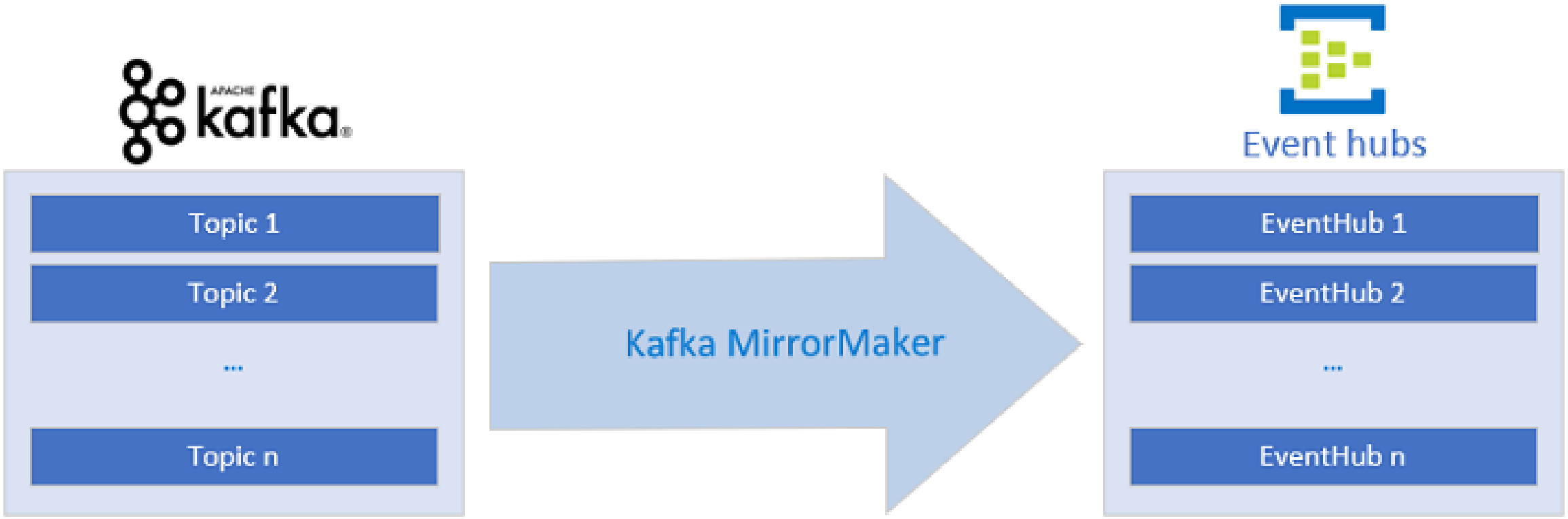
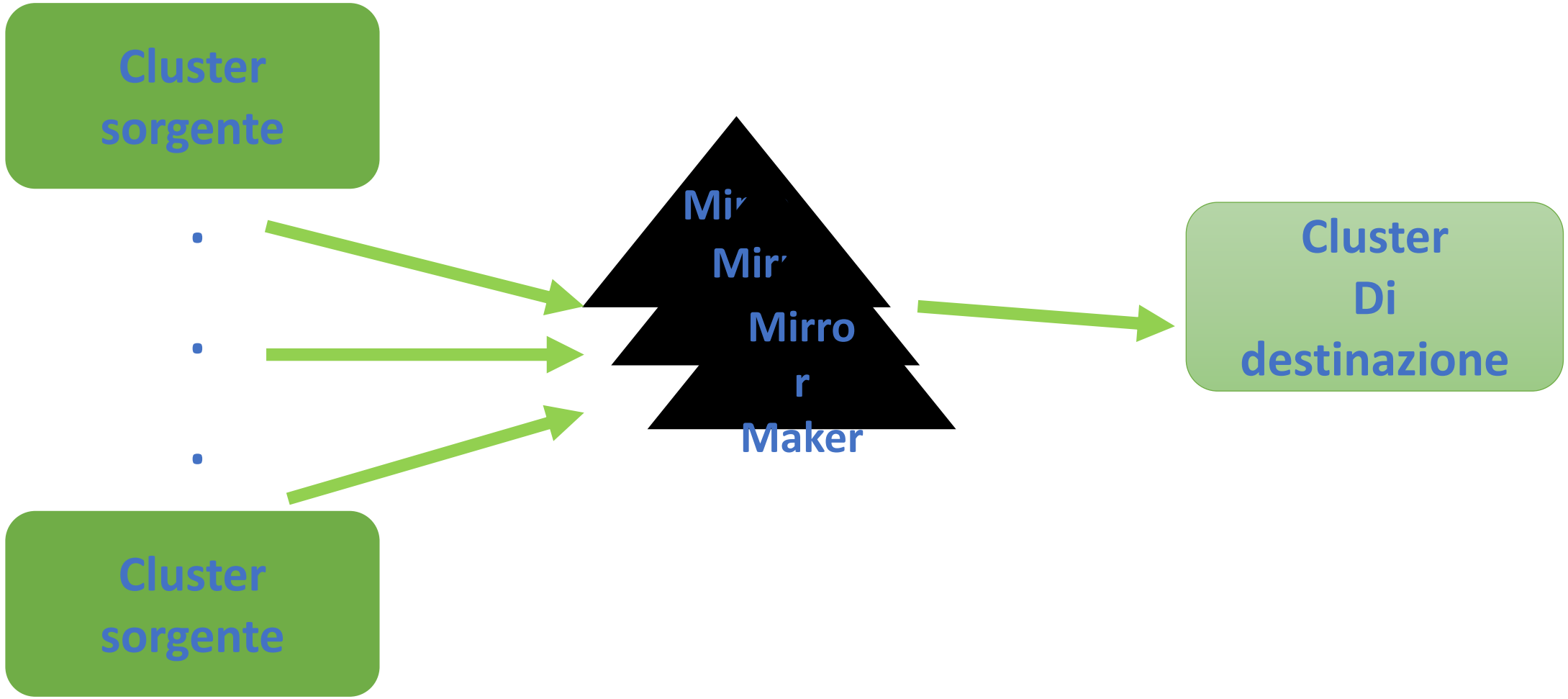

APACHE
kafka®

Componente Mirror Maker per Kafka



- Il processo di replica dei dati tra i **Cluster Kafka** è chiamato **Mirroring**
- Questo processo è utilizzati per differenziare la replica tra **Cluster**
- Un utilizzo comune per il **Mirroring** è quello di mantenere una copia separata di un **Cluster Kafka** in un altro **Data Center**
- Lo strumento **MirrorMaker** di **Kafka** legge i dati dai **Topics** in uno o più **Cluster Kafka** di origine e scrive i corrispondenti **Topics** in un **Cluster Kafka** di destinazione (utilizzando gli stessi nomi dei **Topics**)


Cluster
sorgente
Mir
Mirr
Mirro
r
Maker
Cluster
Di
destinazione
Cluster
sorgente

- Per eseguire il **Mirroring** di più **Cluster** di origine, sarà necessario avviare almeno una Istanza di **MirrorMaker** per ogni **Cluster** di origine

- Sarà inoltre possibile utilizzare più processi **MirrorMaker** per rispecchiare **Topics** all'interno dello stesso **Consumer Group**

- Questo può aumentare **Throughput** (quantità di consegna di dati) e migliorare la tolleranza ai guasti (**Fault Tolerance**)

- Se un processo termina, saranno altri ad assumersi il carico aggiuntivo

- I **Cluster** di origine e di destinazione sono completamente indipendenti, quindi possono avere un numero diverso di partizioni e **Offset** diversi

- Il **Cluster** di destinazione (**Mirror**) non è inteso come un meccanismo per il **Fault Tolerance** in quanto la posizione del **Consumer** sarà diversa

- Si fa presente che il processo **MirrorMaker**, tuttavia, manterrà e utilizzerà il **Message Key for Partitioning**, preservando l'ordine in base alla chiave

- Per il **Fault Tolerance** si consiglia di utilizzare lo standard all'interno del **Cluster** (**Replica**)

- Prerequisito essenziale per l'esecuzione di **MirrorMaker** è che i **Cluster** di origine e di destinazione siano distribuiti ed eseguiti

- Per configurare un **Mirror**, eseguire **kafka.tools.MirrorMaker**

- Nella tabella seguente, verranno elencate le opzioni di configurazione

- **MirrorMaker** richiede uno o più file di configurazione del **Consumer**, un file di configurazione del **Producer** e una **Whitelist** o una **Blacklist** di **Topics**

- Nei file di configurazione del **Consumer** e del **Producer**, indirizzare il **Consumer** al processo **ZooKeeper** sul **Cluster** di origine e indirizzare il **Producer** al processo **ZooKeeper** sul **Cluster** di destinazione (**Mirror**)

Parametro	Descrizione	Esempio
--consumer.config	Specifica un file che contiene la configurazione e le impostazioni per il Cluster di origine	--consumer.config hdp1-consumer.properties
--producer.config	Specifica il file che contiene la configurazione e le impostazioni per il Cluster di destinazione	--producer.config hdp1-producer.properties

Parametro	Descrizione	Esempio
--whitelist/--blacklist	Anche se facoltativo, per un Mirror parziale, si può specificare esattamente un elenco separato da virgole di Topics da includere (--whitelist) o escludere (--blacklist); in generale, queste opzioni accettano modelli Java regex	--whitelist my-topic
--num.streams	Specifica il numero di CST (Consumer Stream Threads) da creare	--num.streams 4
--num.producers	Specifica il numero di Istanze del Producer; impostandolo su un valore maggiore di uno, stabilisce un Pool di Producer che può incrementare il Throughput	--num.producers 2
--queue.size	Numero di messaggi che sono bufferizzati, in termini di numero di messaggi tra Consumer e Producer; il Default è 10000	--queue.size 2000

- Una virgola viene interpretata come simbolo di scelta dell'espressione regolare ('|') per comodità

- Se si specifica **--white-list=".*"**, **MirrorMaker** tenta di recuperare i dati dal **Topic** a livello di sistema **__consumeroffsets** e di produrre tali dati nel **Cluster** di destinazione

- Tutto questo però può causare il seguente errore

Producer cannot send requests to __consumer-offsets

Workaround: Specify topic names, or to replicate all topics, specify --blacklist="__consumer-offsets".

- L'esempio seguente replica **topic1** e **topic2** da **sourceClusterConsumer** a **targetClusterProducer**

```
/usr/hdp/current/kafka-broker/bin/kafka-run-class.sh
kafka.tools.MirrorMaker --consumer.config
sourceClusterConsumer.properties --producer.config
targetClusterProducer.properties --whitelist="topic1, topic"
```

- Il file di configurazione del **Consumer** deve specificare il processo **ZooKeeper** nel **Cluster** di origine

Guarda esempio file di configurazione del Consumer

```
zk.connect=hdp1:2181/kafka
zk.connectiontimeout.ms=1000000
consumer.timeout.ms=-1
groupid=dp-MirrorMaker-test-datap1
shallow.iterator.enable=true
mirror.topics.whitelist=app_log
```

- La configurazione del **Producer** dovrebbe puntare al processo **ZooKeeper** del **Cluster** di destinazione o utilizzare il parametro **broker.list** per specificare un elenco di **Broker** nel **Cluster** di destinazione

Guarda esempio file di configurazione del Producer

```
zk.connect=hdp1:2181/kafka-test
producer.type=async
compression.codec=0
serializer.class=kafka.serializer.DefaultEncoder
max.message.size=10000000
queue.time=1000
queue.enqueueTimeout.ms=-1
```

- È possibile utilizzare lo strumento da riga di comando **Consumer Offset Checker** di **Kafka** per valutare lo stato del **Cluster** di origine e del **Mirror**

- Il **Consumer Offset Checker** controlla il numero di messaggi letti e scritti e segnala il ritardo per ogni **Consumer** in un **Consumer Group** specificato

- Il comando seguente esegue **Consumer Offset Checker** per il gruppo **KafkaMirror**, **topic** t**est-topic**

- L'argomento **--zkconnect** punta all'**Host ZooKeeper** e alla porta sul **Cluster** di origine

```
/usr/hdp/current/kafka/bin/kafka-run-class.sh
kafka.tools.ConsumerOffsetChecker --group KafkaMirror --
zkconnect
Source-cluster-zookeeper:2181 --topic test-topic



Group           Topic        Pid  Offset    logSize    Lag
Owner
-----------------------------------------------------------
------- KafkaMirror    test-topic    0         5           5
0       none
KafkaMirror    test-topic    1         3           4       1
none
```

Guarda le opzioni di controllo Offset del Consumer

Comando	Descrizione
--group	Specifica il **Consumer Group**
--zkconnect	Specifica la stringa di connessione **ZooKeeper**; il **Default** è **localhost:2181**
--broker-info	Elenca le informazioni del **Broken**
--help	Elenca le opzioni **Offset Checker**
--topic	Specifica un elenco separato da virgole di **Consumer Topics**; se non si specificano **Topics**, l'**Offset Checker** visualizzerà le informazioni per tutti i **Topics** sotto il **Consumer Group** specificato

- Se per qualche motivo il **Producer** non può consegnare messaggi che sono stati elaborati e impegnati dal **Consumer**, è possibile che un processo **MirrorMaker** perda i dati

- Per prevenire la perdita di dati, utilizzare le seguenti impostazioni (**Default**)

Consumer

- `auto.commit.enabled=false`

Producer

- `max.in.flight.requests.per.connection=1`
- `retries=Int.MaxValue`
- `acks=-1`
- `block.on.buffer.full=true`

- Specificare l'opzione **--abortOnSendFail** su **MirrorMaker**

- **MirrorMaker** intraprenderà le seguenti azioni
  - **MirrorMaker** invierà una sola richiesta ad un **Broker** alla volta
  - Se viene rilevata una eccezione nel **Thread MirrorMaker**, **MirrorMaker** proverà a confermare gli **Offset** confermati e quindi uscire immediatamente
  - In una **RetriableException** nel **Producer**, il **Producer** riproverà a tempo indeterminato e se il tentativo non funziona, **MirrorMaker** si interromperà alla fine quando il **Buffer** del produttore è pieno
  - In una eccezione non ripristinabile, se viene specificato **--abort.on.send.fail**, **MirrorMaker** verrà interrotto

- Se **--abort.on.send.fail** non è specificato, il meccanismo di **Callback** del **Producer** registrerà il messaggio che non è stato inviato

- In questo caso, **MirrorMaker** continuerà a funzionare ed il messaggio non verrà replicato nel **Cluster** di destinazione

- Per eseguire **MirrorMaker** su un **Cluster** abilitato per **Kerberos**/**SASL**, configurare le proprietà **Producer** e **Consumer** nel modo seguente

- Scegliere o aggiungere un nuovo "**Principal**" per **MirrorMaker** non utilizzando ne **Kafka** ne altri **Service Account**

- Il seguente esempio usa il "**Principal**" **MirrorMaker**; creare **Keytab Kerberos** lato **client** per **MirrorMaker**

```
sudo kadmin.local -q "ktadd -k /tmp/mirrormaker.keytab
mirrormaker/
HOSTNAME@EXAMPLE.COM"
```

- Aggiungere un nuovo file di configurazione **Jaas** al nodo in cui si prevede di eseguire **MirrorMaker**

```
-Djava.security.auth.login.config=/usr/hdp/current/kafka-broker/config/
kafka_mirrormaker_jaas.conf
```

- Aggiungere le seguenti impostazioni alla sezione **KafkaClient** del nuovo file di configurazione **Jaas** ed assicurarsi che il «**Principal**» dispone delle autorizzazioni sia sul **Cluster** di origine che sul **Cluster** di destinazione

```
KafkaClient {
 com.sun.security.auth.module.Krb5LoginModule required
 useKeyTab=true
 keyTab="/tmp/mirrormaker.keytab"
 storeKey=true
 useTicketCache=false
 serviceName="kafka"
 principal="mirrormaker/HOSTNAME@EXAMPLE.COM";
 };
```

- Eseguire il comando **ACL** sui **Cluster Kafka** di origine e destinazione

```
bin/kafka-acls.sh
--topic test-topic
--add
--allow-principal user:mirrormaker
--operation ALL
--config /usr/hdp/current/kafka-
broker/config/server.properties
```

- Nei file **MirrorMaker** **consumer.config** e **producer.config**, specificare **security.protocol=SASL_PLAINTEXT**

- Avviare **MirrorMaker** e specificare l'opzione **new.consumer** oltre alle altre opzioni; di seguito un breve esempio

```
/usr/hdp/current/kafka-broker/bin/kafka-run-class.sh
 kafka.tools.MirrorMaker
--consumer.config consumer.properties
--producer.config target-cluster-producer.properties
--whitelist my-topic
--new.consumer
```