KUBERNETES

—- Zero to Beginner—

- Applications have changed in recent years and how they can now be harder to deploy and manage. We're introduced with Kubernetes and shown how it, together with Docker and other container platforms, helps deploy and manage applications and the infrastructure they run on. You've learned that
- Monolithic apps are easier to deploy, but harder to maintain over time and sometimes impossible to scale.

- Microservices-based application architectures allow easier development of each component, but are harder to deploy and configure to work as a single system. Linux containers provide much the same benefits as virtual machines, but are far more lightweight and allow for much better hardware utilization.

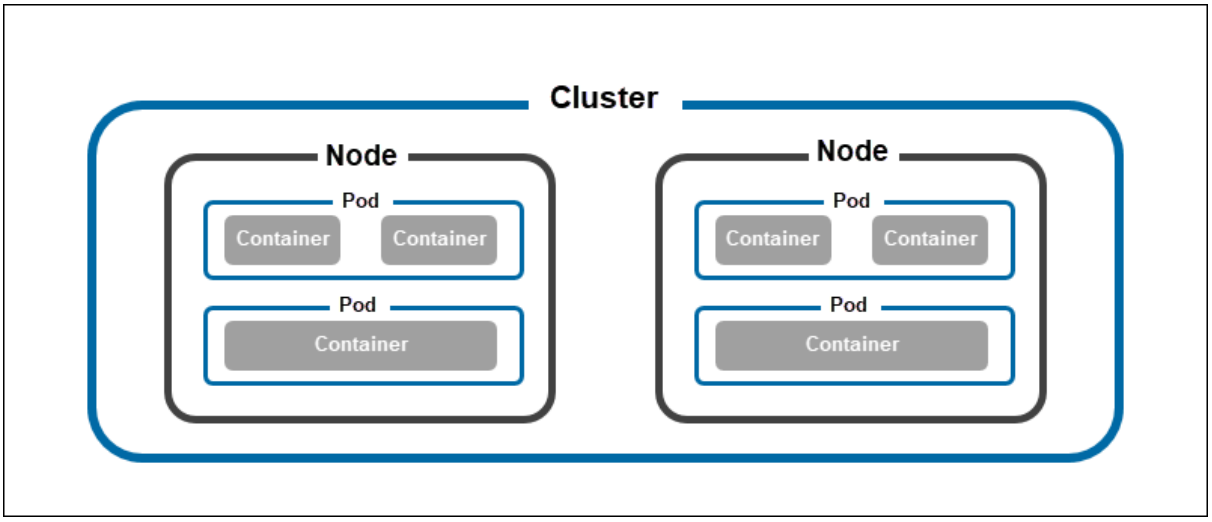
- Docker improved on existing Linux container technologies by allowing easier and faster provisioning of containerized apps together with their OS environments. Kubernetes exposes the whole datacenter as a single computational resource for running applications.
- Developers can deploy apps through Kubernetes without assistance from sysadmins. Sysadmins can sleep better by having Kubernetes deal with failed nodes auto- matically.

There are various advantages of container orchestration. Your application is now highly available as hardware failures do not bring your application down because you have multiple instances of your application running on different nodes. The user traffic is load balanced across the various containers. When demand increases, deploy more instances of the application seamlessly and within a matter of second and we have the ability to do that at a service level. When we run out of hardware resources, scale the number of nodes up/down without having to take down the application. And do all of these easily with a set of declarative object configuration files.

K8 is a container Orchestration technology used to orchestrate the deployment and management of 100s and 1000s of containers in a clustered environment.

NODES :

Let us start with Nodes. A node is a machine – physical or virtual – on which kubernetes is installed. A node is a worker machine and this is were containers will be launched by kubernetes. The below image is the visualization of what a **pod , node , cluster** actually is.



Kubernetes runs your workload by placing containers into Pods to run on *Nodes*. A node may be a virtual or physical machine, depending on the cluster. Each node is managed by the control plane and contains the services necessary to run Pods.

Typically you have several nodes in a cluster; in a learning or resource-limited environment, you might have only one node.

The components on a node include the kubelet, a container runtime, and the kube-proxy.

There are two types of nodes:

- The Kubernetes Master node—runs the Kubernetes control plane which controls the entire cluster. A cluster must have at least one master node; there may be two or more for redundancy. Components of the master node include the API Server, etcd (a database holding the cluster state), Controller Manager, and Scheduler.
- Worker nodes—these are nodes on which you can run containerized workloads. Each node runs the kubelet—an agent that enables the Kubernetes control plane to control the node. Kubernetes nodes are used by

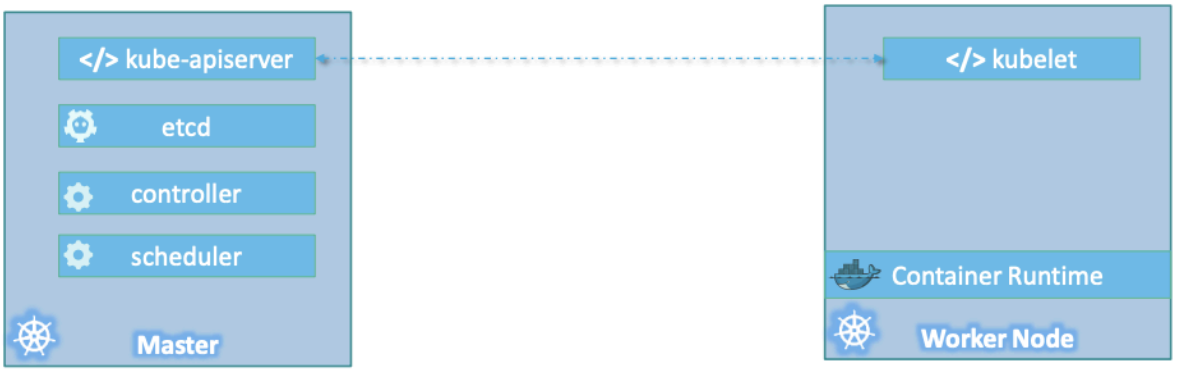
organizations to run a variety of workloads, as a core component in modern DevOps processes.

CLUSTER :

A cluster is a set of nodes grouped together. This way even if one node fails you have your application still accessible from the other nodes. Moreover having multiple nodes helps in sharing load as well.

you can manage, coordinate, and schedule containers at scale. Kubernetes places containers into pods and runs them on nodes. A Kubernetes cluster has, at a minimum, a master node running a container pod and a control plane that manages the cluster. When you deploy Kubernetes, you are essentially running a Kubernetes cluster.

Now we have a cluster, but who is responsible for managing the cluster? Were is the information about the members of the cluster stored? How are the nodes monitored? When a node fails how do you move the workload of the failed node to another worker node? That's were the Master comes in. The master is another node with Kubernetes installed in it, and is configured as a Master. The master watches over the nodes in the cluster and is responsible for the actual orchestration of containers on the worker nodes.



Similarly the worker nodes have the kubelet agent that is responsible for interacting with the master to provide health information of the worker node and carry out actions requested by the master on the worker nodes. 29 All the information gathered are stored in a key-value store on the Master. The key value store is based

on the popular etcd framework as we just discussed. The master also has the controller manager and the scheduler

When you install Kubernetes on a System, you are actually installing the following components. An API Server. An ETCD service. A kubelet service. A Container Runtime, Controllers and Schedulers.

The **API server** acts as the front-end for kubernetes. The users, management devices, Command line interfaces all talk to the API server to interact with the kubernetes cluster.

Next is the **ETCD** key store. ETCD is a distributed reliable key-value store used by kubernetes to store all data used to manage the cluster. Think of it this way, when you have multiple nodes and multiple masters in your cluster, etcd stores all that information on all the nodes in the cluster in a distributed manner. ETCD is responsible for implementing locks within the cluster to ensure there are no conflicts between the Masters.

The **scheduler** is responsible for distributing work or containers across multiple nodes. It looks for newly created containers and assigns them to Nodes.

The **controllers** are the brain behind orchestration. They are responsible for noticing and responding when nodes, containers or endpoints goes down. The controllers makes decisions to bring up new containers in such cases.

The container runtime is the underlying software that is used to run containers. In our case it happens to be Docker.

And finally **kubelet** is the agent that runs on each node in the cluster. The agent is responsible for making sure that the containers are running on the nodes as expected.The kube control tool is used to deploy and manage applications on a kubernetes cluster, to get cluster information, get the status of nodes in the cluster and many other things. The kubectl run command is used to deploy an application on the cluster. The kubectl cluster-info command is used to view information about the cluster and the kubectl get pod command is used to list all the nodes part of the cluster.

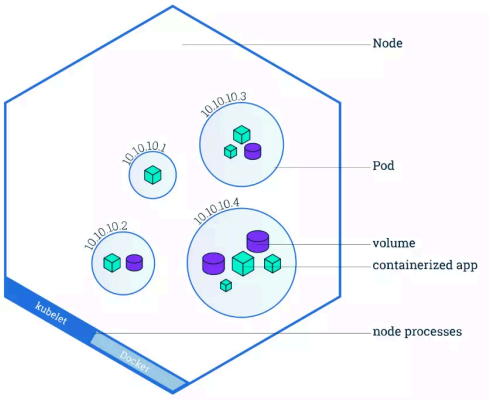
- **kubectl get nodes**
- **kubectl run hello-minikube**

- **kubectl cluster-info**

PODS :

With kubernetes our ultimate aim is to deploy our application in the form of containers on a set of machines that are configured as worker nodes in a cluster. However, kubernetes does not deploy containers directly on the worker nodes. The containers are encapsulated into a Kubernetes object known as PODs. A POD is a single instance of an application. A POD is the smallest object, that you can create in kubernetes.

Pods Horizontal scaling is , when pods get hit by more traffic , you can increase instances of pod . You can do it through replica controllers by telling how many replicas (instances) of pod the node should keep up running always. When a pod gets down/deleted , replica will add another pod and balance the number of healthily running pods. We'll learn about replicas in later chapters.



Before learning about **K8** , **Pods, nodes, clusters, replicas , volumes , services , deployments ,yaml structures** to build these components, Lets learn about learn about Docker , containers , images .Lets Jump to

DOCKER - Zero to Beginner