## AWS Pre-Configuration

Since the deployment is based on AWS service, we need to pre-configure some basic role for a user. After we registered a new account on AWS, this account is a root user by default, which is used for billing and management. So that we need to assign a new IAM user with permission and access key to use resources. As the Figure 3.1 shown below is the creation of new IAM user.

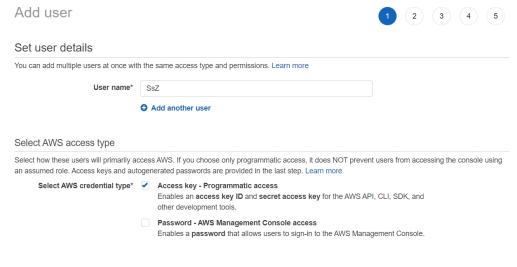


Figure 3.1 Creation of New IAM User

At the last, we will create a new user with permission to the resources that we need. The purpose to use a new user is to control the access to the AWS resources and protect the resources. The Figure 3.2 below shows the access permission for this user, which is permitted to use EC2, S3 and CloudFormation services.

Graphical user interface, text, application, email

Description automatically generated

Figure 3.2 The Access Permission for New User

## YAML Script Design

After configured the permission for user, we can write the YAML deploy script for CloudFormation to set up the virtual server and the storage bucket. As the Figure 3.3 shown below, this template is used to create a EC2 instance for server and a S3 bucket for storage.

Text

Description automatically generated

Figure 3.3 YAML Template Description

### Creation of EC2 Instance

As the Figure 3.4 below shows the YAML script portion of creation of EC2 instance. Meanwhile, it will create a security group for the instance to control the access port. As the Figure 3.5 shows, this EC2 instance can be accessed only by SSH on port 22.

Text

Description automatically generated

Figure 3.4 Creation of EC2 Instance

Text

Description automatically generated

Figure 3.5 The security Group for EC2 Instance

In addition, we also need to define the storage, memory, CPU, and networking capacity in the script. As Figure 3.6 below shows the instance type is EC2 and the default computing type is t2.small, which means the server provides 1 CPU, 2.5 GHz and 2 GB of memory. In this case, the computing capacity is enough for this project.

Text

Description automatically generated

Figure 3.6 Defining Instance Type

### Creation of S3 Storage Bucket

In this script, we also create a S3 storage bucket to store all the application files we need for the website. The script block in Figure 3.7 defines a S3 bucket, and to allow external access to the bucket, we set the bucket to public readable. Therefore, we can extract, or upload files not only directly use S3 service, but can also use EC2 backstage to control the resource files.

Text

Description automatically generated

Figure 3.7 Defining S3 Storage Bucket

In addition, to distinguish the S3 buckets, we need to give a unique default name for this particular bucket. As the Figure 3.8 blow shows the default name of the S3 bucket is “bucketforbooking”.

Text

Description automatically generated

Figure 3.8 Defining S3 Bucket Default Name

## Python Website Design

For this website, the main function is to provide gym booking function and appointments management function for users. To create this website, we are using Django framework, which is a high-level Python web framework that enables rapid website development with security and maintainability. In addition, we are using SQLite database to store the appointment information. The appointment page shown in Figure 3.9 corresponds to the following code in Figure 3.10.

Graphical user interface, application

Description automatically generated

Figure 3.9 Gym Booking Page

Text

Description automatically generated

Figure 3.10 Booking Function in views.py

The following Figure 3.11 and Figure 3.12 shows the process of making an appointment. After user enter the personal information in a correct format, the booking information will be stored in the database, and then the page will jump to a new appointment management page. Before deployed on AWS, the website is running locally.

Graphical user interface, text, application

Description automatically generated

Figure 3.11 Booking Page

Graphical user interface

Description automatically generated

Figure 3.12 Appointment Management Page

## Summary

This chapter presents the methodology of this project including the pre-configuration of AWS, the design of YAML script and the design of Python website. In the next chapter, the implementation of this project will be given based on the preparation we have already done.

# implementation, outcome and discussion

In this chapter, we will continue the content of the previous chapter, and implement the project based on what we have configured and present the process of implementation and the result.

## Auto Deploy AWS Services

As last chapter mentioned, we already created a YAML deploy script for all the services we need. The next step is to deploy this script on CloudFormation. The following Figure 4.1 shows the deployment status detail of CloudFormation, and it shows that a S3 bucket, security group and a EC2 instance have created successfully.

Graphical user interface

Description automatically generated with medium confidence

Figure 4.1 CloudFormation Deployment Status

## Upload Files to S3 Bucket

To be able to access the bucket through EC2, we need to grand public access to the files, the following Figure 4.2 shows that this bucket is public accessible, and the files are uploaded successfully.

Graphical user interface, text, application, email

Description automatically generated

Figure 4.2 Uploading Website Files to S3 Bucket

## Connect to EC2 and Deploy the Website

Continue the previous step, the generated EC2 instance will contain a public IPv4 address and DNS of the address, as the Figure 4.3 shown. Therefore, we use PuTTy to connect to the instance using the SSH key pair.

Graphical user interface, text, application, email

Description automatically generated

Figure 4.3 EC2 Instance Detail

After we connected to the instance, we can now extract the files we uploaded on S3 bucket before. The following Figure 4.4 shows that we get the zip file of the website and unzip the archive successfully.

Text

Description automatically generated

Figure 4.4 Get Website Files

Before we run the website, we need to determine which port to use, in normal case, we choose port 8000. therefore, we need to edit the security group to allow the traffic through the port 8000. The Figure 4.5 shows the operation of adding a new rule for port 8000.

Graphical user interface, application

Description automatically generated

Figure 4.5 Adding New Inbound Rule

In the final step, we run the “manage.py” at port 8000,the following Figure 4.6 and Figure 4.7 confirm that the website deployed successfully.

Graphical user interface, website

Description automatically generated

Figure 4.6 Running the Website on Port 8000

A picture containing text, monitor, screen, screenshot

Description automatically generated

Figure 4.7 Verification of Website