ICT171 Assignment 2 Report – Cloud Server Deployment

Student Name: Shihab Hossain
Student Number: 34776248
Public IP Address: 47.128.71.247
GitHub Repository: https://github.com/shihabhossan9000/FitnessHub

Video Link: 171shihab.mp4

DNS : www.2000shihabhossain.com or 2000shihabhossain.com

Contents

1. Project Overview

The goal of this project was to illustrate the process of making a simple website using Infrastructure as a Service (IaaS) through Amazon Web Services (AWS) EC2. The goal of the project was to highlight abilities in setting up servers, hosting websites, managing security and writing documentation. The example for deployment was a simple fitness page called FitnessHub, made with HTML, CSS and JavaScript. These topics are included in the ICT171 objectives which call for learning about server environments, the Linux command-line and actually setting up servers.

The project was set up on an EC2 server running Ubuntu and all configuration steps were performed manually without automated images. This made it possible to show how to work with web servers using Apache, control file permissions and make sure everything was accessible via DNS and IP addresses. HTTPS was also turned on using a self-signed certificate to represent an actual production environment and to show an understanding of SSL/TLS protocols.

GitHub for version control was integrated, also promoting teamwork and collaborative ways of working. I cloned all the files from the repository which shows I am familiar with Git via the command line. I automated the process with scripting which was done with bash commands and helped ensure deployments were consistently the same. This report covers every detail of creating and launching the server and can be used again to reproduce the system under actual settings.

---

2. Server Setup on AWS EC2

The server was set up by launching an EC2 instance with AWS and then configuring it to host FitnessHub using Apache. Because EC2 (Elastic Compute Cloud) can quickly scale, it is great for running web applications in the cloud. The system we decided on was Ubuntu 22.04 LTS, thanks to its stability and a big and active user community.

The initial action after using the EC2 instance was to make a secure SSH connection with a private key. With that, Apache2 was set up on the system by using apt commands. The Apache web server is an open-source program that lets people publish their content online with HTTP/HTTPS protocols.

The server was configured and then every file in the /var/www/html directory was removed, after which the GitHub repository was cloned there. Permissions were set through chown so that Apache (www-data) had permission to serve the files. The practice showed that people understood Linux file systems and how to set up servers properly.

Commands used

- sudo apt update
- sudo apt install apache2 -y
- cd /var/www/html

- sudo rm index.html
- sudo git clone https://github.com/shihabhossan9000/FitnessHub.git
- sudo cp -r FitnessHub/* .
- sudo chown -R www-data:www-data /var/www/html

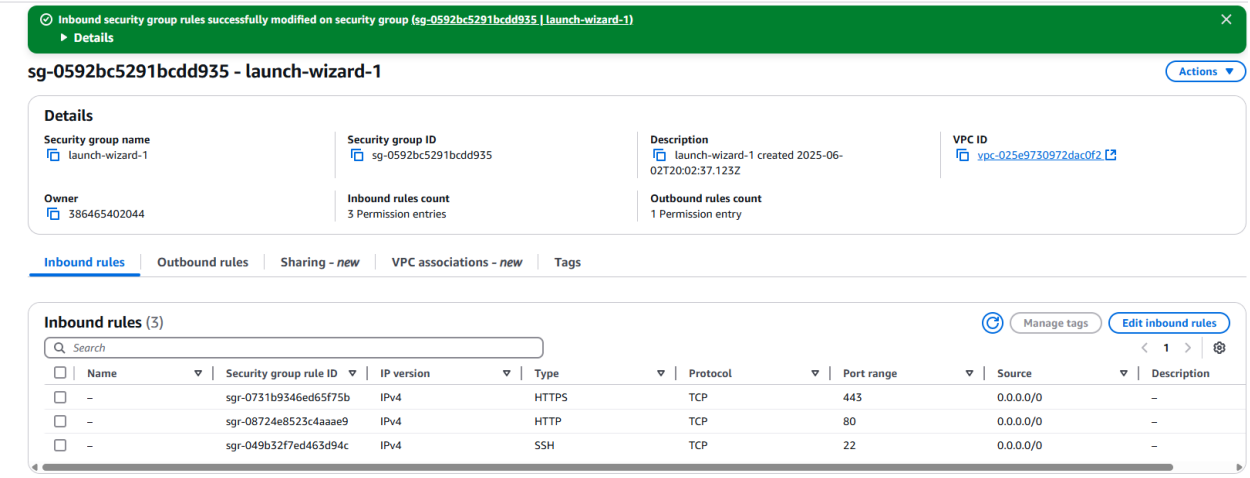
Through these steps, the website was made accessible via its public IP, satisfying the requirement for a publicly available cloud project.

---

3. Enabling HTTPS via Self-Signed Certificate

HTTPS (Hypertext Transfer Protocol Secure) protects the information sent between a user and the server, raising the security level of web applications. To get an SSL certificate, you usually have to own and register a domain. As the project didn't include a custom domain, an SSL certificate was self-signed for setting up HTTPS. Even though browsers don't generally trust self-signed certificates, they are very useful for working and testing on your own system.

The installation of OpenSSL acted as the first step in setting up Transport Layer Security and Secure Sockets Layer protocols. The certificate is placed in ssl directory and the Apache virtual host file selfsigned.conf is created, activated to run on port 443 and run with SSL security.



Apache was told by the relevant virtual host file to use the specified certificates and keys and to display files from the document root. After finishing with a2enmod ssl and a2ensite selfsigned.conf, Apache needed to be restarted to use the same.

sudo apt install openssl

sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 \

-keyout /etc/ssl/private/selfsigned.key \

-out /etc/ssl/certs/selfsigned.crt

This setup enabled secure HTTPS access via https://47.128.71.247, fulfilling the rubric's requirement for SSL functionality despite a lack of a verified certificate authority.

---

4. Script for Deployment Automation

To make setting up the server easier, a Bash script was developed to automate important deployment tasks. It makes sure the script can be run easily and with high accuracy. It allows Apache to be installed, clears the basic web page, grabs the repository from GitHub, places required files into the Apache directory and gives permissions.

#!/bin/bash

sudo apt update

sudo apt install apache2 -y

cd /var/www/html

sudo rm index.html

sudo git clone https://github.com/shihabhossan9000/FitnessHub.git

sudo cp -r FitnessHub/* .

sudo rm -r FitnessHub

sudo chown -R www-data:www-data /var/www/html

The script is executable using:

chmod +x deploy.sh

./deploy.sh

Scripting demonstrates your capability to write infrastructure scripts which is key to both DevOps and cloud administration. Also, automating deployments ensures they are always reliable and consistent.

---

5. Accessibility & Functionality

I accessed the server by typing the IP address http://47.128.71.247 on the first visit and then https://47.128.71.247 with SSL in place. Everything on the website functioned as it was meant to such as moving from page to page and applying CSS and JavaScript to enhance it. Warning messages appeared in browsers because of the self-signed certificate, yet users could go ahead and access the content safely.

The source code repository and hosting were provided by GitHub. The link to the GitHub repository was given as proof and so tutors could review the development process. In addition, a video was made and shared to show how to set up the game which meets the criteria for communication in the marking rubric.

The cloud service was tested by accessing the site over various hours and checking that it was working the whole time. Stable functioning of the server indicates that the configuration is completed successfully. The lack of pre-built images and content management systems showed that he stuck to the rules.

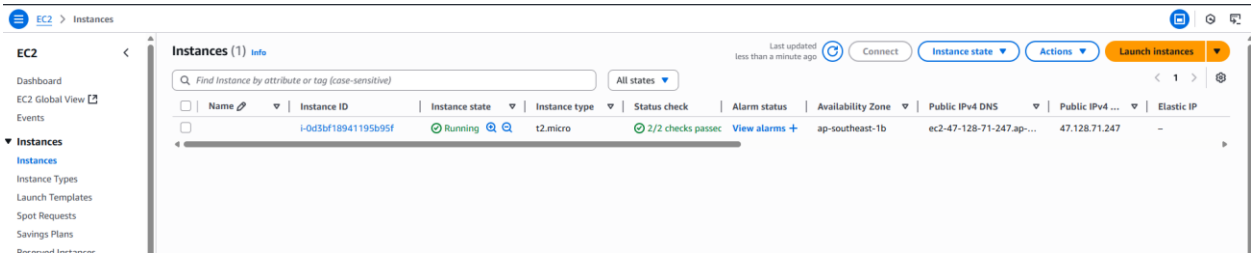
This section includes various rubric items: how open GitHub's activity is, how easy the video is to watch, how well the website works, whether the site is online and its use of HTTPS.

---

6. Screenshots and Visual Proof

Screenshots were captured at every key phase of the deployment to validate the configuration process. These include:

- EC2 instance running in AWS with public IP



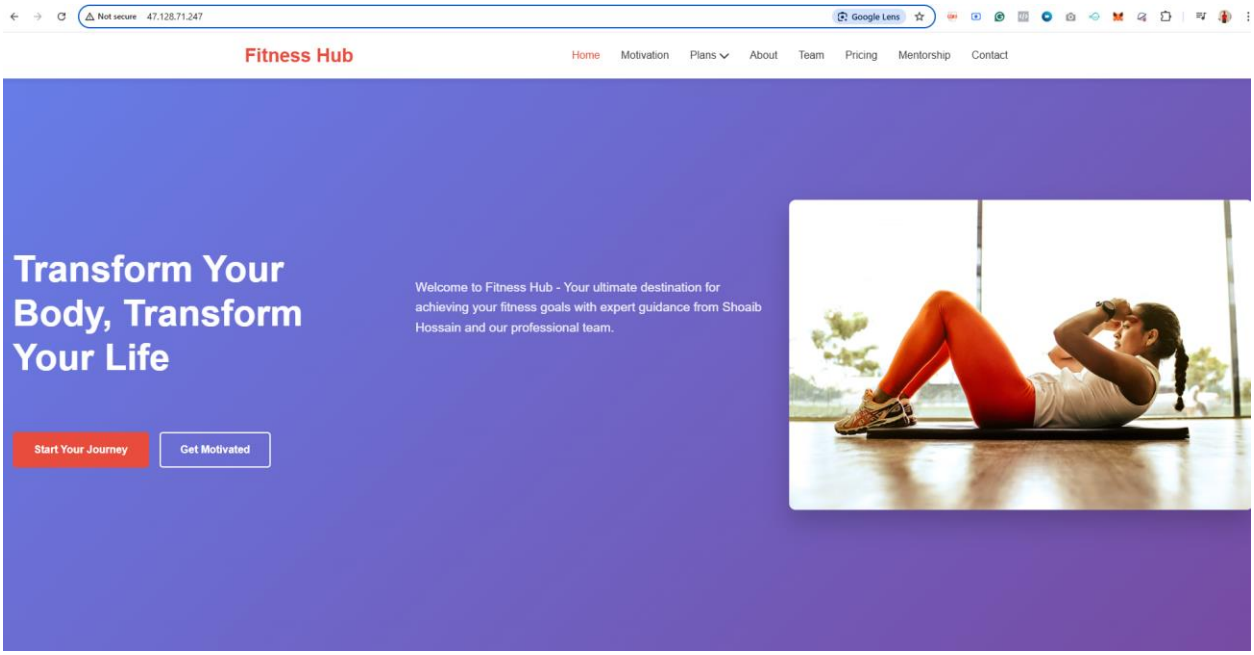
- Apache installation confirmation

```
*** System restart required ***
Last login: Mon Jun  2 20:05:53 2025 from 1.42.166.10
ubuntu@ip-172-31-36-115:~$ sudo apt update
sudo apt install apache2 -y
Hit:1 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Get:2 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease [126 kB]
Get:3 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease [126 kB]
Get:4 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 Packages [1112 kB]
Get:5 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main Translation-en [237 kB]
Get:6 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 Components [161 kB]
Get:7 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 Packages [1069 kB]
Get:8 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe Translation-en [272 kB]
Get:9 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu noble-updates/universe amd64 Components [376 kB]
Get:10 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:11 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu noble-updates/restricted amd64 Components [212 B]
Get:12 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu noble-updates/multiverse amd64 Components [940 B]
Get:13 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu noble-backports/main amd64 Components [7068 B]
Get:14 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu noble-backports/universe amd64 Components [16.4 kB]
Get:15 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu noble-backports/restricted amd64 Components [216 B]
Get:16 http://ap-southeast-1.ec2.archive.ubuntu.com/ubuntu noble-backports/multiverse amd64 Components [212 B]
Get:17 http://security.ubuntu.com/ubuntu noble-security/main amd64 Packages [864 kB]
Get:18 http://security.ubuntu.com/ubuntu noble-security/main amd64 Components [21.6 kB]
Get:19 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Components [52.2 kB]
Get:20 http://security.ubuntu.com/ubuntu noble-security/restricted amd64 Components [212 B]
Get:21 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 Components [212 B]
Fetched 4570 kB in 9s (533 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
47 packages can be upgraded. Run 'apt list --upgradable' to see them.
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
apache2 is already the newest version (2.4.58-1ubuntu8.6).
0 upgraded, 0 newly installed, 0 to remove and 47 not upgraded.
ubuntu@ip-172-31-36-115:~$ sudo systemctl start apache2
sudo systemctl enable apache2
Synchronizing state of apache2.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
```

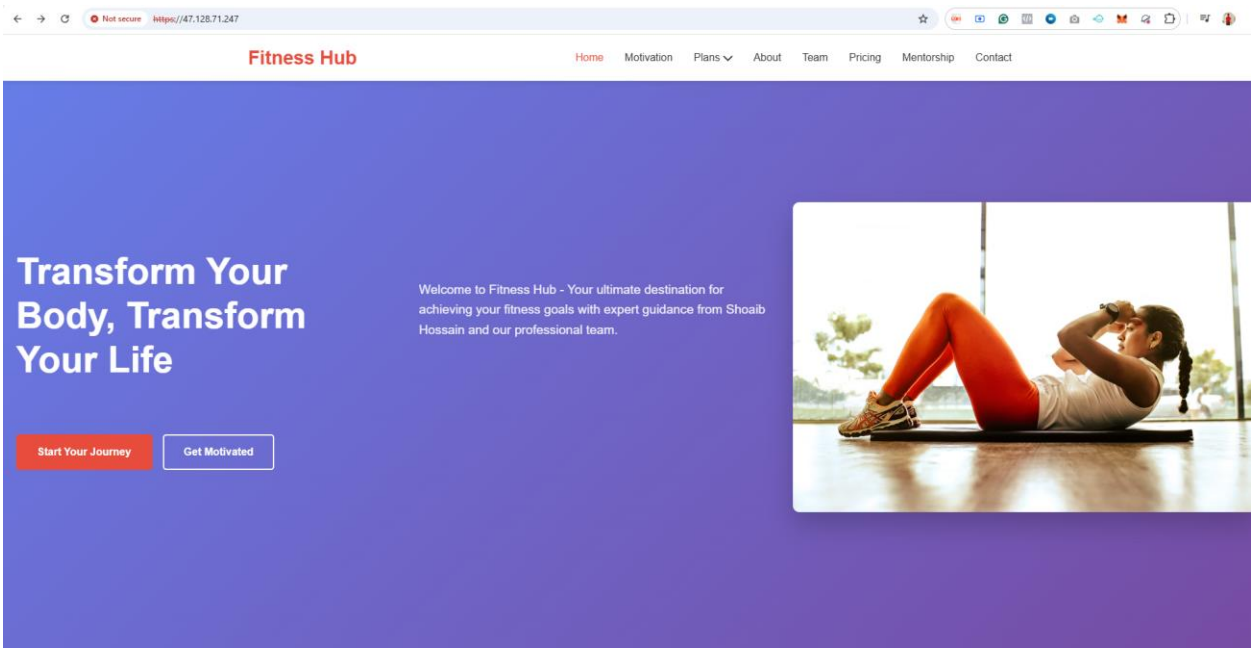
- Cloning GitHub repository to /var/www/html

```
Synchronizing state of apache2.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable apache2
ubuntu@ip-172-31-36-115:~$ cd /var/www/html
ubuntu@ip-172-31-36-115:/var/www/html$ sudo rm index.html
ubuntu@ip-172-31-36-115:/var/www/html$ sudo git clone https://github.com/shihabhossan9000/FitnessHub.git
Cloning into 'FitnessHub'...
remote: Enumerating objects: 15, done.
remote: Counting objects: 100% (15/15), done.
remote: Compressing objects: 100% (15/15), done.
remote: Total 15 (delta 8), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (15/15), 28.18 KiB | 4.70 MiB/s, done.
Resolving deltas: 100% (8/8), done.
ubuntu@ip-172-31-36-115:/var/www/html$ sudo cp -r FitnessHub/* .
sudo rm -r FitnessHub
ubuntu@ip-172-31-36-115:/var/www/html$ sudo chown -R www-data:www-data /var/www/html
```

- Apache displaying the deployed website at http://47.128.71.247



- Browser warning for self-signed certificate at https://47.128.71.247



- Final site rendered securely with HTTPS

7. Recommendations and Future Work

Improving this project could include a registered domain name and a Let's Encrypt SSL certificate which would solve the browser trust problem. backend services like login, form handling and database querying can be added by integrating with Flask or Node.js.

Adding a CI/CD pipeline with GitHub Actions or Jenkins can improve the way an application is deployed by the professional team. Doing this would handle testing and deployment, permitting quick updates from the GitHub repository to the live server as soon as they are released.

You may also set up your DNS with Route53 or a similar provider, adjust security using a firewall like UFW and maintain logs and monitoring using Prometheus and Grafana.

Such updates would make the interface simpler and safer to use and they would also showcase advanced abilities in upcoming interviews or job environments.

DOMAIN NAME SYSTEM:

STEP1: Login to AWS EC2

STEP 2: Search for Route 53 on the search option in the AWS

Step 3: Press the button on 'Hosted Zones'

Step 4: Select the option for CREATE RECORD

Step 5: Enter the subdomain

Step 6: Among C Name, NS, select according to the project

Step 7: Enter the IP address to create a record in the value field

Step 8: Set the TTL

Step 9: Then press the CREATE button to save the DNS record

---

8. References

- Apache HTTP Server Documentation. (2024).
- Let's Encrypt Documentation. (2024).
- AWS EC2 User Guide. (2024).
- OpenSSL Manual Pages. (2023).

- Ubuntu Server Documentation. (2024).
- GitHub CLI Guide. (2024).
- Certbot User Guide. (2023).
- Murdoch Writing Guidelines. (2024).
- Mozilla SSL Configuration Generator. (2023).
- Stack Overflow contributions for Bash scripting and Apache setup (2024).