22AIE305
Introduction to Cloud Computing
2-0-3-3

Krishnapriya P S
Dept. of Computer Science & Engineering
Amrita Vishwa Vidyapeetham

AMRITA VISHWA VIDYAPEETHAM
DEEMED TO BE UNIVERSITY

Kubernets

Kubernetes (K8S)

- Kubernetes is an open-source platform that manages Docker containers in the form of a cluster.

What is Kubernetes (k8s)

- **Kubernetes** is an open-source Container Management tool.
- That automates container deployment, container scaling, descaling, and container load balancing (also called a container orchestration tool).
- It is written in Golang and has a vast community.
- It was first developed by Google and later donated to CNCF (Cloud Native Computing Foundation).
- Kubernetes can group ‘n’ number of containers into one logical unit for managing and deploying them easily. It works brilliantly with all cloud vendors i.e. public, hybrid, and on-premises.

Benefits of Using Kubernetes

1. Automated deployment and management

- If you are using Kubernetes for deploying the application then no need for manual intervention Kubernetes will take care of everything like automating the deployment, scaling, and containerizing the application.
- Kubernetes will reduce the errors that can be made by humans which makes the deployment more effective.

2. Scalability

- You can scale the application containers depending on the incoming traffic Kubernetes offers Horizontal pod scaling the pods will be scaled automatically depending on the load.

3. High availability

- You can achieve high availability for your application with the help of Kubernetes and it will reduce the latency issues for the end users.

Benefits of Using Kubernetes

4. Cost-effectiveness

- If there is unnecessary use of infrastructure the cost will also increase Kubernetes will help you to reduce resource utilization and control the overprovisioning of infrastructure.

5. Improved developer productivity

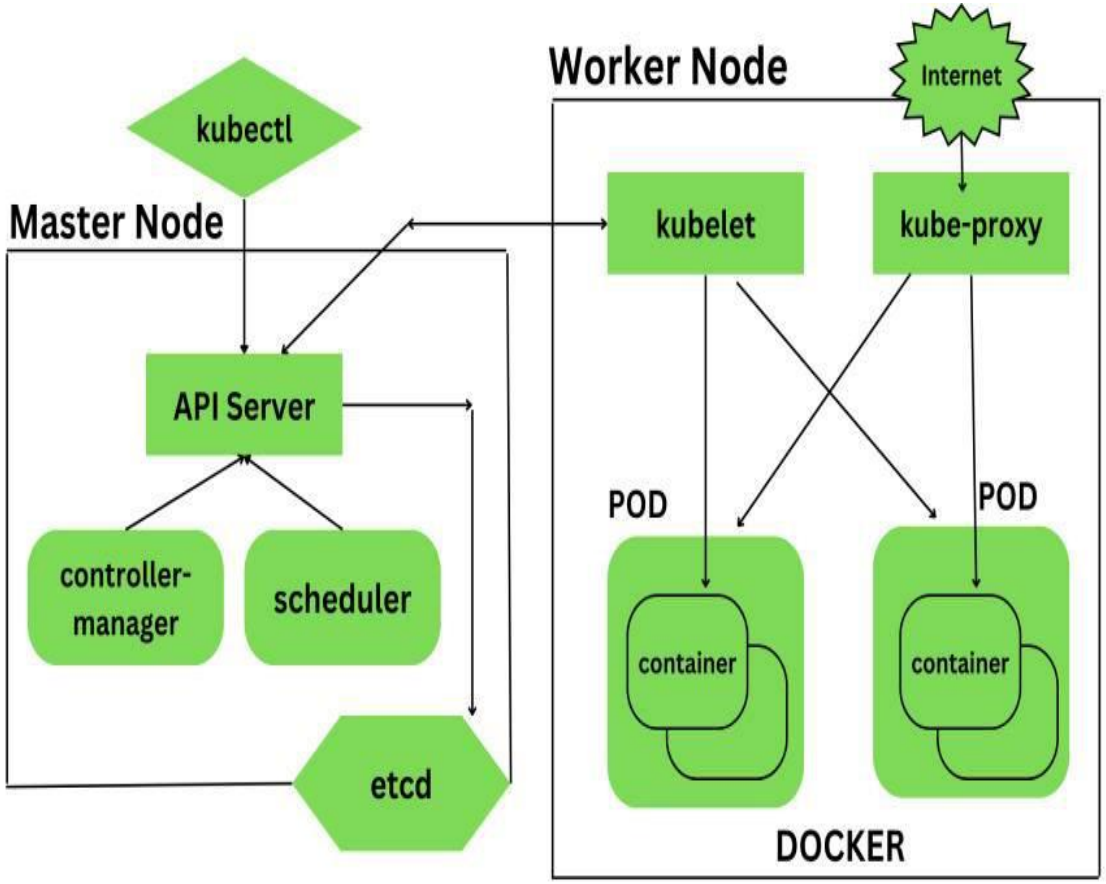
- Developers can concentrate more on the developing part Kubernetes will reduce the efforts of deploying the application.

Deploying and Managing Containerized Applications With Kubernetes

- **Step 1:** Install Kubernetes and setup the Kubernetes cluster there should be a minimum at least one master node and two worker nodes you can set up the Kubernetes cluster in any of the cloud which are providing the Kubernetes as a service.
- **Step 2:** Know create deployment manifest file you can create this manifests in the manifest you can specify the exact number of pods are required and what the container image and what types of resources are required after completion of writing the manifest file apply the file using kubectl command.
- **Step 3:** After creating the pods know you need to expose the service to the outside of the for that you need to write one more manifestfile which contains service type (e.g., LoadBalancer or ClusterIP), ports, and selectors.

Architecture of Kubernetes

- Kubernetes follows the client-server architecture.
- The master installed on one machine and the node on separate Linux machines.
- It follows the master-slave model, which uses a master to manage Docker containers across multiple Kubernetes nodes.
- A master and its controlled nodes(worker nodes) constitute a **“Kubernetes cluster”**.
- A developer can deploy an application in the docker containers with the assistance of the Kubernetes master.



Key components of Kubernetes

- **Kubernetes- Master Node Components**

Kubernetes master is responsible for managing the entire cluster, coordinates all activities inside the cluster, and communicates with the worker nodes to keep the Kubernetes and your application running.

The components of the Kubernetes Master node are:

- **API Server**

The API server is the entry point for all the REST commands used to control the cluster. All the administrative tasks are done by the API server within the master node. If we want to create, delete, update or display in Kubernetes object it has to go through this API server.

- **Scheduler**

It is a service in the master responsible for distributing the workload. It is responsible for tracking the utilization of the working load of each worker node and then placing the workload on which resources are available and can accept the workload. The scheduler is responsible for scheduling pods across available nodes.

Key components of Kubernetes

Controller Manager

Also known as controllers. It is a daemon that runs in a non terminating loop and is responsible for collecting and sending information to the API server.

Basically, the controller watches the desired state of the cluster if the current state of the cluster does not meet the desired state then the control loop takes the corrective steps to make sure that the current state is the same as that of the desired state. The key controllers are the replication controller, endpoint controller, namespace controller, and service account, controller.

Key components of Kubernetes

- **Kubernetes- Worker Node Components**

Kubelet

It is a primary node agent that communicates with the master node and executes on each worker node inside the cluster. It gets the pod specifications through the API server and executes the container associated with the pods.

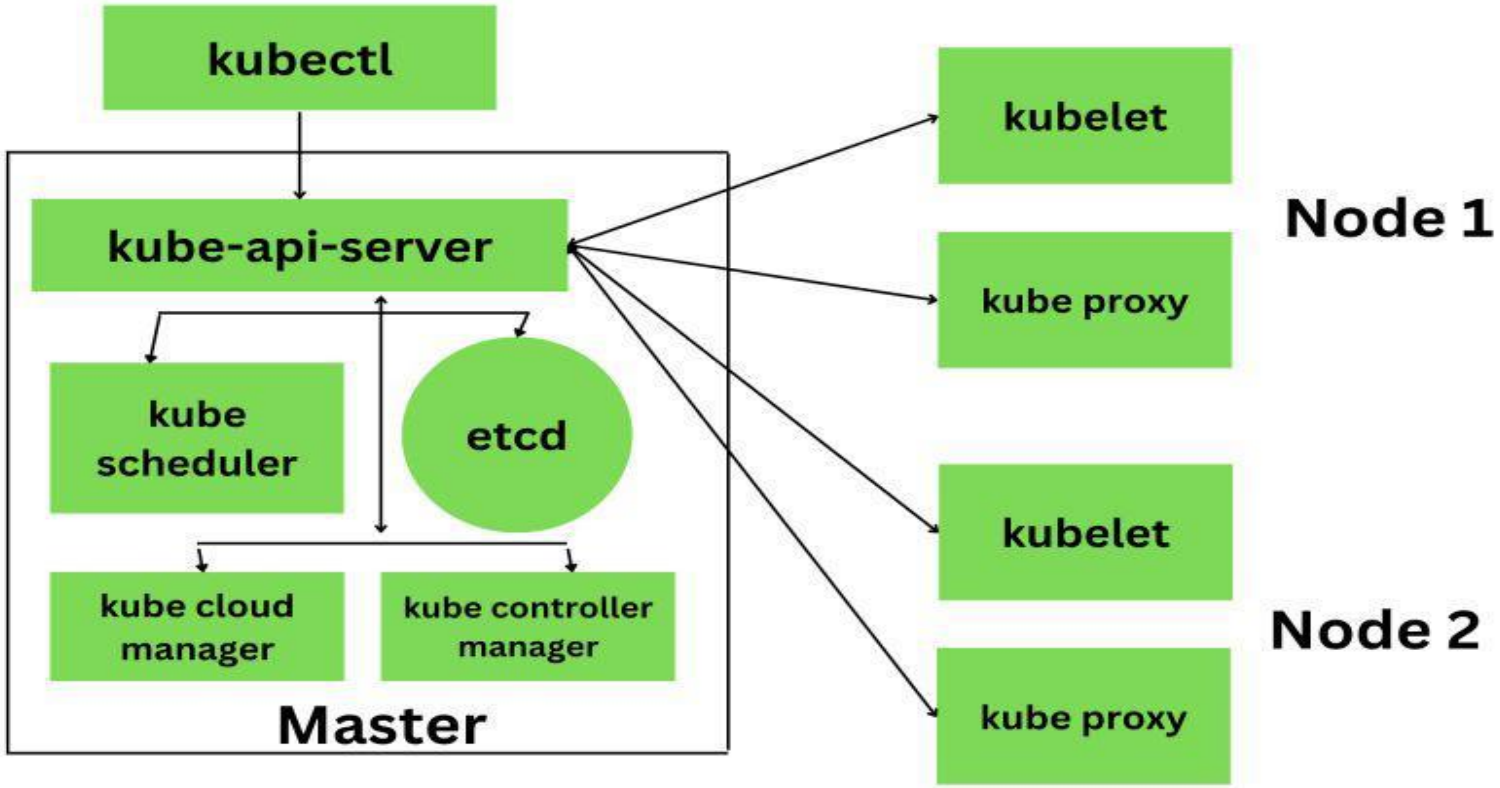
Kube-Proxy

It is the core networking component inside the Kubernetes cluster. It is responsible for maintaining the entire network configuration. Kube-Proxy maintains the distributed network across all the nodes, pods, and containers and exposes the services across the outside world.

Pods

A pod is a group of containers that are deployed together on the same host. With the help of pods, we can deploy multiple dependent containers together so it acts as a wrapper around these containers so we can interact and manage these containers primarily through pods.

Key components of Kubernetes



Key components of Kubernetes

Docker

- Docker is the containerization platform that is used to package your application and all its dependencies together.
- Docker is a tool designed to make it easier to create, deploy, and run applications by using containers.
- Docker is the world's leading software container platform.
- It was launched in 2013 by a company called Dot cloud.
- It is written in the Go language. It has been just six years since Docker was launched yet communities have already shifted to it from VMs.
- Docker is designed to benefit both developers and system administrators making it a part of many DevOps toolchains.

Namah Shivaya