Link to GitHub repository: https://github.com/Rahul5823/Webserver

Writing a Parser:

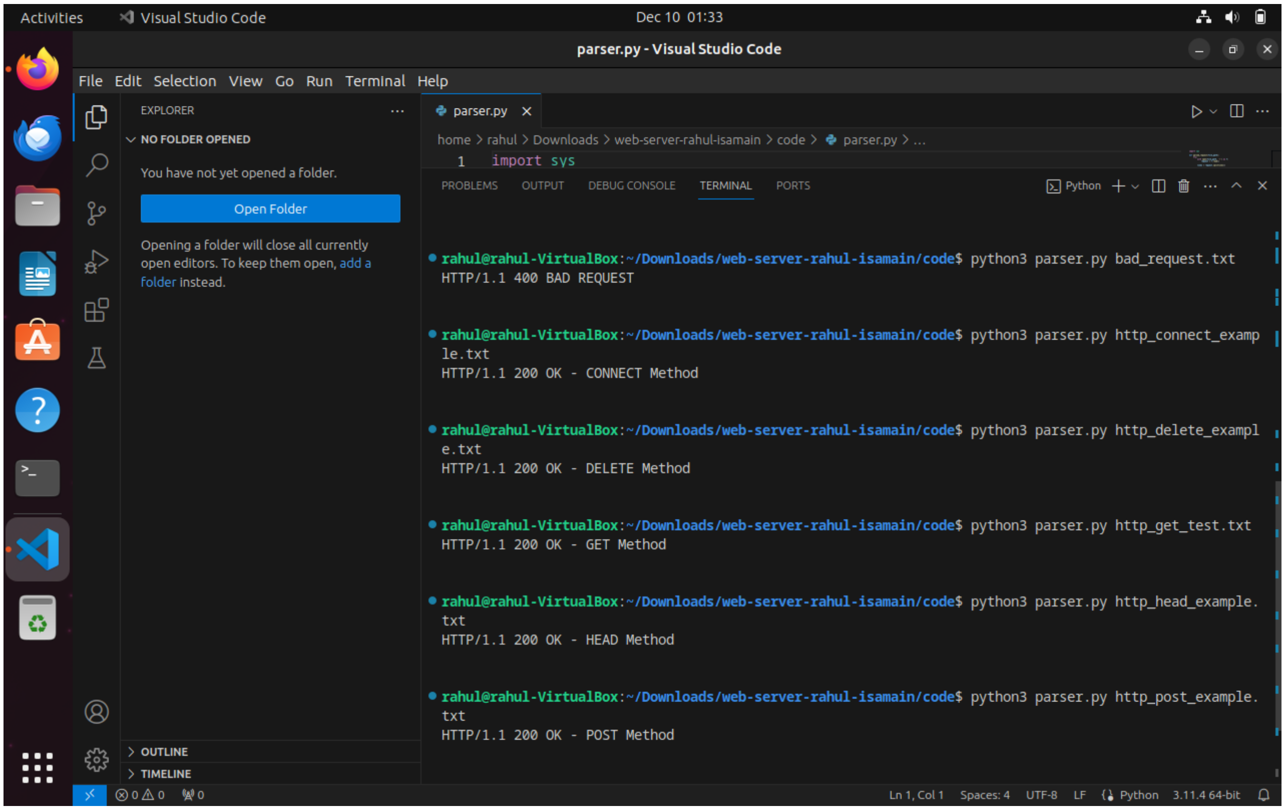
- Wrote a parser that checks whether the request made is valid or not and generate valid response for the request.

- My parser file named 'parser.py' is a python file that first checks if the request method name is one of the following: GET, POST, PUT, DELETE, HEAD and CONNECT.

- If valid, then check for the body and version of the request sent.

- If everything is good and valid, will return 200 with name of the methods as shown below. Or else, returns 400 as Bad Request.

- Will also return 500 Internal Server Error if exception.


Activities
Visual Studio Code
Dec 10 01:33
parser.py - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
NO FOLDER OPENED
You have not yet opened a folder.
Open Folder
Opening a folder will close all currently open editors. To keep them open, add a folder instead.
OUTLINE
TIMELINE
parser.py
home > rahul > Downloads > web-server-rahul-isamain > code > parser.py > ...
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Python
Ln 1, Col 1 Spaces: 4 UTF-8 LF Python 3.11.4 64-bit

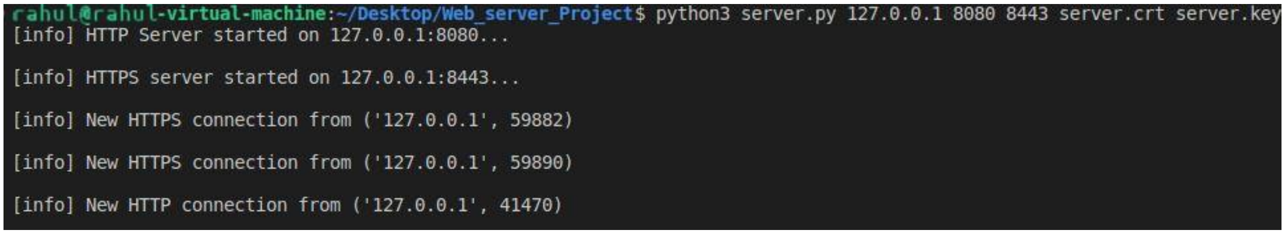
```python
import sys
```

```
rahul@rahul-VirtualBox:~/Downloads/web-server-rahul-isamain/code$ python3 parser.py bad_request.txt
HTTP/1.1 400 BAD REQUEST

rahul@rahul-VirtualBox:~/Downloads/web-server-rahul-isamain/code$ python3 parser.py http_connect_examp
le.txt
HTTP/1.1 200 OK - CONNECT Method

rahul@rahul-VirtualBox:~/Downloads/web-server-rahul-isamain/code$ python3 parser.py http_delete_exampl
e.txt
HTTP/1.1 200 OK - DELETE Method

rahul@rahul-VirtualBox:~/Downloads/web-server-rahul-isamain/code$ python3 parser.py http_get_test.txt
HTTP/1.1 200 OK - GET Method

rahul@rahul-VirtualBox:~/Downloads/web-server-rahul-isamain/code$ python3 parser.py http_head_example.
txt
HTTP/1.1 200 OK - HEAD Method

rahul@rahul-VirtualBox:~/Downloads/web-server-rahul-isamain/code$ python3 parser.py http_post_example.
txt
HTTP/1.1 200 OK - POST Method
```

Fig (a): Output of parser.py

Turning Parser into a Server:

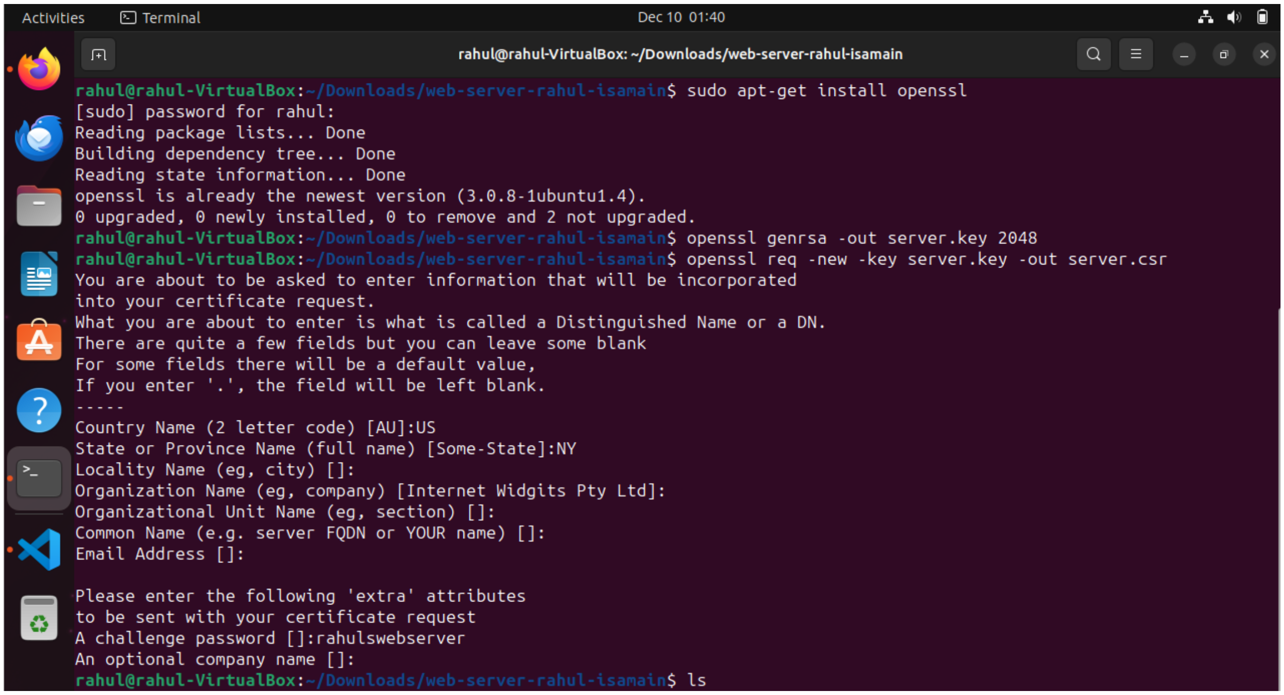
- We turned the parser above into a server, in such a way that it has four line arguments – IP address, Port numbers, Path to Certificate and Path to Private Key associated with x509 certificate.



```
rahul@rahul-virtual-machine:~/Desktop/Web_server_Project$ python3 server.py 127.0.0.1 8080 8443 server.crt server.key
[info] HTTP Server started on 127.0.0.1:8080...

[info] HTTPS server started on 127.0.0.1:8443...

[info] New HTTPS connection from ('127.0.0.1', 59882)

[info] New HTTPS connection from ('127.0.0.1', 59890)

[info] New HTTP connection from ('127.0.0.1', 41470)
```

Fig (b): Output of HTTPS request on server

- So, we need to generate both Certificate and Private Key, if we want to run it as HTTPS request.

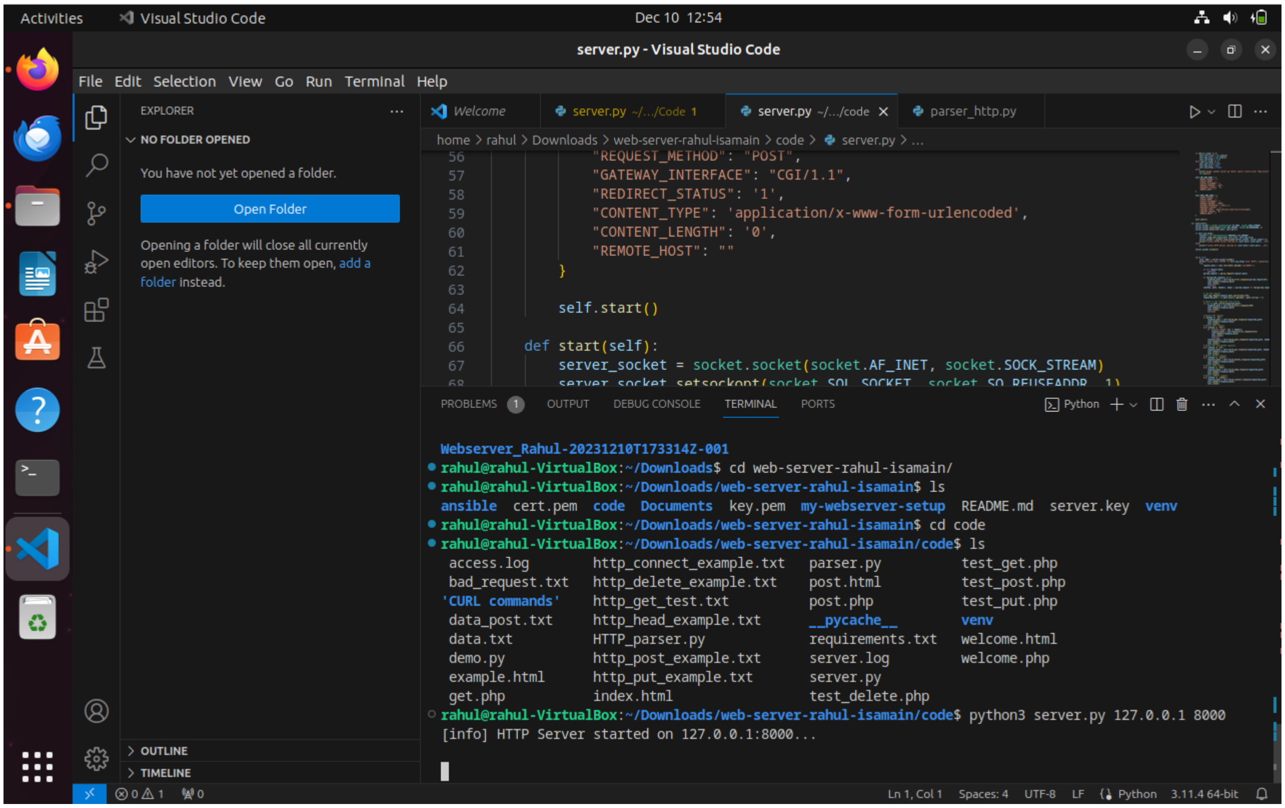


Activities
Terminal
Dec 10 01:40
rahul@rahul-VirtualBox: ~/Downloads/web-server-rahul-isamain

```
rahul@rahul-VirtualBox:~/Downloads/web-server-rahul-isamain$ sudo apt-get install openssl
[sudo] password for rahul:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
openssl is already the newest version (3.0.8-1ubuntu1.4).
0 upgraded, 0 newly installed, 0 to remove and 2 not upgraded.
rahul@rahul-VirtualBox:~/Downloads/web-server-rahul-isamain$ openssl genrsa -out server.key 2048
rahul@rahul-VirtualBox:~/Downloads/web-server-rahul-isamain$ openssl req -new -key server.key -out server.csr
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:US
State or Province Name (full name) [Some-State]:NY
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:rahulswebserver
An optional company name []:
rahul@rahul-VirtualBox:~/Downloads/web-server-rahul-isamain$ ls
```

Fig (c): Process of generating certificates and keys with the help of OpenSSL

- If nothing is sent as line argument, we need to assume it as HTTP request and process.


Activities
Visual Studio Code
Dec 10 12:54
server.py - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
NO FOLDER OPENED
Open Folder
OUTLINE
TIMELINE
Welcome
server.py ~/.../Code 1
server.py ~/.../code
parser_http.py
home > rahul > Downloads > web-server-rahul-isamain > code > server.py > ...
PROBLEMS 1
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
Python
Ln 1, Col 1 Spaces: 4 UTF-8 LF { } Python 3.11.4 64-bit

You have not yet opened a folder.

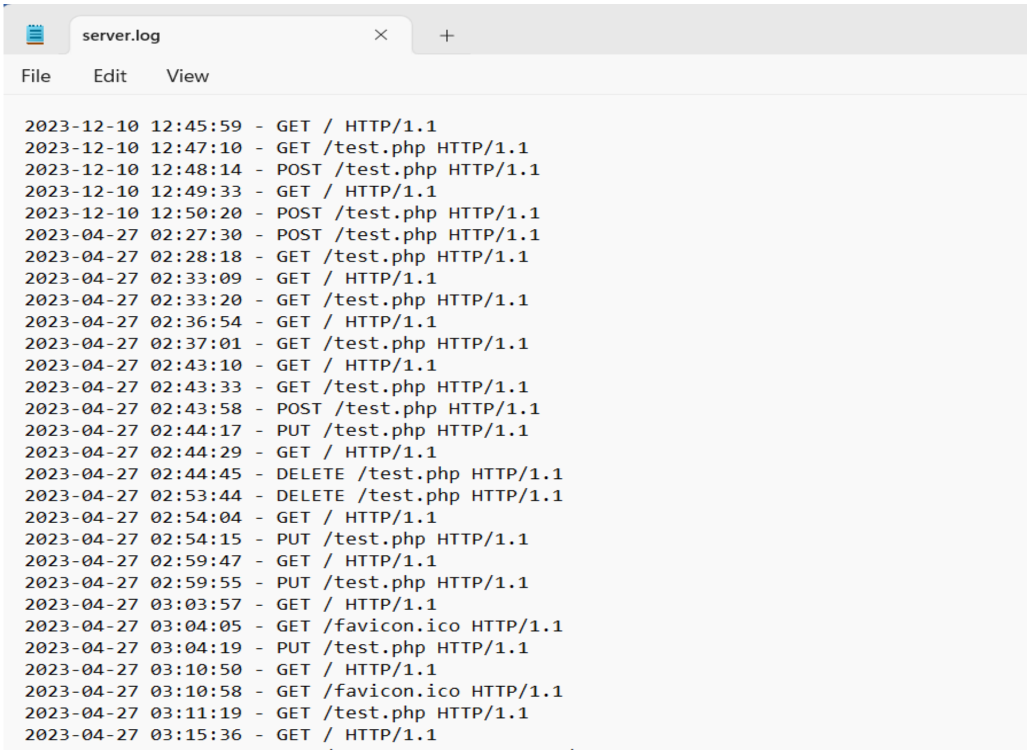
Opening a folder will close all currently open editors. To keep them open, add a folder instead.

```
                "REQUEST_METHOD": "POST",
                "GATEWAY_INTERFACE": "CGI/1.1",
                "REDIRECT_STATUS": '1',
                "CONTENT_TYPE": 'application/x-www-form-urlencoded',
                "CONTENT_LENGTH": '0',
                "REMOTE_HOST": ""
            }

            self.start()

        def start(self):
            server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
            server_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
```

```
Webserver_Rahul-20231210T173314Z-001
rahul@rahul-VirtualBox:~/Downloads$ cd web-server-rahul-isamain/
rahul@rahul-VirtualBox:~/Downloads/web-server-rahul-isamain$ ls
ansible  cert.pem  code  Documents  key.pem  my-webserver-setup  README.md  server.key  venv
rahul@rahul-VirtualBox:~/Downloads/web-server-rahul-isamain$ cd code
rahul@rahul-VirtualBox:~/Downloads/web-server-rahul-isamain/code$ ls
 access.log         http_connect_example.txt   parser.py          test_get.php
 bad_request.txt    http_delete_example.txt    post.html          test_post.php
'CURL commands'     http_get_test.txt          post.php           test_put.php
 data_post.txt      http_head_example.txt      __pycache__        venv
 data.txt           HTTP_parser.py             requirements.txt   welcome.html
 demo.py            http_post_example.txt      server.log         welcome.php
 example.html       http_put_example.txt       server.py
 get.php            index.html                 test_delete.php
rahul@rahul-VirtualBox:~/Downloads/web-server-rahul-isamain/code$ python3 server.py 127.0.0.1 8000
[info] HTTP Server started on 127.0.0.1:8000...
```

Fig (d): Output of HTTP request on server

- We also need to log the requests made by the server into a log file named 'server.log'.



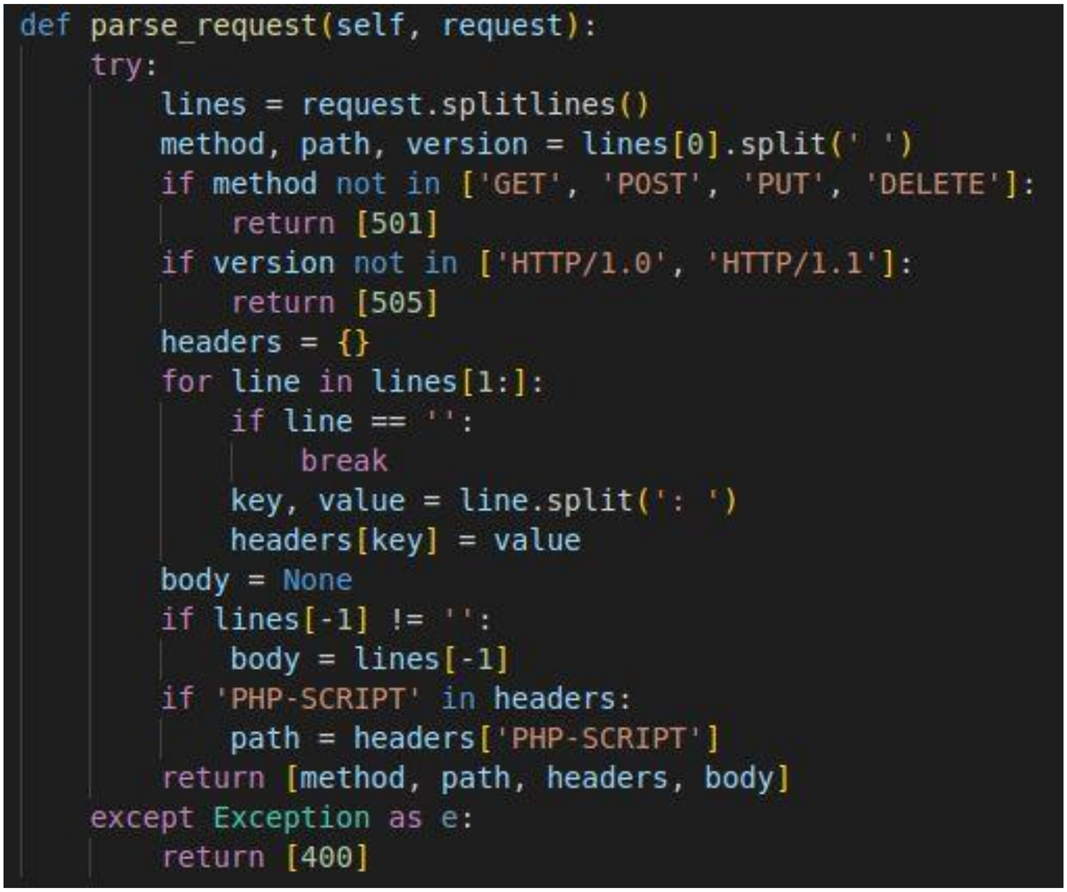
```
2023-12-10 12:45:59 - GET / HTTP/1.1
2023-12-10 12:47:10 - GET /test.php HTTP/1.1
2023-12-10 12:48:14 - POST /test.php HTTP/1.1
2023-12-10 12:49:33 - GET / HTTP/1.1
2023-12-10 12:50:20 - POST /test.php HTTP/1.1
2023-04-27 02:27:30 - POST /test.php HTTP/1.1
2023-04-27 02:28:18 - GET /test.php HTTP/1.1
2023-04-27 02:33:09 - GET / HTTP/1.1
2023-04-27 02:33:20 - GET /test.php HTTP/1.1
2023-04-27 02:36:54 - GET / HTTP/1.1
2023-04-27 02:37:01 - GET /test.php HTTP/1.1
2023-04-27 02:43:10 - GET / HTTP/1.1
2023-04-27 02:43:33 - GET /test.php HTTP/1.1
2023-04-27 02:43:58 - POST /test.php HTTP/1.1
2023-04-27 02:44:17 - PUT /test.php HTTP/1.1
2023-04-27 02:44:29 - GET / HTTP/1.1
2023-04-27 02:44:45 - DELETE /test.php HTTP/1.1
2023-04-27 02:53:44 - DELETE /test.php HTTP/1.1
2023-04-27 02:54:04 - GET / HTTP/1.1
2023-04-27 02:54:15 - PUT /test.php HTTP/1.1
2023-04-27 02:59:47 - GET / HTTP/1.1
2023-04-27 02:59:55 - PUT /test.php HTTP/1.1
2023-04-27 03:03:57 - GET / HTTP/1.1
2023-04-27 03:04:05 - GET /favicon.ico HTTP/1.1
2023-04-27 03:04:19 - PUT /test.php HTTP/1.1
2023-04-27 03:10:50 - GET / HTTP/1.1
2023-04-27 03:10:58 - GET /favicon.ico HTTP/1.1
2023-04-27 03:11:19 - GET /test.php HTTP/1.1
2023-04-27 03:15:36 - GET / HTTP/1.1
```

Fig (e): Server Log file

Adding Server Side Execution:

- In this part of the project, I added the capability of executing and processing PHP scripts to the HTTP Server.

- The server should have two line arguments as IP Address and Port Number to listen on.



```
def parse_request(self, request):
    try:
        lines = request.splitlines()
        method, path, version = lines[0].split(' ')
        if method not in ['GET', 'POST', 'PUT', 'DELETE']:
            return [501]
        if version not in ['HTTP/1.0', 'HTTP/1.1']:
            return [505]
        headers = {}
        for line in lines[1:]:
            if line == '':
                break
            key, value = line.split(': ')
            headers[key] = value
        body = None
        if lines[-1] != '':
            body = lines[-1]
        if 'PHP-SCRIPT' in headers:
            path = headers['PHP-SCRIPT']
        return [method, path, headers, body]
    except Exception as e:
        return [400]
```

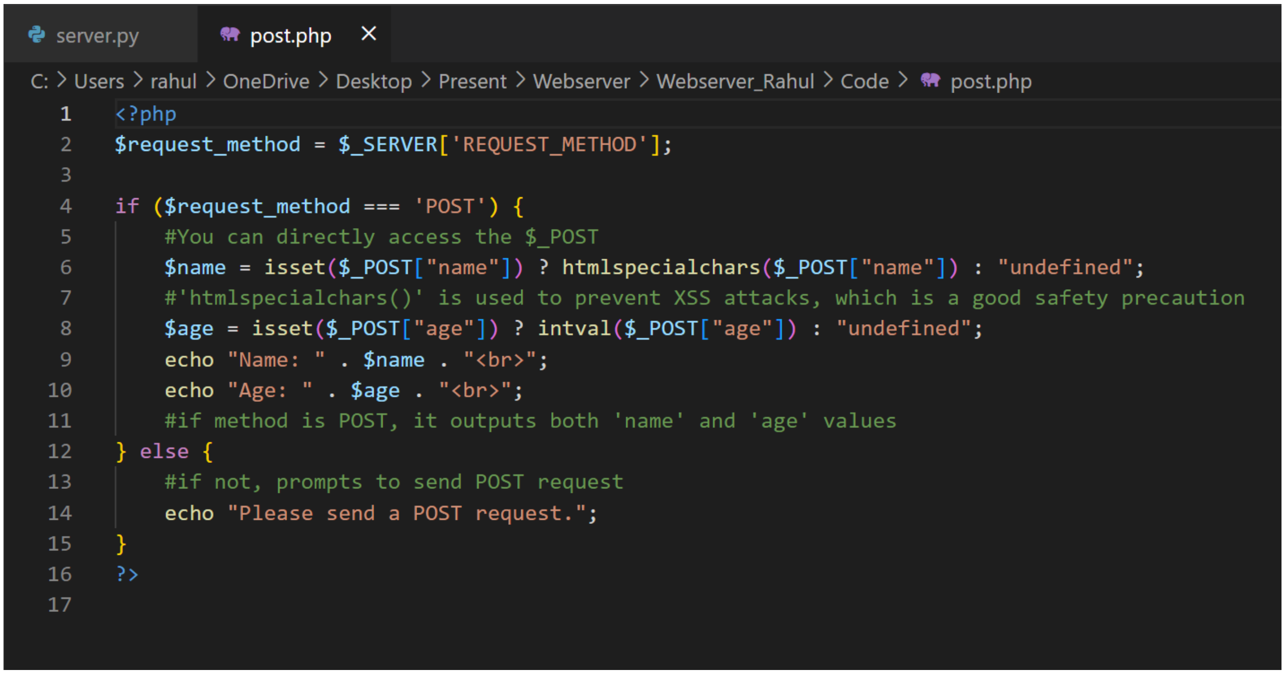
Fig (f): PHP part of server

- In the above figure, we can see that the server is checking if the script is PHP or not.

- Fig (h) shows how a response is created for PHP files like ‘POST.php’ as shown in the Fig (g).

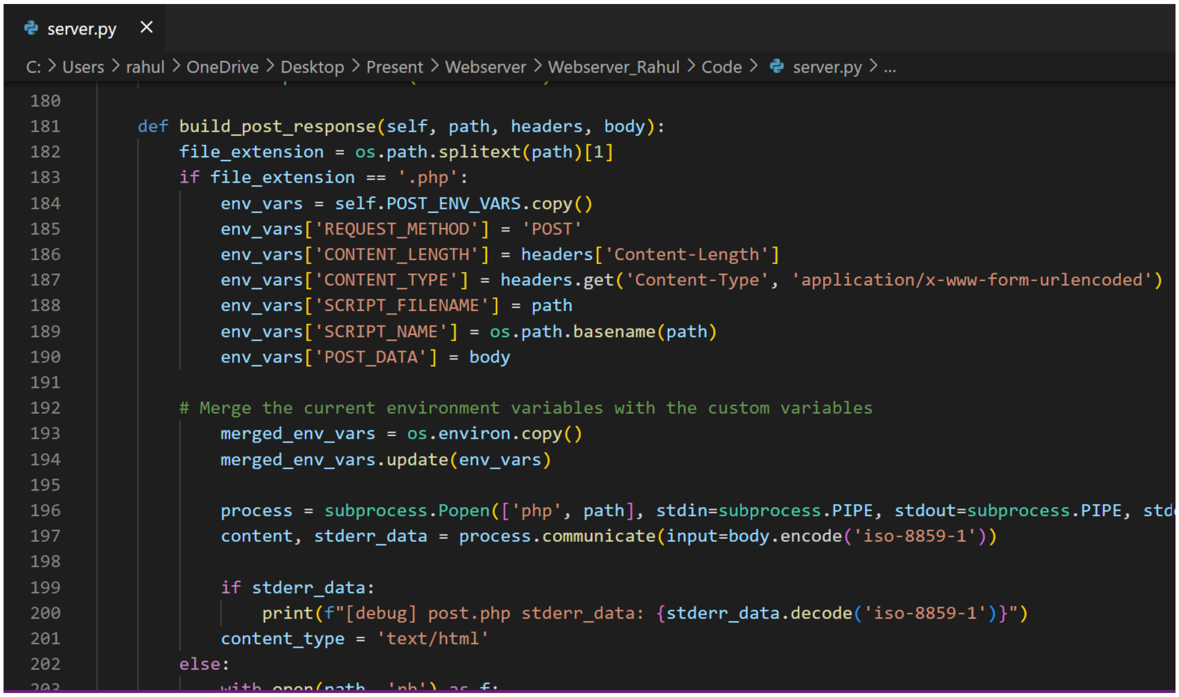
- As we can see in the Fig (g), if the method is POST, outputs both name and age values.

- If not, prompts to send POST requests.


server.py
post.php ×
C: > Users > rahul > OneDrive > Desktop > Present > Webserver > Webserver_Rahul > Code > post.php

```php
<?php
$request_method = $_SERVER['REQUEST_METHOD'];

if ($request_method === 'POST') {
    #You can directly access the $_POST
    $name = isset($_POST["name"]) ? htmlspecialchars($_POST["name"]) : "undefined";
    #'htmlspecialchars()' is used to prevent XSS attacks, which is a good safety precaution
    $age = isset($_POST["age"]) ? intval($_POST["age"]) : "undefined";
    echo "Name: " . $name . "<br>";
    echo "Age: " . $age . "<br>";
    #if method is POST, it outputs both 'name' and 'age' values
} else {
    #if not, prompts to send POST request
    echo "Please send a POST request.";
}
?>

```

Fig (g): Content of post.php


server.py ×
C: > Users > rahul > OneDrive > Desktop > Present > Webserver > Webserver_Rahul > Code > server.py > ...

```python

    def build_post_response(self, path, headers, body):
        file_extension = os.path.splitext(path)[1]
        if file_extension == '.php':
            env_vars = self.POST_ENV_VARS.copy()
            env_vars['REQUEST_METHOD'] = 'POST'
            env_vars['CONTENT_LENGTH'] = headers['Content-Length']
            env_vars['CONTENT_TYPE'] = headers.get('Content-Type', 'application/x-www-form-urlencoded')
            env_vars['SCRIPT_FILENAME'] = path
            env_vars['SCRIPT_NAME'] = os.path.basename(path)
            env_vars['POST_DATA'] = body

        # Merge the current environment variables with the custom variables
            merged_env_vars = os.environ.copy()
            merged_env_vars.update(env_vars)

            process = subprocess.Popen(['php', path], stdin=subprocess.PIPE, stdout=subprocess.PIPE, std
            content, stderr_data = process.communicate(input=body.encode('iso-8859-1'))

            if stderr_data:
                print(f"[debug] post.php stderr_data: {stderr_data.decode('iso-8859-1')}")
            content_type = 'text/html'
        else:
            with open(path, 'rb') as f:
```

Fig (h): Post response function of the server

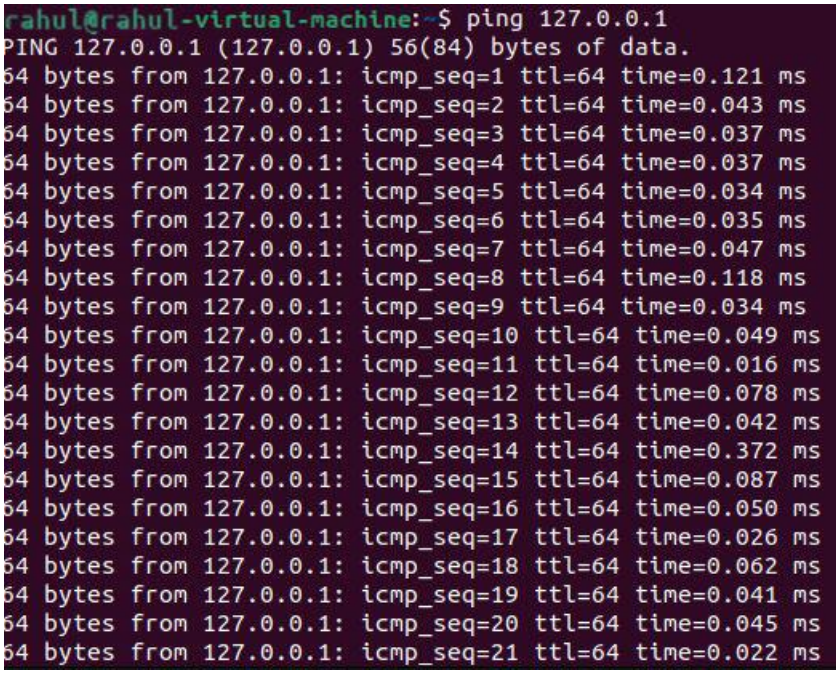
Attacking the Server:

- I used many types of attacks targeting the server, I made a detailed report of how these attacks work and level of damage it does to the server etc., in 'vulnerabiity.md' file in Github

- In this report, I am going to explain how a single attack of DDoS works targeting the server.

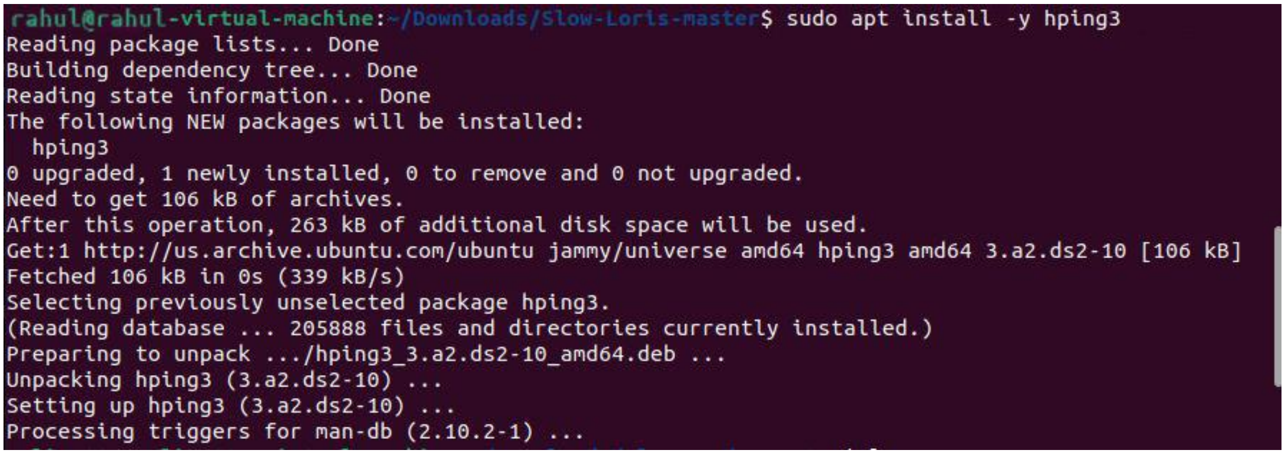
- For this, I used 'hping3' tool to create some customizable packets of any type such as TCP, UDP, ICMP etc.

- Before starting DDoS, make sure that the target is accessible from our node.



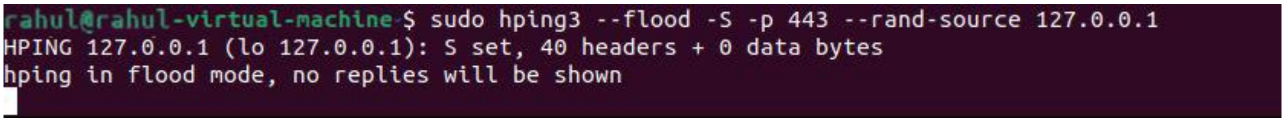
```
rahul@rahul-virtual-machine:~$ ping 127.0.0.1
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.121 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.043 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.037 ms
64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.037 ms
64 bytes from 127.0.0.1: icmp_seq=5 ttl=64 time=0.034 ms
64 bytes from 127.0.0.1: icmp_seq=6 ttl=64 time=0.035 ms
64 bytes from 127.0.0.1: icmp_seq=7 ttl=64 time=0.047 ms
64 bytes from 127.0.0.1: icmp_seq=8 ttl=64 time=0.118 ms
64 bytes from 127.0.0.1: icmp_seq=9 ttl=64 time=0.034 ms
64 bytes from 127.0.0.1: icmp_seq=10 ttl=64 time=0.049 ms
64 bytes from 127.0.0.1: icmp_seq=11 ttl=64 time=0.016 ms
64 bytes from 127.0.0.1: icmp_seq=12 ttl=64 time=0.078 ms
64 bytes from 127.0.0.1: icmp_seq=13 ttl=64 time=0.042 ms
64 bytes from 127.0.0.1: icmp_seq=14 ttl=64 time=0.372 ms
64 bytes from 127.0.0.1: icmp_seq=15 ttl=64 time=0.087 ms
64 bytes from 127.0.0.1: icmp_seq=16 ttl=64 time=0.050 ms
64 bytes from 127.0.0.1: icmp_seq=17 ttl=64 time=0.026 ms
64 bytes from 127.0.0.1: icmp_seq=18 ttl=64 time=0.062 ms
64 bytes from 127.0.0.1: icmp_seq=19 ttl=64 time=0.041 ms
64 bytes from 127.0.0.1: icmp_seq=20 ttl=64 time=0.045 ms
64 bytes from 127.0.0.1: icmp_seq=21 ttl=64 time=0.022 ms
```

Fig (i): Ping test before DDoS



```
rahul@rahul-virtual-machine:~/Downloads/Slow-Loris-master$ sudo apt install -y hping3
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  hping3
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 106 kB of archives.
After this operation, 263 kB of additional disk space will be used.
Get:1 http://us.archive.ubuntu.com/ubuntu jammy/universe amd64 hping3 amd64 3.a2.ds2-10 [106 kB]
Fetched 106 kB in 0s (339 kB/s)
Selecting previously unselected package hping3.
(Reading database ... 205888 files and directories currently installed.)
Preparing to unpack .../hping3_3.a2.ds2-10_amd64.deb ...
Unpacking hping3 (3.a2.ds2-10) ...
Setting up hping3 (3.a2.ds2-10) ...
Processing triggers for man-db (2.10.2-1) ...
```

Fig (i): Installation of hping3



```
rahul@rahul-virtual-machine $ sudo hping3 --flood -S -p 443 --rand-source 127.0.0.1
HPING 127.0.0.1 (lo 127.0.0.1): S set, 40 headers + 0 data bytes
hping in flood mode, no replies will be shown
```

Fig (j): Initiating DDoS attack using hping3

- In the above figure, we can see that we are initiating DDoS attack on '127.0.0.1'. We are targeting the port of 443 which is HTTPS port.

- The '-S' in the arguments mean that the hping3 is sending only 'SYN' packets.

Risk Assessment:

- Overall Threat: Malicious usage of the hping3 tool presents a serious risk to network security. Its ability to perform DoS attacks, map networks, and scan ports makes careful risk assessment and security measures necessary.

- The tool's accessibility and ease of use for users with diverse technical skills make it tempting for harmful intents, hence increasing the likelihood of misuse.

- Impact Range: From small-scale hiccups to significant security breaches. While network mapping and port scanning can reveal weaknesses, increasing the risk of data breaches and unauthorized access, denial-of-service (DoS) assaults can result in financial loss and operational outages.

- Analysis of CVSS:

  - Base Score: 7.5, signifying a considerable degree of danger.

  - The vector components are as follows: AV:N (use remotely), AC:L (low complexity), PR:N (no privileges required), UI:N (no user interaction needed), S:U (unknown scope), I:N (no impact on confidentiality) and C:N (no impact on integrity) but A:H (high impact on availability).

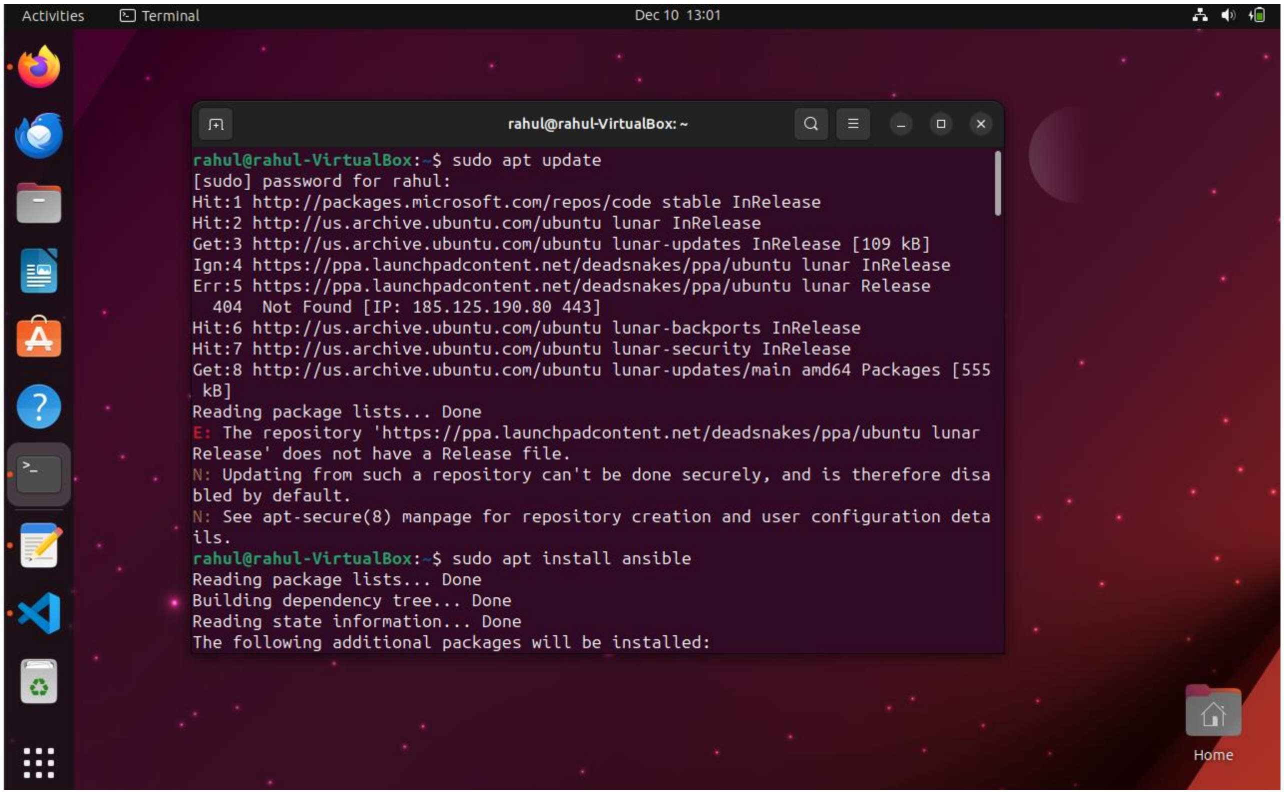
- Exploitable Weaknesses: Possibility of abusing CWEs such as URL Redirection to Untrusted Sites (CWE-601) and Uncontrolled Resource Consumption (CWE-400). Affects all ASVS levels, including implementation, design, and requirements.

- Explained remaining vulnerability's risk assessment in Github.

Deploying the Server:

- For deploying of server, I used Ansible.

- Ansible is open-source automation software used for managing configurations, deploying applications, and orchestrating intricate workflows in the IT industry.



```
rahul@rahul-VirtualBox:~$ sudo apt update
[sudo] password for rahul:
Hit:1 http://packages.microsoft.com/repos/code stable InRelease
Hit:2 http://us.archive.ubuntu.com/ubuntu lunar InRelease
Get:3 http://us.archive.ubuntu.com/ubuntu lunar-updates InRelease [109 kB]
Ign:4 https://ppa.launchpadcontent.net/deadsnakes/ppa/ubuntu lunar InRelease
Err:5 https://ppa.launchpadcontent.net/deadsnakes/ppa/ubuntu lunar Release
  404  Not Found [IP: 185.125.190.80 443]
Hit:6 http://us.archive.ubuntu.com/ubuntu lunar-backports InRelease
Hit:7 http://us.archive.ubuntu.com/ubuntu lunar-security InRelease
Get:8 http://us.archive.ubuntu.com/ubuntu lunar-updates/main amd64 Packages [555
 kB]
Reading package lists... Done
E: The repository 'https://ppa.launchpadcontent.net/deadsnakes/ppa/ubuntu lunar
Release' does not have a Release file.
N: Updating from such a repository can't be done securely, and is therefore disa
bled by default.
N: See apt-secure(8) manpage for repository creation and user configuration deta
ils.
rahul@rahul-VirtualBox:~$ sudo apt install ansible
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
```

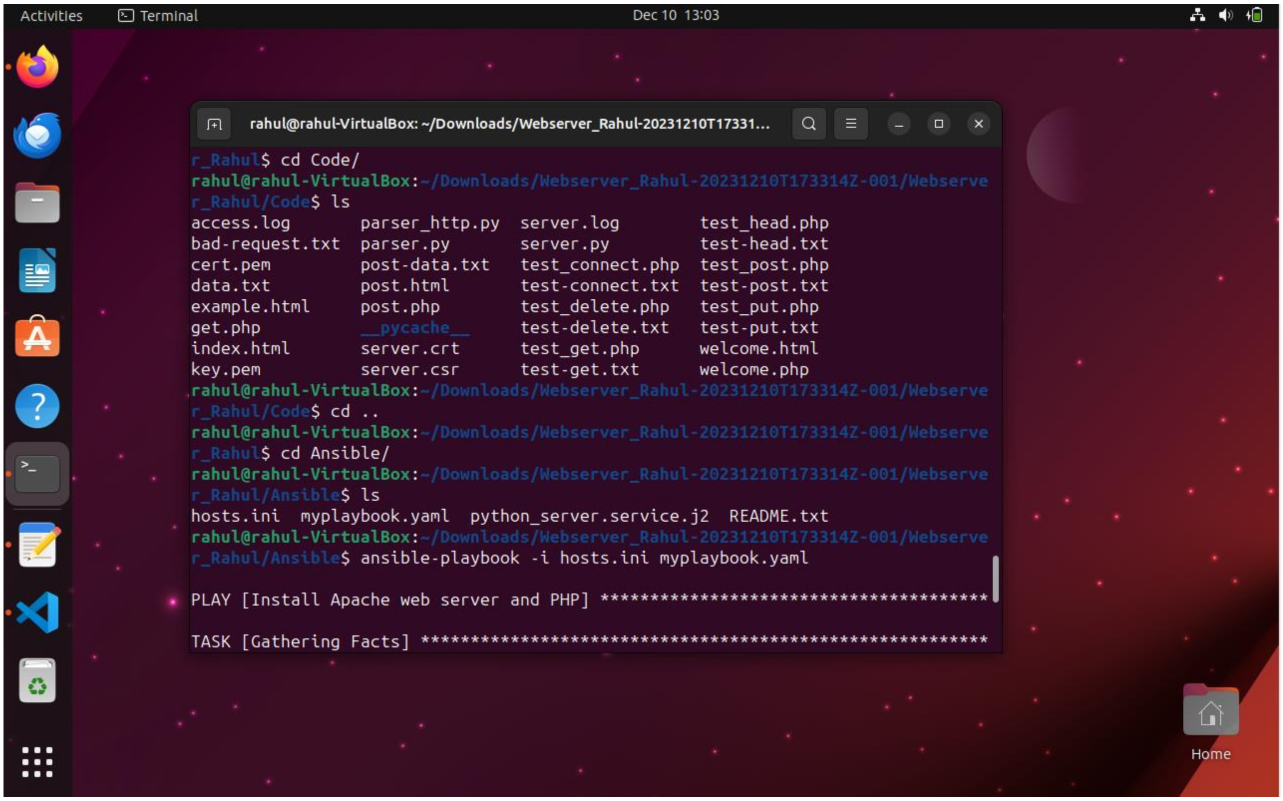
Fig (k): Installation of Ansible

- Configure files in the Ansible folder such as ‘hosts.ini’, ‘myplaybook.yaml’ to make sure that the Ansible is executable and configured properly.

- ‘hosts.ini’ is used to group different types of users involved in this process of deployment using Ansible.

- ‘myplaybook.yaml’ is a playbook which contains the tasks that the Ansible should do including installing Apache server, copy server.py script etc.

- Then execute the Ansible as shown in the Fig. below.

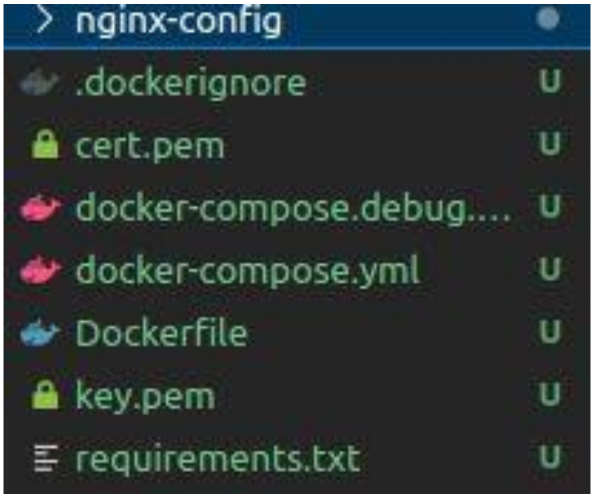

Activities
Terminal
Dec 10 13:03
rahul@rahul-VirtualBox: ~/Downloads/Webserver_Rahul-20231210T17331...
Home

```
r_Rahul$ cd Code/
rahul@rahul-VirtualBox:~/Downloads/Webserver_Rahul-20231210T173314Z-001/Webserve
r_Rahul/Code$ ls
access.log       parser_http.py  server.log        test_head.php
bad-request.txt  parser.py       server.py         test-head.txt
cert.pem         post-data.txt   test_connect.php  test_post.php
data.txt         post.html       test-connect.txt  test-post.txt
example.html     post.php        test_delete.php   test_put.php
get.php          __pycache__     test-delete.txt   test-put.txt
index.html       server.crt      test_get.php      welcome.html
key.pem          server.csr      test-get.txt      welcome.php
rahul@rahul-VirtualBox:~/Downloads/Webserver_Rahul-20231210T173314Z-001/Webserve
r_Rahul/Code$ cd ..
rahul@rahul-VirtualBox:~/Downloads/Webserver_Rahul-20231210T173314Z-001/Webserve
r_Rahul$ cd Ansible/
rahul@rahul-VirtualBox:~/Downloads/Webserver_Rahul-20231210T173314Z-001/Webserve
r_Rahul/Ansible$ ls
hosts.ini  myplaybook.yaml  python_server.service.j2  README.txt
rahul@rahul-VirtualBox:~/Downloads/Webserver_Rahul-20231210T173314Z-001/Webserve
r_Rahul/Ansible$ ansible-playbook -i hosts.ini myplaybook.yaml

PLAY [Install Apache web server and PHP] ***************************************

TASK [Gathering Facts] *********************************************************
```

Fig (l): Executing Ansible deployment

Dockerized Infrastructure:

- In this part of the project, we deploy the whole web infrastructure on docker containers and use container security mechanisms to provide security for the infrastructure.

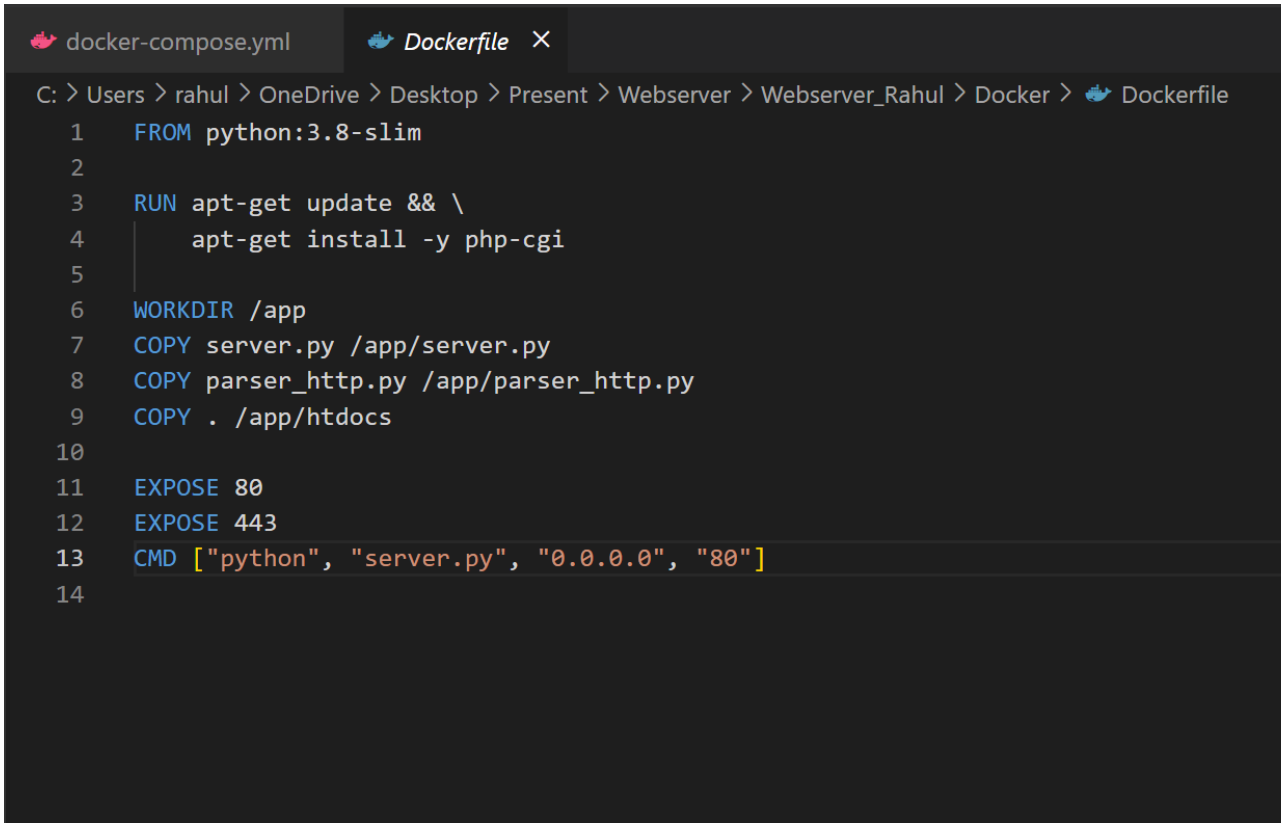


```
> nginx-config
  .dockerignore            U
  cert.pem                 U
  docker-compose.debug.... U
  docker-compose.yml       U
  Dockerfile               U
  key.pem                  U
  requirements.txt         U
```

Fig (m): Structure of Docker

- Out of all docker files, there are some important docker files such as Dockerfile, docker-compose.yml etc.

- The most basic file found in a Docker setup. All the commands an individual could use to put together an image on the command line are contained in a text document called a Dockerfile.

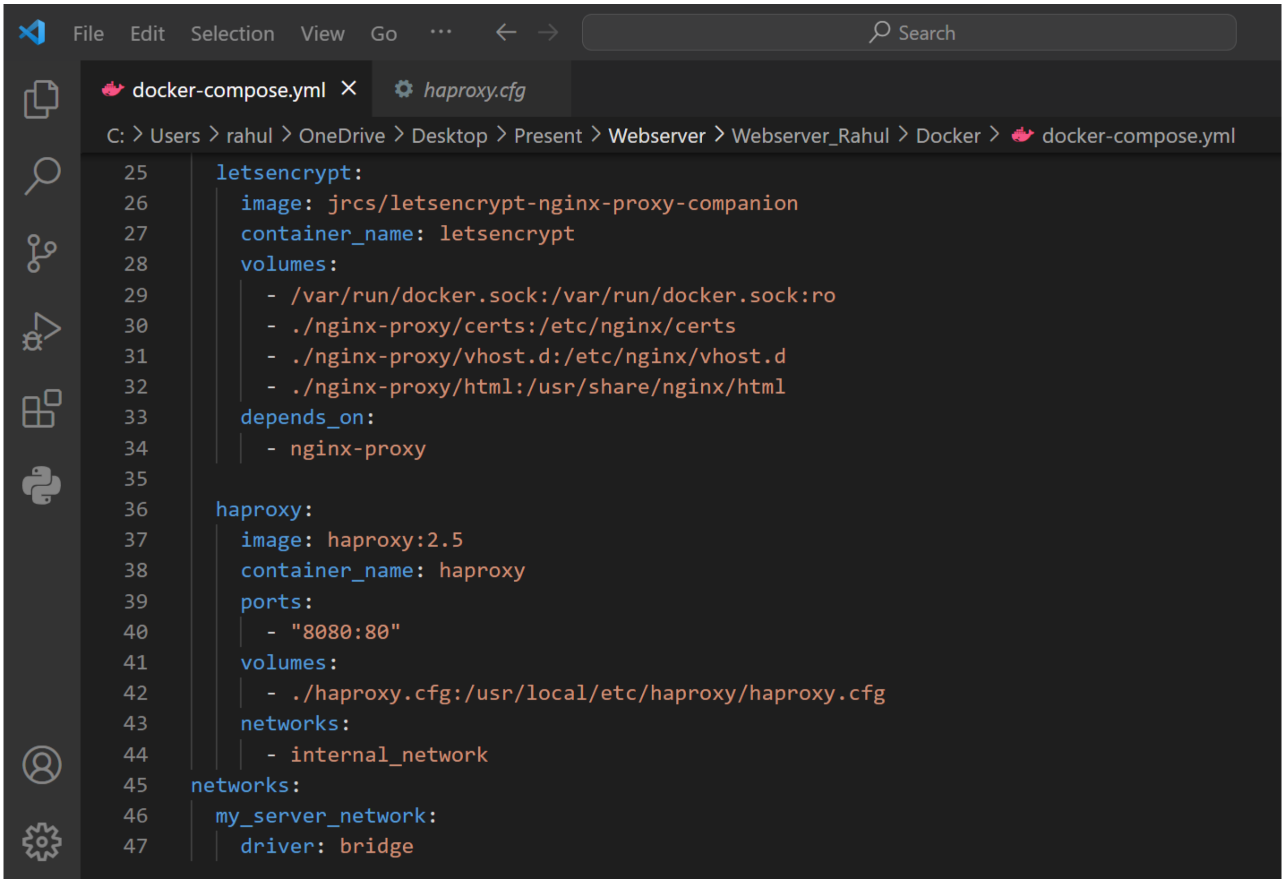


docker-compose.yml
Dockerfile ×
C: > Users > rahul > OneDrive > Desktop > Present > Webserver > Webserver_Rahul > Docker > Dockerfile

```
FROM python:3.8-slim

RUN apt-get update && \
    apt-get install -y php-cgi

WORKDIR /app
COPY server.py /app/server.py
COPY parser_http.py /app/parser_http.py
COPY . /app/htdocs

EXPOSE 80
EXPOSE 443
CMD ["python", "server.py", "0.0.0.0", "80"]

```

Fig (n): Contents of Dockerfile

- This YAML file defines volumes, networks, and services for Docker containers and is used when dealing with Docker Compose. It lets you run several containers as a single service and configure the services of your application.


File
Edit
Selection
View
Go
···
Search
docker-compose.yml
haproxy.cfg
C: > Users > rahul > OneDrive > Desktop > Present > Webserver > Webserver_Rahul > Docker > docker-compose.yml

```
  letsencrypt:
    image: jrcs/letsencrypt-nginx-proxy-companion
    container_name: letsencrypt
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock:ro
      - ./nginx-proxy/certs:/etc/nginx/certs
      - ./nginx-proxy/vhost.d:/etc/nginx/vhost.d
      - ./nginx-proxy/html:/usr/share/nginx/html
    depends_on:
      - nginx-proxy

  haproxy:
    image: haproxy:2.5
    container_name: haproxy
    ports:
      - "8080:80"
    volumes:
      - ./haproxy.cfg:/usr/local/etc/haproxy/haproxy.cfg
    networks:
      - internal_network
networks:
  my_server_network:
    driver: bridge
```

Fig (o): Contents of ‘docker-compose.yml’