SUNWAY UNIVERSITY
SCHOOL OF ENGINEERING AND TECHNOLOGY

Subject Code and Title: NET2103 NETWORK & SYSTEM ADMINISTRATION

Semester: August 2022

FINAL REPORT DEADLINE: 16/7/2023

Lecturer Name:	Prof. Ts. Yap Kian Meng
Tutor Name:	Ms. Teoh Jiehan
Group Number:	13

No.	Student Full Name	Student ID	Program Code	Signature
1	Aneeysa Binti Reduan	21086525	481BCNS	Aneeysa
2	Mou Ryan	21085535	481BCNS	Ryan
3	Yashnni A/P Sugumar	20061156	481BCNS	Yashnni
4	Dharshaan A/L Segaran	20029997	481BCNS	Dharshaan
5	Liivenesh A/L Sundara Raju	20061586	481BCNS	Liivenesh

We hereby swear that the work done on this assignment is our own and we have not given nor received aid that is inappropriate for this assignment. We understand that by the school code, violation of these principles will lead to a zero mark on this assignment.

TABLE OF CONTENT

1.0 WEB SERVER

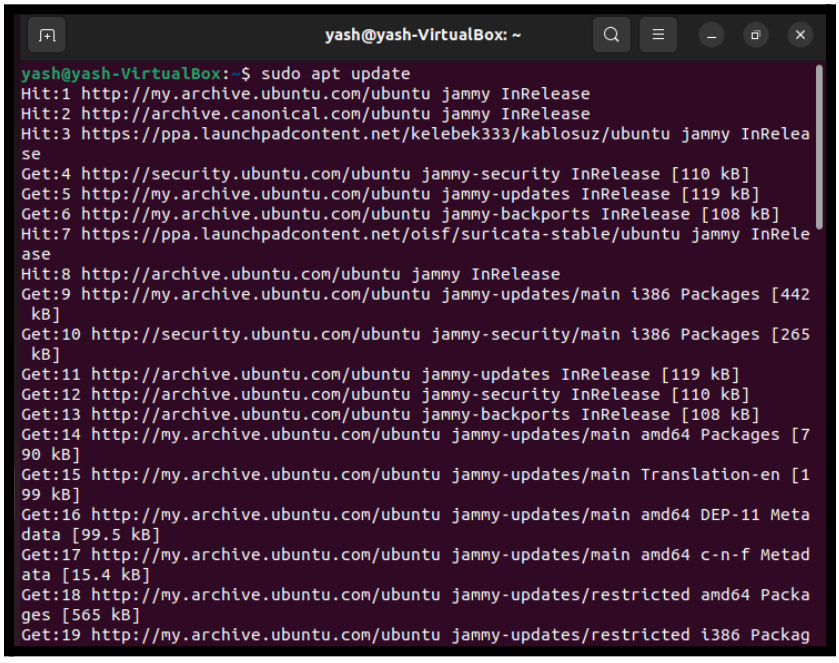
1.1 OVERVIEW

A web server is a software application that hosts websites and web pages for users existing in the same network. When users access a specific website through its unique URL, the web server fetches the request and retrieves the requested web content from its storage in order to deliver it to them on their web browser. Once the browsers receive the package, it will render the web page for the users by interpreting the information obtained such as HTML, CSS, and JavaScript code from the web server's response, thereby enabling users to view and interact with the website's features.

1.2 IMPLEMENTATION DETAILS

1.2.1 REQUIREMENT 1: INSTALLATION

1. **Update Ubuntu: *sudo apt update***



Command ***"sudo"*** is usually run in conjunction with other commands to elevate user privileges and ***"apt"*** is a package management command-line tool used to handle software package operations. In combination with ***"update"***, this line refreshes the local

package lists on our system by retrieving the latest information regarding our available software packages from the configured repositories. Running this command helps ensure our system is working with the most up-to-date information on available packages prior to enacting any installations or upgrades.

2. Upgrade Ubuntu: *sudo apt upgrade -y*

```
yash@yash-VirtualBox:~$ sudo apt upgrade -y
[sudo] password for yash:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Calculating upgrade... Done
#
# An OpenSSL vulnerability has recently been fixed with USN-6188-1 & 6119-1:
# CVE-2023-2650: possible DoS translating ASN.1 object identifiers.
# Ensure you have updated the package to its latest version.
#
The following packages have been kept back:
  libspeechd2 python3-speechd speech-dispatcher
  speech-dispatcher-audio-plugins speech-dispatcher-espeak-ng
0 upgraded, 0 newly installed, 0 to remove and 5 not upgraded.
```

Upgrade the installed packages on our system to their latest version. It is a follow-up command from the last as it considers the updated package list that was obtained before. By utilising the ***"-y"*** option, the upgrade process is automated. Unlike the previous command which only serves as a research step, this command actually downloads the newest version of the package so that our web server has all the latest bug fixes, security patches, and performance enhancement.

3. **Install Apache web server: *sudo apt install apache2 -y***

```
yash@yash-VirtualBox:~$ sudo apt-get install apache2 -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
apache2 is already the newest version (2.4.52-1ubuntu4.5).
0 upgraded, 0 newly installed, 0 to remove and 5 not upgraded.
```

Install Apache web server on Ubuntu Linux operating system. The ***"-y"*** option here enables a non-interactive installation by getting the system to instinctively respond "yes" to any prompts or confirmation during the process. This installation provides the

necessary infrastructure for the system to host websites, handle HTTP requests, and consequently serve web pages to its clients.

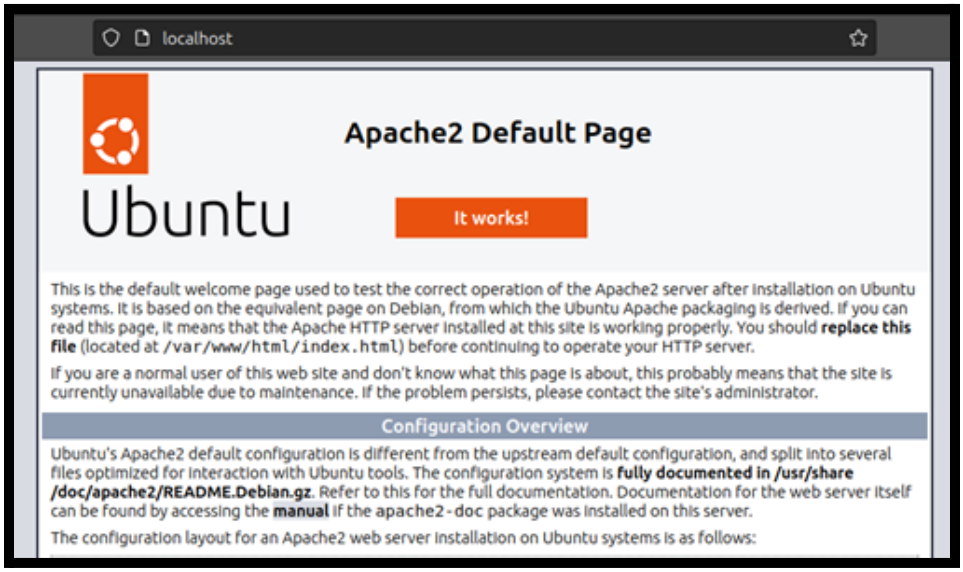
4. Check Apache version*: *apache2 -version

```
yash@yash-VirtualBox:~$ apache2 -version
Server version: Apache/2.4.52 (Ubuntu)
Server built:   2023-03-01T22:43:55
yash@yash-VirtualBox:~$
```

To check the version of our installed Apache web server. This information can lead to extra details about its release which can be a helpful guide for us when troubleshooting any issues we might encounter during our configuration phase.

Test View: *Localhost*

You can check if your Apache web server is working by searching for ***'localhost'*** in your web browser. If it is successfully installed, Apache's default page will appear like below:



1.2.2 REQUIREMENT 2: VIRTUAL HOSTING

1.* Switch directory: *cd /var/www/html

```
yash@yash-VirtualBox:~$ cd /var/www/html
yash@yash-VirtualBox:/var/www/html$
```

"cd" changes the current working directory of the user to another. In this case, we are switching our directory to ***/var/www/html*** where we will be keeping all of our website files.

2. **Create website 1's directory: *sudo mkdir page1.hello.com***

```
yash@yash-VirtualBox:/var/www/html$ sudo mkdir page1.hello.com
```

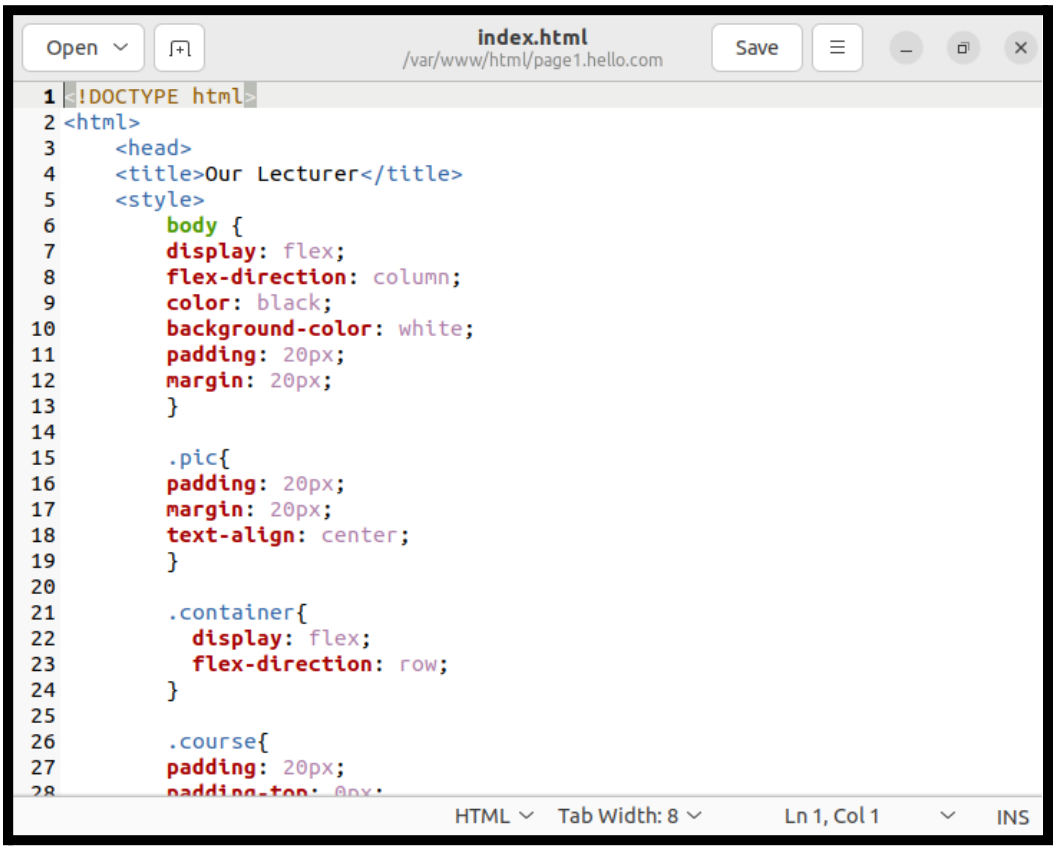
3. **Create website 2's directory: *sudo mkdir page1.bye.com***

```
yash@yash-VirtualBox:/var/www/html$ sudo mkdir page1.bye.com
```

4. **Go into website 1's directory: *cd page1.hello.com***

```
yash@yash-VirtualBox:/var/www/html$ cd page1.hello.com
yash@yash-VirtualBox:/var/www/html/page1.hello.com$ sudo gedit index.html
```

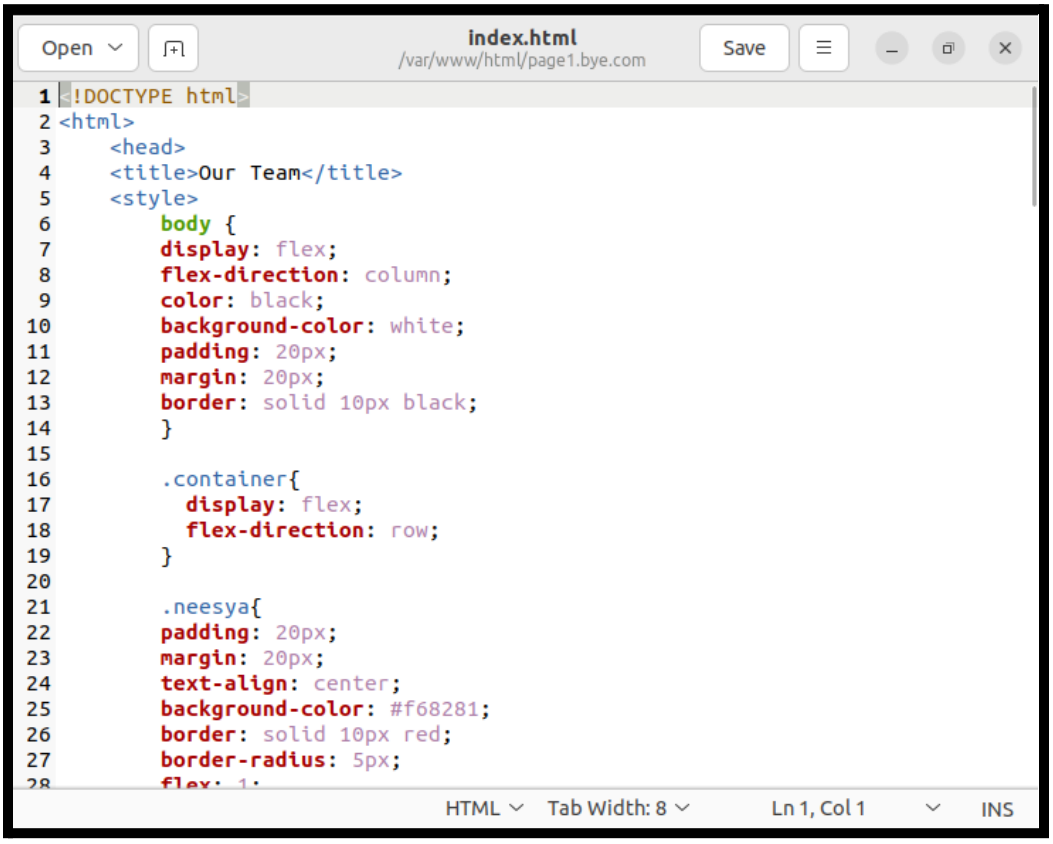
5. **Create and modify website 1's index file: *sudo gedit index.html***



6. **Go into website 1's directory: *cd page1.bye.com***

```
yash@yash-VirtualBox:/var/www/html/page1.hello.com$ cd ..
yash@yash-VirtualBox:/var/www/html$ cd page1.bye.com
yash@yash-VirtualBox:/var/www/html/page1.bye.com$ sudo gedit index.html
```

7. Create and modify website 2's index file: *sudo gedit index.html*

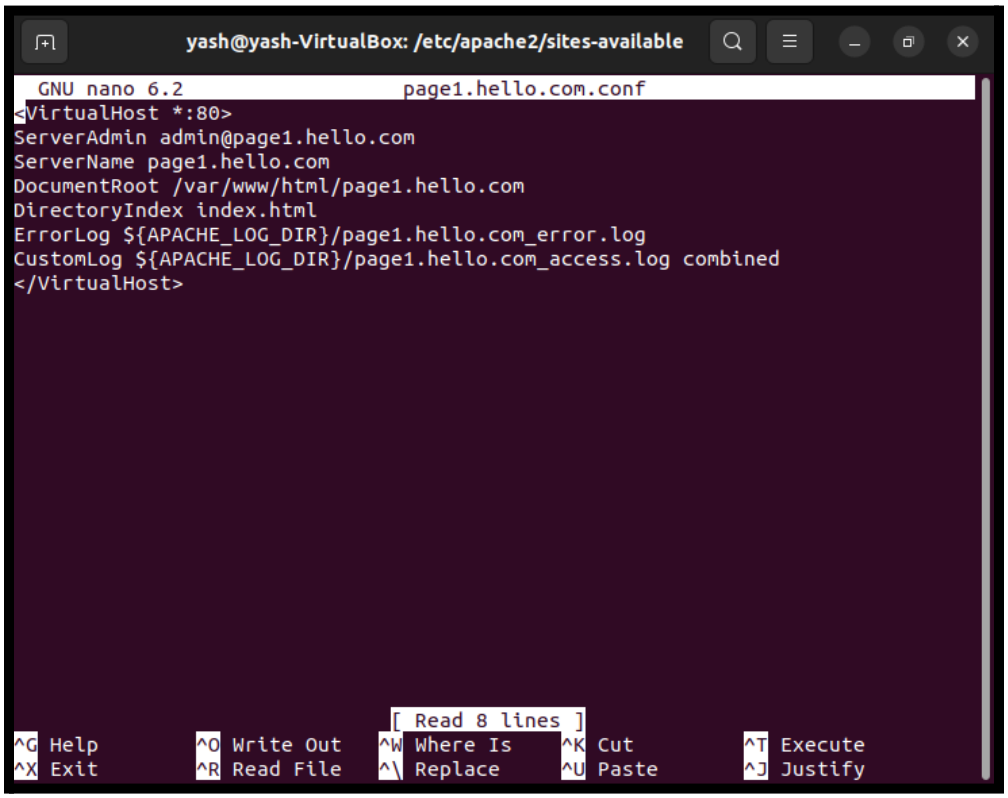


Now in our desired directory, ***"mkdir"***, a command used to create a new directory is applied twice to create an individual directory for both our websites' related files such as their respective index file that holds all of their HTML and CSS code alongside images that appear in them. To create and modify the mentioned index files of each website, the command ***"gedit"*** is used on two separate occasions, first in site 1's directory and second in site 2's directory so as to not confuse the server when fetching a particular web content for clients.

8. **Create and edit website 1's virtual host configuration files:**

sudo nano /etc/apache2/sites-available/page1.hello.com.conf

```
yash@yash-VirtualBox:~$ sudo nano /etc/apache2/sites-available/page1.hello.com.
conf
```

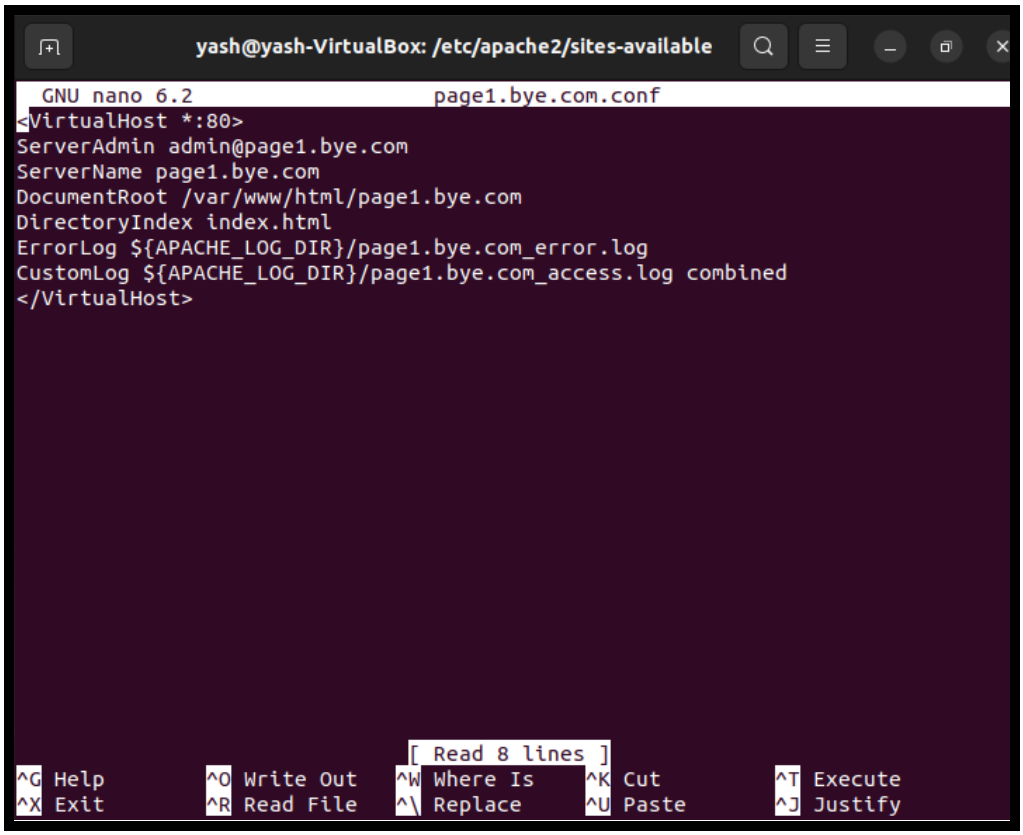


9. **Create and edit website 2's virtual host configuration files:**

sudo nano /etc/apache2/sites-available/page1.bye.com.conf

```
yash@yash-VirtualBox:~$ sudo nano /etc/apache2/sites-available/page1.bye.com.co
nf
```

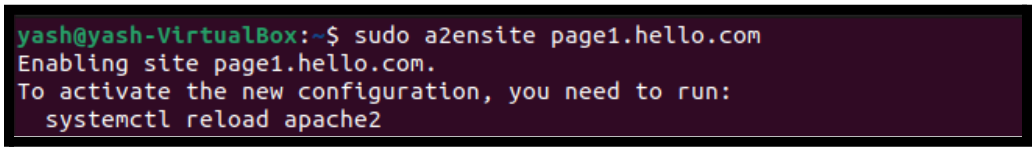
Under directory ***/etc/apache2/sites-available***, we have created two configuration files using ***"nano"*** for each of our websites that define their virtual host settings, the files are named according to the websites' domain name followed by .conf at the end.



The lines of code above are written into the mentioned configuration files with slight adjustments to match the website's distinctive features. With the exception of the starting and ending lines that correspond to the start and the end of the virtual host configuration block, in sequence the line of codes explicitly set the server administrator's email address, associated domain name, website's files directory, said directory's default page, the file path of the virtual host's error and access logs.

10. Activate website 1's virtual host configuration file:

sudo a2ensite site1.example.com



11. **Activate website 2's virtual host configuration file:**

sudo a2ensite site2.example.com

```
yash@yash-VirtualBox:~$ sudo a2ensite page1.bye.com
Enabling site page1.bye.com.
To activate the new configuration, you need to run:
  systemctl reload apache2
```

The command ***"a2ensite"*** activates the websites' virtual host configuration files created earlier. Once active, Apache will recognize and serve those websites based on their designated domain name by creating a symbolic link between the configuration files in ***/etc/apache/sites-available*** to the corresponding files in ***/etc/apache/sites-enabled***.

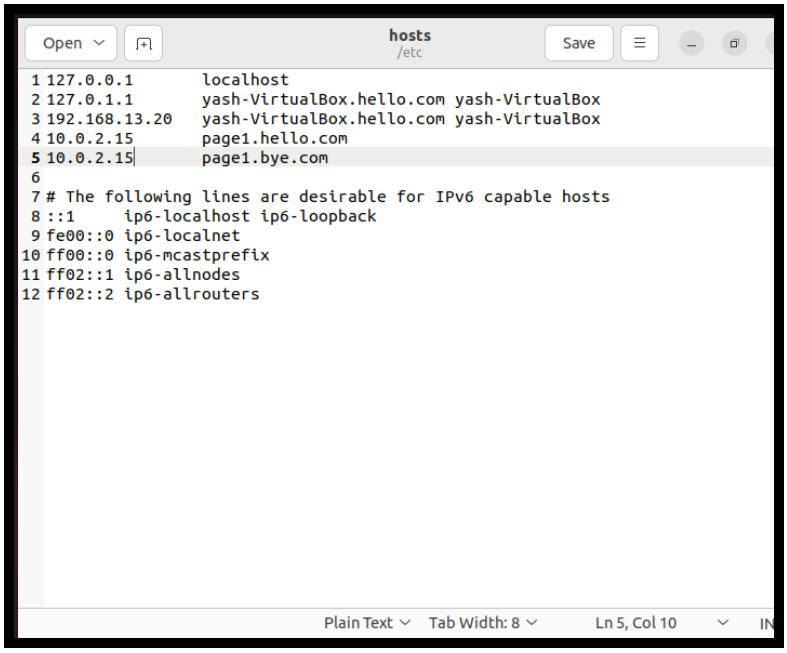
12. **Identify IP address:** ***ifconfig***

```
yash@yash-VirtualBox:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.15  netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fe80::7560:d54b:4996:fb8  prefixlen 64  scopeid 0x20<link>
        ether de:a5:46:eb:31:e0  txqueuelen 1000  (Ethernet)
        RX packets 2475  bytes 2415365 (2.4 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1419  bytes 152263 (152.2 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 19  base 0xd020

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 133  bytes 634091 (634.0 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 133  bytes 634091 (634.0 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

yash@yash-VirtualBox:~$
```

13. **Append web server's IP address and websites' domain name to hosts:**

sudo gedit /etc/hosts

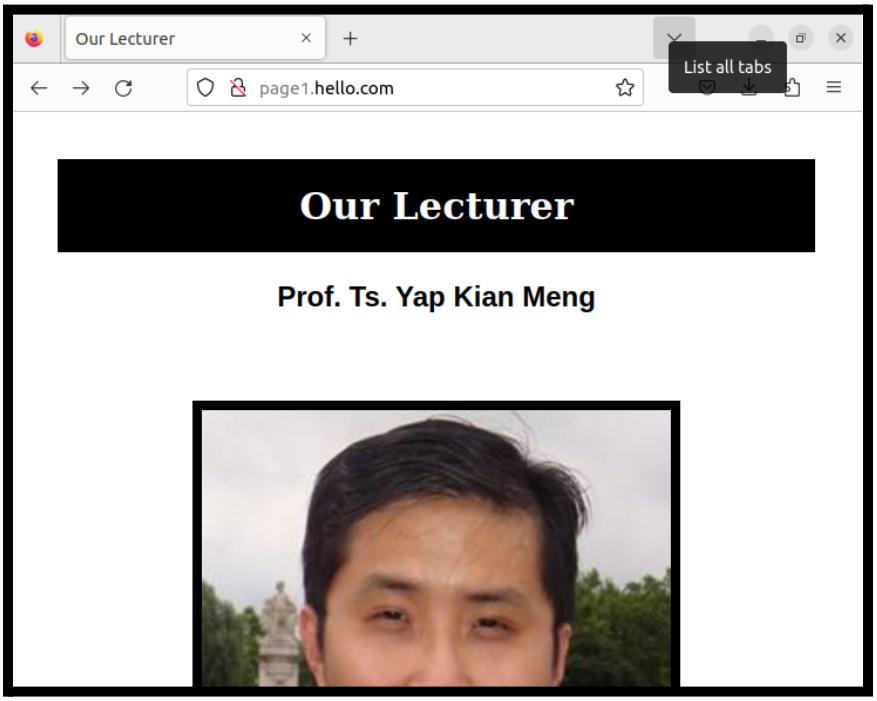
```
yash@yash-VirtualBox:~$ sudo gedit /etc/hosts

(gedit:11038): dconf-WARNING **: 09:24:56.837: failed to commit changes to dcon
f: Failed to execute child process "dbus-launch" (No such file or directory)

(gedit:11038): dconf-WARNING **: 09:24:56.881: failed to commit changes to dcon
f: Failed to execute child process "dbus-launch" (No such file or directory)
```



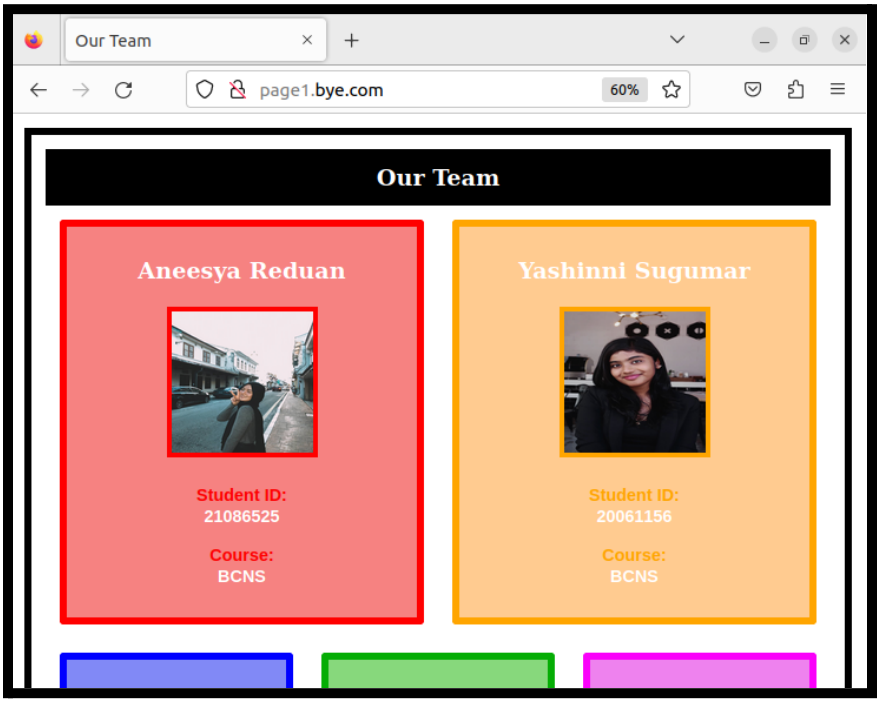
"ifconfig" as a command helps the administrator identify and verify the server's network interface configuration while ***"gedit"*** in this situation edits the ***/etc/hosts*** file. By appending our websites' domain name and server's IP address in ***/etc/hosts***, we are able to map them to each other, enabling us to test said websites before DNS resolution is fully functional.

For testing purposes, set the website's IP address to 10.0.2.15 (Virtual Machine's IP).

Test View: ***page1.hello.com***



Test View: ***page1.bye.com***



1.2.3 REQUIREMENT 3: SIMULTANEOUS USERS

1. **Modify Apache's configuration file:** ***sudo gedit /etc/apache2/apache2.conf***

```
yash@yash-VirtualBox:/etc/apache2/sites-available$ sudo gedit /etc/apache2/apac
he2.conf
```

2. *MaxKeepAliveRequests 2*

```
 92 Timeout 300
 93
 94 #
 95 # KeepAlive: Whether or not to allow persistent connections (more than
 96 # one request per connection). Set to "Off" to deactivate.
 97 #
 98 KeepAlive On
 99
100 #
101 # MaxKeepAliveRequests: The maximum number of requests to allow
102 # during a persistent connection. Set to 0 to allow an unlimited amount.
103 # We recommend you leave this number high, for maximum performance.
104 #
105 MaxKeepAliveRequests 2
106
107 #
108 # KeepAliveTimeout: Number of seconds to wait for the next request from the
109 # same client on the same connection.
110 #
111 KeepAliveTimeout 5
112
113
114 # These need to be set in /etc/apache2/envvars
115 User ${APACHE_RUN_USER}
116 Group ${APACHE_RUN_GROUP}
```

In the same way we have configured all our files thus far, ***"gedit"*** is once again used to access Apache's configuration file and modify its ***MaxKeepAliveRequest*** to 2, which will restrict the number of requests that the server will serve to two over the course of a single persistent connection before closing said connection.

1.2.4 REQUIREMENT 4: USER AUTHENTICATION

1. Install Apache's configuration tools: *sudo apt install apache2-utils*

```
yash@yash-VirtualBox:/etc/apache2/sites-available$ cd
yash@yash-VirtualBox:~$ sudo apt install apache2-utils
[sudo] password for yash:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
apache2-utils is already the newest version (2.4.52-1ubuntu4.5).
0 upgraded, 0 newly installed, 0 to remove and 5 not upgraded.
yash@yash-VirtualBox:~$
```

A command used to install utilities and tools meant for configuring and managing Apache HTTP Server to a greater extent.

2. Create .htpasswd file: *sudo htpasswd -c /etc/apache2/.htpasswd yash*

```
yash@yash-VirtualBox:~$ sudo htpasswd -c /etc/apache2/.htpasswd yash
New password:
Re-type new password:
Adding password for user yash
yash@yash-VirtualBox:~$
```

This command creates a new file, indicated by the ***"-c"*** flag called ***".htpasswd"*** under the ***/etc/apache2*** directory that contains the username ***"yash"***s encrypted password requested in the subsequent line. The file and its content are the key to our website's basic authentication that is protecting certain elements from the public.

3. Modify Apache's configuration: ***sudo gedit /etc/apache2/apache2.conf***

```
yash@yash-VirtualBox:~$ sudo gedit /etc/apache2/apache2.conf
```

```
159 <Directory />
160     Options FollowSymLinks
161     AllowOverride None
162     Require all denied
163 </Directory>
164
165 <Directory /usr/share>
166     AllowOverride None
167     Require all granted
168 </Directory>
169
170 <Directory /var/www/>
171     Options Indexes FollowSymLinks
172     AllowOverride All
173     Require all granted
174 </Directory>
175
176 #<Directory /srv/>
177 #   Options Indexes FollowSymLinks
178 #   AllowOverride None
179 #   Require all granted
180 #</Directory>
```

AllowOverride All

Return to the Apache configuration file once again to update the ***AllowOverride*** directive to ***"All"***. The directive controls the use of ***".htaccess"*** files in Apache's configuration and setting it to All enables said files to override distinct directive there. These affected directives also include ones that enforce our user authentication for specific directory and files, therefore making this minor change much more significant as a whole.

4. Switch directory: ***cd /var/www/html/site1.example.com***

Create a new directory: ***mkdir images***

```
yash@yash-VirtualBox:~$ cd /var/www/html/page1.hello.com
yash@yash-VirtualBox:/var/www/html/page1.hello.com$ mkdir images
```

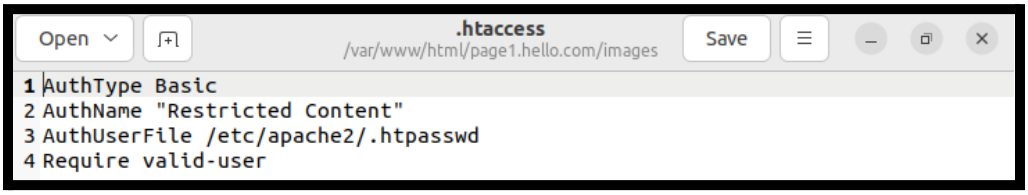
5. Switch directory: *cd images*

```
yash@yash-VirtualBox:/var/www/html/page1.hello.com$ cd images
yash@yash-VirtualBox:/var/www/html/page1.hello.com/images$ ls
```

6. Create and edit index file: *sudo gedit index.html*

```
yash@yash-VirtualBox:/var/www/html/page1.hello.com/images$ sudo gedit index.htm
l
```

7. Modify .htaccess file: *sudo gedit .htaccess*



The website content that requires user authentication, ***"images"*** is a new directory created within the ***/var/www/html/site1.example.com*** directory and inside it contains another two files, ***"index.html"*** and ***".htaccess"***. The index file holds the code that will generate the directory's default page whereas the ***".htaccess"*** file imposes access restrictions to the same directory.

AuthType Basic

AuthName "Restricted Content"

AuthUserFile /etc/apache2/ .htpasswd

Require valid-user

These lines of directives are found etched into the earlier produced ***".htaccess"*** file. They collectively specify the user authentication's attributes:

- ***Type -*** Basic, a simple method where the server prompts the users for a username and a corresponding password.
- ***Realm -*** Short description about the protected area that is displayed to the users when they are trying to access it.
- ***.htpasswd Location -*** Direct the server to where the ***".htpasswd"*** file is kept, which details the expected username and password.
- ***Users -*** Outline the type of users that are granted entry which are valid users that can provide the correct credentials.

8. Restart Apache: ***sudo systemctl restart apache2***

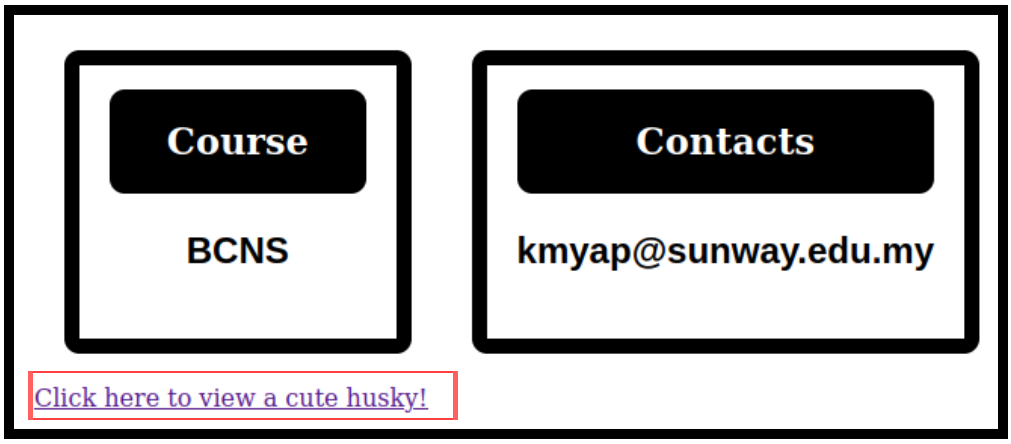
Check Apache's status: ***sudo systemctl status apache2***

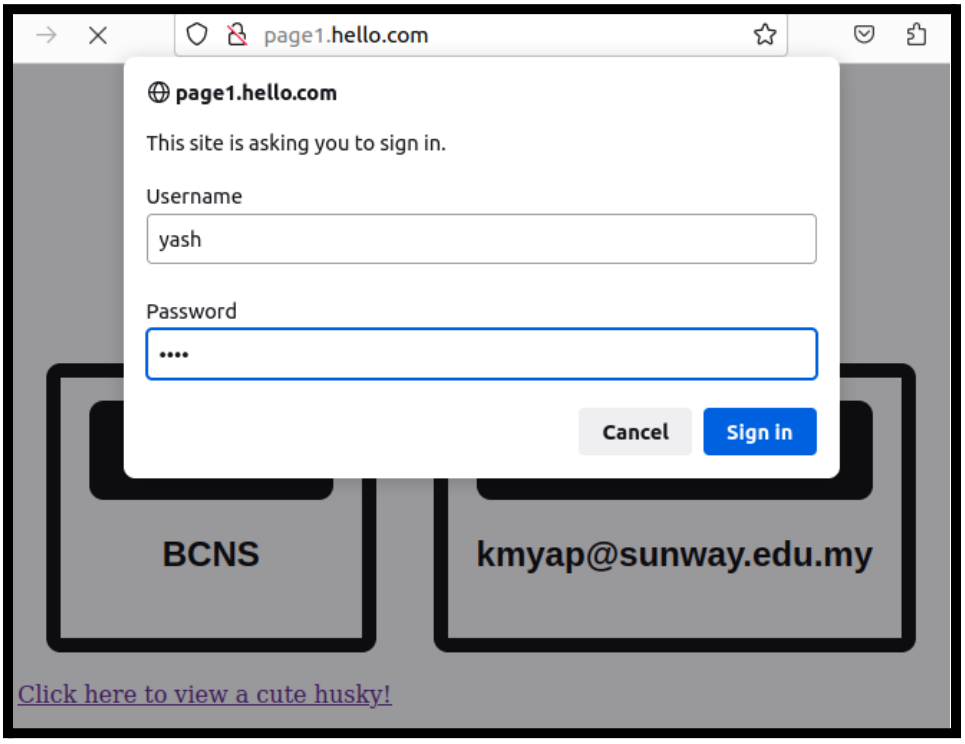
```
yash@yash-VirtualBox:~$ sudo systemctl restart apache2
yash@yash-VirtualBox:~$ sudo systemctl status apache2
● apache2.service - The Apache HTTP Server
     Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor pres>
     Active: active (running) since Fri 2023-07-14 11:09:46 +08; 7s ago
       Docs: https://httpd.apache.org/docs/2.4/
    Process: 32950 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/>
   Main PID: 32955 (apache2)
      Tasks: 55 (limit: 2262)
     Memory: 4.8M
        CPU: 112ms
     CGroup: /system.slice/apache2.service
             ├─32955 /usr/sbin/apache2 -k start
             ├─32956 /usr/sbin/apache2 -k start
             └─32957 /usr/sbin/apache2 -k start

Jul 14 11:09:46 yash-VirtualBox systemd[1]: Starting The Apache HTTP Server...
Jul 14 11:09:46 yash-VirtualBox systemd[1]: Started The Apache HTTP Server.
lines 1-16/16 (END)
```

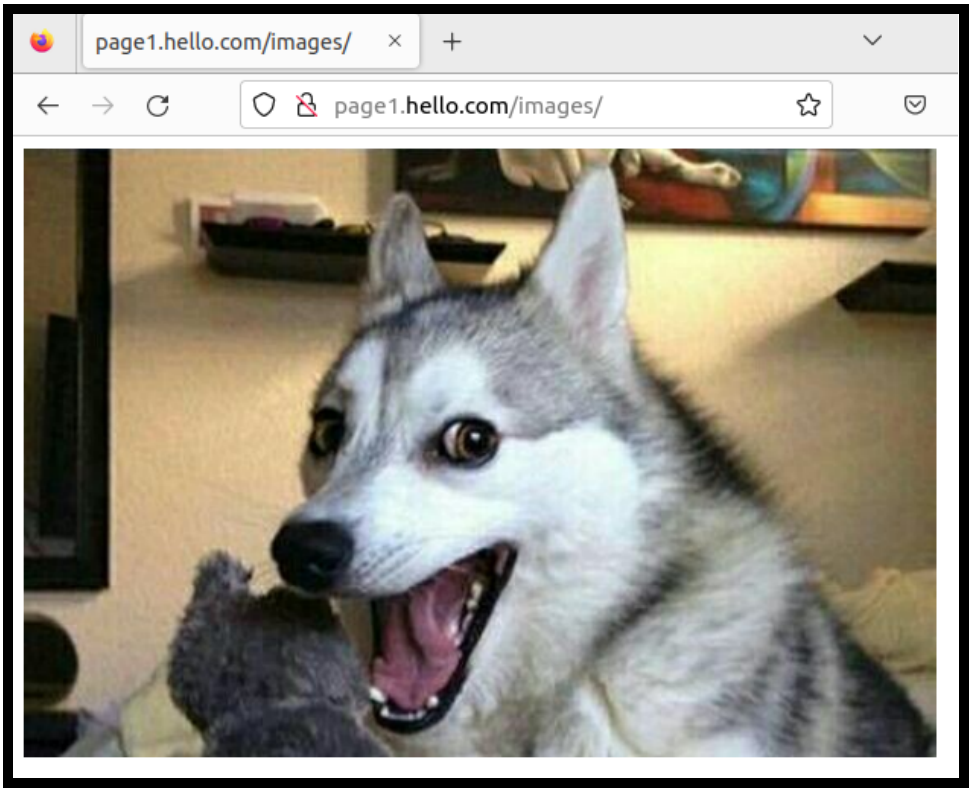
Command ***"systemctl"*** is a control interface for systemd used to manage system services and ***"restart"*** is one of its subcommands that stop and then starts a service

which is the Apache2 here. Given that so many changes were made, restarting the web server can help get those changes to take effect so we may test if everything is as anticipated. Assessing the Apache service's status after to ensure nothing is wrong with it.

User Authentication Test:







1.2.5 REQUIREMENT 5: SHELL SCRIPT

1. Switch directory: *cd /var/www/html*

Write shell script: *sudo gedit webserver_monitor.sh*

```
yash@yash-VirtualBox:~$ cd /var/www/html
yash@yash-VirtualBox:/var/www/html$ sudo gedit webserver_monitor.sh
```

In the directory ***/var/www/html***, create a shell script file called ***"webserver_monitor.sh"*** that is responsible for monitoring our web server's performance.

2. Input the shell script written below:

```
#!/bin/bash
```

```
# Variables
WEB_SERVER="localhost"   # Replace with your web
server hostname or IP address
WEB_SERVER_PORT="80"      # Replace with your web
server port
INTERVAL=5                  # Time interval between
each monitoring cycle (in seconds)
OUTPUT_FILE="webserver_performance.txt"

# Function to get timestamp
get_timestamp() {
  date +"%Y-%m-%d %H:%M:%S"
}

# Main monitoring loop
while true; do
  # Get server status
     server_status=$(curl   -s   -o   /dev/null   -w
"%{http_code}"
"http://$WEB_SERVER:$WEB_SERVER_PORT")

  # Get server load average
  load_average=$(uptime | awk -F'average: ' '{print
$2}')

  # Get free memory
  free_memory=$(free -m | awk 'NR==2{print $4}')

  # Get CPU usage percentage
```

```
    cpu_usage=$(top  -bn1  |  grep  "Cpu(s)"  |  awk
'{print $2 + $4}')

  # Get disk usage percentage
   disk_usage=$(df  -h  |  grep  "/$"  |  awk  '{print
$5}')

  # Get timestamp
  timestamp=$(get_timestamp)

  # Get connected clients
      connected_clients=$(netstat   -an   |   grep
ESTABLISHED | wc -l)



  # Output to file
  echo "Timestamp: $timestamp" >> "$OUTPUT_FILE"
     echo   "Server   Status:   $server_status"   >>
"$OUTPUT_FILE"
      echo   "Load   Average:   $load_average"   >>
"$OUTPUT_FILE"
     echo   "Free   Memory:   $free_memory   MB"   >>
"$OUTPUT_FILE"
  echo "CPU Usage: $cpu_usage%" >> "$OUTPUT_FILE"
  echo "Disk Usage: $disk_usage" >> "$OUTPUT_FILE"
   echo "Connected clients: $connected_clients" >>
“$OUTPUT_FILE"
    echo  "----------------------------------"  >>
"$OUTPUT_FILE"
```

```
  # Display current stats
  echo "Timestamp: $timestamp"
  echo "Server Status: $server_status"
  echo "Load Average: $load_average"
  echo "Free Memory: $free_memory MB"
  echo "CPU Usage: $cpu_usage%"
  echo "Disk Usage: $disk_usage"
  echo “Connected clients: $connected_clients"
  echo "-----------------------------------"

  # Wait for the next interval
  sleep $INTERVAL
done
```

```
yash@yash-VirtualBox:/var/www/html$ ls
index.html      page1.hello.com       webserver_performance.txt
page1.bye.com   webserver_monitor.sh
yash@yash-VirtualBox:/var/www/html$
```

The shell script begins with some variable declaration which we have updated accordingly based on our specific server setup. Next is the ***"get_timestamp()"*** function that fetches and displays the time when the performance monitoring was done. Once the timestamp retrieval is completed, the main monitoring commences where it aims to collect the web server's performance metrics such as status code, load average, free memory, CPU usage, disk usage, and number of connected clients. Gathered data alongside the timestamp are then appended onto the ***"webserver_performance.txt"*** file, so they could later be shown on the console. It is important to note that the script will continue the said monitoring loop, collecting and logging the stated metric at each interval if it is not interrupted on the users' end.

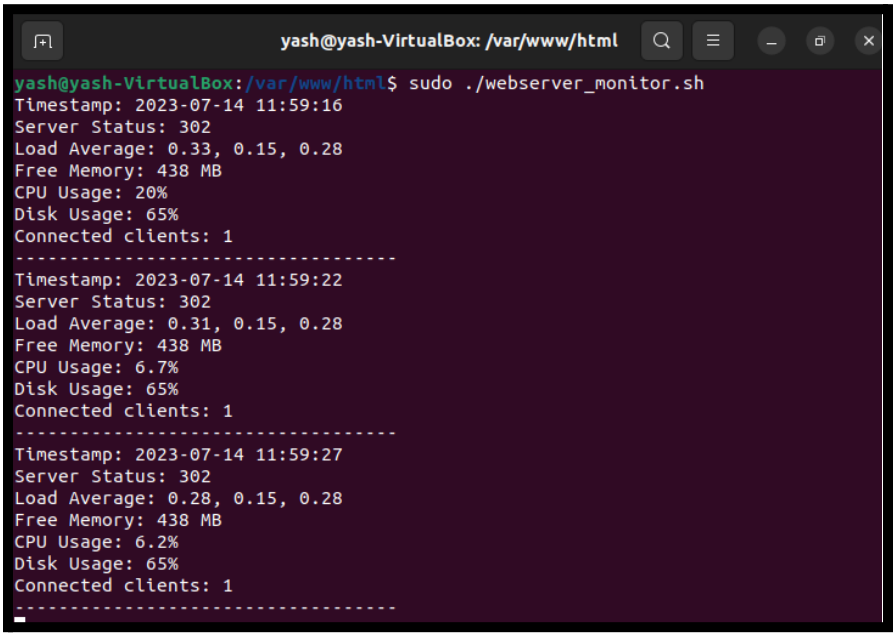
3. Make the shell script file executable: *sudo chmod +x webserver_monitor.sh.*

```
yash@yash-VirtualBox:/var/www/html$ sudo chmod +x webserver_monitor.sh
```

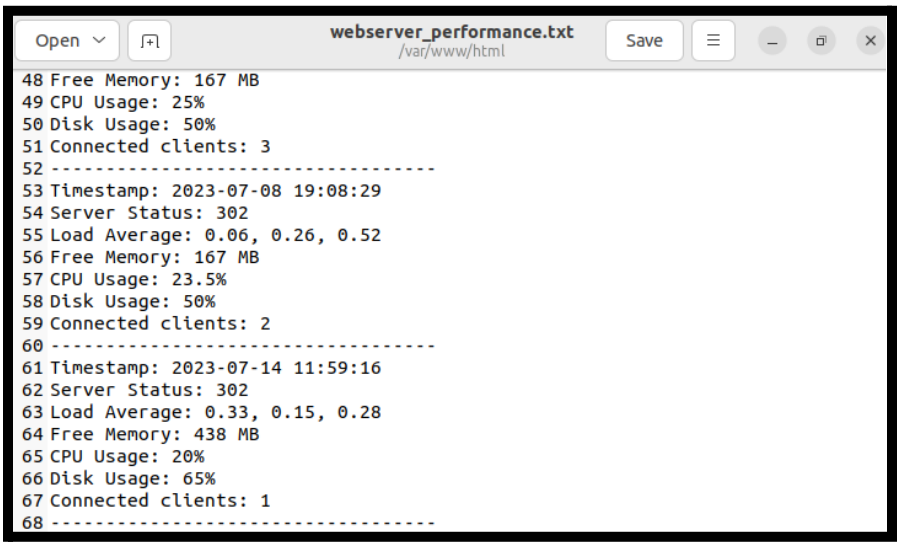
"Chmod" stands for change mode, it is used to change the permission of a file which in this case is our shell script, ***"webserver_monitior.sh"***. The ***"+x"*** option applied here authorises the file in question to be executable. With execute permissions, ***"webserver_monitior.sh"*** now can be run to monitor web server's performance

.

4. Run shell script: *sudo ./webserver_monitor.sh*



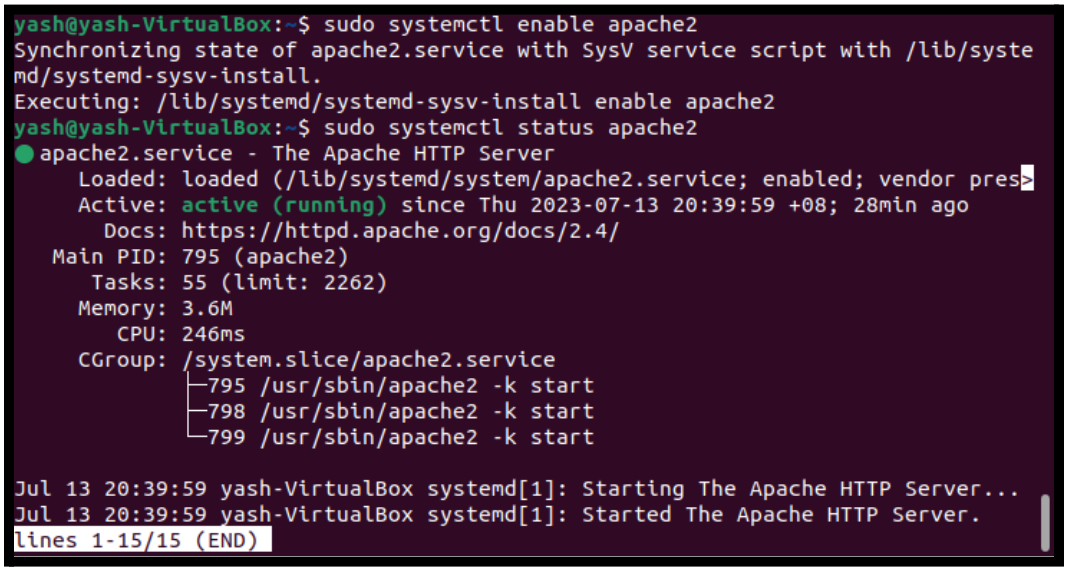
Test running the shell script to verify whether the output information aligns with our expectations and configurations.



1.2.6 REQUIREMENT 6: AUTO-START

1. Enable Apache service: *sudo systemctl enable apache2*

2. Check Apache's status: *sudo systemctl status apache2*



The phrase ***"enable"*** is an option of ***"systemctl"*** that enables a service to start at boot, in this case, the specific service is Apache2. The whole command line seeks to have the Apache web server start automatically during system boot, therefore allowing it to listen and deliver web content requests as soon as the system boots up. Similar to before, the follow-up ***"status"*** option is used to affirm that the Apache service is up and running even after our further configurations.

1.3 CONFIGURATION CHALLENGES AND WORKAROUNDS

One of the significant challenges encountered during web server configuration relates to web authentication. As instructed by the assignment prompt, we established an ***"images"*** directory and attempted to link it to our main webpage, ***"page1.hello.com"***. This involved creating an `**<a>**` tag in ***"page1.hello.com"*** that directs users to the images directory page. However, we initially faced difficulties in establishing the link due to errors in our HTML code. Through persistent trial and error, we identified and rectified the mistake, ultimately resolving the issue.

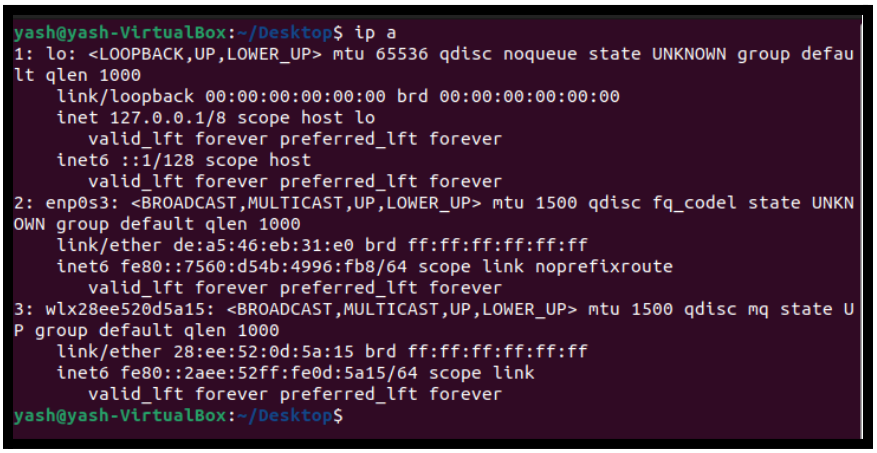
2.0 DNS SERVER

2.1 OVERVIEW

The Domain Name System (DNS) is a fundamental protocol used on the internet to translate domain names into numerical IP addresses. Every device connected to the internet has its own IP address, which other devices use to find and connect to it. DNS servers take care of the behind-the-scenes work, enabling users to access websites and services by using easy-to-remember domain names instead of complicated IP addresses.

2.2 IMPLEMENTATION DETAILS

1. **Find out the list of network interfaces currently available: *ip a***



The command **"*ip a*"** is a command used to display all the information about the network interfaces on the system such as interface name, IP addresses, network masks, and many more. This command helps in configuring the DNS server as well as troubleshooting any network connectivity issues. In this case, identify the network interface of choice that will be used to configure the DNS server.

2. Install BIND for Ubuntu: *sudo apt install bind9*

```
yash@yash-VirtualBox:~$ sudo apt install bind9
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
bind9 is already the newest version (1:9.18.12-0ubuntu0.22.04.2).
0 upgraded, 0 newly installed, 0 to remove and 1 not upgraded.
yash@yash-VirtualBox:~$
```

This command is used to install the BIND9 package. BIND is a widely used DNS server on the internet and it's commonly used to provide DNS resolution services. Running this command will initiate the package installation process. The package manager will fetch the necessary files from Ubuntu software repositories, resolve dependencies and install BIND to the system.

3. Check the version of BIND: *named -v*

```
yash@yash-VirtualBox:~$ named -v
BIND 9.18.12-0ubuntu0.22.04.2-Ubuntu (Extended Support Version) <id:>
yash@yash-VirtualBox:~$
```

The command *"**named -v**"* is used to check the version of the BIND software installed on the system. It will execute the named command with the *"**-v**"* option, which will output the version information of the installed BIND DNS server.

4. Enter BIND directory: *cd /etc/bind*

```
yash@yash-VirtualBox:~$ cd /etc/bind
yash@yash-VirtualBox:/etc/bind$
```

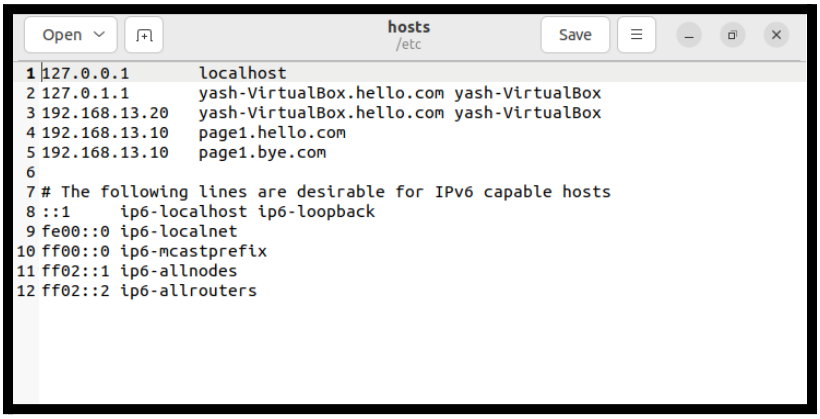
The command ***"cd /etc/bind"*** is used to change the current working directory to *"**/etc/bind**"*. By using this command, the directory path is moved to "**/etc/bind**". This directory is where the configuration files for the BIND server are stored in Ubuntu. By changing to this directory, BIND configuration files such as ***named.conf.local*** and ***named.conf.options*** can be accessed and managed allowing to make modifications and customize the DNS server's settings.

5. Display hostname and status: *hostnamectl status*

```
yash@yash-VirtualBox:/etc/bind$ hostnamectl status
 Static hostname: yash-VirtualBox
       Icon name: computer-vm
         Chassis: vm
      Machine ID: 7c33b92cdfcd4f7985e4b45d49d0a830
         Boot ID: db656043a9d8449e8dba11ff54902680
  Virtualization: oracle
Operating System: Ubuntu 22.04.2 LTS
          Kernel: Linux 5.19.0-46-generic
    Architecture: x86-64
 Hardware Vendor: innotek GmbH
  Hardware Model: VirtualBox
yash@yash-VirtualBox:/etc/bind$
```

The command is used to display the current system hostname and its related information. The *"**hostnamectl**"* command is a system control utility that allows users to query and modify the hostname configuration on a Ubuntu system. By running ***"hostnamectl status"***, detailed retrievals such as the static and transient hostname, the operating system's chassis type, the kernel version, and the current time settings are listed. This command is helpful in verifying the system's hostname configuration, which is relevant for DNS operations as it determines how the system identifies itself on the network and how it can be accessed by other devices.

6. Modify /etc/hosts file and input DNS server IP: *sudo gedit /etc/hosts*



This command is used to open the file ***"/etc/hosts"***. The host file is a local text file that will allow users to manually map IP addresses to hostnames, providing a simple form of DNS resolution on the system. By opening this file with administrative privileges, you

can modify its contents, add or remove hostname-to-IP mappings or make other changes related to local DNS resolution. The ***"hosts"*** file can be used for various purposes such as testing, network configuration, or overriding DNS lookup for specific domains. When you save and exit the editor after making changes, the modified ***"hosts"*** file will be used by the system.

7. Check current hostname: *hostname*

```
yash@yash-VirtualBox:/etc/bind$ hostname
yash-VirtualBox
```

This command is used to display the current system hostname. When this command is executed, the command simply outputs the hostname of the system. The hostname in this case is ***"yash-VirtualBox"***. This is an essential component to identify and address systems on a network.

8. Check DNS domain name: *dnsdomainname*

```
yash@yash-VirtualBox:/etc/bind$ dnsdomainname
hello.com
```

This command is used to display the system's DNS domain name in this case, ***"hello.com"***. The DNS domain name represents the domain in which the system is located within the DNS domain. The DNS domain name is an important component in DNS resolution as it helps to determine how domain names are put up and resolved to IP addresses.

9. Check FQDN (Fully Qualified Domain Name): *hostname - -fqdn*

```
yash@yash-VirtualBox:/etc/bind$ hostname --fqdn
yash-VirtualBox.hello.com
yash@yash-VirtualBox:/etc/bind$
```

This command is used to display the ***fully qualified domain name (FQDN)*** of the system. When this command is executed, it outputs the ***"fqdn"*** of the system which is

yash-VirtualBox.hello.com. Hence, in this case, the hostname is ***"yash-VirtualBox"*** and the host is located within the domain ***"hello.com"***.

10. Make a copy of "*named.conf.options*":

sudo cp named.conf.options named.conf.options.orig

```
yash@yash-VirtualBox:/etc/bind$ sudo cp named.conf.options named.conf.options.r
ig
```

This command is used to make a copy of the ***"named.conf.options"*** file with a new name ***"named.conf.options.orig"***. The reasoning behind this command is to create a backup file of the original file before making any amendments.

11. **Edit the "*named.conf.options*" file: *sudo gedit named.conf.options***

Open | named.conf.options /etc/bind | Save

```
 7
 8         // If your ISP provided one or more IP addresses for stable
 9         // nameservers, you probably want to use them as forwarders.
10         // Uncomment the following block, and insert the addresses replacing
11         // the all-0's placeholder.
12
13         // forwarders {
14         //      0.0.0.0;
15         // };
16
17         //
   ============================================================================
18         // If BIND logs error messages about the root key being expired,
19         // you will need to update your keys.  See https://www.isc.org/bind-
   keys
20         //
   ============================================================================
21         dnssec-validation auto;
22
23         listen-on-v6 { any; };
24         recursion yes;
25         listen-on {192.168.13.20;};
26         allow-transfer {none;};
27
28         forwarders {
29         10.0.2.2;
30         };
31 };
```

This command opens the ***"named.conf.options"*** file for editing with administrative privileges. This file contains configuration options for the BIND DNS server software.

***"listen-on-v6"*:** used to define the IPv6 addresses or IP ranges on which the DNS server listens for DNS queries and connections.

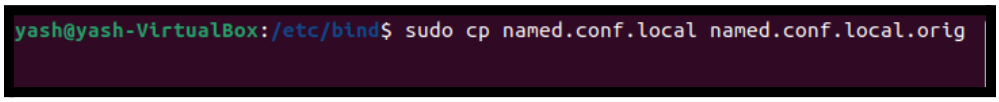
***"recursion"*:** used to enable or disable recursion which allows the DNS server to perform frequent queries on behalf of clients.

***"listen-on"*:** specifies the IP addresses and ports on which the DNS server should listen to incoming queries.

"forwarders": defines the IP addresses of other DNS servers to which queries should be forwarded if the local DNS server does not have the information that they have requested.

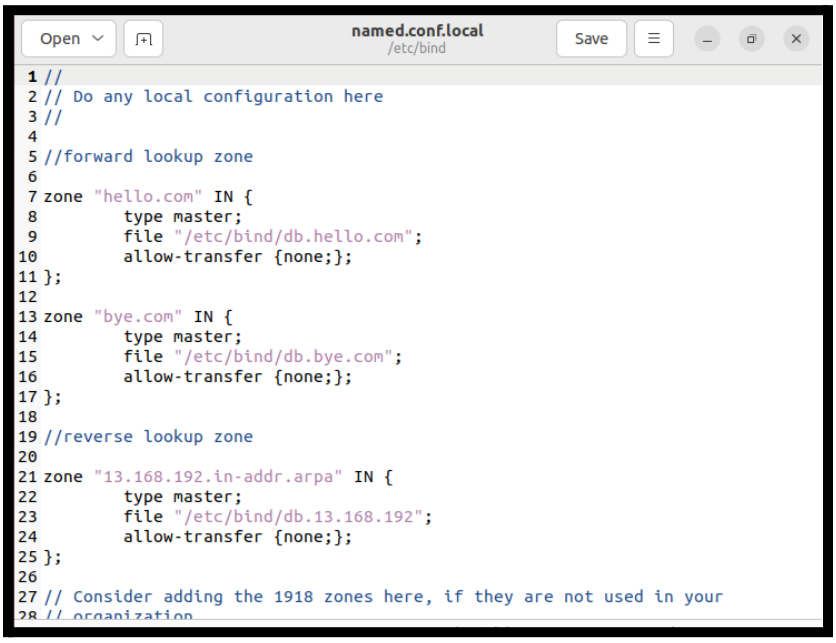
12. Make a copy of *"named.conf.local"*:

sudo cp named.conf.local named.conf.local.orig



This command is used to create a backup copy of the ***"named.conf.local"*** file.

13. Edit the "*named.conf.local*" file: *sudo gedit named.conf.local*



This command opens the ***"named.conf.local"*** file for editing with administrative privileges. Insert the forward and reverse lookup zones in this file.

14. **Check the syntax of "named.conf.local" and "named.conf.options": *named-checkconf***

This command is used to check the syntax and validity of the BIND configuration files. If the command does not produce any output, this signifies that the syntax of the configuration files is correct and valid.

15. Make a copy of "*db.local*" into the first domain's forward zone:

sudo cp db.local db.hello.com

```
yash@yash-VirtualBox:/etc/bind$ sudo cp db.local db.hello.com
```

This command is used to create a copy of the ***"db.local"*** file to the first domain's forward zone file ***"db.hello.com"***. The reason for this duplication is to modify the file independently without disturbing the original file.

2.2.1 REQUIREMENT 1: FORWARD ZONE

16. **Edit the "*db.hello.com*" file*: sudo gedit db.hello.com***

Open | db.hello.com /etc/bind | Save

```
;
; BIND data file for local loopback interface
;
$TTL    604800
@       IN      SOA     ns1.hello.com. root.hello.com. (
                              2         ; Serial
                         604800         ; Refresh
                          86400         ; Retry
                        2419200         ; Expire
                         604800 )       ; Negative Cache TTL
;
@       IN      NS      ns1.hello.com.
ns1     IN      A       192.168.13.20
page1   IN      A       192.168.13.10
www     IN      A       192.168.13.10
lect    IN      CNAME   www
@       IN      AAAA    ::1
```

This command opens the ***"db.hello.com"*** file using ***"gedit"***. It allows modification of the forward zone configuration for the ***"hello.com"*** domain.

SOA: Start of Authority, which specifies the primary DNS server responsible for the zone and includes details like the serial number and so on.

NS: Name Server, which indicates the authoritative name servers for the zone and lists the servers responsible for answering DNS queries for the domain.

A: Address records map hostnames to their corresponding IPv4 addresses. Each **A record** consists of a hostname and its associated IPv4 address.

CNAME: Canonical Name, which provides an alias or canonical name for a hostname. They redirect DNS queries for a hostname to another hostname. ***"lect"*** is our alias for ***"www.hello.com".***

AAAA: Serves the same purpose as A records, but for IPv6 addresses instead.

17. Check the syntax for the forward zone file of "*hello.com*":

named-checkzone hello.com db.hello.com

```
yash@yash-VirtualBox:/etc/bind$ named-checkzone hello.com db.hello.com
zone hello.com/IN: loaded serial 2
OK
yash@yash-VirtualBox:/etc/bind$
```

This command is used to check the validity and syntax of a DNS zone file. It checks on the inconsistencies or potential issues that may cause problems in the DNS zone operation. If the output of the command is as shown above, the syntax of the file is valid and correct.

18. Repeat steps 15, 16, and 17 for the second domain, *"bye.com"*

```
yash@yash-VirtualBox:/etc/bind$ sudo cp db.local db.bye.com
```

Open ▾ | db.bye.com /etc/bind | Save

```
1 ;
2 ; BIND data file for local loopback interface
3 ;
4 $TTL    604800
5 @       IN      SOA     ns1.bye.com. root.bye.com. (
6                               2         ; Serial
7                          604800         ; Refresh
8                           86400         ; Retry
9                         2419200         ; Expire
10                         604800 )       ; Negative Cache TTL
11 ;
12 @       IN      NS      ns1.bye.com.
13 ns1     IN      A       192.168.13.20
14 page1   IN      A       192.168.13.10
15 www     IN      A       192.168.13.10
16 stu     IN      CNAME   www
17 @       IN      AAAA    ::1
```

```
yash@yash-VirtualBox:/etc/bind$ named-checkzone bye.com db.bye.com
zone bye.com/IN: loaded serial 2
OK
yash@yash-VirtualBox:/etc/bind$
```

2.2.2 REQUIREMENT 2: DOMAIN ALIASES

```
11 ;
12 @       IN      NS      ns1.hello.com.
13 ns1     IN      A       192.168.13.20
14 page1   IN      A       192.168.13.10
15 www     IN      A       192.168.13.10
16 lect    IN      CNAME   www
17 @       IN      AAAA    ::1
```

```
11 ;
12 @       IN      NS      ns1.bye.com.
13 ns1     IN      A       192.168.13.20
14 page1   IN      A       192.168.13.10
15 www     IN      A       192.168.13.10
16 stu     IN      CNAME   www
17 @       IN      AAAA    ::1
```

The alias for ***"www.hello.com"*** is ***"lect"*** whereas the alias for ***"www.bye.com"*** is ***"stu".***

2.2.3 REQUIREMENT 3: REVERSE ZONE

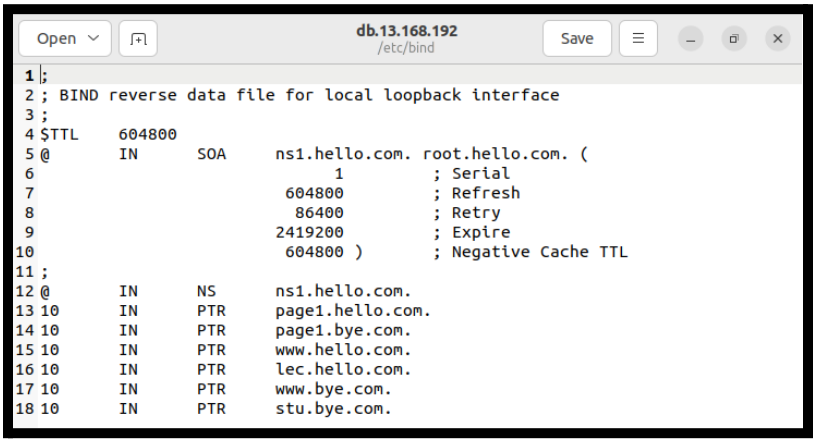
19. **Make a copy of *"db.127"* into the reverse zone file, "*db.13.168.192":***

sudo cp db.127 db.13.168.192

```
yash@yash-VirtualBox:/etc/bind$ sudo cp db.127 db.13.168.192
```

This command is used to create a copy of the ***"db.127"*** file to the reverse zone file, ***"db.13.168.192"***. This is to ensure that there is a backup copy of the original file in order to edit without modifying the original file.

20. Edit the reverse zone file: *sudo gedit db.13.168.192*



This command is used to open the ***"db.13.168.192"*** file. This enables the ability to modify the content of the file to define the DNS records, configure zone-specific settings, and many more. Inside the file, modifications on SOA, NS, and PTR are enabled.

SOA: Start of Authority, which specifies the primary DNS server responsible for the zone and includes details like the serial number and so on.

NS: Name Server, which indicates the authoritative name servers for the zone, listing the servers responsible for answering DNS queries for the domain.

PTR: Pointer, which is used in reverse DNS lookup to map an IP address to a domain name.

21. Check the syntax for the reverse zone file:

named-checkzone 13.168.192.in-addr.arpa db.13.168.192

```
yash@yash-VirtualBox:/etc/bind$ named-checkzone 13.168.192.in-addr.arpa db.13.1
68.192
zone 13.168.192.in-addr.arpa/IN: loaded serial 1
OK
```

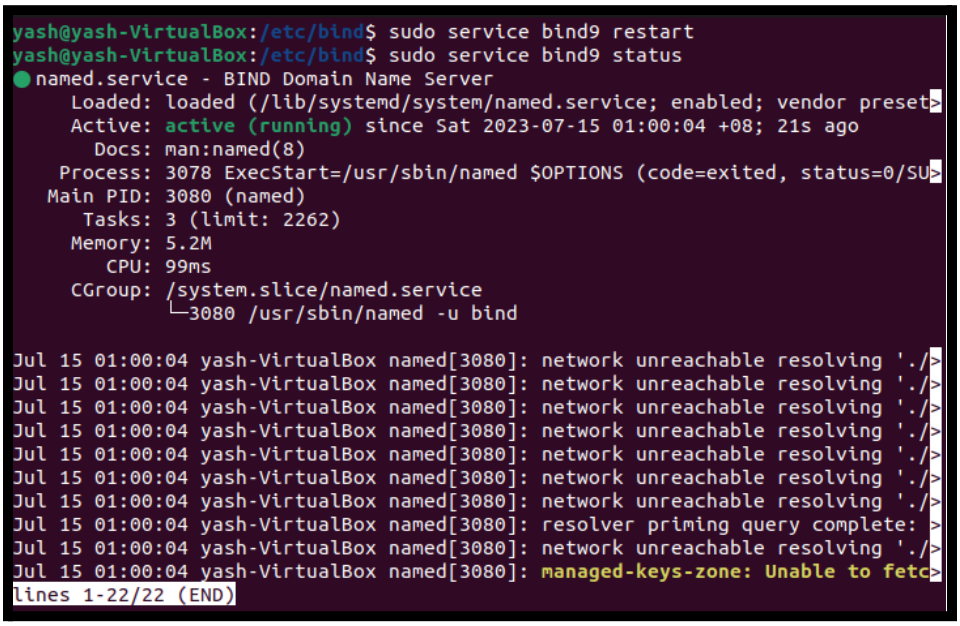
This command is used to check the validity and syntax of a reverse DNS zone file. The output shown above signifies that the syntax of the reverse zone file is correct and valid.

22. **Restart BIND for changes to take place:** ***sudo service bind9 restart***

```
yash@yash-VirtualBox:/etc/bind$ sudo service bind9 restart
yash@yash-VirtualBox:/etc/bind$
```

The command ***"sudo service bind9 restart"*** is used to restart the BIND server. Any currently running instances of the BIND DNS server are stopped, terminating any active processes and freeing up system resources. The service is then started again, initiating a fresh instance of the BIND server with the updated configuration. The DNS server reads the configuration files, including zone files and settings, and initializes its services. Once the BIND server is successfully restarted, it is ready to handle DNS queries and provide DNS resolution services according to the newly loaded configuration.

23. Display status of BIND: *sudo service bind9 status*



This command is used to check the current status of the BIND server service and allows users to know if the server is active(running) or stopped(failing).

24. Query the DNS server to look up *"www.hello.com"* :

nslookup www.hello.com

This command is a command-line tool used to perform DNS queries and retrieve information from DNS servers. When this command is executed followed by a domain name or IP address, it initiates a DNS lookup to retrieve information associated with that domain or IP address. However, in the output above, the server and the address are not stated. Hence, we have to edit the ***"resolv.conf"*** file.

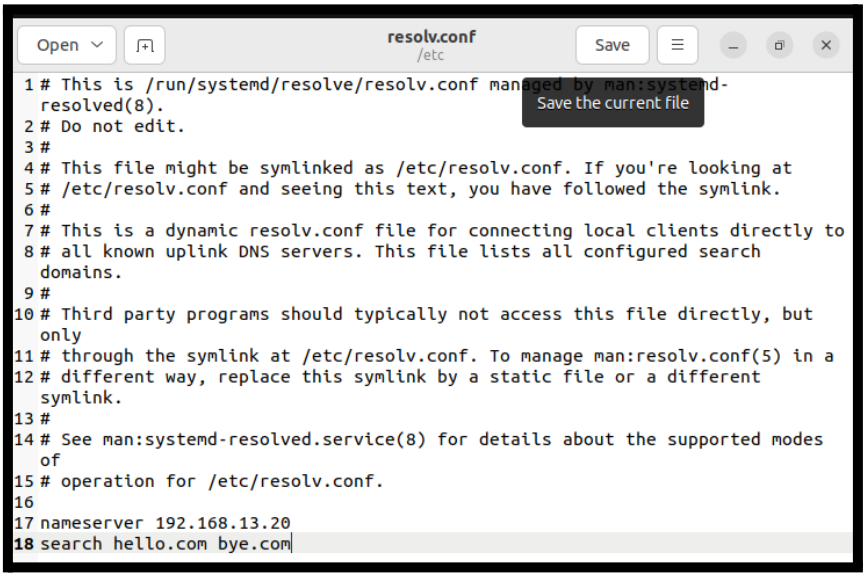
25. **Display the file contents of resolv.conf: *cat /etc/resolv.conf***

```
yash@yash-VirtualBox:/etc/bind$ cat /etc/resolv.conf
# This is /run/systemd/resolve/resolv.conf managed by man:systemd-resolved(8).
# Do not edit.
#
# This file might be symlinked as /etc/resolv.conf. If you're looking at
# /etc/resolv.conf and seeing this text, you have followed the symlink.
#
# This is a dynamic resolv.conf file for connecting local clients directly to
# all known uplink DNS servers. This file lists all configured search domains.
#
# Third party programs should typically not access this file directly, but only
# through the symlink at /etc/resolv.conf. To manage man:resolv.conf(5) in a
# different way, replace this symlink by a static file or a different symlink.
#
# See man:systemd-resolved.service(8) for details about the supported modes of
# operation for /etc/resolv.conf.

# No DNS servers known.
search .
```

The command ***"cat /etc/resolv.conf"*** is used to display the contents of the ***"resolv.conf"*** file. It may also include additional settings like search domains, domain name suffixes, and other configuration directives. The displayed content helps in determining the current DNS configuration of the Ubuntu system.

26. Edit the file to input your DNS server, the server's IP address and domains:

sudo gedit /etc/resolv.conf



This command is used to open and edit the ***"resolv.conf"*** file with the admin privilege. This file contains DNS-related information such as the IP addresses of the DNS servers used and other configuration directives. When this command is executed and if the name server file says that "no DNS server found", users may need to manually key in the DNS' IP address.

27. **Query the DNS server to look up both domains:**

nslookup www.hello.com

nslookup www.bye.com

```
yash@yash-VirtualBox:~/startup$ nslookup www.hello.com
Server:         192.168.13.20
Address:        192.168.13.20#53

Name:   www.hello.com
Address: 192.168.13.10

yash@yash-VirtualBox:~/startup$ nslookup www.bye.com
Server:         192.168.13.20
Address:        192.168.13.20#53

Name:   www.bye.com
Address: 192.168.13.10
```

This command is a command-line tool used to perform DNS queries and retrieve information from DNS servers. When this command is executed followed by a domain name or IP address, it initiates a DNS lookup to retrieve information associated with that domain or IP address.

28. Ping the websites “www.hello.com” and “www.bye.com”:

ping www.hello.com

ping www.bye.com

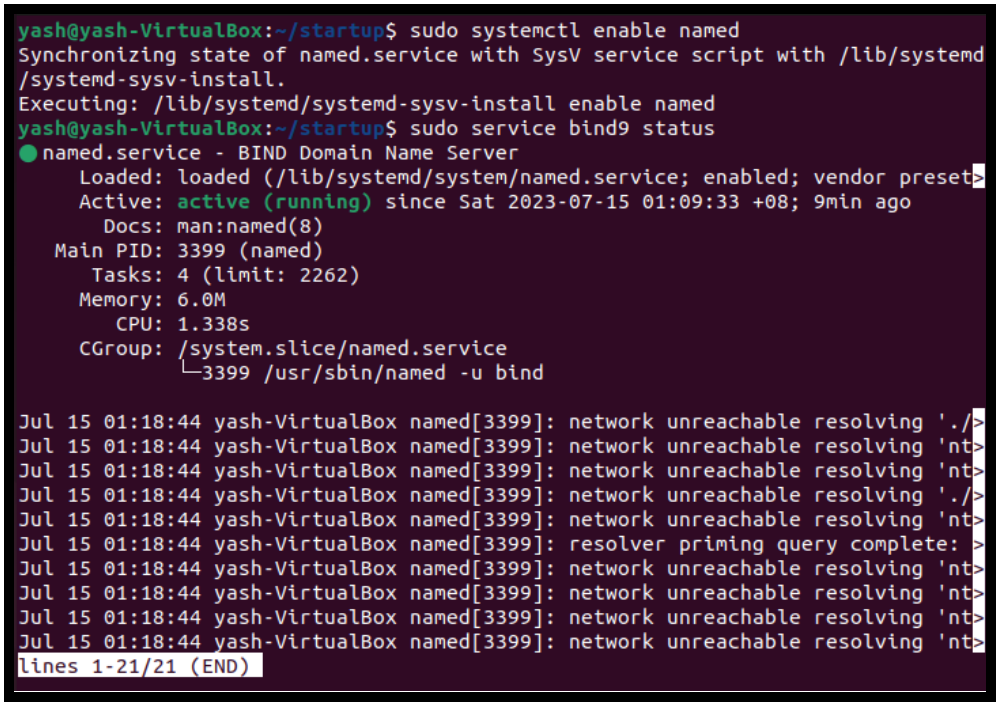
```
yash@yash-VirtualBox:~/startup$ ping www.hello.com
PING www.hello.com (192.168.13.10) 56(84) bytes of data.
64 bytes from www.hello.com (192.168.13.10): icmp_seq=1 ttl=64 time=0.266 ms
64 bytes from www.hello.com (192.168.13.10): icmp_seq=2 ttl=64 time=0.919 ms
64 bytes from page1.hello.com (192.168.13.10): icmp_seq=3 ttl=64 time=1.06 ms
^C
--- www.hello.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 0.266/0.746/1.055/0.344 ms
yash@yash-VirtualBox:~/startup$ ping www.bye.com
PING www.bye.com (192.168.13.10) 56(84) bytes of data.
64 bytes from lec.hello.com (192.168.13.10): icmp_seq=1 ttl=64 time=0.474 ms
64 bytes from www.hello.com (192.168.13.10): icmp_seq=2 ttl=64 time=1.57 ms
64 bytes from lec.hello.com (192.168.13.10): icmp_seq=3 ttl=64 time=2.06 ms
^C
```

The commands ***"ping www.hello.com"*** and ***"ping www.bye.com"*** are used to send ICMP which is known as Internet Control Message Protocol echo requests to the specified domain names and check if they can be reached over the network. By executing the commands, Ubuntu will attempt to resolve the domain names to their corresponding IP addresses using the DNS servers configured in the system. The destination server receives the echo requests and responds with ICMP echo replies if it is reachable. The command keeps sending echo requests until interrupted manually or after a specified number of packets are sent.

2.2.4 REQUIREMENT 4: AUTO-START

29. **Enable autostart for BIND: *sudo systemctl enable named***

```
yash@yash-VirtualBox:~/startup$ sudo systemctl enable named
Synchronizing state of named.service with SysV service script with /lib/systemd
/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable named
```



The command ***"sudo systemctl enable named"*** is used to enable the ***"named"*** service to start automatically at system boot. The command creates symbolic links or unit files in the appropriate system directories. These symbolic links or unit files tell ***"systemd"*** to start the ***"named"*** service during the boot process. It then enables the service to ensure that the BIND DNS server starts automatically whenever the system is powered on or restarted.

2.3 CONFIGURATION CHALLENGES AND WORKAROUNDS

During the configuration of the DNS server in Ubuntu, we encountered a specific issue where websites could not be pinged despite having set up the DNS server. To address this problem, we devised a workaround by establishing a separate virtual interface to connect the web server and the DNS server with the virtual machine's network interface (enp0s3). We opted for the ***macvlan*** type bridge method to achieve this setup.

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defau
lt qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UNKN
OWN group default qlen 1000
    link/ether de:a5:46:eb:31:e0 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic noprefixroute enp0s3
       valid_lft 74211sec preferred_lft 74211sec
    inet6 fe80::7560:d54b:4996:fb8/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
3: eth0@enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state
UP group default qlen 1000
    link/ether 86:32:4e:69:3f:e6 brd ff:ff:ff:ff:ff:ff
    inet 192.168.13.10/24 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::8432:4eff:fe69:3fe6/64 scope link
       valid_lft forever preferred_lft forever
4: eth1@enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state
UP group default qlen 1000
    link/ether 1e:11:63:e7:29:66 brd ff:ff:ff:ff:ff:ff
    inet 192.168.13.20/24 scope global eth1
       valid_lft forever preferred_lft forever
    inet6 fe80::1c11:63ff:fee7:2966/64 scope link
       valid_lft forever preferred_lft forever
yash@yash-VirtualBox:/home$
```

However, we encountered another hurdle in that these virtual interfaces would disappear after rebooting the system. To overcome this challenge, we devised a solution by creating a shell script that would automatically set up the virtual interfaces upon system reboot. This script ensured that the virtual interfaces necessary for the proper functioning of both the web server and DNS server were consistently established. By implementing this workaround and setting up the shell script, we were able to address the issue of websites not being pinged and maintain the stability of the virtual interfaces even after the system reboots. This experience emphasized the importance of finding practical solutions and implementing automated processes to overcome configuration challenges effectively.

```
1 sudo ip link add eth0 link enp0s3 type macvlan mode bridge
2 sudo ip link set dev eth0 up
3 sudo ip addr add 192.168.13.10/24 dev eth0
4 sudo ip link add eth1 link enp0s3 type macvlan mode bridge
5 sudo ip link set dev eth1 up
6 sudo ip addr add 192.168.13.20/24 dev eth1
7 sudo cp /etc/resolv1.conf /etc/resolv.conf
8 sudo ifconfig wlx28ee520d5a15 192.168.13.50
```

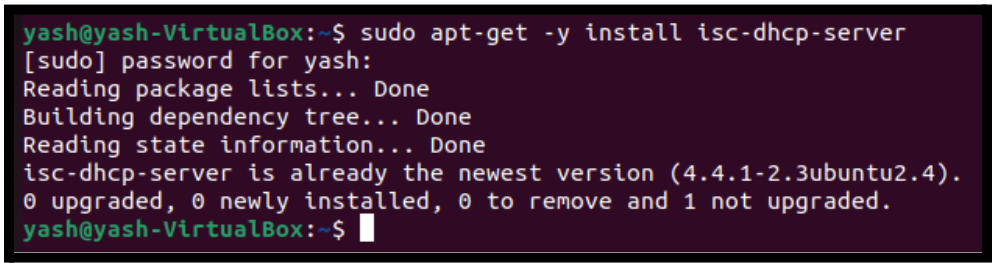
3.0 DHCP SERVER

3.1 OVERVIEW

DHCP is a network protocol used to automate the process of signing IP addresses and other network configuration parameters to devices on a network. It provides a centralised and dynamic method for managing IP address allocation, eliminating the need for manual configurations on individual devices.

3.2 IMPLEMENTATION DETAILS

1. **Install ISC DHCP server:** ***sudo apt-get -y install isc-dhcp-server***

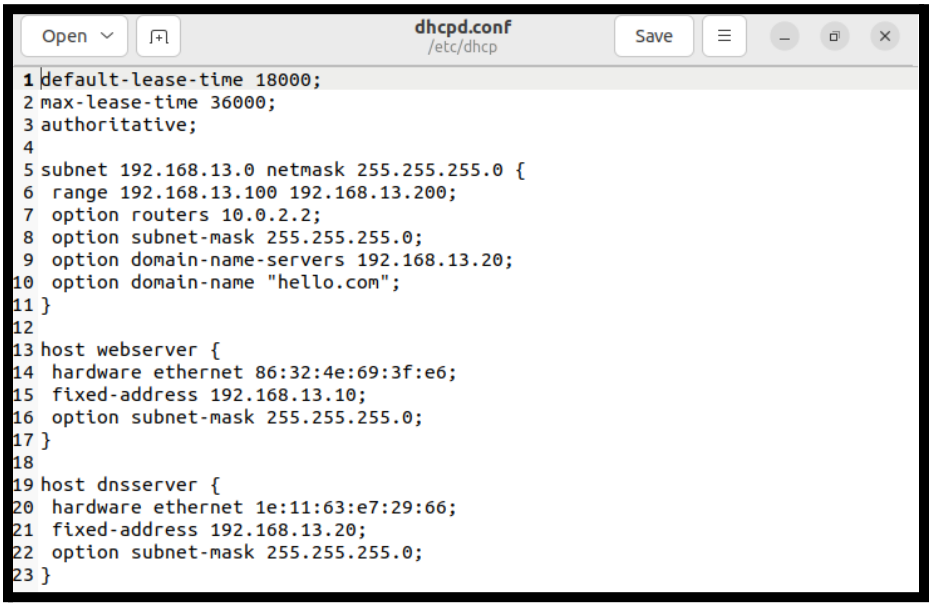


This line of code is to install the ISC DHCP server package without requiring manual confirmation at any stage of the installation process

2. **Create copy file:** ***sudo cp /etc/dhcp/dhcpd.conf /etc/dhcp/dhcpd.conf.orig***

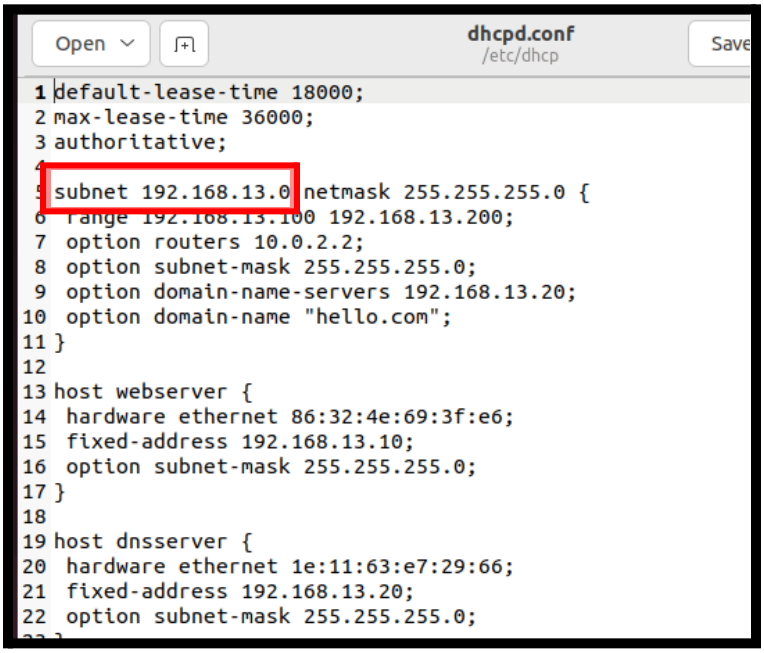
A backup of the original file ***"dhcpd.conf"*** is created by making a copy named ***"dhcpd.conf.orig"***. This is useful in case we need to revert back to the original file or compare it with changes made for future reference. ***"cp"*** is the command used to copy files and directories in Ubuntu. ***/etc/dhcp/dhcpd.conf*** is the source file that we want to duplicate which is located in the ***/etc/dhcp*** directory. ***/etc/dhcp/dhcpd.conf.orig*** is the destination file where the copy will be saved.

3. Configure DHCP Server: *sudo gedit /etc/dhcp/dhcpd.conf*

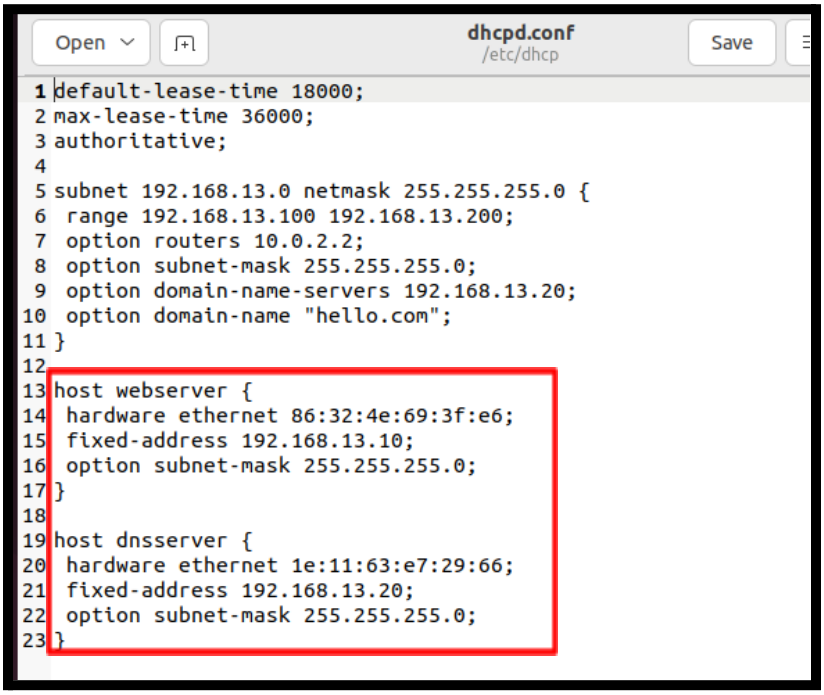


/etc/dhcp/dhcpd.conf is the path to the file we want to open. This line opens the ***"dhcp.conf"*** file in the **gedit** text editor with administrative privileges. It allows us to view and make any necessary changes to the configuration settings of the DHCP server.

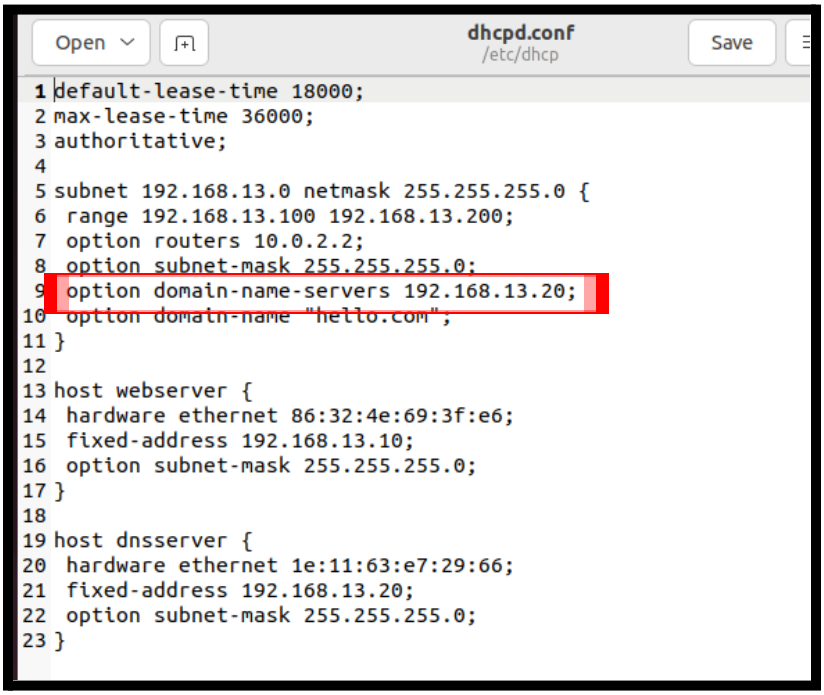
3.2.1 REQUIREMENT 1: CORRECT SUBNET



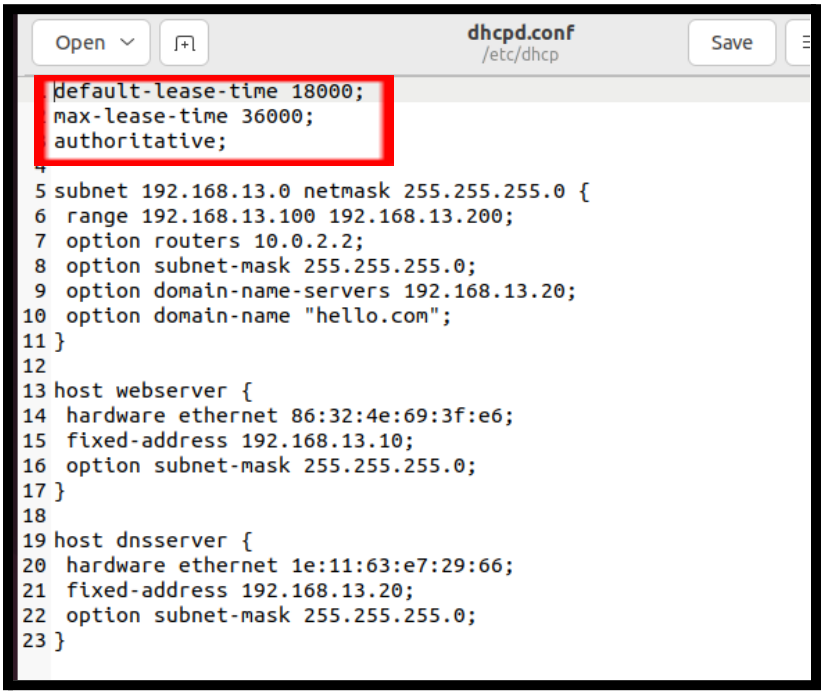
3.2.2 REQUIREMENT 2: STATIC DHCP ASSIGNMENTS



3.2.3 REQUIREMENT 3: DNS ADVERTISEMENT



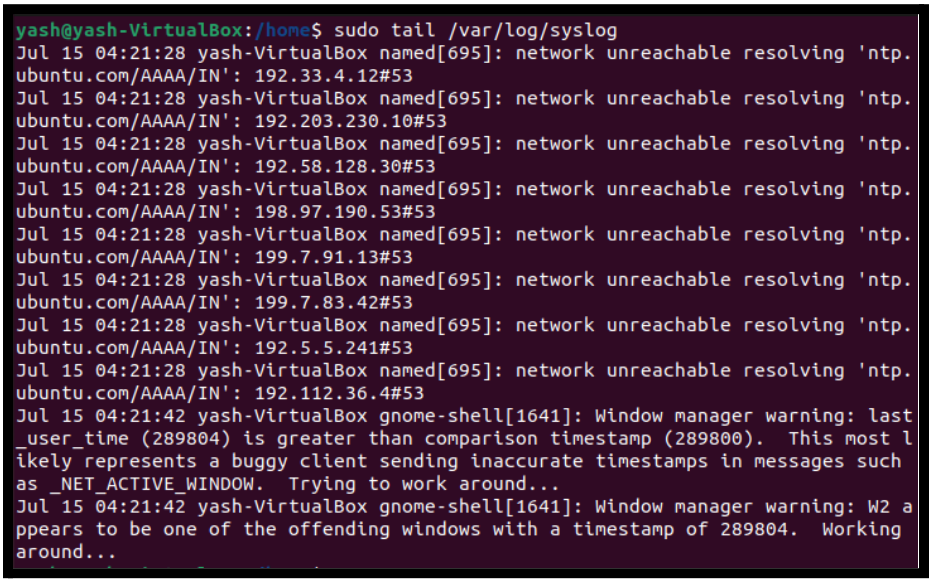
3.2.4 REQUIREMENT 4: DHCP LEASE TIME



4. Restart ISC-DHCP-SERVER: *sudo service isc-dhcp-server restart*

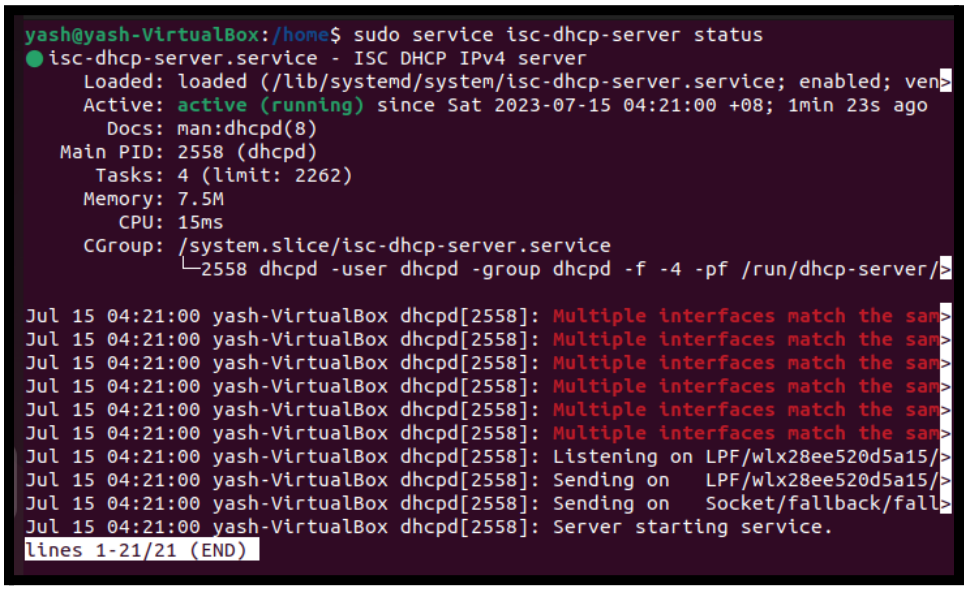
This command is used to interact with the system services in Ubuntu and restart is an argument for the ***"service"*** command which specifies that we want to restart the DHCP server.

5. Display log entries: *sudo tail /var/log/syslog*



"Tail" is the command-line utility that is used to display the last part of a file. This command is often used to view the most recent entries or updates in log files. ***"/var/log/syslog"*** is the path of the log file that is needed to be displayed. It specifies the location and name of the file which is ***"syslog"*** in the ***/var/log*** directory. The syslog file contains system log messages which include the information about various events, processes, and errors that occur on the system. By using this line of code, we are able to view the contents of the syslog to help on troubleshooting issues, monitor system activities and identify potential errors or problems which may have been logged.

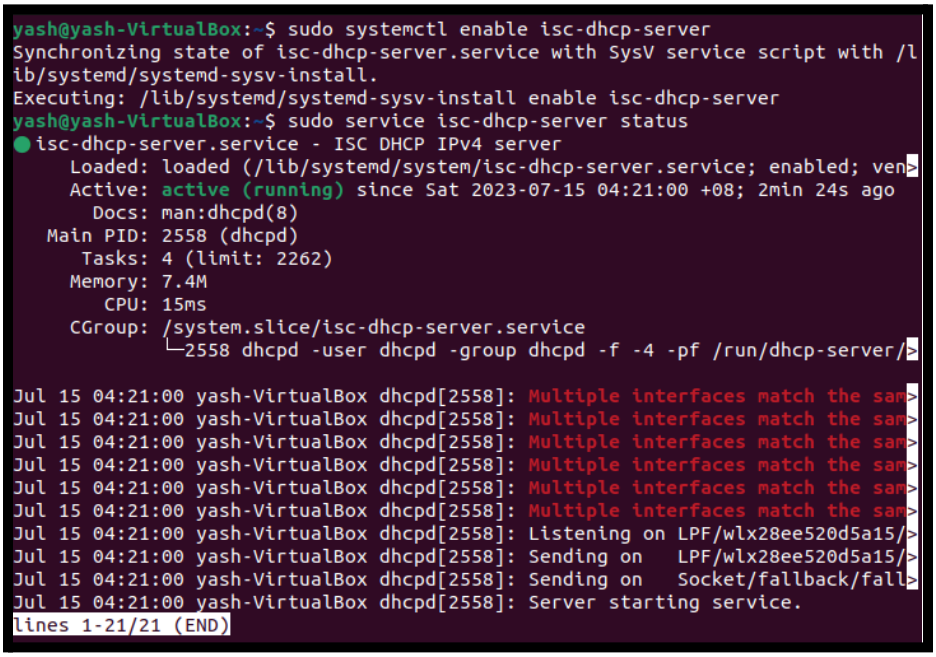
6. Status of ISC-DHCP-SERVER: *sudo service isc-dhcp-server status*



The output indicates whether the service is running, stopped or in any other state, which helps to verify if it is functioning correctly or helps to troubleshoot any issues related to the DHCP server.

3.2.5 REQUIREMENT 5: AUTO-START

7. Enable ISC DHCP SERVER: *sudo systemctl enable isc-dhcp-server*



The command ***"systemctl"*** is used to interact with systemd the initialization system and service manager in Ubuntu. "enable" is the argument of ***"systemctl"*** which specifies that we want to enable a service which means it will start automatically during system boot. This line of code shows that we are instructing the system to enable the ISC DHCP server service, after enabling, it automatically starts whenever the system boots up or is restarted. It ensures that the DHCP server is enabled and running consistently, providing IP address allocation and configuration services to devices on the network.

3.3 CONFIGURATION CHALLENGES AND WORKAROUNDS

The first issue we faced was with the *"**dhcpd.conf**"* file which had the wrong syntax. We then used a command to fix this issue which is ***"dhcpd -t -cf /path/to/dhcpd.conf"***. When the -t flag is specified, the server would only test the configuration file for correct syntax but does not perform any network operation. This can be used to test the new configuration file automatically before installing it. Other than that we also faced an issue where the DHCP keeps failing and gives an error line ***"not configured to listen on to any interfaces"***. We solved this problem by assigning the IPv4 address to the

wireless adapter.

4.0 WIRELESS ACCESS POINT

4.1 OVERVIEW

A wireless access point (WAP) is a networking device that establishes a wireless connection and subsequent communication between network devices and wired networks. It acts as a central hub for wireless devices like phones, tablets, and computers to connect with one another while at the same time granting them access to network resources available such as the internet or local network services. They are commonly found in areas where wireless connectivity is required because they are a convenient and flexible way for devices to connect to a network without the use of any physical wired connections.

4.2 IMPLEMENTATION DETAILS

1. **Install HostAPD: *sudo apt install hostapd***

```
yash@yash-VirtualBox:~$ sudo apt install hostapd
```

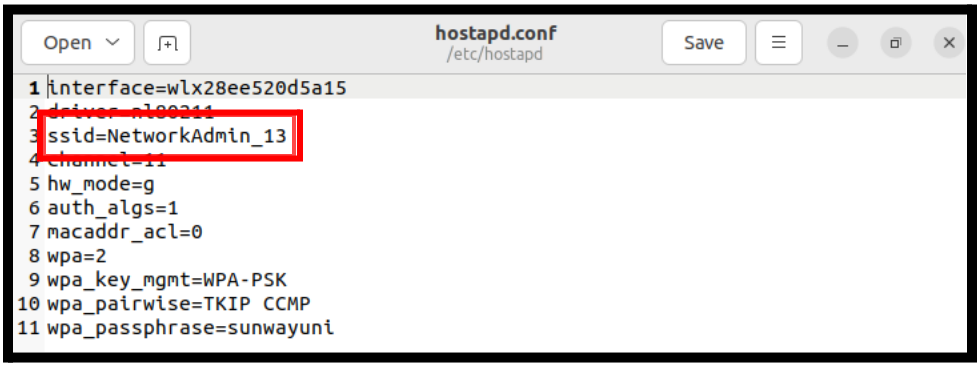
It is a command to install the ***"hostapd"*** package from the repositories onto our system. ***"hostapd"*** stands for Host Access Point Daemon, a software application that creates and manages a wireless network through a wireless adapter, basically turning our computer into a Wi-Fi access point.

2. Configure HostAPD: *sudo gedit /etc/hostapd/hostapd.conf*

```
yash@yash-VirtualBox:~$ sudo gedit /etc/hostapd/hostapd.conf
```

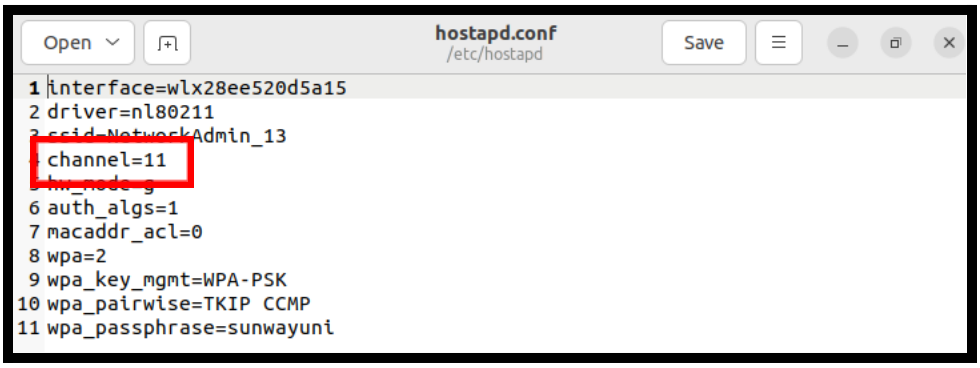
In the ***/etc/hostapd*** directory, the ***"hostapd.conf"*** file consists of various settings and parameters that define the behaviour and characteristics of our Wi-Fi access point created by ***"hostapd"***. By modifying said file's contents, we are able to configure the Wi-Fi access point to perfectly match our requirements.

4.2.1 REQUIREMENT 1: SSID



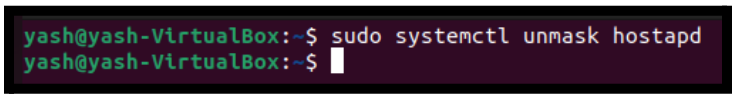
4.2.2 REQUIREMENT 2: WIRELESS CHANNEL

The channel parameter is used to specify the Wi-Fi channel on which the wireless access point (AP) will operate. The channel represents a specific frequency within the Wi-Fi spectrum and determines the range and interference characteristics of the wireless network.



4.2.3 REQUIREMENT 3: AUTHENTICATION

3. **Remove mask*: sudo systemctl unmask hostapd***

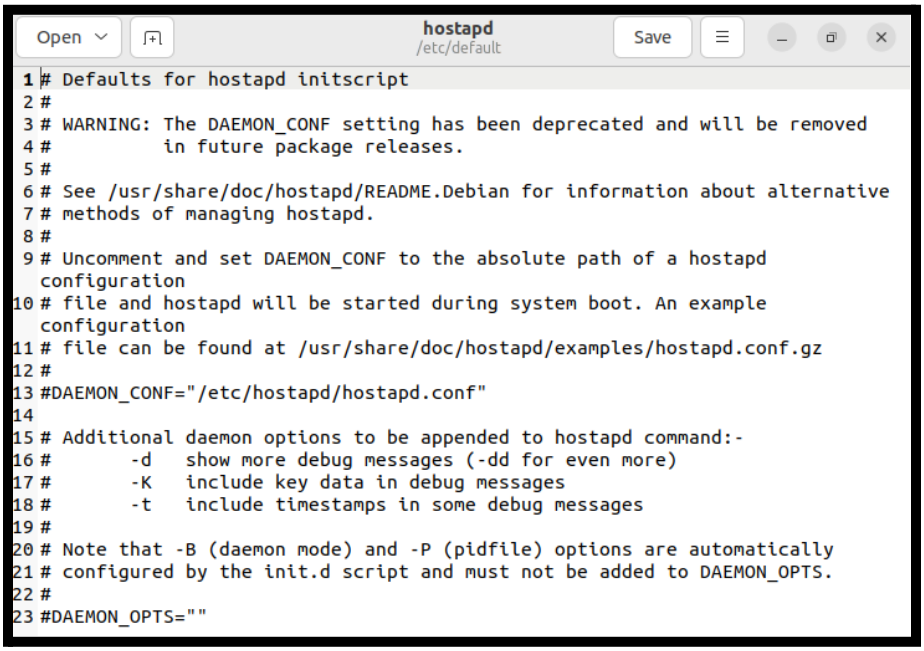


This line instructs the system to unmask the ***"hostapd"*** service unit file that allows ***"hostapd"*** service to be started enabled and managed using the "systemctl" command. ***"unmask"*** is an argument for the systemctl command that specifies wanting to remove

the mask on a service unit file. When masked, it prevents the service from being started or enabled.

4. Configure HostAPD: *sudo gedit /etc/default/hostapd*

```
yash@yash-VirtualBox:~/Desktop$ sudo gedit /etc/default/hostapd
```



This line opens the ***"hostapd"*** configuration file using the ***"gedit"*** text editor which allows users to view and modify configuration settings for the ***"hostapd"*** service.***/etc/default/hostapd*** is the path to the configuration file to be opened

5. Start HostAPD: *sudo systemctl start hostapd*

```
yash@yash-VirtualBox:~$ sudo systemctl start hostapd
```

"**Start**" is an argument of the "**systemctl**" command line which instructs the system to start the "**hostapd**". The command initiates the operation of the "**hostapd**" software which allows it to function as a WAP and provide WIFI services. After the service starts, the AP should be available for clients to connect to.

4.2.4 REQUIREMENT 4: SHELL SCRIPT

6. **Install ifstat: *sudo apt install ifstat***

```
yash@yash-VirtualBox:~$ sudo apt install ifstat
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
ifstat is already the newest version (1.1-8.2).
0 upgraded, 0 newly installed, 0 to remove and 1 not upgraded.
yash@yash-VirtualBox:~$
```

The command ***"sudo apt install ifstat"*** is used to install the ***"ifstat"*** package. The ***"ifstat"*** package is a command-line tool that provides network interface statistics. It can display information such as network bandwidth usage, packet counts, and error rates for each network interface on your system. Once the installation is complete, you can use the ***"ifstat"*** command in the terminal to start monitoring network statistics.

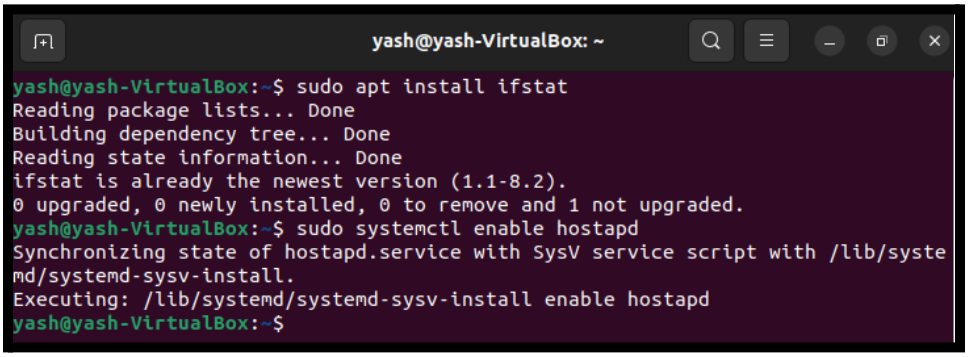
7. **Write a shell script to monitor WAP's activity: *sudo gedit Bandwidth.sh***

```
yash@yash-VirtualBox:~$ sudo gedit Bandwidth.sh
```

This command is used to open the ***"Bandwidth.sh"*** file in the text editor and modify it. The ***".sh"*** in this command denotes a shell script file, which contains a multiple of commands that can be executed in a specific order.

4.2.5 REQUIREMENT 5: AUTO-START

8. **Enable HostAPD: *sudo systemctl enable hostapd***



By using ***"sudo systemctl enable hostapd"***, the ***"hostapd"*** service starts automatically after the system reboot. This ensures that the Ubuntu system acts as a wireless access point consistently, even after the system restarts.

4.3 CONFIGURATION CHALLENGES AND WORKAROUNDS

One of the main challenges that we faced in order to successfully configure WAP was that Ubuntu did not recognize the external wireless adapter. Therefore, we had to install a driver in Ubuntu, but most of the drivers did not end up working. So, **Synaptic** was installed, and then a PPA was added to install the Realtek drivers. The URL belongs to a Launchpad user named **"kelebek333"** and he owned personal archives for all types of drivers, including ours. After that was settled, the wireless adapter was finally recognized. However, HostAPD could not start. After much confusion, we had to reboot the server, and then the HostAPD started working. Other than that, we also faced hardware issues as the USB wireless adapter had to be connected and disconnected multiple times to work. Lastly, for the authentication requirement, the passphrase ***"sunway"*** could not be used as WPA/WPA2 authentication requires a minimum of 8 characters. Hence, we decided to input ***"sunwayuni"*** as the access point's passphrase in order to bypass the error.

5.0 ADDITIONAL FEATURES

5.1 OVERVIEW

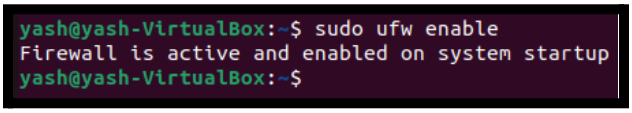
There were multiple aspects taken into consideration before adding additional features to our network. Security management is crucial for maintaining a secure and protected server environment. After much deliberation, we decided to add in a firewall, IDS, and IPS, which happen to be two key components of security management. A firewall acts as a barrier between the server and the outside world, controlling incoming and outgoing network traffic based on predefined rules. It helps protect the servers from unauthorised access, malicious activities, and network-based attacks. On the other hand, Suricata is a popular open-source IDS/IPS tool that helps detect and prevent network intrusions. It analyses network traffic in real-time, looking for patterns and signatures of known attacks or suspicious behaviour. Suricata can be deployed on the Ubuntu server to provide an additional layer of security.

5.2 IMPLEMENTATION DETAILS

5.2.1 FIREWALL

A firewall is a security mechanism that is designed to monitor and control incoming and outgoing network traffic based on predefined rules. The firewall acts as an obstacle between trusted internal networks and untrusted external networks like the Internet. Filtering and blocking malicious traffic while allowing legit traffic is a way firewalls help protect against unauthorised access, malware, and network threats, eventually enhancing the security of systems and data.

1. Enable UFW: *sudo ufw enable*



This line instructs the system to enable the UFW which is a CLI command and a user-friendly command-line tool for managing firewall rules in Ubuntu. When enabled, the firewall will enforce rules configured, regulating network traffic based on the policies.

This helps enhance the security of the system by controlling incoming and outgoing networks.

2. ***Allow outgoing connections:***

sudo ufw default deny incoming && sudo ufw default allow outgoing

```
yash@yash-VirtualBox:~$ sudo ufw default deny incoming && sudo ufw default allo
w outgoing
Default incoming policy changed to 'deny'
(be sure to update your rules accordingly)
Default outgoing policy changed to 'allow'
(be sure to update your rules accordingly)
yash@yash-VirtualBox:~$
```

"Sudo ufw default deny incoming" is not a valid standalone command, it is an incomplete representation of the firewall rule. It lacks the specific command needed to interpret and execute the firewall rules. ***"sudo ufw default allow outgoing"*** enables the UFW and sets default behavior for outgoing traffic. It configures the UFW specifically to allow all outgoing connections .

3. **Allow SSH connections:** ***sudo ufw allow ssh***

```
yash@yash-VirtualBox:~$ sudo ufw allow ssh
Skipping adding existing rule
Skipping adding existing rule (v6)
yash@yash-VirtualBox:~$
```

"ufw allow ssh" instructs the system to modify the UFW configuration to allow incoming SSH connections. SSH is the service or port which we want to allow incoming connections. SSh service is commonly used for secure remote access to systems. When this rule is added the firewall permits incoming SSH connections allowing remote access to the system using SSH. In this case, the output is shown as ***"Skipping adding existing rule"*** because SSH has already been established as a rule in the firewall.

4. **Allow outgoing connections through TCP ports:**

sudo ufw allow out 53,80,443/tcp

```
yash@yash-VirtualBox:~$ sudo ufw allow out 25,53,80,110,443/tcp
Rule added
Rule added (v6)
yash@yash-VirtualBox:~$
```

"53,80,443/tcp" is a list of specific TCP ports which indicates that we want to allow outgoing connections. These ports correspond to common services like SMTP , DNS and HTTP. This line code instructs the system to add outgoing firewall rules that allow connections on the specified TCP ports. This allows outgoing connections, enabling communication with services.

5. Allow outgoing communication ports for DNS resolution and DHCP requests:

sudo ufw allow out 53,67,68/udp

```
yash@yash-VirtualBox:~$ sudo ufw allow out 25,53,80,110,443/tcp
Rule added
Rule added (v6)
yash@yash-VirtualBox:~$ sudo ufw allow out 53,67,68/udp
Skipping adding existing rule
Skipping adding existing rule (v6)
```

"53,67,68/udp" is similar to previously but it corresponds to common services like DNS (53) and DHCP(67 and 68). This command line instructs the system on the outgoing firewall rules like previously stated but now to the specified UDP ports. This enables outgoing communication ports 53, 67 and 68 which facilitates DNS resolution and DHCP requests.

6. Disabling and enabling UFW: *sudo ufw disable && sudo ufw enable*

```
yash@yash-VirtualBox:~$ sudo ufw disable && sudo ufw enable
Firewall stopped and disabled on system startup
Firewall is active and enabled on system startup
```

The command ***"ufw disable"*** disables the UWF and turns off the firewall stopping it from enforcing any configured rules. ***"sudo ufw enable"*** on the other hand, enables UFW which turns on the firewall and allows the enforcement of configured rules. Running both command lines simultaneously effectively disables and enables the UFW.

The sequence is useful especially when temporarily disabling for specific troubleshooting or testing purposes and enabling it back when the task is completed.

7. Display status of UWF: *sudo ufw status verbose*

```
yash@yash-VirtualBox:~$ sudo ufw status verbose
Status: active
Logging: on (low)
Default: deny (incoming), allow (outgoing), disabled (routed)
New profiles: skip

To                         Action      From
--                         ------      ----
22/tcp                     ALLOW IN    Anywhere
22/tcp (v6)                ALLOW IN    Anywhere (v6)

53,80,443/tcp              ALLOW OUT   Anywhere
53,67,68/udp               ALLOW OUT   Anywhere
53,80,443/tcp (v6)         ALLOW OUT   Anywhere (v6)
53,67,68/udp (v6)          ALLOW OUT   Anywhere (v6)
```

"status verbose" are both arguments of the UWF in which status is used to display the current status of the firewall by providing information about the enabled or disabled state of the firewall and the configured rules. Running this command line it instructs the system to display the detailed status and configuration of the UFW. it provides information if the firewall is enabled or disabled with a list of configured rules and their corresponding ports or IP addresses. This helps to understand the current state of the firewall and verify rules that have been applied for network traffic filtering and protection.

5.2.2 IDS (Intrusion Detection System)

IDS is a network security feature implemented by administrators to detect and respond to unauthorized or malicious activities occurring within a network. It qualifies as a security technology due to its ability to monitor and analyze network traffic or system logs for suspicious patterns or signatures. Once it detects any form of questionable traffic, IDS alerts the network administrator or takes automated procedures permitted by them to prohibit potential intrusion or security breach. Our IDS software is Suricata, a type of network-based Intrusion Detection System ***(NIDS)*** that is known for its capabilities to handle network-based attacks even in high-speed network traffic environments.

1. **Download preparation:** ***sudo add-apt-repository ppa:oisf/suricata-stable***

```
yash@yash-VirtualBox:~$ sudo add-apt-repository ppa:oisf/suricata-stable
```

"Add-apt repository" adds a new software repository to our system's package manager and ***"ppa:oisf/suricata-stable"*** is the Personal Package Archive repository URL for Suricata Stable being added here. Running the command means we are able to install and update Suricata through the package manager in the following step.

2. **Install Suricata:** ***sudo apt install suricata***

```
yash@yash-VirtualBox:~$ sudo apt install suricata
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
suricata is already the newest version (1:6.0.13-0ubuntu0).
0 upgraded, 0 newly installed, 0 to remove and 1 not upgraded.
yash@yash-VirtualBox:~$
```

The command orders the package manager, ***"apt"*** to install Suricata and its subsequent dependencies from the operating system's repositories into the actual system where we can start configuring it to our likings.

3. **Enable Suricata service:** ***sudo systemctl enable suricata.service***

```
yash@yash-VirtualBox:~$ sudo systemctl enable suricata.service
suricata.service is not a native service, redirecting to systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable suricata
```

Just like before, Suricata will start automatically whenever the system boots up after being affected by the phrase ***"enable"*** in the command, therefore no manual intervention is needed for it to begin its services.

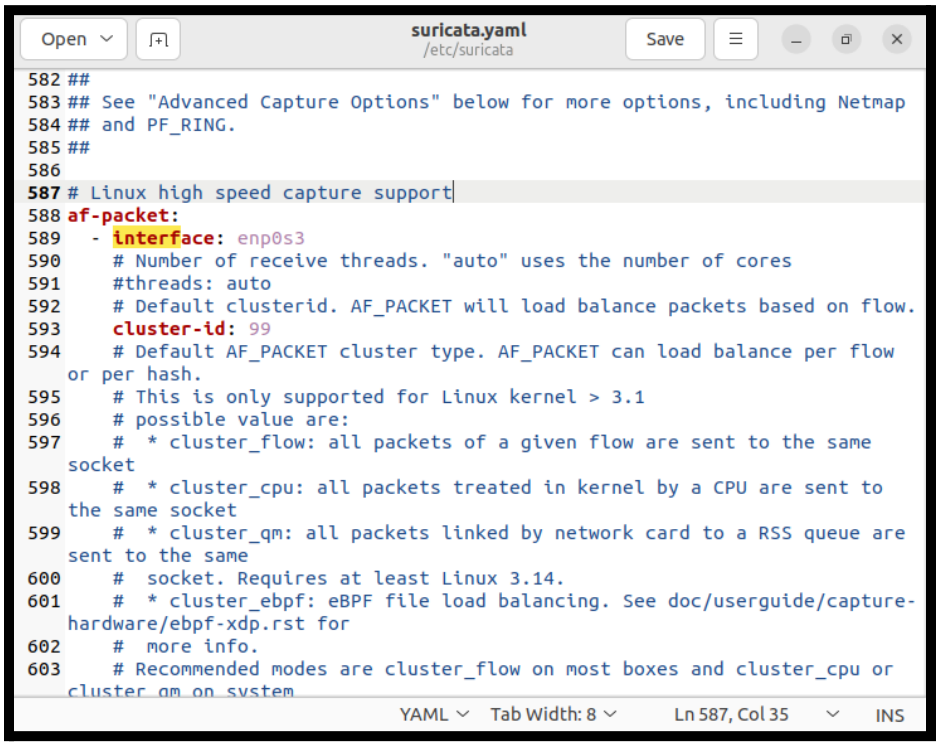
4. **Stop Suricata service:** ***sudo systemctl stop suricata.service***

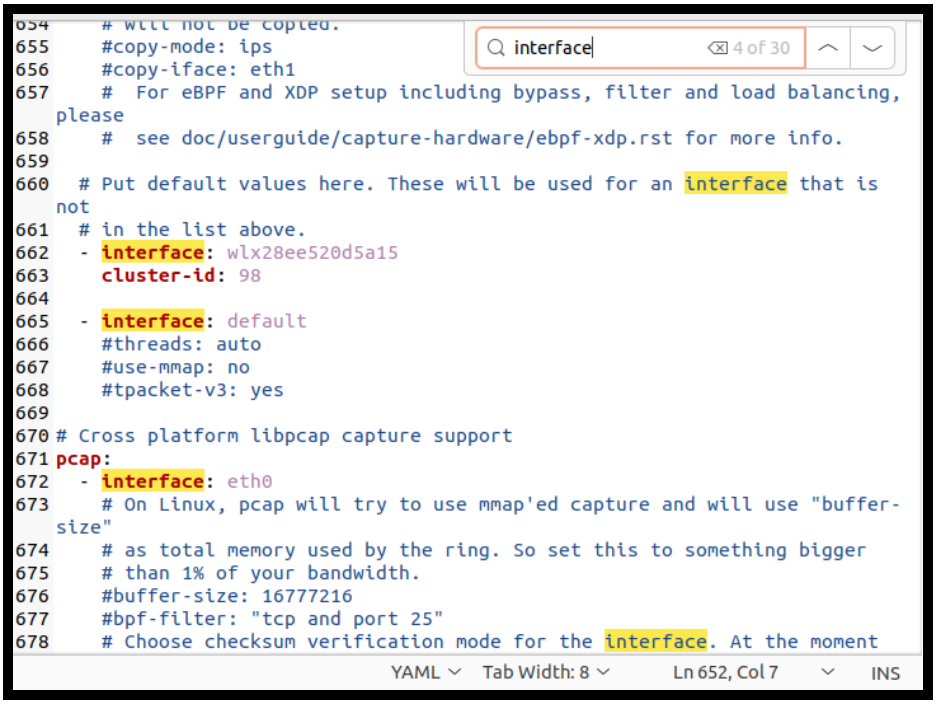
```
yash@yash-VirtualBox:~$ sudo systemctl stop suricata.service
```

"Stop" is a subcommand of ***"systemctl"*** that stops a service. In other words, this command halts Suricata execution as a process and all of its functionalities to the web server. In scenarios where a service needs to be configured or troubleshooted, stopping its operation is the best course of action before taking any further actions.

5. Modify Suricata's configurations: *sudo gedit /etc/suricata/suricata.yaml*

```
yash@yash-VirtualBox:~/Desktop$ sudo gedit /etc/suricata/suricata.yaml
```

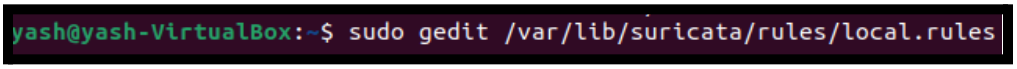


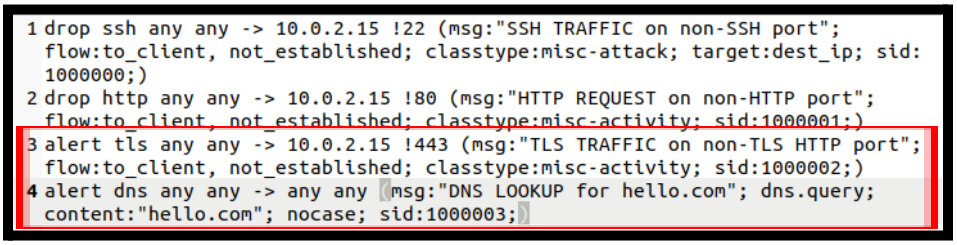


There is a section in this file that specifies the network interfaces that will be monitored by Suricata when it is active. By appending or deleting any information here, we can effectively tailor Suricata's range of protection to only include the network interfaces that we planned on using.

6. Specifies additional rules or edit the existing ones:

sudo gedit /var/lib/suricata/rules/local.rules





This command is a user-defined rule file used by Suricata to specify additional custom intrusion detection rules or modify the existing ones.

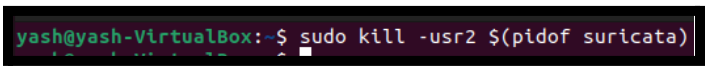
7. **Enable live reload of Suricata's configurations and insert *"local.rules"* under rule-files:**

sudo gedit /etc/suricata/suricata.yaml

By setting ***"rule-reload"*** to ***"true"***, we are able to send ***"SIGUSR2"*** signal to Suricata and it will reload any changed rules into memory. This tiny change to Suricata's configurations will allow us to add, remove, and edit its rules without restarting its process.

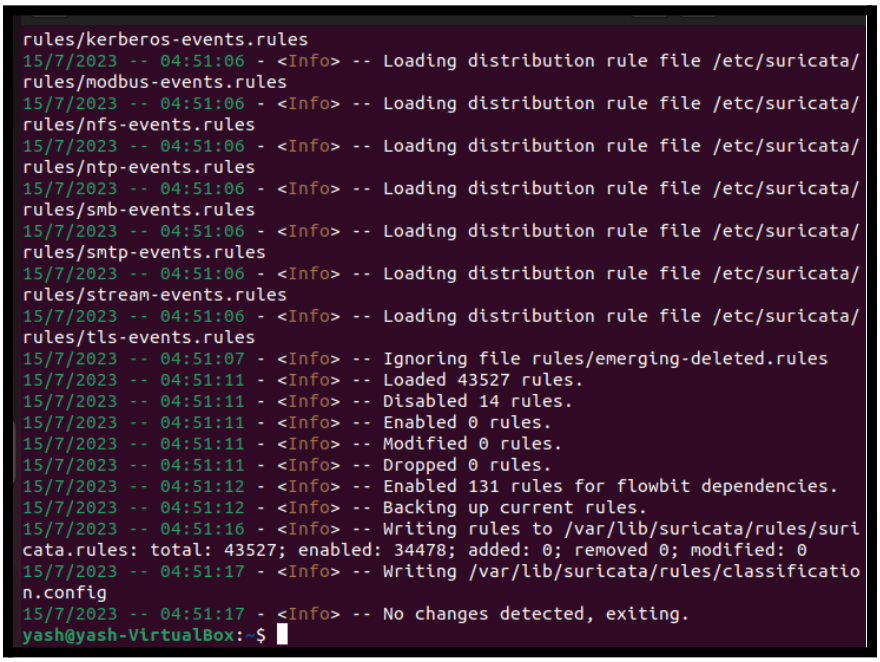
This command is used to modify the content of the ***"suricata.yaml"*** file to customize the configuration of Suricata according to the specific requirements and network environment. Inside the ***"suricata.yaml"*** file, the user is able to define settings such as network interface configuration, logging options, rule management, output modules, and other parameters that influence Suricata's behaviour. In order to view the rule-files, the command ***"sudo suricata - -list-rule-files"*** can be used.

8. **Trigger the live-reload:** ***sudo kill -usr2 $(pidof suricata)***



"Kill" sends signals to Linux processes while ***"-usr2"*** is the ***"SIGUSR2"*** signal handled by Suricata process to trigger specific actions such as configuration reloading. ***"$(pidof suricata)"*** is a command substitution that retrieves the Process ID ***(PID)*** of Suricata. When used in conjunction with one another, we are essentially sending the ***"SIGUSR2"*** signal to the Suricata process that is identified by its ***PID*** to trigger a live reload of the freshly made configuration.

9. Update Suricata: *sudo suricata-update*



Connects Suricata with its update server that hosts the latest version of rules and associated files to download them. The update can consist of knowledge regarding emerging threats, known attack patterns, and vulnerability signature which is extremely vital for the service to adequately defend the network.

10. Test run and research Suricata: *sudo suricata -T -c /etc/suricata/suricata.yaml -v*

```
yash@yash-VirtualBox:~$ sudo suricata -T -c /etc/suricata/suricata.yaml -v
```

"suricata" is a command-line for running Suricata, our IDS and ***"-T"*** is an option of said command that instructs a test run of Suricata without actually starting the monitoring process. Continuing the command with ***"-c /etc/suricata/suricata.yaml"*** directs the service to the configuration file that it should use for its test run. ***"-v"*** puts the test run in verbose mode, a mode that generates a more descriptive output during the test run. All in all, this command is asking Suricata to do a test run with the designated configuration file and produce a more detailed final result for said test run.

11. **Start Suricata service: *sudo systemctl start suricata.service***

```
yash@yash-VirtualBox:~$ sudo systemctl start suricata.service
```

Opposite of ***"stop"***, the command ***"start"*** is an action parameter for ***"systemctl"*** that indicates to ***systemd*** there are certain services users want to start which is Suricata in our situation.

12. **Check Suricata status: *sudo systemctl status suricata.service***

```
yash@yash-VirtualBox:~$ sudo systemctl status suricata.service
● suricata.service - LSB: Next Generation IDS/IPS
     Loaded: loaded (/etc/init.d/suricata; generated)
     Active: active (running) since Sat 2023-07-15 04:53:03 +08; 10s ago
       Docs: man:systemd-sysv-generator(8)
    Process: 4962 ExecStart=/etc/init.d/suricata start (code=exited, status=0/>
      Tasks: 1 (limit: 2262)
     Memory: 259.8M
        CPU: 10.106s
     CGroup: /system.slice/suricata.service
             └─4968 /usr/bin/suricata -c /etc/suricata/suricata.yaml --pidfile>

Jul 15 04:53:03 yash-VirtualBox systemd[1]: Starting LSB: Next Generation IDS/>
Jul 15 04:53:03 yash-VirtualBox suricata[4962]: Starting suricata in IPS (nfqu>
Jul 15 04:53:03 yash-VirtualBox systemd[1]: Started LSB: Next Generation IDS/I>
lines 1-14/14 (END)
```

Per usual, the command ***"status"*** is used after each service initialization to ensure the service is actually active and not inactive due to suffering from some form of error.

13. **Testing the IDS: *curl http://testmynids.org/uid/index.html***

```
yash@yash-VirtualBox:~$ curl http://testmynids.org/uid/index.html
uid=0(root) gid=0(root) groups=0(root)
yash@yash-VirtualBox:~$
```

Command ***"curl"*** is a CLI tool used for making HTTP requests and interacting with target web services through URLs like ***"http://testmynids.org/uid/index.html"***. When run, ***"curl"*** will send a HTTP GET request to the mentioned URL and the web server at ***"testmynids.org"*** will receive and process the request in order to return the content of the ***"index.html***" file located in the ***"uid"*** directory to it. If the URL leads to a questionable site, the IDS will detect it and alert the administrator by marking the instance down in its log.

There are two methods to examine the log file for suspicious network traffic:

14. **Method 1:**

Filter fast.log file: *sudo grep 2100498 /var/log/suricata/fast.log*

```
yash@yash-VirtualBox:~$ grep 2100498 /var/log/suricata/fast.log
07/09/2023-06:04:12.673065  [**] [1:2100498:7] GPL ATTACK_RESPONSE id check ret
urned root [**] [Classification: Potentially Bad Traffic] [Priority: 2] {TCP} 6
5.8.11.2:80 -> 10.0.2.15:33284
07/09/2023-06:45:27.047918  [**] [1:2100498:7] GPL ATTACK_RESPONSE id check ret
urned root [**] [Classification: Potentially Bad Traffic] [Priority: 2] {TCP} 6
5.8.11.56:80 -> 10.0.2.15:43122
07/09/2023-07:26:43.363223  [**] [1:2100498:7] GPL ATTACK_RESPONSE id check ret
urned root [**] [Classification: Potentially Bad Traffic] [Priority: 2] {TCP} 6
5.8.11.118:80 -> 10.0.2.15:36876
07/09/2023-07:27:17.530705  [**] [1:2100498:7] GPL ATTACK_RESPONSE id check ret
urned root [**] [Classification: Potentially Bad Traffic] [Priority: 2] {TCP} 6
5.8.11.2:80 -> 10.0.2.15:48466
07/09/2023-07:29:41.138188  [**] [1:2100498:7] GPL ATTACK_RESPONSE id check ret
urned root [**] [Classification: Potentially Bad Traffic] [Priority: 2] {TCP} 6
5.8.11.118:80 -> 10.0.2.15:55598
07/09/2023-07:39:01.879870  [**] [1:2100498:7] GPL ATTACK_RESPONSE id check ret
urned root [**] [Classification: Potentially Bad Traffic] [Priority: 2] {TCP} 6
5.8.11.2:80 -> 10.0.2.15:55254
07/09/2023-07:39:20.338040  [**] [1:2100498:7] GPL ATTACK_RESPONSE id check ret
urned root [**] [Classification: Potentially Bad Traffic] [Priority: 2] {TCP} 6
5.8.11.118:80 -> 10.0.2.15:42318
07/09/2023-07:42:51.785506  [**] [1:2100498:7] GPL ATTACK_RESPONSE id check ret
urned root [**] [Classification: Potentially Bad Traffic] [Priority: 2] {TCP} 6
5.8.11.88:80 -> 10.0.2.15:60700
07/09/2023-07:53:04.766789  [**] [1:2100498:7] GPL ATTACK_RESPONSE id check ret
urned root [**] [Classification: Potentially Bad Traffic] [Priority: 2] {TCP} 6
```

Command ***"grep"*** is a text pattern searching tool used on text files to pinpoint the exact line of text the users are looking for. In our case, ***"grep"*** is examining the "***fast.log"*** file positioned in the ***/var/log/suricata*** directory for text ***"2100498"***.

15. Method 2:

Install jq package: ***sudo apt install jq***

```
yash@yash-VirtualBox:~$ sudo apt install jq
```

The command instals the ***"jq"*** package from the available repositories onto our system. ***"Jq"*** is a command-line JavaScript Object Notation ***(JSON)*** that enables users to work with JSON data as it can parse, manipulate, and extract data from JSON files using simple and expressive syntax.

Filter Suricata's JSON log file: ***jq 'select(.alert .signature_id==2100498)' /var/log/suricata/eve.json***

```
yash@yash-VirtualBox:~$ jq 'select(.alert .signature_id==2100498)' /var/log/sur
icata/eve.json
{
  "timestamp": "2023-07-09T06:04:12.673065+0800",
  "flow_id": 449540760854462,
  "in_iface": "enp0s3",
  "event_type": "alert",
  "src_ip": "65.8.11.2",
  "src_port": 80,
  "dest_ip": "10.0.2.15",
  "dest_port": 33284,
  "proto": "TCP",
  "alert": {
    "action": "allowed",
    "gid": 1,
    "signature_id": 2100498,
    "rev": 7,
    "signature": "GPL ATTACK_RESPONSE id check returned root",
    "category": "Potentially Bad Traffic",
    "severity": 2,
    "metadata": {
      "created_at": [
        "2010_09_23"
      ],
      "updated_at": [
        "2010_09_23"
      ]
    }
  },
```

"Jq" being a command-line tool for users to interact with JSON files offers a filter expression that comes in the form of ***"select(.alert .signature_id==)"*** enclosed in

single quotation. By completing the expression with ***"2100498"*** and setting the target file as ***"eve.json"*** located under directory ***/var/log/suricata***, the command filters the ***"eve.json"*** file for JSON objects that match the ***"signature_id"*** field which is ***"2100498"*** and outputs it as an ***"alert"*** object.

16. Tests if alert for DNS server is active and the response time: *nslookup www.hello.com*

```
yash@yash-VirtualBox:~$ nslookup www.hello.com
Server:         192.168.13.20
Address:        192.168.13.20#53

Name:   www.hello.com
Address: 192.168.13.10

yash@yash-VirtualBox:~$
```

This command is used to test the availability and response time of the specified URL. The maximum time limit ensures that the request does not hang indefinitely, allowing for quick testing of web server responsiveness. It can be useful in assessing the performance and availability of web services and monitoring the behavior of the IPS when interacting with external resources.

17. Extracts information from the log files: *jq 'select(.alert .signature_id==1000003)' /var/log/suricata/eve.json*

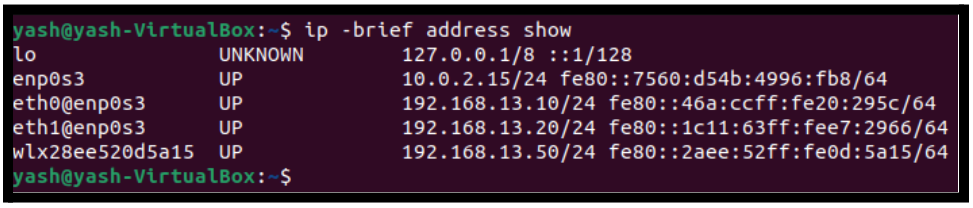

```
"timestamp": "2023-07-11T13:12:51.892116+0800",
"flow_id": 439576561949623,
"event_type": "alert",
"src_ip": "192.168.13.20",
"src_port": 42808,
"dest_ip": "192.168.13.20",
"dest_port": 53,
"proto": "UDP",
"community_id": "1:SMzYHpl0sfQsNVV8oKO+2FsTruo=",
"tx_id": 3,
"alert": {
  "action": "allowed",
  "gid": 1,
  "signature_id": 1000003,
  "rev": 0,
  "signature": "DNS LOOKUP for hello.com",
  "category": "",
  "severity": 3
},
"dns": {
  "query": [
    {
      "type": "query",
      "id": 54912,
      "rrname": "ntp.ubuntu.com.hello.com",
      "rrtype": "AAAA",
      "tx_id": 3,
```

This command is used for or extracting specific information from Suricata's log files based on certain criteria. In this case, it filters and retrieves log events that have an ID which is 1000003. By focusing on specific events, you can gain insights into network intrusion attempts or specific security events identified by Suricata.

5.2.3 IPS (Intrusion Prevention System)

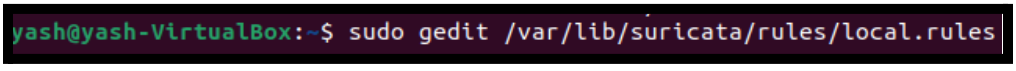
As the IDS helps by alerting potential attacks, the IPS empowers this system by actively preventing and blocking such attacks on network traffic. Using the signature-based detection, the IPS can identify known attack patterns and take immediate action to block malicious traffic. Hence, we decided to also implement the IPS as we strongly believe that a network should have multiple layers of security against any ill-intentioned and suspicious traffic that could possibly induce a massive downtime for our network.

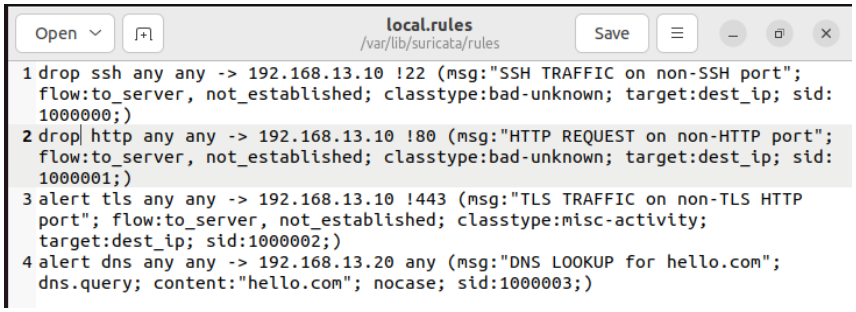
1. **Shows the ip address:** ***ip -brief address show***



The command **"ip"** is a versatile command-line tool used for managing network interfaces, routing, and various aspects of network configuration. It shows both IPv4 and IPv6 addresses associated with the network interfaces, providing a complete overview of the IP configuration

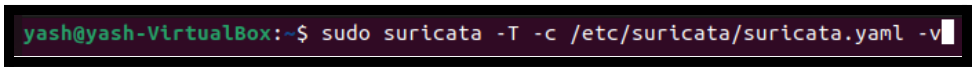
2. **Specifies additional rules or edit the existing ones:** ***sudo gedit /var/lib/suricata/rules/local.rules***





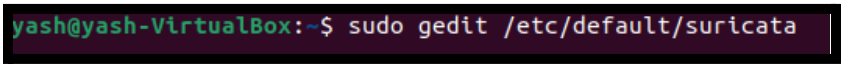
This command is a user-defined rule file used by Suricata to specify additional custom intrusion detection rules or modify the existing ones.

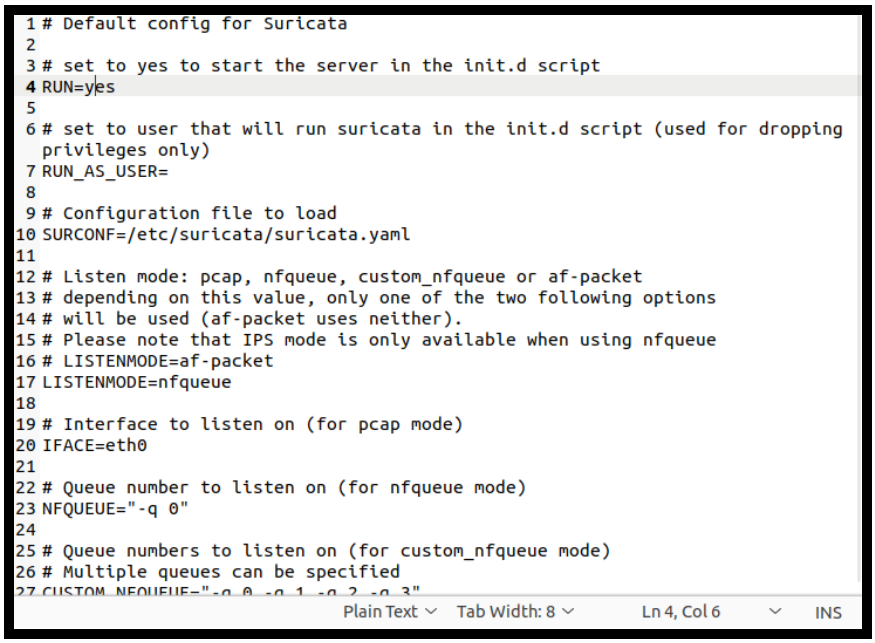
3. Configures test on Suricata and validates the file: *sudo suricata -T -c /etc/suricata/suricata.yaml -v*



This command is used to perform a configuration test on Suricata and validate the specified Suricata configuration file. This helps to prevent potential issues and ensures that Suricata functions effectively in detecting and preventing network intrusions.

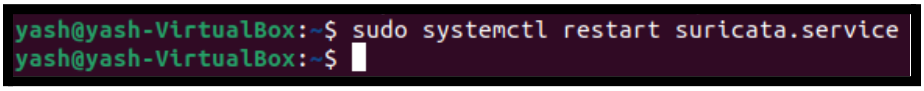
4. Opens and edits Suricata: *sudo gedit /etc/default/suricata*





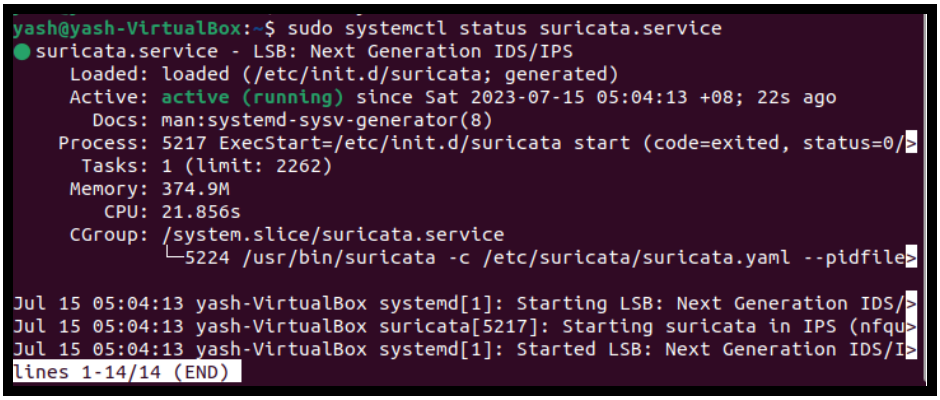
This command is used to open and edit the “suricata” default configuration file and then modify the value of the ***“LISTENMODE"*** parameter. Modifying the ***“LISTENMODE”*** parameter to “nfqueue” is helpful for certain network environments or specific use cases where advanced traffic analysis or interaction with other network tools is required.

5. Restarts Suricata: *sudo systemctl restart suricata.service*



This command is used to restart the Suricata service.

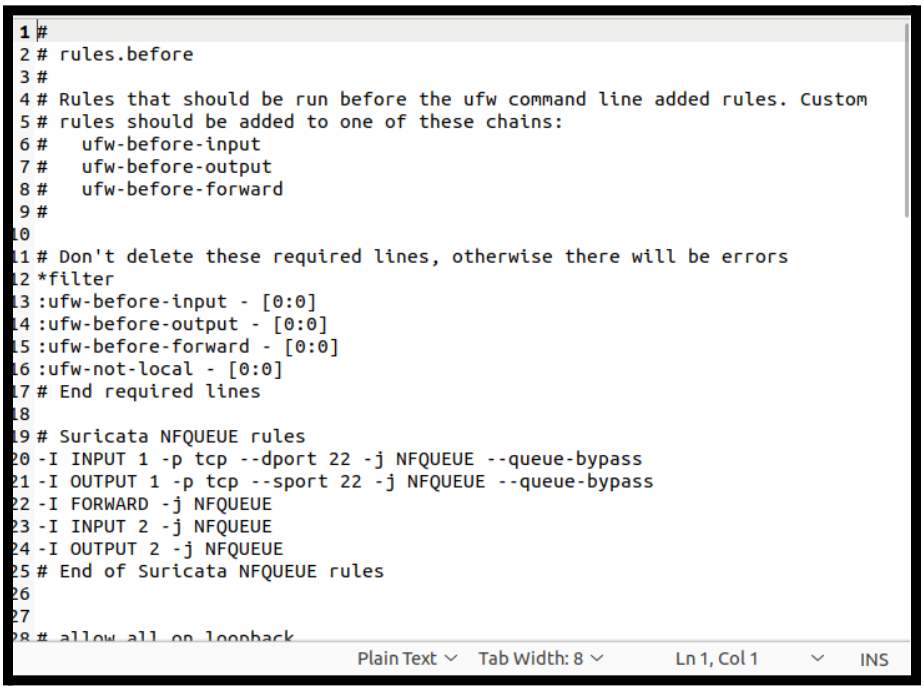
6. Monitors the health and operations of the service: *sudo systemctl status suricata.service*



This command allows monitoring of the service's health and operational state. It provides information about whether Suricata is running correctly, any errors that may have occurred, and the resource utilization of the service. This is useful for troubleshooting and ensuring the continuous operation of the IPS in Ubuntu.

7. Open and edit the file: *sudo gedit /etc/ufw/before.rules*

```
yash@yash-VirtualBox:~$ sudo gedit /etc/ufw/before.rules
```



This command is used to open and edit the ***"before-rules"*** file. This ensures tailoring of the firewall configuration to meet the specific IPS needs, enabling advanced network traffic filtering and intrusion prevention capabilities.

8. Restart UFW: *sudo systemctl restart ufw.service*

```
yash@yash-VirtualBox:~$ sudo systemctl restart ufw.service
yash@yash-VirtualBox:~$
```

This command is used to stop the running UFW service if it is currently active. Once the service is stopped, it frees up the system resources and terminates any active UFW processes. Then the command will start the UFW service again, initiating a fresh instance of UFW with the updated configuration. The UFW service reads the configuration files together with the rules and settings defined in ***/etc/ufw***. Once the

service restarts successfully UFW applies the firewall rules and begins enforcing the network security policies.

9. Debug and capture data within Suricata: ***Test: sudo kill -usr2 $(pidof suricata)***

```
yash@yash-VirtualBox:~$ sudo kill -usr2 $(pidof suricata)
yash@yash-VirtualBox:~$
```

This command is used during testing or debugging scenarios to trigger specific behavior or capture specific data within the Suricata IPS. The exact effect of sending the ***"USR2"*** signal to Suricata depends on how it is implemented and configured in the specific deployment environment.

10. **Display Suricata status to determine the activation of IPS:**

sudo service suricata status

```
● suricata.service - LSB: Next Generation IDS/IPS
     Loaded: loaded (/etc/init.d/suricata; generated)
     Active: active (running) since Sat 2023-07-15 23:53:33 +08; 2h 46min ago
       Docs: man:systemd-sysv-generator(8)
    Process: 1005 ExecStart=/etc/init.d/suricata start (code=exited, status=0/>
      Tasks: 9 (limit: 2262)
     Memory: 437.2M
        CPU: 4min 12.702s
     CGroup: /system.slice/suricata.service
             └─1070 /usr/bin/suricata -c /etc/suricata/suricata.yaml --pidfile>

Jul 15 23:53:33 yash-VirtualBox systemd[1]: Starting LSB: Next Generation IDS/>
Jul 15 23:53:33 yash-VirtualBox suricata[1005]: Starting suricata in IPS (nfqu>
Jul 15 23:53:33 yash-VirtualBox systemd[1]: Started LSB: Next Generation IDS/I>
~
```

```
on IDS/IPS
ta; generated)
 2023-07-15 23:53:33 +08; 2h 46min ago
(8)
/suricata start (code=exited, status=0/SUCCESS)



vice
c /etc/suricata/suricata.yaml --pidfile /var/run/suricata.pid -q 0 -D -vvv

[1]: Starting LSB: Next Generation IDS/IPS...
a[1005]: Starting suricata in IPS (nfqueue) mode... done.
[1]: Started LSB: Next Generation IDS/IPS.
~
```

5.3 CONFIGURATION CHALLENGES AND WORKAROUNDS

When testing out the active IPS, the packets were not dropped as the signature suggested. The problem could stem from the signature itself. Even so, it does not affect the IDS since it is working according to the signatures we have stated in the ***"local.rules"*** file, alerting administrators of any HTTP requests and DNS queries made.

6.0 LESSONS LEARNT

Throughout this assignment on Linux-based server configuration, we have learned several valuable lessons. One of the key lessons was the importance of conducting thorough research before embarking on any task, as well as the significance of proper documentation to ensure backup and reference. For instance, when our Ubuntu server crashed on multiple occasions, we had to start from scratch because we neglected to document our progress initially. If we had documented from the beginning, it would have greatly alleviated our problems.

Furthermore, effective communication skills among team members proved to be crucial during this assignment. As we encountered various issues, clear communication allowed us to understand what actions were taken and what might have caused the problems. Without proper communication, it becomes challenging to identify and troubleshoot errors effectively.

Additionally, this assignment highlighted the importance of time management. Initially, balancing extracurricular activities and the assignment caused significant stress for most of us. However, once we recognized our struggle with time management, we organised our tasks by allocating specific time slots and creating schedules for the assignment. This approach greatly helped us in completing the assignment on time.

These are some of the lessons we have learned through this assignment, and we will certainly keep them in mind and apply them in future assignments.