# Kubernetes Assignment 4

1. What are the benefits of having operators?

Kubernetes Operators makes it easier for third-party developers to build upon the Kubernetes platform, using custom controllers, sometimes called operators, and Custom Resource Definitions (CRDs). In simple words, Operators are Kubernetes native applications which can be controlled using Kubectl.

**Benefits of Using Operators:**

- Package Human Knowledge into Operations — With Operators, we can package, deploy and manage a Kubernetes application which takes human operational knowledge and encodes it into software that is more easily packaged and shared with consumers.
- Promotes Cloud Native Approach — Operators helps in building Cloud Native apps by delivering the automation advantages like deploying, scaling, and backup and restore while being able to run anywhere where Kubernetes is Deployed.
- CRUD — There is no need to apply basic operations on your app as we can use Kubernetes's Extensibility via HTTP and kubectl.
- Basic Operations — Clients like kubectl and dashboard automatically offer list, display, and field edit operations on your resources
- HTTPS — The resources created are Secured and uses HTTPS like any other Kubernetes APIs
- Auth — Same Authentication and Authorization — The resources created can use same Authentication and Authorization access to the extension uses the apiserver.

2. What exactly is GKE?

Google Container Engine (GKE) is an open-source management platform for Docker containers and clusters. This Kubernetes based engine supports only those clusters which run within Google's public cloud services.

3. What is the best way to operate Kubernetes locally?

Minikube is a tool that makes it easy to run Kubernetes locally. This runs a single-node Kubernetes cluster inside a virtual machine.

4. 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.

5. How does Kubernetes make containerized deployment easier?

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.