

## Advanced DevOps Lab

### Experiment 0

**Name:** Sreeekesh Iyer

**Class/Roll No.** D15A/24

**Aim:** To develop a website and host it on your local machine and using a VM on AWS Cloud.

#### **Theory and Pre Requisites:**

To host a website, we need its contents ready, preferably on a GitHub repository for easy fetching. In this example, we will host a portfolio website from [this GitHub Repository](#). This repository has HTML and CSS, which means there are no build commands needed for hosting. If you are hosting a website using a Javascript framework, for example, the build command will be required for hosting it locally or on the Cloud.

#### **Part 1: Hosting a website on localhost**

##### **Steps:**

1. Downloading XAMPP.

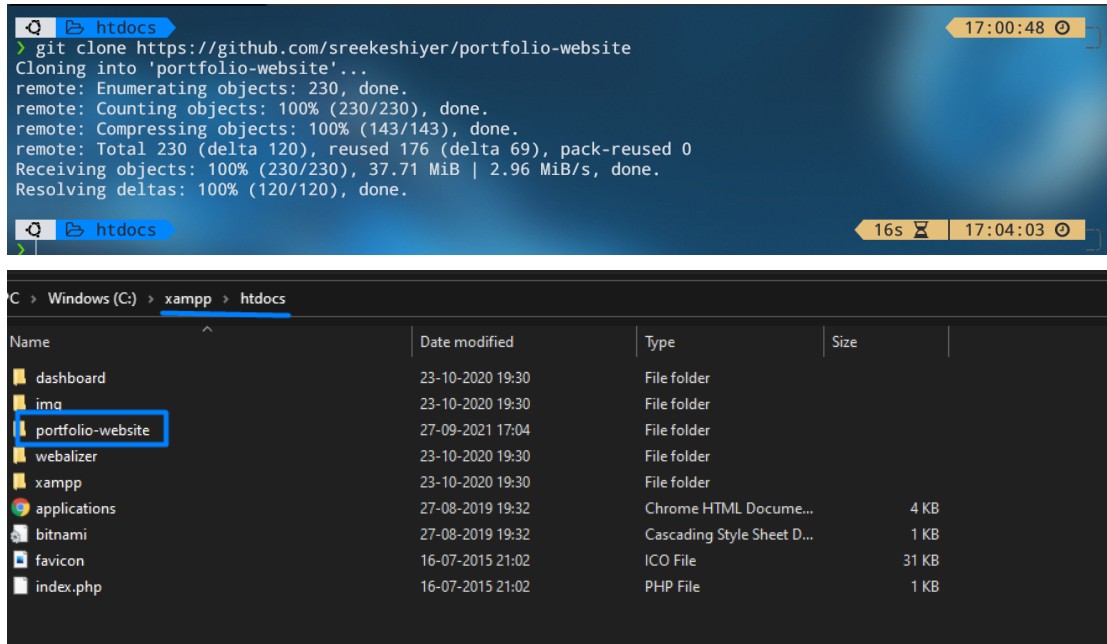


We need to install XAMPP, which creates an apache server for us on our local machine, on which we can host our website. You can also use applications like WAMP or MAMP

depending on your Operating System.

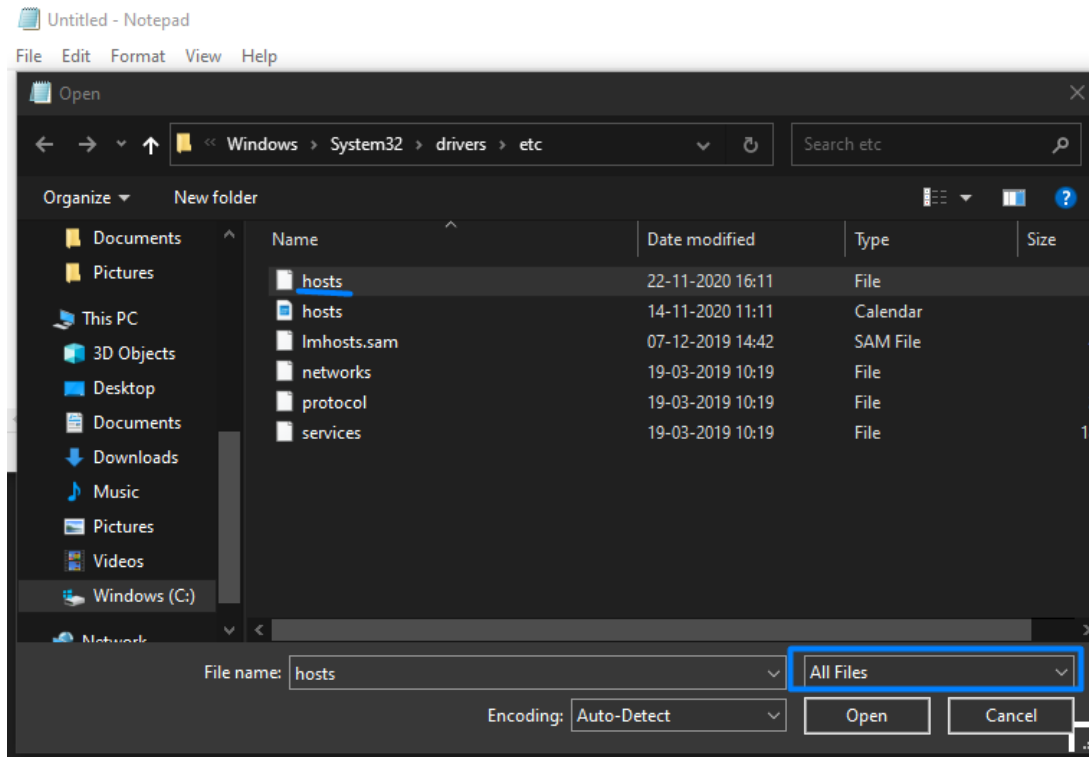
Please Note: Make sure you select Apache when you install XAMPP, so that you can use it with XAMPP later.

## 2. Dropping the Code folder in the **htdocs** folder



Go to the xampp root folder, then locate the htdocs folder, such that you are now in `xampp/htdocs`. Now, open the Terminal here and use the ***git clone*** command to clone your code folder in this directory.

## 3. Update the hosts file to serve localhost as your domain name (will work only on your local machine)



Open the notepad as administrator and open the **hosts** file in *Windows/System32/drivers/etc*. Change the filter to All Files to find the **hosts** file.

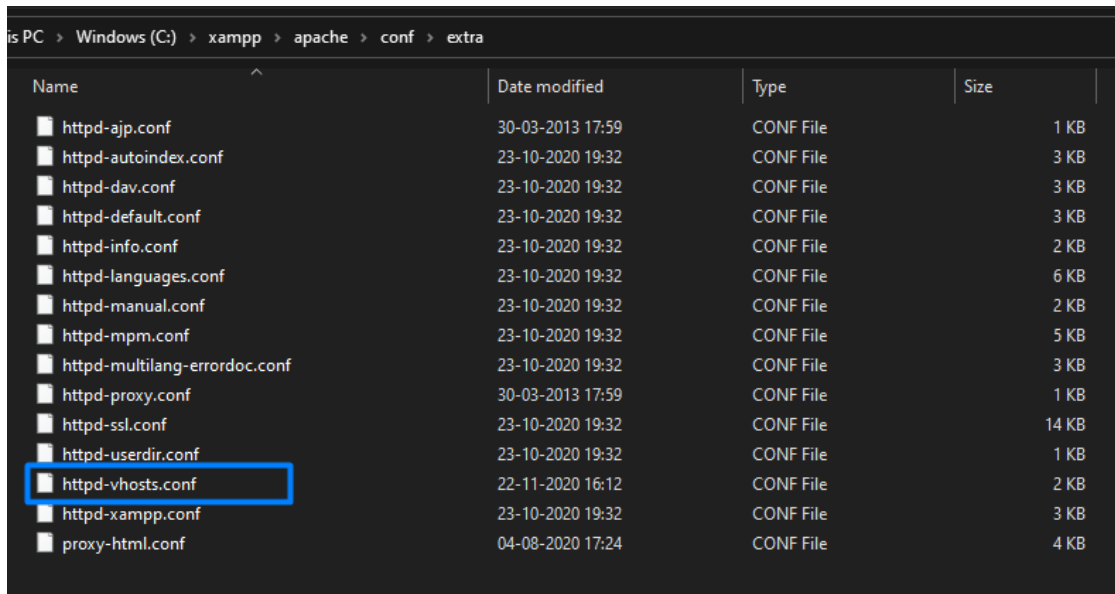
```
hosts - Notepad
File Edit Format View Help
#
# For example:
#
#      102.54.94.97      rhino.acme.com      # source server
#      38.25.63.10      x.acme.com        # x client host

# localhost name resolution is handled within DNS itself.
#      127.0.0.1        localhost
#      ::1              localhost

127.0.0.1 localhost
127.0.0.1 sreekeshiyer.com
```

Then, on a new line, enter localhost and map it with your desired domain name, in my case, it is sreekeshiyer.com, as shown above.

#### 4. Update the httpd-vhosts.conf file



Name	Date modified	Type	Size
httpd-ajp.conf	30-03-2013 17:59	CONF File	1 KB
httpd-autoindex.conf	23-10-2020 19:32	CONF File	3 KB
httpd-dav.conf	23-10-2020 19:32	CONF File	3 KB
httpd-default.conf	23-10-2020 19:32	CONF File	3 KB
httpd-info.conf	23-10-2020 19:32	CONF File	2 KB
httpd-languages.conf	23-10-2020 19:32	CONF File	6 KB
httpd-manual.conf	23-10-2020 19:32	CONF File	2 KB
httpd-mpm.conf	23-10-2020 19:32	CONF File	5 KB
httpd-multilang-errordoc.conf	23-10-2020 19:32	CONF File	3 KB
httpd-proxy.conf	30-03-2013 17:59	CONF File	1 KB
httpd-ssl.conf	23-10-2020 19:32	CONF File	14 KB
httpd-userdir.conf	23-10-2020 19:32	CONF File	1 KB
httpd-vhosts.conf	22-11-2020 16:12	CONF File	2 KB
httpd-xampp.conf	23-10-2020 19:32	CONF File	3 KB
proxy-html.conf	04-08-2020 17:24	CONF File	4 KB

Locate the httpd-vhosts.conf file in xampp/apache/conf/extra. Open this file with your desired text editor, in my case, VSCode.

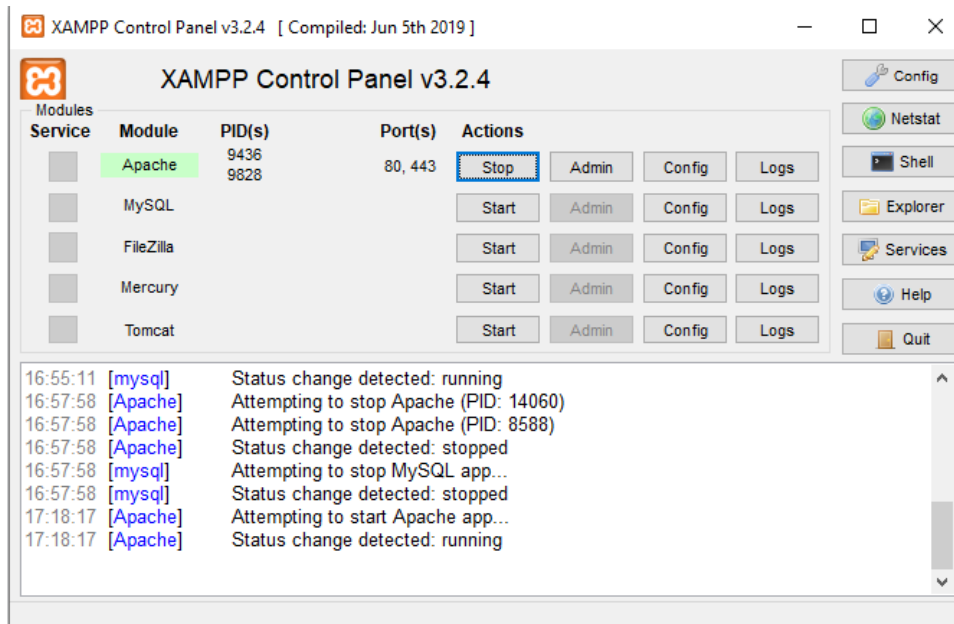
```
##<VirtualHost *:80>
    ##ServerAdmin webmaster@dummy-host2.example.com
    ##DocumentRoot "C:/xampp/htdocs/dummy-host2.example.com"
    ##ServerName dummy-host2.example.com
    ##ErrorLog "logs/dummy-host2.example.com-error.log"
    ##CustomLog "logs/dummy-host2.example.com-access.log" common
##</VirtualHost>

<VirtualHost *:80>
    DocumentRoot "C:/xampp/htdocs/portfolio-website/dist"
    ServerName sreekeshiyer.com
</VirtualHost>
```

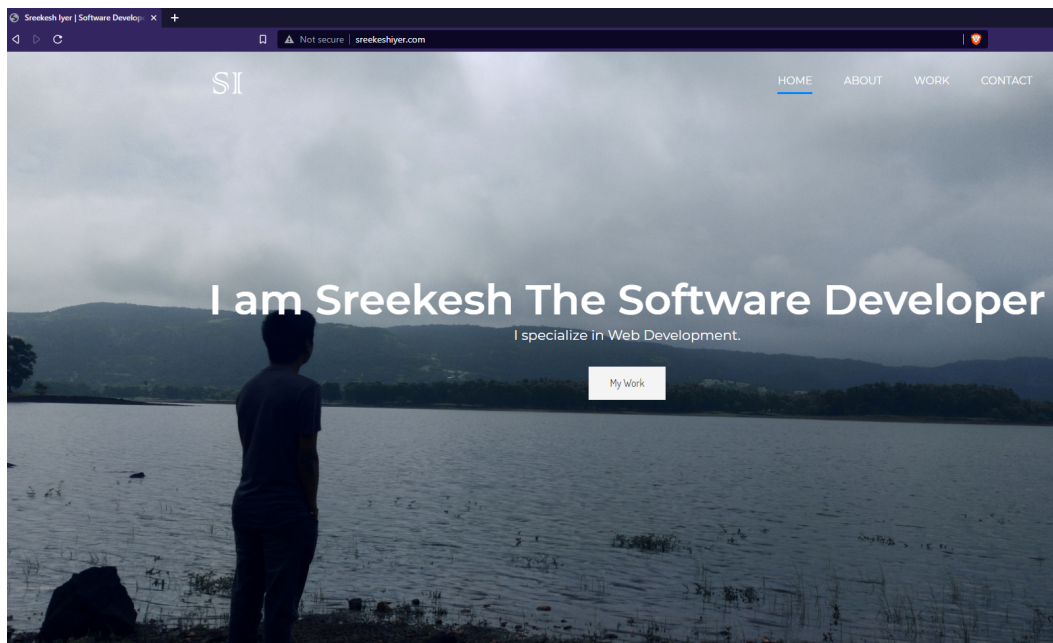
All the code in this file is commented out. Copy the last VirtualHost set which is commented out and paste it out, like so. Inside this tag, make the Document root mapped to the website folder, like so. Change the server name to the mapped domain name, as you did in the hosts folder.

Note: This could be the root folder for you. In my case, I have my HTML in the dist folder.

5. Open the XAMPP control panel and start the apache server



6. Open up your domain name on your browser



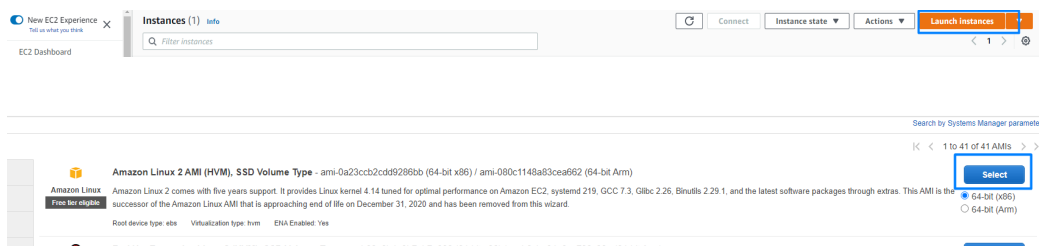
That's it, you have successfully hosted your website locally on your machine. If you have a static IP, you can do this publicly, except you need to buy a domain name for mapping it so that you can make the website public. You can still use services like ngrok to share this website temporarily with your friends, family or colleagues, etc.

## Part 2: Hosting a website on a Cloud VM (AWS EC2 Instance)

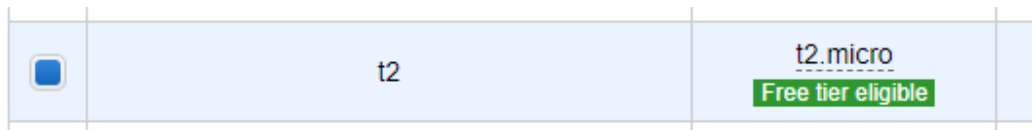
To host our website on Cloud, we need to set up a Virtual Machine, in AWS EC2, in this case. You'd need an AWS free tier account to proceed.

Steps:

### 1. Open up EC2 Console and Launch a new Instance



Choose the Linux 2 AMI.



Choose t2.micro for configuration, which is the free-tier-eligible one.

### Step 5: Add Tags

A tag consists of a case-sensitive key-value pair. For example, you could define a tag with key = Name and value = Webserver.

A copy of a tag can be applied to volumes, instances or both.

Tags will be applied to all instances and volumes. [Learn more](#) about tagging your Amazon EC2 resources.

Key	Value
Name	New-Portfolio-Instance

[Add another tag](#) (Up to 50 tags maximum)

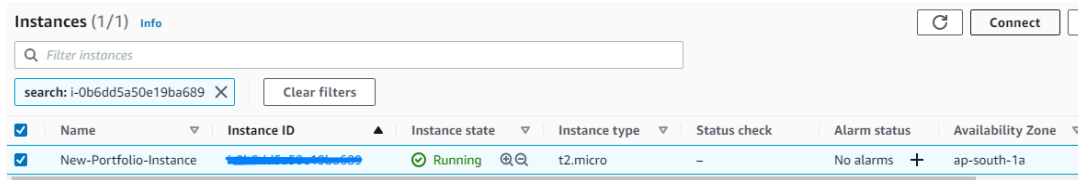
Proceed with everything default, then add a tag optionally, to identify the machine.

Type	Protocol	Port Range	Source
HTTP	TCP	80	0.0.0.0/0
HTTP	TCP	80	:::0
SSH	TCP	22	0.0.0.0/0

In Security groups, add these rules so that you can visit the website from anywhere.

Review the changes and launch. Create and download a Keypair file if asked for.

## 2. Connect to the instance to access the CLI



Click on the Connect button on the top right.

Use EC2 instance connect to directly access the machine on a CLI, opened on the browser.

## 3. Install Git and HTTPD

Use the following commands to install git and httpd.

```
sudo yum install git -y
sudo yum install httpd -y
```

## 4. Set up the GitHub Repository to host the website

Go to var/www/html and clone the GitHub Repository.

```
cd var/www/html
git clone https://github.com/sreekeshiyer/portfolio-website
```

## 5. Move all files inside the folder out to the html folder (/dist/\* for me, it might be /\* for you)

```
mv portfolio-website/dist/* .
```

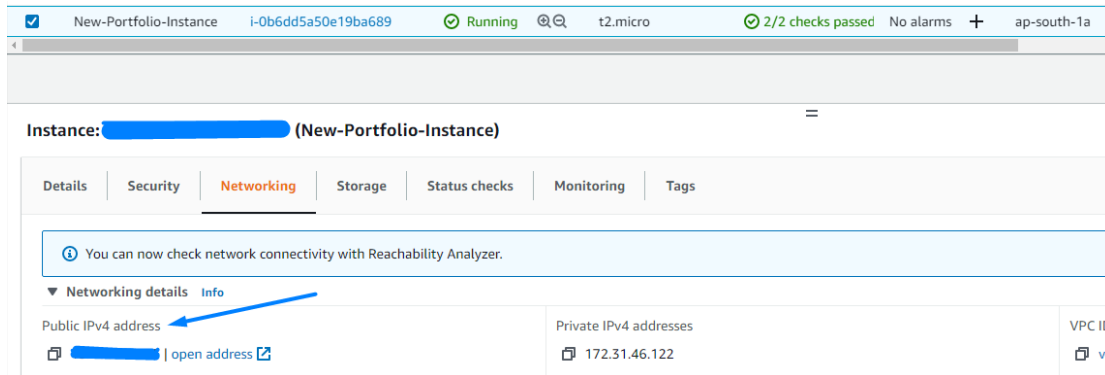
Type the **ls** command just to confirm you have all your html files inside the html directory.

```
about.html contact.html css favicon.ico img index.html js portfolio-website work.html
```

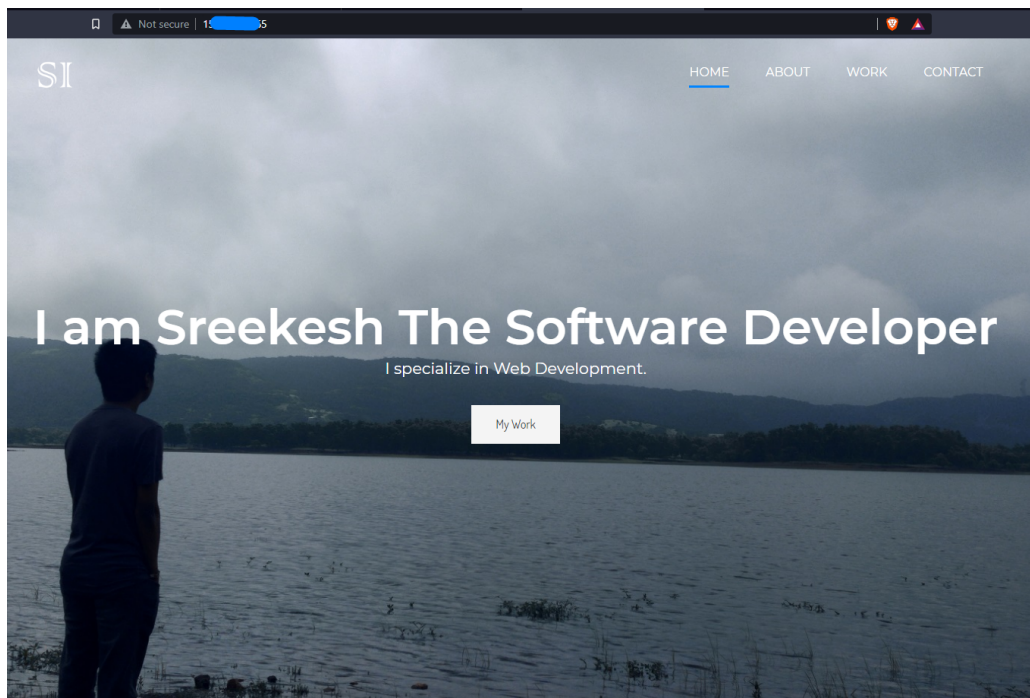
6. Start the httpd server

```
sudo service httpd start
```

7. View your website



Create a new tab on your browser and go to the public IP address of your EC2 instance to confirm your website is live. You can find the Public IP address inside the Networking tab of your EC2 console.



Your website is live!

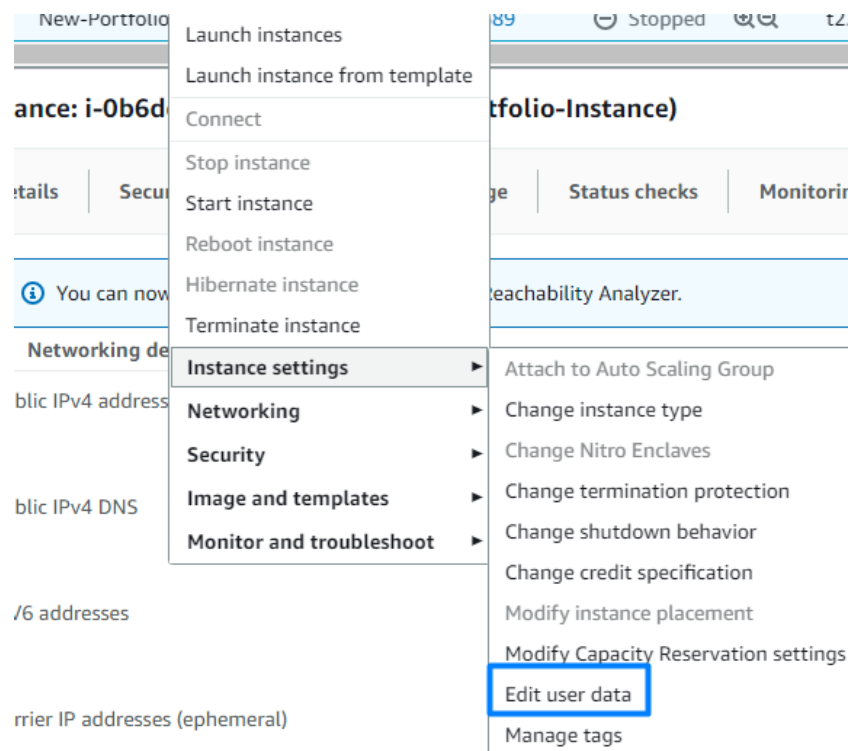


In this way, you can host your website on an AWS EC2 Instance. But in case there is a server downtime in your AZ, you'll have to re-configure your Apache server when the server is back. For this, we can write a shell script that runs on every Instance start.

### Shell Script for starting apache server at launch

To write this shell script, we need to firstly, stop the instance.

Then, Right-click on the instance, choose instance settings and then choose **User Data**.



Content-Type: multipart/mixed; boundary="//"

MIME-Version: 1.0

--//

Content-Type: text/cloud-config; charset="us-ascii"

MIME-Version: 1.0

Content-Transfer-Encoding: 7bit

Content-Disposition: attachment; filename="cloud-config.txt"

```
#cloud-config
cloud_final_modules:
- [scripts-user, always]

--//
Content-Type: text/x-shellscript; charset="us-ascii"
MIME-Version: 1.0
Content-Transfer-Encoding: 7bit
Content-Disposition: attachment; filename="userdata.txt"

#!/bin/bash
sudo su
cd /var/www/html
service httpd start
--//--
```

Paste this inside the User Data and Save the file.

Now, start the instance again, and without further configuration, you can follow the public IP address and view the website live again!

### **Conclusion:**

In this experiment, we learned how to host our websites locally on our machines and on AWS EC2 Instances (Cloud).