

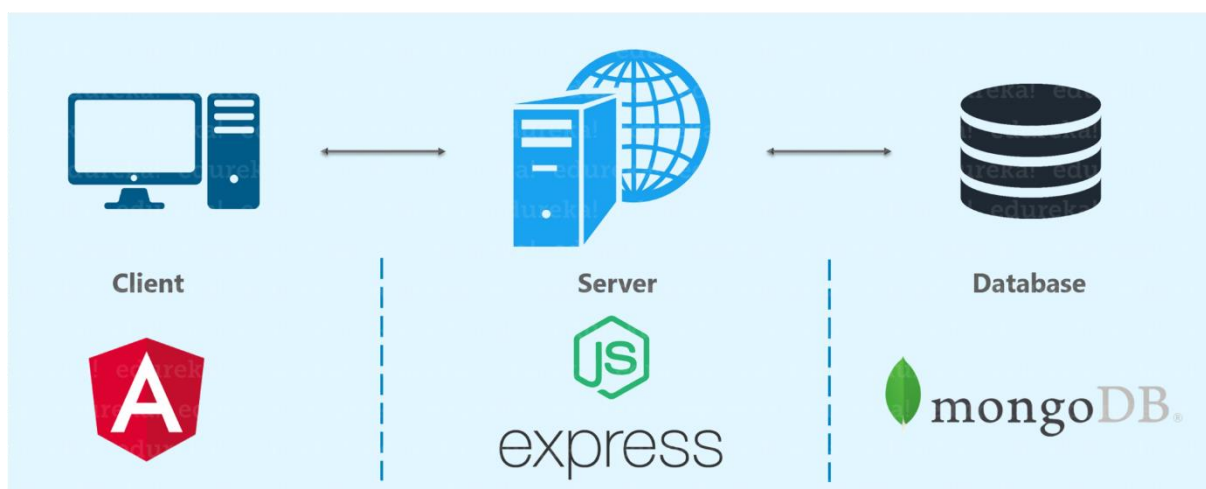
[Type here]

Docker Compose for containerizing a MEAN Stack Application :

Docker Compose can be used to create separate containers (and host them) for each of the stacks in a MEAN stack application. MEAN is the acronym for MongoDB Express Angular & NodeJs.

By using Docker Compose, we can host each of these technologies in separate containers on the same host and get them to communicate with each other. Each container will expose a port for communicating with other containers.

The communication and up-time of these containers will be maintained by Docker Compose.



In our case, we have a full stack application which comprises of MongoDB, ExpressJS, Angular and NodeJS. MongoDB takes care of the back end database, NodeJS and ExpressJS are for server side rendering, and Angular is for front end.

[Type here]

Write a separate dockerfile for building each of the single-container applications. Additionally, we will have to also write a Docker Compose File which will do the actual work. Docker Compose File will execute the different dockerfiles to create the different containers and let them interact with each other.

Creating Docker Containers

1. Dockerfile For Front End

```
1 FROM node:6
2 RUN mkdir -p /usr/src/app
3 WORKDIR /usr/src/app
4 COPY package.json /usr/src/app
5 RUN npm cache clean
6 RUN npm install
7 COPY . /usr/src/app
8 EXPOSE 4200
9 CMD ["npm","start"]
```

2. Dockerfile for Back End:

```
FROM node:6
RUN mkdir -p /usr/src/app
WORKDIR /usr/src/app
COPY package.json /usr/src/app
RUN npm cache clean
RUN npm install
COPY . /usr/src/app
EXPOSE 3000
CMD ["npm","start"]
```

[Type here]

If you are using latest versions :

To download Compose, run the below set of commands.

```
1 sudo curl -L https://github.com/docker/compose/releases/download/1.25.0/docker-compose-\$\(uname -s\)-\$\(uname -m\) -o /usr/local/bin/docker-compose
2 sudo chmod +x /usr/local/bin/docker-compose
```

```
sudo edit docker-compose.yml
```

```
version: '3.0' # specify docker-compose version
```

```
# Define the services/ containers to be run
```

```
services:
```

```
  angular: # name of the first service
```

```
    build: angular-app # specify the directory of the Dockerfile
```

```
    ports:
```

```
      - "4200:4200" # specify port mapping
```

```
  express: # name of the second service
```

```
    build: express-server # specify the directory of the Dockerfile
```

```
    ports:
```

```
      - "3000:3000" #specify ports mapping
```

```
    links:
```

```
      - database # link this service to the database service
```

```
  database: # name of the third service
```

```
    image: mongo # specify image to build container from
```

```
    ports:
```

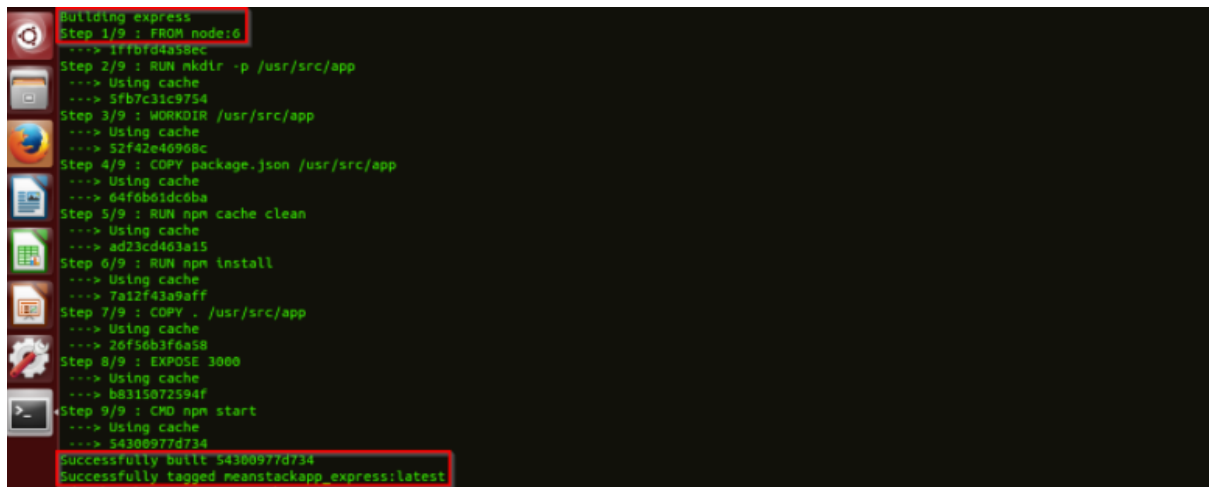
```
      - "27017:27017" # specify port forwarding
```

Run these commands from same folder where Docker compose file is present

```
docker-compose build
```

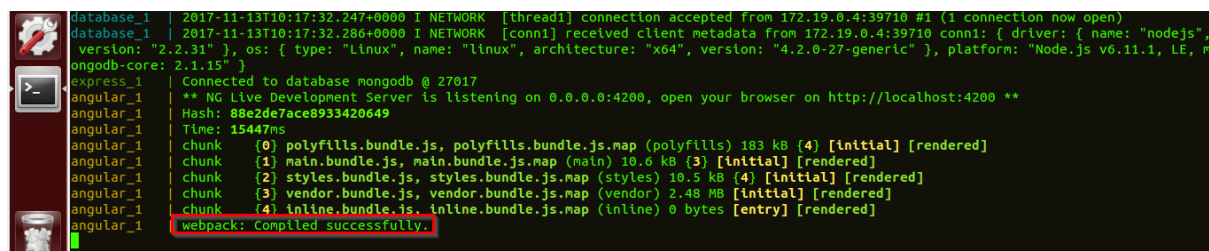
```
docker-compose up
```

[Type here]



```
Building express
Step 1/9 : FROM node:6
--> 1f10d4a88ec
Step 2/9 : RUN mkdir -p /usr/src/app
--> Using cache
--> 5fb7c31c9754
Step 3/9 : WORKDIR /usr/src/app
--> Using cache
--> 52f42e46968c
Step 4/9 : COPY package.json /usr/src/app
--> Using cache
--> 64f0b61dc0ba
Step 5/9 : RUN npm cache clean
--> Using cache
--> ad23cd463a15
Step 6/9 : RUN npm install
--> Using cache
--> 7a12f43a9aff
Step 7/9 : COPY . /usr/src/app
--> Using cache
--> 26f56b3faa58
Step 8/9 : EXPOSE 3000
--> Using cache
--> b8315072594f
Step 9/9 : CMD npm start
--> Using cache
--> 54300977d734
Successfully built 54300977d734
Successfully tagged meanstackapp:express:latest
```

Now run docker-compose up



```
database_1 | 2017-11-13T10:17:32.247+0000 I NETWORK [thread1] connection accepted from 172.19.0.4:39710 #1 (1 connection now open)
database_1 | 2017-11-13T10:17:32.286+0000 I NETWORK [conn1] received client metadata from 172.19.0.4:39710 conn1: { driver: { name: "nodejs",
version: "2.2.31" }, os: { type: "Linux", name: "linux", architecture: "x64", version: "4.2.0-27-generic" }, platform: "Node.js v6.11.1, LE, m
ongod-core: 2.1.15" }
express_1 | Connected to database mongodb @ 27017
angular_1 | ** NG Live Development Server is listening on 0.0.0.0:4200, open your browser on http://localhost:4200 **
angular_1 | Hash: 88e2de7ace8933420649
angular_1 | Time: 15447ms
angular_1 | chunk {0} polyfills.bundle.js, polyfills.bundle.js.map (polyfills) 183 kB [initial] [rendered]
angular_1 | chunk {1} main.bundle.js, main.bundle.js.map (main) 10.6 kB [initial] [rendered]
angular_1 | chunk {2} styles.bundle.js, styles.bundle.js.map (styles) 10.5 kB [initial] [rendered]
angular_1 | chunk {3} vendor.bundle.js, vendor.bundle.js.map (vendor) 2.48 MB [initial] [rendered]
angular_1 | chunk {4} inline.bundle.js, inline.bundle.js.map (inline) 0 bytes [entry] [rendered]
angular_1 | webpack: Compiled successfully.
```

Go type the following port numbers in your web browser to interact with the GUI of the MEAN app.

localhost:4200 – Angular App (Front-end)

localhost:3000 – Express Server & NodeJS (Back-end/ Server-side)

localhost:27017 – MongoDB (Database)

“docker-compose scale='x'” command to easily scale up/ down the number of deployments.

[Type here]

docker-compose scale=5