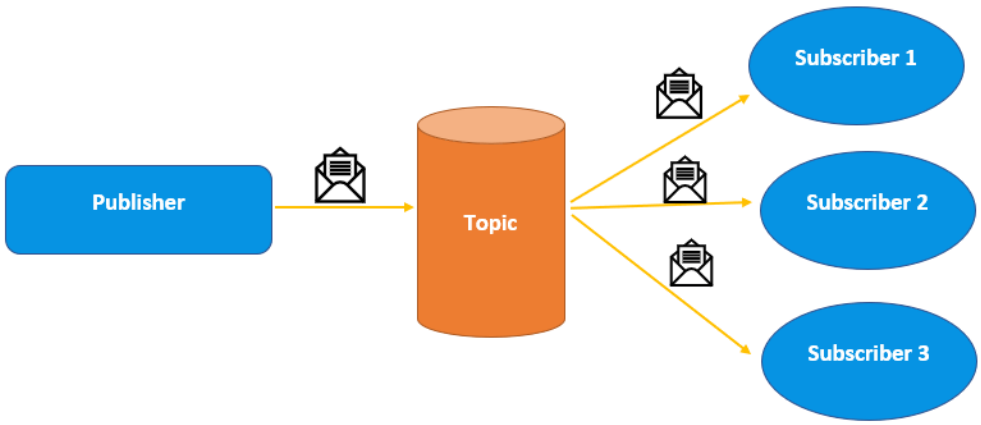
1. Presentation de Kafka

Kafka est un **système de messagerie distribué open source** dont le développement a débuté chez LinkedIn en 2009, et est maintenu depuis 2012 par la fondation Apache. Kafka est **tolérant aux pannes, très performant, hautement distribuable et adapté aux traitements batchs comme stream** .

Kafka a donc initialement été créé pour répondre aux importants besoins de LinkedIn et pour s'adapter au maximum à n'importe quel besoin. Kafka utilise le concept très connu du ***publish/subscribe***, mais en s'appuyant sur le disque dur pour fonctionner.

1.1. Publish/subscribe



Le "**publisher**" publie des messages et les envoie dans un "**topic**" .
Tous les "**subscriber**" qui se sont préalablement abonnés vis à vis ce topic reçoivent alors un exemplaire de ce message .

Vocabulaire de kafka :

Producer	Tout système/appli qui envoie/publie des données
Consumer	Tout système/appli qui lit des données
Broker	Serveur kafka
Cluster	Ensemble de brokers

Variantes d'utilisations :

Un message envoyé dans un topic_xy de kafka sera :
- soit reçus par plusieurs consumers s'ils sont de groupes différents
- soit reçus par un seul des consumers d'un même groupe

1.2. Structure d'un message "kafka"

MessageAndOffset

Champ	Taille	Description
MessageSize	32 bits	Taille du Message.
Offset	64 bits	Index unique de chaque message.
Message	bytes	Voir Objet Message.

Message

Champ	Taille	Description
CRC	32 bits	Contrôle de Redondance Cyclique : permet de vérifier l'intégrité d'un message au niveau du broker et du consumer.
MagicByte	8 bits	Identifiant de la version de Kafka utilisé pour la rétrocompatibilité.
Attributes	8 bits	Cet octet contient les métadonnées du message (codec de compression…).
Timestamp	64 bits	Timestamp du message.
KeySize	32 bits	Taille de la clé.
Key	bytes	La clé est optionnelle (peut être nulle) et est utilisée pour l'assignation à une partition.
ValueSize	32 bits	Taille du contenu.
Value	bytes	Contenu du message dans un byte-array « opaque ».

Remarques importantes :

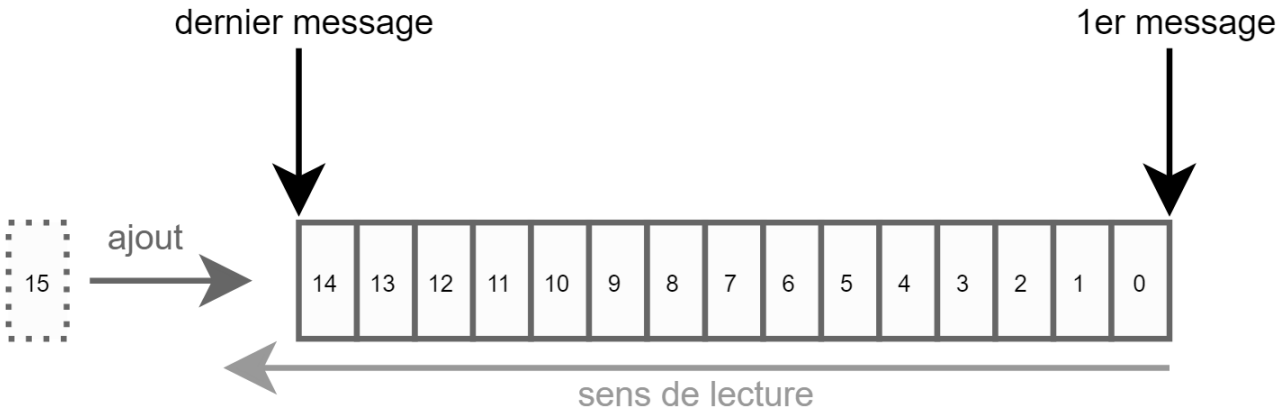
- **Kafka ne définit pas la façon dont sont formatés vos messages : un paquet de "byte" peut correspondre à** du **JSON**, du **XML** ou du binaire quelconque
- **Chaque message dispose d'un CRC permettant de vérifier son intégrité**.
- **Les messages peuvent être compressés de bout en bout.**
- **Chaque message est identifié par un Offset.**
- Avec kafka , la notion de clef (facultative) sert juste à garantir un ordre de réception cohérent si plusieurs partitions ...

1.3. Excellentes performances de kafka

La rapidité de kafka est essentiellement dû à deux aspects/astuces :

- accès séquentiels aux messages (selon la structure des logs)
- mécanisme "zero copy" (les messages ne font que transiter par kafka , exemple : java.nio.channels.FileChannel.transfertTo(position, count, writableChannel) en java)

1.4. Log kafka



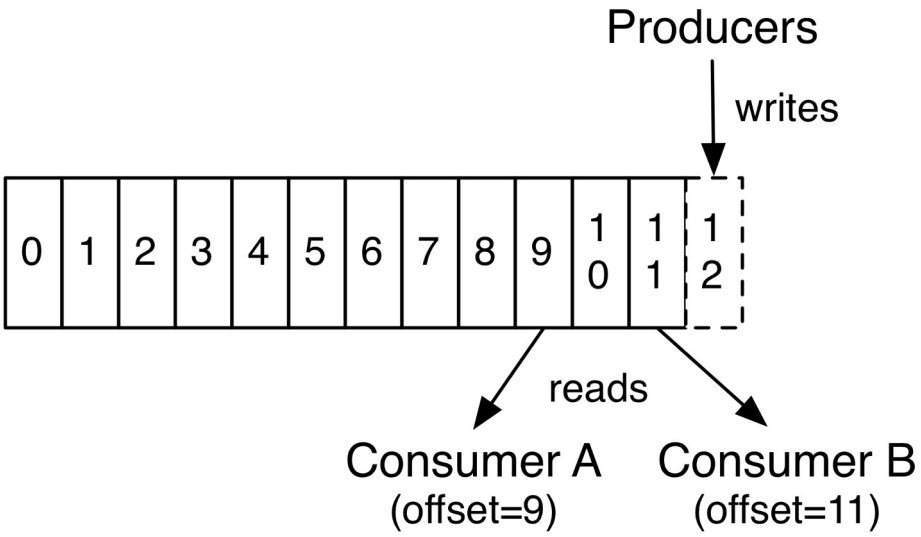
Les messages gérés par kafka sont stockés sur le disque dur dans une structure hôte appelée "**log**" (au sens "file/queue") .

Un log kafka peut être vu comme un tableau de messages ordonnés selon l'ordre de réception au niveau du broker.

En lecture :

- si le consumer précise un offset, il récupère tous les messages qui sont arrivés depuis cet offset dans le log ;
- si le consumer ne donne pas d'offset (ou bien offset à zéro) il récupère l'intégralité des messages.

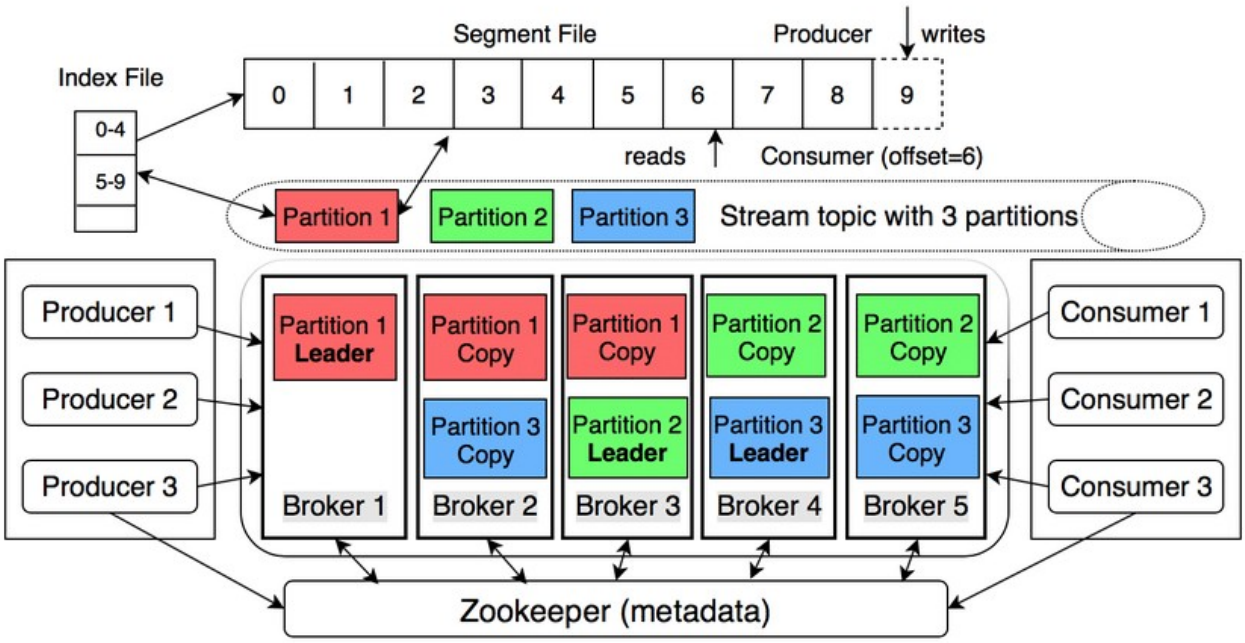
Contrairement à d'autres brokers de messages, Kafka ne maintient pas une liste des consumers et de leur avancement. **Chaque consumer est donc responsable de son avancement dans la lecture des messages** ; c'est à lui de faire attention à ne pas lire un message en double par exemple.

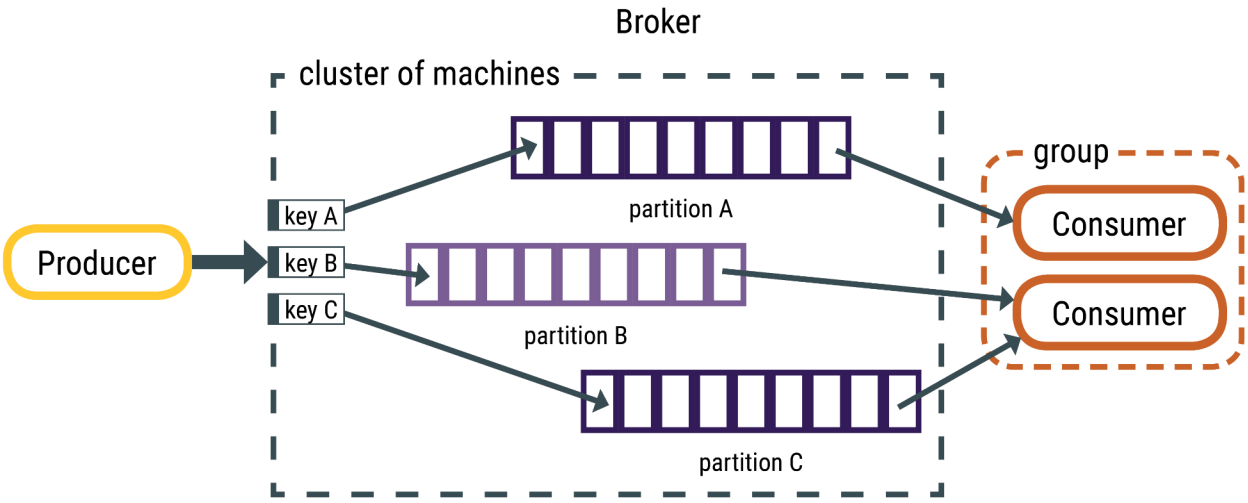


1.5. Topic, partitions et cluster

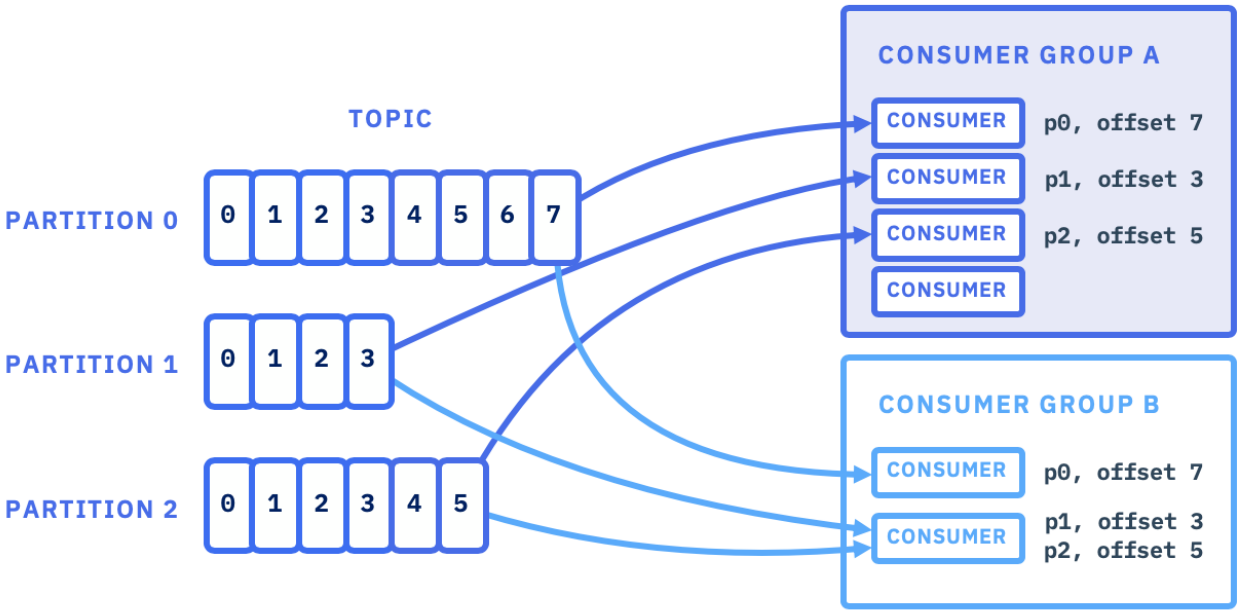
Un topic a un nom qui représente une catégorie de données. C'est un élément logique (assez abstrait)

Un topic peut être subdivisé en partitions
Lorsqu'un ensemble de brokers kafka fonctionnent en cluster , chaque serveur détient l'original d'une partition (il en est "maître") et certains autres membres du cluster vont gérer passivement des réplicas (pour garantir une bonne tolérance aux pannes) . En cas de panne d'un serveur , le nouveau maître d'une partition peut changer.





NB : Chaque consumer n'ira lire que dans la partition qui lui aura été assignée. Les messages avec la même clé iront donc tous dans la même partition/le même consumer.



NB: Un message envoyé dans un topic_xy de kafka sera :

- soit reçus par plusieurs consumers s'ils sont de groupes différents
- soit reçus par un seul des consumers d'un même groupe