Kubernetes Introduction

Arun Gupta, @arungupta

Kubernetes

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

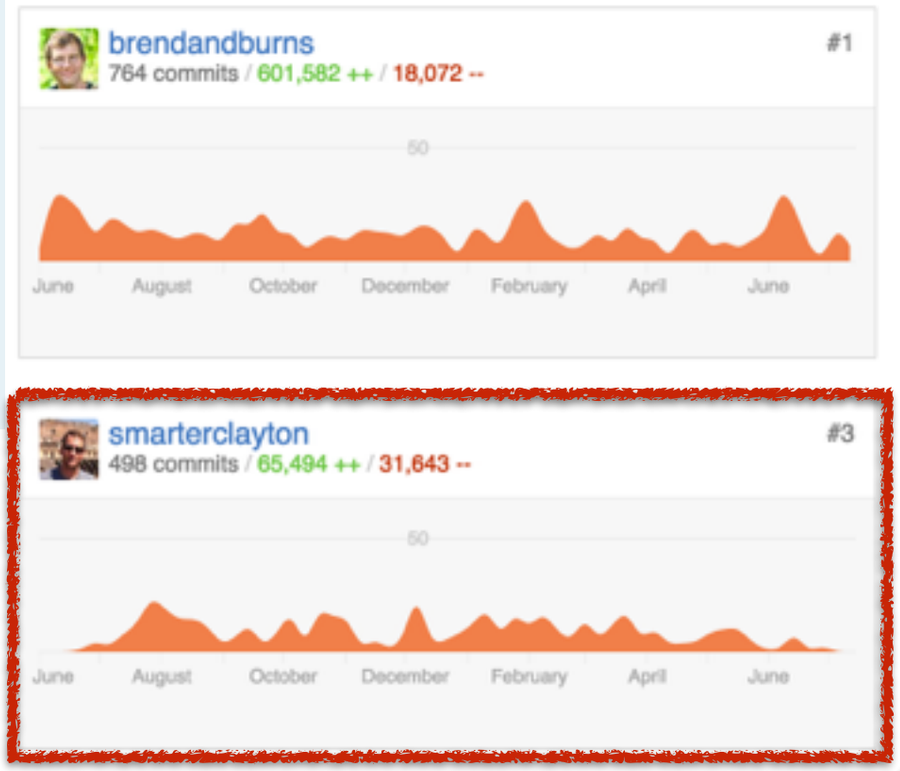
Collect statistics about who wrote kubernetes

gitdm (git data mine) is a tool written by Jonathan Corbet at LWN which he uses to do his 'who wrote the kernel' articles. I spent a couple of minutes to run it against kube. Some intersting results...

```
Developers with the most changesets
Brendan Burns                 643 (10.3%)
Daniel Smith                  485 (7.8%)
Clayton Coleman               453 (7.3%)
Tim Hockin                    312 (5.0%)
derekwaynecarr                209 (3.4%)

Developers with the most changed lines
Brendan Burns                455888 (18.0%)
Nan Monnand Deng             379610 (15.0%)
Yifan Gu                     219191 (8.7%)
Patrick                      169265 (6.7%)
nikhiljindal                 124915 (4.9%)

Top changeset contributors by employer
"Google"                     3562 (58.3%)
"Red Hat"                    1252 (20.5%)
(Unknown)                     899 (14.7%)
"CoreOS"                      144 (2.4%)
"FathomDB"                    142 (2.3%)
```



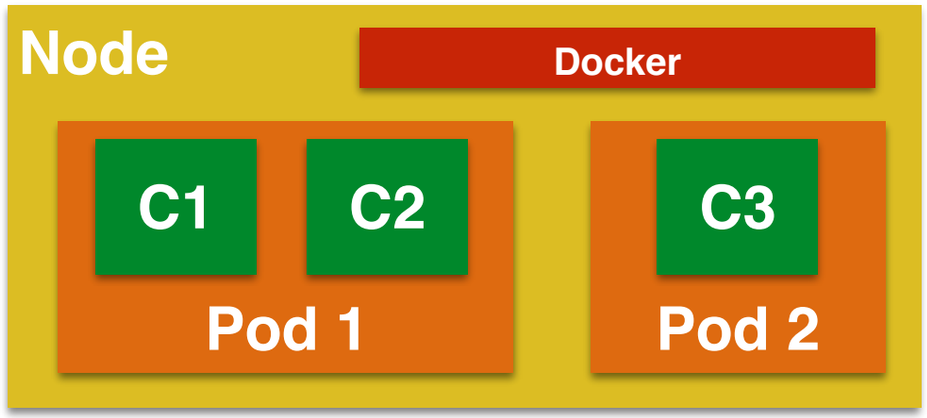
Concepts

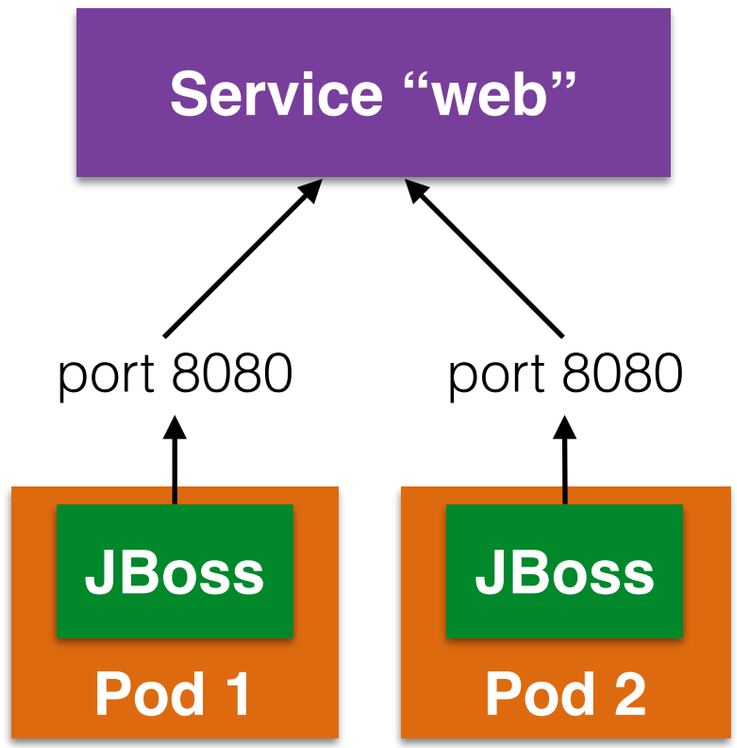
- **Pods**: collocated group of Docker containers that share an IP and storage volume

- **Service**: Single, stable name for a set of pods, also acts as LB

- **Replication Controller**: manages the lifecycle of pods and ensures specified number are running

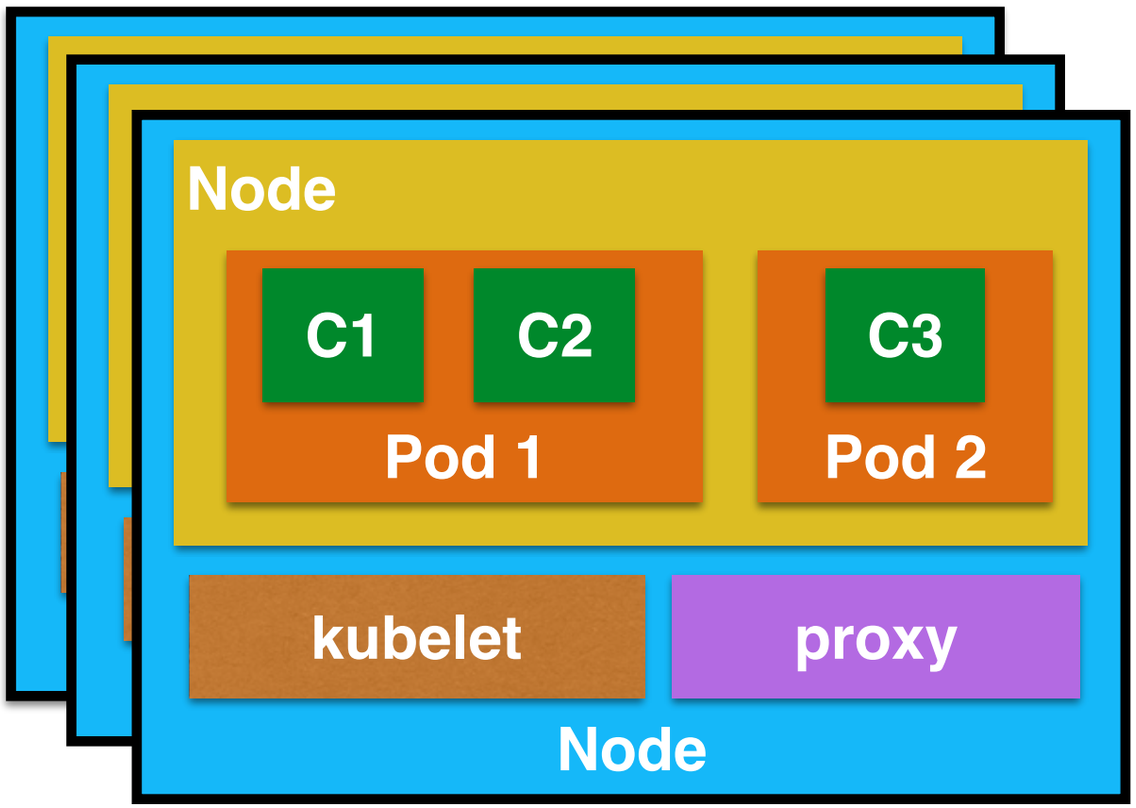
- **Label**: used to organize and select group of objects

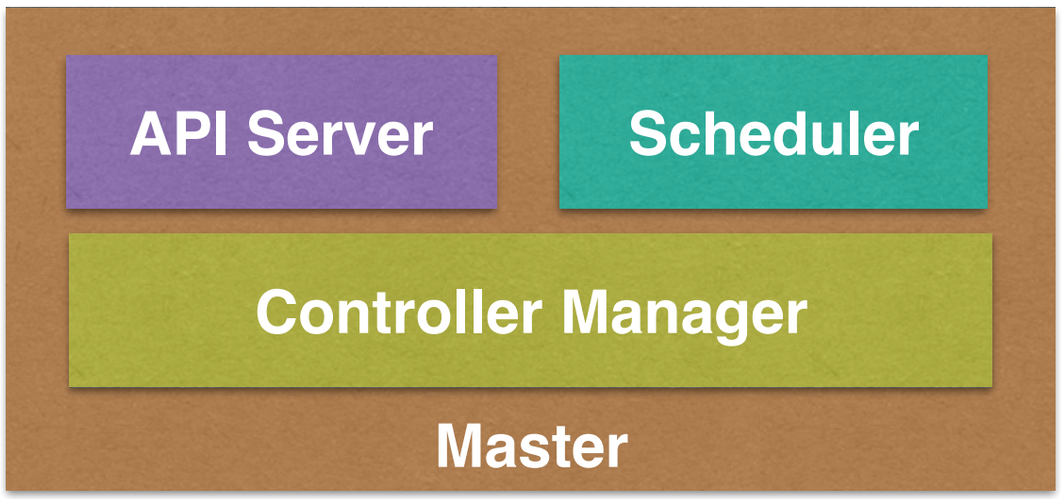




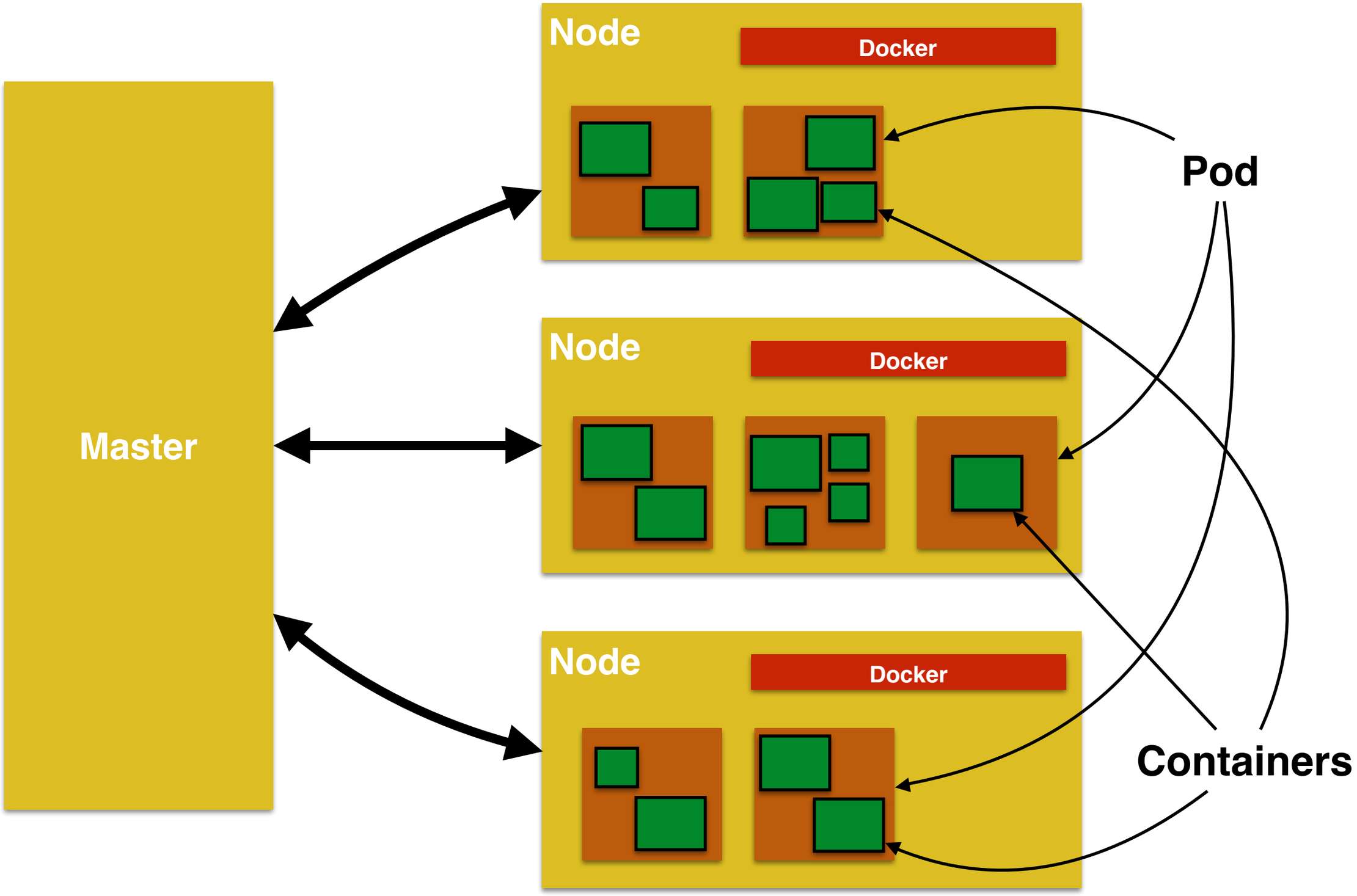
Components

- **Cluster**: compute resources on top of which container are built
- **Node**: Docker host running *kubelet* (node agent) and *proxy* services
- **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



kubectl

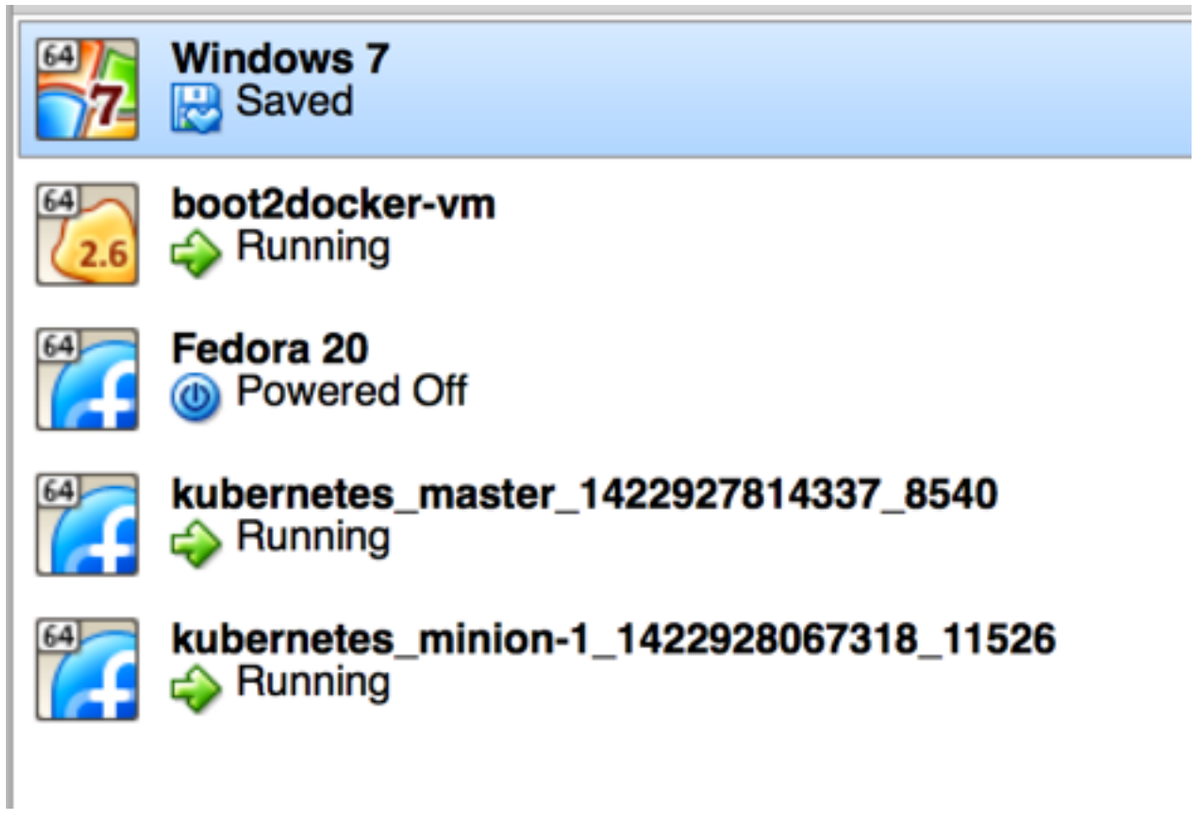
- Controls the Kubernetes cluster manager
- `kubectl get pods` or `minions`
- `kubectl create -f <filename>`
- `kubectl update` or `delete`
- `kubectl resize –replicas=3 replicationcontrollers <name>`

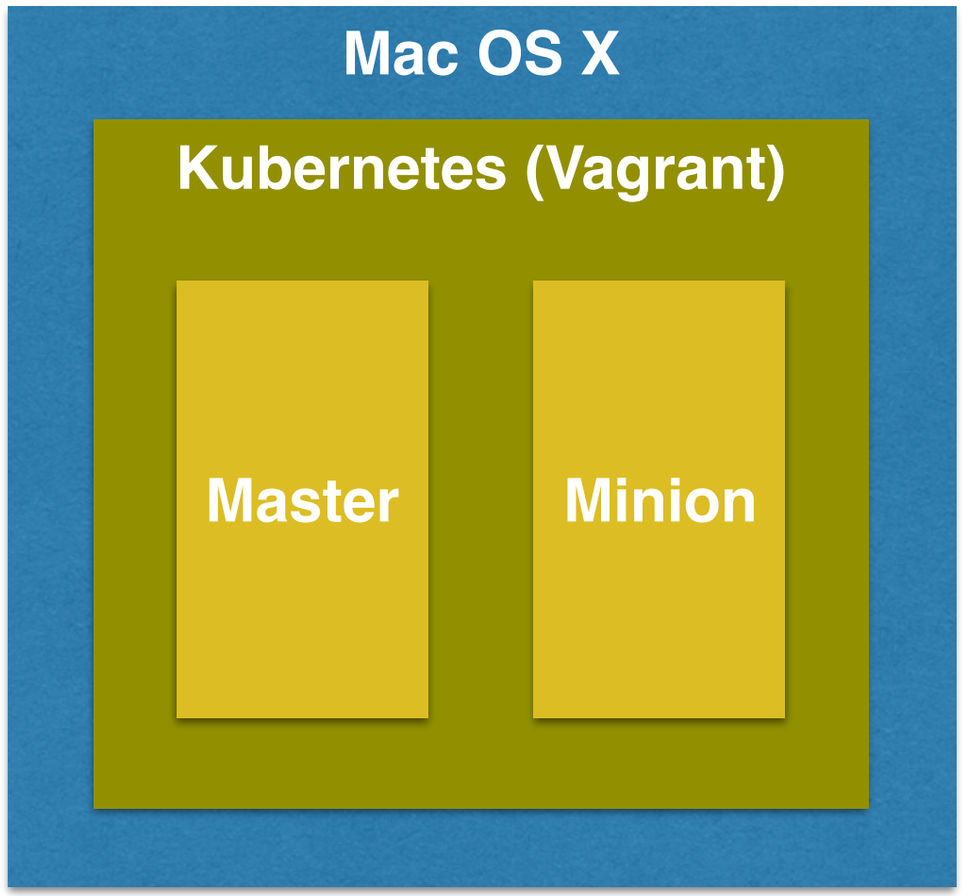
Kubernetes Config

```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
```

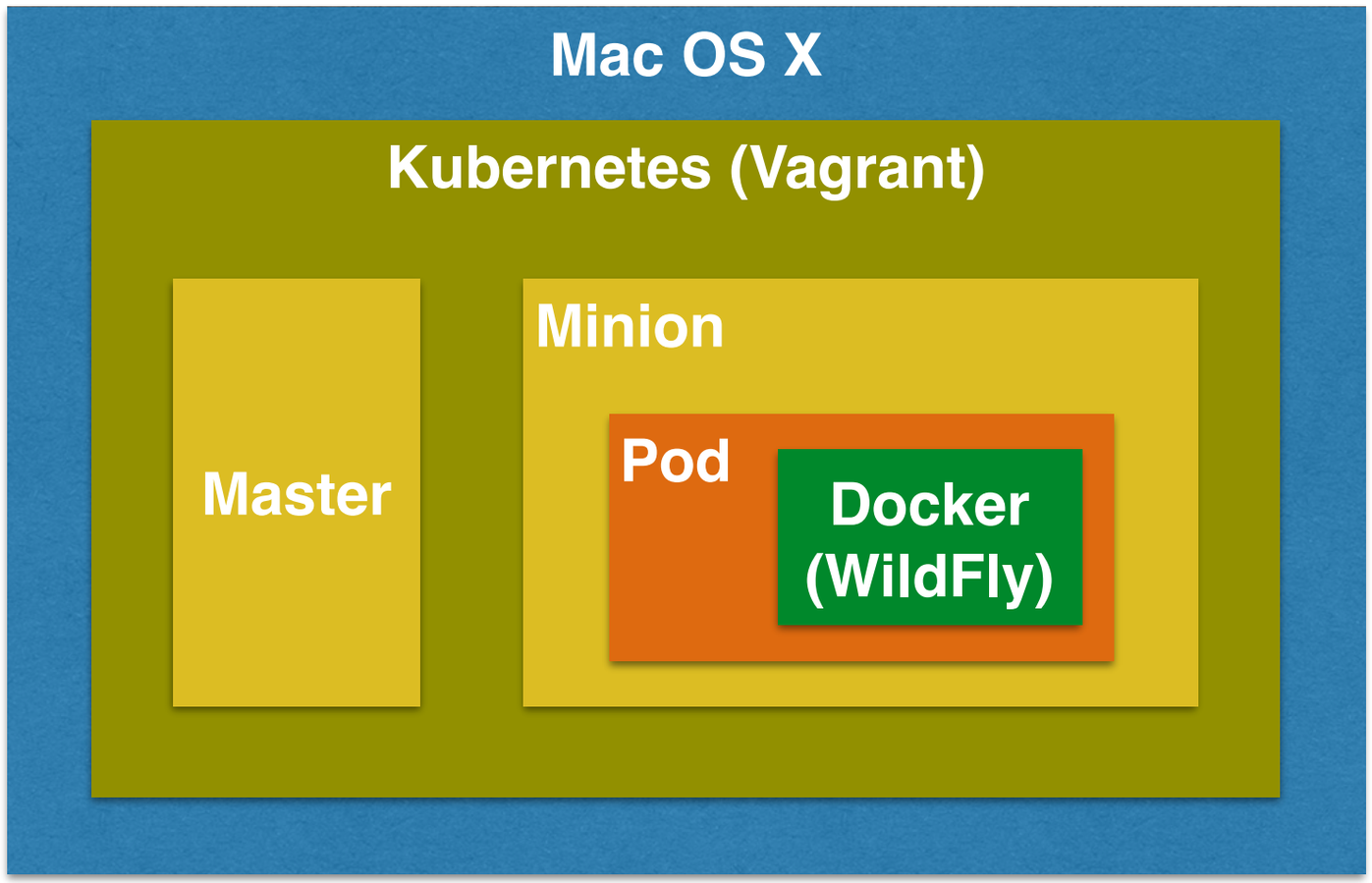
```
export KUBERNETES_PROVIDER=vagrant
./cluster/kube-up.sh
```





A Pod with One Container

```
{
  "id": "wildfly",
  "kind": "Pod",
  "apiVersion": "v1beta1",
  "desiredState": {
    "manifest": {
      "version": "v1beta1",
      "id": "wildfly",
      "containers": [{
        "name": "wildfly",
        "image": "arungupta/javaee7-hol",
        "cpu": 100,
        "ports": [{
          "containerPort": 8080,
          "hostPort": 8080
        },
        {
          "containerPort": 9090,
          "hostPort": 9090
        }]
      }]
    }
  },
  "labels": {
    "name": "wildfly"
  }
}
```



http://blog.arungupta.me/javaee7-wildfly-kubernetes-mac-vagrant/

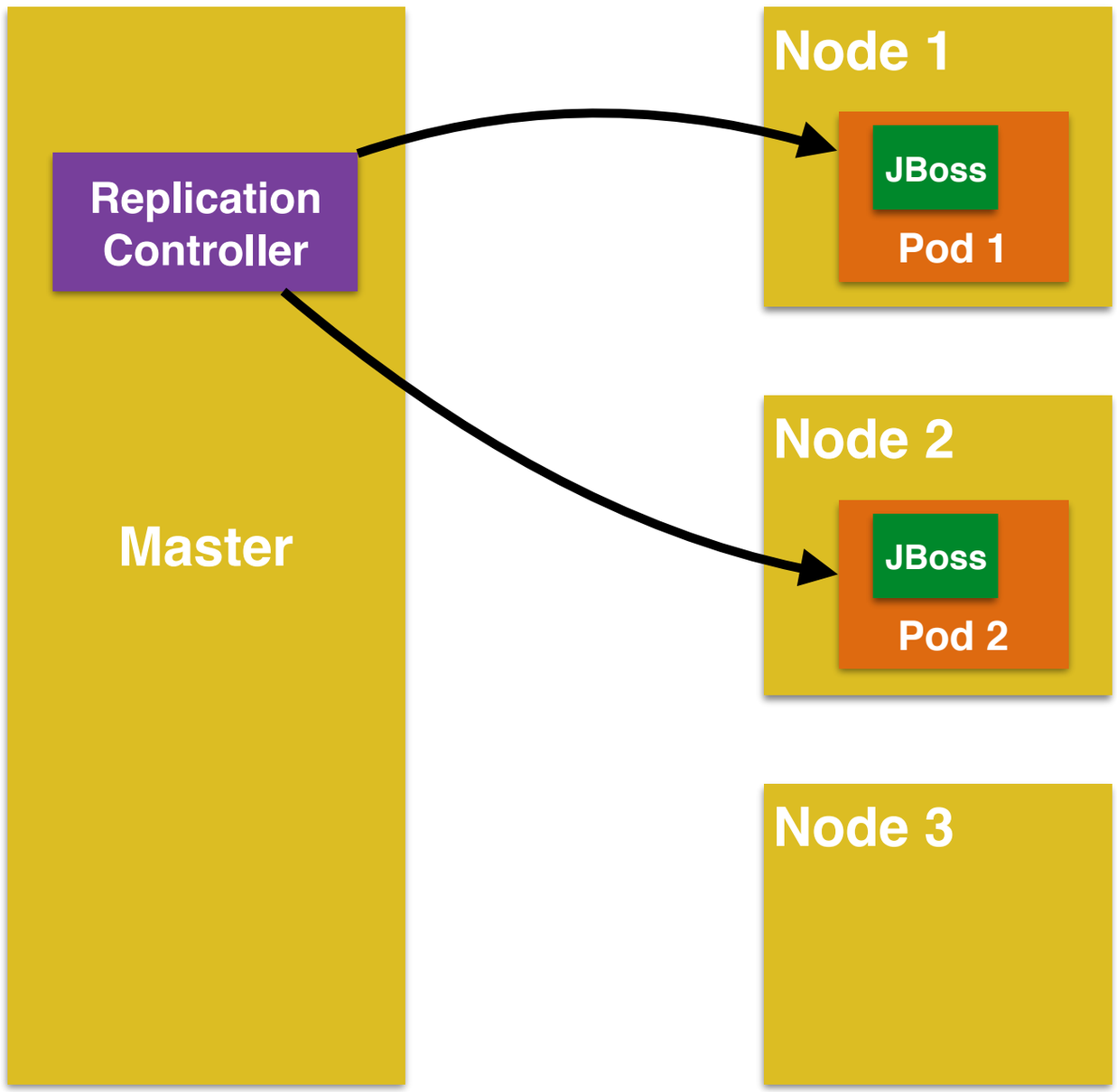
Replication Controller

- Ensures that a specified number of pod "replicas" are running at any one time

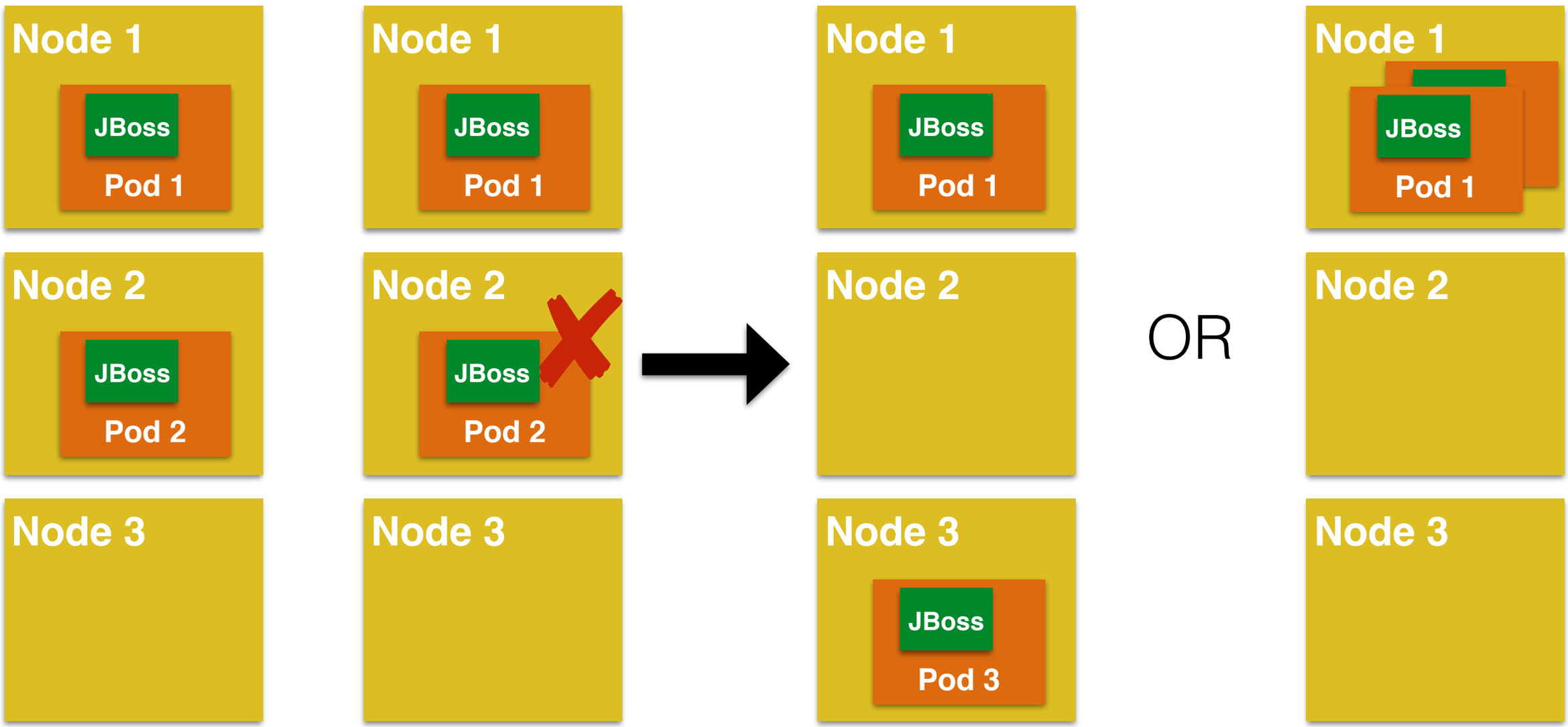
- Recommended to wrap a Pod in a RC

- Only appropriate for Pods with `Restart=Always` policy (default)

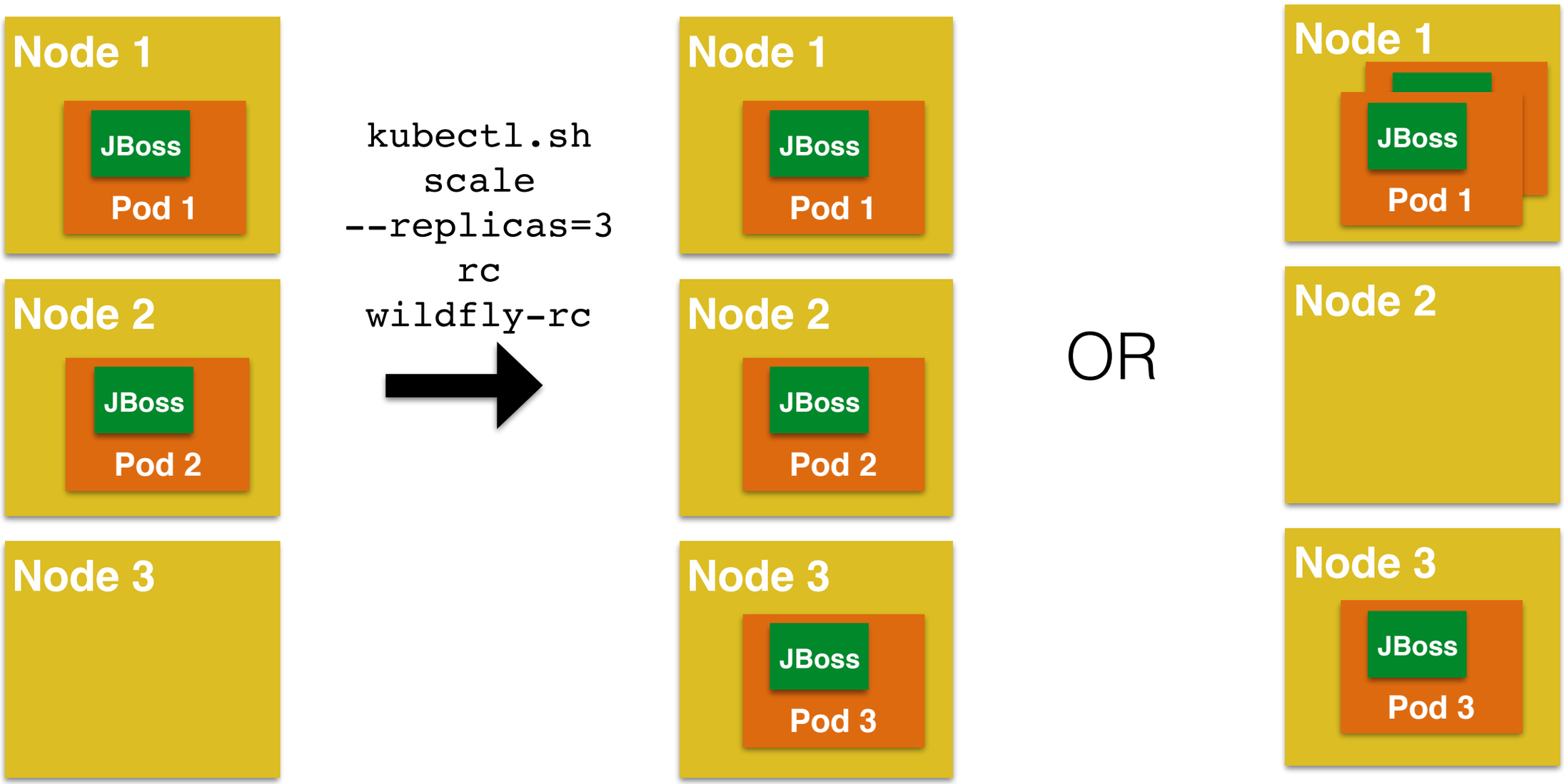
Replication Controller



Replication Controller: Automatic Rescheduling



Replication Controller: Scaling

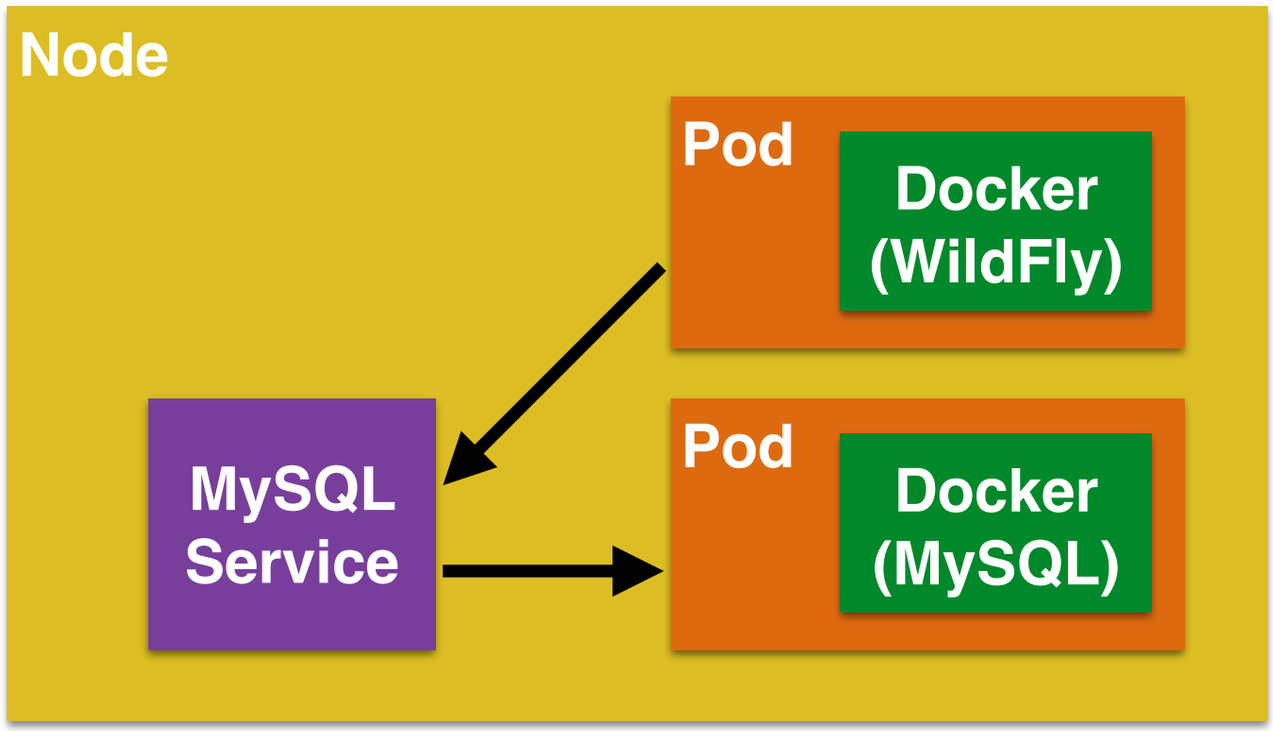


Services

- 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

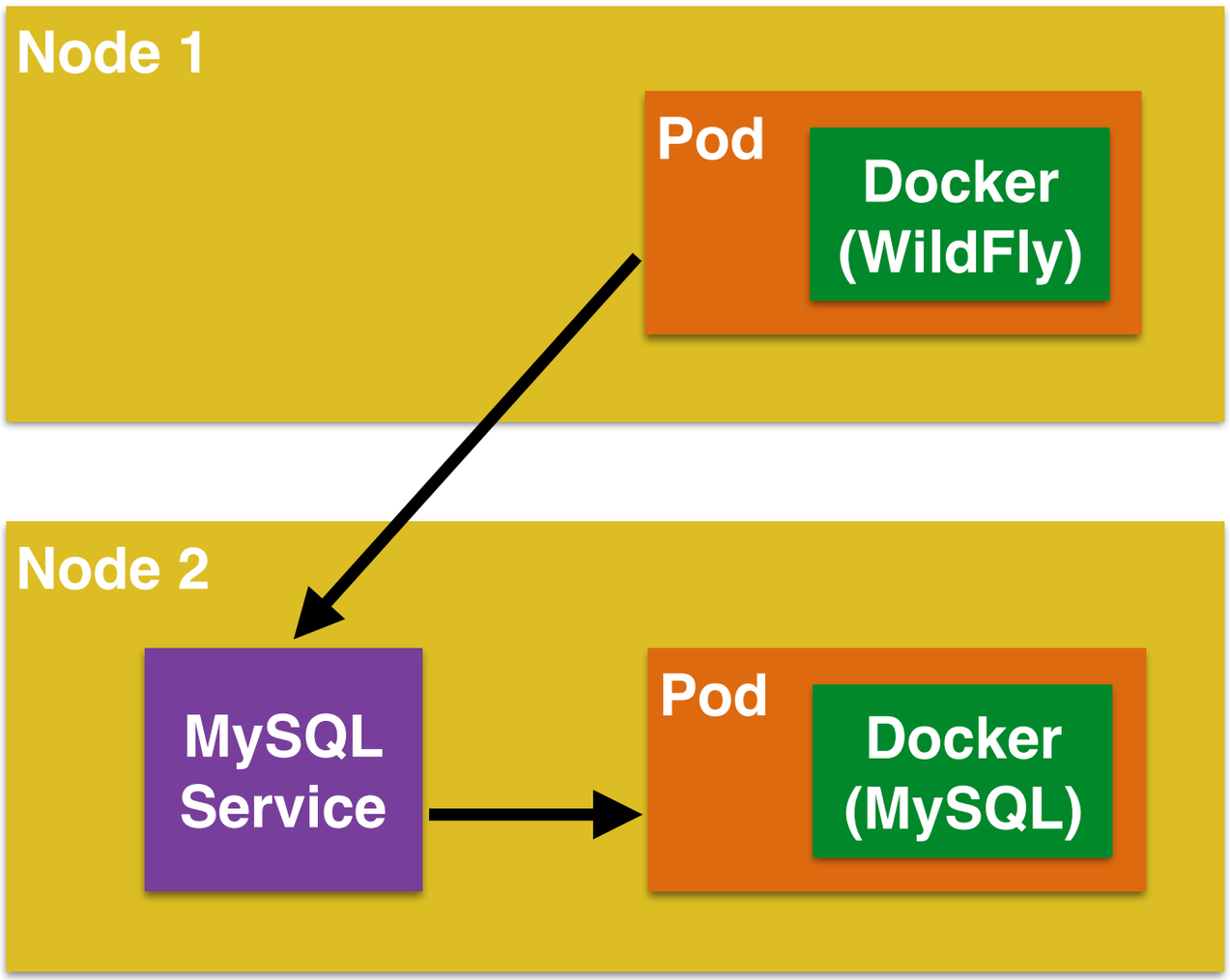
```
{
  "id": "mysql",
  "kind": "Pod",
  "apiVersion": "v1beta1",
  "desiredState": {
    "manifest": {
      "version": "v1beta1",
      "id": "mysql",
      "containers": [{
        "name": "mysql",
        "image": "mysql:latest",
        "cpu": 100,
        "env": [
              {
                  "name": "MYSQL_USER",
                  "value": "mysql"
              },
              {
                  "name": "MYSQL_PASSWORD",
                  "value": "mysql"
              },
              {
                  "name": "MYSQL_DATABASE",
                  "value": "sample"
              },
              {
                  "name": "MYSQL_ROOT_PASSWORD",
                  "value": "supersecret"
              }
          ],
        "ports": [{
          "containerPort": 3306,
          "hostPort": 3306
        }]
      }]
    }
  },
  "labels": {
    "name": "mysql"
  }
}
```




```
{
  "id": "mysql",
  "kind": "Service",
  "apiVersion": "v1beta1",
  "port": 3306,
  "containerPort": 3306,
  "selector": {
    "name": "mysql"
  },
  "labels": {
    "name": "mysql"
  }
}
```

http://blog.arungupta.me/mysql-kubernetes-service-access-wildfly-pod-techtip72/

Two Nodes

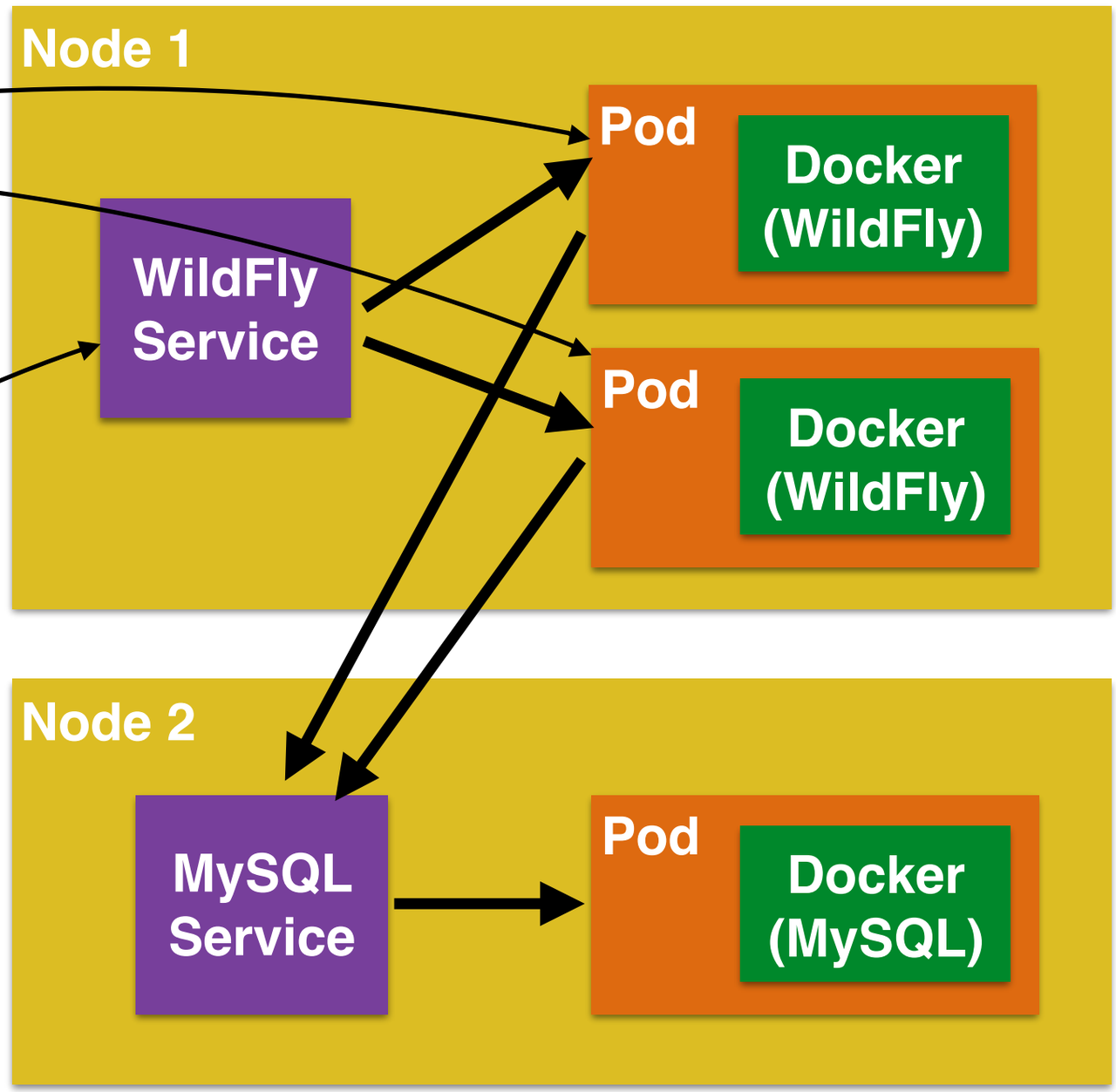

Node 1
Pod
Docker
(WildFly)
Node 2
MySQL
Service
Pod
Docker
(MySQL)

Replication Controller

- Ensures specified number of pod “replicas” are running
- Pod templates are cookie cutters
- Rescheduling
- Manual or auto-scale replicas
- Rolling updates

```yaml
id: wildfly-controller
kind: ReplicationController
apiVersion: v1beta1
desiredState:
  replicas: 2
  replicaSelector:
    name: wildfly
  podTemplate:
    desiredState:
      manifest:
        version: v1beta1
        id: wildfly
        containers:
          - name: javaee7-hol
            image: arungupta/javaee7-hol
    labels:
      name: wildfly
```

```yaml
id: wildfly
kind: Service
apiVersion: v1beta1
port: 8080
containerPort: 8080
selector:
  name: wildfly
labels:
  name: wildfly
```



Kubernetes: Pros and Cons

- PROS
  - Manage related Docker containers as a unit
  - Container communication across hosts
  - Availability and scalability through automated deployment and monitoring of pods and their replicas, across hosts

Kubernetes: Pros and Cons

- CONS
  - Lifecycle of applications - build, deploy, manage, promote
  - Port existing source code to run in Kubernetes
  - DevOps: Dev -> Test -> Production
  - No multi-tenancy
  - On-premise (available on GCE)
    - Assumes inter-pod networking as part of infrastructure
    - Requires explicit load balancer