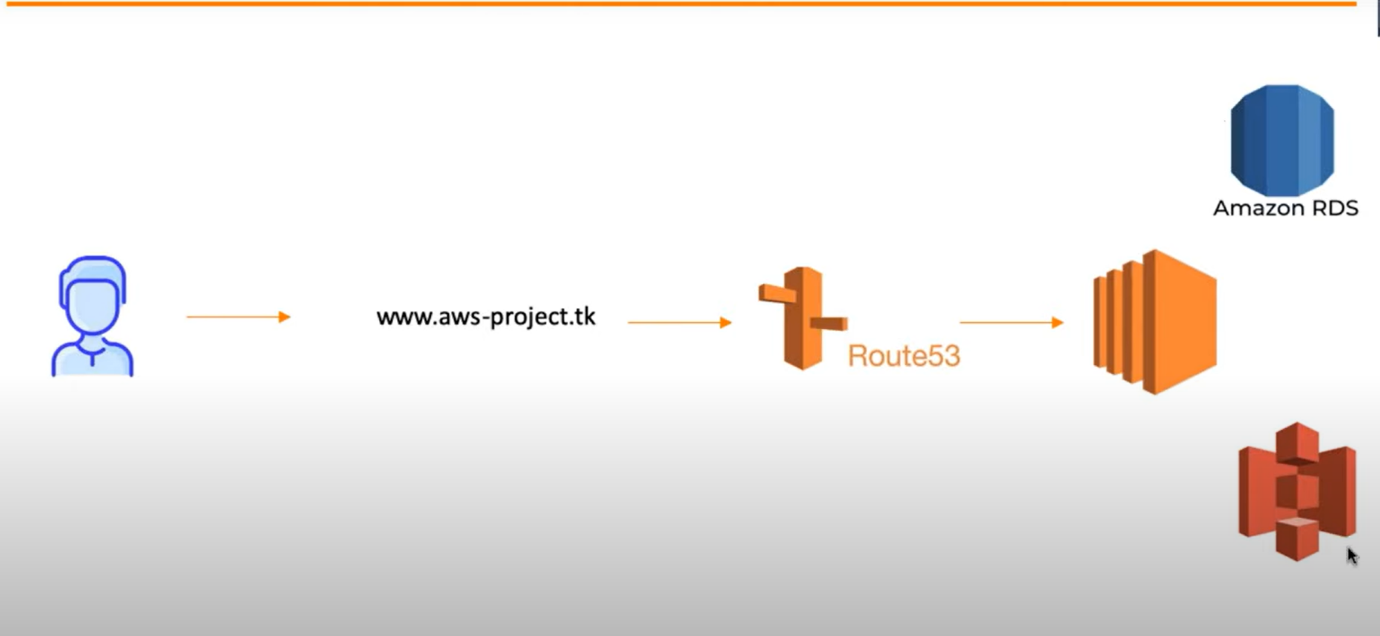
**Aim**: Deploying an end-to-end website on AWS.

**System Requirements:** AWS (RDS,EC2 instance,S3 Bucket,Route53),Putty,Python,Mysql.

**Overview of the Project:**

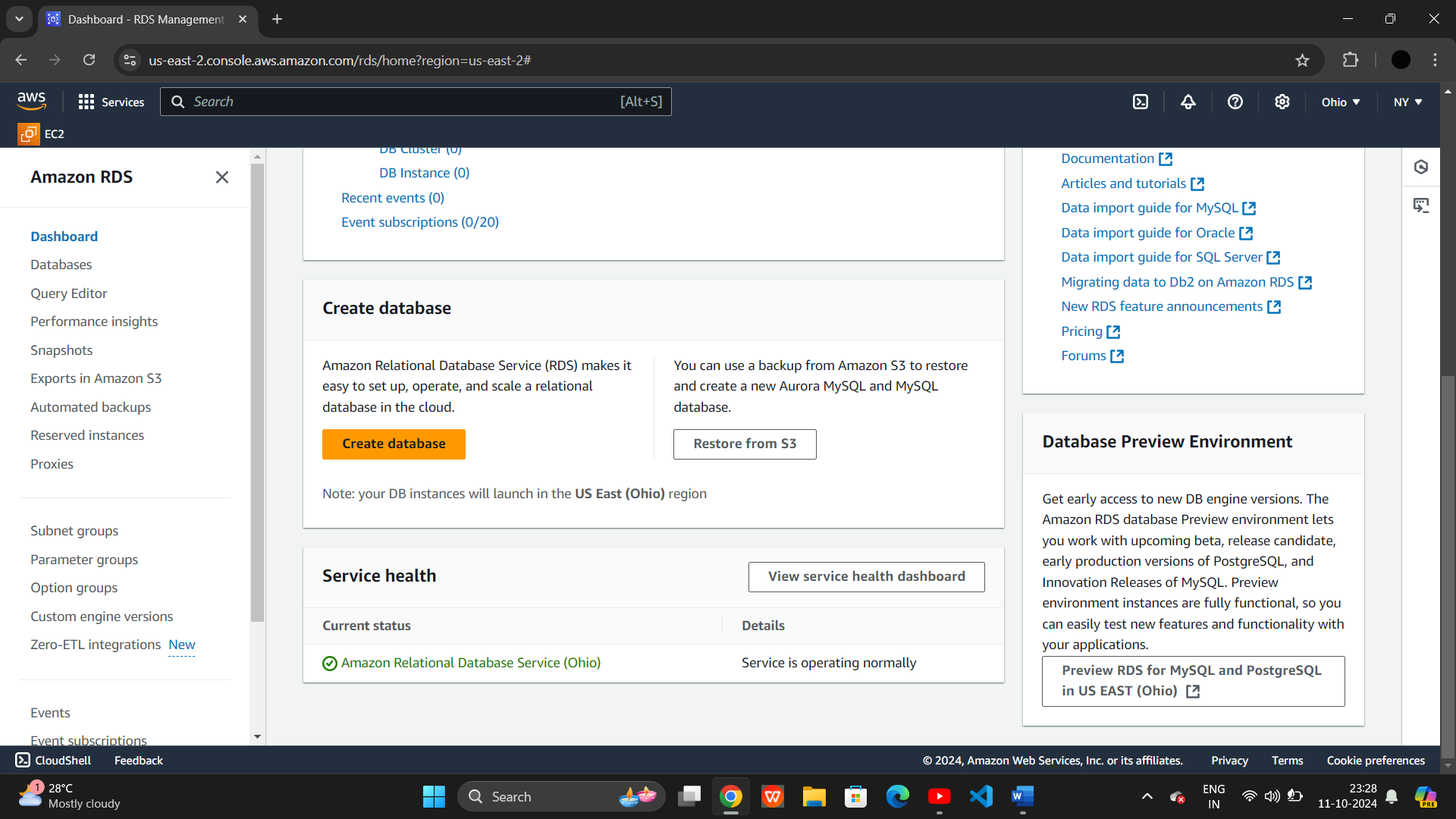


**Step1:** Deploy Mysql database on AWS RDS.

**Short info about Amazon RDS:Aim:**

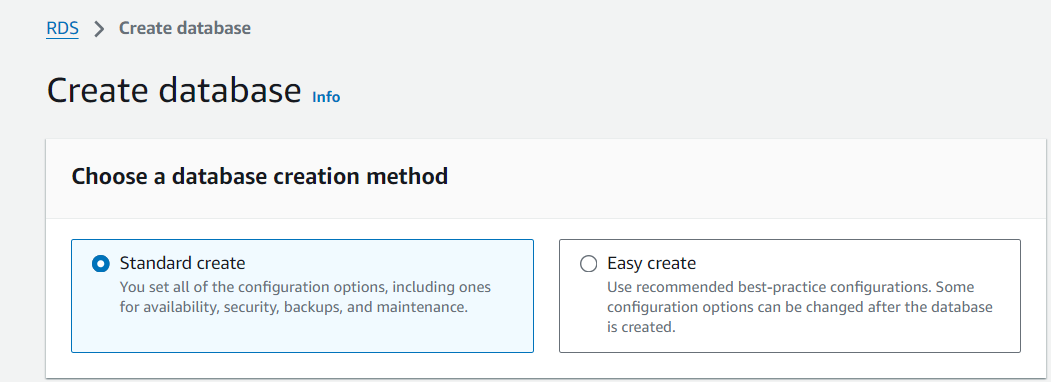
Amazon RDS for MySQL is a managed database service that automates administrative tasks such as backups, patching, and scaling. It offers support for high availability through Multi-AZ deployments, read replicas for performance optimization, and ensures security with data encryption and access control. The service is ideal for scalable web applications, e-commerce platforms, and analytics, providing flexible pricing options and support for various MySQL versions.

🡺interface of Amazon RDS

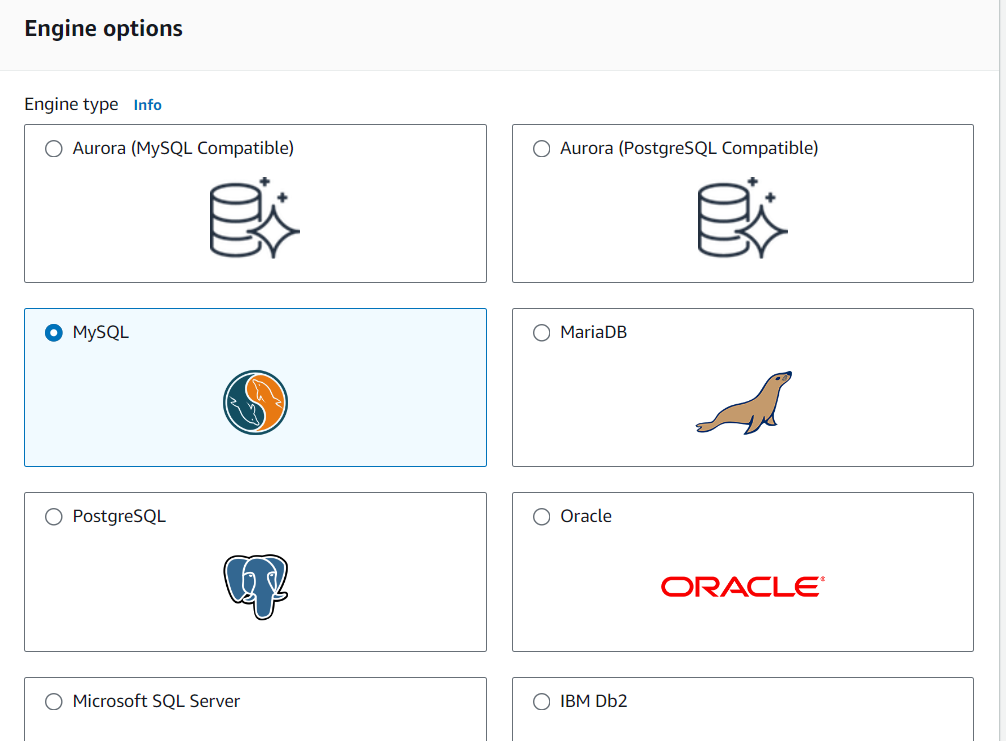


🡺Once you went to rds interface click on create database

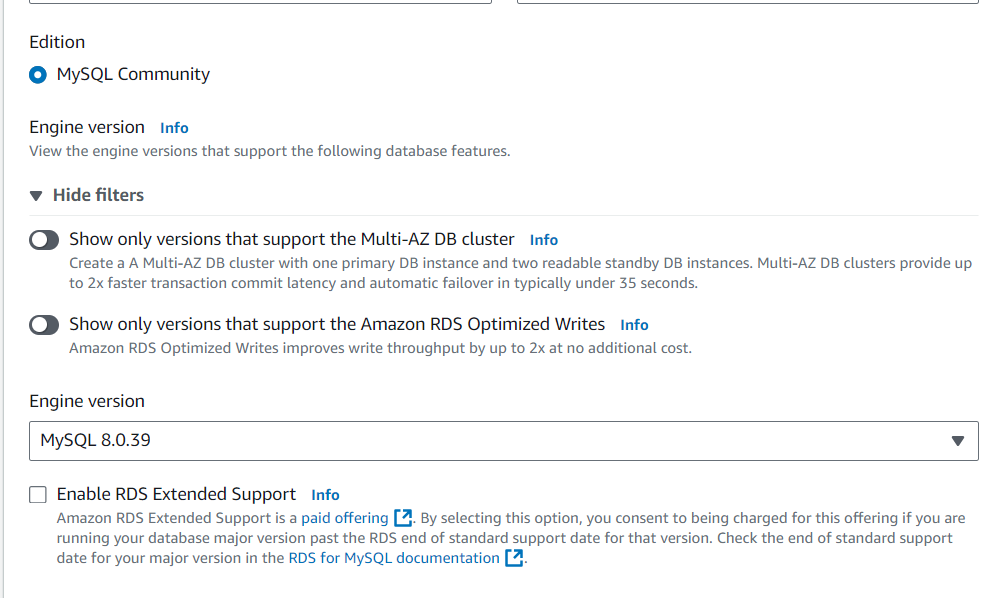
🡺 Click on standard create which is default.



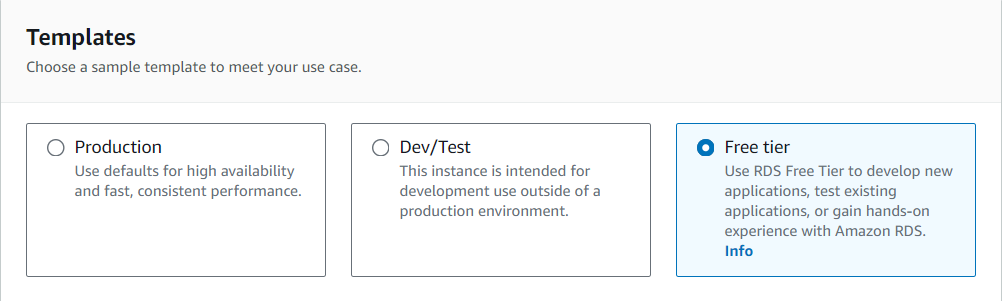
🡺In Engine options click on Mysql server.



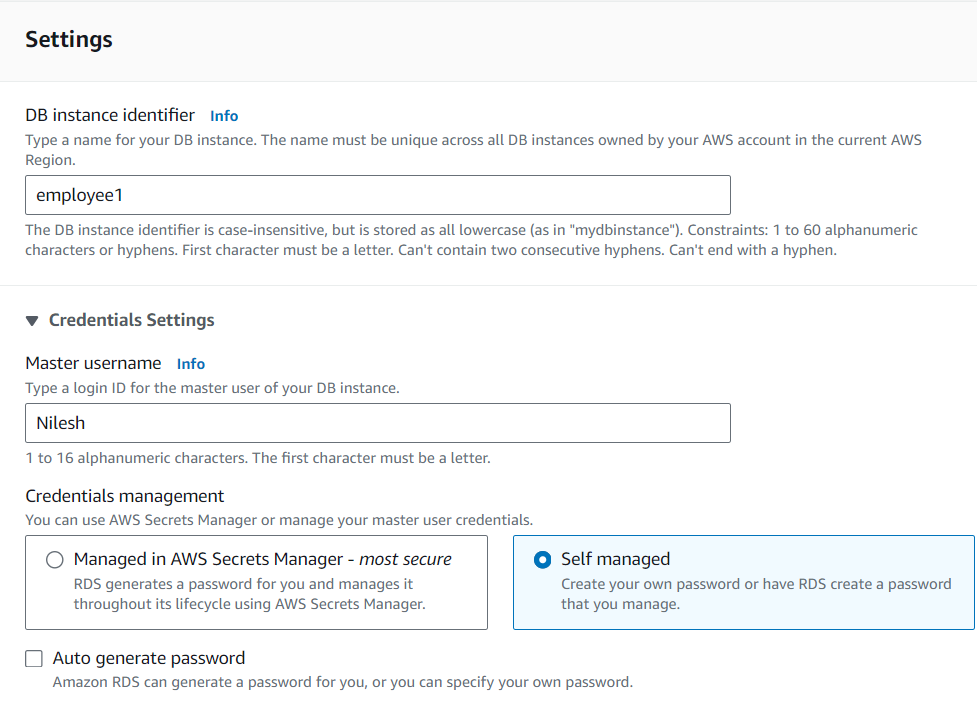
🡺Now, Our Edtion is MYSQl Community and Engine version is MYSQL 8.0.39.

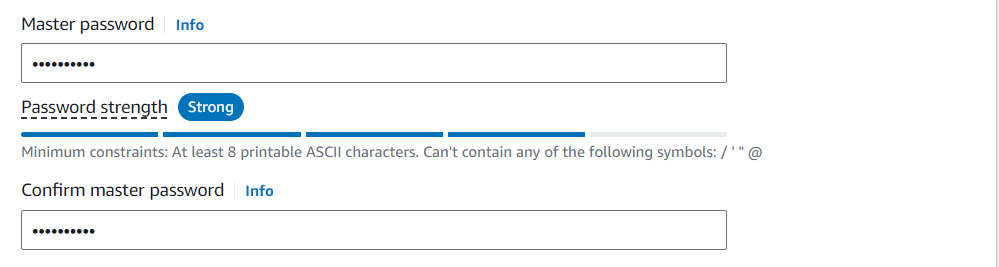


🡺 Now,as we practice on AWS we are using Free Tier to develop new applications.

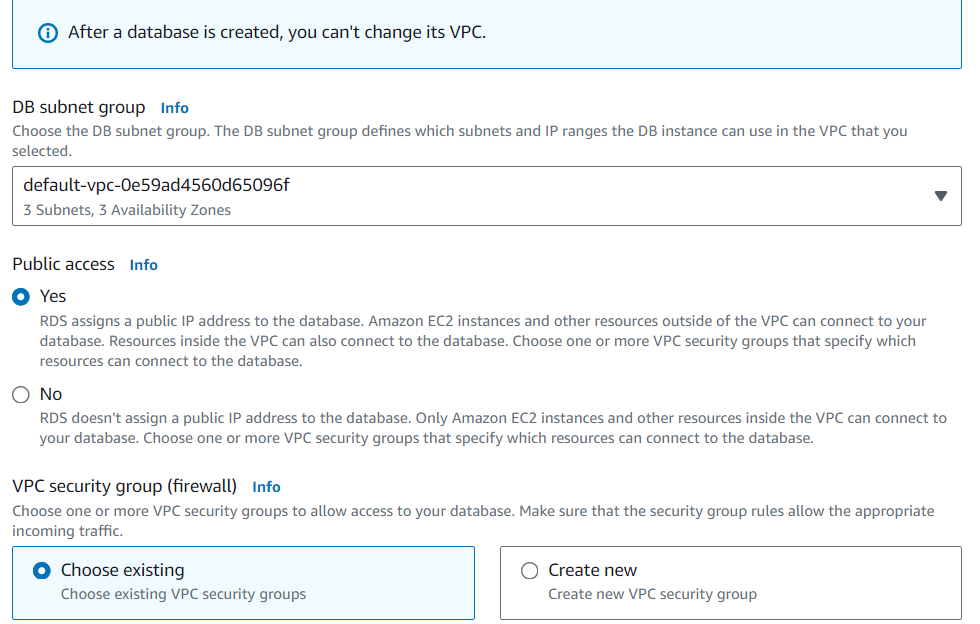


🡺Giving my DB instance name as **employee1** and username as **Nilesh** and set a Password

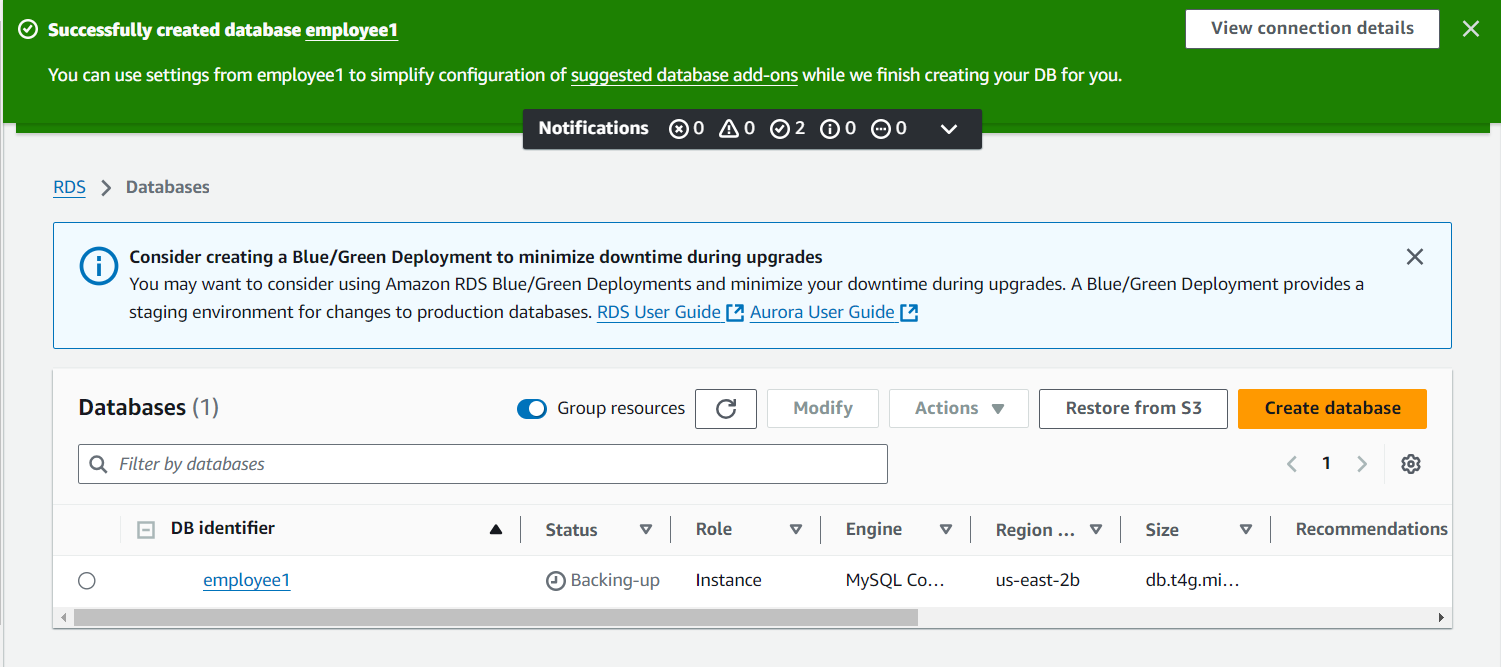




🡺Set Subnet grp as default and make it public accessibly



🡺Now,we successfully created database with employee1.

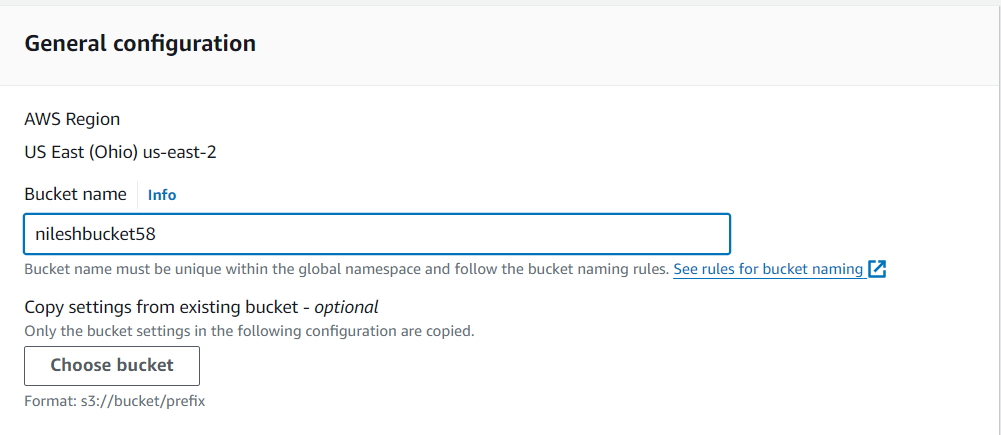


**Step2:** Setup S3 bucket for storing our objects.

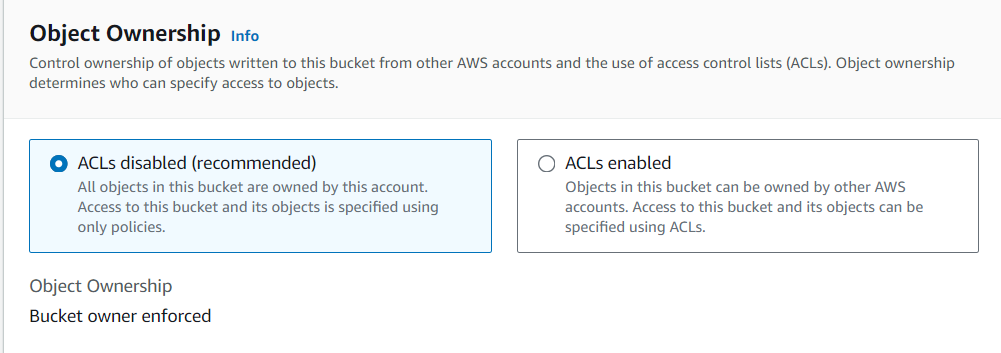
**Short info Amazon S3 Bucket:**

Amazon S3 is a scalable object storage service where data is stored in buckets. It offers unlimited storage, high availability, and 99.999999999% (11 9's) durability. Key features include encryption for security, versioning to track changes, lifecycle management for cost optimization, and flexible access controls. S3 integrates with other AWS services and is commonly used for backups, data lakes, media storage, and hosting static websites.

🡺 Now giving my bucket name as **nileshbucket58** and one imp thing is that the region for bucket is US East(ohio) which is same as Amaxon RDS.

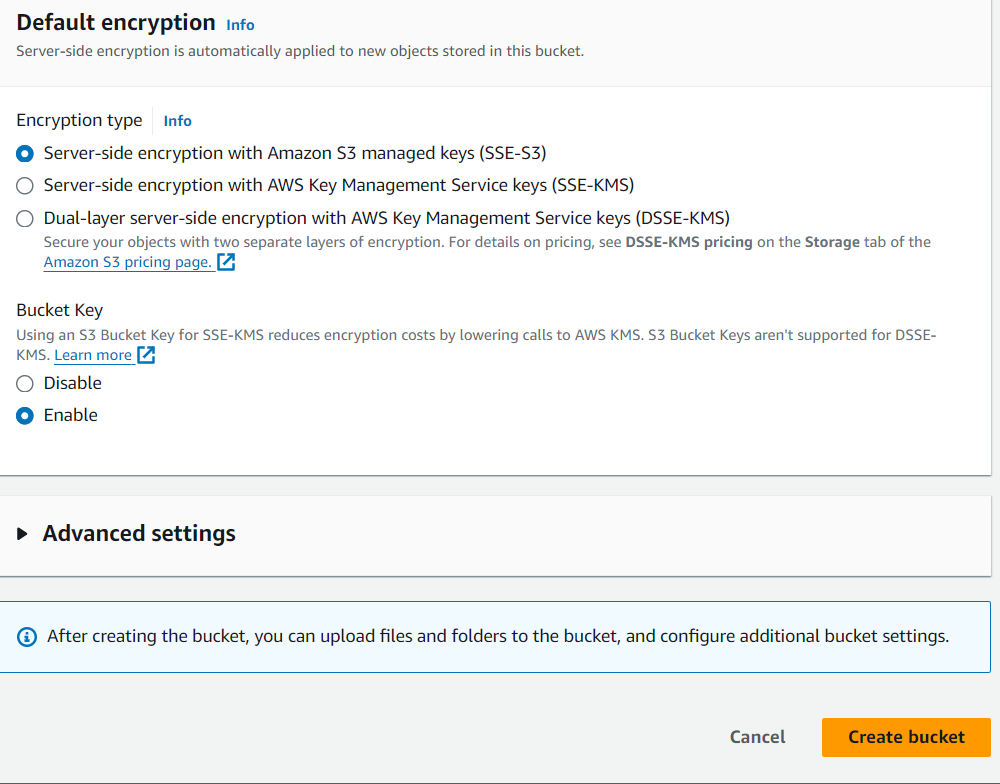


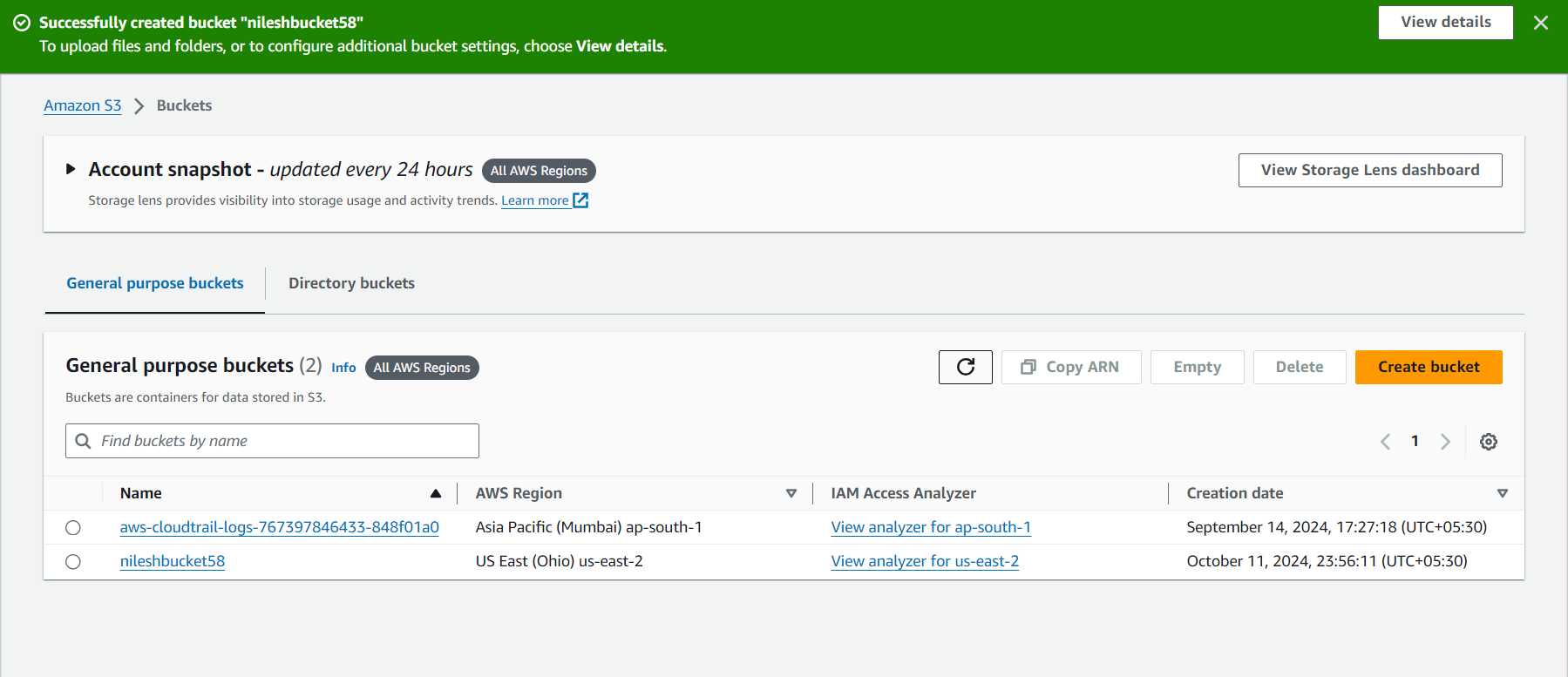
🡺As all objects are owned by me so we choose **ACLs disabled** if objects are owned other aws account too then we enabled that.



🡺As we disabled ALCs it will default blocked the public access settings for this bucket.

🡺In next step,data encryption is by default and click on create bucket.

🡺 Now, our bucket is successfully created with the name nileshbucket58.

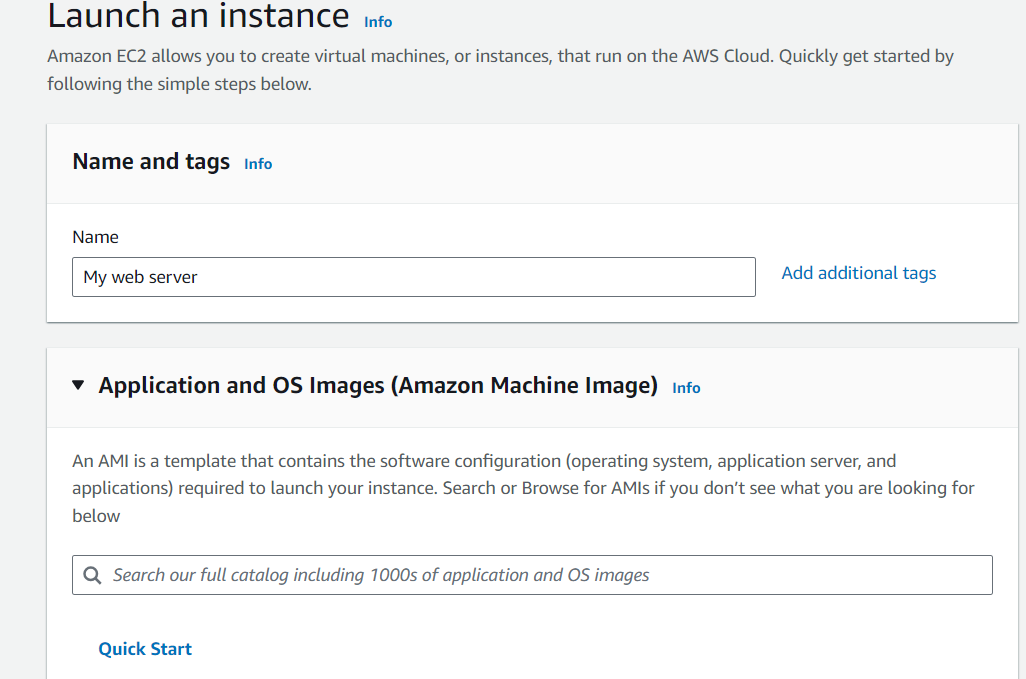


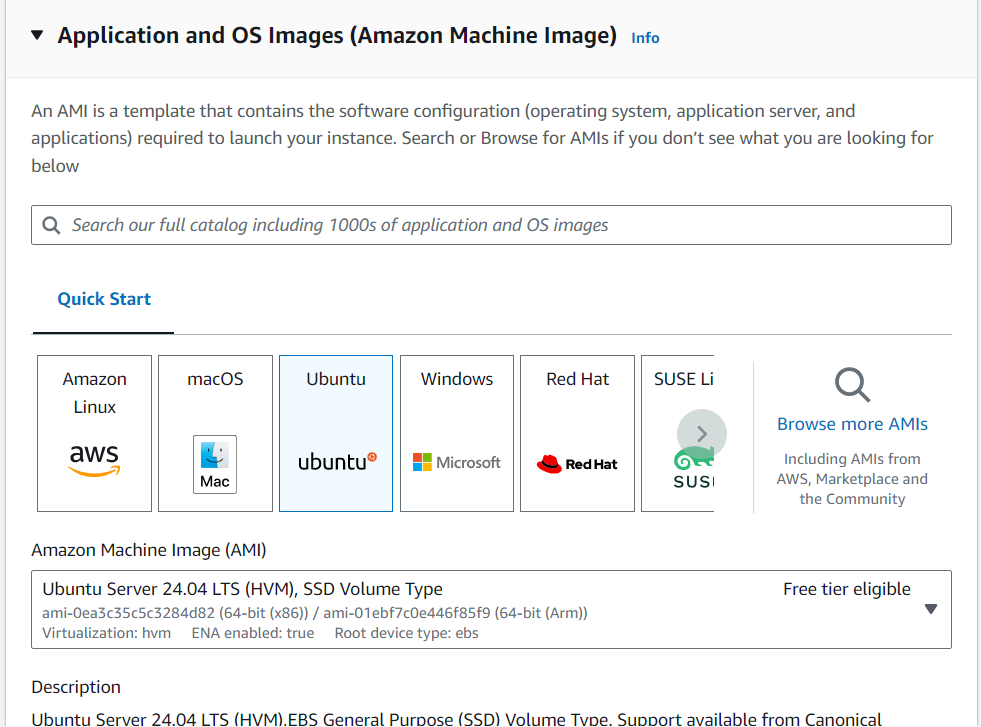
**Step3:** Creating an EC2 Instance.

**Short info EC2 instance:**

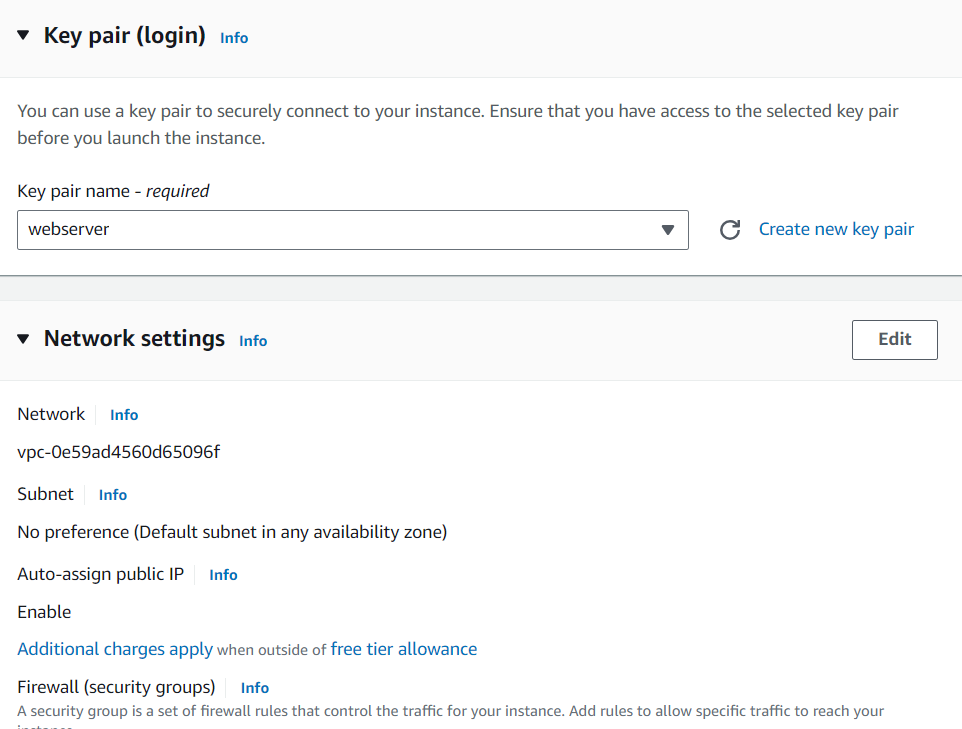
Amazon EC2 provides scalable virtual servers (instances) in the cloud, offering flexibility in computing power, memory, and storage to meet different application needs. Key features include a variety of instance types, elasticity to scale up or down, strong security integration, customizable operating systems, and multiple pricing models (on-demand, reserved, and spot instances). EC2 is widely used for web hosting, application deployment, and complex workloads like machine learning.

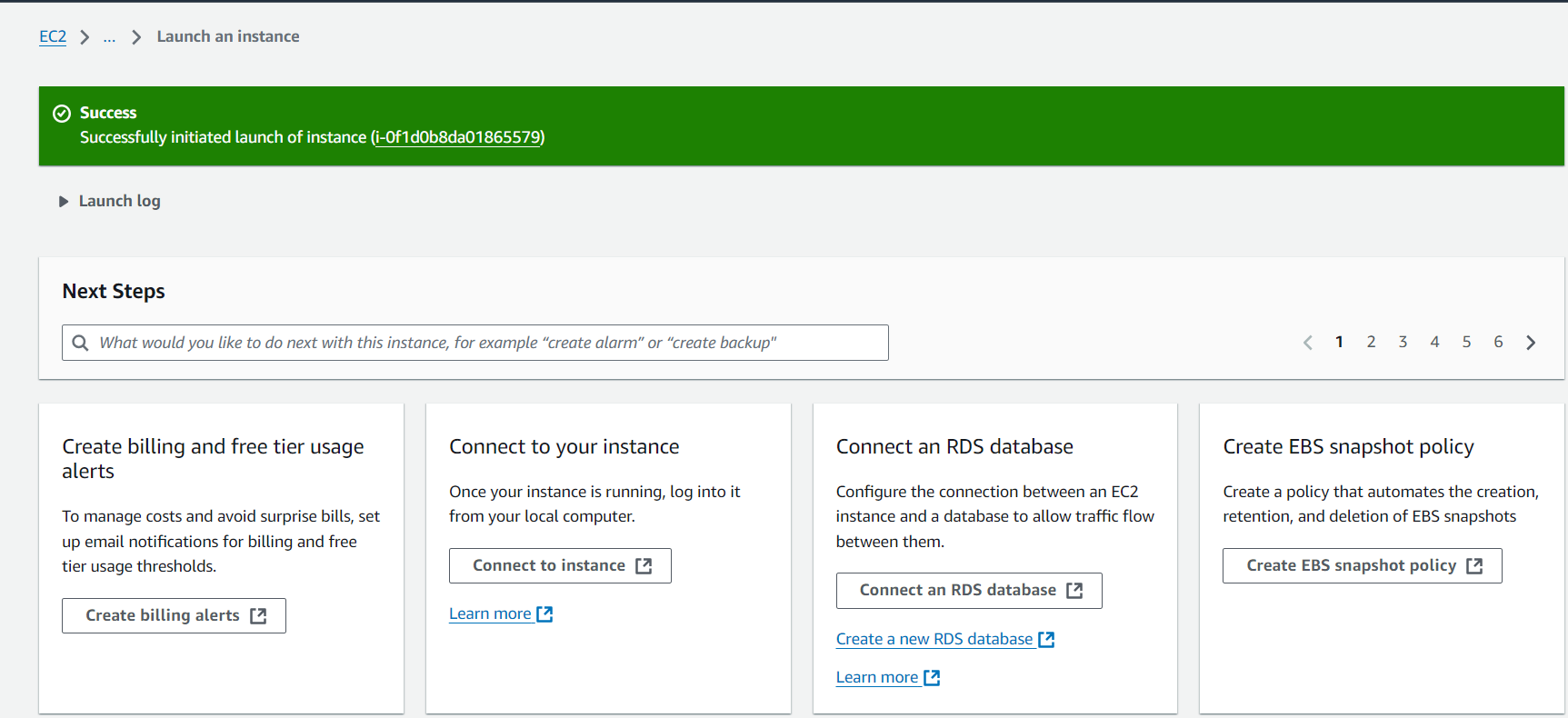
🡺Launch an instance with name my web server

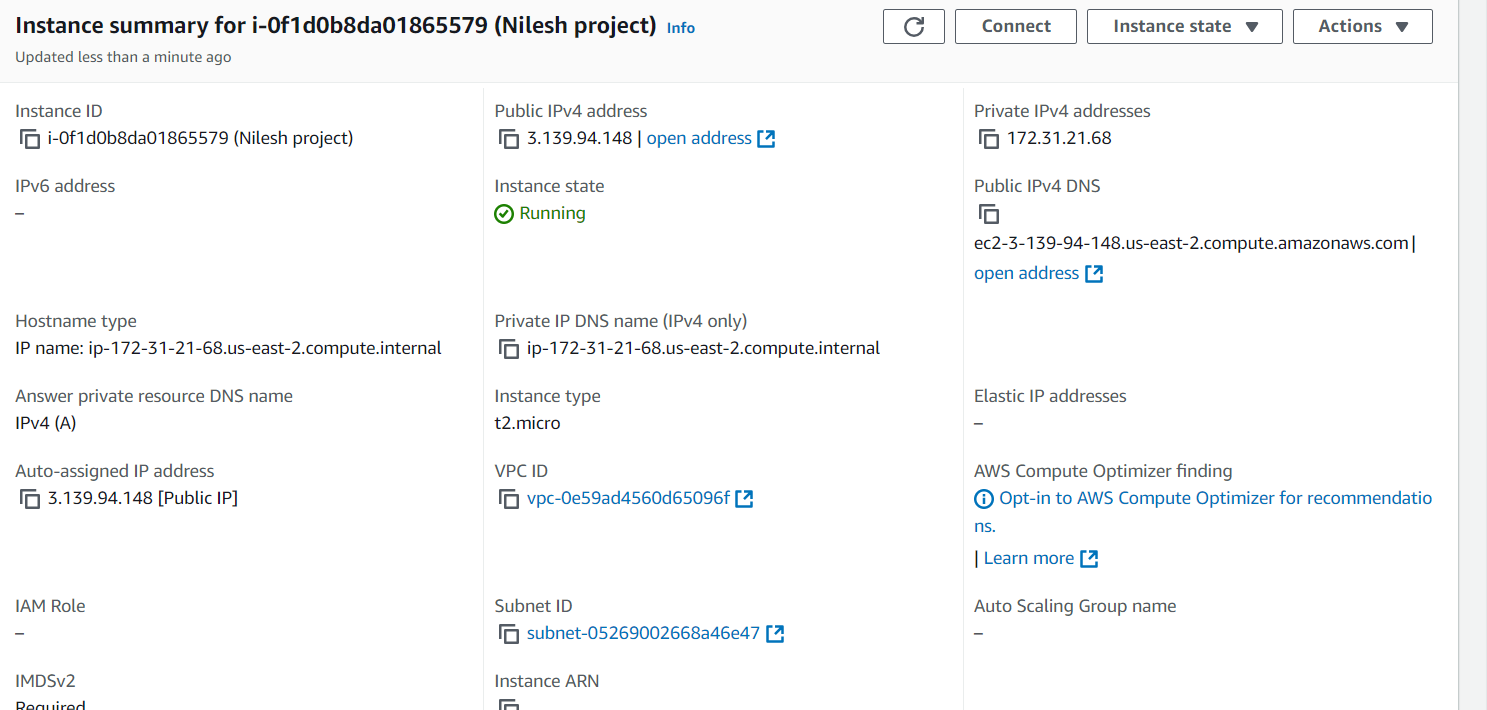
🡺In application and OS image we choose **ubuntu.**



🡺Generate a key pair with name **webserver.**

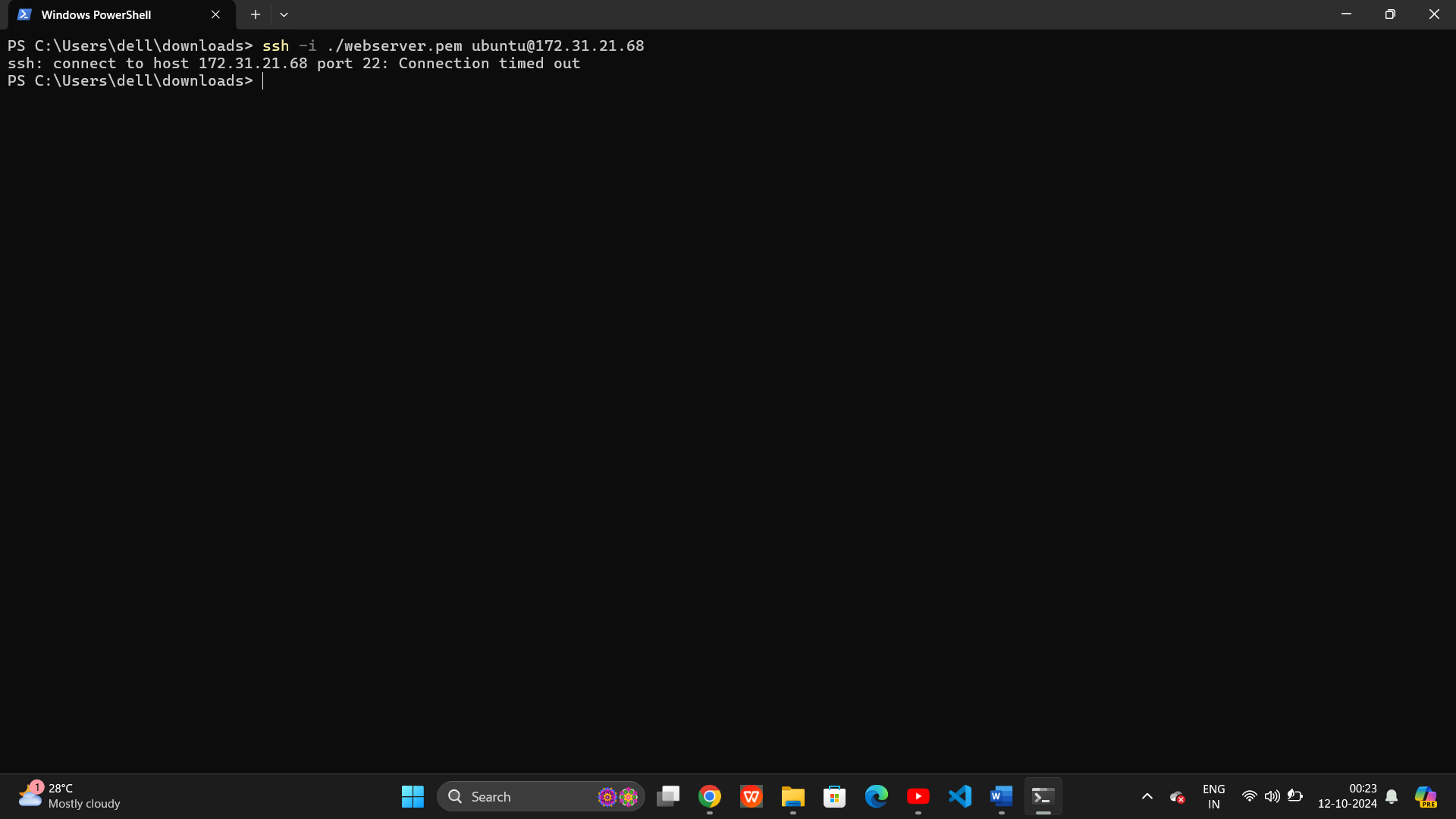
🡺 we successfully created our instance.

🡺properties of our instance.



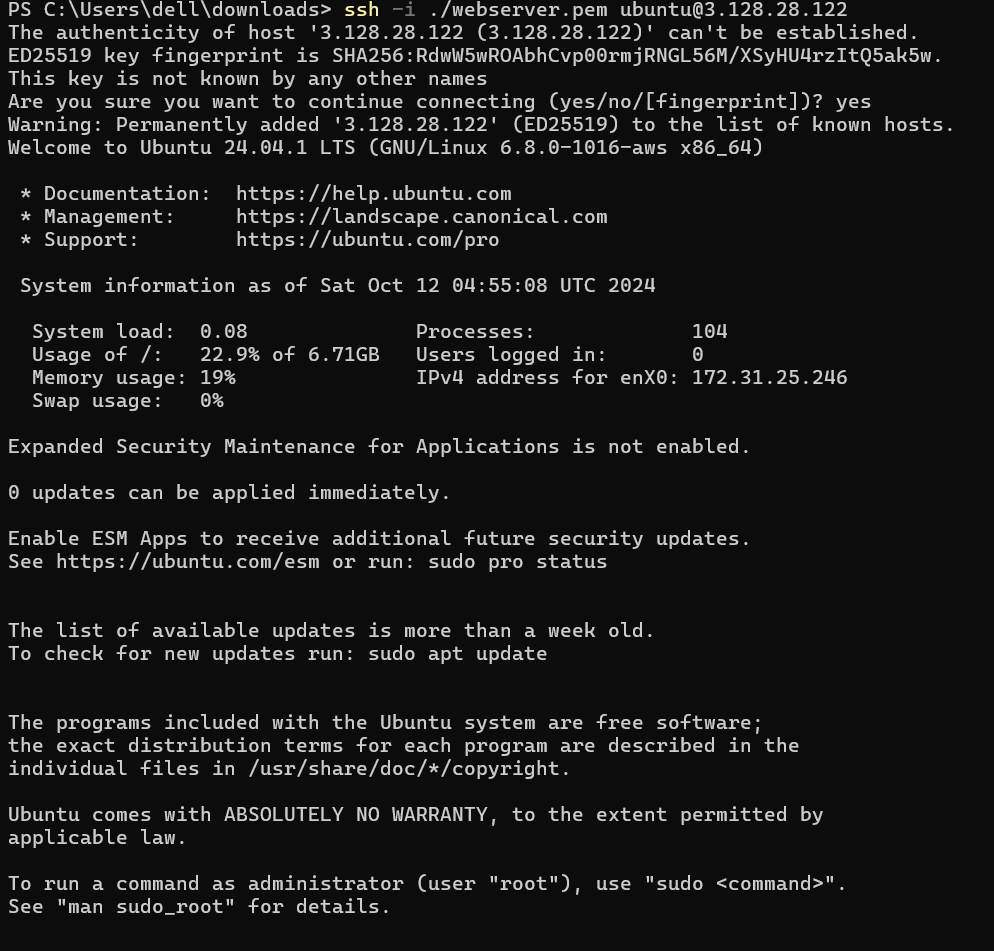
Step4: Connetion btwn instance and local host.

🡺To do so firstly copy the public ip address of instance and went to terminal.



🡺Now,to do so the instruction command is **ssh -i ./keypair name.pem ubuntu@your public ip address.**

In our case ,the command is **ssh -i ./webserver.pem ubuntu@3.128.28.122.**



🡺After performing this we connected to server but the issue is that instruction works on linux not on window,to do on window to connect ssh into an instance we have to download a software called **Putty.**

Now,steps to install Putty and setup for that.

To connect to a Linux instance deployed on AWS using **PuTTY**, you need to follow a few key steps. Since PuTTY does not directly support the .pem (Privacy-Enhanced Mail) key file format used by AWS, you’ll also need to convert your .pem file to a .ppk (PuTTY Private Key) format. Here's a detailed guide:

**Step-by-Step Guide to Connect PuTTY to AWS Linux Instance**

**Step 1: Download and Install PuTTY**

1. **Download PuTTY** from the official site: [PuTTY Download Page](https://www.putty.org/).
2. **Install PuTTY** on your Windows system.

You’ll also need **PuTTYgen**, which is included with the PuTTY installation. This tool helps convert the .pem file to .ppk.

**Step 2: Convert .pem Key to .ppk Using PuTTYgen**

AWS provides the key pair file in .pem format, but PuTTY requires it in .ppk format.

1. Open **PuTTYgen** (Search for "PuTTYgen" in your system).
2. Click **Load** and select the .pem file you downloaded when creating the key pair for your instance.
   * In the file explorer, you may need to change the file type to "All Files" to see the .pem file.
3. After successfully loading the .pem file, click **Save private key**.
   * You can optionally set a passphrase for additional security.
   * Save the file with a .ppk extension.

#### **Step 3: Get Public IP or DNS of Your AWS Instance**

1. Go to the **AWS Management Console**.
2. Navigate to the **EC2 Dashboard** and click on **Instances**.
3. Find your instance and copy the **Public IP** or **Public DNS** from the instance details.

#### **Step 4: Configure PuTTY for SSH Connection**

1. Open **PuTTY**.
2. In the **Host Name (or IP address)** field, enter the **Public IP** or **Public DNS** of your AWS instance. Example:
   * ec2-XX-XXX-XXX-XX.compute-1.amazonaws.com or X.X.X.X
3. **Port**: Ensure the port is set to 22 (SSH default).
4. **Connection Type**: Make sure SSH is selected.

#### **Step 5: Configure the Private Key for Authentication**

1. In PuTTY, on the left-hand side menu, expand **Connection** → **SSH** → **Auth**.
2. Click **Browse** and select the .ppk file you created in step 2.

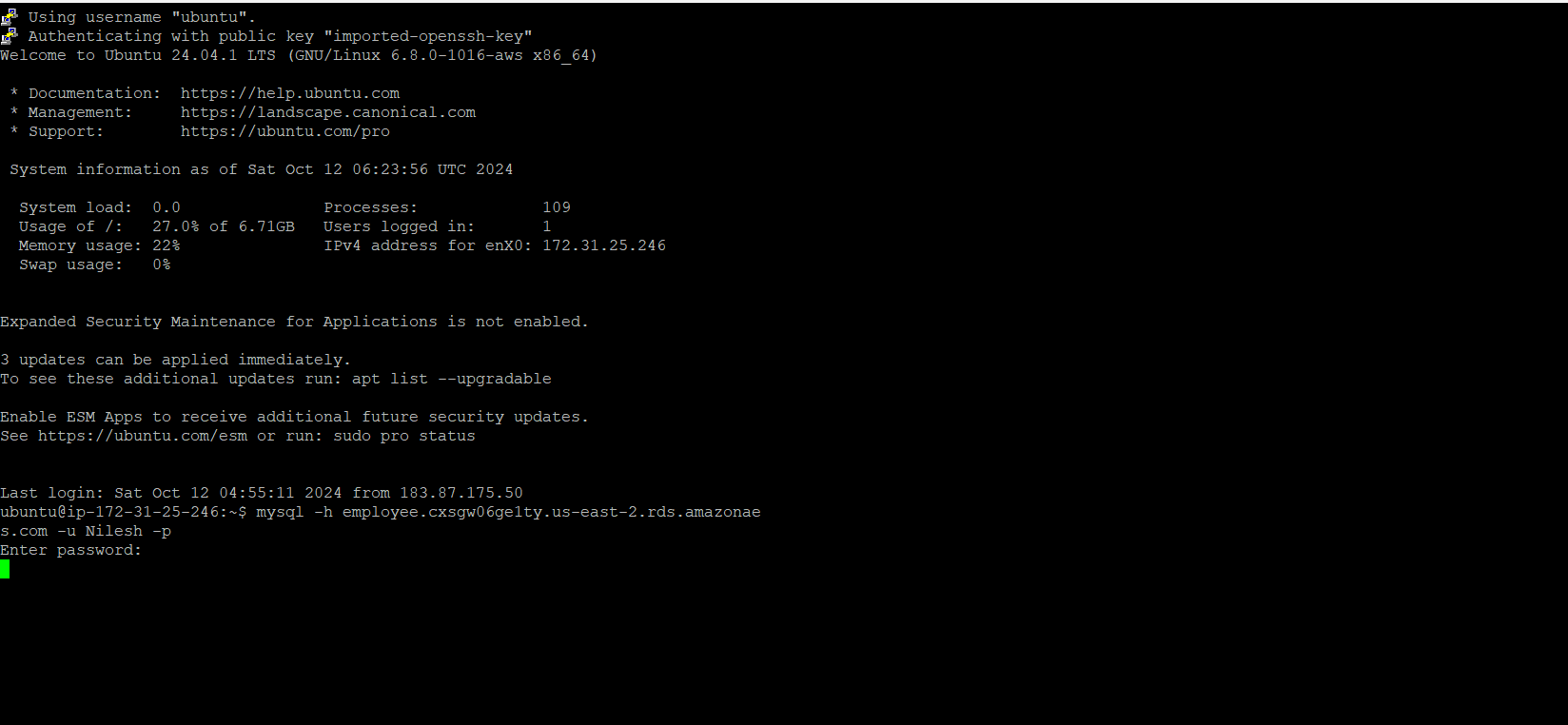
#### **Step 6: Save Your Session (Optional)**

1. In the left-hand side of PuTTY, go back to the **Session** category (at the top).
2. In the **Saved Sessions** field, type a name (e.g., AWS Instance).
3. Click **Save** to reuse the configuration in the future.

#### **Step 7: Connect to the Instance**

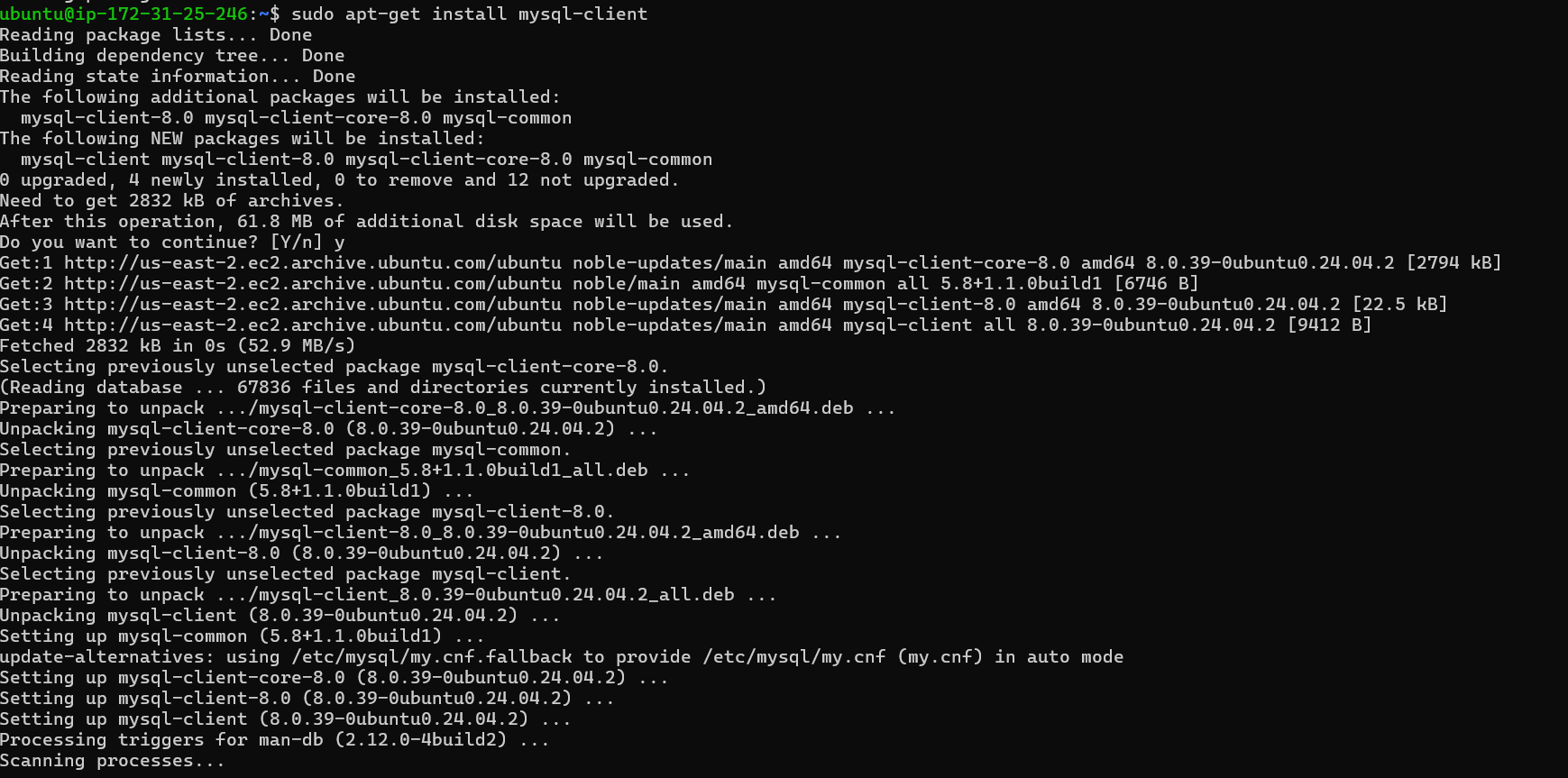
1. Click **Open** in PuTTY to initiate the connection.
2. If this is your first time connecting to this instance, PuTTY will show a **security alert** about the server’s host key not being cached. Click **Yes** to accept the connection.
3. You’ll be prompted to enter a **username**.

🡺Now, I login as **ubuntu** and these is the interface.



🡺Now,connecting to my rds to check everything is fine or not in network point of you

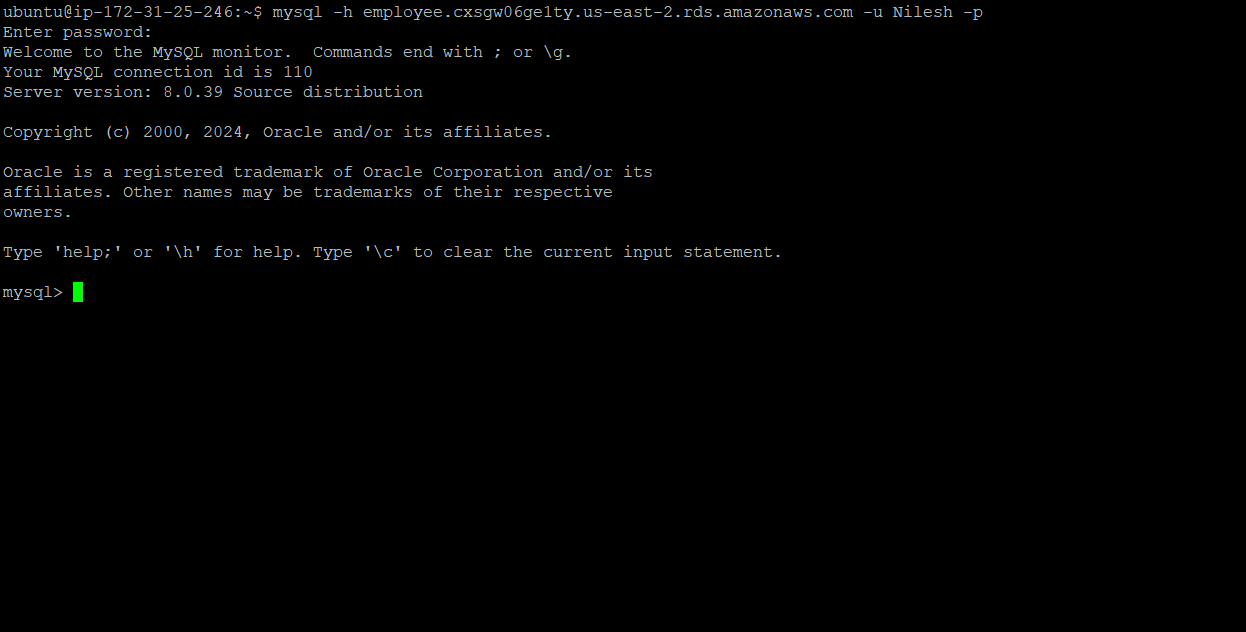
🡺**install mysql-client** which help us to connect with RDS.



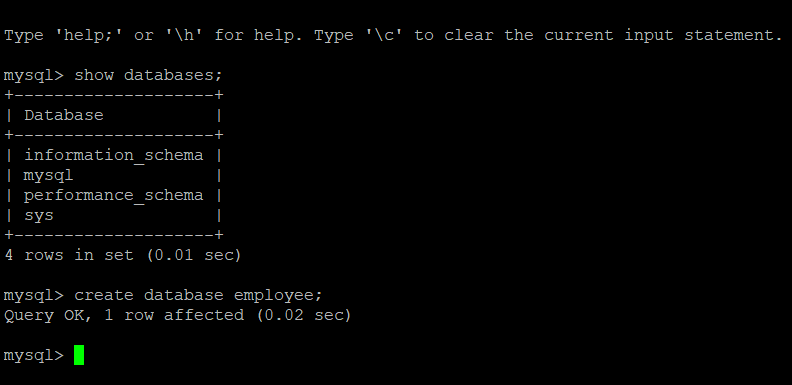
🡺After installing mysql-client we connect to our sql server through this command

**Mysql -h our db instance name.our rds end point -u username -p password**

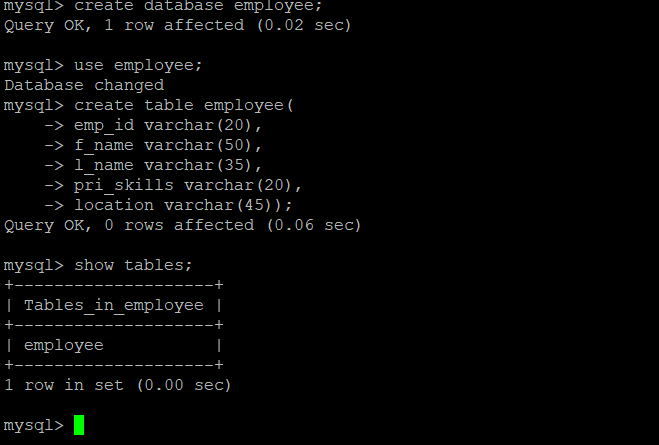
🡺 In our case ,the command is **mysql -h employee1.cxsgw06ge1ty.us-east-2.rds.amazonaws.com -u Nilesh -p**



🡺We successfully conneted to sql server now checking it is successfully connected or not.



🡺Creating a table with the name employee.



🡺After creating a table we exit from our mysql server abd went to our local terminal.

🡺Now,for framework we use python

sudo apt-get install python3

sudo apt-get install python3-flask

sudo apt-get install python3-pymysql

sudo apt-get install python3-boto3

# for running application

sudo python3 Empapp.py

from flask import Flask, render\_template, request

from pymysql import connections

import os

import boto3

from config import \*

app = Flask(\_\_name\_\_)

bucket = custombucket

region = customregion

db\_conn = connections.Connection(

    host='employee.cxsgw06ge1ty.us-east-2.rds.amazonaws.com',

    user='Nilesh',

    password='Nilesh19112005',

    db='employee',

    connect\_timeout=30  *# Increase the timeout to 30 seconds*

)

*# Establish the database connection*

db\_conn.connect()

output = {}

table = 'employee'

@app.route("/", methods=['GET', 'POST'])

def home():

    return render\_template('AddEmp.html')

@app.route("/about", methods=['POST'])

def about():

    return render\_template('www.Nilesh.com')

@app.route("/addemp", methods=['POST'])

def AddEmp():

    emp\_id = request.form['emp\_id']

    first\_name = request.form['first\_name']

    last\_name = request.form['last\_name']

    pri\_skill = request.form['pri\_skill']

    location = request.form['location']

    emp\_image\_file = request.files['emp\_image\_file']

    if not emp\_image\_file:

        return "Please select a file"

    if not emp\_image\_file.filename:

        return "Please select a file"

    insert\_sql = "INSERT INTO employee VALUES (%s, %s, %s, %s, %s)"

    cursor = db\_conn.cursor()

    try:

        cursor.execute(insert\_sql, (emp\_id, first\_name, last\_name, pri\_skill, location))

        db\_conn.commit()

        emp\_name = "" + first\_name + " " + last\_name

*# Uplaod image file in S3 #*

        emp\_image\_file\_name\_in\_s3 = "emp-id-" + *str*(emp\_id) + "\_image\_file"

        s3 = boto3.resource('s3')

        try:

            print("Data inserted in MySQL RDS... uploading image to S3...")

            s3.Bucket(custombucket).put\_object(Key=emp\_image\_file\_name\_in\_s3, Body=emp\_image\_file)

            bucket\_location = boto3.client('s3').get\_bucket\_location(Bucket=custombucket)

            s3\_location = (bucket\_location['LocationConstraint'])

            if s3\_location is None:

                s3\_location = ''

            else:

                s3\_location = '-' + s3\_location

            object\_url = "https://s3{0}.amazonaws.com/{1}/{2}".format(

                s3\_location,

                custombucket,

                emp\_image\_file\_name\_in\_s3)

        except *Exception* as e:

            print(f"Error uploading image to S3: {e}")

            return *str*(e)

    except *Exception* as e:

        db\_conn.rollback()

        print(f"Error inserting data into MySQL: {e}")

        return *str*(e)

    finally:

        cursor.close()

    print("all modification done...")

    return render\_template('AddEmpOutput.html', name=emp\_name)

if \_\_name\_\_ == '\_\_main\_\_':

    app.run(host='0.0.0.0', port=80, debug=True)

and more file **config.py**

customhost = "employee.cxsgw06ge1ty.us-east-2.rds.amazonaws.com"

customuser = "Nilesh"

custompass = "Nilesh19112005"

customdb = "employee1"

custombucket = "nileshbucket58"

customregion = "us-east-2a"

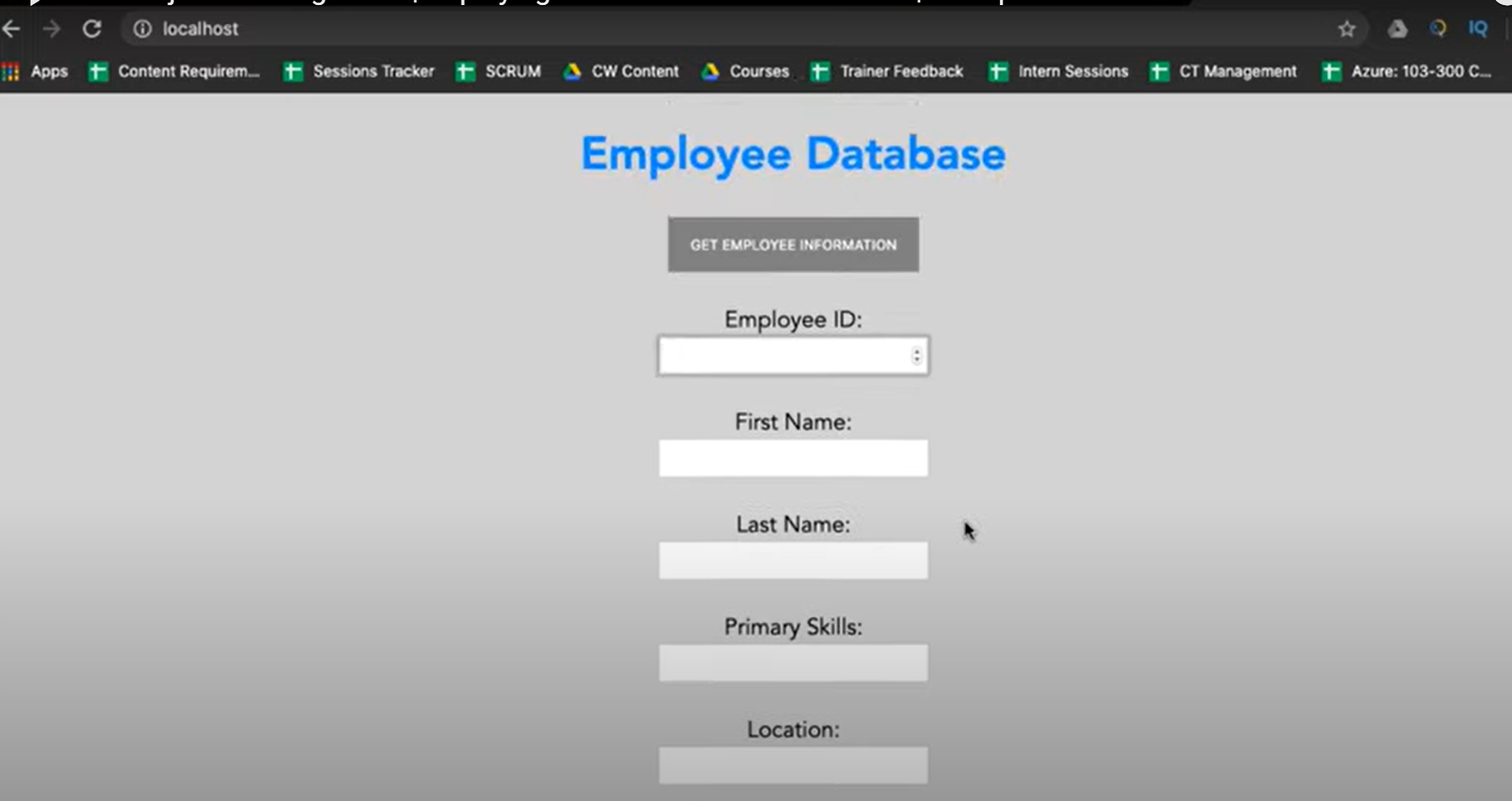
🡺Python module works

1. **python3**: The Python programming language (version 3.x), known for its readability and versatility, supporting various programming paradigms.
2. **python3-flask**: A lightweight web framework for Python that enables quick and easy web application development, ideal for small to medium-sized projects.
3. **python3-pymysql**: A pure-Python MySQL client library that allows Python applications to connect to and interact with MySQL databases for querying and managing data.
4. **python3-boto3**: The AWS SDK for Python, enabling developers to programmatically interact with AWS services like S3 and EC2, facilitating cloud resource management.

🡺Now,to connect python with terminal the command is **Python3 EmpApp.py.**

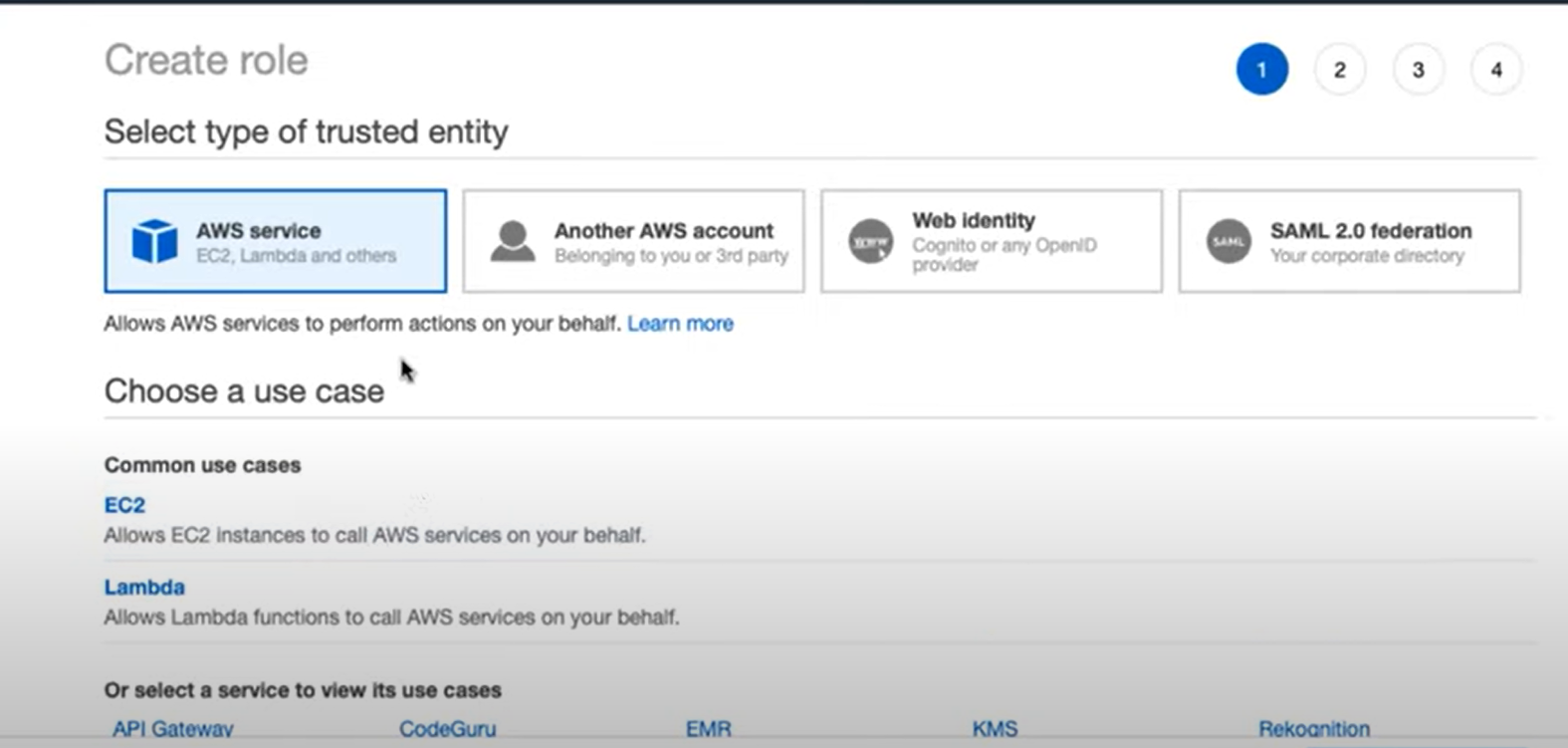


🡺Now,it is successfully connected now,we can run on our local host ,so the result is:

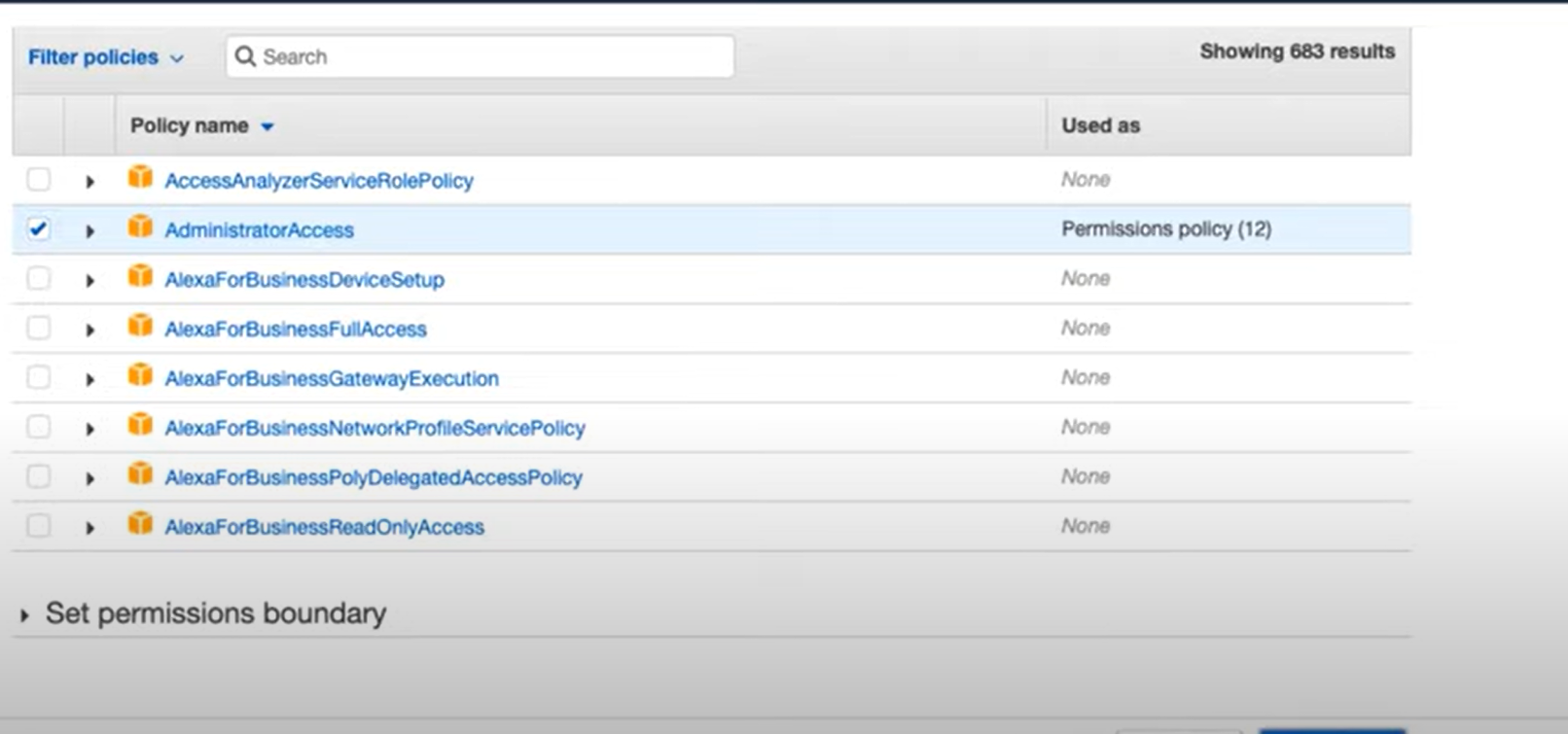


🡺It is connected to local host now,we have to connect to ec2 instance i.e,to our webserver

🡺But,it will directly not allowed to do so ,so we have to create a IAM user



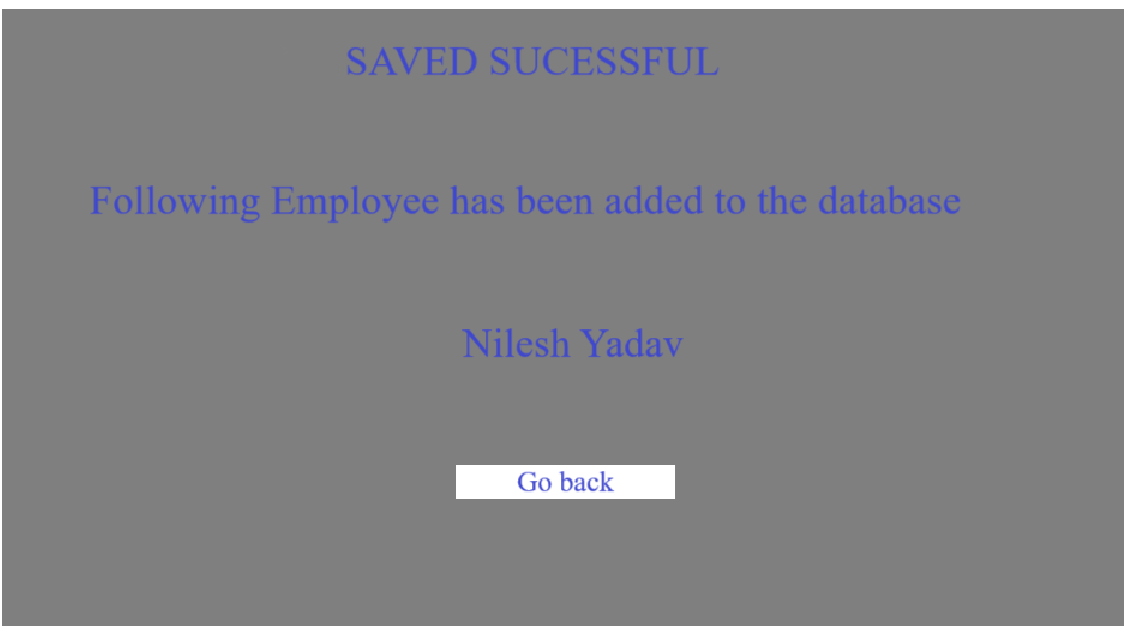
🡺in this we have to choose for which role we have use this in our case it is ec2 and we have choose option some policy



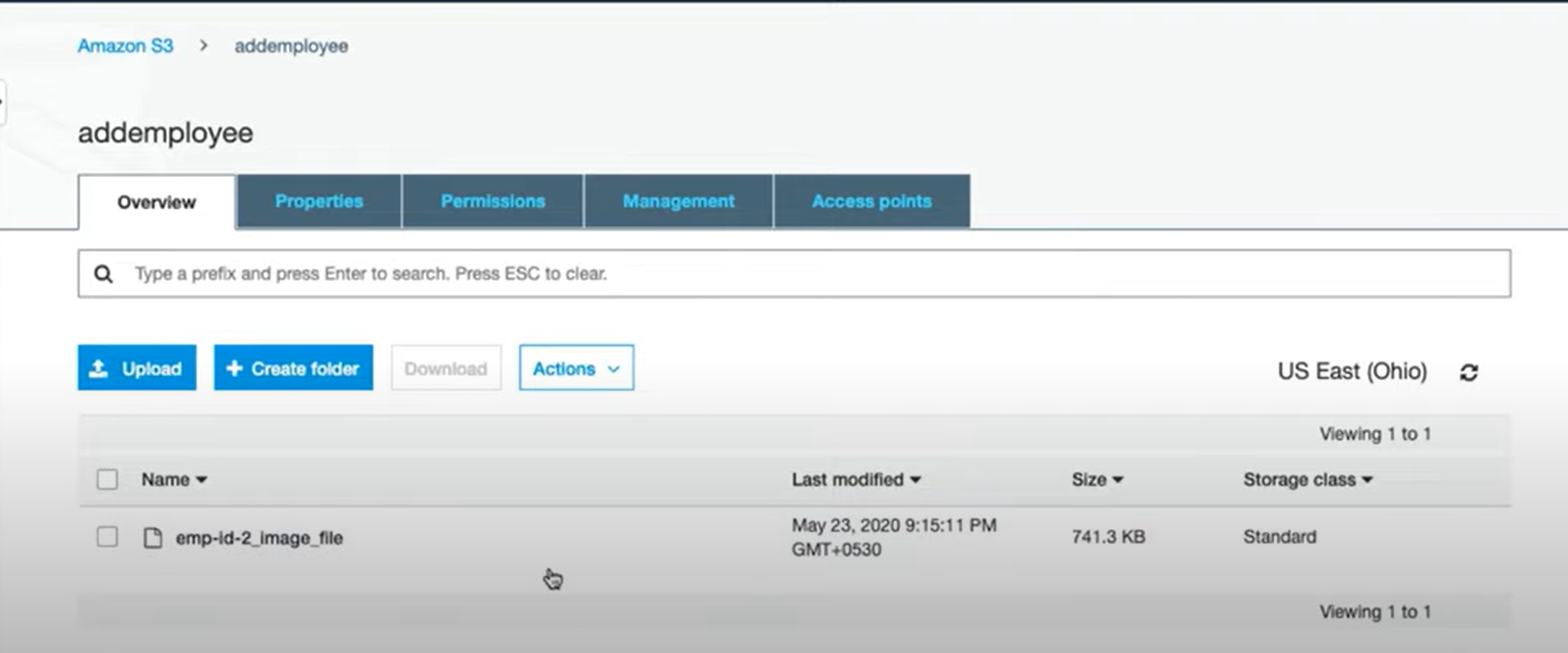
🡺Giving a role name AWSlive



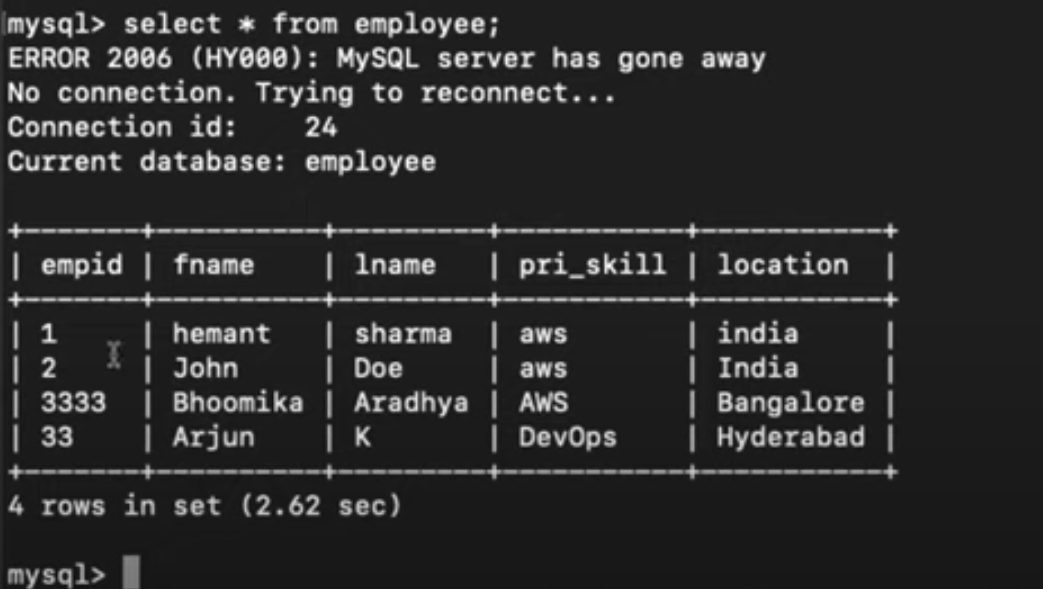
🡺After this we connected to our server and after entrying the data of employee we store that data on mysql server on backend.



🡺The file I stored in database is also stored in S3 bucket



🡺data stored in mysql



🡺now ,WE had connected our server to ec2 instance through ip address but I want that ec2 instance must connect with my domain

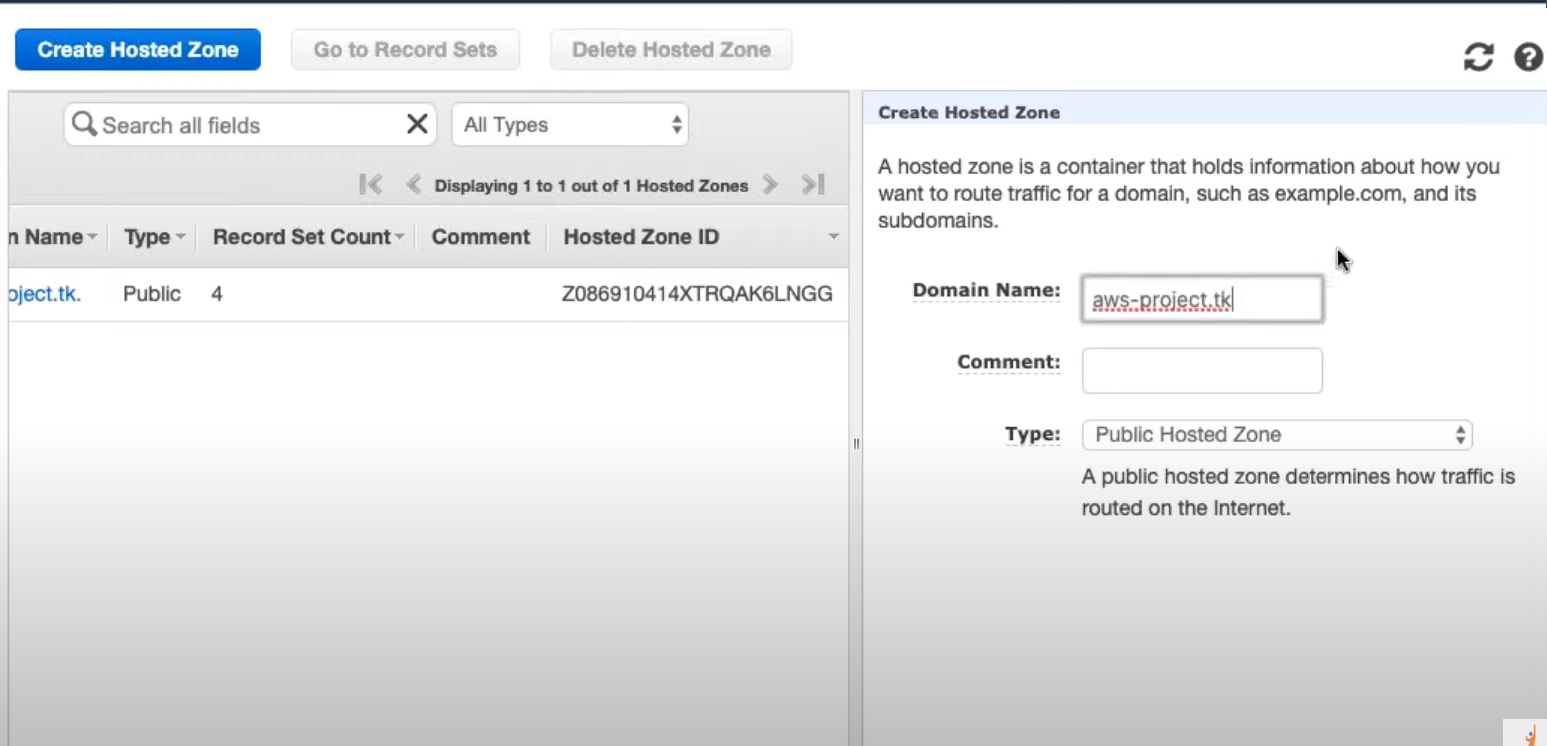
**Step 5**.Connection of domain to ec2 instance through **Route53**

**Short info about Route53:**

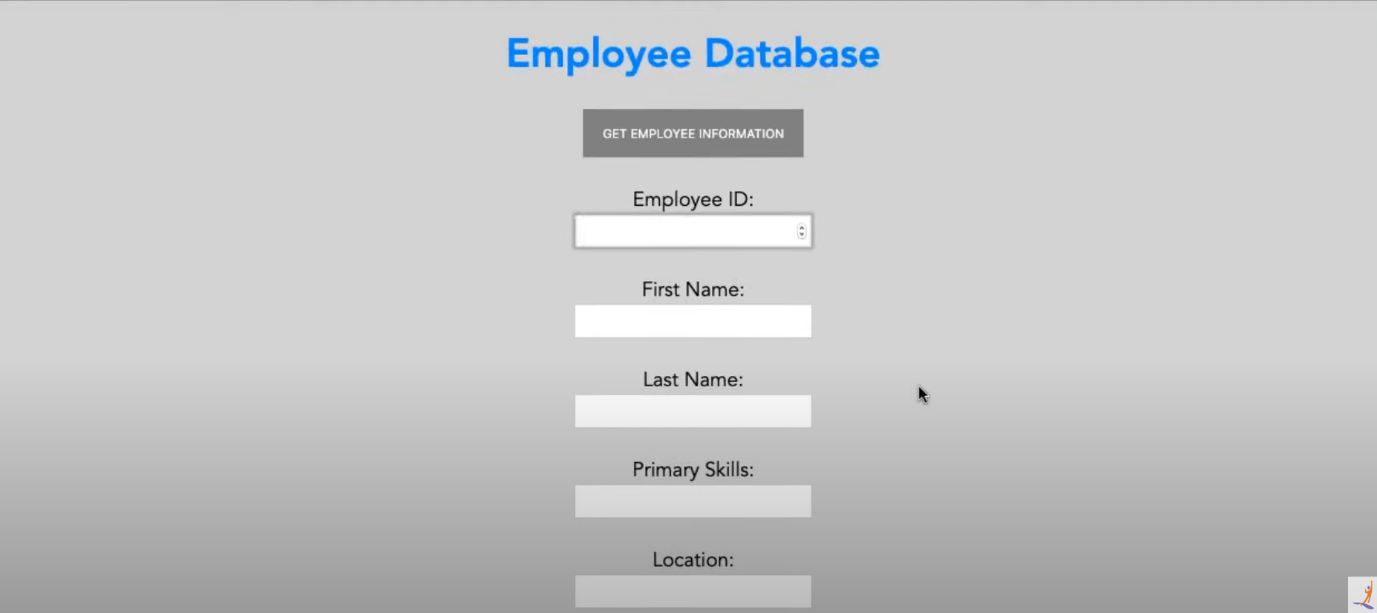
Amazon Route 53 is a scalable DNS web service that translates domain names into IP addresses and offers domain registration. It enables users to create and manage various DNS records while monitoring resource health to reroute traffic when necessary. The service provides multiple routing policies, including simple, weighted, latency-based, geo-location, and failover routing. It integrates seamlessly with other AWS services like CloudFront, S3, and EC2, and features a visual editor for managing complex routing configurations. Route 53 ensures high availability and optimal performance for websites and applications.

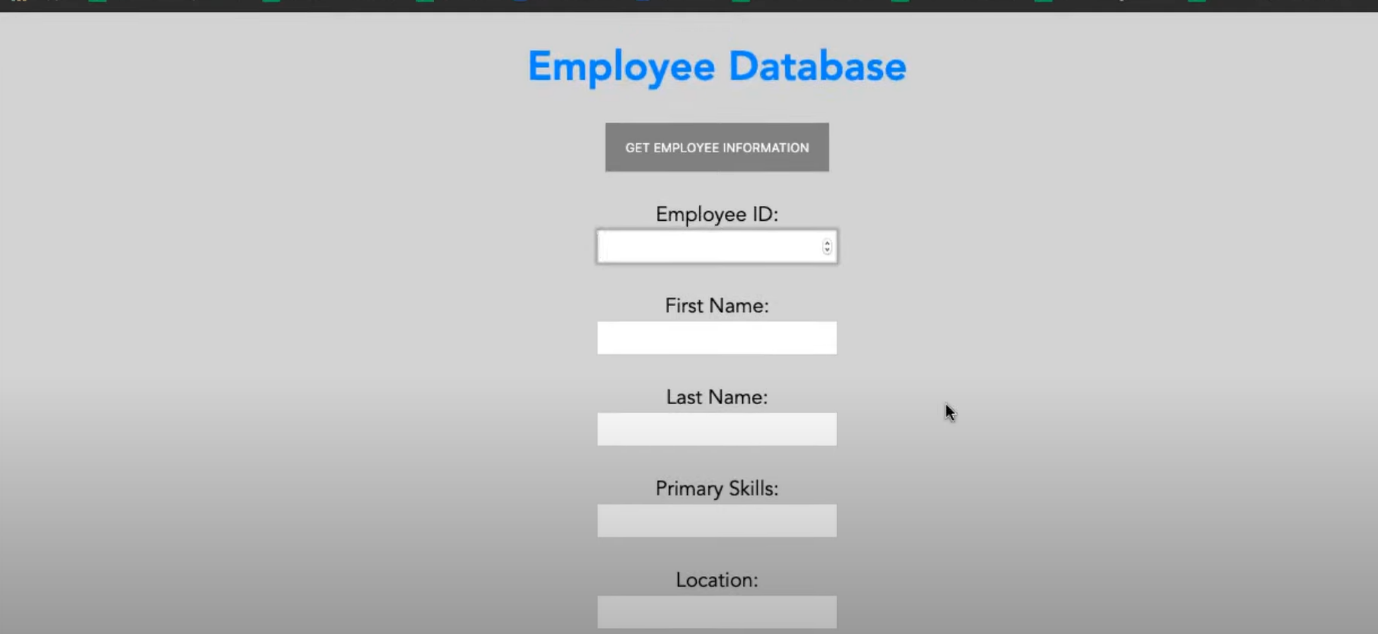
🡺But Route53 take charges for getting a domain instead of paid domain we can use free domain one of free domain provider is **Freenom.**

**🡺** So I took one domain with the name aws-project



🡺Now,using Route53 service we connect our domain to EC2 instance and this is the output:





**In conclusion**, the deployment of an end-to-end website on AWS effectively leverages various cloud services and tools. By using EC2 instances for scalable computing, S3 buckets for reliable storage, RDS for managed database services, and Route 53 for DNS management, you have established a robust infrastructure. Freenom provides domain hosting, while Python libraries like Flask, Boto3, and PyMySQL facilitate seamless application development and interaction with AWS services. Additionally, using the MySQL client and PuTTY software for SSH access to the EC2 instance enhances your ability to manage and run the website efficiently. This integrated approach ensures a reliable, scalable, and well-managed web application environment.