# Placement Empowerment Program

***Cloud Computing and DevOps Centre***

Host a Static Website on a Cloud VM Install Apache on your cloud VM and host a simple HTML website.

Name: Shakthi Lakshmanan V Department: CSE



### **Introduction**

A static website is made up of pre-written HTML, CSS, and JavaScript files that are delivered directly to users without any server-side processing. Hosting a static website on a cloud-based Virtual Machine (VM) is a flexible and cost-effective solution, making it easier for individuals and businesses to make their websites accessible from anywhere. Cloud-based hosting also provides scalability and control over the website’s infrastructure.

### **Overview**

Hosting a static website on a cloud VM involves these key steps:

1. **Setting up a Cloud VM** – Create a virtual machine on a cloud provider like AWS, Azure, or GCP.
2. **Installing a Web Server** – Configure a web server (like Apache or Nginx) to serve website files.
3. **Uploading Website Files** – Place HTML, CSS, and JavaScript files in the web server’s directory.
4. **Configuring Network Access** – Ensure the web server is accessible via HTTP (port 80) from anywhere.
5. **Testing and Launching** – Verify the website’s functionality and make it publicly available.

### **Objectives**

This process helps to:

* **Learn Cloud Computing Basics** – Understand how virtual machines work in a cloud environment.
* **Gain Web Hosting Skills** – Set up and configure web servers like Apache or Nginx.
* **Deploy a Website** – Make a static website live on the internet.
* **Understand Networking** – Learn about firewall rules, security groups, and HTTP configurations.
* **Explore Cost-Effective Hosting** – Discover affordable ways to host websites without managed services.

### **Importance**

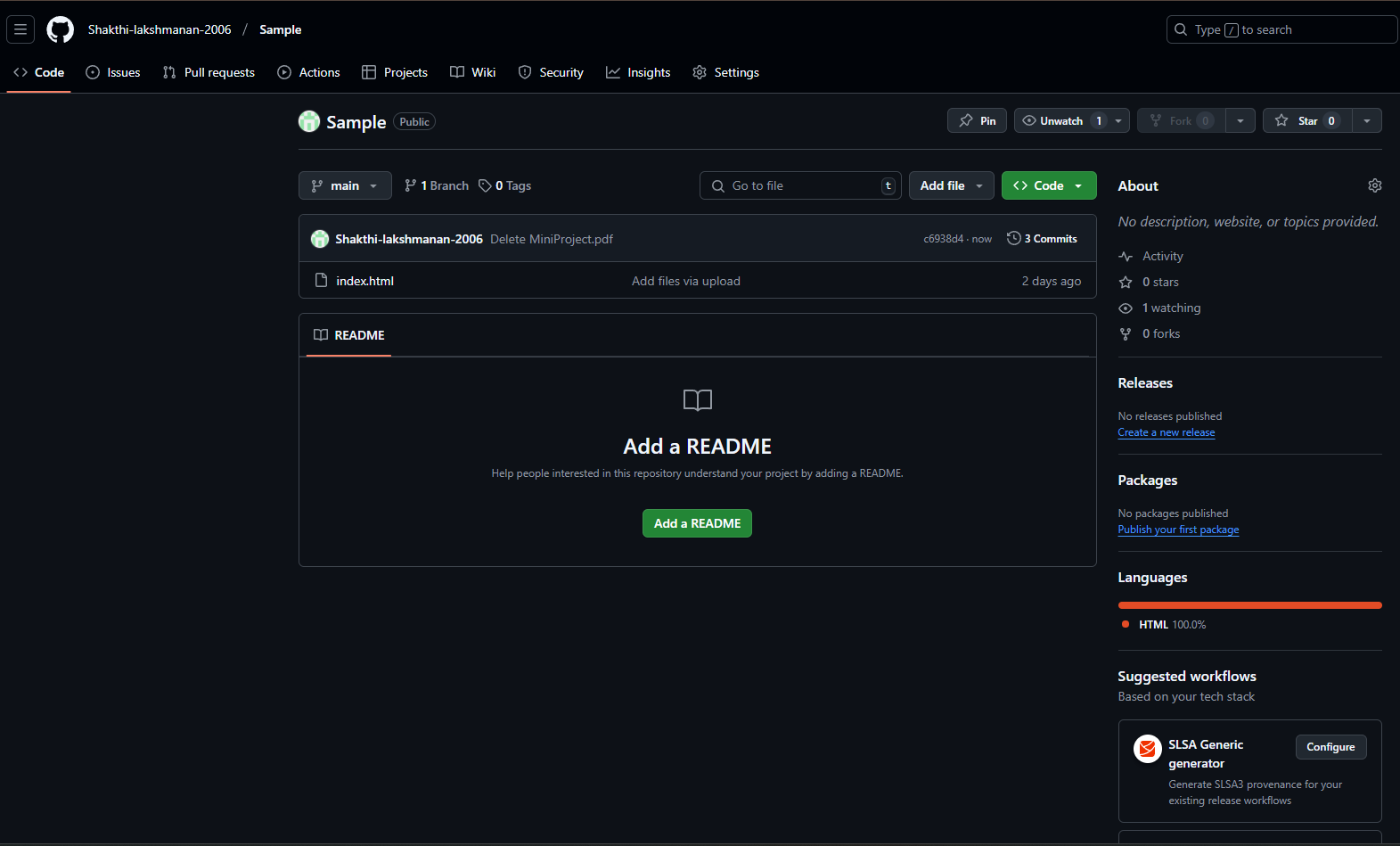
* **Hands-On Cloud Experience** – A great way to start learning cloud platforms and virtual machine management.
* **Scalability** – Easily adjust resources as website traffic increases.
* **Global Accessibility** – A cloud-hosted website can be accessed from anywhere with low latency.
* **Full Control & Customization** – Allows advanced configurations for performance and security.
* **Foundation for Advanced Hosting** – Prepares for hosting dynamic websites, APIs, and load balancing.
* **Career Growth** – Hosting websites on the cloud is a valuable skill for IT and cloud computing roles.

By following this simple process, beginners can gain practical experience in cloud computing and web hosting, setting the stage for more advanced projects. 🚀

# Step-by-Step Overview

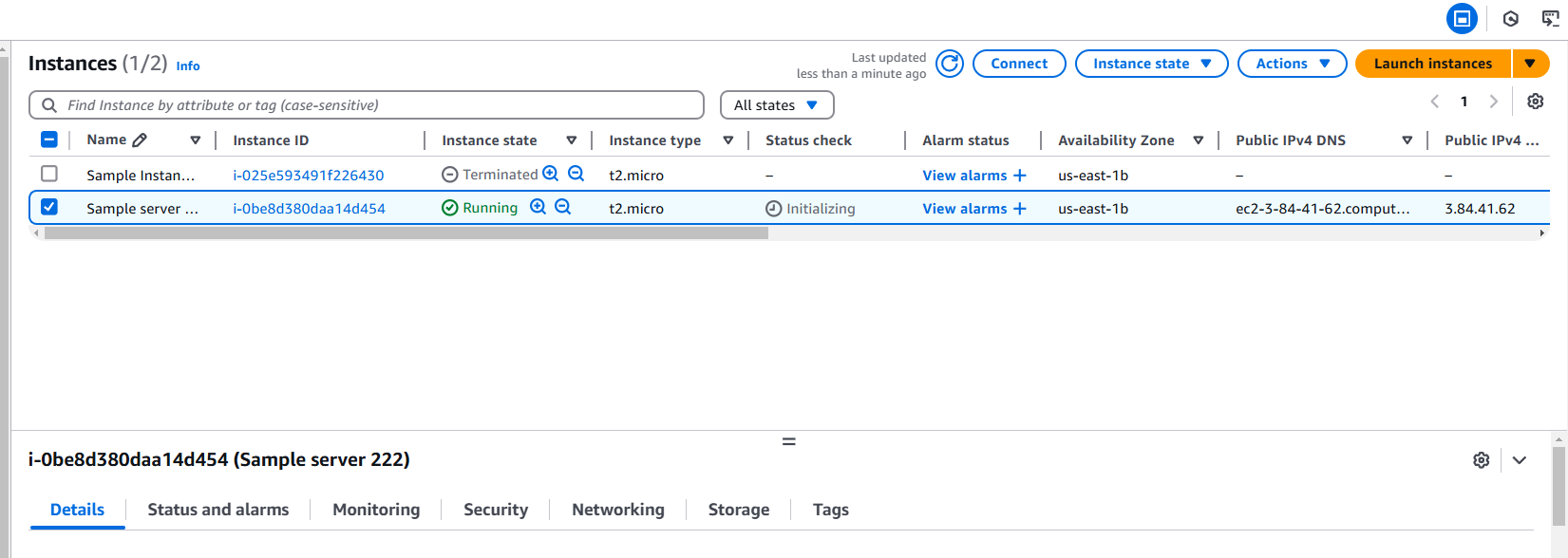
## Step 1:

Have an HTML file (with any related assets like CSS/JavaScript) that you want to host in your GitHub repository



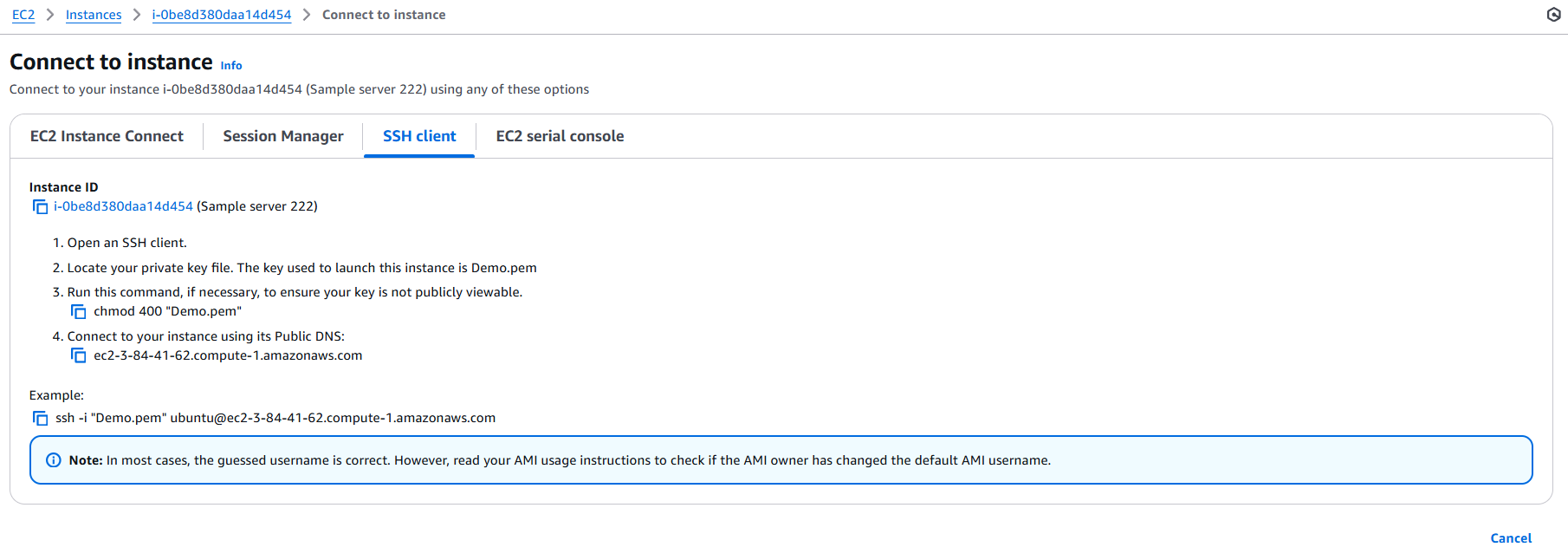
## Step 2:

Launch an EC2 instance, select Ubuntu as the OS, configure security groups to allow all network traffic, create a key pair (e.g., demo . pem), and download it for SSH access



## Step 3:

Click the 'Connect' option on your launched instance, go to the SSH client section, and copy the command provided under the 'Example' section.



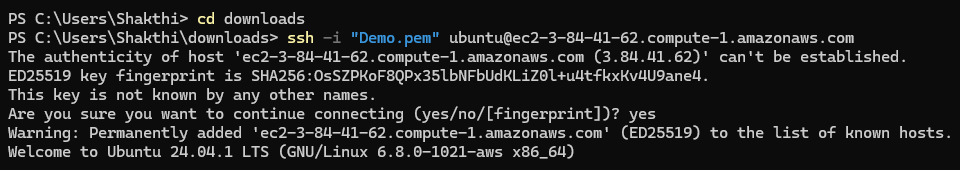
Step 4:

Open PowerShell, navigate to the 'Downloads' directory where the downloaded key pair is located using the **cd Downloads** command



## Step 5:

Paste the command copied from the EC2 Connect's SSH client section, replace the key pair name with your downloaded key (e.g., demo . pem), press Enter, and type 'yes' when prompted.



## Step 6:

Run the command **sudo apt update** to update the package list



## Step 7:

Run the command **sudo apt upgrade**, and press 'Y' to confirm and continue the upgrade process.



## Step 8:

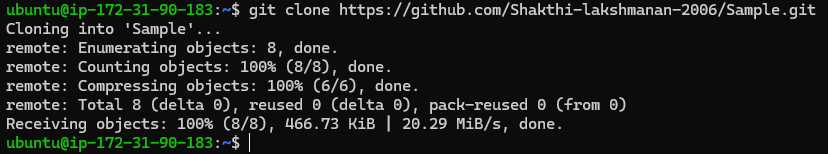
Install the Apache server by running the command **sudo apt install apache2**, and press 'Y' to confirm the installation



## Step 9:

Insert your files by running the command **git clone**

**<repository\_link>** to clone your repository containing the website files



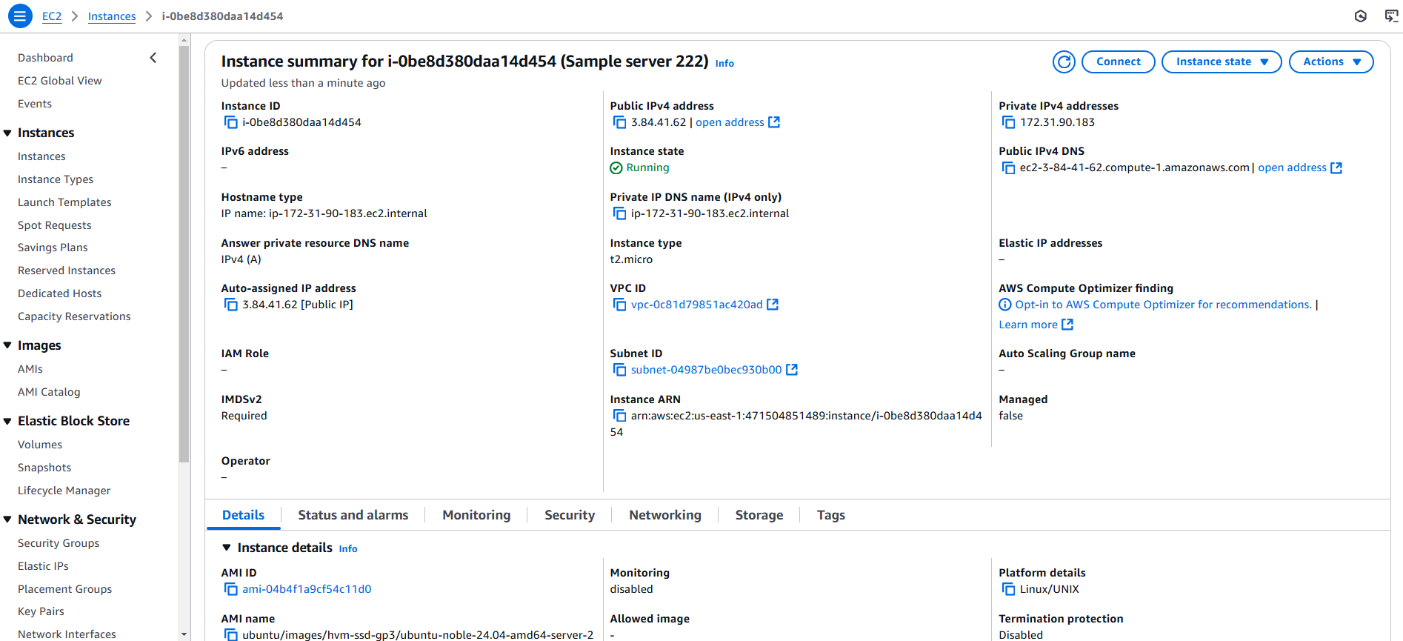
## Step 10:

Run the command **cd /var/www/html** to navigate to the web server's root directory, then type ls to verify that your HTML files from the GitHub repository are present.



## Step 11:

Copy the Public IPv4 DNS from the instance details page in the EC2 console, as shown in the image below.



## Step 12:

Open Chrome and paste the copied Public IPv4 DNS in the address bar to view the content of your index.html file.

=



# Outcome

By completing this PoC of deploying a static website using an EC2 instance, you will:

1. Launch and configure an EC2 instance with Ubuntu as the OS.
2. Install and configure Apache web server to serve your static website.
3. Clone your GitHub repository containing your static website files (HTML, CSS, JavaScript) onto your EC2 instance.
4. Upload and place the website files in the Apache root directory (/var/www/html).
5. Access your static website live on the web using the EC2 instance's Public IPv4 DNS.