**Kubernetes Assignment 1**

1. What is a Kubernetes cluster?

* Kubernetes is used to manage your containers
* It is an open-source container Orchestration Framework.
* it manages containers - to manage applications that are made of containers physical machines, virtual machines, hybrid environments
* A **Kubernetes cluster** is a set of nodes that run containerized applications.
* If you’re running Kubernetes, you’re running a cluster.
* Containerizing applications packages an app with its dependences and some necessary services. They are more lightweight and flexible than virtual machines.
* Kubernetes clusters allow for applications to be more easily developed, moved and managed.
* Kubernetes clusters allow containers to run across multiple machines and environments: virtual, physical, cloud-based, and on-premises.
* Kubernetes containers are not restricted to a specific operating system, unlike virtual machines. Instead, they are able to share operating systems and run anywhere.
* Kubernetes clusters are comprised of **one master node and a number of worker nodes.** These nodes can either be physical computers or virtual machines, depending on the cluster.
* The master node controls the state of the cluster; for example, which applications are running and their corresponding container images. The master node is the origin for all task assignments. It coordinates processes such as:
  + Scheduling and scaling applications
  + Maintaining a cluster’s state
  + Implementing updates
* The worker nodes are the components that run these applications. Worker nodes perform tasks assigned by the master node. They can either be virtual machines or physical computers, all operating as part of one system.
* There must be a minimum of one master node and one worker node for a Kubernetes cluster to be operational.

1. What are the different parts of the Kubernetes architecture?

* In kubernets you can interact with architecture in 2 methods

1) User inteface - this is pretty much easy. So when you install kubernets in local machine so you are going to get user interface with which you can interact with architecture

2) command line interface this is mostly used and you can interact with kubectl

kubectl is command line utlity with which you can intercat with kubernetes architecture

* **First is Master Node** is also known as Control plane

- using control plane you can easily interact with worker node

- worker nodes are the nodes on which application is running

- you can have n no of worker nodes

we are having only one master mode-control plane

Inside Control PLane we have 4 things

1. **API server**

* it server as an entry point to interact with worker node. Exposes a REST interface to all Kubernetes resources.
* Serves as the front end of the Kubernetes control plane. Serves as the front end of the Kubernetes control plane.

1. **Scheduler** - it is used to schedule a node.

* if you want upscale/downscale/upgdee nodes you can schedle them here.Places containers according to resource requirements and metrics. Makes note of Pods with no assigned node, and selects nodes for them to run on.
* Places containers according to resource requirements and metrics. Makes note of Pods with no assigned node, and selects nodes for them to run on.

1. **Controller Manager**

* task of Control Manager is to create note, delete note, update.
* it keeps a track of each resources, ef. what no. of nodes are running. etc.
* Runs controller processes and reconciles the cluster’s actual state with its desired specifications. Manages controllers such as node controllers, endpoints controllers and replication controllers.

1. **etcd**

* server as a key value storage with track and store track of kubernetes architecture
* Stores all cluster data. Consistent and highly available Kubernetes backing store
* **Second is** Worker Node
* you can have N no. of nodes, dpeending on how much big your app is.
* Inside worker node we below:

1. **pod**

* It is smallest unit inside worker node insite which containers are running. it will be encapusalted inside the pod. you can n no. of pods. under which you applications runs

1. **Docker**

* it provides containerzation solution

1. **Kubelet**

* task/use is a node agent which tracks/stores, create, update,delete contrainer. Component is being managed by kubelet.
* Ensures that containers are running in a Pod by interacting with the Docker engine , the default program for creating and managing containers. Takes a set of provided PodSpecs and ensures that their corresponding containers are fully operational.

1. **Kube proxy**

* it is used for network proxy
* Manages network connectivity and maintains network rules across nodes. Implements the Kubernetes Service concept across every node in a given cluster.

1. What exactly do you mean by "container orchestration"?

* Container orchestration is mechanism/process to automate the provisioning, deployment, networking, scaling, availability, and lifecycle management of containers.
* Enterprises that need to deploy and manage hundreds or thousands of Linux containers and hosts can benefit from container orchestration.
* Container orchestration can be used in any environment where you use containers. It can help you to deploy the same application across different environments without needing to redesign it. And microservices in containers make it easier to orchestrate services, including storage, networking, and security.
* Today, Kubernetes is the most popular container orchestration platform, and most leading public cloud providers - including Amazon Web Services (AWS), Google Cloud Platform, IBM Cloud and Microsoft Azure - offer managed Kubernetes services. Other container orchestration tools include Docker Swarm and Apache Mesos.
* Use container orchestration to automate and manage tasks such as:

o Provisioning and deployment

o Configuration and scheduling

o Resource allocation

o Container availability

o Scaling or removing containers based on balancing workloads across your infrastructure

o Load balancing and traffic routing

o Monitoring container health

o Configuring applications based on the container in which they will run

Keeping interactions between containers secure

1. What are the various features of Kubernetes?

* Below are some of the essential Kubernetes features:
* Automated Scheduling/Automates various manual processes - for instance, Kubernetes will control for you which server will host the container, how it will be launched etc.
* Interacts with several groups of containers - Kubernetes is able to manage more cluster at the same time
* Provides additional services - as well as the management of containers, Kubernetes offers security, networking and storage services
* Self-monitoring - Kubernetes checks constantly the health of nodes and containers
* Horizontal scaling - Kubernetes allows you scaling resources not only vertically but also horizontally, easily and quickly
* Storage orchestration: Kubernetes mounts and add storage system of your choice to run apps
* Automates rollouts and rollbacks: if after a change to your application something goes wrong, Kubernetes will rollback for you sometimes also known for Self-Healing Capabilities
* Container balancing: Kubernetes always knows where to place containers, by calculating the “best location” for them
* Run everywhere: Kubernetes is an open source tool and gives you the freedom to take advantage of on-premises, hybrid, or public cloud infrastructure, letting you move workloads to anywhere you want.

1. Explain the relationship between Kubernetes and Docker?

* Docker is used to run containers, while kubernates manages those containerization.
* Even though Docker and Kubernetes are separate technologies, they actually have some relation/complement each other and work great together. In fact, they have a symbiotic relationship.
* Docker is at the core of containerization technology it allows you to create and deploy application containers.
* Suppose your application is still simple, Docker has the essential infrastructure for managing its lifecycle.
* As day by day if suppose you containers/app gets on increasing and more complex, possibly needing multiple clusters and more sophisticated management, Kubernetes becomes a handy tool.
* It offers a useful orchestration platform for your Docker containers. Kubernetes does not create containers; it actually requires a container tool to run, of which Docker is the most popular option.
* So Docker vs. Kubernetes actually points to their ability to work together to realize the promise of the containerization technology—code once and run anywhere, regardless of the scale.
* Docker can run with kubernetes, but Kubernetes either need some of the conaterization to run kubernetes.
* Using Kubernetes with Docker results in the following benefits:
* It enhances the robustness of your infrastructure.
* Your applications are more highly available.
* It improves the scalability of your applications.
* You can easily spin up your applications to handle more load on demand, potentially lowering resource wastages and enhancing user experience.
* Because apps are broken down into smaller constituents, they are easier to maintain.