Capron Dylan

Plateflop_MySQL

Installer MariaDB Server

sudo apt install maria db server
sudo mysql_secure_installation

```
dome@debian12-SI:~$ sudo mysql_secure_installation

NOTE: RUNNING ALL PARTS OF THIS SCRIPT IS RECOMMENDED FOR ALL MariaDB
      SERVERS IN PRODUCTION USE!  PLEASE READ EACH STEP CAREFULLY!

In order to log into MariaDB to secure it, we'll need the current
password for the root user. If you've just installed MariaDB, and
haven't set the root password yet, you should just press enter here.

Enter current password for root (enter for none):
OK, successfully used password, moving on...

Setting the root password or using the unix_socket ensures that nobody
can log into the MariaDB root user without the proper authorisation.

You already have your root account protected, so you can safely answer 'n'.

Switch to unix_socket authentication [Y/n] n
 ... skipping.

You already have your root account protected, so you can safely answer 'n'.

Change the root password? [Y/n] y
New password:
Re-enter new password:
Password updated successfully!
Reloading privilege tables..
 ... Success!


By default, a MariaDB installation has an anonymous user, allowing anyone
to log into MariaDB without having to have a user account created for
them.  This is intended only for testing, and to make the installation
go a bit smoother.  You should remove them before moving into a
production environment.

Remove anonymous users? [Y/n] n
 ... skipping.

Normally, root should only be allowed to connect from 'localhost'.  This
ensures that someone cannot guess at the root password from the network.
```

```
Disallow root login remotely? [Y/n] n
 ... skipping.

By default, MariaDB comes with a database named 'test' that anyone can
access.  This is also intended only for testing, and should be removed
before moving into a production environment.

Remove test database and access to it? [Y/n] n
 ... skipping.

Reloading the privilege tables will ensure that all changes made so far
will take effect immediately.

Reload privilege tables now? [Y/n] n
 ... skipping.

Cleaning up...

All done!  If you've completed all of the above steps, your MariaDB
installation should now be secure.

Thanks for using MariaDB!
```

sudo systemctl status mariadb

```
dome@debian12-SI:~$ sudo systemctl status mariadb
● mariadb.service - MariaDB 10.11.6 database server
     Loaded: loaded (/lib/systemd/system/mariadb.service; enabled; preset: enabled)
     Active: active (running) since Thu 2024-09-19 10:49:41 CEST; 4min 30s ago
       Docs: man:mariadbd(8)
             https://mariadb.com/kb/en/library/systemd/
   Main PID: 2354 (mariadbd)
     Status: "Taking your SQL requests now..."
      Tasks: 9 (limit: 2264)
     Memory: 208.8M
        CPU: 859ms
     CGroup: /system.slice/mariadb.service
             └─2354 /usr/sbin/mariadbd

sept. 19 10:49:41 debian12-SI mariadbd[2354]: 2024-09-19 10:49:41 0 [Note] InnoDB: log sequence number 45598; transaction id 14
sept. 19 10:49:41 debian12-SI mariadbd[2354]: 2024-09-19 10:49:41 0 [Note] InnoDB: Loading buffer pool(s) from /var/lib/mysql/ib_buffer_pool
sept. 19 10:49:41 debian12-SI mariadbd[2354]: 2024-09-19 10:49:41 0 [Note] Plugin 'FEEDBACK' is disabled.
sept. 19 10:49:41 debian12-SI mariadbd[2354]: 2024-09-19 10:49:41 0 [Warning] You need to use --log-bin to make --expire-logs-days or --binlog-expire-logs-s
sept. 19 10:49:41 debian12-SI mariadbd[2354]: 2024-09-19 10:49:41 0 [Note] Server socket created on IP: '127.0.0.1'.
sept. 19 10:49:41 debian12-SI mariadbd[2354]: 2024-09-19 10:49:41 0 [Note] InnoDB: Buffer pool(s) load completed at 240919 10:49:41
sept. 19 10:49:41 debian12-SI mariadbd[2354]: 2024-09-19 10:49:41 0 [Note] /usr/sbin/mariadbd: ready for connections.
sept. 19 10:49:41 debian12-SI mariadbd[2354]: Version: '10.11.6-MariaDB-0+deb12u1'  socket: '/run/mysqld/mysqld.sock'  port: 3306  Debian 12
sept. 19 10:49:41 debian12-SI systemd[1]: Started mariadb.service - MariaDB 10.11.6 database server.
sept. 19 10:49:41 debian12-SI /etc/mysql/debian-start[2385]: Triggering myisam-recover for all MyISAM tables and aria-recover for all Aria tables
lines 1-23/23 (END)
[1]+  Stoppé                 sudo systemctl status mariadb
```

sudo mysql -u root -p

```
dome@debian12-SI:~$ sudo mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 34
Server version: 10.11.6-MariaDB-0+deb12u1 Debian 12

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
```

```
MariaDB [(none)]> CREATE DATABASE plateflop;
Query OK, 1 row affected (0,000 sec)
```

```
MariaDB [(none)]> CREATE USER 'dylan'@'%' IDENTIFIED BY 'quiqui06';
ERROR 1396 (HY000): Operation CREATE USER failed for 'dylan'@'%'
MariaDB [(none)]> CREATE USER 'dylan'@'localhost' IDENTIFIED BY 'quiqui06';
Query OK, 0 rows affected (0,007 sec)
```

```
MariaDB [(none)]> SELECT user, host FROM mysql.user;
+-------------+-----------+
| User        | Host      |
+-------------+-----------+
| dylan       | %         |
| mariadb.sys | localhost |
| mysql       | localhost |
| root        | localhost |
+-------------+-----------+
4 rows in set (0,001 sec)

MariaDB [(none)]> exit
Bye
```

Ajout de la clé SSH

```
21  ssh-keygen -t rsa -b 4096 -f ~/.ssh/id_rsa
22  ls -l ~/.ssh
```

```
35  su -
36  chmod 600 ~/.ssh/id_rsa
```

```
40  eval $(ssh-agent)
41  ssh-add ~/.ssh/id_rsa
42  ssh-add -l
43  ssh-copy-id monitor@srv-sql
```

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

```
monitor@srv-sql:~$ cd ~/.ssh
monitor@srv-sql:~/.ssh$ ls
authorized_keys
monitor@srv-sql:~/.ssh$ cat authorized_keys
ssh-ed25519 AAAAC3NzaC1l2DI1NTE5AAAAIO/vF+5KraeR+nJVftBiCjYMI//D7zlaNBRScAPfUXue thierry@debianSid
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACAQCxqAbPjRefx+lrjlkR3kynslBk6IEYspNJEqM9BZG3iisAiD3NGKwnJJHH7JPVDvTw3M0qgNafD/R9qwS9hI5if3V+tw14vvZU5o6FQ3ySMZXRM58X1URQrz
9SoWvDYEY4glHfeoxk2qp0VKX1XUmqTxEP3Ae8+Y1oI+Ygcr7m//ONtJGXQUFDEQhCoc3W5A3g+beLkCzjL7/bVj3A4HiIJT+AclvmG6s0LJlOIjkCfiR2DJB2AIgHWOAc0Ttlhm8T0LcFjofpipc5s/wW3TnF
+Zd4CdhGi0pCKqoL84he1y5Gg3FfjrXqUWQRr5oKPvpa7Xgd2dxbbBEIDYH3w+W1ypQaIv42XNXOeHc9aotw2TVN17BSUC1xqtLJH9t3Bz4AOfC3F+2biLBAfdJ+uZr7jfXoVjhlPT+O/CQGnK8JfVVQ2+pAVL
Gywresvh8VD5h2ntY5Hqe60ZVvfVlXPlFgebbRg/0+2c0qNJsq1GlByVQv182HKP5s216s6kVD+FbLGA3M+jOld9hxMquLcjrc4fHdM6zf0jztKYBPSdCiIk39YXBpJlmLlSiRruWrr4rVY41NRad3wA0oTa3J
Lyz3as59buLPComxYTE0fhfRWqBjGy2lwS+hFrtvPL4rFldGsToM8YYz76Na/QLw294JtJ5VEuRLj/ExHQ== dylan@cli-dylan
```

Script ssh_login.py

```
GNU nano 7.2                                                    ssh_login.py
#!/usr/bin/env python3

import paramiko
from getpass import getpass  # Importation de getpass pour saisir le mot de passe en toute sécurité

def ssh_login(server_ip, username, password, command):

    try:

        client = paramiko.SSHClient()

        client.set_missing_host_key_policy(paramiko.AutoAddPolicy())

        client.connect(server_ip, username=username, password=password)


        stdin, stdout, stderr = client.exec_command(command)

        print("Output of '{}' :" .format(command))

        for line in stdout:

            print(line.strip())

        client.close()

    except Exception as e:

        print("Error:", e)

server_ip = "192.168.10.152"

username = "monitor"

password = getpass("Entrez le mot de passe SSH :")

command  = "df"


ssh_login(server_ip, username, password, command)
```

Test du script

```
root@cli-dylan:/home/dylan# ./ssh_login.py
Entrez le mot de passe SSH :
Output of 'df' :
Sys. de fichiers blocs de 1K Utilisé Disponible Uti% Monté sur
udev                  984000       0     984000   0% /dev
tmpfs                 201444     568     200876   1% /run
/dev/sda1           15421320 2922328   11693824  20% /
tmpfs                1007204       0    1007204   0% /dev/shm
tmpfs                   5120       0       5120   0% /run/lock
tmpfs                 201440       0     201440   0% /run/user/1000
root@cli-dylan:/home/dylan#
```

La connexion fonctionne.

Script ssh_login_sudo.py

```
#!/usr/bin/env python3

import paramiko
from getpass import getpass

def ssh_execute_sudo_command(hostname, username, ssh_password, sudo_password, command):
    # Création d'un objet SSHClient
    ssh = paramiko.SSHClient()
    ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())

    try:
        # Connexion SSH avec le mot de passe SSH
        ssh.connect(hostname, username=username, password=ssh_password)

        # Préparation de la commande avec sudo
        # Utilisation de -S pour lire le mot de passe à partir de l'entrée standard
        sudo_command = f"echo {sudo_password} | sudo -S {command}"

        # Exécution de la commande sudo
        stdin, stdout, stderr = ssh.exec_command(sudo_command)

        # Lecture de la sortie et des erreurs
        output = stdout.read().decode('utf-8')
        error_output = stderr.read().decode('utf-8')

        # Affichage de la sortie de la commande
        if output:
            print(f"Résultat de la commande '{command}' avec sudo:\n{output}")
        if error_output:
            print(f"Erreur de la commande '{command}' avec sudo:\n{error_output}")

    except paramiko.AuthenticationException:
        print("Erreur d'authentification. Veuillez vérifier vos informations d'identification.")
    except paramiko.SSHException as e:
        print(f"Erreur SSH: {e}")
    except Exception as e:
        print(f"Erreur inattendue: {e}")
    finally:
        # Fermeture de la connexion SSH
        ssh.close()

if __name__ == "__main__":
    # Informations d'authentification pour le serveur SSH
    hostname = '192.168.10.185'  # Remplacez par l'adresse IP de votre serveur
    username = 'monitor'  # Remplacez par le nom d'utilisateur SSH

    # Saisie sécurisée du mot de passe SSH
    ssh_password = getpass("Entrez le mot de passe SSH : ")

    # Saisie sécurisée du mot de passe sudo
    sudo_password = getpass("Entrez le mot de passe sudo : ")

    # Commande shell à exécuter avec sudo
    command_to_execute = "apt update"  # Exemple de commande à exécuter avec sudo, vous pouvez la remplacer

    # Appel de la fonction pour exécuter la commande via SSH avec sudo
    ssh_execute_sudo_command(hostname, username, ssh_password, sudo_password, command_to_execute)
```

Test du script

```
root@cli-dylan:/home/dylan# ./ssh_login_sudo.py
Entrez le mot de passe SSH :
Entrez le mot de passe sudo :
Résultat de la commande 'apt update' avec sudo:
Atteint :1 http://security.debian.org/debian-security bookworm-security InRelease
Atteint :2 http://deb.debian.org/debian bookworm InRelease
Atteint :3 http://deb.debian.org/debian bookworm-updates InRelease
Lecture des listes de paquets…
Construction de l'arbre des dépendances…
Lecture des informations d'état…
Tous les paquets sont à jour.

Erreur de la commande 'apt update' avec sudo:
[sudo] Mot de passe de monitor :
WARNING: apt does not have a stable CLI interface. Use with caution in scripts.
```

La connexion et la commande sudo fonctionnent.

Script ssh_mysql.py

```
#!/usr/bin/env python3

import mysql.connector
import sys

def test_mariadb_connection(host, port, username, password, database):
    try:
        # Connexion à la base de données MariaDB
        connection = mysql.connector.connect(
            host=host,
            port=port,
            user=username,
            password=password,
            database=database
        )

        cursor = connection.cursor()
        cursor.execute("SELECT VERSION()")

        # Récupérer la version du serveur
        server_version = cursor.fetchone()[0]
        print("Connected to MariaDB Server")
        print("Server version:", server_version)

        cursor.close()
        connection.close()

    except mysql.connector.Error as e:
        print(f"Error connecting to MariaDB: {e}")
        sys.exit(1)

if __name__ == "__main__":
    host = "192.168.10.152"  # Remplace par l'adresse IP de ton serveur MariaDB
    port = 3306  # Le port par défaut de MariaDB
    username = "dylan"  # Remplace par ton nom d'utilisateur
    password = "x"  # Remplace par ton mot de passe
    database = "plateflop"  # Remplace par le nom de ta base de données

    test_mariadb_connection(host, port, username, password, database)
```

Test du script

```
dylan@cli-dylan:~$ ./ssh_mysql.py
Connected to MariaDB Server
Server version: 10.11.6-MariaDB-0+deb12u1
```

Connexion au serveur MySQL réussie

Script ssh_mysql_error.py

```
#!/usr/bin/env python3

import paramiko
import mysql.connector
import sys
from datetime import datetime

def ssh_connect(hostname, port, username, password):
    try:
        # Créer une instance SSHClient
        client = paramiko.SSHClient()
        # Charger les clés
        client.load_system_host_keys()
        # Ajouter la clé de l'hôte si elle n'est pas déjà connue
        client.set_missing_host_key_policy(paramiko.AutoAddPolicy())

        # Connexion SSH
        client.connect(hostname, port, username, password)
        print("Connected to SSH server")
        return client
    except Exception as e:
        print(f"Error connecting to SSH: {e}")
        log_error_to_db("SSH Connection Error", str(e))
        sys.exit(1)

def mysql_connect(host, port, username, password, database):
    try:
        # Connexion à la base de données MySQL
        connection = mysql.connector.connect(
            host=host,
            port=port,
            user=username,
            password=password,
            database=database
        )

        cursor = connection.cursor()
        cursor.execute("SELECT VERSION()")
        server_version = cursor.fetchone()[0]
        print("Connected to MySQL Server")
        print("Server version:", server_version)

        cursor.close()
        connection.close()

    except mysql.connector.Error as e:
        print(f"Error connecting to MySQL: {e}")
        log_error_to_db("MySQL Connection Error", str(e))
        sys.exit(1)
```

```
def log_error_to_db(error_type, error_message):
    try:
        # Connexion à la base de données pour enregistrer l'erreur
        connection = mysql.connector.connect(
            host="192.168.10.152",  # Remplace par l'adresse IP ou le nom d'hôte si nécessaire
            port=3306,  # Port par défaut de MySQL
            user="dylan",  # Remplace par ton nom d'utilisateur de base de données
            password="x",  # Remplace par ton mot de passe de base de données
            database="plateflop"  # Remplace par le nom de ta base de données
        )

        cursor = connection.cursor()
        # Enregistrer l'erreur dans la table error_log
        cursor.execute("""
            INSERT INTO error_log (error_type, error_message, timestamp)
            VALUES (%s, %s, %s)
        """, (error_type, error_message, datetime.now()))

        connection.commit()
        cursor.close()
        connection.close()
        print("Error logged to database")

    except mysql.connector.Error as db_error:
        print(f"Error logging to database: {db_error}")

if __name__ == "__main__":
    ssh_host = "192.168.10.152"  # Remplace par l'adresse IP de ton serveur SSH
    ssh_port = 22  # Port SSH par défaut
    ssh_username = "your_ssh_username"  # Remplace par ton nom d'utilisateur SSH
    ssh_password = "your_ssh_password"  # Remplace par ton mot de passe SSH

    db_host = "%"  # Host de la base de données après la connexion SSH
    db_port = 3306  # Le port par défaut de MySQL
    db_username = "dylan"  # Remplace par ton nom d'utilisateur de base de données
    db_password = "x"  # Remplace par ton mot de passe de base de données
    db_name = "plateflop"  # Remplace par le nom de ta base de données

    # Connexion SSH
    ssh_client = ssh_connect(ssh_host, ssh_port, ssh_username, ssh_password)

    # Connexion MySQL
    mysql_connect(db_host, db_port, db_username, db_password, db_name)

    # Fermer la connexion SSH
    ssh_client.close()
```

Test du script

```
dylan@cli-dylan:~$ ./ssh_mysql_error.py
Error connecting to SSH: Authentication failed.
Error logged to database
```

Script ssh_serveur_mail.py

```python
#!/usr/bin/env python3

import mysql.connector

import logging

import smtplib

from email.mime.text import MIMEText


# Configuration du fichier de log

logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(levelname)s - %(message)s')


# Informations de connexion à la base de données

db_host = 'srv-sql'  # Adresse de ton serveur MySQL

db_user = 'dylan'  # Remplace par ton nom d'utilisateur MySQL

db_password = 'x'  # Remplace par ton mot de passe MySQL

db_name = 'plateflop'  # Nom de la base de données


# Informations pour l'envoi de l'email

smtp_server = 'smtp.gmail.com'

smtp_port = 587

smtp_user = 'dylan.capron@laplateforme.io'

smtp_password = 'lzxqxtizmsjgcaid'
```

```python
admin_email = 'dylan.capron@laplateforme.io'

# Récupère les logs depuis la base de données
def retrieve_error_logs():
    try:
        connection = mysql.connector.connect(
            host=db_host,
            user=db_user,
            password=db_password,
            database=db_name
        )
        cursor = connection.cursor()

        # Exécuter une requête pour récupérer le contenu de error.log
        cursor.execute("SELECT id, error_type, error_message, timestamp FROM error_log")  # Remplace log_content par le nom de la colonne
        logs = cursor.fetchall()

        cursor.close()
```

```python
        connection.close()

        # Retourne les logs sous forme de chaîne

        return "\n".join([f"{log[0]} | {log[1]} | {log[2]} | {log[3]}" for log in logs]) if logs else "Aucun log trouvé."

    except mysql.connector.Error as e:
        logging.error(f"Erreur de connexion à MySQL: {e}")
        return None

# Envoie un email à l'administrateur
def send_email(subject, body):
    msg = MIMEText(body)
    msg['Subject'] = subject
    msg['From'] = smtp_user
    msg['To'] = admin_email
```

```python
    try:
        server = smtplib.SMTP(smtp_server, smtp_port)
        server.starttls()
        server.login(smtp_user, smtp_password)
        server.sendmail(smtp_user, admin_email, msg.as_string())
        server.quit()
        logging.info(f"Email envoyé à {admin_email}")
    except Exception as e:
        logging.error(f"Erreur lors de l'envoi de l'email : {str(e)}")

f __name__ == "__main__":
    # Récupérer les logs
    logs = retrieve_error_logs()

    if logs:
        # Envoyer un email avec le contenu des logs
        send_email("Rapport de logs error.log", logs)
    else:
        logging.error("Impossible de récupérer les logs.")
```

Test du script

```
dylan@cli-dylan:~$ ./ssh_serveur_mail.py
2024-09-25 13:46:55,848 - INFO - Email envoyé à dylan.capron@laplateforme.io
```

dylan.capron@laplateforme.io 13:46 (il y a 15 minutes)
À moi

1 | SSH Connection Error | Authentication failed. | 2024-09-25 11:04:47
2 | SSH Connection Error | Authentication failed. | 2024-09-25 11:06:47
3 | SSH Connection Error | [Errno 2] No such file or directory: '/.ssh/id_rsa.pub' | 2024-09-25 11:20:09
4 | SSH Connection Error | Authentication failed. | 2024-09-25 11:22:52
5 | SSH Connection Error | Authentication failed. | 2024-09-25 11:28:38
6 | FTP Error | 2024-09-24 10:40:28,152 srv-ftp proftpd[20189] srv-ftp (cli-amandine.homelab.lan[192.168.10.101]): USER monitor (Login failed): Incorrect password | 2024-09-25 11:30:49
7 | FTP Error | 2024-09-24 10:49:32,191 srv-ftp proftpd[20547] srv-ftp (cli-amandine.homelab.lan[192.168.10.101]): USER monitor (Login failed): Incorrect password | 2024-09-25 11:30:49
8 | FTP Error | 2024-09-25 07:55:32,083 srv-ftp proftpd[63802] srv-ftp (cli-amandine.homelab.lan[192.168.10.101]): USER monitor (Login failed): Incorrect password | 2024-09-25 11:30:49
9 | SSH Connection Error | Authentication failed. | 2024-09-25 11:33:56
10 | FTP Error | 2024-09-24 10:40:28,152 srv-ftp proftpd[20189] srv-ftp (cli-amandine.homelab.lan[192.168.10.101]): USER monitor (Login failed): Incorrect password | 2024-09-25 12:00:02
11 | FTP Error | 2024-09-24 10:49:32,191 srv-ftp proftpd[20547] srv-ftp (cli-amandine.homelab.lan[192.168.10.101]): USER monitor (Login failed): Incorrect password | 2024-09-25 12:00:02
12 | FTP Error | 2024-09-25 07:55:32,083 srv-ftp proftpd[63802] srv-ftp (cli-amandine.homelab.lan[192.168.10.101]): USER monitor (Login failed): Incorrect password | 2024-09-25 12:00:02

Le script fonctionne.

Script ssh_cron_backup.py

```
#!/usr/bin/env python3
import os
import subprocess
import time
from datetime import datetime

# Informations de connexion MySQL
host = "192.168.10.152"
port = 3306
user = "dylan"
password = "x"
database = "plateflop"

# Répertoire de sauvegarde (dossier lié à ton home)
backup_dir = os.path.expanduser("~/backups_mysql")

# Nombre maximum de sauvegardes à conserver
max_backups = 7

# Fonction pour créer le répertoire de sauvegarde s'il n'existe pas
def create_backup_dir():
    if not os.path.exists(backup_dir):
        os.makedirs(backup_dir)

# Fonction pour effectuer la sauvegarde
def perform_backup():
    timestamp = datetime.now().strftime('%Y%m%d_%H%M%S')
    backup_file = os.path.join(backup_dir, f"backup_{database}_{timestamp}.sql")

    dump_command = [
        "mysqldump",
        f"--host={host}",
        f"--port={port}",
        f"--user={user}",
        f"--password={password}",
        database
    ]

    with open(backup_file, 'w') as f:
        result = subprocess.run(dump_command, stdout=f)
        if result.returncode == 0:
            print(f"Sauvegarde réussie : {backup_file}")
        else:
            print("Erreur lors de la sauvegarde.")

# Fonction pour supprimer les anciennes sauvegardes (conserver uniquement les 7 dernières)
def cleanup_old_backups():
    backups = sorted([f for f in os.listdir(backup_dir) if f.startswith("backup_")])
    if len(backups) > max_backups:
        backups_to_delete = backups[:len(backups) - max_backups]
        for backup in backups_to_delete:
            os.remove(os.path.join(backup_dir, backup))
            print(f"Ancienne sauvegarde supprimée : {backup}")

if __name__ == "__main__":
    # Créer le répertoire de sauvegarde s'il n'existe pas
    create_backup_dir()

    # Effectuer la sauvegarde
    perform_backup()

    # Nettoyer les anciennes sauvegardes
    cleanup_old_backups()
```

crontab -e

0 */3 * * * /usr/bin/python3 /home/dylan/ssh_cron_backup.py

Test en lançant le script toutes les minutes :

```
backup_plateflop_20240925_143302.sql  backup_plateflop_20240925_143801.sql  backup_plateflop_20240925_143956.sql
backup_plateflop_20240925_143401.sql  backup_plateflop_20240925_143949.sql
backup_plateflop_20240925_143601.sql  backup_plateflop_20240925_143952.sql
```

Script ssh_system_status.py

```
GNU nano 7.2                                    ssh_system_status.py
#!/usr/bin/env python3

import paramiko

import mysql.connector

from datetime import datetime, timedelta



# Informations de connexion MySQL

host = "192.168.10.152"

user = "dylan"

password = "x"

database = "plateflop"
```

```
# Liste des serveurs à surveiller

servers = [

    {"name": "srv-sql", "ip": "192.168.10.152"},

    # Ajoutez d'autres serveurs ici

]

#Infos Paramiko

username = "monitor"
# password = "monitor"
mySSHK   = "/home/dylan/.ssh/id_rsa.pub"
```

```
# Fonction pour se connecter à un serveur et récupérer l'état des ressources

def get_system_status(server_ip):

    try:

        # Connexion SSH

        client = paramiko.SSHClient()

        client.set_missing_host_key_policy(paramiko.AutoAddPolicy())

        client.connect(server_ip, username=username, key_filename=mySSHK, port=22)


        # Commandes pour récupérer l'état

        cpu_command = "top -bn1 | grep 'Cpu(s)' | sed 's/.*, *\([0-9.]*\)%* id.*/\1/' | awk '{print 100 - $1}'"

        ram_used_command = "free|grep Mem|awk '{print $3}'"
```

```
        ram_total_command = "free | grep Mem | awk '{print $2}'"

        disk_command = "df | grep /dev/sda1 | awk '{print $5}' | sed 's/%//'"

        stdin, stdout, stderr = client.exec_command(cpu_command)

        cpu_usage = float(stdout.read().strip())


        stdin, stdout, stderr = client.exec_command(ram_total_command)

        ram_total = int(stdout.read().strip())

        stdin, stdout, stderr = client.exec_command(ram_used_command)

        ram_used = int(stdout.read().strip())
```

```
        stdin, stdout, stderr = client.exec_command(disk_command)

        disk_usage = float(stdout.read().strip())

        client.close()

        return cpu_usage, ram_total, ram_used, disk_usage


    except Exception as e:

        print(f"Erreur lors de la récupération des données de {server_ip}: {e}")

        return None, None, None
```

```
# Fonction pour insérer les données dans la base de données

def insert_system_status(id, cpu_usage, ram_total, ram_used, disk_usage):

    try:

        connection = mysql.connector.connect(

            host=host,

            user=user,

            password=password,

            database=database

        )

        cursor = connection.cursor()

        timestamp = datetime.now()
```

```
        cursor.execute("INSERT INTO system_status (timestamp, cpu_usage, ram_total, ram_used, disk_usage)  VALUES (%s, %s, %s, %s, %
        (timestamp, cpu_usage, ram_total, ram_used, disk_usage))

        connection.commit()

        cursor.close()

        connection.close()

    except Exception as e:

        print(f"Erreur lors de l'insertion dans la base de données: {e}")
```

```
# Fonction pour supprimer les anciennes entrées (plus de 72 heures)

def cleanup_old_entries():

    try:

        connection = mysql.connector.connect(

            host=host,

            user=user,

            password=password,

            database=database

        )
```

```python
        cursor = connection.cursor()
        cutoff_time = datetime.now() - timedelta(hours=72)
        delete_query = "DELETE FROM system_status WHERE timestamp < %s"
        cursor.execute(delete_query, (cutoff_time,))
        connection.commit()
        cursor.close()
        connection.close()
    except Exception as e:
        print(f"Erreur lors du nettoyage des anciennes entrées: {e}")
```

```python
if __name__ == "__main__":
    # Nettoyer les anciennes entrées
    cleanup_old_entries()

    # Récupérer et insérer l'état système pour chaque serveur
    for server in servers:
        cpu, ram_total, ram_used, disk = get_system_status(server["ip"])
        if cpu is not None:
            insert_system_status(id, cpu, ram_total, ram_used, disk)
```

Test pour vérifier si le script fonctionne :

21	2024-10-01 10:47:02	100	2014412	792152	21.0
22	2024-10-01 10:51:36	100	2014412	804396	21.0

Script ssh_system_status_mail.py

Même script que le précédent, mais avec le rajout de ces lignes :

```python
import smtplib

from email.mime.text import MIMEText

import logging

# Configuration du logging

logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(levelname)s - %(message)s')
```

```python
#Configuration des seuils d'alerte

CPU_THRESHOLD = 70.0

RAM_THRESHOLD = 90.0

DISK_THRESHOLD = 80.0

# Configuration de l'email

SMTP_SERVER = 'smtp.gmail.com'

SMTP_PORT = 587

SMTP_USER = 'dylan.capron@laplateforme.io'

SMTP_PASSWORD = 'lzxqxtizmsjgcaid'

ADMIN_EMAIL = 'dylan.capron@laplateforme.io'
```

```python
def send_alert_email(cpu_usage, ram_usage, disk_usage):
    """Envoie un e-mail d'alerte à l'administrateur système si les seuils sont dépassés."""
    subject = "Alerte : Utilisation des ressources système élevée"
    body = f"""
    Alerte ! Les seuils d'utilisation des ressources ont été dépassés :


    - Utilisation CPU : {cpu_usage:.2f}% (seuil : {CPU_THRESHOLD}%)
    - Utilisation RAM : {ram_usage:.2f}% (seuil : {RAM_THRESHOLD}%)
    - Utilisation Disque : {disk_usage:.2f}% (seuil : {DISK_THRESHOLD}%)
    """


    msg = MIMEText(body)
    msg['Subject'] = subject
    msg['From'] = SMTP_USER
    msg['To'] = ADMIN_EMAIL
```

```python
    try:
        with smtplib.SMTP(SMTP_SERVER, SMTP_PORT) as server:
            server.starttls()
            server.login(SMTP_USER, SMTP_PASSWORD)
            server.send_message(msg)
        logging.info("E-mail d'alerte envoyé avec succès.")
    except Exception as e:
        logging.error(f"Erreur lors de l'envoi de l'e-mail : {e}")
```

```python
        if cpu is not None:
            ram_usage = (ram_used / ram_total) * 100
            insert_system_status(id, cpu, ram_total, ram_used, disk)

            # Vérification des seuils
            if cpu > CPU_THRESHOLD or ram_usage > RAM_THRESHOLD or disk > DISK_THRESHOLD:
                send_alert_email(cpu, ram_usage, disk)
```

Test pour vérifier si le script fonctionne :

23	2024-10-01 13:28:44	100	2014412	953032	21.0

Alerte : Utilisation des ressources système élevée Boîte de réception ×

dylan.capron@laplateforme.io 13:28 (il y a 29 minutes)
À moi

Alerte ! Les seuils d'utilisation des ressources ont été dépassés :

- Utilisation CPU : 100.00% (seuil : 70.0%)
- Utilisation RAM : 47.31% (seuil : 90.0%)
- Utilisation Disque : 21.00% (seuil : 80.0%)

crontab -e

*/5 * * * * /usr/bin/python3 /home/dylan/ssh_system_status_mail.py

```
# m h  dom mon dow   command
0 */3 * * * /usr/bin/python3 /home/dylan/ssh_cron_backup.py
*/5 * * * * /usr/bin/python3 /home/dylan/ssh_system_status_mail.py
```

Test pour vérifier si crontab fonctionne :

Alerte : Utilisation des ressources système élevée Boîte de réception ×

dylan.capron@laplateforme.io 14:00 (il y a 0 minute)
À moi

Alerte ! Les seuils d'utilisation des ressources ont été dépassés :

- Utilisation CPU : 100.00% (seuil : 70.0%)
- Utilisation RAM : 48.93% (seuil : 90.0%)
- Utilisation Disque : 21.00% (seuil : 80.0%)

Répondre Transférer

Alerte : Utilisation des ressources système élevée Boîte de réception ×

dylan.capron@laplateforme.io 14:05 (il y a 0 minute)
À moi

Alerte ! Les seuils d'utilisation des ressources ont été dépassés :

- Utilisation CPU : 100.00% (seuil : 70.0%)
- Utilisation RAM : 48.93% (seuil : 90.0%)
- Utilisation Disque : 21.00% (seuil : 80.0%)

Répondre Transférer

24	2024-10-01 14:00:02	100	2014412	985736	21.0
25	2024-10-01 14:05:03	100	2014412	985724	21.0

Pour faire en sorte de recevoir un mail qu'une fois par heure il faut rajouter ces lignes :

```
import os
```

```
LAST_EMAIL_FILE = '/tmp/last_email_sent.txt'
```

```
def update_last_email_time():
# Mise à jour du fichier de suivi avec la dernière heure d'envoi.
    try:
        with open(LAST_EMAIL_FILE, 'w') as f:
            print(LAST_EMAIL_FILE)
            current_time = datetime.now().isoformat()
            f.write(current_time)
            logging.info(f"Heure du dernier envoi d'email mise à jour : {current_time}")
    except Exception as e:
        logging.error(f"Erreur lors de la mise à jour du fichier de dernier email : {e}")
```

```
def can_send_email():
# Vérifie si un mail a été envoyé dans l'heure précédente.
    if not os.path.exists(LAST_EMAIL_FILE):
        return True

    with open(LAST_EMAIL_FILE, 'r') as f:
        last_email_time = datetime.fromisoformat(f.read().strip())

    return (datetime.now() - last_email_time) > timedelta(hours=1)

    can_send = (datetime.now() - last_email_time) > timedelta(hours=1)
    logging.info(f"Peut envoyer un e-mail ? {can_send}")
    return can_send
```

Résultats :

```
dylan@cli-dylan:~$ cat /tmp/last_email_sent.txt
2024-10-03T07:49:55.887846dylan@cli-dylan:~$
dylan@cli-dylan:~$ cat /tmp/last_email_sent.txt
2024-10-03T08:50:04
```

moi	Alerte : Utilisation des ressources système élevée - Alerte ! Les seuils d'utilisation des res...	08:50
moi	Alerte : Utilisation des ressources système élevée - Alerte ! Les seuils d'utilisation des res...	07:49

Script ssh_space.py

```
#!/usr/bin/env python3

from json import dumps

from httplib2 import Http

import paramiko

import logging

from datetime import datetime


# Configuration du webhook Google Chat

WEBHOOK_URL =
"https://chat.googleapis.com/v1/spaces/AAAAzB7cgLk/messages?key=AIzaSyDdI0hCZtE6vySjMm-W
EfRq3CPzqKqqsHI&token=hgCR8bvtIVhJgWPajj0a6lON2XGvoStTIfCkd7QXmF0"


# Seuils pour envoyer des alertes

CPU_THRESHOLD = 70

RAM_THRESHOLD = 80

DISK_THRESHOLD = 90

# Clé SSH et informations de connexion

username = "monitor"

mySSHK = "/home/dylan/.ssh/id_rsa.pub"


# Liste des serveurs

servers = [{"ip": "192.168.10.152"},

]

def get_system_status(server_ip):

    try:

        client = paramiko.SSHClient()

        client.set_missing_host_key_policy(paramiko.AutoAddPolicy())

        client.connect(server_ip, username=username, key_filename=mySSHK, port=22)
```

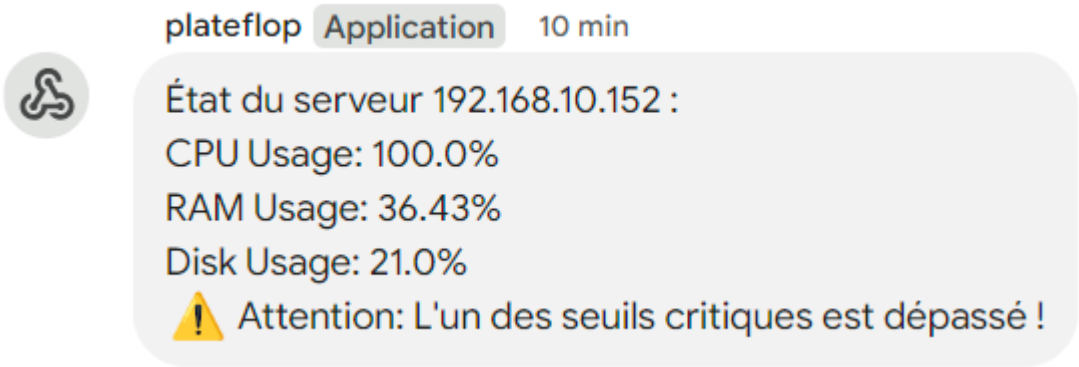
```
        # Commandes pour récupérer les données du serveur

        cpu_command = "top -bn1 | grep 'Cpu(s)' | sed 's/.*, *\\([0-9.]*\\)%* id.*/\\1/' | awk '{print 100 -
$1}'"

        ram_used_command = "free | grep Mem | awk '{print $3}'"

        ram_total_command = "free | grep Mem | awk '{print $2}'"

        disk_command = "df | grep /dev/sda1 | awk '{print $5}' | sed 's/%//'"


        stdin, stdout, stderr = client.exec_command(cpu_command)

        cpu_usage = float(stdout.read().strip())


        stdin, stdout, stderr = client.exec_command(ram_total_command)

        ram_total = int(stdout.read().strip())


        stdin, stdout, stderr = client.exec_command(ram_used_command)

        ram_used = int(stdout.read().strip())


        stdin, stdout, stderr = client.exec_command(disk_command)

        disk_usage = float(stdout.read().strip())

       client.close()

        return cpu_usage, ram_total, ram_used, disk_usage


    except Exception as e:

        logging.error(f"Erreur lors de la récupération des données du serveur {server_ip}: {e}")

        return None, None, None, None
```

```
# Fonction pour envoyer un message à Google Chat
def send_chat_message(message):
    app_message = {"text": message}
    message_headers = {"Content-Type": "application/json; charset=UTF-8"}
    http_obj = Http()
    response = http_obj.request(
        uri=WEBHOOK_URL,
        method="POST",
        headers=message_headers,
        body=dumps(app_message),
    )
    print(response)
# Fonction principale pour récupérer les données du serveur et envoyer un rapport
def main():
    for server in servers:
        cpu, ram_total, ram_used, disk = get_system_status(server["ip"])
        if cpu is not None:
            ram_usage = (ram_used / ram_total) * 100

            # Construction du message à envoyer
            message = (
                f"État du serveur {server['ip']} :\n"
                f"CPU Usage: {cpu}%\n"
                f"RAM Usage: {ram_usage:.2f}%\n"
                f"Disk Usage: {disk}%\n"
            )
```

```
# Vérification des seuils

        if cpu > CPU_THRESHOLD or ram_usage > RAM_THRESHOLD or disk > DISK_THRESHOLD:

            message += "⚠️ Attention: L'un des seuils critiques est dépassé !\n"


        # Envoi du message au chat

        send_chat_message(message)


if __name__ == "__main__":

    main()
```

Résultats :

```
dylan@cli-dylan:~$ ./ssh_space.py
({'content-type': 'application/json; charset=UTF-8', 'vary': 'Origin, X-Origin, Referer', 'date': 'Thu, 03 Oct 2024 08:44:43 GMT', 's
erver': 'ESF', 'cache-control': 'private', 'x-xss-protection': '0', 'x-frame-options': 'SAMEORIGIN', 'x-content-type-options': 'nosni
ff', 'expires': 'Thu, 03 Oct 2024 08:44:43 GMT', 'set-cookie': 'COMPASS=dynamite-integration=CgAQi8_5twYaTQAJa4lXU32QaUwrAVEuQDgZxuDN
x6iiqqEcXLN5JRKKBoaoFsr64EunBY88_ifDMgfLl7rVnNiuGkZb-tV8HoB06KvhYiJbFsiF1mtPMAE; expires=Sun, 13-Oct-2024 08:44:43 GMT; path=/; Secur
e; HttpOnly', 'alt-svc': 'h3=":443"; ma=2592000,h3-29=":443"; ma=2592000', 'transfer-encoding': 'chunked', 'status': '200', 'content-
length': '358', '-content-encoding': 'gzip'}, b'{\n  "name": "spaces/AAAAzB7cgLk/messages/8oghW7ATHag.8oghW7ATHag",\n  "text": "\xc3\
x89tat du serveur 192.168.10.152 :\\nCPU Usage: 100.0%\\nRAM Usage: 36.43%\\nDisk Usage: 21.0%\\n\xe2\x9a\xa0\xef\xb8\x8f Attention:
L\'un des seuils critiques est d\xc3\xa9pass\xc3\xa9 !\\n",\n  "thread": {\n    "name": "spaces/AAAAzB7cgLk/threads/8oghW7ATHag"\n  }
,\n  "space": {\n    "name": "spaces/AAAAzB7cgLk"\n  }\n}\n')
```



crontab -e

```
*/5 * * * * /usr/bin/python3 /home/dylan/ssh_space.py
```

plateflop Application 7 min

État du serveur 192.168.10.152 :
CPU Usage: 100.0%
RAM Usage: 36.84%
Disk Usage: 21.0%
⚠️ Attention: L'un des seuils critiques est dépassé !

plateflop Application 2 min

État du serveur 192.168.10.152 :
CPU Usage: 100.0%
RAM Usage: 37.25%
Disk Usage: 21.0%
⚠️ Attention: L'un des seuils critiques est dépassé !