



10 min \Rightarrow 1 composant Confluent

10 minutes pour implémenter un stockage infini des données Kafka, à coûts maîtrisés, avec Tiered Storage



01 - Tiered (Infinite) Storage



Confluent Platform 5.5 (GA in 6.0)

Operations and Security

Security plugins | Role-Based Access Control

Control Center | Replicator | Auto Data Balancer | Operator

Audit Logs | Schema Validation | Tiered Storage | Multi-Region Cluster

Development & Stream Processing

Connectors

Clients | REST Proxy
MQTT Proxy | Schema Registry

KSQLDB

Apache Kafka

K-Connect

Continuous Commit Log

K-Streams

**Mission-critical
Reliability**

Complete

**Event Streaming
Platform**

Self-Managed Software

Fully-Managed Service

Freedom of Choice

Datacenter

Public Cloud

Confluent Cloud

Notions de persistance dans Kafka



- **Stockage des données**

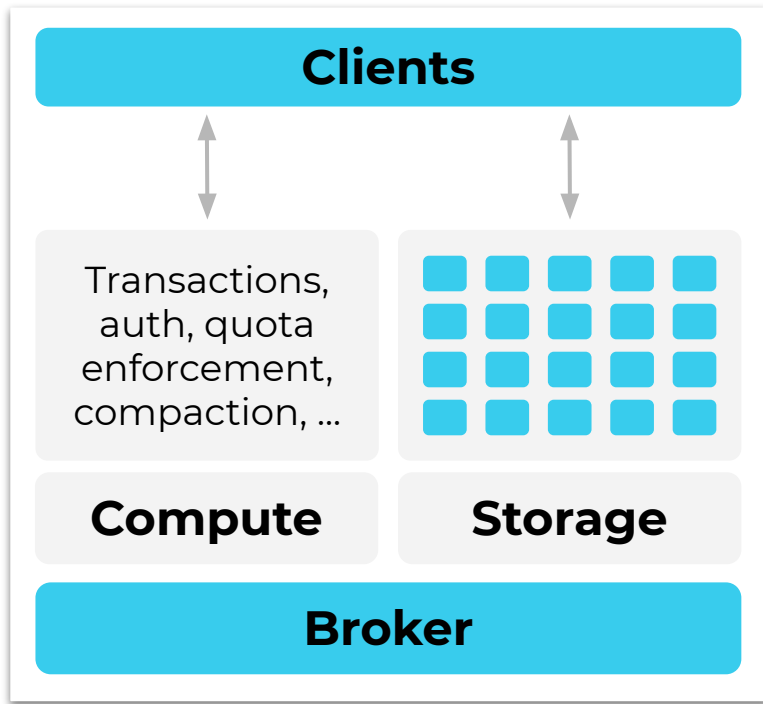
- Configurée par Topic / Broker
- 1 semaine par défaut
- Un cas d'usage nécessitant de garder des données sur longue période implique généralement d'augmenter le nombre de brokers, donc de serveurs, donc les coûts

⇒ Découplage Compute & Storage = **Tiered Storage**

Retain infinite Data on Kafka

With Apache Kafka, all data needs to be stored on the broker, meaning that customers requiring long data retention periods must pay **significant costs to buy additional hardware**.

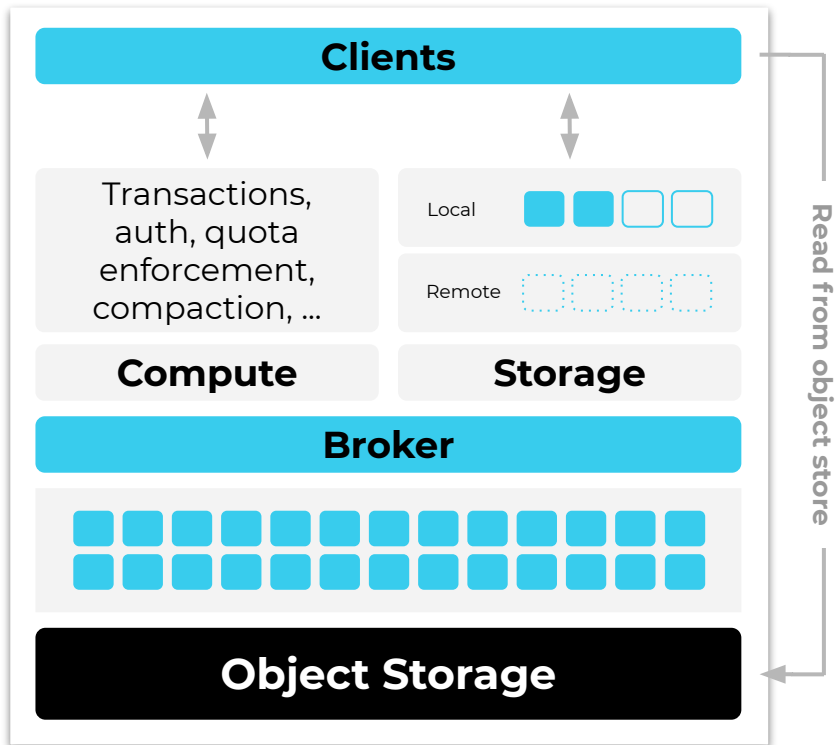
Standard storage on Kafka (no Tiered Storage)



Retain infinite Data on Kafka

- **Save significant costs** by with inexpensive object storage
- **Playback historical data** directly from Kafka topics
- **Meet regulatory compliance** requirements

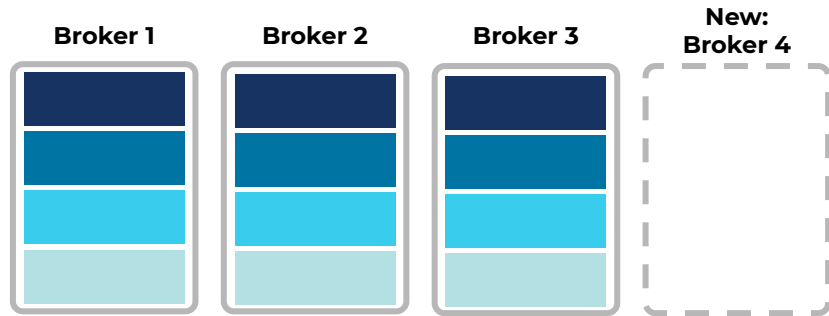
Tiered Storage (GA in 6.0)



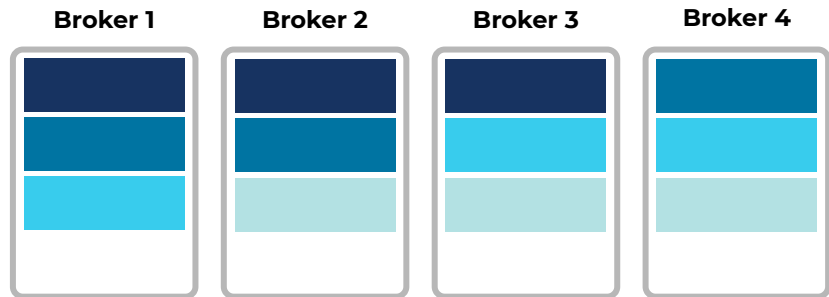
Elastically scale Kafka clusters

With Apache Kafka, all data needs to be stored directly on the broker, resulting in **longer rebalancing time after adding new brokers.**

Adding brokers with Kafka (no Tiered Storage)



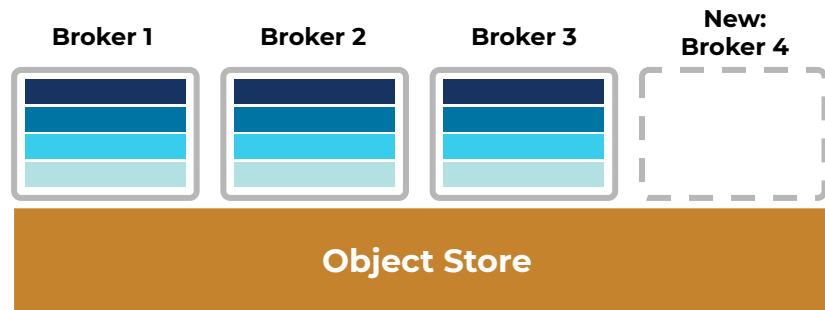
*Lengthy, manual process for
reassigning large topic partitions*



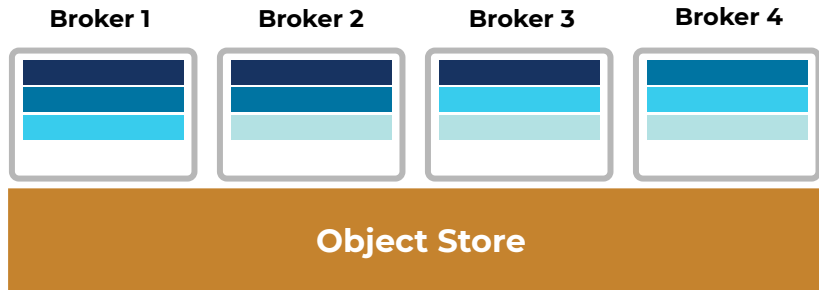
Elastically scale your cluster

With Tiered Storage, less data needs to be stored on the broker, allowing clusters to **scale storage and compute resources independently and without lengthy rebalances.**

Adding brokers with Tiered Storage



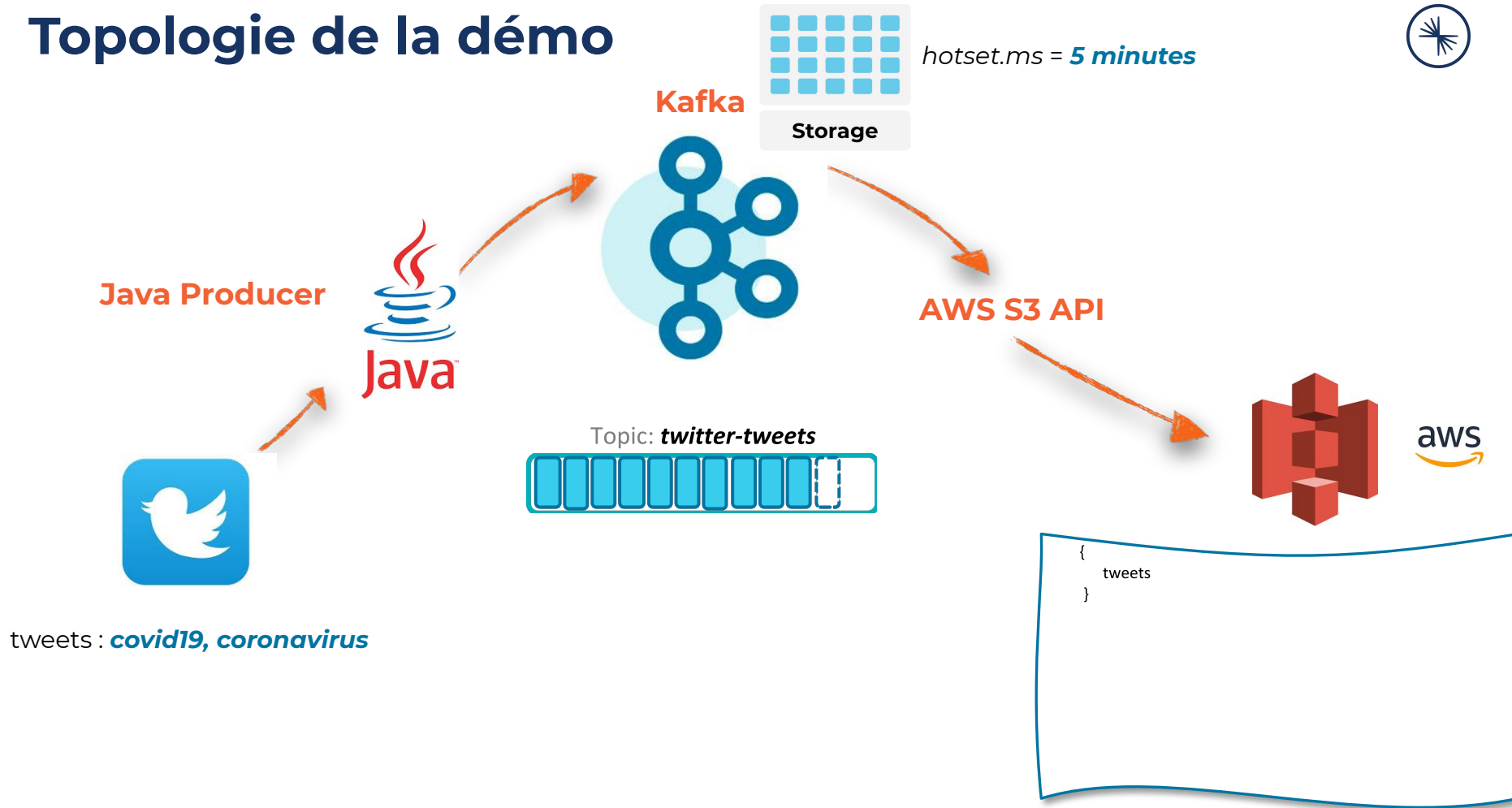
*Fast, automated process for
reassigning small topic partitions*





02 - Implémentation (AWS S3)

Topologie de la démo



Remarques : Configuration



- **Tiered Storage Configuration (server.properties) :**

Tiered Storage to S3 (AWS Keys are in Env variables)

confluent.tier.feature=**true**

confluent.tier.enable=**true**

confluent.tier.backend=**S3**

confluent.tier.s3.bucket=confluent-ola-s3

confluent.tier.s3.region=eu-west-3

confluent.tier.s3.aws.access.key.id=**Your-S3-Id**

confluent.tier.s3.aws.secret.access.key=**Your-S3-Key**

Once a segment file reaches 1 Mo, a new one is created

kafka.log.segment.bytes=**1048576**

Once a segment is closed, it is pushed to S3. After hotset.ms, the segment is locally removed. Here 5 minutes.

confluent.tier.local.hotset.ms=**300000**

Check every 10 min for topic deletion

confluent.tier.topic.delete.check.interval=600000

confluent.tier.metadata.replication.factor=1

600 min = 10H

log.retention.ms=36000000

Pré-requis et informations



- **Téléchargez et installez Confluent Platform 5.5 / 6.0**

- Téléchargement : <https://www.confluent.io/download>
- Documentation : <https://docs.confluent.io/current/getting-started.html>
- Pour la démo : Confluent 5.5 installé on-prem
- Accès à un environnement AWS S3 ou GCP GCS (S3 utilisé pour la démo)

- **Pour produire les messages JSON**

- Option 1 : `kafka-producer-perf-test --topic twitter-tweets --num-records 5000000 --record-size 5000 --throughput -1 --producer-props acks=all bootstrap.servers=localhost:9092 batch.size=8196`
- Option 2 : utilisez un programme pour charger des messages en masse
 - Classe `TwitterProducer` (pour cette démo)

- **Stockage distant**

- En version 6.0, TieredStorage supporte AWS S3 et GCP GCS. D'autres solutions de stockage viendront les compléter par la suite (certification de vendeurs / API S3).

Accédez aux éléments de la démo sur GIT



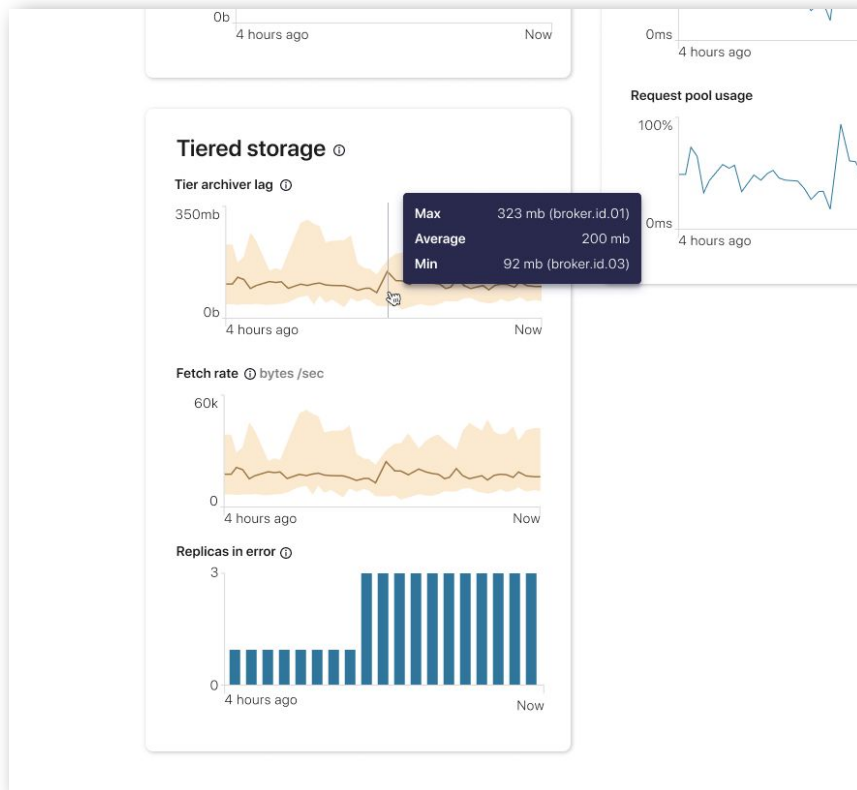
- **Pour cette démo spécifiquement :**

<https://github.com/olaplace/Confluent10Minutes/tree/master/TieredStorage>

Remarques : Fonctionnement



- Les consommateurs peuvent constater une latence légèrement plus importante mais l'impact sur le débit (throughput) est négligeable.
- Suivre l'évolution "Archiver lag" qui peut indiquer la nécessité d'augmenter le cluster.
- Les topics "compacted" ne sont pas supportés.
- Peut être configuré au niveau Cluster ou par Topic





03 - Cas d'usage et valeur

Cas d'usage



- Tableaux de bord comparatifs : trimestriels, semestriels, annuels
- Machine Learning au fil de l'eau
- Utiliser Kafka comme System Of Record
- IoT / Analytics

VALEUR :

Ouvrir Kafka à de nouveaux cas d'usage tout en maîtrisant les coûts.
Réduire les coûts de stockage.

Réduire les temps opérationnels d'ajout / retrait de brokers.

⇒ **Utiliser Kafka comme System of Record, sans augmenter considérablement les coûts inhérents au stockage.**



Merci !

olaplace@confluent.io

<https://developer.confluent.io/>

<https://www.confluent.io/blog/>

<https://docs.confluent.io/current/>