



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:MONISH KJ

DEPT: ADS

REG NO: 312323243103



Introduction

A static website serves pre-written HTML, CSS, and JavaScript files to the end user without requiring server-side processing. Hosting such websites on a cloud-based Virtual Machine (VM) has become a preferred choice for individuals and businesses due to its flexibility, scalability, and cost-effectiveness. By leveraging the cloud, developers can quickly deploy websites accessible from anywhere in the world.

Overview

Hosting a static website on a cloud VM involves the following key steps:

- **1. Provisioning a Cloud VM:** Setting up a virtual machine on a cloud provider (like AWS, Azure, or GCP).
- **2. Installing a Web Server:** Configuring a web server such as Apache to serve the website's static files.
- **3. Uploading Website Files:** Placing HTML, CSS, and JavaScript files in the web server's root directory.
- **4. Configuring Network Access:** Ensuring that the web server is accessible via HTTP (port 80) from anywhere.
- **5. Testing and Launching:** Verifying the functionality of the website to make it publicly accessible

Objectives

The primary objectives of hosting a static website on a cloud VM include:

1. Learning Cloud Computing Fundamentals: Understanding how virtual machines operate in a cloud environment.

- **2. Practical Web Hosting Skills:** Gaining hands-on experience in setting up and configuring web servers like Apache or Nginx.
- **3. Website Deployment:** Successfully deploying and making a static website live on the internet.
- **4. Understanding Networking Basics:** Learning about firewall rules, security groups, and HTTP protocol configurations.
- **5. Cost-Effective Hosting:** Exploring affordable methods to host lightweight websites without needing managed services.

Importance

- **1. Hands-On Cloud Experience:** Hosting a static website on a cloud VM is an excellent starting point for understanding the capabilities of cloud platforms and virtual machine operations.
- **2. Scalability:** Cloud-based hosting provides flexibility to scale resources up or down as the traffic to the website grows.
- 3. Global Accessibility: By deploying on the cloud, the

website

becomes accessible from any part of the world with minimal latency.

- **4. Customization and Control:** Cloud VMs allow complete control over the hosting environment, enabling advanced configurations and optimizations.
- 5. Foundation for Advanced Hosting: It lays the groundwork

for

m

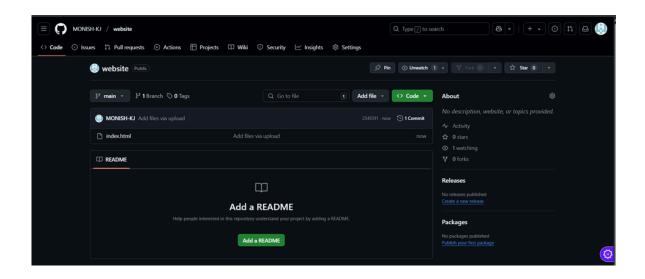
or using load balancers.

6. Professional Development: Learning to host websites on APoIrse, advanced projects, such as hosting dynamic websites,

Step-by-Step Overview

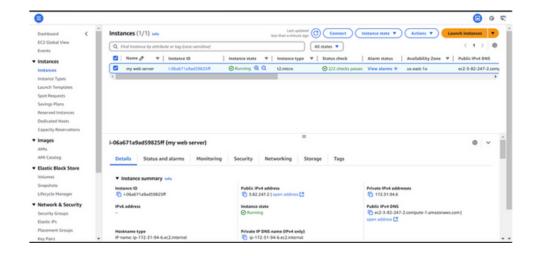
Step 1:

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



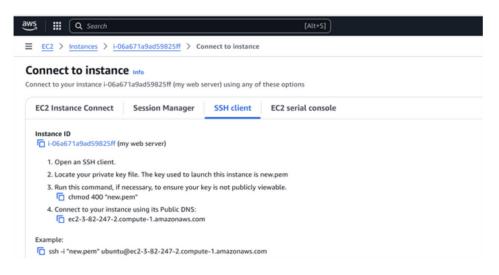
Step 2:

Launch an EC2 instance, select Ubuntu as the OS, configure security groups to allow all network traffic, create a key pair (e.g., new.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

PS C:\Users\Hi> cd Downloads

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., new.pem), press Enter, and type 'yes' when prompted.

Step 6:

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

```
ubuntu@ip-172-31-94-6:~$ sudo apt update
```

Step 7:

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

```
ubuntu@ip-172-31-94-6:~$ sudo apt upgrade
```

Step 8:

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

```
ubuntu@ip-172-31-94-6:~$ sudo apt install apache2
```

Step 9:

Insert your files by running the command **git clone <repository_link>** to clone your repository containing the website files

```
ubuntu@ip-172-31-94-6:~$ git clone https://github.com/SaravanaKrishnan16/my-static-website
Cloning into 'my-static-website'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 3 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (3/3), done.
```

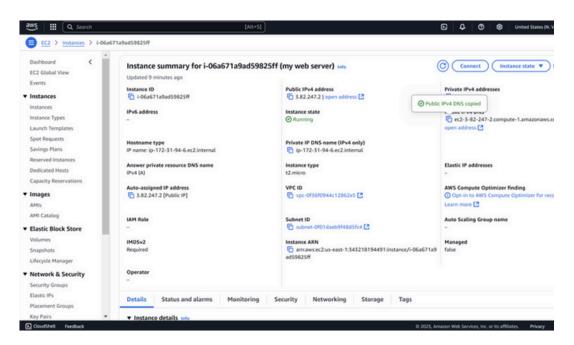
Step 10:

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

```
ubuntu@ip-172-31-94-6:~$ cd /var/www/htmlubuntu@ip-172-31-94-6:/var/www/html$ ls index.html
```

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.

=



Welcome to My Website!

This is a static website hosted on GitHub Pages.

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.