

Building a 2-Tier App with Node and MongoDB using Docker

Introduction

This documentation shows the process of building a 2-tier app project using Node.js and MongoDB, initially by setting up the components manually and later using a Docker Compose YML file for automation.

Components Required:

- Node.js
- MongoDB
- Docker

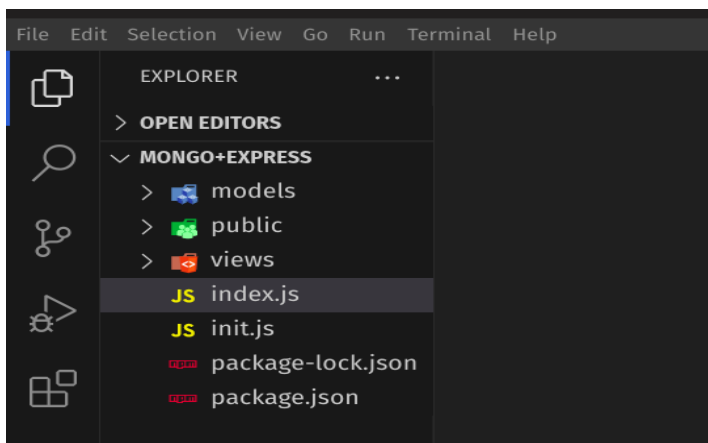
Manual Setup:

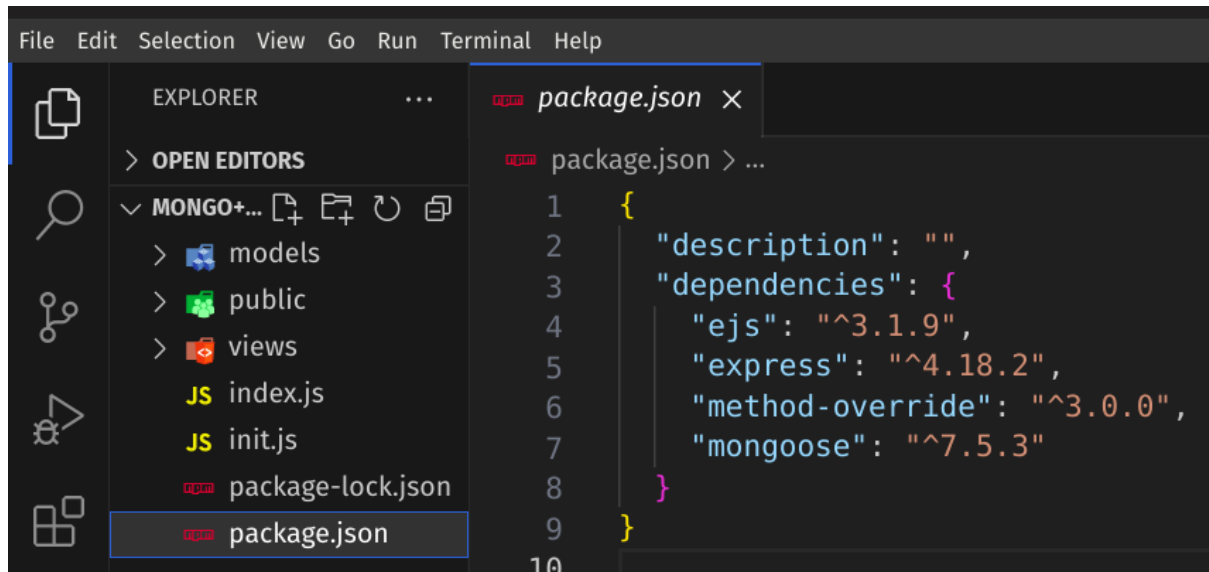
Step 1: Install Node.js

1. Verify the installation by running `node -v` and `npm -v` in your terminal.

Step 2: Create a Node.js App

1. Create a new directory for your app: `mkdir Mongo+Express`
2. Navigate to the directory: `cd Mongo+Express`
3. Initialise a new Node.js project: `npm init -y`
4. All the code is written here.
5. Install all the dependency the node app require, like (ejs, express, mongoose, method-override)





Step 3: Install MongoDB

1. Pull the mongo image from Docker-hub.
2. List the images -> **docker images**

```
● sakib@The-Silly-Sultan:~/Desktop/MongoDb/Mongo+Express$ docker pull mongo
Using default tag: latest
latest: Pulling from library/mongo
Digest: sha256:31bf43c4959c283733a004b0a3a9c4ddc52efafb4cf9a38e42d2da93e9a72546
Status: Image is up to date for mongo:latest
docker.io/library/mongo:latest
● sakib@The-Silly-Sultan:~/Desktop/MongoDb/Mongo+Express$ docker images
REPOSITORY          TAG         IMAGE ID      CREATED        SIZE
appimg               latest     fd61f0afbfe0  9 hours ago   1.12GB
ubuntu               latest     b25dc2abdf78  4 days ago    122MB
utility-container_npm-image latest     cda3acebc4c7  5 days ago    119MB
node                 latest     386e0be86bde  6 days ago    1.1GB
mongo                latest     9576663f05bb  3 weeks ago   736MB
○ sakib@The-Silly-Sultan:~/Desktop/MongoDb/Mongo+Express$
```

Step 4: Write a Dockerfile and Build an image for node-app-img

1. Writing a simple dockerfile for the node-images.
2. Build an image from it -> **docker build -t node-app-img .**

```
FROM node
WORKDIR /app
COPY package.json ./
RUN npm install
COPY ./ ./
EXPOSE 8080
CMD ["node","index.js"]
```

3. The image has been built named node-app-img.

```
● sakib@The-Silly-Sultan:~/Desktop/MongoDb/Mongo+Express$ docker build -t node-app-img .
[+] Building 16.7s (10/10) FINISHED
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 147B
=> [internal] load metadata for docker.io/library/node:latest
=> [internal] load build context
=> => transferring context: 21.83MB
=> [1/5] FROM docker.io/library/node
=> CACHED [2/5] WORKDIR /app
=> [3/5] COPY package.json ./
=> [4/5] RUN npm install
=> [5/5] COPY ./ ./
=> exporting to image
=> => exporting layers
=> => writing image sha256:e70c02b4765145912523209ac57d6cca4af39bf789246f0083854ec68ef6af70
=> => naming to docker.io/library/node-app-img
● sakib@The-Silly-Sultan:~/Desktop/MongoDb/Mongo+Express$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
node-app-img         latest             e70c02b47651       About a minute ago 1.16GB
ubuntu              latest             b25dc2abdf78       4 days ago         122MB
utility-container_npm-image latest             cda3acebc4c7       5 days ago         119MB
node                latest             386e0be86bde       6 days ago         1.1GB
mongo               latest             9576663f05bb       3 weeks ago        736MB
● sakib@The-Silly-Sultan:~/Desktop/MongoDb/Mongo+Express$
```

Step 5: Now create a network and run the mongo image as a mongoDB container.

1. docker network create --driver bridge my-net
2. docker run -d -v myvol:/data/db --network=my-net --name mongoDB mongo.

```
● sakib@The-Silly-Sultan:~/Desktop/MongoDb/Mongo+Express$ docker run -d -v myvol:/data/db --network=my-net --name mongoDB mongo
ffdcf0e62b6cf4f22e3aace658db6ab4c68ba9b4fe63ba9b7246f3dbd797905a
● sakib@The-Silly-Sultan:~/Desktop/MongoDb/Mongo+Express$ docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS        NAMES
ffdcf0e62b6c   mongo     "docker-entrypoint.s..." 4 seconds ago Up 3 seconds  27017/tcp    mongoDB
● sakib@The-Silly-Sultan:~/Desktop/MongoDb/Mongo+Express$
```

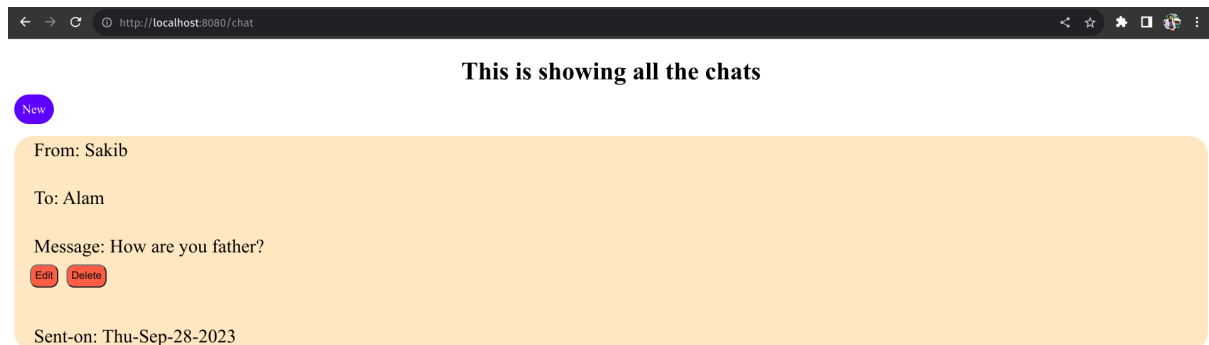
Step 6: Now create a container from node-app-img under the same network of mongoDB containers.

1. docker run -d -v /home/sakib/Desktop/MongoDb/Mongo+Express:/app -v /app/node_modules -p 8080:8080 --network=my-net --name nodeapp node-app-img

```
● sakib@The-Silly-Sultan:~/Desktop/MongoDb/Mongo+Express$ docker run -d -v /home/sakib/Desktop/MongoDb/Mongo+Express:/app -v /app/node_modules -p 8080:8080 --network=my-net --name nodeapp node-app-img
90f1735ac1a1e3843ce05809221355a71976ec655feb5044faa57e380325c488
● sakib@The-Silly-Sultan:~/Desktop/MongoDb/Mongo+Express$ docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS        NAMES
90f1735ac1a1   node-app-img "docker-entrypoint.s..." 3 seconds ago Up 2 seconds  0.0.0.0:8080->8080/tcp, :::8080->8080/t cp
nodeapp
ffdcf0e62b6c   mongo     "docker-entrypoint.s..." 11 minutes ago Up 11 minutes  27017/tcp    mongoDB
mongoDB
```

Step 7: Hence the app is working successfully and the two containers are also running.

```
sakib@The-Silly-Sultan:~/Desktop/MongoDb/Mongo+Express$ docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS        PORTS
90f1735ac1a1   node-app-img  "docker-entrypoint.s..."  4 minutes ago  Up 4 minutes  0.0.0.0:8080->8080/tcp, :::8080->8080/t
cp            nodeapp
ffdcf0e62b6c   mongo       "docker-entrypoint.s..."  16 minutes ago  Up 16 minutes  27017/tcp
mongoDB
sakib@The-Silly-Sultan:~/Desktop/MongoDb/Mongo+Express$
```



Step 8: Make an image from the running container nodeapp named mychat-img.

```
sakib@The-Silly-Sultan:~/Desktop/MongoDb/Mongo+Express$ docker commit nodeapp mychat-img
sha256:f88a5f4da0eachbed7fa9e33f7e69a26ee765be4909b6826b0387bf3c9bea3f17
sakib@The-Silly-Sultan:~/Desktop/MongoDb/Mongo+Express$ docker images
REPOSITORY          TAG         IMAGE ID      CREATED        SIZE
mychat-img          latest     f88a5f4da0ea  7 seconds ago  1.16GB
node-app-img        latest     8219b2b525d8  8 minutes ago  1.16GB
ubuntu              latest     b25dc2abdf78  4 days ago    122MB
utility-container_npm-image latest     cda3acebc4c7  5 days ago    119MB
node                 latest     386e0be86bde  6 days ago    1.1GB
mongo                latest     9576663f05bb  3 weeks ago   736MB
```


Step 9: Tag the image and push to my docker hub mychat-app repo.


```
sakib@The-Silly-Sultan:~/Desktop/MongoDb/Mongo+Express$ docker tag mychat-img:latest sakib3001/mychat-app:latest
sakib@The-Silly-Sultan:~/Desktop/MongoDb/Mongo+Express$ docker push sakib3001/mychat-app:latest
The push refers to repository [docker.io/sakib3001/mychat-app]
9dcc5d355a0f: Pushed
421bba7e1f20: Pushed
6e2f7f9a0489: Pushed
ae386fd8bd64: Pushed
0b75e245b3f9: Mounted from library/node
4f9a73c4906d: Mounted from library/node
817af8548206: Mounted from library/node
abadbde9f447: Mounted from library/node
01d6cdeac539: Mounted from library/node
a981ddddd4c65: Mounted from library/node
f6589095d5b5: Mounted from library/node
7c85cfa30cb1: Mounted from library/node
latest: digest: sha256:a4f3de03ad3466893cfd8b52aef2959503bd927b8393411c8f6e8b41fc05cf19 size: 2840
sakib@The-Silly-Sultan:~/Desktop/MongoDb/Mongo+Express$
```

Link: [Docker-Hub \(Please Have a look \)](#)

sakib3001 / mychat-app



Description

This a 2-tier app project using Node.js and MongoDB, shows the basic CRUD operation. 

 Last pushed: a minute ago

Tags

This repository contains 1 tag(s).

Tag	OS	Type	Pulled	Pushed
 latest		Image	---	a minute ago

[See all](#)

[Go to Advanced Image Management](#)

