**Kubernetes (K8s)**: An open-source platform for automating deployment, scaling, and management of containerized applications.

|  |  |
| --- | --- |
| Component | Explanation |
| Node | A worker machine (physical or virtual) in the Kubernetes cluster. |
| Pod | Smallest deployable unit, containing one or more containers sharing storage/network. |
| Service | Exposes a set of Pods as a network service (e.g., internal or external access). |
| Deployment | Manages Pods and ReplicaSets declaratively, ensuring desired state. |
| ConfigMap | Stores non-confidential configuration data as key-value pairs for apps. |
| Namespace | Logical partitioning of cluster resources for isolation (e.g., dev, prod). |

**Cluster Information**

|  |  |  |
| --- | --- | --- |
| Command | Example | Explanation |
| kubectl cluster-info | kubectl cluster-info | Shows cluster control plane and add-on details. |
| kubectl version | kubectl version | Displays client and server Kubernetes versions. |
| kubectl config view | kubectl config view | Views current kubeconfig (context, cluster, user). |
| kubectl get nodes | kubectl get nodes | Lists all nodes in the cluster. |
| kubectl describe node <node-name> | kubectl describe node node-1 | Detailed node info (e.g., conditions, resources). |

**Namespace Operations**

|  |  |  |
| --- | --- | --- |
| Command | Example | Explanation |
| kubectl get namespaces | kubectl get namespaces | Lists all namespaces (e.g., default). |
| kubectl create namespace <name> | kubectl create namespace dev | Creates a new namespace for resource isolation. |
| kubectl config set-context --current --namespace=<namespace> | kubectl config set-context --current --namespace=dev | Sets default namespace. |

**Application Deployment**

|  |  |  |
| --- | --- | --- |
| Command | Example | Explanation |
| kubectl apply -f <filename.yaml> | kubectl apply -f nginx-app.yml | Applies resources defined in a YAML file. |
| kubectl apply -f <directory> | kubectl apply -f ./configs | Applies all YAML files in a directory. |
| kubectl create deployment <name> --image=<image> | kubectl create deployment nginx-app --image=nginx | Creates a Deployment. |
| kubectl run <pod-name> --image=<image> | kubectl run my-pod --image=nginx | Creates a single Pod (not for production). |

**Viewing Resources**

|  |  |  |
| --- | --- | --- |
| Command | Example | Explanation |
| kubectl get pods | kubectl get pods | Lists Pods in the current namespace. |
| kubectl get pods -o wide | kubectl get pods -o wide | Adds details like node and IP. |
| kubectl get pods --all-namespaces | kubectl get pods --all-namespaces | Lists Pods across all namespaces. |
| kubectl get deployments | kubectl get deployments | Lists Deployments. |
| kubectl get services | kubectl get services | Lists Services. |
| kubectl get all | kubectl get all | Lists Pods, Deployments, Services, etc. |

**Describing Resources**

|  |  |  |
| --- | --- | --- |
| Command | Example | Explanation |
| kubectl describe pod <pod-name> | kubectl describe pod my-pod | Shows Pod details (events, status). |
| kubectl describe deployment <deployment-name> | kubectl describe deployment nginx-app | Deployment details (replicas, strategy). |
| kubectl describe service <service-name> | kubectl describe service nginx-service | Service details (endpoints, ports). |

**Pod Management**

|  |  |  |
| --- | --- | --- |
| Command | Example | Explanation |
| kubectl logs <pod-name> | kubectl logs my-pod | Views logs of the default container. |
| kubectl logs <pod-name> -c <container-name> | kubectl logs my-pod -c nginx | Logs from a specific container. |
| kubectl logs -f <pod-name> | kubectl logs -f my-pod | Streams logs in real-time (like tail -f). |
| kubectl exec <pod-name> -- <command> | kubectl exec my-pod -- ls | Runs a command inside a Pod’s container. |
| kubectl exec -it <pod-name> -- /bin/bash | kubectl exec -it my-pod -- /bin/bash | Opens an interactive shell. |
| kubectl delete pod <pod-name> | kubectl delete pod my-pod | Deletes a specific Pod. |

**Exposing Applications**

|  |  |  |
| --- | --- | --- |
| Command | Example | Explanation |
| kubectl expose deployment <name> --port=<port> --type=<type> | kubectl expose deployment nginx-app --port=80 --type=NodePort | Creates a Service. |
| kubectl port-forward <pod-name> <local-port>:<pod-port> | kubectl port-forward my-pod 8080:80 | Forwards local port to Pod port. |
| minikube service <service-name> --url | minikube service nginx-service --url | Returns the Minikube Service URL. |

**Service Types**

|  |  |
| --- | --- |
| Type | Explanation |
| ClusterIP | Default; exposes Service internally within the cluster. |
| NodePort | Exposes Service on each Node’s IP at a static port (30000-32767). |
| LoadBalancer | Exposes Service externally via a cloud provider’s load balancer. |
| ExternalName | Maps a Service to an external DNS name without proxying. |

**Scaling Applications**

|  |  |  |
| --- | --- | --- |
| Command | Example | Explanation |
| kubectl scale deployment <name> --replicas=<count> | kubectl scale deployment nginx-app --replicas=3 | Scales Pods to <count>. |
| kubectl autoscale deployment <name> --min=<min> --max=<max> --cpu-percent=<percent> | kubectl autoscale deployment nginx-app --min=2 --max=5 --cpu-percent=80 | Autoscales based on CPU. |
| kubectl get hpa | kubectl get hpa | Lists Horizontal Pod Autoscalers (HPA). |

**Update Commands**

|  |  |  |
| --- | --- | --- |
| Command | Example | Explanation |
| kubectl set image deployment/<name> <container>=<new-image> | kubectl set image deployment/nginx-app nginx=nginx:1.19 | Updates container image. |
| kubectl edit deployment <name> | kubectl edit deployment nginx-app | Opens Deployment in an editor for live edits. |
| kubectl rollout status deployment/<name> | kubectl rollout status deployment/nginx-app | Monitors rollout progress. |
| kubectl rollout history deployment/<name> | kubectl rollout history deployment/nginx-app | Shows revision history. |
| kubectl rollout undo deployment/<name> | kubectl rollout undo deployment/nginx-app | Rolls back to previous version. |
| kubectl rollout undo deployment/<name> --to-revision=<revision> | kubectl rollout undo deployment/nginx-app --to-revision=2 | Rolls back to specific revision. |

**ConfigMap Operations**

|  |  |  |
| --- | --- | --- |
| Command | Example | Explanation |
| kubectl create configmap <name> --from-literal=key1=value1 | kubectl create configmap my-config --from-literal=db=prod | Creates from key-value pairs. |
| kubectl create configmap <name> --from-file=<path/to/file> | kubectl create configmap my-config --from-file=config.txt | Creates from a file. |
| kubectl create configmap <name> --from-env-file=<path/to/file.env> | kubectl create configmap my-config --from-env-file=settings.env | Creates from .env. |
| kubectl get configmaps | kubectl get configmaps | Lists all ConfigMaps. |
| kubectl describe configmap <name> | kubectl describe configmap my-config | Shows ConfigMap details. |

**ConfigMap YAML Structure**

apiVersion: v1

kind: ConfigMap

metadata:

name: example-configmap

namespace: default

data:

database\_host: "mysql"

database\_port: "3306"

game.properties: |

enemy.types=aliens,monsters

player.maximum-lives=5

**Example**: kubectl apply -f example-configmap.yml

**Using ConfigMaps in Pods**

**As Environment Variables**

apiVersion: v1

kind: Pod

metadata:

name: configmap-env-pod

spec:

containers:

- name: test-container

image: nginx

env:

- name: DATABASE\_HOST

valueFrom:

configMapKeyRef:

name: example-configmap

key: database\_host

envFrom:

- configMapRef:

name: example-configmap

**Example**: kubectl apply -f configmap-env-pod.yml

**As Volume Mounts**

apiVersion: v1

kind: Pod

metadata:

name: configmap-vol-pod

spec:

containers:

- name: test-container

image: nginx

volumeMounts:

- name: config-volume

mountPath: /etc/config

volumes:

- name: config-volume

configMap:

name: example-configmap

items:

- key: game.properties

path: game.properties

apiVersion: apps/v1

kind: Deployment

metadata:

  name: adservice  # Name of the deployment

  labels:

    app: adservice  # Label to identify the app

spec:

  selector:

    matchLabels:

      app: adservice  # Ensures pods with this label are selected

  template:

    metadata:

      labels:

        app: adservice  # Labels for pod selection

    spec:

      serviceAccountName: adservice  # Service account for the pod

      terminationGracePeriodSeconds: 5  # Grace period before forcefully terminating pods

      securityContext:

        fsGroup: 1000  # Filesystem group ID

        runAsGroup: 1000  # Run the container with this group ID

        runAsNonRoot: true  # Ensures the container does not run as root

        runAsUser: 1000  # Runs the container with this user ID

      containers:

      - name: server  # Container name

        securityContext:

          allowPrivilegeEscalation: false  # Prevent privilege escalation

          capabilities:

            drop:

              - ALL  # Drop all Linux capabilities

          privileged: false  # Prevent privileged mode

          readOnlyRootFilesystem: true  # Ensures the root filesystem is read-only

        image: adservice  # Container image to use

        ports:

        - containerPort: 9555  # Expose port 9555

        env:

        - name: PORT

          value: "9555"  # Set the PORT environment variable

        resources:

          requests:

            cpu: 200m  # Minimum CPU required

            memory: 180Mi  # Minimum memory required

          limits:

            cpu: 300m  # Maximum CPU allowed

            memory: 300Mi  # Maximum memory allowed

        readinessProbe:

          initialDelaySeconds: 20  # Wait before checking readiness

          periodSeconds: 15  # Check every 15 seconds

          grpc:

            port: 9555  # Check readiness on gRPC port 9555

        livenessProbe:

          initialDelaySeconds: 20  # Wait before checking liveness

          periodSeconds: 15  # Check every 15 seconds

          grpc:

            port: 9555  # Check liveness on gRPC port 9555

---

apiVersion: v1

kind: Service

metadata:

  name: adservice  # Name of the service

  labels:

    app: adservice  # Label to match the deployment

spec:

  type: ClusterIP  # Internal service type

  selector:

    app: adservice  # Select pods with this label

  ports:

  - name: grpc  # Name of the port

    port: 9555  # Service port

    targetPort: 9555  # Target port on the container

---

apiVersion: v1

kind: ServiceAccount

metadata:

  name: adservice  # Name of the service account

**Question 1-5 (First Set)**

1. To ensure the cluster's resources match the configuration in your YAML file:

kubectl apply -f webapp-deployment.yaml

1. To increase the number of replicas for the web app Deployment:

kubectl scale deployment web-app --replicas=5

1. To investigate pods stuck in non-running state in the production namespace:

kubectl describe pods -n production

1. To create a ConfigMap named app-config from a config.properties file:

kubectl create configmap app-config --from-file=config.properties

1. To examine the detailed status of a crashing pod:

kubectl describe pod app-12345 -n production

**Question 1-5 (Second Set)**

1. To make the web-app Deployment accessible externally on port 80:

kubectl expose deployment web-app --port=80 --type=LoadBalancer

1. To apply changes from an updated ConfigMap YAML file:

kubectl apply -f app-config.yaml

1. To inspect the detailed status of the web-app Deployment:

kubectl describe deployment web-app

1. To list all externally exposed Services in the default namespace:

kubectl get services --field-selector spec.type=LoadBalancer

1. To temporarily stop the web-app Deployment and then restart it with 3 replicas:

kubectl scale deployment web-app --replicas=0

kubectl scale deployment web-app --replicas=3