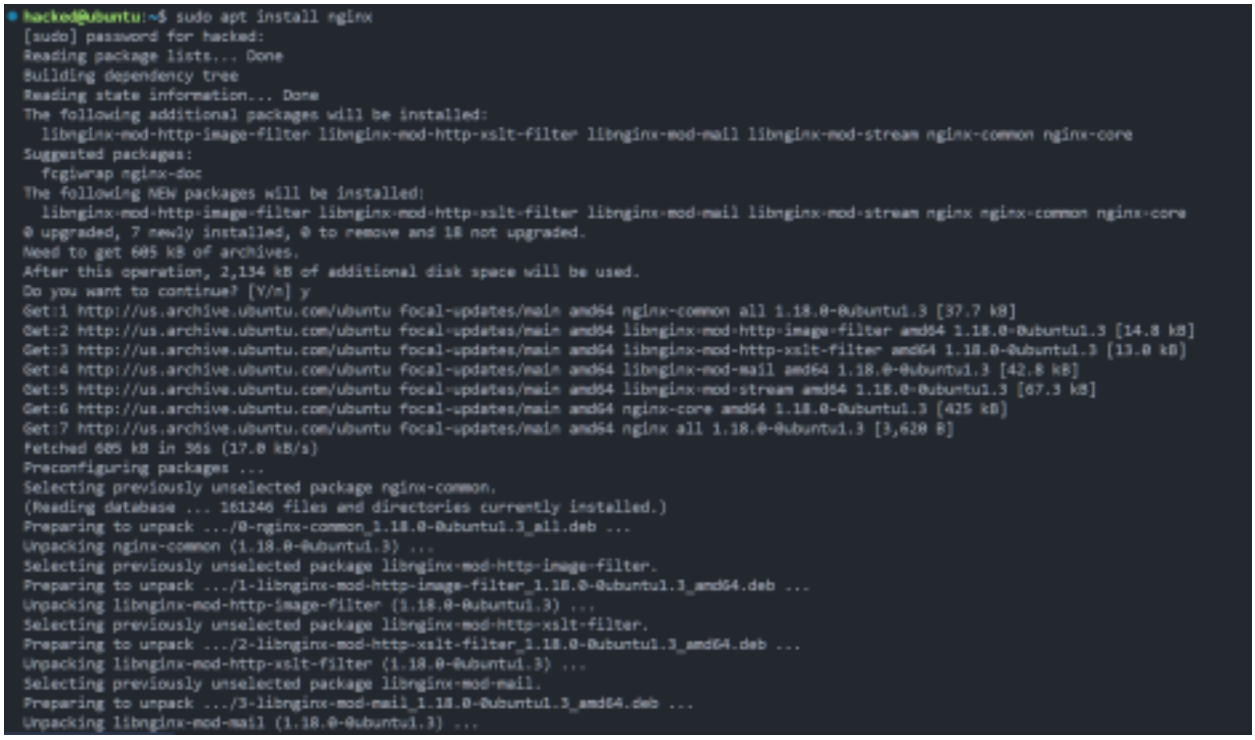
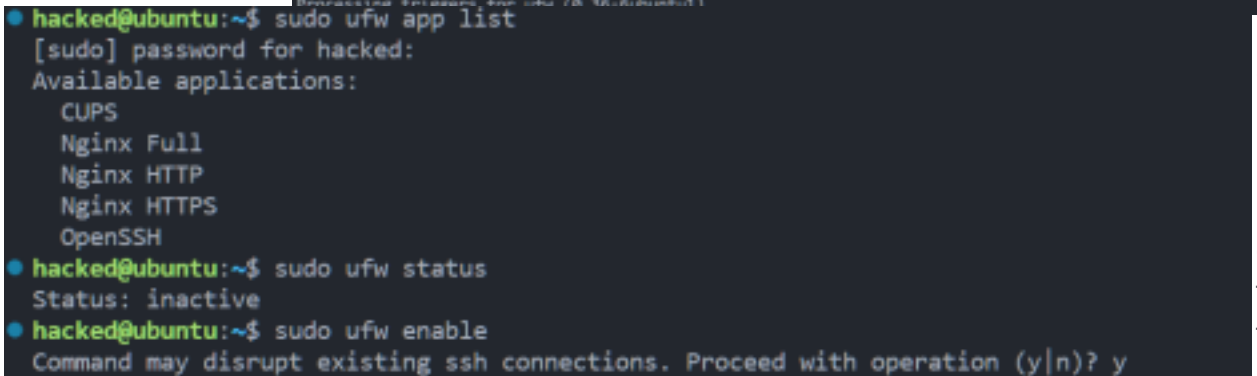
I couldn't follow the exact steps of the documentation that was provided as it required me to host a docker container and there are no free hosting platforms present after Heroku revoked its free services. So, to showcase my skills I have displayed something similar to what I was asked to do but on for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




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)
```

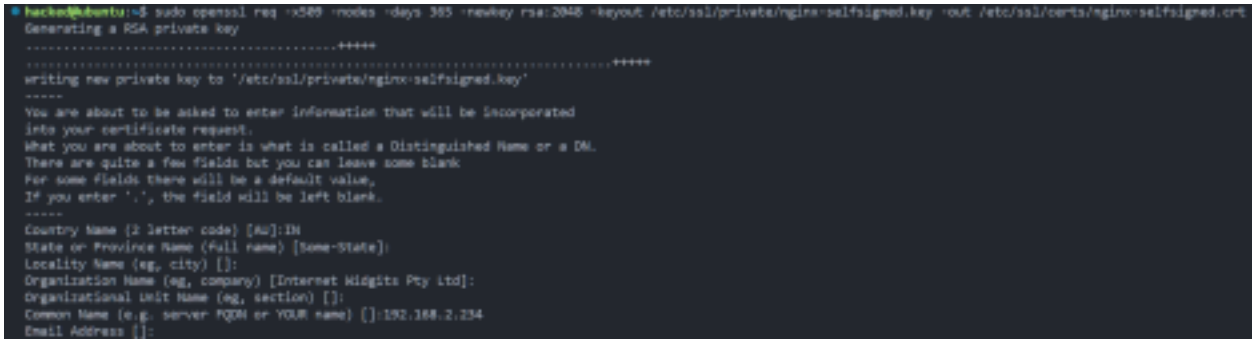
```
hacked@fcs01:~$ sudo ufw allow 'Nginx HTTP'
Rule added
Rule added (v6)
hacked@fcs01:~$ sudo ufw allow 'Nginx HTTPS'
Rule added
Rule added (v6)
hacked@fcs01:~$ sudo ufw status
Status: active

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)
```

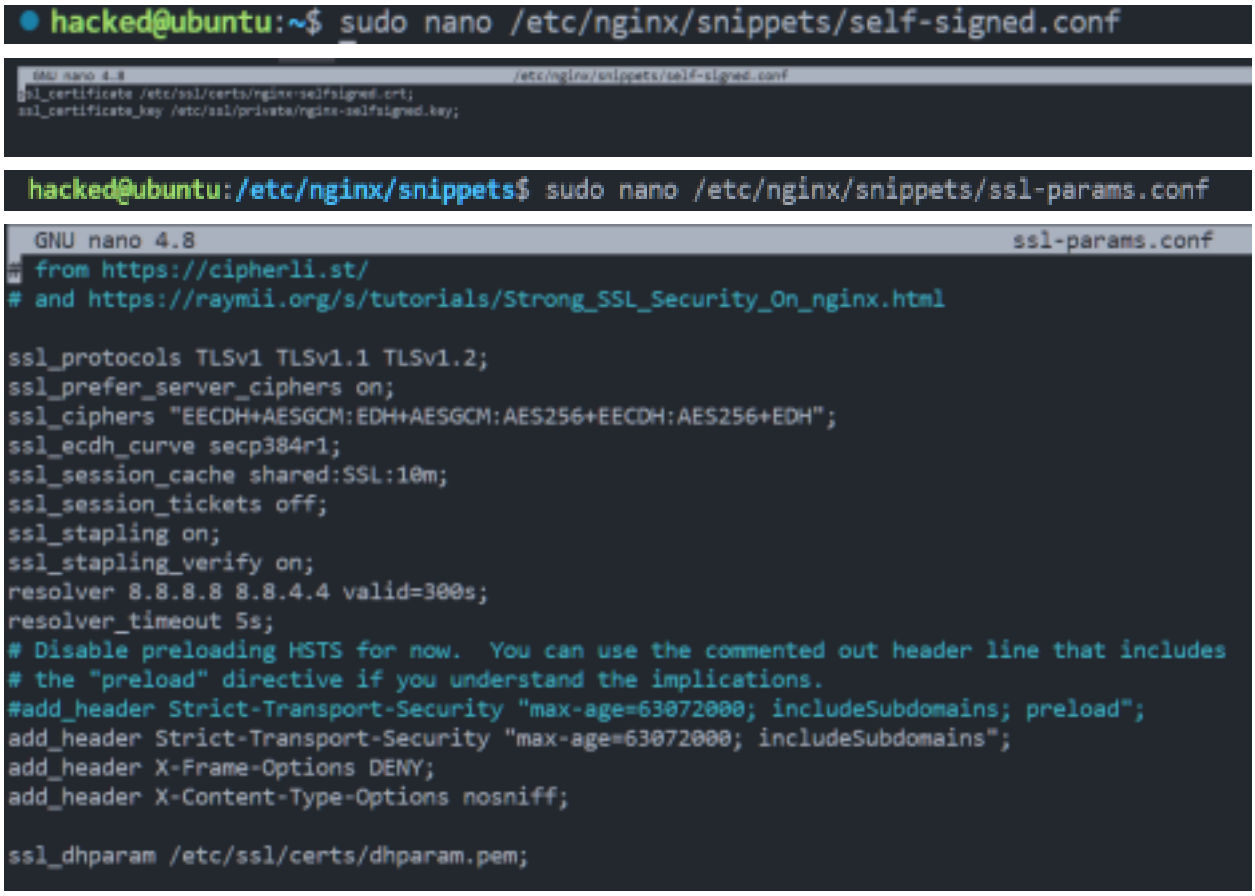
3. Afterfinishingtheconfiguration,wecheckwhethernginx isrunningproperlyornot. It wasconfiguredproperlyandwasactive.

```
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 14:20:17 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.




5. 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.




6. 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).

```
##

# 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 changedthe index.html filetodisplaysimpletextandourgroupnumber.
   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
```