AWS RHEL 8 Nginx Multisite Hosting

In this tutorial, we will be configuring Nginx for multisite hosting using freely registered domain names provided by Freenom, which is a free domain provider. I will be using my AWS RHEL 8 compute instance during the tutorial.

Prerequisites

- an AWS Free Tier account
- AWS RHEL 8 EC2 instance
- Nginx Web Server Installed
- 2 Registered Domain Names
- internet access

We will be using 2 freely registered domains during this tutorial, which you can get by using Freenom, a free domain provider. Learn how to register free domains, using my **Free Domain Name Registration** tutorial, accessible **here**.

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

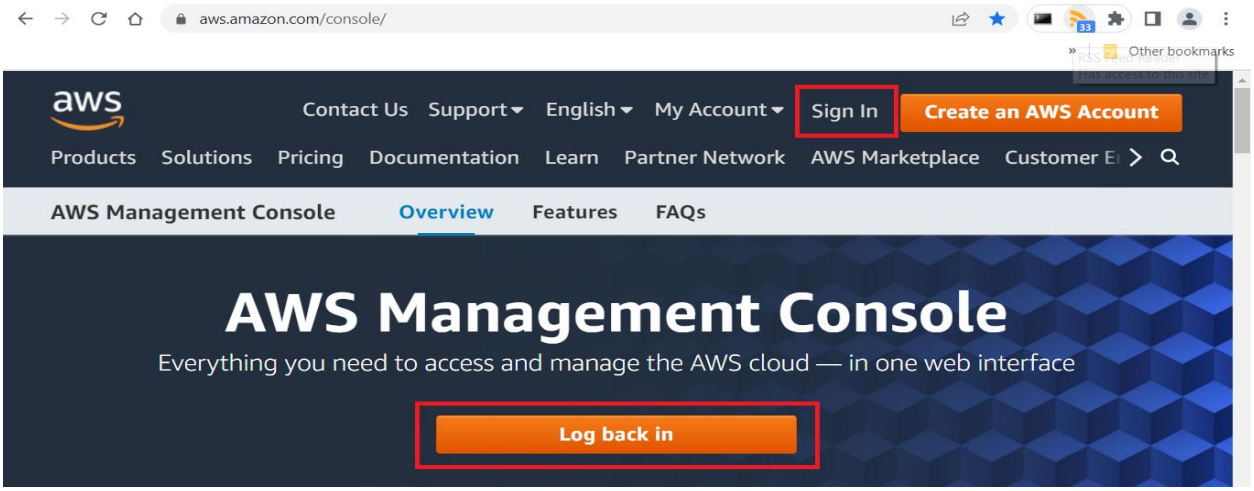
If you do not have an AWS RHEL 8 EC2 instance, my tutorial **Create AWS RHEL 8 EC2 Instance** is **here**.

Finally, if you do not have Nginx installed on your AWS RHEL 8 EC2 instance, complete my other tutorial **AWS RHEL 8 EC2 Nginx Install**, accessible **here**, then return to this tutorial.

Steps to complete tutorial:

Start RHEL 8 EC2 Instance

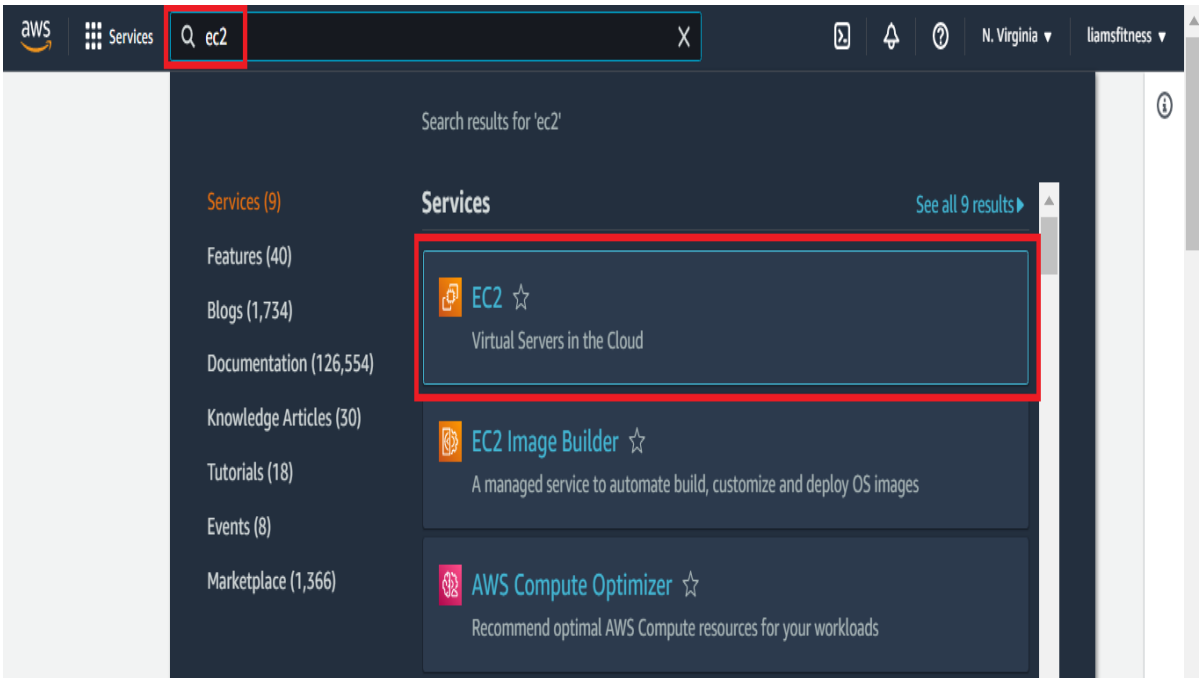
To begin, go to the following website, https://aws.amazon.com/console/ and log in to the console.



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.

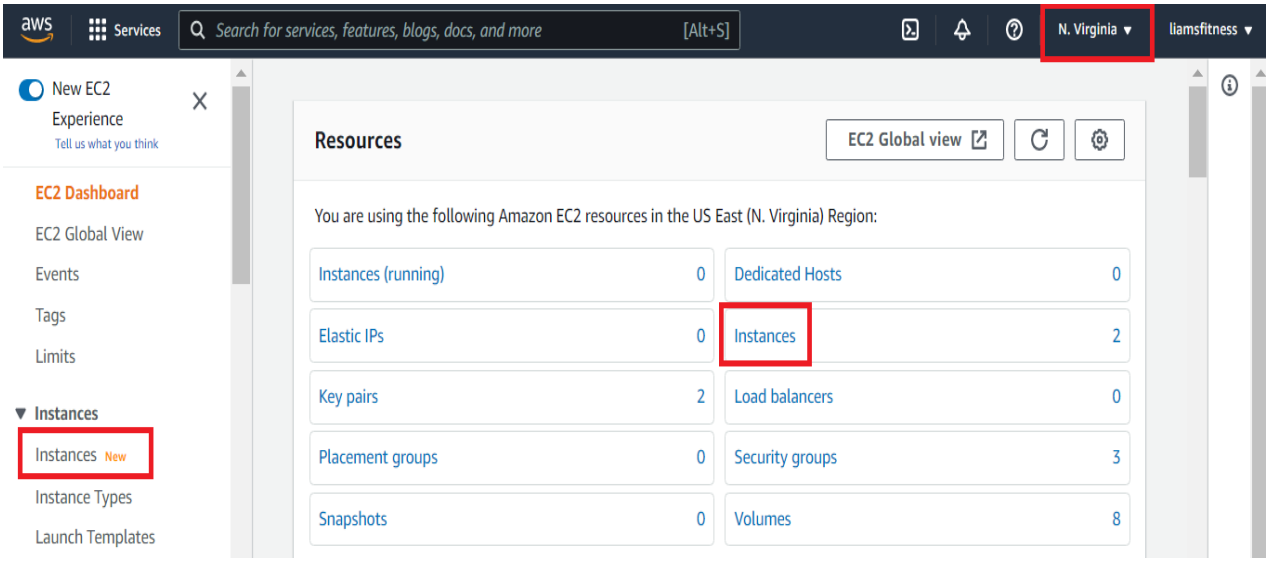
If you already know that your RHEL 8 EC2 instance is running, you can skip this step and go directly to **Register New Domains**.

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.

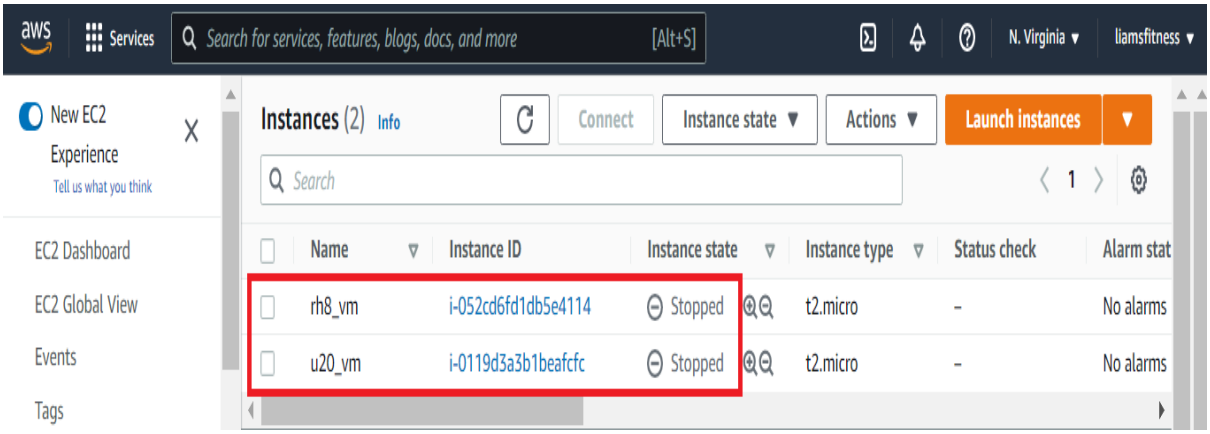


You will be brought to the **EC2** Dashboard. It contains links to the resources being used in the selected AWS region. In my case it's US East (N. Virginia).

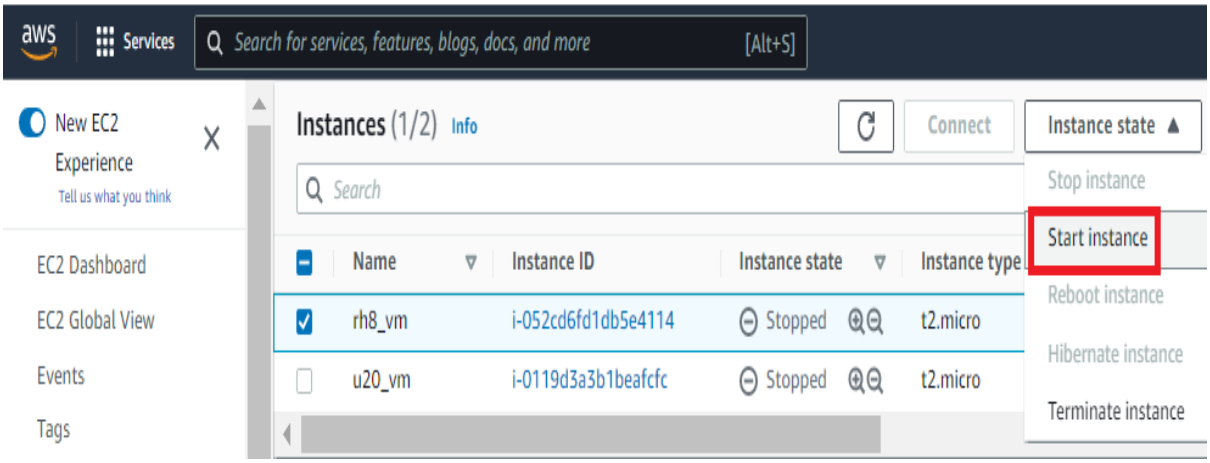
From the EC2 dashboard, click **Instances** (either link will work, your choice).



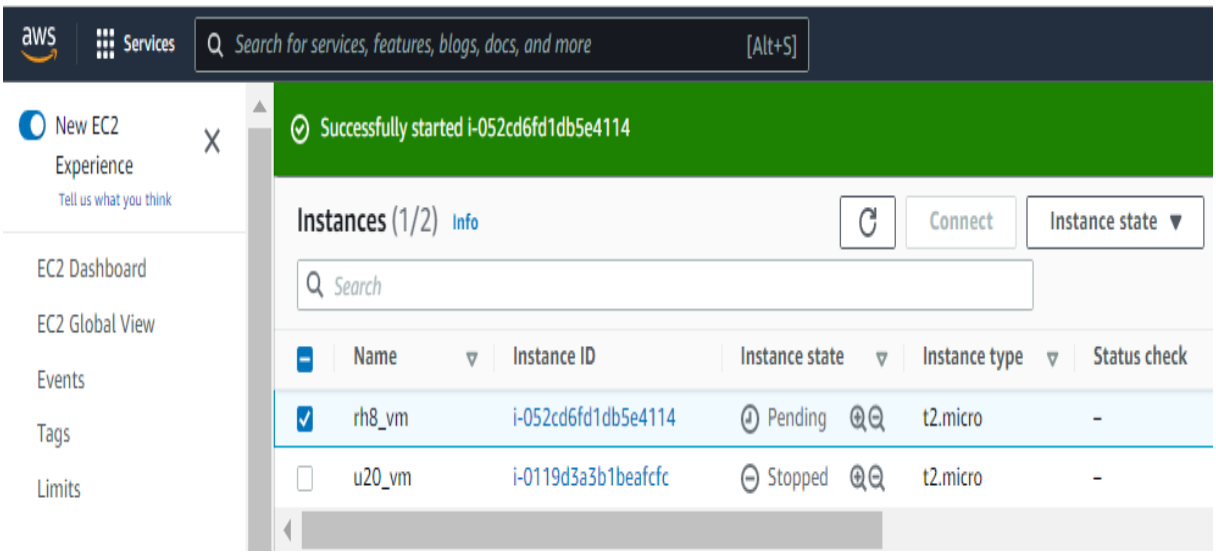
You will notice that both of my instances are in the stopped stated.



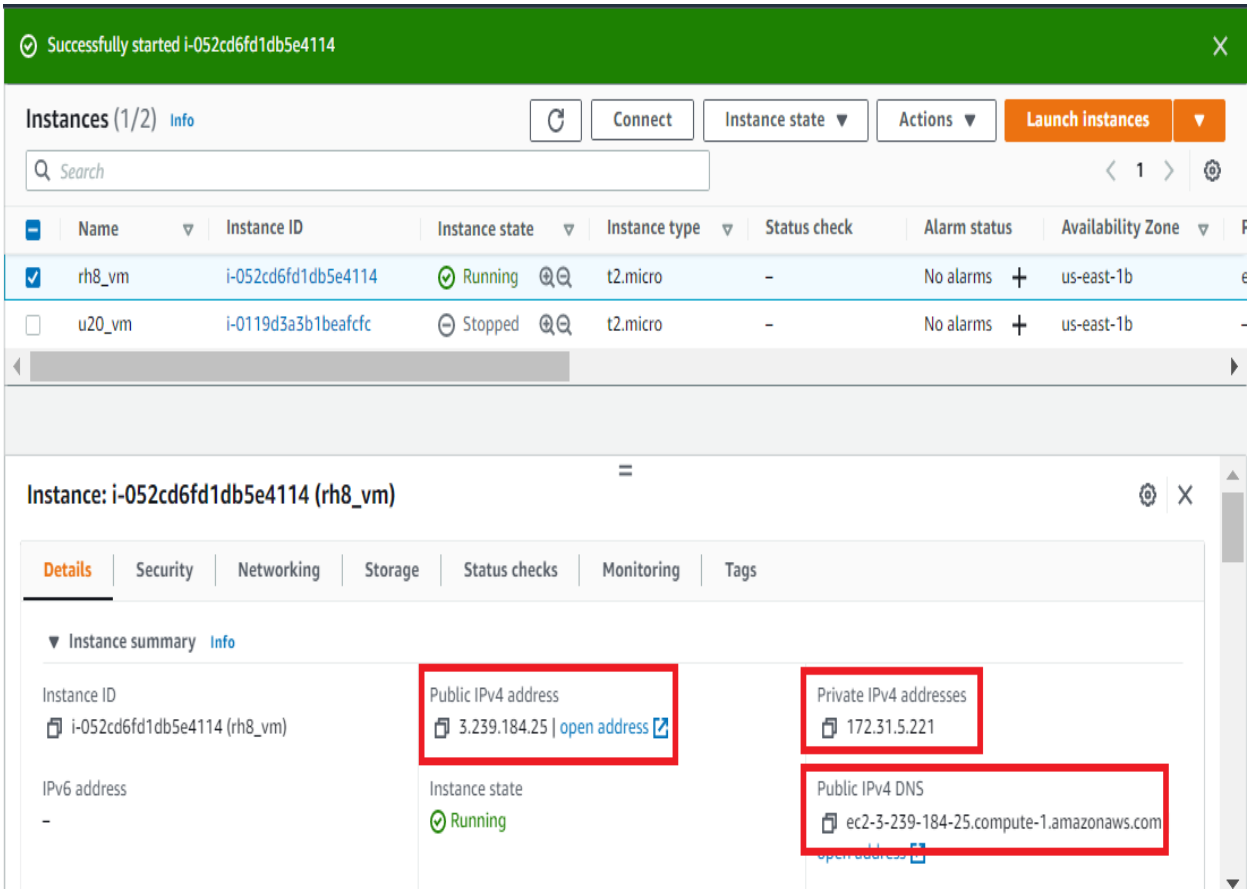
To start my RHEL 8 instance, I will ensure that my **rh8_vm** is selected (checkbox), then I will click the **Instance state** drop-down menu and click **Start instance**.



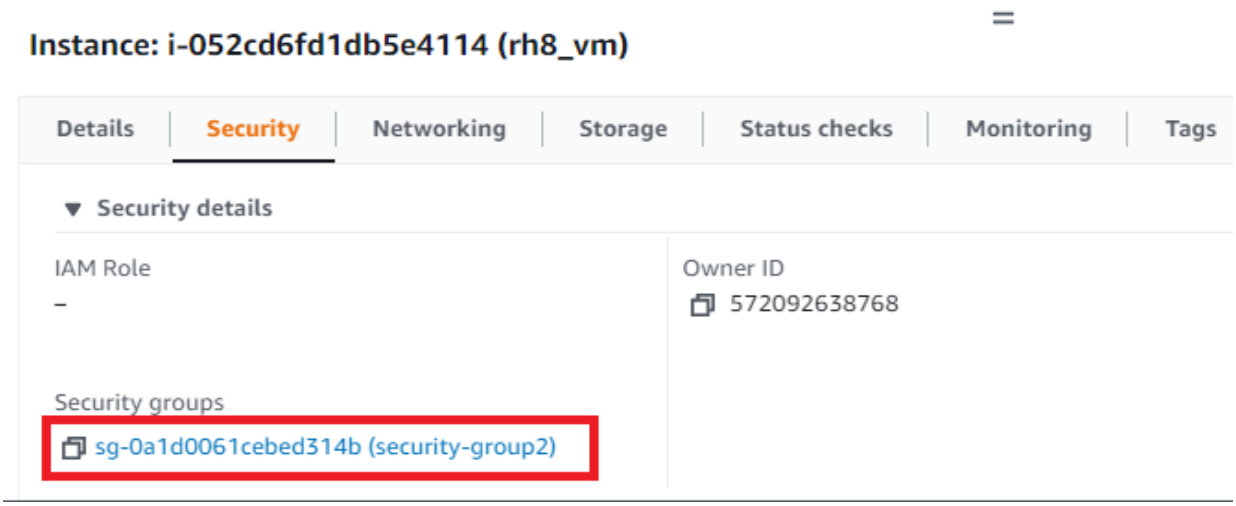
The instance will start and be given new connection details.



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.



Now, click the **Security** tab and note the name, and ID, of the **Security Group**, we will need this later.



Now we can register 2 free domain names. We can also create DNS records using the Public IPv4 address of the running RHEL 8 instance.

Register New Domains

To register 2 free domains, along with creating the corresponding **A** & **CNAME** DNS records, follow the steps in my **Free Domain Name Registration** tutorial, accessible **here**. The names you choose are up to you, but they must be unique.

I have registered the following 2 free domains: **`rh8-nginx-site1.tk & rh8-nginx-site2.tk`**

Please note that your domain names and IP addresses will be different than mine.

I have also created **A (host)** & **CNAME** records for each domain:

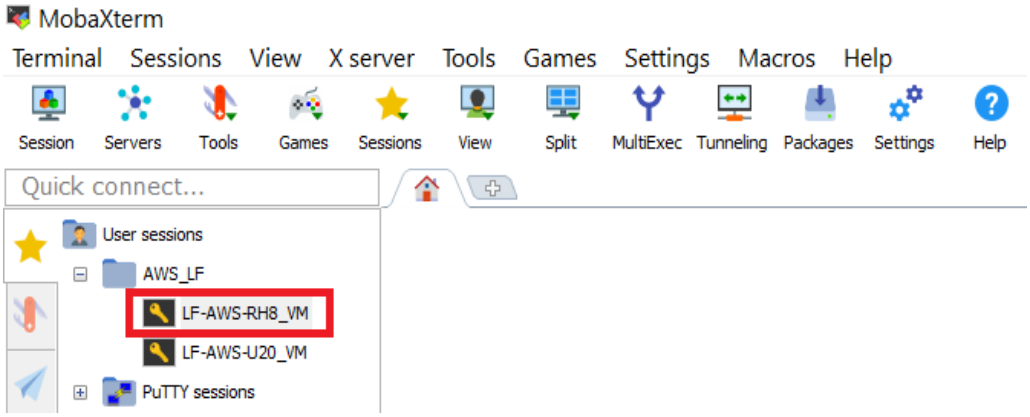
Name	Type	TTL	Target
	A	3600	3.239.184.25
WWW	CNAME	3600	rh8-nginx-site1.tk

Name	Type	TTL	Target
	A	3600	3.239.184.25
WWW	CNAME	3600	rh8-nginx-site2.tk

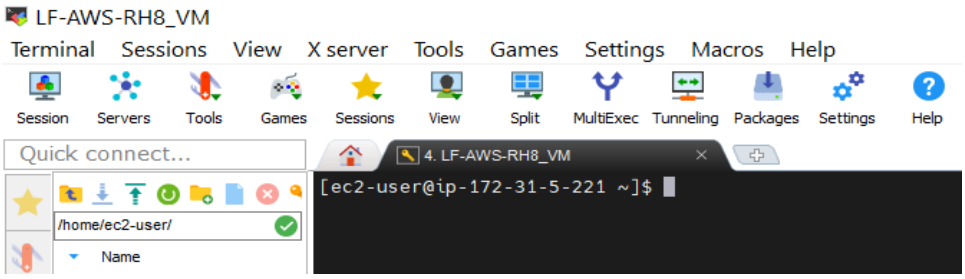
Connect to RHEL 8

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 configure the Nginx web server for multisite hosting.

Configure Nginx for Multisite Hosting

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

```
$ sudo mkdir -p /var/www/rh8-nginx-site1.tk/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/rh8-nginx-site1.tk
```

```
[ec2-user@ip-172-31-5-221 ~]$
[ec2-user@ip-172-31-5-221 ~]$ sudo mkdir -p /var/www/rh8-nginx-site1.tk/html
[ec2-user@ip-172-31-5-221 ~]$
[ec2-user@ip-172-31-5-221 ~]$ sudo chcon -vR system_u:object_r:httpd_sys_content_t:s0 /var/www/rh8-nginx-site1.tk
changing security context of '/var/www/rh8-nginx-site1.tk/html/index.html'
changing security context of '/var/www/rh8-nginx-site1.tk/html'
changing security context of '/var/www/rh8-nginx-site1.tk'
[ec2-user@ip-172-31-5-221 ~]$
[ec2-user@ip-172-31-5-221 ~]$ ls -lZ /var/www/rh8-nginx-site1.tk/
total 0
drwxr-xr-x. 2 root root system_u:object_r:httpd_sys_content_t:s0 24 May 29 14:09 html
[ec2-user@ip-172-31-5-221 ~]$
```

Next, create a basic html page: **`/var/www/rh8-nginx-site1.tk/html/index.html`**

This page will be rendered when we access the site.

```
<html>
    <head>
        <title>Welcome to my RHEL 8 Nginx Test Site #1!</title>
    </head>
    <body>
        <p>It works! Thanks for visiting my
            <b><span style="color:fuchsia">RHEL 8 Nginx Test Site #1</span></b>!
        </p>
    </body>
</html>
```

Next, create the following file: **`/etc/nginx/conf.d/rh8-nginx-site1.conf`**

This file contains a Nginx server block using one of the freely registered domain names created earlier.

(NOTE: your domain names will be different)

```
server {
  listen 80;
  server_name rh8-nginx-site1.tk www.rh8-nginx-site1.tk;
  root /var/www/rh8-nginx-site1.tk/html;
}
```

```
[ec2-user@ip-172-31-5-221 ~]$
[ec2-user@ip-172-31-5-221 ~]$ cat /var/www/rh8-nginx-site1.tk/html/index.html
<html>
    <head>
        <title>Welcome to my RHEL 8 Nginx Test Site #1!</title>
    </head>
    <body>
        <p>It works! Thanks for visiting my
          <b>
            <span style="color:fuchsia">RHEL 8 Nginx Test Site #1</span>
          </b>!
        </p>
    </body>
</html>
[ec2-user@ip-172-31-5-221 ~]$ cat /etc/nginx/conf.d/rh8-nginx-site1.conf
server {
  listen 80;
  server_name rh8-nginx-site1.tk www.rh8-nginx-site1.tk;
  root /var/www/rh8-nginx-site1.tk/html;
}
[ec2-user@ip-172-31-5-221 ~]$
```

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

`$ sudo nginx -t`

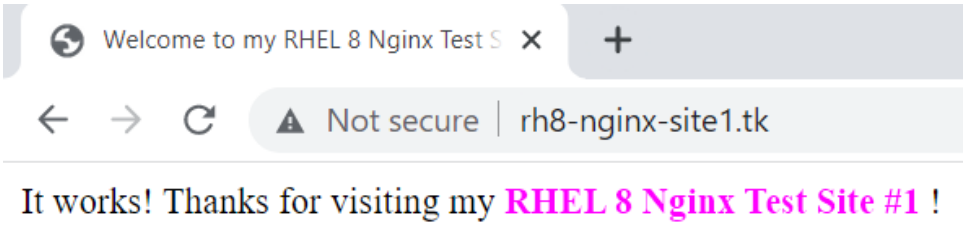
Now, restart the Nginx service

`$ sudo systemctl restart nginx`

```
[ec2-user@ip-172-31-5-221 ~]$
[ec2-user@ip-172-31-5-221 ~]$ 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 ~]$
```

In my **RHEL 8 EC2 Nginx Install** tutorial, accessible **here**, we added an inbound rule to allow HTTP traffic through so we can test our new site by entering either of the following URLs:

http://rh8-nginx-site1.tk OR http://www.rh8-nginx-site1.tk



Now we will configure our second website.

Create a directory to store the website content with the following:

```
$ sudo mkdir -p /var/www/rh8-nginx-site2.tk/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/rh8-nginx-site2.tk
```

```
[ec2-user@ip-172-31-5-221 ~]$
[ec2-user@ip-172-31-5-221 ~]$ sudo mkdir -p /var/www/rh8-nginx-site2.tk/html
[ec2-user@ip-172-31-5-221 ~]$
[ec2-user@ip-172-31-5-221 ~]$ sudo chcon -vR system_u:object_r:httpd_sys_content_t:s0 /var/www/rh8-nginx-site2.tk
changing security context of '/var/www/rh8-nginx-site2.tk/html'
changing security context of '/var/www/rh8-nginx-site2.tk'
[ec2-user@ip-172-31-5-221 ~]$
```

Next, create a basic html page: **/var/www/rh8-nginx-site2.tk/html/index.html**

This page will be rendered when we access the site.

```
<html>
    <head>
        <title>Welcome to my RHEL 8 Nginx Test Site #2!</title>
    </head>
    <body>
        <p>It works! Thanks for visiting my
              <b><span style="color:aqua">RHEL 8 Nginx Test Site #2</span></b>!
         </p>
    </body>
</html>
```

Next, create the following file: **/etc/nginx/conf.d/rh8-nginx-site2.conf**

This file contains a Nginx server block using one of the freely registered domain names created earlier.

(NOTE: your domain names will be different)

```
server {
  listen 80;
  server_name rh8-nginx-site2.tk www.rh8-nginx-site2.tk;
  root /var/www/rh8-nginx-site2.tk/html;
}
```

```
[ec2-user@ip-172-31-5-221 ~]$
[ec2-user@ip-172-31-5-221 ~]$ cat /var/www/rh8-nginx-site2.tk/html/index.html
<html>
    <head>
        <title>Welcome to my RHEL 8 Nginx Test Site #2!</title>
    </head>
    <body>
        <p>It works! Thanks for visiting my
          <b>
            <span style="color:aqua">RHEL 8 Nginx Test Site #2</span>
          </b>!
        </p>
    </body>
</html>
[ec2-user@ip-172-31-5-221 ~]$ cat /etc/nginx/conf.d/rh8-nginx-site2.conf
server {
  listen 80;
  server_name rh8-nginx-site2.tk www.rh8-nginx-site2.tk;
  root /var/www/rh8-nginx-site2.tk/html;
}
[ec2-user@ip-172-31-5-221 ~]$
```

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

```
$ sudo nginx -t
```

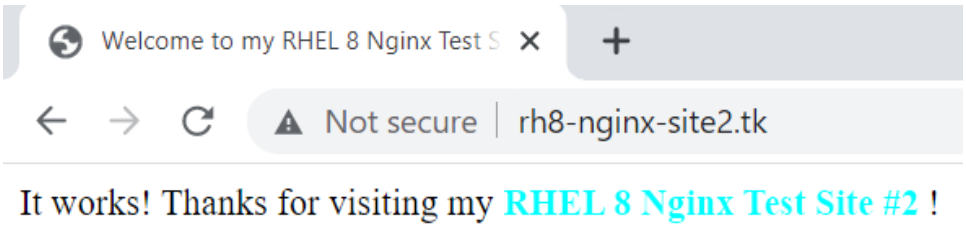
Now, restart the Nginx service

```
$ sudo systemctl restart nginx
```

```
[ec2-user@ip-172-31-5-221 ~]$
[ec2-user@ip-172-31-5-221 ~]$ 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 ~]$
```

We can test our second test site by entering either of the following URLs:

http://rh8-nginx-site2.tk OR http://www.rh8-nginx-site2.tk

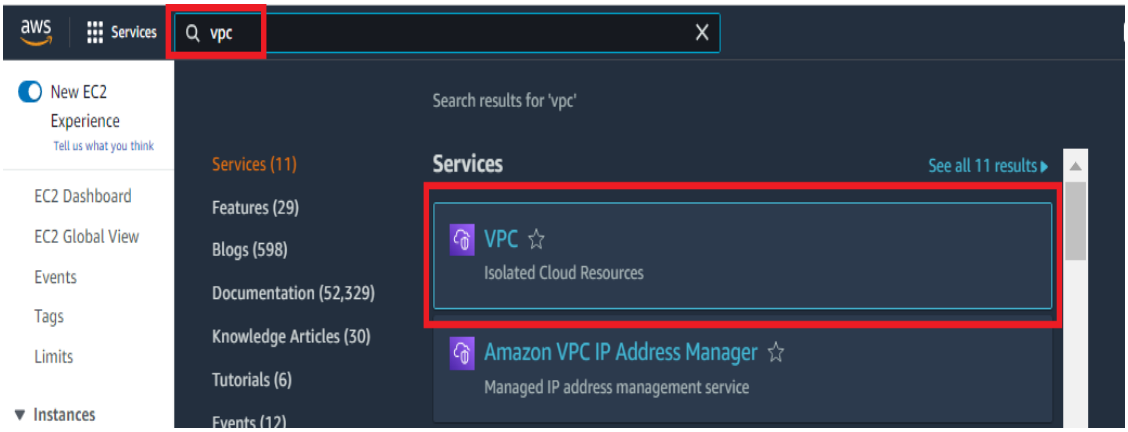


Please note that if you shutdown your RHEL 8 AWS instance, the connection details will change (**Public IPv4 address** & **Public IPv4 DNS**). Therefore, the corresponding change will need to be made to your DNS A (host) record via the **Freenom** DNS management interface.

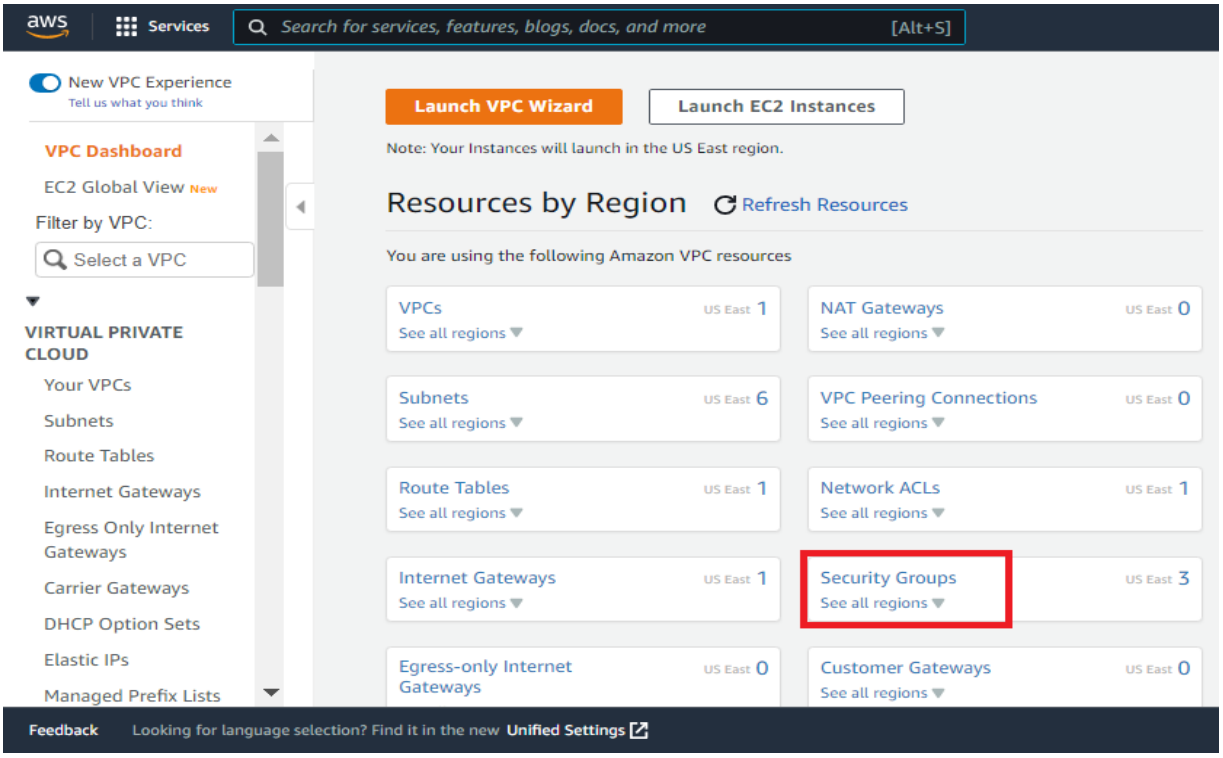
We have successfully configured Nginx for multisite hosting, but the websites are not secured with SSL certificates. To do this, first, we will need to open a port in our firewall to be able to access the secured websites. We will configure our VPC (Virtual Private Cloud) Security Group to allow HTTPS traffic through the firewall. This will allow HTTPS access (port 443) 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)

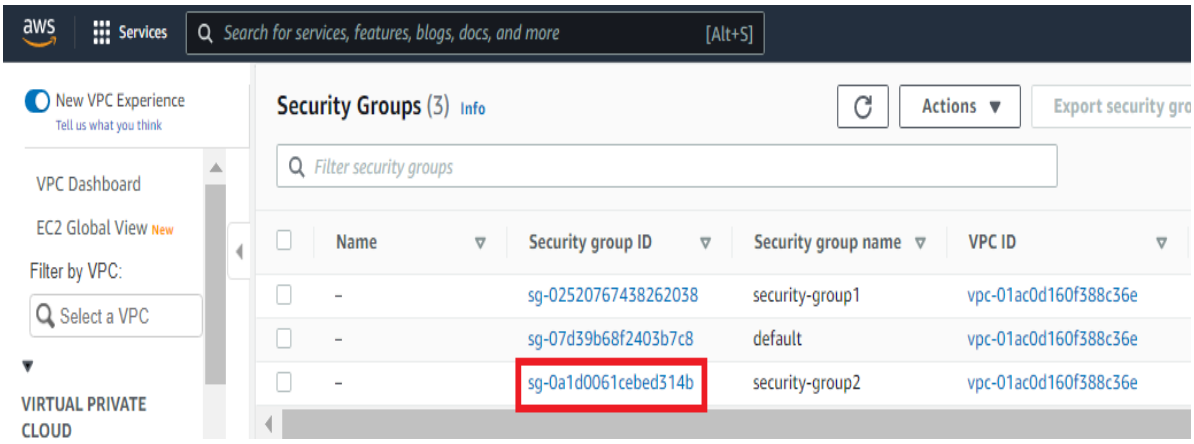
From the AWS Console screen, enter VPC in the search bar and select **VPC** Isolated Cloud Resources.



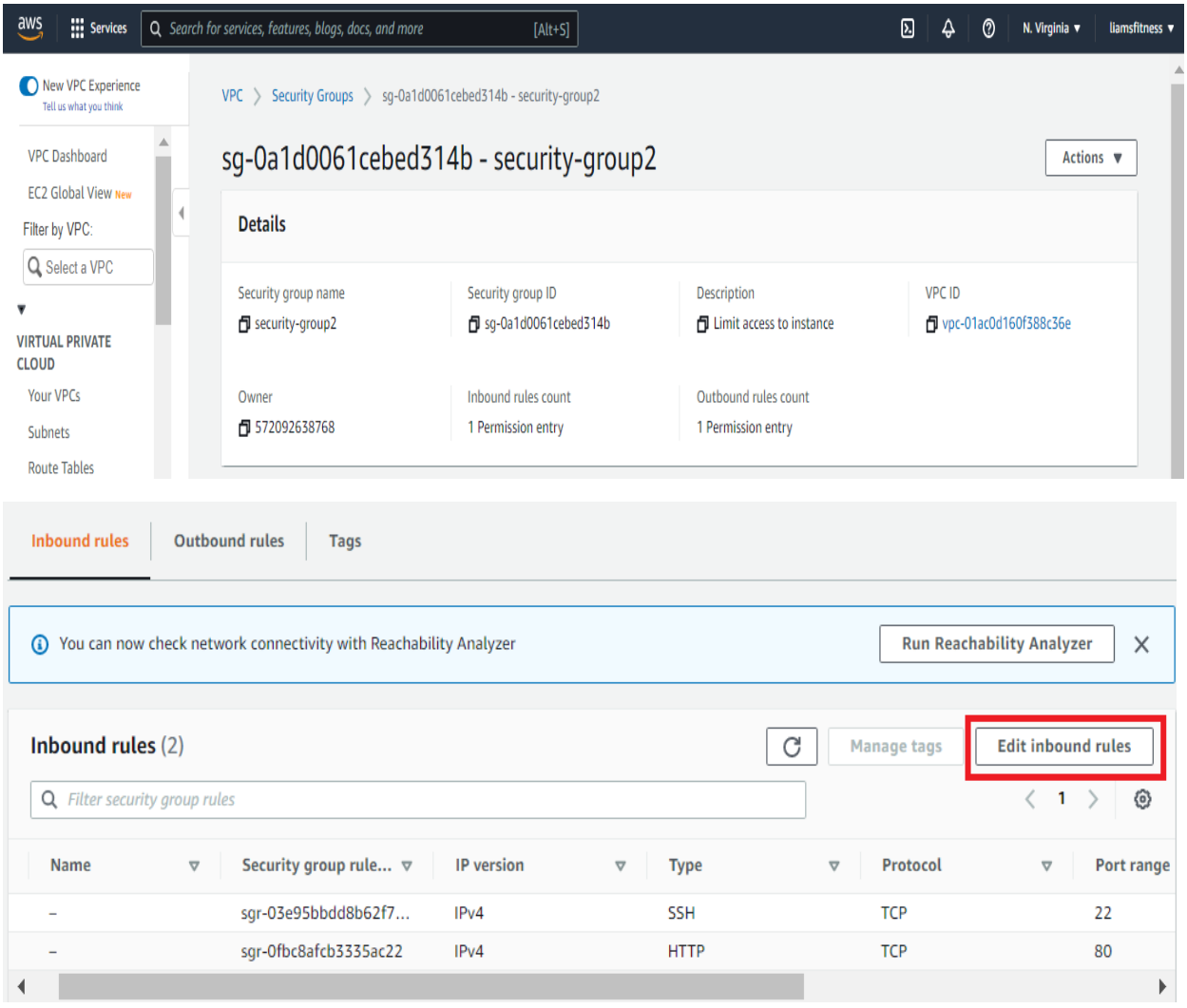
On the VPC Dashboard, select **Security Groups**.



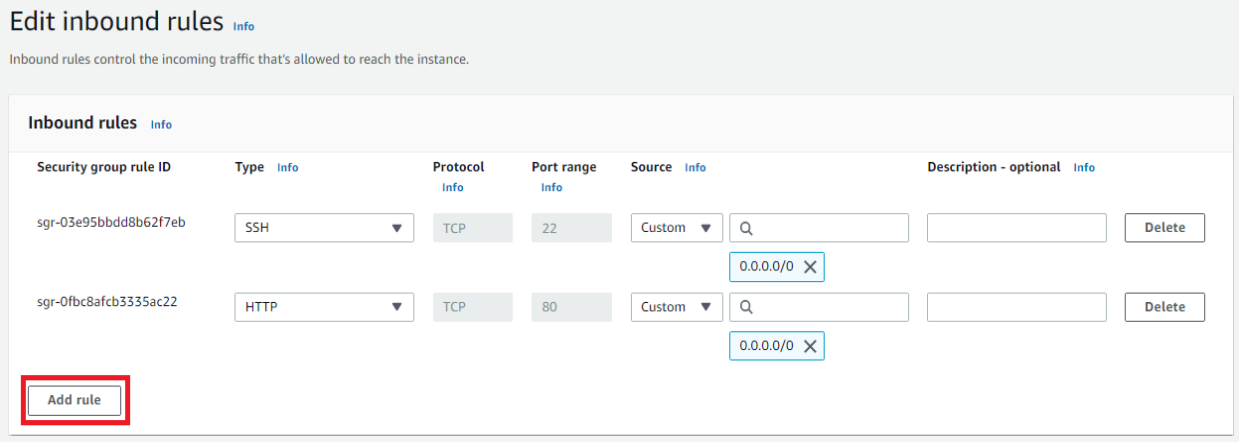
On the **Security Groups** screen, click the **Security group ID** that is associated with your RHEL 8 EC2 instance.



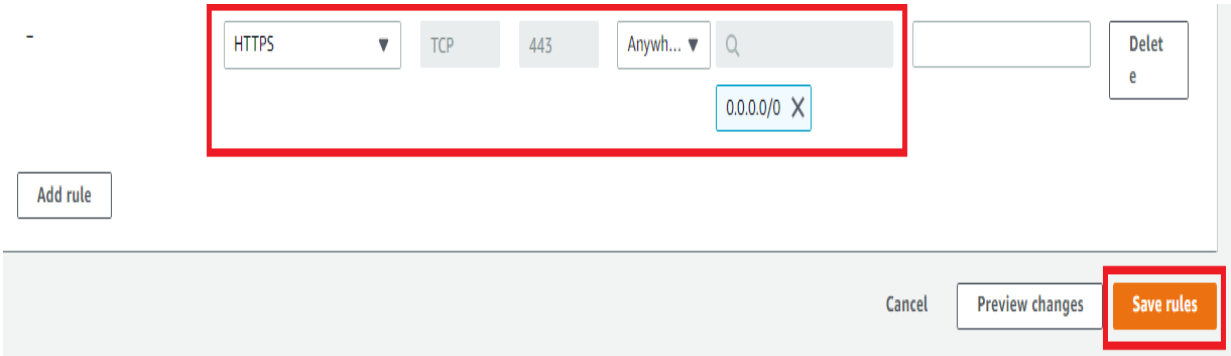
On the security group screen, scroll down and ensure the **Inbound Rules tab** is selected, then click **Edit Inbound Rules**

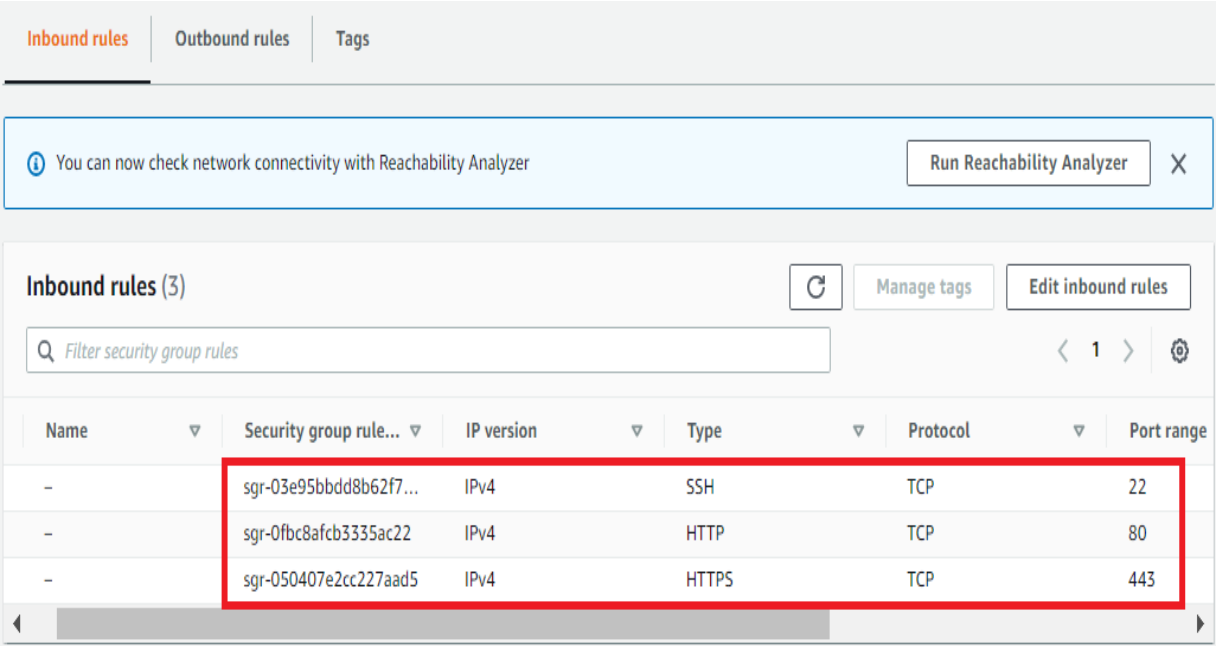


On the **Edit Inbound Rules** screen, click the **Add rule** button



Ensure **Port range** is set to **443** (HTTPS) and **Source** is set to **0.0.0.0/0** (Anywhere-IPv4). Click **Save rules**





Now that we've opened the firewall for HTTPS access, we can connect to the RHEL 8 instance and install the necessary packages to secure our sites with free **Let's Encrypt** SSL certificates.

Secure Nginx with Let's Encrypt

The first step to using Let's Encrypt to obtain an SSL certificate is to install the Certbot software.

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:        8.6
Arch:           x86_64
Status:         Subscribed
Status Details:
Starts:         05/10/2022
Ends:           05/10/2023
```

The certbot software consists of two packages: **`certbot`** & **`python3-certbot-nginx`**

We will first search for these packages in the RHEL 8 default repositories with the following:

```
$ sudo yum search certbot python3-certbot-nginx
```

```
[ec2-user@ip-172-31-5-221 ~]$
[ec2-user@ip-172-31-5-221 ~]$ sudo yum search certbot python3-certbot-nginx
Updating Subscription Management repositories.
Last metadata expiration check: 1:02:36 ago on Thu 02 Jun 2022 09:04:27 AM UTC.
No matches found.
[ec2-user@ip-172-31-5-221 ~]$
```

We will need to install the EPEL repository (Extra Packages for Enterprise Linux).

```
$ sudo yum -y install https://dl.fedoraproject.org/pub/epel/epel-release-latest-8.noarch.rpm
```

```
$
$ sudo yum -y install https://dl.fedoraproject.org/pub/epel/epel-release-latest-8.noarch.rpm
```

After installation completes, we will verify that the EPEL repo is now enabled.

```
$ sudo yum repolist
```

```
[ec2-user@ip-172-31-5-221 ~]$
[ec2-user@ip-172-31-5-221 ~]$ sudo yum repolist
Updating Subscription Management repositories.
repo id                                     repo name
epel                                        Extra Packages for Enterprise Linux 8 - x86_64
epel-modular                                Extra Packages for Enterprise Linux Modular 8 - x86_64
rhel-8-appstream-rhui-rpms                  Red Hat Enterprise Linux 8 for x86_64 - AppStream from RHUI (RPMs)
rhel-8-baseos-rhui-rpms                     Red Hat Enterprise Linux 8 for x86_64 - BaseOS from RHUI (RPMs)
rhui-client-config-server-8                 Red Hat Update Infrastructure 3 Client Configuration Server 8
[ec2-user@ip-172-31-5-221 ~]$
```

We can now install the certbot software packages:

```
$ sudo yum install -y certbot python3-certbot-nginx
```

```
[ec2-user@ip-172-31-5-221 ~]$
[ec2-user@ip-172-31-5-221 ~]$ sudo yum install -y certbot python3-certbot-nginx
```

I can now obtain a free Let's Encrypt SSL certificate for my first test site.

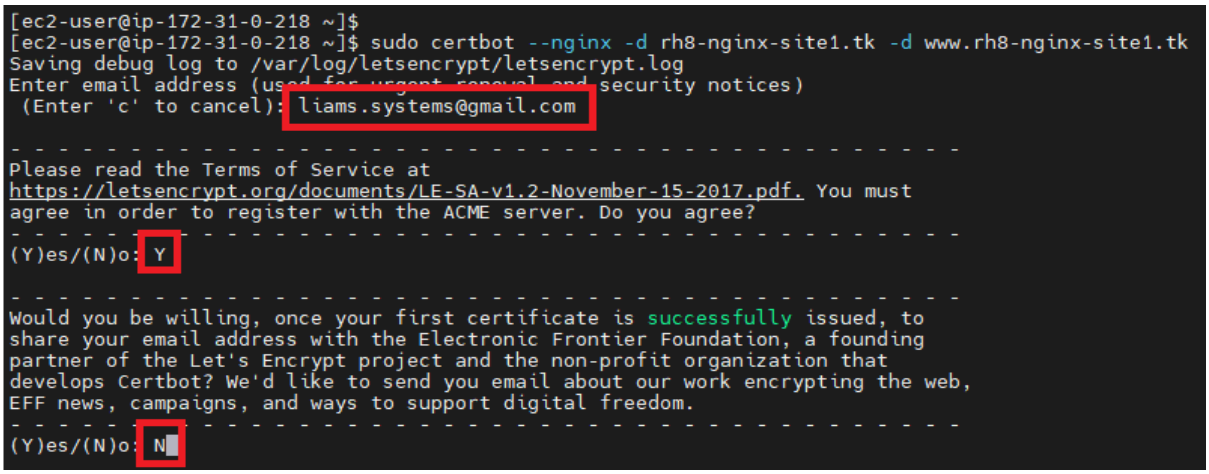
NOTE: use your domain names

```
$ sudo certbot --nginx -d rh8-nginx-site1.tk -d www.rh8-nginx-site1.tk
```

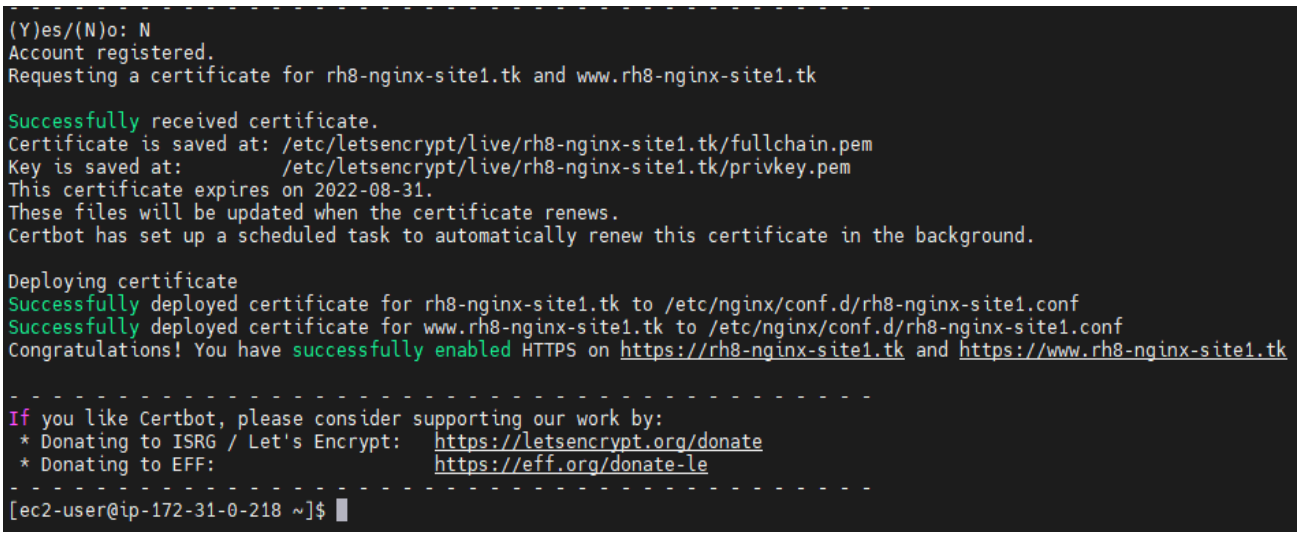
If this is your first time running certbot, you will be prompted to enter an email address and agree to the terms of service, as well as, whether you want to share your email with EFF (Electronic Frontier Foundation).

*Please note that if you completed my **Ubuntu 20 Nginx Multisite Hosting** tutorial and you shared your email with EFF, enter **N** when prompted this time around.*

After doing so, certbot will communicate with the Let's Encrypt server, then run a challenge to verify that you control the domain you're requesting a certificate for.



I have successfully obtained an SSL certificate for **rh8-nginx-site1.tk** & **www.rh8-nginx-site1.tk**



Now, I will obtain a free Let's Encrypt SSL certificate for my second test site.

NOTE: use your domain names

```
$ sudo certbot --nginx -d rh8-nginx-site2.tk -d www.rh8-nginx-site2.tk
```

The certificates have now been downloaded, installed, and loaded for both of my sites.

While obtaining Let's Encrypt certificates for my sites, their corresponding configuration files will automatically be updated. I can verify this with the following:

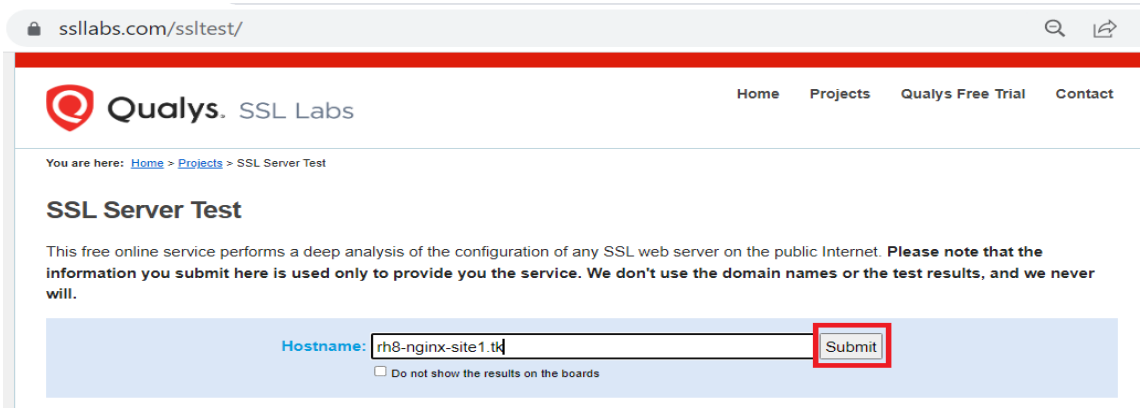
```
$ cat /etc/nginx/conf.d/rh8-nginx-site1.tk
```

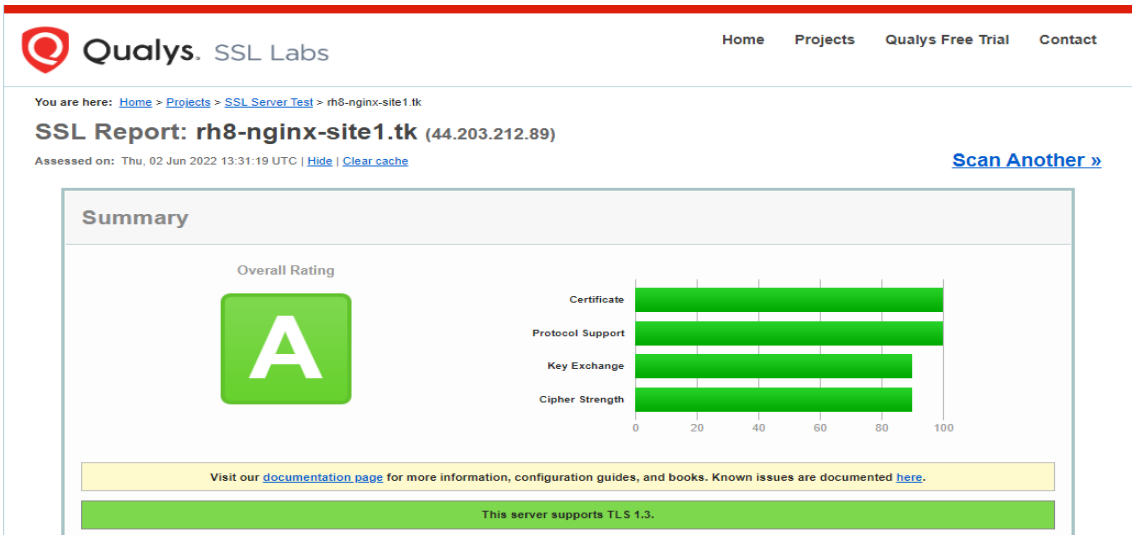
```
$ cat /etc/nginx/conf.d/rh8-nginx-site2.tk
```

```
[ec2-user@ip-172-31-0-218 ~]$
[ec2-user@ip-172-31-0-218 ~]$ cat /etc/nginx/conf.d/rh8-nginx-site1.conf
server {
  server_name rh8-nginx-site1.tk www.rh8-nginx-site1.tk;
  root /var/www/rh8-nginx-site1.tk/html;

    listen 443 ssl; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/rh8-nginx-site1.tk/fullchain.pem; # managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/rh8-nginx-site1.tk/privkey.pem; # managed by Certbot
    include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot


}
server {
    if ($host = www.rh8-nginx-site1.tk) {
        return 301 https://$host$request_uri;
    } # managed by Certbot


    if ($host = rh8-nginx-site1.tk) {
        return 301 https://$host$request_uri;
    } # managed by Certbot


  listen 80;
  server_name rh8-nginx-site1.tk www.rh8-nginx-site1.tk;
    return 404; # managed by Certbot



}[ec2-user@ip-172-31-0-218 ~]$
```

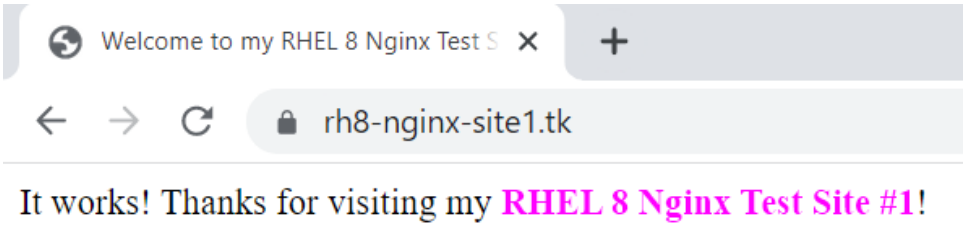
You will notice that all requests will be redirected to secure HTTPS.

We can even test our newly secured websites using **SSL Server Test**.

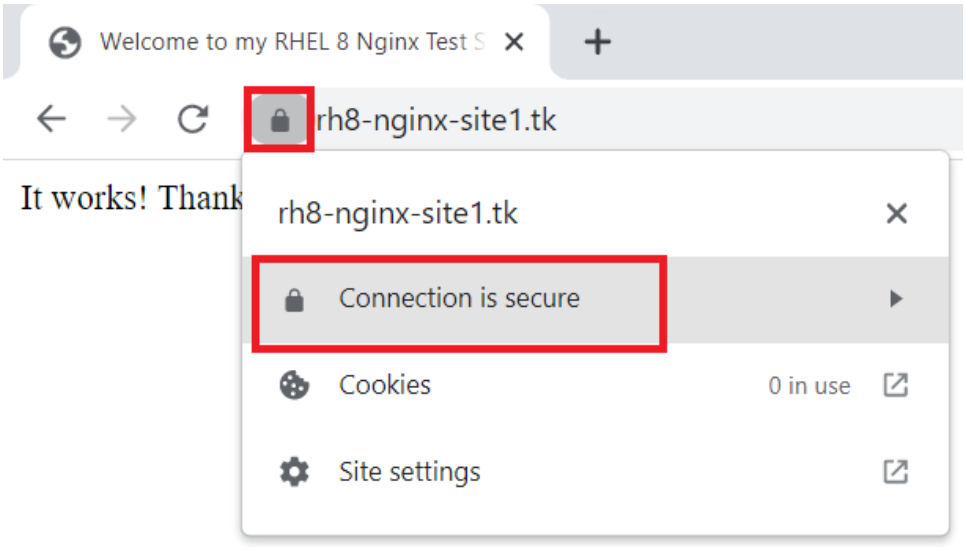


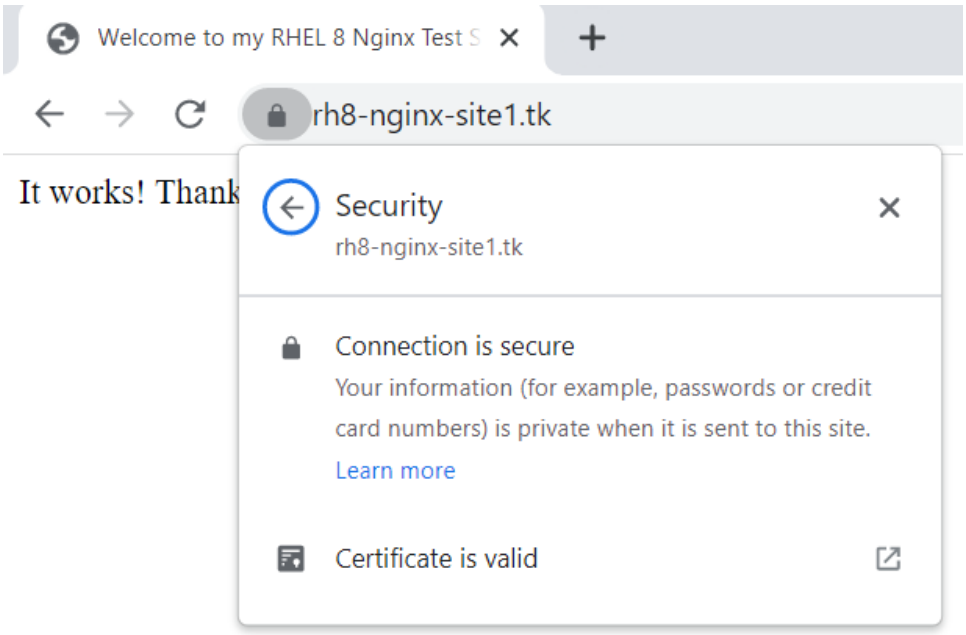


Now if I open my website, using either http:// or https://, redirection will happen automatically.

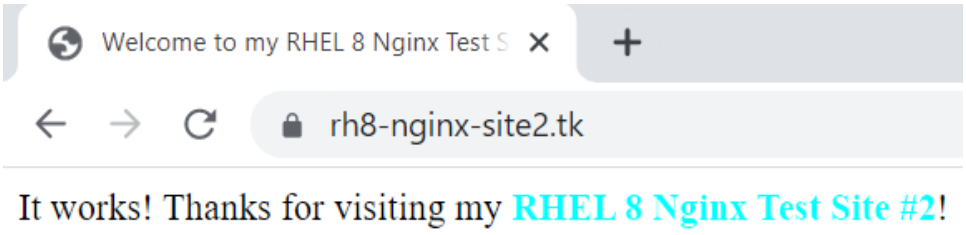


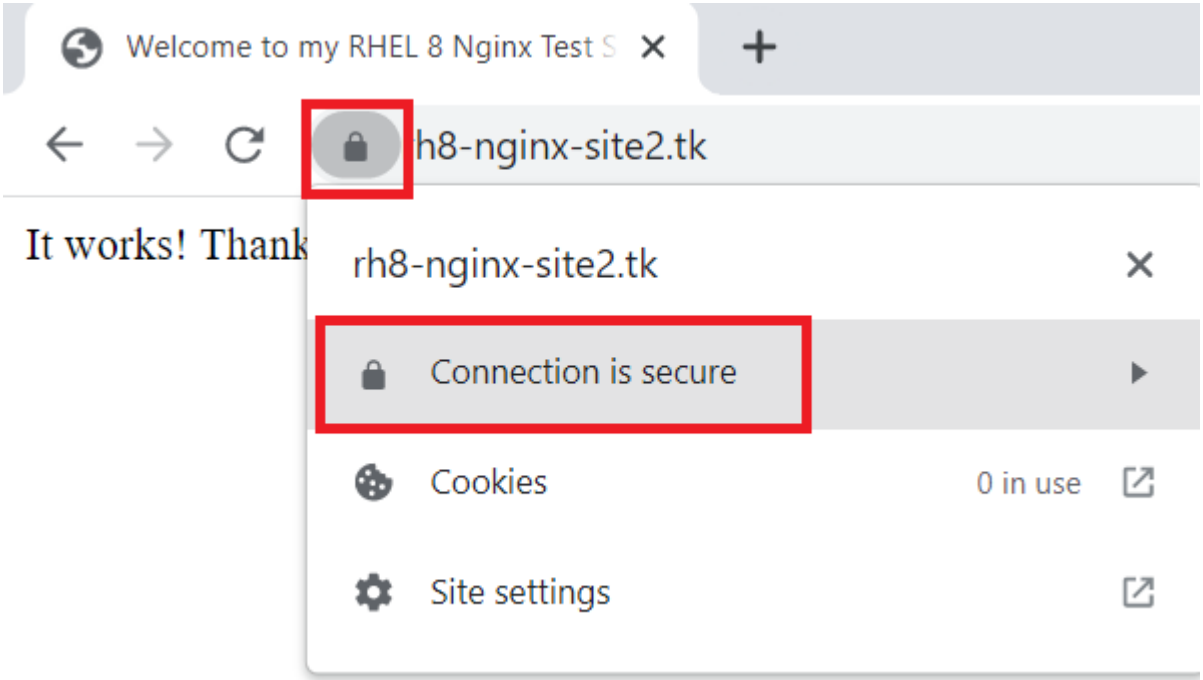
Also notice the browser's security indicator, it should indicate that the site is properly secured with a lock icon.

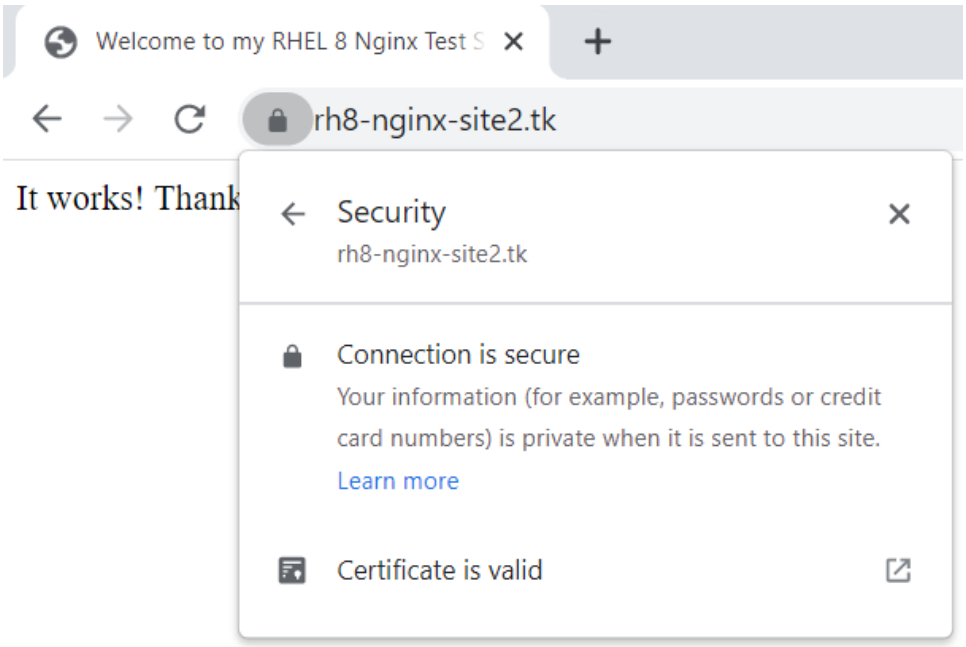




Now, for my second test site.







Let's Encrypt's certificates are only valid for ninety days. This is to encourage users to automate their certificate renewal process. The Certbot Let's Encrypt client has a renew command that checks the currently installed certificates and tries to renew them if they are less than 30 days away from the expiration date.

To manually test the renewal process, use the certbot renew command with the --dry-run option.

```
$ sudo certbot renew --dry-run
```

```
[ec2-user@ip-172-31-0-218 ~]$
[ec2-user@ip-172-31-0-218 ~]$ sudo certbot renew --dry-run
Saving debug log to /var/log/letsencrypt/letsencrypt.log

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Processing /etc/letsencrypt/renewal/rh8-nginx-site1.tk.conf
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Account registered.
Simulating renewal of an existing certificate for rh8-nginx-site1.tk and www.rh8-nginx-site1.tk

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Processing /etc/letsencrypt/renewal/rh8-nginx-site2.tk.conf
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Simulating renewal of an existing certificate for rh8-nginx-site2.tk and www.rh8-nginx-site2.tk

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Congratulations, all simulated renewals succeeded:
  /etc/letsencrypt/live/rh8-nginx-site1.tk/fullchain.pem (success)
  /etc/letsencrypt/live/rh8-nginx-site2.tk/fullchain.pem (success)
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
[ec2-user@ip-172-31-0-218 ~]$
```

Certbot inspects the certificates and confirms they are not due to be renewed, but simulates the process anyway. It displays details regarding whether the renewal would have been successful.

Although I was able to renew my certificates manually, we should automate the process.

Unlike Ubuntu 20, where it's certbot package includes a **systemd** (Linux system & service manager) timer that will run twice a day and automatically renew any certificate that's within thirty days of expiration, we will create a cron job to take care of certificate renewal.

First, open the root user's crontab file with the following:

```
$ sudo crontab -e
```

Then, enter the following:

```
0 12 * * * /usr/bin/certbot renew --quiet
```

Next, to save the file, hit the **Esc** button followed by **:x!**

The above certbot command will run daily at noon and check if certificates will expire within the next 30 days. If yes, they are renewed. The --quiet directive tells certbot not to generate output.

Now we will confirm that the cron job exists with the following:

```
$ sudo crontab -l
```

```
[ec2-user@ip-172-31-0-218 ~]$
[ec2-user@ip-172-31-0-218 ~]$ sudo crontab -e
no crontab for root - using an empty one
crontab: installing new crontab
[ec2-user@ip-172-31-0-218 ~]$
[ec2-user@ip-172-31-0-218 ~]$ sudo crontab -l
0 12 * * * /usr/bin/certbot renew --quiet
[ec2-user@ip-172-31-0-218 ~]$
```

We've successfully configured the Nginx web server for multisite hosting using Freenom's freely registered domains. We have also configured our AWS VPC Security Group to allow HTTPS access to our RHEL 8 compute instance. Next, we secured the sites using free Let's Encrypt SSL certificates. Afterwards, we manually simulated certificate renewal. Finally, we created a cron job to automate certificate renewal and ensure that our sites would remain secure after 90 days.

I hope you've enjoyed this tutorial.

I have another tutorial where I configure Nginx multisite hosting on my **Ubuntu 20** EC2 (Elastic Compute Cloud) instance, accessible **here**. Although the steps 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