Workshop Part-1

By

Prince Khanna

Agenda

+What and Why about Kubernetes?

+Architecture of Kubernetes

+Basic Concepts of Kubernetes

+Hands On

+Exercise

What is Kubernetes?

Kubernetes, also known as K8s, is an open-source system for automating deployment, scaling, and management of containerized applications.

Kubernetes is **an orchestration tool for containerized applications**.

Why Kubernetes?

Kubernetes is an orchestration tool for containerized applications. It is responsible for:

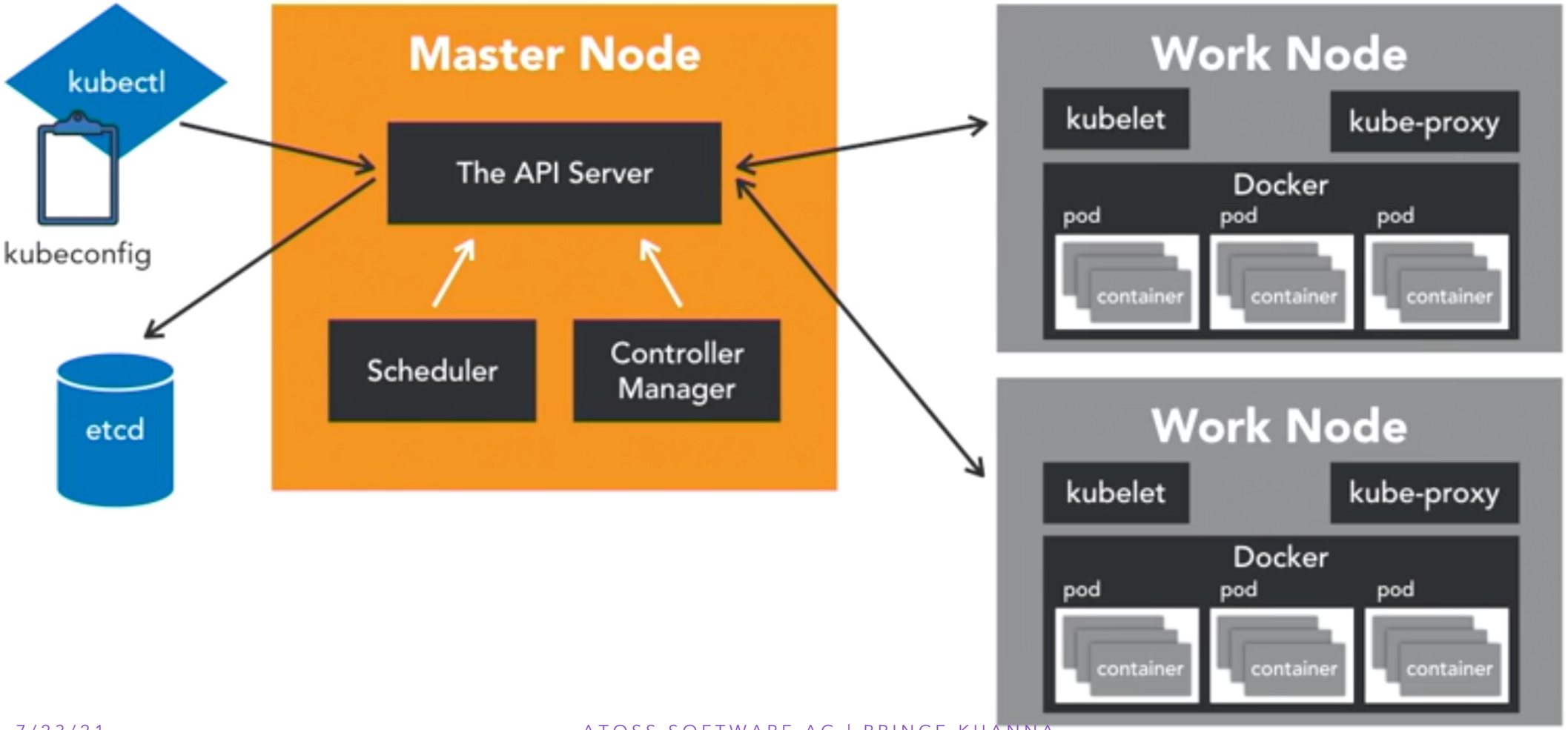
+Deploying images and containers

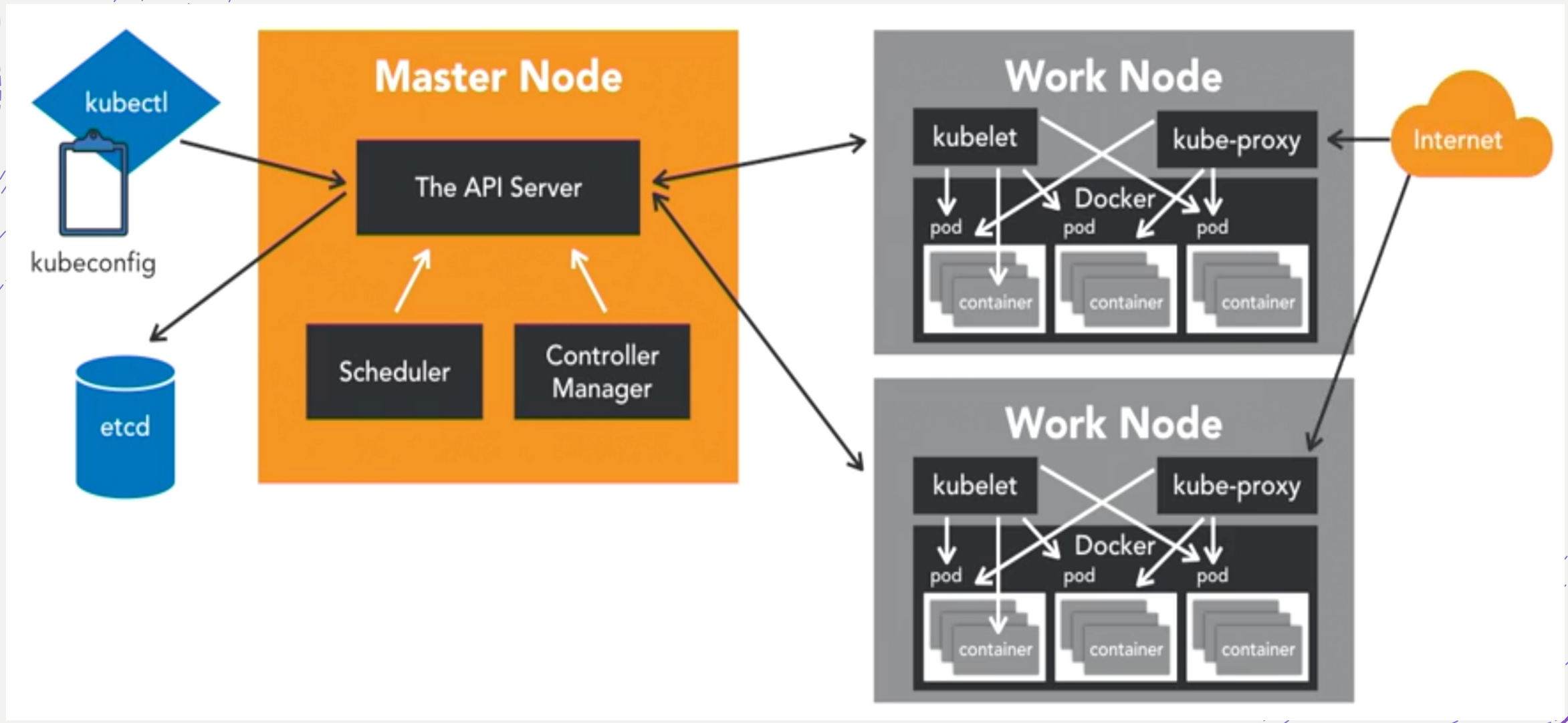
+Managing the scaling of containers and clusters

+Resource balancing containers and clusters

+Traffic management for services

Architecture of Kubernetes




kubectl
kubeconfig
etcd
Master Node
The API Server
Scheduler
Controller Manager
Work Node
kubelet
kube-proxy
Docker
pod
pod
pod
container
container
container
Internet
Work Node
kubelet
kube-proxy
Docker
pod
pod
pod
container
container
container

Basic Concepts

+ Namespace
+ Pods
+ Replica set
+ Deployment
+ Service
+ Config Map
+ Secret
+ Volumes
+ Ingress
+ Job

Pods

Pods are the **smallest** deployable units of **computing** that you can create and manage in Kubernetes.

A *Pod* is a group of one or more **containers**, with shared storage and network resources, and a specification for how to run the containers.

Sample Pod yaml File

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  labels:
    name: nginx
spec:
  containers:
  - name: nginx
    image: nginx
    ports:
    - containerPort: 80
```

Replica set

A ReplicaSet's purpose is to maintain a stable set of replica Pods running at any given time. As such, it is often used to guarantee the availability of a specified number of identical Pods.

Deployment

A *Deployment* provides declarative updates for **Pods** and **ReplicaSets**.

Sample Deployment yaml File

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: nginx-deployment
 labels:
  app: nginx
spec:
 replicas: 3
 selector:
  matchLabels:
   app: nginx
 template:
  metadata:
   labels:
    app: nginx
  spec:
   containers:
   - name: nginx
    image: nginx:1.14.2
    ports:
    - containerPort: 80
```

Service

An abstract way to expose an application running on a set of **Pods** as a

Sometimes you don't need load-balancing and a single Service IP. In this case, you can create what are termed "headless" Services, by explicitly specifying "None" for the cluster IP (.spec.clusterIP). network service.

Sample Service.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  selector:
    app: nginx
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
```

Demo Time

Q&A

References

Pods | Kubernetes

Pod Lifecycle | Kubernetes

Init Containers | Kubernetes

Deployments | Kubernetes

StatefulSets | Kubernetes

ReplicaSet | Kubernetes

TTL Controller for Finished Resources | Kubernetes

Service | Kubernetes