Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Department of Computer Science and Engineering (Data Science)

NAME: Kresha Shah SAP ID: 60009220080

Subject: Big Data Engineering (DJ19DSL604)

AY: 2023-24

Experiment 2 (Infrastructure Implementation)

Aim: Hands on experience of Dockers and Kubernetes**.**

Theory:

Docker Overview

Docker is an open platform for developing, shipping, and running applications. Docker enables you to separate your applications from your infrastructure so you can deliver software quickly. With Docker, you can manage your infrastructure in the same ways you manage your applications. By taking advantage of Docker's methodologies for shipping, testing, and deploying code quickly, you can significantly reduce the delay between writing code and running it in production.

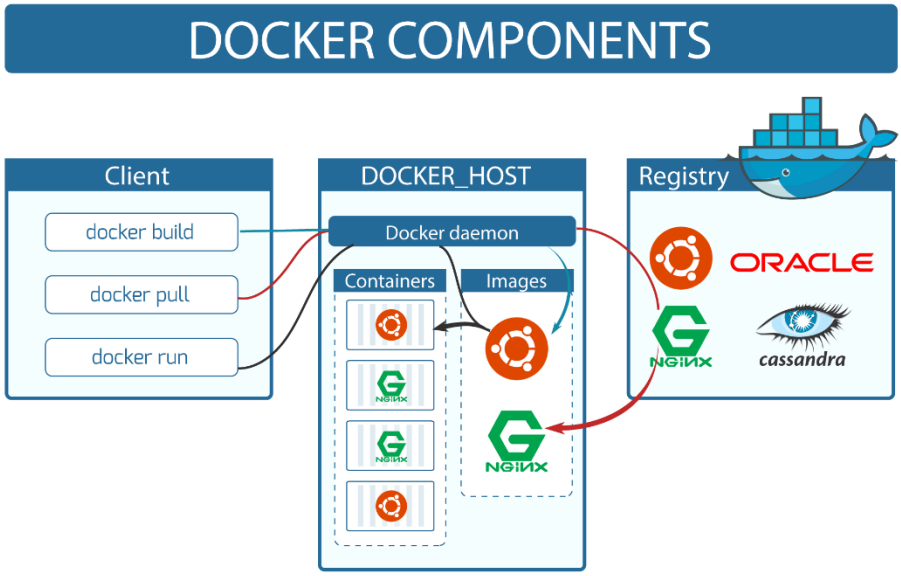
The Docker Platform

Docker provides the ability to package and run an application in a loosely isolated environment called a container. The isolation and security allows you to run many containers simultaneously on a given host. Containers are lightweight and contain everything needed to run the application, so you do not need to rely on what is currently installed on the host. You can easily share containers while you work, and be sure that everyone you share with gets the same container that works in the same way.

Docker provides tooling and a platform to manage the lifecycle of your containers:

- Develop your application and its supporting components using containers.

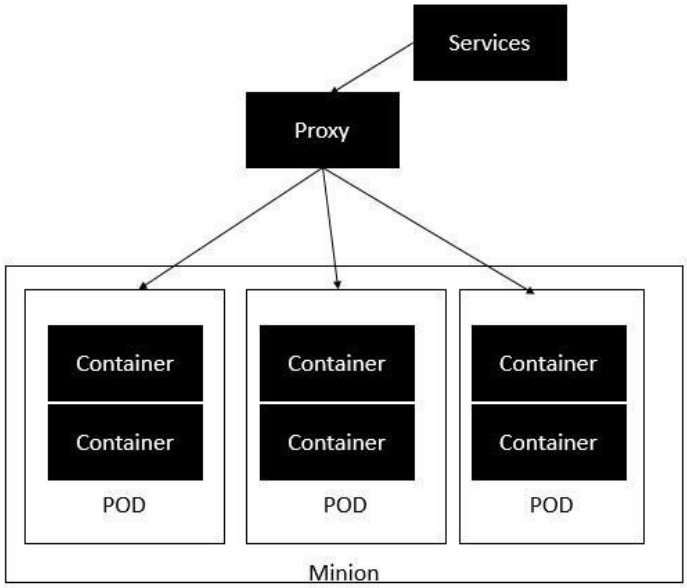
- The container becomes the unit for distributing and testing your application.
- When you're ready, deploy your application into your production environment, as a container or an orchestrated service. This works the same whether your production environment is a local data center, a cloud provider, or a hybrid of the two.



Kubernetes

Kubernetes is an orchestration framework for Docker containers which helps expose containers as services to the outside world. For example, you can have two services − One service would contain nginx and mongoDB, and another service would contain nginx and redis. Each service can have an IP or service point which can be connected by other applications. Kubernetes is then used to manage these services.

The following diagram shows in a simplistic format how Kubernetes works from an architecture point of view.



The minion is the node on which all the services run. You can have many minions running at one point in time. Each minion will host one or more POD. Each POD is like hosting a service. Each POD then contains the Docker containers. Each POD can host a different set of Docker containers. The proxy is then used to control the exposing of these services to the outside world.

Kubernetes has several components in its architecture. The role of each component is explained below &mius;

etcd − This component is a highly available key-value store that is used for storing shared configuration and service discovery. Here the various applications will be able to connect to the services via the discovery service.

Flannel − This is a backend network which is required for the containers.

kube-apiserver − This is an API which can be used to orchestrate the Docker containers.

kube-controller-manager − This is used to control the Kubernetes services. kube-scheduler

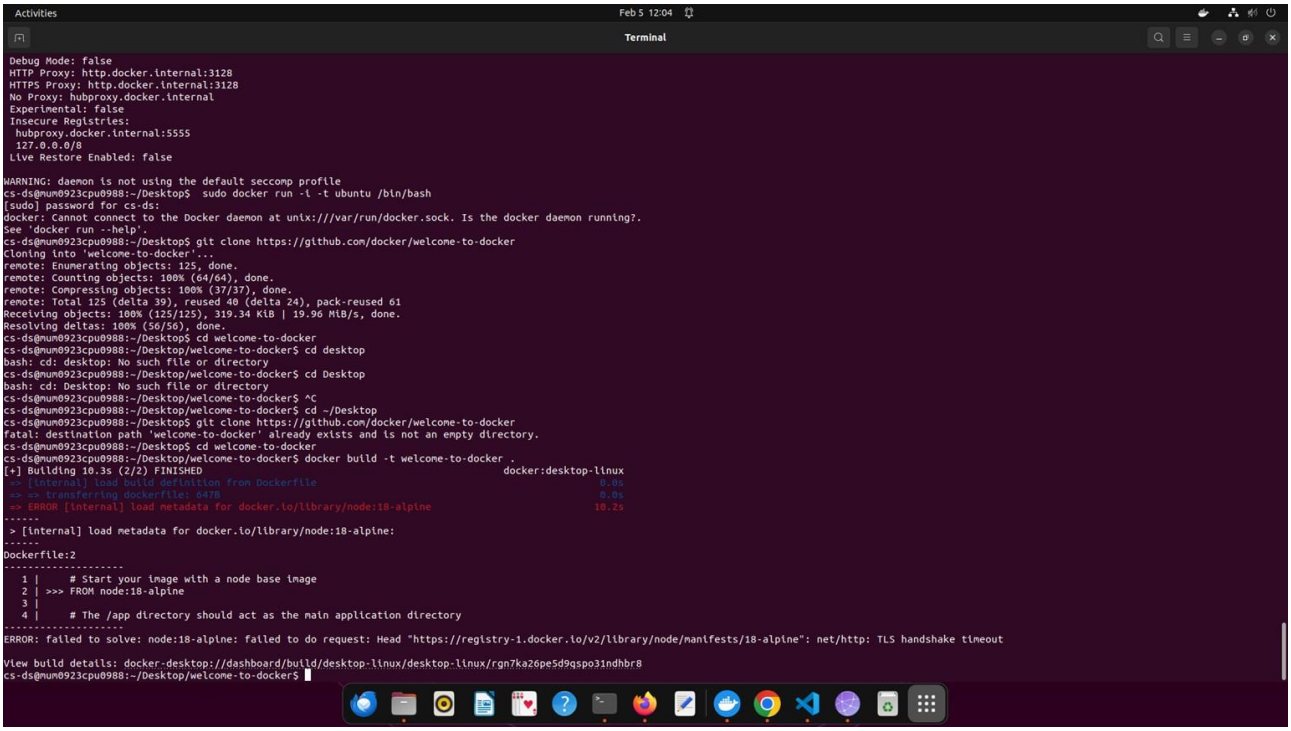
− This is used to schedule the containers on hosts.

Kubelet − This is used to control the launching of containers via manifest files. kube-proxy

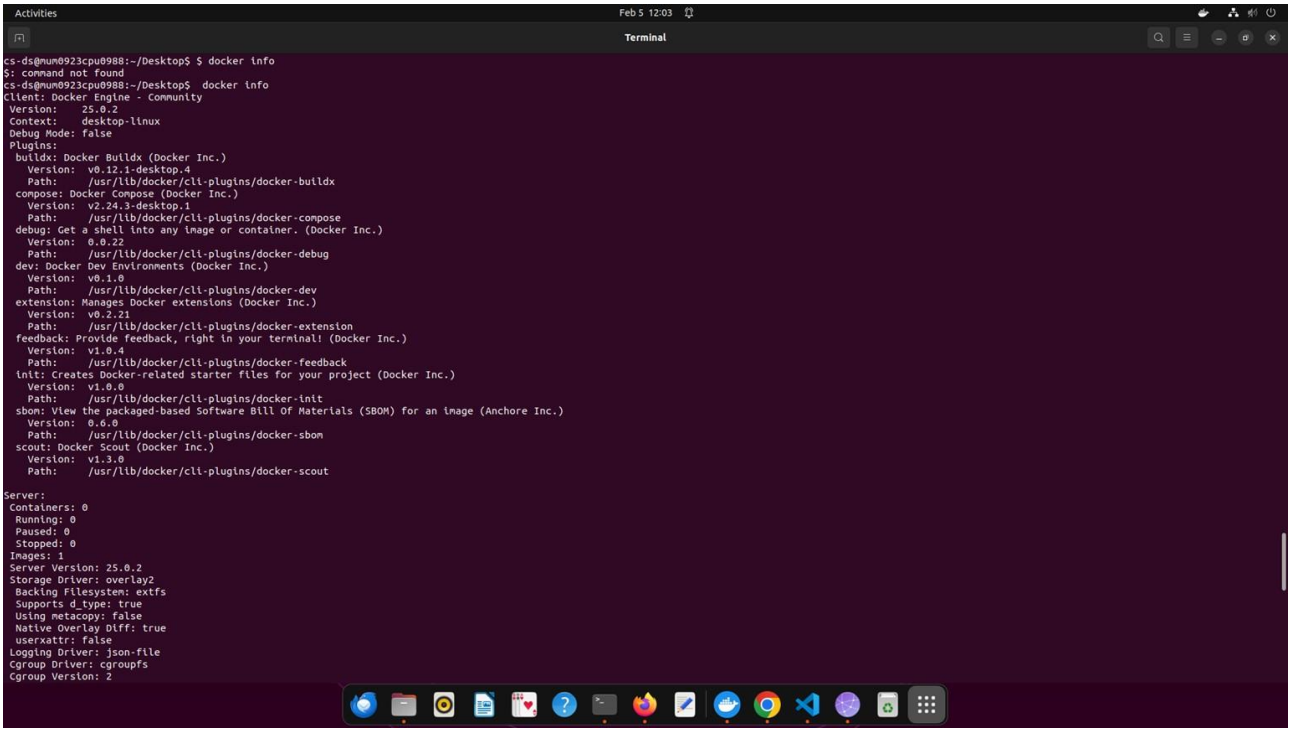
− This is used to provide network proxy services to the outside world.

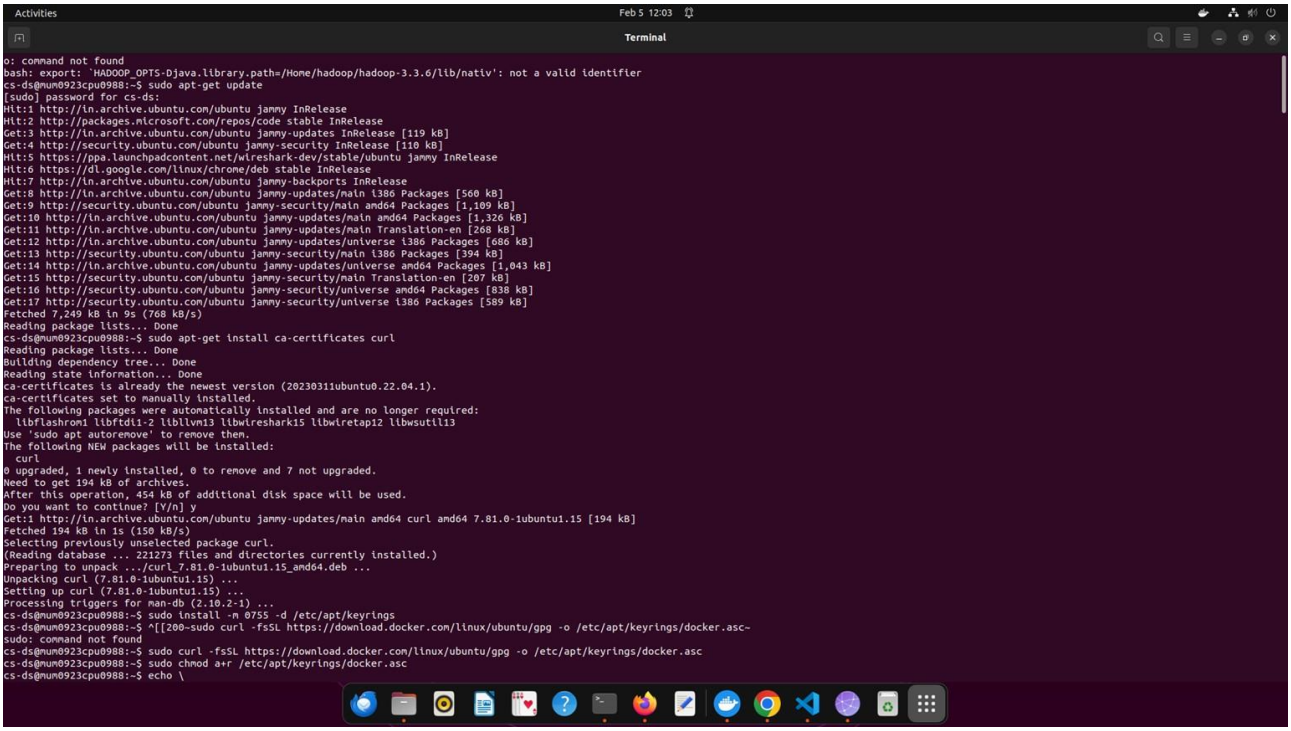
Lab Assignment:

1. Installation of Docker and Kubernetes on Ubuntu 20.4.5.

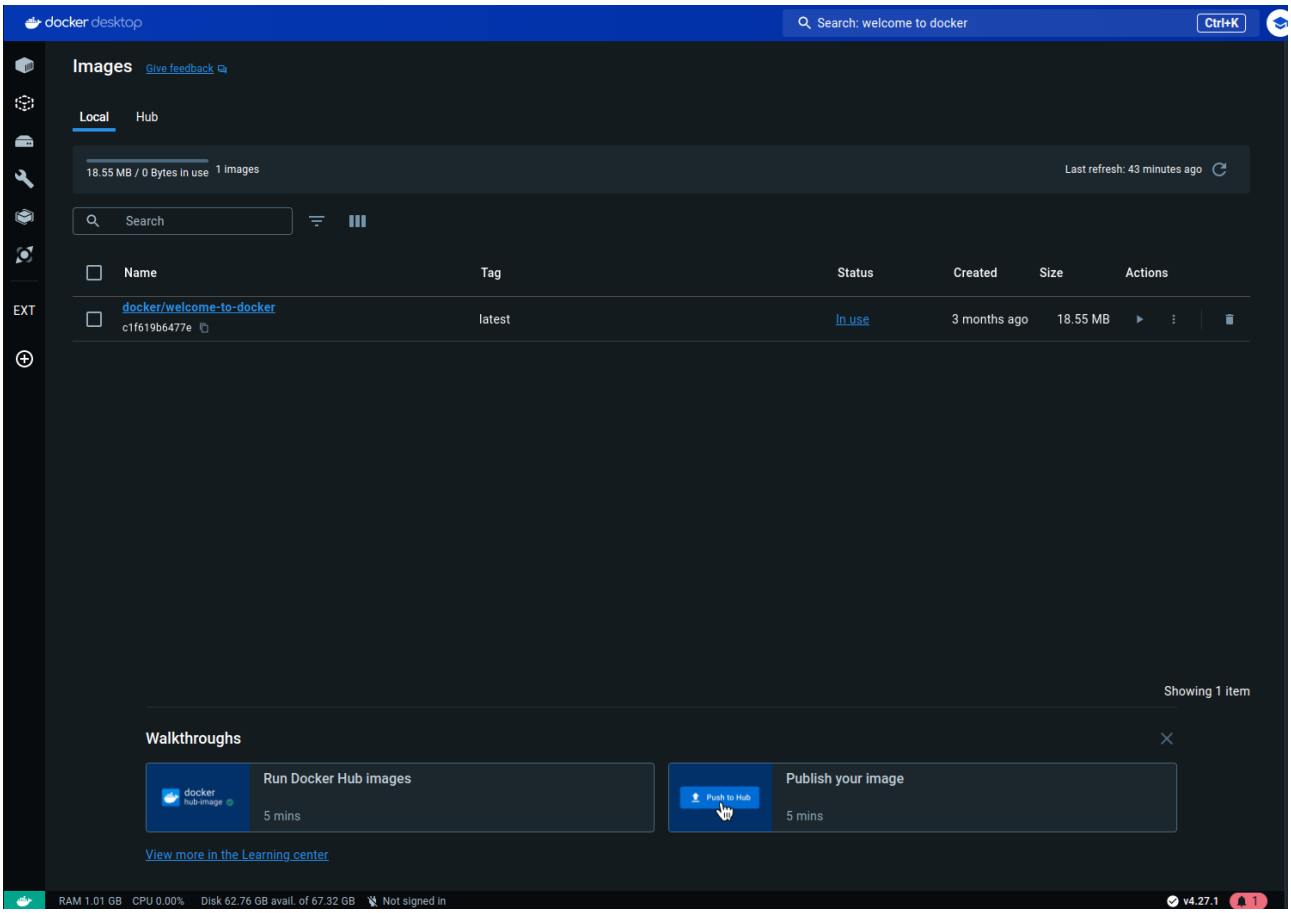


Department of Computer Science and Engineering (Data Science)

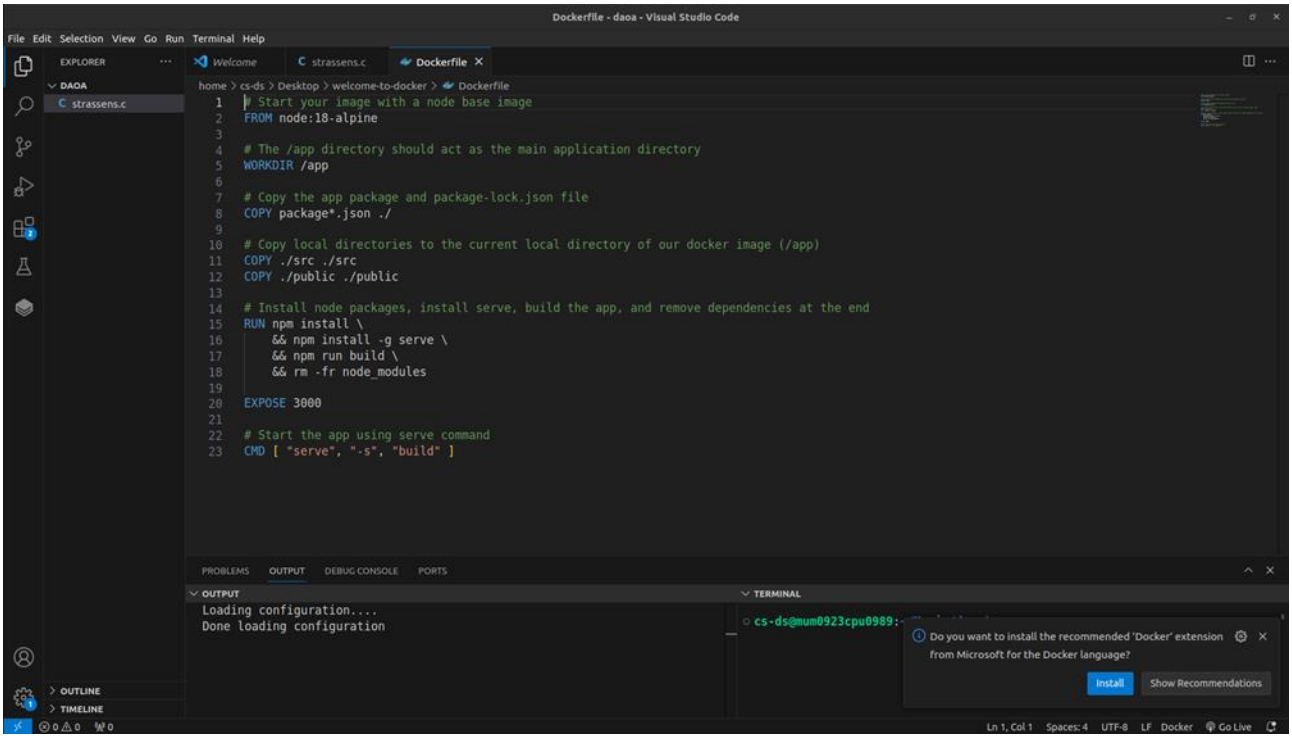


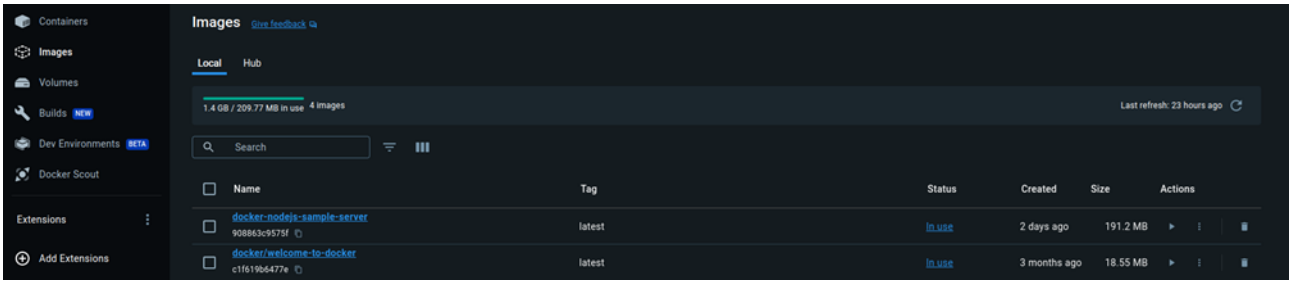


2. Group Activity (3 students): Create two Docker containers. One with an OS image and another with Node.js and React.



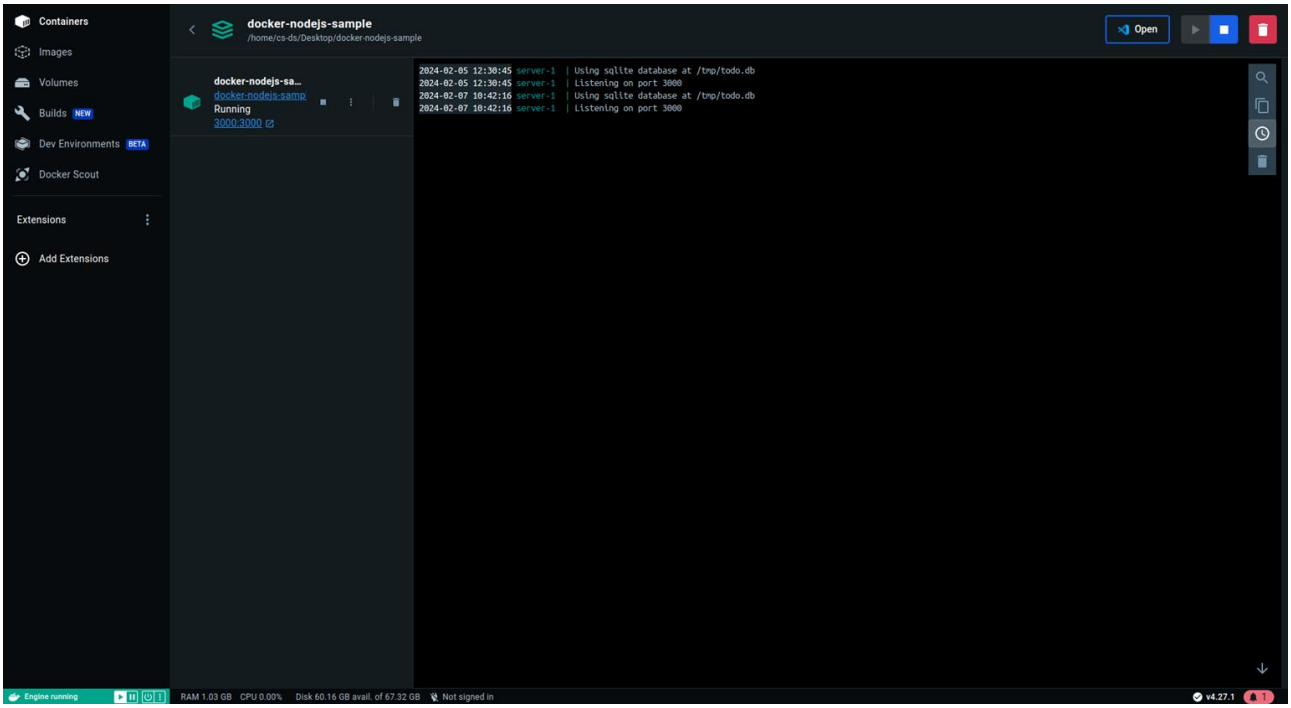
Department of Computer Science and Engineering (Data Science)



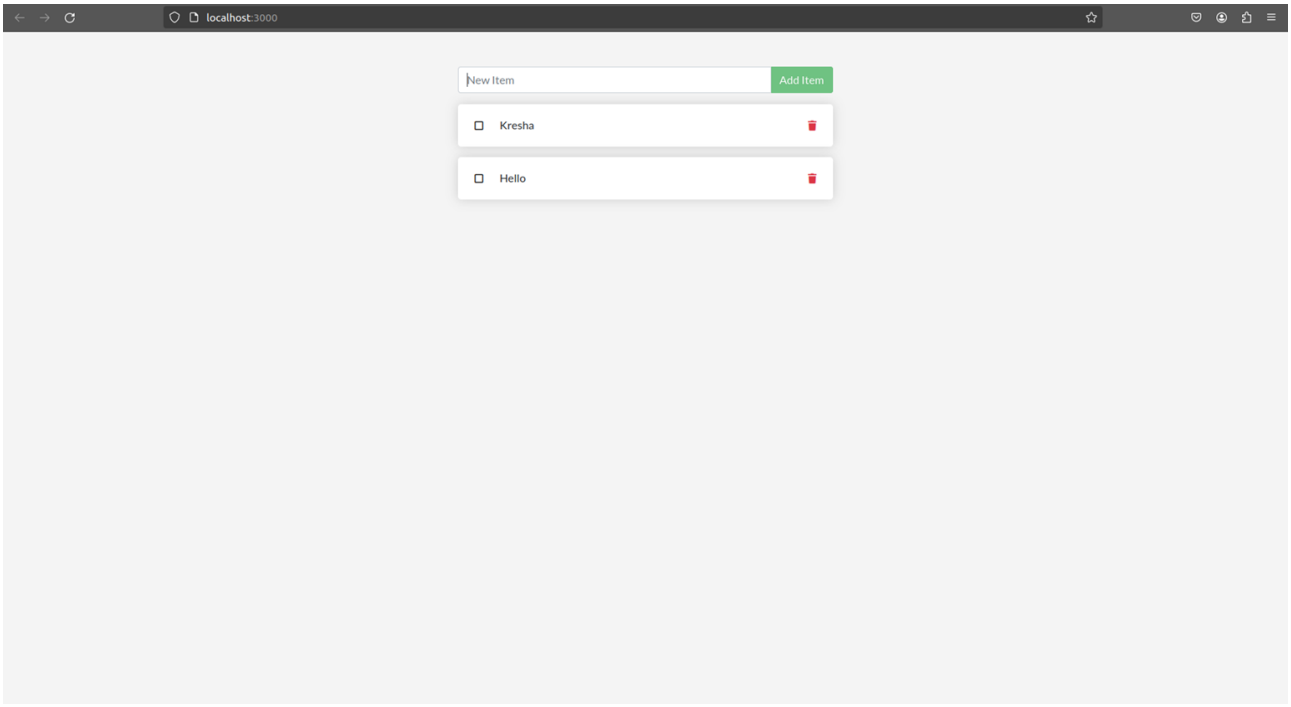


Department of Computer Science and Engineering (Data Science)

```
cs-ds@mum0923cpu0989:~/Desktop/docker-nodejs-sample$ docker init

Welcome to the Docker Init CLI!

This utility will walk you through creating the following files with sensible defaults for your project:
  - .dockerignore
  - Dockerfile
  - compose.yaml
  - README.Docker.md

Let's get started!

? What application platform does your project use? Node
? What version of Node do you want to use? 18.0.0
? Which package manager do you want to use? npm
? What command do you want to use to start the app? node src/index.js
? What port does your server listen on? 3000

CREATED: .dockerignore
CREATED: Dockerfile
CREATED: compose.yaml
CREATED: README.Docker.md

✔ Your Docker files are ready!

Take a moment to review them and tailor them to your application.

When you're ready, start your application by running: docker compose up --build

Your application will be available at http://localhost:3000

Consult README.Docker.md for more information about using the generated files.
cs-ds@mum0923cpu0989:~/Desktop/docker-nodejs-sample$ docker compose up --build
[+] Building 113.4s (6/10)
 => [server internal] load build definition from Dockerfile
 => => transferring dockerfile: 1.21kB
 => [server] resolve image config for docker.io/docker/dockerfile:1
 => [server] docker-image://docker.io/docker/dockerfile:1@sha256:ac85f380a63b13dfcefa89046420e1781752bab202122f8f50032edf31be0021
 => => resolve docker.io/docker/dockerfile:1@sha256:ac85f380a63b13dfcefa89046420e1781752bab202122f8f50032edf31be0021
 => => sha256:ac85f380a63b13dfcefa89046420e1781752bab202122f8f50032edf31be0021 8.40kB / 8.40kB
 => => sha256:657fcc512c7369f4cb3d94ea329150f8daf626bc838b1a1e81f1834c73ecc77e 482B / 482B
 => => sha256:a17ee7fff8f5e97b974f5b48f51647d2cf28d543f2aa6c11aaa0ea431b44bb89 1.27kB / 1.27kB
 => => sha256:9d9c93f4b00be908ab694a4df732570bced3b8a96b7515d70ff93402179ad232 11.80MB / 11.80MB
 => => extracting sha256:9d9c93f4b00be908ab694a4df732570bced3b8a96b7515d70ff93402179ad232
 => [server internal] load metadata for docker.io/library/node:18.0.0-alpine
 => [server internal] load .dockerignore
 => => transferring context: 659B
 => [server stage-0 1/4] FROM docker.io/library/node:18.0.0-alpine@sha256:469ee26d9e00547ea91202a34ff2542f984c2c60a2edbb4007558ccb76b56df2
 => => resolve docker.io/library/node:18.0.0-alpine@sha256:469ee26d9e00547ea91202a34ff2542f984c2c60a2edbb4007558ccb76b56df2
 => => sha256:469ee26d9e00547ea91202a34ff2542f984c2c60a2edbb4007558ccb76b56df2 1.43kB / 1.43kB
 => => sha256:77f86133ced42635d78a2ca2ab27945f8ac9aa06ef092f52c84bb9b2a13055fc 1.16kB / 1.16kB
 => => sha256:de1a9de7d955550de3a7caaa4f6a5085b72b17ec272dc56c13b3a2becba4cab3 6.58kB / 6.58kB
 => => sha256:df9b9388f04ad6279a7410b85cedfdcb2208c0a003da7ab5613af71079148139 1.05MB / 2.81MB
```



Department of Computer Science and Engineering (Data Science)

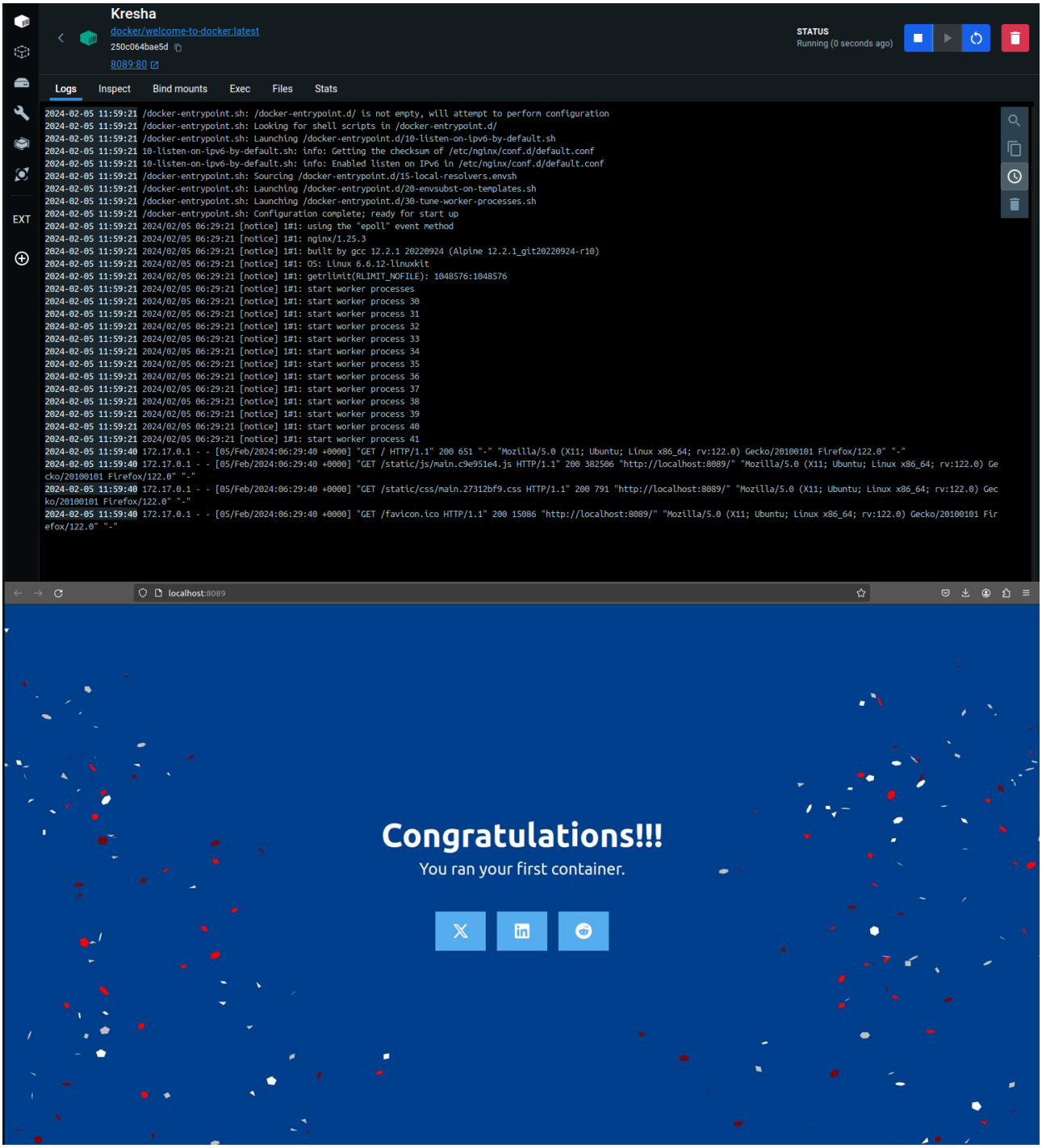


3. Develop a website in one Docker Container, share is with your team mates and allow parallel corrections from all team members.

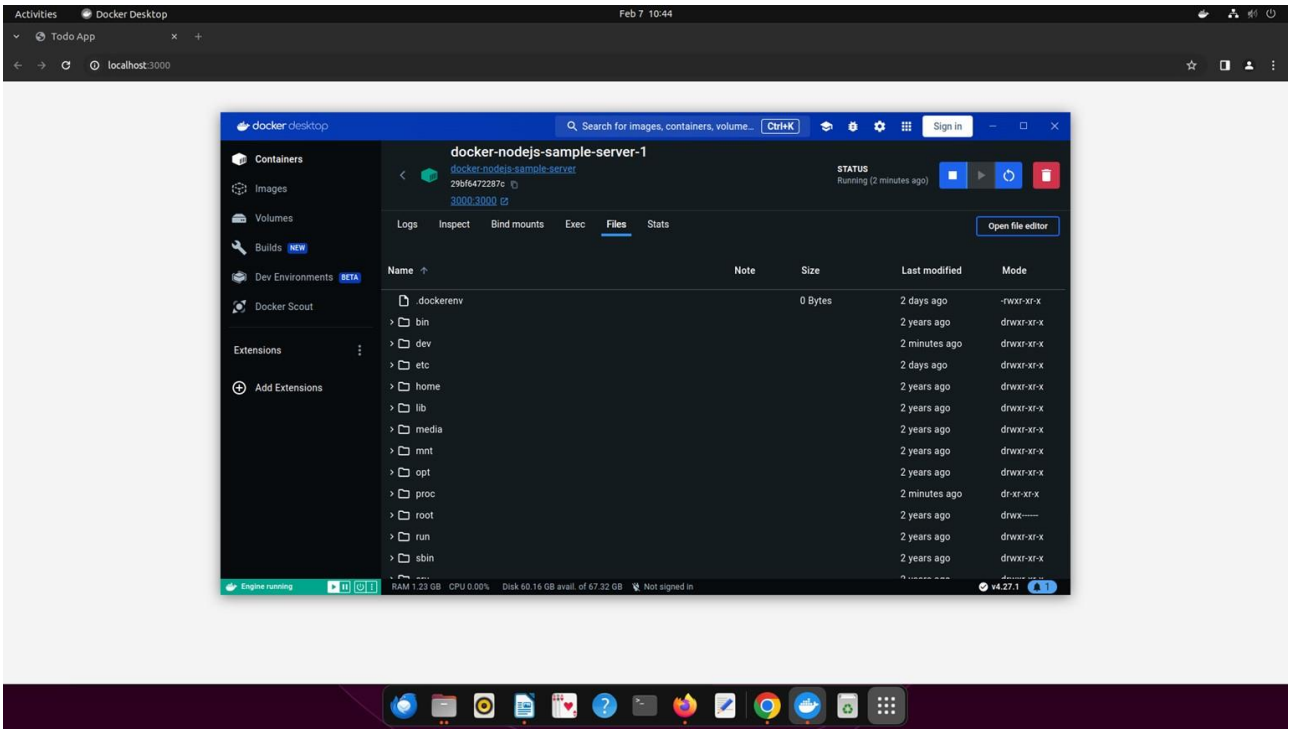
```
cs-ds@num0923cpu0989:~/Desktop$ git clone https://github.com/docker/welcome-to-docker
Cloning into 'welcome-to-docker'...
remote: Enumerating objects: 125, done.
remote: Counting objects: 100% (64/64), done.
remote: Compressing objects: 100% (37/37), done.
remote: Total 125 (delta 39), reused 40 (delta 24), pack-reused 61
Receiving objects: 100% (125/125), 319.34 KiB | 1.60 MiB/s, done.
Resolving deltas: 100% (56/56), done.
cs-ds@num0923cpu0989:~/Desktop$ cd welcome-to-docker
cs-ds@num0923cpu0989:~/Desktop/welcome-to-docker$ docker build -t welcome-to-docker .
[+] Building 24.7s (2/2) FINISHED                                docker:desktop-linux
 => [internal] load build definition from Dockerfile                              0.0s
 => => transferring dockerfile: 647B                                              0.0s
 => ERROR [internal] load metadata for docker.io/library/node:18-alpine          24.6s
------
 > [internal] load metadata for docker.io/library/node:18-alpine:
------
Dockerfile:2
--------------------
   1 |     # Start your image with a node base image
   2 | >>> FROM node:18-alpine
   3 |
   4 |     # The /app directory should act as the main application directory
--------------------
ERROR: failed to solve: node:18-alpine: failed to do request: Head "https://registry-1.docker.io/v2/library/node/manifests/18-alpine": net/http: TLS handshake timeou

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/y9qalbrxclhy5vbehpmx7ip8u
cs-ds@num0923cpu0989:~/Desktop/welcome-to-docker$
```

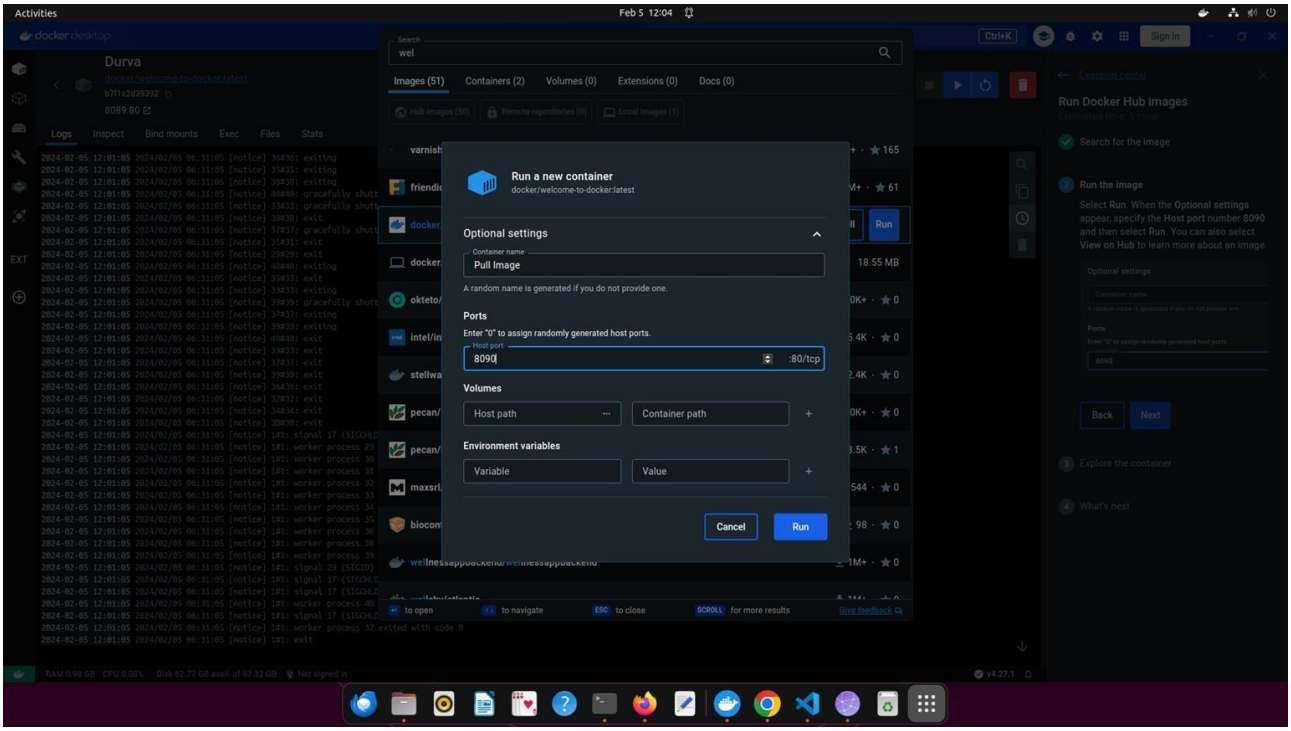
```
cs-ds@num0923cpu0989:~/Desktop/docker-nodejs-sample$ docker init

Welcome to the Docker Init CLI!

This utility will walk you through creating the following files with sensible defaults for your project:
  - .dockerignore
  - Dockerfile
  - compose.yaml
  - README.Docker.md

Let's get started!

? What application platform does your project use? Node
? What version of Node do you want to use? 18.0.0
? Which package manager do you want to use? npm
? What command do you want to use to start the app? node src/index.js
? What port does your server listen on? 3000

CREATED: .dockerignore
CREATED: Dockerfile
CREATED: compose.yaml
CREATED: README.Docker.md

✔ Your Docker files are ready!

Take a moment to review them and tailor them to your application.

When you're ready, start your application by running: docker compose up --build

Your application will be available at http://localhost:3000

Consult README.Docker.md for more information about using the generated files.
cs-ds@num0923cpu0989:~/Desktop/docker-nodejs-sample$ docker compose up --build
[+] Building 113.4s (6/10)
 => [server internal] load build definition from Dockerfile
 => => transferring dockerfile: 1.21kB
 => [server] resolve image config for docker.io/docker/dockerfile:1
 => [server] docker-image://docker.io/docker/dockerfile:1@sha256:ac85f380a63b13dfcefa89046420e1781752bab202122f8f50032edf31be0021
 => => resolve docker.io/docker/dockerfile:1@sha256:ac85f380a63b13dfcefa89046420e1781752bab202122f8f50032edf31be0021
 => => sha256:ac85f380a63b13dfcefa89046420e1781752bab202122f8f50032edf31be0021 8.40kB / 8.40kB
 => => sha256:657fcc512c7369f4cb3d94ea329150f8daf626bc838b1a1e81f1834c73ecc77e 482B / 482B
 => => sha256:a17ee7fff8f5e97b974f5b48f51647d2cf28d543f2aa6c11aaa0ea431b44bb89 1.27kB / 1.27kB
 => => sha256:9d9c93f4b00be908ab694a4df732570bced3b8a96b7515d70ff93402179ad232 11.80MB / 11.80MB
 => => extracting sha256:9d9c93f4b00be908ab694a4df732570bced3b8a96b7515d70ff93402179ad232
 => [server internal] load metadata for docker.io/library/node:18.0.0-alpine
 => [server internal] load .dockerignore
 => => transferring context: 659B
 => [server stage-0 1/4] FROM docker.io/library/node:18.0.0-alpine@sha256:469ee26d9e00547ea91202a34ff2542f984c2c60a2edbb4007558ccb76b56df2
 => => resolve docker.io/library/node:18.0.0-alpine@sha256:469ee26d9e00547ea91202a34ff2542f984c2c60a2edbb4007558ccb76b56df2
 => => sha256:469ee26d9e00547ea91202a34ff2542f984c2c60a2edbb4007558ccb76b56df2 1.43kB / 1.43kB
 => => sha256:77f86133ced42035d78a2ca2ab27945f8ac9aa06ef092f52c84bb9b2a13055fc 1.16kB / 1.16kB
 => => sha256:de1a9de7d955550de3a7caaa4f6a5085b72b17ec272dc56c13b3a2becba4cab3 6.58kB / 6.58kB
 => => sha256:df9b9388f04ad6279a7410b85cedfdcb2208c0a003da7ab5613af71079148139 1.05MB / 2.81MB
```

Department of Computer Science and Engineering (Data Science)

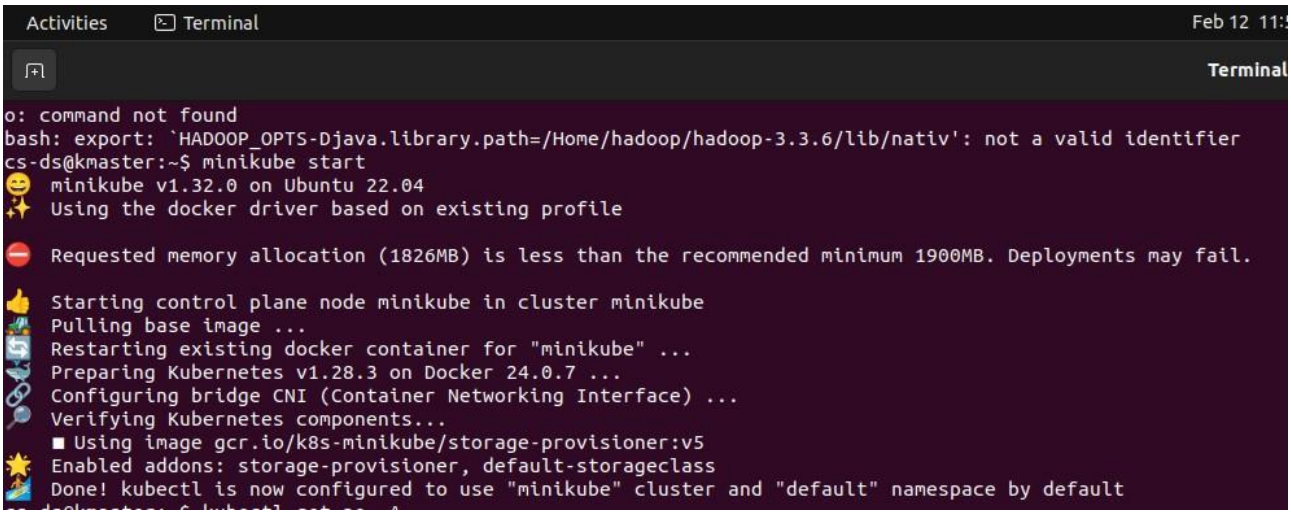


Department of Computer Science and Engineering (Data Science)





4. Connect both the Dockers through Kubernetes and show how new request is handled.



Department of Computer Science and Engineering (Data Science)