

UNIVERSITE DE KINSHASA

FACULTE DES SCIENCES ET TECHNOLOGIES
DEPARTEMENT DE MATHÉMATIQUES, STATISTIQUE ET INFORMATIQUE



RAPPORT DU TRAVAIL DE SYSTEME D'EXPLOITATION

**SUJET : CREATION D'UN SCRIPT BASH POUR SURVEILLER
LES LOGS SYSTEME ET ENVOYER DES ALERTES PAR MAIL**

(GROUPE 23)

Travail présenter par :

1. Mimbayi lieko prince
2. Matiabo bomay emmanuel
3. Kibeye danisi joel
4. Mukoko mandina harodl
5. Inyobo mobemba romeo
6. Landu kunzika david
7. Mobaku makala heritier
8. Jules omesudu babe
9. Makila makita merseigne
10. Omehambe ososmbo Israel

Prof: KASENGEDIA MOTUMBE Pierre

ANNEE ACADEMIQUE 2024-2025

I. Introduction

Aujourd'hui, les systèmes informatiques occupent une place centrale dans le fonctionnement de nombreuses entreprises et organisations. Pour garantir leur bon fonctionnement, il est essentiel de pouvoir surveiller en permanence l'état du système, détecter rapidement les erreurs, et intervenir en cas de problème. C'est justement le rôle des fichiers de logs : ils enregistrent tout ce qui se passe sur une machine — qu'il s'agisse de simples informations de fonctionnement, d'avertissements, ou d'erreurs critiques. Ces fichiers sont donc une source précieuse pour comprendre ce qui se passe réellement sur un système.

Cependant, dans la réalité, ces fichiers peuvent être très volumineux et difficiles à lire. Il est presque impossible, voire irréaliste, pour un administrateur système de les consulter manuellement tous les jours, surtout s'il gère plusieurs serveurs. C'est là qu'intervient l'idée de ce projet : **automatiser la surveillance des logs système**, afin d'être averti immédiatement lorsqu'un événement important ou anormal est détecté.

L'objectif de ce projet est donc de créer un **script Bash simple et efficace**, qui va parcourir régulièrement les fichiers de logs (/var/log/syslog) pour repérer certains mots-clés ou messages d'erreur spécifiques (par exemple : "error", "failed", "unauthorized access", etc.). Dès qu'un de ces éléments est trouvé, le script devra automatiquement **envoyer un mail d'alerte** à l'administrateur pour l'informer du problème.

La problématique à laquelle ce projet répond est la suivante :

Comment être informé rapidement d'un incident système sans avoir à surveiller manuellement les fichiers de logs en continu ?

Bien sûr, il existe déjà des outils très puissants pour la supervision (comme Zabbix, Nagios ou encore Graylog), mais ils peuvent être complexes à mettre en place, lourds, ou non adaptés à des petites structures ou à des projets étudiants. Dans notre cas, un **script Bash léger, facile à comprendre et personnalisable** suffit largement pour répondre au besoin, tout en permettant de mettre en pratique plusieurs notions vues en cours, comme la manipulation de fichiers, les conditions, les boucles, la gestion des tâches planifiées, ou encore l'envoi de mails en ligne de commande.

En résumé : au travers de ce projet nous proposons une solution simple mais utile pour améliorer la réactivité face aux incidents systèmes, tout en renforçant la sécurité et la stabilité du serveur.

II. DEFINITION DES CONCEPTS

1. Le shell Bash

a. Définition

Bash, abréviation de *Bourne Again SHell*, est l'un des interpréteurs de commandes les plus utilisés dans les systèmes Unix et Linux. Il a été conçu pour permettre aux utilisateurs d'exécuter des commandes, mais aussi pour écrire des scripts qui automatisent des tâches répétitives. Cela en fait un outil incontournable dans l'administration système. Grâce à Bash, un administrateur peut programmer des scripts pour surveiller un serveur, gérer des fichiers, contrôler les ressources système, ou même envoyer des alertes en cas de problème.

b. Bref historique

Le **shell Unix historique**, appelé Bourne Shell, a été créé par Stephen Bourne en 1977. Assez limitée en termes de fonctionnalités, cette version a toutefois posé les bases de l'ensemble des interpréteurs de commandes ultérieurs. Dans les années 1980, il a été suivi par le C Shell (csh) de Bill Joy puis le Korn Shell, ou ksh, de David Korn.

2. Les fichiers de logs système

Les fichiers de logs sont comme le journal intime du système. Ils conservent une trace de tout ce qui se passe : démarrage, connexions, erreurs, mises à jour, et bien plus encore. Ces fichiers sont essentiels pour comprendre ce qui s'est passé sur une machine, surtout lorsqu'un incident survient.

Ils sont générés automatiquement par le système d'exploitation et les services installés. Sous Linux, on les retrouve généralement dans le répertoire `/var/log`. Par exemple, `syslog` contient des informations générales sur le système, tandis que `auth.log` regroupe les tentatives de connexion. Ces logs peuvent rapidement devenir volumineux, d'où l'intérêt de savoir les filtrés.

Emplacement courant : `/var/log/`

Exemples de fichiers log :

- `/var/log/auth.log` → journalisation des authentifications
- `/var/log/syslog` → journalisation générale du système

III. Mise en place d'un système d'envoi de mails par Postfix avec SMTP Gmail sur Ubuntu

1. Objectif

Dans le cadre de notre projet, nous avons mis en place un système permettant d'envoyer des mails depuis le terminal Linux. Pour cela, nous avons utilisé le serveur de messagerie **Postfix** ainsi que le serveur SMTP de **Gmail** (smtp.gmail.com).

L'objectif était de pouvoir envoyer des mails directement depuis la ligne de commande, sans passer par une interface graphique. Ce procédé présente plusieurs avantages : il permet d'envoyer des notifications système automatiques, des messages d'information ou encore des alertes en cas de détection d'incidents.

Dans le contexte spécifique de notre projet, cette mise en place nous a permis de recevoir automatiquement des alertes par mail lorsqu'un problème est détecté dans les journaux système. Cela dans le but de faciliter notre suivi et notre réactivité face aux éventuels incidents sur le terminal.

2. Outils et services utilisés

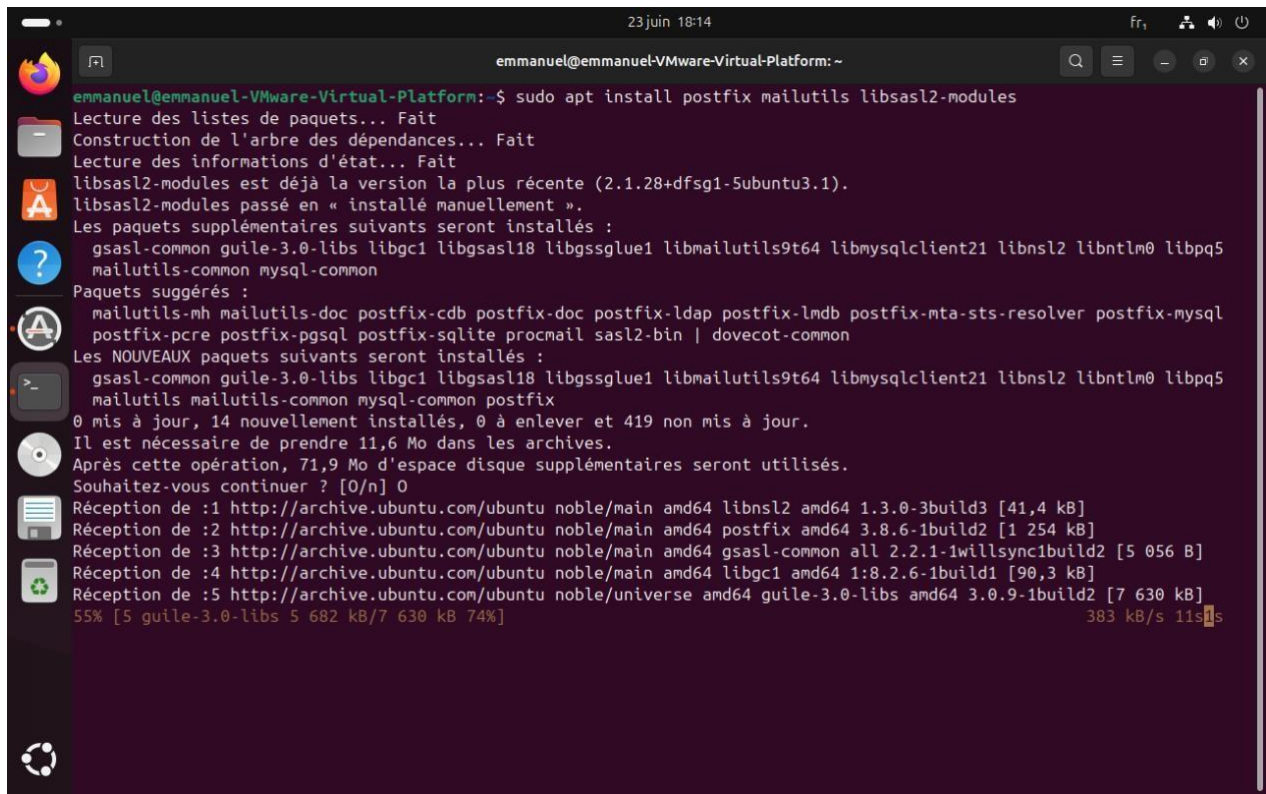
Outil / Service	Rôle
Postfix	Serveur de SMTP (MTA : mail transfert agent) local pour l'envoi
Mailutils	Outil en ligne de commande pour envoyer des mails
libsasl2-modules	Module d'authentification SMTP
smtp.gmail.com:587	Serveur SMTP Gmail utilisé en relais
Mot de passe d'application Gmail	Authentification SMTP sécurisée sans 2FA

3. Installation des paquets nécessaires

La première étape a été l'installation des paquets nécessaires au moyen des commandes ci-dessous.

Commande :

```
sudo apt update sudo apt install postfix mailutils  
libssl2-modules
```



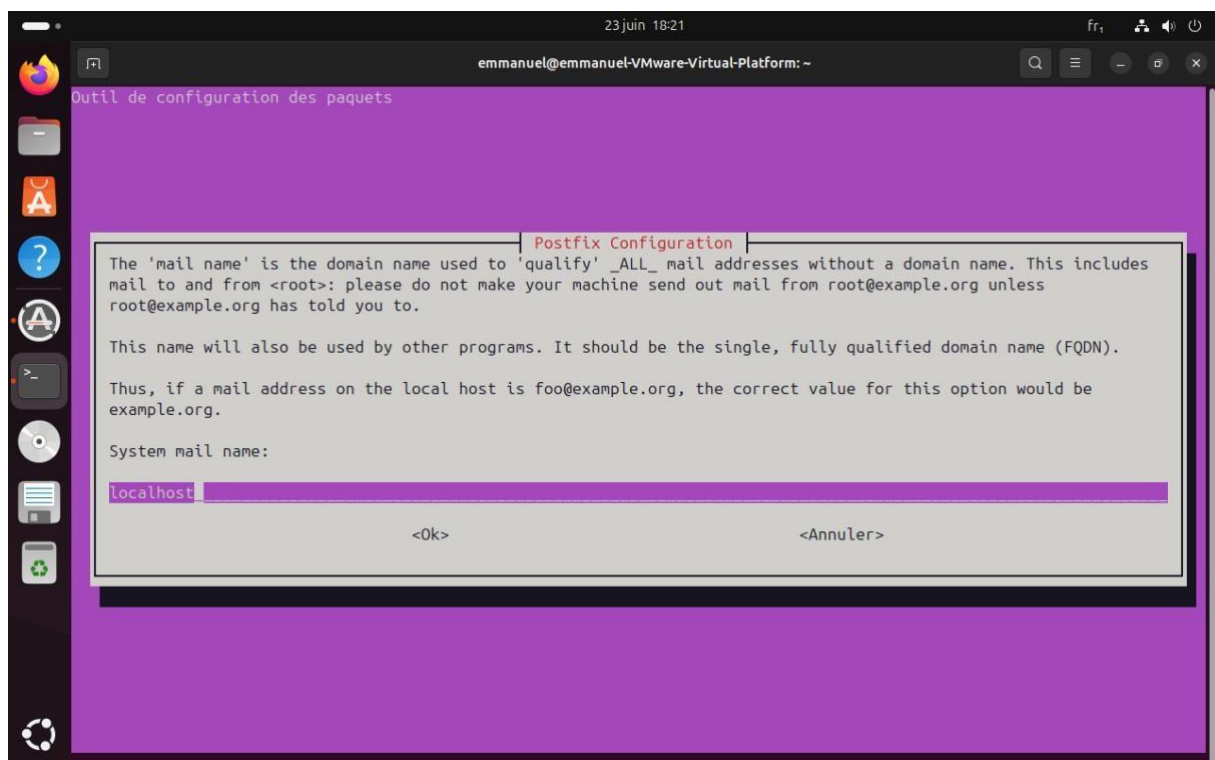
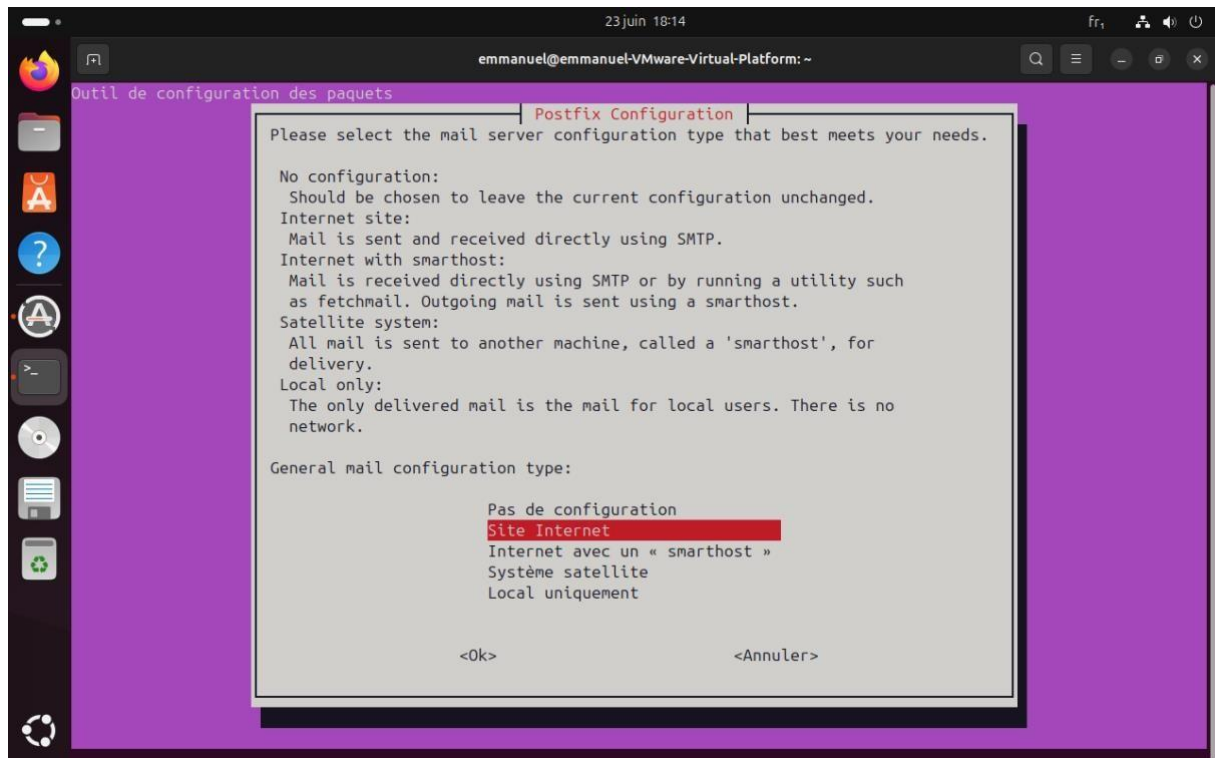
The screenshot shows a terminal window titled "emmanuel@emmanuel-VMware-Virtual-Platform: ~" with a timestamp of "23 juin 18:14". The user has executed the command "sudo apt install postfix mailutils libssl2-modules". The terminal output shows the following steps:

- Reading package lists... Done
- Building dependency tree... Done
- Reading state information... Done
- libssl2-modules is already the latest version (2.1.28+dfsg1-5ubuntu3.1).
- libssl2-modules is already installed manually.
- The following additional packages will be installed:
 - gsasl-common guile-3.0-libs libgc1 libgsasl18 libgssglue1 libmailutils9t64 libmysqlclient21 libnsl2 libntlm0 libpq5 mailutils-common mysql-common
- Suggested packages:
 - mailutils-mh mailutils-doc postfix-cdb postfix-doc postfix-ldap postfix-lmdb postfix-mta-sts-resolver postfix-mysql postfix-pcre postfix-pgsql postfix-sqlite procmail sasl-bin | dovecot-common
- The following NEW packages will be installed:
 - gsasl-common guile-3.0-libs libgc1 libgsasl18 libgssglue1 libmailutils9t64 libmysqlclient21 libnsl2 libntlm0 libpq5 mailutils mailutils-common mysql-common postfix
- 0 upgraded, 14 newly installed, 0 to remove and 419 not upgraded.
- It is necessary to take 11.6 Mo in archives.
- After this operation, 71.9 Mo of disk space will be used.
- Do you want to continue? [Y/n] Y
- Receiving from :1 http://archive.ubuntu.com/ubuntu noble/main amd64 libnsl2 amd64 1.3.0-3build3 [41,4 kB]
- Receiving from :2 http://archive.ubuntu.com/ubuntu noble/main amd64 postfix amd64 3.8.6-1build2 [1 254 kB]
- Receiving from :3 http://archive.ubuntu.com/ubuntu noble/main amd64 gsasl-common all 2.2.1-1willsync1build2 [5 056 B]
- Receiving from :4 http://archive.ubuntu.com/ubuntu noble/main amd64 libgc1 amd64 1:8.2.6-1build1 [90,3 kB]
- Receiving from :5 http://archive.ubuntu.com/ubuntu noble/universe amd64 guile-3.0-libs amd64 3.0.9-1build2 [7 630 kB]
- 55% [5 guile-3.0-libs 5 682 kB/7 630 kB 74%] 383 kB/s 11s

4. Configuration de Postfix

Lors de l'installation :

- Choisir Internet Site
- Définir System mail name : localhost



5. Configuration du relais SMTP Gmail

Nous avons éditer le fichier de configuration `/etc/postfix/main.cf` :

Au moyen de la commande : `sudo nano /etc/postfix/main.cf`

Puis ajouter les lignes suivantes :

```
relayhost = [smtp.gmail.com]:587
```

```
smtp_use_tls = yes
```

```
smtp_sasl_auth_enable = yes
```

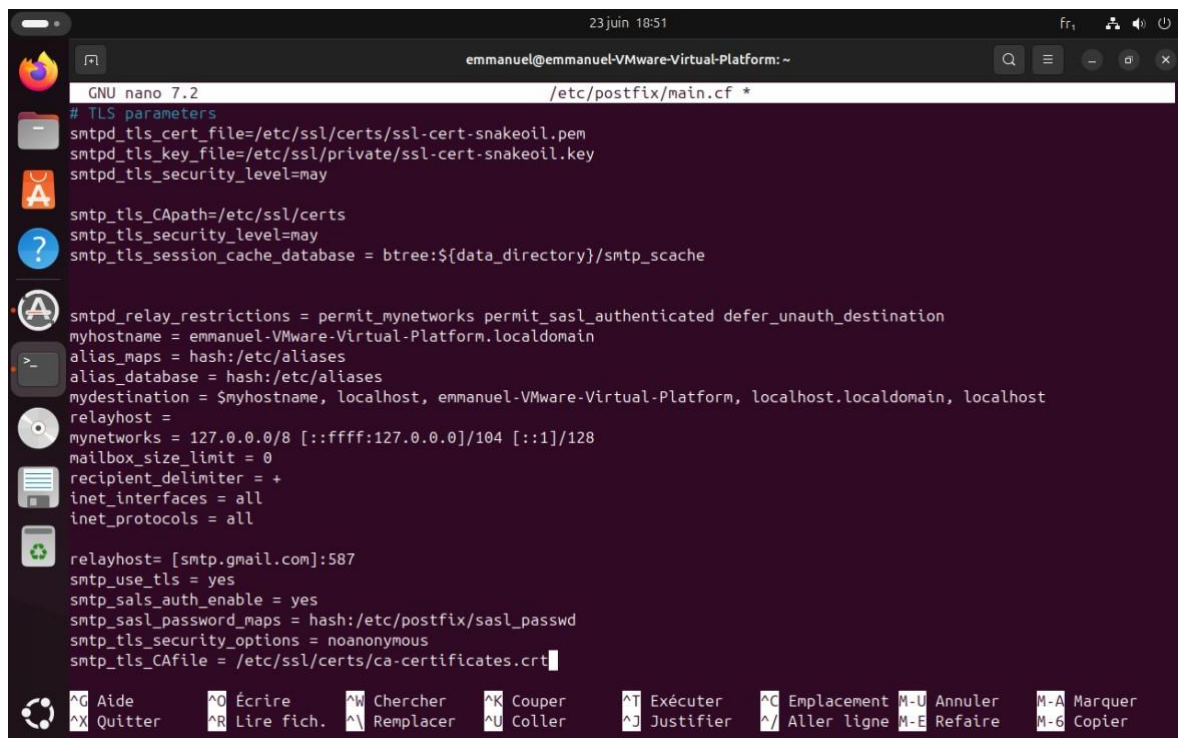
```
smtp_sasl_password_maps =
```

```
hash:/etc/postfix/sasl_passwd
```

```
smtp_sasl_security_options =
```

```
noanonymous smtp_tls_CAfile =
```

```
/etc/ssl/certs/ca-certificates.crt
```



The screenshot shows a terminal window titled "emmanuel@emmanuel-VMware-Virtual-Platform: ~" with a timestamp of "23 juin 18:51". The window displays the configuration file `/etc/postfix/main.cf` being edited with GNU nano 7.2. The configuration includes TLS parameters, relay host settings, and network configurations. The specific lines added for Gmail SMTP are highlighted in the original image.

```
GNU nano 7.2 /etc/postfix/main.cf *
# TLS parameters
smtpd_tls_cert_file=/etc/ssl/certs/ssl-cert-snakeoil.pem
smtpd_tls_key_file=/etc/ssl/private/ssl-cert-snakeoil.key
smtpd_tls_security_level=may

smtp_tls_CApath=/etc/ssl/certs
smtp_tls_security_level=may
smtp_tls_session_cache_database = btree:${data_directory}/smtp_scache

smtpd_relay_restrictions = permit_mynetworks permit_sasl_authenticated defer_unauth_destination
myhostname = emmanuel-VMware-Virtual-Platform.localdomain
alias_maps = hash:/etc/aliases
alias_database = hash:/etc/aliases
mydestination = $myhostname, localhost, emmanuel-VMware-Virtual-Platform, localhost.localdomain, localhost
relayhost =
mynetworks = 127.0.0.0/8 [::ffff:127.0.0.0]/104 [::1]/128
mailbox_size_limit = 0
recipient_delimiter = +
inet_interfaces = all
inet_protocols = all

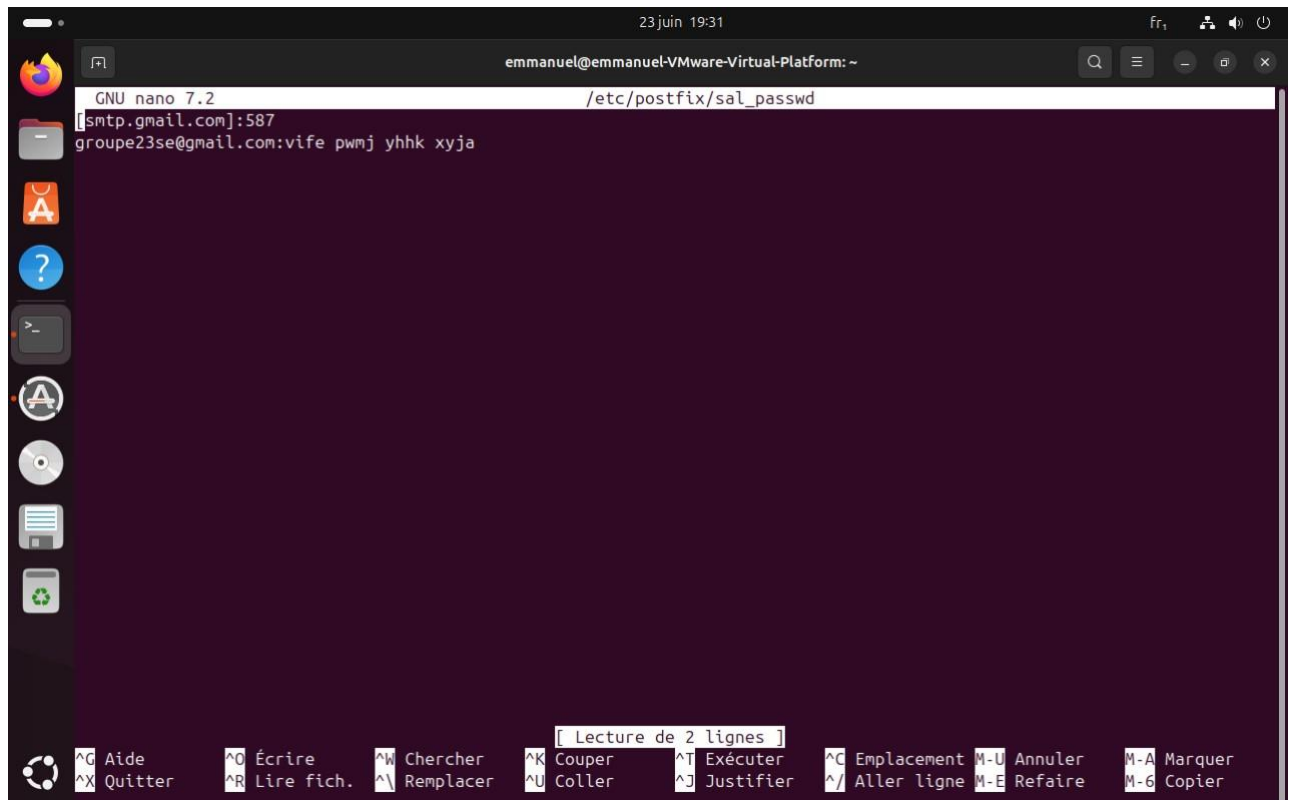
relayhost= [smtp.gmail.com]:587
smtp_use_tls = yes
smtp_sasl_auth_enable = yes
smtp_sasl_password_maps = hash:/etc/postfix/sasl_passwd
smtp_sasl_security_options = noanonymous
smtp_tls_CAfile = /etc/ssl/certs/ca-certificates.crt
```

6. Création du fichier d'identifiants SMTP

Nous avons créé le fichier `/etc/postfix/sasl_passwd` au moyen de la commande `sudo nano /etc/postfix/sasl_passwd`

Et nous y avons placé les identifiants de notre compte google :

```
[smtp.gmail.com]:587 groupe23se@gmail.com:vifepwmjyhkhkxyja
```



7. Génération et sécurisation de la map

Commande :

```
sudo postmap  
/etc/postfix/sasl_passwd  
sudo chmod 600  
/etc/postfix/sasl_passwd  
*
```

8. Redémarrage de Postfix

Commande :

```
sudo service postfix restart
```

9. Test de connexion SMTP Gmail

Commande :

```
telnet smtp.gmail.com 587
```

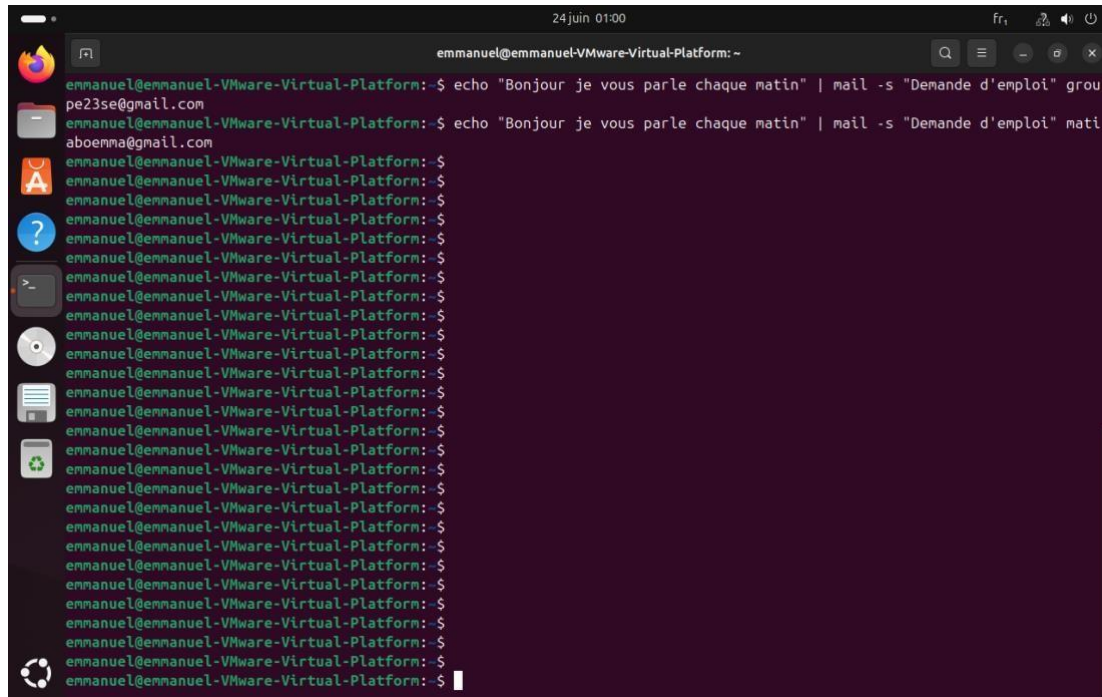
Résultat attendu :

```
Connected to smtp.gmail.com.  
220 smtp.gmail.com ESMTP ...
```


10. Envoi d'un mail de test

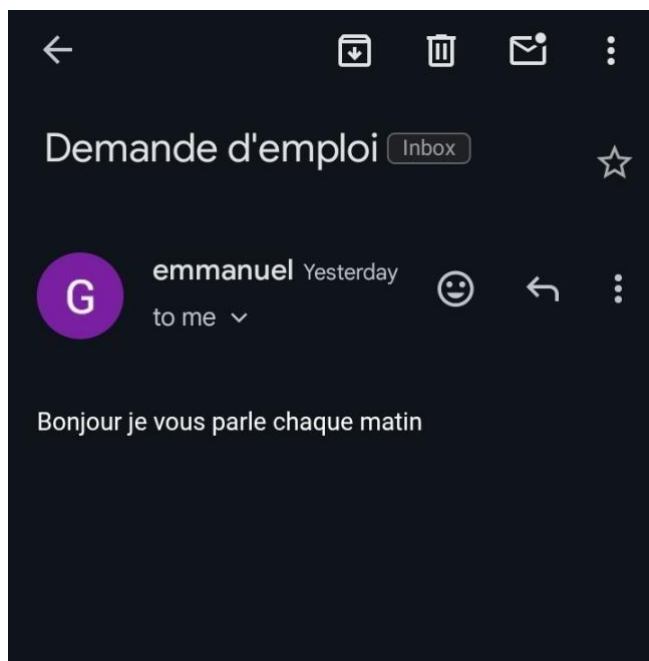
Commande :

```
echo "Bonjour je vous parle chaque matin" | mail -s "Alerte serveur Ubuntu" groupe23se@gmail.com
```



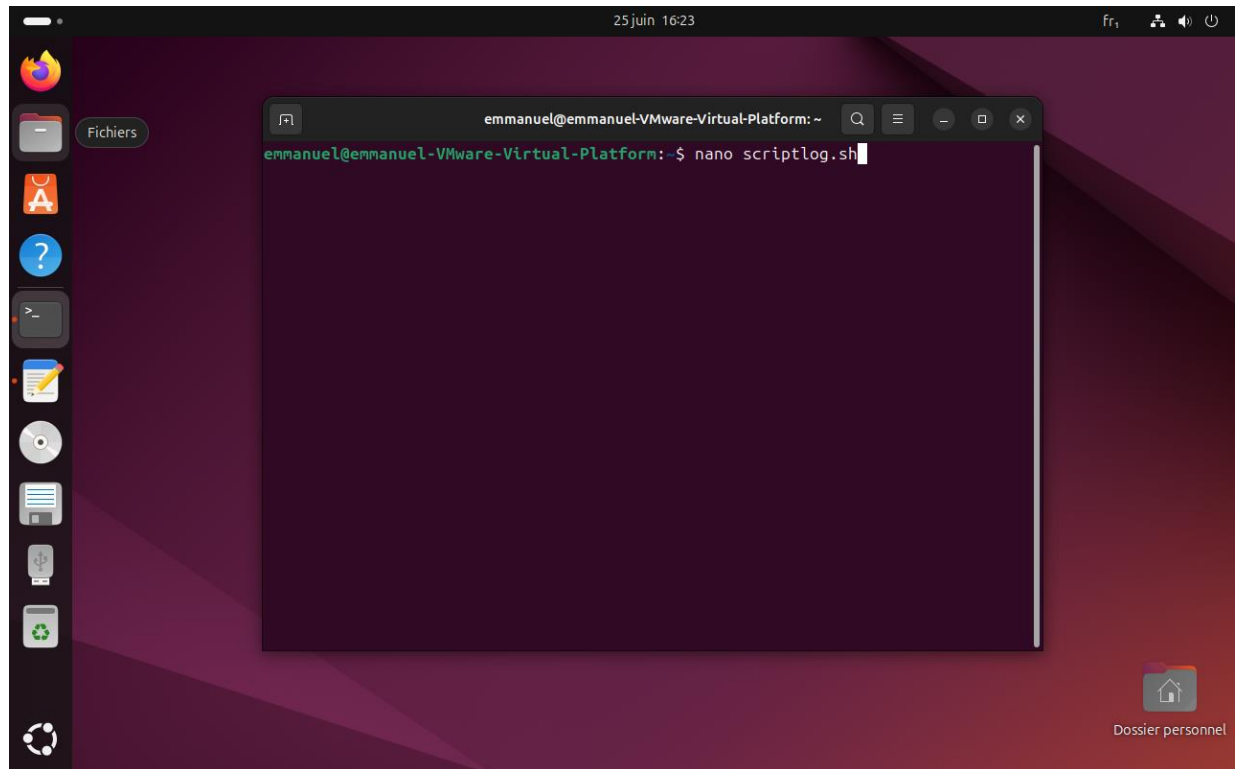
A terminal window titled 'emmanuel@emmanuel-VMware-Virtual-Platform: -' with a dark purple background. The terminal shows the execution of the command `echo "Bonjour je vous parle chaque matin" | mail -s "Demande d'emploi" groupe23se@gmail.com`. The prompt `emmanuel@emmanuel-VMware-Virtual-Platform:~$` is visible at the bottom, followed by a cursor. The left sidebar of the terminal window shows various application icons.

11. Résultat dans le mail de l'administrateur

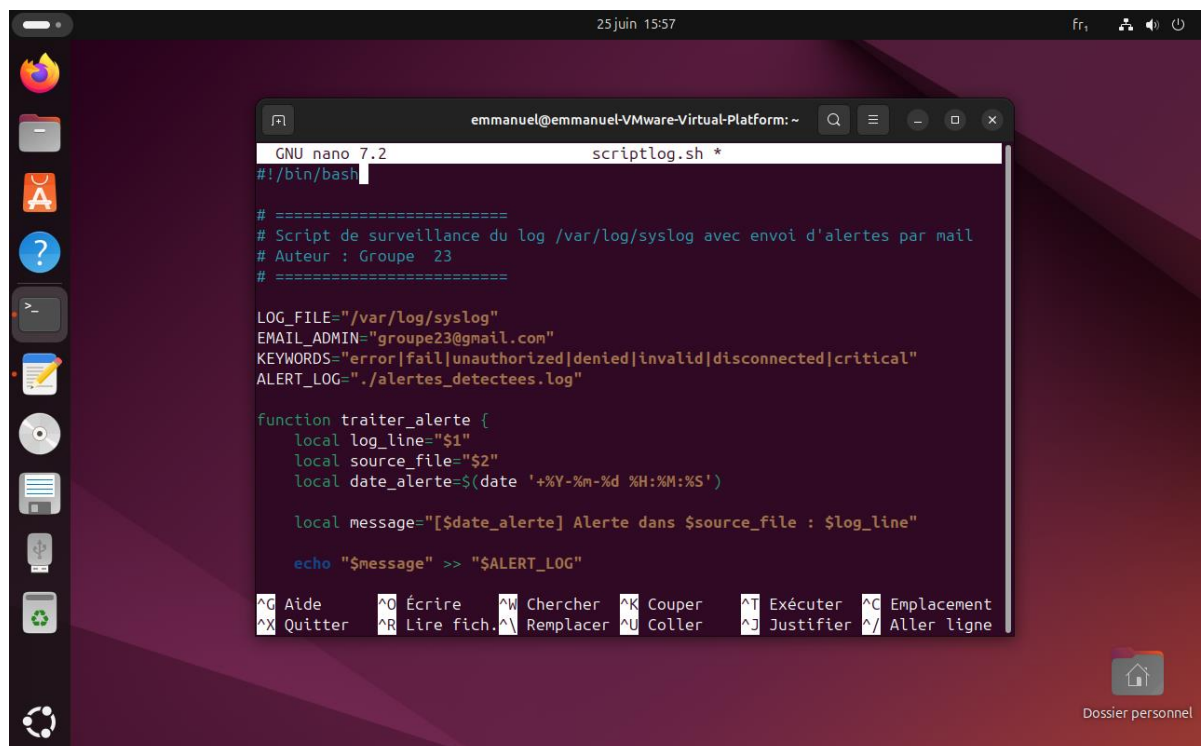


IV. Écriture du script de surveillance de log avec envoi du mail

Nous avons commencé par créer le fichier qui contiendra notre script au moyen de la commande : `nano scriptlog.sh`



Puis nous avons procédé à l'écriture du script :



Voici le script complet :

```
#!/bin/bash

# =====
# Script de surveillance du log /var/log/syslog avec envoi
# d'alertes par mail
# Auteur : Groupe 23
# =====

LOG_FILE="/var/log/syslog"
EMAIL_ADMIN="groupe23se@gmail.com"
KEYWORDS="error|fail|unauthorized|denied|invalid|disconnected|critical"
ALERT_LOG="./alertes_detectees.log"

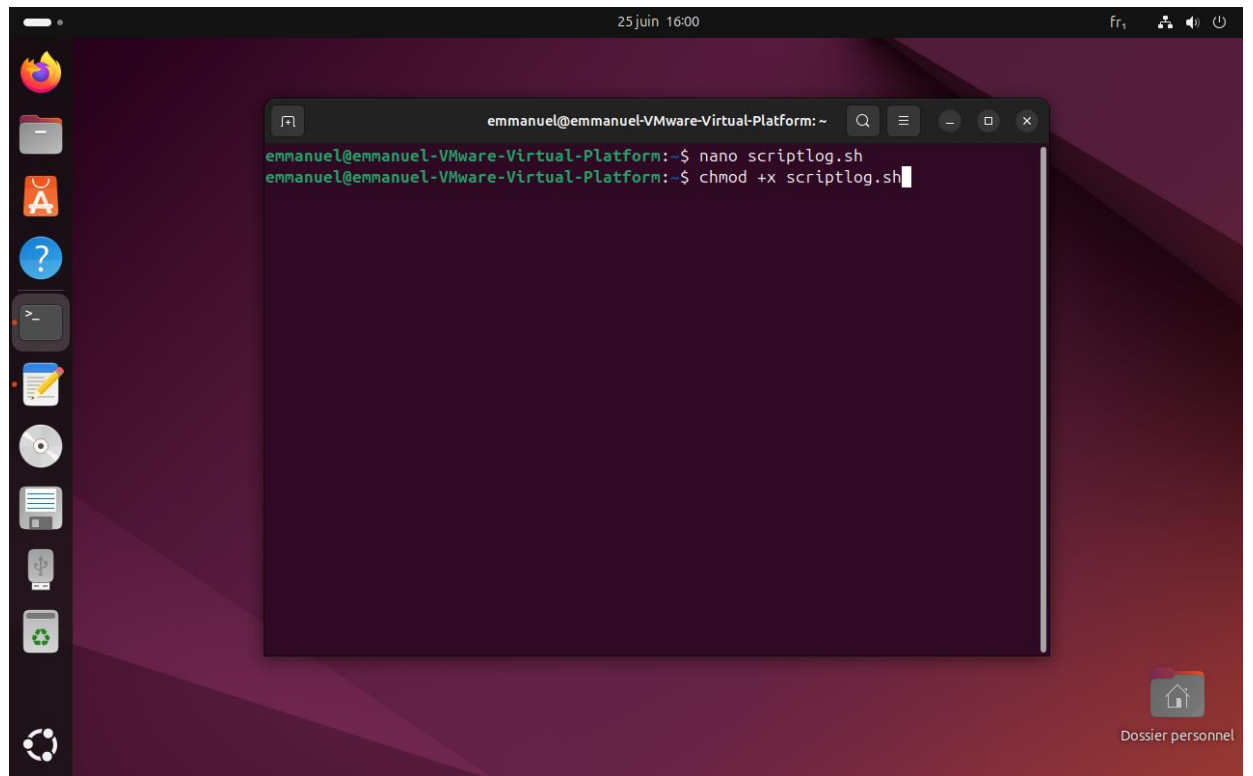
function traiter_alerte {
    local log_line="$1"
    local source_file="$2"
    local date_alerte=$(date '+%Y-%m-%d %H:%M:%S')

    local message="[${date_alerte}] Alerte dans $source_file :
$log_line"

    echo "$message" >> "$ALERT_LOG"
    echo "$message" | mail -s "Alerte Système Détectée"
"$EMAIL_ADMIN"
}

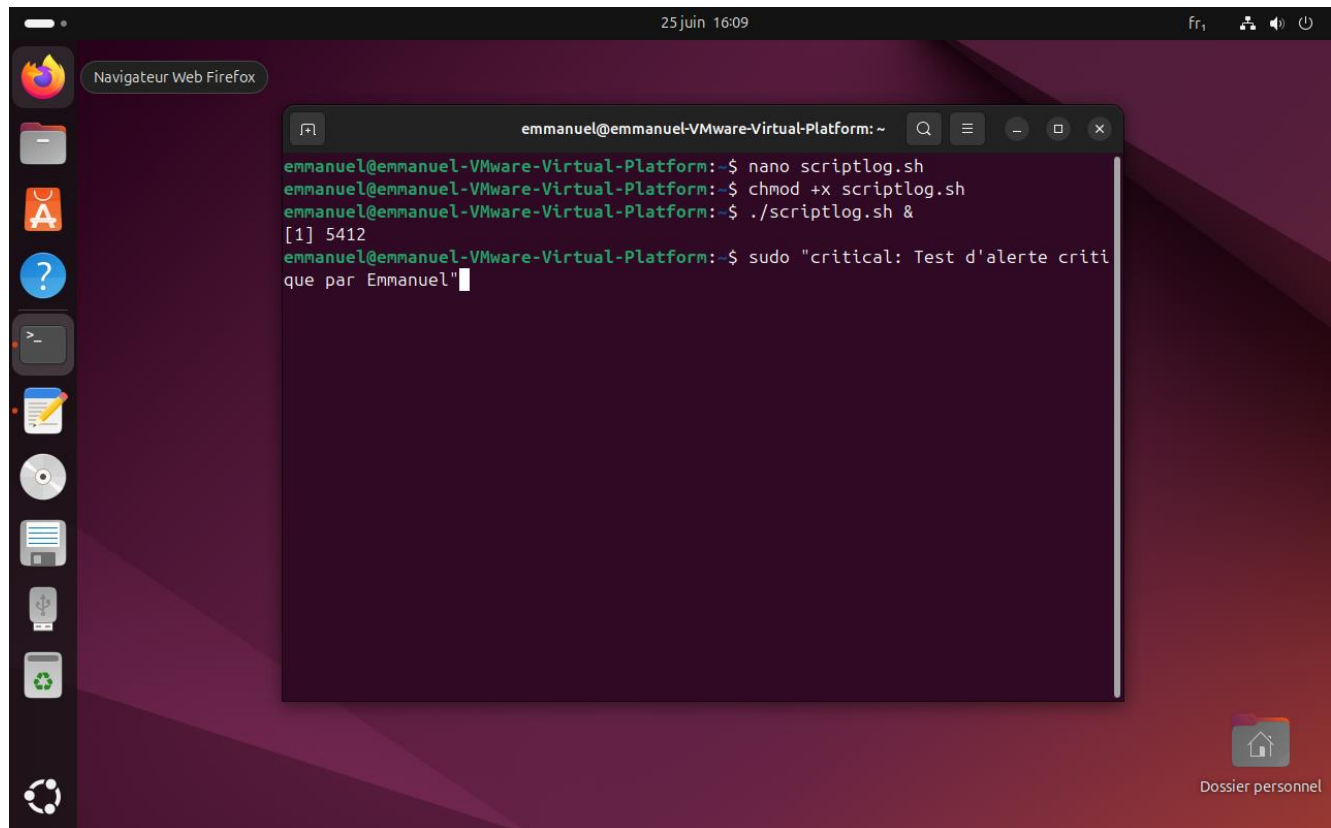
if [ -f "$LOG_FILE" ]; then
    tail -Fn0 "$LOG_FILE" | while read line; do
        echo "$line" | egrep -i "$KEYWORDS" &> /dev/null
        if [ $? = 0 ]; then
            traiter_alerte "$line" "$LOG_FILE"
        fi
    done
else
    exit 1
fi
```

Après l'écriture du script, nous l'avons rendu exécutable au moyen de la commande : `chmod +x scriptlog.sh`

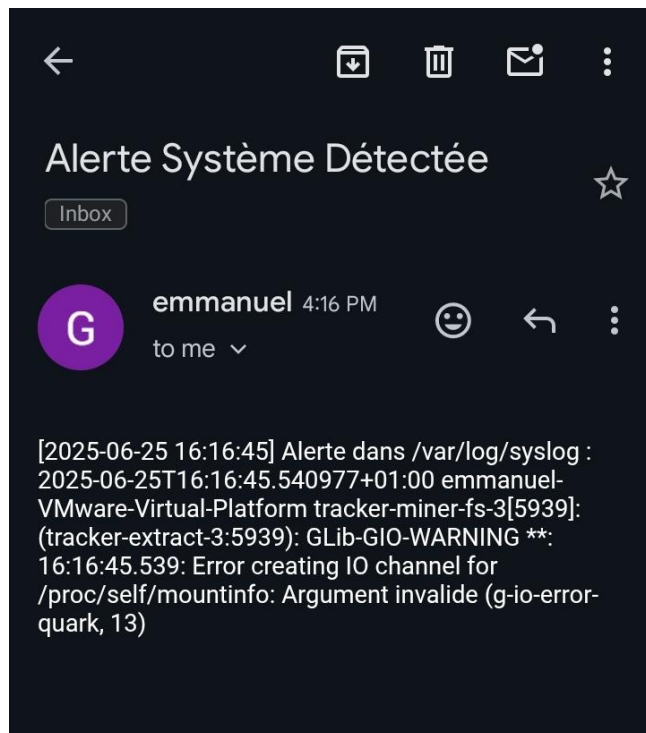


V. Test du système

Nous avons testé le script en simulant une erreur système :



Voici la boîte mail de l'administrateur :



VI. CONCLUSION

Ce projet nous a permis de mettre en pratique plusieurs notions importantes vues en cours, notamment la manipulation des fichiers système, l'écriture de scripts Bash et l'automatisation de tâches. En réalisant un script capable de surveiller les logs système et d'envoyer des alertes par mail en cas d'erreur ou d'événement suspect, nous avons mieux compris le rôle essentiel que jouent les fichiers de logs dans la gestion et la sécurité d'un système Linux. Nous avons également appris à utiliser des outils comme `grep` et `mail`, qui sont très utiles dans le quotidien d'un administrateur système.

Parmi les difficultés rencontrées, la principale a été de bien filtrer les messages réellement pertinents dans les logs. Ces fichiers contenant énormément d'informations, il n'est pas toujours simple de distinguer ce qui est critique de ce qui ne l'est pas. Nous avons donc dû effectuer plusieurs essais pour affiner les expressions à surveiller et limiter les faux positifs ou, au contraire, éviter de rater des alertes importantes. Une autre difficulté a concerné l'envoi de mails via le script : cela dépend fortement de la configuration du système, et nécessite parfois l'installation et le paramétrage de certains paquets supplémentaires.

Concernant les pistes d'amélioration, notre script pourrait être enrichi par une interface de configuration plus conviviale, ou même être réécrit dans un langage comme Python pour faciliter sa maintenance et son déploiement sur plusieurs serveurs. On pourrait également y ajouter un système de journalisation propre, pour garder une trace des alertes envoyées, ou intégrer d'autres moyens de notification comme les SMS ou les applications de messagerie (par exemple Slack ou Telegram).

De manière générale, nous avons trouvé ce projet très formateur et intéressant. Il allie théorie et pratique, et nous a permis de mieux comprendre comment automatiser certaines tâches essentielles dans la gestion d'un système. Ce travail nous a aussi permis de développer notre esprit d'analyse, notre rigueur dans l'écriture de scripts, et surtout notre capacité à collaborer pour résoudre des problèmes concrets. Même si le projet peut sembler simple au départ, il touche à des aspects clés de la sécurité et de la fiabilité des systèmes informatiques, ce qui lui donne une réelle valeur.

I. BIBLIOGRAPHIE :

- Cooper, M. (2011). *Guide avancé d'écriture des scripts Bash*. The Linux Documentation Project.
- KASENGEDIA MOTUMBE pierre, cours de systèmes d'exploitation, L2 LMD Sciences/UNIKIN, 2012-2013
- http://aral.iut-rodez.fr/fr/sanchis/enseignement/IntroProgBash_2022-06-03.pdf (PDF du livre Introduction à la programmation)

Consulté le 25 juin 2025 à 11h 20, via : <https://tldp.org/LDP/abs/html/>

II. WEBOGRAPHIE :

https://youtu.be/UuEx_JwNI2s?si=YkyaZj2GhrDfL0e5 Consulté le 25 Juin 2025 à 11h 30