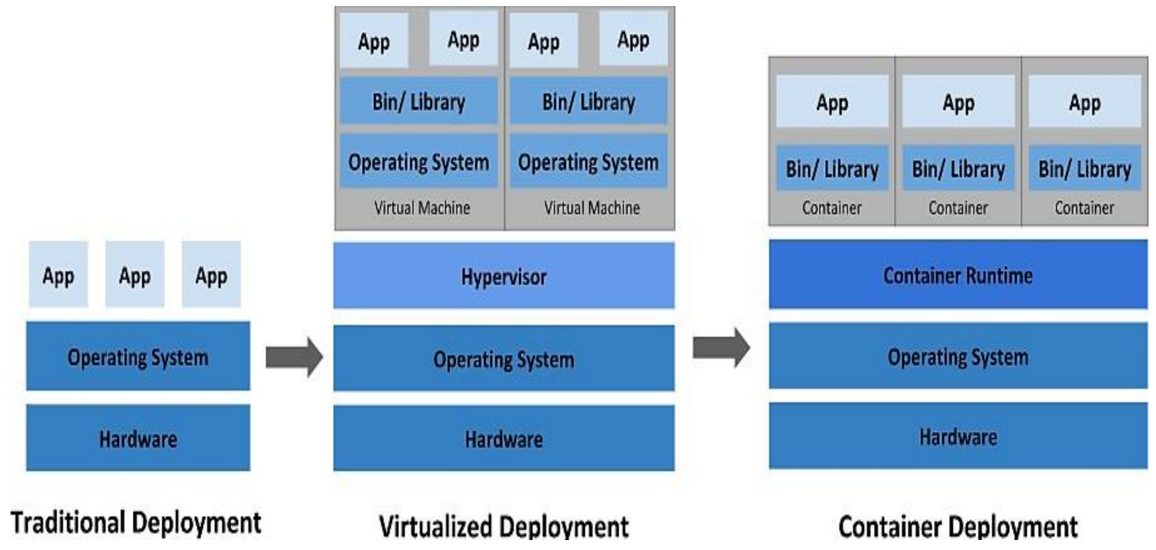
kubernetes

❖ KUBERNETES:

- Kubernetes is a portable, extensible, open-source platform for managing containerized workloads and services, that facilitatesboth declarative configuration and automation.
- Kubernetes also known as k8s or "kube". Kubernetes clusters can span hosts across on-premise, public,private, or hybrid clouds.

➢ TRADITIONAL vs VIRTUALIZATION CONTAINERS:



TRADITIONAL DEPLOYMENT:

- Early on, organizations ran applications on physical servers.
- There was no way to define resource boundaries for applications in a physical server, and this caused resource allocation issues.
- A solution for this would be to run each application on a different physical server. But this did not scale as resources were underutilized, and it was expensive for organizations to maintain many physical servers.

VIRTUALIZED DEPLOYMENT:

- It allows you to run multiple Virtual Machines (VMs) on a single physical server's CPU.
- Virtualization allows applications to be isolated between VMs and provides a level of security as the information of one application cannot be freely accessed by another application.

- Virtualization allows better utilization of resources in a physical server and allows better scalability because an application can be added or updated easily, reduces hardware costs, and much more.
- Each VM is a full machine running all the components, including its own operating system, on top of the virtualized hardware.

CONTAINER DEPLOYMENT:

- Containers are similar to VMs, but they have relaxed isolation properties to share the Operating System (OS) among the applications. Therefore, containers are considered lightweight. Similar to a VM, a container has its own filesystem, share of CPU, memory, process space, and more.
- As they are decoupled from the underlying infrastructure, they are portable across clouds and OS distributions.

➢ CONTAINERS HAVE BECOME POPULAR BECAUSE THEY PROVIDE EXTRA BENEFITS, SUCH AS:

- **Agile application creation and deployment:** increased ease and efficiency of container image creation compared to VM image use.
- **Continuous development, integration, and deployment:** provides for reliable and frequent container image build and deployment with quick and efficient rollbacks (due to image immutability).
- **Dev and Ops separation of concerns:** create application container images at build/release time rather than deployment time, thereby decoupling applications from infrastructure.
- Observability: not only surfaces OS level information and metrics, but also application health and other signals.
- **Environmental consistency across development, testing, and production:** Runs the same on a laptop as it does in the cloud.
- **Cloud and OS distribution portability:** Runs on Ubuntu, RHEL, CoreOS, on-premises, on major public clouds, and anywhere else.
- **Application-centric management:** Raises the level of abstraction from running an OS on virtual hardware to running an application on an OS using logical resources.
- **Loosely coupled, distributed, elastic, liberated micro-services:** applications are broken into smaller, independent pieces and can be deployed and managed dynamically – not a monolithic stack running on one big single-purpose machine.
- **Resource isolation:** predictable application performance.
- **Resource utilization:** high efficiency and density.

➢ KUBERNETES FEATURES:

- Containers are a good way to bundle and run your applications. In a production environment, you need to manage the containers that run the applications and ensure that there is no downtime. For example, if a container goes down, another container needs to start.
- Kubernetes Provides:

SERVICE DISCOVERY AND LOAD BALANCING:

Kubernetes can expose a container using the DNS name or using their own IP address. If traffic to a container is high, Kubernetes is able to load balance and distribute the network traffic so that the deployment is stable.

STORAGE ORCHESTRATION:

Kubernetes allows you to automatically mount a storage system of your choice, such as local storages, public cloud providers, and more.

AUTOMATED ROLLOUTS AND ROLLBACKS:

You can describe the desired state for your deployed containers using Kubernetes, and it can change the actual state to the desired state at a controlled rate. For example, you can automate Kubernetes to create new containers for your deployment, remove existing containers and adopt all their resources to the new container.

AUTOMATIC BIN PACKING:

You provide Kubernetes with a cluster of nodes that it can use to run containerized tasks. You tell Kubernetes how much CPU and memory (RAM) each container needs. Kubernetes can fit containers onto your nodes to make the best use of your resources.

SELF-HEALING:

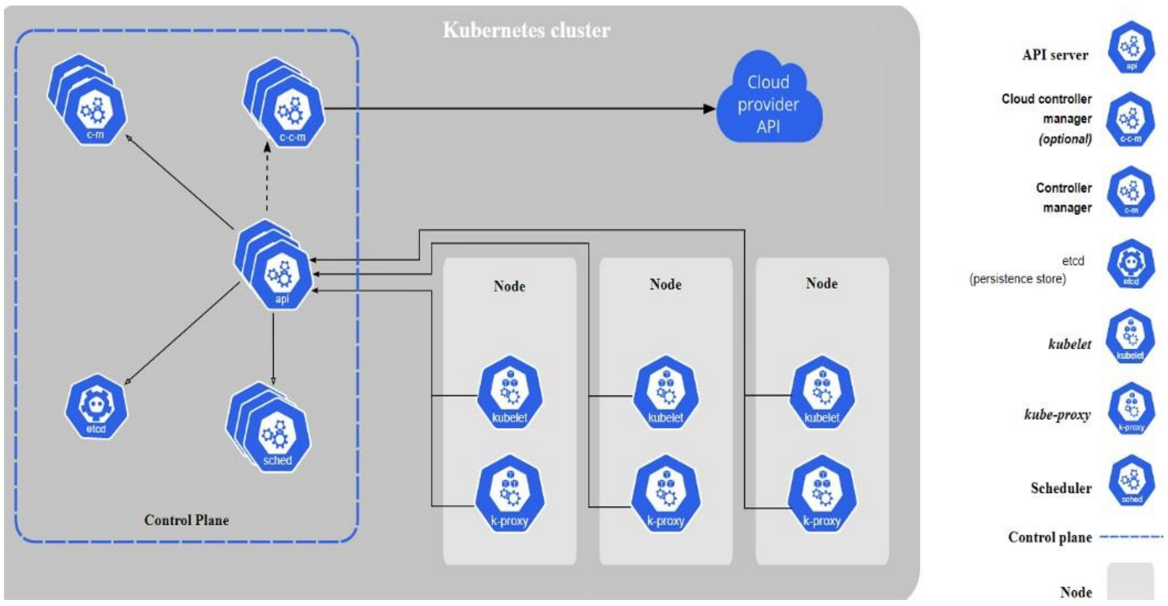
Kubernetes restarts containers that fail, replaces containers, kills containers that don't respond to your user-defined health check, and doesn't advertise them to clients until they are ready to serve.

SECRET AND CONFIGURATION MANAGEMENT:

Kubernetes lets you store and manage sensitive information, such as passwords, OAuth tokens, and SSH keys. You can deploy and update secrets and application configuration without rebuilding your container images, and without exposing secrets in your stack configuration.

➢ KUBERNETES ARCHITECTURE & COMPONENTS:

- A Kubernetes cluster consists of the components that represent the control plane and a set of machines called nodes.
- A **Kubernetes cluster** consists of a set of worker machines, called nodes, that run containerized applications. Every cluster has at least one worker node.
- The **worker node(s)** host the Pods that are the components of the application workload. The control plane manages the worker nodes and the Pods in the cluster. In production environments, the control plane usually runs across multiple computers and a cluster usually runs multiple nodes, providing fault-tolerance and high availability.



KUBERNETES CLUSTER:

- A Kubernetes cluster consists of a set of worker machines, callednodes, that run containerized applications. Every cluster has at least one worker node.
- The Worker Node(s) host the Pods that are the components of the application workload.
- The Control Plane manages the worker nodes and the Pods in thecluster.
- Cluster usually runs multiple nodes, providing Fault-Tolerance and High Availability.

CONTROL PLANE COMPONENTS:

KUBE-APISERVER:

- Kube-APIserver is the main management point of the entire cluster, handling internal and external requests.
- It processes REST operations, validates, and updates thecorresponding objects in etcd.

ETCD:

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

KUBE-SCHEDULER:

- It considers the resource needs of a pod, such as CPU or memory,along with the health of the cluster.
- It schedules the pod to an appropriate compute node.

KUBE-CONTROLLER-MANAGER:

- Kube-Control-Manager runs the control process.
- A control process is a loop that focuses on making the desired stateequal to the current state for any application in any given instance oftime.

CLOUD-CONTROLLER-MANAGER:

- The cloud controller manager links your cluster into your cloud provider's API, and separates out the components that interact withthat cloud platform from components that only interact with your cluster.
- The following controllers can 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

NODE COMPONENTS:

KUBELET:

- An agent that runs on each node in the cluster. It makes sure that containers are running in a Pod.
- When the control plane needs something to happen in a node, the kubelet executes the action.

KUBE-PROXY:

- Kube-proxy is a network proxy that runs on each node in your cluster.
- It maintains network rules on nodes. These network rules allow network communication to your Pods of your cluster.

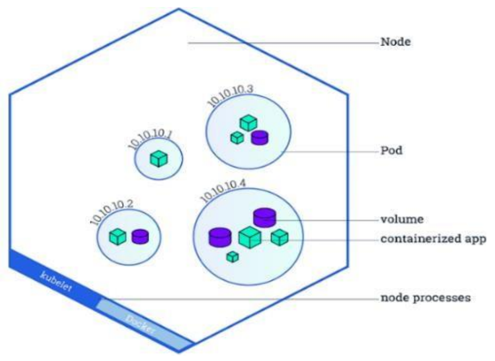
CONTAINER RUNTIME:

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

KUBERNETES NODES & PODS:

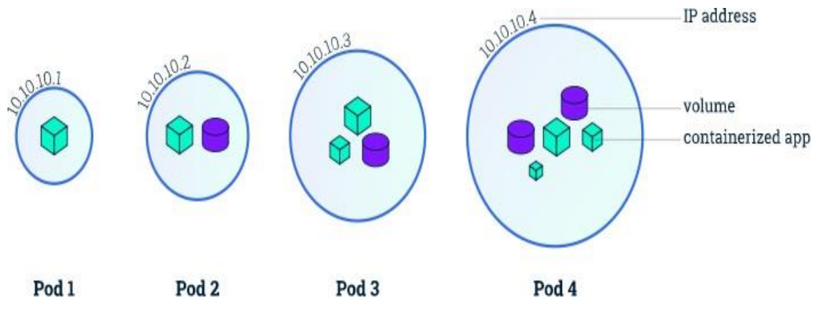
NODES:

- A node is a worker machine in Kubernetes and may be a VM or physical machine, depending on the cluster.
- Kubernetes runs your workload by placing containers into Pods to runon Nodes. A node can have multiple Pods.
- A node includes the kubelet, a container runtime, and kube-proxy.
- Node names are uniqueness in the cluster.



PODS:

- A pod is the smallest & simplest Kubernetes object. It represents asingle instance of a running process in your cluster.
- Pods contain one or more containers, such as Docker containers.
- When a Pod runs multiple containers, the containers are managed asa single entity and share the Pod's resources.
- Pods also contain shared networking and storage resources fortheir containers



NOTE: Running multiple containers in a single Pod is an advanced usecase.