# Kubernetes Lab - March 19, 2025

# Lab 1: Scaling Applications and Updating Applications

# Step 1: Create the nginx deployment

kubectl create deployment nginx --image=nginx:latest

# Step 2: Scale the deployment to 3 replicas

kubectl scale deployment nginx --replicas=3

# Step 3: Verify the pod status

kubectl get pods

# Step 4: Update the nginx deployment's image to version 1.21.0

kubectl set image deployment/nginx nginx=nginx:1.21.0

# Step 5: Check the rollout status

kubectl rollout status deployment/nginx

# Step 6: Get the pods with additional information

kubectl get pods -o wide

# Step 7: Undo the deployment rollout

kubectl rollout undo deployment/nginx

# Step 8: Check the rollout status after the undo

kubectl rollout status deployment/nginx

# -----------------------------------------------------------------------------

# Lab 2: ConfigMaps and Secrets in Kubernetes

# Step 1: Create a ConfigMap

kubectl create configmap my-config --from-literal=app\_name=MyK8sApp --from-literal=version=1.0

# Step 2: Get ConfigMap in YAML format

kubectl get configmap my-config -o yaml

# Step 3: Create a Secret

kubectl create secret generic my-secret --from-literal=username=admin --from-literal=password=SuperSecret123

# Step 4: Get Secret in YAML format

kubectl get secret my-secret -o yaml

# Step 5: List secrets

kubectl get secret

# Step 6: List ConfigMaps

kubectl get configmap

# Step 7: Delete ConfigMap

kubectl delete configmap my-config

# Step 8: Delete Secret

kubectl delete secret my-secret

# -----------------------------------------------------------------------------

# Lab 3: Persistent Volumes and Persistent Volume Claims (PVCs)

# Step 1: Create PersistentVolume YAML (pv.yaml)

nano pv.yaml

apiVersion: v1

kind: PersistentVolume

metadata:

name: my-pv

spec:

capacity:

storage: 1Gi # Defines total storage capacity

accessModes:

- ReadWriteOnce # Single node can read/write

persistentVolumeReclaimPolicy: Retain # Retains data even if PVC is deleted

hostPath:

path: \"/mnt/data\" # Storage path on the host node

# Step 2: Apply PersistentVolume YAML

kubectl apply -f pv.yaml

# Step 3: Get PersistentVolume details

kubectl get pv

# Step 4: Describe PersistentVolume

kubectl describe pv my-pv

# Step 5: Create PersistentVolumeClaim YAML (pvc.yaml)

nano pvc.yaml

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

name: my-pvc

spec:

accessModes:

- ReadWriteOnce # PVC can be mounted by a single node

resources:

requests:

storage: 500Mi # Requests 500MiB from an available PV

# Step 6: Apply PersistentVolumeClaim YAML

kubectl apply -f pvc.yaml

# Step 7: Get PersistentVolumeClaim details

kubectl get pvc

# Step 8: Describe PersistentVolumeClaim

kubectl describe pvc my-pvc

# -----------------------------------------------------------------------------

# Lab 4: Deploying a Stateful Application

# Step 1: Create Stateful Pod YAML (pod.yaml)

nano pod.yaml

apiVersion: v1

kind: Pod

metadata:

name: my-app

spec:

containers:

- name: app-container

image: nginx

volumeMounts:

- mountPath: \"/usr/share/nginx/html\"

name: storage

volumes:

- name: storage

persistentVolumeClaim:

claimName: my-pvc # PVC should be already created

" > pod.yaml

# Step 2: Apply Pod YAML

kubectl apply -f pod.yaml

# Step 3: Get Pod details

kubectl get pods

# Step 4: Describe Pod

kubectl describe pod my-app

# -----------------------------------------------------------------------------

# Lab 5: Deploying a Multi-Container Pod

# Step 1: Create Multi-Container Pod YAML (multi-container-pod.yaml)

echo "

apiVersion: v1

kind: Pod

metadata:

name: multi-container-pod

spec:

containers:

- name: app

image: nginx # Main application container

- name: sidecar

image: busybox # Sidecar container

command: [\"sh\", \"-c\", \"while true; do echo 'Logging...'; sleep 5; done\"]

" > multi-container-pod.yaml

# Step 2: Apply Multi-Container Pod YAML

kubectl apply -f multi-container-pod.yaml

# Step 3: Get Pod details

kubectl get pods

# Step 4: Describe Pod

kubectl describe pod multi-container-pod

# -----------------------------------------------------------------------------

# Lab 6: Running a CronJob

# Step 1: Create CronJob YAML (cronjob.yaml)

nano cronjob.yaml

apiVersion: batch/v1

kind: CronJob

metadata:

name: periodic-job

spec:

schedule: "\*/5 \* \* \* \*" # Runs every 5 minutes

jobTemplate:

spec:

template:

spec:

containers:

- name: cronjob-container

image: busybox

command: ["sh", "-c", "date; echo 'Hello from CronJob'"]

restartPolicy: OnFailure

# Step 2: Apply CronJob YAML

kubectl apply -f cronjob.yaml

# Step 3: Get CronJob details

kubectl get cronjob

# Step 4: Describe CronJob

kubectl describe cronjob periodic-job

# -----------------------------------------------------------------------------

# Lab 7: Deploying Kubernetes DaemonSet

# Step 1: Create DaemonSet YAML (daemonset.yaml)

nano daemonset.yaml

apiVersion: apps/v1

kind: DaemonSet

metadata:

name: logging-agent

spec:

selector:

matchLabels:

name: logging-agent

template:

metadata:

labels:

name: logging-agent

spec:

containers:

- name: logger

image: busybox

command: [\"sh\", \"-c\", \"while true; do echo 'Logging...'; sleep 10; done\"]

# Step 2: Apply DaemonSet YAML

kubectl apply -f daemonset.yaml

# Step 3: Get DaemonSet details

kubectl get daemonset

# Step 4: Get pods in the DaemonSet

kubectl get pods -o wide

# Step 5: View DaemonSet logs (optional)

kubectl logs ds/logging-agent

lab 8

notepad job.yaml

apiVersion: batch/v1

kind: Job

metadata:

name: simple-job

spec:

template:

spec:

containers:

- name: job-container

image: busybox

command:

- "sh"

- "-c"

- "echo 'Hello, Kubernetes!'"

restartPolicy: Never

kubectl apply -f job.yaml

output

kubectl get jobs

kubectl get pods --selector=job-name=simple-job

kubectl logs <pod-name>(optional)

lab9

Simulating a Service Mesh with a Sidecar Container

nano sidecar-pod.yaml

apiVersion: v1

kind: Pod

metadata:

name: nginx-sidecar-pod

spec:

containers:

- name: nginx

image: nginx:latest

ports:

- containerPort: 80

- name: sidecar

image: busybox

command: ["/bin/sh", "-c", "while true; do echo 'Logging traffic...'; sleep 5; done"]

kubectl get pods

kubectl logs nginx-sidecar-pod -c sidecar