

Develop and Deploy Web Application for Amazon Native Database CRUD Operation

(LAB-M07-01)

Version Control	
Document	Develop and Deploy Web Application for Database CRUD Operation
Owner	Ahmad Majeed Zahoory
Version	2.2
Last Change	23 rd May 2024
Description of Change	Task steps updated

Lab duration: 60 minutes

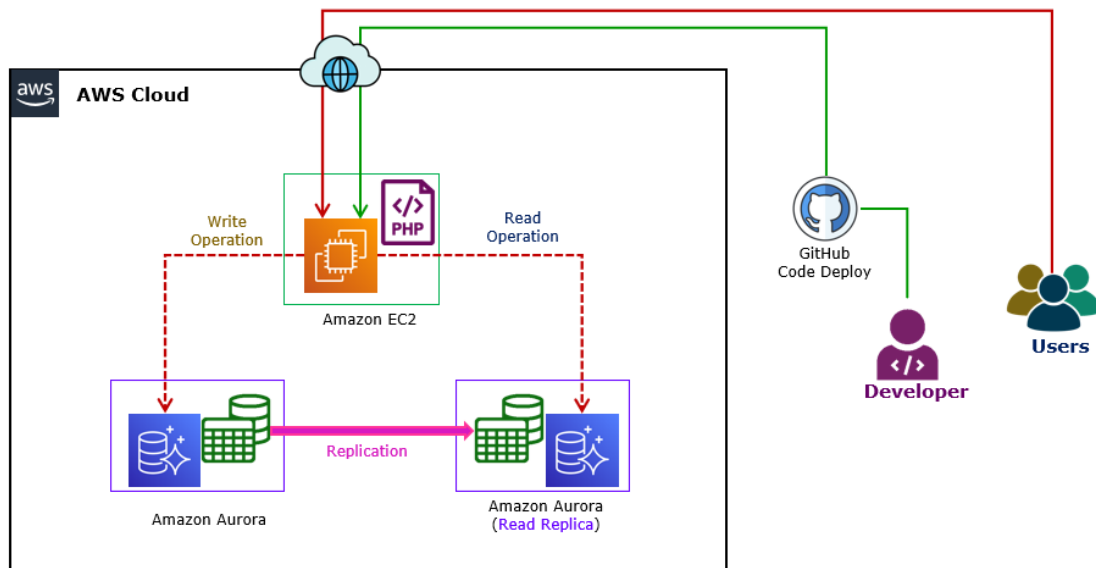
Lab scenario

You're preparing to host a Products web application in AWS that uses to store product details in database. As a development group, your team has decided to host a web application in AWS virtual machine and database in AWS RDS. For performance management development team needs to develop the code to write the data in Write SQL server and read the data from Read SQL Server. You also want to explore the AWS Native database for your services.

Objectives

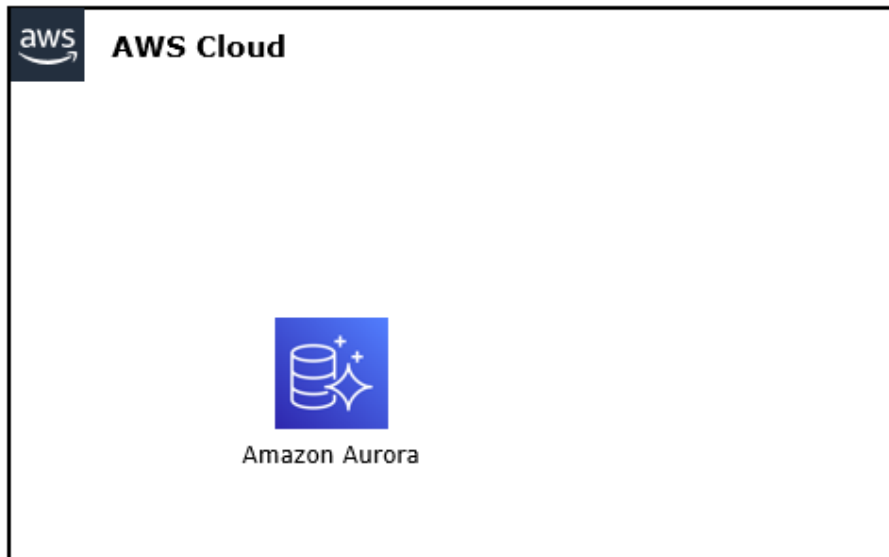
After you complete this lab, you will be able to:

- Create virtual machine using the AWS UI.
- Create SQL database using the AWS UI.
- Perform the Read and Write operation from 2 different SQL Servers.
- Develop the Php Code to perform Read and Write operation from 2 different SQL Servers.



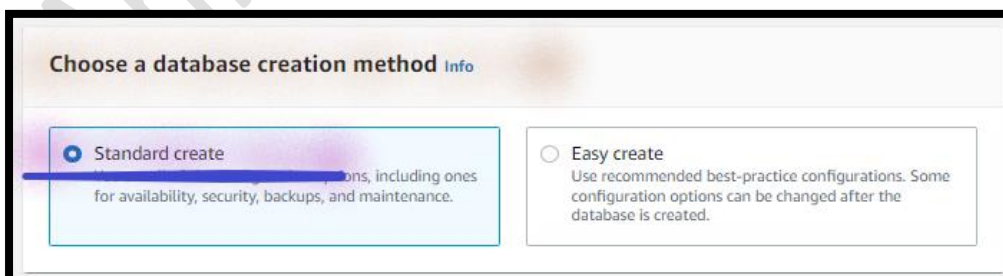
Task 1: Create a Database

In this task, you will create relational (amazon aurora) database.



Step 1: Create an Amazon RDS Instance

1. In the **AWS Management Console**, on the **Services** menu, search and select **RDS**.
2. Choose the **YOUR ALLOCATED REGION**, region list to the right of your account information on the navigation bar.
3. Select **Databases**.
 - a. Select **Create database** and **Configure**:
 - i. In the **Choose a database creation method** section:
 - a) Select **Standard create**.



ii. In the **Engine options** section:

- a) **Engine type:** Select **Amazon Aurora (MySQL-Compatible)**.
- b) **Version:** Dropdown and select **Aurora MySQL [Latest version 8.x]**.

Engine options

Engine type [Info](#)

☒ Aurora (MySQL Compatible)

☐ Aurora (PostgreSQL Compatible)

☐ MySQL

☐ MariaDB

☐ PostgreSQL

☐ Oracle

☐ Microsoft SQL Server

☐ IBM Db2

Engine Version

Aurora MySQL 3.05.2 (compatible with MySQL 8.0.32) - default for major version 8.0

Warning: Parallel query is off by default. To enable it, use a DB instance parameter group with the `aurora_parallel_query` parameter enabled. [Learn more](#)

iii. In the **Templates** section:

- a) Select **Dev/ Test**.

Templates
Choose a sample template to meet your use case.

☐ **Production**
Use defaults for high availability and fast, consistent performance.

☒ **Dev/Test**
ended for development use outside of a production environment.

iv. In the **Settings** section:

- a) **DB instance identifier**: Write **inventory-db**.
- b) **Expand** the **Credentials Settings**.
 - i. **Master username**: Write **master**.
 - ii. **Credentials management**: Select **Self managed**.
 - iii. **Master password**: Write **lab-password**.
 - iv. **Confirm password**: Write **lab-password**.

Note: This is the **username** and **password** you use to login and access your Aurora database instance.

▼ **Credentials Settings**

Master username [Info](#)
Type a login ID for the master user of your DB instance.

master

1 to 32 alphanumeric characters. The first character must be a letter.

Credentials management
You can use AWS Secrets Manager or manage your master user credentials.

☐ **Managed in AWS Secrets Manager - most secure**
RDS generates a password for you and manages it throughout its lifecycle using AWS Secrets Manager.

☒ **Self managed**
Create your own password or have RDS create a password that you manage.

☐ **Auto generate password**
Amazon RDS can generate a password for you, or you can specify your own password.

Master password [Info](#)

Minimum constraints: At least 8 printable ASCII characters. Can't contain any of the following symbols: / ' " @

Confirm master password [Info](#)

- v. In the **Cloud Storage configuration** section:

Note: Leave the details as default.

- vi. In the **Instance configuration** section:

- Enable Include previous generation classes.**
- Select the **Burstable classes (includes t classes).**
- Dropdown and select **db.t3.medium** [*This instance type attract charges*].

Instance configuration

The DB instance configuration options below are limited to those supported by the engine that you selected above.

DB instance class [Info](#)

▼ Hide filters

☒ Include previous generation classes

☐ Serverless v2

☐ Memory optimized classes (includes r classes)

☒ Burstable classes (includes t classes)

db.t3.medium
2 vCPUs 4 GiB RAM Network: 2,085 Mbps

- vii. In the **Availability & durability** section:

- Multi-AZ deployment:** Select **Don't create an Aurora Replica.**

Availability & durability

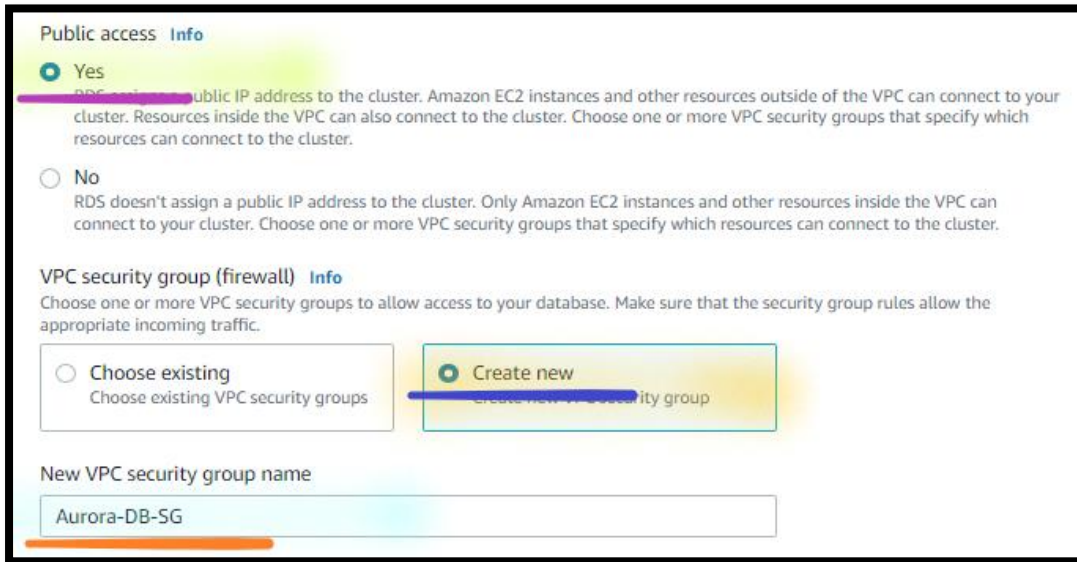
Multi-AZ deployment [Info](#)

☐ Create an Aurora Replica or Reader node in a different AZ (recommended for scaled availability)
Creates an Aurora Replica for fast failover and high availability.

☒ Don't create an Aurora Replica

viii. In the **Connectivity** section:

- a) **Public access:** Select **Yes**.
- b) **VPC security groups:** Select **Create new**.
- c) **New VPC security group name:** Write **Aurora-DB-SG**.



Public access Info

☒ **Yes**
RDS assigns a public IP address to the cluster. Amazon EC2 instances and other resources outside of the VPC can connect to your cluster. Resources inside the VPC can also connect to the cluster. Choose one or more VPC security groups that specify which resources can connect to the cluster.

☐ **No**
RDS doesn't assign a public IP address to the cluster. Only Amazon EC2 instances and other resources inside the VPC can connect to your cluster. Choose one or more VPC security groups that specify which resources can connect to the cluster.

VPC security group (firewall) Info
Choose one or more VPC security groups to allow access to your database. Make sure that the security group rules allow the appropriate incoming traffic.

☐ **Choose existing**
Choose existing VPC security groups

☒ **Create new**
Create new VPC security group

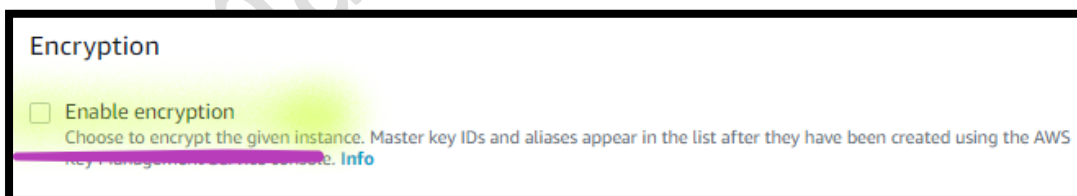
New VPC security group name

Aurora-DB-SG

Note: Leave the other details as default.

ix. In the **Additional configuration** section (*at the bottom of the page*):

- a) **Encryption:** **Unselect** the **Enable encryption**.



Encryption

☐ **Enable encryption**
Choose to encrypt the given instance. Master key IDs and aliases appear in the list after they have been created using the AWS Key Management Service console. Info

- b) **Deletion protection:** **Unselect** the **Enable deletion protection**.



Deletion protection

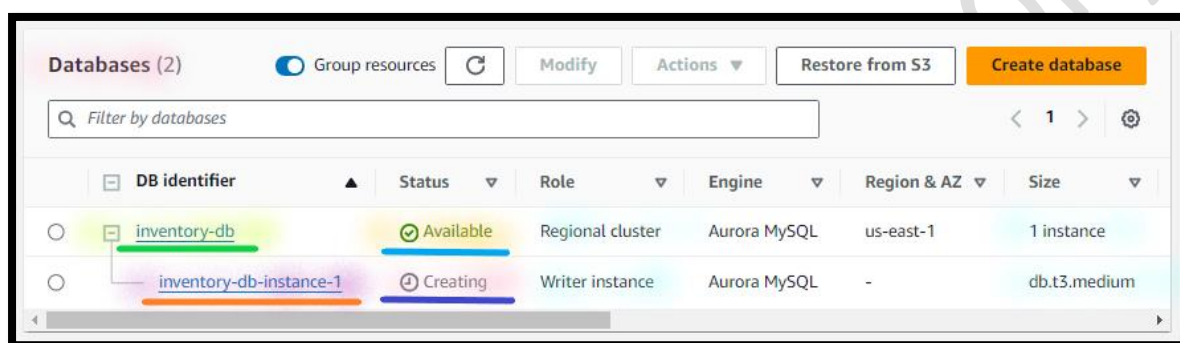
☐ **Enable deletion protection**
Protects the database from being deleted accidentally. While this option is enabled, you can't delete the database.

Note: Leave the other details as default.

- b. Select **Create database**.

Note: You can see the "**Creating database inventory-db**" message.

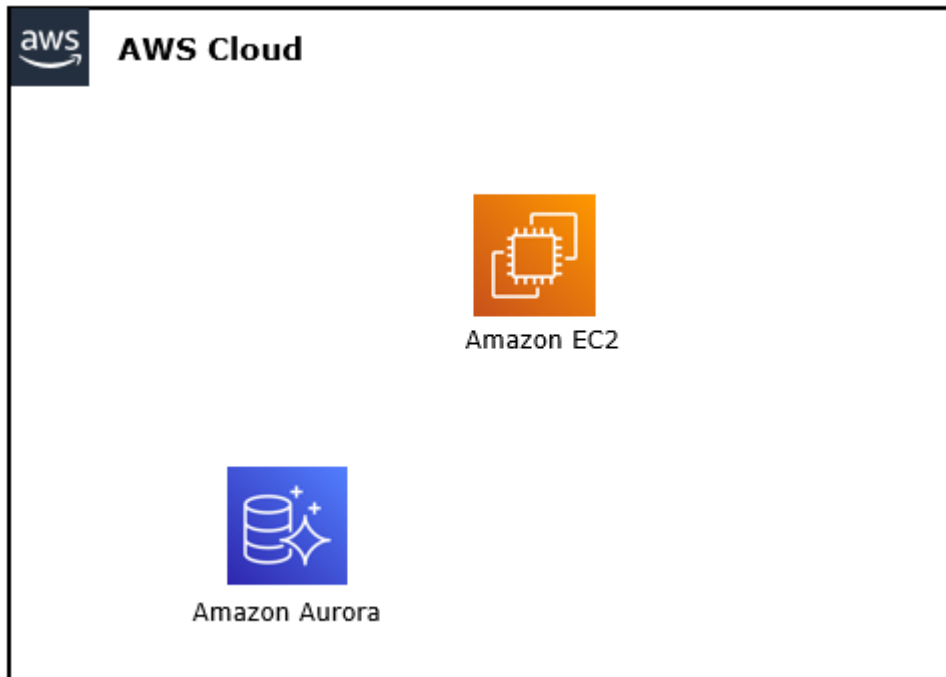
Note: You can see the database instance **Status** as **Creating**.



Note: Database instance creation will take ~20 minutes. **Don't wait, Go** to the **next task**.

Task 2: Deploy the WebApp Server

In this task, you will create Ubuntu virtual machine.



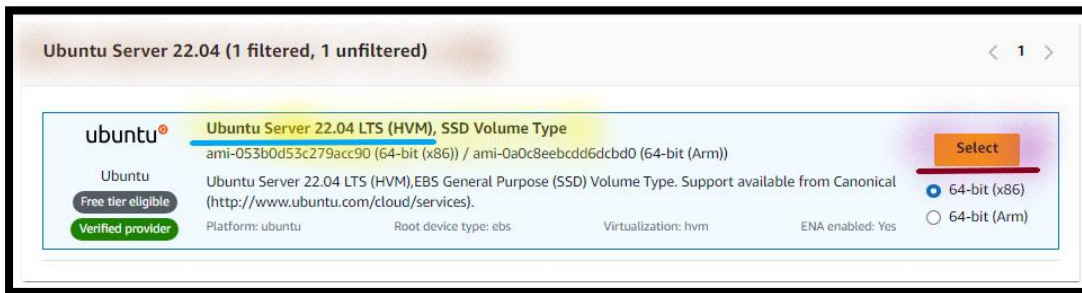
Step 1: Create EC2 Instance

4. In the **AWS Management Console**, on the **Services** menu, search and select **EC2**.
5. Choose the **YOUR ALLOCATED REGION** list to the right of your account information on the navigation bar.
6. Select **Instances**.
7. Select **Launch Instances**.
 - a. In the **Name and tags** section:
 - i. **Name**: Write **Webapp Server**.
 - b. In the **Application and OS Images** section:
 - i. In the **Search box**:
 - a) Type **Ubuntu Server 22.04**.
 - b) Press **Enter** key.

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

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

1) Select **Ubuntu Server 22.04**.



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

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

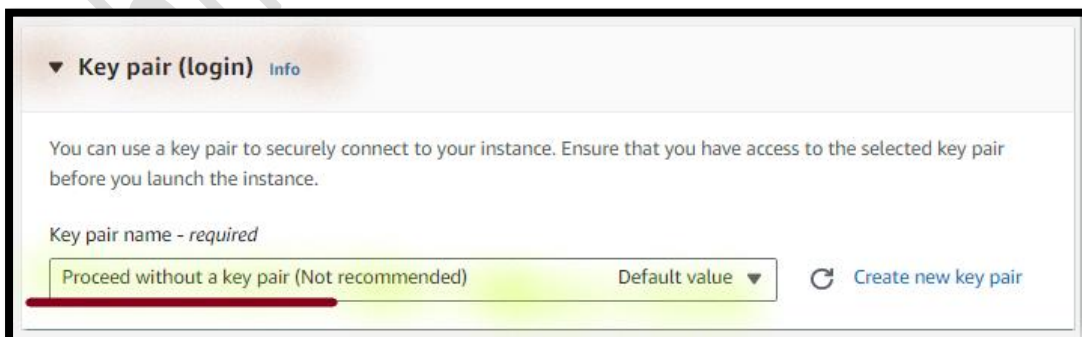
i. **Instance type:** Dropdown and in the **Search box:**

a) Type **t2.micro**.

b) Select **t2.micro**.

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

i. **Key pair name:** Dropdown and select **Proceed without a key pair**.



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

Note: You can see "Allow SSH traffic" is already **enabled** from "Anywhere".

i. **Firewall:** Select **Create security group**.

a) **Allow HTTP traffic from the internet:** **Enable** the **Checkmark**.

Network settings [Info](#) Edit

Network | [Info](#)
vpc-0ebeb9fb98f7f1ec8 | LAB VPC

Subnet | [Info](#)
subnet-07a7e7aef186453a6 | Public subnet-02

Auto-assign public IP | [Info](#)
Enable
Additional charges apply when outside of free tier allowance

Firewall (security groups) | [Info](#)
A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

☒ Create security group ☐ Select existing security group

We'll create a new security group called 'launch-wizard-1' with the following rules:

- ☒ Allow SSH traffic from Anywhere (0.0.0.0/0)
Helps you connect to your instance
- ☐ Allow HTTPS traffic from the internet
To set up an endpoint, for example when creating a web server
- ☒ Allow HTTP traffic from the internet
To set up an endpoint, for example when creating a web server

Warning: Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only. ×

Note: Leave the other details as default.

- f. In the **Advanced details** section (**Expand**):
- i. **User data**: Copy the **below script**, to **set** the **Password**.

```
#!/bin/bash
echo ubuntu:lab-password | chpasswd
sed -i 's|[#]*KbdInteractiveAuthentication no|#KbdInteractiveAuthentication no|g' /etc/ssh/sshd_config
sed -i 's|[#]*PasswordAuthentication yes|PasswordAuthentication yes|g' /etc/ssh/sshd_config
systemctl restart sshd.service
hostnamectl set-hostname webapp-server
```

Note: The script does the **following**:

1. **Set** the **Username** and **Password**.
2. **Set** the **Instance name** as **Webapp-Server**.



- g. In the **Summary** section:
- i. Select **Launch Instances**.

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

8. Select **View all instances**

Note: **Wait**, till you can see the **Webapp Server** Instance **State** is **Running**.

Note: **Wait**, till you can see the **Webapp Server** Instance **Status check** is **2/2 check passed**.

Task 3: Connect to Linux Web Server

In this task, you will log into the Linux web server.

Step 1: Copy the IP Address of Linux Web Server

9. **From** the **EC2** console.
10. Select the **WebApp Server**.
- a. Select the **Details**.

Note: **Copy** the **Public IP address** of **WebApp Server** in the **Notepad**.

Step 2: Connect to Linux Web Server Instance

11. From the **Local Desktop/ Laptop** (Windows Desktop), **Open** the **MobaXterm**.
12. From the **MobaXterm**.
- a. Select **Session**.
- b. Select **SSH**.
- i. **Remote host:** Write **Public IP address** of the **WebApp Server**.
- ii. **Specify username:** **Enable** the **Checkmark**.
- iii. **Specify username:** Write **ubuntu**.
- a) Select **Ok**.

Note: You can see the **Prompt** for **Password**.

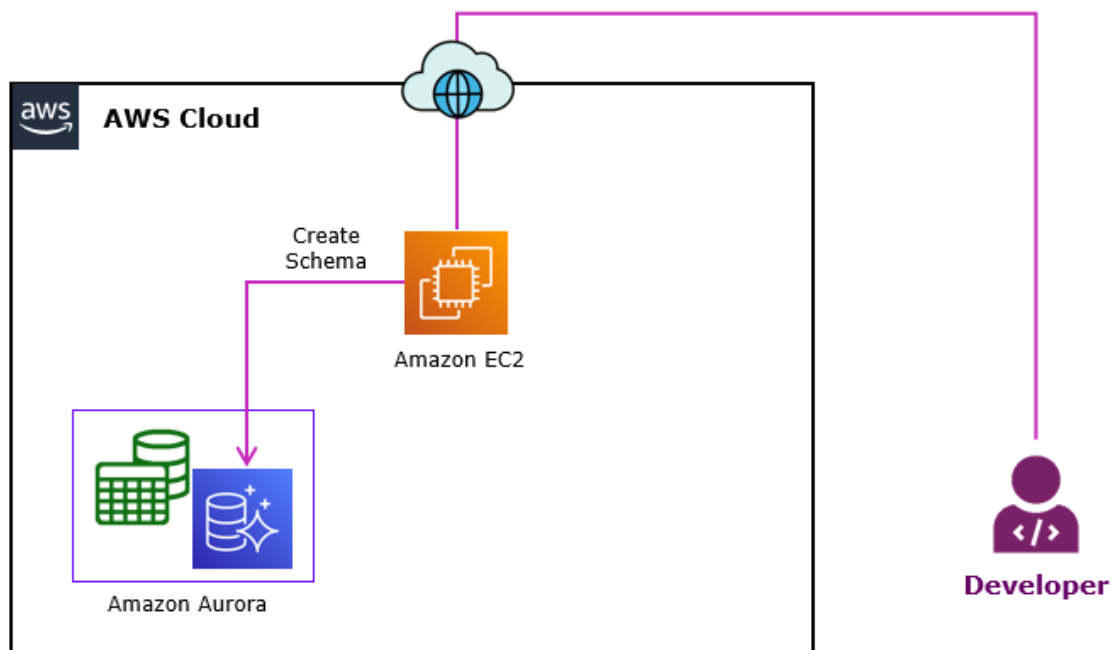
iv. **Password:** Type **lab-password**.

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

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

Task 4: Create Database and Table

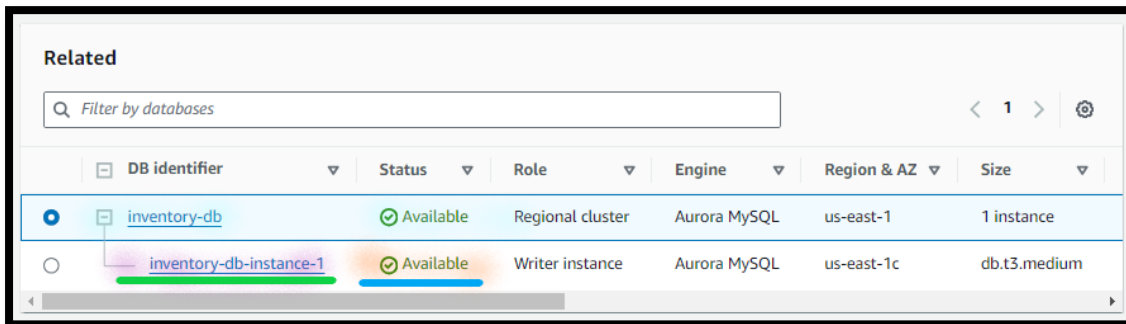
In this task, you will create SQL database, table, and schema.



Step 1: Copy the Amazon Aurora Instance Endpoint

13. In the **AWS Management Console**, on the **Services** menu, search and select **RDS**.
14. Choose the **YOUR ALLOCATED REGION** list to the right of your account information on the navigation bar.
15. Select **Databases**.

Note: **Wait**, till you can see the **inventory-db-instance-1** database instance **Status** as **Available**.

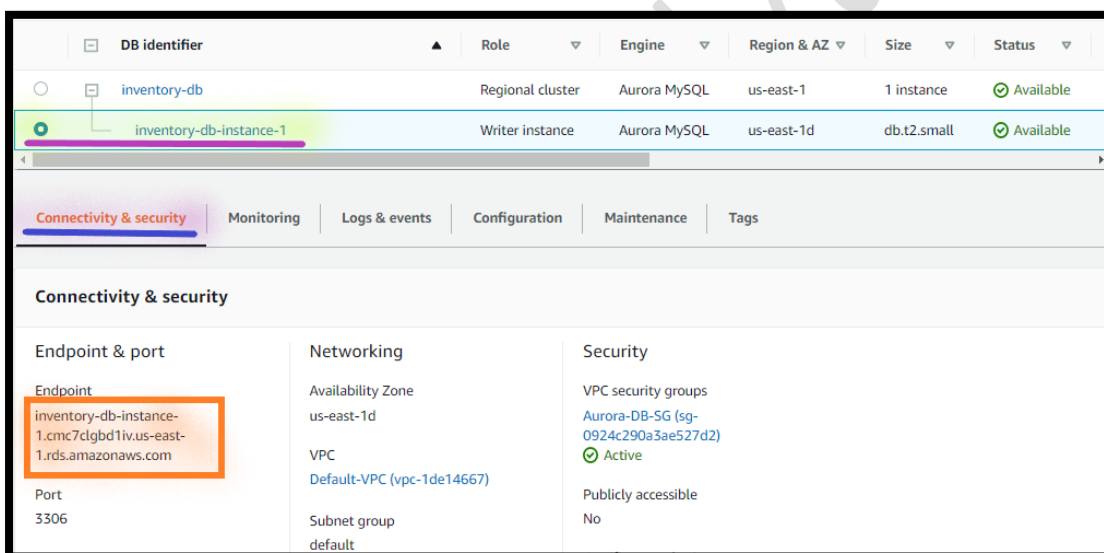


Related

Filter by databases

DB identifier	Status	Role	Engine	Region & AZ	Size
inventory-db	Available	Regional cluster	Aurora MySQL	us-east-1	1 instance
inventory-db-instance-1	Available	Writer instance	Aurora MySQL	us-east-1c	db.t3.medium

- a. Open the **inventory-db-instance-1**.
 - i. Select the **Connectivity & security**.
 - a) **Copy** the **Endpoint** in the **Notepad**.



DB identifier	Role	Engine	Region & AZ	Size	Status
inventory-db	Regional cluster	Aurora MySQL	us-east-1	1 instance	Available
inventory-db-instance-1	Writer instance	Aurora MySQL	us-east-1d	db.t2.small	Available

Connectivity & security | Monitoring | Logs & events | Configuration | Maintenance | Tags

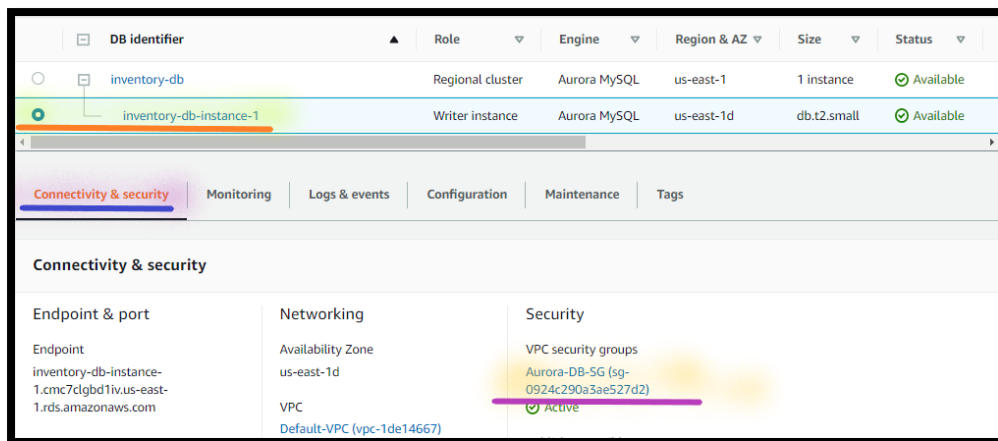
Connectivity & security

Endpoint & port	Networking	Security
Endpoint inventory-db-instance-1.cmc7clgbd1iv.us-east-1.rds.amazonaws.com Port 3306	Availability Zone us-east-1d VPC Default-VPC (vpc-1de14667) Subnet group default	VPC security groups Aurora-DB-SG (sg-0924c290a3ae527d2) Active Publicly accessible No

Step 2: Update Security Group for Database

16. From the **inventory-db** console.
17. Open the **inventory-db-instance-1**.
 - a. Select **Connectivity & security**.

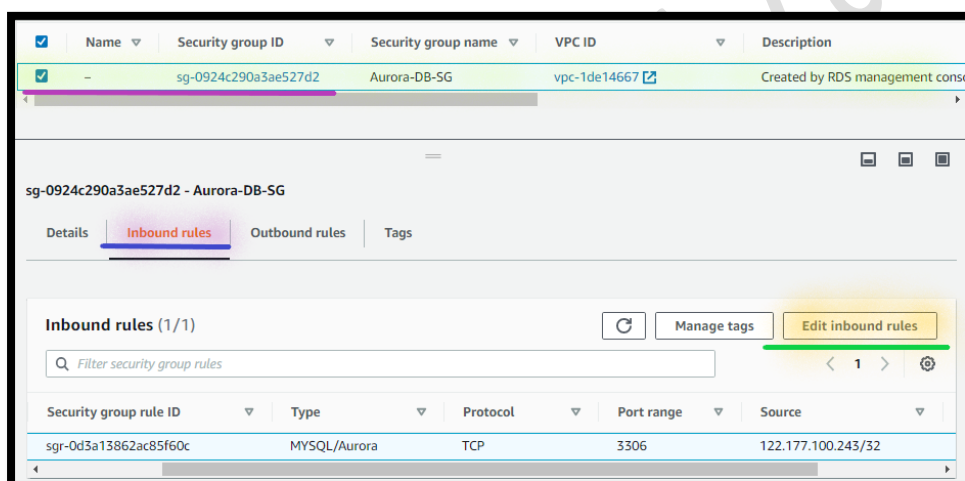
Note: You can see the **Aurora-DB-SG** under **VPC security groups**.



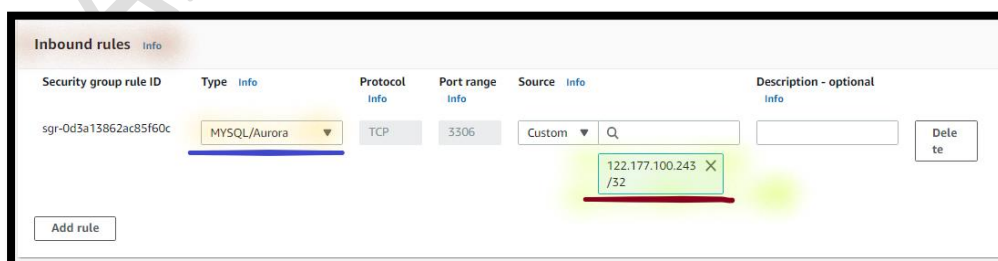
i. Open the **Aurora-DB-SG** security group.

a) Select **Inbound rules**.

b) Select **Edit inbound rules**.



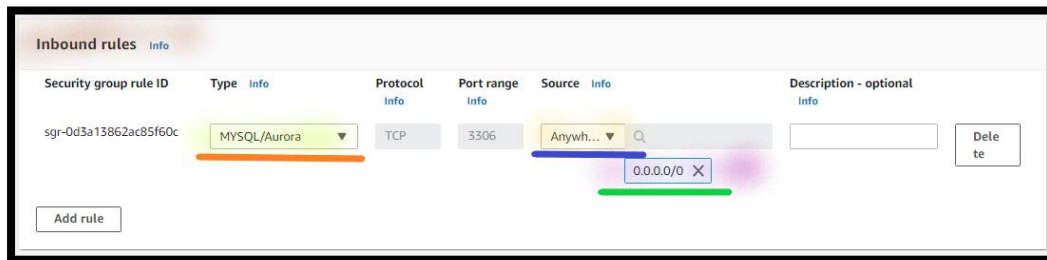
Note: You can see the **Public IP address** under **Source**.



1) **Source:** Dropdown and select **Anywhere IPv4**.

Note: You can see the **0.0.0.0/0** under **Source**.

Note: **0.0.0.0/0** allow the Aurora database instance to accept the traffic from **anywhere**.



2) Select **Save rules**

Step 4: Install the MySQL Client

18. Return to the **WebApp Server**.

19. From the **Linux terminal**:

a. **Execute** the **below command** to **update** the **packages**:

```
sudo apt-get -y update
```

b. **Execute** the **below command** to **install** the **mysql client**:

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

Step 5: Connect to MySQL Instance

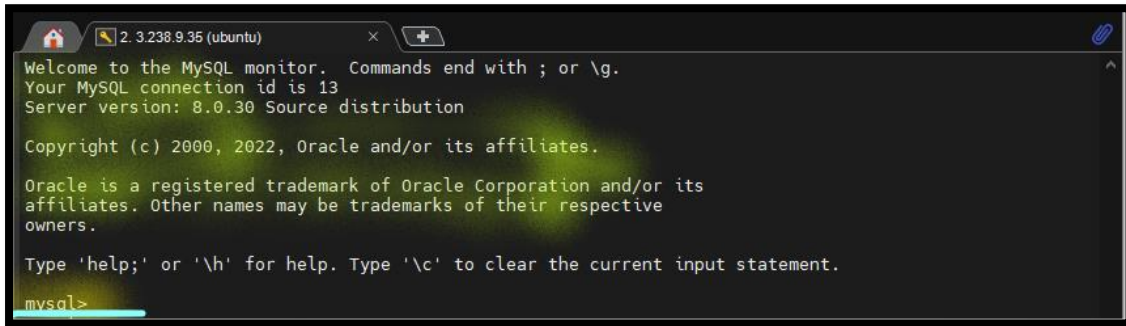
20. From the **Linux terminal**:

a. **Execute** the **below command** to **update** the **packages**:

```
mysql -u master -p -h Hostname
```

Note: Replace the **Hostname** with **inventory-db-instance-1** database instance **endpoint**, which you have copied in the previous step.

Note: You can see the **mysql>** prompt.



```
2. 3.238.9.35 (ubuntu) x +
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 13
Server version: 8.0.30 Source distribution

Copyright (c) 2000, 2022, 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 6: Create Database, Table and Table Schema

21. From the **MySQL console**:

- a. **Execute** the **below command** to **create** the **databases**.

```
create database inventory;
```

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

- b. **Execute** the **below command** to **list** the **existing databases**.

```
show databases;
```

Note: You can see the **inventory** database.

- c. **Execute** the **below command** to **use** the **inventory databases**.

```
use inventory;
```

Note: You can see the **database changed** message.

- d. **Execute** the **below command** to **create** the **products table**.

```
create table `products` (  
  `id` int(11) not null auto_increment,  
  `name` varchar(45) not null,  
  `quantity` varchar(45) not null,  
  `price` varchar(45) not null,  
  primary key (`id`),  
  unique key `id_unique` (`id`));
```

Note: You can see the **Query OK, 0 rows affected** message.

- e. **Execute** the **below command** to **show** the **products table**.

```
show tables;
```

Note: You can see the **products** table.

- f. **Execute** the **below command** to **show** the **products table**.

```
describe products;
```

Note: You can see the **products table** schema.

```
mysql> describe products;  
+-----+-----+-----+-----+-----+-----+  
| Field | Type | Null | Key | Default | Extra |  
+-----+-----+-----+-----+-----+-----+  
| id    | int  | NO   | PRI | NULL    | auto_increment |  
| name  | varchar(45) | NO | | NULL    | |  
| quantity | varchar(45) | NO | | NULL    | |  
| price | varchar(45) | NO | | NULL    | |  
+-----+-----+-----+-----+-----+-----+  
4 rows in set (0.01 sec)  
  
mysql>
```

- g. **Execute** the **below command** to **insert** the **row** in the **products table**.

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

Note: You can see the **Query OK, 1 rows affected** message.

- h. **Execute** the **below command** to **insert** the **row** in the **products table**.

```
insert into products (name, quantity, price) VALUES ('Monitor', '11', '7900');
```

Note: You can see the **Query OK, 1 rows affected** message.

- i. **Execute** the **below command** to **exit** the **MySQL**.

```
exit
```

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

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

Task 5: Develop the Php Application

In this task, you will develop the Php code who can perform read and write operation from single database server.

Step 1: Develop the Code to Perform CRUD Operation

22. **Unzip** the **LAB-m05-01-Code-A.zip** (Php code).

Note: **lab-m05-01-code-a.zip** is available with the **Lab manual**.

Note: You can see the **index.php** and **data.php** files.

- a. Open the **data.php** file in the **Notepad**.
 - i. **Replace** the **TO DO 1** with the **inventory-db-instance-1 database** instance **endpoint** (which you have copied in the previous step).

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

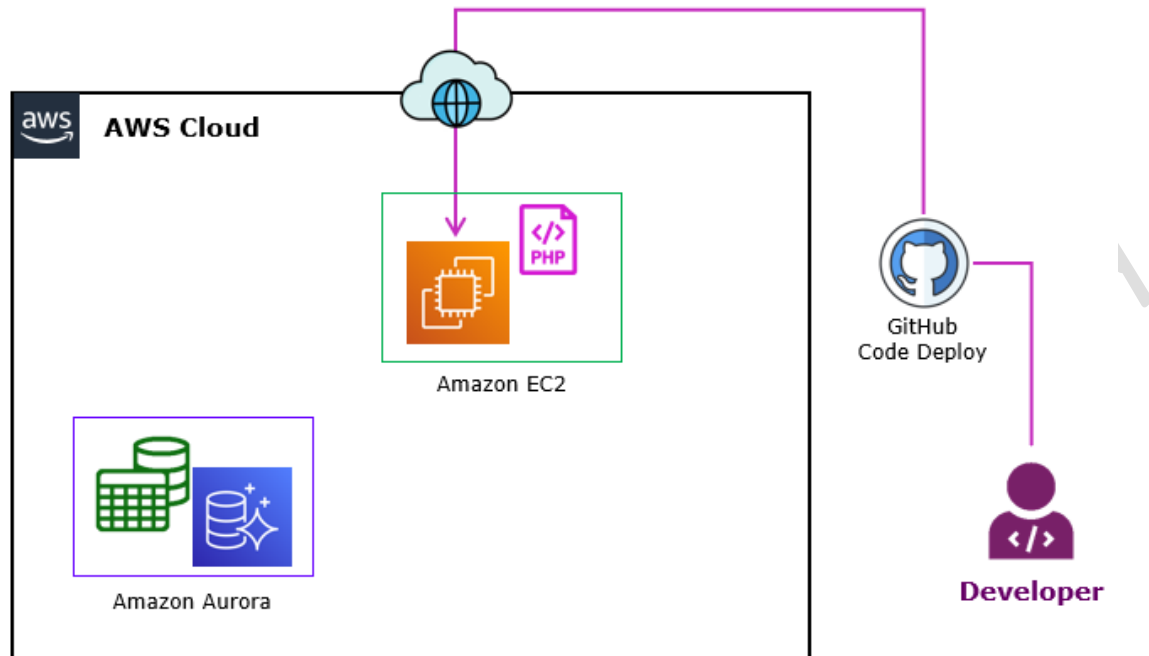
- ii. **Replace** the **TO DO 2** with the **database instance user name master**.
 - iii. **Replace** the **TO DO 3** with the **database instance password lab-password**.
 - iv. **Replace** the **TO DO 4** with the **database name inventory**.
 - v. **Replace** the **TO DO 5** with the **inventory database table name products**.

```
$servername = 'inventory-db-instance-1.cmc7clgbd1iv.us-east-1.rds.amazonaws.com';  
$username = 'master';  
$password = 'lab-password';  
$database = 'inventory';  
$table = 'products';
```

- b. Select **File**.
 - i. Select **Save**.

Task 5: Deploy the Php Application

In this task, you will deploy the Php code into Aws virtual machine and configure the runtime environment.



Step 1: Upload the Code to GitHub Repository

23. **Open** your **GitHub account**.
 - a. Select the **+** sign.
 - i. **From** the **Create a new repository** page:
 - a) **Repository name**: Write **lab-07-01**.
 - b) Select the **Public**.
 - 1) Select the **Create repository**.

Note: You can see the **lab-07-01** repository page.

- b. **From** the **lab-07-01** repository:
 - i. Select the **Uploading an existing file**.
 - 1) **Drag and drop** the **Code** in the **GitHub Repository**.

Note: You need to **Upload** the **index.php** and **data.php** files.

Note: You can see the **files to be uploaded**.

- ii. Select the **Commit Changes**.

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

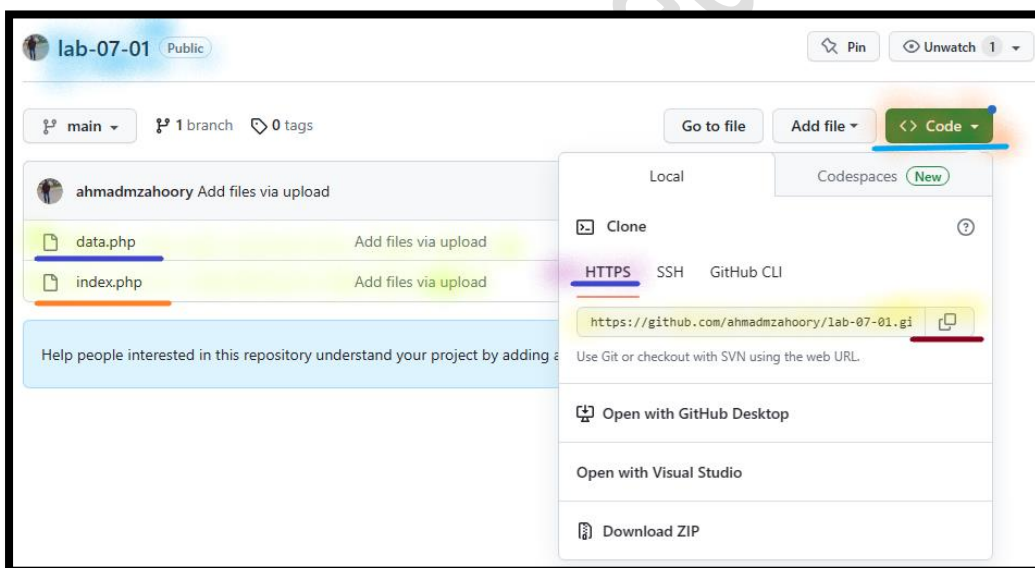
Step 2: Clone the GitHub Repository

24. Open the **lab-07-01** **GitHub repository**.

- a. Select the **Code**.

- i. Select the **HTTPS**.

- a) **Copy** the **Clone URL** in **Notepad**.



Step 3: Build the Runtime Environment

25. **Return** to the **Webapp Server**.

26. **From** the **Linux terminal**:

a. **Execute** the **below command** to **install** the **apache**:

```
sudo apt-get install -y apache2
```

b. **Execute** the **below command** to **add** the **php repository**:

```
sudo add-apt-repository -y ppa:ondrej/php
```

c. **Execute** the **below command** to **update** the **package**:

```
sudo apt-get -y update
```

d. **Execute** the **below command** to **install** the **php 8.2**:

```
sudo apt-get install -y php8.2
```

e. **Execute** the **below command** to **install** the **mysql module for php 8.2**:

```
sudo apt-get install -y php8.2-mysql
```

f. **Execute** the **below command** to **install** the **git**:

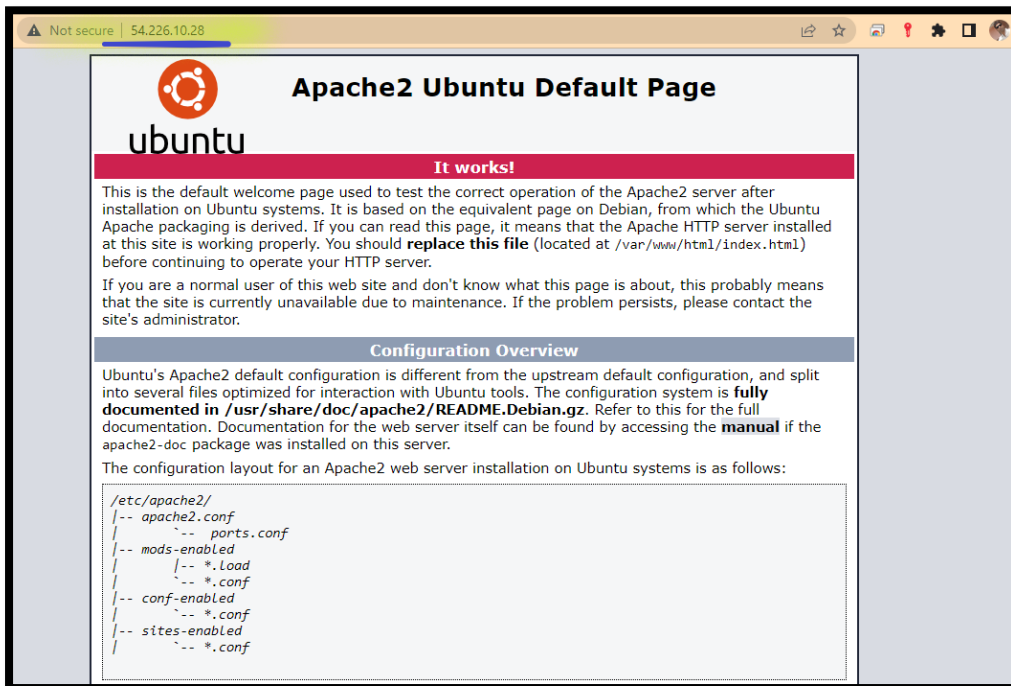
```
sudo apt-get install -y git
```

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

Step 4: Access the Web App Server

27. From your **Local Desktop/ Laptop**, open the **Browser**, write **Public IP Address** of the **WebApp server**, to access the **website**.

Note: You can see the **Default Apache Ubuntu page**.



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

Step 6: Deploy the Php Code

28. Return to the **Webapp Server**.
29. From the **Linux terminal**:
- Execute** the **below command** to **change** the **directory path**:

```
cd /var/www/html/
```

- b. **Execute** the **below command** to **list** the **file and folders**:

```
ls -l
```

Note: You can see the **index.html** (Default Apache Ubuntu page).

- c. **Execute** the **below command** to **remove** the **default apache page**:

```
sudo rm index.html
```

- d. **Execute** the **below command** to **clone** the **git**:

```
sudo git clone CLONE-WEBAPP-URL
```

Note: Replace the **CLONE-WEBAPP-URL** with the **LAB-07-01 GitHub Repository URL**, which have copied in the previous step.

- e. **Execute** the **below command** to **list** the **file and folders**:

```
ls -l
```

Note: You can see the **lab-07-01** folder.

- f. **Execute** the **below command** to **change** the **directory path**:

```
cd lab-07-01
```

- g. **Execute** the **below command** to **move** the **code files** to **default directory**:

```
sudo mv -v /var/www/html/lab-07-01/* /var/www/html/
```

- h. **Execute** the **below command** to **change** to the **parent directory path**:

```
cd ..
```

- i. **Execute** the **below command** to **list** the **file and folders**:

```
ls -l
```

Note: You can see the **data.php** and **index.php** files.

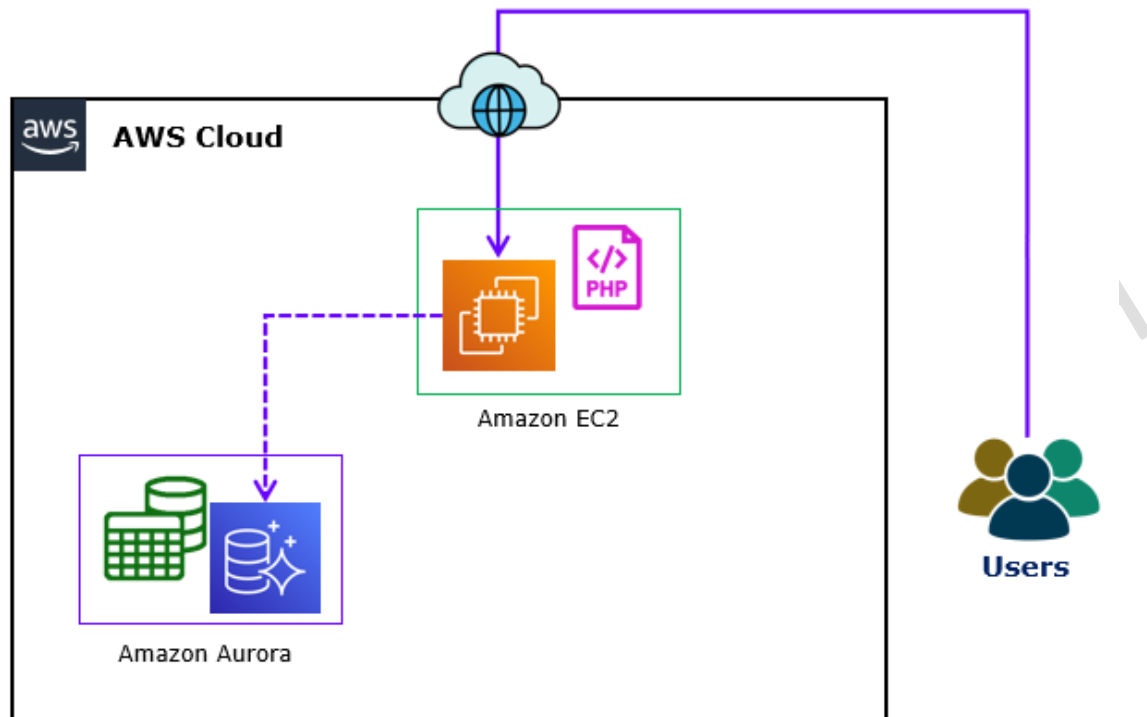
- j. **Execute** the **below command** to **restart** the **apache service**:

```
sudo systemctl restart apache2.service
```

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

Task 6: Access the Web Application

In this task, you will test your deployment by performing the CRUD operation.



Step 1: Access the Php App Server

30. **Refresh** to the **Web browser**, from where you have opened the **WebApp page**.

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

The screenshot shows a web browser window displaying the **AWS Database LAB** application. The browser's address bar shows the URL 35.153.55.123. The application has a teal border and contains the following form:

AWS Database LAB

Product Name

Product Qty.

Product Price

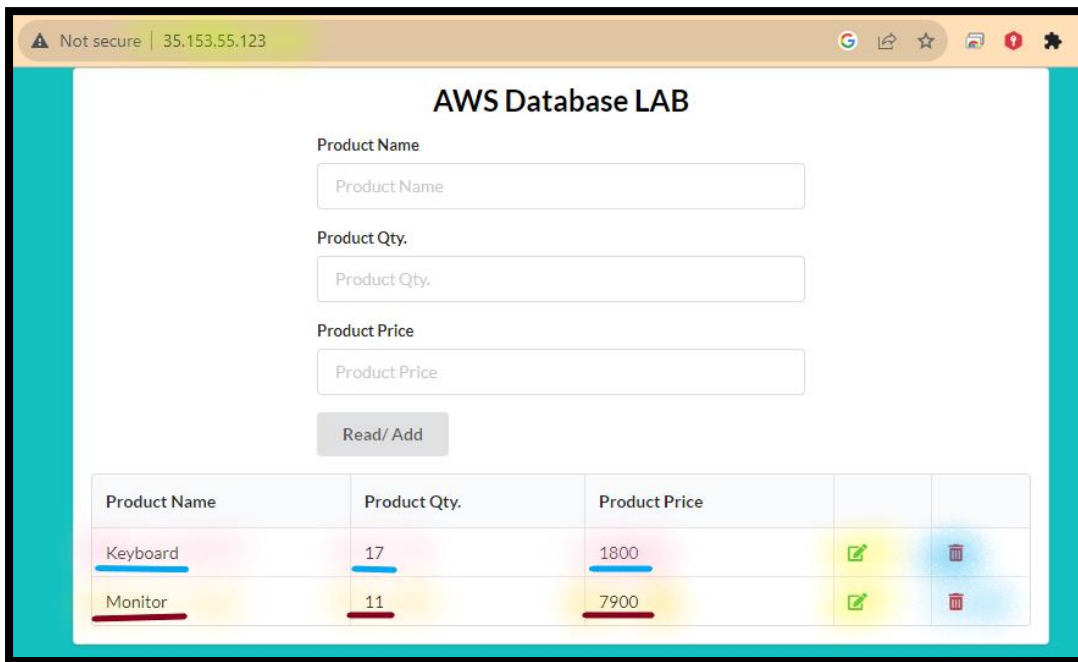
Product Name	Product Qty.	Product Price		
--------------	--------------	---------------	--	--

Step 2: Perform the CRUD Operation

31. From the **WebApp application**.

- a. Click on the **Read** to get the data from the database.

Note: You can see the **rows** added via **mysql**.



The screenshot shows a web browser window with the URL 35.153.55.123. The page is titled "AWS Database LAB". It features a form with three input fields: "Product Name", "Product Qty.", and "Product Price". Below the form is a "Read/ Add" button. At the bottom, there is a table with the following data:

Product Name	Product Qty.	Product Price		
Keyboard	17	1800		
Monitor	11	7900		

32. From the **webapp application**:

a. Add **Inventory data**:

- i. **Product name:** Write **HP Digital Tab**.
- ii. **Product quantity:** Write **14**.
- iii. **Product price:** Write **4200**.
- iv. Select **Add**.

Note: **Inventory data** gets **stored** in the **MySQL database**. You can **see** the same under **details**.

Not secure 18.234.226.71

AWS Database LAB

Product Name
HP Digital Tab

Product Qty.
14

Product Price
4200

Read/ Add

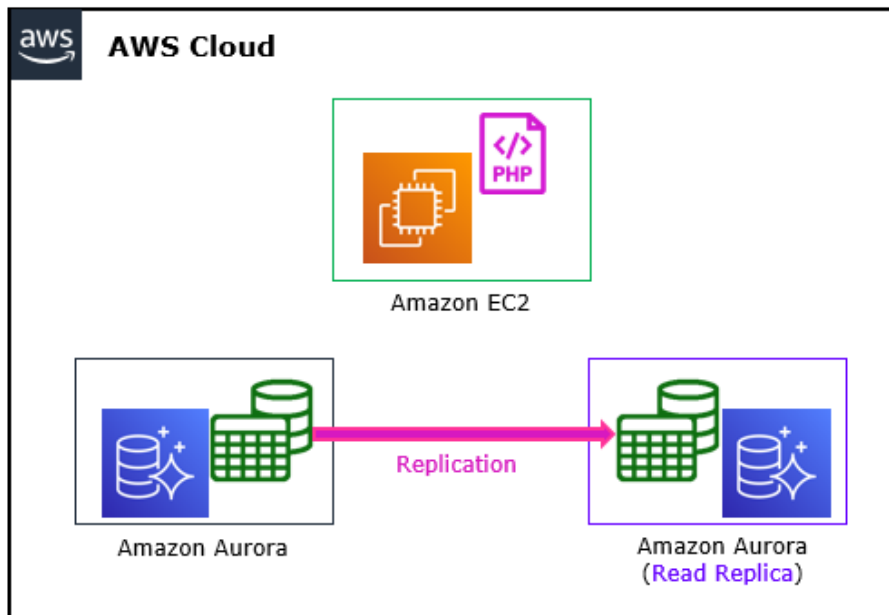
Product Name	Product Qty.	Product Price		
Keyboard	17	1800	✓	✖
Monitor	11	7900	✓	✖
HP Digital Tab	14	4200	✓	✖

Note: You can also **update** or **delete** the rows.

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

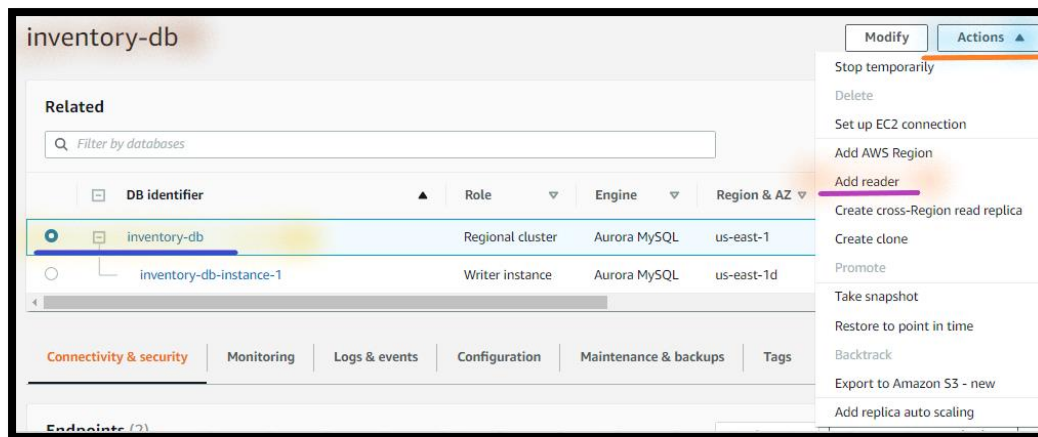
Task 7: Create Database Read Replica

In this task, you will create the database (Amazon Aurora) read replica.



Step 1: Copy the Amazon Aurora Instance Details

33. In the **AWS Management Console**, on the **Services** menu, search and select **RDS**.
34. Choose the **YOUR ALLOCATED REGION** list to the right of your account information on the navigation bar.
35. Select **Databases**.
 - a. Select **inventory-db**.
 - i. Select **Actions**.
 - a) Select **Add Reader**.



Note: You can see the **Add reader** page.

ii. From the **Settings** section:

a) **DB instance identifier:** Write **inventory-db-instance-2**.

Note: Leave the other details as default.



iii. From the **Connectivity** section:

a) **Public access:** Select **Publicly accessible**.

Note: Leave the other details as default.

Connectivity

Public access

☒ **Publicly accessible**
 Public access enables a public IP address to be assigned to the database. Amazon EC2 instances and other resources outside of the VPC can connect to your database. Resources inside the VPC can also connect to the database. Choose one or more VPC security groups that specify which resources can connect to the database.

☐ **Not publicly accessible**
 No IP address is assigned to the DB instance. EC2 instances and devices outside the VPC can't connect.

Availability Zone [Info](#)
 The EC2 Availability Zone that the database will be created in.

No preference

► **Additional configuration**

iv. In the **Additional configuration** section:

a) **Encryption:** **Unselect** the **Enable encryption**.

Note: Leave the other details as default.

v. Select **Add reader**.

Note: You can see the **Reader database** instance **Status** as **Creating**.

DB identifier	Status	Role	Engine	Region & AZ	Size
inventory-db	Available	Regional cluster	Aurora MySQL	us-east-1	2 instances
inventory-db-instance-1	Available	Writer instance	Aurora MySQL	us-east-1c	db.t3.medium
inventory-db-instance-2	Creating	Reader instance	Aurora MySQL	-	db.t3.medium

Note: **Wait (~10 mnts.)**, till **Reader database** instance **Status** as **Available**.

Related						
Filter by databases						
DB identifier	Status	Role	Engine	Region & AZ	Size	
inventory-db	Available	Regional cluster	Aurora MySQL	us-east-1	2 instances	
inventory-db-instance-1	Available	Writer instance	Aurora MySQL	us-east-1c	db.t3.medium	
inventory-db-instance-2	Available	Reader instance	Aurora MySQL	us-east-1b	db.t3.medium	

Task 8: Develop the Php Application

In this task, you will develop the Php code who can perform read and write operation from two different database servers.

Step 1: Copy the Amazon Aurora Read Instance Endpoint

36. In the **AWS Management Console**, on the **Services** menu, click **RDS**.

37. Select **Databases**.

- a. Open the **inventory-db-instance-2**.
 - ii. Select the **Connectivity & security**.
 - a) **Copy** the **Endpoint** in the **Notepad**.

Step 2: Develop the Code to Perform CRUD Operation

38. Unzip the **Lab-m05-01-Code-B.zip** (Php code).

Note: **lab-m05-01-code-b.zip** is available with the **Lab manual**.

Note: You can see the **index.php** and **data.php** files.

- a. Open the **data.php** file in the **Notepad**.
 - i. **Replace** the **TO DO 1 [RW Database]** with the **inventory-db-instance-1 database instance endpoint**.

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

- ii. Replace the **TO DO 1** [**Read Database**] with the **inventory-db-instance-2 database** instance **endpoint**.

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

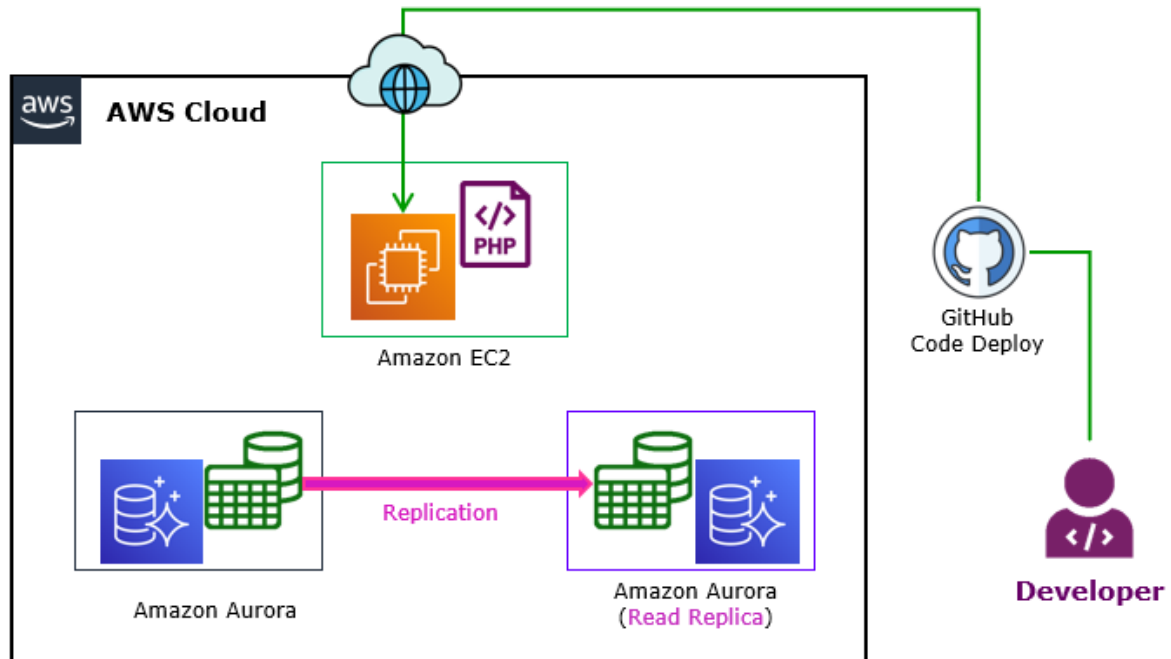
- iii. Replace the **TO DO 2** with the **database instance username** **master**.
- iv. Replace the **TO DO 3** with the **database instance password** **lab-password**.
- v. Replace the **TO DO 4** with the **database name** **inventory**.
- vi. Replace the **TO DO 5** with the **inventory database table name** **products**.

```
$write_servername = 'inventory-db-instance-1.cmc7clgbd1iv.us-east-1.rds.amazonaws.com';  
$read_servername = 'inventory-db-instance-2.cmc7clgbd1iv.us-east-1.rds.amazonaws.com';  
$username = 'master';  
$password = 'lab-password';  
$database = 'inventory';  
$table = 'products';
```

- b. Select **File**.
- c. Select **Save**.

Task 9: Deploy the Php Application

In this task, you will be updating the existing Php code with Php Code (Second version) on AWS virtual machine.



Step 1: Upload the Code to GitHub Repository

39. **Open** your **GitHub account**.

a. **Open** the **lab-07-01** repository.

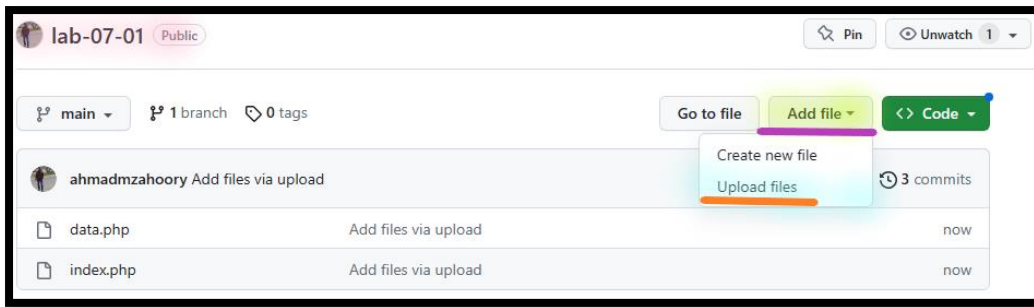
Note: You can see the **index.php** and **data.php** files of **Code-A**.

i. **From** the **lab-07-01** repository:

Note: You are **updating** the older code files with **new code files**.

a) Select **Add file**.

1) Select **Upload files**.



- b) **Drag and drop** the **Code** (**Code-B**) in the **GitHub Repository**.

Note: You need to **Upload** the **index.php** and **data.php** files of **Code-B**.

Note: You can see the **files to be uploaded**.

- c) Select the **Commit Changes**.

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

Step 2: Clone the GitHub Repository

40. **Open** the **lab-07-01 GitHub repository**.

- a. Select the **Code**.
 - i. Select the **HTTPS**.
 - a) **Copy** the **Clone URL** in **Notepad**.

Step 3: Deploy the Updated Code

41. **Return** to the **WebApp Server**.

42. **From** the **Linux terminal**:

- i. **Execute** the **below command** to **change** the **directory path**:

```
cd /var/www/html/
```

- ii. **Execute** the **below command** to **list** the **file and folders**:

```
ls -l
```

Note: You can see the **data.php** and **index.php** files.

- iii. **Execute** the **below command** to **remove** the **files and directory**:

```
sudo rm -r *
```

- iv. **Execute** the **below command** to **list** the **file and folders**:

```
ls -l
```

Note: You can see the **all files** and **folders** gets **deleted**.

- v. **Execute** the **below command** to **clone** the **git**:

```
sudo git clone CLONE-WEBAPP-URL
```

Note: **Replace** the **CLONE-WEBAPP-URL** with the **LAB-05-01 GitHub Repository URL** you have copied in the previous step.

- vi. **Execute** the **below command** to **list** the **file and folders**:

```
ls -l
```

Note: You can see the **lab-07-01** folder.

- vii. **Execute** the **below command** to **change** the **directory path**:

```
cd lab-07-01
```

- viii. **Execute** the **below command** to **move** the **code files** to **default directory**:

```
sudo mv -v /var/www/html/lab-07-01/* /var/www/html/
```

- ix. **Execute** the **below command** to **change** to the **parent directory path**:

```
cd ..
```

- x. **Execute** the **below command** to **list** the **file and folders**:

```
ls -l
```

Note: You can see the **data.php** and **index.php** files.

- xi. **Execute** the **below command** to **restart** the **apache service**:

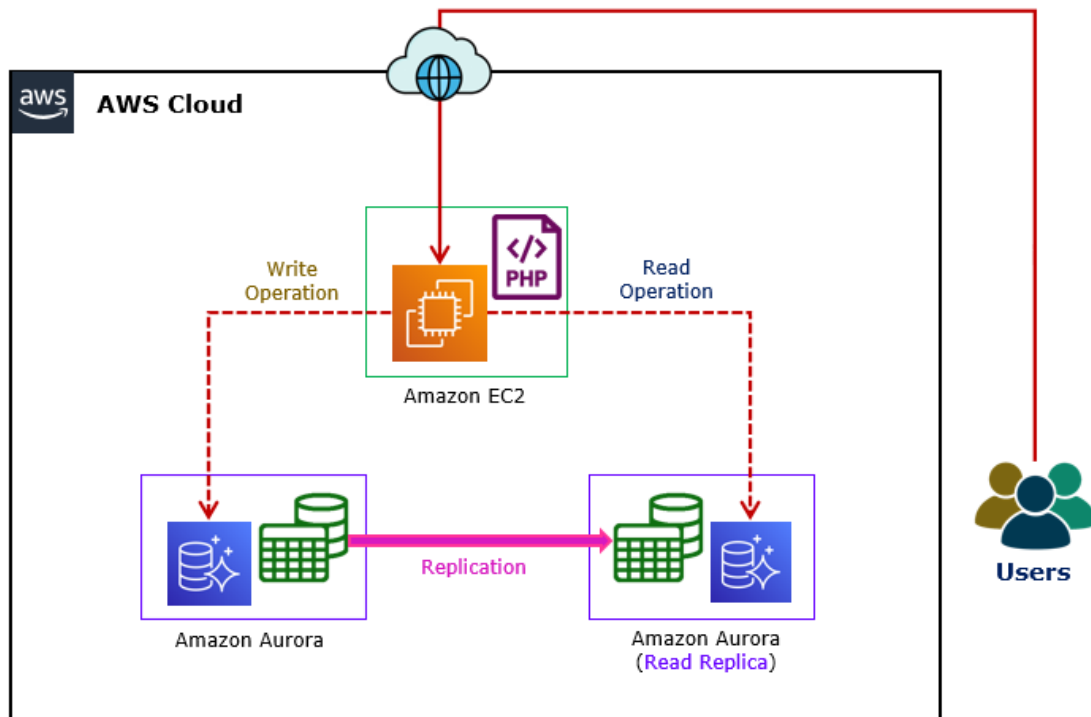
```
sudo systemctl restart apache2.service
```

- xii. **Execute** the **below command** to **exit** the **linux terminal**:

```
exit
```

Task 10: Test your Deployment

In this task, you will test your deployment by performing the CRUD operation.



Step 1: Access the Web App Server

43. **Refresh** to the **Web browser**, from where you have opened the **Webapp page**.

Note: You can see the **new Web page**.

Product Name

Product Qty.

Product Price

Add Read

Product Name	Product Qty.	Product Price		
--------------	--------------	---------------	--	--

Step 2: Perform the CRUD Operation

44. From the **Webapp application**.

- Click on the **Read** to get the data from the database.

Note: You can see the **rows** added in **previous steps**.

Note: **Read operation** is performed via Amazon **Aurora Reader**.

Product Name

Product Qty.

Product Price

Add Read

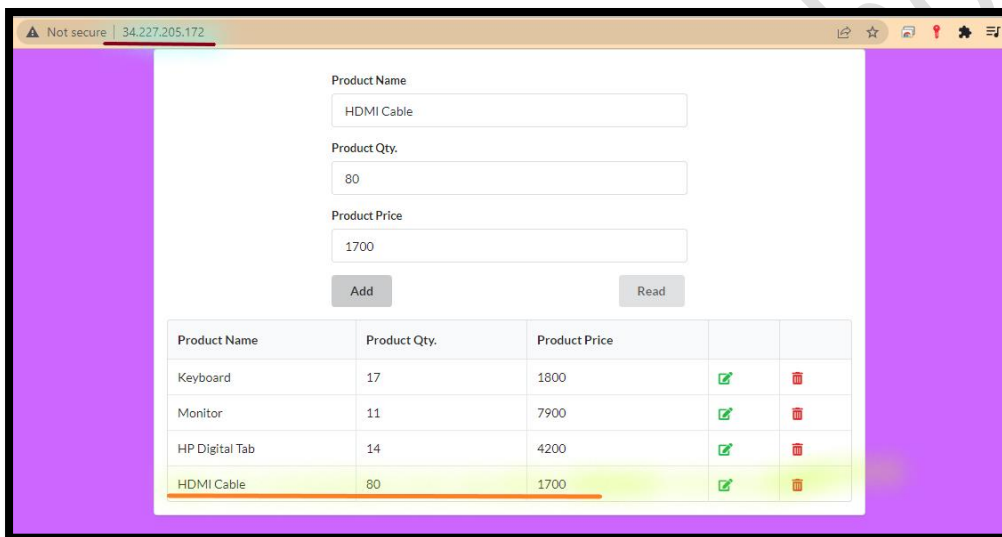
Product Name	Product Qty.	Product Price		
Keyboard	17	1800	✓	✗
Monitor	11	7900	✓	✗
HP Digital Tab	14	4200	✓	✗

45. From the **webapp application**:

a. Add **Inventory data**:

- i. **Product name**: Write **HDMI Cable**.
- ii. **Product quantity**: Write **80**.
- iii. **Product price**: Write **1700**.
- iv. Select **Add**.

Note: **Write operation** is performed via Amazon **Aurora Writer**.



Product Name	Product Qty.	Product Price		
Keyboard	17	1800		
Monitor	11	7900		
HP Digital Tab	14	4200		
HDMI Cable	80	1700		

Note: You can also **update** or **delete** the rows.

Task 12: Clean up the Environment

Step 1: Terminate EC2 Instances

46. In the **AWS Management Console**, on the **Services** menu, click **EC2**.
47. Click **Instances**.
48. Select **WebApp Server**.

- a. Select the **Instance state**.
 - i. Select **Terminate instance**.
 - a) Select **Terminate**.

Step 2: Terminate an RDS

49. In the **AWS Management Console**, on the **Services** menu, click **RDS**.

50. Select **Databases**.

- a. Select **inventory-db-instance-1**.
 - i. Select **Actions**.
 - a) Select **Delete**.
 - 1) When you **get prompt**, type **delete me**.
 - 2) Select **Delete**.
- b. Select **inventory-db-instance-2**.
 - i. Select **Actions**.
 - b) Select **Delete**.
 - 1) When you **get prompt**, type **delete me**.
 - 2) Select **Delete**.
- c. Select **inventory-db**.
 - i. Select **Actions**.
 - c) Select **Delete**.
 - 1) **Create final snapshot**: **Uncheck** the **Checkmark**.
 - 2) **I acknowledge that upon instance deletion**: **Enable** the **Checkmark**.
 - 3) When you **get prompt**, type **delete me**.
 - 4) Select **Delete DB Cluster**.