Comparatif & Procédures : Kafka Connect vs Kafka MirrorMaker 2 (MM2)

Rapport consolidé

Date : 2025-10-27

Introduction

Dans l’écosystème Kafka, on confond souvent **Kafka Connect** — cadre d’intégration entre systèmes externes et Kafka — et **Kafka MirrorMaker 2 (MM2)** — solution de réplication *cluster → cluster*. Ce rapport clarifie les rôles, recense les cas d’usage, fournit des procédures pas à pas et compare la facilité de mise en place.

TL;DR

- Intégrations avec des systèmes externes (DB, S3, SaaS, analytiques) → **Kafka Connect**.
- Réplication de topics entre clusters (DR, migration, multi■région) → **MirrorMaker 2**.
- MM2 tourne sur le framework Connect : ils coexistent souvent.

1) Rôle & périmètre

Critère	Kafka Connect	Kafka MirrorMaker 2 (MM2)
Objectif	Intégrer systèmes externes à Kafka (source/sink).	Réplication de topics entre clusters Kafka.
Base technique	Framework Connect (workers, tasks, REST, SMT).	Implémenté via Connect (connecteurs Mirror*).
Flux	Externe ↔ Kafka.	Kafka A ↔ Kafka B.
Transformations	SMT légères, Schema Registry.	Renommage, checkpoints offsets.
Garanties	Idempotence/transactions selon connecteur.	Pas d'Exactly■Once inter■cluster.
Gouvernance	Intégration naturelle Schema Registry.	SR géré séparément (non répliqué).
Cas phares	CDC, ETL/ELT, data lake, recherche.	DR/BCP, migration, multi■région, agrégation.

2) Cas d’usage

Kafka Connect

- CDC (Debezium) : MySQL/Postgres/Oracle → Kafka → DWH/S3/Elastic.
- Ingestion fichiers/objets : S3/GCS/ADLS/FTP → Kafka.
- Diffusion : Kafka → Snowflake/BigQuery/Elasticsearch/ClickHouse.
- Transformations légères : normalisation et enrichissement via SMT.

MirrorMaker 2

- Disaster Recovery : primaire → secondaire, bascule contrôlée.
- Migration de cluster : on■prem → cloud / upgrade sans coupure.
- Multi■région : proximité des lecteurs, hub & spoke.
- Lecture cross■cluster : consommateurs locaux sur données distantes.

3) Procédures de mise en place

3.1) Kafka Connect — Procédure

Prérequis

- Cluster Kafka accessible (KRaft ou ZooKeeper).
- Java 11+ (si hors Docker).
- Droits de créer des topics (ou auto■create activé).
- Schema Registry recommandé si Avro/Protobuf.

Étapes

A) Préparer le worker (mode distribué)

```
# connect-distributed.properties (extrait)
bootstrap.servers=kafka:9092
group.id=connect-cluster
config.storage.topic=connect-configs
offset.storage.topic=connect-offsets
status.storage.topic=connect-status
key.converter=io.confluent.connect.avro.AvroConverter
value.converter=io.confluent.connect.avro.AvroConverter
key.converter.schema.registry.url=http://schema-registry:8081
value.converter.schema.registry.url=http://schema-registry:8081
plugin.path=/usr/share/java,/opt/connectors

# Lancer le worker
connect-distributed.sh connect-distributed.properties
```

B) Installer les connecteurs (JAR dans plugin.path), redémarrer si besoin.

C) Créer un connecteur (ex. Debezium Postgres)

```
curl -X POST http://connect:8083/connectors -H "Content-Type: application/json" -d '{
  "name": "debezium-postgres-orders",
  "config": {
    "connector.class": "io.debezium.connector.postgresql.PostgresConnector",
    "tasks.max": "2",
    "database.hostname": "pg",
    "database.port": "5432",
    "database.user": "debezium",
    "database.password": "******",
    "database.dbname": "app",
    "database.server.name": "pgapp",
    "table.include.list": "public.orders",
    "tombstones.on.delete": "false",
    "errors.tolerance": "all",
    "errors.deadletterqueue.topic.name": "dlq.debezium",
```

```
    "errors.deadletterqueue.context.headers.enable": "true"
  }
}'
```

D) Vérifier le statut (RUNNING), consommer le topic, surveiller la DLQ.

E) Durcissement : sécurité (SASL/SSL), observabilité (JMX/logs), capacity planning, gouvernance SR.

3.2) MirrorMaker 2 — Procédure

Prérequis

- Deux clusters Kafka joignables (primary/secondary).
- Réseau et DNS/SSL configurés.
- Droits : lecture sur source et écriture sur cible.
- Réplication des ACLs à traiter séparément si nécessaire.

Option A — Script MirrorMaker 2

```
# mm2.properties
clusters = primary, secondary
primary.bootstrap.servers=primary-kafka:9092
secondary.bootstrap.servers=secondary-kafka:9092

primary->secondary.enabled=true
primary->secondary.topics=.*
# ou : primary->secondary.topics=^(orders|payments)\..*

replication.policy.class=org.apache.kafka.connect.mirror.DefaultReplicationPolicy
emit.checkpoints.enabled=true
emit.heartbeats.enabled=true

# Sécurité (exemples)
# primary.security.protocol=SASL_SSL
# primary.sasl.jaas.config=org.apache.kafka.common.security.plain.PlainLoginModule requir
# secondary.security.protocol=SASL_SSL
# secondary.sasl.mechanism=PLAIN

# Lancer
connect-mirror-maker.sh mm2.properties
```

Option B — Connect classique : déployer 3 connecteurs (MirrorSource, MirrorCheckpoint, MirrorHeartbeat) via REST.

Vérifier : topics préfixés sur la cible, topics internes (heartbeats/checkpoints), latence et lag.

Durcissement : préfixes, éviter les clés en conflit en active■active, sécurité des deux côtés, monitoring.

4) Monitoring & opérations

- **Connect** : task-error-rate, deadletter, source-record-poll, sink-send-rate.
- **MM2** : replication-latency-ms, records-lag, topics heartbeats/checkpoints.
- **Capacity planning** : partitions × débit × latence (WAN pour MM2).
- **Sécurité** : SASL/SSL de bout en bout ; ACLs gérées hors MM2.

5) Comparaison — Facilité de mise en place

Critère	Kafka Connect	MirrorMaker 2
Hello world	Worker + plugin + conf REST	Un fichier properties + script
Dépendances	Drivers DB/creds/SR + permissions externes	Deux clusters Kafka uniquement
Sécurité	Kafka + systèmes externes	Kafka (source et cible)
Schémas	Intégration native Schema Registry	À gérer séparément
Exploitation	Connecteurs hétérogènes	Usage homogène (lag/latence)
Échelle	Tasks/workers (varie par connecteur)	Par partitions (simple unidirectionnel)
Global	Plus complexe au démarrage	Plus simple pour réplication cluster→cluster

Notation rapide (1 = très simple, 5 = complexe)

• Mise en place initiale : Connect 3–4/5 ; MM2 2/5

• Durcissement prod : Connect 3–4/5 ; MM2 3/5

• Exploitation courante : Connect 3/5 ; MM2 2–3/5

6) Checklists rapides

Kafka Connect

- Worker distribué opérationnel (API REST OK).
- plugin.path contient connecteurs et drivers.
- Connecteur en RUNNING.
- DLQ et tolérance aux erreurs configurées.
- Schema Registry + compatibilité définie.
- SASL/SSL + observabilité (JMX, logs).

MirrorMaker 2

- Deux clusters joignables avec sécurité.
- mm2.properties OK (topics/policy/heartbeats/checkpoints).
- Topics miroir présents sur cible (.).
- Lag/latence sous contrôle ; heartbeats visibles.
- Stratégie Schema Registry & ACLs hors MM2.

Conclusion

Kafka Connect est le couteau suisse d’intégration avec les systèmes externes ; MirrorMaker 2 est l’outil de réplication inter■cluster pour DR, migration et multi■région. Ils se complètent : déployez Connect pour l’ingestion/sortie et MM2 pour la géo■réplication, avec une stratégie claire pour les offsets, les schémas et la sécurité.