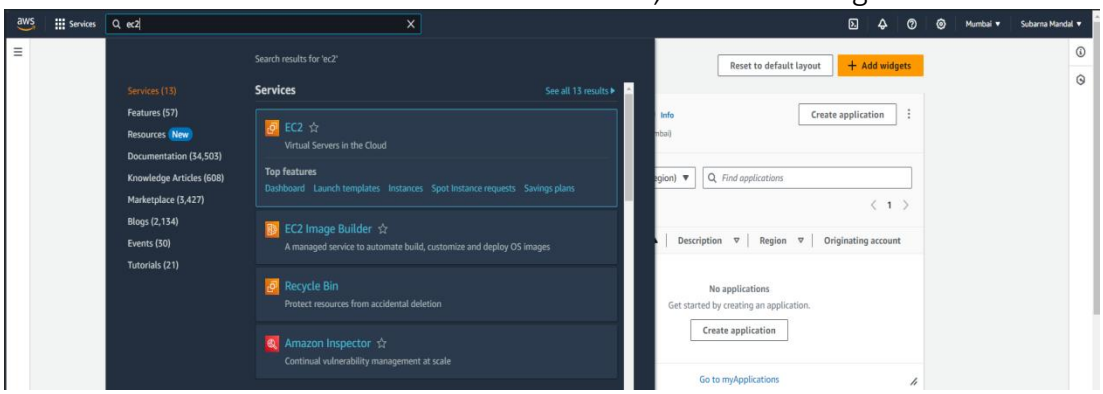
Assignment 7

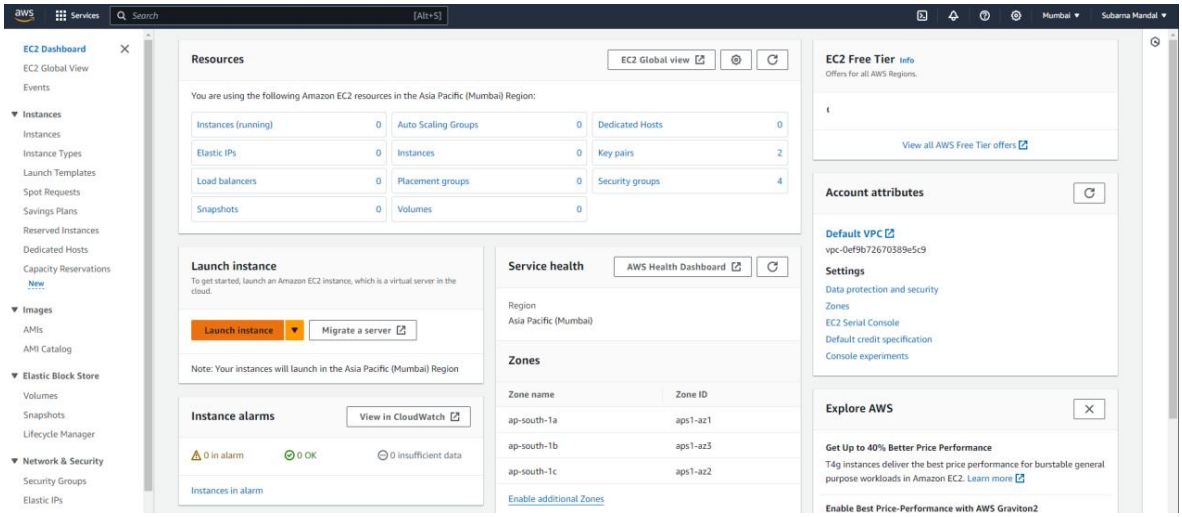
Problem Statement: Host a website on EC2 service of AWS.

Initializing an EC2 instance :

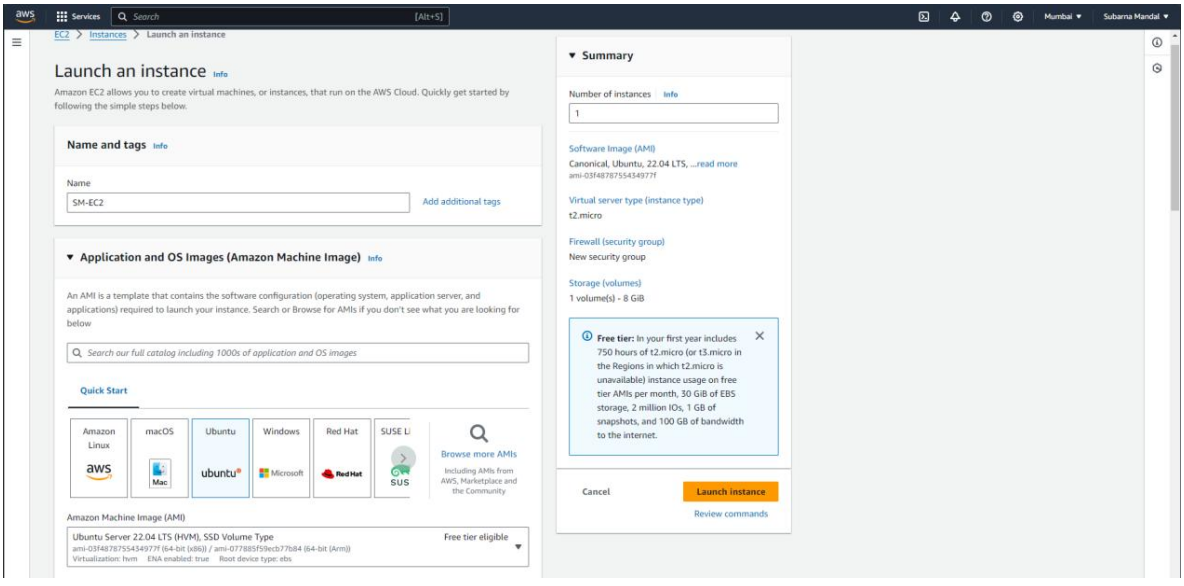
Step1: Login to the AWS console, and search for ec2. Open the first EC2 link. We are directed to a page, where we can view details of all the instances we create, or are running.



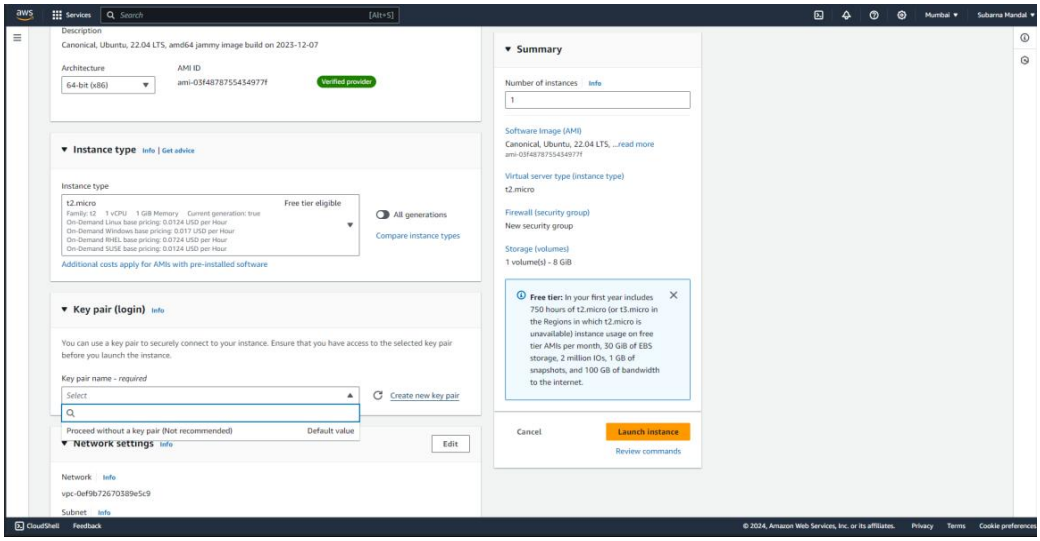
Step2: Click on the Launch Instance button. A dropdown appears, from which select Launch Instance.



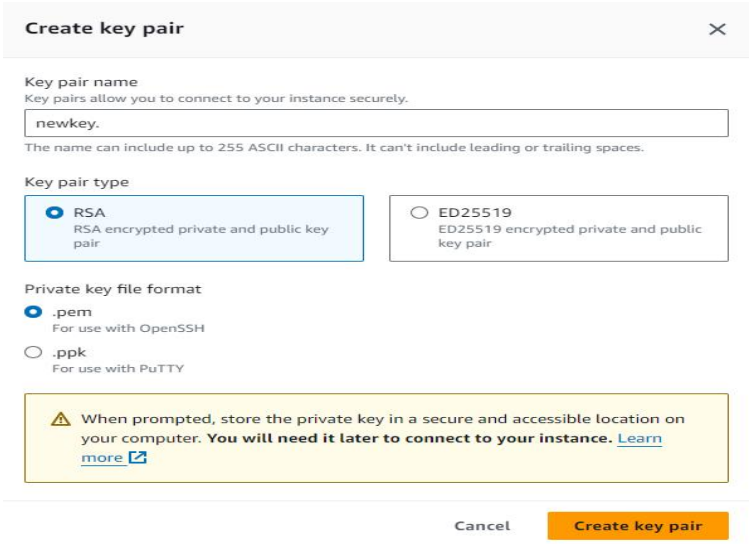
Step3: On the new window that opens, set a name for the instance, and select the Ubuntu platform.



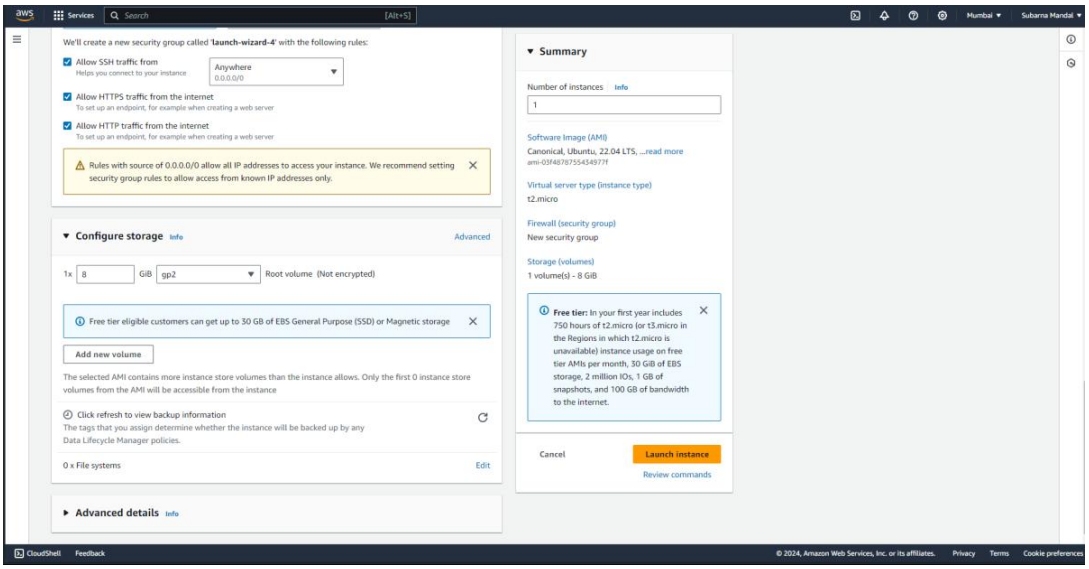
Step4: Create a new key pair, if does not exist already. For this click on Create new key pair.



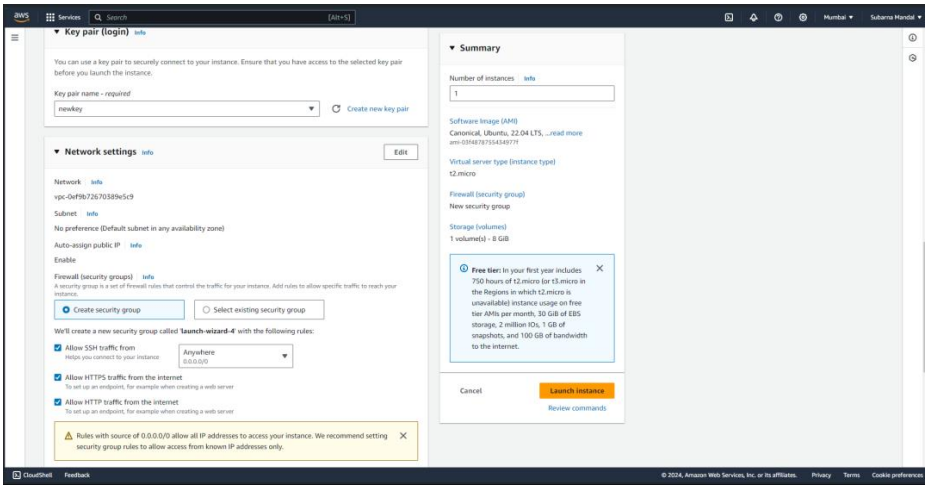
Step5: The new popup appears, where we enter the name of the key pair, and leave all other options default (RSA security option, and the '.pem' extension).Then click on the Create key pair button. Save the file in some directory on your computer.



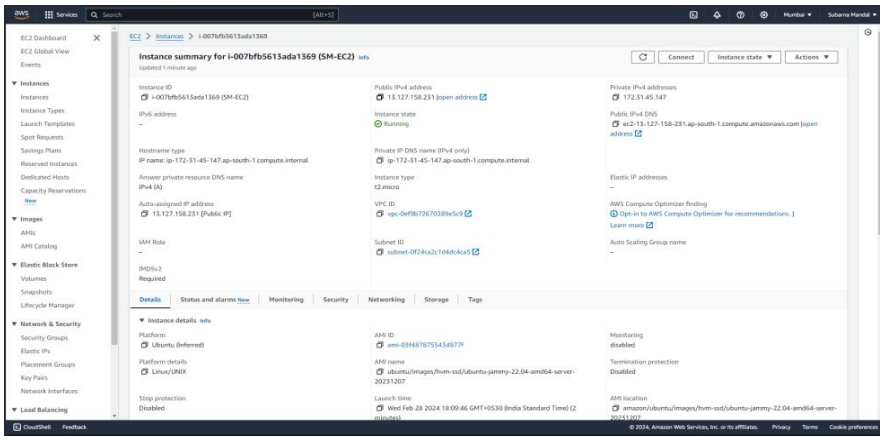
Step6: Then check all the checkboxes, in the firewall settings.



Then click on Launch instance. A success message is shown on a new window. Click on View all instances. After this, a window showing all instances is open.



Step7: Now click on the Instance ID (hidden in the image, but it is actually visible on the website). Copy the public IPv4 address as visible from the new interface that opens.



Step8: Now, our ec2 service is ready to work with.

Creating a static website

Step1: Create two files named index.html and about.html. The file about.html may also be named something else. But, the index.html is a necessary file when the website is hosted on the remote server. Generally, the starting point of a website is index.html. So, index.html will contain the homepage of the website (or, in other words, if your homepage file has some other name, then change it to index.html, for the purpose of this hosting).

Step2: The contents of the files are as follows:

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>index page</title>
</head>
<body>
  <h3 align="center">This is the index page.</h3>
  <p align="center">
    <a href="about.html">about</a>
    <br>
    <a href="contact.html">contact</a>
  </p>
</body>
</html>
```

about.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0"
  <title>about page</title>
</head>
<body>
  <h3 align="center">This is the about page.</h3>

  <p align="center">
    <a href="index.html"> index </a>
    <br>
    <a href="contact.html">contact</a>
  </p>
</body>
</html>
```

About.html

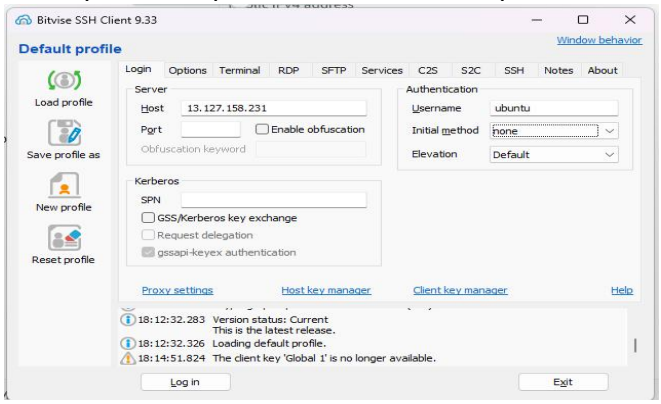
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>contact page</title>
</head>
<body>
  <h3 align="center">This is the next page.</h3>
  <p align="center">
    <a href="about.html">about</a>
    <br>
    <a href="index.html">index</a>
  </p>
</body>
</html>
```

Using the Bitvise SSH client

Step1: If not installed, then go to the website https://www.bitvise.com/ssh-client-download , download the installer, run the installer when downloaded. Now the bitvise client is ready.

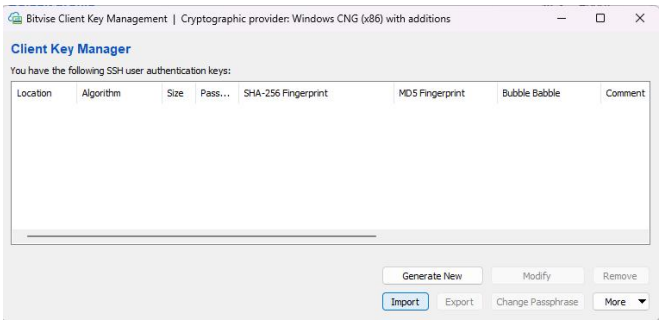
It is useful for connecting to the server of the ec2 service and install necessary tools to proceed with (however that can also be done directly from the ec2 service website). Moreover, the Bitvise client provides an SFTP client, through which we can transfer files from the local machine to the remote machine, just by simple drag and drop.

Step2: Open the Bitvise client, and paste the public IPv4 address copied from the EC2 website in the 'Host' field.

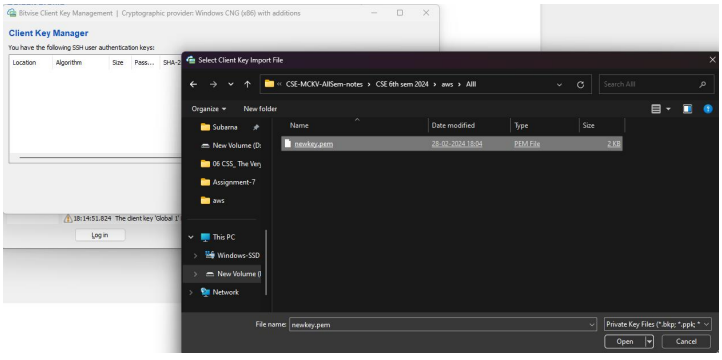


Step3: Then click on the Client key manager.A new window gets opened.

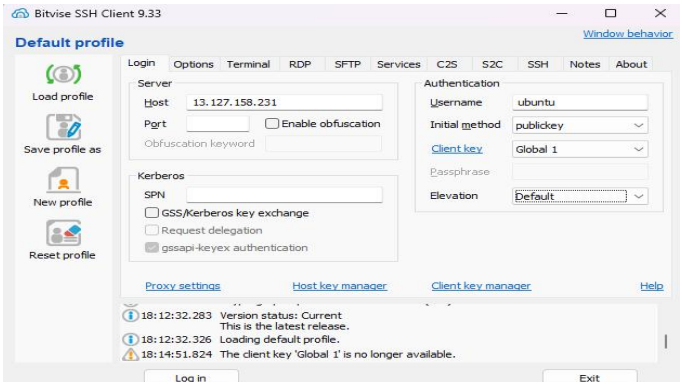
Click on the Import button, and using the file picker, select the downloaded .pem file(key pair file). Then leaving all settings default click on Import.



After Import, the window looks like this:



Then just close the window.

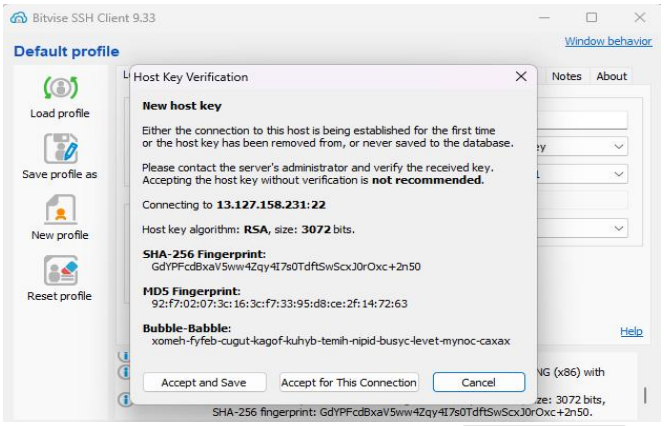


Step4: Then set username as **ubuntu**.

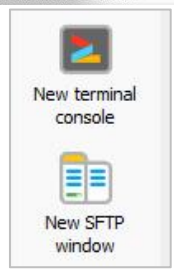
Step5: Then, select Client key as Global 1

Step6: Now click on the Log in button

Step7: On the popup that appears, select Accept and Save button.

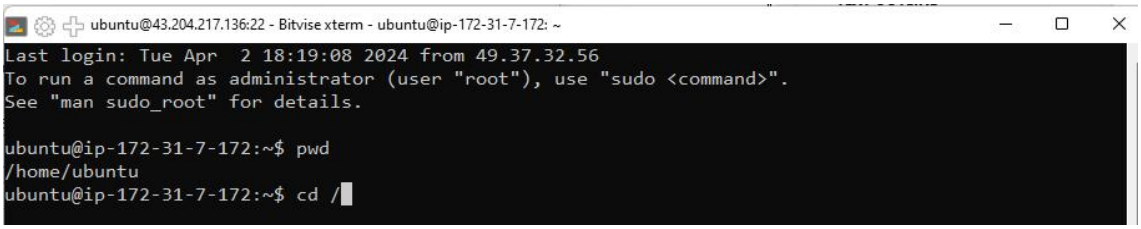


Step8: This makes the connection to the server. Now, two new options open in the Bitvise Client; **New terminal Console** and **New SFTP window.**



Installing the nginx server and transferring website files to the remote server

Step1: In the Bitvise Client (where we just logged in), open the New terminal console.
We can see the current working directory using the pwd command. It shows /home/ubuntu. This confirms the connection with the server, and we have open the terminal in the home directory of the server



Step2: These commands are to be run on this terminal.

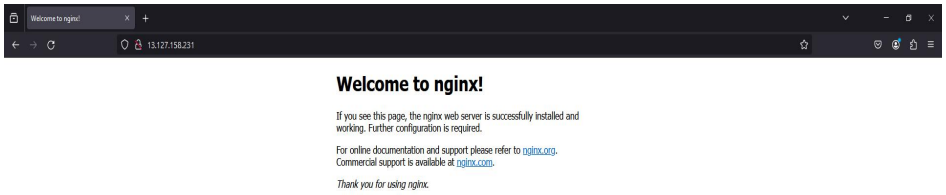
sudo apt-get update

sudo apt-get upgrade (-> Press y when asked, and then in the window that opens, select OK and Press 'Enter'.)

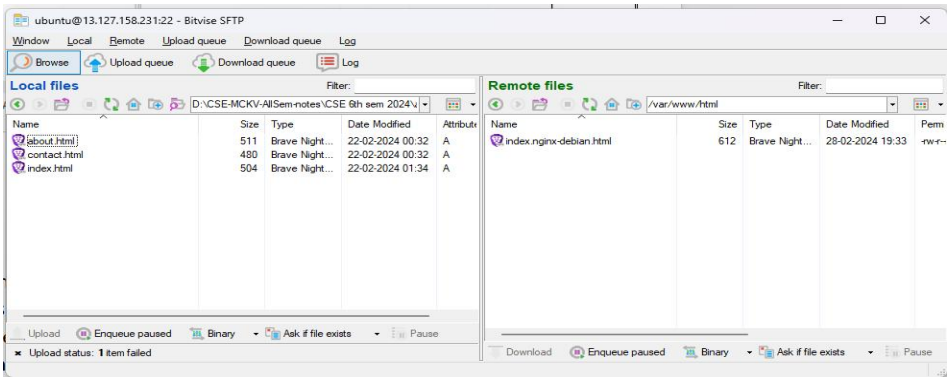
sudo apt-get install nginx (-> Press y when asked, and then in the window that opens, select OK and Press 'Enter'.)

These will update and upgrade the ubuntu server, and install nginx server on the remote ubuntu machine.

Step3: Now open the public IPv4 address in any browser. It shows 'Welcome to nginx!', which confirms the successful installation of nginx on the ec2 remote machine.



Step4: Minimize the terminal window, and maximize the Bitvise SSH client window. Open New SFTP window.



Step5: In the Local files section, open the folder where the static website resides in our computer. (Local files means the files existing on our client machine).

Step6: Go to the root directory in the Remote files section. (Remote files means the files existing in the remote server directory.

Step7: Then open 'var' directory. Then inside it, open the 'www' directory. Then inside it, open the 'html' directory. This is the directory where we shall keep the html files of our static website, for hosting. We see that the html directory is denying the uploading of files, because it does not have the appropriate permissions.

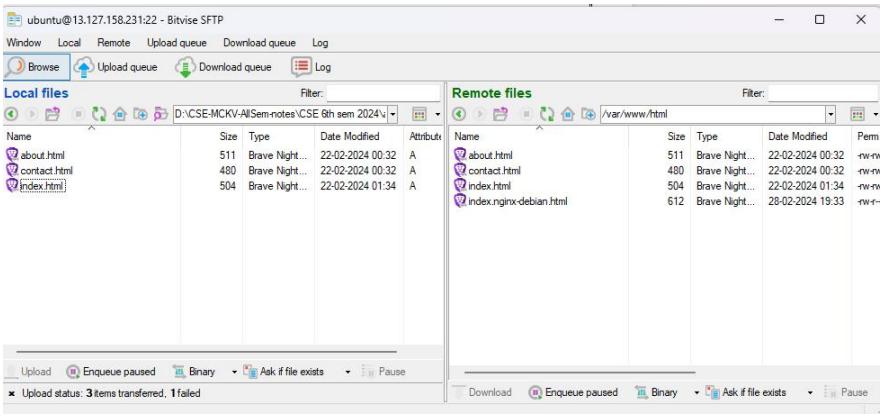
Step8: To add the permissions, run the following commands in the terminal console (the one that was used for installation of the nginx).

cd /var/www (-> This changes the directory to /var/www.)

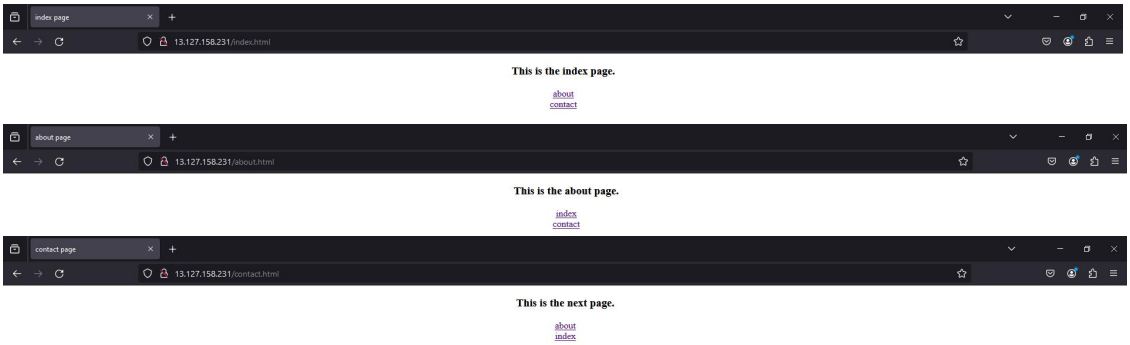
sudo chmod 777 html (-> This gives read, write, and execute permissions for all users, to the html directory.)

```
Setting up fontconfig-config (2.13.1-4.2ubuntu5) ...
Setting up libnginx-mod-stream (1.18.0-6ubuntu14.4) ...
Setting up libtiff5:amd64 (4.3.0-6ubuntu0.8) ...
Setting up libfontconfig1:amd64 (2.13.1-4.2ubuntu5) ...
Setting up libnginx-mod-stream-geoip2 (1.18.0-6ubuntu14.4) ...
Setting up libgd3:amd64 (2.3.0-2ubuntu2) ...
Setting up libnginx-mod-http-image-filter (1.18.0-6ubuntu14.4) ...
Setting up nginx-core (1.18.0-6ubuntu14.4) ...
 * Upgrading binary nginx
Setting up nginx (1.18.0-6ubuntu14.4) ...
Processing triggers for ufw (0.36.1-4ubuntu0.1) ...
Processing triggers for man-db (2.10.2-1) ...
Processing triggers for libc-bin (2.35-0ubuntu3.6) ...
Scanning processes...
Scanning candidates...
Scanning linux images...

Running kernel seems to be up-to-date.

Restarting services...
Service restarts being deferred:
 /etc/needrestart/restart.d/dbus.service
 systemctl restart getty@tty1.service
 systemctl restart networkd-dispatcher.service
 systemctl restart systemd-logind.service
 systemctl restart unattended-upgrades.service
 systemctl restart user@1000.service

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-45-147:/var$ ls
backups  cache  crash  lib  local  lock  log  mail  opt  run  snap  spool  tmp  www
ubuntu@ip-172-31-45-147:/var$ cd www
ubuntu@ip-172-31-45-147:/var/www$ ls
html
ubuntu@ip-172-31-45-147:/var/www$ cd html
ubuntu@ip-172-31-45-147:/var/www/html$ ls
index.nginx-debian.html
ubuntu@ip-172-31-45-147:/var/www/html$ cd ..
ubuntu@ip-172-31-45-147:/var/www$ sudo chmod 777 html
ubuntu@ip-172-31-45-147:/var/www$
```

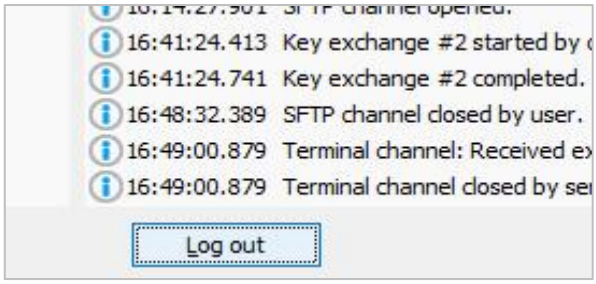
Step9: Now maximize the SFTP window, and drag and drop the files from the Local files to the Remote files section.



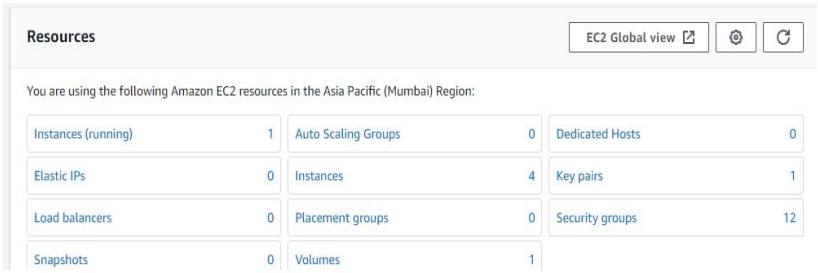
Step10: Open the public IPv4 address (from the ec2 instance), again, on any web browser. Now our website is visible, through this IP. So, finally our website is hosted on the EC2 service.



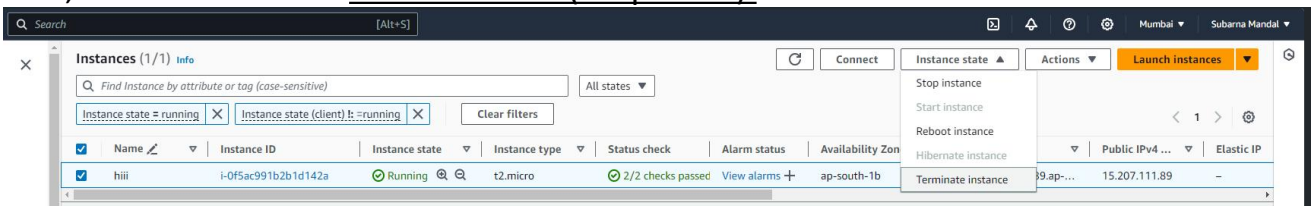
Step11: Now close the SFTP window, and the terminal console.
Step12: Log out of the remote server, from the Bitvise Client,by pressing the Log out button.
Then close the Bitvise Client.



Step13: We know that AWS services are not free forever. They have a limited time free-tier, after which there are charges to be imposed.
So, if we want to keep this website alive, we need to pay for the server.Since, this is for testing purpose, we will just terminate this I nstance, so that we don't face any unwanted service charges.
Go to the AWS console.
Search for ec2; Select the Instances (running).



The new window that opens, shows all the running instances. Select the instance that you created, and then click on Instance state (dropdown).



Click on Terminate instance. On the popup window that opens, click on the Terminate button.
After few moments, we see that the it does not show Running anymore. And, after some hours, the instance gets deleted automatically.