# CLOUD COMPUTING
# CSC-8110
# COURSE-WORK REPORT

NAME: KAILASH BALACHANDIRAN
STUDENT ID: 220243160

**AIM:** To understand the fundamentals of programming and deploying Docker-based (an emerging cloud virtualisation technology) application hosting environment and programming and deploying cloud infrastructures using Terraform. By completing the coursework, you will be able to gain hands-on experience on the following inter-related aspects:

- Configuring a Docker-based application hosting environment;
- Pushing and pulling images from the Docker Hub (global repository of software components' images maintained by the developers);
- Creating and deploying a complex web application stack consisting of multiple software components (e.g., database, web server, etc.);
- Monitoring the ongoing performance of your application stack using the inherent features provided by the Docker environment.
- Deploying a Kubernetes service on Azure Kubernetes Service and running a sample microservice on the cluster

**Task 1:** Deploy a web application component in Docker Environment

**Task Objective:** To understand and learn how to pull and run Docker images from Docker Hub using the command line interface. sample Java web application component image, named "nclcloudcomputing/java benchmark app", has been uploaded to the Docker Hub. The image contains a ready-to-use implementation of a web application deployed in a Tomcat server (an open-source web server). In terms of computational logic, the web application implements a prime number check on a large number. By doing so, the application can generate a high CPU and memory load.

1. Pull and run the Docker image "nclcloudcomputing/javabenchmarkapp" from Docker Hub in the virtual machine running on your laptop/PC. Perform these tasks using the command line interface (CLI).

The commands in *Figure 1.1* are executed in the terminal using

docker CLI

The following command is used to pull the java web application image

*"docker pull nclcloudcomputing/javabenchmarkapp"*

The following command is used to view the pulled images in the docker

*"docker images"*

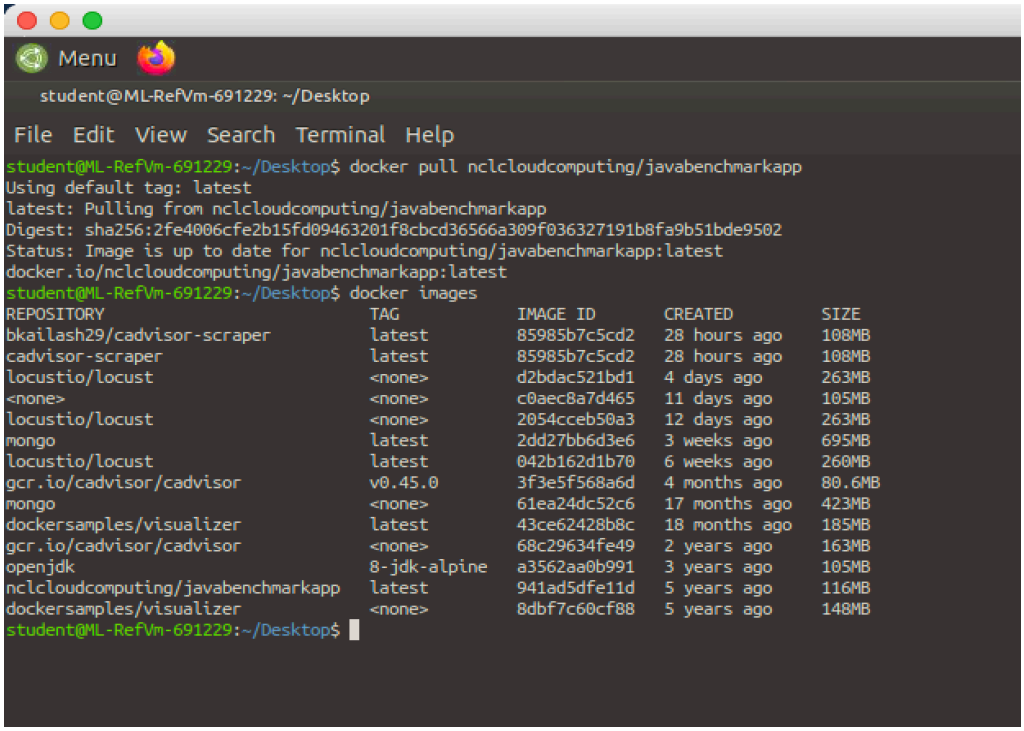


Figure 1.1

In *Figure 1.2* we use docker images to view the images and "*docker container ls"* to view the running containers

The following command is used to run the java web application image on port 8083 in detached mode

*"docker run -dp 8083:8080 nclcloudcomputing/javabenchmarkapp"*

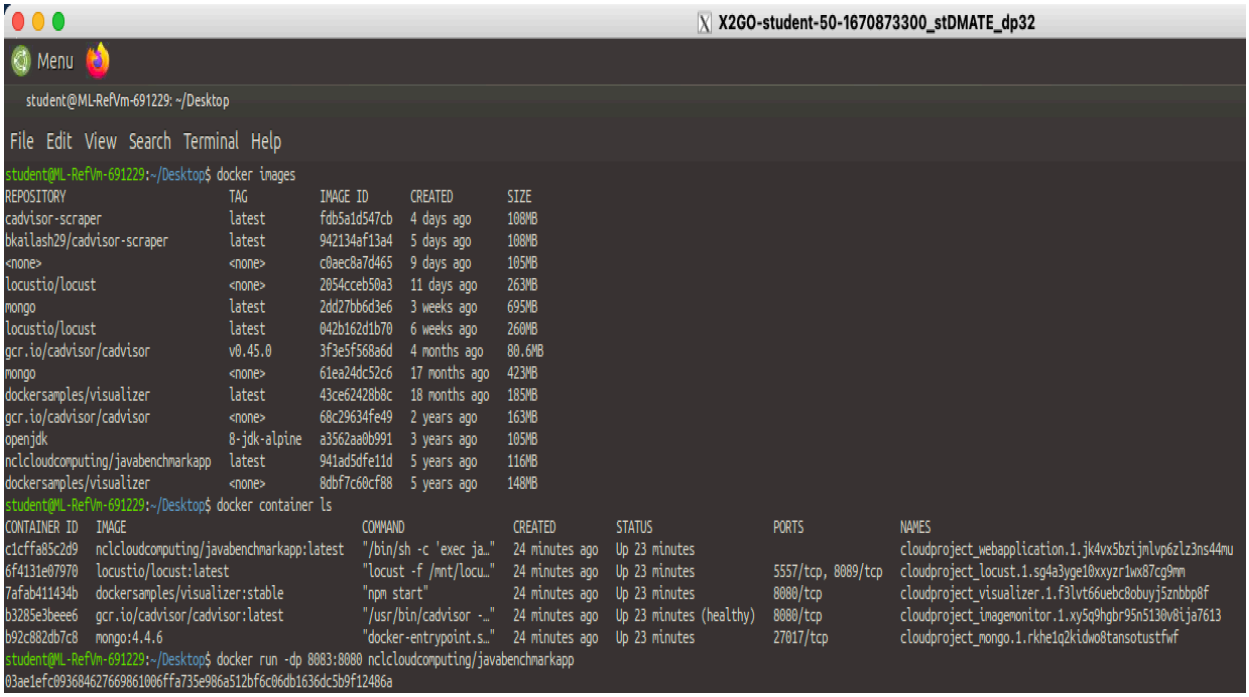


Figure 1.2

2. Verify whether the web application is working by pointing your web browser to [http: //localhost:8083/primecheck]. The page should display the time taken to perform a prime number checking calculation.

   *Figure 1.3* shows that the java web application image running on port 8083 and displaying the time taken to perform the prime number check calculation

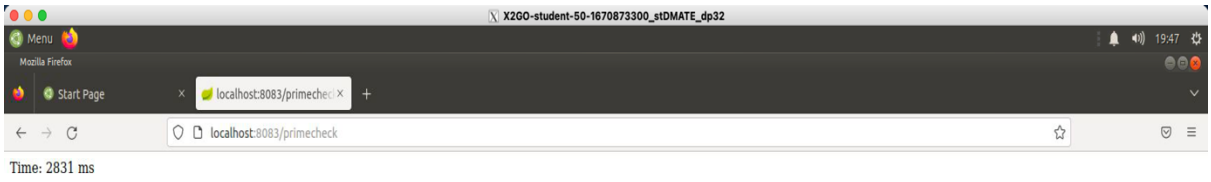


Figure 1.3

**Task 2:** Deploy a complex web application stack in Docker Environment

**Task Objective:** To understand how to create a complex web application stack consisting of multiple Docker images delivering distinct services including a web server, load generation, database server, monitoring engine, and visualisation. This task will also help you in understanding how native Docker Compose and Swarm techniques can be leveraged to manage and deploy such complex web application stacks.

1. Pull the following images from the Docker Hub:

   a) load generator Docker image (locustio/locust);

   *Figure 2.1* shows that the docker image "*locustio/locust"* is pulled to the docker CLI from the docker hub using the command

   *"docker pull locustio/locust"*

   This is used to run load tests.

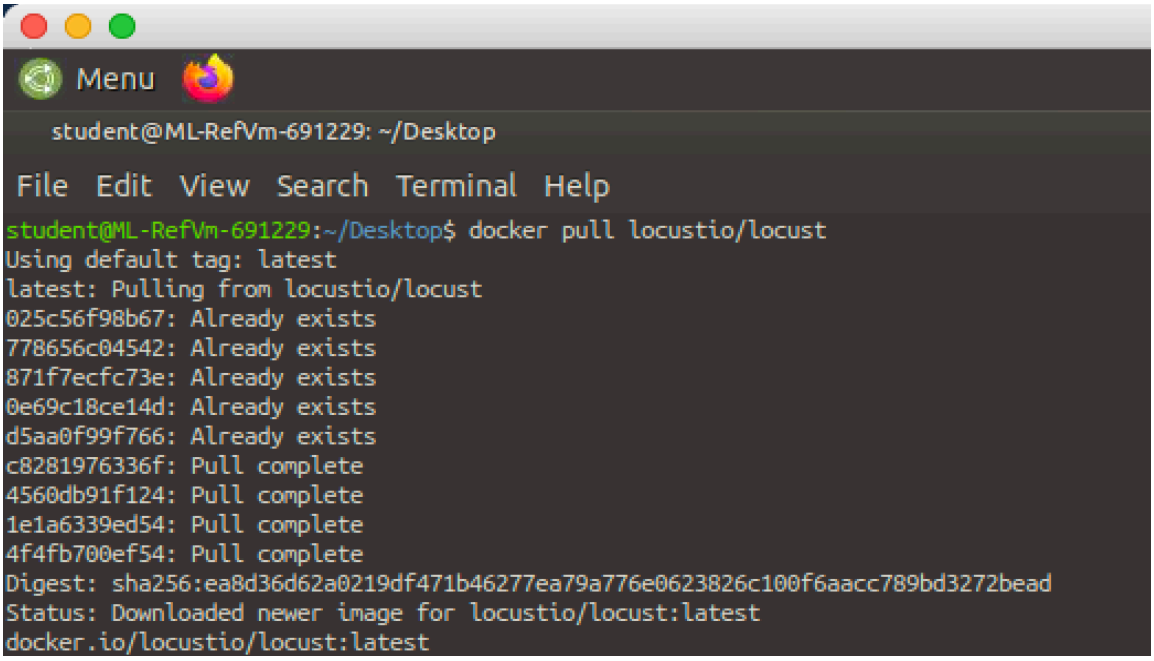


Figure 2.1

   b) MongoDB Docker image (mongo);

   *Figure 2.2* shows that the docker image *"mongo"* is pulled from the docker Hub using the command

   *"docker pull mongo"*

Mongo is a NoSQL database used to store data.


Figure 2.2

c) cAdvisor monitoring Docker image (gcr.io/cadvisor/cadvisor:v0.45.0);

*Figure 2.3* shows that the docker image *"gcr.io/cadvisor/cadvisor:v0.45.0"* is pulled from the docker Hub using the command

*"docker pull gcr.io/cadvisor/cadvisor:v0.45.0"*

cAdvisor is used to monitor containers. It provides monitoring data of CPU, memory, network, etc. and visualizes the data in graph form.

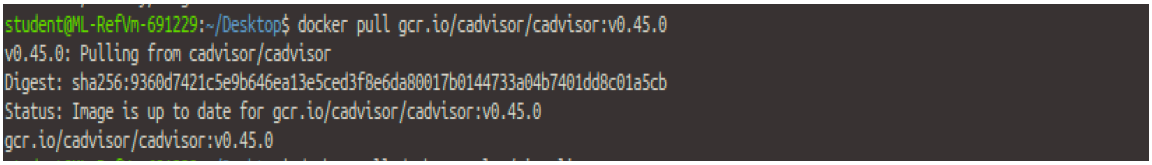


Figure 2.3

d) Docker Swarm Visualizer (dockersamples/visualizer);

*Figure 2.4* shows that the docker image *"dockersamples/visualizer"* is pulled from the docker Hub using the command

*"docker pull dockersamples/visualizer"*

A visualizer is used to ensure that all containers are up and running.

```
student@ML-RefVm-691229:~/Desktop$ docker pull dockersamples/visualizer
Using default tag: latest
latest: Pulling from dockersamples/visualizer
Digest: sha256:530c863672e7830d7560483df66beb4cbbcd375a9f3ec174ff5376616686a619
Status: Image is up to date for dockersamples/visualizer:latest
docker.io/dockersamples/visualizer:latest
```

Figure 2.4

Figure 2.5 shows that all the above-mentioned docker images are pulled to the docker CLI, it can be viewed with the command *"docker images"*

```
X2GO-student-51-16709
Menu
student@ML-RefVm-691229: ~/Desktop
File Edit View Search Terminal Help
student@ML-RefVm-691229:~/Desktop$ docker pull locustio/locust
Using default tag: latest
latest: Pulling from locustio/locust
025c56f98b67: Already exists
778656c04542: Already exists
871f7ecfc73e: Already exists
0e69c18ce14d: Already exists
d5aa0f99f766: Already exists
c8281976336f: Pull complete
4560db91f124: Pull complete
1e1a6339ed54: Pull complete
4f4fb700ef54: Pull complete
Digest: sha256:ea8d36d62a0219df471b46277ea79a776e0623826c100f6aacc789bd3272bead
Status: Downloaded newer image for locustio/locust:latest
docker.io/locustio/locust:latest
student@ML-RefVm-691229:~/Desktop$ docker pull mongo
Using default tag: latest
latest: Pulling from library/mongo
846c0b181fff: Pull complete
ef773e84b43a: Pull complete
2bfad1efb664: Pull complete
84e59a6d63c9: Pull complete
d2f00ac700e0: Pull complete
96d33bf42f45: Pull complete
ebaa69d77b61: Pull complete
aa77b709a7d6: Pull complete
245bd0c9ace2: Pull complete
Digest: sha256:f1b5a4e2acc7db563457f41443103a2d48d1ee5a13332734f82222aa719e2542
Status: Downloaded newer image for mongo:latest
docker.io/library/mongo:latest
student@ML-RefVm-691229:~/Desktop$ docker pull gcr.io/cadvisor/cadvisor:v0.45.0
v0.45.0: Pulling from cadvisor/cadvisor
Digest: sha256:9360d7421c5e9b646ea13e5ced3f8e6da80017b0144733a04b7401dd8c01a5cb
Status: Image is up to date for gcr.io/cadvisor/cadvisor:v0.45.0
gcr.io/cadvisor/cadvisor:v0.45.0
student@ML-RefVm-691229:~/Desktop$ docker pull dockersamples/visualizer
Using default tag: latest
latest: Pulling from dockersamples/visualizer
Digest: sha256:530c863672e7830d7560483df66beb4cbbcd375a9f3ec174ff5376616686a619
Status: Image is up to date for dockersamples/visualizer:latest
docker.io/dockersamples/visualizer:latest
student@ML-RefVm-691229:~/Desktop$ docker images
REPOSITORY                          TAG           IMAGE ID       CREATED          SIZE
locustio/locust                     latest        104d48691d70   13 hours ago     263MB
bkailash29/cadvisor-scraper         latest        85985b7c5cd2   28 hours ago     108MB
cadvisor-scraper                    latest        85985b7c5cd2   28 hours ago     108MB
locustio/locust                     <none>        d2bdac521bd1   4 days ago       263MB
mongo                               latest        0850fead9327   4 days ago       700MB
<none>                              <none>        c0aec8a7d465   11 days ago      105MB
locustio/locust                     <none>        2054cceb50a3   12 days ago      263MB
mongo                               <none>        2dd27bb6d3e6   3 weeks ago      695MB
locustio/locust                     <none>        042b162d1b70   6 weeks ago      260MB
gcr.io/cadvisor/cadvisor            v0.45.0       3f3e5f568a6d   4 months ago     80.6MB
mongo                               <none>        61ea24dc52c6   17 months ago    423MB
dockersamples/visualizer            latest        43ce62428b8c   18 months ago    185MB
gcr.io/cadvisor/cadvisor            <none>        68c29634fe49   2 years ago      163MB
openjdk                             8-jdk-alpine  a3562aa0b991   3 years ago      105MB
nclcloudcomputing/javabenchmarkapp  latest        941ad5dfe11d   5 years ago      116MB
dockersamples/visualizer            <none>        8dbf7c60cf88   5 years ago      148MB
student@ML-RefVm-691229:~/Desktop$
```

Figure 2.5

2. Define a Docker compose file which contains all necessary configurations and instructions for deploying and instantiating the above (step 1) Docker images (web server, locust, MongoDB, cAdvisor, Docker Swarm Visualizer);

A *"docker-compose.yml"* file is created that follows the web application topology shown below. Only one instance is created per image. These images are connected to each other through a bridged network. Swarm creates a default bridge network by default. A default configured network named web net is created in the YAML file. Required volumes and commands are added to ensure the service is running.

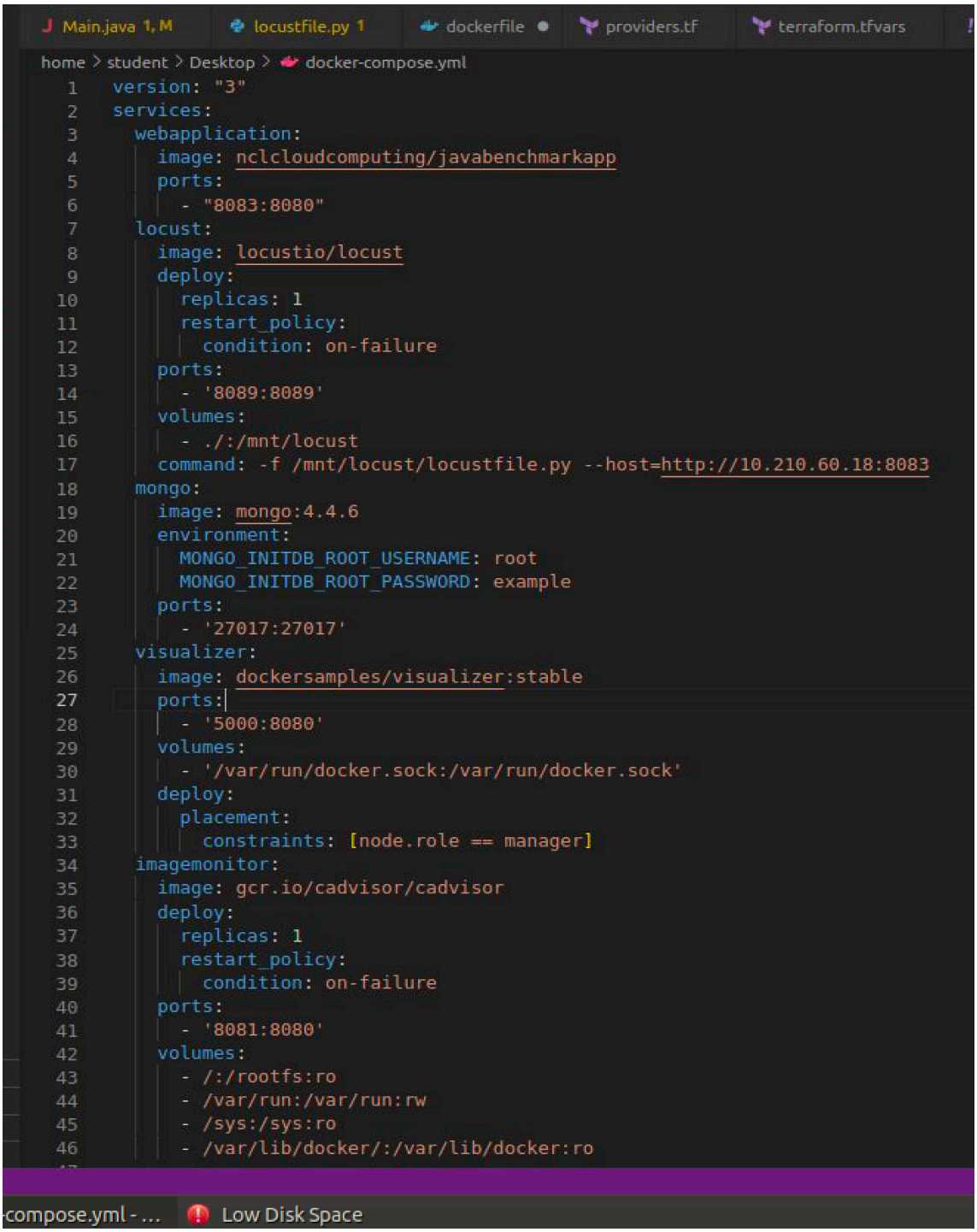


Figure 2.6

3. Create a web application topology, on a single Docker swarm node by using the above (step 2) Docker compose configurations.

   Docker Swarm is a container orchestration tool that allows users to manage deployed containers. Swarm follows a master/slave architecture, with manager nodes managing other nodes. In this task, we will run a swarm on the manager node. Initialize the swarm with the command

   *"docker swarm init"*

   Deploy a new docker stack by running the following command

   *"docker stack deploy -c docker-compose.yml"*

   The following command is used to view the running services in the swarm node

   *"docker service ls"*

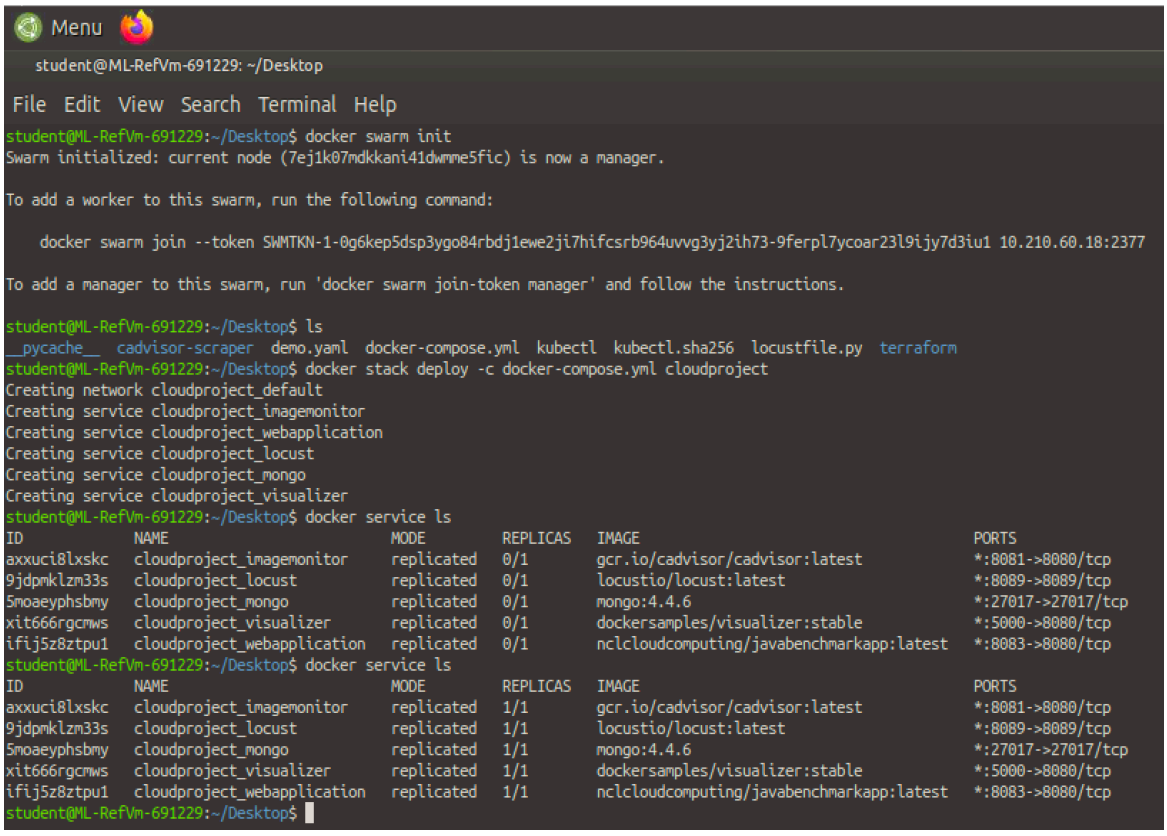


Figure 2.7

*Figure 2.8* shows that the *"mongo"* image is running on port 27017, using the web browser "*http://localhost:27017"*

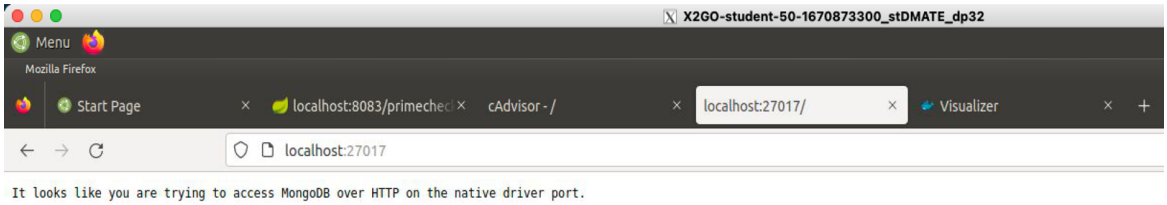


Figure 2.8

*Figure 2.9* shows that the *"gcr.io/cadvisor/cadvisor:v0.45.0"* image is running on port 8081, using the web browser "*http://localhost:8081"*

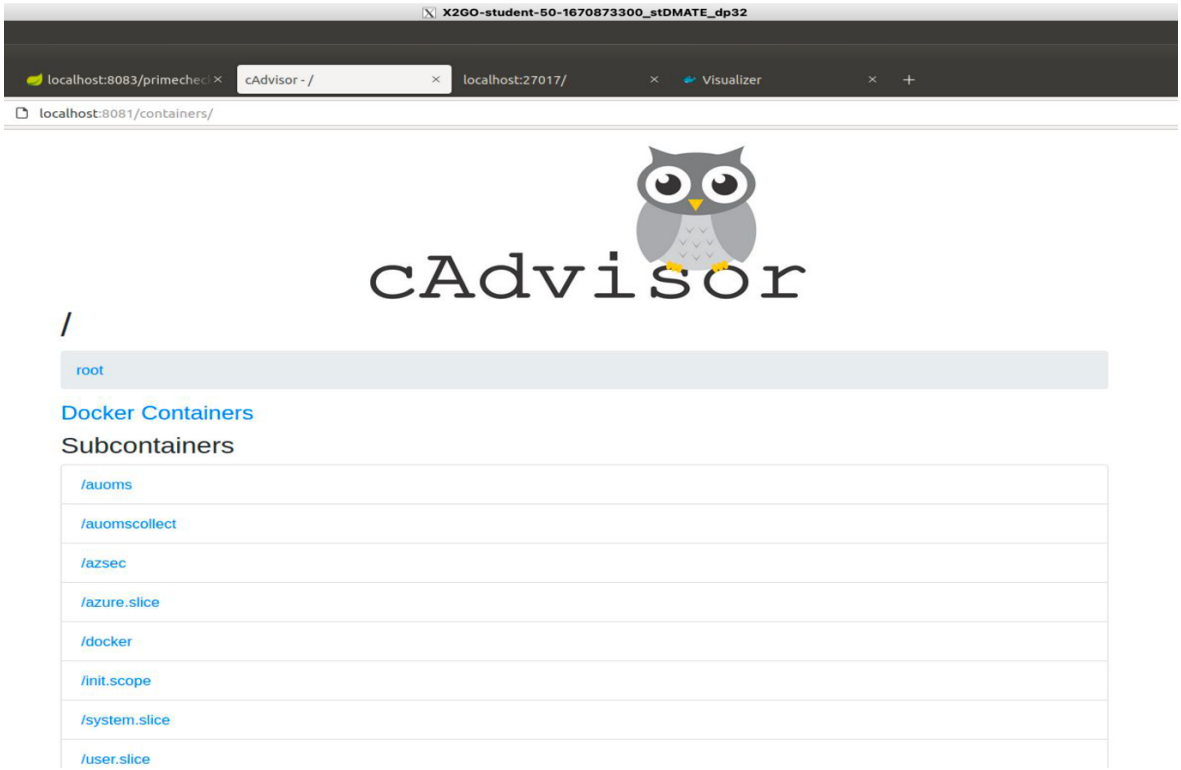


Figure 2.9

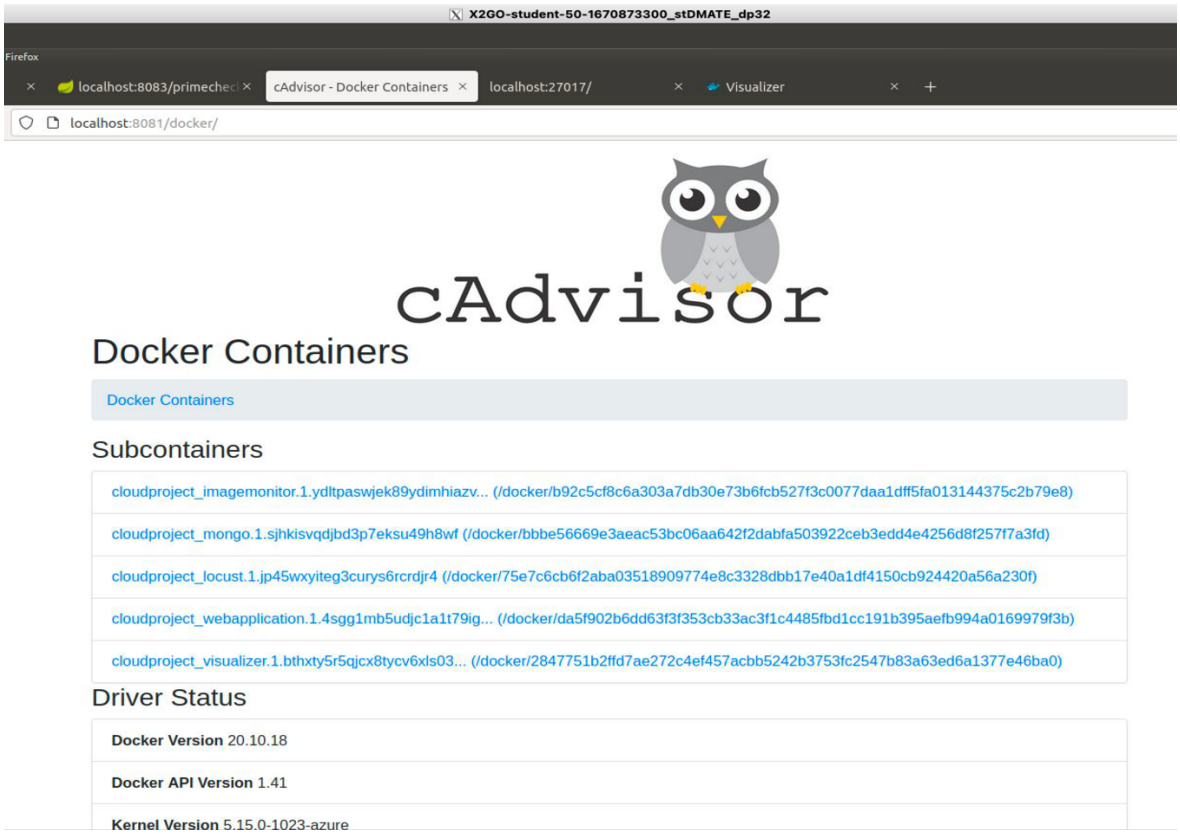


Figure 2.10

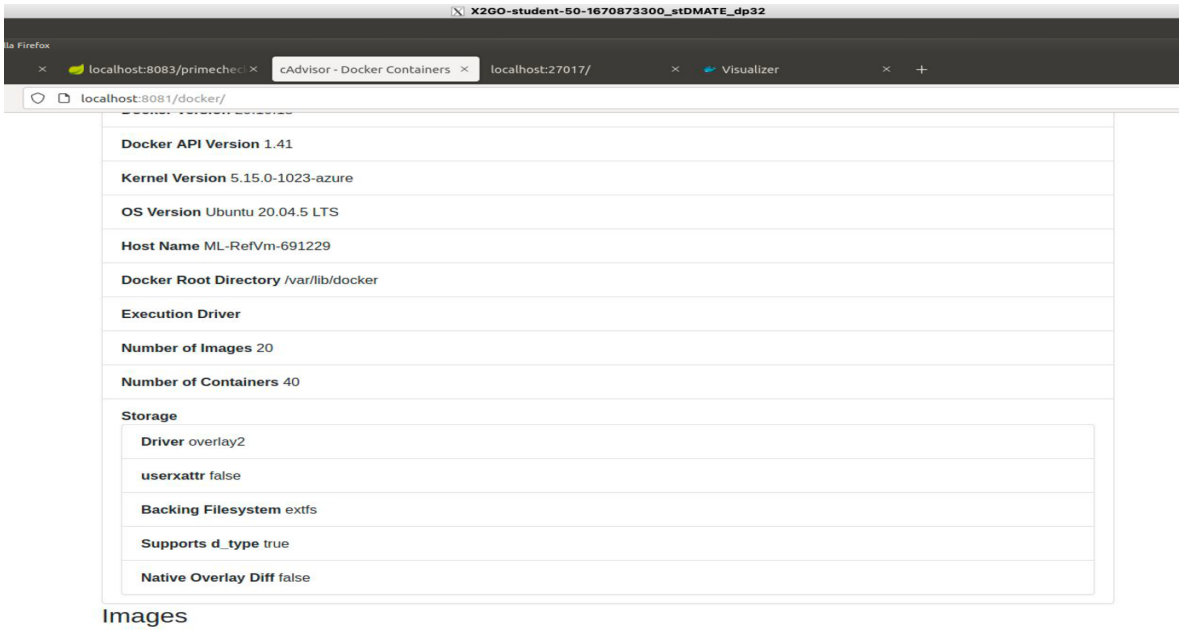


Figure 2.11

*Figure 2.12* shows that the *"dockersamples/visualizer"* image is running on port 5000, using the web browser "*http://localhost:5000"*

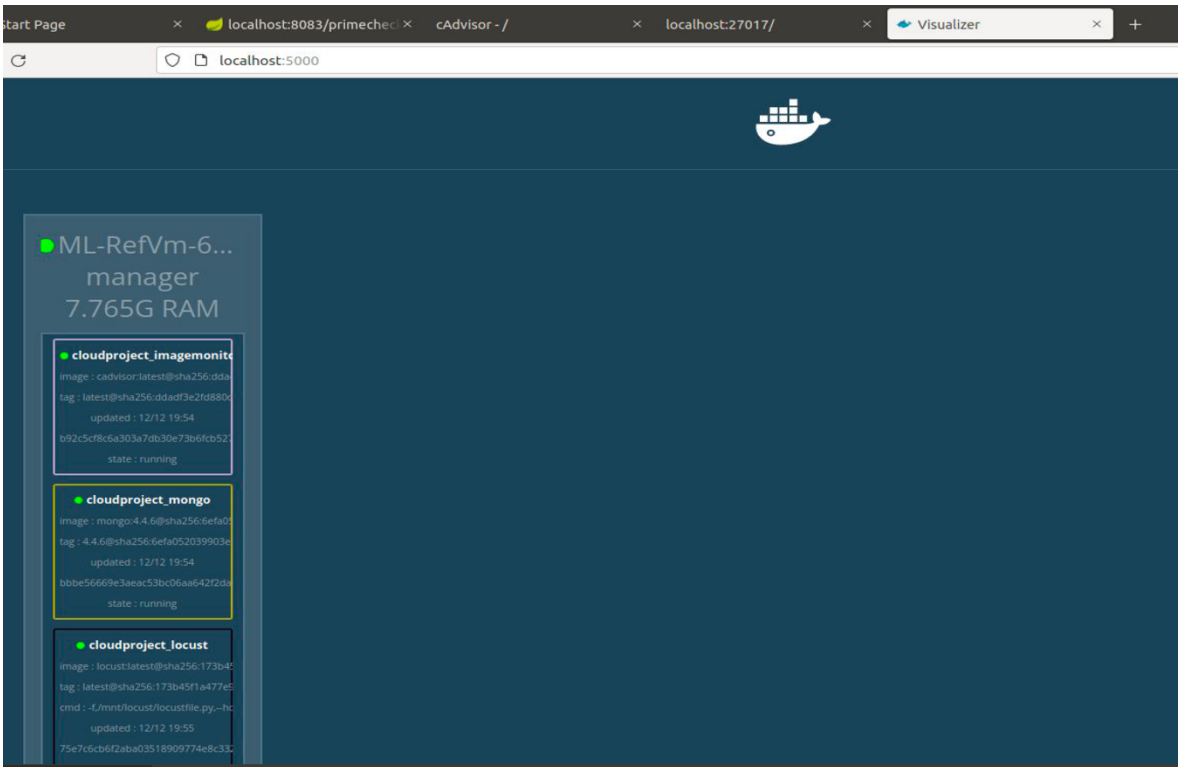


Figure 2.12

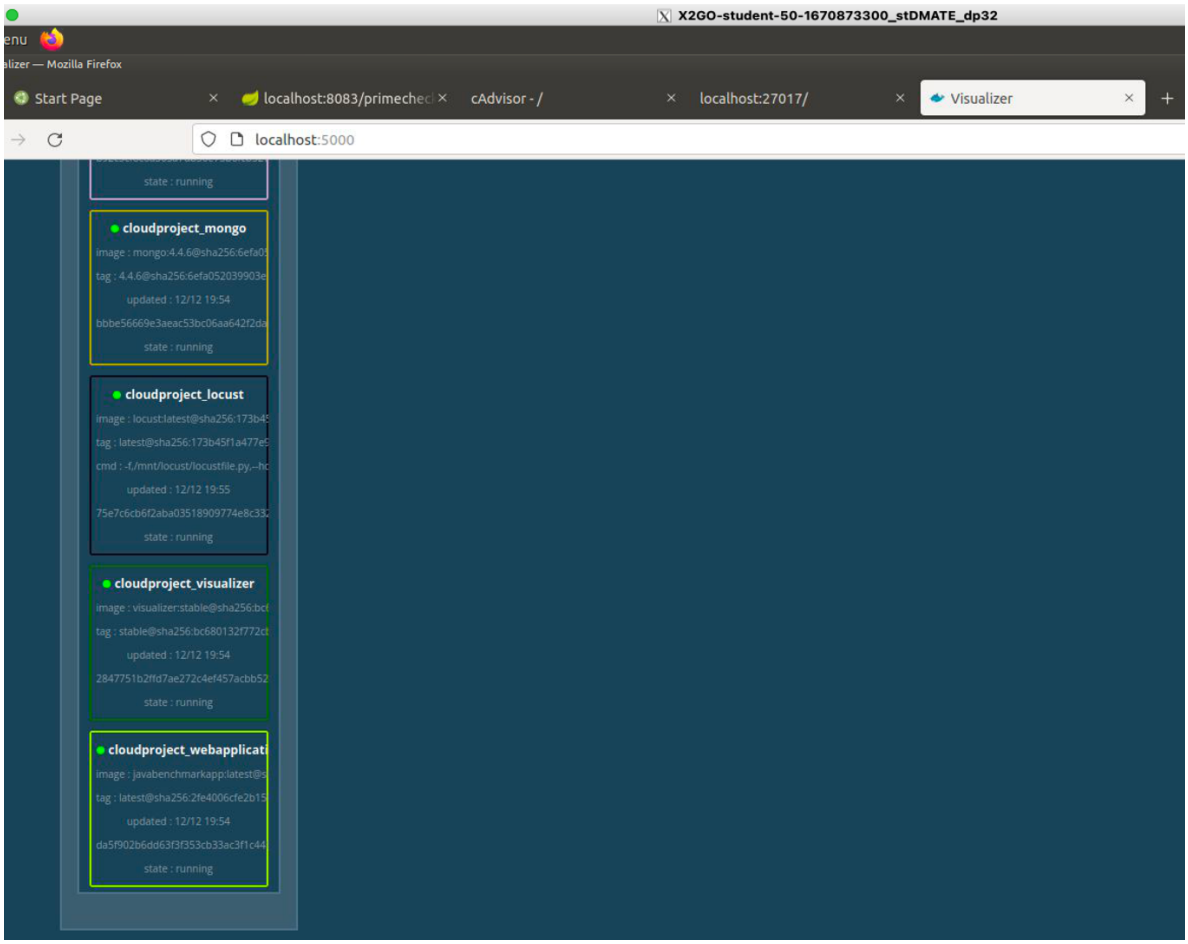


Figure 2.13

Locust: Use locust (locustio/locust) to generate load. In *Figure 2.14* We create locusfile.py to generate the load of the Benchmark app API (http://10.210.60.18:8083).

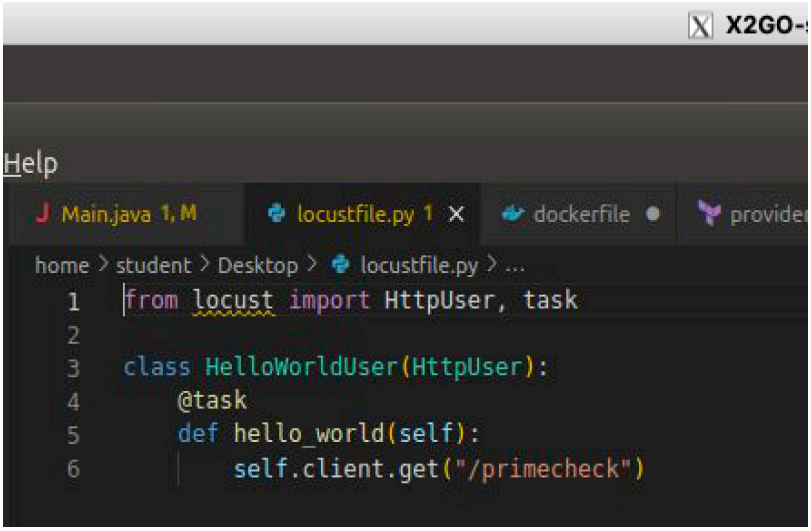


Figure 2.14

*Figure 2.15* the Locust UI, enter the number of concurrent users you want the system to spawn. It also includes the spawning rate. The host is automatically filled from the locust file.

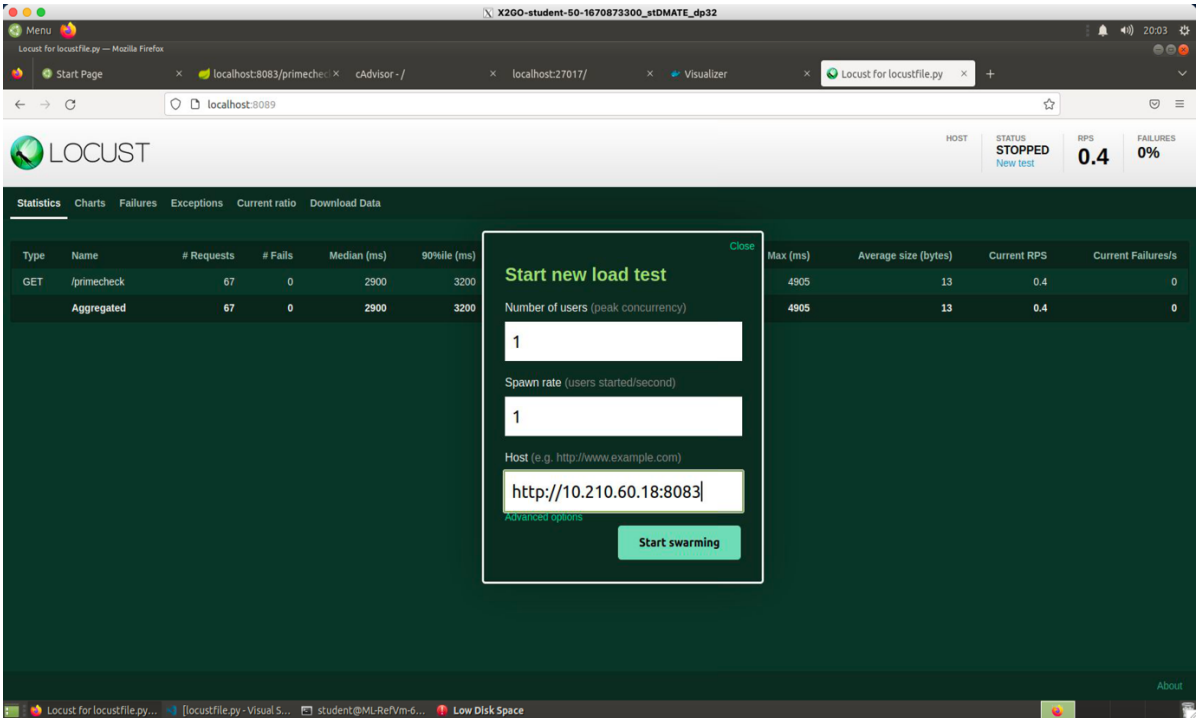


Figure 2.15

Click Start Swarming to generate load.

The screenshots below are from running the load generator for a few minutes.

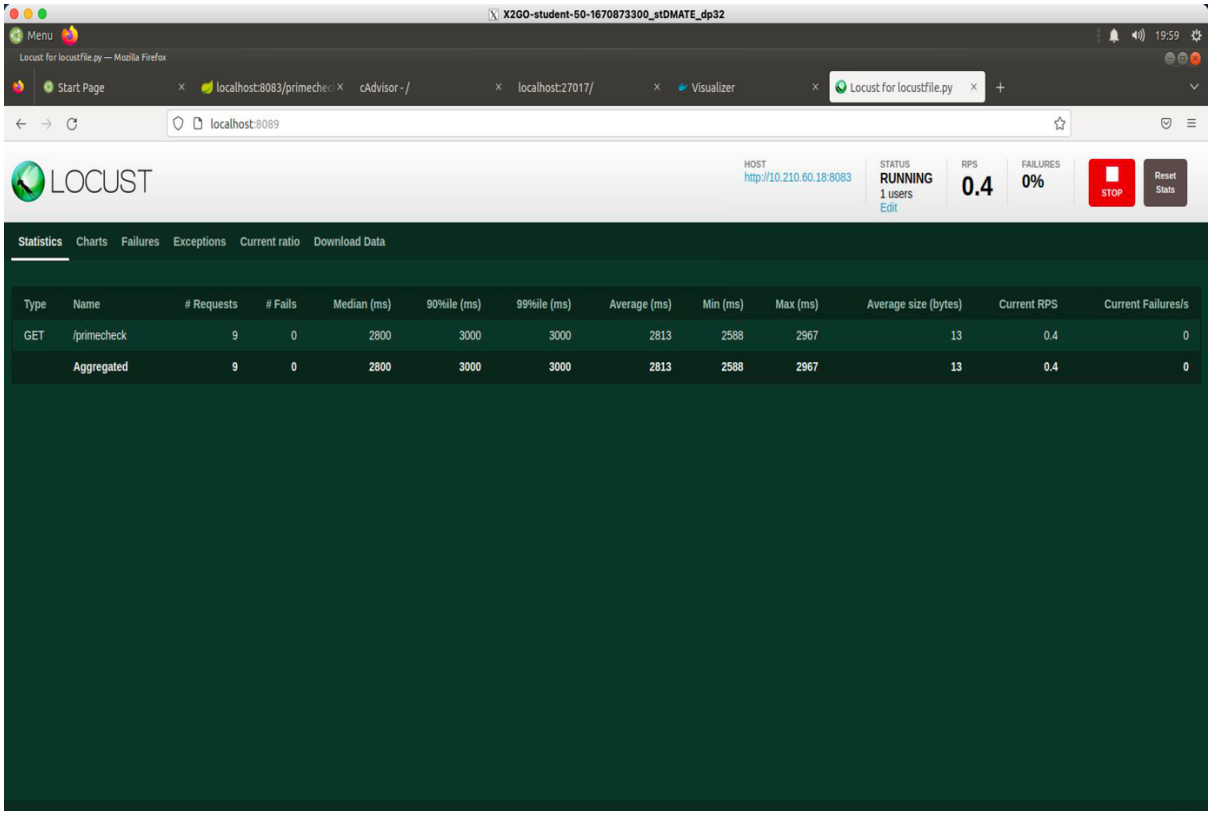


Figure 2.16

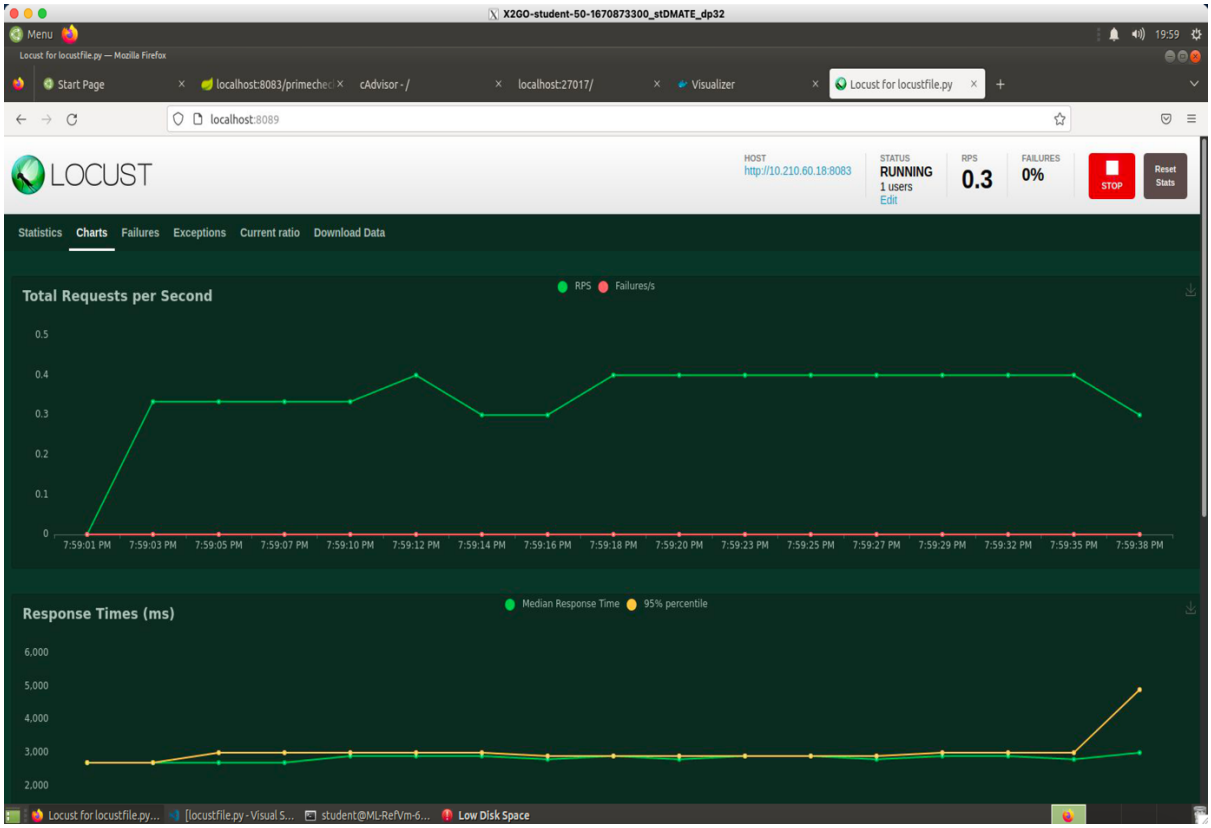


Figure 2.17

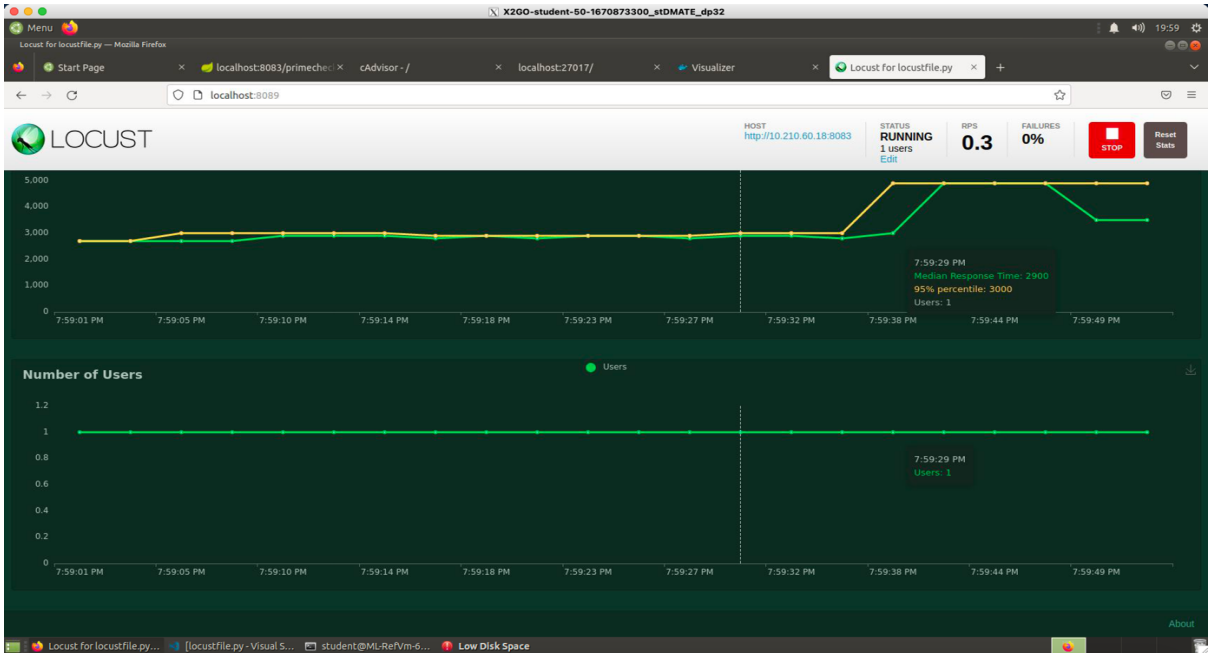


Figure 2.18

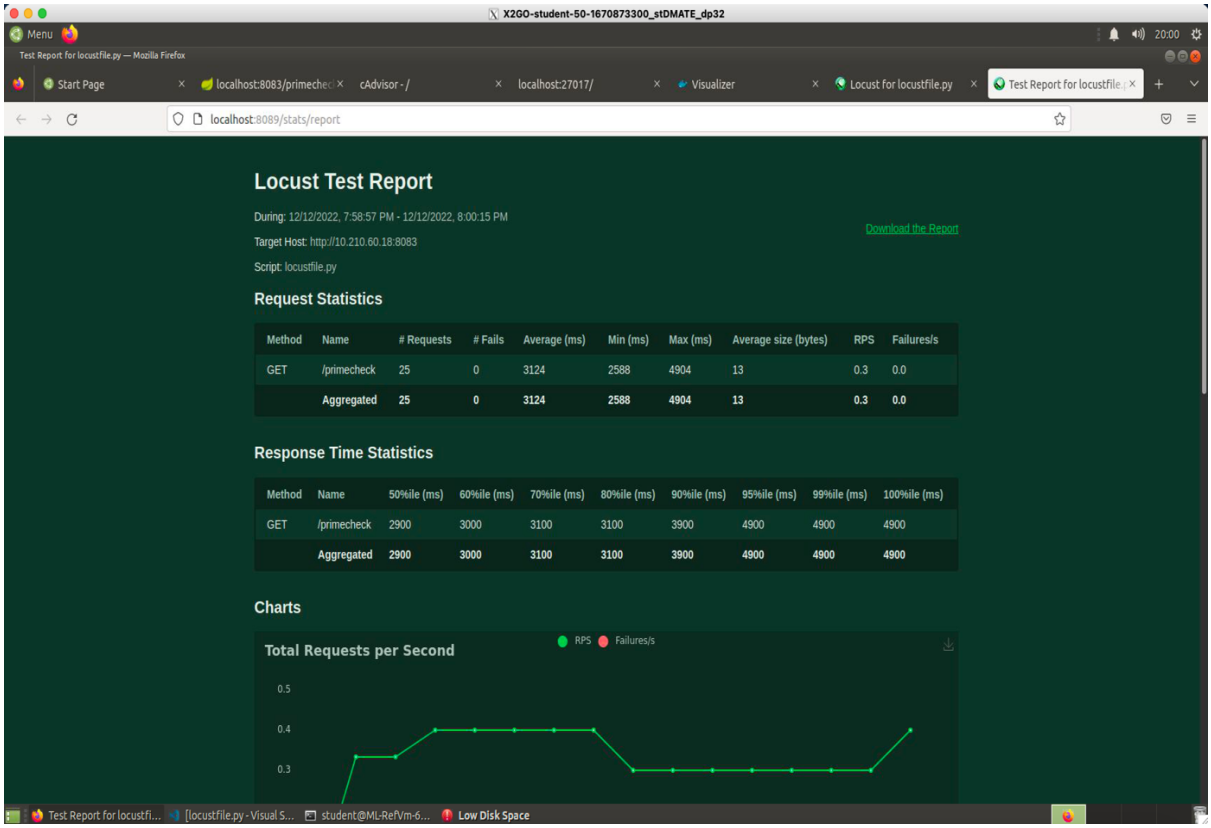


Figure 2.19

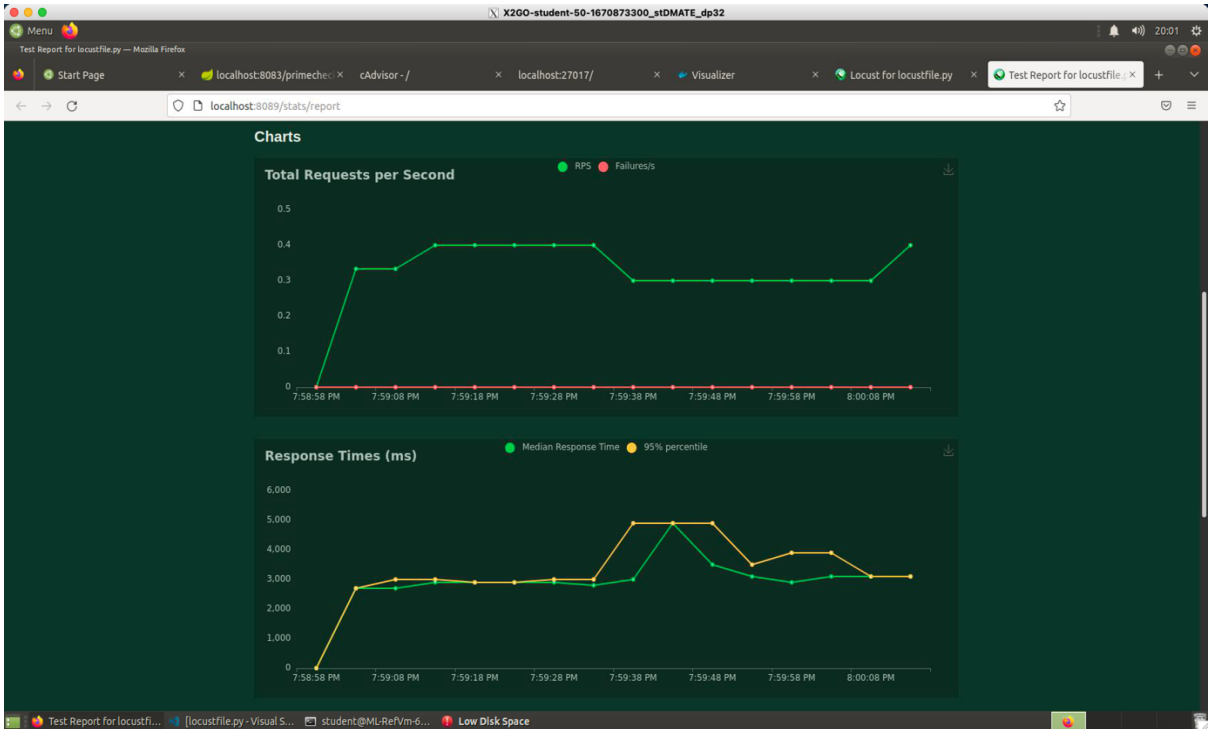


Figure 2.20

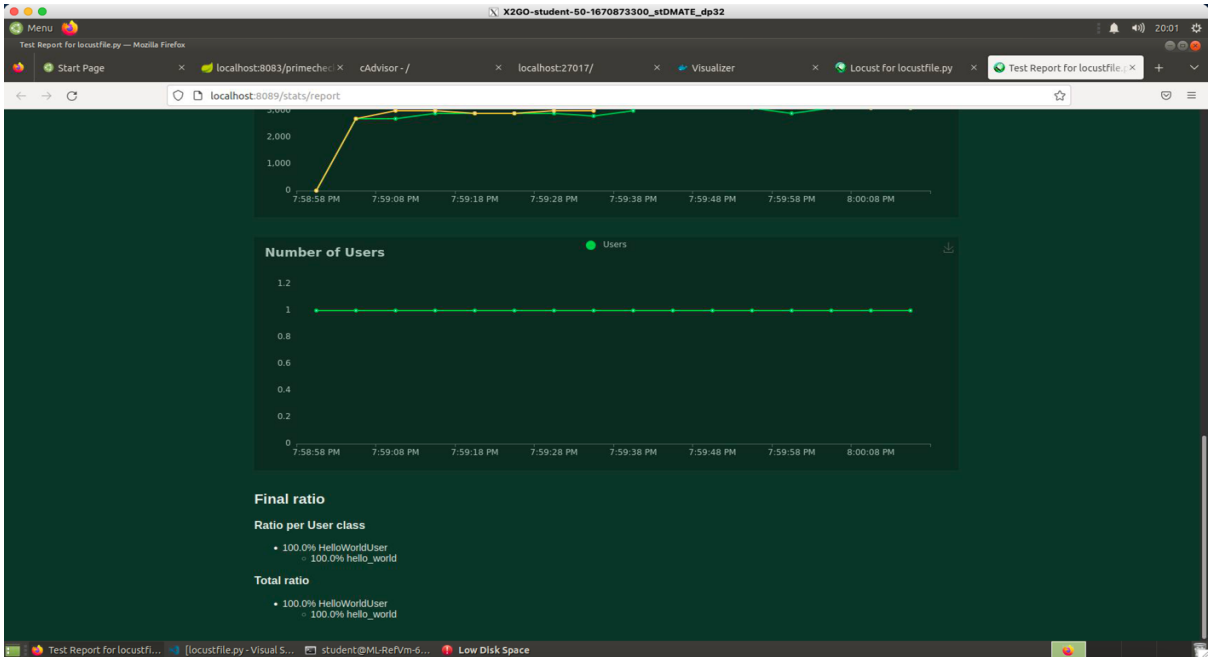


Figure 2.21

Stop docker swarm with the following command
*"docker swarm leave --force"*

**Task 3:** Build your own Docker image and push it to the Docker Hub

**Task Objective:** To understand how to build a Docker image from scratch and how to push it to Docker Hub so that other developers can view, use and extend the same.

1. You need to build a Docker image of a Java program. In terms of the application logic, the program implements features including how to extract the monitoring information from the Docker cAdvisor container (a running instance of gcr.io/cadvisor/cadvisor image) and how to insert this monitoring information into a MongoDB Docker container (a running instance of mongo image). The cAdvisor is able to monitor run-time performance (CPU load, network I/O and memory usage) of all instances of Docker images running within a Docker Swarm.

   The below figure shows the results detected by the cAdvisor user interface.

   The screenshots are after running the load generator for a few minutes


Figure 3.1

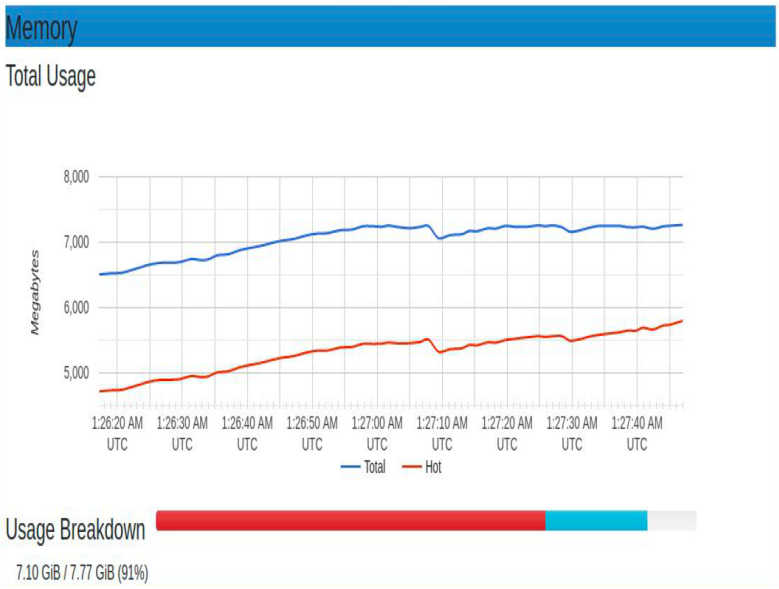


Figure 3.2

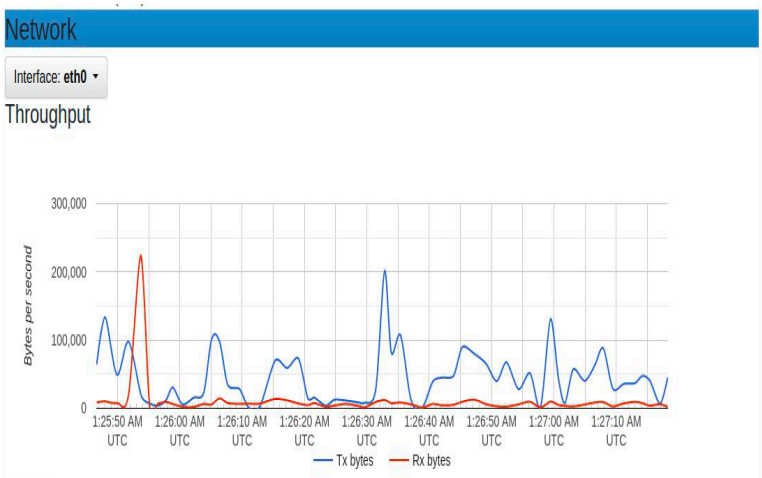


Figure 3.3

We use cAdvisor to monitor benchmark app containers where data is consumed by the cAdvisor scraper application using the APIs provided by cAdvisor.

2. Edit the provided Java code using the IntelliJ editor (already installed in the provided virtual image). The instructions in Java code that need editing include *container ID, MongoDB Address*, and *endpoint* address.

   The application sends requests to cAdvisor, which monitors web applications. cAdvisor's API is used to extract and process the data which is later transferred to mongo DB. Requests are sent every minute because cAdvisor returns metrics at the last minute.

   In the main.java file, make the following changes and build the image. Change the container field, which is the container ID of the Java web application that needs to be monitored. Also, change the MongoDB address which is the mongo DB address and the endpoint which is the cAdvisors endpoint.

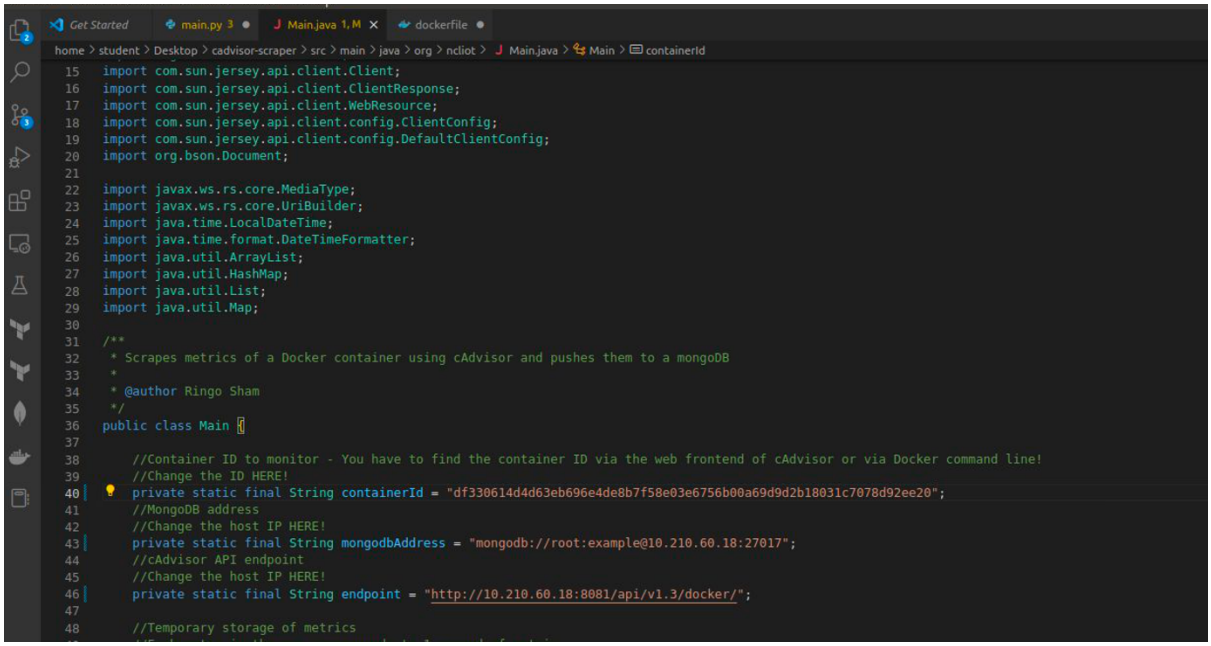


Figure 3.4

Figure 3.2 shows that the java web application address

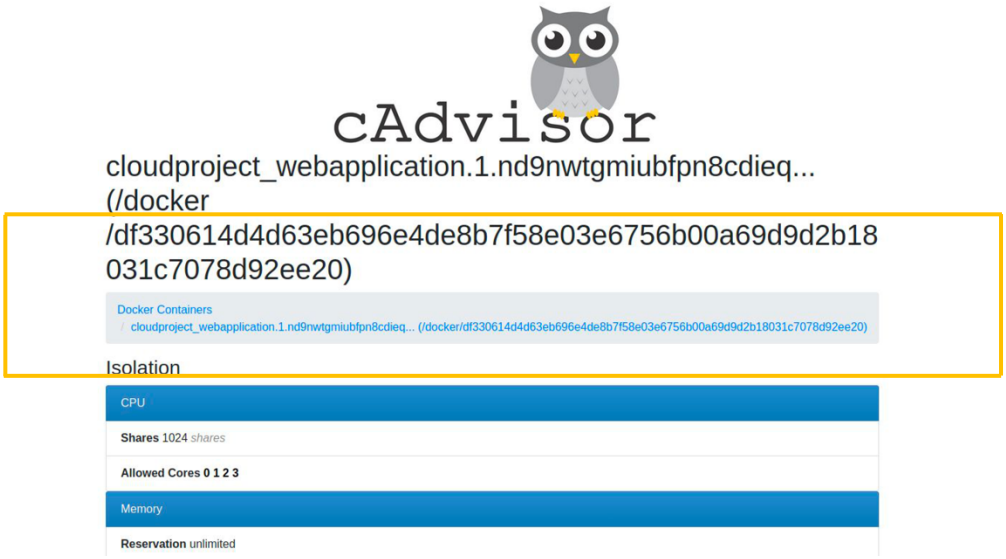


Figure 3.5

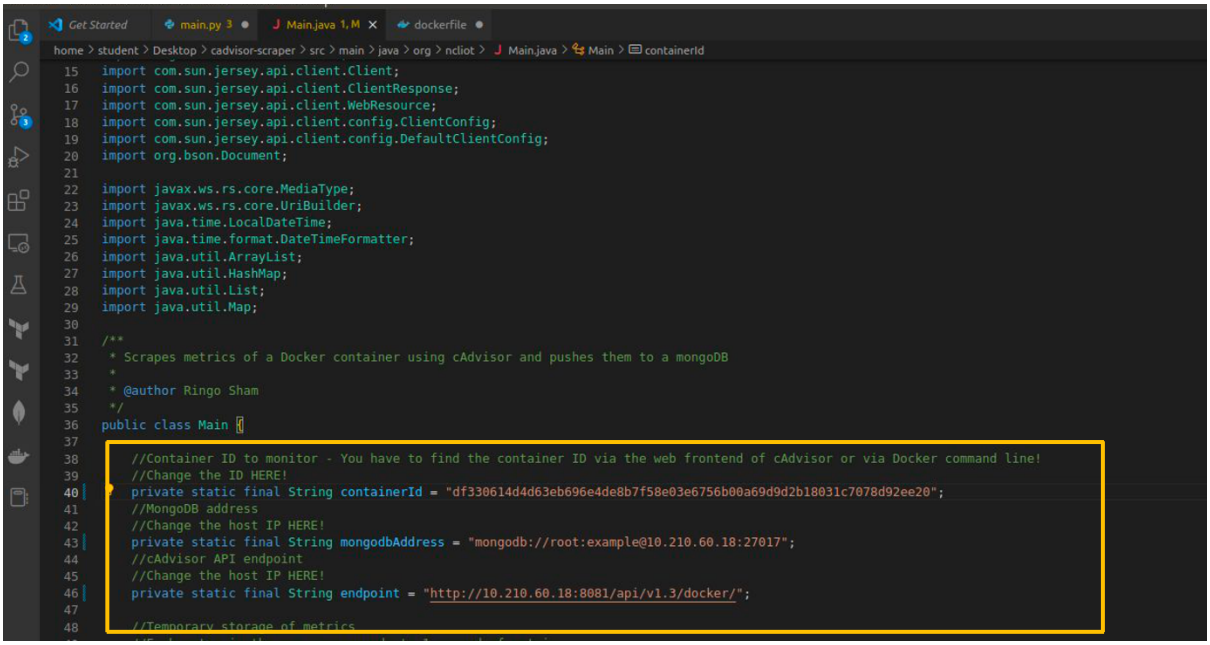


Figure 3.6

Figure 3.4 shows that the maven is built in the docker CLI java web application and sends requests to cAdvisor, which monitors web applications. with the following cmd, we have the dependencies under the fully built jar.

*"mvn clean compile assembly:single"*


Figure 3.7

Figure 3.5 shows that the docker file is created for the java application with the c-Advisor scraper dependencies mentioned on it

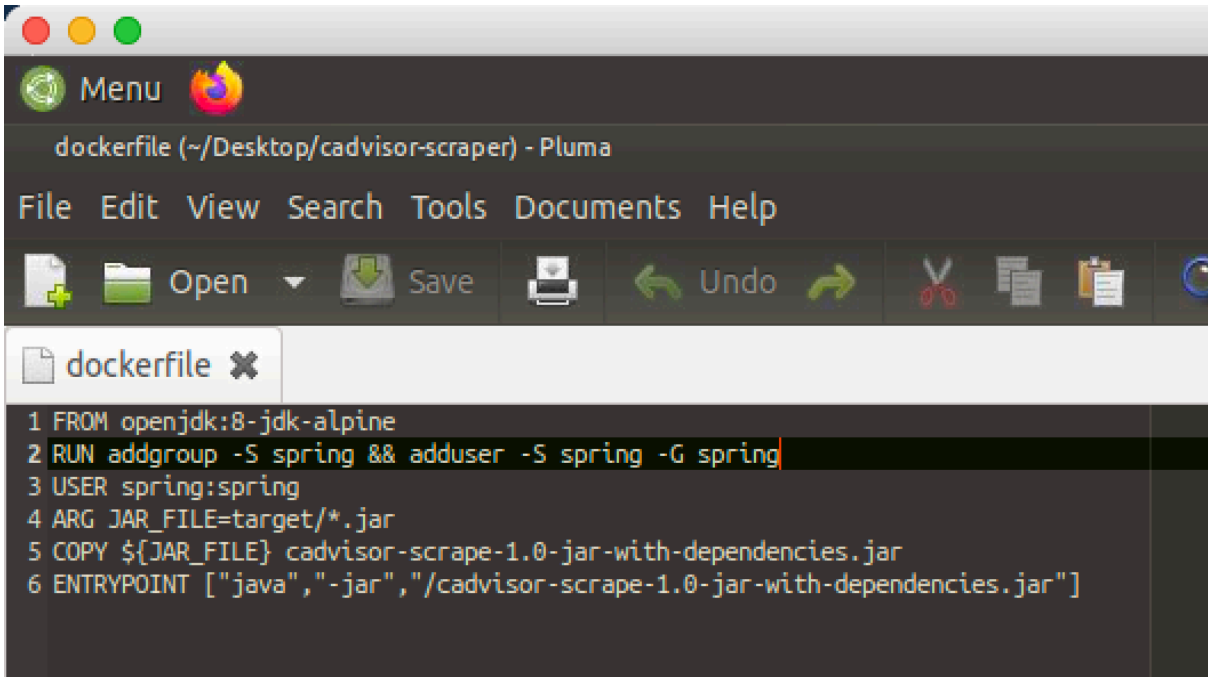


Figure 3.8

Figure 3.6 shows to build the docker image using the following command

*"docker build -t cadvisor-scraper ."*

```
student@ML-RefVm-691229:~/Desktop/cadvisor-scraper$ docker build -t cadvisor-scraper .
Sending build context to Docker daemon  3.709MB
Step 1/6 : FROM openjdk:8-jdk-alpine
 ---> a3562aa0b991
Step 2/6 : RUN addgroup -S spring && adduser -S spring -G spring
 ---> Using cache
 ---> c24531e9c807
Step 3/6 : USER spring:spring
 ---> Using cache
 ---> e0ed6181388a
Step 4/6 : ARG JAR_FILE=target/*.jar
 ---> Using cache
 ---> 139d65b66269
Step 5/6 : COPY ${JAR_FILE} cadvisor-scrape-1.0-jar-with-dependencies.jar
 ---> 971e5868a4e3
Step 6/6 : ENTRYPOINT ["java","-jar","/cadvisor-scrape-1.0-jar-with-dependencies.jar"]
 ---> Running in 5c91ea0642af
Removing intermediate container 5c91ea0642af
 ---> 85985b7c5cd2
Successfully built 85985b7c5cd2
Successfully tagged cadvisor-scraper:latest
student@ML-RefVm-691229:~/Desktop/cadvisor-scraper$
```

Figure 3.9

After building the image successfully as shown in figure 3.6 using the following command we can view the built image in the docker CLI

*"docker images"*

The following command is used to tag and rename the image name as shown in Figure 3.7

*"docker tag --imageid --name"*

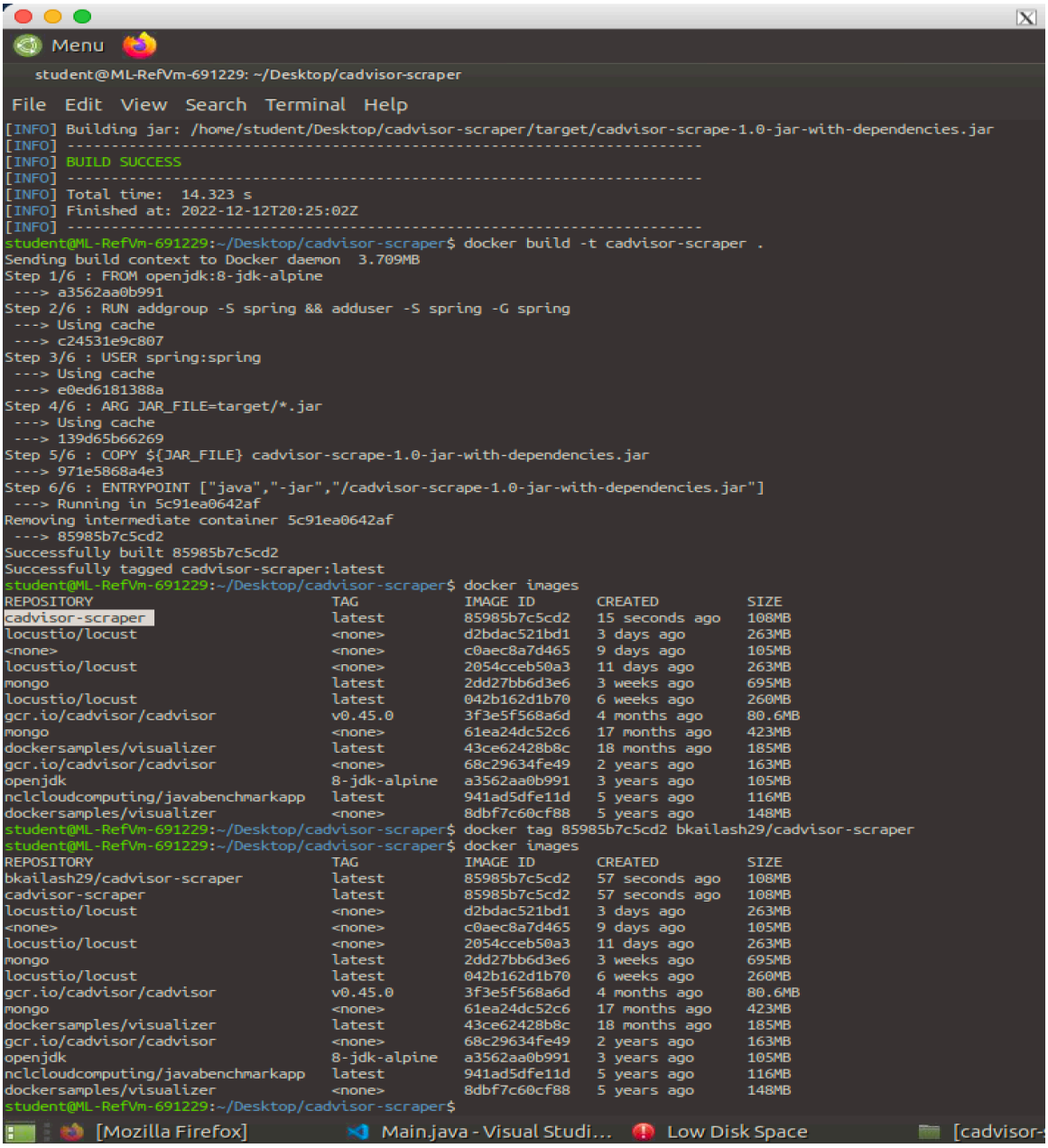


Figure 3.10

Figure 3.8 shows the image after renaming with the name

*"bkailash29/cadvisor-scraper"*

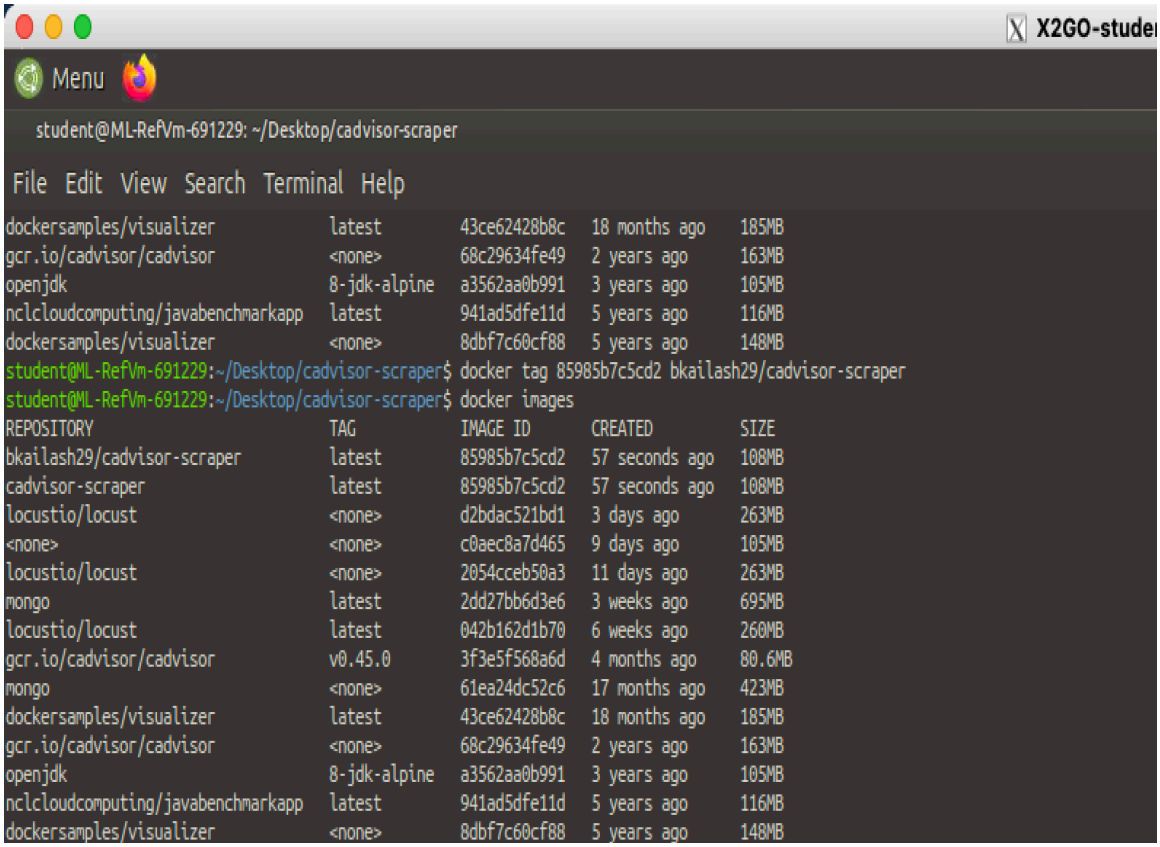


Figure 3.11

3. After code modification, pack the program as a standalone Docker image and push it to the Docker Hub. Name the image as cAdvisor-scraper.

   To push an image to the docker hub, first log into docker with the following command:

   *"docker login --username=username -p password"*

   Use the following command to push the image to Docker Hub.

   *"docker push bkailash29/cadvisor-scraper"*

   Use the following command to push the image to Docker Hub.

   *"docker pull bkailash29/cadvisor-scraper"*

   Use the following command to run the image to Docker Hub.

   *"docker run bkailash29/cadvisor-scraper"*

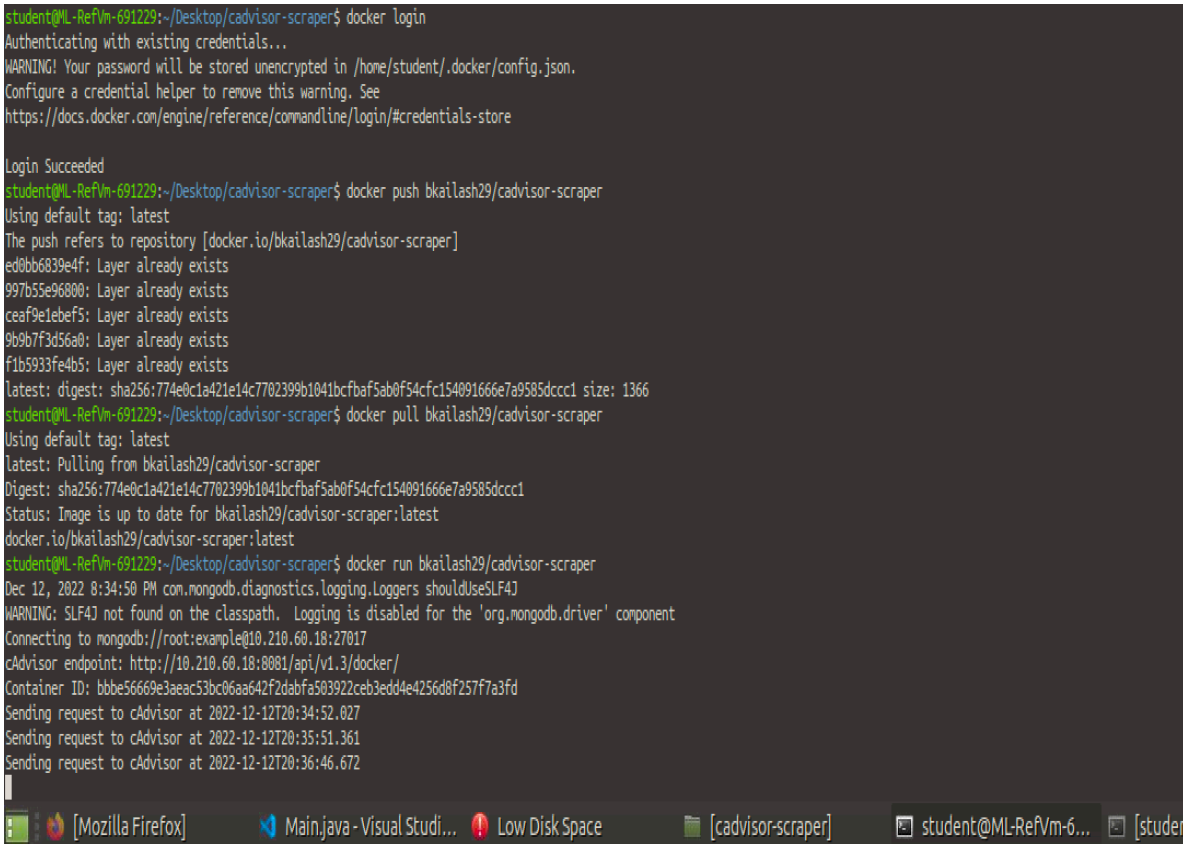


Figure 3.12

After the image is successfully pushed to Docker Hub, you can find the image on the Docker Hub website as shown in Figures 3.10 and 3.11, The URL is:

*"https://hub.docker.com/repository/docker/bkailash29/cadvisor-scraper"*

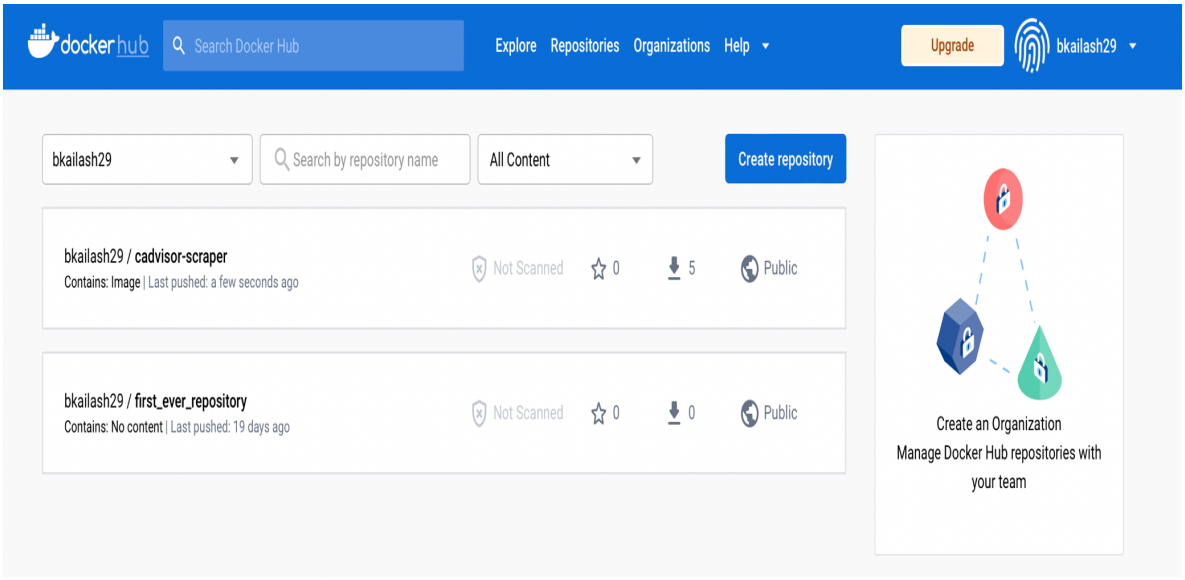


Figure 3.13

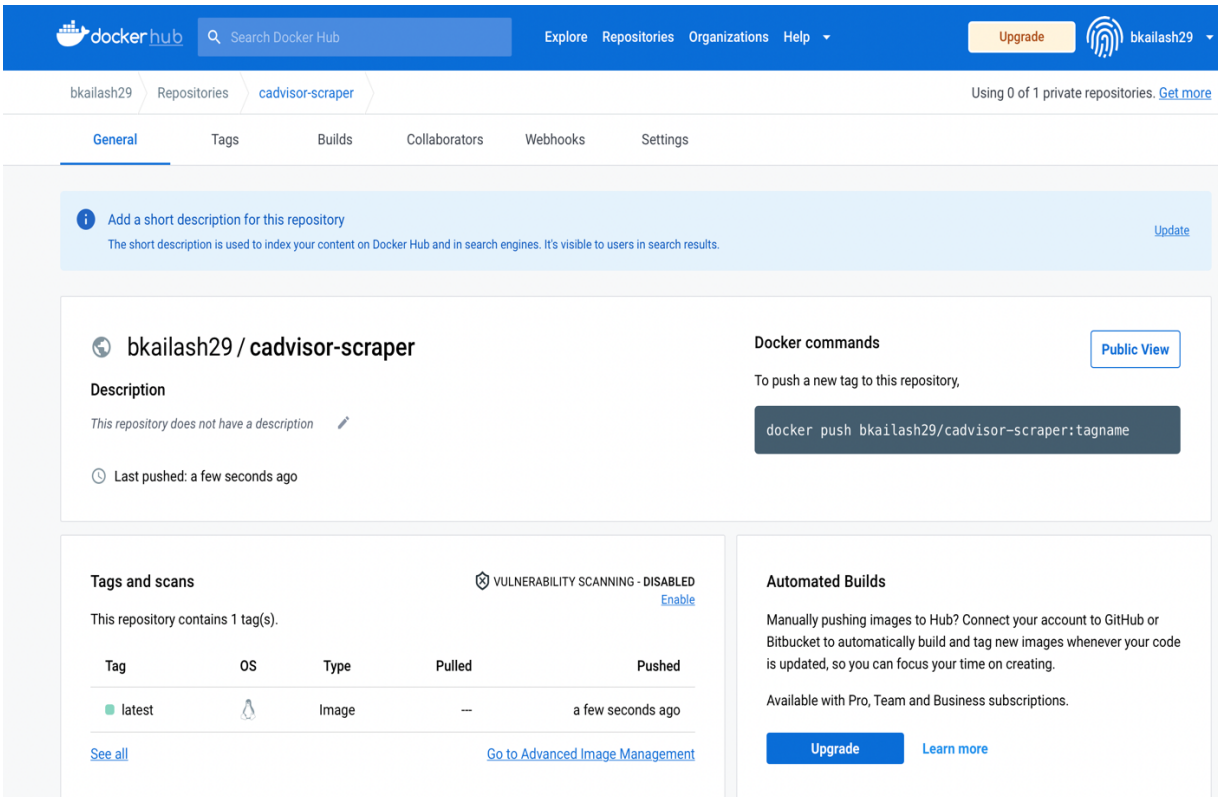


Figure 3.14

**Task 4:** Fully deploy and run the complex web application stack and undertake performance benchmarking activities

**Task Objective:** To learn and understand how to deploy complex web application stack, how to benchmark their run-time performance, and how to store the benchmarking results into a database server.

1. Pull the cAdvisor-scraper image, created in Task 3, back to the virtual machine on your laptop/PC.

   Use the following command to push the image to Docker Hub.

   *"docker push bkailash29/cadvisor-scraper"*

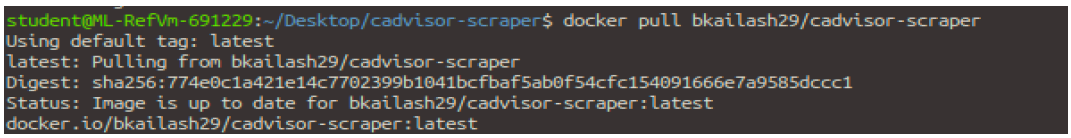


Figure 4.1

2. Make necessary changes to cAdvisor-scraper so that it monitors the swarm

   We can see in Figure 4.2 the data sent to Mongo DB by the web application. I have used mongo express to show the data. The application creates tables for the CPU, memory, each disk and each network interface.

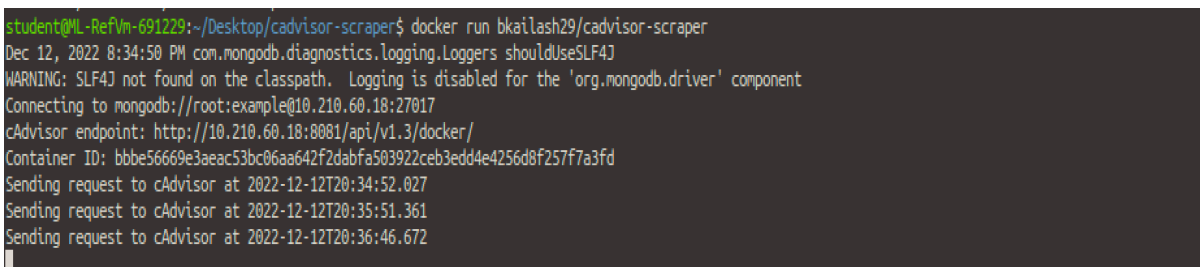


Figure 4.2

3. Launch cAdvisor-scraper as a container separately outside of the swarm

   Since we left the swarm the container ID has changed. Therefore, we change the container ID of the benchmark app and build and run the cAdvisor-scarper app separately in a container as shown in Figure 4.3

```
student@ML-RefVm-691229:~/Desktop/cadvisor-scraper$ docker container ls
CONTAINER ID   IMAGE                                  COMMAND                  CREATED         STATUS                   PORTS                    NAMES
383ac040b574   bkailash29/cadvisor-scraper            "java -jar /cadvisor…"   57 minutes ago  Up 57 minutes                                     distracted_lumiere
75e7c6cb6f2a   locustio/locust:latest                 "locust -f /mnt/locu…"   2 hours ago     Up 2 hours               5557/tcp, 8089/tcp       cloudproject_locust.1.jp45wxyiteg3curys6rcrdjr4
2847751b2ffd   dockersamples/visualizer:stable        "npm start"              2 hours ago     Up 2 hours               8080/tcp                 cloudproject_visualizer.1.bthxty5r5qjcx8tycv6xls03k
bbbe56669e3a   mongo:4.4.6                            "docker-entrypoint.s…"   2 hours ago     Up 2 hours               27017/tcp                cloudproject_mongo.1.sjhkisvqdjbd3p7eksu49h8wf
b92c5cf8c6a3   gcr.io/cadvisor/cadvisor:latest         "/usr/bin/cadvisor -…"   2 hours ago     Up 2 hours (healthy)     8080/tcp                 cloudproject_imagemonitor.1.ydltpaswjek89ydinhiazvkqi
da5f902b6dd6   nclcloudcomputing/javabenchmarkapp:latest  "/bin/sh -c 'exec ja…"   2 hours ago     Up 2 hours                                        cloudproject_webapplication.1.4sgg1mb5udjc1a1t79ig0q8ja
```

Figure 4.3

4. Send dynamic workload (e.g., http requests) to the Web Application Image Instance (in other words Web Application Container) by using the Load Generator Instance.

   Figure 4.4 shows that connecting to the web application container by using the load generator instance by using the following command

   *docker exec -it --mango containerid mango "mangodb://--username:--password@10.210.60.18:27017/"*

   *docker exec -it --bbbe56669e3a mango "mangodb://root:example@10.210.60.18:27017/"*

```
student@ML-RefVm-691229:~/Desktop/cadvisor-scraper$ docker exec -it bbbe56669e3a mongo "mongodb://root:example@10.210.60.18:27017/"
MongoDB shell version v4.4.6
connecting to: mongodb://10.210.60.18:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("3a5ce9cb-7188-4c16-bc00-8b066fd2d74d") }
MongoDB server version: 4.4.6
---
The server generated these startup warnings when booting:
        2022-12-12T19:54:35.195+00:00: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine. See http://dochub.mongodb.org/core/prodnotes-filesystem
        2022-12-12T19:54:44.839+00:00: /sys/kernel/mm/transparent_hugepage/enabled is 'always'. We suggest setting it to 'never'
---
---
        Enable MongoDB's free cloud-based monitoring service, which will then receive and display
        metrics about your deployment (disk utilization, CPU, operation statistics, etc).

        The monitoring data will be available on a MongoDB website with a unique URL accessible to you
        and anyone you share the URL with. MongoDB may use this information to make product
        improvements and to suggest MongoDB products and deployment options to you.

        To enable free monitoring, run the following command: db.enableFreeMonitoring()
        To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
---
```

Figure 4.4

5. Verify that the cAdvisor-scraper instance is logging the run-time performance statistics to the MongoDB instance.

   Figure 4.5 shows the performance statistics of the MongoDB instance

```
student@ML-RefVm-691229:~/Desktop/cadvisor-scraper$ docker exec -it bbbe56669e3a mongo "mongodb://root:example@10.210.60.18:27017/"
MongoDB shell version v4.4.6
connecting to: mongodb://10.210.60.18:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("3a5ce9cb-7188-4c16-bc00-8b066fd2d74d") }
MongoDB server version: 4.4.6
---
The server generated these startup warnings when booting:
        2022-12-12T19:54:35.195+00:00: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine. See http://dochub.mongodb.org/core/prodnotes-filesystem
        2022-12-12T19:54:44.839+00:00: /sys/kernel/mm/transparent_hugepage/enabled is 'always'. We suggest setting it to 'never'
---
---
        Enable MongoDB's free cloud-based monitoring service, which will then receive and display
        metrics about your deployment (disk utilization, CPU, operation statistics, etc).

        The monitoring data will be available on a MongoDB website with a unique URL accessible to you
        and anyone you share the URL with. MongoDB may use this information to make product
        improvements and to suggest MongoDB products and deployment options to you.

        To enable free monitoring, run the following command: db.enableFreeMonitoring()
        To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
---
> show dbs
admin   0.000GB
config  0.000GB
local   0.000GB
metrics 0.002GB
> use metrics
switched to db metrics
> show collections
bbbe5666-cpu
bbbe5666-disk_/dev/sdb
bbbe5666-memory
bbbe5666-network_eth0
bbbe5666-network_eth1
bbbe5666-network_eth2
>
```

Figure 4.5

**Task 5:** Deploy Kubernetes using Terraform and deploy a microservice.

Use Terraform to achieve the following tasks

1. Enable the Azure subscription

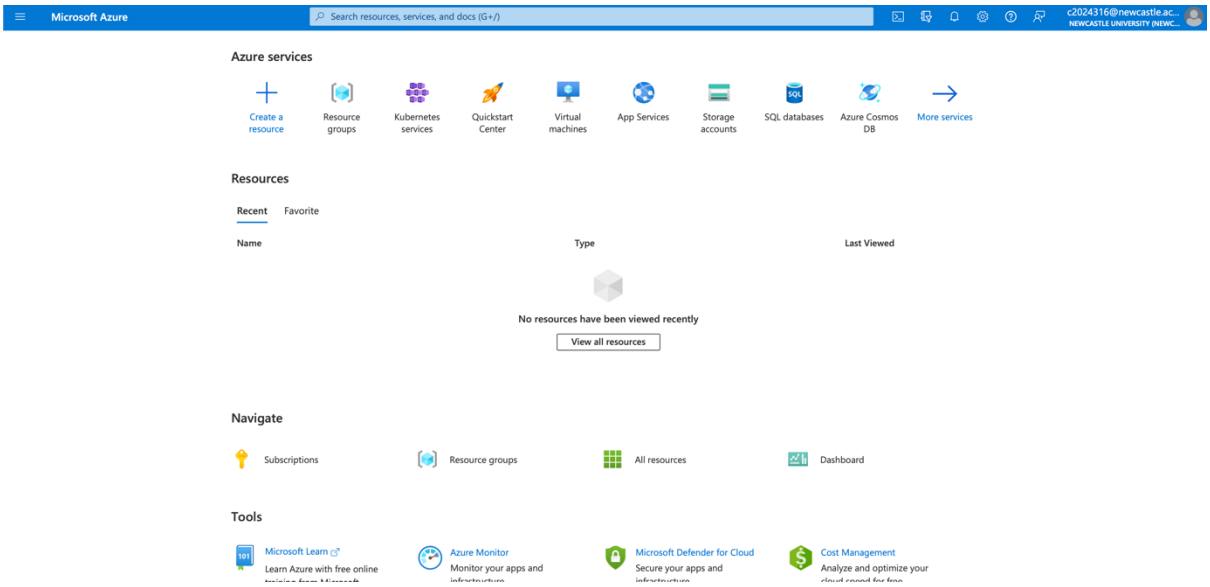


Figure 5.1

2. Create an Azure Kubernetes cluster named "csc8110" using HashiCorp language (HCL)

   The following figures are the files to create the Azure Kubernetes cluster

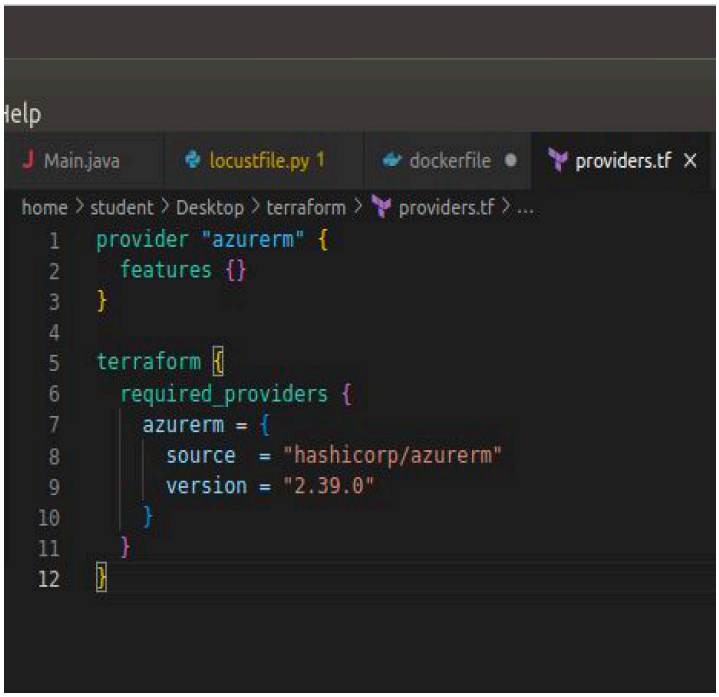


Figure 5.2

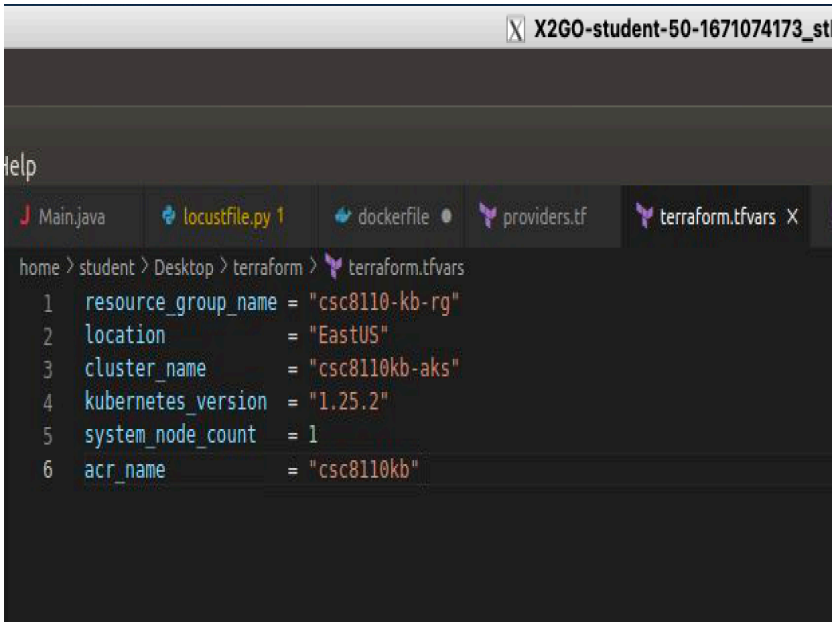


Figure 5.3

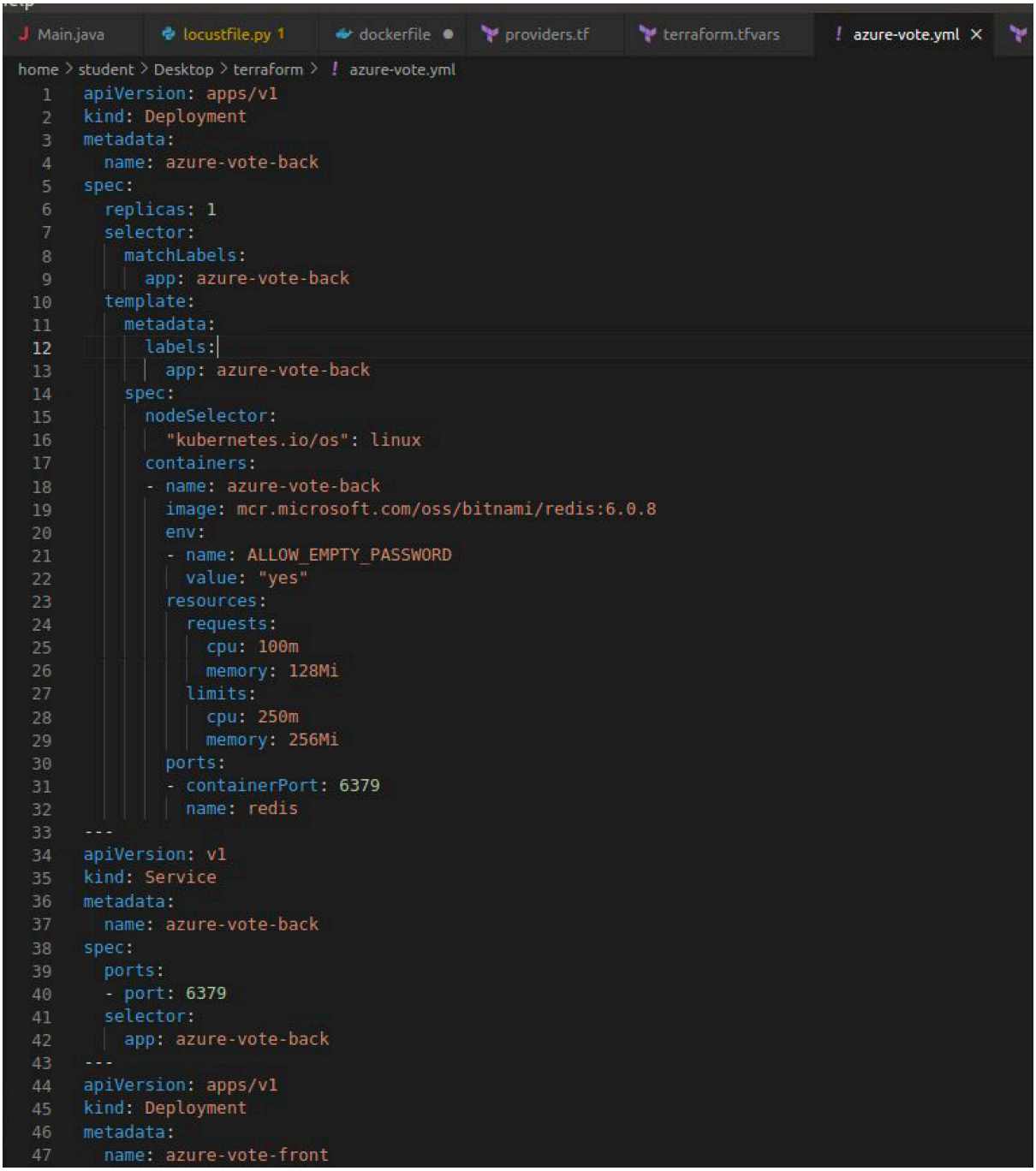
Main.java
locustfile.py 1
dockerfile
providers.tf
terraform.tfvars
azure-vote.yml
home > student > Desktop > terraform > azure-vote.yml
apiVersion: apps/v1
kind: Deployment
metadata:
name: azure-vote-back
spec:
replicas: 1
selector:
matchLabels:
app: azure-vote-back
template:
metadata:
labels:
app: azure-vote-back
spec:
nodeSelector:
"kubernetes.io/os": linux
containers:
- name: azure-vote-back
image: mcr.microsoft.com/oss/bitnami/redis:6.0.8
env:
- name: ALLOW_EMPTY_PASSWORD
value: "yes"
resources:
requests:
cpu: 100m
memory: 128Mi
limits:
cpu: 250m
memory: 256Mi
ports:
- containerPort: 6379
name: redis
---
apiVersion: v1
kind: Service
metadata:
name: azure-vote-back
spec:
ports:
- port: 6379
selector:
app: azure-vote-back
---
apiVersion: apps/v1
kind: Deployment
metadata:
name: azure-vote-front

Figure 5.4

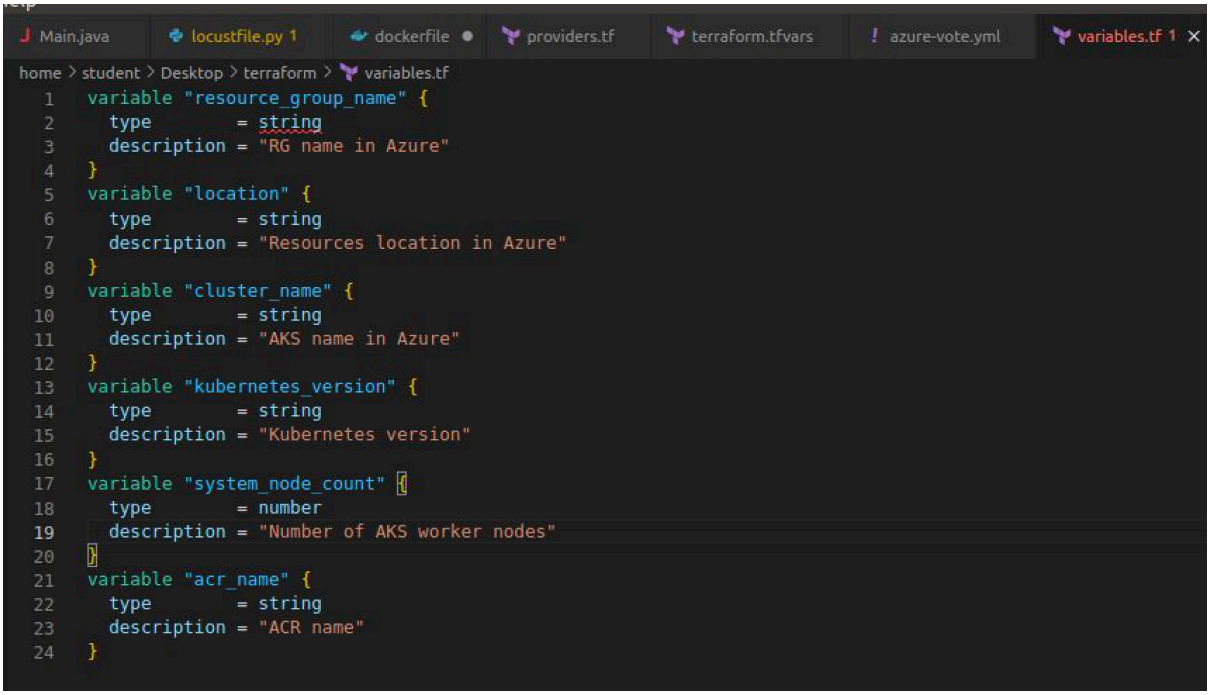
Main.java
locustFile.py 1
dockerfile
providers.tf
terraform.tfvars
azure-vote.yml
variables.tf 1
home > student > Desktop > terraform > variables.tf
variable "resource_group_name" {
type = string
description = "RG name in Azure"
}
variable "location" {
type = string
description = "Resources location in Azure"
}
variable "cluster_name" {
type = string
description = "AKS name in Azure"
}
variable "kubernetes_version" {
type = string
description = "Kubernetes version"
}
variable "system_node_count" {
type = number
description = "Number of AKS worker nodes"
}
variable "acr_name" {
type = string
description = "ACR name"
}

Figure 5.5

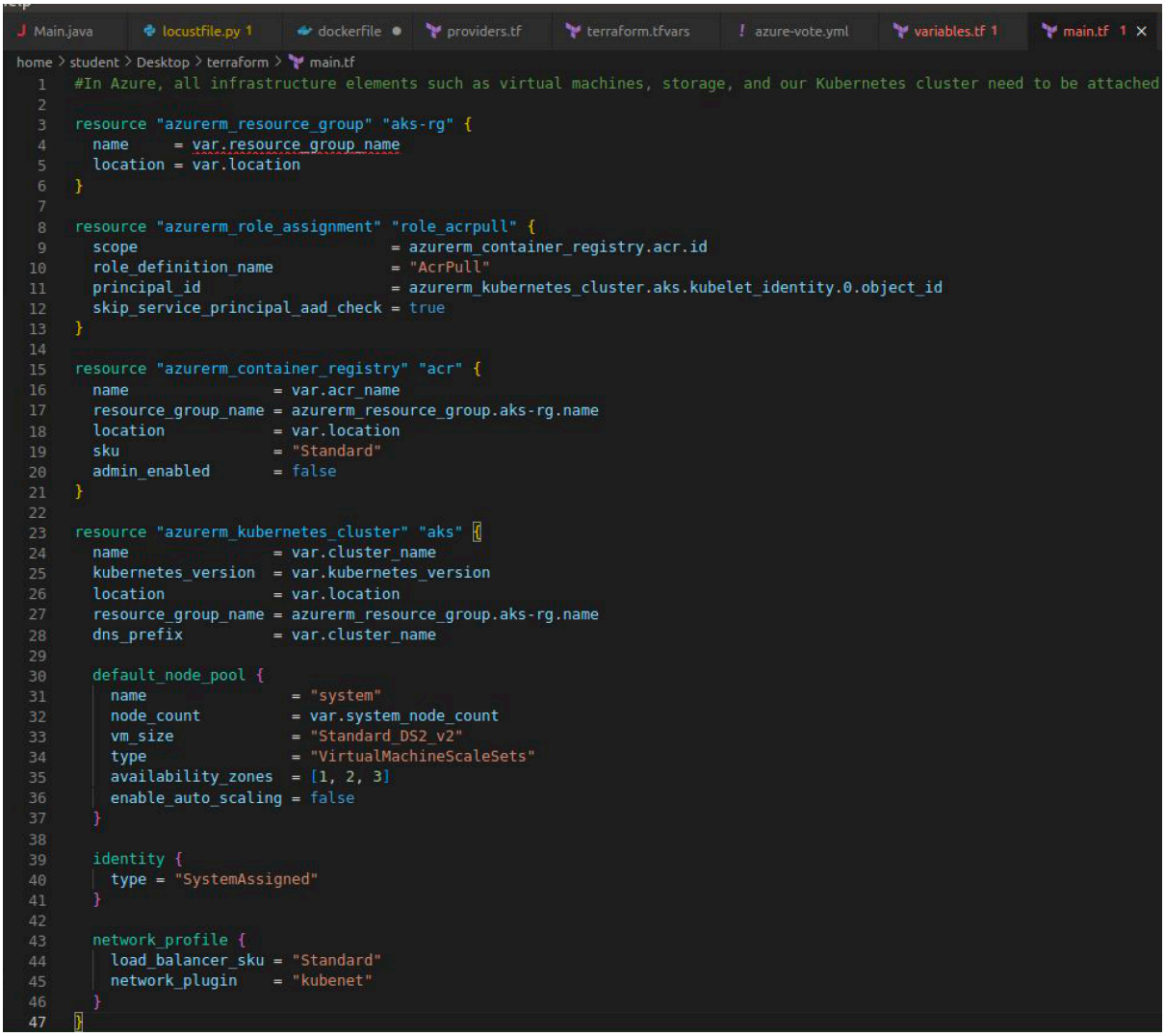


Figure 5.6

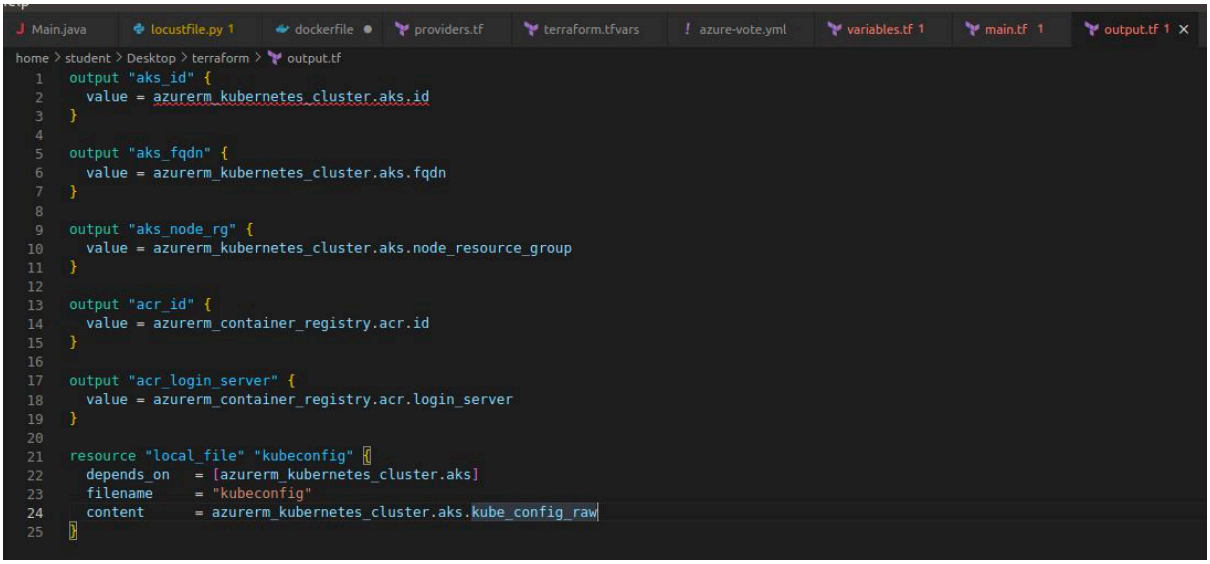


Figure 5.7

Initiate the terraform using the following command in the docker CLI as shown in figure 5.8

*"terraform init"*

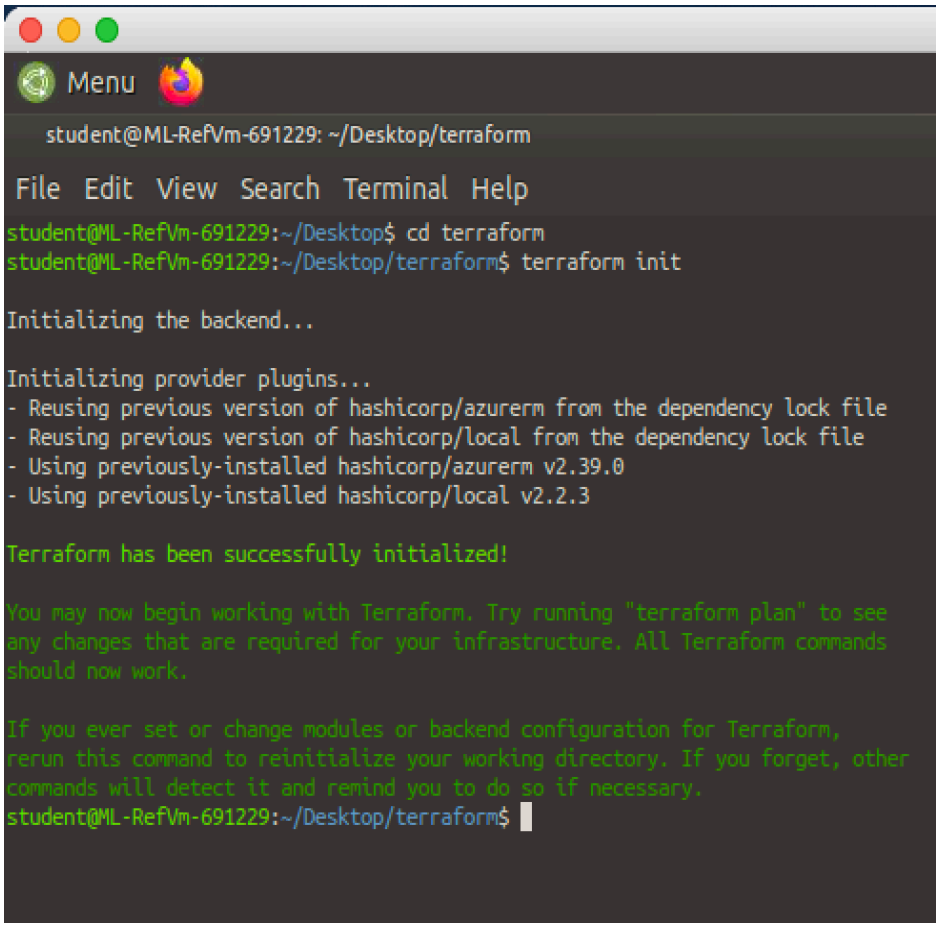


Figure 5.8

The following command executes the terraform

*"terraform plan"*

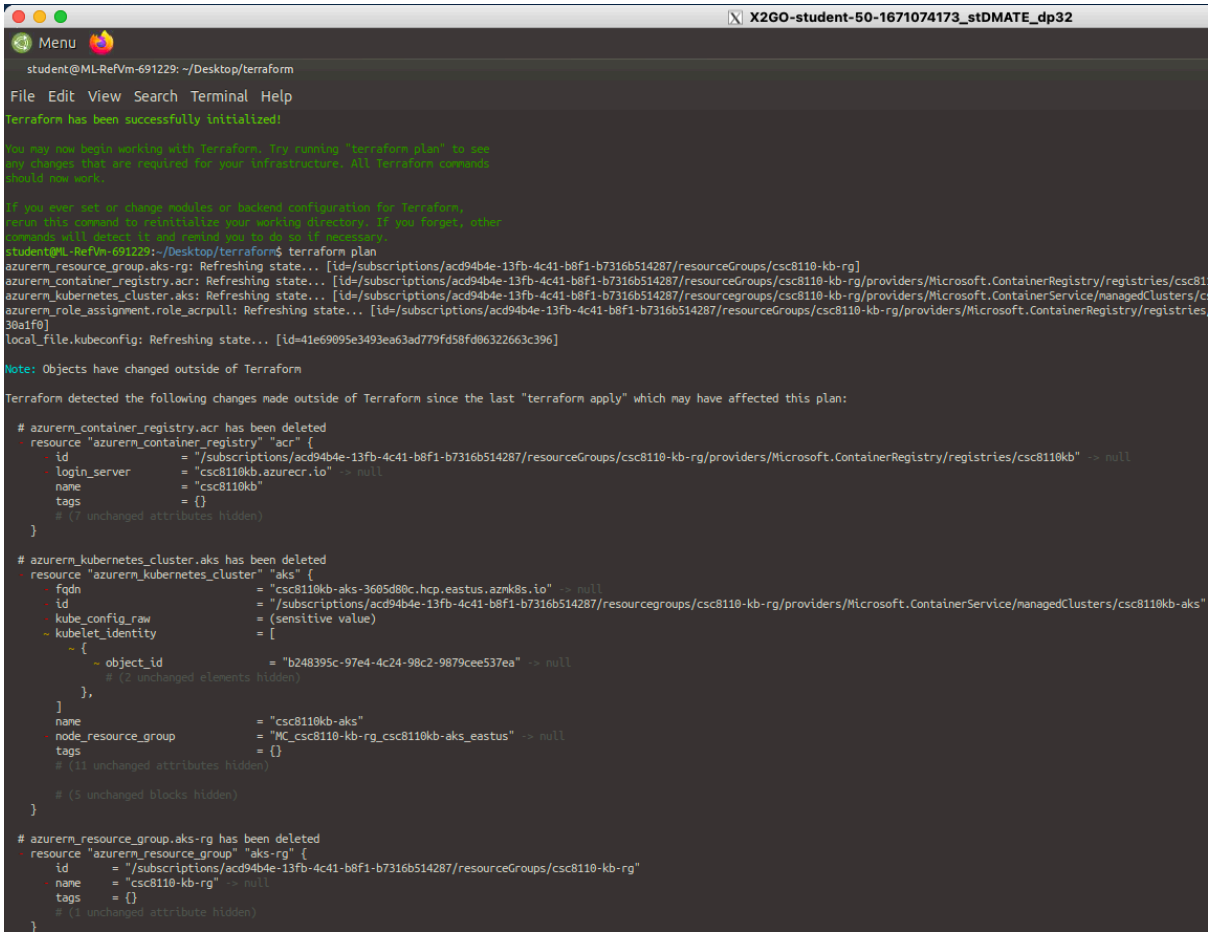
X2GO-student-50-1671074173_stDMATE_dp32
Menu
student@ML-RefVm-691229: ~/Desktop/terraform
File Edit View Search Terminal Help
Terraform has been successfully initialized!
You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.
If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
student@ML-RefVm-691229:~/Desktop/terraform$ terraform plan
azurerm_resource_group.aks-rg: Refreshing state... [id=/subscriptions/acd94b4e-13fb-4c41-b8f1-b7316b514287/resourceGroups/csc8110-kb-rg]
azurerm_container_registry.acr: Refreshing state... [id=/subscriptions/acd94b4e-13fb-4c41-b8f1-b7316b514287/resourceGroups/csc8110-kb-rg/providers/Microsoft.ContainerRegistry/registries/csc81
azurerm_kubernetes_cluster.aks: Refreshing state... [id=/subscriptions/acd94b4e-13fb-4c41-b8f1-b7316b514287/resourcegroups/csc8110-kb-rg/providers/Microsoft.ContainerService/managedClusters/c
azurerm_role_assignment.role_acrpull: Refreshing state... [id=/subscriptions/acd94b4e-13fb-4c41-b8f1-b7316b514287/resourceGroups/csc8110-kb-rg/providers/Microsoft.ContainerRegistry/registries
30a1f0]
local_file.kubeconfig: Refreshing state... [id=41e69095e3493ea63ad779fd58fd06322663c396]
Note: Objects have changed outside of Terraform
Terraform detected the following changes made outside of Terraform since the last "terraform apply" which may have affected this plan:
# azurerm_container_registry.acr has been deleted
- resource "azurerm_container_registry" "acr" {
- id = "/subscriptions/acd94b4e-13fb-4c41-b8f1-b7316b514287/resourceGroups/csc8110-kb-rg/providers/Microsoft.ContainerRegistry/registries/csc8110kb" -> null
- login_server = "csc8110kb.azurecr.io" -> null
name = "csc8110kb"
tags = {}
# (7 unchanged attributes hidden)
}
# azurerm_kubernetes_cluster.aks has been deleted
- resource "azurerm_kubernetes_cluster" "aks" {
- fqdn = "csc8110kb-aks-3605d80c.hcp.eastus.azmk8s.io" -> null
- id = "/subscriptions/acd94b4e-13fb-4c41-b8f1-b7316b514287/resourcegroups/csc8110-kb-rg/providers/Microsoft.ContainerService/managedClusters/csc8110kb-aks"
- kube_config_raw = (sensitive value)
~ kubelet_identity = [
~ {
~ object_id = "b248395c-97e4-4c24-98c2-9879cee537ea" -> null
# (2 unchanged elements hidden)
},
]
name = "csc8110kb-aks"
- node_resource_group = "MC_csc8110-kb-rg_csc8110kb-aks_eastus" -> null
tags = {}
# (11 unchanged attributes hidden)
# (5 unchanged blocks hidden)
}
# azurerm_resource_group.aks-rg has been deleted
- resource "azurerm_resource_group" "aks-rg" {
id = "/subscriptions/acd94b4e-13fb-4c41-b8f1-b7316b514287/resourceGroups/csc8110-kb-rg"
- name = "csc8110-kb-rg" -> null
tags = {}
# (1 unchanged attribute hidden)
}

Figure 5.9

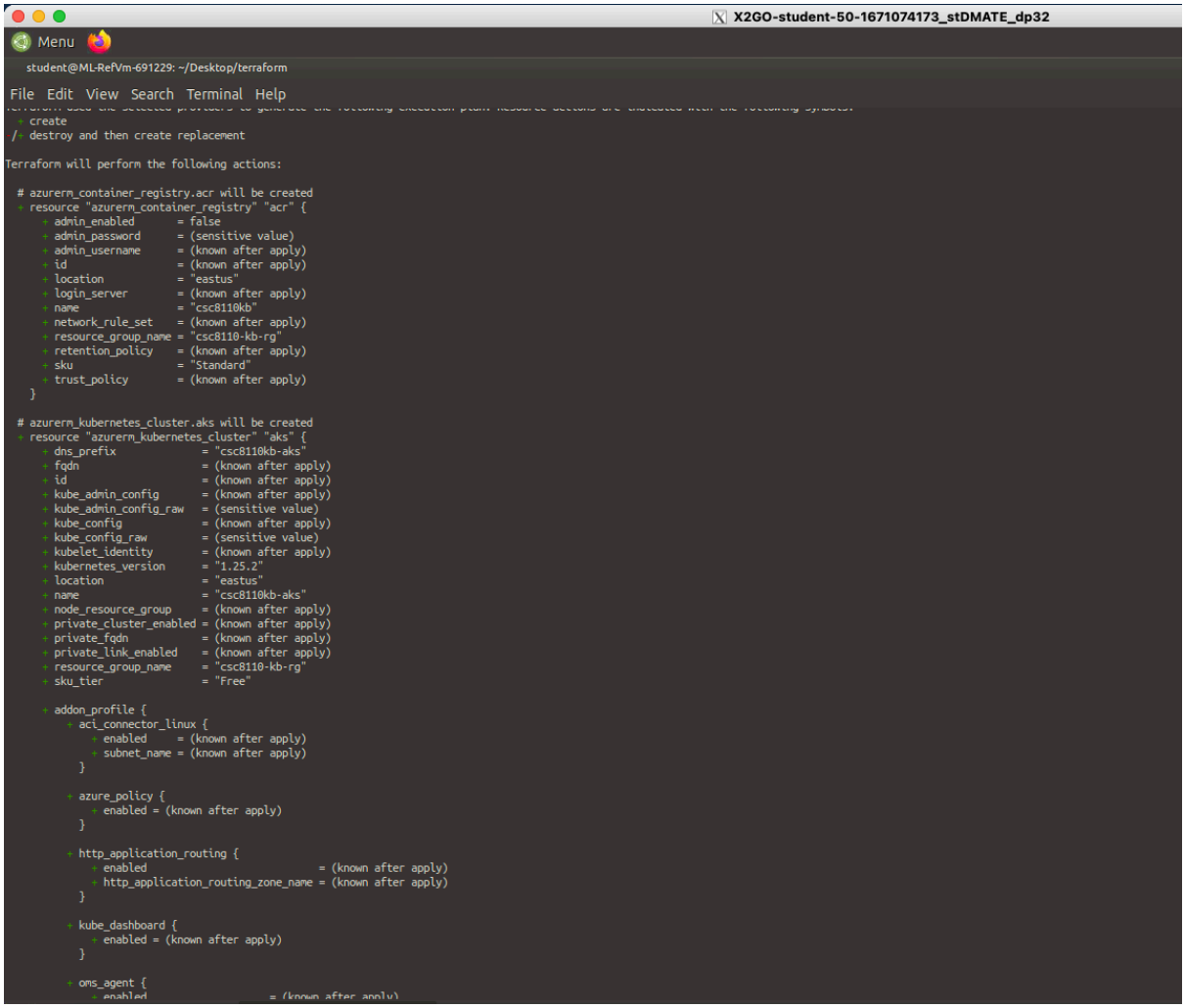
X2GO-student-50-1671074173_stDMATE_dp32
Menu
student@ML-RefVm-691229: ~/Desktop/terraform
File Edit View Search Terminal Help
+ create
/+ destroy and then create replacement
Terraform will perform the following actions:
# azurerm_container_registry.acr will be created
+ resource "azurerm_container_registry" "acr" {
+ admin_enabled = false
+ admin_password = (sensitive value)
+ admin_username = (known after apply)
+ id = (known after apply)
+ location = "eastus"
+ login_server = (known after apply)
+ name = "csc8110kb"
+ network_rule_set = (known after apply)
+ resource_group_name = "csc8110-kb-rg"
+ retention_policy = (known after apply)
+ sku = "Standard"
+ trust_policy = (known after apply)
}
# azurerm_kubernetes_cluster.aks will be created
+ resource "azurerm_kubernetes_cluster" "aks" {
+ dns_prefix = "csc8110kb-aks"
+ fqdn = (known after apply)
+ id = (known after apply)
+ kube_admin_config = (known after apply)
+ kube_admin_config_raw = (sensitive value)
+ kube_config = (known after apply)
+ kube_config_raw = (sensitive value)
+ kubelet_identity = (known after apply)
+ kubernetes_version = "1.25.2"
+ location = "eastus"
+ name = "csc8110kb-aks"
+ node_resource_group = (known after apply)
+ private_cluster_enabled = (known after apply)
+ private_fqdn = (known after apply)
+ private_link_enabled = (known after apply)
+ resource_group_name = "csc8110-kb-rg"
+ sku_tier = "Free"
+ addon_profile {
+ aci_connector_linux {
+ enabled = (known after apply)
+ subnet_name = (known after apply)
}
+ azure_policy {
+ enabled = (known after apply)
}
+ http_application_routing {
+ enabled = (known after apply)
+ http_application_routing_zone_name = (known after apply)
}
+ kube_dashboard {
+ enabled = (known after apply)
}
+ oms_agent {

Figure 5.10

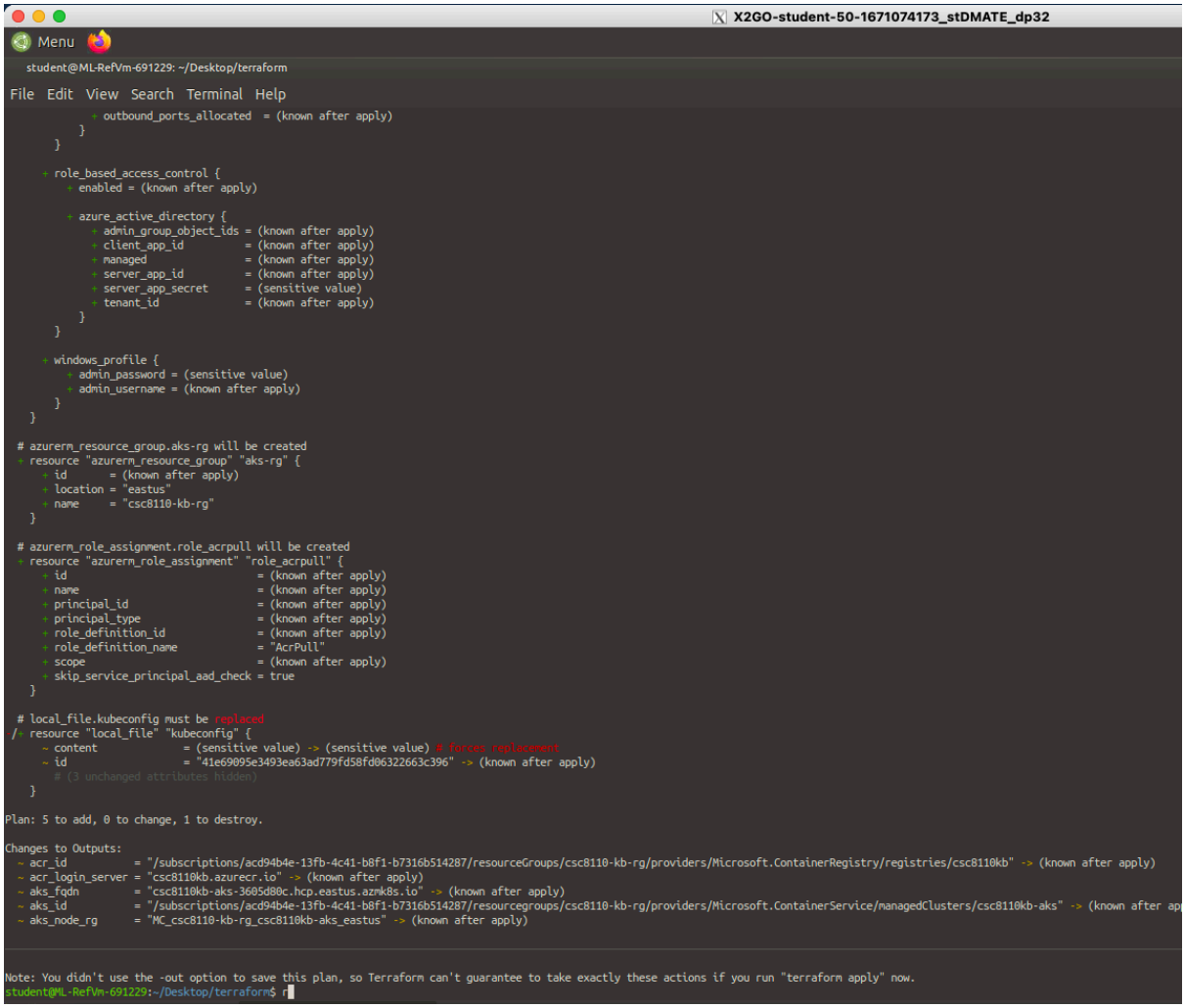


Figure 5.11

To save the executed plan run the following command as shown in the following figures

*"terraform apply"*

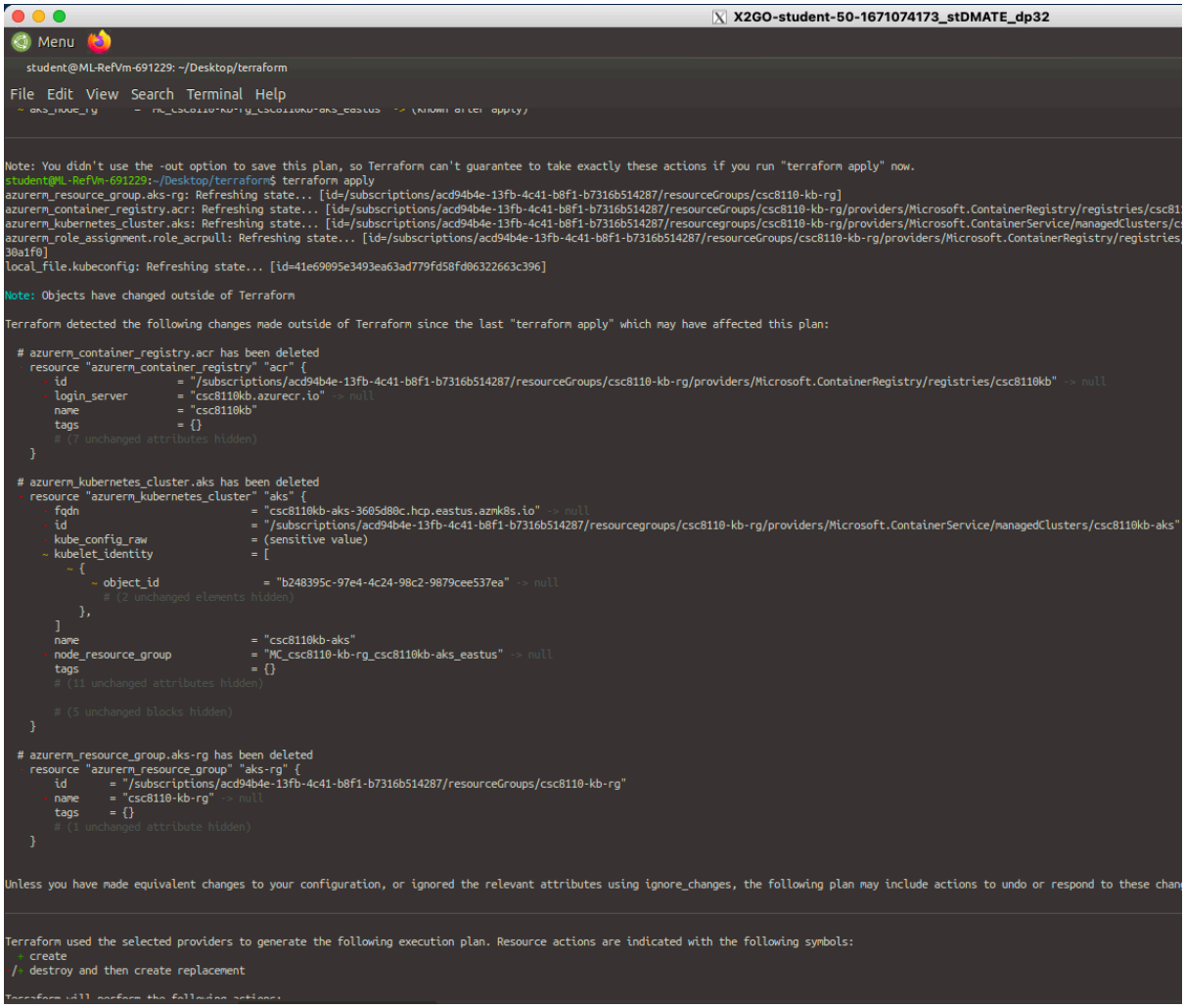
X2GO-student-50-1671074173_stDMATE_dp32
Menu
student@ML-RefVm-691229: ~/Desktop/terraform
File Edit View Search Terminal Help
Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions if you run "terraform apply" now.
student@ML-RefVm-691229:~/Desktop/terraform$ terraform apply
azurerm_resource_group.aks-rg: Refreshing state... [id=/subscriptions/acd94b4e-13fb-4c41-b8f1-b7316b514287/resourceGroups/csc8110-kb-rg]
azurerm_container_registry.acr: Refreshing state... [id=/subscriptions/acd94b4e-13fb-4c41-b8f1-b7316b514287/resourceGroups/csc8110-kb-rg/providers/Microsoft.ContainerRegistry/registries/csc81
azurerm_kubernetes_cluster.aks: Refreshing state... [id=/subscriptions/acd94b4e-13fb-4c41-b8f1-b7316b514287/resourcegroups/csc8110-kb-rg/providers/Microsoft.ContainerService/managedClusters/c
azurerm_role_assignment.role_acrpull: Refreshing state... [id=/subscriptions/acd94b4e-13fb-4c41-b8f1-b7316b514287/resourceGroups/csc8110-kb-rg/providers/Microsoft.ContainerRegistry/registries
30a1f0]
local_file.kubeconfig: Refreshing state... [id=41e69095e3493ea63ad779fd58fd06322663c396]
Note: Objects have changed outside of Terraform
Terraform detected the following changes made outside of Terraform since the last "terraform apply" which may have affected this plan:
# azurerm_container_registry.acr has been deleted
- resource "azurerm_container_registry" "acr" {
- id = "/subscriptions/acd94b4e-13fb-4c41-b8f1-b7316b514287/resourceGroups/csc8110-kb-rg/providers/Microsoft.ContainerRegistry/registries/csc8110kb" -> null
- login_server = "csc8110kb.azurecr.io" -> null
name = "csc8110kb"
tags = {}
# (7 unchanged attributes hidden)
}
# azurerm_kubernetes_cluster.aks has been deleted
- resource "azurerm_kubernetes_cluster" "aks" {
- fqdn = "csc8110kb-aks-3605d80c.hcp.eastus.azmk8s.io" -> null
- id = "/subscriptions/acd94b4e-13fb-4c41-b8f1-b7316b514287/resourcegroups/csc8110-kb-rg/providers/Microsoft.ContainerService/managedClusters/csc8110kb-aks"
- kube_config_raw = (sensitive value)
~ kubelet_identity = [
~ {
~ object_id = "b248395c-97e4-4c24-98c2-9879cee537ea" -> null
# (2 unchanged elements hidden)
},
]
name = "csc8110kb-aks"
- node_resource_group = "MC_csc8110-kb-rg_csc8110kb-aks_eastus" -> null
tags = {}
# (11 unchanged attributes hidden)
# (5 unchanged blocks hidden)
}
# azurerm_resource_group.aks-rg has been deleted
- resource "azurerm_resource_group" "aks-rg" {
id = "/subscriptions/acd94b4e-13fb-4c41-b8f1-b7316b514287/resourceGroups/csc8110-kb-rg"
- name = "csc8110-kb-rg" -> null
tags = {}
# (1 unchanged attribute hidden)
}
Unless you have made equivalent changes to your configuration, or ignored the relevant attributes using ignore_changes, the following plan may include actions to undo or respond to these chan
Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
+ create
-/+ destroy and then create replacement

Figure 5.12

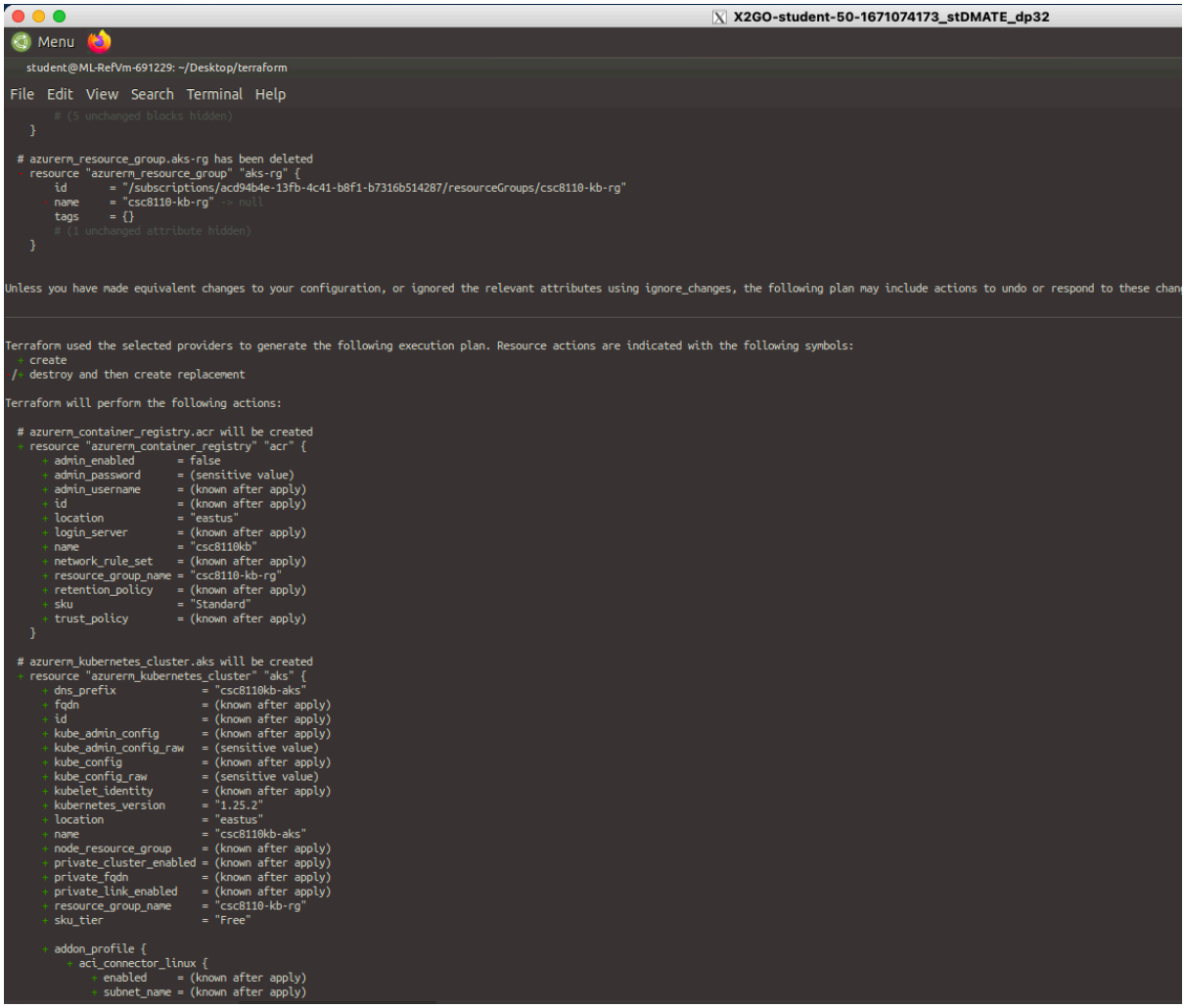
X2GO-student-50-1671074173_stDMATE_dp32
Menu
student@ML-RefVm-691229: ~/Desktop/terraform
File Edit View Search Terminal Help
# (5 unchanged blocks hidden)
}
# azurerm_resource_group.aks-rg has been deleted
- resource "azurerm_resource_group" "aks-rg" {
id = "/subscriptions/acd94b4e-13fb-4c41-b8f1-b7316b514287/resourceGroups/csc8110-kb-rg"
- name = "csc8110-kb-rg" -> null
tags = {}
# (1 unchanged attribute hidden)
}
Unless you have made equivalent changes to your configuration, or ignored the relevant attributes using ignore_changes, the following plan may include actions to undo or respond to these chan
Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
+ create
-/+ destroy and then create replacement
Terraform will perform the following actions:
# azurerm_container_registry.acr will be created
+ resource "azurerm_container_registry" "acr" {
+ admin_enabled = false
+ admin_password = (sensitive value)
+ admin_username = (known after apply)
+ id = (known after apply)
+ location = "eastus"
+ login_server = (known after apply)
+ name = "csc8110kb"
+ network_rule_set = (known after apply)
+ resource_group_name = "csc8110-kb-rg"
+ retention_policy = (known after apply)
+ sku = "Standard"
+ trust_policy = (known after apply)
}
# azurerm_kubernetes_cluster.aks will be created
+ resource "azurerm_kubernetes_cluster" "aks" {
+ dns_prefix = "csc8110kb-aks"
+ fqdn = (known after apply)
+ id = (known after apply)
+ kube_admin_config = (known after apply)
+ kube_admin_config_raw = (sensitive value)
+ kube_config = (known after apply)
+ kube_config_raw = (sensitive value)
+ kubelet_identity = (known after apply)
+ kubernetes_version = "1.25.2"
+ location = "eastus"
+ name = "csc8110kb-aks"
+ node_resource_group = (known after apply)
+ private_cluster_enabled = (known after apply)
+ private_fqdn = (known after apply)
+ private_link_enabled = (known after apply)
+ resource_group_name = "csc8110-kb-rg"
+ sku_tier = "Free"
+ addon_profile {
+ aci_connector_linux {
+ enabled = (known after apply)
+ subnet_name = (known after apply)

Figure 5.13

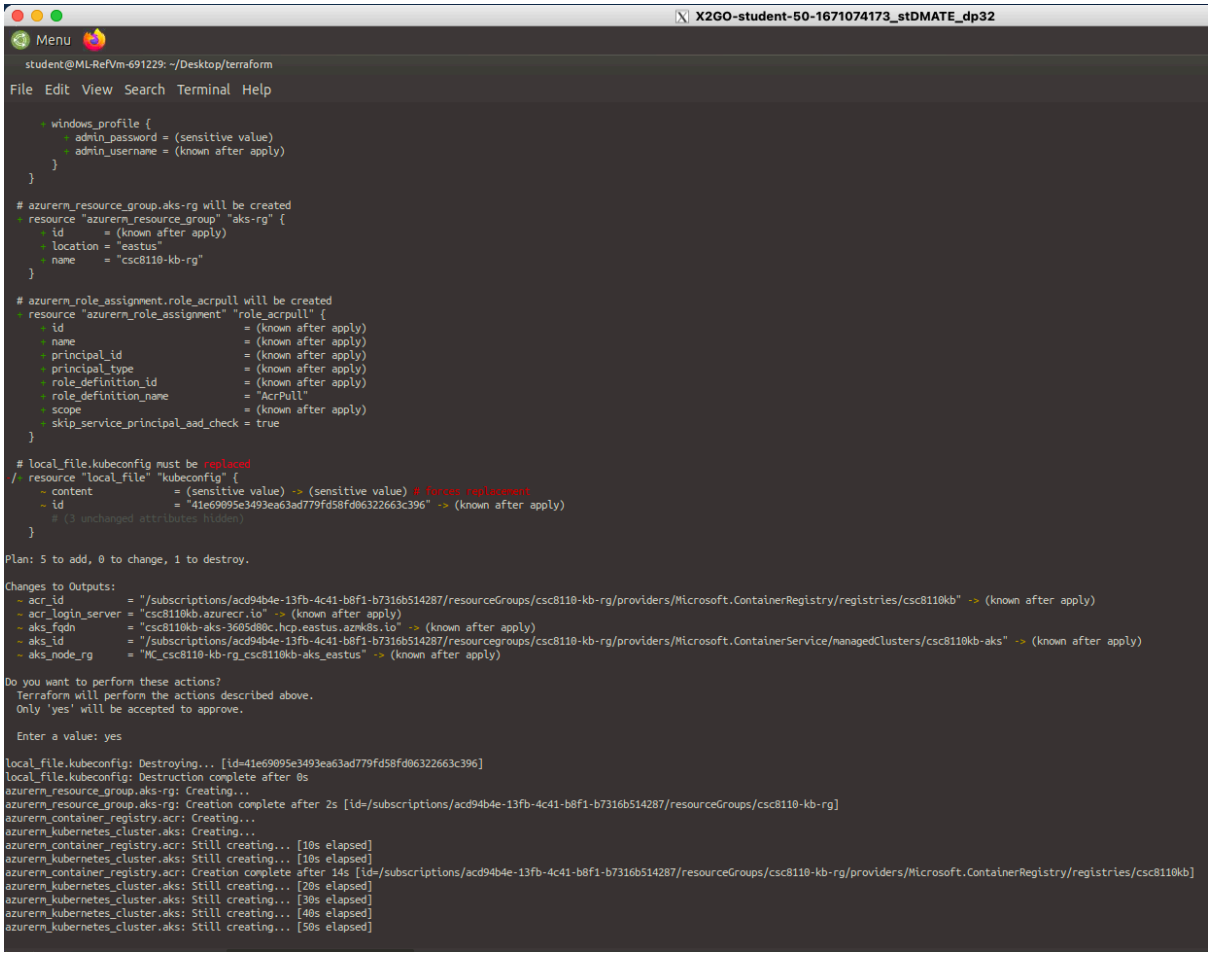
X2GO-student-50-1671074173_stDMATE_dp32
Menu
student@ML-RefVm-691229: ~/Desktop/terraform
File Edit View Search Terminal Help
+ windows_profile {
+ admin_password = (sensitive value)
+ admin_username = (known after apply)
}
}
# azurerm_resource_group.aks-rg will be created
+ resource "azurerm_resource_group" "aks-rg" {
+ id = (known after apply)
+ location = "eastus"
+ name = "csc8110-kb-rg"
}
# azurerm_role_assignment.role_acrpull will be created
+ resource "azurerm_role_assignment" "role_acrpull" {
+ id = (known after apply)
+ name = (known after apply)
+ principal_id = (known after apply)
+ principal_type = (known after apply)
+ role_definition_id = (known after apply)
+ role_definition_name = "AcrPull"
+ scope = (known after apply)
+ skip_service_principal_aad_check = true
}
# local_file.kubeconfig must be replaced
/+ resource "local_file" "kubeconfig" {
~ content = (sensitive value) -> (sensitive value) # forces replacement
~ id = "41e69095e3493ea63ad779fd58fd06322663c396" -> (known after apply)
# (3 unchanged attributes hidden)
}
Plan: 5 to add, 0 to change, 1 to destroy.
Changes to Outputs:
~ acr_id = "/subscriptions/acd94b4e-13fb-4c41-b8f1-b7316b514287/resourceGroups/csc8110-kb-rg/providers/Microsoft.ContainerRegistry/registries/csc8110kb" -> (known after apply)
~ acr_login_server = "csc8110kb.azurecr.io" -> (known after apply)
~ aks_fqdn = "csc8110kb-aks-3605d80c.hcp.eastus.azmk8s.io" -> (known after apply)
~ aks_id = "/subscriptions/acd94b4e-13fb-4c41-b8f1-b7316b514287/resourcegroups/csc8110-kb-rg/providers/Microsoft.ContainerService/managedClusters/csc8110kb-aks" -> (known after apply)
~ aks_node_rg = "MC_csc8110-kb-rg_csc8110kb-aks_eastus" -> (known after apply)
Do you want to perform these actions?
Terraform will perform the actions described above.
Only 'yes' will be accepted to approve.
Enter a value: yes
local_file.kubeconfig: Destroying... [id=41e69095e3493ea63ad779fd58fd06322663c396]
local_file.kubeconfig: Destruction complete after 0s
azurerm_resource_group.aks-rg: Creating...
azurerm_resource_group.aks-rg: Creation complete after 2s [id=/subscriptions/acd94b4e-13fb-4c41-b8f1-b7316b514287/resourceGroups/csc8110-kb-rg]
azurerm_container_registry.acr: Creating...
azurerm_kubernetes_cluster.aks: Creating...
azurerm_container_registry.acr: Still creating... [10s elapsed]
azurerm_kubernetes_cluster.aks: Still creating... [10s elapsed]
azurerm_container_registry.acr: Creation complete after 14s [id=/subscriptions/acd94b4e-13fb-4c41-b8f1-b7316b514287/resourceGroups/csc8110-kb-rg/providers/Microsoft.ContainerRegistry/registries/csc8110kb]
azurerm_kubernetes_cluster.aks: Still creating... [20s elapsed]
azurerm_kubernetes_cluster.aks: Still creating... [30s elapsed]
azurerm_kubernetes_cluster.aks: Still creating... [40s elapsed]
azurerm_kubernetes_cluster.aks: Still creating... [50s elapsed]

Figure 5.14

```
  Enter a value: yes

local_file.kubeconfig: Destroying... [id=41e69095e3493ea63ad779fd58fd06322663c396]
local_file.kubeconfig: Destruction complete after 0s
azurerm_resource_group.aks-rg: Creating...
azurerm_resource_group.aks-rg: Creation complete after 2s [id=/subscriptions/acd94b4e-13fb-4c41-b8f1-b7316b514287/resourceGroups/csc8110-kb-rg]
azurerm_container_registry.acr: Creating...
azurerm_kubernetes_cluster.aks: Creating...
azurerm_container_registry.acr: Still creating... [10s elapsed]
azurerm_kubernetes_cluster.aks: Still creating... [10s elapsed]
azurerm_container_registry.acr: Creation complete after 14s [id=/subscriptions/acd94b4e-13fb-4c41-b8f1-b7316b514287/resourceGroups/csc8110-kb-rg/providers/Micr
azurerm_kubernetes_cluster.aks: Still creating... [20s elapsed]
azurerm_kubernetes_cluster.aks: Still creating... [30s elapsed]
azurerm_kubernetes_cluster.aks: Still creating... [40s elapsed]
azurerm_kubernetes_cluster.aks: Still creating... [50s elapsed]
azurerm_kubernetes_cluster.aks: Still creating... [1m0s elapsed]
azurerm_kubernetes_cluster.aks: Still creating... [1m10s elapsed]
azurerm_kubernetes_cluster.aks: Still creating... [1m20s elapsed]
azurerm_kubernetes_cluster.aks: Still creating... [1m30s elapsed]
azurerm_kubernetes_cluster.aks: Still creating... [1m40s elapsed]
azurerm_kubernetes_cluster.aks: Still creating... [1m50s elapsed]
azurerm_kubernetes_cluster.aks: Still creating... [2m0s elapsed]
azurerm_kubernetes_cluster.aks: Still creating... [2m10s elapsed]
azurerm_kubernetes_cluster.aks: Still creating... [2m20s elapsed]
azurerm_kubernetes_cluster.aks: Still creating... [2m30s elapsed]
azurerm_kubernetes_cluster.aks: Still creating... [2m40s elapsed]
azurerm_kubernetes_cluster.aks: Still creating... [2m50s elapsed]
azurerm_kubernetes_cluster.aks: Still creating... [3m0s elapsed]
azurerm_kubernetes_cluster.aks: Still creating... [3m10s elapsed]
azurerm_kubernetes_cluster.aks: Still creating... [3m20s elapsed]
azurerm_kubernetes_cluster.aks: Still creating... [3m30s elapsed]
azurerm_kubernetes_cluster.aks: Still creating... [3m40s elapsed]
azurerm_kubernetes_cluster.aks: Still creating... [3m50s elapsed]
azurerm_kubernetes_cluster.aks: Still creating... [4m0s elapsed]
azurerm_kubernetes_cluster.aks: Still creating... [4m10s elapsed]
azurerm_kubernetes_cluster.aks: Creation complete after 4m14s [id=/subscriptions/acd94b4e-13fb-4c41-b8f1-b7316b514287/resourcegroups/csc8110-kb-rg/providers/Mi
azurerm_role_assignment.role_acrpull: Creating...
local_file.kubeconfig: Creating...
local_file.kubeconfig: Creation complete after 0s [id=577c98b18a1d7c924f70c1a5af7243c3859ee799]
azurerm_role_assignment.role_acrpull: Still creating... [10s elapsed]
azurerm_role_assignment.role_acrpull: Still creating... [20s elapsed]
azurerm_role_assignment.role_acrpull: Creation complete after 25s [id=/subscriptions/acd94b4e-13fb-4c41-b8f1-b7316b514287/resourceGroups/csc8110-kb-rg/provider
1-8193d9a48ffb]

Apply complete! Resources: 5 added, 0 changed, 1 destroyed.

Outputs:

acr_id = "/subscriptions/acd94b4e-13fb-4c41-b8f1-b7316b514287/resourceGroups/csc8110-kb-rg/providers/Microsoft.ContainerRegistry/registries/csc8110kb"
acr_login_server = "csc8110kb.azurecr.io"
aks_fqdn = "csc8110kb-aks-f490ab55.hcp.eastus.azmk8s.io"
aks_id = "/subscriptions/acd94b4e-13fb-4c41-b8f1-b7316b514287/resourcegroups/csc8110-kb-rg/providers/Microsoft.ContainerService/managedClusters/csc8110kb-aks"
aks_node_rg = "MC_csc8110-kb-rg_csc8110kb-aks_eastus"
student@ML-RefVm-691229:~/Desktop/terraform$
```

Figure 5.15

To run the kubeconfig file following command is used

*"az aks get-credentials --resourse-group csc8110-kb-rg –name csc8110kb-aks"*

```
acr_id = "/subscriptions/acd94b4e-13fb-4c41-b8f1-b7316b514287/resourceGroups/csc8110-kb-rg/providers/Microsoft.ContainerRegistry/registries/csc8110kb"
acr_login_server = "csc8110kb.azurecr.io"
aks_fqdn = "csc8110kb-aks-f490ab55.hcp.eastus.azmk8s.io"
aks_id = "/subscriptions/acd94b4e-13fb-4c41-b8f1-b7316b514287/resourcegroups/csc8110-kb-rg/providers/Microsoft.ContainerService/managedClusters/csc8110kb-aks"
aks_node_rg = "MC_csc8110-kb-rg_csc8110kb-aks_eastus"
student@ML-RefVm-691229:~/Desktop/terraform$ az aks get-credentials --resource-group csc8110-kb-rg --name csc8110kb-aks
A different object named csc8110kb-aks already exists in your kubeconfig file.
Overwrite? (y/n): y
A different object named clusterUser_csc8110-kb-rg_csc8110kb-aks already exists in your kubeconfig file.
Overwrite? (y/n): yes
A different object named clusterUser_csc8110-kb-rg_csc8110kb-aks already exists in your kubeconfig file.
Overwrite? (y/n): y
Merged "csc8110kb-aks" as current context in /home/student/.kube/config
student@ML-RefVm-691229:~/Desktop/terraform$
```

Figure 5.16

The following command is used to get the Kubernetes node

*"kubectl get nodes"*

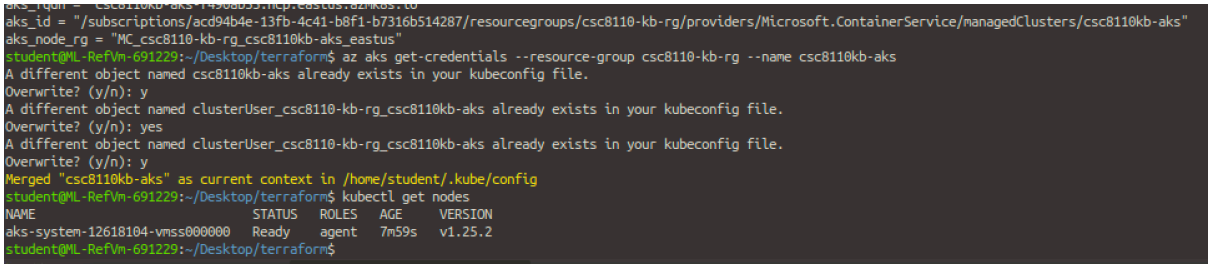


Figure 5.17

The following command is used to create the service and deploy the application by running the azure-vote.yml file

*"kubectl apply -f azure-vote.yml"*

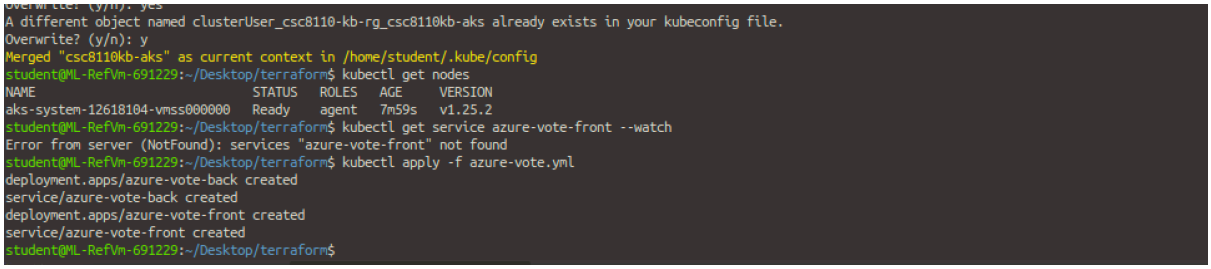


Figure 5.18

3. Deploy any sample microservice application once the cluster is ready

Figure 5.19 shows that the microservice application was created and running in 20.242.152.59

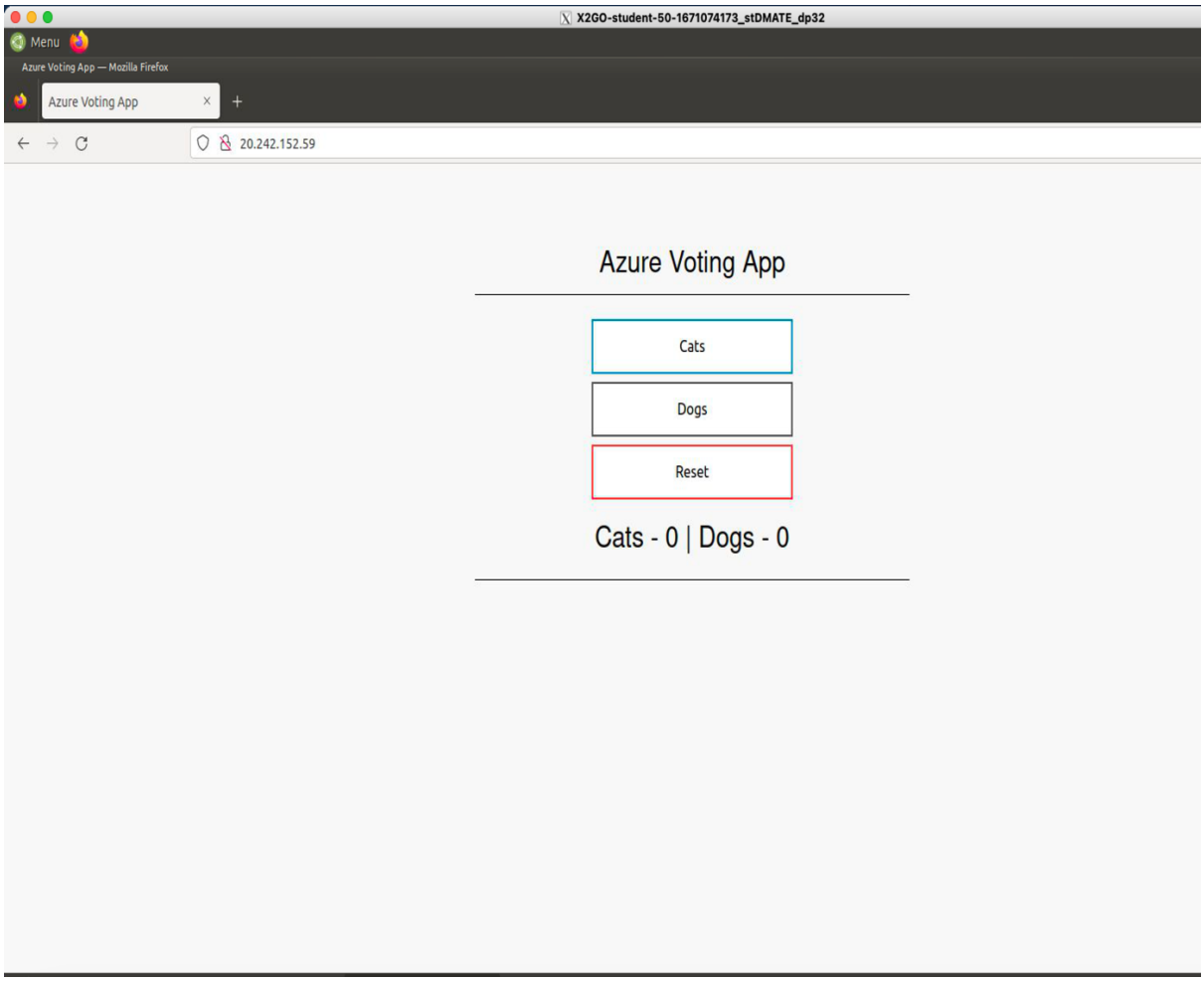


Figure 5.19

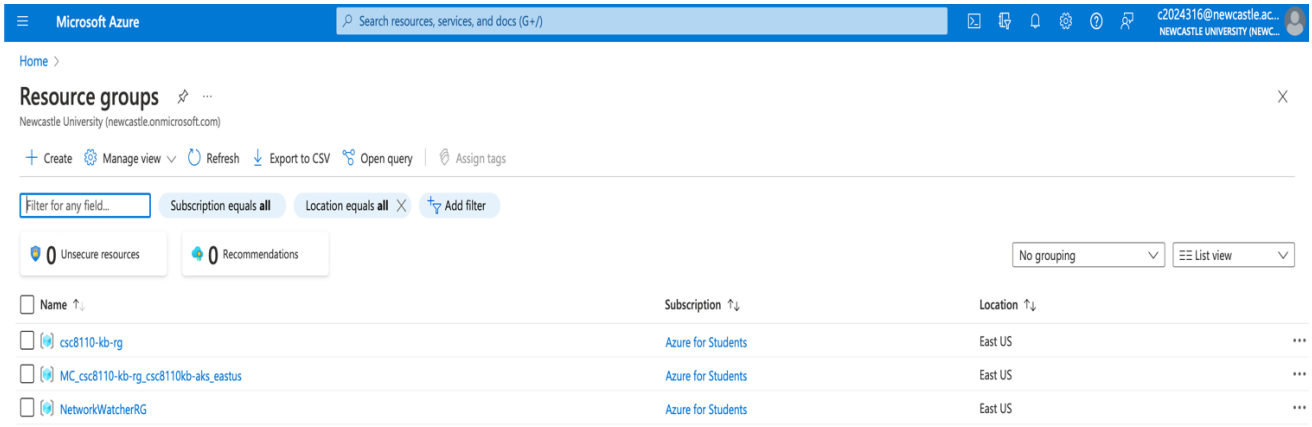


Figure 5.20

To view my course-work demonstration video click on the below zoom link

"https://newcastleuniversity.zoom.us/rec/share/J3A5tQfW6wJKQHsKSo5kDABemThXRcPiKWCM7RHkMib5uj2K-SdS3CXkvSB3-Lds.8xsYHYLtASNXK8UD"

Passcode: w9cDJS&q