Open Source Datacenter Systems

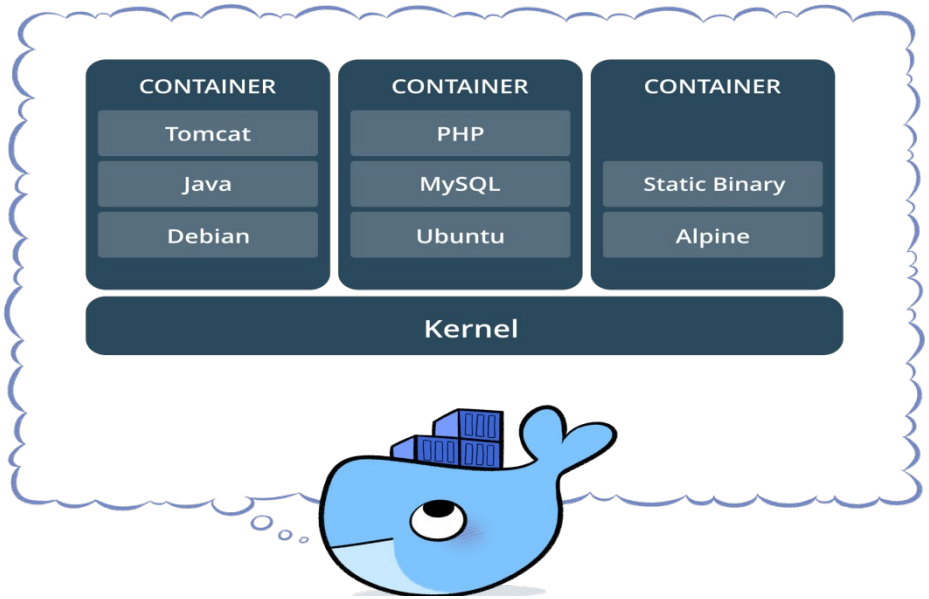
Kubernetes

Containers

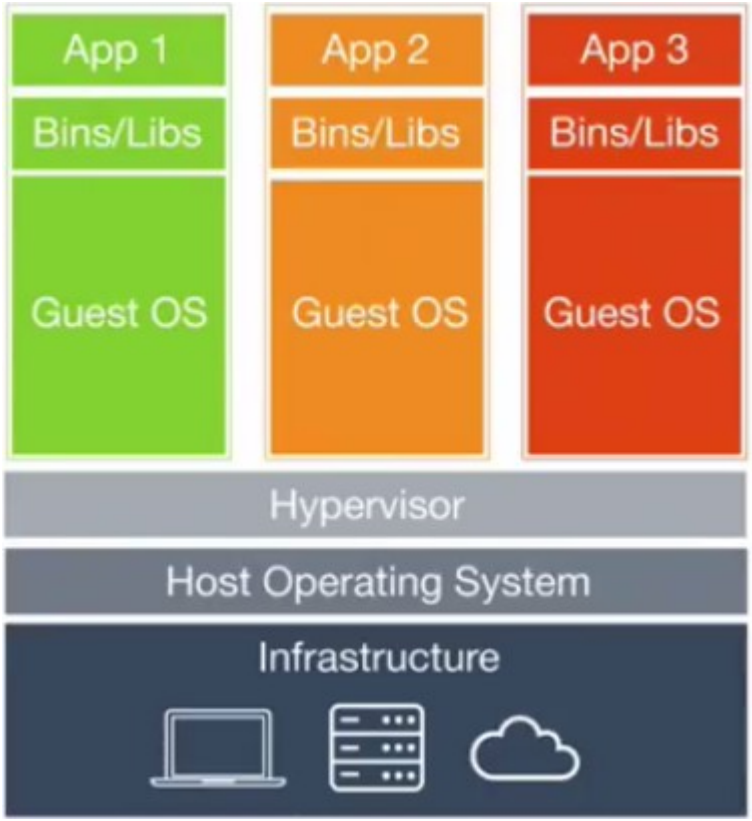
- A container image is a lightweight, stand-alone, executable package of a piece of software that includes everything needed to run it: code, runtime, system tools, system libraries, settings. Available for both Linux and Windows based apps, containerized software will always run the same, regardless of the environment. Containers isolate software from its surroundings, for example differences between development and staging environments and help reduce conflicts between teams running different software on the same infrastructure.

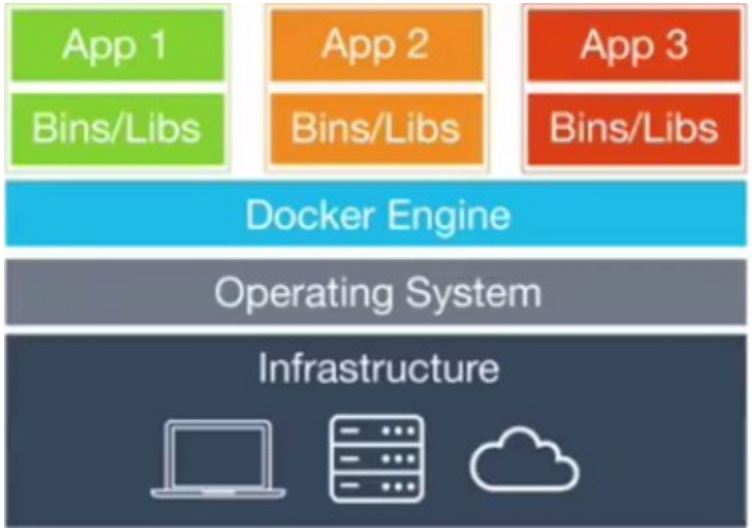
```
version: "2"
services:
  db:
    container_name: "db"
    image: couchbase
    ports:
      - 8091:8091
      - 8092:8092
      - 8093:8093
      - 11210:11210
  web:
    image: arungupta/wildfly
    environment:
      - COUCHBASE_URI=db
    ports:
      - 8080:8080
      - 9990:9990
```

docker-compose.yml



Container Vs Virtual Machine





Kubernetes Intro

Kubernetes is a production-grade, open-source platform that orchestrates the placement (scheduling) and execution of application containers within and across computer clusters.

Developed at Google, based on Borg. Kubernetes is written in GO.

Masters manage the cluster and the nodes are used to host the running applications.

A Kubernetes cluster can be deployed on either physical or virtual machines or on cloud.

A Kubernetes cluster consists of two types of resources:

- The Master coordinates the cluster
- Nodes are the workers that run applications

The Kubernetes Master is a collection of three processes that run on a single node in a cluster, which is designated as the master node. Those processes are: kube-apiserver, kube-controller-managerand kube-scheduler.

Container Orchestration

- Container Orchestration refers to the automated arrangement, coordination, and management of software containers.
- Many tools including the following provide orchestration:
  - **Kubernetes**
  - AWS ECS
  - Docker Swarm
  - Mesos
  - Nomad
- Container agnostic (Docker, rkt, LXC, Vagrant)
- Self-healing
- Auto-restarting
- Schedule across hosts
- Replicating
- Provides declarative primitives for a desired state

Kubernetes

Core Concepts

Master node: Runs multiple controllers that are responsible for the health of the cluster, replication, scheduling, endpoints (linking *Services* and *Pods*), Kubernetes API, interacting with the underlying cloud providers etc. Generally it makes sure everything is running as it should be and looks after *worker nodes*.

Worker node (minion): Runs the Kubernetes agent that is responsible for running *Pod* containers via Docker or rkt, requests secrets or configurations, mounts required *Pod* volumes, does health checks and reports the status of *Pods* and the node to the rest of the system.

Pod: The smallest and simplest unit in the Kubernetes object model that one can create or deploy. It represents a running process in the cluster. Can contain one or multiple containers.

Kubernetes

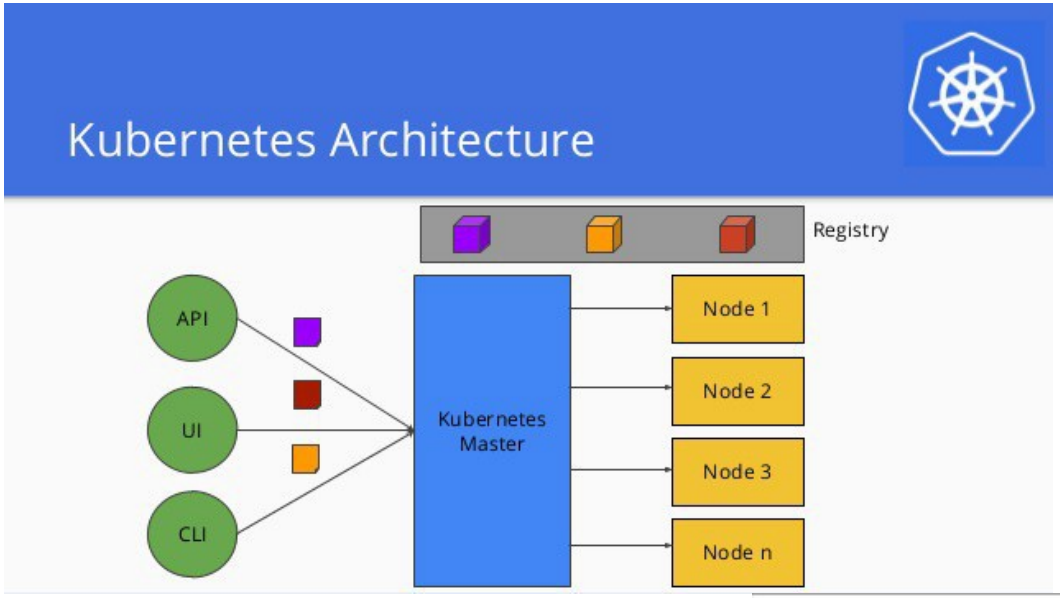
Core Concepts

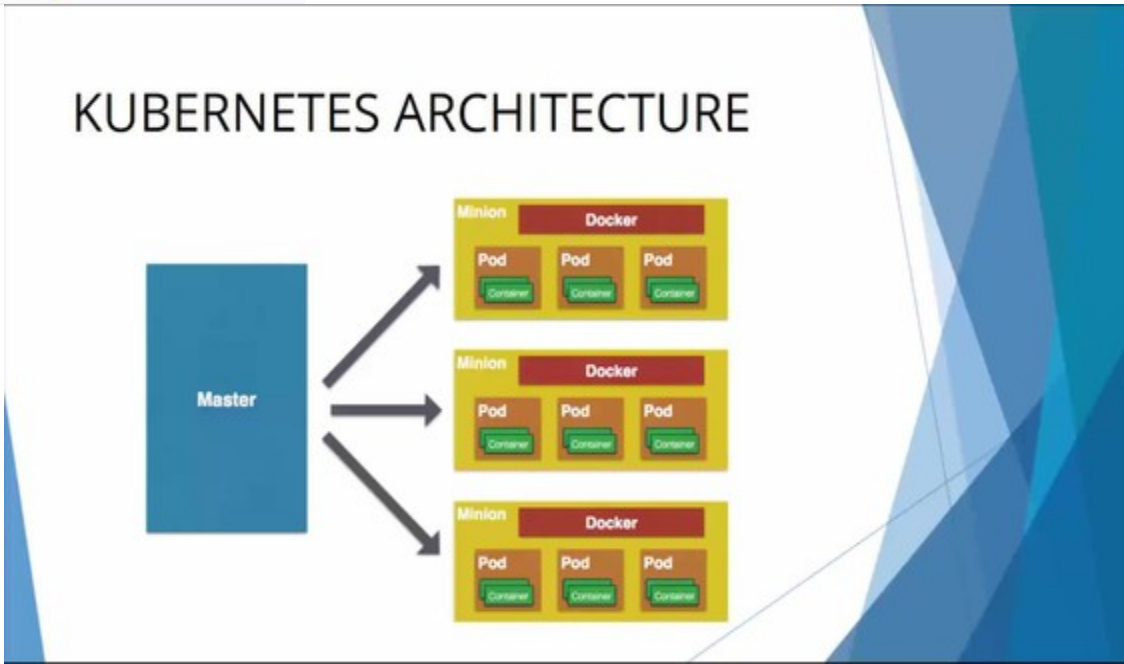
Deployment: Provides declarative updates for *Pods* (like the template for the *Pods*), for example the Docker image(s) to use, environment variables, how many *Pod* replicas to run, labels, node selectors, volumes etc.

ReplicaSet: Controller that ensures a specified number of *Pod* replicas (defined in the *Deployment*) is running at any given time.

Service: An abstraction which defines a logical set of *Pods* and a policy by which to access them (determined by a label selector). Generally it's used to expose Pods to other services within the cluster (using targetPort) or externally (using NodePort or LoadBalancer objects).

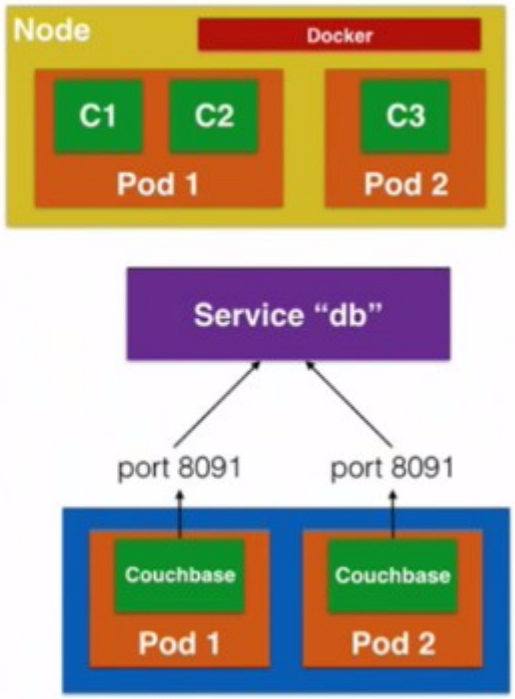
Kubernetes Architecture





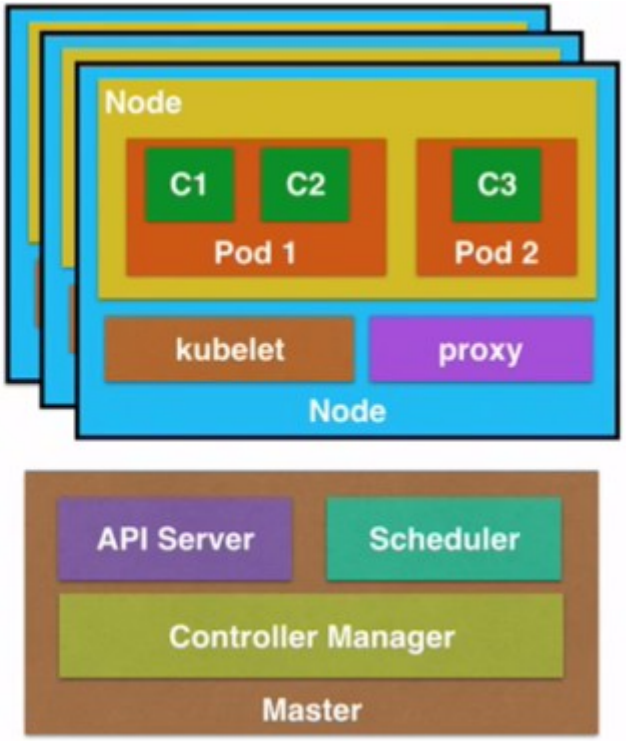
Kubernetes Architecture

- **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

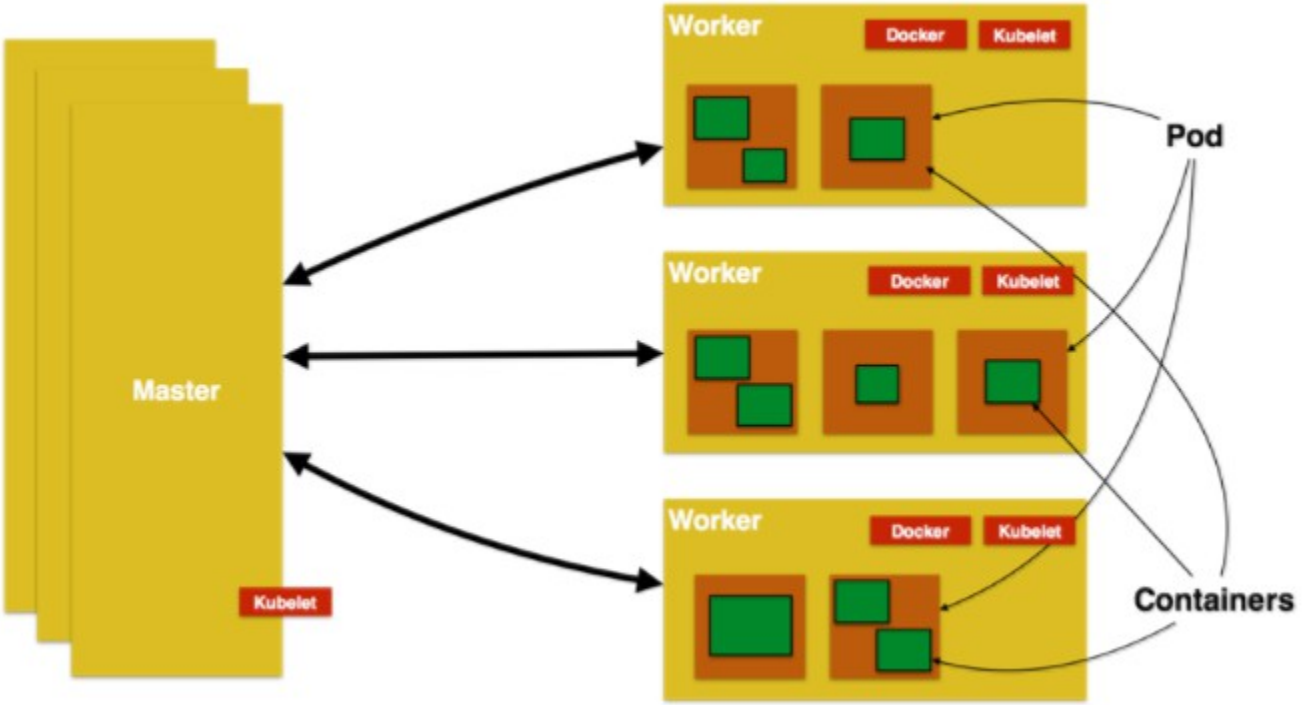


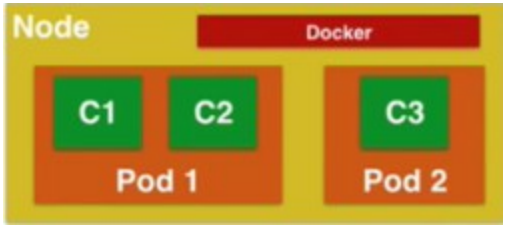
Kubernetes Architecture

- **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

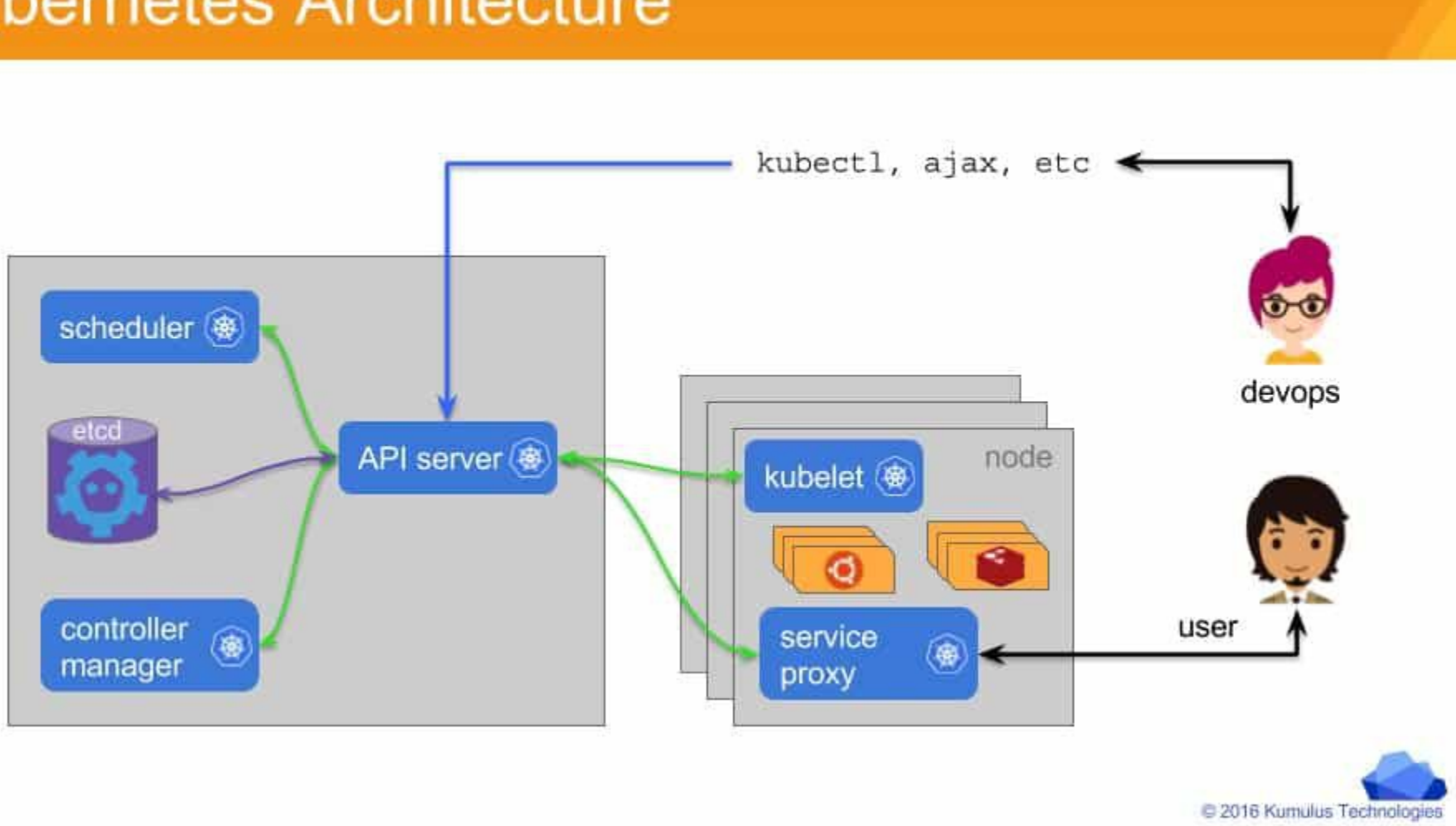


Kubernetes Architecture

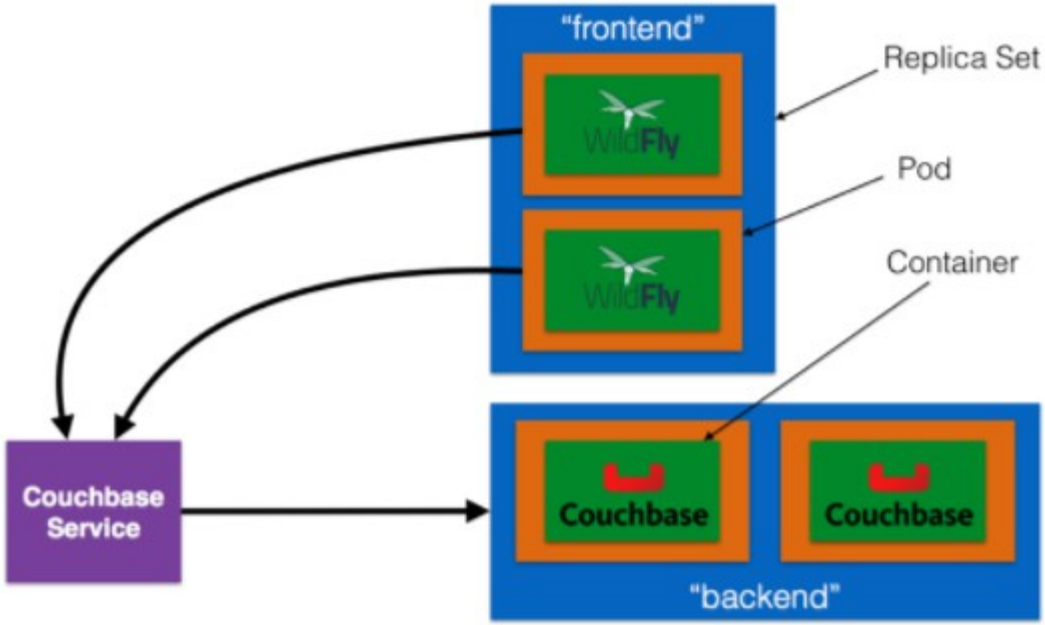




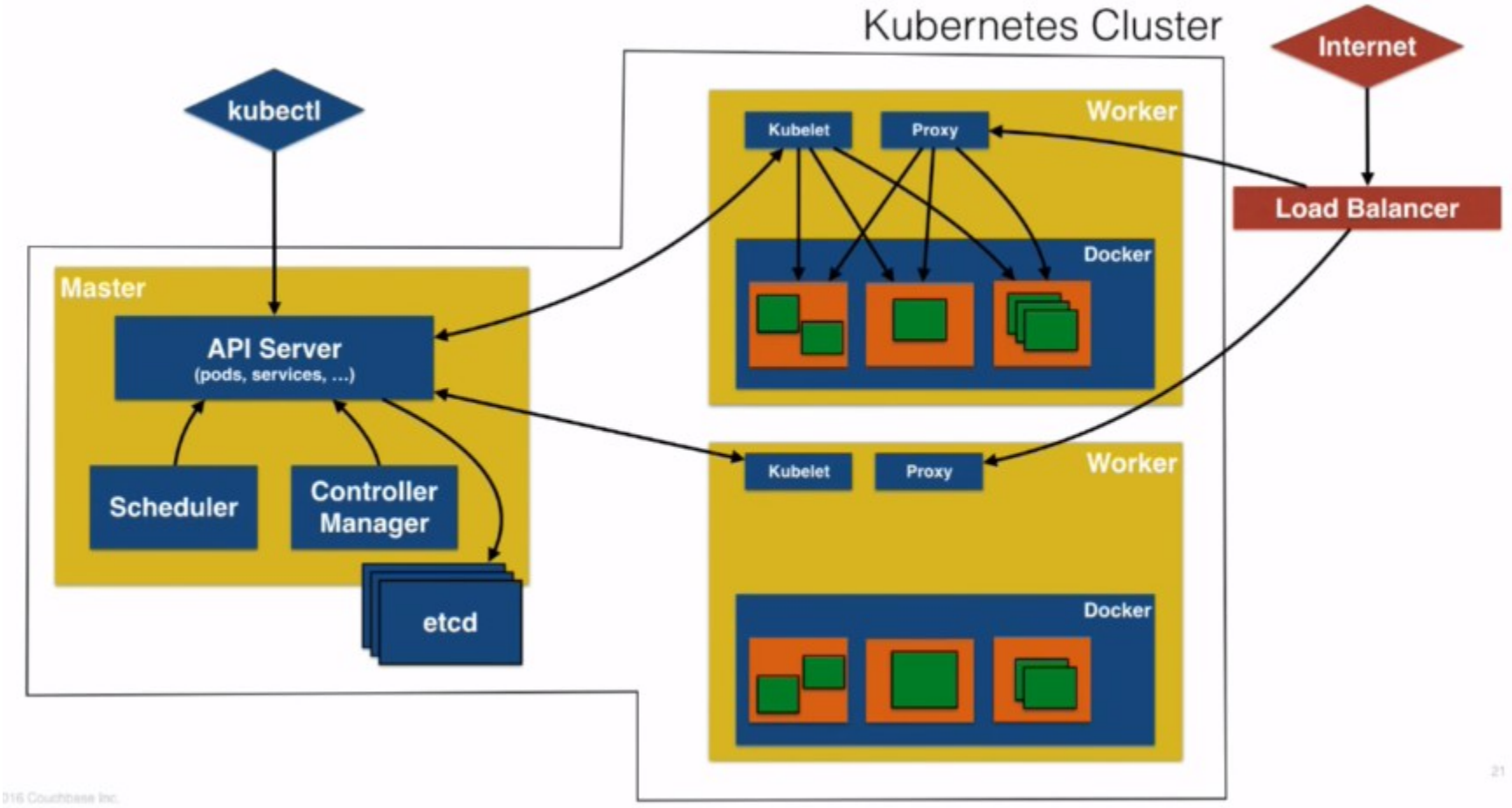
Kubernetes Architecture



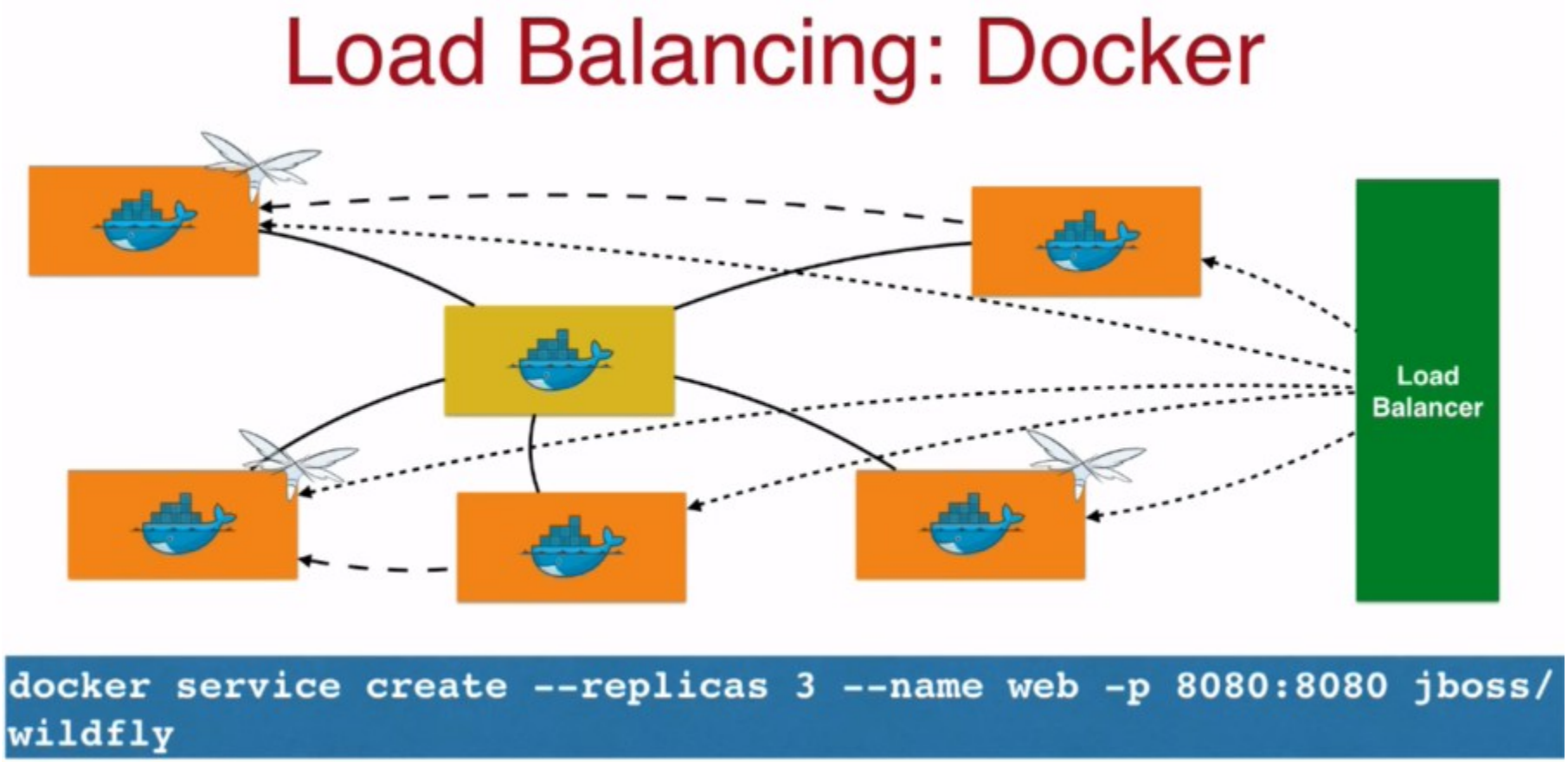
Kubernetes Deployment



Kubernetes Deployment

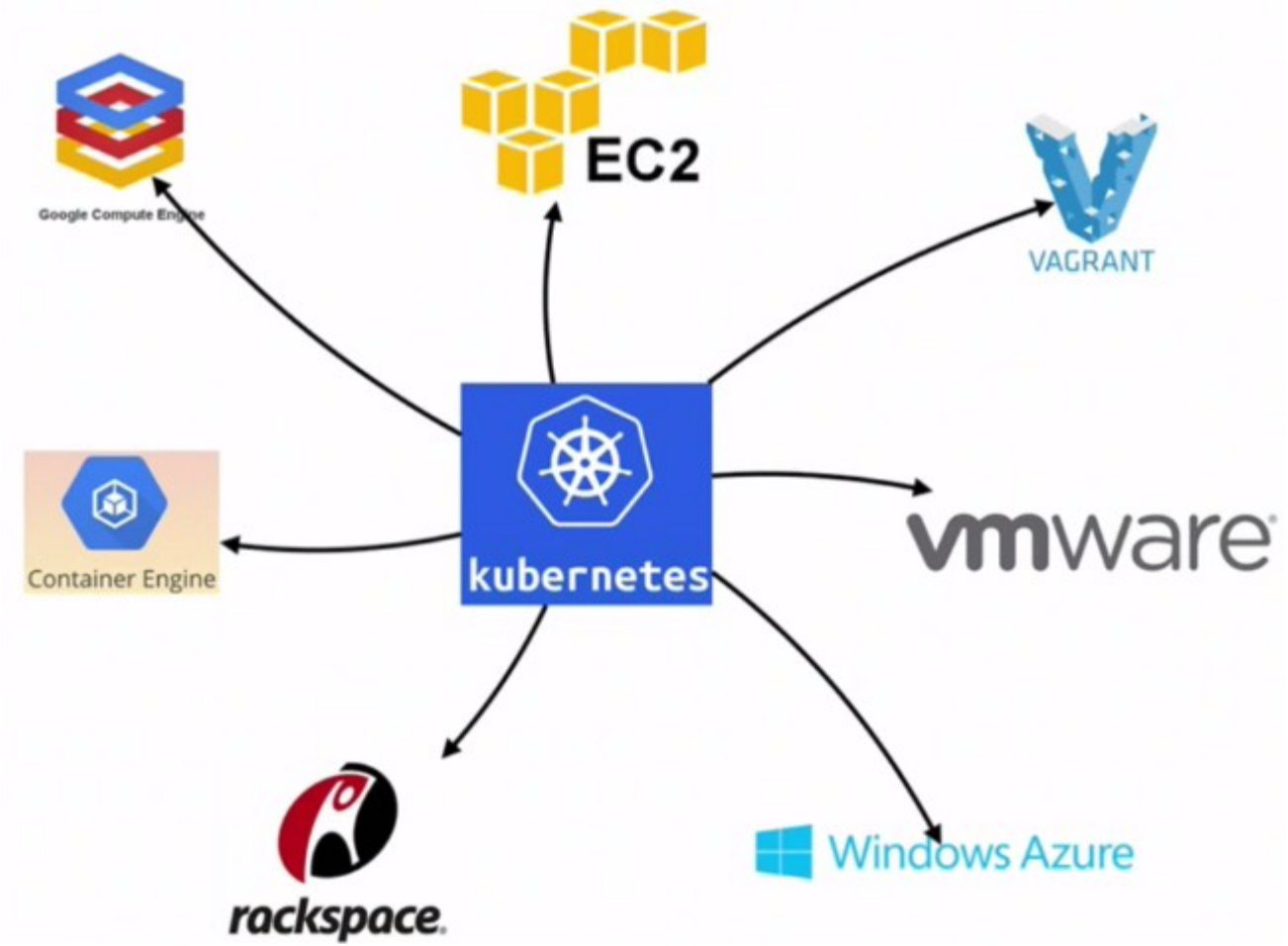


Kubernetes Deployment




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

Kubernetes Platform



Kubernetes in the cloud:

- Google Kubernetes Engine (GKE)
- Azure Kubernetes Service (AKS)
- Amazon Elastic Kubernetes Service (Amazon EKS)

Kubernetes Tools -- Kubectl

COMMAND	PURPOSE
kubectl create -f couchbase-pod.yml	Create a Couchbase pod
kubectl create -f couchbase-rs.yml	Create a Couchbase Replica Set
kubectl delete -f couchbase-pod.yml	Deletes/Removes the Couchbase pod
kubectl get pods	List all the pods
kubectl describe pod couchbase-pod	Describe the Couchbase pod
kubectl --help	Shows the complete list of available commands.

- 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 Tools

- cAdvisor - container usage and performance
- Heapster – container state
- Prometheus – monitoring
- Portworx - persistent volumes
- Minikube – single node test environment

Template

nginx-deployment.yaml docs/concepts/overview/working-with-objects

```yaml
apiVersion: apps/v1 # for versions before 1.9.0 use apps/v1beta2
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.7.9
        ports:
        - containerPort: 80
```

Key Takeaways

- Kubernetes is the leading container orchestration tool from Google
- Highly scalable
- Highly manageable
- Resilient to failures
- Supports rolling updates
- Container agnostic
- Can be deployed on GCE, AWS, Azure and other cloud providers
- Resources/Further Reading
  - Kubernetes.io
  - Docker.io
  - http://Hub.docker.com
  - http://blog.arungupta.me/ (material referenced)