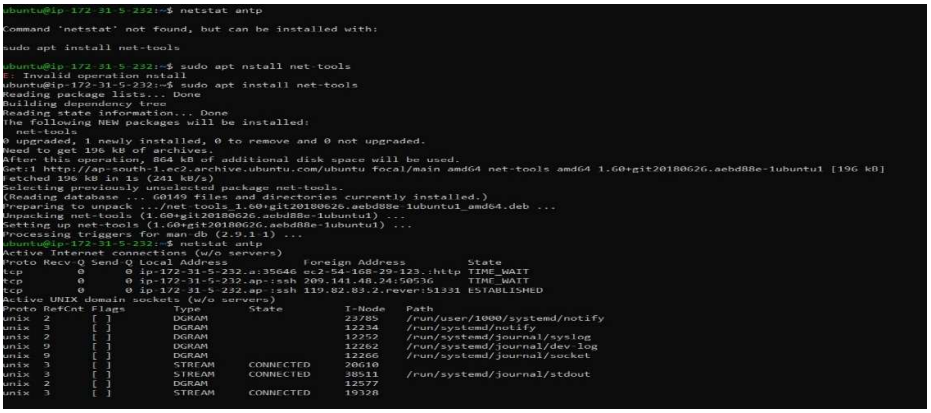
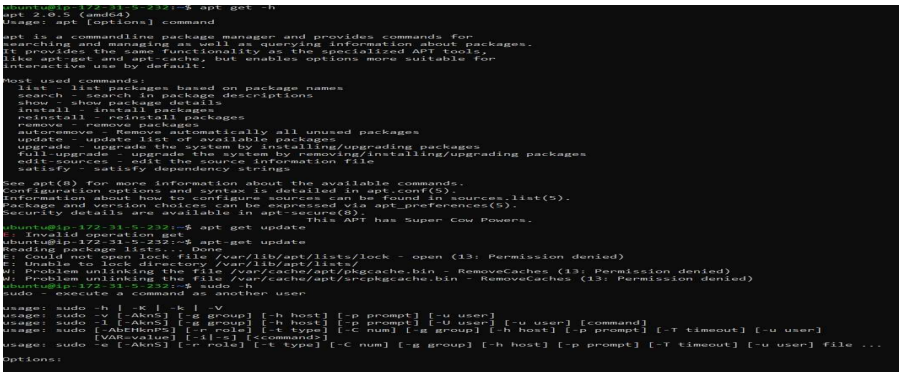
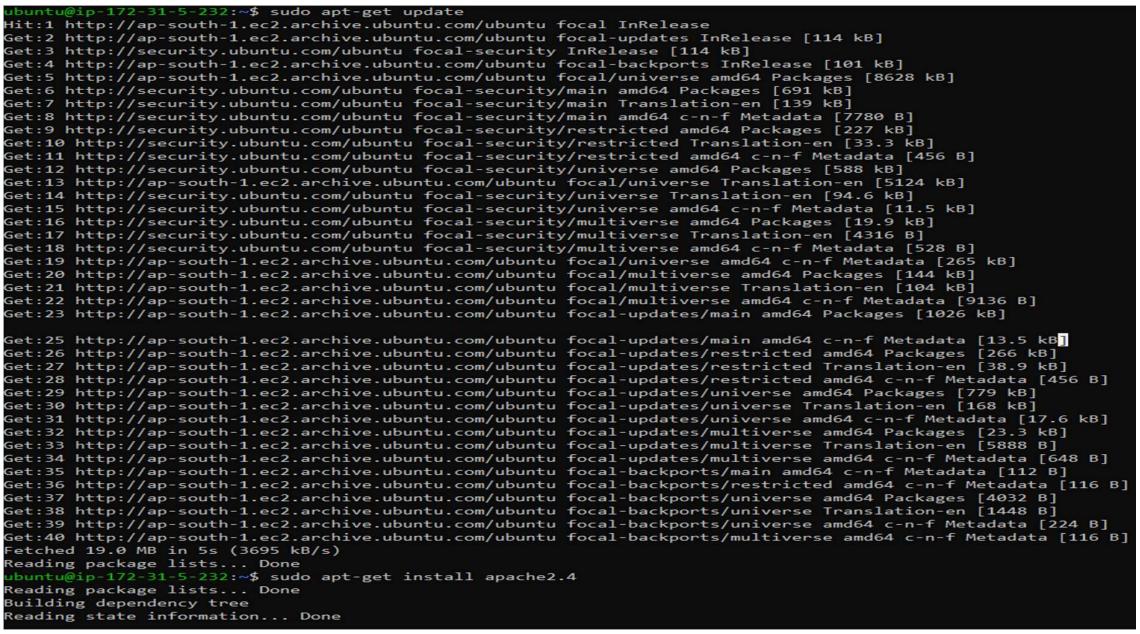






Go to this link:-

https://linuxbeast.com/tutorials/aws/install-apache2-on-amazon-ec2-ubuntu-18-04/

Installation & Process Link :-

https://www.techrepublic.com/article/how-to-use-the-apache-web-server-to-install-and-configure-a-website/

Step 1. Installing Apache2 Web Server

First, SSH remote into your EC2 instance using your great Linux terminal console and add the Apache2 system packages below, type command:

```
sudo add-apt-repository ppa:ondrej/apache2Copy
```

Then press `Enter` on the screen to continue.

Update the packages currently installed on the system, type command:

```
sudo apt updateCopy
```

Now install Apache2 Web Server, type command:

```
sudo apt-get install apache2 -yCopy
```

After installation completed, verify the Apache2 version.

```
sudo apache2ctl -v
```

The output looks like this:

```
Server version: Apache/2.4.41 (Ubuntu)

Server built:   2019-04-02T20:30:26Copy
```

Step 2. Testing Apache2 Web Server

After the installation process, the Apache2 service will runs automatically. To make sure the Apache2 is active (`Running`) online, type command:

```
sudo service apache2 statusCopy
```

The output looks like this:

We assume that Apache2 is already install and active. Next, You need to verify the default Apache2 landing page in the web browser if the Apache2 web server is working properly.

Note: Make sure HTTP Port 80 is publicly allowed from the incoming traffic in security group on your EC2 instance.

```
ubuntu@ip-[redacted]:~$ sudo service apache2 status
● apache2.service - The Apache HTTP Server
     Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor preset: enabled)
     Active: active (running) since Mon 2021-06-07 09:25:17 UTC; 46s ago
       Docs: https://httpd.apache.org/docs/2.4/
   Main PID: 14513 (apache2)
      Tasks: 55 (limit: 1160)
     Memory: 5.2M
     CGroup: /system.slice/apache2.service
             ├─14513 /usr/sbin/apache2 -k start
             ├─14516 /usr/sbin/apache2 -k start
             └─14517 /usr/sbin/apache2 -k start

Jun 07 09:25:17 ip-[redacted] systemd[1]: Starting The Apache HTTP Server...
Jun 07 09:25:17 ip-[redacted] systemd[1]: Started The Apache HTTP Server.
```

To get the public IP address on your server, type command:

```
curl http://169.254.169.254/latest/meta-data/public-ipv4Copy
```

The output looks like this:

```
52.XX.XX.XXCopy
```

Open your web browser and paste your public IP address.

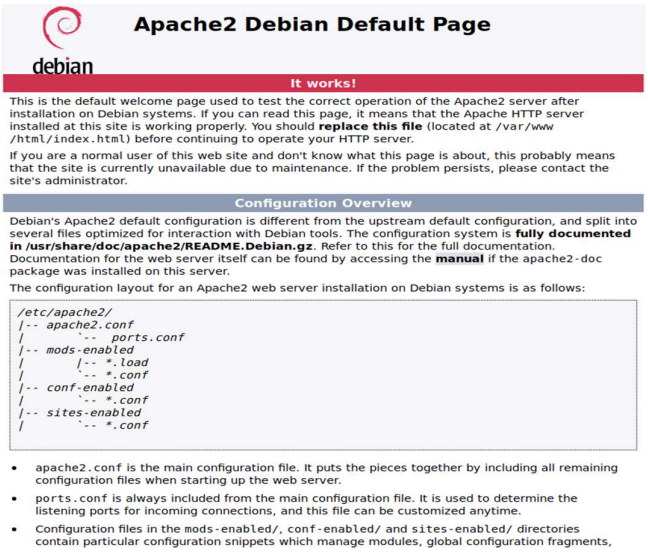
```
http://52.XX.XX.XXCopy
```

The output looks like this:

Open your web browser and paste your public IP address.

```
http://52.XX.XX.XXCopy
```

The output looks like this:



Apache2 Debian Default Page

debian

It works!

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Debian systems. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should **replace this file** (located at `/var/www/html/index.html`) before continuing to operate your HTTP server.

If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

Configuration Overview

Debian's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Debian tools. The configuration system is **fully documented in /usr/share/doc/apache2/README.Debian.gz**. Refer to this for the full documentation. Documentation for the web server itself can be found by accessing the **manual** if the apache2-doc package was installed on this server.

The configuration layout for an Apache2 web server installation on Debian systems is as follows:

```
/etc/apache2/
|-- apache2.conf
|       `--  ports.conf
|-- mods-enabled
|       |-- *.load
|       `-- *.conf
|-- conf-enabled
|       `-- *.conf
|-- sites-enabled
|       `-- *.conf
```

- `apache2.conf` is the main configuration file. It puts the pieces together by including all remaining configuration files when starting up the web server.
- `ports.conf` is always included from the main configuration file. It is used to determine the listening ports for incoming connections, and this file can be customized anytime.
- Configuration files in the `mods-enabled/`, `conf-enabled/` and `sites-enabled/` directories contain particular configuration snippets which manage modules, global configuration fragments,

Step 3. Apache2 Commands

You can manage your Apache2 web server common commands:

```
# Start Apache service
sudo service apache2 start
# Stop Apache service
sudo service apache2 stop
# Restart Apache service
sudo service apache2 restart
# Reload Apache service without dropping connections
sudo service apache2 reload
# Enable Apache service on startup boot
sudo systemctl enable apache2
# Disable Apache service
sudo systemctl disable apache2
```

What is that page Apache is serving up? If you look in /var/www/html, you'll find the index.html file--let's change it.

Back at the terminal window, rename that index.html file with the command:

```
sudo mv /var/www/html/index.html /var/www/html/index.html.bak
```

Now, let's create a new welcome file. Issue the command:

```
sudo nano /var/www/html/index.html
```

In that file, paste the following:

```
<!DOCTYPE html>
<html>
<body>
<h1>Hello, Tech Republic!</h1>
<p>How are you doing?</p>
</body>
</html>
```

Save and close the file. Reload the web page in your browser and you should see the change

Hello, TechRepublic!

How are you doing?

Our new index.html page is being served by Apache.

How to create a site for Apache

What we're going to do now is create a virtual host for Apache to serve up. A virtual host is a fancy name for a website that's served by Apache. You can have numerous virtual hosts served up on a single Apache server. In fact, you are only limited to the power of your hosting server and the bandwidth of your network.

So let's create a virtual host called *test*.

The first thing we're going to do is create a directory to house test with the command:

```
sudo mkdir -p /var/www/html/test
```

Next, we'll give the new directory the proper ownership with the command:

```
sudo chown -R $USER:$USER /var/www/html/test
```

Finally, we'll grant the proper permissions with the command:

```
sudo chmod -R 755 /var/www/html/test
```

Copy our new index.html file into the test directory with the command:

```
sudo cp /var/www/html/index.html /var/www/html/test/
```

Now we have to create the virtual host configuration so Apache knows where test is. This will be housed in /etc/apache/sites-available. To do that we'll create the test.conf file with the command:

```
sudo nano /etc/apache2/sites-available/test.conf
```

In that file paste the following:

```
<VirtualHost *:80>

    ServerAdmin admin@example.com

    ServerName example.com

    ServerAlias www.example.com

    DocumentRoot /var/www/html/test

    ErrorLog ${APACHE_LOG_DIR}/error.log

    CustomLog ${APACHE_LOG_DIR}/access.log combined

</VirtualHost>
```

The most important line above begins with *DocumentRoot*, as that instructs Apache where the files for the virtual host will be found. Save and close that file.

At this point, we've created the directory to house the files, given it the proper ownership and permissions, and created a configuration for the virtual host. However, Apache is still not aware of the new site. Why? Because the configuration file lives in *sites-available*. What we have to do is create a link from that configuration into the */etc/apache2/sites-enabled* directory. Only those configurations found in *sites-enabled* are active on the Apache server.

On non-Ubuntu servers, you have to use the *ln* (for *link*) command to do this. However, on Ubuntu there's a handy utility that will create that site for you. Said utility is *a2ensite*. If we run the command:

```
sudo a2ensite test.conf
```

Our test virtual host will then be enabled.

After that command succeeds, you then must reload Apache (which will only reload the configuration files, not restart the web server) with the command:

```
sudo systemctl reload apache2
```

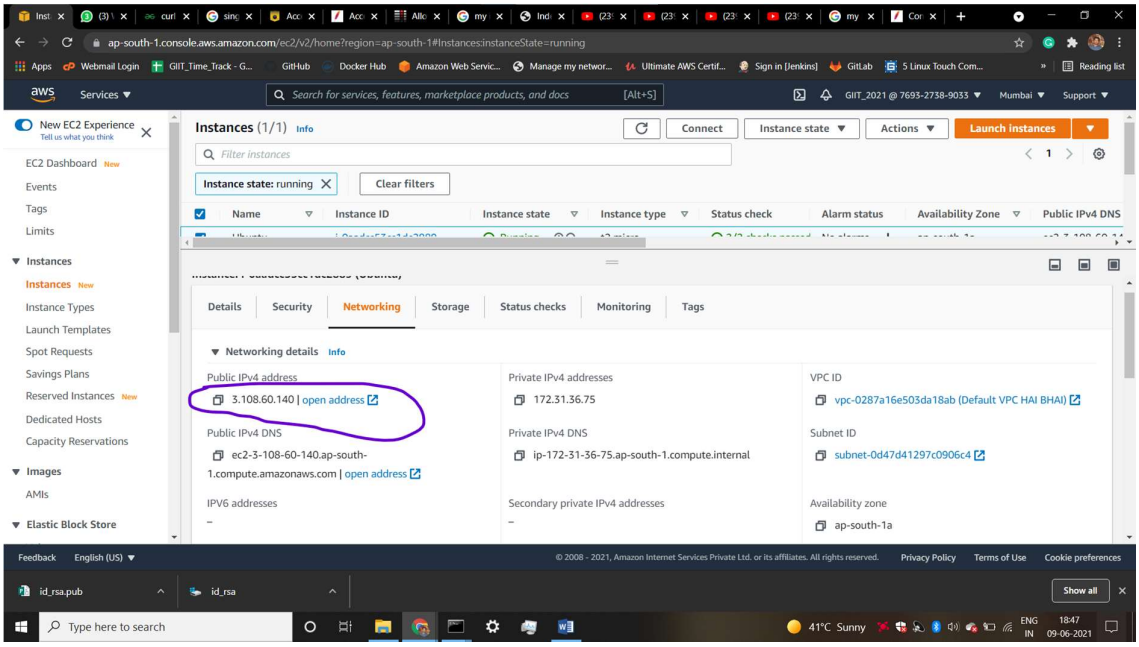
Now, if you point your browser to *http://SERVER_IP/test* (where SERVER_IP is the IP address of the server) you should see the same Hello, TechRepublic welcome as you did with the basic index.html file, only it's being served from our newly-created virtual host.

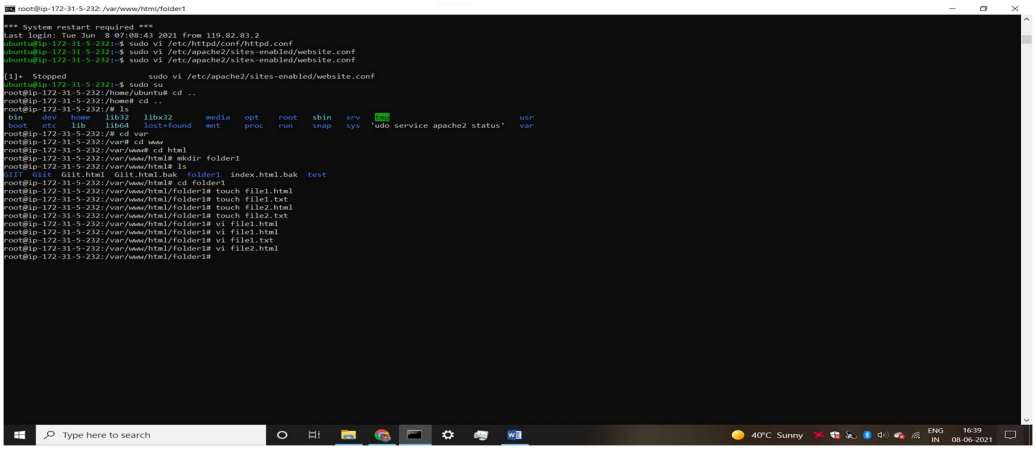
You've just installed the Apache web server, edited the index.html file, and then created your very own virtual host. You can take this simple how-to and use it as a basis for spinning up all the Apache-served websites you need.

➔ If we want to create a multiple page so we can create new file using same command only file name will be changed

Allow access to Apache server from only one IP address

https://serverfault.com/questions/776252/allow-access-to-apache-server-from-only-one-ip-address/776256

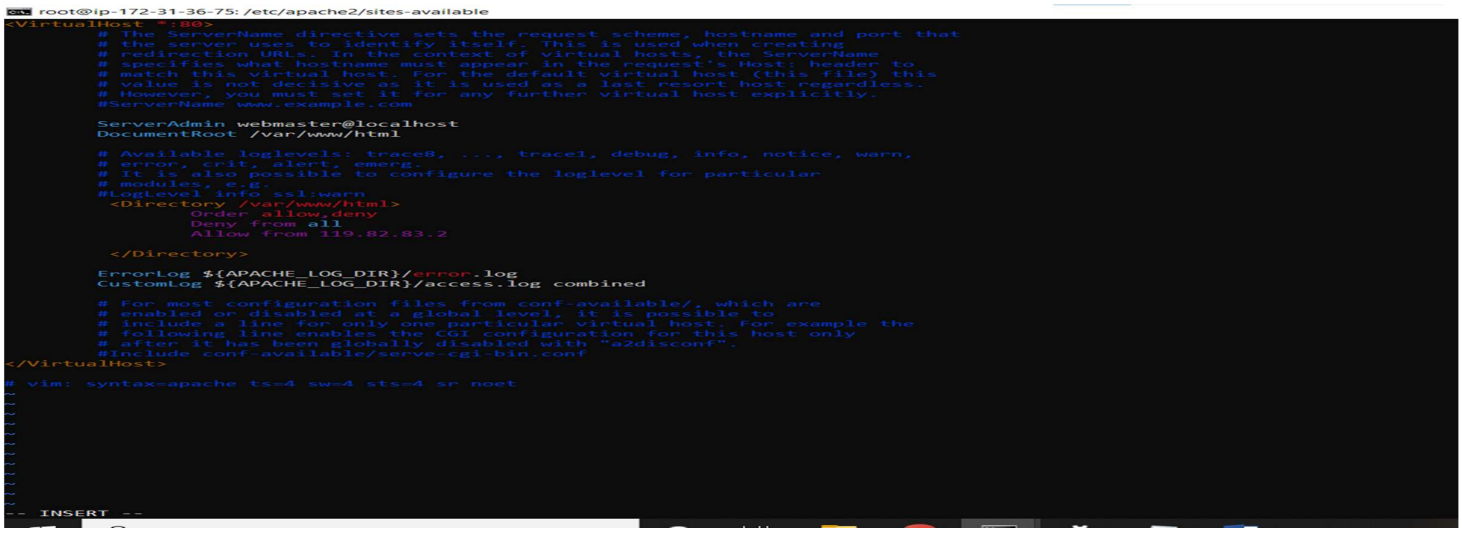




root@ip-172-31-5-232:/# vi /etc/apache2/sites-available/000-default.conf

root@ip-172-31-36-75: /etc/apache2/sites-available

```
root@ip-172-31-36-75:/home/ubuntu# cd..
cd..: command not found
root@ip-172-31-36-75:/home/ubuntu# cd ..
root@ip-172-31-36-75:/home# cd etc/apache2/sites-available
bash: cd: etc/apache2/sites-available: No such file or directory
root@ip-172-31-36-75:/home# cd ..
root@ip-172-31-36-75:/# cd ..
root@ip-172-31-36-75:/# cd etc/a
acpi/          alternatives/ apparmor/     apport/       at.deny
adduser.conf   apache2/      apparmor.d/   apt/
root@ip-172-31-36-75:/# cd etc/ap
apache2/    apparmor/   apparmor.d/ apport/     apt/
root@ip-172-31-36-75:/# cd etc/apache2/sites-available/
root@ip-172-31-36-75:/etc/apache2/sites-available# ls
000-default.conf  default-ssl.conf  test.conf
root@ip-172-31-36-75:/etc/apache2/sites-available# vi 000-default.conf
root@ip-172-31-36-75:/etc/apache2/sites-available#
```



After this restart apache 2 server