Lab 2 Report

50.046 CCIoT

SUTD 2022 ISTD

Group 3

Han Xing Yi	(1004330)
Ang Song Gee	(1004589)
James Raphael Tiovalen	(1004555)
Velusamy Sathiakumar Ragul Balaji	(1004101)

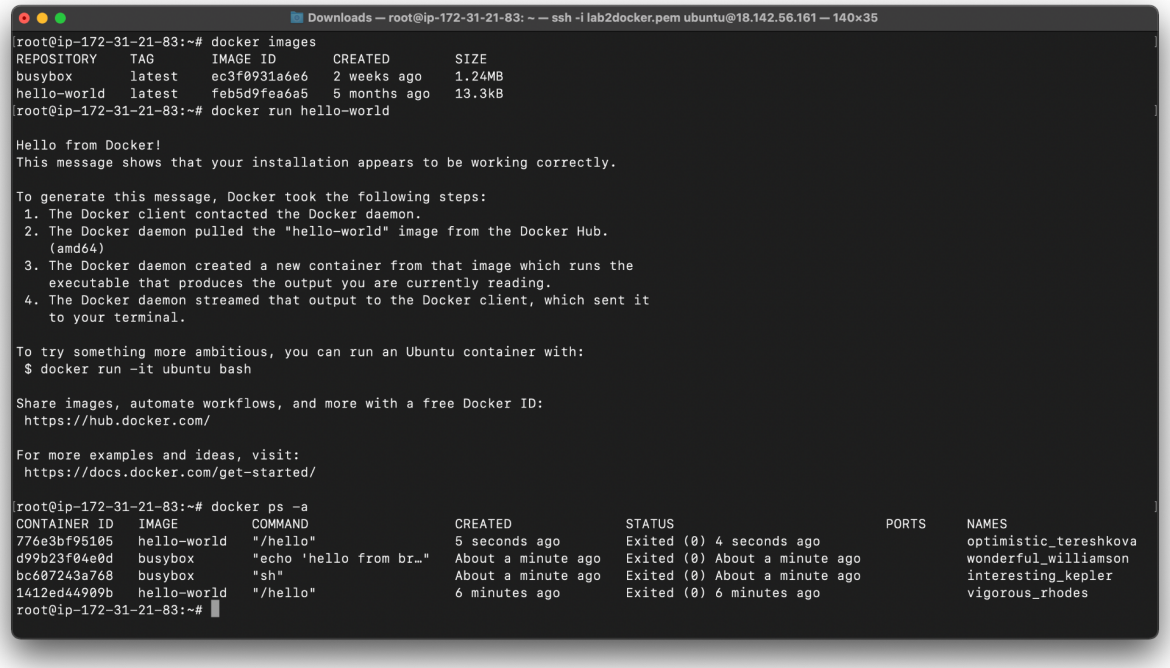
Table of Contents

Task 1: Set up a Docker Environment

✅ Print all local docker images with `docker images`.

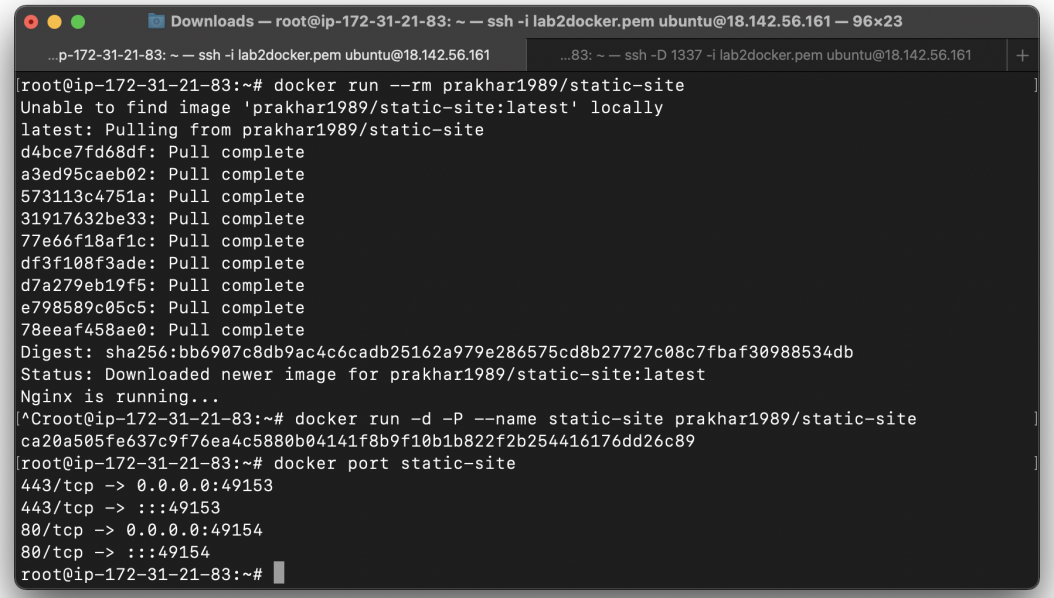
✅ Run the 'hello-world' container in docker with `docker run hello-world`.

✅ Print all running docker process status with `docker ps -a`.

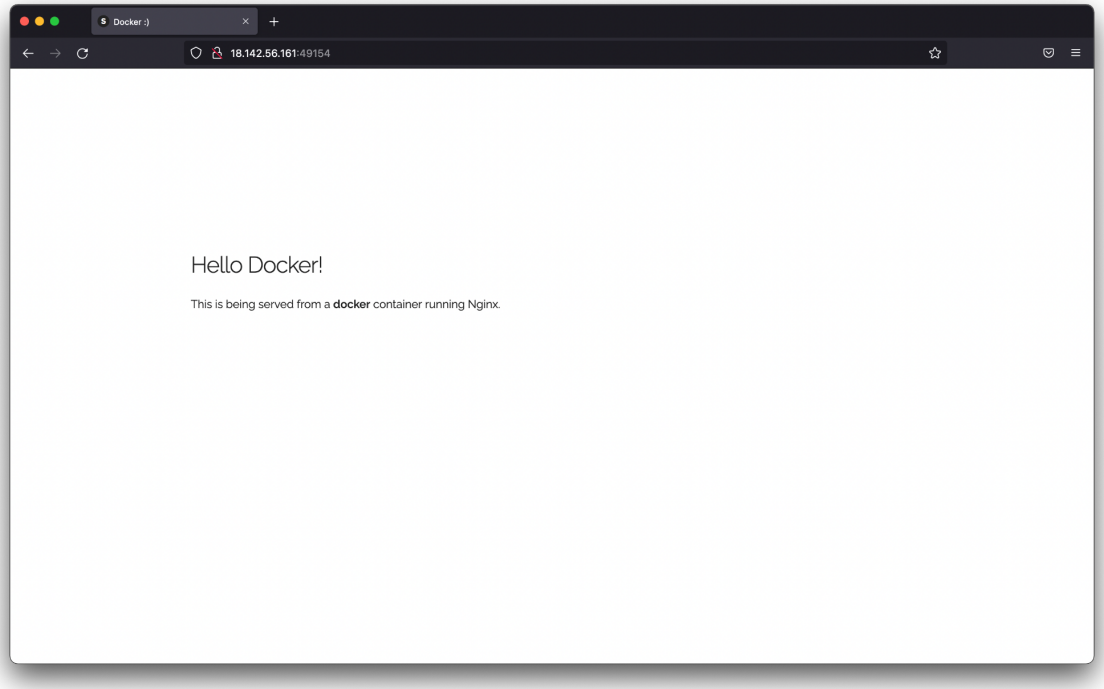


Task 2: Run a Webapp with Docker

✅ Set up static site with `docker run prakhar1989/static-site`.

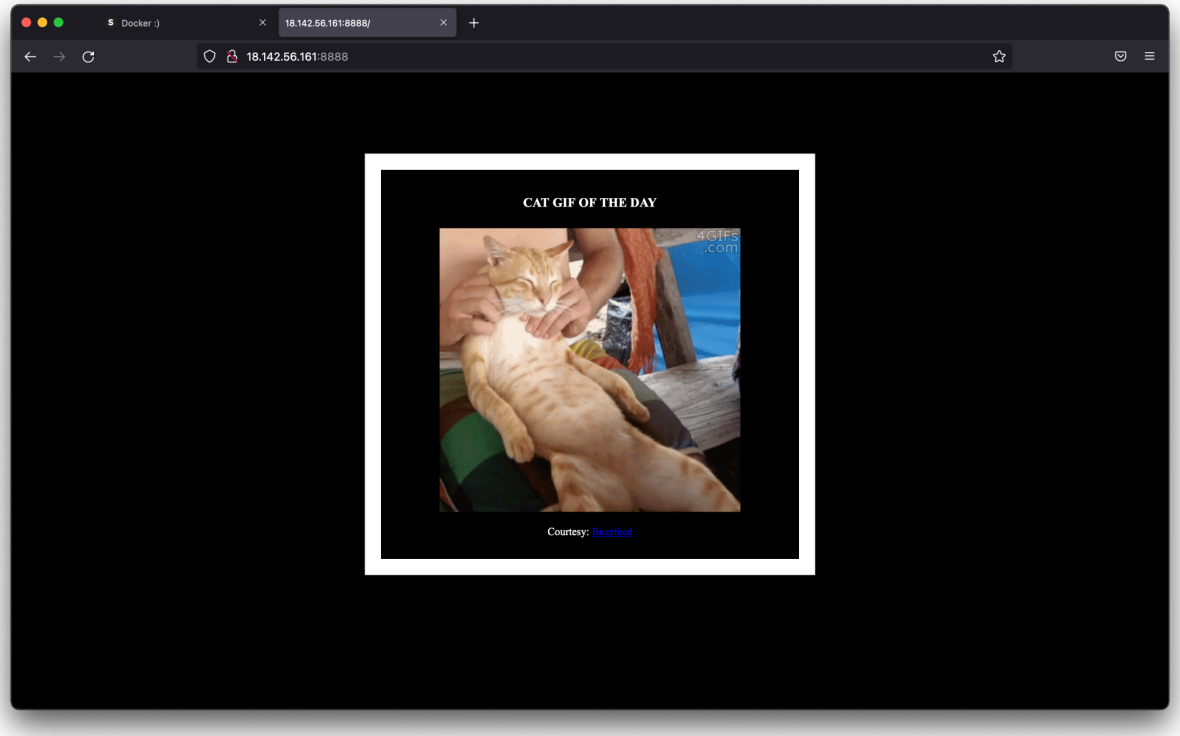


✅ Server running properly at localhost



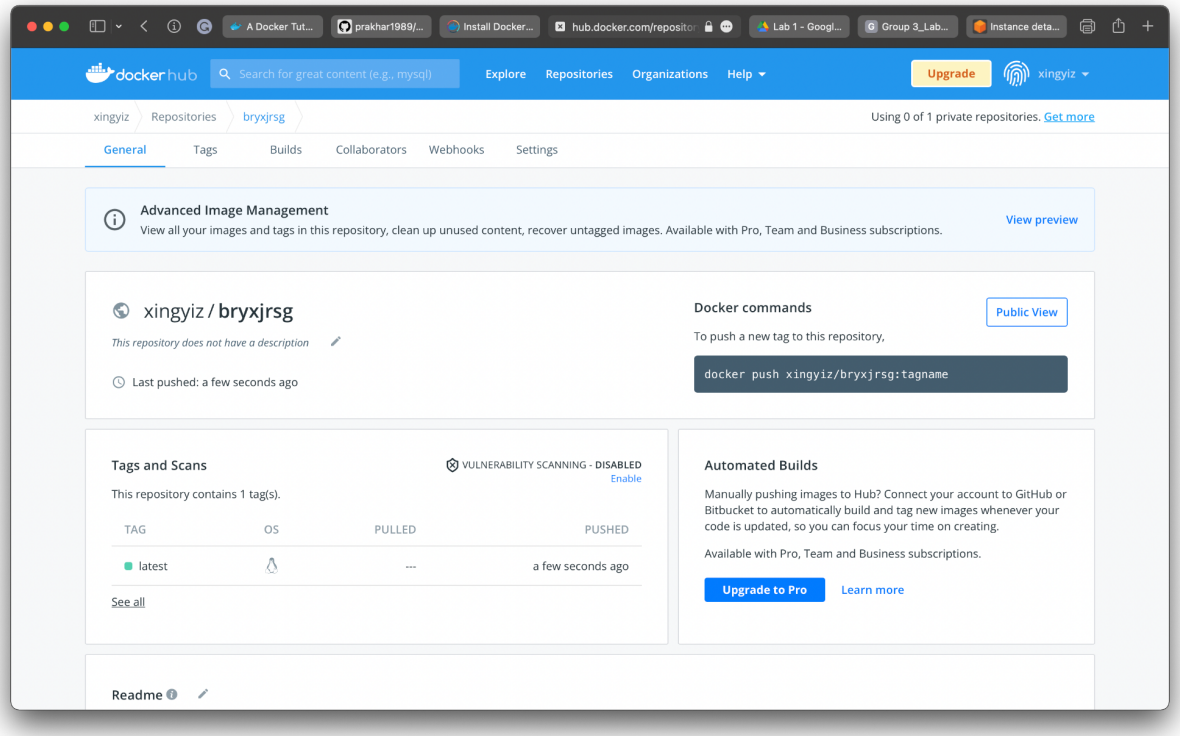
Task 3: Make Your First Docker Image

✅ Compile Dockerfile and run on localhost

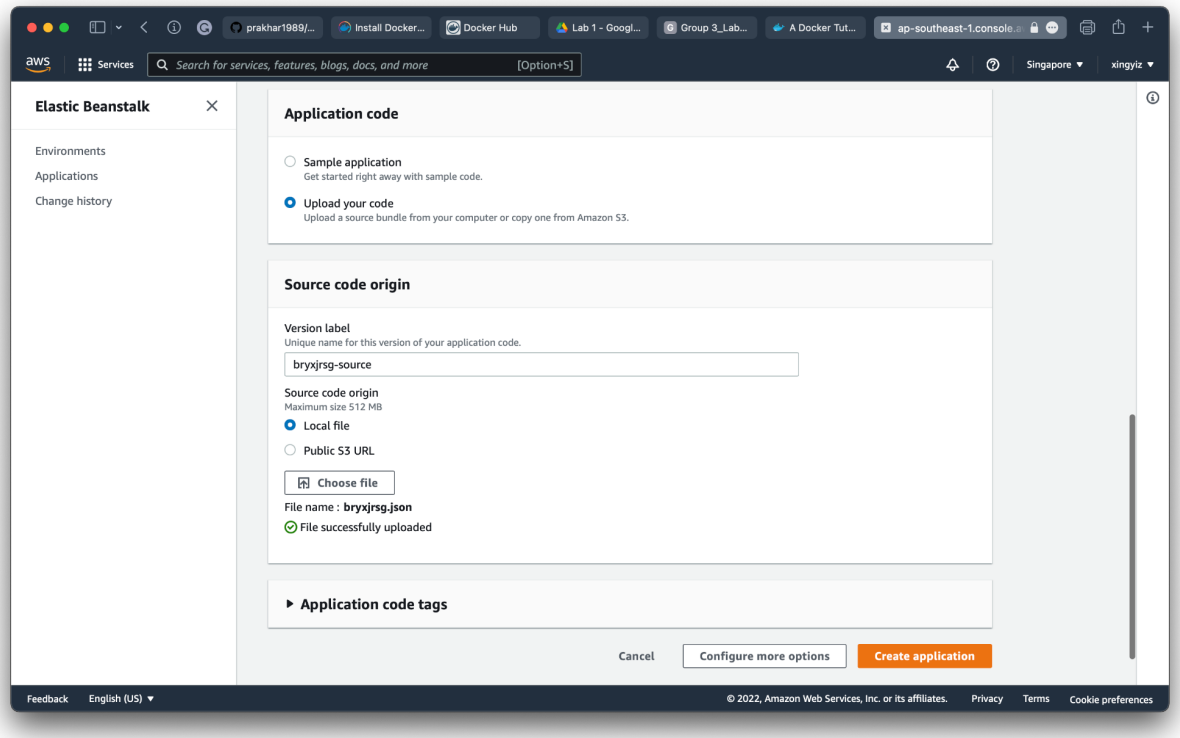


✅ Compile Docker image from Dockerfile and upload to my repository

```
root@ip-172-31-21-83:~# docker images
REPOSITORY                                    TAG       IMAGE ID       CREATED          SIZE
bryxjrsg                                      latest    315898121af6   45 minutes ago   921MB
xingyiz/bryxjrsg                              latest    315898121af6   45 minutes ago   921MB
python                                        3.9       4819be0df942   7 hours ago      912MB
busybox                                       latest    ec3f0931a6e6   2 weeks ago      1.24MB
hello-world                                   latest    feb5d9fea6a5   5 months ago     13.3kB
prakhar1989/foodtrucks-web                    latest    38fd22ce00b5   21 months ago    614MB
docker.elastic.co/elasticsearch/elasticsearch 6.3.2     96dd1575de0f   3 years ago      826MB
prakhar1989/static-site                       latest    f01030e1dcf3   6 years ago      134MB
root@ip-172-31-21-83:~#
```



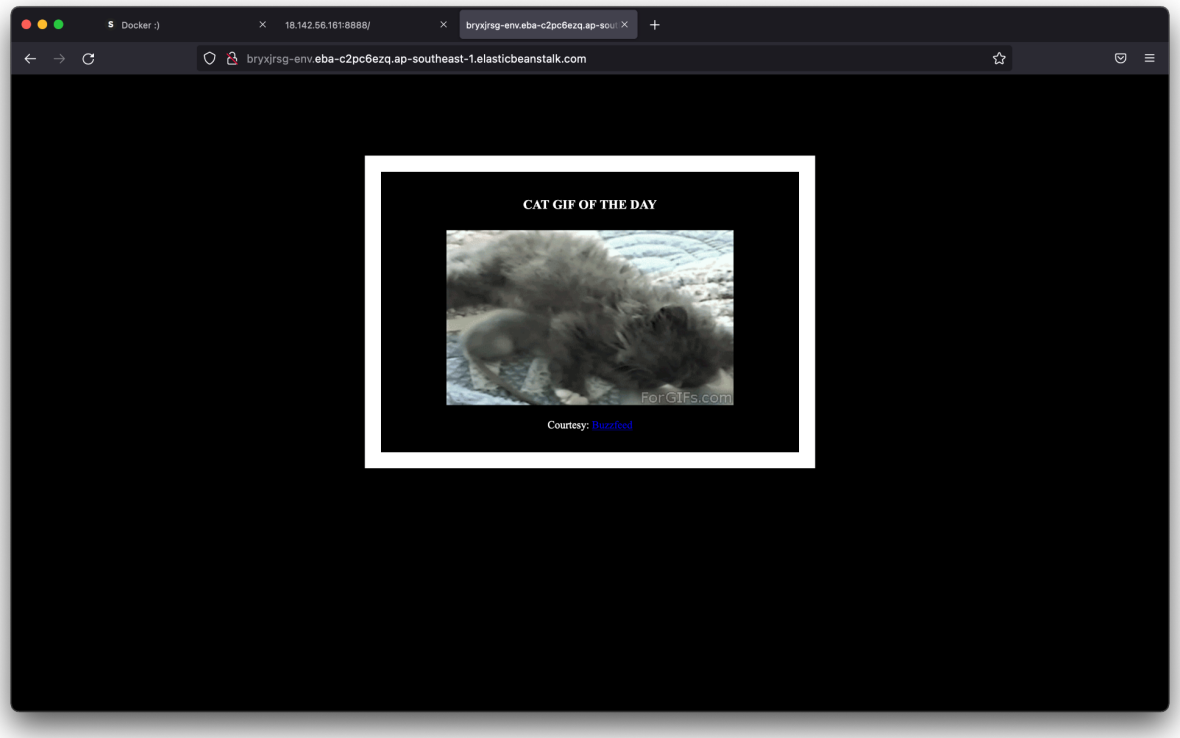
✅ Create Elastic Beanstalk website with our json pointing to our uploaded repo.

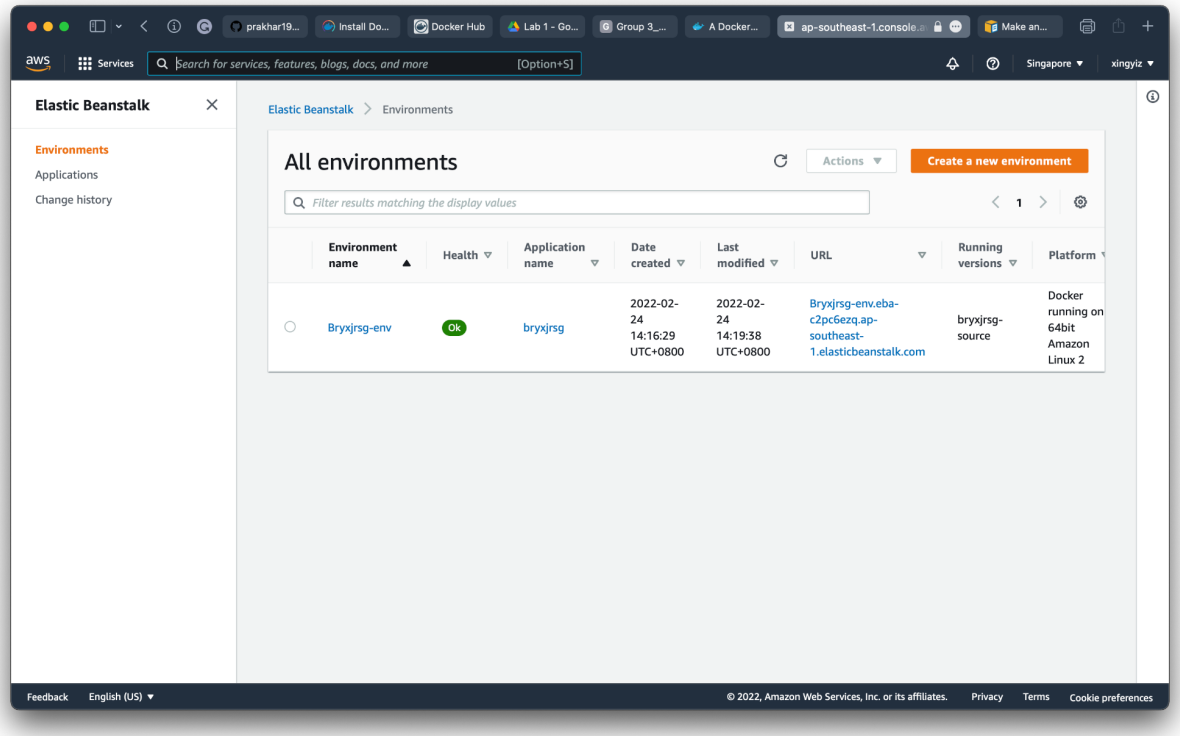


bryxjrsg.json

```
{
    "AWSEBDockerrunVersion": "1",
    "Image": {
      "Name": "xingyiz/bryxjrsg",
      "Update": "true"
    },
    "Ports": [
      {
        "ContainerPort": 5000,
        "HostPort": 8000
      }
    ],
    "Logging": "/var/log/nginx"
}
```

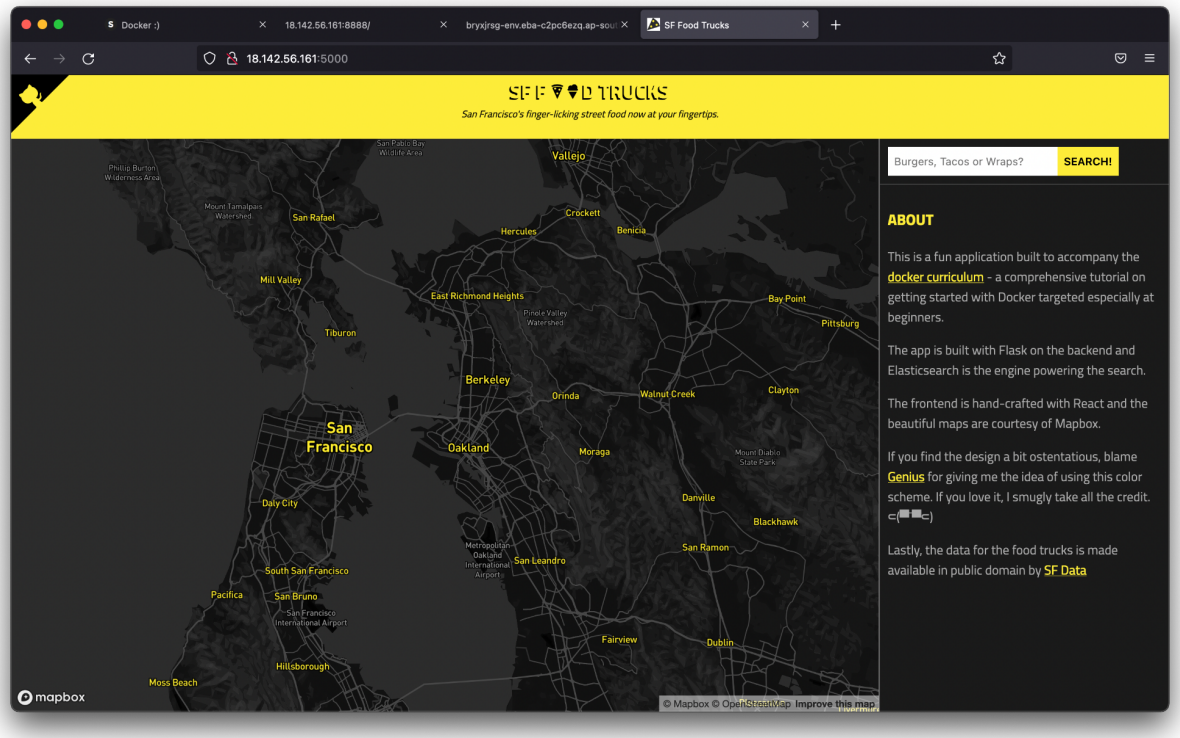
✅ Elastic Beanstalk loads Docker image and we can access the URL provided




Elastic Beanstalk
Environments
Applications
Change history
Elastic Beanstalk > Environments
All environments
Actions
Create a new environment
Filter results matching the display values
Environment name
Health
Application name
Date created
Last modified
URL
Running versions
Platform
Bryxjrsg-env
Ok
bryxjrsg
2022-02-24 14:16:29 UTC+0800
2022-02-24 14:19:38 UTC+0800
Bryxjrsg-env.eba-c2pc6ezq.ap-southeast-1.elasticbeanstalk.com
bryxjrsg-source
Docker running on 64bit Amazon Linux 2
Feedback
English (US)
© 2022, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

Task 4: Multi-Container Environments

✅ Able to run the FoodTrucks example



4.1 Insufficient Memory during Setup

The ElasticSearch container has a massive heap size of > 1GB and thus it could not be started because there was insufficient memory allocated.

For our case, we added a swap section by following this article to add a swap space of 4GB: How To Add Swap Space on Ubuntu 20.04 | DigitalOcean.

If the ElasticSearch container is killed with error code 137 (out of memory), additional RAM/swap space can be added by attaching additional Amazon EBS volumes to the EC2 instance and mounting it accordingly.

Explain how a microservice architecture is implemented using the example of Docker Network and Docker Compose.

In task 4 of this lab, we used a *docker-compose.yml* file to define two microservices, *es* and *web*. Docker Compose reads from this *docker-compose.yml* to spawn 2 containers, each acting as one microservice. This allows **isolation between different microservices** and allows us to **spin up multiple microservices using one file** using the '`docker-compose up`' command, instead of having to manually set up each service one by one. This makes the whole application **more portable and maintainable**.

Furthermore, as different services have different resource needs and grow at different rates, another advantage of Docker Compose is that separating each service into individual containers allows for **easier scalability and management** of different container resource needs.

By default, all containers are connected to the default *'bridge'* network, which allows them to communicate with each other. Docker Network also allows us to **create user-defined networks**, which allows **isolation of the network to particular containers**, since the '*bridge'* network is shared by every container by default.

When we use the default 'bridge' network, we were not able to connect the es service to the web service. This is because, inside the Flask web app's code, there is this line: `es = Elasticsearch(host='es')`. To do this properly, the web app would need to resolve and access each other by name or alias. However, containers on the default *'bridge'* network can only access each other by IP addresses because, by default, Docker runs all the containers on `` `0.0.0.0` ``. On the contrary, user-defined networks provide automatic DNS resolution between containers. Hence, this allows the web app container to connect to the Elasticsearch container via the `` `es` `` alias. An advantage of this would be to **provide network isolation between the containers**, which would **enhance security**.