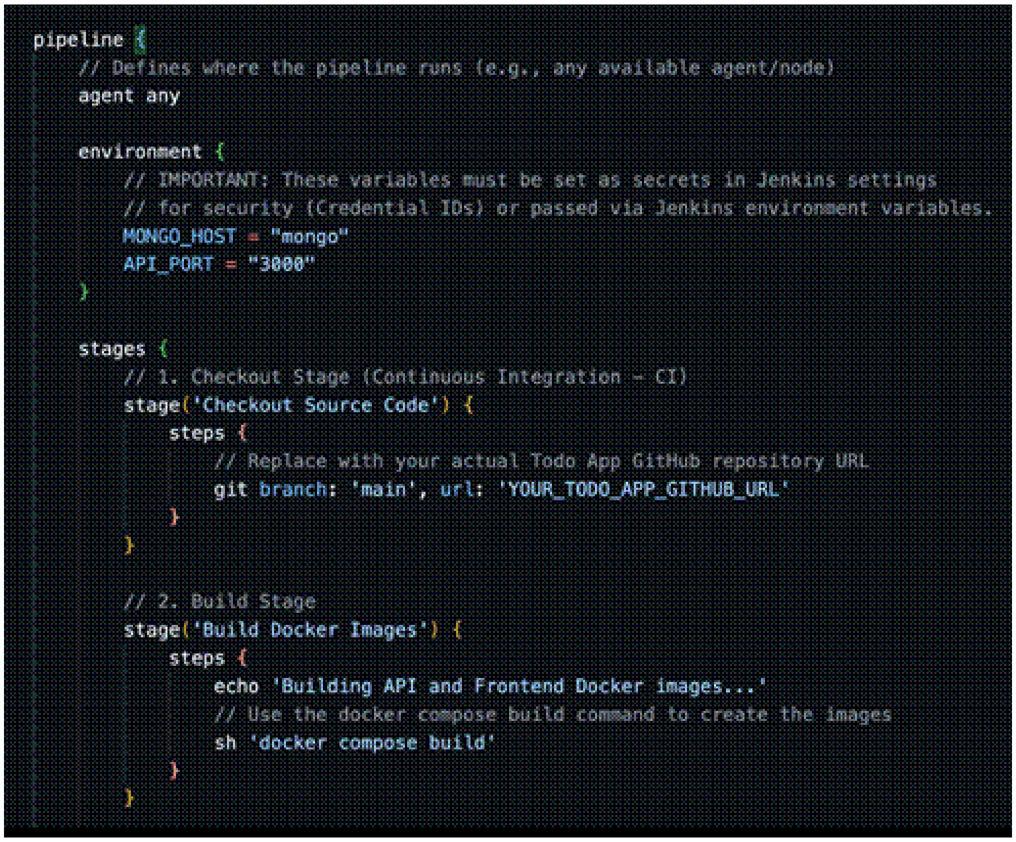
Taniya Nagdev **D15B / 50**

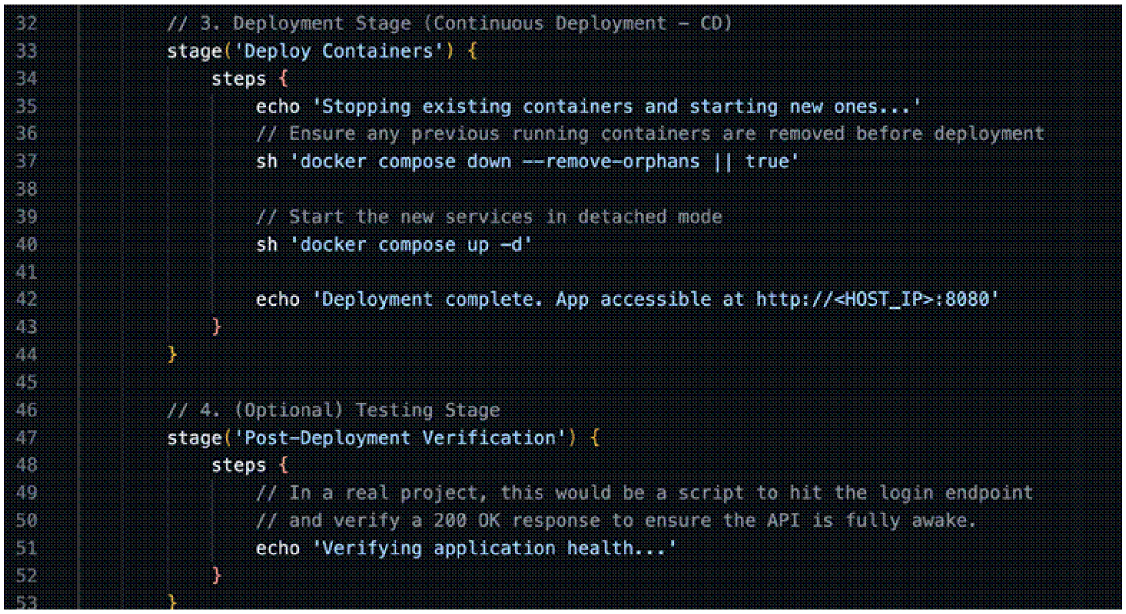
Experiment 10

Source code:

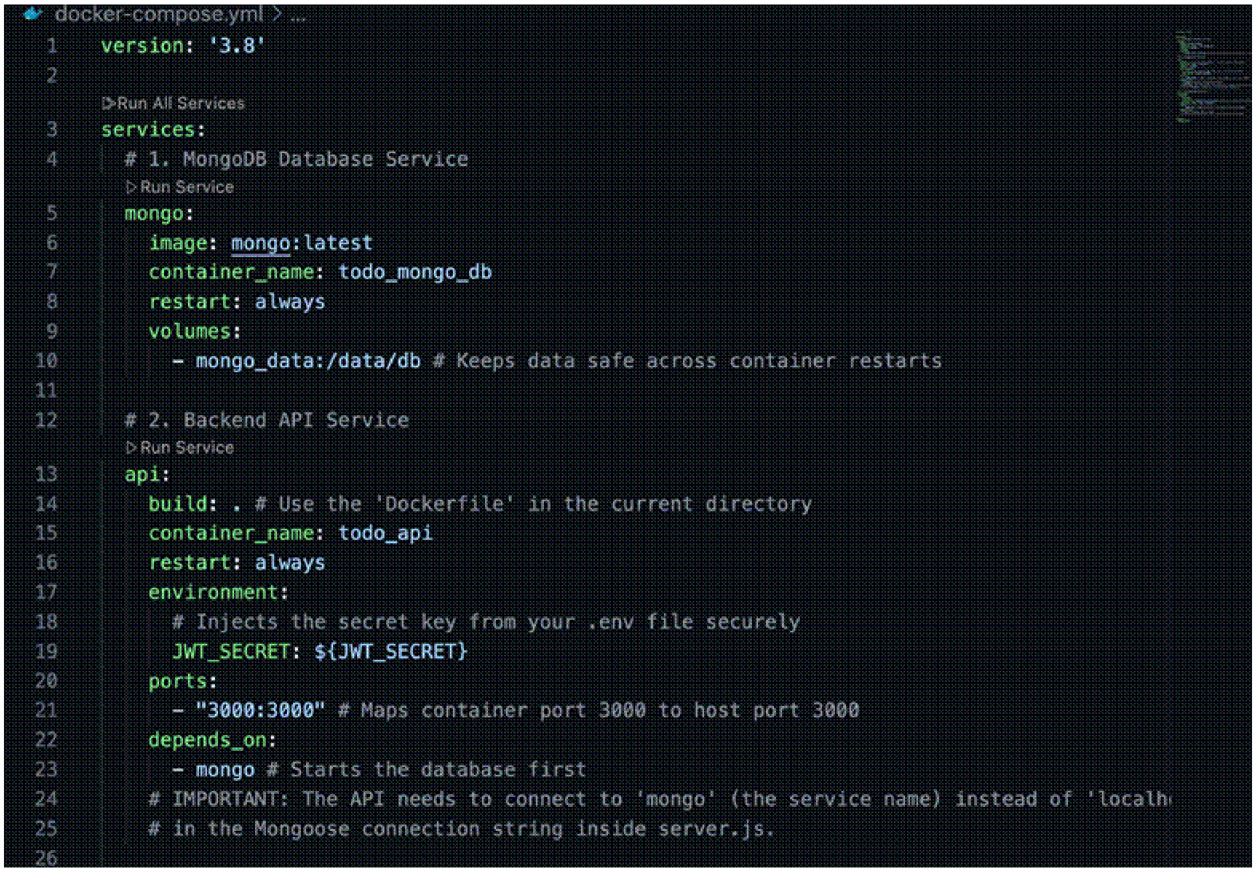
Jenkins pipeline :



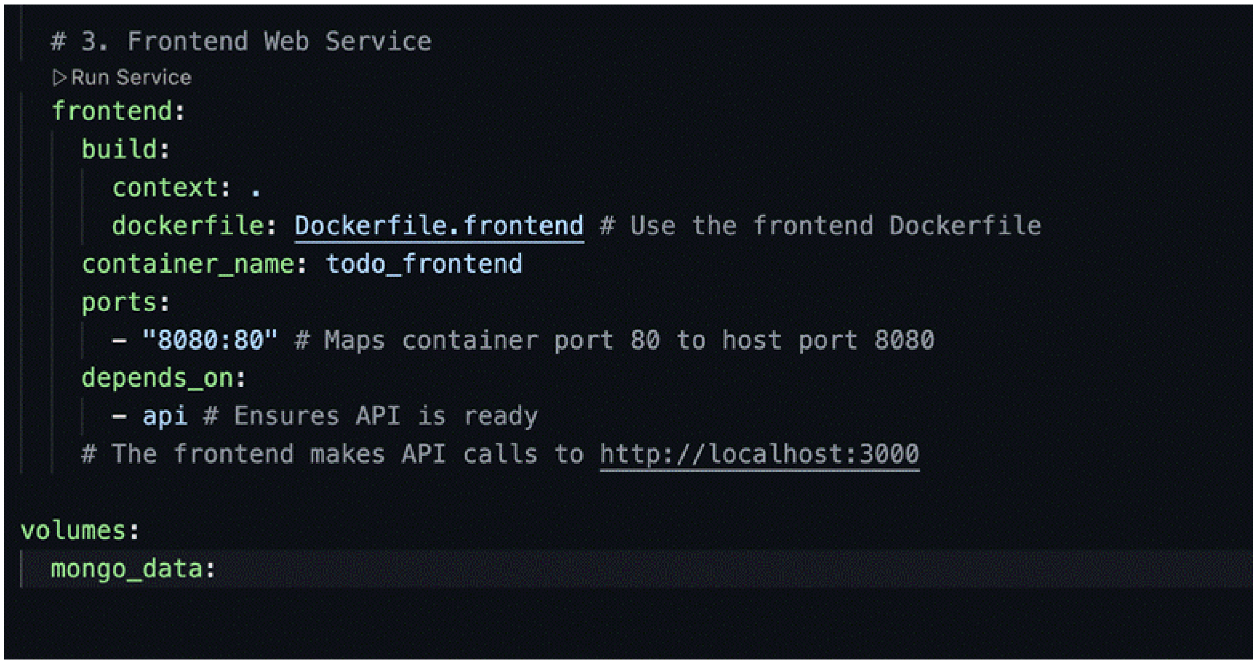
```
pipeline {
    // Defines where the pipeline runs (e.g., any available agent/node)
    agent any

    environment {
        // IMPORTANT: These variables must be set as secrets in Jenkins settings
        // for security (Credential IDs) or passed via Jenkins environment variables.
        MONGO_HOST = "mongo"
        API_PORT = "3000"
    }

    stages {
        // 1. Checkout Stage (Continuous Integration - CI)
        stage('Checkout Source Code') {
            steps {
                // Replace with your actual Todo App GitHub repository URL
                git branch: 'main', url: 'YOUR_TODO_APP_GITHUB_URL'
            }
        }

        // 2. Build Stage
        stage('Build Docker Images') {
            steps {
                echo 'Building API and Frontend Docker images...'
                // Use the docker compose build command to create the images
                sh 'docker compose build'
            }
        }
```



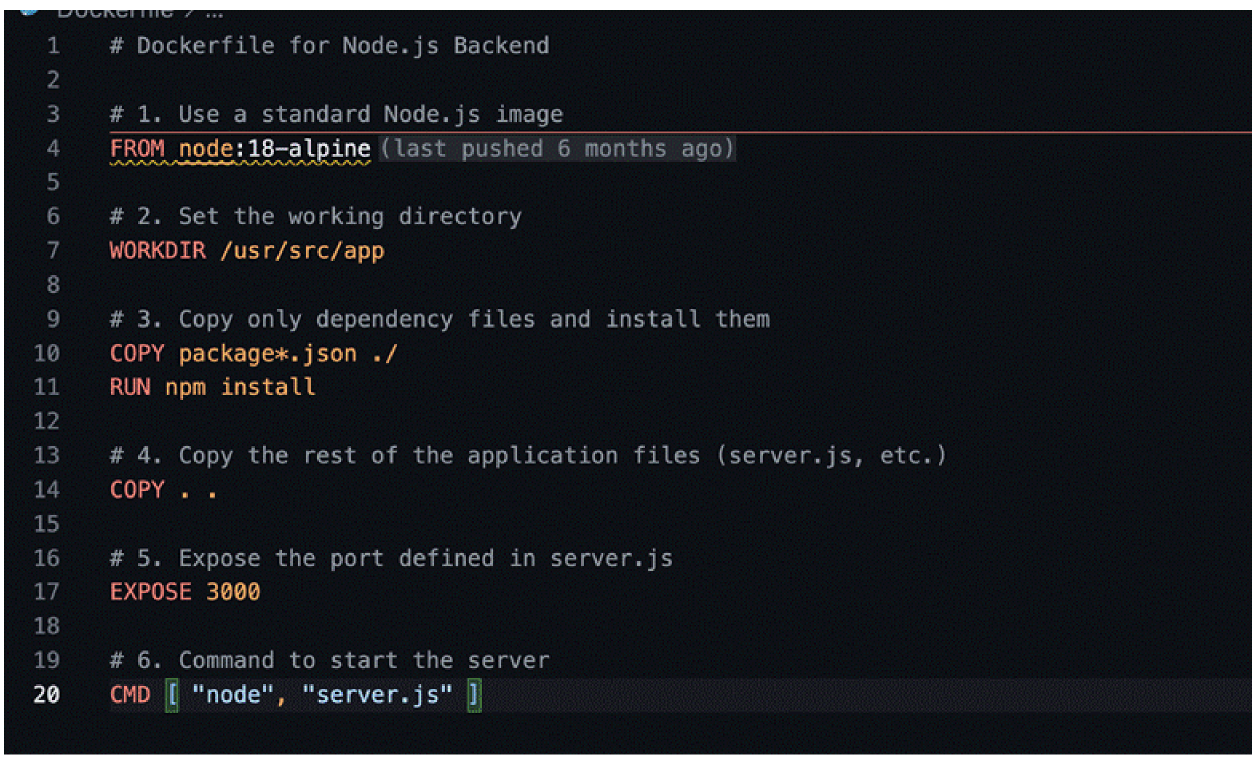
```
        // 3. Deployment Stage (Continuous Deployment - CD)
        stage('Deploy Containers') {
            steps {
                echo 'Stopping existing containers and starting new ones...'
                // Ensure any previous running containers are removed before deployment
                sh 'docker compose down --remove-orphans || true'

                // Start the new services in detached mode
                sh 'docker compose up -d'

                echo 'Deployment complete. App accessible at http://<HOST_IP>:8080'
            }
        }

        // 4. (Optional) Testing Stage
        stage('Post-Deployment Verification') {
            steps {
                // In a real project, this would be a script to hit the login endpoint
                // and verify a 200 OK response to ensure the API is fully awake.
                echo 'Verifying application health...'
            }
        }
```



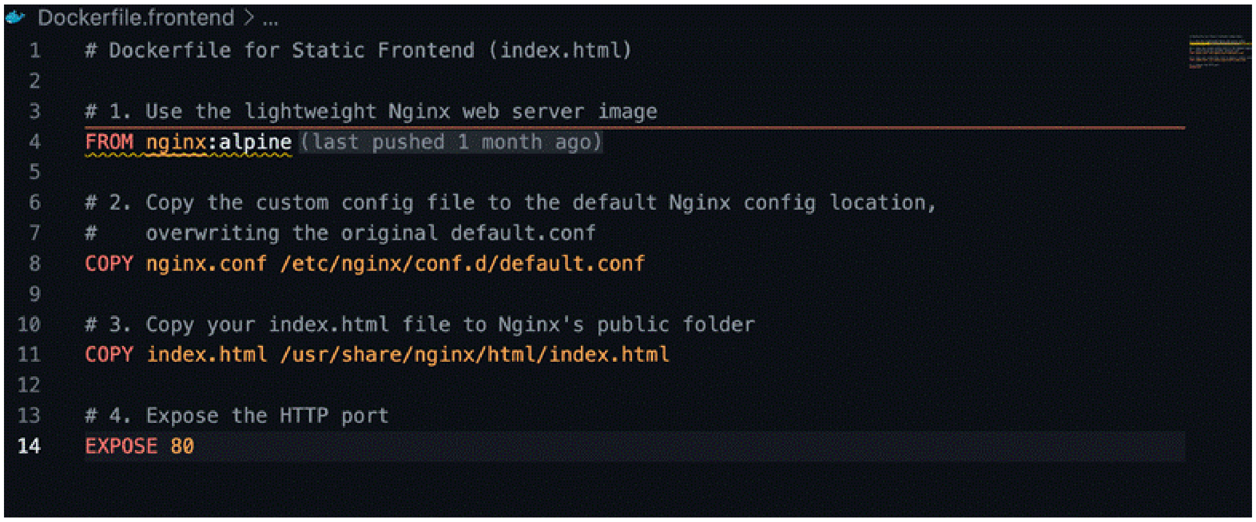
```yaml
docker-compose.yml > ...
version: '3.8'

▷Run All Services
services:
  # 1. MongoDB Database Service
  ▷Run Service
  mongo:
    image: mongo:latest
    container_name: todo_mongo_db
    restart: always
    volumes:
      - mongo_data:/data/db # Keeps data safe across container restarts

  # 2. Backend API Service
  ▷Run Service
  api:
    build: . # Use the 'Dockerfile' in the current directory
    container_name: todo_api
    restart: always
    environment:
      # Injects the secret key from your .env file securely
      JWT_SECRET: ${JWT_SECRET}
    ports:
      - "3000:3000" # Maps container port 3000 to host port 3000
    depends_on:
      - mongo # Starts the database first
    # IMPORTANT: The API needs to connect to 'mongo' (the service name) instead of 'localh
    # in the Mongoose connection string inside server.js.

```



```yaml
  # 3. Frontend Web Service
  ▷Run Service
  frontend:
    build:
      context: .
      dockerfile: Dockerfile.frontend # Use the frontend Dockerfile
    container_name: todo_frontend
    ports:
      - "8080:80" # Maps container port 80 to host port 8080
    depends_on:
      - api # Ensures API is ready
    # The frontend makes API calls to http://localhost:3000

volumes:
  mongo_data:
```



```
# Dockerfile for Node.js Backend

# 1. Use a standard Node.js image
FROM node:18-alpine (last pushed 6 months ago)

# 2. Set the working directory
WORKDIR /usr/src/app

# 3. Copy only dependency files and install them
COPY package*.json ./
RUN npm install

# 4. Copy the rest of the application files (server.js, etc.)
COPY . .

# 5. Expose the port defined in server.js
EXPOSE 3000

# 6. Command to start the server
CMD [ "node", "server.js" ]
```

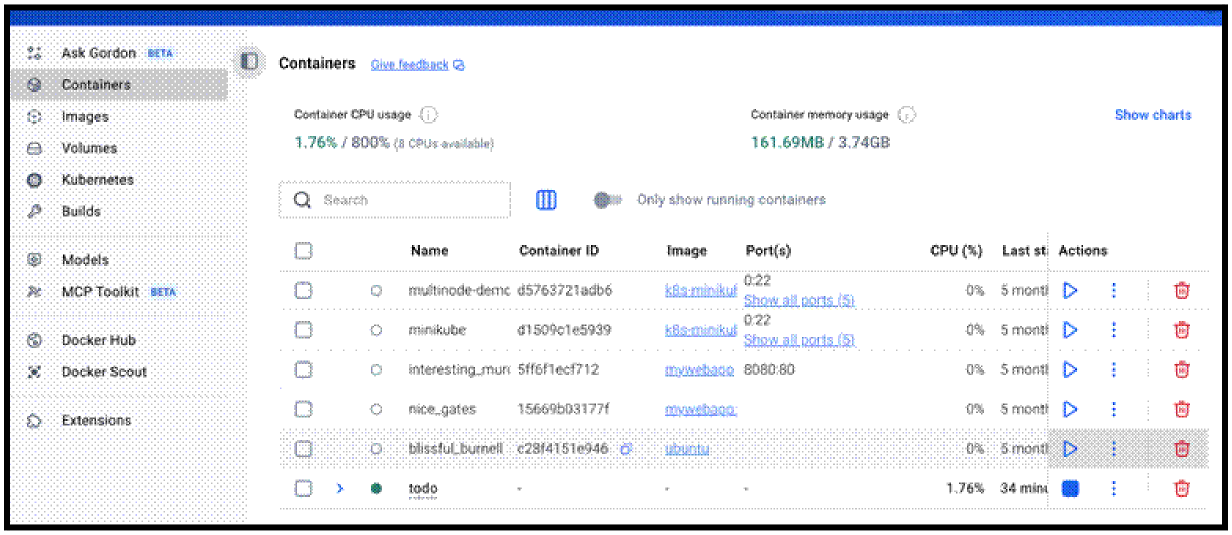


Dockerfile.frontend > ...

```
# Dockerfile for Static Frontend (index.html)

# 1. Use the lightweight Nginx web server image
FROM nginx:alpine (last pushed 1 month ago)

# 2. Copy the custom config file to the default Nginx config location,
#    overwriting the original default.conf
COPY nginx.conf /etc/nginx/conf.d/default.conf

# 3. Copy your index.html file to Nginx's public folder
COPY index.html /usr/share/nginx/html/index.html

# 4. Expose the HTTP port
EXPOSE 80
```

Output :

Fig : docker container



Containers Give feedback

Container CPU usage
1.76% / 800% (8 CPUs available)

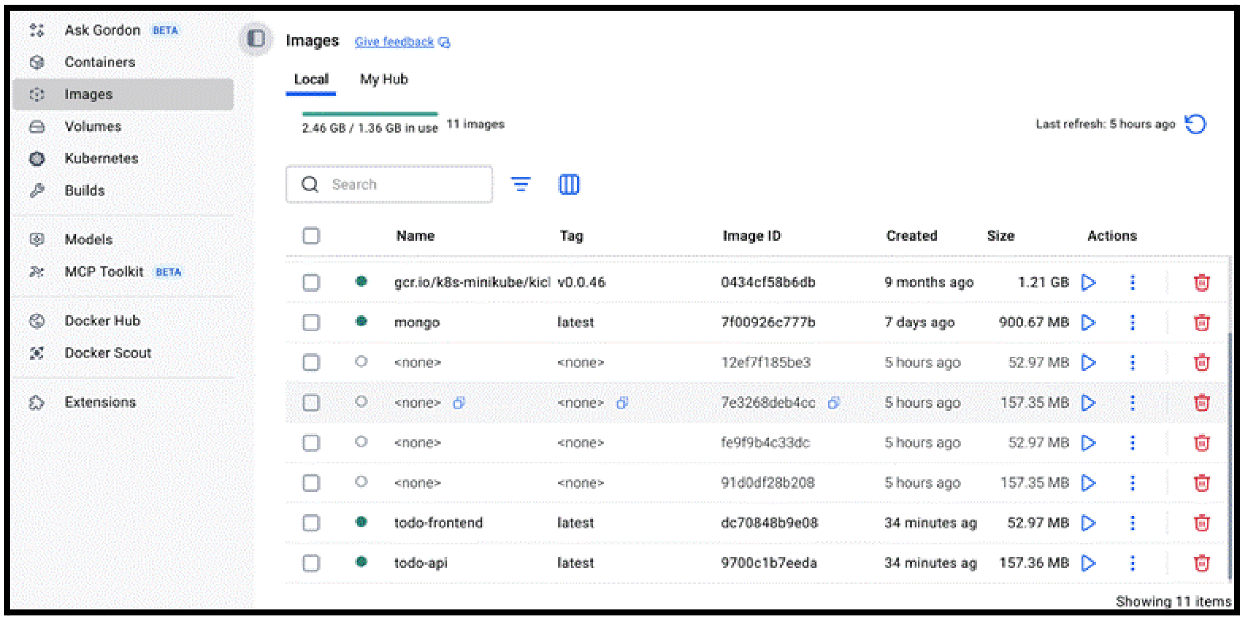
Container memory usage
161.69MB / 3.74GB

Show charts

Only show running containers

		Name	Container ID	Image	Port(s)	CPU (%)	Last st	Actions
	○	multinode-demc	d5763721adb6	k8s-minikul	0:22 Show all ports (5)	0%	5 montl	
	○	minikube	d1509c1e5939	k8s-minikul	0:22 Show all ports (5)	0%	5 montl	
	○	interesting_mur	5ff6f1ecf712	mywebapp	8080:80	0%	5 montl	
	○	nice_gates	15669b03177f	mywebapp:		0%	5 montl	
	○	blissful_burnell	c28f4151e946	ubuntu		0%	5 montl	
	●	todo	-	-	-	1.76%	34 minu	

Fig : docker images



Images Give feedback

Local My Hub

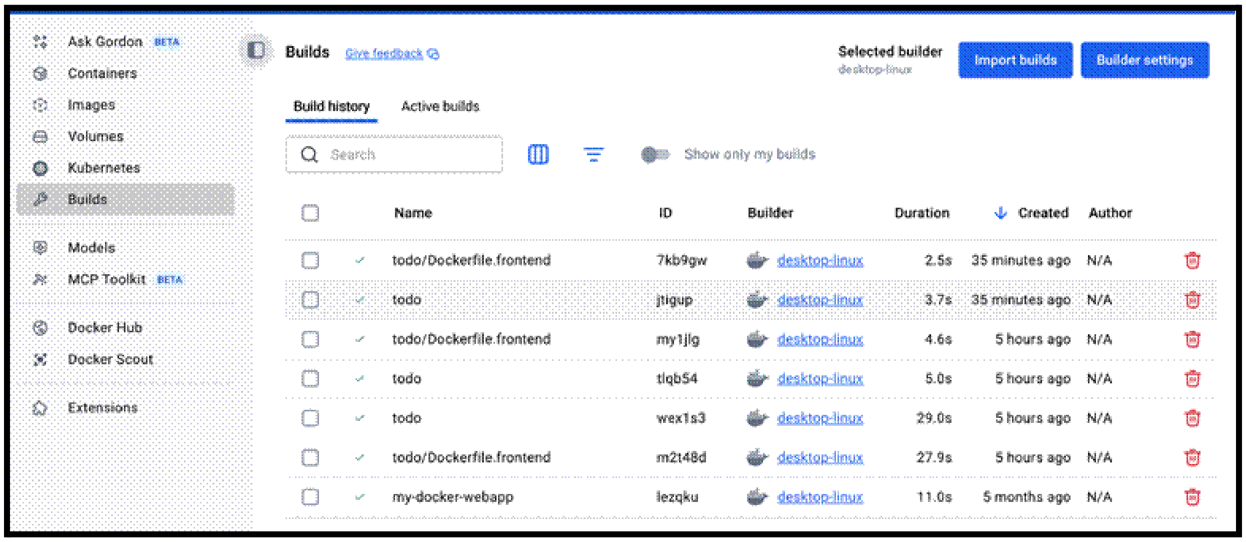
2.46 GB / 1.36 GB in use 11 images

Last refresh: 5 hours ago

		Name	Tag	Image ID	Created	Size	Actions
	●	gcr.io/k8s-minikube/kicl	v0.0.46	0434cf58b6db	9 months ago	1.21 GB	
	●	mongo	latest	7f00926c777b	7 days ago	900.67 MB	
	○	<none>	<none>	12ef7f185be3	5 hours ago	52.97 MB	
	○	<none>	<none>	7e3268deb4cc	5 hours ago	157.35 MB	
	○	<none>	<none>	fe9f9b4c33dc	5 hours ago	52.97 MB	
	○	<none>	<none>	91d0df28b208	5 hours ago	157.35 MB	
	●	todo-frontend	latest	dc70848b9e08	34 minutes ag	52.97 MB	
	●	todo-api	latest	9700c1b7eeda	34 minutes ag	157.36 MB	

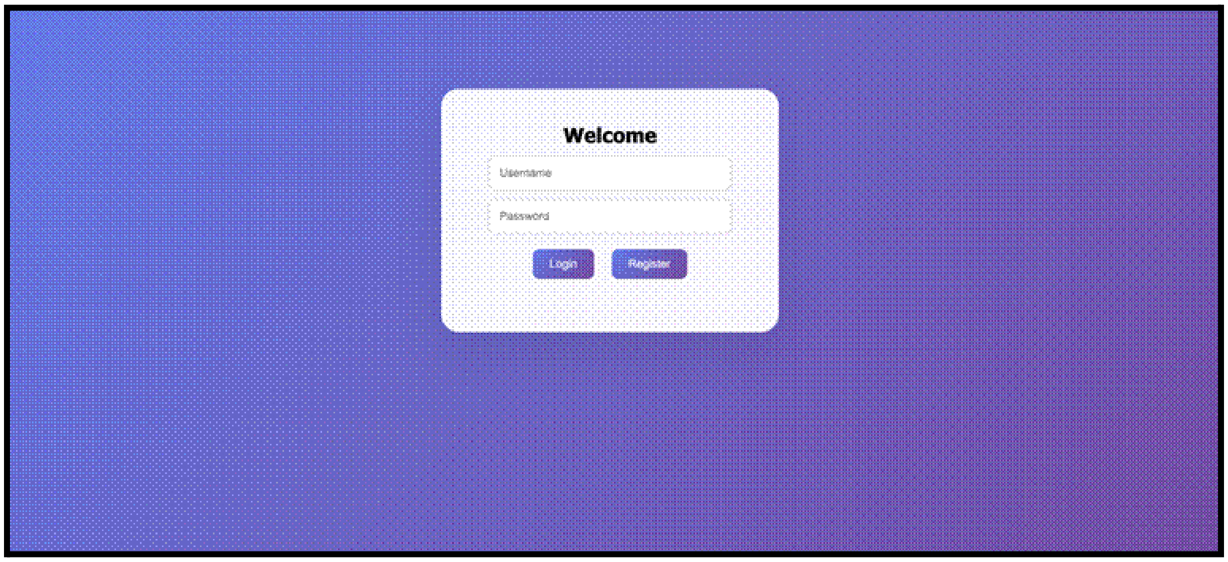
Showing 11 items

Fig : docker build



Name	ID	Builder	Duration	Created	Author
todo/Dockerfile.frontend	7kb9gw	desktop-linux	2.5s	35 minutes ago	N/A
todo	jtigup	desktop-linux	3.7s	35 minutes ago	N/A
todo/Dockerfile.frontend	my1jlg	desktop-linux	4.6s	5 hours ago	N/A
todo	tlqb54	desktop-linux	5.0s	5 hours ago	N/A
todo	wex1s3	desktop-linux	29.0s	5 hours ago	N/A
todo/Dockerfile.frontend	m2t48d	desktop-linux	27.9s	5 hours ago	N/A
my-docker-webapp	lezqku	desktop-linux	11.0s	5 months ago	N/A

Fig : app is successfully running on the local host


Welcome
Username
Password
Login
Register