**Deploy Angular Application in Docker Container**

By

Sayan Bhattacharjee

M1056174

Mindtree Ltd.

Email –sayan.bhattacharjee2@mindtree.com

# Problem Statement

HTQual Technology Solutions hired you as a MEAN Stack Developer. The organization decided to implement DevOps to develop and deliver the products. Since HTQual is an Agile organization, they follow Scrum methodology to develop the projects incrementally. The Company decided to develop their website on Mean stack. Since you are the MEAN stack developer, you have to demonstrate that deploying the Angular application on Docker is always the best approach to develop a project and test it incrementally. You agreed upon the following:

    • Setting up an image for code development

    • Build the application in Docker and host it in Docker Hub

    • List the advantages, disadvantages, and document the tasks involved

Your goal is to demonstrate the Angular application and run it in Docker container.

Setting up an image for code development

## **This will involve following steps:**

**Step 1**

First of all for any operating system whether it is windows or linux for running angular into your system you must have node modules install in your operating system.

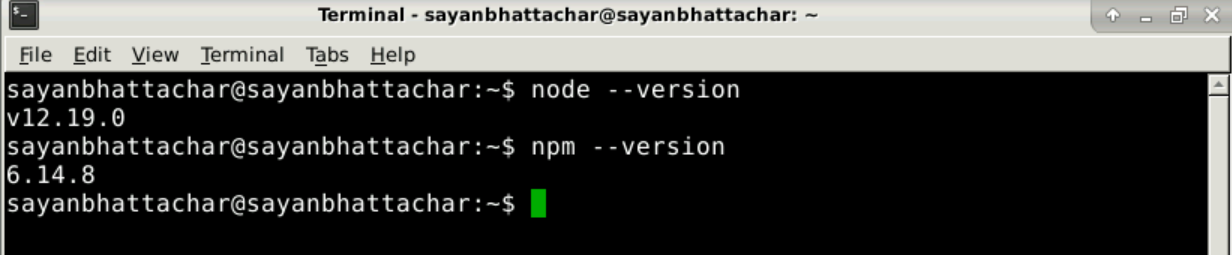
You can check whether it is there in your os or not by following command

**node --version**

if it is not present in your Linux system you have to install the node modules by following command

**curl -sL https://deb.nodesource.com/setup\_12.x | sudo -E bash -**

**sudo apt-get install -y nodejs**



**Step 2**

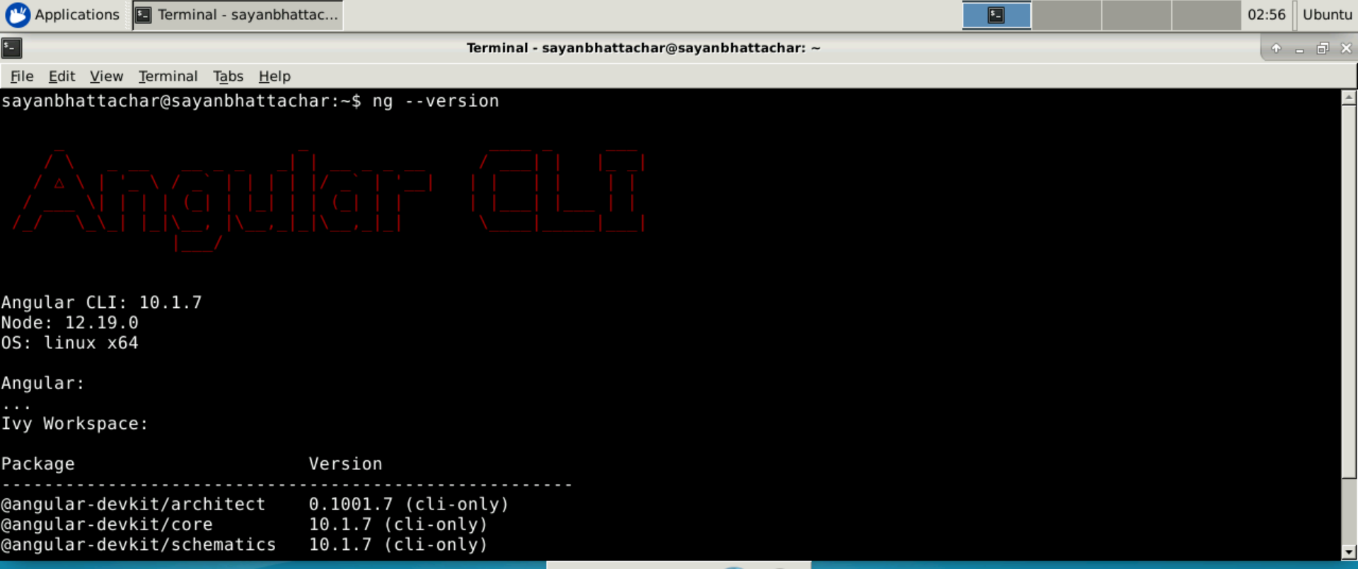
So now once you have install node into your system you can also install an angular into your system

By following command

**npm install -g @angular/cli**

after you can also check which version you have installed by above command for this you need to write

**ng –version**

****

**Step 3**

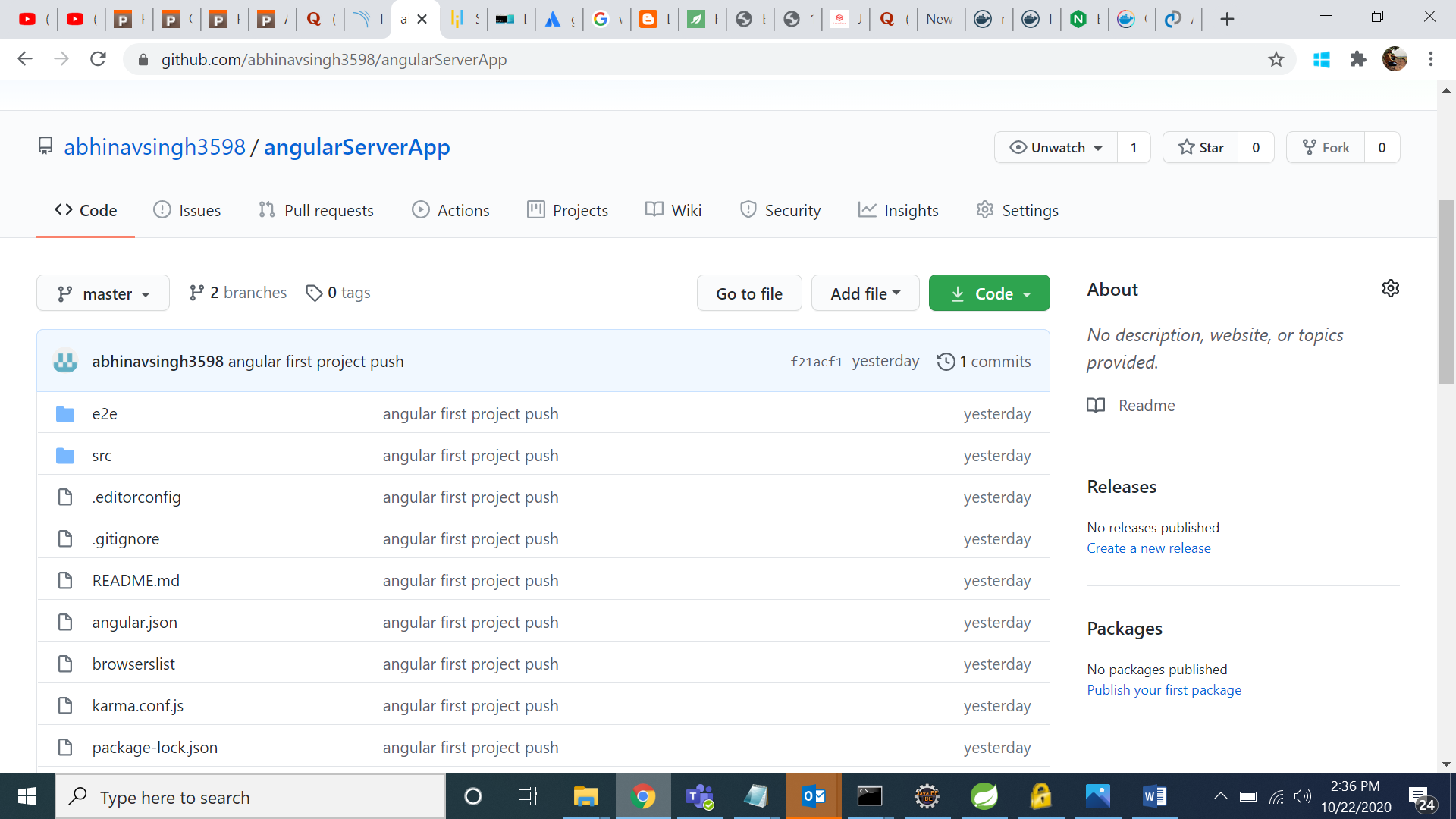
Once you have install angular in your os now you need a angular to run so that you can create a docker image out of it

So in my case I have created a project already which is already present in git hub and I will clone that project into my system

For that you have to clone that project in a folder you want

So the command for cloning my angular project is

**git clone** [**https://github.com/abhinavsingh3598/angularServerApp.git**](https://github.com/abhinavsingh3598/angularServerApp.git)

****

**Step 4**

Now if you have clone the project after that you need to install the node modules for angular app so that project will run and build successfully and all necessary modules will get install for the project

Command for installing the node modules is

**npm install or npm i**

**Step 5**

For building the angular app in your directory the command you will use is

**ng build –prod**

it will create dist folder in your directory

**Step 6**

Soo now once you are done with your angular project installation, building and all related things now work for Docker will begin

In order to create a docker image for a particular project you need Docker file which is nothing but a set of instructions which is written to generate a Docker image out of it

Command to create a dockerfile is

**touch Dockerfile**

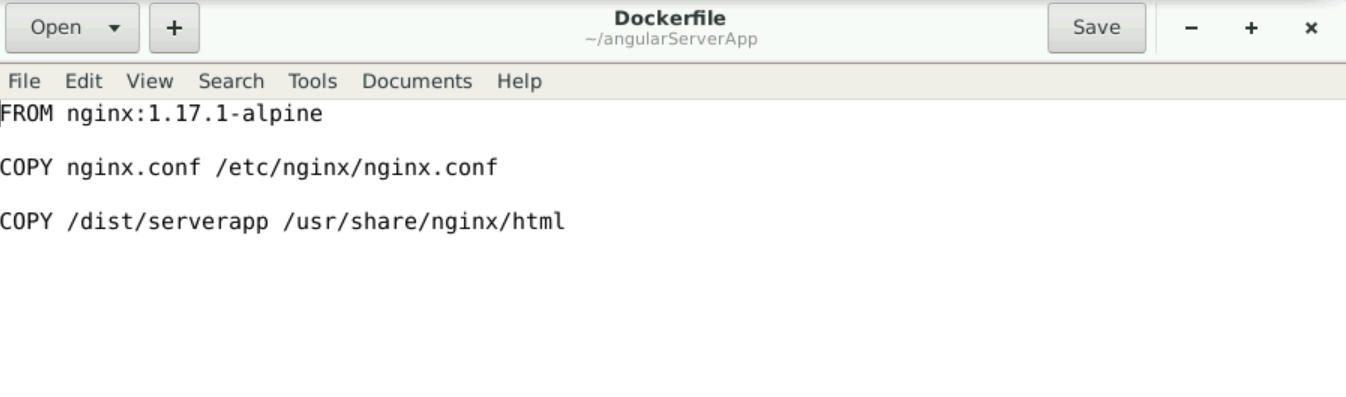
**note : make sure f should be small**

**Step 7**

if you want to edit anything and open the file you can use any editor vi, gedit and lots of others are present

**gedit Dockerfile**

Content in docker file inside project directory



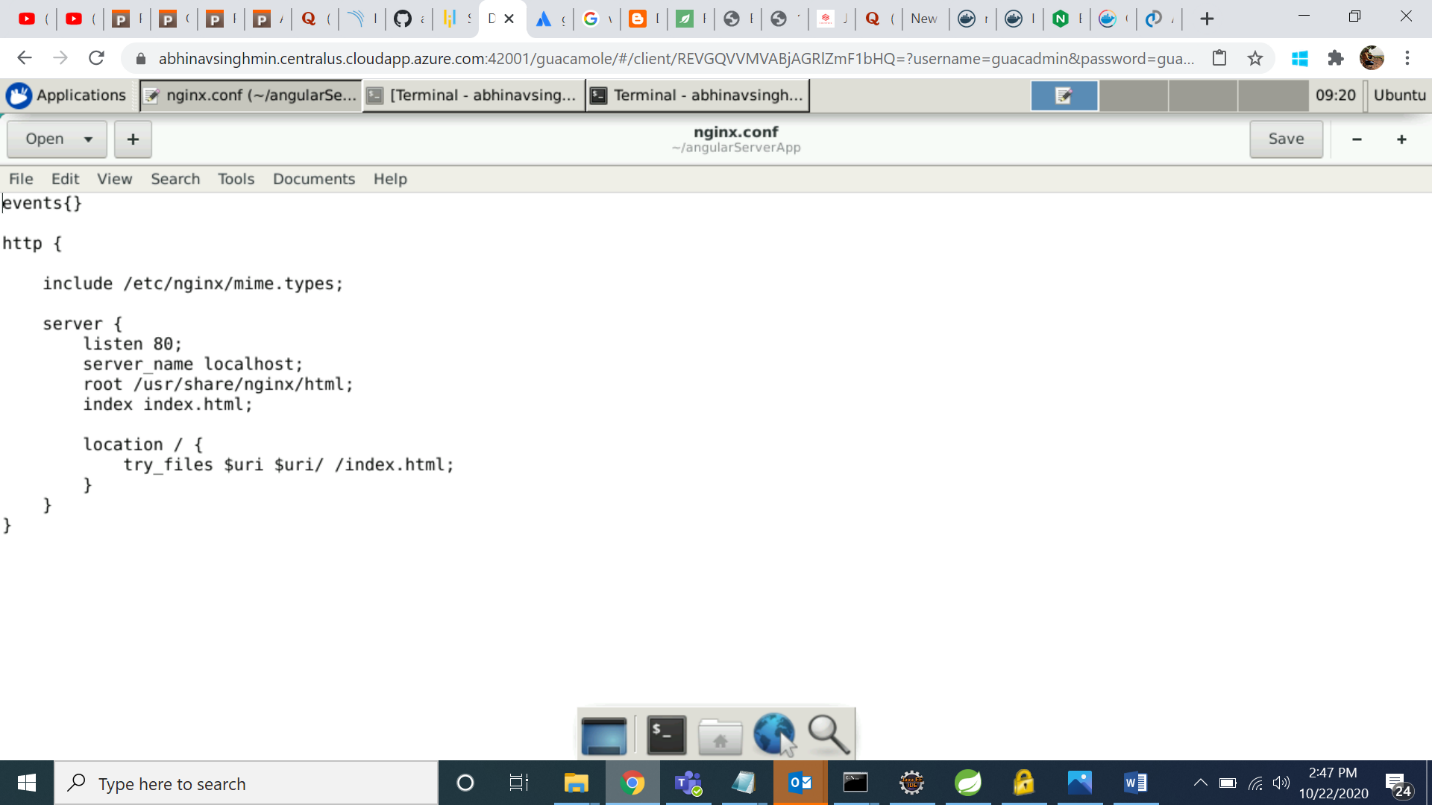
first to get a nginx Docker image from Docker Hub tagged with 1.17.1-alpine (it’s like a version number),

then copy-paste the default nginx configuration,

and finally copy-paste the compiled application (we done it in previous step) to the container.

**Step 8**

create a default nginx.conf file inside project directory



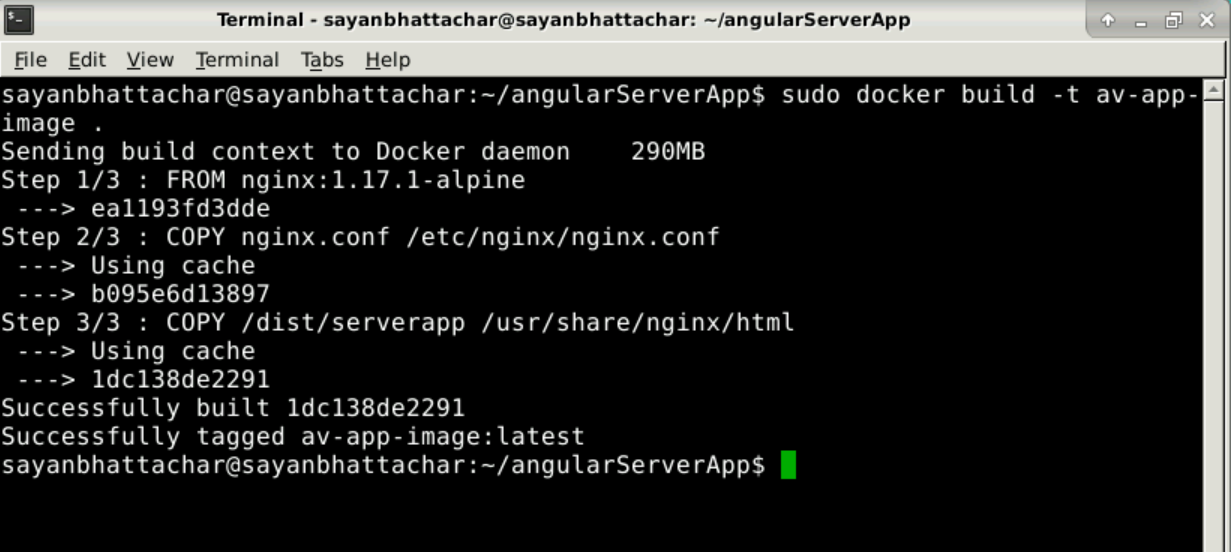
nginx consists of modules which are controlled by directives specified in the configuration file. Directives are divided into simple directives and block directives. A simple directive consists of the name and parameters separated by spaces and ends with a semicolon (;). A block directive has the same structure as a simple directive, but instead of the semicolon it ends with a set of additional instructions surrounded by braces ({ and }). If a block directive can have other directives inside braces, it is called a context (examples: [events](http://nginx.org/en/docs/ngx_core_module.html#events), [http](http://nginx.org/en/docs/http/ngx_http_core_module.html#http), [server](http://nginx.org/en/docs/http/ngx_http_core_module.html#server), and [location](http://nginx.org/en/docs/http/ngx_http_core_module.html#location)).

This location block specifies the “/” prefix compared with the URI from the request. For matching requests, the URI will be added to the path specified in the [root](http://nginx.org/en/docs/http/ngx_http_core_module.html#root) directive, that is, to /data/www, to form the path to the requested file on the local file system.

**Step 9**

To create Docker image from dockerfile we need to build it by following command but in our usecase it is not require becuase we will be using docker compose for creating image

**docker build -t av-app-image .**

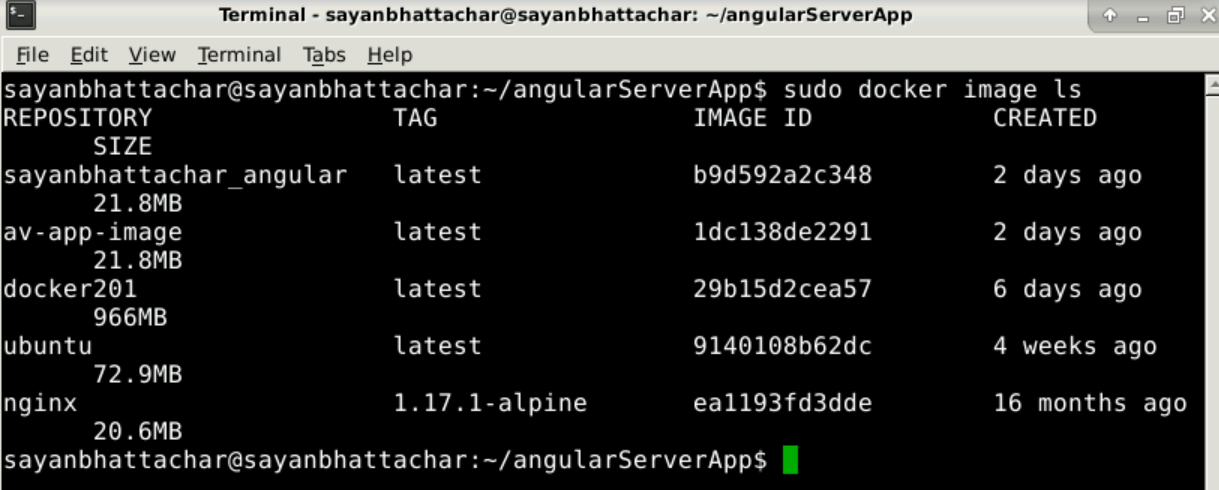
ll

for running Docker container

**docker run --name av-app-container -d -p 8080:80 av-app-image**

to verify image is created

**docker image ls**



to verify container

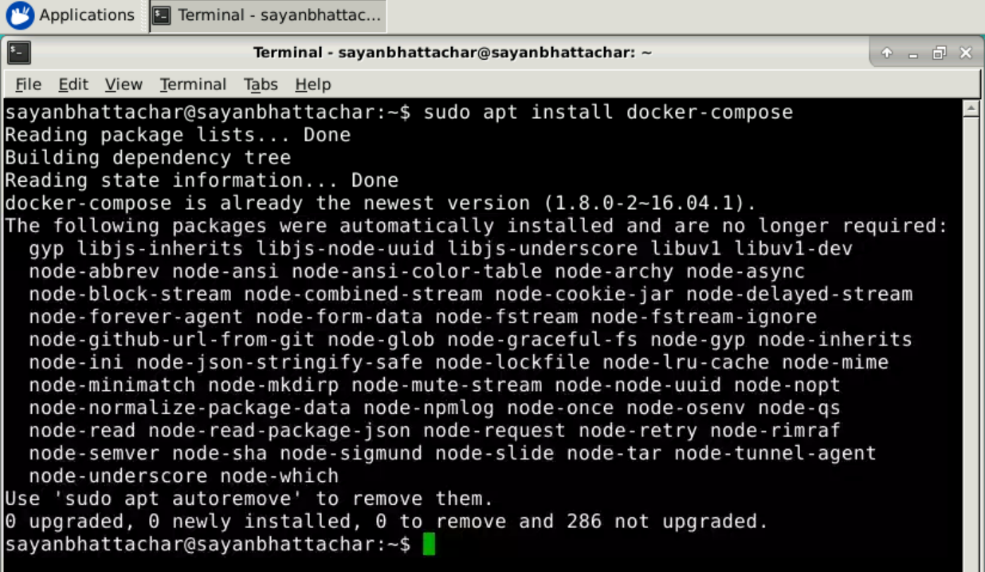
**docker container ls**

**Step 10**

Compose is a tool for defining and running multi-container Docker applications. With Compose, you use a YAML file to configure your application’s services. Then, with a single command, you create and start all the services from your configuration

to install dockercompose into your directory

**sudo apt install docker-compose**



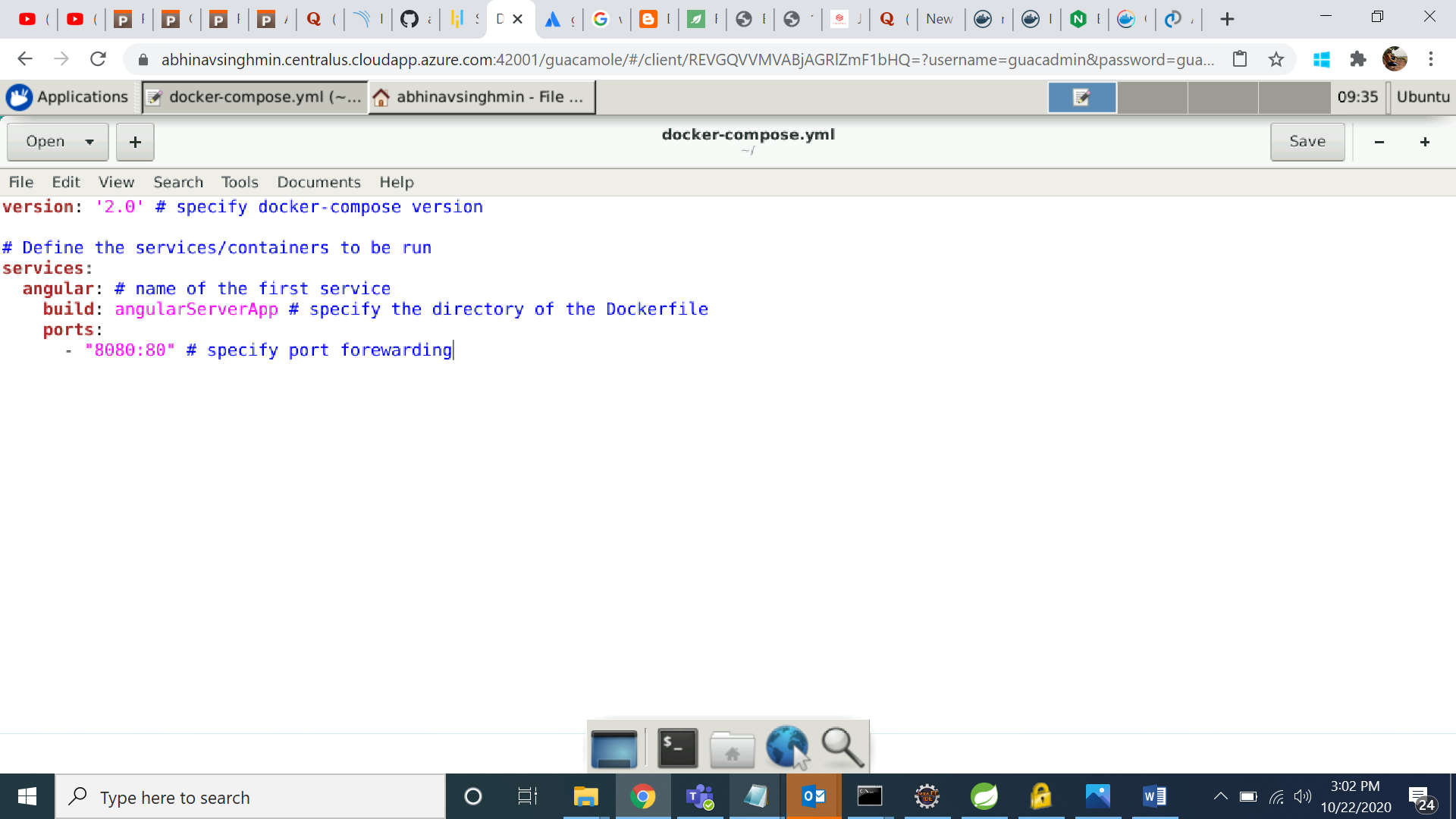
**Step 11**

To create dockercompose file

**touch docker-compose.yml**

**Step 12**

to open it with **gedit docker-compose.yml**



Content inside the docker-compose file is

**version: '2' # specify docker-compose version**

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

**services:**

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

**build: angularServerApp # specify the directory of the Dockerfile**

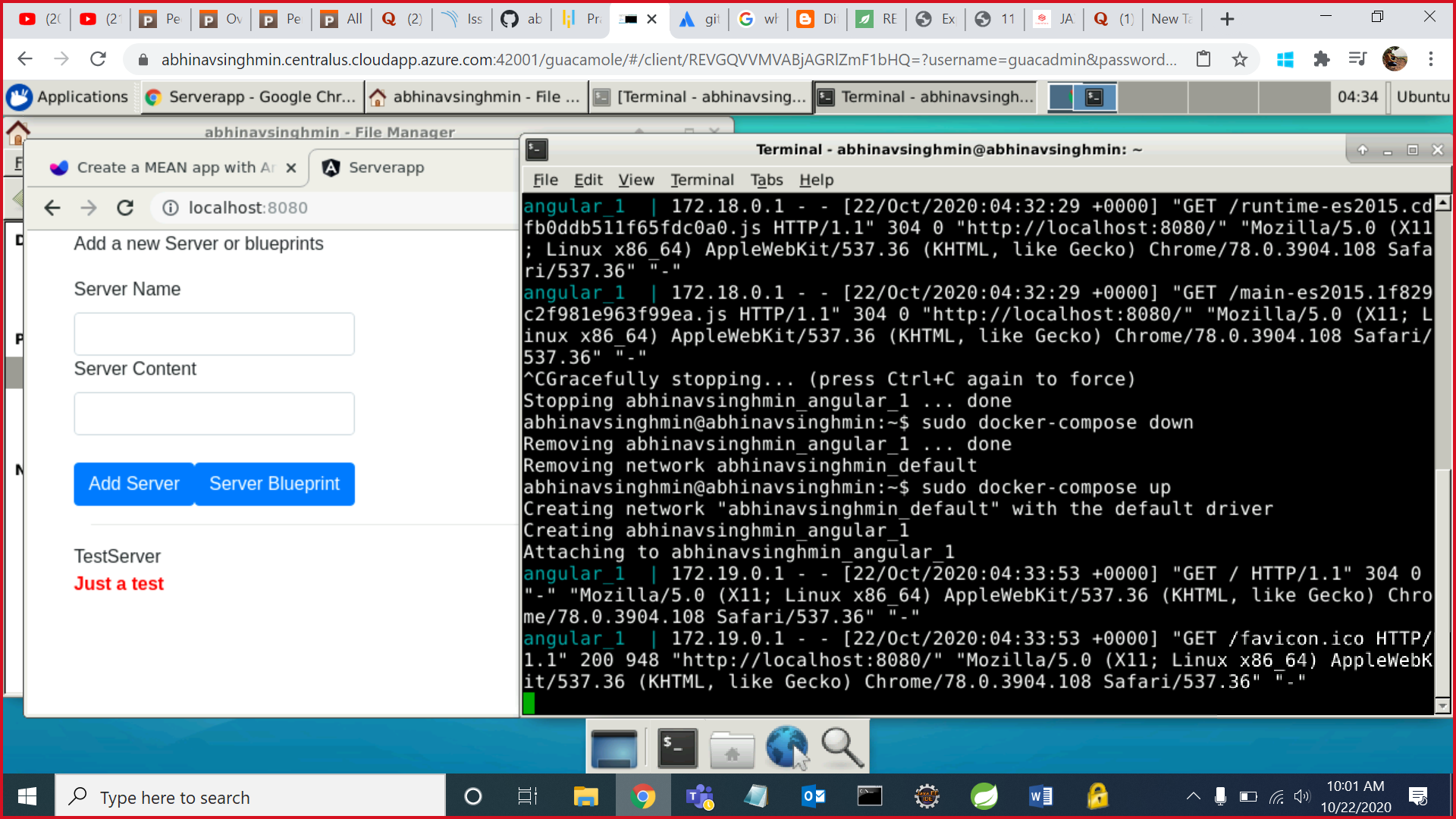
**ports:**

**- "8080:80" # specify port forewarding**

**Step 13**

To start the docker compose file

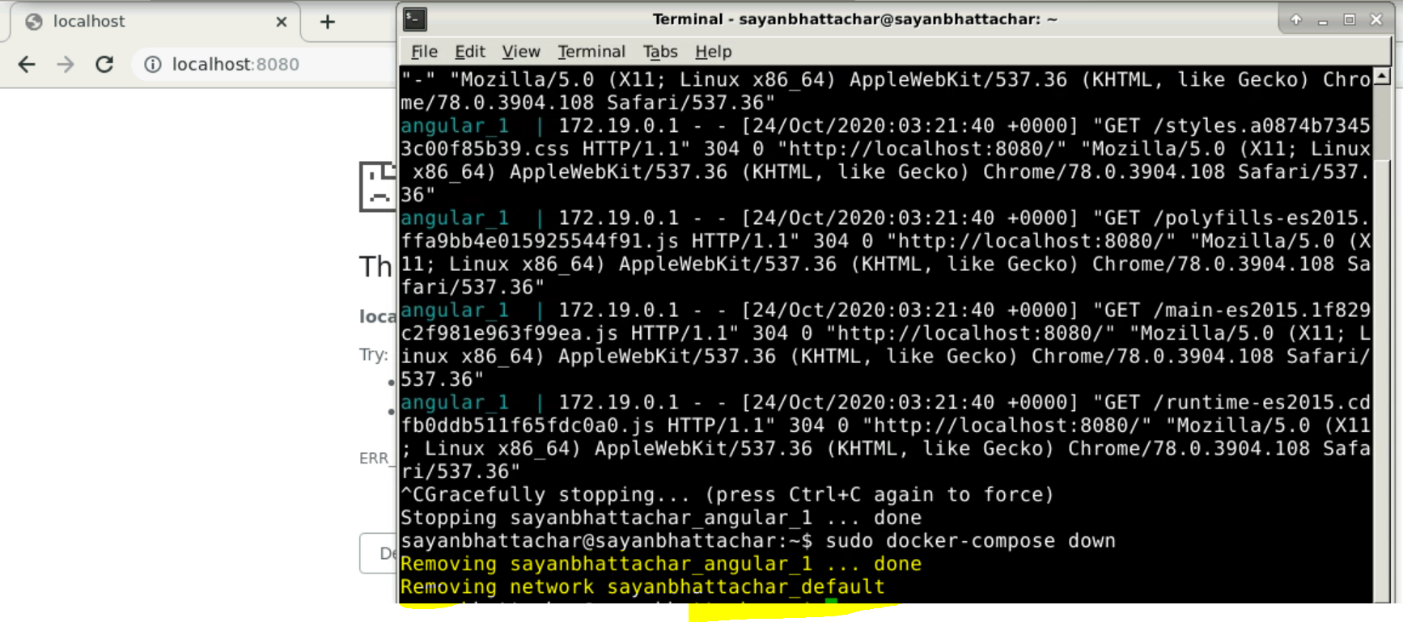
**docker-compose up**

****

See on browser we are running our angular app on 8080 which we assigned in the docker-compose.yml file and when we have start compose file it is running

to stop the docker-compose file

**docker-compose down**

****

now when we stop the server then it is not running on browser on left hand side

**Step 14**

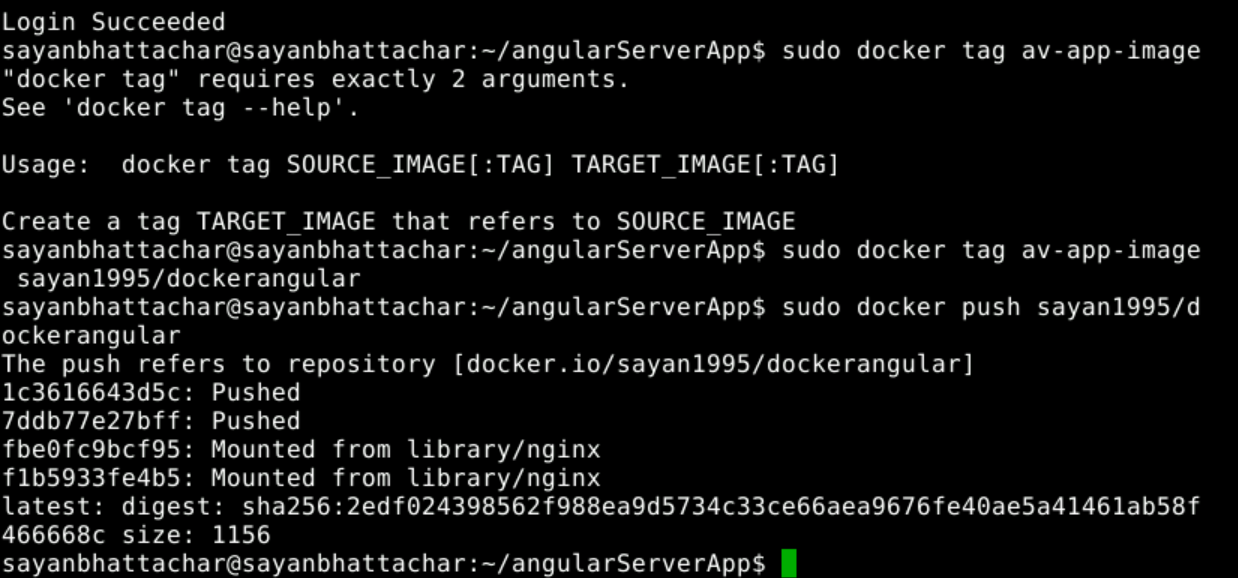
Create an account on docker hub because now we will push our image to docker hub repository

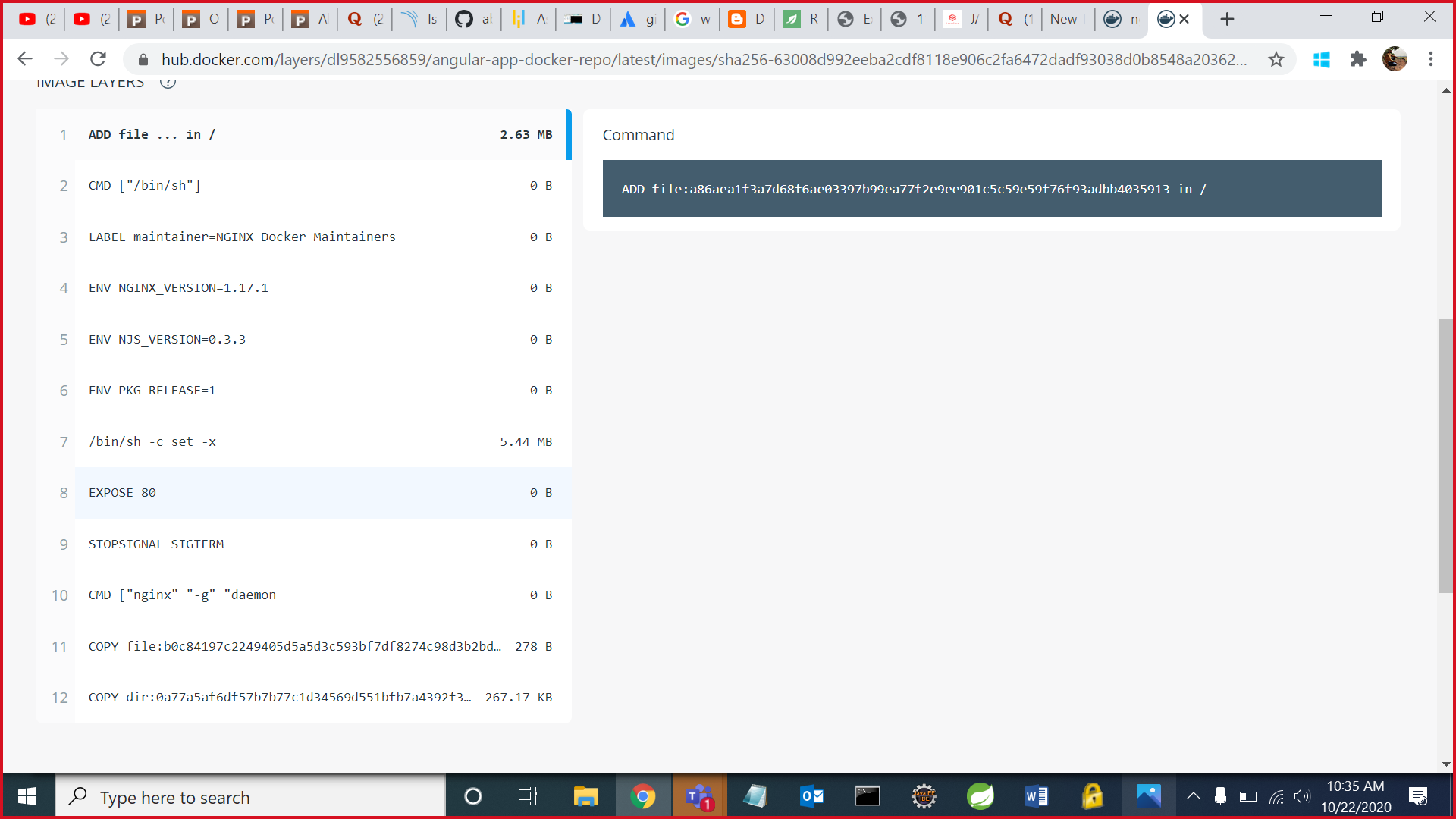
Create a repository after account creation

Remember the credentials username and password. Once logged in run the below two commands for pushing into the dockerhub.

**1)sudo docker tag av-app-image sayan1995/dockerangular**

**2)sudo docker push sayan1995/dockerangular**





# **Advantages and Disadvantages of Docker**

### **Benefits of Docker**

#### **Return on Investment and Cost Savings**

Dockers first advantage is ROI. Especially for large, established companies, which need to generate steady revenue over the long term, the solution is only better if it can drive down costs while raising profits.

#### **Rapid Deployment**

It can decrease deployment to seconds. It is because of the fact that it can create a container for every process and even does not boot an OS. So, even without worrying about the cost to bring it up again, it would be higher than what is affordable, Data can be created as well as destroyed.

#### **Security**

Docker makes sure that applications that are running on containers are completely segregated and isolated from each other, from a security point of view, by granting us complete control over traffic flow and management.

#### **Simplicity and Faster Configurations**

The way Docker simplifies the matters is one of the key benefits of it. It gives flexibility to users to take their own configuration, put that into the code, and further deploy it without any problems. However, the requirements of the infrastructure are no longer linked with the environment of the application, as Docker can be used in a wide variety of environments.

#### **CI Efficiency**

With the help of a Docker, we can build a container image and can further use that same image over every step of the deployment process. The advantage of it is the ability to separate non-dependent steps and also run them in parallel. In addition, the duration of time it takes from build to production may speed up notably.

### **Limitations of Docker**

#### **Missing features**

There are a ton of feature requests are under progress, like container self-registration, and self-inspects, copying files from the host to the container, and many more.

#### **Data in the container**

There are times when a container goes down, so after that, it needs a backup and recovery strategy, although we have several solutions for that they are not automated or not very scalable yet.

#### **Run applications as fast as a bare-metal serve**

In comparison with the virtual machines, Docker containers have less overhead but not zero overhead. If we run, an application directly on a bare-metal server we get true bare-metal speed even without using containers or virtual machines. However, Containers don’t run at bare-metal speeds.

#### **Provide cross-platform compatibility**

The one major issue is if an application designed to run in a Docker container on Windows, then it can’t run on Linux or vice versa. However, Virtual machines are not subject to this limitation. So, this limitation makes Docker less attractive in some highly heterogeneous environments which are composed of both Windows and Linux servers.

#### **Run applications with graphical interfaces**

In general, Docker is designed for hosting applications which run on the command line. Though we have a few ways (like X11 forwarding) by which we can make it possible to run a graphical interface inside a Docker container, however, this is clunky. Hence we can say, for applications that require rich interfaces, Docker is not a good solution.

#### **Solve all your security problems**

In simple words,  we need to evaluate the Docker-specific security risks and make sure we can handle them before moving workloads to Docker. The reason behind it is that Docker creates new security challenges like the difficulty of monitoring multiple moving pieces within a large-scale, dynamic Docker environment.