# **Linux Server Configuration**

## **Server Details**

IP address : 54.169.103.230

SSH port : 2200

EC2 URL : [https://ec2-54-169-103-230.ap-southeast-1.compute.amazonaws.com/](http://ec2-18-196-167-64.us-west-2.compute.amazonaws.com/)

# Note: HTTPS Required for Facebook Login that’s why I am using URL as “https://” insted of “http://” for more details see here: [https://developers.facebook.com/blog/post/2018/06/08/enforce-https-facebook-login/](https://developers.facebook.com/blog/post/2018/06/08/enforce-https-facebook-login/" \o "https://developers.facebook.com/blog/post/2018/06/08/enforce-https-facebook-login/" \t "https://hub.udacity.com/rooms/community:nd004:617415-project-7/_blank)

IMP NOTE: That’s why I have to change the firewall setting (allow port 443) at aws instence and also at SSH server

## **Configuration steps**

### **Create an instance in AWS Lightsail**

* Go to AWS Lightsail and create a new account / sign in with your account.
* Click Create instance and choose Linux/Unix, OS only Ubuntu 16.04LTS
* Choose a payment plan (the cheapest plan is enough for now and it's free for first month)
* Click Create button to create an instance.

****Reference****

* ServerPilot, [How to Create a Server on Amazon Lightsail](https://serverpilot.io/community/articles/how-to-create-a-server-on-amazon-lightsail.html).

### **Set up SSH key**

* Go to account page from your AWS account. You will find your SSH key there.
* Download your SSH key, the file name will be like LightsailDefaultPrivateKey-\*.pem
* Navigate to the directory where your file is stored in your terminal.(/c/Users/Shikha/.ssh)
* Run chmod 600 LightsailDefaultPrivateKey-\*.pem to restrict the file permission.
* Change name to lightsail\_key1.rsa.
* Run a command  ssh -i ~/.ssh/lightsail\_key1.rsa ubuntu@54.169.103.230 in your terminal to connect to the instance via the terminal, where 54.169.103.230 is the public IP address of the instance.

## **Secure the server**

### **Update and upgrade installed packages**

sudo apt-get update

sudo apt-get upgrade

### **Change the SSH port from 22 to 2200**

* Edit the /etc/ssh/sshd\_config file: sudo nano /etc/ssh/sshd\_config.
* Change the port number from 22 to 2200.
* Save and exit using CTRL+X and confirm with Y.
* Restart SSH: sudo service ssh restart.

### **Configure the Uncomplicated Firewall (UFW)**

* Configure the default firewall for Ubuntu to only allow incoming connections for SSH (port 2200), HTTP (port 80), and NTP (port 123).

sudo ufw status # The UFW should be inactive.

sudo ufw default deny incoming # Deny any incoming traffic.

sudo ufw default allow outgoing # Enable outgoing traffic.

sudo ufw allow 2200/tcp # Allow incoming tcp packets on port 2200.

sudo ufw allow www # Allow HTTP traffic in.

sudo ufw allow 123/udp # Allow incoming udp packets on port 123.

sudo ufw deny 22 # Deny tcp and udp packets on port 22.

* Turn UFW on: sudo ufw enable. The output should be like this:

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

Firewall is active and enabled on system startup

* Check the status of UFW to list current roles: sudo ufw status.

The output should be like this:

Status: active

To Action From

-- ------ ----

2200/tcp ALLOW Anywhere

80/tcp ALLOW Anywhere

123/udp ALLOW Anywhere

22 DENY Anywhere

2200/tcp (v6) ALLOW Anywhere (v6)

80/tcp (v6) ALLOW Anywhere (v6)

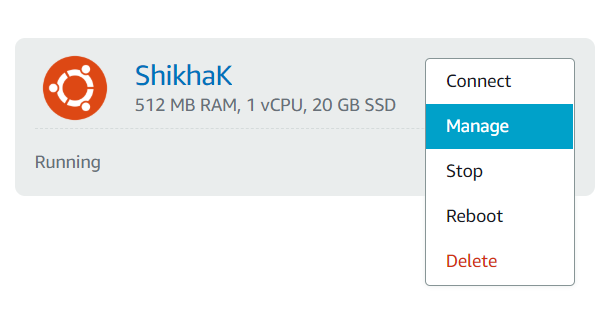
123/udp (v6) ALLOW Anywhere (v6)

22 (v6) DENY Anywhere (v6)

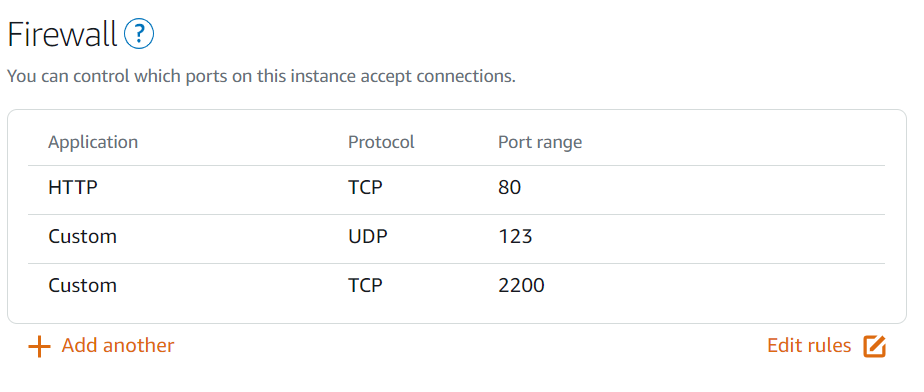
Exit the SSH connection: exit.

### **Update Firewall Setting at AWS instence**

* Click on the Manage option of the Amazon Lightsail Instance, then the Networking tab, and then change the firewall configuration to match the internal firewall settings above.



Allow ports 80(TCP), 123(UDP), and 2200(TCP), and deny the default port 22.



From your local terminal, run:

ssh -i ~/.ssh/lightsail\_key1.rsa -p 2200 [ubuntu@54.169.103.230](mailto:ubuntu@54.169.103.230)

(where 54.169.103.230 is the public IP address of the instance)

****References****

* Official Ubuntu Documentation, [UFW - Uncomplicated Firewall](https://help.ubuntu.com/community/UFW).
* TechRepublic, [How to install and use Uncomplicated Firewall in Ubuntu](https://www.techrepublic.com/article/how-to-install-and-use-uncomplicated-firewall-in-ubuntu/).

### **Use Fail2Ban to ban attackers**

* Fail2Ban is an intrusion prevention software framework that protects computer servers from brute-force attacks.
* Install Fail2Ban: sudo apt-get install fail2ban.
* Install sendmail for email notice: sudo apt-get install sendmail iptables-persistent.
* Create a copy of a file: sudo cp /etc/fail2ban/jail.conf /etc/fail2ban/jail.local.
* sudo nano /etc/fail2ban/jail.local and added my mail to the destmail
* Under [sshd] change port = ssh by port = 2200.
* Restart the service: sudo service fail2ban restart.

****References****

* [Fail2Ban Official website](http://www.fail2ban.org/wiki/index.php/Main_Page).

### **Automatically install updates**

* The unattended-upgrades package can be used to automatically install important system updates.
* Enable automatic (security) updates: sudo apt-get install unattended-upgrades
* Edit /etc/apt/apt.conf.d/50unattended-upgrades

sudo nano /etc/apt/apt.conf.d/50unattended-upgrades

* Uncomment the line ${distro\_id}:${distro\_codename}-updates and save it.
* Modify /etc/apt/apt.conf.d/20auto-upgrades file so that the upgrades are downloaded and installed every day:

APT::Periodic::Update-Package-Lists "1";

APT::Periodic::Download-Upgradeable-Packages "1";

APT::Periodic::AutocleanInterval "7";

APT::Periodic::Unattended-Upgrade "1";

* Enable it: sudo dpkg-reconfigure --priority=low unattended-upgrades.

sudo apt-get install apache2

* Restart Apache: sudo service apache2 restart.

****References****

* Official Ubuntu Documentation, [Automatic Updates](https://help.ubuntu.com/lts/serverguide/automatic-updates.html).
* Ubuntu Wiki, [AutomaticSecurityUpdates](https://help.ubuntu.com/community/AutomaticSecurityUpdates).

### **Updated packages to most recent versions**

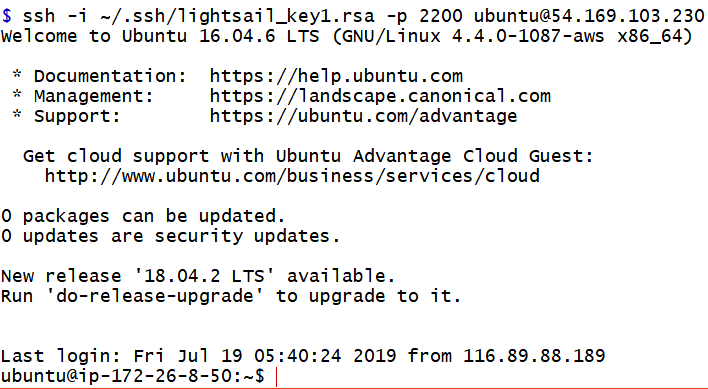
* Some packages have not been updated because the server need to be rebooted.

sudo apt-get update

sudo apt-get dist-upgrade

sudo shutdown -r now

* Logged back in, and I now see this message:



### **Give grader access**

* **Create a new user account named grader**
* While logged in as ubuntu, add user: sudo adduser grader.
* Enter a password (twice) and fill out information for this new user.
* **Give grader the permission to sudo**
* Edits the sudoers file:

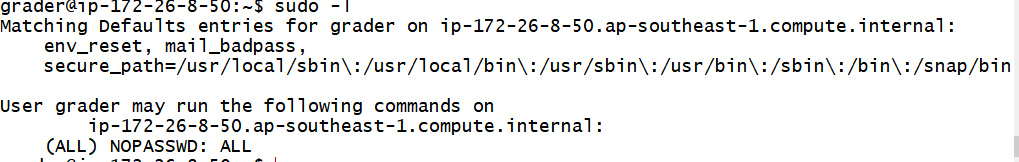
 sudo touch /etc/sudoers.d/grader

sudo nano /etc/sudoers.d/grader

Edit the file with following:

grader ALL=(ALL) NOPASSWD:ALL

* Save and exit using CTRL+X and confirm with Y.
* Verify that grader has sudo permissions. Run
* su - grader, enter the password,
* Run sudo -l and enter the password again. The output should be like this:



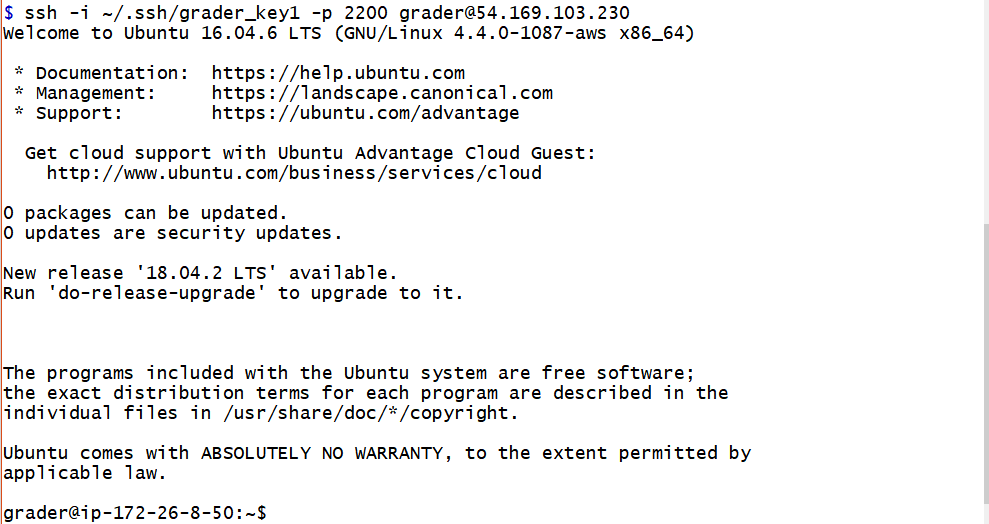
****Resources****

* DigitalOcean, [How To Add and Delete Users on an Ubuntu 14.04 VPS](https://www.digitalocean.com/community/tutorials/how-to-add-and-delete-users-on-an-ubuntu-14-04-vps)

### **Create an SSH key pair for grader using the ssh-keygen tool**

* **On the local machine:**
  + Run ssh-keygen
  + Enter file in which to save the key (I gave the name grader\_key1) in the local directory ~/.ssh
  + Enter in a passphrase twice. Two files will be generated ( ~/.ssh/grader\_key1 and ~/.ssh/grader\_key1.pub)
  + Run cat ~/.ssh/grader\_key.pub and copy the contents of the file
  + Log in to the grader's virtual machine
* **On the grader's virtual machine:**
  + Create a new directory called ~/.ssh (mkdir .ssh)
  + touch .ssh/authorized\_keys
  + Run sudo nano ~/.ssh/authorized\_keys and paste the content into this file, save and exit
  + Give the permissions: chmod 700 .ssh and chmod 644 .ssh/authorized\_keys
  + Check in /etc/ssh/sshd\_config file if PasswordAuthentication is set to no
  + Restart SSH: sudo service ssh restart
* **On the local machine, run:**

ssh -i ~/.ssh/grader\_key1 -p 2200 grader@54.169.103.230.



****References****

* DigitalOcean, [How To Set Up SSH Keys](https://www.digitalocean.com/community/tutorials/how-to-set-up-ssh-keys--2).
* Ubuntu Wiki, [SSH/OpenSSH/Keys](https://help.ubuntu.com/community/SSH/OpenSSH/Keys).

### **Disable root login**

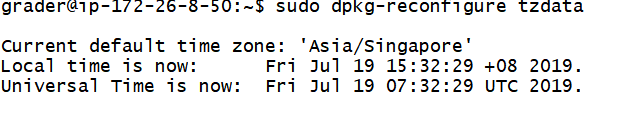
* $ sudo nano /etc/ssh/sshd\_config
* Find the PermitRootLogin line and edit to no
* $ sudo service ssh restart

### **Configure the local timezone to UTC**

* While logged in as grader, configure the time zone:

sudo dpkg-reconfigure tzdata.

* You should see something like that:



****References****

* Ubuntu Wiki, [UbuntuTime](https://help.ubuntu.com/community/UbuntuTime)
* Ask Ubuntu, [How do I change my timezone to UTC/GMT?](https://askubuntu.com/questions/138423/how-do-i-change-my-timezone-to-utc-gmt/138442)

### **Install and configure Apache to serve a Python mod\_wsgi application**

* Install Apache sudo apt-get install apache2
* Install mod\_wsgi sudo apt-get install python-setuptools libapache2-mod-wsgi
* Restart Apache sudo service apache2 restart
* Enable mod\_wsgi
* $ sudo a2enmod wsgi
* $ sudo service apache2 start
* Clone the Catalog app from Github
* Install git using: sudo apt-get install git
* cd /var/www
* sudo mkdir catalog
* Change owner of the newly created catalog folder sudo chown -R grader:grader catalog
* cd catalog
* Clone your project from github
* git clone <https://github.com/shikhakhanna19/Product-Catalog.git> catalog
* Create a catalog.wsgi file, then add this inside:

import sys

import logging

logging.basicConfig(stream=sys.stderr)

sys.path.insert(0, "/var/www/catalog/")

from catalog import app as application

application.secret\_key = 'supersecretkey'

* Rename application.py to ****init****.py :

mv application.py \_\_init\_\_.py

* Install virtual environment
* Install pip: sudo apt-get install python-pip
* Install the virtual environment sudo pip install virtualenv
* Create a new virtual environment with sudo virtualenv venv
* Activate the virutal environment source venv/bin/activate
* Change permissions sudo chmod -R 777 venv
* Install Flask and other dependencies

pip install httplib2

pip install requests

pip install --upgrade oauth2client

pip install sqlalchemy

pip install flask

Pip install sqlalchemy\_utils

pip install psycopg2

* Update path of client\_secrets.json file

nano \_\_init\_\_.py

* Change client\_secrets.json path to

/var/www/catalog/catalog/client\_secrets.json

* Configure and enable a new virtual host

Run this: sudo nano /etc/apache2/sites-available/catalog.conf

Paste the following code in this open file:

<VirtualHost \*:80>

ServerName 54.169.103.230

ServerAlias ec2-54-169-103-230.ap-southeast-1.compute.amazonaws.com

ServerAdmin admin@54.169.103.230

WSGIDaemonProcess catalog python-path=/var/www/catalog:/var/www/catalog/venv/lib/python2.7/site-packages

WSGIProcessGroup catalog

WSGIScriptAlias / /var/www/catalog/catalog.wsgi

<Directory /var/www/catalog/catalog/>

Order allow,deny

Allow from all

</Directory>

Alias /static /var/www/catalog/catalog/static

<Directory /var/www/catalog/catalog/static/>

Order allow,deny

Allow from all

</Directory>

ErrorLog ${APACHE\_LOG\_DIR}/error.log

LogLevel warn

CustomLog ${APACHE\_LOG\_DIR}/access.log combined

</VirtualHost>

* Enable the virtual host sudo a2ensite catalog
* **Install and configure PostgreSQL**
* sudo apt-get install libpq-dev python-dev
* sudo apt-get install postgresql postgresql-contrib
* sudo su - postgres
* psql
* CREATE USER catalog WITH PASSWORD 'password';
* ALTER USER catalog CREATEDB;
* CREATE DATABASE catalog WITH OWNER catalog;
* \c catalog
* REVOKE ALL ON SCHEMA public FROM public;
* GRANT ALL ON SCHEMA public TO catalog;
* \q
* Exit
* Change create engine line in \_\_init\_\_.py,listofcatalog.py and database\_setup.py to

 engine = create\_engine('postgresql://catalog:password@localhost/catalog')

Run python /var/www/catalog/catalog/database\_setup.py

* Put the client\_id value:

"502197887292-069vartk0ej9l0qga7mhvel1p6vale40.apps.googleusercontent.com”

In /var/www/catalog/catalog/templates/login.html file in following function

**function start() {**

**gapi.load('auth2', function() {**

**auth2 = gapi.auth2.init({**

**client\_id:'502197887292-069vartk0ej9l0qga7mhvel1p6vale40.apps.googleusercontent.com'**

**NOTE**: Change the following line in /var/www/catalog/catalog/templates/login.html

<link href='http://fonts.googleapis.com/css?family=Roboto:400,300,700' rel='stylesheet' type='text/css'>

To

<link href='https://fonts.googleapis.com/css?family=Roboto:400,300,700' rel='stylesheet' type='text/css'>

### **Steps To Create a Self-Signed SSL Certificate for Apache in Ubuntu 16.04**

## Prerequisites

* Before you begin, you should have a non-root user configured with sudo privileges.
* You will also need to have the Apache web server installed.
* When you have completed the prerequisites, continue below.

## Step 1 – Creating the SSL Certificate

We can create a self-signed key and certificate pair with OpenSSL in a single command:

sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/ssl/private/apache-selfsigned.key -out /etc/ssl/certs/apache-selfsigned.crt

Fill out the prompts appropriately. **The most important line is the one that requests the**Common Name (e.g. server FQDN or YOUR name)**. You need to enter the domain name associated with your server or, more likely, your server's public IP address.**

The entirety of the prompts will look something like this:

Output

Country Name (2 letter code) [AU]:US

State or Province Name (full name) [Some-State]:New York

Locality Name (eg, city) []:New York City

Organization Name (eg, company) [Internet Widgits Pty Ltd]:Bouncy Castles, Inc.

Organizational Unit Name (eg, section) []:Ministry of Water Slides

Common Name (e.g. server FQDN or YOUR name) []:server\_IP\_address

Email Address []:admin@your\_domain.com

Both of the files you created will be placed in the appropriate subdirectories under /etc/ssl.

## Step 2 – Configuring Apache to Use SSL

We have created our key and certificate files under the /etc/ssl directory. Now we just need to modify our Apache configuration to take advantage of these.

We will make a few adjustments to our configuration:

### Creating an Apache Configuration Snippet with Strong Encryption Settings

Create a new snippet in the /etc/apache2/conf-available directory. We will name the file ssl-params.conf to make its purpose clear:

sudo nano /etc/apache2/conf-available/ssl-params.conf

Paste the following configuration into the ssl-params.conf file we opened:

/etc/apache2/conf-available/ssl-params.conf

SSLCipherSuite EECDH+AESGCM:EDH+AESGCM:AES256+EECDH:AES256+EDH

SSLProtocol All -SSLv2 -SSLv3 -TLSv1 -TLSv1.1

SSLHonorCipherOrder On# Disable preloading HSTS for now. You can use the commented out header line that includes# the "preload" directive if you understand the implications.# Header always set Strict-Transport-Security "max-age=63072000; includeSubDomains; preload"

Header always set X-Frame-Options DENY

Header always set X-Content-Type-Options nosniff

# Requires Apache >= 2.4

SSLCompression off

SSLUseStapling on

SSLStaplingCache "shmcb:logs/stapling-cache(150000)"

# Requires Apache >= 2.4.11

SSLSessionTickets Off

Save and close the file when you are finished.

### Modifying the Default Apache SSL Virtual Host File

Next, let's modify /etc/apache2/sites-available/default-ssl.conf, the default Apache SSL Virtual Host file. If you are using a different server block file, substitute its name in the commands below.

Before we go any further, let's back up the original SSL Virtual Host file:

sudo cp /etc/apache2/sites-available/default-ssl.conf /etc/apache2/sites-available/default-ssl.conf.bak

Now, open the SSL Virtual Host file to make adjustments:

sudo nano /etc/apache2/sites-available/default-ssl.conf

Inside, with most of the comments removed, the Virtual Host file should look something like this by default:

/etc/apache2/sites-available/default-ssl.conf

<IfModule mod\_ssl.c>

<VirtualHost \_default\_:443>

ServerAdmin webmaster@localhost

DocumentRoot /var/www/html

ErrorLog ${APACHE\_LOG\_DIR}/error.log

CustomLog ${APACHE\_LOG\_DIR}/access.log combined

SSLEngine on

SSLCertificateFile /etc/ssl/certs/ssl-cert-snakeoil.pem

SSLCertificateKeyFile /etc/ssl/private/ssl-cert-snakeoil.key

<FilesMatch "\.(cgi|shtml|phtml|php)$">

SSLOptions +StdEnvVars

</FilesMatch>

<Directory /usr/lib/cgi-bin>

SSLOptions +StdEnvVars

</Directory>

</VirtualHost>

</IfModule>

We will be making some minor adjustments to the file. We will set the normal things we'd want to adjust in a Virtual Host file (ServerAdmin email address, ServerName, etc., and adjust the SSL directives to point to our certificate and key files.

After making these changes, your server block should look similar to this:

/etc/apache2/sites-available/default-ssl.conf

<IfModule mod\_ssl.c>

<VirtualHost \_default\_:443>

ServerAdmin your\_email@example.com

ServerName server\_domain\_or\_IP

ServerName 54.169.103.230

ServerAlias ec2-54-169-103-230.ap-southeast-1.compute.amazonaws.com

ServerAdmin admin@54.169.103.230

DocumentRoot /var/www/catalog

WSGIProcessGroup catalog

WSGIScriptAlias / /var/www/catalog/catalog.wsgi

<Directory /var/www/catalog/catalog/>

Order allow,deny

Allow from all

</Directory>

Alias /static /var/www/catalog/catalog/static

<Directory /var/www/catalog/catalog/static/>

Order allow,deny

Allow from all

</Directory>

ErrorLog ${APACHE\_LOG\_DIR}/error.log

CustomLog ${APACHE\_LOG\_DIR}/access.log combined

SSLEngine on

SSLCertificateFile /etc/ssl/certs/apache-selfsigned.crt

SSLCertificateKeyFile /etc/ssl/private/apache-selfsigned.key

<FilesMatch "\.(cgi|shtml|phtml|php)$">

SSLOptions +StdEnvVars

</FilesMatch>

<Directory /usr/lib/cgi-bin>

SSLOptions +StdEnvVars

</Directory>

</VirtualHost>

</IfModule>

Save and close the file when you are finished.

### (Recommended) Modifying the HTTP Host File to Redirect to HTTPS

As it stands now, the server will provide both unencrypted HTTP and encrypted HTTPS traffic. For better security, it is recommended in most cases to redirect HTTP to HTTPS automatically. If you do not want or need this functionality, you can safely skip this section.

To adjust the unencrypted Virtual Host file to redirect all traffic to be SSL encrypted, we can open the /etc/apache2/sites-available/000-default.conf file:

* sudo nano /etc/apache2/sites-available/000-default.conf

Inside, within the VirtualHost configuration blocks, we need to add a Redirect directive, pointing all traffic to the SSL version of the site:

/etc/apache2/sites-available/000-default.conf

<VirtualHost \*:80>

. . .

Redirect "/" "https://your\_domain\_or\_IP/"

. . .

</VirtualHost>

Save and close the file when you are finished.

## Adjusting the Firewall

If you have the ufw firewall enabled, as recommended by the prerequisite guides, you might need to adjust the settings to allow for SSL traffic. Luckily, Apache registers a few profiles with ufwupon installation.

We can see the available profiles by typing:

sudo ufw app list

You should see a list like this:

Output

Available applications:

Apache

Apache Full

Apache Secure

OpenSSH

You can see the current setting by typing:

sudo ufw status

To additionally let in HTTPS traffic, we can allow the "Apache Full" profile,443/tcp and then delete the redundant "Apache" profile allowance:

sudo ufw allow 'Apache Full'

sudo ufw delete allow 'Apache'

Your status should look like this now:

sudo ufw status

## Enabling the Changes in Apache

Now that we've made our changes and adjusted our firewall, we can enable the SSL and headers modules in Apache, enable our SSL-ready Virtual Host, and restart Apache.

We can enable mod\_ssl, the Apache SSL module, and mod\_headers, needed by some of the settings in our SSL snippet, with the a2enmod command:

sudo a2enmod ssl

sudo a2enmod headers

Next, we can enable our SSL Virtual Host with the a2ensite command:

sudo a2ensite default-ssl

We will also need to enable our ssl-params.conf file, to read in the values we set:

sudo a2enconf ssl-params

At this point, our site and the necessary modules are enabled. We should check to make sure that there are no syntax errors in our files. We can do this by typing:

sudo apache2ctl configtest

If everything is successful, you will get a result that looks like this:

Output

AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 127.0.1.1. Set the 'ServerName' directive globally to suppress this message

Syntax OK

The first line is just a message telling you that the ServerName directive is not set globally. If you want to get rid of that message, you can set ServerName to your server's domain name or IP address in /etc/apache2/apache2.conf. This is optional as the message will do no harm.

If your output has Syntax OK in it, your configuration file has no syntax errors. We can safely restart Apache to implement our changes:

sudo systemctl restart apache2

## Testing Encryption

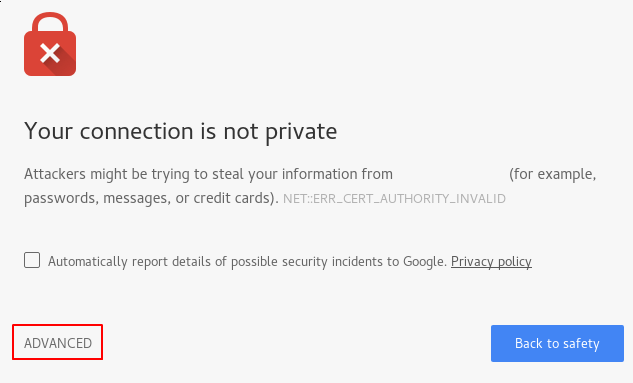
Now, we're ready to test our SSL server.

Open your web browser and type https:// followed by your server's domain name or IP into the address bar:

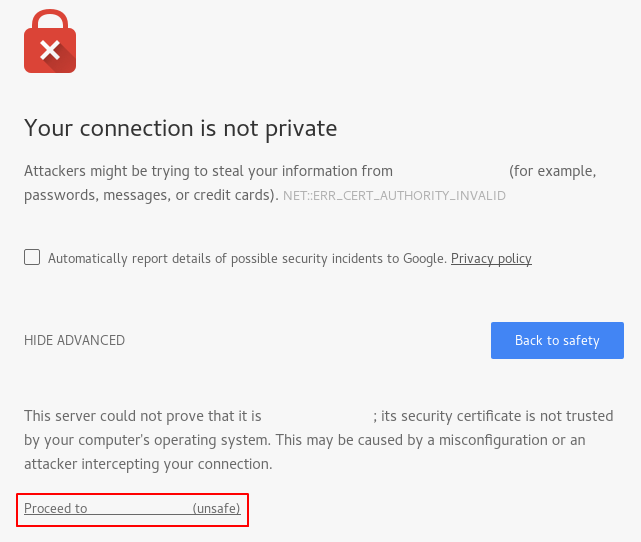
https://server\_domain\_or\_IP

**Like**: [https://ec2-54-169-103-230.ap-southeast-1.compute.amazonaws.com/](http://ec2-18-196-167-64.us-west-2.compute.amazonaws.com/)

Because the certificate we created isn't signed by one of your browser's trusted certificate authorities, you will likely see a scary looking warning like the one below:



This is expected and normal. We are only interested in the encryption aspect of our certificate, not the third party validation of our host's authenticity. Click "ADVANCED" and then the link provided to proceed to your host anyways:



You should be taken to your site. If you look in the browser address bar, you will see a lock with an "x" over it. In this case, this just means that the certificate cannot be validated. It is still encrypting your connection.

If you configured Apache to redirect HTTP to HTTPS, you can also check whether the redirect functions correctly:

http://server\_domain\_or\_IP

If this results in the same icon, this means that your redirect worked correctly.

## Step 6 – Changing to a Permanent Redirect

If your redirect worked correctly and you are sure you want to allow only encrypted traffic, you should modify the unencrypted Apache Virtual Host again to make the redirect permanent.

Open your server block configuration file again:

sudo nano /etc/apache2/sites-available/000-default.conf

Find the Redirect line we added earlier. Add permanent to that line, which changes the redirect from a 302 temporary redirect to a 301 permanent redirect:

/etc/apache2/sites-available/000-default.conf

<VirtualHost \*:80>

. . .

Redirect permanent "/" "https://your\_domain\_or\_IP/"

. . .

</VirtualHost>

Save and close the file.

Check your configuration for syntax errors:

sudo apache2ctl configtest

When you're ready, restart Apache to make the redirect permanent:

sudo systemctl restart apache2

**Reference:**

<https://www.digitalocean.com/community/tutorials/how-to-create-a-self-signed-ssl-certificate-for-apache-in-ubuntu-18-04>

### **Launch the Web Application**

* Restart Apache again: sudo service apache2 restart.
* Open your browser to

[https://ec2-54-169-103-230.ap-southeast-1.compute.amazonaws.com/](http://ec2-18-196-167-64.us-west-2.compute.amazonaws.com/)

Note: Test user for checking App with facebook

login : [shikha\_rnijomn\_testuser@tfbnw.net](mailto:shikha_rnijomn_testuser@tfbnw.net) password: awstestuser