

I couldn't follow the exact steps of the documentation that was provided as it required me to host a docker container and there are no free hosting platforms present after Heroku revoked its free services. So, to showcase my skills I have displayed something similar to what I was asked to do but on for a simpler use case. I used by knowledge of Cyber Security to do the following:

Steps required to install and configure the web server along with SSL/TLS certificates:

We are using Nginx to host our website because it is a lightweight web server.

1. Installing Nginx:

```
hacked@ubuntu:~$ sudo apt install nginx
[sudo] password for hacked:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libnginx-mod-http-image-filter libnginx-mod-http-xslt-filter libnginx-mod-mail libnginx-mod-stream nginx-common nginx-core
Suggested packages:
  fcgiwrap nginx-doc
The following NEW packages will be installed:
  libnginx-mod-http-image-filter libnginx-mod-http-xslt-filter libnginx-mod-mail libnginx-mod-stream nginx nginx-common nginx-core
0 upgraded, 7 newly installed, 0 to remove and 18 not upgraded.
Need to get 695 kB of archives.
After this operation, 2,134 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://us.archive.ubuntu.com/ubuntu focal-updates/main amd64 nginx-common all 1.18.0-0ubuntu1.3 [37.7 kB]
Get:2 http://us.archive.ubuntu.com/ubuntu focal-updates/main amd64 libnginx-mod-http-image-filter amd64 1.18.0-0ubuntu1.3 [14.3 kB]
Get:3 http://us.archive.ubuntu.com/ubuntu focal-updates/main amd64 libnginx-mod-http-xslt-filter amd64 1.18.0-0ubuntu1.3 [13.0 kB]
Get:4 http://us.archive.ubuntu.com/ubuntu focal-updates/main amd64 libnginx-mod-mail amd64 1.18.0-0ubuntu1.3 [42.8 kB]
Get:5 http://us.archive.ubuntu.com/ubuntu focal-updates/main amd64 libnginx-mod-stream amd64 1.18.0-0ubuntu1.3 [67.3 kB]
Get:6 http://us.archive.ubuntu.com/ubuntu focal-updates/main amd64 nginx-core amd64 1.18.0-0ubuntu1.3 [425 kB]
Get:7 http://us.archive.ubuntu.com/ubuntu focal-updates/main amd64 nginx all 1.18.0-0ubuntu1.3 [3,629 B]
Fetched 695 kB in 36s (17.0 kB/s)
Preconfiguring packages ...
Selecting previously unselected package nginx-common.
(Reading database ... 161246 files and directories currently installed.)
Preparing to unpack .../0-nginx-common_1.18.0-0ubuntu1.3_all.deb ...
Unpacking nginx-common (1.18.0-0ubuntu1.3) ...
Selecting previously unselected package libnginx-mod-http-image-filter.
Preparing to unpack .../1-libnginx-mod-http-image-filter_1.18.0-0ubuntu1.3_amd64.deb ...
Unpacking libnginx-mod-http-image-filter (1.18.0-0ubuntu1.3) ...
Selecting previously unselected package libnginx-mod-http-xslt-filter.
Preparing to unpack .../2-libnginx-mod-http-xslt-filter_1.18.0-0ubuntu1.3_amd64.deb ...
Unpacking libnginx-mod-http-xslt-filter (1.18.0-0ubuntu1.3) ...
Selecting previously unselected package libnginx-mod-mail.
Preparing to unpack .../3-libnginx-mod-mail_1.18.0-0ubuntu1.3_amd64.deb ...
Unpacking libnginx-mod-mail (1.18.0-0ubuntu1.3) ...
Selecting previously unselected package libnginx-mod-stream.
Preparing to unpack .../4-libnginx-mod-stream_1.18.0-0ubuntu1.3_amd64.deb ...
Unpacking libnginx-mod-stream (1.18.0-0ubuntu1.3) ...
Selecting previously unselected package nginx-core.
Preparing to unpack .../5-nginx-core_1.18.0-0ubuntu1.3_amd64.deb ...
Unpacking nginx-core (1.18.0-0ubuntu1.3) ...
Selecting previously unselected package nginx.
Preparing to unpack .../6-nginx_1.18.0-0ubuntu1.3_all.deb ...
Unpacking nginx (1.18.0-0ubuntu1.3) ...
Setting up nginx-common (1.18.0-0ubuntu1.3) ...
Created symlink /etc/systemd/system/multi-user.target.wants/nginx.service → /lib/systemd/system/nginx.service.
Setting up libnginx-mod-http-xslt-filter (1.18.0-0ubuntu1.3) ...
Setting up libnginx-mod-mail (1.18.0-0ubuntu1.3) ...
Setting up libnginx-mod-http-image-filter (1.18.0-0ubuntu1.3) ...
Setting up libnginx-mod-stream (1.18.0-0ubuntu1.3) ...
Setting up nginx-core (1.18.0-0ubuntu1.3) ...
Setting up nginx (1.18.0-0ubuntu1.3) ...
Processing triggers for systemd (245.4-4ubuntu3.17) ...
Processing triggers for man-db (2.9.1-1) ...
Processing triggers for libc-bin (2.31-0ubuntu1)
```

```
hacked@ubuntu:~$ sudo ufw app list
```

```
[sudo] password for hacked:
```

```
Available applications:
```

```
CUPS
Nginx Full
Nginx HTTP
Nginx HTTPS
OpenSSH
```

```
hacked@ubuntu:~$ sudo ufw status
```

```
Status: inactive
```

```
hacked@ubuntu:~$ sudo ufw enable
```

```
Command may disrupt existing ssh connections. Proceed with operation (y|n)? y
```

2. After installing Nginx, we need to configure the firewall to allow

access to the website. I enabled ufw and then allowed HTTP(port 80) and HTTPS(port 443).(This was the section where we faced a glitch during our demo as the HTTPS rule got deleted when I tried to enable Nginx Full. We fixed it by deleting Nginx Full and allowing only Nginx HTTP and HTTPS)

```
● hacked@fcs01:~$ sudo ufw status
Status: active

To Action From
--
OpenSSH ALLOW Anywhere
OpenSSH (v6) ALLOW Anywhere (v6)
```

```
● hacked@fcs01:~$ sudo ufw allow 'Nginx HTTP'
Rule added
Rule added (v6)
● hacked@fcs01:~$ sudo ufw allow 'Nginx HTTPS'
Rule added
Rule added (v6)
● hacked@fcs01:~$ sudo ufw status
Status: active

To Action From
--
OpenSSH ALLOW Anywhere
Nginx HTTP ALLOW Anywhere
Nginx HTTPS ALLOW Anywhere
OpenSSH (v6) ALLOW Anywhere (v6)
Nginx HTTP (v6) ALLOW Anywhere (v6)
Nginx HTTPS (v6) ALLOW Anywhere (v6)
```

3. After finishing the configuration, we check whether nginx is running properly or not. It was configured properly and was active.

```
● hacked@fcs01:~$ systemctl status nginx
● nginx.service - A high performance web server and a reverse proxy server
   Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset: enabled)
   Active: active (running) since Fri 2022-09-30 14:28:17 UTC; 1 day 1h ago
     Docs: man:nginx(8)
  Process: 398207 ExecStartPre=/usr/sbin/nginx -t -q -g daemon on; master_process on; (code=exited, status=0/SUCCESS)
  Process: 398222 ExecStart=/usr/sbin/nginx -g daemon on; master_process on; (code=exited, status=0/SUCCESS)
  Process: 432622 ExecReload=/usr/sbin/nginx -g daemon on; master_process on; -s reload (code=exited, status=0/SUCCESS)
 Main PID: 398224 (nginx)
    Tasks: 3 (limit: 2273)
   Memory: 6.7M
    CGroup: /system.slice/nginx.service
            └─398224 nginx: master process /usr/sbin/nginx -g daemon on; master_process on;
               └─432623 nginx: worker process
                  └─432624 nginx: worker process
```

4. Now, we have to create a self-signed key and certificate pair. We used OpenSSL with the following parameters and common name as the IP address. We also needed to ensure that our session keys would never be compromised and hence ensure Forward Secrecy.

```

hacked@ubuntu:~$ sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/ssl/private/nginx-selfsigned.key -out /etc/ssl/certs/nginx-selfsigned.crt
Generating a RSA private key
.....+++++
writing new private key to '/etc/ssl/private/nginx-selfsigned.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:IN
State or Province Name (full name) [Some-State]:
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:192.168.2.234
Email Address []:

hacked@ubuntu:~$ sudo openssl dhparam -out /etc/ssl/certs/dhparam.pem 2048
Generating DH parameters, 2048 bit long safe prime, generator 2
This is going to take a long time
.....

```

- Created configuration snippet for the SSL Key and Certificate with strong encryption settings by setting `ssl_certificate` file to our certificate, `ssl_certificate_key` to our SSL key and using recommendations on the Cipherli.st site.

```

hacked@ubuntu:~$ sudo nano /etc/nginx/snippets/self-signed.conf

GNU nano 4.8 /etc/nginx/snippets/self-signed.conf
ssl_certificate /etc/ssl/certs/nginx-selfsigned.crt;
ssl_certificate_key /etc/ssl/private/nginx-selfsigned.key;

hacked@ubuntu:/etc/nginx/snippets$ sudo nano /etc/nginx/snippets/ssl-params.conf

GNU nano 4.8 ssl-params.conf
from https://cipherli.st/
# and https://raymii.org/s/tutorials/Strong_SSL_Security_On_nginx.html

ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
ssl_prefer_server_ciphers on;
ssl_ciphers "EECDH+AESGCM:EDH+AESGCM:AES256+EECDH:AES256+EDH";
ssl_ecdh_curve secp384r1;
ssl_session_cache shared:SSL:10m;
ssl_session_tickets off;
ssl_stapling on;
ssl_stapling_verify on;
resolver 8.8.8.8 8.8.4.4 valid=300s;
resolver_timeout 5s;
# Disable preloading HSTS for now. You can use the commented out header line that includes
# the "preload" directive if you understand the implications.
#add_header Strict-Transport-Security "max-age=63072000; includeSubdomains; preload";
add_header Strict-Transport-Security "max-age=63072000; includeSubdomains";
add_header X-Frame-Options DENY;
add_header X-Content-Type-Options nosniff;

ssl_dhparam /etc/ssl/certs/dhparam.pem;

```

- Adjusted Nginx configuration to enable SSL. Made changes to the default server block file by adding a `server_name` directive and setting it to our server's IP address and uncommenting the `listen 443` line to use HTTPS. I also added a redirection snippet so that our site redirects to HTTPS even when it is open via HTTP connection.(a little hiccup was faced here as the redirection was stuck in an infinite loop and was resolved using scheme detection).

```

hacked@ubuntu:/etc/nginx$ sudo nano /etc/nginx/sites-available/default

```

```
##
# Default server configuration
#
server {
    listen 80 default_server;
    listen [::]:80 default_server;
    server_name 192.168.2.234;
    if ($scheme = "http") {
        return 302 https://$server_name$request_uri;
    }
    # SSL configuration
    listen 443 ssl default_server;
    listen [::]:443 ssl default_server;
    include snippets/self-signed.conf;
    include snippets/ssl-params.conf;
    #
    # Note: You should disable gzip for SSL traffic.
    # See: https://bugs.debian.org/773332
    #
    # Read up on ssl_ciphers to ensure a secure configuration.
    # See: https://bugs.debian.org/765782
    #
    # Self signed certs generated by the ssl-cert package
    # Don't use them in a production server!
    #
    # include snippets/snakeoil.conf;

    root /var/www/html;

    # Add index.php to the list if you are using PHP
    index index.html index.htm index.nginx-debian.html;
}
```

7. Then I changed the index.html file to display simple text and our group number. Finally our website was deployed

```
hacked@fcs01:/$ cd var
hacked@fcs01:/var$ ls
backups cache crash lib local lock log mail opt run snap spool tmp www
hacked@fcs01:/var$ cd www
hacked@fcs01:/var/www$ ls
html
hacked@fcs01:/var/www$ cd html
hacked@fcs01:/var/www/html$ ls
404.html index.html index.nginx-debian.html
hacked@fcs01:/var/www/html$ sudo nano index.html
```



