# CCS6344 T2410 Assignment 2 Submission

Group Name: Group 16

| | |
|---|---|
| ASYRANI SYAZWAN BIN YUHANIS | 1211103222 |
| OOI PUI KEAT | 1201101582 |
| DIVYASHREE A/P SELVANYGAM | 1221303777 |

# Design Implementation:

- ## Client Business Model

The wedding and catering management system is designed to facilitate the organization, management,and execution of wedding and catering services. Our client, a wedding and catering service provider needs an efficient system to enhance their operations through a comprehensive management system. The wedding and catering management system aims to streamline the entire process of planning, organizing, and executing wedding and catering events, ensuring efficient management and high customer satisfaction. The key feature of the system is for managing and registration of wedding and catering staff.

- ## Model Analysis

The modules that we used in Amazon Web Services(AWS) for our client's business model are Amazon VPC, Amazon EC2 and Amazon RDS.

**Amazon VPC (Virtual Private Cloud):** Aims to provide a secure and isolated network environment for the wedding and catering management system. By using a VPC, we get to control the network settings and ensure that the resources are secure and properly configured.

**Advantages:**

**Security:** Isolates resources within a private network, enhancing security

**Customizability:** Allows customization of network configurations

**Control:** Provides fine-grained control over inbound and outbound network traffic through security groups and network ACLs

**Amazon EC2 (Elastic Compute Cloud):** This is to provide scalable and flexible compute capacity in the cloud. For the wedding and catering management system, EC2 is used to host the web server and application, ensuring that the system is always available and can handle varying levels of traffic.

**Advantages:**

**Scalability:** Easily scale up or down based on demand, ensuring optimal performance during peak times

**Flexibility:** Choose from a variety of instance types and configurations to meet specific needs and budget constraints

**Reliability:** Benefit from AWS's infrastructure, which provides high availability and redundancy.

**Amazon RDS (Relational Database Service):** To provide a managed relational database service that simplifies the setup, operation, and scaling of a relational database. RDS is used for the wedding and catering management system to manage the MySQL database, which stores all the critical data related to staff.

**Advantages:**

**Managed service:** Automates routine database tasks such as provisioning, patching, backup, recovery, and scaling

**Scalability:** Easily scale the database storage and compute resources

**Security:** Provides encryption at rest and in transit, and integrates with AWS IAM for access control

**High availability:** Supports Multi A-Z deployments for enhanced availability and durability

# Model Implementation:

## Design Description

**VPC (Virtual Private Cloud):**

1. Configure the VPC:
    - Enter details such as name, number of availability zones, subnets, and NAT gateways.
    - Create the VPC.

**EC2 Instance:**

1. Configure EC2 Instance:
    - Select Amazon Linux as the AMI for compatibility with AWS services.
    - Choose t2.micro instance type for cost-effectiveness and expected load.
    - Select the created VPC for networking control.

2. Configure Security Group:
    - Create a security group named ‘WeddingCatering’.
    - Allow HTTP, HTTPS, and SSH access.

3. Key Pair:
    - Create a key pair named ‘WeddingCatering.pem’ for secure login

4. Configuring the Web Server on EC2:
    - Connect using SSH: ‘ssh i /path/to/WeddingCatering.pem ec2-user@ip-address’.
    - Create a temporary directory: ‘mkdir temp’.
    - Copy application files to the temp directory: ‘scp -i WeddingCatering.pem user_reg.zip ec2-user@ip-address:temp’.
    - Unzip the application files: ‘unzip user_reg.zip’.
    - Moves files to ‘/var/www/html’: ‘sudo mv* /var/www/html’.
    - Install Apache: ‘yum install -y httpd’
    - Start Apache: ‘systemctl enable httpd’ and ‘systemctl start httpd’

**RDS (Relational Database Service):**

1. Create RDS Instance:
    - Choose MySQL as the engine type.
    - Use the Free tier template for cost management.
    - Set DB Instance Identifier to ‘WeddingCatering’.
    - Set master username and password to ‘admin’ and ‘admin123’

2. Configure Security Group:
    - Use the existing VPC and add a rule to allow MySQL access.

3. Database Endpoint:
    - Connect the application to the RDS instance using the endpoint ‘weddingcatering.cpffhoagrni.us-east-1.rds.amazonaws.com’.

Connecting EC2 and RDS:

1. Update Database Configuration:
    - Modify ‘database.php’ to use the RDS endpoint. Master username, and master password: ‘sudo nano database.php’.

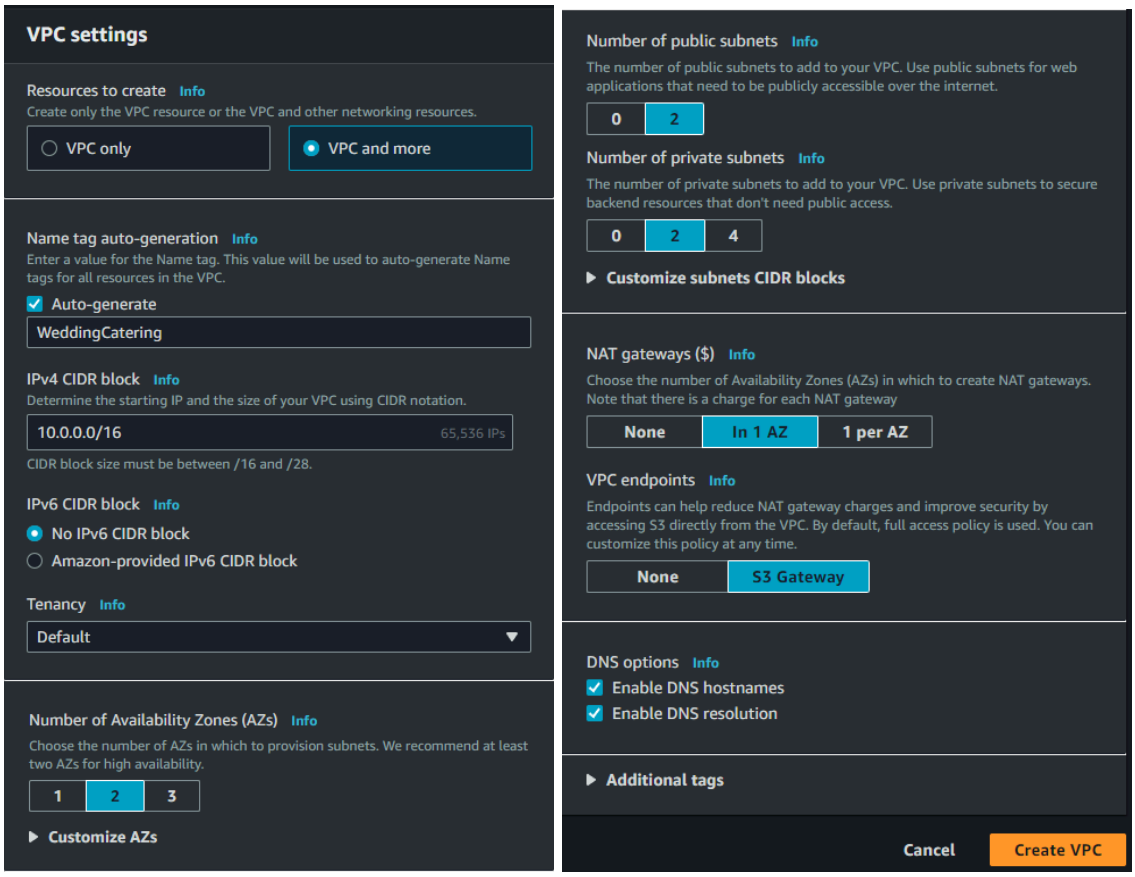
**Testing:**

1. HTTP:
    - Access the web application using ‘http://ip-address’.
    - Some functions may show the PHP code instead of executing.

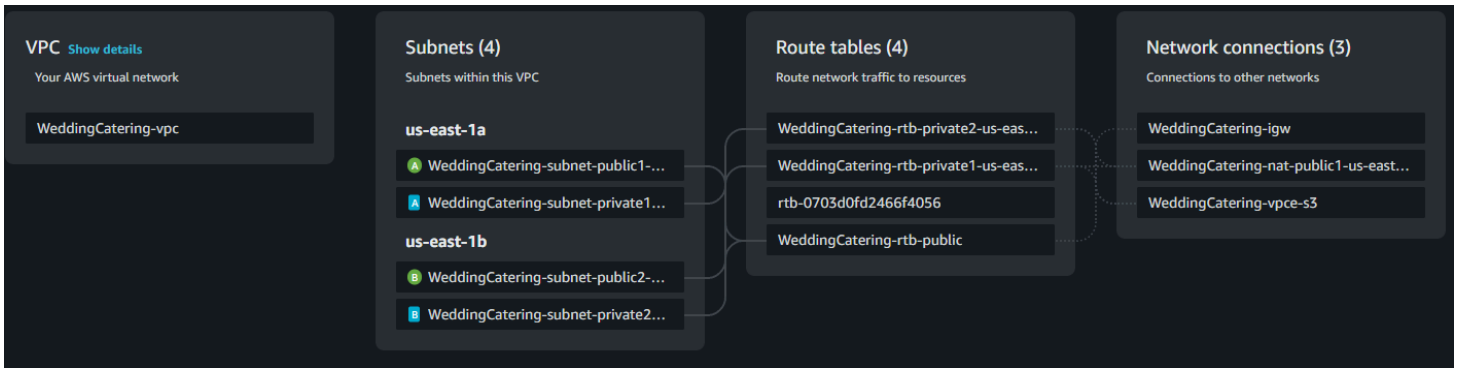
2. HTTPS:
    - Access using ‘https://ip-address’ may not work correctly

# Implementation Process

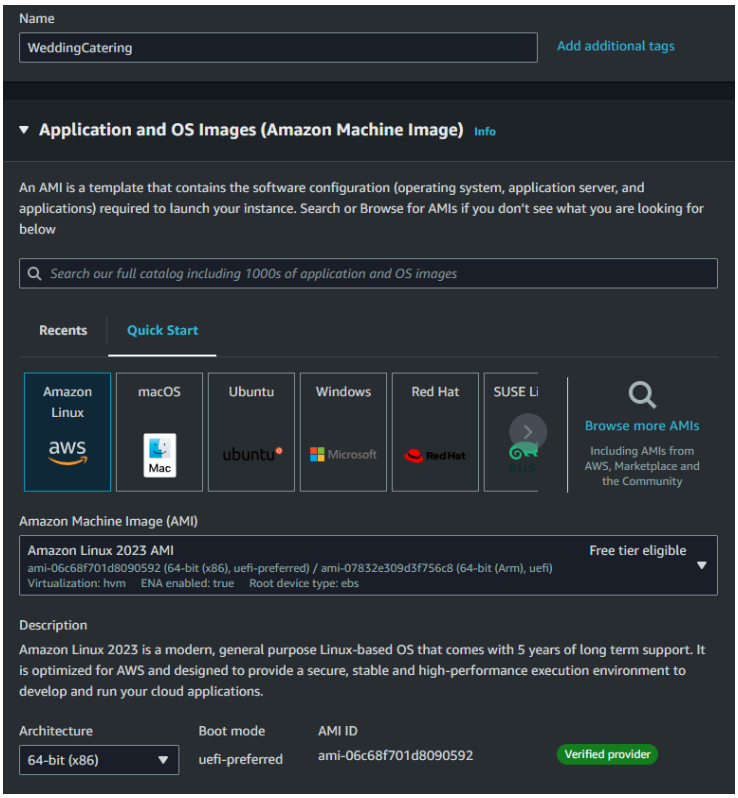
- **VPC**
  1. **Configure the VPC** - Enter the details needed for the VPC settings such as the Name and Number of Availability Zones (AZs), subnets and NAT Gateways. Click “Create VPC” to finish configuring.

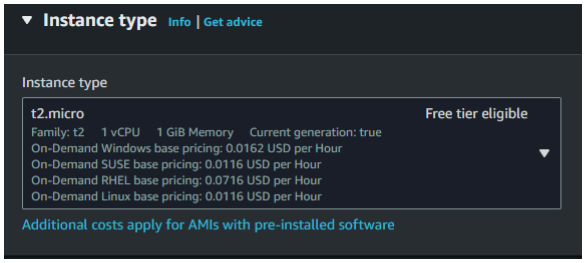


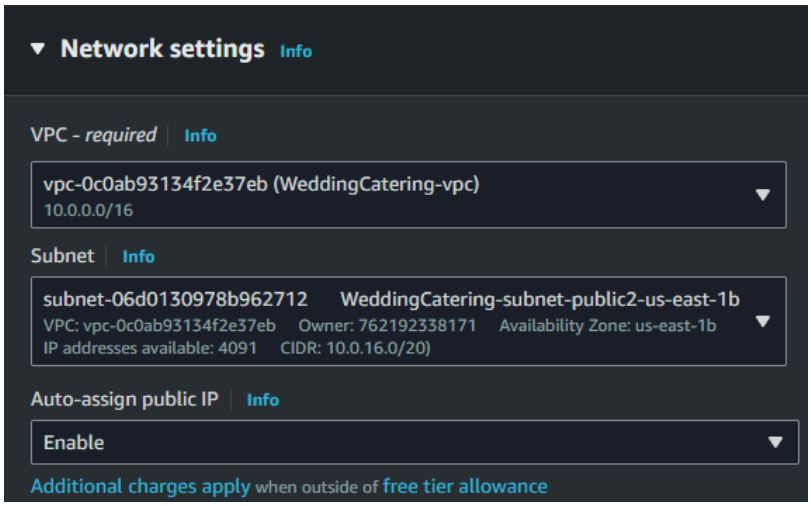

  2. **VPC is created**



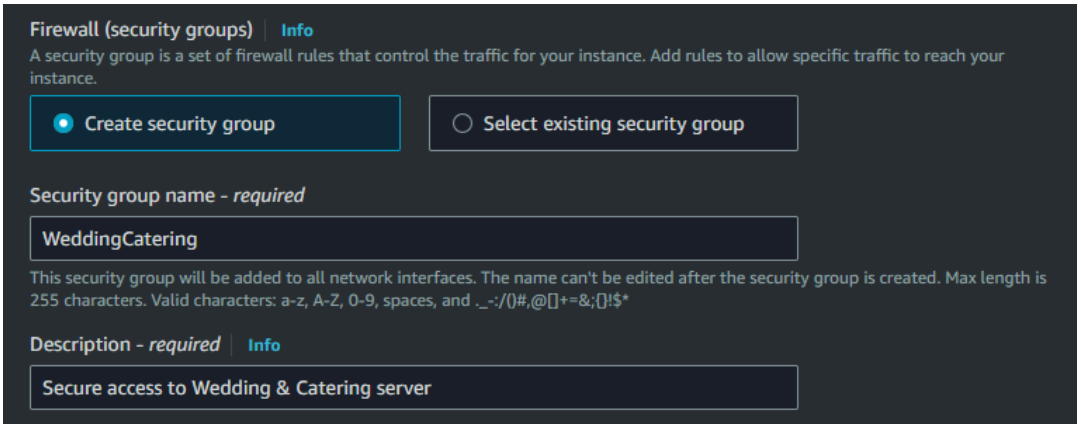
- **EC2 Instance**
    1. **Configure EC2 Instance** - We choose Amazon Linux as the AMI for its compatibility with AWS services. For the Instance Type, we select t2.micro based on expected load and cost consideration. As for the network, we selected the WeddingCatering-vpc for easier networking control within the application.

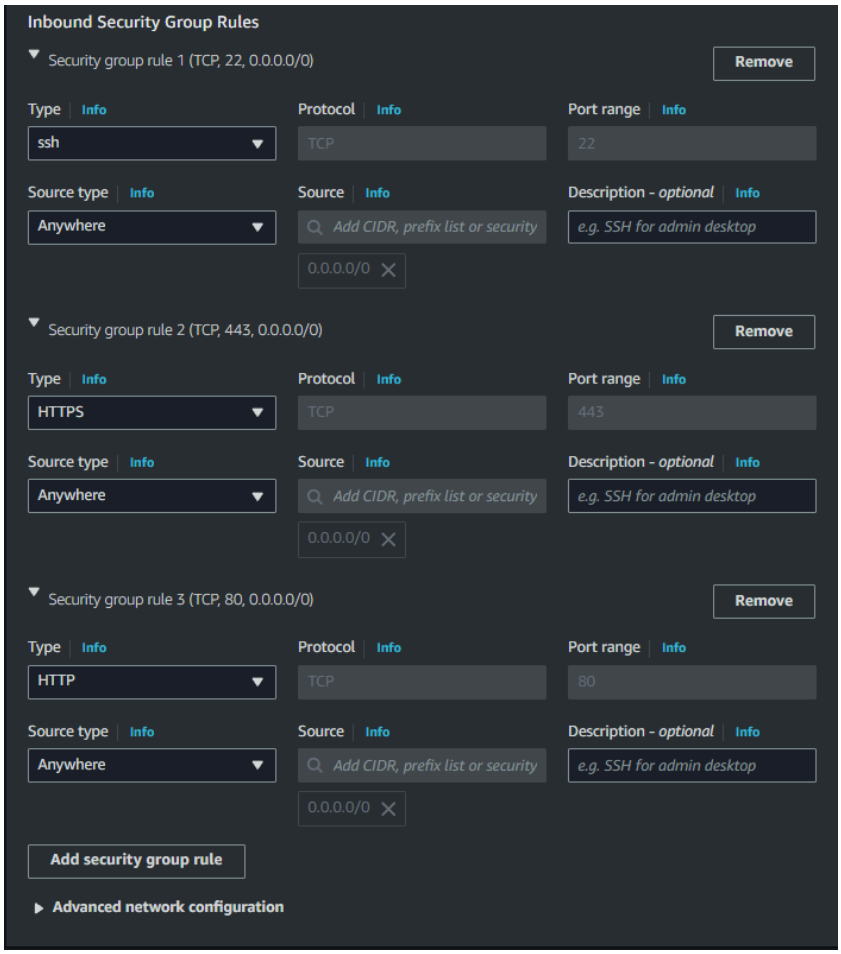


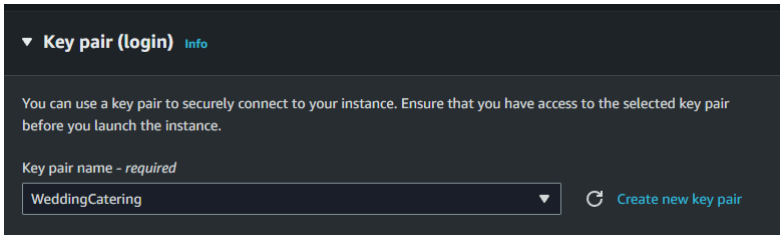



2. **Configure Security Group** - Create a security group to control inbound and outbound traffic. We called the security group's name as WeddingCatering. This will allow HTTP, HTTPS and SSH to be accessed through the IP Address obtained from EC2 Instance.



3. **Key Pair** - By having a key pair, it can provide a secure login information for the instance. The key pair name is WeddingCatering.pem




4. **Configuring the Web Server on EC2** - First, we need to connect with SSH so that we can access the EC2 Instance. We can do so by running “ssh -i (path to WeddingCatering.pem) ec2-user@ip-address”.

```
C:\Users\Asyer>ssh -i "C:\Users\Asyer\OneDrive\Desktop\MMU\DEGREE\Semester 5\Database and Cloud Securi
ty\Assignment\Ass 2\WeddingCatering.pem" ec2-user@ec2-54-159-133-108.compute-1.amazonaws.com
       #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
```

Next, we need to create a temp directory by using “mkdir temp”.

```
[ec2-user@ip-10-0-24-185 ~]$ mkdir temp
[ec2-user@ip-10-0-24-185 ~]$ ls
temp
[ec2-user@ip-10-0-24-185 ~]$ cd temp
```

After that, in another powershell tab, we need to copy the Application files that contains the php files into the temp directory inside the EC2 Instance. To do this, we can use “scp -i WeddingCatering.pem user_reg.zip ec2-user@IP-address:temp”.

```
PS C:\Users\Asyer\OneDrive\Desktop\MMU\DEGREE\Semester 5\Database and Cloud Security\Assignment\Ass 2>
 scp -i WeddingCatering.pem user_reg.zip ec2-user@ec2-54-159-133-108.compute-1.amazonaws.com:temp
user_reg.zip                                                  100%   10KB  36.4KB/s   00:00
PS C:\Users\Asyer\OneDrive\Desktop\MMU\DEGREE\Semester 5\Database and Cloud Security\Assignment\Ass 2>
```

Now, we can see in the temp directory that we have the application zip file. From here, we can unzip the file using “unzip user_reg”

```
[ec2-user@ip-10-0-24-185 temp]$ ls
user_reg.zip
[ec2-user@ip-10-0-24-185 temp]$ unzip user_reg
Archive:  user_reg.zip
  inflating: user_reg/add_staff.php
  inflating: user_reg/catering_staff.php
  inflating: user_reg/database.php
  inflating: user_reg/delete_staff.php
  inflating: user_reg/index.php
  inflating: user_reg/login.php
  inflating: user_reg/logout.php
  inflating: user_reg/style.css
  inflating: user_reg/update_staff.php
  inflating: user_reg/user_registration.sql
  inflating: user_reg/wedding_staff.php
[ec2-user@ip-10-0-24-185 temp]$ ls
user_reg  user_reg.zip
```

After unzipping the file, we need to create another directory called /var/www/html and also move all the application files into this directory using the commend “sudo mv * ?var/www/html”.

```
[ec2-user@ip-10-0-24-185 user_reg]$ sudo mkdir -p /var/www/html
```

```
[ec2-user@ip-10-0-24-185 user_reg]$ sudo mv * /var/www/html
```

```
[ec2-user@ip-10-0-24-185 user_reg]$ cd /var/www/html
[ec2-user@ip-10-0-24-185 html]$ ls
add_staff.php       database.php      index.php  logout.php  update_staff.php       wedding_staff.php
catering_staff.php  delete_staff.php  login.php  style.css   user_registration.sql
[ec2-user@ip-10-0-24-185 html]$
```

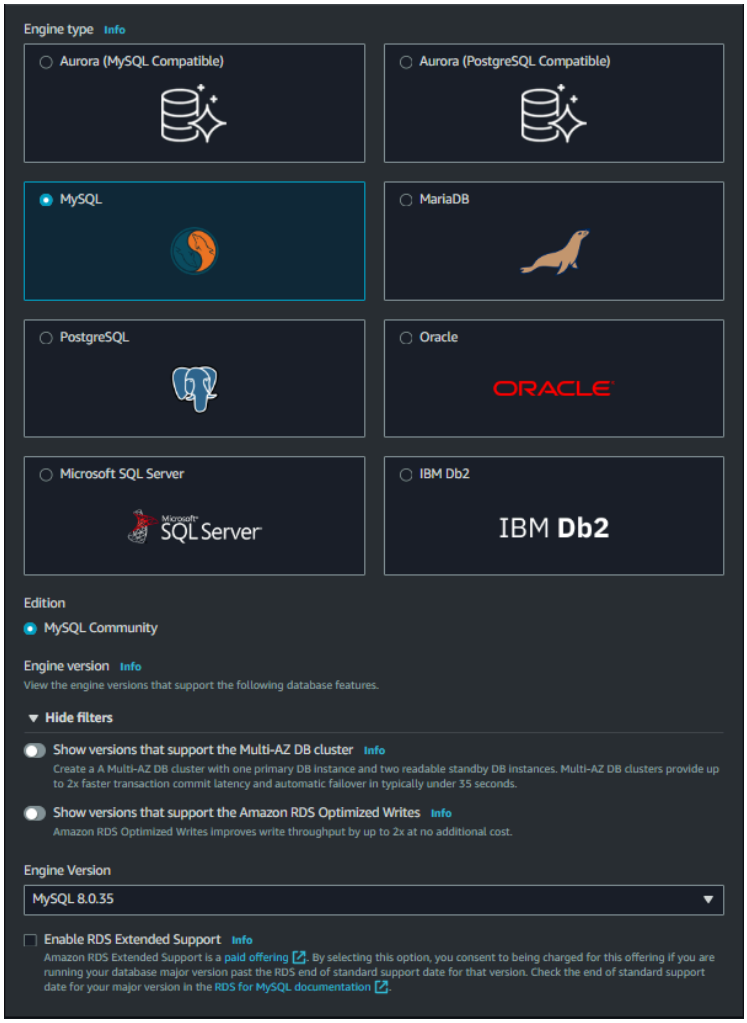
Now that all the application files are in the EC2 Instance, we can proceed on installing Apache by using the command “yum install -y httpd”. Make sure to be in the root user by running the “sudo su -”.

```
[ec2-user@ip-10-0-24-185 html]$ sudo su -
Last login: Thu Jul  4 22:40:11 UTC 2024 on pts/2
[root@ip-10-0-24-185 ~]# yum update -y
Last metadata expiration check: 0:16:07 ago on Thu Jul  4 22:24:34 2024.
Dependencies resolved.
Nothing to do.
Complete!
[root@ip-10-0-24-185 ~]# yum install -y httpd
Last metadata expiration check: 0:16:42 ago on Thu Jul  4 22:24:34 2024.
Dependencies resolved.
================================================================================================
 Package                    Architecture   Version                        Repository          Size
================================================================================================
Installing:
 httpd                      x86_64         2.4.59-2.amzn2023              amazonlinux          47 k
Installing dependencies:
 apr                        x86_64         1.7.2-2.amzn2023.0.2           amazonlinux         129 k
 apr-util                   x86_64         1.6.3-1.amzn2023.0.1           amazonlinux          98 k
 generic-logos-httpd        noarch         18.0.0-12.amzn2023.0.3         amazonlinux          19 k
 httpd-core                 x86_64         2.4.59-2.amzn2023              amazonlinux         1.4 M
```

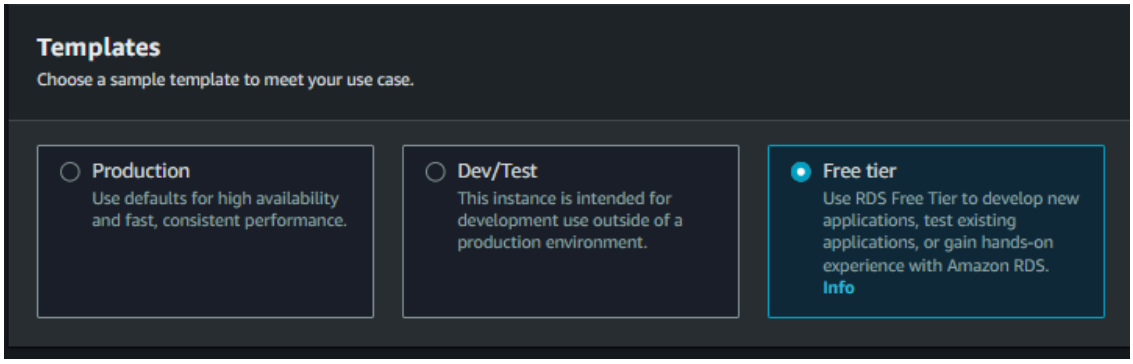
After install, we can proceed to check start the web server by running the “systemctl enable httpd” and “systemctl start httpd”. We can also view the status by running “systemctl status httpd”.

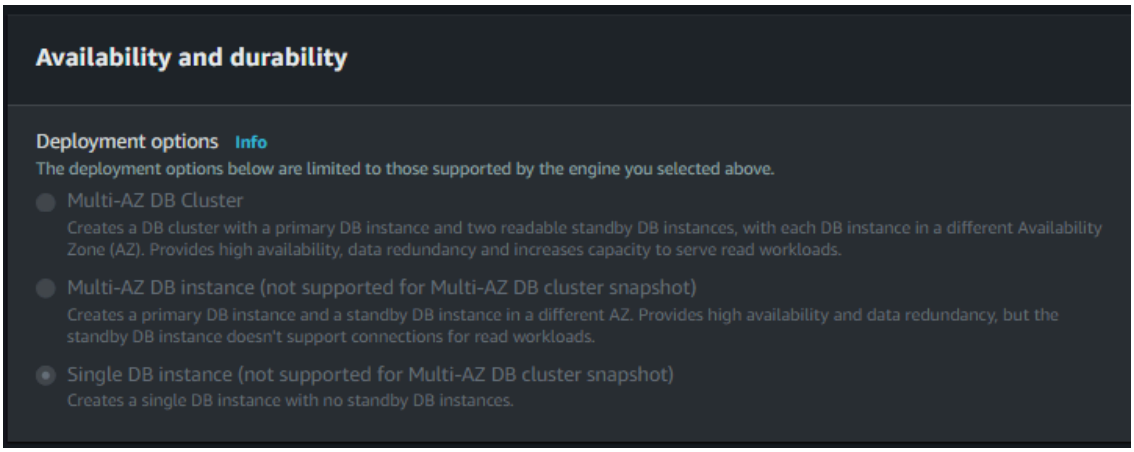
```
Complete!
[root@ip-10-0-24-185 ~]# systemctl status httpd
○ httpd.service - The Apache HTTP Server
     Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled; preset: disabled)
     Active: inactive (dead)
       Docs: man:httpd.service(8)
[root@ip-10-0-24-185 ~]# systemctl enable httpd
Created symlink /etc/systemd/system/multi-user.target.wants/httpd.service → /usr/lib/systemd/system/ht
tpd.service.
[root@ip-10-0-24-185 ~]# systemctl start httpd
[root@ip-10-0-24-185 ~]# systemctl status httpd
● httpd.service - The Apache HTTP Server
     Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; preset: disabled)
     Active: active (running) since Thu 2024-07-04 22:43:55 UTC; 6s ago
       Docs: man:httpd.service(8)
   Main PID: 26226 (httpd)
     Status: "Started, listening on: port 80"
      Tasks: 177 (limit: 1114)
     Memory: 12.9M
        CPU: 66ms
     CGroup: /system.slice/httpd.service
             ├─26226 /usr/sbin/httpd -DFOREGROUND
             ├─26227 /usr/sbin/httpd -DFOREGROUND
             ├─26228 /usr/sbin/httpd -DFOREGROUND
             ├─26229 /usr/sbin/httpd -DFOREGROUND
             └─26230 /usr/sbin/httpd -DFOREGROUND
```

- **RDS**
    1. **Create RDS Instance** - We will be using MySQL as the engine type, since the application also used MySQL database.

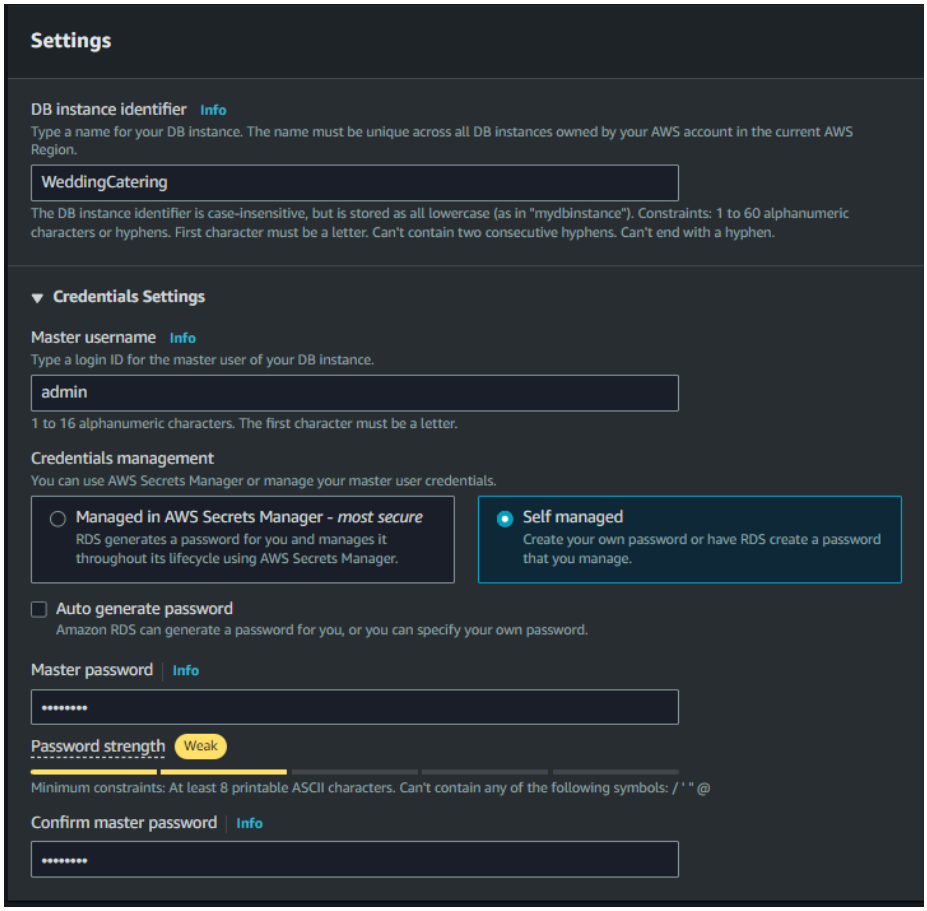


For the templates, we use the Free tier for easier cost management. This will also automatically provide the Single DB Instance for the availability and durability.

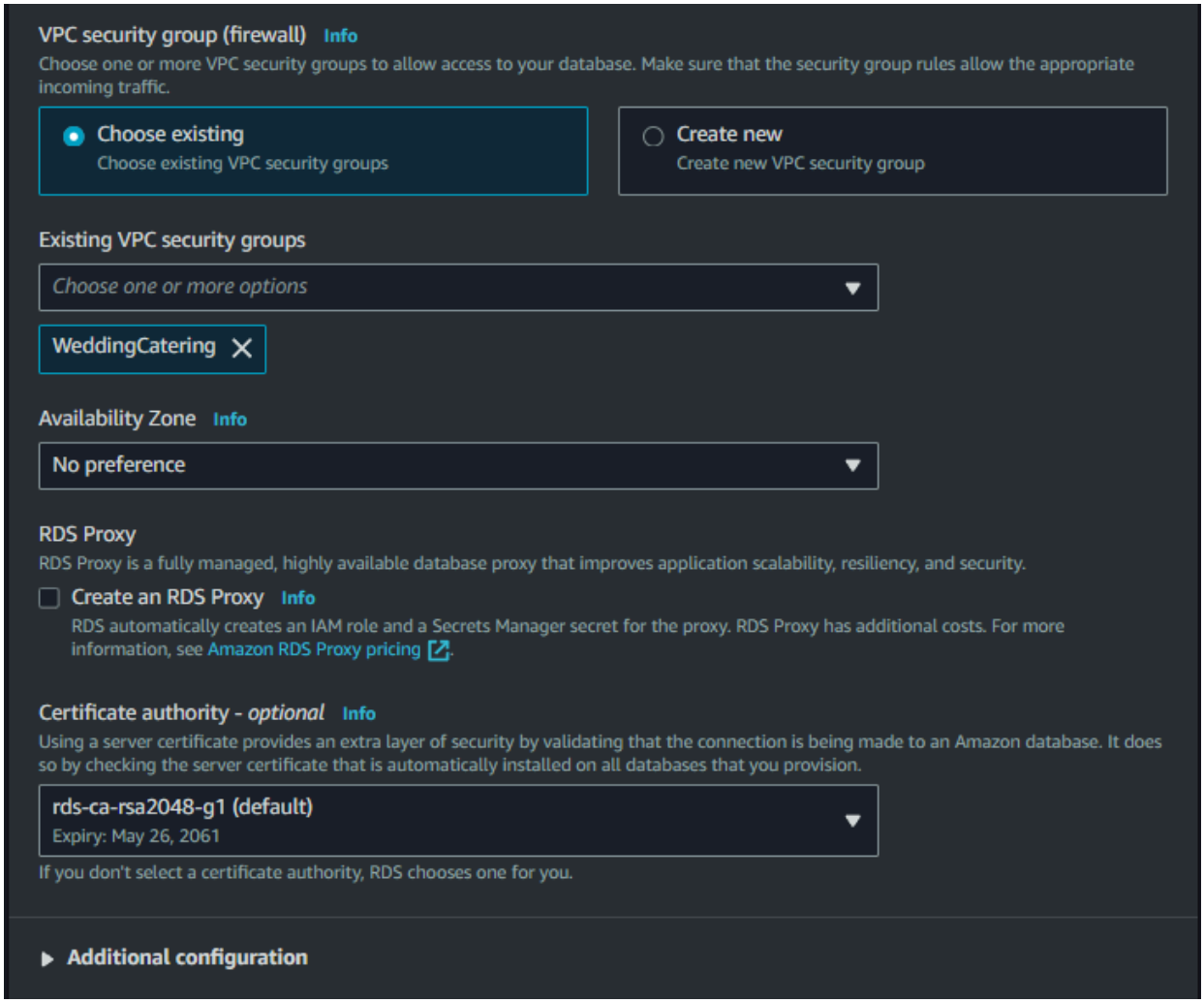


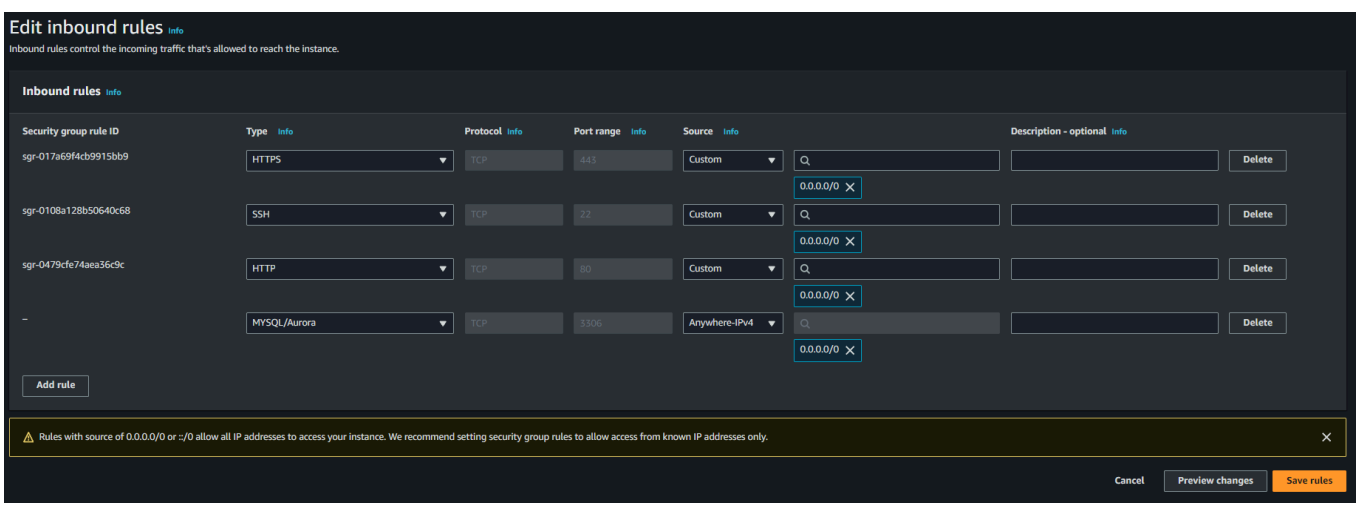



As for the settings, we use WeddingCatering as the DB Instance Identifier for easier management. For the credential settings, we assign a master username and master password called “admin” and “admin123” to make the database more secure.



2. **Configure Security Group** - We choose the existing VPC as the security group for easier management and also add a new rule for the Inbound Rules to allow MySQL to be accessed by the IP address.



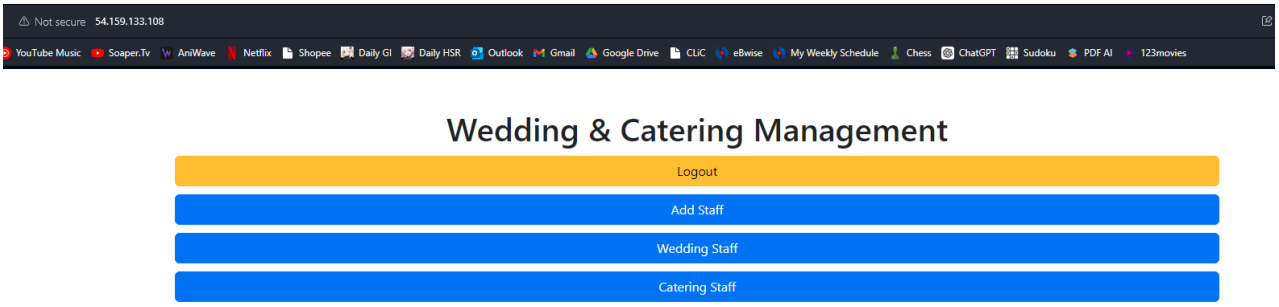
3. **Database Endpoint** - The database endpoint is used to connect the application to the database; “weddingcatering.cpffhyoagrni.us-east-1.rds.amazonaws.com”

- **Connecting EC2 and RDS**
  1. **Update Database Configuration** - To ensure the application can connect the RDS instance, we need to modify the “database.php” file to use the RDS endpoint, master username and master password. We can edit the php file by using “sudo nano database.php” and change the details needed.

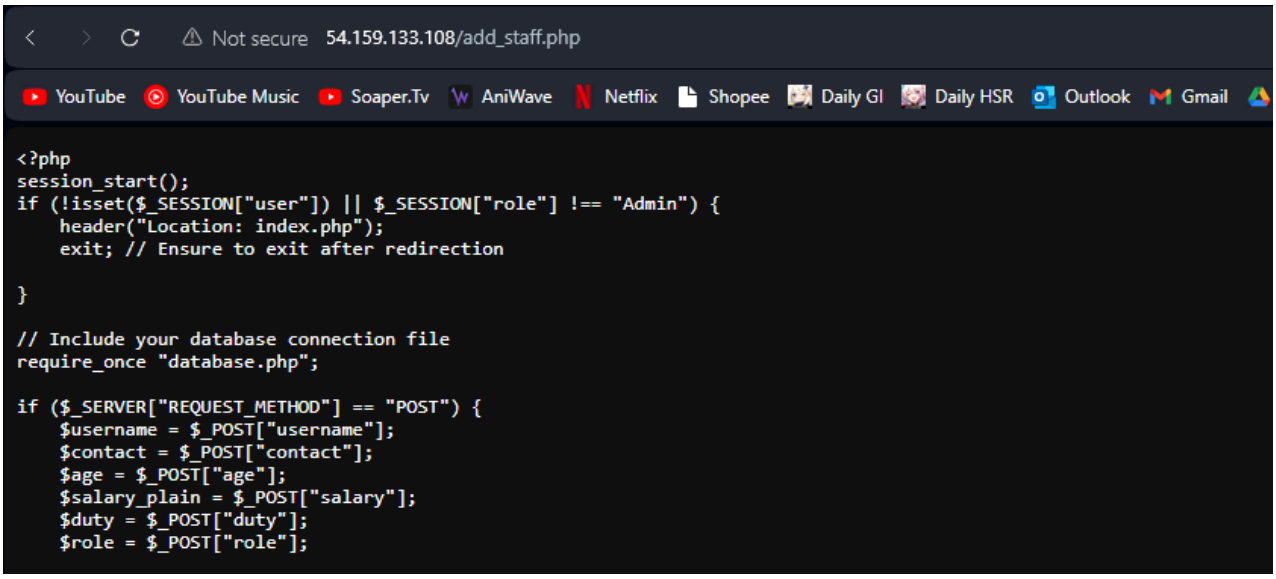
```
[ec2-user@ip-10-0-24-185 html]$ ls
add_staff.php       database.php      index.php  logout.php  update_staff.php       wedding_staff.php
catering_staff.php  delete_staff.php  login.php  style.css   user_registration.sql
[ec2-user@ip-10-0-24-185 html]$ sudo nano database.php
```

```
  GNU nano 5.8                                  database.php
<?php

$hostName = "weddingcatering.cpffhyoagrni.us-east-1.rds.amazonaws.com";
$dbUser = "admin";
$dbPassword = "admin123";
$dbName = "user_registration";
$conn = mysqli_connect($hostName, $dbUser, $dbPassword, $dbName);
if (!$conn) {
    die("Something went wrong;");
}

?>
```

- **Testing**
  1. **HTTP (Slightly work)** - We test the web application by entering "http://ip-address" in the browser. We manage to enter application.




However, when we clicked the functions which will bring us to another php file, it broke and just shows the code inside the php file.



2. **HTTPS (Error)** - We also tested using “https://ip-address” but that entirely didn’t let us enter the web application.