Kubernetes Assignment 3

1. How do you monitor the Kubernetes cluster?

You can use various tools to monitor the operation of and the state of containers running within Kubernetes. One of the most commonly used tools is Prometheus, and it has multiple components, as described below.

- The server of Prometheus scrapes and stores time-series data
- It comes with the client libraries that help in instrumenting the application code
- It has a push gateway to help in supporting short-lived jobs
- There are special-purpose exporters for various container services, like StatsD, HAProxy, Graphite, etc.
- You will also get an alert manager for handling alerts on various support tools

2. How do we get POD's central logs?

You can use either of the logging patterns for getting central logs from the pod.

- Use a node-level logging agent
- Stream the sidecar container
- Use the sidecar container with the logging agent
- Export the logs directly from the application

3. What are the components of a Kubernetes node?

Node components run on each node where they manage pods and provide the runtime environment to Kubernetes. The following are the node components:

- **Kubelet:** Makes sure that the containers are running in a pod
- **Kube-proxy:** Maintains the desired network rules on nodes. These rules allow the network communication from sessions inside or outside of your Kubernetes cluster to your pods.
- **Container runtime:** This software runs containers. Kubernetes supports various container runtimes, such as Containerd, CRI-O, Docker, or any Kubernetes Container Runtime Interface (CRI) implementation.

4. How does Kubernetes make containerized deployment more manageable?

As a typical application would have a cluster of containers running across multiple hosts, all these containers would need to talk to each other. So, to do this you need something big that would load balance, scale & monitor the containers. Since Kubernetes is cloud-agnostic and can run on any public/private providers it must be your choice simplify containerized deployment.

5. What is the difference between Kubernetes and Docker Swarm?

Docker Swarm is Docker's native, open-source container orchestration platform that is used to cluster and schedule Docker containers. Swarm differs from Kubernetes in the following ways:

- Docker Swarm is more convenient to set up but doesn't have a robust cluster, while Kubernetes is more complicated to set up but the benefit of having the assurance of a robust cluster
- Docker Swarm can't do auto-scaling (as can Kubernetes); however, Docker scaling is five times faster than Kubernetes
- Docker Swarm doesn't have a GUI; Kubernetes has a GUI in the form of a dashboard
- Docker Swarm does automatic load balancing of traffic between containers in a cluster, while Kubernetes requires manual intervention for load balancing such traffic
- Docker requires third-party tools like ELK stack for logging and monitoring, while Kubernetes has integrated tools for the same
- Docker Swarm can share storage volumes with any container easily, while Kubernetes can only share storage volumes with containers in the same pod
- Docker can deploy rolling updates but can't deploy automatic rollbacks; Kubernetes can deploy rolling updates as well as automatic rollbacks.

6. What exactly does Kubernetes controller manager imply?

The controller manager is a daemon that is used for embedding core control loops, garbage collection, and Namespace creation. It enables the running of multiple processes on the master node even though they are compiled to run as a single process.