- **Project Title:**
  Project 2: Configuring and Securing a Linux Server
- **Student Name:**
  Pelumi Johnson
- **Submission Date:**
  11/24/2025

---

1. Part 1: Install and Configure a Web Server

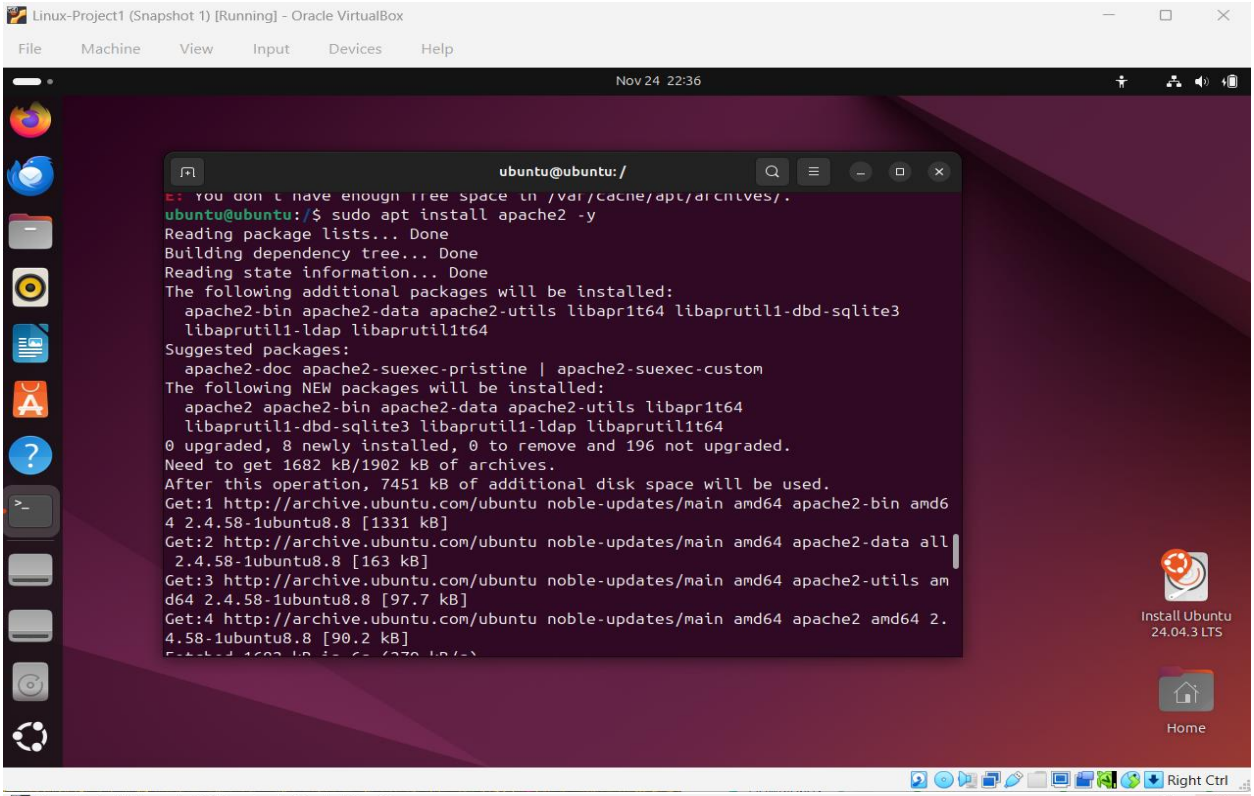
- **Documentation:**
  Describe your configuration of the web server (Apache or Nginx) and how you hosted a simple web page.
  ANSWER:

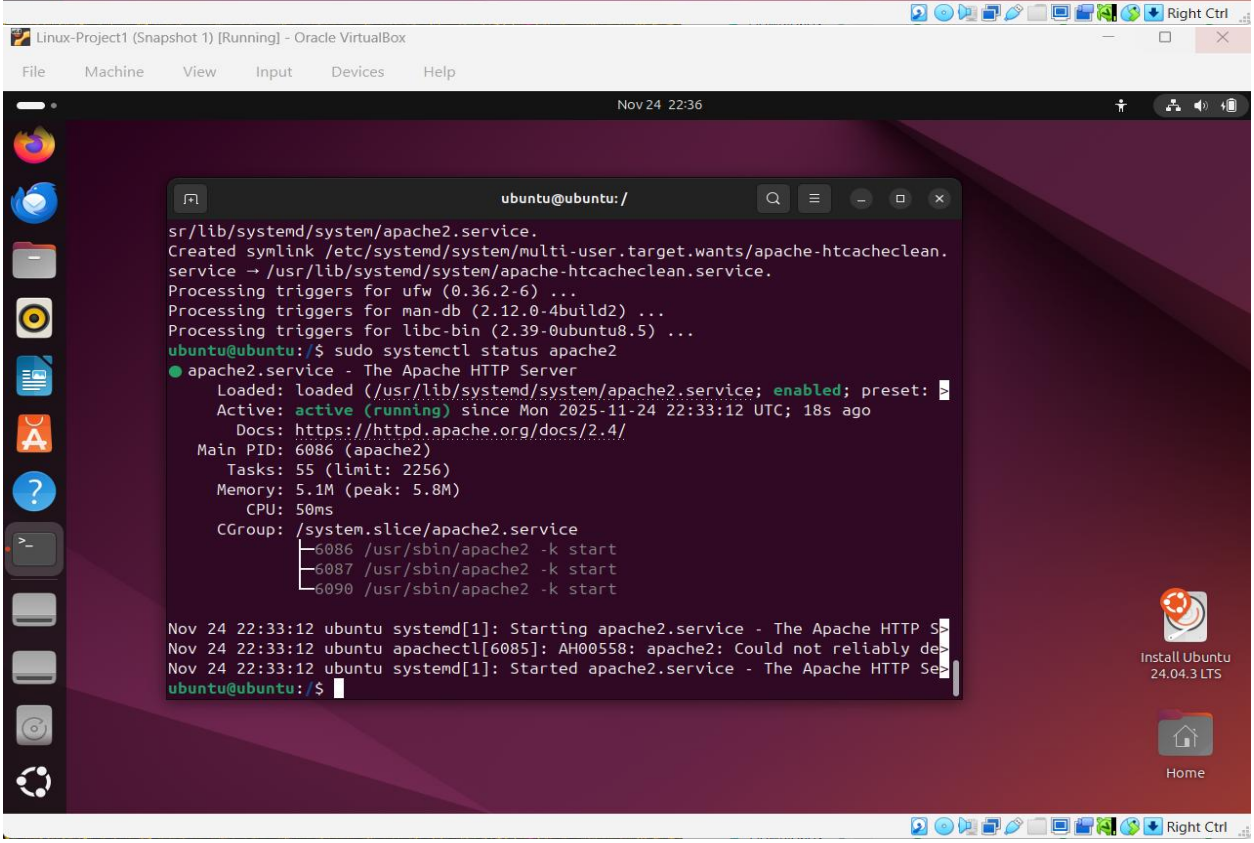
Apache is a web server that allows a Linux system to host websites and deliver web pages to anyone who visits the server's IP address. To begin this step, I started by updating my Ubuntu virtual machine so that all packages were current. I opened the terminal and ran `sudo apt update` followed by `sudo apt upgrade -y`. After updating, I installed Apache using `sudo apt install apache2 -y`. Once the installation finished, I checked the service with `sudo systemctl status apache2` to make sure it was active and running. Seeing "active (running)" confirmed that the web server was functioning correctly.

Next, I found the IP address of my VM using ip a, which I needed to test the server. When I opened a browser and entered the IP address (10.0.2.15), the default Apache page appeared, showing that the server was hosting content successfully. To complete the task, I went into /var/www/html, renamed the default index page, and created my own index.html with a simple message and date/time. After saving the file, I restarted Apache and refreshed the browser. My custom webpage loaded successfully, confirming that Apache was installed, configured, and serving my page.

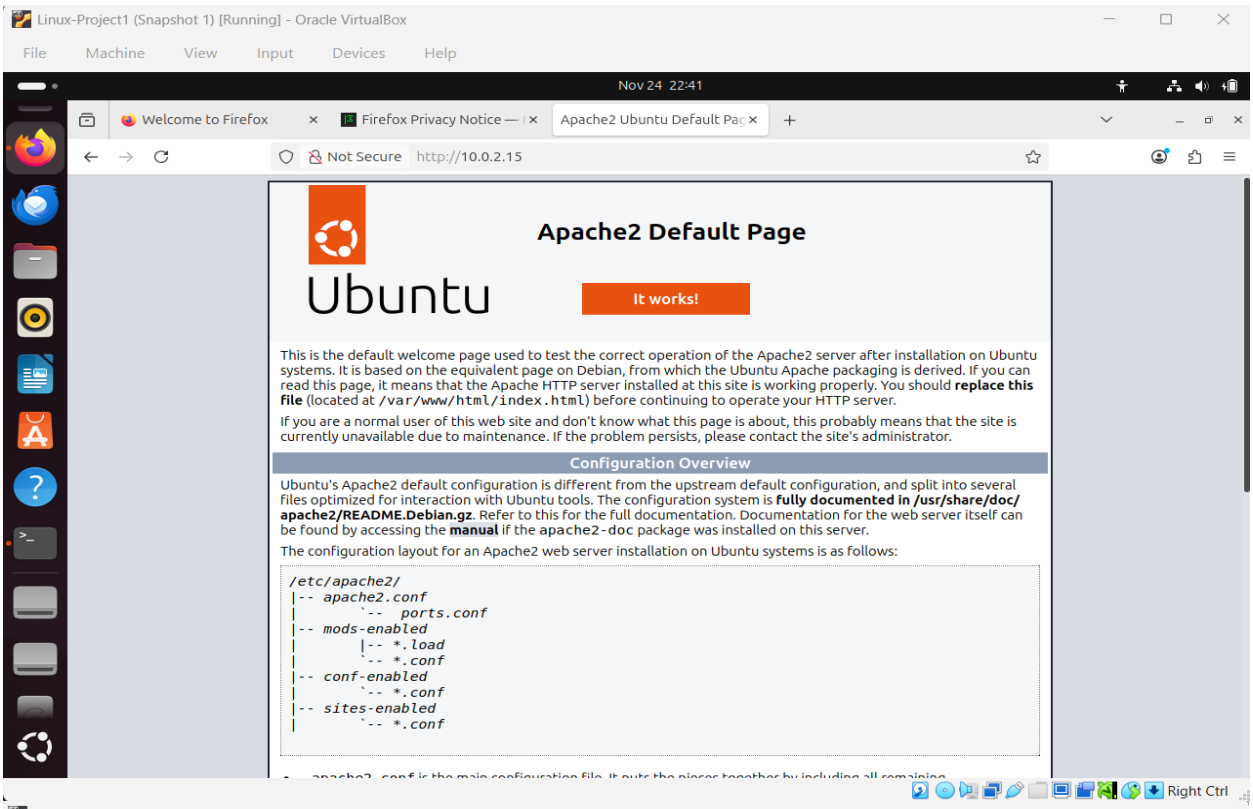
- Lo {means computer talking to itself} is just the loopback (127.0.0.1)
- enp0s3 is your actual network interface
- The IPv4 address attached to it (10.0.2.15) is your VM's address inside its virtual network

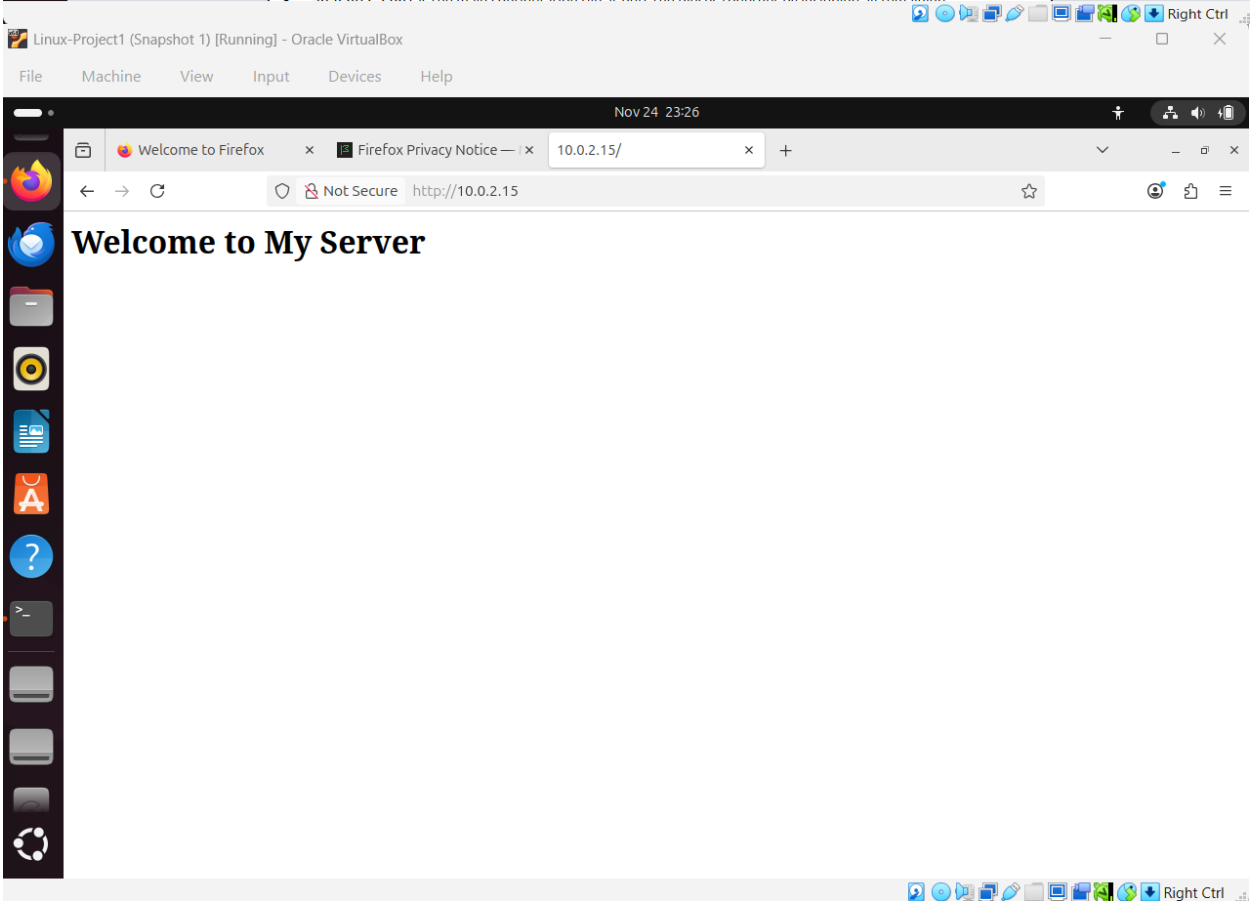
- **Screenshots:**
  - Web server setup confirmation





- Hosted webpage display (with URL and visible Date/Time)


Linux-Project1 (Snapshot 1) [Running] - Oracle VirtualBox
File Machine View Input Devices Help
Nov 24 22:41
Welcome to Firefox
Firefox Privacy Notice —
Apache2 Ubuntu Default Pag
Not Secure http://10.0.2.15
Apache2 Default Page
Ubuntu
It works!
This is the default welcome page used to test the correct operation of the Apache2 server after installation on Ubuntu systems. It is based on the equivalent page on Debian, from which the Ubuntu Apache packaging is derived. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should replace this file (located at /var/www/html/index.html) before continuing to operate your HTTP server.
If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.
Configuration Overview
Ubuntu's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Ubuntu tools. The configuration system is fully documented in /usr/share/doc/apache2/README.Debian.gz. Refer to this for the full documentation. Documentation for the web server itself can be found by accessing the manual if the apache2-doc package was installed on this server.
The configuration layout for an Apache2 web server installation on Ubuntu systems is as follows:
/etc/apache2/
|-- apache2.conf
| `-- ports.conf
|-- mods-enabled
| |-- *.load
| `-- *.conf
|-- conf-enabled
| `-- *.conf
|-- sites-enabled
| `-- *.conf
Right Ctrl


Linux-Project1 (Snapshot 1) [Running] - Oracle VirtualBox
File Machine View Input Devices Help
Nov 24 23:26
Welcome to Firefox
Firefox Privacy Notice —
10.0.2.15/
Not Secure http://10.0.2.15
Welcome to My Server
Right Ctrl

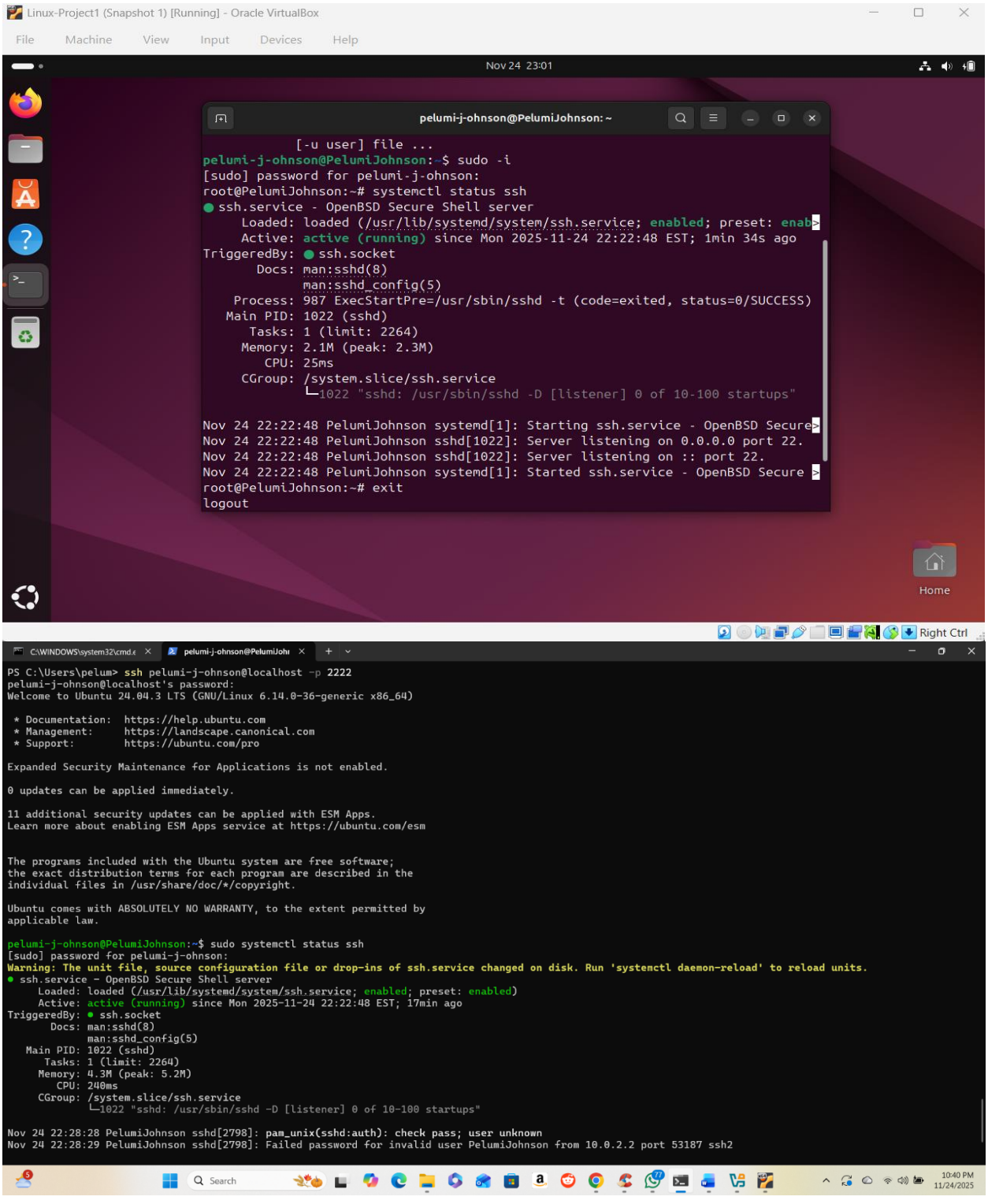
2. Part 2: Set Up SSH

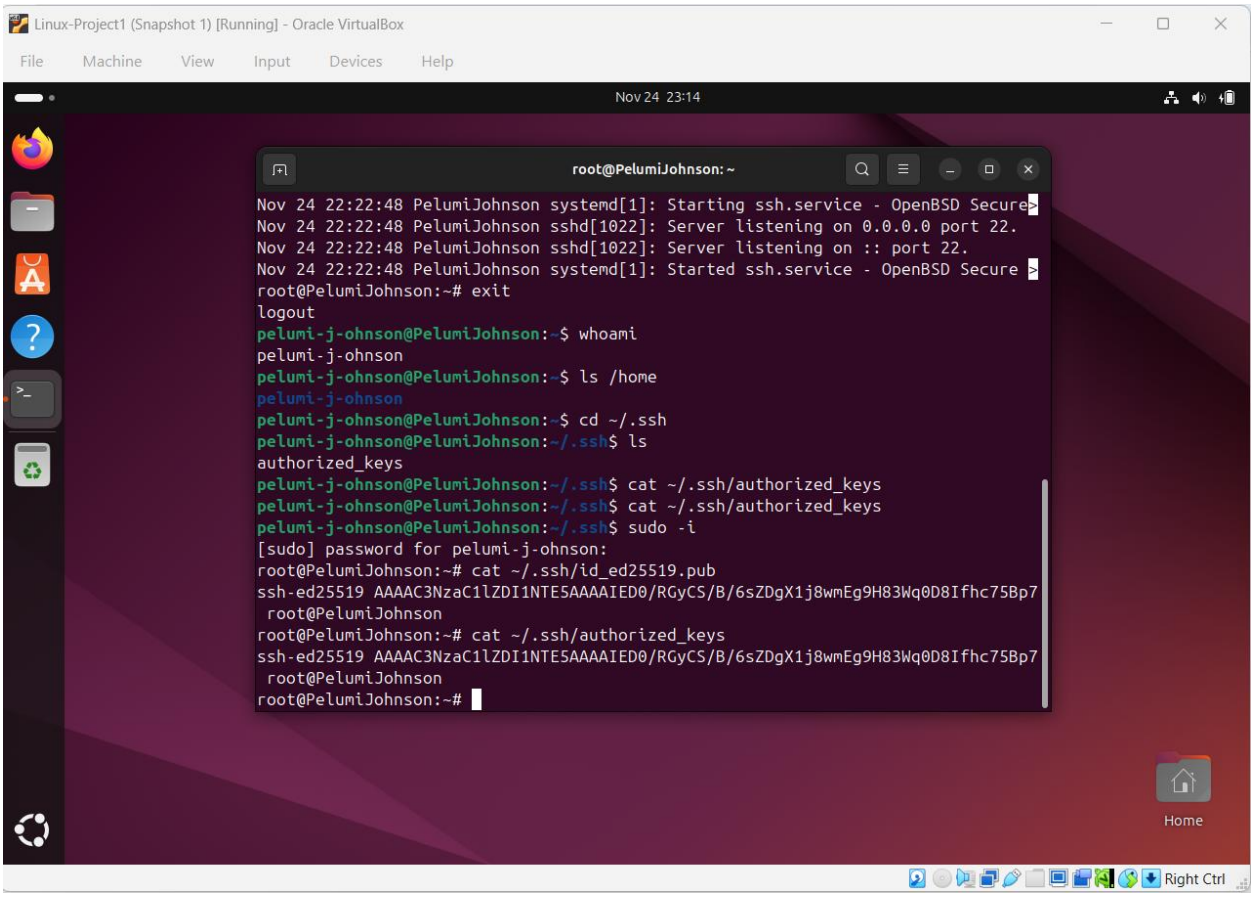
- **Documentation:**
  Explain how you installed and configured SSH, enabled key-based authentication, and disabled password-based authentication.
  ANSWER:

  Here's a smooth, clean, single-paragraph version you can paste directly into your assignment: SSH (Secure Shell) is a protocol that allows secure, encrypted remote access to a machine, and setting it up on my Ubuntu VM made it possible to manage the system from my Windows host. I began by installing and enabling the OpenSSH server using "sudo apt update" and "sudo apt install openssh-server," then verified that the service was active with "sudo systemctl status ssh." Because VirtualBox uses NAT networking, I configured a port-forwarding rule so my host could reach the VM by mapping host port 2222 to the VM's SSH port 22. This let me connect from PowerShell using "ssh pelumi-j-ohnson@localhost -p 2222."

  After confirming SSH access, I strengthened security by setting up key-based authentication: I generated an SSH key pair on my Windows machine with "ssh-keygen" and copied the public key into the VM using "ssh-copy-id," allowing login without a password. Once key authentication was working, I secured the configuration further by editing "/etc/ssh/sshd_config" and disabling password-based authentication, then restarting the SSH service. This ensures that only trusted machines with valid keys can access the VM, providing a more secure and professional setup.

- **Screenshots:**
  - SSH service status
  - Successful SSH connection and key-based authentication evidence


Linux-Project1 (Snapshot 1) [Running] - Oracle VirtualBox
File Machine View Input Devices Help
Nov 24 23:01
pelumi-j-ohnson@PelumiJohnson: ~
[-u user] file ...
pelumi-j-ohnson@PelumiJohnson:~$ sudo -i
[sudo] password for pelumi-j-ohnson:
root@PelumiJohnson:~# systemctl status ssh
● ssh.service - OpenBSD Secure Shell server
Loaded: loaded (/usr/lib/systemd/system/ssh.service; enabled; preset: enab
Active: active (running) since Mon 2025-11-24 22:22:48 EST; 1min 34s ago
TriggeredBy: ● ssh.socket
Docs: man:sshd(8)
man:sshd_config(5)
Process: 987 ExecStartPre=/usr/sbin/sshd -t (code=exited, status=0/SUCCESS)
Main PID: 1022 (sshd)
Tasks: 1 (limit: 2264)
Memory: 2.1M (peak: 2.3M)
CPU: 25ms
CGroup: /system.slice/ssh.service
└1022 "sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups"
Nov 24 22:22:48 PelumiJohnson systemd[1]: Starting ssh.service - OpenBSD Secure
Nov 24 22:22:48 PelumiJohnson sshd[1022]: Server listening on 0.0.0.0 port 22.
Nov 24 22:22:48 PelumiJohnson sshd[1022]: Server listening on :: port 22.
Nov 24 22:22:48 PelumiJohnson systemd[1]: Started ssh.service - OpenBSD Secure
root@PelumiJohnson:~# exit
logout
Home
Right Ctrl
C:\WINDOWS\system32\cmd.e
pelumi-j-ohnson@PelumiJohn
PS C:\Users\pelum> ssh pelumi-j-ohnson@localhost -p 2222
pelumi-j-ohnson@localhost's password:
Welcome to Ubuntu 24.04.3 LTS (GNU/Linux 6.14.0-36-generic x86_64)
* Documentation: https://help.ubuntu.com
* Management: https://landscape.canonical.com
* Support: https://ubuntu.com/pro
Expanded Security Maintenance for Applications is not enabled.
0 updates can be applied immediately.
11 additional security updates can be applied with ESM Apps.
Learn more about enabling ESM Apps service at https://ubuntu.com/esm
The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.
Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.
pelumi-j-ohnson@PelumiJohnson:~$ sudo systemctl status ssh
[sudo] password for pelumi-j-ohnson:
Warning: The unit file, source configuration file or drop-ins of ssh.service changed on disk. Run 'systemctl daemon-reload' to reload units.
● ssh.service - OpenBSD Secure Shell server
Loaded: loaded (/usr/lib/systemd/system/ssh.service; enabled; preset: enabled)
Active: active (running) since Mon 2025-11-24 22:22:48 EST; 17min ago
TriggeredBy: ● ssh.socket
Docs: man:sshd(8)
man:sshd_config(5)
Main PID: 1022 (sshd)
Tasks: 1 (limit: 2264)
Memory: 4.3M (peak: 5.2M)
CPU: 240ms
CGroup: /system.slice/ssh.service
└1022 "sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups"
Nov 24 22:28:28 PelumiJohnson sshd[2798]: pam_unix(sshd:auth): check pass; user unknown
Nov 24 22:28:29 PelumiJohnson sshd[2798]: Failed password for invalid user PelumiJohnson from 10.0.2.2 port 53187 ssh2
Search
10:40 PM
11/24/2025



---

3. Part 3: Configure the Firewall

ANSWER:

A **firewall** is a security tool that **controls what network traffic is allowed in or out** of a computer or system, helping protect it from unauthorized access.

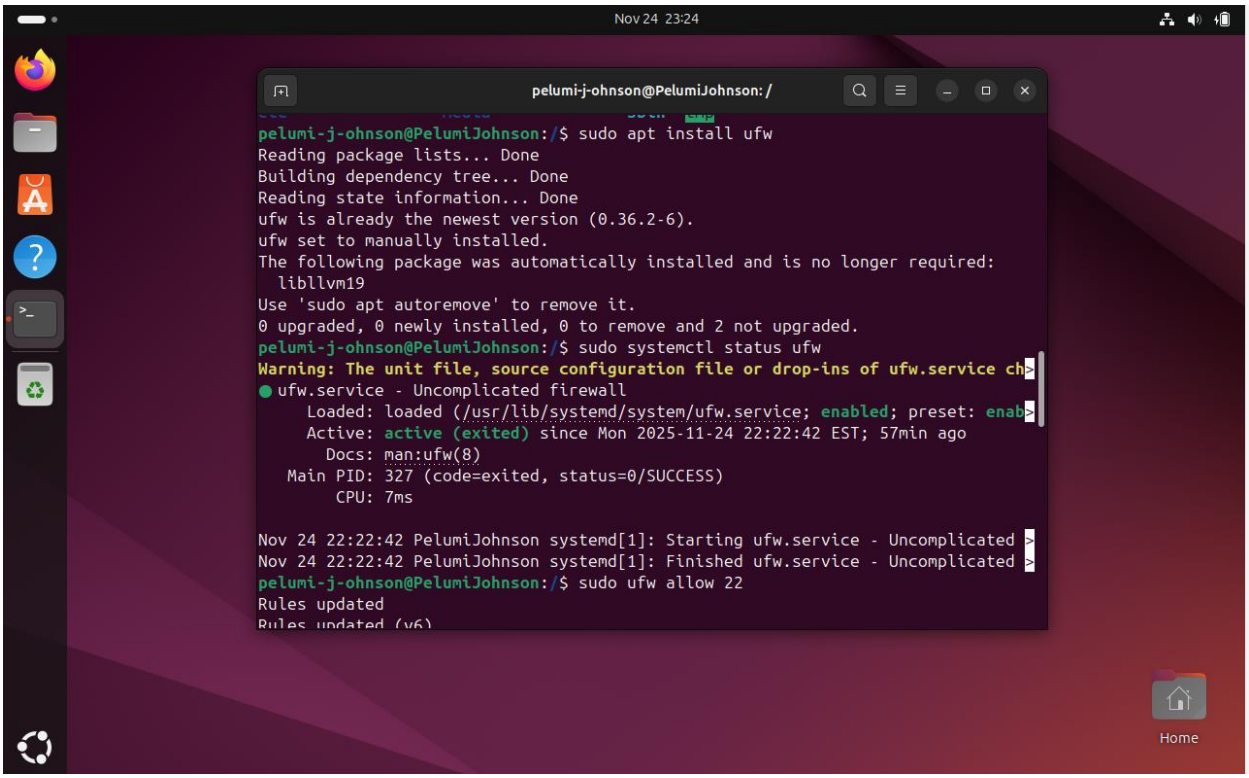
Outline the configuration of your firewall, including the allowed ports for SSH, HTTP, and HTTPS. To secure my Ubuntu virtual machine, I configured the Uncomplicated Firewall (UFW) to control which network services are allowed to communicate with the system. After installing and enabling UFW, I added rules to ensure only the necessary service ports were open. I allowed SSH so I could remotely access and administer the machine, and I enabled HTTP and HTTPS to support standard and secure web traffic. Each protocol serves a different purpose and is essential depending on the type of network activity being supported:

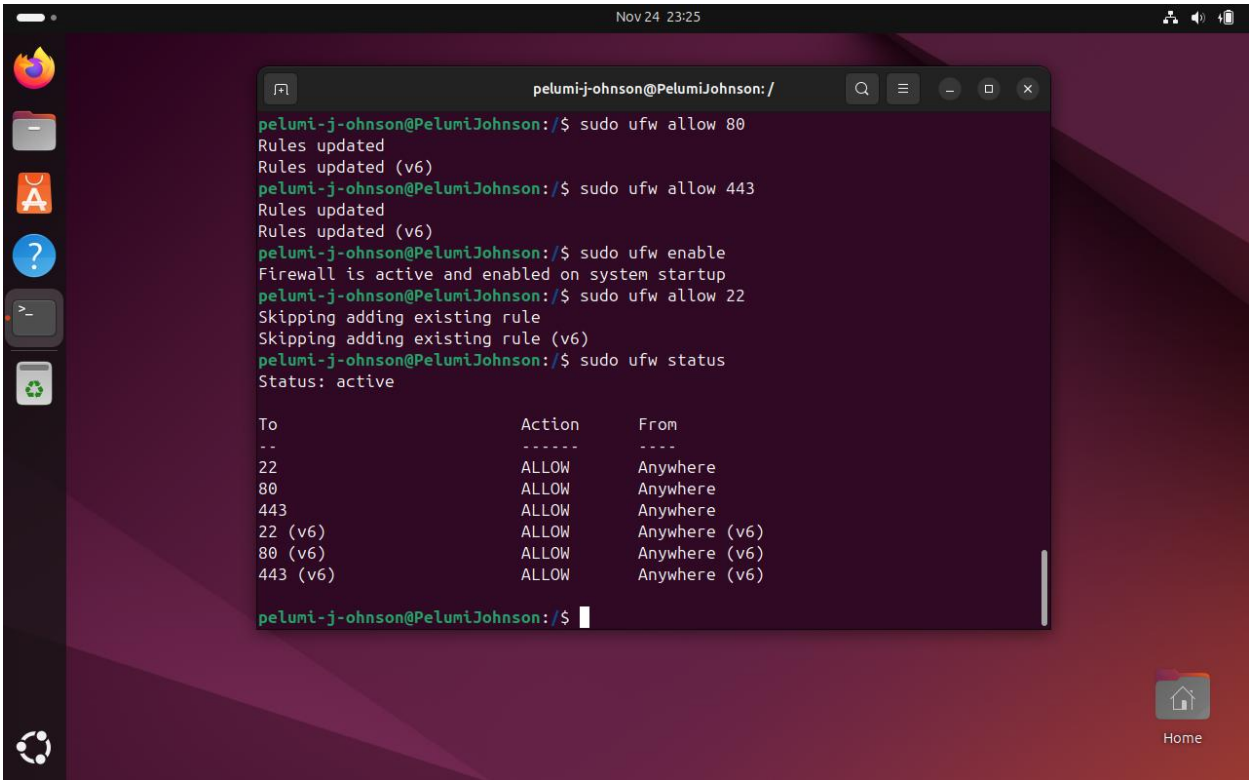
- **SSH (Port 22):** Provides encrypted remote access and secure command-line management.
- **HTTP (Port 80):** Handles unencrypted web traffic and delivers basic website content.

- **HTTPS (Port 443):** Encrypts web communication using SSL/TLS to protect data and ensure secure browsing. **SSL/TLS** are security protocols that encrypt data sent over the internet. SSL is the older version, and TLS is the newer, more secure version.

  After adding these rules, I enabled the firewall and verified the configuration using “sudo ufw status,” confirming that only ports 22, 80, and 443 were open. This restricted and focused setup strengthens the security posture of the VM by allowing only essential, trusted network traffic.

- **Screenshots:**
  - Display of the active firewall rules
  - Confirmation of firewall settings





---

4. Part 4: Implement Log Monitoring

ANSWER:

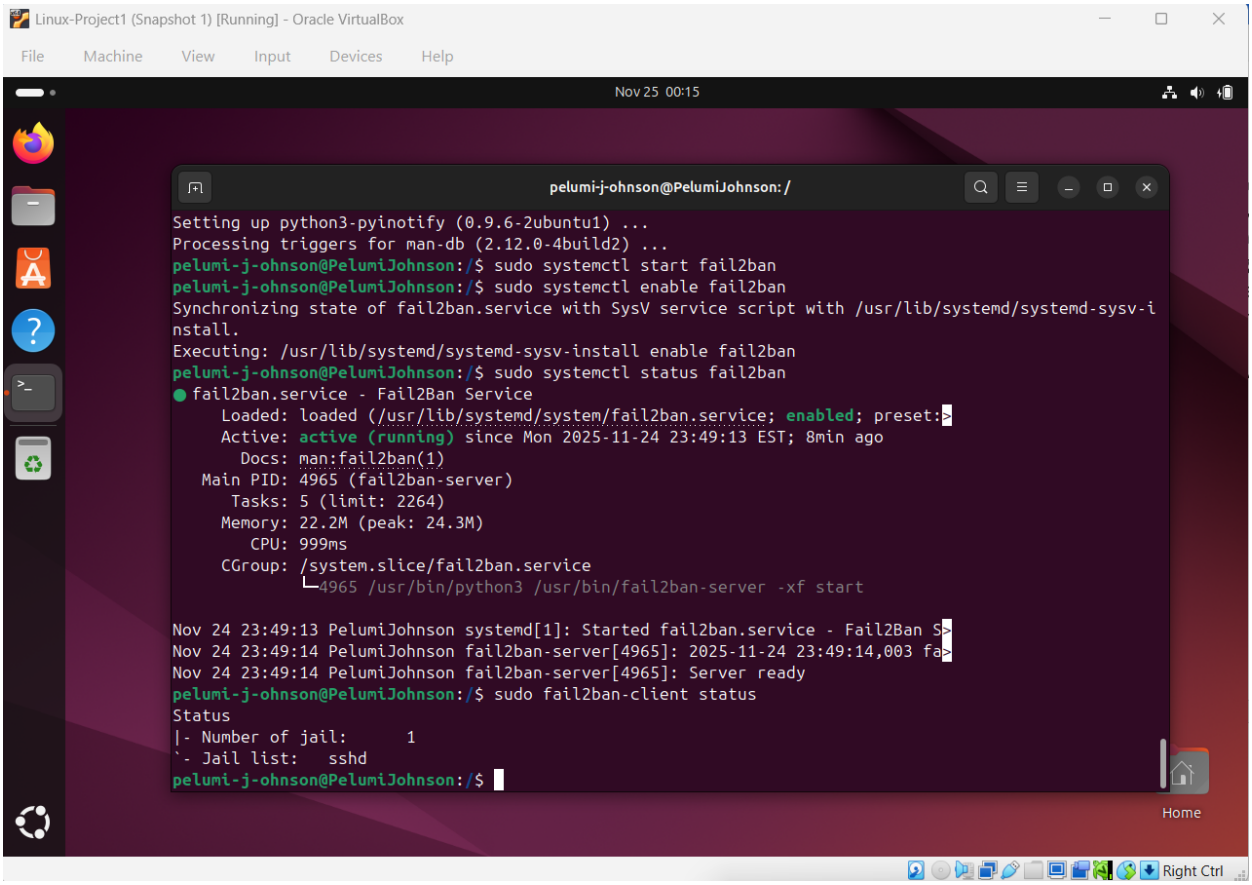
Describe how you implemented log monitoring, including the installation and verification of Fail2ban.

To strengthen the security of my virtual machine, I implemented log monitoring and intrusion prevention using Fail2ban. This tool scans system log files for repeated failed authentication attempts and automatically blocks any IP address that shows suspicious or brute-force behavior. I began by installing Fail2ban through the package manager using “sudo apt install fail2ban,” which added the service and its default configuration files to the system. After installation, I verified that the Fail2ban service was active by running “sudo systemctl status fail2ban,” confirming that it was loaded, enabled, and monitoring the system’s authentication logs. Fail2ban’s default jail configuration already includes protection for SSH, so it immediately began watching “/var/log/auth.log” for failed login attempts. By combining Fail2ban with the existing firewall and SSH security measures, my system can now automatically detect hostile login

behavior and temporarily ban attackers, providing continuous log monitoring and proactive defense without manual intervention.

Screenshots:

- Terminal output of log file monitoring (e.g., recent SSH logs)
- Fail2ban status showing its activity



---

5. Reflection and Lessons Learned

- **Challenges Encountered:**
  Discuss any issues you faced during the configuration process and how you resolved them.

ANSWER:

- One challenge I faced was that my SSH connection kept timing out. I later learned that, with VirtualBox in NAT mode, the host cannot reach the VM's internal IP directly. I fixed this by creating a port-forwarding rule that mapped host port 2222 to the VM's port 22, which finally allowed the SSH connection to work through localhost -p 2222.

- Another issue was repeated “Permission denied” errors when logging in through SSH. This happened because I was using the wrong username. Ubuntu had created a username with dashes (pelumi-j-ohnson), which I didn’t recognize at first. After checking with whoami and listing the /home directory, I identified the correct username and the login succeeded.

- Lastly, I was unsure whether Fail2ban was active because the output didn’t look like I expected. By using fail2ban-client status, I confirmed that the sshd jail was running and monitoring authentication logs properly. This helped me better understand how Fail2ban reports its activity and how to verify that intrusion protection is enabled.

Lessons Learned:
Reflect on what you learned from this project and how it might inform your future work.

- I learned the purpose of the /var directory, which stands for “variable.” It stores data that constantly changes while the system runs, such as logs in /var/log, caches, and runtime files. Understanding this helped me see how Linux tracks system activity and where important security logs are stored.
- I gained practical experience with security tools like Fail2ban and UFW, learning how log monitoring, automated bans, and firewall rules work together to protect a system from brute-force attacks and unauthorized access.
- I strengthened my overall Linux confidence by troubleshooting SSH issues, configuring port forwarding, using systemctl, managing services, and learning essential commands that will support my future work in cybersecurity and system administration.

---

6. References

List any resources, websites, or documentation you used to complete the project.

Ubuntu Community. “UFW Firewall Documentation.” *Ubuntu Documentation*.
https://help.ubuntu.com/community/UFW

Ubuntu Community. “SSH and Port Forwarding.” *Ubuntu Documentation*.
https://help.ubuntu.com/community/SSH/OpenSSH/PortForwarding