NAME: MULLA AFRAH AKKAS ALI

ROLL NO.: 612038

BRANCH: T.E. – I.T.

SEMESTER: ODD SEMESTER 5

COURSE: Advance DevOPs (ITL504)

DATE: 24-10-2022

EXPERIMENT 12

1. What is Kubernetes?

Kubernetes (also known as k8s or “kube”) is an open source container orchestration platform that automates many of the manual processes involved in deploying, managing, and scaling containerized applications.

While in practice Kubernetes is most often used with Docker, the most popular containerization platform, it can also work with any container system that conforms to the Open Container Initiative (OCI) standards for container image formats and runtimes. And because Kubernetes is open source, with relatively few restrictions on how it can be used, it can be used freely by anyone who wants to run containers, most anywhere they want to run them—on-premises, in the public cloud, or both.

Kubernetes automates operational tasks of container management and includes built-in commands for deploying applications, rolling out changes to your applications, scaling your applications up and down to fit changing needs, monitoring your applications, and more—making it easier to manage applications.

2. How is Kubernetes related to Docker?

Often misunderstood as a choice between one or the other, Kubernetes and Docker are different yet complementary technologies for running containerized applications.

Docker lets you put everything you need to run your application into a box that can be stored and opened when and where it is required. Once you start boxing up your applications, you need a way to manage them; and that's what Kubernetes does.

Kubernetes is a Greek word meaning ‘captain’ in English. Like the captain is responsible for the safe journey of the ship in the seas, Kubernetes is responsible for carrying and delivering those boxes safely to locations where they can be used.

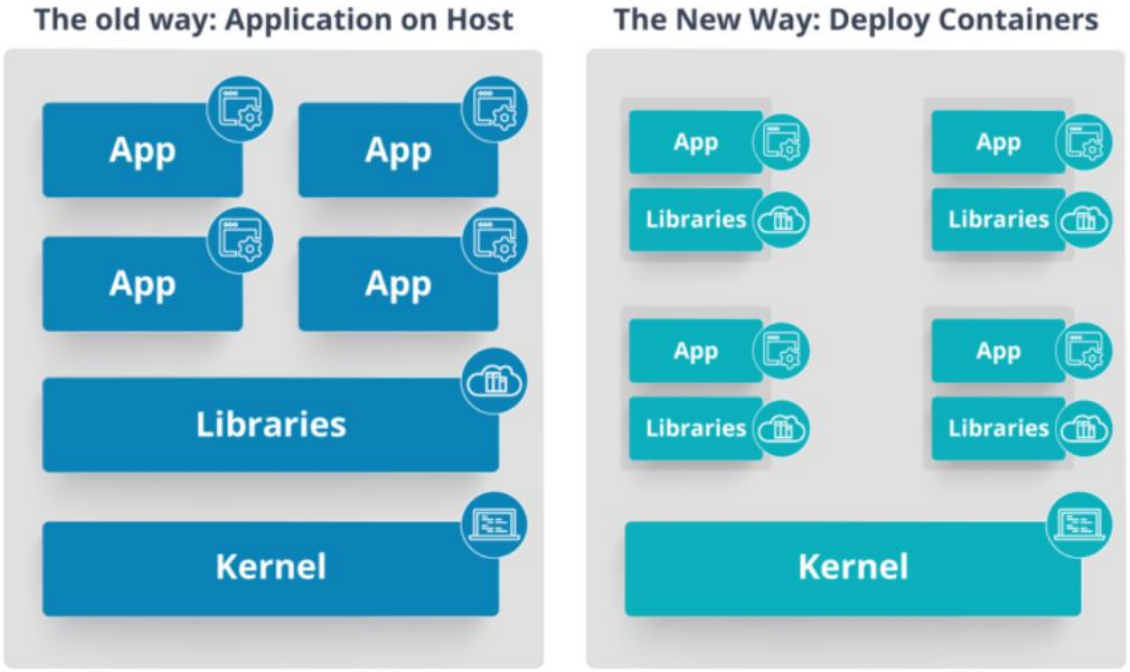
- Kubernetes can be used with or without Docker.
- Docker is not an alternative to Kubernetes, so it’s less of a “Kubernetes vs. Docker” question. It’s about using Kubernetes with Docker to containerize your applications and run them at scale.
- The difference between Docker and Kubernetes relates to the role each play in containerizing and running your applications.
- Docker is an open industry standard for packaging and distributing applications in containers.

- Kubernetes uses Docker to deploy, manage, and scale containerized applications.

Kubernetes is open-source orchestration software that provides an API to control how and where those containers will run. It allows you to run your Docker containers and workloads and helps you to tackle some of the operating complexities when moving to scale multiple containers, deployed across multiple servers.

3. What is the difference between deploying applications on hosts and containers?

The left side architecture represents deploying applications on hosts. So, this kind of architecture will have an operating system and then the operating system will have a kernel which will have various libraries installed on the operating system needed for the application. So, in this kind of framework you can have n number of applications and all the applications will share the libraries present in that operating system whereas while deploying applications in containers the architecture is a little different.



This kind of architecture will have a kernel and that is the only thing that's going to be the only thing common between all the applications. So, if there's a particular application which needs Java then that particular application we'll get access to Java and if there's another application which needs Python then only that particular application will have access to Python.

The individual blocks that you can see on the right side of the diagram are basically containerized and these are isolated from other applications. So, the applications have the necessary libraries and binaries isolated from the rest of the system, and cannot be encroached by any other application.

4. Read the steps to Install Kubernetes on ubuntu server from AWS EC2 [one Master and one or more Worker Node] [Refer attached manual]

5. Read the steps to install Docker Install Kubectl, kubelet and kubeadm on all nodes.

Steps to Install Kubernetes on Ubuntu:

Set up Docker

Step 1: Install Docker

1. Update the package list with the command:

```
on-master&slave$sudo apt-get update
```

2. Next, install Docker with the command:

```
on-master&slave$sudo apt-get install docker.io
```

3. Repeat the process on each server that will act as a node.
4. Check the installation (and version) by entering the following:

```
on-master&slave$docker --version
```

Step 2: Start and enable docker.

1. Set Docker to launch at boot by entering the following:

```
on-master&slave$sudo systemctl enable docker
```

2. Verify Docker is running:

```
on-master&slave$sudo systemctl status docker
```

To start Docker if it's not running:

```
on-master&slave$sudo systemctl start docker
```

3. Repeat on all the other nodes.

Install Kubernetes

Step 3: Add Kubernetes Signing Key using commands.
Step 4: Add Software Repositories by entering commands.
Repeat on each server node.
Step 5: Kubernetes Installation Tools

Kubeadm (Kubernetes Admin) is a tool that helps initialize a cluster. It fast-tracks setup by using community-sourced best practices. Kubelet is the work package, which runs on every node and starts containers. The tool gives you command-line access to clusters.

1. Install Kubernetes tools with the command:

```
    on-master&slave$sudo apt-get install -y kubelet=1.20.2-
00kubeadm=1.20.2-00 kubectl=1.20.2-00
on-master&slave$sudo apt-mark hold kubeadm kubelet kubectl
```

Allow the process to complete.

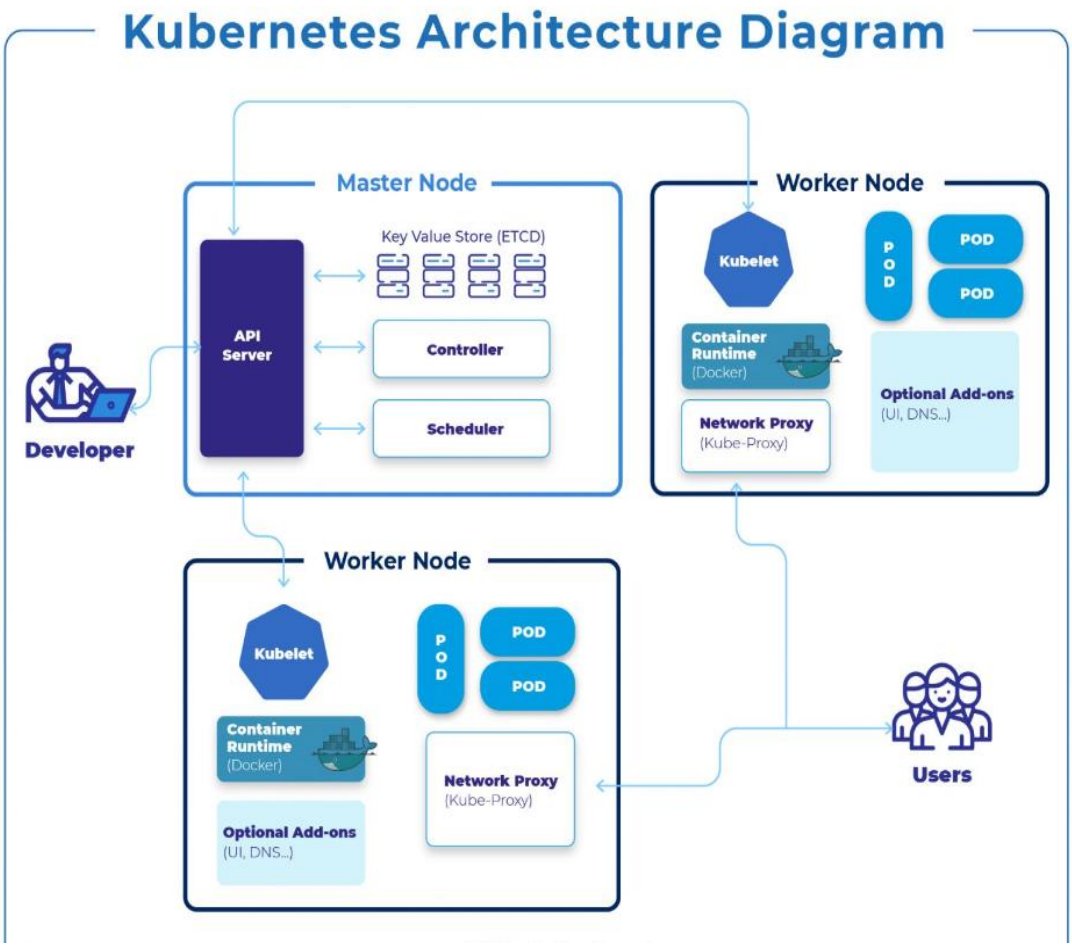
2. Verify the installation with:

```
on-master&slave$kubeadm version
```

6. Explain Kubernetes architecture with neat diagram and explain all the component of Kubernetes architecture.

Kubernetes has a decentralized architecture that does not handle tasks sequentially. It functions based on a declarative model and implements the concept of a 'desired state.' These steps illustrate the basic Kubernetes process:

1. An administrator creates and places the desired state of an application into a manifest file.
2. The file is provided to the Kubernetes API Server using a CLI or UI. Kubernetes' default command-line tool is called kubectl.
3. Kubernetes stores the file (an application's desired state) in a database called the Key-Value Store (etcd).
4. Kubernetes then implements the desired state on all the relevant applications within the cluster.
5. Kubernetes continuously monitors the elements of the cluster to make sure the current state of the application does not vary from the desired state.



Components of Kubernetes architecture diagram are nodes (set of machines) and the control plane. Now, let's get deeper into those components.

Master Node is the starting point for all administrative tasks, and its responsibility is managing the Kubernetes cluster architecture.

It's possible to have more than one master node within the cluster, and what's required for checking the fault tolerance as more master nodes will place the system in the mode known as "High Availability." However, one master node has the role of the main node that performs all the tasks.

Master Node Components:

API Server

The API Server is the front-end of the control plane and the only component in the control plane that we interact with directly. Internal system components, as well as external user components, all communicate via the same API.

Key-Value Store (etcd)

The Key-Value Store, also called etcd, is a database Kubernetes uses to back-up all cluster data. It stores the entire configuration and state of the cluster. The Master node queries etcd to retrieve parameters for the state of the nodes, pods, and containers.

Controller

The role of the Controller is to obtain the desired state from the API Server. It checks the current state of the nodes it is tasked to control, and determines if there are any differences, and resolves them, if any.

Scheduler

A Scheduler watches for new requests coming from the API Server and assigns them to healthy nodes. It ranks the quality of the nodes and deploys pods to the best-suited node. If there are no suitable nodes, the pods are put in a pending state until such a node appears.

What is Worker Node in Kubernetes Architecture?

Worker nodes listen to the API Server for new work assignments; they execute the work assignments and then report the results back to the Kubernetes Master node.

Kubelet

The kubelet runs on every node in the cluster. It is the principal Kubernetes agent. By installing kubelet, the node's CPU, RAM, and storage become part of the broader cluster. It watches for tasks sent from the API Server, executes the task, and reports back to the Master. It also monitors pods and reports back to the control panel if a pod is not fully functional. Based on that information, the Master can then decide how to allocate tasks and resources to reach the desired state.

Container Runtime

The container runtime pulls images from a container image registry and starts and stops containers. A 3rd party software or plugin, such as Docker, usually performs this function.

Kube-proxy

The kube-proxy makes sure that each node gets its IP address, implements local iptables and rules to handle routing and traffic load-balancing.

Pod

A pod is the smallest element of scheduling in Kubernetes. Without it, a container cannot be part of a cluster. If you need to scale your app, you can only do so by adding or removing pods.

The pod serves as a ‘wrapper’ for a single container with the application code. Based on the availability of resources, the Master schedules the pod on a specific node and coordinates with the container runtime to launch the container.

In instances where pods unexpectedly fail to perform their tasks, Kubernetes does not attempt to fix them. Instead, it creates and starts a new pod in its place. This new pod is a replica, except for the DNS and IP address.