DOCKER

S
u
j
a
t
a
B
a
t
r
a

WHAT IS DOCKER ?

- Docker is an **open-source centralized platform designed** to create, deploy, and run applications.
- Docker uses **container** on the host's operating system to run applications.
- It allows applications to use the same **Linux kernel** as a system on the host computer, rather than creating a whole virtual operating system.
- Containers ensure that our application works in any environment like development, test, or production.

DOCKER CONTAINERS

- Docker containers are the **lightweight** alternatives of the virtual machine.
- It allows developers to package up the application with all its libraries and dependencies, and ship it as a single package.
- The advantage of using a docker container is that you don't need to allocate any RAM and disk space for the applications.
- It automatically generates storage and space according to the application requirement.

VIRTUAL MACHINE

- A virtual machine is a software that allows us to install and use other operating systems (Windows, Linux, and Debian) simultaneously on our machine.
- The operating system in which virtual machine runs are called virtualized operating systems.
- These virtualized operating systems can run programs and preforms tasks that we perform in a real operating system.

CONTAINER VS VIRTUAL MACHINE

CONTAINERS

- Integration in a container is faster and cheap.
- No wastage of memory.
- It uses the same kernel, but different distribution.

VIRTIAL MACHINE

- Integration in virtual is slow and costly.
- Wastage of memory.
- It uses multiple independent operating systems.

WHY DOCKER

- Docker allows us to easily install and run software without worrying about setup or dependencies.
- Developers use Docker to eliminate machine problems, i.e. "**but code is worked on my laptop**." when working on code together with co-workers.
- Operators use Docker to run and manage apps in isolated containers for better compute density.
- Enterprises use Docker to securely built agile software delivery pipelines to ship new application features faster and more securely.
- Since docker is not only used for the deployment, but it is also a great platform for development, that's why we can efficiently increase our customer's satisfaction.

ADVANTAGES OF DOCKER

- It runs the container in seconds instead of minutes.
- It uses less memory.
- It provides lightweight virtualization.
- It does not a require full operating system to run applications.
- It uses application dependencies to reduce the risk.
- Docker allows you to use a remote repository to share your container with others.
- It provides continuous deployment and testing environment.

DISADVANTAGES OF DOCKER

- It increases complexity due to an additional layer.
- In Docker, it is difficult to manage large amount of containers.
- Some features such as container self -registration, containers self-inspects, copying files form host to the container, and more are missing in the Docker.
- Docker is not a good solution for applications that require rich graphical interface.
- Docker provides cross-platform compatibility means if an application is designed to run in a Docker container on Windows, then it can't run on Linux or vice versa.

KUBERNETES

WHAT IS KUBERNETES

- **Kunernetes** is also known as **‘k8s’.**
- **Kubernetes** is an extensible, portable, and open-source platform designed by **Google** in **2014**.
- It is mainly used to automate the deployment, scaling, and operations of the container-based applications across the cluster of nodes.
- It is also designed for managing the services of containerized apps using different methods which provide the scalability, predictability, and high availability.
- It is actually an enhanced version of '**Borg**' for managing the long-running processes and batch jobs.
- Nowadays, many cloud services offer a Kubernetes-based infrastructure on which it can be deployed as the platform-providing service.
- This technique or concept works with many container tools, like **docker,** and follows the client-server architecture.

KEY OBJECTS OF KUBERNETES

- **Pod**
  - It is the smallest and simplest basic unit of the Kubernetes application. This object indicates the processes which are running in the cluster.
- **Node**
  - A **node** is nothing but a single host, which is used to run the virtual or physical machines. A node in the Kubernetes cluster is also known as a minion.
- **Service**
  - A **service** in a Kubernetes is a logical set of pods, which works together. With the help of services, users can easily manage load balancing configurations.
- **ReplicaSet**
  - A **ReplicaSet** in the Kubernetes is used to identify the particular number of pod replicas are running at a given time. It replaces the replication controller because it is more powerful and allows a user to use the "set-based" label selector.
- **Namespace**
  - **Kubernetes** supports various virtual clusters, which are known as namespaces. It is a way of dividing the cluster resources between two or more users.

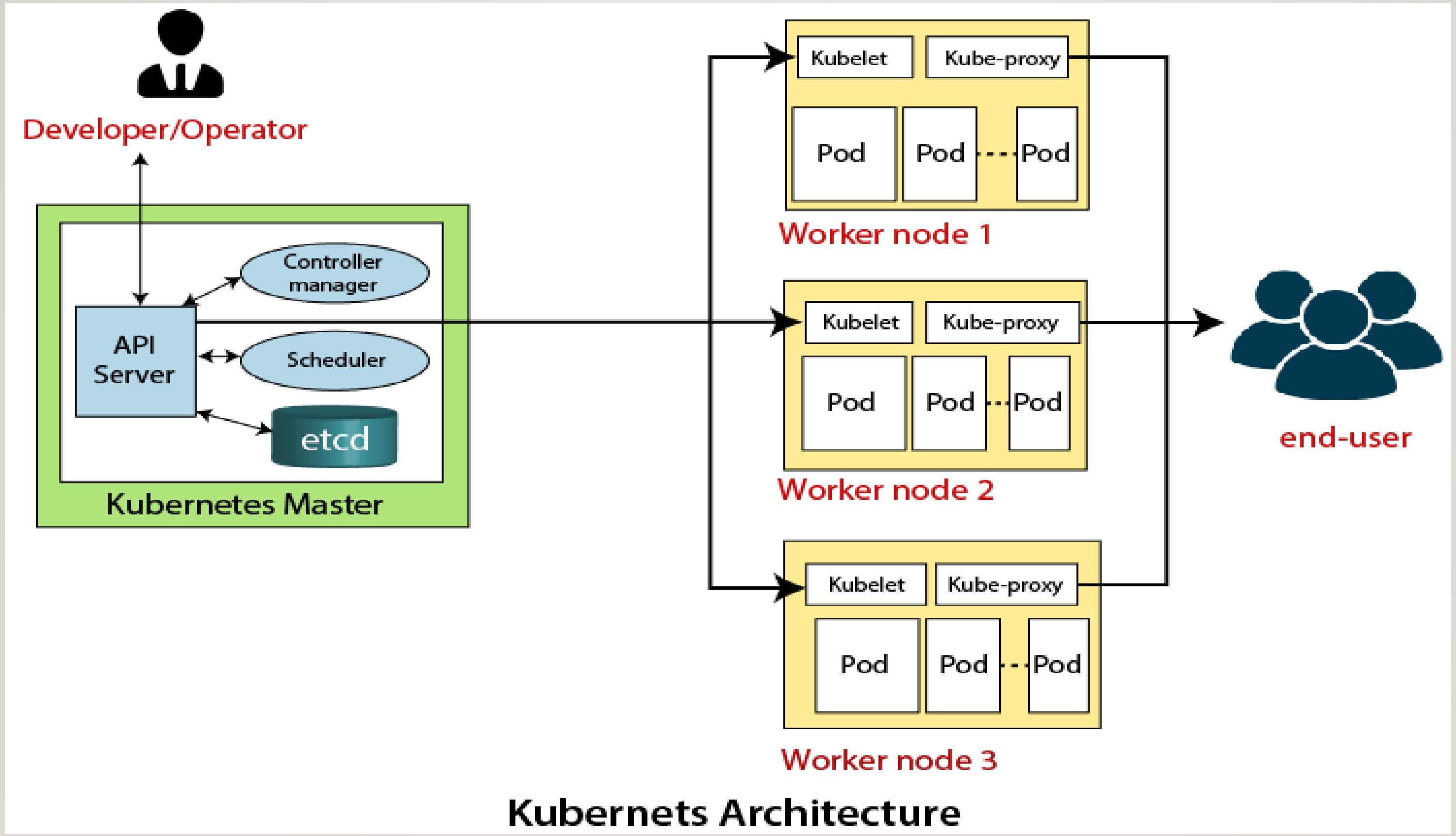
FEATURES OF KUBERNETES

1. **Pod:** It is a deployment unit in Kubernetes with a single Internet protocol address.
2. **Horizontal Scaling:** It is an important feature in the Kubernetes. This feature uses a **HorizontalPodAutoscalar** to automatically increase or decrease the number of pods in a deployment, replication controller, replica set, or stateful set on the basis of observed CPU utilization.
3. **Automatic Bin Packing:** Kubernetes helps the user to declare the maximum and minimum resources of computers for their containers.
4. **Service Discovery and load balancing:** Kubernetes assigns the IP addresses and a Name of DNS for a set of containers, and also balances the load across them.
5. **Automated rollouts and rollbacks:** Using the rollouts, Kubernetes distributes the changes and updates to an application or its configuration. If any problem occurs in the system, then this technique rollbacks those changes for you immediately.

FEATURES OF KUBERNETES

1. **Persistent Storage:** Kubernetes provides an essential feature called '**persistent storage'** for storing the data, which cannot be lost after the pod is killed or rescheduled. Kubernetes supports various storage systems for storing the data, such as **Google Compute Engine's Persistent Disks (GCE PD) or Amazon Elastic Block Storage (EBS).** It also provides the distributed file systems: **NFS or GFS**.
2. **Self-Healing:** This feature plays an important role in the concept of Kubernetes. Those containers which are failed during the execution process, Kubernetes restarts them automatically. And, those containers which do not reply to the user-defined health check, it stops them from working automatically.

KUBERNETES ARCHITECTURE



KUBERNETES ARCHITECTURE

- The architecture of Kubernetes actually follows the client-server architecture. It consists of the following two main components:

1. Master Node (Control Plane)
2. Slave/worker node

MASTER NODE OR KUNERNETES CONTROL PLANE

- The master node in a Kubernetes architecture is used to manage the states of a cluster. It is actually an entry point for all types of administrative tasks. In the Kubernetes cluster, more than one master node is present for checking the fault tolerance.
- Following are the four different components which exist in the Master node or Kubernetes Control plane:

1. API Server
2. Scheduler
3. Controller Manager
4. ETCD

COMPONENTS OF MASTER NODE

- **API Server**
  - The Kubernetes API server receives the REST commands which are sent by the user. After receiving, it validates the REST requests, process, and then executes them. After the execution of REST commands, the resulting state of a cluster is saved in '**etcd**' as a distributed key-value store.
- **Scheduler**
  - The scheduler in a master node schedules the tasks to the worker nodes. And, for every worker node, it is used to store the resource usage information. In other words, it is a process that is responsible for assigning pods to the available worker nodes.

COMPONENTS OF MASTER NODE

- **Controller Manager**
  - The Controller manager is also known as a controller. It is a daemon that executes in the non-terminating control loops. The controllers in a master node perform a task and manage the state of the cluster. In the Kubernetes, the controller manager executes the various types of controllers for handling the nodes, endpoints, etc.
- **ETCD**
  - It is an open-source, simple, distributed key-value storage which is used to store the cluster data. It is a part of a master node which is written in a GO programming language.

WORKER/SLAVE NODE

- The Worker node in a Kubernetes is also known as minions.
- A worker node is a physical machine that executes the applications using pods.
- It contains all the essential services which allow a user to assign the resources to the scheduled containers.

COMPONENTS OF WORKER/SLAVE NODES

- Kubelet
  - This component is an agent service that executes on each worker node in a cluster.
  - It ensures that the pods and their containers are running smoothly.
  - Every **kubelet** in each worker node communicates with the master node.
  - It also starts, stops, and maintains the containers which are organized into pods directly by the master node.
- Kube-proxy
  - It is a proxy service of Kubernetes, which is executed simply on each worker node in the cluster.
  - The main aim of this component is request forwarding.
  - Each node interacts with the Kubernetes services through **Kube-proxy**.

COMPONENTS OF WORKER/SLAVE NODES

- Pods
  - A **pod** is a combination of one or more containers which logically execute together on nodes. One worker node can easily execute multiple pods.

DOCKER VS KUBERNETES

DOCKER

- Docker is developed by **Docker Inc**.
- It was first released in **2013**
- It is a container based technology used to create isolated environment for applications
- It allows us to use **third-party tools** like ELK for logging and monitoring.
- Its public cloud service provider is **only Azure**
- It is **less customizable**.
- Its container limit is **95000**.
- It is **easy** to install.
- It **cannot** do **auto-scaling.**
- It **does not** provide any **dashboard.**

KUBERNETES

- Kubernetes is developed by **Google**
- It was first released in **2014**.
- It is an infrastructure for managing multiple containers.
- It allows us to use **in-built tools** for logging and monitoring.
- Its public cloud service providers are **Google**, **Azure**, and **AWS**.
- It is **highly customizable**.
- Its container limit is **300000**.
- It is **complex** to install.
- It can **do auto-scaling**.
- It provides a **Web UI dashboard**.