

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>

- 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.*

```

ubuntu@ip-172-24-200-75:~/testing$ sudo docker run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/

ubuntu@ip-172-24-200-75:~/testing$

```

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-$(unam
e -s)-$(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

```

```

ubuntu@ip-172-24-200-75:~$ sudo docker-compose --version
docker-compose version 1.26.2, build eefe0d31
ubuntu@ip-172-24-200-75:~$

```

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:
```

```
ubuntu@ip-172-24-200-75: ~/testing
GNU nano 4.8 docker-compose.yml
version: "3"

services:
  traefik:
    image: traefik:2.2
    ports:
      - "80:80"
      - "443:443"
    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]

```
ubuntu@ip-172-24-200-75: ~/testing
GNU nano 4.8 traefik.toml Modified
[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

```
ubuntu@ip-172-24-200-75:~/testing$ sudo docker-compose up -d
Recreating traefik ... done
ubuntu@ip-172-24-200-75:~/testing$
```

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 package name, version, description, entry point... etc
\$ mkdir docker-deploy
\$ cd docker-deploy

```

ubuntu@ip-172-24-200-75:~/testing/docker-deploy$ npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help json` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (docker-deploy)
version: (1.0.0)
description:
entry point: (index.js)
test command:
git repository:
keywords:
author:
license: (ISC)
About to write to /home/ubuntu/testing/docker-deploy/package.json:

{
  "name": "docker-deploy",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC"
}

Is this OK? (yes) yes
ubuntu@ip-172-24-200-75:~/testing/docker-deploy$ npm install express --save
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN docker-deploy@1.0.0 No description
npm WARN docker-deploy@1.0.0 No repository field.

+ express@4.17.1
added 50 packages from 37 contributors and audited 50 packages in 1.583s
found 0 vulnerabilities

ubuntu@ip-172-24-200-75:~/testing/docker-deploy$

```

You should see a folder that contains some packages called `node_modules` and some json files

```

ubuntu@ip-172-24-200-75:~/testing/docker-deploy$ ls
node_modules package-lock.json package.json

```

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

```

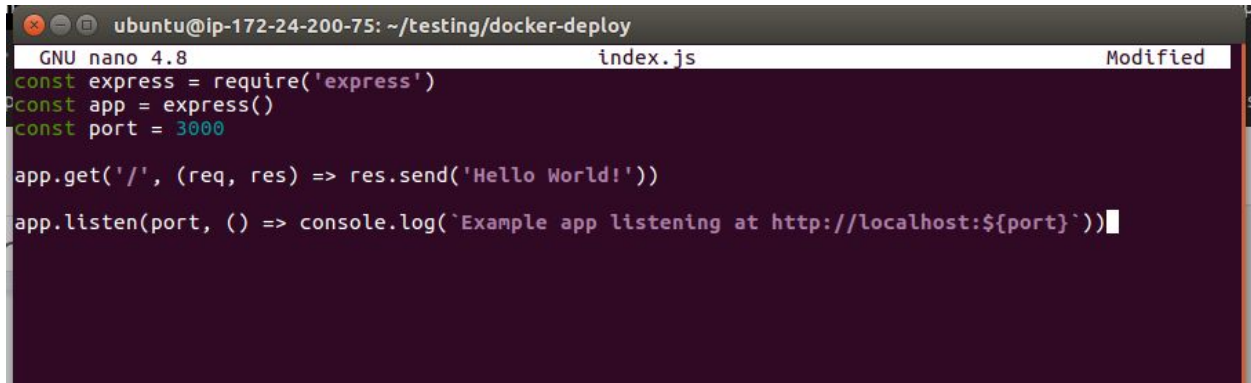
const express = require('express')
const app = express()
const port = 3000

```



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

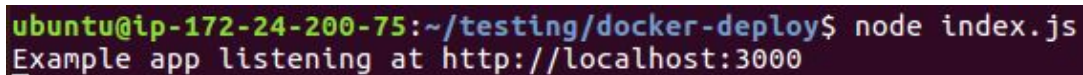
```
app.listen(port, () => console.log(`Example app listening at http://localhost:\${port}`))
```



```
ubuntu@ip-172-24-200-75: ~/testing/docker-deploy
GNU nano 4.8 index.js Modified
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}`))
```

- Check to see that it is working by running
\$ node index.js
You should see something like



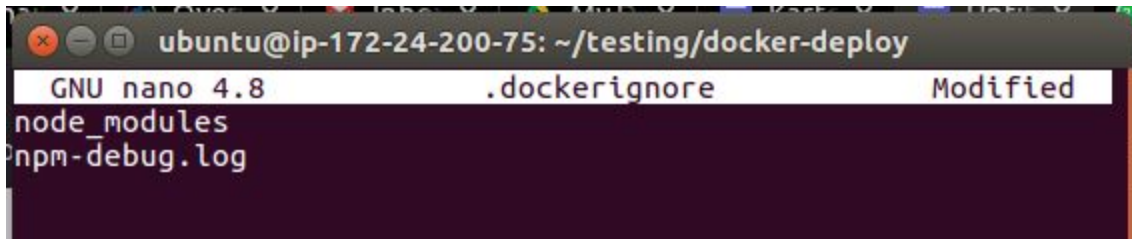
```
ubuntu@ip-172-24-200-75:~/testing/docker-deploy$ node index.js
Example app listening at http://localhost:3000
```

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 lines below 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 by running the command line below to also tag it

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

You will see

Successfully build ..


```

ubuntu@ip-172-24-200-75:~/testing/docker-deploy$ sudo docker build -t toplaa/express-sample .
Sending build context to Docker daemon 19.97kB
Step 1/7 : FROM node:13.12-alpine3.10
--> b529a862f234
Step 2/7 : WORKDIR /usr/src/app
--> Using cache
--> 8105dab3e0fa
Step 3/7 : COPY package*.json ./
--> a45a34219f67
Step 4/7 : RUN npm ci --only=production
--> Running in 78f5476ac5db
added 50 packages in 0.879s
Removing intermediate container 78f5476ac5db
--> ffe1f4e02cc6
Step 5/7 : COPY . .
--> 987c75c95010
Step 6/7 : EXPOSE 3000
--> Running in 731693e84369
Removing intermediate container 731693e84369
--> 4626bd4ca335
Step 7/7 : ENTRYPOINT ["node","index.js"]
--> Running in f55a51344e80
Removing intermediate container f55a51344e80
--> 66061fd69022
Successfully built 66061fd69022
Successfully tagged toplaa/express-sample:latest
ubuntu@ip-172-24-200-75:~/testing/docker-deploy$

```

- You can check the image by running
\$ sudo docker image ls

```

ubuntu@ip-172-24-200-75:~/testing/docker-deploy$ sudo docker image ls
REPOSITORY          TAG             IMAGE ID        CREATED         SIZE
toplaa/express-sample latest          66061fd69022   3 minutes ago  116MB
toplaa/express-sample 1.0.0          8097f95da366   21 hours ago   116MB
traefik              2.2            c6e6fde07226   4 weeks ago    78.4MB
node                  13.12-alpine3.10 b529a862f234   5 months ago   114MB
hello-world           latest         bf756fb1ae65    7 months ago   13.3kB
ubuntu@ip-172-24-200-75:~/testing/docker-deploy$

```

- 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

```

ubuntu@ip-172-24-200-75:~/testing/docker-deploy$ sudo docker run --rm -d -p 3000:3000 toplaa/express-sample
4cc4b94a34266288e972553870c7c91395c6d0c05195918d2533b64f122aa998
ubuntu@ip-172-24-200-75:~/testing/docker-deploy$

```

- 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

```
ubuntu@tp-172-24-200-75:~/testing/docker-deploy$ sudo docker login
Authenticating with existing credentials...
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
ubuntu@tp-172-24-200-75:~/testing/docker-deploy$ sudo docker tag toplaa/express-sample:latest toplaa/express-sample:1.0.0
ubuntu@tp-172-24-200-75:~/testing/docker-deploy$ sudo docker push toplaa/express-sample:1.0.0
The push refers to repository [docker.io/toplaa/express-sample]
b83f405c12b4: Pushed
f0bd04b388a7: Pushed
a63b19864f40: Pushed
0371f11bc6cd: Layer already exists
1fbb01ef7573: Layer already exists
b54ada1169f0: Layer already exists
0586a03753aa: Layer already exists
531743b7098c: Layer already exists
1.0.0: digest: sha256:faeda5beef6d43efd7ed59820aa3716c148340eca746ccb660fb51f9e92a04d1 size: 1992
ubuntu@tp-172-24-200-75:~/testing/docker-deploy$
```

- **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

```
ubuntu@ip-172-24-200-75:~/testing/docker-deploy$ sudo docker-compose up -d
ERROR: Network env_public declared as external, but could not be found. Please
create the network manually using `docker network create env_public` and try
again.
ubuntu@ip-172-24-200-75:~/testing/docker-deploy$
```

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

