# **TASK-TIDE**

# Website for Google keep

# **TEAM MEMBERS:**

- Sandeep K 21ADL065
- Sankar V 21ADL066

# **DOCKER**

Docker is a platform that allows developers to build package, and run applications in lightweight, portable containers. It's used to ensure consistent environments across different systems, enabling easier deployment and scalability.

**STEP 1:** Listing the files in the directory to confirm the presence of essential files like docker-compose.yml and Dockerfile

Command: Is

**STEP 2:** Attempting to start a MongoDB container. If the image isn't found locally, Docker will download it.

Command: docker run -d --name mongodb -p 27017:27017 mongo

```
Anaconda-2020.11-Linux-x86_64.sh

pple

Anaconda-2020.11-Linux-x86_64.sh

pple

Anaconda-2020.11-Linux-x86_64.sh

pple

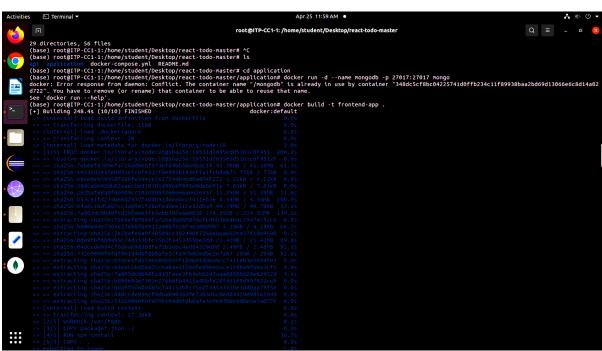
Anaconda-2020.11-Linux-x86_64.sh

Anaconda-2020.11-Linux
```

#### STEP 3:

- Building the frontend Docker image.
- Navigating to the appropriate directory and using the Dockerfile to create the image.

### Command: docker build -t frontend-app.



**STEP 4:** Running the frontend application in Docker, mapping port **8080** on the host to port **3000** in the container

Command: docker run -d -p 8080:3000 frontend-app



**STEP 5**: Checking Docker Network Configuration

Command: docker network inspect bridge



**STEP 6:** Building the Docker image for the backend application.

Command: docker build -t backend-app.



STEP7:Running the backend application in Docker, mapping port 3002 on the host to port 3333 in the container

Command: docker run -d -p 3002:3333 backend-app

STEP 8: Listing the running Docker containers to confirm everything is running properly.

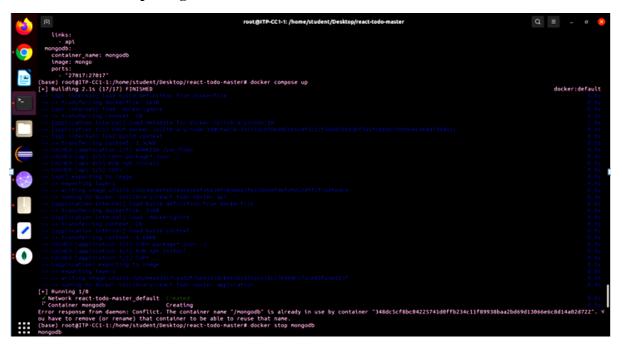
Command: docker ps



# **STEP 9:** Handling the Docker Container Name Conflict

• If there's a conflict with an existing container name, stopping the conflicting container.

## Command: docker stop mongodb

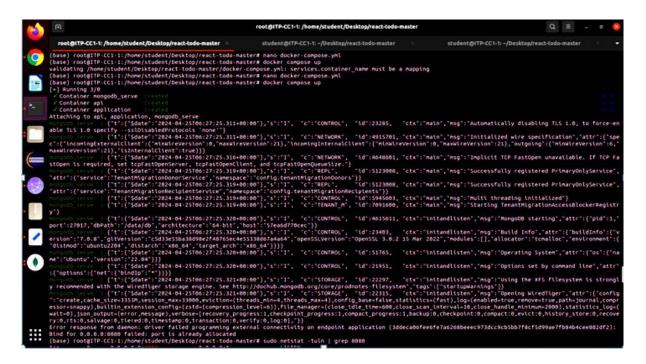


#### STEP 10: Building Docker Services

• Ensuring whether the Docker Compose setup is correct, and then building the services specified in the docker-compose.yml file

#### Command: docker compose up

```
(base) root@ITP.CC::I:/home/student/Desktop/react-todo-master# mano docker-compose.yml
(base) root@ITP.CC::I:/home/student/Desktop/react-todo-master# docker-compose.yml
(container manogod)_serve
(con
```





## **KUBERNETES**

Kubernetes is an orchestration system for managing containerized applications, providing tools for scaling, balancing loads, and automating deployment and recovery.

STEP 11: Starting Minikube to set up a local Kubernetes cluster.

Command: minikube start

## **STEP 12:** Applying Kubernetes Deployments

• Deploying Kubernetes resources using kubectl apply with the respective deployment files.

#### Command:

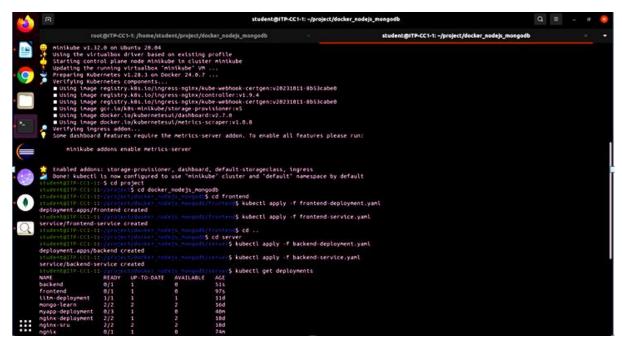
- kubectl apply -f frontend-deployment.yaml
- kubectl apply -f backend-deployment.yaml

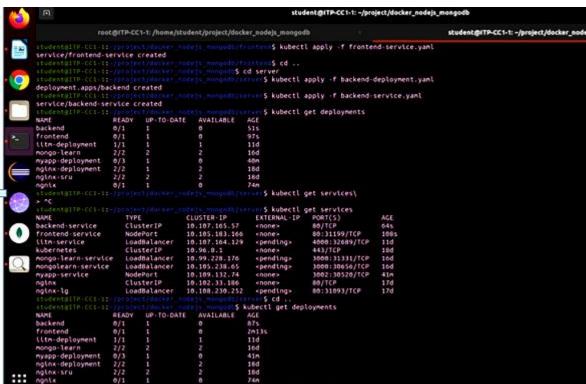
## **STEP 13:** Applying Kubernetes Services

• Similarly, applying the corresponding service configurations to expose the deployments.

#### Command:

- kubectl apply -f frontend-service.yaml
- kubectl apply -f backend-service.yaml





**STEP 14:** Verifying Kubernetes Deployments and Services

- Checking the status of the deployments and services to ensure that they are running as expected.
- List Deployments This command shows the readiness, availability, and age of each deployment. Look for READY, UP-TO-DATE, and AVAILABLE fields to assess their status.

# Command: kubectl get deployments

• List Services - This command lists the services and their associated ports, cluster IPs, and external IPs. Check for any pending external IPs or port conflicts.

## Command: kubectl get services

## **STEP 15:** Monitoring Minikube and Kubernetes Components

• Ensuring whether the Minikube and the Kubernetes components are running smoothly by checking the cluster status.

## Command: kubectl get nodes -o wide

This command provides information about the nodes in your Kubernetes cluster, including their internal IP, external IP, and other configurations.

**STEP 16:** Creating Ingress Resource

Applying an Ingress resource to route external traffic to the services.

# Command: kubectl apply -f ingress.yaml

```
*** student@ITP-CCI-1:-/project/docker_nodejs_nongodb$ kubectl apply-fingress.yaml ingress.networking.k8s.io/todo-task created
```

**STEP 17:** Displaying a list of services running in a Minikube Kubernetes cluster.

Command: minkube service list

# STEP 18: Mapped domain names to specific IP addresses can be viewed

# Command: sudo gedit /etc/hosts



