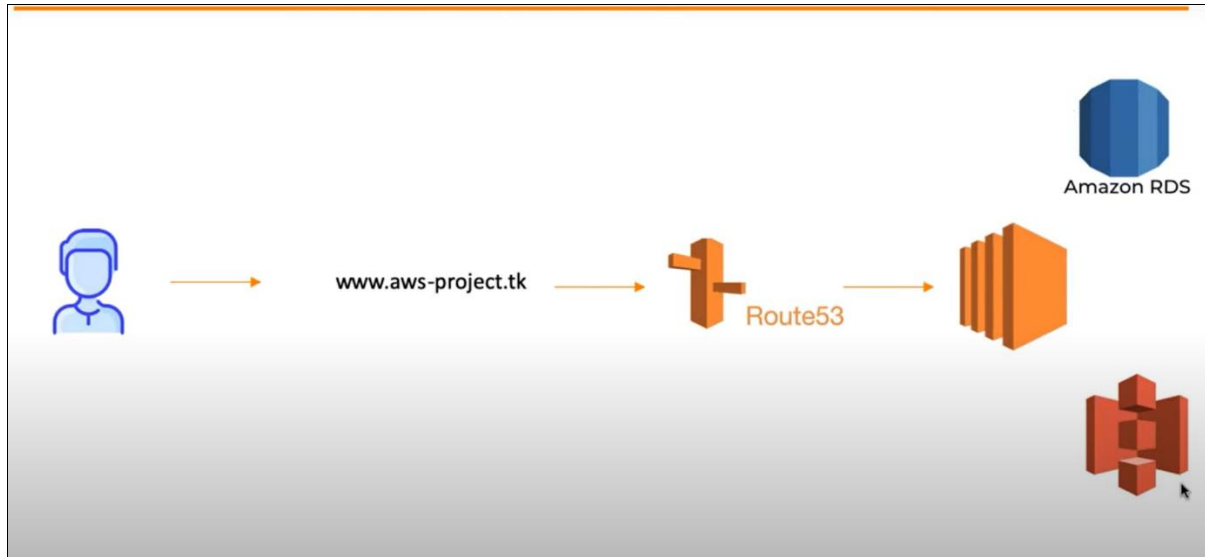


Aim: Deploying an end-to-end website on AWS.

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

Overview of the Project:

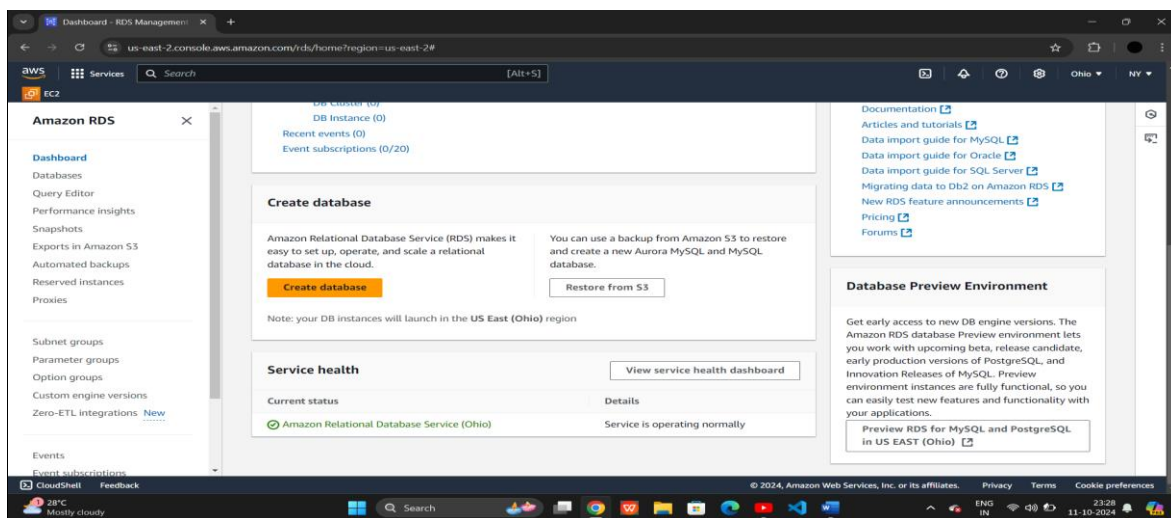


Step1: Deploy MySQL database on AWS RDS.

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.

[RDS](#) > Create database

Create database [Info](#)

Choose a database creation method


☒ **Standard create**
You set all of the configuration options, including ones for availability, security, backups, and maintenance.


☐ **Easy create**
Use recommended best-practice configurations. Some configuration options can be changed after the database is created.


→ In Engine options click on Mysql server.


Engine options


Engine type [Info](#)


☐ Aurora (MySQL Compatible)



☐ Aurora (PostgreSQL Compatible)



☒ **MySQL**


☐ MariaDB


☐ PostgreSQL


☐ Oracle


☐ Microsoft SQL Server


☐ IBM Db2


→ Now, Our Edition is MYSQL Community and Engine version is MYSQL 8.0.39.

Edition

☒ MySQL Community

Engine version [Info](#)

View the engine versions that support the following database features.

▼ Hide filters

☒ Show only versions that support the Multi-AZ DB cluster [Info](#)
 Create a Multi-AZ DB cluster with one primary DB instance and two readable standby DB instances. Multi-AZ DB clusters provide up to 2x faster transaction commit latency and automatic failover in typically under 35 seconds.

☐ Show only versions that support the Amazon RDS Optimized Writes [Info](#)
 Amazon RDS Optimized Writes improves write throughput by up to 2x at no additional cost.

Engine version

MySQL 8.0.39 ▼

☐ Enable RDS Extended Support [Info](#)
 Amazon RDS Extended Support is a [paid offering](#). By selecting this option, you consent to being charged for this offering if you are running your database major version past the RDS end of standard support date for that version. Check the end of standard support date for your major version in the [RDS for MySQL documentation](#).

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

Templates

Choose a sample template to meet your use case.

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

☐ Dev/Test
 This instance is intended for development use outside of a production environment.

☒ Free tier
 Use RDS Free Tier to develop new applications, test existing applications, or gain hands-on experience with Amazon RDS.
[Info](#)

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

Settings

DB instance identifier [Info](#)

Type a name for your DB instance. The name must be unique across all DB instances owned by your AWS account in the current AWS Region.

employee1

The DB instance identifier is case-insensitive, but is stored as all lowercase (as in "mydbinstance"). Constraints: 1 to 60 alphanumeric characters or hyphens. First character must be a letter. Can't contain two consecutive hyphens. Can't end with a hyphen.

▼ Credentials Settings

Master username [Info](#)

Type a login ID for the master user of your DB instance.

Nilesh

1 to 16 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](#)

Password strength **Strong**

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

Confirm master password [Info](#)

➔Set Subnet grp as default and make it public accessibly

i After a database is created, you can't change its VPC.

DB subnet group [Info](#)

Choose the DB subnet group. The DB subnet group defines which subnets and IP ranges the DB instance can use in the VPC that you selected.

default-vpc-0e59ad4560d65096f

3 Subnets, 3 Availability Zones

Public access [Info](#)

☒ Yes

RDS assigns a public IP address 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.

☐ No

RDS doesn't assign a public IP address to the database. Only Amazon EC2 instances and other resources inside the VPC can connect to your database. Choose one or more VPC security groups that specify which resources can connect to the database.

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

➔Now,we successfully created database with employee1.

✔ Successfully created database employee1 [View connection details](#) ✕

You can use settings from employee1 to simplify configuration of suggested database add-ons while we finish creating your DB for you.

Notifications 0 0 2 0 0 0

[RDS](#) > Databases

i Consider creating a Blue/Green Deployment to minimize downtime during upgrades ✕

You may want to consider using Amazon RDS Blue/Green Deployments and minimize your downtime during upgrades. A Blue/Green Deployment provides a staging environment for changes to production databases. [RDS User Guide](#) [Aurora User Guide](#)

Databases (1)

☒ Group resources ⌂ Modify Actions Restore from S3 Create database

< 1 > ⚙

<input type="checkbox"/>	DB identifier	Status	Role	Engine	Region ...	Size	Recommendations
<input type="radio"/>	employee1	⌚ Backing-up	Instance	MySQL Co...	us-east-2b	db.t4g.mi...	

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 99.99999999% (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.

General configuration

AWS Region
US East (Ohio) us-east-2

Bucket name [Info](#)

nileshbucket58

Bucket name must be unique within the global namespace and follow the bucket naming rules. [See rules for bucket naming](#)

Copy settings from existing bucket - *optional*
Only the bucket settings in the following configuration are copied.

Choose bucket

Format: s3://bucket/prefix

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

Object Ownership [Info](#)

Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership determines who can specify access to objects.

☒ **ACLs disabled (recommended)**

All objects in this bucket are owned by this account. Access to this bucket and its objects is specified using only policies.

☐ **ACLs enabled**

Objects in this bucket can be owned by other AWS accounts. Access to this bucket and its objects can be specified using ACLs.

Object Ownership
Bucket owner enforced

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

Block Public Access settings for this bucket

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to this bucket and its objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to this bucket or objects within, you can customize the individual settings below to suit your specific storage use cases. [Learn more](#)

☒ Block all public access

Turning this setting on is the same as turning on all four settings below. Each of the following settings are independent of one another.

☒ Block public access to buckets and objects granted through **new** access control lists (ACLs)

S3 will block public access permissions applied to newly added buckets or objects, and prevent the creation of new public access ACLs for existing buckets and objects. This setting doesn't change any existing permissions that allow public access to S3 resources using ACLs.

☒ Block public access to buckets and objects granted through **any** access control lists (ACLs)

S3 will ignore all ACLs that grant public access to buckets and objects.

☒ Block public access to buckets and objects granted through **new** public bucket or access point policies

S3 will block new bucket and access point policies that grant public access to buckets and objects. This setting doesn't change any existing policies that allow public access to S3 resources.

☒ Block public and cross-account access to buckets and objects through **any** public bucket or access point policies

S3 will ignore public and cross-account access for buckets or access points with policies that grant public access to buckets and objects.

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

Default encryption [Info](#)

Server-side encryption is automatically applied to new objects stored in this bucket.

Encryption type [Info](#)

☒ Server-side encryption with Amazon S3 managed keys (SSE-S3)

☐ Server-side encryption with AWS Key Management Service keys (SSE-KMS)

☐ Dual-layer server-side encryption with AWS Key Management Service keys (DSSE-KMS)

Secure your objects with two separate layers of encryption. For details on pricing, see [DSSE-KMS pricing](#) on the **Storage** tab of the [Amazon S3 pricing page](#).

Bucket Key

Using an S3 Bucket Key for SSE-KMS reduces encryption costs by lowering calls to AWS KMS. S3 Bucket Keys aren't supported for DSSE-KMS. [Learn more](#)

☐ Disable

☒ Enable

Advanced settings

[i](#) After creating the bucket, you can upload files and folders to the bucket, and configure additional bucket settings.

Cancel

Create bucket

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

✔ Successfully created bucket "nileshbucket58" [View details](#) ✕

To upload files and folders, or to configure additional bucket settings, choose [View details](#).

Amazon S3 > Buckets

▶ Account snapshot - updated every 24 hours All AWS Regions

Storage lens provides visibility into storage usage and activity trends. [Learn more](#)

[View Storage Lens dashboard](#)

[General purpose buckets](#) | [Directory buckets](#)

General purpose buckets (2) [Info](#) All AWS Regions

Buckets are containers for data stored in S3.

Name	AWS Region	IAM Access Analyzer	Creation date
aws-cloudtrail-logs-767397846433-848f01a0	Asia Pacific (Mumbai) ap-south-1	View analyzer for ap-south-1	September 14, 2024, 17:27:18 (UTC+05:30)
nileshbucket58	US East (Ohio) us-east-2	View analyzer for us-east-2	October 11, 2024, 23:56:11 (UTC+05:30)

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

Launch an instance Info

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

Name and tags Info

Name
[Add additional tags](#)

▼ Application and OS Images (Amazon Machine Image) Info

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

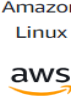
[Quick Start](#)


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


▼ Application and OS Images (Amazon Machine Image) Info


An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below


[Quick Start](#)

















[Browse more AMIs](#)

Including AMIs from AWS, Marketplace and the Community

Amazon Machine Image (AMI)

Ubuntu Server 24.04 LTS (HVM), SSD Volume Type

ami-0ea3c35c3284d82 (64-bit (x86)) / ami-01ebf7c0e446f85f9 (64-bit (Arm))

Virtualization: hvm ENA enabled: true Root device type: ebs

Free tier eligible

Description

Ubuntu Server 24.04 LTS (HVM).EBS General Purpose (SSD) Volume Type. Support available from Canonical

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

Key pair (login)
[Info](#)

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - *required*

↕

↻
[Create new key pair](#)

Network settings
[Info](#)

[Edit](#)

Network | [Info](#)

vpc-0e59ad4560d65096f

Subnet | [Info](#)

No preference (Default subnet in any availability zone)

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.

→ we successfully created our instance.

[EC2](#) > ... > Launch an instance

✓ Success

Successfully initiated launch of instance (i-0f1d0b8da01865579)

[Launch log](#)

Next Steps

< 1 2 3 4 5 6 >

Create billing and free tier usage alerts

To manage costs and avoid surprise bills, set up email notifications for billing and free tier usage thresholds.

[Create billing alerts](#)

Connect to your instance

Once your instance is running, log into it from your local computer.

[Connect to instance](#)

[Learn more](#)

Connect an RDS database

Configure the connection between an EC2 instance and a database to allow traffic flow between them.

[Connect an RDS database](#)

[Create a new RDS database](#)
[Learn more](#)

Create EBS snapshot policy

Create a policy that automates the creation, retention, and deletion of EBS snapshots

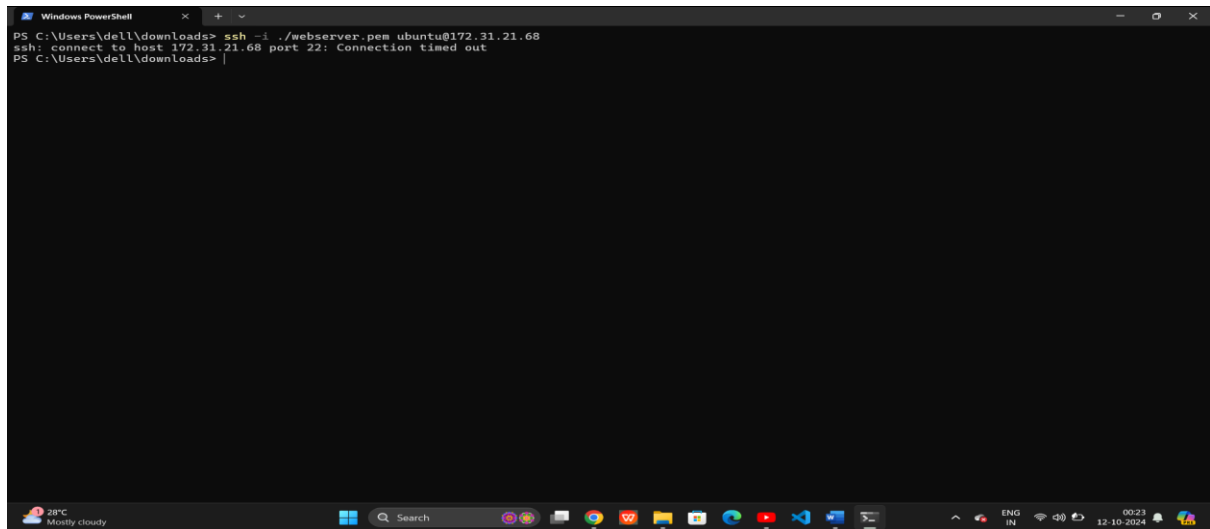
[Create EBS snapshot policy](#)

→ properties of our instance.

Instance summary for i-0f1d0b8da01865579 (Nilesh project) Info <div> ↻ Connect Instance state ▼ Actions ▼ </div>		
Updated less than a minute ago		
Instance ID i-0f1d0b8da01865579 (Nilesh project)	Public IPv4 address 3.139.94.148 open address	Private IPv4 addresses 172.31.21.68
IPv6 address -	Instance state Running	Public IPv4 DNS ec2-3-139-94-148.us-east-2.compute.amazonaws.com open address
Hostname type IP name: ip-172-31-21-68.us-east-2.compute.internal	Private IP DNS name (IPv4 only) ip-172-31-21-68.us-east-2.compute.internal	
Answer private resource DNS name IPv4 (A)	Instance type t2.micro	Elastic IP addresses -
Auto-assigned IP address 3.139.94.148 [Public IP]	VPC ID vpc-0e59ad4560d65096f	AWS Compute Optimizer finding Opt-in to AWS Compute Optimizer for recommendations. Learn more
IAM Role -	Subnet ID subnet-05269002668a46e47	Auto Scaling Group name -
IMDSv2 Required	Instance ARN 	

Step4: Connection between instance and local host.

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

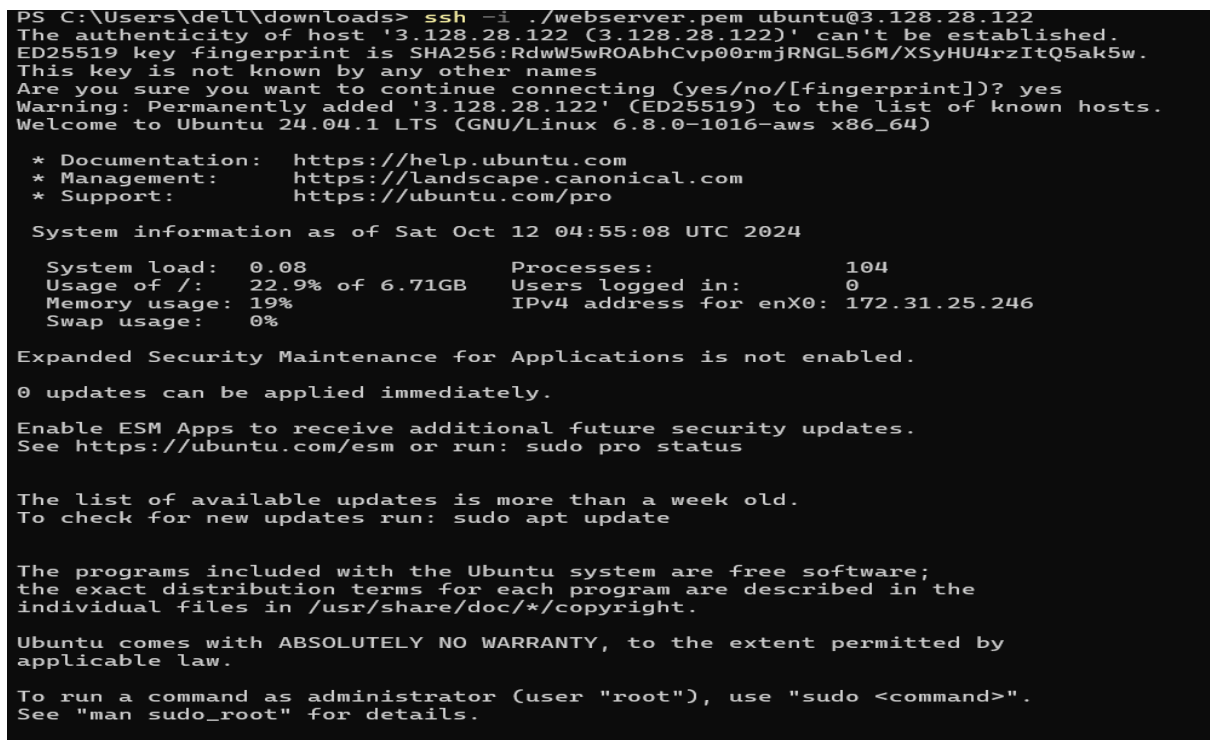


```

PS C:\Users\de\l\downloads> ssh -i .\webserver.pem ubuntu@172.31.21.68
ssh: connect to host 172.31.21.68 port 22: Connection timed out
PS C:\Users\de\l\downloads>
  
```

→ 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**.



```

PS C:\Users\de\l\downloads> ssh -i .\webserver.pem ubuntu@3.128.28.122
The authenticity of host '3.128.28.122 (3.128.28.122)' can't be established.
ED25519 key fingerprint is SHA256:RdwW5wROAbhCvp00rmjRNGl56M/XSyHU4rzItQ5ak5w.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '3.128.28.122' (ED25519) to the list of known hosts.
Welcome to Ubuntu 24.04.1 LTS (GNU/Linux 6.8.0-1016-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/pro

System information as of Sat Oct 12 04:55:08 UTC 2024

System load:  0.08      Processes:            104
Usage of /:   22.9% of 6.71GB   Users logged in:     0
Memory usage: 19%      IPv4 address for enx0: 172.31.25.246
Swap usage:   0%

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

The list of available updates is more than a week old.
To check for new updates run: sudo apt update

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.
  
```

→ 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](#).
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.

```

Using username "ubuntu".
Authenticating with public key "imported-openssh-key"
Welcome to Ubuntu 24.04.1 LTS (GNU/Linux 6.8.0-1016-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/pro

System information as of Sat Oct 12 06:23:56 UTC 2024

System load:  0.0          Processes:    109
Usage of /:   27.0% of 6.71GB   Users logged in: 1
Memory usage: 22%          IPv4 address for enx0: 172.31.25.246
Swap usage:   0%

Expanded Security Maintenance for Applications is not enabled.

3 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

Last login: Sat Oct 12 04:55:11 2024 from 183.87.175.50
ubuntu@ip-172-31-25-246:~$ mysql -h employee.cxsgw06gelty.us-east-2-rds.amazonaws.com -u Nilesh -p
Enter password:

```

➔ 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.

```

ubuntu@ip-172-31-25-246:~$ sudo apt-get install mysql-client
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  mysql-client-8.0 mysql-client-core-8.0 mysql-common
The following NEW packages will be installed:
  mysql-client mysql-client-8.0 mysql-client-core-8.0 mysql-common
0 upgraded, 4 newly installed, 0 to remove and 12 not upgraded.
Need to get 2832 kB of archives.
After this operation, 61.8 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://us-east-2.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 mysql-client-core-8.0 amd64 8.0.39-0ubuntu0.24.04.2 [2794 kB]
Get:2 http://us-east-2.ec2.archive.ubuntu.com/ubuntu noble/main amd64 mysql-common all 5.8+1.1.0build1 [6746 B]
Get:3 http://us-east-2.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 mysql-client-8.0 amd64 8.0.39-0ubuntu0.24.04.2 [22.5 kB]
Get:4 http://us-east-2.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 mysql-client all 8.0.39-0ubuntu0.24.04.2 [9412 B]
Fetched 2832 kB in 0s (52.9 MB/s)
Selecting previously unselected package mysql-client-core-8.0.
(Reading database ... 67836 files and directories currently installed.)
Preparing to unpack .../mysql-client-core-8.0_8.0.39-0ubuntu0.24.04.2_amd64.deb ...
Unpacking mysql-client-core-8.0 (8.0.39-0ubuntu0.24.04.2) ...
Selecting previously unselected package mysql-common.
Preparing to unpack .../mysql-common_5.8+1.1.0build1_all.deb ...
Unpacking mysql-common (5.8+1.1.0build1) ...
Selecting previously unselected package mysql-client-8.0.
Preparing to unpack .../mysql-client-8.0_8.0.39-0ubuntu0.24.04.2_amd64.deb ...
Unpacking mysql-client-8.0 (8.0.39-0ubuntu0.24.04.2) ...
Selecting previously unselected package mysql-client.
Preparing to unpack .../mysql-client_8.0.39-0ubuntu0.24.04.2_all.deb ...
Unpacking mysql-client (8.0.39-0ubuntu0.24.04.2) ...
Setting up mysql-common (5.8+1.1.0build1) ...
update-alternatives: using /etc/mysql/my.cnf.fallback to provide /etc/mysql/my.cnf (my.cnf) in auto mode
Setting up mysql-client-core-8.0 (8.0.39-0ubuntu0.24.04.2) ...
Setting up mysql-client-8.0 (8.0.39-0ubuntu0.24.04.2) ...
Setting up mysql-client (8.0.39-0ubuntu0.24.04.2) ...
Processing triggers for man-db (2.12.0-4build2) ...
Scanning processes...

```

→ 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**

```
ubuntu@ip-172-31-25-246:~$ mysql -h employee1.cxsgw06ge1ty.us-east-2.rds.amazonaws.com -u Nilesh -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 110
Server version: 8.0.39 Source distribution

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

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

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

mysql>
```

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

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

mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| sys |
+-----+
4 rows in set (0.01 sec)

mysql> create database employee;
Query OK, 1 row affected (0.02 sec)

mysql>
```

→ Creating a table with the name employee.

```
mysql> create database employee;
Query OK, 1 row affected (0.02 sec)

mysql> use employee;
Database changed
mysql> create table employee(
    -> emp_id varchar(20),
    -> f_name varchar(50),
    -> l_name varchar(35),
    -> pri_skills varchar(20),
    -> location varchar(45));
Query OK, 0 rows affected (0.06 sec)

mysql> show tables;
+-----+
| Tables_in_employee |
+-----+
| employee            |
+-----+
1 row in set (0.00 sec)

mysql> █
```

➔ After creating a table we exit from our mysql server and 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

        # Upload 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"
```

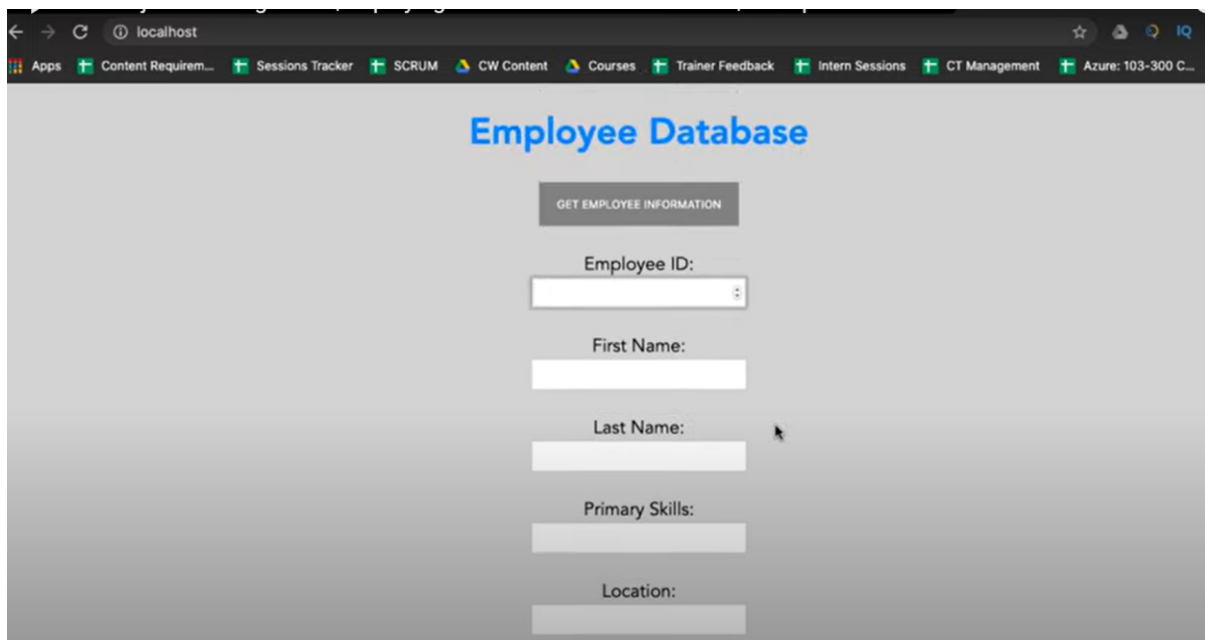
➔ Python module works

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**.

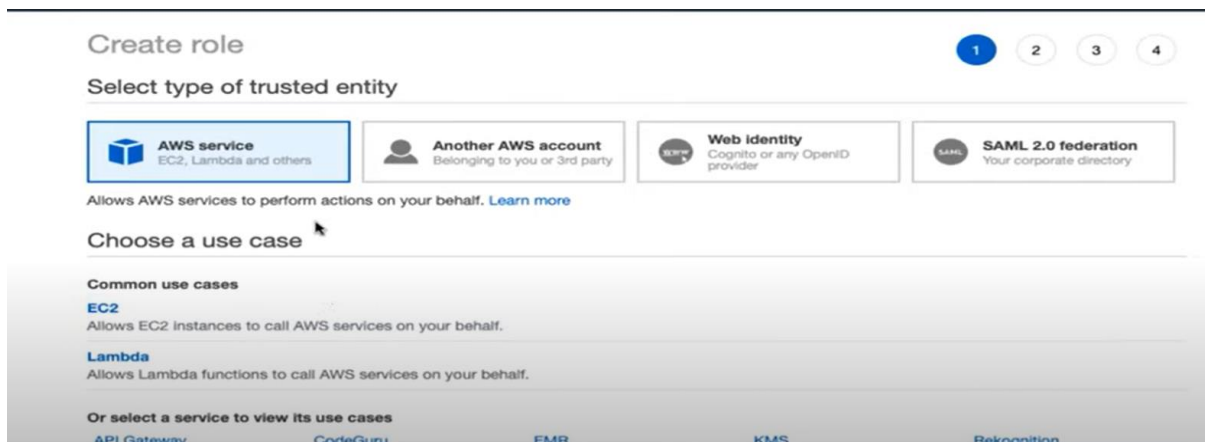
```
* Serving Flask app "EmpApp" (lazy loading)
* Environment: production
  WARNING: Do not use the development server in a production environment.
  Use a production WSGI server instead.
* Debug mode: on
* Running on http://0.0.0.0:80/ (Press CTRL+C to quit)
* Restarting with stat
* Debugger is active!
* Debugger PIN: 878-979-043
```

→ 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

Filter policies Q Search		Showing 683 results
	Policy name	Used as
<input type="checkbox"/>	AccessAnalyzerServiceRolePolicy	None
<input checked="" type="checkbox"/>	AdministratorAccess	Permissions policy (12)
<input type="checkbox"/>	AlexaForBusinessDeviceSetup	None
<input type="checkbox"/>	AlexaForBusinessFullAccess	None
<input type="checkbox"/>	AlexaForBusinessGatewayExecution	None
<input type="checkbox"/>	AlexaForBusinessNetworkProfileServicePolicy	None
<input type="checkbox"/>	AlexaForBusinessPolyDelegatedAccessPolicy	None
<input type="checkbox"/>	AlexaForBusinessReadOnlyAccess	None
Set permissions boundary		

→ Giving a role name AWSlive

Create role

1 2 3 4

Review

Provide the required information below and review this role before you create it.

Role name*

AWSlive

Use alphanumeric and "+=, @ - ." characters. Maximum 64 characters.

Role description

Allows EC2 instances to call AWS services on your behalf.

Maximum 1000 characters. Use alphanumeric and "+=, @ - ." characters.

Trusted entities

AWS service: ec2.amazonaws.com

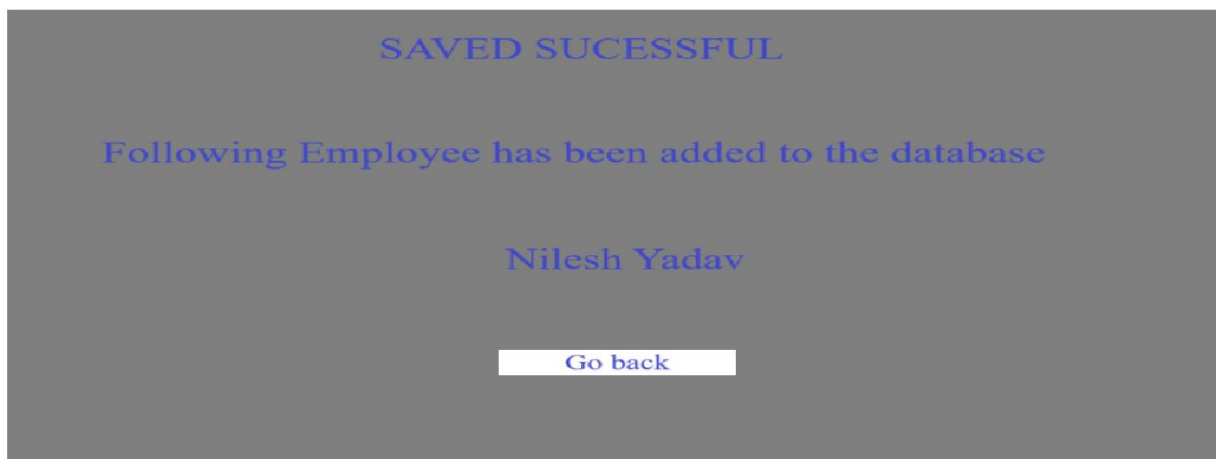
Policies

AdministratorAccess

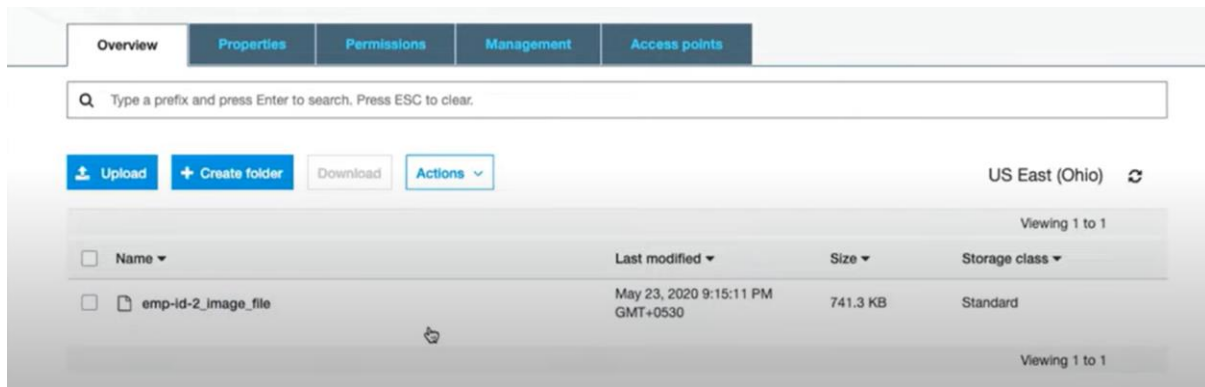
Permissions boundary

Permissions boundary is not set

→ After this we connected to our server and after entering 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

```
mysql> select * from employee;
ERROR 2006 (HY000): MySQL server has gone away
No connection. Trying to reconnect...
Connection id: 24
Current database: employee
```

empid	fname	lname	pri_skill	location
1	hemant	sharma	aws	india
2	John	Doe	aws	India
3333	Bhoomika	Aradhya	AWS	Bangalore
33	Arjun	K	DevOps	Hyderabad

```
4 rows in set (2.62 sec)

mysql>
```

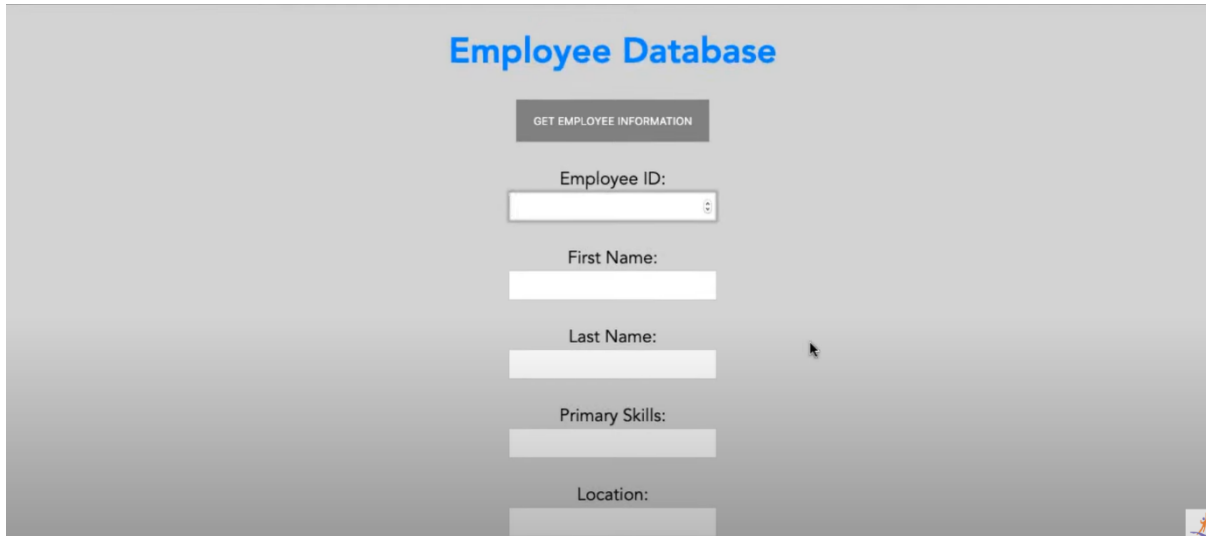
→ now , WE had connected our server to ec2 instance through ip address but I want that ec2 instance must connect with my domain

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



Employee Database

GET EMPLOYEE INFORMATION

Employee ID:

First Name:

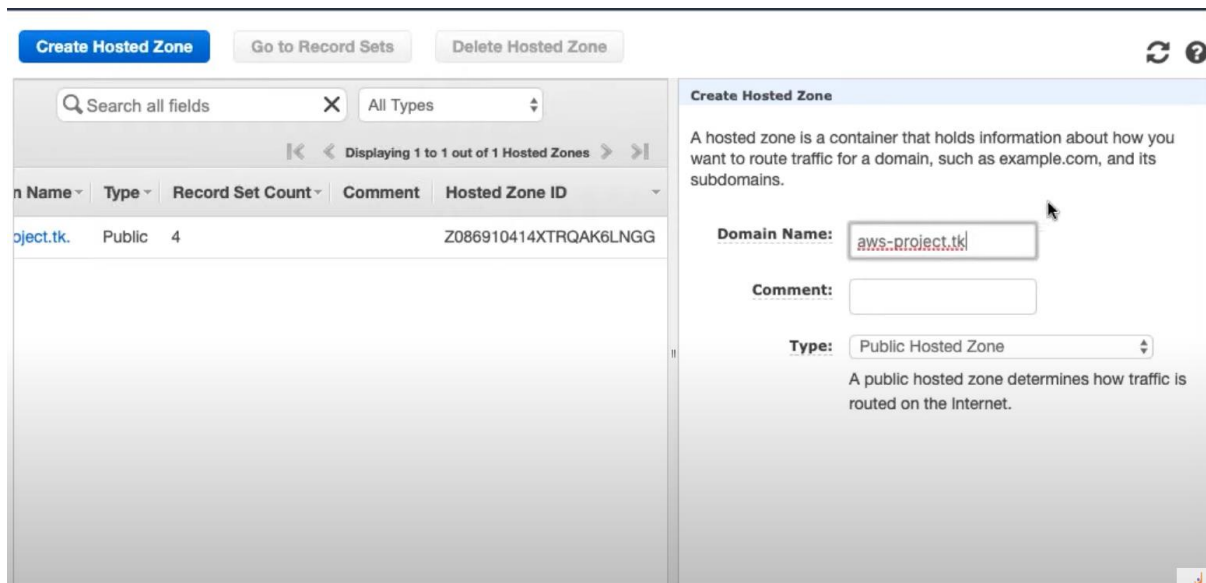
Last Name:

Primary Skills:

Location:

domain one of free domain provider is **Freenom**.

→ So I took one domain with the name aws-project



Buttons: Create Hosted Zone, Go to Record Sets, Delete Hosted Zone

Search all fields All Types

Displaying 1 to 1 out of 1 Hosted Zones

Name	Type	Record Set Count	Comment	Hosted Zone ID
aws-project.tk	Public	4		Z086910414XTRQAK6LNNGG

Create Hosted Zone

A hosted zone is a container that holds information about how you want to route traffic for a domain, such as example.com, and its subdomains.

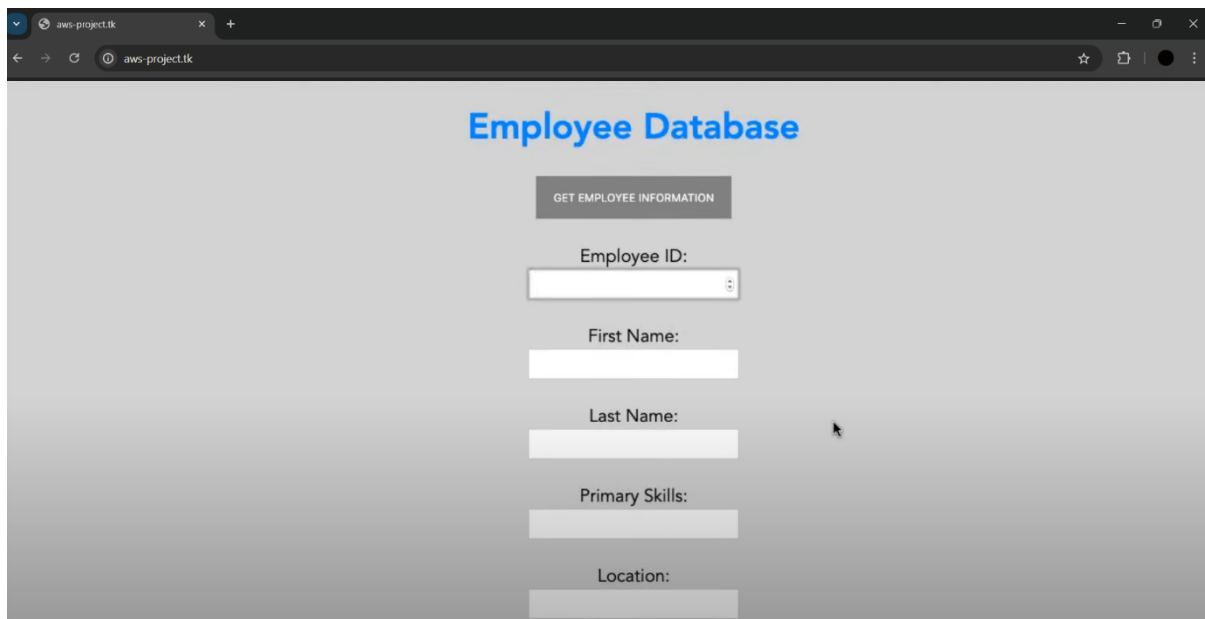
Domain Name:

Comment:

Type:

A public hosted zone determines how traffic is routed on the Internet.

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



Employee Database

GET EMPLOYEE INFORMATION

Employee ID:

First Name:

Last Name:

Primary Skills:

Location:

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.