# AWS RHEL 8 EC2 Nginx Install

In this tutorial, we will be installing the Nginx web server on an AWS RHEL 8 compute instance.

## Prerequisites

- AWS RHEL 8 EC2 instance
- an AWS Free Tier account
- internet access

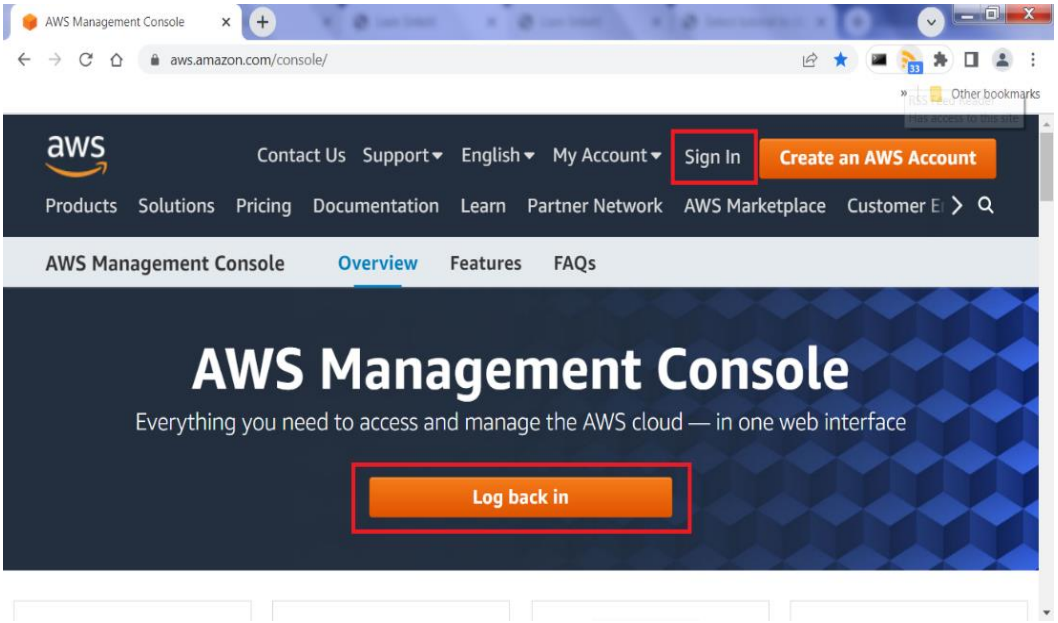
If you do not have an AWS account, you can access my **AWS Create Free Tier Account** tutorial **here**.

If you do not have an AWS RHEL 8 EC2 instance, my tutorial **Create AWS RHEL 8 EC2 Instance** is **here**.

Steps to complete tutorial:

- Start RHEL 8 EC2 Instance
- Update Virtual Private Cloud (VPC)
- Connect to RHEL 8
- Install Nginx Web Server
- Configure Nginx to Host a Website

To begin, go to the following website, https://aws.amazon.com/console/ and log in to the console.

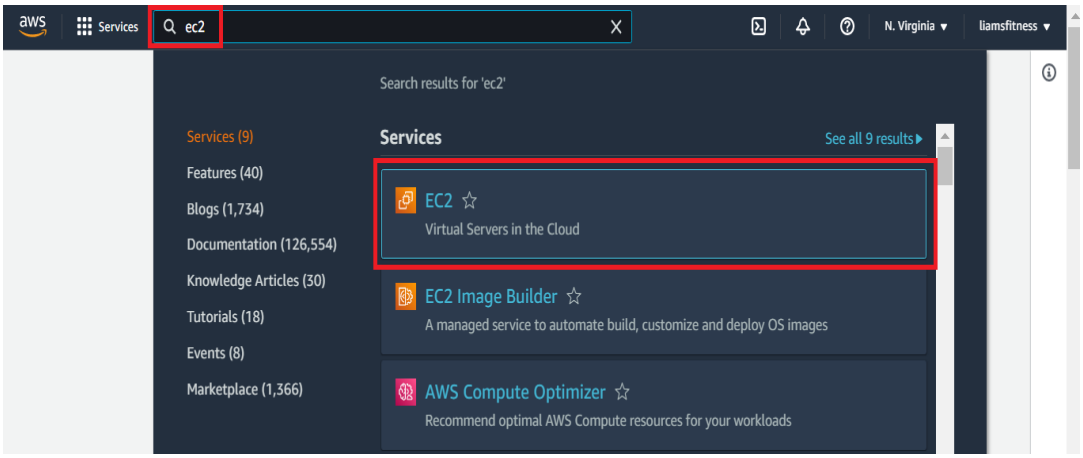



## Start RHEL 8 EC2 Instance

At the end of my **Create AWS RHEL 8 EC2 Instance** tutorial, I suggested shutting down your EC2 instances when they are not being used due to the AWS Free Tier EC2 limit of 750 hours per month.

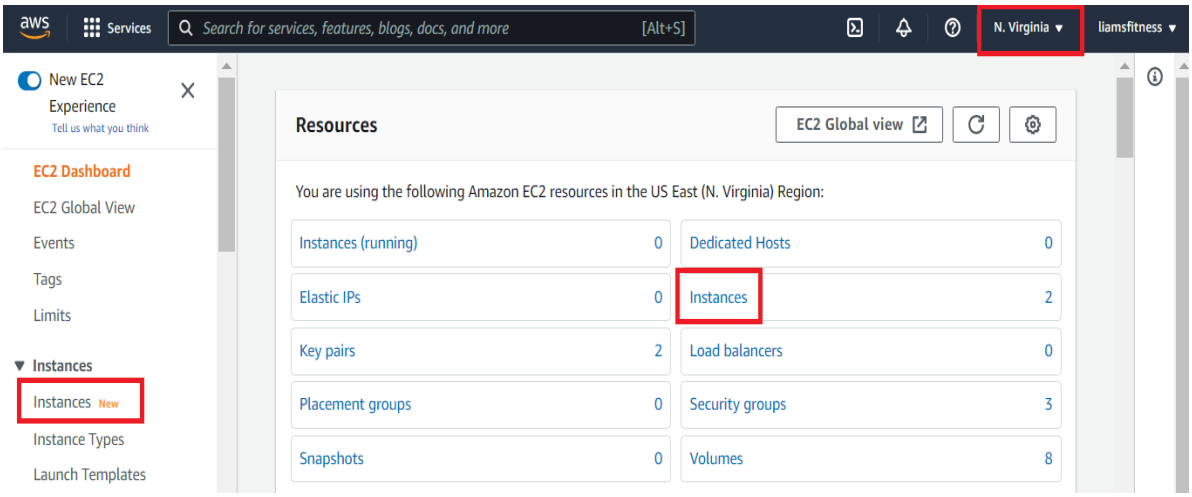
If you already know that your RHEL 8 EC2 instance is running, you can skip this step and go directly to **Update Virtual Private Cloud (VPC)**.

Otherwise, we need to ensure that our instance is running. Once on the **Console Home** screen, enter **EC2** in the search bar and select the 1st EC2 listing.

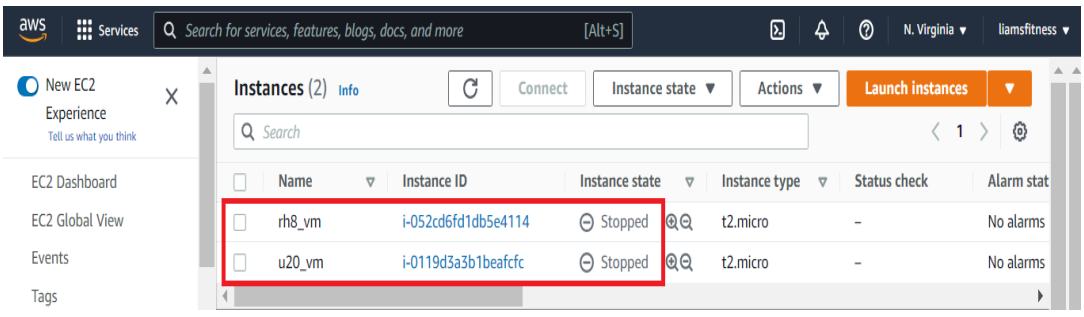



You will be brought to the **EC2** Dashboard. It contains links to the resources being used in the selected AWS region. In my case it's US East (N. Virginia).

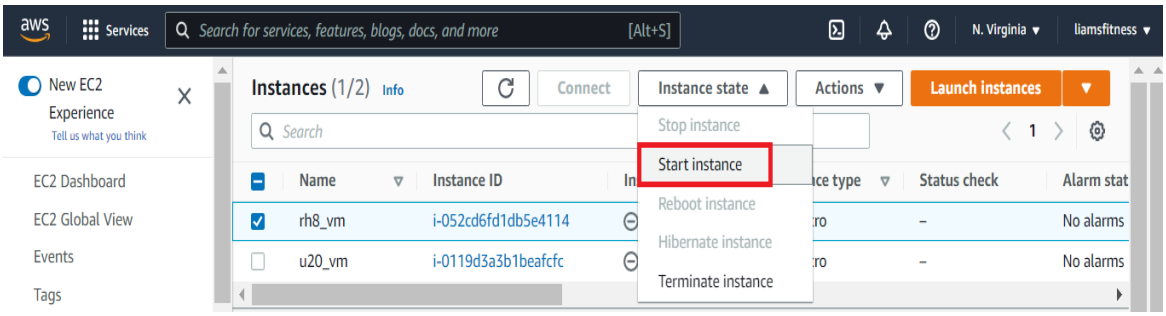
From the EC2 dashboard, click **Instances** (either link will work, your choice).



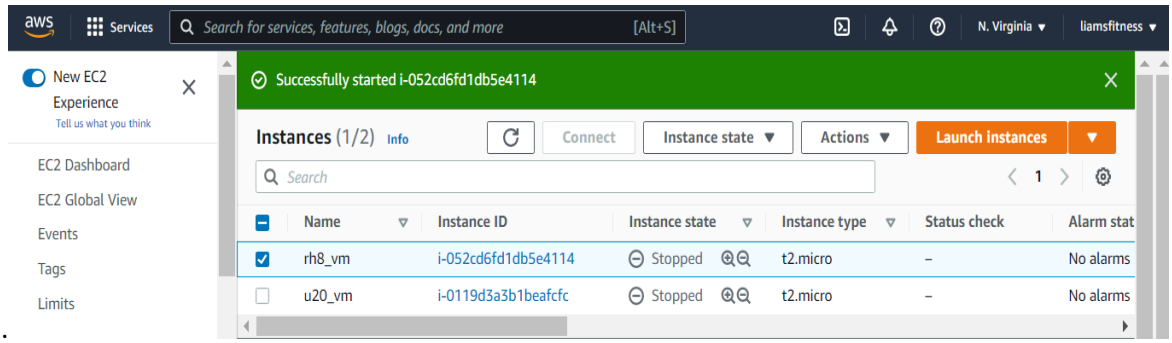

You will notice that both of my instances are in the stopped stated.



To start my RHEL 8 instance, I will ensure that my **rh8_vm** is selected (checkbox), then I will click the **Instance state** drop-down menu and click **Start instance**.

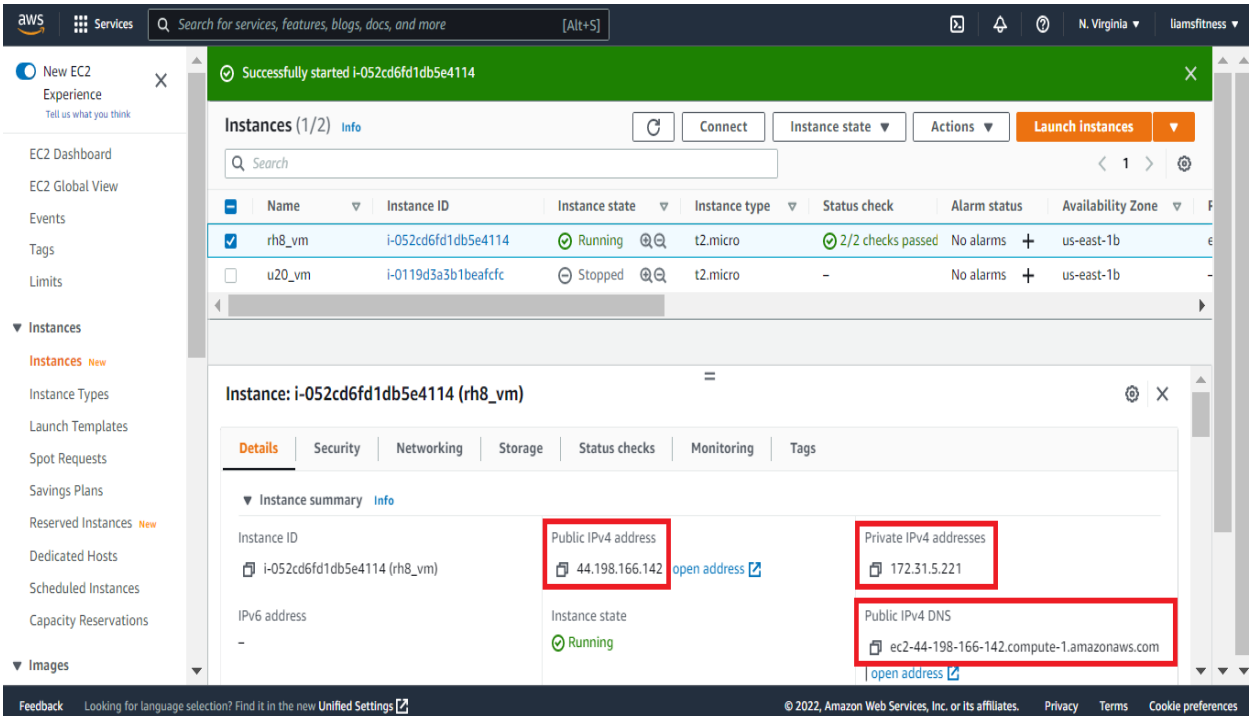



The instance will start and be given new connection details.

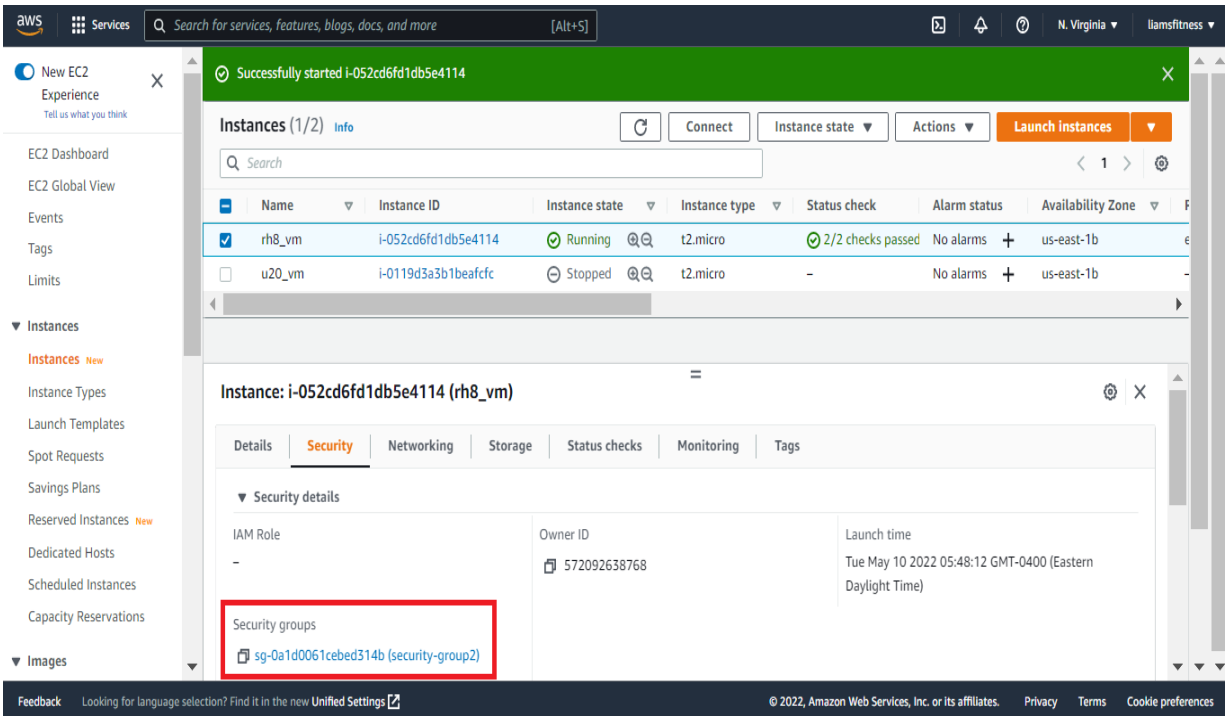



.

Ensure your new instance is selected (**rh8_vm**), then on the **Details** tab, note the value for **Public IPv4 DNS**. I usually keep the instance's name, public IP, private IP and public IPv4 DNS stored for easy access. We will need this to connect to the instance.



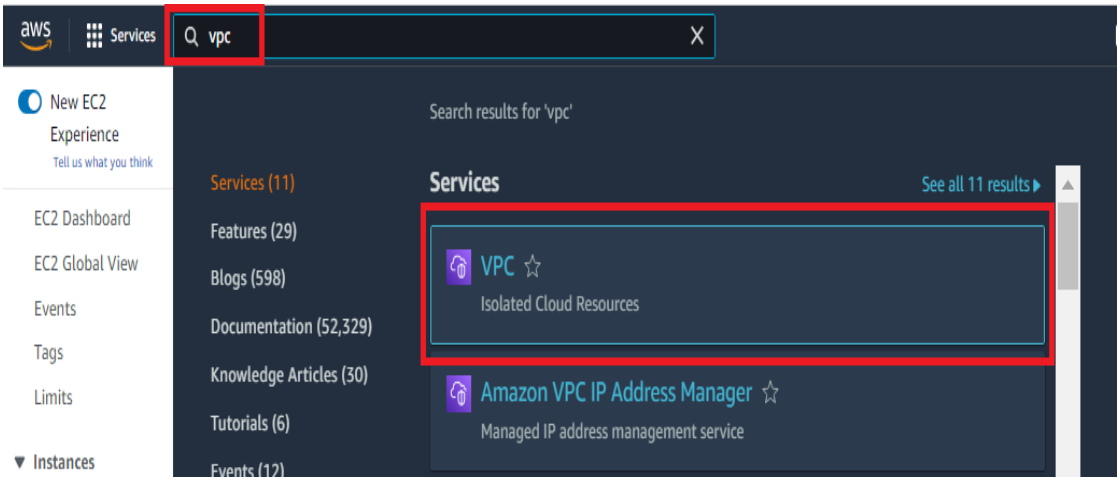
Next, click the **Security** tab and note the Security Group associated with the RHEL 8 EC2 instance. In my case, it is **security-group2**.



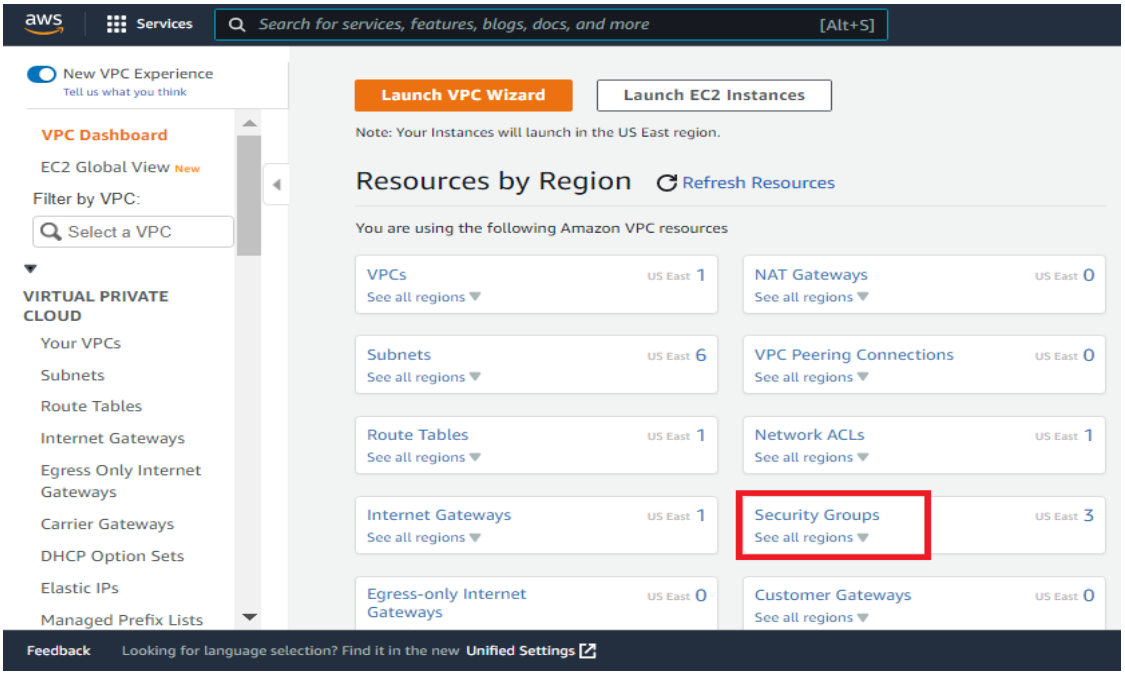

Before we install Nginx, we will need to open a port in our firewall to be able to access the website we create after the Nginx install. To do this we will configure our VPC (Virtual Private Cloud) Security Group to allow HTTP traffic through the firewall. This will allow access to the Nginx web server that will be running on our RHEL 8 EC2 instance in our isolated virtual network (VPC).

## Update Virtual Private Cloud (VPC)

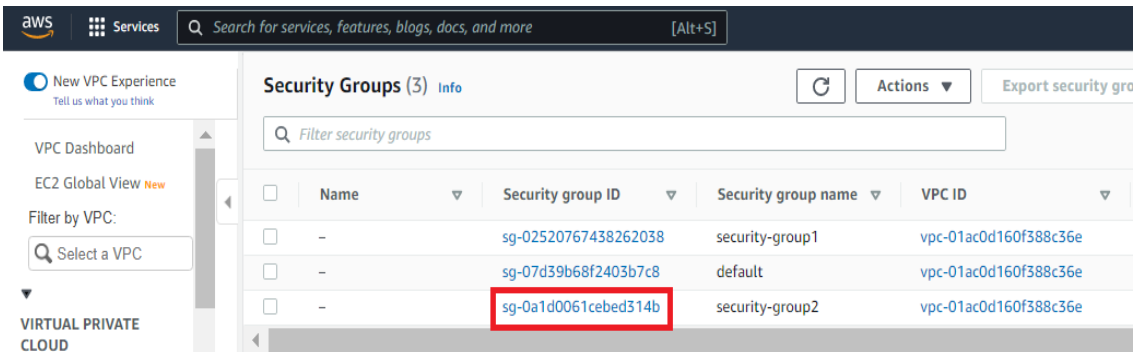
From the EC2 dashboard, enter VPC in the search bar and select **VPC** Isolated Cloud Resources.



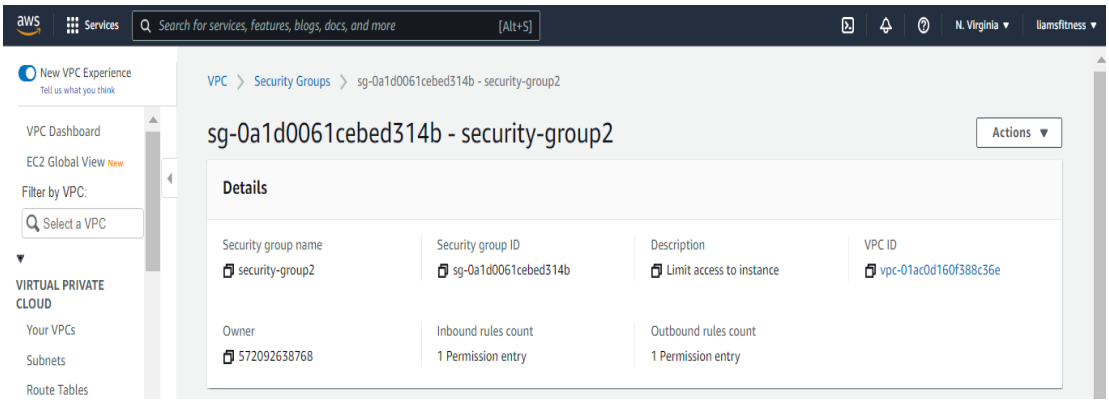
On the VPC Dashboard, select **Security Groups**.

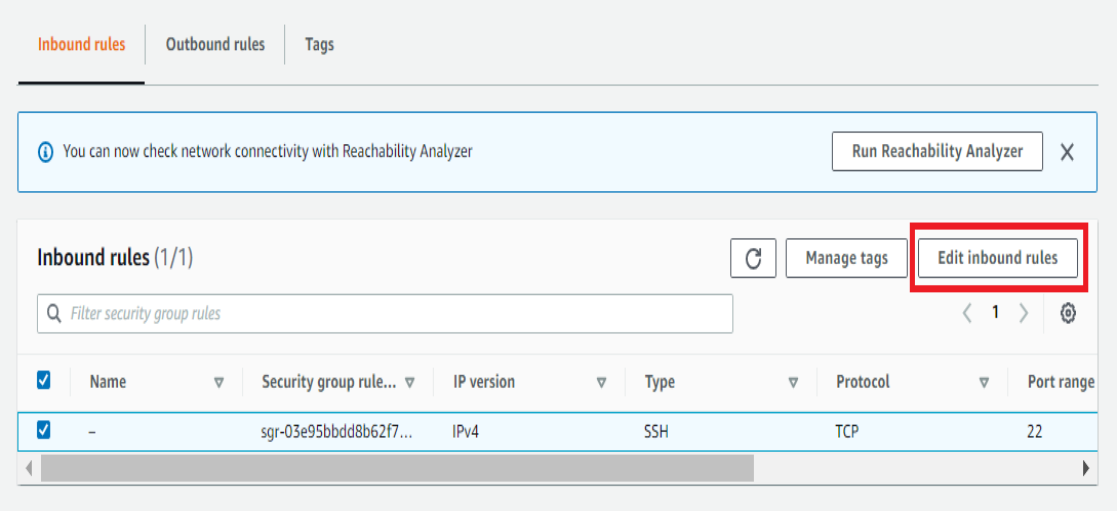



On the **Security Groups** screen, click the **Security group ID** that is associated with your RHEL 8 EC2 instance.

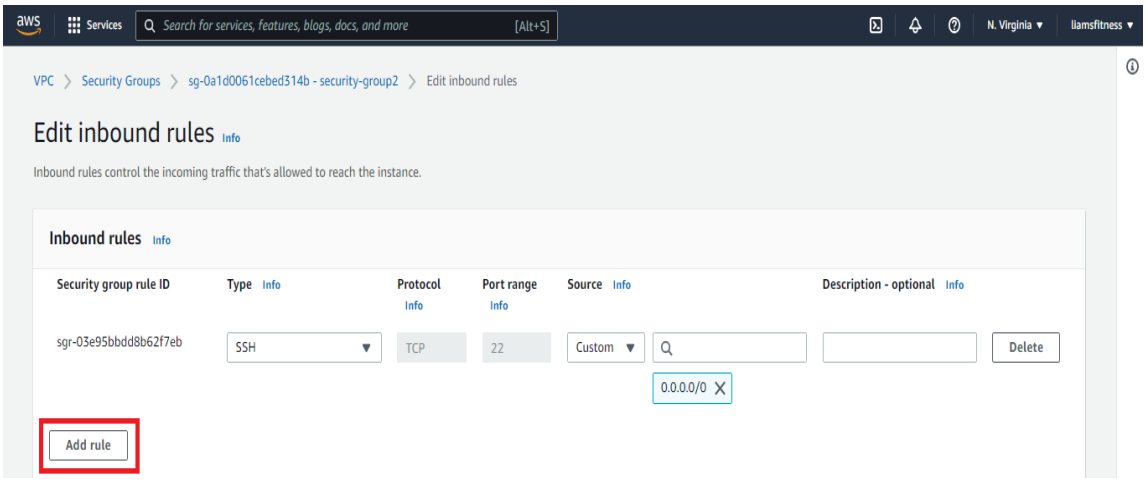



On the security group screen, scroll down and ensure the **Inbound Rules tab** is selected, then click **Edit Inbound Rules**

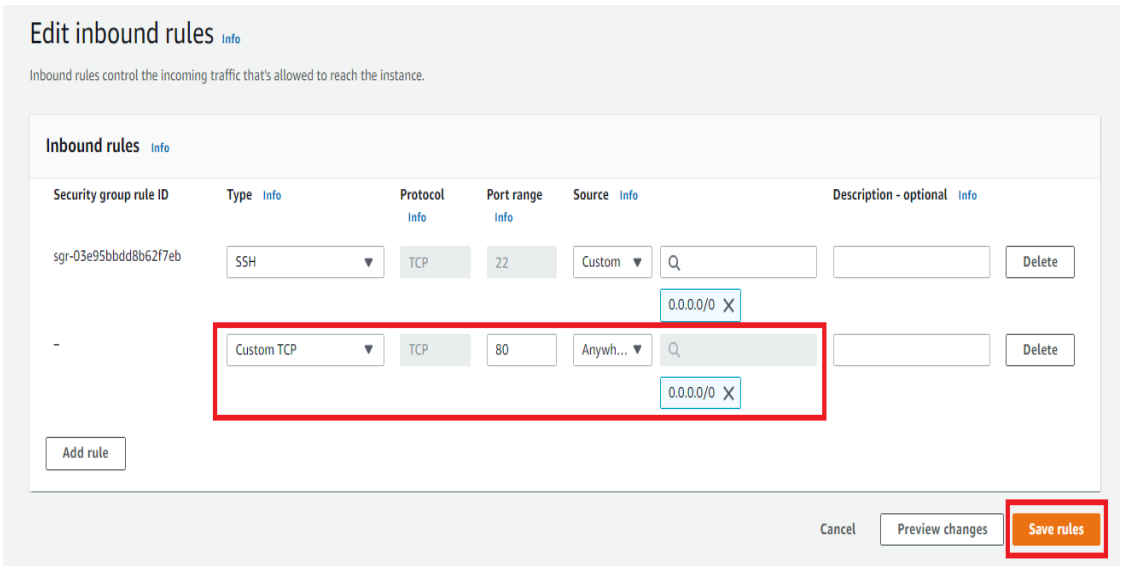


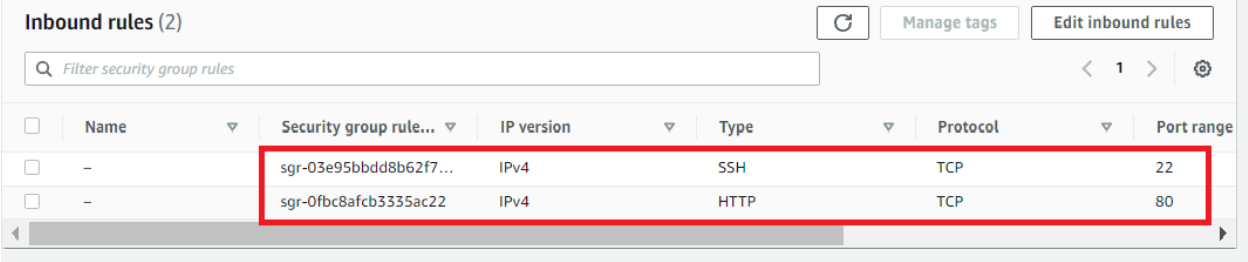



On the Edit Inbound Rules screen, click the **Add rule** button




Ensure **Port range** is set to **80** (HTTP) and **Source** is set to **0.0.0.0/0** (Anywhere-IPv4) and click **Save rules**



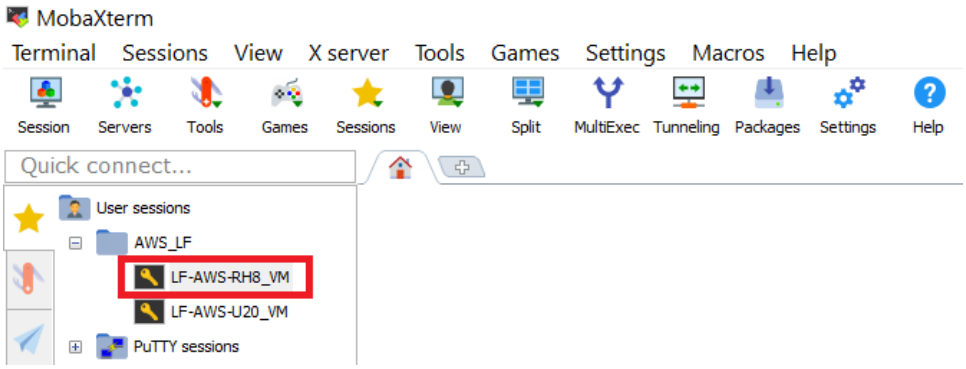



Now that we've opened the firewall for HTTP access, we can connect to the RHEL 8 instance and install the Nginx web server.

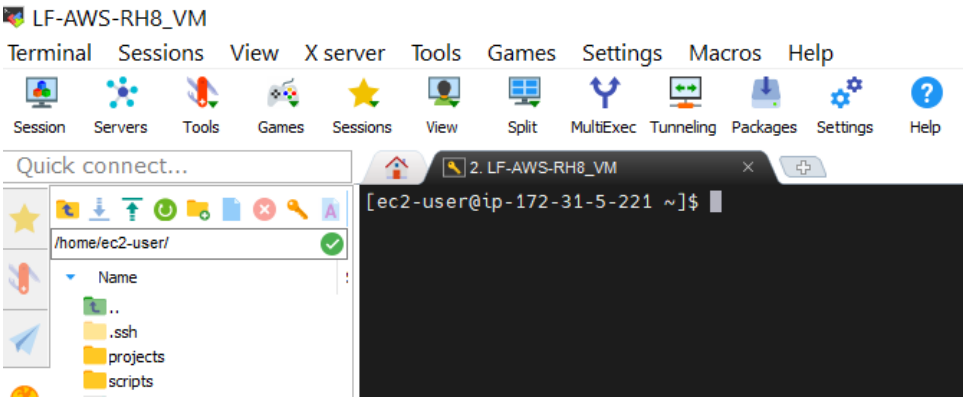
## Connect to RHEL 8

As I mentioned in my **Create AWS RHEL 8 EC2 Instance** tutorial, accessible **here,** I am using **MobaXterm Portable** as my SSH client. Since I restarted my RHEL 8 EC2 instance, the connection details changed and I had to update my saved SSH session by changing the **Remote Host:** to the updated **Public IPv4 DNS** of my RHEL 8 instance.

After updating my saved SSH connection, from the MobaXterm main interface, I double clicked my SSH session to connect to my RHEL 8 EC2 instance.




After the session opened, I executed the **`clear`** command to clear the terminal.




We are now ready to install the Nginx web server.

## Install Nginx Web Server

Please note that your RHEL 8 instance **must** be registered, and have access to a Red Hat subscription, in order to install packages. RHEL 8 registration was covered in my **Create AWS RHEL 8 EC2 Instance** tutorial, accessible **here**. To verify system registration, issue the following:

```
$ sudo subscription-manager list
```

```
[ec2-user@ip-172-31-5-221 ~]$
[ec2-user@ip-172-31-5-221 ~]$ sudo subscription-manager list
+-------------------------------------------+
    Installed Product Status
+-------------------------------------------+
Product Name:   Red Hat Enterprise Linux for x86_64
Product ID:     479
Version:        8.4
Arch:           x86_64
Status:         Subscribed
Status Details:
Starts:         05/10/2022
Ends:           05/10/2023
```

We can now install the Nginx web server by issuing the following command:

```
$ sudo yum install nginx -y
```

```
ubuntu@ip-172-31-4-185:~$ sudo apt install nginx -y
```

By default, the Nginx web server is disabled after installation. We will confirm this with the following:

```
$ sudo systemctl status nginx
```

```
  perl-libs-4:5.26.3-421.el8.x86_64                 perl-macros-4:5.26.3-421.el8.x86_64
  perl-parent-1:0.237-1.el8.noarch                  perl-podlators-4.11-1.el8.noarch
  perl-threads-1:2.21-2.el8.x86_64                  perl-threads-shared-1.58-2.el8.x86_64

Complete!
[ec2-user@ip-172-31-5-221 ~]$ sudo systemctl status nginx
● nginx.service - The nginx HTTP and reverse proxy server
   Loaded: loaded (/usr/lib/systemd/system/nginx.service; disabled; vendor preset: disabled)
   Active: inactive (dead)

May 10 12:50:37 ip-172-31-5-221.ec2.internal systemd[1]: nginx.service: Unit cannot be reloaded because it is inactive.
[ec2-user@ip-172-31-5-221 ~]$
```

We will now start the nginx service and ensure it runs on startup with the following:

```
$ sudo systemctl start nginx
$ sudo systemctl enable nginx
$ sudo systemctl is-enabled nginx
```

```
[ec2-user@ip-172-31-5-221 ~]$
[ec2-user@ip-172-31-5-221 ~]$ sudo systemctl start nginx
[ec2-user@ip-172-31-5-221 ~]$
[ec2-user@ip-172-31-5-221 ~]$ sudo systemctl enable nginx
Created symlink /etc/systemd/system/multi-user.target.wants/nginx.service → /usr/lib/systemd/system/nginx.service.
[ec2-user@ip-172-31-5-221 ~]$ sudo systemctl is-enabled nginx
enabled
[ec2-user@ip-172-31-5-221 ~]$
```

We will now confirm that it is running before performing our test.

**`$ sudo systemctl status nginx`**

```
[ec2-user@ip-172-31-5-221 ~]$
[ec2-user@ip-172-31-5-221 ~]$ sudo systemctl status nginx
● nginx.service - The nginx HTTP and reverse proxy server
   Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; vendor preset: disabled)
   Active: active (running) since Tue 2022-05-10 12:57:37 UTC; 3min 19s ago
 Main PID: 12536 (nginx)
    Tasks: 2 (limit: 4821)
   Memory: 3.6M
   CGroup: /system.slice/nginx.service
           ├─12536 nginx: master process /usr/sbin/nginx
           └─12537 nginx: worker process

May 10 12:57:37 ip-172-31-5-221.ec2.internal systemd[1]: Starting The nginx HTTP and reverse proxy server...
May 10 12:57:37 ip-172-31-5-221.ec2.internal nginx[12532]: nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
May 10 12:57:37 ip-172-31-5-221.ec2.internal nginx[12532]: nginx: configuration file /etc/nginx/nginx.conf test is successful
May 10 12:57:37 ip-172-31-5-221.ec2.internal systemd[1]: nginx.service: Failed to parse PID from file /run/nginx.pid: Invalid argument
May 10 12:57:37 ip-172-31-5-221.ec2.internal systemd[1]: Started The nginx HTTP and reverse proxy server.
[ec2-user@ip-172-31-5-221 ~]$ ^C
[ec2-user@ip-172-31-5-221 ~]$
```

From my googling, I was able to determine that this is a minor bug where there is a race between **/run/nginx.pid** being written to during startup and **systemd** (Linux initialization system and service manager) reading from the file. So, we will proceed with our test.

Since we've already added an inbound rule to allow HTTP traffic, we can test it out by using the **Public IPv4 address** associated with the RHEL 8 EC2 instance.
***(NOTE: your Public IPv4 address will be different)***

## Configure Nginx to Host a Website

Now we will configure Nginx to host a basic website using a Nginx server block. This will override the default test page that was returned when we initially accessed the web server via the public IP address of our AWS RHEL 8 instance.

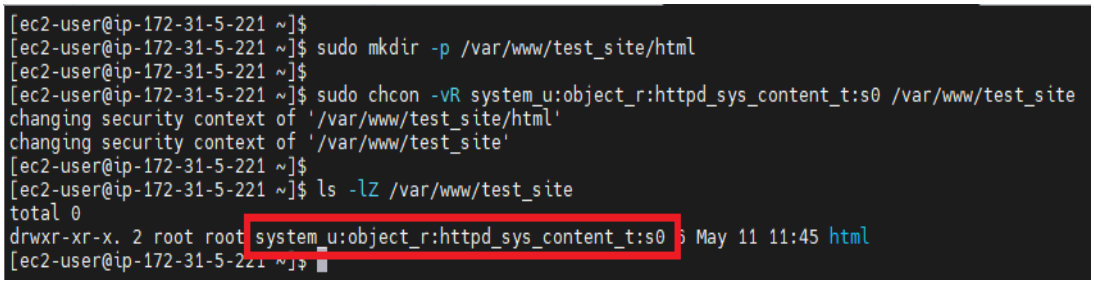
To begin, we will create a directory to store the website content with the following:

```
$ sudo mkdir -p /var/www/test_site/html
```

In order for the Nginx service to be able to access the newly created directory's contents, we will need to set the **SELinux** context type of the directory.

Set the **httpd_sys_content_t** context type on the newly created directory:

```
$ sudo chcon -vR system_u:object_r:httpd_sys_content_t:s0 /var/www/test_site
```




Next, we will create a basic html page that will be rendered when we access the site:

```
sudo vi /var/www/test_site/html/index.html
```

```
<html>
    <head>
        <title>Welcome to my Test Site!</title>
    </head>
    <body>
       <p>It works! Thanks for visiting my
             <b><span style="color:green">Test Site</span></b>!
       </p>
    </body>
</html>
```

Next, create the following file: `/etc/nginx/conf.d/test_site.conf`

This file contains a Nginx server block using the public IP address of my AWS RHEL 8 instance.

```
server {
  listen 80;
  server_name 3.93.33.164; # refers to Public IPv4 address of AWS instance
  root /var/www/test_site/html;
}
```

Verify that server block syntax in the configuration file is valid with the following:

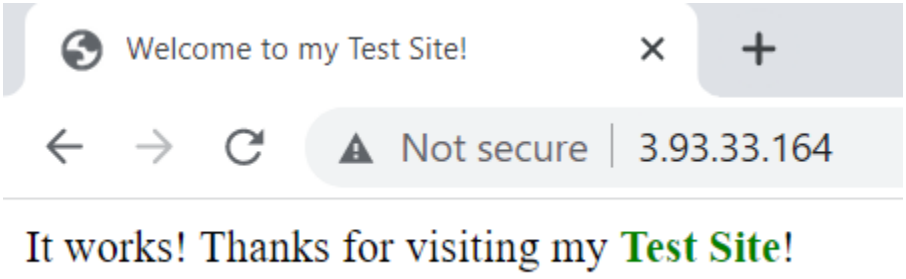
```
$ sudo nginx -t
```

Now, restart the Nginx service

```
$ sudo systemctl restart nginx
```

```
[ec2-user@ip-172-31-5-221 ~]$
[ec2-user@ip-172-31-5-221 ~]$ cat /var/www/test_site/html/index.html
<html>
    <head>
        <title>Welcome to my Test Site!</title>
    </head>
    <body>
       <p>It works! Thanks for visiting my <b><span style="color:green">Test Site</span></b>!</p>
    </body>
</html>
[ec2-user@ip-172-31-5-221 ~]$ cat /etc/nginx/conf.d/test_site.conf
server {
  listen 80;
  server_name 3.93.33.164; # refers to Public IPv4 address of AWS instance
  root /var/www/test_site/html;
}
[ec2-user@ip-172-31-5-221 ~]$ sudo nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
[ec2-user@ip-172-31-5-221 ~]$
[ec2-user@ip-172-31-5-221 ~]$ sudo systemctl restart nginx
[ec2-user@ip-172-31-5-221 ~]$
```

Open your browser and enter **your** instance's public IP address as the URL to confirm that it's working.




Please note that if you shutdown your RHEL 8 AWS instance, the connection details will change (**Public IPv4 address** & **Public IPv4 DNS**). Therefore, the corresponding change will need to be made in:

**/etc/nginx/conf.d/test_site.conf**

```
server {
  listen 80;
  server_name 3.93.33.164; # refers to Public IPv4 address of AWS instance
                           # MUST UPDATE after restarting AWS instance
  root /var/www/test_site/html;
}
```

We've successfully installed the Nginx web server and configured it to host a single website using the RHEL 8 instance's IP address. To host more than one site, we would need to use domain names. I will be demonstrating this process in a future tutorial.

I hope you've enjoyed this tutorial.

I have another tutorial where I install the Nginx web server on my **Ubuntu 20** EC2 (Elastic Compute Cloud) instance, accessible **here**. Although the Nginx installations are practically identical, I wanted to have two different instances for testing purposes. This will enable me to test my shell scripts that monitor services locally, as well as, remotely. If you're interested, the scripts can be accessed via my Linux tutorials page, **here**, while my main tutorials page is accessible **here**.

Please note that the free tier allows for 750 hours per month of Amazon EC2. You can create many EC2 instances but beware of the limit. If you go over that limit, you will pay the cost. My advice to you is to shutdown your instance/s after you've done your work.

Back to Top