The IP of Your application

1. Briefly summarize what you have learned about docker and Kubernetes including the used terminologies and their descriptions.

Docker is a tool used to automate the deployment of applications in a lightweight container. Containers are software packages that contain all the dependencies required to run the application and can be maintained in isolated environments. Docker provides better and more consistent performance and is easier to set up and scale. Docker works through a client-server relationship, with the Docker engine installed on the host machine and a REST API used for communication between the client and server.

Kubernetes is an open-source platform for managing containerized workloads and services. It helps make sure each container is where it's supposed to be and that the containers can work together. Kubernetes automates most of the repetition and inefficiencies of managing containers, allowing app developers to tell Kubernetes what they want the cluster to look like, and Kubernetes makes it happen. Kubernetes is a portable, extensible platform that can be used on different cloud providers and on-premise.

Some commonly used terminologies in **Docker** include:

- Container: A lightweight, portable, and executable package of software that includes everything needed to run an application.
- Dockerfile: A script that contains instructions for building a Docker image.
- Image: A read-only template that contains the instructions for creating a container.
- Registry: A storage and distribution system for Docker images.
- Repository: A collection of Docker images.

Some commonly used terminologies in **Kubernetes** include:

- Pod: The smallest deployable unit in Kubernetes that represents a single instance of a running process.
- ReplicaSet: A group of pods that are identical and can be scaled up or down as needed.
- Deployment: A higher-level object that manages a ReplicaSet and provides declarative updates to the pods.
- Service: An object that provides network connectivity to a set of pods.
- Node: A physical or virtual machine that runs one or more pods.

2. What are the advantages and disadvantages of using docker images?

Advantages of using Docker images include their portability, consistency, and isolation. Docker images can be easily moved between different environments, ensuring consistency across development, testing, and production environments. Containers provide isolation between applications and the host operating system, preventing conflicts with other applications or dependencies. Docker images are also easy to create, distribute, and update, making it easier for developers to deploy and maintain their applications.

Disadvantages of using Docker images include the potential for security vulnerabilities, performance overhead, and complexity. As with any software, Docker images can have security vulnerabilities that could be exploited by attackers. Running applications in containers can introduce some performance overhead compared to running them natively on the host operating system. Finally, Docker images and containers add complexity to the deployment and management of applications, which can be challenging for some teams to manage.

3. Instructions you used to create and deploy your application.

Method 1:

```
cd ~/SOFE3980U-Lab3/BinaryCalculatorWebapp
mvn clean
mvn package

docker build -t gcr.io/<Project-ID>/binarycalculator .
docker push gcr.io/<Project-ID>/binarycalculator

kubectl create deployment binarycalculator-deployment --image
gcr.io/<Project-ID\>/binarycalculator --port=8080

kubectl expose deployment binarycalculator-deployment --type=LoadBalancer
--name=binarycalculator-service
```

Method 2:

```
cd ~/SOFE3980U-Lab3/BinaryCalculatorWebapp
mvn clean
mvn package

docker build -t gcr.io/<Project-ID>/binarycalculator .
docker push gcr.io/<Project-ID>/binarycalculator

cd ~/SOFE3980U-Lab3/MySQL
kubectl delete -f mysql-deploy.yaml
```