DOSSIER DE MAINTENANCE

Projet Amazon Reviews :

Analyse et classification des avis Clients

BONNAT Jonathan

Data Engineer - B2B-LP-DESFL

💬 INTRODUCTION GENERALE

Ce dossier de maintenance décrit l'ensemble des procédures nécessaires pour assurer la continuité de fonctionnement, la fiabilité et l'évolution du système d'analyse et de classification des avis clients. Ce document complète le dossier d'accompagnement utilisateur en fournissant une vision opérationnelle centrée sur la gestion des incidents, la maintenance corrective et préventive, ainsi que la planification des évolutions futures.

Le périmètre couvre l'ensemble des composants déployés en production : pipeline Airflow, services Docker, API FastAPI, stockage S3 (Data Lake), base de données PostgreSQL (source) , base de données MongoDB (archivage) et la couche NLP intégrée à l'ETL.

Les procédures décrites s'adressent principalement aux équipes Data Engineering, DataOps et DevOps responsables de l'exploitation et du maintien en condition opérationnelle du système.

L'objectif est de fournir un cadre clair, reproductible et actionnable pour diagnostiquer les anomalies, appliquer les correctifs, prévenir les dégradations de performance et accompagner les futures évolutions fonctionnelles ou techniques du système, tout en garantissant la conformité réglementaire et la qualité des données.

GESTION DES INCIDENTS

La gestion des incidents a pour objectif d'assurer la continuité de service du système d'analyse des avis clients en production. Elle regroupe les pratiques permettant d'identifier, diagnostiquer, corriger et documenter les anomalies affectant les composants du pipeline : Airflow, stockage S3, PostgreSQL, MongoDB, Fast API et infrastructure Docker.

Cette section repose sur les principes ITIL simplifiés et adaptés au contexte Data/ETL.

TYPOLOGIE DES INCIDENTS

Les incidents sont classés selon leur gravité, leur impact opérationnel et la rapidité d'intervention requise.

- **Incident mineur :**
  - Impact faible sur le système ou les utilisateurs.
  - Exemples : augmentation légère du temps d'exécution du DAG, latence API plus élevée qu'à l'habitude, warnings dans les logs.
  - Traitement : correction simple ou observation.
- **Incident majeur :**
  - Fonctionnalité dégradée, mais le service reste partiellement accessible.
  - Exemples : DAG qui échoue sur une tâche non critique, erreur ponctuelle de chargement parquet, pics anormaux de rejets ETL.
  - Traitement : intervention sous quelques heures + escalade N1 → N2.
- **Incident critique :**
  - Interruption ou impossibilité d'utiliser le système.
  - Exemples : DAG en échec complet, fichier Parquet manquant, API FastAPI indisponible, corruption de données, S3 inaccessible.
  - Traitement : intervention immédiate + escalade N2 → N3.

PROCESSUS DE TRAITEMENT DES INCIDENTS

Une procédure standardisée permet une prise en charge homogène et efficace :

1. **Détection :**
   - Monitoring Splunk
   - Alertes Airflow
   - Tests automatisés (/health)
   - Signalement par un utilisateur (analyste, développeur)

2. **Diagnostic :**
    - le composant défaillant (Airflow, API, S3, etc.)
    - les symptômes (logs, erreurs HTTP, état du DAG)
    - la cause probable (erreur réseau, credentials invalides, bug NLP, etc.)
3. **Correction :**
    - relance d'un DAG
    - redémarrage d'un conteneur
    - correction de connexion ou credentials
    - nettoyage de données corrompues
    - reprise de traitement ETL
4. **Validation et clôture :**
    - vérification du retour à la normale
    - analyse REX si incident majeur
    - documentation dans le journal des incidents
    - mise à jour de la matrice des risques si nécessaire

PRIORISATION (IMPACT + URGENCE)

La matrice suivante permet de déterminer la priorité de traitement :

↓ IMPACT / URGENCE →	FAIBLE	MOYEN	ÉLEVÉ
FAIBLE	P3	P3	P2
MOYEN	P3	P2	P1
ÉLEVÉ	P2	P1	P0

- **P0** (Critique) : intervention immédiate
- **P1** (Haut) : intervention < 4h ouvrées
- **P2** (Normal) : correction dans la journée
- **P3** (Faible) : correction planifiée

PROCESSUS D'ESCALADE

- **Niveau 1 : Support technique / Analyste**
    - Vérifie le problème
    - Relève les symptômes
    - Ouvre un ticket
    - Tente les actions simples (test /health, consultation logs API)
- **Niveau 2 : Data Engineer / DataOps**
    - Diagnostic approfondi du pipeline
    - Analyse Airflow, S3, MongoDB
    - Correction technique (relance DAG, correctifs ETL)
- **Niveau 3 : DevOps / Administrateur système**
    - Intervention sur :
        - environnement Docker
        - réseau
        - ressources machine
        - configuration avancée
    - Gestion des erreurs critiques d'infrastructure

L'escalade peut être directe en cas de :

- d'arrêt complet de l'API
- d'échec persistant du DAG après retries
- d'impossibilité de lire/écrire S3
- de corruption de données

GESTION ET JOURNALISATION DES INCIDENTS

La gestion des incidents s'appuie sur un outil de ticketing dédié (ex. Jira Service Management), permettant de centraliser l'ensemble des anomalies détectées sur le pipeline Airflow, l'API FastAPI, le stockage S3, les bases de données ou l'infrastructure Docker.

Chaque incident fait l'objet d'un ticket unique servant de référence pour l'analyse, le traitement et la traçabilité. Le ticket doit contenir au minimum :

- ID de l'incident
- Date et heure de détection

- Composant impacté
- Description du symptôme
- Logs associés (Airflow, API, Docker, Splunk)
- Priorité (P0–P3)
- Actions correctives appliquées
- Résultat et validation
- Responsable de la résolution

L'outil de ticketing permet également :

- la gestion des escalades (N1 → N2 → N3)
- le suivi de l'historique
- l'analyse des incidents récurrents (aussi appelés problèmes)
- l'identification des actions de maintenance préventive nécessaires
- le reporting opérationnel

🏗️ MAINTENANCE CORRECTIVE

La maintenance corrective regroupe l'ensemble des actions menées pour rétablir le fonctionnement normal du système suite à un incident détecté. Elle s'applique aux composants critiques du pipeline : Airflow, API FastAPI, stockage S3, PostgreSQL, MongoDB, services Docker et modèles NLP. Elle vise à restaurer les services rapidement tout en garantissant l'intégrité des données.

SITUATIONS D'INCIDENT FRÉQUENTES ET PROCÉDURES DE CORRECTION

Les incidents ci-dessous sont les plus courants dans l'environnement d'orchestration ETL / NLP. Pour chacun, la procédure suit un schéma systématique : symptômes → diagnostic → correctif → validation.

Échec d'un DAG Airflow :

- Symptômes :
    - tâche en rouge dans Airflow
    - DAG bloqué en "failed" malgré les retries

        - logs Airflow contenant des erreurs SQL, S3, ou NLP
- Diagnostic :
    - consulter les logs de la tâche échouée
    - vérifier la connexion PostgreSQL ou S3
    - vérifier les credentials (Airflow Connections)
    - vérifier si le Worker manque de RAM (modèle NLP)
- Actions correctives :
    - relancer uniquement la tâche échouée
    - nettoyer les fichiers temporaires (S3 → rejects)
    - redémarrer le worker si OOM
    - mettre à jour les credentials dans Airflow si expirés
- Validation :
    - DAG exécuté entièrement
    - mise à jour du Parquet S3
    - absence de logs d'erreur sur le run suivant

Parquet S3 non mis à jour / fichier absent :

- Symptômes :
    - date de dernière mise à jour > 24h
    - fichier corrompu (erreur de lecture côté API)
    - taille du fichier trop faible
- Diagnostic :
    - logs de la tâche load_cleaned ou load_curated
    - vérifier le chemin exact S3
    - vérifier les permissions IA
    - vérifier la présence d'un fichier temporaire
- Actions correctives :
    - relancer uniquement la tâche de load
    - régénérer le fichier Parquet depuis les données brutes
    - si corrompu : supprimer le fichier + relancer le DAG
- Validation :
    - Parquet lisible
    - API /top renvoie les données attendues

API FastAPI indisponible :

- Symptômes :
  - /health renvoie une erreur
  - logs contenant FileNotFoundError, S3Error, ou ImportError
  - erreur 500 renvoyée au frontend
- Diagnostic :
  - docker compose logs api
  - test direct de lecture du fichier Parquet depuis un shell
  - test de dépendances (transformers, pyarrow)
- Actions correctives :
  - redémarrer le conteneur API
  - valider les credentials d'accès S3
  - reconstruire l'image Docker si dépendance manquante
- Validation :
  - /health → {"status": "ok"}
  - endpoint /reviews/{id}/top fonctionnel

Erreurs MongoDB lors de l'archivage :

- Symptômes :
  - exceptions PyMongo
  - refus d'insertion
  - champs manquants ou invalides
- Diagnostic :
  - vérifier la disponibilité du service Mongo
  - consulter les logs load_mongodb
  - vérifier le schéma attendu dans la collection
- Actions correctives :
  - redémarrer MongoDB (docker compose restart mongo)
  - nettoyer les documents corrompus
  - réindexer la collection au besoin
- Validation :
  - documents insérés correctement
  - absence d'erreur lors du prochain run Airflow

Volume élevé de rejets ETL :

- Symptômes :
  - augmentation du dossier rejects/
  - anomalie dans les valeurs (rating = null, product_id manquant)
- Diagnostic :
  - analyser les fichiers CSV de rejet
  - vérifier l'intégrité des données sources
  - rechercher un changement de schéma
- Actions correctives :
  - adapter le code ETL si un champ a changé
  - ajouter des règles de normalisation supplémentaires
  - contacter l'équipe applicative si la source est incorrecte
- Validation :
  - taux de rejet revenu au niveau normal
  - aucun rejet bloquant

SCRIPTS D'EXPLOITATION DISPONIBLES

Les scripts ci-dessous seront destinés à assister les DataOps dans la résolution d'incidents récurrents.

SCRIPT	FONCTION	USAGE
check_parquet.py	vérifie l'existence et la cohérence des fichiers parquet	Diagnostic S3
clean_logs.sh	archive / nettoie les anciens logs Airflow	Maintenance préventive
restart_services.sh	redémarre toute la stack Docker	Incident API / DAG

MAINTENANCE PREVENTIVE

La maintenance préventive vise à éviter l'apparition d'incidents en anticipant les dégradations de performance, les dérives de stockage, les erreurs de configuration ou les dépendances obsolètes. Elle s'appuie sur des actions planifiées, des contrôles réguliers et une supervision continue du système.

TÂCHES RÉGULIÈRES

Routine quotidienne :

Objectif : garantir la disponibilité du pipeline et de l'API.

- Vérification du statut des DAG Airflow
- Vérification du statut /health de l'API FastAPI
- Surveillance des erreurs critiques Splunk (alerte P0/P1)
- Vérification de la mise à jour du fichier Parquet (timestamp + taille)
- Contrôle du taux de rejets ETL

Livrable attendu : aucun incident non pris en compte en fin de journée.

Routine hebdomadaire :

Objectif : limiter les dérives et prévenir les erreurs structurelles.

- Nettoyage des logs Airflow (rotation automatique)
- Contrôle des connexions Airflow (DB, S3, Mongo)
- Vérification des quotas et volumétrie du bucket S3
- Redémarrage contrôlé des services Docker (hors production critique)
- Vérification de l'état des index sur MongoDB

Livrable attendu : stabilité du pipeline confirmée + capacité de stockage maîtrisée.

Routine mensuelle :

Objectif : anticiper les dégradations long-terme et maintenir la fiabilité du système.

- Audit global du pipeline (temps d'exécution, performance NLP)
- Analyse des incidents du mois (récurrence, patterns)
- Purge des rejets ETL de plus de 30 jours
- Mise à jour des dépendances majeures (transformers, pandas, pyarrow)
- Vérification de la conformité RGPD et suppression des données sensibles périmées
- Vérification des accès RBAC (droits obsolètes, comptes inactifs)

Livrable attendu : rapport de maintenance préventive mensuel.

SURVEILLANCE ET OPTIMISATION CONTINUE

Des dashboards Splunk surveillent en continu :

- les erreurs API (HTTP 4xx / 5xx)
- les logs Airflow critiques
- les tailles de fichier Parquet
- le volume des rejets ETL
- les anomalies NLP (temps d'inférence, modèle indisponible)

Des alertes automatiques (e-mail) sont générées pour :

- API down
- DAG échoué deux fois consécutivement
- absence de mise à jour du Parquet > 24h
- dépassement du volume S3 attendu

TESTS RÉGULIERS DE CONFORMITÉ OPÉRATIONNELLE

Des tests techniques automatisés permettent de garantir la stabilité et la cohérence du système.

Tests API :

- /health doit répondre en < 200 ms
- /reviews/{id}/top doit renvoyer un résultat valide

Tests Data Lake (S3) :

- Lecture du Parquet par un test Airflow dédié
- Vérification de la cohérence du schéma
- Validation de la taille minimale du fichier

Tests NLP :

- Test rapide d'inférence pour valider la disponibilité du modèle
- Alerte en cas de temps d'inférence > seuil défini

Bénéfice : détection anticipée des problèmes avant impact sur le pipeline.

NETTOYAGE, ARCHIVAGE ET OPTIMISATION

Pour éviter la dégradation des performances, les actions suivantes sont mises en place :

- Rotation automatique des logs Airflow (> 7 jours)
- Purge des rejets ancien > 30 jours
- Archivage MongoDB sur collections datées
- Suppression des fichiers temporaires et checkpoints inutilisés
- Optimisation et réduction des index MongoDB si fragmentation élevée
- Compression optimisée du Parquet (Snappy)

Ces opérations garantissent la stabilité du système et sa capacité à absorber la croissance des données.

MAINTENANCE EVOLUTIVE

La maintenance évolutive vise à faire évoluer le système pour répondre aux nouveaux besoins métier, aux améliorations techniques ou aux contraintes réglementaires. Elle garantit la capacité du dispositif à rester pertinent dans la durée, en intégrant de nouvelles fonctionnalités, en optimisant les performances et en assurant la compatibilité avec les évolutions des outils utilisés (Airflow, FastAPI, NLP, S3, MongoDB)

RECUEIL ET QUALIFICATION DES DEMANDES D'ÉVOLUTION

Les demandes d'évolution proviennent principalement :

- des équipes métier (ajout de nouvelles catégories d'analyse, KPI supplémentaires)
- des analystes (besoin d'indicateurs supplémentaires ou de nouvelles métriques)
- des Data Engineers (optimisation du pipeline ou refonte du stockage)
- des responsables conformité (RGPD, AI Act, audit des données)

Les demandes sont centralisées dans l'outil de ticketing (ex : Jira), sous le type "Feature Request" avec les éléments suivants :

- description du besoin
- composant impacté (pipeline, API, stockage…)
- justification métier
- criticité
- estimation d'effort
- impacts potentiels sur les données

Chaque demande est ensuite évaluée par un comité technique (DataOps + Tech Lead Data Engineering).

EXEMPLES D'ÉVOLUTIONS POSSIBLES

Les évolutions peuvent prendre plusieurs formes :

Évolutions fonctionnelles :

- Ajout d'un nouveau thème NLP (ex. : "durabilité", "sécurité produit")

- Mise à jour du modèle zéro-shot avec une version plus performante
- Ajout d'un endpoint API (ex : /reviews/{id}/summary)
- Intégration de scores supplémentaires : sentiment, subjectivité, polarité

Évolutions techniques :

- Partitionnement du Data Lake (S3)
- Migration du stockage vers Iceberg / Delta Lake
- Optimisation du DAG Airflow (Task Groups, parallélisation NLP)
- Ajout d'un cache local pour réduire la latence de FastAPI

Évolutions réglementaires :

- Mise à jour des mécanismes d'anonymisation (RGPD)
- Ajout d'un endpoint pour la suppression complète des données d'un utilisateur
- Conformité au futur AI Act (explicabilité minimale du modèle NLP)

ANALYSE D'IMPACT

Avant toute évolution, une analyse d'impact est réalisée :

- Impact technique : modifications du code ETL, taille des conteneurs, dépendances NLP
- Impact data : modification du schéma Parquet, besoin de migration ou de refonte
- Impact infra : plus de RAM GPU/CPU pour NLP, mises à jour Docker
- Impact sécurité : mise à jour des droits RBAC, gestion de nouveaux champs sensibles
- Impact surveillance : nouveaux dashboards Splunk ou métriques API

Chaque évolution doit être accompagnée d'un plan de tests et d'un plan de rollback.

PROCESSUS DE MISE EN PRODUCTION D'UNE ÉVOLUTION

Le cycle standard de livraison est le suivant :

1. Développement :
    - Implémentation dans une branche dédiée

    - Tests unitaires + tests de charge si NLP modifié
    - Validation technique par le Data Engineer référent
2. Tests en environnement de staging :
    - Exécution du pipeline avec jeux de données réels
    - Vérification du bon fonctionnement des nouveaux endpoints
    - Comparaison des temps d'exécution du DAG
    - Validation de non-régression
3. Mise en production :
    - Déploiement des conteneurs (API / Workers)
    - Déploiement du code ETL
    - Mise à jour contrôlée du modèle NLP si nécessaire
    - Vérification du bon démarrage de l'infrastructure
4. Surveillance post-déploiement (48h) :
    - contrôle des logs Airflow
    - vérification des volumes S3
    - monitoring API via Splunk
    - vérification des rejets ETL

Toute anomalie déclenche soit un retour en staging, soit l'activation du plan de rollback.

GESTION DU CYCLE DE VIE DU MODÈLE NLP

Comme le projet repose sur du NLP, une maintenance évolutive spécifique aux modèles est indispensable :

- Validation régulière du modèle (mises à jour critiques)
- Mesure trimestrielle de la performance (cohérence des scores)
- Tests d'exactitude sur un échantillon de références
- Gestion d'un répertoire versionné des modèles (model_v1, model_v2, etc.)
- Procédure de rollback immédiat en cas de dérive du modèle

GESTION DES RISQUES

La gestion des risques vise à identifier, anticiper et réduire les menaces pouvant affecter la disponibilité, la qualité ou la conformité du système d'analyse des avis clients. Cette section présente les principaux risques opérationnels, techniques et réglementaires, ainsi que les mesures de mitigation associées. La matrice est réévaluée de manière trimestrielle ou après tout incident critique.

MATRICE DE RISQUES

RISQUE	DESCRIPTION	PROBA	IMPACT	NIVEAU	MESURES
Défaillance du modèle NLP	Mauvaise classification, dérive des scores, erreurs du modèle	Moyen	Élevé	**Critique**	Versioning des modèles, tests trimestriels, rollback possible
Indisponibilité S3 / Parquet corrompu	API incapable de lire les données, rupture de service	Faible	Élevé	**Majeur**	Tests automatiques quotidiens, régénération Parquet, monitoring Splunk
Échec récurrent du DAG Airflow	Pipeline bloqué, données non mises à jour	Moyen	Moyen	**Moyen**	Relance automatique, optimisation worker, supervision régulière
Montée en charge API	Latence élevée ou indisponibilité sous pic de trafic	Moyen	Moyen	**Moyen**	Cache local, optimisation dépendances, autoscaling (évolution future)
Problème de droits RBAC	Accès excessif ou insuffisant à certaines données	Faible	Moyen	**Modéré**	Revue mensuelle des accès, nettoyage comptes inactifs

Non-conformité RGPD / AI Act	Retard sur suppression données, audit non conforme	Faible	Élevé	**Critique**	Procédures de suppression complètes, audit RGPD mensuel
Corruption MongoDB / Index brisé	Archivage impossible, perte de traçabilité	Faible	Moyen	**Modéré**	Ré-indexation mensuelle, backup régulier
Dépendance forte au stockage unique	S3 devient point de défaillance unique	Faible	Élevé	**Critique**	Redondance multi-buckets (évolution), monitoring strict

STRATÉGIE DE MITIGATION

Les mesures de prévention appliquées sont :

- Surveillance renforcée via Splunk (incidents API, DAG, stockage)
- Automatisation des tests : lecture Parquet, vérification endpoints, disponibilité NLP
- Revue trimestrielle du modèle NLP et tests de performance
- Contrôles réguliers des autorisations RBAC
- Rotation et nettoyage des données (logs, rejets, archivage)
- Documentation systématique des incidents critiques pour mise à jour de la matrice

Les risques classés “Critiques” (NLP, RGPD, dépendance S3) font l’objet d’une surveillance prioritaire.

SUIVI ET MISE À JOUR DE LA MATRICE

La matrice des risques est revue :

- tous les trois mois
- après un incident majeur
- après toute évolution importante (mise à jour NLP, refonte Data Lake, mise en place DWH, modification API)
- à chaque audit réglementaire interne

🏁 CONCLUSION

Le système d'analyse et de classification des avis clients est désormais déployé en production et accompagné d'un ensemble complet de procédures garantissant sa stabilité, sa disponibilité et son évolution dans le temps. Ce dossier de maintenance formalise l'ensemble des pratiques nécessaires au maintien en condition opérationnelle : gestion des incidents, maintenance corrective et préventive, suivi des risques et intégration des évolutions fonctionnelles ou réglementaires.

Les méthodes décrites permettent aux équipes DataOps et Data Engineering d'intervenir efficacement, tout en assurant la fiabilité du pipeline Airflow, de l'API FastAPI, du stockage S3, de MongoDB et du modèle NLP. Grâce à ces processus, le système est capable de s'adapter aux besoins métier futurs et de garantir une exploitation durable et conforme du dispositif.

Ce document constitue la référence opérationnelle pour les équipes responsables de l'exploitation et doit être mis à jour régulièrement afin d'intégrer les retours d'expérience, l'évolution des technologies utilisées et les nouvelles exigences réglementaires.