Heading 2

1. Control Plane Components

kube-apiserver

The API server is a component of the Kubernetes control plane that exposes the Kubernetes API.

etcd

Consistent and highly-available key value store used as Kubernetes' backing store for all cluster data

kube-scheduler

Control plane component that watches for newly created Pods with no assigned node, and selects a node for them to run on.

kube-controller-manager

- Control Plane component that runs controller processes
- Logically, each controller is a separate process, but to reduce complexity, they are all compiled into a single binary and run in a single process.

These controllers include (4 Controler):(Node, Replication, Endpoint, Service Account and Token Controler)

1. Node controller: Responsible for noticing and responding when nodes go down.

2. Replication controller: Responsible for maintaining the correct number of pods for every replication controller object in the system.

3. Endpoints controller: Populates the Endpoints object (that is, joins Services & Pods).

4. Service Account & Token controllers: Create default accounts and API access tokens for new namespaces.

cloud-controller-manager

A Kubernetes control plane component that embeds cloud-specific control logic.

I. Node controller: For checking the cloud provider to determine if a node has been deleted in the cloud after it stops responding

II. Route controller: For setting up routes in the underlying cloud infrastructure

III. Service controller: For creating, updating and deleting cloud provider load balancer

2. Node Components

Note: Node components run on every node, maintaining running pods and providing the Kubernetes runtime environment.

1) kubelet

An agent that runs on each node in the cluster. It makes sure that containers are running in a Pod.

The kubelet takes a set of PodSpecs that are provided through various mechanisms and ensures that the containers described in those PodSpecs are running and healthy. The kubelet doesn't manage containers which were not created by Kubernetes.

2) kube-proxy

kube-proxy is a network proxy that runs on each node in your cluster, implementing part of the Kubernetes Service concept.
kube-proxy maintains network rules on nodes. These network rules allow network communication to your Pods from network sessions inside or outside of your cluster.

kube-proxy uses the operating system packet filtering layer if there is one and it's available. Otherwise, kube-proxy forwards the traffic itself

3) Container Runtime

The container runtime is the software that is responsible for running containers. Kubernetes supports several container runtimes: Docker, containerd, CRI-O, and any implementation of the Kubernetes CRI (Container Runtime Interface).

3. Addons

1) DNS

While the other addons are not strictly required, all Kubernetes clusters should have cluster DNS, as many examples rely on it.

Cluster DNS is a DNS server, in addition to the other DNS server(s) in your environment, which serves DNS records for Kubernetes services.

Containers started by Kubernetes automatically include this DNS server in their DNS searches.

2) Web UI (Dashboard)

Dashboard is a general purpose, web-based UI for Kubernetes clusters. It allows users to manage and troubleshoot applications running in the cluster, as well as the cluster itself.

3) Container Resource Monitoring

Container Resource Monitoring records generic time-series metrics about containers in a central database, and provides a UI for browsing that data.

4) Cluster-level Logging

A cluster-level logging mechanism is responsible for saving container logs to a central log store with search/browsing interface.