Cas pratique 2 (E5)

Surveiller une application d'intelligence artificielle, Résoudre les incidents techniques

Projet AniMOV : Surveillance et Analyse Comportementale des Chèvres avec l'IA

Formation Développeur en Intelligence Artificielle
RNCP 37827
Promotion 2023-2024

Manuel CALDEIRA

SIMPLON .CO

académie Orléans-Tours
greta VAL DE LOIRE
FORMEZ-VOUS À DEMAIN

Microsoft en France
Ecole IA

Table des matières

1. INTRODUCTION

Le Livrable E5 s'attache à répondre à des standards de gestion et de suivi nécessaires au bon fonctionnement d'applications d'intelligence artificielle. La section C20 se concentre sur l'identification des indicateurs de performance clés et le déploiement de solutions de monitoring adaptées pour garantir la surveillance continue et la réactivité du système. La section C21, quant à elle, traite de la gestion des incidents techniques en établissant des procédures pour apporter des correctifs efficaces et assurer la résilience de l'application.

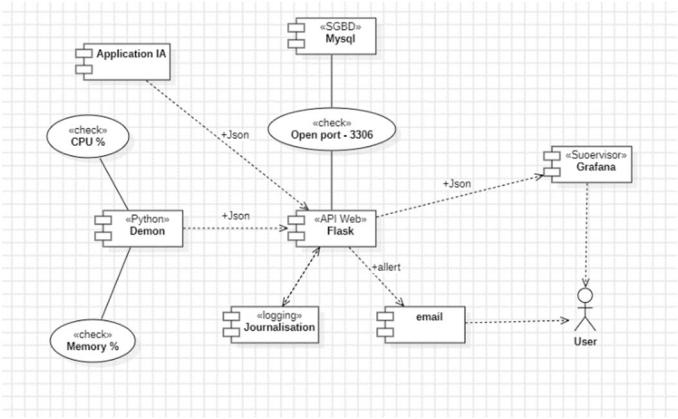
Cette application fournit une solution automatisée pour la surveillance de la santé et des performances d'un système serveur, permettant une intervention proactive pour maintenir une haute disponibilité et fiabilité du système.

1.1. OBJECTIF DE L'APPLICATION :

L'objectif principal de cette application est de surveiller les indicateurs clés de performance (KPI) tels que l'utilisation du CPU et de la mémoire, ainsi que la disponibilité du port de la base de données (3306 pour MySQL), pour assurer que l'application fonctionne de manière optimale et pour prévenir les interruptions de service. En cas de problèmes détectés, comme une utilisation élevée des ressources ou des problèmes de connectivité, l'application vise à alerter les administrateurs systèmes ou les opérateurs de manière rapide et automatisée.

1.2. FONCTIONNEMENT DE L'APPLICATION :

Le schéma présente une infrastructure de monitoring en place pour une application d'intelligence artificielle (IA) qui utilise une base de données MySQL. Voici une explication de son fonctionnement :



1. Surveillance des Performances Système :

Un processus Python (Daemon) est dédié à la surveillance de l'utilisation du CPU et de la mémoire du serveur sur lequel l'application d'IA est exécutée. Ces métriques sont cruciales car elles peuvent directement influencer les performances et la réactivité de l'application IA. Un usage élevé du CPU ou de la mémoire peut signaler un besoin d'optimisation ou d'ajustement des ressources.

2. Journalisation et Gestion des Alertes :

Les données de performance collectées par le daemon sont envoyées en format JSON à une API Web Flask. Cette API Flask est responsable de la journalisation des données, fournissant ainsi un historique des performances pour des analyses ultérieures. Flask gère également les alertes, déclenchant des notifications par email à l'utilisateur si les métriques surveillées dépassent des

seuils prédéfinis, ce qui indiquerait un problème potentiel nécessitant une attention.

3. Surveillance de la Disponibilité de la Base de Données :

   En parallèle, le système vérifie la disponibilité du port 3306, qui est le port standard pour les connexions à une base de données MySQL. Cela est essentiel pour s'assurer que l'application d'IA peut communiquer sans interruption avec sa base de données, ce qui est critique pour les opérations qui dépendent de l'accès aux données, comme le traitement des requêtes et la mise à jour des modèles d'IA.

4. Visualisation et Supervision :

   Grafana est utilisé comme outil de supervision, recevant les données de performance au format JSON. Il permet de visualiser ces données à travers des tableaux de bord, facilitant ainsi la surveillance en temps réel et l'analyse des tendances sur le long terme.

5. Interaction avec l'Utilisateur :

   L'utilisateur final du système de monitoring reçoit des alertes par email générées par Flask en cas de détection de conditions anormales. En outre, l'utilisateur peut visualiser avec Grafana afin de surveiller l'état du système et analyser les performances de l'application d'IA et de l'infrastructure sous-jacente en temps réel.

2. MONITORING DE L'APPLICATION

2.1 DEFINITION DES METRIQUES

Le suivi des métriques telles que l'utilisation du CPU, de la mémoire et la disponibilité du port de la base de données MySQL est crucial pour maintenir la santé et l'efficacité d'un système informatique, particulièrement pour une application d'intelligence artificielle.

L'utilisation du CPU reflète la charge de travail du serveur, indiquant si le modèle d'IA est activement engagé dans le traitement de données ou la réalisation d'inférences. De même, une forte consommation de mémoire peut révéler la taille importante des données ou des modèles utilisés, ou peut-être une gestion de la mémoire suboptimale. Ces deux métriques influent directement sur la rapidité et l'efficacité de l'application d'IA.

Quant à la disponibilité du port de la base de données, elle est vitale car les applications d'IA dépendent souvent des bases de données pour l'accès aux données d'entraînement, le stockage des résultats, ou pour effectuer des inférences en temps réel. Un port de base de données non disponible peut entraîner une interruption des services, impactant la fonctionnalité de l'application.

Le monitoring de ces aspects est indispensable pour avoir une vue complète de l'état du serveur et pour garantir la performance optimale de l'application d'IA. Ceci

permet une gestion proactive et une intervention rapide en cas de dysfonctionnement, assurant ainsi la continuité et la fiabilité des services d'IA.

2.2 DEFINITION DES SEUILS ET ALERTES

Le choix des seuils et des alertes dans un système de monitoring repose sur une approche stratégique vis-à-vis de l'application. Les seuils sont définis en fonction des besoins spécifiques de l'application et de ses exigences en matière de performance et de disponibilité. Ils sont calibrés pour réagir rapidement aux problèmes critiques qui pourraient avoir un impact significatif sur l'application, comme la perte de connexion à la base de données MySQL. De plus, les seuils sont configurés pour éviter les surcharges du système tout en optimisant l'utilisation des ressources. Cette approche vise à maintenir un équilibre entre la réactivité aux incidents majeurs et la réduction des faux positifs, assurant ainsi une surveillance efficace et pertinente de l'application.

Les faux positifs sont des alertes qui se déclenchent à tort, sans qu'il y ait un véritable problème, et ils peuvent être source de perturbations inutiles pour l'équipe de maintenance. Pour minimiser ces faux positifs, les seuils sont soigneusement ajustés en tenant compte des caractéristiques de l'application.

En définissant des seuils appropriés, l'équipe de maintenance peut anticiper les problèmes potentiels en analysant les tendances au fil du temps. Par exemple, la surveillance de la croissance des logs ou des alertes sur une période permet de prévoir les besoins en ressources futurs de l'application. Le choix des seuils et des alertes est une décision stratégique visant à garantir la stabilité, la performance et la disponibilité de l'application d'intelligence artificielle, tout en minimisant les interruptions inutiles du service.

2.3 SOLUTION DU MONITORING

Le choix de cette solution de monitoring repose sur trois piliers stratégiques pour la gestion d'une application d'intelligence artificielle (IA) : la fiabilité, la réactivité et la maintenabilité.

- La **fiabilité** est garantie grâce à la surveillance en temps réel des métriques systèmes essentiels comme l'utilisation du CPU et de la mémoire, évitant ainsi les surcharges et maintenant des performances stables.
- La **réactivité** est assurée par la génération d'alertes immédiates en cas de dépassement des seuils, minimisant ainsi les temps d'arrêt.
- Enfin, la **maintenabilité** est renforcée par la journalisation des données, facilitant le diagnostic des problèmes et l'amélioration continue de l'application.

La solution décrite fonctionne en plusieurs étapes. Tout d'abord, un daemon Python est chargé de collecter les données, jouant un rôle essentiel dans le suivi des performances du système. Ensuite, pour la journalisation, une API Flask est utilisée. Cette API ne se contente pas simplement de collecter les journaux d'événements, mais prend également en charge leur gestion, ce qui permet de créer un historique détaillé et structuré des événements.

Il est important de noter que l'API Flask joue ici un double rôle. Non seulement elle facilite la collecte et l'organisation des données de journalisation, mais elle rend aussi ces données accessibles pour leur visualisation dans Grafana. Grafana, quant à lui, est utilisé pour afficher ces métriques en temps réel et pour analyser les tendances historiques.

3. PROCEDURE D'INSTALLATION ET DE CONFIGURATION :

3.1 INSTALLATION DES DEPENDANCES :

- Installer Python et les packages nécessaires (`psutil`, `docker`, `yaml`, `requests`, `flask`).
- Installer Grafana et configurer.

3.2 LANCEMENT DU DAEMON PYTHON :

- Lancer le script Python pour qu'il s'exécute, collectant les données systèmes pour les envoyant à l'API Flask.

3.3 MISE EN PLACE DE L'API FLASK :

- Déployer l'application Flask, configurer les endpoints pour la réception des données, la journalisation, et la génération d'alertes.
- Configurer les seuils d'alerte selon les exigences de performance de l'application d'IA.

3.4 CONFIGURATION DE GRAFANA :

- Créer des tableaux de bord dans Grafana pour afficher les métriques collectées.
- Définir des requêtes dans Grafana pour interroger les données transmises par Flask.

3.5 CONFIGURATION DE LA JOURNALISATION :

- Définir les niveaux de log dans Flask et configurer le FileHandler pour écrire les logs dans un fichier spécifique.

3.6 CONFIGURATION DES ALERTES :

- Établir la logique d'alerte dans Flask pour envoyer des notifications par email quand les seuils d'alerte sont franchis.

4. RESOLUTION D'INCIDENTS TECHNIQUES

4.1 DEFINITION DE L'INCIDENT

Pour vérifier la disponibilité du port 3306 parmi la liste des ports ouverts le système effectue une vérification périodique en interrogeant la liste des ports actuellement ouverts sur l'adresse du conteneur Docker de la base de données. Cette approche ne nécessite pas d'établir une connexion directe avec le port 3306, ce qui permet de minimiser les interactions potentiellement problématiques.

En cas de détection de l'absence du port 3306 dans la liste des ports ouverts, le système prend des mesures pour résoudre le problème de manière automatique. L'une de ces mesures est de redémarrer le conteneur Docker de la base de données afin de rétablir la disponibilité du port. Une fois le conteneur redémarré, le système réitère la vérification pour s'assurer que le port 3306 est désormais ouvert. Si la vérification est réussie, l'incident est considéré comme résolu. Parallèlement, le système enregistre les détails de l'incident dans les logs via Flask et génère des alertes par email pour tenir informées les personnes concernées de l'incident survenu et de sa résolution.

4.2 IDENTIFICATION DE LA/LES CAUSE(S)

Lorsqu'un incident survient, plusieurs étapes d'investigation sont cruciales pour comprendre la nature du problème, déterminer ses causes et trouver des solutions. Tout d'abord, l'analyse des logs générés par l'application et le système fournit des indices sur le moment de l'incident, les messages d'erreur, et les tentatives de reconnexion, permettant d'identifier la cause sous-jacente. Ensuite, les alertes générées sont examinées en détail pour comprendre la gravité de l'incident.

L'examen du code source de l'application, du daemon Python, et de l'API Flask est essentiel pour vérifier la gestion des erreurs et des mécanismes de reconnexion. La reproduction de l'incident dans un environnement de développement permet de tester différentes hypothèses sur sa cause. L'analyse approfondie des données collectées, y compris les logs, les alertes, et les résultats de la reproduction, révèle des schémas et des tendances pour identifier la source du problème.

Une fois la cause identifiée, des mesures correctives sont prises, incluant des ajustements dans le code, des améliorations de configuration, et des modifications des seuils d'alerte. Des mesures préventives sont également mises en place pour éviter la récurrence de l'incident. Enfin, toutes les étapes de l'investigation et les actions entreprises sont documentées pour constituer une base de connaissances utile à l'équipe de maintenance et pour faciliter la gestion future des incidents similaires.

4.3 SOLUTIONS

La solution que nous proposons, axée sur l'automatisation du redémarrage du conteneur Docker hébergeant MySQL en cas de détection de la perte du port 3306, est une stratégie proactive visant à garantir la disponibilité continue de la base de données. Dans cette méthode, nous avons des étapes bien définies : le daemon Python est responsable de la surveillance et de la remontée des données du serveur, tandis que l'API Flask gère en interne la détection de la perte du port et le déclenchement du redémarrage.

La phase de test et de validation est essentielle dans notre démarche. En simulant manuellement la perte du port 3306, nous pouvons confirmer que le système fonctionne correctement : c'est l'API Flask qui surveille directement le port et, en cas de problème détecté, elle procède elle-même au redémarrage du conteneur Docker hébergeant MySQL. Cette approche d'automatisation des réactions aux incidents contribue de manière significative à une maintenance proactive et renforce la stabilité globale du système.

Nous veillons également à suivre les bonnes pratiques de gestion de versions avec Git pour toutes les modifications apportées, assurant ainsi une traçabilité et une maintenance efficaces du système sur le long terme.

ANNEXES

ANEXES

I. TABLEAU DES METRIQUES DE SURVEILLANCE SYSTEME

Métrique	Description	Seuil d'alerte	Seuil de fin d'alerte
Utilisation du CPU	Pourcentage d'utilisation du processeur	> 80%	< 70%
Utilisation de la mémoire	Quantité de RAM utilisée	> 75%	< 60%
Utilisation du disque	Espace disque utilisé	> 90%	< 80%
Intervalles de temps	Fréquence de collecte des métriques	Variable	Variable
État de la ligne de vie	État de fonctionnement de l'application	Défaillance	Normal
Disponibilité du port 3306	Accessibilité du port MySQL	Non disponible	Disponible

II. CAPTURE D'ECRAN