

Documentation AWS-Docker Assignment

Sahiru Galappaththi

Contents

Section 1: Introduction	2
Section 2: How I approached the assignment	2
Section 3: Setting up in Cloud environment.	6

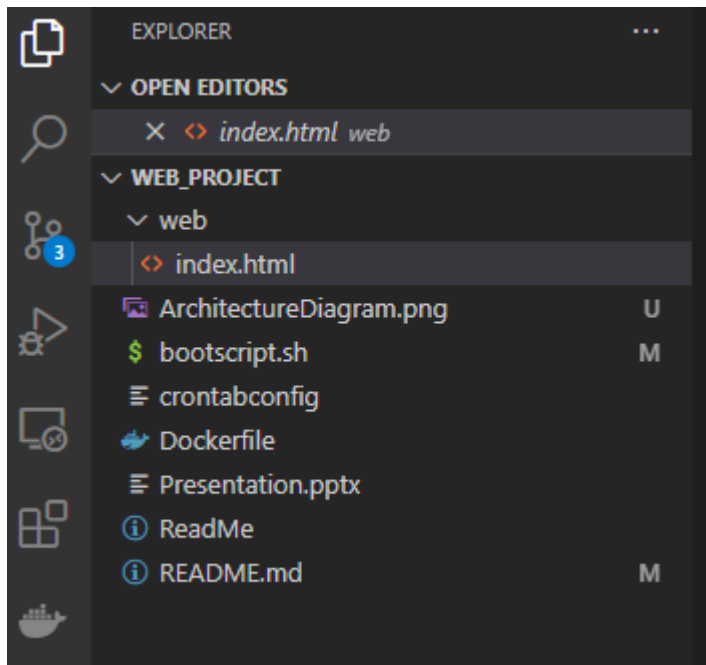
Section 1: Introduction

Docker is used to create a containerized environment to host multiple web servers. NGINX is the type of webserver in use. Dockerfile was added into the project directory which holds the vital steps. These steps will be explained in detail in the following sections.

A bootstrap script is attached to the Docker image, this script will retrieve the time from <http://worldtimeapi.org/> and check with the local server time. These times will dynamically update the homepage as well.

Section 2: How I approached the assignment

Use Visual studio code to manipulate and edit the files as it's easier and gives a better view of the project structure.



Create the index.html file

```
<> index.html X
web > <> index.html > ...
 1  <!doctype html>
 2  <html lang="en">
 3  <head>
 4    <meta charset="utf-8">
 5    <title>Web project Sahiru</title>
 6  </head>
 7  <body>
 8    <h1>Nginx container active</h1>
 9    <h3>Fetched time: 00:00:00</h3>
10    <h3>Local time: 00:00:00</h3>
11  </body>
12  </html>
```

Create the bootstrap script

```
#!/bin/bash

TIMEZONE=`cat /etc/timezone`
#Printing out server timezone
echo "This is the server timezone - $TIMEZONE"

#Reading the time from worldtime API for server time zone
WORLD_TIME_LINK="http://worldtimeapi.org/api/timezone/$TIMEZONE"
WORLD_TIME=$(curl "$WORLD_TIME_LINK" | jq -r '.unixtime')
LOCAL_TIME=$(date +%s)
echo "Fetched UNIX Time: $WORLD_TIME"
echo "Server UNIX Time: $LOCAL_TIME"

#Check if local and world times are equal or not
if [ "$WORLD_TIME" = "$LOCAL_TIME" ]; then
    echo "Fetched Time and Server Time are equal."
else
    echo "Fetched Time and Server Time are not equal."
fi
```

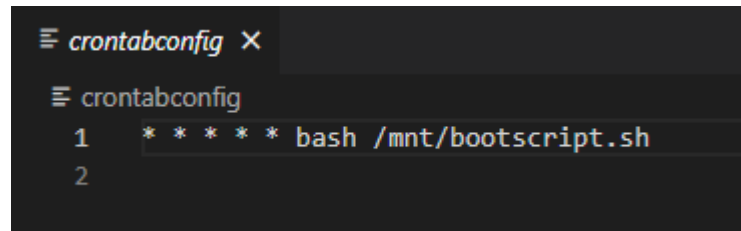
```

#Update index.html with world time and local time
DISPLAY_WORLD_TIME=$(printf '(%F %T)T\n' $WORLD_TIME)
DISPLAY_LOCAL_TIME=$(printf '(%F %T)T\n' $LOCAL_TIME)
sed -i "/Fetched time/c\Fetched time: '$DISPLAY_WORLD_TIME'"
/usr/share/nginx/html/index.html
sed -i "/Local time/c\Local time: '$DISPLAY_LOCAL_TIME'"
/usr/share/nginx/html/index.html

#Checking response code of the server
STATUS_CODE=$(curl --write-out "%{http_code}\n" "http://localhost:80" --output
output.txt --silent)
echo "Response Code from web server: $STATUS_CODE"
if [ "$STATUS_CODE" = "200" ]; then
    echo "Web server is responding the status code 200"
    echo "Server is responding as expected. " | mail -v -s "Server status report"
rowsahiru@gmail.com
else
    echo "Web server is not responding the status code 200"
    echo "Server not returning code 200, please take a look. " | mail -v -s
"Server status report" rowsahiru@gmail.com
fi

```

Create the Crontab config file and set it to run the bootstrap script every 1 minute



The screenshot shows a crontab editor window titled 'crontabconfig'. The editor contains a single line of configuration: '1 * * * * bash /mnt/bootscrip.sh'. The line number '1' is on the left, and the configuration text is on the right. Below this line, the number '2' is visible, indicating the next line in the file.

```

crontabconfig
1  * * * * * bash /mnt/bootscrip.sh
2

```

Create the Dockerfile in the same directory,

```
Dockerfile X
Dockerfile > FROM
1 FROM nginx:latest
2
3 COPY ./web/index.html /usr/share/nginx/html/index.html
4
5 # attach the bootstrap script to the docker image
6 ADD bootscrip.sh /mnt/bootscrip.sh
7
8 # Give execution rights on the cron scripts
9 RUN chmod 777 /mnt/bootscrip.sh
10
11 #updating packages
12 RUN apt-get update
13 #Installing Cron and JQ
14 RUN apt-get -y install cron && apt-get -y install jq
15
16 ADD crontabconfig /mnt/crontabconfig
17
18 RUN crontab /mnt/crontabconfig
19
20 CMD [ "sh", "-c", "cron && nginx -g 'daemon off;'" ]
```

This file contains the necessary steps for our image to run.

Next step is to push these files to Github.

main 1 branch 0 tags

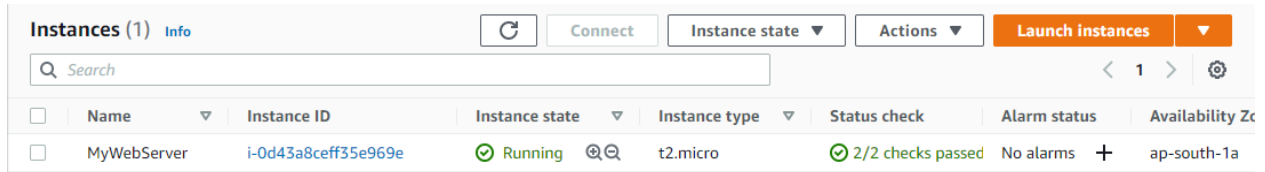
Go to file Add file Code

SahiruGala added email send option when server is responsive 334253e 2 days ago 4 commits

web	initial commit with resource files	2 days ago
Dockerfile	initial commit with resource files	2 days ago
Presentation.pptx	initial commit with resource files	2 days ago
README.md	Initial commit	2 days ago
ReadMe	initial commit with resource files	2 days ago
bootscrip.sh	added email send option when server is responsive	2 days ago
crontabconfig	initial commit with resource files	2 days ago

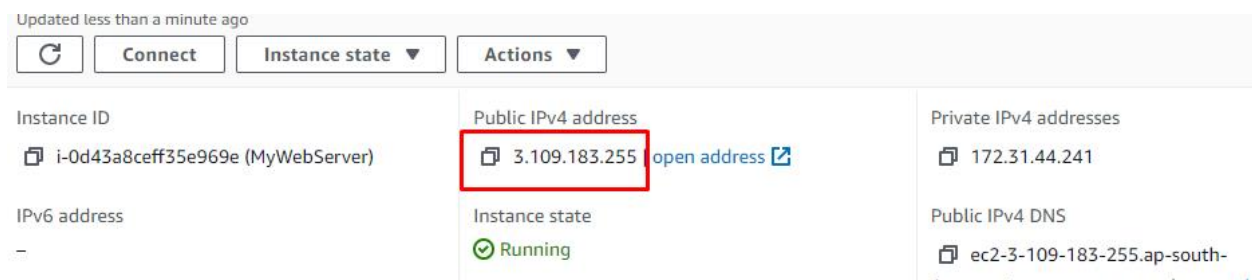
Section 3: Setting up in Cloud environment.

Login into amazon AWS and create an EC2 instance.



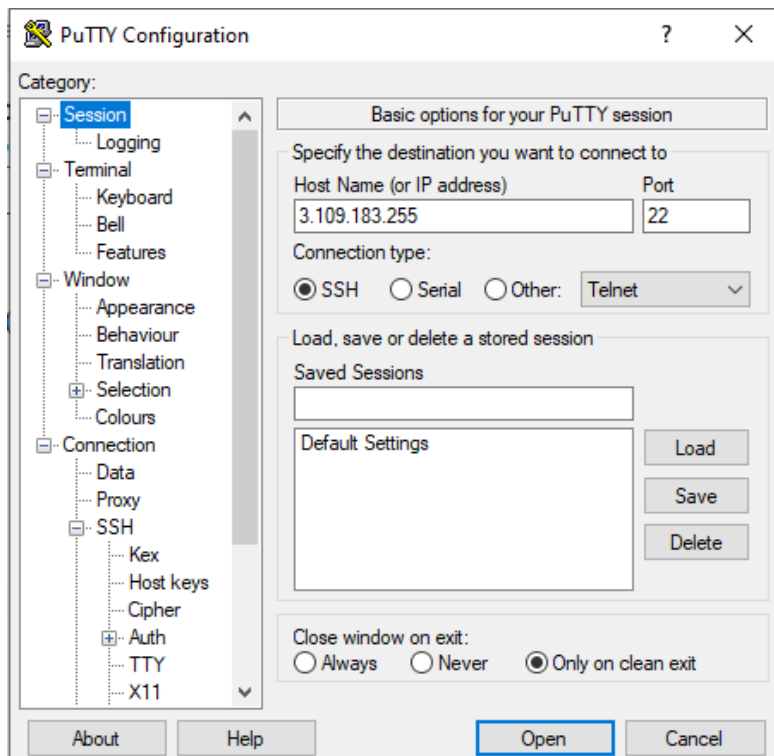
	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
<input type="checkbox"/>	MyWebServer	i-0d43a8ceff35e969e	Running	t2.micro	2/2 checks passed	No alarms	ap-south-1a

Use the following IP address to login to the instance.

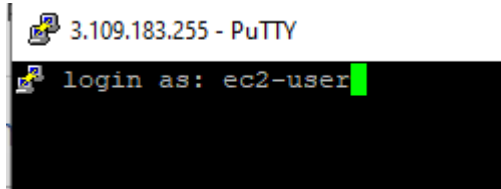


Updated less than a minute ago	Connect	Instance state	Actions
Instance ID	Public IPv4 address		Private IPv4 addresses
i-0d43a8ceff35e969e (MyWebServer)	3.109.183.255 open address		172.31.44.241
IPv6 address	Instance state		Public IPv4 DNS
-	Running		ec2-3-109-183-255.ap-south-

Using an app like PuTTY will be helpful.



Login as ec2-user



Clone the repo which I pushed to earlier,

```
[ec2-user@ip-172-31-44-241 ~]$ pwd
/home/ec2-user
[ec2-user@ip-172-31-44-241 ~]$ git clone https://github.com/SahiruGala/Web_Project.git
Cloning into 'Web_Project'...
remote: Enumerating objects: 18, done.
remote: Counting objects: 100% (18/18), done.
remote: Compressing objects: 100% (11/11), done.
remote: Total 18 (delta 4), reused 13 (delta 2), pack-reused 0
Receiving objects: 100% (18/18), 833.58 KiB | 4.25 MiB/s, done.
Resolving deltas: 100% (4/4), done.
[ec2-user@ip-172-31-44-241 ~]$
```

Move into the Web_Project folder and give permissions to the files,

```
[ec2-user@ip-172-31-44-241 ~]$ pwd
/home/ec2-user
[ec2-user@ip-172-31-44-241 ~]$ ll
total 4
-rw----- 1 ec2-user ec2-user 758 May  2 19:39 dead.letter
drwxrwxr-x 3 ec2-user ec2-user  25 May  3 05:38 project
drwxrwxr-x 4 ec2-user ec2-user 145 May  4 22:06 Web_Project
[ec2-user@ip-172-31-44-241 ~]$ cd Web_Project/
[ec2-user@ip-172-31-44-241 Web_Project]$ chmod 775 *
[ec2-user@ip-172-31-44-241 Web_Project]$
```

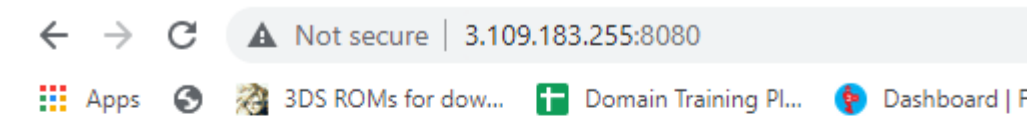
Build Image in aws environment.

```
[ec2-user@ip-172-31-44-241 Web_Project]$ sudo docker build -t webproject .
Sending build context to Docker daemon 1.785MB
Step 1/9 : FROM nginx:latest
----> fa5269854a5e
Step 2/9 : COPY ./web/index.html /usr/share/nginx/html/index.html
----> Using cache
----> e14025dceb9e
Step 3/9 : ADD bootscrip.sh /mnt/bootscrip.sh
----> Using cache
----> e27d83a702ad
Step 4/9 : RUN chmod 777 /mnt/bootscrip.sh
----> Using cache
----> 20b34b0bb974
Step 5/9 : RUN apt-get update
----> Using cache
----> 0b79d2c6c435
Step 6/9 : RUN apt-get -y install cron && apt-get -y install jq
----> Using cache
----> f57ed7e4ee99
Step 7/9 : ADD crontabconfig /mnt/crontabconfig
----> Using cache
----> 29f4102cldde
Step 8/9 : RUN crontab /mnt/crontabconfig
----> Using cache
----> bee5fb325c2b
Step 9/9 : CMD [ "sh", "-c", "cron && nginx -g 'daemon off;'" ]
----> Using cache
----> 86e6906363c2
Successfully built 86e6906363c2
Successfully tagged webproject:latest
[ec2-user@ip-172-31-44-241 Web_Project]$
```

Run the docker image 3 times on different ports,

```
[ec2-user@ip-172-31-44-241 Web_Project]$ sudo docker run -it --rm -d -p 8080:80 --name website webproject
9af7bccf0cbb15b2led53b0d240801dclc97663ae7bcbbfa76ee9d0bc6926803
[ec2-user@ip-172-31-44-241 Web_Project]$ sudo docker run -it --rm -d -p 8081:80 --name website2 webproject
3b01d51c8b40a202d0cdfb37f3715da2a449252bf91feccl20a9add590287b7
[ec2-user@ip-172-31-44-241 Web_Project]$ sudo docker run -it --rm -d -p 8082:80 --name website3 webproject
0688c6f12319e7efa700a29304e8f940d73d54b3e6146048480b272d2000624c
[ec2-user@ip-172-31-44-241 Web_Project]$
```

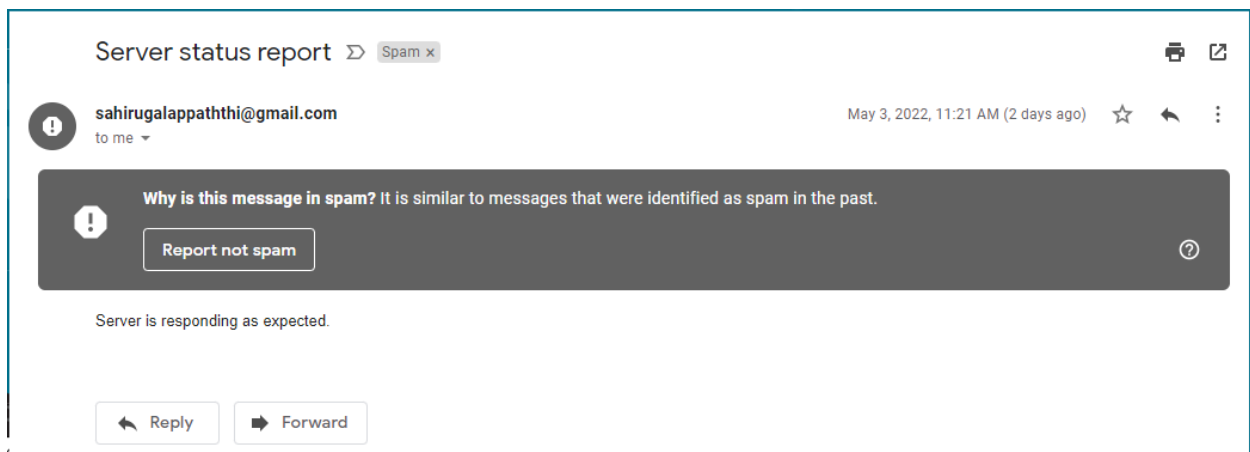

To validate, retrieve the IP address again from EC2 instance and add port at the end,



Nginx container active

Fetch time: '2022-05-04 22:18:02' Local time: '2022-05-04 22:18:02'

An email will be sent to the support team after checking the http response code.
If its 200, then it is responding as expected. If not 200, it is not responding as expected.



#Checking response code of the server

```
STATUS_CODE=$(sudo curl -o /dev/null -s -w "%{http_code}\n" http://localhost:8080)
```

Curl command will return the http status code, and since its assigned to the STATUS_CODE variable, it will not store the returned status code from the curl command.

After that I used this variable and passed it into an IF condition to assign two different tasks when status code is 200 and not 200.

