Filière

Business Intelligence and Analytics

(Bi&a)

Rapport du projet IR

Projet:

Analyse et Visualisation des Logs Firefox
Utilisation du Stack ELK.

Encadré par :

M. Noureddine Kerzazi

Réalisé par :

M. MONCIF Mouad

M. OUHNI Kamal

Année universitaire 2025-2026

1 Introduction

Ce rapport présente l'implémentation d'une solution complète d'analyse et de visualisation des logs de build Firefox en utilisant la stack ELK (Elasticsearch, Logstash, Kibana). Le projet couvre l'ingestion des données, leur traitement, leur indexation et leur visualisation à travers des dashboards interactifs, complétés par des fonctionnalités de Machine Learning pour la détection d'anomalies.

2 Architecture du Projet

2.1 Vue d'Ensemble

L'architecture mise en place repose sur une approche conteneurisée utilisant Docker, permettant un déploiement cohérent et reproductible de l'ensemble des composants de la stack ELK.

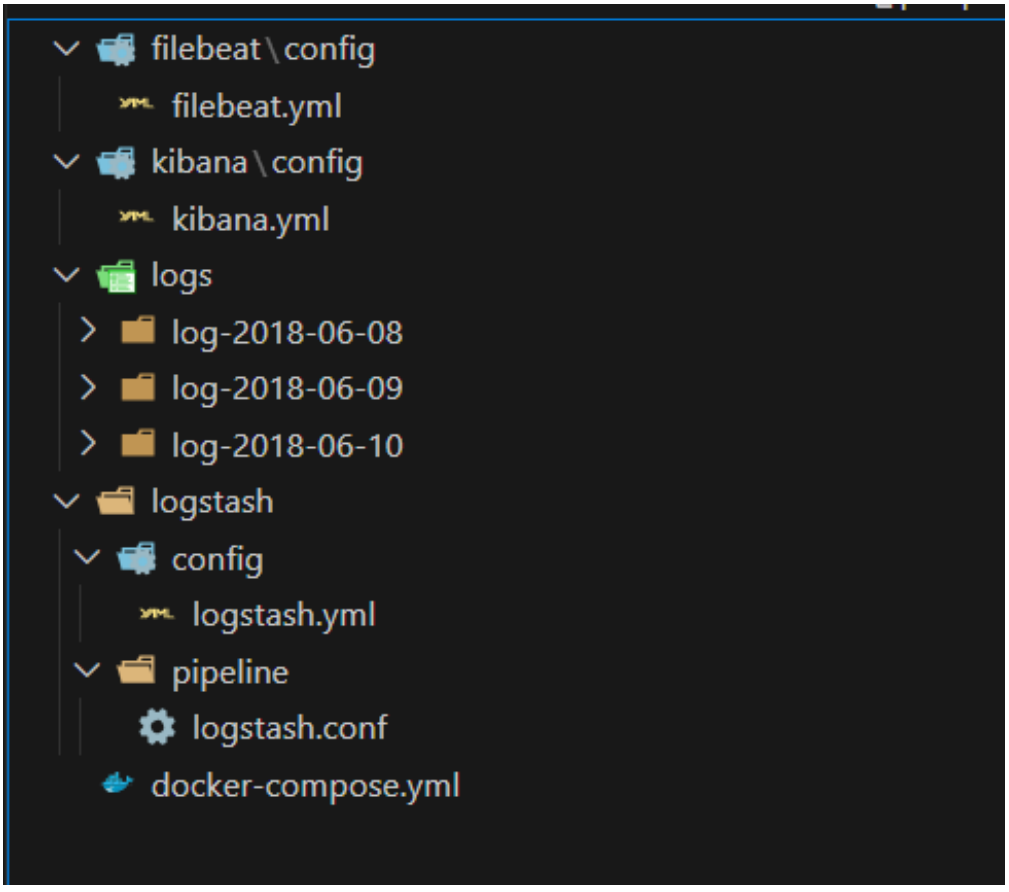


FIGURE 1 – Structure du projet avec les composants principaux

2.2 Composants de l'Architecture

L'architecture se compose des éléments suivants :

Filebeat : Agent léger de collecte de logs qui surveille les fichiers de logs et transmet les données vers Logstash.

Logstash : Pipeline de traitement qui parse, enrichit et transforme les logs avant leur indexation.

Elasticsearch : Moteur de recherche et d'analyse qui stocke et indexe les données.

Kibana : Interface de visualisation et d'exploration des données indexées.

Docker Compose : Orchestration des conteneurs pour un déploiement simplifié.

3 Ingestion des Données

3.1 Configuration de Filebeat

Filebeat a été configuré pour surveiller les fichiers de logs situés dans le répertoire `logs/log-*/`, avec une gestion automatique de la rotation des fichiers et du multiline parsing pour les logs complexes.

3.2 Pipeline Logstash

Le pipeline Logstash effectue les opérations suivantes :

— Réception des événements depuis Filebeat
— Parsing des logs avec des filtres Grok
— Extraction des métadonnées (builder, slave, timestamps, résultats)
— Détection des erreurs et warnings
— Enrichissement avec des tags (error, warning, anomaly)
— Envoi vers Elasticsearch

3.3 Résultat de l'Ingestion

Après la configuration et le démarrage des services, l'ingestion des logs a été réalisée avec succès.

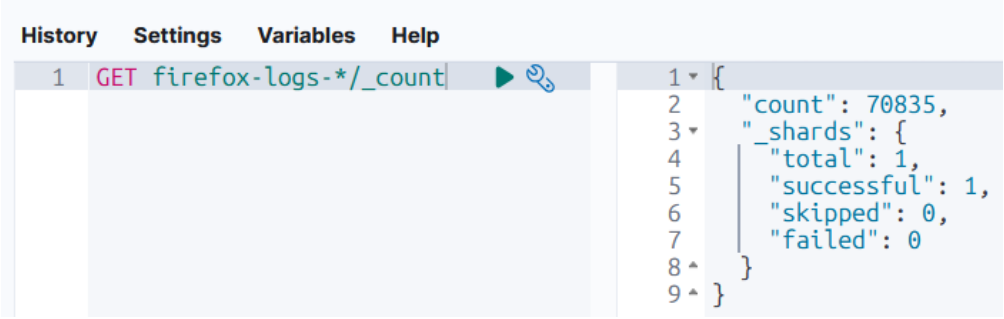


FIGURE 2 – Confirmation du nombre de documents ingérés dans Elasticsearch

4 Mapping Elasticsearch

4.1 Définition du Mapping

Un mapping personnalisé a été créé pour optimiser l'indexation et les performances de recherche. Le mapping définit les types de champs, les analyseurs et les propriétés d'indexation.

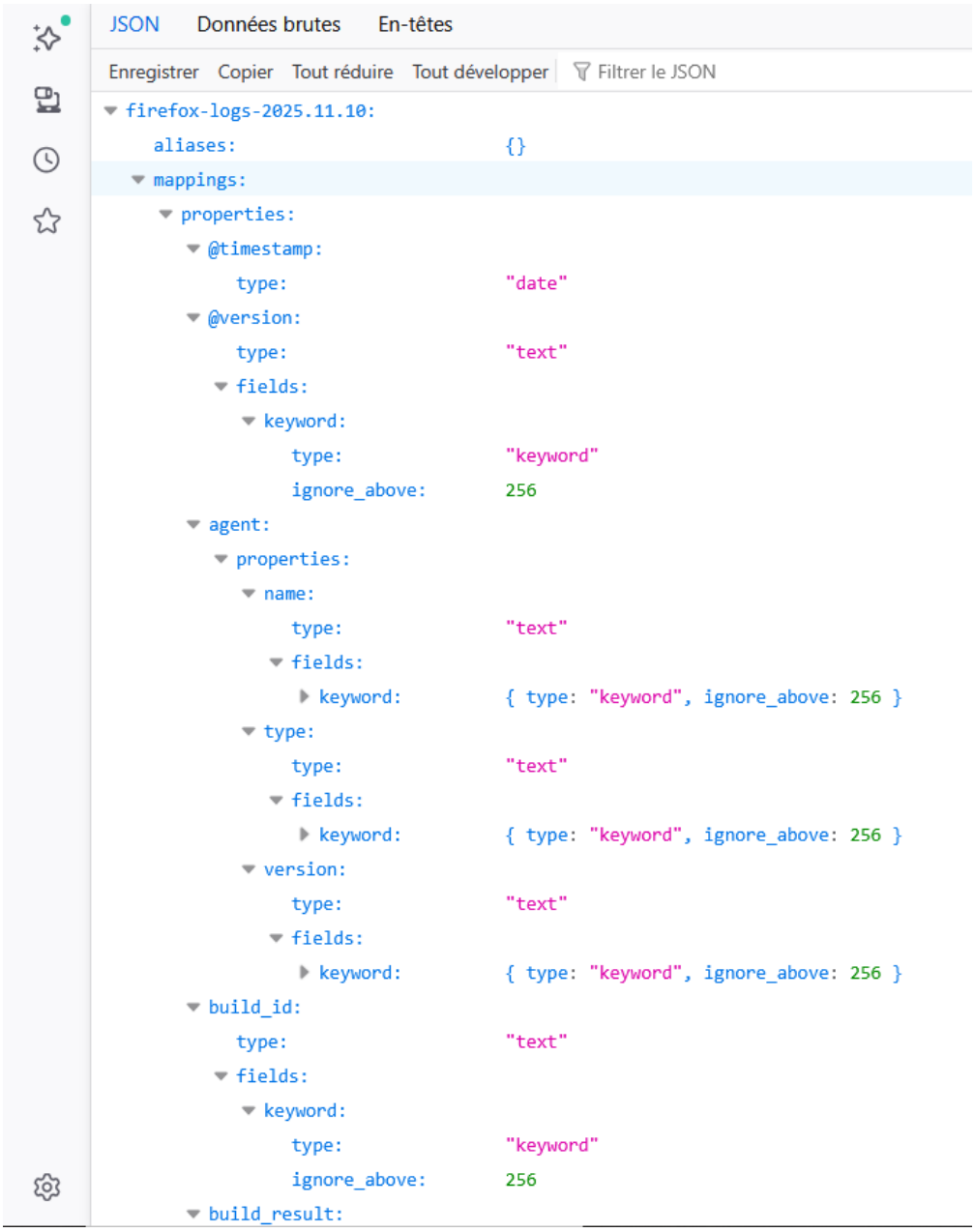


FIGURE 3 – Configuration du mapping Elasticsearch pour l'index firefox-log

Le mapping inclut :
— Types de champs appropriés (keyword, text, date, float, integer)
— Index pattern : `firefox-log-*`
— Configuration des champs analysés et non-analysés
— Optimisation pour les aggregations et recherches

5 Visualisations Kibana

Trois dashboards principaux ont été créés pour analyser les logs sous différents angles.

5.1 Dashboard : Firefox Builds - Overview

Ce dashboard fournit une vue d'ensemble de l'activité des builds Firefox, permettant d'identifier rapidement les tendances globales et les métriques clés.

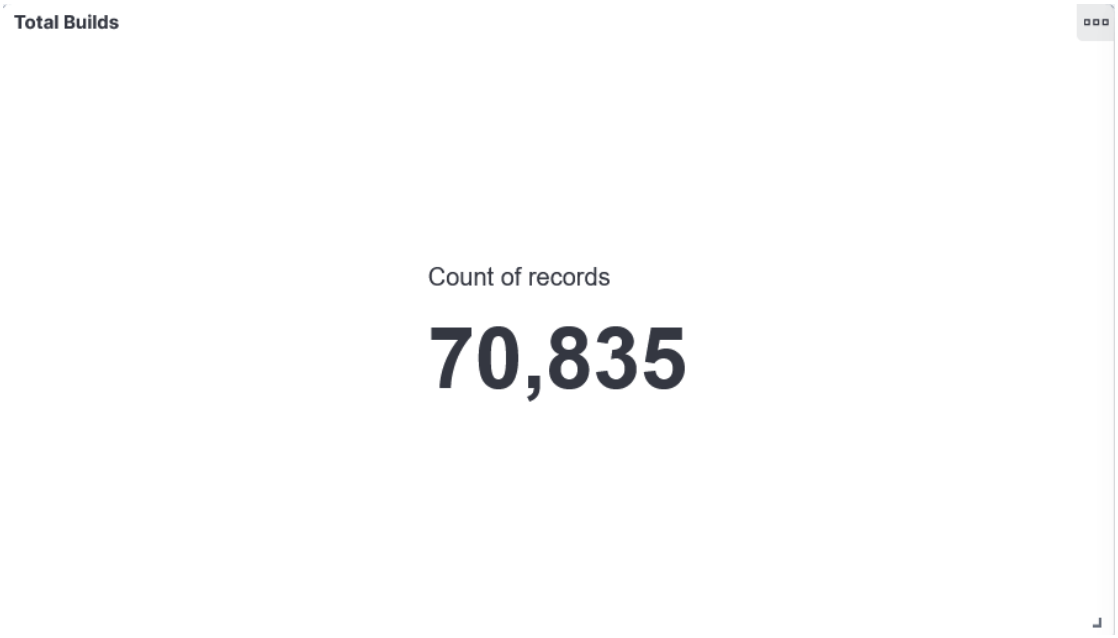


FIGURE 4 – Métrique du nombre total de builds

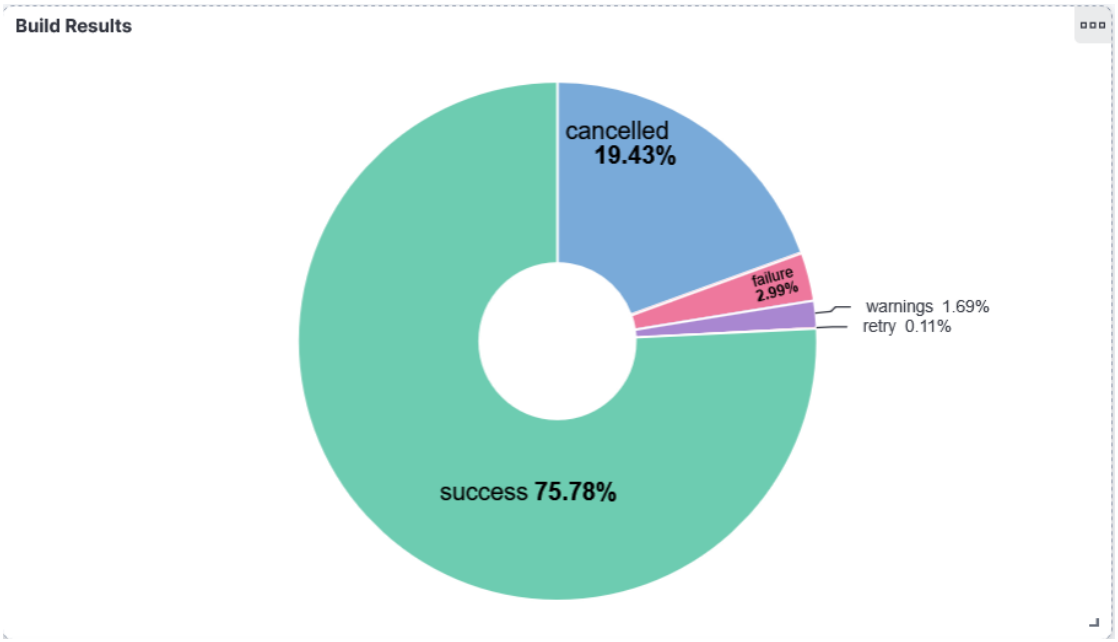


FIGURE 5 – Gauge du taux de succès des builds

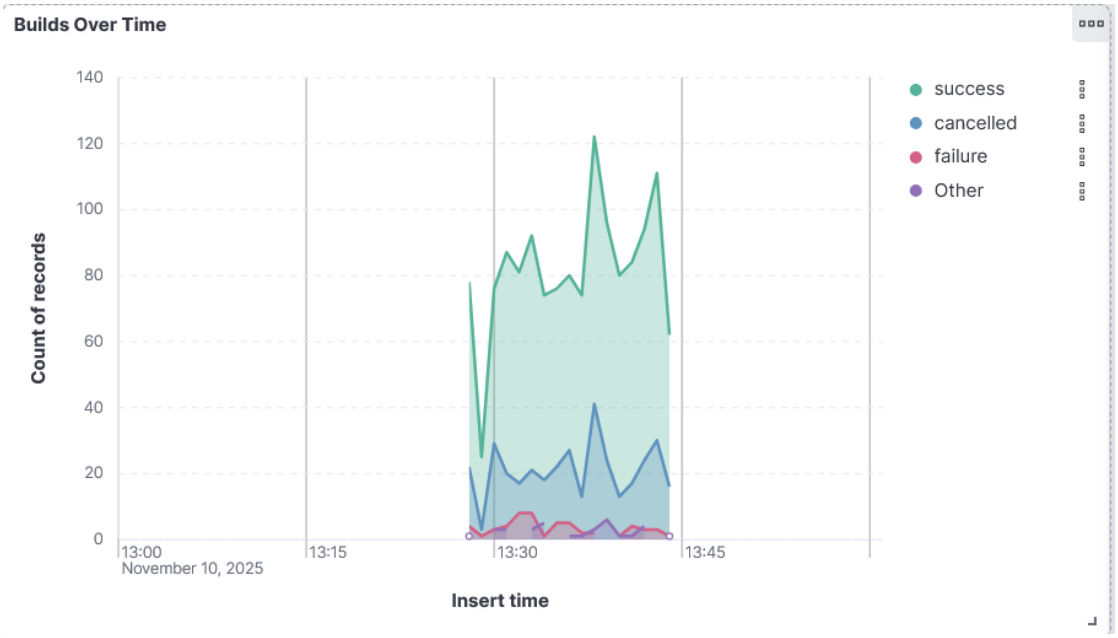
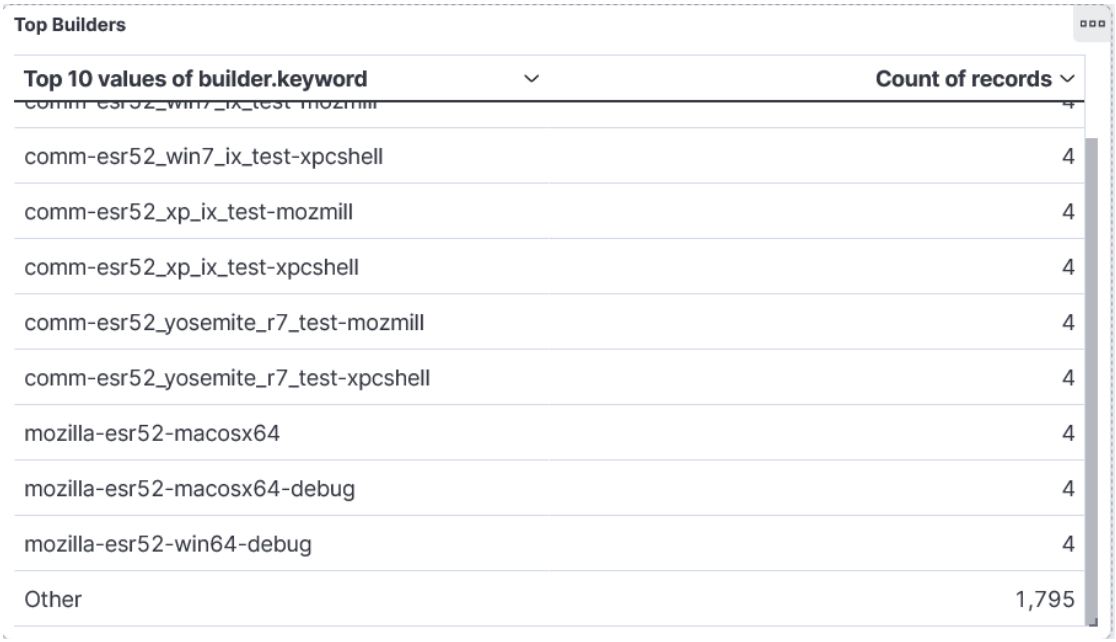


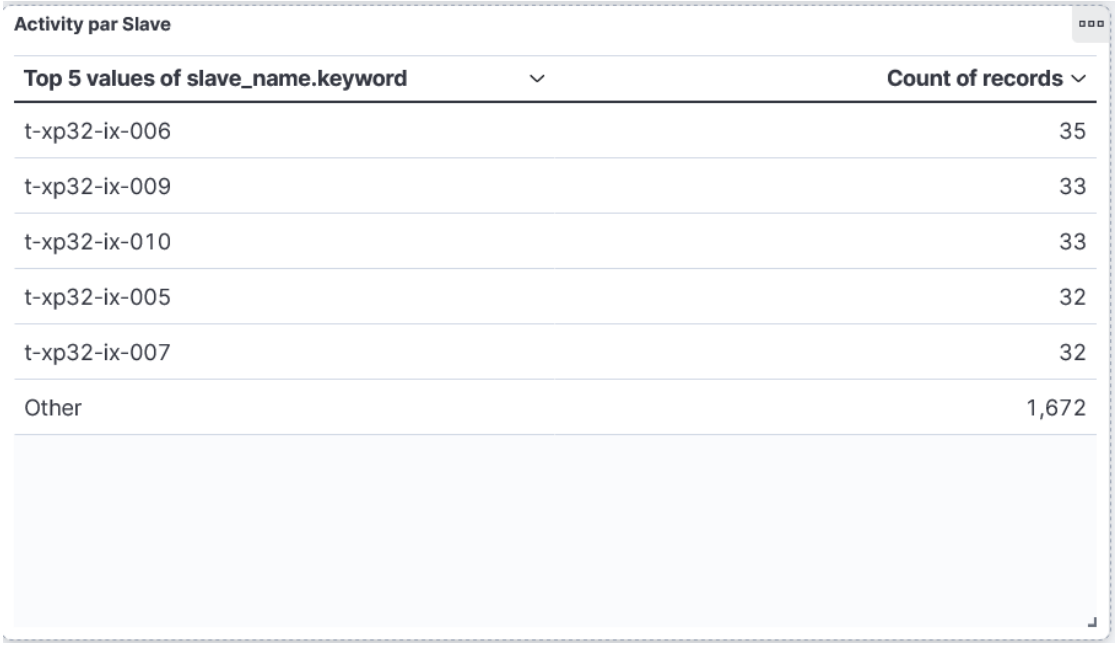
FIGURE 6 – Distribution des résultats de builds



Top Builders

Top 10 values of builder.keyword	Count of records
comm-esr52_win7_ix_test-xpcshell	4
comm-esr52_xp_ix_test-mozmill	4
comm-esr52_xp_ix_test-xpcshell	4
comm-esr52_yosemite_r7_test-mozmill	4
comm-esr52_yosemite_r7_test-xpcshell	4
mozilla-esr52-macosx64	4
mozilla-esr52-macosx64-debug	4
mozilla-esr52-win64-debug	4
Other	1,795

FIGURE 7 – Top 10 builders par volume d'activité



Activity par Slave

Top 5 values of slave_name.keyword	Count of records
t-xp32-ix-006	35
t-xp32-ix-009	33
t-xp32-ix-010	33
t-xp32-ix-005	32
t-xp32-ix-007	32
Other	1,672

FIGURE 8 – Top 5 slaves par volume d'activité

5.2 Dashboard : Firefox Builds - Performance

Ce dashboard se concentre sur l'analyse des performances, permettant d'identifier les goulots d'étranglement et les opportunités d'optimisation.

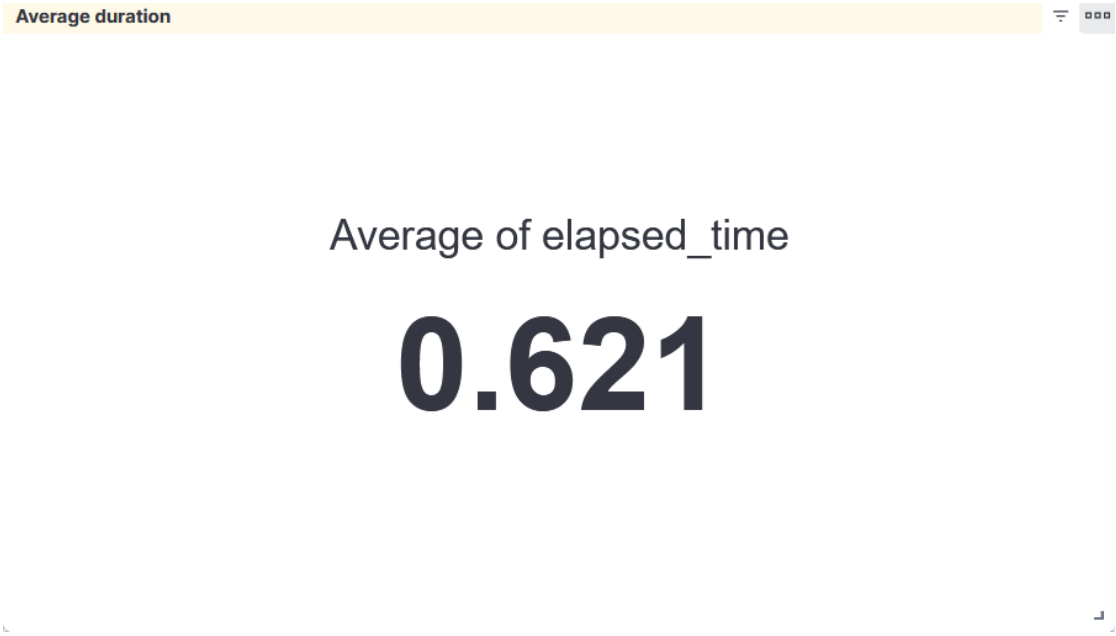


FIGURE 9 – Métrique de la durée moyenne des steps

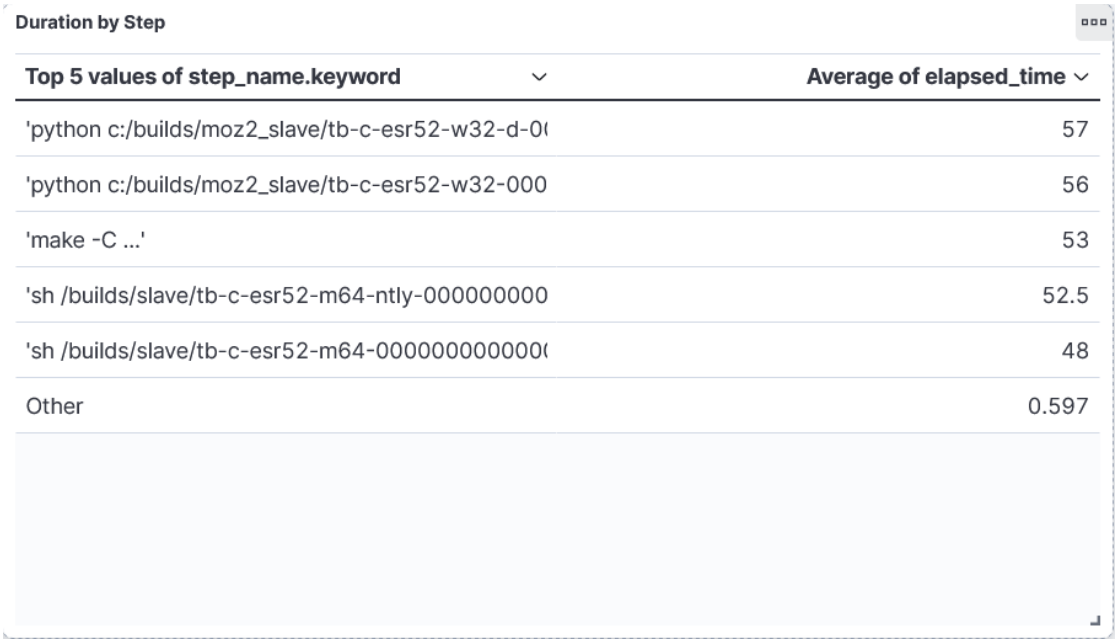


FIGURE 10 – Top 5 des steps les plus lentes

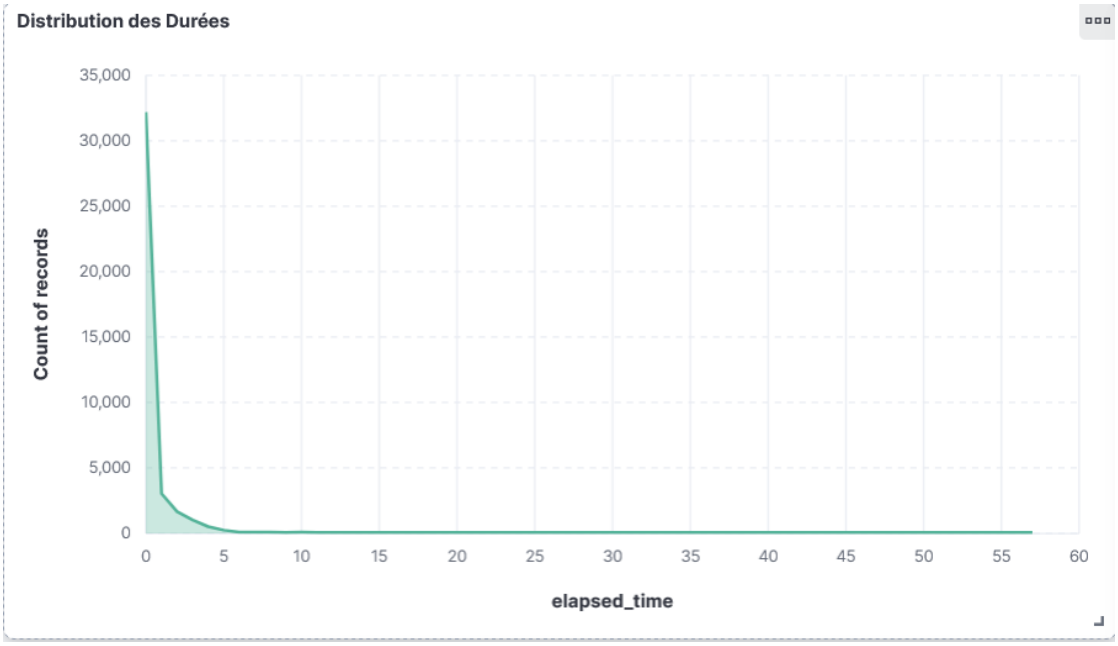


FIGURE 11 – Distribution des durées par le temps

FIGURE 12 – Distribution des durées par temps d'insertion

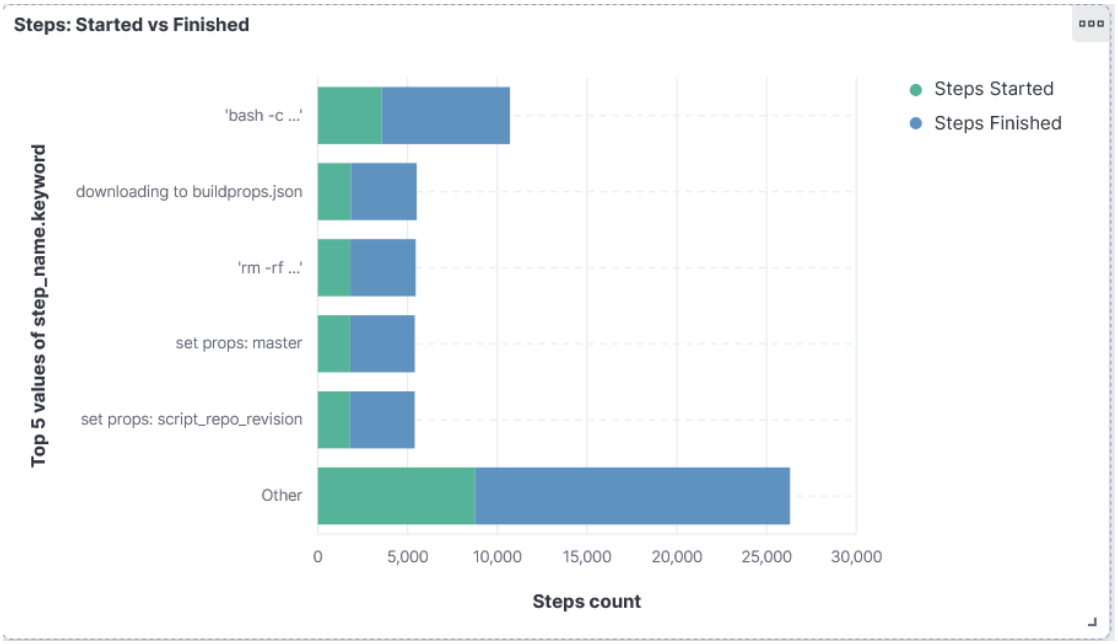


FIGURE 13 – Top 5 des steps par steps started et steps finished

5.3 Dashboard : Firefox Builds - Anomalies Detection

Ce dashboard permet d'identifier rapidement les anomalies et les problèmes nécessitant une attention immédiate.

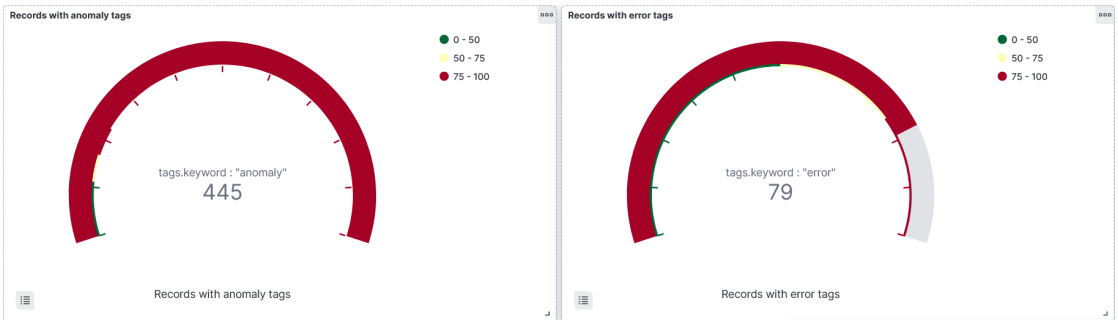


FIGURE 14 – Métriques des erreurs détectées

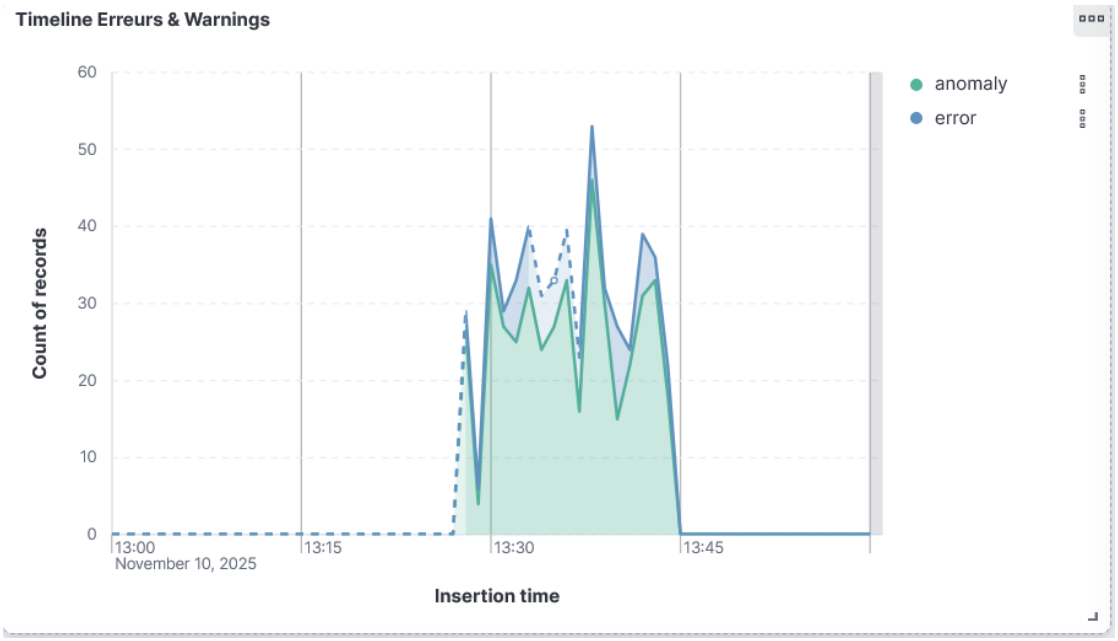


FIGURE 15 – Timeline des erreurs et warnings

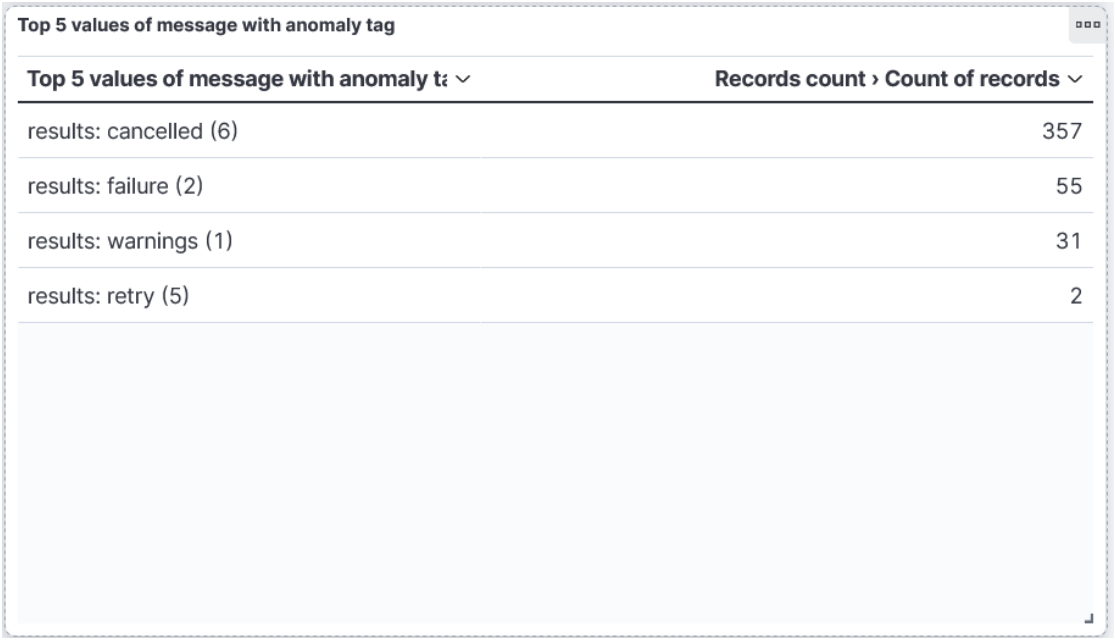


FIGURE 16 – Tableau des top messages d'erreurs

FIGURE 17 – Builds en échec avec détails

6 Machine Learning

6.1 Détection d'Anomalies

Un job de Machine Learning a été configuré pour détecter automatiquement les durées d'exécution anormales des steps de build. Le job utilise une fonction de moyenne (`mean`) sur le champ `elapsed_time` avec un bucket span de 15 minutes.

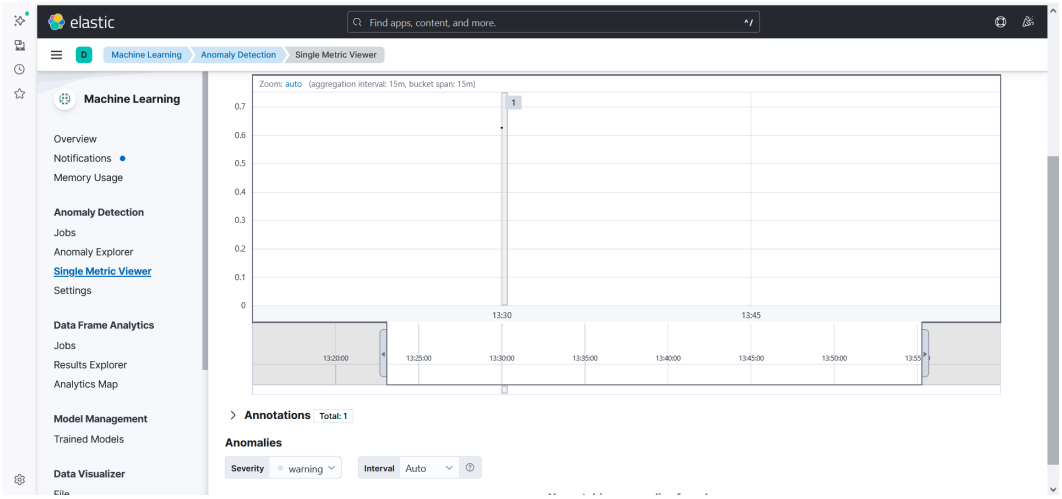


FIGURE 18 – Configuration du job Machine Learning pour la détection des durées anormales

6.2 Fonctionnement

Le job ML analyse continuellement les données et :
— Établit un modèle de comportement normal basé sur l'historique
— Détecte les variations significatives de la durée moyenne
— Assigne un score de sévérité aux anomalies détectées
— Permet la création d'alertes automatiques pour les anomalies critiques

7 Résultats et Analyses

7.1 Métriques Clés

L'analyse des logs a permis d'extraire les métriques suivantes :
— Nombre total de builds traités
— Taux de succès global des builds
— Distribution des résultats (succès, échecs, warnings)
— Durée moyenne d'exécution des différents steps
— Identification des builders les plus actifs
— Détection des patterns temporels d'activité

7.2 Insights Opérationnels

Les dashboards permettent d'identifier :
— Les steps nécessitant une optimisation (durées élevées)

— Les patterns d'erreurs récurrents
— Les variations de performance dans le temps
— Les corrélations entre builders et types d'erreurs
— Les anomalies nécessitant une investigation

8 Technologies Utilisées

8.1 Stack ELK

Elasticsearch 8.11.0 : Moteur de recherche et d'analyse distribué, stockage et indexation des logs.

Logstash 8.11.0 : Pipeline de traitement de données avec filtres Grok pour le parsing.

Kibana 8.11.0 : Interface de visualisation avec support du Machine Learning.

Filebeat 8.11.0 : Agent léger de collecte et forwarding des logs.

8.2 Infrastructure

Docker : Conteneurisation des services pour portabilité et isolation.

Docker Compose : Orchestration multi-conteneurs avec gestion des dépendances.

9 Conclusion

Ce projet démontre l'efficacité de la stack ELK pour l'analyse à grande échelle des logs de build Firefox. L'architecture mise en place permet :

— Une ingestion automatisée et continue des logs
— Un traitement et enrichissement intelligent des données
— Une visualisation intuitive et interactive
— Une détection proactive des anomalies via Machine Learning
— Une base solide pour l'amélioration continue des processus de build

La solution est scalable et peut être étendue pour inclure d'autres sources de logs, des alertes automatiques avancées, et des analyses prédictives supplémentaires.