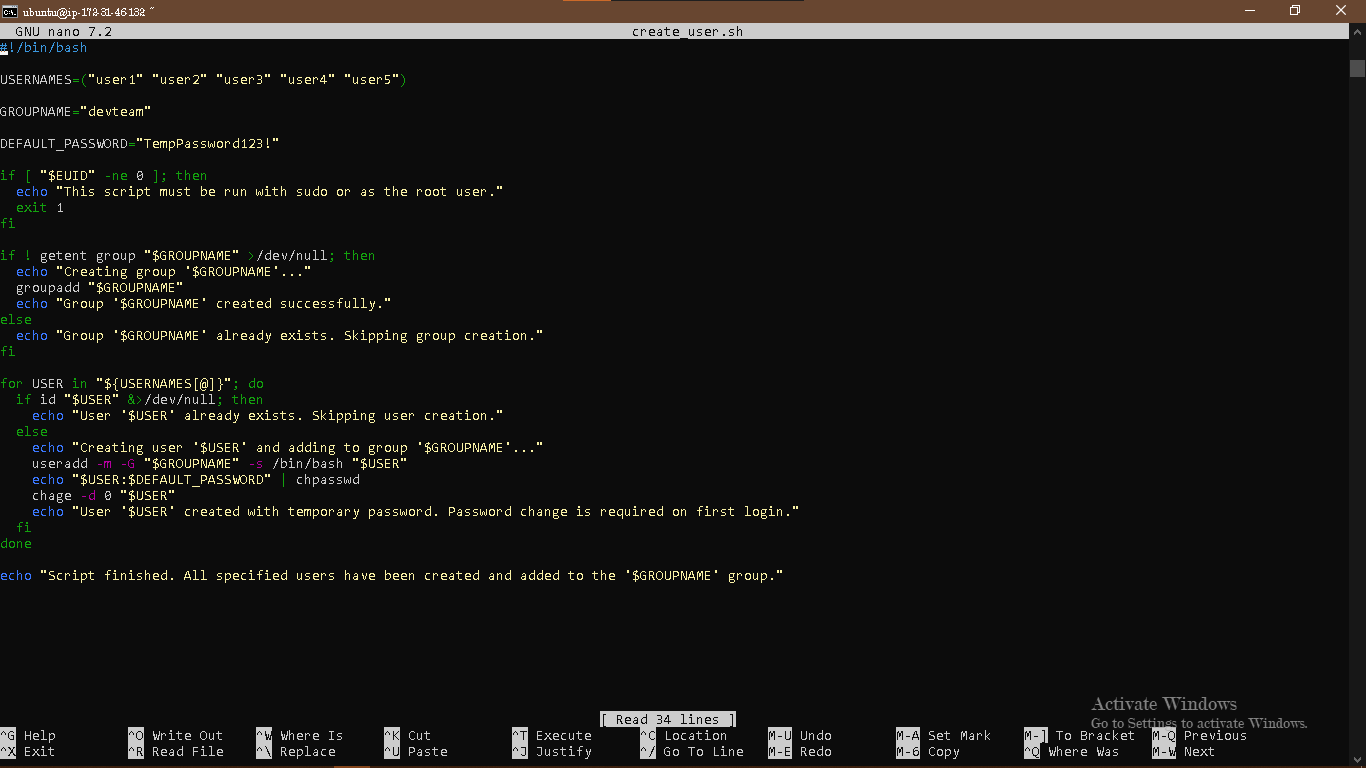
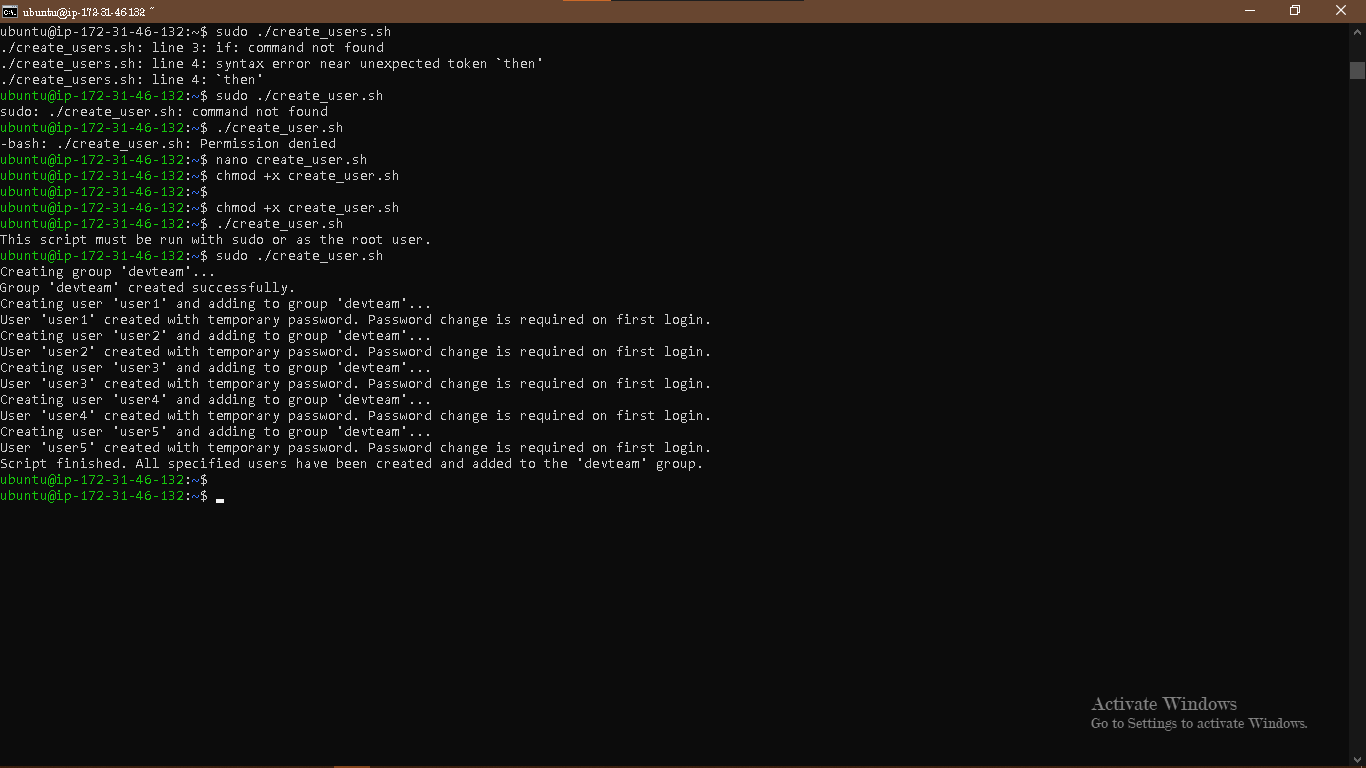
Linux Admininstration Works Scripts and Solution

1. . User &GroupManagementAutomation
   1. Script: 

1. Screenshot of Solution :
2. . File Permissions&ACLs Project
3. Script

: #!/bin/bash

SHARED\_DIR="/shared\_data"

GROUP\_NAME="devteam"

GUEST\_USER="guestuser"

if [ "$EUID" -ne 0 ]; then

  echo "This script must be run with sudo or as the root user."

  exit 1

fi

echo "--- Starting Directory and Permissions Setup ---"

if [ ! -d "$SHARED\_DIR" ]; then

  echo "Creating shared directory: $SHARED\_DIR"

  mkdir -p "$SHARED\_DIR"

else

  echo "Shared directory $SHARED\_DIR already exists."

fi

if ! getent group "$GROUP\_NAME" >/dev/null; then

  echo "Creating group '$GROUP\_NAME'..."

  groupadd "$GROUP\_NAME"

  echo "Group '$GROUP\_NAME' created successfully."

else

  echo "Group '$GROUP\_NAME' already exists."

fi

echo "Changing group ownership of $SHARED\_DIR to $GROUP\_NAME..."

chgrp "$GROUP\_NAME" "$SHARED\_DIR"

echo "Setting permissions for $SHARED\_DIR to rwx for group members..."

chmod 2770 "$SHARED\_DIR"

echo "The directory permissions are now set. "

echo "Members of '$GROUP\_NAME' can read/write but not delete each other's files."

echo ""

echo "--- Starting ACL Setup ---"

if ! id "$GUEST\_USER" &>/dev/null; then

  echo "Creating guest user '$GUEST\_USER' for ACL demonstration..."

  useradd -m "$GUEST\_USER"

  echo "User '$GUEST\_USER' created. Now setting up ACL."

else

  echo "User '$GUEST\_USER' already exists. Skipping user creation."

fi

echo "Granting read-only access to user '$GUEST\_USER' on $SHARED\_DIR..."

setfacl -m u:"$GUEST\_USER":r-x "$SHARED\_DIR"

echo "ACL has been set. The user '$GUEST\_USER' can now read and list files in $SHARED\_DIR."

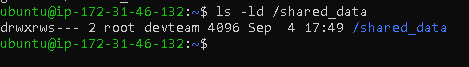
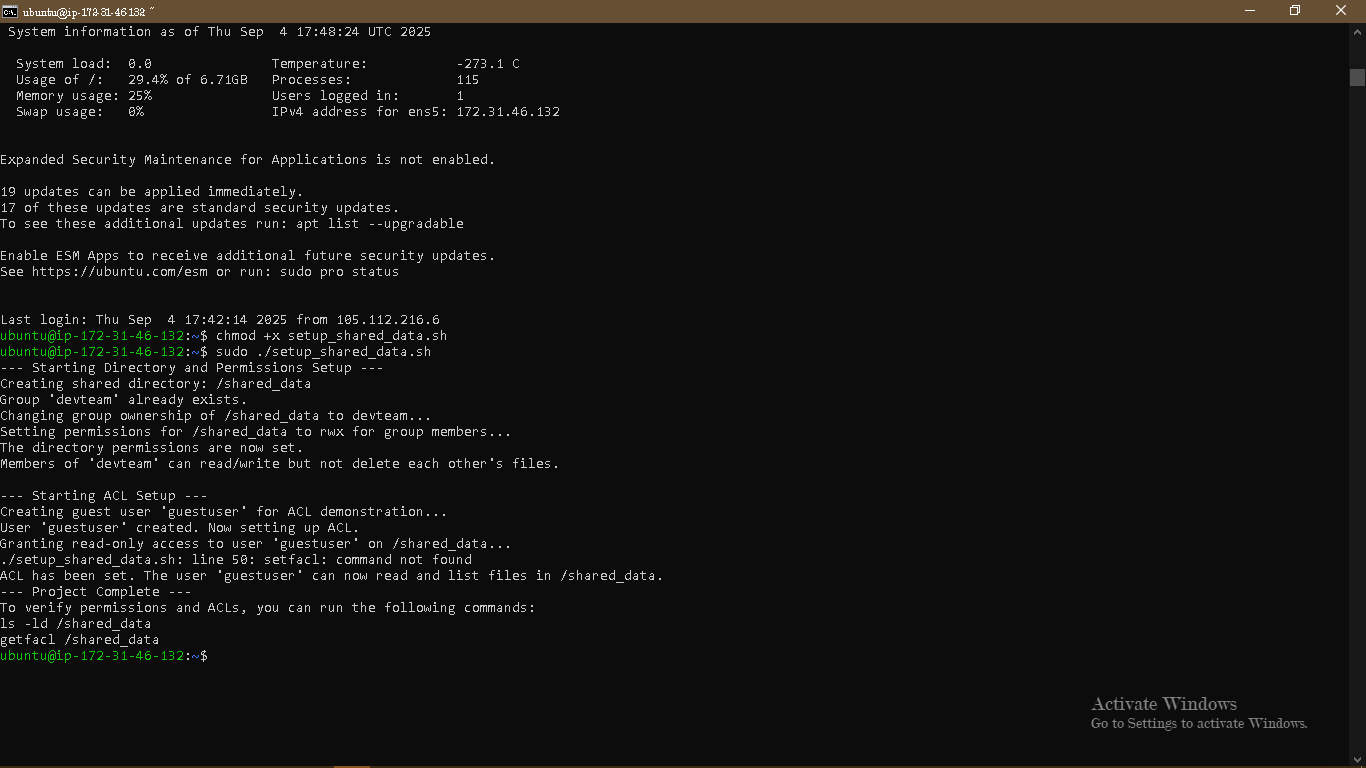
echo ""

echo "--- Project Complete ---"

echo "To verify permissions and ACLs, you can run the following commands:"

echo "ls -ld $SHARED\_DIR"

echo "getfacl $SHARED\_DIR"

Solution: 

3.  ApacheVirtualHosts Setup

Script: #!/bin/bash

SITE1\_NAME="site1.local"

SITE2\_NAME="site2.local"

SITE\_ROOT="/var/www"

APACHE\_CONF\_DIR="/etc/apache2/sites-available"

if [[ "$EUID" -ne 0 ]]; then

  echo "This script must be run with sudo or as the root user."

  exit 1

fi

if ! command -v apache2 &> /dev/null; then

  echo "Apache is not installed. Please run the following command to install it:"

  echo "sudo apt-get update && sudo apt-get install apache2"

  exit 1

fi

echo "--- Starting Apache Virtual Host Setup ---"

mkdir -p "$SITE\_ROOT/$SITE1\_NAME/public\_html"

mkdir -p "$SITE\_ROOT/$SITE1\_NAME/logs"

mkdir -p "$SITE\_ROOT/$SITE2\_NAME/public\_html"

mkdir -p "$SITE\_ROOT/$SITE2\_NAME/logs"

echo "<html><body><h1>Welcome to $SITE1\_NAME</h1></body></html>" > "$SITE\_ROOT/$SITE1\_NAME/public\_html/index.html"

echo "<html><body><h1>Welcome to $SITE2\_NAME</h1></body></html>" > "$SITE\_ROOT/$SITE2\_NAME/public\_html/index.html"

cat > "$APACHE\_CONF\_DIR/$SITE1\_NAME.conf" << EOF

<VirtualHost \*:80>

    ServerName $SITE1\_NAME

    ServerAlias www.$SITE1\_NAME

    DocumentRoot $SITE\_ROOT/$SITE1\_NAME/public\_html

    ErrorLog $SITE\_ROOT/$SITE1\_NAME/logs/error.log

    CustomLog $SITE\_ROOT/$SITE1\_NAME/logs/access.log combined

    <Directory "$SITE\_ROOT/$SITE1\_NAME/public\_html">

        Options Indexes FollowSymLinks

        AllowOverride All

        Require all granted

    </Directory>

</VirtualHost>

EOF

cat > "$APACHE\_CONF\_DIR/$SITE2\_NAME.conf" << EOF

<VirtualHost \*:80>

    ServerName $SITE2\_NAME

    ServerAlias www.$SITE2\_NAME

    DocumentRoot $SITE\_ROOT/$SITE2\_NAME/public\_html

    ErrorLog $SITE\_ROOT/$SITE2\_NAME/logs/error.log

    CustomLog $SITE\_ROOT/$SITE2\_NAME/logs/access.log combined

    <Directory "$SITE\_ROOT/$SITE2\_NAME/public\_html">

        Options Indexes FollowSymLinks

        AllowOverride All

        Require all granted

    </Directory>

</VirtualHost>

EOF

a2ensite "$SITE1\_NAME.conf"

a2ensite "$SITE2\_NAME.conf"

a2dissite 000-default.conf

apache2ctl configtest

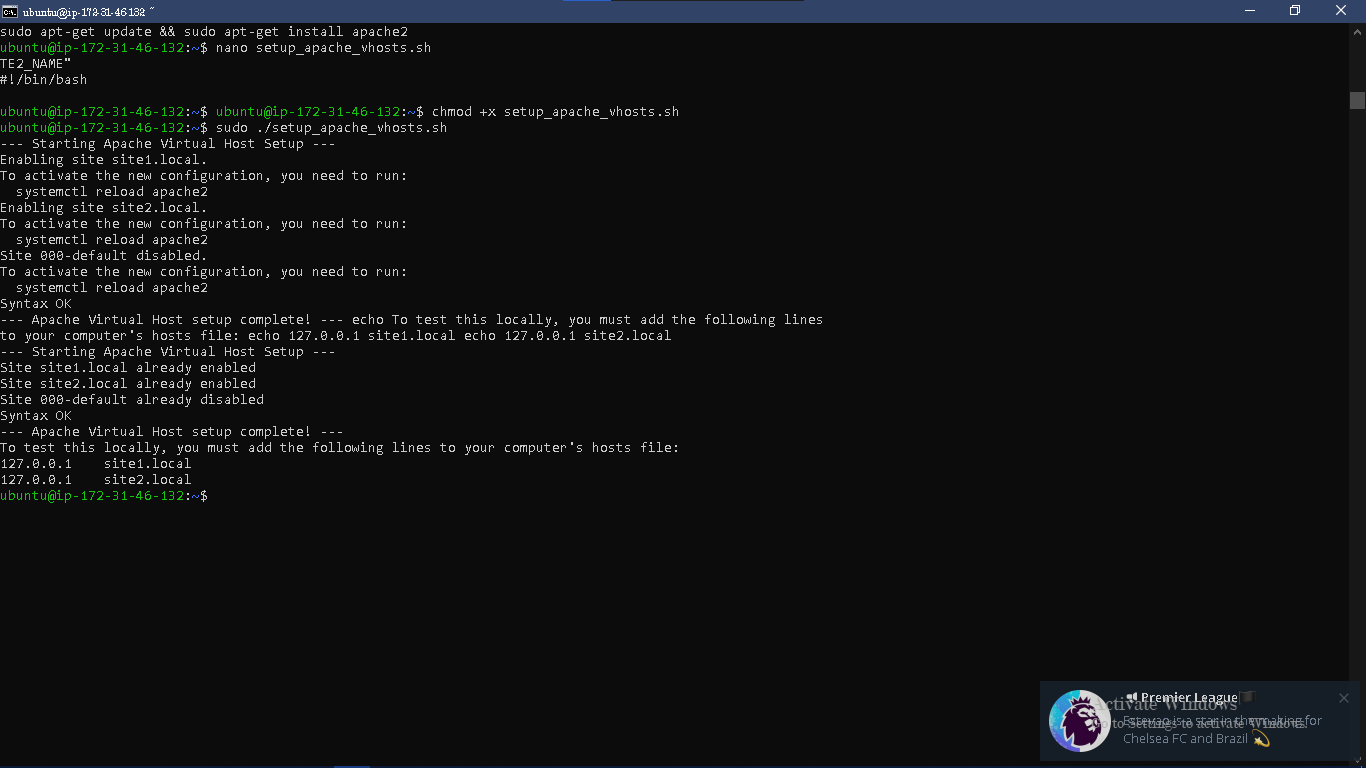
systemctl reload apache2

echo "--- Apache Virtual Host setup complete! ---"

echo "To test this locally, you must add the following lines to your computer's hosts file:"

echo "127.0.0.1    $SITE1\_NAME"

echo "127.0.0.1    $SITE2\_NAME"

Solution: 

4. SSL/TLS Implementation:

Script : #!/bin/bash

SITE\_NAME="site1.local"

SSL\_CERT\_DIR="/etc/ssl/certs"

SSL\_KEY\_DIR="/etc/ssl/private"

APACHE\_CONF\_DIR="/etc/apache2/sites-available"

if [[ "$EUID" -ne 0 ]]; then

echo "This script must be run with sudo or as the root user."

exit 1

fi

if ! command -v apache2 &> /dev/null; then

echo "Apache is not installed. Please run the following command to install it:"

echo "sudo apt-get update && sudo apt-get install apache2"

exit 1

fi

if ! command -v openssl &> /dev/null; then

echo "OpenSSL is not installed. Please run the following command to install it:"

echo "sudo apt-get update && sudo apt-get install openssl"

exit 1

fi

echo "--- Starting SSL Virtual Host Setup ---"

mkdir -p "$SSL\_CERT\_DIR"

mkdir -p "$SSL\_KEY\_DIR"

echo "Generating self-signed SSL certificate..."

echo -e "NG\nLagos\nLagos\n\n\n$SITE\_NAME\n\n\n" | openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout "$SSL\_KEY\_DIR/server.key" -out "$SSL\_CERT\_DIR/server.crt" &> /dev/null

echo "Creating HTTPS Virtual Host configuration for $SITE\_NAME..."

cat > "$APACHE\_CONF\_DIR/$SITE\_NAME-ssl.conf" << EOF

<VirtualHost \*:443>

ServerName $SITE\_NAME

ServerAlias www.$SITE\_NAME

DocumentRoot /var/www/$SITE\_NAME/public\_html

ErrorLog /var/www/$SITE\_NAME/logs/error.log

CustomLog /var/www/$SITE\_NAME/logs/access.log combined

SSLEngine On

SSLCertificateFile $SSL\_CERT\_DIR/server.crt

SSLCertificateKeyFile $SSL\_KEY\_DIR/server.key

<Directory "/var/www/$SITE\_NAME/public\_html">

Options Indexes FollowSymLinks

AllowOverride All

Require all granted

</Directory>

</VirtualHost>

EOF

echo "Enabling the SSL module..."

a2enmod ssl &> /dev/null

echo "Enabling the new HTTPS virtual host..."

a2ensite "$SITE\_NAME-ssl.conf" &> /dev/null

echo "Testing Apache configuration for syntax errors..."

apache2ctl configtest

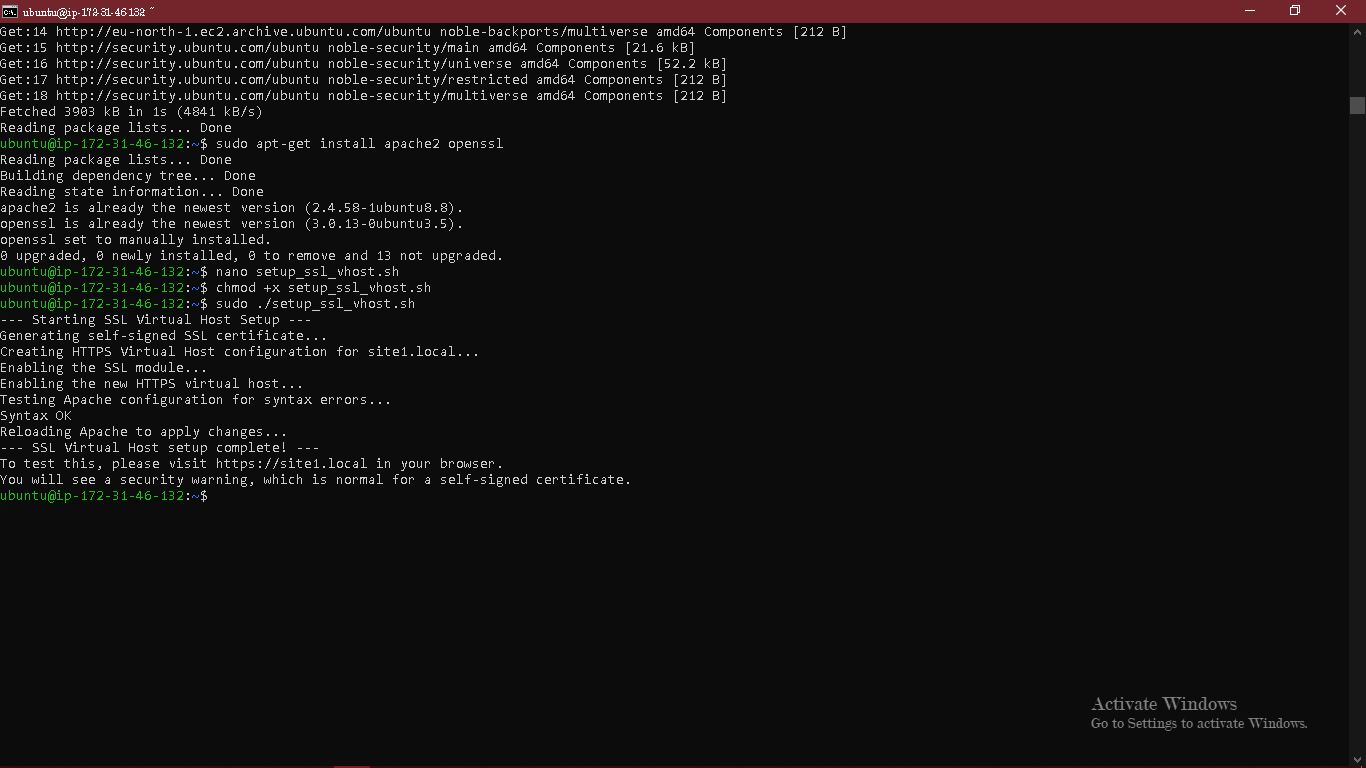
echo "Reloading Apache to apply changes..."

systemctl reload apache2

echo "--- SSL Virtual Host setup complete! ---"

echo "To test this, please visit https://$SITE\_NAME in your browser."

echo "You will see a security warning, which is normal for a self-signed certificate."

Solution: 

5. MySQLRemoteAccess&Security:

Script: #!/bin/bash

DB\_USER="remote\_user"

DB\_PASS="StrongPassword123!"

DB\_NAME="remote\_db"

if [[ "$EUID" -ne 0 ]]; then

echo "This script must be run with sudo or as the root user."

exit 1

fi

if ! command -v mysql &> /dev/null; then

echo "MySQL is not installed. Please run the following command to install it:"

echo "sudo apt-get update && sudo apt-get install mysql-server"

exit 1

fi

echo "--- Starting MySQL Remote Access Configuration ---"

sudo sed -i 's/bind-address = 127.0.0.1/bind-address = 0.0.0.0/' /etc/mysql/mysql.conf.d/mysqld.cnf

sudo systemctl restart mysql

mysql -e "CREATE DATABASE IF NOT EXISTS $DB\_NAME;"

mysql -e "CREATE USER IF NOT EXISTS '$DB\_USER'@'%' IDENTIFIED BY '$DB\_PASS';"

mysql -e "GRANT SELECT, INSERT, UPDATE, DELETE ON $DB\_NAME.\* TO '$DB\_USER'@'%';"

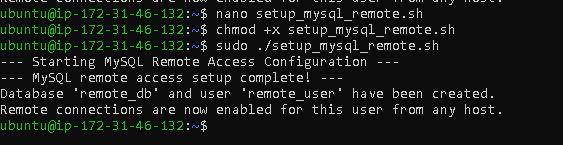
mysql -e "FLUSH PRIVILEGES;"

echo "--- MySQL remote access setup complete! ---"

echo "Database '$DB\_NAME' and user '$DB\_USER' have been created."

echo "Remote connections are now enabled for this user from any host."

Solution :



6. Firewall Configuration:

Script: #!/bin/bash

ALLOWED\_IP\_RANGE="192.168.1.0/24"

if [[ "$EUID" -ne 0 ]]; then

echo "This script must be run with sudo or as the root user."

exit 1

fi

if ! command -v ufw &> /dev/null; then

echo "ufw is not installed. Please run the following command to install it:"

echo "sudo apt-get update && sudo apt-get install ufw"

exit 1

fi

echo "--- Starting UFW Firewall Configuration ---"

ufw --force reset

ufw default deny incoming

ufw default allow outgoing

ufw allow from $ALLOWED\_IP\_RANGE to any port 22

ufw allow from $ALLOWED\_IP\_RANGE to any port 80

ufw allow from $ALLOWED\_IP\_RANGE to any port 443

ufw allow from $ALLOWED\_IP\_RANGE to any port 3306

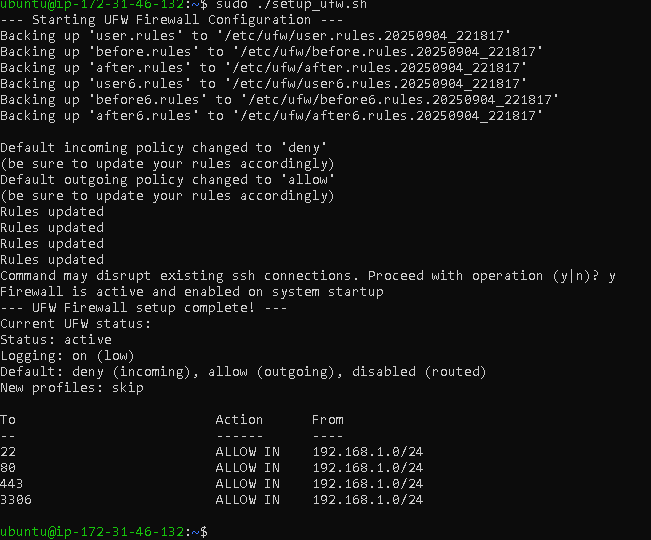
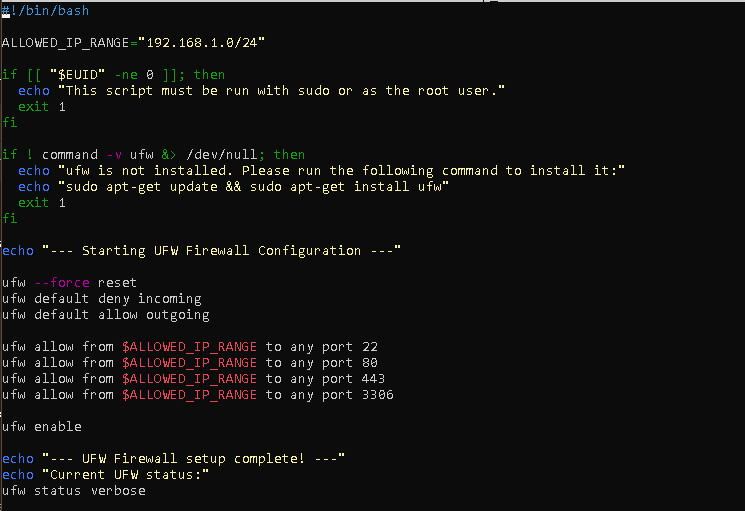
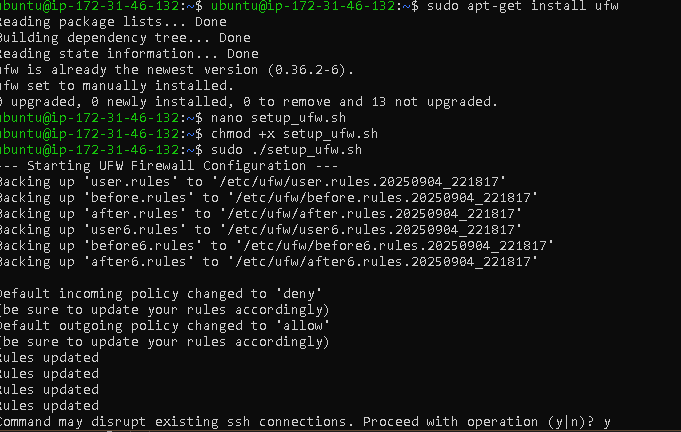
ufw enable

echo "--- UFW Firewall setup complete! ---"

echo "Current UFW status:"

ufw status verbose

Solution:



7. System MonitoringScript :

Script: #!/bin/bash

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

echo "--- System Health Report ---" >> $LOG\_FILE

echo "Timestamp: $(date)" >> $LOG\_FILE

echo "--- CPU Usage ---" >> $LOG\_FILE

iostat >> $LOG\_FILE

echo "--- Memory Usage ---" >> $LOG\_FILE

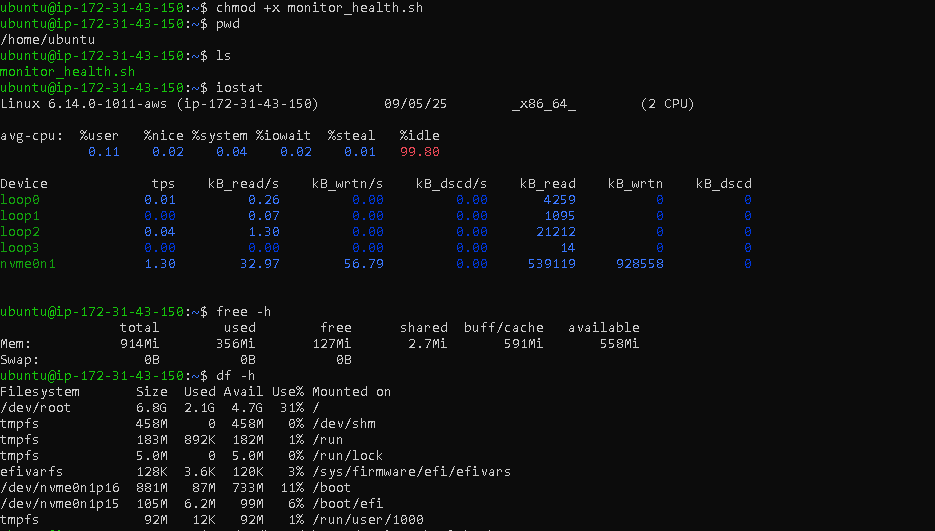
free -h >> $LOG\_FILE

echo "--- Disk Usage ---" >> $LOG\_FILE

df -h >> $LOG\_FILE

echo "" >> $LOG\_FILE

Solution:



8. . Log Rotation Setup

Script: #!/bin/bash

LOG\_FILE="/var/log/my\_app.log"

CONF\_FILE="/etc/logrotate.d/my\_app"

APP\_NAME="my\_app"

if [[ "$EUID" -ne 0 ]]; then

echo "This script must be run with sudo or as the root user."

exit 1

fi

if ! command -v logrotate &> /dev/null; then

echo "logrotate is not installed. Please run the following command to install it:"

echo "sudo apt-get update && sudo apt-get install logrotate"

exit 1

fi

echo "--- Starting Log Rotation Setup for $APP\_NAME ---"

echo "Creating a dummy log file for testing..."

echo "This is a custom log entry." > $LOG\_FILE

echo "Creating logrotate configuration file at $CONF\_FILE..."

cat > $CONF\_FILE << EOF

$LOG\_FILE {

daily

rotate 7

compress

missingok

notifempty

su root root

}

EOF

echo "Logrotate configuration created. To test it, you can run the following command:"

echo "sudo logrotate -f $CONF\_FILE"

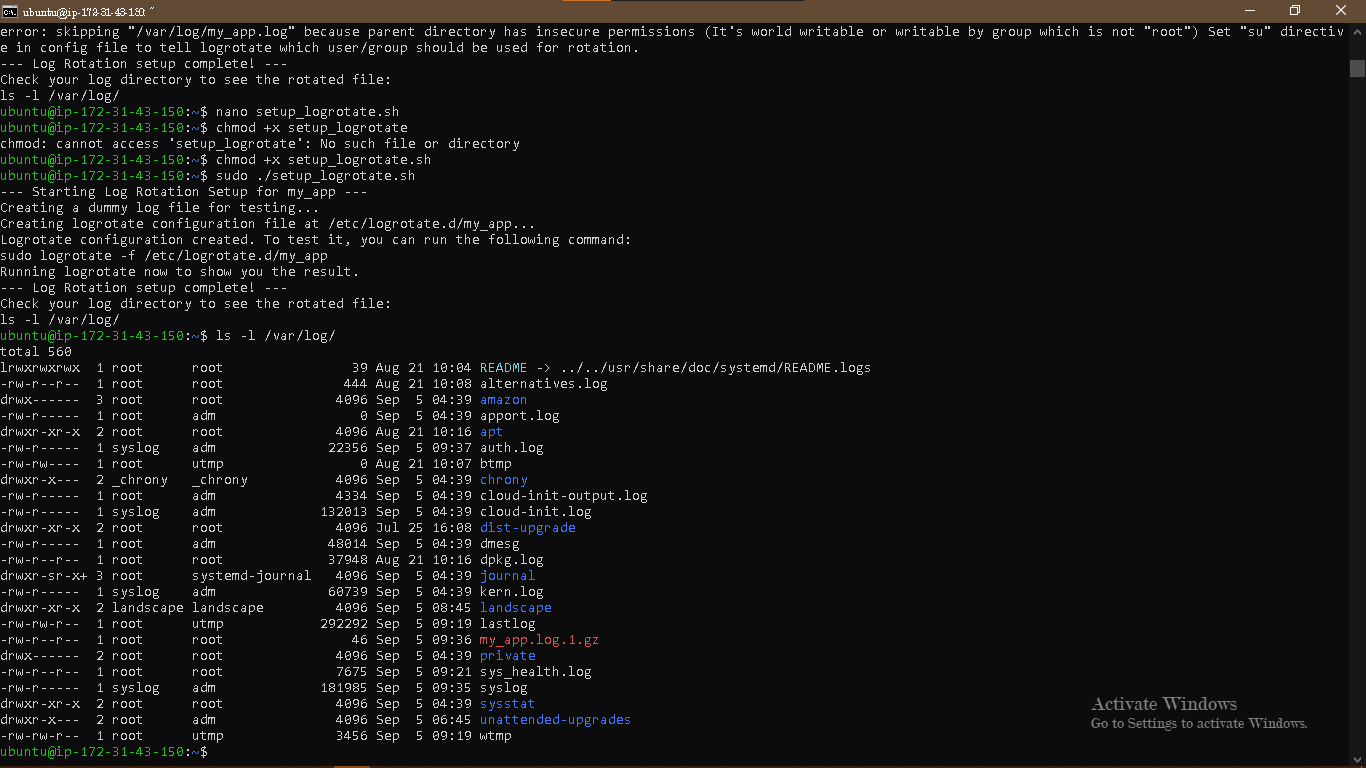
echo "Running logrotate now to show you the result."

logrotate -f $CONF\_FILE

echo "--- Log Rotation setup complete! ---"

echo "Check your log directory to see the rotated file:"

echo "ls -l /var/log/"

Solution ; 

9. . DNSServerSetup

Script:

#!/bin/bash

BIND\_CONF\_DIR="/etc/bind"

ZONE\_NAME="myuniversity.local"

ZONE\_FILE="$BIND\_CONF\_DIR/$ZONE\_NAME.db"

if [[ "$EUID" -ne 0 ]]; then

echo "This script must be run with sudo or as the root user."

exit 1

fi

if ! command -v named-checkconf &> /dev/null; then

echo "bind9 is not installed. Please run the following command to install it:"

echo "sudo apt-get update && sudo apt-get install bind9 bind9utils"

exit 1

fi

echo "--- Starting BIND9 DNS Server Setup ---"

echo "Configuring named.conf.options for caching and forwarding..."

sed -i 's/dnssec-validation auto;/dnssec-validation no;/' "$BIND\_CONF\_DIR/named.conf.options"

sed -i '/listen-on-v6 { any; };/a \

\ forwarders { \

\ 8.8.8.8; \

\ 8.8.4.4; \

\ }; \

\ allow-query { any; };' "$BIND\_CONF\_DIR/named.conf.options"

echo "Adding custom zone to named.conf.local..."

cat >> "$BIND\_CONF\_DIR/named.conf.local" << EOF

zone "$ZONE\_NAME" {

type master;

file "$ZONE\_FILE";

};

EOF

echo "Creating zone file for $ZONE\_NAME..."

cat > "$ZONE\_FILE" << EOF

\$TTL 86400

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

2024040901 ; Serial

3600 ; Refresh

1800 ; Retry

604800 ; Expire

86400 ; Negative Cache TTL

)

@ IN NS ns1.myuniversity.local.

ns1 IN A 127.0.0.1

@ IN A 127.0.0.1

www IN A 127.0.0.1

mail IN MX 10 mail.myuniversity.local.

mail IN A 127.0.0.1

EOF

echo "Setting correct ownership and permissions for zone file..."

chown bind:bind "$ZONE\_FILE"

chmod 644 "$ZONE\_FILE"

echo "Testing BIND9 configuration for syntax errors..."

named-checkconf

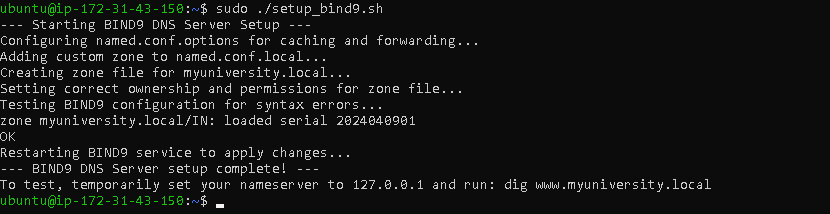
named-checkzone "$ZONE\_NAME" "$ZONE\_FILE"

echo "Restarting BIND9 service to apply changes..."

systemctl restart bind9

echo "--- BIND9 DNS Server setup complete! ---"

echo "To test, temporarily set your nameserver to 127.0.0.1 and run: dig [www.myuniversity.local](http://www.myuniversity.local)"

Solution: 

10. SSHKeyAuthentication+Hardening:

Script: #!/bin/bash

SSH\_DIR="/etc/ssh"

SSHD\_CONFIG="$SSH\_DIR/sshd\_config"

SSH\_USER=""

if [[ "$EUID" -ne 0 ]]; then

echo "This script must be run with sudo or as the root user."

exit 1

fi

if ! command -v sshd &> /dev/null; then

echo "OpenSSH server is not installed. Please run the following command to install it:"

echo "sudo apt-get update && sudo apt-get install openssh-server"

exit 1

fi

echo "--- Starting SSH Hardening Setup ---"

read -p "Enter the username to configure SSH for (e.g., ubuntu): " SSH\_USER

if [[ -z "$SSH\_USER" ]]; then

echo "Username cannot be empty. Exiting."

exit 1

fi

if [[ "$SSH\_USER" == "root" ]]; then

echo "This script is designed to disable root login. Please enter a standard user. Exiting."

exit 1

fi

if ! id "$SSH\_USER" &> /dev/null; then

echo "User '$SSH\_USER' does not exist. Please create the user first. Exiting."

exit 1

fi

echo "Creating .ssh directory and authorized\_keys for user '$SSH\_USER'..."

mkdir -p /home/$SSH\_USER/.ssh

touch /home/$SSH\_USER/.ssh/authorized\_keys

chown -R $SSH\_USER:$SSH\_USER /home/$SSH\_USER/.ssh

chmod 700 /home/$SSH\_USER/.ssh

chmod 600 /home/$SSH\_USER/.ssh/authorized\_keys

echo "Generating an SSH key pair for user '$SSH\_USER'..."

ssh-keygen -t rsa -b 4096 -f /home/$SSH\_USER/.ssh/id\_rsa -N ""

echo "Copying the public key to authorized\_keys..."

cat /home/$SSH\_USER/.ssh/id\_rsa.pub >> /home/$SSH\_USER/.ssh/authorized\_keys

echo "Configuring sshd\_config to disable password authentication and root login..."

sed -i 's/^#\?PubkeyAuthentication.\*/PubkeyAuthentication yes/' "$SSHD\_CONFIG"

sed -i 's/^#\?PasswordAuthentication.\*/PasswordAuthentication no/' "$SSHD\_CONFIG"

sed -i 's/^#\?PermitRootLogin.\*/PermitRootLogin no/' "$SSHD\_CONFIG"

echo "Restarting the SSH service to apply changes..."

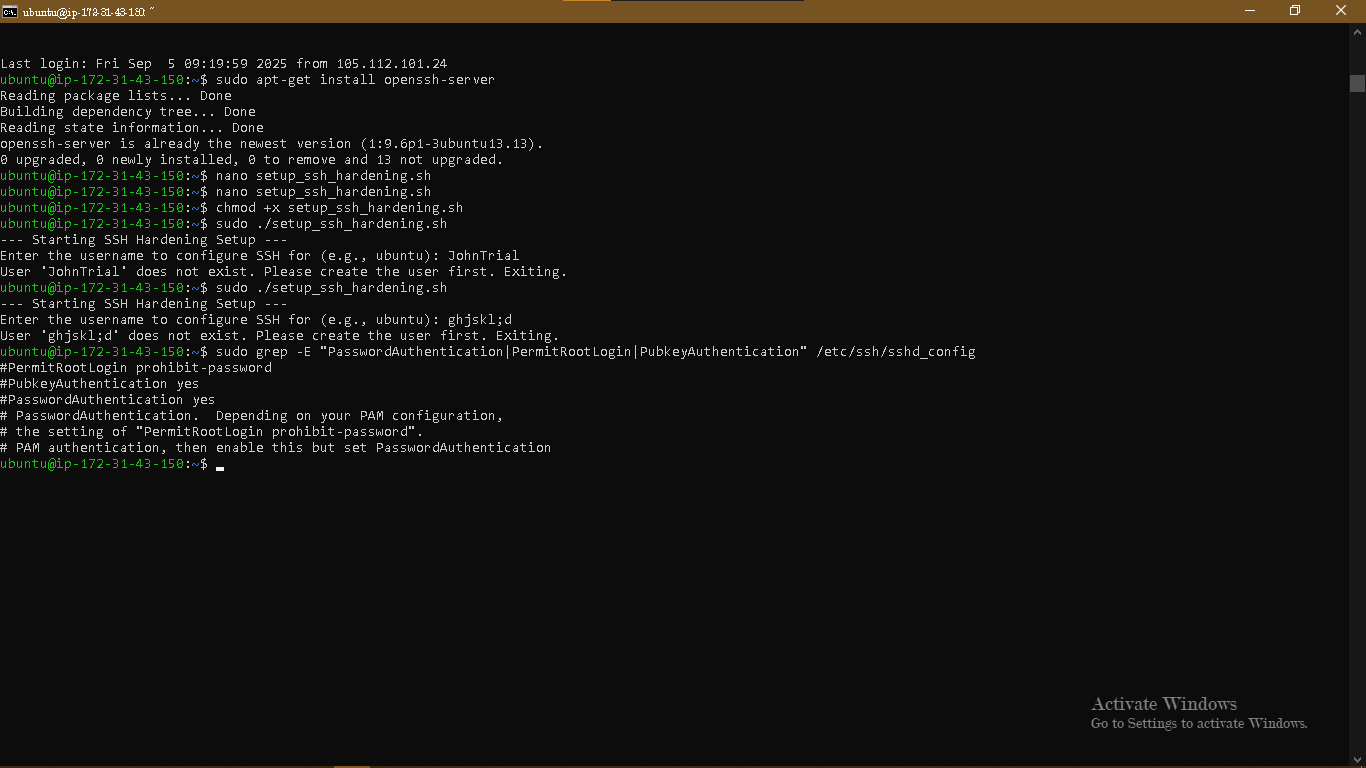
systemctl restart sshd

echo "--- SSH Hardening complete! ---"

echo "The SSH private key has been saved to: /home/$SSH\_USER/.ssh/id\_rsa"

echo "Copy this private key to your local machine to connect."

echo "Use the command: ssh $SSH\_USER@your\_server\_ip"

solution: 

11. Script:

#!/bin/bash

set -e

echo "=== Step 1: Creating application script ==="

cat << 'EOF' | sudo tee /usr/local/bin/my\_app.sh > /dev/null

#!/bin/bash

echo "Hello World! The service ran at $(date)" >> /var/log/my\_app.log

EOF

sudo chmod +x /usr/local/bin/my\_app.sh

echo "=== Step 2: Creating systemd service file ==="

cat << 'EOF' | sudo tee /etc/systemd/system/my\_app.service > /dev/null

[Unit]

Description=My Hello World App

After=network.target

[Service]

Type=simple

ExecStart=/usr/local/bin/my\_app.sh

Restart=always

[Install]

WantedBy=multi-user.target

EOF

echo "=== Step 3: Reloading systemd and enabling service ==="

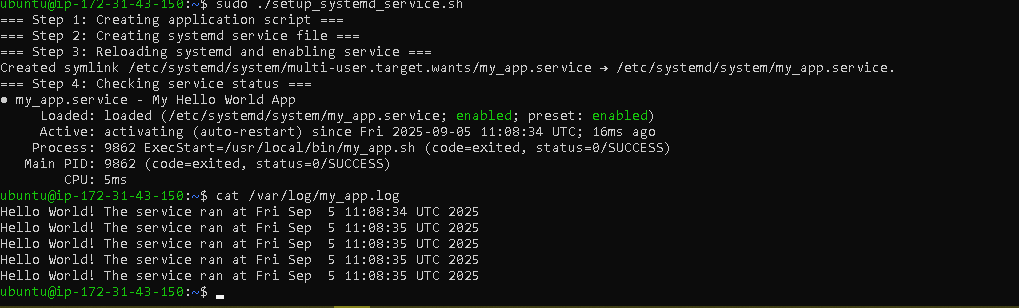
sudo systemctl daemon-reload

sudo systemctl enable my\_app.service

sudo systemctl start my\_app.service

echo "=== Step 4: Checking service status ==="

sudo systemctl status my\_app.service --no-pager

solution: 

12. DiskPartitioning & Mounting

Script:

#!/bin/bash

set -e

echo "=== Step 1: Create a 200MB virtual disk file ==="

DISK\_FILE=/mnt/virtualdisk.img

sudo dd if=/dev/zero of=$DISK\_FILE bs=1M count=200

echo "=== Step 2: Attach the file as a loop device ==="

LOOP\_DEVICE=$(sudo losetup -f --show $DISK\_FILE)

echo "Loop device created: $LOOP\_DEVICE"

echo "=== Step 3: Partition the loop device (single primary partition) ==="

echo -e "n\np\n1\n\n\nw" | sudo fdisk $LOOP\_DEVICE

echo "=== Step 4: Refresh loop devices and map partitions ==="

sudo losetup -d $LOOP\_DEVICE

LOOP\_DEVICE=$(sudo losetup -f --show $DISK\_FILE)

sudo partprobe $LOOP\_DEVICE

PARTITION=${LOOP\_DEVICE}p1

echo "Partition created: $PARTITION"

echo "=== Step 5: Format partition as ext4 ==="

sudo mkfs.ext4 -F $PARTITION

echo "=== Step 6: Create mount point and mount temporarily ==="

MOUNT\_POINT=/mnt/mydata

sudo mkdir -p $MOUNT\_POINT

sudo mount $PARTITION $MOUNT\_POINT

echo "=== Step 7: Verify temporary mount ==="

df -h | grep $MOUNT\_POINT

echo "=== Step 8: Add to /etc/fstab for persistence ==="

UUID=$(sudo blkid -s UUID -o value $PARTITION)

echo "UUID=$UUID $MOUNT\_POINT ext4 defaults 0 2" | sudo tee -a /etc/fstab

echo "=== Step 9: Test fstab by unmounting and remounting ==="

sudo umount $MOUNT\_POINT

sudo mount -a

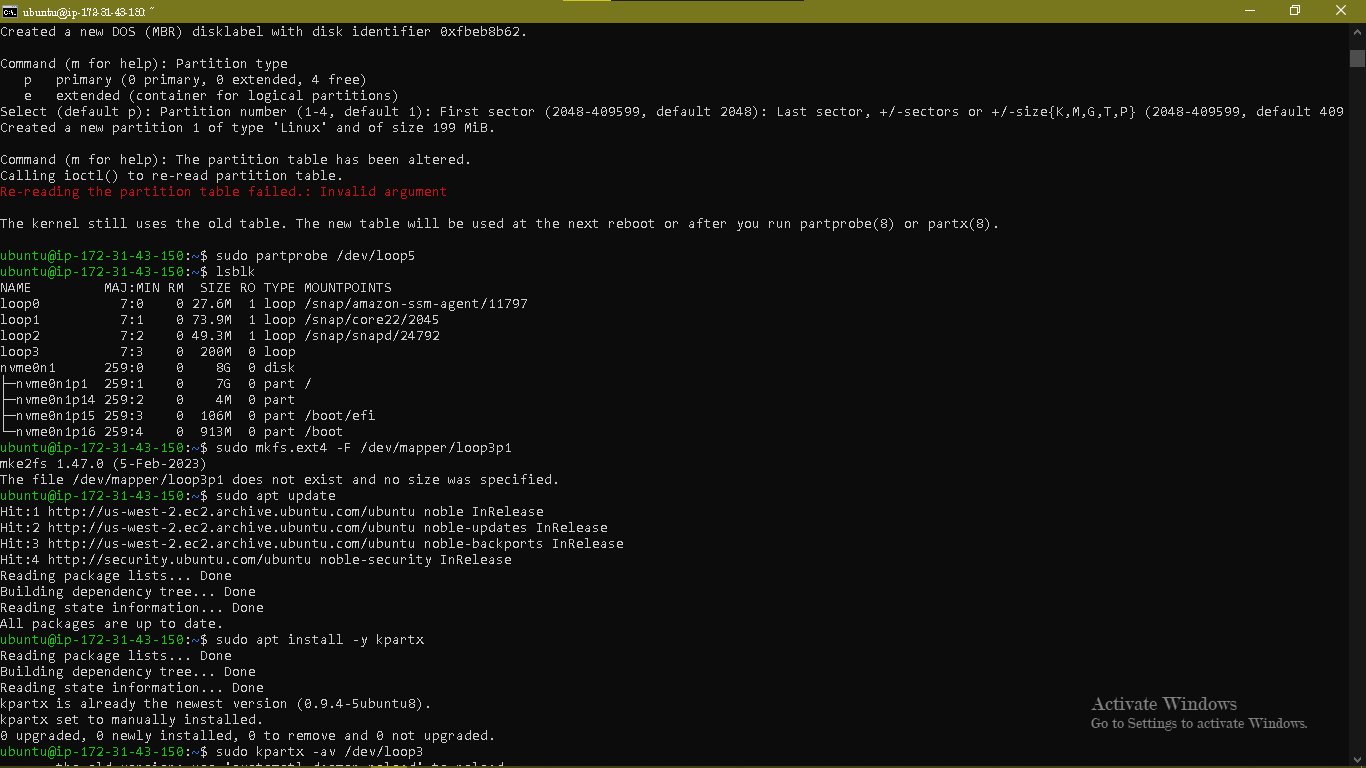
df -h | grep $MOUNT\_POINT

echo "=== Step 10: Reboot test (simulated) ==="

echo "Normally you'd run: sudo reboot"

echo "After reboot, check: df -h | grep $MOUNT\_POINT"

Solution:



13. . Postfix Mail Server (Local Only)

Script: #!/bin/bash

set -e

echo "=== Step 1: Updating packages ==="

sudo apt update -y

echo "=== Step 2: Installing Postfix (local only) and mailutils ==="

# Preseed Postfix configuration (local only)

echo "postfix postfix/mailname string localhost" | sudo debconf-set-selections

echo "postfix postfix/main\_mailer\_type string Local only" | sudo debconf-set-selections

sudo DEBIAN\_FRONTEND=noninteractive apt install -y postfix mailutils

echo "=== Step 3: Checking Postfix status ==="

sudo systemctl enable postfix

sudo systemctl start postfix

sudo systemctl status postfix --no-pager

echo "=== Step 4: Creating test users (alice & bob) ==="

if ! id alice &>/dev/null; then

sudo adduser --disabled-password --gecos "" alice

fi

if ! id bob &>/dev/null; then

sudo adduser --disabled-password --gecos "" bob

fi

echo "=== Step 5: Sending test mail from alice to bob ==="

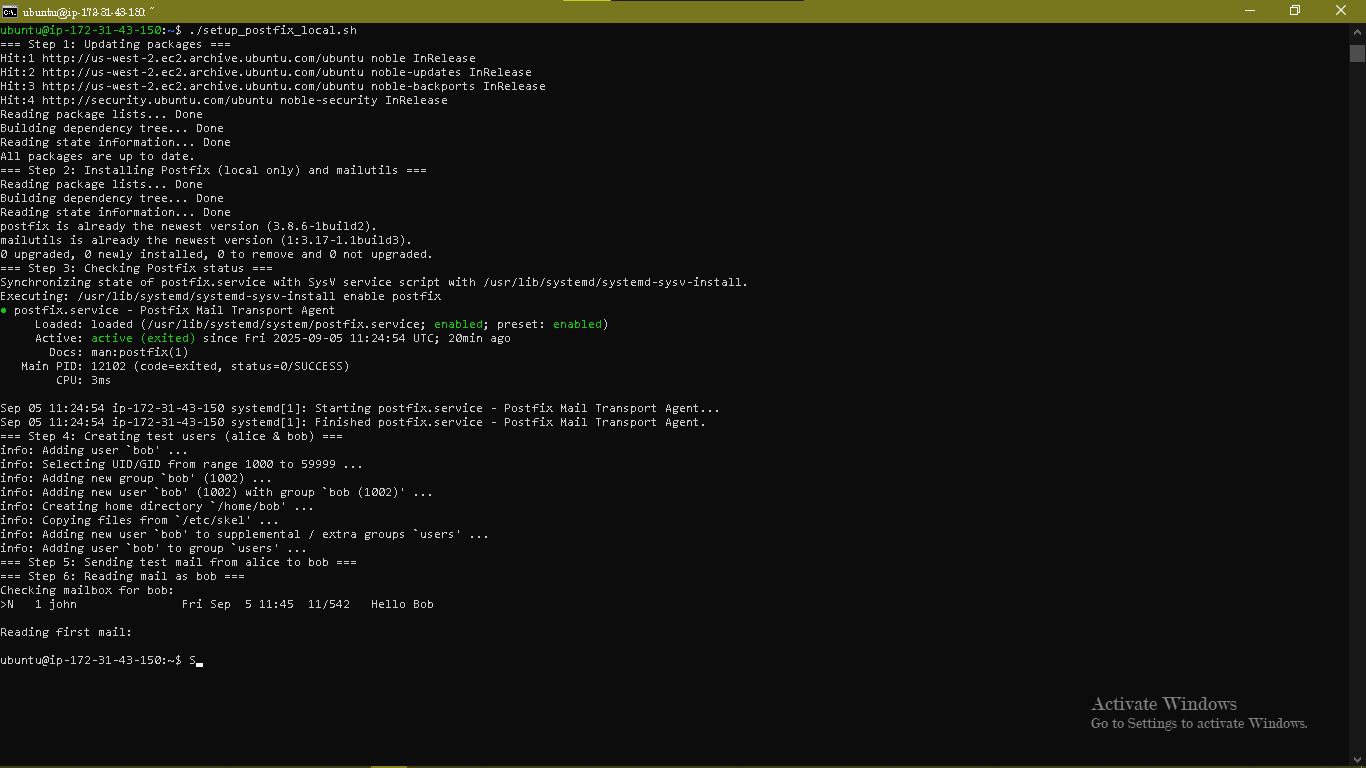
echo -e "Subject: Hello Bob\nHi Bob, this is a local mail test from Alice." | sudo -u alice sendmail bob

sleep 2 # Give Postfix time to deliver

echo "=== Step 6: Reading mail as bob ==="

sudo -u bob bash -c "echo 'Checking mailbox for bob:' && mail -H && echo && echo 'Reading first mail:' && echo 1 | mail -f"

Solution:



14. Backup&RestoreProject

Script:

#!/bin/bash

set -e

# Directories

SOURCE\_DIR="/var/www/html"

BACKUP\_DIR="/backup"

# Ensure backup directory exists

sudo mkdir -p $BACKUP\_DIR

# Create a timestamp

TIMESTAMP=$(date +"%Y%m%d\_%H%M%S")

# Backup filename

BACKUP\_FILE="$BACKUP\_DIR/html\_backup\_$TIMESTAMP.tar.gz"

echo ">>> Creating backup of $SOURCE\_DIR to $BACKUP\_FILE"

# Create the backup

sudo tar -czf $BACKUP\_FILE -C /var/www html

echo ">>> Backup created successfully!"

# List backups

ls -lh $BACKUP\_DIR/html\_backup\_\*.tar.gz

# Test restore: Extract latest backup into /var/www/html\_restored

LATEST\_BACKUP=$(ls -t $BACKUP\_DIR/html\_backup\_\*.tar.gz | head -n 1)

RESTORE\_DIR="/var/www/html\_restored"

echo ">>> Restoring latest backup $LATEST\_BACKUP to $RESTORE\_DIR"

# Ensure restore directory exists

sudo rm -rf $RESTORE\_DIR

sudo mkdir -p $RESTORE\_DIR

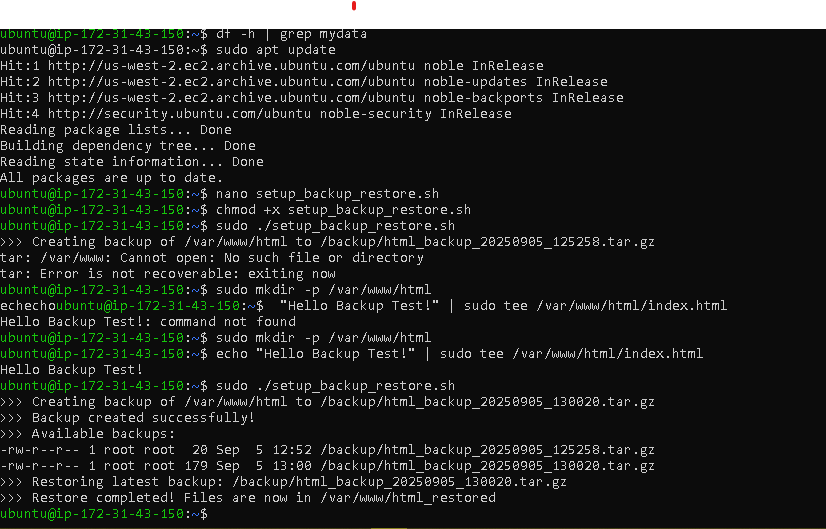
# Extract

sudo tar -xzf $LATEST\_BACKUP -C /var/www

sudo mv /var/www/html $RESTORE\_DIR

echo ">>> Restore completed! Files are in $RESTORE\_DIR"

Solution:



15. Containerization Challenge

Script: #!/bin/bash

set -e

echo ">>> Updating packages..."

sudo apt update -y

echo ">>> Installing prerequisites..."

sudo apt install -y apt-transport-https ca-certificates curl software-properties-common

echo ">>> Installing Docker..."

# Add Docker’s official GPG key

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg

# Add stable repo

echo \

"deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] \

https://download.docker.com/linux/ubuntu $(lsb\_release -cs) stable" | \

sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

sudo apt update -y

sudo apt install -y docker-ce docker-ce-cli containerd.io

echo ">>> Enabling and starting Docker..."

sudo systemctl enable docker

sudo systemctl start docker

echo ">>> Pulling Nginx container..."

sudo docker pull nginx:latest

echo ">>> Running Nginx container on port 8080..."

# Stop any existing container named mynginx

if [ "$(sudo docker ps -aq -f name=mynginx)" ]; then

sudo docker rm -f mynginx

fi

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

echo ">>> Checking container status..."

sudo docker ps | grep mynginx

echo ">>> Testing Nginx locally..."

curl -I http://localhost:8080 || true

echo ">>> Done! Visit http://<your-server-public-ip>:8080 in your browser."

Solution:

