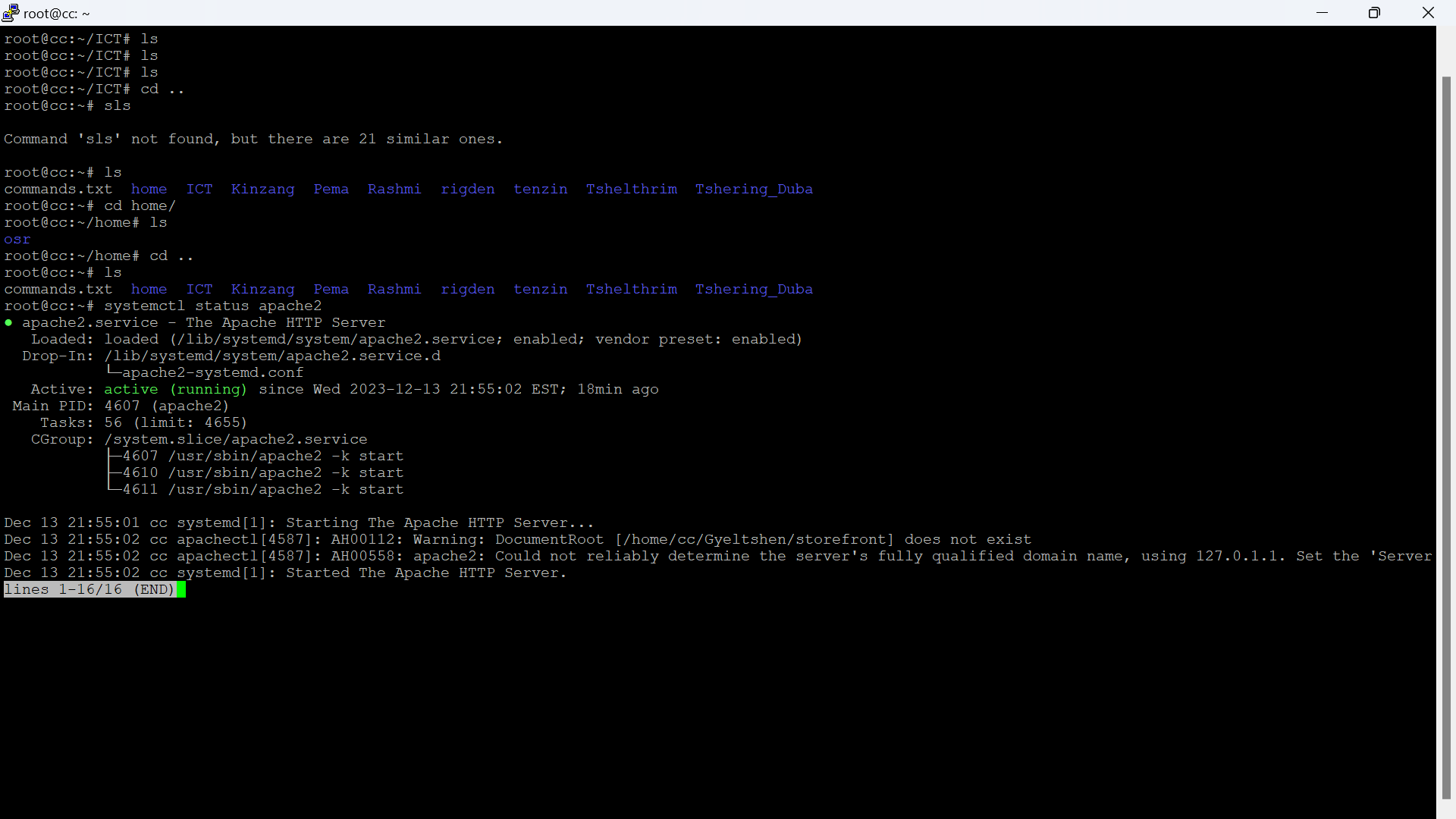
**How to Install Django with Apache on Ubuntu server?**

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

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

Step 3: systemctl start apache2 (This commands 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

## **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 commands: python3 -m venv *envfoldername.*

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

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

Step 5: pip install mysqlclient(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

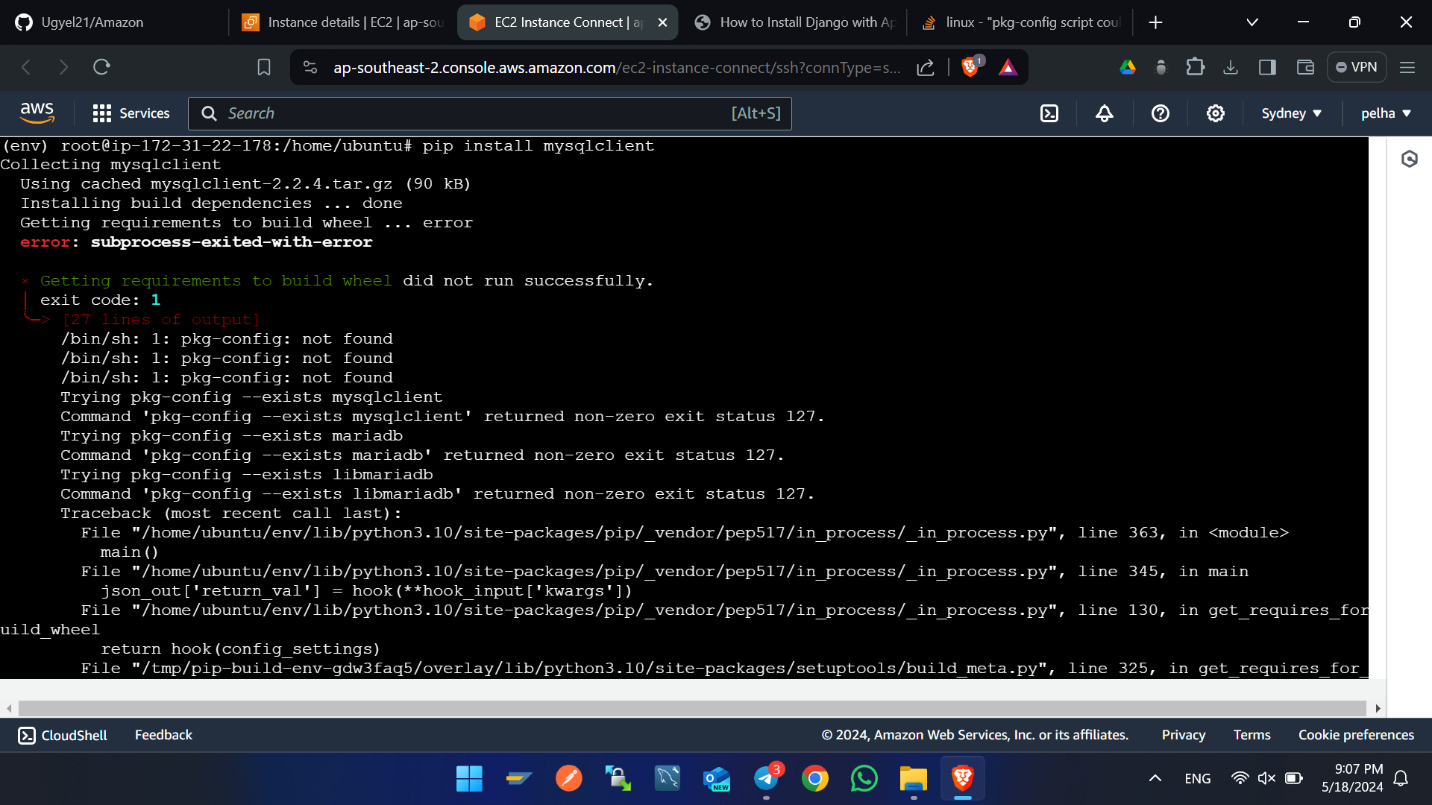
tail –f /var/log/apache2/error.log

tail –f /var/log/mysql/error.log

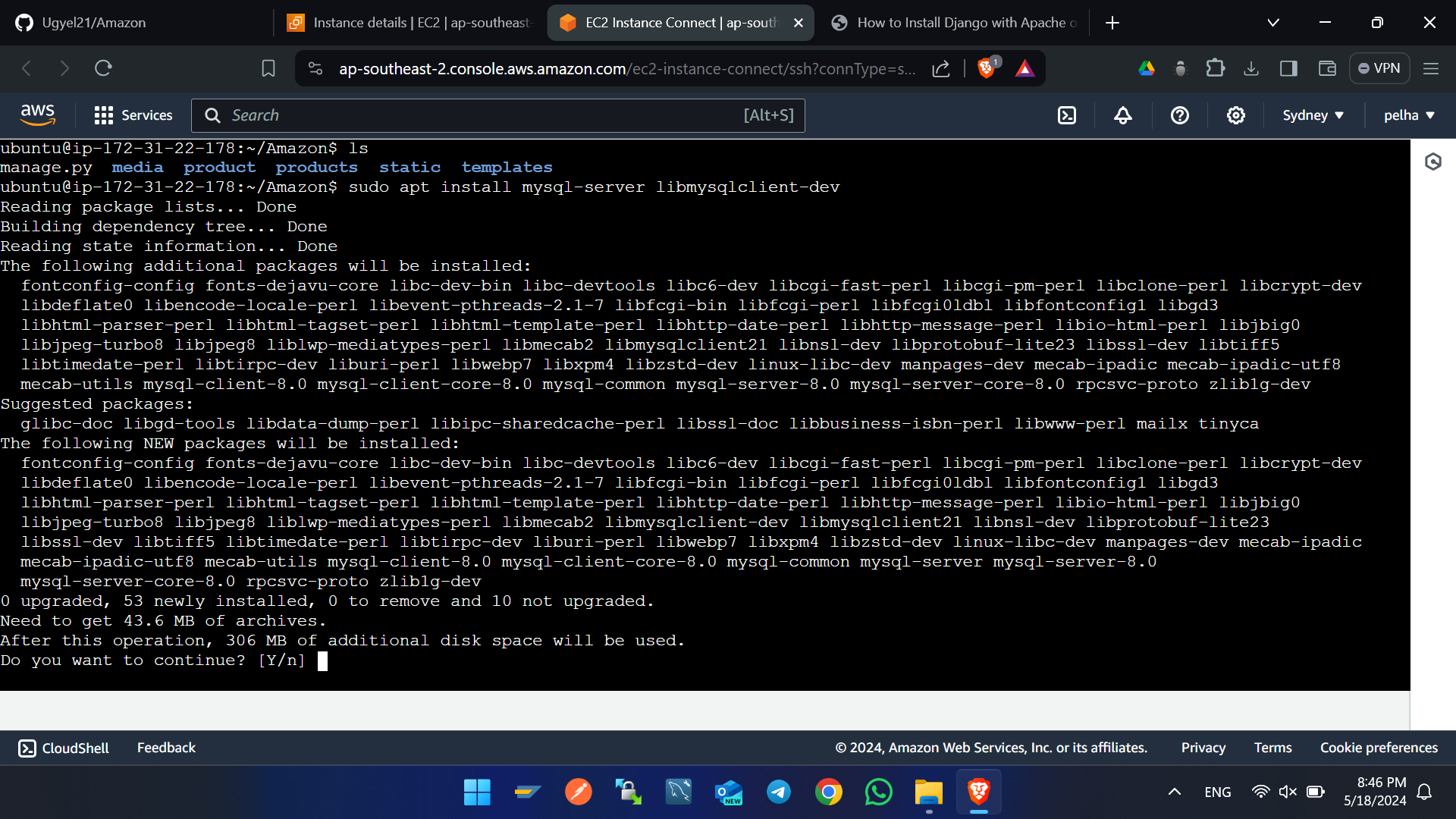
tail -f /var/log/syslog

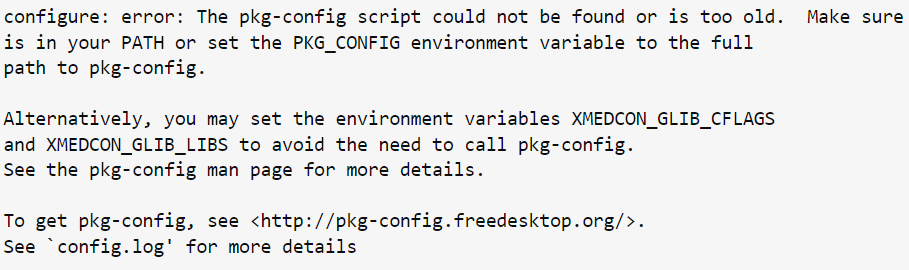
**Some of the issue face and solution**

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

****

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

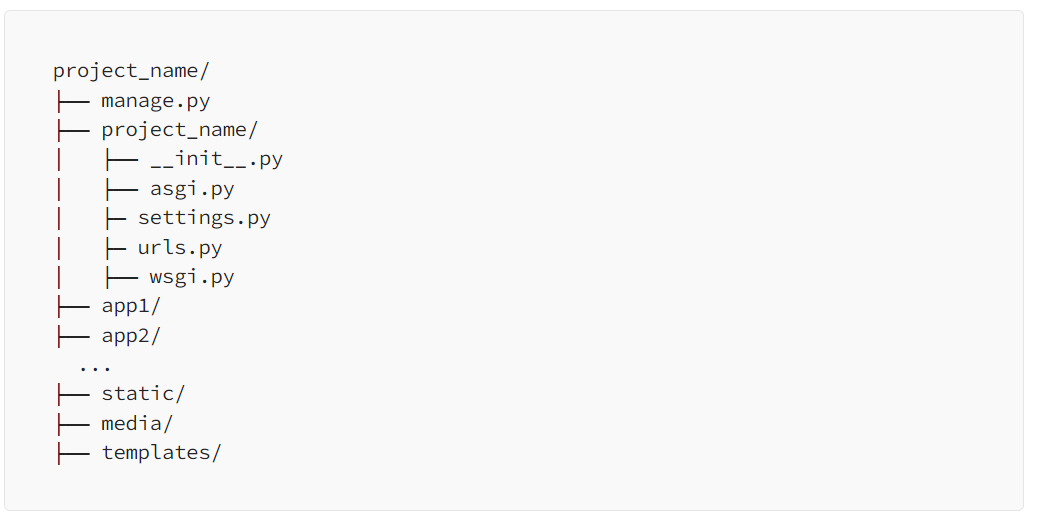
****

Error below

Solution: apt-get install -y pkg-config

**Best practice for project structure**

## **Directory Hierarchy**



**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**

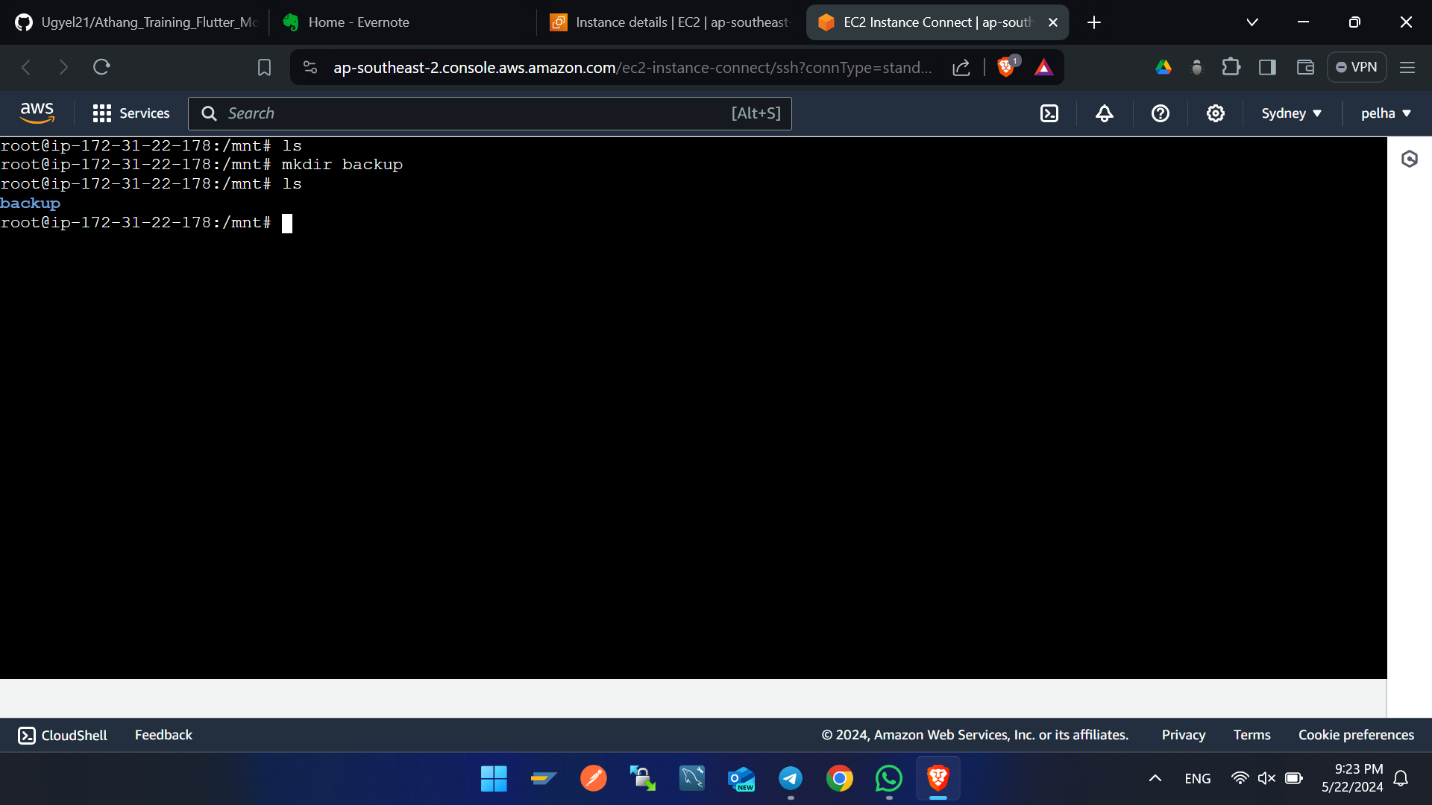
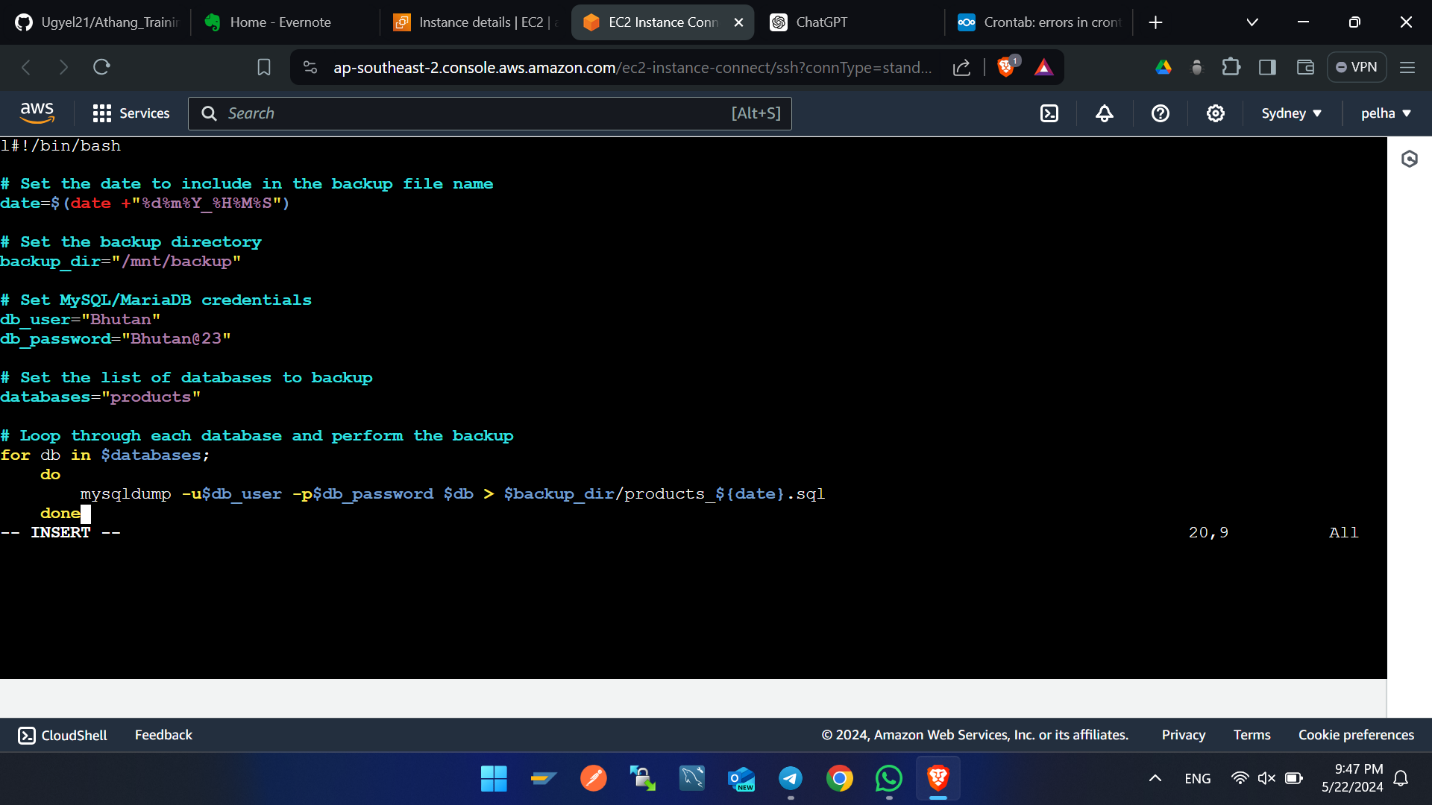
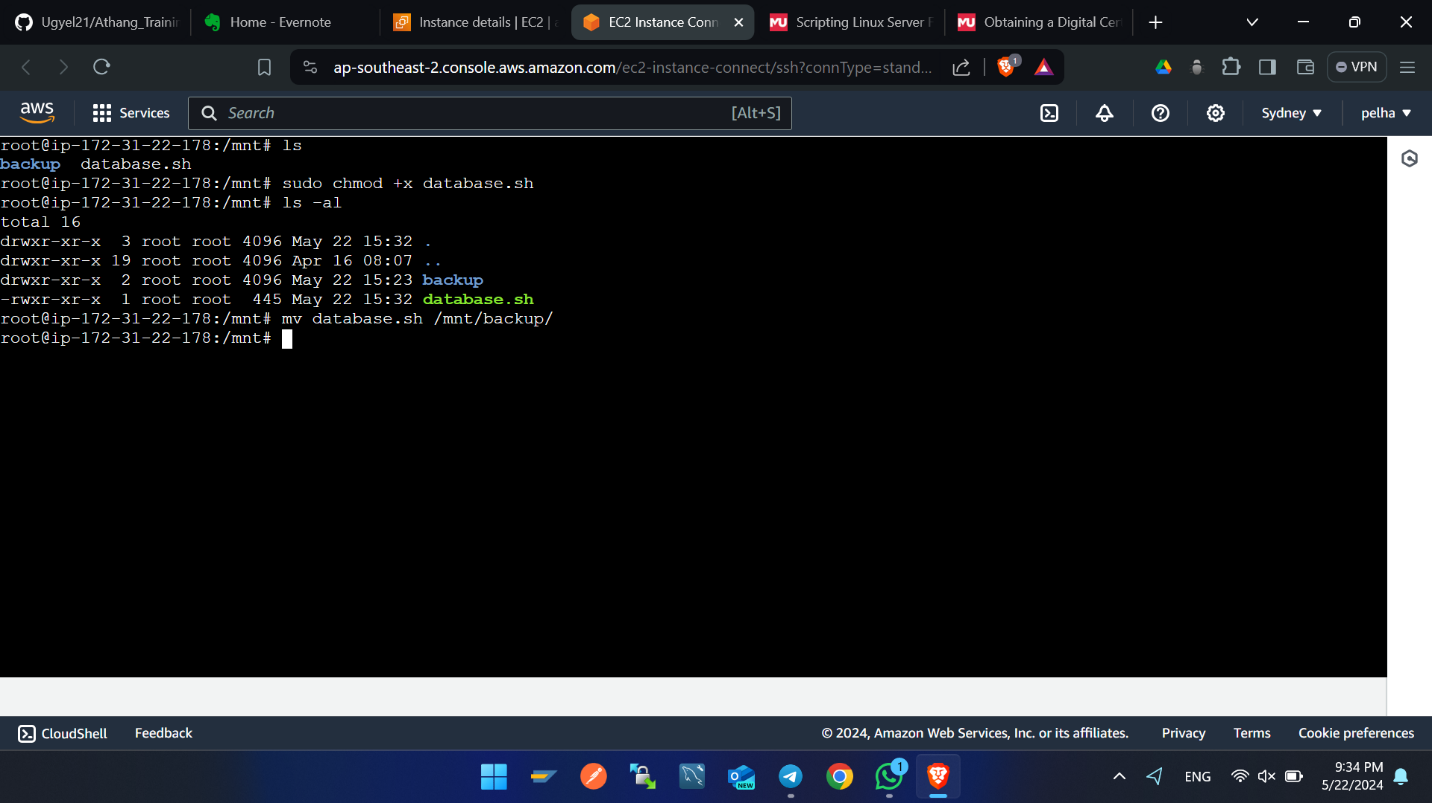
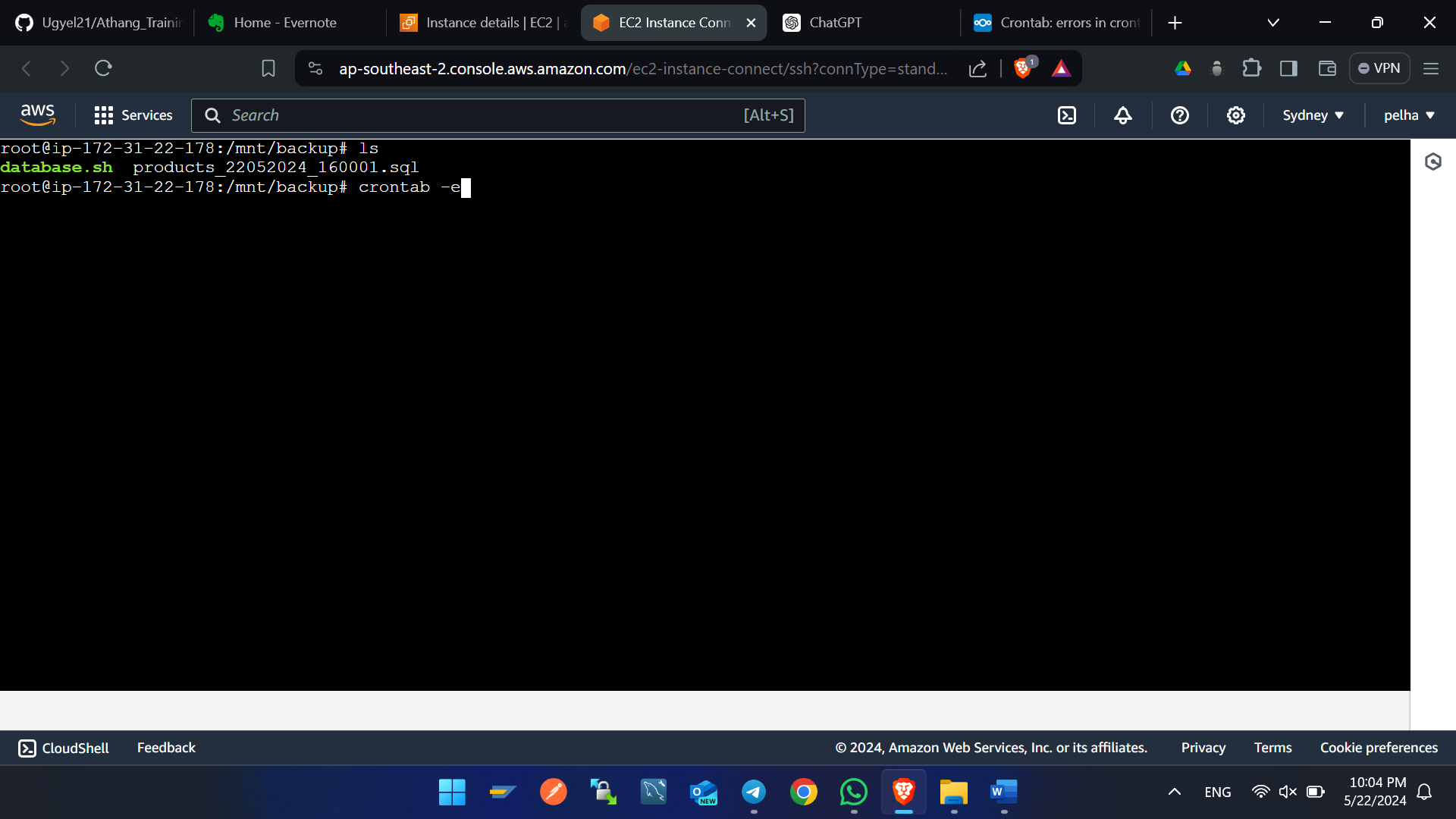
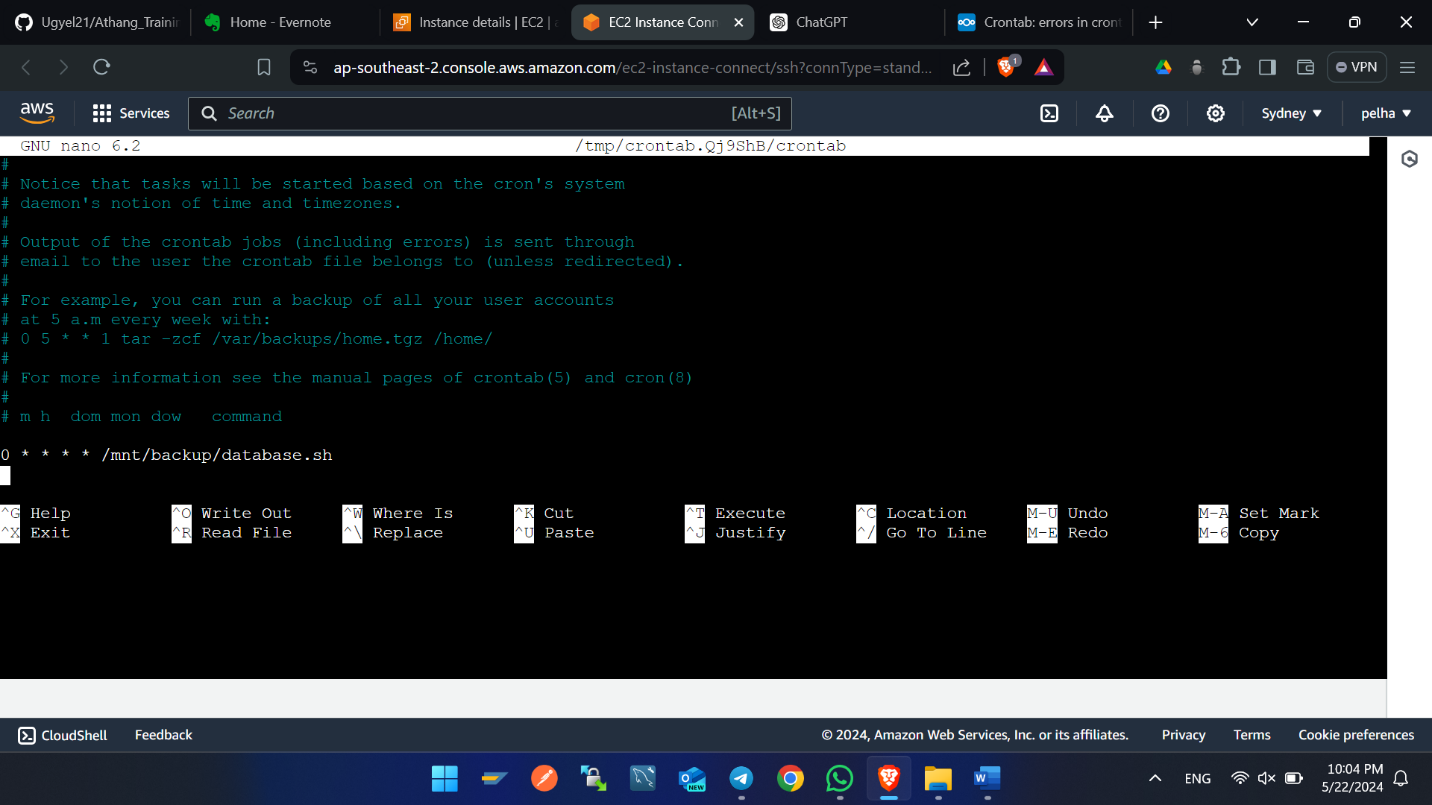
Organize your codebase into modular components, making each component focused on a specific task. This makes code easier to understand, test, and maintain.

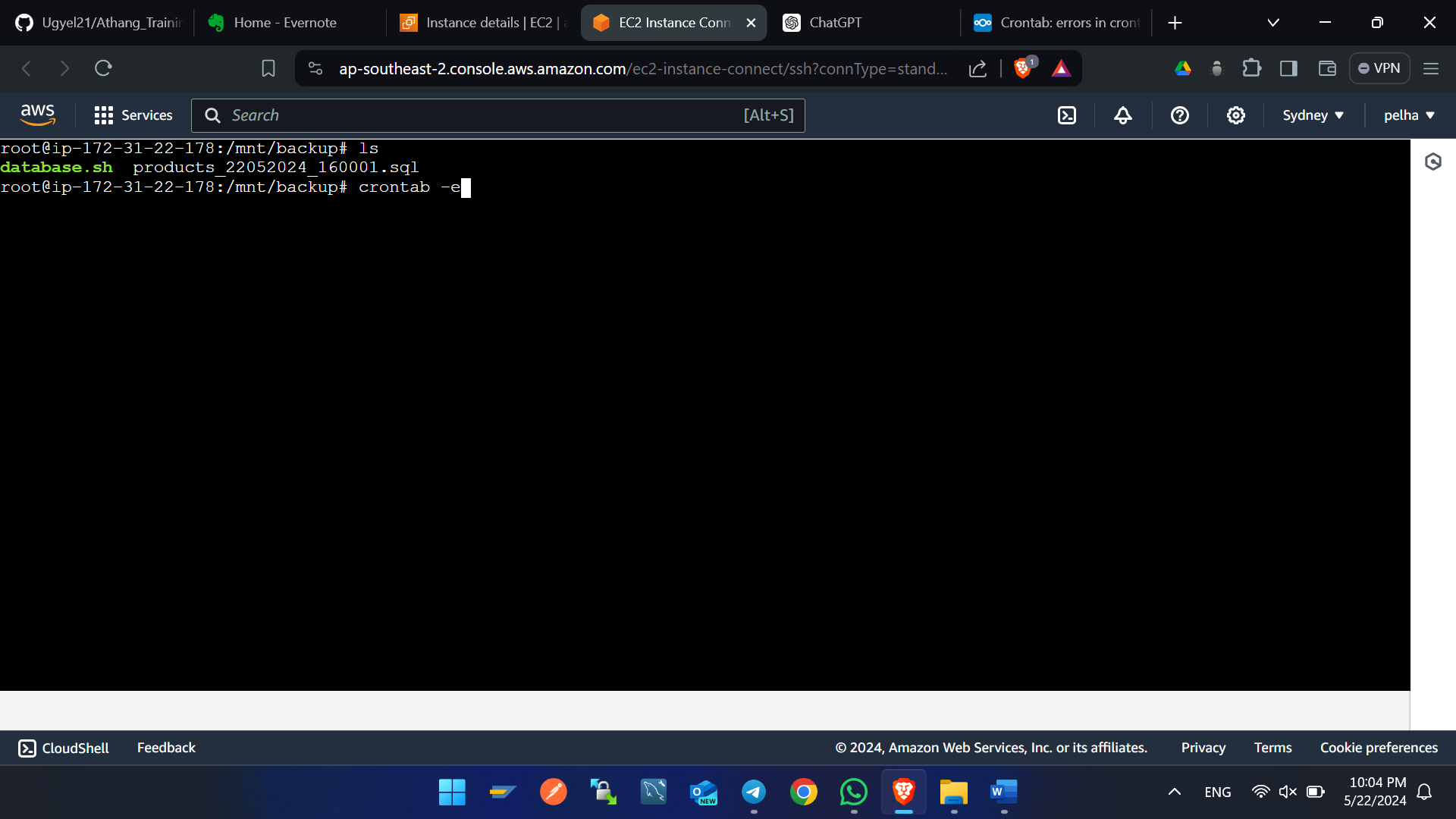
* Apps: Divide your project into multiple apps based on functionality. Each app should be self-contained, handling a specific feature.
* Views: Keep your views concise and focused on handling HTTP requests. Use class-based views for better organization.
* Models: Organize models in a way that reflects 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**

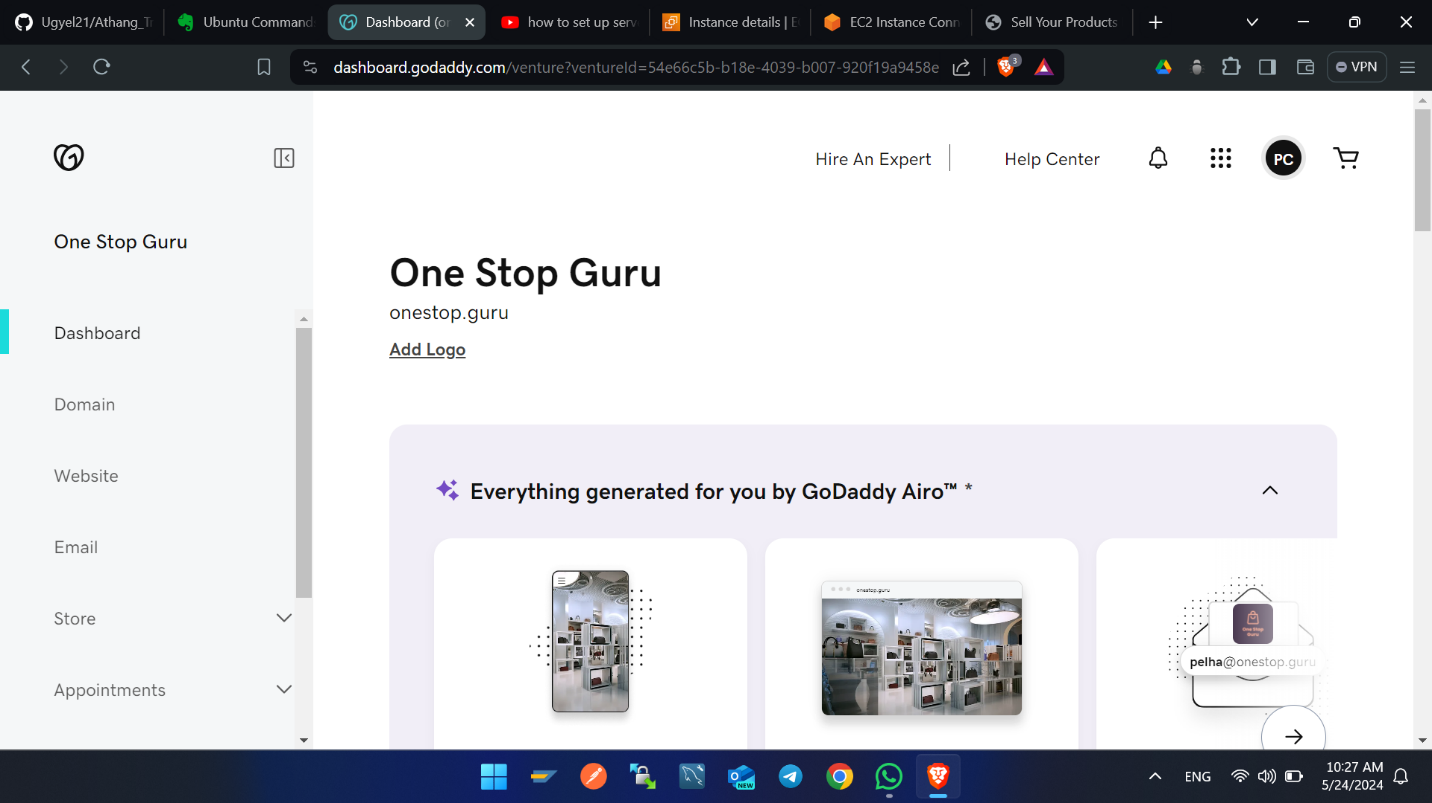
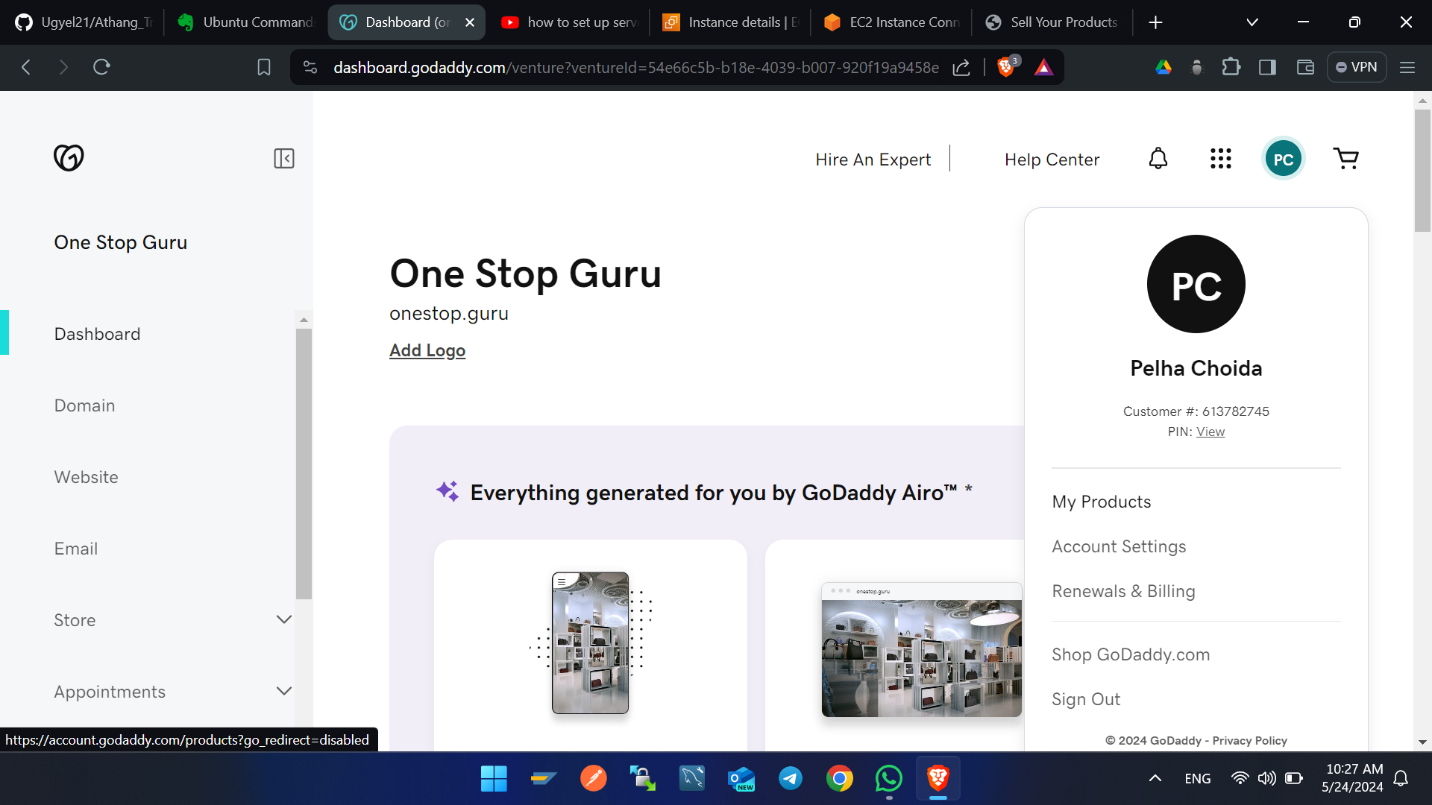
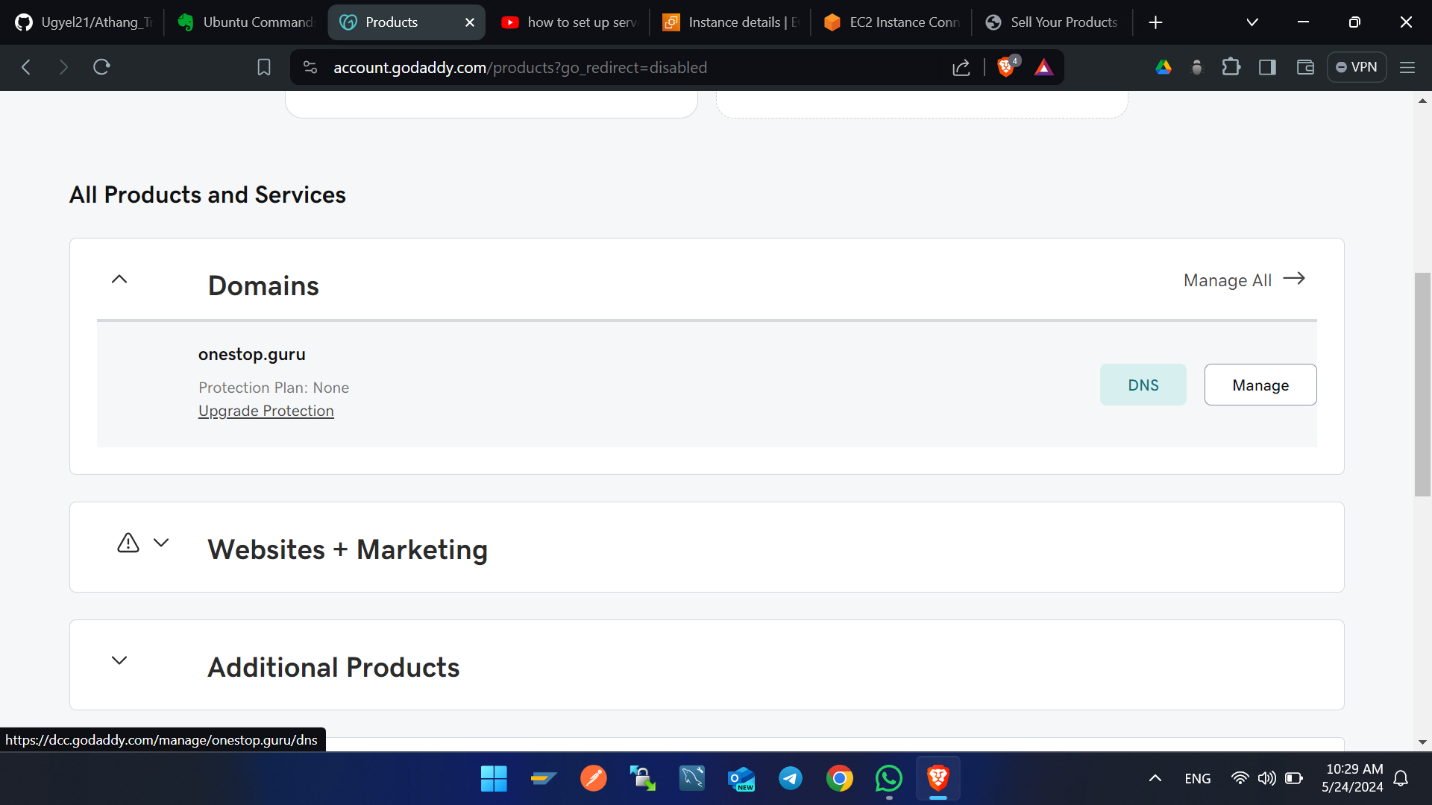
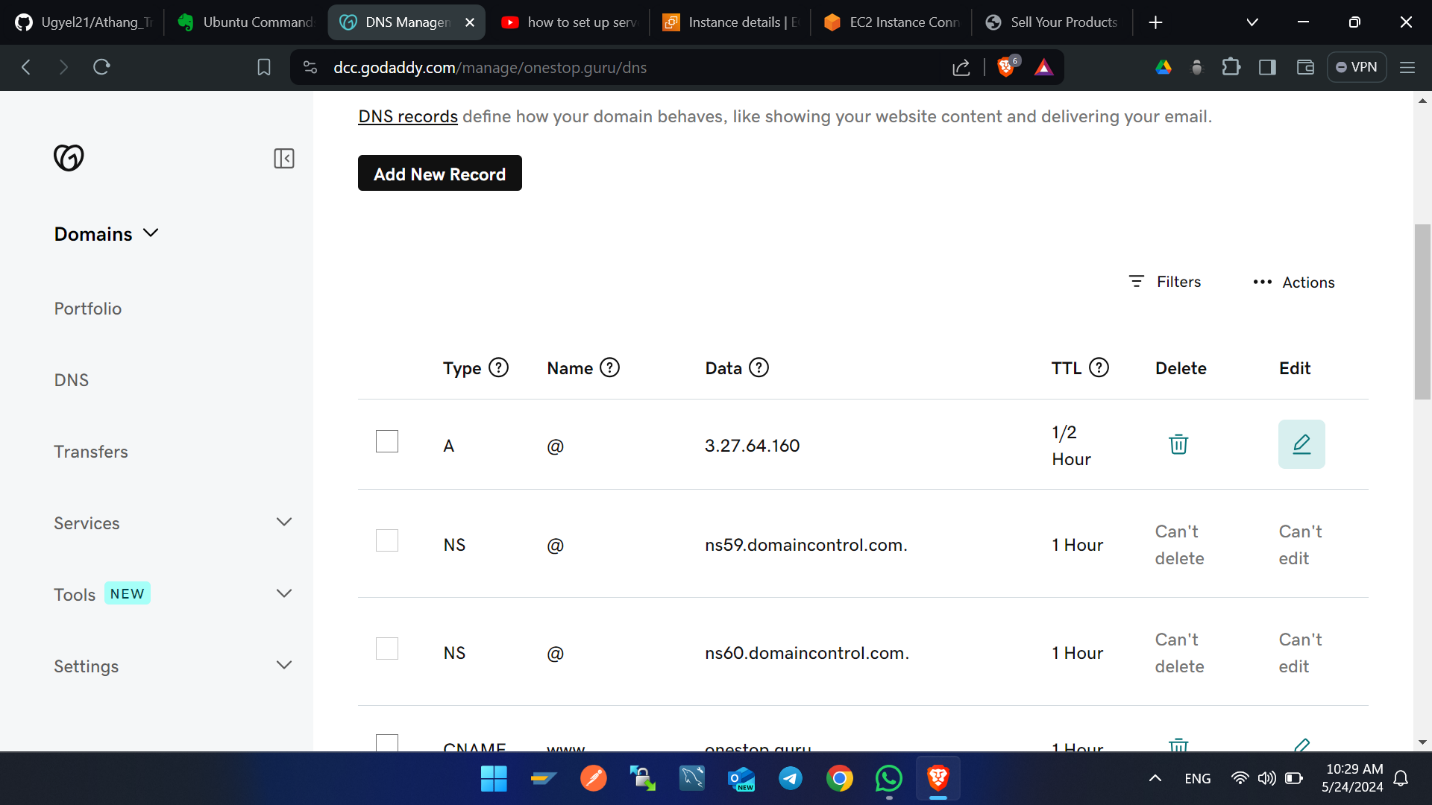
Keep your project settings in the settings.py file within the inner project directory. Use environment variables for sensitive information.

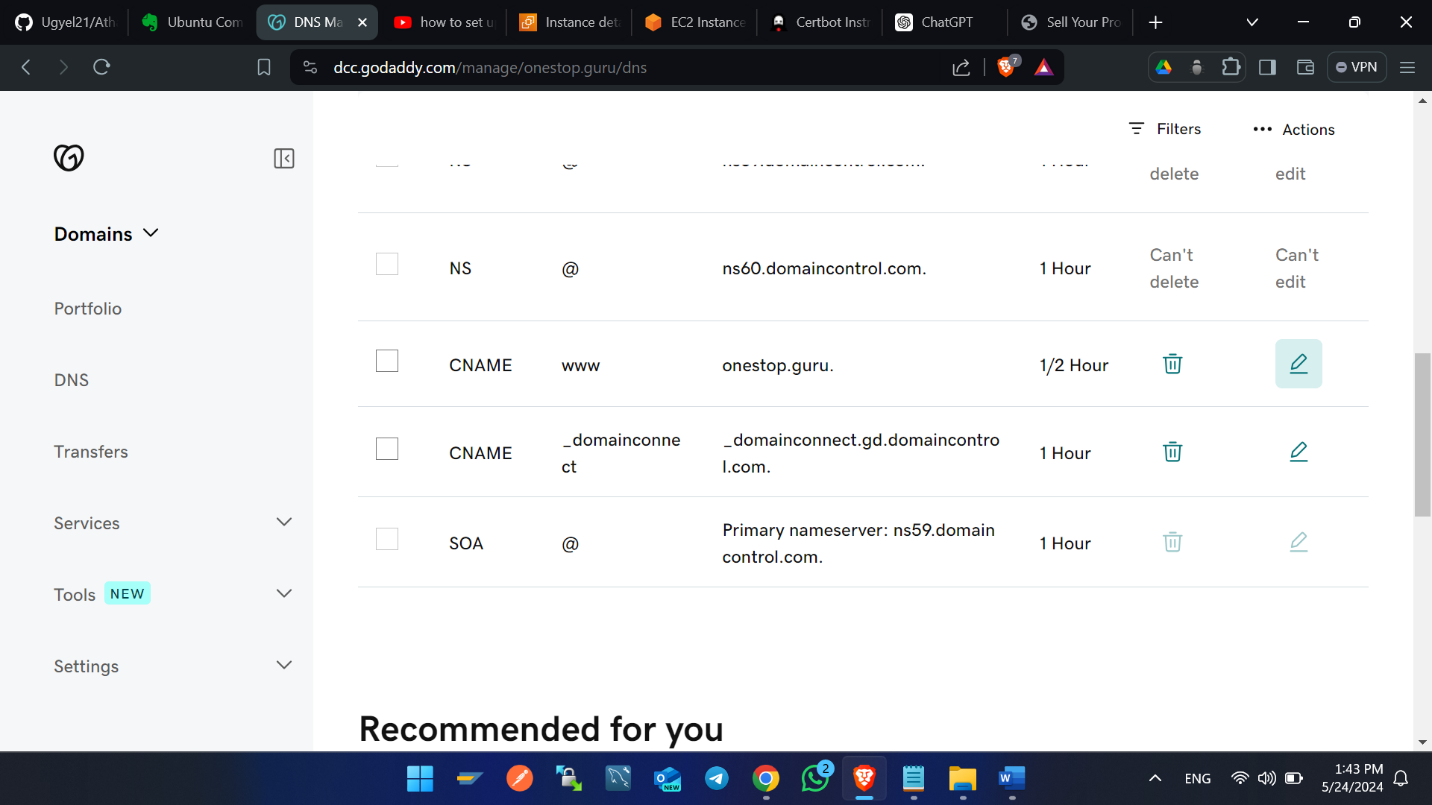
**Database backup using Script and Cron job**

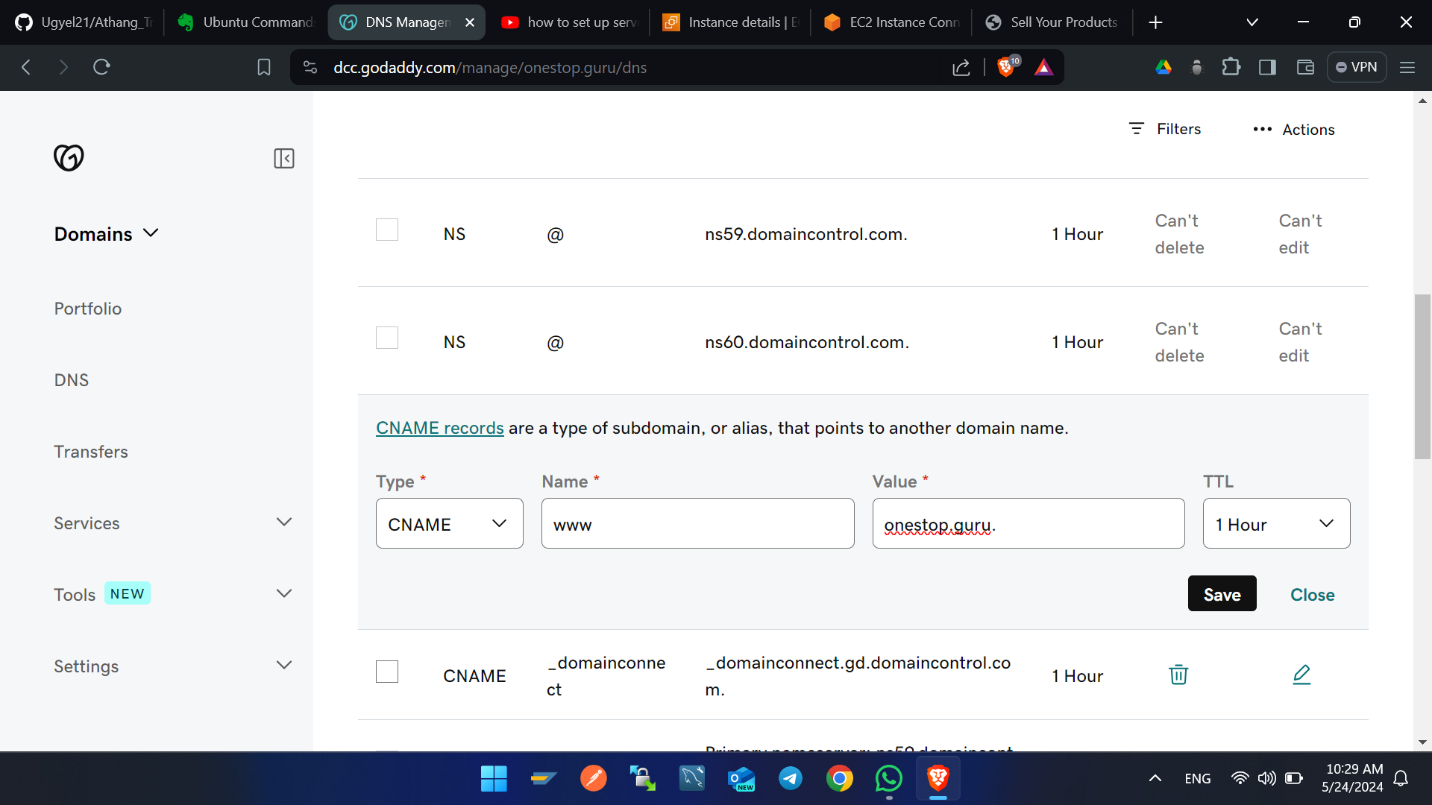
1. Create folder in directory /mnt
2. To go to above directory cd /mnt
3. The create folder called backup by using this command mkdir backup
4. Inside backup folder creates one script file called database
5. Command to create backup file -> touch database.sh
6. Enter this line codes using by opening above database.sh file by command nano database.sh
7. 
8. Set the permission of the database.sh file by this command sudo chmod +x database.sh
9. 
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.
11. This are the files auto backup using crontab and script file.

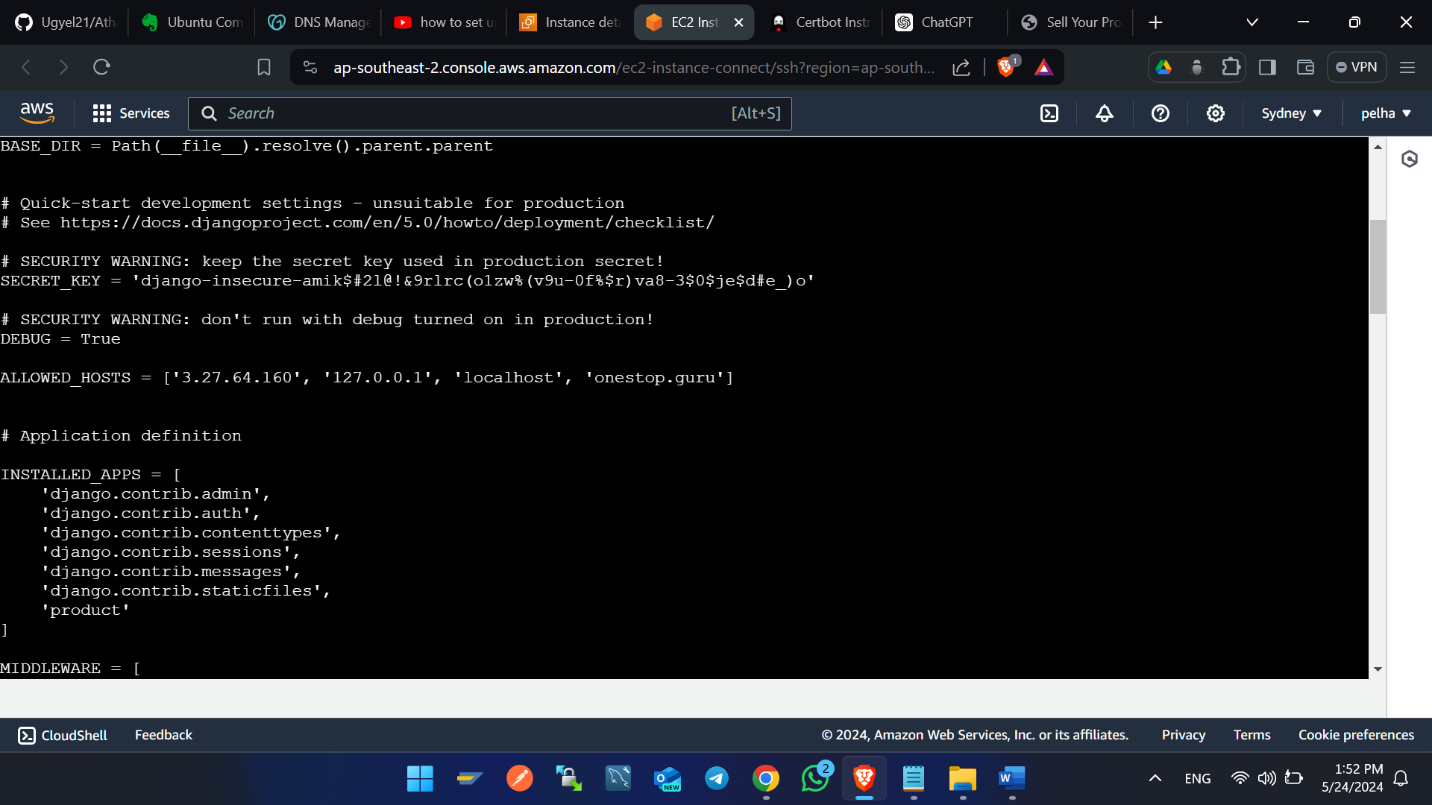


**Setting up the Go Daddy Domain Name with server**

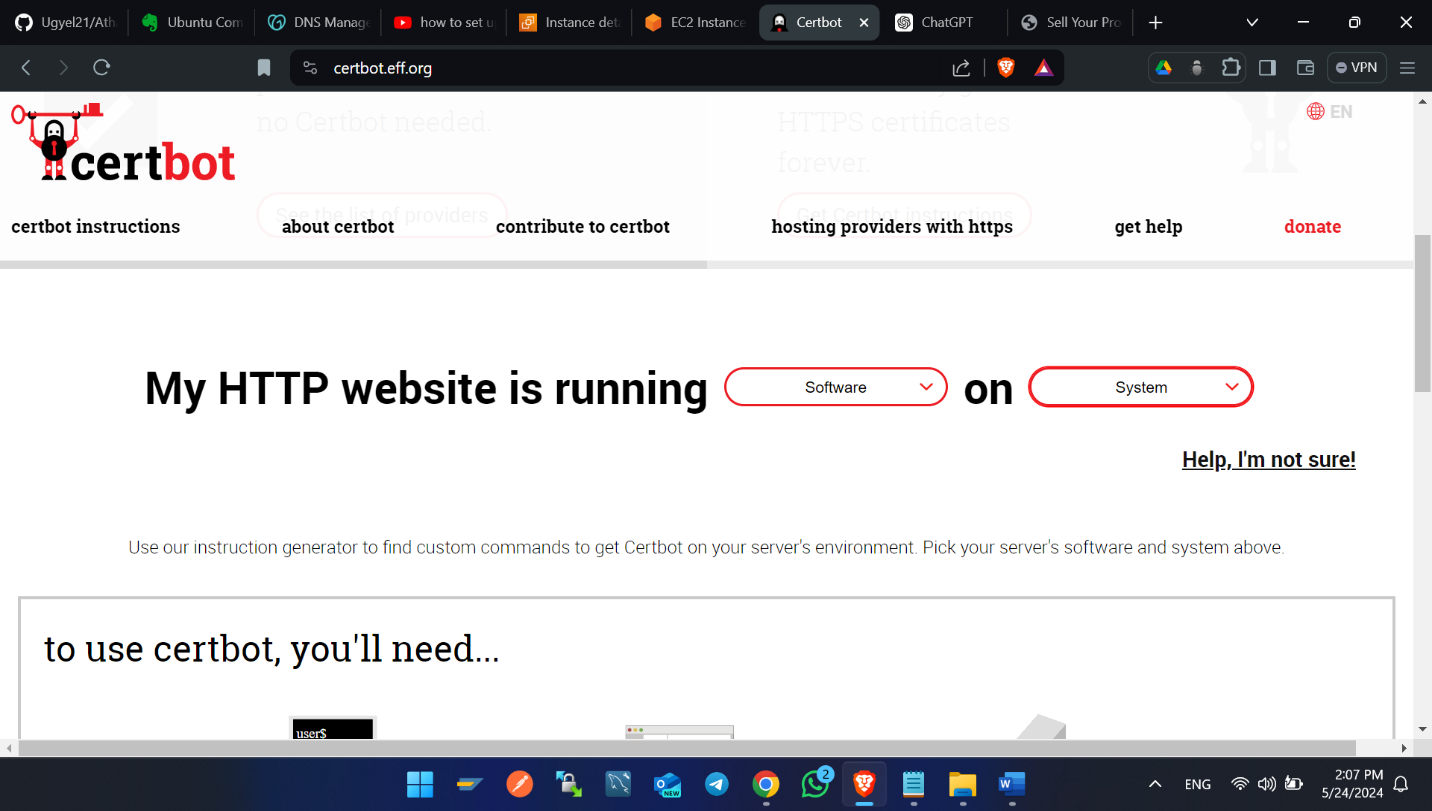
1. Vist -> <https://www.godaddy.com/en-uk>
2. Sign with credentials
3. In nav bar you will find profile logo, click on that.
4. Click on my products.
5. Click on DNS.
6. Click on edit button of type A. Enter the IP of server in value filed. Then click save.
7. Click on edit button of type CNAME and name www.

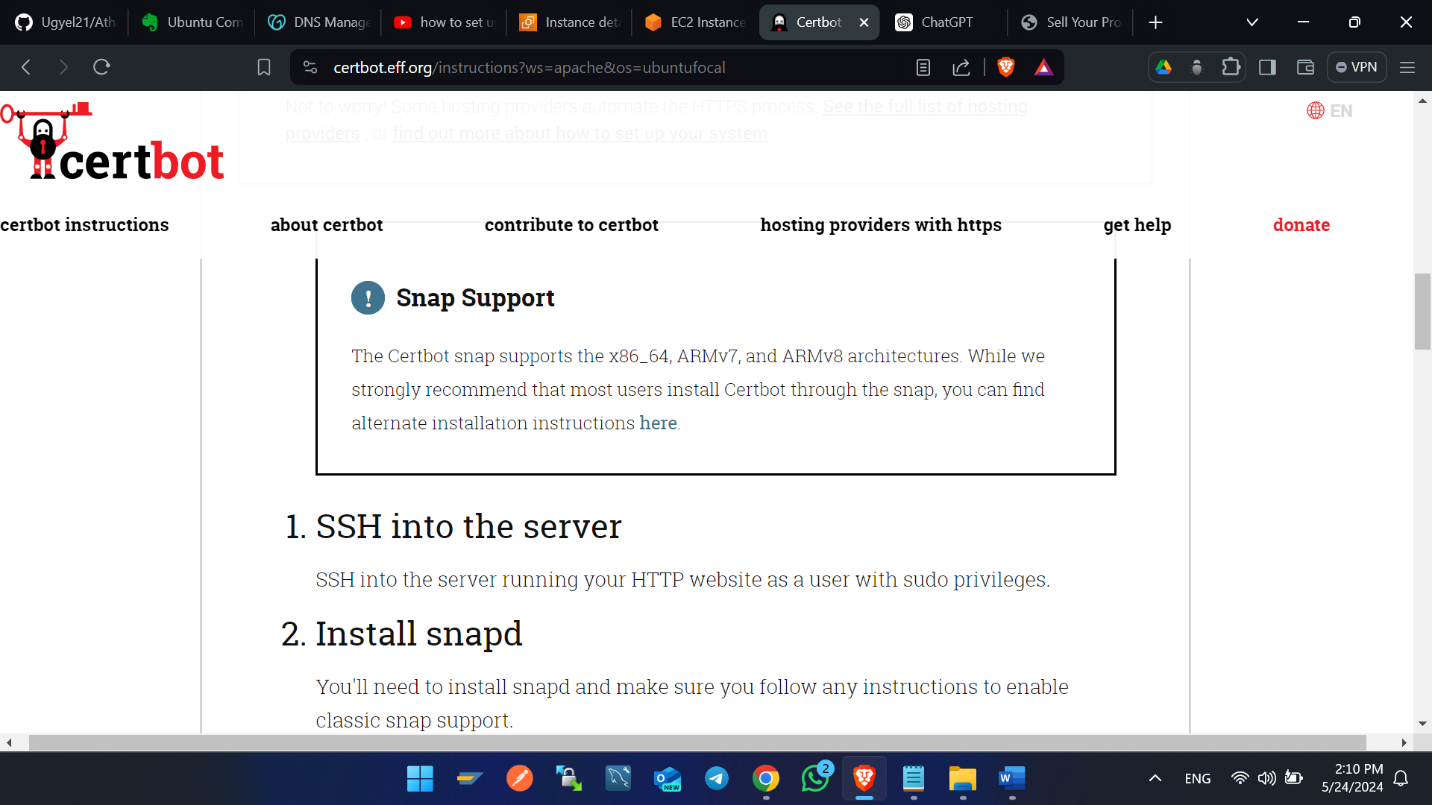


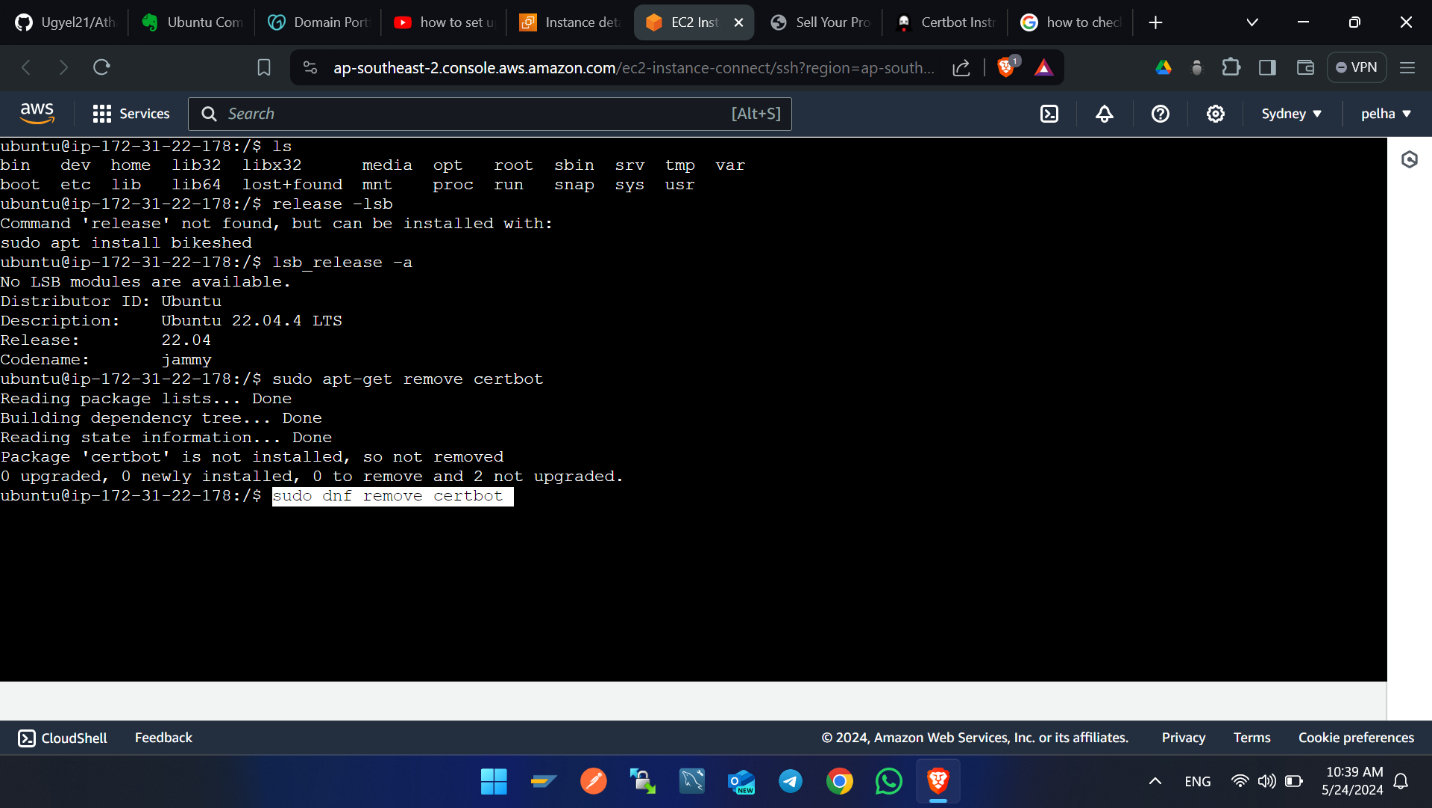
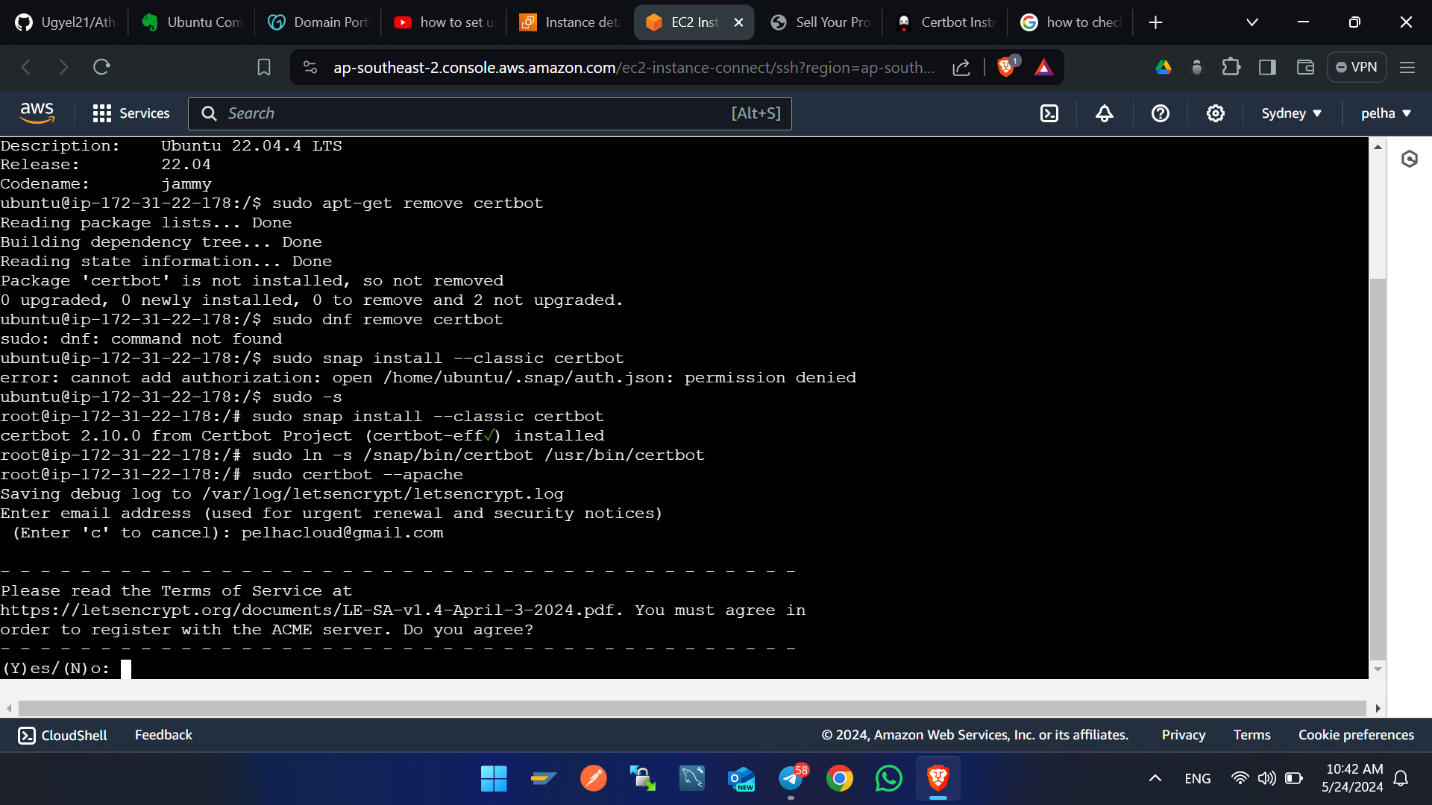
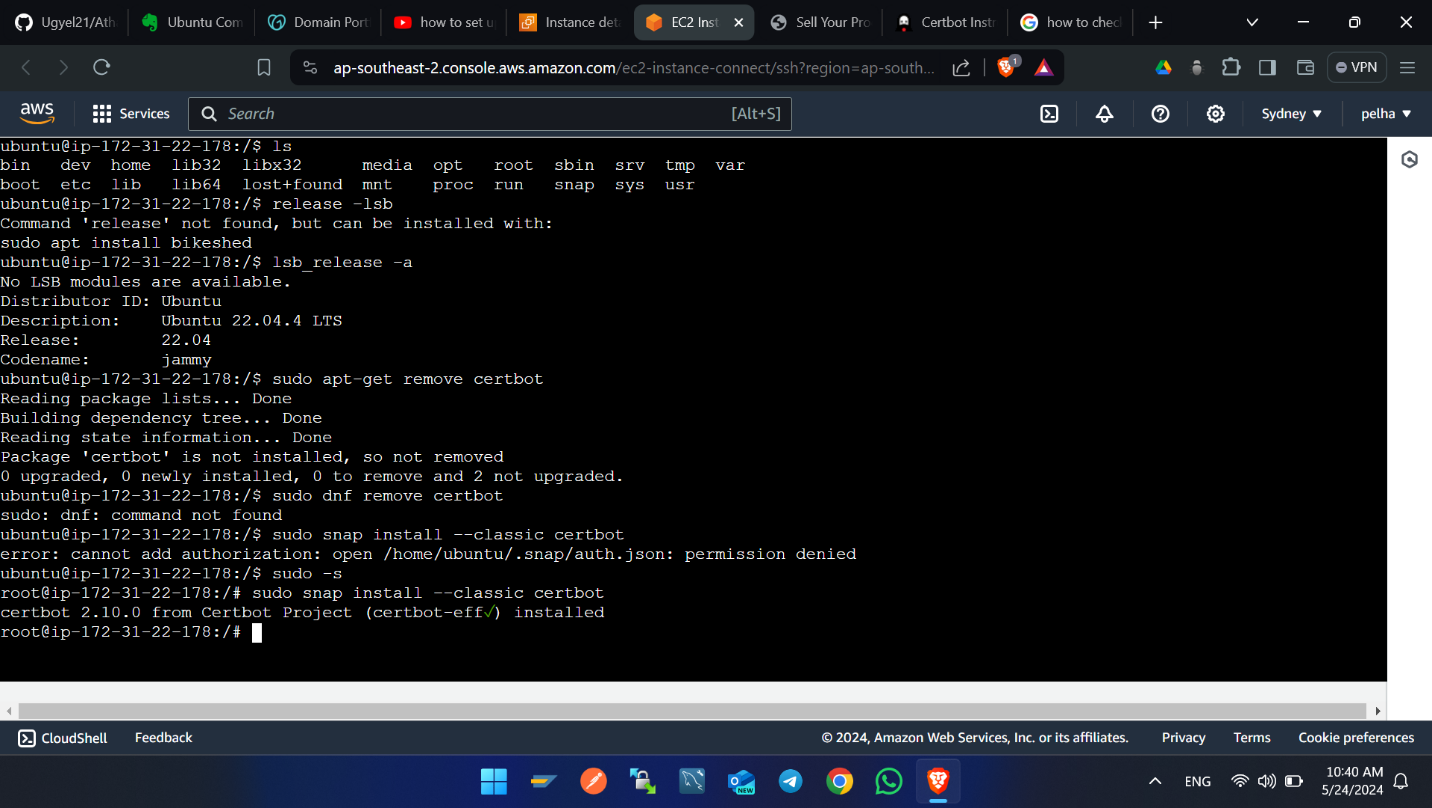
1. Enter the value as your domain name.

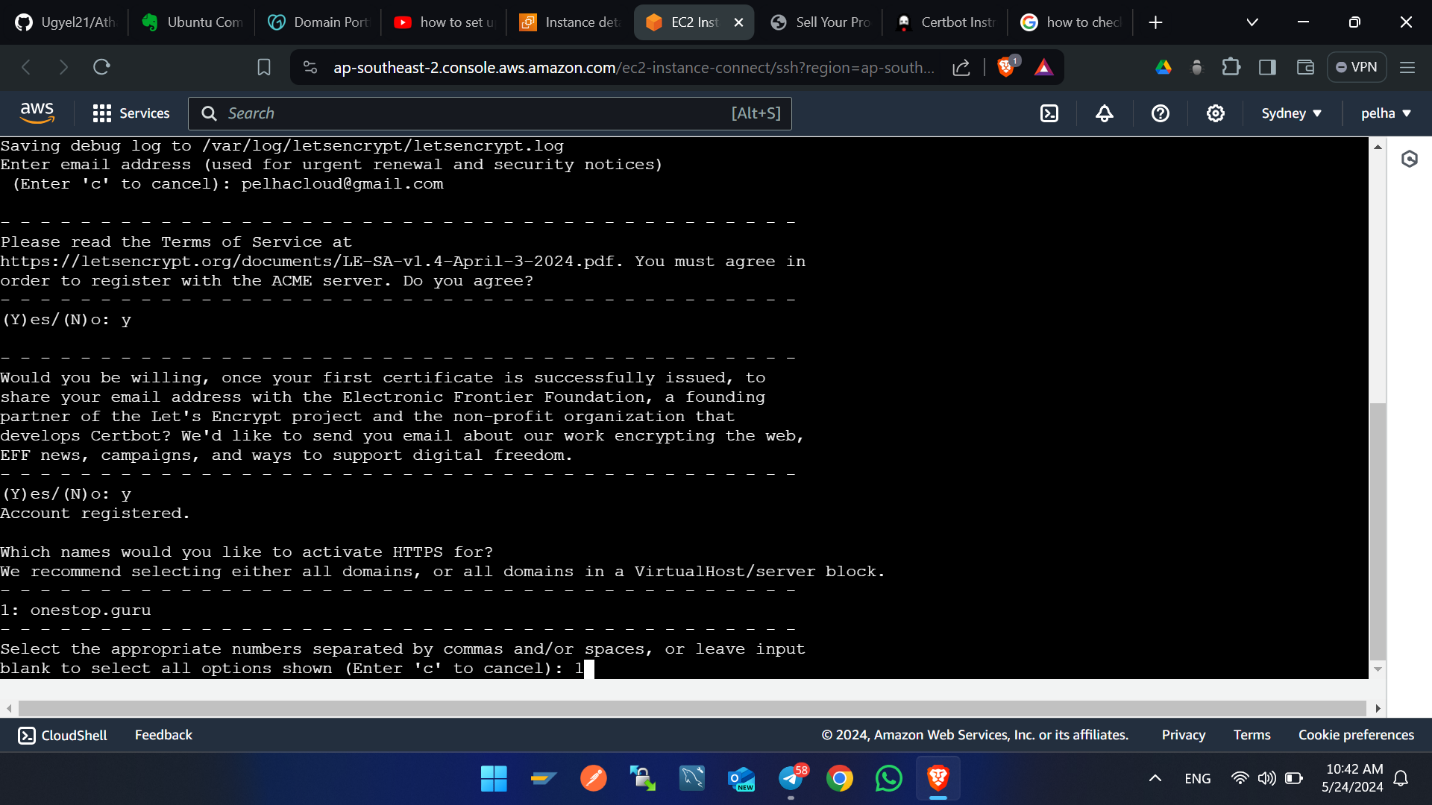
After Domain Name setup, enter the domain name in setting file of your Django project in the allowed host array.

**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.



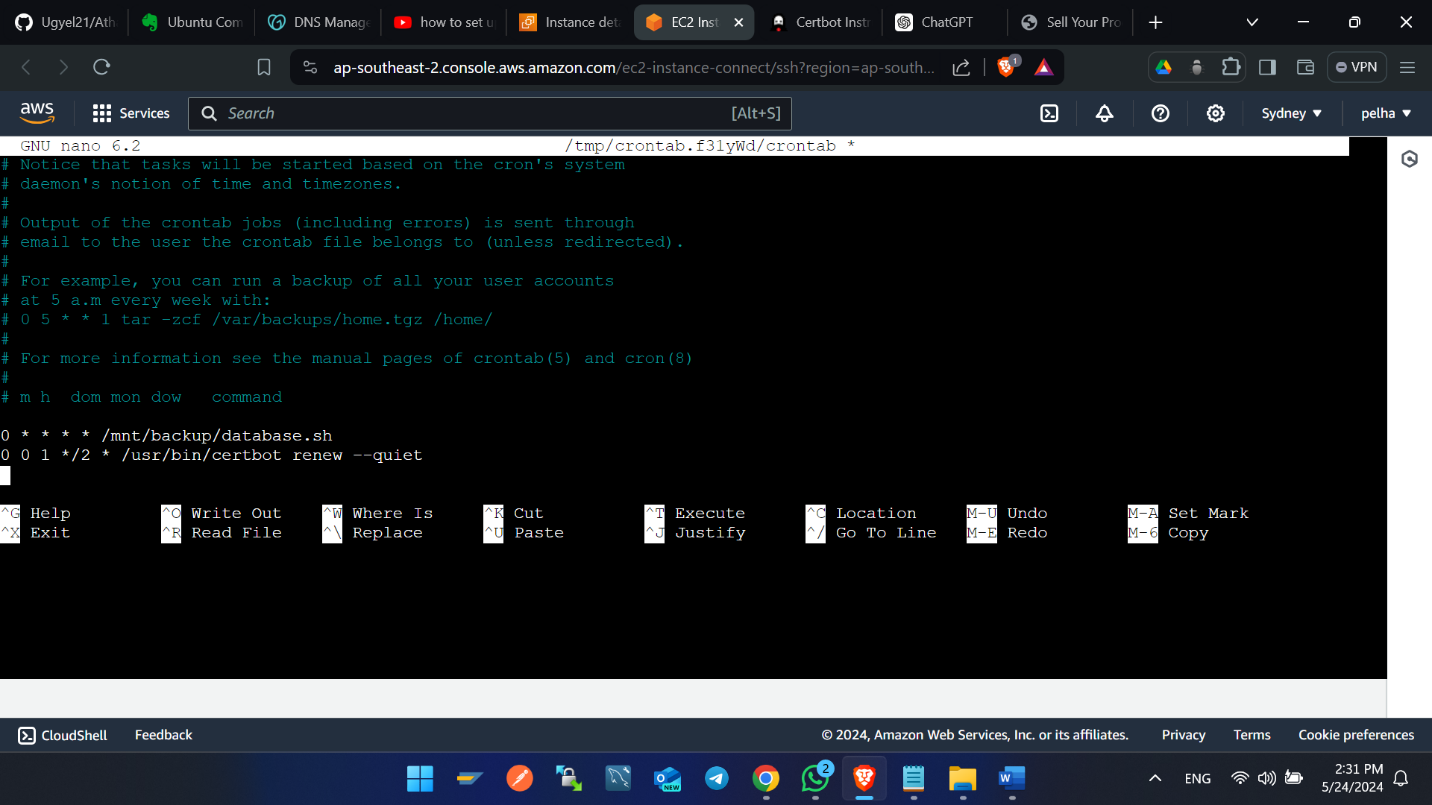
1. Run sudo apt-get remove certbot, sudo dnf remove certbot. This command removes if cerbot is installed before.
2. Run sudo snap install --classic certbot. Enter information if it is pop up and enter y to confirm and hit enter.
3. Select domain name hit enter. After that the https will finally configured.



1. Prepare the Certbot command -> sudo ln -s /snap/bin/certbot /usr/bin/certbot
2. 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 Script commands used using building server and building website**

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

**References link**

<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/>