Cloudera Stream Processing 2.0.0

SRM Overview

Date published: 2019-09-13
Date modified: 2019-09-13

https://docs.cloudera.com/

Legal Notice

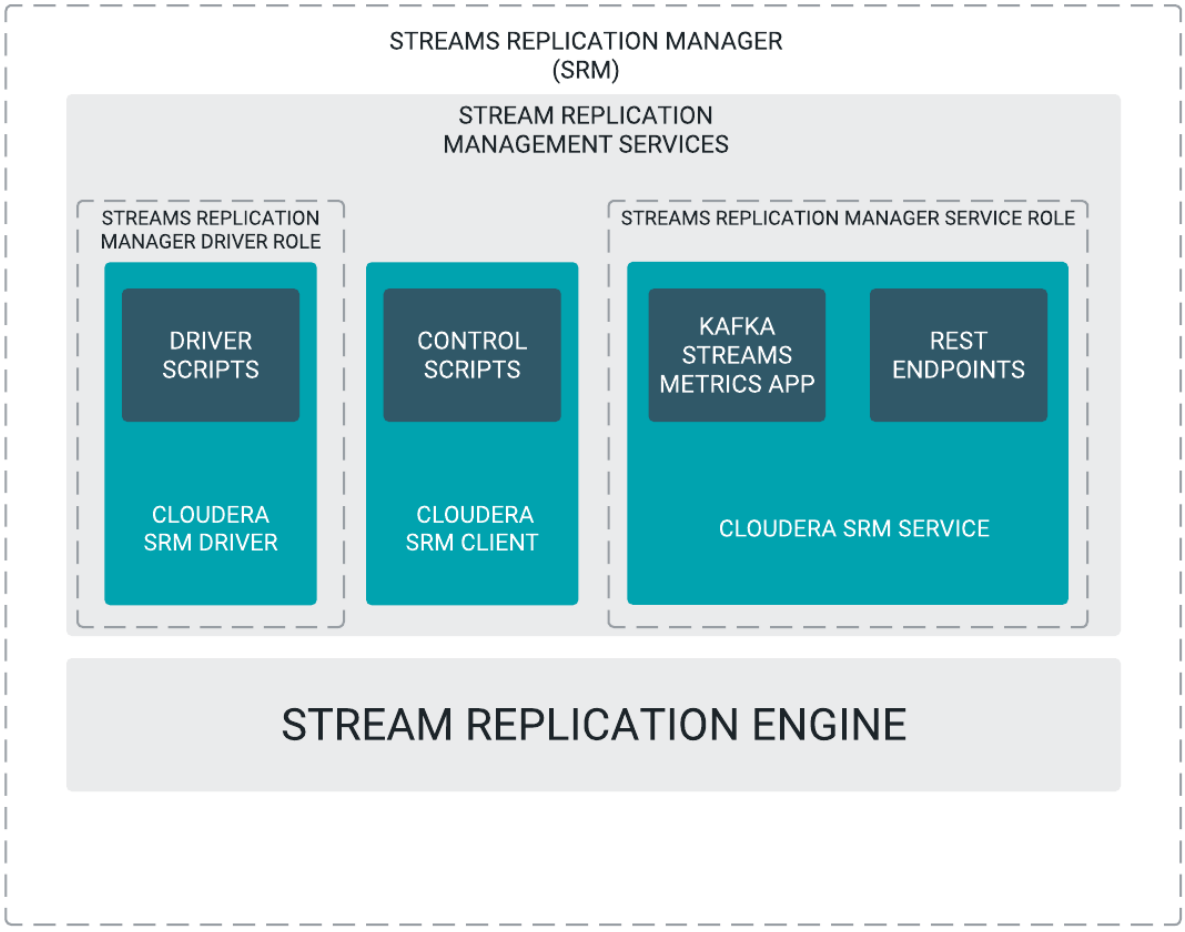
Contents

Overview

Get familiar with Streams Replication Manager and its components.

Streams Replication Manager (SRM) is an enterprise-grade replication solution that enables fault tolerant, scalable and robust cross-cluster Kafka topic replication. SRM provides the ability to dynamically change configurations and keeps the topic properties in sync across clusters at high performance. SRM also delivers custom extensions that facilitate installation, management and monitoring making SRM a complete replication solution that is built for mission critical workloads. Streams Replication Manager consists of two main components. The Stream Replication Engine and the Stream Replication Management Services.

Figure 1: Streams Replication Manager Overview



Stream Replication Engine

The Stream Replication Engine is a next generation multi-cluster and cross-datacenter replication engine for Kafka clusters.

Stream Replication Management Services

Stream Replication Management Services are services powered by open source Cloudera extensions which utilize the capabilities of the Stream Replication Engine. These services provide:

- Easy installation
- Lifecycle management

- Management and monitoring of replication flows across clusters

The Stream Replication Management Services includes the following custom extensions:

Cloudera SRM Driver

The Cloudera SRM Driver is a small wrapper around the Stream Replication Engine that adds the extensions provided by Cloudera. It provides the ability to spin up SRM clusters and has a metrics reporter. The driver is managed by Cloudera Manager and is represented by the Streams Replication Manager Driver role.

Cloudera SRM Client

The Cloudera SRM Client provides users with command line tools that enable replication management for topics and consumer groups. The command line tool associated with the Cloudera SRM Client is called srm-control.

Cloudera SRM Service

The Cloudera SRM Service consist of a REST API and a Kafka Streams application to aggregate and expose cluster, topic and consumer group metrics. The service is managed by Cloudera Manager and is represented by the Streams Replication Manager Service role.

Key Features

SRM has the following main features.

Remote topics

Remote topics are replicated topics referencing a source cluster via a naming convention. For example, the “topic1” topic from the “us-west” source cluster creates the “us-west.topic1” remote topic on the target cluster. SRM automatically applies this configurable "replication policy" across your organization, enabling tooling to distinguish remote topics from source topics. For more information regarding remote topics, see Understanding Replication Flows.

Consistent semantics

Partitioning and record offsets are synchronized between replicated clusters to ensure consumers can migrate from one cluster to another without losing data or skipping records.

Cross cluster configuration

Topic-level configuration properties and ACL policies are synced across clusters. For example, principals that can read a source topic are automatically able to read any corresponding remote topics. This simplifies managing topics across multiple Kafka clusters.

Consumer group checkpoints

In addition to data and configuration, SRM replicates consumer group progress via periodic checkpoints. At configurable intervals, checkpoint records are emitted to downstream clusters, encoding the latest offsets for whitelisted consumer groups and topic-partitions. As with topics, groups are matched against an allowlist which can be updated dynamically with srm-control. Normally, consumer group offsets are not portable between Kafka clusters, as offsets are not consistent between otherwise identical topic-partitions on different clusters. SRM’s checkpoint records account for this by including offsets which are automatically translated from one cluster to another. This offset translation feature works in both directions; a consumer group can be migrated from one cluster to another (failover) and then back again (failback) without skipping records or losing progress.

Automatic topic and partition detection

SRM monitors Kafka clusters for new topics, partitions, and consumer groups as they are created. These are compared with configurable whitelists, which may include regular expressions.

Tooling to automate consumer migration

SRM tooling enables operators to translate offsets between clusters and to migrate consumer groups while preserving state.

Centralized configuration for multi-cluster environments

SRM leverages a single top-level configuration file to enable replication across multiple Kafka clusters. Moreover, command-line tooling can alter which topics and consumer groups are replicated in real-time.

Replication monitoring

Since cluster replication will mainly be used for highly critical Kafka applications, it is crucial for customers to be able to easily and reliably monitor the Kafka cluster replications. The custom extensions included with SRM collect and aggregate Kafka replication metrics and make them available through a REST API. This REST API is used by Streams Messaging Manager (SMM) to display metrics. Customers could also use the REST API to implement their own monitoring solution or plug it into third party solutions. The metrics make the state of cluster replication visible to end users who then can take corrective action if needed.

Related Information

Monitoring Cluster Replications Overview
Understanding Replication Flows

Main Use Cases

Learn about the main use cases of SRM.

Apache Kafka has become an essential component of enterprise data pipelines and is used for tracking clickstream event data, collecting logs, gathering metrics, and being the enterprise data bus in a microservices based architectures. Kafka supports internal replication to support data availability within a cluster. However with Kafka based applications becoming critical, enterprises require that the data availability and durability guarantees span entire cluster and site failures.

Replication of data across clusters and sites is key for the following use cases:

Disaster Recovery

Common enterprise use cases for cross-cluster replication is for guaranteeing business continuity in the presence of cluster or data center-wide outages.

Aggregation for Analytics

Aggregate data from multiple streaming pipelines possibly across multiple data centers to run batch analytics jobs that provide a holistic view across the enterprise.

Data Deployment after Analytics

This is the opposite of the aggregation use case in which the data generated by the analytics application in one cluster (say the aggregate cluster) is broadcast to multiple clusters possibly across data centers for end user consumption.

Isolation

Due to performance or security reasons, data needs to be replicated between different environments to isolate access. In many deployments the ingestion cluster is isolated from the consumption clusters.

Geo Proximity

In geographically distributed access patterns where low latency is required, replication is used to move data closer to the access location.

Cloud Migration

As more enterprises have an on-premise and cloud presence, Kafka replication can be used to migrate data to the public or private cloud and back.

Legal and Compliance

Much like the isolation uses case, a policy driven replication is used to limit what data is accessible in a cluster to meet legal and compliance requirements.

Use Case Architectures

Highly available and cluster migration architecture examples for SRM.

Highly Available Kafka Architectures

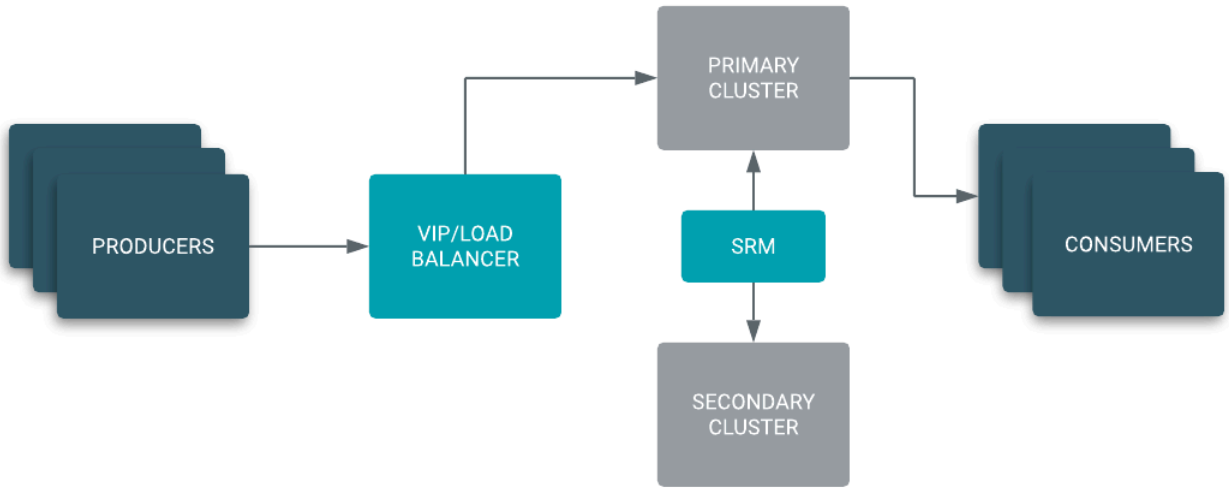
A highly available Kafka deployment must be able to survive a full single cluster outage while continuing to process events without data loss. With SRM, you can implement highly available Apache Kafka deployments which either follow an Active / Stand-by or an Active / Active model.

Active / Stand-by Architecture

Active / Stand-by architecture example for SRM.

In an Active / Stand-by scenario, you set up two Kafka clusters and configure SRM to replicate topics bi-directionally between both clusters. A VIP or load balancer directs your producers to ingest messages into the active cluster from which consumer groups are reading from.

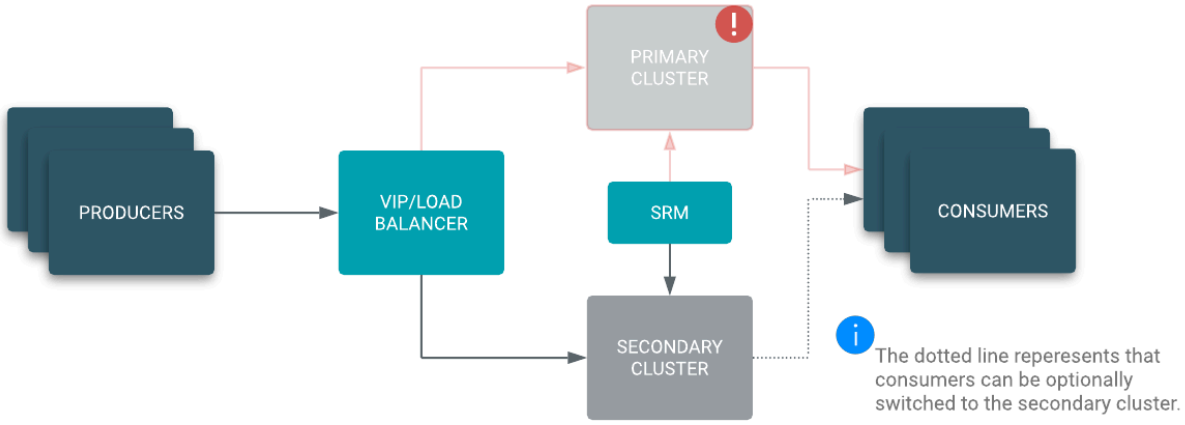
Figure 2: Active / Stand-by Architecture Standard Operation



In case of a disaster, the VIP or load balancer directs the producers to the stand-by cluster. You can easily migrate your consumer groups to start reading from the stand-by cluster or simply wait until the primary cluster is restored if the resulting consumer lag is acceptable for your use case.

While the primary cluster is down, your producers are still able to ingest. Once the primary cluster is restored, SRM automatically takes care of synchronizing both clusters making failback seamless.

Figure 3: Active / Stand-by Architecture Cluster Failure



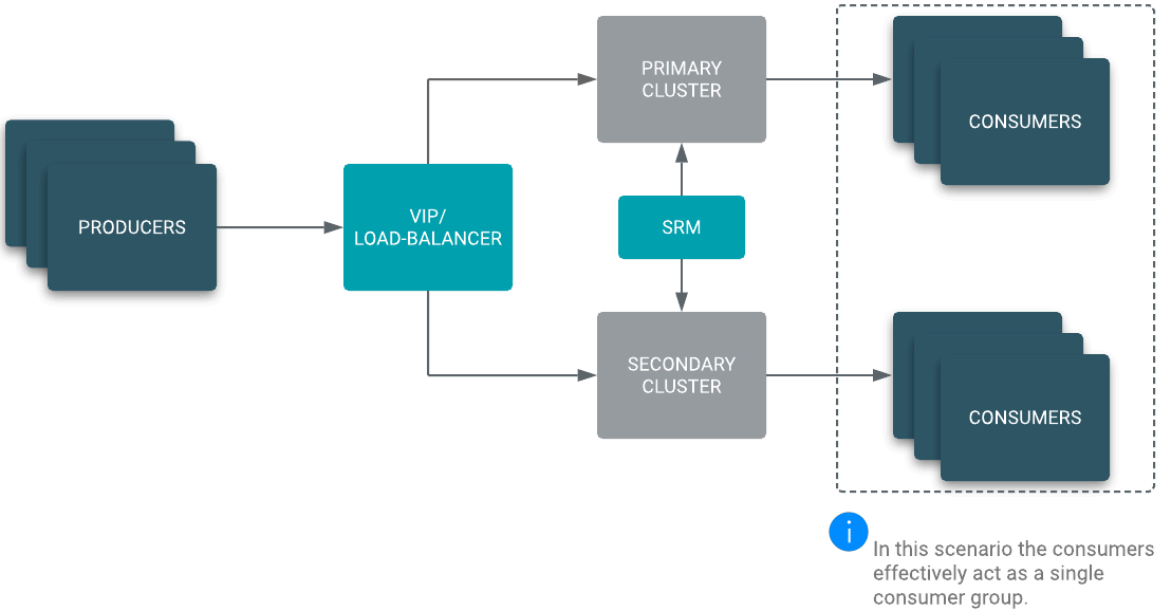
Implementing an Active/Stand-by architecture is the logical choice when an existing disaster recovery site with established policies is already available, and your goals include not losing ingest capabilities during a disaster and having a backup in your disaster recovery site.

Active / Active Architecture

Active / Active architecture example for SRM.

In an Active / Active scenario, your producers can be load balanced to either your primary or secondary cluster. SRM is configured to replicate topics bi-directionally between both clusters. What makes this architecture Active / Active, is the fact that you now have consumers reading from both clusters at the same time, essentially acting like a cross-cluster consumer group. In case of a disaster the VIP or load balancer directs the producers to the secondary cluster and the secondary cluster consumer group is still able to process messages.While the primary cluster is down, your producers are still able to ingest and your consumers are still able to process messages. This results in a 0 downtime and hands-off failover in case of a disaster. Once the primary cluster is back online, SRM automatically takes care of synchronizing both clusters and your primary consumer group resumes processing messages.

Figure 4: Active / Active Architecture



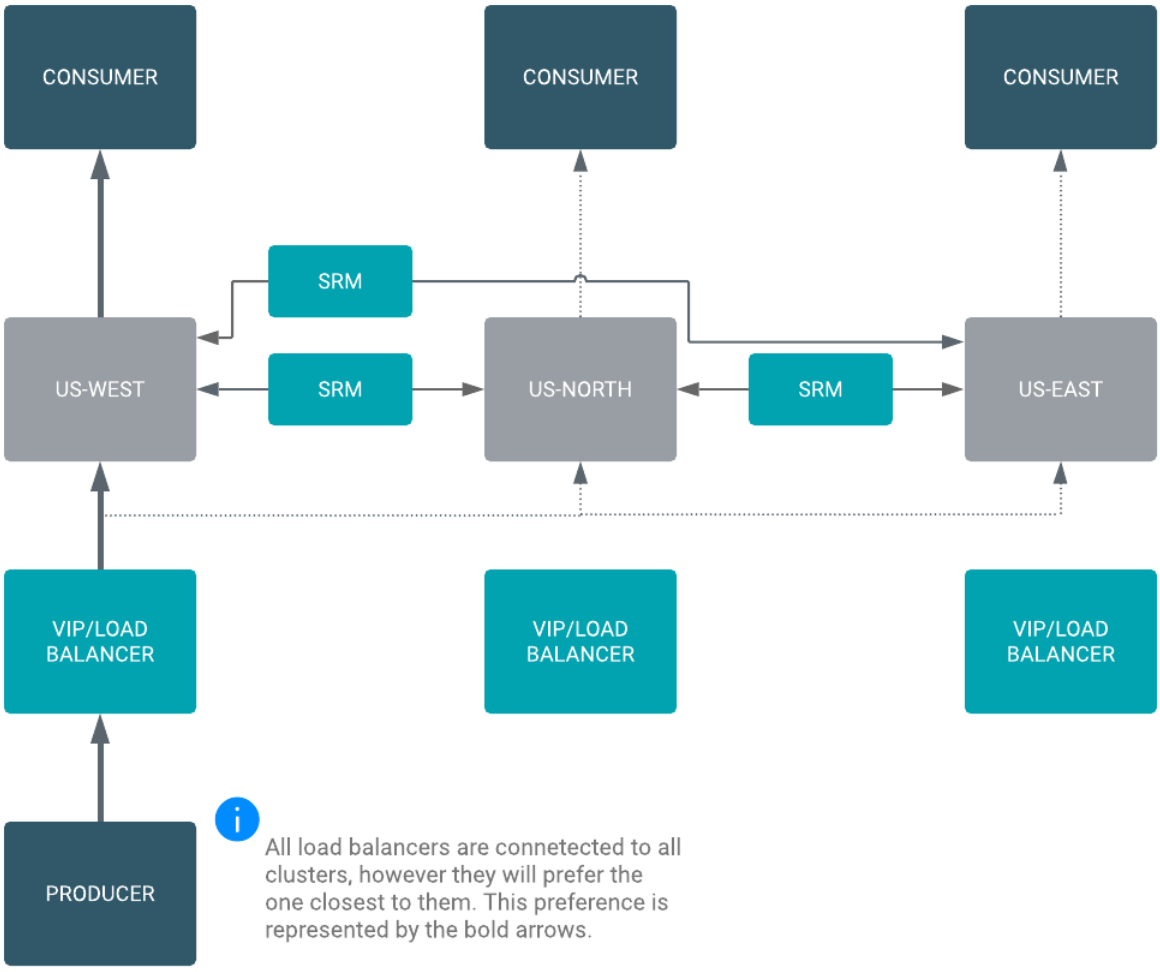
Cross Data Center Replication

Cross data center replication architecture example for SRM.

Certain applications not only require local high availability within one data center or one availability zone, but have to be highly available across data centers too. You can use SRM to set up replication between Kafka clusters in different data centers which results in messages being available to consumers in each of your data centers.

A load balancer directs your producers to the local data center or closest data center if the primary data center is down. SRM is configured to replicate topics between all data centers. If you are using more than two data centers, SRM is configured to create a “replication circle”, ensuring a single data center failure (for example us-north in the example below) does not halt replication between the remaining clusters.

Figure 5: Cross Data Center Replication Architecture



Cluster Migration Architectures

Example cluster migration architectures.

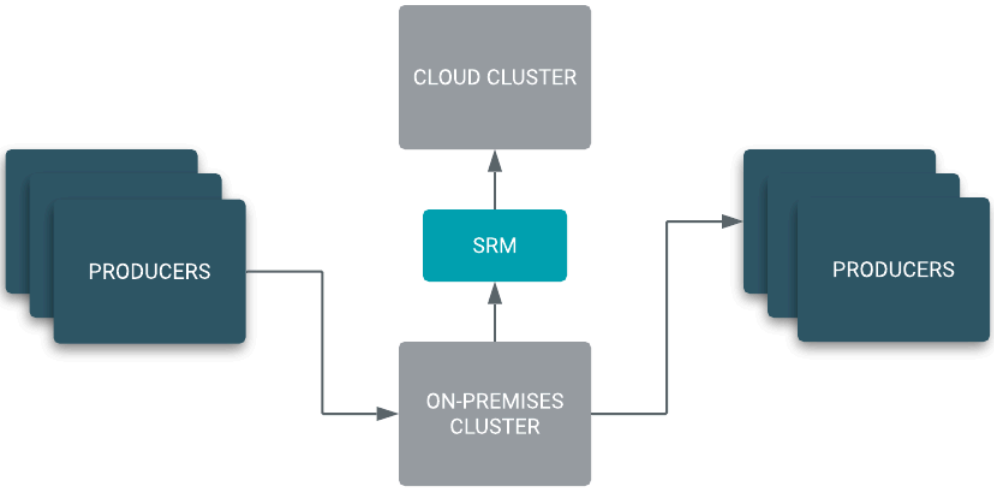
On-premise to Cloud and Kafka Version Upgrade

On-premise to cloud and Kafka version upgrade example architectures for SRM.

If you have an on-premises Apache Kafka cluster that you want to migrate to the cloud, not only do you have to migrate consumers and producers, you also have to migrate topics and their messages to the new cloud based cluster.

After you have set up replication through SRM, you only need to point your consumers to the new brokers before you can start processing messages from the cloud cluster. This approach ensures that the historical data kept in the on-premises Kafka cluster is migrated to the cloud cluster allowing you to replay messages directly from the cloud without having to go back to your on-premises cluster.

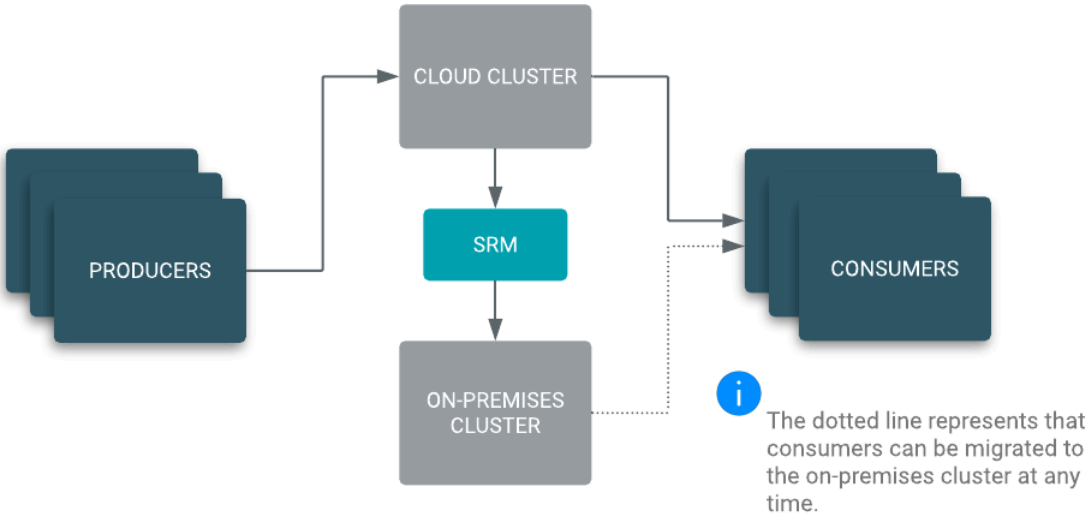
Figure 6: Cluster Migration On-premise



Producers and Consumers are using the on-premises cluster while SRM is replicating messages.

Once you have migrated your cluster, producers, and consumers to the cloud, you can use SRM to turn-around the replication direction and use the on-premises cluster as your DR cluster.

Figure 7: Cluster Migration Cloud



Producers and Consumers have been migrated to the cloud cluster and the on-premises cluster is used for disaster recovery.

Kafka Version Upgrade

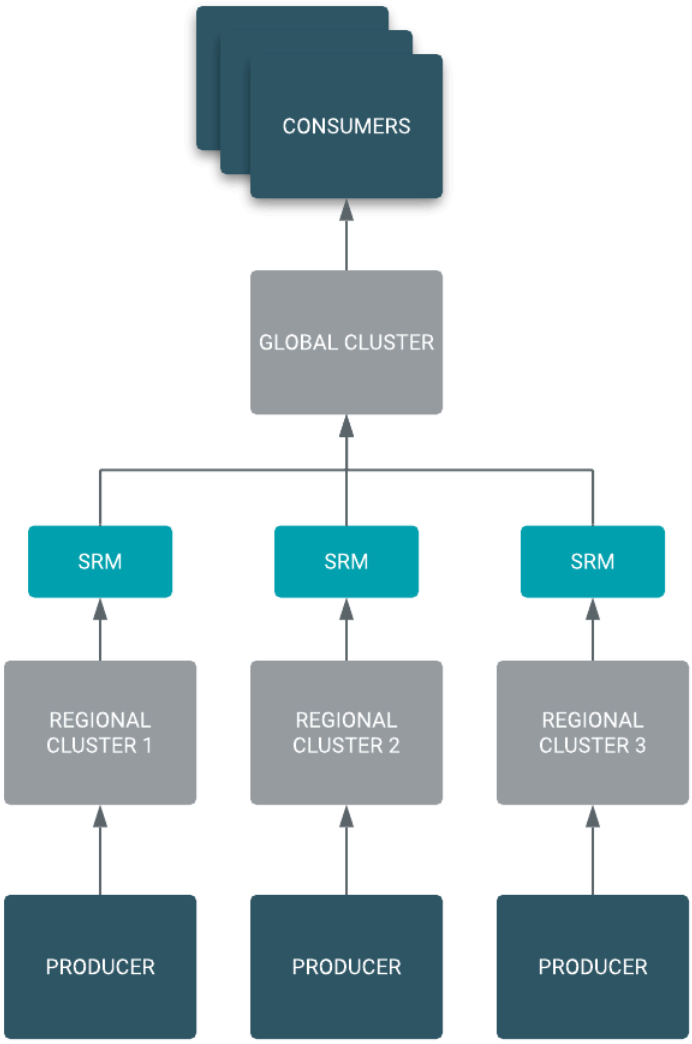
If you have to upgrade your Kafka cluster to a newer version and an in-place upgrade is not possible, you can use the same migration approach to provision a new cluster, use SRM to replicate all existing topics and messages before migrating your producers and consumers to interact with the new cluster.

Aggregation for Analytics

Aggregation for analytics architecture example for SRM.

SRM can be used to aggregate data from multiple streaming pipelines, possibly across multiple data centers, to run batch analytics jobs that provide a holistic view across the enterprise.

Figure 8: Aggregation for Analytics



Understanding Replication Flows

Get familiar with the concept of replications and replication flows and learn more about how they can be set up.

Replication Flows Overview

Get familiar with the concept of replications and replication flows.

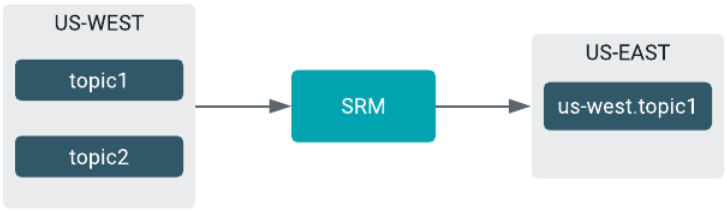
Replication involves sending records from a source cluster to a target cluster. In SRM a replication refers to a source and target cluster pair, the direction in which data is flowing and the topics that are being replicated. Source target cluster pairs can be specified in Cloudera Manager; they are notated source->target. Initially, when source->target pairs are set up they are considered inactive, as no data is being replicated between them. To start replication users need to specify which topics to replicate with the srm-control command line tool.

It is important to understand that replication in SRM is configured independently for each source->target cluster pair. Moreover, configuration is done on a per topic basis. This means that each topic in a source cluster can have a different direction or target that it is being replicated to. A set of topics in the source cluster can be replicated to multiple target clusters while others are being replicated to only one target cluster. This allows users to set up powerful, topic specific replication flows.

The term replication flow is used to specify all replications set up in a system. This document uses the term when referring to the visual representation of SRM replications.

A basic example of a replication flow is when topics are being sent from one cluster to another cluster in a different geographical location. Note that in this example there is only one replication or source->target pair. Moreover, only one of the two topics on the source cluster are being replicated to the target cluster.

Figure 9: Simple Replication Flow Example

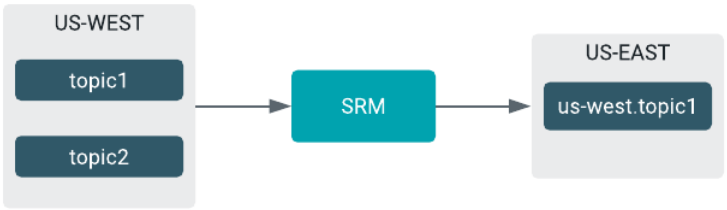


Remote Topics

Learn about SRM's remote topics.

In any replication flow, the selected source topics are replicated to remote topics on the target cluster. Remote topics reference the source cluster via a naming convention. For example, the topic1 topic from the us-west source cluster creates the us-west.topic1 remote topic on the target cluster.

Figure 10: Simple Replication Flow Example



Remote topics can themselves be replicated. In this case, the remote topic references all source and target clusters. The prefix in the name will start with the cluster closest to the final target cluster. For example, the topic1 topic replicated from the us-west source cluster to the us-east cluster and then to the eu-west cluster will be named us-east. us-west.topic1.

Figure 11: Complex Replication Flow Example

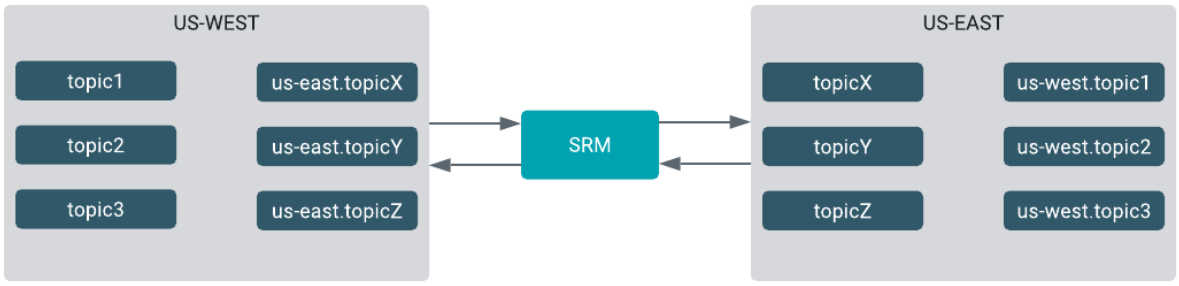
Tip: You might want to have your Kafka consumers read messages from both source and remote topics simultaneously. To achieve this, Kafka consumers should include a wildcard topic name pattern. For example, suppose that you want your consumer to read from topic1 located in us-west and its remote counterpart, us-west.topic1, located in us-east. In such a case, you can use the .*topic1 pattern, which matches any topic that ends with topic1.

Bi-directional Replication Flows

Learn more about bi-directional replication flows.

SRM understands cycles and will never replicate records in an infinite loop. This enables bi-directional replication flows in which clusters are mutually replicated. In this case, records sent to one cluster will be replicated to the other and the other way around. You can configure any number of clusters in this way.

Figure 12: Bi-directional Replication Flow

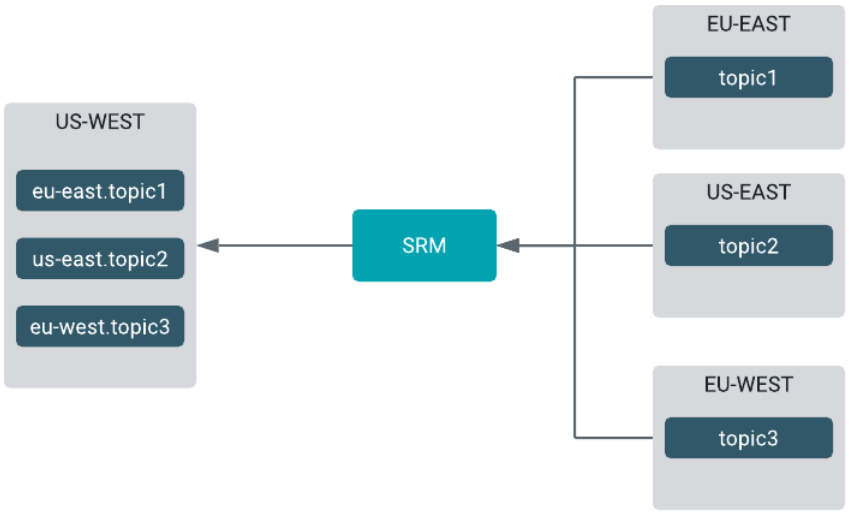


Fan-in and Fan-out Replication Flows

Learn about fan-in and fan-out replication flows.

You can construct fan-in replication flows, where records from multiple source clusters are aggregated in a single target cluster.

Figure 13: Fan-in Replication Flow



Similarly, you can construct fan-out replication flows as well, where a single cluster is replicated to multiple target clusters.

Figure 14: Fan-out Replication Flow