# Livrable E5 : Stratégie de Monitoring, Gestion des Incidents et Plan de Continuité

**Nom du Projet :** CSV Analyzer  
**Date :** 24/07/2024

## Table des Matières

1. [Introduction](#Xe3d0fc0bea9a42ce7605565d0964033d7f6ee47)
   * 1.1. Finalité de ce Document
   * 1.2. Philosophie du Monitoring : De la Réactivité à la Proactivité
   * 1.3. Public Cible
2. [Stratégie de Monitoring Holistique](#X30807461025c0b0debc96edcd3d354fa32ecf81)
   * 2.1. Les Piliers du Monitoring
   * 2.2. Métriques Clés (KPIs) et Seuils
   * 2.3. Stack Technologique de Monitoring
3. [Processus de Gestion des Incidents](#X68b847921266c0254ef84fa7ce555add34478dd)
   * 3.1. Cycle de Vie d’un Incident
   * 3.2. Rôles et Responsabilités
   * 3.3. Playbooks de Résolution d’Incidents
4. [Analyse Post-Incident et Amélioration Continue](#X029296a7e8b4134b71474fc6c30d166f3674a26)
   * 4.1. L’Importance du Post-Mortem
   * 4.2. Processus de Post-Mortem
   * 4.3. Suivi des Actions Correctives
5. [Plan de Continuité d’Activité (PCA) et Reprise après Sinistre (PRA)](#X4c23411478811e42655dee31fa907662ac1f24f)
   * 5.1. Analyse d’Impact sur l’Activité (BIA)
   * 5.2. Stratégies de Sauvegarde et de Restauration
   * 5.3. Plan de Reprise après Sinistre
6. [Conclusion](#X6b297cc947e0db5526be369b1f0f4ddc67b2415)
7. [Techniques de Monitoring Avancées](#X9b426677015be4d6d11811e449c2fd0b8bab5c3)
   * 7.1. Monitoring Synthétique et RUM
   * 7.2. Tracing Distribué
8. [Bonnes Pratiques d’Observabilité](#Xccc7917ae4ca9344d41f8a88b3d3bfe6b815fa3)
   * 8.1. SLIs, SLOs et Budgets d’Erreur
   * 8.2. Culture de l’Observabilité
9. [Gestion des Incidents de Sécurité](#Xd4ed35902a879018148ec119eb73745661a1427)
   * 9.1. Détection et Réponse aux Incidents de Sécurité
   * 9.2. Forensic et Analyse Post-Mortem Sécurité
10. [Optimisation des Performances et Planification de Capacité](#X101f499b60c69b989270733f8df042e17bc6c1a)
    * 10.1. Analyse des Goulots d’Étranglement
    * 10.2. Planification de Capacité
11. [Conformité et Audit](#X10f0c8ebdfef55645546590e650780797bab2a6)
    * 11.1. Conformité Réglementaire
    * 11.2. Pistes d’Audit
12. [Approfondissement des Outils](#X8c1b4bf7780657d93235842994d5c5ff218e63d)
    * 12.1. Configuration Avancée de Prometheus et Grafana
    * 12.2. Utilisation de Loki pour les Logs Structurés
    * 12.3. MLflow pour le Monitoring de Modèles en Production
13. [Feuille de Route et Améliorations Futures](#X56d7da17ec8faccaf37acc52a6bb51ffb186e02)
    * 13.1. Évolution de la Stack de Monitoring
    * 13.2. Intégration avec des Systèmes Tiers

## 1. Introduction

### 1.1. Finalité de ce Document

Ce document établit un cadre de gouvernance complet pour la surveillance, la gestion des incidents et la continuité des services de l’application **CSV Analyzer**. Il vise à garantir une disponibilité, une fiabilité et une performance optimales, tout en assurant la qualité des prédictions des modèles de Machine Learning.

### 1.2. Philosophie du Monitoring : De la Réactivité à la Proactivité

Nous adoptons une approche proactive du monitoring. L’objectif n’est pas seulement de réagir aux pannes, mais de les anticiper en identifiant des signaux faibles et des tendances de dégradation. Cela implique une surveillance à plusieurs niveaux, de l’infrastructure matérielle à la performance du modèle de ML.

### 1.3. Public Cible

Ce document s’adresse aux : - **Ingénieurs SRE/DevOps :** Pour la mise en œuvre et la maintenance de la stack de monitoring. - **Développeurs :** Pour comprendre comment instrumenter l’application et analyser les performances. - **Data Scientists :** Pour le monitoring spécifique des modèles de ML. - **Chefs de Projet :** Pour la supervision de la qualité de service (SLA/SLO).

## 2. Stratégie de Monitoring Holistique

### 2.1. Les Piliers du Monitoring

Notre stratégie repose sur trois piliers, souvent appelés les “Trois Piliers de l’Observabilité” : 1. **Logs :** Enregistrements granulaires et horodatés des événements. Ils sont essentiels pour le débogage et l’analyse forensique. 2. **Métriques :** Données numériques agrégées sur des intervalles de temps (ex: utilisation CPU, latence). Elles sont idéales pour le suivi des tendances et les alertes. 3. **Traces :** Représentation du cycle de vie d’une requête à travers les différents services de l’application. Elles sont cruciales pour identifier les goulots d’étranglement dans une architecture de microservices.

### 2.2. Métriques Clés (KPIs) et Seuils

#### 2.2.1. Métriques “Golden Signals” (Application)

* **Latence :** Temps de réponse des requêtes. Seuil d’alerte : > 500ms (moyenne sur 1 min).
* **Trafic :** Nombre de requêtes par seconde (RPS). Seuil d’alerte : Baisse ou pic anormal de 50%.
* **Erreurs :** Taux de requêtes en erreur (5xx). Seuil d’alerte : > 2% sur 5 minutes.
* **Saturation :** À quel point le service est proche de sa pleine capacité. Seuil d’alerte : > 80% d’utilisation des ressources.

#### 2.2.2. Métriques d’Infrastructure

* **Utilisation CPU :** Seuil d’alerte : > 80% pendant 10 minutes.
* **Utilisation Mémoire (RAM) :** Seuil d’alerte : > 85%.
* **Utilisation Disque :** Seuil d’alerte : > 90%.

#### 2.2.3. Métriques du Modèle de Machine Learning

* **Performance (Accuracy, F1-Score) :** Seuil d’alerte : Baisse > 10% par rapport au baseline.
* **Dérive des Données (Data Drift) :** Mesurée par le test de Kolmogorov-Smirnov. Seuil d’alerte : p-value < 0.05.
* **Latence de Prédiction :** Seuil d’alerte : > 200ms.

### 2.3. Stack Technologique de Monitoring

* **Prometheus :** Collecte de métriques.
* **Grafana :** Visualisation et tableaux de bord.
* **Alertmanager :** Gestion des alertes.
* **MLflow :** Monitoring des modèles ML.
* **Loki & Promtail :** Agrégation de logs.

## 3. Processus de Gestion des Incidents

### 3.1. Cycle de Vie d’un Incident

1. **Détection :** Une alerte est déclenchée par Prometheus.
2. **Notification :** Alertmanager notifie l’équipe d’astreinte via Slack et email.
3. **Triage :** L’ingénieur d’astreinte évalue la criticité de l’incident.
4. **Diagnostic :** Analyse des dashboards Grafana, des logs (Loki) et des traces.
5. **Résolution :** Application d’un playbook de résolution.
6. **Communication :** Information des parties prenantes sur l’état de l’incident.
7. **Clôture :** L’incident est résolu, le service est rétabli.

### 3.2. Rôles et Responsabilités

* **Ingénieur d’Astreinte (On-Call Engineer) :** Premier point de contact pour les alertes. Responsable du triage et de la résolution initiale.
* **Commandant d’Incident (Incident Commander) :** Coordonne la réponse pour les incidents majeurs, gère la communication.
* **Experts Techniques (SMEs) :** Développeurs, Data Scientists, experts en base de données, etc., qui sont sollicités pour leur expertise.

### 3.3. Playbooks de Résolution d’Incidents

Des guides détaillés sont créés pour les scénarios d’incidents les plus courants.

**Exemple de Playbook : Application Down (5xx Errors)** - **Symptômes :** Alerte Prometheus HighErrorRate, plaintes des utilisateurs. - **Étapes de Diagnostic :** 1. Vérifier le dashboard Grafana principal : quel service est en erreur ? 2. Consulter les logs du service web : docker-compose logs -f web. 3. Vérifier la connectivité à la base de données. - **Actions de Résolution :** 1. **Action rapide :** Redémarrer le service : docker-compose restart web. 2. **Si le problème persiste :** Analyser les logs pour une cause racine (ex: déploiement récent, bug). 3. **Action de dernier recours :** Rollback à la version précédente.

## 4. Analyse Post-Incident et Amélioration Continue

### 4.1. L’Importance du Post-Mortem

Le but d’un post-mortem n’est pas de blâmer, mais de comprendre. C’est un outil d’apprentissage essentiel pour construire un système plus résilient.

### 4.2. Processus de Post-Mortem

* **Déclenchement :** Pour tout incident ayant un impact significatif sur les utilisateurs.
* **Réunion :** Rassembler toutes les personnes impliquées.
* **Documentation :** Remplir un template de post-mortem standard.

**Template de Post-Mortem :** - **Résumé :** Qu’est-il arrivé ? Quel impact ? - **Chronologie détaillée :** Qui a fait quoi et quand ? - **Analyse de la Cause Racine (Root Cause Analysis) :** Utiliser la méthode des “5 Pourquoi”. - **Actions Correctives :** Tâches concrètes avec un propriétaire et une date d’échéance.

### 4.3. Suivi des Actions Correctives

Les actions issues des post-mortems sont intégrées dans le backlog de développement avec une haute priorité.

## 5. Plan de Continuité d’Activité (PCA) et Reprise après Sinistre (PRA)

### 5.1. Analyse d’Impact sur l’Activité (BIA)

* **RTO (Recovery Time Objective) :** Temps maximum acceptable pour restaurer le service après une panne. **Objectif : 4 heures.**
* **RPO (Recovery Point Objective) :** Perte de données maximale acceptable. **Objectif : 1 heure.**

### 5.2. Stratégies de Sauvegarde et de Restauration

* **Base de Données (PostgreSQL) :**
  + **Sauvegardes :** Snapshots automatiques toutes les heures.
  + **Restauration :** Procédure documentée pour restaurer à partir d’un snapshot.
* **Modèles de ML (MLflow) :**
  + **Sauvegardes :** Le backend de stockage des artefacts (ex: S3) est sauvegardé.
* **Code Source :**
  + **Sauvegardes :** Géré par Git et hébergé sur une plateforme redondante (GitHub).

### 5.3. Plan de Reprise après Sinistre

Ce plan est activé en cas de sinistre majeur (ex: perte d’une zone de disponibilité cloud). - **Infrastructure as Code (IaC) :** L’ensemble de l’infrastructure est défini dans des fichiers (Terraform, Docker Compose), permettant de la recréer rapidement dans une autre région. - **Tests de Reprise :** Des exercices de reprise après sinistre sont menés tous les trimestres pour valider le plan et entraîner les équipes.

## 6. Conclusion

Cette stratégie intégrée de monitoring, de gestion des incidents et de continuité d’activité fournit à l’équipe du **CSV Analyzer** les outils et les processus nécessaires pour maintenir un service de haute qualité. C’est un document vivant, destiné à être amélioré en continu à travers les leçons apprises de chaque incident et de chaque exercice de simulation.

## 7. Techniques de Monitoring Avancées

### 7.1. Monitoring Synthétique et RUM

* **Monitoring Synthétique :** Simule des transactions utilisateur critiques (ex: connexion, upload CSV) depuis des emplacements géographiques variés pour mesurer la disponibilité et la performance perçue.
* **Real User Monitoring (RUM) :** Collecte des données directement depuis les navigateurs des utilisateurs finaux pour obtenir une vue précise de l’expérience utilisateur réelle (temps de chargement, erreurs JavaScript, etc.).

### 7.2. Tracing Distribué

* **OpenTelemetry :** Instrumente l’application pour générer des traces distribuées, permettant de suivre le chemin d’une requête à travers tous les microservices et composants (API, base de données, modèle ML, etc.). Essentiel pour diagnostiquer les latences dans les architectures complexes.

## 8. Bonnes Pratiques d’Observabilité

### 8.1. SLIs, SLOs et Budgets d’Erreur

* **Service Level Indicators (SLIs) :** Mesures quantifiables de la performance d’un service (ex: taux de succès des requêtes, latence).
* **Service Level Objectives (SLOs) :** Cibles pour les SLIs (ex: 99.9% de requêtes réussies, latence < 200ms pour 95% des requêtes).
* **Budgets d’Erreur :** La quantité de temps qu’un service peut être en dessous de son SLO sans conséquences majeures. Permet de prendre des décisions éclairées sur le déploiement de nouvelles fonctionnalités vs. la stabilité.

### 8.2. Culture de l’Observabilité

* Intégrer l’observabilité dès la conception des services.
* Responsabiliser les équipes de développement pour le monitoring de leurs propres services.
* Utiliser les données d’observabilité pour guider les décisions d’ingénierie et d’affaires.

## 9. Gestion des Incidents de Sécurité

### 9.1. Détection et Réponse aux Incidents de Sécurité

* **SIEM (Security Information and Event Management) :** Agrégation et corrélation des logs de sécurité (authentification, accès, modifications de configuration) pour détecter les activités suspectes.
* **Playbooks de Réponse aux Incidents de Sécurité :** Procédures spécifiques pour gérer les brèches de données, les attaques DDoS, les injections SQL, etc.

### 9.2. Forensic et Analyse Post-Mortem Sécurité

* Collecte et analyse des preuves numériques après un incident de sécurité pour comprendre la portée de l’attaque, identifier les vulnérabilités exploitées et prévenir de futures occurrences.

## 10. Optimisation des Performances et Planification de Capacité

### 10.1. Analyse des Goulots d’Étranglement

* Utiliser les métriques de latence et de saturation (CPU, RAM, I/O disque, requêtes DB) pour identifier les composants qui limitent la performance globale de l’application.
* Profilage du code pour identifier les fonctions ou les requêtes les plus coûteuses en ressources.

### 10.2. Planification de Capacité

* Basée sur l’analyse des tendances historiques des métriques de trafic et de ressources, anticiper les besoins futurs en infrastructure pour supporter la croissance de l’application.
* Mettre en place des tests de charge réguliers pour valider la capacité actuelle et identifier les points de rupture.

## 11. Conformité et Audit

### 11.1. Conformité Réglementaire

* Assurer que les pratiques de monitoring et de gestion des données respectent les réglementations en vigueur (ex: RGPD pour la protection des données personnelles, HIPAA pour la santé).
* Documenter les processus de collecte, de stockage et d’accès aux données de monitoring pour prouver la conformité.

### 11.2. Pistes d’Audit

* Maintenir des logs d’audit détaillés pour toutes les actions critiques (connexions, modifications de configuration, accès aux données sensibles) afin de pouvoir retracer les événements en cas d’incident ou d’exigence d’audit.

## 12. Approfondissement des Outils

### 12.1. Configuration Avancée de Prometheus et Grafana

* **Prometheus :** Utilisation de relabel\_configs pour un scraping dynamique, intégration avec des service discoveries (Kubernetes, Consul).
* **Grafana :** Création de dashboards dynamiques avec des variables, utilisation de templates, intégration avec des sources de données multiples.

### 12.2. Utilisation de Loki pour les Logs Structurés

* **Loki :** Système d’agrégation de logs léger, optimisé pour les logs non structurés, mais capable de gérer des logs structurés via des parsers (ex: JSON, Logfmt).
* **Promtail :** Agent de collecte de logs qui envoie les logs à Loki, avec des capacités de transformation et de filtrage.

### 12.3. MLflow pour le Monitoring de Modèles en Production

* Utilisation du **MLflow Model Registry** pour gérer les transitions de modèles (Staging -> Production).
* Intégration des métriques de performance du modèle (accuracy, F1-score) et de dérive (data drift, concept drift) directement dans MLflow Tracking pour une vue centralisée.

## 13. Feuille de Route et Améliorations Futures

### 13.1. Évolution de la Stack de Monitoring

* Exploration de solutions de monitoring plus intégrées (ex: Datadog, New Relic) pour une observabilité “out-of-the-box” plus poussée.
* Mise en place d’un système d’AIOps pour automatiser la détection d’anomalies et la corrélation d’événements.

### 13.2. Intégration avec des Systèmes Tiers

* Connexion avec des systèmes de gestion de tickets (Jira, ServiceNow) pour automatiser la création d’incidents.
* Intégration avec des outils de communication (Microsoft Teams, Google Chat) pour des notifications d’alertes plus riches.