Cloudera Runtime 7.3.2

Migrating Kafka clusters from ZooKeeper to KRaft in Cloudera on cloud

Date published: 2026-03-31
Date modified: 2026-03-31

https://docs.cloudera.com/

Legal Notice

Contents

Migrating Kafka from ZooKeeper to KRaft

Learn about migrating an existing Zookeeper-based Kafka cluster to KRaft. Migration is performed in Cloudera Management Console, using the migration action available under the Cloudera Data Hub cluster management. Additionally, migration is done in a rolling fashion requiring no downtime.

Note: Migrating Kafka from ZooKeeper to KRaft is in Technical Preview for Cloudera Runtime 7.3.2.0. The migration creates a three node setup for KRaft, which cannot be scaled.

Apache Kafka Raft (KRaft) is a consensus protocol used for metadata management that was developed as a replacement for Apache ZooKeeper. Using KRaft for managing Kafka metadata instead of ZooKeeper offers various benefits including a simplified architecture and a reduced operational footprint.

Important: Cloudera encourages you to migrate existing clusters to KRaft as soon as you have completed your upgrade to Cloudera Runtime 7.3.2. This is the only Cloudera Runtime version where migration is possible. Neither previous or future major, minor, and maintenance versions support migration.

Migration at glance

Migration is completed in two steps, start and finalize, with the option to roll back at any time before finalization. You manage a migration (start, finalize, or rollback) using the migration action available under the Cloudera Data Hub cluster management in Cloudera Management Console. Configuration changes needed for the migration are automated by the migration actions.

The migration actions in Cloudera Data Hub are as follows:

- Migrate Kafka to KRaft – Starts migration of Kafka brokers from ZooKeeper to KRaft for the specified Kafka service. Migrates the cluster up to a sate where rollback is still possible.

  When this action finishes, brokers will run in KRaft mode and will be disconnected from ZooKeeper. KRaft controllers will still be connected to ZooKeeper and will continue to write metadata to ZooKeeper, but are ready to disconnect (dual-write mode).
- Finalize KRaft Migration – Finalizes migration by disconnecting KRaft controllers from ZooKeeper. Reverting to ZooKeeper is not possible once this action starts.
- Rolling back KRaft Migration – Reverts migration of Kafka brokers from ZooKeeper to KRaft for the specified Kafka service using rolling restarts.

You can only start these actions if your cluster meets the required prerequisites. If prerequisites are not met, the actions will be disabled.

Migrating a Cloudera Streams Messaging cluster to KRaft

You migrate an existing ZooKeeper-based Cloudera Streams Messaging cluster to KRaft by using the migration feature available in Cloudera Management Console.

Before you begin

- Migrating an existing ZooKeeper-based Cloudera Streams Messaging cluster KRaft is only available if you are on Cloudera Manager 7.13.2 or later and Cloudera Runtime 7.3.2.

  Migration is only possible with this combination of versions. This is because:

  - Earlier Cloudera Manager versions do not include the necessary Kafka service actions.
  - Cloudera Runtime 7.3.2 is the only version where migration is possible. Neither previous or future major, minor, and maintenance versions support migration.

- The Kafka service must run with inter-broker protocol version 3.9.

  Verify the version by checking the value of the Kafka Inter-Broker Protocol Version property in Cloudera Manager Kafka service Configuration . The value of the property must be 3.9 or empty. An empty value means that version is set to the default, which is 3.9 in Cloudera Runtime 7.3.2.

Warning: In case you need to repair the KRaft group, do not select the Delete attached disk option as it can result in cluster failure.

Tip: If the migration command is not available in Cloudera Management Console, your cluster does not satisfy requirements for migration. Check that all prerequisites are met.

Procedure

1. Navigate to your Streams Messaging cluster in Cloudera Management Console.
2. Select Upgrade on the **Data Hub** details page.

   If you are on Cloudera Runtime 7.3.2, the migration to KRaft will appear.
3. Click Start to confirm that you want to start the migration from ZooKeeper to KRaft.

   The migration remains in progress until a success message is displayed.

   

   Note: In case the migration has failed, use the Roll back button to revert back to ZooKeeper and to start the migration process again.

   When the migration is completed, you can either finalize the migration or roll back the Kafka service to the previous state.
4. Finalize or Roll back the migration.

   For Finalize the migration

   a. Ensure that all applications and services connecting to Kafka operate as expected.
   b. Click Finalize.
   c. Click Finalize to confirm that you want to continue the migration. This action cannot be undone and rolling back to ZooKeeper will no longer be possible after the finalization.
   d. Check the status of the migration and if it shows **Migration completed**, the migration from ZooKeeper to KRaft was successful. Kafka is running in KRaft mode in your Cloudera Data Hub cluster.

      In case the finalization has failed, use the Retry Finalization button to start the process again.

   For Roll back the migration

   a. Click Roll back.
   b. Click Roll back to confirm that you want to revert back to ZooKeeper.You can reinitiate the migration to KRaft again after rolling back to ZooKeeper.
   c. Check the status of the migration and if the Start Migration button appears again, the rollback to ZooKeeper was successful. Kafka is running in ZooKeeper mode in your Cloudera Data Hub cluster.

      In case the rollback has failed, use the Retry Rollback button to start the process again.

Results

ZooKeeper to KRaft migration for your Cloudera Streams Messaging cluster is completed.

Troubleshooting ZooKeeper to KRaft migration

Common issues and solutions for troubleshooting problems that might occur during the migration of Kafka clusters from ZooKeeper to KRaft mode.

KRaft Controllers were started without migration configuration

Condition

When KRaft Controller roles are added to the cluster in Cloudera Private Cloud Base, by default they are stopped, but it is possible to start them without enabling migration mode. If this is done, then the controllers start the KRaft cluster in full KRaft mode. This is a problem because the internal KRaft migration state will be initialized in the terminal KRaft cluster state. Migration will not happen and cannot be initiated in this state.

Confirm this by looking at Cloudera Manager command history and the health history of the controller instances. If KRaft Controllers are running but there was no migration started, then controllers were started manually.

Further confirmation can be done by reviewing the kafka_zk_migration_state metric. Use the following query in Charts Chart Builder to review the metric.

```
SELECT kafka_zk_migration_state
```

A metric value of 0 indicates that controllers were started without enabling migration mode and the cluster is started in full KRaft mode.

Cause

KRaft Controller role instances were added to the cluster and started manually without migration configuration. That is, they were started manually and not through the Migrate Kafka to KRaft action.

Remedy

Procedure

1. Stop the KRaft Controller roles.
2. Wipe the log directories of the KRaft Controller roles.
   This resets the controllers to a state where migration can be initiated.
3. Start migration with the Migrate Kafka to KRaft.

Newly provisioned KRaft controllers cannot maintain leader or quorum

Condition

The KRaft controllers cannot maintain a stable leader election or quorum, the cluster's core metadata management is at risk. The active controller chart is frequently updated with different controllers assuming leadership.

Cause

Networking (high likelihood) or disk issues (low likelihood).

Remedy

Collect diagnostic data and troubleshoot potential issues in your cluster. Look for possible networking or disk issues. In stretch clusters, you will most likely need to adjust buffer sizes.

Broker connectivity issues

Condition

Brokers fail to correctly register with the KRaft controller quorum after their configuration is updated and started in migration mode.

Cause

Brokers show concerning (yellow) or bad (red) health in Cloudera Manager. Additionally, the logs show that brokers cannot register to the KRaft quorum.

Remedy

Determine if the issue is a misconfiguration or a bug in the migration integration. Likely causes of misconfiguration can be a custom listener setup that the migration integration was not prepared for, or authentication issues.

The remedy is highly dependent on the specific problem, and you will need to review logs to find the root cause. Before you review logs, ensure that brokers are connected to ZooKeeper and serve traffic normally.

Resume migration if the issue is resolved. Alternatively, revert the migration.

Performance degradation

Condition

Experiencing unexpected or significant performance drops, high latency, or increased resource utilization (CPU/memory) on the brokers or controllers. Relevant charts in Cloudera Manager or in any other metrics monitoring applications show the performance degradation of brokers, controllers, or clients.

Cause

This issue might surface during migration, but it might be unrelated to it. Consumer and Connect group rebalances might cause a performance degradation with unbalanced clusters.

Remedy

Revert migration and troubleshoot performance issues in your cluster. If the issues remain even after a revert, then it is highly likely that the performance degradation is caused by an inherent cluster imbalance.

Metadata consistency issues

Condition

Observing errors in metadata synchronization between the KRaft quorum and ZooKeeper during the dual-write phase and during migration to KRaft mode.

You might experience issues like:

- Brokers have inconsistent information about partition leaders, causing partitions to fall out of the ISR.
- Topic configurations might differ for topics in Zookeeper and in KRaft.
- Brokers might be different in Zookeeper and KRaft metadata.

Cause

As a result of network issues, bugs, unsupported configurations, or misconfiguration, metadata consistency issues might happen when the brokers are being restarted in KRaft mode. During this phase, a part of the cluster already reads metadata from KRaft, while the other parts still use ZooKeeper. Parts of the cluster that still use ZooKeeper might learn about updates much later if there is a network glitch or a bug.

Remedy

- Do network diagnostics to ensure that buffers are set up for the latency of the network (especially in stretch clusters).
- Check DNS response times. Specifically check if Kafka spends a lot of time resolving DNS addresses. Sometimes DNS resolution problems can cause unstable messaging in Kafka.
- Check for authentication or encryption issues. A slow or inconsistent Key Distribution Center (KDC) might cause connections or authentication requests to lag. This in turn slows down in-sync replicas (ISR) and UpdateMetadata requests which may affect metadata propagation.
- Consider reverting the migration, fixing any network issues, then restart the migration.

Client compatibility

Condition

A critical application or tooling component (especially older ones) relies directly on ZooKeeper paths or functionality that is broken during the dual-write phases. Such errors have a very low chance since the last Java client that directly accesses ZooKeeper has been removed in Kafka 0.10, but third-party clients not supported by Cloudera might still work with ZooKeeper connections. Ideally, this problem should only surface in development or test environments.

Cause

Client applications cannot communicate with the brokers. The crash happens when the brokers enter their dual-write phase.

Remedy

Revert the migration and upgrade your clients so that they are compatible with KRaft. Restart migration afterwards.

Migration fails with `NoAuthException`

Condition

Zookeeper Access Control List (ACL) synchronization is done by the migration command as a preparatory step. Any subsequent manual changes to the ACLs can potentially break the migration. Such issues can happen if you use Kafka CLI tools to modify topic configuration or change the topics. Essentially, changing the ZooKeeper structure with any CLI commands can break migration.

Kafka or KRaft migration fails in any step and the Kafka or KRaft logs include `NoAuthException` stack traces, similar to the following example:

```
Exception in thread "main" org.apache.zookeeper.KeeperException$NoAuthExcept
ion: KeeperErrorCode = NoAuth for /kafka-specific-node
```

Cause

ZooKeeper ACLs were changed during migration. Kafka command line tools were used to update topics or their configurations.

Remedy

Procedure

1. Apply the same ACLs to the node that is printed in the exception as all the others.
2. Resume migration.

The kraft user is not authorized to perform actions

Condition

Authorization errors are included in the Kafka and KRaft logs that say that the kraft user is not authorized to perform actions.

Cause

Ranger policies are not updated for KRaft, or are misconfigured.

Remedy

Procedure

1. Add the kraft user to required Ranger policies.
   a) In the Ranger Admin Web UI, select the Kafka resource-based service (default cm_kafka).
   b) Add the kraft user to all policies that include the kafka user.

      The kraft user must have the same permission in all policies as the kafka user.

      At minimum, you must add the kraft user to the following default policies:

      - all - consumergroup
      - all - topic
      - all - transactionalid
      - all - cluster
      - all - delegationtoken
      - connect internal - topic

      Tip: The kafka user/principal is the default for Kafka. If you have a custom Kerberos principals configured for the Kafka service, the user/principal might be different. Check the Cloudera Manager Kafka Configuration Kerberos Principal property. Its value is the default user/principal Kafka runs as.
2. Create a new policy that restricts access to the __cluster_metadata topic with the following permissions:
   - kraft user – All permissions
   - kafka user – **Describe** (describe), **Describe Configs**(describe_configs), and **Consume** (consume).

   Policy example in JSON:

```
{
  "isEnabled": true,
  "service": "cm_kafka",
  "name": "kraft internal - topic",
  "policyType": 0,
  "policyPriority": 0,
  "description": "Policy for kraft internal - topic",
  "isAuditEnabled": true,
  "resources": {
    "topic": {
      "values": [
```

```
        "__cluster_metadata"
      ],
      "isExcludes": false,
      "isRecursive": false
    }
  },
  "policyItems": [
    {
      "accesses": [
        {
          "type": "create",
          "isAllowed": true
        },
        {
          "type": "delete",
          "isAllowed": true
        },
        {
          "type": "configure",
          "isAllowed": true
        },
        {
          "type": "alter",
          "isAllowed": true
        },
        {
          "type": "alter_configs",
          "isAllowed": true
        },
        {
          "type": "describe",
          "isAllowed": true
        },
        {
          "type": "describe_configs",
          "isAllowed": true
        },
        {
          "type": "consume",
          "isAllowed": true
        },
        {
          "type": "publish",
          "isAllowed": true
        }
      ],
      "users": [
        "kraft"
      ],
      "delegateAdmin": false
    },
    {
      "accesses": [
        {
          "type": "describe",
          "isAllowed": true
        },
        {
          "type": "describe_configs",
          "isAllowed": true
        },
        {
          "type": "consume",
          "isAllowed": true
```

```
        }
      ],
      "users": [
        "kafka"
      ],
      "delegateAdmin": false
    }
  ],
  "serviceType": "kafka",
  "isDenyAllElse": true
}
```