📊 Surveillance des Ressources et Alerte sur le Pool de Connexion MySQL

🚀 Objectif

Nous allons mettre en place une **surveillance automatique** des ressources de notre infrastructure ainsi qu'un **système d'alerte par email** en cas de surcharge des connexions MySQL.

🔍 Ce que nous allons surveiller :

1. **Les ressources de la VM** (CPU, RAM, disque) → **via Node Exporter**.
2. **Les ressources des conteneurs Docker (`jogetapp` et `jogetdb`)** → **via cAdvisor**.
3. **Le pool de connexion MySQL** :
   - Vérifier combien de connexions sont utilisées.
   - **Envoyer un email d'alerte** si un seuil est dépassé.

🛠️ Configuration et Installation

Nous allons configurer **Prometheus + Grafana** pour la surveillance et un **script Python** pour envoyer des alertes.

✅ Étape 1 : Installation et Configuration des Conteneurs

Nous allons installer **Grafana** pour la visualisation et **Prometheus** pour collecter les métriques.

🔷 Installation de Prometheus

Lance la commande suivante :

```
docker run -d --name prom -p 9090:9090 prom/prometheus
```

Vérification :

Accède à Prometheus via ton navigateur :

```
http://192.168.193.133:9090
```

🔷 Installation de Grafana

```
docker run -d --name grafana -p 3000:3000 grafana/grafana
```

Vérification :

Accède à Grafana via :

```
http://192.168.193.133:3000
```

Identifiants par défaut :

- **Utilisateur** : `admin`
- **Mot de passe** : `admin` (à modifier après connexion)

✅ Étape 2 : Surveiller les Ressources de la VM avec Node Exporter

📌 **Node Exporter** permet de collecter les métriques système comme **CPU, RAM, et disque**.

🔷 Installation de Node Exporter

Lance cette commande sur la VM :

```
docker run -d --name node-exporter -p 9100:9100 prom/node-exporter
```

🔷 Vérification

Accède à :

```
http://192.168.193.133:9100/metrics
```

🔷 Ajouter Node Exporter à Prometheus

Modifie le fichier `prometheus.yml` et ajoute :

```
scrape_configs:
  - job_name: 'node_exporter'
    static_configs:
      - targets: ['192.168.193.133:9100']
```

Puis, redémarre **Prometheus** :

```
docker restart prom
```

✅ Étape 3 : Surveiller les Conteneurs Docker avec cAdvisor

📌 **cAdvisor** collecte des métriques détaillées sur **l'utilisation CPU/Mémoire/Disque des conteneurs**.

🔷 Installation de cAdvisor

```
docker run -d --name=cadvisor -p 8086:8080 \
  --volume=/var/run/docker.sock:/var/run/docker.sock:ro \
  --volume=/sys:/sys:ro \
  --volume=/var/lib/docker/:/var/lib/docker:ro \
  gcr.io/cadvisor/cadvisor
```

🔷 Vérification

Accède à :

```
http://192.168.193.133:8086
```

🔷 Ajouter cAdvisor à Prometheus

Ajoute ceci à `prometheus.yml` :

```
scrape_configs:
  - job_name: 'cadvisor'
    static_configs:
      - targets: ['192.168.193.133:8086']
```

Puis, redémarre **Prometheus** :

```
docker restart prom
```

✅ Étape 4 : Configurer Grafana pour Voir les Statistiques

📌 **Grafana** permet d'afficher les données collectées par Prometheus.

1. **Accéder à Grafana** :

   ```
   http://192.168.193.133:3000
   ```

2. **Ajouter Prometheus comme source de données** :
   - URL : `http://prom:9090`
   - Enregistrer.
3. **Importer un Dashboard** :
   - Aller dans "Dashboards" → "Import".
   - **ID pour Node Exporter** : `1860`
   - **ID pour cAdvisor** : `193`
   - Importer et observer les métriques 🎉.

✅ Étape 5 : Vérifier le Pool de Connexion MySQL

📌 Nous allons surveiller le nombre de connexions actives sur `jogetdb`.

🔷 Se connecter à MySQL

```
docker exec -it jogetdb mysql -u root -p
```

🔷 Vérifier le nombre de connexions

```
SHOW STATUS WHERE Variable_name = 'Threads_connected';
```

Si le nombre dépasse un **seuil critique (ex: 50 connexions)**, **nous enverrons un email**.

✅ Étape 6 : Automatiser l'Alerte Email

📌 Nous allons créer un **script Python** qui :

- Vérifie **le nombre de connexions**.
- **Envoie un email** si le seuil est dépassé.

🔷 Créer le script `monitor_mysql.py`

```
nano monitor_mysql.py
```

Ajoute ce code :

```
import smtplib
import pymysql
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart

# Configuration MySQL
MYSQL_HOST = "192.168.193.128"  # IP de la VM avec MySQL
MYSQL_USER = "joget"
MYSQL_PASSWORD = "joget"
MYSQL_DATABASE = "jwdb"
```

```
MAX_CONN = 50  # Seuil d'alerte

# Configuration SMTP
SMTP_SERVER = "smtp.gmail.com"
SMTP_PORT = 587
EMAIL_SENDER = "mouhamedtrabelsi.28@gmail.com"
EMAIL_PASSWORD = "qjuj hhes hwgd ryxr"
EMAIL_RECEIVER = "mouhamedtrabelsi.28@gmail.com"

def check_mysql_connections():
    try:
        conn = pymysql.connect(host=MYSQL_HOST, user=MYSQL_USER, pass
word=MYSQL_PASSWORD, database=MYSQL_DATABASE)
        cursor = conn.cursor()
        cursor.execute("SHOW STATUS WHERE Variable_name = 'Threads_conn
ected';")
        result = cursor.fetchone()
        conn.close()

        if result:
            current_connections = int(result[1])
            print(f"🔍 Connexions MySQL actuelles : {current_connections}")

            if current_connections > MAX_CONN:
                send_email_alert(current_connections)
            else:
                print("✅ Le nombre de connexions est normal.")

    except pymysql.Error as e:
        print(f"❌ Erreur MySQL : {e}")

def send_email_alert(current_connections):
    try:
        subject = "⚠️ Alerte MySQL : Trop de connexions"
        body = f"Le nombre de connexions MySQL a atteint {current_connection
s}, dépassant le seuil de {MAX_CONN}."
```

```
        msg = MIMEMultipart()
        msg["From"] = EMAIL_SENDER
        msg["To"] = EMAIL_RECEIVER
        msg["Subject"] = subject
        msg.attach(MIMEText(body, "plain"))

        server = smtplib.SMTP(SMTP_SERVER, SMTP_PORT)
        server.starttls()
        server.login(EMAIL_SENDER, EMAIL_PASSWORD)
        server.sendmail(EMAIL_SENDER, EMAIL_RECEIVER, msg.as_string())
        server.quit()

        print("✅ Email d'alerte envoyé avec succès !")

    except Exception as e:
        print(f"❌ Erreur d'envoi de l'email : {e}")

if __name__ == "__main__":
    check_mysql_connections()
```

✅ Étape 7 : Automatiser avec Crontab

```
crontab -e
```

Ajoute :

```
*/5 * * * * /usr/bin/python3 /home/user/monitor_mysql.py
```

✅ Le script fonctionne automatiquement !

🎯 Résultat Final

✅ Surveillance en temps réel via Prometheus & Grafana.

✅ **Alerte email si MySQL dépasse 50 connexions.**

✅ **Système entièrement automatisé.**