

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 interesting results...

...

Developers with the most changesets

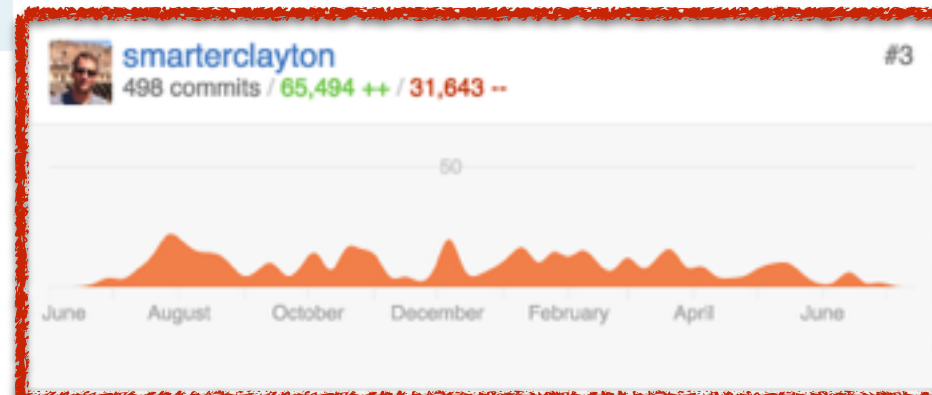
Brendan Burns	643 (10.3%)
Daniel Smith	485 (7.8%)
Clayton Coleman	453 (7.3%)
Tim Hockin	312 (5.0%)
derekwaynecarr	209 (3.4%)

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%)

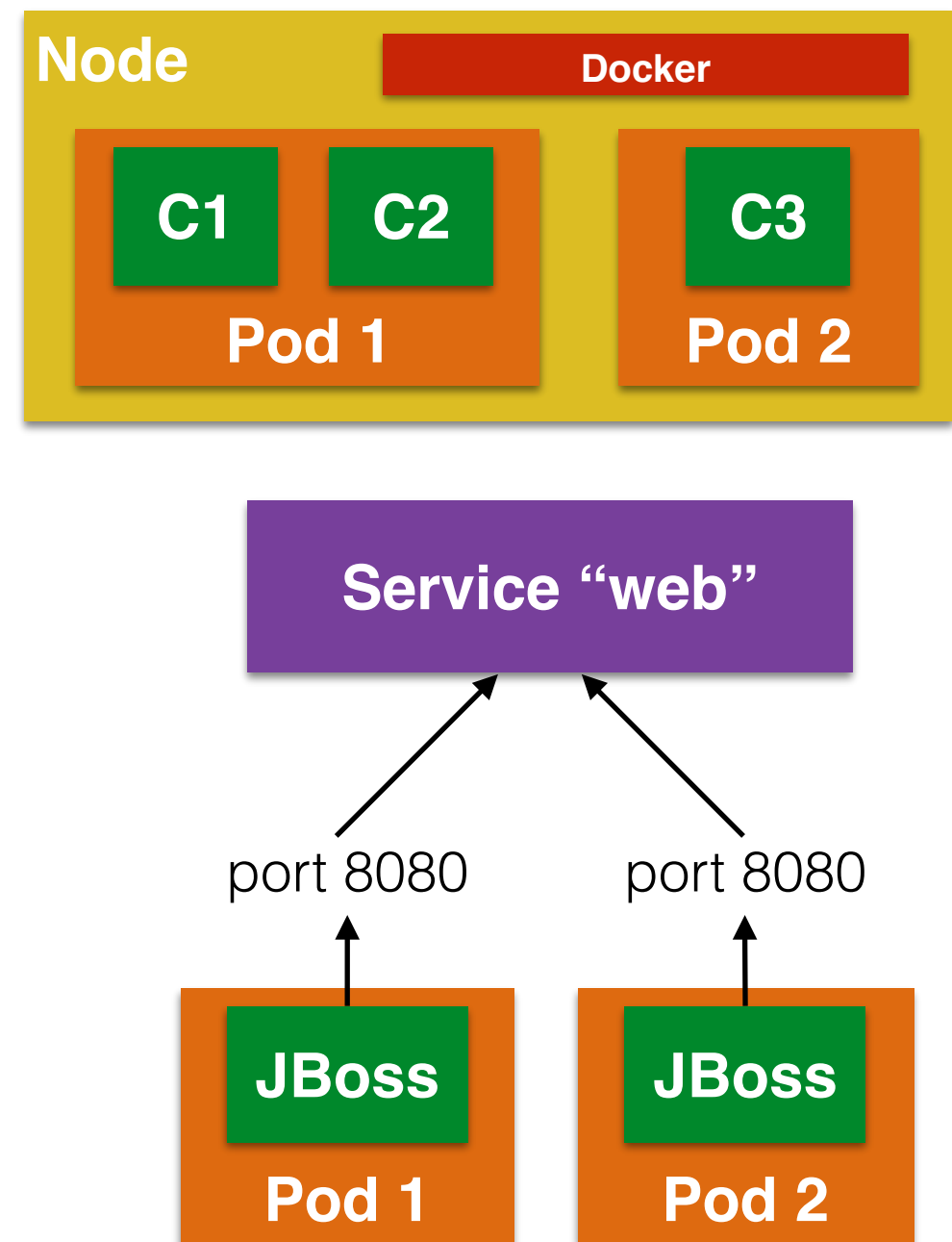
Top changeset contributors by employer

"Google"	3562 (58.3%)
"Red Hat"	1252 (20.5%)
(Unknown)	899 (14.7%)
"CoreOS"	144 (2.4%)
"FathomDB"	142 (2.3%)



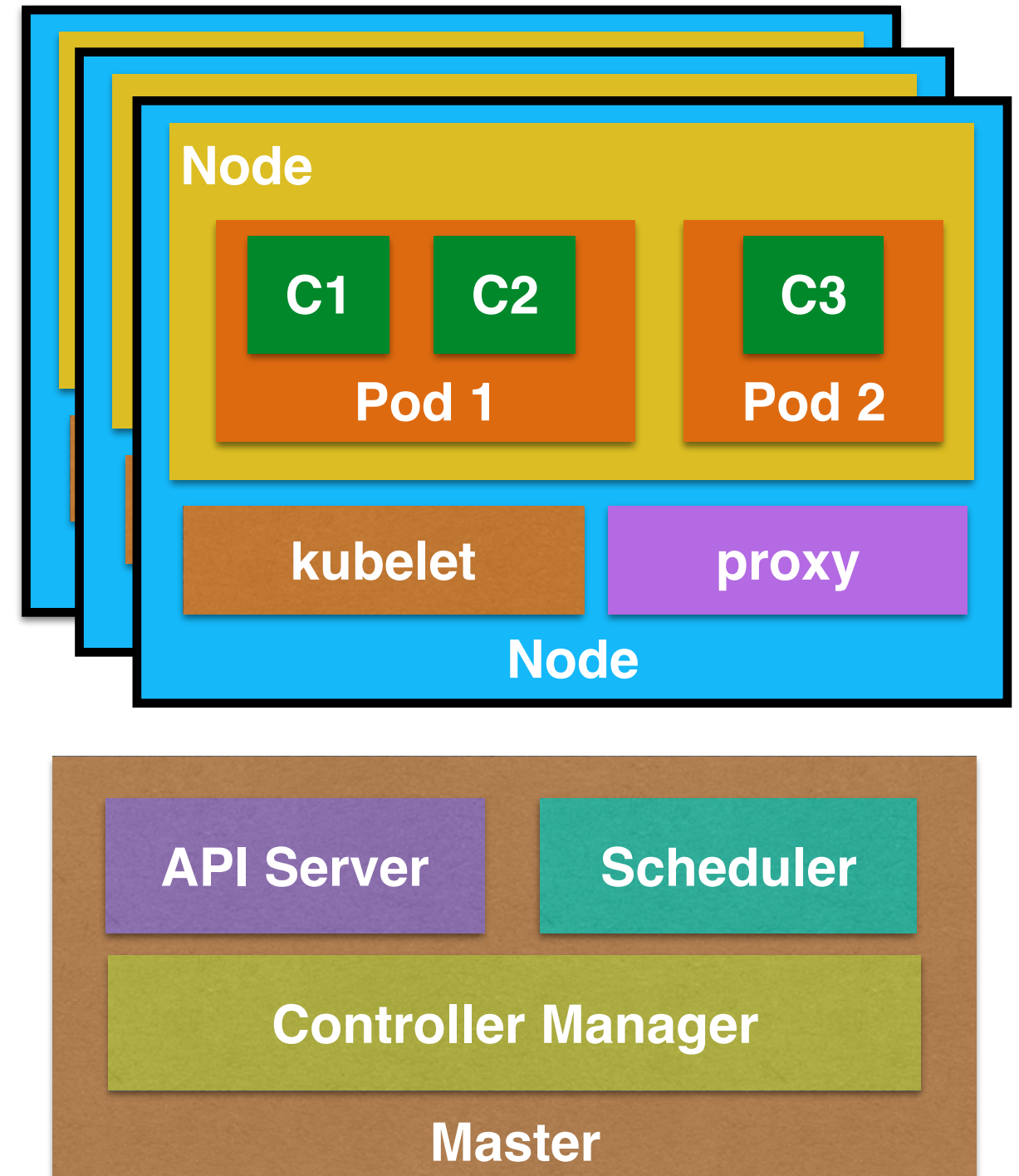
Concepts

- **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
- **Label:** used to organize and select group of objects
- **Replication Controller:** manages the lifecycle of pods and ensures specified number are running

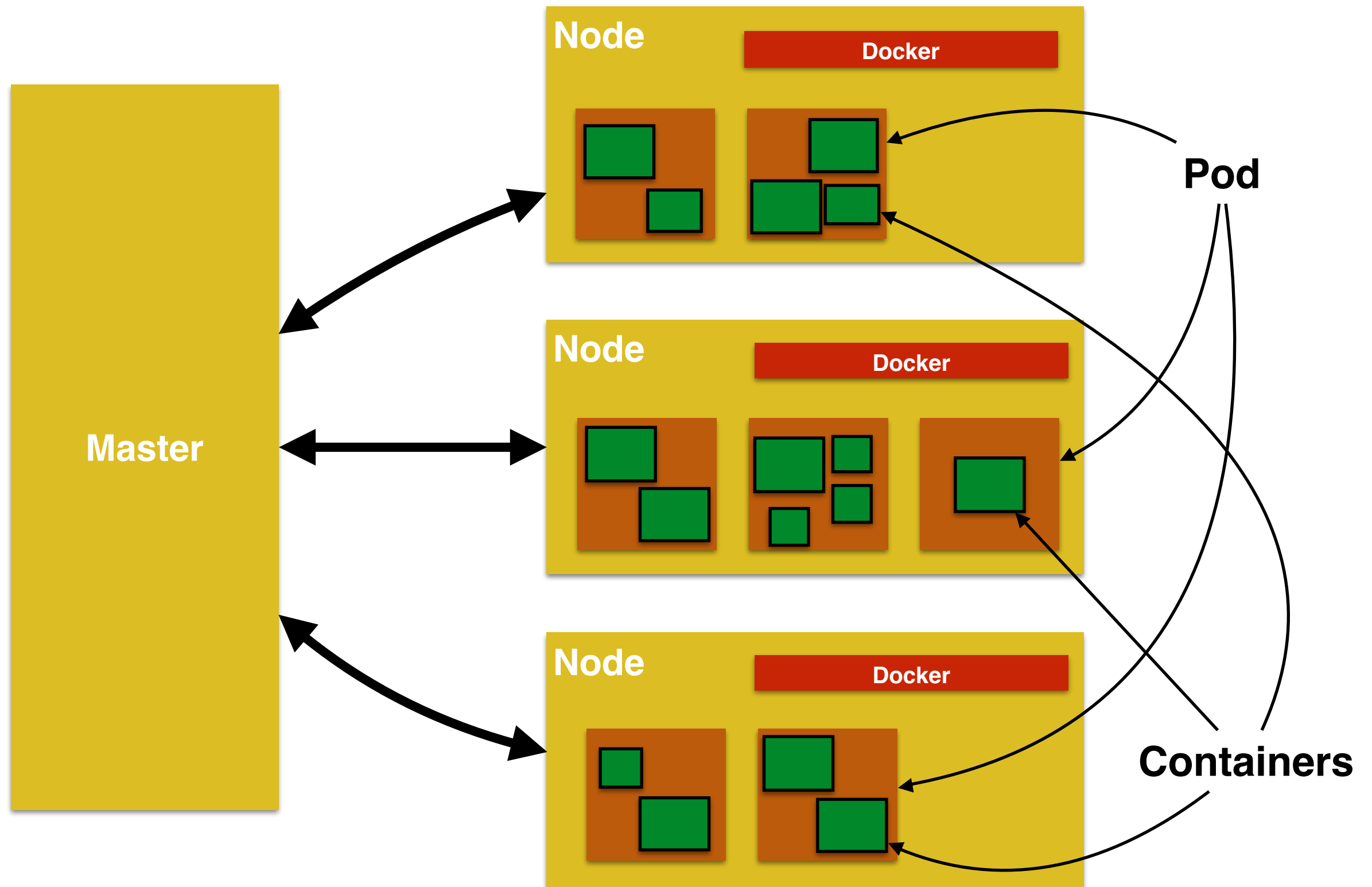


Components

- **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

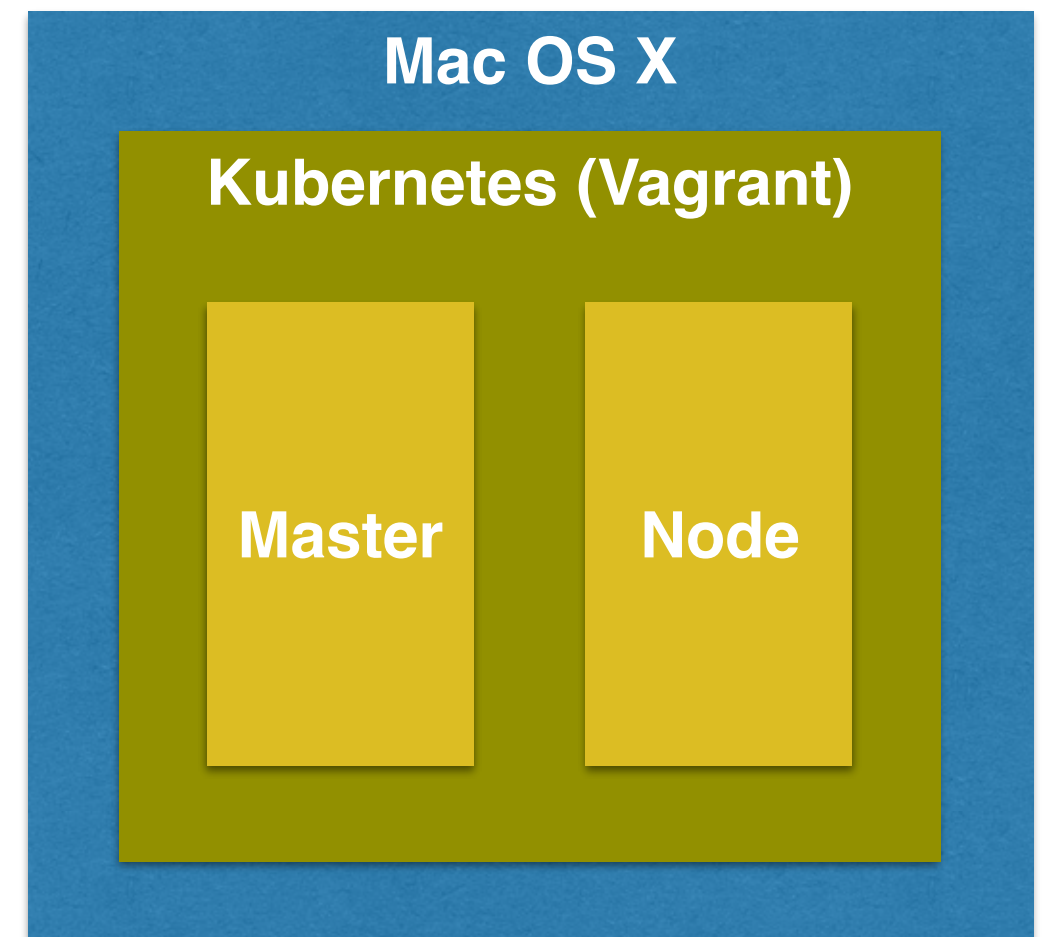
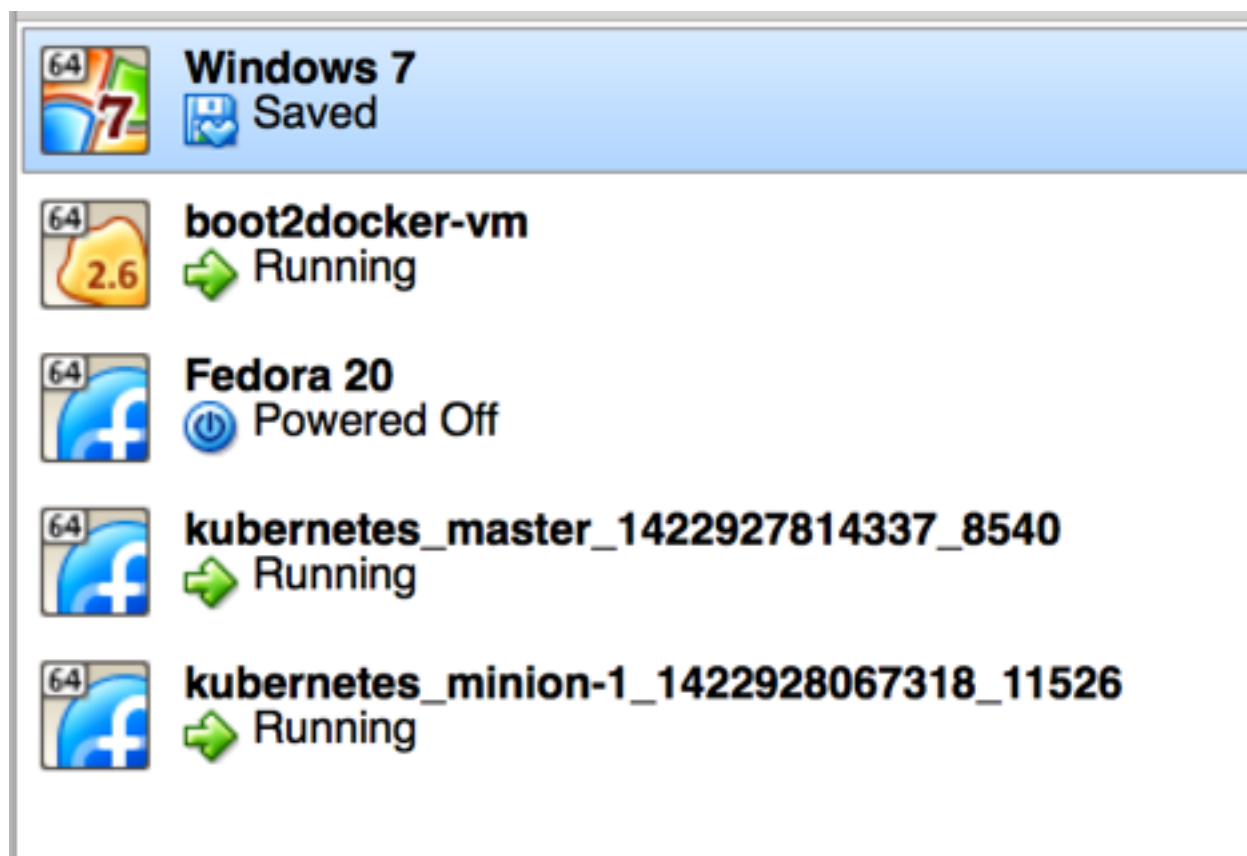
- Controls the Kubernetes cluster manager
- `kubectl get pods or minions`
- `kubectl create -f <filename>`
- `kubectl update or delete`
- `kubectl resize --replicas=3
replicationcontrollers <name>`

Kubernetes Config

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

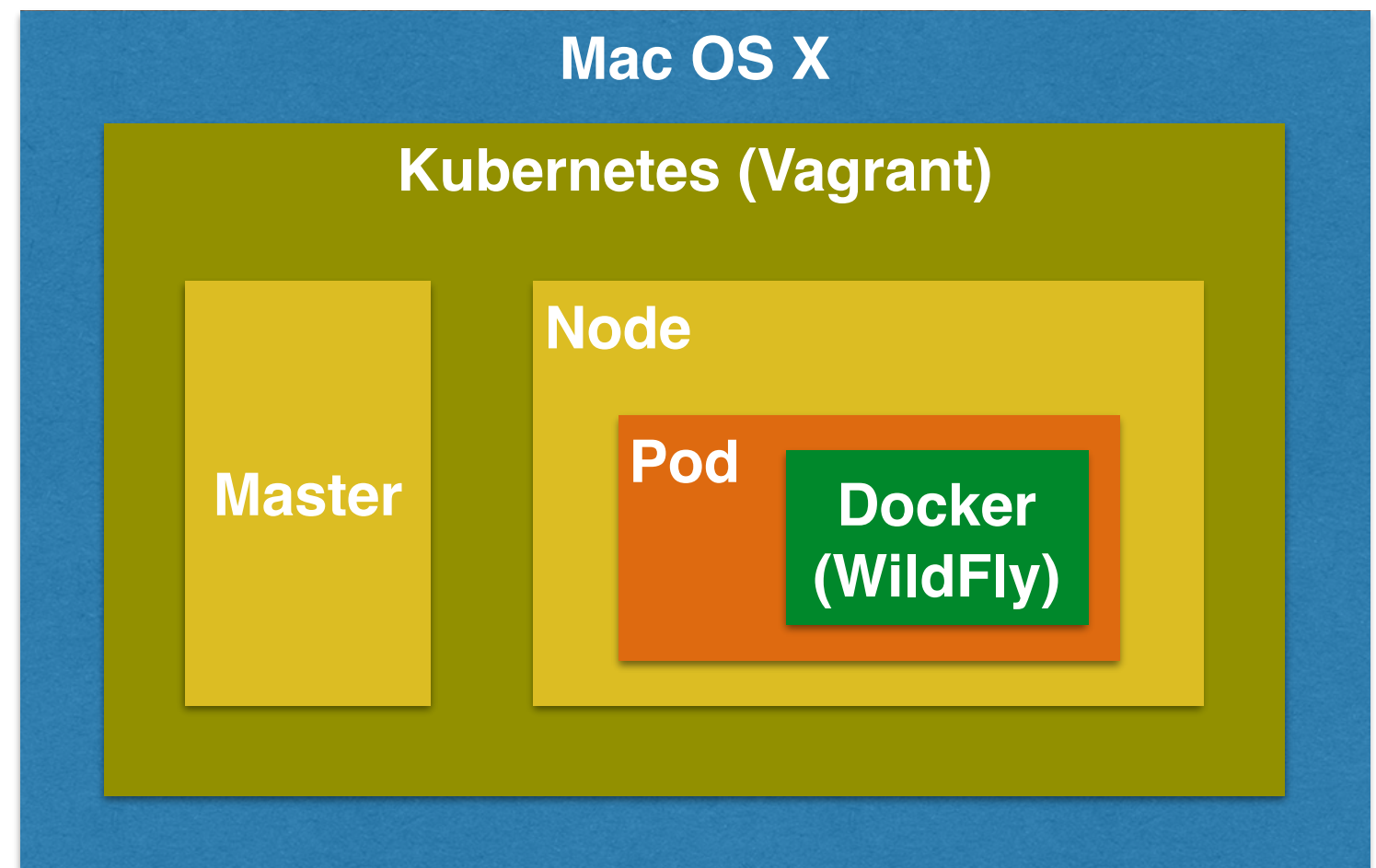
```
1  apiVersion: v1
2  kind: ReplicationController
3  metadata:
4    name: wildfly-rc
5    labels:
6      name: wildfly
7  spec:
8    replicas: 2
9    template:
10      metadata:
11        labels:
12          name: wildfly
13      spec:
14        containers:
15          - name: wildfly-rc-pod
16  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

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



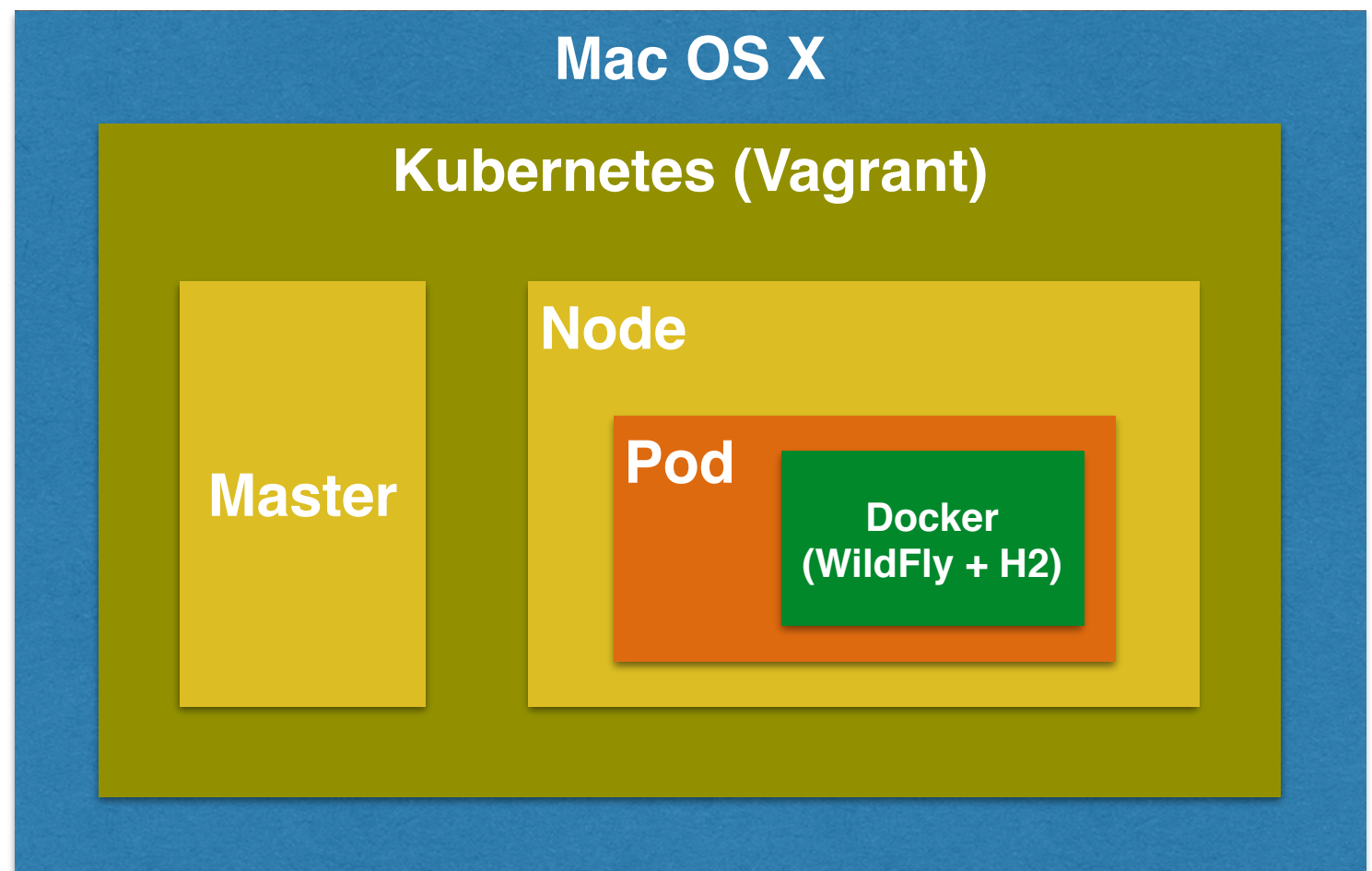
A Pod with One Container

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



Java EE Application Deployed in a Pod with One Container

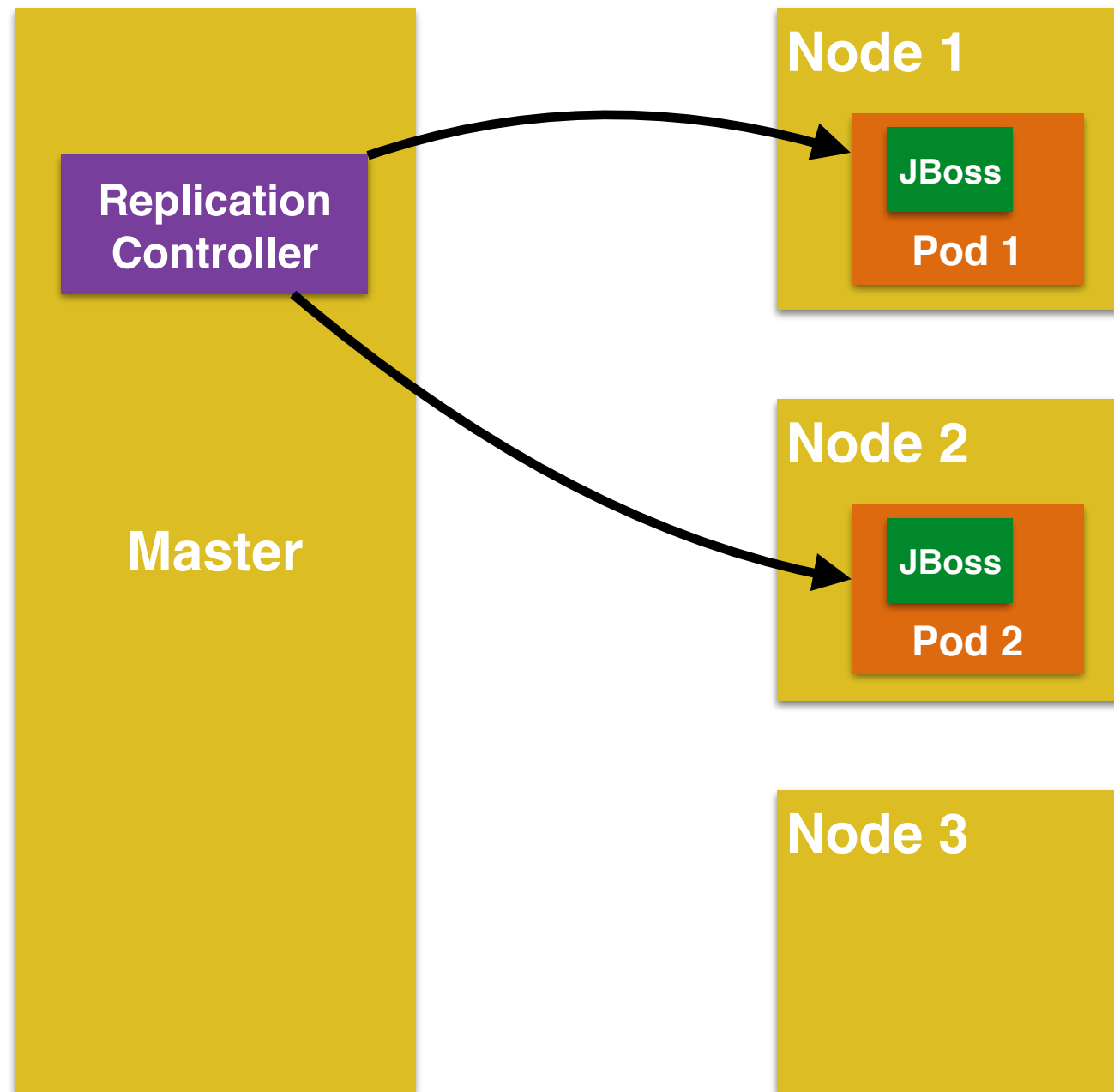
```
1  apiVersion: v1
2  kind: ReplicationController
3  metadata:
4    name: javaee7-hol
5    labels:
6      name: javaee7-hol
7  spec:
8    replicas: 1
9    selector:
10     name: javaee7-hol
11   template:
12     metadata:
13       labels:
14         name: javaee7-hol
15     spec:
16       containers:
17       - name: master
18         image: arungupta/javaee7-hol
19         ports:
20         - containerPort: 8080
21           hostPort: 8080
```



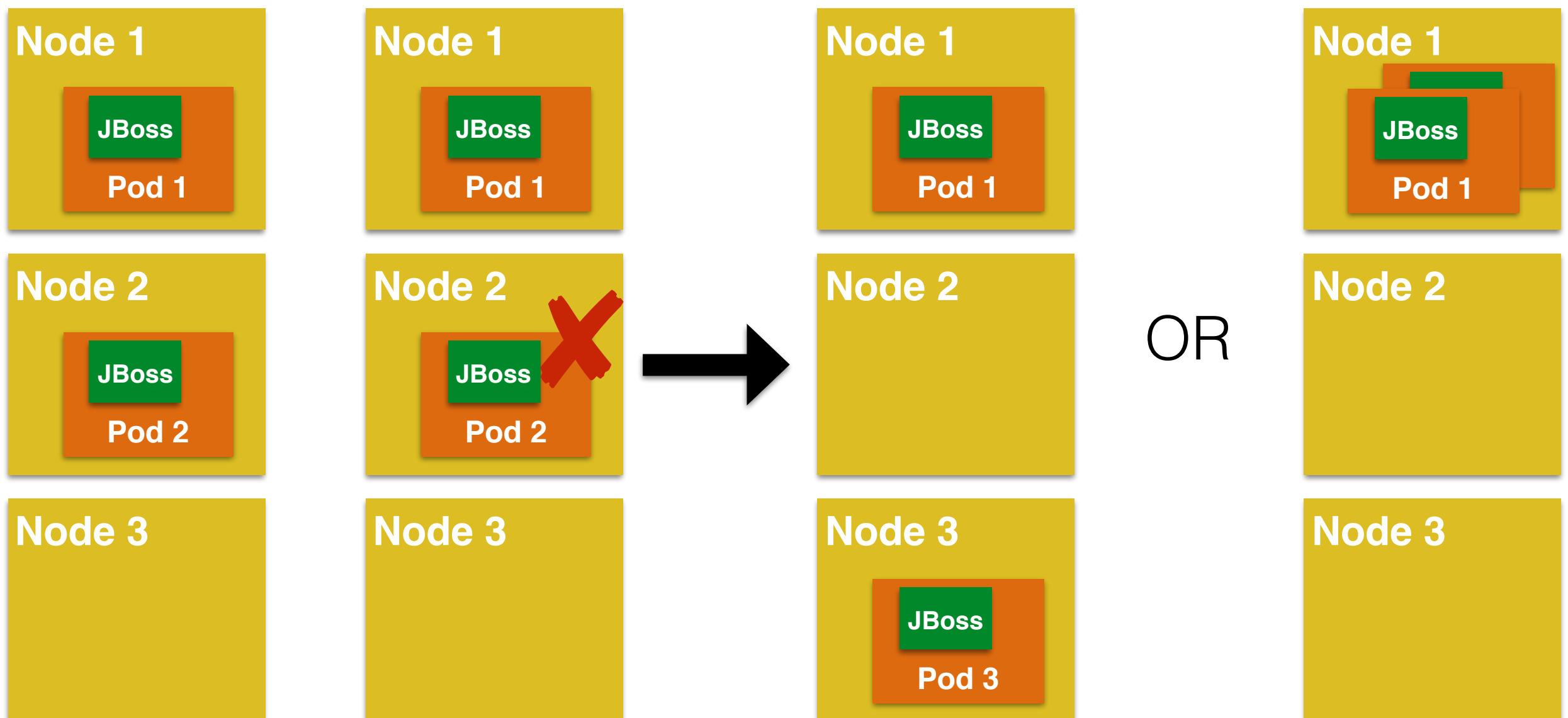
Replication Controller

- 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)

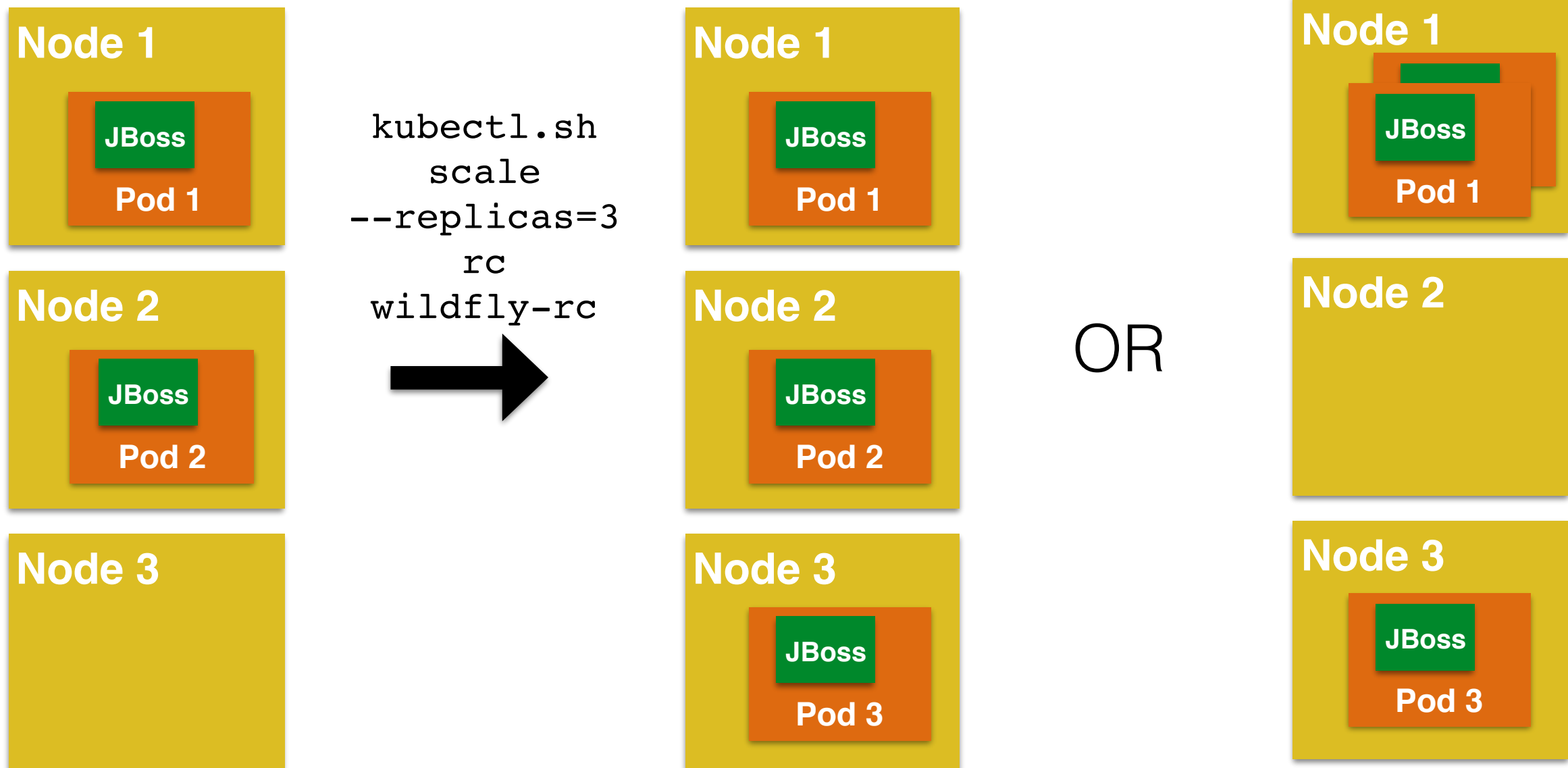
Replication Controller



Replication Controller: Automatic Rescheduling



Replication Controller: Scaling



Services

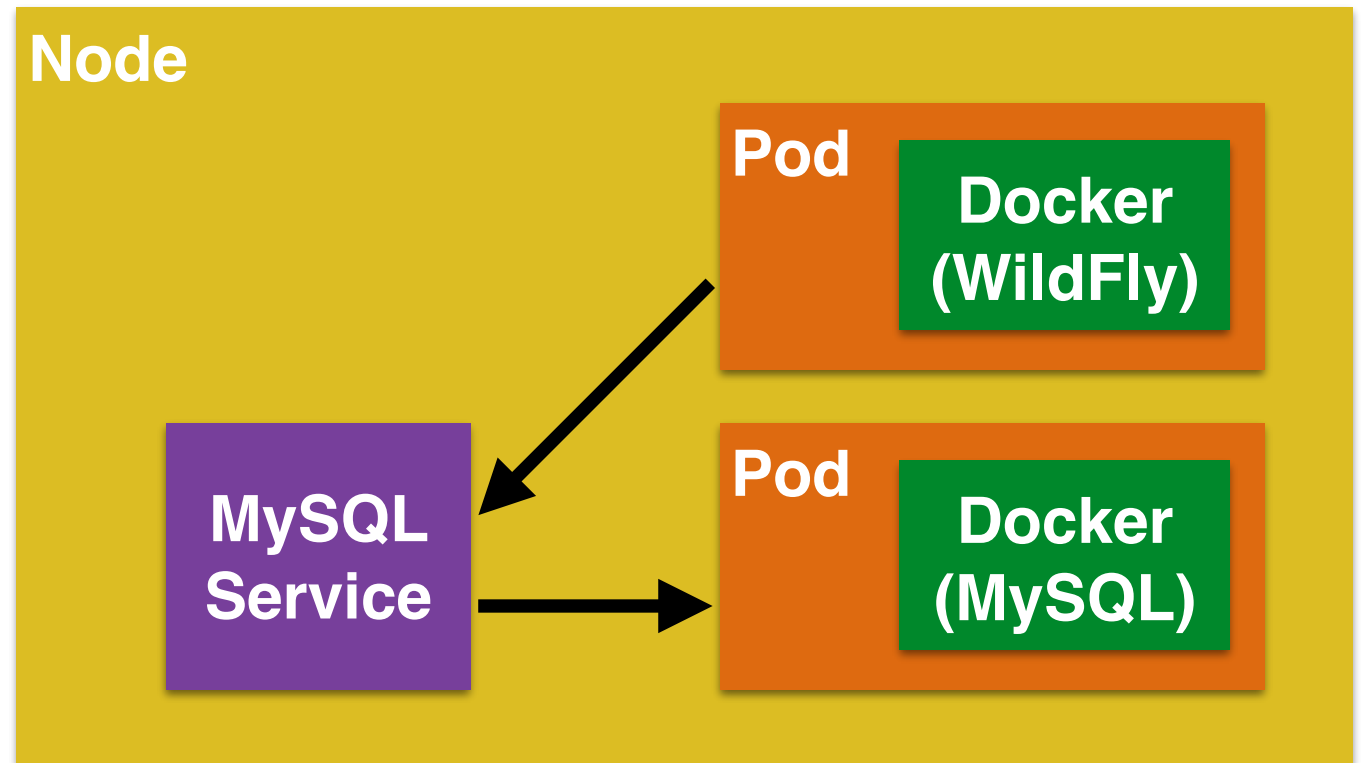
- Abstract a set of pods as a single IP and port
 - Simple TCP/UDP load balancing
- Creates environment variables in other pods
 - Like “Docker links” but across hosts
- Stable endpoint for pods to reference
 - Allows list of pods to change dynamically

Services

```

1 {
2   "id": "mysql",
3   "kind": "Pod",
4   "apiVersion": "v1beta1",
5   "desiredState": {
6     "manifest": {
7       "version": "v1beta1",
8       "id": "mysql",
9       "containers": [{
10        "name": "mysql",
11        "image": "mysql:latest",
12        "cpu": 100,
13        "env": [
14          {
15            "name": "MYSQL_USER",
16            "value": "mysql"
17          },
18          {
19            "name": "MYSQL_PASSWORD",
20            "value": "mysql"
21          },
22          {
23            "name": "MYSQL_DATABASE",
24            "value": "sample"
25          },
26          {
27            "name": "MYSQL_ROOT_PASSWORD",
28            "value": "supersecret"
29          }
30        ],
31        "ports": [{
32          "containerPort": 3306,
33          "hostPort": 3306
34        }]
35      }]
36    }
37  },
38  "labels": {
39    "name": "mysql"
40  }
41 }
42

```

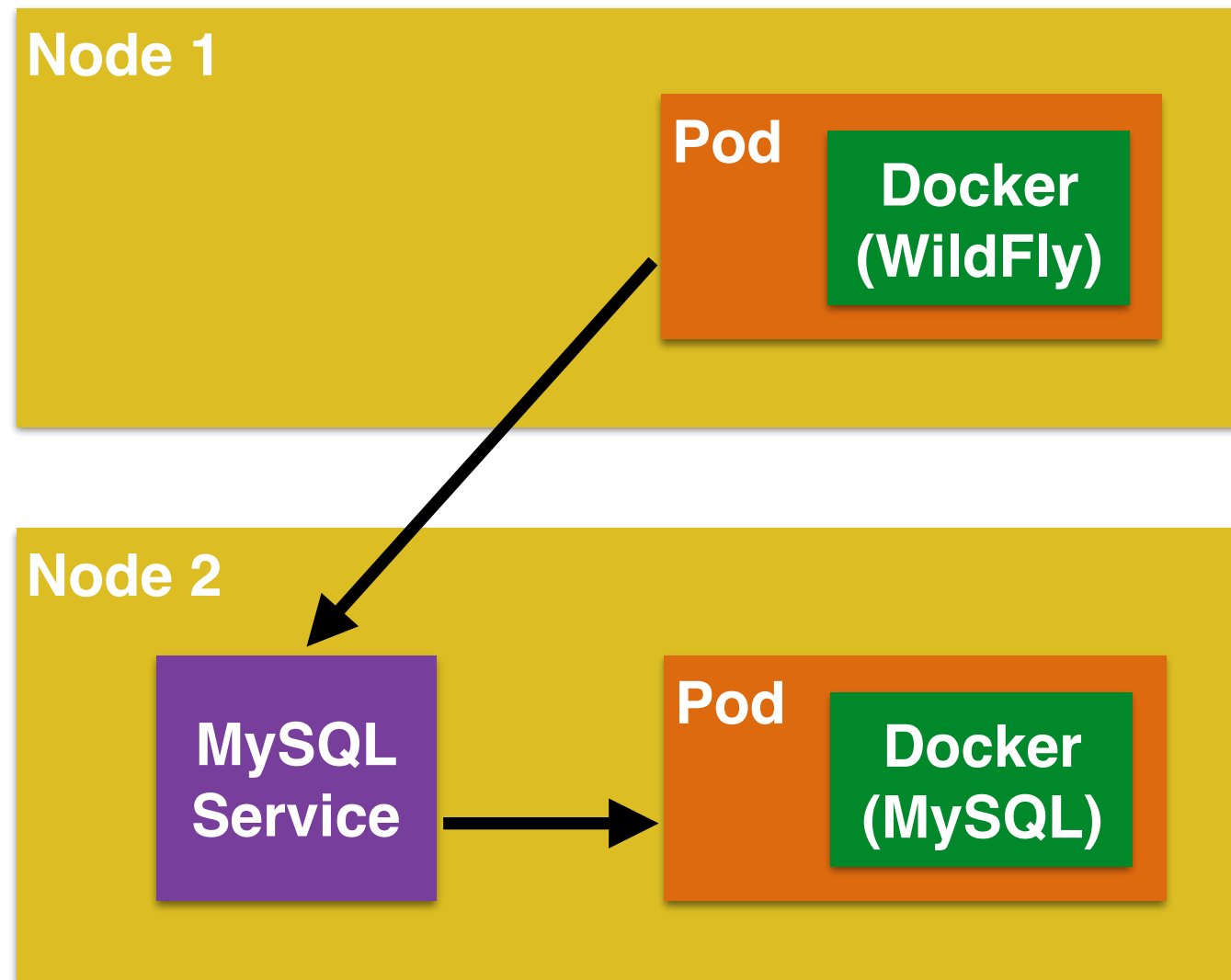


```

1 {
2   "id": "mysql",
3   "kind": "Service",
4   "apiVersion": "v1beta1",
5   "port": 3306,
6   "containerPort": 3306,
7   "selector": {
8     "name": "mysql"
9   },
10  "labels": {
11    "name": "mysql"
12  }
13 }

```

Two Nodes



Replication Controller

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


```
1 id: wildfly-controller
2 kind: ReplicationController
3 apiVersion: v1beta1
4 desiredStateReplicas: 2
5 replicaSelector:
6   name: wildfly
7 podTemplate:
8   desiredState:
9     manifest:
10      version: v1beta1
11      id: wildfly
12      containers:
13        - name: javaee7-hol
14          image: arungupta/javaee7-hol
15 labels:
16   name: wildfly
```

```
1 id: wildfly
2 kind: Service
3 apiVersion: v1beta1
4 port: 8080
5 containerPort: 8080
6 selector:
7   name: wildfly
8 labels:
9   name: wildfly
```

Node 1

WildFly
Service

Pod

Docker
(WildFly)

Pod

Docker
(WildFly)

Node 2

MySQL
Service

Pod

Docker
(MySQL)

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