

**ASSIGNMENT COVER PAGE**

**Please fill in all the required details for your assignment to be accepted.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Student’s Name** | **Ekwonu Theresa Chidimma** | | |
| **Student’s Matric No** | **010240378** | | |
| **Year/Semester** | **Year 2/ Semester 8** | | |
| **Program** | **Software Engineering** | | |
| **Subject Name / Subject** | **BIT 6314** | | |
| **Lecturer’s Name** | **Mr Jafar** | | |
| **Assignment Title** | **Linux Administration task** | | |
| **No. of Page (excluding this page)** | **16** | | |
| **Required words** |  | **Actual  of words** |  |
| **Soft copy included** | **Yes                    /                 No** | | |
| **DECLARATION BY STUDENTS:**  ***I certify that this assignment is my own work in my own words. All resources have been acknowledged and the content has not been previously submitted for assessment to LINCOLN or elsewhere. I also confirm that I have kept a copy of this assignment.***  **Sign:                                                                      Date: 05/09/2025** | | | |

# Task 1: User & Group Management Automation (7 Marks) Write a Bash script that creates 5 users, assigns them to a group named 'devteam', sets initial passwords, and ensures they change their password upon first login.

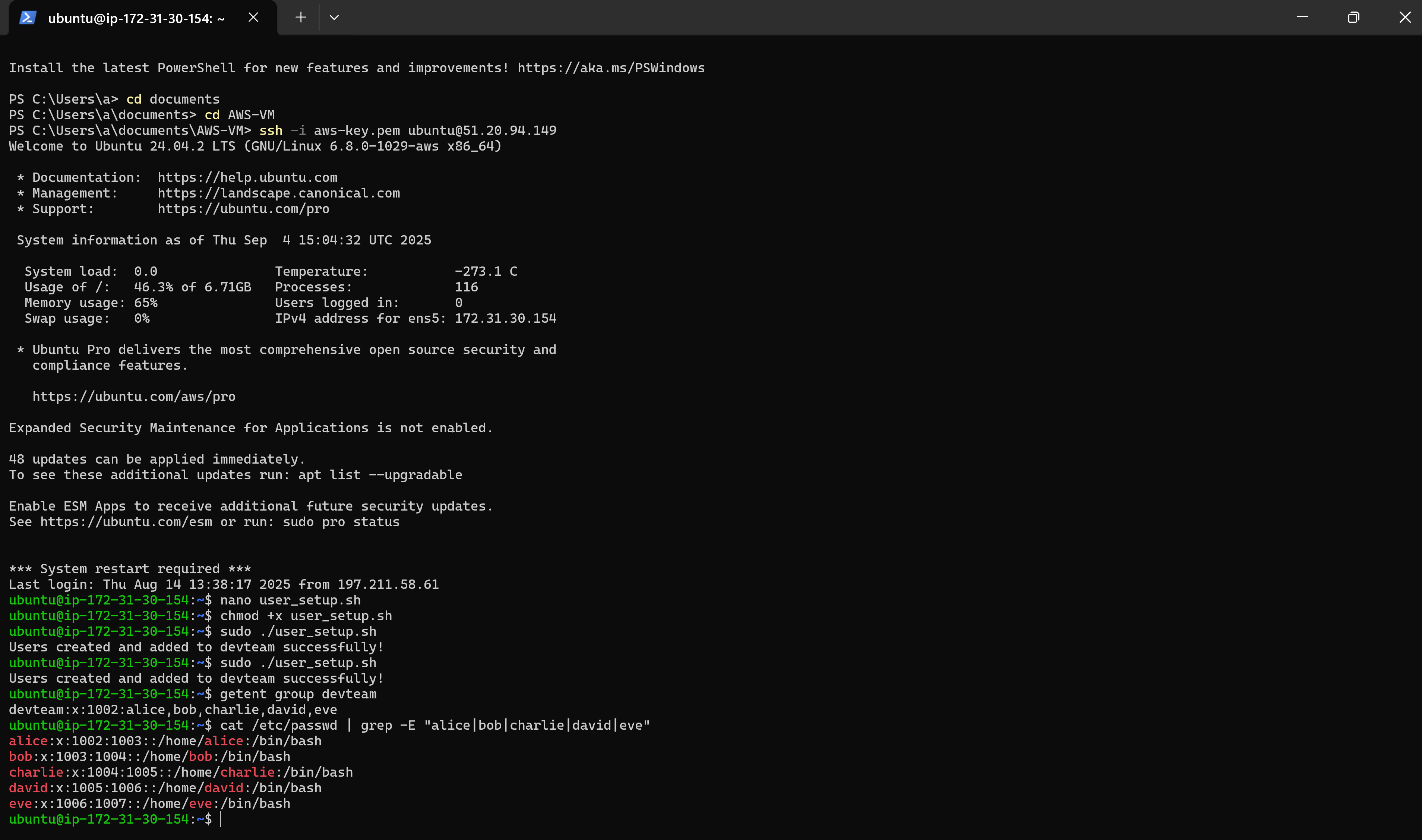
**Steps:**

1. Open your VPS terminal and create the script:  
 sudo nano user\_group\_management.sh

2. Save and exit (Ctrl+X, Y, Enter).

3. Make the script executable:  
 sudo chmod +x user\_group\_management.sh

4. Run the script:  
 sudo ./user\_group\_management.sh



# Task 2: File Permissions & ACLs Project (6 Marks)

Create a shared directory `/shared\_data` where group members can read/writebut not delete others' files. Use ACL to grant read-only access to one extra user outside the group

**Steps:**

1. Install ACL tools:  
 sudo apt update && sudo apt install acl -y

2. Create the shared directory:  
 sudo mkdir /shared\_data

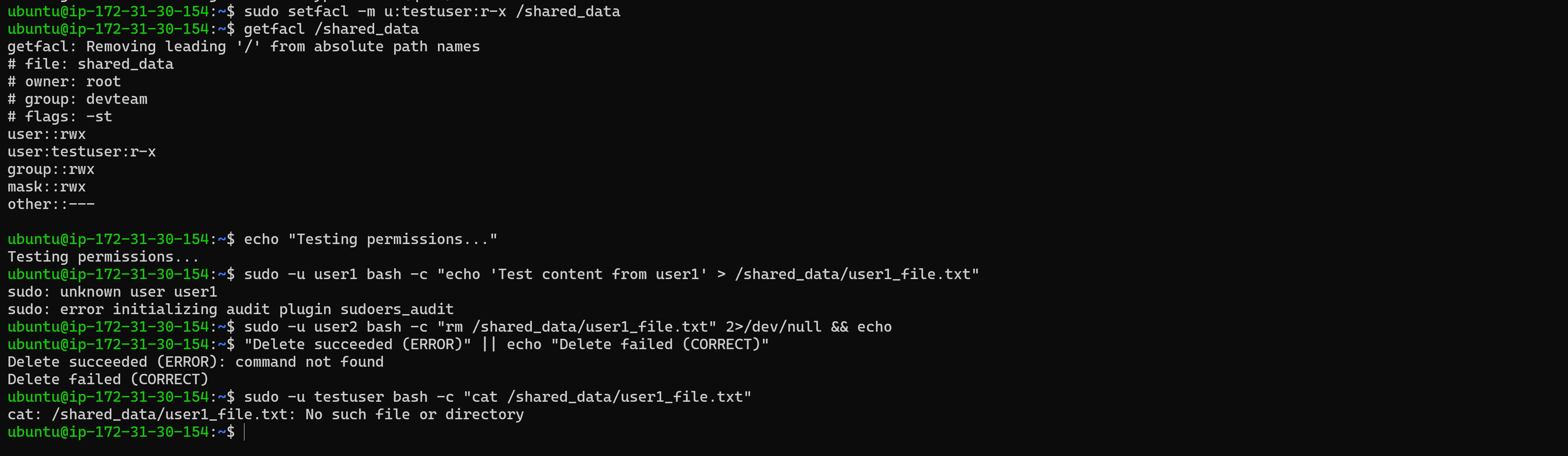
3. Create group and users:  
 sudo groupadd datagroup  
 sudo useradd -m user1  
 sudo useradd -m user2  
 sudo useradd -m readonly\_user

4. Add users to the group:  
 sudo usermod -aG datagroup user1  
 sudo usermod -aG datagroup user2

5. Set folder ownership and permissions:  
 sudo chown root:datagroup /shared\_data  
 sudo chmod 2775 /shared\_data

6. Enable sticky bit:  
 sudo chmod +t /shared\_data

7. Apply ACL for readonly\_user:  
 sudo setfacl -m u:readonly\_user:r-x /shared\_data



# Task 3: Apache Virtual Hosts Setup

Configure Apache to host two websites (`site1.local`, `site2.local`) with separate document roots and logs.

**Steps:**

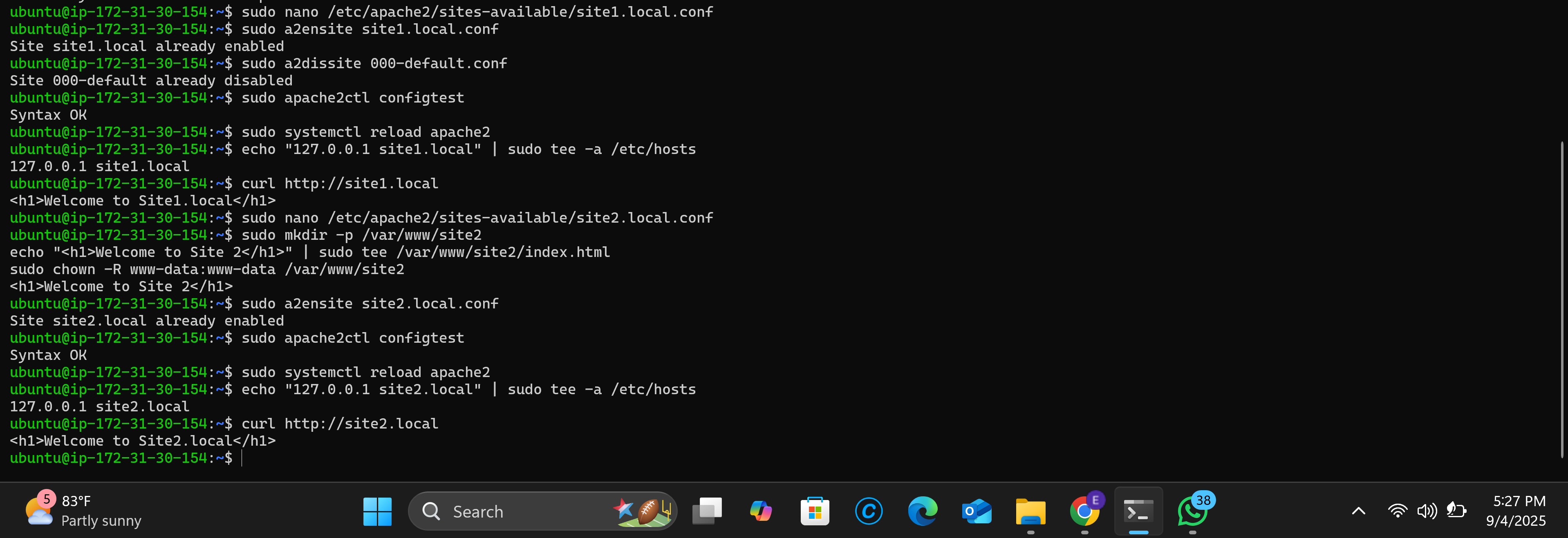
1. Create a script:  
 nano apache\_virtual\_hosts\_setup.sh

2. Make it executable:  
 chmod +x apache\_virtual\_hosts\_setup.sh

3. Run the script:  
 sudo ./apache\_virtual\_hosts\_setup.sh

4. Add host entries:  
 sudo nano /etc/hosts

5. Test with curl:  
 curl http://site1.local



# Task 4: SSL/TLS Configuration

Generate a self-signed SSL certificate using `openssl` and enable HTTPS for one of your virtual hosts.

**Steps:**

1. Confirm OpenSSL installation:  
 openssl version

2. Create directories for certificates:  
 sudo mkdir -p /etc/ssl/private /etc/ssl/certs

3. Generate private key:  
 sudo openssl genrsa -out /etc/ssl/private/example.com.key 2048

4. Create CSR:  
 sudo openssl req -new -key /etc/ssl/private/example.com.key -out /etc/ssl/certs/example.com.csr

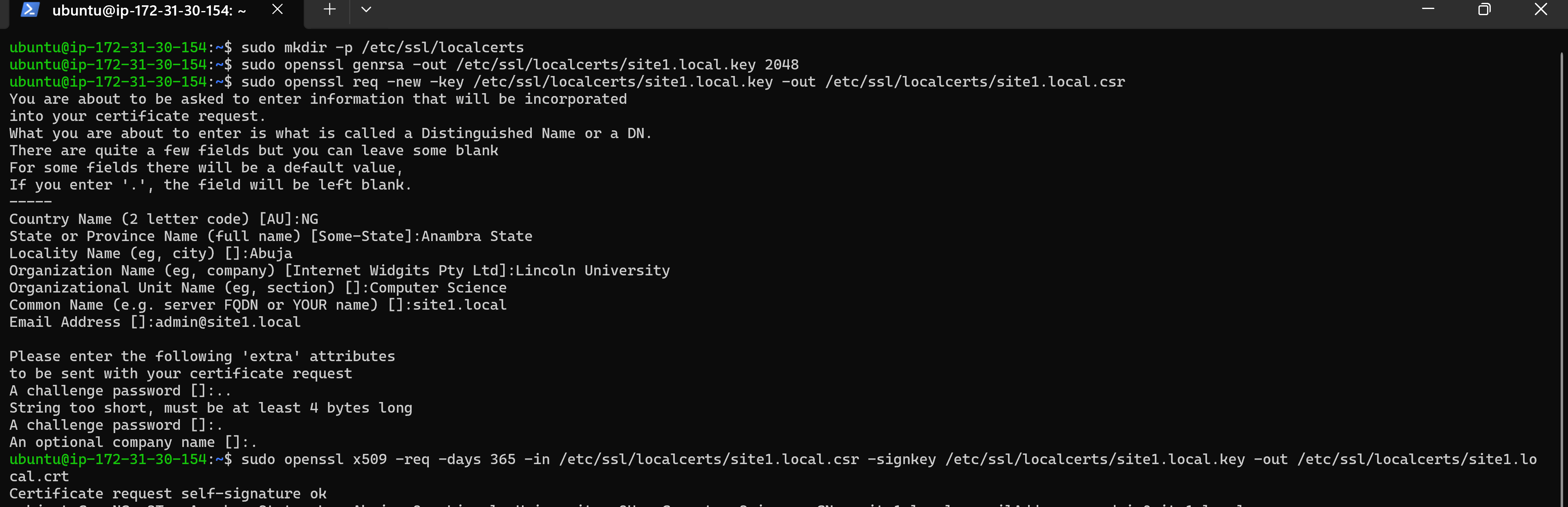
5. Generate certificate:  
 sudo openssl x509 -req -days 365 -in /etc/ssl/certs/example.com.csr -signkey /etc/ssl/private/example.com.key -out /etc/ssl/certs/example.com.crt

6. Enable Apache SSL module:  
 sudo a2enmod ssl

7. Create SSL virtual host configuration.

8. Enable the site and restart Apache:  
 sudo a2ensite example.com-ssl.conf  
 sudo systemctl restart apache2

9. Verify SSL with OpenSSL client:  
 openssl s\_client -connect example.com:443



# Task 5: MySQL Remote Access & Security

Configure MySQL to allow secure remote connections, create a database and user with least privilege for remote access.  
 **Steps:**

1. Check MySQL status:  
 sudo systemctl status mysql

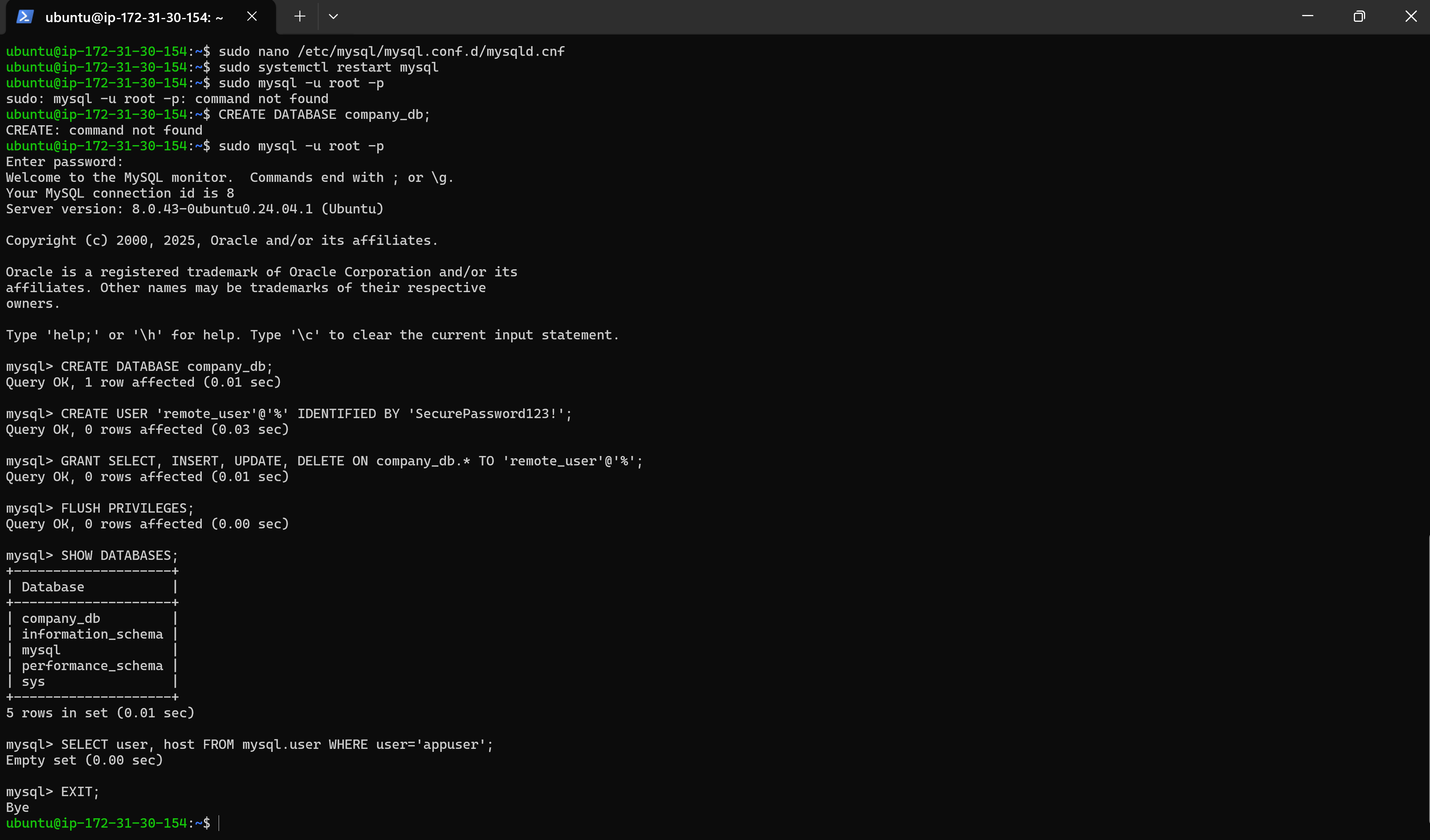
2. Edit configuration file:  
 sudo nano /etc/mysql/mysql.conf.d/mysqld.cnf

3. Change bind-address to 0.0.0.0.

4. Restart MySQL:  
 sudo systemctl restart mysql

5. Secure installation:  
 sudo mysql\_secure\_installation

6. Create database and remote user with limited privileges.



# Task 6: Firewall Configuration

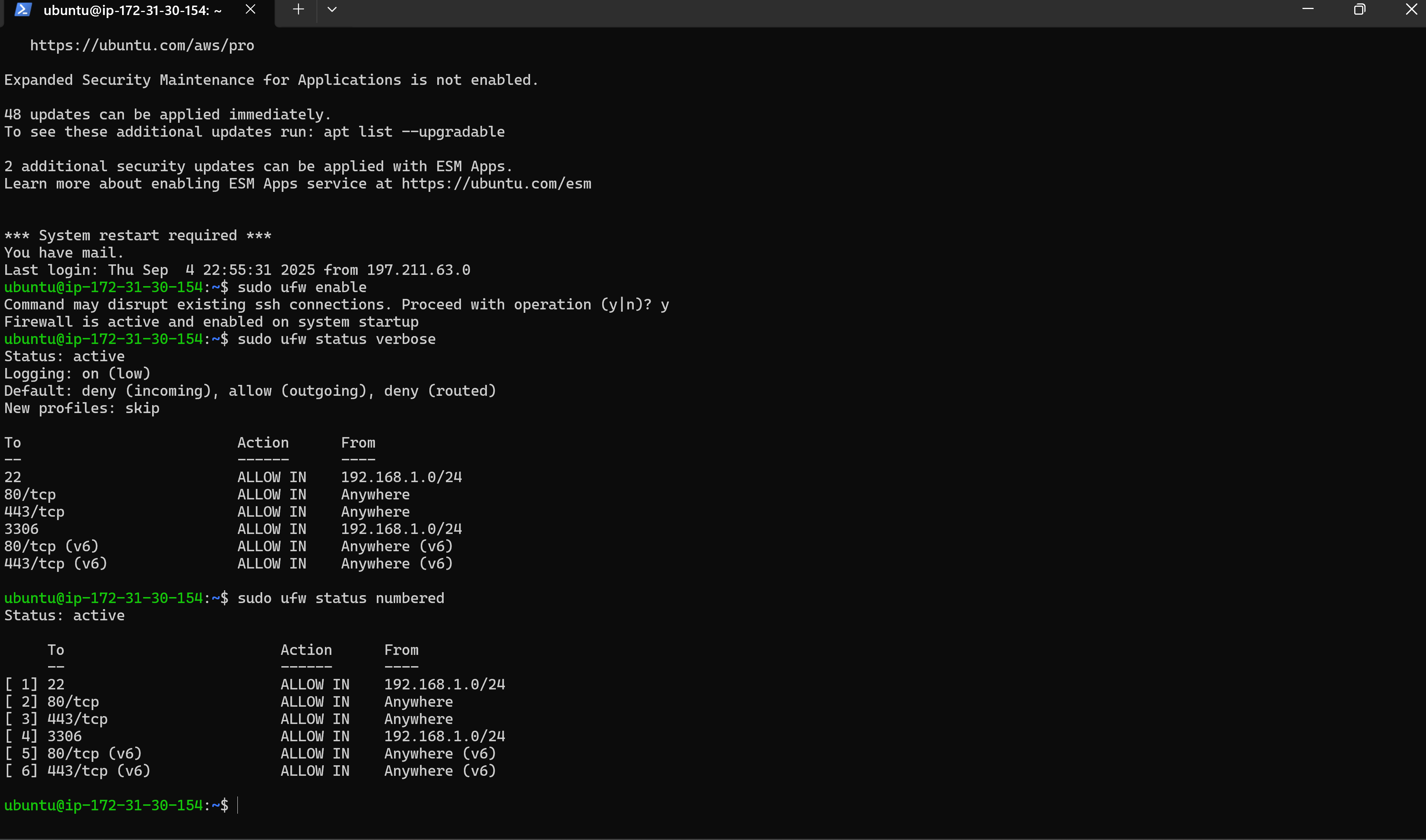
Configure `ufw` or `iptables` to allow only HTTP(80), HTTPS(443), SSH(22), and mysql(3306) from a specific IP range.

**Steps:**

1. Check and install UFW if needed.

2. Reset UFW to defaults.

3. Define allowed IP ranges and apply allow rules.



# Task 7: System Monitoring Script

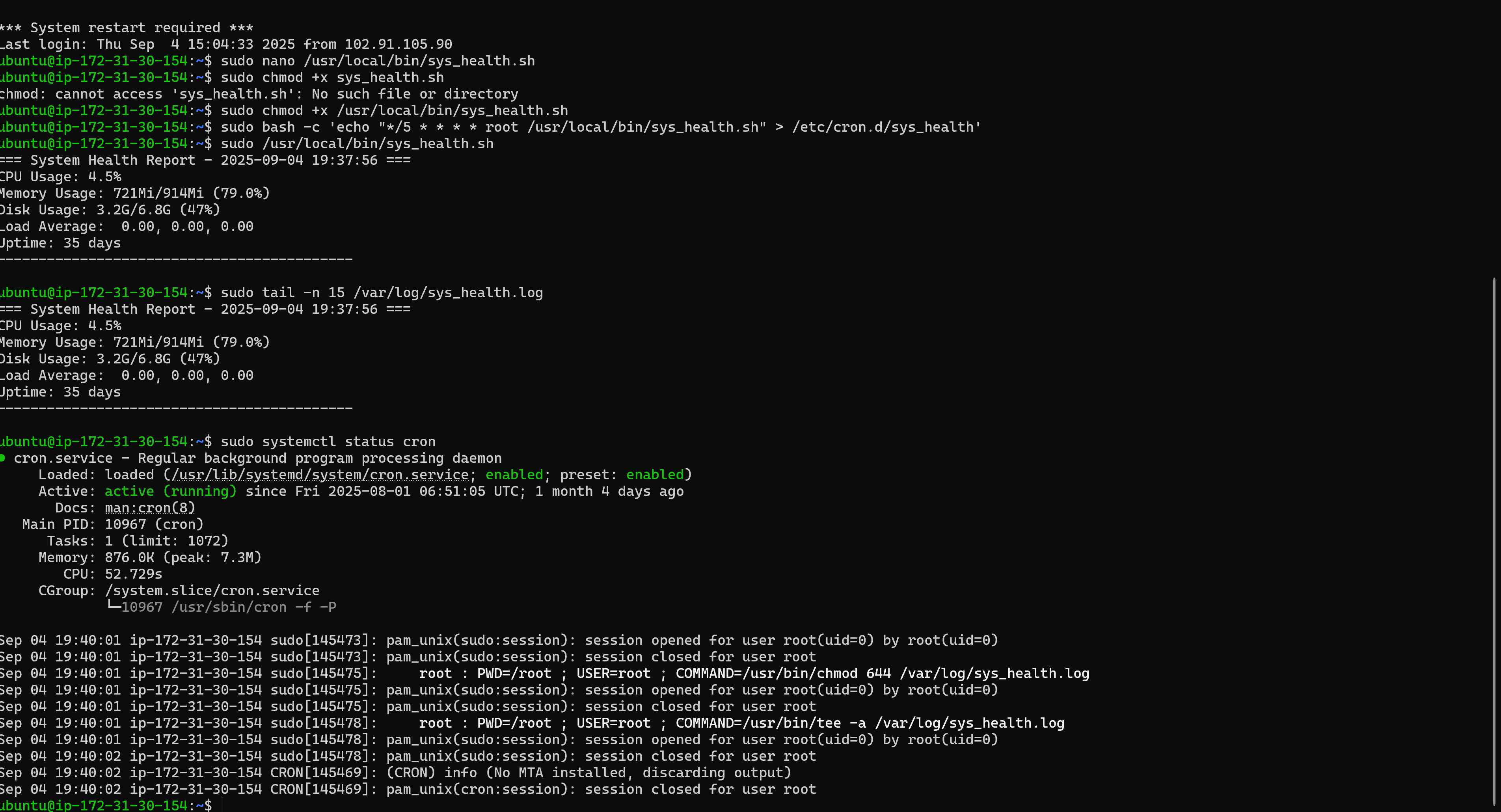
Write a script to log CPU, Memory, and Disk usage every 5 minutes into `/var/log/sys\_health.log` and set as a cron job.

**Steps:**

1. Create monitoring script:  
 nano system\_monitor\_complete.sh

2. Make it executable:  
 chmod +x system\_monitor\_complete.sh

3. Set up cron job using the script's setup mode.



# Task 8: Log Rotation Setup

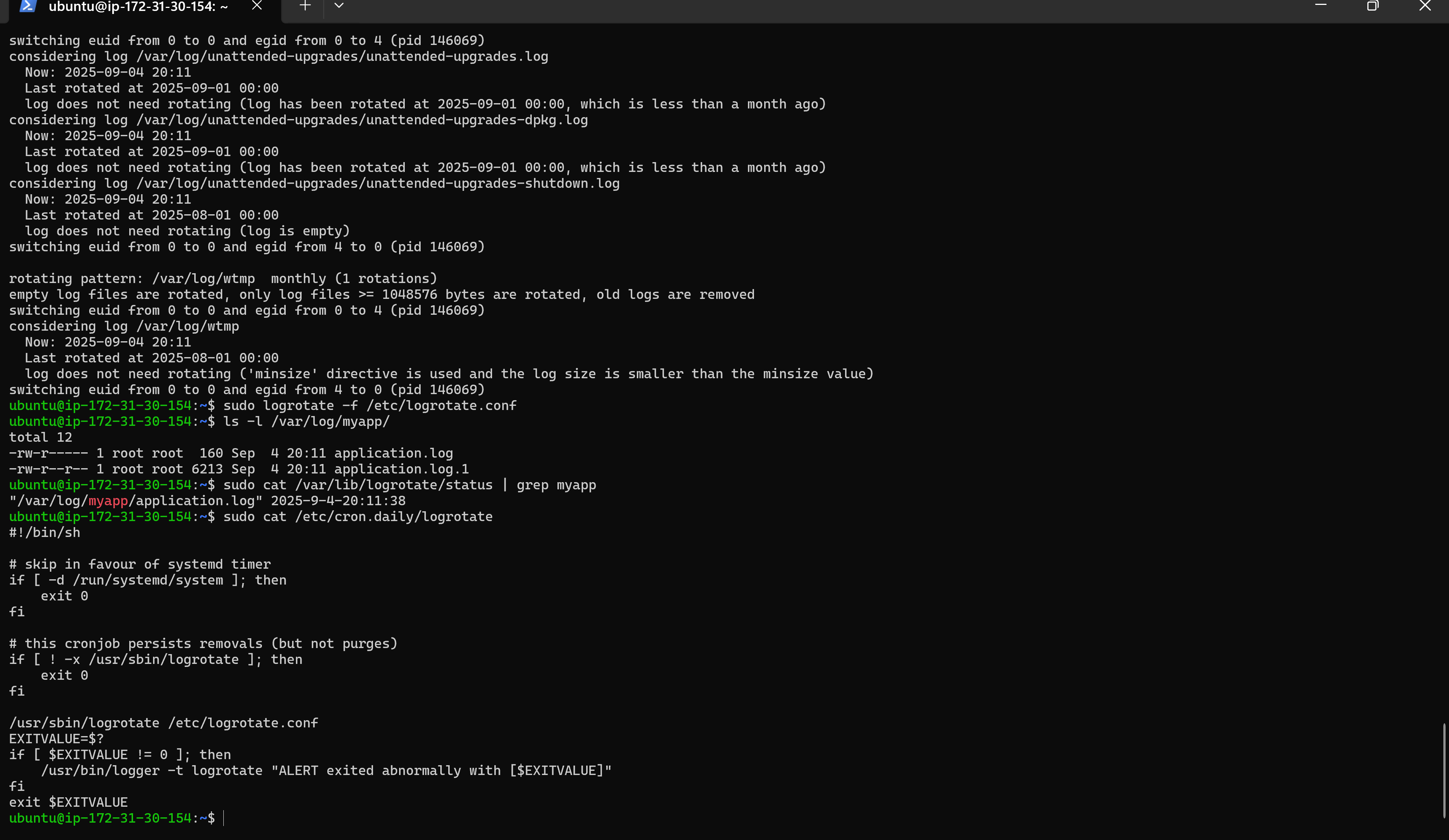
Configure `logrotate` for a custom application log to rotate daily, compress old logs, and keep only 7 days.

**Steps:**

1. Create application log file at /var/log/myapp/application.log.

2. Configure logrotate at /etc/logrotate.d/myapp.

3. Test with:  
 sudo logrotate -d /etc/logrotate.d/myapp



# Task 9: DNS Server Setup

Install and configure `bind9` as a local caching DNS server with a custom zone for `myuniversity.local`.

**Steps:**

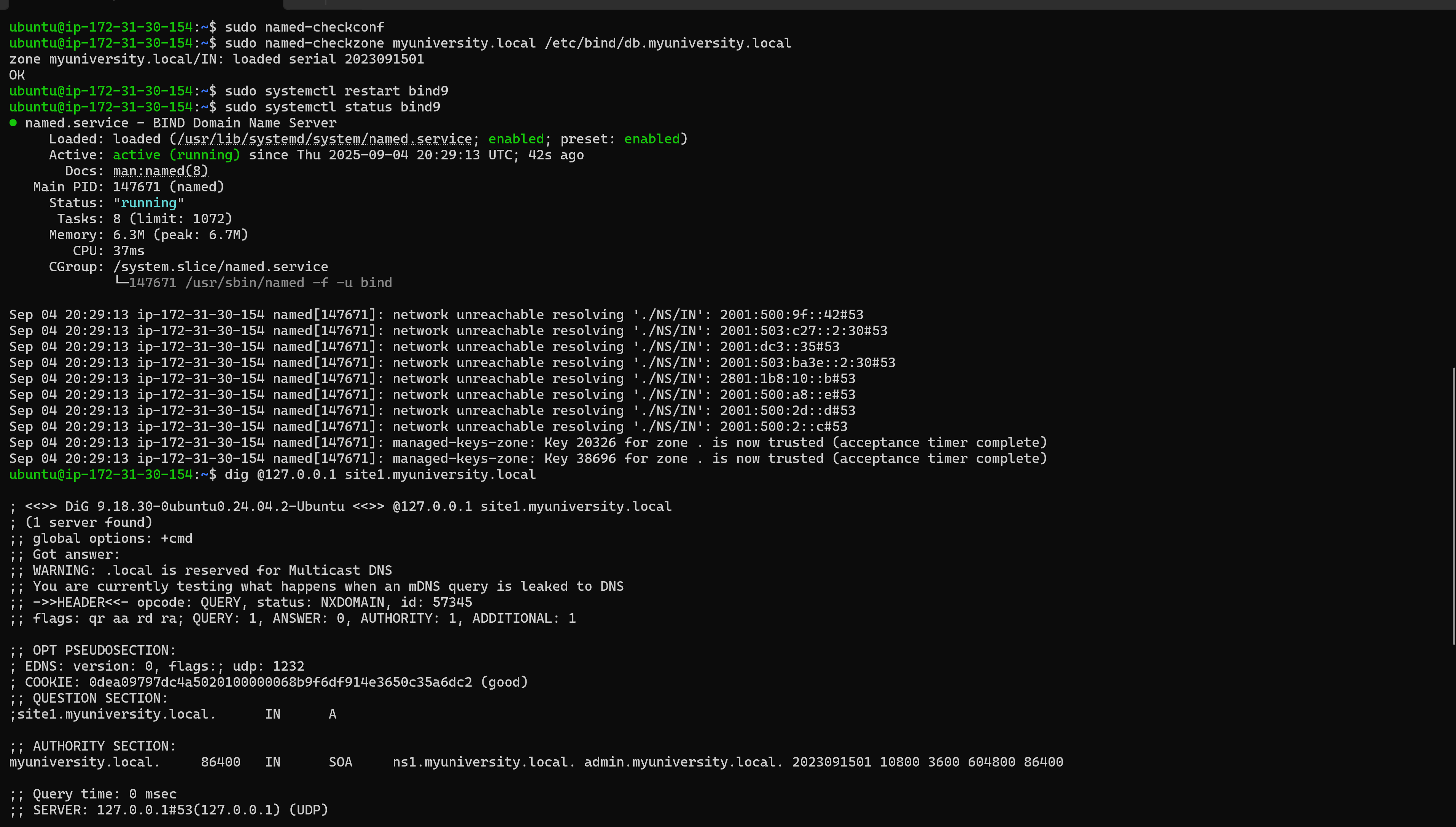
1. Install BIND9 and tools.

2. Configure options in /etc/bind/named.conf.options.

3. Create zone files in /etc/bind/zones/

4. Start and enable bind9 service.

5. Test with nslookup and dig.



# Task 10: SSH Key Authentication and Hardening

Configure SSH key-based login, disable password authentication, and disable root login in`sshd\_config`**.**

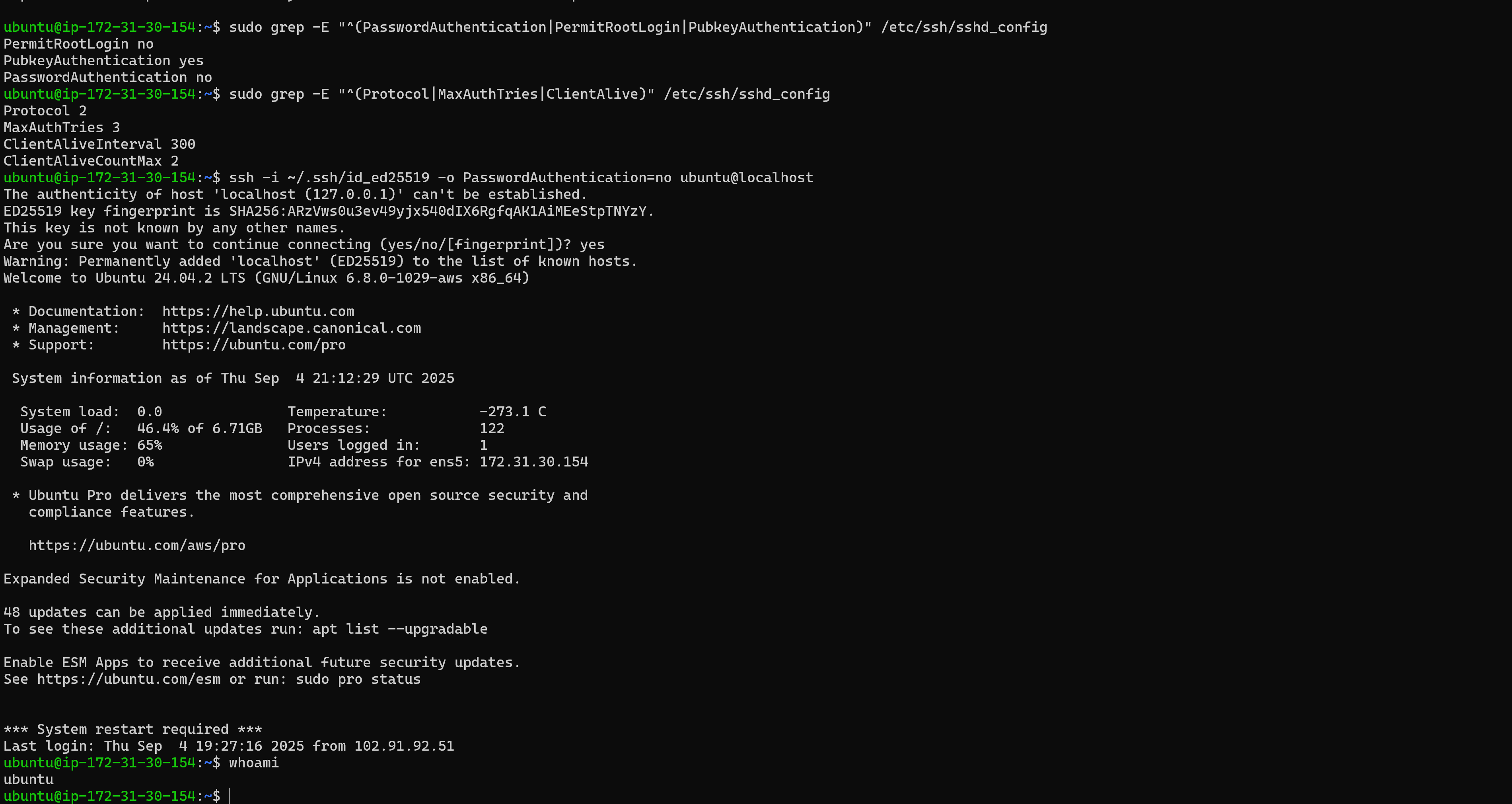
**Steps:**

1. Generate SSH key pair on client.

2. Copy public key to server.

3. Disable root and password authentication in sshd\_config.

4. Restart SSH service and test.



# Task 11: Systemd Service Creation

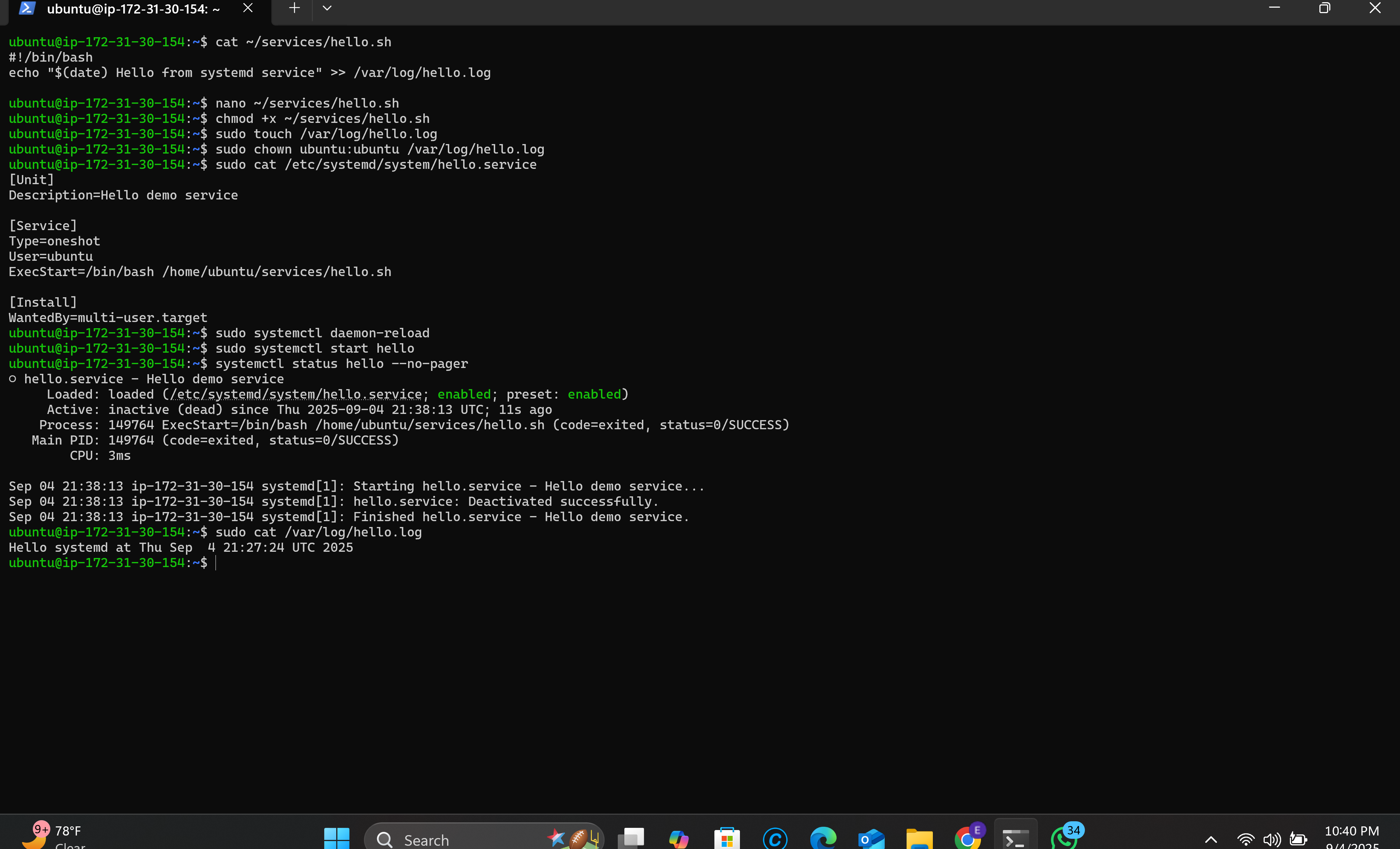
Write a simple script and create a `systemd` service to run it automatically at boot.

**Steps:**

1. Write script under /usr/local/bin/

2. Create service file under /etc/systemd/system/

3. Reload systemd and enable service.



# Task 12: Disk Partitioning & Mounting

Create a new partition, format as ext4, mount it permanently using `/etc/fstab`, and test reboot persistence.   
 **Steps:**

1. Create virtual disk using dd.

2. Attach with losetup and partition with fdisk.

3. Format partition as ext4.

4. Add entry to /etc/fstab and test.

# Task 13: Postfix Mail Server (Local Only)

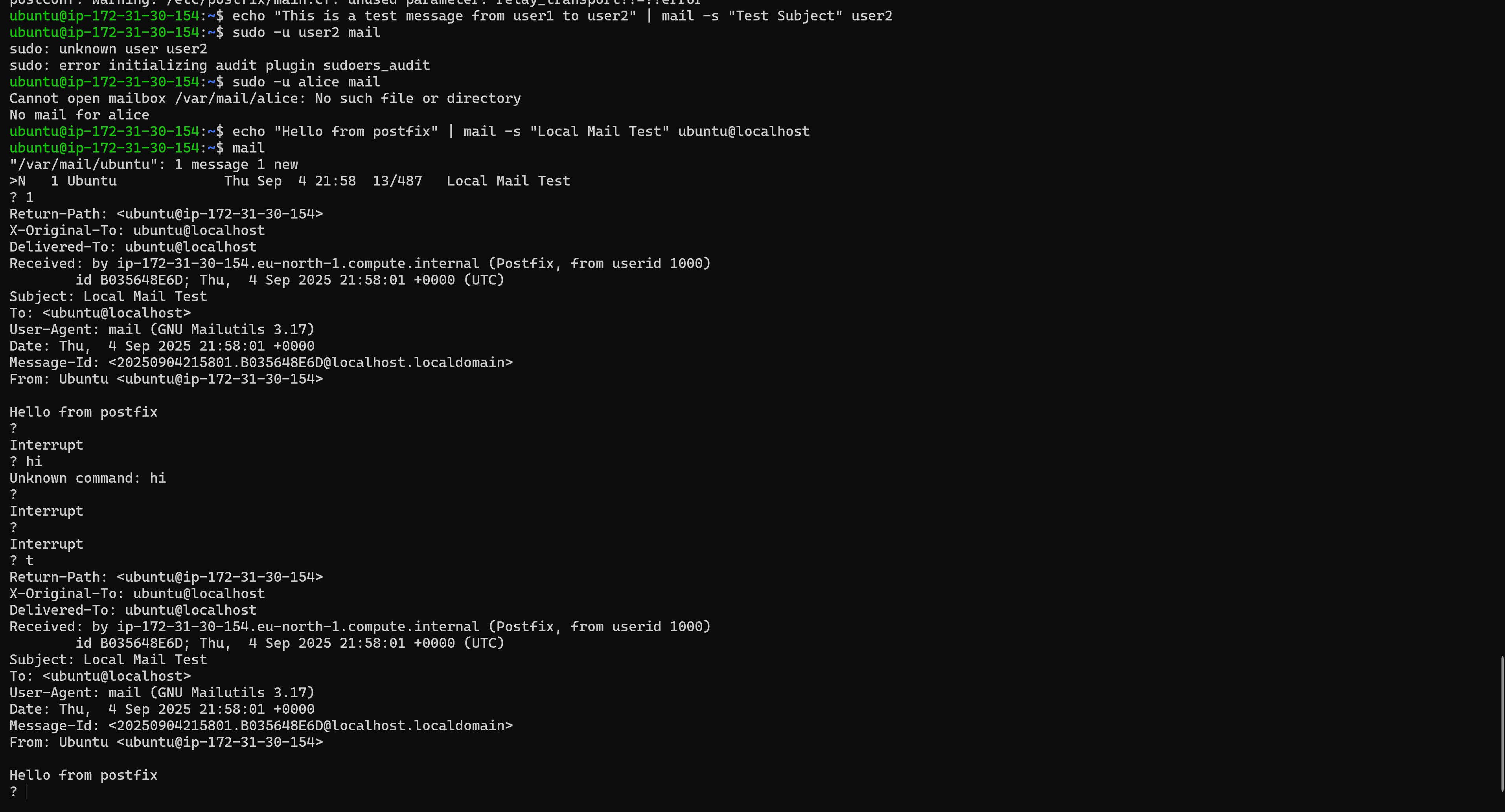
Install and configure Postfix for local mail delivery and send a test mail between users

**Steps:**

1. Install postfix and mailutils.

2. Configure /etc/postfix/main.cf for local delivery.

3. Restart postfix and test sending emails between test users.



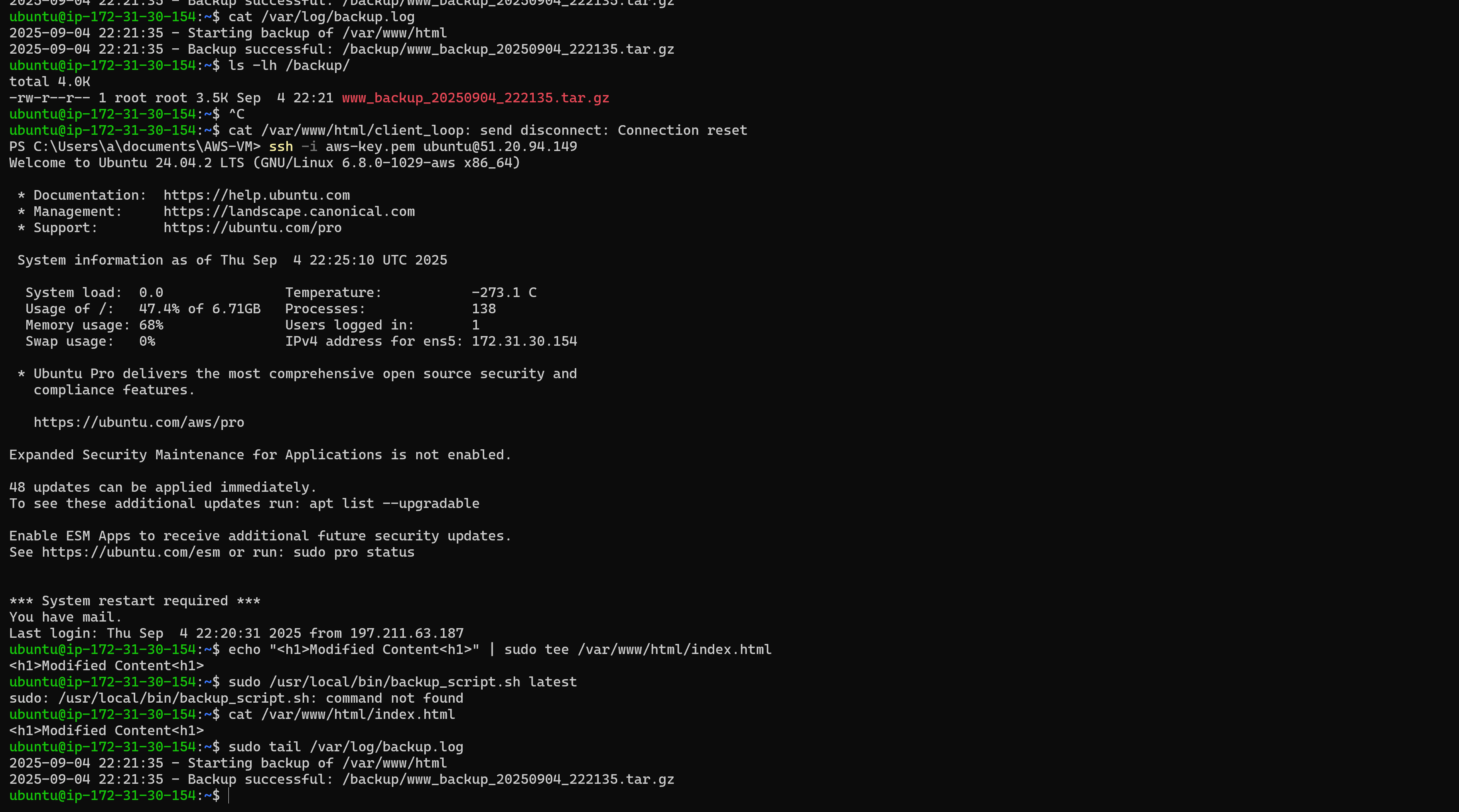
# Task 14: Backup & Restore Project

Write a script to back up `/var/www/html` to `/backup/` with a timestamp and test restoring.  
 **Steps:**

1. Create backup and restore scripts in /usr/local/bin/.

2. Make them executable.

3. Run backup and restore, and verify restored files.



# Task 15: Containerization Challenge

Install Docker/Podman, create a container running Nginx, map it to port 8080, and verifyservice.

**Steps:**

1. Install Docker.

2. Pull nginx:latest image.

3. Run container mapping 8080:80.

4. Verify with curl http://localhost:8080

