| | | |
|---|---|---|
| Marwadi University<br>Marwadi Chandarana Group | **Marwadi University**<br>**Faculty of Engineering and Technology**<br>**Department of Information and Communication Technology** | |
| **Subject: Cloud Developing (01CT0720)** | **Aim: Migrate a Web Application to Docker Containers.** | |
| **Experiment No: 07** | **Date:** | **Enrolment No: 92200133030** |

**Aim** :- Migrate a Web Application to Docker Containers.

**Lab overview and objectives**

In this lab, you will migrate a web application to run on Docker containers. The application is installed directly on the guest operating systems (OSs) of two Amazon Elastic Compute Cloud (Amazon EC2) instances. You will migrate the application to run on Docker containers.

After completing this lab, you should be able to:

- Create a Dockerfile.
- Create a Docker image by using a Dockerfile.
- Run a container from a Docker image.
- Interact with and administer your containers.
- Create an Amazon Elastic Container Registry (Amazon ECR) repository.
- Authenticate the Docker client to Amazon ECR.
- Push a Docker image to Amazon ECR.

**AWS service restrictions**

In this lab environment, access to AWS services and service actions might be restricted to the ones that are needed to complete the lab instructions. You might encounter errors if you attempt to access other services or perform actions beyond the ones that are described in this lab.

**Scenario**

The café owners have noticed how popular their gourmet coffee offerings have become. Customers cannot seem to get enough of their cappuccinos and lattes. Meanwhile, the café owners have been challenged to consistently source the highest quality coffee beans. Recently, the owners learned that one of their favorite coffee suppliers wants to sell her company. Frank and Martha jumped at the opportunity to buy the company.

The acquired coffee supplier runs an inventory tracking application on an AWS account. Sofía has been tasked to understand how the application works and then create a plan to integrate the application into the café's existing application infrastructure.

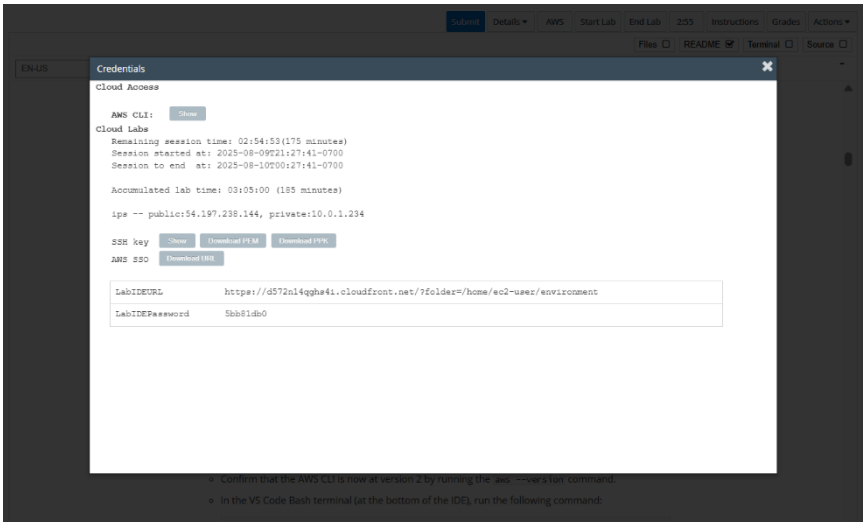
In this lab, you will again play the role of Sofía, and you will work to migrate the application to run on containers.

**Task 1: Preparing the lab**

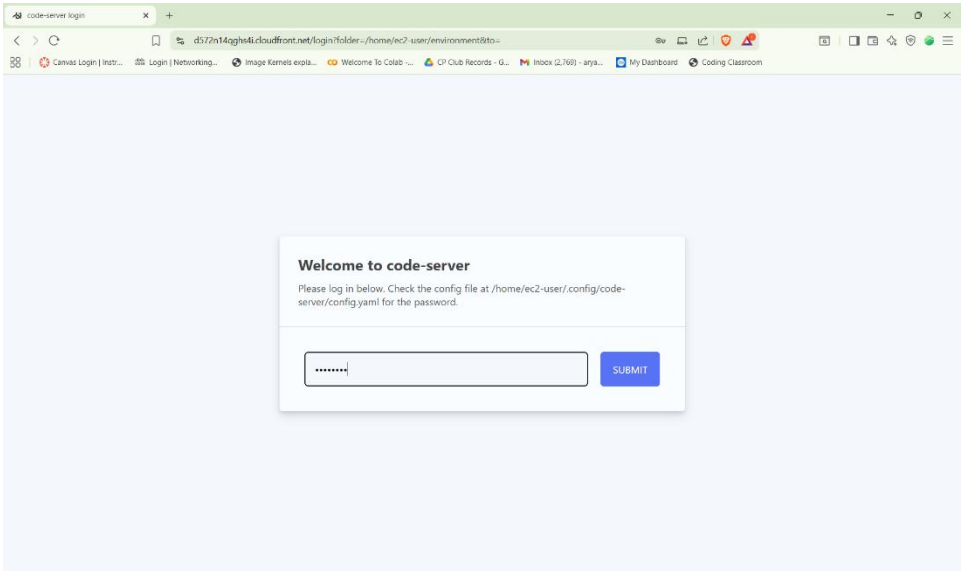
Connect to the VS Code IDE.

1. At the top of these instructions, choose Details followed by **AWS: Show**
2. Copy values from the table **similar** to the following and paste it into an editor of your choice for use later.
    a. **LabIDEURL**

b. **LabIDEPassword**

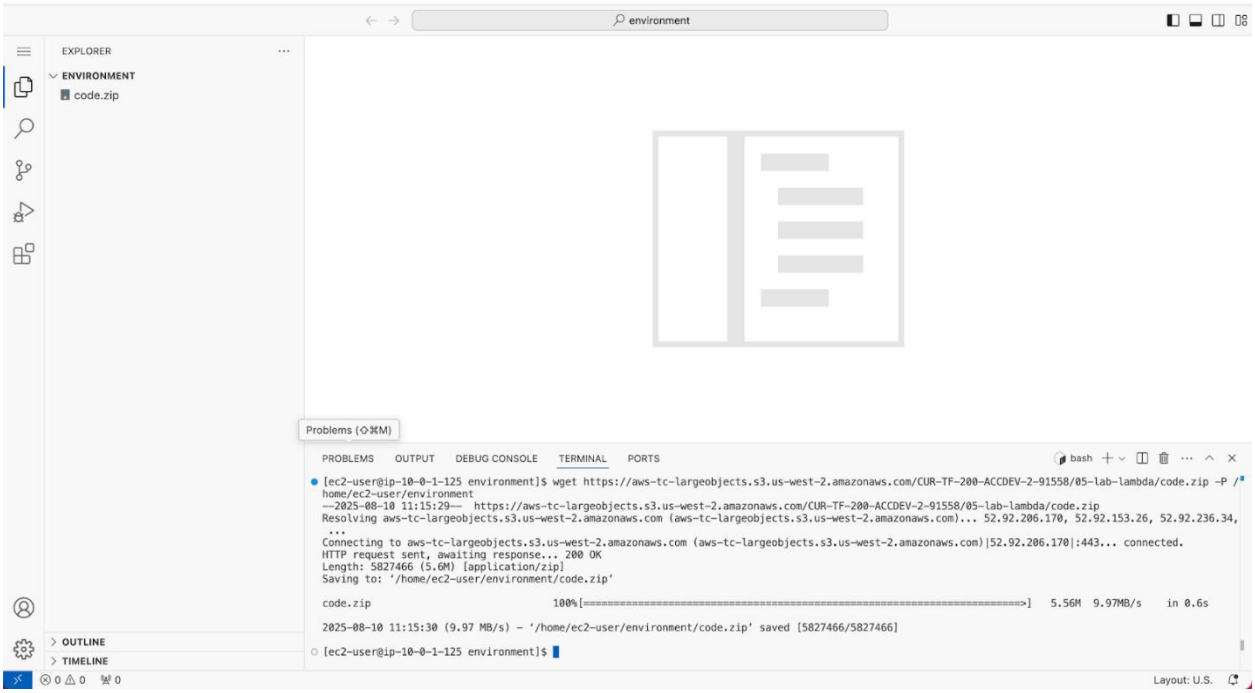



3. In a new browser tab, paste the value for **LabIDEURL** to open the VS Code IDE.
4. On the prompt window **Welcome to code-server**, enter the value for **LabIDEPassword** you copied to the editor earlier, choose **Submit** to open the VS Code IDE.

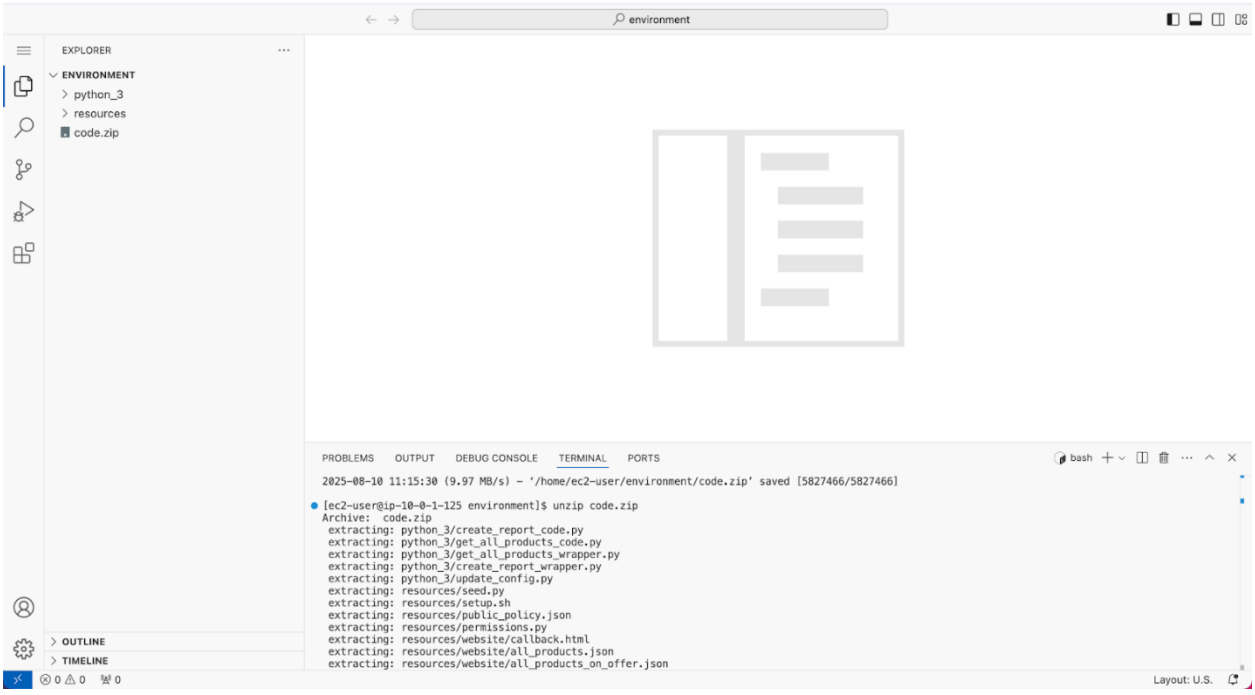



5. Download and extract the files that you need for this lab.
   - In the VS Code bash terminal (located at the bottom of the IDE), run the following commands:

```
wget https://aws-tc-largeobjects.s3.us-west-2.amazonaws.com/CUR-TF-200-ACCDEV-2-91558/06-lab-containers/code.zip -P /home/ec2-user/environment
```




6. You should see that the **code.zip** file was downloaded to the VS Code IDE and is now in the left navigation pane.
   - Extract the file by running the following command:
     `unzip code.zip`



| Marwadi University<br>Marwadi Chandarana Group | **Marwadi University**<br>**Faculty of Engineering and Technology**<br>**Department of Information and Communication Technology** | |
|---|---|---|
| **Subject: Cloud Developing (01CT0720)** | **Aim: Migrate a Web Application to Docker Containers.** | |
| **Experiment No: 07** | **Date:** | **Enrolment No: 92200133030** |

7. Run a script that upgrades the version of the AWS CLI installed on the VS Code IDE.
   - To set permissions on the script and then run it, run the following commands in the Bash terminal:

```
chmod +x ./resources/setup.sh && ./resources/setup.sh
```

```
[ec2-user@ip-10-0-1-150 environment]$ chmod +x resources/setup.sh && resources/setup.sh
Please enter a valid IP address:
152.58.63.192
IP address:152.58.63.192
upload: resources/website/all_products_on_offer.json to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvvevyyhv5l/all_products_on_offer.json
upload: resources/website/callback.html to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvvevyyhv5l/callback.html
upload: resources/website/all_products.json to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvvevyyhv5l/all_products.json
upload: resources/website/beans.json to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvvevyyhv5l/beans.json
upload: resources/website/images/beans/excelsa.png to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvvevyyhv5l/images/beans/excelsa.png
upload: resources/website/config.js to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvvevyyhv5l/config.js
upload: resources/website/images/items/blueberry_bagel.png to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvvevyyhv5l/images/items/blueberry_bagel.pn
g
upload: resources/website/images/items/apple_pie.jpeg to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvvevyyhv5l/images/items/apple_pie.jpeg
upload: resources/website/images/items/blueberry_jelly_doughnut.jpeg to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvvevyyhv5l/images/items/blueberr
y_jelly_doughnut.jpeg
upload: resources/website/images/beans/robusta.png to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvvevyyhv5l/images/beans/robusta.png
upload: resources/website/images/items/boston_cream_doughnut.jpeg to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvvevyyhv5l/images/items/boston_crea
m_doughnut.jpeg
upload: resources/website/images/expanded.png to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvvevyyhv5l/images/expanded.png
upload: resources/website/images/beans/liberica.png to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvvevyyhv5l/images/beans/liberica.png
upload: resources/website/images/items/apple_pie_slice.png to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvvevyyhv5l/images/items/apple_pie_slice.pn
g
upload: resources/website/images/items/apple_pie.png to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvvevyyhv5l/images/items/apple_pie.png
upload: resources/website/images/beans/arabica.png to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvvevyyhv5l/images/beans/arabica.png
upload: resources/website/images/items/boston_cream_doughnut.png to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvvevyyhv5l/images/items/boston_crea
_doughnut.png
upload: resources/website/favicon.ico to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvvevyyhv5l/favicon.ico
upload: resources/website/images/items/apple_pie_slice.jpeg to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvvevyyhv5l/images/items/apple_pie_slice.j
peg
upload: resources/website/images/items/cherry_pie.png to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvvevyyhv5l/images/items/cherry_pie.png
upload: resources/website/images/items/blueberry_bagel.jpeg to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvvevyyhv5l/images/items/blueberry_bagel.j
peg
upload: resources/website/images/items/blueberry_jelly_doughnut.png to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvvevyyhv5l/images/items/blueberry
_jelly_doughnut.png
upload: resources/website/images/items/cherry_pie_slice.png to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvvevyyhv5l/images/items/cherry_pie_slice
png
upload: resources/website/images/items/cherry_pie.jpeg to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvvevyyhv5l/images/items/cherry_pie.jpeg
upload: resources/website/images/items/cherry_pie_slice.jpeg to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvvevyyhv5l/images/items/cherry_pie_slice
.jpeg
upload: resources/website/images/items/chocolate_chip_cupcake.jpeg to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvvevyyhv5l/images/items/chocolate_
```



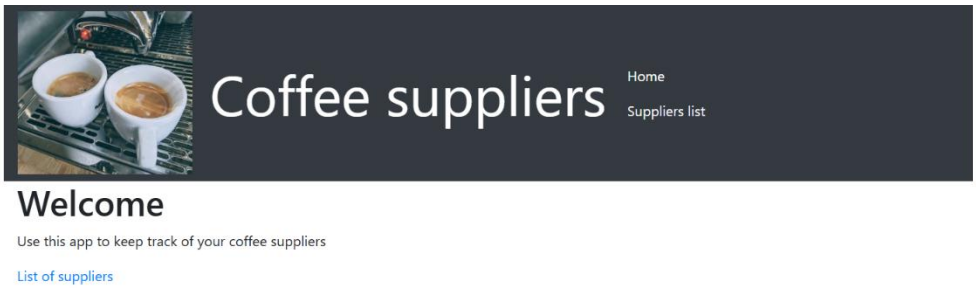
8. Verify the AWS CLI version and also verify that the SDK for Python is installed.
   - Confirm that the AWS CLI is now at version 2 by running the **`aws --version`** command.
   - In the VS Code Bash terminal (at the bottom of the IDE), run the following command: **`pip3 show boto3`**

```
[ec2-user@ip-10-0-1-234 environment]$ aws --version
aws-cli/2.28.6 Python/3.13.4 Linux/6.1.147-172.266.amzn2023.x86_64 exe/x86_64.amzn.2023
[ec2-user@ip-10-0-1-234 environment]$ pip3 show boto3
Name: boto3
Version: 1.40.6
Summary: The AWS SDK for Python
Home-page: https://github.com/boto/boto3
Author: Amazon Web Services
Author-email:
License: Apache License 2.0
Location: /usr/local/lib/python3.11/site-packages
Requires: botocore, jmespath, s3transfer
Required-by:
[ec2-user@ip-10-0-1-234 environment]$
```

| Marwadi University<br>Marwadi Chandarana Group | **Marwadi University**<br>**Faculty of Engineering and Technology**<br>**Department of Information and Communication Technology** | |
|---|---|---|
| **Subject: Cloud Developing (01CT0720)** | **Aim: Migrate a Web Application to Docker Containers.** | |
| **Experiment No: 07** | **Date:** | **Enrolment No: 92200133030** |

**Task 2: Analyzing the existing application infrastructure**

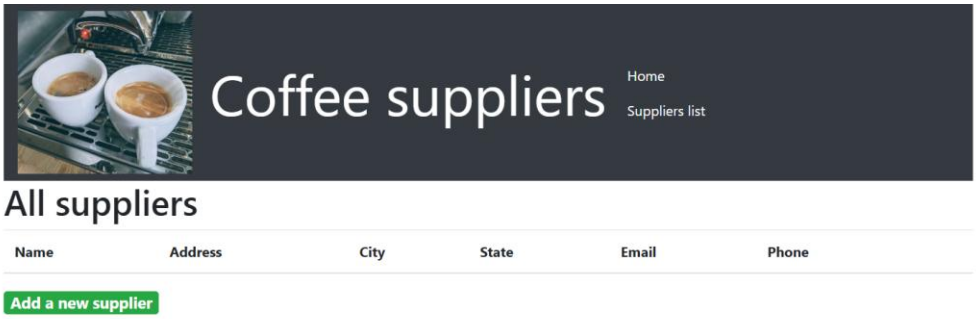
9. Open the existing coffee supplier application in a browser tab.
   - Return to the browser tab labeled **Your environments**, and navigate to the EC2 console.
   - Choose **Instances**.
   - Notice that three instances are running.
   - One instance is the VS Code IDE that you used in the previous task.
   - The other two instances (MysqlServerNode and AppServerNode) support the application that you will containerize in this lab.
   - Choose the **AppServerNode** instance, and copy the **Public IPv4 address** value.
   - Open a new browser tab and navigate to the IP address.
   - The coffee suppliers website displays.

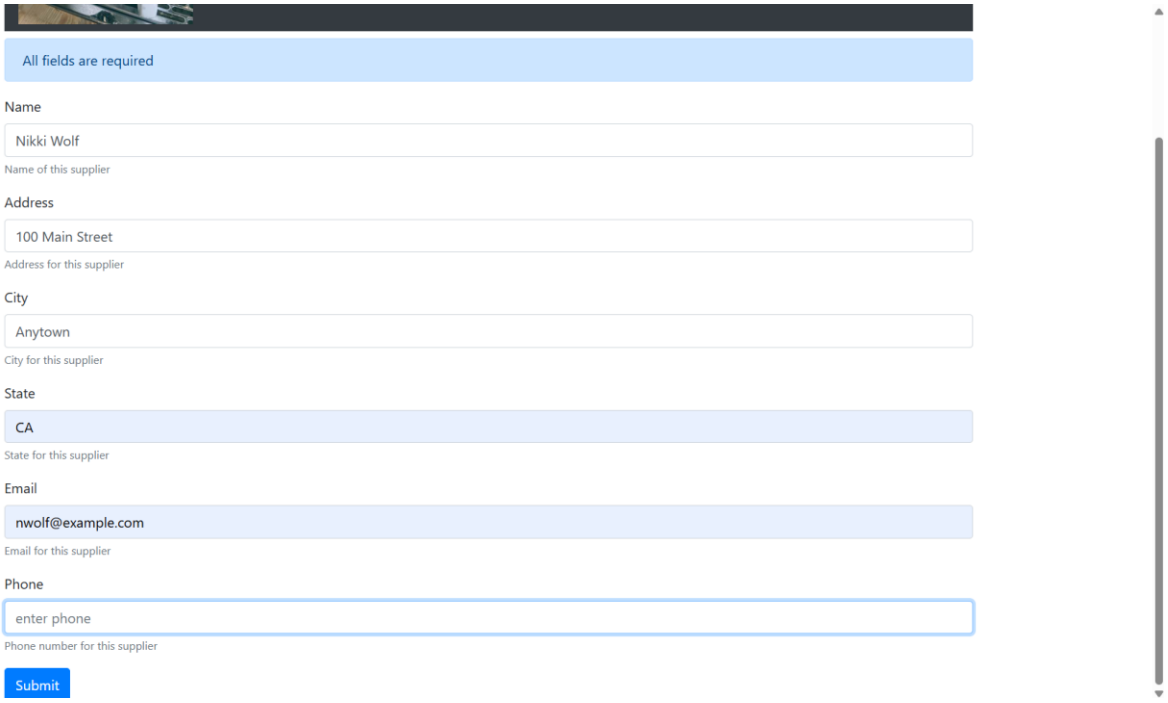



10. Test the web application functionality.

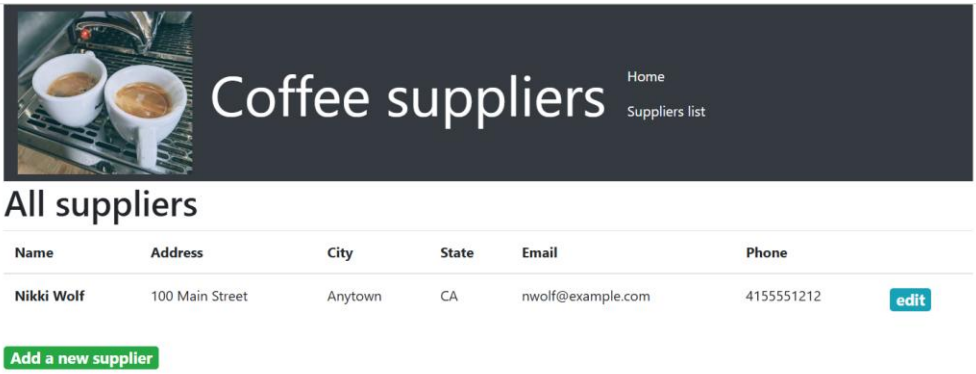
- Choose **List of suppliers** and then choose **Add a new supplier**.
- Fill in all of the fields with values. For example:
  - **Name**: Nikki Wolf
  - **Address**: 100 Main Street
  - **City**: Anytown
  - **State**: CA
  - **Email**: nwolf@example.com
  - **Phone**: 4155551212

- Choose **Submit**.
    - The **All suppliers** page displays and includes the record that you submitted.
- Choose **edit** and change the record (for example, modify the phone number).
- To save the change, choose **Submit**.



| Marwadi University<br>Marwadi Chandarana Group | Marwadi University<br>Faculty of Engineering and Technology<br>Department of Information and Communication Technology | |
|---|---|---|
| **Subject: Cloud Developing (01CT0720)** | **Aim: Migrate a Web Application to Docker Containers.** | |
| **Experiment No: 07** | **Date:** | **Enrolment No: 92200133030** |




**Task 3: Migrating the application to a Docker container**

In this task, you will migrate an application that is installed directly on the guest OS of an Ubuntu Linux EC2 instance to instead run in a Docker container. The Docker container is portable and could run on any OS that has the Docker engine installed.

For convenience, you will run the container on the same EC2 instance that hosts the VS Code IDE that you are using. You will use this IDE to build the Docker image and launch the Docker container.

13. Create a working directory for the node application, and move the source code into the new directory.
    - In the VS Code IDE, to create and navigate to a directory to store your Docker container code, run the following commands:
      ```
      mkdir containers
      cd containers
      ```
    - To create and navigate to a directory named **node_app** inside of the **containers** directory, run the following commands:
      ```
      mkdir node_app
      cd node_app
      ```
    - To move the code base, which you copied earlier, into the new **node_app** directory, run the following command:
      ```
      mv ~/environment/resources/codebase_partner
      ~/environment/containers/node_app
      ```

| | | |
|---|---|---|
| Marwadi University<br>Marwadi Chandarana Group | **Marwadi University**<br>**Faculty of Engineering and Technology**<br>**Department of Information and Communication Technology** | |
| **Subject: Cloud Developing (01CT0720)** | **Aim: Migrate a Web Application to Docker Containers.** | |
| **Experiment No: 07** | **Date:** | **Enrolment No: 92200133030** |

```
[ec2-user@ip-10-0-1-182 environment]$ mkdir containers
[ec2-user@ip-10-0-1-182 environment]$ cd containers
[ec2-user@ip-10-0-1-182 containers]$ mkdir node_app
[ec2-user@ip-10-0-1-182 containers]$ cd node_app
[ec2-user@ip-10-0-1-182 node_app]$ mv ~/environment/resources/codebase_partner ~/environment/containers/node_app
[ec2-user@ip-10-0-1-182 node_app]$
```

Ln 1, Col 41 Spaces: 4 UTF-8 LF Plain Text Layout: US

14. Create a Dockerfile.

- To create a new Dockerfile named **Dockerfile** in the **node_app/codebase_partner** directory, run the following command:

```
cd ~/environment/containers/node_app/codebase_partner
touch Dockerfile
```

- In the left navigation pane, browse to and open the empty Dockerfile that you just created.
- Copy and paste the following code into the Dockerfile:

```
FROM node:11-alpine
RUN mkdir -p /usr/src/app
WORKDIR /usr/src/app
COPY . .
RUN npm install
EXPOSE 3000
CMD ["npm", "run", "start"]
```

- Save the changes.

15. Build an image from the Dockerfile.

- In the VS Code IDE Bash terminal, run the following command:

```
docker build --tag node_app .
```

| Marwadi University<br>Marwadi Chandarana Group | **Marwadi University**<br>**Faculty of Engineering and Technology**<br>**Department of Information and Communication Technology** | |
|---|---|---|
| **Subject: Cloud Developing (01CT0720)** | **Aim: Migrate a Web Application to Docker Containers.** | |
| **Experiment No: 07** | **Date:** | **Enrolment No: 92200133030** |

```
[ec2-user@ip-10-0-1-182 node_app]$ mv ~/environment/resources/codebase_partner ~/environment/containers/node_app
[ec2-user@ip-10-0-1-182 node_app]$ cd ~/environment/containers/node_app/codebase_partner
touch Dockerfile
[ec2-user@ip-10-0-1-182 codebase_partner]$ docker build --tag node_app .
[+] Building 6.4s (10/10) FINISHED                                                                                  docker:default
 => [internal] load build definition from Dockerfile                                                                          0.0s
 => => transferring dockerfile: 225B                                                                                          0.0s
 => [internal] load metadata for docker.io/library/node:11-alpine                                                             0.3s
 => [internal] load .dockerignore                                                                                             0.0s
 => => transferring context: 2B                                                                                               0.0s
 => [1/5] FROM docker.io/library/node:11-alpine@sha256:8bb56bab197299c8ff820f1a55462890caf08f57ffe3b91f5fa6945a4d505932       2.1s
 => => resolve docker.io/library/node:11-alpine@sha256:8bb56bab197299c8ff820f1a55462890caf08f57ffe3b91f5fa6945a4d505932       0.0s
 => => extracting sha256:e7c96db7181be991f19a9fb6975cdbbd73c65f4a2681348e63a141a2192a5f10                                     0.1s
 => => sha256:8bb56bab197299c8ff820f1a55462890caf08f57ffe3b91f5fa6945a4d505932 1.42kB / 1.42kB                                0.0s
 => => sha256:914ff2c2145de019a19c080a9e42b5763c826194110ec8e02c8e92845799fba6 1.16kB / 1.16kB                                0.0s
 => => sha256:f18da2f58c3dabd7d06c02f4a76719ba0c786737aa83d4bfc8f1e5cb0800b96a 5.66kB / 5.66kB                                0.0s
 => => sha256:e7c96db7181be991f19a9fb6975cdbbd73c65f4a2681348e63a141a2192a5f10 2.76MB / 2.76MB                                0.1s
 => => sha256:0119aca4464934fdf40121363b1cf0537e464c07c790df99058ba587b14aaadd 21.66MB / 21.66MB                              0.3s
 => => sha256:40df19605a18a2dc4e3c6c0cb5d0a93fd2f7615e2a92eccd743698c24c23fe84 1.33MB / 1.33MB                                0.2s
 => => sha256:82194b8b4a64bd286671a1464d3fe319832aaa67c4e41c455fdfdd295ae7aa73 279B / 279B                                    0.1s
 => => extracting sha256:0119aca4464934fdf40121363b1cf0537e464c07c790df99058ba587b14aaadd                                     1.6s
 => => extracting sha256:40df19605a18a2dc4e3c6c0cb5d0a93fd2f7615e2a92eccd743698c24c23fe84                                     0.0s
 => => extracting sha256:82194b8b4a64bd286671a1464d3fe319832aaa67c4e41c455fdfdd295ae7aa73                                     0.0s
 => [internal] load build context                                                                                            0.0s
 => => transferring context: 1.25MB                                                                                           0.0s
 => [2/5] RUN mkdir -p /usr/src/app                                                                                           0.5s
 => [3/5] WORKDIR /usr/src/app                                                                                                0.0s
 => [4/5] COPY . .                                                                                                            0.1s
 => [5/5] RUN npm install                                                                                                     3.0s
 => exporting to image                                                                                                        0.2s
 => => exporting layers                                                                                                       0.2s
 => => writing image sha256:1f14352929e2b521d9b753eaf62b126c39276b60b671f41d01b2c1617d62aff9                                  0.0s
 => => naming to docker.io/library/node_app                                                                                   0.0s
[ec2-user@ip-10-0-1-182 codebase_partner]$
```

Ln 7, Col 28  Spaces: 4  UTF-8  LF  Docker  Layout: US

16. Verify that the Docker image was created.

- To list the Docker images that your Docker client is aware of, run the following command:

  **`docker images`**

```
[ec2-user@ip-10-0-1-182 codebase_partner]$ docker images
REPOSITORY   TAG       IMAGE ID       CREATED              SIZE
node_app     latest    1f14352929e2   About a minute ago   84.5MB
[ec2-user@ip-10-0-1-182 codebase_partner]$
```

17. Create and run a Docker container based on the Docker image.

- To create and run a Docker container from the image, run the following command:

  **`docker run -d --name node_app_1 -p 3000:3000 node_app`**

```
[ec2-user@ip-10-0-1-182 codebase_partner]$ docker run -d --name node_app_1 -p 3000:3000 node_app
9a8e54b20d06c569c31b0bc411e3d7b72da22dbf745a8b10f4669b332284af74
[ec2-user@ip-10-0-1-182 codebase_partner]$
```

Ln 7, Col 28  Spaces: 4  UTF-8  LF  Docker  Layout: US

- To view the Docker containers that are currently running on the host, run the following command:
  **`docker container ls`**

```
[ec2-user@ip-10-0-1-182 codebase_partner]$ docker container ls
CONTAINER ID   IMAGE      COMMAND                  CREATED              STATUS              PORTS                                       NAMES
9a8e54b20d06   node_app   "docker-entrypoint.s…"   About a minute ago   Up About a minute   0.0.0.0:3000->3000/tcp, :::3000->3000/tcp   node_app_1
[ec2-user@ip-10-0-1-182 codebase_partner]$
```

18. Verify that the node application is now running in the container.

- To check that the container is working on the correct port, run the following command:
  **`curl http://localhost:3000`**

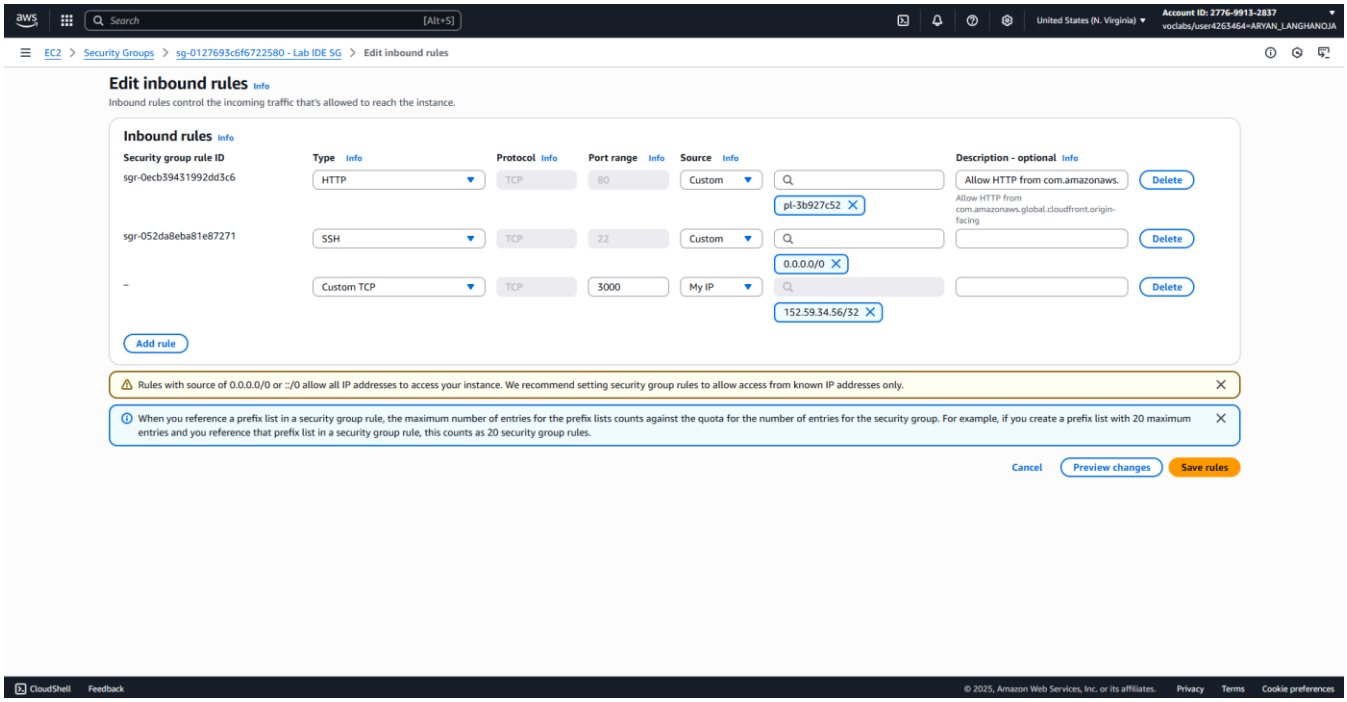
```
[ec2-user@ip-10-0-1-182 codebase_partner]$ curl http://localhost:3000
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <link rel="stylesheet" href="/css/bootstrap.min.css">
    <link rel="stylesheet" href="/css/base.css">
    <title>Coffee suppliers</title>
</head>
<body>

<div class="container">
    <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
        <img src="/img/espresso.jpg" width="200">
        <div><a class="navbar-brand page-title" href="/supplier">Coffee suppliers</a></div>
        <div class="collapse navbar-collapse" id="navbarSupportedContent">
            <ul class="navbar-nav mr-auto">
                <li class="nav-item active">
                    <a class="nav-link" href="/">Home</a>
                    <a class="nav-link" href="/suppliers">Suppliers list</a>
                </li>
            </ul>
        </div>
    </nav>    <div class="container">
        <h1>Welcome</h1>
        <p>Use this app to keep track of your coffee suppliers</p>
        <p><a href="/suppliers">List of suppliers</a></p>
    </div>
</div>
<script src="/js/jquery-3.6.0.min.js"></script>
<script src="/js/bootstrap.min.js"></script>
</body>
[ec2-user@ip-10-0-1-182 codebase_partner]$
```

19. Adjust the security group of the VS Code IDE instance to allow network traffic on port 3000 from your computer.

    **Note**: Because you are using the VS Code IDE instance to run the container, you must open TCP port 3000 for inbound traffic.

    - Return to the AWS Management Console browser tab, and navigate to the EC2 console.
    - Locate and select the **Lab IDE** instance.
    - Choose the **Security** tab, and choose the security group hyperlink.
    - Choose the **Inbound rules** tab, and choose **Edit inbound rules**.
    - Choose **Add rule** and configure the following:
        1. Type: **Custom TCP**
        2. Port range: **3000**
        3. Source: **My IP**
    - Choose **Save rules**.

| Marwadi University<br>Marwadi Chandarana Group | **Marwadi University**<br>**Faculty of Engineering and Technology**<br>**Department of Information and Communication Technology** | |
|---|---|---|
| **Subject: Cloud Developing (01CT0720)** | **Aim: Migrate a Web Application to Docker Containers.** | |
| **Experiment No: 07** | **Date:** | **Enrolment No: 92200133030** |




20. Access the web interface of the application, which is now running in a container.
    - In the EC2 console, choose **Instances** and choose the **Lab IDE** instance.
    - On the **Details** tab, copy the **Public IPv4 address** value.
    - Open a new browser tab. Paste the IP address into the address bar, and add :3000 at the end of the address.
    - The web application loads in the browser. You have seen this page before; however, you previously accessed the application that was running directly on the AppServerNode EC2 instance guest OS. This time you accessed the application running in a container on the VS Code IDE instance's Docker hypervisor.

21. Return to the VS Code IDE browser tab.
- Open the **config.js** file in the **containers/node_app/codebase_partner/app/config/** directory.
- Establish a terminal connection to the container to observe the settings.
    - To find the container ID, run the following command:

```
docker ps
```

    - To connect your terminal to the container, run the following commands, one at a time. Replace `<container-id>` with the actual container ID value that you just retrieved:

```
docker exec -ti <container-id> sh
whoami
```

- To observe the environment variables that are present in the node user's environment, run the following commands:

```
su node
env
```

```
[ec2-user@ip-10-0-1-182 codebase_partner]$ docker exec -ti 9a8e54b20d06 sh
whoami
/usr/src/app # su node
/usr/src/app $ env
USER=node
NODE_VERSION=11.15.0
HOSTNAME=9a8e54b20d06
YARN_VERSION=1.15.2
SHLVL=2
HOME=/home/node
LOGNAME=node
TERM=xterm
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
SHELL=/bin/sh
PWD=/usr/src/app
/usr/src/app $ exit
/usr/src/app # exit
ec2-user
[ec2-user@ip-10-0-1-182 codebase_partner]$
```

Notice that the **APP_DB_HOST** variable is not present.

- To disconnect from the container, run the following commands:

```
exit
exit
```

The first exit command makes you the root user again. The second command disconnects you from the container.

22. Stop and remove the container that has the database connectivity issue.

- To get the ID of the running container, run the following command:

```
docker ps
```

- Notice the name of the application that is returned in the **NAMES** column.
- To stop and remove the container, run the following command:

```
docker stop node_app_1 && docker rm node_app_1
```

- To verify that the application is no longer running, run the following curl command:

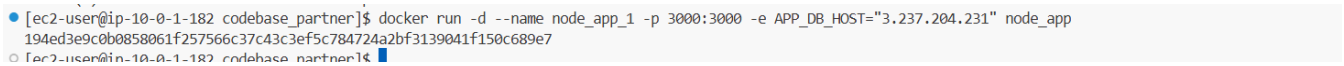
```
curl http://localhost:3000
```

- The output indicates a failure to connect to the application (*Connection refused*).

```
[ec2-user@ip-10-0-1-182 codebase_partner]$ docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED          STATUS          PORTS                                       NAMES
9a8e54b20d06   node_app   "docker-entrypoint.s…"   21 minutes ago   Up 21 minutes   0.0.0.0:3000->3000/tcp, :::3000->3000/tcp   node_app_1
[ec2-user@ip-10-0-1-182 codebase_partner]$ docker stop node_app_1 && docker rm node_app_1
node_app_1
node_app_1
[ec2-user@ip-10-0-1-182 codebase_partner]$ curl http://localhost:3000
curl: (7) Failed to connect to localhost port 3000 after 0 ms: Could not connect to server
[ec2-user@ip-10-0-1-182 codebase_partner]$
```

23. Launch a new container. This time, you will pass an environment variable to tell the node application the correct location of the database.
    - Return to the EC2 console, and copy the **Public IPv4 address** value of the **MysqlServerNode** EC2 instance.
    - Return to the VS Code IDE Bash terminal.
    - To run the application in a container and pass an environment variable to specify the database location, run the following command. Replace <ip-address> with the actual public IPv4 address of the MysqlServerNode EC2 instance:

```
docker run -d --name node_app_1 -p 3000:3000 -e APP_DB_HOST="<ip-address>" node_app
```




24. Verify that the database connection is now working.

- Try to access the web application again.
    - If you still have the page open, refresh the browser tab. Otherwise, to navigate to the application in a new browser tab, go to `http://<LabIDE-public-ip>:3000` (replace `<LabIDE-public-ip>` with the actual public IPv4 address of your VS Code IDE).
- The application is working. Choose **List of suppliers** to go to the `http://<LabIDE-public-ip>:3000/suppliers` page.
- The page displays the supplier entry that you created earlier. This indicates that your container is connecting to the MysqlServerNode EC2 instance where that data is stored.

**Task 4: Migrating the MySQL database to a Docker container**

In this task, you will work to migrate the MySQL database to a container as well. To accomplish this task, you will dump the latest data that is stored in the database and use that to seed a new MySQL database running in a new Docker container.

25. Create a mysqldump file from the data that is currently in the MySQL database.
- Return to the VS Code IDE, and close any file tabs that are open in the text editor.
- Choose **File** > **New File** and then paste the following code into the new file:

```
mysqldump -P 3306 -h  <mysql-host-ip-address> -u nodeapp -p --databases COFFEE > ../../my_sql.sql
```

- Next, go to the EC2 console and copy the **Public IPv4 address** value of the **MysqlServerNode** instance.
- Return to the text file in VS Code IDE and replace <mysql-host-ip-address> in the code with the IP address that you copied.

| Marwadi University<br>Marwadi Chandarana Group | **Marwadi University**<br>**Faculty of Engineering and Technology**<br>**Department of Information and Communication Technology** | |
|---|---|---|
| **Subject: Cloud Developing (01CT0720)** | **Aim: Migrate a Web Application to Docker Containers.** | |
| **Experiment No: 07** | **Date:** | **Enrolment No: 92200133030** |

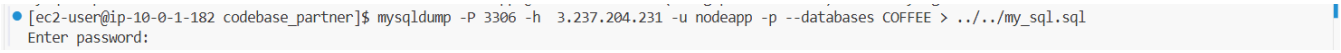
- In the terminal, to ensure that you are in the correct directory, run the following command:
  **`cd /home/ec2-user/environment/containers/node_app/codebase_partner`**
- Finally, copy the command that you created in the text editor into the terminal and run the command.

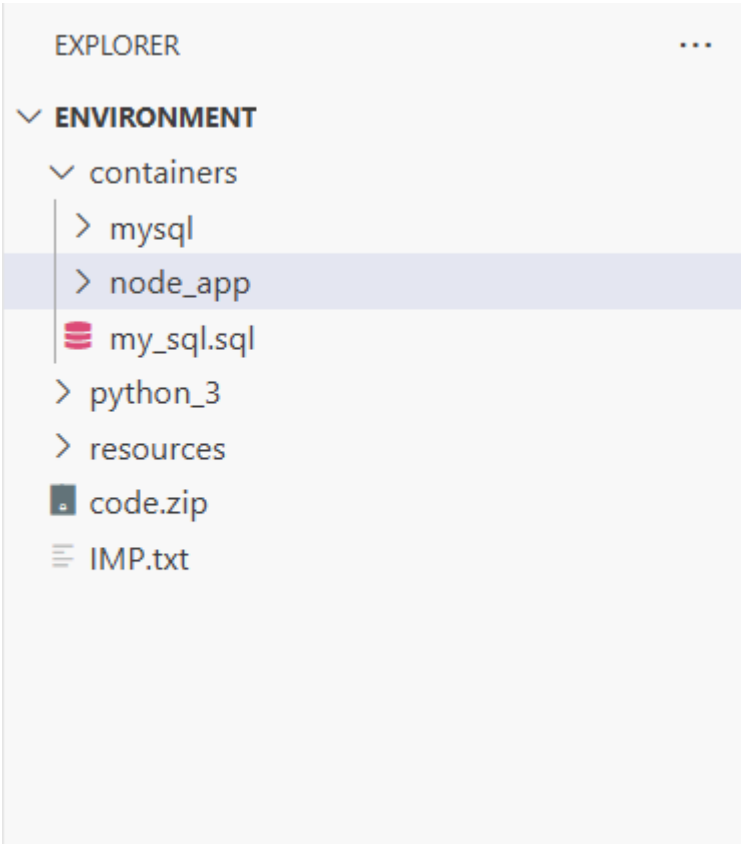
Your command will look similar to the following example, but your IP address will be different:

```
mysqldump -P 3306 -h 100.27.45.2 -u nodeapp -p --databases COFFEE >
../../my_sql.sql
```

- The mysqldump utility prompts you for a password. Enter the following password:
  **`coffee`**

If successful, the terminal does not show any output. However, in the left navigation panel, notice that the **`mysql.sql`** file now appears in the **containers** directory.



26. In the terminal, to create a directory to store your mysql container code and navigate into the directory, run the following commands:

```
cd /home/ec2-user/environment/containers
mkdir mysql
cd mysql
```

```
[ec2-user@ip-10-0-1-182 codebase_partner]$ cd /home/ec2-user/environment/containers
[ec2-user@ip-10-0-1-182 containers]$ mkdir mysql
[ec2-user@ip-10-0-1-182 containers]$ cd mysql
```

27. Create a Dockerfile.

- To create a new Dockerfile, run the following command:

```
touch Dockerfile
```

- To move the sqldump file into the new **mysql** directory, run the following command:

```
mv ../my_sql.sql .
```

- Open the empty Dockerfile (in **containers/mysql/**) and then copy and paste the following code into the file:

```
FROM mysql:8.0.23
COPY ./my_sql.sql /
EXPOSE 3306
```

- Save the changes.

28. Attempt to free up some disk space on the VS Code IDE instance by removing unneeded files.
    - Run the following command:

```
docker rmi -f $(docker image ls -a -q)
```

    - Finally, run the following command:

```
sudo docker image prune -f && sudo docker container prune -f
```

```
[ec2-user@ip-10-0-1-182 mysql]$ docker rmi -f $(docker image ls -a -q)
Error response from daemon: conflict: unable to delete 1f14352929e2 (cannot be forced) - image is being used by running container 194ed3e9c0b0
[ec2-user@ip-10-0-1-182 mysql]$ sudo docker image prune -f && sudo docker container prune -f
Total reclaimed space: 0B
Total reclaimed space: 0B
[ec2-user@ip-10-0-1-182 mysql]$
```

29. To build an image from the Dockerfile, run the following command:

```
docker build --tag mysql_server .
```

| Marwadi University<br>Marwadi Chandarana Group | **Marwadi University**<br>**Faculty of Engineering and Technology**<br>**Department of Information and Communication Technology** | |
|---|---|---|
| **Subject: Cloud Developing (01CT0720)** | **Aim: Migrate a Web Application to Docker Containers.** | |
| **Experiment No: 07** | **Date:** | **Enrolment No: 92200133030** |

```
[ec2-user@ip-10-0-1-182 mysql]$ docker build --tag mysql_server .
[+] Building 8.9s (7/7) FINISHED                                                                                    docker:default
 => [internal] load build definition from Dockerfile                                                                        0.0s
 => => transferring dockerfile: 141B                                                                                        0.0s
 => [internal] load metadata for docker.io/library/mysql:8.0.23                                                             0.3s
 => [internal] load .dockerignore                                                                                           0.0s
 => => transferring context: 2B                                                                                             0.0s
 => [internal] load build context                                                                                           0.0s
 => => transferring context: 2.55kB                                                                                         0.0s
 => [1/2] FROM docker.io/library/mysql:8.0.23@sha256:6e0014cdd88092545557dee5e9eb7e1a3c84c9a14ad2418d5f2231e930967a38       7.3s
 => => resolve docker.io/library/mysql:8.0.23@sha256:6e0014cdd88092545557dee5e9eb7e1a3c84c9a14ad2418d5f2231e930967a38       0.0s
 => => sha256:6e0014cdd88092545557dee5e9eb7e1a3c84c9a14ad2418d5f2231e930967a38 320B / 320B                                  0.0s
 => => sha256:355617769102e9d2ebb7d5879263a12d230badb7271c91748b2c7b0ac6971083 2.83kB / 2.83kB                              0.0s
 => => sha256:181cefd361ceae3da481ffcd156bd5b6e22f32a9345167199fe4773d70a6f133 1.42MB / 1.42MB                              0.2s
 => => sha256:f7ec5a41d630a33a2d1db59b95d89d93de7ae5a619a3a8571b78457e48266eba 27.14MB / 27.14MB                            0.3s
 => => sha256:cbe8815cbea8fb86ce7d3169a82d05301e7dfe1a8d4228941f23f4f115a887f2 7.10kB / 7.10kB                              0.0s
 => => sha256:9444bb562699de912e7afcf6409615f9c0033118cc3aa35831d937a8d15e851c 1.73kB / 1.73kB                              0.1s
 => => sha256:6a4207b96940576ff42802b9c440bed0f1b5ef419c5a0c2576477776f217ef35 4.18MB / 4.18MB                              0.2s
 => => sha256:15f235e0d7eefe6c5153946a57daebaf2d0744fb1d0676642f269554fc68ba2a 13.45MB / 13.45MB                            0.6s
 => => sha256:8a2090759d8af87290c70a8a9cef4e7e894b800d55d069b8c4cf0bcd3a5d9f17 149B / 149B                                  0.3s
 => => sha256:d870539cd9db00c747403f1d6c494a807052bce070f4ef243954b804a1ebb652 2.40kB / 2.40kB                              0.3s
 => => extracting sha256:f7ec5a41d630a33a2d1db59b95d89d93de7ae5a619a3a8571b78457e48266eba                                    1.7s
 => => sha256:5726073179b6fb503d82a3b6cf6c7fe83081484ba600364a37b58105dec30677 225B / 225B                                  0.4s
 => => sha256:eadfac8b2520b41436dd678f1f822ffcc0b416ce5d481a9070c4c4531b3d3efd 113.13MB / 113.13MB                          2.5s
 => => sha256:f5936a8c3f2baf76cbebb0eaf333510aa2f52c6e825cf216638ac4b827ccc1d3 845B / 845B                                  0.5s
 => => sha256:cca8ee89e625e42d7fcd5fd01d23c53c630d3aff4511b9a5f386713dc2e89ce4 5.52kB / 5.52kB                              0.6s
 => => sha256:6c79df02586a3bcf87a8109d34a6d6b32593583b49b5ae106cf89b70b04742c6 119B / 119B                                  0.7s
 => => extracting sha256:9444bb562699de912e7afcf6409615f9c0033118cc3aa35831d937a8d15e851c                                   0.0s
 => => extracting sha256:6a4207b96940576ff42802b9c440bed0f1b5ef419c5a0c2576477776f217ef35                                   0.1s
 => => extracting sha256:181cefd361ceae3da481ffcd156bd5b6e22f32a9345167199fe4773d70a6f133                                   0.1s
 => => extracting sha256:8a2090759d8af87290c70a8a9cef4e7e894b800d55d069b8c4cf0bcd3a5d9f17                                   0.0s
 => => extracting sha256:15f235e0d7eefe6c5153946a57daebaf2d0744fb1d0676642f269554fc68ba2a                                   0.8s
 => => extracting sha256:d870539cd9db00c747403f1d6c494a807052bce070f4ef243954b804a1ebb652                                   0.0s
 => => extracting sha256:5726073179b6fb503d82a3b6cf6c7fe83081484ba600364a37b58105dec30677                                   0.0s
 => => extracting sha256:eadfac8b2520b41436dd678f1f822ffcc0b416ce5d481a9070c4c4531b3d3efd                                   3.5s
 => => extracting sha256:f5936a8c3f2baf76cbebb0eaf333510aa2f52c6e825cf216638ac4b827ccc1d3                                   0.0s
 => => extracting sha256:cca8ee89e625e42d7fcd5fd01d23c53c630d3aff4511b9a5f386713dc2e89ce4                                   0.0s
 => => extracting sha256:6c79df02586a3bcf87a8109d34a6d6b32593583b49b5ae106cf89b70b04742c6                                   0.0s
 => [2/2] COPY ./my sql.sql /                                                                                               1.1s
```

30. Verify that the Docker image was created.

- To list the Docker images that your Docker client is aware of, run the following command:
  **`docker images`**

```
[ec2-user@ip-10-0-1-182 mysql]$ docker images
REPOSITORY     TAG       IMAGE ID       CREATED              SIZE
mysql_server   latest    ed14cf08e6fb   About a minute ago   546MB
node_app       latest    1f14352929e2   51 minutes ago       84.5MB
[ec2-user@ip-10-0-1-182 mysql]$
```

31. Create and run a Docker container based on the Docker image.

- To create and run a Docker container from the image, run the following command:
  `docker run --name mysql_1 -p 3306:3306 -e MYSQL_ROOT_PASSWORD=rootpw -d mysql_server`

```
[ec2-user@ip-10-0-1-182 mysql]$ docker run --name mysql_1 -p 3306:3306 -e MYSQL_ROOT_PASSWORD=rootpw -d mysql_server
7e38f9fc95ecf605c3e49bb5d2fd8da9b21faf134da7b1712aa20ecf48efc433
[ec2-user@ip-10-0-1-182 mysql]$
```

- The terminal returns the container ID for the container.
- To view the Docker containers that are currently running on the host, run the following command:
  **`docker container ls`**
- Two containers are now running. One hosts the node application, and the other hosts the MySQL database.

| Marwadi University<br>Marwadi Chandarana Group | **Marwadi University**<br>**Faculty of Engineering and Technology**<br>**Department of Information and Communication Technology** | |
|---|---|---|
| **Subject: Cloud Developing (01CT0720)** | **Aim: Migrate a Web Application to Docker Containers.** | |
| **Experiment No: 07** | **Date:** | **Enrolment No: 92200133030** |

```
[ec2-user@ip-10-0-1-182 mysql]$ docker container ls
CONTAINER ID   IMAGE          COMMAND                  CREATED              STATUS              PORTS                                                NAMES
7e38f9fc95ec   mysql_server   "docker-entrypoint.s…"   About a minute ago   Up About a minute   0.0.0.0:3306->3306/tcp, :::3306->3306/tcp, 33060/tcp   mysql_1
194ed3e9c0b0   node_app       "docker-entrypoint.s…"   25 minutes ago       Up 25 minutes       0.0.0.0:3000->3000/tcp, :::3000->3000/tcp              node_app_
1
[ec2-user@ip-10-0-1-182 mysql]$
```

32. Import the data into the MySQL database and define a database user.
- Run the following command:
- ⚠️ Note that space is *not* included between -p and rootpw in the command.

```
sed -i '1d' my_sql.sql
docker exec -i mysql_1 mysql -u root -prootpw < my_sql.sql
```

- Ignore the warning about using a password on the command line interface being insecure.

33. To create a database user for the node application to use, run the following command:

```
docker exec -i mysql_1 mysql -u root  -prootpw -e "CREATE USER 'nodeapp'
IDENTIFIED WITH mysql_native_password BY 'coffee'; GRANT all privileges on
*.* to 'nodeapp'@'%';"
```

## Task 5: Testing the MySQL container with the node application

34. To stop and remove the node application server container, run the following command:
    **`docker stop node_app_1 && docker rm node_app_1`**

```
[ec2-user@ip-10-0-1-182 mysql]$ docker stop node_app_1 && docker rm node_app_1
node_app_1
node_app_1
[ec2-user@ip-10-0-1-182 mysql]$
```

35. Discover the network connectivity information.
- To find the IPv4 address of the mysql_1 container on the network, run the following command:
  **`docker inspect <CONTAINER ID>`**

| Marwadi University<br>Marwadi Chandarana Group | Marwadi University<br>Faculty of Engineering and Technology<br>Department of Information and Communication Technology | |
|---|---|---|
| **Subject: Cloud Developing (01CT0720)** | **Aim: Migrate a Web Application to Docker Containers.** | |
| **Experiment No: 07** | **Date:** | **Enrolment No: 92200133030** |

```
[ec2-user@ip-10-0-1-182 mysql]$ docker inspect 7e38f9fc95ec
[
    {
        "Id": "7e38f9fc95ecf605c3e49bb5d2fd8da9b21faf134da7b1712aa20ecf48efc433",
        "Created": "2025-08-13T04:57:54.621125671Z",
        "Path": "docker-entrypoint.sh",
        "Args": [
            "mysqld"
        ],
        "State": {
            "Status": "running",
            "Running": true,
            "Paused": false,
            "Restarting": false,
            "OOMKilled": false,
            "Dead": false,
            "Pid": 53180,
            "ExitCode": 0,
            "Error": "",
            "StartedAt": "2025-08-13T04:57:55.036276745Z",
            "FinishedAt": "0001-01-01T00:00:00Z"
        },
        "Image": "sha256:ed14cf08e6fbda44c350162ded2061b080eab683cab01520de204b47ba17296a",
        "ResolvConfPath": "/var/lib/docker/containers/7e38f9fc95ecf605c3e49bb5d2fd8da9b21faf134da7b1712aa20ecf48efc433/resolv.conf",
        "HostnamePath": "/var/lib/docker/containers/7e38f9fc95ecf605c3e49bb5d2fd8da9b21faf134da7b1712aa20ecf48efc433/hostname",
        "HostsPath": "/var/lib/docker/containers/7e38f9fc95ecf605c3e49bb5d2fd8da9b21faf134da7b1712aa20ecf48efc433/hosts",
        "LogPath": "/var/lib/docker/containers/7e38f9fc95ecf605c3e49bb5d2fd8da9b21faf134da7b1712aa20ecf48efc433/7e38f9fc95ecf605c3e49bb5d2fd8da9b21faf134da7b1712aa20ecf48efc433-json.log",
        "Name": "/mysql_1",
        "RestartCount": 0,
        "Driver": "overlay2",
        "Platform": "linux",
        "MountLabel": "",
        "ProcessLabel": "",
        "AppArmorProfile": "",
        "ExecIDs": null,
        "HostConfig": {
            "Binds": null,
            "ContainerIDFile": "",
            "LogConfig": {
```

36. Start a new node application Docker container.

- In this step, you run the Docker command to start a new container from the **node_app** Docker image. However, this time you pass the **APP_DB_HOST** environment variable to the container environment.
- Run the following command. Replace <ip-address> with the actual IPv4 address value that you just discovered. You do *not* need to surround the IP address in quotes.

```
docker run -d --name node_app_1 -p 3000:3000 -e APP_DB_HOST=<ip-address> node_app
```

```
[ec2-user@ip-10-0-1-182 mysql]$ docker run -d --name node_app_1 -p 3000:3000 -e APP_DB_HOST=172.17.0.3 node_app
a73ad036d80f3f44a897c4addcb08c6e71590ac97b251d93afccc588306df50e
[ec2-user@ip-10-0-1-182 mysql]$
```

37. To verify that both containers are running again, run the following command:

```
docker ps
```

```
[ec2-user@ip-10-0-1-182 mysql]$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED              STATUS              PORTS                                                  NAMES
a73ad036d80f   node_app       "docker-entrypoint.s…"   About a minute ago   Up About a minute   0.0.0.0:3000->3000/tcp, :::3000->3000/tcp              node_app_
1
7e38f9fc95ec   mysql_server   "docker-entrypoint.s…"   21 minutes ago       Up 21 minutes       0.0.0.0:3306->3306/tcp, :::3306->3306/tcp, 33060/tcp   mysql_1
[ec2-user@ip-10-0-1-182 mysql]$
```

| Marwadi University<br>Marwadi Chandarana Group | **Marwadi University**<br>**Faculty of Engineering and Technology**<br>**Department of Information and Communication Technology** | |
|---|---|---|
| **Subject: Cloud Developing (01CT0720)** | **Aim: Migrate a Web Application to Docker Containers.** | |
| **Experiment No: 07** | **Date:** | **Enrolment No: 92200133030** |

## Task 6: Adding the Docker images to Amazon ECR

38. Authorize your Docker client to connect to the Amazon ECR service.

- Discover your AWS account ID.
- In the AWS Management Console, in the upper-right corner, choose your user name. Your user name begins with voclab/user.
- Copy the My Account value from the menu. This is your AWS account ID.
- Next, return to the VS Code IDE Bash terminal.
- To authorize your VS Code IDE Docker client, run the following command. Replace <account-id> with the actual account ID that you just found:
- aws ecr get-login-password \
- --region us-east-1 | docker login --username AWS \
- --password-stdin <account-id>.dkr.ecr.us-east-1.amazonaws.com
- A message indicates that the login succeeded.

```
[ec2-user@ip-10-0-1-182 mysql]$ aws ecr get-login-password \
--region us-east-1 | docker login --username AWS \
--password-stdin 277699132837.dkr.ecr.us-east-1.amazonaws.com
WARNING! Your password will be stored unencrypted in /home/ec2-user/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
```

39. To create the repository, run the following command:
    **`aws ecr create-repository --repository-name node-app`**

```
[ec2-user@ip-10-0-1-182 mysql]$ aws ecr create-repository --repository-name node-app
{
    "repository": {
        "repositoryArn": "arn:aws:ecr:us-east-1:277699132837:repository/node-app",
        "registryId": "277699132837",
        "repositoryName": "node-app",
        "repositoryUri": "277699132837.dkr.ecr.us-east-1.amazonaws.com/node-app",
        "createdAt": "2025-08-13T05:23:46.764000+00:00",
        "imageTagMutability": "MUTABLE",
        "imageScanningConfiguration": {
            "scanOnPush": false
        },
        "encryptionConfiguration": {
            "encryptionType": "AES256"
```

40. Tag the Docker image.
41. In this step, you will tag the image with your unique **registryId** value to make it easier to manage and keep track of this image.
42. Run the following command. Replace <registry-id> with your actual registry ID number.
43. docker tag node_app:latest <registry-id>.dkr.ecr.us-east-1.amazonaws.com/node-app:latest
44. The command does not provide a response.

```
[ec2-user@ip-10-0-1-182 mysql]$ docker tag node_app:latest 277699132837.dkr.ecr.us-east-1.amazonaws.com/node-app:latest
[ec2-user@ip-10-0-1-182 mysql]$
```

| | | |
|---|---|---|
| Marwadi University<br>Marwadi Chandarana Group | **Marwadi University**<br>**Faculty of Engineering and Technology**<br>**Department of Information and Communication Technology** | |
| **Subject: Cloud Developing (01CT0720)** | **Aim: Migrate a Web Application to Docker Containers.** | |
| **Experiment No: 07** | **Date:** | **Enrolment No: 92200133030** |

**Push the Docker image to the Amazon ECR repository.**

- **To push your image to Amazon ECR, run the following command. Replace <registry-id> with your actual registry ID number:**

**docker push <registry-id>.dkr.ecr.us-east-1.amazonaws.com/node-app:latest**

- **The output is similar to the following:**

```
[ec2-user@ip-10-0-1-182 mysql]$ docker push 277699132837.dkr.ecr.us-east-1.amazonaws.com/node-app:latest
The push refers to repository [277699132837.dkr.ecr.us-east-1.amazonaws.com/node-app]
d4ae98dfcd5d: Pushed
68a0e64d91ec: Pushed
5f70bf18a086: Pushed
08e123598c01: Pushed
d81d715330b7: Pushed
1dc7f3bb09a4: Pushed
dcaceb729824: Pushed
f1b5933fe4b5: Pushed
latest: digest: sha256:173ca70c43caf08c2d2b307bf1bff2fccc8db6c33cf8495b663d069028c9f8f2 size: 1992
[ec2-user@ip-10-0-1-182 mysql]$
```

**To confirm that the node-app image is now stored in Amazon ECR, run the following aws ecr list-images command:**

**aws ecr list-images --repository-name node-app**

**Conclusion:-**

In this lab,

- I learned how to migrate a web application and MySQL database to Docker containers.
- I created Dockerfiles for both the Node.js app and MySQL, built images, and ran containers.
- I tested the application with database connectivity using environment variables.
- I also used commands to export the MySQL database, import it into a MySQL container, and connect it with the Node.js container. Finally,
- I pushed the Docker image to Amazon ECR by creating a repository, tagging the image, and uploading it.

Commands Used :-

- wget, unzip – for downloading and extracting code.
- chmod +x ./resources/setup.sh && ./resources/setup.sh – for AWS CLI upgrade.
- docker build --tag <image_name> . – building Docker images.
- docker run -d --name <container_name> -p <host_port>:<container_port> <image_name> – running containers.
- docker ps, docker images – verifying running containers and available images.
- docker exec -ti <container_id> sh – accessing container shell.
- mysqldump – exporting MySQL database.
- docker rmi, docker image prune, docker container prune – cleaning up resources.
- aws ecr create-repository, docker tag, docker push – integrating with Amazon ECR.

| | | |
|---|---|---|
| Marwadi University<br>Marwadi Chandarana Group | **Marwadi University**<br>**Faculty of Engineering and Technology**<br>**Department of Information and Communication Technology** | |
| **Subject: Cloud Developing (01CT0720)** | **Aim: Migrate a Web Application to Docker Containers.** | |
| **Experiment No: 07** | **Date:** | **Enrolment No: 92200133030** |

**Result :-**

| Total score | 35/35 |
|---|---|
| [TASK 2A] get_all_products function was created | 5/5 |
| [TASK 2B] get_all_products returns DynamoDB data | 5/5 |
| [TASK 3] /products GET method invokes get_all_products | 5/5 |
| [TASK 3B] /on_offer GET method invokes get_all_products | 5/5 |
| [TASK 3C] /on_offer GET method mapping template configured | 5/5 |
| [TASK 4] create_report function was created | 5/5 |
| [TASK 5] POST method invokes create_report | 5/5 |