Kubernetes concepts and architecture

1. Introduction

Kubernetes is an open-source platform designed for automating the deployment, scaling, and management of containerized applications. It provides a robust framework for managing containerized workloads and services across a cluster of machines.

2. Kubernetes Architecture

Components

Kubernetes architecture is comprised of several key components that work together to manage containerized applications:

- **Control Plane:** Manages the overall state of the cluster, including scheduling and scaling of applications. It consists of various components:
  - **API Server:** The central component that exposes the Kubernetes API.
  - **Controller Manager:** Ensures that the desired state of the cluster is maintained.
  - **Scheduler:** Assigns work (Pods) to nodes in the cluster.
  - **etcd:** A distributed key-value store that holds the cluster state and configuration.
- **Node Components:** These run on each node and manage the running of containers:
  - **Kubelet:** An agent that ensures containers are running as expected.
  - **Kube-Proxy:** Manages network communication both inside and outside the cluster.
  - **Container Runtime:** Software responsible for running containers (e.g., Docker).

Control Plane

The control plane orchestrates and maintains the desired state of the cluster. It consists of:

- **API Server:** Exposes the Kubernetes API and is the entry point for all REST commands.
- **Controller Manager:** Handles controllers that manage the state of the cluster.
- **Scheduler:** Assigns workloads (Pods) to nodes based on resource availability.
- **etcd:** Stores the state and configuration of the cluster.

Node Components

Each node in the cluster runs:

- **Kubelet:** Ensures containers are running and healthy according to the specifications.
- **Kube-Proxy:** Handles networking and load balancing within the cluster.
- **Container Runtime:** Executes the containers (e.g., Docker, containerd).

3. Core Kubernetes Concepts

Pods

A Pod is the smallest deployable unit in Kubernetes, consisting of one or more containers that share the same network namespace and storage. Pods can be managed independently or as part of a larger deployment.

Deployments

A Deployment is a higher-level concept that manages the lifecycle of Pods, ensuring a specified number of replicas are running at all times. Deployments facilitate rolling updates and rollback functionality.

Services

Services expose a set of Pods as a network service. They provide a stable endpoint for accessing the Pods, load balancing traffic, and service discovery.

ReplicaSets

A ReplicaSet ensures that a specified number of Pod replicas are running at any given time. It is used by Deployments to manage the desired state of Pods.

4. Advanced Concepts

Namespaces

Namespaces are used to organize and manage resources within a Kubernetes cluster. They provide a way to divide cluster resources between multiple users or teams.

ConfigMaps and Secrets

- **ConfigMaps:** Allow you to manage configuration data for your applications in a key-value format.
- **Secrets:** Used to manage sensitive data, such as passwords and API tokens, securely.

Persistent Volumes and Persistent Volume Claims

- **Persistent Volumes (PV):** Abstract the storage resource in the cluster, allowing applications to use storage independently of the lifecycle of a Pod.
- **Persistent Volume Claims (PVC):** Requests storage resources by users or applications.

Horizontal Pod Autoscaler (HPA)

HPA automatically scales the number of Pods in a Deployment or ReplicaSet based on observed CPU utilization or other select metrics.

5. Load Balancing in Kubernetes

Kubernetes provides several mechanisms for load balancing:

- **Service Load Balancing:** Distributes traffic across Pods within a Service.
- **Ingress Controllers:** Manage external access to services and provide advanced routing, SSL termination, and load balancing.

Commands Reference:

- **Get Pods:** `kubectl get pods`
- **Get Deployments:** `kubectl get deployments`
- **Get Services:** `kubectl get services`
- **Get HPA:** `kubectl get hpa`
- **Create Deployment:** `kubectl create deployment <deployment-name> --image=<image-name>`
- **Apply Configuration:** `kubectl apply -f <file-name>.yaml`
- **Delete Deployment:** `kubectl delete deployment <deployment-name>`