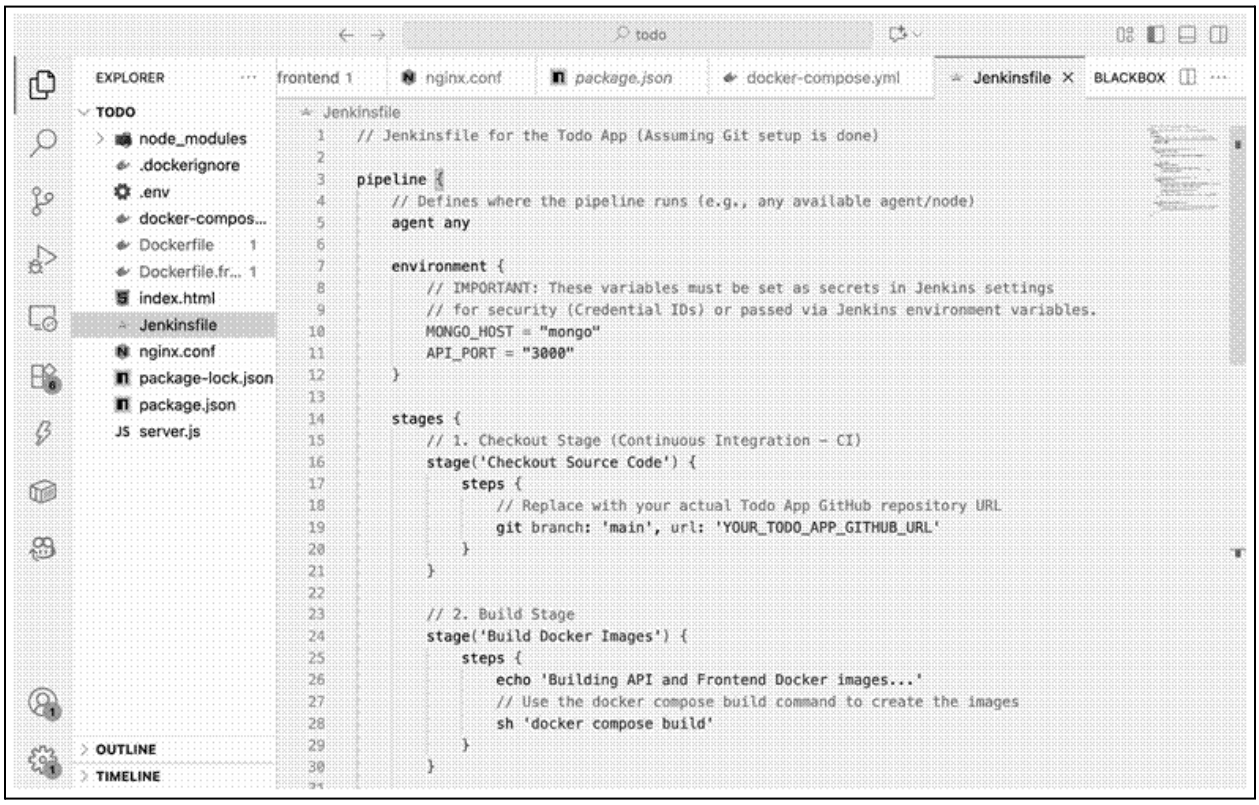
Source code:

Jenkins pipeline :




```
        // 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...'
            }
        }
    }
}
```

docker-compose.yml > ...

```yaml
version: '3.8'

services:
  # 1. MongoDB Database 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
  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
  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
# 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
# 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

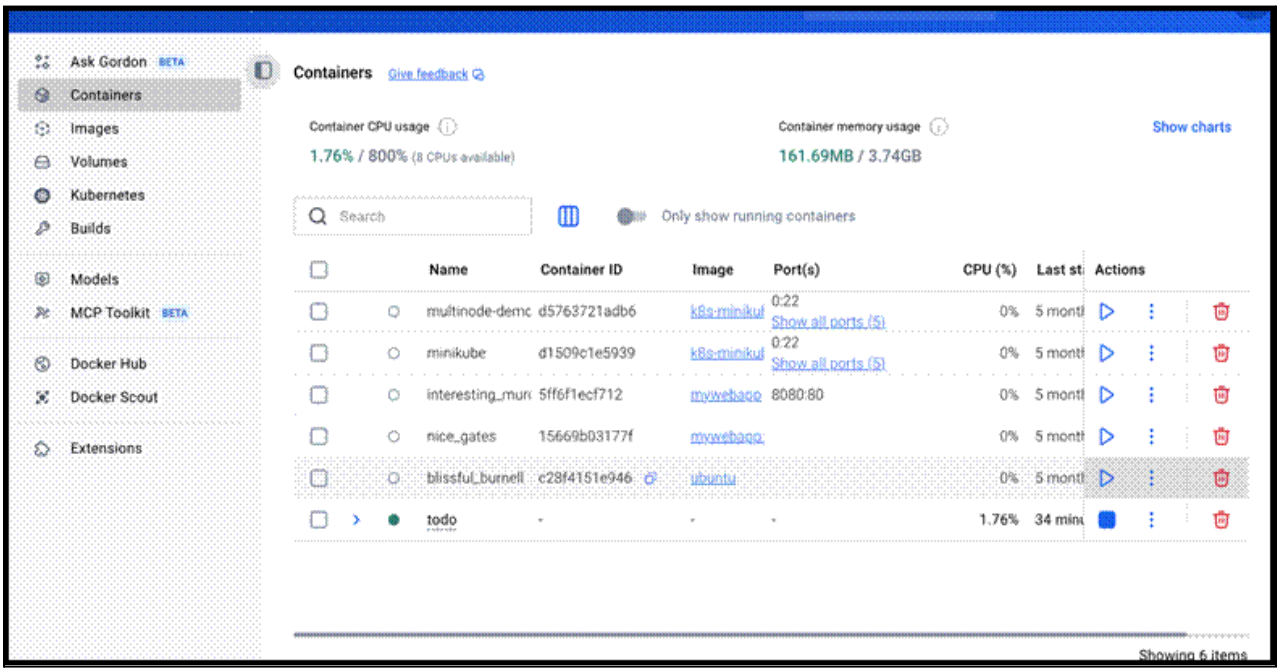


Fig : docker images

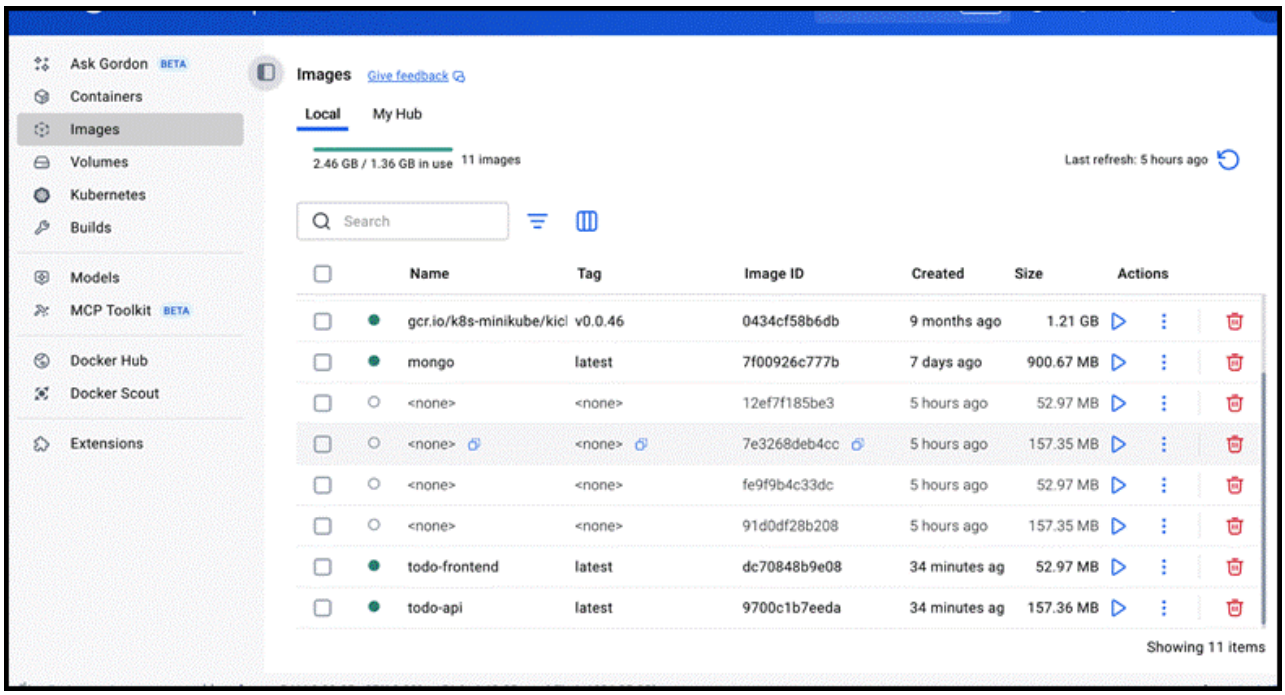


Fig : docker build

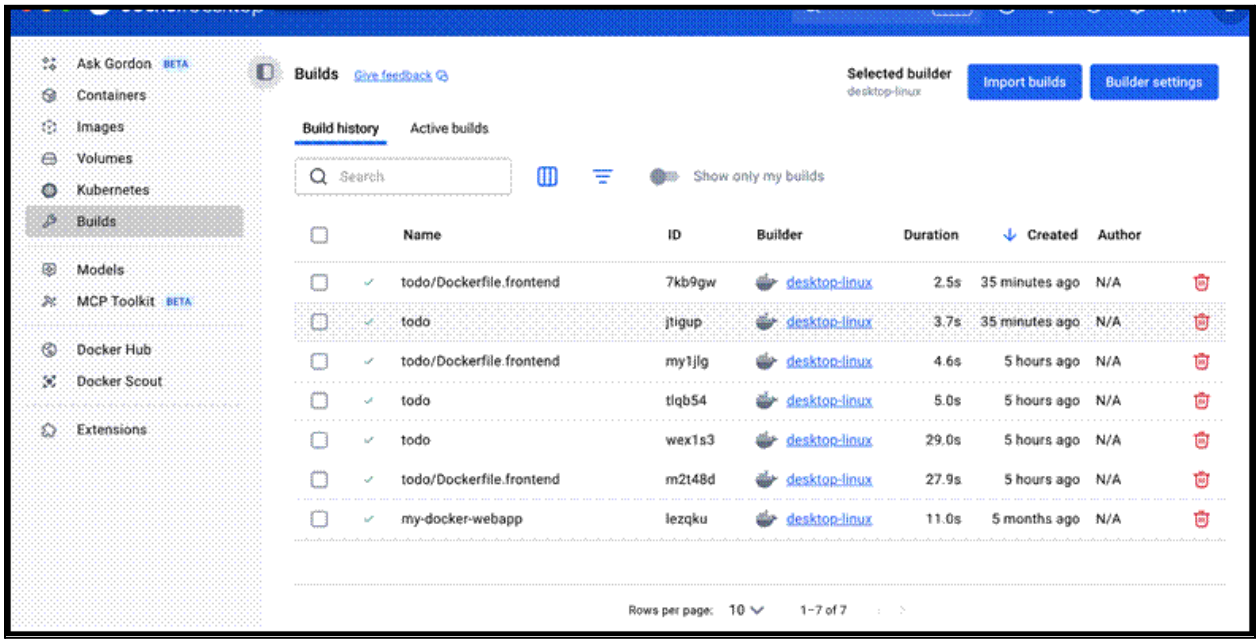


Fig : app is successfully running on the local host