Develop and Deploy Web Application in Container

(LAB-M11-01)

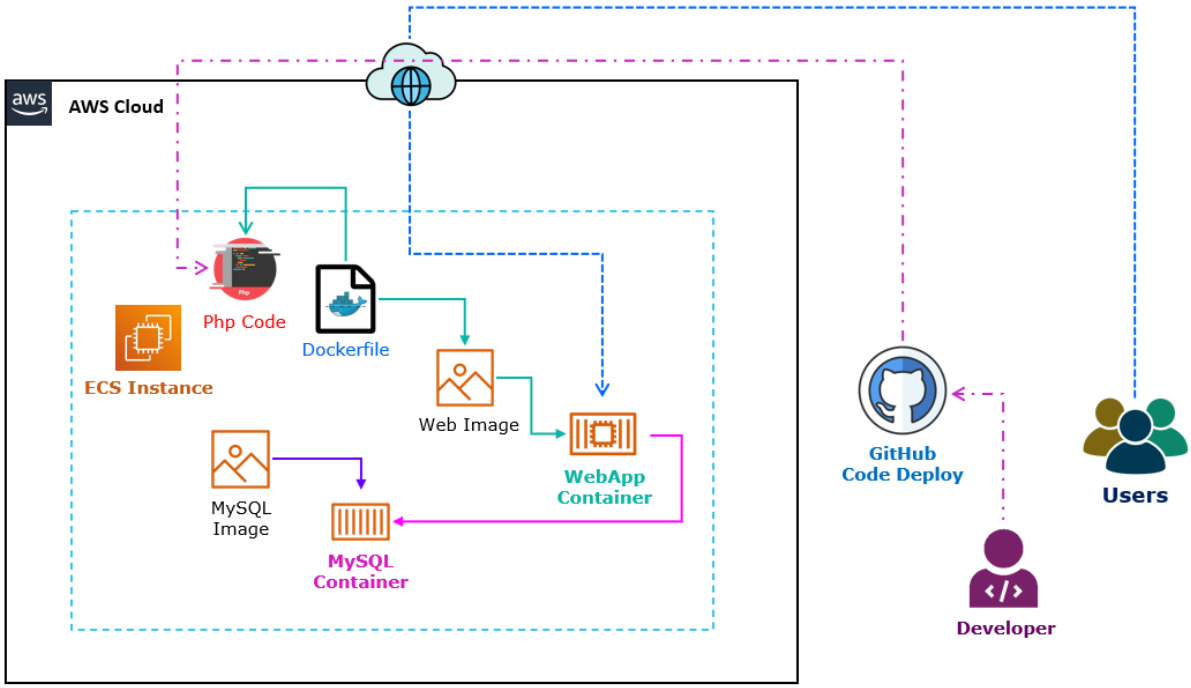
Lab scenario

You're preparing to host a web application in Container. You need to explore how to set up the web application in ECS Instance.

Objectives

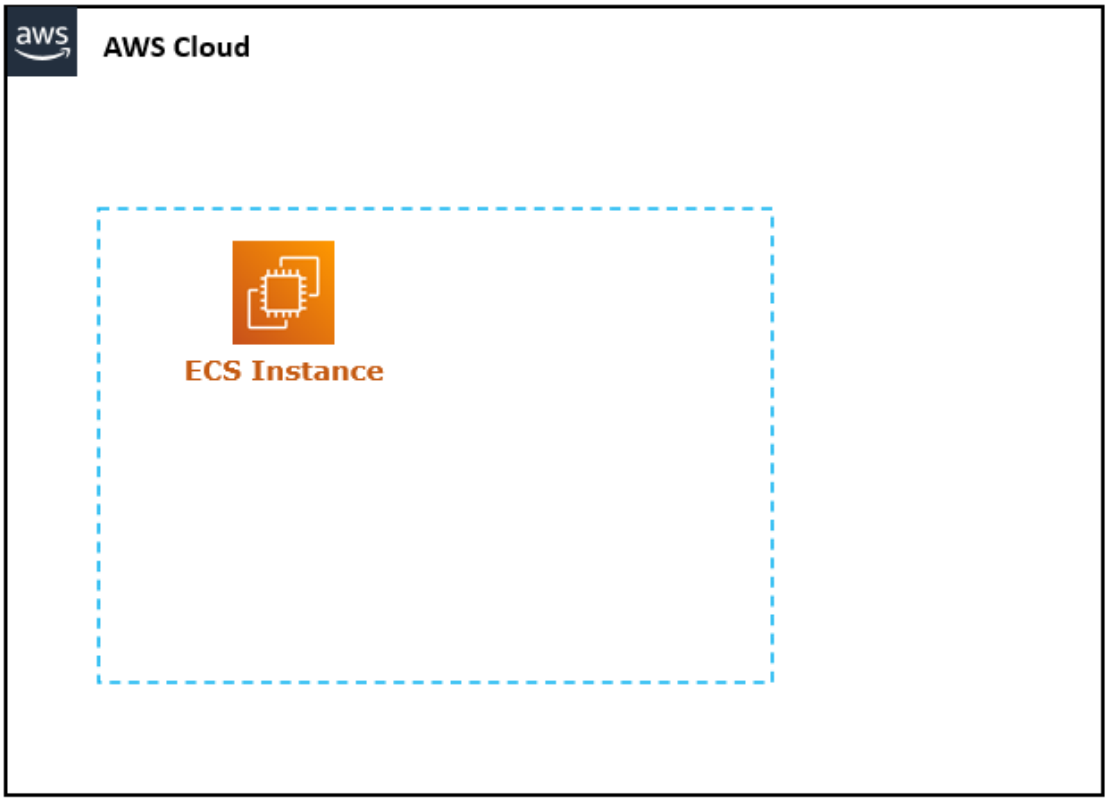
After you complete this lab, you will be able to:

- Create ECS Instance.
- Create Container Image for MySQL.
- Create Docker Container using MySQL Image.
- Create database and table in MySQL Container.
- Create Container Image for Web application.
- Create Docker Container using Web application.
- Access Web application.



Task 1: Create ECS Instance

In this task, you will launch an Amazon ECS optimised instance using the management console. The instance will be used to deploy the Docker Web Application and Database.



Step 1: Create EC2 Instance

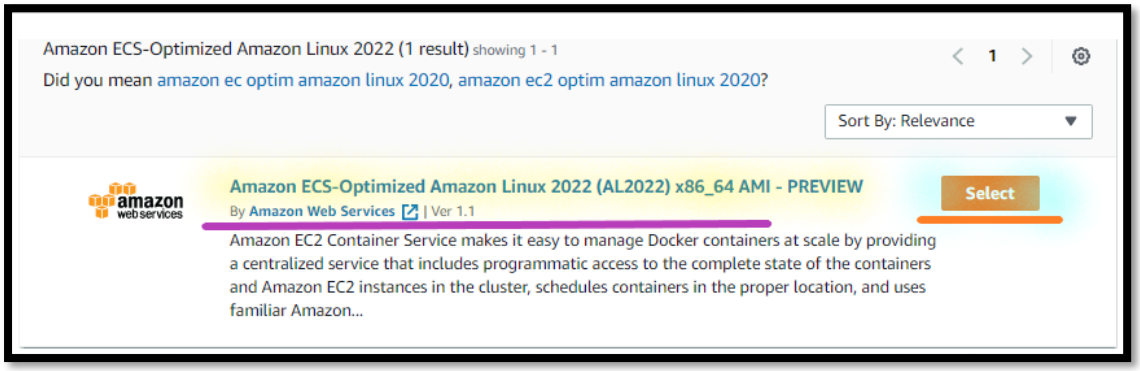
1. In the **AWS Management Console**, on the **Services** menu Search and Select **EC2**.
2. Choose the **US East (N. Virginia)** region list to the right of your account information on the navigation bar.
3. Select **Instances**.
4. Select **Launch Instances**.
    a. In the **Name and tags** section:
        i. **Name**: Write **ECS-Docker**.
    b. In the **Application and OS Images** section:
        i. In the **Search box**:
            a) Type **Amazon ECS-Optimized Amazon Linux 2022**.

b) Press **Enter** ***key***.

Note: You can see the **Choose an Amazon Machine Image** page.

c) **From** the **Choose an Amazon Machine Image** page:

1) Select **Amazon ECS-Optimized Amazon Linux 2022**.



2) Select **Continue**.

Note: You can see the **Launch an Instance** page.

c. In the **Instance Type** section:

i. **Instance type**: Dropdown and in the **Search box**:

a) Type **t2.micro**.

b) Select **t2.micro**.

d. In the **Key pair (login)** section:

i. **Key pair name**: Dropdown and select **My-Dev-LAB-KP**.

e. In the **Network setting** section:

i. Select **Edit**.

a) **Firewall**: Select **Create security group**.

1) **Security group name**: Write **Docker-Server-SG**.

2) **Description**: Write **Docker Server Group**.

3) **Inbound security groups rules**:

I. In the **Security group rule 1**:

1) **Type**: Dropdown and select **SSH**.

2) **Source type**: Dropdown and select **Anywhere**.

II. Select **Add Security group rule**.

III. In the **Security group rule 2**:

1) **Type**: Dropdown and select **HTTP**.

2) **Source type**: Dropdown and select **Anywhere**.

Note: Leave the other details as default.

f. In the **Summary** section:

i. Select **Launch Instances**.

Note: **Wait**, till you can see the **message** "**Successfully initiated launch of instance**".

g. Select **View all instances**

Note: **Wait**, till you can see the **ECS-Docker** Instance **State** is **Running**.

Note: **Wait**, till you can see the **ECS-Docker** Instance **Status check** is **2/2 check passed**.

Step 4: Copy the ECS-Docker Server Public IP address

5. In the **AWS Management Console**, on the **Services** menu, click **EC2**.
6. Click **Instances**.
7. Select **ECS-Docker**.

i. Go below and click on **Details**.

ii. Expand **Instance summary**.

iii. **Copy** the **Public IP address**.

Step 5: Connect to ECS-Docker Server

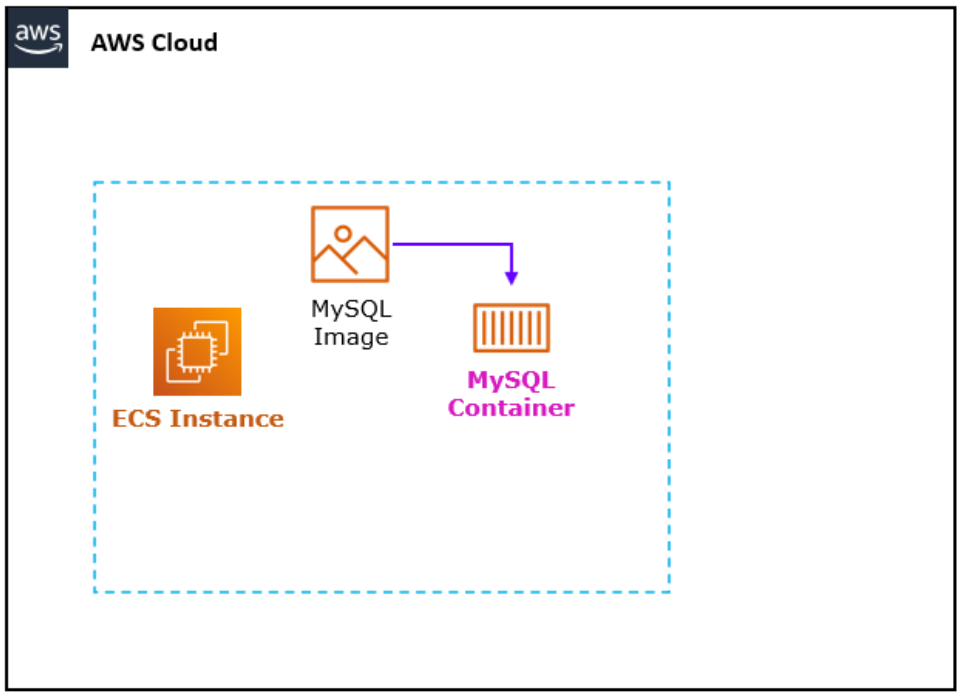
8. **Connect** to **ECS-Docker** using **MobaXterm**.

   a. **Navigate** and Select the **My-Dev-LAB-KP**.ppk file that you generated in previous step.

   b. **Login as**: Write username **ec2-user**.

Task 2: Create DB Container

In this task, you will create MySQL Database Container with database and table.

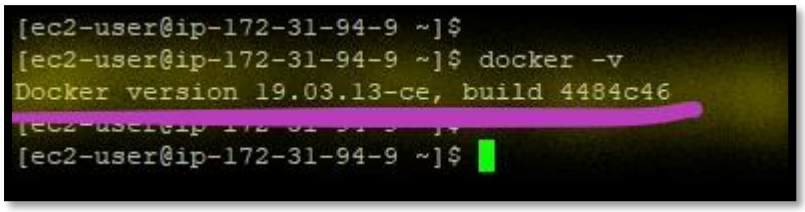


Step 1: Verify the Docker Version

9. From the ECS-Docker **terminal console**, **execute** the following command, to **verify** the **Docker installed version**:

```
docker -v
```

Note: In the **Output**, you can see the **Docker version**.



Step 2: Create MySQL Container

10. **From** the ECS-Docker **terminal console**:

    a. **Execute** the following command, to **download** the **MySQL 5.6 Image**:

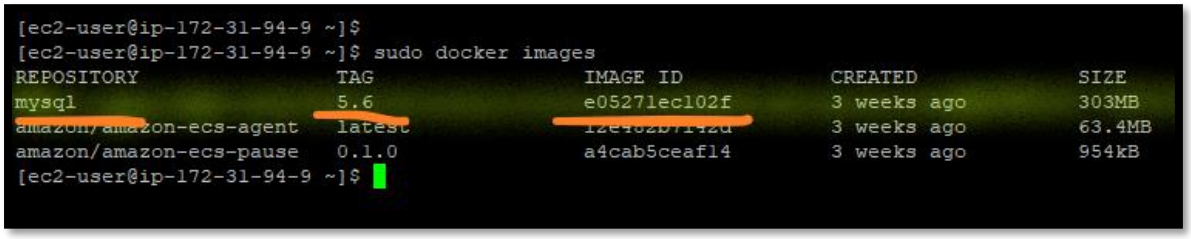
```
sudo docker pull mysql:5.6
```

Note: In the **Output**, you can see the **Docker image** getting **downloaded**.

    b. **Execute** the following command, to **verify** the **Docker image**:

```
sudo docker images
```

Note: In the **Output**, you can see the **mysql:5.6 Docker images**.



    c. **Execute** the following command, to **create** the **MySQL Docker Container**:

```
sudo docker run --name db -p 3306 -v /mysql-data:/var/lib/mysql -e
MYSQL_ROOT_HOST='%' -e MYSQL_ROOT_PASSWORD=password -d mysql:5.6
```

Note: Following is the options used to create MySQL docker container.:

1. **--name db** – To specify container name.
2. **-v /mysql-data:/var/lib/mysql** - It mounts the relative path of /mysql-data from the host to the path /var/lib/mysql in the container.
3. **-e MYSQL_ROOT_HOST='%'** – To specify host to access MySQL DB for root user. '%' is to allow to access from any other containers.
4. **-e MYSQL_ROOT_PASSWORD=password** – To set root user password.

Note: In the **Output**, you can see the **STDOUT**.

d. **Execute** the following command, to **view** the **Container Status**:

```
sudo docker ps -a
```

Note: In the **Output**, you can see the container name as **db** and status as **up**.

Note: **Copy** the **db container Container ID** in the **Notepad**.

e. **Execute** the following command, to **get details** of the **DB Container**:

```
sudo docker inspect <DB-CONTAINER-ID>
```

Note: **Replace** the **DB-CONTAINER-ID** with the **DB Container ID** which you have copied in the previous step.

Note: **Copy** the **Private IP address** of the **db container** in the **Notepad**. Scroll below in the Docker Console to view the details.

Note: **Copy** the **db container Host Port** in the **Notepad**.

f. **Execute** the following command, to **install** the **MySQL Client**:

```
sudo yum install -y mysql
```

g. **Execute** the following command, to **connect** to the **DB Container**:

```
sudo mysql -P <DB-HOSTPORT-NO> --protocol=tcp -u root -p
```

Note: **Replace** the **DB-HOSTPORT-NO** with the **DB Host Port No.** which you have copied in the previous step.

Note: **Remove** the starting and end **<>** brackets.

i. When you get **prompt** to enter the **password**, write **password**.

Note: You can see the **MySQL Console**.

11. **From** the **MySQL terminal console**:

    a. **Execute** the following command, to **create database**, **prod_schema**:

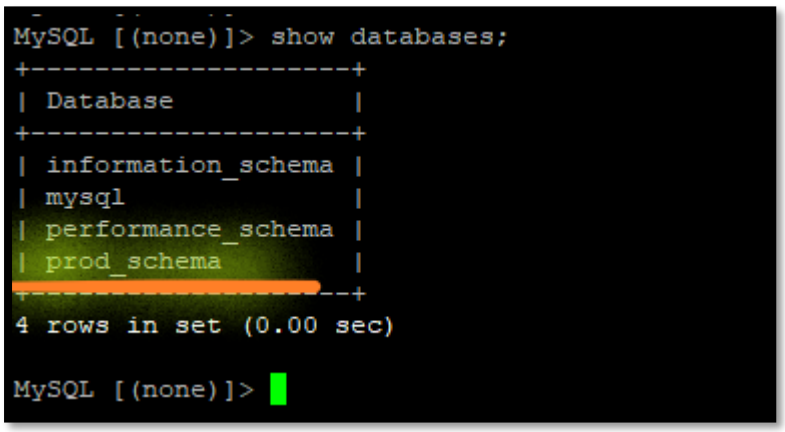
```
create database prod_schema;
```

Note: In the Output you can see "**Query OK, 0 rows affected**" message.

    b. **Execute** the following command, to **show databases**:

```
show databases;
```

Note: In the database, you can see the **prod_schema** database.



c. **Execute** the following command, to **use** the **prod_schema** database as the **default**:

```
use prod_schema;
```

Note: In the output, should show "**database changed**" message.

d. **Execute** the following command, to **create table**, **products** with names of the **columns** and **datatypes**:

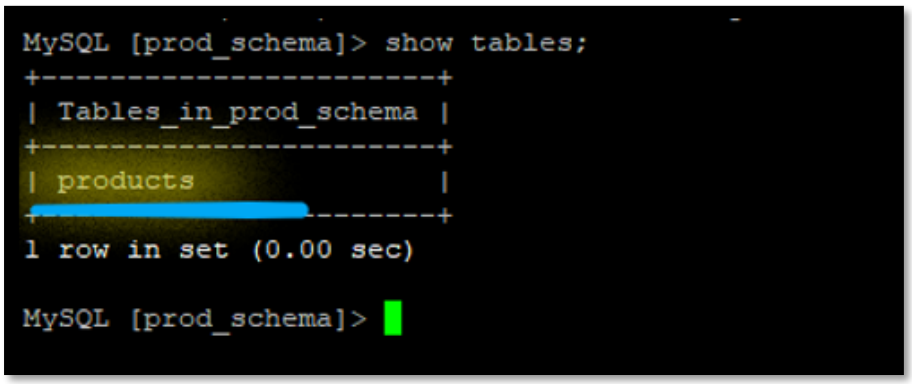
```
create table products (id int NOT NULL AUTO_INCREMENT, name varchar(255), quantity varchar(255), price varchar(255), PRIMARY KEY (id));
```

Note: In the Output you can see "**Query OK, 0 rows affected**" message.

e. **Execute** the following command, to **show tables**:

```
show tables;
```

Note: In the tables, you can see the **products** table.



f. **Execute** the following command, to **exit mysql**:

```
exit
```

Note: You can now see the **linux prompt**.

Task 3: Create WebApp container

In this task, you will create Container with Php application.

Step 1: Develop the Code for WebApp container

12. **Unzip** the **LAB-11-01-code.zip** (**Php code)**.

Note: **Lab-11-01-code.zip** code file is available with the Lab manual.

13. **Open** the **data.php** in the **Notepad**.
14. **Update** the ***database*** details in the code:
    a. **Replace** the **TO DO 1** with the ***db-container*** **Private IP address**, which you have copied in the previous step.

Note: **Don't remove** the starting and end quote (' ')and semicolon (;).

    b. **Replace** the **TO DO 2** with the database instance ***username*** as **root**.
    c. **Replace** the **TO DO 3** with the database instance ***password*** as **password**.
    d. **Replace** the **TO DO 4** with the ***database*** name **prod_schema**.
    e. **Replace** the **TO DO 5** with the ***prod_schema database table*** name **products**.

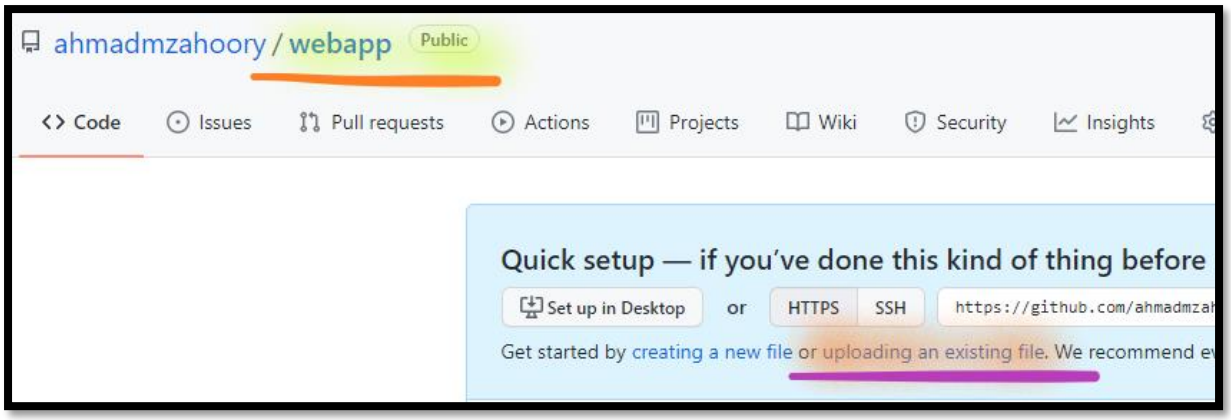
15. Select **File**.
16. Select **Save**.

Step 2: Create GitHub Repository

17. **Login** into your GitHub account.
18. **To create repository**, go to right top side and select **+** sign.
    a. Select **New repository** and **configure**:
        i. **Repository name**: Write **webapp**.
        ii. Select **Public**.
        iii. Select **Create repository**.

Note: Once repository created, **webapp repsoitory** page gets opened.

19. **From** the **webapp** **repository:**
    a. Select **uploading an existing file**.

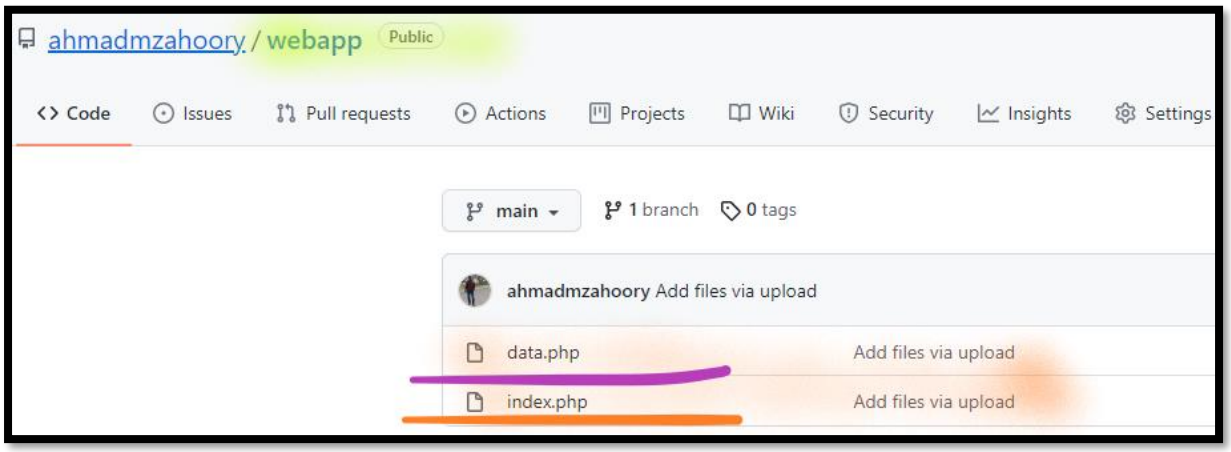


b. Select **Choose your files**.

c. Select **index.php** and **data.php** *(which you have updated in the last step)*.

d. Select **Commit Changes**.

Note: Once code **uploaded successfully**, you can see them in repository.

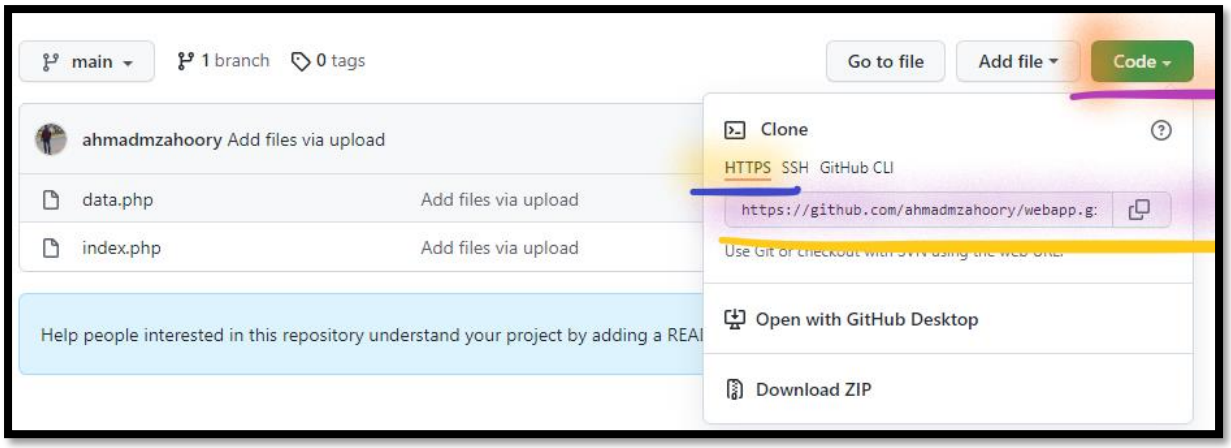


20. **From** the **webapp** **repository:**

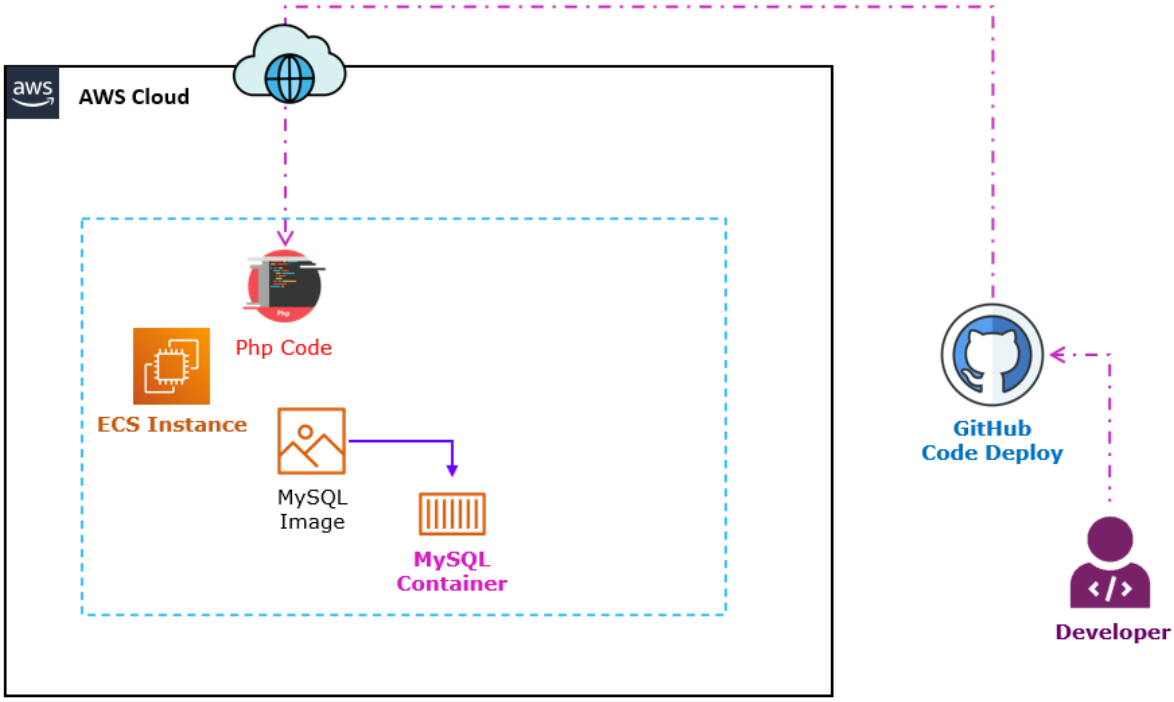
a. Select **Code**.

b. Select **HTTPS**.

c. **Copy** the **Clone URL** in **Notepad**.



Step 3: Clone the Code files



21. **Return** to the **ECS-Docker**.

22. **From** the ECS-Docker **terminal console**:

    a. **Execute** the following command, to **install** the **Git**:

```
sudo yum install -y git
```

    b. **Execute** the following command, to **clone** the **Git Repository**:

```
sudo git clone CLONE-WEB-URL
```

Note: **Replace** the **CLONE-WEB-URL** with the Github URL you have copied in the previous step.

c. **Execute** the following command, to **list** the **files and folders**:

```
ls -l
```

Note: You can see the **webapp** folder.

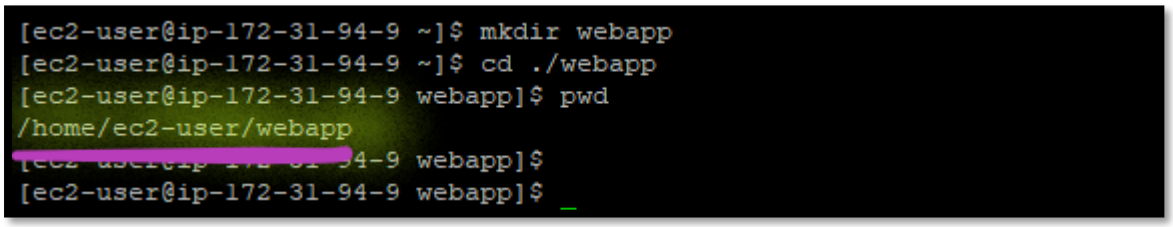
d. **Execute** the following command, to **change directory** to **webapp**:

```
cd ./webapp
```

e. **Execute** the following command, to **verify** the **current path**:

```
pwd
```

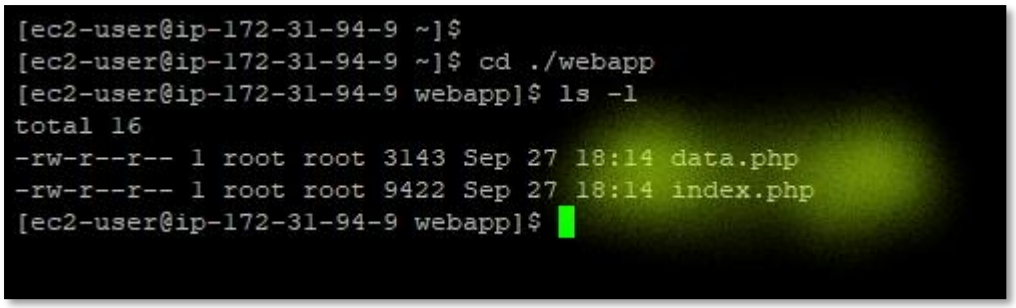
Note: In the **Output**, you can see the **/home/ec2-user/webapp** path.



f. **Execute** the following command, to **list** the **files and folders**:

```
ls -l
```

Note: In the **Output**, you can see the **index.php** and **data.php** file.



g. **Execute** the following command, to **install** the **nano**:

```
sudo yum install -y nano
```

h. **Execute** the following command, to **verify** the **index.php content**:

```
sudo nano index.php
```

Note: In the **Output**, you can see the **index.php content**.

Note: Press **CTRL + X** to **exit** the **nano editor**.

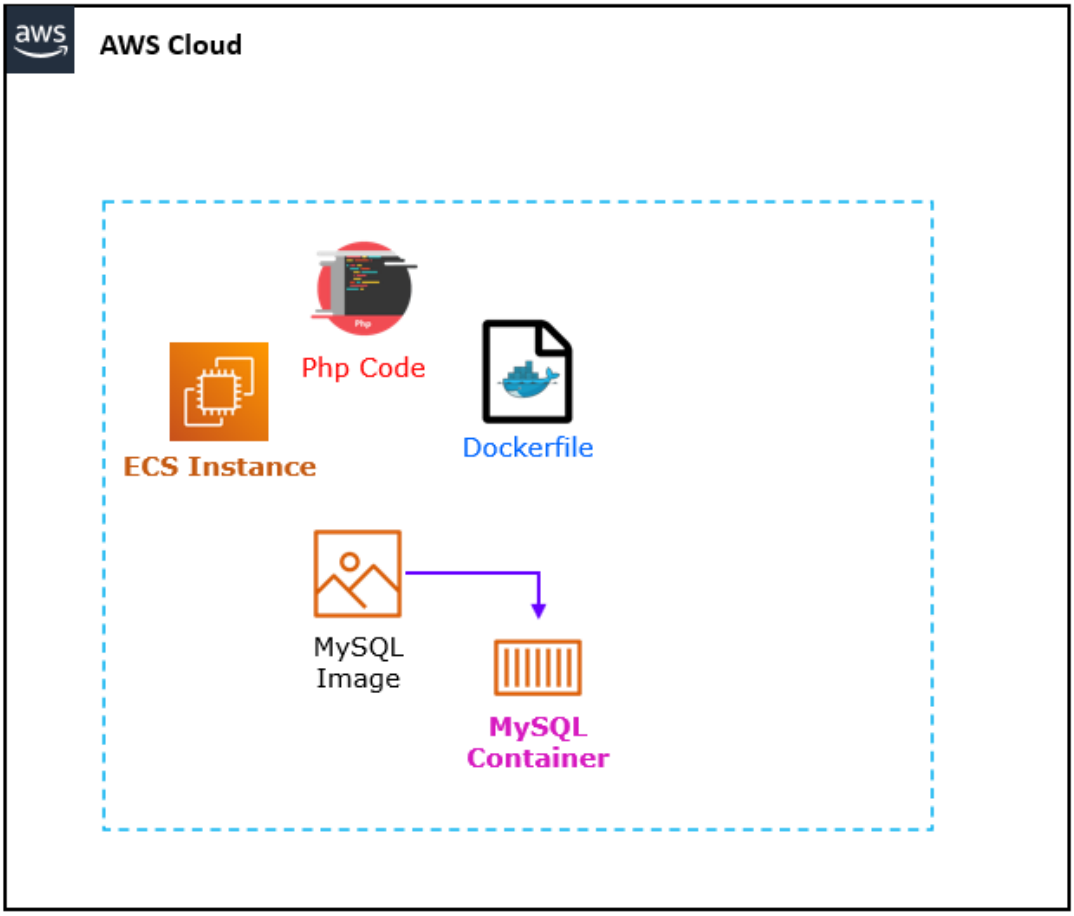
i. **Execute** the following command, to **verify** the **data.php content**:

```
sudo nano data.php
```

Note: In the **Output**, you can see the **data.php content**.

Note: Press **CTRL + X** to **exit** the **nano editor**.

Step 4: Create Dockerfile



23. **From** the ECS-Docker **terminal console**:

a. **Execute** the following command, to **change** back to **parent directory**:

```
cd ..
```

b. **Execute** the following command, to **list** the **file and folders**:

```
ls -l
```

Note: In the **Output**, you can see the **webapp** folder.

c. **Execute** the following command, to **verify** the **current path**:

```
pwd
```

Note: In the **Output**, you can see the **/home/ec2-user** path.

d. **Execute** the following command, to **install** the **wget**:

```
sudo yum install -y wget
```

e. **Execute** the following command, to **download** the **Dockerfile**:

```
sudo wget
https://raw.githubusercontent.com/ahmadzahoory/awsdev/main/Dockerfile
```

f. **Execute** the following command, to **list** the **file and folders**:

```
ls -l
```

Note: In the **Output**, you can see the **Dockerfile** and **webapp** folder.

g. **Execute** the following command, to **view** the **Dockerfile content**:

```
sudo nano Dockerfile
```

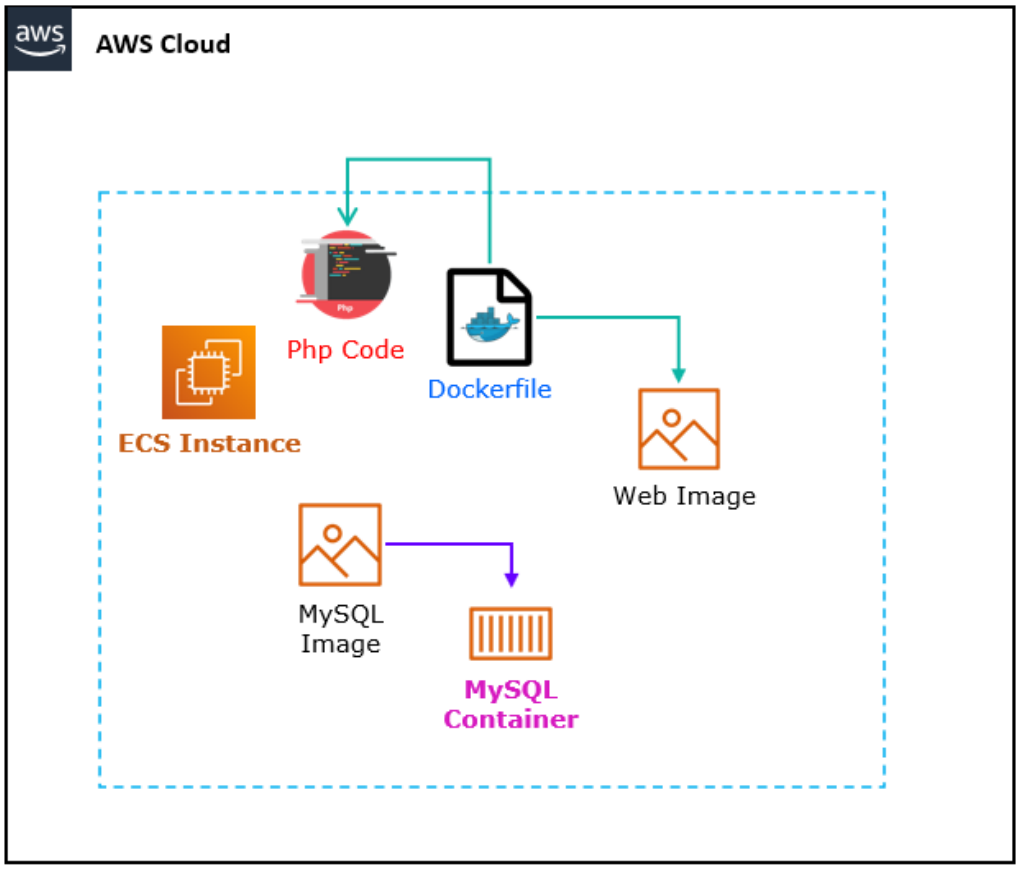
Note: In the **Output**, you can see the **Dockerfile content**.

Note: Your dockerfile does the following:

1. Downloads the apache httpd in conjunction with php from an image repository.
2. Installing php extensions and php extenstion for mysql driver.
3. Copies your web application into the image.

Note: Press **CTRL + X** to **exit** the **nano editor**.

Step 5: Create Docker Image



25. **From** the ECS-Docker **terminal console**,

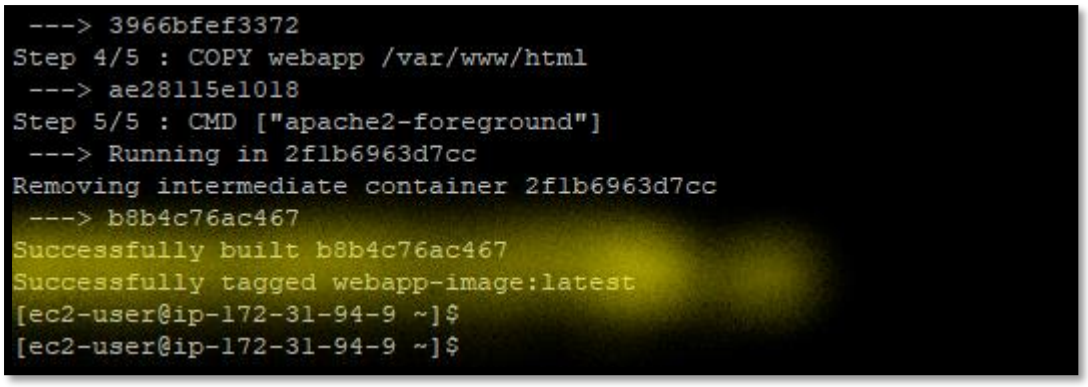
a. **Execute** the following command, to **build** a **Docker image**:

```
sudo docker build -t webapp-image .
```

Note: **Make sure** to **copy** the whole command **including the** '**.**' .

Note: This command builds an image from a Dockerfile located in '.' (the current directory). Then, it will tag the image with a name *webapp*.

Note: In the **Output**, you can see the **Successfully built** and **Successfully tagged message**.

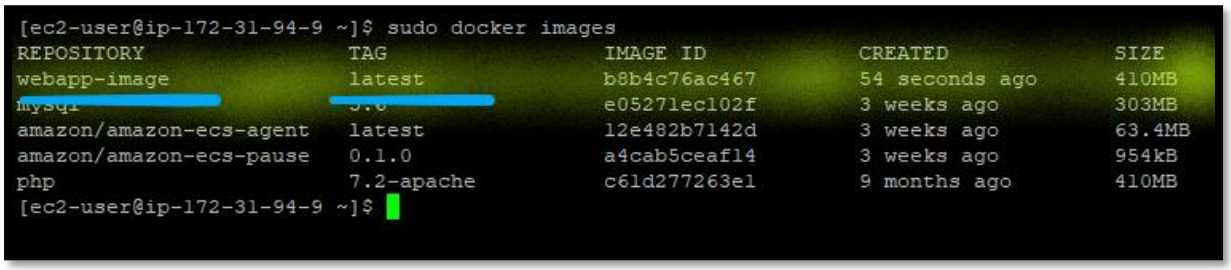


b. **Execute** the following command, to **verify** a **Docker image**:

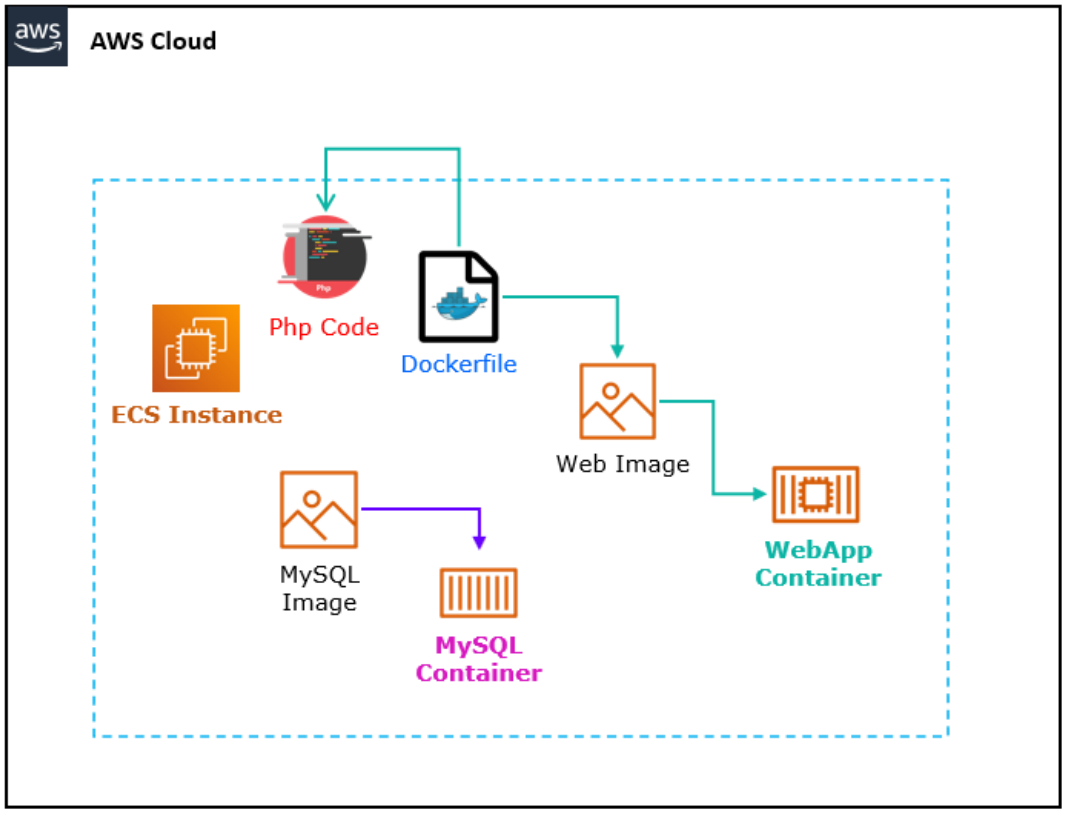
```
sudo docker images
```

Note: In the **Output**, you can see the **webapp-image**.

Note: You should see a couple of docker images listed along with the image you have created in previous step.



Step 6: Run a Docker Container



26. **From** the ECS-Docker **terminal console**:

    a. **Execute** the following command, to **launch** a **container** from the **Docker image** you **build**:

```
sudo docker run --name webapp -d -p 80:80 webapp-image
```

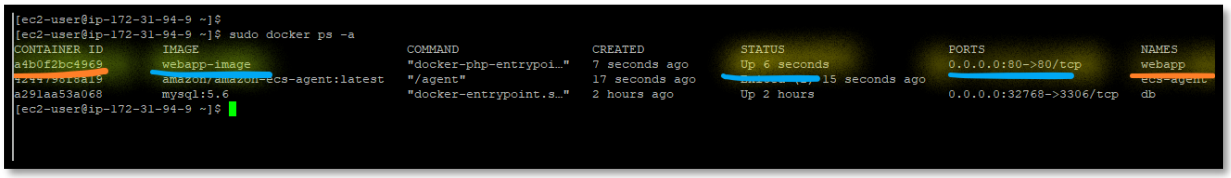
Note: This command requests Docker to run a container, with the name *webapp*, in daemon mode (non-interactive) and map tcp/80 outside the container to tcp/80 on the inside of the container.

Note: In the **Output**, you can see the **STDOUT**.

b. **Execute** the following command, to **view** the **Container Status**:

```
sudo docker ps -a
```

Note: In the **Output**, you can see the container name as **webapp** and status as **up**.



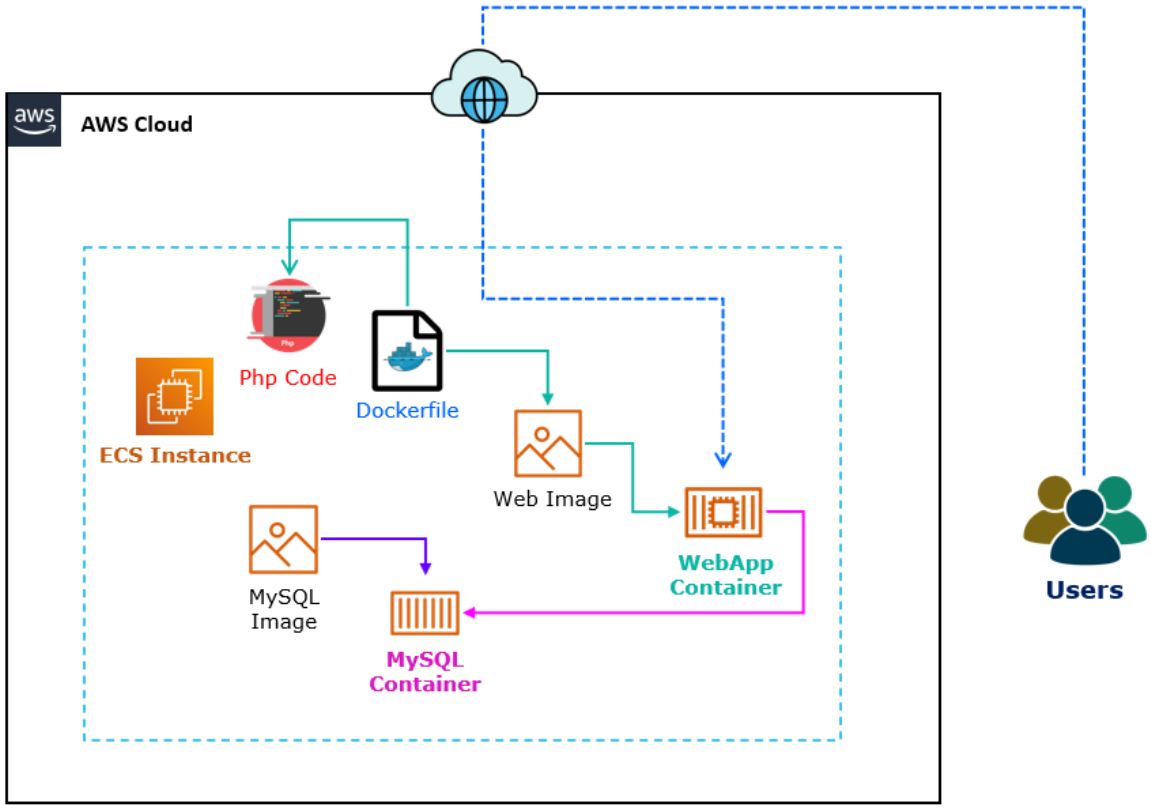
c. **Execute** the following command, to **get details** of the **webapp Container**:

```
sudo docker inspect <WEBAPP-CONTAINER-ID>
```

Note: **Replace** the **WEBAPP-CONTAINER-ID** with the **WebApp Container ID** which you have copied in the previous step.

Note: You can view the **webapp container Host Port** and **Private IP** address.

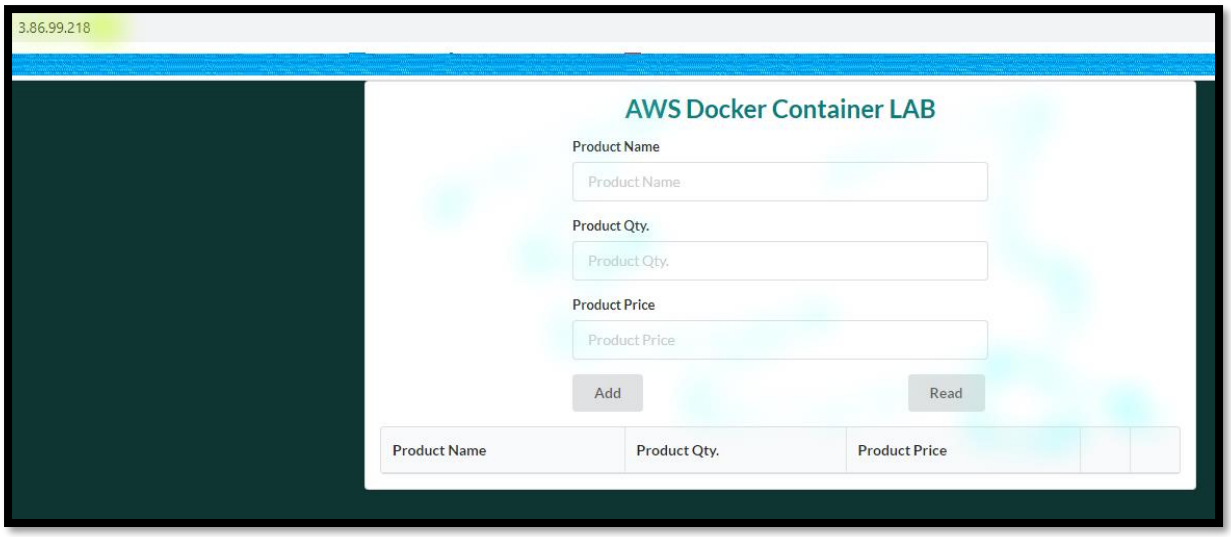
Task 4: Access the WebApp container



Step 1: Access the WebApp hosted in Container

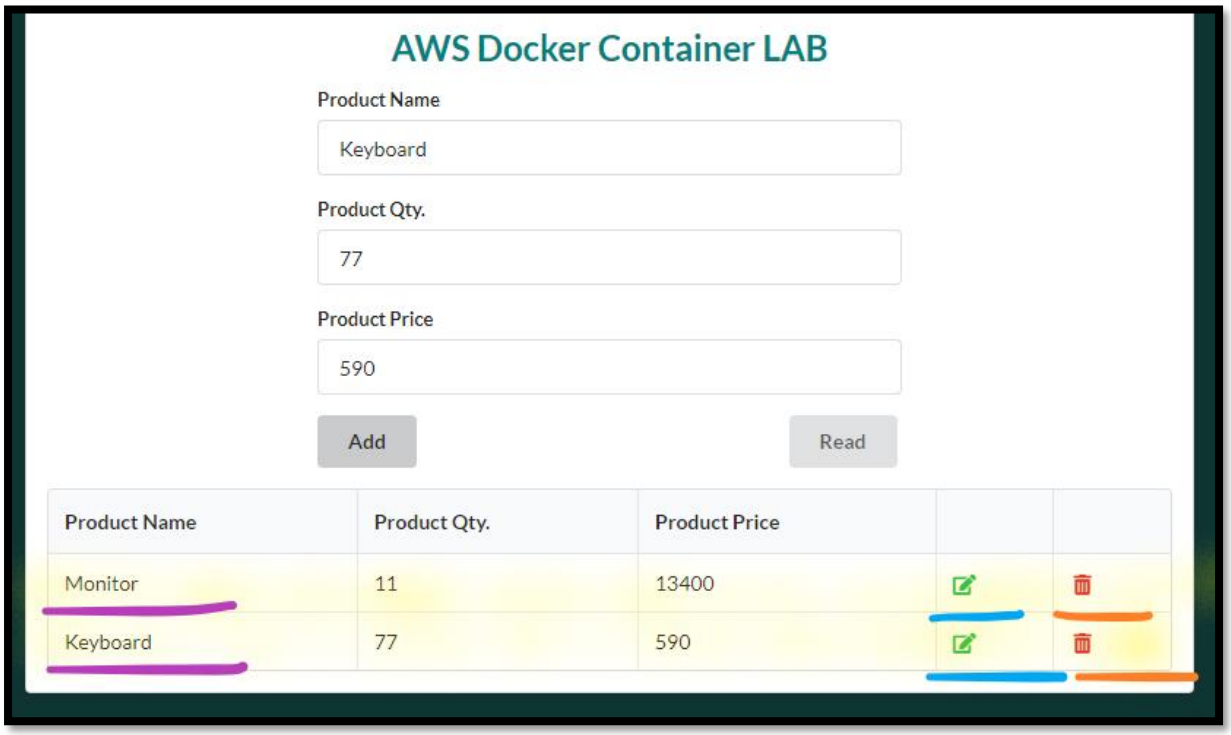
27. **From** your **Local desktop/ laptop**, Open the **Browser** and **Copy** the **Public IP address** (that you noted earlier) to access the website

Note: You can see the **WebApp**.



Step 2: Perform the CRUD Operation

28. **Perform** the **CRUD operation**.
    a. **Add** the ***Product Data***.
    b. You can also **Update** the ***Product Data***.
    c. You can also **Delete** the ***Product Data***.



Note: Go to the next task, but **Don't close** the **WebApp website**.

Step 3: Add Data from MySQL Container

29. **From** the ECS-Docker **terminal console**:

    a. **Execute** the following command, to **connect** to the **DB Container**:

```
mysql -u root -p -h <DB-CONTAINER-PRIVATE-IP-ADDRESS>
```

Note: **Replace** the **DB-CONTAINER-PRIVATE-IP-ADDRESS** with the **DB Container Private IP addres** which you have copied in the previous step.

Note: **Remove** the starting and end **<>** brackets.

    i. When you get **prompt** to enter the **password**, write **password**.

Note: You can see the **MySQL Console**.

30. **From** the **MySQL terminal console**:

    a. **Execute** the following command, to **use** the **prod_schema** database as the **default**:

```
use prod_schema;
```

Note: In the output, should show "**database changed**" message.

    b. **Execute** the following command, to **show tables**:

```
show tables;
```

Note: In the tables, you can see the **products** table.

c. **Execute** the following command, to **show data** from **products table**:

```
select * from products;
```

Note: In the database, you can see the **data added** from **WebApp** Container.

d. **Execute** the following command, to **add data** into **products table**:

```
insert into products (name, quantity, price) VALUES ('Web Camera', '17', '1800');
```

Note: In the Output you can see "**Query OK, 1 row affected**" message.

e. **Execute** the following command, to **exit mysql**:

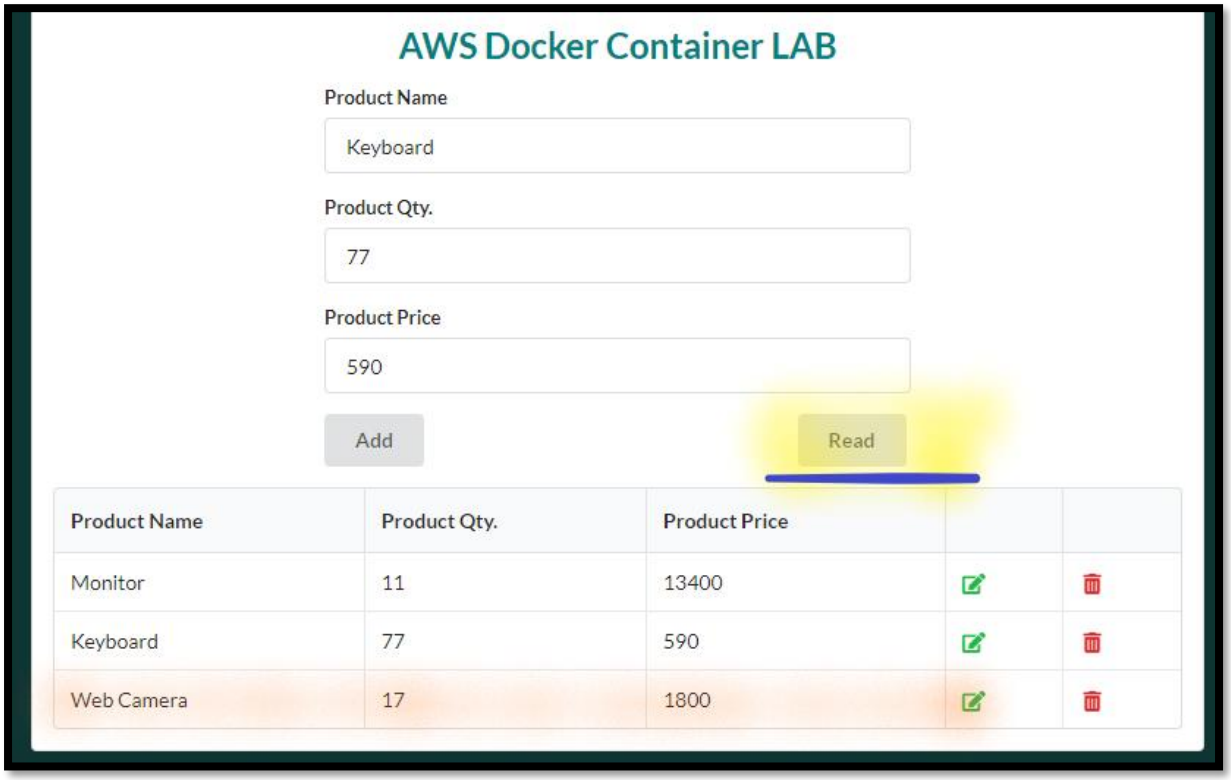
```
exit
```

Note: You can now see the **linux prompt**.

Step 4: Access the WebApp hosted in Container

31. **Return** to the **Browser** (from where you were accessing the WebApp website) and Select **Read**.

Note: You can see the **data added** from **MySQL DB** Container.



Task 5: Delete the Environment

Step 1: Terminate EC2 Instances

18. In the **AWS Management Console**, on the **Services** menu, click **EC2**.
19. Click **Instances**.
20. Select **ECS-Docker** instance.
    a. Click on **Instance state**.
    b. Select **Terminate instance**.
    c. Select **Terminate**.