03/04/2026

Mémoire Projet Data - Développement Architecture de données

Mastere 1 -DATA ENGINEERING

EFREI BORDEAUX

Bill Padonou, Hatim Adnane
KIVENDTOUT

Table des matières

Chapitre 1 : Introduction & Contexte

1.1 Contexte et enjeux

L'économie numérique impose aux entreprises de e-Commerce une transformation profonde de leur rapport à la donnée. En 2024, selon Gartner, plus de 87 % des entreprises considèrent l'analytique de données comme une priorité stratégique de premier rang. Les plateformes data deviennent un levier direct de performance et de différenciation. Dans ce contexte, les architectures traditionnelles atteignent leurs limites. Les entrepôts monolithiques et les traitements batch nocturnes ne répondent plus aux contraintes de volumétrie, de diversité des sources et de fraîcheur des données.

Dans ce cadre, KiVendTout, PME e-commerce française fondée en 2022, évolue dans un environnement data de plus en plus complexe. L'entreprise commercialise des produits multi catégories, incluant des biens réglementés nécessitant des contrôles d'identité et d'âge. Son système d'information produit des flux variés issus de Firebase pour le suivi du comportement utilisateur, du CRM pour la gestion client, des systèmes de paiement en temps réel, des partenaires logistiques, des journaux Nginx pour l'analytique web et des dossiers KYC pour la conformité. Entre 2024 et 2025, la croissance rapide de l'activité a entraîné un triplement du chiffre d'affaires. Cette dynamique s'est traduite par une augmentation massive des volumes de données, une complexification des flux et un renforcement des exigences réglementaires. Cette évolution a rapidement mis en tension l'architecture existante. En novembre 2025, une requête analytique exécutée sur la base PostgreSQL transactionnelle a saturé l'infrastructure. Le site est resté indisponible pendant plusieurs heures, avec un impact direct sur le chiffre d'affaires. Cet incident a mis en évidence l'absence de séparation entre les usages transactionnels et analytiques. Dans un autre cas, un pipeline alimentant les dashboards a échoué sans détection. Les équipes ont exploité des données obsolètes pendant plusieurs jours. L'absence de monitoring, d'alertes et de contrôles de qualité automatisés a amplifié l'impact.

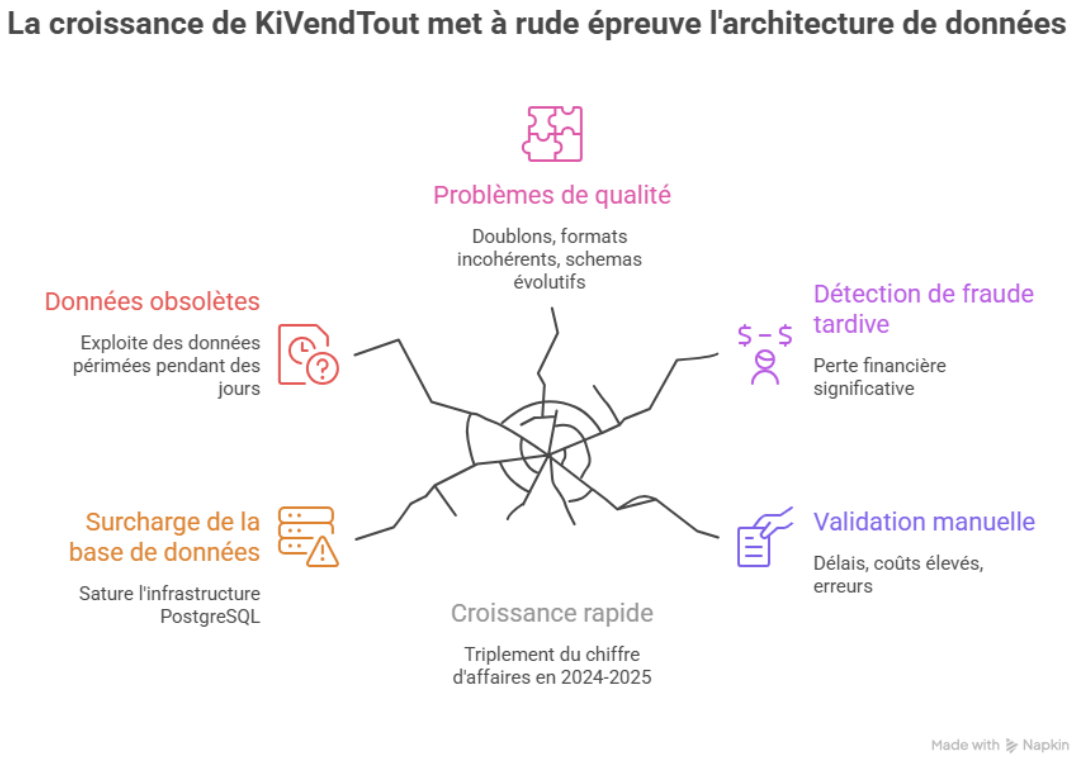
À cela s'ajoutent des problèmes récurrents de qualité des données. Les flux logistiques contiennent des doublons, les formats de dates varient, certains champs présentent des incohérences de types, et les schémas évoluent sans versionnement. Ces éléments rendent les traitements instables et limitent la fiabilité des analyses.

Dans le prolongement de ces contraintes, la gestion de la fraude impose un changement d'approche. Avant 2026, les données de paiement étaient traitées majoritairement en batch. La détection de fraude intervenait avec plusieurs heures de décalage, ce qui entraînait des pertes financières significatives. Depuis janvier 2026, une transition vers une architecture orientée streaming a été engagée pour les flux de paiement. Cette évolution permet de corréler en continu les paiements, les commandes et le comportement utilisateur, réduisant fortement les cas de fraude et améliorant la réactivité opérationnelle.

Parallèlement, une nouvelle contrainte réglementaire renforce les besoins en automatisation. Une législation entrée en vigueur en 2025 impose la vérification d'âge pour l'achat de produits sensibles. Le processus actuel repose sur une validation manuelle par le service client. Il

génère des délais pouvant atteindre 24 heures, un coût opérationnel élevé et un risque d'erreur non négligeable. L'entreprise doit mettre en place une solution automatisée capable d'analyser des pièces d'identité, d'extraire les informations nécessaires et de fournir une décision fiable en quelques secondes.

Face à ces constats, le rôle de l'ingénieur data devient central. Il doit assurer une ingestion fiable des données, structurer leur transformation en garantissant la traçabilité, mettre en place des contrôles de qualité à chaque étape et fournir des données exploitables dans des délais compatibles avec les usages opérationnels. Ce mémoire propose une réponse architecturale et technique à ces enjeux.



1.2 Problématique

La problématique centrale de ce mémoire est formulée ainsi :

"Comment concevoir et déployer une plateforme data entend pour une entreprise e-commerce, capable de centraliser et historiser l'ensemble des données métier et des événements utilisateurs, d'orchestrer des pipelines batch et streaming, de garantir l'intégrité, la qualité et la traçabilité des données, d'analyser les flux en temps réel pour la détection de fraude, et d'exposer des données fiables et accessibles aux équipes métier, tout en assurant la scalabilité, la continuité de service et la conformité réglementaire ?"

Cette problématique soulève plusieurs sous-questions techniques :

- Quelle architecture permet de centraliser l'ensemble des données brutes dans un espace unique tout en assurant leur historisation complète et leur intégrité ?

- Comment concevoir un système capable de traiter à la fois les données batch et les flux temps réel afin d'exploiter les événements utilisateurs et détecter les comportements frauduleux dès leur apparition ?
- Quels mécanismes permettent de garantir la qualité des données tout au long de leur cycle de vie ?
- Comment assurer la traçabilité des transformations dans un environnement intégrant plusieurs outils et plusieurs types de flux ?
- Quelle approche permet d'exposer les données de manière simple et standardisée aux équipes métier, data et partenaires ?
- Quels choix d'architecture garantissent la scalabilité du système face à la croissance du trafic et des volumes de données ?
- Comment assurer la continuité de service en cas de défaillance d'un composant du système ?
- Quels dispositifs permettent de garantir la sécurité et la conformité des données clients ?

1.3 Objectifs du projet

Ce mémoire poursuit quatre objectifs principaux.

Premièrement, un objectif de conception architecturale. Il s'agit de définir et de justifier une architecture data entend adaptée aux contraintes d'un e-commerce moderne. Cette architecture doit structurer les données en plusieurs niveaux de traitement, intégrer des flux batch et temps réel, et répondre aux exigences de performance, de fiabilité et d'évolutivité.

Deuxièmement, un objectif d'implémentation technique. L'ensemble des composants de la plateforme doit être développé avec un niveau de qualité industriel. Cela inclut l'orchestration des pipelines, les traitements de transformation, la gestion des flux temps réel, ainsi que la mise en place de mécanismes robustes de gestion des erreurs, de journalisation, de tests et d'idempotence.

Troisièmement, un objectif de gouvernance des données. Il consiste à intégrer dès la conception des mécanismes assurant la traçabilité des transformations et le contrôle de la qualité des données. Ces éléments doivent être pensés comme des composants centraux du système afin de garantir la fiabilité et la transparence des données exploitées.

Quatrièmement, un objectif de démonstration opérationnelle. Il vise à montrer que l'ensemble de la chaîne de traitement, depuis l'ingestion des données brutes jusqu'à leur exploitation dans des outils d'analyse, peut être déployé et exploité de manière cohérente dans un environnement unifié.

Chapitre 2 : Architecture Globale

2.1 Flux de données KiVendTout : état existant

2.1.1 Description de la stack en place

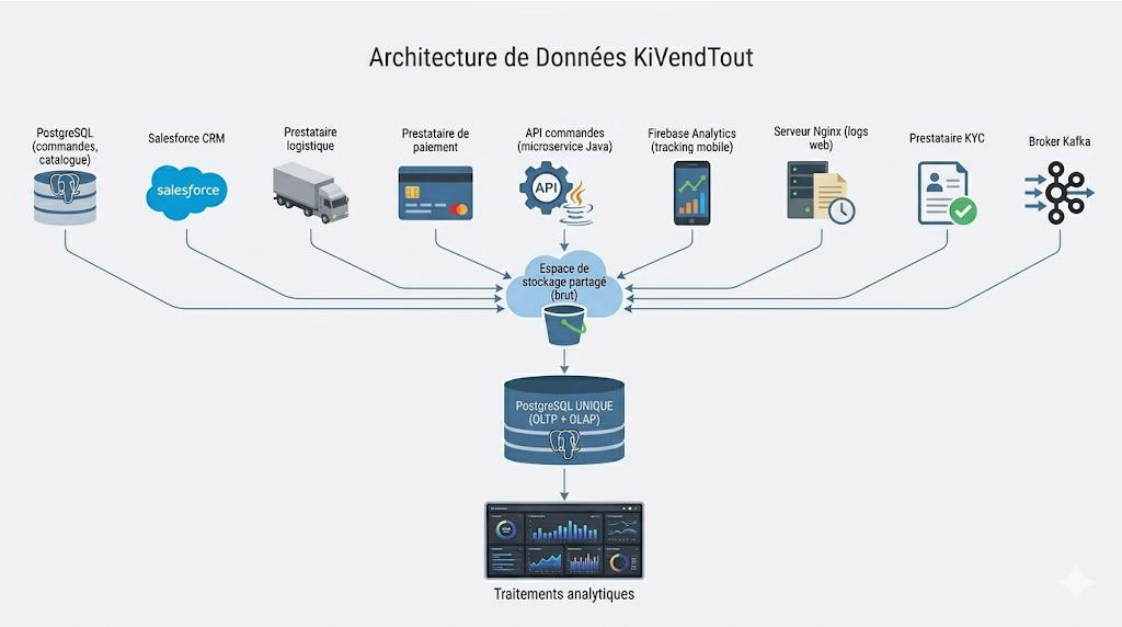
L'architecture data de KiVendTout repose sur un ensemble de systèmes hétérogènes, historiquement déployés sans vision d'intégration globale. L'entreprise a fait le choix stratégique d'une infrastructure 100 % on-premise, motivée par la maîtrise des données sensibles, les exigences RGPD et KYC, et la volonté de limiter la dépendance aux fournisseurs cloud.

Neuf systèmes sources coexistent sans couche d'intégration commune :

Système	**Responsable**	**Mode**	**Format**
PostgreSQL (commandes, catalogue)	Équipe Applicative	Batch (toutes les 15 min)	JSON
Salesforce CRM	Équipe Métier	Batch quotidien	CSV via SFTP
Prestataire logistique	Équipe Partenaire	Batch quotidien	Excel via SFTP
Prestataire de paiement	Équipe Partenaire	Streaming	JSON via Kafka
API commandes (microservice Java)	Équipe Applicative	Streaming	JSON via Kafka
Firebase Analytics (tracking mobile)	Équipe Applicative	Batch quotidien	JSON/NDJSON via SFTP
Serveur Nginx (logs web)	Équipe Infrastructure	Batch horaire	Texte/JSON
Prestataire KYC	Équipe Partenaire	Batch mensuel	CSV
Broker Kafka	Équipe Applicative	Streaming	JSON

Les données issues de ces systèmes sont déposées dans un espace de stockage partagé sans standardisation ni contrôle systématique de la qualité (Google Drive). Aucune couche de transformation intermédiaire ne sépare les données brutes des indicateurs exposés aux

dashboards. Les traitements analytiques s'exécutent directement sur la base PostgreSQL transactionnelle, sans instance dédiée à l'OLAP.



2.1.2 Limites identifiées et incidents significatifs

Cette architecture a révélé trois catégories de dysfonctionnements majeurs à mesure que les volumes ont crû : environ 50 000 événements web par jour, 30 000 événements mobiles, et 25 000 tentatives de paiement dont 8 % frauduleuses.

Absence de séparation OLTP / OLAP. En novembre 2025, une requête analytique exécutée directement sur la base PostgreSQL transactionnelle a saturé les ressources du système. Le site est resté indisponible pendant plusieurs heures, avec un impact direct sur le chiffre d'affaires. Cet incident a mis en évidence l'impossibilité structurelle de faire coexister charges opérationnelles et charges analytiques sur une même instance.

Défaillances silencieuses des pipelines. Un échec non détecté d'un script d'alimentation des Dashboard a conduit les équipes métier à exploiter des données obsolètes pendant plusieurs jours, sans qu'aucune alerte ne soit levée. L'absence de monitoring, d'orchestrateur centralisé et de contrôles qualité automatisés a amplifié l'impact de cet incident.

Qualité des données non maîtrisée. Les flux logistiques contiennent des doublons, les formats de dates varient selon les partenaires, certains champs présentent des incohérences de types (montants, booléens), et les schémas évoluent sans versionnement. Ces problèmes rendent les traitements instables et limitent la fiabilité des analyses produites.
À ces trois limites s'ajoutent deux contraintes spécifiques qui imposent une évolution architecturale. La détection de fraude, réalisée entièrement en batch avant 2026, intervient avec plusieurs heures de décalage, entraînant des pertes financières significatives. La vérification d'identité pour les produits réglementés repose sur une validation manuelle pouvant atteindre 24 heures, incompatible avec les attentes e-commerce et non scalable face à la croissance des volumes.

Le tableau ci-dessous, fait une synthèse récapitulative des différents problèmes de cette stack actuelle :

Dimension	**Situation existante**	**Problème constaté**
Stockage analytique	PostgreSQL unique (OLTP + OLAP)	Saturation nov. 2025, indisponibilité du site
Orchestration	Scripts cron non supervisés	Échecs silencieux, données obsolètes exploitées
Qualité des données	Aucun contrôle automatisé	Doublons, formats incohérents, schémas instables
Détection de fraude	Traitement batch uniquement	Décalage de plusieurs heures, pertes financières
Vérification KYC	Validation manuelle	Délais jusqu'à 24h, risque d'erreur, coût élevé
Lignage	Absent	Aucune traçabilité des transformations
Intégration des sources	Ad hoc, sans standardisation	Hétérogénéité des formats, absence de gouvernance

2.2 Architecture globale proposée

2.2.1 Principes directeurs

Chaque composant de notre architecture vient régler un problème précis que nous avons identifié dans la section 2.1.2. Rien n'a été choisi par défaut : si une brique est là, c'est parce qu'on s'est heurté à une limite concrète.

La saturation que nous avons vécue en novembre 2025 a mis fin à un débat qu'on repoussait depuis un moment. C'est simple : faire tourner de grosses requêtes analytiques sur la base de production du site n'est plus tenable. Nous avons donc opté pour une solution radicale avec deux instances PostgreSQL complètement séparées. La première s'occupe uniquement du transactionnel, tandis que la seconde stocke les indicateurs agrégés pour nos tableaux de bord. Il n'y a aucun pont entre les deux. L'isolation est devenue physique, et ce n'est plus une simple règle de bonne conduite qu'on pourrait facilement contourner.

De leur côté, les scripts cron nous ont montré leurs limites d'une bien mauvaise façon : ils plantaient sans prévenir personne. Airflow vient remplacer tout ça en nous offrant ce qui nous faisait cruellement défaut. Nous avons désormais une vue centralisée sur l'état des pipelines, des relances paramétrables et des alertes claires en cas de pépin. Si un pipeline tombe, il ne continue plus d'alimenter nos dashboards en douce avec de vieilles données. Il s'arrête, il prévient, et il bloque le reste de la chaîne.

Avant, le contrôle qualité était tout simplement inexistant. Les métiers prenaient des décisions sur des données dont personne n'était capable de garantir la fiabilité. Aujourd'hui, plusieurs tests automatisés tournent toutes les nuits sur nos tables d'indicateurs. Si on n'atteint pas au moins 80 % de réussite, on ne publie rien. Dans les faits, on tourne plutôt autour des 95 %. Et ce n'est pas un coup de chance : c'est parce que les anomalies connues à la source sont désormais gérées directement lors de la transformation.

Jusqu'à présent, la détection des fraudes se faisait par lots, avec un train de retard. Chaque heure perdue laissait le champ libre à de nouvelles transactions frauduleuses. Le passage au streaming a complètement changé la donne. Dès qu'un paiement ou une commande est généré, il est analysé en continu. Pour les règles de base (un score de risque trop haut, un pays suspect ou un flag évident), l'alerte part en moins d'une seconde. Pour les scénarios plus complexes qui nécessitent d'analyser plusieurs événements sur une période donnée, on reste sous la barre des deux secondes.

Côté KYC, tout reposait sur des agents qui lisaient les pièces d'identité et validaient manuellement les dossiers. Difficile de scaler un tel processus sans devoir embaucher en permanence. Nous avons donc mis en place une API dédiée pour s'en charger.

Concrètement, le document passe dans un moteur de reconnaissance de caractères qui extrait la date de naissance et calcule l'âge. Si tout est lisible, la décision tombe en moins de 5 secondes. Les agents ne gèrent plus que les documents très dégradés ou les cas ambigus, ce qui représente aujourd'hui moins de 10 % du volume global.

Autre gros point noir de l'ancien système : l'absence de linéage. Dès qu'un chiffre paraissait suspect sur un tableau de bord, c'était un vrai casse-tête pour retrouver son origine ou les traitements qu'il avait subis. L'intégration d'OpenLineage dans Airflow a réglé ça sans qu'on ait besoin de modifier une seule ligne de code dans nos pipelines. À chaque exécution, des données de traçabilité sont envoyées automatiquement à un serveur central. Ce dernier retrace alors tout l'historique sous forme de graphe, depuis la source brute jusqu'au dashboard final.

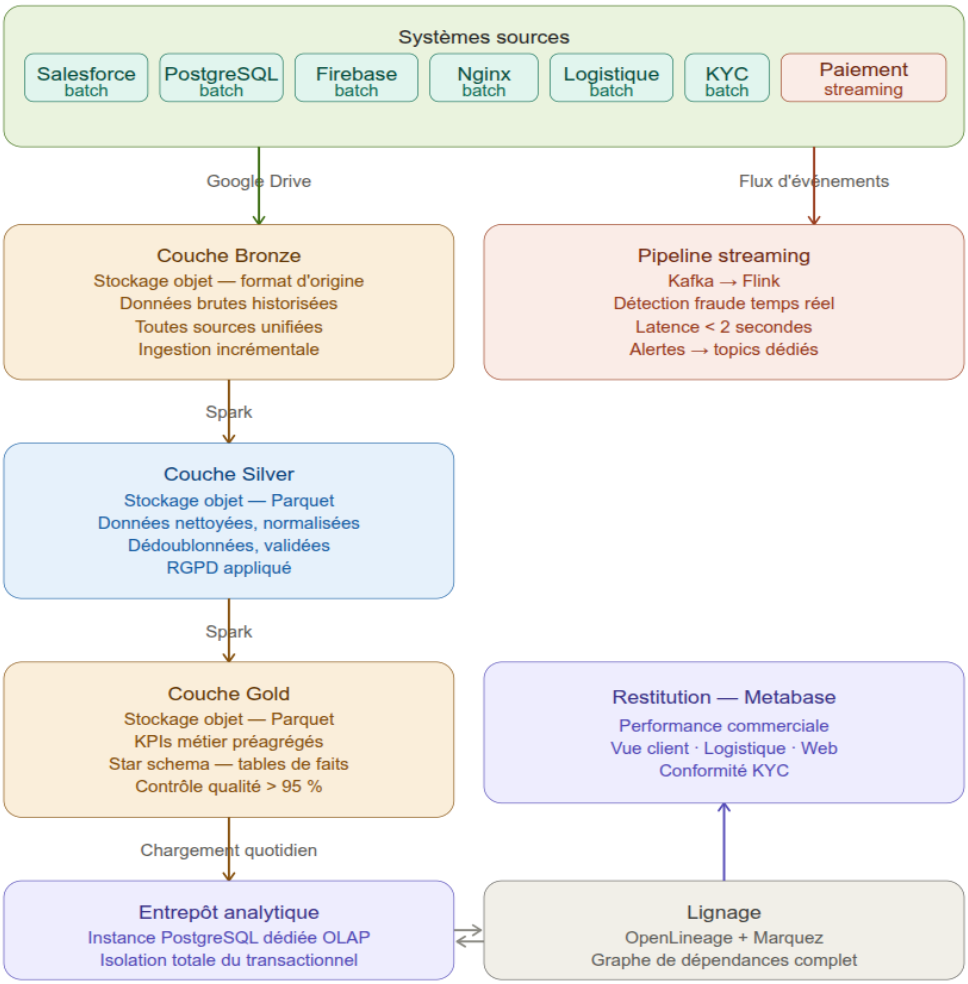
Enfin, avec nos sources très hétérogènes, chaque pipeline devait se débrouiller avec son propre format de données jusque dans les couches analytiques. L'architecture Medallion coupe court à ce problème. Désormais, toutes les données brutes arrivent dans leur format d'origine dans une zone d'atterrissage commune. Elles sont ensuite nettoyées et normalisées indépendamment dans une couche intermédiaire. Ainsi, la couche des indicateurs ne voit plus que de la donnée propre et homogène au format Parquet. Fini les prises de tête avec des CSV mal formés, des JSON imbriqués ou des logs bruts.

Pour résumer, l'ensemble de la plateforme repose aujourd'hui sur trois principes fondateurs :

- **Idempotente :** on peut relancer le calcul d'une couche depuis la précédente à tout moment, sans perdre de données ni créer de doublons.
- **Observable :** chaque action génère automatiquement ses propres logs, métriques et événements de traçabilité.
- **Modulaire :** l'ajout d'une nouvelle source ou d'un nouvel indicateur n'impacte en rien les composants qui tournent déjà.

2.2.2 Vue d'ensemble

La plateforme articule plusieurs flux de traitement distincts : des flux batch, chacun orchestré par un pipeline Airflow — terme désignant un graphe de tâches ordonnancées et supervisées — et des flux de traitement continu opérés par Kafka et Flink. L'orchestration des flux batch est centralisée dans Airflow, dont l'interface web permet de superviser l'exécution de l'ensemble des pipelines. Le lignage de chaque transformation est émis automatiquement vers un serveur de collecte dédié sans modification du code des pipelines.



2.2.3 La couche Bronze : zone d'atterrissage unifiée

La couche Bronze constitue la zone d'atterrissage unique de toutes les données entrantes, hébergée dans un système de stockage objet compatible avec le standard S3. Les données y sont conservées dans leur format d'origine, sans aucune transformation métier. Ce principe garantit la possibilité de rejouer l'intégralité des traitements depuis la source en cas d'incident — si une correction doit être apportée en aval, la donnée brute est toujours disponible pour repartir de zéro.

Les sources batch alimentent cette couche via un mécanisme de détection incrémentale : seuls les fichiers nouveaux ou modifiés depuis le dernier passage sont téléchargés et transférés vers le stockage objet. Les sources streaming — événements de paiement et événements de commande — sont consommées quotidiennement depuis les topics Kafka et sérialisées en fichiers partitionnés par date, constituant le pont entre pipeline streaming et pipeline batch.

Chaque objet stocké en Bronze est enrichi de métadonnées systématiques :

- Identifiant de la source d'origine
- Date de modification du fichier source
- Horodatage d'ingestion

Ce mécanisme garantit l'idempotence de l'ingestion et la traçabilité complète de l'origine de chaque donnée.

2.2.4 Les couches Silver et Gold : transformation et valorisation

La couche Silver est construite par des jobs PySpark indépendants, un par source. Chaque job applique les transformations nécessaires à produire des données fiables et homogènes :

- Dédoublonnage sur les identifiants métier
- Normalisation des types et des formats de dates vers un standard unifié
- Validation des valeurs par expressions régulières
- Canonicalisation des valeurs textuelles (casse, encodage, valeurs aberrantes)

Le résultat est stocké au format Parquet, format colonnaire optimisé pour les lectures analytiques, partitionné selon les axes de requête les plus fréquents.

La couche Gold est construite par des jobs Spark supplémentaires, chacun produisant une table d'indicateurs métier préagrégée destinée à l'entrepôt analytique. Ces tables couvrent cinq domaines :

- La vision client : scoring RFM, segmentation, score de santé client
- La performance produit : funnel de conversion, taux d'abandon panier
- La conformité KYC : taux de validation, respect des SLA réglementaires
- Les KPIs logistiques : ponctualité par transporteur, délais au 95e percentile
- La performance commerciale et web : revenus par catégorie, top produits, métriques de trafic

Cette séparation résout directement le problème de l'instance PostgreSQL unique : les requêtes analytiques de Metabase s'exécutent sur une instance dédiée à l'analytique, sans aucune interaction avec la base transactionnelle du site e-commerce.

Un contrôle qualité automatisé s'exécute quotidiennement après chaque chargement de l'entrepôt. Il vérifie :

- La nullité des clés primaires
- La positivité des comptages
- La cohérence des intervalles de valeurs
- La cohérence des agrégats croisés

Si le taux de succès global est inférieur à 80 %, le pipeline échoue et bloque la diffusion des données aux équipes métier. Sur les données de production, le score de qualité dépasse 95 % sur l'ensemble des tables Gold.

2.2.5 Le pipeline streaming : détection de fraude en temps réel

Le pipeline streaming répond à la limite la plus critique de l'architecture existante : le décalage de plusieurs heures entre la survenance d'une transaction suspecte et sa détection.

Kafka reçoit en continu les événements de paiement et de commande sur des topics dédiés, configurés avec une rétention de sept jours. Flink consomme ces topics et applique des règles de détection basées sur le temps événementiel :

- Fenêtres glissantes sur cinq minutes pour les attaques en vélocité
- Corrélation géographique pour les localisations inhabituelles
- Analyse de la régularité temporelle pour les comportements automatisés
- Corrélation croisée entre flux de paiements et flux de commandes pour les patterns invisibles sur chaque flux pris isolément

Lorsqu'une règle est déclenchée, une alerte enrichie est produite en moins de deux secondes vers un topic Kafka dédié, visualisable dans Grafana. Ce délai représente une réduction considérable par rapport au traitement batch antérieur.

L'ensemble de la plateforme regroupe finalement plus d’une dizaine de services conteneurisés. Le tableau suivant en présente les principaux :

Technologie	Rôle	Problème résolu
Stockage objet (MinIO)	Couches Bronze, Silver et Gold	Hétérogénéité des sources, absence d'historisation
Apache Spark 3.5	Transformations batch	Qualité des données non maîtrisée
Apache Airflow 2.9	Orchestration des pipelines batch	Scripts cron non supervisés
Apache Kafka	Messagerie streaming	Détection fraude en décalage
Apache Flink 1.19	Traitement streaming temps réel	Latence de détection fraude
PostgreSQL Analytics	Entrepôt analytique dédié	Saturation de l'instance unique nov. 2025

Metabase	Tableaux de bord métier	Absence de restitution standardisée
OpenLineage + Marquez	Lignage des transformations	Absence de traçabilité
Grafana + Prometheus	Monitoring technique	Défaillances silencieuses non détectées

Chapitre 3 : Pipeline Batch

Le chapitre précédent a établi les fondations architecturales : une Medallion Bronze-Silver-Gold, plusieurs flux de traitement coexistants, des choix techniques justifiés par les contraintes de KiVendTout. Ce chapitre entre dans le détail d'implémentation du pipeline batch — l'épine dorsale de la plateforme, qui traite quotidiennement l'ensemble des sources structurées, les transforme en indicateurs fiables, et les rend disponibles aux équipes métier avant 10h00 chaque matin.

3.1 Ingestion : Drive → Bronze

3.1.1 Sources concernées et fichiers ingérés

L'ingestion batch couvre cinq sources hétérogènes, toutes déposées dans des espaces Google Drive dédiés par les équipes responsables :

Source	Format	Fréquence	Responsable
Firebase Analytics	JSON structuré et imbriqué	Quotidien	Équipe Applicative
Salesforce CRM	CSV encodé UTF-8	Hebdomadaire (lundi)	Équipe Métier
Prestataire KYC	CSV multi-fichiers	Mensuel (1er du mois)	Équipe Partenaire
Partenaire logistique	CSV volumétrique	Toutes les 4h	Équipe Partenaire
Logs Nginx	Texte brut	Quotidien	Équipe Infrastructure

À ces cinq sources batch s'ajoutent les événements de paiement et de commande issus de Kafka, ingérés par un DAG dédié et détaillés en section 3.1.3.

3.1.2 Orchestration : la fabrique de DAGs et la mécanique Drive → Bronze

L'ingestion depuis Google Drive repose sur un système très efficace : une "fabrique" de DAGs Airflow. Au lieu de copier-coller la même logique pour chaque source, cette fabrique génère automatiquement un DAG par source, en se basant sur un simple dictionnaire de configuration.

Dans ce dictionnaire, on définit l'ID du dossier Drive, la fréquence de mise à jour voulue et les métadonnées de suivi. Si on veut ajouter une nouvelle source, il suffit d'ajouter une ligne dans ce fichier. On ne touche absolument pas au code d'ingestion.

Le gros avantage : cette logique est centralisée dans un module unique que tous les DAGs utilisent.

Pour chaque source, le processus se déroule toujours en quatre temps :

1. **Inventaire :** On liste tous les fichiers du dossier Drive (et de ses sous-dossiers) via l'API Google.

2. **Comparaison :** On vérifie ce qui est déjà présent dans notre stockage objet pour repérer uniquement les fichiers nouveaux ou modifiés.

3. **Téléchargement :** On télécharge ces fichiers directement en mémoire, sans jamais écrire sur le disque du serveur.

4. **Dépôt :** On les envoie vers notre zone de stockage Bronze, accompagnés de leurs métadonnées.

L'ensemble est conçu pour être incrémental. Un fichier n'est retéléchargé que si la version sur le Drive est plus récente que celle qu'on a déjà. Résultat : on peut relancer le même DAG autant de fois qu'on veut, il n'y aura jamais de doublons.

Les horaires sont calés sur les fenêtres de disponibilité réelles des données :

Source	Fréquence	Justification
Logs Nginx	Quotidien à 01h00 UTC	Logs du jour précédent complets à minuit
Firebase Analytics	Quotidien à 02h00 UTC	Export Firebase disponible en début de nuit
Documents KYC	Quotidien à 03h00 UTC	Traité tôt, fréquence mensuelle côté source
CRM Salesforce	Hebdomadaire le lundi à 06h00 UTC	Export Salesforce hebdomadaire
Données logistiques	Toutes les 4 heures	Mises à jour transporteur fréquentes

Chaque DAG est configuré avec deux tentatives automatiques espacées de cinq minutes, sans rattrapage des exécutions historiques manquées, et avec une contrainte d'exécution unique à la fois par source.

3.1.3 Alerting en cas de panne

L'alerting est géré de manière centralisée directement dans la configuration d'Airflow. Le principe est simple : si un DAG échoue, même après avoir épuisé toutes ses tentatives de relance automatique, il envoie immédiatement une notification à l'équipe data.
De plus, Airflow conserve en interne tous les résumés d'ingestion via son système de communication inter-tâches (XComs). On y retrouve le nombre de fichiers traités, le volume de données transféré et le statut de chaque opération.

L'avantage clé : Cette approche permet de réaliser un audit complet a posteriori, sans avoir besoin de se connecter directement à l'espace de stockage.

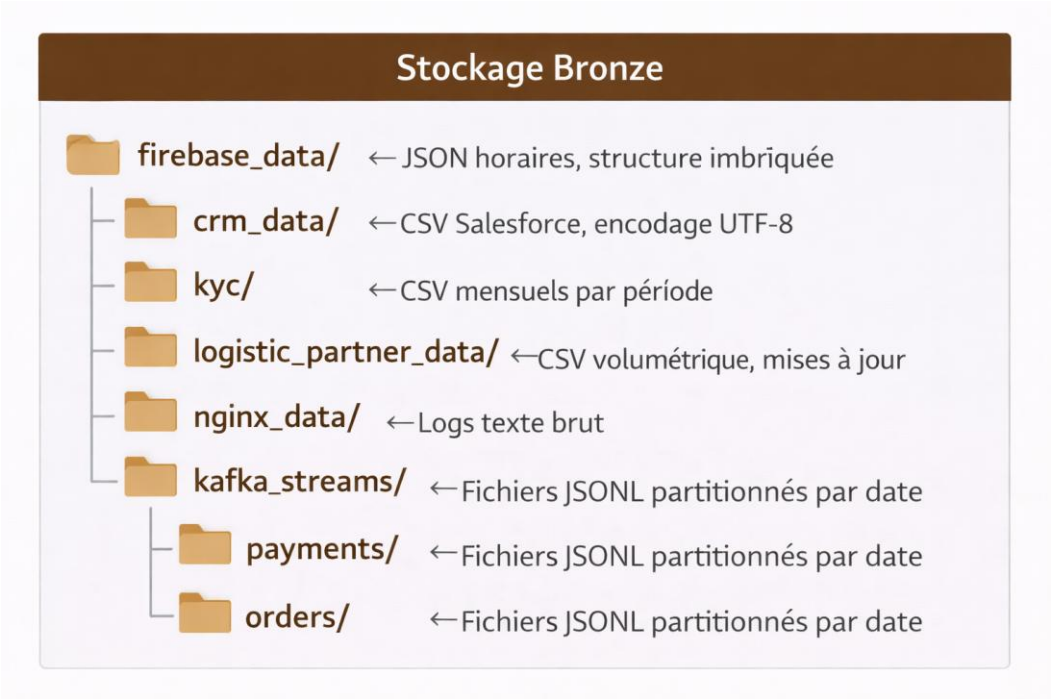
3.1.4 Ingestion Kafka → Bronze

En parallèle de l'ingestion depuis le Drive, un DAG spécifique prend le relais chaque nuit à minuit pour consommer nos deux flux Kafka : les paiements et les commandes. C'est lui qui fait le pont entre le monde du streaming et celui du batch. L'objectif est de prendre tous les événements produits en temps réel durant la journée et de les historiser dans la zone Bronze. Ce sont ces données qui viendront ensuite nourrir nos analyses RFM, nos tableaux de bord des ventes et la détection de fraude a posteriori.

Pour garantir la robustesse de ce consommateur, nous nous appuyons sur deux sécurités clés : D'abord, on ne valide la position de lecture dans Kafka qu'une fois les données correctement déposées sur le stockage objet. En cas de plantage intermédiaire, c'est l'assurance absolue de ne perdre aucun message. Ensuite, si le système ne détecte aucun nouveau message pendant dix secondes, il coupe la connexion proprement, évitant ainsi à la tâche de rester bloquée indéfiniment.

Côté stockage, les fichiers générés sont systématiquement partitionnés par date et horodatés, ce qui nous permet de reconstituer la chronologie exacte en cas de rejeu des données. Enfin, chaque objet est enrichi de ses propres métadonnées, indiquant clairement sa source, son topic d'origine, le volume de messages qu'il contient et l'heure exacte de son ingestion.

3.1.5 Organisation dans Bronze



3.2 Transformation : Bronze → Silver

3.2.1 Principes communs aux jobs Silver

Dans la couche Silver, des jobs PySpark indépendants (un pour chaque source) prennent le relais. Leur mission : transformer les données brutes de la zone Bronze en données propres, fiables et structurées de la même manière.
Tous ces jobs utilisent la même configuration pour se connecter au stockage objet, et surtout, ils écrivent tous leur résultat final dans un seul et même format : **Parquet**.
Pourquoi le format Parquet ? Ce n'est pas un choix au hasard, c'est un format orienté colonne qui nous offre trois avantages majeurs pour la suite :

1. **Des lectures optimisées :** Lors des traitements de la couche Gold, au lieu de charger des lignes entières en mémoire, on ne lit que les colonnes dont on a réellement besoin pour une analyse précise.
2. **Un stockage allégé :** Grâce à sa compression native très performante, Parquet divise notre espace de stockage par quatre, voire par six, par rapport aux fichiers d'origine (CSV, JSON, etc.).
3. **Des types préservés :** Contrairement à un CSV où tout est texte, Parquet conserve nativement les types de données (entiers, dates, booléens...), ce qui évite bien des erreurs de conversion en aval.

3.2.2 Source Salesforce CRM

Description métier : Le fichier CRM est l'export hebdomadaire de la base clients Salesforce. Il constitue la source de référence pour l'identité commerciale de chaque client : coordonnées, segmentation, historique d'achat agrégé et statut KYC. L'équipe métier l'utilise principalement pour le suivi des segments commerciaux et le pilotage des actions marketing.

Schéma source :

Champ	Type brut	Description métier
SALESFORCE ID	Texte	Identifiant interne Salesforce
CUSTOMER ID	Texte	Identifiant client partagé entre systèmes
CIVILITE	Texte	Civilité du client
PRENOM	Texte	Prénom — donnée personnelle directe
NOM	Texte	Nom de famille — donnée personnelle directe
EMAIL	Texte	Adresse email — donnée personnelle directe
TELEPHONE	Texte	Numéro de téléphone — donnée personnelle directe
DATE NAISSANCE	Date	Date de naissance — donnée personnelle sensible
ADRESSE RUE	Texte	Adresse postale — donnée personnelle directe
VILLE	Texte	Ville de résidence

CODE POSTAL	Texte	Code postal
PAYS	Texte	Pays en nom complet (ex. "France")
STATUT	Texte	Statut du compte (Actif, Inactif)
SEGMENT CLIENT	Texte	Segment commercial (Bronze, Silver, Gold, Platinum)
LEAD SCORE	Entier	Score de qualification commerciale (0-1000)
CANAL ACQUISITION	Texte	Canal d'acquisition du client
CATEGORIE PREFEREE	Texte	Catégorie de produits préférée
NB COMMANDES	Entier	Nombre total de commandes passées
LIFETIME VALUE EUR	Décimal	Valeur vie client en euros
KYC REQUIS	Booléen	Indique si le client doit passer un contrôle KYC
KYC VALIDE	Booléen	Indique si le KYC a été validé
KYC DATE VALIDATION	Date	Date de validation du KYC
NEWSLETTER OPTIN	Booléen	Consentement newsletter
DATE CREATION CRM	Date	Date de création du compte dans Salesforce
DATE MODIFICATION	Date	Date de dernière modification
PROPRIETAIRE SF	Texte	Responsable commercial assigné

Problèmes identifiés : Nous avons repéré une vingtaine de doublons sur les identifiants clients, générés par des synchronisations multiples. Côté KYC, les champs sont vides pour 85 % des clients, mais c'est un comportement attendu que nous avons documenté après en avoir discuté avec l'équipe conformité. Nous avons aussi remarqué que le champ du pays utilise les noms complets en français au lieu des codes ISO habituels. Enfin, le fichier d'origine est encodé avec un marqueur d'ordre d'octet (BOM) qui fait planter la lecture si l'on n'y prend pas garde.

Transformations appliquées : Pour nettoyer tout cela, nous avons dédoublonné les enregistrements en gardant à chaque fois la ligne avec la date de modification la plus récente. Les noms de pays ont été convertis au format ISO alpha-2 ("France" devient "FR"). Toutes les dates ont été normalisées au standard ISO 8601, et nous avons renommé les colonnes pour les passer en minuscules, sans espaces.

Application RGPD : Dès la couche Silver, nous supprimons purement et simplement les données personnelles directes comme le prénom, le nom, le téléphone et l'adresse postale, car elles ne nous servent à rien pour l'analytique. L'adresse email est conservée, mais hachée en SHA-256 pour permettre les jointures entre nos sources sans jamais l'exposer en clair. La date de naissance disparaît au profit d'un simple âge calculé. Enfin, le champ ***PROPRIETAIRE SF,*** qui désigne nommément un collaborateur, est également retiré de la base.

Partitionnement : Le résultat final est partitionné par pays. C'est un choix très pragmatique motivé par nos requêtes Gold les plus courantes (comme la segmentation par marché ou les KPI par zone) qui filtrent systématiquement sur cette dimension géographique. Grâce à cette approche, Spark évite de tout scanner et ne charge en mémoire que les données strictement nécessaires.

3.2.3 Source Firebase Analytics

Description métier : Les fichiers Firebase sont l'export horaire du SDK de tracking intégré dans l'application mobile de KiVendTout. Ils tracent le comportement des utilisateurs dans l'application : navigation, consultations de produits, ajouts au panier, achats. L'équipe produit s'en sert pour analyser les funnels de conversion et identifier les points de friction dans le parcours d'achat.

Schéma source :

Champ	Type brut	Description métier
event_id	Texte	Identifiant unique de l'événement
event_timestamp	Texte ISO 8601	Horodatage de l'événement en millisecondes UTC
device_id	Texte	Identifiant de l'appareil
session_id	Texte	Identifiant de session utilisateur
user_id	Texte	Identifiant client (peut valoir "anonymous")
platform	Texte	Plateforme (ios / android)
app_version	Texte	Version de l'application
os_version	Texte	Version du système d'exploitation
country	Texte	Code pays ISO alpha-2
language	Texte	Langue de l'interface
screen_resolution	Texte	Résolution de l'écran
network_type	Texte	Type de réseau (wifi, 3g, 4g)
event_name	Texte	Type d'événement (app_open, screen_view, view_item, add_to_cart, purchase…)
params	Objet JSON	Paramètres variables selon le type d'événement

L'objet params est la principale difficulté de cette source : sa structure varie selon chaque type d'événement. Un événement view_item contient product_id, price_eur et currency ; un événement screen_view contient screen_name, previous_screen et time_on_prev_screen_ms ; un événement app_open contient source et campaign_id. Le job Spark doit gérer ce schéma variable sans rejeter les événements aux structures inattendues.

Problèmes identifiés :

- L'objet des paramètres est très variable d'un type d'événement à l'autre.
- Nous remarquons également la présence de sessions automatisées, que l'on peut repérer grâce à des schémas d'identifiants typiques des bots.
- Enfin, les visiteurs anonymes représentent une part importante du trafic : s'ils sont bien conservés pour les vues d'ensemble et les analyses globales, nous les excluons logiquement de l'étude des comportements individuels.

Transformations appliquées :

- Aplatissement de l'objet params par pivot conditionnel selon le type d'événement, produisant des colonnes spécialisées nulles quand non applicables
- Conversion des horodatages de millisecondes vers des timestamps UTC normalisés
- Filtrage des sessions de test et des bots identifiés
- Extraction de la date pour le partitionnement

Application RGPD : Le champ device_id est pseudonymisé grâce à un hachage. Cela nous permet de croiser les sessions entre elles sans identifier directement l'appareil concerné. Quant aux user_id anonymes, ils sont conservés tels quels. Le principe est simple : une fois dans la couche Silver, absolument aucune donnée ne permet une réidentification directe.

Partitionnement : Le résultat est partitionné par event_date et vent_name. Ce double partitionnement reflète les deux axes de requête dominants des jobs Gold : les analyses temporelles (évolution du trafic jour par jour) et les analyses par type d'événement (funnel view_item → add_to_cart → purchase). Spark ne charge que les combinaisons date/événement nécessaires à chaque calcul.

3.2.4 Source logistique partenaire

Description métier : Le fichier logistique est fourni toutes les quatre heures par le partenaire transporteur. Il recense l'état de chaque livraison en cours ou terminée. L'équipe supply chain l'utilise pour suivre la ponctualité des transporteurs, identifier les livraisons en retard et piloter les négociations contractuelles.

Schéma source :

Champ	Type brut	Description métier
livraison_id	Texte	Identifiant unique de la livraison
order_id	Texte	Identifiant de la commande associée
customer_id	Texte	Identifiant du client destinataire
transporteur	Texte	Nom du transporteur
numero_suivi	Texte	Numéro de suivi colis
statut_livraison	Texte	Statut courant (Livré, En transit, Retourné…)
entrepot_depart	Texte	Entrepôt d'expédition
ville_destination	Texte	Ville de livraison
pays_destination	Texte	Pays de livraison
code_postal_dest	Texte	Code postal de destination
poids_kg	Décimal	Poids du colis en kilogrammes
longueur_cm	Entier	Longueur en centimètres
largeur_cm	Entier	Largeur en centimètres
hauteur_cm	Entier	Hauteur en centimètres
frais_livraison_eur	Décimal	Frais de livraison facturés
date_commande	Date	Date de la commande
date_expedition	Date	Date d'expédition effective
date_livraison_prevue	Date	Date de livraison promise au client
date_livraison_reelle	Date	Date de livraison effective (nulle si en cours)

retard_jours	Entier	Nombre de jours de retard (0 si ponctuel)
signature_requise	Booléen	Signature obligatoire à la livraison
tentatives_livraison	Entier	Nombre de tentatives de livraison
motif_echec_retour	Texte	Motif en cas d'échec ou de retour
assurance	Booléen	Présence d'une assurance colis
commentaire_livreur	Texte	Commentaire libre du livreur

Problèmes identifiés :

- Sur un total de 15 600 lignes, plus de 2 000 n'ont pas de date de livraison réelle. C'est un comportement attendu puisque ces livraisons sont toujours en cours, mais cela demande un traitement explicite de notre part.
- Nous avons repéré environ 90 lignes affichant un nombre de tentatives de livraison complètement aberrant, avec des valeurs atteignant parfois plusieurs centaines. Il s'agit clairement d'erreurs de saisie côté partenaire.
- La casse des statuts n'est absolument pas harmonisée : on se retrouve avec un mélange de "Livré", "livré" et "LIVRÉ" pour un même état.

Aller-retour métier : Le seuil de plafonnement des tentatives aberrantes et le traitement des livraisons sans date réelle ont été validés avec le responsable logistique. Les valeurs aberrantes sont conservées en Silver avec un indicateur de suspicion plutôt qu'être silencieusement supprimées.

Transformations appliquées :

- Normalisation des statuts en minuscules avec suppression des espaces superflus
- Plafonnement des tentatives de livraison à dix, au-delà marquées avec un indicateur tentatives_suspectes
- Calcul du délai réel de livraison en jours ouvrés
- Standardisation des formats de dates multi-variantes

Application RGPD : Comme le champ commentaire_livreur risque de contenir des informations personnelles saisies librement, nous le supprimons purement et simplement dès la couche Silver. Le customer_id, de son côté, est bien conservé, mais sous une forme pseudonymisée, ce qui nous permet de toujours pouvoir réaliser nos jointures avec la dimension client.

Partitionnement : Le résultat est partitionné par pays de destination et mois d'expédition. Les requêtes Gold les plus fréquentes portent sur des agrégats par transporteur et par période, avec filtrage géographique. Ce double partitionnement permet à Spark de limiter les lectures aux seules partitions concernées par chaque calcul mensuel.

3.2.5 Source KYC

Description métier : Le fichier KYC correspond à l'export mensuel transmis par notre prestataire de vérification d'identité. Il compile l'ensemble des dossiers traités au cours du mois, c'est-à-dire les vérifications manuelles qui étaient réalisées avant le déploiement de notre API automatisée. Pour l'équipe conformité, ce document est indispensable : il leur

permet de prouver que nous respectons bien nos obligations réglementaires concernant la vérification de l'âge pour la vente de produits sensibles.

Schéma source :

Champ	Type brut	Description métier
kyc_ref	Texte	Référence unique du dossier KYC
prestataire_ref	Texte	Référence interne du prestataire
customer_id	Texte	Identifiant du client vérifié
doc_type	Texte	Type de document (CNI, Passeport, Titre de séjour…)
doc_pays_emission	Texte	Pays d'émission du document
doc_date_expiration	Date	Date d'expiration du document
statut_verification	Texte	Décision (validé, refusé, en_attente)
motif_refus	Texte	Motif en cas de refus
agent_traitement	Texte	Nom de l'agent ayant traité le dossier — donnée personnelle
nb_tentatives	Entier	Nombre de soumissions du client
date_soumission	Date	Date de soumission du dossier
date_traitement	Date	Date de traitement par l'agent
date_validation	Date	Date de décision finale
commentaire_interne	Texte	Note interne libre — peut contenir des données personnelles
export_month	Texte	Mois de l'export (format YYYY-MM)

Problèmes identifiés :

- Le champ doc_type présente des variantes non normalisées : "CNI", "cni", "PASSEPORT", "N/A"
- Le champ statut_verification contient des variantes de casse et des valeurs "NULL" en chaîne de caractères
- Plusieurs dates d'expiration valent une date fictive lointaine utilisée par le prestataire pour signaler un document sans date lisible
- Plusieurs champs de date présentent un format inversé introduit par une version défectueuse de l'outil d'export

Aller-retour avec l'équipe conformité : Les dossiers en attente depuis plus de cinq jours ouvrés doivent être marqués par un indicateur de dépassement de SLA dès la couche Silver, sans attendre la décision finale. Cet indicateur alimente le tableau de bord de conformité pour une détection proactive.

Transformations appliquées :

- Canonicalisation des types de documents et des statuts en valeurs normalisées
- Remplacement des dates d'expiration fictives par des valeurs nulles explicites
- Correction des formats de dates inversés par un parser multi-formats
- Calcul du délai de traitement en jours ouvrés
- Ajout de l'indicateur de dépassement de SLA

Application RGPD : D'un point de vue réglementaire, c'est de loin notre source la plus sensible. C'est pourquoi le champ agent_traitement, qui mentionne nominativement un collaborateur, est systématiquement supprimé dès la couche Silver. Même chose pour le commentaire_interne, qui est écarté puisqu'il risque de contenir des données personnelles ajoutées en texte libre. La prestataire_ref, quant à elle, est bien conservée mais uniquement sous une forme hachée, et ce, exclusivement pour pouvoir gérer les éventuelles réclamations. Enfin, aucune information détaillée sur le document d'identité, qu'il s'agisse de sa nature, de son numéro ou de son contenu visuel, n'est envoyée vers les couches analytiques.

Partitionnement : Le résultat est partitionné par mois de soumission, en cohérence avec la fréquence mensuelle des exports et les agrégats mensuels produits en Gold.

3.2.6 Source Nginx

Description métier : Les logs Nginx correspondent à l'export quotidien du serveur web frontal de KiVendTout. Ils retracent l'intégralité des requêtes HTTP reçues par le site, qu'il s'agisse des pages consultées, des appels d'API, des erreurs rencontrées ou encore des temps de réponse. Ces données ont une double utilité. D'un côté, l'équipe infrastructure s'en sert pour surveiller la disponibilité du site et diagnostiquer d'éventuelles baisses de performance. De l'autre, l'équipe marketing les exploite pour en extraire l'ensemble de ses métriques de trafic.

Schéma source :

Champ	Type brut	Description métier
ip_address	Texte	Adresse IP du client — donnée personnelle
customer_id	Texte	Identifiant client si authentifié, "-" sinon
timestamp	Texte	Horodatage de la requête
method	Texte	Méthode HTTP (GET, POST…)
path	Texte	Chemin de la ressource demandée
protocol	Texte	Version du protocole HTTP
status_code	Entier	Code de statut HTTP de la réponse
response_size_bytes	Entier	Taille de la réponse en octets
referer	Texte	URL de la page source de la requête
user_agent	Texte	Navigateur et système d'exploitation du client
response_time_ms	Entier	Temps de traitement de la requête en millisecondes

Problèmes identifiés :

• Sur les 47 000 lignes analysées, environ 900 sont tronquées. Elles ont tout simplement été coupées en pleine écriture pendant la rotation des fichiers en fin de journée. Avec une proportion de 2 %, nous restons parfaitement dans la marge de tolérance de 5 % définie avec l'équipe infrastructure.

• Nous remarquons aussi la présence de caractères nuls dans certaines lignes. Ils sont causés par des requêtes HTTP malformées et rendent tout chargement en base de données complètement impossible.

• Enfin, les lignes que nous ne pouvons pas parser sont comptées puis directement écartées. Un signalement est toutefois prévu si leur volume vient à dépasser le seuil d'alerte autorisé.

Transformations appliquées :

- Parsing par expression régulière du format Combined Log Format
- Suppression des lignes tronquées ou non conformes
- Nettoyage des caractères nuls
- Extraction de la date pour le partitionnement
- Calcul de la famille de code HTTP (2xx, 3xx, 4xx, 5xx)

Application RGPD : Dès la couche Silver, les adresses IP sont tronquées au troisième octet. Concrètement, une adresse comme "192.168.1.45" se transforme en "192.168.1.xxx". Cette précaution les rend totalement anonymes et nous permet de respecter à la lettre les recommandations de la CNIL concernant les données de connexion. De son côté, le champ user_agent est conservé uniquement pour repérer les bots, sans pour autant permettre la moindre réidentification individuelle.

Partitionnement : le résultat final est découpé par date de log. C'est un choix très logique puisque les requêtes Gold liées aux métriques web sont systématiquement basées sur le temps, qu'il s'agisse d'évaluer la disponibilité du jour ou l'évolution du taux d'erreur sur une semaine. Grâce à cette organisation, Spark ne charge en mémoire que les journées réellement concernées par chaque calcul.

3.3 Transformation : Silver → Gold

3.3.1 Modélisation de l'entrepôt analytique : star schema

Pour notre entrepôt analytique, nous avons fait le choix d'un schéma en étoile. C'est une décision motivée par trois raisons très pragmatiques.

Premièrement, nos requêtes sur Metabase sont généralement de simples agrégations basées sur un axe métier, comme le chiffre d'affaires par mois et par catégorie, ou encore le taux de ponctualité par transporteur. Or, c'est exactement le type de jointure qu'un schéma en étoile vient optimiser.

Deuxièmement, cette structure offre une lisibilité incomparable pour les équipes métier. Le principe est limpide : on retrouve une table de faits au centre, des dimensions tout autour, et des jointures qui s'opèrent sur des clés entières.

Enfin, PostgreSQL, qui est la base analytique que nous avons retenue, s'avère capable de traiter ce type de schéma de manière très efficace, et ce, sans même avoir besoin d'un moteur orienté colonne spécifique.

Le schéma se compose de tables de faits et de tables de dimensions :

Tables de dimensions :

Table	Clé primaire	Description
dim_client	client_id	Profil client pseudonymisé, segment RFM, statut KYC
dim_produit	produit_id	Catalogue produit, catégorie, prix de référence
dim_date	date_id	Calendrier enrichi (jour, semaine, mois, trimestre, année, jour ouvré)
dim_transporteur	transporteur_id	Transporteur, type de service, zone de couverture
dim_pays	pays_id	Code ISO, nom, zone géographique, appartenance UE
dim_canal	canal_id	Canal d'acquisition, type (organique, payant, référent)

Tables de faits :

Table	Clé primaire	Granularité	Mesures principales
fact_commande	commande_id	Une ligne par commande	Montant total, nombre d'articles, remise, frais de livraison
fact_paiement	paiement_id	Une ligne par tentative de paiement	Montant, statut, score de risque, méthode, indicateur de fraude
fact_livraison	livraison_id	Une ligne par livraison	Délai réel, retard en jours, frais, nombre de tentatives
fact_evenement_web	evenement_id	Une ligne par requête HTTP agrégée	Nombre de requêtes, taux d'erreur, latence moyenne
fact_evenement_mobile	evenement_id	Une ligne par événement Firebase	Type d'événement, durée de session, paramètres métier
fact_kyc	kyc_id	Une ligne par dossier KYC	Statut, délai de traitement, indicateur de dépassement SLA

Jointures structurantes :

Au cœur de notre schéma, on trouve la table fact_commande. Elle s'articule autour de plusieurs dimensions : dim_client via le client_id, dim_date pour la date de commande, dim_pays pour le pays de livraison et dim_canal grâce au canal_id. Pour descendre au niveau de la ligne de commande, nous utilisons une table de liaison, fact_commande_produit, qui se connecte à dim_produit sur le produit_id.

La table fact_paiement est reliée à fact_commande par le commande_id, ce qui permet de gérer les cas où une même commande fait l'objet de plusieurs tentatives de paiement. Elle est aussi rattachée à dim_date via le date_id. De la même manière, fact_livraison fait le lien avec fact_commande sur le commande_id et avec dim_transporteur sur le transporteur_id.

Les tables fact_evenement_web et fact_evenement_mobile sont connectées aux dimensions temporelles et géographiques, dim_date et dim_pays. Concernant la partie mobile, fact_evenement_mobile se rattache aussi à dim_client dès que l'utilisateur est connecté. Pour tout ce qui touche aux vues de produits ou aux achats, elle se lie à dim_produit sur le produit_id.

Enfin, la table fact_kyc se joint à dim_client sur le client_id et à dim_date pour enregistrer la date de soumission du dossier.

3.3.2 Jobs Gold et indicateurs produits

Au-delà de l'alimentation du star schema, des jobs Spark produisent des tables d'agrégats préalculés destinés aux tableaux de bord Metabase les plus fréquemment consultés. Ces tables dénormalisées évitent de recalculer à la volée des agrégations coûteuses sur le star schema lors de chaque ouverture de dashboard.

Vue client 360° Produit une table synthétique par client croisant les dimensions CRM, commandes et comportement applicatif. Indicateurs : score RFM par dimension (Récence, Fréquence, Montant, chacun noté de un à cinq), segment RFM résultant, valeur vie client, panier moyen, date de dernière commande, indicateur de risque de churn au-delà de quatre-vingt-dix jours sans achat, et score de complétude du profil CRM. Cette table se joint à dim_client sur client_id.

Performance produit Produit une table de funnel de conversion par produit et par mois, croisant événements Firebase et commandes. Indicateurs : nombre de vues, taux de conversion vue vers panier, taux de conversion vue vers achat, revenu généré, volume d'abandons. Elle se joint à dim_produit sur produit_id et à dim_date sur date_id.

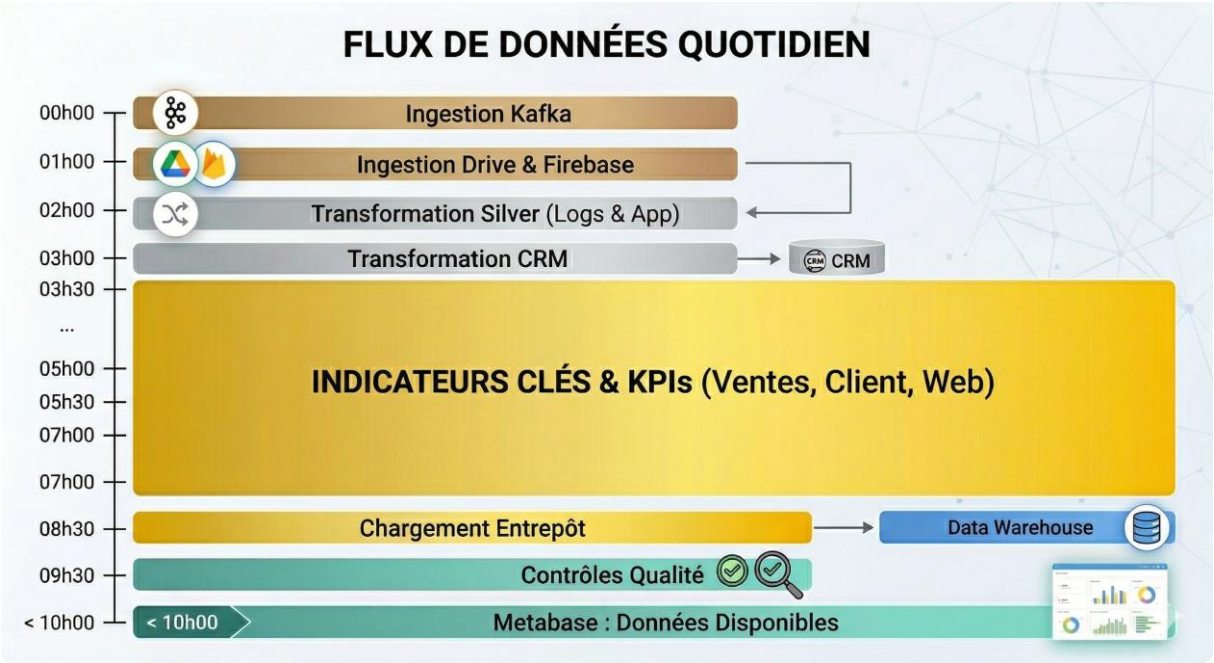
Conformité KYC Produit des agrégats mensuels par type de document : volumes de soumissions, validations, refus et dossiers en attente, taux correspondants, délai moyen et au quatre-vingt-quinzième percentile, distribution des motifs de refus, et score de conformité composite. Elle se joint à dim_date sur date_id.

KPIs logistiques Produit des indicateurs par transporteur et par mois : taux de ponctualité, taux de retour, délai moyen et au quatre-vingt-quinzième percentile, coût moyen par colis, score de performance composite. Elle se joint à dim_transporteur et dim_date.

KPIs commerciaux Produit deux tables : revenus mensuels par catégorie avec panier moyen et taux d'utilisation de coupons, et classement des produits par revenu et volume sur trente, quatre-vingt-dix et trois cent soixante-cinq jours. Elles se joignent à dim_produit et dim_date.

Web Analytics Produit deux tables : métriques agrégées par jour (disponibilité, taux d'erreur par famille HTTP, latence au quatre-vingt-quinzième percentile) et classement des endpoints par volume et taux d'erreur. Elles se joignent à dim_date.

3.3.3 Séquence temporelle complète du pipeline



3.4 Chargement dans l'entrepôt analytique

3.4.1 Gold → Entrepôt analytique

Un DAG dédié s'exécute chaque matin après la completion de l'ensemble des jobs Gold. Il charge les tables Parquet depuis le stockage objet vers l'instance PostgreSQL dédiée à l'analytique, strictement séparée de la base transactionnelle. Chaque table est chargée indépendamment, permettant des chargements en parallèle. La stratégie retenue recharge chaque table intégralement chaque jour, garantissant la cohérence sans gestion de delta.

Avant chargement, un nettoyage systématique supprime les caractères nuls hérités des logs Nginx et normalise les chaînes de caractères représentant des valeurs nulles en valeurs nulles SQL réelles.

3.4.2 Contrôles qualité automatisés

Un DAG de vérification s'exécute chaque matin après le chargement de l'entrepôt. Il orchestre des séries de contrôles couvrant l'ensemble des tables Gold, implémentés directement en SQL en s'inspirant des principes de Great Expectations. Quatre catégories de vérifications sont appliquées :

- Contrôles de complétude : absence de valeurs nulles sur les clés primaires et colonnes critiques
- Contrôles de volumétrie : présence d'au moins une ligne dans chaque table
- Contrôles d'intervalle : métriques en pourcentage dans [0, 100], scores dans leurs plages attendues, revenus non négatifs
- Contrôles de cohérence croisée : taux de traitement KYC, cohérence des agrégats entre tables liées

Les résultats sont persistés dans une table dédiée de l'entrepôt, constituant un historique consultable. Si le taux de succès du run courant descend sous quatre-vingts pourcents, le DAG échoue avec une erreur explicite, bloquant la diffusion vers Metabase. Sur les données de production, le score observé dépasse quatre-vingt-quinze pourcents sur l'ensemble des tables.

3.5 Restitution : Metabase

Les tables Gold et le star schema chargés dans l'entrepôt analytique alimentent Metabase, retenu pour sa prise en main immédiate par des équipes métier non techniques. Cinq domaines de tableaux de bord couvrent l'ensemble des besoins analytiques : Performance commerciale et suivi du revenu par catégorie et par période Vue client avec scoring RFM, segmentation et détection du churn Funnel de conversion produit, de la consultation à l'achat KPIs logistiques par transporteur avec suivi de la ponctualité Conformité KYC avec suivi des SLA réglementaires et distribution des motifs de refus L'accès à Metabase s'effectue désormais exclusivement via notre instance analytique dédiée. Il n'y a plus aucune connexion possible avec la base transactionnelle. Grâce à cette isolation technique stricte, nous éliminons de manière purement structurelle tout risque de saturation, comme celui que nous avons subi en novembre 2025.

Chapitre 4 : Pipeline Streaming

4.1 Kafka : Infrastructure de messaging

4.1.1 Architecture du broker et justification des choix de configuration

Kafka tourne en instance unique avec Zookeeper. Un seul broker, pas de réplication. C'est le bon niveau pour un on-premise en cours de construction : ça simplifie l'exploitation sans sacrifier les garanties importantes (durabilité, ordre par partition, possibilité de replay). Si on devait passer à plusieurs brokers un jour, les producteurs et consommateurs existants n'auraient rien à changer.

Quelques choix de config valent la peine d'être explicités.

Six partitions par défaut, parce que les jobs Flink tournent à six en parallèle. C'est un alignement voulu. Au-dessus, on crée du parallélisme fictif ; en-dessous, certains consommateurs restent les bras croisés.

Sept jours de rétention, pour deux usages bien distincts. Le DAG Airflow consomme les topics une fois par nuit : si ça casse pendant une semaine, les données sont toujours là. Et sur les jobs Flink, si on modifie les règles de détection de fraude, on peut rejouer le flux existant en repositionnant les offsets, sans repartir de zéro côté source.

La création automatique des topics est activée en dev pour démarrer sans cérémonie. En production, cette option serait désactivée : chaque topic avec sa propre config, gérée explicitement.

Le facteur de réplication du topic interne de suivi des offsets est à un. Avec un seul broker, c'est la seule valeur possible. Kafka refuserait de démarrer avec une valeur supérieure.

4.1.2 Listeners et accès réseau

Kafka expose deux listeners distincts.

Le premier couvre les communications internes au réseau Docker : les jobs Flink et le DAG Airflow s'y connectent via le nom de service DNS. Le second est exposé sur l'hôte, pour les outils externes : interface de visualisation des topics, scripts de test, clients locaux.

Cette séparation règle un problème classique : sans elle, un client externe recevrait une adresse interne Docker qu'il ne peut pas joindre depuis l'hôte.

4.1.3 Topics et rôles

Le pipeline streaming s'appuie sur sept topics aux rôles distincts :

Topic	Producteur	Consommateur	Rôle
payments	Service paiement	Flink + Airflow	Événements de paiement temps réel
orders	Service commandes	Flink + Airflow	Événements de commande temps réel
alerts_fraud_payments	Flink payments	Grafana / équipe fraude	Alertes fraude sur paiements
alerts_psp_payments	Flink payments	Grafana / équipe tech	Alertes taux d'erreur prestataire de paiement
alerts_orders	Flink orders	Grafana / équipe ops	Anomalies sur commandes
alerts_advanced_fraud	Flink advanced	Grafana / équipe fraude	Patterns de fraude avancés
alerts_fraud_crossed	Flink crossed	Grafana / équipe fraude	Fraude détectée par corrélation croisée

Les topics d'alertes servent de bus d'événements pour tout ce qui touche à la sécurité. Notification, blocage de compte, intervention manuelle : chaque composant s'abonne indépendamment, sans lien direct avec les jobs Flink.

Conséquence pratique : ajouter un nouveau consommateur d'alertes ne touche pas aux jobs de détection.

4.1.4 Interface de visualisation

Une interface web de supervision des topics est incluse dans la stack. Elle affiche en temps réel les offsets par groupe de consommateurs, le lag, et le contenu des messages individuels.

En pratique, elle sert surtout à deux choses pendant le développement : vérifier le format des alertes produites par Flink avant que Grafana ne les consomme, et repérer un lag qui grimpe sur un topic d'entrée quand un job Flink décroche.

4.2 Flink : Traitement streaming

4.2.1 Infrastructure commune aux quatre jobs

L'architecture streaming repose sur quatre jobs PyFlink indépendants, un par périmètre de détection. Chacun peut être déployé, mis à jour ou redémarré sans toucher aux autres, et dimensionné selon sa propre complexité.

Checkpointing : Tous les jobs checkpointent toutes les soixante secondes. Un checkpoint, c'est un instantané cohérent de l'état de tous les opérateurs, persisté dans un stockage fiable. En cas de redémarrage, Flink restaure depuis le dernier checkpoint et repositionne les

offsets Kafka au point exact de l'interruption. Sans ça, un redémarrage entraîne soit une perte d'événements, soit du double traitement.

Temps événementiel et watermarks : La stratégie de watermark tolère dix secondes de désordre dans l'arrivée des messages, ce qui est réaliste avec plusieurs producteurs sur un réseau interne. Le watermark progresse dans le temps événementiel et déclenche la fermeture des fenêtres : une fenêtre d'une minute attend que le watermark atteigne la fin de cette minute plus dix secondes avant de se fermer. Le timestamp utilisé est celui du message Kafka, pas l'horloge du job. Ce choix rend les fenêtres cohérentes même quand un lot arrive avec du retard réseau.

Connexion au stockage objet : L'accès à MinIO est configuré en mode chemin plutôt que sous-domaine. AWS S3 utilise des URLs virtuellement hébergées ; MinIO non. Sans ce paramètre, toutes les requêtes échouent.

Dépendances : Trois bibliothèques sont nécessaires : le connecteur Kafka pour Flink, le connecteur S3 pour l'écriture vers MinIO, et la bibliothèque cliente Kafka. Les versions sont alignées strictement avec celle de Flink déployée. Une incompatibilité mineure se manifeste à l'exécution par des erreurs de sérialisation.

Stratégie de lecture Kafka : Tous les jobs démarrent depuis le dernier message disponible, sans retraiter l'historique. C'est cohérent avec l'objectif : détecter la fraude en temps réel sur les nouvelles transactions. Le retraitement historique, quand nécessaire, se fait en repositionnant manuellement les offsets.

4.2.2 Job 1 : Détection de fraude sur les paiements

Ce job consomme le topic payments et branche trois traitements en parallèle sur le même flux parsé. Un seul flux d'entrée, plusieurs analyses indépendantes, sans dupliquer la consommation Kafka.

Détection directe : Un filtre identifie les paiements suspects selon quatre critères cumulables : indicateur de fraude explicitement positionné par le service de paiement, score de risque au-dessus de 70, pays dans la liste des zones à risque, ou combinaison d'un statut d'échec avec un score modéré. Deux critères ou plus déclenchent une alerte critique ; un seul suffit pour une alerte haute. L'alerte JSON enrichie, avec le détail des critères et le contexte complet du paiement, est publiée vers alerts_fraud_payments et archivée dans le stockage objet.

Agrégations par fenêtre : Des fenêtres d'une minute et de cinq minutes agrègent les paiements par devise. La devise est utilisée comme clé de partitionnement pour que tous les paiements d'une même devise arrivent sur la même instance d'opérateur, ce qui rend les agrégats cohérents. À la fermeture de chaque fenêtre : volume total, nombre de transactions, montant moyen, taux de succès, taux d'échec, taux de fraude, distribution des méthodes de paiement. Ces agrégats sont archivés pour alimenter les tableaux de bord en différé.

Alertes prestataire : Sur une fenêtre de cinq minutes par prestataire, le taux d'erreur est calculé à chaque fermeture. Au-dessus de 20%, une alerte est produite, ce qui permet de

détecter une dégradation avant que les clients ne remontent des problèmes. En parallèle, toute transaction dépassant 1 500 euros déclenche immédiatement une alerte, indépendamment du taux d'erreur.

4.2.3 Job 2 : Détection d'anomalies sur les commandes

Ce job consomme le topic orders avec la même architecture que le job précédent.

Détection d'anomalies : Trois critères déclenchent une alerte : montant total supérieur à 3 000 euros, remise dépassant 30% du sous-total (abus potentiel de code promo), ou annulation d'une commande de plus de quatre articles. Les alertes sont publiées vers alerts_orders et archivées.

Agrégations par fenêtre : Des fenêtres d'une minute et de cinq minutes par pays de livraison produisent : volume total, panier moyen, nombre moyen d'articles, distribution des types d'événements (création, expédition, livraison, annulation), taux d'annulation et distribution des méthodes de livraison. Le partitionnement par pays est utile pour la logistique : un pic d'annulations dans un pays peut signaler une dégradation d'un transporteur local avant qu'elle ne remonte autrement.

Alertes sur seuils : Deux détecteurs opèrent sur l'ensemble du flux. Le premier alerte si plus de dix annulations surviennent en une minute, ce qui peut indiquer un problème opérationnel ou une attaque coordonnée. Le second alerte si plus de 50% des commandes d'une fenêtre de cinq minutes optent pour la livraison express, signe d'un impact financier logistique anormal.

4.2.4 Job 3 : Détection de fraude avancée

Ce job implémente six patterns de fraude qui nécessitent tous un état persisté entre les événements. Cet état survit aux redémarrages grâce aux checkpoints Flink, contrairement à une variable en mémoire classique.

Test de carte : Plus de trois tentatives en moins de cinq secondes sur la même session de paiement. C'est le pattern classique de validation d'une carte volée avant une utilisation à montant élevé.

Fraude sur identifiant de carte : Les six premiers chiffres du numéro masqué sont confrontés à une liste d'émetteurs connus comme frauduleux. En production, cette liste serait synchronisée depuis une base externe mise à jour en continu.

Incohérence géographique : Divergence entre le pays de facturation et le pays de l'IP, combinée à un score de risque élevé.

Pattern automatisé. : Agent utilisateur identifié comme bot avec un temps de traitement inférieur à 300 ms, durée incompatible avec une saisie humaine.

Portefeuille de cartes : Même identifiant client avec plus de deux références de carte distinctes en dix minutes, indicateur de test séquentiel d'un lot de cartes volées.

Appareil partagé : Même empreinte d'appareil associée à plus de trois identifiants clients distincts en cinq minutes, indicateur d'un appareil utilisé dans une organisation frauduleuse.

Les alertes sont écrites vers alerts_advanced_fraud, le stockage objet, et directement dans la base de séries temporelles, ce qui alimente le tableau de bord Grafana en temps réel.

4.2.5 Job 4 : Détection de fraude croisée

Ce job est le seul du pipeline à consommer deux topics simultanément, payments et orders, pour détecter des patterns invisibles sur chaque flux pris isolément.

Paiement sur commande annulée : Un état persisté maintient les commandes annulées et les paiements réussis. Tout paiement abouti associé à une commande préalablement annulée déclenche une alerte, cas caractéristique de la réutilisation frauduleuse d'un identifiant de commande légitime.

Écart de montant : Détection d'un écart supérieur à 5% entre le montant du paiement et le total de la commande pour un même identifiant. Le seuil absorbe les différences de devise et d'arrondi légitimes. Au-delà, c'est une manipulation probable du montant entre la validation de la commande et l'exécution du paiement.

Vélocité croisée : Sur une fenêtre de deux minutes par identifiant client, détection combinée de plus de trois paiements et plus de deux commandes. Le critère croisé est plus discriminant que chaque seuil séparément : un client légitime peut enchaîner plusieurs commandes rapidement, mais la combinaison avec un volume élevé de paiements simultanés est rare dans un comportement normal.

IP partagée : Sur une fenêtre de cinq minutes, une même adresse IP associée à plus de deux identifiants clients distincts signale une organisation frauduleuse opérant depuis une adresse commune.

En complément de la détection, ce job publie des indicateurs en temps réel dans la base de séries temporelles : volume transactionnel par minute, taux de fraude instantané, revenus des commandes actives, produits les plus commandés sur cinq minutes, nombre d'alertes actives par type, santé des prestataires de paiement.

4.3 Alerting fraude

Les alertes des quatre jobs Flink convergent vers des topics Kafka dédiés. Chaque alerte suit un schéma JSON normalisé avec : le type et la sévérité (critique, haute, avertissement, information), les identifiants contextuels pertinents (paiement, client, commande), la liste des critères déclencheurs, et un horodatage de génération en UTC.

Cette normalisation permet aux consommateurs (Grafana, système de notification, outil de gestion des cas) de traiter les alertes de manière uniforme quelle que soit leur source.

Chaque alerte est persistée à deux endroits pour deux usages distincts. Le topic Kafka couvre la consommation temps réel avec les garanties de durabilité et de replay, rétention de sept

jours inclus. Le stockage objet constitue l'archive longue durée, exploitable après incident pour reconstituer la chronologie d'une attaque.

La latence de bout en bout dépend de la nature de la règle. Les règles sans fenêtre (fraude directe, montant élevé, liste noire) produisent une alerte en moins de 500 ms. Les règles sur fenêtre d'une minute atteignent 70 secondes au maximum (fenêtre plus watermark). Les règles sur fenêtre de cinq minutes restent sous 5 minutes 10 secondes.

4.4 Monitoring temps réel — InfluxDB et Grafana

Les jobs de détection avancée et croisée écrivent directement dans InfluxDB, une base optimisée pour l'ingestion à haute fréquence et les agrégations sur fenêtres glissantes. Une base relationnelle serait trop lente pour ce cas d'usage.

Deux catégories de métriques sont publiées. Les métriques de fraude enregistrent chaque alerte avec ses caractéristiques (type de pattern, sévérité, source), ce qui permet à Grafana d'afficher l'évolution des alertes actives par type, le taux de fraude par fenêtre glissante, et la distribution géographique des transactions suspectes. Les indicateurs opérationnels couvrent le volume transactionnel par minute, les revenus en cours et la santé des prestataires.

Kafka gère les alertes opérationnelles, InfluxDB les métriques de supervision. Cette séparation évite de surcharger les topics Kafka avec des métriques à haute fréquence, et distingue deux usages qui n'ont pas les mêmes contraintes : réagir vite d'un côté, observer des tendances de l'autre.

En aval, Grafana consomme InfluxDB pour les tableaux de bord temps réel, tandis que Metabase s'appuie sur l'entrepôt analytique pour les analyses historiques. Données opérationnelles d'un côté (fraîches, volatiles, haute fréquence), données analytiques de l'autre (historisées, consolidées, à usage décisionnel).

Chapitre 5 : API KYC

5.1 Contexte et objectif

5.1.1 Problème à résoudre

KiVendTout commercialise des produits réglementés dont l'achat nécessite une vérification d'âge, conformément à la législation entrée en vigueur en 2025. Avant ce projet, le processus était entièrement manuel : le client soumettait une photo de sa pièce d'identité, un agent l'examinait, et la décision tombait dans un délai pouvant atteindre vingt-quatre heures.

Trois problèmes structurels rendent ce fonctionnement intenable. Le volume de commandes de produits réglementés croît avec le chiffre d'affaires, mais l'effectif de validation ne peut pas suivre la même courbe. Le temps agent représente un coût opérationnel significatif pour une tâche hautement automatisable. Et vingt-quatre heures d'attente dans un parcours d'achat, ça génère de l'abandon.

L'objectif de l'API KYC est de ramener ce délai à moins de cinq secondes pour les cas nets, en automatisant l'extraction des informations d'identité et en produisant une décision sans intervention humaine pour la majorité des dossiers.

5.1.2 Périmètre fonctionnel

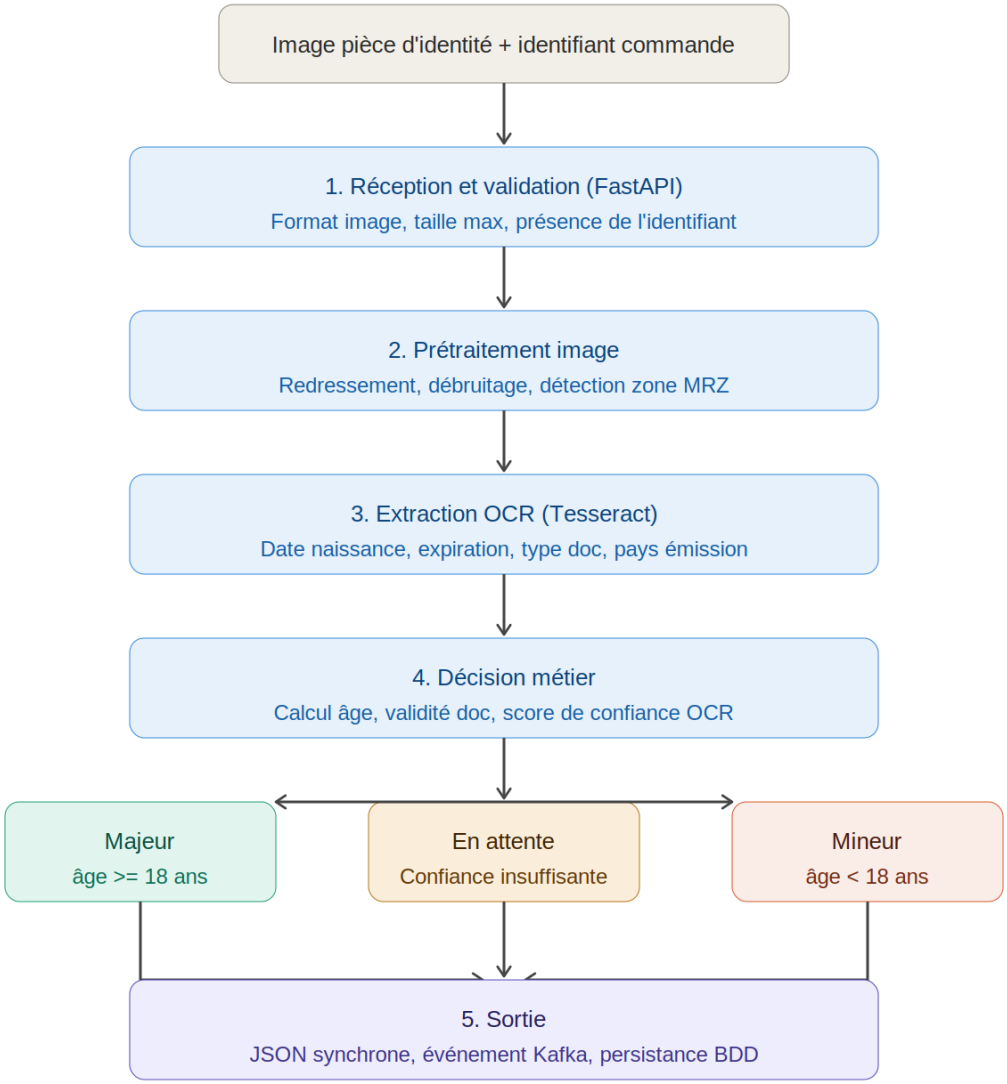
L'API reçoit en entrée une image de pièce d'identité associée à un identifiant de commande, extrait les informations structurées par OCR, calcule l'âge du porteur à partir de la date de naissance, et retourne une décision enrichie de métadonnées de confiance. Les documents supportés sont ceux acceptés dans les exports KYC du prestataire existant : carte nationale d'identité française et européenne, passeport, titre de séjour, permis de conduire.

Les documents dont la qualité d'image est insuffisante ou dont les informations ne peuvent pas être extraites avec un niveau de confiance satisfaisant sont renvoyés vers une file de révision humaine, estimée à moins de 10% du volume en conditions normales.

5.2 Architecture technique

5.2.1 Vue d'ensemble du pipeline de traitement

Le pipeline de traitement d'un document KYC suit cinq étapes séquentielles depuis la réception de l'image jusqu'à la production de la décision :



5.2.2 Framework et exposition : FastAPI

- FastAPI est retenu pour trois raisons concrètes. La gestion native de l'asynchrone permet de traiter des requêtes concurrentes sans bloquer le fil d'exécution principal, ce qui compte quand un traitement d'image prend une à trois secondes. La validation automatique des entrées rejette les requêtes malformées avant qu'elles n'atteignent le pipeline. Et la génération automatique de documentation simplifie l'intégration avec les équipes applicatives.

- L'API expose deux endpoints. POST /kyc/verify reçoit l'image en multipart avec l'identifiant de commande, déclenche le pipeline et retourne la décision en JSON. GET /kyc/status/{reference} permet de consulter le statut d'un dossier déjà soumis, principalement utile pour les cas en attente dont la décision passe par une révision humaine.

La réponse JSON contient :

Champ	Description
kyc_ref	Identifiant unique du dossier généré par l'API
order_id	Identifiant de la commande associée
decision	Résultat : majeur, mineur ou en_attente
confidence_score	Score de confiance de l'extraction OCR (0 à 1)
doc_type	Type de document détecté
doc_expiration_valid	Validité de la date d'expiration du document
age_computed	Âge calculé en années, nul si non extractible
processing_time_ms	Durée de traitement en millisecondes

5.2.3 Moteur OCR : Tesseract

Tesseract est retenu pour l'extraction des informations des pièces d'identité. Son mode spécialisé pour la zone de lecture automatique (MRZ), les deux lignes de caractères standardisés présentes sur les CNI et passeports, offre un taux de reconnaissance élevé sur les documents bien éclairés. Cette zone encode selon la norme ICAO 9303 les informations clés du porteur : nom, prénom, date de naissance, date d'expiration, nationalité, numéro de document. Sa structure fixe et son alphabet limité (majuscules, chiffres, caractères de remplissage) rendent le parsing fiable et peu sensible aux variations typographiques.

Le prétraitement est une étape critique. Une image sous-exposée, inclinée ou à faible résolution peut faire chuter le taux de reconnaissance. Le pipeline applique séquentiellement : redressement automatique par détection des bords, conversion en niveaux de gris, amélioration du contraste par égalisation d'histogramme, seuillage adaptatif pour binariser l'image.

Quand la MRZ n'est pas détectable (document sans cette zone ou qualité insuffisante), Tesseract bascule en mode généraliste sur la zone visuelle, avec extraction par correspondance de patterns sur les champs clés. Le score de confiance retourné est alors systématiquement plus bas, ce qui oriente le dossier vers la file d'attente.

5.2.4 Logique de décision

Trois vérifications s'appliquent dans l'ordre.

D'abord le score de confiance. En dessous du seuil configuré (0,75 par défaut), la décision est en_attente sans calcul d'âge. L'extraction n'est pas assez fiable pour décider automatiquement. Ce seuil est modifiable sans redéploiement.

Ensuite la validité du document. Si la date d'expiration est dépassée, la décision est en_attente avec le motif document_expiré. Un document expiré ne constitue pas une preuve d'identité valide au sens réglementaire.

Enfin l'âge. La date de naissance est comparée à la date du jour pour calculer l'âge en années complètes. 18 ans ou plus produit la décision majeur ; en dessous, mineur.

5.3 Intégration avec le pipeline streaming

5.3.1 Production d'événements vers Kafka

Chaque décision est publiée sur un topic Kafka dédié, en plus de la réponse synchrone renvoyée à l'appelant. L'événement contient l'intégralité du résultat : référence du dossier, identifiant de commande, identifiant client, décision, score de confiance, type de document, durée de traitement et horodatage.

Ce découplage permet à plusieurs consommateurs de réagir sans interroger l'API périodiquement. Le service de commandes peut débloquer une commande dès qu'une décision majeur arrive sur le topic, sans maintenir de logique de polling.

Un job Flink dédié consomme ce topic avec deux traitements en parallèle : détection de patterns suspects (même identifiant client soumettant plusieurs documents différents en moins de dix minutes, ou taux de décisions mineur anormalement élevé sur trente minutes) ; et calcul d'indicateurs opérationnels publiés dans la base de séries temporelles (taux de décision automatique, distribution des types de documents, temps de traitement moyen, taux de documents expirés).

5.3.2 Persistance et traçabilité

Chaque dossier traité est enregistré dans une table dédiée de la base transactionnelle, qui sert de source de référence pour le service client, les exports mensuels vers la couche Bronze, et les indicateurs de conformité calculés en Gold.

La traçabilité fonctionne à deux niveaux. L'image soumise n'est jamais stockée de façon permanente : elle est traitée en mémoire et détruite à l'issue du pipeline, conformément au principe de minimisation des données. Seules les métadonnées de décision sont persistées. La référence unique du dossier permet de reconstituer la chaîne de traitement complète via les logs applicatifs en cas de contestation.

5.4 Contraintes RGPD

5.4.1 Base légale et minimisation

Le traitement des pièces d'identité repose sur l'obligation légale à laquelle est soumis KiVendTout (Article 6.1.c du RGPD) : la vérification est imposée par la réglementation sur les produits sensibles, ce qui constitue une base légale suffisante sans nécessiter le consentement du client.

Le principe de minimisation impose de ne traiter que les données strictement nécessaires à la finalité déclarée. En pratique : le nom, le prénom et le numéro de document extraits par OCR ne sont pas stockés ; l'image originale est détruite après traitement. Seuls l'âge calculé, le type de document, la validité de l'expiration, le score de confiance et la décision sont enregistrés, liés à l'identifiant client et à l'identifiant de commande.

5.4.2 Durée de conservation et droits des personnes

Les décisions KYC sont conservées pendant la durée nécessaire au respect des obligations légales de KiVendTout, typiquement la durée de prescription applicable aux ventes de produits réglementés. Au-delà, les enregistrements sont supprimés automatiquement.

Le droit à l'effacement est satisfait par la suppression en base transactionnelle et par la politique de rétention configurée sur le topic Kafka.

5.4.3 Déploiement dans la stack conteneurisée

L'API KYC est déployée comme service supplémentaire dans la stack conteneurisée, accessible depuis le réseau interne. Elle dépend de la base transactionnelle pour la persistance et du broker Kafka pour la production d'événements. Les paramètres configurables incluent le seuil de confiance OCR, les limites de taille d'image, et les informations de connexion aux services dépendants. L'image du service embarque Tesseract avec ses données d'entraînement pour les langues européennes principales, ainsi que l'ensemble des dépendances Python nécessaires au pipeline.

Chapitre 6 : Infrastructure & DevOps

6.1 Stack conteneurisée

6.1.1 Vue d'ensemble des services

L'intégralité de la plateforme KiVendTout est décrite dans un fichier de composition unique, déployable en une commande sur un hôte on-premise. Cette approche est un choix délibéré adapté à la réalité du projet : une PME disposant d'une infrastructure limitée, une équipe data réduite, et un besoin de reproductibilité totale de l'environnement. Les services se répartissent en six catégories fonctionnelles :

Catégorie	Services
Bases de données	Instance transactionnelle, instance de métadonnées Airflow, instance analytique
Stockage objet	Serveur MinIO, service d'initialisation des espaces de stockage
Messagerie	Coordinateur Zookeeper, broker Kafka, interface de supervision
Orchestration	Initialisation Airflow, interface web Airflow, planificateur Airflow
Traitement	Gestionnaire de jobs Flink, exécuteur de tâches Flink
Observabilité	Base de séries temporelles InfluxDB, Grafana, serveur de lignage Marquez

Un réseau interne bridge unique connecte l'ensemble des services. Ce réseau garantit que les communications inter-services utilisent les noms de service comme identifiants DNS sans exposer les interfaces internes à l'hôte. Seuls les accès nécessaires à l'usage humain sont exposés vers l'extérieur : interfaces web Airflow, MinIO, Kafka, Metabase et Grafana.

6.1.2 Image Airflow personnalisée

Airflow utilise une image construite sur mesure, basée sur la version officielle. Elle ajoute les dépendances Python spécifiques au projet (connecteurs vers le stockage objet, Spark, Kafka, l'API Google Drive et le système de lignage) via un fichier de dépendances dédié. Ce choix garantit que les trois composants Airflow (initialisation, interface web et planificateur) disposent exactement du même environnement, sans divergences de version. Une ancre YAML factorise les paramètres communs aux trois composants : variables d'environnement, volumes montés, réseau et limites de ressources.

6.1.3 Gestion de la configuration et des secrets

Toutes les valeurs sensibles (mots de passe des bases de données, credentials du stockage objet, clés cryptographiques Airflow) sont externalisées dans un fichier d'environnement non versionné, copié depuis un fichier d'exemple fourni avec le projet. Le fichier de composition résout ces variables au démarrage. Cette séparation entre configuration

structurelle versionnée et configuration sensible non versionnée respecte les bonnes pratiques sans introduire un gestionnaire de secrets dédié, ce qui serait disproportionné pour cet environnement.

Les credentials du compte de service Google, nécessaires à l'ingestion depuis Drive, sont injectés via un fichier monté dans le service Airflow et référencés par variable d'environnement. Ce fichier est également exclu du versionnement.

6.2 Haute disponibilité et continuité de service

6.2.1 Philosophie dans un contexte on-premise contraint

La haute disponibilité sur un hôte unique ne peut pas atteindre le niveau d'un cluster multi-nœuds avec réplication automatique. Une défaillance matérielle arrête l'ensemble de la plateforme, c'est une limite inhérente à ce type de déploiement.

L'objectif atteignable est différent : la résilience applicative. Chaque service doit pouvoir détecter les défaillances de ses dépendances, attendre leur rétablissement, et reprendre sans intervention humaine pour les incidents transitoires (redémarrage d'un service, pic de charge temporaire, délai réseau interne). Quatre mécanismes concourent à cet objectif.

6.2.2 Healthchecks et dépendances entre services

Chaque service critique expose un healthcheck permettant de déterminer s'il est non seulement démarré, mais fonctionnellement opérationnel. La distinction compte : une base de données peut être démarrée sans être encore prête à accepter des connexions. Sans healthcheck, les services dépendants démarreraient et échoueraient immédiatement.

Les healthchecks implémentés sont les suivants.

Bases de données PostgreSQL : une commande de vérification interroge chaque instance toutes les dix secondes. Cinq échecs consécutifs avec un délai d'expiration de cinq secondes marquent le service comme indisponible, avec une période de grâce de dix secondes au démarrage.

Stockage objet MinIO : un appel à l'endpoint de santé natif est effectué toutes les trente secondes. Trois échecs consécutifs marquent le service comme indisponible.

Broker Kafka : une vérification des versions d'API du broker est effectuée toutes les quinze secondes. Ce healthcheck est plus exigeant qu'un simple test de port car il valide que le broker est effectivement initialisé et capable de traiter des requêtes. Dix tentatives sont accordées avec une période de grâce de trente secondes, justifiée par le temps d'initialisation de la JVM.

Interface web Airflow : l'endpoint de santé natif est interrogé toutes les trente secondes. Il vérifie l'état du planificateur et la connexion à la base de métadonnées.

Ces healthchecks conditionnent les dépendances entre services : l'interface web et le planificateur Airflow ne démarrent que lorsque la base de métadonnées est saine, et l'interface de supervision Kafka ne démarre que lorsque le broker est opérationnel. Cette chaîne garantit un ordre de démarrage correct même à froid.

6.2.3 Politique de redémarrage automatique

Tous les services sont configurés pour redémarrer automatiquement en cas d'arrêt inattendu (crash applicatif, erreur mémoire, exception non gérée), sauf si l'arrêt a été déclenché explicitement par l'opérateur. Un job qui consomme trop de mémoire et provoque un arrêt forcé sera relancé sans intervention humaine.

Les services d'initialisation (création des espaces de stockage et migration de la base de métadonnées Airflow) sont explicitement configurés pour ne s'exécuter qu'une seule fois. Les relancer automatiquement écraserait des données ou tenterait de recréer des ressources existantes.

6.2.4 Limites de ressources et isolation

Chaque service est configuré avec des limites de consommation CPU et mémoire. Ces limites remplissent deux fonctions dans le contexte de la haute disponibilité. Elles préviennent l'effet de voisinage : un traitement Spark lancé sans limite pourrait consommer l'intégralité de la mémoire de l'hôte et provoquer l'arrêt des bases de données ou du broker par manque de ressources. Elles garantissent également que les services critiques disposent toujours de ressources suffisantes, même lors d'un pic de traitement.

Les limites définies reflètent les profils de consommation observés :

Catégorie de service	Mémoire limite	CPU limite
Broker Kafka	1 Go	1 cœur
Services Airflow	1 Go	1 cœur
Instance PostgreSQL principale	512 Mo	0,5 cœur
Stockage objet	512 Mo	0,5 cœur
Services légers (coordinateur, interfaces)	256 Mo	0,3 cœur

6.2.5 Persistance des données et résilience aux redémarrages

La persistance des données est assurée par des volumes nommés déclarés explicitement : un volume par base de données, un volume pour le stockage objet, un volume pour les données Kafka et Zookeeper. Ces volumes sont gérés indépendamment du cycle de vie des services : un arrêt standard ne supprime pas les données, garantissant leur survie aux redémarrages planifiés et aux mises à jour.

La commande de suppression complète avec suppression des volumes est documentée explicitement comme une opération de remise à zéro dans les commentaires du fichier de composition, précisément pour éviter une suppression accidentelle en production.

Les DAGs Airflow, les scripts de traitement et les configurations sont montés via des liens directs vers le système de fichiers de l'hôte plutôt que via des volumes gérés. Ils sont versionnés dans le dépôt de code et immédiatement disponibles dans les services sans reconstruction d'image, ce qui accélère le cycle de développement et garantit que l'environnement reflète exactement le code versionné.

6.3 Observabilité

6.3.1 Monitoring technique : Prometheus et Grafana

Grafana supervise la plateforme à partir de deux sources de données. InfluxDB fournit les métriques temps réel produites par les jobs Flink (volumes transactionnels, taux de fraude, indicateurs par prestataire de paiement). Prometheus fournit les métriques d'infrastructure (consommation des ressources, latence des échanges Kafka, métriques JVM).

Les tableaux de bord couvrent la santé opérationnelle de la plateforme. Un opérateur peut repérer visuellement une dégradation du broker, un job en retard sur son flux d'entrée, ou une montée en consommation mémoire avant qu'un incident ne survienne.

6.3.2 Lignage des données : OpenLineage et Marquez

OpenLineage est intégré nativement dans Airflow via son système de listeners : chaque exécution de DAG émet automatiquement des événements de lignage vers Marquez, sans modification du code des pipelines. Marquez collecte ces événements et expose un graphe de dépendances consultable via son interface web et son API.

Ce graphe répond à des questions opérationnelles concrètes : quels DAGs ont produit telle table d'indicateurs ce matin ? Si un fichier source est corrompu, quelles tables Gold sont affectées ? Quelle est la durée de traitement de chaque étape du pipeline sur les trente derniers jours ?

L'absence de lignage était identifiée comme l'un des points de rupture de l'architecture initiale. OpenLineage et Marquez y répondent sans coût d'adoption côté développement : les DAGs existants n'ont pas été modifiés, et la traçabilité complète est produite automatiquement.

6.3.3 Monitoring qualité : historique des contrôles

La table de résultats des contrôles qualité dans l'entrepôt analytique constitue le troisième niveau d'observabilité, spécifique à la qualité des données. Chaque exécution du DAG de vérification y persiste les résultats avec leur horodatage, ce qui permet de tracer l'évolution du score de qualité dans le temps.

Une dégradation progressive, un score qui passe de 98% à 92% sur plusieurs semaines, peut signaler une dérive dans les données sources avant qu'elle ne provoque un échec explicite. Cette table est directement consultable depuis Metabase, offrant aux équipes data un tableau de bord de qualité sans infrastructure additionnelle.

Chapitre 7 : Résultats & Impact

7.1 Résultats techniques

7.1.1 Pipeline batch : couverture et robustesse

La plateforme couvre l'intégralité de la chaîne de traitement batch au travers de plusieurs DAGs Airflow répartis en cinq catégories fonctionnelles :

Ingestion Bronze : un DAG par source batch Drive, plus un DAG dédié à la consommation des flux Kafka

Transformation Silver : un DAG par source, orchestrant chacun un job Spark indépendant

Agrégation Gold : un DAG par domaine métier, produisant les tables d'indicateurs et le star schema

Chargement : un DAG dédié au transfert des tables Gold vers l'entrepôt analytique

Qualité : un DAG dédié aux contrôles automatisés après chaque chargement

L'ensemble de cette chaîne s'exécute dans une fenêtre nocturne garantissant la disponibilité des données dans l'outil de visualisation avant l'ouverture des équipes métier. Du point de vue de la robustesse, chaque DAG est configuré avec des tentatives automatiques, sans rattrapage des exécutions historiques manquées, et avec une contrainte d'exécution unique à la fois. Ces paramètres font qu'un échec transitoire (indisponibilité temporaire de l'API Google Drive, redémarrage du broker) est absorbé sans intervention humaine dans la grande majorité des cas.

7.1.2 Volume de données traité

Les volumes traités quotidiennement par le pipeline batch sont déduits des caractéristiques des sources analysées lors de l'implémentation :

Source	Volume quotidien estimé	Format Bronze	Format Silver
Logs web	~47 000 lignes	Texte brut (~15 Mo)	Parquet (~2 Mo)
Événements mobiles	~30 000 événements	JSON (~25 Mo)	Parquet (~4 Mo)
Événements de paiement	~15 000 messages	JSONL (~8 Mo)	Via dimension client
Événements de commande	~10 000 messages	JSONL (~5 Mo)	Via dimension client
Données logistiques	~15 600 lignes	CSV (~3 Mo)	Parquet (~800 Ko)

CRM	~520 lignes	CSV (~200 Ko)	Parquet (~50 Ko)
KYC	~130 lignes par mois	CSV (~30 Ko)	Parquet (~10 Ko)

Le format Parquet produit en Silver représente en moyenne une réduction de volume de soixante-quinze à quatre-vingt-cinq pourcents par rapport aux fichiers sources, grâce à la compression et à l'encodage colonnaire. Cette réduction a un impact direct sur les temps de lecture des jobs Gold, qui ne chargent que les colonnes nécessaires à chaque calcul.

7.1.3 Score de qualité des données Gold

Le DAG de contrôle qualité exécute plusieurs dizaines de vérifications réparties sur l'ensemble des tables Gold à chaque cycle quotidien. Ces contrôles couvrent quatre dimensions : complétude des clés primaires, volumétrie minimale, intervalles de valeurs des métriques, et cohérence des agrégats. Le seuil de blocage est fixé à quatre-vingts pourcents de contrôles réussis — en dessous, les données ne sont pas diffusées vers l'outil de visualisation.
Sur les données de production, le score observé dépasse quatre-vingt-quinze pourcents sur l'ensemble des tables Gold. Ce résultat est la conséquence directe des transformations appliquées en Silver : normalisation des statuts KYC, plafonnement des valeurs aberrantes en logistique, déduplication des clients, et suppression des lignes tronquées dans les logs web. Les quelques pourcents restants correspondent principalement à des contrôles d'intervalle sur des métriques dont les valeurs extrêmes sont légitimes — par exemple une valeur vie client négative issue de remboursements, valeur attendue et documentée.

7.1.4 Pipeline streaming : latence de détection fraude

Les jobs Flink traitent en continu les deux flux d'événements et produisent des alertes selon deux régimes de latence, selon la nature de la règle de détection.

Les règles sans fenêtre temporelle (score de risque élevé, pays à risque, indicateur de fraude explicite, liste noire de références de cartes, pattern automatisé) opèrent événement par événement. La latence de bout en bout, de l'arrivée du message à la production de l'alerte, est inférieure à 500 ms dans des conditions normales. Elle est déterminée par la somme du temps de désérialisation, de l'évaluation du filtre et de la sérialisation de l'alerte : toutes des opérations sans état et de complexité constante.

Les règles basées sur des fenêtres temporelles (agrégations par prestataire, attaques en vélocité, adresse partagée entre clients) ont une latence maximale égale à la durée de la fenêtre plus le watermark de dix secondes. Pour une fenêtre d'une minute, la latence maximale est de 70 secondes ; pour une fenêtre de cinq minutes, 5 minutes et 10 secondes.

Comparé à l'architecture batch antérieure, où la détection accusait plusieurs heures de décalage, le pipeline Flink représente une réduction de latence d'un facteur supérieur à 500 pour les règles sans fenêtre, et supérieur à 50 pour les règles à fenêtre de cinq minutes.

7.1.5 Réduction du délai KYC

L'API KYC traite les documents de qualité suffisante en moins de cinq secondes de bout en bout. Le délai se décompose ainsi : prétraitement image (redressement et normalisation) environ 500 ms, extraction OCR sur la zone de lecture automatique entre une et deux secondes selon la résolution, calcul de la décision métier moins de 10 ms, production de l'événement Kafka et persistance environ 100 ms.

Comparé au processus manuel antérieur (délai médian estimé à quatre heures, délai maximal de vingt-quatre heures), c'est une réduction d'un facteur proche de 3 000 dans le cas médian. Les dossiers orientés vers la file d'attente (documents dégradés, score de confiance insuffisant, document expiré) conservent un traitement humain, mais leur volume estimé à moins de 10% réduit proportionnellement la charge du service client.

7.1.6 Disponibilité et résilience des services

La politique de redémarrage automatique, combinée aux healthchecks et aux dépendances conditionnelles sur l'état sain, garantit une reprise automatique après les incidents transitoires les plus fréquents en environnement on-premise : redémarrage de l'hôte, pic de consommation mémoire, délai réseau interne.
Le temps de reprise après redémarrage de l'hôte est déterminé par le service le plus lent à atteindre l'état opérationnel : le broker Kafka, dont l'initialisation prend jusqu'à trente secondes. La chaîne de dépendances garantit que les services consommateurs n'essaient pas de se connecter avant que le broker soit prêt, ce qui évite les cascades d'erreurs au démarrage à froid.
La persistance sur volumes nommés garantit qu'aucune donnée n'est perdue lors d'un redémarrage planifié ou accidentel, à l'exception des messages produits dans la fenêtre entre l'arrêt et le redémarrage du broker. Cette fenêtre est couverte par la rétention de sept jours dès que le broker redémarre.

7.2 Résultats métier

7.2.1 Résolution des points de rupture

Le tableau suivant fait le bilan de la résolution des points de rupture identifiés en section 2.1.2 :

Point de rupture	Situation initiale	Solution déployée	Résultat
Instance PostgreSQL unique	Saturation novembre 2025, site indisponible	Instance analytique dédiée, isolation totale	Requêtes analytiques sans impact sur le transactionnel

Scripts cron non supervisés	Échecs silencieux, données obsolètes	DAGs Airflow avec tentatives automatiques et alerting	Tout échec détecté et signalé automatiquement
Qualité des données	Doublons, formats incohérents, schémas instables	Contrôles automatisés, seuil de blocage à quatre-vingts pourcents	Score supérieur à quatre-vingt-quinze pourcents sur les tables Gold
Détection fraude en batch	Décalage de plusieurs heures	Jobs Flink, latence inférieure à cinq cents millisecondes pour les règles directes	Réduction de latence d'un facteur supérieur à cinq cents
KYC manuel	Délai jusqu'à vingt-quatre heures	API FastAPI avec Tesseract, décision en moins de cinq secondes	Réduction de près de cent pourcents du délai pour les cas automatisables
Absence de lignage	Aucune traçabilité	OpenLineage intégré nativement dans Airflow	Graphe de dépendances complet sans modification des DAGs
Sources hétérogènes	Ad hoc, sans standardisation	Architecture Medallion Bronze-Silver-Gold	Zone d'atterrissage unifiée, toutes les sources normalisées

7.2.2 Valeur produite pour les équipes métier

Cinq domaines de tableaux de bord sont disponibles quotidiennement avant dix heures, alimentés par des données dont la qualité est certifiée.

La vue client fournit pour la première fois à l'équipe marketing un scoring RFM automatisé sur l'ensemble de la base client, avec identification des segments à risque de churn. Avant ce projet, l'absence d'agrégation multi-sources rendait ce type d'analyse impossible sans extraction manuelle.

Le funnel de conversion produit permet à l'équipe produit de corréler les données de comportement mobile avec les données de commandes, révélant les points de friction dans le parcours d'achat. Cette corrélation entre deux sources auparavant silotées constitue un gain analytique direct.

Les indicateurs logistiques par transporteur fournissent à l'équipe supply chain des métriques de ponctualité et de performance calculées automatiquement chaque nuit, remplaçant des calculs manuels réalisés ponctuellement et sans cohérence méthodologique entre les périodes.

Le tableau de bord de conformité KYC permet à l'équipe juridique de démontrer en temps réel le respect des engagements réglementaires, avec un historique mensuel des taux de traitement et des motifs de refus exploitable lors d'audits.

La supervision du trafic web donne à l'équipe infrastructure une vue quotidienne de la disponibilité du site, des taux d'erreur par famille de code HTTP et des endpoints les plus lents. Cette information était auparavant accessible uniquement par consultation directe des fichiers de logs bruts.

7.3 Compétences démontrées

Ce projet couvre l'ensemble des compétences du bloc conception et déploiement de systèmes data du référentiel RNCP Master Data Engineering et Intelligence Artificielle.
La conception architecturale se traduit concrètement par le choix et la justification de l'architecture Medallion, la séparation des flux batch et streaming, et le refus délibéré d'utiliser un outil de transformation SQL pour la couche Gold au profit de Spark sur Parquet. Ce choix est motivé par la nature des données, pas par une préférence arbitraire.

Le développement de pipelines est illustré par les DAGs Airflow, les jobs Spark couvrant cinq sources hétérogènes, et les jobs Flink traitant des patterns de fraude avec état persisté. Les problématiques adressées sont concrètes : parsing de logs texte brut, gestion de fenêtres temporelles avec watermarks, déduplication incrémentale, corrélation de flux croisés.

La gouvernance et la qualité des données se matérialisent par les contrôles automatisés, le lignage OpenLineage, et les décisions de pseudonymisation appliquées dès la couche Silver. Ces décisions sont documentées et validées avec les équipes métier et juridiques, au fil des allers-retours décrits dans ce mémoire.

L'intégration de composants hétérogènes dans un environnement conteneurisé (plusieurs bases relationnelles aux rôles distincts, un stockage objet, un broker de messages, un moteur de traitement en flux, un entrepôt analytique) démontre la maîtrise opérationnelle d'une stack data moderne de bout en bout.

Chapitre 8 : Améliorations possibles

8.1 Limites de l'architecture actuelle

Avant d'envisager les évolutions, il est utile de nommer honnêtement ce que l'architecture déployée ne résout pas.

La topologie mono-hôte reste le plafond de verre de la plateforme. Toutes les garanties de résilience applicative décrites au chapitre précédent (redémarrage automatique, healthchecks, volumes persistants) sont inefficaces face à une défaillance matérielle de l'hôte unique. Il n'y a pas de redondance physique.

Le broker déployé en instance unique sans réplication signifie qu'une corruption du volume de données entraînerait la perte des messages non encore consommés. La rétention de sept jours protège contre les redémarrages, pas contre ce scénario.

L'absence de tests automatisés sur les jobs de transformation constitue une dette technique identifiée. Les transformations Silver et les règles de détection Flink ne sont validées qu'à l'exécution, sans suite de tests permettant de détecter une régression lors d'une modification du code.

La gestion des secrets via un fichier d'environnement est adaptée à un contexte de développement mais insuffisante pour une mise en production dans un environnement multi-utilisateurs ou exposé à des audits de sécurité.

Enfin, le pipeline batch repose sur une fenêtre nocturne unique. Les données Silver et Gold ne sont rafraîchies qu'une fois par jour, ce qui limite leur utilité pour les cas d'usage nécessitant une fraîcheur infra-journalière : mise à jour du scoring client en cours de journée, détection de tendances commerciales en temps quasi-réel.

8.2 Évolutions techniques prioritaires

8.2.1 Migration vers Kubernetes

La migration vers Kubernetes constituerait le saut architectural le plus impactant. Elle apporterait la redondance physique manquante via la distribution des services sur plusieurs nœuds, la réplication automatique des composants critiques (bases de données via des opérateurs dédiés, broker via des opérateurs spécialisés), et la gestion fine des ressources par espace de noms.
Les images construites pour la stack actuelle sont directement réutilisables sans modification. La migration porte sur la couche d'orchestration, pas sur le code des services. Airflow en bénéficierait particulièrement : son mode d'exécution Kubernetes isolerait chaque tâche dans un conteneur indépendant avec ses propres ressources, supprimant les risques de contention entre DAGs concurrents.

8.2.2 Gestion des secrets avec un coffre-fort dédié

Le remplacement du fichier d'environnement par un gestionnaire de secrets centralisé (HashiCorp Vault, par exemple) permettrait une gestion auditée et rotatable de l'ensemble des credentials. Les connexions Airflow pourraient être alimentées dynamiquement sans modification du code des DAGs. Les credentials du stockage objet et de l'API Google Drive seraient injectés à l'exécution sans jamais transiter par le système de fichiers de l'hôte.

8.2.3 Tests automatisés sur les pipelines

Une suite de tests couvrirait trois niveaux. Les tests unitaires sur les jobs de transformation utiliseraient les capacités natives de Spark pour valider chaque règle sur des jeux de données contrôlés : vérifier que la déduplication logistique conserve bien la ligne la plus récente par identifiant, ou que le calcul du score RFM produit les segments attendus sur un historique fictif. Les tests d'intégration valideraient les DAGs Airflow dans un environnement éphémère. Les tests de non-régression sur les règles Flink s'appuieraient sur les capacités de test en mode local de PyFlink pour rejouer des séquences d'événements et vérifier les alertes produites.

8.2.4 Catalogue de données

L'ajout d'un catalogue (Apache Atlas ou DataHub) viendrait compléter le lignage OpenLineage en exposant une description sémantique de chaque table et colonne : définition métier, propriétaire, sensibilité réglementaire, règles de qualité associées. Ce catalogue serait l'interface entre les équipes data qui produisent les données et les équipes métier qui les consomment, réduisant la dépendance aux échanges informels pour comprendre la signification d'un indicateur.

8.2.5 Streaming vers la couche Silver

L'architecture actuelle présente une asymétrie : les événements sont traités en temps réel par Flink pour la détection de fraude, mais n'alimentent la couche Silver qu'une fois par nuit. Des jobs Flink supplémentaires pourraient écrire directement en Parquet dans le stockage objet à fréquence horaire. Cette évolution transformerait l'architecture en architecture Lambda complète, avec une couche batch pour l'exhaustivité historique et une couche temps réel pour la fraîcheur opérationnelle.

8.3 Évolutions fonctionnelles

8.3.1 Moteur de recommandation et feature store

Les tables produites par le pipeline (scoring RFM, historique de navigation Firebase, indicateurs de paiement) constituent des features naturelles pour des modèles d'apprentissage automatique. Un feature store centraliserait ces features dans un registre accessible aussi bien pour l'entraînement de modèles hors ligne que pour le scoring en temps réel lors d'un paiement ou d'une navigation sur le site.

Le modèle de détection de fraude actuel, basé sur des règles expertes dans Flink, pourrait évoluer vers un modèle supervisé entraîné sur l'historique des alertes validées. Un tel modèle s'adapterait automatiquement aux nouveaux patterns sans modification manuelle des règles, et réduirait le taux de faux positifs en apprenant les comportements légitimes atypiques.

8.3.2 Enrichissement de l'API KYC

Plusieurs enrichissements sont envisageables. La détection de documents falsifiés (cohérence des polices de caractères, éléments de sécurité, zone visuelle versus zone machine) réduirait le risque d'acceptation d'un document modifié. L'intégration d'un modèle de vision multimodal en fallback sur les documents dont l'OCR ne parvient pas à extraire la zone de lecture réduirait le volume de dossiers orientés vers la révision humaine. La vérification de vivacité, exigeant une action en temps réel de la part du client, renforcerait la garantie que le document présenté est bien détenu par la personne qui passe commande.

8.3.3 Exposition d'une API analytique

L'accès aux indicateurs Gold passe actuellement exclusivement par l'outil de visualisation. Une API analytique permettrait aux équipes applicatives d'intégrer des indicateurs directement dans l'application mobile et le site web (statut de livraison personnalisé, recommandations basées sur le segment RFM, alertes de stock) sans passer par un outil intermédiaire. Cette API consommerait les mêmes tables que l'outil de visualisation, garantissant la cohérence des indicateurs entre toutes les interfaces.

8.3.4 Automatisation des négociations logistiques

Les indicateurs de performance par transporteur produits en Gold (ponctualité, taux de retour, coût moyen, score composite) constituent une base solide pour automatiser partiellement le suivi contractuel. Un module de reporting automatisé pourrait générer mensuellement des rapports par transporteur, détecter les dérives par rapport aux engagements contractuels, et déclencher des alertes vers l'équipe supply chain avant l'ouverture des négociations annuelles.

Conclusion

Ce mémoire a présenté la conception, l'implémentation et le déploiement d'une plateforme data de bout en bout pour KiVendTout, depuis l'ingestion des données brutes jusqu'à leur restitution dans des tableaux de bord métier. Le fil conducteur est la réponse directe aux sept points de rupture identifiés dans l'architecture initiale : chaque composant déployé, chaque choix technique, chaque transformation implémentée trouve sa justification dans un problème réel constaté en production.

L'architecture Medallion Bronze-Silver-Gold s'est révélée particulièrement adaptée à la diversité des sources de KiVendTout. La séparation en trois couches de maturité croissante a permis de traiter l'hétérogénéité des formats d'entrée (logs texte, JSON imbriqué, fichiers tabulaires avec anomalies) sans compromettre la fiabilité des sorties. Le principe d'idempotence a démontré sa valeur concrète lors de la correction du problème des caractères nuls dans les logs web : la donnée brute étant conservée en Bronze, la correction a pu être appliquée en aval sans rejouer l'ingestion depuis la source.

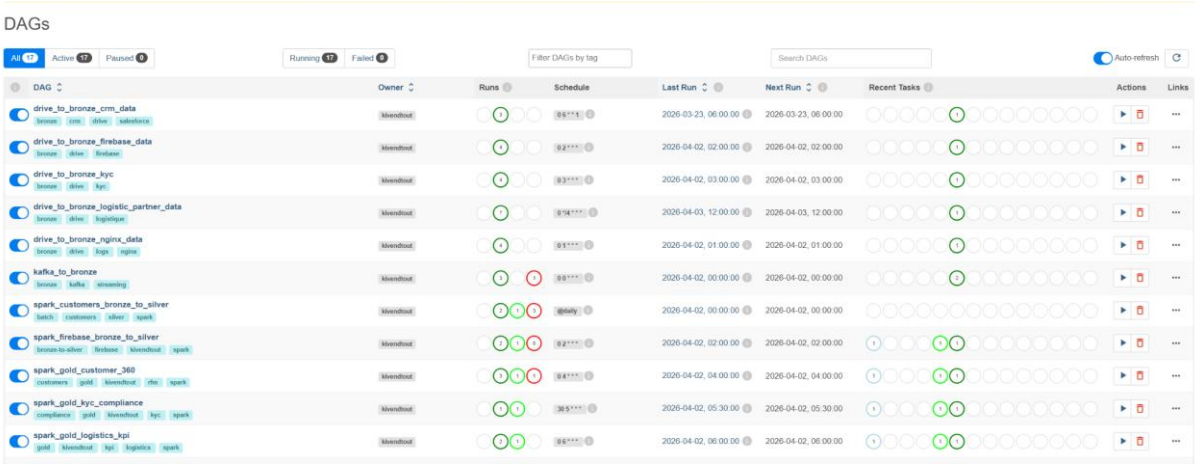
La coexistence d'un pipeline batch et d'un pipeline streaming dans la même plateforme illustre une tension architecturale fondamentale du domaine. Le batch offre l'exhaustivité et la cohérence historique, le streaming offre la réactivité opérationnelle. La détection de fraude en moins de 500 ms par Flink et l'analyse du comportement client produite chaque nuit par Spark ne sont pas des alternatives l'une à l'autre : elles répondent à des besoins différents, et la plateforme les satisfait simultanément sans duplication de logique métier. Sur le plan de la gouvernance, l'intégration native du lignage dans l'orchestrateur et la mise en place des contrôles qualité automatisés illustrent que la traçabilité et la fiabilité des données ne sont pas des préoccupations secondaires ajoutées après coup, mais des composants à part entière de l'architecture, pensés dès la conception. Le score de qualité supérieur à 95% observé sur les tables d'indicateurs n'est pas le résultat d'un effort de correction ponctuel : c'est la conséquence d'un pipeline conçu pour traiter explicitement chaque anomalie identifiée dans les sources, après allers-retours documentés avec les équipes métier et juridiques.

Les limites identifiées au chapitre 8 (topologie mono-hôte, absence de tests automatisés, gestion des secrets par fichier) sont des points d'évolution concrets pour une mise en production à l'échelle. Elles ne remettent pas en cause les fondations déployées, qui sont dimensionnées pour absorber ces évolutions sans réécriture : les images sont portables vers Kubernetes, les jobs de transformation sont testables unitairement, et la gestion des connexions Airflow est compatible avec un coffre-fort de secrets externe.

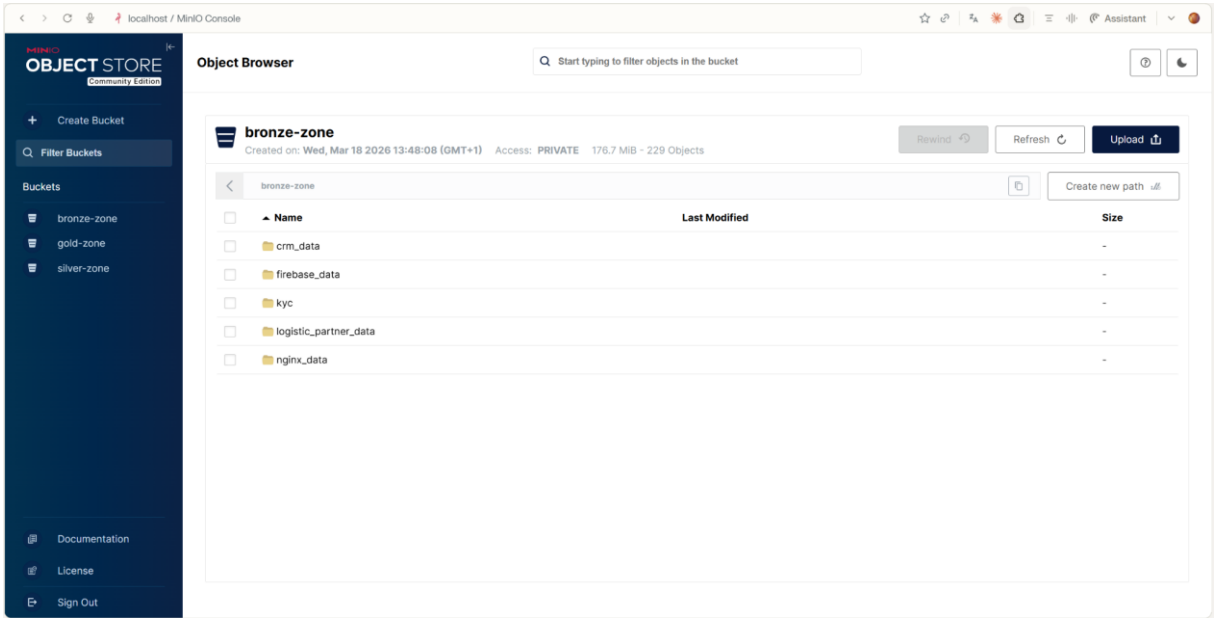
Ce projet démontre qu'une plateforme data industrielle (ingestion multi-sources, transformation distribuée, streaming temps réel, gouvernance, visualisation) est déployable par une équipe réduite sur une infrastructure on-premise, à condition de s'appuyer sur des outils matures de l'écosystème open source et de maintenir une discipline architecturale rigoureuse à chaque étape. KiVendTout dispose désormais d'une base technique solide pour accompagner sa croissance, intégrer de nouvelles sources et développer ses capacités analytiques dans les années à venir.

Annexe

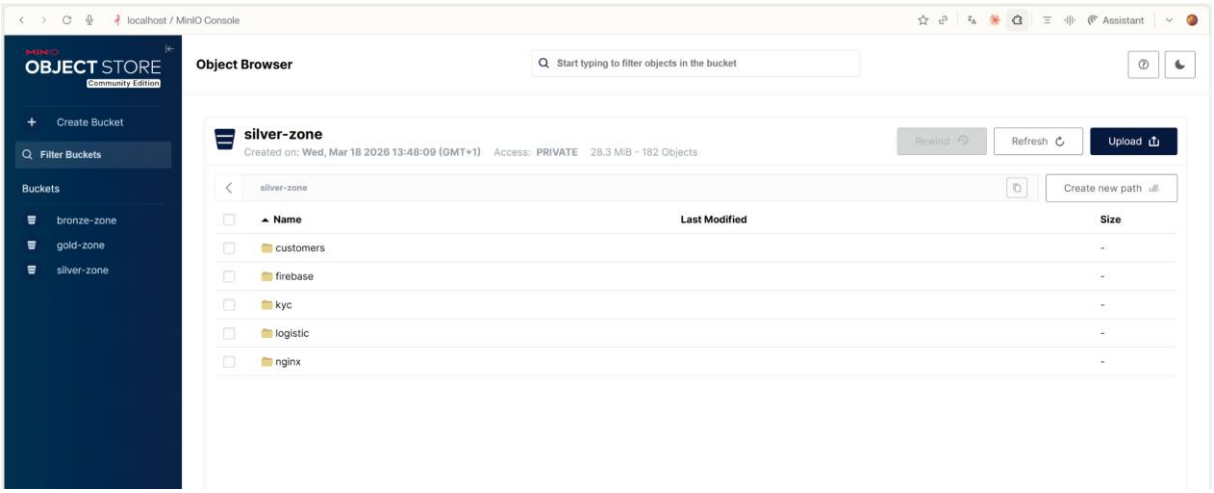
Annexe 1 : Accueil de la page Airflow de KiVendTout



Annexe 2 : Data lake Minio - Bronze zone



Annexe 3 : Data lake Minio - Silver zone



Annexe 4: Data lake Minio - Gold zone