St. JOSEPH'S GROUP OF INSTITUTIONS
OMR, CHENNAI - 119

aws

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: CHANDRU S

Department: INFORMATION TECHNOLOGY

St. JOSEPH'S COLLEGE OF ENGINEERING | St. JOSEPH'S INSTITUTE OF TECHNOLOGY

AUTONOMOUS INSTITUTIONS, AFFILIATED TO ANNA UNIVERSITY

Introduction

A static website delivers pre-written HTML, CSS, and JavaScript files directly to users without requiring server-side processing. Hosting such websites on a cloud-based Virtual Machine (VM) has become a popular choice due to its flexibility, scalability, and cost-effectiveness. By leveraging cloud infrastructure, developers can efficiently deploy websites that are 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 or Nginx to serve static files.
3. **Uploading Website Files** – Placing HTML, CSS, and JavaScript files in the web server's root directory.
4. **Configuring Network Access** – Ensuring the web server is accessible via HTTP (port 80) from the internet.
5. **Testing and Deployment** – Verifying the website's functionality and making it publicly accessible.

Objectives

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

1. **Understanding Cloud Computing** – Learning how virtual machines operate in a cloud environment.
2. **Hands-on Web Hosting** – Gaining experience in setting up and configuring web servers like Apache or Nginx.
3. **Deploying a Website** – Successfully hosting a static website and making it live on the internet.
4. **Networking Fundamentals** – Learning about firewall rules, security groups, and HTTP protocol configurations.
5. **Cost-Effective Hosting** – Exploring affordable ways to host lightweight websites without relying on managed services.

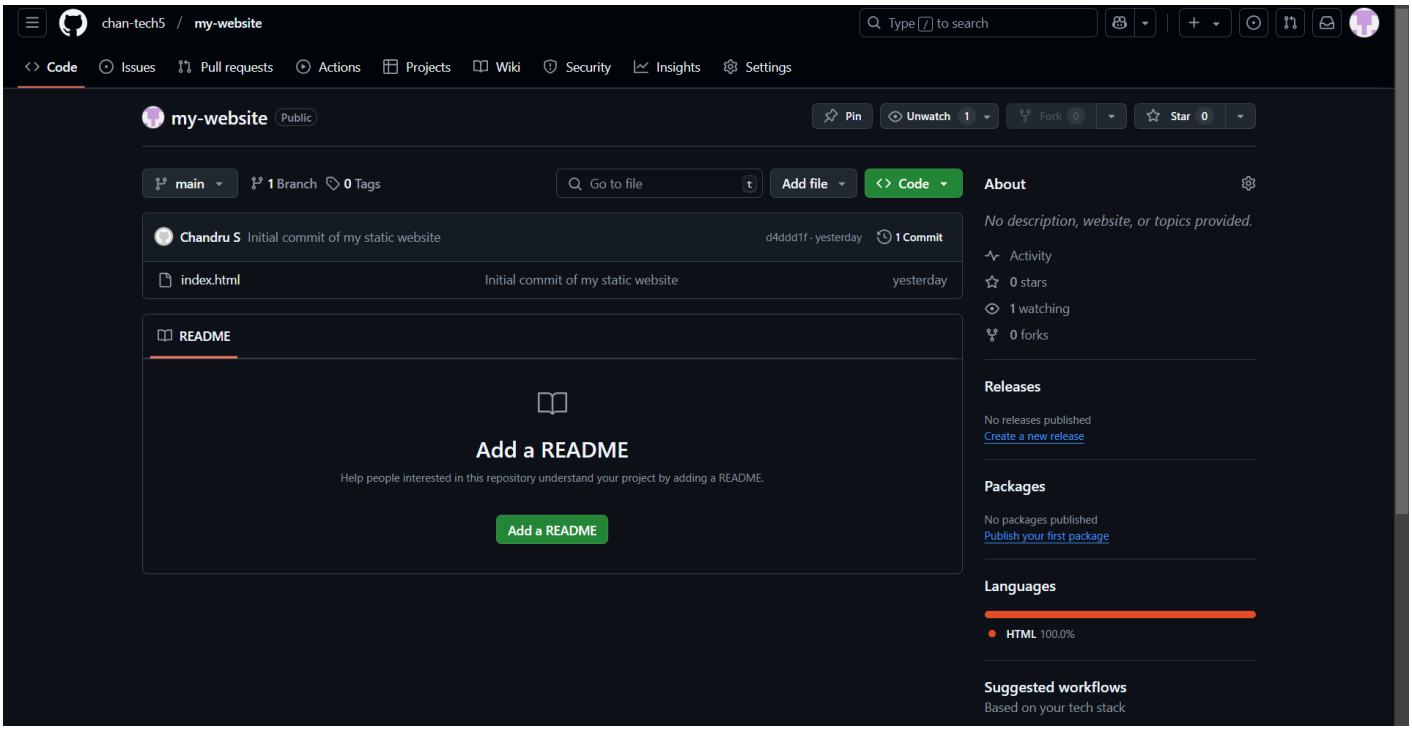
Importance

- **Hands-On Cloud Experience** – Hosting a website on a cloud VM provides practical exposure to cloud platforms and virtual machine management.
- **Scalability** – Cloud hosting allows for easy scaling of resources as website traffic increases.
- **Global Accessibility** – Cloud deployment ensures low-latency access to the website from anywhere in the world.
- **Customization & Control** – Cloud VMs offer complete control over the hosting environment, enabling advanced configurations.
- **Foundation for Advanced Hosting** – Lays the groundwork for hosting dynamic websites, APIs, or using load balancers.
- **Professional Growth** – Enhances cloud computing and web hosting skills, valuable for career advancement.

Step-by-Step Overview

Step 1:

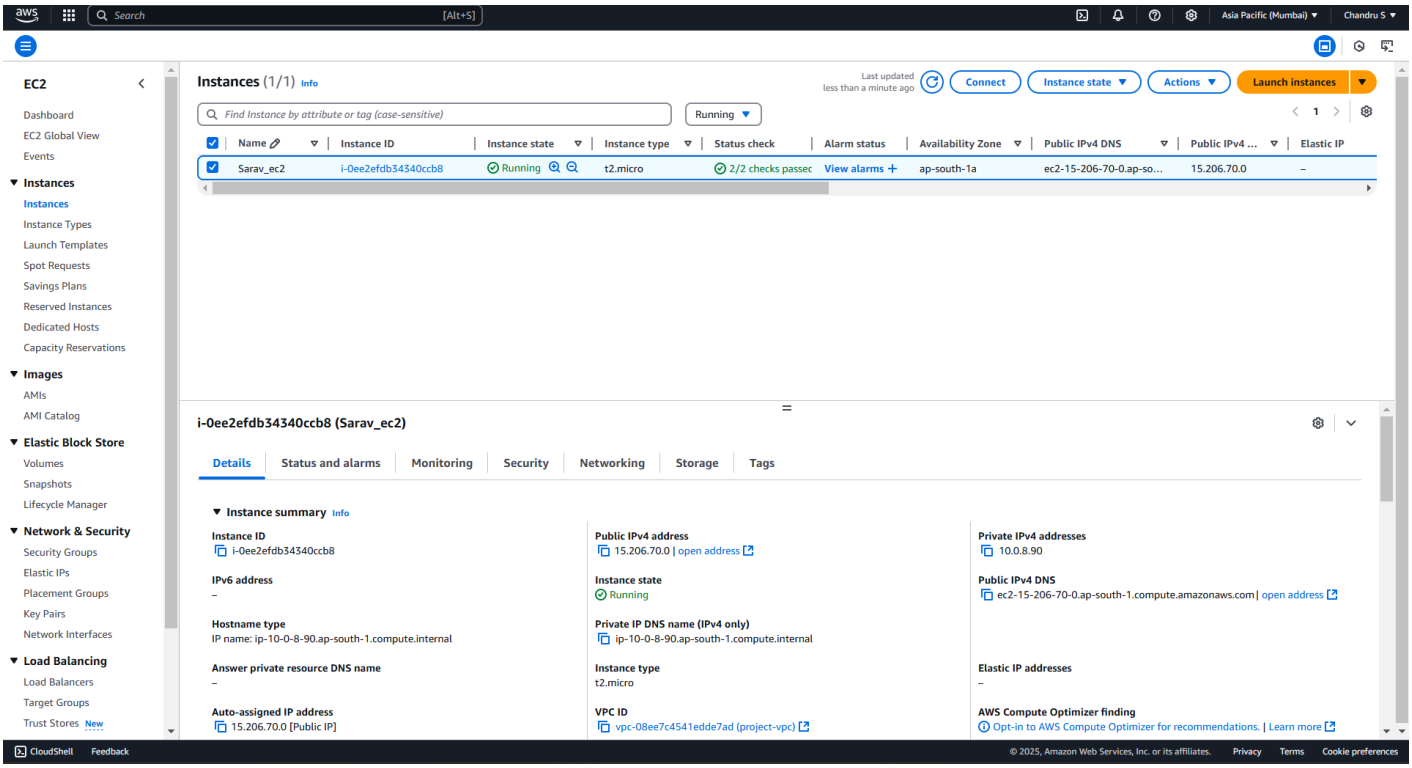
Ensure you have an HTML file (along with any related CSS and JavaScript files) stored in a GitHub repository.



Step 2:

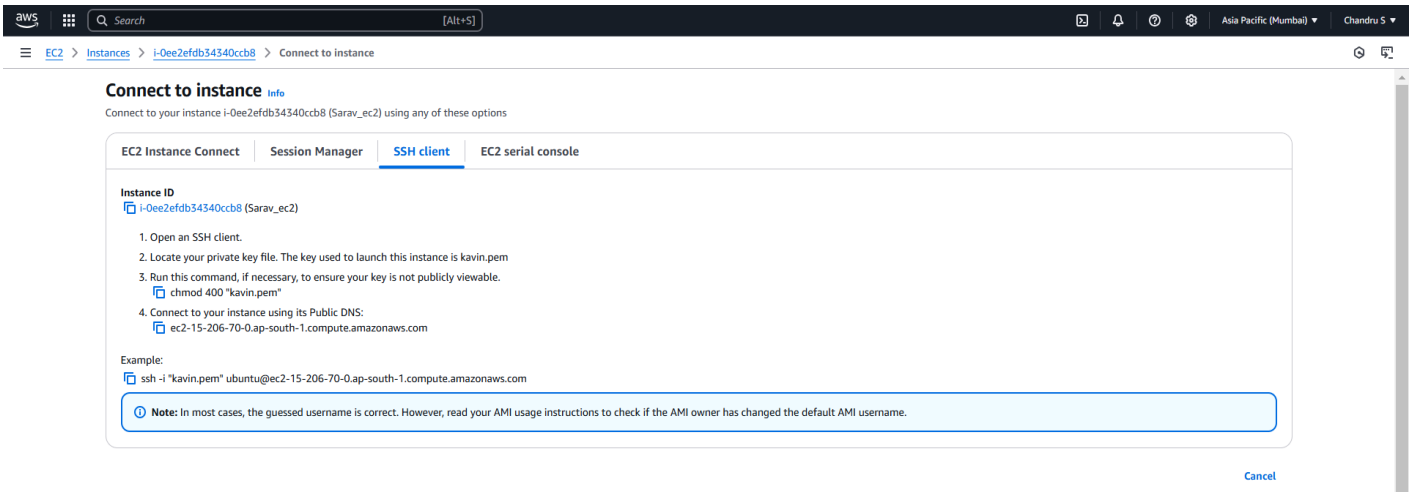
- Open the AWS Management Console and navigate to EC2.
- Click **"Launch Instance"** and choose **Ubuntu** as the operating system.

- Configure Security Groups to allow HTTP (port 80) and SSH (port 22) access.
- Create a key pair (e.g., new.pem) and download it for SSH access.



Step 3:

- In the EC2 dashboard, select your running instance.
- Click **"Connect"** and go to the **SSH Client** section.
- Copy the provided SSH command under the **"Example"** section.



Step 4:

- Open PowerShell (or Terminal on macOS/Linux).
- Navigate to the directory where the key pair (new.pem) is stored:
  cd Downloads

```
PS C:\Users\chandru> cd Downloads
```

Step 5:

- Paste the **SSH command** copied from the EC2 **Connect** page.
- Replace the key pair name with your downloaded key (e.g., new.pem).
- Press **Enter**, then type **"yes"** when prompted to establish the connection.

```
PS C:\Users\chandru\Downloads> ssh -i "kavin.pem" ubuntu@ec2-15-206-70-0.ap-south-1.compute.amazonaws.com
The authenticity of host 'ec2-15-206-70-0.ap-south-1.compute.amazonaws.com (15.206.70.0)' can't be established.
ED25519 key fingerprint is SHA256:FSWnn052cGFXiV69IkPAoHpFRaUX5jmSG0xgMd3NI5s.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
```

Step 6:

Run the following command to update the system package list:
sudo apt update

```
ubuntu@ip-10-0-8-90:~$ sudo apt update
```

Step 7:

Upgrade all installed packages to their latest versions:
sudo apt upgrade

```
ubuntu@ip-10-0-8-90:~$ sudo apt upgrade
```

Step 8:

Install Apache to serve your static website:
sudo apt install apache2

```
ubuntu@ip-10-0-8-90:~$ sudo apt install apache2
```

Step 9:

Download your website files from GitHub:
git clone <repository_link>

```
ubuntu@ip-10-0-8-90:~$ git clone https://github.com/chan-tech5/my-website.git
Cloning into 'my-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.
```

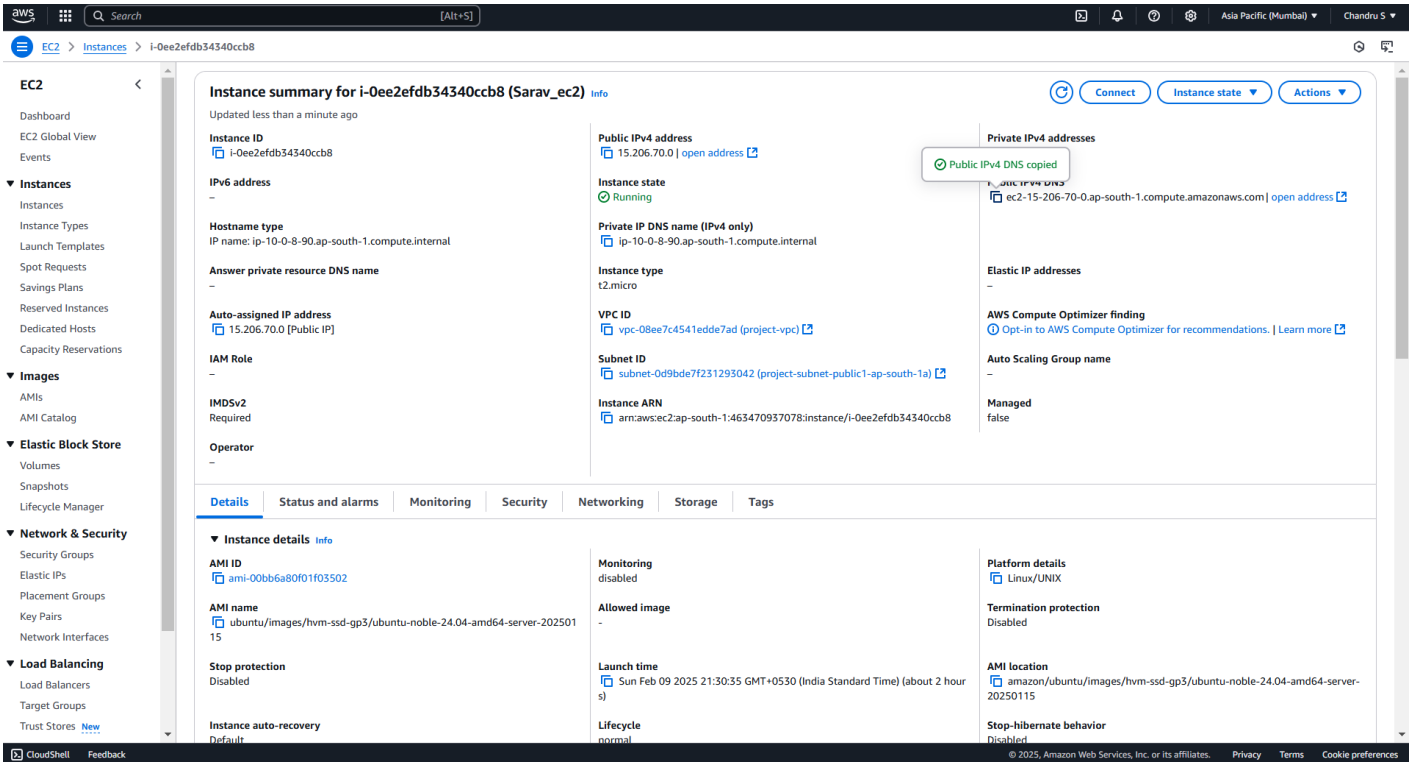
Step 10:

- Navigate to the web server's root directory:
  cd /var/www/html
- Verify that your HTML files from the GitHub repository are present:
  ls

```
ubuntu@ip-10-0-8-90:~$ cd /var/www/html
ubuntu@ip-10-0-8-90:/var/www/html$ ls
index.html
```

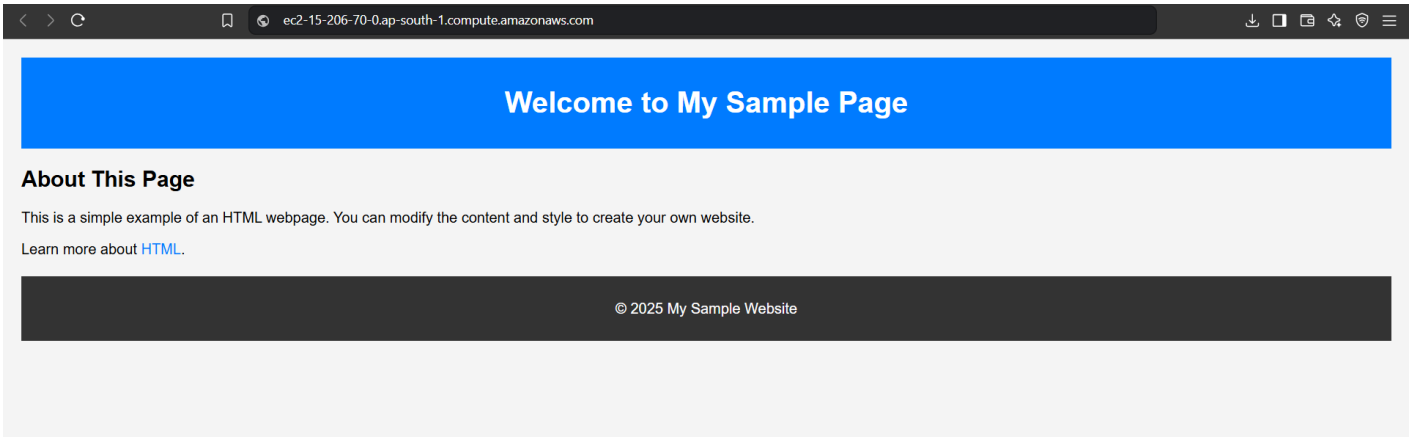
Step 11:

- Go to the EC2 dashboard in AWS.
- Copy the Public IPv4 DNS from the instance details page.



Step 12:

- Open a web browser (Chrome, Edge, or Firefox).
- Paste the Public IPv4 DNS into the address bar and press Enter.
- Your **index.html** file should be displayed, confirming the successful deployment of your static website.



Outcome

By completing this Proof of Concept (PoC) for deploying a static website on an EC2 instance, you will:

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