

# Rapport : Système de surveillance de pression artérielle en streaming avec FHIR, Kafka, Elasticsearch et Kibana

## 1. Introduction

La surveillance médicale continue est devenue un élément essentiel dans le suivi des patients, notamment pour les pathologies cardiovasculaires. La pression artérielle constitue un indicateur critique permettant d'identifier rapidement des situations dangereuses comme l'hypertension sévère ou l'hypotension. Dans un contexte hospitalier moderne, les données sont produites en continu par des capteurs et doivent être analysées immédiatement afin de déclencher des alertes rapides.

L'objectif de ce projet est de concevoir une architecture de traitement de données en temps réel capable de simuler un flux de données médicales, de les analyser automatiquement et d'identifier les anomalies. Pour garantir l'interopérabilité médicale, les données sont représentées selon le standard FHIR (Fast Healthcare Interoperability Resources). Ce standard est aujourd'hui largement utilisé pour échanger des informations de santé entre systèmes hospitaliers.

Le système développé repose sur une architecture de streaming inspirée des pipelines Big Data. Les messages médicaux sont générés, envoyés à un système de messagerie distribué, analysés automatiquement, puis stockés dans un moteur d'indexation afin d'être visualisés dans un tableau de bord interactif. Ainsi, ce projet reproduit le fonctionnement d'une plateforme de télésurveillance médicale temps réel.

## 2. Architecture générale du système

Le système est organisé autour d'un flux de données continu. Chaque composant a un rôle précis dans le traitement de l'information médicale.

Le générateur produit des observations médicales simulées correspondant à des mesures de pression artérielle pour différents patients. Ces observations sont structurées au format FHIR afin d'imiter un système hospitalier réel.

Les données sont ensuite envoyées dans Kafka qui agit comme un bus de communication. Kafka joue le rôle d'intermédiaire entre la production et le traitement des données. Il permet de stocker temporairement les messages et d'assurer leur transmission fiable.

Un programme consommateur récupère les messages depuis Kafka. Ce composant représente le module d'analyse médicale. Il lit chaque observation, vérifie les valeurs et détermine si elles correspondent à un cas normal ou pathologique.

Les résultats sont ensuite traités différemment selon la situation :

Lorsque les valeurs sont normales, elles sont archivées localement dans des fichiers JSON.

Lorsque les valeurs sont anormales, elles sont envoyées vers Elasticsearch pour être indexées et surveillées.

Enfin Kibana permet d'explorer et visualiser les anomalies via des graphiques dynamiques.

Le flux global peut être résumé ainsi :

Génération médicale → Transmission Kafka → Analyse → Indexation → Visualisation

Cette architecture correspond à un pipeline de données médicales temps réel.

### 3. Génération des données médicales

Afin de simuler un environnement hospitalier, un module Python génère des observations de pression artérielle pour plusieurs patients fictifs. Les valeurs systoliques et diastoliques sont produites aléatoirement dans des plages réalistes.

Chaque observation respecte la structure FHIR Observation. Le message contient notamment :

- un identifiant patient
- une date de mesure
- une valeur systolique
- une valeur diastolique

L'utilisation du standard FHIR rend le projet cohérent avec les systèmes médicaux existants.

Même si les données sont simulées, leur format correspond à un vrai échange hospitalier.

Le producteur envoie ensuite chaque observation dans un topic Kafka, ce qui reproduit un flux continu provenant d'appareils médicaux.

### 4. Transmission avec Kafka

Kafka agit comme un système de streaming distribué. Son rôle est de recevoir les messages du producteur et de les rendre disponibles au consommateur.

Dans un contexte réel, Kafka permet de connecter plusieurs systèmes hospitaliers sans dépendance directe entre eux. Le producteur n'a pas besoin de connaître le consommateur et inversement. Chaque observation médicale devient un message Kafka. Le consommateur lit ces messages au fur et à mesure de leur arrivée, ce qui permet une analyse en temps réel.

Cette séparation garantit la scalabilité du système. On pourrait ajouter plusieurs consommateurs ou plusieurs producteurs sans modifier l'architecture.

## 5. Analyse des données médicales

Le consommateur Python représente le module d'intelligence médicale. Il applique des règles cliniques simples pour identifier les anomalies.

Les seuils utilisés correspondent aux recommandations médicales :

Une pression systolique trop élevée indique une hypertension.

Une pression trop faible indique une hypotension.

Le programme examine chaque observation et attribue une catégorie médicale. Le résultat devient soit normal soit anormal.

Cette étape reproduit un système d'aide à la décision médicale automatisé.

## 6. Traitement et stockage des résultats

Après l'analyse, les données suivent deux chemins différents.

Les mesures normales sont sauvegardées dans des fichiers JSON locaux. Cela représente un archivage standard de données patient.

Les mesures anormales sont indexées dans Elasticsearch. Chaque document contient les valeurs et le type d'anomalie détectée.

Elasticsearch agit ici comme une base orientée recherche permettant de surveiller les cas critiques. Il offre une consultation rapide et prépare la visualisation.

Cette distinction entre normal et anormal correspond au fonctionnement réel d'un système hospitalier où seules les alertes nécessitent une attention immédiate.

## 7. Visualisation avec Kibana

Kibana est utilisé pour analyser graphiquement les anomalies détectées. Les données indexées sont transformées en graphiques temporels et statistiques.

Le tableau de bord permet notamment d'observer :

- l'évolution des anomalies dans le temps
- la distribution des pressions artérielles
- la proportion de cas critiques

Grâce à la mise à jour automatique, l'interface agit comme un écran de monitoring hospitalier. Cela permet de comprendre rapidement l'état général des patients simulés.

## 8. Reproductibilité avec Docker

Le projet est entièrement exécuté via des conteneurs Docker. Chaque composant fonctionne dans un environnement isolé : Kafka, Elasticsearch et Kibana.

Cette approche garantit que le projet fonctionne sur n'importe quelle machine sans configuration complexe. Il suffit de lancer les services pour recréer l'infrastructure complète.

La conteneurisation constitue une pratique essentielle dans les systèmes Big Data modernes.

## 9. Conclusion

Ce projet démontre la mise en place d'un système complet de surveillance médicale en temps réel. Il combine génération de données, transport distribué, analyse automatique et visualisation.

L'utilisation du standard FHIR assure la cohérence avec les systèmes médicaux réels. Kafka permet la gestion du flux continu. Elasticsearch et Kibana fournissent l'observation et le suivi des anomalies.

Le système reproduit le fonctionnement d'une plateforme de télémédecine capable de détecter automatiquement des situations critiques. Ce type d'architecture peut être étendu pour intégrer de véritables capteurs médicaux ou un modèle d'intelligence artificielle afin d'améliorer la précision du diagnostic.