Installing an SSL certificate on a website hosted on AWS

Introduction

This document provides a comprehensive overview of installing Secure Sockets Layer (SSL) certificate on Ubuntu webservers hosted on Amazon Web Services (AWS) Elastic Compute Cloud (EC2) instance, along with configuring AWS Route 53 for improved domain management and accessibility.

Prerequisites:

1. AWS Instance
    - learn how to create a AWS instance: https://github.com/samuelvaz/lamp_setup_on_aws/blob/main/EC2_INSTANCE_SETUP.md
2. Domain Name

Contents

Connecting to EC2 instance

Run below command to connect to EC2 instance

```
ssh -i <your_key>.pem <username>@<ip_address>
```

Replace:

- `<your_key.pem>` with the key .pem key pair provided by the AWS
- `<username>` withe user name of your instance
- `<ip_address>` with the public IP address of your instance

Installing Apache2

Install the apache2 web server using following commands:

```
sudo apt update
sudo apt install apache2
```

After installing the Apache2 web server, start the web server using:

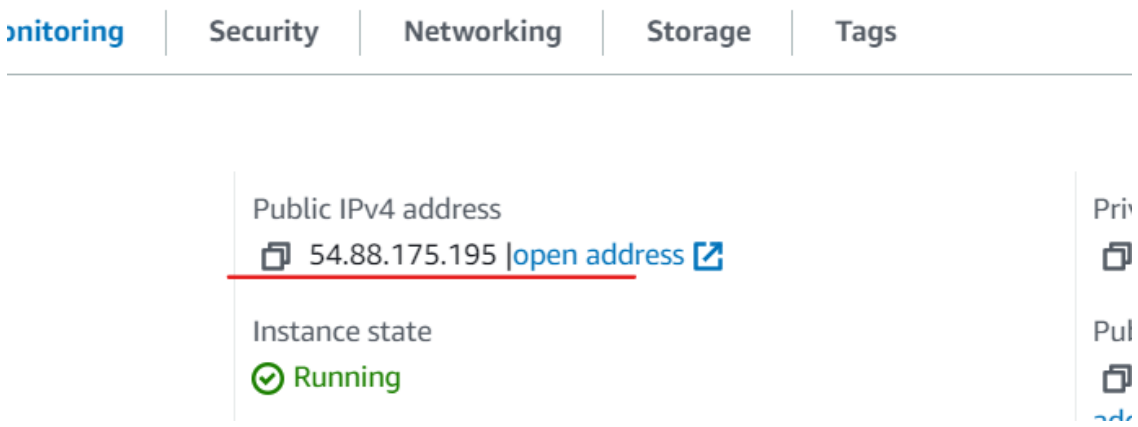
```
sudo systemctl start apache2


#check service status of apache2 web server
sudo systemctl status apache2
```

Enable the Apache2 service to start automatically after boot:

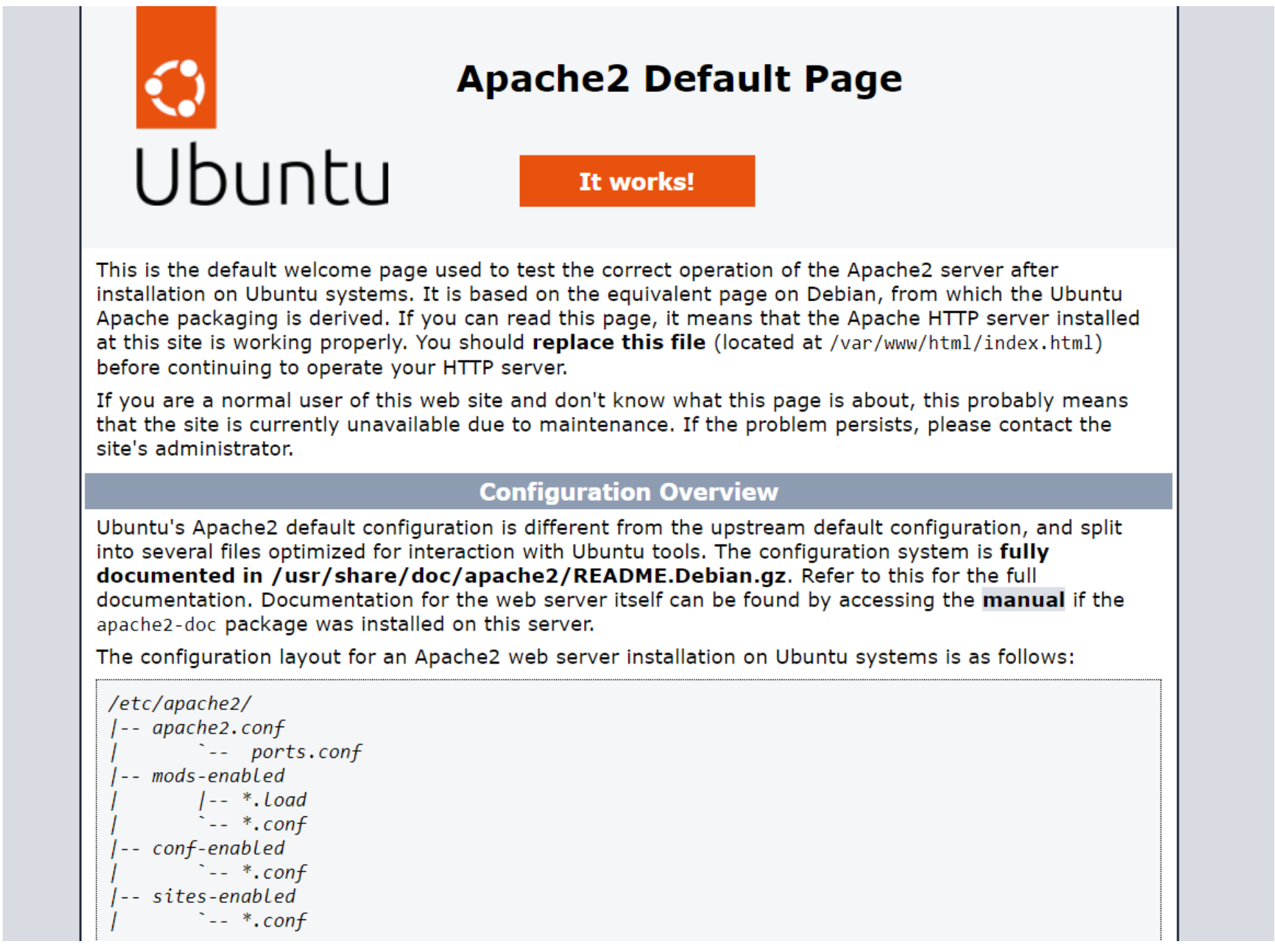
```
sudo systemctl enable apache2
```

Paste the public IP address of your instance in your browser:

#note: make sure to use http and not https.



This is how your browser should look like:



[Optional] Head over to '*/var/www/html*' and a paste your HTML code:

```
nano /var/www/html/index.html
```

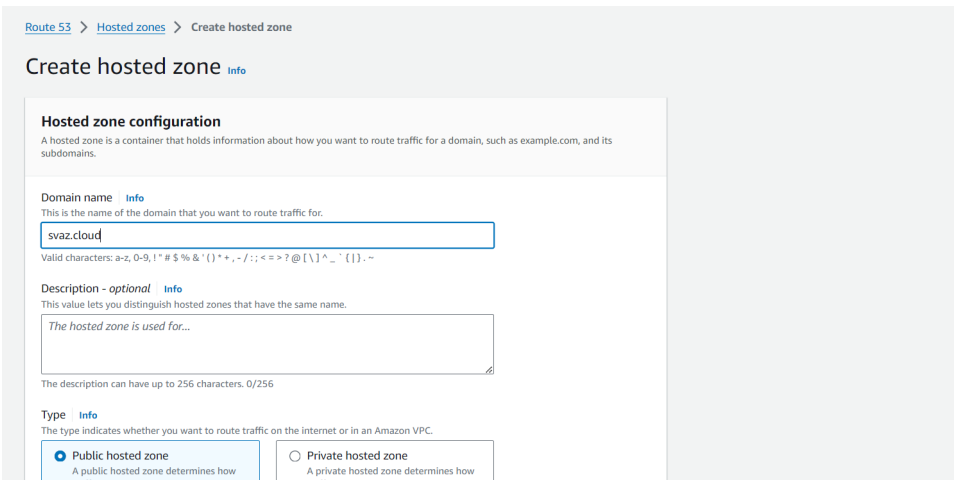
Creating a Hosted Zone using AWS Route 53

AWS Route 53: A scalable and highly available domain name system (DNS) web service provided by Amazon Web Services (AWS). It allows users to register and manage domain names (like example.com), and it also provides DNS services to route end-user requests to globally distributed resources, such as web servers or load balancers.

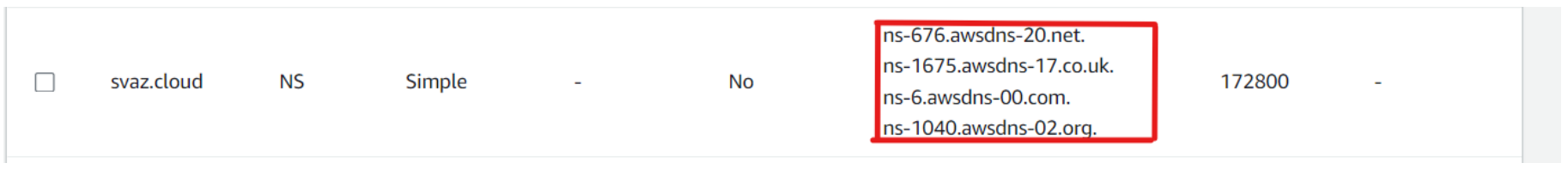
Hosted Zone: In AWS Route 53, a hosted zone is a container for holding DNS records that define how domain names map to IP addresses, enabling the management of domain settings and configurations. It serves as the authoritative record for a specific domain within the Route 53 DNS service.

Now that we have installed Apache2 and it is running. Its now time to set a domain name for your website. Follow steps below to:

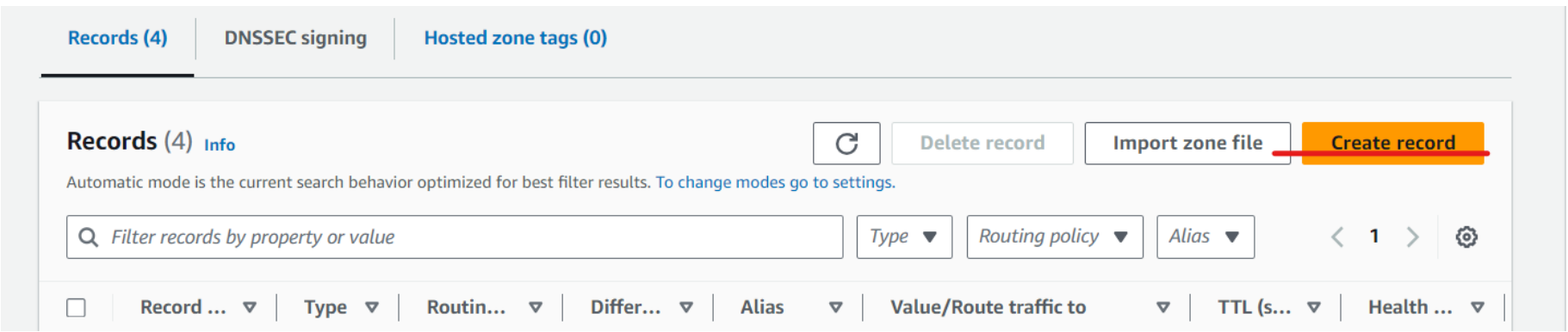
1. Open Route 53 in you AWS console
2. Click ‘Create Hosted Zone’
3. Enter your domain Name and click ‘Create hosted zone’



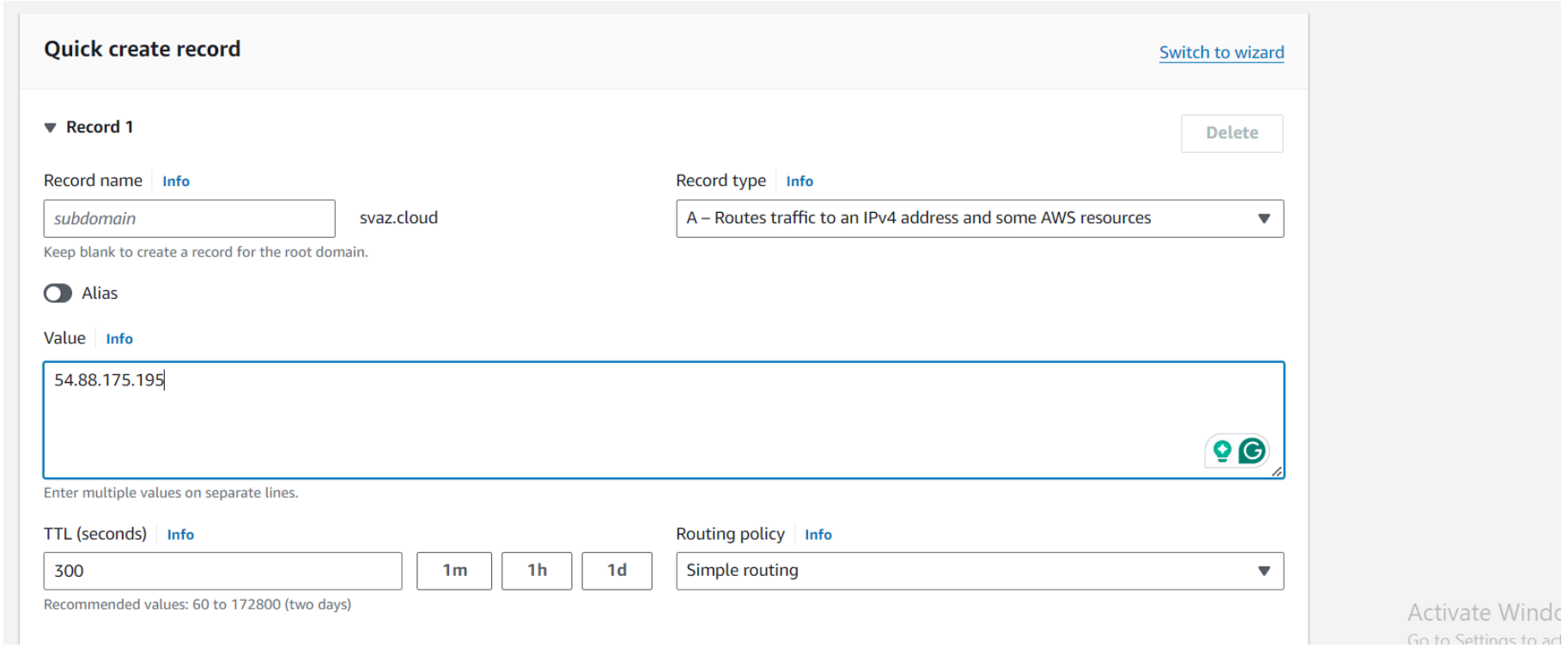
4. Head over to your Domain registrar’s website and paste the Name Servers provided by the AWS Route 53 in place of the Name Servers provided by your Domain Provider:



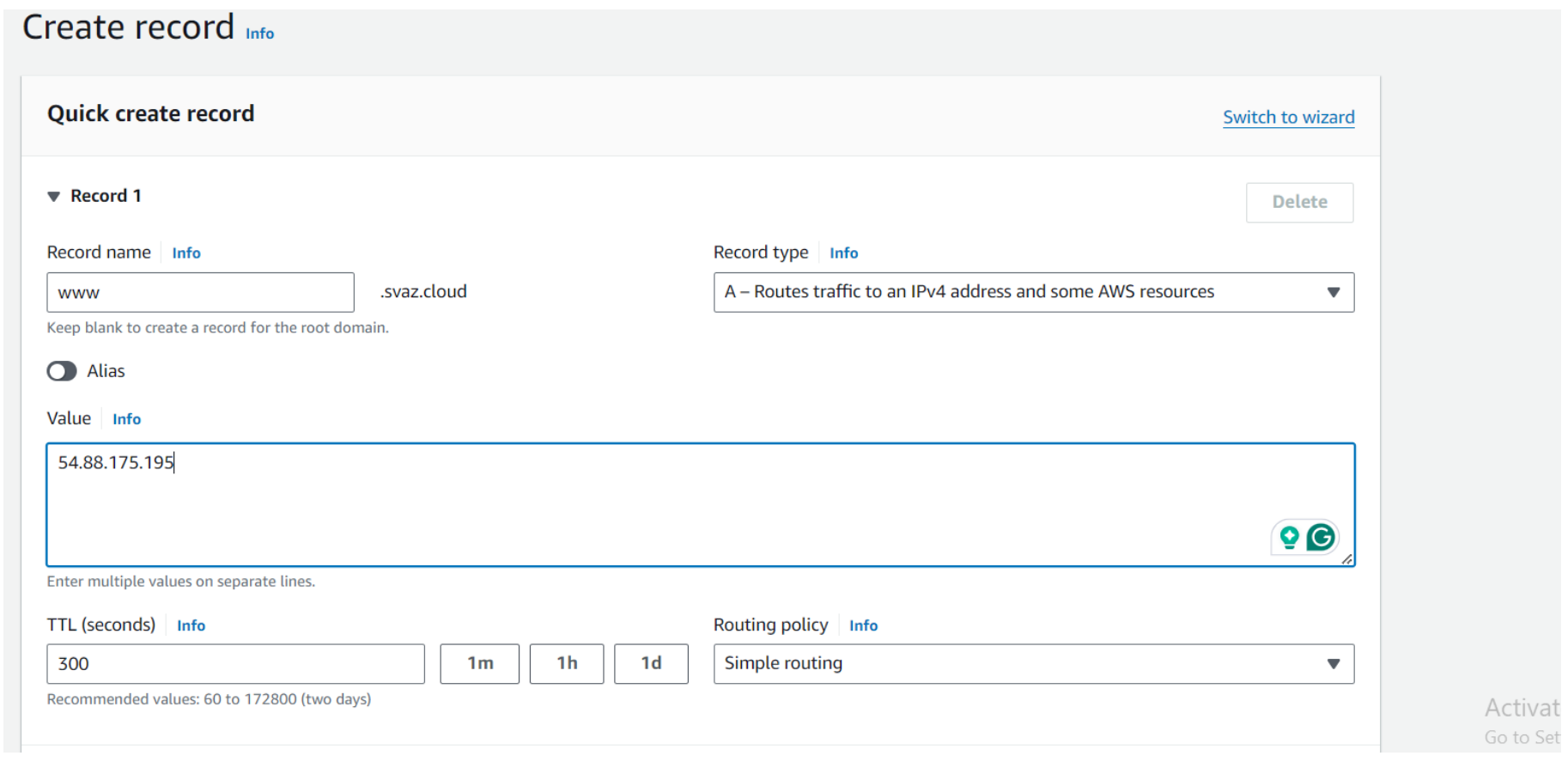
5. After creating hosted zone click on ‘Create record’



6. Create two records:
    a. Make sure to paste you instance’s public IP address in the “value” box



b.



After creating the records enter your domain name

Install SSL

In this step we'll be installing an SSL certificate using cerbot. Certbot is a fully-featured, extensible client for the Let's Encrypt CA (or any other CA that speaks the ACME protocol)

Run this command to install cerbot.

```
sudp apt-get install certbot
sudo apt-get install certbot python3-certbot-apache
```

Run this command to get a certificate and have Certbot edit your apache configuration automatically to serve it, turning on HTTPS access in a single step.

```
sudo certbot --apache
```

After running the command follow the instruction and enter information accordingly

Running Apache on Port 443

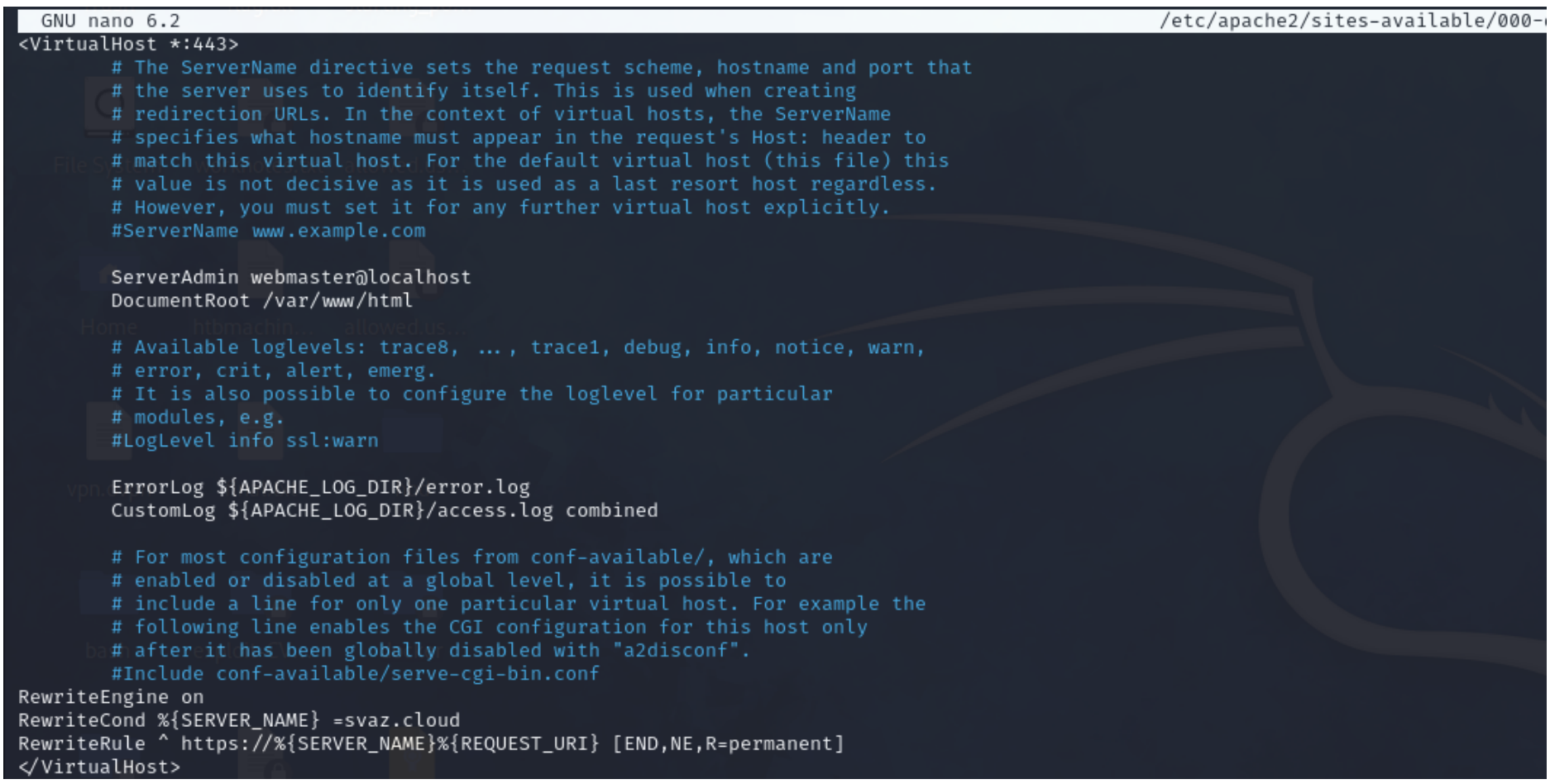
Now that SSL is install, set the webserver to run on port 443

Run this command to edit 000-default.conf file

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

and replace `'<VirtualHost *:80>'` with `'<VirtualHost *:443>'` and save the file

Refer to the screenshot below



Now set a redirect all the http(port 80) traffic to https(port 443)

Run the following command

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

Enter the below code and save the file

```
<VirtualHost *:80>
ServerName www.svaz.cloud
Redirect permanent / https://www.svaz.cloud/
</VirtualHost>
```

#note: replace ‘svaz.cloud’ with your domain name

Refer to the screenshot below

```
 GNU nano 6.2                                                    /etc/apache2/sites-available/000-
<VirtualHost *:443>
        # The ServerName directive sets the request scheme, hostname and port that
        # the server uses to identify itself. This is used when creating
        # redirection URLs. In the context of virtual hosts, the ServerName
        # specifies what hostname must appear in the request's Host: header to
        # match this virtual host. For the default virtual host (this file) this
        # value is not decisive as it is used as a last resort host regardless.
        # However, you must set it for any further virtual host explicitly.
        #ServerName www.example.com

        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/html

        # Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
        # error, crit, alert, emerg.
        # It is also possible to configure the loglevel for particular
        # modules, e.g.
        #LogLevel info ssl:warn

        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined

        # For most configuration files from conf-available/, which are
        # enabled or disabled at a global level, it is possible to
        # include a line for only one particular virtual host. For example the
        # following line enables the CGI configuration for this host only
        # after it has been globally disabled with "a2disconf".
        #Include conf-available/serve-cgi-bin.conf
RewriteEngine on
RewriteCond %{SERVER_NAME} =svaz.cloud
RewriteRule ^ https://%{SERVER_NAME}%{REQUEST_URI} [END,NE,R=permanent]
</VirtualHost>
<VirtualHost *:80>
ServerName www.svaz.cloud
Redirect permanent / https://www.svaz.cloud/
</VirtualHost>
# vim: syntax=apache ts=4 sw=4 sts=4 sr noet
```

Conclusion

By following this guide, you have successfully implemented an SSL certificate on your Ubuntu web server hosted on an AWS EC2 instance, utilizing the powerful combination of Let's Encrypt's `certbot`, Apache2 web server, and AWS Route 53 for domain management.

Remember to renew your SSL certificate regularly to maintain the security and trust earned with your enhanced website. Consider automating the renewal process with `certbot`'s built-in features for effortless upkeep.