ICT171 Assignment 2 – Cloud Server Project & Video Explainer

NAME: TABITHA MUKUYU

STUDENT ID: 35402328

CAMPUS: MURDOCH UNIVERSITY SOUTH STREET CAMPUS

DATE:30/05/2025

Access Details

Public IP Address: 54.173.233.107
Domain Name: https://tabbyyy.click
GitHub Repository: https://github.com/Mukuyu1/PetView-Showcase
Video Explainer:

Project Overview

This project involved designing and deploying a secure, cloud-based web server hosted on AWS EC2, accessible via the custom domain **https://tabbyyy.click**. The server was configured manually using the Infrastructure as a Service (IaaS) model, leveraging Ubuntu 24.04, Apache2, and Let's Encrypt SSL via Certbot.

The purpose of this project was to demonstrate real-world server deployment and documentation skills while showcasing the ability to automate key tasks using a Bash script. The server hosts a custom HTML/CSS website that could serve as a personal portfolio, club homepage, or small business landing page.

This project also aimed to develop technical proficiency in using the Linux command-line environment, Git and GitHub for version control, and AWS for cloud resource provisioning.

The use of IaaS ensured that the server setup required manual installation, configuration, and maintenance of all necessary components, thereby giving full control and visibility over the infrastructure. Bash scripting was used to automate repetitive installation tasks, improving efficiency and reproducibility.

Comprehensive documentation—including setup steps, code, and configuration details—ensures the server can be fully rebuilt by another ICT171 student without requiring external research. This reinforces the importance of professional documentation skills in IT roles.

Learning Outcomes Demonstrated

This project enabled me to meet the key learning outcomes for ICT171 by engaging in the end-to-end deployment and documentation of a secure cloud server. The following outcomes were demonstrated:

🔹 Linux Command Line Proficiency

I used the Linux terminal extensively to update packages, install Apache, configure services, and navigate the file system within a remote Ubuntu server. SSH access was used to interact directly with the cloud-based instance using `.pem` key authentication.

🔹 Hosting and Configuring a Linux Server

I successfully configured a live web server using Apache2 on Ubuntu 24.04, deployed a static HTML/CSS website, and ensured it was publicly accessible through a browser over both HTTP and HTTPS.

🔹 Bash Scripting

A Bash script (`setup.sh`) was written to automate the installation and startup of the Apache web server. This showcased my ability to write executable, reusable scripts for system administration tasks.

🔹 Version Control with GitHub

I created a public GitHub repository to track and share my project files, including the website, Bash script, and documentation. Git was used to initialize the repo, commit changes, and push updates using best practices for version control.

🔹 IaaS Implementation via AWS EC2

The project was deployed on an AWS EC2 instance running Ubuntu, representing a practical application of Infrastructure as a Service. I manually configured the server, demonstrating a deep understanding of cloud-hosted server provisioning.

🔹 SSL/TLS Configuration (HTTPS)

Security was implemented using Let’s Encrypt’s Certbot to issue an SSL certificate. The website was configured to automatically redirect all HTTP traffic to HTTPS, providing encrypted, secure communication for users.

🔹 Professional Technical Documentation

This report includes clear, detailed, and reproducible documentation of all steps required to rebuild the server. Commands are formatted for clarity, and the structure of the report ensures that another student or IT staff member could replicate the setup independently.

Step-by-Step Server Setup

This section outlines the precise steps and commands used to provision and configure the cloud-based server on AWS, utilizing Infrastructure as a Service (IaaS) principles.

🔸 EC2 Setup (Infrastructure as a Service - IaaS)

The foundation of this project is an Amazon Elastic Compute Cloud (EC2) instance, chosen for its flexibility and control, embodying the IaaS model.

Launching Ubuntu EC2 on AWS:

1. **Login to AWS Management Console:** Navigated to https://console.aws.amazon.com/.
2. **Navigate to EC2 Dashboard:** From the services menu, selected **EC2**.
3. **Launch Instance Wizard:** Clicked on "Instances" in the left navigation pane, then "Launch Instances."
4. **Choose Amazon Machine Image (AMI):** Selected **Ubuntu Server 24.04 LTS (HVM), SSD Volume Type** (64-bit x86) as the base operating system. This provides a stable and widely supported Linux environment.
5. **Choose Instance Type:** Selected **t2.micro** instance type, which is eligible under the AWS Free Tier, sufficient for the assignment's requirements.
6. **Create Key Pair:** A new key pair named `pet.pem` was created using the **RSA** algorithm and `.pem` file format. This private key file was downloaded and is crucial for secure SSH access to the instance.
    - *Note:* The SHA256 fingerprint of the private key is `06 CC 68 BC 28 CB 65 45 1C C8 A6 76 34 4B 86 5B 7D 6C AE 76 1F 8F F3 53 FC E2 D1 8E 6B 2D 2F 98`.

Security Group Setup (Ports 22, 80, 443):

A new security group was configured to control inbound and outbound traffic to the EC2 instance, acting as a virtual firewall. The following inbound rules were added:

- **SSH (Port 22):**
    - **Type:** SSH
    - **Protocol:** TCP
    - **Port range:** 22
    - **Source:** My IP (or 0.0.0.0/0 for broad access, though My IP is more secure)
    - *Purpose:* Allows secure shell access to the instance for administration and command execution.
- **HTTP (Port 80):**
    - **Type:** HTTP
    - **Protocol:** TCP
    - **Port range:** 80
    - **Source:** 0.0.0.0/0
    - *Purpose:* Enables public access to the web server over standard unencrypted HTTP.
- **HTTPS (Port 443):**
    - **Type:** HTTPS
    - **Protocol:** TCP
    - **Port range:** 443

- **Source:** 0.0.0.0/0
- *Purpose:* Enables public access to the web server over secure, encrypted HTTPS (required for SSL/TLS).

SSH Access with `.pem` File:

After the instance launched and transitioned to the "running" state, SSH access was established from a local terminal using the downloaded `pet.pem` key file and the instance's Public IPv4 address (`54.173.233.107`).

1. **Set Key File Permissions:**

chmod 400 /home/Tabby/Downloads/pet.pem

This command ensures that only the owner has read permissions for the private key, which is a security requirement for SSH.

2. **Connect to EC2 Instance:**

ssh -i /home/Tabby/Downloads/pet.pem ubuntu@54.173.233.107

The username `ubuntu` is the default for Ubuntu AMIs. Upon successful connection, the terminal prompt changed, indicating access to the server.

🔸 **Apache Web Server Installation**

With SSH access established, the Apache HTTP Server was installed and configured to host web content.

Commands Used:

1. **Update Package Lists:**

sudo apt update

This command refreshes the list of available packages and their versions from the Ubuntu repositories.

2. Install Apache2:

sudo apt install apache2 -y

This command installs the Apache web server package (`apache2`) and automatically confirms any prompts (`-y`).

3. **Start Apache Service:**

sudo systemctl start apache2

This command initiates the Apache web server service.

4. **Enable Apache to Start on Boot:**

sudo systemctl enable apache2

This command configures Apache to automatically start whenever the EC2 instance reboots, ensuring continuous service availability.

After these steps, the default Apache welcome page became accessible by navigating to `http://54.173.233.107` in a web browser. The server's web root directory is located at `/var/www/html`.

Script with Commentary

To automate the setup of the web server, a bash script named `setup.sh` was created. This script streamlines the process of installing and configuring Apache2 on the Ubuntu EC2 instance, ensuring consistency and efficiency.

```
#!/bin/bash
sudo apt update -y
sudo apt install apache2 -y
sudo systemctl start apache2
sudo systemctl enable apache2
```

Script Explanation:

- `#!/bin/bash`: This is called a **shebang**. It tells the operating system to execute this script using the `bash` interpreter, ensuring the commands are run correctly in a bash shell environment.
- `sudo apt update -y`: This command updates the local package index. It fetches the latest information about available software packages from Ubuntu's repositories. The `-y` flag automatically answers "yes" to any prompts during the update process, allowing for unattended execution.
- `sudo apt install apache2 -y`: This command installs the Apache2 web server software. `apt` is the package manager for Ubuntu, and `apache2` is the name of the Apache package. The `-y` flag again automates the confirmation for the installation.
- `sudo systemctl start apache2`: After installation, this command initiates the Apache2 web server service. This makes the server active and ready to serve web content.
- `sudo systemctl enable apache2`: This command configures the Apache2 service to automatically start every time the EC2 instance boots up. This is crucial for ensuring that the web server is always running without manual intervention after a reboot.

GitHub Version Control

For effective project management and collaboration, **Git and GitHub** were utilized to control versions of all project files. This ensures all server configurations, web content, and documentation are securely stored and changes are tracked.

A **new public repository** was created on GitHub for this project. Locally on the EC2 instance, a dedicated project directory was set up. All relevant files, including the **HTML and CSS web files** and the `setup.sh` **script**, were copied into this directory.

Git was then **initialized** within this directory. User identity was configured, and all project files were **added and committed** to the local repository with a descriptive message. Finally, the local repository was **linked to the remote GitHub repository**, and the committed changes were **pushed** to the `main` branch using a Personal Access Token for authentication.

The GitHub repository now contains all necessary project files, including the website content, the server setup script, and a `README.md` **file** that provides an overview and links to the live server and video.

References

- Amazon Web Services. (n.d.). *Amazon EC2 Documentation*. Retrieved from https://docs.aws.amazon.com/ec2/
- Apache HTTP Server Project. (n.d.). *Apache HTTP Server Documentation*. Retrieved from https://httpd.apache.org/docs/
- Certbot. (n.d.). *Certbot Documentation*. Retrieved from https://certbot.eff.org/docs/