Rajat Chauhan

Kubernetes Architecture

Kubernetes Architecture

Rajat Chauhan

What is Kubernetes?

Kubernetes, often abbreviated as K8s, is an open-source platform designed for automating the deployment, scaling, and management of containerized applications. It was originally developed by Google and released as open source in 2014. The name "Kubernetes" comes from the Greek word for "helmsman" or "pilot," which symbolizes its role in guiding and managing containerized workloads. The abbreviation "K8s" is derived by replacing the eight letters between "K" and "s" with the digit "8."

Benefits of Using Kubernetes

Kubernetes offers a wide range of benefits for organizations looking to manage containerized applications at scale:

1. **Automated Container Orchestration**: Kubernetes automatically manages the deployment, scaling, and operation of application containers, ensuring that the desired state of the application is maintained.

2. **Scalability**: Kubernetes can easily scale applications up or down based on demand, ensuring optimal resource utilization.

3. **High Availability**: Kubernetes ensures that applications are always running and accessible, even in the event of node failures, through automated load balancing and failover mechanisms.

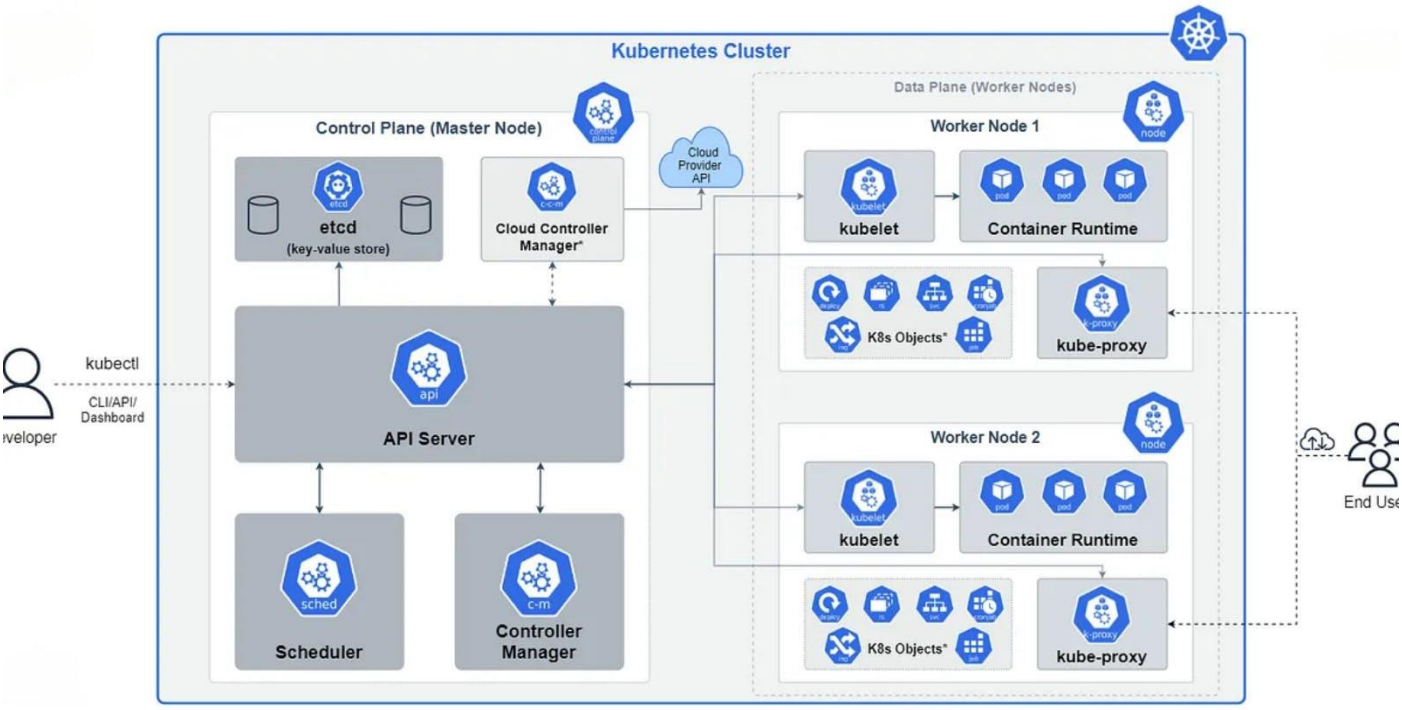
4. **Portability**: Kubernetes abstracts the underlying infrastructure, allowing applications to run consistently across different environments, including on-premises, cloud, or hybrid setups.

5. **Self-Healing**: Kubernetes automatically restarts failed containers, replaces containers, kills containers that don't respond to user-defined health checks, and ensures that only healthy containers are available to serve traffic.

6. **Declarative Configuration**: Kubernetes allows users to declare the desired state of the system using YAML or JSON configuration files, making it easier to manage complex deployments.

Kubernetes Architecture

Kubernetes architecture is built on a modular system that includes several key components, each playing a vital role in managing containerized applications. These components are distributed across the Control Plane and Worker Nodes.



Control Plane

The Control Plane is the brain of the Kubernetes cluster. It manages the overall state of the cluster and is responsible for making global decisions about the cluster, such as scheduling, scaling, and responding to cluster events.

- **API Server**: The API server is the entry point for all administrative tasks in the Kubernetes cluster. It exposes the Kubernetes API, which is used by the command-line tool, other components, and external users to interact with the cluster.
- **etcd**: etcd is a distributed key-value store that holds the cluster's configuration data, representing the overall state of the cluster. It serves as the backing store for all cluster data.
- **Controller Manager**: The controller manager is responsible for maintaining the desired state of the cluster by running various controllers that handle tasks such as node management, replication, and endpoints.
- **Scheduler**: The scheduler is responsible for assigning newly created pods to nodes in the cluster based on resource availability and other constraints.

Worker Nodes

Worker Nodes are the machines where the actual application containers run. Each node contains the following components:

- **kubelet**: The kubelet is an agent that runs on each worker node, ensuring that containers are running in a pod as expected. It communicates with the Control Plane to receive instructions and report back on the state of the node.
- **Container Runtime**: The container runtime, such as Docker or containerd, is responsible for pulling container images from a registry and running them on the node.
- **kube-proxy**: kube-proxy is a network proxy that maintains network rules on nodes, allowing communication between pods and services inside and outside the cluster.

Difference Between kubectl and Kubelets

- **kubectl**: kubectl is the command-line tool used to interact with the Kubernetes API server. It allows users to manage Kubernetes resources, deploy applications, and inspect cluster resources. Essentially, it acts as the user interface for communicating with the cluster.
- **kubelet**: The kubelet is a component that runs on each worker node in the Kubernetes cluster. It ensures that the containers specified in the PodSpec are running and healthy. The kubelet communicates with the API server to receive updates and report the status of the node.

The Role of the API Server

The API server is a critical component of the Kubernetes Control Plane, acting as the front-end for the Kubernetes control plane. It exposes the Kubernetes API, which is a RESTful interface used by all other components to communicate with each other and by external users to manage the cluster. The API server handles all the requests, validates them, and then processes them by updating the etcd store or dispatching the necessary tasks to other components, such as the scheduler or controller manager.