Cover Page

Name:
Standard:
Year: 2024
Tutor:

How to Install Django with Apache on Ubuntu server?

Step 1: sudo apt update && sudo apt upgrade –y (This command updates the system to latest version)

Step 2: sudo apt install apache2 libapache2-mod-wsgi-py3 (These commands install the wsgi (web server gateyway interface). This wsgi helps to allow apache2 to work with python3)

Step 3: systemctl start apache2 (This command is instructing the system to start the Apache web server service, making it actively listen for incoming web requests and serving content.)

Step 4: systemctl enable apache2 (you are telling the system to configure Apache to start automatically during the system boot process.)

Step 5: systemctl status apache2

```
root@cc:~# systemctl status apache2
● apache2.service - The Apache HTTP Server
   Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor preset: enabled)
  Drop-In: /lib/systemd/system/apache2.service.d
           └─apache2-systemd.conf
   Active: active (running) since Wed 2023-12-13 21:55:02 EST; 18min ago
 Main PID: 4607 (apache2)
    Tasks: 56 (limit: 4655)
   CGroup: /system.slice/apache2.service
           ├─4607 /usr/sbin/apache2 -k start
           ├─4610 /usr/sbin/apache2 -k start
           └─4611 /usr/sbin/apache2 -k start

Dec 13 21:55:01 cc systemd[1]: Starting The Apache HTTP Server...
Dec 13 21:55:02 cc apachectl[4587]: AH00112: Warning: DocumentRoot [/home/cc/Gyeltshen/storefront] does not exist
Dec 13 21:55:02 cc apachectl[4587]: AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 127.0.1.1. Set the 'Server
Dec 13 21:55:02 cc systemd[1]: Started The Apache HTTP Server.
lines 1-16/16 (END)
```

Install MySQL and create a database

Step 1: sudo apt install mysql-server libmysqlclient-dev (**libmysqlclient-dev**: This is another package you are installing. It contains development files and libraries that are necessary for compiling and linking programs with the MySQL client library.)

Step 2: systemctl start mysql

Step 3: systemctl enable mysql

Step 4: systemctl status mysql

Step 5: mysql -u root
mysql> CREATE DATABASE django_db;
mysql> CREATE USER *'django_user'*@'localhost' IDENTIFIED BY *'Pa$$word'*;
mysql> GRANT ALL ON *django_db*.* TO *'django_user'*@'localhost';
mysql> FLUSH PRIVILEGES;
mysql> EXIT

Note: Italic words should replace by yourself.

Install Pip on Ubuntu 22.04

Step 1: sudo apt install python3-venv python3-pip (pip3 –version)

Install Django Using Virtualenv

Step 1: Create project directory using mkdir *foldername*. (You can create your project directory anywhere you want to create)

Step 2: Create your env inside your project directory using this command: python3 -m venv *envfoldername*.

Step 3: Activate your virtual environment by using this command: source *envfoldername*/bin/activate

Step 4: pip install django (django-admin –version)

Step 5: pip install MySQL client (Optional)

Creating your Django project

Note: Italic words should replace by yourself.

Step 1: django-admin startproject *django_app*. (Dot indicates the project is creating in same directory)

Step 2: nano django_app/settings.py (This opens the settings.py file)

Step 3: ALLOWED_HOSTS = ['your_server_ip', 'your-domain.com'] (Edit this line of code in settings.py file)

Step 4: DATABASES = {

'default': {
'ENGINE': 'django.db.backends.mysql',
'NAME': *'django_db'*,

'USER': *'django_user'*,
'PASSWORD': *'Pa$$word'*,
'HOST': '127.0.0.1',
'PORT' : '3306',
}
}

Step 5: import os

STATIC_URL='*/static/*'
STATIC_ROOT=os.path.join(BASE_DIR, *'static/*')

MEDIA_URL='*/media/*'
MEDIA_ROOT=os.path.join(BASE_DIR, *'media/*')

Note: Step 3-5 should edit in settings.py file.

Step 6: Save the file and exit.

Step 7: ./manage.py makemigrations(This commands make instance to migrate to db format)

Step 8: ./manage.py migrate(This commands migrate to db format)

Step 9: ./manage.py createsuperuser(This commands helps to create admin users)

Example:

Username (leave blank to use 'root'): **admin**

Email address: **admin@your-domain.com**
Password:
Password (again):
Superuser created successfully.

Step 10: ./manage.py collectstatic(For collecting all static files)

Note: Step 5-10 are optional.

Step 11: deactivate (This exits the env)

Configure Apache Web Server for Django

Note: Italic words should replace by yourself.
Step 1: sudo nano /etc/apache2/sites-available/*django.conf*

Step 2: Enter the code as shown below.

<VirtualHost *:80>

```
ServerAdmin admin@your-domain.com
ServerName your-domain.com
ServerAlias www.your-domain.com

DocumentRoot /var/www/django_project/ (Project directory)

ErrorLog ${APACHE_LOG_DIR}/your-domain.com_error.log
CustomLog ${APACHE_LOG_DIR}/your-domain.com_access.log combined

Alias /static /var/www/django_project/static
<Directory /var/www/django_project/static>
        Require all granted
</Directory>

Alias /media /var/www/django_project/media
<Directory /var/www/django_project/media>
        Require all granted
 </Directory>

<Directory /var/www/django_project/django_app>
        <Files wsgi.py>
                Require all granted
        </Files>
</Directory>

WSGIDaemonProcess django_app python-path=/var/www/django_project python-
home=/var/www/django_project/django_env

WSGIProcessGroup django_app

WSGIScriptAlias / /var/www/django_project/django_app/wsgi.py

</VirtualHost>
```

Step 3: sudo nano /etc/apache2/sites-available/*Django-ssl.conf*

Step 4: Add the code as shown below:

```
Alias /static /var/www/django_project/static
        <Directory /var/www/django_project/static>
                Require all granted
        </Directory>

        Alias /media /var/www/django_project/media
        <Directory /var/www/django_project/media>
                Require all granted
```

```
 </Directory>

<Directory /var/www/django_project/django_app>
    <Files wsgi.py>
        Require all granted
    </Files>
</Directory>

WSGIDaemonProcess django_app python-path=/var/www/django_project python-home=/var/www/django_project/django_env

WSGIProcessGroup django_app

WSGIScriptAlias / /var/www/django_project/django_app/wsgi.py
```

Step 5: sudo chown -R www-data:www-data */home/django_project/* OR sudo chown -R www-data:www-data /home/ubuntu/second_products/ OR sudo chmod -R 755 /home/ubuntu/second_products/ (This gives permission to execute the project by apache2 server)

Step 5: a2ensite *django.conf* (Make sure to enter into the location of the italic file)

Step 6: sudo a2enmod wsgi

Step 7: systemctl restart apache2

Step 8: Enter your server ip or domain whether your website is up or not

Checking logs

Cat –t /var/log/apache2/error.log

tail -n 50 /var/log/apache2/your-domain.com_error.log

tail –f /var/log/apache2/error.log

tail –f /var/log/mysql/error.log

tail -f /var/log/syslog

Some of the issue face and solution

Step a. If error occurs like below, the solution is given in step b.

```
(env) root@ip-172-31-22-178:/home/ubuntu# pip install mysqlclient
Collecting mysqlclient
  Using cached mysqlclient-2.2.4.tar.gz (90 kB)
  Installing build dependencies ... done
  Getting requirements to build wheel ... error
  error: subprocess-exited-with-error

  × Getting requirements to build wheel did not run successfully.
  │ exit code: 1
  ╰─> [27 lines of output]
      /bin/sh: 1: pkg-config: not found
      /bin/sh: 1: pkg-config: not found
      /bin/sh: 1: pkg-config: not found
      Trying pkg-config --exists mysqlclient
      Command 'pkg-config --exists mysqlclient' returned non-zero exit status 127.
      Trying pkg-config --exists mariadb
      Command 'pkg-config --exists mariadb' returned non-zero exit status 127.
      Trying pkg-config --exists libmariadb
      Command 'pkg-config --exists libmariadb' returned non-zero exit status 127.
      Traceback (most recent call last):
        File "/home/ubuntu/env/lib/python3.10/site-packages/pip/_vendor/pep517/in_process/_in_process.py", line 363, in <module>
          main()
        File "/home/ubuntu/env/lib/python3.10/site-packages/pip/_vendor/pep517/in_process/_in_process.py", line 345, in main
          json_out['return_val'] = hook(**hook_input['kwargs'])
        File "/home/ubuntu/env/lib/python3.10/site-packages/pip/_vendor/pep517/in_process/_in_process.py", line 130, in get_requires_for
uild_wheel
          return hook(config_settings)
        File "/tmp/pip-build-env-gdw3faq5/overlay/lib/python3.10/site-packages/setuptools/build_meta.py", line 325, in get_requires_for
```

Step b. sudo apt install mysql-server libmysqlclinet-dev

```
aws  Services  Search [Alt+S]  Sydney ▼  pelha ▼
ubuntu@ip-172-31-22-178:~/Amazon$ ls
manage.py  media  product  products  static  templates
ubuntu@ip-172-31-22-178:~/Amazon$ sudo apt install mysql-server libmysqlclient-dev
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  fontconfig-config fonts-dejavu-core libc-dev-bin libc-devtools libc6-dev libcgi-fast-perl libcgi-pm-perl libclone-perl libcrypt-dev
  libdeflate0 libencode-locale-perl libevent-pthreads-2.1-7 libfcgi-bin libfcgi-perl libfcgi0ldbl libfontconfig1 libgd3
  libhtml-parser-perl libhtml-tagset-perl libhtml-template-perl libhttp-date-perl libhttp-message-perl libio-html-perl libjbig0
  libjpeg-turbo8 libjpeg8 liblwp-mediatypes-perl libmecab2 libmysqlclient21 libnsl-dev libprotobuf-lite23 libssl-dev libtiff5
  libtimedate-perl libtirpc-dev liburi-perl libwebp7 libxpm4 libzstd-dev linux-libc-dev manpages-dev mecab-ipadic mecab-ipadic-utf8
  mecab-utils mysql-client-8.0 mysql-client-core-8.0 mysql-common mysql-server-8.0 mysql-server-core-8.0 rpcsvc-proto zlib1g-dev
Suggested packages:
  glibc-doc libgd-tools libdata-dump-perl libipc-sharedcache-perl libssl-doc libbusiness-isbn-perl libwww-perl mailx tinyca
The following NEW packages will be installed:
  fontconfig-config fonts-dejavu-core libc-dev-bin libc-devtools libc6-dev libcgi-fast-perl libcgi-pm-perl libclone-perl libcrypt-dev
  libdeflate0 libencode-locale-perl libevent-pthreads-2.1-7 libfcgi-bin libfcgi-perl libfcgi0ldbl libfontconfig1 libgd3
  libhtml-parser-perl libhtml-tagset-perl libhtml-template-perl libhttp-date-perl libhttp-message-perl libio-html-perl libjbig0
  libjpeg-turbo8 libjpeg8 liblwp-mediatypes-perl libmecab2 libmysqlclient-dev libmysqlclient21 libnsl-dev libprotobuf-lite23
  libssl-dev libtiff5 libtimedate-perl libtirpc-dev liburi-perl libwebp7 libxpm4 libzstd-dev linux-libc-dev manpages-dev mecab-ipadic
  mecab-ipadic-utf8 mecab-utils mysql-client-8.0 mysql-client-core-8.0 mysql-common mysql-server mysql-server-8.0
  mysql-server-core-8.0 rpcsvc-proto zlib1g-dev
0 upgraded, 53 newly installed, 0 to remove and 10 not upgraded.
Need to get 43.6 MB of archives.
After this operation, 306 MB of additional disk space will be used.
Do you want to continue? [Y/n]
```

Error below

```
configure: error: The pkg-config script could not be found or is too old.  Make sure
is in your PATH or set the PKG_CONFIG environment variable to the full
path to pkg-config.

Alternatively, you may set the environment variables XMEDCON_GLIB_CFLAGS
and XMEDCON_GLIB_LIBS to avoid the need to call pkg-config.
See the pkg-config man page for more details.
```

Solution: apt-get install -y pkg-config

Best practice for project structure

Directory Hierarchy

```
project_name/
├── manage.py
├── project_name/
│   ├── __init__.py
│   ├── asgi.py
│   ├ settings.py
│   ├ urls.py
│   ├── wsgi.py
├── app1/
├── app2/
  ...
├── static/
├── media/
├── templates/
```

Project_name: The root directory of your project.

Project_name/project_name: This inner directory holds core project settings and configuration.

app1, app2: These are the individual apps you create within the project.

static: Houses static files like CSS, JavaScript, and images.

media: Stores user-uploaded files.

templates: Contains HTML templates.

Naming Conventions

Consistency in naming conventions enhances code readability. Follow these conventions:

- Apps: Use lowercase names, with underscores instead of spaces. Example: my_app.
- Modules: Use lowercase names with underscores for module files. Example: my_module.py.
- Classes: Use CamelCase for class names. Example: MyClass.
- Functions and Variables: Use lowercase with underscores for function and variable names. Example: my_function

Modular Code Design

Organize your codebase into modular components, making each component focused on a specific task. This makes code easier to understand, test, and maintain.

- Apps: Divide your project into multiple apps based on functionality. Each app should be self-contained, handling a specific feature.
- Views: Keep your views concise and focused on handling HTTP requests. Use class-based views for better organization.
- Models: Organize models in a way that reflects your project's data structure. Utilize model inheritance and related fields.
- Templates: Use template inheritance to avoid code duplication. Create reusable templates and keep them organized.

- Utils: For utility functions or classes that are used across the project, create a utils module within your app.

Settings

Keep your project settings in the settings.py file within the inner project directory. Use environment variables for sensitive information.

Database backup using Script and Cron job

1) Create folder in directory /mnt
2) To go to above directory cd /mnt
3) The create folder called backup by using this command mkdir backup

```
root@ip-172-31-22-178:/mnt# ls
root@ip-172-31-22-178:/mnt# mkdir backup
root@ip-172-31-22-178:/mnt# ls
backup
root@ip-172-31-22-178:/mnt#
```

4) Inside backup folder creates one script file called database
5) Command to create backup file -> touch database.sh
6) Enter this line codes using by opening above database.sh file by command nano database.sh

7)
```
1#!/bin/bash

# Set the date to include in the backup file name
date=$(date +"%d%m%Y_%H%M%S")

# Set the backup directory
backup_dir="/mnt/backup"

# Set MySQL/MariaDB credentials
db_user="Bhutan"
db_password="Bhutan@23"

# Set the list of databases to backup
databases="products"

# Loop through each database and perform the backup
for db in $databases;
    do
        mysqldump -u$db_user -p$db_password $db > $backup_dir/products_${date}.sql
    done
-- INSERT --                                                                    20,9          All
```

8) Set the permission of the database.sh file by this command sudo chmod +x database.sh

9)
```
root@ip-172-31-22-178:/mnt# ls
backup  database.sh
root@ip-172-31-22-178:/mnt# sudo chmod +x database.sh
root@ip-172-31-22-178:/mnt# ls -al
total 16
drwxr-xr-x  3 root root 4096 May 22 15:32 .
drwxr-xr-x 19 root root 4096 Apr 16 08:07 ..
drwxr-xr-x  2 root root 4096 May 22 15:23 backup
-rwxr-xr-x  1 root root  445 May 22 15:32 database.sh
root@ip-172-31-22-178:/mnt# mv database.sh /mnt/backup/
root@ip-172-31-22-178:/mnt#
```

10) Enter this command **crontab -e** in terminal, this command will open the editor page of crontab and one line of code in the editor at end as shown in figure below. In the below script the auto backup of db is set at every one minute.

```
  GNU nano 6.2                          /tmp/crontab.Qj9ShB/crontab
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command

0 * * * * /mnt/backup/database.sh
```

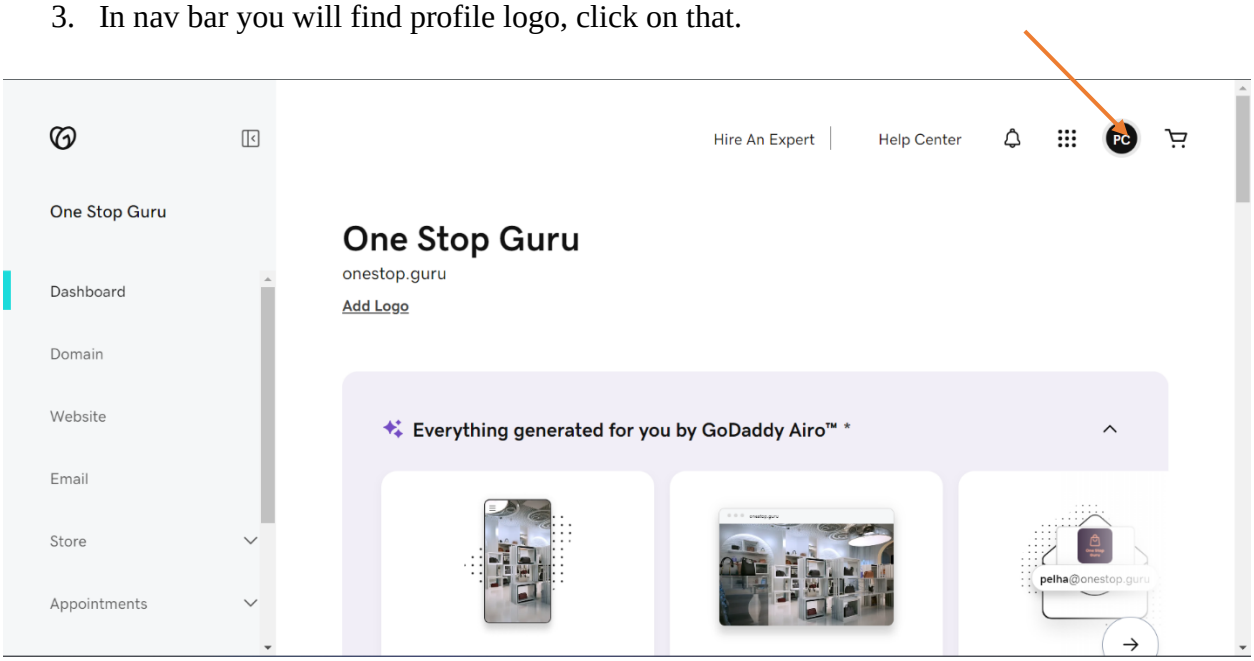
11) This are the files auto backup using crontab and script file.

```
root@ip-172-31-22-178:/mnt/backup# ls
database.sh  products_22052024_160001.sql
```

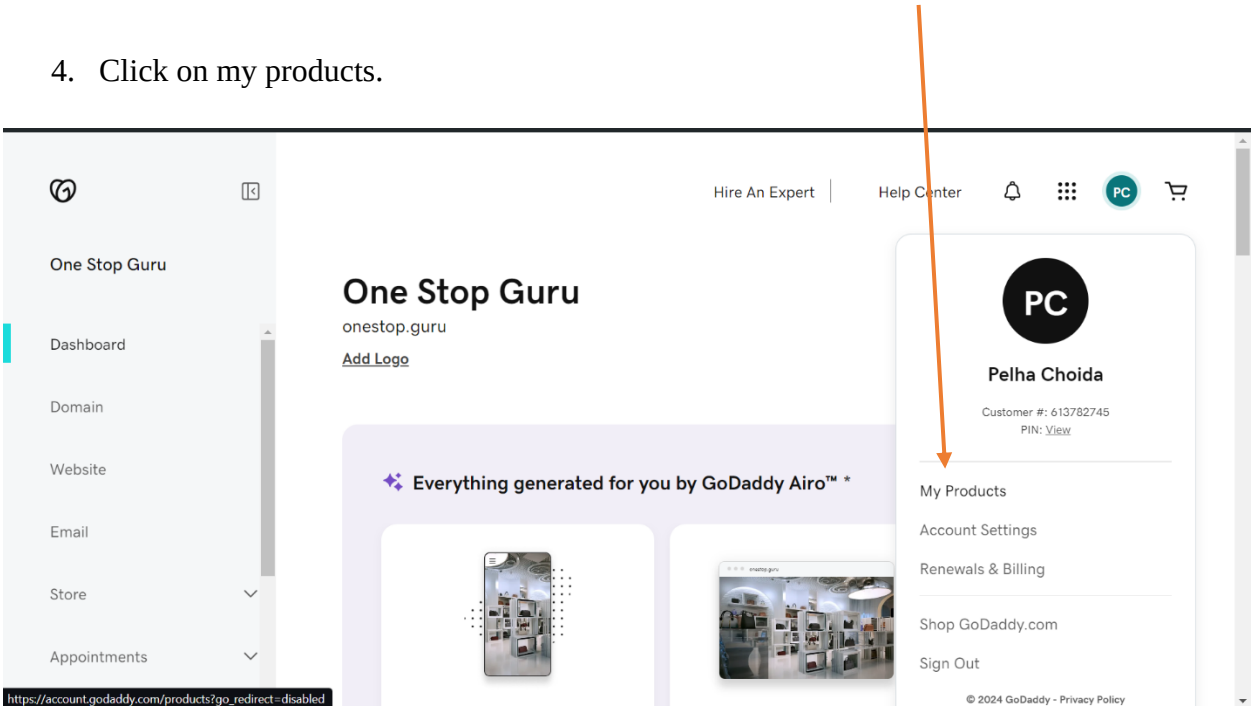
Setting up the Go Daddy Domain Name with server

1. Vist -> https://www.godaddy.com/en-uk
2. Sign with credentials

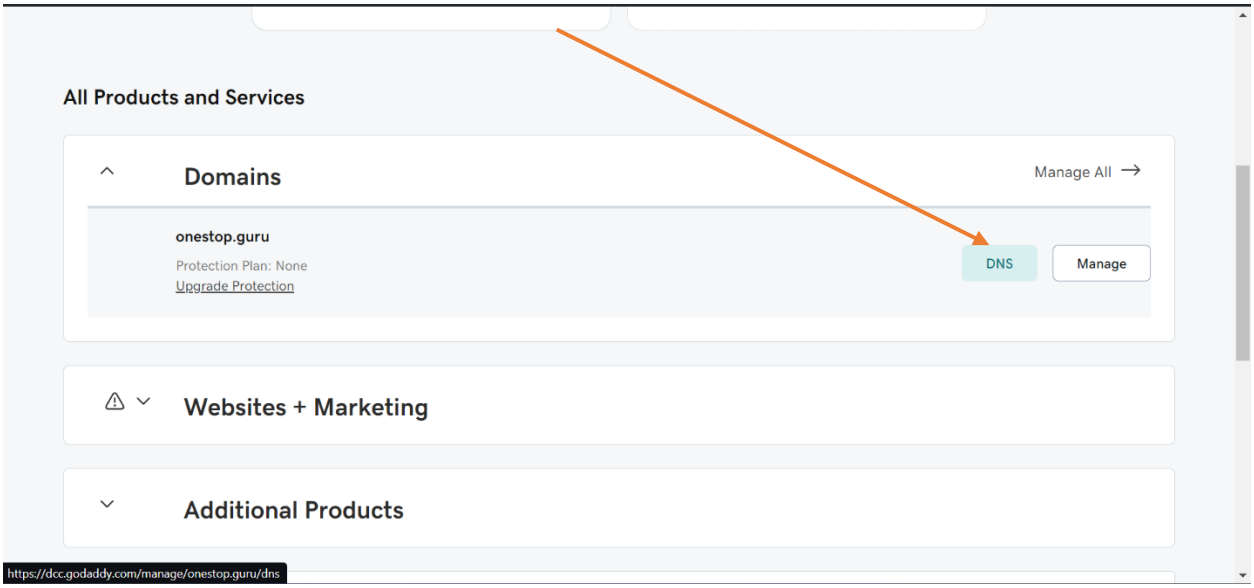
3. In nav bar you will find profile logo, click on that.



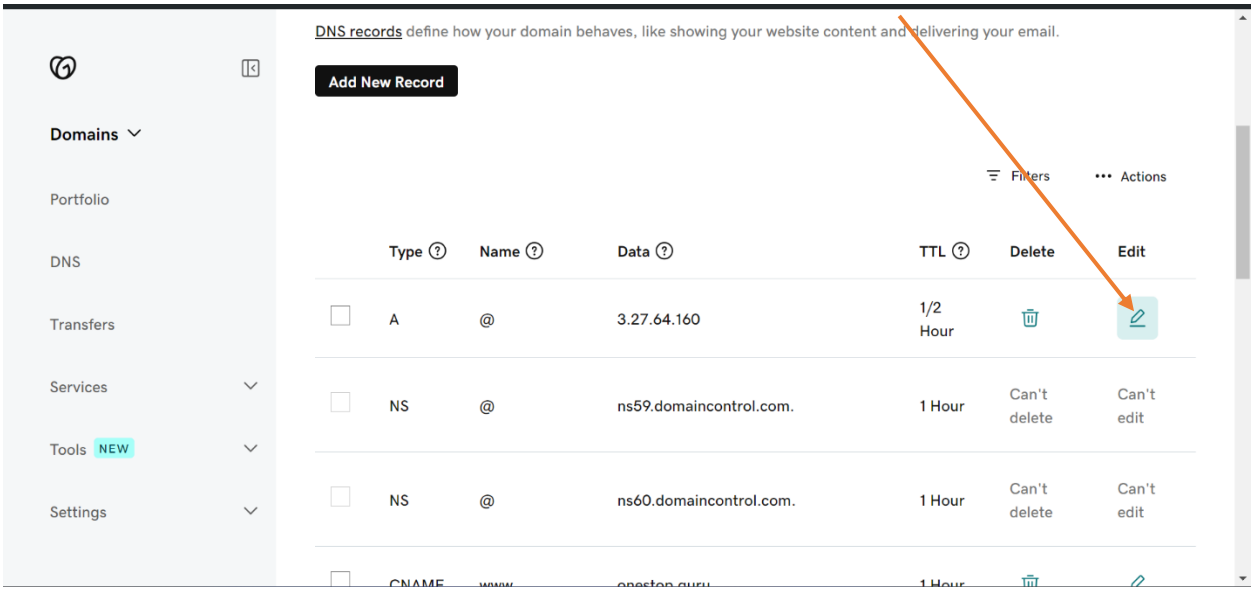
4. Click on my products.



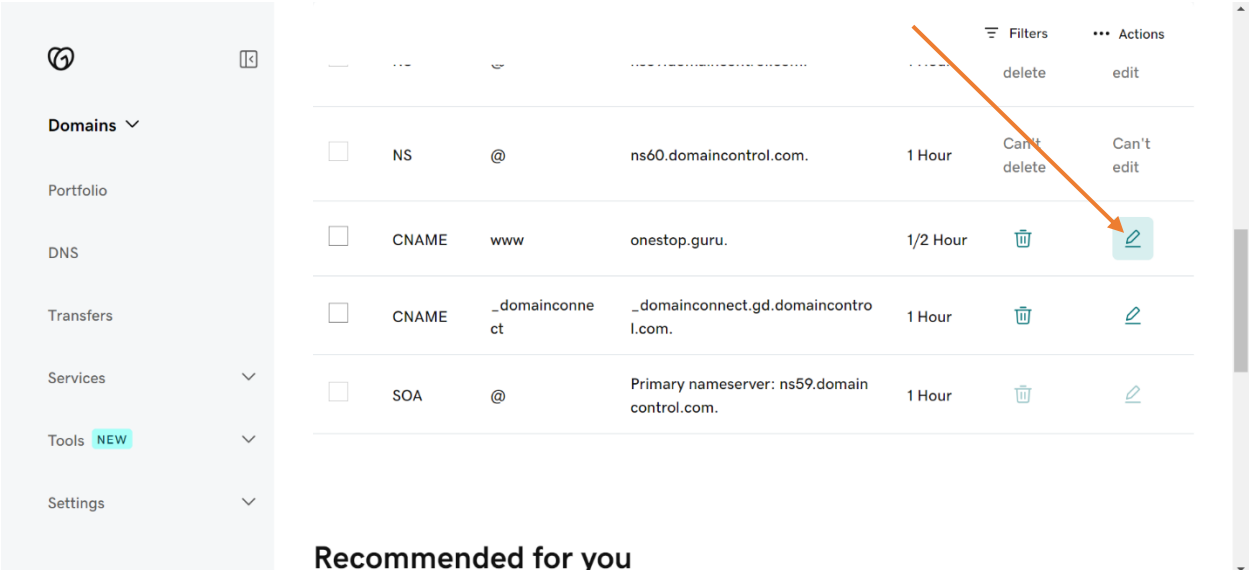
5. Click on DNS.



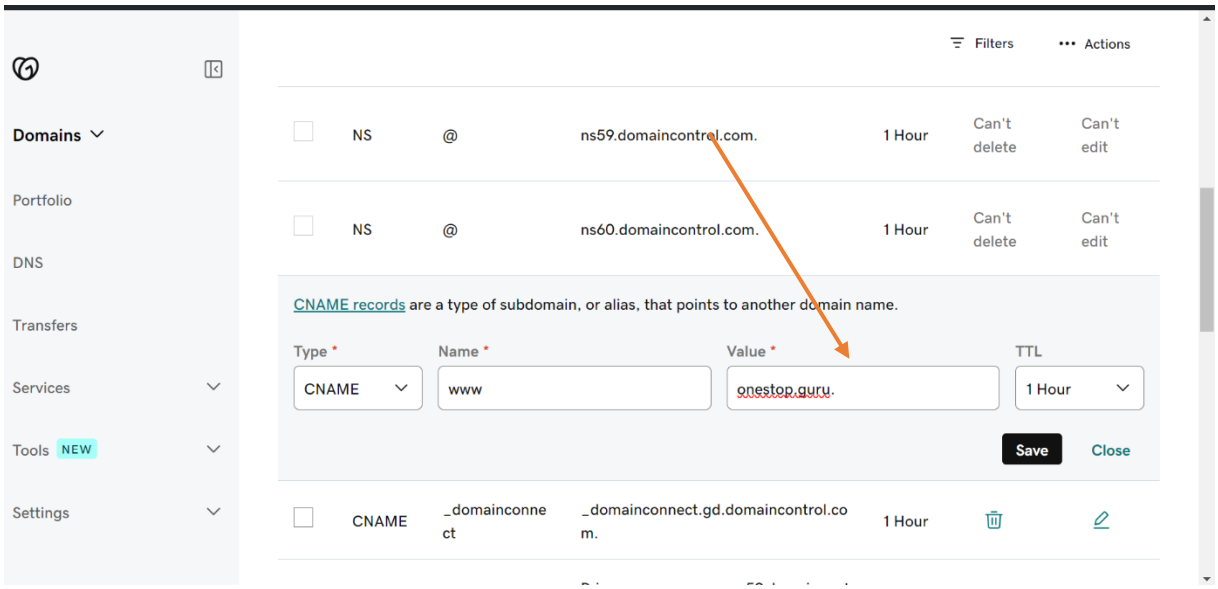
6. Click on edit button of type A. Enter the IP of server in value filed. Then click save.



7. Click on edit button of type CNAME and name www.



8. Enter the value as your domain name.



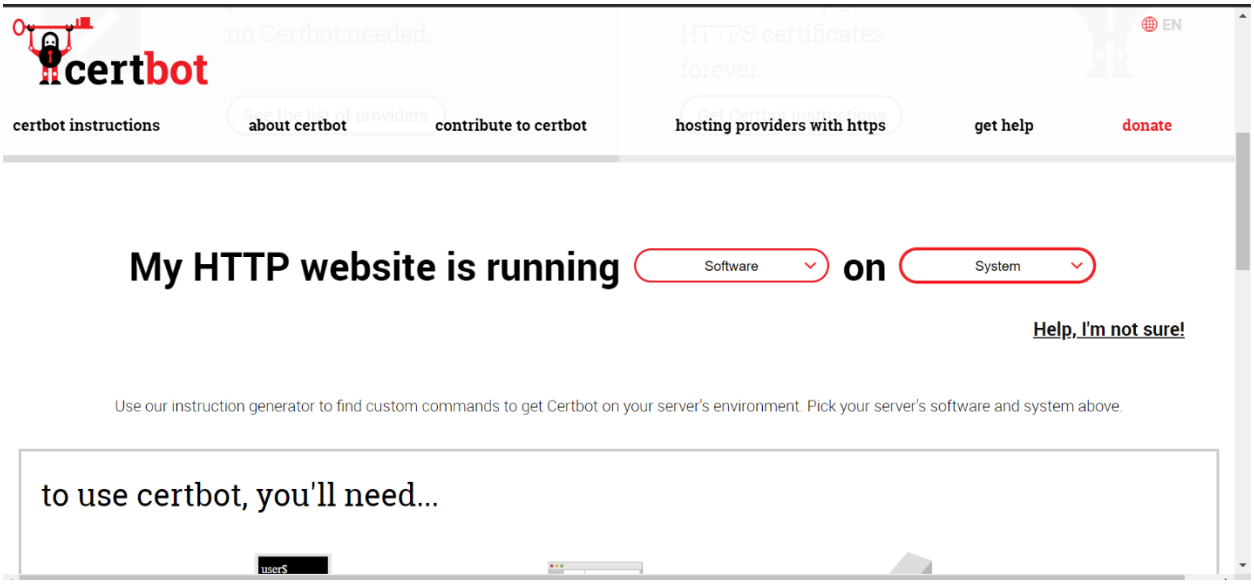
After Domain Name setup, enter the domain name in setting file of your Django project in the allowed host array.

```
BASE_DIR = Path(__file__).resolve().parent.parent


# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/5.0/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'django-insecure-amik$#2l@!&9rlrc(olzw%(v9u-0f%$r)va8-3$0$je$d#e_)o'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = ['3.27.64.160', '127.0.0.1', 'localhost', 'onestop.guru']


# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'product'
]

MIDDLEWARE = [
```

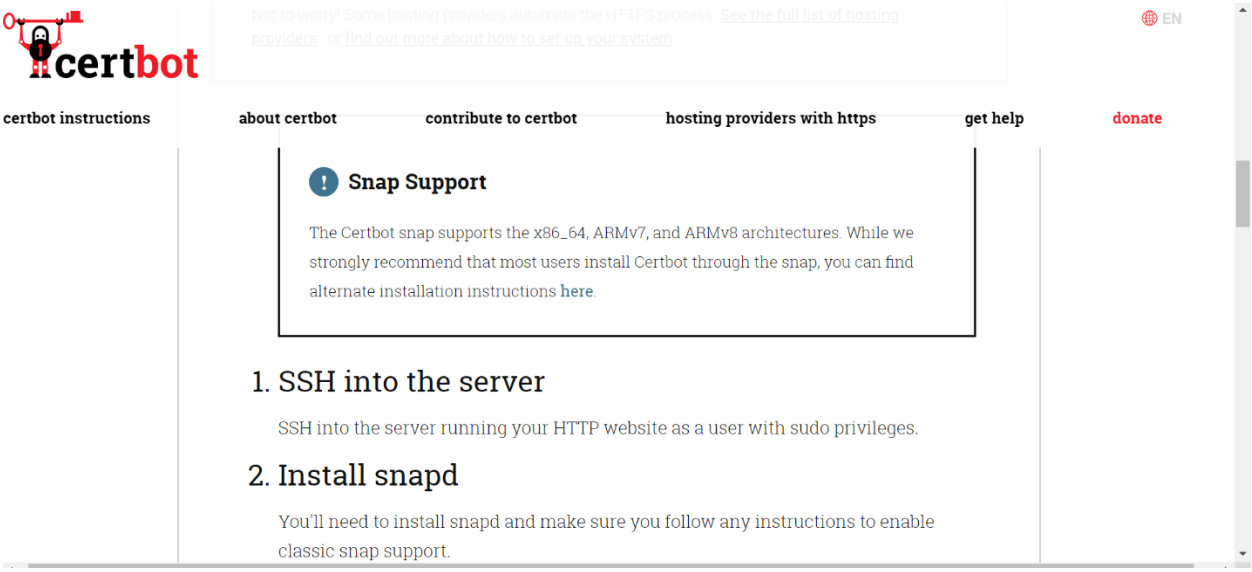
Enable port 443 using Cerbot

1) Visit -> https://certbot.eff.org/

2) Choose web server and your system.



3) After choosing and scroll you find this type of view as shown figure below.



4) Run sudo apt-get remove certbot, sudo dnf remove certbot. This command removes if cerbot is installed before.

```
ubuntu@ip-172-31-22-178:/$ sudo apt-get remove certbot
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Package 'certbot' is not installed, so not removed
0 upgraded, 0 newly installed, 0 to remove and 2 not upgraded.
ubuntu@ip-172-31-22-178:/$ sudo dnf remove certbot
```

5) Run sudo snap install --classic certbot. Enter information if it is pop up and enter y to confirm and hit enter.

```
ubuntu@ip-172-31-22-178:/$ sudo snap install --classic certbot
error: cannot add authorization: open /home/ubuntu/.snap/auth.json: permission denied
ubuntu@ip-172-31-22-178:/$ sudo -s
root@ip-172-31-22-178:/# sudo snap install --classic certbot
certbot 2.10.0 from Certbot Project (certbot-eff✓) installed
root@ip-172-31-22-178:/# sudo ln -s /snap/bin/certbot /usr/bin/certbot
root@ip-172-31-22-178:/# sudo certbot --apache
Saving debug log to /var/log/letsencrypt/letsencrypt.log
Enter email address (used for urgent renewal and security notices)
 (Enter 'c' to cancel): pelhacloud@gmail.com

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Please read the Terms of Service at
https://letsencrypt.org/documents/LE-SA-v1.4-April-3-2024.pdf. You must agree in
order to register with the ACME server. Do you agree?
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
(Y)es/(N)o:
```

```
root@ip-172-31-22-178:/# sudo snap install --classic certbot
certbot 2.10.0 from Certbot Project (certbot-eff✓) installed
root@ip-172-31-22-178:/#
```

6) Select domain name hit enter. After that the https will finally configured.

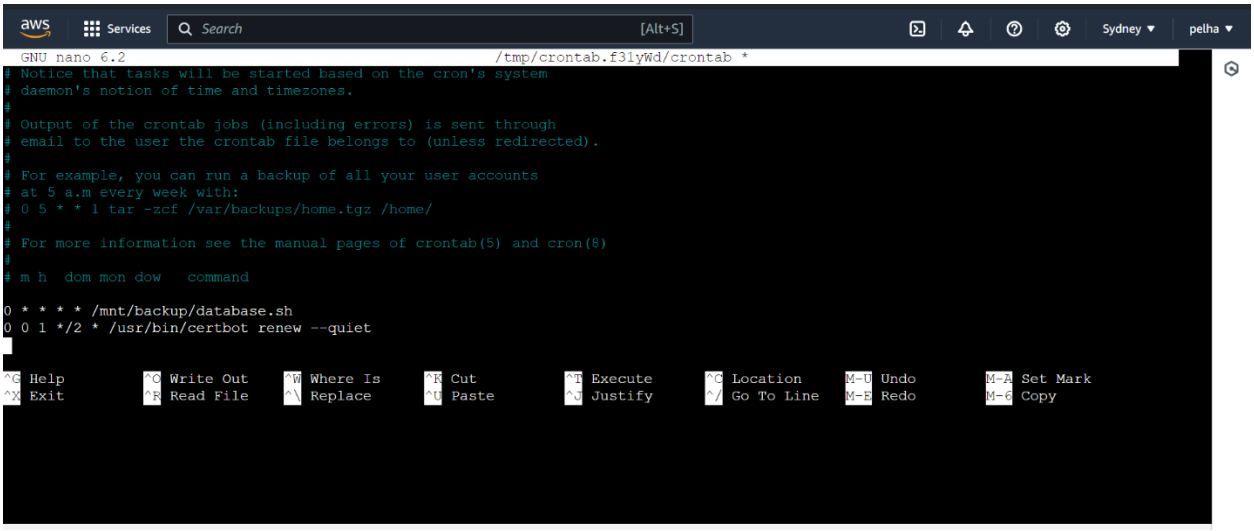
```
Saving debug log to /var/log/letsencrypt/letsencrypt.log
Enter email address (used for urgent renewal and security notices)
 (Enter 'c' to cancel): pelhacloud@gmail.com

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Please read the Terms of Service at
https://letsencrypt.org/documents/LE-SA-v1.4-April-3-2024.pdf. You must agree in
order to register with the ACME server. Do you agree?
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
(Y)es/(N)o: y

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Would you be willing, once your first certificate is successfully issued, to
share your email address with the Electronic Frontier Foundation, a founding
partner of the Let's Encrypt project and the non-profit organization that
develops Certbot? We'd like to send you email about our work encrypting the web,
EFF news, campaigns, and ways to support digital freedom.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
(Y)es/(N)o: y
Account registered.

Which names would you like to activate HTTPS for?
We recommend selecting either all domains, or all domains in a VirtualHost/server block.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
1: onestop.guru
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Select the appropriate numbers separated by commas and/or spaces, or leave input
blank to select all options shown (Enter 'c' to cancel): 1
```

7) Prepare the Certbot command -> sudo ln -s /snap/bin/certbot /usr/bin/certbot
8) Install certificate using this command -> sudo certbot –apache

Auto Renew the Certificates

contab -e

Enter this line of code in editor as shown in the figure below. 0 0 1 */2 * /usr/bin/certbot renew –quiet. This line of codes renews the https certificates every after 2 months.



Some of the lunix commands used

cd – Change directory. Ex> cd /home/ubunutu/second_products

mkdir – This creates the folder. Ex> mkdir foldername

ls – This list the folders in current directory. Ex> ls

touch – This creates the file. Ex> touch txt.py

pwd – This displays the path of the present working directory. Ex> pwd

sudo -s – Entering these commands enter into root directory. Ex sudo -s

ls -al – This list hidden and non-hidden files and folder. Ex ls -al

vi – This command mode enters into editing command mode of file. Ex vi file.py

cp – This copies files and folder from one directory to another directory. Ex> cp filename/folder /home/ubuntu/

mv – This moves or rename the file. Ex mv filename filename1, mv file /home/ubuntu.

rm – This removes the files and folder. Ex> rm file/folder

rm -f – This removes the parent folder plus inside files and folders forcefully. Ex> rm -f folder.

datetime – Displays date and time of server. Ex> datetime

chown -R – These commands of files and directories. Ex> sudo chown -R www-data:www-data /home/ubuntu/second_products

chomd – These changed the permission of file and folder. Ex> chmod 755 filename

7 (4 + 2 + 1): Read, write, and execute

6 (4 + 2): Read and write

5 (4 + 1): Read and execute

Scripts

Copying files and folders from one directory to another directory

vi backup_db.sh

```
#!/bin/bash
# This is a simple backup script

echo "Starting the backup process..."
cp -r /mnt/backup/ /home/ubuntu/db_backup
echo "Backup completed successfully!"
```

chomd +x backup_db.sh -> This makes files executable

./backup_db.sh -> this executes the commands

Copying files by checking new files

```
#!/bin/bash
# This script checks for new files in a directory and copies them to a backup location.

SOURCE_DIR="/mnt/backup"
BACKUP_DIR="/home/ubuntu/db_backup"
LOG_FILE="/mnt/backup/backup.log"

echo "Backup started at $(date)" >> $LOG_FILE

# Loop through all files in the source directory
for file in $SOURCE_DIR/*; do
  if [ -f "$file" ]; then  # Check if it is a file
    cp "$file" "$BACKUP_DIR"
    echo "Copied $file to $BACKUP_DIR" >> $LOG_FILE
  fi
done

echo "Backup completed at $(date)" >> $LOG_FILE
```

This script deletes all .tmp files in the specified directory.

```
#!/bin/bash
# This script deletes all .tmp files in the specified directory

TARGET_DIR="/path/to/directory"

echo "Deleting .tmp files in $TARGET_DIR"
rm -v $TARGET_DIR/*.tmp

echo "Deletion completed!"

-- INSERT --                                                    4,31          All
```

This script deletes files older than 30 days in the specified directory.

```
#!/bin/bash
# This script deletes files older than 30 days in the specified directory

TARGET_DIR="/home/ubuntu/db_backup"
DAYS_OLD=30

echo "Deleting files older than $DAYS_OLD days in $TARGET_DIR"
find $TARGET_DIR -type f -mtime +$DAYS_OLD -exec rm -v {} \;

echo "Deletion completed!"
```

This script deletes empty directories in a specified directory.

```
#!/bin/bash
# This script deletes empty directories in the specified directory

TARGET_DIR="/home/ubuntu/db_backup"

echo "Deleting empty directories in $TARGET_DIR"
find $TARGET_DIR -type d -empty -exec rmdir -v {} \;

echo "Deletion completed!"
```

This script deletes all files in a specified directory.

```
#!/bin/bash
# This script deletes all files in the specified directory
# WARNING: This will delete all files in the directory!

TARGET_DIR="/home/ubuntu/db_backup"

echo "Deleting all files in $TARGET_DIR"
rm -v $TARGET_DIR/*

echo "Deletion completed!"
```

This script moves all files and directories from a source directory to a destination directory.

```
#!/bin/bash
# This script moves all files and directories from the source directory to the destination directory

SOURCE_DIR="/home/ubuntu/test"
DEST_DIR="/home/ubuntu/db_backup"

echo "Moving all files and directories from $SOURCE_DIR to $DEST_DIR"
mv $SOURCE_DIR/* $DEST_DIR/

echo "Move completed!"
```

References

https://certbot.eff.org/instructions?ws=apache&os=ubuntufocal

https://stackoverflow.com/questions/23202146/pkg-config-script-could-not-be-found-on-osx

https://www.youtube.com/watch?v=kmRQ2Z0-Si0

https://medium.com/django-unleashed/django-project-structure-a-comprehensive-guide-4b2ddbf2b6b8

https://crontab.guru/