UNIVERSITÉ DE LORRAINE | IUT nancy Charlemagne

BUT informatique 3A, parcours DACS

(pré-)SAÉ de projet tuteuré

Centralisation et analyse de logs
21 octobre 2024

1 Objectif

Dans ce projet, il s'agit de développer une solution de centralisation et d'analyse des logs provenant de plusieurs serveurs ou conteneurs Docker. Il sera nécessaire de configurer un serveur web pour visualiser les logs, utiliser une base de données pour les stocker, et mettre en place une interface pour détecter et afficher des anomalies dans ces logs.

Le travail s'effectue par groupe de 2 ou 3 étudiants.

2 Étapes principales

1. *Installation de 2 serveurs.* Deux serveurs (un web et un autre SGBD) doivent être installés, chacun dans un conteneur Docker. Une fois ces serveurs installés, déployer une application (type Wordpress), utilisant ces 2 serveurs, qui permettra d'alimenter « naturellement » les logs de vos serveurs.
2. *Collecte des logs.* Les logs des différents services (serveur web, SGB) doivent être collectés automatiquement depuis plusieurs sources (par exemple `/var/log`, les journaux d'applications web ou de bases de données). Un script Ruby sera utilisé pour récupérer ces logs régulièrement (via `rsyslog`, `journalctl`, ou des commandes comme `cat`, `tail`). Les logs seront transférés vers un serveur central via des protocoles comme `rsync`, `scp`, ou des `API`.
3. *Stockage des logs dans un SGBD.* Les logs collectés seront stockés dans une base de données relationnelle (ex : MySQL ou PostgreSQL) pour permettre leur requêtage et analyse. Pour cela, il sera nécessaire de créer un schéma de base de données pour structurer les logs (tables avec des colonnes comme ltimestamp, source, type de log, message, etc.). Là encore, Ruby peut être utilisé pour insérer les logs dans la base de données après les avoir récupérés. Il peut être intéressant d'utiliser l'indexation pour améliorer les performances lors des requêtes.
4. *Détection d'anomalies.* Une anomalie est définie comme tout événement ou comportement sortant de l'ordinaire. Cela peut inclure :
   — une série d'erreurs 500 sur le serveur web,
   — des tentatives de connexion échouées répétées (pouvant indiquer une attaque par force brute),
   — des pics soudains dans l'utilisation des ressources (CPU/mémoire).

   À cette fin, prévoir le développement d'un script Ruby qui analyse les logs stockés dans la base de données et détecte des motifs d'anomalies. Des conditions simples peuvent être utilisées (ex : plus de 10 erreurs 500 en une heure) mais également plus complexes (ex : un taux d'erreurs supérieur à la normale sur un intervalle de temps donné).

Il pourra être utile de définir des règles d'anomalies comme :

— *Erreurs serveur web* : plus de 5 erreurs 500 en moins de 5 minutes.
— *Requêtes lentes dans la base de données* : requêtes prenant plus de 2 secondes répétées plusieurs fois (configurer alors MySQL/PostgreSQL pour qu'ils enregistrent les requêtes lentes, les échecs de connexion et les erreurs SQL dans les logs).
— *Utilisation excessive du CPU* : détection de pics d'utilisation CPU dans les logs de `top` ou `htop` (voir l'option `-b` par exemple).

3 Résultats attendus

— Un dépôt Git avec le code Ruby pour la collecte, l'analyse et la gestion des anomalies.
— Les fichiers de configuration Docker et Docker Compose.
— Une démonstration du résultat obtenu vendredi 25 octobre après-midi.