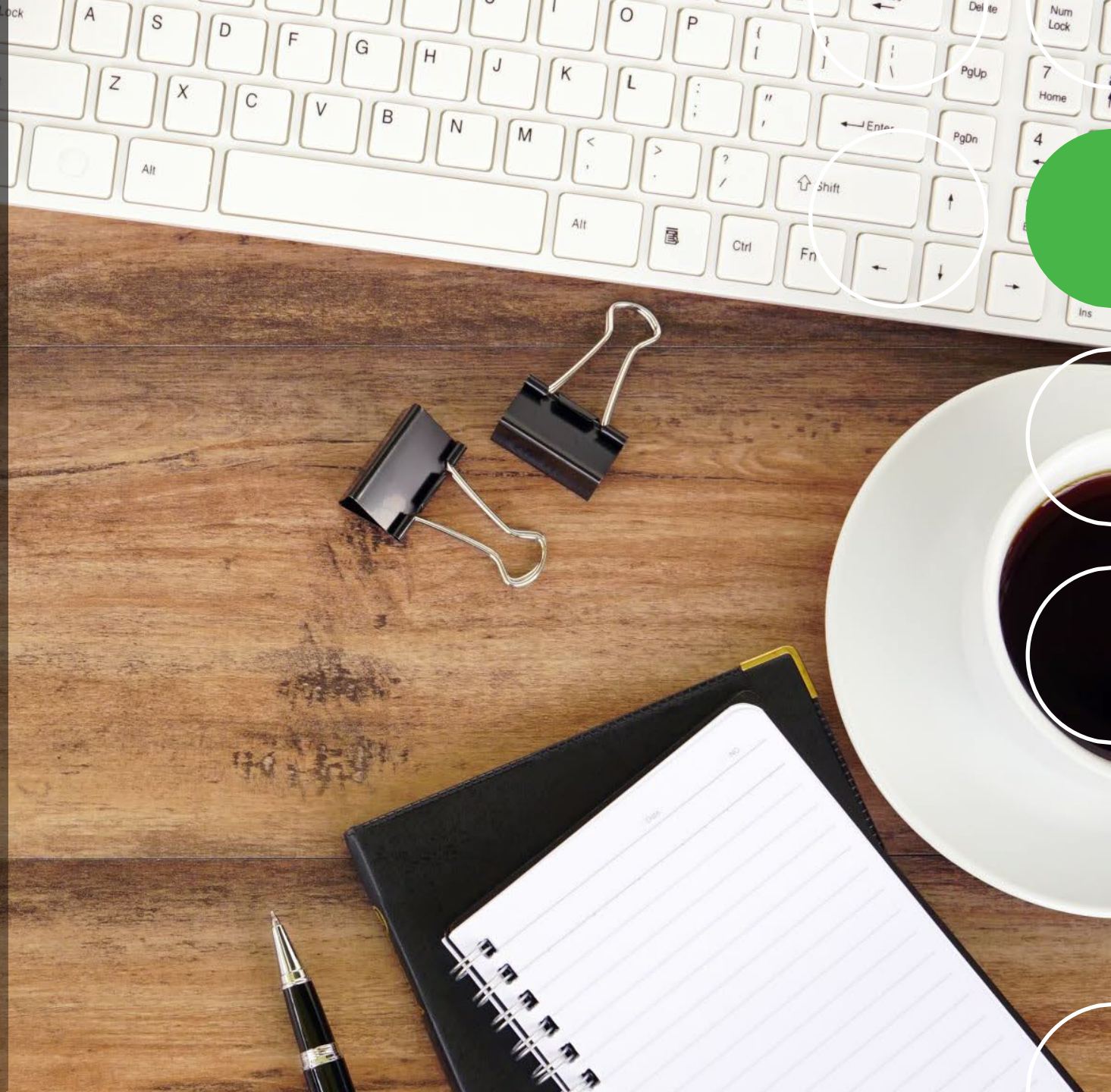
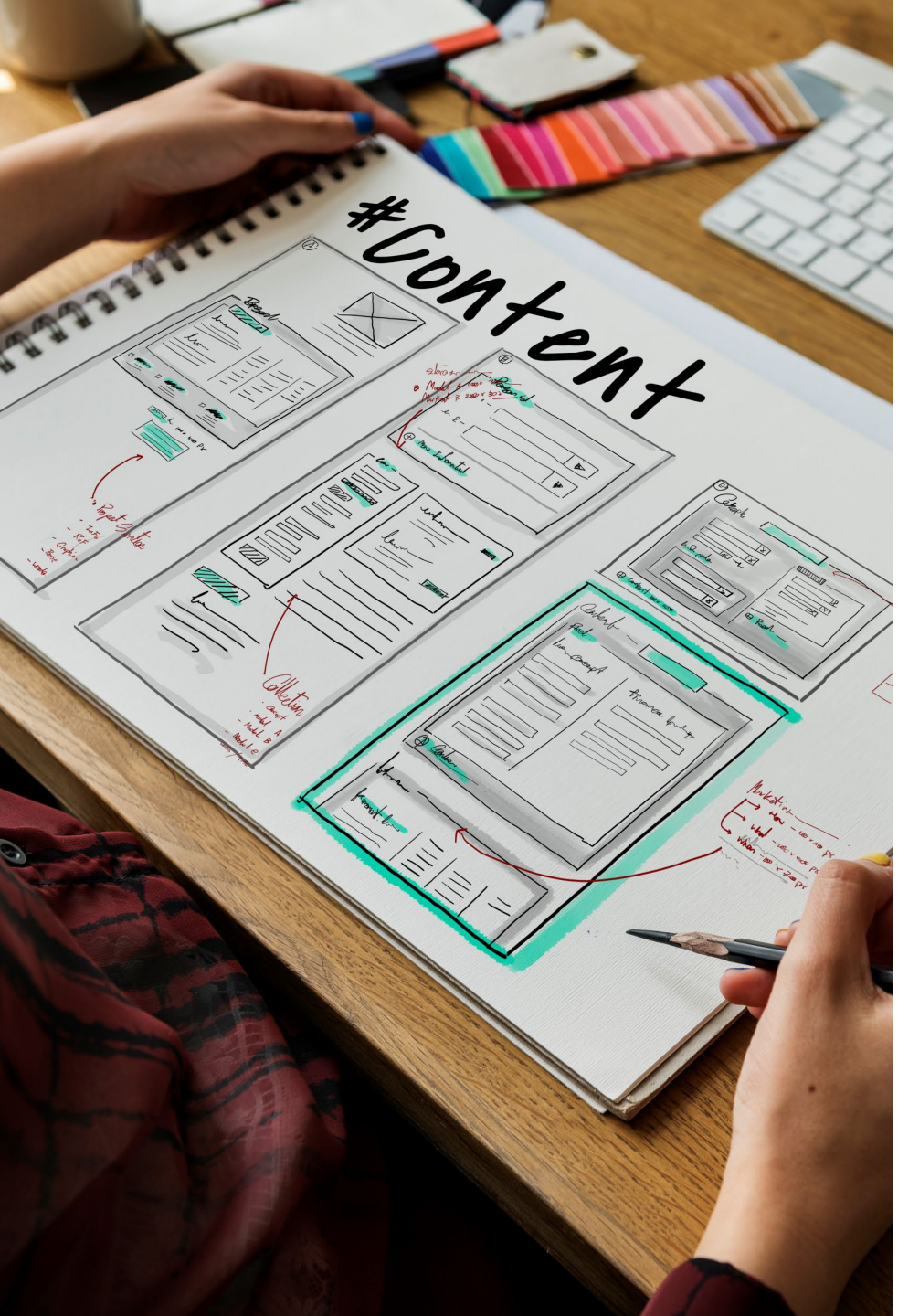
Architecture Streaming Data

Architecture d'ingestion des données

Amina MARIE





Ordre du jour

Méthodes d'ingestion des données en streaming

Optimisation de la performance de l'ingestion des données

Gestion de la scalabilité et de la résilience dans l'ingestion des données

Exercice 1 : Mise en place d'une ingestion de données en streaming avec une méthode push-based à l'aide d'Apache Kafka Connect.

Exercice 2 : Configuration de la scalabilité et de la résilience de l'ingestion des données avec Apache Kafka et Apache Flink.

Méthodes d'ingestion des données en streaming

Rappel : qu’est-ce que l’ingestion de données en streaming

Processus d'acquisition et de traitement continu des données à mesure qu'elles sont générées.

On traite les données dès qu'elles deviennent disponibles, sans attendre la fin d'un cycle de traitement.

Ce processus d'ingestion de données en streaming implique généralement les étapes suivantes :

1. **Acquisition des données** : Les données sont collectées à partir de différentes sources telles que des capteurs IoT, des journaux d'applications, des bases de données, des API, des clics utilisateur sur des sites web, etc.
2. **Transmission des données** : Les données collectées sont ensuite transmises à un système de traitement en continu. Cela peut se faire via des protocoles de communication comme HTTP, MQTT, ou des technologies spécifiques comme Apache Kafka, Amazon Kinesis, ou encore des outils ETL (Extract, Transform, Load) comme Apache NiFi.
3. **Traitement des données** : Une fois les données ingérées, elles peuvent être transformées, filtrées, agrégées, enrichies ou analysées en temps réel en fonction des besoins métier. Cette étape peut être réalisée à l'aide de divers outils et frameworks de traitement de données en streaming tels que Apache Spark Streaming, Apache Flink, ou des outils de traitement événementiel.
4. **Stockage ou publication des données** : Après traitement, les données peuvent être stockées dans des bases de données en temps réel, des entrepôts de données, des systèmes de fichiers distribués ou publiées vers d'autres systèmes pour une utilisation ultérieure, par exemple pour la visualisation, le reporting ou l'analyse avancée.

Les méthodes push-based et pull-based

Push-Based (Poussée)

Les données sont envoyées ou "poussées" vers le système de traitement en continu dès qu'elles sont disponibles.

Les producteurs de données sont responsables d'envoyer activement les données vers le système de traitement en continu

Cette approche convient bien aux cas d'utilisation où les données sont générées de manière proactive et doivent être traitées en temps réel dès leur arrivée.

Exemples : Apache Kafka, Amazon Kinesis Data Streams, MQTT.

Pull-Based (Tirée)

Le système de traitement en continu récupère activement les données à partir des sources de données disponibles dès qu'il est prêt à les traiter.

Les consommateurs de données contrôlent le processus d'acquisition des données en interrogeant ou en "tirant" les données depuis les sources, généralement à intervalles réguliers.

Cette approche convient aux cas d'utilisation où les données sont stockées dans des sources accessibles et peuvent être récupérées à la demande.

Exemples : Apache Flume, Apache NiFi, Logstash.

Cas d’utilisation push-based

1. **IoT (Internet des Objets)** :

   Dans un scénario IoT, des capteurs et des dispositifs connectés peuvent envoyer activement des données telles que des mesures de température, d'humidité, ou de géolocalisation vers une plateforme de traitement en continu.

   Les données sont envoyées dès qu'elles sont générées, ce qui permet de surveiller en temps réel les conditions environnementales, les performances des équipements, ou de détecter des incidents potentiels.

2. **Analyse des logs en temps réel** :

   Dans un environnement informatique, les serveurs, les applications et les services génèrent continuellement des logs et des événements.

   En utilisant une méthode push-based, les logs peuvent être envoyés en temps réel vers une plateforme de traitement en continu comme Apache Kafka ou Amazon Kinesis pour une analyse en temps réel, la détection des anomalies, ou la surveillance des performances.

Cas d’utilisation pull-based

1. **Intégration de données hétérogènes** :

   Dans un environnement d'entreprise, les données peuvent être stockées dans différentes sources telles que des bases de données, des systèmes de fichiers, ou des services web.

   Un processus de traitement en continu peut être configuré pour récupérer activement les données à partir de ces sources en utilisant une méthode pull-based, en interrogeant les API, en interrogeant les bases de données, ou en surveillant les répertoires de fichiers.

2. **Traitement de flux de données batch** :

   Parfois, les données sont générées sous forme de batch plutôt que de manière continue.

   Un processus de traitement en continu peut être configuré pour récupérer périodiquement des données à partir de systèmes de stockage tels que Hadoop HDFS, Amazon S3, ou Azure Data Lake Storage à l'aide d'une méthode pull-based, convertissant ainsi les flux de données batch en flux de données en streaming.

La méthode event-based

Les données sont ingérées en fonction de la survenue d'événements déclencheurs spécifiques.

L'ingestion des données est déclenchée par des événements spécifiques qui se produisent dans le système ou l'environnement.

Lorsqu'un événement se produit, le système réagit en conséquence en ingérant et en traitant les données associées à cet événement.

Cette méthode offre une flexibilité et une réactivité accrues en permettant aux systèmes de réagir de manière dynamique aux événements et aux actions spécifiques.

Cas d’utilisation event-based

Déclenchement par Changement de Statut

L'ingestion des données est déclenchée chaque fois qu'un changement de statut se produit dans le système. Par exemple, lorsqu'un nouvel utilisateur s'inscrit sur un site web, un événement est déclenché, déclenchant l'ingestion des données utilisateur associées.

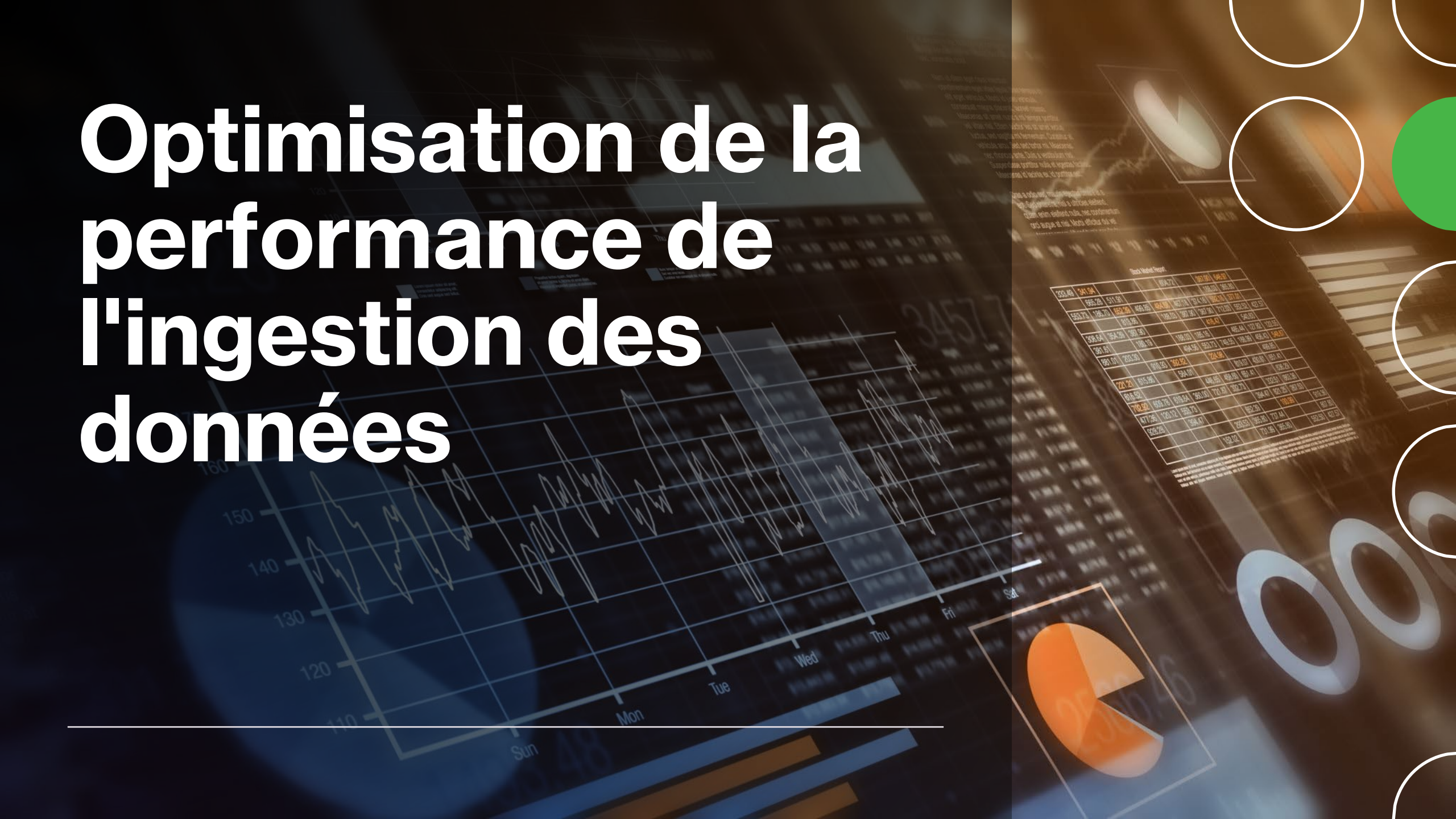
Déclenchement par Interaction Utilisateur

Les données sont ingérées en réponse à des interactions utilisateur spécifiques. Par exemple, lorsqu'un utilisateur effectue un achat en ligne, un événement est déclenché, déclenchant l'ingestion des données de transaction associées.

Déclenchement par Capteur IoT

Les données sont ingérées en fonction des lectures de capteurs IoT. Par exemple, lorsqu'un capteur de température détecte une augmentation soudaine de la température, un événement est déclenché, déclenchant l'ingestion des données de température.

Optimisation de la performance de l'ingestion des données



Pourquoi optimiser la performance de l’ingestion des données

Temps de réponse en temps réel

Une ingestion de données optimisée garantit que les données sont traitées rapidement, ce qui permet aux applications de réagir rapidement aux événements en temps réel.

Évolutivité

Une ingestion de données optimisée permet de maintenir les performances même lorsque le volume de données augmente, en permettant la scalabilité horizontale pour ajouter de nouveaux nœuds ou partitions au système.

Utilisation efficace des ressources

Une ingestion de données inefficace peut entraîner une utilisation excessive des ressources système, telles que la CPU, la mémoire et le réseau. Une optimisation de la performance permet de minimiser l'utilisation des ressources tout en maximisant le débit de traitement des données.

Pourquoi optimiser la performance de l’ingestion des données

Fiabilité et disponibilité

Une ingestion de données optimisée contribue à garantir la fiabilité et la disponibilité du système en réduisant les risques de goulets d'étranglement, de pannes et de temps d'arrêt imprévus.

Cela permet de maintenir la continuité des opérations même en cas de fluctuations de charge ou de pannes matérielles.

Coûts opérationnels

Une ingestion de données inefficace peut entraîner des coûts opérationnels plus élevés, notamment en termes de matériel, de bande passante réseau et de maintenance.

Une optimisation de la performance permet de réduire les coûts opérationnels en maximisant l'efficacité des ressources disponibles.

Comment optimiser?

En mettant en œuvre des stratégies et bonnes pratiques!

Partitionnement des données :

Utilisez le partitionnement des données pour répartir la charge de manière équilibrée entre les différents nœuds ou partitions.

Cela permet de paralléliser le traitement des données et d'éviter les goulets d'étranglement lors de l'ingestion.

Optimisation des schémas de données :

Concevez des schémas de données efficaces et optimisés pour réduire la taille des données et améliorer les performances de traitement.

Évitez les structures de données complexes et les champs non utilisés pour réduire la surcharge de traitement.

Compression des données :

Utilisez la compression des données pour réduire la taille des données transférées sur le réseau et minimiser l'utilisation des ressources.

Choisissez des algorithmes de compression efficaces adaptés aux types de données et aux flux de données en streaming.

Comment optimiser?

Optimisation du réseau :

Optimisez les configurations réseau pour minimiser la latence et maximiser la bande passante disponible.

Utilisez des protocoles de communication réseau efficaces et configurez les paramètres réseau pour optimiser les performances.

Mise en cache des données :

Utilisez des caches pour stocker temporairement les données fréquemment utilisées et réduire le temps de traitement.

Choisissez des technologies de mise en cache adaptées aux besoins de votre application, telles que Redis, Memcached, ou des solutions de mise en cache distribuées.

Scalabilité horizontale :

Concevez votre système pour permettre la scalabilité horizontale afin de gérer efficacement les charges de données croissantes.

Utilisez des solutions distribuées et des architectures modulaires pour ajouter facilement de nouveaux nœuds ou partitions au système en cas de besoin.

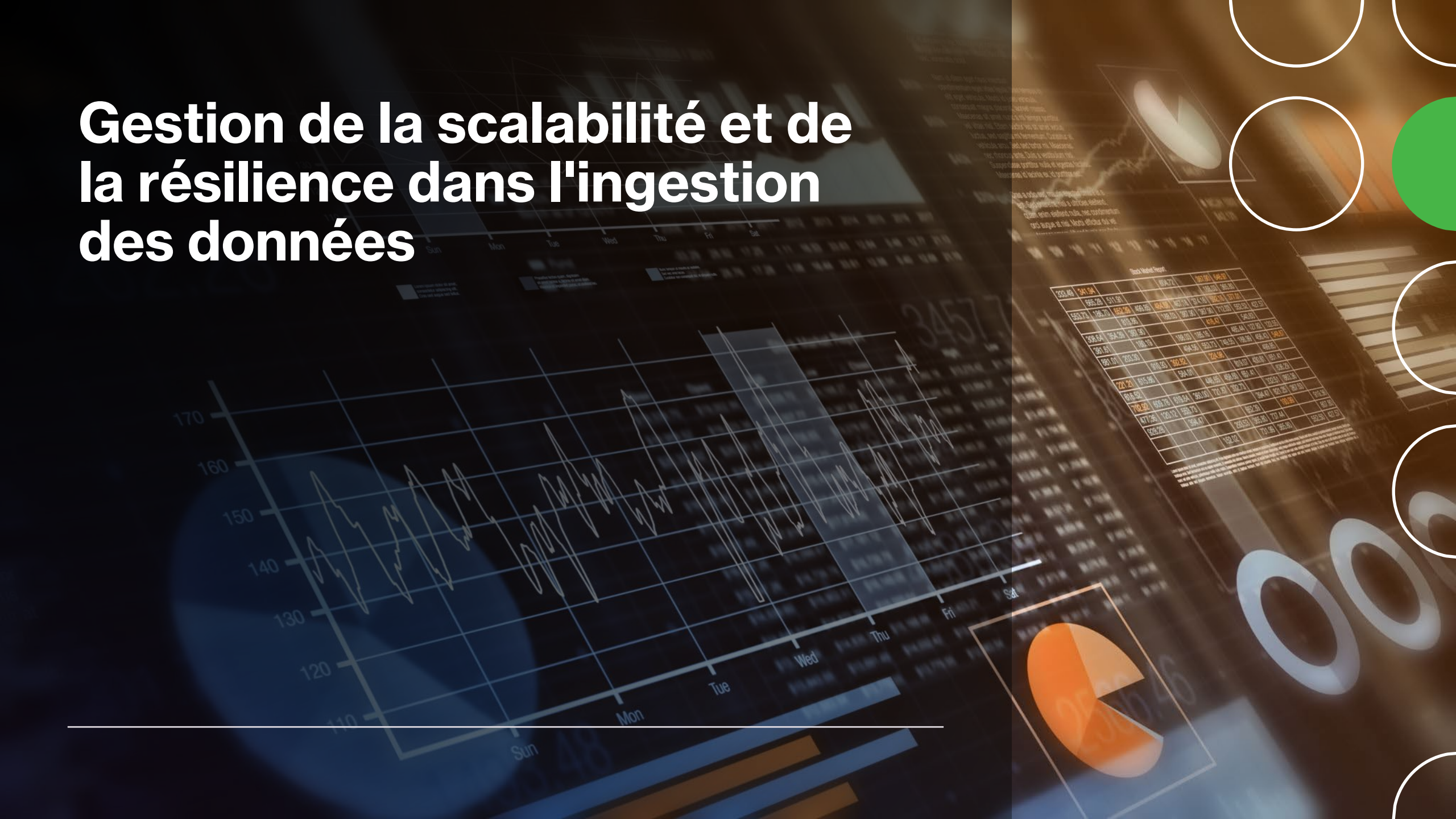
Comment optimiser?

Optimisation des requêtes et des transformations :

Optimisez les requêtes et les transformations de données pour réduire la complexité et améliorer les performances.

Utilisez des algorithmes efficaces et des techniques d'optimisation des requêtes pour minimiser le temps de traitement et la consommation de ressources.

Gestion de la scalabilité et de la résilience dans l'ingestion des données



C’est quoi scalabilité et résilience?

Scalabilité :

La scalabilité fait référence à la capacité d'un système à gérer efficacement une augmentation de la charge de données sans sacrifier les performances.

Dans le contexte de l'ingestion des données en streaming, la scalabilité signifie que le système peut évoluer pour traiter un volume croissant de données en continu sans compromettre sa capacité à répondre rapidement et efficacement.

Résilience :

La résilience se réfère à la capacité d'un système à continuer à fonctionner de manière fiable malgré les pannes matérielles, les pannes logicielles, ou d'autres événements indésirables.

Dans le contexte de l'ingestion des données en streaming, la résilience signifie que le système est conçu pour détecter et gérer les pannes de manière transparente, en minimisant les temps d'arrêt et en assurant la continuité des opérations.

En combinant la scalabilité et la résilience, les systèmes d'ingestion des données en streaming peuvent évoluer pour gérer efficacement les charges de données croissantes, tout en restant disponibles et fiables même en cas de pannes ou de défaillances.

Cela garantit que les données sont traitées de manière efficace, en temps réel et de manière fiable, ce qui est essentiel pour de nombreuses applications et cas d'utilisation en temps réel.

Comment gérer la scalabilité dans l’ingestion de données en streaming?

Scalabilité horizontale :

Concevez votre système pour permettre la scalabilité horizontale, c'est-à-dire la capacité à ajouter de nouveaux nœuds ou partitions au système en réponse à une augmentation de la charge de données.

Utilisez des architectures distribuées et des technologies de traitement de données en streaming qui prennent en charge la scalabilité horizontale, telles que Apache Kafka, Apache Flink..

Partitionnement des données :

Utilisez le partitionnement des données pour répartir la charge de manière équilibrée entre les différents nœuds ou partitions.

Assurez-vous que le partitionnement est conçu de manière à permettre une distribution uniforme des données et à éviter les goulets d'étranglement.

Élasticité automatique :

Mettez en place des mécanismes d'élasticité automatique pour ajuster dynamiquement les ressources en fonction de la charge de données en temps réel.

Utilisez des services cloud et des plates-formes qui offrent des fonctionnalités d'élasticité automatique, telles que AWS Auto Scaling ou Azure Autoscale.

Comment gérer la scalabilité dans l’ingestion de données en streaming?

L'élasticité automatique dans l'ingestion de données en streaming fait référence à la capacité d'un système à ajuster automatiquement ses ressources en fonction des fluctuations de charge de données, sans nécessiter d'intervention manuelle de la part des opérateurs du système.

Cette fonctionnalité permet au système de s'adapter dynamiquement aux variations de la charge de données, en augmentant ou en réduisant les ressources de traitement en fonction des besoins.

Comment gérer la résilience dans l’ingestion de données en streaming?

Réplication des données :

Utilisez la réplication des données pour garantir la redondance et la disponibilité des données en cas de défaillance d'un nœud ou d'une partition.

Configurez des stratégies de réplication appropriées pour assurer la cohérence des données et minimiser les risques de perte de données.

Gestion des défaillances :

Mettez en place des mécanismes de gestion des défaillances pour détecter et gérer automatiquement les pannes matérielles, les pannes logicielles ou les erreurs de traitement.

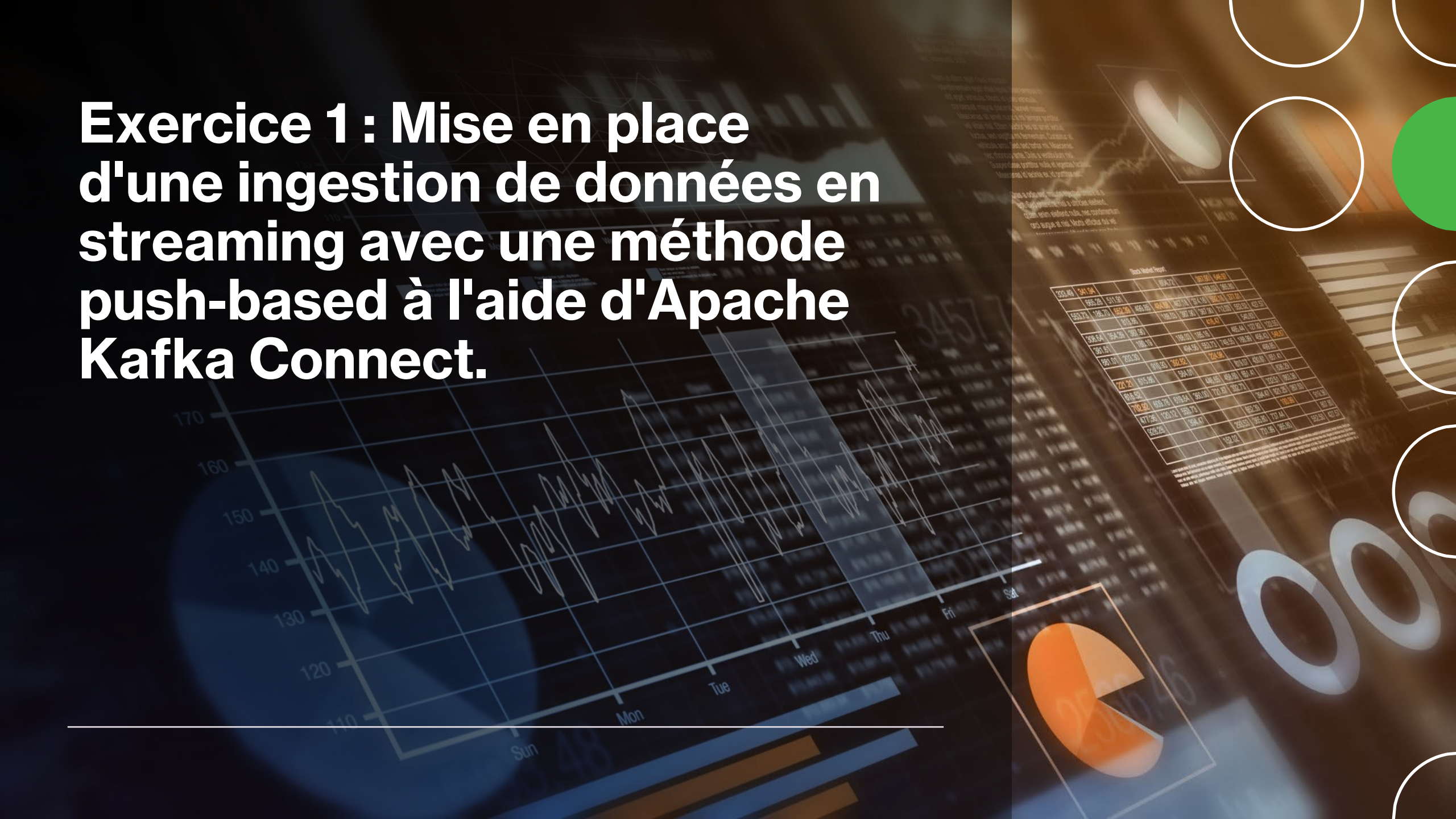
Utilisez des architectures tolérantes aux pannes et des technologies de haute disponibilité pour minimiser les temps d'arrêt et garantir la continuité des opérations.

Surveillance et alertes :

Mettez en place un système de surveillance et d'alerte pour surveiller en permanence les performances du système, la charge de données et les événements de défaillance.

Configurez des seuils d'alerte appropriés pour détecter les anomalies et les problèmes potentiels avant qu'ils ne deviennent critiques.

Exercice 1 : Mise en place d'une ingestion de données en streaming avec une méthode push-based à l'aide d'Apache Kafka Connect.



Exercice

Objectif

Configurer un connecteur Apache Kafka Connect pour ingérer des données à partir d'un fichier CSV vers un topic Kafka.

Les étapes

- Installation

Assurez-vous d'avoir Apache Kafka et Apache Kafka Connect installés et configurés sur votre système

- Création d'un fichier CSV de test

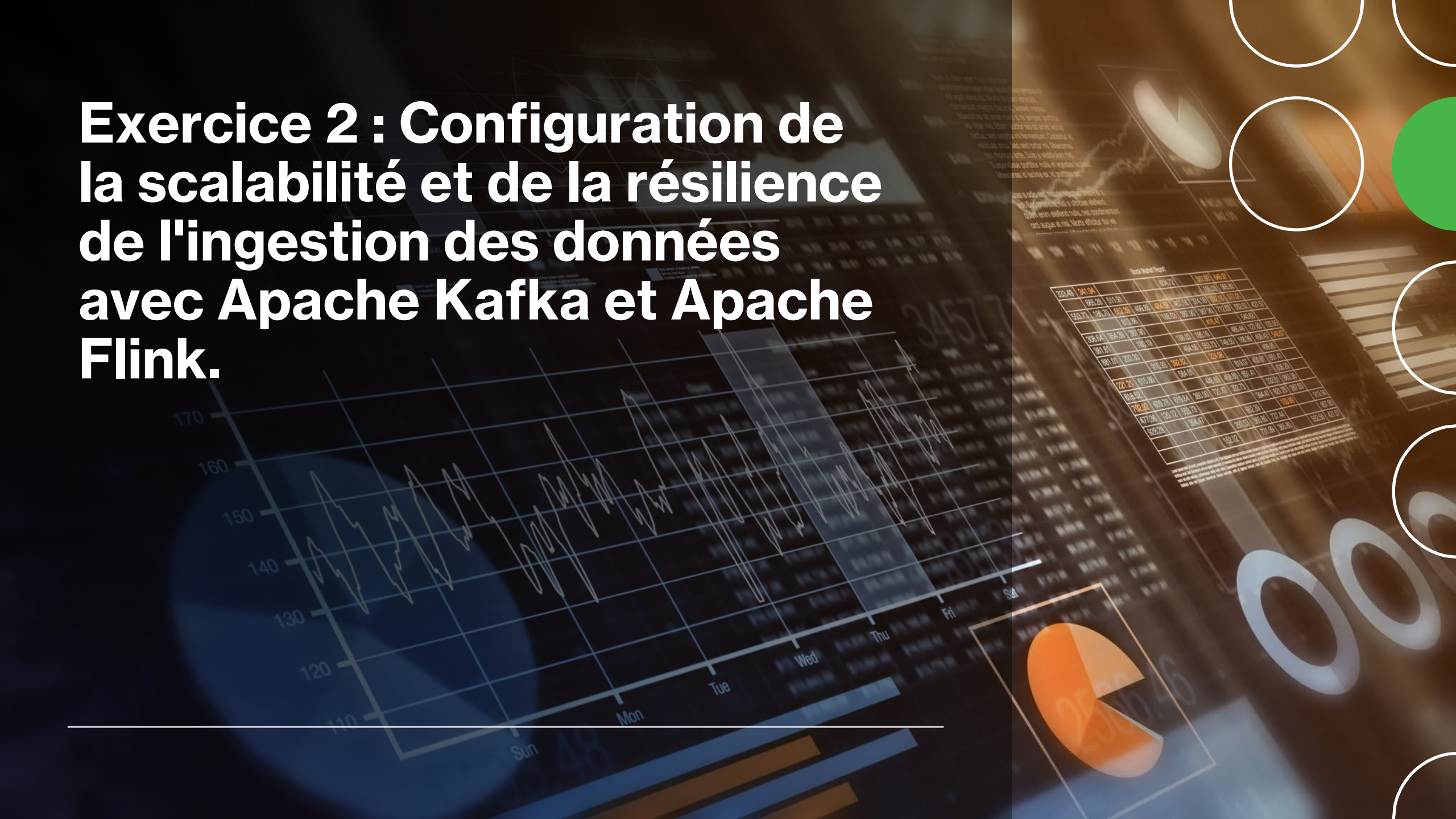
Créez un fichier CSV de test contenant des données que vous souhaitez ingérer dans Apache Kafka.

- Configuration du connecteur Kafka Connect :

Créez un fichier de configuration pour le connecteur Kafka Connect. Voici un exemple de configuration minimale pour ingérer des données à partir d'un fichier CSV :

```properties
name=file-source
connector.class=FileStreamSource
tasks.max=1
file=/chemin/vers/votre/fichier.csv
topic=nom_du_topic_kafka
```

Exercice 2 : Configuration de la scalabilité et de la résilience de l'ingestion des données avec Apache Kafka et Apache Flink.



Exercice

- Reprendre l’exercice 1 et me faire un rendu où vous devez :

m’expliquer comment configurer la scalabilité et de la résilience de l'ingestion des données avec Apache Kafka

- M’expliquer de façon globale comment vous feriez pour configurer la scalabilité et de la résilience de l'ingestion des données Apache Flink