

# CCT College Dublin

## Assessment Cover Page

---

<b>Module Title:</b>	Networking & Virtualisation
<b>Assessment Title:</b>	Proof of Concept Linux Virtual Network Project
<b>Lecturer Name:</b>	Michael Weiss
<b>Student Full Name:</b>	Publio Lima de Mello Filho
<b>Student Number:</b>	2024028
<b>Assessment Due Date:</b>	29/09/2024
<b>Date of Submission:</b>	29/09/2024

---

### Declaration

By submitting this assessment, I confirm that I have read the CCT policy on Academic Misconduct and understand the implications of submitting work that is not my own or does not appropriately reference material taken from a third party or other source. I declare it to be my own work and that all material from third parties has been appropriately referenced. I further confirm that this work has not previously been submitted for assessment by myself or someone else in CCT College Dublin or any other higher education institution.

## **Summary**

The project aims to establish a virtualized network infrastructure composed by two virtual machines: a Domain Controller Server and a Web server. Through a series of systematic steps, the network infrastructure is set up to facilitate efficient communication, centralized management, and reliable service hosting.

### **Important Actions:**

**Installation of Virtual Machine:** Two virtual machines are constructed, each operating a Linux Server (ubuntuserver) and a Linux Client (ubuntuclient) VM. The ubuntuserver hosts a simple DCC web page, while the ubuntuclient tests connectivity and access to the web server. Additionally, remote connectivity via SSH and basic security configurations are implemented within the Linux environment.

**Server Renaming:** To provide consistent identification across the network, the UbuntuServer and UbuntuClient are renamed, respectively.

**Static IP Address Assignment:** In order to guarantee reliable network connectivity, static IP addresses are assigned to each server. IP addresses, default gateways, and DNS server addresses must all be specified in accordance with the guidelines provided during configuration.

**Testing for Connectivity:** The PING command is used to confirm that there is network connectivity between the two servers. This test verifies correct configuration and makes sure packets can move over the network.

## **Installation of Virtual Machine**

Using virtualization software, two virtual machines have been created to establish a robust network infrastructure. The first virtual machine, designated as ubuntuserver, functions as the Domain Controller Server. It is responsible for managing user accounts, security policies, and network resources. This server plays a crucial role in centralizing authentication and authorization services, facilitating streamlined network management. The second virtual machine, called ubuntuclient, is configured to operate as the web server, tasked with hosting and serving web applications and content to clients accessing the network. Both virtual machines are running Ubuntu, ensuring compatibility and interoperability within the network environment. Together, they are pivotal components of the infrastructure, working synergistically to deliver reliable services and support seamless communication across the network.

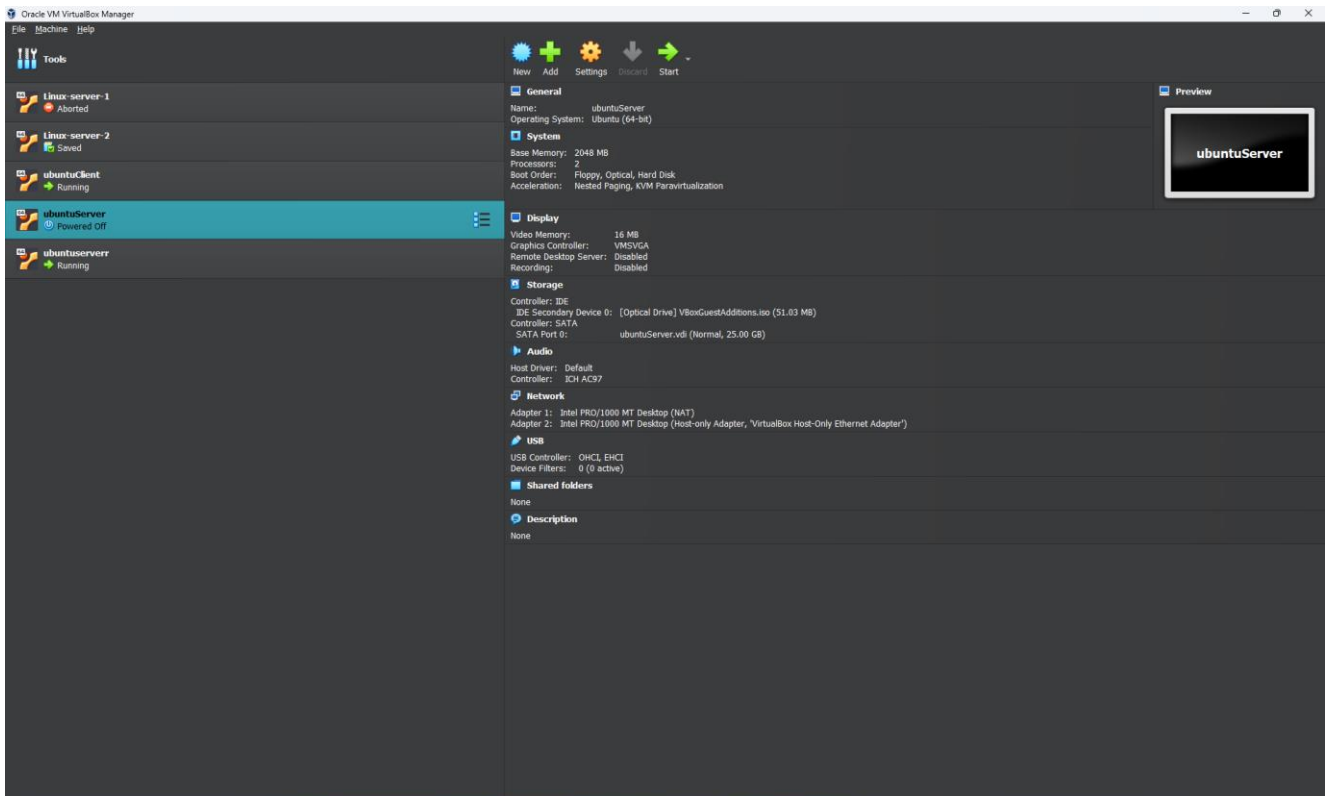


Image 1: First Virtual machine name ubuntuServer

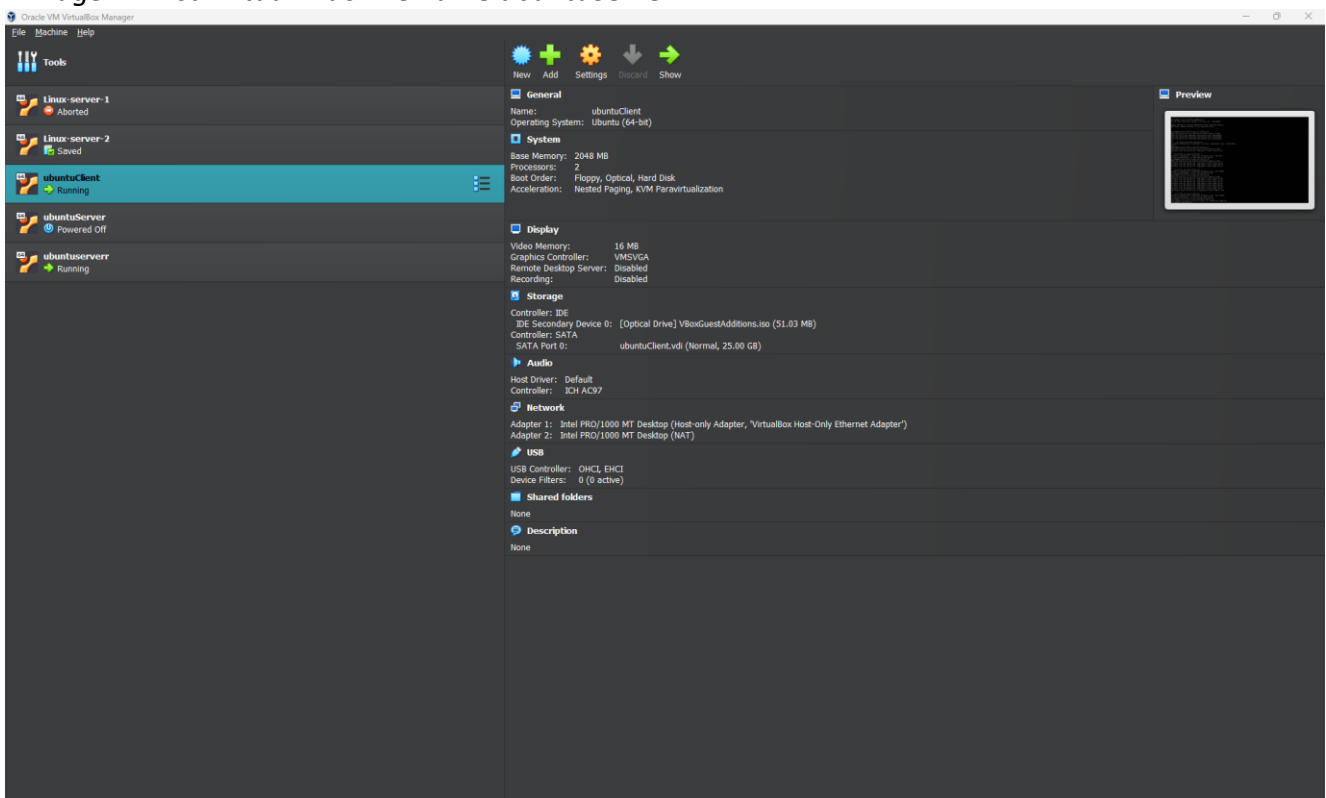


Image 2: Second Virtual machine named ubuntuClient

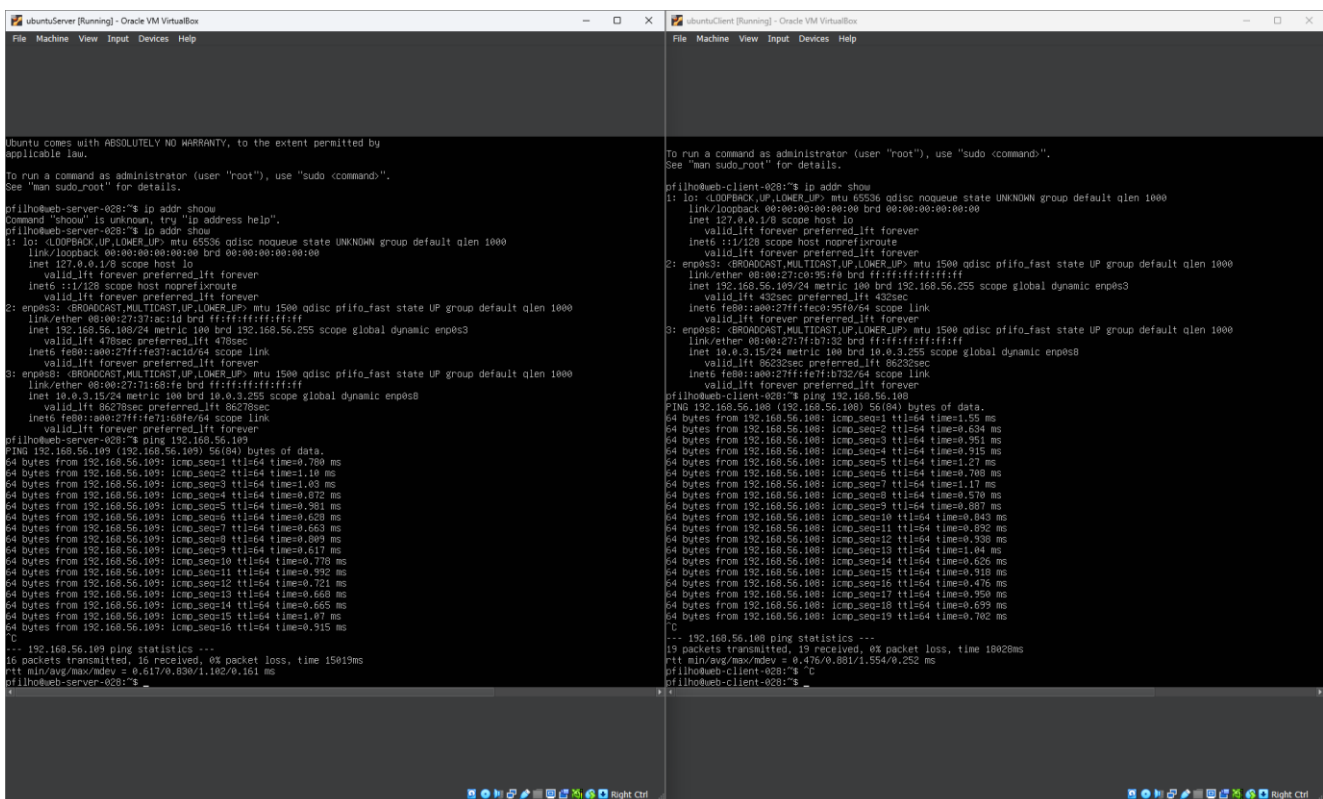
## Part 1A: Virtual Linux Network and Web server

### 1 – Linux updates:

After using the command `sudo apt update` we have all the commands update and also tests our connectivity with the internet.

## 2 – Test Connectivity between both systems:

We used the internal IP addresses of the two virtual machines, ubuntu-server and ubuntu-client, 192.168.56.108 and 192.168.56.109 respectively, to do a series of ping tests to check connectivity between them. Both virtual machines and the host computer can communicate with each other when the Host-Only network setup is used. The two virtual machines (VMs) successfully responded to ping requests, indicating that their internal network connection is intact. Additionally, we confirmed that there was no disruption in communication between the two Linux virtual machines (VMs) and the host computer by using ping. Furthermore, we verified that the virtual machines could access the internet without any problems by pinging popular websites like Google to test external connectivity. All of these tests show that the network configuration is stable and enables dependable internal and external connectivity.



The image shows two terminal windows side-by-side. The left window is titled 'ubuntu-server [Running] - Oracle VM VirtualBox' and the right window is titled 'ubuntu-client [Running] - Oracle VM VirtualBox'. Both windows show the output of the 'ifconfig' command, displaying network interface details for 'lo' and 'eth0'. The left window also shows the output of the 'ping' command, indicating successful connectivity to the right window's IP address (192.168.56.109). The right window shows the output of the 'ping' command, indicating successful connectivity to the left window's IP address (192.168.56.108). The ping results show 16 packets transmitted, 16 received, 0% packet loss, and a time of 1802ms.

Image 3: Machines pinging to each other

## 3 – Testing Access to the DCC Website:

We tested access to the hosted web page labeled "DCC Website Under Construction" after successfully installing Apache on the Ubuntu server. To achieve this, open the Lynx browser on the Ubuntu client virtual machine and go to the Ubuntu server's website. The web page appeared in the Lynx browser when the URL was entered, indicating that the Apache web server was set up correctly and was serving content. This test made sure that clients could successfully access the hosted website and confirmed the web server's operation within the network infrastructure.



Image 4: Webserver working.

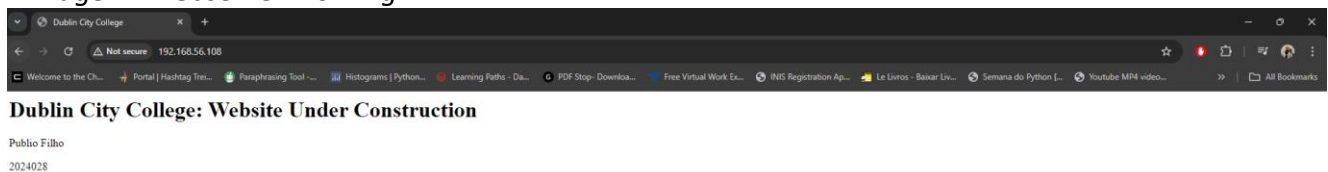


Image 5: Web page edited to show Under construction.

#### 4 – Testing Access to the DCC Website from Host Operating System:

We tested the "DCC Website Under Construction" from the host operating system (Windows or Mac) in order to confirm its accessibility even further. We were able to visit the hosted web page by starting a web browser on the host computer and typing in the URL for the Ubuntu server.

Through this test, it was verified that the Ubuntu server's Apache web server is operational and reachable from other networked devices. The fact that the host environment can access the "DCC Under Construction" page demonstrates how well the web server is configured and supports the reliability of the network infrastructure.

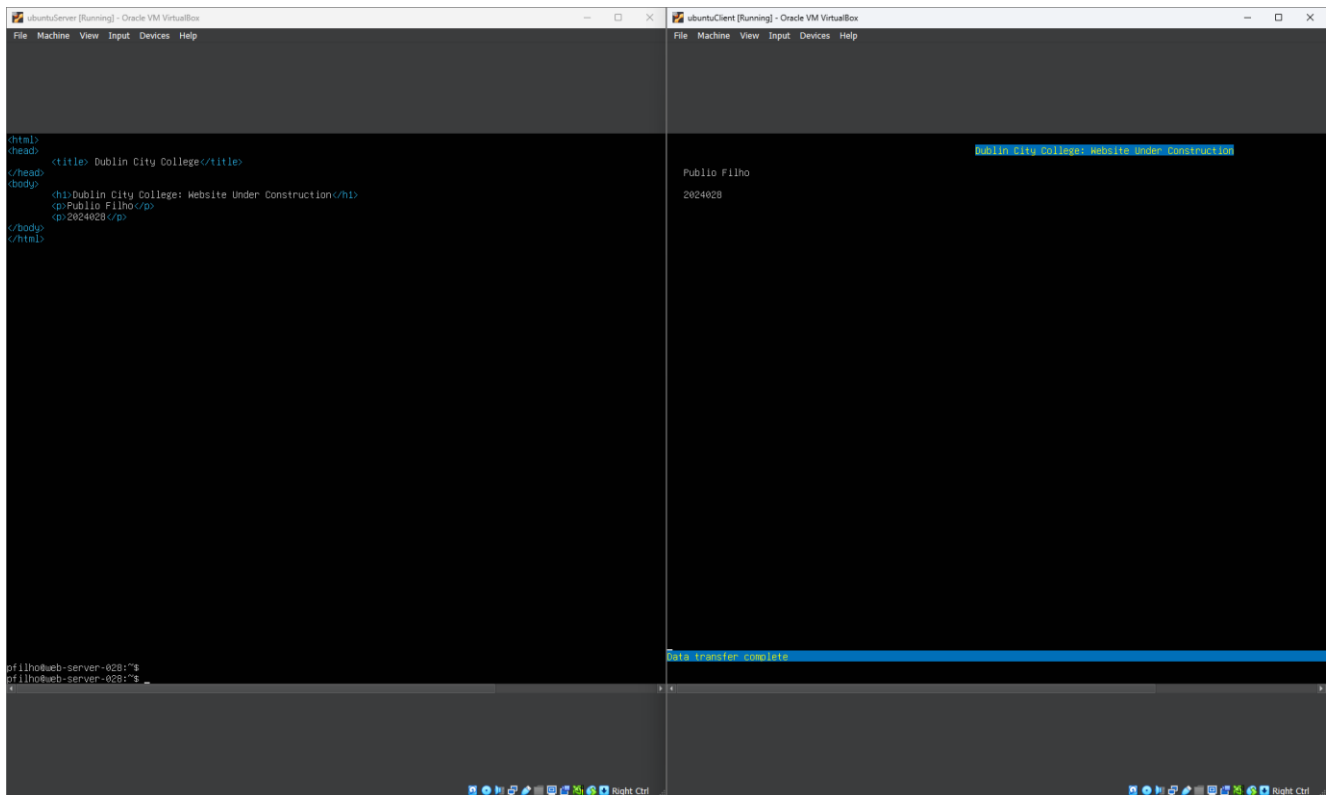


Image 6: Webpage accessed by the unbuntuclient.

## 5 – Monitoring ICMP Traffic Using Wireshark:

We used Wireshark to record network traffic and ping the Ubuntu server IP address from the host PC (Windows or Mac) in order to further examine network connectivity. We saw the ICMP traffic between the host operating system and the Ubuntu web server as the ping function ran. This data showed that the two devices were successfully communicating and responding to each other. As proof of the network communication, we took a screenshot of the Wireshark interface showing the ICMP packets for documentation reasons. This study validates that the host can easily communicate with the virtual machine and shows how successful the network arrangement is.

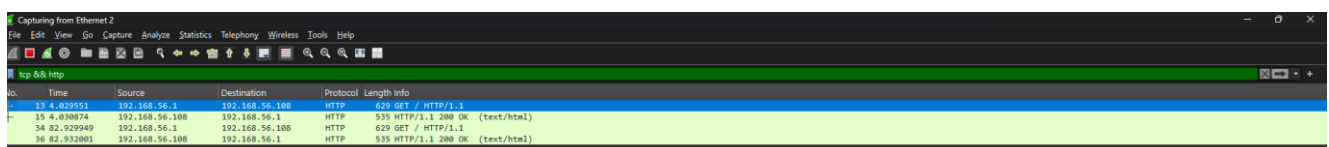


Image 7: Ping in my cmd.

No.	Time	Source	Destination	Protocol	Length	Info
19	24.440844	192.168.56.1	192.168.56.103	TCP	54	60776 → 80 [FIN, ACK] Seq=578 Ack=83 Win=262144 Len=0
20	24.441600	192.168.56.103	192.168.56.1	TCP	60	80 → 60776 [ACK] Seq=483 Ack=577 Win=63744 Len=0
21	35.647250	192.168.56.103	192.168.56.1	TCP	66	[TCP Retransmission] 80 → 60777 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM US=128
22	35.647265	192.168.56.1	192.168.56.103	TCP	66	[TCP Dup ACK 1921] 60777 → 80 [ACK] Seq=1 Ack=1 Win=262056 Len=0 SLE=0 SRE=1
23	43.031250	192.168.56.1	192.168.56.103	TCP	55	[TCP Keep-Alive] 60777 → 80 [ACK] Seq=0 Ack=1 Win=262056 Len=0
24	49.031657	192.168.56.103	192.168.56.1	TCP	66	[TCP Dup ACK 1181] 80 → 60777 [ACK] Seq=1 Ack=1 Win=64256 Len=0 SLE=0 SRE=1
25	55.670145	192.168.56.103	192.168.56.1	TCP	60	80 → 60777 [FIN, ACK] Seq=1 Ack=1 Win=64256 Len=0
26	55.670183	192.168.56.1	192.168.56.103	TCP	54	60777 → 80 [ACK] Seq=1 Ack=2 Win=262056 Len=0
28	82.929005	192.168.56.1	192.168.56.103	TCP	54	60777 → 80 [FIN, ACK] Seq=1 Ack=2 Win=262056 Len=0
30	82.929359	192.168.56.1	192.168.56.103	TCP	66	60796 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 US=256 SACK_PERM
31	82.929673	192.168.56.103	192.168.56.1	TCP	60	80 → 60797 [FIN, ACK] Seq=0 Ack=1 Win=64256 Len=0
32	82.929800	192.168.56.103	192.168.56.1	TCP	66	80 → 60796 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM US=128
33	82.929838	192.168.56.1	192.168.56.103	TCP	54	60796 → 80 [ACK] Seq=1 Ack=1 Win=262056 Len=0
34	82.929940	192.168.56.1	192.168.56.103	HTTP	620	GET / HTTP/1.1
35	82.930670	192.168.56.103	192.168.56.1	TCP	60	80 → 60796 [ACK] Seq=1 Ack=576 Win=63744 Len=0
36	82.932001	192.168.56.103	192.168.56.1	HTTP	535	HTTP/1.1 200 OK (text/html)
37	82.932714	192.168.56.1	192.168.56.103	TCP	66	60797 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 US=256 SACK_PERM
38	82.932990	192.168.56.103	192.168.56.1	TCP	66	80 → 60797 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM US=128
39	82.933020	192.168.56.1	192.168.56.103	TCP	54	60797 → 80 [ACK] Seq=1 Ack=1 Win=2097920 Len=0
40	82.973221	192.168.56.1	192.168.56.103	TCP	54	60796 → 80 [ACK] Seq=576 Ack=482 Win=262144 Len=0
41	87.939067	192.168.56.103	192.168.56.1	TCP	60	80 → 60796 [FIN, ACK] Seq=482 Ack=576 Win=63744 Len=0
42	87.939123	192.168.56.1	192.168.56.103	TCP	54	60796 → 80 [ACK] Seq=576 Ack=483 Win=262144 Len=0
43	114.501746	192.168.56.103	192.168.56.1	TCP	66	[TCP Retransmission] 80 → 60797 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM US=128
44	114.501774	192.168.56.1	192.168.56.103	TCP	66	[TCP Dup ACK 1981] 60797 → 80 [ACK] Seq=1 Ack=1 Win=2097920 Len=0 SLE=0 SRE=1
53	127.934536	192.168.56.1	192.168.56.103	TCP	55	[TCP Keep-Alive] 60797 → 80 [ACK] Seq=0 Ack=1 Win=2097920 Len=0
54	127.935157	192.168.56.103	192.168.56.1	TCP	66	[TCP Dup ACK 1981] 80 → 60797 [ACK] Seq=1 Ack=1 Win=64256 Len=0 SLE=0 SRE=1
55	132.925537	192.168.56.1	192.168.56.103	TCP	55	[TCP Keep-Alive] 60796 → 80 [ACK] Seq=576 Ack=483 Win=262144 Len=0
56	132.940144	192.168.56.103	192.168.56.1	TCP	60	[TCP Keep-Alive ACK] 80 → 60796 [ACK] Seq=483 Ack=576 Win=63744 Len=0
61	134.523849	192.168.56.103	192.168.56.1	TCP	60	80 → 60797 [FIN, ACK] Seq=1 Ack=1 Win=64256 Len=0

Image 8: Ping my ubuntuServer.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.56.1	192.168.56.103	ICMP	74	Echo (ping) request id=0x0001, seq=17/4352, ttl=128 (reply in 2)
2	0.000621	192.168.56.103	192.168.56.1	ICMP	74	Echo (ping) reply id=0x0001, seq=17/4352, ttl=64 (request in 1)
3	1.002572	192.168.56.1	192.168.56.103	ICMP	74	Echo (ping) request id=0x0001, seq=18/4608, ttl=128 (reply in 4)
4	1.002598	192.168.56.103	192.168.56.1	ICMP	74	Echo (ping) reply id=0x0001, seq=18/4608, ttl=64 (request in 3)
5	2.005640	192.168.56.1	192.168.56.103	ICMP	74	Echo (ping) request id=0x0001, seq=19/4864, ttl=128 (reply in 6)
6	2.006301	192.168.56.103	192.168.56.1	ICMP	74	Echo (ping) reply id=0x0001, seq=19/4864, ttl=64 (request in 5)
7	3.008854	192.168.56.1	192.168.56.103	ICMP	74	Echo (ping) request id=0x0001, seq=20/5120, ttl=128 (reply in 8)
8	3.009063	192.168.56.103	192.168.56.1	ICMP	74	Echo (ping) reply id=0x0001, seq=20/5120, ttl=64 (request in 7)
85	292.909191	192.168.56.1	192.168.56.103	TCP	66	54192 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 US=256 SACK_PERM
86	292.909373	192.168.56.1	192.168.56.103	TCP	66	54193 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 US=256 SACK_PERM
87	292.910068	192.168.56.103	192.168.56.1	TCP	60	80 → 54192 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM US=128
88	292.910148	192.168.56.1	192.168.56.103	TCP	54	54192 → 80 [ACK] Seq=1 Ack=1 Win=262056 Len=0
89	292.910271	192.168.56.103	192.168.56.1	TCP	66	80 → 54193 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM US=128
90	292.910308	192.168.56.1	192.168.56.103	TCP	54	54193 → 80 [ACK] Seq=1 Ack=1 Win=262056 Len=0
91	292.910745	192.168.56.1	192.168.56.103	HTTP	631	GET / HTTP/1.1
92	292.911543	192.168.56.103	192.168.56.1	TCP	60	80 → 54192 [ACK] Seq=1 Ack=578 Win=63744 Len=0
93	292.913287	192.168.56.103	192.168.56.1	TCP	1514	80 → 54193 [ACK] Seq=1 Ack=578 Win=3744 Len=1460 [TCP PDUs reassembled in 96]
94	292.913347	192.168.56.103	192.168.56.1	TCP	1514	80 → 54192 [ACK] Seq=1461 Ack=578 Win=3744 Len=1460 [TCP PDUs reassembled in 96]
95	292.913390	192.168.56.1	192.168.56.103	TCP	54	54192 → 80 [ACK] Seq=578 Ack=2921 Win=262056 Len=0
96	292.913454	192.168.56.103	192.168.56.1	HTTP	594	HTTP/1.1 200 OK (text/html)
97	292.954250	192.168.56.1	192.168.56.103	TCP	54	54192 → 80 [ACK] Seq=578 Ack=3461 Win=262144 Len=0
98	297.921109	192.168.56.103	192.168.56.1	TCP	60	80 → 54192 [FIN, ACK] Seq=3461 Ack=578 Win=63744 Len=0
99	297.921180	192.168.56.1	192.168.56.103	TCP	54	54192 → 80 [ACK] Seq=578 Ack=3462 Win=262144 Len=0
100	325.922524	192.168.56.103	192.168.56.1	TCP	66	[TCP Retransmission] 80 → 54192 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM US=128
101	325.922547	192.168.56.1	192.168.56.103	TCP	66	[TCP Dup ACK 9081] 54192 → 80 [ACK] Seq=1 Ack=1 Win=262056 Len=0 SLE=0 SRE=1
104	337.910876	192.168.56.1	192.168.56.103	TCP	55	[TCP Keep-Alive] 54192 → 80 [ACK] Seq=0 Ack=1 Win=262056 Len=0
105	337.911515	192.168.56.103	192.168.56.1	TCP	60	[TCP Dup ACK 1001] 80 → 54192 [ACK] Seq=577 Ack=3462 Win=262144 Len=0
106	342.923020	192.168.56.1	192.168.56.103	TCP	55	[TCP Keep-Alive] 54192 → 80 [ACK] Seq=577 Ack=3462 Win=262144 Len=0
107	342.924232	192.168.56.103	192.168.56.1	TCP	60	[TCP Keep-Alive ACK] 80 → 54192 [ACK] Seq=3462 Ack=578 Win=63744 Len=0

Image 9: Ping my ubuntuClient.

## Part 1B: Hostname Management: Renaming Linux Servers

We were able to successfully rename the Linux servers inside the network infrastructure in order to comply with DCC's requirements. UbuntuServer and UbuntuClient were renamed to web-server-028 and web-client-028, respectively, with "028" standing for my student number's final three digits. In order to guarantee that these modifications are irreversible, the relevant configuration files were updated, enabling the servers to keep their new hostnames even after rebooting. We took screenshots of the modified hostnames on both Ubuntu servers to record this procedure and ensure that the renaming was carried out properly. This step is vital for maintaining organized network management and guaranteeing clarity in server identification.

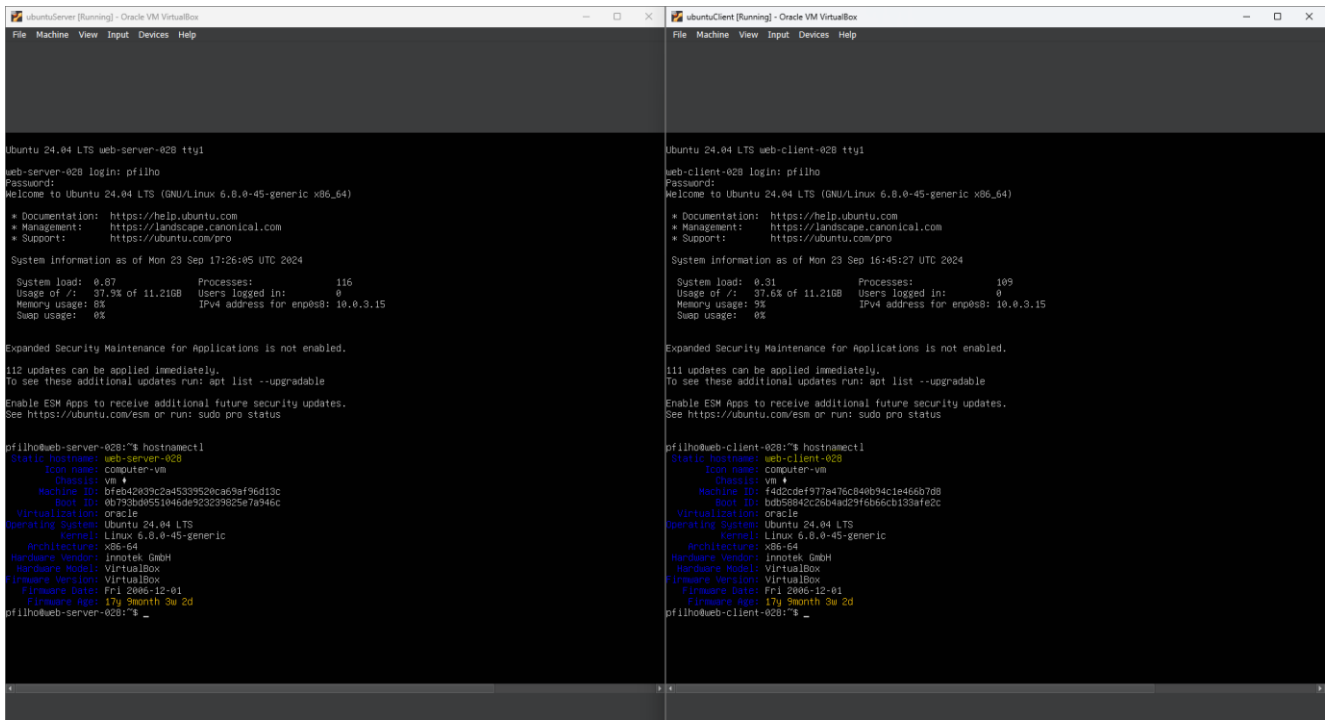


Image 10: Hostnames changed permanently.

## Part 1C: IP Address Management: Configuring Permanent IP Addresses

We used the nano text editor to alter the Netplan configuration file in order to create a reliable network setup for the Linux servers. We gave the Ubuntu server a permanent IP address of 192.168.56.100/24. We implemented the new settings to make sure they take effect after making the required changes to the configuration files to make this change permanent. Next, we used the ping command to check connectivity between the two Linux virtual computers. The fact that both VMs could successfully communicate and that the IP address setting was accurate was validated by the successful ping responses. Ensuring that the servers are reachable within the network and preserving dependable network interactions depend on this phase.



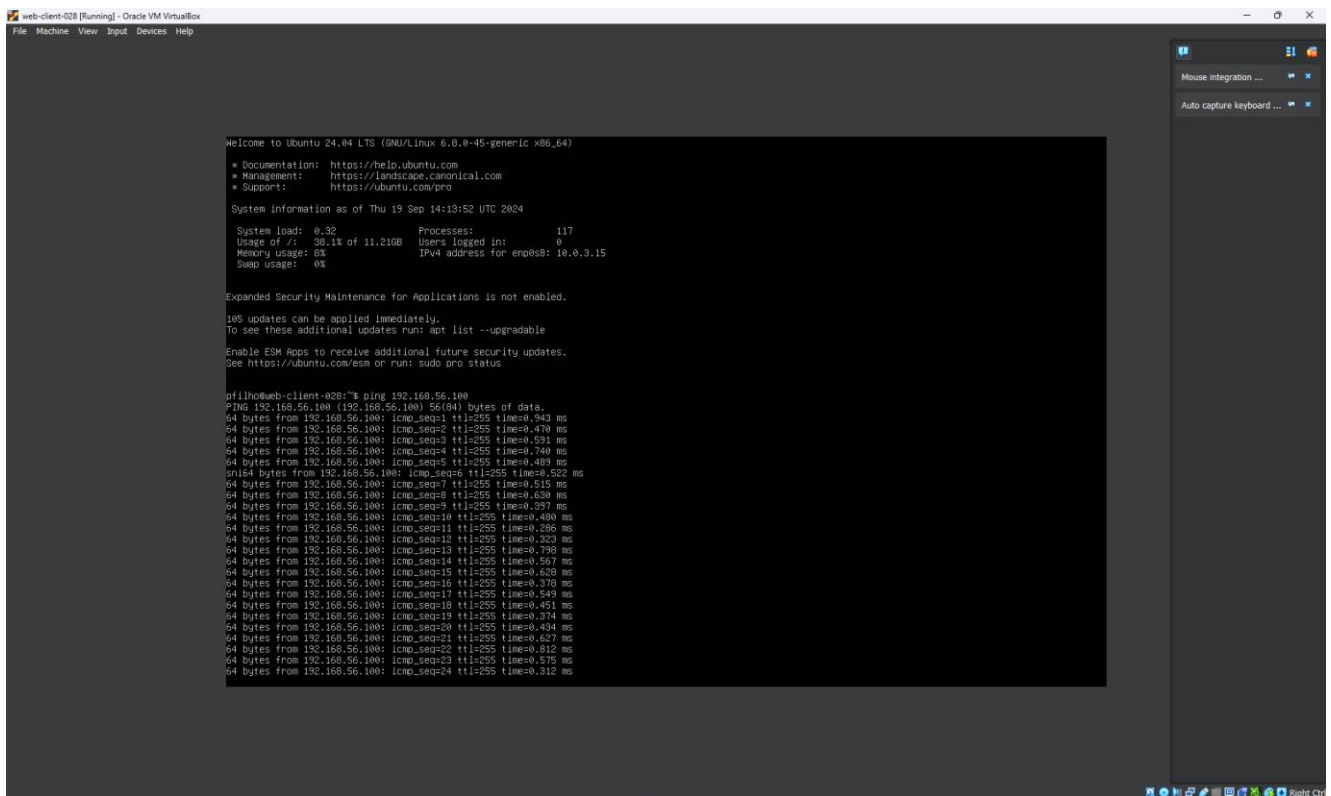


Image 11: Ip address changed and pinging with the UbuntuClient.

## Part 1D: IP Address Management with DHCP Configuration

We installed and set up a DHCP server on web-server-028 to enable dynamic IP address allocation for the web-client-123 virtual machine. With a DHCP range of 192.168.56.150 to 192.168.56.200, the DHCP server was configured to administer the subnet 192.168.56.0/24. By doing this, the web-client-028 computer is guaranteed to get 192.168.56.150, which is the first IP address in the pool that becomes accessible. Furthermore, we set up 192.168.56.1 as the default gateway and DNS address to facilitate easy network routing and name resolution. Following configuration completion, the client was able to dynamically obtain the IP address from the DHCP server, validating that the network and DHCP functions were configured correctly.

## Part 2: Using SSH for Secure Data Transfer Between Servers

We utilized SSH to create a remote connection between the two virtual machines in order to facilitate safe data transfer. We set up PuTTY (for Windows) on the host machine in order to establish a remote SSH connection to web-server-028. After entering the IP address of the server into PuTTY, we were able to log into web-server-028 remotely from the host computer. This proved that SSH was set up correctly and was operating as planned, enabling safe connection between the host and Ubuntu server. The successful remote connection is verified by a screenshot taken from PuTTY of the SSH session.

```
pfilho@web-server-028: ~  
  
* Documentation:  https://help.ubuntu.com  
* Management:    https://landscape.canonical.com  
* Support:        https://ubuntu.com/pro  
  
System information as of Tue 24 Sep 09:58:39 UTC 2024  
  
System load:  0.24          Processes:           117  
Usage of /:   37.8% of 11.21GB Users logged in:       1  
Memory usage: 10%          IPv4 address for enp0s8: 10.0.3.15  
Swap usage:   0%  
  
Expanded Security Maintenance for Applications is not enabled.  
  
199 updates can be applied immediately.  
83 of these updates are standard security updates.  
To see these additional updates run: apt list --upgradable  
  
Enable ESM Apps to receive additional future security updates.  
See https://ubuntu.com/esm or run: sudo pro status  
  
pfilho@web-server-028:~$
```

Image 12: UbuntuServer connected to Putty.

We utilized Wireshark on the host operating system to confirm the encryption of communication between the web-client-028 virtual machine and web-server-028. Wireshark was configured to record network activity when the client was connected to web-server-028 via SSH. We were able to ensure that all data exchanged over the connection was securely encrypted during the session by observing encrypted SSH traffic between the client and the server. To identify the precise packets displaying the SSH encrypted traffic, a screenshot was obtained. The correct security of data exchanges between the client and server is confirmed by this examination using SSH.

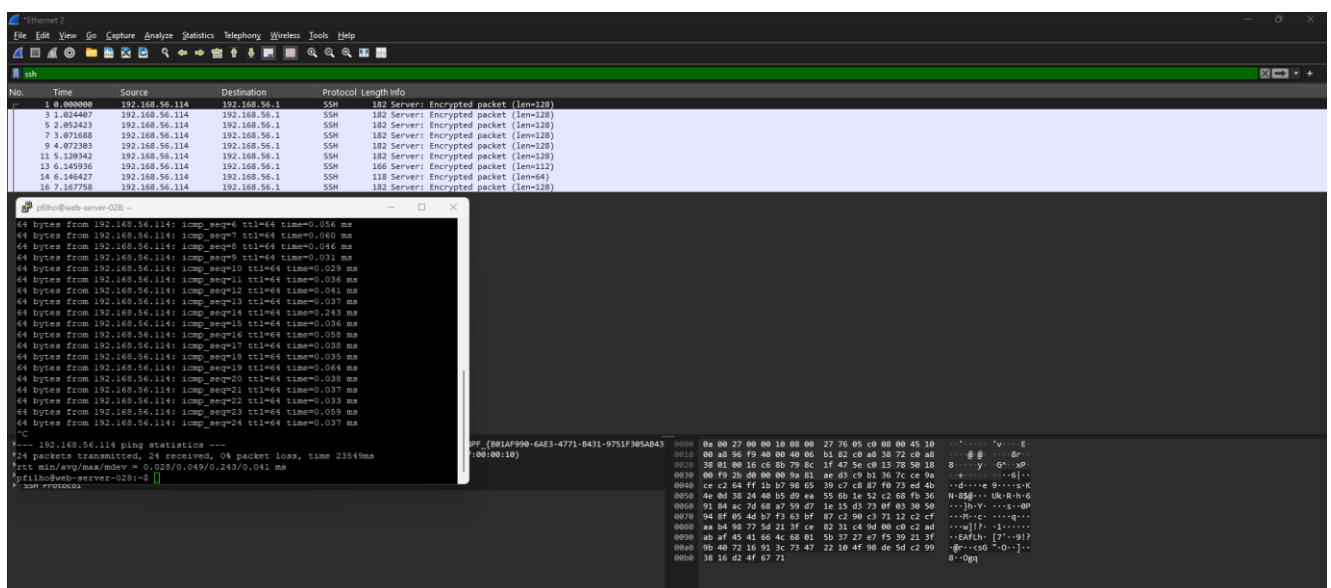
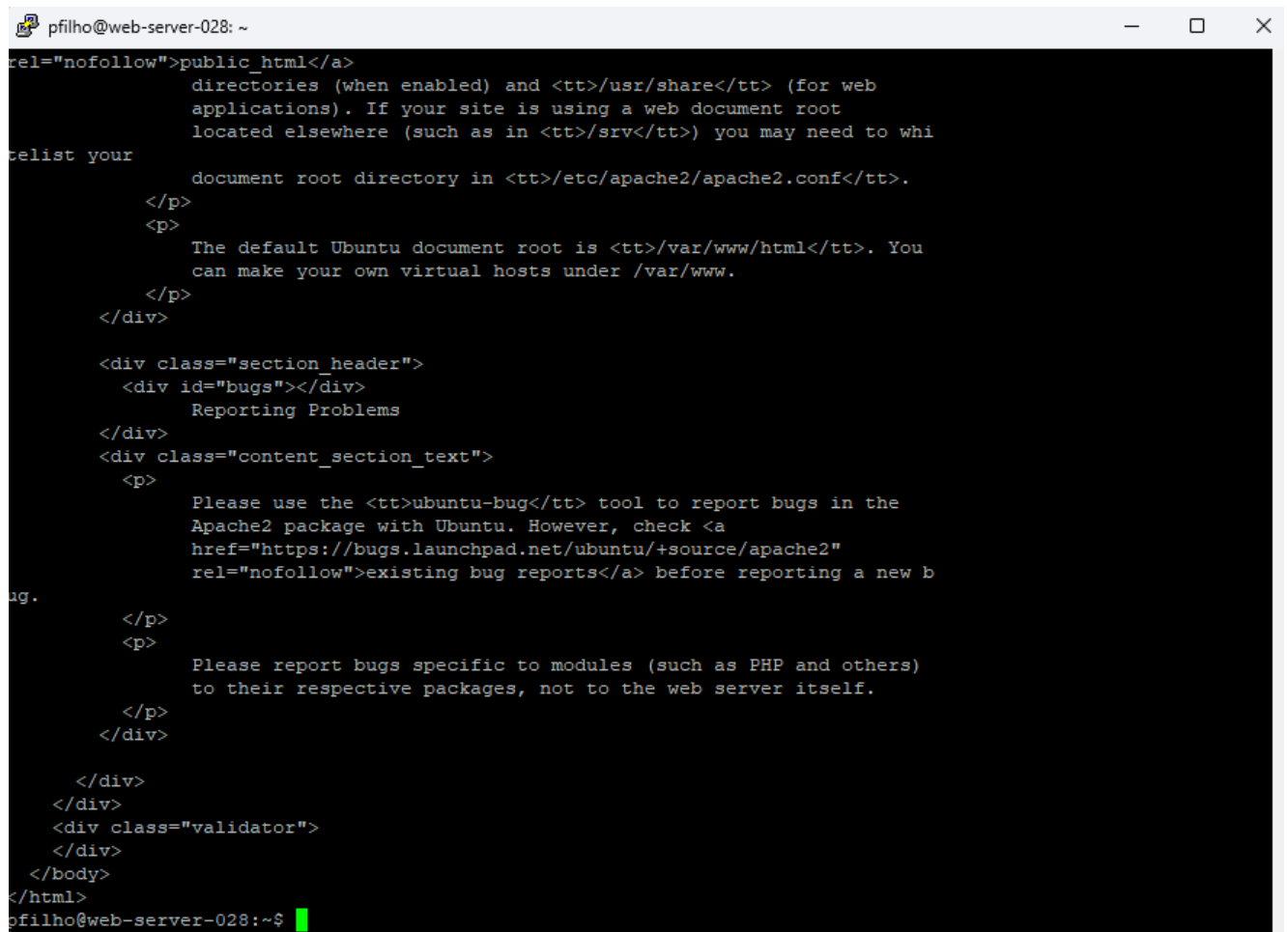


Image 13: UbuntuServer pinging in Putty confirmed in Whiteshark.

### Part 3: Security Configuration

In this task, I configured the Uncomplicated Firewall (UFW) on Ubuntu to control both HTTP and SSH traffic. UFW is a user-friendly firewall utility that allows the creation of host-based firewall rules. By default, UFW is disabled, and it requires manual configuration to control network traffic. The following steps outline how I blocked and allowed both HTTP and SSH traffic, along with evidence in the form of screenshots to demonstrate the firewall's functionality.



```
pfilho@web-server-028: ~  
rel="nofollow">public_html</a>  
    directories (when enabled) and <tt>/usr/share</tt> (for web  
    applications). If your site is using a web document root  
    located elsewhere (such as in <tt>/srv</tt>) you may need to whi  
celist your  
    document root directory in <tt>/etc/apache2/apache2.conf</tt>.  
</p>  
<p>  
    The default Ubuntu document root is <tt>/var/www/html</tt>. You  
    can make your own virtual hosts under /var/www.  
</p>  
</div>  
  
<div class="section_header">  
    <div id="bugs"></div>  
        Reporting Problems  
</div>  
<div class="content_section_text">  
    <p>  
        Please use the <tt>ubuntu-bug</tt> tool to report bugs in the  
        Apache2 package with Ubuntu. However, check <a  
        href="https://bugs.launchpad.net/ubuntu/+source/apache2"  
        rel="nofollow">existing bug reports</a> before reporting a new b  
ug.  
</p>  
<p>  
        Please report bugs specific to modules (such as PHP and others)  
        to their respective packages, not to the web server itself.  
</p>  
</div>  
  
</div>  
</div>  
<div class="validator">  
</div>  
</body>  
</html>  
pfilho@web-server-028:~$
```

Image 14: Connection HTTP working

```
pfilho@web-server-028: ~  
<div class="content_section_text">  
  <p>  
    Please use the <tt>ubuntu-bug</tt> tool to report bugs in the  
    Apache2 package with Ubuntu. However, check <a  
      href="https://bugs.launchpad.net/ubuntu/+source/apache2"  
      rel="nofollow">existing bug reports</a> before reporting a new b  
  </p>  
  <p>  
    Please report bugs specific to modules (such as PHP and others)  
    to their respective packages, not to the web server itself.  
  </p>  
</div>  
  
</div>  
</div>  
<div class="validator">  
</div>  
</body>  
</html>  
pfilho@web-server-028:~$ curl http://web-server-028  
curl: (7) Failed to connect to web-server-028 port 80 after 1 ms: Couldn't connect to server  
pfilho@web-server-028:~$
```

Image 15: Connection HTTP denied

```
pfilho@web-server-028: ~  
* Documentation:  https://help.ubuntu.com  
* Management:    https://landscape.canonical.com  
* Support:        https://ubuntu.com/pro  
  
System information as of Thu 26 Sep 13:32:46 UTC 2024  
  
System load:  0.01          Processes:            136  
Usage of /:   38.0% of 11.21GB Users logged in:           1  
Memory usage: 17%          IPv4 address for enp0s8: 10.0.3.15  
Swap usage:   0%  
  
Expanded Security Maintenance for Applications is not enabled.  
  
111 updates can be applied immediately.  
To see these additional updates run: apt list --upgradable  
  
Enable ESM Apps to receive additional future security updates.  
See https://ubuntu.com/esm or run: sudo pro status  
  
Last login: Thu Sep 26 13:28:04 2024 from 192.168.56.1  
pfilho@web-server-028:~$
```

Image 16: Connection SSH working

```
File "/usr/lib/python3.12/socketserver.py", line 457, in init
pfilho@web-server-028: ~
* Documentation: https://help.ubuntu.com
* Management: https://landscape.canonical.com
* Support: https://ubuntu.com/pro
System information as of Thu 26 Sep 13:32:46 UTC 2024
System load: 0.01 Processes: 136
Usage of /: 38.0% of 11.21GB Users logged in: 1
Memory usage: 17% IPv4 address for enp0s8: 10.0.3.15
Swap usage: 0%
Expanded Security Maintenance for Applications is not enabled.
All updates can be applied immediately.
To see these additional updates run: apt list --upgradable
Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status
Last login: Thu Sep 26 13:28:04 2024 from 192.168.56.1
pfilho@web-server-028:~$
pfilho@web-server-028:~$ sudo iptables -A INPUT -p tcp --dport 22 -j REJECT
```

Image 17: Connection SSH denied

#### Part 4: The SAMBA file server

On web-server-123 (Ubuntu), we set up and installed Samba in order to facilitate file sharing via the SMB protocol between Ubuntu and Windows PCs. Nano was used to add a test file, my-samba-file.txt, to the shared folder samba-folder. To ensure secure access, permissions were set to need a password from users in order to access the folder. The host computer was able to access the Samba share with success, and we verified that we could add new files and modify existing ones, such my-samba-file.txt. Samba is an essential tool for mixed OS situations since it allows cross-platform file and printer sharing.



Image 18: Giving permission.

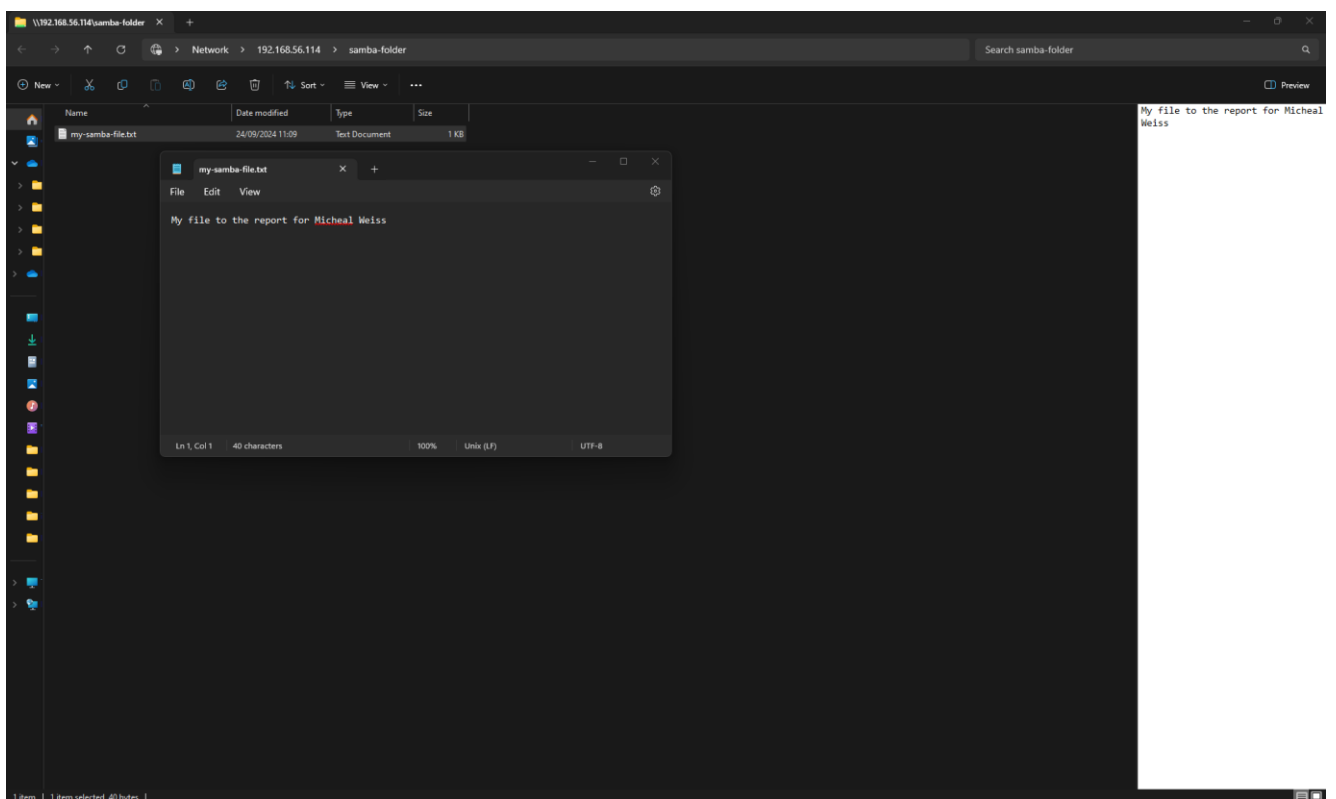


Image 19: Samba file opened in my computer.

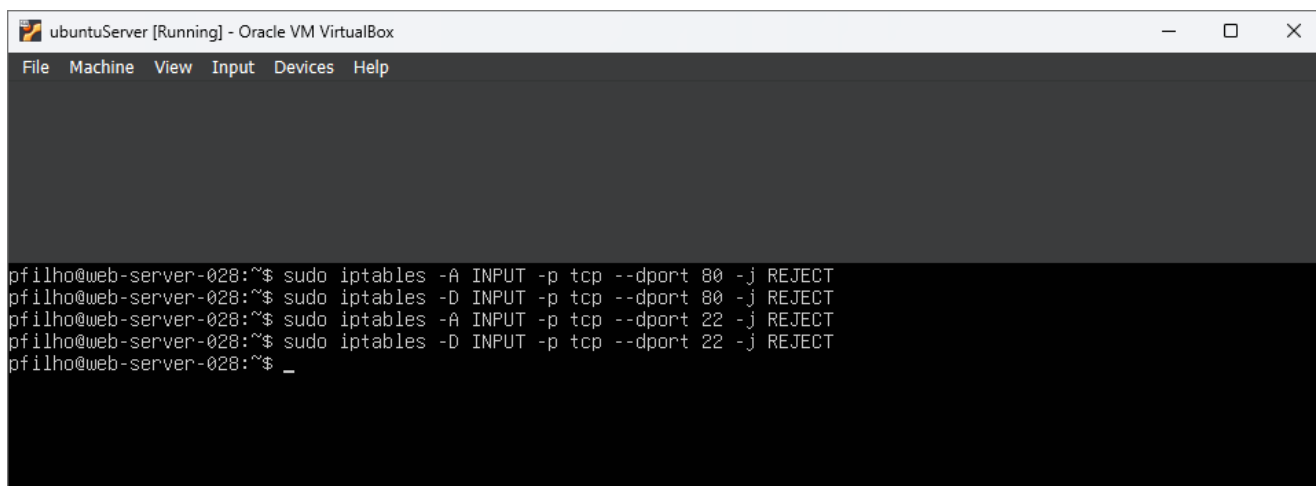


*Image 20: file edited in my computer showed in my ubuntuserver.*

## **Part 5: Research and challenge activities**

### **Part A: Security configuration #2**

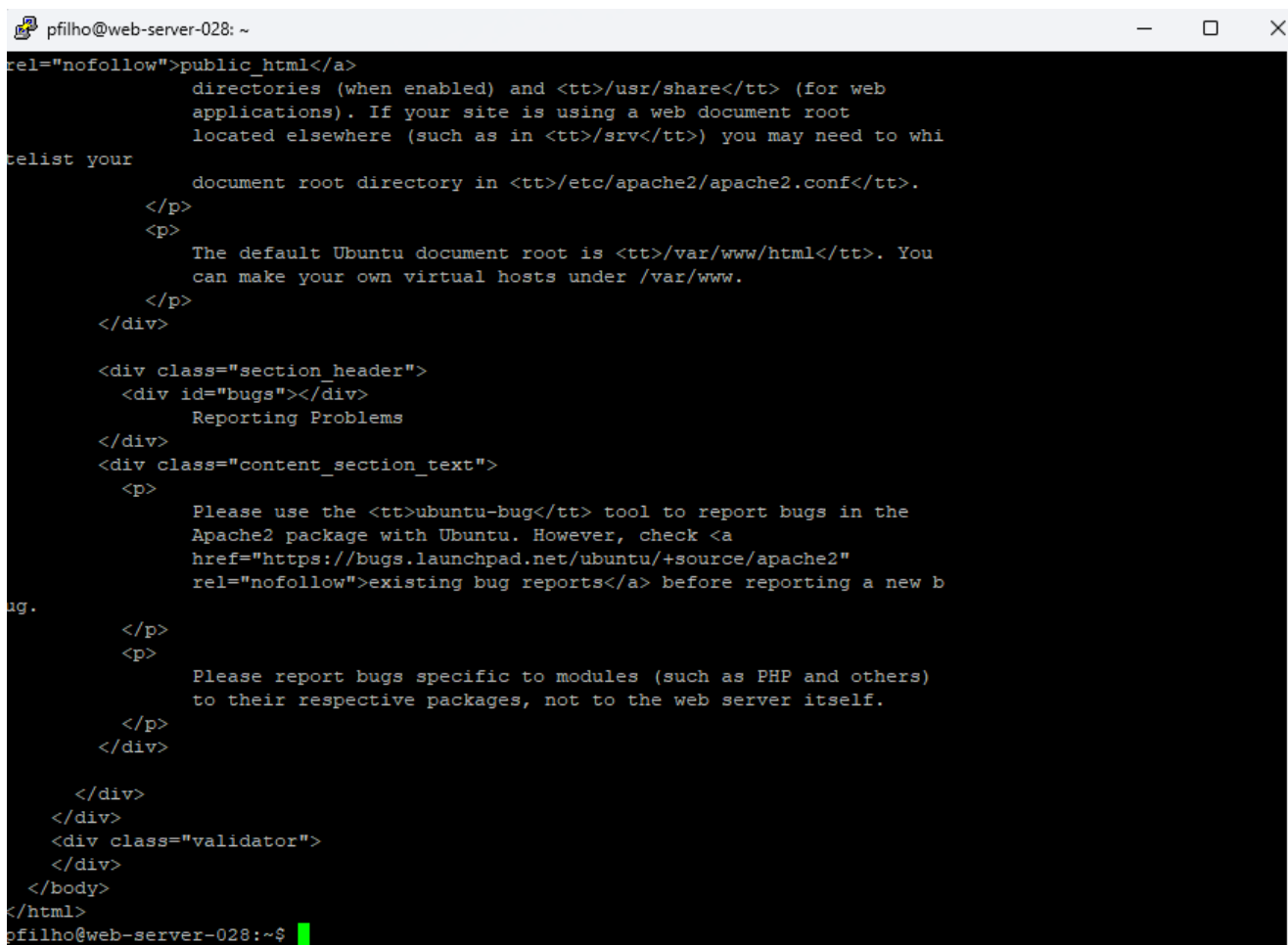
In this task, I configured iptables, a traditional firewall utility, to control the HTTP and SSH traffic entering and exiting the server. Before the introduction of Uncomplicated Firewall (UFW), Linux administrators commonly used iptables to manage firewall rules. At the request of my supervisor at 'Consult & Connect Ltd', I demonstrated the use of iptables to block and allow both HTTP and SSH traffic. Below are the steps I followed to configure iptables, along with evidence of their effectiveness.



```
ubuntuServer [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help

pfilho@web-server-028:~$ sudo iptables -A INPUT -p tcp --dport 80 -j REJECT
pfilho@web-server-028:~$ sudo iptables -D INPUT -p tcp --dport 80 -j REJECT
pfilho@web-server-028:~$ sudo iptables -A INPUT -p tcp --dport 22 -j REJECT
pfilho@web-server-028:~$ sudo iptables -D INPUT -p tcp --dport 22 -j REJECT
pfilho@web-server-028:~$ _
```

Image 21: codes used to block and allow SSH and HTTP.



```
pfilho@web-server-028: ~
rel="nofollow">public_html</a>
    directories (when enabled) and <tt>/usr/share</tt> (for web
    applications). If your site is using a web document root
    located elsewhere (such as in <tt>/srv</tt>) you may need to whi
telist your
    document root directory in <tt>/etc/apache2/apache2.conf</tt>.
</p>
<p>
    The default Ubuntu document root is <tt>/var/www/html</tt>. You
    can make your own virtual hosts under /var/www.
</p>
</div>

<div class="section_header">
  <div id="bugs"></div>
    Reporting Problems
  </div>
  <div class="content_section_text">
    <p>
      Please use the <tt>ubuntu-bug</tt> tool to report bugs in the
      Apache2 package with Ubuntu. However, check <a
      href="https://bugs.launchpad.net/ubuntu/+source/apache2"
      rel="nofollow">existing bug reports</a> before reporting a new b
ug.
    </p>
    <p>
      Please report bugs specific to modules (such as PHP and others)
      to their respective packages, not to the web server itself.
    </p>
  </div>
</div>
<div class="validator">
</div>
</body>
</html>
pfilho@web-server-028:~$
```

Image 22: Connection HTTP working.



```
pfilho@web-server-028: ~  
<div class="content_section_text">  
  <p>  
    Please use the <tt>ubuntu-bug</tt> tool to report bugs in the  
    Apache2 package with Ubuntu. However, check <a  
    href="https://bugs.launchpad.net/ubuntu/+source/apache2"  
    rel="nofollow">existing bug reports</a> before reporting a new b  
  </p>  
  <p>  
    Please report bugs specific to modules (such as PHP and others)  
    to their respective packages, not to the web server itself.  
  </p>  
</div>  
  
</div>  
<div class="validator">  
</div>  
</body>  
</html>  
pfilho@web-server-028:~$ curl http://web-server-028  
curl: (7) Failed to connect to web-server-028 port 80 after 1 ms: Couldn't conne  
ct to server  
pfilho@web-server-028:~$
```

Image 23: Connection HTTP denied.

```
pfilho@web-server-028: ~  
* Documentation:  https://help.ubuntu.com  
* Management:    https://landscape.canonical.com  
* Support:        https://ubuntu.com/pro  
  
System information as of Thu 26 Sep 13:32:46 UTC 2024  
  
System load:  0.01          Processes:            136  
Usage of /:   38.0% of 11.21GB Users logged in:          1  
Memory usage: 17%          IPv4 address for enp0s8: 10.0.3.15  
Swap usage:   0%  
  
Expanded Security Maintenance for Applications is not enabled.  
  
0 111 updates can be applied immediately.  
To see these additional updates run: apt list --upgradable  
  
ze Enable ESM Apps to receive additional future security updates.  
ze See https://ubuntu.com/esm or run: sudo pro status  
  
v1  
  
v1 Last login: Thu Sep 26 13:28:04 2024 from 192.168.56.1  
pfilho@web-server-028:~$
```

Image 24: Connection SSH working.

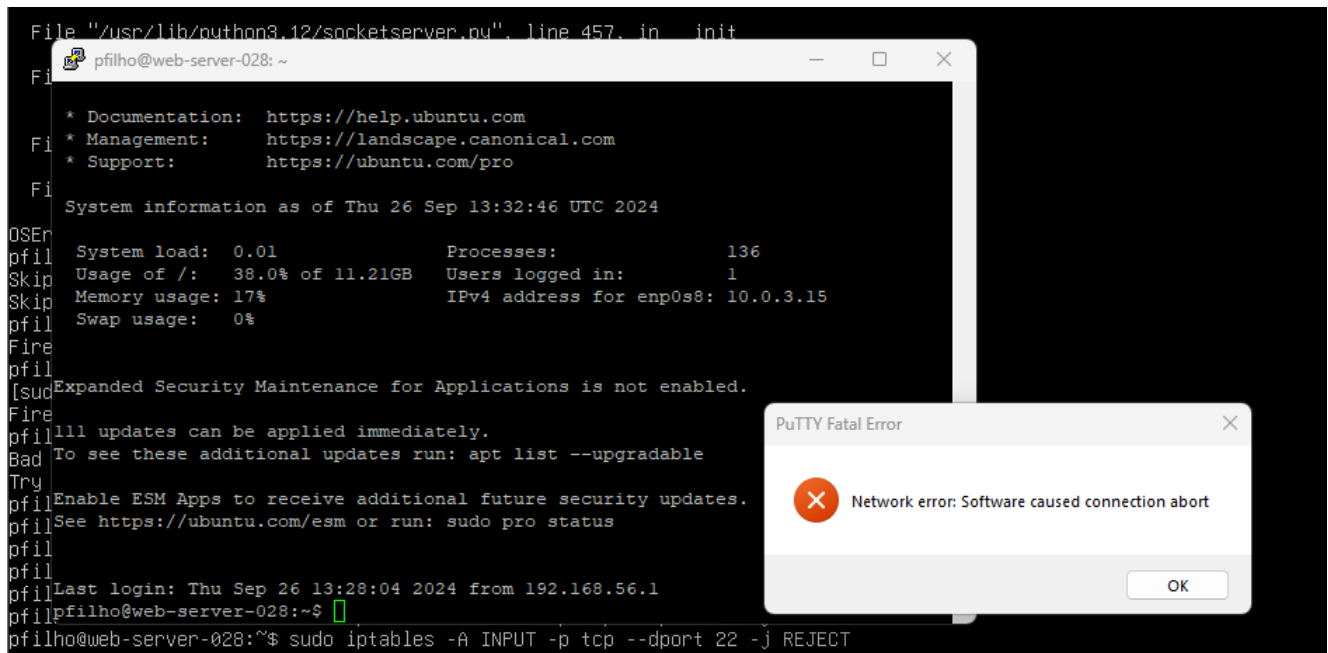


Image 25: Connection SSH denied.

#### Part D: Forgotten password

In this task, I assisted an associate who had forgotten the password to virtual machines running both Linux and Windows operating systems in Oracle VirtualBox. Rather than creating new virtual machines, I demonstrated how to regain access by resetting the passwords on both systems. For Linux, I used the GRUB boot loader to access recovery mode and reset the password through the root shell. For Windows, I used the installation disk to access the command prompt and reset the password via the Ease of Access tool. Below are the steps I followed for each system, along with evidence of their effectiveness.

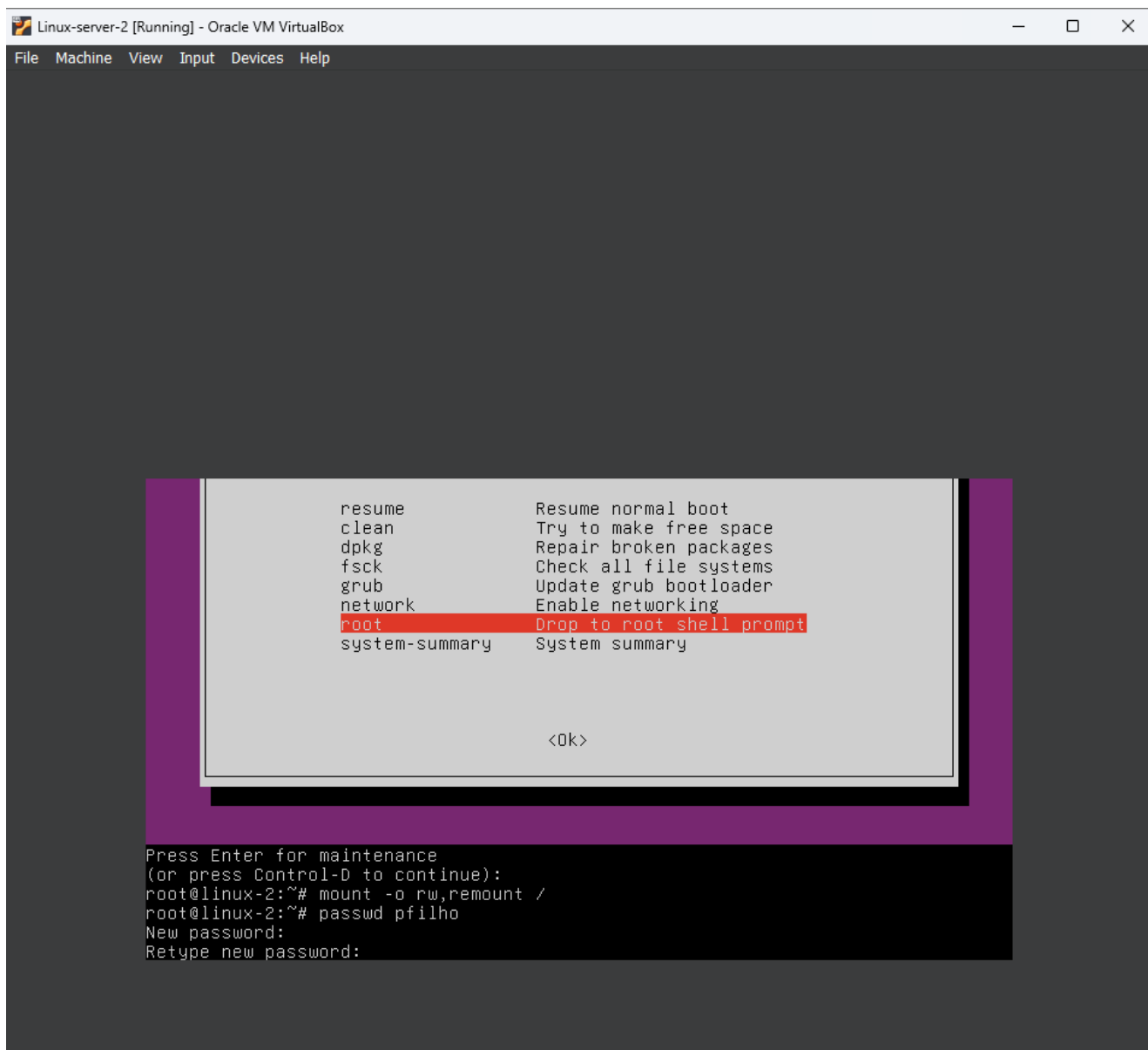


Image 26: Password reset.