DAY4—TASK--KUBERNETES

Kubernetes, often abbreviated as K8s, is a powerful open-source platform for automating the deployment, scaling, and management of containerized applications. Developed originally by Google, Kubernetes is now maintained by the Cloud Native Computing Foundation (CNCF). Below is a comprehensive guide covering the key concepts, architecture, and best practices for Kubernetes.

****1. Introduction to Kubernetes****

****What is Kubernetes?****

Kubernetes is a container orchestration platform that automates the deployment, scaling, and operations of application containers across clusters of machines. It abstracts the underlying infrastructure and provides a unified API for managing containerized applications.

****1.1 Key Features of Kubernetes****

- **Automated Deployment and Scaling:** Automatically deploy, manage, and scale applications.
- **Self-Healing:** Automatically replaces failed containers and restarts applications.
- **Load Balancing:** Distributes traffic across containers.
- **Storage Orchestration:** Manages storage resources for applications.
- **Service Discovery:** Provides DNS names for services and load balances across them.
- **Configuration Management:** Manages configuration and secrets for applications.
- **Rolling Updates:** Performs rolling updates to applications with zero downtime.

1.2 Basic Terminology

Term	**Description**
Cluster	A set of nodes (physical or virtual machines) running Kubernetes components.
Node	A machine (VM or physical) that runs Kubernetes components like the Kubelet and Docker.
Pod	The smallest deployable unit that can contain one or more containers.
Deployment	A higher-level abstraction for managing and scaling a set of Pods.
Service	Defines how to access Pods, providing load balancing and service discovery.
Namespace	Virtual clusters within a Kubernetes cluster to organize resources.
ReplicaSet	Ensures a specified number of pod replicas are running at any given time.
StatefulSet	Manages stateful applications, maintaining unique identities for Pods.
DaemonSet	Ensures a copy of a Pod runs on all (or some) Nodes.
Job	Manages the execution of one-time tasks and ensures completion.
CronJob	Creates Jobs on a scheduled time basis.
ConfigMap	Manages configuration data for applications.
Secret	Stores sensitive data, such as passwords or tokens.

| **Ingress** | Manages external access to services, typically via HTTP/HTTPS. |

| **PersistentVolume (PV)** | A piece of storage in the cluster provisioned by an administrator. |

| **PersistentVolumeClaim (PVC)** | A request for storage by a user. |

| **Helm** | A package manager for Kubernetes applications. |

---

2. Kubernetes Architecture

2.1 Kubernetes Components

| **Component** | **Description** |

|---|---|

| **Master Node** | Manages the Kubernetes cluster. Contains components like API server, controller manager, and scheduler. |

| **Kubelet** | An agent that runs on each worker node, ensuring that containers are running in Pods. |

| **Kube-Proxy** | Maintains network rules for Pod communication and load balancing. |

| **API Server** | The entry point for all REST commands used to control the cluster. |

| **Controller Manager** | Ensures that the desired state of the cluster is maintained. |

| **Scheduler** | Assigns Pods to Nodes based on resource availability and constraints. |

| **etcd** | A distributed key-value store used for storing all Kubernetes cluster data. |

2.2 Kubernetes Object Lifecycle

Lifecycle Stage	**Description**
Creation	Define objects using YAML or JSON manifests and apply them using `kubectl apply`.
Update	Modify the configuration and apply changes using `kubectl apply`.
Scaling	Adjust the number of Pods using `kubectl scale`.
Deletion	Remove objects using `kubectl delete`.

Diagram of Kubernetes Architecture:

(https://www.redhat.com/cms/managed-files/k8s-architecture-2020-09-17.jpg)

Source: Red Hat

3. Core Kubernetes Concepts

3.1 Pods

- **What is a Pod?**

- A Pod is the smallest and simplest Kubernetes object. A Pod encapsulates one or more containers.
- **Example YAML for a Pod:**

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: my-pod
spec:
  containers:
  - name: my-container
    image: nginx:latest
    ports:
    - containerPort: 80
```

3.2 Deployments

- **What is a Deployment?**
- A higher-level abstraction for managing a set of Pods. It ensures the desired state is maintained.
- **Example YAML for a Deployment:**

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
```

```
  name: my-deployment
spec:
  replicas: 3
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
      labels:
        app: my-app
    spec:
      containers:
      - name: my-container
        image: nginx:latest
        ports:
        - containerPort: 80
```

3.3 Services

- **What is a Service?**
  - Provides a stable IP address and DNS name for a set of Pods.
  - **Example YAML for a Service:**

```yaml
apiVersion: v1
kind: Service
metadata:
```

```
  name: my-service
spec:
  selector:
    app: my-app
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
```

3.4 ConfigMaps and Secrets

- **ConfigMaps**: Manage configuration data for applications.
- **Secrets**: Store sensitive data such as passwords and tokens.
- **Example YAML for a ConfigMap:**

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: my-config
data:
  key1: value1
  key2: value2
```

- **Example YAML for a Secret:**

```yaml
apiVersion: v1
kind: Secret
metadata:
  name: my-secret
type: Opaque
data:
  username: dXNlcg==  # Base64 encoded username
  password: cGFzc3dvcmQ=  # Base64 encoded password
```

3.5 Persistent Storage

- **PersistentVolume (PV)**: A storage resource in the cluster.
- **PersistentVolumeClaim (PVC)**: A request for storage.
- **Example YAML for a PV:**

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: my-pv
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
```

```
    hostPath:
      path: "/mnt/data"
```

- **Example YAML for a PVC:**

```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-pvc
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
```

3.6 Ingress

- **What is Ingress?**
  - Manages external access to services, typically HTTP/HTTPS.
  - **Example YAML for an Ingress:**

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
```

```
  metadata:
    name: my-ingress
  spec:
    rules:
    - host: myapp.example.com
      http:
        paths:
        - path: /
          pathType: Prefix
          backend:
            service:
              name: my-service
              port:
                number: 80
```

3.7 Helm Charts

- **What is Helm?**
  - A package manager for Kubernetes, similar to apt for Debian-based systems or yum for Red Hat-based systems.
- **Basic Commands:**
  - **Install a Chart:**
    ```bash
    helm install my-release stable/nginx
    ```
  - **Upgrade a Release:**
    ```bash
    ```

```
helm upgrade my-release stable/nginx
```

- **Uninstall a Release:**
```bash
helm uninstall my-release
```

Helm Chart Example:

```yaml
apiVersion: v2
name: mychart
description: A Helm chart for Kubernetes
version: 0.1.0

dependencies:
  - name: nginx
    version: 1.16.0
    repository: https://charts.bitnami.com/bitnami

templates:
  - name: deployment.yaml
    apiVersion: apps/v1
    kind: Deployment
    spec:
      replicas: 3
      selector:
        matchLabels:
```

```
    app: my-app
template:
  metadata:
    labels:
      app: my-app
  spec:
    containers:
```