

**Rotimi Ayomidimeji Edward**

**0102407165**

**Semester: 8**

**Linux Administration Tasks.**

**Task 1: User & Group Management Automation**

**Objective**

Write a Bash script to create 5 new users, add them to a group devteam, set their passwords, and force them to change the password on first login.

**Step-by-Step Commands Used**

**Step 1: Create the script**

sudo nano /usr/local/sbin/provision\_devteam.sh

paste inside

#!/usr/bin/env bash

set -euo pipefail

GROUP=devteam

USERS=(dev1 dev2 dev3 dev4 dev5)

TEMP\_PASS='P@ssw0rd123'

# Create group if missing

getent group "$GROUP" >/dev/null || sudo groupadd "$GROUP"

# Create users

for u in "${USERS[@]}"; do

if ! id "$u" &>/dev/null; then

sudo useradd -m -s /bin/bash -G "$GROUP" "$u"

echo "$u:$TEMP\_PASS" | sudo chpasswd

sudo chage -d 0 "$u" # force password change on first login

fi

done

echo "Provisioned: ${USERS[\*]} in group $GROUP"

**Step 2: Make it executable**

sudo chmod +x /usr/local/sbin/provision\_devteam.sh

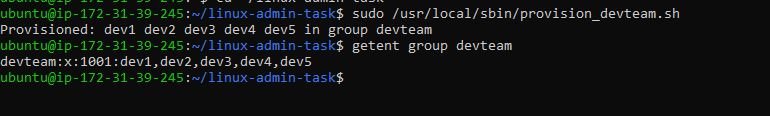
**Step 3: Run the script**

sudo /usr/local/sbin/provision\_devteam.sh

**Step 4: Verify**

Check if the group and users exist:

getent group devteam



**Task 2: File Permissions & ACLs Project**

**Objective**

Create a shared directory /shared\_data where group members can **read and write** but not delete each other’s files. Use ACLs to grant **read-only access** to one extra user outside the group.

**Step-by-Step Commands Used**

**Step 1: Create the shared directory**

sudo mkdir -p /shared\_data

sudo chgrp devteam /shared\_data

**Step 2: Set permissions**

# rwx for owner and group, none for others

# sticky bit (1) prevents users from deleting others’ files

sudo chmod 1770 /shared\_data

**Step 3: Add default ACLs**

This ensures new files in /shared\_data automatically get group permissions.

sudo setfacl -d -m g:devteam:rwx /shared\_data

**Step 4: Create the outside user**

sudo useradd -m -s /bin/bash auditor

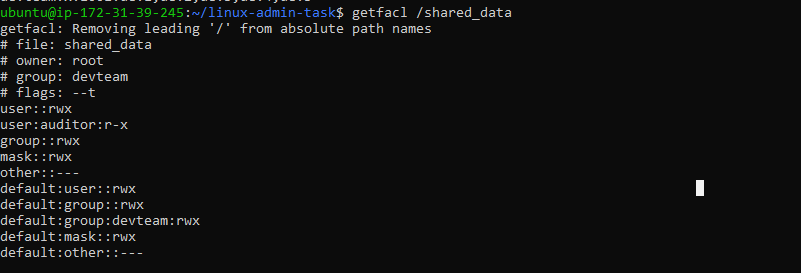
Give them read-only access:

sudo setfacl -m u:auditor:rx /shared\_data

**Step 5: Verify**

Check ACLs:

getfacl /shared\_data



**Task 3: Apache Virtual Hosts Setup**

**Objective**

Configure Apache to host two websites (site1.local and site2.local) with separate document roots and log files.

**Step-by-Step Commands Used**

**Step 1. Create the doc roots and example pages**

sudo mkdir -p /var/www/site1.local/public\_html /var/www/site2.local/public\_html

echo "<h1>Site 1</h1>" | sudo tee /var/www/site1.local/public\_html/index.html

echo "<h1>Site 2</h1>" | sudo tee /var/www/site2.local/public\_html/index.html

**Step 2. Set ownership so Apache can serve files**

sudo chown -R www-data:www-data /var/www/site1.local /var/www/site2.local

**Step 3. Create the site1 virtual host file (fast method using heredoc)**

sudo tee /etc/apache2/sites-available/site1.local.conf >/dev/null <<'EOF'

<VirtualHost \*:80>

ServerName site1.local

DocumentRoot /var/www/site1.local/public\_html

ErrorLog ${APACHE\_LOG\_DIR}/site1\_error.log

CustomLog ${APACHE\_LOG\_DIR}/site1\_access.log combined

<Directory /var/www/site1.local/public\_html>

AllowOverride All

Require all granted

</Directory>

</VirtualHost>

EOF

**Step 4. Create the site2 virtual host file**

sudo tee /etc/apache2/sites-available/site2.local.conf >/dev/null <<'EOF'

<VirtualHost \*:80>

ServerName site2.local

DocumentRoot /var/www/site2.local/public\_html

ErrorLog ${APACHE\_LOG\_DIR}/site2\_error.log

CustomLog ${APACHE\_LOG\_DIR}/site2\_access.log combined

<Directory /var/www/site2.local/public\_html>

AllowOverride All

Require all granted

</Directory>

</VirtualHost>

EOF

**Step 5. Disable the default site and enable your new sites**

sudo a2dissite 000-default.conf # disable the default demo site

sudo a2ensite site1.local.conf site2.local.conf

sudo systemctl reload apache2

**Step 6. Verify**

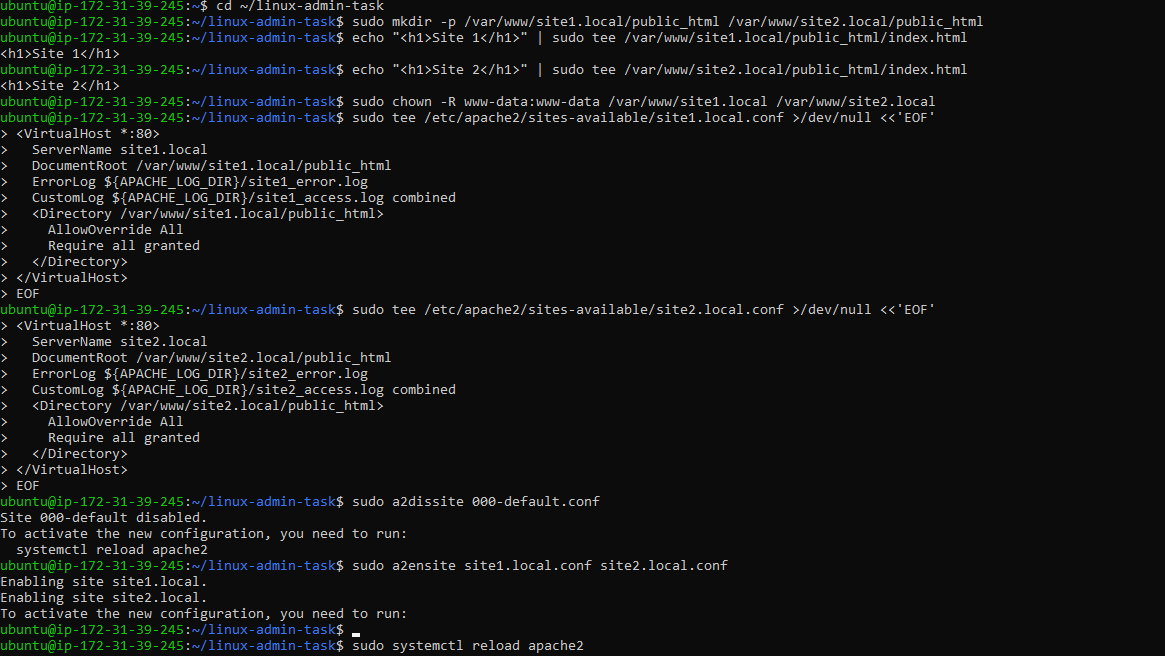
Show which virtual hosts Apache thinks are active:

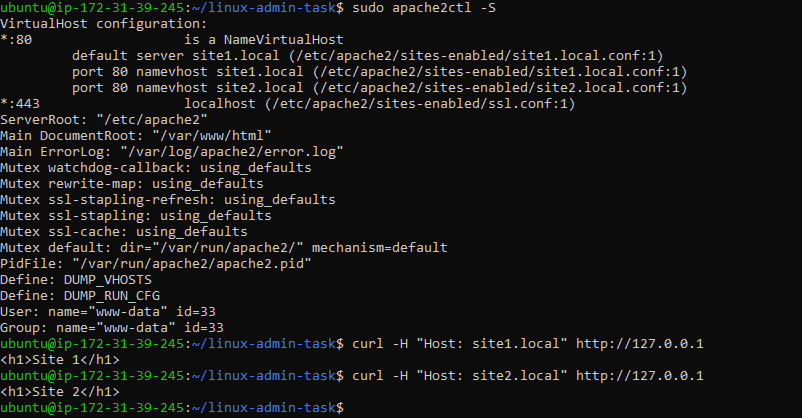
sudo apache2ctl -S

Test locally on the server (no DNS needed):

curl -H "Host: site1.local" http://127.0.0.1

curl -H "Host: site2.local" <http://127.0.0.1>





**Task 4: SSL/TLS Implementation**

**Objective**

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

**Step-by-Step Commands Used**

**Step 1 – Generate SSL cert + key for site1.local**

sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 \

-keyout /etc/ssl/private/site1.local.key \

-out /etc/ssl/certs/site1.local.crt \

-subj "/C=NG/ST=FCT/L=Abuja/O=LinuxClass/OU=IT/CN=site1.loca

**Step 2 – Create the SSL VirtualHost config**

sudo nano /etc/apache2/sites-available/site1.local-ssl.conf

Paste inside nano:

<IfModule mod\_ssl.c>

<VirtualHost \*:443>

ServerName site1.local

DocumentRoot /var/www/site1.local/public\_html

ErrorLog ${APACHE\_LOG\_DIR}/site1\_ssl\_error.log

CustomLog ${APACHE\_LOG\_DIR}/site1\_ssl\_access.log combined

SSLEngine on

SSLCertificateFile /etc/ssl/certs/site1.local.crt

SSLCertificateKeyFile /etc/ssl/private/site1.local.key

<Directory /var/www/site1.local/public\_html>

AllowOverride All

Require all granted

</Directory>

Header always set Strict-Transport-Security "max-age=31536000"

</VirtualHost>

</IfModule>

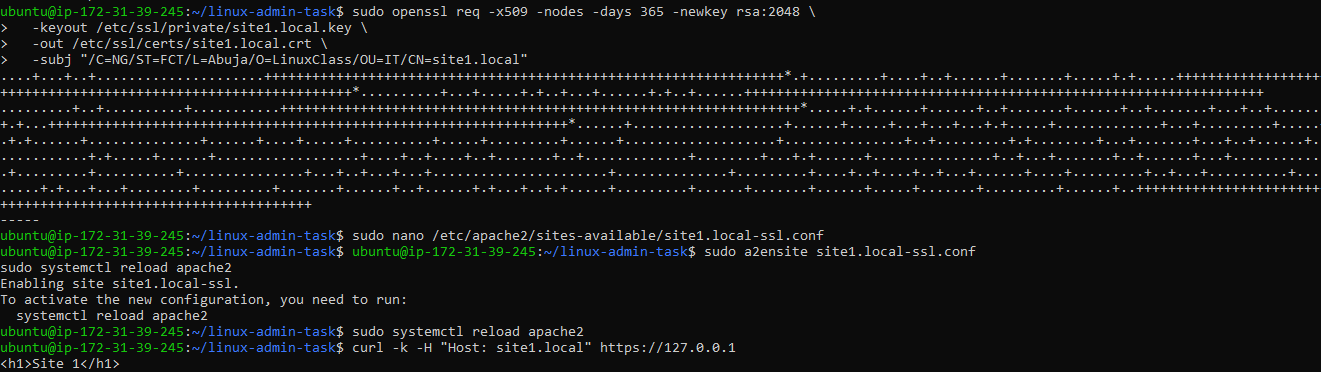
**Step 3 – Enable the SSL site & reload Apache**

sudo a2ensite site1.local-ssl.conf

sudo systemctl reload apache2

**Step 4 – Test HTTPS (self-signed, so ignore warning)**

curl -k -H "Host: site1.local" https://127.0.0.1

****

**Task 5. MySQL Remote Access & Security**

**Objective**

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

**Step-by-Step Commands Used**

**Step 1 – Allow MySQL to listen on all interfaces**

sudo sed -i 's/^\s\*bind-address\s\*=.\*/bind-address = 0.0.0.0/' /etc/mysql/mysql.conf.d/mysqld.cnf

sudo systemctl restart mysql

**Step 2 – Log into MySQL as root**

sudo mysql

**Step 3 – Inside the MySQL shell, create DB + remote user**

CREATE DATABASE IF NOT EXISTS classdb;

CREATE USER IF NOT EXISTS 'remote\_user'@'10.0.0.%' IDENTIFIED BY 'StrongPass!23';

GRANT SELECT, INSERT, UPDATE, DELETE ON classdb.\* TO 'remote\_user'@'10.0.0.%';

FLUSH PRIVILEGES;

Then exit

EXIT;

**Step 4 – Verify**

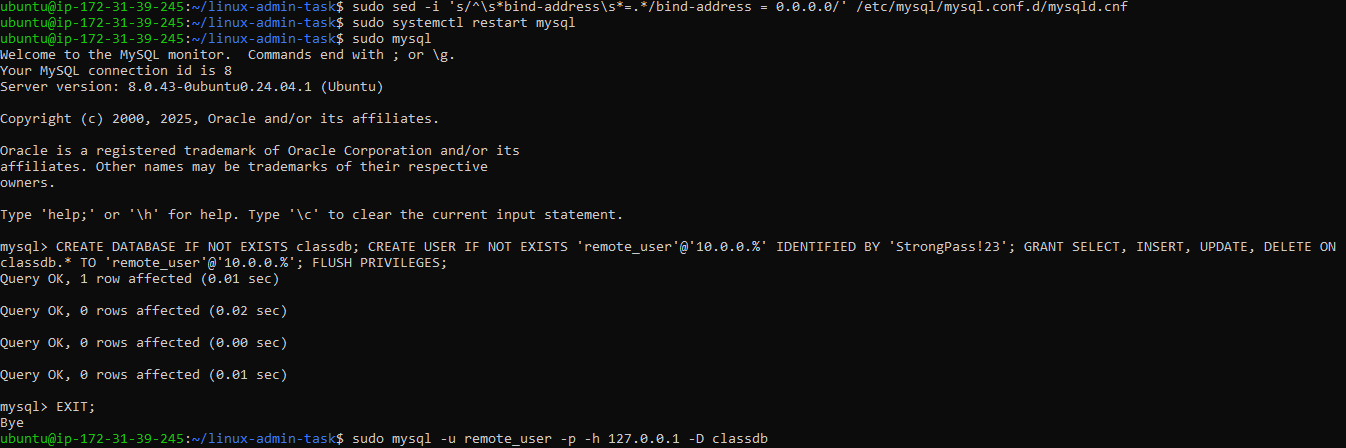
Back on Linux shell:

sudo mysql -u remote\_user -p -h 127.0.0.1 -D classdb

Enter password: StrongPass!23

At the prompt run:

SHOW DATABASES;



**Task 6. Firewall Configuration**

**Objective**

Configure `ufw` or `iptables` to allow only HTTP(80), HTTPS(443), SSH(22), and MySQL(3306) fromaspecific IP range.

**Step-by-Step Commands Used**

**Step 1 .Install ufw if missing**

sudo apt update

sudo apt install -y ufw

**Step 2. Detect your current SSH client**

Run this exactly (it tries a couple of methods and prints the detected value):

MY\_IP=$(echo ${SSH\_CLIENT:-$(who -m | awk '{print $5}' | tr -d '()')} | awk '{print $1}'); \

echo "Detected SSH client IP: $MY\_IP"

**Step 3. Set a safety rollback timer**

sudo bash -c '(sleep 120 && ufw disable) &' && echo "Rollback timer set for 120s (to cancel: sudo pkill -f \"sleep 120 && ufw disable\")"

**Step 4. Set default policy (deny incoming, allow outgoing)**

sudo ufw default deny incoming

sudo ufw default allow outgoing

**Step 5. Allow SSH from your current IP only**

sudo ufw allow from $MY\_IP to any port 22 proto tcp

**Step 6. Allow HTTP (80) and HTTPS (443) from anywhere**

sudo ufw allow 80/tcp

sudo ufw allow 443/tcp

**Step 7. Allow MySQL (3306) only from IP**

sudo ufw allow from $MY\_IP/32 to any port 3306 proto tcp

**Step 8. Enable UFW**

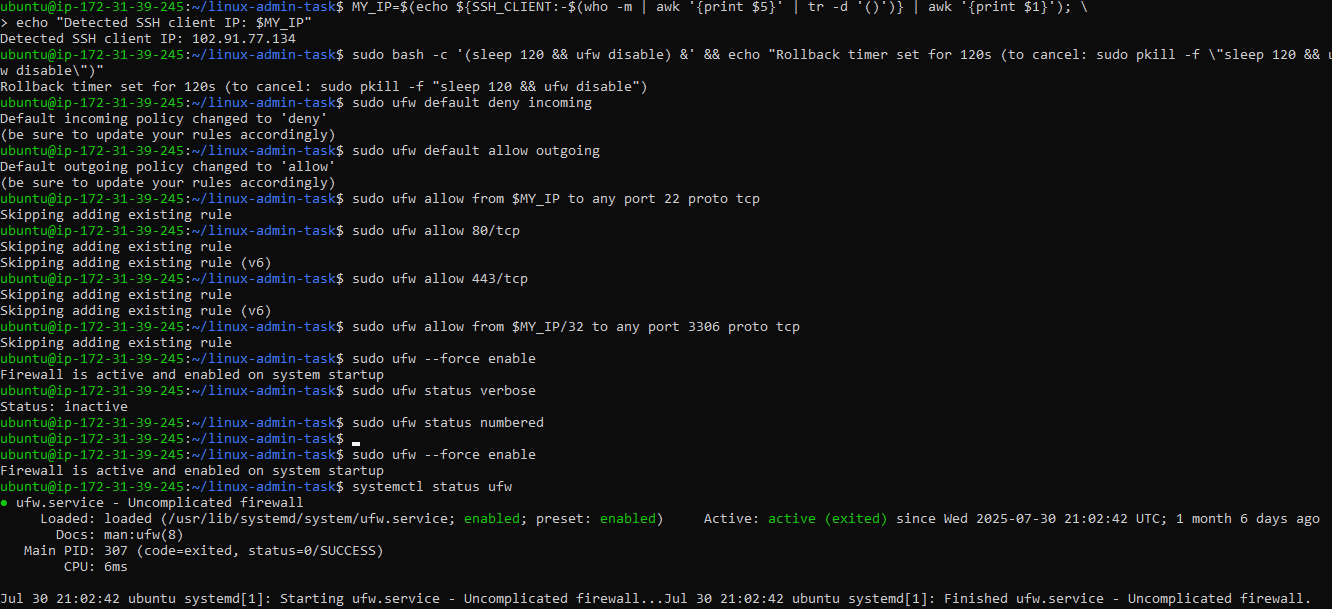
sudo ufw --force enable

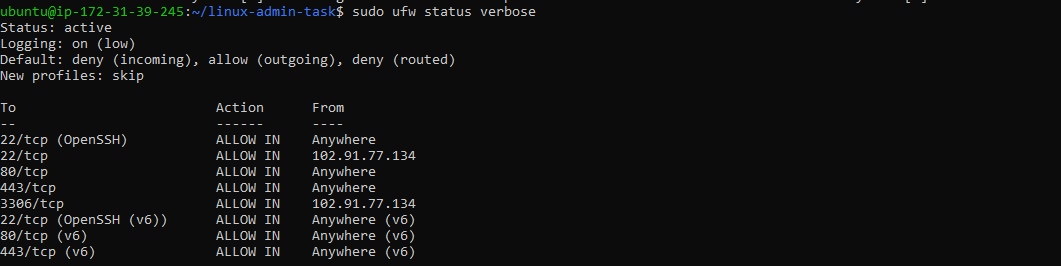
**Step 9. Verify rules**

sudo ufw status verbose

# or for numbered listing:

sudo ufw status numbered

****



**Task 7. System Monitoring Script**

**Objective**

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

**Step-by-Step Commands Used**

**Step 1: Create the monitoring script**

Open a terminal and run the following command to create the script

/usr/local/sbin/sys\_health.sh:

sudo tee /usr/local/sbin/sys\_health.sh >/dev/null <<'EOF'

#!/usr/bin/env bash

set -euo pipefail

LOG=/var/log/sys\_health.log

{

echo "==== $(date) ===="

echo "-- CPU --"

top -bn1 | head -n 5

echo "-- MEMORY --"

free -m

echo "-- DISK --"

df -h

echo

} >> "$LOG"

EOF

**Step 2: Make the script executable**

sudo chmod +x /usr/local/sbin/sys\_health.sh

**Step 3: Prepare the log file**

sudo touch /var/log/sys\_health.log

sudo chmod 644 /var/log/sys\_health.log

**Step 4: Set up the cron job**

Add a cron job that runs every 5 minutes:

echo "\*/5 \* \* \* \* root /usr/local/sbin/sys\_health.sh" | sudo tee /etc/cron.d/s

**Step 5: Restart cron service**

sudo systemctl restart cron

**Step 6. Check if cron service is running:**

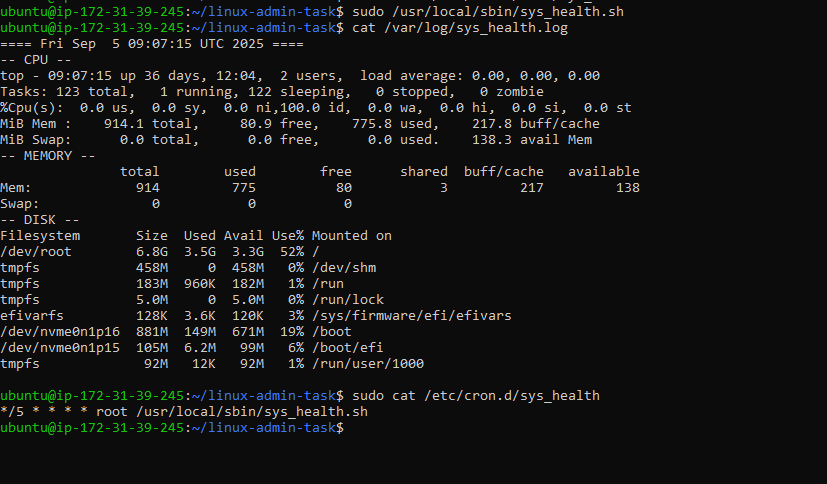
sudo systemctl status cron

**Step 7. Test the script manually before cron:**

sudo /usr/local/sbin/sys\_health.sh

cat /var/log/sys\_health.log





**Task 8. Log Rotation Setup**

**Objective**

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

**Step-by-Step Commands Used**

**Step 1: Configure BIND options**

sudo tee /etc/bind/named.conf.options >/dev/null <<'EOF'

options {

directory "/var/cache/bind";

recursion yes;

allow-query { any; };

listen-on { any; };

forwarders { 8.8.8.8; 1.1.1.1; };

dnssec-validation auto;

auth-nxdomain no;

listen-on-v6 { any; };

};

EOF

**Step 2: Create a test log file**

sudo mkdir -p /var/log/myapp && sudo bash -c 'echo test >> /var/log/myapp/app.log'

**Step 3: Run logrotate in dry-run mode (just simulates, won’t actually rotate yet)**

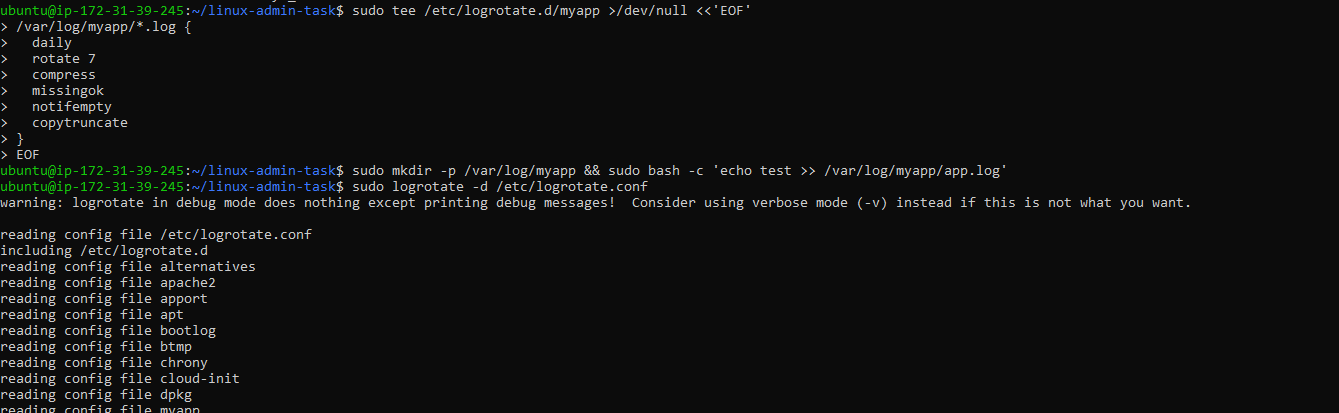
sudo logrotate -d /etc/logrotate.conf

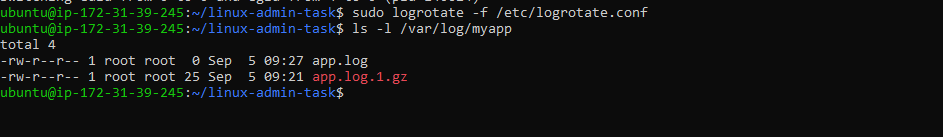
**Step 4: Force log rotation (not dry-run)**

sudo logrotate -f /etc/logrotate.conf

**🔹 Step 5: Check the log folder**

ls -l /var/log/myapp

****



**Task 9. DNS Server Setup**

**Objective**

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

**Step-by-Step Commands Used**

**Step 1: Configure BIND options**

sudo tee /etc/bind/named.conf.options >/dev/null <<'EOF'

options {

directory "/var/cache/bind";

recursion yes;

allow-query { any; };

listen-on { any; };

forwarders { 8.8.8.8; 1.1.1.1; };

dnssec-validation auto;

auth-nxdomain no;

listen-on-v6 { any; };

};

EOF

**Step 2: Create custom zone file**

sudo mkdir -p /etc/bind/zones

SERVER\_IP=$(hostname -I | awk '{print $1}')

sudo tee /etc/bind/zones/db.myuniversity.local >/dev/null <<EOF

$TTL 604800

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

1 ; Serial

604800 ; Refresh

86400 ; Retry

2419200 ; Expire

604800 ) ; Negative Cache TTL

;

@ IN NS ns1.myuniversity.local.

ns1 IN A ${SERVER\_IP}

web IN A ${SERVER\_IP}

site1 IN A ${SERVER\_IP}

site2 IN A ${SERVER\_IP}

EOF

**Step 3: Add zone to BIND local config**

sudo tee /etc/bind/named.conf.local >/dev/null <<'EOF'

zone "myuniversity.local" {

type master;

file "/etc/bind/zones/db.myuniversity.local";

};

EOF

**Step 4: Restart BIND service**

sudo systemctl restart bind9

**Step 5: Make server use its own DNS resolver**

echo -e "[Resolve]\nDNS=127.0.0.1\nDomains=~." | sudo tee /etc/systemd/resolved.conf

sudo systemctl restart systemd-resolved

**Step 6: Check BIND syntax**

Run:

sudo named-checkconf

If nothing prints, config is OK. If there’s an error, it will show.

**Step 7: Check your zone file syntax**

sudo named-checkzone myuniversity.local /etc/bind/zones/db.myuniversity.local

**Step 8: Restart BIND again**

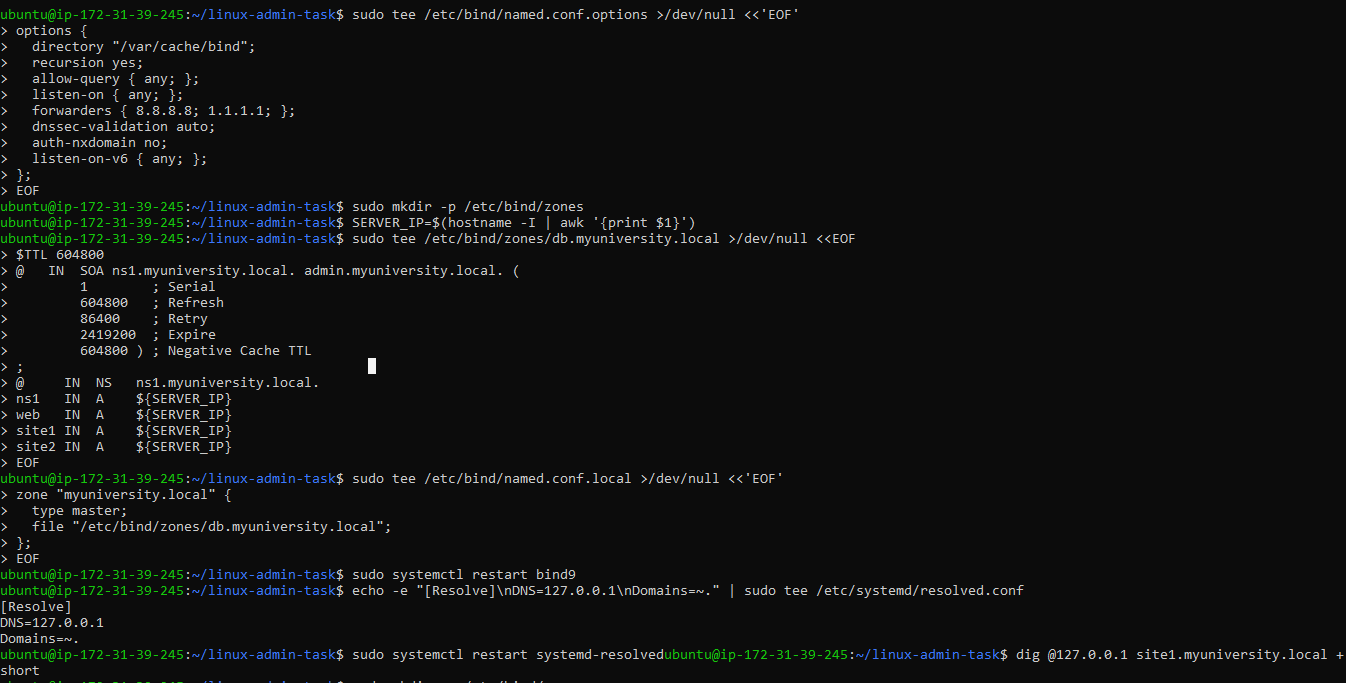
sudo systemctl restart bind9

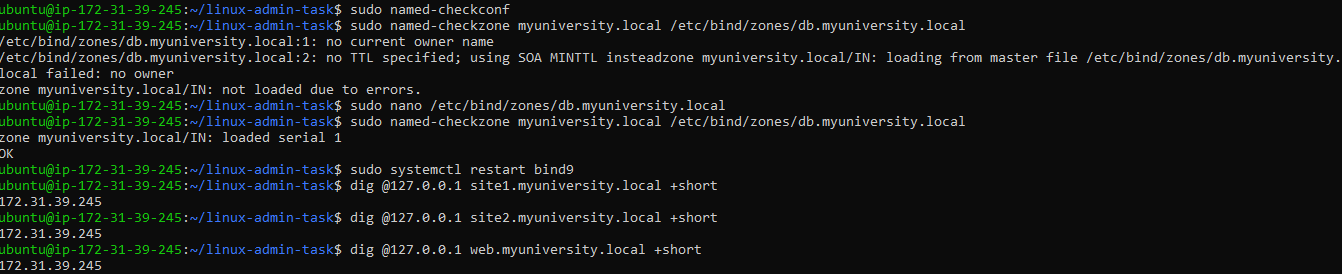
**Step 9: Test with dig again**

dig @127.0.0.1 site1.myuniversity.local +short

dig @127.0.0.1 site2.myuniversity.local +short

dig @127.0.0.1 web.myuniversity.local +short

****



**Task 10. SSH Key Authentication + Hardening**

**Objective**

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

**Step-by-Step Commands Used**

**Step 1 . Create a new SSH key pair on your laptop (PowerShell)**

ssh-keygen -t ed25519 -f $env:USERPROFILE\.ssh\aws\_ed -N ""

**Step 2. Copy your public key to the server . Manual (works everywhere — PowerShell / Git Bash)**

This uploads the pub key, appends to authorized\_keys, and sets correct permissions:

# copy the pub file to server temp location

scp -i "C:\Users\Timi\.ssh\Linux\_key.pem" $env:USERPROFILE\.ssh\aws\_ed.pub ubuntu@16.171.136.71:~/.ssh/temp\_pub

# on laptop, run:

ssh -i "C:\Users\Timi\.ssh\Linux\_key.pem" ubuntu@16.171.136.71 "mkdir -p ~/.ssh && cat ~/.ssh/temp\_pub >> ~/.ssh/authorized\_keys && rm ~/.ssh/temp\_pub && chmod 700 ~/.ssh && chmod 600 ~/.ssh/authorized\_keys"

**Step 3. Verify permissions on the server (run in the EC2 shell)**

verify .ssh exists and ownership/permissions are correct

ls -ld ~/.ssh

ls -l ~/.ssh/authorized\_keys

fix if needed (on server)

chmod 700 ~/.ssh

chmod 600 ~/.ssh/authorized\_keys

chown -R $(whoami):$(whoami) ~/.ssh

**Step 4. Test the new key (open a NEW terminal on your laptop — keep the original session open!)**

ssh -i "C:\Users\Timi\.ssh\aws\_ed" [ubuntu@16.171.136.71](mailto:ubuntu@16.171.136.71)

**Step 5. Backup sshd\_config then harden it (on the server)**

1. Backup:

sudo cp /etc/ssh/sshd\_config /etc/ssh/sshd\_config.bak.$(date +%F-%H%M%S)

1. Edit

sudo nano /etc/ssh/sshd\_config

Find and set (uncomment/modify) these lines:

PubkeyAuthentication yes

PasswordAuthentication no

ChallengeResponseAuthentication no

PermitRootLogin no

# (Optional but recommended) PermitEmptyPasswords no

# Ensure AuthorizedKeysFile is set (default is fine):

AuthorizedKeysFile .ssh/authorized\_keys

1. Test the sshd config syntax:

sudo sshd -t

If that prints nothing, the file is syntactically OK.

1. Reload SSH (do not close your current session):

sudo systemctl reload ssh

**Step 6 Final verification**

1. From laptop open a new terminal and try (power shell)

ssh -i "C:\Users\Timi\.ssh\aws\_ed" [ubuntu@16.171.136.71](mailto:ubuntu@16.171.136.71)

1. the login succeeds— try also an interactive test: open another terminal and attempt password login (should fail):

ssh -o PreferredAuthentications=password -o PubkeyAuthentication=no ubuntu@1

1. If the new-key login fails and you’re still in the original session: revert quickly:

# in your original working EC2 SSH session:

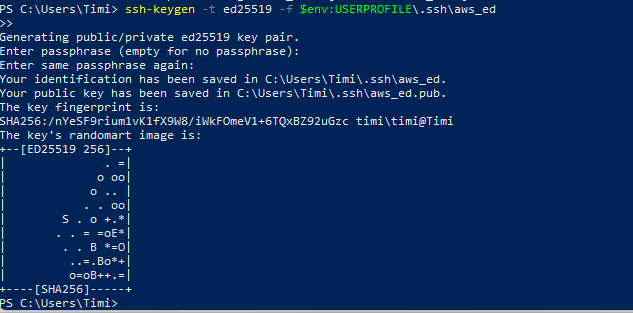
sudo cp /etc/ssh/sshd\_config.bak.<timestamp> /etc/ssh/sshd\_config

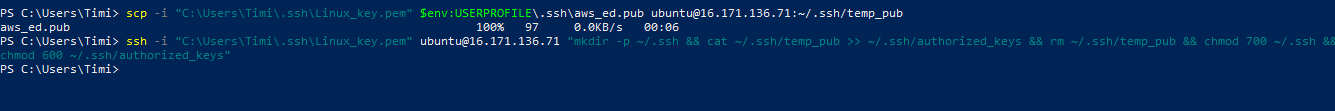
sudo systemctl reload ssh

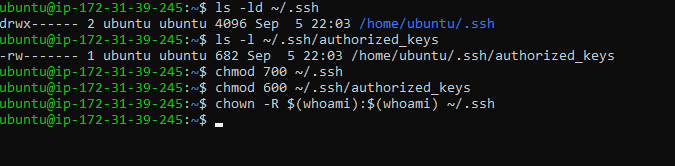
Or re-enable password auth immediately:

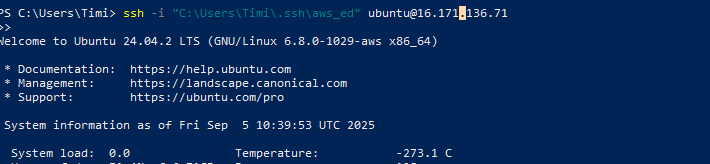
sudo sed -i 's/^PasswordAuthentication no/PasswordAuthentication yes/' /etc/ssh/sshd\_config

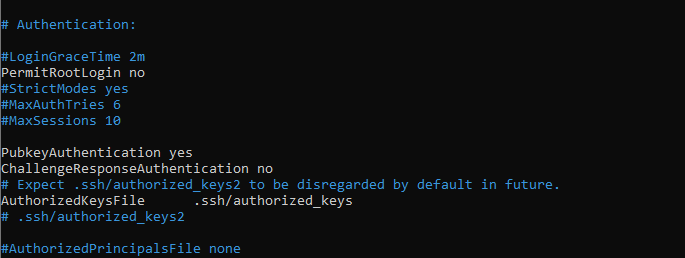
sudo systemctl reload ssh

****

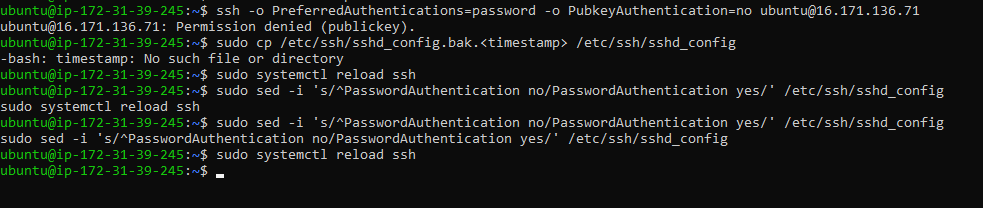












**Task 11. Systemd Service Creation**

**Objective**

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

**Step-by-Step Commands Used**

**Step 1. Create the script**

sudo tee /usr/local/bin/hello.sh >/dev/null <<'EOF'

#!/usr/bin/env bash

while true; do

echo "Hello from systemd at $(date)" >> /var/log/hello.log

sleep 60

done

EOF

**Step 2. Make it executable**

sudo chmod +x /usr/local/bin/hello.sh

**Step 3. Create the systemd unit file**

sudo tee /etc/systemd/system/hello.service >/dev/null <<'EOF'

[Unit]

Description=Hello Demo Service

After=network.target

[Service]

ExecStart=/usr/local/bin/hello.sh

Restart=always

User=root

[Install]

WantedBy=multi-user.target

EOF

**Step 4. Reload systemd and enable service**

sudo systemctl daemon-reload

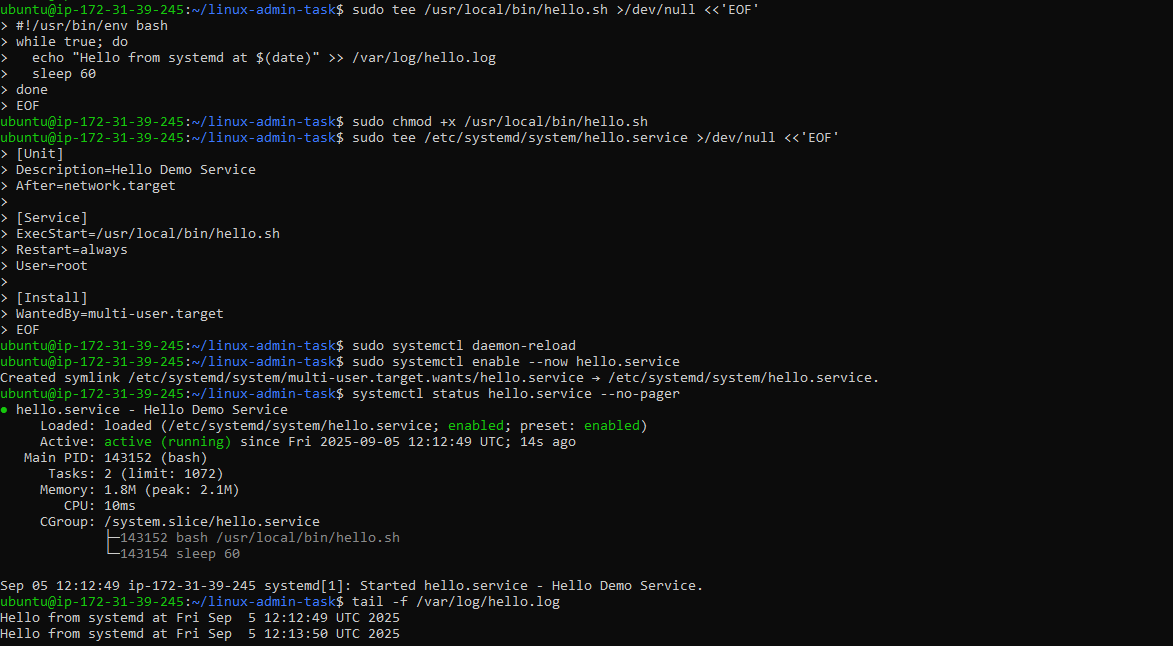
sudo systemctl enable --now hello.service

**Step 5. Check service status**

systemctl status hello.service --no-pager

**Step 6. Verify**

tail -f /var/log/hello.log



**Task 12: Disk Partitioning & Mounting**

**Objective**

Create a newpartition, format as ext4, mount it permanently using `/etc/fstab`, and test reboot persistence.

**Step-by-Step Commands Used**

**Step 1. Identify your extra disk**

Check if xvdf exists:

**lsblk**

**Step 2. Partition the disk**

Run:

sudo fdisk /dev/ nvme1n1p1

Inside fdisk, do the following:

n → new partition

p → primary

1 → partition number

*Enter* →

*Enter* →

w → write changes

**Step 3. Format the partition as ext4**

sudo mkfs.ext4 /dev/nvme1n1p1

**Step 4. Create a mount point**

sudo mkdir -p /mnt/data

**Step 5. Add entry to /etc/fstab**

echo '/dev/nvme1n1p1 /mnt/data ext4 defaults,nofail 0 2' | sudo tee -a /etc/fstab

**Step 6. Mount all partitions**

sudo mount -a

**7. Verify**

df -h | grep /mnt/data

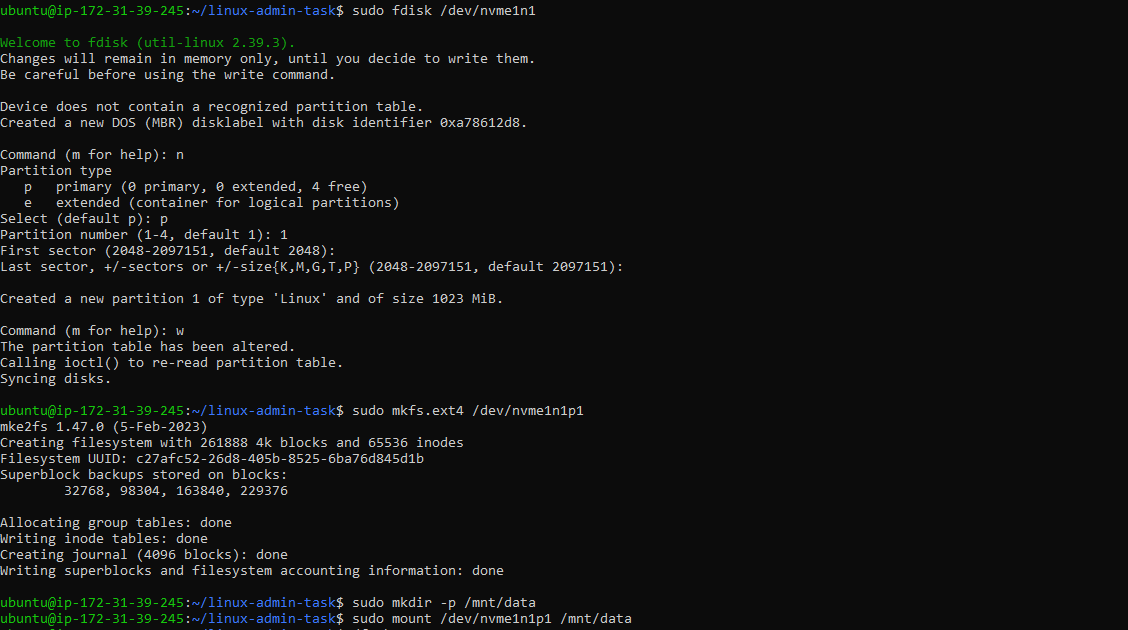
**8. Test reboot persistence**

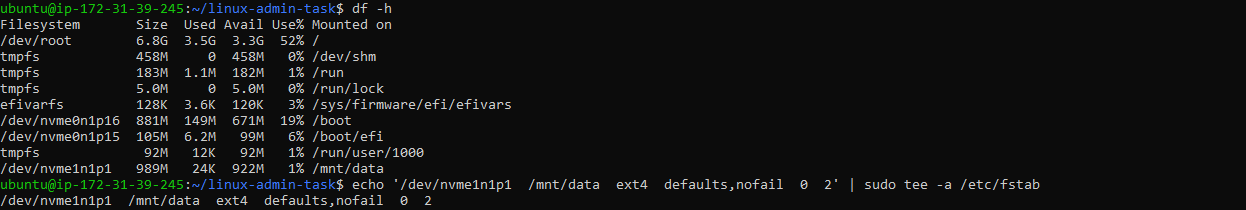
Reboot the instance:

sudo reboot

Then after logging back in, run:

df -h | grep /mnt/data





**Task 13: Postfix Mail Server (Local Only)**

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

**Step-by-Step Commands Used**

**Step 1. Install Postfix & Mail tools**

Run:

sudo apt update

sudo apt install -y postfix mailutils

**Step 2. Configure Postfix manually (in case dialog doesn’t appear)**

sudo postconf -e 'mydestination = $myhostname, localhost.localdomain, localhost'

sudo postconf -e 'inet\_interfaces = loopback-only'

sudo systemctl restart postfix

**Step 3. Create two local users**

sudo adduser --disabled-password --gecos "" usera || true

sudo adduser --disabled-password --gecos "" userb || true

**Step 4.**

**Send a reply as usera**

echo "Hello userb from usera" | sudo -u usera mail -s "Local Test" userb

**Send reply as userb**

echo "Hello back usera from userb" | sudo -u userb mail -s "Reply Test" usera

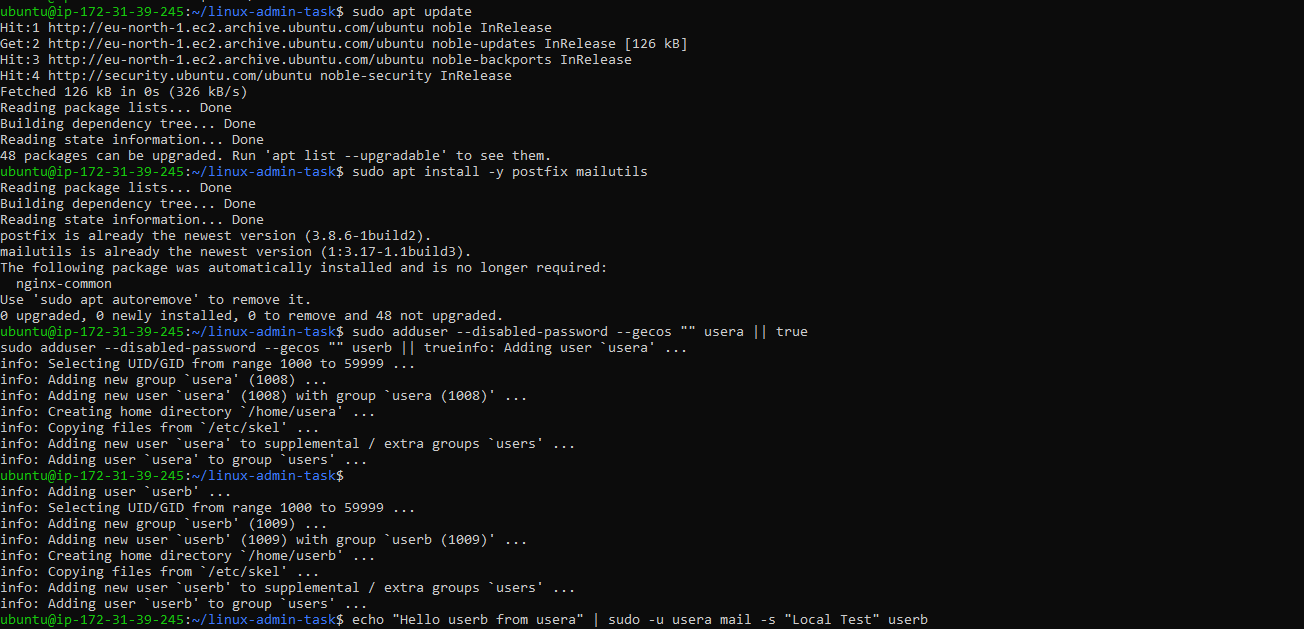
**Step 5.**

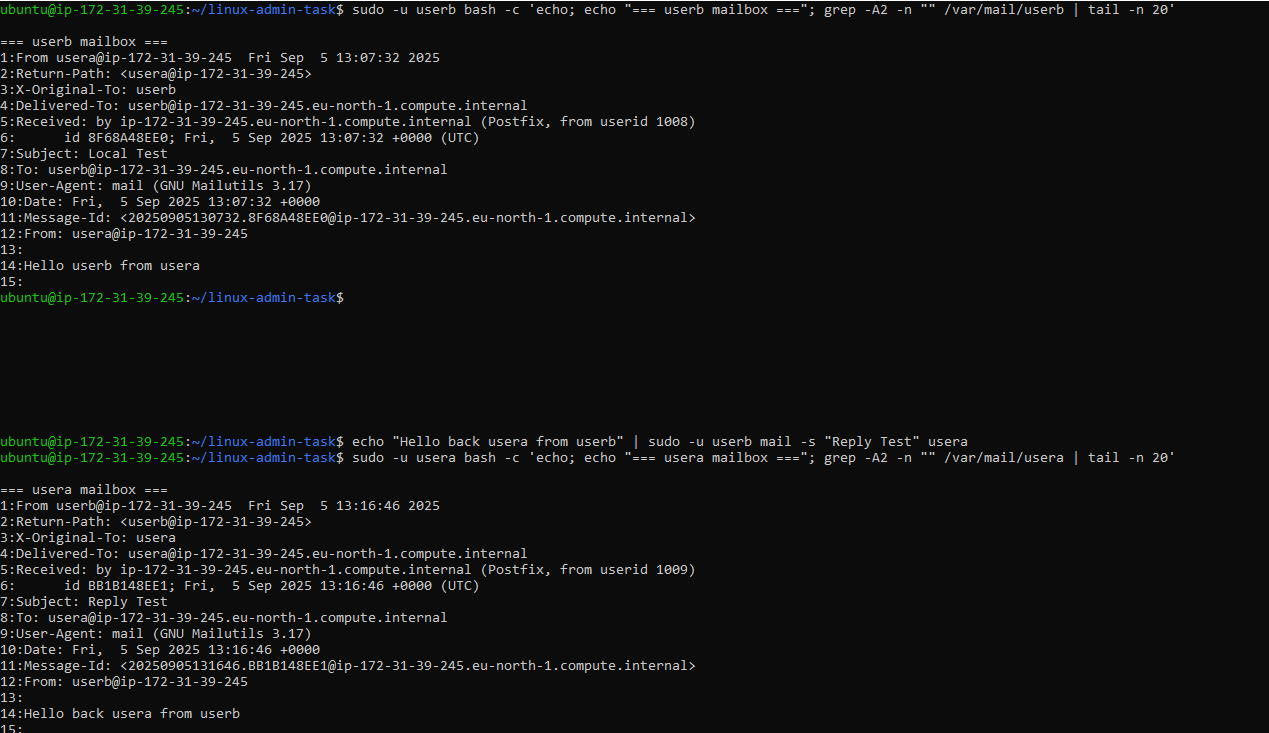
**Check mailbox of userb**

sudo -u userb bash -c 'echo; echo "=== userb mailbox ==="; sudo grep -A2 -n "" /var/mail/userb | tail -n 20'

**Check mailbox of usera**

sudo -u usera bash -c 'echo; echo "=== usera mailbox ==="; grep -A2 -n "" /var/mail/usera | tail -n 20'



****

**Task 14. Backup & Restore Project**

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

**Step-by-Step Commands Used**

**Step 1. Create script with heredoc (paste this whole block)**

sudo mkdir -p /backup

sudo tee /usr/local/sbin/backup\_web.sh >/dev/null <<'EOF'

#!/usr/bin/env bash

set -euo pipefail

SRC="/var/www/html"

DST="/backup"

TS=$(date +%F-%H%M%S)

ARCHIVE="$DST/html-$TS.tar.gz"

mkdir -p "$DST"

tar -czf "$ARCHIVE" "$SRC"

echo "Created $ARCHIVE"

EOF

sudo chmod +x /usr/local/sbin/backup\_web.sh

**Step 2. Create a small test file so you can verify the backup contains something**

sudo mkdir -p /var/www/html

echo "Hello from Task 14 - $(date)" | sudo tee /var/www/html/test-backup.tx

**Step 3. Run the backup script**

sudo /usr/local/sbin/backup\_web.sh

ls -lh /backup

**Step 4 . Restore the latest backup to a temporary folder and verify contents**

sudo mkdir -p /tmp/restore

LATEST=$(ls -1t /backup/html-\*.tar.gz | head -n1)

echo "Restoring: $LATEST"

sudo tar -xzf "$LATEST" -C /tmp/restore

ls -l /tmp/restore/var/www/html

# Optionally view the test file:

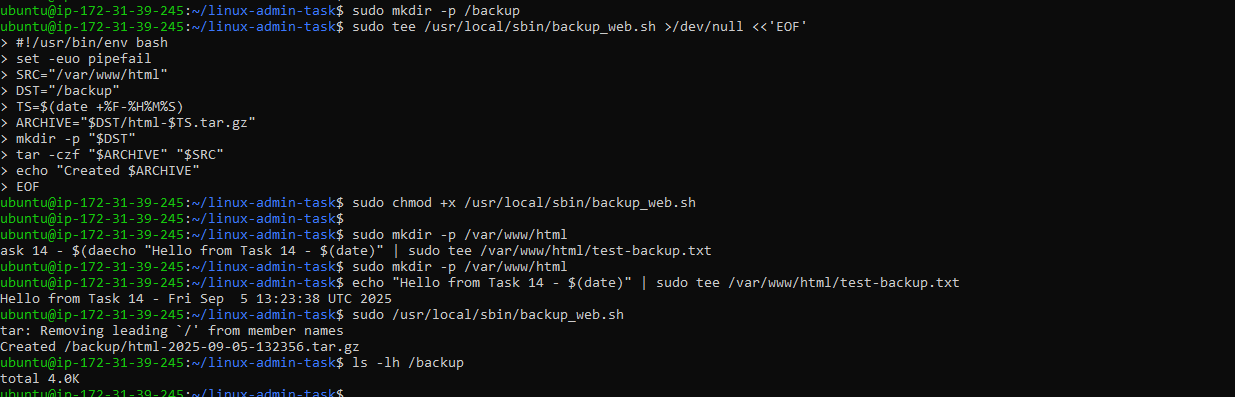
sudo cat /tmp/restore/var/www/html/test-backup.txt

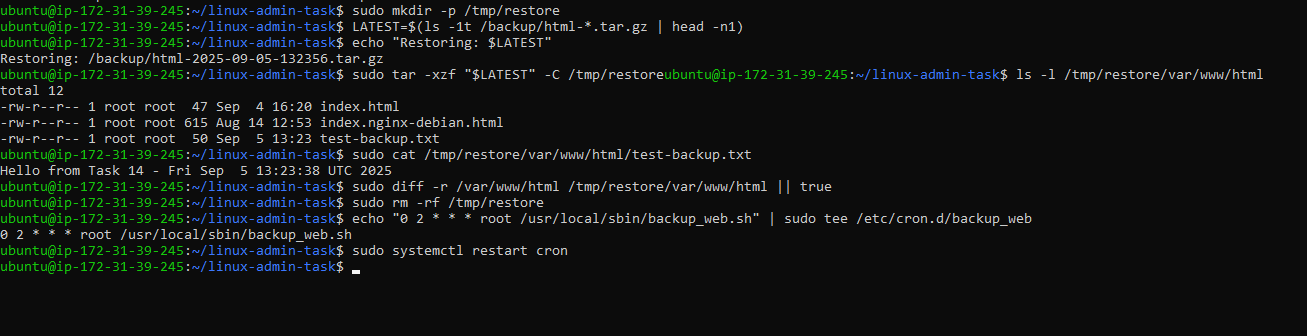
**Step 5. Compare original and restored**

sudo diff -r /var/www/html /tmp/restore/var/www/html || true

**Step 6. Remove test restore**

sudo rm -rf /tmp/restore

****



**Task 15. Containerization Challenge**

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

**Step-by-Step Commands Used**

**Step 1. Install Docker (if not already installed)**

**# Update system**

sudo apt update && sudo apt install -y ca-certificates curl gnupg

**# Add Docker’s official GPG key**

sudo install -m 0755 -d /etc/apt/keyrings

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg

sudo chmod a+r /etc/apt/keyrings/docker.gpg

**# Set up the repo**

echo \

"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu \

$(lsb\_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

**# Install Docker**

sudo apt update

sudo apt install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin

**Verify Docker installed:**

docker –version

**Step 2. Pull the Nginx image**

sudo docker pull nginx:latest

**Step 3. Run the Nginx container on port 8080**

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

**Step 4. Verify container is running**

sudo docker ps

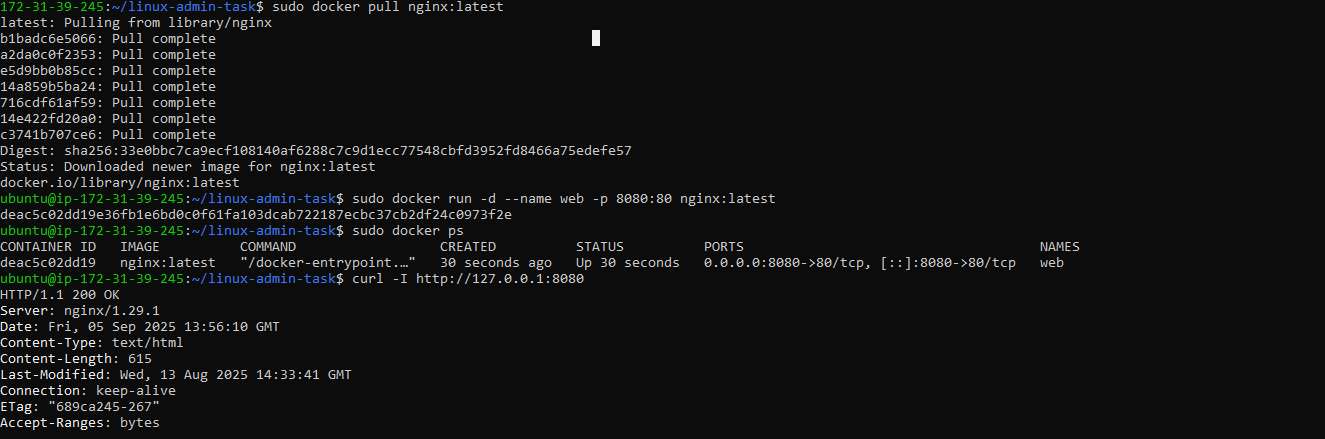
**Step 5 Test locally (inside the server)**

curl -I <http://127.0.0.1:8080>

**Step 6. Test externally (from your laptop browser)**

Open:

http://<EC2-PUBLIC-IP>:8080

****

