



## **ASSIGNMENT COVER PAGE**

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

<b><u>Student's Name</u></b>	Daniel_PatrickMmasichukwu		
<b><u>Student's Matric No</u></b>	0102303110		
<b><u>Year/Semester</u></b>	Year 3/ Semester 8		
<b><u>Program</u></b>	Software Engineering		
<b><u>Subject Name / Subject</u></b>	BIT 6314		
<b><u>Lecturer's Name</u></b>	Mr jafar		
<b><u>Assignment Title</u></b>	Linux task		
<b><u>No. of Page (excluding this page)</u></b>	1		
<b><u>Required words</u></b>		<b><u>Actual words</u></b>	
<b><u>Soft copy included</u></b>	Yes	/	No <input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>
<b><u>DECLARATION BY STUDENTS:</u></b>			
<i>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.</i>			
<b>Sign: Daniel</b>		<b>Date: 07/09/2025</b>	

# 1, User & Group Management Automation (7 Marks)

Write a Bash script that will automate the following tasks:

- Create 5 new users
  - Add them to a group called `devteam`
  - Set their passwords
  - Force them to change their password upon first login
- 

## Step-by-Step Instructions

### Step 1: Create the Script File

After connecting to your VPS, run the following command to create and open the script file:

```
sudo nano user_group_management.sh
```

Once the nano editor opens:

- Write or paste your script
- To save and exit:
  - Press *Cmd + X* (or *Ctrl + X* on most keyboards)
  - Press *Y* to confirm saving
  - Then press *Enter* to finalize

## Step 2: Make the Script Executable

Run this command to give the script execution permission:

```
sudo chmod +x user_group_management.sh
```

## Step 3: Execute the Script

Finally, run the script using:

```
sudo ./user_group_management.sh
```

Let me know if you'd like help writing or formatting the actual script content inside `user_group_management.sh`.

```
User 'dev3' setup completed
-----
Processing user: dev4
Successfully created user 'dev4'
Setting password for user 'dev4'...
Password set for user 'dev4'...
Configuring password expiry for user 'dev4'...
Password expiry configured for user 'dev4'
User 'dev4' setup completed

-----
Processing user: dev5
Successfully created user 'dev5'
Setting password for user 'dev5'...
Password set for user 'dev5'...
Configuring password expiry for user 'dev5'...
Password expiry configured for user 'dev5'
User 'dev5' setup completed

==== SETUP SUMMARY ====
Group Information:
devteam:x:1001:dev1,dev2,dev3,dev4,dev5

User Information:
User: dev1
- Groups: dev1 devteam
- Home Directory: /home/dev1
- Shell: /bin/bash
- Password Status: P

User: dev2
- Groups: dev2 devteam
- Home Directory: /home/dev2
- Shell: /bin/bash
- Password Status: P

User: dev3
- Groups: dev3 devteam
- Home Directory: /home/dev3
- Shell: /bin/bash
- Password Status: P

User: dev4
- Groups: dev4 devteam
- Home Directory: /home/dev4
- Shell: /bin/bash
- Password Status: P

User: dev5
- Groups: dev5 devteam
- Home Directory: /home/dev5
- Shell: /bin/bash
- Password Status: P

Script execution completed successfully!
Note: All users have been set with default password 'TempPass123!'
Users will be forced to change password on first login
Do you want to create a cleanup script? (y/n): ^[OA
All tasks completed successfully!
ubuntu@ip-172-31-31-88:~$ chmod danny-key.pem
```

## 2, File Permissions & ACLs Project (6 Marks)

Create a shared directory `/shared_data` where group members can read and write files, but cannot delete files created by others. Additionally, use ACL (Access Control Lists) to give **read-only access** to one extra user outside the group.

### Step 1: Install ACL Utilities

Update the package index and install ACL:

```
sudo apt update  
sudo apt install acl -y
```

```

Get:27 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/main amd64 Packages [39.9 kB]
Get:28 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/main Translation-en [9152 B]
Get:29 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/main amd64 Components [7076 B]
Get:30 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/main amd64 c-n-f Metadata [272 B]
Get:31 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/universe amd64 Packages [30.2 kB]
Get:32 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/universe Translation-en [17.4 kB]
Get:33 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/universe amd64 Components [19.2 kB]
Get:34 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/universe amd64 c-n-f Metadata [1304 B]
Get:35 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/restricted amd64 Components [216 B]
Get:37 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/multiverse amd64 Components [212 B]
Get:38 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-backports/multiverse amd64 c-n-f Metadata [116 B]
Get:39 http://security.ubuntu.com/ubuntu noble-security InRelease [126 kB]
Get:40 http://security.ubuntu.com/ubuntu noble-security/main amd64 Packages [1118 kB]
Get:41 http://security.ubuntu.com/ubuntu noble-security/main Translation-en [191 kB]
Get:42 http://security.ubuntu.com/ubuntu noble-security/main amd64 Components [21.6 kB]
Get:43 http://security.ubuntu.com/ubuntu noble-security/main amd64 c-n-f Metadata [8712 B]
Get:44 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Packages [879 kB]
Get:45 http://security.ubuntu.com/ubuntu noble-security/universe Translation-en [194 kB]
Get:46 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Components [52.3 kB]
Get:47 http://security.ubuntu.com/ubuntu noble-security/universe amd64 c-n-f Metadata [18.0 kB]
Get:48 http://security.ubuntu.com/ubuntu noble-security/restricted amd64 Packages [1705 kB]
Get:49 http://security.ubuntu.com/ubuntu noble-security/restricted Translation-en [382 kB]
Get:50 http://security.ubuntu.com/ubuntu noble-security/restricted amd64 Components [208 B]
Get:51 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 Packages [18.5 kB]
Get:52 http://security.ubuntu.com/ubuntu noble-security/multiverse Translation-en [4288 B]
Get:53 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 Components [208 B]
Get:54 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 c-n-f Metadata [380 B]
Fetched 37.0 Mb in 6s (6564 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
13 packages can be upgraded. Run 'apt list --upgradable' to see them.
ubuntu@ip-172-31-31-80:~$ sudo apt install acl -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  acl
0 upgraded, 1 newly installed, 0 to remove and 13 not upgraded.
Need to get 39.4 kB of archives.
After this operation, 197 kB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 acl amd64 2.3.2-1build1.1 [39.4 kB]
Fetched 39.4 kB in 0s (2617 kB/s)
Selecting previously unselected package acl.
(Reading database ... 71718 files and directories currently installed.)
Preparing to unpack .../acl_2.3.2-1build1.1_amd64.deb ...
Unpacking acl (2.3.2-1build1.1) ...
Setting up acl (2.3.2-1build1.1) ...
Processing triggers for man-db (2.12.0-4build2) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-31-80:~$ █

```

## Step 2: Create the Shared Directory

Create the directory that will be shared:

```
sudo mkdir /shared_data
```

## Step 3: Create Group and Users

Create a new group and three test users:

```
sudo groupadd datagroup
sudo useradd -m user1
```

```
sudo useradd -m user2  
sudo useradd -m readonly_user
```

## Step 4: Add Users to the Group

Add `user1` and `user2` to the `datagroup`:

```
sudo usermod -aG datagroup user1  
sudo usermod -aG datagroup user2
```

## Step 5: Set Directory Ownership and Permissions

Assign group ownership of the shared directory and set permissions:

```
sudo chown root:datagroup /shared_data  
sudo chmod 2775 /shared_data
```

The `2` in `2775` sets the **setgid** bit, ensuring that new files inherit the group.

## Step 6: Set Sticky Bit

Set the **sticky bit** to prevent users from deleting each other's files:

```
sudo chmod +t /shared_data
```

## Step 7: Verify Directory Permissions

Check the current permissions:

```
ls -la /shared_data
```

```

Reading state information... Done
13 packages can be upgraded. Run 'apt list --upgradable' to see them.
ubuntu@ip-172-31-31-88:~$ sudo apt install acl -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  acl
0 upgraded, 1 newly installed, 0 to remove and 13 not upgraded.
Need to get 39.4 kB of archives.
After this operation, 107 kB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu noble-updates/main amd64 acl amd64 2.3.2-1build1.1 [39.4 kB]
Fetched 0 kB in 0s (2617 kB/s)
Selecting previously unselected package acl.
(Reading database .../acl_2.3.2-1build1.1_amd64.deb ...
Preparing to unpack .../acl_2.3.2-1build1.1_amd64.deb ...
Unpacking acl (2.3.2-1build1.1) ...
Setting up acl (2.3.2-1build1.1) ...
Processing triggers for man-db (2.12.0-4build2) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-31-88:~$ ls -la /shared_data
ls: cannot access '/shared_data': No such file or directory
ubuntu@ip-172-31-31-88:~$ sudo mkdir /shared_data

ubuntu@ip-172-31-31-88:~$ sudo groupadd datagroup
ubuntu@ip-172-31-31-88:~$ sudo useradd -m user1
ubuntu@ip-172-31-31-88:~$ sudo useradd -m user2
ubuntu@ip-172-31-31-88:~$ sudo useradd -m readonly_user
ubuntu@ip-172-31-31-88:~$ sudo usermod -aG datagroup user1
ubuntu@ip-172-31-31-88:~$ sudo usermod -aG datagroup user2

ubuntu@ip-172-31-31-88:~$ sudo chown root:datagroup /shared_data
ubuntu@ip-172-31-31-88:~$ sudo chmod 2775 /shared_data
ubuntu@ip-172-31-31-88:~$ sudo chmod +t /shared_data
ubuntu@ip-172-31-31-88:~$ ls -la /shared_data
total 8
drwxrwsr-t 2 root datagroup 4096 Sep  5 20:47 .
drwxr-xr-x 23 root root   4096 Sep  5 20:47 ..
ubuntu@ip-172-31-31-88:~$ █

```

You should see something like:

```
drwxrwsr-t 2 root datagroup 4096 ... /shared_data
```

## Step 8: Set ACL for Read-Only User

Grant `readonly_user` read and execute permissions:

```
sudo setfacl -m u:readonly_user:r-x /shared_data
sudo setfacl -m d:u:readonly_user:r-x /shared_data
```

The second command sets **default ACLs** for future files.

## Step 9: View ACL Settings

Check the current ACLs on the directory:

```
getfacl /shared_data
```

```
Last login: Fri Sep  5 20:44:04 2025 from 105.115.7.252
ubuntu@ip-172-31-31-80:~$ getfacl /shared_data
getfacl: Removing leading '/' from absolute path names
# file: shared_data
# owner: root
# group: datagroup
# flags: -st
user::rwx
group::rwx
other::r-x

ubuntu@ip-172-31-31-80:~$
```

## Step 10: Test Group Member File Creation

Switch to `user1` and create a file:

```
sudo su - user1
touch /shared_data/user1_file.txt
echo "Content by user1" > /shared_data/user1_file.txt
exit
```

Now switch to `user2` and do the same:

```
sudo su - user2
touch /shared_data/user2_file.txt
echo "Content by user2" > /shared_data/user2_file.txt
exit
```

```
ubuntu@ip-172-31-31-80:~$ sudo su - user1
$ touch /shared_data/user1_file.txt
$ echo "Content by user1" > /shared_data/user1_file.txt
$ exit
ubuntu@ip-172-31-31-80:~$ sudo su - user2
$ touch /shared_data/user2_file.txt
$ echo "Content by user2" > /shared_data/user2_file.txt
$ exit
ubuntu@ip-172-31-31-80:~$
```

## Step 11: Test Write Access Between Group Members

Check if `user2` can modify `user1`'s file:

```
sudo su - user2
echo "Modified by user2" >> /shared_data/user1_file.txt
Exit
```

```
$ exit
ubuntu@ip-172-31-31-80:~$ sudo su - user2
$ echo "Modified by user2" >> /shared_data/user1_file.txt
-sh: 1: cannot create /shared_data/user1_file.txt: Permission denied
$ exit
ubuntu@ip-172-31-31-80:~$
```

---

## Step 12: Test Deletion Prevention

Attempt to delete another user's file (should fail):

```
sudo su - user2
rm /shared_data/user1_file.txt
# This should fail due to the sticky bit
Exit
```

```
$ exit
ubuntu@ip-172-31-31-80:~$ sudo su - user2
$ rm /shared_data/user1_file.txt
rm: cannot remove '/shared_data/user1_file.txt': Operation not permitted
$ exit
ubuntu@ip-172-31-31-80:~$
```

---

## Step 13: Test Read-Only Access for `readonly_user`

Switch to `readonly_user`:

```
sudo su - readonly_user
ls -la /shared_data
cat /shared_data/user1_file.txt
```

Attempt to create a new file (should fail):

```
touch /shared_data/readonly_test.txt
```

```
Exit
```

```
$ exit
ubuntu@ip-172-31-31-80:~$ sudo su - readonly_user
$ ls -la /shared_data
total 16
drwxrwsr-t 2 root datagroup 4096 Sep  5 21:25 .
drwxr-xr-x 23 root root    4096 Sep  5 20:47 ..
-rw-rw-r--  1 user1 datagroup   17 Sep  5 21:24 user1_file.txt
-rw-rw-r--  1 user2 datagroup   17 Sep  5 21:25 user2_file.txt
$ cat /shared_data/user1_file.txt
Content by user1
$ touch /shared_data/readonly_test.txt
touch: cannot touch '/shared_data/readonly_test.txt': Permission denied
$ exit
ubuntu@ip-172-31-31-80:~$
```

## Apache Virtual Hosts Setup

Configure Apache to host **two separate websites** — `site1.local` and `site2.local` — each with its own document root and log files.

### Step 1: Create the Setup Script

Create a new Bash script to automate the setup:

```
nano apache_virtual_hosts_setup.sh
```

Inside the file, you would typically include commands to:

- Create directories for each site
- Create index.html files
- Set permissions
- Create virtual host config files
- Enable the sites
- Reload Apache

After writing or pasting your script, save and exit:  
Press *Ctrl + X*, then *Y*, then *Enter*.

## Step 2: Make the Script Executable

Give the script execution permissions:

```
chmod +x apache_virtual_hosts_setup.sh
```

## Step 3: Run the Script

Execute the script with superuser privileges:

```
sudo ./apache_virtual_hosts_setup.sh
```

## Step 4: Add Entries to the Hosts File

Edit your local `/etc/hosts` file to map the domain names:

```
sudo nano /etc/hosts
```

Add the following lines:

```
127.0.0.1 site1.local
```

```
127.0.0.1 site2.local
```

Save and exit the file:  
Press *Ctrl + X*, then *Y*, then *Enter*.

## Step 5: Test the Setup

Use `curl` to verify that the sites are working:

```
curl http://site1.local
```

You should see the HTML content returned from your site.

Repeat for the second site if needed:

```
curl http://site2.local
```

```
curl http://site1.local  
chmod +x apache_virtual_hosts_setup.sh  
sudo ./apache_virtual_hosts_setup.sh
```

## 4, SSL/TLS Implementation (6 Marks)

In this task, you will generate a self-signed SSL certificate using `openssl` and enable HTTPS for one of your virtual hosts on Apache.

### Step 1: Verify OpenSSL Installation

Check if OpenSSL is installed by typing:

```
openssl version
```

## Step 2: Create Directories for SSL Certificates

Create the necessary directories and set proper permissions:

```
sudo mkdir -p /etc/ssl/private  
sudo mkdir -p /etc/ssl/certs
```

Set permissions:

```
sudo chmod 700 /etc/ssl/private  
sudo chmod 755 /etc/ssl/certs
```

```
$ exec  
ubuntu@ip-172-31-31-80:~$ sudo mkdir -p /etc/ssl/private  
ubuntu@ip-172-31-31-80:~$ sudo mkdir -p /etc/ssl/certs  
ubuntu@ip-172-31-31-80:~$ sudo chmod 700 /etc/ssl/private  
ubuntu@ip-172-31-31-80:~$ sudo chmod 755 /etc/ssl/certs  
ubuntu@ip-172-31-31-80:~$
```

## Step 3: Generate a Private Key

Generate a 2048-bit RSA private key:

```
sudo openssl genrsa -out /etc/ssl/private/example.com.key 2048
```

Secure the private key:

```
sudo chmod 600 /etc/ssl/private/example.com.key
```

## Step 4: Generate a Certificate Signing Request (CSR)

Create a CSR using the private key:

```
sudo openssl req -new -key /etc/ssl/private/example.com.key -out  
/etc/ssl/certs/example.com.csr
```

```
40E7B16E097B0000:error:80000002:system library:file_open:No such file or directory:../providers/implementations/storemgmt/file_store.c:267
ubuntu@ip-172-31-31-80:~$ sudo openssl genrsa -out /etc/ssl/private/example.com.key 2048
ubuntu@ip-172-31-31-80:~$ sudo chmod 600 /etc/ssl/private/example.com.key

ubuntu@ip-172-31-31-80:~$ sudo openssl req -new -key /etc/ssl/private/example.com.key -out /etc/ssl/certs/example.com.csr
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:      NG
String too long, must be at most 2 bytes long
Country Name (2 letter code) [AU]:NG
State or Province Name (full name) [Some-State]:ABUJA
Locality Name (eg, city) []:ABUJA
Organization Name (eg, company) [Internet Widgits Pty Ltd]:LINCOLN
Organizational Unit Name (eg, section) []:12345
Common Name (e.g. server FQDN or YOUR name) []:DANIE
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:NONE
ubuntu@ip-172-31-31-80:~$ █
```

## Step 5: Generate a Self-Signed SSL Certificate

Use the CSR and private key to generate a self-signed certificate valid for 365 days:

```
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
```

Set permissions on the certificate:

```
sudo chmod 644 /etc/ssl/certs/example.com.crt
```

```
ubuntu@ip-172-31-31-80:~$ 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
Certificate request self-signature ok
subject=C = NG, ST = ABUJA, L = ABUJA, O = LINCOLN, OU = 12345, CN = DANIE
ubuntu@ip-172-31-31-80:~$ sudo chmod 644 /etc/ssl/certs/example.com.crt
ubuntu@ip-172-31-31-80:~$ █
```

## Step 6: Enable Apache SSL Module

Enable the necessary Apache modules:

```
sudo a2enmod ssl  
sudo a2enmod rewrite
```

Restart Apache:

```
sudo systemctl restart apache2
```

```
No VM guests are running outdated hypervisor (qemu) binaries on this host.  
ubuntu@ip-172-31-31-80:~$ sudo a2enmod ssl  
Considering dependency mime for ssl:  
Module mime already enabled  
Considering dependency socache_shmcb for ssl:  
Enabling module socache_shmcb.  
Enabling module ssl.  
Enabling module rewrite.  
See /usr/share/doc/apache2/README.Debian.gz on how to configure SSL and create self-signed certificates.  
To activate the new configuration, you need to run:  
    systemctl restart apache2  
ubuntu@ip-172-31-31-80:~$ sudo a2enmod rewrite  
Enabling module rewrite.  
To activate the new configuration, you need to run:  
    systemctl restart apache2  
ubuntu@ip-172-31-31-80:~$ sudo systemctl restart apache2  
ubuntu@ip-172-31-31-80:~$ █
```

## Step 7: Create an SSL Virtual Host Configuration

Open or create the SSL virtual host config file:

```
sudo nano /etc/apache2/sites-available/example.com-ssl.conf
```

Paste the following configuration:

```
<VirtualHost *:443>
```

```
    ServerName example.com
```

```
    ServerAlias www.example.com
```

```
    DocumentRoot /var/www/example.com
```

```
# SSL Configuration
```

```
    SSLEngine on
```

```
    SSLCertificateFile /etc/ssl/certs/example.com.crt
```

```
SSLCertificateKeyFile /etc/ssl/private/example.com.key
```

#### # Security Headers

```
Header always set Strict-Transport-Security "max-age=63072000; includeSubDomains;  
preload"
```

```
Header always set X-Frame-Options DENY
```

```
Header always set X-Content-Type-Options nosniff
```

#### # Log files

```
ErrorLog ${APACHE_LOG_DIR}/example.com-ssl-error.log
```

```
CustomLog ${APACHE_LOG_DIR}/example.com-ssl-access.log combined
```

#### # Directory Configuration

```
<Directory /var/www/example.com>
```

```
    Options Indexes FollowSymLinks
```

```
    AllowOverride All
```

```
    Require all granted
```

```
</Directory>
```

```
</VirtualHost>
```

Save and exit the file:

Press *Ctrl + X*, then *Y*, then *Enter*.

## Step 8: Enable the SSL Virtual Host

Enable the site:

```
sudo a2ensite example.com-ssl.conf
```

## Test Apache configuration:

```
sudo apache2ctl configtest
```

## Step 9: Restart Apache and Test

Restart Apache to apply the changes:

```
sudo systemctl restart apache2
```

Check the Apache status:

```
sudo systemctl status apache2
```

```
ubuntu@ip-172-31-31-80:~$ sudo systemctl restart apache2
ubuntu@ip-172-31-31-80:~$ sudo systemctl status apache2
● apache2.service - The Apache HTTP Server
    Loaded: loaded (/usr/lib/systemd/system/apache2.service; enabled; preset: enabled)
    Active: active (running) since Fri 2025-09-05 21:56:06 UTC; 1s ago
      Docs: https://httpd.apache.org/docs/2.4/
   Process: 4005 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUCCESS)
 Main PID: 4010 (apache2)
    Tasks: 55 (limit: 1008)
   Memory: 6.3M (peak: 6.5M)
      CPU: 38ms
     CGroup: /system.slice/apache2.service
             ├─4010 /usr/sbin/apache2 -k start
             ├─4012 /usr/sbin/apache2 -k start
             └─4013 /usr/sbin/apache2 -k start

Sep 05 21:56:06 ip-172-31-31-80 systemd[1]: Starting apache2.service - The Apache HTTP Server...
Sep 05 21:56:06 ip-172-31-31-80 systemd[1]: Started apache2.service - The Apache HTTP Server.
ubuntu@ip-172-31-31-80:~$
```

### **Step 10: Verify the SSL Certificate**

## Test SSL connectivity:

```
openssl s_client -connect example.com:443 -servername example.com
```

```

SRP username: None
TLS session ticket lifetime hint: 83100 (seconds)
TLS session ticket:
0000 - 02 9d ff e6 4b c4 0b-fa 2c 9e a2 2e 47 b8 d9 .....K.....G.
0010 - d3 9d 3b 53 84 d6 f0 82-46 36 95 93 14 2e 1c ea .N!S...F6....p...
0020 - ff 92 d4 af 13 5c 61 04-a9 fc 18 cc 79 eb f1 0c .....a....p...
0030 - 2a 02 0f ca 18 bf e3 28-48 9d a8 04 47 49 a9 49 *.....G!I.
0040 - 32 15 c1 b3 d1 5a f0 33-78 04 88 05 fb f7 02 4a 2....Z.3x.....J
0050 - 1c 71 c2 c5 77 4a 1a 54-30 da 3d d0 e7 ce 3a 0c .q....K.T0.=...>.
0060 - 10 f5 a7 c8 14 01 33 2e-7b 26 87 99 d1 01 2d dd .....3.{&....-
0070 - 3d b0 01 08 56 cc b6 23-f7 29 6d a5 25 df e4 9b =.Hv.#.m%.<.
0080 - 5c 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 ..1.(....8?.....
0090 - 59 73 89 b8 ff c5 e3 e5-a9 f7 9e ba d8 c1 bb 79 {0....P....P.
00a0 - c8 14 b1 be 08 38 08 6a-b0 19 ba ab 08 8d 72 bd .....8.j....R.
00b0 - 67 4f 59 0c 5d 76 77 5d-8d 38 7f b0 ab 82 09 98 g0V.iw1.B.....
00c0 - 4f e2 b1 a1 87 fa 61 bc-f8 fa bc 49 9a 73 c7 64 O....a...l.s.d
00d0 - bb 44 96 b6 2c 96 15 62-bd 17 2d bc fa 76 bc cd D....b.-.v.c.
00e0 - fd ca d0 c1 7d 04 c6 bb-7f 4e b5 6a 6c 43 a5 ae .....N.jlc.

Start Time: 1757109635
Timeout : 7200 (sec)
Verify return code: 0 (ok)
Extended master secret: no
Max Early Data: 0
-----
read R BLOCK
-----
Post-Handshake New Session Ticket arrived:
SSL-Session:
Protocol : TLSv1.3
Cipher : TLS_AES_256_GCM_SHA384
Session-ID: 0711B3050F342A46D471BBE2A8EC5E575E33F1ABB847A255DAB872EF615925C89
Session-ID-ctx:
Resumption PSK: AC488A8FDD0D5C4BA6A5B59D38A6ECFB0384A126A6F997CE435D434B35EB11E3B6E48AD5A827B389A93DE175BB07CBF9
PSK Identity: None
PSK Identity Hint: None
SRP username: None
TLS session ticket lifetime hint: 83100 (seconds)
TLS session ticket:
0000 - 00 02 0d ff e6 4b c4 0b-fa 2c 9e a2 2e 47 b8 d9 .....K.....G.
0010 - 3e c8 d7 7b e8 07 3d f3-79 5a 18 a2 b1 0a 40 76 >..{.=yZ....@v
0020 - ef ed db b8 74 31 1e 95-01 86 be b5 38 c8 d9 ba ..t;.....8..
0030 - 12 41 2b 9f dd 40 99 05-91 c9 cc 06 06 06 19 fa ba {A}.....
0040 - 4c 5d 7d 5c 66 06 06 06 06 06 06 06 06 06 06 25 L....Gf...h...X...
0050 - 15 5d 70 de cc 08 56 0e-b3 cc 5f 44 92 58 b6 18 [p....^....]....X.
0060 - c8 cc 83 37 ae 9f cb 5d-45 c8 67 7d 1f e5 b2 d3 ..7....]E.g].....
0070 - c4 dd 90 ee 37 0b ae a6-14 75 94 4b 7c e8 78 88 ..7....uK].....
0080 - 89 bb cb be 1e 2b 69 62-47 13 9e bf d8 f1 d8 bb ....+iB.....
0090 - 6d 4b a5 b9 83 45 e2 12-5a 89 56 03 23 33 3f 5d mK...E..Z.V.#3?
00a0 - 70 9b ff 5a 67 cc 23 fb-d4 f3 9f c4 59 89 b5 d8 p..Zg.#....Y...
00b0 - 93 09 78 96 f0 10 93 16-72 be 47 1a 84 8d ee 3e ..p....G....>
00c0 - 7c f5 ed 48 ae 1e f3 7d-47 58 56 00 71 98 57 63 [..E....]K^..q.W.
00d0 - 68 70 86 98 d3 da 49 5e-66 56 02 6c 41 9f 29 h.....Zfn..\\L.
00e0 - 9e 44 fc c9 d9 73 0d b2-7e da b8 df 89 db ae 3d .J....s.-....=.

Start Time: 1757109635
Timeout : 7200 (sec)
Verify return code: 0 (ok)
Extended master secret: no
Max Early Data: 0
-----
read R BLOCK

```

## View certificate details:

```
openssl x509 -in /etc/ssl/certs/example.com.crt -text -noout
```

```

00e0 - 9e 4a fc c9 d9 73 8d b2-7e da b0 df 09 db a6 3d .J...s..~.....=>

Start Time: 1757109635
Timeout : 3000 (secs)
Verify return codes: 0 (ok)
Extended master secret: no
Max Early Data: 0
---
read R BLOCK
closed
ubuntu@ip-172-31-31-80:~$ openssl x509 -in /etc/ssl/certs/example.com.crt -text -noout
Certificate:
Data:
Version: 1 (0x0)
Serial Number:
  1b:07:93:6b:28:31:3d:4a:9e:69:a9:c2:ff:51:93:7e:4e:59:a9:50
Signature Algorithm: sha256WithRSAEncryption
Issuer: C = NG, ST = ABUJA, L = ABUJA, O = LINCOLN, OU = 12345, CN = DANIE
Validity
  Not Before: Sep 5 21:48:49 2025 GMT
  Not After : Sep 5 21:48:49 2026 GMT
Subject: C = NG, ST = ABUJA, L = ABUJA, O = LINCOLN, OU = 12345, CN = DANIE
Subject Public Key Info:
  Public Key Algorithm: rsaEncryption
  Public-Key: (2048 bit)
    Modulus:
      99:da:82:8f:8c:28:94:9f:8e:b7:8a:c7:42:e8:a3:
      e4:dd:4f:ec:c1:d7:a4:c4:8a:be:f8:a2:1a:b7:
      9b:55:eb:0a:8c:ee:28:ca:38:9b:52:43:83:4b:b7:
      be:e7:f4:a4:77:13:58:6a:0e:dd:45:c2:c7:a0:8f:
      b3:20:66:ec:5c:bd:22:87:88:83:e0:1e:3c:19:82:
      91:8e:6b:00:35:8c:00:0b:14:73:ce:08:54:45:
      1f:40:07:0f:01:35:8c:00:0b:14:73:ce:08:54:45:
      11:ba:6f:0b:b3:3c:0a:c7:d4:ac:1b:e4:75:5:42:
      8e:83:6b:0b:bd:40:d1:2d:6a:5c:bd:76:ad:94:d1:2a:
      0c:62:2a:02:64:36:bd:a6:76:8c:f4:d6:6:8d:
      cb:3b:9b:75:00:2f:fc:01:a8:4a:8:c5:3b:8c:5a:
      a7:52:9e:77:d6:c9:d1:4d:2c:18:23:fe:dd:5b:da:
      8f:6f:92:93:8e:d7:d2:de:c4:4b:a4:23:cb:9a:25:
      72:3b:9b:75:00:2f:fc:01:a8:4a:8:c5:3b:8c:5a:
      eb:87:9e:74:08:rd:94:c1:b9:98:6a:d9:61:6:3e:
      3:ccb:a3:ff:33:ab:2c:c8:6d:05:45:1b:a7:1b:34:
      2b:76:01:20:b1:8e:30:4f:c5:07:fb:e3:da:8c:ba:
      0f:0f
    Exponent: 65537 (0x10001)
Signature Algorithm: sha256WithRSAEncryption
Signature Value:
  83:1a:3d:80:a9:03:f5:bc:5c:23:34:f7:cf:3a:bc:53:dc:37:
  cd:12:80:3a:aa:fd:51:25:k4:48:56:11:ch:bf:02:0c:93:99:
  da:3f:4f:cc:4:69:ff:90:6f:32:7b:a0:39:64:dd:43:56:11:k1:
  19:18:b6:db:dd:69:8f:90:1a:5:a0:5d:a0:0:3a:2b:bb:5e:ad:cd:c1:
  c3:57:9b:90:01:72:1f:ca:3b:3c:c7:d7:d8:89:33:c5:51:9f:
  7f:78:54:ca:4:d7:bb:37:0:32:37:0:0:ef:fe:ad:64:1e:a:
  cb:96:ec:98:3e:5b:bd:22:35:8c:05:05:7d:19:1c:ee:03:30:
  9b:78:6d:51:c1:14:9d:9e:01:4f:bb:72:05:34:44:f6:22:0:
  7b:5b:11:9e:44:e3:dd:07:67:ab:36:2c:17:a4:a5:13:74:47:
  c6:1b:81:c0:3:bd:86:dd:5:6:7:e2:fe:i1:7:57:8f:dd:cc:50:1c:22:
  d9:76:a8:2f:f9:a8:e7:28:b0:e0:91:bb:8a:f4:3a:34:26:8b:
  7f:74:8c:ac:72:5f:d7:f4:a5:4:f9:b9:4e:8b:5:4b:1d:86:
  eb:7:2d:2f:f7:57:f7:78:cb:8b:b6:f0:ac:8d:4e:42:36:e1:c5:
  97:d7:dc:22
ubuntu@ip-172-31-31-80:~$ 

```

## Step 11: Configure Firewall (If UFW is Enabled)

Allow HTTPS traffic through the firewall:

```

sudo ufw allow 'Apache Full'
sudo ufw allow 443/tcp

```

Check UFW status:

```

sudo ufw status

```

## 5. MySQL Remote Access & Security (7 Marks)

In this task, you'll configure MySQL to allow secure **remote connections**, create a **least-privileged user**, and **secure your MySQL server** against unauthorized access.

### Step 1: Check MySQL Installation and Status

Check if MySQL is running:

```
sudo systemctl status mysql
```

Check the installed MySQL version:

```
mysql --version
```

### Step 2: Configure MySQL for Remote Access

#### 2.1 Edit MySQL Configuration File

First, create a backup of the default configuration:

```
sudo cp /etc/mysql/mysql.conf.d/mysqld.cnf  
/etc/mysql/mysql.conf.d/mysqld.cnf.backup
```

Then open the config file for editing:

```
sudo nano /etc/mysql/mysql.conf.d/mysqld.cnf
```

#### 2.2 Modify the **bind-address** Setting

Inside the file, find the line that begins with:

bind-address = 127.0.0.1

Change it to:

bind-address = 0.0.0.0

This change allows MySQL to accept connections from **remote IP addresses**.

Save and exit:

- Press *Ctrl + X*, then *Y*, then *Enter*

```
Setting the MySQL server deployment.
Connecting to MySQL using a blank password.

VALIDATE PASSWORD COMPONENT can be used to test passwords
and improve security. It checks the strength of password
and allows the users to set only those passwords which are
secure enough. Would you like to setup VALIDATE PASSWORD component?

[Press y|Y for Yes, any other key for No: y

There are three levels of password validation policy:

LOW    Length >= 8
MEDIUM Length >= 8, numeric, mixed case, and special characters
STRONG Length >= 8, numeric, mixed case, special characters and dictionary      file

[Please enter 0 = LOW, 1 = MEDIUM and 2 = STRONG: 1

Skipping password set for root as authentication with auth_socket is used by default.
If you would like to use password authentication instead, this can be done with the "ALTER_USER" command.
See https://dev.mysql.com/doc/refman/8.0/en/alter-user.html#alter-user-password-management for more information.

By default, a MySQL installation has an anonymous user,
allowing anyone to log into MySQL without having to have
a user account created for them. This is intended only for
testing, and to make the installation go a bit smoother.
You should remove them before moving into a production
environment.

[Remove anonymous users? (Press y|Y for Yes, any other key for No) : y
Success.

Normally, root should only be allowed to connect from
'localhost'. This ensures that someone cannot guess at
the root password from the network.

[Disallow root login remotely? (Press y|Y for Yes, any other key for No) : y
Success.

By default, MySQL comes with a database named 'test' that
anyone can access. This is also intended only for testing,
and should be removed before moving into a production
environment.

[Remove test database and access to it? (Press y|Y for Yes, any other key for No) : y
  - Dropping test database...
Success.

  - Removing privileges on test database...
Success.

Reloading the privilege tables will ensure that all changes
made so far will take effect immediately.

[Reload privilege tables now? (Press y|Y for Yes, any other key for No) : y
Success.

All done!
ubuntu@ip-172-31-31-80:~$ ]
```

## Step 3: Restart MySQL Service

Apply the configuration changes by restarting MySQL:

```
sudo systemctl restart mysql
```

Check the error log for issues:

```
sudo tail -f /var/log/mysql/error.log
```

```

[ubuntu@ip-172-31-31-80: ~] $ mysql --version
mysql Ver 8.0.43-0ubuntu0.24.04.1 for Linux on x86_64 ((Ubuntu))
[ubuntu@ip-172-31-31-80: ~] $ sudo nano /etc/mysql/mysql.conf.d/mysql.cnf
[ubuntu@ip-172-31-31-80: ~] $ sudo systemctl restart mysql
[ubuntu@ip-172-31-31-80: ~] $ sudo tail -f /var/log/mysql/error.log
2025-09-06T13:25:09.707340Z 0 [System] [MY-013172] [Server] Received SHUTDOWN from user <via user signal>. Shutting down mysqld (Version: 8.0.43-0ubuntu0.24.04.1) (Ubuntu).
2025-09-06T13:25:09.773117Z 0 [System] [MY-013172] [Server] Received SHUTDOWN from user <via user signal>. Shutting down mysqld (Version: 8.0.43-0ubuntu0.24.04.1) (Ubuntu).
2025-09-06T13:26:01.056162Z 0 [System] [MY-018910] [Server] /usr/sbin/mysqld: Shutdown complete (mysqld 8.0.43-0ubuntu0.24.04.1) (Ubuntu).
2025-09-06T13:26:01.718939Z 0 [System] [MY-018116] [Server] /usr/sbin/mysqld (mysqld 8.0.43-0ubuntu0.24.04.1) starting as process 12847
2025-09-06T13:26:01.742989Z 1 [System] [MY-013576] [InnoDB] InnoDB initialization has started.
2025-09-06T13:26:02.446163Z 1 [System] [MY-013577] [InnoDB] InnoDB initialization has ended.
2025-09-06T13:26:03.041864Z 0 [Warning] [MY-010681] [Server] X certificate ca.pem is self signed.
2025-09-06T13:26:03.041864Z 0 [Warning] [MY-010680] [Server] X temporary keyring ca-keyring is self signed to support TLS. Encrypted connections are now supported for this channel.
2025-09-06T13:26:03.146312Z 0 [System] [MY-011323] [Server] X Plugin ready for connections. Bind-address: '127.0.0.1' port: 3306, socket: /var/run/mysqld/mysqld.sock
2025-09-06T13:26:03.146473Z 0 [System] [MY-018931] [Server] /usr/sbin/mysqld: ready for connections. Version: '8.0.43-0ubuntu0.24.04.1' socket: '/var/run/mysqld/mysqld.sock' port: 3306 (Ubuntu).
^C
[ubuntu@ip-172-31-31-80: ~] $ sudo mysql_secure_installation

Securing the MySQL server deployment.

Connecting to MySQL using a blank password.
The 'validate_password' component is installed on the server.
The subsequent steps will run with the existing configuration
of the component.

Skipping password set for root as authentication with auth_socket is used by default.
If you would like to use password authentication instead, this can be done with the "ALTER_USER" command.
See https://dev.mysql.com/doc/refman/8.0/en/alter-user.html#alter-user-password-management for more information.

By default, a MySQL installation has an anonymous user,
allowing anyone to log into MySQL without having to have
a user account created for them. This is intended only for
testing, and to make the installation go a bit smoother.
You should remove them before moving into a production
environment.

Remove anonymous users? (Press y|Y for Yes, any other key for No) : ^[[


```

## Step 4: Secure MySQL Installation

Run the built-in security script:

```
sudo mysql_secure_installation
```

Follow the prompts to:

- Set root password
- Remove anonymous users
- Disallow remote root login
- Remove test database
- Reload privilege tables

## Step 5: Create a Database and Remote User

### 5.1 Login to MySQL as Root

```
sudo mysql -u root -p
```

### 5.2 Create a New Database

Run the following inside the MySQL shell:

```
CREATE DATABASE company_db;  
SHOW DATABASES;
```

### 5.3 Create a Remote User with Least Privileges

Create a user that can connect from a specific IP address (replace with your actual IP):

```
CREATE USER 'remote_user'@'192.168.1.100' IDENTIFIED BY 'SecurePassword123!';
```

Or allow connections from any IP in a specific range:

```
CREATE USER 'remote_user'@'192.168.1.%' IDENTIFIED BY 'SecurePassword123!';
```

Grant limited privileges to the user on the database:

```
GRANT SELECT, INSERT, UPDATE, DELETE ON company_db.* TO  
'remote_user'@'192.168.1.%';
```

Or, for **read-only access**:

```
GRANT SELECT ON company_db.* TO 'remote_user'@'192.168.1.%';
```

Apply privilege changes:

```
FLUSH PRIVILEGES;
```

### 5.4 Verify User and Privileges

Check the user's host and creation:

```
SELECT User, Host FROM mysql.user WHERE User = 'remote_user';
```

Check granted permissions:

```
SHOW GRANTS FOR 'remote_user'@'192.168.1.%';
```

Exit the MySQL shell:

```
EXIT;
```

```
EXIT;
```

```
Server version: 8.0.43-0ubuntu0.24.04.1 (Ubuntu)
Copyright (c) 2000, 2025, Oracle and/or its affiliates.
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

[mysql]> \c
[mysql]> '\c'
| -> ^C
[mysql]> ^C
[mysql]> ^C
mysql> CREATE DATABASE company_db;
Query OK, 1 row affected (0.00 sec)

mysql> SHOW DATABASES;
+-----+
| Database |
+-----+
| company_db |
| information_schema |
| mysql |
| performance_schema |
| sys |
+-----+
5 rows in set (0.00 sec)

mysql> CREATE USER 'remote_user'@'192.168.1.100' IDENTIFIED BY 'SecurePassword123!';
Query OK, 0 rows affected (0.02 sec)

mysql> CREATE USER 'remote_user'@'192.168.1.%' IDENTIFIED BY 'SecurePassword123!';
Query OK, 0 rows affected (0.01 sec)

mysql> GRANT SELECT, INSERT, UPDATE, DELETE ON company_db.* TO 'remote_user'@'192.168.1.%';
Query OK, 0 rows affected (0.01 sec)

mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.01 sec)

mysql> SELECT User, Host FROM mysql.user WHERE User = 'remote_user';
+-----+-----+
| User | Host |
+-----+-----+
| remote_user | 192.168.1.% |
| remote_user | 192.168.1.100 |
+-----+-----+
2 rows in set (0.00 sec)

mysql> SHOW GRANTS FOR 'remote_user'@'192.168.1.%';
+-----+
| Grants for remote_user@192.168.1.% |
+-----+
| GRANT USAGE ON *.* TO `remote_user`@`192.168.1.%` |
| GRANT SELECT, INSERT, UPDATE, DELETE ON `company_db`.* TO `remote_user`@`192.168.1.%` |
+-----+
2 rows in set (0.00 sec)

mysql> EXIT;
Bye
ubuntu@ip-172-31-31-80:~$
```

## Step 6: Configure Firewall

## 6.1 Using UFW (Ubuntu Firewall)

```
# Allow MySQL port from specific IP  
  
sudo ufw allow from 192.168.1.100 to any port 3306  
  
# Or allow from specific subnet  
  
sudo ufw allow from 192.168.1.0/24 to any port 3306
```

## 6. Firewall Configuration (6 Marks)

In this task, you will configure the **UFW firewall** (or [iptables](#)) to allow access to specific services only from **defined IP ranges**. Services to be allowed:

- HTTP (port 80)
- HTTPS (port 443)
- SSH (port 22)
- MySQL (port 3306)

### Step 1: Check UFW Status and Install If Needed

Check if `ufw` is installed:

```
which ufw
```

If not installed, update your package index and install:

```
sudo apt update  
sudo apt install ufw
```

Check current UFW status:

```
sudo ufw status verbose
```

## Step 2: Reset UFW to Default State

Reset UFW to remove existing rules and start from a clean slate:

```
sudo ufw --force reset
```

Set the default firewall policies:

```
sudo ufw default deny incoming  
sudo ufw default allow outgoing
```

Check current status to confirm:

```
sudo ufw status verbose
```

## Step 3: Define IP Ranges

Define environment variables for your IP ranges (for easier use):

```
ADMIN_NETWORK="192.168.1.0/24"      # SSH Access  
  
WEB_NETWORK="10.0.0.0/16"           # HTTP/HTTPS Access  
  
DB_NETWORK="172.16.0.0/24"          # MySQL Access  
  
OFFICE_NETWORK="203.0.113.0/24"     # Optional trusted network
```

## Step 4: Add UFW Rules for Specific IP Ranges

Allow SSH from the admin network:

```
sudo ufw allow from 192.168.1.0/24 to any port 22 proto tcp
```

Allow HTTP and HTTPS from the web network:

```
sudo ufw allow from 10.0.0.0/16 to any port 80 proto tcp  
sudo ufw allow from 10.0.0.0/16 to any port 443 proto tcp
```

Allow MySQL access from the database network:

```
sudo ufw allow from 172.16.0.0/24 to any port 3306 proto tcp
```

## Step 5: Enable the Firewall

After configuring the rules, enable UFW:

```
sudo ufw enable
```

Then check the final rule set:

```
sudo ufw status numbered
```

## 7. System Monitoring Script (6 Marks)

Create a script that monitors **CPU**, **Memory**, and **Disk usage** every 5 minutes and logs the output to `/var/log/sys_health.log`. The script should also support multiple commands for setup, status checks, and viewing logs.

## Step 1: Create the Monitoring Script

Open a new file using:

```
nano system_monitor_complete.sh
```

Inside the file, you would define functionality for logging system metrics and handling arguments like `setup`, `status`, `logs`, etc.

(If you'd like the actual script content, let me know!)

After pasting or writing the script, save and exit:

Press `Ctrl + X`, then `Y`, then `Enter`.

```
if [ "$EUID" -ne 0 ]; then
    print_error "Please run as root (use sudo)"
    exit 1
fi
test_setup
;;
"help"|"--h"|"--help")
    show_usage
;;
"")
    # Interactive mode
    print_header "SYSTEM MONITORING INTERACTIVE SETUP"
    echo "This will set up automated system monitoring."
    echo -n "Do you want to proceed? (y/N): "
    read -r response
    if [[ "$response" =~ ^[Yy]$ ]]; then
        complete_setup
    else
        echo "Setup cancelled."
        exit 0
    fi
;;
*)
    print_error "Unknown option: $1"
    show_usage
    exit 1
;;
esac
exit 0
```

## Step 2: Make the Script Executable

Run the following to make your script executable:

```
chmod +x system_monitor_complete.sh
```

## Step 3: Run the Script

Run the script with the setup argument to initialize logging and create a cron job:

```
sudo ./system_monitor_complete.sh setup
```

```
[ubuntu@ip-172-31-31-88:~]$ nano system_monitor_complete.sh
[ubuntu@ip-172-31-31-88:~]$ nano system_monitor_complete.sh
[ubuntu@ip-172-31-31-88:~]$ nano ./system_monitor_complete.sh
chattr: cannot access './system_monitor_complete.sh': not a file or directory
[ubuntu@ip-172-31-31-88:~]$ nano system_monitor_complete.sh
[ubuntu@ip-172-31-31-88:~]$ chmod +x system_monitor_complete.sh
[ubuntu@ip-172-31-31-88:~]$ sudo ./system_monitor_complete.sh setup
=====
SYSTEM MONITORING SETUP
=====
This script will set up system monitoring with the following:
• CPU, Memory, and Disk usage monitoring
• Logging every 5 minutes to /var/log/sys_health.log
• Automatic log rotation
[INFO] Creating monitoring script at /usr/local/bin/system_monitor.sh
[INFO] Monitoring script created and configured
[INFO] Setting up log file at /var/log/sys_health.log
[INFO] Log file created and initialized
[INFO] Setting up cron job to run every 5 minutes
[INFO] Cron job added successfully
[INFO] Setting up log rotation
[INFO] Log rotation configured
[INFO] Testing the monitoring system...
[INFO] ✓ Monitoring test successful!

Recent log entries:
=====
[2025-09-06 13:57:09] System monitoring initialized
[2025-09-06 13:57:09] =====
[2025-09-06 13:57:09] SYSTEM HEALTH REPORT
[2025-09-06 13:57:09] CPU Usage: 4.3%
[2025-09-06 13:57:09] Memory Usage: 78.4% (0.7GB/0.9GB)
[2025-09-06 13:57:09] Disk Usage: 43% (2.9G/6.8G)
[2025-09-06 13:57:09] Hostname: ip-172-31-31-88
[2025-09-06 13:57:09] Load Average: 0.00, 0.00, 0.00
[2025-09-06 13:57:09] =====

=====
SETUP COMPLETE
=====
[INFO] System monitoring is now active!
[INFO] Logs will be written to: /var/log/sys_health.log
[INFO] Monitoring runs every 5 minutes

[INFO] To view logs in real-time: sudo tail -f /var/log/sys_health.log
[INFO] To check status: ./system_monitor_complete.sh status
[INFO] To view recent logs: ./system_monitor_complete.sh logs
ubuntu@ip-172-31-31-88:~]$
```

## Available Commands

You can use the following commands to manage the system monitoring script:

Command	Description
<code>sudo ./system_monitor_complete.sh setup</code>	Set up logging and create a cron job
<code>sudo ./system_monitor_complete.sh status</code>	Display current CPU, memory, and disk usage
<code>sudo ./system_monitor_complete.sh logs</code>	View recent entries in <code>/var/log/sys_health.log</code>
<code>sudo ./system_monitor_complete.sh test</code>	Run one-time manual check
<code>sudo ./system_monitor_complete.sh</code>	Launch interactive mode (menu-based)
<code>./system_monitor_complete.sh help</code>	Show help and usage instructions

## 8. Log Rotation Setup (6 Marks)

Configure `logrotate` for a **custom application log** so that logs are:

- Rotated **daily**
- **Compressed** after rotation
- **Retained for 7 days**
- Automatically managed without manual cleanup

### Step 1: Create or Identify the Application Log File

For this example, we'll use a custom log file located at:

`/var/log/myapp/application.log`

**Create the log directory:**

```
sudo mkdir -p /var/log/myapp
```

**Create the log file:**

```
sudo touch /var/log/myapp/application.log
```

**Set ownership and permissions:**

```
sudo chown root:root /var/log/myapp/application.log  
sudo chmod 644 /var/log/myapp/application.log
```

**Add sample content:**

```
echo "$(date) - Application started" | sudo tee -a  
/var/log/myapp/application.log  
echo "$(date) - Sample log entry" | sudo tee -a  
/var/log/myapp/application.log
```

### Verify file and content:

```
ls -la /var/log/myapp/  
cat /var/log/myapp/application.log
```

```
Last login: Sat Sep  6 15:13:56 on ttys000  
Restored session: Sat Sep  6 15:16:00 WAT 2025  
[mac@Macs-MacBook-Pro ~]$ ssh -i danny-key.pem ubuntu@54.226.237.140  
Welcome to Ubuntu 24.04.3 LTS (GNU/Linux 6.14.0-1011-aws x86_64)  
  
 * Documentation:  https://help.ubuntu.com  
 * Management:    https://landscape.canonical.com  
 * Support:       https://ubuntu.com/pro  
  
System information as of Sat Sep  6 14:16:50 UTC 2025  
  
 System load:  0.01      Temperature:     -273.1 °C  
 Usage of /:   42.1% of 6.71GB  Processes:        122  
 Memory usage: 67%          Users logged in:  1  
 Swap usage:   0%           IPv4 address for ens5: 172.31.31.80  
  
Expanded Security Maintenance for Applications is not enabled.  
2 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  
  
*** System restart required ***  
Last login: Sat Sep  6 14:09:54 2025 from 105.112.123.182  
[ubuntu@ip-172-31-31-80:~]$ sudo touch /var/log/myapp/application.log  
[ubuntu@ip-172-31-31-80:~]$ sudo chown root:root /var/log/myapp/application.log  
[ubuntu@ip-172-31-31-80:~]$ sudo chmod 644 /var/log/myapp/application.log  
[ubuntu@ip-172-31-31-80:~]$ echo "$(date) - Application started" | sudo tee -a /var/log/myapp/application.log  
Sat Sep  6 14:17:35 UTC 2025 - Application started  
[ubuntu@ip-172-31-31-80:~]$ echo "$(date) - Sample log entry" | sudo tee -a /var/log/myapp/application.log  
Sat Sep  6 14:18:00 UTC 2025 - Sample log entry  
[ubuntu@ip-172-31-31-80:~]$ ls -la /var/log/myapp/  
total 12  
drwxr-xr-x  2 root root  4096 Sep  6 14:16 .  
drwxrwxr-x 14 root root  4096 Sep  6 14:09 ..  
-rw-r--r--  1 root root   99 Sep  6 14:18 application.log  
[ubuntu@ip-172-31-31-80:~]$ cat /var/log/myapp/application.log  
Sat Sep  6 14:17:35 UTC 2025 - Application started  
Sat Sep  6 14:18:00 UTC 2025 - Sample log entry  
[ubuntu@ip-172-31-31-80:~]$
```

## Step 2: Create a Logrotate Configuration File

Create a new configuration file under `/etc/logrotate.d/`:

```
sudo nano /etc/logrotate.d/myapp
```

Paste the following configuration:

```
/var/log/myapp/application.log {  
    daily  
    rotate 7  
    compress  
    delaycompress  
    missingok  
    notifempty  
    create 644 root root  
    postrotate  
        # Optional: Signal application to reopen the log file  
        # /usr/bin/systemctl reload myapp || true  
        # /usr/bin/killall -SIGUSR1 myapp || true  
    endscript  
}
```

Save and exit:

Press `Ctrl + X`, then `Y`, then `Enter`

```
GNU nano 7.2
bash/var/log/myapp/application.log {
    daily
    rotate 7
    compress
    delaycompress
    missingok
    notifempty
    create 644 root root
    postrotate
        # Optional: Signal application to reopen log file
        # /usr/bin/systemctl reload myapp || true
        # /usr/bin/killall -SIGUSR1 myapp || true
    endscript
}


```

### Step 3: Set Proper Permissions on the Config File

Ensure proper file permissions:

```
sudo chmod 644 /etc/logrotate.d/myapp
sudo chown root:root /etc/logrotate.d/myapp
```

Verify the file:

```
ls -la /etc/logrotate.d/myapp
cat /etc/logrotate.d/myapp
```

```
[ubuntu@ip-172-31-31-80:~$ sudo chmod 644 /etc/logrotate.d/myapp
[ubuntu@ip-172-31-31-80:~$ sudo chown root:root /etc/logrotate.d/myapp
[ubuntu@ip-172-31-31-80:~$ ls -la /etc/logrotate.d/myapp
-rw-r--r-- 1 root root 348 Sep 6 14:20 /etc/logrotate.d/myapp
[ubuntu@ip-172-31-31-80:~$ cat /etc/logrotate.d/myapp
bash/var/log/myapp/application.log {

    daily
    rotate 7
    compress
    delaycompress
    missingok
    notifempty
    create 644 root root
    postrotate
        # Optional: Signal application to reopen log file
        # /usr/bin/systemctl reload myapp || true
        # /usr/bin/killall -SIGUSR1 myapp || true
    endscript
}

ubuntu@ip-172-31-31-80:~$ ]
```

## Step 4: Test the Logrotate Configuration

**Test syntax (dry run):**

```
sudo logrotate -d /etc/logrotate.d/myapp
```

This checks if your config is correct without making changes.

```
ubuntu@ip-172-31-31-80:~$ sudo logrotate -d /etc/logrotate.d/myapp
warning: logrotate in debug mode does nothing except printing debug messages! Consider using verbose mode (-v) instead if this is not what you want.

reading config file /etc/logrotate.d/myapp
error: /etc/logrotate.d/myapp:1 keyword 'bash' not properly separated, found 0x2f
Reading state from file: /var/lib/logrotate/status
Allocating hash table for state file, size 64 entries
Creating new state
Handling 0 logs
ubuntu@ip-172-31-80:~$ ]
```

**Force log rotation:**

```
sudo logrotate -f /etc/logrotate.d/myapp
```

**Verify rotation occurred:**

```
ls -la /var/log/myapp/
```

```
[ubuntu@ip-172-31-31-80:~]$ ls -la /var/log/myapp/
total 12
drwxr-xr-x 2 root root 4096 Sep 6 14:16 .
drwxrwxr-x 14 root syslog 4096 Sep 6 14:09 ..
-rw-r--r-- 1 root root 99 Sep 6 14:18 application.log
ubuntu@ip-172-31-31-80:~$
```

Here is your **DNS Server Setup (7 Marks)** write-up, fully rewritten and formatted for **Google Docs**. All command-line instructions are in *italics*, the structure is clean and professional, and it's ready to be copied into your documentation.

## 9. DNS Server Setup (7 Marks)

You are required to install and configure **BIND9** as a local caching DNS server, with a custom zone for `myuniversity.local`.

### Step 1: Install BIND9 and Utilities

Update your package repository and install required tools:

```
sudo apt update
sudo apt install bind9 bind9utils bind9-doc dnsutils -y
sudo apt install dig nslookup -y
```

#### Verify Installation:

```
bind9 -v
sudo systemctl status bind9
```

```
[ubuntu@ip-172-31-31-80:~]$ sudo systemctl status bind9
● named.service - BIND Domain Name Server
   Loaded: loaded (/usr/lib/systemd/system/named.service; enabled; preset: enabled)
     Active: active (running) since Sat 2025-09-06 13:10:21 UTC; 37min ago
       Main PID: 1592 (named)
          Status: "running"
             Tasks: 1 (since 13:10:21)
            Memory: 7.3M (peak: 8.0M)
              CPU: 17ms
             CGroup: /user.slice/named.service
                     └─ 1592 /usr/sbin/named -f -u bind

Sep 06 13:10:21 ip-172-31-31-80 named[11592]: network unreachable resolving './DNSKEY/IN': 2001:600:2f::f#63
Sep 06 13:10:21 ip-172-31-31-80 named[11592]: network unreachable resolving './NS/IN': 2001:600:2f::f#63
Sep 06 13:10:21 ip-172-31-31-80 named[11592]: network unreachable resolving './DNSKEY/IN': 2001:603:c27::2:30#63
Sep 06 13:10:21 ip-172-31-31-80 named[11592]: network unreachable resolving './NS/IN': 2001:603:c27::2:30#63
Sep 06 13:10:21 ip-172-31-31-80 named[11592]: network unreachable resolving './DNSKEY/IN': 2001:603:c27::2:30#63
Sep 06 13:10:21 ip-172-31-31-80 named[11592]: network unreachable resolving './NS/IN': 2001:603:c27::2:30#63
Sep 06 13:10:21 ip-172-31-31-80 named[11592]: managed-keys-zone: Initializing automatic trust anchor management for zone '.'; DNSKEY ID 20326 is now trusted, waiving the normal 30-day waiting period.
Sep 06 13:10:21 ip-172-31-31-80 named[11592]: managed-keys-zone: Initializing automatic trust anchor management for zone '.'; DNSKEY ID 38696 is now trusted, waiving the normal 30-day waiting period.
ubuntu@ip-172-31-31-80:~$
```

### Step 2: Configure Main BIND9 Options

**Backup Existing Configuration:**

```
sudo cp /etc/bind/named.conf /etc/bind/named.conf.backup  
sudo cp /etc/bind/named.conf.options /etc/bind/named.conf.options.backup  
sudo cp /etc/bind/named.conf.local /etc/bind/named.conf.local.backup
```

**Edit the Main Options File:**

```
sudo nano /etc/bind/named.conf.options
```

Replace the content with:

```
acl trusted {
```

```
    127.0.0.0/8;
```

```
    10.0.0.0/8;
```

```
    172.16.0.0/12;
```

```
    192.168.0.0/16;
```

```
};
```

```
options {
```

```
    directory "/var/cache/bind";
```

```
    dump-file "/var/cache/bind/cache_dump.db";
```

```
    statistics-file "/var/cache/bind/named_stats.txt";
```

```
    memstatistics-file "/var/cache/bind/named_mem_stats.txt";
```

```
    listen-on port 53 { any; };
```

```
    listen-on-v6 port 53 { any; };
```

```
    allow-query { trusted; };
```

```
    allow-recursion { trusted; };
```

```
forwarders {  
    8.8.8.8;  
    8.8.4.4;  
    1.1.1.1;  
    1.0.0.1;  
};  
  
forward first;  
  
recursion yes;  
  
dnssec-validation auto;  
  
version none;  
  
allow-transfer { none; };  
  
response-policy { zone "rpz.local"; };  
};  
  
  
logging {  
    channel default_debug {  
        file "data/named.run";  
        severity dynamic;  
    };  
    channel query_log {  
        file "/var/log/named/query.log" versions 3 size 5m;  
        severity info;  
        print-time yes;  
    };
```

```
print-category yes;  
print-severity yes;  
};  
category queries { query_log; };  
};
```

Save and exit: *Ctrl + X*, then *Y*, then *Enter*

### Step 3: Configure Local Zones

Edit the local zone definitions:

```
sudo nano /etc/bind/named.conf.local
```

Add:

```
zone "myuniversity.local" {  
    type master;  
    file "/etc/bind/zones/db.myuniversity.local";  
};
```

```
zone "1.168.192.in-addr.arpa" {  
    type master;  
    file "/etc/bind/zones/db.192.168.1";  
};
```

```
GNU nano 7.2                               /etc/bind/named.conf.local
zone "myuniversity.local" {
    type master;
    file "/etc/bind/zones/db.myuniversity.local";
};

zone "1.168.192.in-addr.arpa" {
    type master;
    file "/etc/bind/zones/db.192.168.1";
};

// Do any local configuration here
//

// Consider adding the 1918 zones here, if they are not used in your
// organization
//include "/etc/bind/zones.rfc1918";
```

## Step 4: Create Zone Files

### Create Directories:

```
sudo mkdir -p /etc/bind/zones
sudo mkdir -p /var/log/named
sudo chown bind:bind /var/log/named
```

### Forward Zone File

```
sudo nano /etc/bind/zones/db.myuniversity.local
```

Paste the following:

```
$TTL 604800

@ IN SOA ns1.myuniversity.local. admin.myuniversity.local. (
    2024010501 ; Serial
    604800    ; Refresh
    86400     ; Retry
    2419200   ; Expire
```

604800 ) ; Negative Cache TTL

; Name Servers

```
@ IN NS ns1.myuniversity.local.  
@ IN NS ns2.myuniversity.local.
```

; A Records

```
ns1 IN A 192.168.1.10  
ns2 IN A 192.168.1.11  
@ IN A 192.168.1.20  
www IN A 192.168.1.20  
mail IN A 192.168.1.21  
ftp IN A 192.168.1.22  
webmail IN A 192.168.1.23  
library IN A 192.168.1.24  
student IN A 192.168.1.25  
faculty IN A 192.168.1.26  
admin IN A 192.168.1.27  
lab IN A 192.168.1.30  
lab1 IN A 192.168.1.31  
lab2 IN A 192.168.1.32  
lab3 IN A 192.168.1.33
```

; CNAME Records

```
portal      IN  CNAME  student.myuniversity.local.  
cms        IN  CNAME  www.myuniversity.local.  
intranet    IN  CNAME  admin.myuniversity.local.
```

; MX Record

@ IN MX 10 mail.myuniversity.local.

## ; TXT Records

```
@ IN TXT "v=spf1 mx ~all"  
@ IN TXT "myuniversity.local domain"
```

```
[root@ip-172-31-31-80 ~]# managed-keys-zone: initializing automatic client anchor management for zone . ; socket 10_80070 is now clustered, awaiting the maximum 30-day waiting period
```

## Reverse Zone File

```
sudo nano /etc/bind/zones/db.192.168.1
```

Paste the following:

\$TTL 604800

@ IN SOA ns1.myuniversity.local. admin.myuniversity.local. (

2024010501

604800

86400

2419200

604800 )

@ IN NS ns1.myuniversity.local.

```
@ IN NS ns2.myuniversity.local.

10 IN PTR ns1.myuniversity.local.
11 IN PTR ns2.myuniversity.local.
20 IN PTR www.myuniversity.local.
21 IN PTR mail.myuniversity.local.
22 IN PTR ftp.myuniversity.local.
23 IN PTR webmail.myuniversity.local.
24 IN PTR library.myuniversity.local.
25 IN PTR student.myuniversity.local.
26 IN PTR faculty.myuniversity.local.
27 IN PTR admin.myuniversity.local.
30 IN PTR lab.myuniversity.local.
31 IN PTR lab1.myuniversity.local.
32 IN PTR lab2.myuniversity.local.
33 IN PTR lab3.myuniversity.local.
```

## Step 5: Set Ownership and Permissions

```
sudo chown -R bind:bind /etc/bind/
sudo chown -R bind:bind /var/cache/bind/
sudo chown -R bind:bind /var/log/named/

sudo chmod -R 755 /etc/bind/zones/
sudo chmod 644 /etc/bind/zones/*
sudo chmod 644 /etc/bind/named.conf*
```

## Step 6: Validate Configuration Files

```
sudo named-checkconf  
sudo named-checkzone myuniversity.local  
/etc/bind/zones/db.myuniversity.local  
sudo named-checkzone 1.168.192.in-addr.arpa /etc/bind/zones/db.192.168.1
```

(If using RPZ):

```
sudo named-checkzone rpz.local /etc/bind/zones/db.rpz.local
```

```
[ubuntu@ip-172-31-31-80:~]$ sudo cp /etc/bind/named.conf /etc/bind/named.conf.backup  
[ubuntu@ip-172-31-31-80:~]$ sudo cp /etc/bind/named.conf.options /etc/bind/named.conf.options.backup  
[ubuntu@ip-172-31-31-80:~]$ sudo cp /etc/bind/named.conf.local /etc/bind/named.conf.local.backup  
[ubuntu@ip-172-31-31-80:~]$ sudo nano /etc/bind/named.conf.options  
[ubuntu@ip-172-31-31-80:~]$ sudo nano /etc/bind/named.conf.local  
[ubuntu@ip-172-31-31-80:~]$ sudo chmod -p /etc/bind/zones  
[ubuntu@ip-172-31-31-80:~]$ sudo chmod -R /var/log/named  
[ubuntu@ip-172-31-31-80:~]$ sudo chown bindbind /var/log/named  
[ubuntu@ip-172-31-31-80:~]$ sudo nano /etc/bind/zones/db.myuniversity.local  
[ubuntu@ip-172-31-31-80:~]$ sudo chown -R bindbind:bind /etc/bind/  
[ubuntu@ip-172-31-31-80:~]$ sudo chown -R bindbind:bind /var/cache/bind/  
[ubuntu@ip-172-31-31-80:~]$ sudo chmod -R 644 /etc/bind/zones  
[ubuntu@ip-172-31-31-80:~]$ sudo chmod 644 /etc/bind/named.conf*  
[ubuntu@ip-172-31-31-80:~]$ sudo named-checkconf  
/etc/bind/named.conf.options:1: unknown option 'bashacl'  
/etc/bind/named.conf.options:1:38: 'options' redefined near 'options'  
[ubuntu@ip-172-31-31-80:~]$ sudo named-checkzone myuniversity.local /etc/bind/zones/db.myuniversity.local  
zone myuniversity.local/IN: loaded serial 2024010501  
OK  
[ubuntu@ip-172-31-31-80:~]$ sudo named-checkzone 1.168.192.in-addr.arpa /etc/bind/zones/db.192.168.1  
zone 1.168.192.in-addr.arpa/IN: loading from master file /etc/bind/zones/db.192.168.1 failed: file not found  
zone 1.168.192.in-addr.arpa/IN: not loaded due to errors.  
[ubuntu@ip-172-31-31-80:~]$ sudo named-checkzone rpz.local /etc/bind/zones/db.rpz.local  
zone rpz.local/IN: loading from master file /etc/bind/zones/db.rpz.local failed: file not found  
zone rpz.local/IN: not loaded due to errors.  
[ubuntu@ip-172-31-31-80:~]$
```

## Step 7: Start and Enable BIND9

Start and enable BIND9 service:

```
sudo systemctl start bind9  
sudo systemctl enable bind9  
sudo systemctl status bind9
```

```
[ubuntu@ip-172-31-31-80:~]$ sudo systemctl start bind9  
[ubuntu@ip-172-31-31-80:~]$ sudo systemctl start bind9  
[ubuntu@ip-172-31-31-80:~]$ sudo ufw allow 53/udp  
Rules updated  
Rules updated (v6)  
[ubuntu@ip-172-31-31-80:~]$ sudo ufw allow 53/tcp  
Rules updated  
Rules updated (v6)  
[ubuntu@ip-172-31-31-80:~]$ sudo ufw status  
Status: inactive  
[ubuntu@ip-172-31-31-80:~]$ sudo systemctl enable bind9  
Failed to enable unit: Refusing to operate on alias name or linked unit file: bind9.service  
[ubuntu@ip-172-31-31-80:~]$ sudo systemctl status bind9  
● bind9.service - BIND9 Main Name Server  
   Loaded: loaded (/usr/lib/systemd/system/named.service; enabled; preset: enabled)  
   Active: active (running) since Sat 2025-09-06 13:10:21 UTC; 2h 3min ago  
     Docs: man:named(8)  
   Main PID: 11592 (named)  
   Status: "running"  
     Tasks: 8 (limit: 1008)  
       Memory: 1.0M (peak: 6.0M)  
         CPU: 21ms  
        CGroup: /system.slice/named.service  
              └─11592 /usr/sbin/named -f -u bind  
  
Sep 06 13:10:21 ip-172-31-31-80 named[11592]: network unreachable resolving './DNSKEY/IN': 2001:500:2f::f#53  
Sep 06 13:10:21 ip-172-31-31-80 named[11592]: network unreachable resolving './NS/IN': 2001:500:2f::f#53  
Sep 06 13:10:21 ip-172-31-31-80 named[11592]: network unreachable resolving './DNSKEY/IN': 2001:500:c3:35#53  
Sep 06 13:10:21 ip-172-31-31-80 named[11592]: network unreachable resolving './NS/IN': 2001:500:c3:35#53  
Sep 06 13:10:21 ip-172-31-31-80 named[11592]: network unreachable resolving './DNSKEY/IN': 2001:503:c27::2:30#53  
Sep 06 13:10:21 ip-172-31-31-80 named[11592]: network unreachable resolving './NS/IN': 2001:503:c27::2:30#53  
Sep 06 13:10:21 ip-172-31-31-80 named[11592]: network unreachable resolving './DNSKEY/IN': 2001:500:a8:e#53  
Sep 06 13:10:21 ip-172-31-31-80 named[11592]: network unreachable resolving './NS/IN': 2001:500:a8:e#53  
Sep 06 13:10:21 ip-172-31-31-80 named[11592]: managed-keys-zone: Initializing automatic trust anchor management for zone '.'; DNSKEY ID 20326 is now trusted, waiving the normal 30-day waiting period.  
Sep 06 13:10:21 ip-172-31-31-80 named[11592]: managed-keys-zone: Initializing automatic trust anchor management for zone '.'; DNSKEY ID 38696 is now trusted, waiving the normal 30-day waiting period.  
[ubuntu@ip-172-31-31-80:~]$
```

## Step 8: Configure the Firewall

Allow DNS traffic through UFW:

```
sudo ufw allow 53/udp  
sudo ufw allow 53/tcp  
sudo ufw status
```

## Step 9: Test DNS Server

**Forward Lookup:**

```
nslookup www.myuniversity.local 127.0.0.1  
dig @127.0.0.1 www.myuniversity.local
```

```
[ubuntu@ip-172-31-31-80:~]$ dig @127.0.0.1 www.myuniversity.local  
; <>> DIG 9.18.30-0ubuntu0.24.04.2-Ubuntu <>> @127.0.0.1 www.myuniversity.local  
; (1 server found)  
; global options: +cmd  
; Got answer:  
; WARNING: .local is reserved for Multicast DNS  
; You are currently testing what happens when an mDNS query is leaked to DNS  
;-->HEADER<- opcode: QUERY, status: NXDOMAIN, id: 11382  
; flags: qr rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 1  
;; OPT PSEUDOSECTION:  
; EDNS: version: 0, flags:; udp: 1232  
; COOKIE: 7731928d257664850100000068bc50af781f74837373ebc6 (good)  
; QUESTION SECTION:  
;www.myuniversity.local. IN A  
;; AUTHORITY SECTION:  
; . 10800 IN SOA a.root-servers.net. nstld.verisign-grs.com. 2025090600 1800 900 604800 86400  
;; Query time: 7 msec  
;; SERVER: 127.0.0.1#53(127.0.0.1) (UDP)  
;; WHEN: Sat Sep 06 15:18:07 UTC 2025  
;; MSG SIZE rcvd: 154  
ubuntu@ip-172-31-31-80:~$
```

**Reverse Lookup:**

```
nslookup 192.168.1.20 127.0.0.1  
dig @127.0.0.1 -x 192.168.1.20
```

```
[ubuntu@ip-172-31-31-80:~]$ dig @127.0.0.1 -x 192.168.1.20  
; <>> DIG 9.18.30-0ubuntu0.24.04.2-Ubuntu <>> @127.0.0.1 -x 192.168.1.20  
; (1 server found)  
; global options: +cmd  
; Got answer:  
;-->HEADER<- opcode: QUERY, status: NXDOMAIN, id: 34530  
; flags: qr aa rd ra; QUERY: 1, ANSWER: 0, AUTHORITY: 1, ADDITIONAL: 1  
;; OPT PSEUDOSECTION:  
; EDNS: version: 0, flags:; udp: 1232  
; COOKIE: 31635b9defc3ab690100000068bc50e56d99a218697583fa (good)  
; QUESTION SECTION:  
;192.168.1.20.in-addr.arpa. IN PTR  
;; AUTHORITY SECTION:  
168.192.IN-ADDR.ARPA. 86400 IN SOA 168.192.IN-ADDR.ARPA. . 0 28800 7200 604800 86400  
;; Query time: 4 msec  
;; SERVER: 127.0.0.1#53(127.0.0.1) (UDP)  
;; WHEN: Sat Sep 06 15:19:01 UTC 2025  
;; MSG SIZE rcvd: 137  
ubuntu@ip-172-31-31-80:~$ $xxxxxx$
```

## External DNS Caching:

```
nslookup google.com 127.0.0.1
dig @127.0.0.1 google.com
```

```
[ubuntu@ip-172-31-31-80:~$ dig @127.0.0.1 google.com
; <>> DIG 9.18.30-0ubuntu0.24.04.2-Ubuntu <>> @127.0.0.1 google.com
; (1 server found)
; Got answer:
;+>>>HEADER<<- opcode: QUERY, status: NOERROR, id: 57308
; flags: qr rd ra; QUERY: 1, ANSWER: 6, AUTHORITY: 0, ADDITIONAL: 1
;
; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: udp: 1232
; COOKIE: dd5ff73c662ae9bb01000000680c512414620891bc92ed51 (good)
;QUESTION SECTION:
.google.com.           IN      A
;
; ANSWER SECTION:
google.com.          300    IN      A      172.253.122.138
google.com.          300    IN      A      172.253.122.102
google.com.          300    IN      A      172.253.122.100
google.com.          300    IN      A      172.253.122.101
google.com.          300    IN      A      172.253.122.113
google.com.          300    IN      A      172.253.122.139
;
; Query time: 50 msec
; SERVER: 127.0.0.1#53(127.0.0.1) (UDP)
; WHEN: Sat Sep 06 15:28:04 UTC 2025
; MSG SIZE rcvd: 163
ubuntu@ip-172-31-31-80:~$ ]
```

## 10. SSH Key Authentication + SSH Hardening (6 Marks)

This section describes how to set up secure SSH access using **key-based authentication** and harden the SSH configuration by **disabling password login** and **root access**.

### Step 1: Generate SSH Key Pair (On Client Machine)

On the client (your local machine), generate a secure SSH key pair:

```
ssh-keygen -t rsa -b 4096 -C "your_email@domain.com"
```

- When prompted, press **Enter** to accept the default location:  
`~/ssh/id_rsa`
- Set a **passphrase** for added security:  
Example: `test123`

```
[ubuntu@ip-172-31-31-80:~]$ ssh-keygen -t rsa -b 4096 -C "your_email@domain.com"
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ubuntu/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ubuntu/.ssh/id_rsa
Your public key has been saved in /home/ubuntu/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:@gusd1Rq91Af9+o7ib/pjV3exYzseM0erah/doc your_email@domain.com
The key's randomart image is:
+---[RSA 4096]---+
| o . o .
| . + + .
| . = + .
| o . o S o .
| = . o + o o ..
| + . o o o o o o
| . * . o Eee
| . B+o o o o o o
+---[SHA256]---+
ubuntu@ip-172-31-31-80:~$ ]
```

## Step 2: Copy Public Key to Server

Use `ssh-copy-id` to copy your public key to the server:

```
ssh-copy-id username@server_ip
```

Replace `username` and `server_ip` with your actual server credentials.

## Step 3: Set Up `authorized_keys` on Server (Manual Method)

If `ssh-copy-id` is not available, manually configure the key:

### A. Create `.ssh` Directory

```
mkdir -p ~/.ssh
```

### B. Append Public Key

Assuming you uploaded `id_rsa.pub` to the server:

```
cat ~/id_rsa.pub >> ~/.ssh/authorized_keys
```

### C. Set Correct Permissions

```
chmod 700 ~/.ssh
```

```
chmod 600 ~/.ssh/authorized_keys
```

#### D. Clean Up

```
rm ~/id_rsa.pub
```

### Step 4: Test SSH Key Authentication

From the client, test SSH login:

```
ssh username@server_ip
```

You should be prompted for the **passphrase** of your SSH key, not the account password.

### Step 5: Harden SSH Server Configuration

Edit the SSH daemon config file:

```
sudo nano /etc/ssh/sshd_config
```

Update or add the following lines:

```
PermitRootLogin no
```

```
PasswordAuthentication no
```

```
ChallengeResponseAuthentication no
```

```
UsePAM yes
```

Optional (for extra security):

```
PermitEmptyPasswords no
```

```
AllowUsers username
```

Replace `username` with your actual non-root user.

## Step 6: Restart SSH Service

Apply the changes by restarting the SSH daemon:

```
sudo systemctl restart ssh
```

## Step 7: Final Verification

Try connecting from the client again:

```
ssh username@server_ip
```

You should be able to log in using only the SSH key.

**Password login and root login should now be disabled.**

# 11. Systemd Service Creation (6 Marks)

Create a simple script and configure a `systemd` service to run it automatically at system boot.

## Step 1: Create a Simple Script

1. Create the script directory:

```
sudo mkdir -p /usr/local/bin
```

**2. Create a new script file:**

```
sudo nano /usr/local/bin/my-service.sh
```



**Inside the script, you can add something simple, for example:**

```
#!/bin/bash
echo "Service started at $(date)" >> /var/log/my-service.log
```

## **Step 2: Make the Script Executable**

**1. Set executable permissions:**

```
sudo chmod +x /usr/local/bin/my-service.sh
```

**2. Set ownership to root:**

```
sudo chown root:root /usr/local/bin/my-service.sh
```

**3. Verify the permissions:**

```
ls -la /usr/local/bin/my-service.sh
```

## **Step 3: Create the systemd Service File**

**1. Create a new service file:**

```
sudo nano /etc/systemd/system/my-service.service
```

**Content to put in file :**

*[Unit]*

*Description=My Custom Service*

*After=network.target*

*[Service]*

*ExecStart=/usr/local/bin/my-service.sh*

*Restart=always*

*User=root*

*[Install]*

*WantedBy=multi-user.target*

## Step 4: Set Service File Permissions

1. **Set correct file permissions:**

```
sudo chmod 644 /etc/systemd/system/my-service.service
```

2. **Set file ownership:**

```
sudo chown root:root /etc/systemd/system/my-service.service
```

## Step 5: Reload systemd and Enable the Service

1. **Reload systemd to recognize the new service:**

```
sudo systemctl daemon-reload
```

2. **Enable the service to start at boot:**

```
sudo systemctl enable my-service.service
```

3. **Start the service immediately:**

```
sudo systemctl start my-service.service
```

**4. Check the status of the service:**

```
sudo systemctl status my-service.service
```

## **Step 6: Verify the Service is Working**

**1. Check if the service is currently active:**

```
sudo systemctl is-active my-service.service
```

**2. Confirm the service is enabled on boot:**

```
sudo systemctl is-enabled my-service.service
```

**3. View real-time logs for the service:**

```
sudo journalctl -u my-service.service -f
```

**4. Check the script's log output:**

```
sudo tail -f /var/log/my-service.log
```

## **Step 7: Test Auto-Start at Boot**

**1. Reboot the system:**

```
sudo reboot
```

**2. After reboot, confirm the service has started:**

```
sudo systemctl status my-service.service  
sudo systemctl is-active my-service.service
```

## 12. Disk Partitioning & Mounting (7 Marks)

Create a new partition, format it as `ext4`, mount it permanently using `/etc/fstab`, and test persistence across reboots.

### Step 1: List Available Disks

List all block devices on the system:

```
lsblk
```

### Step 2: Create a Virtual Disk File

Create a 100MB virtual disk file for partitioning:

```
sudo dd if=/dev/zero of=/tmp/virtual_disk.img bs=1M count=100
```

### Step 3: Create Loop Device

Associate the virtual disk with a loop device:

```
sudo losetup /dev/loop6 /tmp/virtual_disk.img
```

### Step 4: Verify Loop Device

Check if the loop device is created:

`lsblk`

You should see `/dev/loop6` listed.

## Step 5: Partition the Loop Device

Create a partition on the loop device:

`sudo fdisk /dev/loop`

Use the following keys inside `fdisk`:

- `n` (to create new partition)
- `p` (for primary)
- `1` (partition number)
- `Enter` (default first sector)
- `Enter` (default last sector)
- `w` (to write and exit)

## Step 6: Update Kernel Partition Table

Force the kernel to read the new partition table:

`sudo partprobe /dev/loop6`

## Step 7: Verify Partition

Check the new partition:

`lsblk`

You should now see `/dev/loop6p1`.

## **Step 8: Format the Partition as ext4**

Format the partition with the ext4 filesystem:

```
sudo mkfs.ext4 /dev/loop6p1
```

## **Step 9: Create a Mount Point**

Create a directory to mount the new partition:

```
sudo mkdir /mnt/newdisk
```

## **Step 10: Get the Partition UUID**

Fetch the UUID of the new partition:

```
sudo blkid /dev/loop6p1
```

Copy the `UUID=...` string (you can use UUID or device name in fstab)

## **Step 11: Add Entry to `/etc/fstab`**

Edit the fstab file to make the mount persistent:

```
sudo nano /etc/fstab
```

Add the following line at the end of the file:

```
/dev/loop6p1 /mnt/newdisk ext4 defaults 0 2
```

Or use the UUID for better stability:

```
UUID=your-uuid-here /mnt/newdisk ext4 defaults 0 2
```

## **Step 12: Test Mounting**

Reload systemd and check disk mounts:

```
sudo systemctl daemon-reload
df -h
```

Confirm that `/mnt/newdisk` is listed.

### **Step 13: Test Write Access**

Create a test file to ensure write access:

```
sudo touch /mnt/newdisk/test_file
ls -la /mnt/newdisk/
```

### **Final Step (Recommended)**

To ensure the disk auto-mounts after reboot:

```
sudo reboot
```

Then check again:

```
lsblk
df -h
```

## **13. Postfix Mail Server (Local Only) (6 Marks)**

Install and configure Postfix for local-only mail delivery, and send a test mail between users on the same system.

### **Step 1: Update Package Lists**

Run the following command to update the system package lists:

```
sudo apt update
```

## **Step 2: Install Postfix and Mail Utilities**

Install the required packages:

```
sudo apt install postfix mailutils -y
```

During installation:

- Choose “**Local only**” when prompted.
- For **System mail name**, accept the default (your system hostname).

## **Step 3: Configure Postfix for Local Delivery**

Edit the main configuration file:

```
sudo nano /etc/postfix/main.cf
```

Ensure the following lines are correctly set:

```
myhostname = localhost
```

```
mydomain = localdomain
```

```
myorigin = $mydomain
```

```
inet_interfaces = loopback-only
```

```
mydestination = $myhostname, localhost.$mydomain, localhost
```

## **Step 4: Restart and Enable Postfix**

```
sudo systemctl restart postfix
```

```
sudo systemctl enable postfix
```

## Step 5: Check Postfix Service Status

```
sudo systemctl status postfix
```

## Step 6: Create Test Users

```
sudo adduser testuser1  
sudo adduser testuser2
```

## Step 7: Send a Test Email

Send a test email from `testuser1` to `testuser2`:

```
echo "This is a test email from testuser1" | mail -s "Test Subject" testuser2
```

## Step 8: Check the Mail Queue

```
mailq
```

## Step 9: Read Mail as testuser2

Switch to the user:

```
sudo su - testuser2 mail
```

Inside the mail utility:

- Press `1` to read the first message.
- Press `q` to quit.

## Step 10: Verify Mail Delivery in Logs

```
sudo tail -f /var/log/mail.log
```

## **Step 11: Test Bidirectional Email**

Reply from `testuser2` to `testuser1`:

```
sudo su - testuser2
echo "Reply from testuser2" | mail -s "Reply Test" testuser1
exit
```

Check the mail as `testuser1`:

```
sudo su - testuser1 mail
```

## **14. Backup & Restore Project (6 Marks)**

Write a script to back up `/var/www/html` to `/backup/` with a timestamp and test restoring it

### **Step 1: Create Backup Directory and Test Data**

```
sudo mkdir -p /backup
sudo mkdir -p /var/www/html
echo "Original website content" | sudo tee /var/www/html/index.html
echo "Test page content" | sudo tee /var/www/html/test.html
sudo mkdir /var/www/html/images
echo "Image placeholder" | sudo tee /var/www/html/images/logo.txt
```

## **Step 2: Create Backup Script**

```
sudo nano /usr/local/bin/backup_www.sh
```

Example content for the script:

```
#!/bin/bash

TIMESTAMP=$(date +"%Y%m%d_%H%M%S")

tar -czf /backup/www_backup_$TIMESTAMP.tar.gz /var/www/html
```

## **Step 3: Create Restore Script**

```
sudo nano /usr/local/bin/restore_www.sh
```

Example content for the script:

```
#!/bin/bash

BACKUP_FILE="/backup/$1"

if [ -f "$BACKUP_FILE" ]; then

    sudo rm -rf /var/www/html/*
    sudo tar -xzf "$BACKUP_FILE" -C /
    echo "Restore completed from $BACKUP_FILE"

else

    echo "Backup file not found!"
```

## **Step 4: Make Scripts Executable**

```
sudo chmod +x /usr/local/bin/backup_www.sh  
sudo chmod +x /usr/local/bin/restore_www.sh
```

## **Step 5: Test Backup Creation**

```
sudo /usr/local/bin/backup_www.sh
```

## **Step 6: List Created Backup**

```
ls -la /backup/
```

## **Step 7: Modify Original Files (Simulate Changes)**

```
echo "Modified content - version 2" | sudo tee /var/www/html/index.html  
echo "New file after backup" | sudo tee /var/www/html/new_file.txt
```

## **Step 8: Test Restore**

Get the name of the backup file:

```
BACKUP_NAME=$(ls /backup/www_backup_.tar.gz | head -1 | xargs basename)*
```

Run the restore:

```
sudo /usr/local/bin/restore_www.sh "$BACKUP_NAME"
```

### **Step 9: Verify Restore**

```
cat /var/www/html/index.html  
ls -la /var/www/html/
```

## **15. Containerization Challenge (8 Marks)**

Install Docker (or Podman), create a container running Nginx, map it to port 8080, and verify the service is accessible.

### **Step 1: Update Package Lists**

```
sudo apt update
```

### **Step 2: Install Docker**

```
sudo apt install docker.io -y
```

### **Step 3: Start and Enable Docker Service**

```
sudo systemctl start docker  
sudo systemctl enable docker
```

### **Step 4: Add User to Docker Group (Optional)**

```
sudo usermod -aG docker $USER
```

## **Step 5: Verify Docker Installation**

```
sudo docker --version  
sudo docker info
```

## **Step 6: Pull Nginx Image**

```
sudo docker pull nginx:latest
```

## **Step 7: Create and Run Nginx Container**

```
sudo docker run -d --name nginx-container -p 8080:80 nginx:latest
```

## **Step 8: Verify Container is Running**

```
sudo docker ps
```

## **Step 9: Check Container Logs**

```
sudo docker logs nginx-container
```

## **Step 10: Test Nginx Service Locally**

```
curl http://localhost:8080
```

## **Step 11: Check Port Binding**

```
sudo lsof -i :8080
```

## **Step 12: Test Service Response (Headers Only)**

*curl -I <http://localhost:8080>*

## **Step 13: View Container Details**

*sudo docker inspect nginx-container*