Technical task

1.In this task I have just created the ec2 instance by using terraform and I have attached below, and in this ec2 instance I have installed docker, Minikube by using following commands.

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg

echo \

"deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu \

$(lsb\_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

sudo apt-get update

sudo apt-get install docker-ce docker-ce-cli containerd.io

curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube\_latest\_amd64.deb

sudo dpkg -i minikube\_latest\_amd64.deb

minikube start --nodes 2 -p multinode-demo

curl -LO https://dl.k8s.io/release/v1.23.0/bin/linux/amd64/kubectl

udo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl

kubectl version –client

2.After that I set up the Node.js application by creating the file index.js, by using the command node index.js the server will start.

3. after the second step I set up the express package setup by creating the file server.js

4. now I have created Dockerfile to dockerizing the RESTful Node.js Express application.

This Dockerfile uses npm to install modules in our RESTful application so I set up the docker-compose configuration file that we’ll use to launch the Node.js Express application

After this I Setup YAML service to deploy Dockerized Node.js Express application by creating the file service,yaml

After this I have created the new file called deployment.yaml. This file will deploy the application to the Kubernetes engine.

1. Service - The service acts as the load balancer. A load balancer is used to distribute requests to the various available servers.
2. Deployment will act as the intended application. The user request hits the load balancer, then the load balancer distributes the request by creating the number of replicas defined in the deployment.yaml file. For example, in our case, we have five replicas for scalability, meaning that we will have 5 instances running at a time.
3. The deployment.yaml file is connected to the Docker image created earlier, therefore, to deploy the application to the Kubernetes cluster, we use the Docker image. The image will automatically create containers for the application when we deploy the application.