



Container Orchestration using Docker & Kubernetes

By Mahendra Shinde

Microsoft
Partner

Silver Learning
Gold Cloud Platform



Educate

Advise

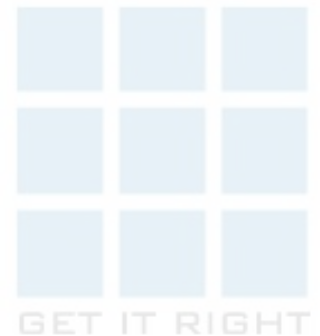
Implement

Manage

Agenda

Module 6: Working with Kubernetes (Architecture)

Module 7: Working with Kubernetes (Deployment)





Module 6

Working with Kubernetes (Architecture)

Educate

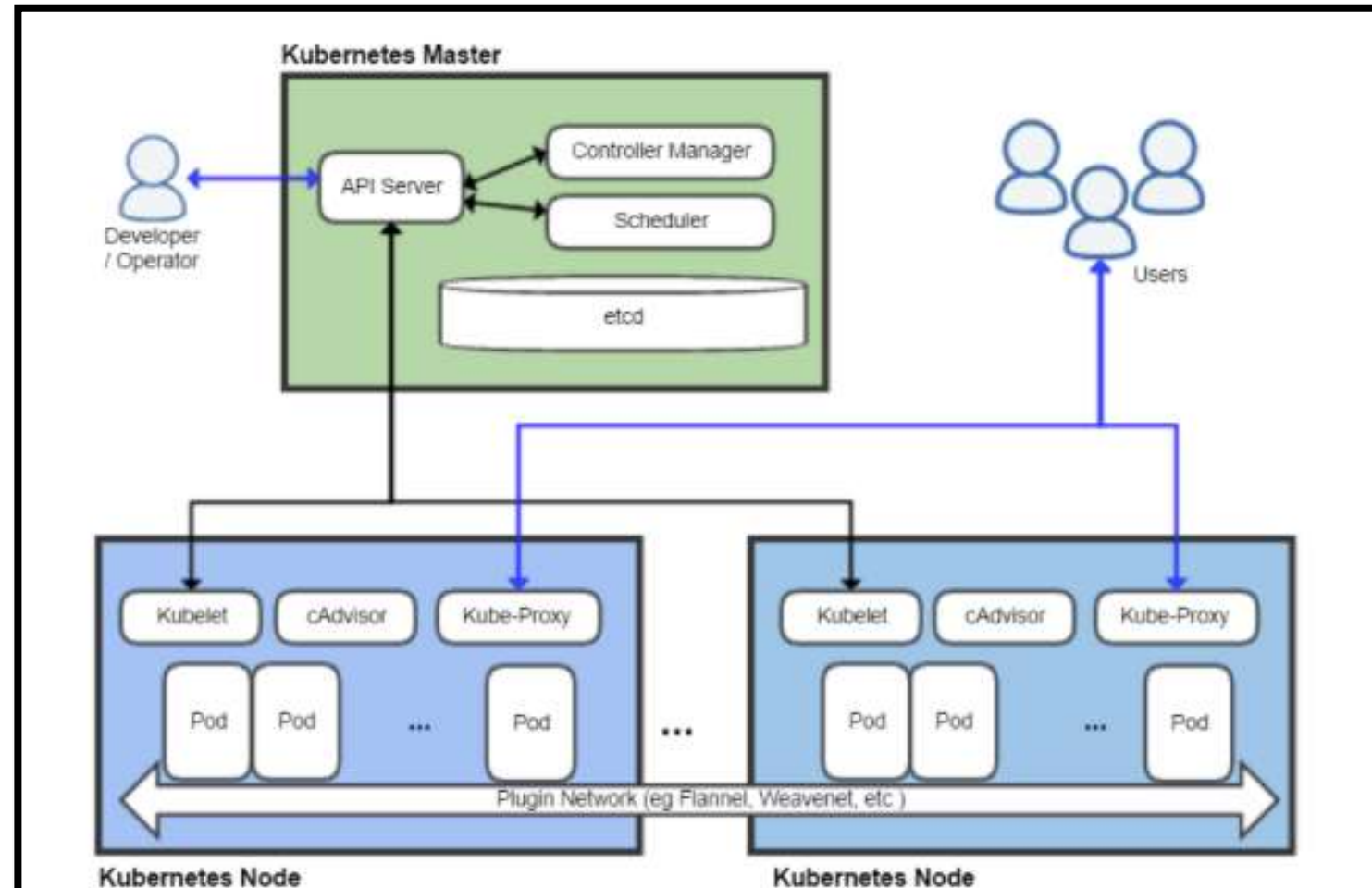
Advise

Implement

Manage

Kubernetes Architecture

“Kubernetes is an open-source system for automating deployment, scaling, and management of containerized applications.”



Kubernetes Components

■ Nodes

- A Physical or Virtual machines on kubernetes cluster.
- Can be either a master (manager) node or worker (deployment target) node.
- All the nodes must be in a private network
- Master nodes must have linux based operating system.
- Worker nodes can be either linux or windows based systems.
- Master node need few additional components than worker nodes



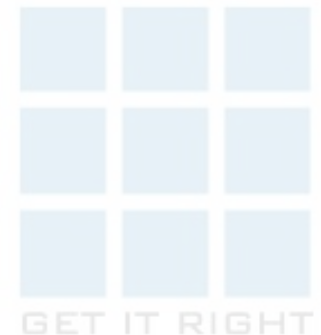
Kubernetes : Master Components

- api-server
 - A Front end for kubernetes control plane.
 - Designed to scale horizontally.
- Etcd
 - Consistent & Highly available key-value store
 - Used as data store for all cluster data
- Scheduler
 - Put a pod on node!
 - Scheduling decision is based on
 - Affinity
 - Anti-affinity
 - Policy/Resource/Hardware/Software constraints
- Controller Manager
 - Responsible for following managers
 - Node controller
 - Replication controller
 - Endpoint Controller
 - Service Account and tokens controller



Kubernetes : Node (Worker) Components

- Kubelet
 - An agent that runs on each node.
- Kube-proxy
 - Enables network abstraction by maintaining network rules on the host
 - Does connection forwarding
- Container runtime
 - Container runtime either Docker or RKT
 - Any other container runtime based on runc & oci
- Add-Ons
 - Cluster DNS
 - Web-UI



Kubernetes Client tool (kubectl)

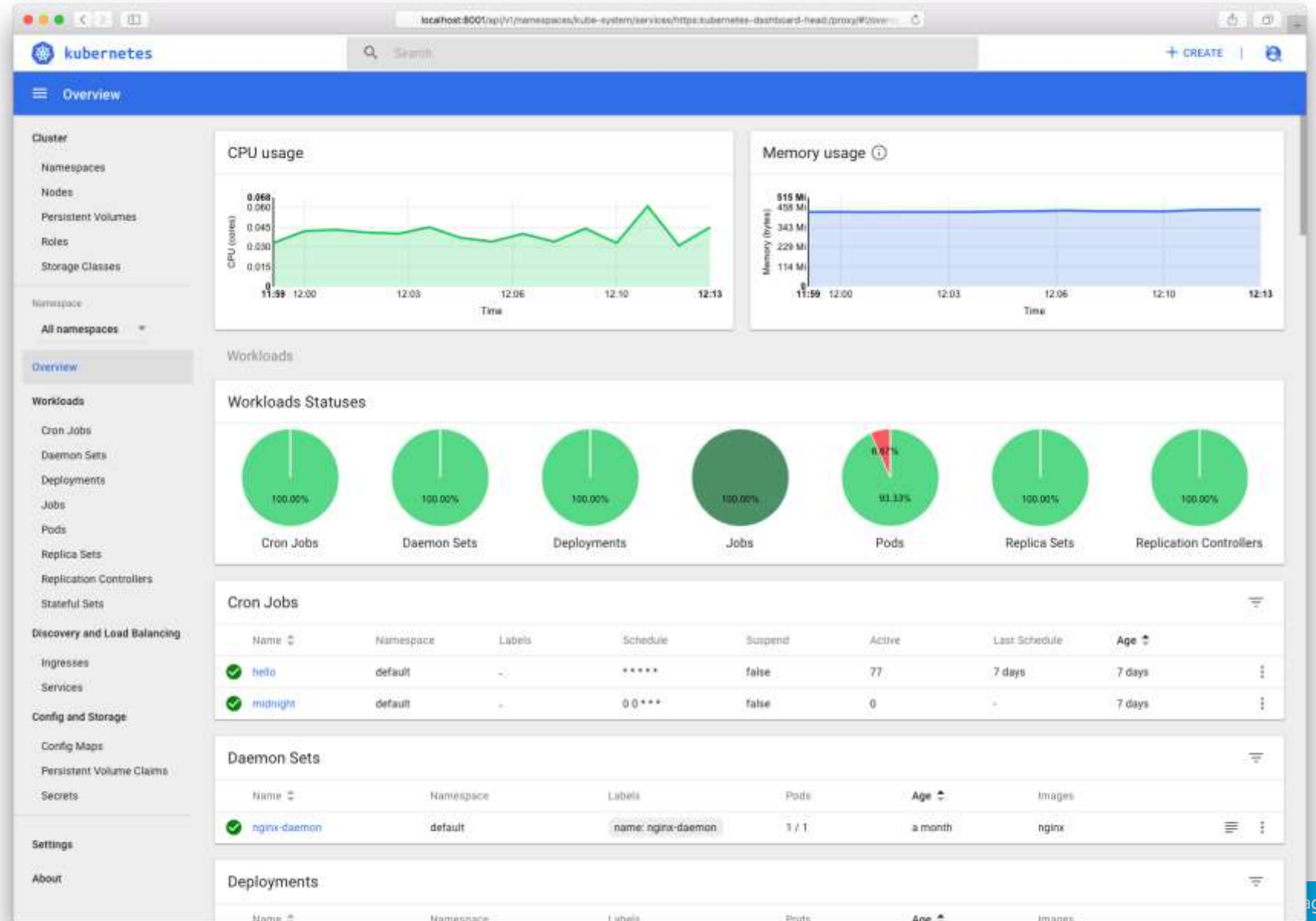
- CLI for managing kubernetes cluster
- Syntax :

`$ kubectl [command] [TYPE] [NAME] [flags]`

- Command: An operation like create, get, describe or delete
 - TYPE: A resource type like pod, service etc
Case Sensitive, can be plural, like pods instead of pod
 - NAME: Name of resource to be operated
 - Flags: Optional flag, depends on resource type
- Example:
 - `$ kubectl get pods`
 - `$ kubectl describe deployment app1`
 - `$ kubectl get pod -f mypod.yml`

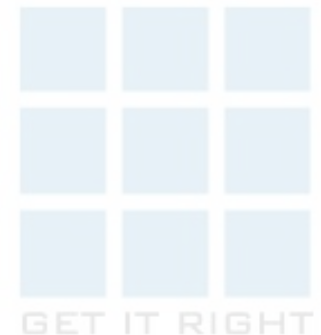


Kubernetes Dashboard [Web-UI]



Single Host deployment for developers

- Minikube :
A simple standalone dev cluster for anyone who wants to get started with kubernetes.
- Uses Hypervisor like Oracle VirtualBox or HyperV.
- On Linux host, hypervisor could be set to NONE.



Minikube architecture

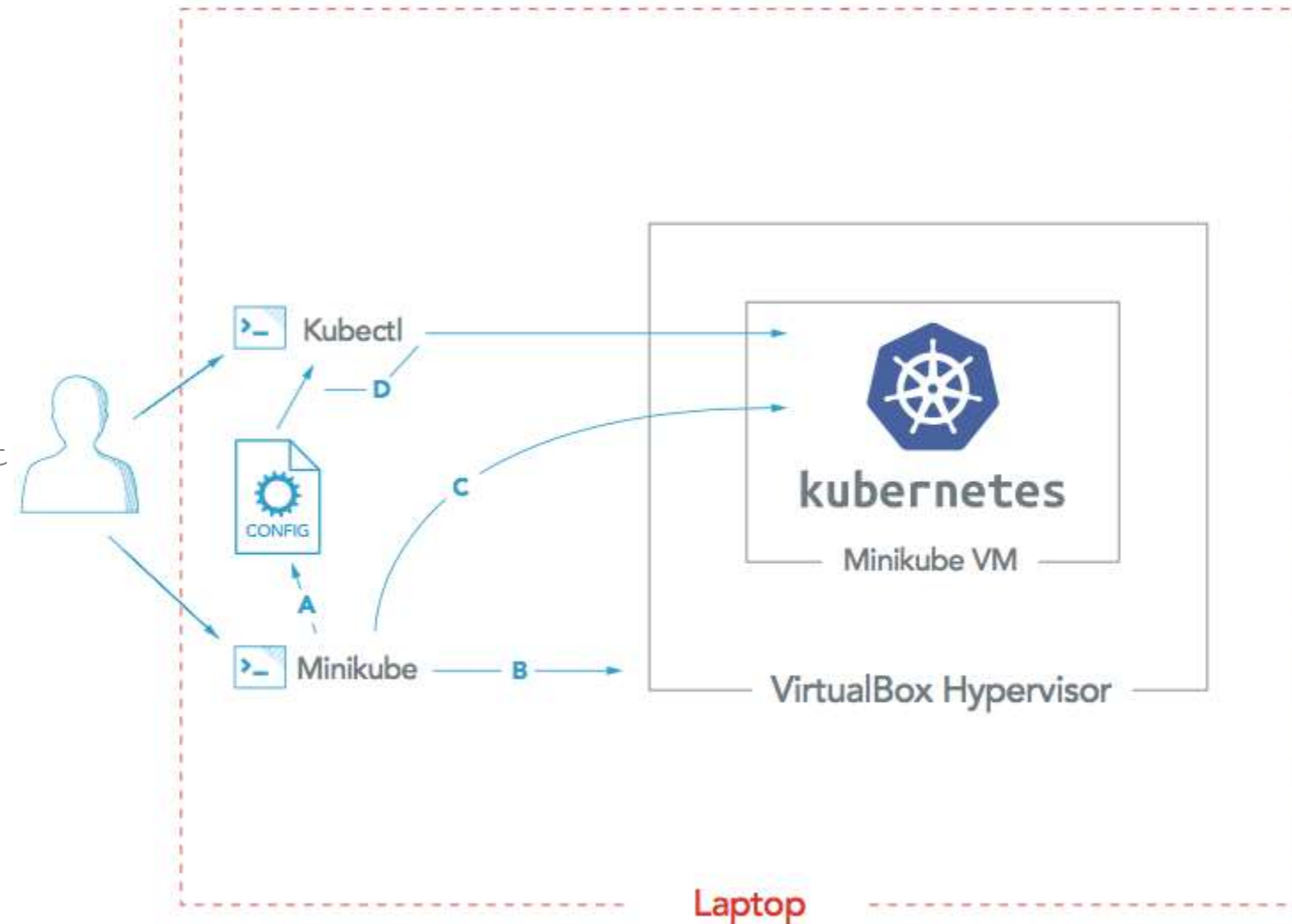
- Entire cluster as single VM
- Host system as user workstation
- Use kubectl to access cluster

- Simple commands:

```
$ minikube start -vm-driver hyperv
  --hyperv-virtual-switch="Swit
```

```
$ minikube delete
```

```
$ minikube dashboard
```



A: Minikube generates kubeconfig file
B: Minikube creates Minikube VM

C: Minikube sets up Kubernetes in Minikube VM
D: Kubectl uses kubeconfig to work with Kubernetes

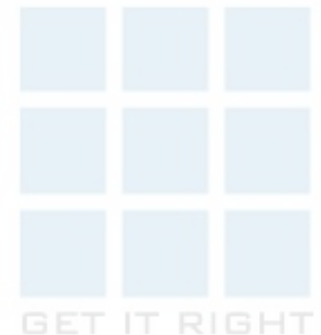
Multi Host deployments

- Ideal for production like environments
- Provides High availability [by multiple master nodes]
- Load balance workload across multiple worker nodes
- Secure & reliable cluster experience
- Minikube is ideal only for first time users to learn kubernetes
- Kubernetes provide following options
 - On cloud (By vendors like Microsoft Azure, AWS, Google)
 - On Premise (Using kubeadm)



kubeadm

- Kubeadm is tool to bootstrap kubernetes cluster (either on premise or on cloud)
- Uses Signed certificates for secure communication across cluster.
- Most common operations
 - Kubeadm init (bootstrap a cluster from Master node)
 - Kubeadm join (Join existing cluster from Worker node)
- System requirement
 - One or more machines running Debian or RedHat based linux
 - 2 GB or more RAM per machine
 - 2 or more CPUs per machine
 - Full network connectivity between all machines (Private/Public Network)





Module 7

Working with kubernetes (Deployment)

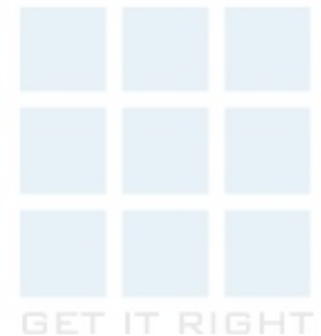
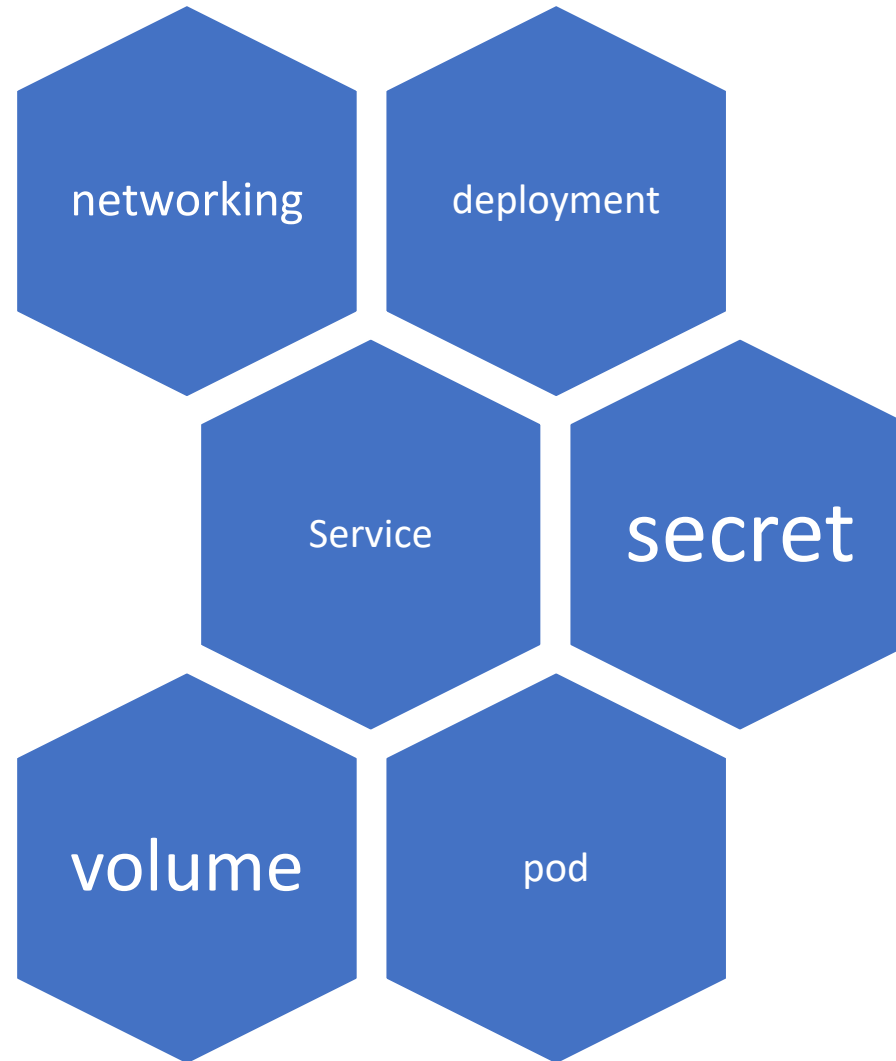
Educate

Advise

Implement

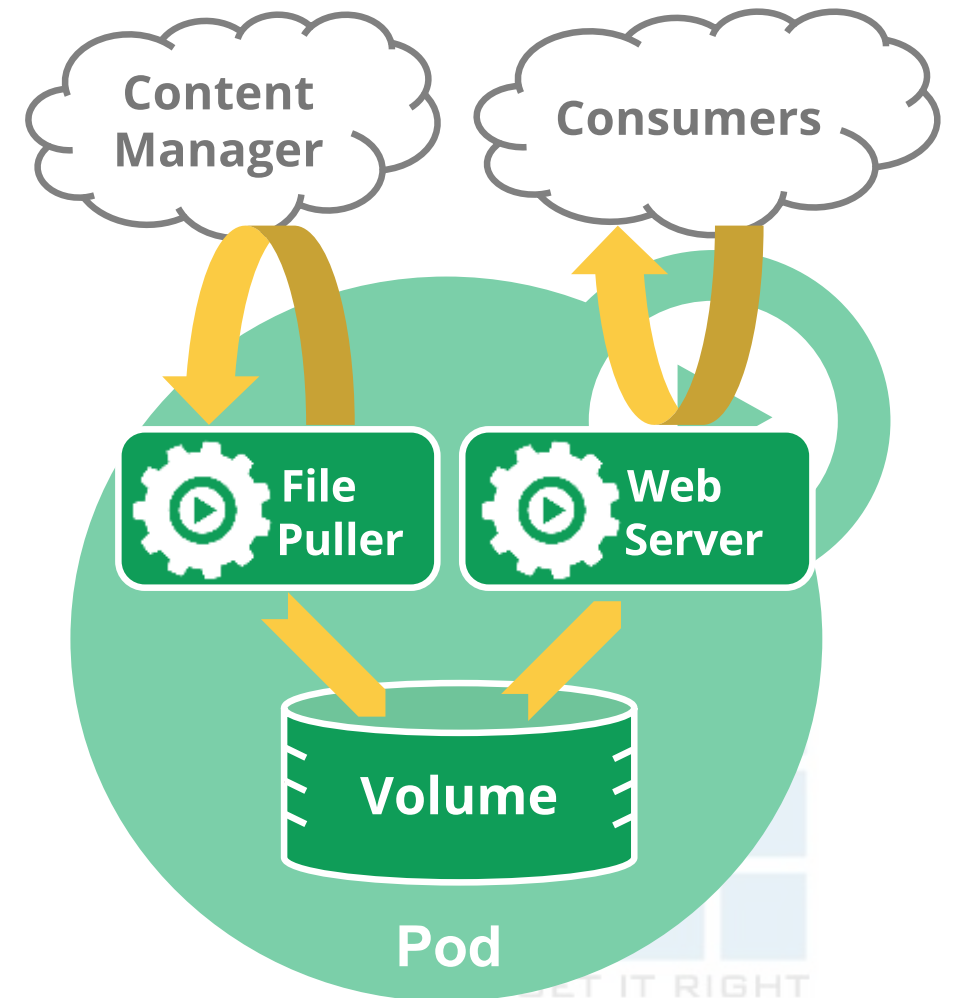
Manage

Kubernetes Concepts



Kubernetes Concepts : pods

- Smallest deployable unit for kubernetes.
- Ephemeral, disposable entities (ref cattle vs pets)
- May contain one or more container
- Deployed / destroyed / scaled as single unit
- All containers in a pod are sharing resources [Storage, Compute & Network]
- All container in a pod are tightly coupled.



Kubernetes Concepts : pods

| Phase | Description |
|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Pending | Waiting for image downloading, not all containers ready yet. |
| Running | Pod deployed to a node, either all containers started or one is starting (images downloaded in pending phase) |
| Succeeded | All Containers in the Pod have terminated in success, and will not be restarted. |
| Failed | All Containers in the Pod have terminated, and at least one Container has terminated in failure. That is, the Container either exited with non-zero status or was terminated by the system. |
| Unknown | For some reason the state of the Pod could not be obtained, typically due to an error in communicating with the host of the Pod. |



Kubernetes Concepts : service

- An abstraction which defines logical set of pods and policy to access them.
- Services establish a single endpoint for collection of replicated pods, distributing inbound traffic based on label selectors.
- In kubernetes modelling language they represent a “Load Balancer”.
- Pods and Services exists independently, have disjoint lifecycle.
- Supports both tcp & udp protocols (default: tcp)
- Kubernetes assigns VIP to service.
- Kubernetes DNS scheduler should assign a DNS label to each service.
 - Eg: service named “db” can be accessed by other services by name “db”



Kubernetes Concepts : deployment

- A Desired State of Services, Pods, ReplicaSet etc.
- Can be written using YAML or JSON syntax
- Comparable to “docker-compose.yml” used in day 1
- Even deployments initiated by kubectl command are internally stored as YAML files.
- The sample deployment named “nginx-deployment”

```
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:1.7.9
          ports:
            - containerPort: 80
```

Kubernetes Concepts : Secrets

- Intended to hold sensitive information like passwords, OAuth tokens or SSH keys
- Kubectl has few command to manage secrets

\$ kubectl get secrets

\$ kubectl create secret

A reference to secret can be added to deployment [yaml] file

```
apiVersion: v1
kind: Pod
metadata:
  name: mypod
spec:
  containers:
  - name: mypod
    image: redis
    volumeMounts:
    - name: foo
      mountPath: "/etc/foo"
      readOnly: true
  volumes:
  - name: foo
    secret:
      secretName: mysecret
```

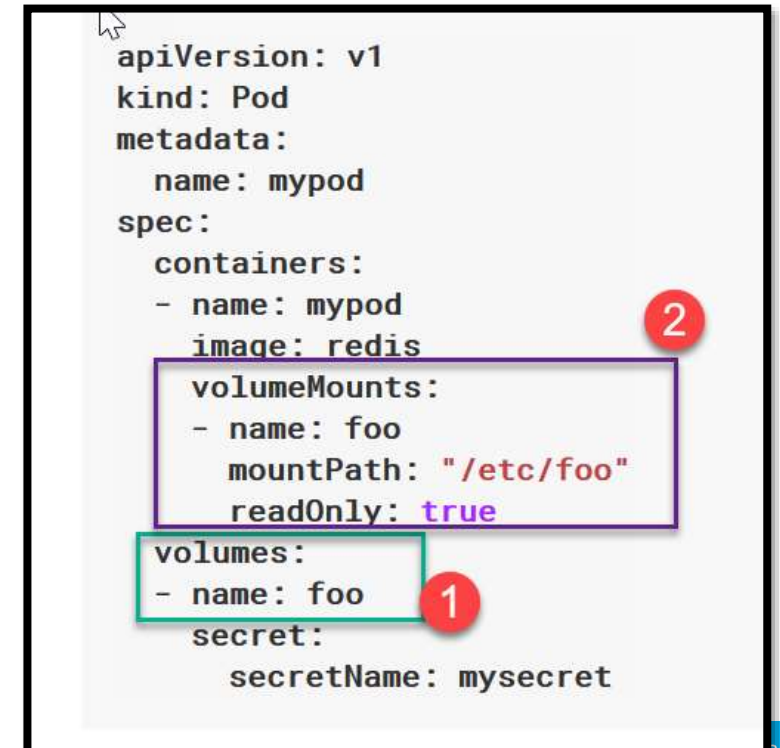
Kubernetes Concepts : Volumes

- A Persistent data store for containers
- A container restart results in creating “clean container” (data in Writable layer lost!)
- Volumes can persist data between services or pods (containers sharing single pod)
- Deep down, Its just a directory made accessible to one or more containers / pods.
- Kubernetes supports following volumes
 - azureDisk
 - azureFile
 - Cephfs
 - Csi

```

apiVersion: v1
kind: Pod
metadata:
  name: mypod
spec:
  containers:
    - name: mypod
      image: redis
      volumeMounts:
        - name: foo
          mountPath: "/etc/foo"
          readOnly: true
  volumes:
    - name: foo
      secret:
        secretName: mysecret

```



Kubernetes Concepts: Networking

- Networking in kubernetes cluster is different than that of docker
 - Highly coupled Container to Container communication inside a pod. (use localhost)
 - Pod to Pod communication (Pods have IP address)
 - Pod to service communication (Services have VIP)
 - External to Service
- Implementations
 - ACI
 - Cilium
 - Contiv
 - Flannel
 - Weave Net



Advanced Concepts

- Storage
 - Volumes
 - Persistent Volumes : Independent of pods, services or deployments
 - Storage classes (Used with Persistent Volumes)
- Compute
 - Pods can optionally define amount of resources like CPU or MEMORY
- Security
 - A Pod security policy (cluster level resource)
 - Can be managed by admission control plugin

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: azurefile
provisioner: kubernetes.io/azure-file
parameters:
  skuName: Standard_LRS
  location: eastus
  storageAccount: azure_storage_account_name
```

Advanced Concepts

- Stateful Sets
 - Targeted for Stateful applications
 - Ordered Pod creation
 - Stable network identity to each pod
 - Uses PersistentVolumeClaims

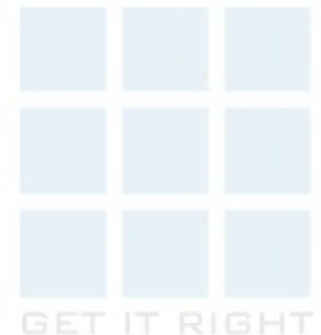
```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: web
spec:
  selector:
    matchLabels:
      app: nginx # has to match .spec.template.metadata.labels
  serviceName: "nginx"
  replicas: 3 # by default is 1
  template:
    metadata:
      labels:
        app: nginx # has to match .spec.selector.matchLabels
    spec:
      terminationGracePeriodSeconds: 10
      containers:
        - name: nginx
          image: k8s.gcr.io/nginx-slim:0.8
          ports:
            - containerPort: 80
              name: web
          volumeMounts:
            - name: www
              mountPath: /usr/share/nginx/html
      volumeClaimTemplates:
        - metadata:
            name: www
          spec:
            accessModes: [ "ReadWriteOnce" ]
            storageClassName: "my-storage-class"
            resources:
              requests:
                storage: 1Gi
```


Advanced Concepts

- Scaling (Manual / Automatic)

- Kubernetes uses “ReplicaSet” to ensure desired number of instances.
- Manual scaling requires updating “replicas” property.
- Auto (Horizontal) scaling can be created using “kubectl” command

```
$ kubectl autoscale rs app1rs --min=2 --max=5 --cpu-percent=80
```



Advanced Concepts

Rolling Update

- Update a service without an outage.
- Its works by
 - Creating a new replication controller with the updated configuration.
 - Increasing/decreasing the replica count on the new and old controllers until the correct number of replicas is reached.
 - Deleting the original replication controller.

- Example (Deprecated Version):

```
$ kubectl rolling-update NAME [NEW_NAME] --image=IMAGE:TAG
```

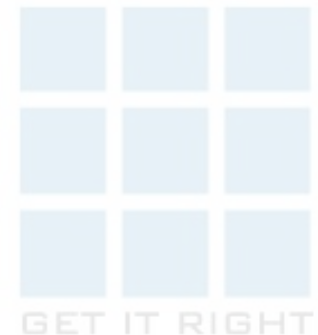
- Example (New Version):

- Just change the value of “replicas” in deployment file

```
$ kubectl apply -f <deployment-file>
```

Undo Previous Update (New Version):

```
$ kubectl rollout undo -f <deployment-file>
```

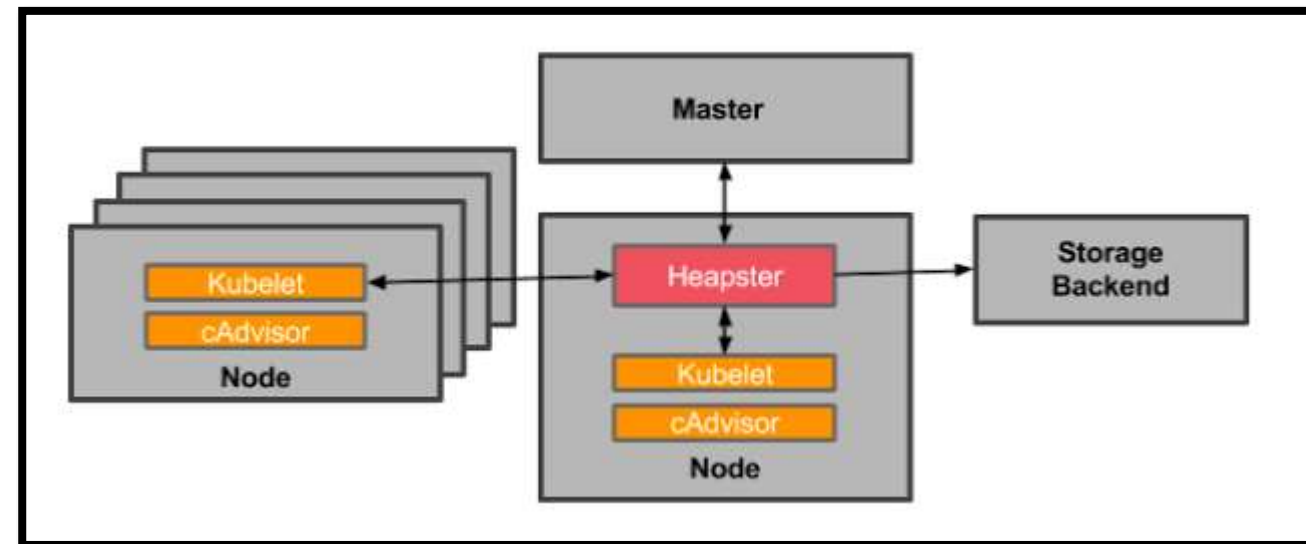


Advanced Concept

Monitoring (Infrastructure & Applications)

- Kubernetes dashboard provides basic monitoring
- Addon called “Heapster” can provide monitoring for entire cluster
- Uses cAdvisor

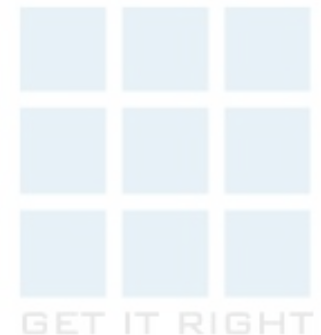
Container resource
monitoring (native to docker)



Advanced Concepts

Disaster Recovery

- etcd is a backing store for entire cluster data.
- All data is stored as JSON objects
- Always have a backup plan for etcd
- Run etcd as cluster of odd numbers
- Restore etcd data into new cluster





End Note

Educate

Advise

Implement

Manage



Q/A

Thank You

Educate

Advise

Implement

Manage