Cloudera Runtime 7.3.2

# Planning for Streams Replication Manager

**Date published: 2019-08-22**
**Date modified: 2026-03-31**

**https://docs.cloudera.com/**

# Legal Notice

# Contents

# Streams Replication Manager requirements

When planning and designing your Streams Replication Manager deployment that has multiple instances of the Streams Replication Manager service, there are a number of practices that you must follow. Otherwise, you might encounter issues when you are replicating data with Streams Replication Manager.

**Kafka cluster aliases must be uniform across clusters**

When you have multiple Streams Replication Manager services, the cluster aliases must be defined identically in all instances. Streams Replication Manager utilizes the cluster alias to detect replication loops.

**The heartbeats topic must be present on all source clusters**

To be able to properly monitor and discover replications, the heartbeats topic must be present on all source clusters. The heartbeats topic is created by the Streams Replication Manager Driver when a target cluster is set with the Streams Replication Manager Driver Target Cluster property and when heartbeating is enabled for the replication. This means that to ensure that the heartbeats topic is created, and has data written into it periodically, at least a single Streams Replication Manager Driver cluster must target each source cluster.

**Driver targets must be managed correctly**

The Streams Replication Manager Cluster alias and the Streams Replication Manager Driver Target Cluster properties must be managed correctly when you have multiple Streams Replication Manager services. If a target is specified for the replication in the Streams Replication Manager Driver Target Cluster property, a Connect worker is created for each possible cluster pair based on the aliases present in Streams Replication Manager Cluster alias. As a result of this, it is possible for different Streams Replication Manager Driver clusters to merge through their Connect workers. To avoid this, ensure that different Streams Replication Manager Driver clusters target different Kafka clusters.

**Service targets must be managed correctly**

The Streams Replication Manager Cluster alias and the Streams Replication Manager Service Target Cluster properties must be managed correctly when you have multiple Streams Replication Manager services. If a target is specified for monitoring in the Streams Replication Manager Service Target Cluster property, a Kafka Streams application instance is created for each cluster present in the Streams Replication Manager Service Target Cluster property. As a result of this, it is possible for different Streams Replication Manager Service clusters to merge through their Kafka Streams application instances. To avoid this, ensure that different Streams Replication Manager Service clusters target different Kafka clusters.

**All Streams Replication Manager services must use the same replication policy**

Streams Replication Manager ships with and supports multiple replication policies. Using a mix of these policies in a single deployment is not supported. Decide which replication policy you want to use and ensure that all Streams Replication Manager services are configured to use your chosen policy. In addition transitioning from one policy to another is not recommended.

**Ensure that you do not create replication loops if using the IdentityReplicationPolicy**

The `IdentityReplicationPolicy` cannot detect replication loops. As a result, if you are planning to create a setup with bidirectional replication using the `IdentityReplicationPolicy`, you must ensure that you do not create any replication loops when you set up your allowlists and denylists (topic filters) with srm-control.

# Streams Replication Manager recommended deployment architecture

Learn about pull mode, which is the Cloudera-recommended deployment architecture for Streams Replication Manager.

While Streams Replication Manager can be deployed in many different ways, the Cloudera recommended setup is pull mode. Pull mode refers to a Streams Replication Manager deployment where data replication happens by pulling data from remote source clusters, rather than pushing data into remote target clusters. A Streams Replication Manager deployment that is in pull mode conforms to the following:

- An instance of the Streams Replication Manager service (Driver and Service roles) is deployed on all clusters that host a target Kafka cluster. In other words, each target Kafka cluster in the deployment has a co-located Streams Replication Manager service.
- All instances of the Streams Replication Manager service use the same replication policy.
- Streams Replication Manager Driver roles only execute replications that target their co-located Kafka cluster.
- Streams Replication Manager Service roles only target their co-located Kafka cluster.
- If using the `IdentityReplicationPolicy` in a bidirectional setup, no replication loops are present.

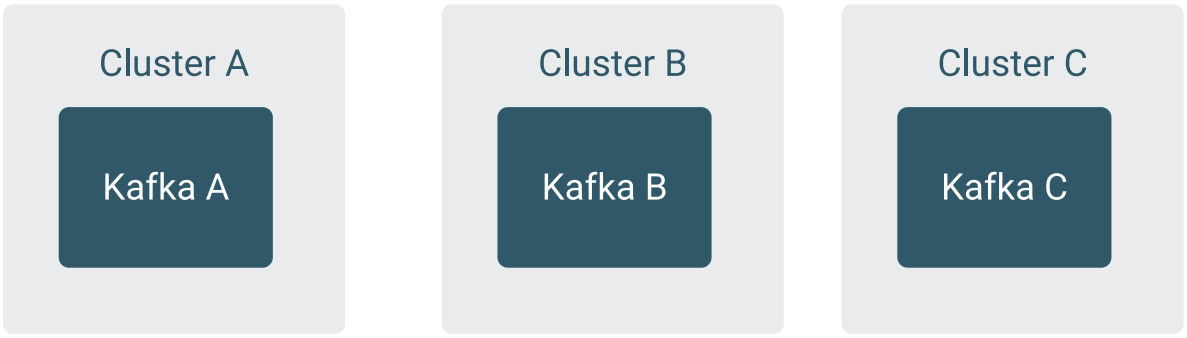
The reason why pull mode is recommended is because this is the deployment type that was found to provide the highest amount of resilience against various timeout and network instability issues. For example:

- Streams Replication Manager Driver roles run Connect workers which are coordinating through the target cluster of the replication flow. This means that Connect workers are more closely tied to the target cluster than the source cluster. Having the Streams Replication Manager Drivers and the target Kafka cluster closely located to each other minimizes group membership and rebalance related timeout issues. Additionally, this also minimizes the network instability on the producer side of the Streams Replication Manager Drivers which reduces data duplication in the target cluster.
- Streams Replication Manager Service roles also coordinate through the target cluster. This makes them more sensitive to timeout and network partition issues tied to the target cluster. In addition, the Streams Replication Manager Service reads from and writes to the target Kafka through the Kafka Streams application. Having the Streams Replication Manager Services and the target Kafka clusters closely located to each other can minimize timeout and network partition issues. Additionally, for Kafka clusters in the cloud, hosting the Streams Replication Manager Service in the same data center as the Kafka cluster can help keep cloud costs to a minimum.
- In a situation where there is a network partition, or one of the clusters in the replication is unavailable, it is preferable to let the target cluster pull the data when the connection is established, rather than the source cluster trying to push data indefinitely.

## Pull mode deployment example

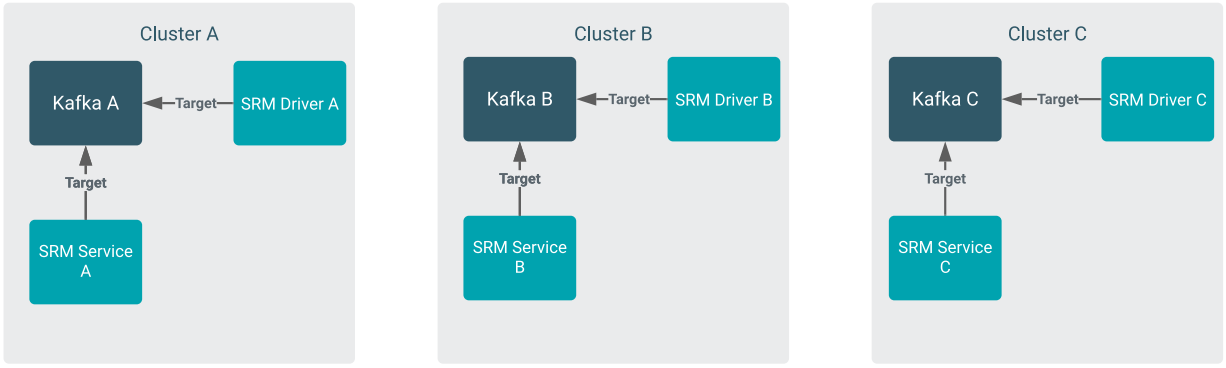
**Note:** The following simple example does not go into the detail regarding the exact configuration steps required to set up Streams Replication Manager or your clusters. It is meant to provide you with a general overview of how a pull mode deployment looks like. For configuration how to documentation see, How to Streaming Configuring Streams Replication Manager in the Cloudera Runtime documentation.

Consider a simple deployment that has three clusters. Cluster A, B, and C. Each of them has a Kafka cluster.

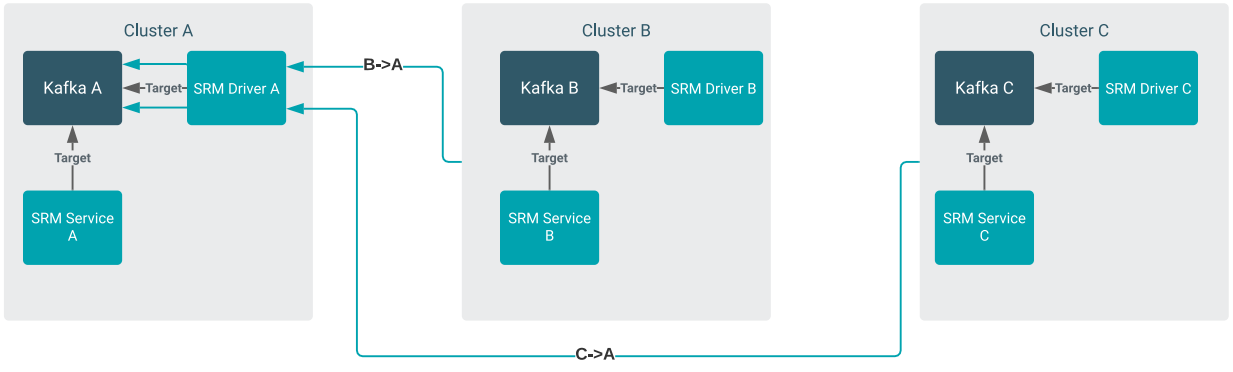



To achieve a pull mode deployment, you must deploy the Streams Replication Manager service (both Driver and Service roles) on all three clusters. The Driver and Service roles must target their co-located Kafka cluster.

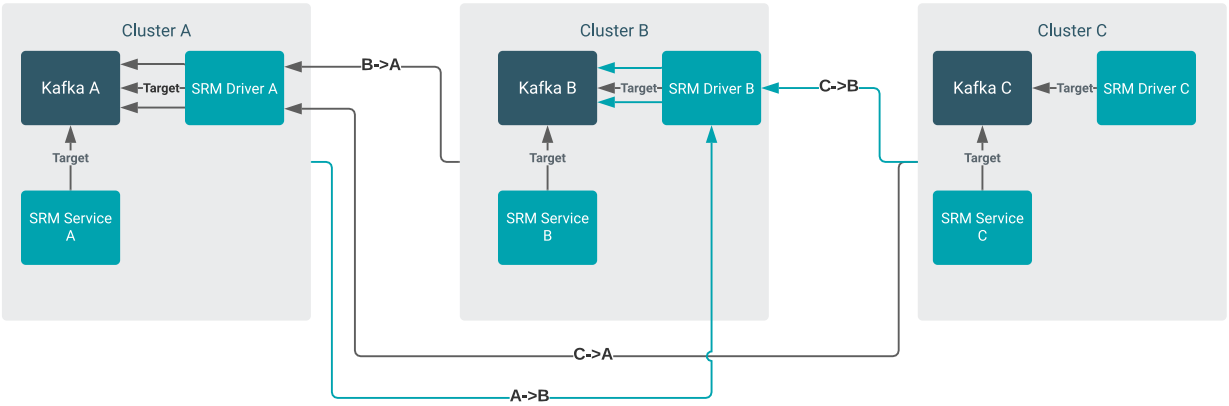
To be more precise, in the case of Driver targets, what you must ensure is that each source cluster is targeted by at least a single Driver. This is required so that a heartbeats topic is created in that cluster. This is ensured in the this example because each Kafka has a co-located Driver targeting it. However, if you have a unidirectional replication setup, where the source Kafka cluster does not have a co-located Streams Replication Manager service, you must ensure that one of your Drivers is targeting that Kafka cluster.




Once setup is complete, you can start configuring your replications. For example, assume that you want to have data replicated from Cluster B and C to cluster A. In this case, you need to enable two separate replications, B->A and C->A. To achieve pull mode, the two replications must be executed by Streams Replication Manager Driver A.




Any number of replications can be set up between the clusters, but you must always ensure that each Driver is only executing the replications targeting their co-located cluster. For example, assume that in addition to the replications set up previously, you also want to set up replication to Cluster B from Cluster A and Cluster C. In this case, the deployment would changes as follows.




# Reference architecture for reverse checkpointing

To effectively use reverse checkpointing, implement a four-topic architecture across your primary and backup clusters. This setup ensures that consumers receive all records—newly produced and replicated—while maintaining offset consistency during failover and failback.

**Figure 1: Reference architecture for reverse checkpointing**

### Topic configuration

Configure your topics for bidirectional replication with distinct original topics on each cluster:

**Primary cluster**

Create a topic to serve as the original topic during normal operation. This corresponds to the Paym ents topic on the left side of the diagram.

**Backup cluster**

Create a topic with the same name to serve as the original topic during disaster recovery. This corresponds to the Payments topic on the right side of the diagram.

With bidirectional replication enabled, SRM creates replica topics for each of your original topics. These are shown as the prefixed P.Payments and B.Payments topics on the diagram.

## Client configuration

Configure producers and consumers based on the operational state of the cluster:

**Normal operation**

**Producers**

Write to the original topic on the primary cluster (Payments on the left).

**Consumers**

Subscribe to the original topic on the primary cluster (Payments on the left) and the replica topic from the Backup cluster (B.Payments on the left).

**Disaster recovery**

**Producers**

Write to the original topic on the backup cluster (Payments on the right).

**Consumers**

Subscribe to the original topic on the backup cluster (Payments on the right) and the replica topic from the Primary cluster (P.Payments on the right).

## Failback synchronization

During failback, offset synchronization to the primary cluster is not instantaneous. To prevent consumers from reading old offsets or reprocessing data consumed on the backup, wait for the synchronization to finish before redirecting consumers back to the primary cluster.

**Important:** Produce data only to the original topics. Producing data to the replica topic can cause data loss during the failover or failback.