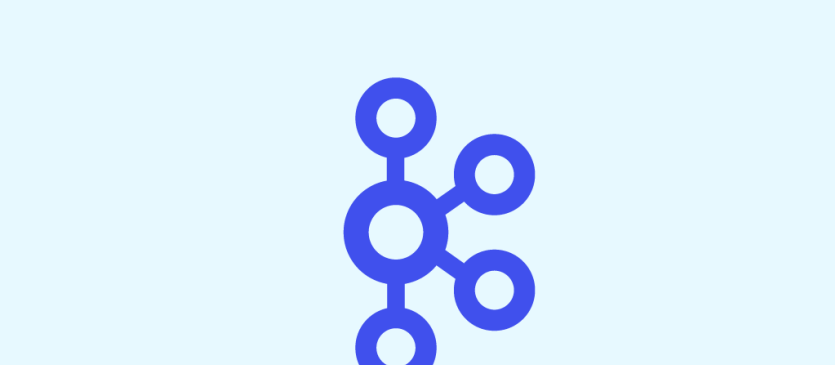

# Apache Kafka : la plateforme de traitement de données en temps réel

**Margot P**

**30 avril 2021**

**6 min**

Apache Kafka est une plateforme de traitement de données en streaming temps réel. Découvrez tout ce qu'il ya a à savoir pour maitriser Kafka.

Le **traitement de données en streaming** offre de nombreux avantages. Cette approche permet notamment de mettre en place une architecture de Data Engineering plus efficacement. Toutefois, des technologies additionnelles sont nécessaires. L'une de ces technologies est Apache Kafka.

## Qu'est-ce que Apache Kafka ?

Apache Kafka est **une plateforme de streaming de données open source**. À l'origine, elle fut développée en interne par LinkedIn en guise de queue de messagerie. Toutefois, cet outil largement évolué et ses cas d'usage se sont multipliés.

Cette plateforme est écrite **en Scala et Java**. Elle est toutefois compatible avec une large variété de langages de programmation.

À la différence de queues de messagerie traditionnelles comme RabbitMQ, **Kafka retient les messages** après qu'ils aient été consommés pendant une certaine période de temps. Les messages ne sont pas supprimés immédiatement après confirmation de réception.

En outre, les queues de messageries sont d'ordinaire conçues pour s'étendre verticalement via l'ajout de puissance à une machine. De son côté, **Kafka s'étend horizontalement grâce à l'ajout de noeuds** supplémentaires au cluster de serveurs.

Il faut savoir que **Kafka est distribué**. Cela signifie que ses capacités sont élastiques. Il suffit d'ajouter des noeuds, c'est-à-dire des serveurs, à un cluster pour l'étendre.

Une autre particularité de Kafka est **sa faible latence**. Cela signifie qu'elle peut prendre en charge le traitement de nombreuses données en temps réel.

## Les principaux concepts Apache Kafka

Pour comprendre le fonctionnement de Apache Kafka, il est nécessaire de comprendre plusieurs concepts. Tout d'abord, **un " évènement "** est un morceau atomique de donnée. Un événement est par exemple créé dès qu'un utilisateur s'enregistre sur un système.

Un événement peut aussi être perçu comme **un message contenant des données**. Ce message peut être traité et sauvegardé quelque part si nécessaire. Pour reprendre l'exemple de l'enregistrement sur un système, l'événement sera un message contenant des informations comme le nom d'utilisateur, l'adresse email ou le mot de passe. Ainsi, Kafka est une plateforme permettant de travailler avec les flux d'événements.

Les **événements sont écrits par des " producteurs "** : un autre concept phare. Il existe différents types de producteurs : les serveurs web, les composants d'applications, les appareils IoT… tous écrivent des événements et les transmettent à Kafka. Par exemple, un thermomètre connecté produira chaque heure des " événements " contenant des informations sur la température, l'humidité ou la vitesse du vent.

À l'inverse, **le " consumer " ou " consommateur "** est une entité utilisant les événements de données. Il reçoit les données écrites par le producteur et les utilise. En guise d'exemple, on peut citer les bases de données, les Data Lakes ou encore les applications analytiques. Une entité peut être à la fois producteur et consommateur, à l'instar des applications ou des composants d'applications.

Les producteurs **publient les événements sur des " topics "** Kafka. Les consommateurs peuvent s'abonner pour obtenir l'accès aux données dont ils ont besoin. Les topics sont des séquences

d'événement, et chacun peut servir des données à de multiples consommateurs. Ainsi, les producteurs sont parfois appelés " publishers " et les consommateurs " subscribers ".

En réalité, **Kafka fait office d'intermédiaire** entre les applications générant des données et les applications consommant des données. Un cluster Kafka comporte de multiples serveurs appelés " noeuds ".

Les **" brokers " sont des composants logiciels** exécutés sur un noeud. Les données sont distribués entre plusieurs brokers d'un cluster Kafka, et c'est pourquoi il s'agit d'une solution distribuée.

Il existe **plusieurs copies des données sur un même cluster**, et ces copies sont appelées " répliques ". Ce mécanisme rend Kafka plus stable, tolérant aux erreurs et fiable. Les informations ne sont pas perdues en cas de problème sur un broker. Un autre prendra le relais.

Enfin, **les partitions servent à répliquer les données** entre les brokers. Chaque topic Kafka est divisé en partitions multiples, et chaque partition peut être placée sur un noeud séparé.



# Quels sont les cas d'usage d'Apache Kafka ?

Les cas d'usage d'Apache Kafka sont nombreux. On l'utilise pour **le traitement de données en temps réel**. En effet, pour fonctionner, de nombreux systèmes modernes requièrent que les données soient traitées dès qu'elles sont disponibles.

Par exemple, **dans le domaine de la finance**, il est indispensable de bloquer immédiatement les transactions frauduleuses. De même, pour la maintenance prédictive, les flux de données en provenance des équipements doivent être surveillés continuellement pour donner l'alerte dès que les problèmes sont détectés.

Les **objets connectés** requièrent eux aussi un traitement de données en temps réel. Dans ce contexte, Kafka est très utile puisqu'il permet le transfert et le processing en streaming

puisqu'il permet le transfert et le processing en streaming.

À l'origine, Kafka fut créé par Linkedin **pour le tracking d'activité d'application**. C'est donc son cas d'usage originel. Chaque événement survenant dans l'application peut être publié vers le topic Kafka correspondant.

Les **clics d'utilisateurs, les inscriptions, les " likes "**, le temps passé sur une page… autant d'événements pouvant être envoyés vers les topics de Kafka. Les applications " consumers " peuvent souscrire à ces topics et traiter les données à différentes fins : surveillance, analyse, rapports, flux d'actualité, personnalisation…

En outre, Apache Kafka est utilisé pour **les systèmes de logging et de monitoring**. Il est possible de publier des logs sur les topics Kafka, et ces logs peuvent être stockés sur un cluster Kafka durant un certain temps. Ils peuvent ensuite être agrégés et traités.

Il est possible de **construire des pipelines**, composés de plusieurs producteurs et consumers où les logs sont transformés d'une certaine manière. Ensuite, les logs peuvent être sauvegardés sur des solutions traditionnelles.

Si un système comporte **un composant dédié au monitoring**, ce composant peut lire les données à partir des topics Kafka. C'est ce qui rend cet outil utile pour le monitoring en temps réel.

## Quels sont les avantages de Kafka ?

L'utilisation de Apache Kafka apporte plusieurs avantages majeurs pour les entreprises. Cet outil est conçu pour **répondre à trois besoins spécifiques** : fournir un modèle de messagerie publish/subscribe pour la distribution et la consommation de données, permettre le stockage de données à long terme, et permettre l'accès et le traitement des données en temps réel.

C'est dans ces trois domaines que Kafka excelle. Bien que moins polyvalente que d'autres systèmes de messagerie, cette solution **se focalise sur la distribution et sur un modèle publish/subscribe** compatible avec le traitement en stream.

Par ailleurs, Apache Kafka rayonne par **ses capacités en persistance de données**, en tolérance aux fautes et répétabilité. Les données sont répliquées sur le cluster, et l'élasticité permet le partage de données sur les partitions pour l'augmentation des charges et volumes de données. Les topics et les partitions simplifient aussi l'accès aux données.

Conçu comme **une couche de communication** pour le traitement de log en temps réel, Apache Kafka convient naturellement aux applications de traitement stream en temps réel. Cet outil convient donc idéalement aux applications exploitant une infrastructure de communication capable de distribuer de hauts volumes de données en temps réel.

En **combinant les fonctionnalités de messagerie** et de streaming, Kafka délivre une capacité unique à publier, souscrire, stocker, et traiter des enregistrements en temps réel. Le stockage persistant de données sur un cluster permet une tolérance aux pannes.

Par ailleurs, cette plateforme permet de **déplacer les données rapidement et efficacement** sous forme d'enregistrements, de messages ou de streams. C'est la clé de l'interconnectivité, et c'est ce qui permet d'inspecter, de transformer et d'exploiter les données en temps réel.

Enfin, **le Connector API permet d'intégrer** de nombreuses solutions tierces, d'autres systèmes de messageries ou des applications anciennes par le biais de connecteurs ou d'outils open source. En fonction des besoins de l'application, différents connecteurs sont disponibles.

Découvrir Apache Kafka

# Quelles sont les limites de Kafka ?

Néanmoins, Kafka ne convient pas à toutes les situations. Cet outil n'est pas adapté au **traitement d'un petit volume de messages journaliers**. Il est conçu pour de larges volumes. Jusqu'à quelques milliers de messages par jour, des queues de messages traditionnelles comme RabbitMQ seront plus adéquates.

Par ailleurs, Kafka ne permet de **transformer facilement les données à la volée**. Il est nécessaire de construire un pipeline d'interactions complexe entre les producteurs et les consommateurs et de maintenir le système dans son intégralité. Ceci requiert beaucoup de temps et d'efforts. Il est donc préférable d'éviter cette solution pour les tâches d'ETL, surtout lorsqu'un traitement en temps réel est nécessaire.

Enfin, il n'est pas pertinent d'utiliser Kafka à la place d'une base de données. Cette plateforme n'est pas adaptée au stockage à long terme. Les données peuvent être conservées pendant une période spécifiée, mais cette période ne doit pas être trop longue. De plus, Kafka conserve des copies des données et les coûts de stockage s'en trouvent accrus. Il est préférable d'opter pour une base de données optimisée pour le stockage de données, compatible avec divers langages de requêtes et permettant l'insertion et la récupération de données.

# Comment apprendre à utiliser Apache Kafka ? Les formations DataScientest

La maîtrise de Kafka est **une compétence très recherchée en entreprise**, car beaucoup d'organisations ont aujourd'hui des besoins en traitement de données en temps réel. Par conséquent, apprendre à manier cet outil peut ouvrir de nombreuses portes.

Pour acquérir cette maîtrise, vous pouvez opter pour l**es formations DataScientest**. En effet, Apache Kafka est au coeur du module " Big Data Vitesse " de notre formation Data Engineer aux côtés de Spark Streaming.

Cette formation vous permet d'acquérir toutes les compétences et les connaissances requises pour devenir ingénieur des données. Les autres modules abordent la programmation, les bases de données, le Big Data Volume et l'automatisation.

Il s'agit d'une **formation accessible à Bac+3**. Elle peut être effectuée en neuf mois en Formation Continue, ou 11 semaines en Bootcamp. Comme tous nos cursus, l'approche Blended Learning combine présentiel et distanciel pour offrir le meilleur des deux mondes.

À l'issue du parcours, les apprenants reçoivent un diplôme certifié par l'Université de la Sorbonne. Parmi les alumnis, **93% trouvent un travail immédiatement** après la formation.

Le coût total s'élève à 5000€. Précisons que cette formation est **éligible au Compte Personnel de Formation**, et peut aussi être financée par Pôle Emploi via l'AIF. N'attendez plus, et découvrez notre cursus Data Engineer !