LINUX FUNDAMENTALS PROJECT:
BUILD A MULTI-USER LINUX SERVER

DOCUMENTATION

CHERISH P. GOHEE

ROSEMARIE C. MAHINAY

REX C. SUMALINOG

July 2025

INTRODUCTION

In the evolving landscape of information technology, the Linux operating system has emerged as a critical component in modern server environments due to its reliability, flexibility, and open-source nature. It serves as the backbone for a vast number of systems ranging from small-scale deployments to enterprise-grade infrastructures. As industries increasingly rely on Linux-based systems for hosting services, managing networks, and securing data, the need for proficient system administrators capable of configuring and maintaining such environments has become more essential than ever.

This project, entitled “Linux Fundamentals Project: Multi-User Secure Server Environment,” is designed to simulate a real-world Linux system administration task. It involves configuring a secure and efficient multi-user server setup using CentOS Stream 9 as the server operating system and Ubuntu as the client. The goal is to demonstrate key system administration competencies, including user and group management, secure SSH configuration, file permission handling, firewall setup, web server deployment, and automated system monitoring through scripting and cron jobs.

Through this undertaking, the project aims to provide a practical environment where theoretical knowledge gained in the classroom can be applied to realistic administrative tasks. The configuration of a secure server-client architecture not only enhances the participants’ technical skills but also reinforces best practices in system security, access control, and performance monitoring. By the end of the project, the team is expected to deliver a fully functional and secure Linux environment that reflects real-world administrative challenges and solutions.

TASK AND REQUIREMENTS

USER AND GROUP MANAGEMENT

1. Create users: adminuser, devuser, guestuser On CentOS (server)

   These accounts simulate different access levels:

```
[server@server ~]$ sudo useradd adminuser
[sudo] password for server:
[server@server ~]$ sudo useradd devuser
[server@server ~]$ sudo useradd guestuser
```

   Add password:

```
[server@server ~]$ sudo passwd adminuser
Changing password for user adminuser.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
[server@server ~]$ sudo passwd devuser
Changing password for user devuser.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
[server@server ~]$ sudo passwd guestuser
Changing password for user guestuser.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
```

   On Ubuntu (client)

   These accounts are for testing SSH access from the client:

```
client@client:~$ sudo useradd -m ubuntu_admin
[sudo] password for client:
client@client:~$ sudo passwd ubuntu_admin
New password:
Retype new password:
passwd: password updated successfully
client@client:~$ sudo useradd -m ubuntu_dev
client@client:~$ sudo passwd ubuntu_dev
New password:
Retype new password:
passwd: password updated successfully
client@client:~$ sudo useradd -m ubuntu_guest
client@client:~$ sudo passwd ubuntu_guest
New password:
Retype new password:
passwd: password updated successfully
```

   Creates matching local user accounts on the client with home directories (-m), and sets their passwords.

2. Create group: developers

```
[server@server ~]$ sudo groupadd developers
```

Creates a group called developers, used for shared access and permission control.

3. Assign:

   o devuser to developers

   o adminuser to sudo/wheel

   On CentOS (server)

```
[server@server ~]$ sudo usermod -aG developers devuser
[server@server ~]$ sudo usermod -aG wheel adminuser
```

Adds devuser to developers group and gives adminuser sudo privileges by adding them to the wheel group.

Verify:

```
Last login: Mon Jul 28 08:19:36 2025 from 192.168.219.9
[adminuser@server ~]$ id adminuser
uid=1001(adminuser) gid=1001(adminuser) groups=1001(adminuser),10(wheel)
[adminuser@server ~]$ id devuser
uid=1002(devuser) gid=1002(devuser) groups=1002(devuser),1004(developers)
[adminuser@server ~]$ id guestuser
uid=1003(guestuser) gid=1003(guestuser) groups=1003(guestuser)
```

Look For:

*adminuser should include **wheel***

*devuser should include **developers***

guestuser should have its own UID and groups

PASSWORD AND POLICIES

1. Password expires every 60 days
2. Warning issued 14 days before expiration

```
[server@server ~]$ sudo chage -M 60 -W 14 adminuser
[server@server ~]$ sudo chage -M 60 -W 14 devuser
[server@server ~]$ sudo chage -M 60 -W 14 guestuser
```

Sets password expiration to 60 days and shows a warning 14 days before for all users.

Verify:

```
[adminuser@server ~]$ sudo chage -l adminuser
[sudo] password for adminuser:
Last password change                                    : Jul 27, 2025
Password expires                                        : Sep 25, 2025
Password inactive                                       : never
Account expires                                         : never
Minimum number of days between password change          : 0
Maximum number of days between password change          : 60
Number of days of warning before password expires       : 14
[adminuser@server ~]$ sudo chage -l devuser
Last password change                                    : Jul 27, 2025
Password expires                                        : Sep 25, 2025
Password inactive                                       : never
Account expires                                         : never
Minimum number of days between password change          : 0
Maximum number of days between password change          : 60
Number of days of warning before password expires       : 14
[adminuser@server ~]$ sudo chage -l guestuser
Last password change                                    : Jul 27, 2025
Password expires                                        : Sep 25, 2025
Password inactive                                       : never
Account expires                                         : never
Minimum number of days between password change          : 0
Maximum number of days between password change          : 60
Number of days of warning before password expires       : 14
```

Look For:

Maximum number of days between password change: 60

Number of days of warning before password expires: 14

SHARED DIRECTORY

1. Create /srv/devshare for the developers group

```
[server@server ~]$ sudo mkdir -p /srv/devshare
```

Creates the shared folder path for developers.

2. Ownership: root:developers

```
[server@server ~]$ sudo chown root:developers /srv/devshare
```

Sets ownership to root and group ownership to developers.

3. Permissions:

   o Group read/write

   o setgid enabled

```
[server@server ~]$ sudo chmod 2770 /srv/devshare
```

Gives full access to owner and group. The 2 sets setgid, so new files inherit the group.

4. guestuser must have read-only access to /srv/devshare

   Install acl package:

```
[server@server ~]$ sudo dnf install -y acl
CentOS Stream 9 - BaseOS                        2.4 MB/s | 8.7 MB     00:03
CentOS Stream 9 - AppStream                     3.2 MB/s |  24 MB     00:07
```

Installs ACL (Access Control Lists) to manage detailed file permissions.

read-only access to /srv/devshare

```
[server@server ~]$ sudo setfacl -m u:guestuser:r-X /srv/devshare
```

Gives guestuser read-only access to the shared directory.

Verify:

```
[adminuser@server ~]$ ls -ld /srv/devshare
drwxrws---+ 2 root developers 6 Jul 28 00:33 /srv/devshare
```

Permissions like drwxrws---

Owner = root

Group = developers

SSH CONFIGURATION

1. SSH key-based login from Ubuntu client for both members
   Adminuser

```
client@client:~$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/client/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/client/.ssh/id_rsa
Your public key has been saved in /home/client/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:JTK5mQ+MYjFqs9dJWbm5gMhHswYev4TDLjYF3qqmzjI client@client
The key's randomart image is:
+---[RSA 3072]----+
|                 |
|        ..       |
| +oo  +o. .      |
|=.Xo+oo*oo       |
|.@oX.+*oS        |
|o.X.+ oo.        |
|.* o o ..        |
|E.o              |
|B+               |
+----[SHA256]-----+
client@client:~$ ssh-copy-id adminuser@192.168.219.10
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/client/.ssh/id_rsa.pub"
The authenticity of host '192.168.219.10 (192.168.219.10)' can't be established.
ED25519 key fingerprint is SHA256:UFwK1BnEBXKywPzPteU6aq2gtVREdN7rpwHJ3to0rJE.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yea
Please type 'yes', 'no' or the fingerprint: yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
adminuser@192.168.219.10's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'adminuser@192.168.219.10'"
and check to make sure that only the key(s) you wanted were added.
```

   Devuser

```
client@client:~$ ssh-copy-id devuser@192.168.219.10
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/client/.ssh/id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
devuser@192.168.219.10's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'devuser@192.168.219.10'"
and check to make sure that only the key(s) you wanted were added.
```

   Generates and copies public keys from Ubuntu client to CentOS server for passwordless login.

2. Disable password login for adminuser
   Open the SSH configuration file:

```
[server@server ~]$ sudo vi /etc/ssh/sshd_config
[sudo] password for server:
```

   Opens the SSH config.

Scroll to the bottom and add this block:

```
Subsystem       sftp    /usr/libexec/openssh/sftp-server

Match user adminuser
        PasswordAuthentication no
```

Keeps password login enabled for others but disables it only for adminuser.

Verify:

```
client@client:~$ ssh adminuser@192.168.219.10
Activate the web console with: systemctl enable --now cockpit.socket

Last login: Mon Jul 28 06:18:20 2025 from 192.168.219.9
[adminuser@server ~]$ ssh devuser@192.168.219.10
devuser@192.168.219.10's password:
Last login: Mon Jul 28 06:18:42 2025 from 192.168.219.10
[devuser@server ~]$ ssh adminuser@192.168.219.10
adminuser@192.168.219.10: Permission denied (publickey,gssapi-keyex,gssapi-with-mic).
[devuser@server ~]$
```

You should not be prompted for a password if key is set up.
Use another machine → Should show: Permission denied

FIREWALL AND NETWORK CONFIGURATION

1. CentOS and Ubuntu client must be within the same network

CentOS

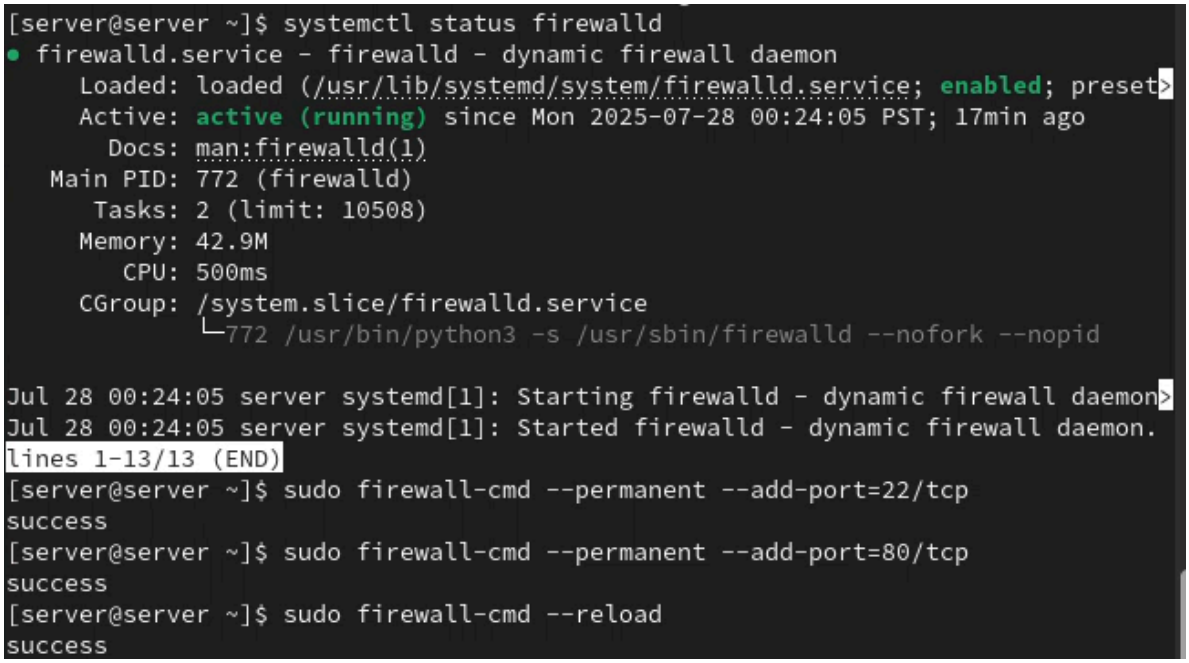
```
[server@server ~]$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defaul
t qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP gr
oup default qlen 1000
    link/ether 08:00:27:57:5e:66 brd ff:ff:ff:ff:ff:ff
    inet 192.168.219.10/24 brd 192.168.219.255 scope global dynamic noprefixrout
e enp0s3
       valid_lft 409sec preferred_lft 409sec
    inet6 fe80::a00:27ff:fe57:5e66/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP gr
oup default qlen 1000
    link/ether 08:00:27:15:2f:9f brd ff:ff:ff:ff:ff:ff
    inet 10.0.3.15/24 brd 10.0.3.255 scope global dynamic noprefixroute enp0s8
       valid_lft 84109sec preferred_lft 84109sec
    inet6 fd17:625c:f037:3:a00:27ff:fe15:2f9f/64 scope global dynamic noprefixro
ute
       valid_lft 86274sec preferred_lft 14274sec
    inet6 fe80::a00:27ff:fe15:2f9f/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
```

Ubuntu

```
client@client:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 08:00:27:1d:12:31 brd ff:ff:ff:ff:ff:ff
    inet 192.168.219.9/24 metric 100 brd 192.168.219.255 scope global dynamic enp0s3
       valid_lft 519sec preferred_lft 519sec
    inet6 fe80::a00:27ff:fe1d:1231/64 scope link
       valid_lft forever preferred_lft forever
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 08:00:27:d5:20:dc brd ff:ff:ff:ff:ff:ff
    inet 10.0.3.15/24 metric 100 brd 10.0.3.255 scope global dynamic enp0s8
       valid_lft 84820sec preferred_lft 84820sec
    inet6 fd17:625c:f037:3:a00:27ff:fed5:20dc/64 scope global dynamic mngtmpaddr noprefixroute
       valid_lft 86040sec preferred_lft 14040sec
    inet6 fe80::a00:27ff:fed5:20dc/64 scope link
       valid_lft forever preferred_lft forever
```

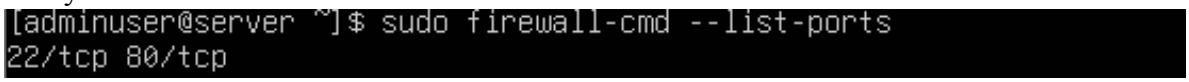
Ensures CentOS and Ubuntu are on the same subnet (192.168.219.10)

2. CentOS server must allow only ports 22 (SSH) and 80 (HTTP). Deny all other incoming traffic



Opens SSH and HTTP ports. All others are blocked by default zone.

Verify:



Only 22/tcp and 80/tcp should be listed

WEB SERVER DEPLOYMENT

1. Install a web server (Apache or Nginx) on the CentOS server

```
[server@server ~]$ sudo dnf install -y httpd
Last metadata expiration check: 0:03:05 ago on Mon 28 Jul 2025 12:40:59 AM PST.
```

```
Complete!
[server@server ~]$ sudo systemctl enable --now httpd
Created symlink /etc/systemd/system/multi-user.target.wants/httpd.service → /usr
/lib/systemd/system/httpd.service.
```

Installs and starts Apache web server.

2. Ensure Ubuntu client can access default web page via port 80

```
client@client:~$ curl 192.168.219.10:80
```

Output:

```
          </li>
        </ul>
      </section>
      <section class="links">
        <h6><i class="fas fa-users"></i> Community</h6>
        <ul>
          <li>
            <a href="https://wiki.centos.org/Contribute">Contribute</a>
          </li>
          <li>
            <a href="https://www.centos.org/forums/">Forums</a>
          </li>
          <li>
            <a href="https://wiki.centos.org/GettingHelp/ListInfo">Mailing Lists</a>
          </li>
          <li>
            <a href="https://wiki.centos.org/irc">IRC</a>
          </li>
          <li>
            <a href="/community/calendar/">Calendar &amp;amp; IRC Meeting List</a>
          </li>
          <li>
            <a href="http://planet.centos.org/">Planet</a>
          </li>
          <li>
            <a href="https://wiki.centos.org/ReportBugs">Submit Bug</a>
          </li>
        </ul>
      </section>
      <section class="project">
        <h2>The CentOS Project</h2>
        <p class="lead">Community-driven free software effort focused on delivering a robust open source ecosystem ar
        <div class="lead social">
          <a href="https://www.facebook.com/groups/centosproject/"><i class="fab fa-facebook-f"></i></a> <a href="htt
-twitter"></i></a> <a href="https://youtube.com/TheCentOSProject"><i class="fab fa-youtube"></i></a> <a href="https://w
ab fa-linkedin"></i></a> <a href="https://www.reddit.com/r/CentOS/"><i class="fab fa-reddit"></i></a>
        </div>
      </section>
    </div>
    <div class="row">
      <section class="copyright">
        <p>Copyright © 2021 The CentOS Project | <a href="/legal">Legal</a> | <a href="/legal/privacy">Privacy</a> |
ntos.org">Site source</a></p>
      </section>
    </div>
  </div>
 </footer>
</body>
</html>
client@client:~$
```

You should see HTML content of Apache test page.

SYSTEM MONITORING AND LOGGING

Create two BASH scripts:

1. CPU Utilization Monitor

```
[server@server ~]$ echo $PATH
/home/server/.local/bin:/home/server/bin:/usr/local/bin:/usr/local/sbin:/usr/bin
:/usr/sbin
[server@server ~]$ sudo vi /usr/local/bin/cpu_monitoring.sh
```

Write this inside /usr/local/bin/cpu_monitoring.sh

```
#!/bin/bash

LOG_DIR="/var/log/monitoring_logs"
LOG_FILE="${LOG_DIR}/cpu_monitoring.log"

WARNING_THRESHOLD=80                            # Threshold for the warning message (Used for CPU)
CRITICAL_THRESHOLD=90                           # Critical Range (Used for CPU)

sudo mkdir -p "$LOG_DIR"
sudo touch "$LOG_FILE"

TIMESTAMP=$(date +"%Y-%m-%d %H:%M:%S")          # Prints timestamp in "YYYY-MM-DD H:M:S" format


# Parses the idle/free cpu then subtracts it from 100 to calculate the usage
# CPU_USAGE will hold the floating-point result (e.g., 14.3)
CPU_USAGE=$(echo "100 - $(top -bn1 | grep '%Cpu(s):' | cut -d, -f 4 | awk '{print $1}')" | bc)


# Rounds off the CPU_USAGE output for it to work with arithmetic comparison
CPU_USAGE_ROUND=$(printf "%.0f\n" "$CPU_USAGE")


# Conditional to determine the status of CPU
if (( CPU_USAGE_ROUND < WARNING_THRESHOLD )); then
    STATUS="OK"
    EXIT_CODE=0
elif (( CPU_USAGE_ROUND >= WARNING_THRESHOLD && CPU_USAGE_ROUND < CRITICAL_THRESHOLD )); then
    STATUS="WARNING"
    EXIT_CODE=1
else
    STATUS="CRITICAL"
    EXIT_CODE=2
fi


echo "$TIMESTAMP - $STATUS - CPU Usage: ${CPU_USAGE_ROUND}%" | sudo tee -a "$LOG_FILE" > /dev/null
exit $EXIT_CODE
```

Make it executable:

```
[server@server ~]$ sudo chmod +x /usr/local/bin/cpu_monitoring.sh
[server@server ~]$ sudo ls -l /usr/local/cpu_monitoring.sh
ls: cannot access '/usr/local/cpu_monitoring.sh': No such file or directory
[server@server ~]$ sudo ls -l /usr/local/bin/cpu_monitoring.sh
-rwxr-xr-x. 1 root root 729 Jul 28 01:31 /usr/local/bin/cpu_monitoring.sh
```

Verify:

```
[adminuser@server ~]$ cat /var/log/monitoring_logs/cpu_monitoring.log
2025-07-28 01:39:14 - OK - CPU Usage: 20%
```

Line like: 2025-07-28 01:39:14 - OK - CPU Usage: 20%

2. Memory Utilization Monitor

```
[server@server ~]$ sudo vi /usr/local/bin/mem_monitoring.sh
```

Write this inside /usr/local/bin/mem_monitoring.sh

```
#!/bin/bash

LOG_DIR="/var/log/monitoring_logs"
LOG_FILE="${LOG_DIR}/mem_monitoring.log"

WARNING_THRESHOLD=80                            # Threshold for the warning message (Used for Memory)
CRITICAL_THRESHOLD=90                           # Critical Range (Used for Memory)

mkdir -p "$LOG_DIR"
touch "$LOG_FILE"

TIMESTAMP=$(date +"%Y-%m-%d %H:%M:%S")          # Get the current timestamp

# Calculates memory utilization percentage
# This computes ((Total - Available) / Total) * 100.
MEM_UTIL_FLOAT=$(free -m | awk '/Mem:/ {printf "%.2f\n", (($2 - $7) / $2) * 100}')

# Convert to integer percentage for comparison
MEM_UTIL_INT=$(printf "%.0f\n" "$MEM_UTIL_FLOAT")

# Determine the status based on the specified threshold
if (( MEM_UTIL_INT < WARNING_THRESHOLD )); then
    STATUS="OK"
    EXIT_CODE=0
elif (( MEM_UTIL_INT >= WARNING_THRESHOLD && MEM_UTIL_INT < CRITICAL_THRESHOLD )); then
    STATUS="WARNING"
    EXIT_CODE=1
else
    STATUS="CRITICAL"
    EXIT_CODE=2
fi

# Log the result to the specified file
echo "$TIMESTAMP - $STATUS - Mem Usage: ${MEM_UTIL_INT}%" | sudo tee -a "$LOG_FILE" > /dev/null

# Exit with the determined code (useful for external monitoring systems)
exit $EXIT_CODE
```

Make it executable:

```
[server@server ~]$ sudo chmod +x /usr/local/bin/mem_monitoring.sh
[server@server ~]$ mem_monitoring.sh
```

Verify:

```
[server@server ~]$ sudo cat /var/log/monitoring_logs/mem_monitoring.log
2025-07-28 01:51:38 - OK - Mem Usage: 0%
2025-07-28 01:53:49 - OK - Mem Usage: 55%
```

Log lines like: 2025-07-28 01:53:49 - OK - Mem Usage: 55%

Cron Job:
Scripts must run every 10 minutes via cron

```
[server@server ~]$ sudo crontab -e
```

Add:

```
*/10 * * * * /usr/local/bin/cpu_monitoring.sh
*/10 * * * * /usr/local/bin/mem_monitoring.sh
```

Schedules both scripts to run every 10 minutes and log system performance.

Verify:
After 10 minutes:

cat /var/log/monitoring_logs/cpu_monitoring.log

```
[adminuser@server ~]$ cat /var/log/monitoring_logs/cpu_monitoring.log
2025-07-28 01:39:14 - OK - CPU Usage: 20%
2025-07-28 02:00:01 - OK - CPU Usage: 25%
2025-07-28 02:40:02 - OK - CPU Usage: 25%
2025-07-28 03:20:02 - OK - CPU Usage: 35%
2025-07-28 03:30:01 - OK - CPU Usage: 31%
2025-07-28 03:40:01 - OK - CPU Usage: 45%
2025-07-28 03:50:02 - OK - CPU Usage: 39%
2025-07-28 04:00:02 - OK - CPU Usage: 35%
2025-07-28 04:10:02 - OK - CPU Usage: 39%
2025-07-28 04:20:02 - OK - CPU Usage: 55%
2025-07-28 04:30:02 - OK - CPU Usage: 53%
```

cat /var/log/monitoring_logs/mem_monitoring.log

```
[adminuser@server ~]$ cat /var/log/monitoring_logs/mem_monitoring.log
2025-07-28 01:51:38 - OK - Mem Usage: 0%
2025-07-28 01:53:49 - OK - Mem Usage: 55%
2025-07-28 02:00:01 - OK - Mem Usage: 55%
2025-07-28 02:40:02 - OK - Mem Usage: 55%
2025-07-28 03:20:02 - OK - Mem Usage: 55%
2025-07-28 03:30:01 - OK - Mem Usage: 55%
2025-07-28 03:40:01 - OK - Mem Usage: 55%
2025-07-28 03:50:02 - OK - Mem Usage: 55%
2025-07-28 04:00:02 - OK - Mem Usage: 55%
2025-07-28 04:10:02 - OK - Mem Usage: 55%
2025-07-28 04:20:02 - OK - Mem Usage: 55%
```

New entries with timestamps matching every 10 minutes