Todo List Application - Comprehensive Documentation

Table of Contents

Overview

This is a full-stack Todo List application built with modern web technologies, containerized using Docker Compose for easy deployment and development. The application features a React frontend with Material-UI components, a Node.js/Express backend API, and a MongoDB database.

Technology Stack

- **Frontend**: React 18, Material-UI, Vite
- **Backend**: Node.js, Express.js
- **Database**: MongoDB
- **Containerization**: Docker, Docker Compose
- **Web Server**: Nginx (for frontend)

Prerequisites

Before setting up the application, ensure you have the following installed on your system:

- **Docker** (version 20.10 or higher)
- **Docker Compose** (version 1.29.2 or higher)
- **Git** (for cloning the repository)

System Requirements

- **RAM**: Minimum 4GB (8GB recommended)
- **Storage**: At least 2GB free space
- **OS**: Linux, macOS, or Windows with Docker support

Verify Installation

- Check Docker version
docker --version

- Check Docker Compose version
docker-compose --version

- Verify Docker is running
docker info

```
yaw@AMALITECH-PC-TKD-EQ-10684:~/Documents/github.com/Azubi-Talent-Mobility-Project/fullstack-todo-list$ docker -v
Docker version 28.1.1, build 4eba377
yaw@AMALITECH-PC-TKD-EQ-10684:~/Documents/github.com/Azubi-Talent-Mobility-Project/fullstack-todo-list$ docker-compose -v
docker-compose version 1.29.2, build unknown
yaw@AMALITECH-PC-TKD-EQ-10684:~/Documents/github.com/Azubi-Talent-Mobility-Project/fullstack-todo-list$ docker info
Client: Docker Engine - Community
 Version:    28.1.1
 Context:    default
 Debug Mode: false
 Plugins:
  buildx: Docker Buildx (Docker Inc.)
    Version:  v0.23.0
    Path:     /usr/libexec/docker/cli-plugins/docker-buildx
  compose: Docker Compose (Docker Inc.)
    Version:  v2.35.1
```

Setup Instructions

Step 1: Clone the Repository

git clone https://github.com/Lay-ke/Full-Stack-Todo-List.git
cd Full-Stack-Todo-List

Step 2: Verify Project Structure

Ensure your project structure matches the following:

```
fullstack-todo-list/
├── docker-compose.yml
├── Frontend/
│   ├── Dockerfile
│   ├── package.json
│   ├── src/
│   └── ...
├── Backend/
│   ├── Dockerfile
```

```
│   ├── package.json
│   ├── index.js
│   └── …
├── test-containers.sh
└── README.md
```

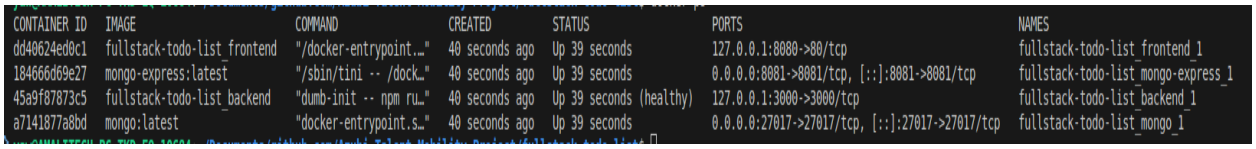
Step 3: Build and Start Containers

- Build and Start all services
docker-compose up -d --build

- View live logs
docker-compose logs -f

Step 4: Verify Services are Running

- Check container status
docker-compose ps



CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
dd40624ed0c1	fullstack-todo-list_frontend	"/docker-entrypoint.…"	40 seconds ago	Up 39 seconds	127.0.0.1:8080->80/tcp	fullstack-todo-list_frontend_1
184666d69e27	mongo-express:latest	"/sbin/tini -- /dock…"	40 seconds ago	Up 39 seconds	0.0.0.0:8081->8081/tcp, [::]:8081->8081/tcp	fullstack-todo-list_mongo-express_1
45a9f87873c5	fullstack-todo-list_backend	"dumb-init -- npm ru…"	40 seconds ago	Up 39 seconds (healthy)	127.0.0.1:3000->3000/tcp	fullstack-todo-list_backend_1
a7141877a8bd	mongo:latest	"docker-entrypoint.s…"	40 seconds ago	Up 39 seconds	0.0.0.0:27017->27017/tcp, [::]:27017->27017/tcp	fullstack-todo-list_mongo_1

Step 5: Access the Application

- **Frontend**: http://localhost:8080
- **Backend API**: http://localhost:3000
- **MongoDB Express**: http://localhost:8081
- **MongoDB**: localhost:27017

Step 6: Stop the Application

- Stop and remove volumes (clears database data)
docker-compose down -v

Network and Security Configurations

Docker Network Configuration

The application uses a custom bridge network (app-network) for secure inter-service communication:

```
networks:
  app-network:
    driver: bridge
```

Port Exposures

Service	Internal Port	External Port	Purpose
Frontend	80	127.0.0.1:8080	Web application
Backend	3000	127.0.0.1:3000	API server
MongoDB	27017	127.0.0.1:27017	Database
Mongo Express	8081	127.0.0.1:8081	Database management UI

Security Settings

Database Credentials

- MongoDB Root User
  - MONGO_INITDB_ROOT_USERNAME: admin
  - MONGO_INITDB_ROOT_PASSWORD: adminpassword

- Backend Connection
  - MONGO_URI: mongodb://admin:adminpassword@mongo:27017/?authSource=admin

Environment Variables

- Backend Environment
  - MONGO_URI=mongodb://admin:adminpassword@mongo:27017/?authSource=admin

- Frontend Environment (if needed)
  - REACT_APP_API_URL=http://localhost:3000

Security Considerations

- **Database Access**: MongoDB is accessible only within the Docker network
- **API Access**: Backend API is bound to localhost only

- **Frontend**: Served through Nginx with proper CORS configuration
- **Credentials**: Default credentials should be changed in production

Application Architecture

Service Dependencies

Frontend → Backend → MongoDB
  ↓   ↓   ↓
 Nginx   Express   MongoDB

Data Flow

1. **Frontend** (React) sends HTTP requests to **Backend**
2. **Backend** (Express) processes requests and interacts with **MongoDB**
3. **MongoDB** stores and retrieves todo data
4. **Backend** returns JSON responses to **Frontend**
5. **Frontend** updates UI based on responses

Database Schema

```
{
  _id: ObjectId,
  title: String,
  description: String,
  date: String,
  strStatus: Boolean (default: false),
  createdAt: Date,
  updatedAt: Date
}
```

API Documentation

Base URL

http://localhost:3000/

Endpoints

1. Create Todo

POST /api/todos

2. Get All Todos
GET /api/gettodos

3. Update Todo Status
PUT /api/todos/:id

4. Delete Todo
DELETE /api/todos/:id

Container Testing Script

Automated Testing Script

In the git repo you'll find test-containers.sh file to be used for automated testing. Change this file permission to executable using the command for linux: **"chmod +x test-containers.sh"** and run **"./test-containers.sh"**.

Troubleshooting Guide

Common Issues and Solutions

1. Container Startup Issues

Problem: Containers fail to start

Check logs for specific errors
docker-compose logs

- Restart containers
docker-compose down
docker-compose up -d --build

Problem: Port conflicts

- Check what process is using the ports
sudo netstat -tulpn | grep :<port number> **OR**
sudo lsof -i :<port number>

- Either stop the process using the port using the command
  “sudo kill <process ID>” **OR**
  Change ports in docker-compose.yml if needed

2. Database Connection Issues

Problem: MongoDB authentication failed

- Check MongoDB logs

```
docker-compose logs mongo
```

- Verify environment variables

```
docker-compose exec backend env | grep MONGO
```

- Reset MongoDB data

```
docker-compose down -v
docker-compose up -d
```

Problem: Backend can't connect to MongoDB

- Check network connectivity

```
docker-compose exec backend ping -c 4 mongo
```

- Verify MongoDB is running

```
docker-compose exec mongo mongosh --eval "db.runCommand('ping')"
```

3. Frontend Issues

Problem: Frontend not loading

- Check Nginx configuration

```
docker-compose exec frontend nginx -t
```

- Check frontend build

```
docker-compose exec frontend ls -la /usr/share/nginx/html/
```

- Rebuild frontend

```
docker-compose build frontend
docker-compose up -d frontend
```

Problem: API calls failing from frontend

- Check CORS configuration

```
docker-compose logs backend
```

- Verify API endpoints

```
curl http://localhost:3000/api/gettodos
```

4. Performance Issues

Problem: Slow application response

- Check container resources usage

```
docker stats
```

This comprehensive documentation should help you successfully deploy, test, and maintain the Todo List application in a containerized environment.