Day 5 – Kubernetes Architecture

1. Explain Kubernetes Architecture in depth.

Ans) Kubernetes has a modular and distributed architecture, designed for scalability, resilience, and declarative automation. At a high level, Kubernetes is made up of a control plane (master components) and a data plane (worker nodes).

🧩 Components in Detail

🔹 A. Control Plane (Master Components)

a. kube-apiserver

- Acts as the gateway to the Kubernetes cluster.
- All interactions (via kubectl, other tools, or internal components) go through this.
- Exposes the Kubernetes API (REST) for clients/developers to interact.

b. etcd

- A distributed, consistent key-value store.
- Stores all cluster data: nodes, pods, secrets, config maps, etc.
- Highly available and fault-tolerant.

c. kube-scheduler

- Watches for newly created Pods without assigned nodes.
- Selects the best node to run a Pod based on Resource Availability, Affinity Rules, Taints and Tolerations, Node Selectors etc.

d. kube-controller-manager

- Runs controller loops to reconcile cluster state.
- Includes:

i) Node Controller – detects and reacts to node failures.
ii) Replication Controller – maintains desired number of pod replicas.
iii) Job Controller, Endpoint Controller, etc.

🔹 B. Node Components (Worker Nodes)

a. kubelet

- Agent that runs on each worker node.
- Ensures containers are running in a Pod.
- Communicates with the control plane.

b. kube-proxy

- Handles networking and traffic routing.
- Maintains network rules on nodes.
- Enables service discovery and load balancing.

c. Container Runtime

- The software that actually runs containers (e.g., containerd, CRI-O, or Docker)
- Interfaces with kubelet via the Container Runtime Interface (CRI).

d. Pod:

- The smallest deployable unit in Kubernetes.
- A Pod can hold one or more containers that share network namespace (same IP/ port space), Storage Volumes, Lifecycle (Start/Stop together)
- Pods are ephemeral; Kubernetes replaces them automatically if they fail or are evicted.

🔄 How It All Works Together (Flow)

a. User submits deployment via kubectl (talks to kube-apiserver).

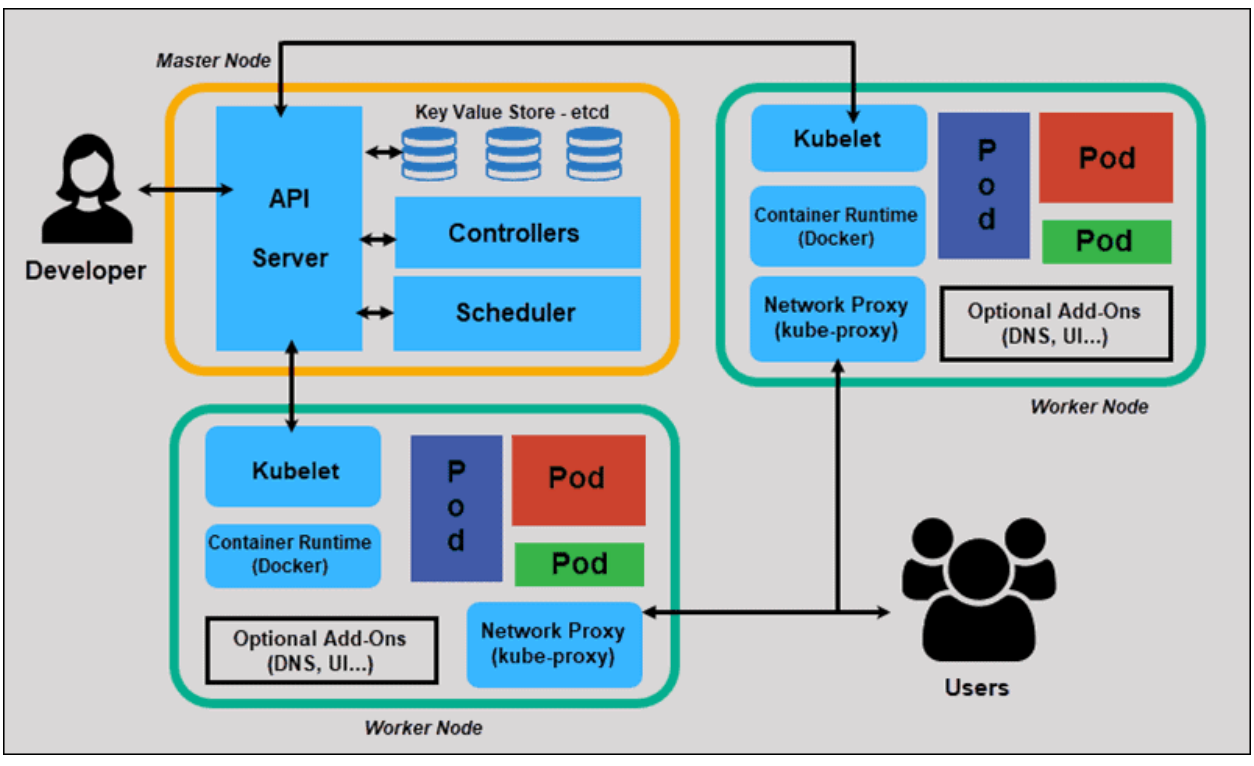
b. kube-apiserver validates and stores the object in etcd.

c. Controllers notice the desired state and take action (e.g., create Pods).

d. Scheduler assigns Pods to appropriate nodes.

e. kubelet on each node receives the Pod spec and launches containers.

f. kube-proxy sets up routing so services can reach the right Pods.



2. What are the different functionalities of kube-apiserver?

Ans) The kube-apiserver is the central management component and entry point for all operations in a Kubernetes cluster. It exposes the Kubernetes API, which is used by all components—internal (like controllers, schedulers) and external (like kubectl, CI/CD tools)—to communicate with and manage the cluster.

🧠 Core Functionalities of kube-apiserver

a. 🔐 Authentication

-> Validates the identity of users and components making requests.

-> Supports multiple authentication methods:

- Certificates
- Bearer Tokens
- Service Accounts
- OIDC (OAuth2)
- Webhooks

b. ✅ Authorization

-> Once authenticated, determines what actions the entity is allowed to perform.

-> Uses:

- Role-based Access Control (RBAC)
- Attribute-based Access Control (ABAC)
- Webhook Authorization
- Node Authorization

c. 🔍 Admission Control

-> Intercepts and validates or mutates requests after authentication and authorization.

-> Types of admission controllers:

- ValidatingAdmissionWebhook – Validate the request.
- MutatingAdmissionWebhook – Modify the object before saving.
- Examples: NamespaceLifecycle, LimitRanger, ResourceQuota

d. 🗄️ API Handling & Validation

-> Parses incoming requests (usually via REST).

-> Validates API objects against schemas before storing.

-> Converts between different API versions (e.g., apps/v1, v1beta1).

e. 📦 State Persistence (etcd interaction)

-> Stores and retrieves the desired state of the cluster from etcd.

-> Every object created/modified through the API (Pods, Deployments, Services) is stored here.

-> Acts as a source of truth for the cluster state.

f. 📡 Cluster Communication Hub

-> Serves as the hub for internal component communication:

- The scheduler fetches unscheduled Pods.
- The controller manager monitors objects and ensures desired state.
- kubelet watches for Pods assigned to its node.

g. 📩 RESTful Interface for Clients

-> Exposes REST APIs used by: kubectl CLI, Dashboard UI, CI/CD tools like ArgoCD, Gitlab CI, third party apps integrating with Kubernetes.

The kube-apiserver is like the brainstem of Kubernetes—every single cluster interaction passes through it.