This documents gives a detailed steps on how to deploy a web application and secure it with Docker on an Ubuntu Linux OS

# Summary of Steps taken

1. Install Docker and Docker Compose
2. Prepare the server
3. Create the web application [Skip this step if you have your app]
4. Deploy the application
5. Secure the application

# Install Docker and Docker Compose on ubuntu machine

**Docker** for developing shipping and running application ---> containerising our application

**Docker compose** relies on Docker engine and helps to run multi-container applications. Services are defined in a docker-compose.yml file

## Install Docker

Reference document: [https://docs.docker.com/engine/install/ubuntu](https://docs.docker.com/engine/install/ubuntu/)

* Uninstall old version, if there is any

$ sudo apt-get remove docker docker-engine docker.io containerd runc

* Get update and install Docker engine

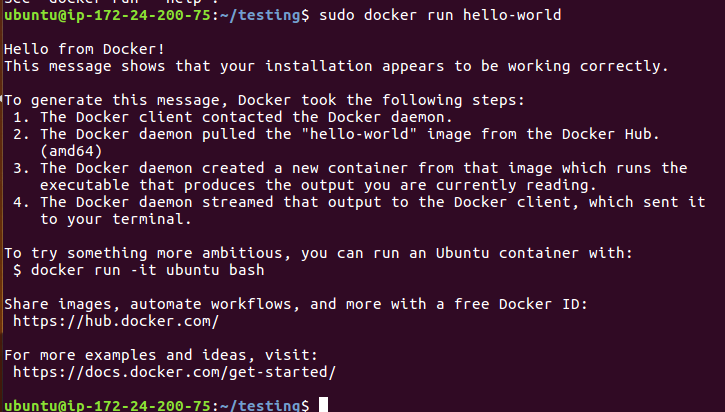
$ sudo apt-get update

$ sudo apt-get install docker-ce docker-ce-cli containerd.io

* Test to ensure docker is working

$ sudo docker run hello-world

You should see a message that your ***...installation appears to be working correctly.***



## Install Docker Compose

Reference document : <https://docs.docker.com/compose/install/>

* **Follow these lines of command**

$ sudo curl -L "https://github.com/docker/compose/releases/download/1.26.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

$sudo chmod +x /usr/local/bin/docker-compose

* **Test your installation by checking the version**

$ sudo docker-compose --version



Refer to the **Reference document** if you have any issue.

# Prepare the server

* create a working directory and navigate into the directory . Let’s call it ***testing*** in this case

$ mkdir testing

$ cd testing

* Create a docker-compose.yml file to use traefik. Traefik will act as the ingress router to direct incoming traffic into our application (expressjs app). Copy content below into the **docker-compose.yml file**

version: "3"

services:

traefik:

image: traefik:2.2

ports:

- "80:80"

volumes:

- /var/run/docker.sock:/var/run/docker.sock

- ./traefik.toml:/etc/traefik/traefik.toml:ro

container\_name: traefik

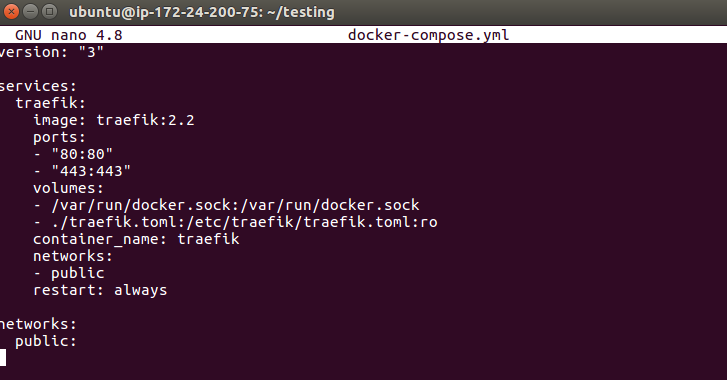
networks:

- public

restart: always

networks:

public:



* Create a **traefik.toml file** and paste the content below in the file

[entryPoints]

[entryPoints.web]

address = ":80"

[providers]

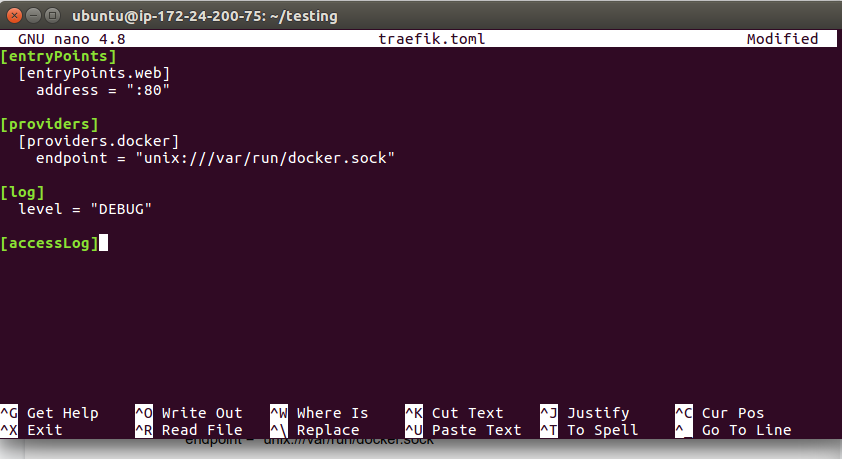
[providers.docker]

endpoint = "unix:///var/run/docker.sock"

[log]

level = "DEBUG"

[accessLog]



* **Start traefik by running :**

$ sudo docker compose up -d

You should get a message that it is done



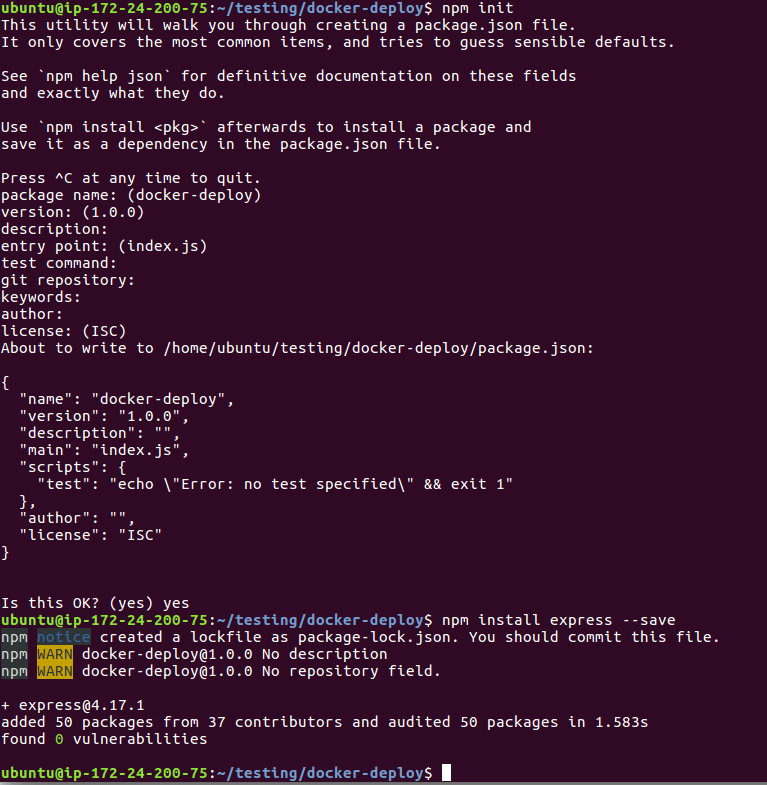
# Create the application

Here a sample ***expressjs*** application is created on the local machine. You can skip this step if you have your app

* Make a directory called **docker-deploy** and navigate into the directory and run the following commands. You might be prompted to install npm. Follow the instruction and use default setting by just pressing enter when requesting for packagename, version, description, entry point… etc

$mkdir docker-deploy

$ cd docker-deploy



You should see a folder that contains some packages called node\_modules and some json files



* Create a file called **index.js** and paste the content below;

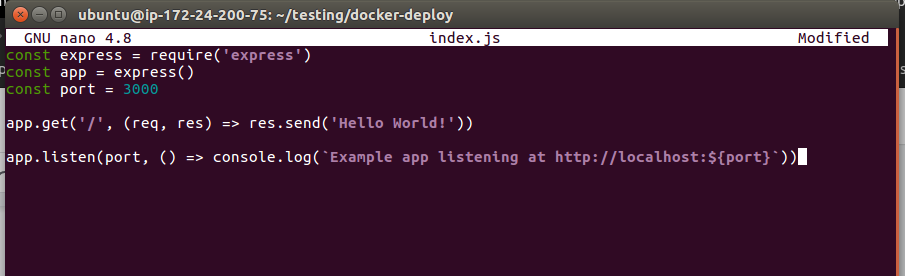
const express = require('express')

const app = express()

const port = 3000

app.get('/', (req, res) => res.send('Hello World!'))

app.listen(port, () => console.log(`Example app listening at [http://localhost:${port](about:blank)}`))



* Check to see that it is woking by running

$ node index.js

You should see something like



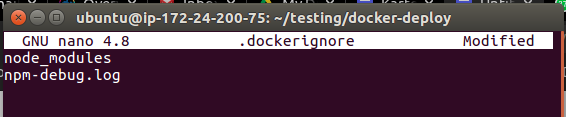
# Deploying the application

To start deploying the application we need to create a Docker image to hold our application from which a docker container will be created. We first need to create the files to ignore while building the image in a **.dockerignore** file.

* Create a .dockerignore file and paste the line blow inside it.

node\_modules

Npm-debug.log



* Create the **Dockerfile** and paste lines below inside it

FROM node:13.12-alpine3.10

WORKDIR /usr/src/app

COPY package\*.json ./

RUN npm ci --only=production

COPY . .

EXPOSE 3000

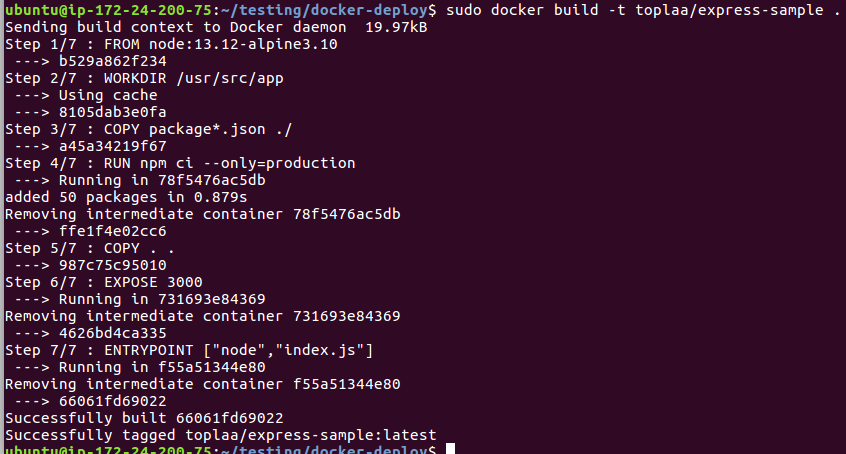
ENTRYPOINT ["node","index.js"]

* Before building the image, ensure that you have registered on docker hub. The name will be needed to push the docker image to dockerhub. Herein, my dockerhub name is **toplaa.** You replace toplaa with your docker hub name
* Create the image my running the command line below to also tag it

docker build -t **toplaa**/express-sample .

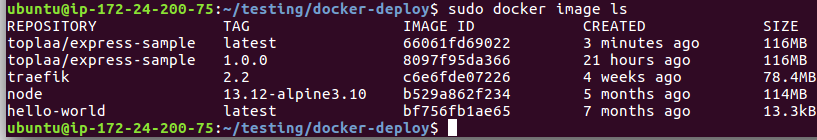
You will see

Successfully build ..



* You can check the image by running

$ sudo docker image ls



* Now that everything is ok, you will create a new container with it by running

$sudo docker run --rm -d -p 3000:3000 toplaa/express-sample



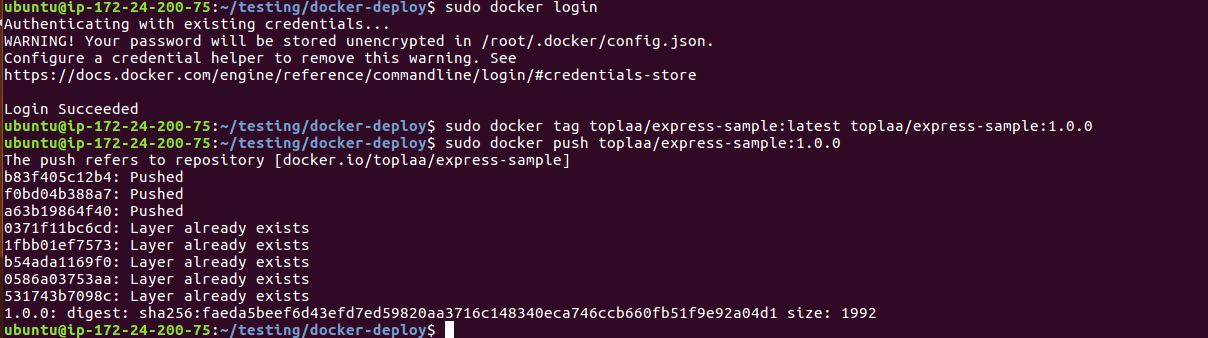
* Tag it with a new version and push the docker image to docker hub before deploying it. Follow the following steps:

$sudo docker login

$sudo docker tag toplaa/express-sample:latest toplaa/express-sample:1.0.0

$sudo docker push toplaa/express-sample:1.0.0

You should have something like this



* **Deploying our docker image**

Here we will use our docker compose file to define our service. The key point here is that we are making the environment **public**

Create a **docker-compose.yml** file and paste the code below inside

version: "3"

services:

express:

image: toplaa/express-sample:1.0.0

labels:

- "traefik.http.routers.express-sample.rule=Host(`express-sample.toplaa.com`)"

- "traefik.http.services.express-sample.loadbalancer.server.port=3000"

container\_name: express-sample

networks:

- env\_public

restart: unless-stopped

networks:

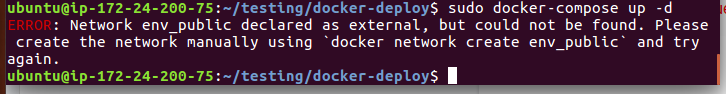
env\_public:

external: true

* Deploy the image by using

sudo docker-compose up -d

You might encounter an error such as the one below. Follow the instruction to trouble shoot manually



# Securing the application

Now we need to secure our application from HTTP to HTTPS using SSL

* Edit the traefik.toml file with the following lines of command

traefik:

image: traefik:2.2

labels:

- "traefik.http.middlewares.redirect-to-https.redirectscheme.scheme=https"

- "traefik.http.routers.global-redirect.rule=HostRegexp(`{host:.+}`)"

- "traefik.http.routers.global-redirect.entrypoints=web"

- "traefik.http.routers.global-redirect.middlewares=redirect-to-https"

ports:

- "80:80"

- "443:443"

* Also add the following lines to the docker-compose file.yml file

labels:

- "traefik.http.routers.express-sample.tls=true"

- "traefik.http.routers.express-sample.tls.certresolver=myresolver

* Finally run the command below to implement the services

docker-compose up -d

# Simplified Architecture

