

# AWS Advanced

**RDS | EC2 database program**

(Create a MySQL instance using RDS and access it using a custom program from an EC2 instance using an appropriate role)



Learning Outcomes

* Get an EC2 instance using the 7 step workflow & SSH in it
* Create a MySQL database using RDS
* Write the custom application and deploy on EC2
* Able to apply roles to EC2 instances

Final Goal

AWS (N Virginia)

IAM

Cache

Laptop

RDS

(MySQL)

Browser

SSH

EC2

DB

App



# What is needed?

1. AWS Account Credentials
2. EC2 Instances (Linux)
3. Shell script environment (any text editor of your choice)
4. Full access to - EC2, RDS, IAM, Elasticache



How to do it? - 1

1. Ensure your region is set to "N Virginia"
2. Create a private MySQL RDS
   1. Create a "Dev/Test" MySQL instance
   2. Select "Non publicly" available DB
   3. DB instance = "flipbasket", UID/PWD = root/password
   4. AZ = 1a, create a new SG
   5. IMP When the RDS instance is created check the SG to ensure that the "inbound" rule does not have any specific IP address, instead use the VPC CIDR block of the EC2 instance. Otherwise we will not be able to connect from the EC2 instance
3. Create 1 EC2 instance using the 7 step workflow
   1. Use the usual Ubuntu 20.04 LTS AMI in AZ1
   2. Download a new PEM file and SSH to the instance
4. IAM
   1. Attach the pre-created role “labinstanceprofile” to the instance OR use an IAM role using

the permission “RDSFullAccess”



# How to do it? - 2

1. SSH to the EC2 instance and fire the following commands
   1. sudo apt update
   2. sudo apt install mysql-client
2. Connect to the RDS instance & create the data
   1. wget https://d6opu47qoi4ee.cloudfront.net/employees.sql
   2. mysql -h [RDS end point] -u admin -p

You should get the MySQL prompt after entering the password

* 1. create database employees;
  2. use employees;
  3. source employees.sql
  4. describe employees;
  5. Note the column names in a text editor from the output of the above command
  6. Close the mysql terminal



# How to do it? - 3

* Follow the steps for the Python program connecting to the RDS instance from EC2
  + sudo apt install python2-minimal
  + sudo apt install python3-pip
  + sudo pip3 install boto3
  + sudo pip3 install mysql-connector-python
  + sudo pip3 install pymysql

In case the pip install fails execute the following 3 lines -

export LC\_ALL="en\_US.UTF-8" export LC\_CTYPE="en\_US.UTF-8" sudo dpkg-reconfigure locales

* Use the rds.py as the template and update the hostname to the endpoint of your RDS instance

and run the program

* + wget https://d6opu47qoi4ee.cloudfront.net/rds.py
  + Modify the python program in all the places marked with "TBD"
  + python3 rds.py
  + In case you get an error for the mysql connector just re-execute the sudo pip install command

## Q: state your observations when you remove the role and rerun the program (Hint - RDS SG)



How to do it? – 4 (Advanced \*)

* Once you have updated the program to connect with the database, do the following
  + Create a Redis cluster (self managed / Elasticache)
  + Modify the program to include another method "getOrder(...)"
  + Add a class that represents the order
  + This method needs to check if the order exists in the cache, if found then get from cache and return else go to the DB, put in cache and return
    - Use the python modules - redis & pickle

\* This step of the assignment is NOT mandatory, and will not have any impact on the grades. You can choose

to do / skip it only from the learning perspective.



# Mandatory step: Resource Clean-Up

* Cloud is always **pay per use model** and all resources/services that we consume are chargeable. Cleaning up when you’ve completed your lab or project is always necessary. This is true whether you’re doing a lab or implementing a project at your workplace.

## After completing with the lab, make sure to delete each resource created in the reverse chronological order.

* Check resources in each cloud region that you have worked on before logging off.
* Since the dashboard doesn’t show cross-region resources, it is up to you to find and delete them.



# Marks Break Down –

Please find the marks breakdown below (Screenshots must

be provided for every step)

1. Creation of RDS instance : 15 marks
2. Data Population in RDS : 15 marks
3. Installation of Python libraries : 5 marks
4. Execution of Python program : 15 marks

**Installation steps and screenshot below**

**1. Ensure Your Region is Set to "N Virginia"**

When you're working with AWS, the region setting is crucial as it determines where your resources (like EC2, RDS) are created. To ensure you're working in the correct region:

1. **Go to the AWS Management Console.**
2. **In the top-right corner**, you will see the region drop-down (it might say something like "US East (N. Virginia)" or something else).
3. **Select "US East (N. Virginia)"** from the list.

Now, you're ready to work in the **N. Virginia** region.

**2. Create a Private MySQL RDS Instance**

1. **Go to the RDS Dashboard:**
   * Open the **AWS Management Console**.
   * In the search bar, type **RDS** and select **RDS**.
2. **Create a MySQL Database Instance:**
   * In the left sidebar, under **Databases**, click **Create database**.
   * Select **Standard Create**.
   * Choose **MySQL** as the database engine.
   * Select the **"Dev/Test"** template (or the closest option, as the name might change).
3. **Set Database Details:**
   * **DB Instance Identifier**: flipbasket
   * **Master Username**: root
   * **Master Password**: password
   * **Confirm Password**: password
4. **Set Availability and Connectivity:**
   * **Availability & Durability**: Choose **Single-AZ** and select **Availability Zone**: us-east-1a (This is the Availability Zone 1a).
   * **Connectivity**:
     + **VPC**: Select the default VPC or your custom VPC.
     + **Publicly Accessible**: Set this to **No**, making the DB private (this is important for security).
   * **VPC Security Group**:
     + You will need to create a new security group (SG) or use an existing one that allows traffic from the EC2 instance to the RDS instance.
     + **Important**: Ensure that the inbound rule of the security group does **not** have specific IPs but instead uses the **VPC CIDR block** of your EC2 instance. This ensures that only EC2 instances within your VPC can connect.
5. **Storage**: You can leave the default settings.
6. **Review and Create**: Review all the settings and click **Create database**.

**3. Create an EC2 Instance Using the 7-Step Workflow**

1. **Go to the EC2 Dashboard:**
   * Open the **AWS Management Console**.
   * Search for and select **EC2**.
2. **Launch Instance**:
   * Click **Launch Instance**.
   * **AMI**: Choose **Ubuntu 20.04 LTS**.
   * **Instance Type**: Select the free tier (t2.micro) or any other instance type.
   * **Key Pair**: Either create a new key pair or use an existing one. Ensure you **download the PEM file** (you will need it for SSH access).
   * **Network Settings**: Select the same **VPC** and **Availability Zone** (us-east-1a) as your RDS instance.
   * **Security Group**: Either create a new security group (allow SSH access from your IP) or select an existing one.
3. **Launch the EC2 Instance**: After reviewing, click **Launch**.

**4. Attach IAM Role to EC2 Instance**

1. **Create IAM Role**:
   * In the AWS Management Console, search for **IAM**.
   * Go to **Roles** and click **Create role**.
   * Select **AWS service** as the trusted entity and choose **EC2**.
   * Under **Permissions**, search for and select **RDSFullAccess** (if you have this pre-created) or select the custom role labinstanceprofile if specified.
   * Name the role and click **Create role**.
2. **Attach the Role to the EC2 Instance**:
   * Go back to your **EC2 Dashboard**.
   * Select your running EC2 instance.
   * In the **Actions** menu, choose **Security** → **Modify IAM role**.
   * Select the IAM role (either labinstanceprofile or RDSFullAccess) and click **Update IAM role**.

**How to Do It? - Step 2**

**1. SSH into the EC2 Instance**

Once your EC2 instance is up and running, SSH into it. Here's how:

1. Open your terminal (on Mac/Linux) or **PowerShell** (on Windows).
2. Navigate to the directory where your **.pem** file is located.
3. Run the following command (replace your-key.pem and your-ec2-public-ip with your actual key file and EC2 public IP):

bash

Copy code

chmod 400 your-key.pem ssh -i "your-key.pem" ubuntu@your-ec2-public-ip

This will log you into the EC2 instance.

**2. Install MySQL Client**

Once logged into your EC2 instance, update the system and install the MySQL client:

bash

Copy code

sudo apt update sudo apt install mysql-client

**3. Connect to the RDS Instance & Create Data**

Now, you'll connect to your RDS instance:

1. **Download the employees.sql File**:

bash

Copy code

wget https://d6opu47qoi4ee.cloudfront.net/employees.sql

1. **Connect to MySQL RDS Instance**:

Replace [RDS-endpoint] with the actual endpoint of your RDS instance (you can find this in the RDS dashboard under **Connectivity & Security**).

bash

Copy code

mysql -h [RDS-endpoint] -u root -p

Enter the password (password).

1. **Create the Database and Import Data**:

Once you're in the MySQL prompt, run the following commands:

sql

Copy code

CREATE DATABASE employees; USE employees; SOURCE employees.sql;

1. **Verify the Data**:

After the script runs, you can check the tables and columns with:

sql

Copy code

DESCRIBE employees;

1. **Exit MySQL**:

Type exit to leave the MySQL prompt.

**How to Do It? - Step 3**

**1. Install Python and Required Libraries**

SSH into your EC2 instance and install the following dependencies:

bash

Copy code

sudo apt install python2-minimal sudo apt install python3-pip sudo pip3 install boto3 mysql-connector-python pymysql

In case you get locale errors, run the following commands:

bash

Copy code

export LC\_ALL="en\_US.UTF-8" export LC\_CTYPE="en\_US.UTF-8" sudo dpkg-reconfigure locales

**2. Download and Modify the Python Script**

1. **Download the Python Script**:

bash

Copy code

wget https://d6opu47qoi4ee.cloudfront.net/rds.py

1. **Modify the Python Script**:
   * Open the Python script (rds.py) using a text editor like nano or vi:

bash

Copy code

nano rds.py

* + Replace any placeholders (like TBD) with the correct values, especially the **RDS endpoint**.

1. **Run the Python Script**:

bash

Copy code

python3 rds.py

1. **Observe the Behavior**:
   * If you remove the IAM role or change the RDS security group, the program may fail to connect. If you do this, the script will raise errors related to access permissions.

**How to Do It? - Step 4 (Advanced)**

**1. Create a Redis Cluster**

1. Go to **Elasticache** in the AWS Management Console.
2. Create a **Redis** cluster (self-managed or managed).
3. Once created, note the endpoint and port.

**2. Modify the Python Script to Use Redis**

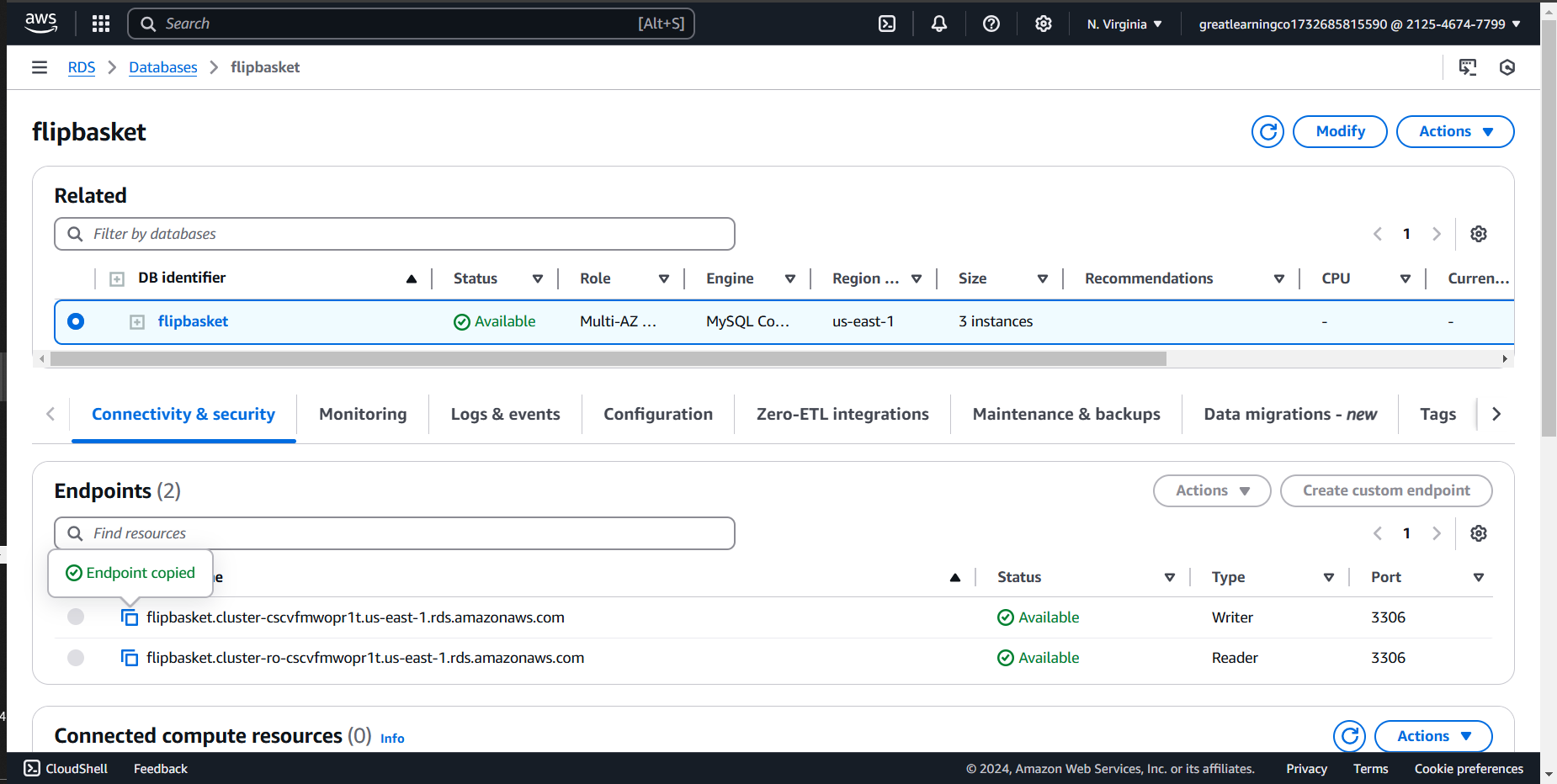
1. Install the Redis module:

bash

Copy code

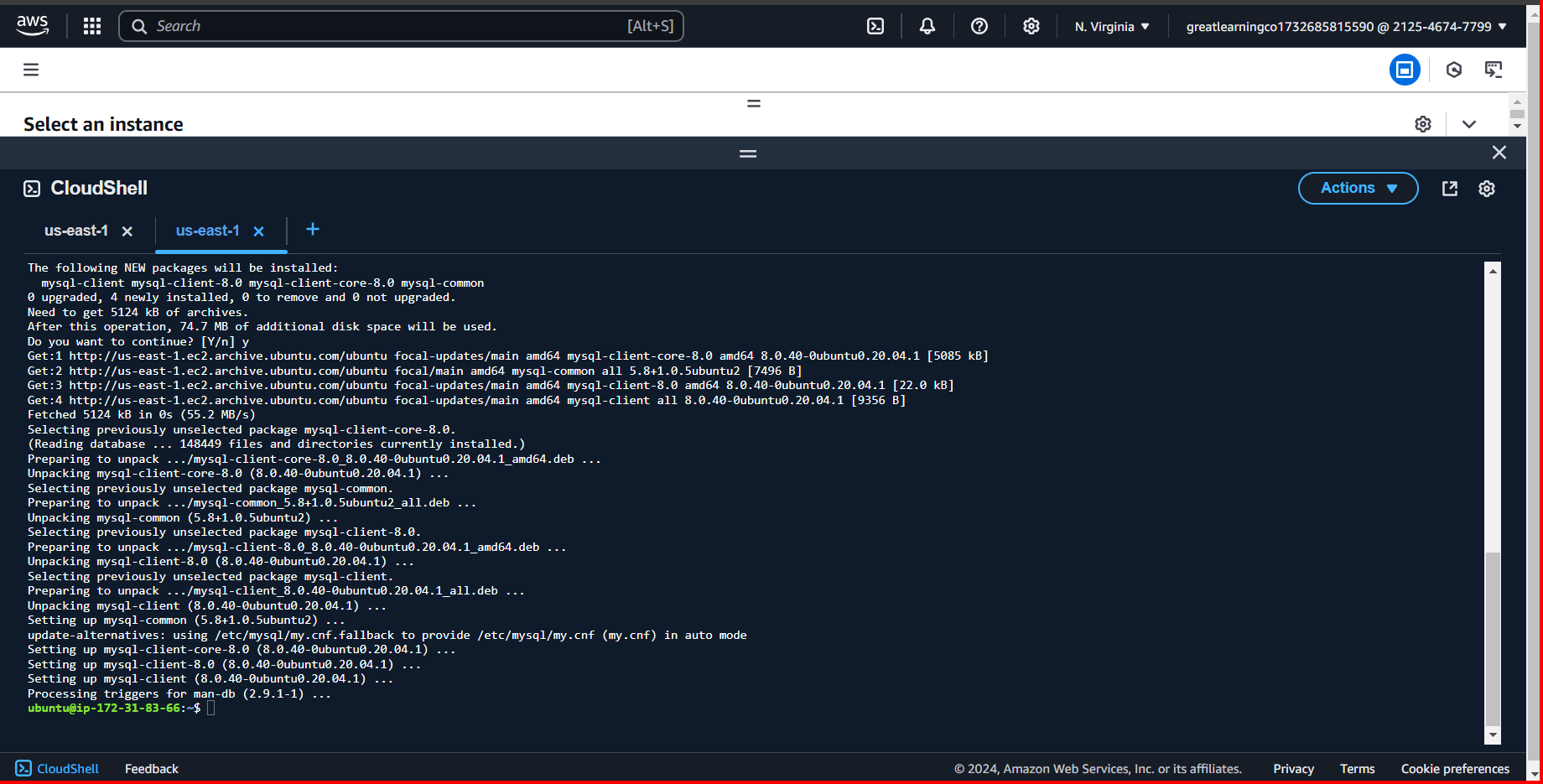
sudo pip3 install redis

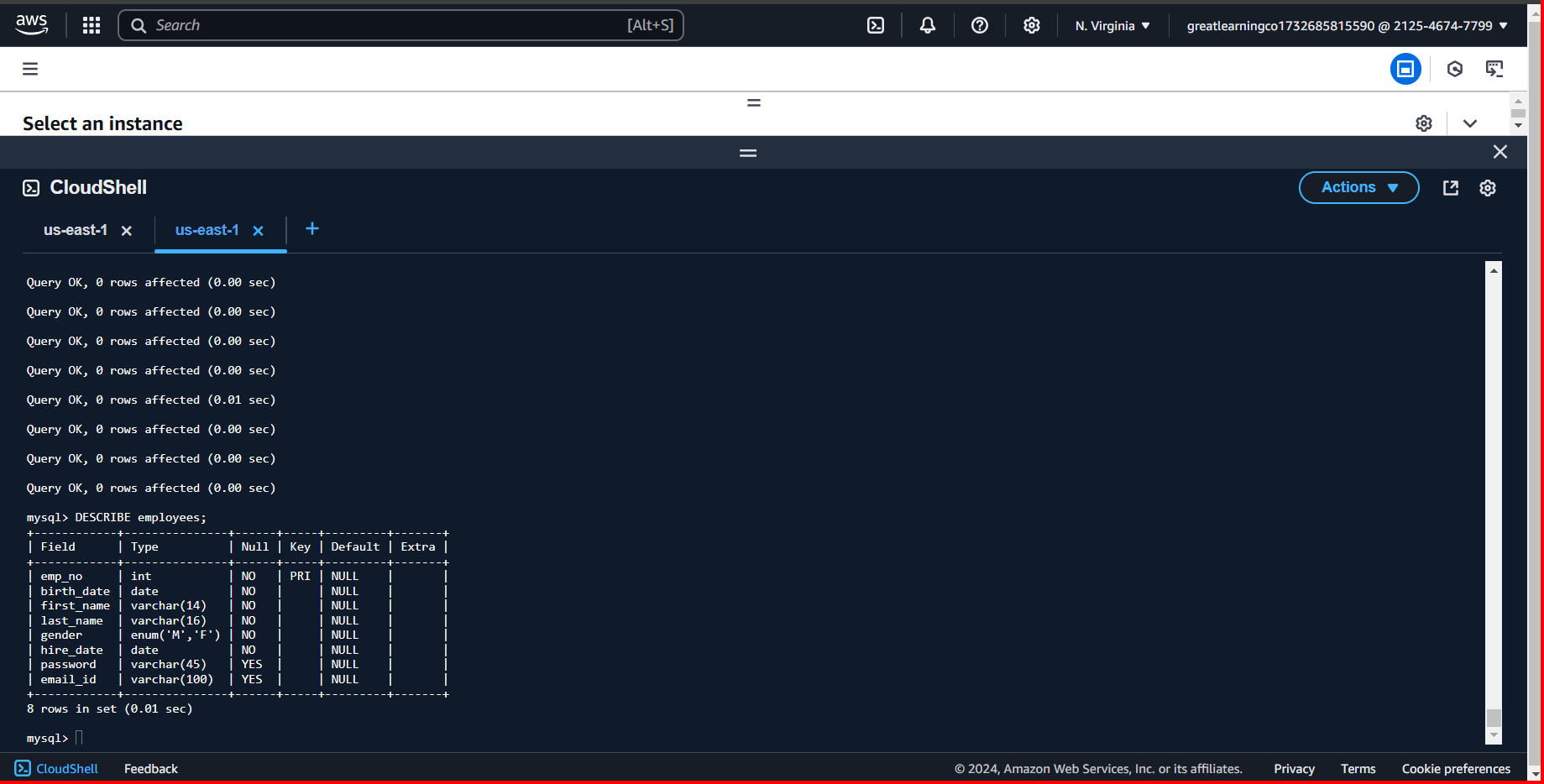
1. Modify your Python script to include a Redis cache. Use the redis and pickle modules to store and retrieve data from the cache. Implement the getOrder(...) method to check if the data is in the cache, and if not, fetch it from the database and store it in the cache.

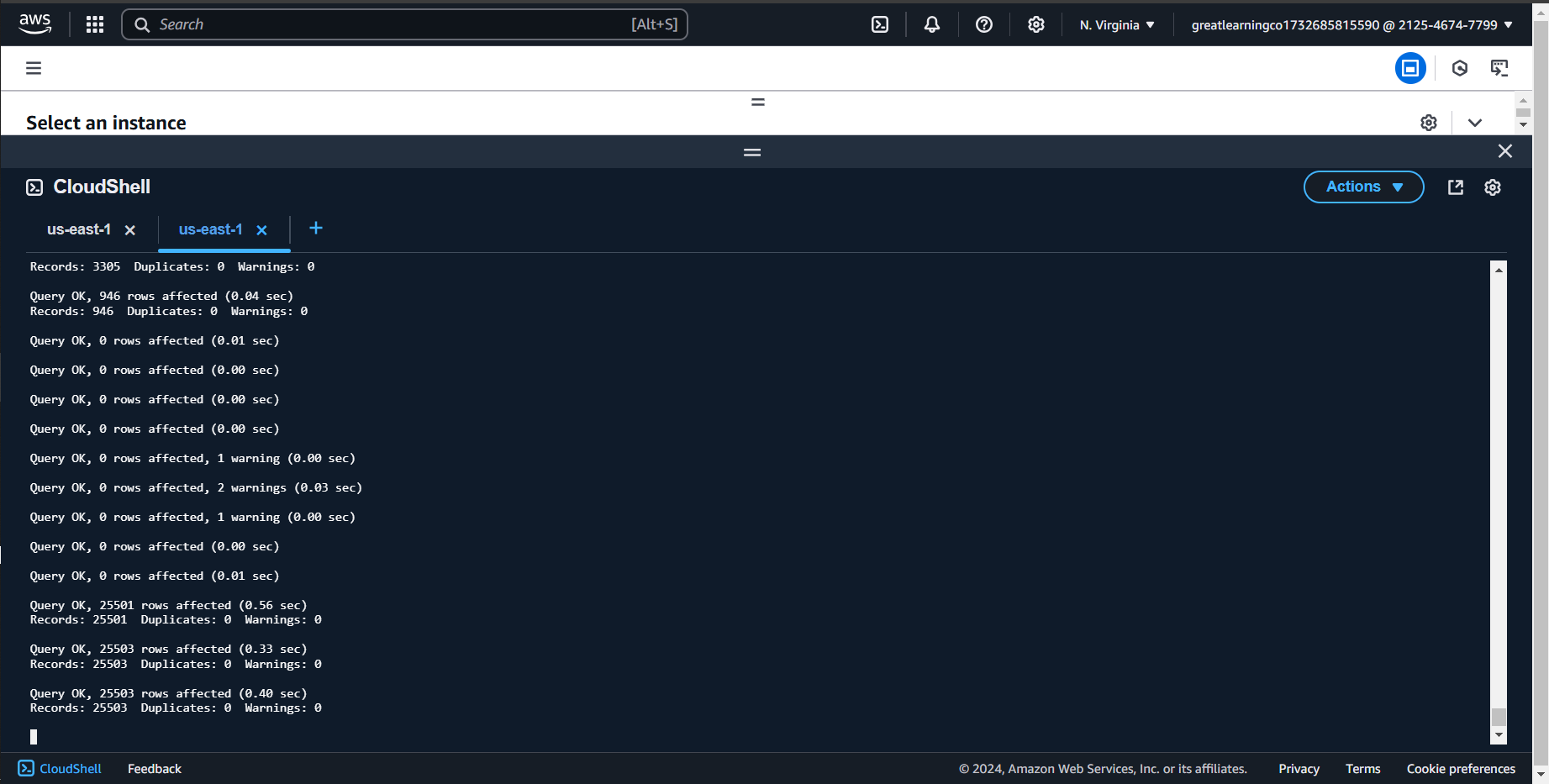


A screenshot of a computer

Description automatically generated







A screenshot of a computer

Description automatically generated

