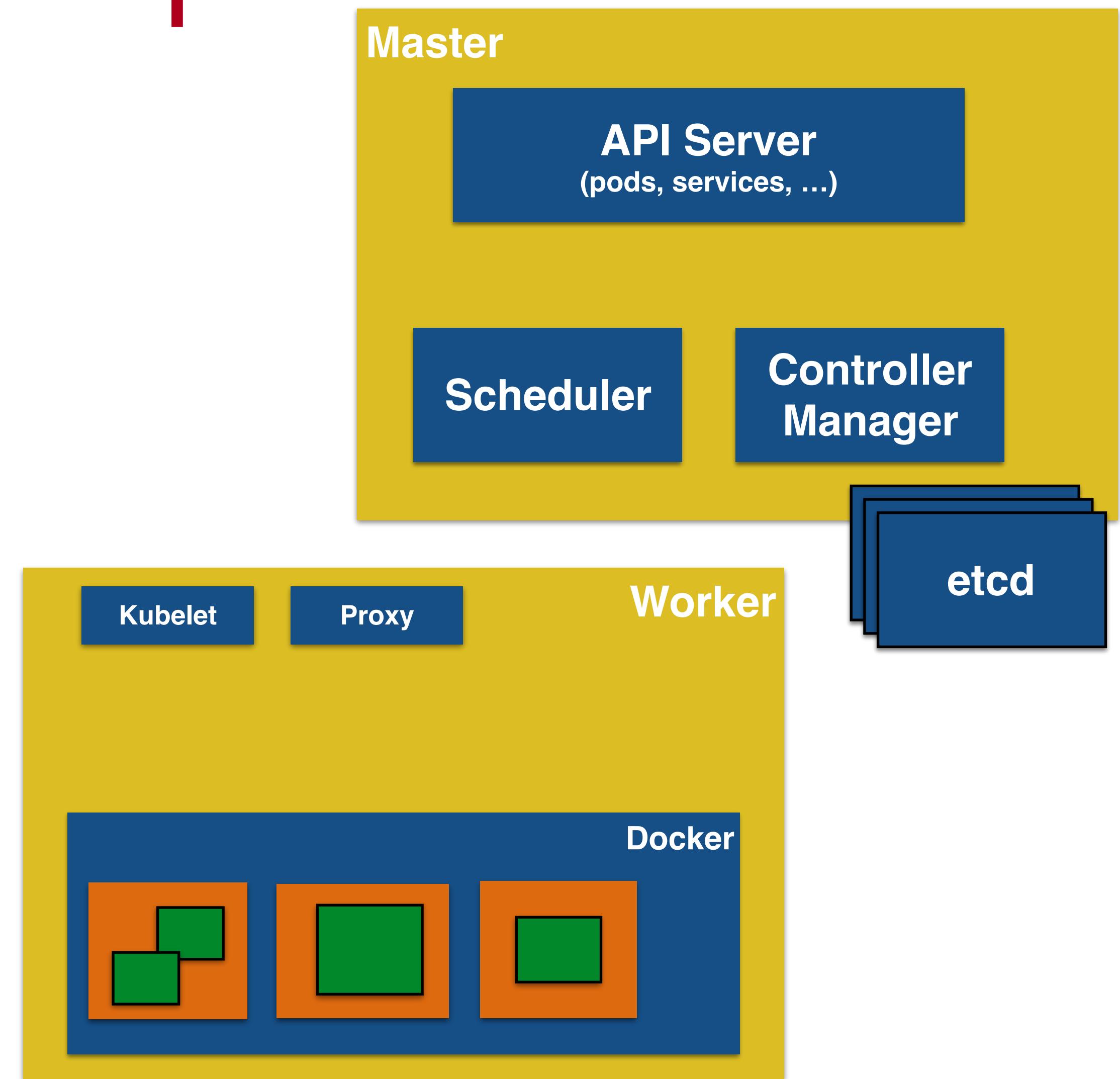
Kubernetes Concepts

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

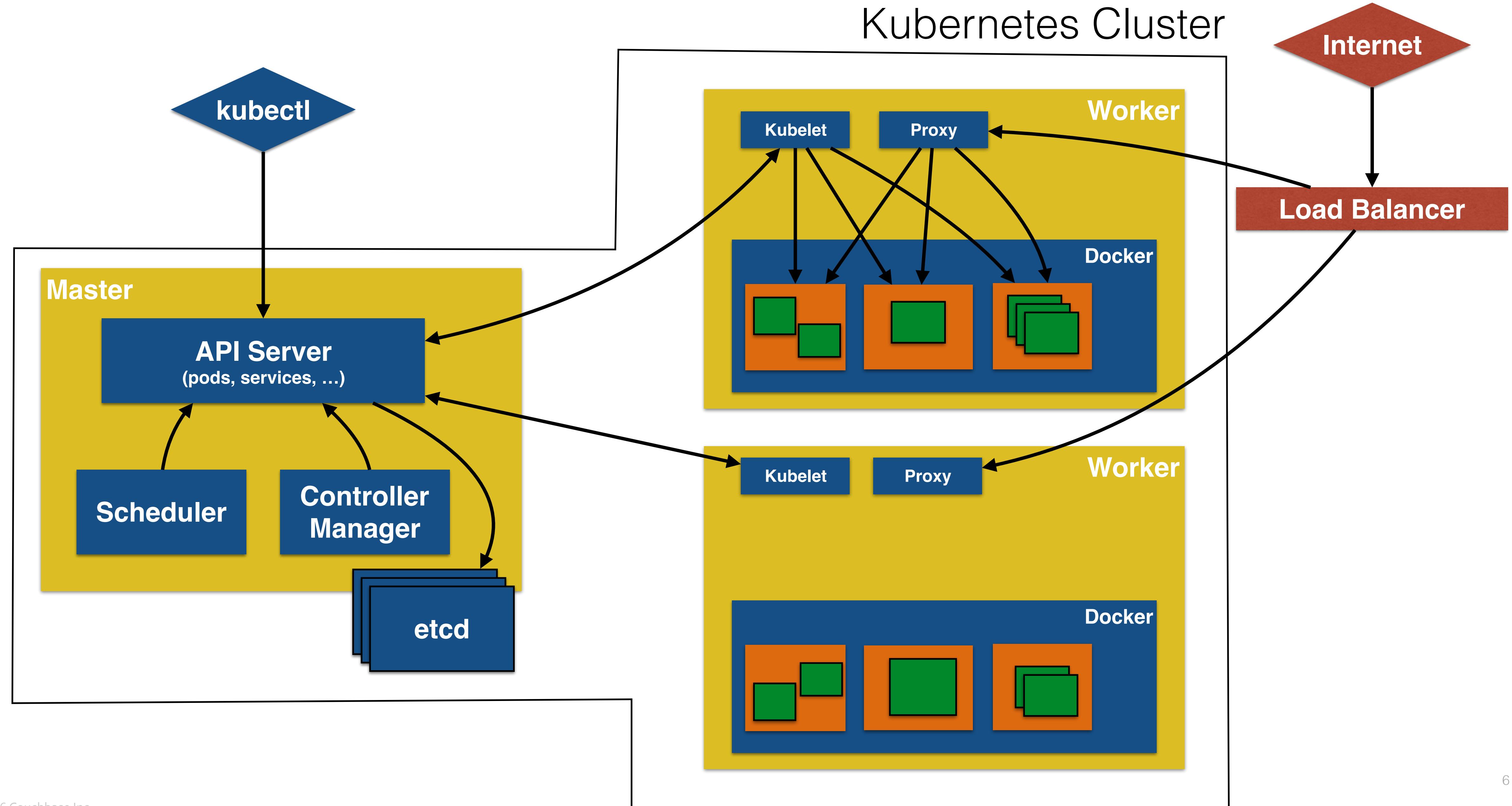


Kubernetes Components

- **Node**: Machine or VM in the cluster
- **Master**: Central control plane, provides unified view of the cluster
 - **etcd**: distributed key-value store used to persist Kubernetes system state
- **Worker**: Docker host running *kubelet* (node agent) and *proxy* services
 - Runs pods and containers
 - Monitored by *systemd* (CentOS) or *monit* (Debian)



Kubernetes Cluster





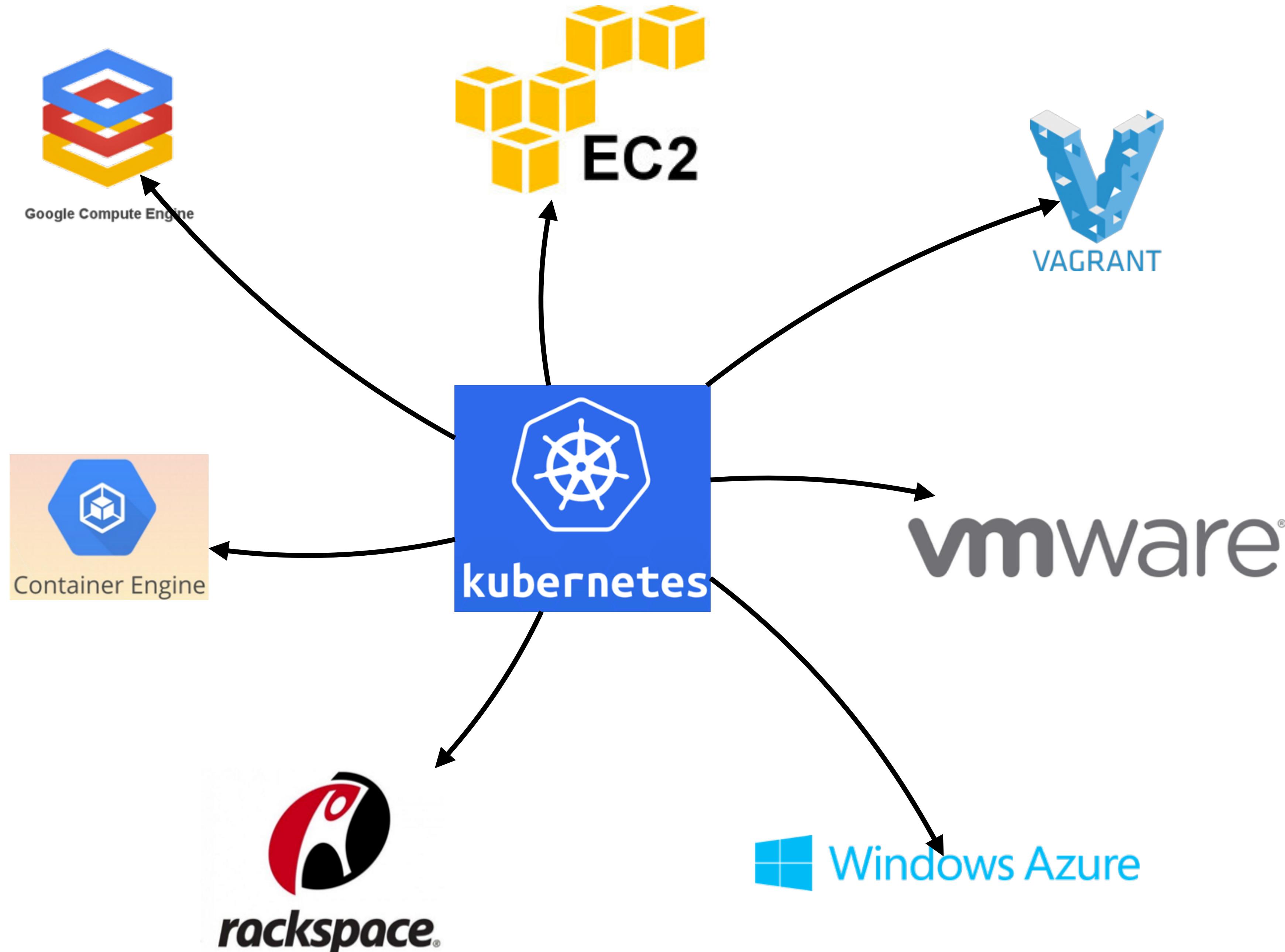
kubectl

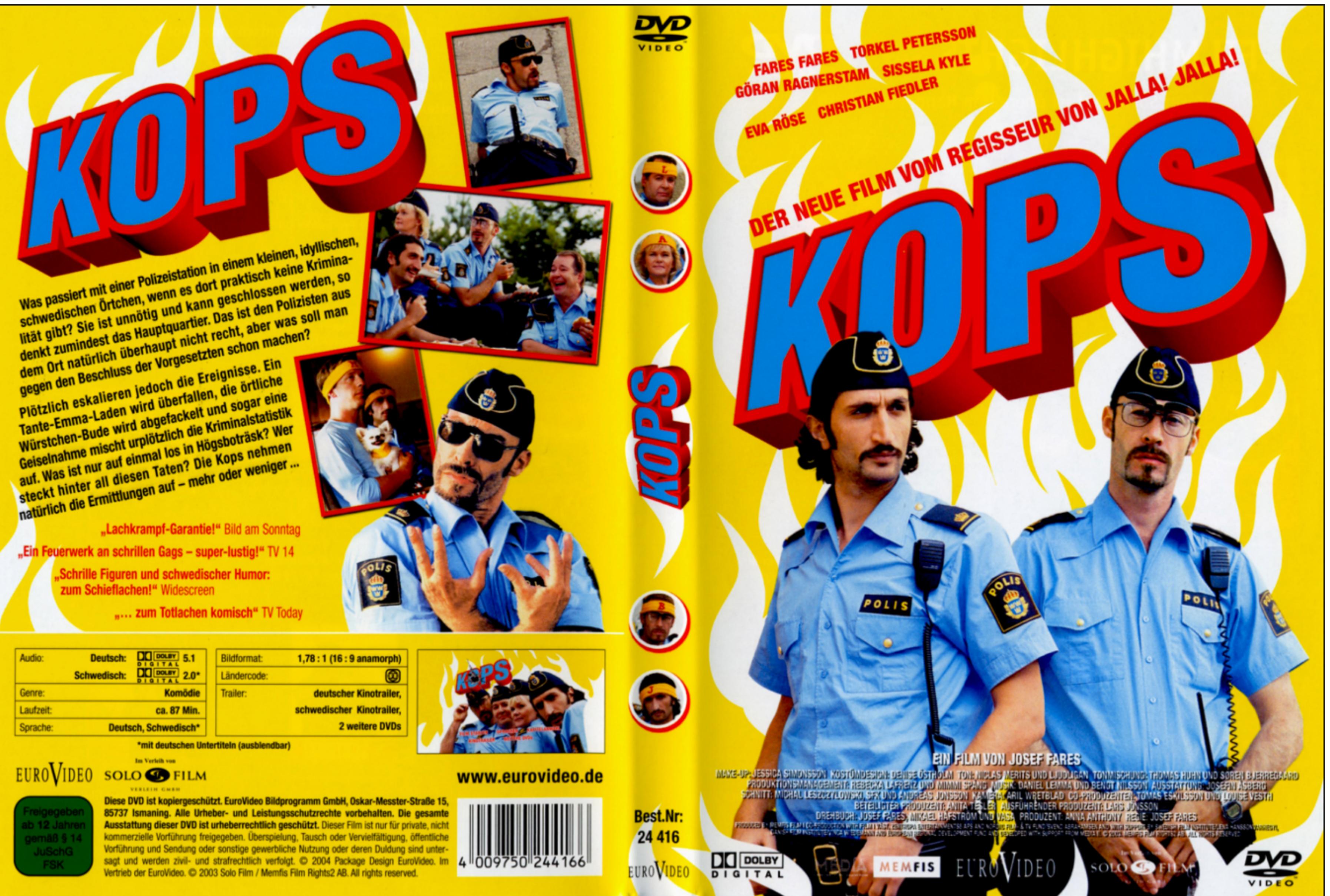
- Controls the Kubernetes cluster manager
- CRUD Kubernetes resources
 - `create, get, describe, delete, ...`
 - `kubectl create -f <filename>`
- `kubectl get nodes or pods`
- `kubectl scale --replicas=3 rc/<name>`



Minikube

- Runs a single node cluster in a VM
- Targeted for local development
- `minikube start, stop, docker-env, ...`
- Requires `kubectl` CLI
- github.com/kubernetes/minikube/releases





Start Kubernetes Cluster

Kubernetes Pod Configuration

```
1  apiVersion: v1
2  kind: Pod
3  metadata:
4    name: wildfly-pod
5    labels:
6      name: wildfly-pod
7  spec:
8    containers:
9      - name: wildfly
10     image: jboss/wildfly
11     ports:
12       - containerPort: 8080
```

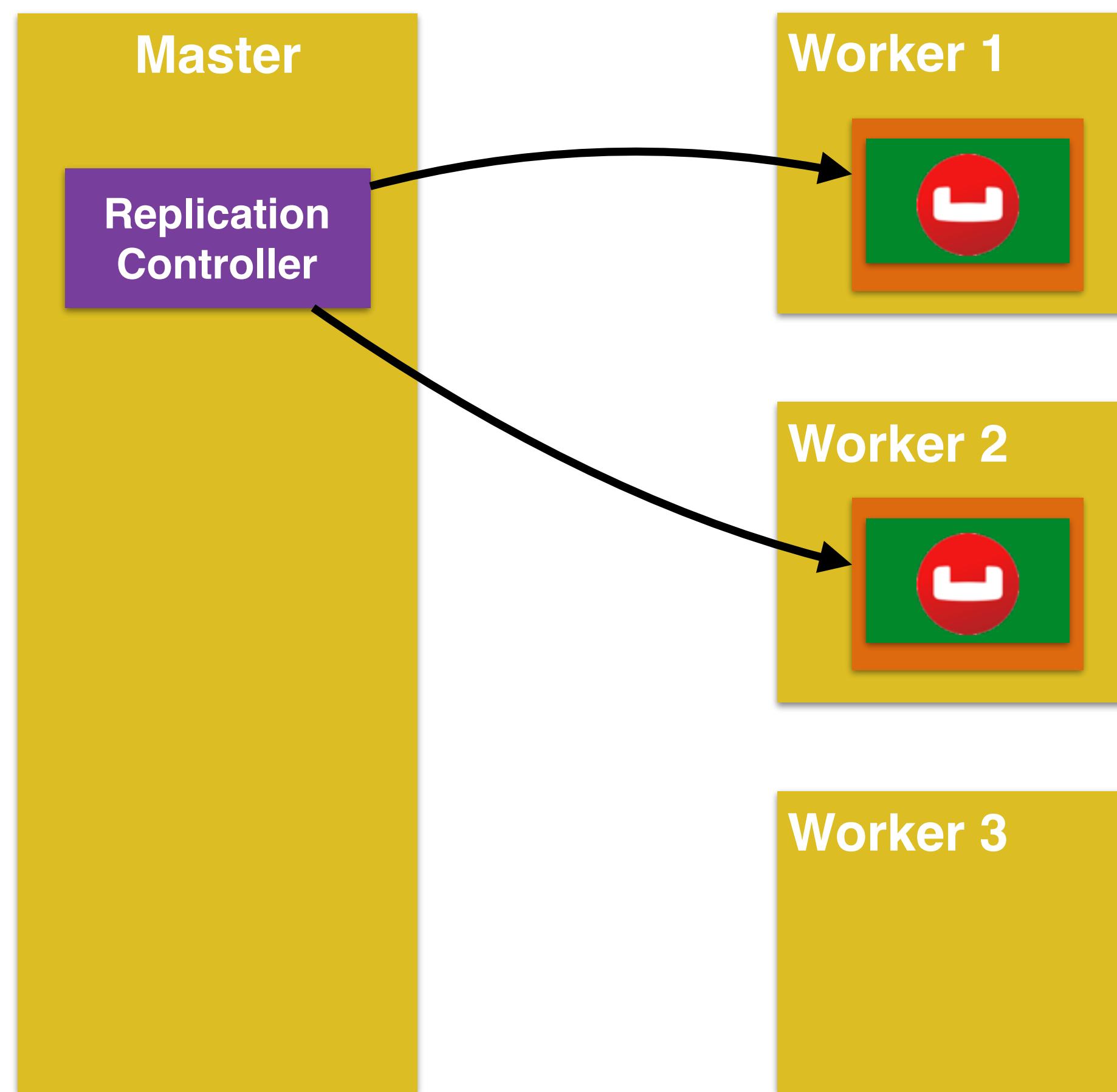
Replication Controller

- Ensures that a specified number of pod "replicas" are running
 - Pod templates are cookie cutters
 - Rescheduling
 - Manual or auto-scale replicas
 - Rolling updates
- Generally wrap a pod in a RC
- Only appropriate for pods with `Restart=Always` policy (default)

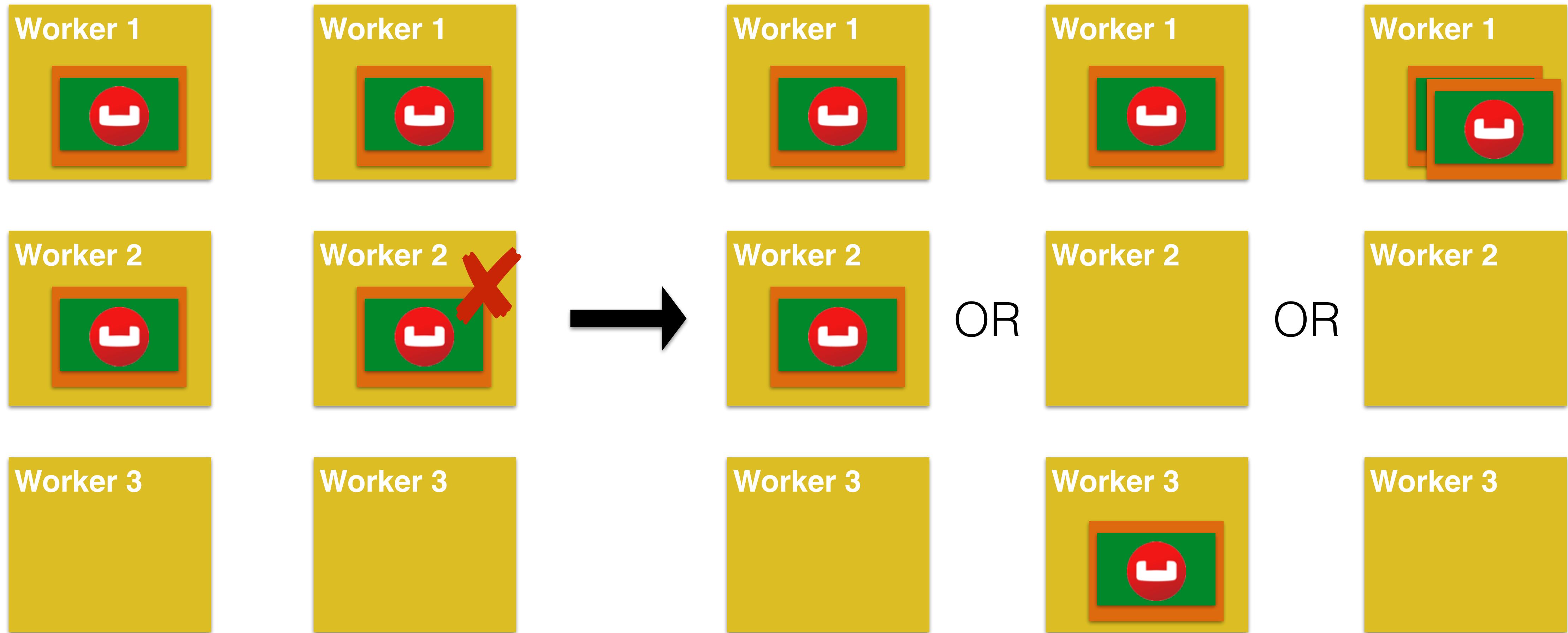
Kubernetes Replication Controller Configuration

```
1  apiVersion: v1
2  kind: ReplicationController
3  metadata:
4    name: wildfly-rc
5  spec:
6    replicas: 2
7    selector:
8      app: wildfly-rc-pod
9    template:
10      metadata:
11        labels:
12          app: wildfly-rc-pod
13      spec:
14        containers:
15          - name: wildfly
16            image: jboss/wildfly
17        ports:
18          - containerPort: 8080
```

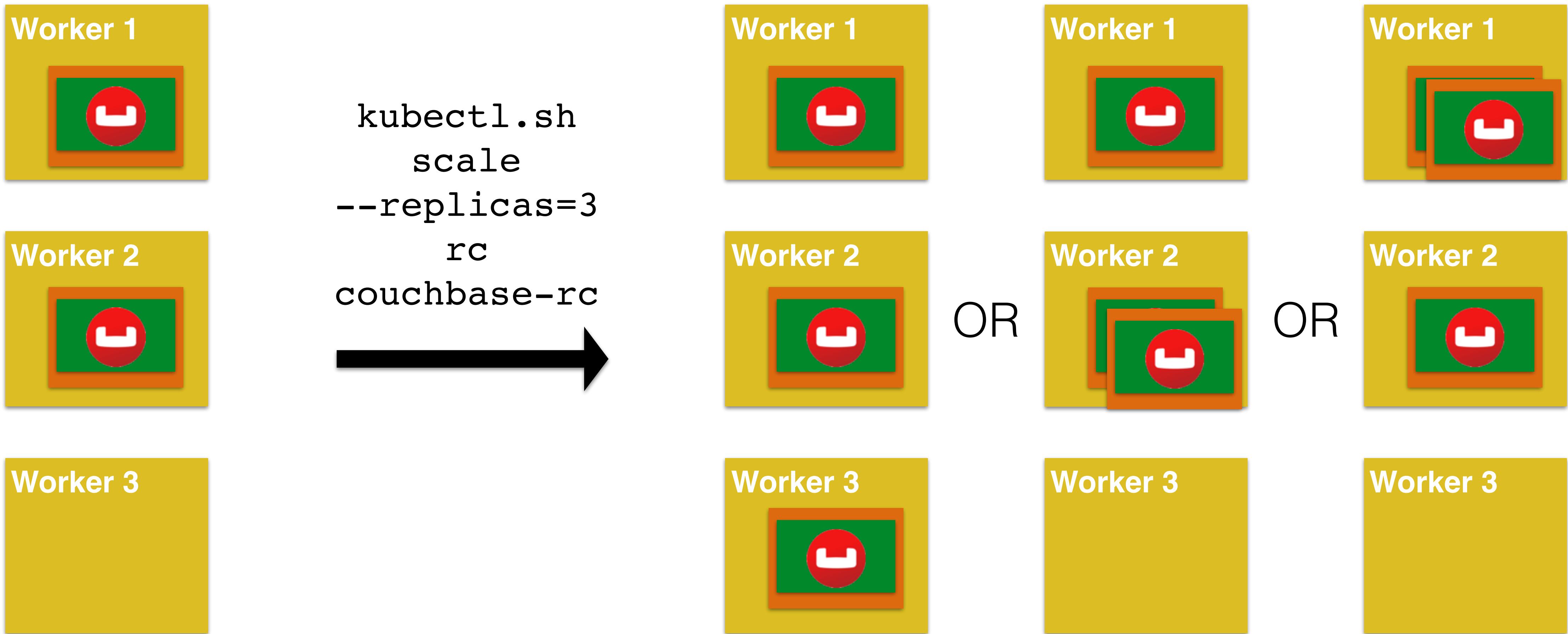
Replication Controller



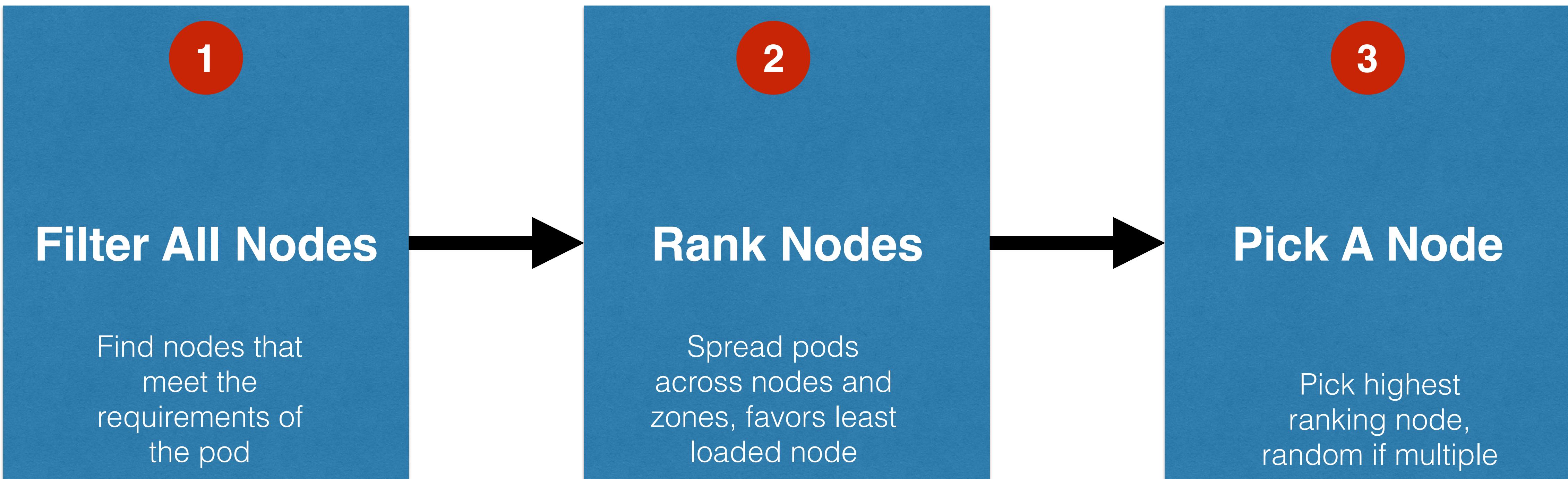
RC: “Actual” vs “Desired” State



RC: Scale Pods



Kubernetes Scheduling Algorithm



Replica Set

- Next generation Replication Controller
- Set-based selector requirement
 - Expression: key, operator, value
 - Operators: In, NotIn, Exists, DoesNotExist
- Generally created with Deployment
- Enables Horizontal Pod Autoscaling

Replica Set Configuration

```
1  apiVersion: extensions/v1beta1
2  kind: ReplicaSet
3  metadata:
4    name: wildfly-rs
5  spec:
6    replicas: 2
7    selector:
8      matchLabels:
9        app: wildfly-rs-pod
10     matchExpressions:
11       - {key: tier, operator: In, values: ["backend"]}
12       - {key: environment, operator: NotIn, values: ["dev"]}
13   template:
14     metadata:
15       labels:
16         app: wildfly-rs-pod
17         tier: backend
18         environment: dev
19     spec:
20       containers:
21         - name: wildfly
22           image: jboss/wildfly
23       ports:
24         - containerPort: 8080
```

Services

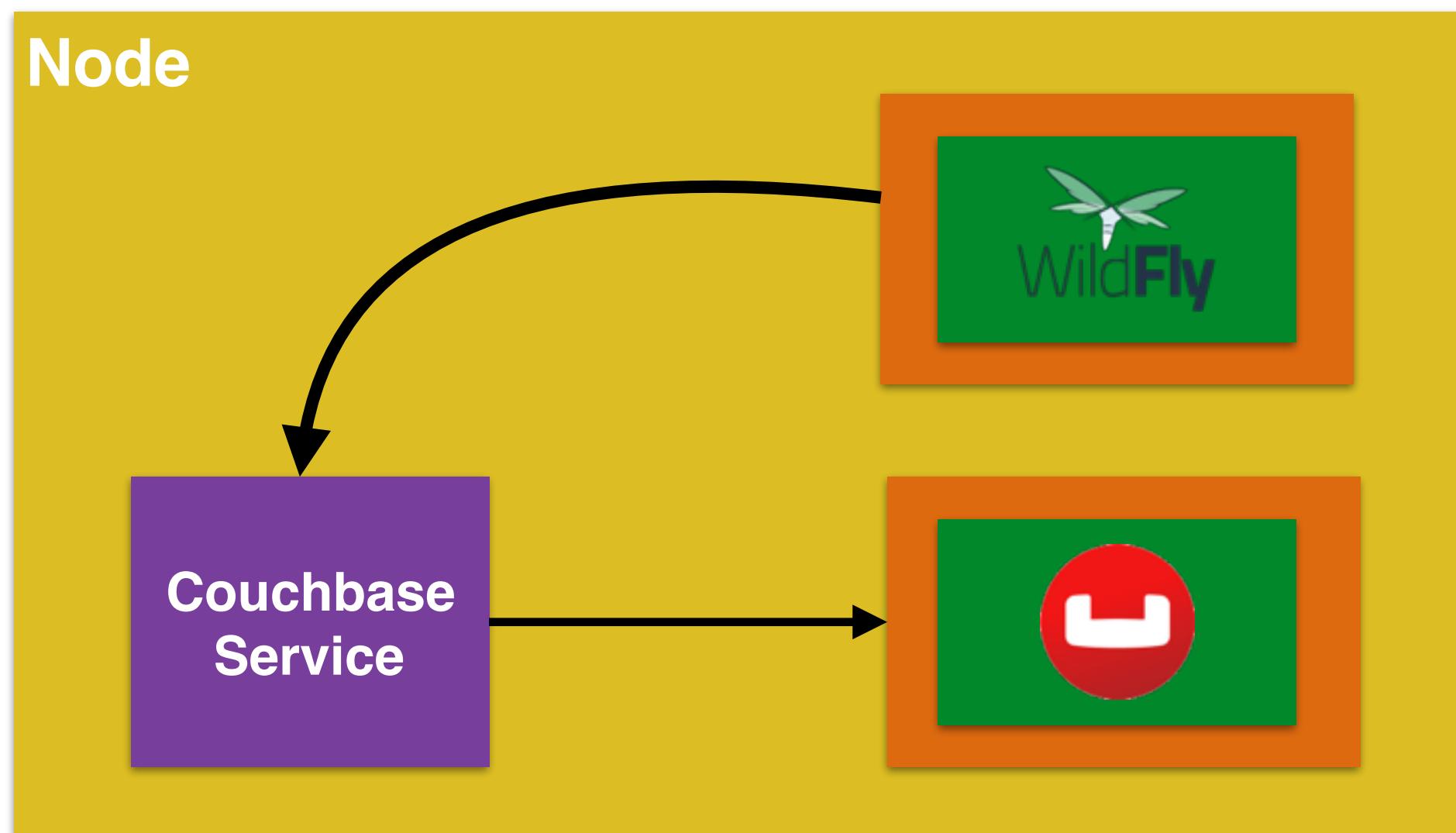
- Abstract a set of pods as a single IP and port
 - Simple TCP/UDP load balancing
- Creates environment variables in other pods or DNS resolution
- Stable endpoint for pods to reference
 - Allows list of pods to change dynamically

Kubernetes Service Configuration

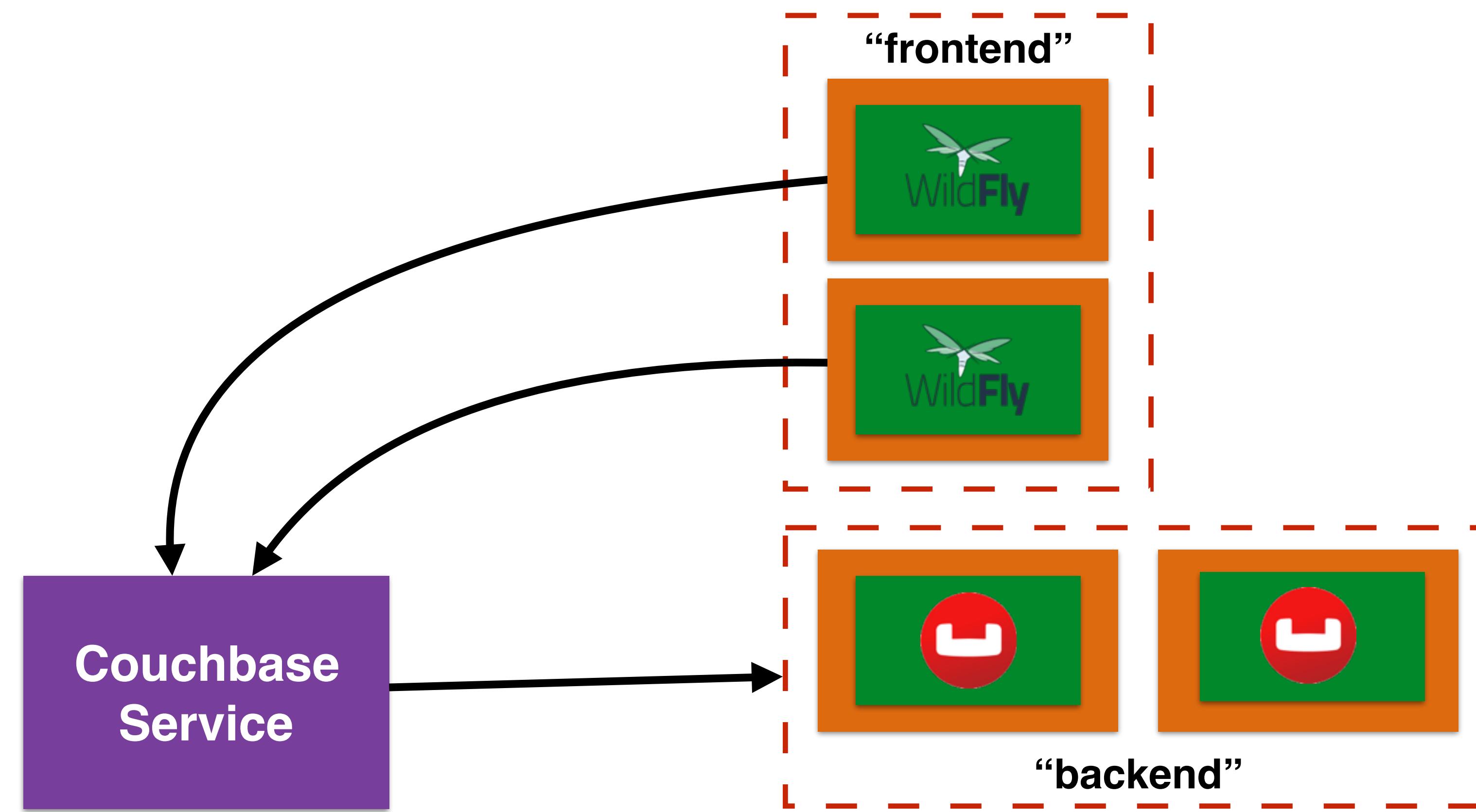
```
1  apiVersion: v1
2  kind: Service
3  metadata:
4    name: couchbase-service
5  spec:
6    selector:
7      app: couchbase-rc-pod
8    ports:
9      - name: admin
10     port: 8091
11      - name: query
12        port: 8093
13  ---
14
15  apiVersion: v1
16  kind: ReplicationController
17  metadata:
18    name: couchbase-rc
19  spec:
20    replicas: 2
21    selector:
22      app: couchbase-rc-pod
23    template:
24      metadata:
25        labels:
26          app: couchbase-rc-pod
27
28        containers:
29          - name: couchbase
30            image: couchbase
31            ports:
32              - containerPort: 8091
```

```
1  apiVersion: v1
2  kind: Service
3  metadata:
4    name: couchbase-service
5  spec:
6    selector:
7      app: couchbase-rc-pod
8    ports:
9      - name: admin
10     port: 8091
11      - name: query
12        port: 8093
13  ---
14
15  apiVersion: v1
16  kind: ReplicationController
17  metadata:
18    name: couchbase-rc
19  spec:
20    replicas: 2
21    selector:
22      app: couchbase-rc-pod
23    template:
24      metadata:
25        labels:
26          app: couchbase-rc-pod
27
28        containers:
29          - name: couchbase
30            image: couchbase
31            ports:
32              - containerPort: 8091
```

Couchbase Service



Service and Replication Controller

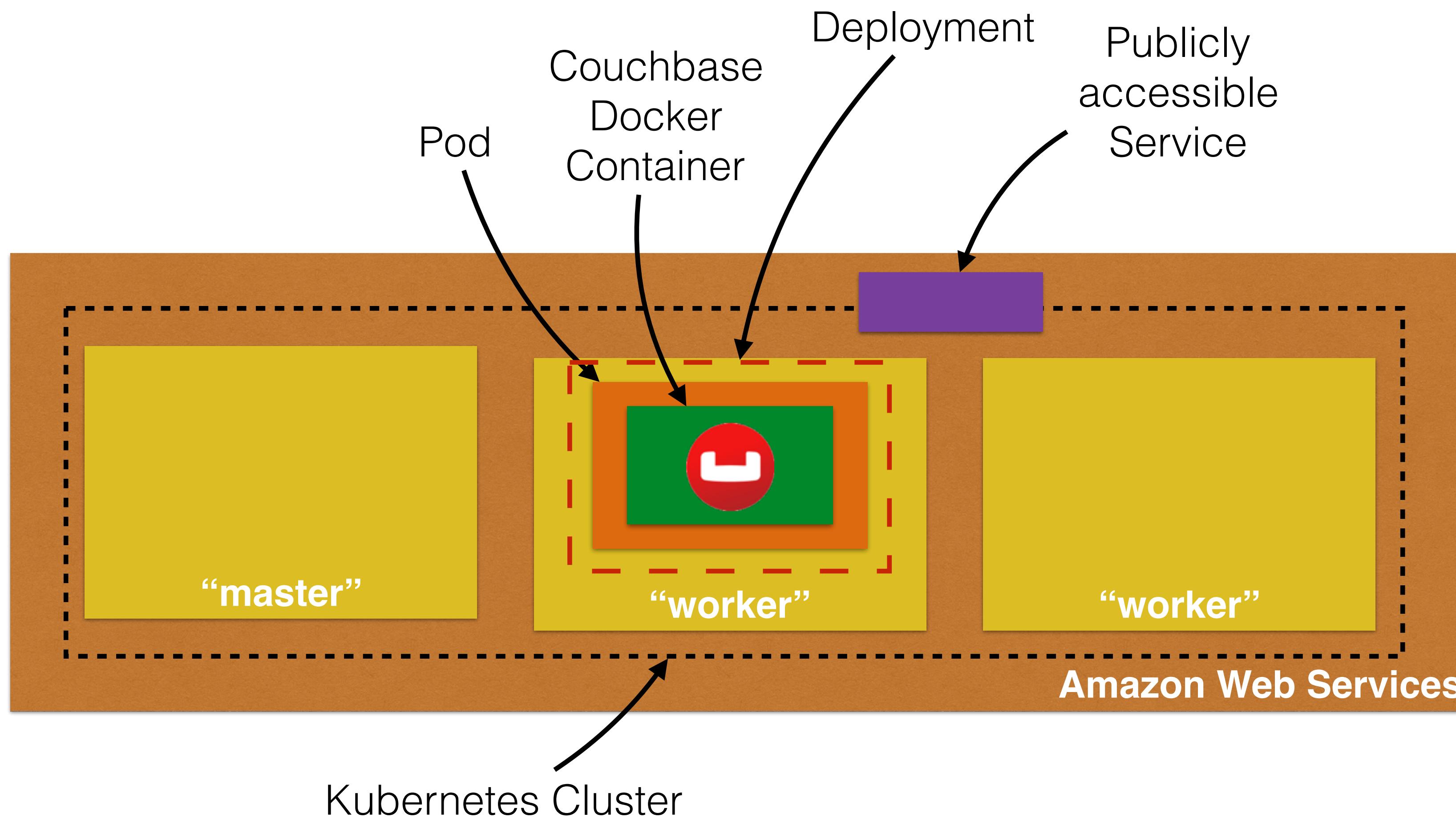


Exposing Service

- Service may be exposed outside cluster or on Internet using **type**
 - **ClusterIP** (default)
 - **NodePort**: A port on each node

```
spec:  
  type: NodePort  
  ports:  
    - name: admin  
      port: 8091  
      nodePort: 30001  
    - name: query  
      port: 8093  
      nodePort: 30002
```
 - **LoadBalancer**: On cloud providers that support external LB

Exposing Service on Amazon



Deployment

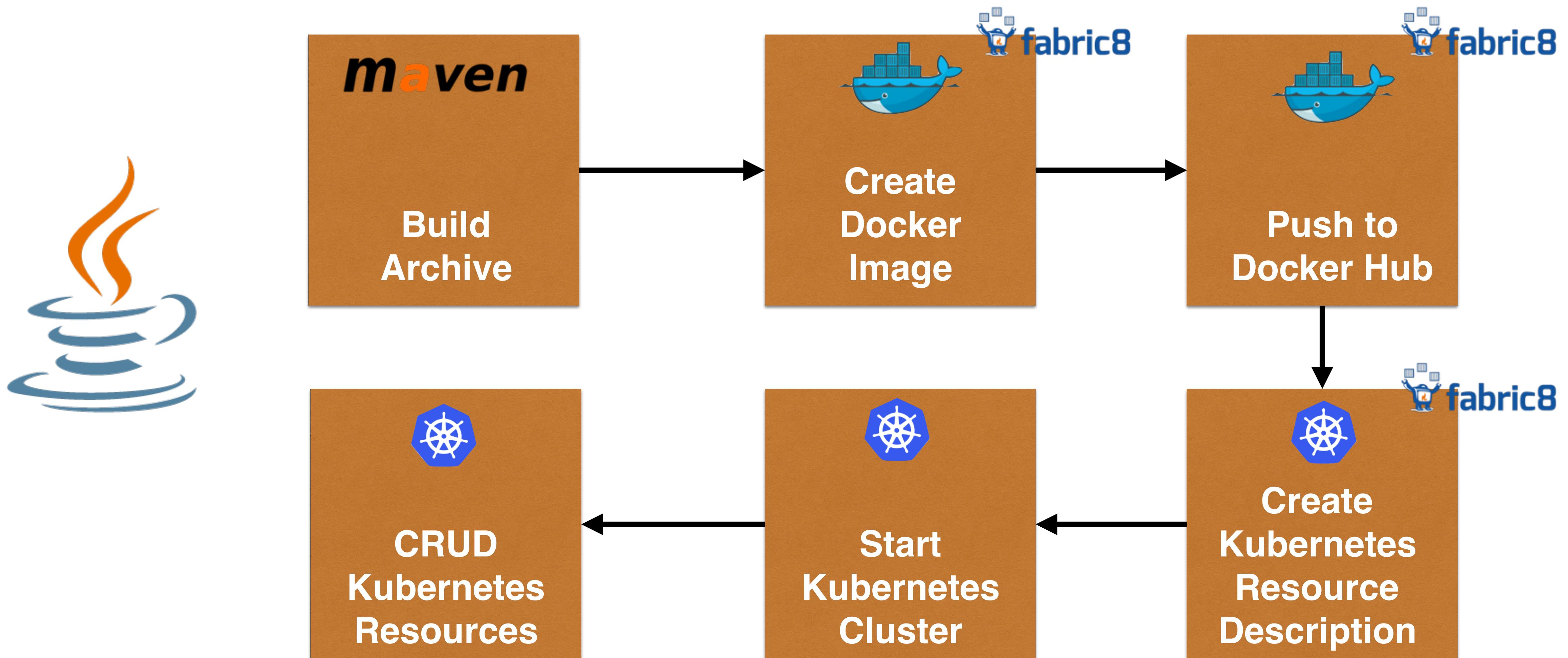
- Declarative updates for pods and replica sets
 - For example: rolling updates
- Differences from `kubectl`
 - Declarative instead of imperative
 - Server-side, and so is faster
 - More features, e.g. rollback to previous version

Deployment Configuration

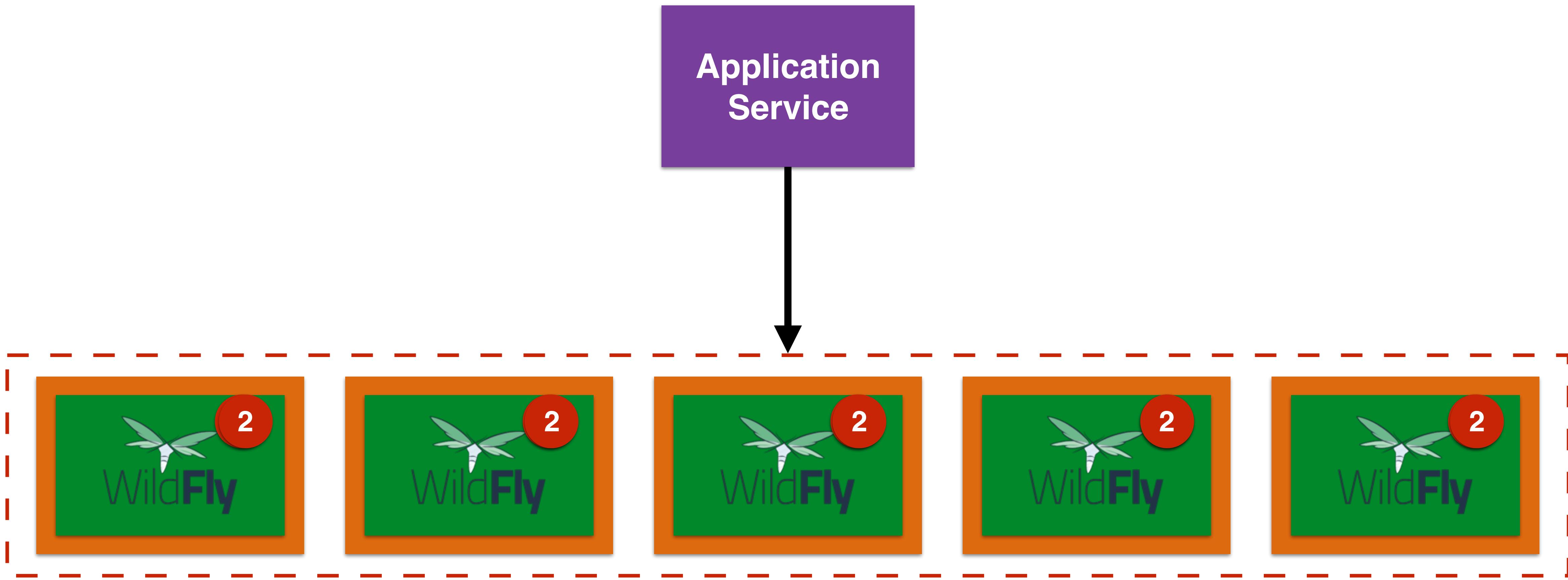
```
1  apiVersion: extensions/v1beta1
2  kind: Deployment
3  metadata:
4    name: wildfly-deployment
5  spec:
6    replicas: 3
7    template:
8      metadata:
9        labels:
10          app: wildfly
11        spec:
12          containers:
13            - name: wildfly
14              image: jboss/wildfly
15            ports:
16              - containerPort: 8080
```

CRUD Kubernetes Resources

Kubernetes and Java Developers



Rolling Updates

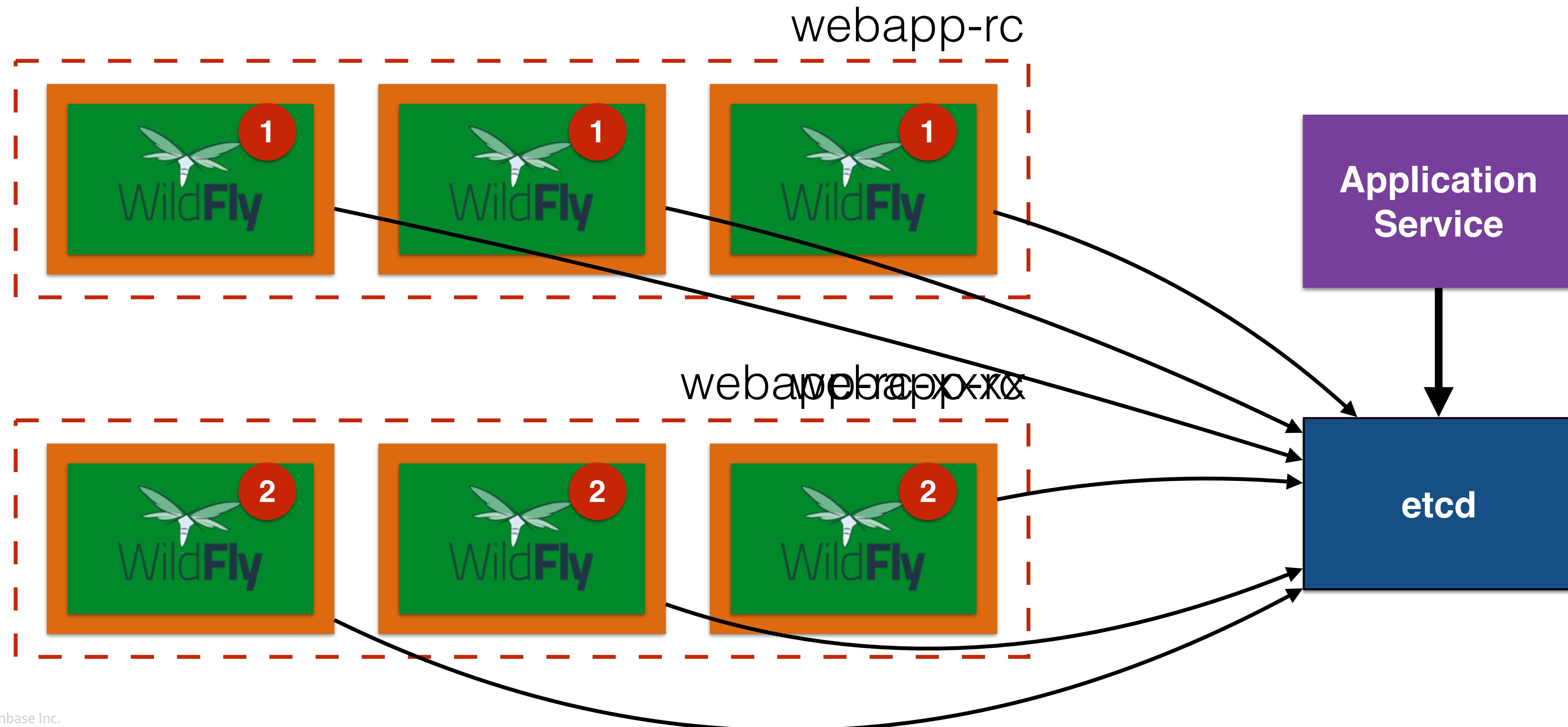


Rolling Updates - Replication Controller

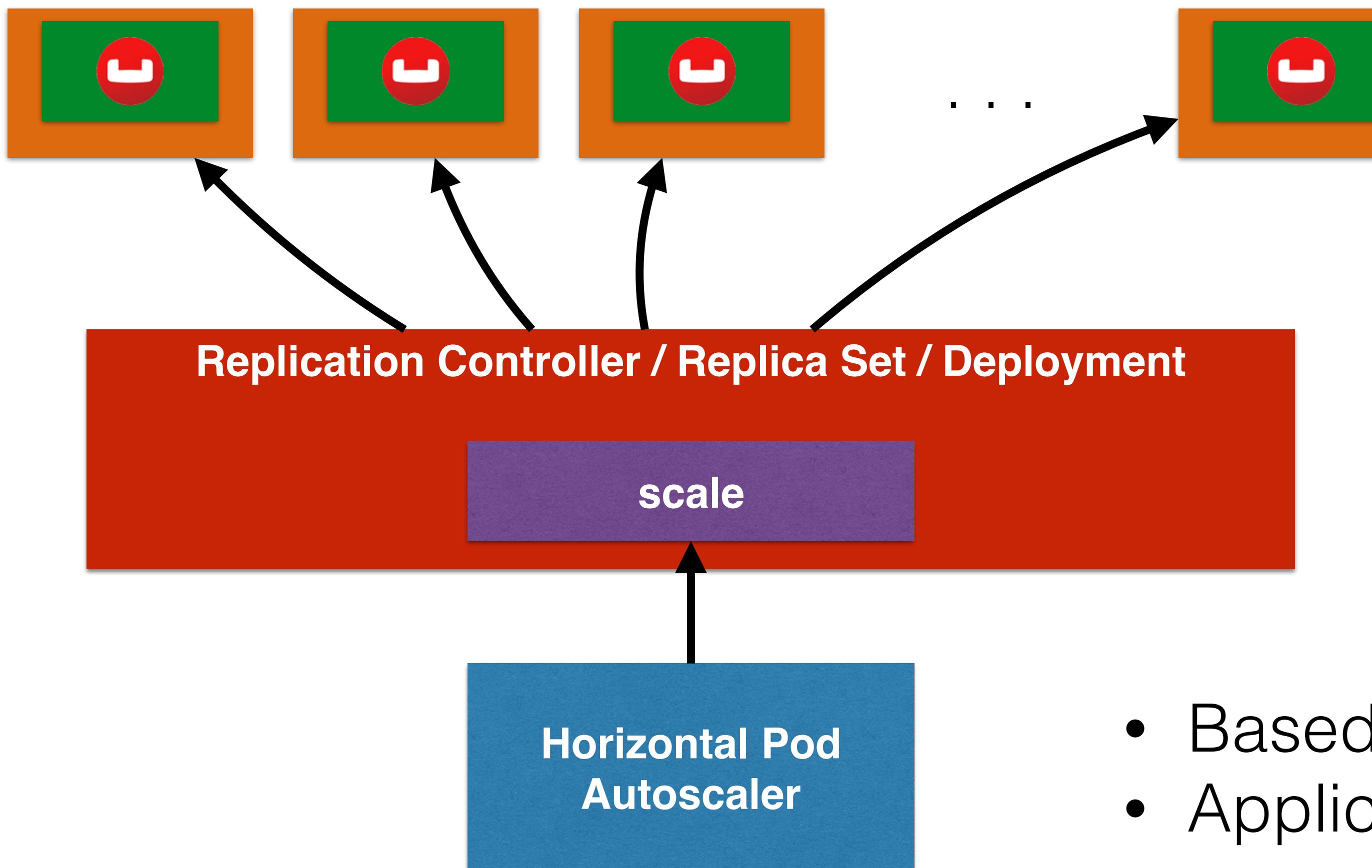
- `kubectl rolling-update webapp-rc -f webapp-rc2.json`
- `kubectl rolling-update webapp-rc --image=arungupta/wildfly-app:2`

```
Created webapp-rc-5a11f15230716f6026c407eb9c1a60ca
Scaling up webapp-rc-5a11f15230716f6026c407eb9c1a60ca from 0 to 2, scaling down
webapp-rc from 2 to 0 (keep 2 pods available, don't exceed 3 pods)
Scaling webapp-rc-5a11f15230716f6026c407eb9c1a60ca up to 1
Scaling webapp-rc down to 1
Scaling webapp-rc-5a11f15230716f6026c407eb9c1a60ca up to 2
Scaling webapp-rc down to 0
Update succeeded. Deleting old controller: webapp-rc
Renaming webapp-rc-5a11f15230716f6026c407eb9c1a60ca to webapp-rc
replicationcontroller "webapp-rc" rolling updated
```

Rolling Updates



Horizontal Pod Autoscaling



- Based on observed CPU utilization
- Application provided metrics (health)

Horizontal Pod Autoscaling

- Typical usage
 - `kubectl autoscale deployment | rc | rs --min=<PODS> --max=<PODS> --cpu-percent=<CPU>`
- Autoscale a deployment with number of pods between 2 and 10
 - `kubectl autoscale deployment db --min=2 --max=10`
- Autoscale a Replication with maximum number of pods 5, target CPU utilization of 80%
 - `kubectl autoscale rc --max=5 --cpu-percent=80`

HPA Configuration

```
1 apiVersion: autoscaling/v1
2 kind: HorizontalPodAutoscaler
3 metadata:
4   name: wildfly-scaler
5 spec:
6   scaleTargetRef:
7     kind: ReplicaSet
8     name: wildfly-rs
9   minReplicas: 3
10  maxReplicas: 10
11  targetCPUUtilizationPercentage: 50
```

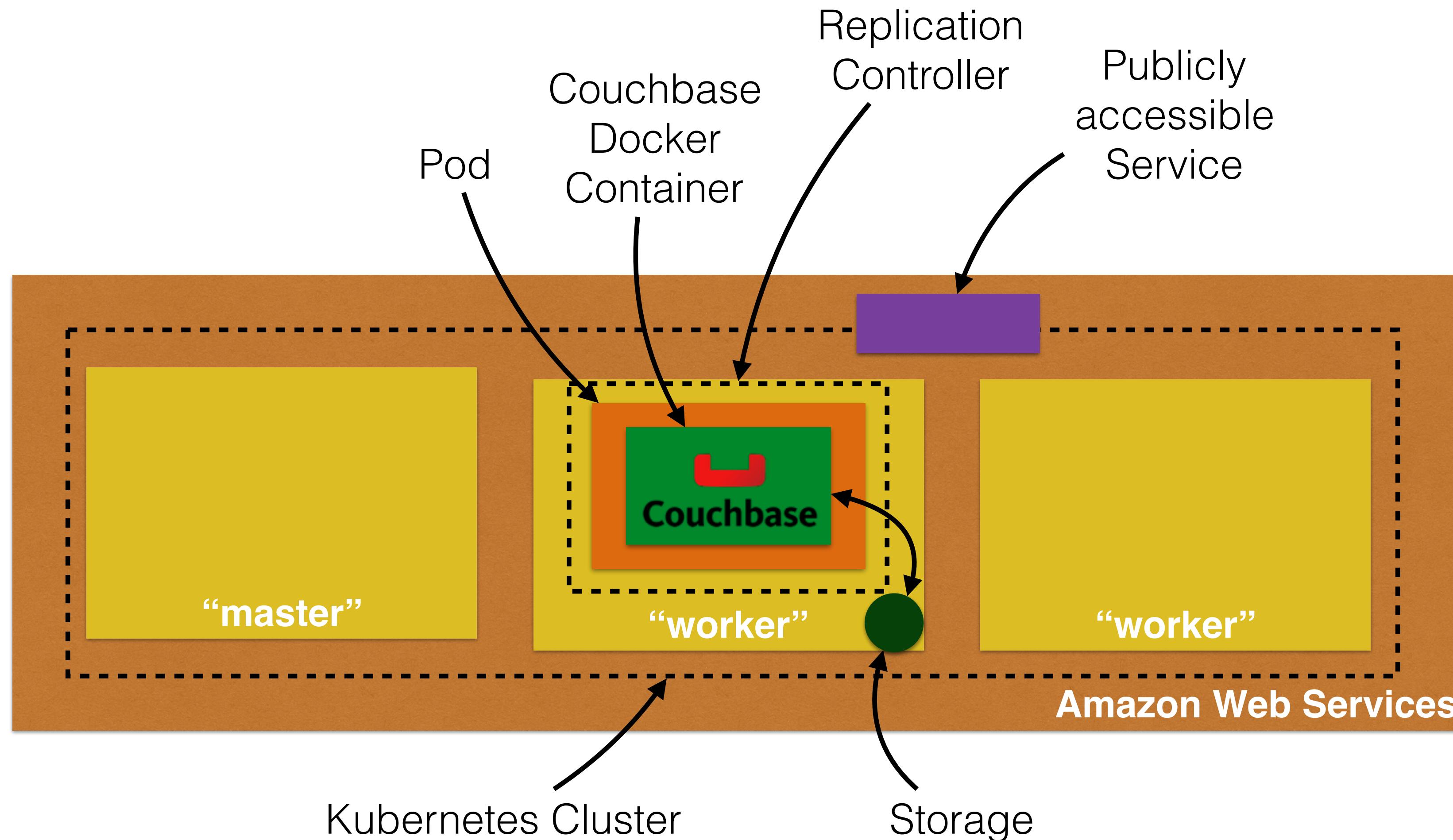
Volumes

- Directory accessible to the containers in a pod
- Volume outlives any containers in a pod
- Common types
 - hostPath
 - nfs
 - awsElasticBlockStore
 - gcePersistentDisk

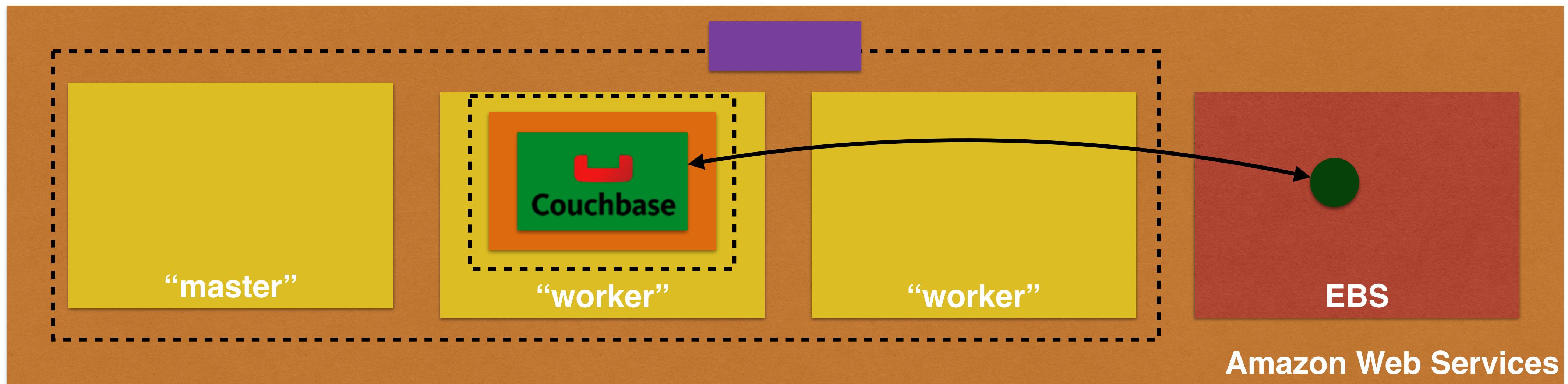
Volume Configuration

```
1  apiVersion: v1
2  kind: Pod
3  metadata:
4    name: couchbase-pod
5    labels:
6      name: couchbase-pod
7  spec:
8    containers:
9      - name: couchbase
10        image: arungupta/couchbase-oreilly:k8s
11    ports:
12      - containerPort: 8091
13    volumeMounts:
14      - mountPath: /var/couchbase/lib
15        name: couchbase-data
16    volumes:
17      - name: couchbase-data
18        hostPath:
19          path: /opt/data
```

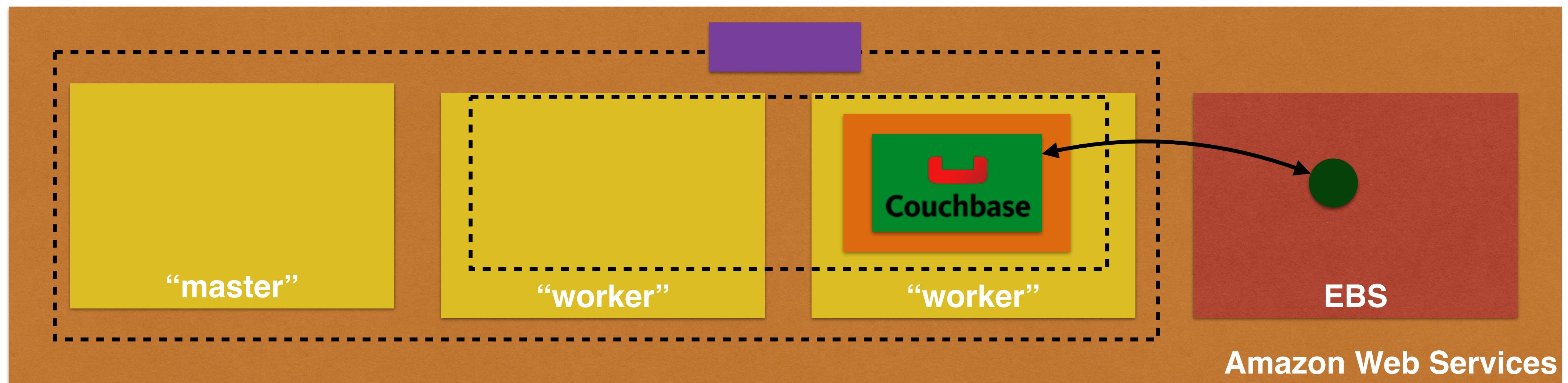
Stateful Containers



Stateful Containers



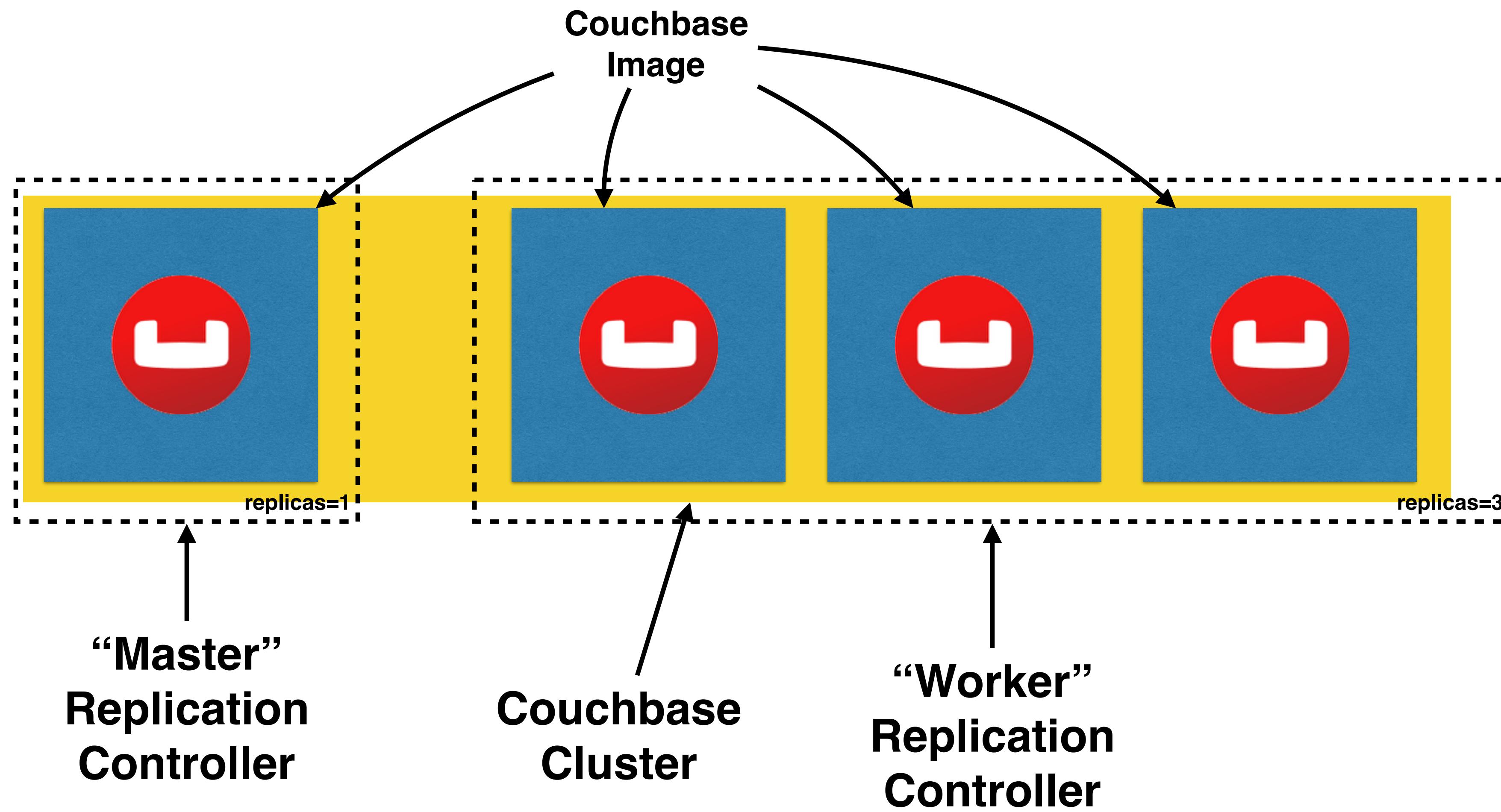
Stateful Containers



Persistent Volume

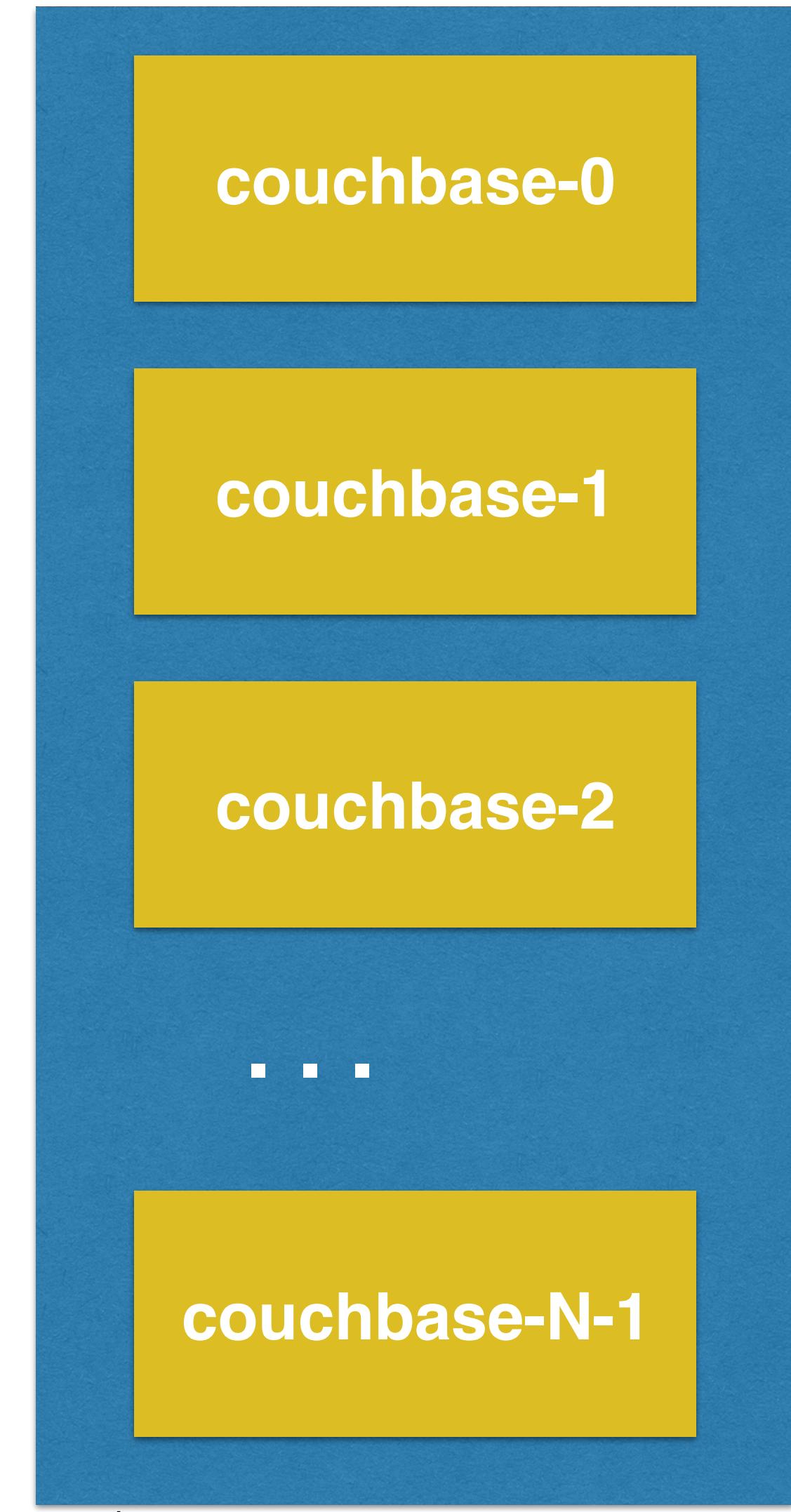


Couchbase Cluster on Kubernetes



Pet Set

- Alpha resource introduced in 1.3
- Stateful pods
- PetSet has 0..N-1 Pets
- Each Pet has a deterministic name, and a unique identity
 - stable hostname
 - ordinal index
 - stable storage linked to ordinal & hostname
- Each Pet has at most one pod
- Each Pet Set has at most one Pet with a given identity



Multitenancy - Namespace

- Namespace allows to partition resources into a logical group
- Each namespace provides:
 - **scope** for resources to avoid collisions
 - **policies** to ensure appropriate authority to trusted users
 - **constraints** for resource consumption
- Anti-pattern: Separate slightly different resources, e.g. different version
 - Use labels instead

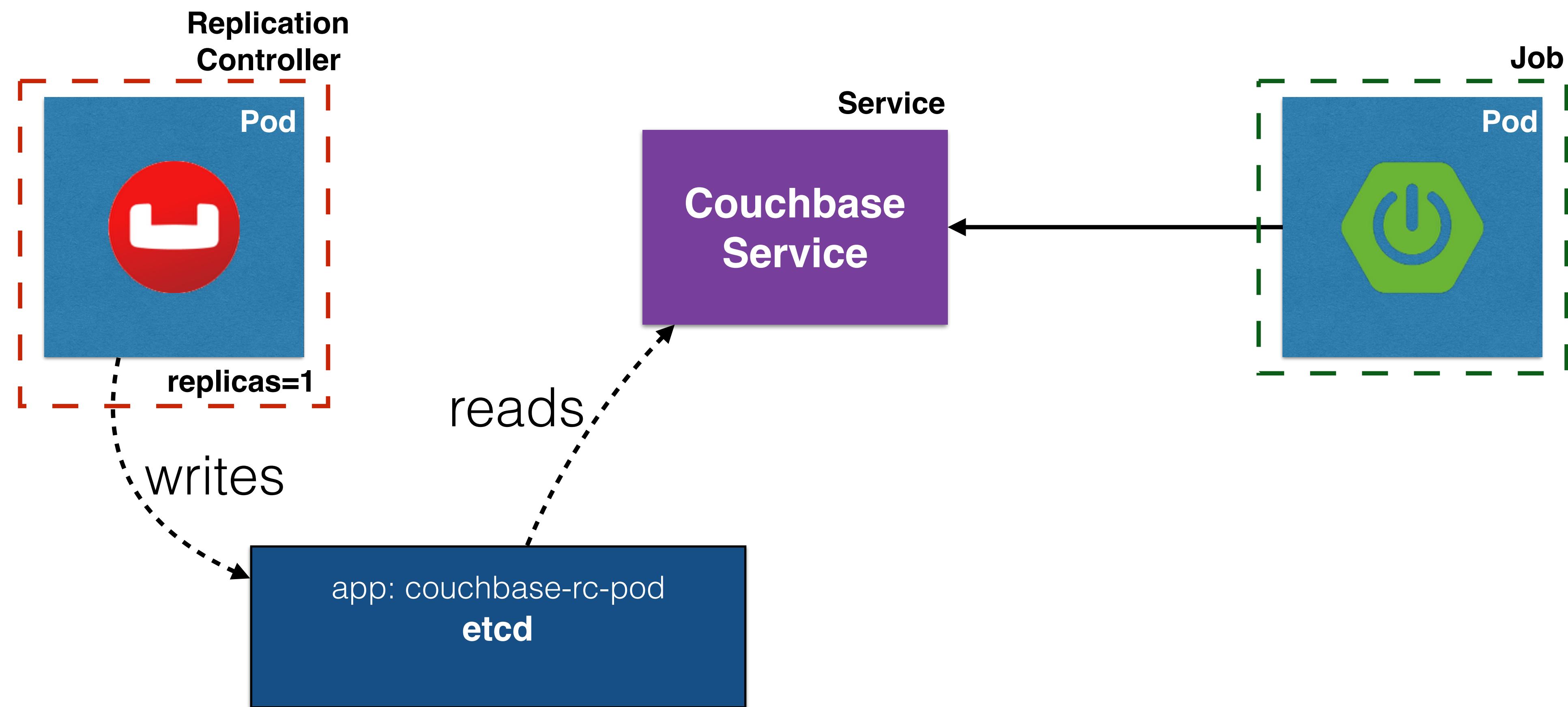
Batch Jobs

- Run-once jobs
 - Replication Controller, Replica Set or Deployments not suitable
- Three types
 - Non-parallel: only one pod is started
 - Parallel:
 - With a fixed completion count
 - With a work queue

Run-Once Job Specification

```
1  apiVersion: batch/v1
2  kind: Job
3  metadata:
4    name: wait
5  spec:
6    template:
7      metadata:
8        name: wait
9      spec:
10        containers:
11          - name: wait
12            image: ubuntu
13            command: ["sleep", "20"]
14        restartPolicy: Never
```

Java Application in Kubernetes

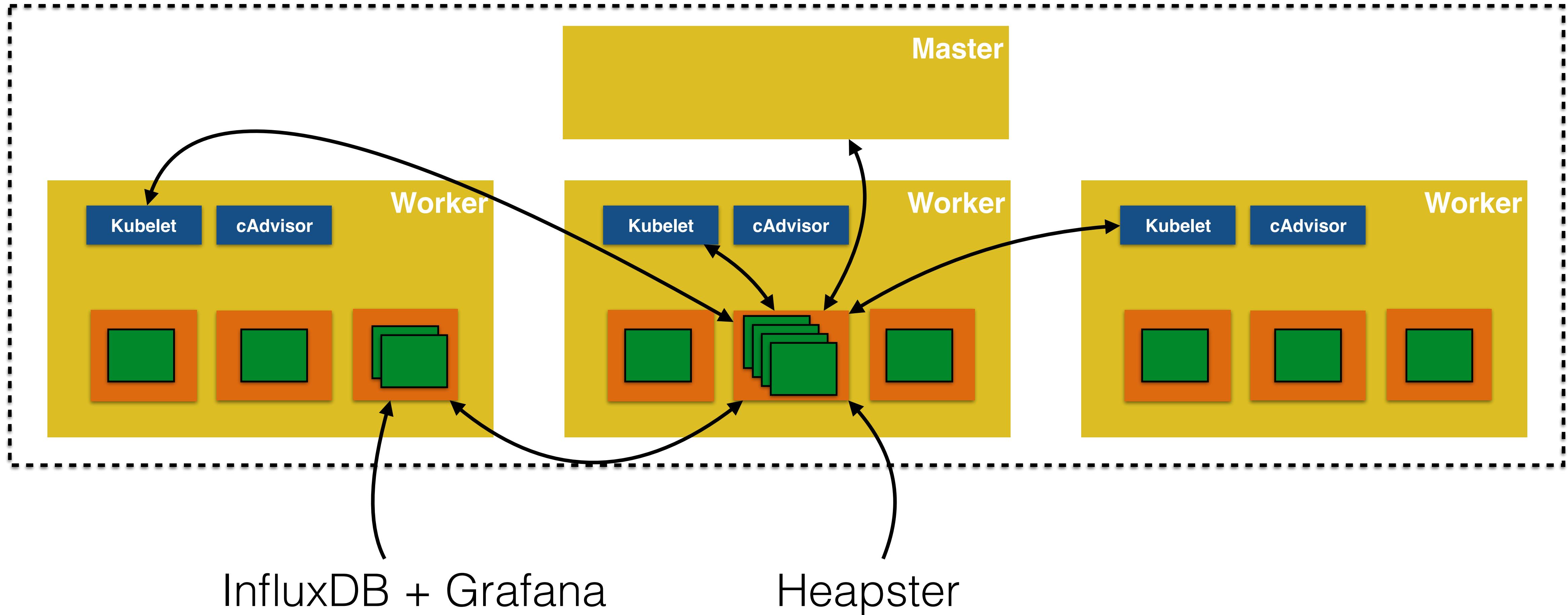


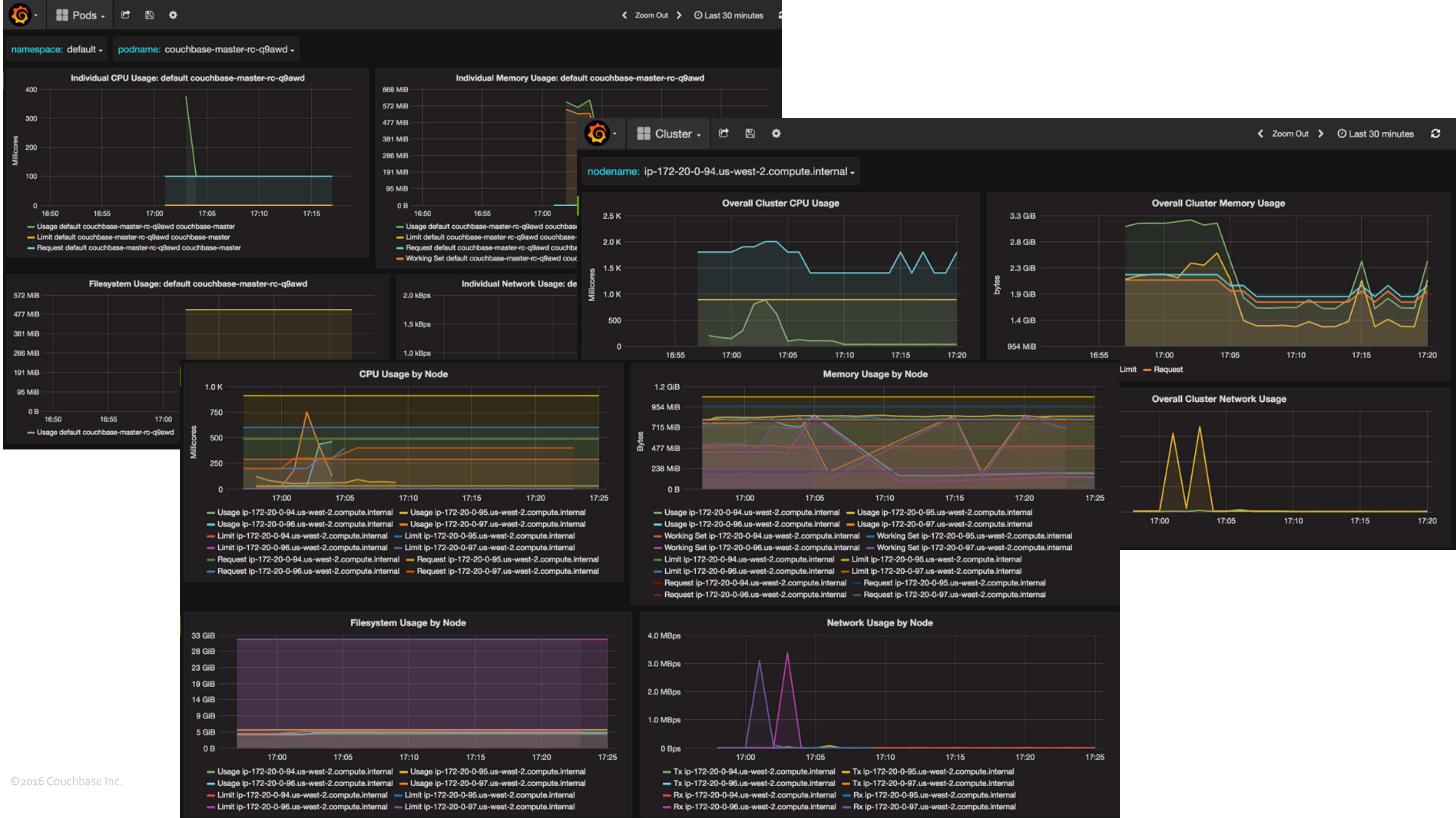
Health Checks

- Health checks performed by Kubelet
- Probes - Liveness and Readiness
- Diagnostics
 - Container Exec
 - TCP Socket
 - HTTP

```
1  apiVersion: extensions/v1beta1
2  kind: Deployment
3  metadata:
4    name: couchbase
5  spec:
6    replicas: 1
7    template:
8      metadata:
9        labels:
10       app: couchbase
11     spec:
12       containers:
13         - name: couchbase
14           image: arungupta/couchbase
15         ports:
16           - containerPort: 8091
17         livenessProbe:
18           httpGet:
19             path: /pools
20             port: 8091
21             initialDelaySeconds: 30
22             timeoutSeconds: 1
```

Monitoring Kubernetes Resources

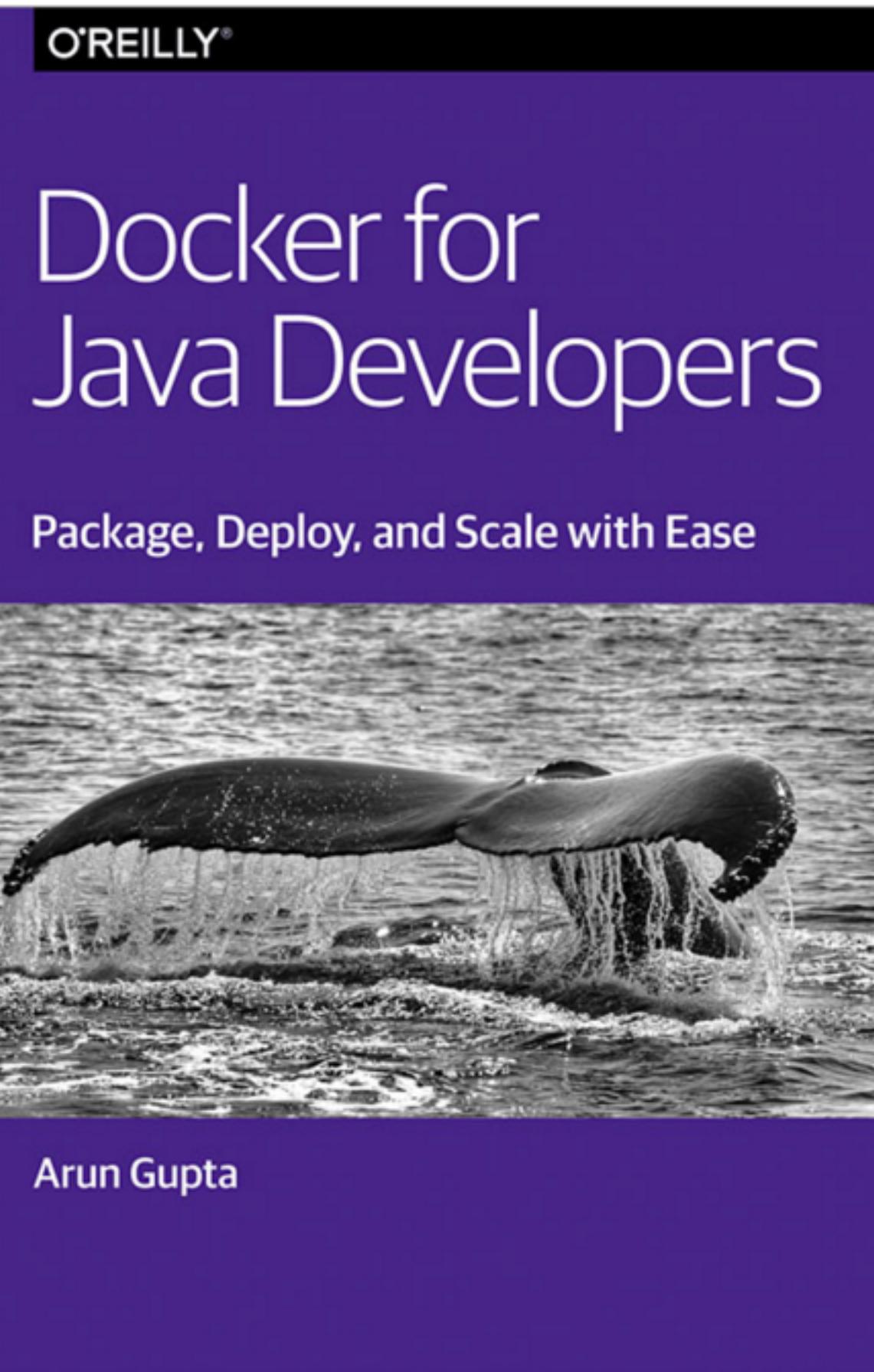




Kubernetes Monitoring



bit.ly/dockerjava



bit.ly/kubejava



References

- github.com/arun-gupta/kubernetes-java-sample
- kubernetes.io
- Containers recipe: couchbase.com/containers