**Kubernetes Components**

Reference: <https://kubernetes.io/docs/concepts/overview/components/>

<https://kubernetes.io/docs/reference/glossary/?all=true#term-control-plane>

When you deploy Kubernetes, you get a cluster.

A Kubernetes cluster consists of a set of worker machines, called [node](https://kubernetes.io/docs/concepts/architecture/nodes/) , that run containerized applications. Every cluster has at least one worker node.

*Node* – A node is a worker machine in Kubernetes. Refer image at last page.

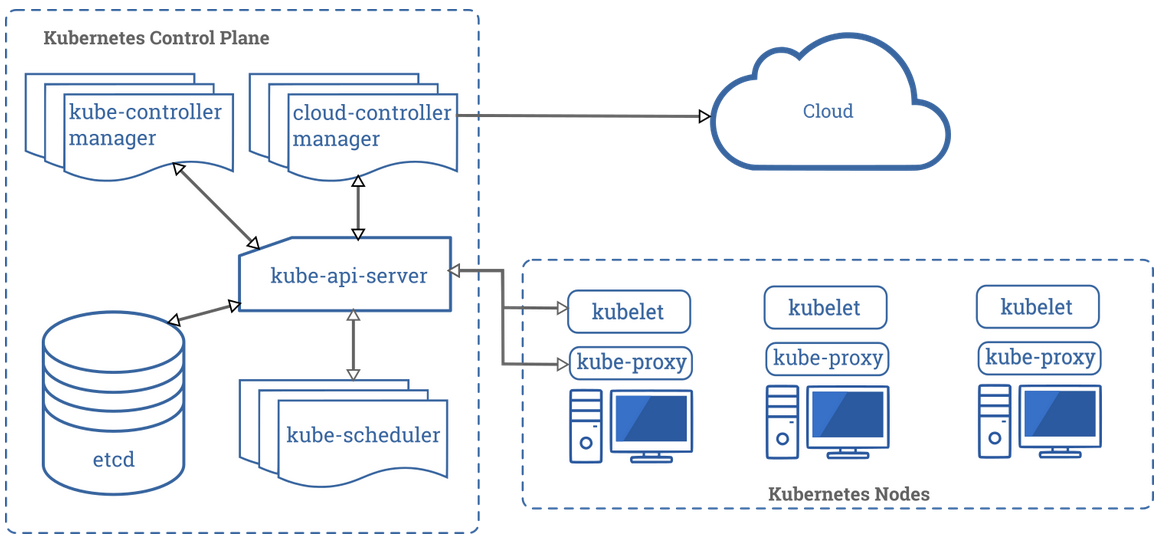
The worker node(s) host the [Pods](https://kubernetes.io/docs/concepts/workloads/pods/pod-overview/) that are the components of the application workload.

*Pod* – A Pod represents a set of running containers on your cluster. It is the smallest and simplest Kubernetes object. Refer image at last page.

The [control plane](https://kubernetes.io/docs/reference/glossary/?all=true#term-control-plane) manages the worker nodes and the Pods in the cluster.

*Control plane* - The container orchestration layer that exposes the API and interfaces to define, deploy, and manage the lifecycle of containers.

**Kubernetes Cluster with its components**



The components are of below types –

* Control Plane components
* Node components
* Addons
* **Control Plane components**

The Control Plane’s components make global decisions about the cluster (for example, scheduling), as well as detecting and responding to cluster events (for example, starting up a new pod when a deployment’s replicas field is unsatisfied).

* **Kube-api-server**

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

* **etcd**

Consistent and highly-available key value store used as Kubernetes’ backing store for all cluster data. All Kubernetes objects are stored on etcd. Periodically backing up the etcd cluster data is important to recover Kubernetes clusters under disaster scenarios, such as losing all master nodes.

References: <https://kubernetes.io/docs/tasks/administer-cluster/configure-upgrade-etcd/#backing-up-an-etcd-cluster>

<https://etcd.io/docs/v3.4.0/>

* **Kube-Scheduler**

Control plane component that watches for newly created [Pods](https://kubernetes.io/docs/concepts/workloads/pods/pod-overview/) with no assigned [node](https://kubernetes.io/docs/concepts/architecture/nodes/) , and selects a node for them to run on.

* **Kube-controller-manager**

Control Plane component that runs [controller](https://kubernetes.io/docs/concepts/architecture/controller/) processes. These controllers include:

* **Node Controller**: Responsible for noticing and responding when nodes go down.
* **Replication Controller**: Responsible for maintaining the correct number of pods for every replication controller object in the system.
* **Endpoints Controller**: Populates the Endpoints object (that is, joins Services & Pods)
* **Service Account & Token Controllers:** Create default accounts and API access tokens for new namespaces.

*Controller* - In Kubernetes, controllers are control loops that watch the state of your [cluster](https://kubernetes.io/docs/reference/glossary/?all=true#term-cluster) , then make or request changes where needed. Each controller tries to move the current cluster state closer to the desired state.

* **Cloud-controller-manager**

[cloud-controller-manager](https://kubernetes.io/docs/tasks/administer-cluster/running-cloud-controller/) runs controllers that interact with the underlying cloud providers.

The following controllers have cloud provider dependencies:

* **Node Controller**: For checking the cloud provider to determine if a node has been deleted in the cloud after it stops responding
* **Route Controller**: For setting up routes in the underlying cloud infrastructure
* **Service Controller**: For creating, updating and deleting cloud provider load balancers
* **Volume Controller**: For creating, attaching, and mounting volumes, and interacting with the cloud provider to orchestrate volumes
* **Node Components**

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

* **Kubelet**

An agent that runs on each [node](https://kubernetes.io/docs/concepts/architecture/nodes/) in the cluster. It makes sure that [containers](https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/#why-containers) are running in a [Pod](https://kubernetes.io/docs/concepts/workloads/pods/pod-overview/) .

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. Please refer to Node image at last page for kubelet.

* **Kube-proxy**

kube-proxy is a network proxy that runs on each [node](https://kubernetes.io/docs/concepts/architecture/nodes/) in your cluster, implementing part of the Kubernetes [Service](https://kubernetes.io/docs/concepts/services-networking/service/) concept.

[kube-proxy](https://kubernetes.io/docs/reference/command-line-tools-reference/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.

*Service* – A way to expose an application running on a set of [Pods](https://kubernetes.io/docs/concepts/workloads/pods/pod-overview/) as a network service.

* **Container Runtime**

The container runtime is the software that is responsible for running containers.

Kubernetes supports several container runtimes: : [Docker](https://docs.docker.com/engine/) , [containerd](https://containerd.io/docs/) , [CRI-O](https://cri-o.io/#what-is-cri-o) , and any implementation of the [Kubernetes CRI (Container Runtime Interface)](https://github.com/kubernetes/community/blob/master/contributors/devel/sig-node/container-runtime-interface.md).

* **Addons**

Few addons are listed below

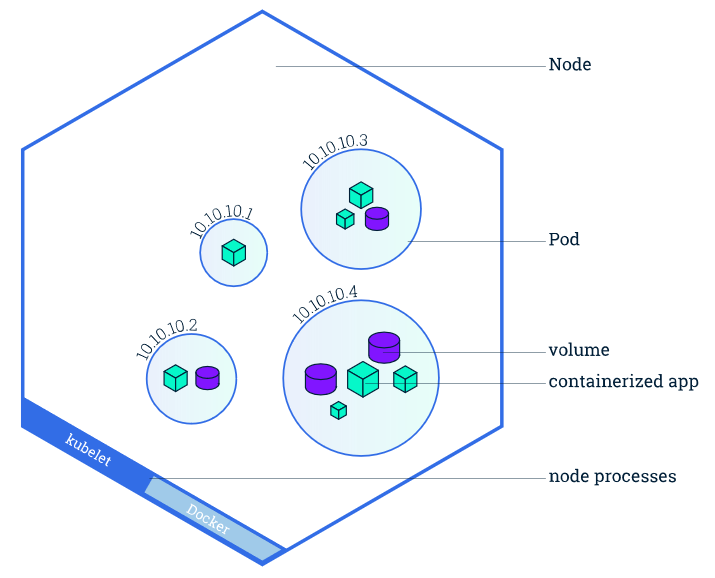
DNS

Web UI (Dashboard)

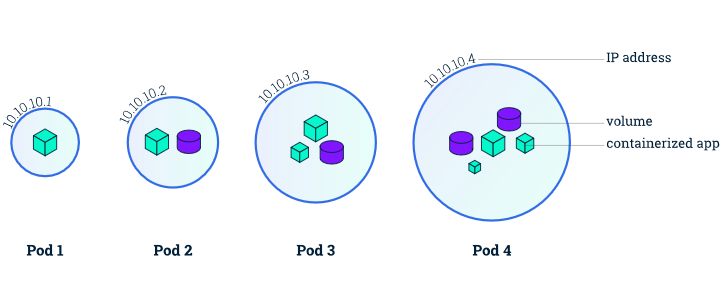
Container Resource Monitoring

Cluster-Level logging

**Node – Image**



**Pod – Image**



Reference: <https://kubernetes.io/docs/tutorials/kubernetes-basics/explore/explore-intro/>