Develop and Deploy Web Application in Container

(LAB-M11-01)

Version Control	
Document	Develop and Deploy Web Application in Container
Owner	Ahmad Majeed Zahoory
Version	2.1
Last Change	28th May 2024
Description of Change	Tasks steps updated

Lab duration: **60 minutes**

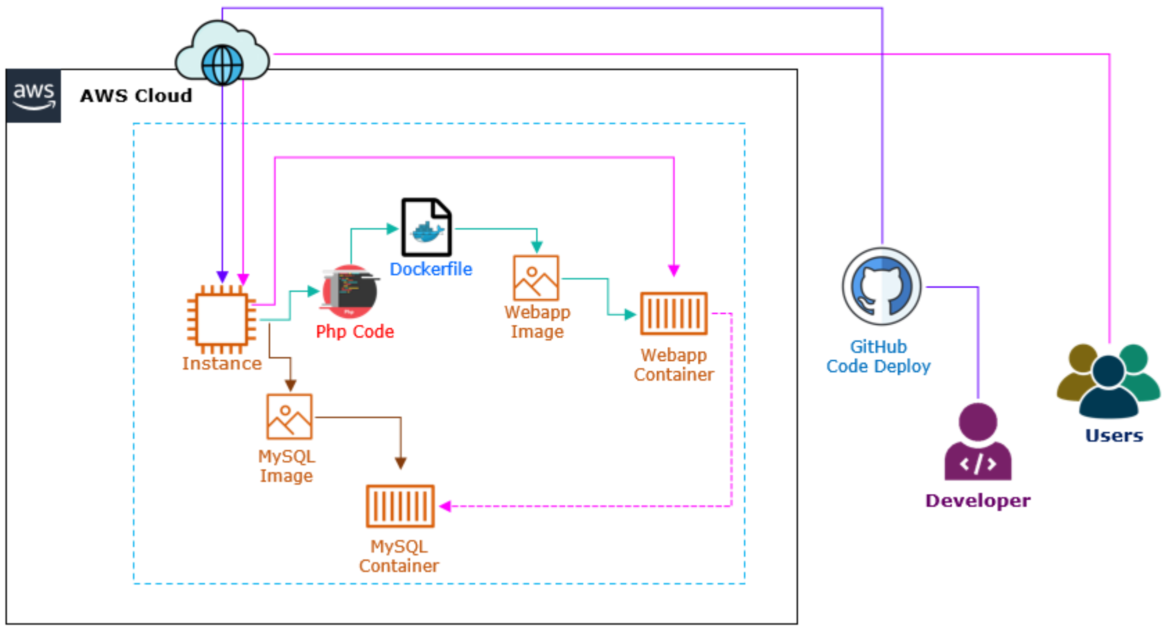
Lab scenario

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

Objectives

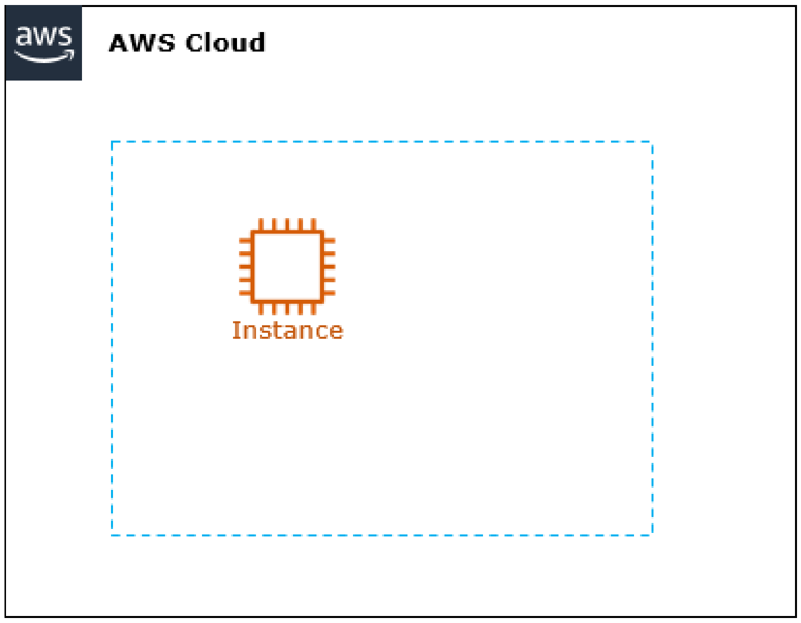
After you complete this lab, you will be able to:

- Create the instance with docker.
- Create container image for mysql database.
- Create docker container using mysql database image.
- Create database and table in mysql database container.
- Create container image for web application.
- Create docker container using web application.
- Access containerized web application.



Task 1: Deploy Web servers

In this task, you will create ec2 instances and install the docker to create the docker images and docker.



Step 1: Create Docker Server

1. In the **AWS Management Console**, on the **Services** menu, search and Select **CloudFormation**.
2. Choose the **YOUR ALLOCATED REGION** list to the right of your account information on the navigation bar.
3. Select **Create stack** and configure:
   a. In the **Create stack** page:
      i. **Prepare template**: Select **Template is ready**.
      ii. **Template source**: Select **Upload a template file**.
      iii. **Choose file**: Click on **Choose file**.
         a) **Navigate** and **select** the **LAB-Docker**.yaml file.

Note: **LAB-SQS**.yaml template is provided with the **Lab manual**.

Note: AWS template **performing** the **following** tasks:

1. **Creating Linux instances**.
2. **Creating t2.micro** instance.
3. **Install** the **docker**.
4. **Set** the **ubuntus's Password**.

iv. Select **Next**.

b. In the **Specify stack details** page:

i. **Stack name**: Write **LAB-Docker**.

Note: Leave the other details as default.

ii. Select **Next**.

c. In the **Configure stack options** page:

Note: Leave all the details as default.

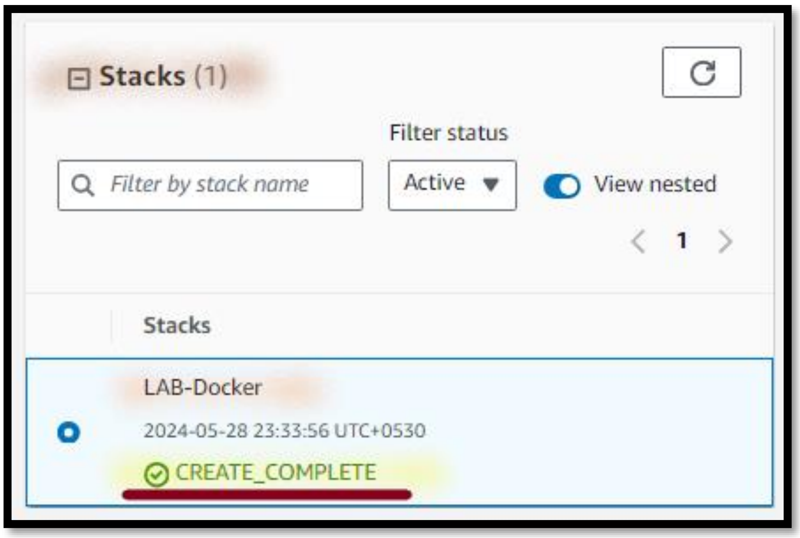
i. Select **Next**.

d. In the **Review** page:

i. Select **Submit**.

Note: You can see the **Stack** status as **CREATE_IN_PROGRESS**.

Note: **Wait**, till you can see the **Stack** status as **CREATE_COMPLETE**. You can **Refresh** your screen



Stacks (1)

Filter status

Filter by stack name | Active | View nested

Stacks

LAB-Docker
2024-05-28 23:33:56 UTC+0530
CREATE_COMPLETE

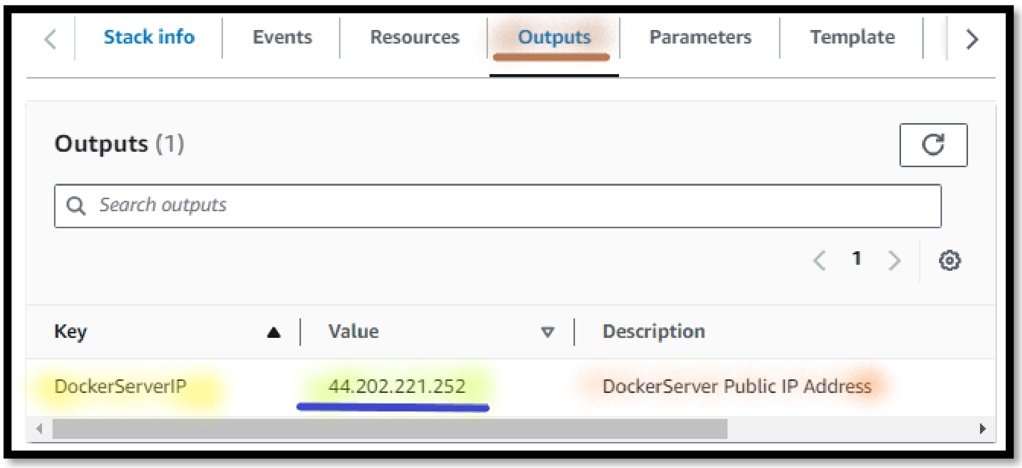
Step 2: View the Output

4. From the LAB-Docker CloudFormation console:

   a. Select Outputs.

Note: You can see the **resources details**.

Note: **Copy** the **WebServer Public IP** in the **Notepad**.



Outputs (1)

Search outputs

Key	Value	Description
DockerServerIP	44.202.221.252	DockerServer Public IP Address

Step 3: Connect to Docker Server

5. From the **Local Desktop/ Laptop** (*Windows desktop*), **Open** the **MobaXterm**.

6. From the **MobaXterm**.

   a. Select **Session**.

   b. Select **SSH**.

      i. **Remote host**: Write **Public IP address** of the **DockerServer**.

      ii. **Specify username**: **Enable** the **Checkmark**.

         a) **Specify username**: Write **ubuntu**.

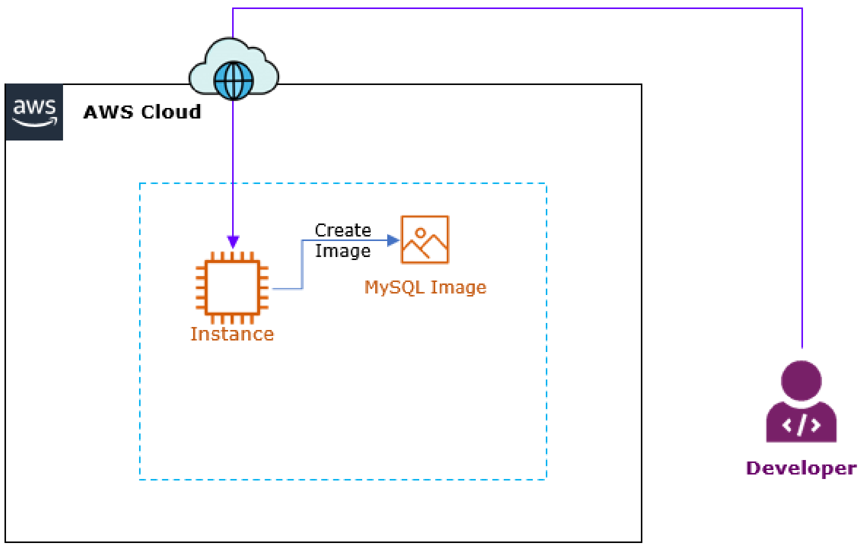
      iii. Select **Ok**.

      iv. **Password**: Write the **lab-password** (*which you have set using the user data*).

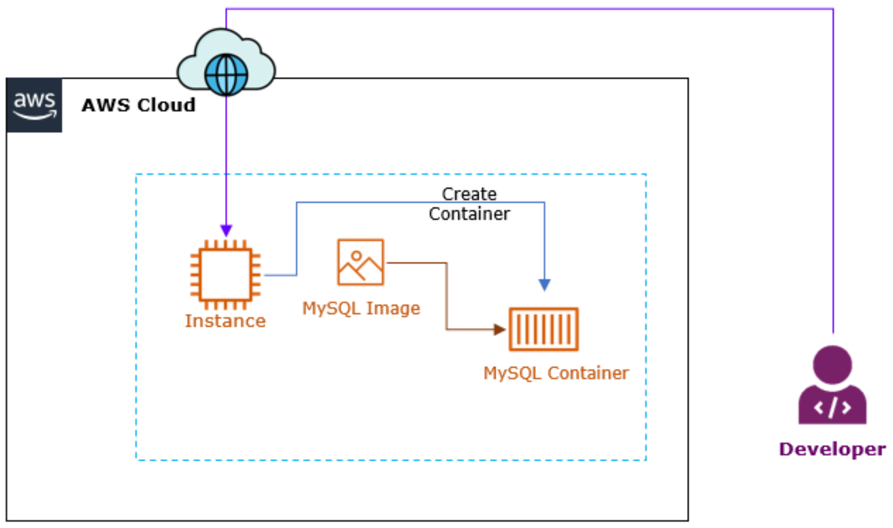
Note: You can see the **Linux Console**.

Task 2: Create DB Container

In this task, you will create the database (mysql) container image.



You then, using the database (mysql) container image, you are creating the database container with database and table schema.



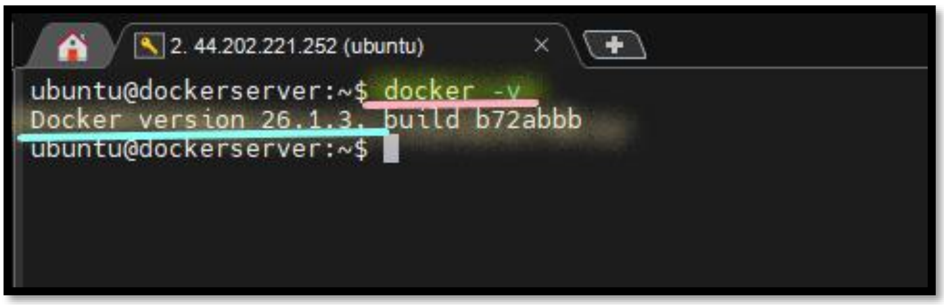
Step 1: Verify the Docker Version

7. From the **DockerServer** **terminal**:

   a. **Execute** the *below command*, to **verify** the **docker version**:

```
docker -v
```

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



```
ubuntu@dockerserver:~$ docker -v
Docker version 26.1.3, build b72abbb
ubuntu@dockerserver:~$
```

Step 2: Create Database Container

8. **From** the **DockerServer** **terminal**:

   a. **Execute** the *below command*, to **download** the **MySQL 5.6 image**:

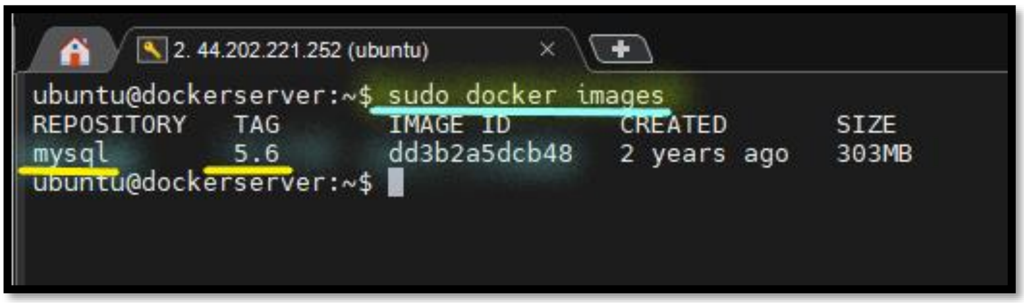
```
sudo docker pull mysql:5.6
```

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

   b. **Execute** the *below command*, to **verify** the **docker image**:

```
sudo docker images
```

Note: In the **Output**, you can see the **mysql:5.6 docker image**.



```
ubuntu@dockerserver:~$ sudo docker images
REPOSITORY    TAG        IMAGE ID       CREATED        SIZE
mysql         5.6        dd3b2a5dcb48   2 years ago    303MB
ubuntu@dockerserver:~$
```

   c. **Execute** the *below 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.

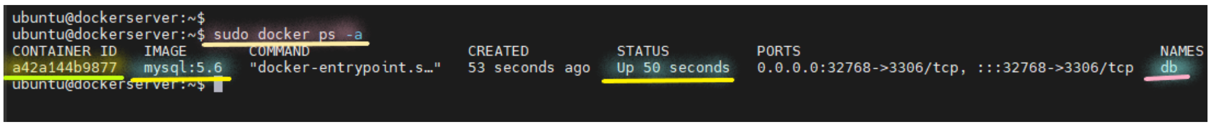
Note: In the **Output**, you can see the **STDOUT**.

d. **Execute** the *below 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**.



```
ubuntu@dockerserver:~$
ubuntu@dockerserver:~$ sudo docker ps -a
CONTAINER ID   IMAGE       COMMAND                  CREATED          STATUS          PORTS                                         NAMES
a42a144b9877   mysql:5.6   "docker-entrypoint.s…"   53 seconds ago   Up 50 seconds   0.0.0.0:32768->3306/tcp, :::32768->3306/tcp   db
ubuntu@dockerserver:~$
```

Step 3: Connect to Database Container

9. **From** the **DockerServer terminal**:

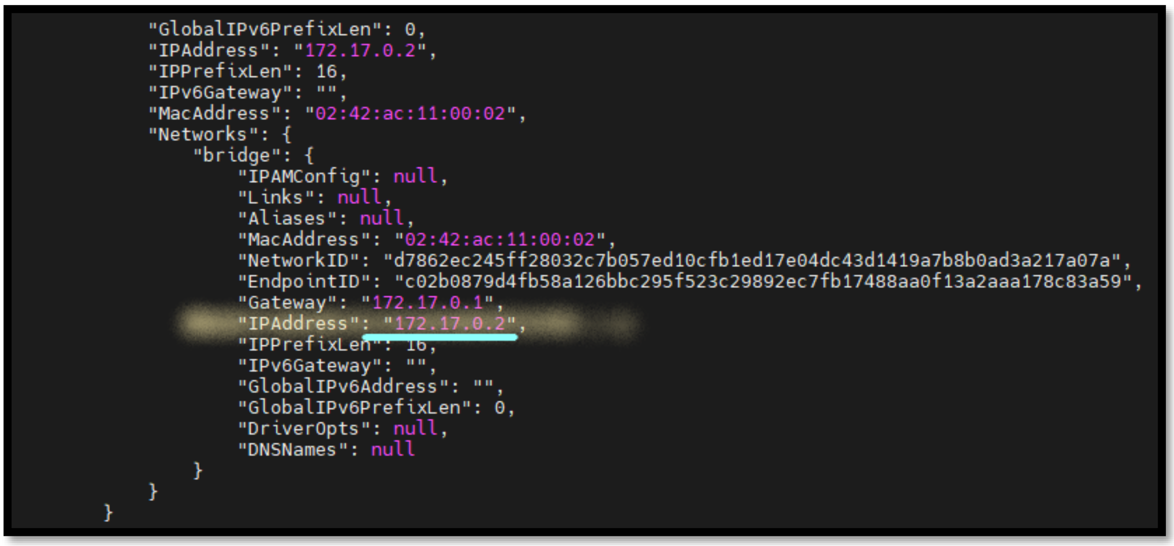
a. **Execute** the *below command*, to **get details** of the **db container**:

```
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**.

Note: **Scroll below** in the docker console to **view** the **details**.



```
            "GlobalIPv6PrefixLen": 0,
            "IPAddress": "172.17.0.2",
            "IPPrefixLen": 16,
            "IPv6Gateway": "",
            "MacAddress": "02:42:ac:11:00:02",
            "Networks": {
                "bridge": {
                    "IPAMConfig": null,
                    "Links": null,
                    "Aliases": null,
                    "MacAddress": "02:42:ac:11:00:02",
                    "NetworkID": "d7862ec245ff28032c7b057ed10cfb1ed17e04dc43d1419a7b8b0ad3a217a07a",
                    "EndpointID": "c02b0879d4fb58a126bbc295f523c29892ec7fb17488aa0f13a2aaa178c83a59",
                    "Gateway": "172.17.0.1",
                    "IPAddress": "172.17.0.2",
                    "IPPrefixLen": 16,
                    "IPv6Gateway": "",
                    "GlobalIPv6Address": "",
                    "GlobalIPv6PrefixLen": 0,
                    "DriverOpts": null,
                    "DNSNames": null
                }
            }
        }
```

b. **Execute** the *below command*, to **install** the **MySQL** **db client**:

```
sudo apt-get install -y mysql-client
```

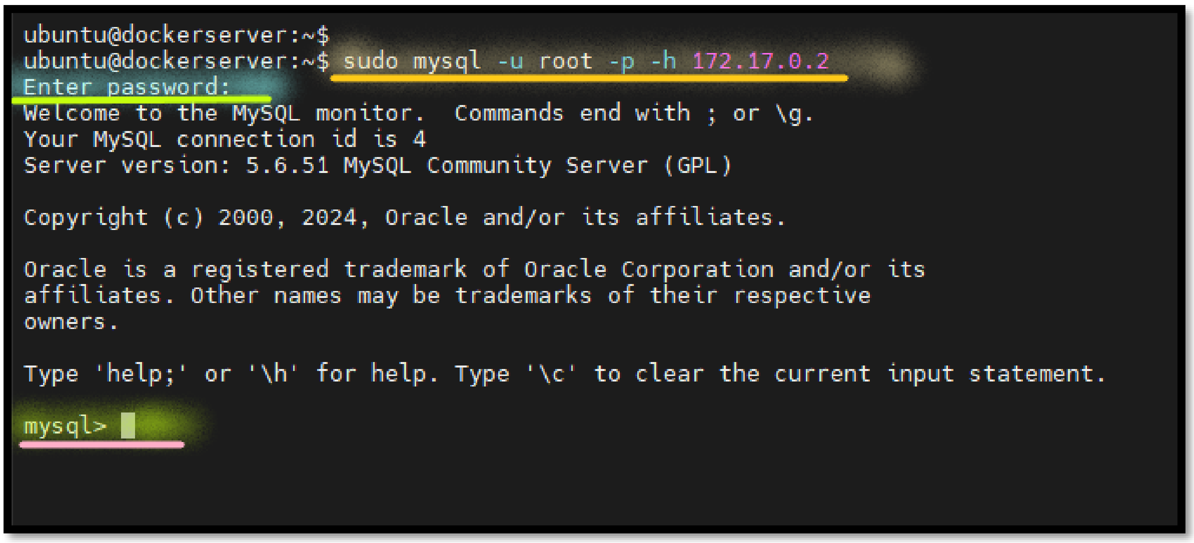
c. **Execute** the *below command*, to **connect** to the **db container**:

```
sudo mysql -u root -p -h DB-PRIVATE-IP
```

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

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

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



```
ubuntu@dockerserver:~$
ubuntu@dockerserver:~$ sudo mysql -u root -p -h 172.17.0.2
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 4
Server version: 5.6.51 MySQL Community Server (GPL)

Copyright (c) 2000, 2024, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

Step 4: Create Database and Table Schema

10. **From** the **MySQL terminal**:

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

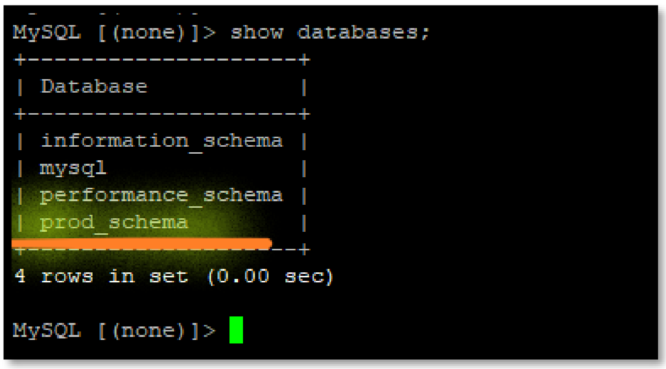
```
create database prod_schema;
```

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

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

```
show databases;
```

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



```
MySQL [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| prod_schema        |
+--------------------+
4 rows in set (0.00 sec)

MySQL [(none)]>
```

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

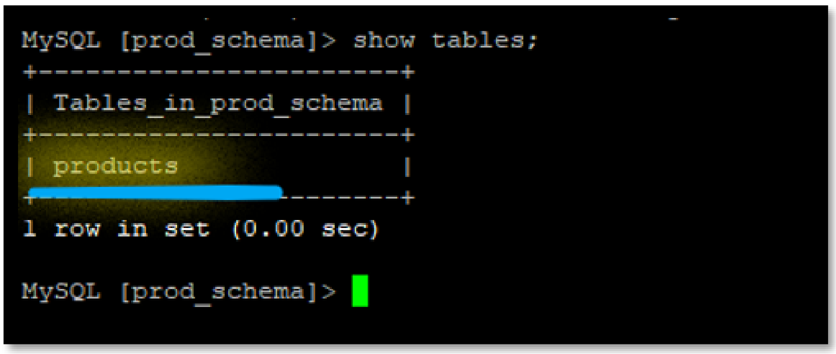
```
create table products (id int NOT NULL AUTO_INCREMENT, name varchar(255),
quantity varchar(255), price varchar(255), PRIMARY KEY (id));
```

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

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

```
show tables;
```

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



```
MySQL [prod_schema]> show tables;
+------------------------+
| Tables_in_prod_schema  |
+------------------------+
| products               |
+------------------------+
1 row in set (0.00 sec)

MySQL [prod_schema]>
```

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

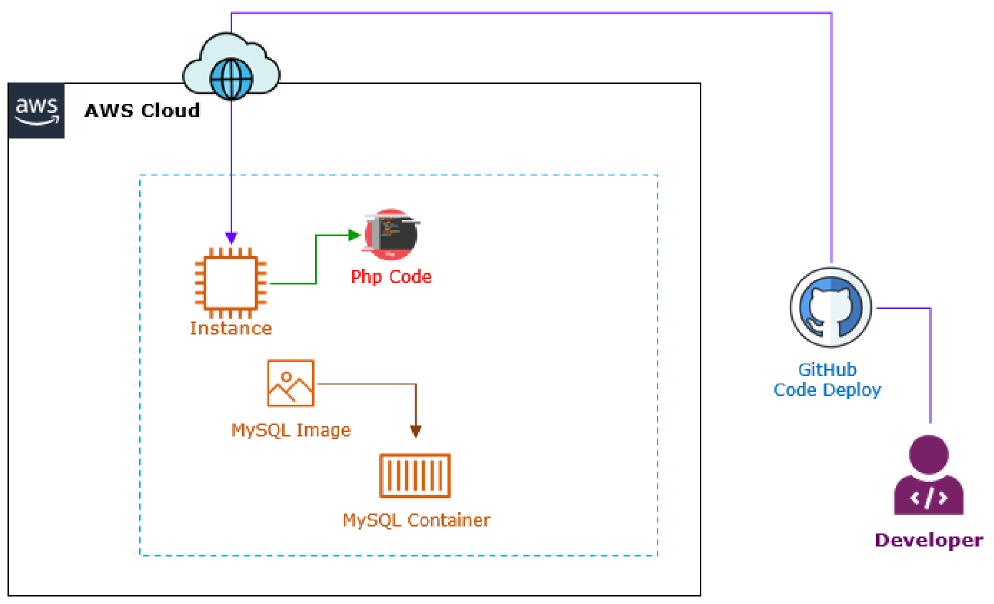
```
exit
```

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

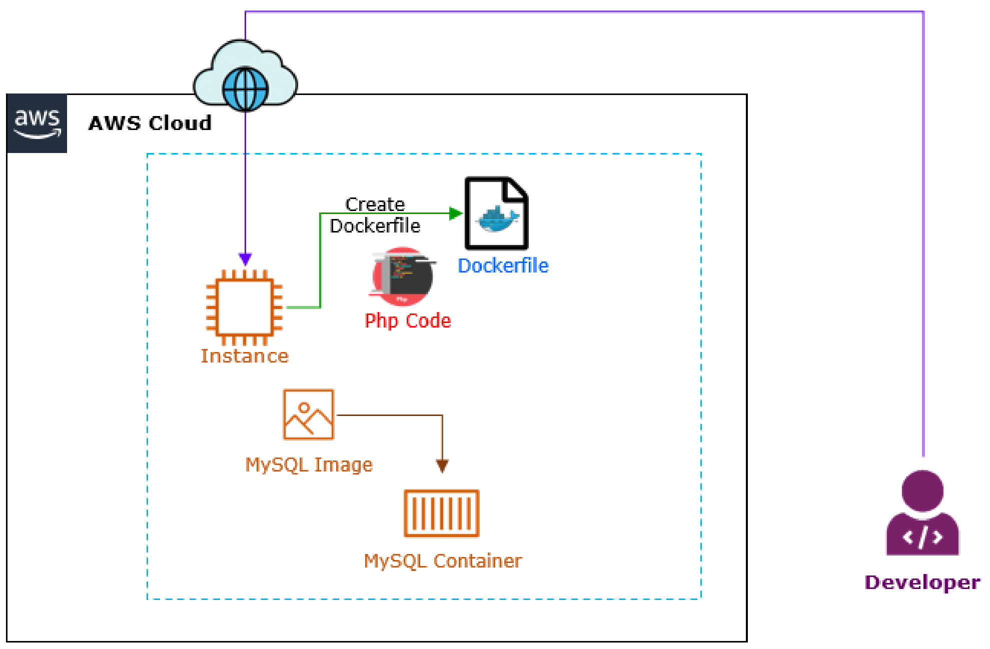
Note: **Go to the next task.** But **Don't close** the **Linux terminal**.

Task 3: Create WebApp container

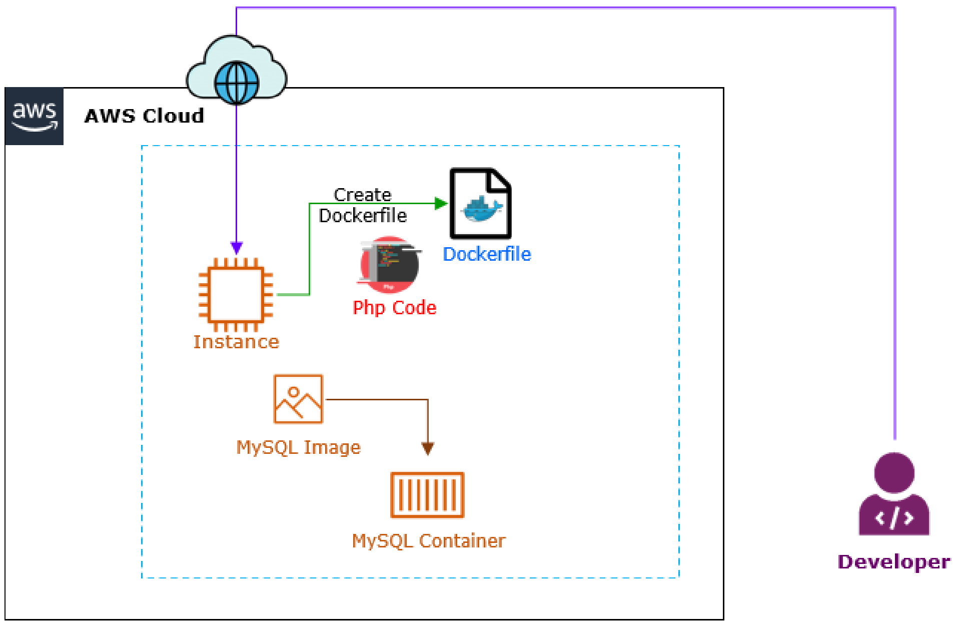
In this task, you will deploy the php web application.



You will create the web application using the docker with php web application.



You will create the web application container using the php web application container image.



Step 1: Develop the Code for WebApp Container

11. **Unzip** the **web-code**.zip (**Php code**).

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

12. **Open** the **data.php** in the **Notepad**.

13. **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.

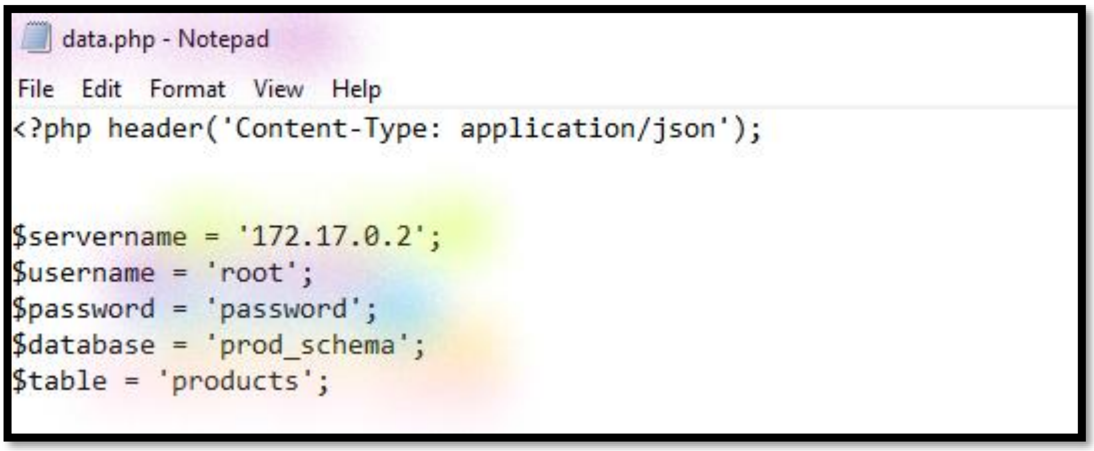
Note: **Don't remove** the **starting** and **ending** **quote** (' ') and semi-colon (**;**).

   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 **database table** name **products**.



```
<?php header('Content-Type: application/json');


$servername = '172.17.0.2';
$username = 'root';
$password = 'password';
$database = 'prod_schema';
$table = 'products';
```

i. Select **File**.

a) Select **Save**.

Step 2: Create GitHub Repository

14. **Login** into your **GitHub account**.

15. Go to **right top side** and select **+** sign.

a. Select **New repository**.

i. **Repository name**: Write **webapp**.

ii. Select **Public**.

iii. Select **Create repository**.

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

b. **From** the **webapp** repository:

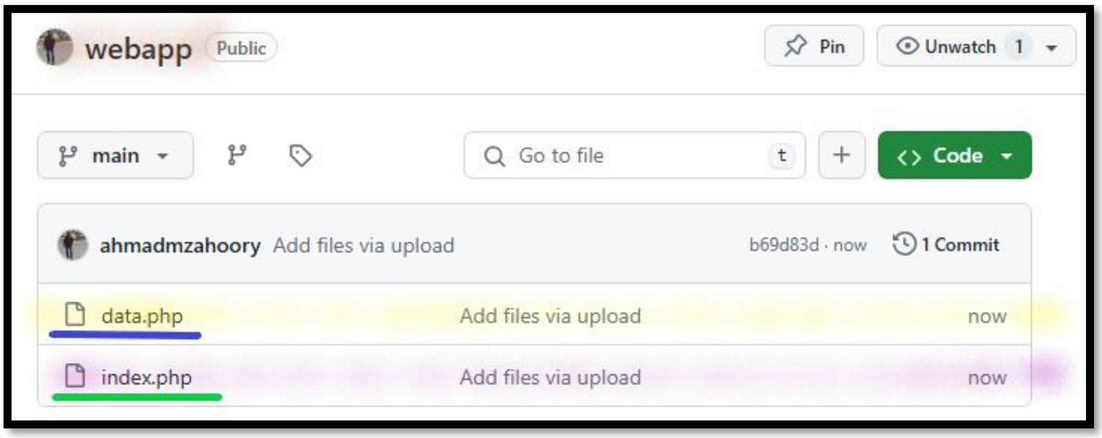
i. Select **uploading an existing file**.

a) Select **Choose your files**.

1) **Navigate** and **select** **index.php** and **data.php** (*which you have updated in the last step*).

2) Select **Commit Changes**.

Note: You can see the **index.php** and **data.php** in the **github repository**.



ahmadmzahoory Add files via upload b69d83d · now 1 Commit

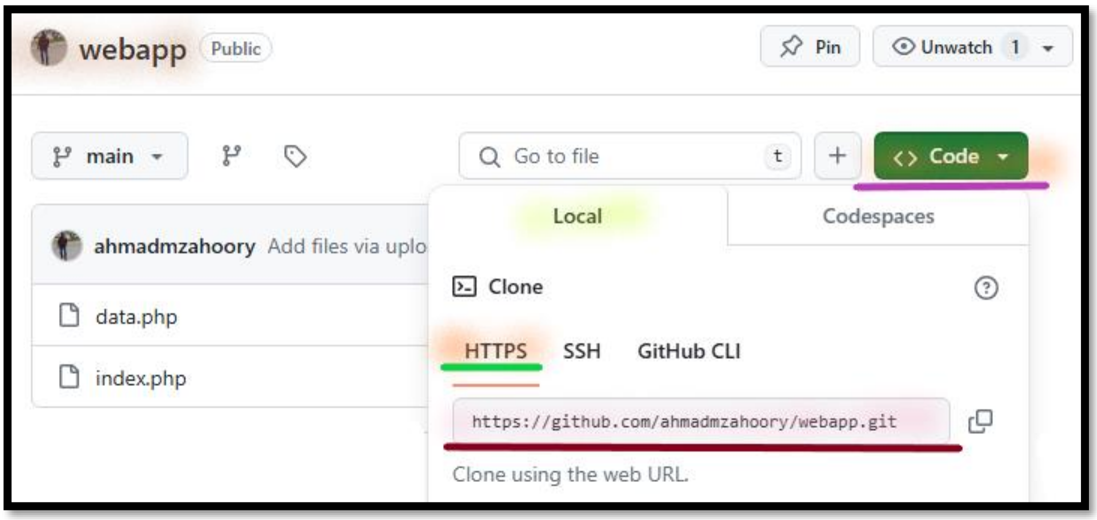
data.php	Add files via upload	now
index.php	Add files via upload	now

c. **From** the **webapp** repository:

i. Select **Code**.

a) Select **HTTPS**.

1) **Copy** the **Clone URL** in **Notepad**.



ahmadmzahoory Add files via uplo

data.php

index.php

```
https://github.com/ahmadmzahoory/webapp.git
```

Clone using the web URL.

Step 3: Clone the Code Files

16. **Return** to the **DockerServer**.

17. **From** the **DockerServer** **terminal**:

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

```
sudo apt-get install -y git
```

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

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

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

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

```
ls -l
```

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

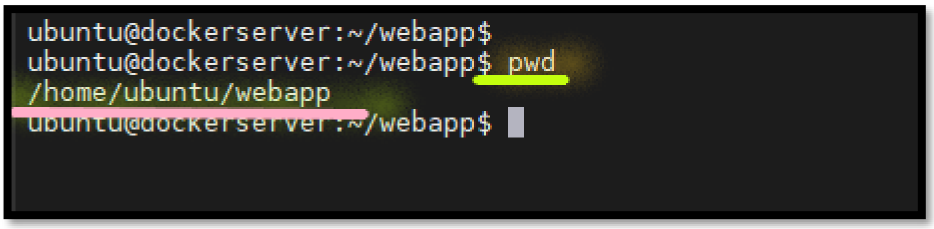
d. **Execute** the *below command*, to **change directory** to **webapp**:

```
cd ./webapp
```

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

```
pwd
```

Note: In the **Output**, you can see the **/home/ubuntu/webapp** path.

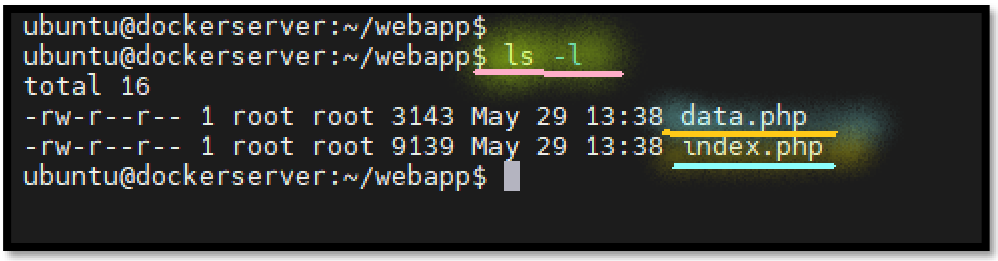


```
ubuntu@dockerserver:~/webapp$
ubuntu@dockerserver:~/webapp$ pwd
/home/ubuntu/webapp
ubuntu@dockerserver:~/webapp$
```

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

```
ls -l
```

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



```
ubuntu@dockerserver:~/webapp$
ubuntu@dockerserver:~/webapp$ ls -l
total 16
-rw-r--r-- 1 root root 3143 May 29 13:38 data.php
-rw-r--r-- 1 root root 9139 May 29 13:38 index.php
ubuntu@dockerserver:~/webapp$
```

g. **Execute** the *below command*, to **install** the **nano editor**:

```
sudo apt-get install -y nano
```

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

```
sudo nano index.php
```

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

i. Press **CTRL + X**, to **exit** the nano editor.

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

```
sudo nano data.php
```

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

i. Press **CTRL + X**, to **exit** the nano editor.

Step 4: Create Dockerfile

18. **From** the DockerServer terminal:

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

```
cd ..
```

   b. Execute the *below command*, to verify the current path:

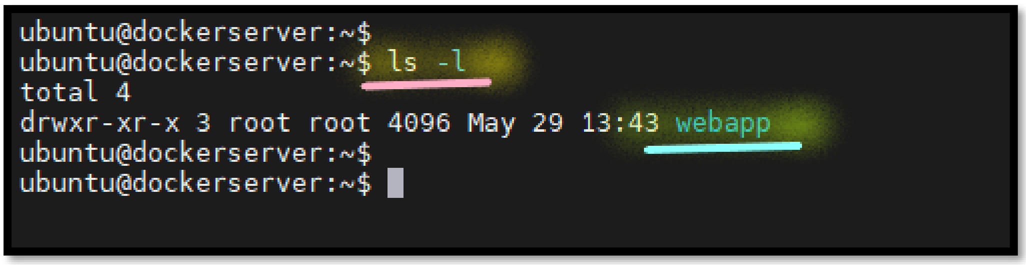
```
pwd
```

Note: In the **Output**, you can see the **/home/ubuntu** path.

   c. Execute the *below command*, to list the file and folders:

```
ls -l
```

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



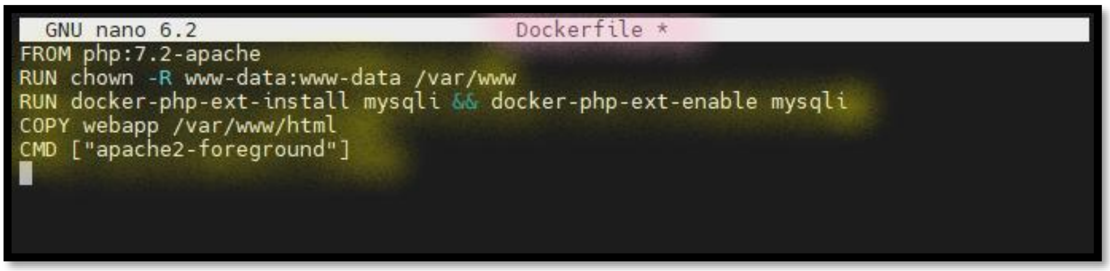
```
ubuntu@dockerserver:~$
ubuntu@dockerserver:~$ ls -l
total 4
drwxr-xr-x 3 root root 4096 May 29 13:43 webapp
ubuntu@dockerserver:~$
ubuntu@dockerserver:~$
```

   d. Execute the *below command* to create the file named **Dockerfile**.

```
sudo nano Dockerfile
```

      a) Copy the instructions in the **Dockerfile**.

```
FROM php:7.2-apache
RUN chown -R www-data:www-data /var/www
RUN docker-php-ext-install mysqli && docker-php-ext-enable mysqli
COPY webapp /var/www/html
CMD ["apache2-foreground"]
```



```
  GNU nano 6.2                          Dockerfile *
FROM php:7.2-apache
RUN chown -R www-data:www-data /var/www
RUN docker-php-ext-install mysqli && docker-php-ext-enable mysqli
COPY webapp /var/www/html
CMD ["apache2-foreground"]
```

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**.

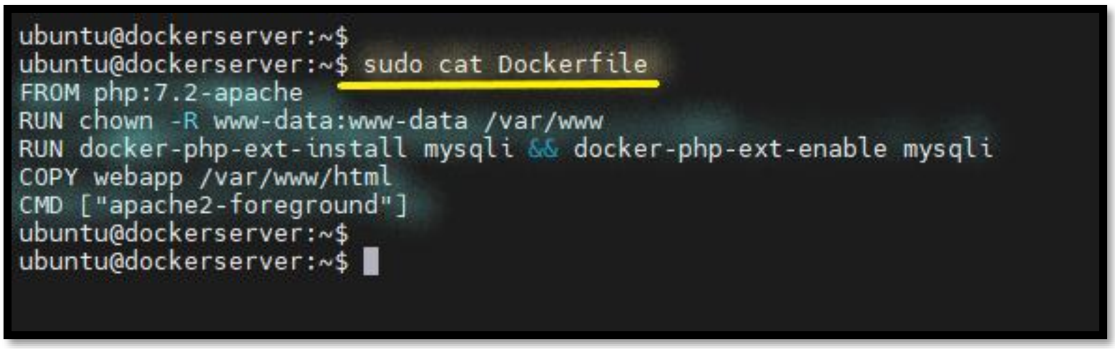
1) Press **CTRL + O** and press **Enter** key (*to save*).

2) Press **CTRL + X** (*to exit*).

e. **Execute** the *below command*, to **view** the **dockerfile content**:

```
sudo cat Dockerfile
```

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



```
ubuntu@dockerserver:~$
ubuntu@dockerserver:~$ sudo cat Dockerfile
FROM php:7.2-apache
RUN chown -R www-data:www-data /var/www
RUN docker-php-ext-install mysqli && docker-php-ext-enable mysqli
COPY webapp /var/www/html
CMD ["apache2-foreground"]
ubuntu@dockerserver:~$
ubuntu@dockerserver:~$
```

Step 5: Create Docker Image

19. **From** the **DockerServer** **terminal**:

    a. **Execute** the *below command*, to **build** a **docker image**:

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

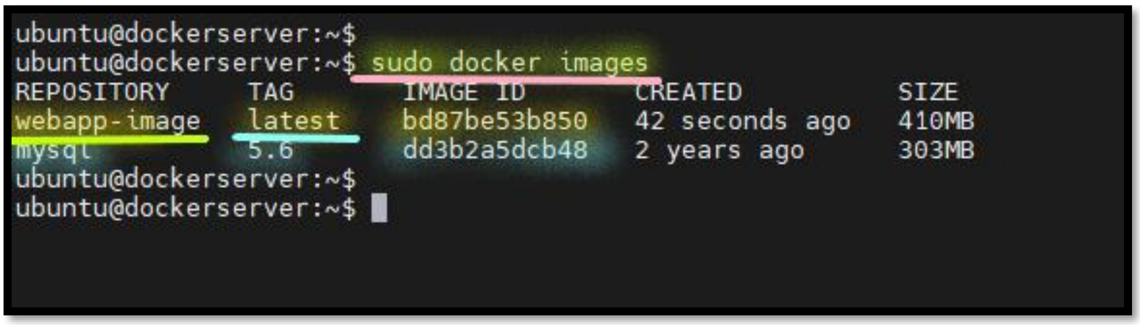
Note: **Make sure** to **copy** the whole command **including** the '.' .

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

    b. **Execute** the *below command*, to **verify** a **docker image**:

```
sudo docker images
```

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



```
ubuntu@dockerserver:~$
ubuntu@dockerserver:~$ sudo docker images
REPOSITORY     TAG       IMAGE ID       CREATED          SIZE
webapp-image   latest    bd87be53b850   42 seconds ago   410MB
mysql          5.6       dd3b2a5dcb48   2 years ago      303MB
ubuntu@dockerserver:~$
ubuntu@dockerserver:~$
```

Step 6: Run a Docker Container

20. **From** the DockerServer terminal:

    a. Execute the *below command*, to launch a container from the docker image you **build**:

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

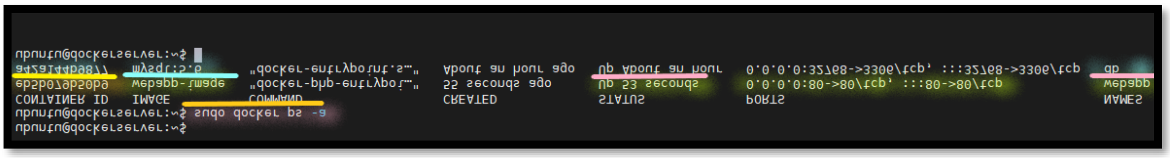
> **Note**: This command requests Docker to run a container, with the name *webapp*, in daemon mode (non-interactive) and map tcp/80 outside the container to tcp/80 on the inside of the container.

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

    b. Execute the *below 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**.



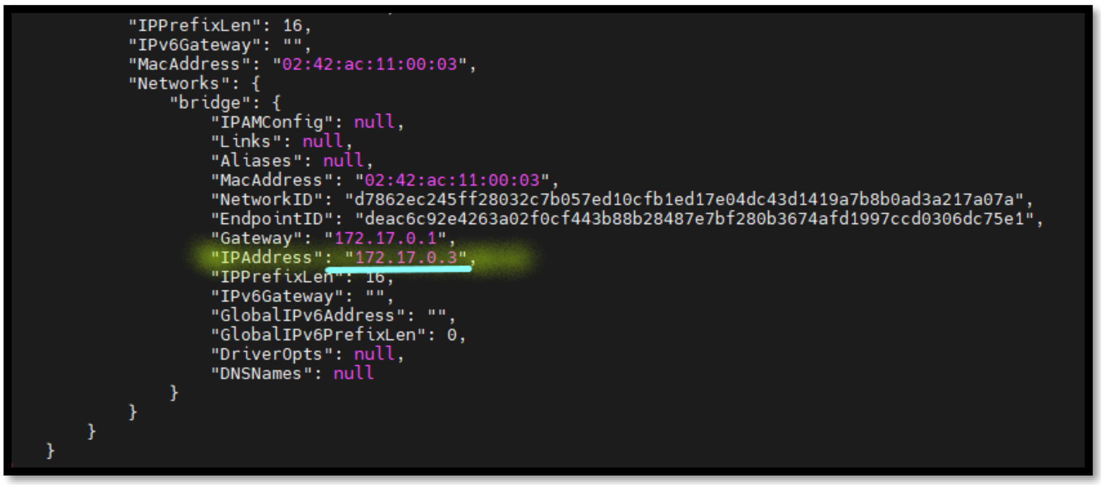
```
ubuntu@dockerserver:~$
ubuntu@dockerserver:~$ sudo docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED              STATUS              PORTS                                         NAMES
[illegible]    webapp-image   "docker-php-entrypoi…"   22 seconds ago       Up 23 seconds       0.0.0.0:80->80/tcp, :::80->80/tcp             webapp
[illegible]    mysql:5.6      "docker-entrypoint.s…"   About an hour ago    Up About an hour    0.0.0.0:32768->3306/tcp, :::32768->3306/tcp   db
ubuntu@dockerserver:~$
```

    c. Execute the *below 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.

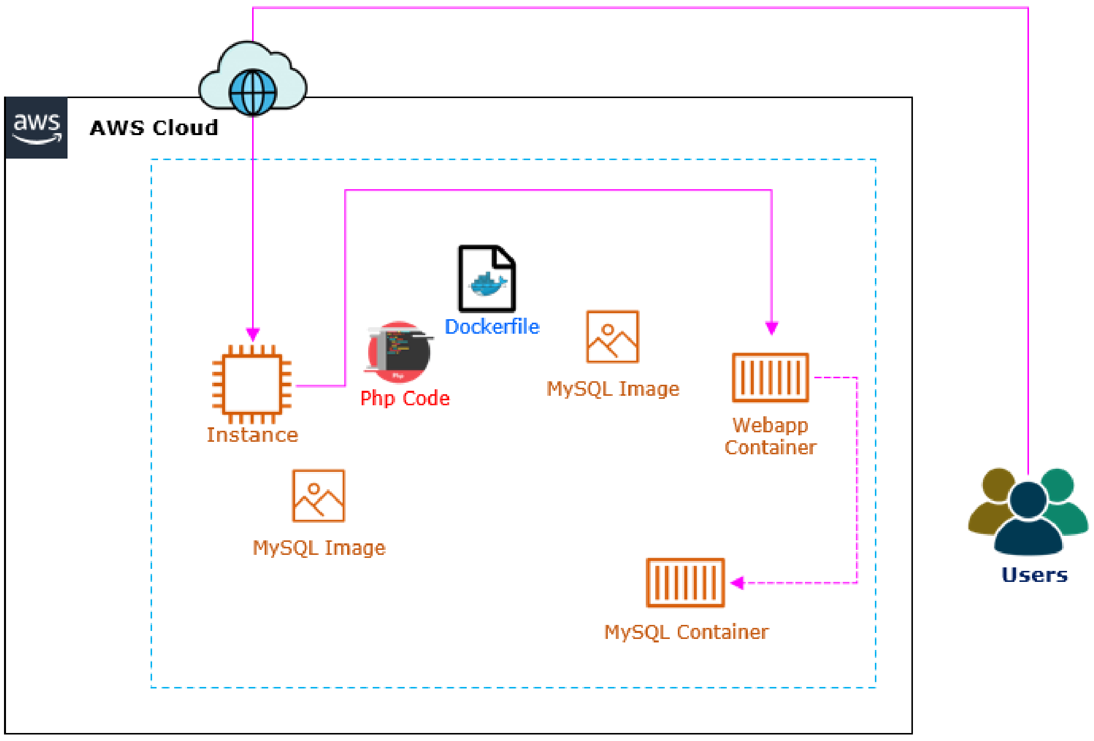
Note: You can view the **webapp container Private IP address**.



```
        "IPPrefixLen": 16,
        "IPv6Gateway": "",
        "MacAddress": "02:42:ac:11:00:03",
        "Networks": {
            "bridge": {
                "IPAMConfig": null,
                "Links": null,
                "Aliases": null,
                "MacAddress": "02:42:ac:11:00:03",
                "NetworkID": "d7862ec245ff28032c7b057ed10cfb1ed17e04dc43d1419a7b8b0ad3a217a07a",
                "EndpointID": "deac6c92e4263a02f0cf443b88b28487e7bf280b3674afd1997ccd0306dc75e1",
                "Gateway": "172.17.0.1",
                "IPAddress": "172.17.0.3",
                "IPPrefixLen": 16,
                "IPv6Gateway": "",
                "GlobalIPv6Address": "",
                "GlobalIPv6PrefixLen": 0,
                "DriverOpts": null,
                "DNSNames": null
            }
        }
    }
}
```

Task 4: Access the WebApp container

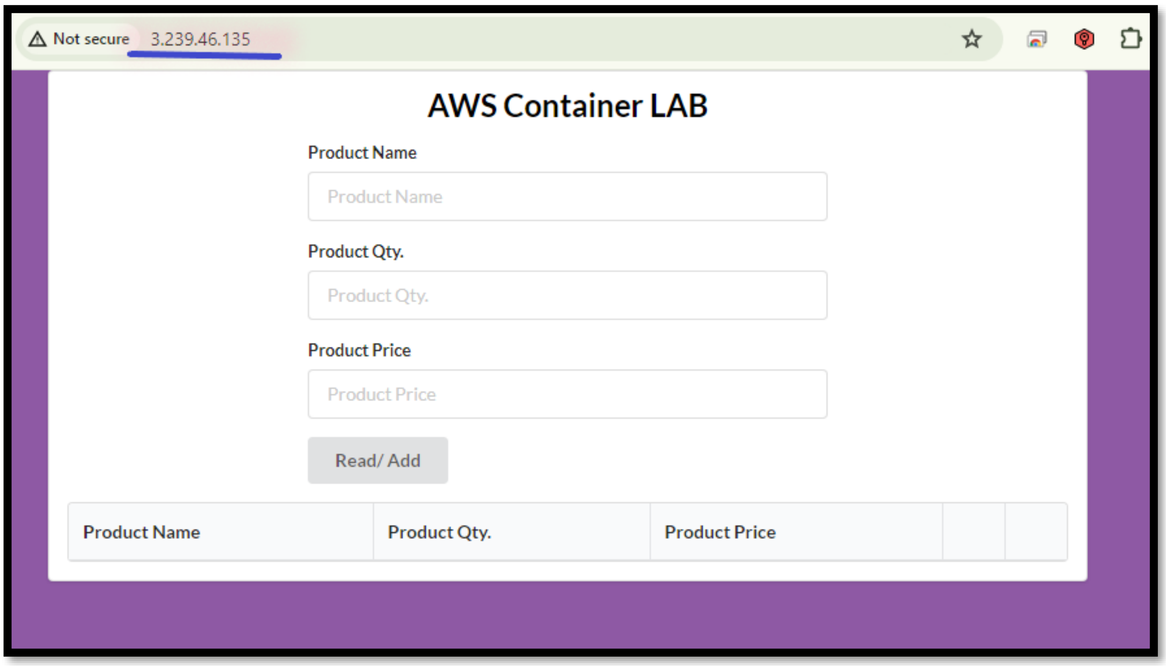
In this task, you can access the web application hosted in container.



Step 1: Access the WebApp Container

21. **From** your **Local desktop/ laptop** (*Windows desktop*) Open the **Browser** and **Copy** the **Public IP address** of the **Docker Server** to access the **website**.

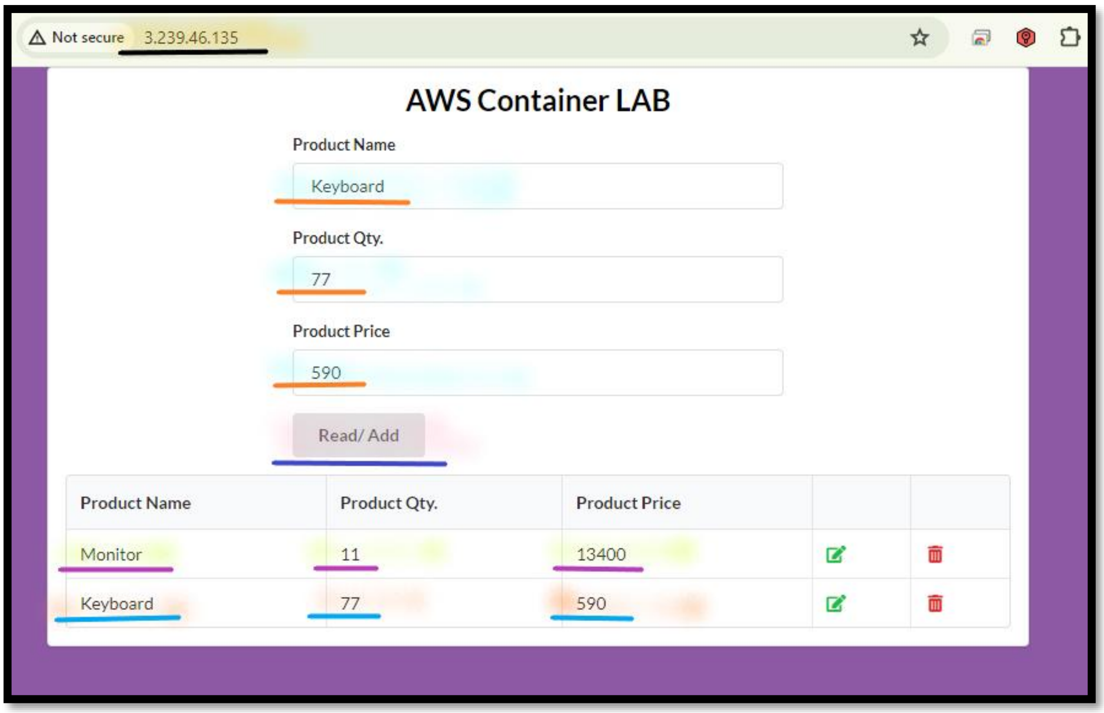
Note: You can see the **WebApp**.



Product Name	Product Qty.	Product Price		

a. From the **Webapp**:

   i. **Add** the **Product Data**.



AWS Container LAB

Product Name: Keyboard

Product Qty.: 77

Product Price: 590

Product Name	Product Qty.	Product Price		
Monitor	11	13400		
Keyboard	77	590		

a) You can also **Update** the **Product Data**.

b) You can also **Delete** the **Product Data**.

Note: **Go to the next task**. But **Don't close** the **Webapp** page.

Step 3: Add Data from MySQL Container

22. **From** the **DockerServer** **terminal**:

a. **Execute** the *below command*, to **connect** to the **db container**:

```
sudo mysql -u root -p -h DB-PRIVATE-IP
```

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

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

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

23. **From** the **MySQL** **terminal**:

a. **Execute** the *below 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 *below command*, to **show** **tables**:

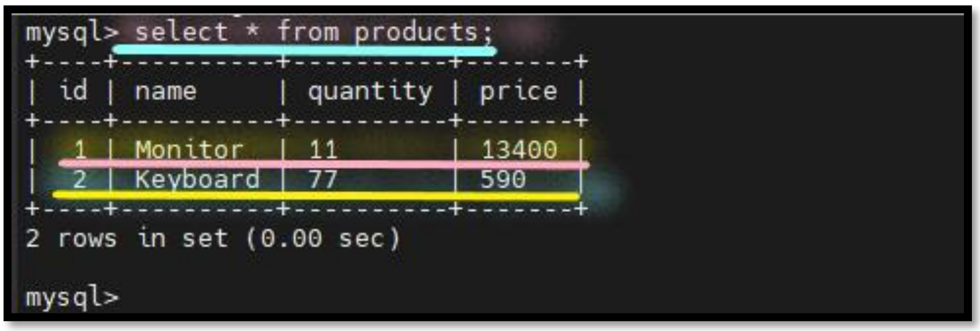
```
show tables;
```

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

c. **Execute** the *below command*, to **show** **data** from **products** **tables**:

```
select * from products;
```

Note: In the **Database**, you can see the **data** **added** from **webapp** Container.



```
mysql> select * from products;
+----+----------+----------+-------+
| id | name     | quantity | price |
+----+----------+----------+-------+
|  1 | Monitor  | 11       | 13400 |
|  2 | Keyboard | 77       | 590   |
+----+----------+----------+-------+
2 rows in set (0.00 sec)

mysql>
```

d. **Execute** the *below 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 *below command*, to **exit** **mysql**:

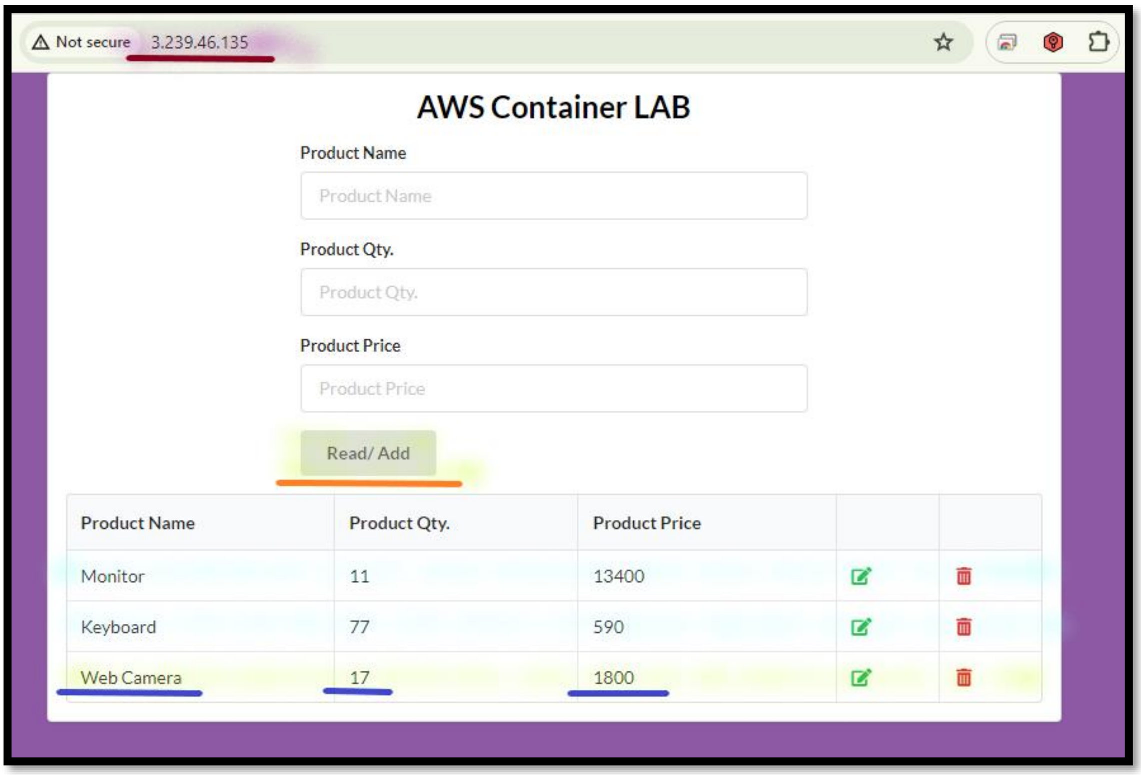
```
exit
```

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

Step 4: Access the WebApp Container

24. **Return** to the **Browser** (*from where you were accessing the webapp website*) and select **Read/ Add**.

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



Product Name	Product Qty.	Product Price		
Monitor	11	13400		
Keyboard	77	590		
Web Camera	17	1800		

Task 5: Delete the Environment

Step 1: Delete the Stack

25. In the **AWS Management Console**, on the **Services** menu, search and select **CloudFormation**.
26. Select **Stack**.
    a. Select **LAB-Docker**.
        i. Select **Delete**.
            a) Select **Delete**.

Step 2: Delete the Buckets

27. In the **AWS Management Console**, on the **Services** menu, search and select **S3**.
28. Select **Buckets**.
    a. Select **cf-templates-xxx** bucket.
        i. Select **Empty**.
            a) **Type** **permanently delete** to delete all the objects.
            b) Select **Empty**.
            c) Select **Exit**.
    b. Select **cf-templates-xxx** bucket.
        i. Select **Delete**.
            a) Type **cf-templates-xxx** bucket name to delete bucket.
            b) Select **Delete bucket**.