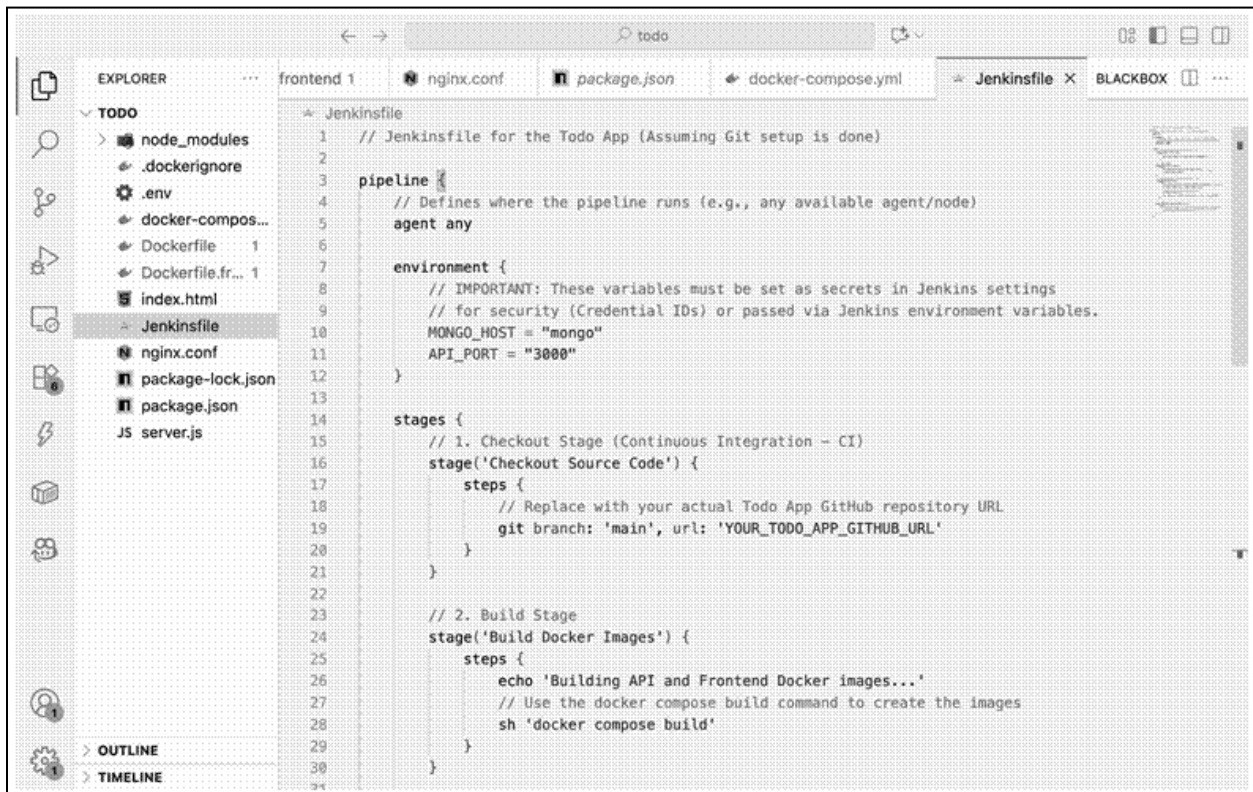


Source code:

Jenkins pipeline :



```
1 // Jenkinsfile for the Todo App (Assuming Git setup is done)
2
3 pipeline {
4     // Defines where the pipeline runs (e.g., any available agent/node)
5     agent any
6
7     environment {
8         // IMPORTANT: These variables must be set as secrets in Jenkins settings
9         // for security (Credential IDs) or passed via Jenkins environment variables.
10        MONGO_HOST = "mongo"
11        API_PORT = "3000"
12    }
13
14    stages {
15        // 1. Checkout Stage (Continuous Integration - CI)
16        stage('Checkout Source Code') {
17            steps {
18                // Replace with your actual Todo App GitHub repository URL
19                git branch: 'main', url: 'YOUR_TODO_APP_GITHUB_URL'
20            }
21        }
22
23        // 2. Build Stage
24        stage('Build Docker Images') {
25            steps {
26                echo 'Building API and Frontend Docker images...'
27                // Use the docker compose build command to create the images
28                sh 'docker compose build'
29            }
30        }
31    }
32}
```

```
32 // 3. Deployment Stage (Continuous Deployment - CD)
33 stage('Deploy Containers') {
34     steps {
35         echo 'Stopping existing containers and starting new ones...'
36         // Ensure any previous running containers are removed before deployment
37         sh 'docker compose down --remove-orphans || true'
38
39         // Start the new services in detached mode
40         sh 'docker compose up -d'
41
42         echo 'Deployment complete. App accessible at http://<HOST_IP>:8080'
43     }
44 }
45
46 // 4. (Optional) Testing Stage
47 stage('Post-Deployment Verification') {
48     steps {
49         // In a real project, this would be a script to hit the login endpoint
50         // and verify a 200 OK response to ensure the API is fully awake.
51         echo 'Verifying application health...'
52     }
53 }
54 }
55 }
```

```
docker-compose.yml > ...
1  version: '3.8'
2
   ▷Run All Services
3  services:
4    # 1. MongoDB Database Service
   ▷Run Service
5    mongo:
6      image: mongo:latest
7      container_name: todo_mongo_db
8      restart: always
9      volumes:
10       - mongo_data:/data/db # Keeps data safe across container restarts
11
12   # 2. Backend API Service
   ▷Run Service
13   api:
14     build: . # Use the 'Dockerfile' in the current directory
15     container_name: todo_api
16     restart: always
17     environment:
18       # Injects the secret key from your .env file securely
19       JWT_SECRET: ${JWT_SECRET}
20     ports:
21       - "3000:3000" # Maps container port 3000 to host port 3000
22     depends_on:
23       - mongo # Starts the database first
24     # IMPORTANT: The API needs to connect to 'mongo' (the service name) instead of 'localhost'
25     # in the Mongoose connection string inside server.js.
26
```

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 > ...
1  # Dockerfile for Node.js Backend
2
3  # 1. Use a standard Node.js image
4  FROM node:18-alpine (last pushed 6 months ago)
5
6  # 2. Set the working directory
7  WORKDIR /usr/src/app
8
9  # 3. Copy only dependency files and install them
10 COPY package*.json ./
11 RUN npm install
12
13 # 4. Copy the rest of the application files (server.js, etc.)
14 COPY . .
15
16 # 5. Expose the port defined in server.js
17 EXPOSE 3000
18
19 # 6. Command to start the server
20 CMD ["node", "server.js"]
```

```
Dockerfile.frontend > ...
1  # Dockerfile for Static Frontend (index.html)
2
3  # 1. Use the lightweight Nginx web server image
4  FROM nginx:alpine (last pushed 1 month ago)
5
6  # 2. Copy the custom config file to the default Nginx config location,
7  #    overwriting the original default.conf
8  COPY nginx.conf /etc/nginx/conf.d/default.conf
9
10 # 3. Copy your index.html file to Nginx's public folder
11 COPY index.html /usr/share/nginx/html/index.html
12
13 # 4. Expose the HTTP port
14 EXPOSE 80
```

Output :

Fig : docker container

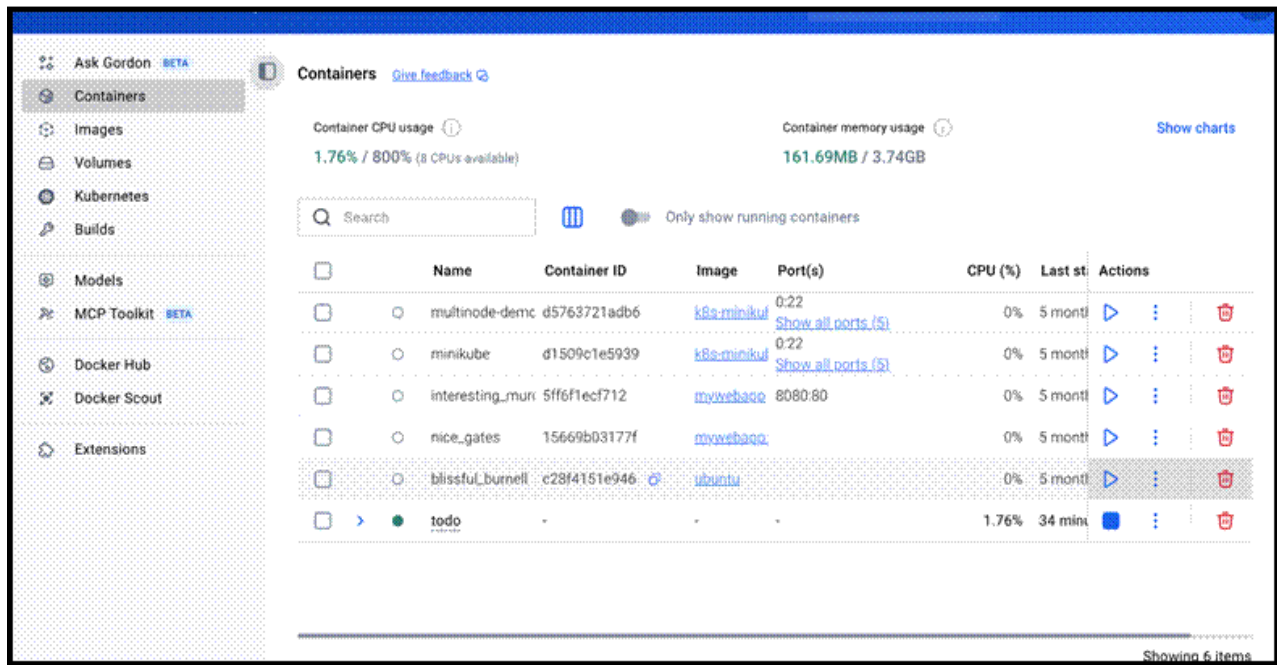


Fig : docker images

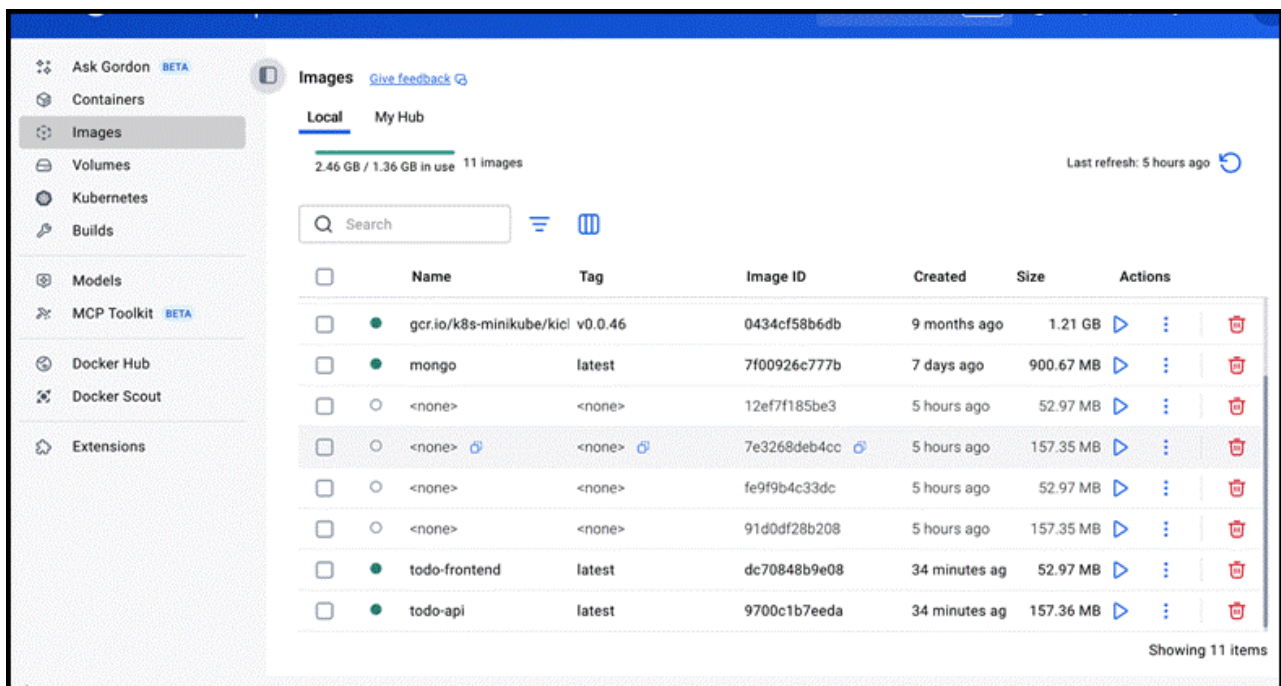


Fig : docker build

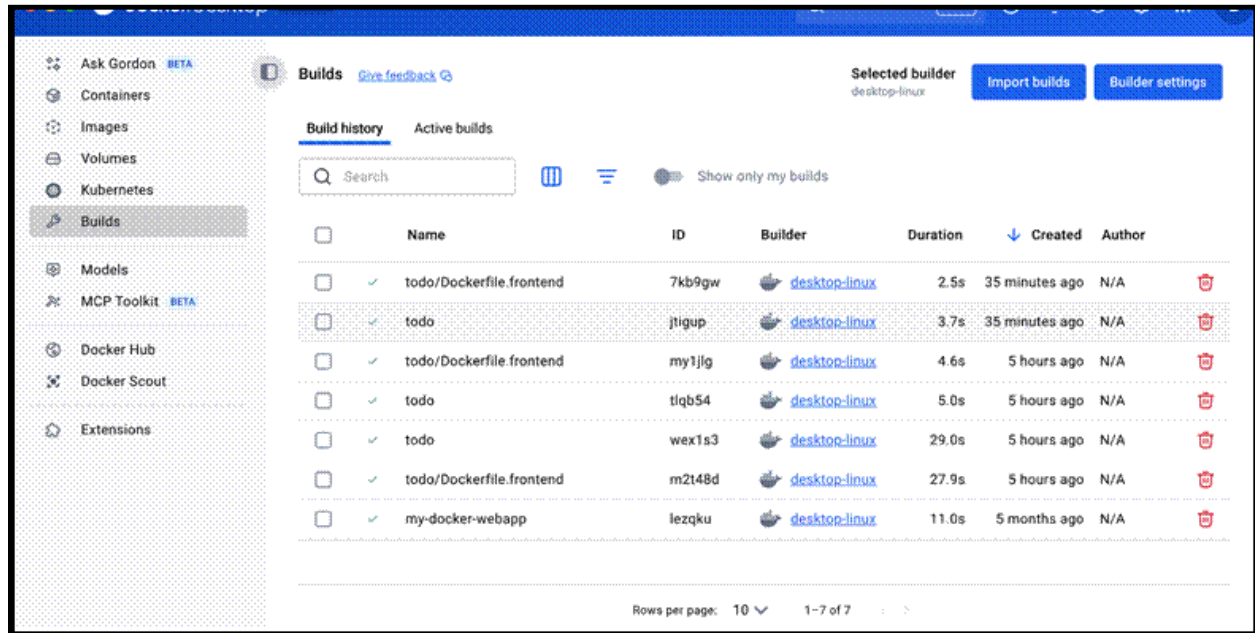


Fig : app is successfully running on the local host

