CS 623 : CLOUD COMPUTING **FINAL PROJECT REPORT**

USING CLOUD IMPLEMENTING WEB APPLICATION OF EMPLOYEE'S DATA

BY -
AKSHITH SIMHA KATRAGADA (yd8937)
GANGAREDDY NACHU (kt6301)
MANIKANTA KODALI (nv6200)

Table of contents

CHAPTER 1: INTRODUCTION

1.1 Problem Summary

An organization or company wants to store employee details and photos very efficiently and if any new data is added then they need to update in all their databases and map them accordingly. This company doesn't want to put upfront investment on resources like hardware, servers as they don't know about the company size in future.

The company does not want to rely on primitive technology such as manual storage or local server storage which creates many problems while the company grows and it's also very slow while retrieving the employee information from the database and they don't map according to all their databases. So, the company requirements are to provide a system where large data can be stored based on size and storing employee information and helping in retrieving the data with the help of the employee ID which we can use as the primary key where this overall process is done in a fast and efficient manner.

1.2 Solution

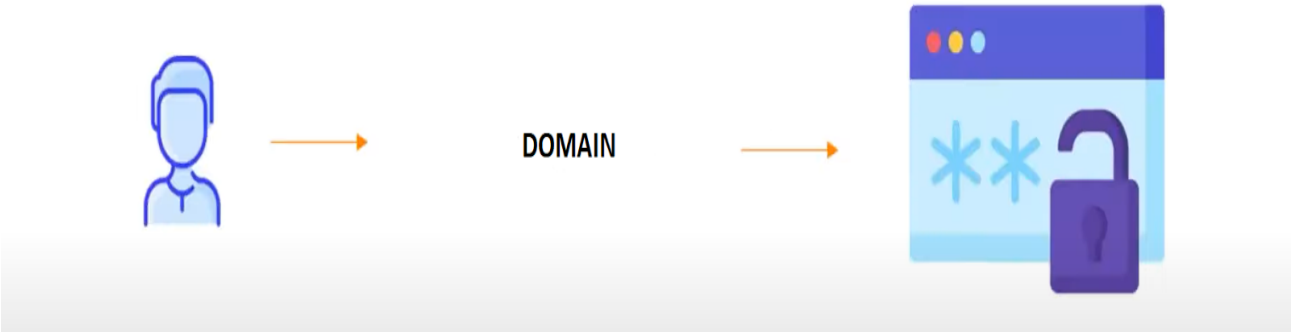
To solve the above mentioned problem, our proposed system will eliminate all the above issues and provide efficient results by using cloud computing technology, this system is implemented as a web application on AWS.

The web application we developed is deployed on AWS cloud platform where EC2 is used for launching instances S3 bucket is used for storing for example images and Relational Database Services for connecting to databases, the whole idea is to deploy web application on an EC2 instance which acts as Infrastructure as a Service (IaaS). In this web application we are providing search functionality to retrieve stored information about employees very quickly and efficiently.

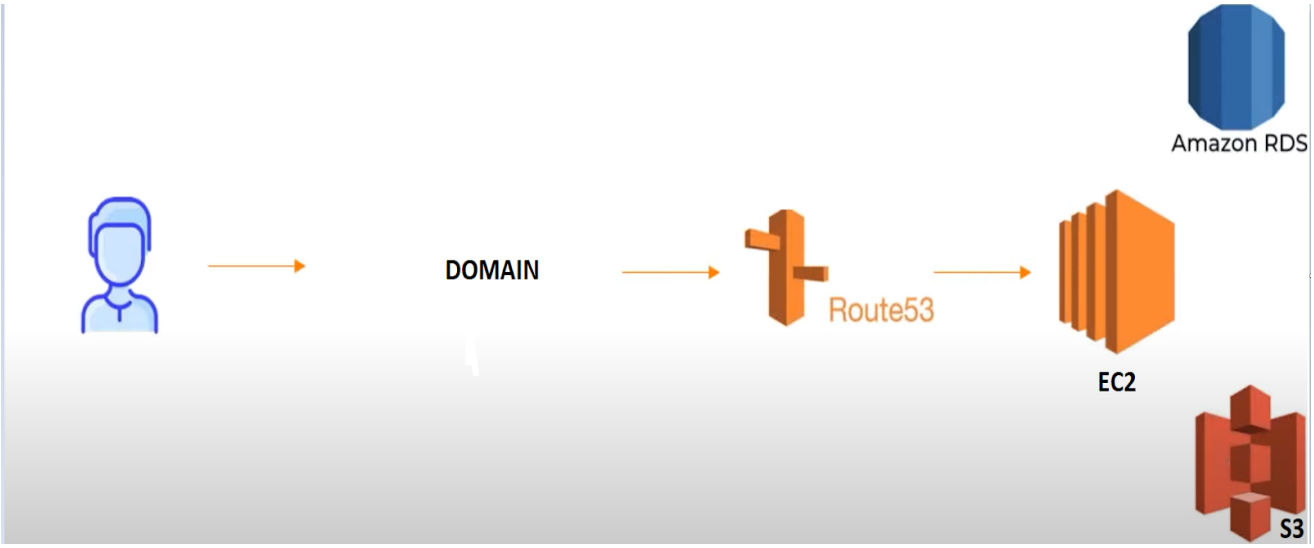
So, by this system the company or organization will be more benefited in numerous ways like companies don’t need to put upfront investment on purchasing resources as scaling up and scaling down is done whenever it’s needed automatically and pay as per usage of the resources only.

We are using many AWS services to implement this system like RDS, S3, EC2 and AWS boto and python for programming. The cloud platform offers more storage space, so large data can be stored, the computation speed is very fast so that administrators can retrieve or analyze employee data very efficiently.

CHAPTER 2: IMPLEMENTATION WORKFLOW



OVERVIEW DIAGRAM



WORKFLOW DIAGRAM

2.1 Architectural Overview

Here we are going to discuss the Architecture of the system and Relation between services.

2.1.1 Architecture of the system

The overall architecture of the system can be roughly divided into three components:

1. Selecting and configuring Amazon services
2. Building client-side application
3. Routing domain to EC2 server

In selecting and configuring Amazon services we are using three AWS namely -

1. RDS
2. S3
3. EC2

We need to select these services one by one From AWS dashboard and click appropriate options to set it up.

In building client-side applications, we are using Python and Html for the coding part.

In routing domain to EC2 server we need to use another Amazon service called Route 53, which is used for routing purposes and next we should purchase domain names and then create hosted zones and configure them.

2.1.2 Relation between services

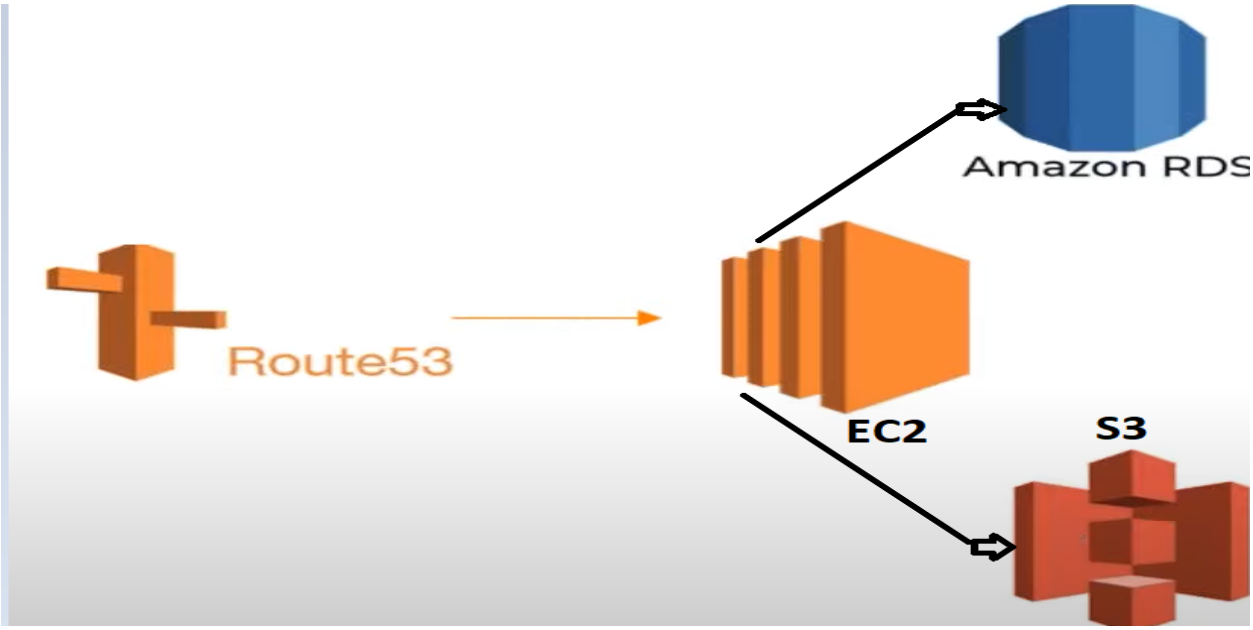
- We are using Amazon **RDS** service, which is used to operate and scale a relational database in the cloud. It provides cost-efficient, resizable capacity for an industry-standard relational database and manages common database administration tasks. In this we are using MySQL database and this is used to store the entries. This RDS service will be connected by the EC2 server, this relation is used to access the database.

- We are using Amazon **S3** service, which is a Storage for the internet. We can use it to store and retrieve any amount of data at any time, from anywhere on the web. We should give EC2 instance access to S3, basically to upload the data. To make this relation we will give an IAM role to our instance.

- We are using Amazon **EC2** service, it is a web service for launching and managing Linux/UNIX and Windows Server instances in Amazon's data centers. It provides you with complete control of your computing resources and lets you run on Amazon’s proven computing environment. Here EC2

should be connected to RDS, S3 as described above, and the website is also deployed in EC2 server.

- We are using an **IAM** service, which is used when we connect with our RDS. We should give authentication to our S3 service, so we need to provide an IAM role for our instance. Then we are using **Route 53** service which is used to route domain to EC2 server.



RELATION DIAGRAM

CHAPTER 3: DEVELOPMENT PROCESS

Here we are going to discuss the Description of the development process and challenges the team faced during this project building.

3.1 Description of the development process

1) The first phase is to select and configure the Amazon services.

We are using three Amazon service initially -

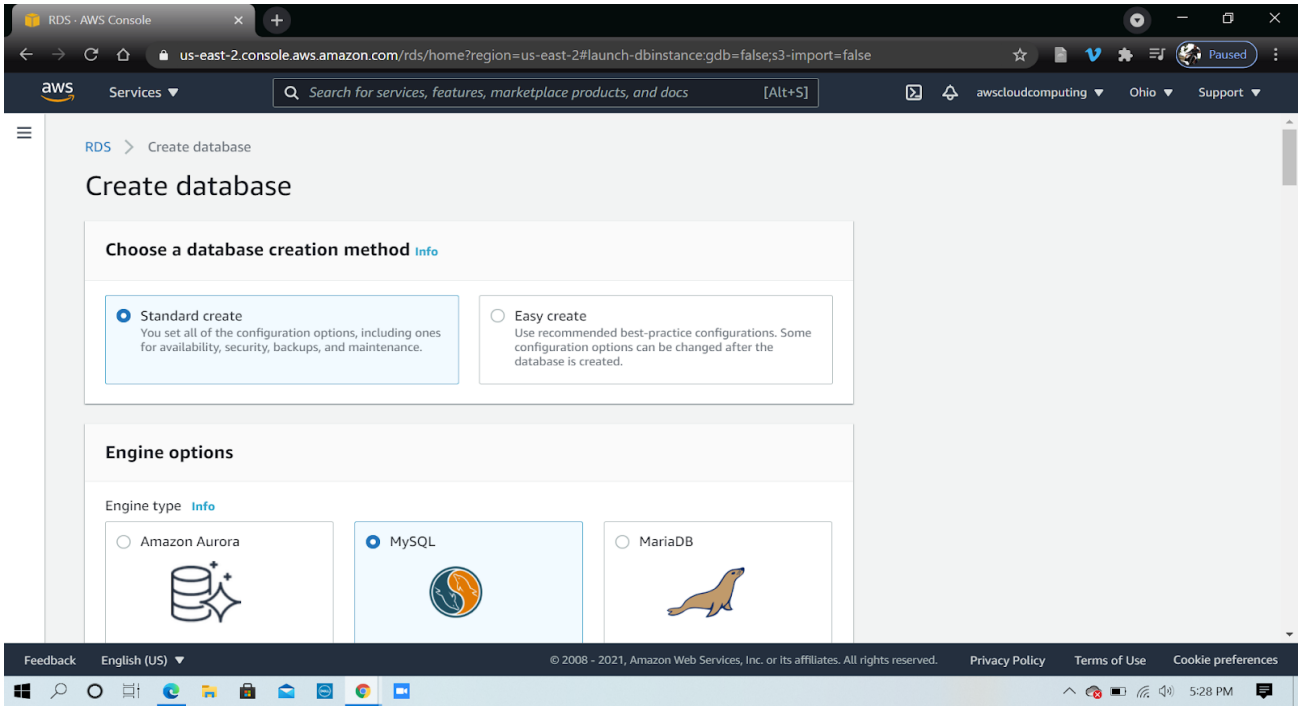
1. RDS
2. S3
3. EC2

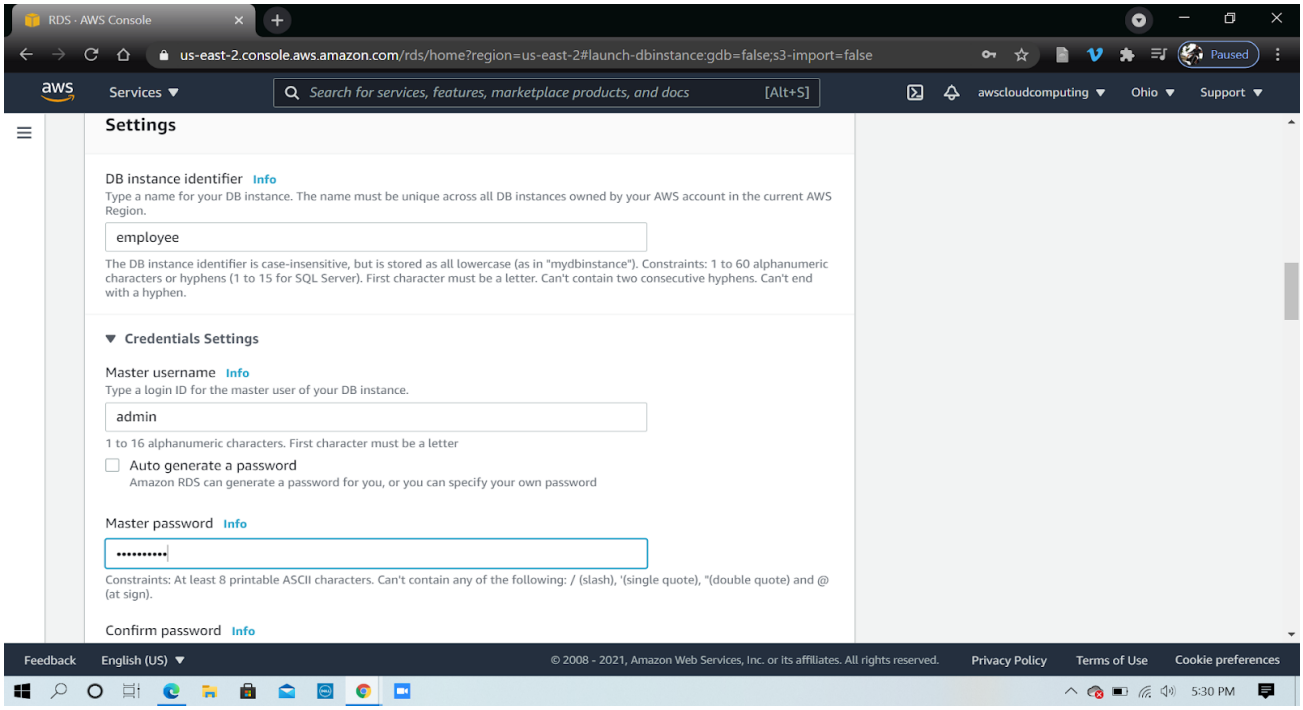
- Deploy MySQL database on AWS

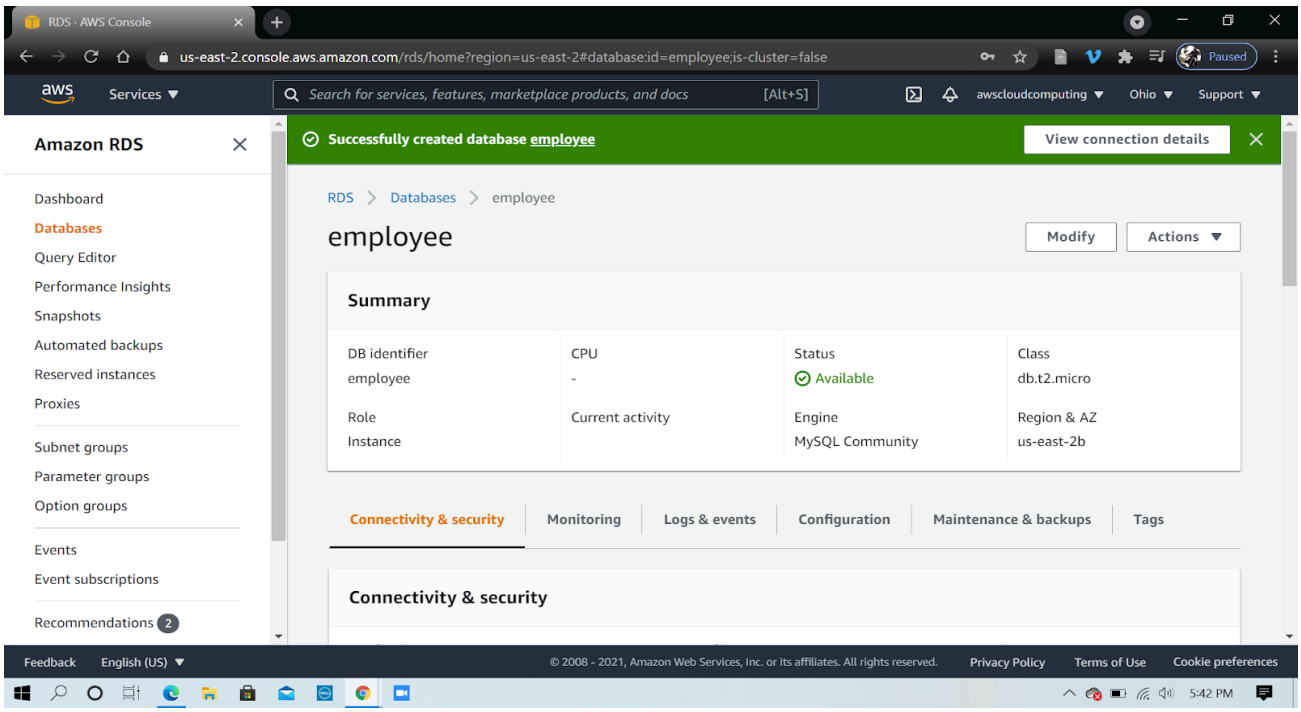
It is done by using Amazon RDS, for this we should select RDS service from AWS management console dashboard and then select MySQL database engine. There are many engines for this project. We are using MySQL engine.

There are many advantages of using RDS, we don't need to launch a server, install MySQL on it and configure it manually instead of that we are just using AWS dashboard and select right options which will be managed automatically by AWS.

Some key advantages are easy to administer, highly scalable, available & durable, fast, secure and inexpensive.



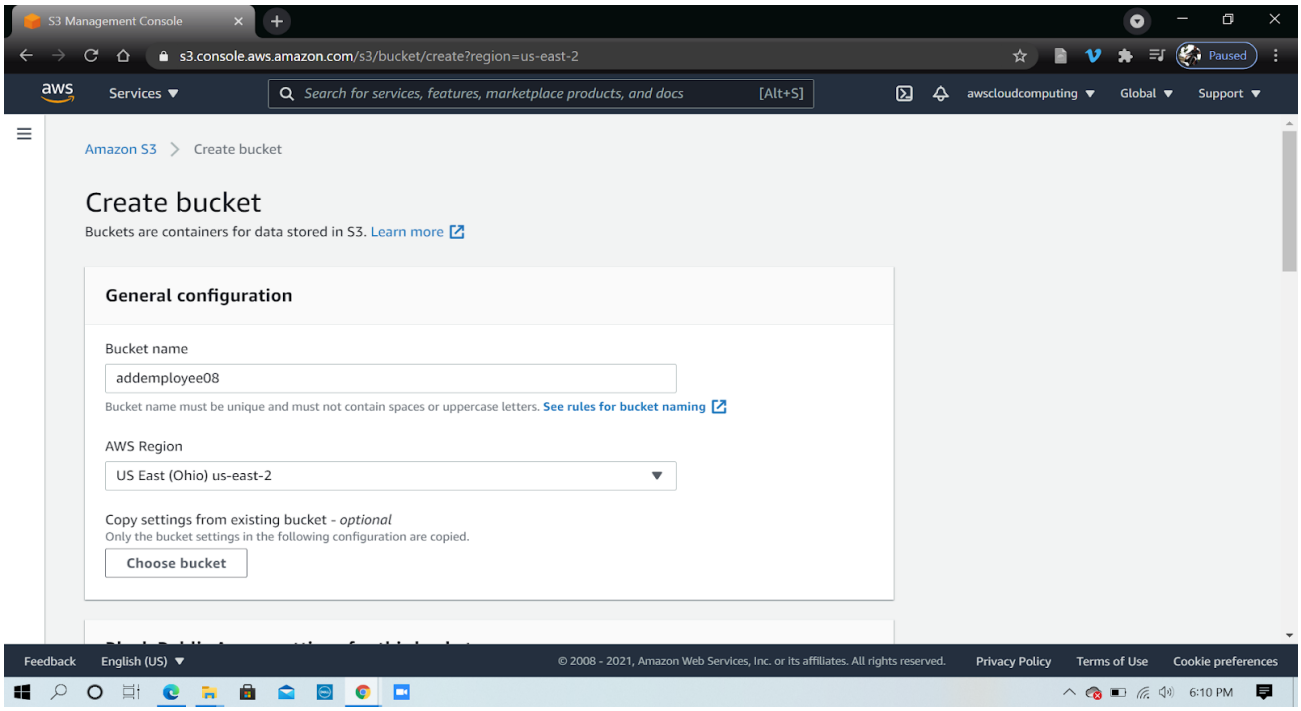


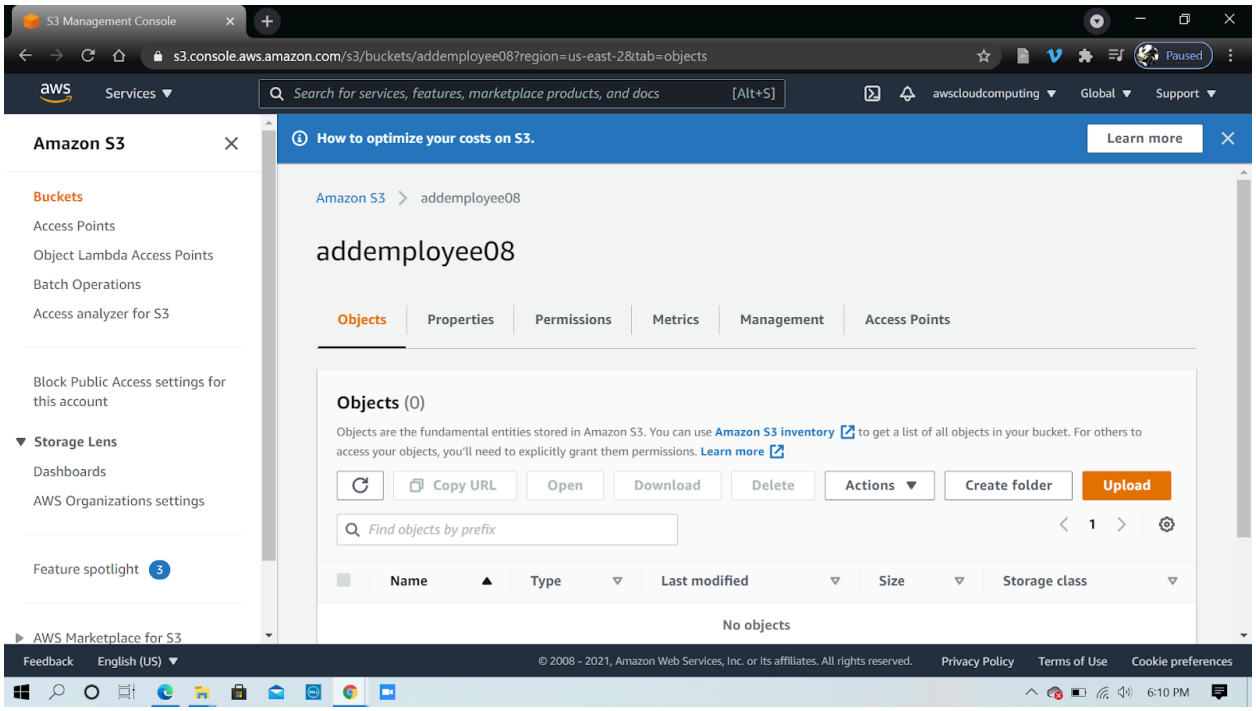


- **Create S3 bucket**

It is done by using Amazon S3, for this we should select S3 service from AWS management console dashboard and create it. Check whether your bucket has been deployed in the correct region.

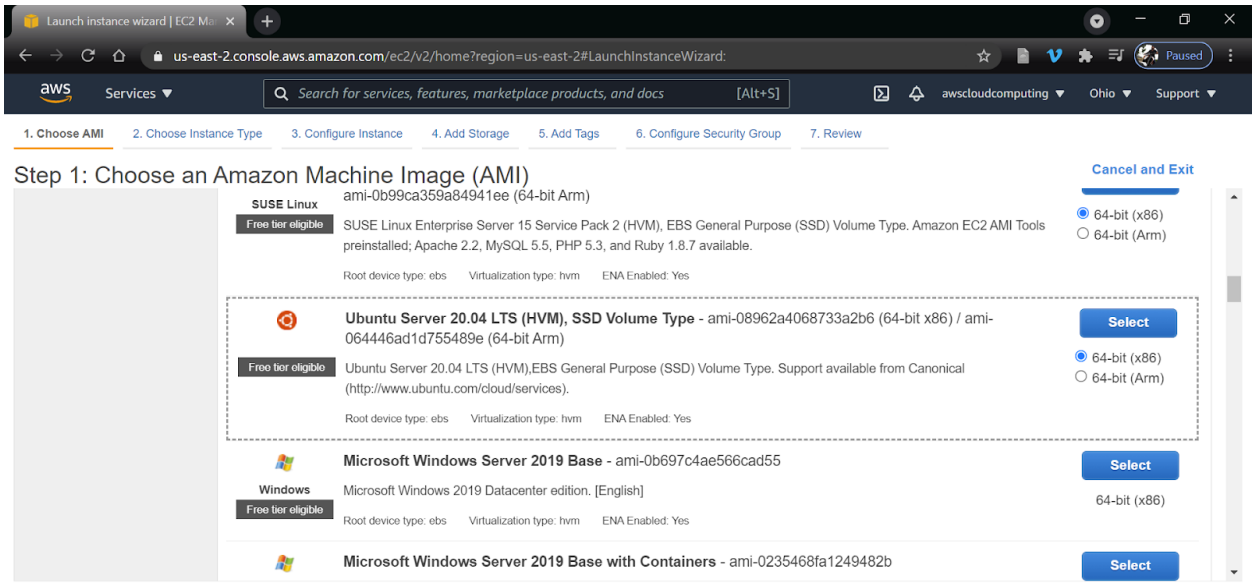
Some key advantages are Cost effective storage classes, easily managed data and access controls, query-in-place and process on-request, performance, scalability, availability and durability.

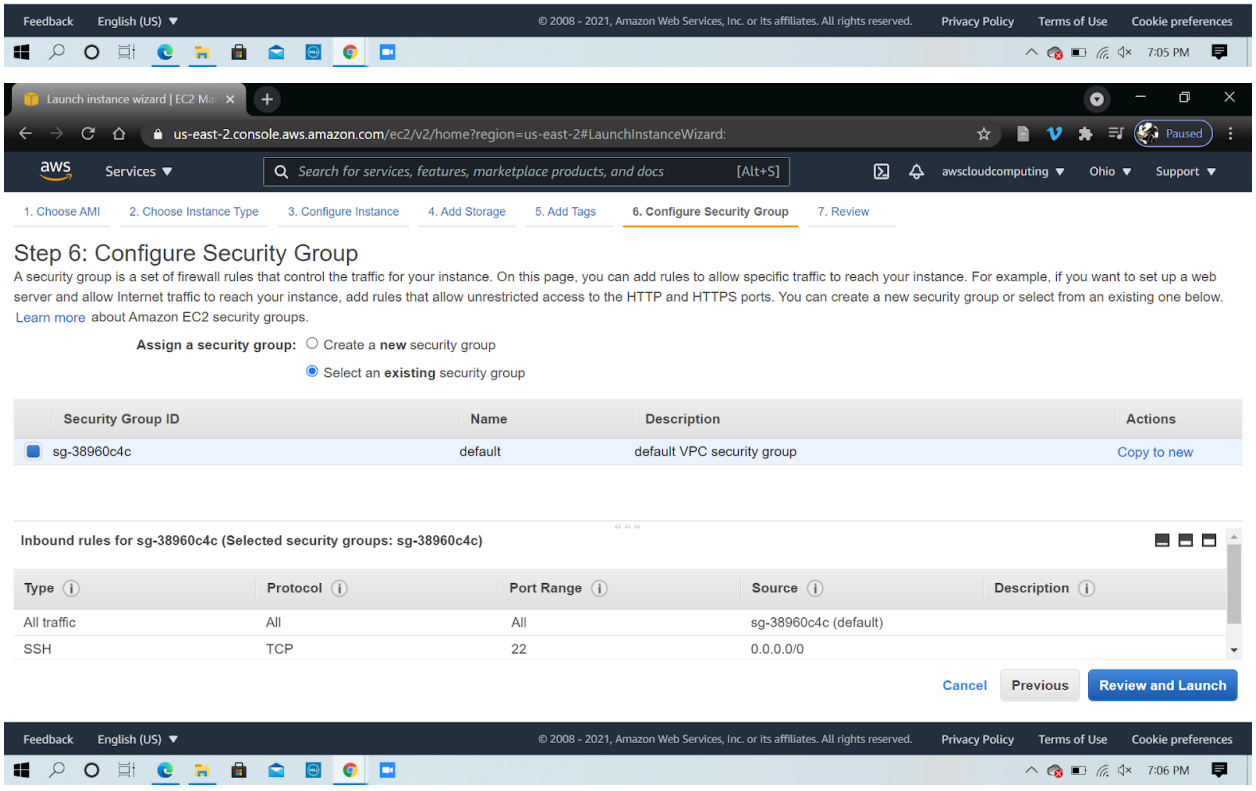


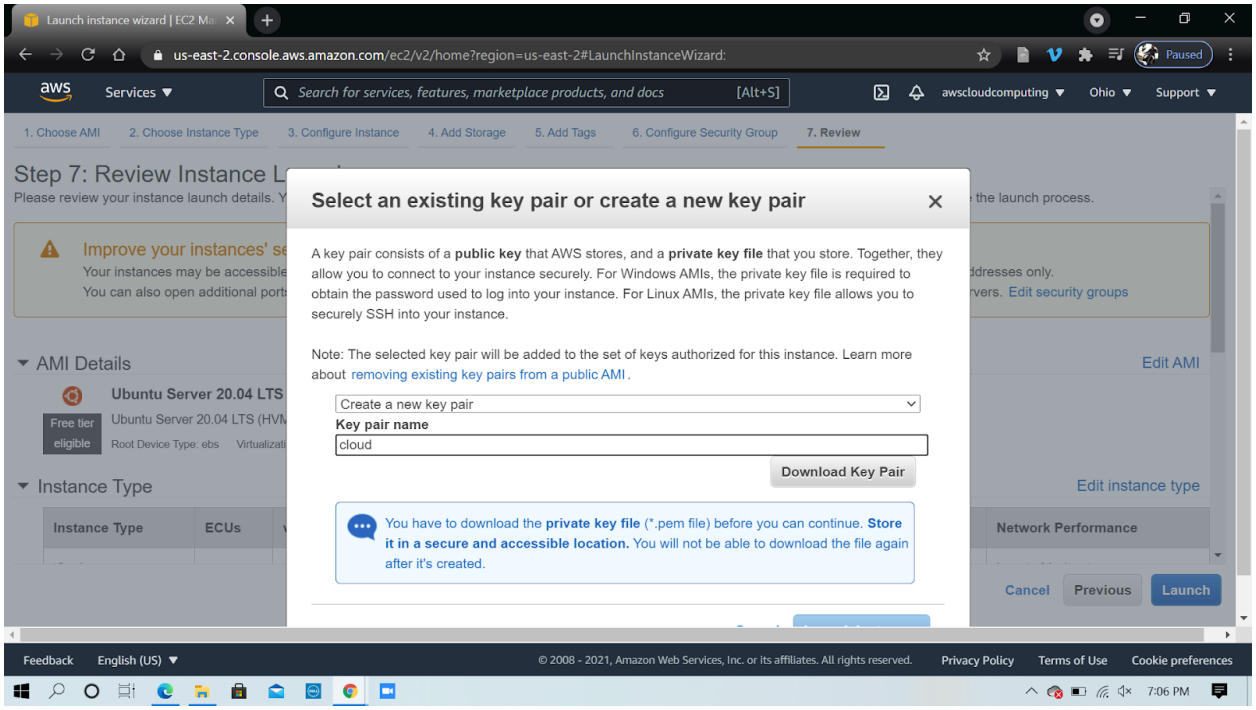


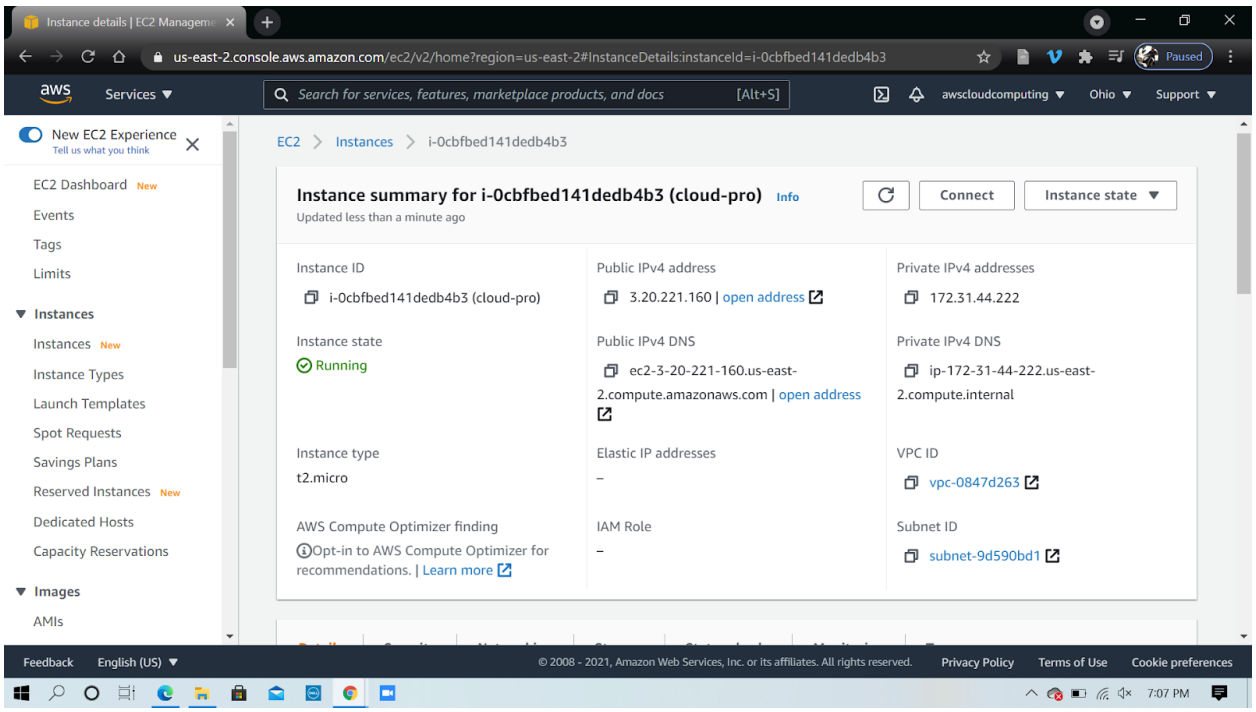
- **Deploy a EC2 server**

It is done by using Amazon EC2 service, for this we should select EC2 service from AWS management console dashboard. Then launch an instance, where the ubuntu server is used. Select all the desirable options, configure security groups, create a new key pair or select existing one if you have one, and then review and launch. Some key advantages are reliable, scalable and infrastructure on demand.







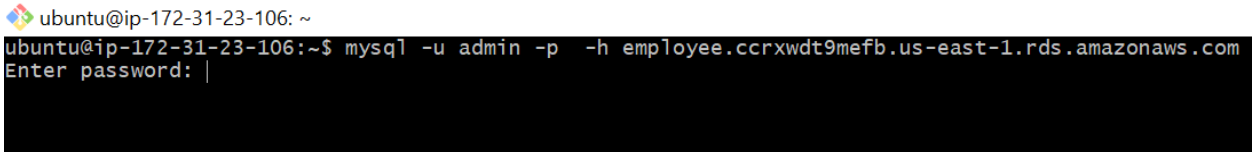


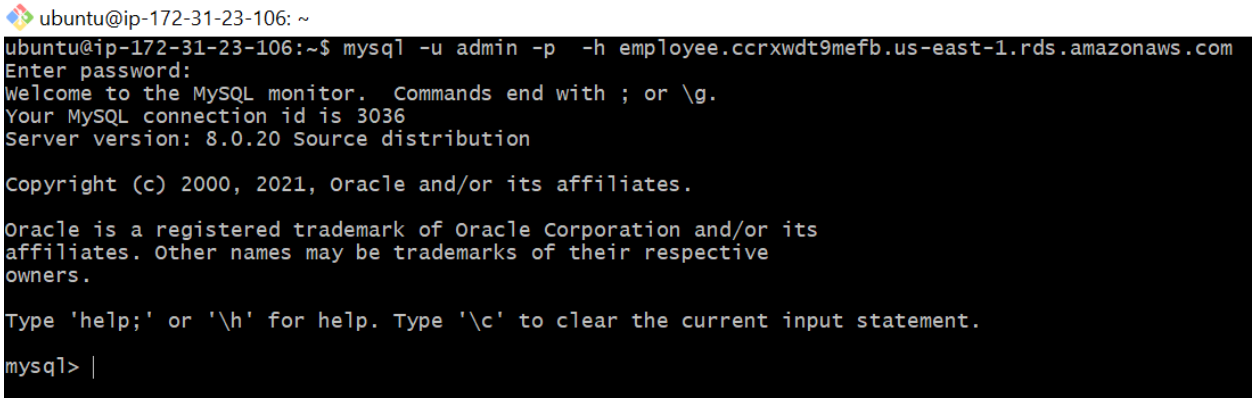
Next connect the server to our machine.

To connect from windows to server one can use Putty or Git bash or Super Putty which is a premium tool.
To connect from a mac one can use the terminal itself, this will connect to an instance deployed in AWS.

Next connect to our RDS

First deploy MySQL client on the server which helps to connect to our RDS machine, for this install MySQL client.





After installing MySQL client software now we connect to the RDS, so that now we will be in a MySQL shell.

Next deploy database on server

For this first create a database and then create a table in that database for the information to be stored.

// commands

After logging into databases:

Show databases; // gives list of Databases present

```
mysql> Show databases;
+--------------------+
| Database           |
+--------------------+
| employee           |
| information_schema |
| mysql              |
| performance_schema |
+--------------------+
4 rows in set (0.03 sec)

mysql>
```

(need to create a database now)

create database <database-name>; // will create a database with the given name.

```
mysql> create database EMP;
Query OK, 1 row affected (0.05 sec)

mysql>
```

(need to switch to created database)

Use <databasename>;

```
mysql> use EMP;
Database changed
mysql>
```

(now we can create tables in the created database)

Create table <table-name>(name varchar (20), age varchar (20)); // creates a table inside the database.

```
mysql> use EMP;
Database changed
mysql> create table EMP (name varchar (20), age varchar (20));
Query OK, 0 rows affected (0.09 sec)

mysql>
```

Show tables;

```
mysql> show tables;
+---------------+
| Tables_in_EMP |
+---------------+
| EMP           |
+---------------+
1 row in set (0.01 sec)

mysql>
```

2) The second phase is to build the website, configure the website code and deploy the website on EC2 server.

- Build the website

Here to build the website we have used the Python Flask framework. Flask is known as a simple but extensible framework.

- Configure the website code

1) Provide Host name of RDS in host region of config code.
2) Provide username, password and database name in config code.
3) Provide bucket name and region in the configuration code.

- Deploy website on EC2 server

Make the website available on EC2 server.

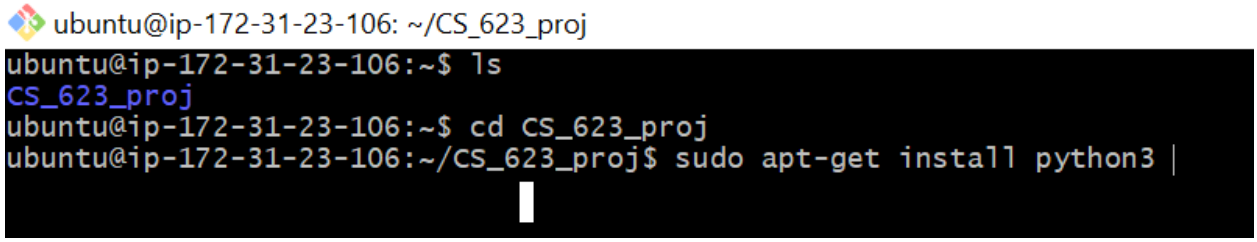
After connecting to EC2 instance we need to clone a git repository:

```
ubuntu@ip-172-31-23-106: ~
ubuntu@ip-172-31-23-106:~$ git clone "https://github.com/mkodali37/CS_623_proj.git"
```

Then navigate to the project folder and install the required packages:

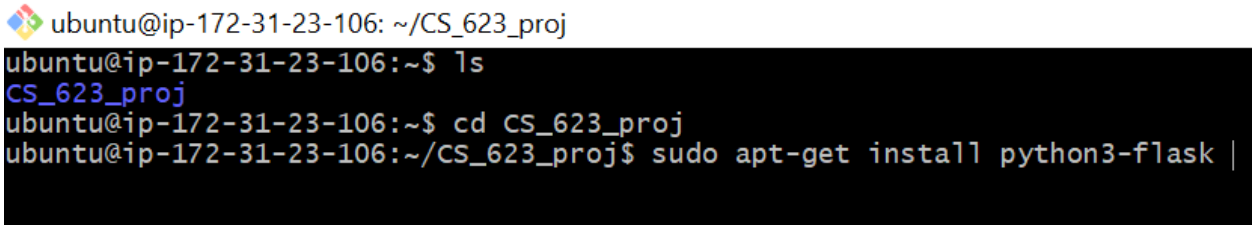
Execute all the below commands on the server to install them on server.

- **sudo apt-get install python3**



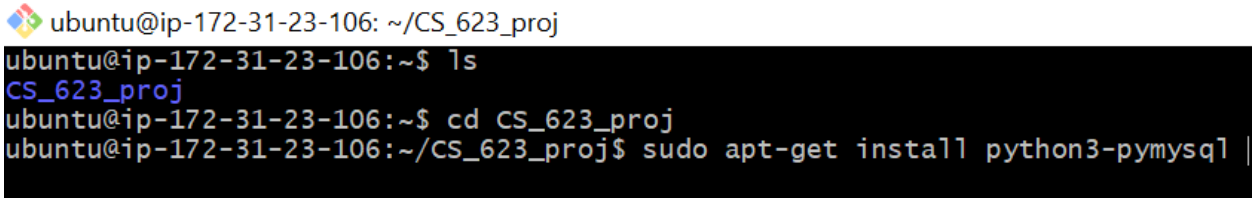
Python3 is the language that is coded for the website, we must install it on an existing system where we are running the website.

- **sudo apt-get install python3-flask**



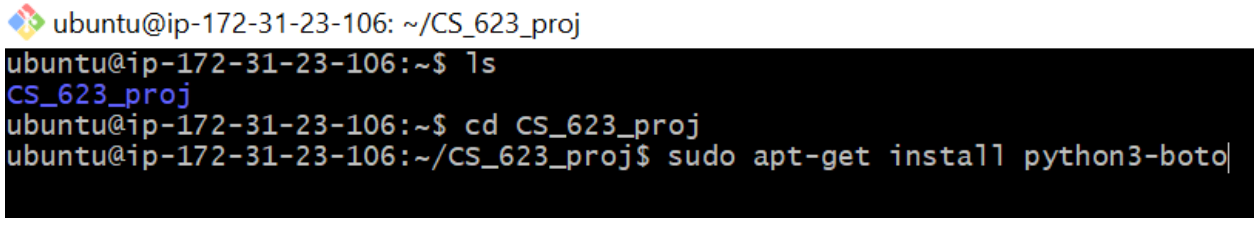
Flask helps to go ahead and publish a website on to that server.

- **sudo apt-get install python3-pymysql**



It is a library which helps the website to connect to the MySQL server.

- **sudo apt-get install python3-boto**



It is an SDK which helps to connect to S3 service where we upload files.

Then give EC2 instance or server access to basically upload data to S3. We need to do this because on the server we are running some code and that code is trying to interact with the AWS S3 service, when we connect with our RDS we should give authentication to our S3 service, so for that we must give the IAM role to our instance.

To give IAM role to our instance, follow the below steps -

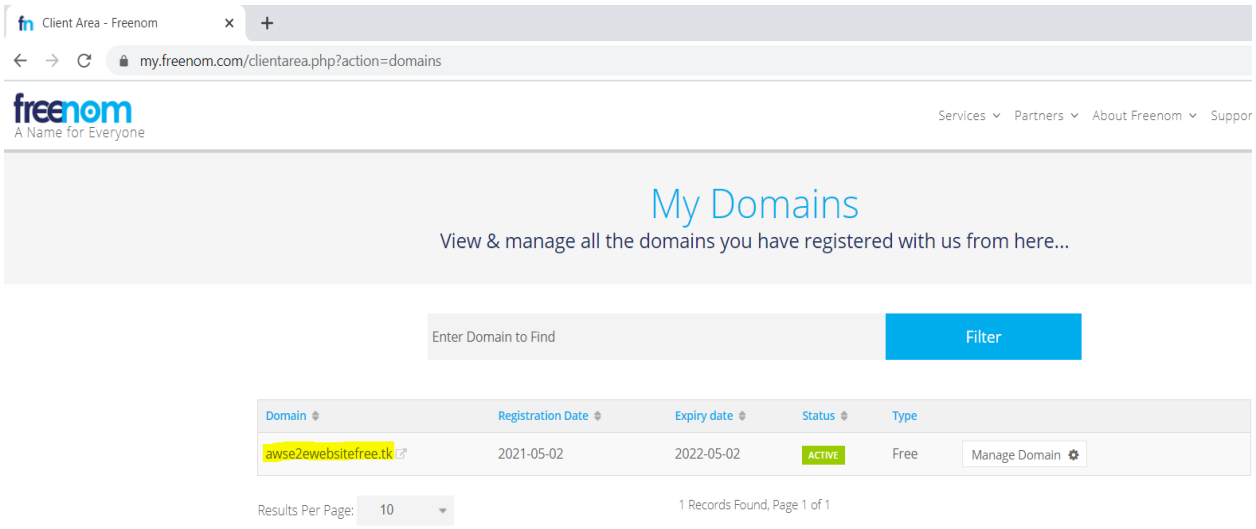
- Select IAM service from AWS management console dashboard.
- Then create an admin role for our AWS account.
- Then select roles for our EC2 instance.
- Then we should go to our EC2 service and click on the attach/replace IAM role and select the role which we created from it. Now EC2 is connected to RDS, EC2 is connected to S3, and website deployed in EC2 server.

3) The third phase is routing domain to EC2 server.

To route domain to EC2 server we need to use Route 53 service from AWS management console dashboard and then connect Route 53 service to EC2 server.

- Route 53 service

First, select Route 53 service from AWS management console dashboard.
Then, purchase a domain name for the website.
Then, create a hosted zone and mention the purchased domain name here.



Route 53 > Hosted zones > Create hosted zone

Create hosted zone Info

Hosted zone configuration

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 Info

This is the name of the domain that you want to route traffic for.

awse2ewebsitefree.tk

Valid characters: a-z, 0-9, ! " # $ % & ' ( ) * + , - / : ; < = > ? @ [\ ] ^ _ ` { | } . ~

Description - *optional* Info

This value lets you distinguish hosted zones that have the same name.

The hosted zone is used for...

The description can have up to 256 characters. 0/256

Type Info

The type indicates whether you want to route traffic on the internet or in an Amazon VPC.

- (•) **Public hosted zone** — A public hosted zone determines how traffic is routed on the internet.
- ( ) **Private hosted zone** — A private hosted zone determines how traffic is routed within an Amazon VPC.

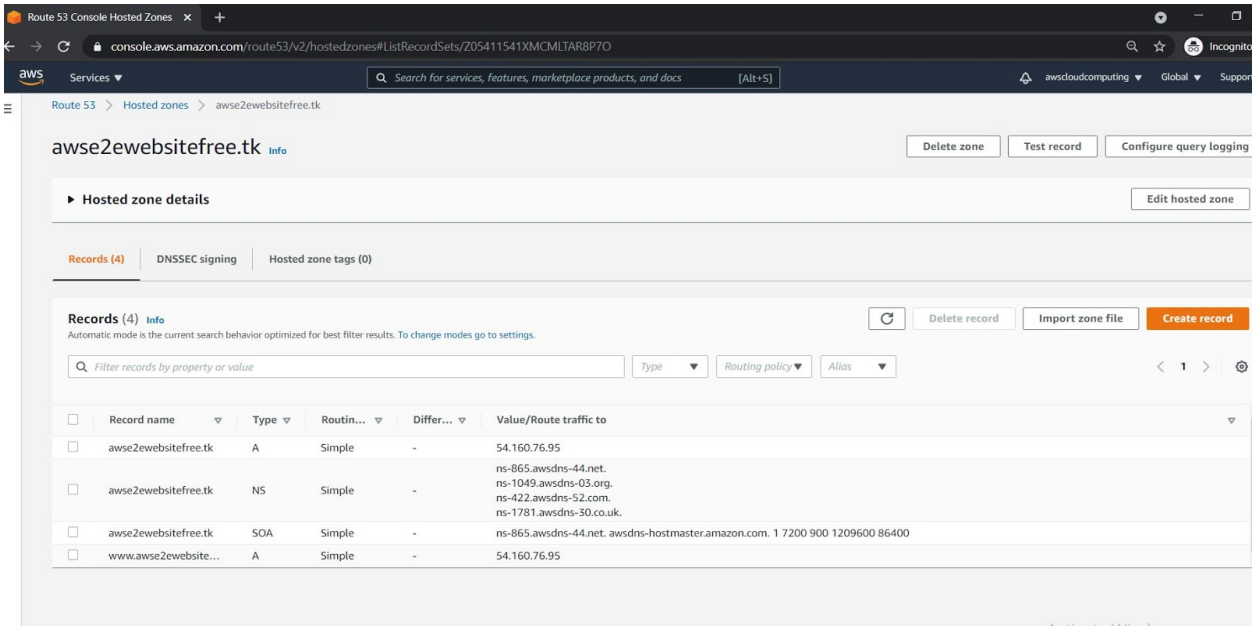
Tags Info

Apply tags to hosted zones to help organize and identify them.

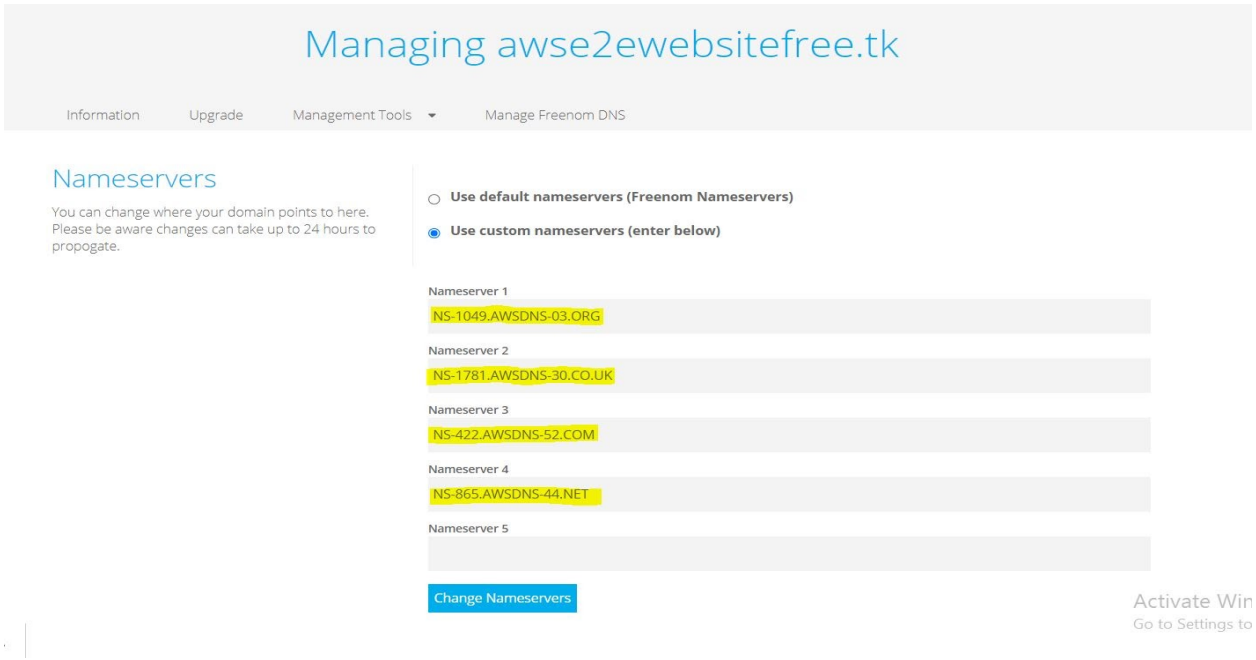
No tags associated with the resource.

Add tag

You can add up to 50 more tags.

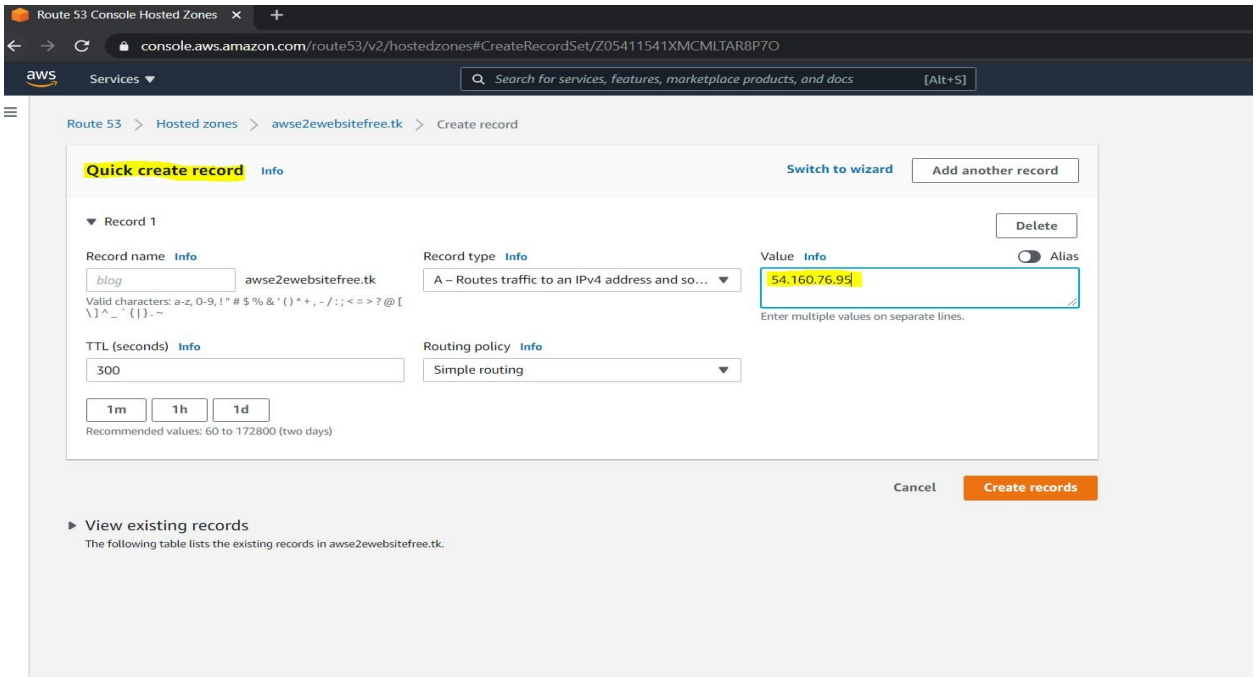


Now, configure the Name Servers.

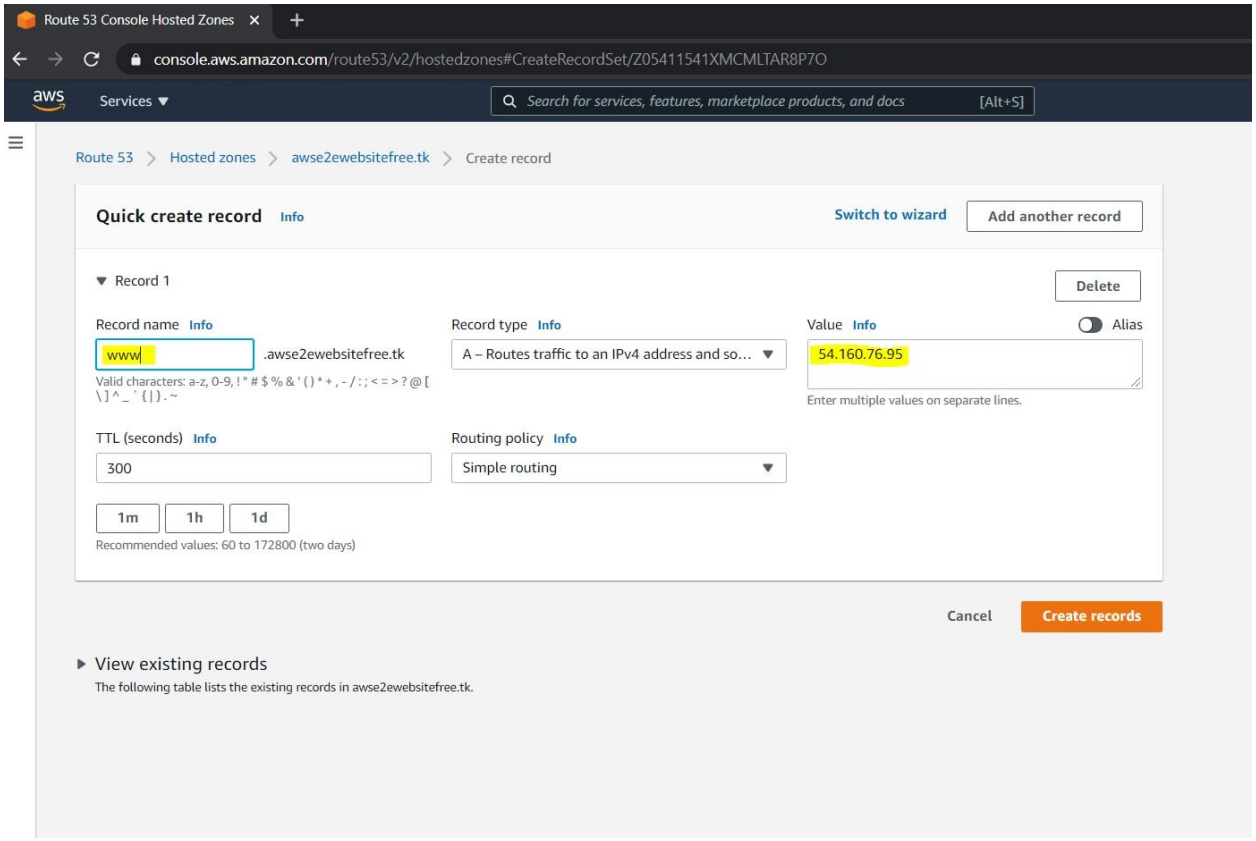


- Connect Route 53 service to EC2 server

For this first create a record set and mention EC2 ip address in the value box.



Then, create another record set and mention the name as WWW and put the same ip address in the value box.



3.2 Challenges team faced during building this project

- Different Amazon services being used in this project and connection to RDS from Git bash/ Putty is one of the challenges we faced.
- All the commands to install flask, boto and pymysql in the server.
- In the second phase initially we didn't give EC2 instance access to upload data to S3, after that we made a research on IAM roles and used IAM service to upload pictures to S3 bucket.
- Usage of **Route 53** service to route domain to EC2 server as it is not available to all. We purchased a domain name for implementing Route 53 service.

CHAPTER 4: CODE & TECHNOLOGY

4.1 Programming language

1) **Frontend**
In frontend we used Python and Html languages.

2) **Backend**
In backend we used RDS (MySQL Database)

4.2 Cloud services

1) **Amazon RDS**:
We are using Amazon RDS service, which is used to operate and scale a relational database in the cloud. In this we are using MySQL database and this is used to store the entries. It provides cost-efficient, resizable capacity for an industry-standard relational database and manages common database administration tasks.

2) **Amazon S3**:
We are using Amazon S3 service, which is a Storage for the internet. We can use it to store and retrieve any amount of data at any time, from anywhere on the web.

3) **Amazon EC2** :
We are using Amazon EC2 service, it is a web service for launching and managing Linux/UNIX and Windows Server instances in Amazon's data centers.

4) **Amazon IAM**
AWS Identity and Access Management (IAM) is a web service that helps you securely control access to AWS resources. When we connect with our RDS we should give authentication to our S3 service, so that we must give the IAM role to our instance.

5) **Amazon Route 53**
AWS Route 53 is a service we are using to route domain to EC2 server.

CHAPTER 5: GIT ACCESS & SOURCE CODE

Below is the GitHub Link for our project:
https://github.com/gangareddynachu/AWS_E2E_Website

We have a component where employees can provide their information and when they click to update the database this code will run and establishes a connection to store data and to retrieve the data also. The python code for this is below -

The python code :

EmpApp.py is the entry point of our application:

EmpApp.py

```python
import io

from flask import Flask, render_template, request
from pymysql import connections
import os
import boto3
from config import *
from PIL import Image
import json

app = Flask(__name__)

bucket = custombucket
region = customregion

db_conn = connections.Connection(
    host=customhost,
    port=3306,
    user=customuser,
    password=custompass,
    db=customdb

)
output = {}
table = 'employee'


@app.route("/", methods=['GET', 'POST'])
def home():
    return render_template('AddEmployee.html')

```

EmpApp.py

```
@app.route("/about", methods=['GET','POST'])
def about():
    return render_template('about.html')


@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']

    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)
```

EmpApp.py

```
        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:
            return render_template('Error1.html')

    finally:
        cursor.close()

    print("all modification done...")
    return render_template('EmployeeAdd_Success.html', name=emp_name)

@app.route("/admin", methods=['GET', 'POST'])
def admin():
    return render_template("admin.html")
```

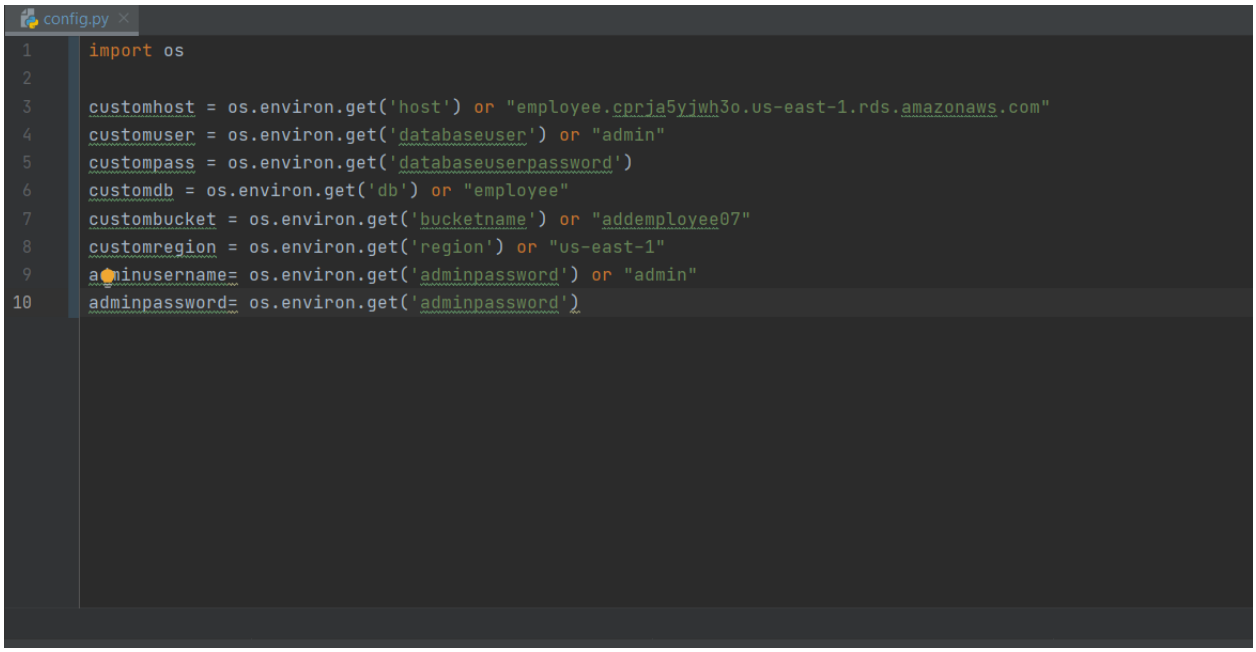
EmpApp.py

```python
@app.route("/getemp", methods=['GET', 'POST'])
def GetEmp():
    return render_template("GetEmployeeInfo.html")

@app.route("/fetchdata", methods=['POST'])
def FetchEmp():
    emp_id = request.form['emp_id']

    output = {}
    select_sql = "SELECT empid, fname, lname, pri_skill, location from employee where empid=%s"
    cursor = db_conn.cursor()
    emp_image_file_name_in_s3 = "emp-id-" + str(emp_id) + "_image_file"
    s3 = boto3.resource('s3')

    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

    image_url = "https://s3{0}.amazonaws.com/{1}/{2}".format(
        s3_location,
        custombucket,
        emp_image_file_name_in_s3)

    try:
        cursor.execute(select_sql, (emp_id))
        result = cursor.fetchone()
```

EmpApp.py

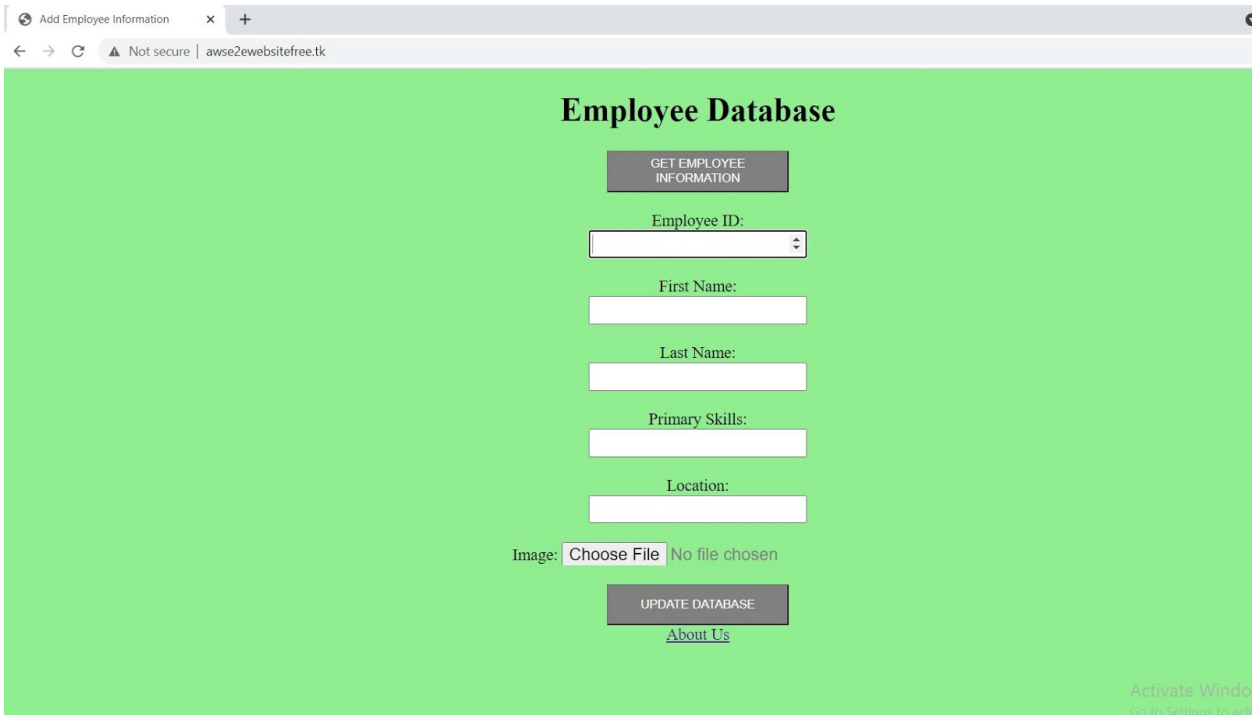
```python
    try:
        cursor.execute(select_sql, (emp_id))
        result = cursor.fetchone()

        output["emp_id"] = result[0]
        print('EVERYTHING IS FINE TILL HERE')
        output["first_name"] = result[1]
        output["last_name"] = result[2]
        output["primary_skills"] = result[3]
        output["location"] = result[4]
        print(output["emp_id"])

        return render_template("EmployeeInfo_Output.html", id=output["emp_id"], fname=output["first_name"],
                               lname=output["last_name"], interest=output["primary_skills"], location=output["location"],

    except Exception as e:
        print(e)
        return render_template('Error.html')
if __name__ == '__main__':
    app.run(host='0.0.0.0', port=80, debug=True)
```

There is a small python code below for the configuration - **Config.py**

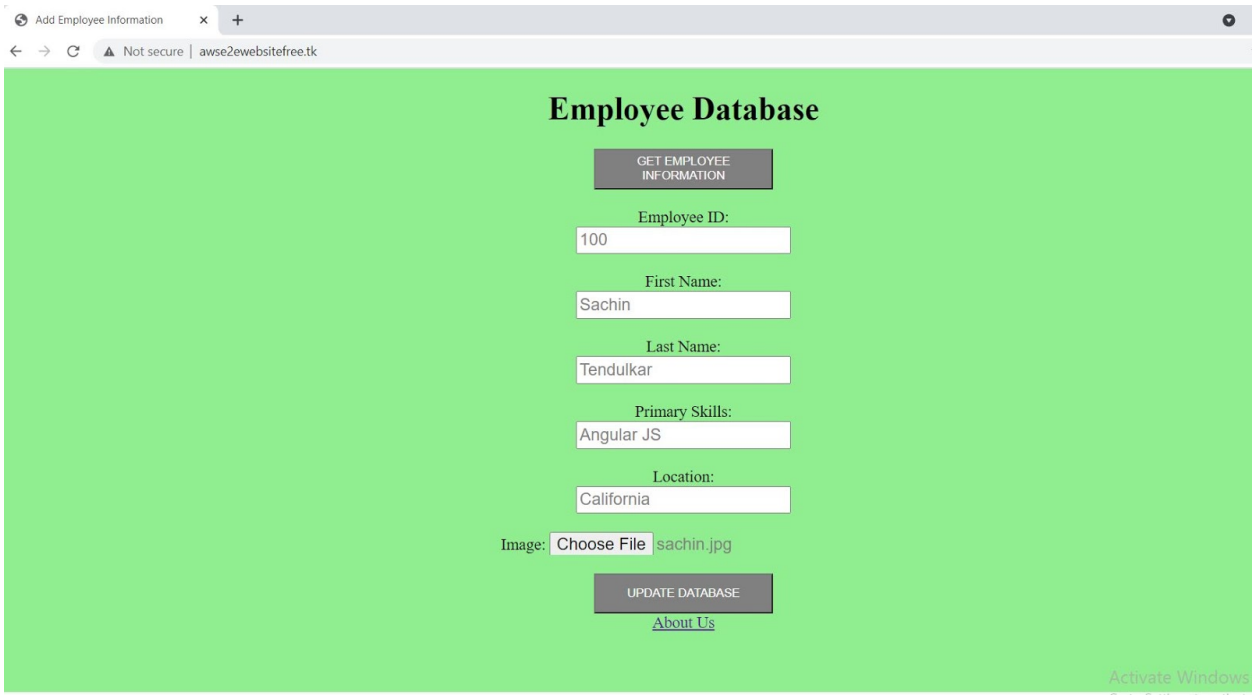


CHAPTER 6: DEMO SCREENSHOTS FROM WEBSITE

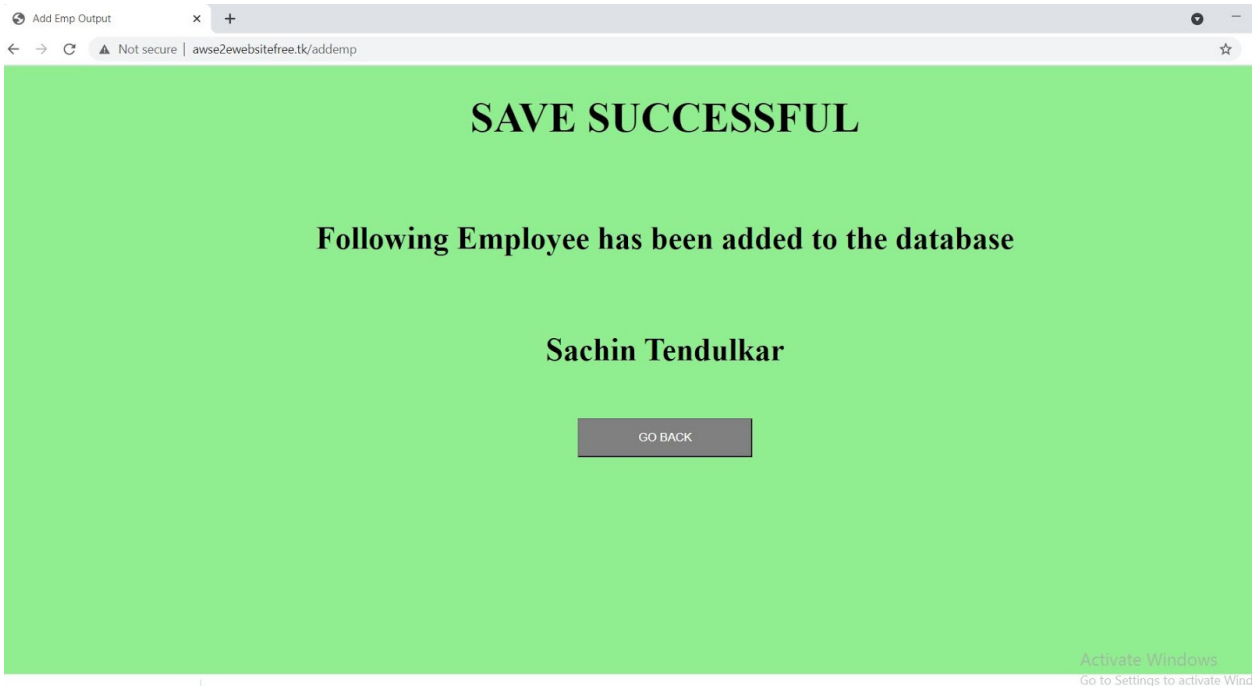
Home Page:



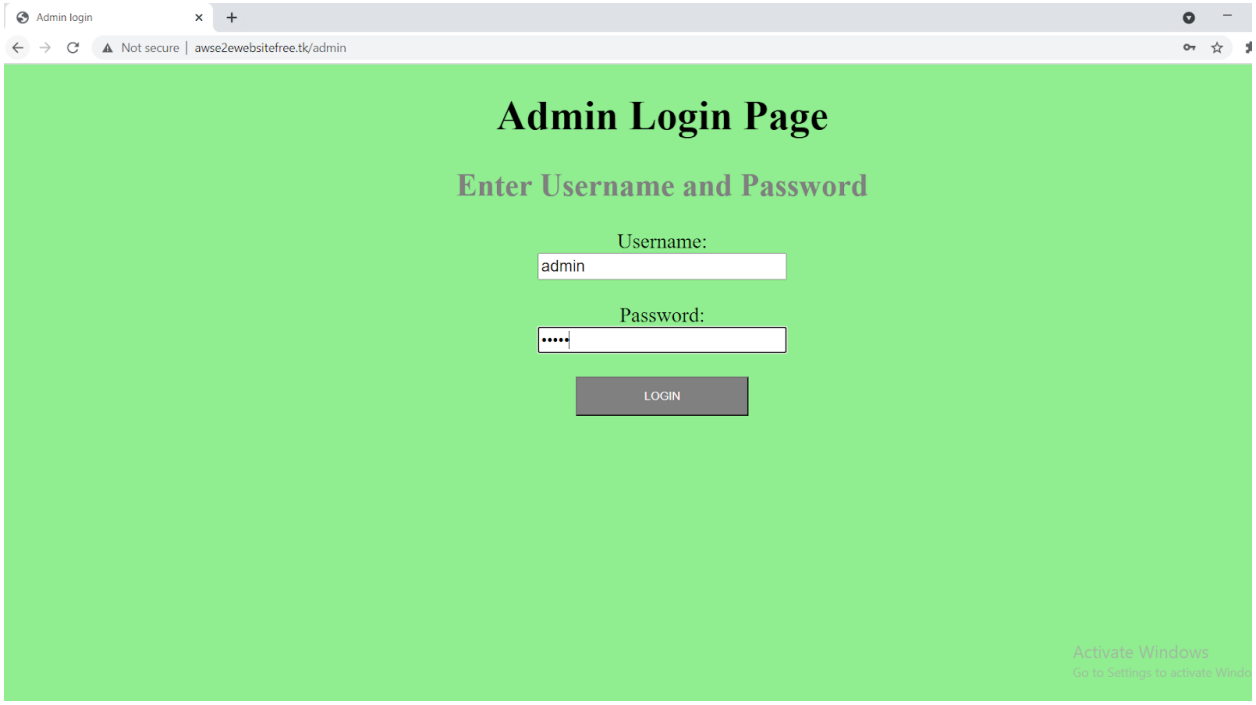
Add New Employee Page:



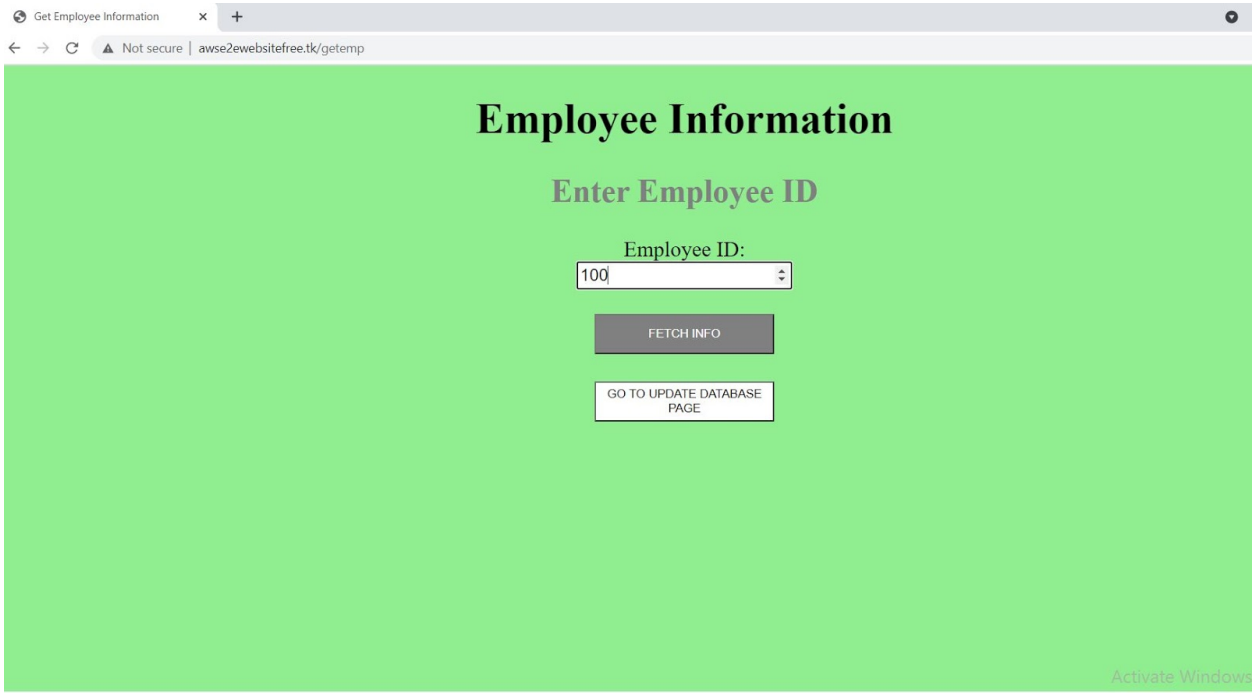
Add Employee Success Page:



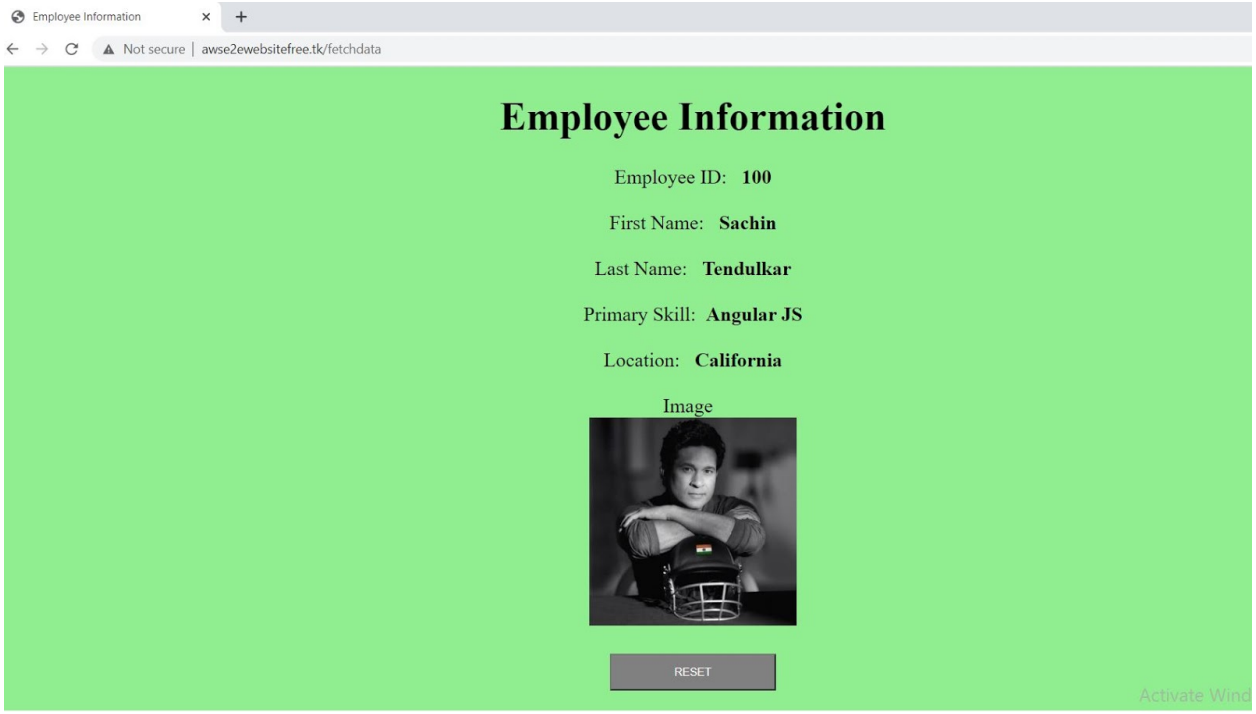
Logging as Administrator:



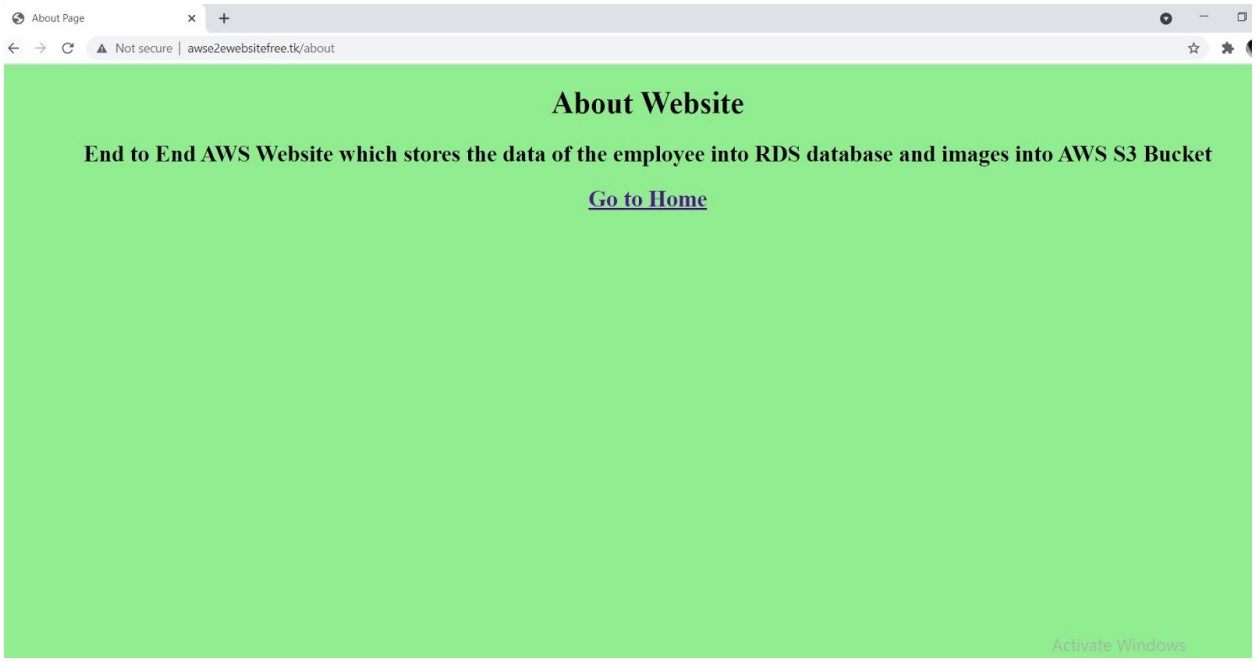
After Logging as Administrator, will redirect to Get employee page where we Admin needs to enter Employee ID to retrieve the employee details:



Display Employee Info Page:



About Page:


About Page
Not secure | awse2ewebsitefree.tk/about
About Website
End to End AWS Website which stores the data of the employee into RDS database and images into AWS S3 Bucket
Go to Home
Activate Windows

CHAPTER 7: OUTSTANDING FEATURES OF THIS PROJECT

1. Usage of different cloud services viz, EC2, S3, RDS, Route 53, IAM
2. Elimination of Infrastructure, and upfront investment.
3. Enhanced level of flexibility and scalability in comparison to traditional data centres.
4. Storing and retrieving of data in a fast and efficient manner.
5. Simple Project management.
6. Accessible from anywhere in the world and Increased productivity.

CHAPTER 8: FUTURE ENHANCEMENTS

1. Implementation and integration of AWS chatbot to our web application where employees can interact with it and have answers to their questions.

2. Implementation of AWS DynamoDB instead of MYSQL database, as AWS DynamoDB is a key-value and document database that delivers single-digit millisecond performance at any scale.DynamoDB can handle more than 10 trillion requests per day and can support peaks of more than 20 million requests per second. So, AWS DynamoDB can be one of our future enhancements.

CHAPTER 9: REFERENCES

1. https://docs.aws.amazon.com/
2. https://docs.aws.amazon.com/s3/index.html
3. https://docs.aws.amazon.com/pythonsdk/
4. https://docs.aws.amazon.com/route53/
5. [1] “Flask web development, one drop at a time”, https://flask.palletsprojects.com/en/1.1.x/
   This is the official documentation of the Flask web application framework.
6. [2] "Introducing Python: modern computing in simple packages (Second edition.)", Bill Lubanovic, O'Reilly Media, November, 2019., Chapter 18.
   This is a book that discusses the basic usage of Python programming language within various topics, including the web systems. Bill Lubanovic is a senior software engineer currently working for Internet Archive. O'Reilly is a popular publisher that has published various books related to software development.