Cloudera Streams Messaging Operator for Kubernetes 1.6.0

Kafka Operations

Date published: 2024-06-11
Date modified: 2026-01-27

https://docs.cloudera.com/

Legal Notice

Contents

Managing topics

Topics can be managed declarativaly with the Strimzi Topic Operator.

The Strimzi Topic Operator runs in unidirectional mode. With this mode, you can create topics with the `KafkaTopic` resource, which are then managed by the Strimzi Topic Operator. Since the Strimzi Topic Operator only manages topics which have corresponding `KafkaTopic` resources, if a managed topic's configuration is changed directly in the Kafka cluster, those changes will be reverted.

Note: Bidirectional mode for the Strimzi Topic Operator is deprecated.

```
apiVersion: kafka.strimzi.io/v1
kind: KafkaTopic
metadata:
  name: my-topic
  labels:
    strimzi.io/cluster: my-cluster
spec:
  partitions: 1
  replicas: 1
  config:
    retention.ms: 7200000
    segment.bytes: 1073741824
```

You can list topic resources with kubectl get.

```
kubectl get kafkatopics --output wide --namespace [***NAMESPACE***]
```

Related Information

Using the Topic Operator to manage Kafka topics | Strimzi

KafkaTopic schema reference | Strimzi API reference

Scaling brokers

Learn how to upscale or downscale the number of brokers in your Kafka cluster.

To scale Kafka brokers in Cloudera Streams Messaging Operator for Kubernetes you:

- Scale the number of broker replicas in your `KafkaNodePool` resources.
- Move your data by rebalancing with Cruise Control.

The order in which you scale and rebalance is different for upscale and downscale operations. In case of an upscale, you scale brokers first and then rebalance. In case of a downscale, you rebalance first and scale brokers afterward.

By default, rebalancing with Cruise Control is initiated manually. You must create a `KafkaRebalance` resource and then review and approve the rebalance proposal that is generated to move data.

Alternatively, you can enable auto-rebalancing in the `Kafka` resource. In this case, the Strimzi Cluster Operator automatically rebalances your cluster with Cruise Control. The Strimzi Cluster Operator initiates the rebalance operation using the rebalancing templates (`KafkaRebalance` resources) that you create and configure.

Note: Cloudera recommends that you enable auto-rebalancing as this makes scaling your cluster faster and easier.

When you scale a `KafkaNodePool` resource, the Strimzi Cluster Operator automatically adds or deletes broker nodes. Initiating a rebalance process with Cruise Control automatically moves data between brokers based on a proposal generated by Cruise Control.

Enabling auto-rebalancing

Complete these steps to enable auto-rebalancing for Kafka brokers. Enabling auto-rebalancing automates the rebalancing step of a scaling operation.

About this task

You enable auto-rebalancing for Kafka by creating one or more Kafka rebalancing templates. Additionally, you add the spec.cruiseControl.autoRebalance property to your `Kafka` resource. This property enables auto-rebalancing and specifies which rebalancing templates should be used.

A rebalancing template is a `KafkaRebalance` resource that includes the strimzi.io/rebalance-templat e: "true" annotation. This annotation designates the resource as a rebalancing template. When you create a `KafkaRebalance` resource that includes the annotation, an optimization proposal is not generated. In other words, the resource is created, but is not executed immediately by the Strimzi Cluster Operator. Rebalancing templates do not need to include the mode or brokers properties.

You can create any number of templates, however, typically you would create two, one for upscale and one for downscale operations. You can also create a single template and use it for both upscale and downscale operations.

Procedure

1. Create a YAML configuration describing your rebalancing templates.

   Take note of the resource names you configure. You will need to specify them in your `Kafka` resource.

   The following is an example of a `KafkaRebalance` resource with some typical goals configured. Ensure that you set goals appropriately for your use case.

   ```
   #...
   kind: KafkaRebalance
   metadata:
     name: [***REBALANCE TEMPLATE NAME***]
     annotations:
       strimzi.io/rebalance-template: "true"
   spec:
     goals:
       - CpuCapacityGoal
       - NetworkInboundCapacityGoal
       - DiskCapacityGoal
       - RackAwareGoal
       - MinTopicLeadersPerBrokerGoal
       - NetworkOutboundCapacityGoal
       - ReplicaCapacityGoal
     skipHardGoalCheck: true
   ```

2. Create the resource.

   ```
   kubectl apply --filename [***YAML CONFIG***] --namespace [***NAMESPACE***]
   ```

3. Enable auto-rebalancing in the Kafka resource.

   To do this, add the spec.cruiseControl.autoRebalance property.

   ```
   #...
   kind: Kafka
   metadata:
     name: my-cluster
   ```

```
spec:
  cruiseControl:
    autoRebalance:
      - mode: add-brokers
        template:
          name: [***REBALANCE TEMPLATE NAME***]
      - mode: remove-brokers
        template:
          name: [***REBALANCE TEMPLATE NAME***]
```

- spec.CruiseControl.autoRebalance.mode - Specifies the mode for automatically rebalancing when brokers are added or removed. Supported modes are add-brokers and remove-brokers.
- spec.CruiseControl.autoRebalance.tempalate - The KafkaRebalance resource to use for rebalancing.

What to do next

Scale your cluster. Complete either Upscaling Kafka brokers on page 6 or Downscaling Kafka brokers on page 9.

Upscaling Kafka brokers

Complete these steps to upscale the number of brokers in your Kafka cluster

Before you begin

- Ensure that Cruise Control is deployed in your cluster. See Deploying Cruise Control.
- You can control the IDs of newly added brokers with the strimzi.io/next-node-ids annotation. See Configuring Kafka broker node IDs.

Procedure

1. Add new brokers to your cluster.

   This is done by updating the replica count in your `KafkaNodePool` resources, which you can do in the following two ways.

   - Update the value of spec.replicas directly in the resource and apply your changes.
   - Scale the resource with kubectl scale.

   ```
   kubectl scale kafkanodepool [***NODE POOL NAME***] \
     --namespace [***NAMESPACE***] \
     --replicas=[***COUNT***]
   ```

2. Wait until readiness checks are complete and all new brokers are in a Ready state.

   Use the following command to monitor cluster state.

   ```
   kubectl get pods --namespace [***NAMESPACE***] --output wide --watch
   ```

   Note: New brokers are assigned the next available node IDs. If you configured an ID range to use, the IDs are selected from the configured range.

3. Rebalance your cluster.

   The actions you take differ depending on whether auto-rebalancing is enabled.

   For Auto-rebalancing enabled

   With auto-rebalancing, your cluster is automatically rebalanced based on the rebalancing templates you configured when you enabled auto-rebalancing. Check rebalance status by viewing your `Kafka` resource.

   ```
   kubectl get kafka [***CLUSTER NAME***] \
   ```

```
    --namespace [***NAMESPACE***] \
    --output yaml
```

You can find the current state and status of the rebalance operation in status.autoRebalance.

```
#...
kind: Kafka
status:
  autoRebalance:
    lastTransitionTime: 2025-02-02T10:57:45.817792952Z
    state: RebalanceOnScaleUp
    modes:
      - mode: add-brokers
        brokers: [1,2,3,4,5,6]
      - mode: remove-brokers
        brokers: [1,2,3,4,5,6]
```

If the rebalancing is finished, status.autoRebalance.state will be Idle and status.conditions.type will be Ready. Additionally, status.autorebalance.modes will no longer be in the output.

```
#...
kind: Kafka
status:
  autoRebalance:
    lastTransitionTime: "2025-02-02T11:00:00.071202549Z"
    state: Idle
  clusterId: k3Ll-qP0RnSDn7m5SP01ag
  conditions:
    - lastTransitionTime: "2025-02-02T11:00:00.071309841Z"
      status: "True"
      type: Ready
```

For Auto-rebalancing disabled

a. Create a YAML configuration describing your `KafkaRebalance` resource. For example:

```
apiVersion: kafka.strimzi.io/v1
kind: KafkaRebalance
metadata:
  name: my-rebalance
  labels:
    strimzi.io/cluster: my-cluster
spec:
  mode: add-brokers
  brokers: [3]
```

- Note down the resource name you specify in metadata.name. You will need to specify the name in the steps that follow.
- spec.mode specifies the mode to use for rebalancing. The add-brokers mode is used after upscaling a cluster. This mode moves replicas to newly added brokers.
- spec.brokers is a list of the new broker IDs.

b. Create the resource.

```
kubectl apply --filename [***YAML CONFIG***] --names
pace [***NAMESPACE***]
```

Creating the resource generates an optimization proposal from Cruise Control.

c. View the generated optimization proposal.

```
kubectl get kafkarebalance [***RESOURCE NAME***] \
```

```
  --namespace [***NAMESPACE***] \
  --output yaml
```

The generated optimization proposal will be similar to the following example.

```
#...
status:
  conditions:
    - lastTransitionTime: "2025-02-18T09:02:59.731337639Z"
      status: "True"
      type: ProposalReady
  observedGeneration: 1
  optimizationResult:
    afterBeforeLoadConfigMap: my-rebalance
    dataToMoveMB: 3
    excludedBrokersForLeadership: []
    excludedBrokersForReplicaMove: []
    excludedTopics: []
    intraBrokerDataToMoveMB: 0
    monitoredPartitionsPercentage: 100
    numIntraBrokerReplicaMovements: 0
    numLeaderMovements: 0
    numReplicaMovements: 87
    onDemandBalancednessScoreAfter: 86.20996100734439
    onDemandBalancednessScoreBefore: 82.30801521650507
    provisionRecommendation: ""
    provisionStatus: RIGHT_SIZED
    recentWindows: 1
  sessionId: b5ee9d4a-779c-4dcb-a2e3-1ea6ece106d2
```

- status.conditions.type - Shows whether the proposal is ready. ProposalReady means that the proposal is ready.
- status.optimizationResult - Describes the recommended optimization.

d. Approve or refresh the proposal.

Approving or refreshing the proposal is done using annotation. Approve the proposal if you are satisfied with it. Approving the proposal starts the rebalance process. Refresh the proposal if you are not satisfied with it or if it became outdated.

Approve

```
kubectl annotate kafkarebalance [***RESOURCE NAME***] \
  strimzi.io/rebalance=approve \
  --namespace [***NAMESPACE***]
```

Refresh

```
kubectl annotate kafkarebalance [***RESOURCE NAME***] \
  strimzi.io/rebalance=refresh \
  --namespace [***NAMESPACE***]
```

e. Monitor the rebalance process.

```
kubectl get kafkarebalance [***RESOURCE NAME***] \
  --namespace [***NAMESPACE***] \
  --output yaml
```

The status.conditions.type property in the output shows the current status of the rebalance process. While the rebalance is in progress, the type will be Rebalancing. For example:

```
#...
```

```
status:
  conditions:
    - lastTransitionTime: "2025-02-18T09:21:12.077981255Z"
      status: "True"
      type: Rebalancing
  observedGeneration: 1
```

The rebalance is finished once status.conditions.type changes to Ready.

f. **Optional:** Delete the `KafkaRebalance` resource.

After the rebalance process completes successfully, you can choose to delete the `KafkaRebalance` resource if you no longer need it. Alternatively, you can keep the resource for later use. If you keep the resource, ensure that you refresh the generated proposal before initiating a new rebalance.

Downscaling Kafka brokers

Complete these steps to downscale the number of brokers in your Kafka cluster.

Before you begin

Ensure that Cruise Control is deployed in your cluster. See Deploying Cruise Control.

Procedure

1. Choose and annotate the broker you want to remove from the cluster.

   The ID of the broker that you want to remove must be set with an annotation on the `KafkaNodePool` resource. Annotating the broker ID tells the Strimzi Cluster operator that this is the broker that should be removed when a downscale operation is initiated.

   ```
   kubectl annotate kafkanodepool [***RESOURCE NAME***] \
     --namespace [***NAMESPACE***] \
     strimzi.io/remove-node-ids="[[***BROKER IDS***]]"
   ```

2. **If auto-rebalancing is disabled:** Rebalance your cluster with Cruise Control.

   When downscaling the cluster, Cruise Control rebalancing is used to move data from brokers.

   a) Create a YAML configuration describing your `KafkaRebalance` resource.

   For example:

   ```
   apiVersion: kafka.strimzi.io/v1
   kind: KafkaRebalance
   metadata:
     name: my-downscale
     labels:
       strimzi.io/cluster: my-cluster
   spec:
     mode: remove-brokers
     brokers: [3]
   ```

   - Note down the resource name you specify in metadata.name. You will need to specify the name in the steps that follow.
   - spec.mode specifies the mode to use for rebalancing. The remove-brokers mode is used before downscaling a cluster. This mode moves replicas from brokers that will be deleted.
   - spec.brokers is a list of the broker IDs that will be removed from the cluster.

b) Create the resource.

```
kubectl apply --filename [***YAML CONFIG***] --names
pace [***NAMESPACE***]
```

Creating the resource generates an optimization proposal from Cruise Control.

c) View the generated optimization proposal.

```
kubectl get kafkarebalance [***RESOURCE NAME***] \
  --namespace [***NAMESPACE***] \
  --output yaml
```

The generated optimization proposal will be similar to the following example.

```
#...
status:
  conditions:
    - lastTransitionTime: "2025-02-18T13:07:31.983312926Z"
      status: "True"
      type: ProposalReady
  observedGeneration: 1
  optimizationResult:
    afterBeforeLoadConfigMap: my-downscale
    dataToMoveMB: 6
    excludedBrokersForLeadership: []
    excludedBrokersForReplicaMove: []
    excludedTopics: []
    intraBrokerDataToMoveMB: 0
    monitoredPartitionsPercentage: 100
    numIntraBrokerReplicaMovements: 0
    numLeaderMovements: 5
    numReplicaMovements: 89
    onDemandBalancednessScoreAfter: 89.4347095948149
    onDemandBalancednessScoreBefore: 76.88845859311024
    provisionRecommendation: ""
    provisionStatus: RIGHT_SIZED
    recentWindows: 1
  sessionId: 591260ea-8830-4645-bff8-9662d0063010
```

d) Approve or refresh the proposal.

Approving or refreshing the proposal is done using annotation. Approve the proposal if you are satisfied with it. Approving the proposal starts the rebalance process. Refresh the proposal if you are not satisfied with it or if it became outdated.

Approve

```
kubectl annotate kafkarebalance [***RESOURCE NAME***] \
  strimzi.io/rebalance=approve \
  --namespace [***NAMESPACE***]
```

Refresh

```
kubectl annotate kafkarebalance [***RESOURCE NAME***] \
  strimzi.io/rebalance=refresh \
  --namespace [***NAMESPACE***]
```

e) Monitor the rebalance process.

```
kubectl get kafkarebalance [***RESOURCE NAME***] \
  --namespace [***NAMESPACE***] \
```

```
  --output yaml
```

The status.conditions.type property in the output shows the current status of the rebalance process. While the rebalance is in progress, the type will be Rebalancing. For example:

```
#...
status:
  conditions:
    - lastTransitionTime: "2025-02-18T13:17:44.722605626Z"
      status: "True"
      type: Rebalancing
  observedGeneration: 1
```

The rebalance is finished once status.conditions.type changes to Ready.

Warning: Before you continue with the next step, ensure that all data is moved from the brokers that will be deleted.

3. Remove the Kafka brokers from your cluster.

   This is done by updating the replica count in your `KafkaNodePool` resources, which you can do in the following two ways.

   - Update the value of spec.replicas directly in the resource and apply your changes.
   - Scale the resource with kubectl scale.

   ```
   kubectl scale kafkanodepool [***NODE POOL NAME***] \
     --namespace [***NAMESPACE***] \
     --replicas=[***COUNT***]
   ```

   If auto-rebalancing is enabled, a rebalance with Cruise Control is automatically triggered when you remove the brokers. The rebalance runs before the Strimzi Cluster Operator removes the brokers.

   The Strimzi Cluster Operator blocks the downscale operation if there are still replicas on the broker targeted for removal. If required, you can bypass this blocking mechanism.

4. **If auto-rebalancing is enabled:** Check rebalance status by viewing your `Kafka` resource.

   ```
   kubectl get kafka [***CLUSTER NAME***] \
     --namespace [***NAMESPACE***] \
     --output yaml
   ```

   You can find the current state and status of the rebalance operation in status.autoRebalance.

   ```
   #...
   kind: Kafka
   status:
     autoRebalance:
       lastTransitionTime: 2025-02-02T10:57:45.817792952Z
       state: RebalanceOnScaleDown
       modes:
         - mode: add-brokers
           brokers: [1,2,3,4,5,6]
         - mode: remove-brokers
           brokers: [1,2,3,4,5,6]
   ```

   If the rebalancing is finished, status.autoRebalance.state will be Idle and status.conditions.type will be Ready. Additionally, status.autorebalance.modes will no longer be in the output.

   ```
   #...
   kind: Kafka
   status:
     autoRebalance:
   ```

```
    lastTransitionTime: "2025-02-02T11:00:00.071202549Z"
    state: Idle
  clusterId: k3Ll-qP0RnSDn7m5SP01ag
  conditions:
    - lastTransitionTime: "2025-02-02T11:00:00.071309841Z"
      status: "True"
      type: Ready
```

5. Remove the annotation from the `KafkaNodePool` resource.

   This annotation was added in a previous step and was used to influence which node should be removed from the cluster.

```
kubectl annotate kafkanodepool [***NODE POOL NAME***] \
  --namespace [***NAMESPACE***] \
  strimzi.io/remove-node-ids-
```

Related Information

Skipping checks on scale-down operations | Strimzi