Linux Server Environment Toolkit

Overview

This project documents my journey creating a personal Linux server environment toolkit designed to automate system monitoring and alerting, with a focus on disk space checks and scheduled maintenance.

Initial Goals

- Automate disk space monitoring with email notifications
- Schedule routine maintenance tasks using cron
- Implement secure handling of sensitive information like email passwords

Step 1: Environment Setup and Package Installation

Starting with an Ubuntu system, I identified the essential tools needed: `msmtp` for sending emails, `mailutils` for composing mail, `cron` for task scheduling, and `gpg` for encryption.

Installed packages using:

```
sudo apt update
sudo apt install -y msmtp mailutils cron gpg
```

This step laid the foundation for automation and secure communications.

Step 2: Configuring Email Sending via msmtp

To send alert emails without running a full mail server, I configured `msmtp` to use Gmail's SMTP server. I created the `~/.msmtprc` file with proper authentication and TLS settings:

```
defaults
auth           on
tls            on
tls_trust_file /etc/ssl/certs/ca-certificates.crt
```

```
logfile        ~/.msmtp.log

account default
host           smtp.gmail.com
port           587
from           myemail@gmail.com
user           myemail@gmail.com
passwordeval   "gpg --quiet --for-your-eyes-only --no-tty --decrypt ~/.msmtp-password.gpg"
```

I enforced strict permissions to secure the configuration:

```
chmod 600 ~/.msmtprc
```

This setup ensured reliable and secure email delivery from the command line.

Step 3: Securing Credentials with GPG

Recognizing the risks of storing passwords in plaintext, I encrypted my SMTP password using GPG symmetric encryption:

```
echo "my-gmail-password" | gpg --symmetric --cipher-algo AES256 -o ~/.msmtp-password.gpg
chmod 600 ~/.msmtp-password.gpg
```

This approach integrates seamlessly with `msmtp` and maintains credential confidentiality.

Step 4: Developing the Disk Monitoring Script

I wrote a bash script to monitor disk usage and trigger an email alert when usage exceeds a threshold (80%):

```
#!/bin/bash

THRESHOLD=80
EMAIL="myemail@gmail.com"

used=$(df / | grep / | awk '{ print $5 }' | sed 's/%//g')

if [ "$used" -gt "$THRESHOLD" ]; then
    echo "Warning: Disk usage is above $THRESHOLD% ($used%) on / partition." | mail -s "Disk Space Alert" "$EMAIL"
fi
```

After making the script executable (`chmod +x disk_monitor.sh`), it was ready for automated execution.

Step 5: Automating Execution with Cron

To ensure continuous monitoring, I scheduled the script to run hourly by adding this entry to my crontab (`crontab -e`):

0 * * * * /home/myuser/disk_monitor.sh

This automation guarantees timely alerts without manual intervention.

Step 6: Testing and Validation

I verified the email setup by sending a test message:

echo "Test email from msmtp" | mail -s "Test Email" myemail@gmail.com

Additionally, I ran the disk monitoring script manually to confirm it correctly detects disk usage and sends alerts.

Reflections

This project strengthened my understanding of Linux automation tools, secure credential management, and email integration. It serves as a scalable foundation for further system monitoring and maintenance enhancements.