k8s

Kubernetes

- Kubernetes (k8s) is developed by google

- Kubernetes (k8s) is an open-source container-orchestration system for automating application deployment, scaling, load-balancing and management

Architecture

Node(Minions)	-	It is a machine(s) where kubernetes is installed
Kubernetes Cluster	-	Multiple nodes grouped together
Master	-	It controls all nodes in the kubernetes cluster
Worker	-	It is used to run containerized application(s)

Components in Kubernetes Master

api server	-	It is a frontend which interacts with kubernetes via UI / CLI
etcd	-	It is a key-value store used to store and manage cluster data
controller	-	It is a control loop that watches the state of the cluster
kube-scheduler	-	It is responsible for distributing container across multiple nodes

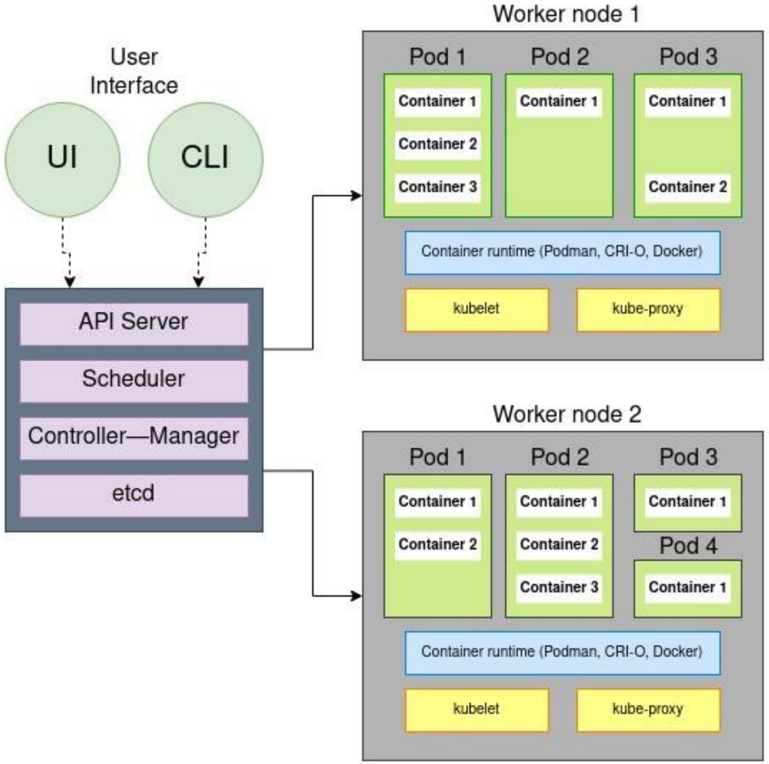
Components in Kubernetes Worker

kubelet	-	It is an agent runs on each node(s) in kubernetes cluster
container runtime	-	underlying software to run container eg: docker, rkt, cri-o

Pod

- container(s) are encapsulated into a kubernetes object
- It is a single instance of application
- Smallest object that can be created

Pods | Kubernetes


User Interface
UI
CLI
API Server
Scheduler
Controller—Manager
etcd
Worker node 1
Pod 1
Container 1
Container 2
Container 3
Pod 2
Container 1
Pod 3
Container 1
Container 2
Container runtime (Podman, CRI-O, Docker)
kubelet
kube-proxy
Worker node 2
Pod 1
Container 1
Container 2
Pod 2
Container 1
Container 2
Container 3
Pod 3
Container 1
Pod 4
Container 1
Container runtime (Podman, CRI-O, Docker)
kubelet
kube-proxy

kubectl	-	It is a command to run and manage container in cluster
minikube	-	It is a single node kubernetes cluster

Installing Kubectl and Minikube

Install Tools | Kubernetes

#follow the above link and install kubectl and minikube

Verify kubectl and minikube

#verify the version of kubectl

```
kubectl version --output=yaml | json
```

#start single node minikube cluster

```
minikube start
```

#verify the status of cluster

```
minikube status
```

Verify Nodes and Pods

```
#verify the nodes in kubernetes cluster

kubectl get nodes

#check the status of pods

kubectl get pods
```

Run a pod with Single Container

#run a pod in kubernetes cluster (pod name can be any name and image is by default from docker registry)

```
kubectl run <pod_name> --image=<image_name>
```

#check the status of pod with ip and node where it is running

```
kubectl get pods -o wide
```

#to verify the details about the pod

```
kubectl describe pod <pod_name>
```

Delete a pod

```
#delete the pod

kubectl delete pod <pod_name>
```

Creating Pod(YAML)

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: app2
  labels:
    app: web
    dept: devops
spec:
  containers:
   - name: demo-pod
     image: httpd
```

#create pod from yaml configuration

```
kubectl apply -f pod_definition.yml
```

Replication Controller

It ensures that a specified number of pod replicas are running at any one time for high availability (it keeps the desired pods)

Replication controller can spans across multiple nodes

ReplicationController | Kubernetes

Creating Replication Controller(YAML)

```yaml
apiVersion: v1
kind: ReplicationController
metadata:
  name: demo-rc
  labels:
   app: demo-rc
spec:
  replicas: 3
  selector:
    app: web
    dept: devops
  template:
```

```yaml
   metadata:
    name: app3
    labels:
      app: web
      dept: devops
   spec:
    containers:
      - name: demo-pod
        image: httpd
```

```
#create replication controller with pod

kubectl apply -f replication_controller.yml

#verify the replication controller

kubectl get rc

#verify the status of replication controller

kubectl describe rc/demo-rc

#delete the replication controller

kubectl delete rc demo-rc
```

Replication Set

Similar to replication controller

ReplicaSet | Kubernetes

Creating Replica Set(YAML)

```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
 name: demo-rs
 labels:
  app: demo-rs
spec:
 replicas: 3
 selector:
  matchLabels:
   app: web
   dept: devops
 template:
```

```yaml
  metadata:
   name: app4
   labels:
    app: web
    dept: devops
  spec:
   containers:
    - name: demo-pod
      image: httpd
```

```
#create replica set with pod

kubectl apply -f replica_set.yml

#verify the replica set

kubectl get rs

#verify the status of replica set

kubectl describe rs/demo-rs

#delete the replica set

kubectl delete rs demo-rs
```

Scaling the Pods

#scale the number of pods by modifying the yaml

```
replicas: 5
```

```
kubectl replace -f replica_set.yml
```

#scale out the number of pods

```
kubectl scale --replicas=6 -f replica_set.yml
```

or

```
kubectl scale --replicas=2 rs demo-rs
```

Deployment

Deployment allows to

- Run pods with replica set
- Upgrade versions of pod(Rolling update)
- Rollback to previous version if needed
- Pause and resume updates

Deployments | Kubernetes

Deployment

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: demo-deploy
  labels:
    app: web
spec:
  replicas: 4
  selector:
   matchLabels:
     app: web
  template:
```

```
  metadata:
    name: app5
    labels:
      app: web
      dept: devops
  spec:
    containers:
      - name: demo-pod
        image: httpd
```

```
#create deployment

kubectl apply -f deployment.yml

#verify the deployment

kubectl get deployment

#verify the status of deployment

kubectl describe deployment/demo-deploy

#delete the deployment

kubectl delete deployment demo-deploy
```

Two Deployment Strategy

- Recreate

- Rolling Update

Note: Rolling update is the default deployment strategy

#create the deployment with a specific version to verify rollout update

```
kubectl apply -f deployment.yml --record
```

#check the status of rollout

```
kubectl rollout status deployment/demo-deploy
```

#verify the history of rollout

```
kubectl rollout history deployment/demo-deploy
```

Upgrading the Deployment

#change the image version by modifying the yaml

```
image: httpd:bulleye
```

```
kubectl edit deployment/demo-deploy --record
```

or

#change the image version

```
kubectl set image deployment/demo-deploy demo-pod=httpd:latest
```

Rollout to older revision

```
#rollout to older revision
kubectl rollout undo deployment/demo-deploy
```

or

```
kubectl rollout undo deployment/demo-deploy --to-revision=2
```

Services

It allows users to connect with kubernetes pods and kubernetes pods to connect with other applications

Creating Service

#create a pod and then a service from yaml configuration

```
kubectl apply -f pod_definition.yml
```

```yaml
apiVersion: v1
kind: Service
metadata:
  name: demo-service
spec:
  type: NodePort
  ports:
    - targetPort: 80
      port: 80
      nodePort: 30001
  selector:
      app: web
      dept: devops
```

```
kubectl create -f service_definition.yml
```