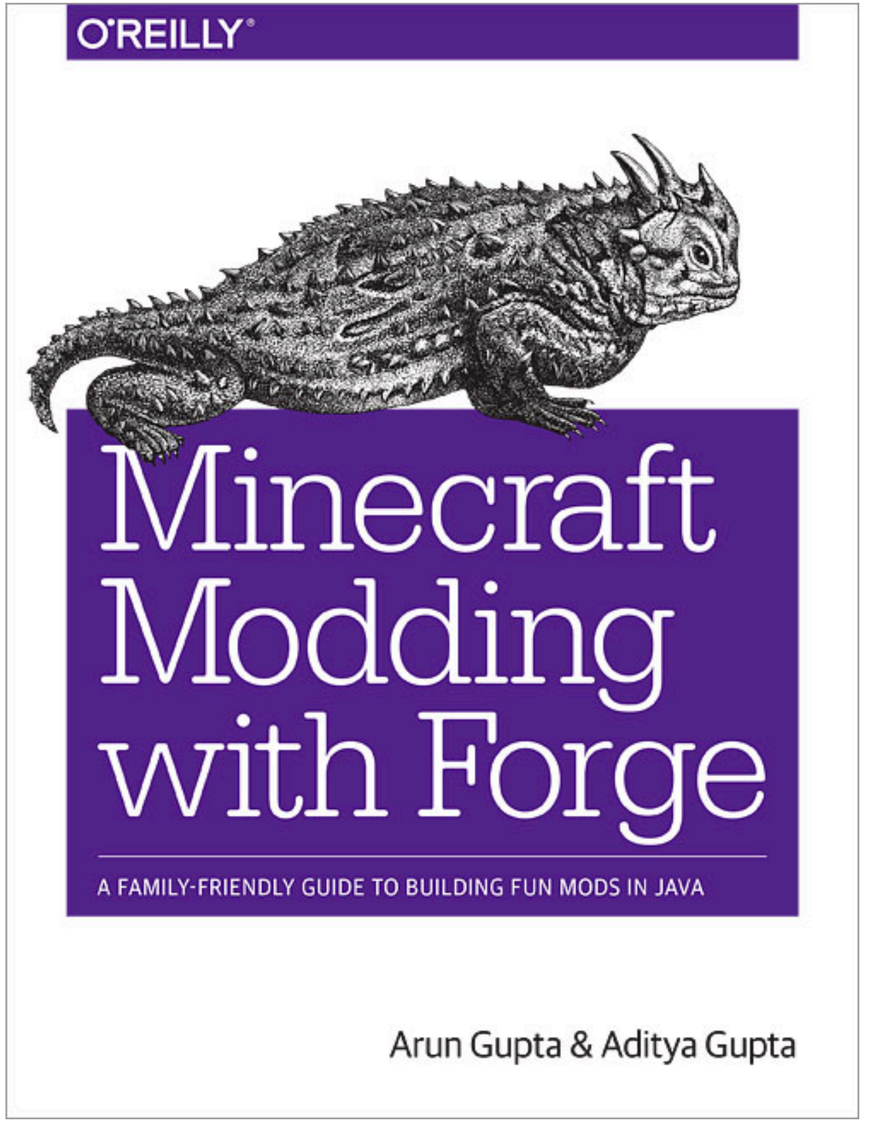
Getting Started with Kubernetes

Arun Gupta, @arungupta
VP Developer Advocacy, Couchbase

Couchbase

DEVOXX™ 4KIDS USA

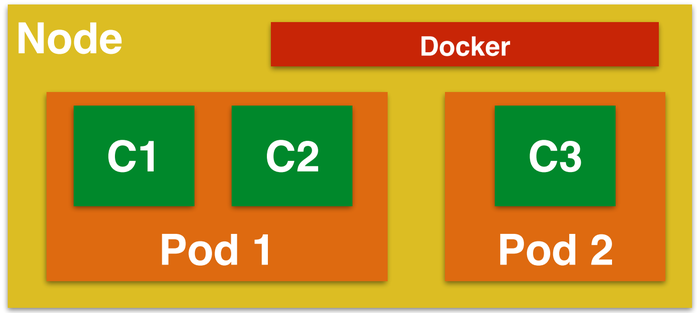


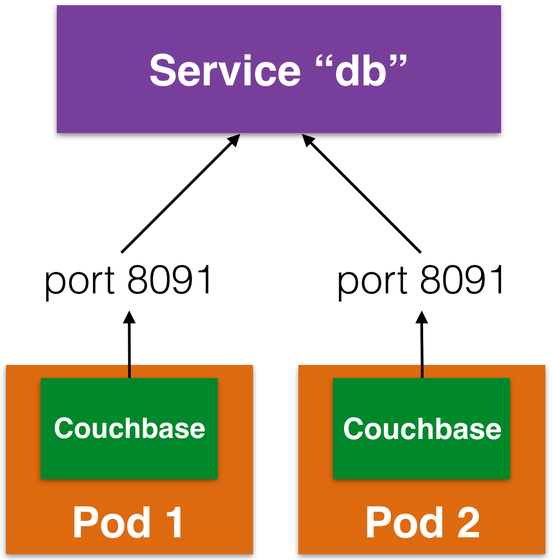
Kubernetes

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

Concepts

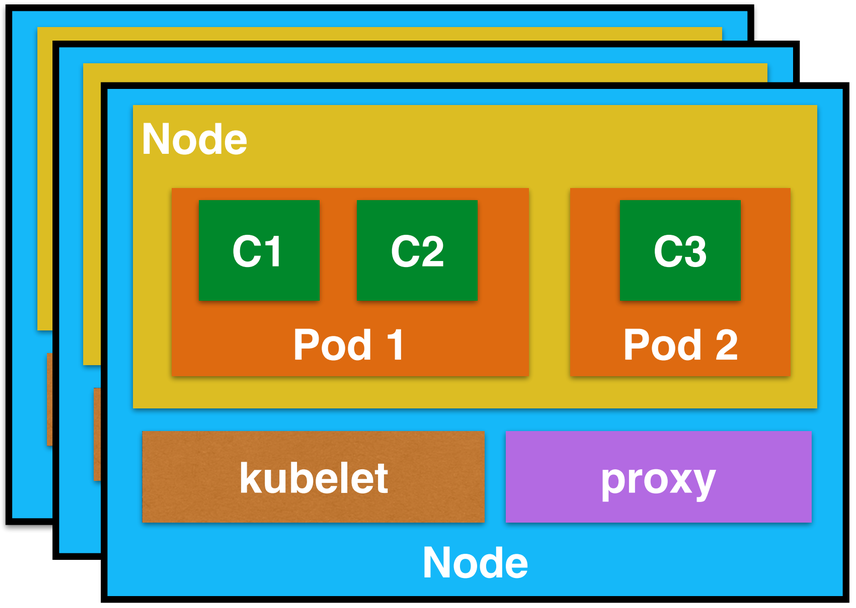
- **Pods**: collocated group of Docker containers that share an IP and storage volume
- **Service**: Single, stable name for a set of pods, also acts as LB
- **Label**: used to organize and select group of objects
- **Replication Controller**: manages the lifecycle of pods and ensures specified number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

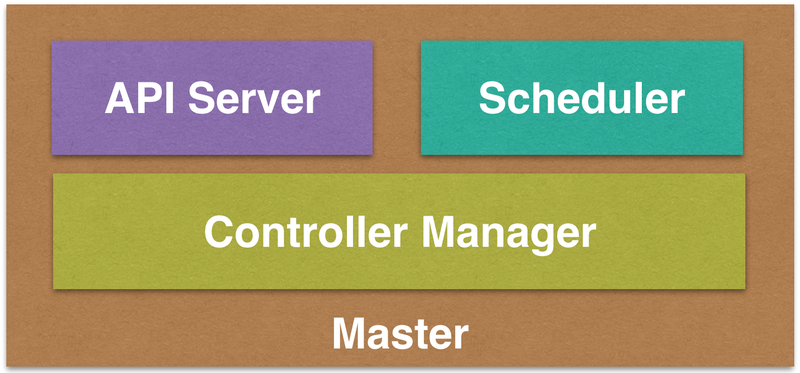




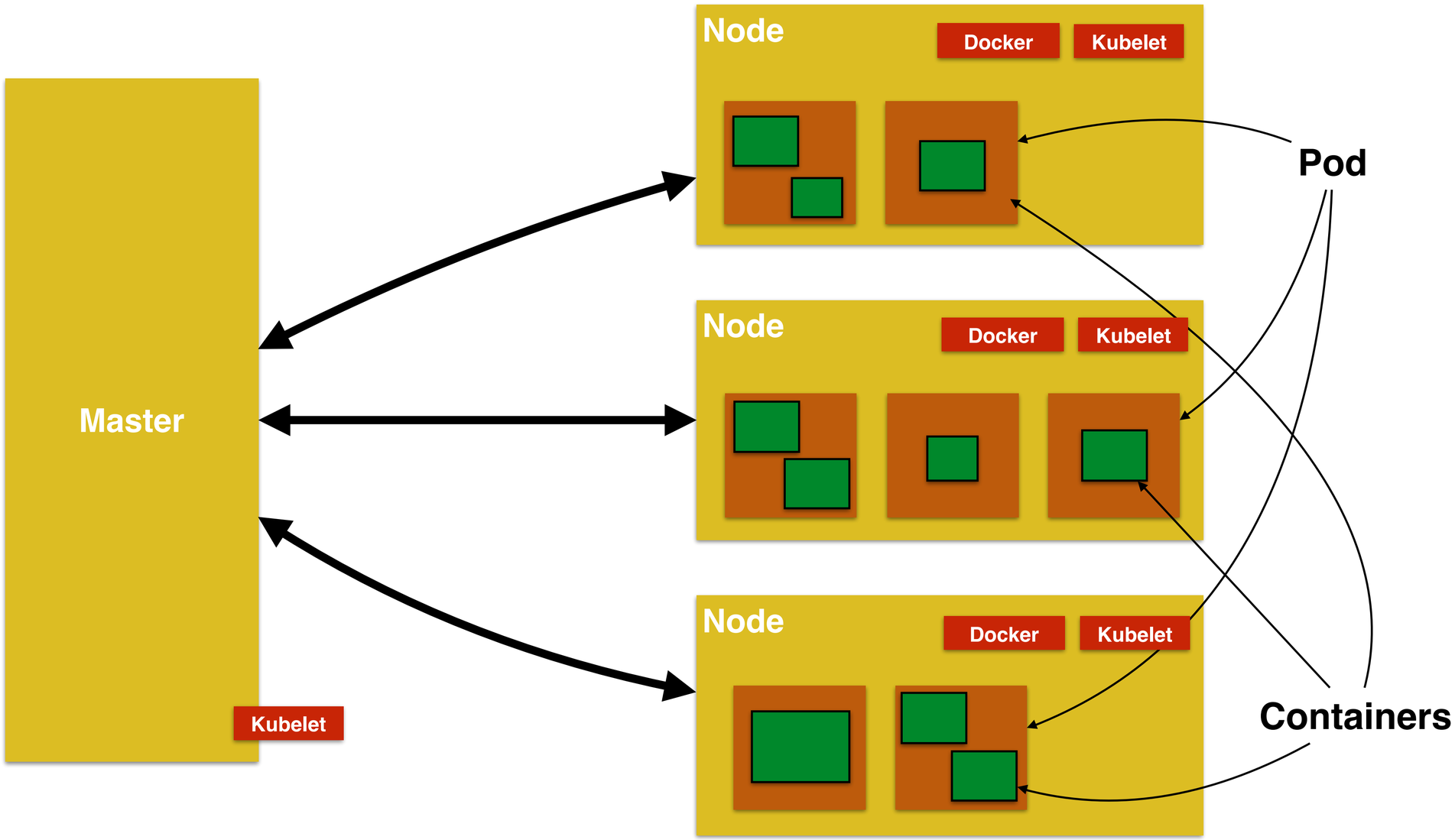
Components

- **Node**: Docker host running *kubelet* (node agent) and *proxy* services
  - Monitored by *systemd* (CentOS) or *monit* (Debian)
- **Master**: hosts cluster-level control services, including the API server, scheduler, and controller manager
- **etcd**: distributed key-value store used to persist Kubernetes system state





Architecture



Master High Availability

- Hack by running a `podmaster` utility
- Proposal
  - Hot Standby
  - Warm Standby
  - Active-Active (Load Balanced)

http://kubernetes.io/v1.1/docs/proposals/high-availability.html

kubectl

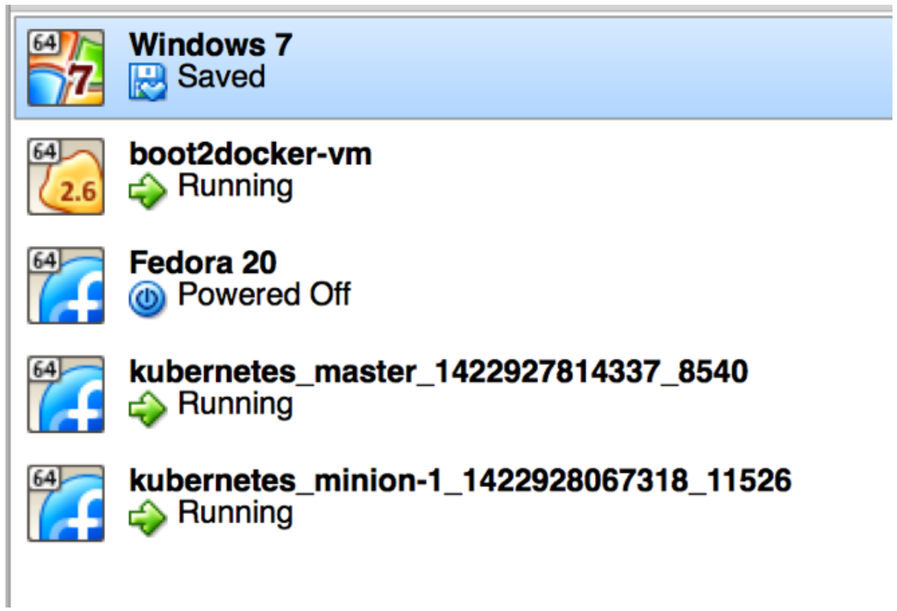
- Controls the Kubernetes cluster manager
- `kubectl get pods` or `minions`
- `kubectl create -f <filename>`
- `kubectl update` or `delete`
- `kubectl resize –replicas=3 replicationcontrollers <name>`

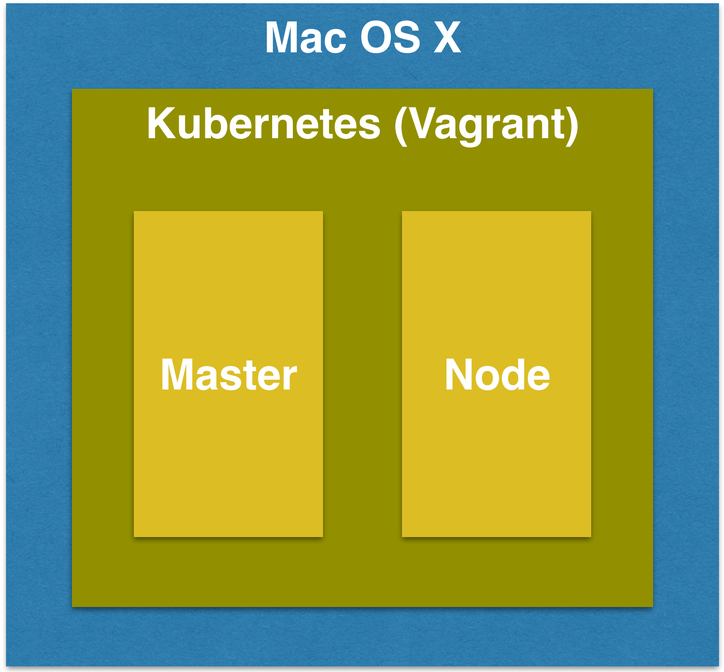
Kubernetes Config

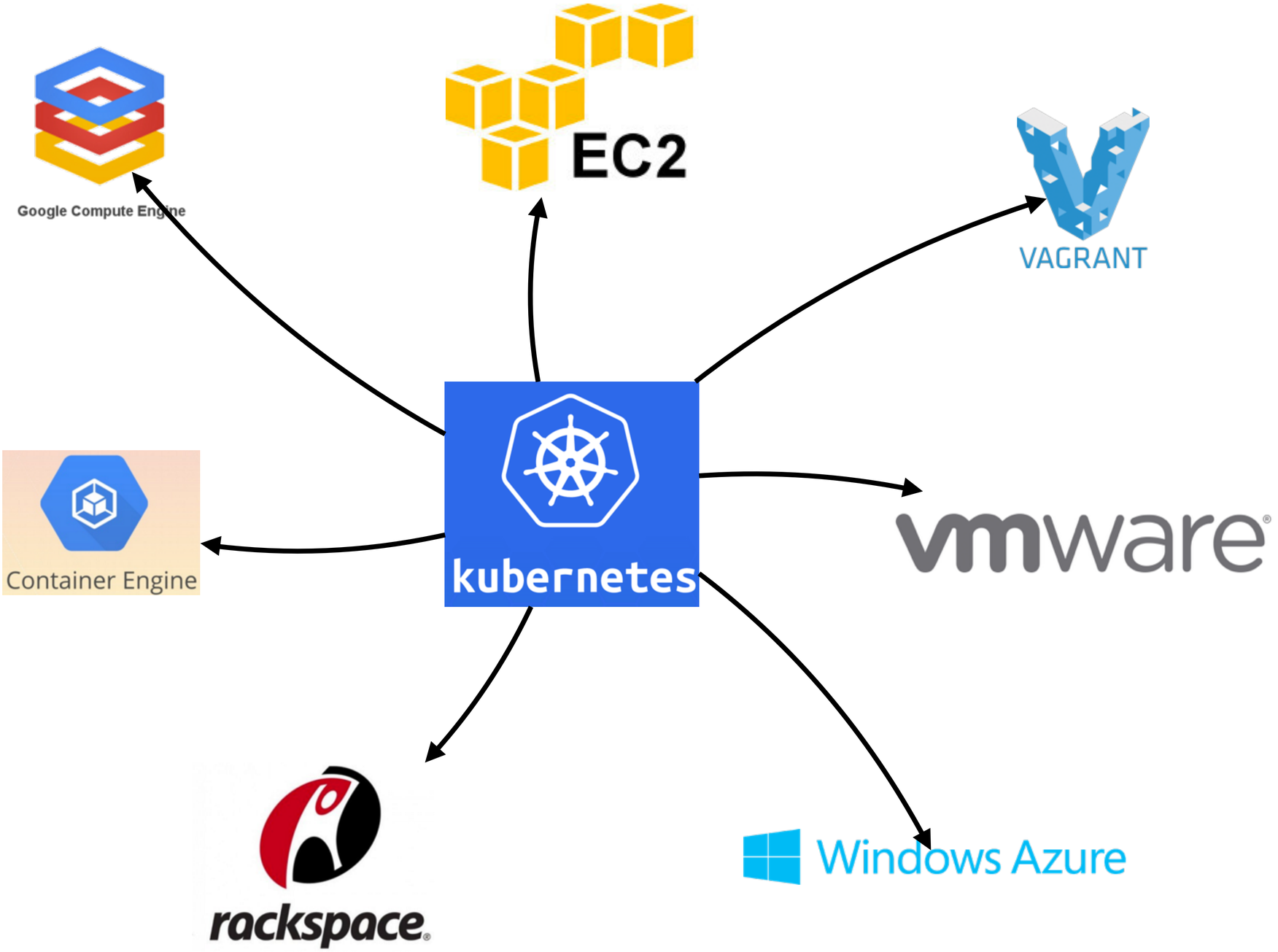
```yaml
apiVersion: v1
kind: Pod
metadata:
  name: wildfly-pod
  labels:
    name: wildfly
spec:
  containers:
    - image: jboss/wildfly
      name: wildfly-pod
      ports:
        - containerPort: 8080
```

```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
```

```
export KUBERNETES_PROVIDER=vagrant
./cluster/kube-up.sh
```



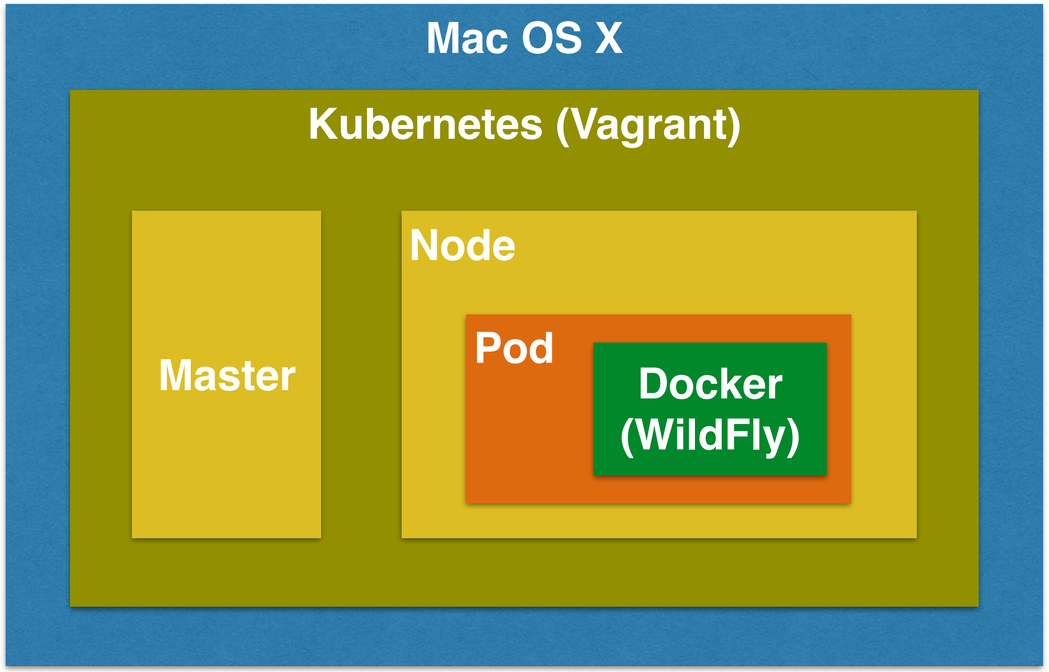




https://get.k8s.io/

A Pod with One Container

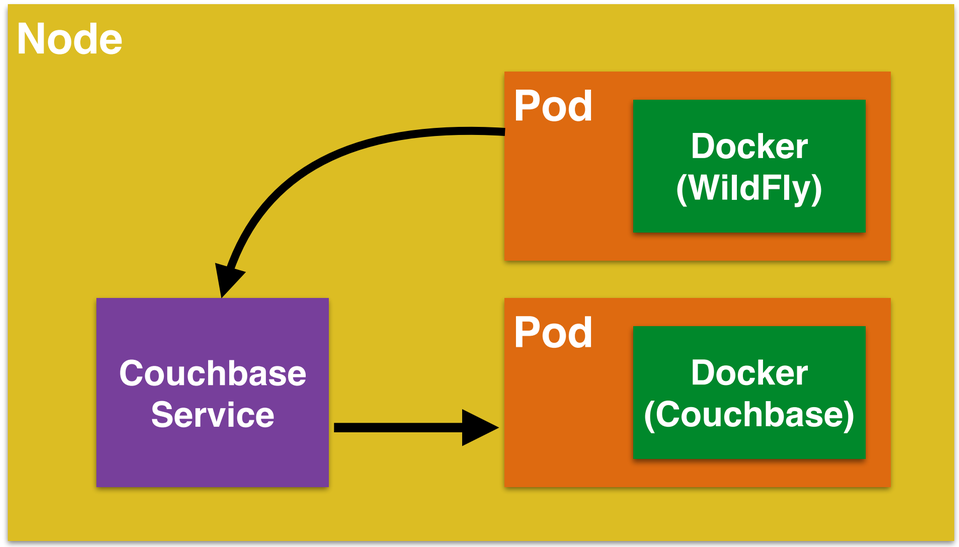
```yaml
apiVersion: v1
kind: Pod
metadata:
  name: wildfly-pod
  labels:
    name: wildfly
spec:
  containers:
    - image: jboss/wildfly
      name: wildfly-pod
      ports:
        - containerPort: 8080
```



Services

- Abstract a set of pods as a single IP and port
  - Simple TCP/UDP load balancing
- Creates environment variables in other pods
  - Like “Docker links” but across hosts
- Stable endpoint for pods to reference
  - Allows list of pods to change dynamically

Services

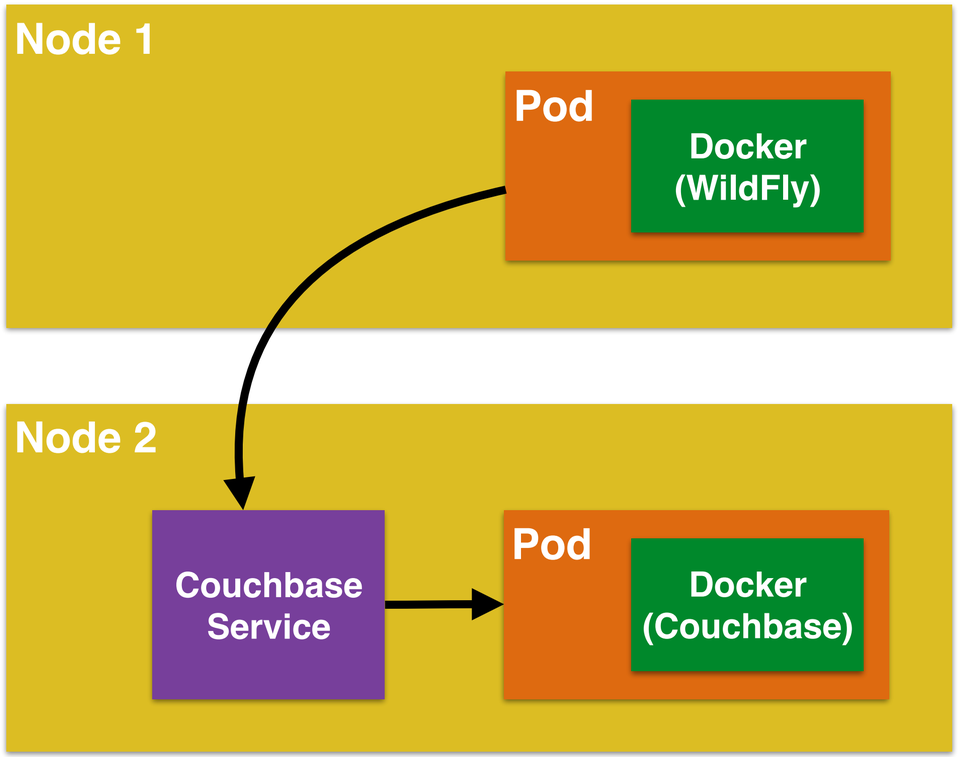


Services

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: mysql-pod
  labels:
    name: mysql-pod
    context: docker-k8s-lab
spec:
  containers:
    -
      name: mysql
      image: mysql:latest
      env:
        -
          name: "MYSQL_USER"
          value: "mysql"
        -
          name: "MYSQL_PASSWORD"
          value: "mysql"
        -
          name: "MYSQL_DATABASE"
          value: "sample"
        -
          name: "MYSQL_ROOT_PASSWORD"
          value: "supersecret"
      ports:
        -
          containerPort: 3306
```

```yaml
apiVersion: v1
kind: Service
metadata:
  name: mysql-service
  labels:
    name: mysql-pod
    context: docker-k8s-lab
spec:
  ports:
    # the port that this service should
    - port: 3306
  # label keys and values that must mat
  selector:
    name: mysql-pod
    context: docker-k8s-lab
```

Service across Two Nodes



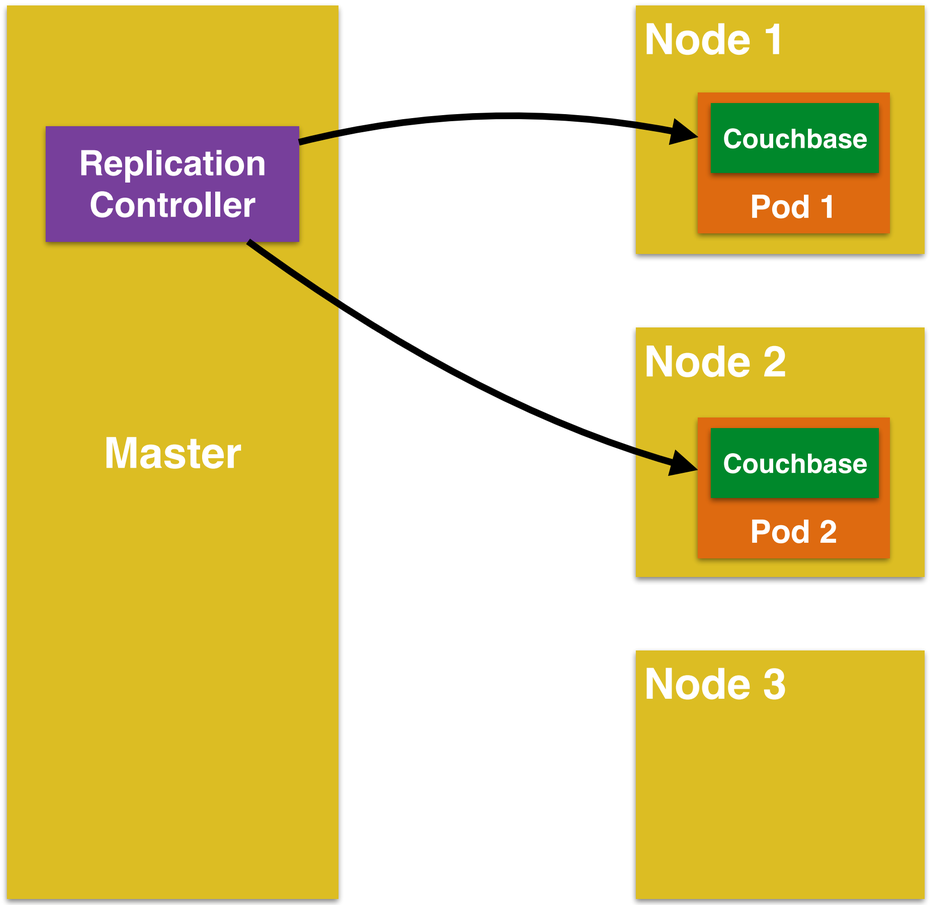
Replication Controller

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

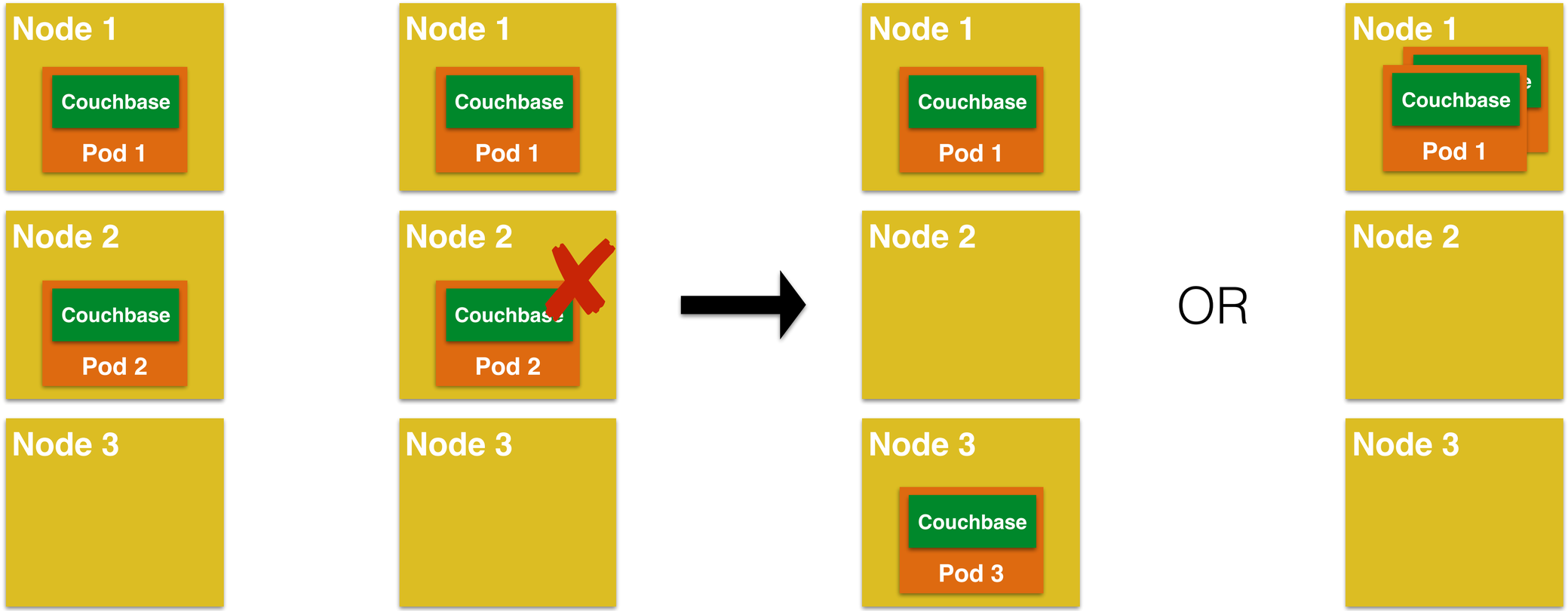
Replication Controller Configuration

```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
```

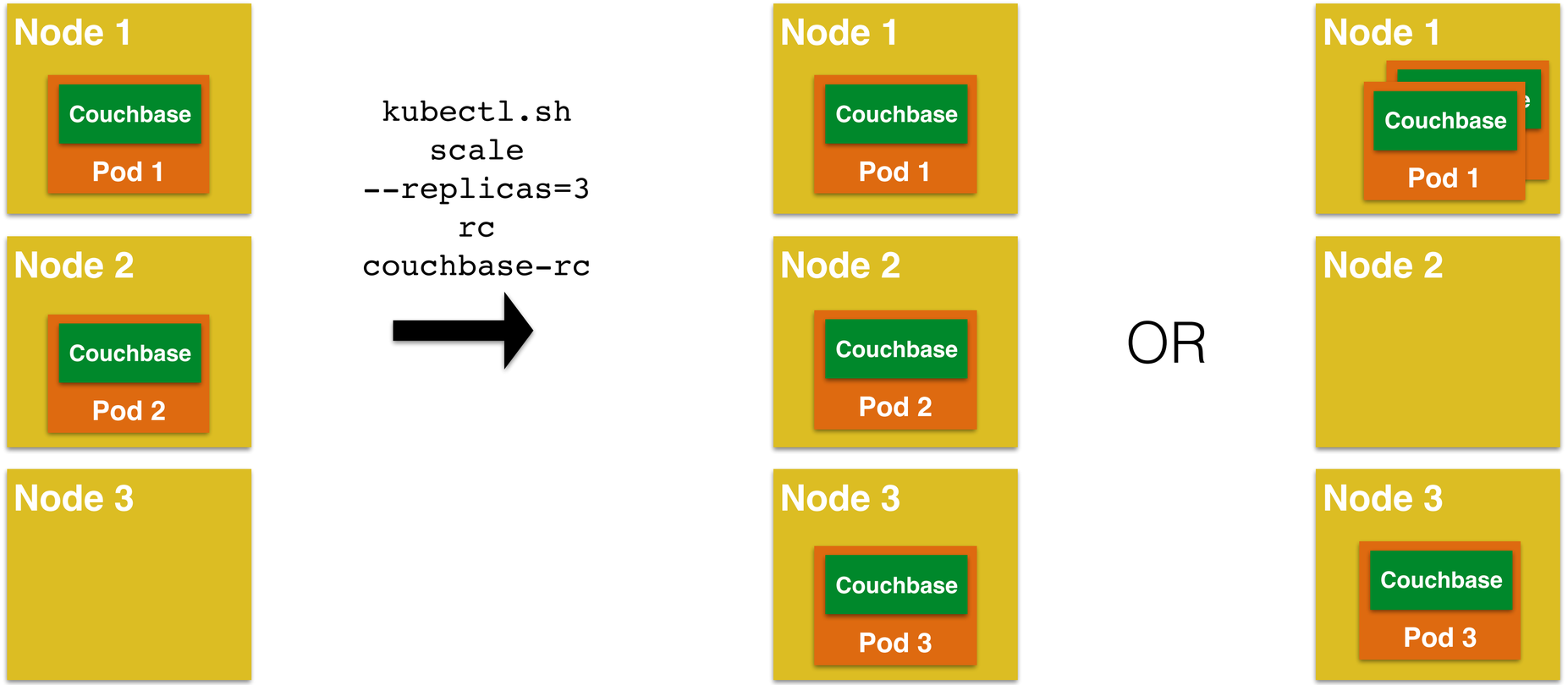
Replication Controller



Replication Controller: Automatic Rescheduling



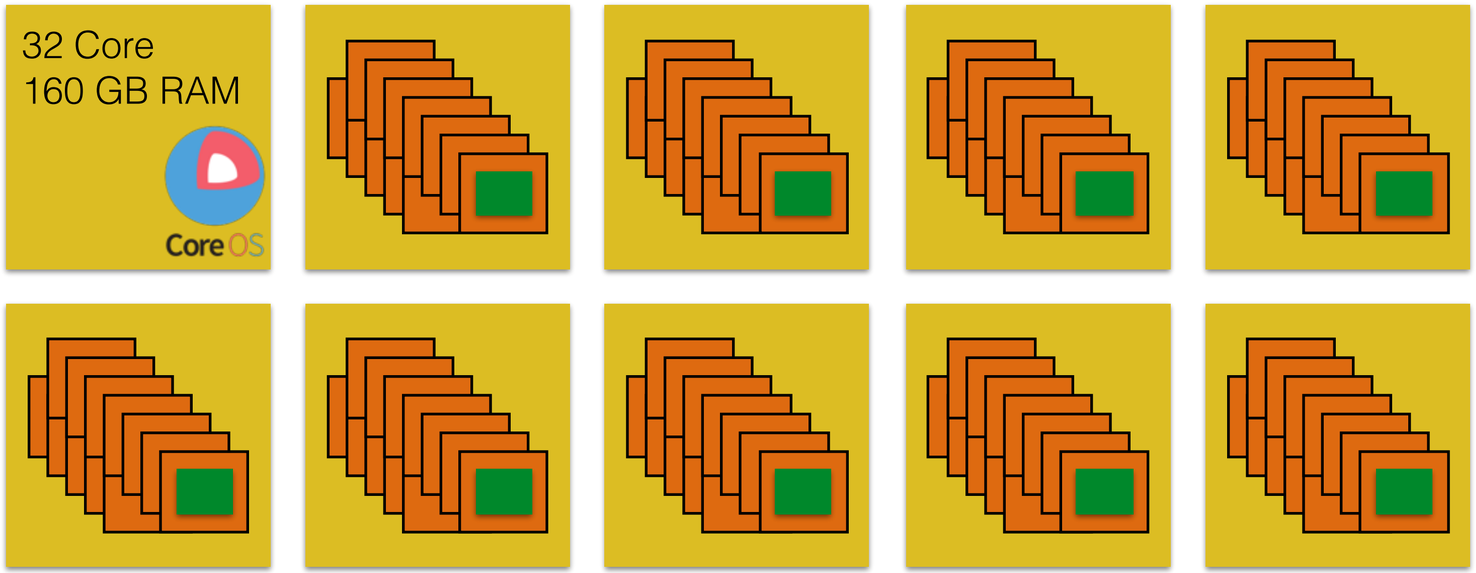
Replication Controller: Scaling



Sample Production Deployment

- www.wombatsoftware.de
- shopadvisors.de: E-commerce optimization and monitoring tools for increase of sales

Sample Production Deployment



Load	Containers
Normal	400
Peak	600

Health Checks

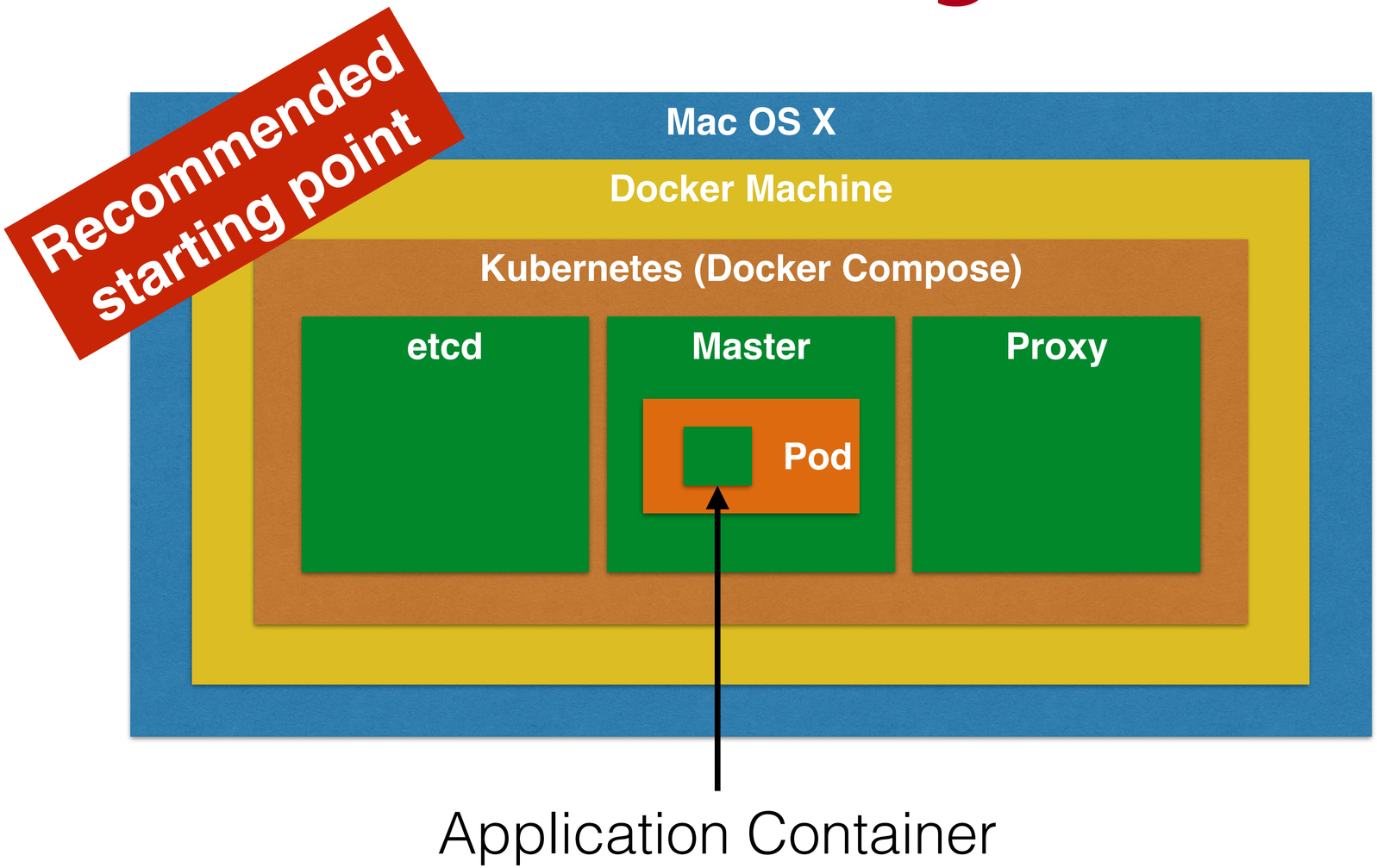
- Restarts Pod, if wrapped in RC
- Application-level health checks
  - HTTP
  - Container Exec
  - TCP Socket
- Health checks performed by Kubelet

Kubernetes using Docker

```
etcd:
  image: gcr.io/google_containers/etcd:2.0.9
  net: "host"
  entrypoint: /usr/local/bin/etcd --addr=127.0.0.1:4001 --bind-addr=0.0.0.0:4001 --data-dir=/
master:
  image: gcr.io/google_containers/hyperkube:v0.21.2
  net: "host"
  volumes:
    - /var/run/docker.sock:/var/run/docker.sock
  entrypoint: /hyperkube kubelet --api_servers=http://localhost:8080 --v=2 --address=0.0.0.0
proxy:
  image: gcr.io/google_containers/hyperkube:v0.21.2
  net: "host"
  privileged: true
  entrypoint: /hyperkube proxy --master=http://127.0.0.1:8080 --v=2
```

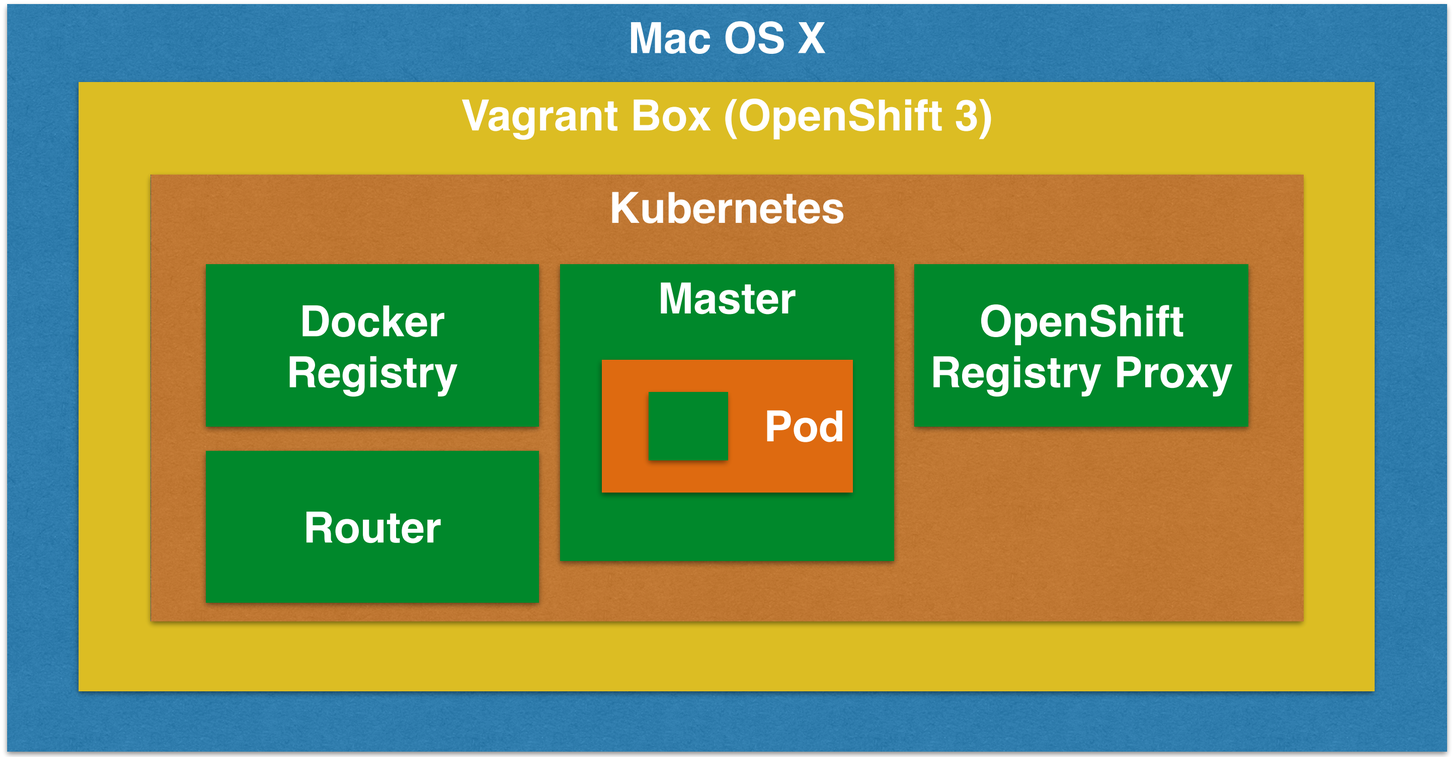

https://github.com/arun-gupta/docker-images/blob/master/kubernetes/docker-compose.yml

Kubernetes using Docker



http://kubernetes.io/v1.0/docs/getting-started-guides/docker.html

OpenShift 3



References

- github.com/javaee-samples/docker-java
- kubernetes.io/v1.1/docs