Documentation de Maintenance

Projet : Amazon Review Analysis

Auteur : Amara NAIT SAIDI

Date : 12 novembre 2025

Version 1.0

Table des matières

1. Système de Gestion des Incidents

1.1. Vue d'ensemble

Le Système de Gestion des Incidents du pipeline ETL *Amazon Reviews Analysis* et de l'application de visualisation fournit une approche structurée pour détecter, classifier, diagnostiquer et résoudre les incidents en production.

Il s'appuie sur :

- un système d'alertes e-mail automatique,
- une journalisation centralisée (MongoDB),
- des tests automatisés (pytest),
- un monitoring continu des DAG Airflow,
- et un workflow d'escalade clair.

1.2. Matrice de Classification des Incidents

Gravité	**Définition**	**Temps de réponse**	**Escalade**	**Exemples**
P0 – Critique	Panne totale du système, impact complet pour tous les utilisateurs	15 min	Escalade immédiate à l'ingénieur d'astreinte	- Rupture de connexion Snowflake- Tous les DAG en échec- Application hors service
P1 – Haute	Fonctionnalité majeure dégradée, impact sévère	1 h	Escalade après 2 h	- Échec extraction S3 pour toutes les tables- MongoDB indisponible pour les rejets- Alertes e-mail inactives
P2 – Moyenne	Fonctionnalité limitée, contournement possible	4 h	Escalade après 24 h	- Extraction échouée pour 1 table- UI Streamlit lente- Tests de qualité échoués
P3 – Basse	Problème mineur, aucun impact immédiat	1 jour ouvré	Escalade après 1 semaine	- Verbosité excessive des logs- Avertissements mineurs- Défauts cosmétiques

1.3 Mécanismes de Détection des Incidents

1.3.1 Alertes Automatisées

Alerte 1 : Echec d'Extraction S3

Le système d'alerte se déclenche si une ou plusieurs tables ne sont pas extraite. Un mail est systématiquement envoyé afin d'Alerter et de prendre les actions nécessaire pour investiguer et corriger le problème.

Gravité : P1 si toutes les tables échouent.

P2 une ou une partie des tables échouent.

Alerte 3 : Echec de chargement Snowflake

Le système se déclenche si Aucune ligne n'est inséré dans le datawarehouse Snowflake.

Gravité : P1

1.3.2 Journalisation Centralisée (MongoDB)

Toutes les opérations du pipeline ETL sont systématiquement enregistrées. Cela permet de rechercher rapidement les erreurs et d'analyser les événements récents. Cette fonctionnalité est assurée par un système de gestion des logs et de suivi des données rejetées, avec stockage dans la base de données NoSQL **MongoDB**. Grâce à cette approche, l'équipe peut identifier efficacement les incidents, comprendre leur origine et prendre les mesures correctives appropriées.

1.3.3 Monitoring de la qualité des données (tests automatisés)

Une batterie de tests automatisées ont été mis en place afin de vérifier des points critiques :

- **Connexion PostgreSQL** — Gravité P0
- **Validité des notes (1-5)** — Gravité P2
- **Détection de doublons** — Gravité P2
- **Intégrité référentielle** — Gravité P1

1.4 Procédure de Réponse aux incidents

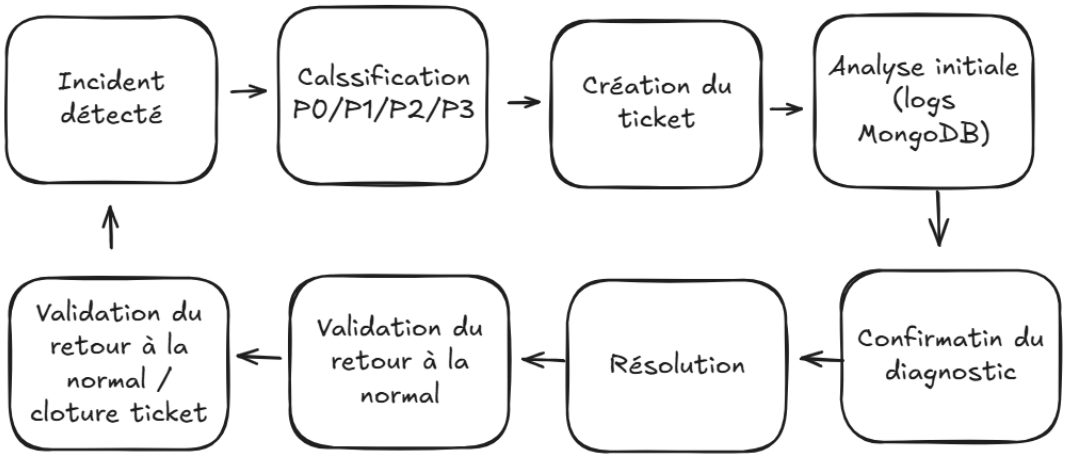

Figure 1 : Procédure Réponse aux incidents

1.4.1 Matrice d'Escalade

Rôle	**Responsabilités**	**Contact**	**Disponibilité**
L1 Support	Triage initial, diagnostics simples	Email / Slack	Heures ouvrées
Data Engineer	Pipeline, DAG, qualité des données	Slack / Tel	Heures ouvrées
DevOps Engineer	Airflow, AWS, infrastructure	Tel (P0/P1)	Astreinte
DBA	Snowflake / PostgreSQL / MongoDB	Tel (P0/P1)	Astreinte
Product Manager	Analyse d'impact	Email	Heures ouvrées

Déclencheurs :

- P0 : Escalade immédiate
- P1 : Après 2 h
- P2 : Après 24 h

- P3 : Après 1 semaine

1.4.2 Scénario d'incidents courant et résolution

Scénario 1 : Toutes les extractions S3 échouent, des alertes multiples, DAG complet en échec.

Actions recommandées :

- Vérifier la connectivité PostgreSQL.
- Contrôler les permissions S3 et la configuration Airflow.
- Relancer le DAG concerné.
- Créer et suivre le ticket dans JIRA pour documenter l'incident, les étapes effectuées et l'état de résolution.

Scénario 2 : Timeout de connexion Snowflake, la tâche de chargement échouée, erreur de timeout, un email d'Alerte a été reçu.

Actions recommandées :

- Vérifier la connectivité réseau et les statuts du Warehouse Snowflake.
- Relancer la connexion et ajuster les paramètres de timeout si nécessaire.
- Documenter toutes les étapes dans JIRA, y compris les actions correctives et l'état final.

Scénario 3 : Échec de l'application ou de l'API

Actions recommandées :

- Vérifier la connectivité Snowflake et les logs applicatifs.
- Redémarrer les services si nécessaire.
- Enregistrer l'incident dans JIRA, en incluant les logs et la séquence de diagnostic pour référence future.

1.4.3 Recommandations pour le Monitoring

Pour assurer la fiabilité et la performance du pipeline ETL et de l'application, il est essentiel de mettre en place un suivi rigoureux des indicateurs clés. Les principaux éléments à monitorer incluent le taux de succès des DAG, qui doit rester supérieur à 95%, la fraîcheur des données avec un décalage inférieur à 2 heures, le taux de rejet des données qui doit rester inférieur à 5 %, le temps de réponse aux alertes et la disponibilité globale des applications, qui doit excéder 99,9 %.

Ces métriques peuvent être suivies à l'aide d'outils spécialisés tels que l'interface Airflow pour les DAG, Grafana et Prometheus pour la supervision de l'infrastructure, MongoDB

Compass pour l'analyse des logs et des données rejetées, ainsi que l'historique des requêtes Snowflake pour la performance des chargements.

1.4.4 Model de Post-Incident Review

Chaque incident majeur doit faire l'objet d'une revue post-incident complète afin d'améliorer les processus et prévenir les récidives. Le modèle de Post-Incident Review doit inclure les informations suivantes : l'identifiant de l'incident, l'heure de détection, le temps de réponse initial et le temps de résolution total, la cause racine identifiée, l'impact sur les utilisateurs et le business, les actions correctives mises en œuvre, les mesures préventives adoptées et les leçons tirées pour améliorer la résilience future du système. Cette documentation permet de capitaliser sur l'expérience opérationnelle et d'alimenter le suivi des incidents dans JIRA pour un historique centralisé et exploitable.

2. Maintenance Corrective

2.1 Vu d'ensemble

La maintenance corrective du pipeline ETL Amazon Reviews et de l'application de visualisation consiste à diagnostiquer et réparer les défaillances identifiées en production. Cette section détaille les procédures de correction spécifiques aux composants du système, basées sur l'analyse approfondie de l'architecture technique déployée.

Le système comprend trois couches principales nécessitant des approches correctives distinctes :

- Couche d'extraction : Pipeline Airflow extrayant les données depuis PostgreSQL vers AWS S3
- Couche de transformation : Traitement, nettoyage et chargement vers Snowflake et MongoDB
- Couche applicative : API FastAPI et interface Streamlit pour la visualisation

2.2 Cartographie des défaillance par composant

2.2.1 Pipeline ETL – Extraction PostgreSQL vers S3

Criticité : Haute

Défaillance 1 : Connexion PostgreSQL impossible

- Manifestation : Échec du DAG dès la première tâche, erreurs de timeout ou d'authentification.
- Causes possibles : Conteneur arrêté, credentials invalides, firewall bloquant le port 5432, schéma inexistant, base en maintenance.

Procédure de diagnostic :

- Vérifier l'état du serveur PostgreSQL et sa connectivité réseau depuis Airflow.
- Valider les credentials dans Airflow et tester la connexion via client PostgreSQL.
- Vérifier le réseau : DNS, TCP port 5432, règles de pare-feu et sécurité cloud.
- Contrôler le schéma et les permissions sur les tables nécessaires.

Actions correctives :

- Redémarrer PostgreSQL si arrêté.
- Corriger ou recréer les credentials dans Airflow.
- Ajuster les règles réseau et firewall.
- Accorder les permissions nécessaires sur les tables.

Défaillance 2 : Échec d'upload vers AWS S3

- Manifestation : Extraction réussie mais fichiers CSV non uploadés.
- Causes possibles : Credentials AWS invalides, bucket inexistant ou renommé, permissions IAM insuffisantes, région incorrecte, quota atteint, problème réseau.

Procédure de diagnostic :

- Vérifier les credentials AWS et leur statut.
- Contrôler l'existence et la configuration du bucket S3.
- Analyser les permissions IAM et conditions associées.
- Tester la connectivité réseau vers S3 depuis Airflow.
- Vérifier quotas et limites de stockage.

Actions correctives :

- Régénérer et mettre à jour les credentials.
- Ajuster les permissions IAM ou corriger la politique.
- Créer ou renommer le bucket selon la configuration.
- Ajuster le réseau et le proxy pour autoriser le trafic HTTPS.

2.2.2 ETL – Transformation et Chargement

Criticité : Haute

Défaillance 3 : Fichiers CSV S3 manquants ou corrompus

- Manifestation : Échec du DAG transform_load_data, fichiers introuvables ou illisibles.
- Causes possibles : Échec d'extraction, suppression manuelle, corruption, chemins S3 incorrects, région erronée, lifecycle S3 inadapté.

Actions correctives :

- Relancer l'extraction pour les fichiers manquants ou corrompus.
- Vérifier et corriger les chemins S3 et les règles lifecycle.

Défaillance 4 : Échec de join SQL – saturation mémoire

- Manifestation : Tâche join_tables échoue, OOM killer active.
- Causes possibles : Volume de données élevé, pandasql charge tout en mémoire, limitations du worker, DAGs concurrents.

Actions correctives :

- Filtrer les données avant jointure (batchs, 30 derniers jours).
- Augmenter mémoire du worker ou migrer vers instance plus puissante.
- Séquencer les DAGs, limiter workers parallèles.
- Optimiser le code (pandas.merge, traitement par batch, Dask).

Défaillance 5 : Taux de rejet anormal

- Manifestation : Peu de données arrivent dans Snowflake, collection MongoDB rejected_reviews saturée.
- Causes possibles : Dégradation des données sources, règles de validation trop strictes, champs obligatoires nuls, ratings hors plage, descriptions vides.

Actions correctives :

- Ajuster temporairement les règles de validation.
- Corriger les données sources ou normaliser les ratings.
- Supprimer les duplicatas et ajouter des contraintes d'unicité si nécessaire.

Défaillance 6 : Échec d'insertion dans Snowflake

- Manifestation : Tâche load_clean_to_snowflake échoue, timeout ou connexion refusée.
- Causes possibles : Warehouse suspendu, credentials expirés, quota dépassé, table inexistante, problème réseau.

Actions correctives :

- Réactiver le warehouse et configurer AUTO_RESUME.

- Restaurer la table via Time Travel si nécessaire.
- Optimiser requêtes SQL et ajuster timeout.

2.2.3 Application Streamlit & FastAPI

Composant : Interface utilisateur et backend API
Criticité : Critique

Défaillance 7 : API inaccessible

- Manifestation : Page blanche, erreur de connexion au backend.
- Causes possibles : FastAPI arrêté, port utilisé, firewall, variable API_URL incorrecte, backend sans connexion à Snowflake.

Actions correctives :

- Redémarrer FastAPI, vérifier logs et port.
- Ajuster API_URL et configuration réseau Docker.
- Ouvrir le port dans le firewall.

Défaillance 8 : Erreurs Snowflake depuis l'API

- Manifestation : Endpoints retournent 500, logs montrent timeout ou erreurs Snowflake.
- Causes possibles : Credentials incorrects, warehouse suspendu, requêtes lentes, table vide, problème réseau.

Actions correctives :

- Mettre à jour les credentials, redémarrer backend.
- Optimiser requêtes SQL et mettre en cache côté backend.
- Réactiver warehouse et ajuster timeout.
- Vérifier pipeline ETL pour s'assurer que les données sont chargées.

2.3 Procédure générale de rollback

- Identifier le dernier commit stable dans Git.
- Créer une branche rollback et restaurer le code.
- Redéployer conteneurs Docker : dépendances → Airflow → backend → Streamlit.
- Restaurer données Snowflake via Time Travel si nécessaire.
- Nettoyer MongoDB des documents incorrects.
- Valider le rollback avec tests automatisés et exécution manuelle des DAGs.

2.4 Checklist de clôture d'intervention

- Symptôme initial résolu.
- Tests automatisés de qualité de données passés.
- Pas de nouvelles alertes email.
- Logs vérifiés et archivés.
- Actions documentées et communiquées aux équipes concernées.

3. Maintenance Préventive

3.1 Vue d'ensemble

La maintenance préventive vise à anticiper et prévenir les défaillances avant qu'elles n'impactent le système en production. Elle s'organise autour de vérifications régulières, d'optimisations proactives et de mises à jour planifiées.

3.2 Calendrier de maintenance

Fréquence	**Tâches**	**Responsable**	**Durée estimée**
Quotidien	Vérification des DAGs Airflow, consultation alertes email, monitoring logs MongoDB	Data Engineer	15 min
Hebdomadaire	Tests qualité données, analyse taux de rejet, vérification espace disque S3/Snowflake	Data Engineer	1 heure
Mensuel	Revue performance requêtes, optimisation warehouse Snowflake, nettoyage logs	Data Engineer	2 heures
Trimestriel	Mise à jour dépendances Python, audit sécurité, revue architecture	DevOps + Data Engineer	4 heures
Annuel	Migration versions majeures, refactoring, audit complet infrastructure	Équipe complète	2 jours

3.3 Tâches de maintenance quotidienne

3.3.1 Vérification des exécutions Airflow

Pour détecter rapidement les échecs de pipeline avant qu'ils n'entraînent un retard, il est recommandé de suivre une procédure structurée. Tout d'abord, il convient d'accéder au tableau de bord Airflow afin de vérifier le statut des DAGs pour les dernières 24 heures, en s'assurant en particulier que les DAGs extract_to_s3 et transform_load_data se sont exécutés avec succès. Il est également important de consulter la durée d'exécution de chaque DAG et de la comparer avec les moyennes historiques afin de détecter tout ralentissement ou anomalie, même si le DAG a été marqué comme réussi. Les seuils d'alerte incluent un temps d'exécution supérieur à 150 % de la moyenne habituelle, plus de deux échecs consécutifs sur un même DAG, ou l'absence d'exécution d'un DAG quotidien au cours des dernières 24 heures. Toute anomalie détectée doit être notée pour permettre un suivi et une résolution rapide.

3.3.2 Consultation des alertes email

Pour la consultation des alertes email, l'objectif est de traiter rapidement les notifications automatiques du système. Il convient de vérifier la boîte email configurée pour recevoir les alertes Airflow, de prioriser les notifications selon leur sévérité (échec total versus partiel) et de créer un ticket d'incident pour chaque alerte nécessitant une investigation. Les fausses alertes doivent être identifiées et les seuils ajustés si nécessaire afin d'éviter les notifications superflues.

3.3.3 Monitoring des logs MongoDB

Concernant le monitoring des logs MongoDB, l'objectif est d'identifier les patterns d'erreurs récurrents avant qu'ils ne deviennent critiques. Il faut consulter les collections de logs des dernières 24 heures, compter les messages de niveau ERROR et WARNING, et repérer les erreurs qui se répètent fréquemment. Il est également essentiel de vérifier le taux de rejet dans la collection rejected_reviews. Les seuils d'alerte définis incluent plus de 10 messages ERROR en 24 heures, un taux de rejet supérieur à 5 % du volume total traité, ainsi qu'une augmentation soudaine de plus de 50 % des rejets par rapport à la veille.

3.4 Tâches de maintenance Mensuelle

Les tâches de maintenance mensuelle visent à assurer la performance, l'espace de stockage et la sécurité du système.

Pour l'optimisation des performances Snowflake, l'objectif est de maintenir des temps de réponse acceptables tout en contrôlant les coûts. Il est nécessaire de consulter l'historique des requêtes du mois écoulé, d'identifier les 10 requêtes les plus lentes ou les plus fréquentes, d'analyser leurs plans d'exécution pour repérer les goulots d'étranglement et d'évaluer l'opportunité de créer des tables matérialisées ou des indexes. Les axes d'optimisation comprennent l'ajustement de la taille du warehouse en fonction de l'utilisation réelle, la réduction du nombre de colonnes sélectionnées dans les requêtes API, la mise en place d'un système de cache applicatif pour les données peu changeantes, ainsi que le partitionnement de la table reviews par date si son volume dépasse 10 millions de lignes.

Le nettoyage des logs et des données temporaires a pour objectif de libérer de l'espace et d'améliorer les performances des requêtes sur les logs. Chaque mois, il convient de supprimer dans MongoDB les logs Airflow de plus de 3 mois, d'archiver les rejected_reviews de plus de 6 mois vers un stockage froid, de tronquer ou d'archiver dans Snowflake les anciennes versions de tables si le Time Travel n'est plus nécessaire, et d'appliquer dans S3 une politique de lifecycle pour déplacer les fichiers de plus de 30 jours vers Glacier.

Enfin, la revue des credentials et accès a pour but de maintenir la sécurité et d'éviter les interruptions liées à l'expiration des credentials. Il est nécessaire de lister tous les credentials utilisés (PostgreSQL, AWS, Snowflake, MongoDB), de vérifier les dates d'expiration des Access Keys AWS et de les renouveler si moins de 30 jours restent, de contrôler que les mots de passe respectent la politique de rotation (90 jours maximum), d'auditer les permissions des comptes de service pour supprimer les accès inutiles, et de tester chaque connexion Airflow afin de confirmer leur validité.

3.5 Tâches de maintenance trimestrielle

L'objectif de cette procédure est de corriger les vulnérabilités de sécurité et de bénéficier des améliorations apportées par les nouvelles versions des bibliothèques Python utilisées. La démarche consiste à lister toutes les bibliothèques avec leurs versions actuelles, puis à consulter les changelogs pour identifier celles incluant des correctifs de sécurité. Les mises à jour doivent d'abord être testées dans un environnement de staging avant d'être déployées en production. Une fois validées, les fichiers requirements.txt sont mis à jour et les images Docker sont reconstruites avec les nouvelles dépendances.

Les dépendances critiques à surveiller incluent notamment pandas et pandasql pour le traitement des données, snowflake-connector-python pour la connexion à Snowflake, fastapi et uvicorn pour le backend API, streamlit pour l'interface utilisateur, ainsi que airflow et ses providers.

3.6 Audit de sécurité

L'objectif de cette procédure est d'identifier et de corriger les failles de sécurité potentielles. Les contrôles à effectuer incluent la vérification que les fichiers .env ne sont pas versionnés dans Git, le contrôle que les credentials ne figurent pas en clair dans les logs, et la confirmation que le hachage des buyer_id utilise un sel fort et constant. Il est également essentiel de valider que les buckets S3 ne disposent pas d'accès public non intentionnel, de contrôler les règles de pare-feu et les groupes de sécurité dans le cloud, et de s'assurer que SSL/TLS est activé sur toutes les connexions, notamment vers Snowflake et les API.

3.7 Tâches de maintenance Annuelle

Les tâches de maintenance annuelle incluent notamment la migration des versions majeures des différents composants pour rester à jour et bénéficier du support éditeur. Les systèmes concernés sont Airflow, pour lequel il faut planifier la migration vers la dernière version stable ; Python, en évaluant le passage à la version suivante (par exemple de 3.11 à 3.12) ; PostgreSQL, en vérifiant la possibilité de migration vers une version plus récente ; et FastAPI ainsi que Streamlit, en suivant leurs versions LTS (Long Term Support).

L'approche recommandée consiste à créer un environnement de test complet, à migrer d'abord les environnements non-production, puis à exécuter tous les tests automatisés après la migration. Il est important de valider les changements avec les équipes métier avant le déploiement en production et de prévoir un plan de rollback pour pouvoir revenir en arrière en cas de problème.

3.8 Indicateurs de santé du système

Indicateur	Cible	Alerte si
Taux de succès DAGs	> 98%	< 95%
Temps moyen d'exécution extract_to_s3	< 15 min	> 25 min
Temps moyen d'exécution transform_load_data	< 30 min	> 45 min
Taux de rejet MongoDB	< 5%	> 10%
Nombre de lignes chargées Snowflake/jour	Stable ±20%	Variation > 50%
Temps de réponse API (p95)	< 2s	> 5s
Temps de chargement Streamlit	< 3s	> 10s
Coût mensuel Snowflake	Budget défini	Dépassement > 20%
Espace disque S3 utilisé	< 80%	> 85%
Nombre d'incidents P0/P1	0	> 2

4. Grille de Suivi des Risques (Risk Tracking Grid)

Le **Risk Tracking GRID** est un outil de suivi des risques permettant de monitorer de manière structurée les indicateurs critiques d'un système, d'identifier rapidement les écarts et de déclencher des actions correctives. Dans le cadre du suivi mensuel, plusieurs métriques clés doivent être surveillées.

La méthodologie d'évaluation des risques repose sur deux axes principaux : la probabilité d'occurrence et l'impact potentiel. L'échelle de probabilité définit la fréquence estimée d'un événement sur une base annuelle : très faible (1) pour moins de 5 % de chance, faible (2) pour 5 à 25 %, moyenne (3) pour 25 à 50 %, élevée (4) pour 50 à 75 % et très élevée (5) pour plus de 75 %. L'échelle d'impact mesure les conséquences sur le service et les utilisateurs : négligeable (1) si aucun impact utilisateur et résolution en moins d'une heure, faible (2) pour un impact limité et résolution en moins de 4 heures, moyen (3) pour un service partiellement dégradé et résolution en moins d'un jour, élevé (4) si le service est indisponible pour certains utilisateurs avec résolution en moins de 3 jours, et critique (5) pour une indisponibilité complète du service ou une perte de données.

La criticité d'un risque est calculée en multipliant la probabilité par l'impact. Selon le résultat, les actions diffèrent : une criticité faible (1-4) nécessite une surveillance passive, une criticité moyenne (5-9) implique une surveillance active avec plan d'action préparé, une criticité élevée (10-15) requiert une action préventive, et une criticité critique (16-25) impose une action immédiate.

4.1 Grille des risques techniques

ID	Risque	Prob.	Impact	Crit.	Mesures préventives	Mesures correctives	Responsable	Statut
R-T01	Saturation mémoire lors du join SQL	4	4	16	Monitoring utilisation RAM, limite volumétrie par batch, alerte si RAM > 80%	Augmentation RAM worker, traitement par chunks, optimisation requête pandasql	Data Engineer	Actif
R-T02	Épuisement crédits Snowflake	3	4	12	Alerte à 70% consommation, budget mensuel défini, revue hebdo consommation	Suspension warehouse temporaire, optimisation requêtes, réduction taille warehouse	DBA Snowflake	Actif
R-T03	Corruption fichiers CSV sur S3	2	4	8	Validation intégrité post-upload, versioning S3 activé, checksum MD5	Restauration depuis version S3, re-extraction depuis PostgreSQL, validation manuelle	DevOps	Actif
R-T04	Expiration credentials AWS/Snowflake	3	5	15	Calendrier rotation 90 jours, alerte 30j avant expiration, test hebdo connexions	Régénération credentials urgente, mise à jour Airflow immédiate, documentation incident	DevOps	Actif
R-T05	Perte anonymisation buyer_id	1	5	5	Salt stocké en gestionnaire de secrets, backup salt chiffré, test cohérence hachage	Restauration salt depuis backup, re-traitement données si nécessaire, audit impact	Data Engineer	Actif
R-T06	Dérive qualité données source	4	3	12	Tests qualité automatisés quotidiens, SLA avec équipe source, alertes seuils rejet	Contact équipe source, assouplissement temporaire règles, analyse cause racine	Data Engineer	Actif
R-T07	Warehouse Snowflake suspendu	4	3	12	Configuration AUTO_RESUME, monitoring statut warehouse,	Redémarrage manuel warehouse, vérification budget,	DBA Snowflake	Actif

					timeout connexion adapté	ajustement AUTO_SUSPEND		
R-T08	Saturation espace S3	2	3	6	Monitoring utilisation quotidien, politique lifecycle (archivage 90j), alerte 80%	Suppression anciennes extractions, migration vers Glacier, augmentation quota	DevOps	Actif
R-T09	Échec connexion PostgreSQL source	3	4	12	Monitoring santé PostgreSQL, connection pool configuré, retry automatique 3 fois	Redémarrage service PostgreSQL, vérification firewall/réseau, validation credentials	DBA PostgreSQL	Actif
R-T10	Timeout requêtes Snowflake depuis API	3	3	9	Optimisation requêtes, cache applicatif 5 min, indexation table reviews	Augmentation timeout, redimensionnement warehouse, refactoring requêtes	Backend Dev	Actif

4.2 Grille des risques opérationnels

ID	Risque	Prob.	Impact	Crit.	Mesures préventives	Mesures correctives	Responsable	Statut
R-O01	Indisponibilité membre clé équipe	3	4	12	Documentation à jour, formation croisée équipe, procédures opérationnelles écrites	Réaffectation tâches, support externe temporaire, priorisation incidents critiques	Manager	Actif
R-O02	Dépassement fenêtre maintenance	3	3	9	Planning maintenance défini, buffer temps 25%, tests en staging préalables	Communication utilisateurs, rollback si nécessaire, finalisation hors fenêtre	DevOps	Actif

R-O03	Non-respect SLA avec équipes métier	2	3	6	SLA documentés et signés, monitoring temps réponse, revue mensuelle performance	Communication proactive, plan de rattrapage, revue SLA si irréaliste	Product Owner	Actif
R-O04	Perte historique logs MongoDB	2	2	4	Backup logs mensuels, politique rétention 6 mois, réplication MongoDB	Restauration depuis backup, reconstruction partielle possible, documentation incident	DevOps	Actif
R-O05	Erreur déploiement en production	2	4	8	Tests automatisés obligatoires, staging identique prod, checklist déploiement	Rollback immédiat, analyse cause racine, amélioration process CI/CD	DevOps	Actif

4.3 Grille des risques sécurité

ID	Risque	Prob.	Impact	Crit.	Mesures préventives	Mesures correctives	Responsable	Statut
R-S01	Exposition credentials dans logs	2	5	10	Masquage automatique des credentials, revue code sécurité, scan secrets dans Git	Rotation des credentials exposés, nettoyage des logs compromis, audit des accès	Security Officer	Actif
R-S02	Accès non autorisé bucket S3	2	5	10	Bucket policies restrictives, blocage accès public, audit IAM trimestriel	Revue accès urgente, révocation permissions, investigation accès	DevOps	Actif
R-S03	Vulnérabilité dépendances Python	3	3	9	Scan vulnérabilités hebdomadaire, mise à jour trimestrielle, abonnement alertes CVE	Patch urgence si critique, test régression, déploiement prioritaire	DevOps	Actif
R-S04	Fuite données buyer_id non anonymisés	1	5	5	Validation anonymisation tests, audit trails activés, formation équipe RGPD	Notification CNIL si applicable, investigation périmètre, suppression données exposées	DPO	Actif
R-S05	Injection SQL dans API	1	4	4	Requêtes paramétrées uniquement, validation inputs stricte, WAF si applicable	Patching code urgent, audit logs accès, notification SI sécurité	Backend Dev	Actif

4.4 Grille des risques métier

ID	Risque	Prob.	Impact	Crit.	Mesures préventives	Mesures correctives	Responsable	Statut
R-M01	Dégradation qualité recommendations	3	4	12	Tests A/B réguliers, feedback utilisateurs, KPI pertinence suivis	Ajustement algorithme scoring, revue règles métier, formation modèle ML	Data Scientist	Actif
R-M02	Augmentation volumétrie 300%	2	4	8	Projection croissance mensuelle, architecture scalable, load testing trimestriel	Scale horizontal infrastructure, optimisation pipeline, révision architecture	Architect	Actif
R-M03	Changement structure données source	2	4	8	Contrat interface avec source, tests schéma automatisés, communication proactive	Adaptation pipeline urgent, mapping ancien/nouveau format, tests non-régression	Data Engineer	Actif
R-M04	Coûts cloud > budget 150%	2	3	6	Budget mensuel défini, alertes à 80% et 90%, optimisation continue	Analyse postes coûteux, réduction taille resources, révision budget ou périmètre	Manager	Actif

4.4 Plan d'action prioritaire (Criticité ≥ 12)

Les **risques critiques nécessitant une action immédiate** sont priorisés afin d'éviter des impacts majeurs sur le système et les opérations. Les principaux risques identifiés incluent :

- **R-T01 – Saturation mémoire (Criticité : 16)**
  - Action : Implémenter un traitement par chunks avant la fin du premier trimestre.
  - Budget : Potentiel upgrade serveur estimé à 2000 €.
  - Deadline : 31/03/2026.
- **R-T04 – Expiration des credentials (Criticité : 15)**
  - Action : Automatiser la rotation des credentials avec Vault ou Secrets Manager.
  - Budget : Licence gestionnaire de secrets 500 €/an.
  - Deadline : 28/02/2026.
- **R-T02 – Épuisement des crédits Snowflake (Criticité : 12)**
  - Action : Mettre en place un système d'alertes automatiques et optimiser les requêtes.
  - Budget : Temps de développement estimé à 5 jours.
  - Deadline : 31/01/2026.
- **R-T06 – Dérive de la qualité des données (Criticité : 12)**
  - Action : Négocier un SLA avec l'équipe source et mettre en place un monitoring proactif.
  - Budget : Temps de développement estimé à 3 jours.
  - Deadline : 15/02/2026.
- **R-O01 – Indisponibilité d'un membre clé (Criticité : 12)**
  - Action : Lancer un programme de formation croisée et documenter complètement les procédures.
  - Budget : Temps de formation estimé à 10 jours.
  - Deadline : 30/04/2026.
  -
- **R-M01 – Dégradation de la qualité des recommandations (Criticité : 12)**
  - Action : Mettre en place un framework de tests A/B et un dashboard KPI en temps réel.
  - Budget : Temps de développement estimé à 8 jours.
  - Deadline : 31/03/2026.

La **revue et la mise à jour de la grille des risques** se font de manière régulière : mensuelle pour les risques critiques (criticité ≥ 12) et trimestrielle pour les autres. Le processus comprend :

- Revue systématique après chaque incident P0/P1.
- Réévaluation de la probabilité basée sur l'historique réel.
- Ajout des nouveaux risques identifiés lors des audits.
- Clôture des risques mitigés ou devenus obsolètes.
- Validation par le comité technique trimestriel.

Les **indicateurs de suivi** pour garantir l'efficacité du risk tracking incluent :

- Nombre de risques critiques (cible : ≤ 3).
- Taux de réalisation des risques prévus versus risques réels (cible : < 20 %).
- Temps moyen de résolution des risques matérialisés (cible : < 24 h).
- Budget consommé pour la mitigation des risques.

5. Exigences d'Evolution

Le système actuel répond aux besoins fonctionnels initiaux mais présente des limitations techniques et métier identifiées lors de l'analyse de production. Les exigences d'évolution ci-dessous visent à améliorer la performance, la scalabilité, la maintenabilité et la valeur métier du système.

5.1 Exigences fonctionnelles

L'enrichissement des critères de pertinence des reviews constitue une priorité élevée avec un effort estimé à huit jours et un ROI important. Actuellement, l'algorithme de scoring se base uniquement sur le rating, la longueur du texte et la présence d'une image. Les utilisateurs souhaitent des reviews plus contextualisées. Pour répondre à ce besoin, il est nécessaire d'intégrer le profil de l'acheteur, en tenant compte du nombre total d'achats et de l'ancienneté du compte, de pondérer les reviews selon leur fraîcheur afin que les plus récentes soient mises en avant, et de valoriser les reviews issues d'achats vérifiés. De plus, un système de vote utilisateur doit être implémenté pour renforcer l'interaction, et les pondérations doivent pouvoir s'adapter dynamiquement au comportement des utilisateurs. L'acceptation de cette fonctionnalité sera mesurée par une augmentation du taux de clics sur les reviews affichées de 20 %, une réduction de 15 % du bounce rate sur l'application Streamlit et un feedback utilisateur positif supérieur à 80 %.

Le tableau de bord de monitoring temps réel représente également une priorité élevée avec un effort de dix jours et un ROI significatif. Les équipes consultent actuellement les logs MongoDB et Airflow manuellement, ce qui limite leur réactivité. La mise en place d'un dashboard centralisé permettra de visualiser les métriques clés telles que le taux de succès des DAGs, la volumétrie traitée, le taux de rejet et la performance de Snowflake. Des alertes visuelles signaleront tout dépassement des seuils critiques, et un historique de trente jours permettra d'analyser les tendances. Une section dédiée aux coûts cloud, incluant Snowflake et AWS S3, sera intégrée, et l'accès sera limité en lecture seule pour les équipes métier.

L'API RESTful publique pour les partenaires constitue une priorité moyenne avec un effort de quinze jours et un ROI modéré. Les partenaires externes souhaitent accéder aux reviews pertinentes pour les intégrer dans leurs systèmes. L'API devra assurer une authentification par clé avec limitation du nombre de requêtes, fournir des endpoints permettant la récupération des reviews par produit, par catégorie et par période, et générer automatiquement une documentation OpenAPI/Swagger. Le versioning sera pris en compte (v1, v2) et des logs d'audit des accès seront conservés pour garantir la conformité.

5.1 Exigences techniques

Le remplacement de pandasql par un traitement optimisé représente une priorité critique avec un effort de douze jours et un ROI très élevé. Le module actuel charge l'intégralité des DataFrames en mémoire, entraînant des saturations régulières (risque R-T01, criticité 16). Il est nécessaire de substituer pandasql par des opérations pandas natives plus efficaces, d'implémenter un traitement par chunks de 50 000 lignes maximum, de réduire l'empreinte mémoire d'au moins 60 %, tout en maintenant la compatibilité avec les tests existants et en documentant la nouvelle approche dans le code.

L'implémentation d'un cache Redis pour l'API est une priorité élevée avec un effort de cinq jours et un ROI significatif. Chaque requête Streamlit interroge actuellement Snowflake, générant coûts et latence. Le cache Redis permettra de stocker les données peu volatiles avec un TTL de dix minutes, d'invalider intelligemment les données lors de nouvelles insertions et de préchauffer automatiquement les produits populaires. Cette solution vise à réduire de 80 % les requêtes répétitives vers Snowflake tout en offrant un fallback gracieux en cas d'indisponibilité du cache.

La migration vers un orchestrateur moderne, comme Prefect ou Dagster, est une priorité basse sur un horizon de vingt jours avec un ROI moyen à long terme. Airflow est aujourd'hui complexe à maintenir et présente des limitations pour le versioning des DAGs. L'approche recommandée consiste à évaluer les solutions disponibles, réaliser un proof of concept sur un environnement de test, migrer progressivement DAG par DAG, former l'équipe et documenter l'ensemble du processus.

Enfin, la mise en place d'un système de notifications multi-canal est une priorité moyenne avec un effort de six jours. Les alertes actuelles, uniquement par email, sont facilement manquées hors des heures de bureau. Le nouveau système intégrera Slack pour les alertes en temps réel, le SMS pour les incidents critiques et PagerDuty pour la gestion des astreintes, avec escalade automatique si aucune prise en charge n'est réalisée dans les trente minutes et une configuration flexible des canaux selon le type d'alerte.

5.2 Exigences de performance

L'objectif principal de réduction du temps d'exécution du pipeline vise à descendre sous la barre des 30 minutes alors qu'actuellement il varie entre 45 et 60 minutes. Pour y parvenir, il est prévu de paralléliser l'extraction des tables PostgreSQL, d'optimiser les requêtes JOIN avec indexation appropriée, d'augmenter la taille du warehouse Snowflake uniquement pendant les chargements, et de compresser les fichiers CSV sur S3.

Pour améliorer le temps de réponse de l'application, la cible est un P95 inférieur à 1 seconde, contre 2 à 3 secondes actuellement. La stratégie inclut l'implémentation du cache Redis (ET-02), la pagination côté backend avec un affichage de 20 reviews par défaut, le lazy loading des images de reviews, et la minification ainsi que le bundling des assets frontend.

5.3 Exigences de sécurité et conformité

L'audit trail complet des accès aux données est une priorité haute avec un effort de quatre jours pour assurer la conformité RGPD. Il devra permettre de tracer qui a accédé à quoi, quand et depuis où, avec une rétention des logs d'au moins deux ans et une détection des anomalies d'accès. Un reporting mensuel sera transmis au DPO.

Le chiffrement des données au repos et en transit est également prioritaire, avec un effort de trois jours pour répondre à la norme ISO 27001. Les exigences incluent le chiffrement S3 avec clés KMS, l'obligation SSL/TLS sur toutes les connexions, la rotation automatique des certificats et le chiffrement des collections MongoDB sensibles.

5.4 Exigences d'exploitabilité

L'automatisation complète des déploiements est une priorité moyenne avec un effort de huit jours et un ROI moyen. Le pipeline CI/CD doit couvrir tests unitaires, tests d'intégration, déploiement en staging,

validation et production, avec rollback automatique en cas d'échec du healthcheck, blue/green deployment pour un downtime nul et notifications Slack à chaque étape.

Le self-service pour les équipes métier est une priorité basse avec un effort de dix jours. Il permettra de déclencher manuellement les DAGs sans accès à Airflow, de visualiser le statut du pipeline en temps réel, d'exporter les reviews au format CSV pour des analyses ad-hoc, et de configurer les seuils d'alertes métier sans intervention technique.

5.5 Roadmap priorisée

Pour le premier trimestre 2026, les actions prioritaires incluent le remplacement de pandasql (ET-01), l'enrichissement du scoring de pertinence (EF-01), l'implémentation du cache Redis (ET-02) et le chiffrement complet (ES-02). Le second trimestre sera consacré au dashboard de monitoring Grafana (EF-02), aux notifications multi-canal (ET-04), à l'audit trail (ES-01) et aux optimisations de performance du pipeline (EP-01). Le troisième trimestre portera sur l'API publique partenaires (EF-03), l'automatisation CI/CD (EO-01) et l'optimisation de l'application (EP-02). Enfin, le quatrième trimestre prévoit l'évaluation de la migration vers un nouvel orchestrateur (ET-03) et le self-service pour les équipes métier (EO-02).

5.6 Budget prévisionnel global

Catégorie	Effort (jours)	Coût développement	Coût infrastructure	Total
Évolutions fonctionnelles	33	26 400 €	2 000 €	28 400 €
Évolutions techniques	43	34 400 €	5 000 €	39 400 €
Sécurité & conformité	7	5 600 €	1 500 €	7 100 €
Exploitabilité	18	14 400 €	1 000 €	15 400 €
Total	101 jours	80 800 €	9 500 €	90 300 €

L'hypothèse retenue est un taux journalier moyen de 800 €.

6. Maintenance Evolutive

La maintenance évolutive vise à faire évoluer le système au-delà de la simple correction de bugs ou de la prévention des pannes. Elle s'inscrit dans une démarche d'amélioration continue alignée sur les objectifs métier et les contraintes techniques identifiées lors de l'analyse du système en production.

6.2 Propositions d'évolution de l'architecture actuelle

Première proposition :

Actuellement le pipeline ETL fonctionne en mode batch quotidien avec des DAGs Airflow séquentiels. Les données sont disponibles avec un délai de 24h minimum, limitant les cas d'usage temps réel. Il serai pertinent d'implémenter une architecture event-driven avec Apache Kfka ou AWS Kinesis. Cette dernière permettrai le déclenchement automatique du pipeline lors de l'ajout de nouvelles reviews dans PostgreSQL et le traitement en micro-batches toutes les 15 minutes au lieu d'une fois par jour ainsi que la mise à jour quasi temps réel de l'application Streamlit.

Deuxième proposition :

Actuellement les fichiers CSV bruts sont stockés sur S3 sans structure, indexation ou partitionnement. Les requêtes exploratoires nécessitent de télécharger et parser l'intégralité des fichiers.il serai pertinent d'adopter un format colonnaire optimisé (Parquet) au lieu du CSV et partitionner par date et catégorie. Ces proposition permettront de réduire considérablement la taille du stockage s3 et l'amélioration des performance de lecture.

Troisième proposition :

Actuellement PostgreSQL est sollicité directement par le pipeline ETL, créant une charge supplémentaire sur la base transactionnelle. Il serai pertinent de mettre en place un replica PostgreSQL read-only dédié à l'ETL ce qui aura zéro impact sur les performances de l'application source et la possibilité d'exécuter le pipeline à tout moment sans contraintes horaire

6.3 Matrice de priorisation des propositions

Proposition	Valeur métier	Complexité	Effort	ROI	Priorité	Horizon
Event-driven (Kafka/Kinesis)	Très élevée	Très élevée	30j	Moyen	1	Q4 2026
Data Lake Parquet	Élevée	Moyenne	12j	Élevé	2	Q2 2026
Réplication PostgreSQL	Moyenne	Moyenne	8j	Moyen	3	Q2 2026

6.4 Plan de déploiement recommandé

Phase 1 – Fondations et performance (Q1-Q2 2026, 20 jours, estimation 30 000€)

La première étape consiste à restructurer le stockage et optimiser l'accès aux données. La migration des fichiers CSV vers un Data Lake en format Parquet partitionné (Proposition 2) permettra de réduire la taille des fichiers S3 et d'accélérer considérablement les requêtes exploratoires. Parallèlement, la mise en place d'un réplica PostgreSQL en lecture seule dédié à l'ETL (Proposition 3) protégera la base transactionnelle et permettra de lancer le pipeline à tout moment sans contraintes sur l'application source.

Phase 2 – Temps réel et automatisation (Q3 2026, 25 jours, estimation 40 000€)

La deuxième phase porte sur l'implémentation d'une architecture event-driven (Proposition 1). Le pipeline sera déclenché automatiquement dès l'ajout de nouvelles reviews dans PostgreSQL et traitera les données en micro-batches toutes les 15 minutes. Cette approche permet une mise à jour quasi temps réel de l'application Streamlit et ouvre la voie à des cas d'usage temps réel jusqu'alors impossibles avec le batch quotidien.

Le budget total pour le déploiement des trois initiatives est estimé à 70 000€ sur 9 mois, avec un gain immédiat sur la performance et la disponibilité des données.

6.5 Critères de succès et suivi

Le succès des trois initiatives sera évalué selon quatre axes :

- **Performance** : Réduction du temps d'accès aux données et diminution de la charge sur PostgreSQL.
- **Qualité** : Taux d'erreurs ETL réduit grâce au Data Lake Parquet et aux réplications PostgreSQL.
- **Métier** : Amélioration de la disponibilité des données pour l'application Streamlit, permettant des décisions plus rapides et plus fiables.
- **Technique** : Adoption d'une architecture event-driven avec micro-batches, réduction de la dette technique et meilleure scalabilité du pipeline.

La gouvernance comprendra un suivi mensuel par comité de pilotage, des revues techniques trimestrielles, ainsi qu'un retour utilisateur continu via des tests et enquêtes pour mesurer l'impact sur la satisfaction et l'efficacité des traitements.

Conclusion

Le présent document a présenté une vision structurée pour l'amélioration et l'évolution du pipeline de données, en combinant performance, fiabilité et valeur métier. Les propositions retenues — migration vers un Data Lake Parquet, réplication PostgreSQL pour l'ETL et mise en place d'une architecture event-driven — constituent des leviers essentiels pour moderniser l'infrastructure existante et répondre aux besoins croissants de disponibilité et de réactivité des données.

Le plan de déploiement recommandé, réparti en phases progressives, permet d'équilibrer gains rapides et investissements plus ambitieux, tout en assurant un suivi rigoureux des KPI techniques, métier et financiers. La gouvernance proposée garantit que chaque étape est validée, mesurée et ajustée selon les retours utilisateurs et les objectifs de performance.

En combinant ces initiatives, l'organisation pourra non seulement optimiser l'exploitation des données existantes mais également ouvrir la voie à de nouveaux cas d'usage temps réel, tout en maîtrisant les coûts et en réduisant la dette technique. Ce projet représente donc un investissement stratégique, aligné sur la vision métier et technique de l'entreprise, et prépare le système à évoluer durablement face aux besoins futurs.