1. Introduction

1.1. Objectifs du projet

Le secteur de l'aviation connaît une transformation rapide grâce à la digitalisation et à l'avancement des technologies du big data. L'un des enjeux majeurs de cette évolution réside dans la collecte, l'analyse et l'exploitation efficace d'un large volume de données liées aux vols. Dans ce contexte, le projet proposé vise à mettre en place pour le client aéroport de Paris une infrastructure permettant la collecte, la transformation et le chargement de données d'aviation, afin de rendre ces informations plus accessibles et plus utiles pour les utilisateurs finaux.

Cette infrastructure permet de recueillir des données en temps réel provenant de multiples sources, notamment les systèmes de suivi des vols, les données des compagnies aériennes, les informations de trafic aérien, et bien plus encore. Ces données sont ensuite transformées et visualisées pour offrir une vue d'ensemble des opérations aériennes, avec des informations sur les trajets, les horaires, les altitudes, les vitesses et les durées des vols. Le système fournit des tableaux de bord dynamiques et des cartes interactives permettant aux utilisateurs de prendre des décisions éclairées et d'optimiser les opérations.

Initialement destiné aux intervenants des opérations aéroportuaires et du contrôle aérien, ce système pourra également intéresser les autres acteurs du secteur de l'aviation civile :

- Opérateurs du transport aérien
- Voyageurs

1.2. Compétences démontrées

- Gestion de projet de données de bout en bout avec méthodologies agiles
- Utilisation d'une approche DevOps
- Collecte, transformation et stockage des données
- Visualisation des données stockées dans un tableau de bord
- Monitoring et prédiction de performances de l'infrastructure à l'aide de modèles de machine learning

2. Problématisation et contexte du projet

Les données relatives aux vols aériens sont souvent dispersées dans plusieurs sources, difficiles à exploiter en temps réel, et les outils existants de visualisation ne permettent pas une analyse rapide et précise de la performance des vols. La problématique que ce projet tente de résoudre réside dans la gestion complexe et souvent disparate des données relatives à l'aviation. La multiplication des sources de données et l'absence d'une vue consolidée rendent difficile l'analyse efficace et la prise de décision rapide. Ainsi, la question à laquelle nous répondons est :

Comment offrir une infrastructure fluide, performante, économique et autonome permettant une collecte, une transformation et une visualisation optimisée des données d'aviation, au service de l'ensemble des acteurs du secteur aérien civil pour améliorer la gestion des vols, la sécurité et l'expérience passager ?

3. Proposition de solution

3.1. Technologies utilisées

Les technologies que nous avons choisies pour répondre à cette problématique sont :

- Talend pour l'ingestion des données et la préparation de leur envoi vers un data lake
- Zookeeper, Kafka et Python pour la collecte et le transport de données de scraping vers le data lake
- API d'aviation et site internet pour la collecte des données relatives aux vols et compagnies aériennes
- Localstack en mode développement puis AWS S3 en mode production pour la création de buckets S3 où sont déposées les données brutes ingérées et collectées
- Apache Spark et PySpark pour le téléchargement, la transformation et l'indexation des données vers un moteur de recherche et d'analyse distribué
- Elasticsearch pour le stockage des données indexées et recherches avancées en temps réel
- Kibana pour la visualisation, l'analyse et l'interaction intuitive avec les données indexées dans un dashboard dynamique
- Docker pour l'indépendance des services et un déploiement simplifié de l'infrastructure
- Linux pour le fonctionnement des services en mode conteneurisé offrant une plateforme légère pour l'infrastructure
- Windows pour l'exécution de Talend et l'envoi des données brutes vers Localstack sous Linux

- Curl sous Windows Powershell pour tester la connectivité et la **disponibilité du service Localstack avec requête http**
- Prometheus et Scikit-Learn pour collecte des données de fonctionnement de l'infrastructure et la prédiction de ses performances
- Apache Airflow pour automatisation des tâches de l'infrastructure et lancement d'actions si notification d'alertes issues du modèle de machine learning
- Elastic Kubernetes Service (EKS) pour orchestration des services en environnement de production cloud-native avec AWS et fonctions RGPD avec KMS/IAM/Secrets Manager
- Github Actions pour la création de workflows automatisés CI/CD entre les environnements de développement et de production

3.2. Architecture de la solution

L'architecture de la solution se décompose en plusieurs parties :

1. Premier pipeline de récupération en temps réel des données de l'API aviationstack
2. Deuxième pipeline de scraping des données de l'API Opensky et du site internet Skytrax
3. Stockage des données brutes dans un data lake S3 sous localstack (ou AWS) récupérant le contenu des pipelines au format JSON
4. Transformation des données : Spark et PySpark sont utilisés pour télécharger les données du data lake, les nettoyer, les transformer et les enrichir puis les indexer dans un data warehouse
5. Stockage des données traitées dans le data warehouse avec Elasticsearch pour un accès rapide
6. Visualisation de ces données dans un dashboard interactif avec Kibana
7. Collecte des métriques de l'infrastructure avec Prometheus puis analyse avec Scikit-Learn pour anticiper l'apparition de goulots d'étranglement dans les flux
8. Automatisation et surveillance des tâches de l'infrastructure avec Airflow en réalisant les étapes selon horaires et déclencheurs définis et effectuant des opérations de contrôle pouvant émettre une alerte suivie d'actions
9. Création d'un cluster EKS sur AWS pour déploiement des services précités via manifests (le job Talend sous Windows est maintenu hors cluster et déclenché depuis la CI/CD via un Github Actions Self-Hosted Runner Windows) et création micro services RGPD

La construction des pipelines CI/CD d'intégration continue et de déploiement continu avec Github Actions s'effectue en parallèle du suivi de ces étapes.

Un diagramme de l'architecture complète en mode DevOps est consultable en annexe A.

4. Cahier des charges, rétroplanning et budget prévisionnel

4.1. Cahier des charges

Le projet devra remplir les objectifs suivants:

- Extraire toutes les 8 heures les données brutes pour chacun des deux pipelines
- Créer un data lake pour leur stockage en attendant leur traitement
- Nettoyer, transformer et enrichir les données brutes
- Indexer les données préparées dans un data warehouse adapté pour l'analyse et la visualisation
- Créer une visualisation interactive et analytique des données dans un tableau de bord dynamique
- Mettre en place un monitoring prédictif de l'infrastructure pour anticiper et régler les problèmes
- Automatiser avec DevOps les workflows et gérer les erreurs ou alertes pour un fonctionnement fiable de l'infrastructure avec sécurisation des données et conformité au RGPD
- Appliquer une démarche socio-organisationnelle et humaine visant à sécuriser l'adoption de la nouvelle infrastructure et de ses outils par les utilisateurs finaux

Les contraintes incluent:

- Atteindre les objectifs du projet dans un délai de 18 semaines.
- Travailler dans un environnement hybride Windows/Linux, ce qui nécessite la compatibilité entre le composant Talend fonctionnant nativement sous Windows et le reste de l'infrastructure opérant sous Linux au moyen d'interfaces comme cURL ou des services réseaux
- Conteneuriser via Docker la partie de l'architecture fonctionnant sous Linux impliquant une gestion des services dans un environnement isolé, tout en garantissant la compatibilité des configurations entre les différents conteneurs.
- Choisir dynamiquement pour le data lake Localstack en Dev ou AWS S3 en Ops selon la phase du workflow CI/CD

4.2. Rétroplanning

- Phase 1 : Analyse des besoins et rédaction du cahier des charges 5 jours
- Phase 2 : Configuration initiale des environnements Windows et Linux 5 jours
- Phase 3 : Création du pipeline d'ingestion Aviationstack 4 jours
- Phase 4 : Création du pipeline de scraping Opensky/Skytrax et du data lake S3 6 jours

- Phase 5 : Transformation des données brutes et création du data warehouse 6 jours
- Phase 6 : Création du tableau de bord interactif de visualisation 14 jours
- Phase 7 : Développement du système d'analyse prédictive des performances 11 jours
- Phase 8 : Développement de la gestion automatisée des workflows et notifications d'alertes 11 jours
- Phase 9 : Construction de la partie cloud (Ops) de l'infrastructure 14 jours
- Phase 10 : Tests et déploiement final pour mise en production 7 jours
- Phase 11 : Accompagnement des utilisateurs finaux pour sa prise en main 12 jours

Soit 95 jours ouvrables pour une durée calendaire de 18 semaines ou 4 mois.

4.3. Budget prévisionnel

Poste	Coût estimé (Eur)
Ressources humaines	145 350 Euros
Outils et logiciels	10 000 Euros
Frais d'abonnement aux API	3 400 Euros
Serveurs sur cloud	524 Euros
Documentation finale	3 700 Euros
Frais de repas et de transport du personnel	15 810 Euros

Soit un budget total prévisionnel du projet estimé à : 178 784 Euros.

Le chiffrage détaillé est disponible en annexe B.

5. Suivi du projet et indicateurs de performance

5.1. Suivi du projet

Le projet de mise en place de l'infrastructure pour aéroport de Paris sera piloté avec la méthode agile Scrum. Les 11 phases du rétroplanning correspondent chacune à un sprint comprenant des tâches et un jalon confirmant la disponibilité des livrables.

L'ensemble peut être visualisé en annexe C sous la forme d'un planning traditionnel sous Microsoft Project. Ce planning sera transformé en backlog Scrum avec un outil tel que Jira ou Azure DevOps afin de réévaluer les tâches en fonction des retours et de l'évolution du projet et laisser une flexibilité dans la planification par des ajustements en collaboration avec l'équipe selon les retours après chaque sprint.

Les tâches seront également priorisées par valeur métier pour sensibiliser les intervenants sur les tâches critiques et renforcer l'impact positif du travail collaboratif. Cette valeur sera exprimée à l'aide de user stories et des méthodes de priorisation des tâches comme MOSCOW.

Ces dispositions permettront d'aboutir à un backlog évolutif et collaboratif pour que le projet puisse avancer dans la bonne direction.

5.2. Indicateurs de performance

La progression du projet et la qualité des livrables pourront être mesurées avec des key performances indicators (KPIs) regroupés en trois catégories : KPIs pour la livraison et le suivi des sprints, KPIs pour la robustesse de l'infrastructure, KPIs pour la valeur métier ainsi que pour l'adoption par l'équipe et les utilisateurs finaux de la solution d'architecture.

KPIs liés au planning

- nombre de user stories réalisées par Sprint (accroître la capacité de livraison)
- nombre de tâches restantes à accomplir jusqu'à la fin du Sprint (détecter un retard)
- délai de mise en production d'une fonctionnalité (optimiser le flux de travail)

KPIs liés à la robustesse de l'infrastructure

- taux d'échec du pipeline Aviationstack (mesurer la fiabilité du workflow)
- taux d'échec du pipeline Opensky/ Skytrax (mesurer la fiabilité du workflow)
- temps moyen d'indexation des données dans Elasticsearch (détecter des goulots d'étranglement)
- nombre d'alertes émises par Airflow (détecter des anomalies dans les processus automatisés)

KPIs liés à la valeur métier et l'adoption

- nombre de requêtes utilisateurs sur Kibana (voir si les dashboards sont utilisés)
- pourcentage de précision du modèle prédictif sur l'infrastructure (évaluer la justesse de prédiction d'événements comme les goulots d'étranglement)
- taux de satisfaction client sur la qualité des données traitées et des dashboards (ajuster les transformations de données)

6. Description technique du projet

6.1. Collecte des données

Des données brutes sont extraites par deux pipelines parallèles. Le premier récupère des informations de vols commerciaux via l'API Aviationstack avec un job Talend sous Windows et un token d'accès. Le second sous Linux extrait des données d'aéronefs en vol de l'API Opensky et l'évaluation des compagnies du site Skytraxratings avec une application conteneurisée utilisant un script python ainsi que les servives Kafka et Zookeeper.

Les données des deux pipelines sont ensuite envoyées au format JSON dans des buckets S3 sous Localstack (phase de test) puis AWS (phase de production) représentant le data lake.

6.2. Transformation des données

Un script conteneurisé utilisant Spark et PySpark télécharge les fichiers des buckets S3 pour les nettoyer en retirant les données avec valeurs manquantes, puis les enrichir avec les coordonnées géographiques des aéroports de départ et d'arrivée. Des transformations sont également faites pour obtenir les heures UTC de départ/arrivée en secondes unix, la durée des vols ou encore le jour de la semaine et l'heure de départ.

Le même script se charge également d'indexer dans Elasticsearch les données traitées dans 3 index distincts Aviationstack, Opensky et Skytrax en veillant à ne pas envoyer de documents déjà existants dans ces index.

6.3. Stockage des données

Un fichier docker compose synchronise le démarrage des conteneurs Elasticsearch et Kibana représentant le data warehouse à partir duquel seront réalisés les dashboards dynamiques.

C'est donc dans ces conteneurs que figurent les 3 index des données traitées pouvant persister après arrêt et redémarrage des services.

6.4. Visualisation des données

Un tableau de bord est réalisé sous Kibana à partir des données des 3 index. Il comprend des visualisations interactives comme une carte mondiale des trajets, une carte mondiale de la position des aéronefs avec informations sur l'altitude, la vitesse et le cap, un histogramme sur le nombre de vols opérés par les compagnies aériennes ainsi que des métriques sur le temps de vol et les intervalles longitude/latitude.

6.5. Analyse prédictive des performances et gestion des workflows

Des points de collecte des performances sur l'infrastructure envoient des données sous forme de séries temporelles à Prometheus. Ces données historisées sont ensuite envoyées vers scikit-learn pour entraîner un modèle d'IA et prédire des événements comme l'apparition de goulots d'étranglement. Les alertes, anomalies et actions correctives issues de l'analyse prédictive sont ensuite déclenchées automatiquement par Apache Airflow sur les zones monitorées de l'architecture.

Aussi , Github Actions est utilisé pour créer des pipelines d'intégration continue (CI) et de livraison continue (CD) opérationnels dans l'environnement de développement et prêts pour un déploiement de l'infrastructure en production.

6.6. Test de l'infrastructure et déploiement final

Les pipelines CI/CD sont utilisés pour réaliser des workflows de test à chaque modification du code en incluant des tests unitaires et des tests d'intégration pour vérifier le bon fonctionnement de chaque partie de l'infrastructure et vérifier que tous les services fonctionnent correctement entre eux. Pour la partie Ops, un cluster EKS est créé sur AWS puis les services conteneurisés en Dev deviennent des pods tournant sur les nœuds du cluster en Ops. Les données sont chiffrées avec AWS KMS et les accès aux pods et données contrôlés avec IAM Roles et Secrets Manager.

Apres validation de ces tests, l'infrastructure est vérifiée dans un environnement de staging via Github Actions afin d'obtenir une validation finale en s'assurant de la conformité de l'infrastructure aux exigences. Des tests fonctionnels sur données nouvelles sont réalisés puis des déploiements automatisés dans l'environnement final de production sont effectués pour assurer la continuité entre les environnements de développement, staging et production et faciliter les mises à jour continues.

6.7. Accompagnement des utilisateurs finaux

La première partie de ce Sprint est consacrée à la réalisation d'une documentation technique d'exploitation et de maintenance de l'infrastructure pour fournir une vue détaillée de l'infrastructure et promouvoir de bonnes pratiques d'utilisation.

La seconde partie vise à former le personnel du client aéroport de Paris sur l'utilisation et la surveillance de l'infrastructure ainsi que sur l'exploitation des informations fournies par le dashboard.

Cette démarche socio-organisationnelle et humaine vise à obtenir l'adhésion de la solution déployée et implique une analyse des retours utilisateurs pour des ajustements et ainsi obtenir une transition réussie vers la nouvelle infrastructure.

7. Direction et gestion du projet

7.1. Gestion de l'équipe

Le projet sera réalisé par une équipe comprenant:

- un Chef de projet représentant du client et incluant dans ses missions le rôle de Product Owner
- un Scrum Master
- un Data Engineer
- un Ingénieur DevOps
- un Data Scientist
- un UX/UI Designer

Les rôles de chacun sont répartis de la façon suivante :

- Chef de projet (maitrise d'ouvrage) : définit la vision du projet, priorise les besoins et assure la liaison avec les parties prenantes
- Scrum Master (maîtrise d'œuvre) : facilite l'utilisation de la méthodologie Agile, veille à la fluidité des échanges et à la levée des obstacles
- Data Engineer : configure les environnements, conçoit et met en place les pipelines de collecte, transformation et stockage des données
- Ingénieur DevOps : automatise le déploiement de l'infrastructure en mode production et assure sa fiabilité via les pipelines CI/CD et monitoring des performances en lien avec le Data Engineer
- Data Scientist : développe et entraîne le modèle de Machine Learning pour l'analyse prédictive des performances de l'infrastructure et la création d'actions correctives
- UX/UI Designer : apporte son expertise pour concevoir des tableaux de bord de visualisation ergonomiques, intuitifs et permettant aux utilisateurs finaux d'interagir avec les données

Chaque Sprint intégrera les réunions suivantes :

- Daily Scrum, meeting court le matin pour résoudre les points durs et synchroniser la progression des membres de l'équipe

- Sprint planning, en début de Sprint pour confirmer la répartition des missions selon les priorités
- Sprint Review, en fin de Sprint pour validation des livrables et axes d'amélioration pour la suite du projet

7.2. Suivi des ressources et respect des délais

Pour un suivi efficace, un outil comme Jira Software sera utilisé pour gérer le backlog produit, attribuer les tâches aux membres de l'équipe et suivre l'avancement des sprints via des tableaux Kanban pour suivi en temps réel de la progression des tâches et des burndown charts pour visualiser et ajuster la charge de travail restante dans le sprint.

En complément, une analyse des risques sera effectuée avec un outil comme Confluence permettant d'identifier, catégoriser, évaluer la criticité et la probabilité d'occurrence du risque et d'établir les parades pour la maîtrise de ce risque. Cette analyse sera centralisée dans un registre accessible par l'ensemble de l'équipe projet.

L'intégration de **GitHub Actions pour l'automatisation des pipelines CI/CD** et l'utilisation de **workflows collaboratifs** permettra aux membres de l'équipe travaillant sur l'environnement de développement et sur l'environnement de production de synchroniser leurs livrables respectifs pour un déploiement rapide et fiable de l'infrastructure.

8. Conclusion

Ce projet a démontré les compétences acquises dans la gestion d'un projet de collecte, transformation, stockage et visualisation dynamique de données traitées sur les vols aériens. La solution développée offre une infrastructure résistant aux indisponibilités des sources de par la redondance des pipelines d'extraction, robuste de par la mise en place d'actions correctives apportées par l'intelligence artificielle et modulable pour son évolution et sa scalabilité.

En intégrant des technologies comme le streaming et l'Internet des Objets (IoT), cette architecture peut être enrichie pour améliorer la fluidité du trafic aérien. Par exemple, l'utilisation de capteurs IoT apporterait une surveillance en temps réel des mouvements des passagers dans les aéroports pour en déduire les zones de congestion et ainsi anticiper la gestion des flux pour réduire le temps d'attente pour l'embarquement.

En exploitant dans la durée l'architecture déployée et en y apportant de nouvelles fonctionnalités comme celles précédemment décrites, les gestionnaires d'aéroport pourront offrir aux voyageurs une expérience de voyage plus fluide et efficace en assurant un embarquement à l'heure et sans stress des passagers, ce qui bénéficie également aux compagnies aériennes.