

AWS Ubuntu 20 EC2 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 Ubuntu 20 compute instance during the tutorial.

Prerequisites

- an AWS Free Tier account
- AWS Ubuntu 20 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 Ubuntu 20 EC2 instance, my tutorial **Create AWS Ubuntu 20 EC2 Instance** is [here](#).

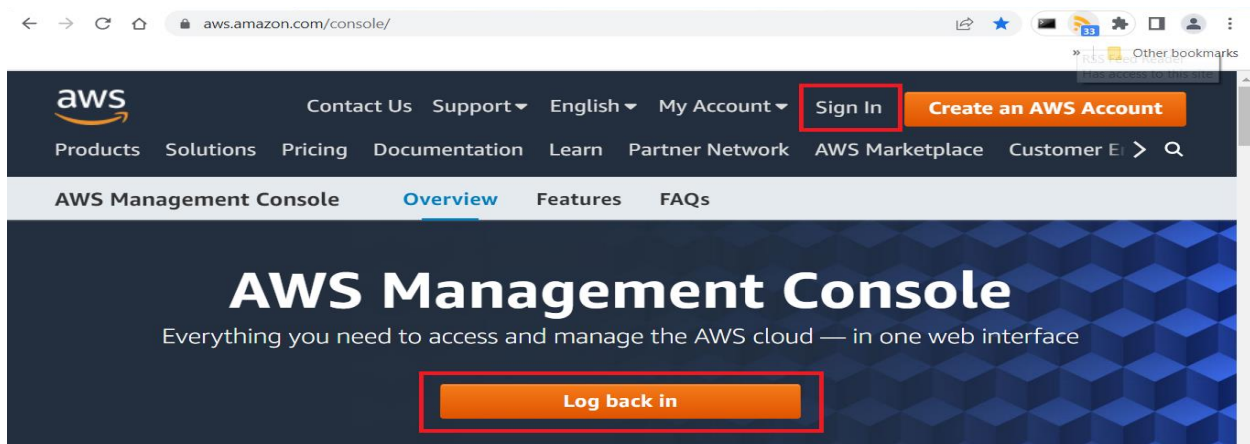
Finally, if you do not have Nginx installed on your AWS Ubuntu 20 EC2 instance, complete my other tutorial **AWS Ubuntu 20 EC2 Nginx Install**, accessible [here](#), then return to this tutorial.

Steps to complete tutorial:

- [Start Ubuntu 20 EC2 Instance](#)
- [Register New Domains](#)
- [Connect to Ubuntu 20](#)
- [Configure Nginx for Multisite Hosting](#)
- [Update Virtual Private Cloud \(VPC\)](#)
- [Secure Nginx with Let's Encrypt](#)

Start Ubuntu 20 EC2 Instance

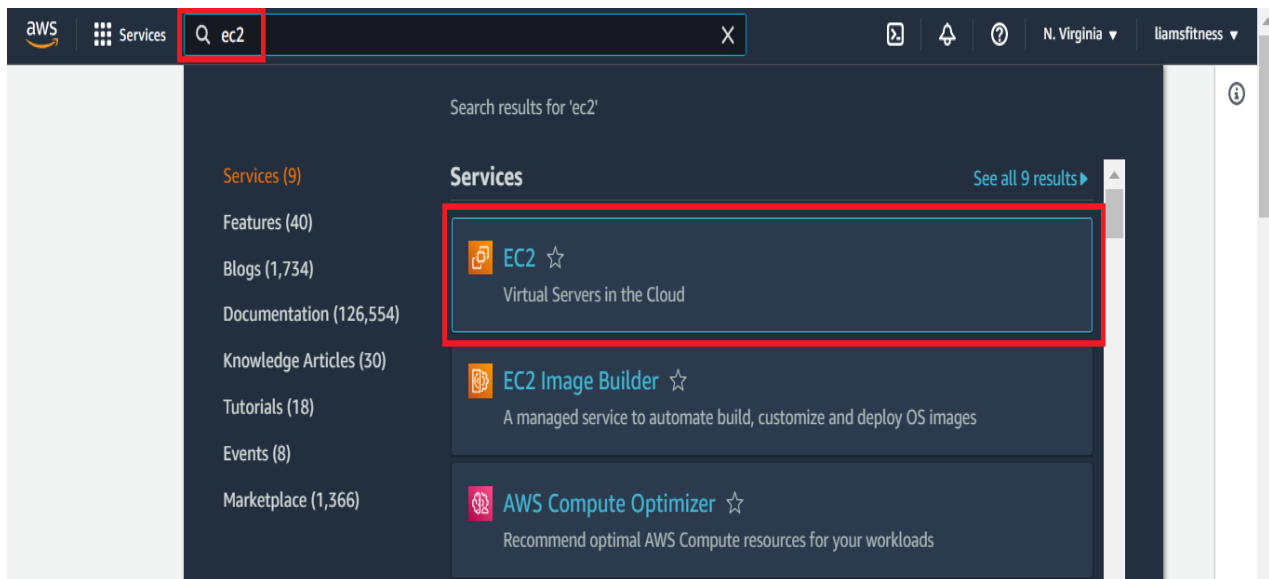
To begin, go to the following website, <https://aws.amazon.com/console/> and log in to the console.



At the end of my **Create AWS Ubuntu 20 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.

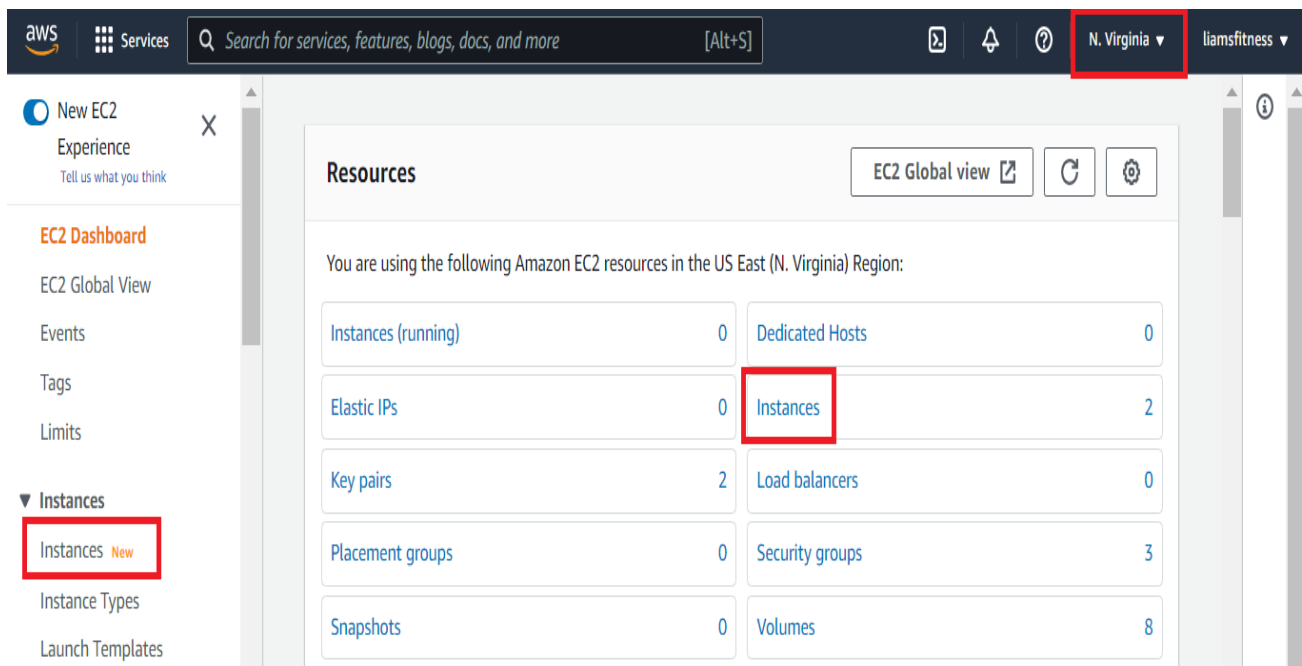
If you already know that your Ubuntu 20 EC2 instance is running, you can skip this step and go directly to [Register New Domains](#).

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.

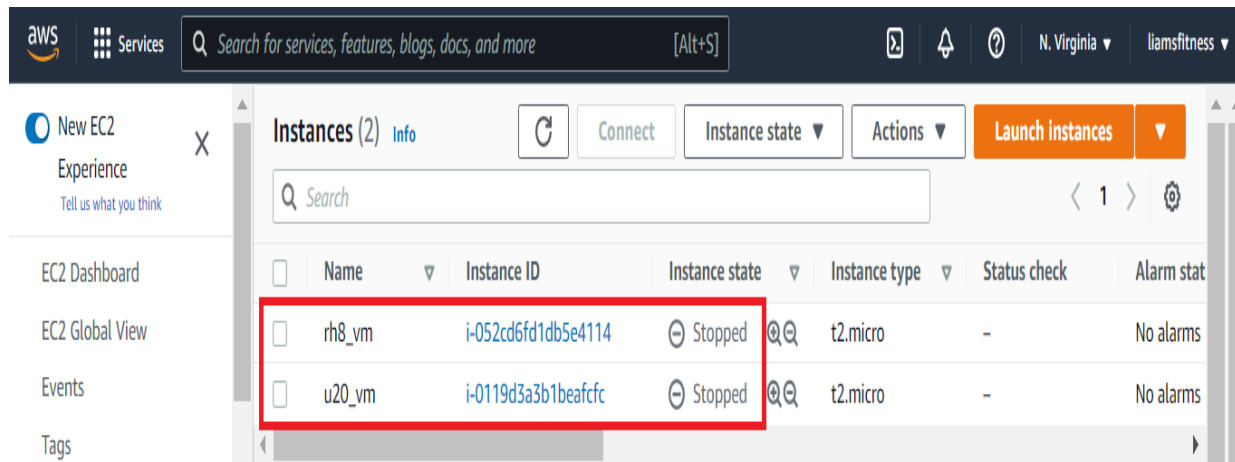


You will be brought to the **EC2 Dashboard**. It contains links to the resources being used in the selected AWS region. In my case it's US East (N. Virginia).

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



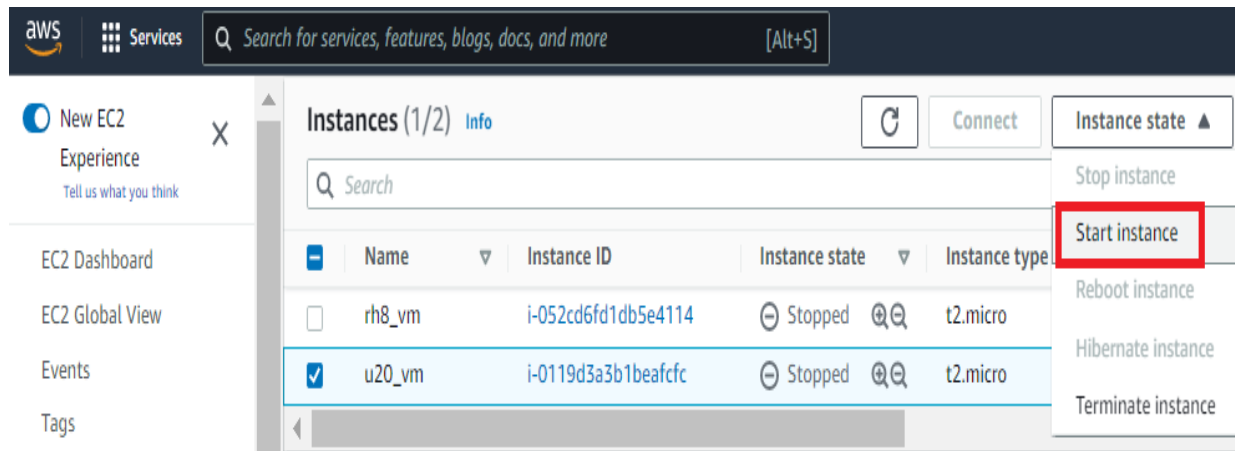
You will notice that both of my instances are in the stopped stated.



The screenshot shows the AWS Management Console interface. On the left is a navigation sidebar with links to 'New EC2 Experience', 'EC2 Dashboard', 'EC2 Global View', 'Events', and 'Tags'. The main content area is titled 'Instances (2)' and contains a table of EC2 instances. A red rectangular box highlights the first two rows of the table, which are both in the 'Stopped' state.

<input type="checkbox"/>	Name	Instance ID	Instance state	Instance type	Status check	Alarm stat
<input type="checkbox"/>	rh8_vm	i-052cd6fd1db5e4114	Stopped	t2.micro	-	No alarms
<input type="checkbox"/>	u20_vm	i-0119d3a3b1beafcfc	Stopped	t2.micro	-	No alarms

To start my Ubuntu 20 instance, I will ensure that my **u20_vm** is selected (checkbox), then I will click the **Instance state** drop-down menu and click **Start instance**.

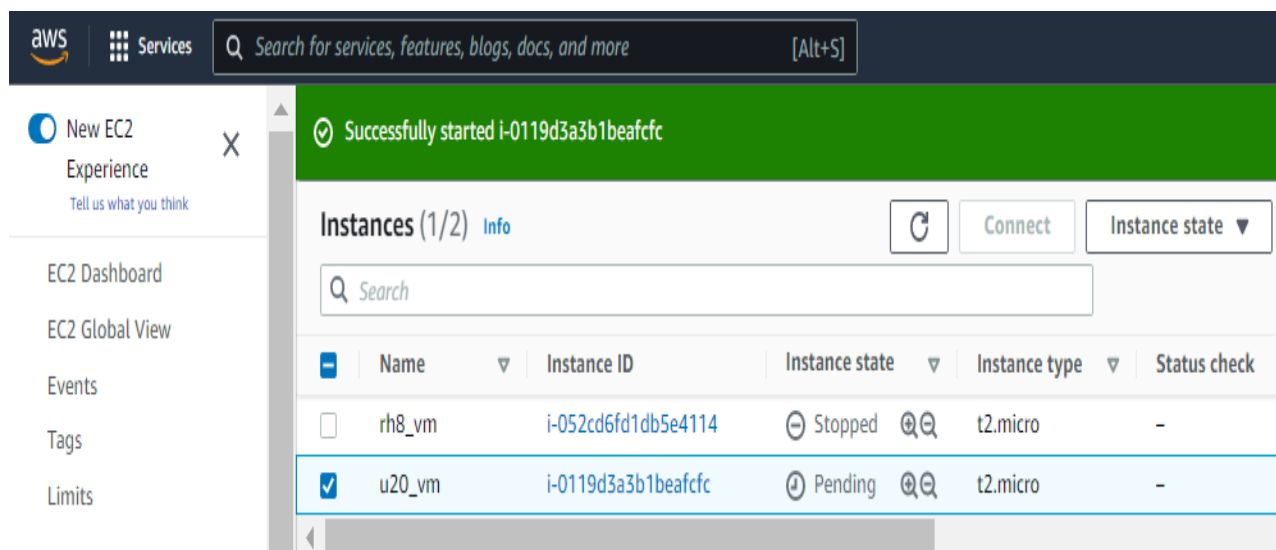


This screenshot shows the same AWS Management Console interface, but with the 'Instance state' dropdown menu open for the 'u20_vm' instance. The 'Start instance' option is highlighted with a red box. The 'u20_vm' instance is also selected with a checkbox.

<input checked="" type="checkbox"/>	Name	Instance ID	Instance state	Instance type
<input type="checkbox"/>	rh8_vm	i-052cd6fd1db5e4114	Stopped	t2.micro
<input checked="" type="checkbox"/>	u20_vm	i-0119d3a3b1beafcfc	Stopped	t2.micro

- Stop instance
- Start instance**
- Reboot instance
- Hibernate instance
- Terminate instance

The instance will start and be given new connection details.



This screenshot shows the AWS Management Console after the instance has been started. A green banner at the top displays the message 'Successfully started i-0119d3a3b1beafcfc'. Below this, the 'u20_vm' instance is now in the 'Pending' state, as indicated by the 'Instance state' column in the table.

<input checked="" type="checkbox"/>	Name	Instance ID	Instance state	Instance type	Status check
<input type="checkbox"/>	rh8_vm	i-052cd6fd1db5e4114	Stopped	t2.micro	-
<input checked="" type="checkbox"/>	u20_vm	i-0119d3a3b1beafcfc	Pending	t2.micro	-

Ensure your new instance is selected (**u20_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.

Successfully started i-0119d3a3b1beafcfc

Instances (1/2) Info Refresh Connect Instance state ▼ Actions ▼ Launch Instances

Search < 1 >

	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
<input type="checkbox"/>	rh8_vm	i-052cd6fd1db5e4114	Stopped	t2.micro	-	No alarms +	us-east-1b
<input checked="" type="checkbox"/>	u20_vm	i-0119d3a3b1beafcfc	Running	t2.micro	-	No alarms +	us-east-1b

Instance: i-0119d3a3b1beafcfc (u20_vm) Settings >

Details Security Networking Storage Status checks Monitoring Tags

▼ Instance summary Info

Instance ID i-0119d3a3b1beafcfc (u20_vm)	Public IPv4 address 3.238.54.41 open address	Private IPv4 addresses 172.31.4.185
IPv6 address -	Instance state Running	Public IPv4 DNS ec2-3-238-54-41.compute-1.amazonaws.com open address

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

Instance: i-0119d3a3b1beafcfc (u20_vm) =

Details **Security** Networking Storage Status checks Monitoring Tags

▼ Security details

IAM Role -	Owner ID 572092638768
Security groups sg-02520767438262038 (security-group1)	

Now we can register 2 free domain names. We can also create DNS records using the Public IPv4 address of the running Ubuntu 20 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: **u20-nginx-site1.tk** & **u20-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:

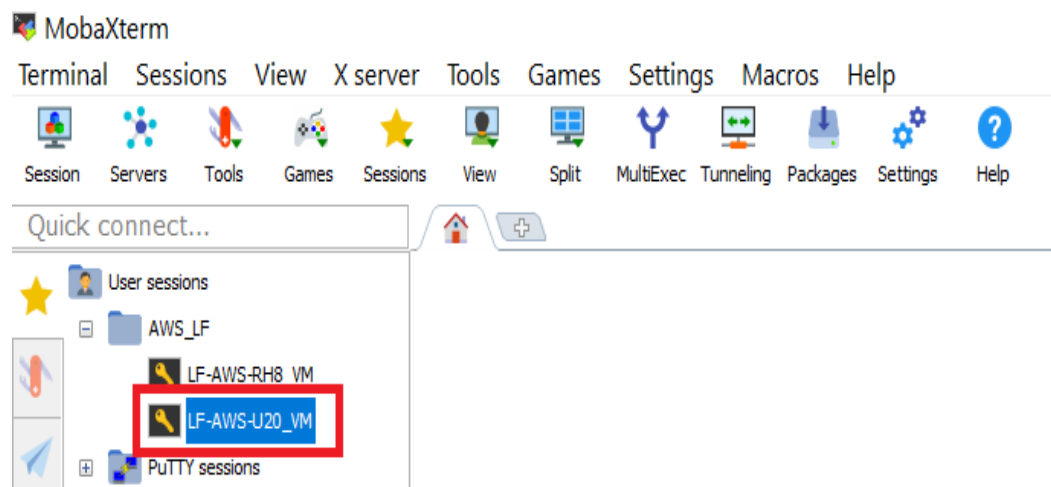
Name	Type	TTL	Target
	A	3600	3.238.54.41
WWW	CNAME	3600	u20-nginx-site1.tk

Name	Type	TTL	Target
	A	3600	3.238.54.41
WWW	CNAME	3600	u20-nginx-site2.tk

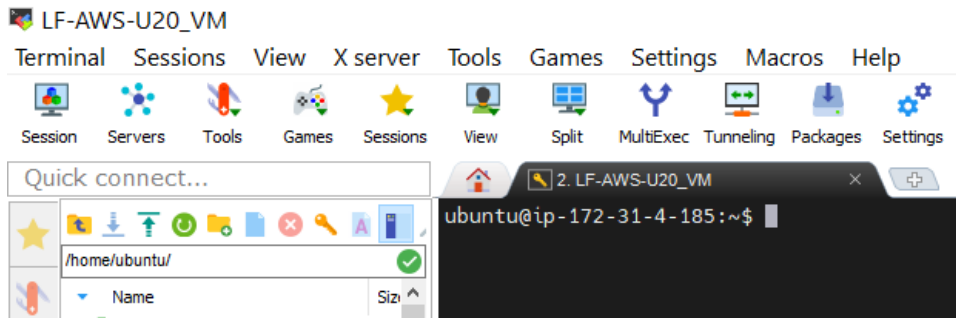
Connect to Ubuntu 20

As I mentioned in my **Create AWS Ubuntu 20 EC2 Instance** tutorial, accessible [here](#), I am using **MobaXterm Portable** as my SSH client. Since I restarted my Ubuntu 20 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 Ubuntu 20 instance.

After updating my saved SSH connection, from the MobaXterm main interface, I double clicked my SSH session to connect to my Ubuntu 20 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/u20-nginx-site1.tk/html
```

Next, create a basic html page: `/var/www/u20-nginx-site1.tk/html/index.html`

This page will be rendered when we access the site.

```
<html>
  <head>
    <title>Welcome to my Ubuntu 20 Nginx Test Site #1!</title>
  </head>
  <body>
    <p>It works! Thanks for visiting my
      <b><span style="color:blue">Ubuntu 20 Nginx Test Site #1</span></b>!
    </p>
  </body>
</html>
```

```
ubuntu@ip-172-31-4-185:~$
ubuntu@ip-172-31-4-185:~$ sudo mkdir -p /var/www/u20-nginx-site1.tk/html
ubuntu@ip-172-31-4-185:~$
ubuntu@ip-172-31-4-185:~$ cat /var/www/u20-nginx-site1.tk/html/index.html
<html>
  <head>
    <title>Welcome to my Ubuntu 20 Nginx Test Site #1!</title>
  </head>
  <body>
    <p>It works! Thanks for visiting my
      <b>
        <span style="color:blue">Ubuntu 20 Nginx Test Site #1</span>
      </b>!
    </p>
  </body>
</html>
```

Next, create the following file: `/etc/nginx/sites-available/u20-nginx-site1`

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

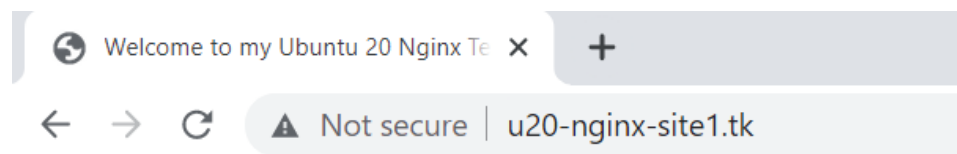
*(NOTE: your **domain names** will be different)*

```
server {
    listen 80;
    server_name u20-nginx-site1.tk www.u20-nginx-site1.tk;
    root /var/www/u20-nginx-site1.tk/html;
}
```

```
ubuntu@ip-172-31-4-185:~$ cat /etc/nginx/sites-available/u20-nginx-site1
server {
    listen 80;
    server_name u20-nginx-site1.tk www.u20-nginx-site1.tk;
    root /var/www/u20-nginx-site1.tk/html;
}
ubuntu@ip-172-31-4-185:~$
```

In my **Ubuntu 20 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://u20-nginx-site1.tk> OR <http://www.u20-nginx-site1.tk>



It works! Thanks for visiting my **Ubuntu 20 Nginx Test Site #1 !**

Now we will configure our second website.

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

```
$ sudo mkdir -p /var/www/u20-nginx-site2.tk/html
```

Next, create a basic html page: `/var/www/u20-nginx-site2.tk/html/index.html`

This page will be rendered when we access the site.

```
<html>
  <head>
    <title>Welcome to my Ubuntu 20 Nginx Test Site #2!</title>
  </head>
  <body>
    <p>It works! Thanks for visiting my
      <b><span style="color:green">Ubuntu 20 Nginx Test Site #2</span></b>!
    </p>
  </body>
</html>
```

Next, create the following file: `/etc/nginx/sites-available/u20-nginx-site2`

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 u20-nginx-site2.tk www.u20-nginx-site2.tk;
    root /var/www/u20-nginx-site2.tk/html;
}
```

```
ubuntu@ip-172-31-4-185:~$
ubuntu@ip-172-31-4-185:~$ sudo mkdir -p /var/www/u20-nginx-site2.tk/html
ubuntu@ip-172-31-4-185:~$
ubuntu@ip-172-31-4-185:~$ sudo cat /var/www/u20-nginx-site2.tk/html/index.html
<html>
  <head>
    <title>Welcome to my Ubuntu 20 Nginx Test Site #2!</title>
  </head>
  <body>
    <p>It works! Thanks for visiting my
      <b>
        <span style="color:green">Ubuntu 20 Nginx Test Site #2</span>
      </b>!
    </p>
  </body>
</html>
ubuntu@ip-172-31-4-185:~$ cat /etc/nginx/sites-available/u20-nginx-site2
server {
    listen 80;
    server_name u20-nginx-site2.tk www.u20-nginx-site2.tk;
    root /var/www/u20-nginx-site2.tk/html;
}
ubuntu@ip-172-31-4-185:~$ █
```

Link this file with the Nginx enabled sites directory to ensure that this website will be served when accessed.

```
$ sudo ln -s /etc/nginx/sites-available/u20-nginx-site2 /etc/nginx/sites-enabled/
```

Verify that the server block syntax is valid with the following:

```
$ sudo nginx -t
```

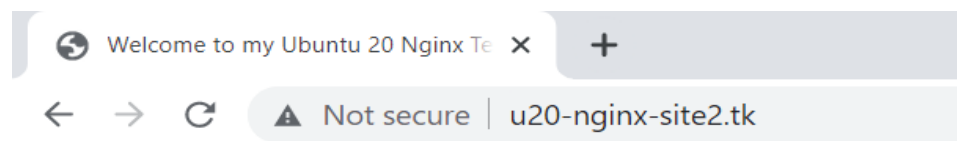
Now, restart the Nginx service

```
$ sudo systemctl restart nginx
```

```
ubuntu@ip-172-31-4-185:~$
ubuntu@ip-172-31-4-185:~$ sudo ln -s /etc/nginx/sites-available/u20-nginx-site2 /etc/nginx/sites-enabled/
ubuntu@ip-172-31-4-185:~$
ubuntu@ip-172-31-4-185:~$ sudo nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
ubuntu@ip-172-31-4-185:~$
ubuntu@ip-172-31-4-185:~$ sudo systemctl restart nginx
ubuntu@ip-172-31-4-185:~$ █
```

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

<http://u20-nginx-site2.tk> OR <http://www.u20-nginx-site2.tk>



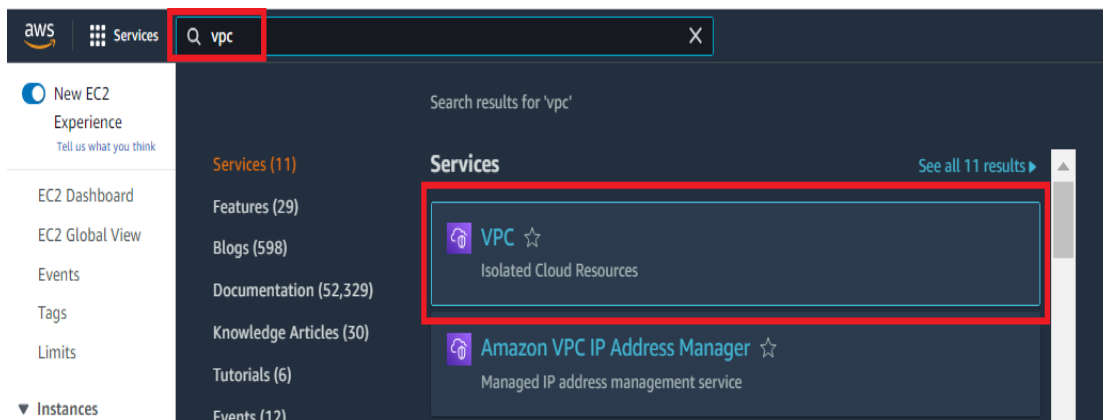
It works! Thanks for visiting my **Ubuntu 20 Nginx Test Site #2** !

Please note that if you shutdown your Ubuntu 20 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.

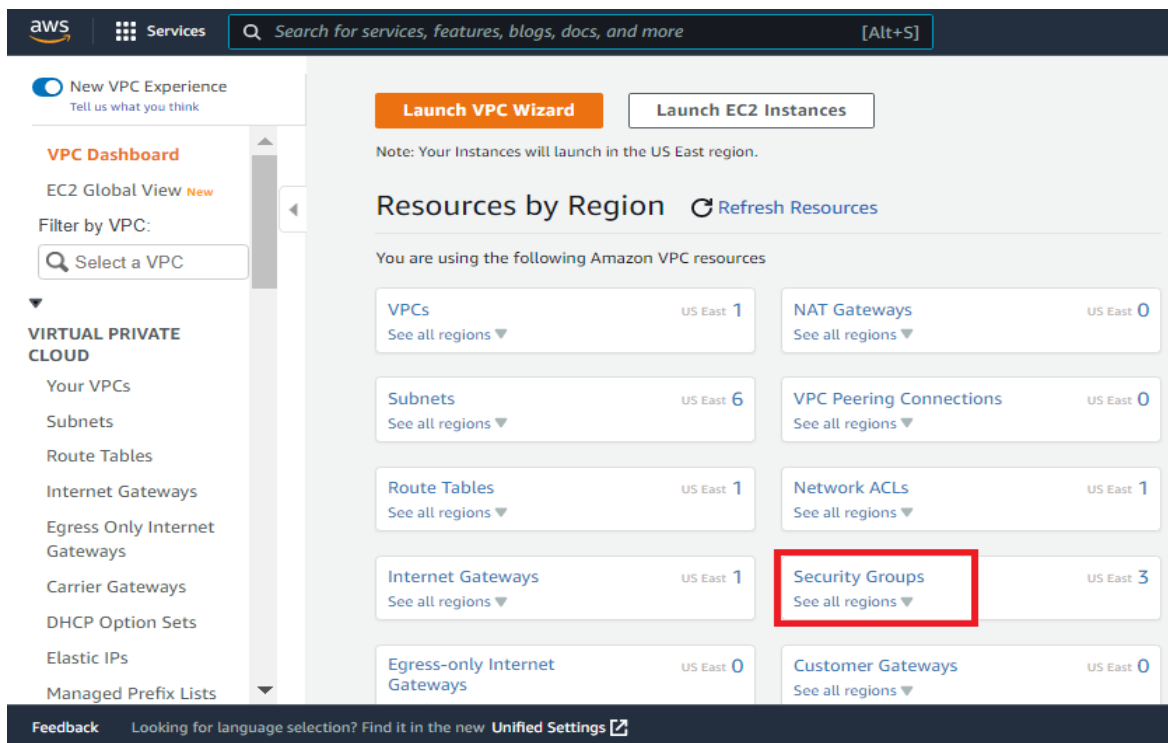
We have successfully configured Nginx for multisite hosting, but the websites are not secured with SSL certificates. To do this, first, we will need to open a port in our firewall to be able to access the secured websites. We will configure our [VPC \(Virtual Private Cloud\) Security Group](#) to allow HTTPS traffic through the firewall. This will allow HTTPS access (port 443) to the Nginx web server that will be running on our Ubuntu 20 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 Ubuntu 20 EC2 instance.

The screenshot shows the AWS Management Console 'Security Groups' page. The left sidebar contains navigation links for 'VPC Dashboard', 'EC2 Global View', and 'Filter by VPC'. The main content area displays a table of security groups. The 'Security group ID' column is highlighted with a red box, showing the ID 'sg-02520767438262038' for the security group named 'security-group1'.

Name	Security group ID	Security group name	VPC ID
-	sg-02520767438262038	security-group1	vpc-01ac0d160f388c36e
-	sg-07d39b68f2403b7c8	default	vpc-01ac0d160f388c36e
-	sg-0a1d0061cebed314b	security-group2	vpc-01ac0d160f388c36e

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

The screenshot shows the AWS Management Console 'Security Group' details page for 'sg-02520767438262038 - security-group1'. The 'Inbound rules' tab is selected. The 'Details' section shows the security group name, ID, description, VPC ID, owner, and rule counts. Below the details, there is a notification bar about the Reachability Analyzer. The 'Inbound rules (2)' section shows a table of rules. The 'Edit inbound rules' button is highlighted with a red box.

Details

Security group name	Security group ID	Description	VPC ID
security-group1	sg-02520767438262038	Limit access to instance	vpc-01ac0d160f388c36e

Owner	Inbound rules count	Outbound rules count
572092638768	1 Permission entry	1 Permission entry

Inbound rules (2)

Name	Security group rule...	IP version	Type	Protocol	Port range
-	sgr-0be85105ca435d1...	IPv4	HTTP	TCP	80
-	sgr-05aa0248d0edfc4eb	IPv4	SSH	TCP	22

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

Edit inbound rules [Info](#)

Inbound rules control the incoming traffic that's allowed to reach the instance.

Security group rule ID	Type Info	Protocol Info	Port range Info	Source Info	Description - optional Info	
sgr-0be85105ca435d1e8	HTTP	TCP	80	Custom <input type="text" value="Q"/>		Delete
				<input type="text" value="0.0.0.0/0"/>		
sgr-05aa0248d0edfc4eb	SSH	TCP	22	Custom <input type="text" value="Q"/>		Delete
				<input type="text" value="0.0.0.0/0"/>		

Add rule

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

-	HTTPS	TCP	443	Anywh... <input type="text" value="Q"/>		Delete
				<input type="text" value="0.0.0.0/0"/>		

Add rule

Cancel **Preview changes** **Save rules**

Inbound rules | Outbound rules | Tags

You can now check network connectivity with Reachability Analyzer [Run Reachability Analyzer](#)

Inbound rules (3)

< 1 > [Manage tags](#) [Edit inbound rules](#)

	Name	Security group rule...	IP version	Type	Protocol	Port range
<input type="checkbox"/>	-	sgr-0be85105ca435d1...	IPv4	HTTP	TCP	80
<input type="checkbox"/>	-	sgr-0ee1cf6505bb75643	IPv4	HTTPS	TCP	443
<input type="checkbox"/>	-	sgr-05aa0248d0edfc4eb	IPv4	SSH	TCP	22

Now that we've opened the firewall for HTTPS access, we can connect to the Ubuntu 20 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.

```
$ sudo apt 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 u20-nginx-site1.tk -d www.u20-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).

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.

```
ubuntu@ip-172-31-4-185:~$  
ubuntu@ip-172-31-4-185:~$ sudo certbot --nginx -d u20-nginx-site1.tk -d www.u20-nginx-site1.tk  
Saving debug log to /var/log/letsencrypt/letsencrypt.log  
Plugins selected: Authenticator nginx, Installer nginx  
Enter email address (used for urgent renewal and security notices) (Enter 'c' to  
cancel): liams.systems@gmail.com  
  
-----  
Please read the Terms of Service at  
https://letsencrypt.org/documents/LE-SA-v1.2-November-15-2017.pdf. You must  
agree in order to register with the ACME server at  
https://acme-v02.api.letsencrypt.org/directory  
-----  
(A)gree/(C)ancel: A
```

If that's successful, certbot will ask how you'd like to configure your HTTPS settings. I chose to have all requests redirect to secure HTTPS by entering 2 when prompted.

```
(Y)es/(N)o: Y  
Obtaining a new certificate  
Performing the following challenges:  
http-01 challenge for u20-nginx-site1.tk  
http-01 challenge for www.u20-nginx-site1.tk  
Waiting for verification...  
Cleaning up challenges  
Deploying Certificate to VirtualHost /etc/nginx/sites-enabled/u20-nginx-site1  
Deploying Certificate to VirtualHost /etc/nginx/sites-enabled/u20-nginx-site1  
  
Please choose whether or not to redirect HTTP traffic to HTTPS, removing HTTP access.  
-----  
1: No redirect - Make no further changes to the webserver configuration.  
2: Redirect - Make all requests redirect to secure HTTPS access. Choose this for  
new sites, or if you're confident your site works on HTTPS. You can undo this  
change by editing your web server's configuration.  
-----  
Select the appropriate number [1-2] then [enter] (press 'c' to cancel): 2
```

I have successfully obtained an SSL certificate for `u20-nginx-site1.tk` & `www.u20-nginx-site1.tk`

```
Select the appropriate number [1-2] then [enter] (press 'c' to cancel): 2
Redirecting all traffic on port 80 to ssl in /etc/nginx/sites-enabled/u20-nginx-site1
Redirecting all traffic on port 80 to ssl in /etc/nginx/sites-enabled/u20-nginx-site1

Congratulations! You have successfully enabled https://u20-nginx-site1.tk and
https://www.u20-nginx-site1.tk

You should test your configuration at:
https://www.ssllabs.com/ssltest/analyze.html?d=u20-nginx-site1.tk
https://www.ssllabs.com/ssltest/analyze.html?d=www.u20-nginx-site1.tk
-----

IMPORTANT NOTES:
- Congratulations! Your certificate and chain have been saved at:
  /etc/letsencrypt/live/u20-nginx-site1.tk/fullchain.pem
  Your key file has been saved at:
  /etc/letsencrypt/live/u20-nginx-site1.tk/privkey.pem
  Your cert will expire on 2022-08-30. To obtain a new or tweaked
  version of this certificate in the future, simply run certbot again
  with the "certonly" option. To non-interactively renew *all* of
  your certificates, run "certbot renew"
- Your account credentials have been saved in your Certbot
  configuration directory at /etc/letsencrypt. You should make a
  secure backup of this folder now. This configuration directory will
  also contain certificates and private keys obtained by Certbot so
  making regular backups of this folder is ideal.
- If you like Certbot, please consider supporting our work by:

  Donating to ISRG / Let's Encrypt: https://letsencrypt.org/donate
  Donating to EFF: https://eff.org/donate-le

ubuntu@ip-172-31-4-185:~$
```

Now, I will obtain a free Let's Encrypt SSL certificate for my second test site. When prompted for request redirect response, I will enter 2 to force all requests to be handled by secure HTTPS.

NOTE: use your domain names

```
$ sudo certbot --nginx -d u20-nginx-site2.tk -d www.u20-nginx-site2.tk
```

The certificates have 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/sites-enabled/u20-nginx-site1
```

```
$ cat /etc/nginx/sites-enabled/u20-nginx-site2
```

```
ubuntu@ip-172-31-4-185:~$
ubuntu@ip-172-31-4-185:~$ cat /etc/nginx/sites-enabled/u20-nginx-site1
server {
    server_name u20-nginx-site1.tk www.u20-nginx-site1.tk;
    root /var/www/u20-nginx-site1.tk/html;

    listen 443 ssl; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/u20-nginx-site1.tk/fullchain.pem; # managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/u20-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 ($host = www.u20-nginx-site1.tk) {
        return 301 https://$host$request_uri;
    } # managed by Certbot

    if ($host = u20-nginx-site1.tk) {
        return 301 https://$host$request_uri;
    } # managed by Certbot

    listen 80;
    server_name u20-nginx-site1.tk www.u20-nginx-site1.tk;
    return 404; # managed by Certbot
}
ubuntu@ip-172-31-4-185:~$
```

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

We can even test our newly secured websites using [SSL Server Test](#).

sslabs.com/sslltest/

Qualys. SSL Labs

Home Projects Qualys Free Trial Contact

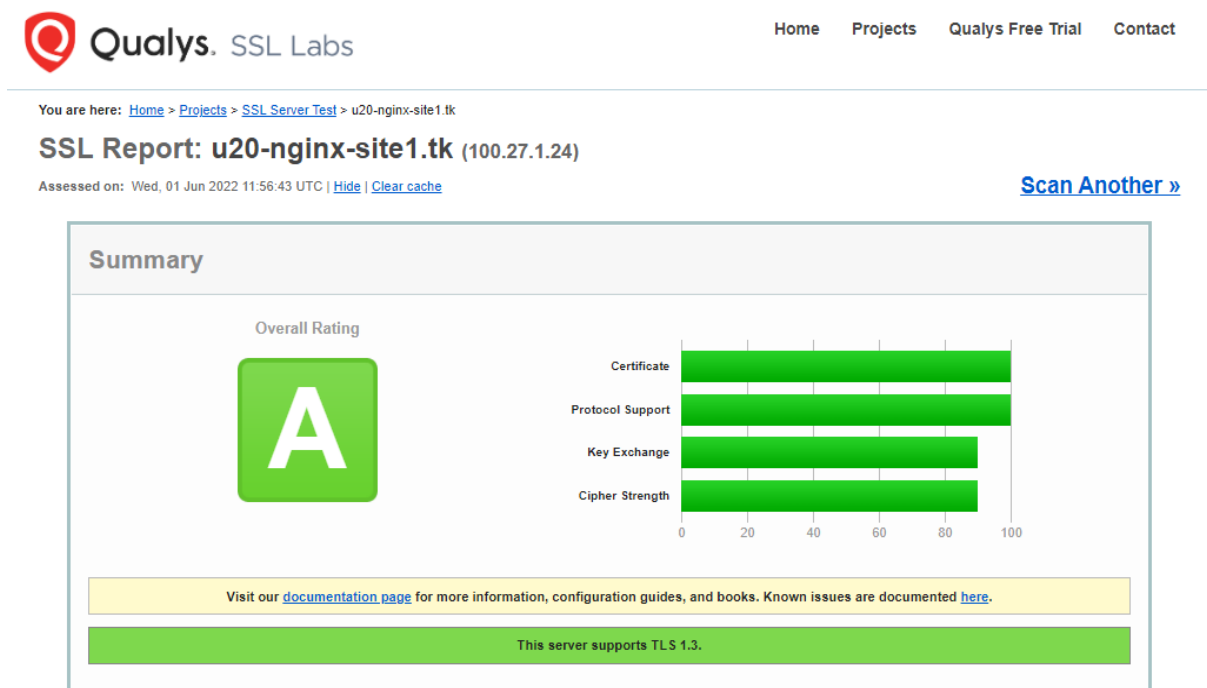
You are here: [Home](#) > [Projects](#) > SSL Server Test

SSL Server Test

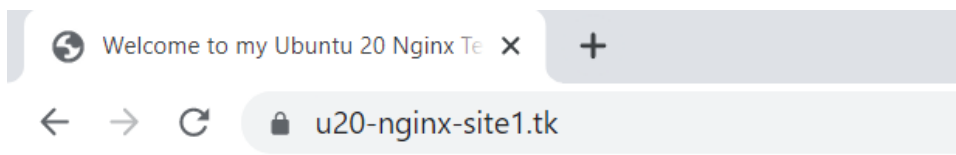
This free online service performs a deep analysis of the configuration of any SSL web server on the public Internet. **Please note that the information you submit here is used only to provide you the service. We don't use the domain names or the test results, and we never will.**

Hostname:

☐ Do not show the results on the boards

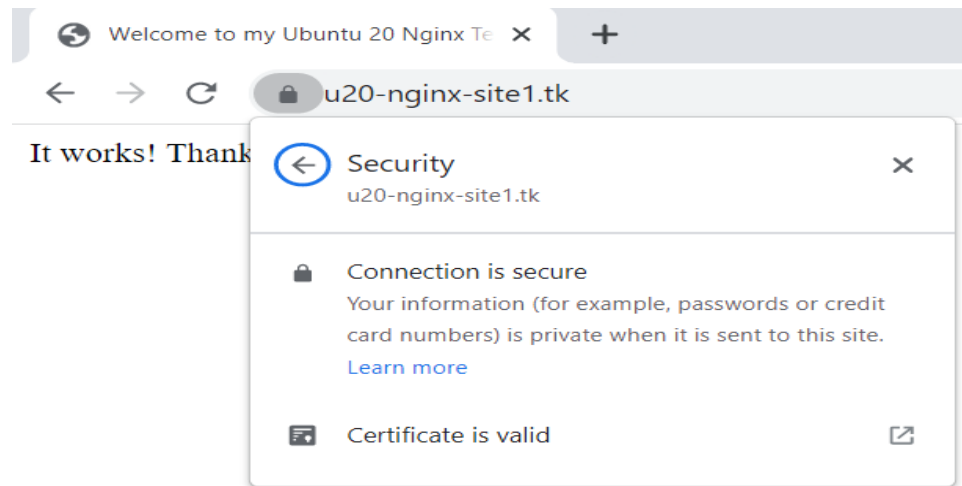
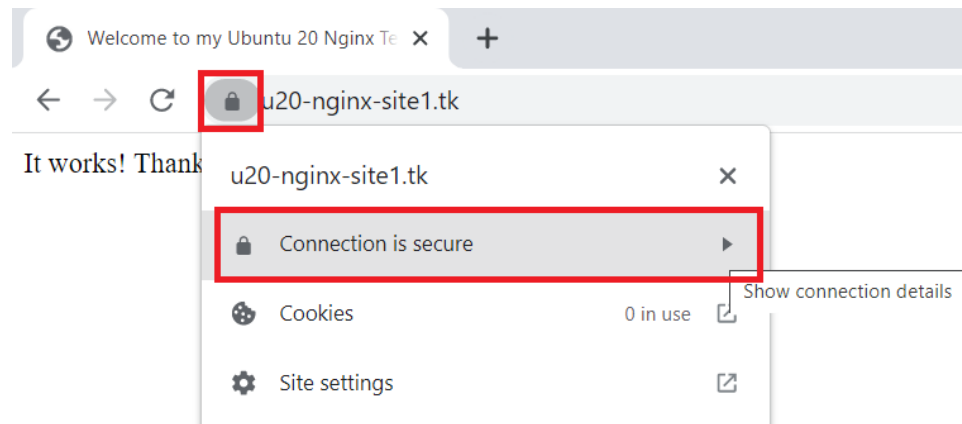


Now if I open my website, using either <http://> or <https://>, redirection will happen automatically.

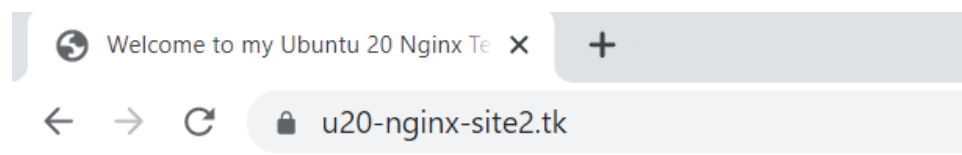


It works! Thanks for visiting my **Ubuntu 20 Nginx Test Site #1** !

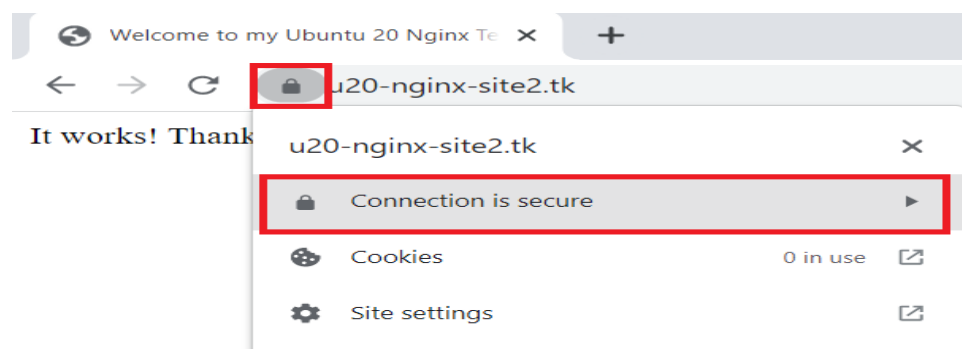
Also notice the browser's security indicator, it should indicate that the site is properly secured with a lock icon.

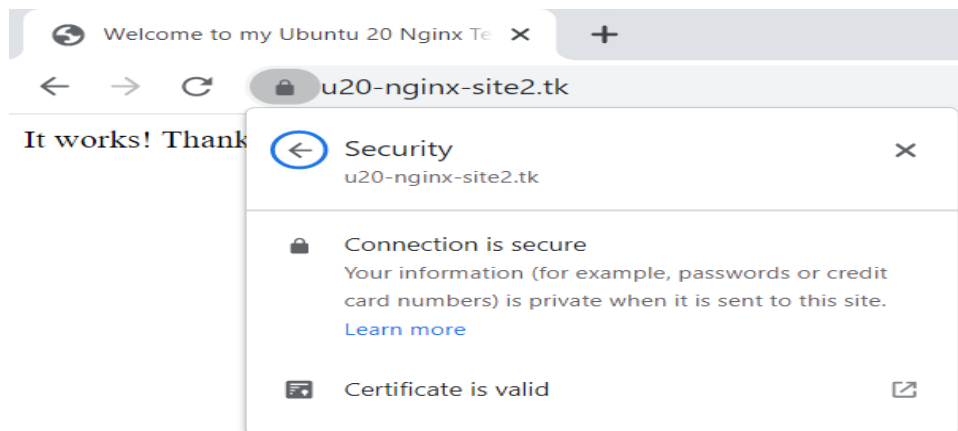


Now, for my second test site.



It works! Thanks for visiting my **Ubuntu 20 Nginx Test Site #2 !**





Let's Encrypt's certificates are only valid for ninety days. This is to encourage users to automate their certificate renewal process. The certbot package we installed takes care of this for us by adding 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.

```
$ sudo systemctl status certbot.timer
```

```
ubuntu@ip-172-31-4-185:~$  
ubuntu@ip-172-31-4-185:~$ sudo systemctl status certbot.timer  
● certbot.timer - Run certbot twice daily  
   Loaded: loaded (/lib/systemd/system/certbot.timer; enabled; vendor preset: enabled)  
   Active: active (waiting) since Wed 2022-06-01 11:31:35 UTC; 2h 7min ago  
   Trigger: Wed 2022-06-01 23:45:24 UTC; 10h left  
   Triggers: ● certbot.service  
  
Jun 01 11:31:35 ip-172-31-4-185 systemd[1]: Started Run certbot twice daily.  
ubuntu@ip-172-31-4-185:~$
```

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

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

```
ubuntu@ip-172-31-4-185:~$  
ubuntu@ip-172-31-4-185:~$ sudo certbot renew --dry-run  
Saving debug log to /var/log/letsencrypt/letsencrypt.log  
  
-----  
Processing /etc/letsencrypt/renewal/u20-nginx-site1.tk.conf  
-----  
Cert not due for renewal, but simulating renewal for dry run  
Plugins selected: Authenticator nginx, Installer nginx  
Renewing an existing certificate  
Performing the following challenges:  
http-01 challenge for u20-nginx-site1.tk  
http-01 challenge for www.u20-nginx-site1.tk  
Waiting for verification...  
Cleaning up challenges  
  
-----  
new certificate deployed with reload of nginx server; fullchain is  
/etc/letsencrypt/live/u20-nginx-site1.tk/fullchain.pem  
-----  
  
-----  
Processing /etc/letsencrypt/renewal/u20-nginx-site2.tk.conf  
-----  
Cert not due for renewal, but simulating renewal for dry run  
Plugins selected: Authenticator nginx, Installer nginx  
Renewing an existing certificate  
Performing the following challenges:  
http-01 challenge for u20-nginx-site2.tk  
http-01 challenge for www.u20-nginx-site2.tk  
Waiting for verification...  
Cleaning up challenges
```


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.

```
-----
new certificate deployed with reload of nginx server; fullchain is
/etc/letsencrypt/live/u20-nginx-site2.tk/fullchain.pem
-----

** DRY RUN: simulating 'certbot renew' close to cert expiry
**          (The test certificates below have not been saved.)

Congratulations, all renewals succeeded. The following certs have been renewed:
/etc/letsencrypt/live/u20-nginx-site1.tk/fullchain.pem (success)
/etc/letsencrypt/live/u20-nginx-site2.tk/fullchain.pem (success)
** DRY RUN: simulating 'certbot renew' close to cert expiry
**          (The test certificates above have not been saved.)
-----

IMPORTANT NOTES:
- Your account credentials have been saved in your Certbot
  configuration directory at /etc/letsencrypt. You should make a
  secure backup of this folder now. This configuration directory will
  also contain certificates and private keys obtained by Certbot so
  making regular backups of this folder is ideal.
ubuntu@ip-172-31-4-185:~$
```

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 Ubuntu 20 compute instance. Next, we secured the sites using free Let's Encrypt SSL certificates. Afterwards, we manually simulated certificate renewal. Finally, we confirmed that the systemd certbot timer service was running 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 **RHEL 8** 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](#)