Red Hat AMQ Streams 2.6

AMQ Streams API Reference

Configure a deployment of AMQ Streams 2.6 on OpenShift Container Platform

Last Updated: 2023-12-06

Red Hat AMQ Streams 2.6 AMQ Streams API Reference

Configure a deployment of AMQ Streams 2.6 on OpenShift Container Platform

Legal Notice

Abstract

Use the configuration properties of the AMQ Streams API to fine-tune your deployment.

Table of Contents

MAKING OPEN SOURCE MORE INCLUSIVE

Red Hat is committed to replacing problematic language in our code, documentation, and web properties. We are beginning with these four terms: master, slave, blacklist, and whitelist. Because of the enormity of this endeavor, these changes will be implemented gradually over several upcoming releases. For more details, see our CTO Chris Wright's message.

CHAPTER 1. USING SCHEMA PROPERTIES TO CONFIGURE CUSTOM RESOURCES

Custom resources offer a flexible way to manage and fine-tune the operation of AMQ Streams components using configuration properties. This reference guide describes common configuration properties that apply to multiple custom resources, as well as the configuration properties available for each custom resource schema available with AMQ Streams. Where appropriate, expanded descriptions of properties and examples of how they are configured are provided.

The properties defined for each schema provide a structured and organized way to specify configuration for the custom resources. Whether it's adjusting resource allocation or specifying access controls, the properties in the schemas allow for a granular level of configuration. For example, you can use the properties of the **KafkaClusterSpec** schema to specify the type of storage for a Kafka cluster or add listeners that provide secure access to Kafka brokers.

Some property options within a schema may be constrained, as indicated in the property descriptions. These constraints define specific options or limitations on the values that can be assigned to those properties. Constraints ensure that the custom resources are configured with valid and appropriate values.

CHAPTER 2. COMMON CONFIGURATION PROPERTIES

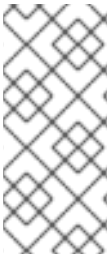
Use Common configuration properties to configure AMQ Streams custom resources. You add common configuration properties to a custom resource like any other supported configuration for that resource.

2.1. REPLICAS

Use the **replicas** property to configure replicas.

The type of replication depends on the resource.

- **KafkaTopic** uses a replication factor to configure the number of replicas of each partition within a Kafka cluster.
- Kafka components use replicas to configure the number of pods in a deployment to provide better availability and scalability.



NOTE

When running a Kafka component on OpenShift it may not be necessary to run multiple replicas for high availability. When the node where the component is deployed crashes, OpenShift will automatically reschedule the Kafka component pod to a different node. However, running Kafka components with multiple replicas can provide faster failover times as the other nodes will be up and running.

2.2. BOOTSTRAPSERVERS

Use the **bootstrapServers** property to configure a list of bootstrap servers.

The bootstrap server lists can refer to Kafka clusters that are not deployed in the same OpenShift cluster. They can also refer to a Kafka cluster not deployed by AMQ Streams.

If on the same OpenShift cluster, each list must ideally contain the Kafka cluster bootstrap service which is named ***CLUSTER-NAME*-kafka-bootstrap** and a port number. If deployed by AMQ Streams but on different OpenShift clusters, the list content depends on the approach used for exposing the clusters (routes, ingress, nodeports or loadbalancers).

When using Kafka with a Kafka cluster not managed by AMQ Streams, you can specify the bootstrap servers list according to the configuration of the given cluster.

2.3. SSL (SUPPORTED TLS VERSIONS AND CIPHER SUITES)

You can incorporate SSL configuration and cipher suite specifications to further secure TLS-based communication between your client application and a Kafka cluster. In addition to the standard TLS configuration, you can specify a supported TLS version and enable cipher suites in the configuration for the Kafka broker. You can also add the configuration to your clients if you wish to limit the TLS versions and cipher suites they use. The configuration on the client must only use protocols and cipher suites that are enabled on the broker.

A cipher suite is a set of security mechanisms for secure connection and data transfer. For example, the cipher suite **TLS_AES_256_GCM_SHA384** is composed of the following mechanisms, which are used in conjunction with the TLS protocol:

- AES (Advanced Encryption Standard) encryption (256-bit key)

- GCM (Galois/Counter Mode) authenticated encryption
- SHA384 (Secure Hash Algorithm) data integrity protection

The combination is encapsulated in the **TLS_AES_256_GCM_SHA384** cipher suite specification.

The **ssl.enabled.protocols** property specifies the available TLS versions that can be used for secure communication between the cluster and its clients. The **ssl.protocol** property sets the default TLS version for all connections, and it must be chosen from the enabled protocols. Use the **ssl.endpoint.identification.algorithm** property to enable or disable hostname verification (configurable only in components based on Kafka clients - Kafka Connect, MirrorMaker 1/2, and Kafka Bridge).

Example SSL configuration

```
# ...
config:
  ssl.cipher.suites: TLS_AES_256_GCM_SHA384,
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (1)
  ssl.enabled.protocols: TLSv1.3, TLSv1.2 (2)
  ssl.protocol: TLSv1.3 (3)
  ssl.endpoint.identification.algorithm: HTTPS (4)
# ...
```

1. Cipher suite specifications enabled.
2. TLS versions supported.
3. Default TLS version is **TLSv1.3**. If a client only supports TLSv1.2, it can still connect to the broker and communicate using that supported version, and vice versa if the configuration is on the client and the broker only supports TLSv1.2.
4. Hostname verification is enabled by setting to **HTTPS**. An empty string disables the verification.

2.4. TRUSTEDCERTIFICATES

Having set **tls** to configure TLS encryption, use the **trustedCertificates** property to provide a list of secrets with key names under which the certificates are stored in X.509 format.

You can use the secrets created by the Cluster Operator for the Kafka cluster, or you can create your own TLS certificate file, then create a **Secret** from the file:

```
oc create secret generic MY-SECRET \
--from-file=MY-TLS-CERTIFICATE-FILE.crt
```

Example TLS encryption configuration

```
tls:
  trustedCertificates:
    - secretName: my-cluster-cluster-cert
      certificate: ca.crt
    - secretName: my-cluster-cluster-cert
      certificate: ca2.crt
```

If certificates are stored in the same secret, it can be listed multiple times.

If you want to enable TLS encryption, but use the default set of public certification authorities shipped with Java, you can specify **trustedCertificates** as an empty array:

Example of enabling TLS with the default Java certificates

```
tls:
  trustedCertificates: []
```

For information on configuring mTLS authentication, see the **KafkaClientAuthenticationTls** schema reference.

2.5. RESOURCES

Configure resource *requests* and *limits* to control resources for AMQ Streams containers. You can specify requests and limits for **memory** and **cpu** resources. The requests should be enough to ensure a stable performance of Kafka.

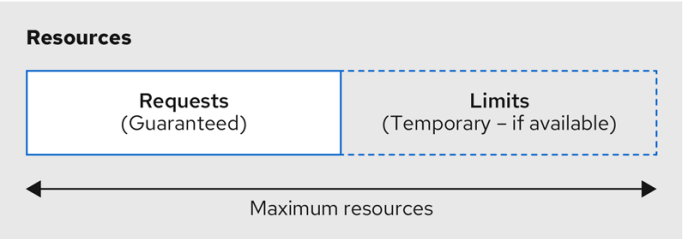
How you configure resources in a production environment depends on a number of factors. For example, applications are likely to be sharing resources in your OpenShift cluster.

For Kafka, the following aspects of a deployment can impact the resources you need:

- Throughput and size of messages
- The number of network threads handling messages
- The number of producers and consumers
- The number of topics and partitions

The values specified for resource requests are reserved and always available to the container. Resource limits specify the maximum resources that can be consumed by a given container. The amount between the request and limit is not reserved and might not be always available. A container can use the resources up to the limit only when they are available. Resource limits are temporary and can be reallocated.

Resource requests and limits



212_Streams_0322

If you set limits without requests or vice versa, OpenShift uses the same value for both. Setting equal requests and limits for resources guarantees quality of service, as OpenShift will not kill containers unless they exceed their limits.

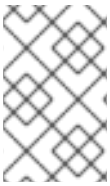
You can configure resource requests and limits for one or more supported resources.

Example resource configuration

```
apiVersion: kafka.strimzi.io/v1beta2
kind: Kafka
metadata:
  name: my-cluster
spec:
  kafka:
    #...
    resources:
      requests:
        memory: 64Gi
        cpu: "8"
      limits:
        memory: 64Gi
        cpu: "12"
  entityOperator:
    #...
    topicOperator:
      #...
      resources:
        requests:
          memory: 512Mi
          cpu: "1"
        limits:
          memory: 512Mi
          cpu: "1"
```

Resource requests and limits for the Topic Operator and User Operator are set in the **Kafka** resource.

If the resource request is for more than the available free resources in the OpenShift cluster, the pod is not scheduled.



NOTE

AMQ Streams uses the OpenShift syntax for specifying **memory** and **cpu** resources. For more information about managing computing resources on OpenShift, see Managing Compute Resources for Containers.

Memory resources

When configuring memory resources, consider the total requirements of the components. Kafka runs inside a JVM and uses an operating system page cache to store message data before writing to disk. The memory request for Kafka should fit the JVM heap and page cache. You can configure the **jvmOptions** property to control the minimum and maximum heap size.

Other components don't rely on the page cache. You can configure memory resources without configuring the **jvmOptions** to control the heap size.

Memory requests and limits are specified in megabytes, gigabytes, mebibytes, and gibibytes. Use the following suffixes in the specification:

- **M** for megabytes
- **G** for gigabytes
- **Mi** for mebibytes

- **Gi** for gibibytes

Example resources using different memory units

```
# ...
resources:
  requests:
    memory: 512Mi
  limits:
    memory: 2Gi
# ...
```

For more details about memory specification and additional supported units, see Meaning of memory.

CPU resources

A CPU request should be enough to give a reliable performance at any time. CPU requests and limits are specified as *cores* or *millicpus/millicores*.
CPU cores are specified as integers (**5** CPU core) or decimals (**2.5** CPU core). 1000 *millicores* is the same as **1** CPU core.

Example CPU units

```
# ...
resources:
  requests:
    cpu: 500m
  limits:
    cpu: 2.5
# ...
```

The computing power of 1 CPU core may differ depending on the platform where OpenShift is deployed.

For more information on CPU specification, see Meaning of CPU.

2.6. IMAGE

Use the **image** property to configure the container image used by the component.

Overriding container images is recommended only in special situations where you need to use a different container registry or a customized image.

For example, if your network does not allow access to the container repository used by AMQ Streams, you can copy the AMQ Streams images or build them from the source. However, if the configured image is not compatible with AMQ Streams images, it might not work properly.

A copy of the container image might also be customized and used for debugging.

You can specify which container image to use for a component using the **image** property in the following resources:

- **Kafka.spec.kafka**

- **Kafka.spec.zookeeper**
- **Kafka.spec.entityOperator.topicOperator**
- **Kafka.spec.entityOperator.userOperator**
- **Kafka.spec.entityOperator.tlsSidecar**
- **KafkaConnect.spec**
- **KafkaMirrorMaker.spec**
- **KafkaMirrorMaker2.spec**
- **KafkaBridge.spec**

Configuring the **image** property for Kafka, Kafka Connect, and Kafka MirrorMaker

Kafka, Kafka Connect, and Kafka MirrorMaker support multiple versions of Kafka. Each component requires its own image. The default images for the different Kafka versions are configured in the following environment variables:

- **STRIMZI_KAFKA_IMAGES**
- **STRIMZI_KAFKA_CONNECT_IMAGES**
- **STRIMZI_KAFKA_MIRROR_MAKER_IMAGES**

These environment variables contain mappings between the Kafka versions and their corresponding images. The mappings are used together with the **image** and **version** properties:

- If neither **image** nor **version** are given in the custom resource then the **version** will default to the Cluster Operator's default Kafka version, and the image will be the one corresponding to this version in the environment variable.
- If **image** is given but **version** is not, then the given image is used and the **version** is assumed to be the Cluster Operator's default Kafka version.
- If **version** is given but **image** is not, then the image that corresponds to the given version in the environment variable is used.
- If both **version** and **image** are given, then the given image is used. The image is assumed to contain a Kafka image with the given version.

The **image** and **version** for the different components can be configured in the following properties:

- For Kafka in **spec.kafka.image** and **spec.kafka.version**.
- For Kafka Connect and Kafka MirrorMaker in **spec.image** and **spec.version**.

WARNING

It is recommended to provide only the **version** and leave the **image** property unspecified. This reduces the chance of making a mistake when configuring the custom resource. If you need to change the images used for different versions of Kafka, it is preferable to configure the Cluster Operator's environment variables.

Configuring the image property in other resources

For the **image** property in the other custom resources, the given value will be used during deployment. If the **image** property is missing, the **image** specified in the Cluster Operator configuration will be used. If the **image** name is not defined in the Cluster Operator configuration, then the default value will be used.

- For Topic Operator:
  1. Container image specified in the **STRIMZI_DEFAULT_TOPIC_OPERATOR_IMAGE** environment variable from the Cluster Operator configuration.
  2. **registry.redhat.io/amq-streams/strimzi-rhel8-operator:2.6.0** container image.
- For User Operator:
  1. Container image specified in the **STRIMZI_DEFAULT_USER_OPERATOR_IMAGE** environment variable from the Cluster Operator configuration.
  2. **registry.redhat.io/amq-streams/strimzi-rhel8-operator:2.6.0** container image.
- For Entity Operator TLS sidecar:
  1. Container image specified in the **STRIMZI_DEFAULT_TLS_SIDECAR_ENTITY_OPERATOR_IMAGE** environment variable from the Cluster Operator configuration.
  2. **registry.redhat.io/amq-streams/kafka-36-rhel8:2.6.0** container image.
- For Kafka Exporter:
  1. Container image specified in the **STRIMZI_DEFAULT_KAFKA_EXPORTER_IMAGE** environment variable from the Cluster Operator configuration.
  2. **registry.redhat.io/amq-streams/kafka-36-rhel8:2.6.0** container image.
- For Kafka Bridge:
  1. Container image specified in the **STRIMZI_DEFAULT_KAFKA_BRIDGE_IMAGE** environment variable from the Cluster Operator configuration.
  2. **registry.redhat.io/amq-streams/bridge-rhel8:2.6.0** container image.
- For Kafka broker initializer:
  1. Container image specified in the **STRIMZI_DEFAULT_KAFKA_INIT_IMAGE** environment variable from the Cluster Operator configuration.

2. **registry.redhat.io/amq-streams/strimzi-rhel8-operator:2.6.0** container image.

Example container image configuration

```
apiVersion: kafka.strimzi.io/v1beta2
kind: Kafka
metadata:
  name: my-cluster
spec:
  kafka:
    # ...
    image: my-org/my-image:latest
    # ...
  zookeeper:
    # ...
```

2.7. LIVENESSPROBE AND READINESSPROBE HEALTHCHECKS

Use the **livenessProbe** and **readinessProbe** properties to configure healthcheck probes supported in AMQ Streams.

Healthchecks are periodical tests which verify the health of an application. When a Healthcheck probe fails, OpenShift assumes that the application is not healthy and attempts to fix it.

For more details about the probes, see Configure Liveness and Readiness Probes.

Both **livenessProbe** and **readinessProbe** support the following options:

- **initialDelaySeconds**
- **timeoutSeconds**
- **periodSeconds**
- **successThreshold**
- **failureThreshold**

Example of liveness and readiness probe configuration

```
# ...
readinessProbe:
  initialDelaySeconds: 15
  timeoutSeconds: 5
livenessProbe:
  initialDelaySeconds: 15
  timeoutSeconds: 5
# ...
```

For more information about the **livenessProbe** and **readinessProbe** options, see the Probe schema reference.

2.8. METRICSCONFIG

Use the **metricsConfig** property to enable and configure Prometheus metrics.

The **metricsConfig** property contains a reference to a ConfigMap that has additional configurations for the Prometheus JMX Exporter. AMQ Streams supports Prometheus metrics using Prometheus JMX exporter to convert the JMX metrics supported by Apache Kafka and ZooKeeper to Prometheus metrics.

To enable Prometheus metrics export without further configuration, you can reference a ConfigMap containing an empty file under **metricsConfig.valueFrom.configMapKeyRef.key**. When referencing an empty file, all metrics are exposed as long as they have not been renamed.

Example ConfigMap with metrics configuration for Kafka

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: my-configmap
data:
  my-key: |
    lowercaseOutputName: true
    rules:
    # Special cases and very specific rules
    - pattern: kafka.server<type=(.+), name=(.+), clientId=(.+), topic=(.+), partition=(.*)><>Value
      name: kafka_server_$1_$2
      type: GAUGE
      labels:
       clientId: "$3"
       topic: "$4"
       partition: "$5"
    # further configuration
```

Example metrics configuration for Kafka

```
apiVersion: kafka.strimzi.io/v1beta2
kind: Kafka
metadata:
  name: my-cluster
spec:
  kafka:
    # ...
    metricsConfig:
      type: jmxPrometheusExporter
      valueFrom:
        configMapKeyRef:
          name: my-config-map
          key: my-key
    # ...
  zookeeper:
    # ...
```

When metrics are enabled, they are exposed on port 9404.

When the **metricsConfig** (or deprecated **metrics**) property is not defined in the resource, the Prometheus metrics are disabled.

For more information about setting up and deploying Prometheus and Grafana, see Introducing Metrics to Kafka.

2.9. JVMOPTIONS

The following AMQ Streams components run inside a Java Virtual Machine (JVM):

- Apache Kafka
- Apache ZooKeeper
- Apache Kafka Connect
- Apache Kafka MirrorMaker
- AMQ Streams Kafka Bridge

To optimize their performance on different platforms and architectures, you configure the **jvmOptions** property in the following resources:

- **Kafka.spec.kafka**
- **Kafka.spec.zookeeper**
- **Kafka.spec.entityOperator.userOperator**
- **Kafka.spec.entityOperator.topicOperator**
- **Kafka.spec.cruiseControl**
- **KafkaNodePool.spec**
- **KafkaConnect.spec**
- **KafkaMirrorMaker.spec**
- **KafkaMirrorMaker2.spec**
- **KafkaBridge.spec**

You can specify the following options in your configuration:

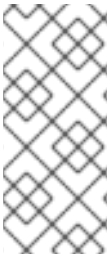
-Xms
: Minimum initial allocation heap size when the JVM starts

-Xmx
: Maximum heap size

-XX
: Advanced runtime options for the JVM

javaSystemProperties
: Additional system properties

gcLoggingEnabled
: Enables garbage collector logging



NOTE

The units accepted by JVM settings, such as **-Xmx** and **-Xms**, are the same units accepted by the JDK **java** binary in the corresponding image. Therefore, **1g** or **1G** means 1,073,741,824 bytes, and **Gi** is not a valid unit suffix. This is different from the units used for memory requests and limits, which follow the OpenShift convention where **1G** means 1,000,000,000 bytes, and **1Gi** means 1,073,741,824 bytes.

-Xms and -Xmx options

In addition to setting memory request and limit values for your containers, you can use the **-Xms** and **-Xmx** JVM options to set specific heap sizes for your JVM. Use the **-Xms** option to set an initial heap size and the **-Xmx** option to set a maximum heap size.

Specify heap size to have more control over the memory allocated to your JVM. Heap sizes should make the best use of a container's memory limit (and request) without exceeding it. Heap size and any other memory requirements need to fit within a specified memory limit. If you don't specify heap size in your configuration, but you configure a memory resource limit (and request), the Cluster Operator imposes default heap sizes automatically. The Cluster Operator sets default maximum and minimum heap values based on a percentage of the memory resource configuration.

The following table shows the default heap values.

Table 2.1. Default heap settings for components

Component	Percent of available memory allocated to the heap	Maximum limit
Kafka	50%	5 GB
ZooKeeper	75%	2 GB
Kafka Connect	75%	None
MirrorMaker 2	75%	None
MirrorMaker	75%	None
Cruise Control	75%	None
Kafka Bridge	50%	31 Gi

If a memory limit (and request) is not specified, a JVM's minimum heap size is set to **128M**. The JVM's maximum heap size is not defined to allow the memory to increase as needed. This is ideal for single node environments in test and development.

Setting an appropriate memory request can prevent the following:

- OpenShift killing a container if there is pressure on memory from other pods running on the node.

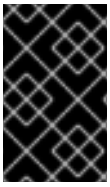
- OpenShift scheduling a container to a node with insufficient memory. If **-Xms** is set to **-Xmx**, the container will crash immediately; if not, the container will crash at a later time.

In this example, the JVM uses 2 GiB (=2,147,483,648 bytes) for its heap. Total JVM memory usage can be a lot more than the maximum heap size.

Example -Xmx and -Xms configuration

```
# ...
jvmOptions:
  "-Xmx": "2g"
  "-Xms": "2g"
# ...
```

Setting the same value for initial (**-Xms**) and maximum (**-Xmx**) heap sizes avoids the JVM having to allocate memory after startup, at the cost of possibly allocating more heap than is really needed.



IMPORTANT

Containers performing lots of disk I/O, such as Kafka broker containers, require available memory for use as an operating system page cache. For such containers, the requested memory should be significantly higher than the memory used by the JVM.

-XX option

-XX options are used to configure the **KAFKA_JVM_PERFORMANCE_OPTS** option of Apache Kafka.

Example -XX configuration

```
jvmOptions:
  "-XX":
    "UseG1GC": true
    "MaxGCPauseMillis": 20
    "InitiatingHeapOccupancyPercent": 35
    "ExplicitGCInvokesConcurrent": true
```

JVM options resulting from the -XX configuration

```
-XX:+UseG1GC -XX:MaxGCPauseMillis=20 -XX:InitiatingHeapOccupancyPercent=35 -
XX:+ExplicitGCInvokesConcurrent -XX:-UseParNewGC
```

NOTE

When no **-XX** options are specified, the default Apache Kafka configuration of **KAFKA_JVM_PERFORMANCE_OPTS** is used.

javaSystemProperties

javaSystemProperties are used to configure additional Java system properties, such as debugging utilities.

Example javaSystemProperties configuration

```
jvmOptions:
  javaSystemProperties:
    - name: javax.net.debug
      value: ssl
```

For more information about the **jvmOptions**, see the **JvmOptions** schema reference.

2.10. GARBAGE COLLECTOR LOGGING

The **jvmOptions** property also allows you to enable and disable garbage collector (GC) logging. GC logging is disabled by default. To enable it, set the **gcLoggingEnabled** property as follows:

Example GC logging configuration

```
# ...
jvmOptions:
  gcLoggingEnabled: true
# ...
```

CHAPTER 3. KAFKA SCHEMA REFERENCE

Property	Description
spec	The specification of the Kafka and ZooKeeper clusters, and Topic Operator.
KafkaSpec	
status	The status of the Kafka and ZooKeeper clusters, and Topic Operator.
KafkaStatus	

CHAPTER 4. KAFKASPEC SCHEMA REFERENCE

Used in: **Kafka**

Property	Description
kafka	Configuration of the Kafka cluster.
KafkaClusterSpec	
zookeeper	Configuration of the ZooKeeper cluster.
ZookeeperClusterSpec	
entityOperator	Configuration of the Entity Operator.
EntityOperatorSpec	
clusterCa	Configuration of the cluster certificate authority.
CertificateAuthority	
clientsCa	Configuration of the clients certificate authority.
CertificateAuthority	
cruiseControl	Configuration for Cruise Control deployment. Deploys a Cruise Control instance when specified.
CruiseControlSpec	
jmxTrans	**The jmxTrans property has been deprecated.** JMXTrans is deprecated and related resources removed in AMQ Streams 2.5. As of AMQ Streams 2.5, JMXTrans is not supported anymore and this option is ignored.
JmxTransSpec	
kafkaExporter	Configuration of the Kafka Exporter. Kafka Exporter can provide additional metrics, for example lag of consumer group at topic/partition.
KafkaExporterSpec	
maintenanceTimeWindows	A list of time windows for maintenance tasks (that is, certificates renewal). Each time window is defined by a cron expression.
string array	

CHAPTER 5. KAFKACLUSTERSPEC SCHEMA REFERENCE

Used in: **KafkaSpec**

Full list of **KafkaClusterSpec** schema properties

Configures a Kafka cluster.

5.1. LISTENERS

Use the **listeners** property to configure listeners to provide access to Kafka brokers.

Example configuration of a plain (unencrypted) listener without authentication

```
apiVersion: kafka.strimzi.io/v1beta2
kind: Kafka
spec:
  kafka:
    # ...
    listeners:
      - name: plain
        port: 9092
        type: internal
        tls: false
    # ...
  zookeeper:
    # ...
```

5.2. CONFIG

Use the **config** properties to configure Kafka broker options as keys.

The values can be one of the following JSON types:

- String
- Number
- Boolean

Exceptions

You can specify and configure the options listed in the Apache Kafka documentation.

However, AMQ Streams takes care of configuring and managing options related to the following, which cannot be changed:

- Security (encryption, authentication, and authorization)
- Listener configuration
- Broker ID configuration
- Configuration of log data directories

- Inter-broker communication
- ZooKeeper connectivity

Properties with the following prefixes cannot be set:

- **advertised.**
- **authorizer.**
- **broker.**
- **controller**
- **cruise.control.metrics.reporter.bootstrap.**
- **cruise.control.metrics.topic**
- **host.name**
- **inter.broker.listener.name**
- **listener.**
- **listeners.**
- **log.dir**
- **password.**
- **port**
- **process.roles**
- **sasl.**
- **security.**
- **servers,node.id**
- **ssl.**
- **super.user**
- **zookeeper.clientCnxnSocket**
- **zookeeper.connect**
- **zookeeper.set.acl**
- **zookeeper.ssl**

If the **config** property contains an option that cannot be changed, it is disregarded, and a warning message is logged to the Cluster Operator log file. All other supported options are forwarded to Kafka, including the following exceptions to the options configured by AMQ Streams:

- Any **ssl** configuration for supported TLS versions and cipher suites

- Configuration for the **zookeeper.connection.timeout.ms** property to set the maximum time allowed for establishing a ZooKeeper connection
- Cruise Control metrics properties:
  - **cruise.control.metrics.topic.num.partitions**
  - **cruise.control.metrics.topic.replication.factor**
  - **cruise.control.metrics.topic.retention.ms**
  - **cruise.control.metrics.topic.auto.create.retries**
  - **cruise.control.metrics.topic.auto.create.timeout.ms**
  - **cruise.control.metrics.topic.min.insync.replicas**
- Controller properties:
  - **controller.quorum.election.backoff.max.ms**
  - **controller.quorum.election.timeout.ms**
  - **controller.quorum.fetch.timeout.ms**

Example Kafka broker configuration

```
apiVersion: kafka.strimzi.io/v1beta2
kind: Kafka
metadata:
  name: my-cluster
spec:
  kafka:
    # ...
    config:
      num.partitions: 1
      num.recovery.threads.per.data.dir: 1
      default.replication.factor: 3
      offsets.topic.replication.factor: 3
      transaction.state.log.replication.factor: 3
      transaction.state.log.min.isr: 1
      log.retention.hours: 168
      log.segment.bytes: 1073741824
      log.retention.check.interval.ms: 300000
      num.network.threads: 3
      num.io.threads: 8
      socket.send.buffer.bytes: 102400
      socket.receive.buffer.bytes: 102400
      socket.request.max.bytes: 104857600
      group.initial.rebalance.delay.ms: 0
      zookeeper.connection.timeout.ms: 6000
    # ...
```

5.3. BROKERRACKINITIMAGE

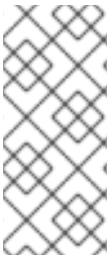
When rack awareness is enabled, Kafka broker pods use init container to collect the labels from the

OpenShift cluster nodes. The container image used for this container can be configured using the **brokerRackInitImage** property. When the **brokerRackInitImage** field is missing, the following images are used in order of priority:

1. Container image specified in **STRIMZI_DEFAULT_KAFKA_INIT_IMAGE** environment variable in the Cluster Operator configuration.
2. **registry.redhat.io/amq-streams/strimzi-rhel8-operator:2.6.0** container image.

Example brokerRackInitImage configuration

```
apiVersion: kafka.strimzi.io/v1beta2
kind: Kafka
metadata:
  name: my-cluster
spec:
  kafka:
    # ...
    rack:
      topologyKey: topology.kubernetes.io/zone
    brokerRackInitImage: my-org/my-image:latest
    # ...
```



NOTE

Overriding container images is recommended only in special situations, where you need to use a different container registry. For example, because your network does not allow access to the container registry used by AMQ Streams. In this case, you should either copy the AMQ Streams images or build them from the source. If the configured image is not compatible with AMQ Streams images, it might not work properly.

5.4. LOGGING

Kafka has its own configurable loggers, which include the following:

- **log4j.logger.org.I0Itec.zkclient.ZkClient**
- **log4j.logger.org.apache.zookeeper**
- **log4j.logger.kafka**
- **log4j.logger.org.apache.kafka**
- **log4j.logger.kafka.request.logger**
- **log4j.logger.kafka.network.Processor**
- **log4j.logger.kafka.server.KafkaApis**
- **log4j.logger.kafka.network.RequestChannel$**
- **log4j.logger.kafka.controller**
- **log4j.logger.kafka.log.LogCleaner**

- **log4j.logger.state.change.logger**
- **log4j.logger.kafka.authorizer.logger**

Kafka uses the Apache **log4j** logger implementation.

Use the **logging** property to configure loggers and logger levels.

You can set the log levels by specifying the logger and level directly (inline) or use a custom (external) ConfigMap. If a ConfigMap is used, you set **logging.valueFrom.configMapKeyRef.name** property to the name of the ConfigMap containing the external logging configuration. Inside the ConfigMap, the logging configuration is described using **log4j.properties**. Both **logging.valueFrom.configMapKeyRef.name** and **logging.valueFrom.configMapKeyRef.key** properties are mandatory. A ConfigMap using the exact logging configuration specified is created with the custom resource when the Cluster Operator is running, then recreated after each reconciliation. If you do not specify a custom ConfigMap, default logging settings are used. If a specific logger value is not set, upper-level logger settings are inherited for that logger. For more information about log levels, see Apache logging services.

Here we see examples of **inline** and **external** logging. The **inline** logging specifies the root logger level. You can also set log levels for specific classes or loggers by adding them to the loggers property.

Inline logging

```
apiVersion: kafka.strimzi.io/v1beta2
kind: Kafka
spec:
  # ...
  kafka:
    # ...
    logging:
      type: inline
      loggers:
        kafka.root.logger.level: INFO
        log4j.logger.kafka.coordinator.transaction: TRACE
        log4j.logger.kafka.log.LogCleanerManager: DEBUG
        log4j.logger.kafka.request.logger: DEBUG
        log4j.logger.io.strimzi.kafka.oauth: DEBUG
        log4j.logger.org.openpolicyagents.kafka.OpaAuthorizer: DEBUG
  # ...
```

NOTE

Setting a log level to **DEBUG** may result in a large amount of log output and may have performance implications.

External logging

```
apiVersion: kafka.strimzi.io/v1beta2
kind: Kafka
spec:
  # ...
  logging:
    type: external
    valueFrom:
```

```
      configMapKeyRef:
        name: customConfigMap
        key: kafka-log4j.properties
# ...
```

Any available loggers that are not configured have their level set to **OFF**.

If Kafka was deployed using the Cluster Operator, changes to Kafka logging levels are applied dynamically.

If you use external logging, a rolling update is triggered when logging appenders are changed.

Garbage collector (GC)

Garbage collector logging can also be enabled (or disabled) using the **jvmOptions** property.

5.5. KAFKACLUSTERSPEC SCHEMA PROPERTIES

Property	Description
version	The kafka broker version. Defaults to 3.6.0. Consult the user documentation to understand the process required to upgrade or downgrade the version.
string	
replicas	The number of pods in the cluster.
integer	
image	The docker image for the pods. The default value depends on the configured **Kafka.spec.kafka.version**.
string	
listeners	Configures listeners of Kafka brokers.
GenericKafkaListener array	

Property	Description
config map	Kafka broker config properties with the following prefixes cannot be set: listeners, advertised., broker., listener., host.name, port, inter.broker.listener.name, sasl., ssl., security., password., log.dir, zookeeper.connect, zookeeper.set.acl, zookeeper.ssl, zookeeper.clientCnxnSocket, authorizer., super.user, cruise.control.metrics.topic, cruise.control.metrics.reporter.bootstrap.servers,node.id, process.roles, controller., metadata.log.dir (with the exception of: zookeeper.connection.timeout.ms, sasl.server.max.receive.size,ssl.cipher.suites, ssl.protocol, ssl.enabled.protocols, ssl.secure.random.implementation,cruise.control.metrics.topic.num.partitions, cruise.control.metrics.topic.replication.factor, cruise.control.metrics.topic.retention.ms,cruise.control.metrics.topic.auto.create.retries, cruise.control.metrics.topic.auto.create.timeout.ms,cruise.control.metrics.topic.min.insync.replicas,controller.quorum.election.backoff.max.ms, controller.quorum.election.timeout.ms, controller.quorum.fetch.timeout.ms).
storage **EphemeralStorage**, **PersistentClaimStorage**, **JbodStorage**	Storage configuration (disk). Cannot be updated. The type depends on the value of the **storage.type** property within the given object, which must be one of [ephemeral, persistent-claim, jbod].
authorization **KafkaAuthorizationSimple**, **KafkaAuthorizationOpa**, **KafkaAuthorizationKeycloak**, **KafkaAuthorizationCustom**	Authorization configuration for Kafka brokers. The type depends on the value of the **authorization.type** property within the given object, which must be one of [simple, opa, keycloak, custom].
rack **Rack**	Configuration of the **broker.rack** broker config.
brokerRackInitImage string	The image of the init container used for initializing the **broker.rack**.
livenessProbe **Probe**	Pod liveness checking.
readinessProbe	Pod readiness checking.

Property	Description
Probe	
jvmOptions	JVM Options for pods.
JvmOptions	
jmxOptions	JMX Options for Kafka brokers.
KafkaJmxOptions	
resources	CPU and memory resources to reserve. For more information, see the external documentation for core/v1 resourcerequirements.
ResourceRequirements	
metricsConfig	Metrics configuration. The type depends on the value of the **metricsConfig.type** property within the given object, which must be one of [jmxPrometheusExporter].
JmxPrometheusExporterMetrics	
logging	Logging configuration for Kafka. The type depends on the value of the **logging.type** property within the given object, which must be one of [inline, external].
InlineLogging, **ExternalLogging**	
template	Template for Kafka cluster resources. The template allows users to specify how the OpenShift resources are generated.
KafkaClusterTemplate	

CHAPTER 6. GENERICKAFKALISTENER SCHEMA REFERENCE

Used in: **KafkaClusterSpec**

Full list of **GenericKafkaListener** schema properties

Configures listeners to connect to Kafka brokers within and outside OpenShift.

You configure the listeners in the **Kafka** resource.

Example Kafka resource showing listener configuration

```
apiVersion: kafka.strimzi.io/v1beta2
kind: Kafka
metadata:
  name: my-cluster
spec:
  kafka:
    #...
    listeners:
      - name: plain
        port: 9092
        type: internal
        tls: false
      - name: tls
        port: 9093
        type: internal
        tls: true
        authentication:
          type: tls
      - name: external1
        port: 9094
        type: route
        tls: true
      - name: external2
        port: 9095
        type: ingress
        tls: true
        authentication:
          type: tls
        configuration:
          bootstrap:
            host: bootstrap.myingress.com
          brokers:
          - broker: 0
            host: broker-0.myingress.com
          - broker: 1
            host: broker-1.myingress.com
          - broker: 2
            host: broker-2.myingress.com
    #...
```

6.1. LISTENERS

You configure Kafka broker listeners using the **listeners** property in the **Kafka** resource. Listeners are defined as an array.

Example listener configuration

```
listeners:
  - name: plain
    port: 9092
    type: internal
    tls: false
```

The name and port must be unique within the Kafka cluster. By specifying a unique name and port for each listener, you can configure multiple listeners. The name can be up to 25 characters long, comprising lower-case letters and numbers.

6.2. PORT

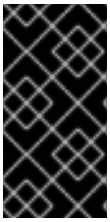
The port number is the port used in the Kafka cluster, which might not be the same port used for access by a client.

- **loadbalancer** listeners use the specified port number, as do **internal** and **cluster-ip** listeners
- **ingress** and **route** listeners use port 443 for access
- **nodeport** listeners use the port number assigned by OpenShift

For client connection, use the address and port for the bootstrap service of the listener. You can retrieve this from the status of the **Kafka** resource.

Example command to retrieve the address and port for client connection

```
oc get kafka <kafka_cluster_name> -o=jsonpath='{.status.listeners[?
(@.name=="<listener_name>")].bootstrapServers}{"\n"}'
```



IMPORTANT

When configuring listeners for client access to brokers, you can use port 9092 or higher (9093, 9094, and so on), but with a few exceptions. The listeners cannot be configured to use the ports reserved for interbroker communication (9090 and 9091), Prometheus metrics (9404), and JMX (Java Management Extensions) monitoring (9999).

6.3. TYPE

The type is set as **internal**, or for external listeners, as **route**, **loadbalancer**, **nodeport**, **ingress** or **cluster-ip**. You can also configure a **cluster-ip** listener, a type of internal listener you can use to build custom access mechanisms.

internal

You can configure internal listeners with or without encryption using the **tls** property.

Example internal listener configuration

```
#...
```

```
spec:
  kafka:
    #...
    listeners:
      #...
      - name: plain
        port: 9092
        type: internal
        tls: false
      - name: tls
        port: 9093
        type: internal
        tls: true
        authentication:
          type: tls
    #...
```

route

Configures an external listener to expose Kafka using OpenShift **Routes** and the HAProxy router.
A dedicated **Route** is created for every Kafka broker pod. An additional **Route** is created to serve as a Kafka bootstrap address. Kafka clients can use these **Routes** to connect to Kafka on port 443. The client connects on port 443, the default router port, but traffic is then routed to the port you configure, which is **9094** in this example.

Example route listener configuration

```
#...
spec:
  kafka:
    #...
    listeners:
      #...
      - name: external1
        port: 9094
        type: route
        tls: true
    #...
```

ingress

Configures an external listener to expose Kafka using Kubernetes **Ingress** and the Ingress NGINX Controller for Kubernetes.
A dedicated **Ingress** resource is created for every Kafka broker pod. An additional **Ingress** resource is created to serve as a Kafka bootstrap address. Kafka clients can use these **Ingress** resources to connect to Kafka on port 443. The client connects on port 443, the default controller port, but traffic is then routed to the port you configure, which is **9095** in the following example.

You must specify the hostnames used by the bootstrap and per-broker services using **GenericKafkaListenerConfigurationBootstrap** and **GenericKafkaListenerConfigurationBroker** properties.

Example ingress listener configuration

```
#...
```

```
spec:
  kafka:
    #...
    listeners:
      #...
      - name: external2
        port: 9095
        type: ingress
        tls: true
        authentication:
          type: tls
        configuration:
          bootstrap:
            host: bootstrap.myingress.com
          brokers:
          - broker: 0
            host: broker-0.myingress.com
          - broker: 1
            host: broker-1.myingress.com
          - broker: 2
            host: broker-2.myingress.com
  #...
```

NOTE

External listeners using **Ingress** are currently only tested with the Ingress NGINX Controller for Kubernetes.

loadbalancer

Configures an external listener to expose Kafka using a **Loadbalancer** type **Service**.
A new loadbalancer service is created for every Kafka broker pod. An additional loadbalancer is created to serve as a Kafka *bootstrap* address. Loadbalancers listen to the specified port number, which is port **9094** in the following example.

You can use the **loadBalancerSourceRanges** property to configure source ranges to restrict access to the specified IP addresses.

Example loadbalancer listener configuration

```
#...
spec:
  kafka:
    #...
    listeners:
      - name: external3
        port: 9094
        type: loadbalancer
        tls: true
        configuration:
          loadBalancerSourceRanges:
            - 10.0.0.0/8
            - 88.208.76.87/32
    #...
```

nodeport

Configures an external listener to expose Kafka using a **NodePort** type **Service**.
Kafka clients connect directly to the nodes of OpenShift. An additional **NodePort** type of service is created to serve as a Kafka bootstrap address.

When configuring the advertised addresses for the Kafka broker pods, AMQ Streams uses the address of the node on which the given pod is running. You can use **preferredNodePortAddressType** property to configure the first address type checked as the node address.

Example nodeport listener configuration

```
#...
spec:
  kafka:
    #...
    listeners:
      #...
      - name: external4
        port: 9095
        type: nodeport
        tls: false
        configuration:
          preferredNodePortAddressType: InternalDNS
    #...
```

NOTE

TLS hostname verification is not currently supported when exposing Kafka clusters using node ports.

cluster-ip

Configures an internal listener to expose Kafka using a per-broker **ClusterIP** type **Service**.
The listener does not use a headless service and its DNS names to route traffic to Kafka brokers. You can use this type of listener to expose a Kafka cluster when using the headless service is unsuitable. You might use it with a custom access mechanism, such as one that uses a specific Ingress controller or the OpenShift Gateway API.

A new **ClusterIP** service is created for each Kafka broker pod. The service is assigned a **ClusterIP** address to serve as a Kafka *bootstrap* address with a per-broker port number. For example, you can configure the listener to expose a Kafka cluster over an Nginx Ingress Controller with TCP port configuration.

Example cluster-ip listener configuration

```
#...
spec:
  kafka:
    #...
    listeners:
      - name: clusterip
        type: cluster-ip
```

```
    tls: false
    port: 9096
  #...
```

6.4. TLS

The TLS property is required.

By default, TLS encryption is not enabled. To enable it, set the **tls** property to **true**.

For **route** and **ingress** type listeners, TLS encryption must be enabled.

6.5. AUTHENTICATION

Authentication for the listener can be specified as:

- mTLS (**tls**)
- SCRAM-SHA-512 (**scram-sha-512**)
- Token-based OAuth 2.0 (**oauth**)
- Custom (**custom**)

6.6. NETWORKPOLICYPEERS

Use **networkPolicyPeers** to configure network policies that restrict access to a listener at the network level. The following example shows a **networkPolicyPeers** configuration for a **plain** and a **tls** listener.

In the following example:

- Only application pods matching the labels **app: kafka-sasl-consumer** and **app: kafka-sasl-producer** can connect to the **plain** listener. The application pods must be running in the same namespace as the Kafka broker.
- Only application pods running in namespaces matching the labels **project: myproject** and **project: myproject2** can connect to the **tls** listener.

The syntax of the **networkPolicyPeers** property is the same as the **from** property in **NetworkPolicy** resources.

Example network policy configuration

```
listeners:
  #...
  - name: plain
    port: 9092
    type: internal
    tls: true
    authentication:
      type: scram-sha-512
    networkPolicyPeers:
      - podSelector:
```

```
        matchLabels:
          app: kafka-sasl-consumer
      - podSelector:
        matchLabels:
          app: kafka-sasl-producer
  - name: tls
    port: 9093
    type: internal
    tls: true
    authentication:
      type: tls
    networkPolicyPeers:
      - namespaceSelector:
        matchLabels:
          project: myproject
      - namespaceSelector:
        matchLabels:
          project: myproject2
# ...
```

6.7. GENERICKAFKALISTENER SCHEMA PROPERTIES

Property	Description
name string	Name of the listener. The name will be used to identify the listener and the related OpenShift objects. The name has to be unique within given a Kafka cluster. The name can consist of lowercase characters and numbers and be up to 11 characters long.
port integer	Port number used by the listener inside Kafka. The port number has to be unique within a given Kafka cluster. Allowed port numbers are 9092 and higher with the exception of ports 9404 and 9999, which are already used for Prometheus and JMX. Depending on the listener type, the port number might not be the same as the port number that connects Kafka clients.

Property	Description
type string (one of [ingress, internal, route, loadbalancer, cluster-ip, nodeport])	Type of the listener. Currently the supported types are **internal**, **route**, **loadbalancer**, **nodeport** and **ingress**. • **internal** type exposes Kafka internally only within the OpenShift cluster. • **route** type uses OpenShift Routes to expose Kafka. • **loadbalancer** type uses LoadBalancer type services to expose Kafka. • **nodeport** type uses NodePort type services to expose Kafka. • **ingress** type uses OpenShift Nginx Ingress to expose Kafka with TLS passthrough. • **cluster-ip** type uses a per-broker **ClusterIP** service.
tls boolean	Enables TLS encryption on the listener. This is a required property.
authentication **KafkaListenerAuthenticationTls**, **KafkaListenerAuthenticationScramSha512**, **KafkaListenerAuthenticationOAuth**, **KafkaListenerAuthenticationCustom**	Authentication configuration for this listener. The type depends on the value of the **authentication.type** property within the given object, which must be one of [tls, scram-sha-512, oauth, custom].
configuration **GenericKafkaListenerConfiguration**	Additional listener configuration.
networkPolicyPeers NetworkPolicyPeer array	List of peers which should be able to connect to this listener. Peers in this list are combined using a logical OR operation. If this field is empty or missing, all connections will be allowed for this listener. If this field is present and contains at least one item, the listener only allows the traffic which matches at least one item in this list. For more information, see the external documentation for networking.k8s.io/v1 networkpolicypeer.

CHAPTER 7. KAFKALISTENERAUTHENTICATIONTLS SCHEMA REFERENCE

Used in: **GenericKafkaListener**

The **type** property is a discriminator that distinguishes use of the **KafkaListenerAuthenticationTls** type from **KafkaListenerAuthenticationScramSha512**, **KafkaListenerAuthenticationOAuth**, **KafkaListenerAuthenticationCustom**. It must have the value **tls** for the type **KafkaListenerAuthenticationTls**.

Property	Description
type	Must be **tls**.
string	

CHAPTER 8. KAFKALISTENERAUTHENTICATIONSCRAMSHA512 SCHEMA REFERENCE

Used in: **GenericKafkaListener**

The **type** property is a discriminator that distinguishes use of the **KafkaListenerAuthenticationScramSha512** type from **KafkaListenerAuthenticationTls**, **KafkaListenerAuthenticationOAuth**, **KafkaListenerAuthenticationCustom**. It must have the value **scram-sha-512** for the type **KafkaListenerAuthenticationScramSha512**.

Property	Description
type	Must be **scram-sha-512**.
string	

CHAPTER 9. KAFKALISTENERAUTHENTICATIONOAUTH SCHEMA REFERENCE

Used in: **GenericKafkaListener**

The **type** property is a discriminator that distinguishes use of the **KafkaListenerAuthenticationOAuth** type from **KafkaListenerAuthenticationTls**, **KafkaListenerAuthenticationScramSha512**, **KafkaListenerAuthenticationCustom**. It must have the value **oauth** for the type **KafkaListenerAuthenticationOAuth**.

Property	Description
accessTokenIsJwt boolean	Configure whether the access token is treated as JWT. This must be set to **false** if the authorization server returns opaque tokens. Defaults to **true**.
checkAccessTokenType boolean	Configure whether the access token type check is performed or not. This should be set to **false** if the authorization server does not include 'typ' claim in JWT token. Defaults to **true**.
checkAudience boolean	Enable or disable audience checking. Audience checks identify the recipients of tokens. If audience checking is enabled, the OAuth Client ID also has to be configured using the **clientId** property. The Kafka broker will reject tokens that do not have its **clientId** in their **aud** (audience) claim.Default value is **false**.
checkIssuer boolean	Enable or disable issuer checking. By default issuer is checked using the value configured by **validIssuerUri**. Default value is **true**.
clientAudience string	The audience to use when making requests to the authorization server's token endpoint. Used for inter-broker authentication and for configuring OAuth 2.0 over PLAIN using the **clientId** and **secret** method.
clientId string	OAuth Client ID which the Kafka broker can use to authenticate against the authorization server and use the introspect endpoint URI.
clientScope string	The scope to use when making requests to the authorization server's token endpoint. Used for inter-broker authentication and for configuring OAuth 2.0 over PLAIN using the **clientId** and **secret** method.
clientSecret **GenericSecretSource**	Link to OpenShift Secret containing the OAuth client secret which the Kafka broker can use to authenticate against the authorization server and use the introspect endpoint URI.

Property	Description
connectTimeoutSeconds integer	The connect timeout in seconds when connecting to authorization server. If not set, the effective connect timeout is 60 seconds.
customClaimCheck string	JsonPath filter query to be applied to the JWT token or to the response of the introspection endpoint for additional token validation. Not set by default.
disableTlsHostnameVerification boolean	Enable or disable TLS hostname verification. Default value is **false**.
enableECDSA boolean	**The enableECDSA property has been deprecated.** Enable or disable ECDSA support by installing BouncyCastle crypto provider. ECDSA support is always enabled. The BouncyCastle libraries are no longer packaged with AMQ Streams. Value is ignored.
enableMetrics boolean	Enable or disable OAuth metrics. Default value is **false**.
enableOauthBearer boolean	Enable or disable OAuth authentication over SASL_OAUTHBEARER. Default value is **true**.
enablePlain boolean	Enable or disable OAuth authentication over SASL_PLAIN. There is no re-authentication support when this mechanism is used. Default value is **false**.
failFast boolean	Enable or disable termination of Kafka broker processes due to potentially recoverable runtime errors during startup. Default value is **true**.
fallbackUserNameClaim string	The fallback username claim to be used for the user id if the claim specified by **userNameClaim** is not present. This is useful when **client_credentials** authentication only results in the client id being provided in another claim. It only takes effect if **userNameClaim** is set.

Property	Description
fallbackUserNamePrefix string	The prefix to use with the value of **fallbackUserNameClaim** to construct the user id. This only takes effect if **fallbackUserNameClaim** is true, and the value is present for the claim. Mapping usernames and client ids into the same user id space is useful in preventing name collisions.
groupsClaim string	JsonPath query used to extract groups for the user during authentication. Extracted groups can be used by a custom authorizer. By default no groups are extracted.
groupsClaimDelimiter string	A delimiter used to parse groups when they are extracted as a single String value rather than a JSON array. Default value is ',' (comma).
httpRetries integer	The maximum number of retries to attempt if an initial HTTP request fails. If not set, the default is to not attempt any retries.
httpRetryPauseMs integer	The pause to take before retrying a failed HTTP request. If not set, the default is to not pause at all but to immediately repeat a request.
includeAcceptHeader boolean	Whether the Accept header should be set in requests to the authorization servers. The default value is **true**.
introspectionEndpointUri string	URI of the token introspection endpoint which can be used to validate opaque non-JWT tokens.
jwksEndpointUri string	URI of the JWKS certificate endpoint, which can be used for local JWT validation.
jwksExpirySeconds integer	Configures how often are the JWKS certificates considered valid. The expiry interval has to be at least 60 seconds longer then the refresh interval specified in **jwksRefreshSeconds**. Defaults to 360 seconds.
jwksIgnoreKeyUse boolean	Flag to ignore the 'use' attribute of **key** declarations in a JWKS endpoint response. Default value is **false**.

Property	Description
jwksMinRefreshPauseSeconds integer	The minimum pause between two consecutive refreshes. When an unknown signing key is encountered the refresh is scheduled immediately, but will always wait for this minimum pause. Defaults to 1 second.
jwksRefreshSeconds integer	Configures how often are the JWKS certificates refreshed. The refresh interval has to be at least 60 seconds shorter then the expiry interval specified in **jwksExpirySeconds**. Defaults to 300 seconds.
maxSecondsWithoutReauthentication integer	Maximum number of seconds the authenticated session remains valid without re-authentication. This enables Apache Kafka re-authentication feature, and causes sessions to expire when the access token expires. If the access token expires before max time or if max time is reached, the client has to re-authenticate, otherwise the server will drop the connection. Not set by default - the authenticated session does not expire when the access token expires. This option only applies to SASL_OAUTHBEARER authentication mechanism (when **enableOauthBearer** is **true**).
readTimeoutSeconds integer	The read timeout in seconds when connecting to authorization server. If not set, the effective read timeout is 60 seconds.
tlsTrustedCertificates **CertSecretSource** array	Trusted certificates for TLS connection to the OAuth server.
tokenEndpointUri string	URI of the Token Endpoint to use with SASL_PLAIN mechanism when the client authenticates with **clientId** and a **secret**. If set, the client can authenticate over SASL_PLAIN by either setting **username** to **clientId**, and setting **password** to client **secret**, or by setting **username** to account username, and **password** to access token prefixed with **$accessToken:**. If this option is not set, the **password** is always interpreted as an access token (without a prefix), and **username** as the account username (a so called 'no-client-credentials' mode).
type string	Must be **oauth**.

Property	Description
userInfoEndpointUri string	URI of the User Info Endpoint to use as a fallback to obtaining the user id when the Introspection Endpoint does not return information that can be used for the user id.
userNameClaim string	Name of the claim from the JWT authentication token, Introspection Endpoint response or User Info Endpoint response which will be used to extract the user id. Defaults to **sub**.
validIssuerUri string	URI of the token issuer used for authentication.
validTokenType string	Valid value for the **token_type** attribute returned by the Introspection Endpoint. No default value, and not checked by default.

CHAPTER 10. GENERICSECRETSOURCE SCHEMA REFERENCE

Used in: **KafkaClientAuthenticationOAuth**, **KafkaListenerAuthenticationCustom**, **KafkaListenerAuthenticationOAuth**

Property	Description
key	The key under which the secret value is stored in the OpenShift Secret.
string	
secretName	The name of the OpenShift Secret containing the secret value.
string	

CHAPTER 11. CERTSECRETSOURCE SCHEMA REFERENCE

Used in: **ClientTls**, **KafkaAuthorizationKeycloak**, **KafkaAuthorizationOpa**, **KafkaClientAuthenticationOAuth**, **KafkaListenerAuthenticationOAuth**

Property	Description
certificate	The name of the file certificate in the Secret.
string	
secretName	The name of the Secret containing the certificate.
string	

CHAPTER 12. KAFKALISTENERAUTHENTICATIONCUSTOM SCHEMA REFERENCE

Used in: **GenericKafkaListener**

Full list of **KafkaListenerAuthenticationCustom** schema properties

To configure custom authentication, set the **type** property to **custom**.

Custom authentication allows for any type of Kafka-supported authentication to be used.

Example custom OAuth authentication configuration

```
spec:
  kafka:
    config:
      principal.builder.class: SimplePrincipal.class
    listeners:
      - name: oauth-bespoke
        port: 9093
        type: internal
        tls: true
        authentication:
          type: custom
          sasl: true
          listenerConfig:
            oauthbearer.sasl.client.callback.handler.class: client.class
            oauthbearer.sasl.server.callback.handler.class: server.class
            oauthbearer.sasl.login.callback.handler.class: login.class
            oauthbearer.connections.max.reauth.ms: 999999999
            sasl.enabled.mechanisms: oauthbearer
            oauthbearer.sasl.jaas.config: |
              org.apache.kafka.common.security.oauthbearer.OAuthBearerLoginModule required ;
          secrets:
            - name: example
```

A protocol map is generated that uses the **sasl** and **tls** values to determine which protocol to map to the listener.

- SASL = True, TLS = True → SASL_SSL
- SASL = False, TLS = True → SSL
- SASL = True, TLS = False → SASL_PLAINTEXT
- SASL = False, TLS = False → PLAINTEXT

12.1. LISTENERCONFIG

Listener configuration specified using **listenerConfig** is prefixed with **listener.name.*<listener_name>-<port>***. For example, **sasl.enabled.mechanisms** becomes **listener.name.*<listener_name>-<port>*.sasl.enabled.mechanisms**.

12.2. SECRETS

Secrets are mounted to **/opt/kafka/custom-authn-secrets/custom-listener-*<listener_name>*-*<port>*/*<secret_name>*** in the Kafka broker nodes' containers.

For example, the mounted secret (**example**) in the example configuration would be located at **/opt/kafka/custom-authn-secrets/custom-listener-oauth-bespoke-9093/example**.

12.3. PRINCIPAL BUILDER

You can set a custom principal builder in the Kafka cluster configuration. However, the principal builder is subject to the following requirements:

- The specified principal builder class must exist on the image. *Before* building your own, check if one already exists. You'll need to rebuild the AMQ Streams images with the required classes.
- No other listener is using **oauth** type authentication. This is because an OAuth listener appends its own principle builder to the Kafka configuration.
- The specified principal builder is compatible with AMQ Streams.

Custom principal builders must support peer certificates for authentication, as AMQ Streams uses these to manage the Kafka cluster.

NOTE

Kafka's default principal builder class supports the building of principals based on the names of peer certificates. The custom principal builder should provide a principal of type **user** using the name of the SSL peer certificate.

The following example shows a custom principal builder that satisfies the OAuth requirements of AMQ Streams.

Example principal builder for custom OAuth configuration

```
public final class CustomKafkaPrincipalBuilder implements KafkaPrincipalBuilder {

    public KafkaPrincipalBuilder() {}

    @Override
    public KafkaPrincipal build(AuthenticationContext context) {
        if (context instanceof SslAuthenticationContext) {
            SSLSession sslSession = ((SslAuthenticationContext) context).session();
            try {
                return new KafkaPrincipal(
                    KafkaPrincipal.USER_TYPE, sslSession.getPeerPrincipal().getName());
            } catch (SSLPeerUnverifiedException e) {
                throw new IllegalArgumentException("Cannot use an unverified peer for authentication", e);
            }
        }

        // Create your own KafkaPrincipal here
        ...
    }
}
```

12.4. KAFKALISTENERAUTHENTICATIONCUSTOM SCHEMA PROPERTIES

The **type** property is a discriminator that distinguishes use of the **KafkaListenerAuthenticationCustom** type from **KafkaListenerAuthenticationTls**, **KafkaListenerAuthenticationScramSha512**, **KafkaListenerAuthenticationOAuth**. It must have the value **custom** for the type **KafkaListenerAuthenticationCustom**.

Property	Description
listenerConfig map	Configuration to be used for a specific listener. All values are prefixed with listener.name.*<listener_name>*.
sasl boolean	Enable or disable SASL on this listener.
secrets **GenericSecretSource** array	Secrets to be mounted to /opt/kafka/custom-authn-secrets/custom-listener-*<listener_name>*-*<port>*/*<secret_name>*.
type string	Must be **custom**.

CHAPTER 13. GENERICKAFKALISTENERCONFIGURATION SCHEMA REFERENCE

Used in: **GenericKafkaListener**

Full list of **GenericKafkaListenerConfiguration** schema properties

Configuration for Kafka listeners.

13.1. BROKERCERTCHAINANDKEY

The **brokerCertChainAndKey** property is only used with listeners that have TLS encryption enabled. You can use the property to provide your own Kafka listener certificates.

Example configuration for a loadbalancer external listener with TLS encryption enabled

```
listeners:
  #...
  - name: external3
    port: 9094
    type: loadbalancer
    tls: true
    authentication:
      type: tls
    configuration:
      brokerCertChainAndKey:
        secretName: my-secret
        certificate: my-listener-certificate.crt
        key: my-listener-key.key
# ...
```

13.2. EXTERNALTRAFFICPOLICY

The **externalTrafficPolicy** property is used with **loadbalancer** and **nodeport** listeners. When exposing Kafka outside of OpenShift you can choose **Local** or **Cluster**. **Local** avoids hops to other nodes and preserves the client IP, whereas **Cluster** does neither. The default is **Cluster**.

13.3. LOADBALANCERSOURCERANGES

The **loadBalancerSourceRanges** property is only used with **loadbalancer** listeners. When exposing Kafka outside of OpenShift use source ranges, in addition to labels and annotations, to customize how a service is created.

Example source ranges configured for a loadbalancer listener

```
listeners:
  #...
  - name: external3
    port: 9094
    type: loadbalancer
    tls: false
    configuration:
```

```
      externalTrafficPolicy: Local
      loadBalancerSourceRanges:
        - 10.0.0.0/8
        - 88.208.76.87/32
      # ...
# ...
```

13.4. CLASS

The **class** property is only used with **ingress** listeners. You can configure the **Ingress** class using the **class** property.

Example of an external listener of type ingress using Ingress class nginx-internal

```
listeners:
  #...
  - name: external2
    port: 9094
    type: ingress
    tls: true
    configuration:
      class: nginx-internal
    # ...
# ...
```

13.5. PREFERREDNODEPORTADDRESSTYPE

The **preferredNodePortAddressType** property is only used with **nodeport** listeners.

Use the **preferredNodePortAddressType** property in your listener configuration to specify the first address type checked as the node address. This property is useful, for example, if your deployment does not have DNS support, or you only want to expose a broker internally through an internal DNS or IP address. If an address of this type is found, it is used. If the preferred address type is not found, AMQ Streams proceeds through the types in the standard order of priority:

1. ExternalDNS
2. ExternalIP
3. Hostname
4. InternalDNS
5. InternalIP

Example of an external listener configured with a preferred node port address type

```
listeners:
  #...
  - name: external4
    port: 9094
    type: nodeport
    tls: false
    configuration:
```

```
      preferredNodePortAddressType: InternalDNS
      # ...
# ...
```

13.6. USESERVICEDNSDOMAIN

The **useServiceDnsDomain** property is only used with **internal** and **cluster-ip** listeners. It defines whether the fully-qualified DNS names that include the cluster service suffix (usually **.cluster.local**) are used. With **useServiceDnsDomain** set as **false**, the advertised addresses are generated without the service suffix; for example, **my-cluster-kafka-0.my-cluster-kafka-brokers.myproject.svc**. With **useServiceDnsDomain** set as **true**, the advertised addresses are generated with the service suffix; for example, **my-cluster-kafka-0.my-cluster-kafka-brokers.myproject.svc.cluster.local**. Default is **false**.

Example of an internal listener configured to use the Service DNS domain

```
listeners:
  #...
  - name: plain
    port: 9092
    type: internal
    tls: false
    configuration:
      useServiceDnsDomain: true
      # ...
# ...
```

If your OpenShift cluster uses a different service suffix than **.cluster.local**, you can configure the suffix using the **KUBERNETES_SERVICE_DNS_DOMAIN** environment variable in the Cluster Operator configuration.

13.7. GENERICKAFKALISTENERCONFIGURATION SCHEMA PROPERTIES

Property	Description
brokerCertChainAndKey **CertAndKeySecretSource**	Reference to the **Secret** which holds the certificate and private key pair which will be used for this listener. The certificate can optionally contain the whole chain. This field can be used only with listeners with enabled TLS encryption.
externalTrafficPolicy string (one of [Local, Cluster])	Specifies whether the service routes external traffic to node-local or cluster-wide endpoints. **Cluster** may cause a second hop to another node and obscures the client source IP. **Local** avoids a second hop for LoadBalancer and Nodeport type services and preserves the client source IP (when supported by the infrastructure). If unspecified, OpenShift will use **Cluster** as the default.This field can be used only with **loadbalancer** or **nodeport** type listener.

Property	Description
loadBalancerSourceRanges string array	A list of CIDR ranges (for example **10.0.0.0/8** or **130.211.204.1/32**) from which clients can connect to load balancer type listeners. If supported by the platform, traffic through the loadbalancer is restricted to the specified CIDR ranges. This field is applicable only for loadbalancer type services and is ignored if the cloud provider does not support the feature. This field can be used only with **loadbalancer** type listener.
bootstrap **GenericKafkaListenerConfigurationBootstrap**	Bootstrap configuration.
brokers **GenericKafkaListenerConfigurationBroker** array	Per-broker configurations.
ipFamilyPolicy string (one of [RequireDualStack, SingleStack, PreferDualStack])	Specifies the IP Family Policy used by the service. Available options are **SingleStack**, **PreferDualStack** and **RequireDualStack**. **SingleStack** is for a single IP family. **PreferDualStack** is for two IP families on dual-stack configured clusters or a single IP family on single-stack clusters. **RequireDualStack** fails unless there are two IP families on dual-stack configured clusters. If unspecified, OpenShift will choose the default value based on the service type.
ipFamilies string (one or more of [IPv6, IPv4]) array	Specifies the IP Families used by the service. Available options are **IPv4** and **IPv6**. If unspecified, OpenShift will choose the default value based on the **ipFamilyPolicy** setting.
createBootstrapService boolean	Whether to create the bootstrap service or not. The bootstrap service is created by default (if not specified differently). This field can be used with the **loadBalancer** type listener.
class string	Configures a specific class for **Ingress** and **LoadBalancer** that defines which controller will be used. This field can only be used with **ingress** and **loadbalancer** type listeners. If not specified, the default controller is used. For an **ingress** listener, set the **ingressClassName** property in the **Ingress** resources. For a **loadbalancer** listener, set the **loadBalancerClass** property in the **Service** resources.

Property	Description
finalizers string array	A list of finalizers which will be configured for the **LoadBalancer** type Services created for this listener. If supported by the platform, the finalizer **service.kubernetes.io/load-balancer-cleanup** to make sure that the external load balancer is deleted together with the service.For more information, see https://kubernetes.io/docs/tasks/access-application-cluster/create-external-load-balancer/#garbage-collecting-load-balancers. This field can be used only with **loadbalancer** type listeners.
maxConnectionCreationRate integer	The maximum connection creation rate we allow in this listener at any time. New connections will be throttled if the limit is reached.
maxConnections integer	The maximum number of connections we allow for this listener in the broker at any time. New connections are blocked if the limit is reached.
preferredNodePortAddressType string (one of [ExternalDNS, ExternalIP, Hostname, InternalIP, InternalDNS])	Defines which address type should be used as the node address. Available types are: **ExternalDNS**, **ExternalIP**, **InternalDNS**, **InternalIP** and **Hostname**. By default, the addresses will be used in the following order (the first one found will be used): • **ExternalDNS** • **ExternalIP** • **InternalDNS** • **InternalIP** • **Hostname** This field is used to select the preferred address type, which is checked first. If no address is found for this address type, the other types are checked in the default order. This field can only be used with **nodeport** type listener.
useServiceDnsDomain boolean	Configures whether the OpenShift service DNS domain should be used or not. If set to **true**, the generated addresses will contain the service DNS domain suffix (by default **.cluster.local**, can be configured using environment variable **KUBERNETES_SERVICE_DNS_DOMAIN**). Defaults to **false**.This field can be used only with **internal** and **cluster-ip** type listeners.

CHAPTER 14. CERTANDKEYSECRETSOURCE SCHEMA REFERENCE

Used in: **GenericKafkaListenerConfiguration**, **KafkaClientAuthenticationTls**

Property	Description
certificate	The name of the file certificate in the Secret.
string	
key	The name of the private key in the Secret.
string	
secretName	The name of the Secret containing the certificate.
string	

CHAPTER 15. GENERICKAFKALISTENERCONFIGURATIONBOOTSTRAP SCHEMA REFERENCE

Used in: **GenericKafkaListenerConfiguration**

Full list of **GenericKafkaListenerConfigurationBootstrap** schema properties

Broker service equivalents of **nodePort**, **host**, **loadBalancerIP** and **annotations** properties are configured in the **GenericKafkaListenerConfigurationBroker** schema.

15.1. ALTERNATIVENAMES

You can specify alternative names for the bootstrap service. The names are added to the broker certificates and can be used for TLS hostname verification. The **alternativeNames** property is applicable to all types of listeners.

Example of an external route listener configured with an additional bootstrap address

```
listeners:
  #...
  - name: external1
    port: 9094
    type: route
    tls: true
    authentication:
      type: tls
    configuration:
      bootstrap:
        alternativeNames:
          - example.hostname1
          - example.hostname2
# ...
```

15.2. HOST

The **host** property is used with **route** and **ingress** listeners to specify the hostnames used by the bootstrap and per-broker services.

A **host** property value is mandatory for **ingress** listener configuration, as the Ingress controller does not assign any hostnames automatically. Make sure that the hostnames resolve to the Ingress endpoints. AMQ Streams will not perform any validation that the requested hosts are available and properly routed to the Ingress endpoints.

Example of host configuration for an ingress listener

```
listeners:
  #...
  - name: external2
    port: 9094
    type: ingress
    tls: true
    authentication:
      type: tls
```

```
    configuration:
      bootstrap:
        host: bootstrap.myingress.com
      brokers:
      - broker: 0
        host: broker-0.myingress.com
      - broker: 1
        host: broker-1.myingress.com
      - broker: 2
        host: broker-2.myingress.com
# ...
```

By default, **route** listener hosts are automatically assigned by OpenShift. However, you can override the assigned route hosts by specifying hosts.

AMQ Streams does not perform any validation that the requested hosts are available. You must ensure that they are free and can be used.

Example of host configuration for a route listener

```
# ...
listeners:
  #...
  - name: external1
    port: 9094
    type: route
    tls: true
    authentication:
      type: tls
    configuration:
      bootstrap:
        host: bootstrap.myrouter.com
      brokers:
      - broker: 0
        host: broker-0.myrouter.com
      - broker: 1
        host: broker-1.myrouter.com
      - broker: 2
        host: broker-2.myrouter.com
# ...
```

15.3. NODEPORT

By default, the port numbers used for the bootstrap and broker services are automatically assigned by OpenShift. You can override the assigned node ports for **nodeport** listeners by specifying the requested port numbers.

AMQ Streams does not perform any validation on the requested ports. You must ensure that they are free and available for use.

Example of an external listener configured with overrides for node ports

```
# ...
listeners:
```

```
  #...
  - name: external4
    port: 9094
    type: nodeport
    tls: true
    authentication:
      type: tls
    configuration:
      bootstrap:
        nodePort: 32100
      brokers:
      - broker: 0
        nodePort: 32000
      - broker: 1
        nodePort: 32001
      - broker: 2
        nodePort: 32002
# ...
```

15.4. LOADBALANCERIP

Use the **loadBalancerIP** property to request a specific IP address when creating a loadbalancer. Use this property when you need to use a loadbalancer with a specific IP address. The **loadBalancerIP** field is ignored if the cloud provider does not support the feature.

Example of an external listener of type loadbalancer with specific loadbalancer IP address requests

```
# ...
listeners:
  #...
  - name: external3
    port: 9094
    type: loadbalancer
    tls: true
    authentication:
      type: tls
    configuration:
      bootstrap:
        loadBalancerIP: 172.29.3.10
      brokers:
      - broker: 0
        loadBalancerIP: 172.29.3.1
      - broker: 1
        loadBalancerIP: 172.29.3.2
      - broker: 2
        loadBalancerIP: 172.29.3.3
# ...
```

15.5. ANNOTATIONS

Use the **annotations** property to add annotations to OpenShift resources related to the listeners. You can use these annotations, for example, to instrument DNS tooling such as External DNS, which automatically assigns DNS names to the loadbalancer services.

Example of an external listener of type `loadbalancer` using `annotations`

```
# ...
listeners:
  #...
  - name: external3
    port: 9094
    type: loadbalancer
    tls: true
    authentication:
      type: tls
    configuration:
      bootstrap:
        annotations:
          external-dns.alpha.kubernetes.io/hostname: kafka-bootstrap.mydomain.com.
          external-dns.alpha.kubernetes.io/ttl: "60"
      brokers:
      - broker: 0
        annotations:
          external-dns.alpha.kubernetes.io/hostname: kafka-broker-0.mydomain.com.
          external-dns.alpha.kubernetes.io/ttl: "60"
      - broker: 1
        annotations:
          external-dns.alpha.kubernetes.io/hostname: kafka-broker-1.mydomain.com.
          external-dns.alpha.kubernetes.io/ttl: "60"
      - broker: 2
        annotations:
          external-dns.alpha.kubernetes.io/hostname: kafka-broker-2.mydomain.com.
          external-dns.alpha.kubernetes.io/ttl: "60"
# ...
```

15.6. GENERICKAFKALISTENERCONFIGURATIONBOOTSTRAP SCHEMA PROPERTIES

Property	Description
alternativeNames string array	Additional alternative names for the bootstrap service. The alternative names will be added to the list of subject alternative names of the TLS certificates.
host string	The bootstrap host. This field will be used in the Ingress resource or in the Route resource to specify the desired hostname. This field can be used only with **route** (optional) or **ingress** (required) type listeners.
nodePort integer	Node port for the bootstrap service. This field can be used only with **nodeport** type listener.

Property	Description
loadBalancerIP string	The loadbalancer is requested with the IP address specified in this field. This feature depends on whether the underlying cloud provider supports specifying the **loadBalancerIP** when a load balancer is created. This field is ignored if the cloud provider does not support the feature.This field can be used only with **loadbalancer** type listener.
annotations map	Annotations that will be added to the **Ingress**, **Route**, or **Service** resource. You can use this field to configure DNS providers such as External DNS. This field can be used only with **loadbalancer**, **nodeport**, **route**, or **ingress** type listeners.
labels map	Labels that will be added to the **Ingress**, **Route**, or **Service** resource. This field can be used only with **loadbalancer**, **nodeport**, **route**, or **ingress** type listeners.

CHAPTER 16. GENERICKAFKALISTENERCONFIGURATIONBROKER SCHEMA REFERENCE

Used in: **GenericKafkaListenerConfiguration**

Full list of **GenericKafkaListenerConfigurationBroker** schema properties

You can see example configuration for the **nodePort**, **host**, **loadBalancerIP** and **annotations** properties in the **GenericKafkaListenerConfigurationBootstrap** schema, which configures bootstrap service overrides.

Advertised addresses for brokers

By default, AMQ Streams tries to automatically determine the hostnames and ports that your Kafka cluster advertises to its clients. This is not sufficient in all situations, because the infrastructure on which AMQ Streams is running might not provide the right hostname or port through which Kafka can be accessed.

You can specify a broker ID and customize the advertised hostname and port in the **configuration** property of the listener. AMQ Streams will then automatically configure the advertised address in the Kafka brokers and add it to the broker certificates so it can be used for TLS hostname verification. Overriding the advertised host and ports is available for all types of listeners.

Example of an external **route** listener configured with overrides for advertised addresses

```
listeners:
  #...
  - name: external1
    port: 9094
    type: route
    tls: true
    authentication:
      type: tls
    configuration:
      brokers:
      - broker: 0
        advertisedHost: example.hostname.0
        advertisedPort: 12340
      - broker: 1
        advertisedHost: example.hostname.1
        advertisedPort: 12341
      - broker: 2
        advertisedHost: example.hostname.2
        advertisedPort: 12342
# ...
```

16.1. GENERICKAFKALISTENERCONFIGURATIONBROKER SCHEMA PROPERTIES

Property	Description
broker	ID of the kafka broker (broker identifier). Broker IDs start from 0 and correspond to the number of broker replicas.

Property	Description
integer	
advertisedHost string	The host name used in the brokers' **advertised.listeners**.
advertisedPort integer	The port number used in the brokers' **advertised.listeners**.
host string	The broker host. This field will be used in the Ingress resource or in the Route resource to specify the desired hostname. This field can be used only with **route** (optional) or **ingress** (required) type listeners.
nodePort integer	Node port for the per-broker service. This field can be used only with **nodeport** type listener.
loadBalancerIP string	The loadbalancer is requested with the IP address specified in this field. This feature depends on whether the underlying cloud provider supports specifying the **loadBalancerIP** when a load balancer is created. This field is ignored if the cloud provider does not support the feature.This field can be used only with **loadbalancer** type listener.
annotations map	Annotations that will be added to the **Ingress** or **Service** resource. You can use this field to configure DNS providers such as External DNS. This field can be used only with **loadbalancer**, **nodeport**, or **ingress** type listeners.
labels map	Labels that will be added to the **Ingress**, **Route**, or **Service** resource. This field can be used only with **loadbalancer**, **nodeport**, **route**, or **ingress** type listeners.

CHAPTER 17. EPHEMERALSTORAGE SCHEMA REFERENCE

Used in: **JbodStorage**, **KafkaClusterSpec**, **KafkaNodePoolSpec**, **ZookeeperClusterSpec**

The **type** property is a discriminator that distinguishes use of the **EphemeralStorage** type from **PersistentClaimStorage**. It must have the value **ephemeral** for the type **EphemeralStorage**.

Property	Description
id	Storage identification number. It is mandatory only for storage volumes defined in a storage of type 'jbod'.
integer	
sizeLimit	When type=ephemeral, defines the total amount of local storage required for this EmptyDir volume (for example 1Gi).
string	
type	Must be **ephemeral**.
string	

CHAPTER 18. PERSISTENTCLAIMSTORAGE SCHEMA REFERENCE

Used in: **JbodStorage**, **KafkaClusterSpec**, **KafkaNodePoolSpec**, **ZookeeperClusterSpec**

The **type** property is a discriminator that distinguishes use of the **PersistentClaimStorage** type from **EphemeralStorage**. It must have the value **persistent-claim** for the type **PersistentClaimStorage**.

Property	Description
type string	Must be **persistent-claim**.
size string	When type=persistent-claim, defines the size of the persistent volume claim (i.e 1Gi). Mandatory when type=persistent-claim.
selector map	Specifies a specific persistent volume to use. It contains key:value pairs representing labels for selecting such a volume.
deleteClaim boolean	Specifies if the persistent volume claim has to be deleted when the cluster is un-deployed.
class string	The storage class to use for dynamic volume allocation.
id integer	Storage identification number. It is mandatory only for storage volumes defined in a storage of type 'jbod'.
overrides **PersistentClaimStorageOverride** array	Overrides for individual brokers. The **overrides** field allows to specify a different configuration for different brokers.

CHAPTER 19. PERSISTENTCLAIMSTORAGEOVERRIDE SCHEMA REFERENCE

Used in: **PersistentClaimStorage**

Property	Description
class string	The storage class to use for dynamic volume allocation for this broker.
broker integer	Id of the kafka broker (broker identifier).

CHAPTER 20. JBODSTORAGE SCHEMA REFERENCE

Used in: **KafkaClusterSpec**, **KafkaNodePoolSpec**

The **type** property is a discriminator that distinguishes use of the **JbodStorage** type from **EphemeralStorage**, **PersistentClaimStorage**. It must have the value **jbod** for the type **JbodStorage**.

Property	Description
type string	Must be **jbod**.
volumes **EphemeralStorage**, **PersistentClaimStorage** array	List of volumes as Storage objects representing the JBOD disks array.

CHAPTER 21. KAFKAAUTHORIZATIONSIMPLE SCHEMA REFERENCE

Used in: **KafkaClusterSpec**

Full list of **KafkaAuthorizationSimple** schema properties

For simple authorization, AMQ Streams uses Kafka's built-in authorization plugins: the **StandardAuthorizer** for KRaft mode and the **AclAuthorizer** for ZooKeeper-based cluster management. ACLs allow you to define which users have access to which resources at a granular level.

Configure the **Kafka** custom resource to use simple authorization. Set the **type** property in the **authorization** section to the value **simple**, and configure a list of super users.

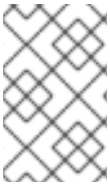
Access rules are configured for the **KafkaUser**, as described in the ACLRule schema reference.

21.1. SUPERUSERS

A list of user principals treated as super users, so that they are always allowed without querying ACL rules.

An example of simple authorization configuration

```
apiVersion: kafka.strimzi.io/v1beta2
kind: Kafka
metadata:
  name: my-cluster
  namespace: myproject
spec:
  kafka:
    # ...
    authorization:
      type: simple
      superUsers:
        - CN=client_1
        - user_2
        - CN=client_3
    # ...
```



NOTE

The **super.user** configuration option in the **config** property in **Kafka.spec.kafka** is ignored. Designate super users in the **authorization** property instead. For more information, see Kafka broker configuration.

21.2. KAFKAAUTHORIZATIONSIMPLE SCHEMA PROPERTIES

The **type** property is a discriminator that distinguishes use of the **KafkaAuthorizationSimple** type from **KafkaAuthorizationOpa**, **KafkaAuthorizationKeycloak**, **KafkaAuthorizationCustom**. It must have the value **simple** for the type **KafkaAuthorizationSimple**.

Property	Description
type	Must be **simple**.
string	
superUsers	List of super users. Should contain list of user principals which should get unlimited access rights.
string array	

CHAPTER 22. KAFKAAUTHORIZATIONOPA SCHEMA REFERENCE

Used in: **KafkaClusterSpec**

Full list of **KafkaAuthorizationOpa** schema properties

To use Open Policy Agent authorization, set the **type** property in the **authorization** section to the value **opa**, and configure OPA properties as required. AMQ Streams uses Open Policy Agent plugin for Kafka authorization as the authorizer. For more information about the format of the input data and policy examples, see Open Policy Agent plugin for Kafka authorization .

22.1. URL

The URL used to connect to the Open Policy Agent server. The URL has to include the policy which will be queried by the authorizer. **Required.**

22.2. ALLOWONERROR

Defines whether a Kafka client should be allowed or denied by default when the authorizer fails to query the Open Policy Agent, for example, when it is temporarily unavailable. Defaults to **false** - all actions will be denied.

22.3. INITIALCACHECAPACITY

Initial capacity of the local cache used by the authorizer to avoid querying the Open Policy Agent for every request. Defaults to **5000**.

22.4. MAXIMUMCACHESIZE

Maximum capacity of the local cache used by the authorizer to avoid querying the Open Policy Agent for every request. Defaults to **50000**.

22.5. EXPIREAFTERMS

The expiration of the records kept in the local cache to avoid querying the Open Policy Agent for every request. Defines how often the cached authorization decisions are reloaded from the Open Policy Agent server. In milliseconds. Defaults to **3600000** milliseconds (1 hour).

22.6. TLSTRUSTEDCERTIFICATES

Trusted certificates for TLS connection to the OPA server.

22.7. SUPERUSERS

A list of user principals treated as super users, so that they are always allowed without querying the open Policy Agent policy.

An example of Open Policy Agent authorizer configuration

```
apiVersion: kafka.strimzi.io/v1beta2
```

```
kind: Kafka
metadata:
  name: my-cluster
  namespace: myproject
spec:
  kafka:
    # ...
    authorization:
      type: opa
      url: http://opa:8181/v1/data/kafka/allow
      allowOnError: false
      initialCacheCapacity: 1000
      maximumCacheSize: 10000
      expireAfterMs: 60000
      superUsers:
        - CN=fred
        - sam
        - CN=edward
    # ...
```

22.8. KAFKAAUTHORIZATIONOPA SCHEMA PROPERTIES

The **type** property is a discriminator that distinguishes use of the **KafkaAuthorizationOpa** type from **KafkaAuthorizationSimple**, **KafkaAuthorizationKeycloak**, **KafkaAuthorizationCustom**. It must have the value **opa** for the type **KafkaAuthorizationOpa**.

Property	Description
type string	Must be **opa**.
url string	The URL used to connect to the Open Policy Agent server. The URL has to include the policy which will be queried by the authorizer. This option is required.
allowOnError boolean	Defines whether a Kafka client should be allowed or denied by default when the authorizer fails to query the Open Policy Agent, for example, when it is temporarily unavailable). Defaults to **false** - all actions will be denied.
initialCacheCapacity integer	Initial capacity of the local cache used by the authorizer to avoid querying the Open Policy Agent for every request Defaults to **5000**.
maximumCacheSize integer	Maximum capacity of the local cache used by the authorizer to avoid querying the Open Policy Agent for every request. Defaults to **50000**.

Property	Description
expireAfterMs integer	The expiration of the records kept in the local cache to avoid querying the Open Policy Agent for every request. Defines how often the cached authorization decisions are reloaded from the Open Policy Agent server. In milliseconds. Defaults to **3600000**.
tlsTrustedCertificates **CertSecretSource** array	Trusted certificates for TLS connection to the OPA server.
superUsers string array	List of super users, which is specifically a list of user principals that have unlimited access rights.
enableMetrics boolean	Defines whether the Open Policy Agent authorizer plugin should provide metrics. Defaults to **false**.

CHAPTER 23. KAFKAAUTHORIZATIONKEYCLOAK SCHEMA REFERENCE

Used in: **KafkaClusterSpec**

The **type** property is a discriminator that distinguishes use of the **KafkaAuthorizationKeycloak** type from **KafkaAuthorizationSimple**, **KafkaAuthorizationOpa**, **KafkaAuthorizationCustom**. It must have the value **keycloak** for the type **KafkaAuthorizationKeycloak**.

Property	Description
type string	Must be **keycloak**.
clientId string	OAuth Client ID which the Kafka client can use to authenticate against the OAuth server and use the token endpoint URI.
tokenEndpointUri string	Authorization server token endpoint URI.
tlsTrustedCertificates **CertSecretSource** array	Trusted certificates for TLS connection to the OAuth server.
disableTlsHostnameVerification boolean	Enable or disable TLS hostname verification. Default value is **false**.
delegateToKafkaAcls boolean	Whether authorization decision should be delegated to the 'Simple' authorizer if DENIED by Red Hat Single Sign-On Authorization Services policies. Default value is **false**.
grantsRefreshPeriodSeconds integer	The time between two consecutive grants refresh runs in seconds. The default value is 60.
grantsRefreshPoolSize integer	The number of threads to use to refresh grants for active sessions. The more threads, the more parallelism, so the sooner the job completes. However, using more threads places a heavier load on the authorization server. The default value is 5.
grantsGcPeriodSeconds integer	The time, in seconds, between consecutive runs of a job that cleans stale grants from the cache. The default value is 300.

Property	Description
grantsAlwaysLatest boolean	Controls whether the latest grants are fetched for a new session. When enabled, grants are retrieved from Red Hat Single Sign-On and cached for the user. The default value is **false**.
superUsers string array	List of super users. Should contain list of user principals which should get unlimited access rights.
connectTimeoutSeconds integer	The connect timeout in seconds when connecting to authorization server. If not set, the effective connect timeout is 60 seconds.
readTimeoutSeconds integer	The read timeout in seconds when connecting to authorization server. If not set, the effective read timeout is 60 seconds.
httpRetries integer	The maximum number of retries to attempt if an initial HTTP request fails. If not set, the default is to not attempt any retries.
enableMetrics boolean	Enable or disable OAuth metrics. The default value is **false**.
includeAcceptHeader boolean	Whether the Accept header should be set in requests to the authorization servers. The default value is **true**.
grantsMaxIdleTimeSeconds integer	The time, in seconds, after which an idle grant can be evicted from the cache. The default value is 300.

CHAPTER 24. KAFKAAUTHORIZATIONCUSTOM SCHEMA REFERENCE

Used in: **KafkaClusterSpec**

Full list of **KafkaAuthorizationCustom** schema properties

To use custom authorization in AMQ Streams, you can configure your own **Authorizer** plugin to define Access Control Lists (ACLs).

ACLs allow you to define which users have access to which resources at a granular level.

Configure the **Kafka** custom resource to use custom authorization. Set the **type** property in the **authorization** section to the value **custom**, and the set following properties.

IMPORTANT

The custom authorizer must implement the **org.apache.kafka.server.authorizer.Authorizer** interface, and support configuration of **superusers** using the super.users configuration property.

24.1. AUTHORIZERCLASS

(Required) Java class that implements the **org.apache.kafka.server.authorizer.Authorizer** interface to support custom ACLs.

24.2. SUPERUSERS

A list of user principals treated as super users, so that they are always allowed without querying ACL rules.

You can add configuration for initializing the custom authorizer using **Kafka.spec.kafka.config**.

An example of custom authorization configuration under Kafka.spec

```
apiVersion: kafka.strimzi.io/v1beta2
kind: Kafka
metadata:
  name: my-cluster
  namespace: myproject
spec:
  kafka:
    # ...
    authorization:
      type: custom
      authorizerClass: io.mycompany.CustomAuthorizer
      superUsers:
        - CN=client_1
        - user_2
        - CN=client_3
    # ...
    config:
      authorization.custom.property1=value1
      authorization.custom.property2=value2
    # ...
```

In addition to the **Kafka** custom resource configuration, the JAR file containing the custom authorizer class along with its dependencies must be available on the classpath of the Kafka broker.

The AMQ Streams Maven build process provides a mechanism to add custom third-party libraries to the generated Kafka broker container image by adding them as dependencies in the **pom.xml** file under the **docker-images/kafka/kafka-thirdparty-libs** directory. The directory contains different folders for different Kafka versions. Choose the appropriate folder. Before modifying the **pom.xml** file, the third-party library must be available in a Maven repository, and that Maven repository must be accessible to the AMQ Streams build process.

NOTE

The **super.user** configuration option in the **config** property in **Kafka.spec.kafka** is ignored. Designate super users in the **authorization** property instead. For more information, see Kafka broker configuration.

Custom authorization can make use of group membership information extracted from the JWT token during authentication when using **oauth** authentication and configuring **groupsClaim** configuration attribute. Groups are available on the **OAuthKafkaPrincipal** object during authorize() call as follows:

```
    public List<AuthorizationResult> authorize(AuthorizableRequestContext requestContext,
List<Action> actions) {

        KafkaPrincipal principal = requestContext.principal();
        if (principal instanceof OAuthKafkaPrincipal) {
            OAuthKafkaPrincipal p = (OAuthKafkaPrincipal) principal;

            for (String group: p.getGroups()) {
                System.out.println("Group: " + group);
            }
        }
    }
```

24.3. KAFKAAUTHORIZATIONCUSTOM SCHEMA PROPERTIES

The **type** property is a discriminator that distinguishes use of the **KafkaAuthorizationCustom** type from **KafkaAuthorizationSimple**, **KafkaAuthorizationOpa**, **KafkaAuthorizationKeycloak**. It must have the value **custom** for the type **KafkaAuthorizationCustom**.

Property	Description
type string	Must be **custom**.
authorizerClass string	Authorization implementation class, which must be available in classpath.

Property	Description
superUsers string array	List of super users, which are user principals with unlimited access rights.
supportsAdminApi boolean	Indicates whether the custom authorizer supports the APIs for managing ACLs using the Kafka Admin API. Defaults to **false**.

CHAPTER 25. RACK SCHEMA REFERENCE

Used in: **KafkaBridgeSpec**, **KafkaClusterSpec**, **KafkaConnectSpec**, **KafkaMirrorMaker2Spec**

Full list of **Rack** schema properties

The **rack** option configures rack awareness. A *rack* can represent an availability zone, data center, or an actual rack in your data center. The *rack* is configured through a **topologyKey**. **topologyKey** identifies a label on OpenShift nodes that contains the name of the topology in its value. An example of such a label is **topology.kubernetes.io/zone** (or **failure-domain.beta.kubernetes.io/zone** on older OpenShift versions), which contains the name of the availability zone in which the OpenShift node runs. You can configure your Kafka cluster to be aware of the *rack* in which it runs, and enable additional features such as spreading partition replicas across different racks or consuming messages from the closest replicas.

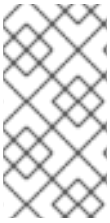
For more information about OpenShift node labels, see Well-Known Labels, Annotations and Taints . Consult your OpenShift administrator regarding the node label that represents the zone or rack into which the node is deployed.

25.1. SPREADING PARTITION REPLICAS ACROSS RACKS

When rack awareness is configured, AMQ Streams will set **broker.rack** configuration for each Kafka broker. The **broker.rack** configuration assigns a rack ID to each broker. When **broker.rack** is configured, Kafka brokers will spread partition replicas across as many different racks as possible. When replicas are spread across multiple racks, the probability that multiple replicas will fail at the same time is lower than if they would be in the same rack. Spreading replicas improves resiliency, and is important for availability and reliability. To enable rack awareness in Kafka, add the **rack** option to the **.spec.kafka** section of the **Kafka** custom resource as shown in the example below.

Example rack configuration for Kafka

```
apiVersion: kafka.strimzi.io/v1beta2
kind: Kafka
metadata:
  name: my-cluster
spec:
  kafka:
    # ...
    rack:
      topologyKey: topology.kubernetes.io/zone
    # ...
```



NOTE

The *rack* in which brokers are running can change in some cases when the pods are deleted or restarted. As a result, the replicas running in different racks might then share the same rack. Use Cruise Control and the **KafkaRebalance** resource with the **RackAwareGoal** to make sure that replicas remain distributed across different racks.

When rack awareness is enabled in the **Kafka** custom resource, AMQ Streams will automatically add the OpenShift **preferredDuringSchedulingIgnoredDuringExecution** affinity rule to distribute the Kafka brokers across the different racks. However, the *preferred* rule does not guarantee that the brokers will be spread. Depending on your exact OpenShift and Kafka configurations, you should add additional

affinity rules or configure **topologySpreadConstraints** for both ZooKeeper and Kafka to make sure the nodes are properly distributed accross as many racks as possible. For more information see Configuring pod scheduling.

25.2. CONSUMING MESSAGES FROM THE CLOSEST REPLICAS

Rack awareness can also be used in consumers to fetch data from the closest replica. This is useful for reducing the load on your network when a Kafka cluster spans multiple datacenters and can also reduce costs when running Kafka in public clouds. However, it can lead to increased latency.

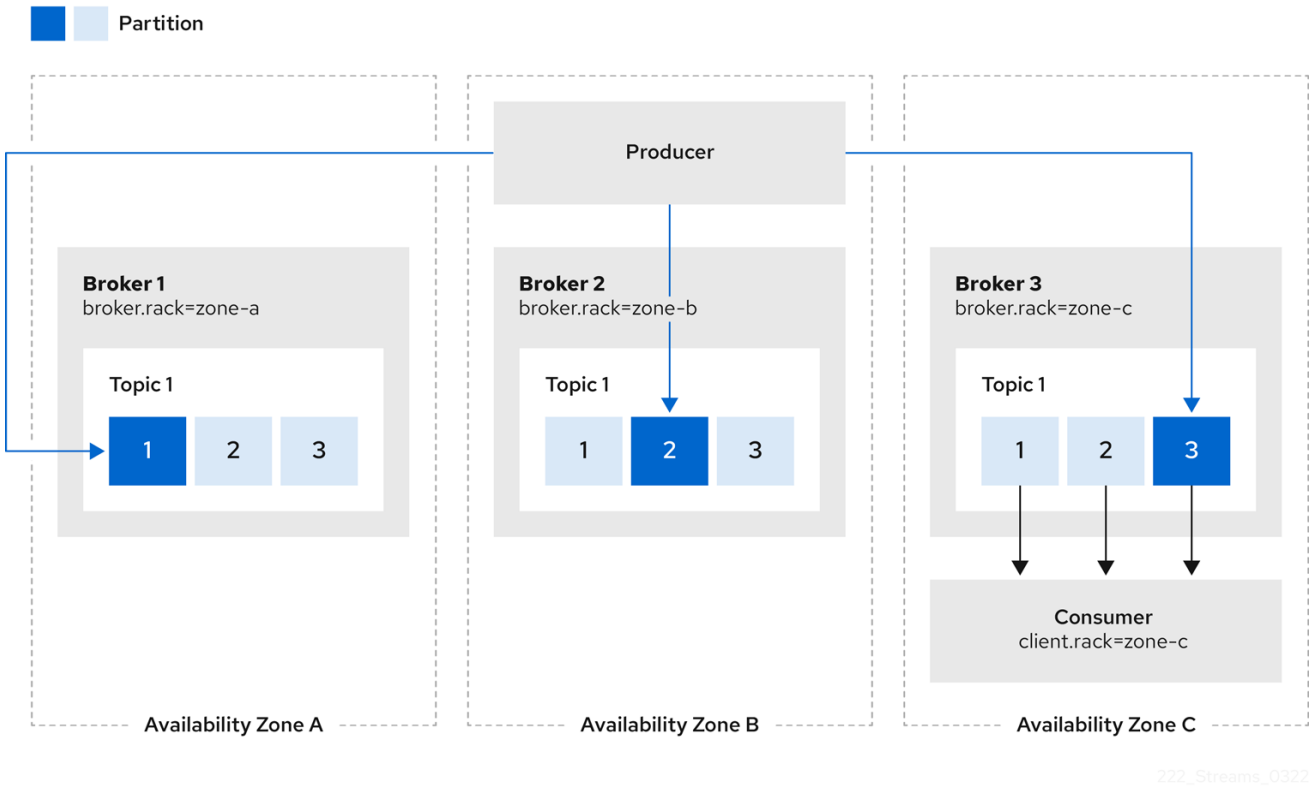
In order to be able to consume from the closest replica, rack awareness has to be configured in the Kafka cluster, and the **RackAwareReplicaSelector** has to be enabled. The replica selector plugin provides the logic that enables clients to consume from the nearest replica. The default implementation uses **LeaderSelector** to always select the leader replica for the client. Specify **RackAwareReplicaSelector** for the **replica.selector.class** to switch from the default implementation.

Example rack configuration with enabled replica-aware selector

```
apiVersion: kafka.strimzi.io/v1beta2
kind: Kafka
metadata:
  name: my-cluster
spec:
  kafka:
    # ...
    rack:
      topologyKey: topology.kubernetes.io/zone
    config:
      # ...
      replica.selector.class: org.apache.kafka.common.replica.RackAwareReplicaSelector
    # ...
```

In addition to the Kafka broker configuration, you also need to specify the **client.rack** option in your consumers. The **client.rack** option should specify the *rack ID* in which the consumer is running. **RackAwareReplicaSelector** associates matching **broker.rack** and **client.rack** IDs, to find the nearest replica and consume from it. If there are multiple replicas in the same rack, **RackAwareReplicaSelector** always selects the most up-to-date replica. If the rack ID is not specified, or if it cannot find a replica with the same rack ID, it will fall back to the leader replica.

Figure 25.1. Example showing client consuming from replicas in the same availability zone



You can also configure Kafka Connect, MirrorMaker 2 and Kafka Bridge so that connectors consume messages from the closest replicas. You enable rack awareness in the **KafkaConnect**, **KafkaMirrorMaker2**, and **KafkaBridge** custom resources. The configuration does does not set affinity rules, but you can also configure **affinity** or **topologySpreadConstraints**. For more information see Configuring pod scheduling.

When deploying Kafka Connect using AMQ Streams, you can use the **rack** section in the **KafkaConnect** custom resource to automatically configure the **client.rack** option.

Example rack configuration for Kafka Connect

```
apiVersion: kafka.strimzi.io/v1beta2
kind: KafkaConnect
# ...
spec:
  # ...
  rack:
    topologyKey: topology.kubernetes.io/zone
  # ...
```

When deploying MirrorMaker 2 using AMQ Streams, you can use the **rack** section in the **KafkaMirrorMaker2** custom resource to automatically configure the **client.rack** option.

Example rack configuration for MirrorMaker 2

```
apiVersion: kafka.strimzi.io/v1beta2
kind: KafkaMirrorMaker2
# ...
spec:
```

```
  # ...
  rack:
    topologyKey: topology.kubernetes.io/zone
  # ...
```

When deploying Kafka Bridge using AMQ Streams, you can use the **rack** section in the **KafkaBridge** custom resource to automatically configure the **client.rack** option.

Example rack configuration for Kafka Bridge

```
apiVersion: kafka.strimzi.io/v1beta2
kind: KafkaBridge
# ...
spec:
  # ...
  rack:
    topologyKey: topology.kubernetes.io/zone
  # ...
```

25.3. RACK SCHEMA PROPERTIES

Property	Description
topologyKey string	A key that matches labels assigned to the OpenShift cluster nodes. The value of the label is used to set a broker's **broker.rack** config, and the **client.rack** config for Kafka Connect or MirrorMaker 2.

CHAPTER 26. PROBE SCHEMA REFERENCE

Used in: **CruiseControlSpec**, **EntityTopicOperatorSpec**, **EntityUserOperatorSpec**, **KafkaBridgeSpec**, **KafkaClusterSpec**, **KafkaConnectSpec**, **KafkaExporterSpec**, **KafkaMirrorMaker2Spec**, **KafkaMirrorMakerSpec**, **TlsSidecar**, **ZookeeperClusterSpec**

Property	Description
failureThreshold integer	Minimum consecutive failures for the probe to be considered failed after having succeeded. Defaults to 3. Minimum value is 1.
initialDelaySeconds integer	The initial delay before first the health is first checked. Default to 15 seconds. Minimum value is 0.
periodSeconds integer	How often (in seconds) to perform the probe. Default to 10 seconds. Minimum value is 1.
successThreshold integer	Minimum consecutive successes for the probe to be considered successful after having failed. Defaults to 1. Must be 1 for liveness. Minimum value is 1.
timeoutSeconds integer	The timeout for each attempted health check. Default to 5 seconds. Minimum value is 1.

CHAPTER 27. JVMOPTIONS SCHEMA REFERENCE

Used in: **CruiseControlSpec**, **EntityTopicOperatorSpec**, **EntityUserOperatorSpec**, **KafkaBridgeSpec**, **KafkaClusterSpec**, **KafkaConnectSpec**, **KafkaMirrorMaker2Spec**, **KafkaMirrorMakerSpec**, **KafkaNodePoolSpec**, **ZookeeperClusterSpec**

Property	Description
-XX map	A map of -XX options to the JVM.
-Xms string	-Xms option to to the JVM.
-Xmx string	-Xmx option to to the JVM.
gcLoggingEnabled boolean	Specifies whether the Garbage Collection logging is enabled. The default is false.
javaSystemProperties **SystemProperty** array	A map of additional system properties which will be passed using the **-D** option to the JVM.

CHAPTER 28. SYSTEMPROPERTY SCHEMA REFERENCE

Used in: **JvmOptions**

Property	Description
name	The system property name.
string	
value	The system property value.
string	

CHAPTER 29. KAFKAJMXOPTIONS SCHEMA REFERENCE

Used in: **KafkaClusterSpec**, **KafkaConnectSpec**, **KafkaMirrorMaker2Spec**, **ZookeeperClusterSpec**

Full list of **KafkaJmxOptions** schema properties

Configures JMX connection options.

Get JMX metrics from Kafka brokers, ZooKeeper nodes, Kafka Connect, and MirrorMaker 2. by connecting to port 9999. Use the **jmxOptions** property to configure a password-protected or an unprotected JMX port. Using password protection prevents unauthorized pods from accessing the port.

You can then obtain metrics about the component.

For example, for each Kafka broker you can obtain bytes-per-second usage data from clients, or the request rate of the network of the broker.

To enable security for the JMX port, set the **type** parameter in the **authentication** field to **password**.

Example password-protected JMX configuration for Kafka brokers and ZooKeeper nodes

```
apiVersion: kafka.strimzi.io/v1beta2
kind: Kafka
metadata:
  name: my-cluster
spec:
  kafka:
    # ...
    jmxOptions:
      authentication:
        type: "password"
    # ...
  zookeeper:
    # ...
    jmxOptions:
      authentication:
        type: "password"
    #...
```

You can then deploy a pod into a cluster and obtain JMX metrics using the headless service by specifying which broker you want to address.

For example, to get JMX metrics from broker *0* you specify:

```
"CLUSTER-NAME-kafka-0.CLUSTER-NAME-kafka-brokers"
```

***CLUSTER-NAME*-kafka-0** is name of the broker pod, and ***CLUSTER-NAME*-kafka-brokers** is the name of the headless service to return the IPs of the broker pods.

If the JMX port is secured, you can get the username and password by referencing them from the JMX Secret in the deployment of your pod.

For an unprotected JMX port, use an empty object **{}** to open the JMX port on the headless service. You deploy a pod and obtain metrics in the same way as for the protected port, but in this case any pod can read from the JMX port.

Example open port JMX configuration for Kafka brokers and ZooKeeper nodes

```
apiVersion: kafka.strimzi.io/v1beta2
kind: Kafka
metadata:
  name: my-cluster
spec:
  kafka:
    # ...
    jmxOptions: {}
    # ...
  zookeeper:
    # ...
    jmxOptions: {}
    # ...
```

Additional resources

- For more information on the Kafka component metrics exposed using JMX, see the Apache Kafka documentation.

29.1. KAFKAJMXOPTIONS SCHEMA PROPERTIES

Property	Description
authentication	Authentication configuration for connecting to the JMX port. The type depends on the value of the **authentication.type** property within the given object, which must be one of [password].
KafkaJmxAuthenticationPassword	

CHAPTER 30. KAFKAJMXAUTHENTICATIONPASSWORD SCHEMA REFERENCE

Used in: **KafkaJmxOptions**

The **type** property is a discriminator that distinguishes use of the **KafkaJmxAuthenticationPassword** type from other subtypes which may be added in the future. It must have the value **password** for the type **KafkaJmxAuthenticationPassword**.

Property	Description
type	Must be **password**.
string	

CHAPTER 31. JMXPROMETHEUSEXPORTERMETRICS SCHEMA REFERENCE

Used in: **CruiseControlSpec**, **KafkaClusterSpec**, **KafkaConnectSpec**, **KafkaMirrorMaker2Spec**, **KafkaMirrorMakerSpec**, **ZookeeperClusterSpec**

The **type** property is a discriminator that distinguishes use of the **JmxPrometheusExporterMetrics** type from other subtypes which may be added in the future. It must have the value **jmxPrometheusExporter** for the type **JmxPrometheusExporterMetrics**.

Property	Description
type	Must be **jmxPrometheusExporter**.
string	
valueFrom	ConfigMap entry where the Prometheus JMX Exporter configuration is stored. For details of the structure of this configuration, see the Prometheus JMX Exporter.
ExternalConfigurationReference	

CHAPTER 32. EXTERNALCONFIGURATIONREFERENCE SCHEMA REFERENCE

Used in: **ExternalLogging**, **JmxPrometheusExporterMetrics**

Property	Description
configMapKeyRef	Reference to the key in the ConfigMap containing the configuration. For more information, see the external documentation for core/v1 configmapkeyselector.
ConfigMapKeySelector	

CHAPTER 33. INLINELOGGING SCHEMA REFERENCE

Used in: **CruiseControlSpec**, **EntityTopicOperatorSpec**, **EntityUserOperatorSpec**, **KafkaBridgeSpec**, **KafkaClusterSpec**, **KafkaConnectSpec**, **KafkaMirrorMaker2Spec**, **KafkaMirrorMakerSpec**, **ZookeeperClusterSpec**

The **type** property is a discriminator that distinguishes use of the **InlineLogging** type from **ExternalLogging**. It must have the value **inline** for the type **InlineLogging**.

Property	Description
type	Must be **inline**.
string	
loggers	A Map from logger name to logger level.
map	

CHAPTER 34. EXTERNALLOGGING SCHEMA REFERENCE

Used in: **CruiseControlSpec**, **EntityTopicOperatorSpec**, **EntityUserOperatorSpec**, **KafkaBridgeSpec**, **KafkaClusterSpec**, **KafkaConnectSpec**, **KafkaMirrorMaker2Spec**, **KafkaMirrorMakerSpec**, **ZookeeperClusterSpec**

The **type** property is a discriminator that distinguishes use of the **ExternalLogging** type from **InlineLogging**. It must have the value **external** for the type **ExternalLogging**.

Property	Description
type	Must be **external**.
string	
valueFrom	**ConfigMap** entry where the logging configuration is stored.
ExternalConfigurationReference	

CHAPTER 35. KAFKACLUSTERTEMPLATE SCHEMA REFERENCE

Used in: **KafkaClusterSpec**

Property	Description
statefulset **StatefulSetTemplate**	**The statefulset property has been deprecated.** Support for StatefulSets was removed in AMQ Streams 2.5. This property is ignored. Template for Kafka **StatefulSet**.
pod **PodTemplate**	Template for Kafka **Pods**.
bootstrapService **InternalServiceTemplate**	Template for Kafka bootstrap **Service**.
brokersService **InternalServiceTemplate**	Template for Kafka broker **Service**.
externalBootstrapService **ResourceTemplate**	Template for Kafka external bootstrap **Service**.
perPodService **ResourceTemplate**	Template for Kafka per-pod **Services** used for access from outside of OpenShift.
externalBootstrapRoute **ResourceTemplate**	Template for Kafka external bootstrap **Route**.
perPodRoute **ResourceTemplate**	Template for Kafka per-pod **Routes** used for access from outside of OpenShift.
externalBootstrapIngress **ResourceTemplate**	Template for Kafka external bootstrap **Ingress**.
perPodIngress **ResourceTemplate**	Template for Kafka per-pod **Ingress** used for access from outside of OpenShift.

Property	Description
persistentVolumeClaim	Template for all Kafka **PersistentVolumeClaims**.
ResourceTemplate	
podDisruptionBudget	Template for Kafka **PodDisruptionBudget**.
PodDisruptionBudgetTemplate	
kafkaContainer	Template for the Kafka broker container.
ContainerTemplate	
initContainer	Template for the Kafka init container.
ContainerTemplate	
clusterCaCert	Template for Secret with Kafka Cluster certificate public key.
ResourceTemplate	
serviceAccount	Template for the Kafka service account.
ResourceTemplate	
jmxSecret	Template for Secret of the Kafka Cluster JMX authentication.
ResourceTemplate	
clusterRoleBinding	Template for the Kafka ClusterRoleBinding.
ResourceTemplate	
podSet	Template for Kafka **StrimziPodSet** resource.
ResourceTemplate	

CHAPTER 36. STATEFULSETTEMPLATE SCHEMA REFERENCE

Used in: **KafkaClusterTemplate**, **ZookeeperClusterTemplate**

Property	Description
metadata	Metadata applied to the resource.
MetadataTemplate	
podManagementPolicy	PodManagementPolicy which will be used for this StatefulSet. Valid values are **Parallel** and **OrderedReady**. Defaults to **Parallel**.
string (one of [OrderedReady, Parallel])	

CHAPTER 37. METADATATEMPLATE SCHEMA REFERENCE

Used in: **BuildConfigTemplate**, **DeploymentTemplate**, **InternalServiceTemplate**, **PodDisruptionBudgetTemplate**, **PodTemplate**, **ResourceTemplate**, **StatefulSetTemplate**

Full list of **MetadataTemplate** schema properties

Labels and **Annotations** are used to identify and organize resources, and are configured in the **metadata** property.

For example:

```
# ...
template:
  pod:
    metadata:
      labels:
        label1: value1
        label2: value2
      annotations:
        annotation1: value1
        annotation2: value2
# ...
```

The **labels** and **annotations** fields can contain any labels or annotations that do not contain the reserved string **strimzi.io**. Labels and annotations containing **strimzi.io** are used internally by AMQ Streams and cannot be configured.

37.1. METADATATEMPLATE SCHEMA PROPERTIES

Property	Description
labels	Labels added to the OpenShift resource.
map	
annotations	Annotations added to the OpenShift resource.
map	

CHAPTER 38. PODTEMPLATE SCHEMA REFERENCE

Used in: **CruiseControlTemplate**, **EntityOperatorTemplate**, **JmxTransTemplate**, **KafkaBridgeTemplate**, **KafkaClusterTemplate**, **KafkaConnectTemplate**, **KafkaExporterTemplate**, **KafkaMirrorMakerTemplate**, **KafkaNodePoolTemplate**, **ZookeeperClusterTemplate**

Full list of **PodTemplate** schema properties

Configures the template for Kafka pods.

Example PodTemplate configuration

```
# ...
template:
  pod:
    metadata:
      labels:
        label1: value1
      annotations:
        anno1: value1
    imagePullSecrets:
      - name: my-docker-credentials
    securityContext:
      runAsUser: 1000001
      fsGroup: 0
    terminationGracePeriodSeconds: 120
# ...
```

38.1. HOSTALIASES

Use the **hostAliases** property to a specify a list of hosts and IP addresses, which are injected into the **/etc/hosts** file of the pod.

This configuration is especially useful for Kafka Connect or MirrorMaker when a connection outside of the cluster is also requested by users.

Example hostAliases configuration

```
apiVersion: kafka.strimzi.io/v1beta2
kind: KafkaConnect
#...
spec:
  # ...
  template:
    pod:
      hostAliases:
      - ip: "192.168.1.86"
        hostnames:
        - "my-host-1"
        - "my-host-2"
      #...
```

38.2. PODTEMPLATE SCHEMA PROPERTIES

Property	Description
metadata **MetadataTemplate**	Metadata applied to the resource.
imagePullSecrets LocalObjectReference array	List of references to secrets in the same namespace to use for pulling any of the images used by this Pod. When the **STRIMZI_IMAGE_PULL_SECRETS** environment variable in Cluster Operator and the **imagePullSecrets** option are specified, only the **imagePullSecrets** variable is used and the **STRIMZI_IMAGE_PULL_SECRETS** variable is ignored. For more information, see the external documentation for core/v1 localobjectreference.
securityContext PodSecurityContext	Configures pod-level security attributes and common container settings. For more information, see the external documentation for core/v1 podsecuritycontext.
terminationGracePeriodSeconds integer	The grace period is the duration in seconds after the processes running in the pod are sent a termination signal, and the time when the processes are forcibly halted with a kill signal. Set this value to longer than the expected cleanup time for your process. Value must be a non-negative integer. A zero value indicates delete immediately. You might need to increase the grace period for very large Kafka clusters, so that the Kafka brokers have enough time to transfer their work to another broker before they are terminated. Defaults to 30 seconds.
affinity Affinity	The pod's affinity rules. For more information, see the external documentation for core/v1 affinity.
tolerations Toleration array	The pod's tolerations. For more information, see the external documentation for core/v1 toleration.
priorityClassName string	The name of the priority class used to assign priority to the pods. For more information about priority classes, see Pod Priority and Preemption.
schedulerName string	The name of the scheduler used to dispatch this **Pod**. If not specified, the default scheduler will be used.

Property	Description
hostAliases HostAlias array	The pod's HostAliases. HostAliases is an optional list of hosts and IPs that will be injected into the Pod's hosts file if specified. For more information, see the external documentation for core/v1 hostalias.
tmpDirSizeLimit string	Defines the total amount (for example **1Gi**) of local storage required for temporary EmptyDir volume (/**tmp**). Default value is **5Mi**.
enableServiceLinks boolean	Indicates whether information about services should be injected into Pod's environment variables.
topologySpreadConstraints TopologySpreadConstraint array	The pod's topology spread constraints. For more information, see the external documentation for core/v1 topologyspreadconstraint.

CHAPTER 39. INTERNALSERVICETEMPLATE SCHEMA REFERENCE

Used in: **CruiseControlTemplate**, **KafkaBridgeTemplate**, **KafkaClusterTemplate**, **KafkaConnectTemplate**, **ZookeeperClusterTemplate**

Property	Description
metadata **MetadataTemplate**	Metadata applied to the resource.
ipFamilyPolicy string (one of [RequireDualStack, SingleStack, PreferDualStack])	Specifies the IP Family Policy used by the service. Available options are **SingleStack**, **PreferDualStack** and **RequireDualStack**. **SingleStack** is for a single IP family. **PreferDualStack** is for two IP families on dual-stack configured clusters or a single IP family on single-stack clusters. **RequireDualStack** fails unless there are two IP families on dual-stack configured clusters. If unspecified, OpenShift will choose the default value based on the service type.
ipFamilies string (one or more of [IPv6, IPv4]) array	Specifies the IP Families used by the service. Available options are **IPv4** and **IPv6**. If unspecified, OpenShift will choose the default value based on the **ipFamilyPolicy** setting.

CHAPTER 40. RESOURCETEMPLATE SCHEMA REFERENCE

Used in: **CruiseControlTemplate**, **EntityOperatorTemplate**, **JmxTransTemplate**, **KafkaBridgeTemplate**, **KafkaClusterTemplate**, **KafkaConnectTemplate**, **KafkaExporterTemplate**, **KafkaMirrorMakerTemplate**, **KafkaNodePoolTemplate**, **KafkaUserTemplate**, **ZookeeperClusterTemplate**

Property	Description
metadata	Metadata applied to the resource.
MetadataTemplate	

CHAPTER 41. PODDISRUPTIONBUDGETTEMPLATE SCHEMA REFERENCE

Used in: **CruiseControlTemplate**, **KafkaBridgeTemplate**, **KafkaClusterTemplate**, **KafkaConnectTemplate**, **KafkaMirrorMakerTemplate**, **ZookeeperClusterTemplate**

Full list of **PodDisruptionBudgetTemplate** schema properties

A **PodDisruptionBudget** (PDB) is an OpenShift resource that ensures high availability by specifying the minimum number of pods that must be available during planned maintenance or upgrades. AMQ Streams creates a PDB for every new **StrimziPodSet** or **Deployment**. By default, the PDB allows only one pod to be unavailable at any given time. You can increase the number of unavailable pods allowed by changing the default value of the **maxUnavailable** property.

StrimziPodSet custom resources manage pods using a custom controller that cannot use the **maxUnavailable** value directly. Instead, the **maxUnavailable** value is automatically converted to a **minAvailable** value when creating the PDB resource, which effectively serves the same purpose, as illustrated in the following examples:

- If there are three broker pods and the **maxUnavailable** property is set to **1** in the **Kafka** resource, the **minAvailable** setting is **2**, allowing one pod to be unavailable.
- If there are three broker pods and the **maxUnavailable** property is set to **0** (zero), the **minAvailable** setting is **3**, requiring all three broker pods to be available and allowing zero pods to be unavailable.

Example PodDisruptionBudget template configuration

```
# ...
template:
  podDisruptionBudget:
    metadata:
      labels:
        key1: label1
        key2: label2
      annotations:
        key1: label1
        key2: label2
    maxUnavailable: 1
# ...
```

41.1. PODDISRUPTIONBUDGETTEMPLATE SCHEMA PROPERTIES

Property	Description
metadata **MetadataTemplate**	Metadata to apply to the **PodDisruptionBudgetTemplate** resource.

Property	Description
maxUnavailable integer	Maximum number of unavailable pods to allow automatic Pod eviction. A Pod eviction is allowed when the **maxUnavailable** number of pods or fewer are unavailable after the eviction. Setting this value to 0 prevents all voluntary evictions, so the pods must be evicted manually. Defaults to 1.

CHAPTER 42. CONTAINERTEMPLATE SCHEMA REFERENCE

Used in: **CruiseControlTemplate**, **EntityOperatorTemplate**, **JmxTransTemplate**, **KafkaBridgeTemplate**, **KafkaClusterTemplate**, **KafkaConnectTemplate**, **KafkaExporterTemplate**, **KafkaMirrorMakerTemplate**, **KafkaNodePoolTemplate**, **ZookeeperClusterTemplate**

Full list of **ContainerTemplate** schema properties

You can set custom security context and environment variables for a container.

The environment variables are defined under the **env** property as a list of objects with **name** and **value** fields. The following example shows two custom environment variables and a custom security context set for the Kafka broker containers:

```
# ...
template:
  kafkaContainer:
    env:
    - name: EXAMPLE_ENV_1
      value: example.env.one
    - name: EXAMPLE_ENV_2
      value: example.env.two
    securityContext:
      runAsUser: 2000
# ...
```

Environment variables prefixed with **KAFKA_** are internal to AMQ Streams and should be avoided. If you set a custom environment variable that is already in use by AMQ Streams, it is ignored and a warning is recorded in the log.

42.1. CONTAINERTEMPLATE SCHEMA PROPERTIES

Property	Description
env	Environment variables which should be applied to the container.
ContainerEnvVar array	
securityContext	Security context for the container. For more information, see the external documentation for core/v1 securitycontext.
SecurityContext	

CHAPTER 43. CONTAINERENVVAR SCHEMA REFERENCE

Used in: **ContainerTemplate**

Property	Description
name	The environment variable key.
string	
value	The environment variable value.
string	

CHAPTER 44. ZOOKEEPERCLUSTERSPEC SCHEMA REFERENCE

Used in: **KafkaSpec**

Full list of **ZookeeperClusterSpec** schema properties

Configures a ZooKeeper cluster.

44.1. CONFIG

Use the **config** properties to configure ZooKeeper options as keys.

The values can be one of the following JSON types:

- String
- Number
- Boolean

Exceptions

You can specify and configure the options listed in the ZooKeeper documentation.

However, AMQ Streams takes care of configuring and managing options related to the following, which cannot be changed:

- Security (encryption, authentication, and authorization)
- Listener configuration
- Configuration of data directories
- ZooKeeper cluster composition

Properties with the following prefixes cannot be set:

- **4lw.commands.whitelist**
- **authProvider**
- **clientPort**
- **dataDir**
- **dataLogDir**
- **quorum.auth**
- **reconfigEnabled**
- **requireClientAuthScheme**
- **secureClientPort**
- **server.**

- **snapshot.trust.empty**
- **standaloneEnabled**
- **serverCnxnFactory**
- **ssl.**
- **sslQuorum**

If the **config** property contains an option that cannot be changed, it is disregarded, and a warning message is logged to the Cluster Operator log file. All other supported options are forwarded to ZooKeeper, including the following exceptions to the options configured by AMQ Streams:

- Any **ssl** configuration for supported TLS versions and cipher suites

Example ZooKeeper configuration

```
apiVersion: kafka.strimzi.io/v1beta2
kind: Kafka
spec:
  kafka:
    # ...
  zookeeper:
    # ...
    config:
      autopurge.snapRetainCount: 3
      autopurge.purgeInterval: 2
    # ...
```

44.2. LOGGING

ZooKeeper has a configurable logger:

- **zookeeper.root.logger**

ZooKeeper uses the Apache **log4j** logger implementation.

Use the **logging** property to configure loggers and logger levels.

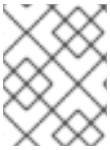
You can set the log levels by specifying the logger and level directly (inline) or use a custom (external) ConfigMap. If a ConfigMap is used, you set **logging.valueFrom.configMapKeyRef.name** property to the name of the ConfigMap containing the external logging configuration. Inside the ConfigMap, the logging configuration is described using **log4j.properties**. Both **logging.valueFrom.configMapKeyRef.name** and **logging.valueFrom.configMapKeyRef.key** properties are mandatory. A ConfigMap using the exact logging configuration specified is created with the custom resource when the Cluster Operator is running, then recreated after each reconciliation. If you do not specify a custom ConfigMap, default logging settings are used. If a specific logger value is not set, upper-level logger settings are inherited for that logger. For more information about log levels, see Apache logging services.

Here we see examples of **inline** and **external** logging. The **inline** logging specifies the root logger level. You can also set log levels for specific classes or loggers by adding them to the loggers property.

Inline logging

-

```
apiVersion: kafka.strimzi.io/v1beta2
kind: Kafka
spec:
  # ...
  zookeeper:
    # ...
    logging:
      type: inline
      loggers:
        zookeeper.root.logger: INFO
        log4j.logger.org.apache.zookeeper.server.FinalRequestProcessor: TRACE
        log4j.logger.org.apache.zookeeper.server.ZooKeeperServer: DEBUG
    # ...
```



NOTE

Setting a log level to **DEBUG** may result in a large amount of log output and may have performance implications.

External logging

```
apiVersion: kafka.strimzi.io/v1beta2
kind: Kafka
spec:
  # ...
  zookeeper:
    # ...
    logging:
      type: external
      valueFrom:
        configMapKeyRef:
          name: customConfigMap
          key: zookeeper-log4j.properties
  # ...
```

Garbage collector (GC)

Garbage collector logging can also be enabled (or disabled) using the **jvmOptions** property.

44.3. ZOOKEEPERCLUSTERSPEC SCHEMA PROPERTIES

Property	Description
replicas	The number of pods in the cluster.
integer	
image	The docker image for the pods.
string	

Property	Description
storage **EphemeralStorage**, **PersistentClaimStorage**	Storage configuration (disk). Cannot be updated. The type depends on the value of the **storage.type** property within the given object, which must be one of [ephemeral, persistent-claim].
config map	The ZooKeeper broker config. Properties with the following prefixes cannot be set: server., dataDir, dataLogDir, clientPort, authProvider, quorum.auth, requireClientAuthScheme, snapshot.trust.empty, standaloneEnabled, reconfigEnabled, 4lw.commands.whitelist, secureClientPort, ssl., serverCnxnFactory, sslQuorum (with the exception of: ssl.protocol, ssl.quorum.protocol, ssl.enabledProtocols, ssl.quorum.enabledProtocols, ssl.ciphersuites, ssl.quorum.ciphersuites, ssl.hostnameVerification, ssl.quorum.hostnameVerification).
livenessProbe **Probe**	Pod liveness checking.
readinessProbe **Probe**	Pod readiness checking.
jvmOptions **JvmOptions**	JVM Options for pods.
jmxOptions **KafkaJmxOptions**	JMX Options for Zookeeper nodes.
resources ResourceRequirements	CPU and memory resources to reserve. For more information, see the external documentation for core/v1 resourcerequirements.
metricsConfig **JmxPrometheusExporterMetrics**	Metrics configuration. The type depends on the value of the **metricsConfig.type** property within the given object, which must be one of [jmxPrometheusExporter].
logging **InlineLogging**, **ExternalLogging**	Logging configuration for ZooKeeper. The type depends on the value of the **logging.type** property within the given object, which must be one of [inline, external].

Property	Description
template **ZookeeperClusterTemplate**	Template for ZooKeeper cluster resources. The template allows users to specify how the OpenShift resources are generated.

CHAPTER 45. ZOOKEEPERCLUSTERTEMPLATE SCHEMA REFERENCE

Used in: **ZookeeperClusterSpec**

Property	Description
statefulset **StatefulSetTemplate**	**The statefulset property has been deprecated.** Support for StatefulSets was removed in AMQ Streams 2.5. This property is ignored. Template for ZooKeeper **StatefulSet**.
pod **PodTemplate**	Template for ZooKeeper **Pods**.
clientService **InternalServiceTemplate**	Template for ZooKeeper client **Service**.
nodesService **InternalServiceTemplate**	Template for ZooKeeper nodes **Service**.
persistentVolumeClaim **ResourceTemplate**	Template for all ZooKeeper **PersistentVolumeClaims**.
podDisruptionBudget **PodDisruptionBudgetTemplate**	Template for ZooKeeper **PodDisruptionBudget**.
zookeeperContainer **ContainerTemplate**	Template for the ZooKeeper container.
serviceAccount **ResourceTemplate**	Template for the ZooKeeper service account.
jmxSecret **ResourceTemplate**	Template for Secret of the Zookeeper Cluster JMX authentication.
podSet **ResourceTemplate**	Template for ZooKeeper **StrimziPodSet** resource.

CHAPTER 46. ENTITYOPERATORSPEC SCHEMA REFERENCE

Used in: **KafkaSpec**

Property	Description
topicOperator **EntityTopicOperatorSpec**	Configuration of the Topic Operator.
userOperator **EntityUserOperatorSpec**	Configuration of the User Operator.
tlsSidecar **TlsSidecar**	TLS sidecar configuration.
template **EntityOperatorTemplate**	Template for Entity Operator resources. The template allows users to specify how a **Deployment** and **Pod** is generated.

CHAPTER 47. ENTITYTOPICOPERATORSPEC SCHEMA REFERENCE

Used in: **EntityOperatorSpec**

Full list of **EntityTopicOperatorSpec** schema properties

Configures the Topic Operator.

47.1. LOGGING

The Topic Operator has a configurable logger:

- **rootLogger.level**

The Topic Operator uses the Apache **log4j2** logger implementation.

Use the **logging** property in the **entityOperator.topicOperator** field of the Kafka resource **Kafka** resource to configure loggers and logger levels.

You can set the log levels by specifying the logger and level directly (inline) or use a custom (external) ConfigMap. If a ConfigMap is used, you set **logging.valueFrom.configMapKeyRef.name** property to the name of the ConfigMap containing the external logging configuration. Inside the ConfigMap, the logging configuration is described using **log4j2.properties**. Both **logging.valueFrom.configMapKeyRef.name** and **logging.valueFrom.configMapKeyRef.key** properties are mandatory. A ConfigMap using the exact logging configuration specified is created with the custom resource when the Cluster Operator is running, then recreated after each reconciliation. If you do not specify a custom ConfigMap, default logging settings are used. If a specific logger value is not set, upper-level logger settings are inherited for that logger. For more information about log levels, see Apache logging services.

Here we see examples of **inline** and **external** logging. The **inline** logging specifies the root logger level. You can also set log levels for specific classes or loggers by adding them to the loggers property.

Inline logging

```
apiVersion: kafka.strimzi.io/v1beta2
kind: Kafka
metadata:
  name: my-cluster
spec:
  kafka:
    # ...
  zookeeper:
    # ...
  entityOperator:
    # ...
    topicOperator:
      watchedNamespace: my-topic-namespace
      reconciliationIntervalSeconds: 60
      logging:
        type: inline
        loggers:
          rootLogger.level: INFO
          logger.top.name: io.strimzi.operator.topic (1)
          logger.top.level: DEBUG (2)
```

```
      logger.toc.name: io.strimzi.operator.topic.TopicOperator 3
      logger.toc.level: TRACE 4
      logger.clients.level: DEBUG 5
  # ...
```

1. Creates a logger for the **topic** package.
2. Sets the logging level for the **topic** package.
3. Creates a logger for the **TopicOperator** class.
4. Sets the logging level for the **TopicOperator** class.
5. Changes the logging level for the default **clients** logger. The **clients** logger is part of the logging configuration provided with AMQ Streams. By default, it is set to **INFO**.

NOTE

When investigating an issue with the operator, it's usually sufficient to change the **rootLogger** to **DEBUG** to get more detailed logs. However, keep in mind that setting the log level to **DEBUG** may result in a large amount of log output and may have performance implications.

External logging

```
apiVersion: kafka.strimzi.io/v1beta2
kind: Kafka
metadata:
  name: my-cluster
spec:
  kafka:
    # ...
  zookeeper:
    # ...
  entityOperator:
    # ...
    topicOperator:
      watchedNamespace: my-topic-namespace
      reconciliationIntervalSeconds: 60
      logging:
        type: external
        valueFrom:
          configMapKeyRef:
            name: customConfigMap
            key: topic-operator-log4j2.properties
  # ...
```

Garbage collector (GC)

Garbage collector logging can also be enabled (or disabled) using the **jvmOptions** property.

47.2. ENTITYTOPICOPERATORSPEC SCHEMA PROPERTIES

Property	Description
watchedNamespace	The namespace the Topic Operator should watch.
string	
image	The image to use for the Topic Operator.
string	
reconciliationIntervalSeconds	Interval between periodic reconciliations.
integer	
zookeeperSessionTimeoutSeconds	Timeout for the ZooKeeper session.
integer	
startupProbe	Pod startup checking.
Probe	
livenessProbe	Pod liveness checking.
Probe	
readinessProbe	Pod readiness checking.
Probe	
resources	CPU and memory resources to reserve. For more information, see the external documentation for core/v1 resourcerequirements.
ResourceRequirements	
topicMetadataMaxAttempts	The number of attempts at getting topic metadata.
integer	
logging	Logging configuration. The type depends on the value of the **logging.type** property within the given object, which must be one of [inline, external].
InlineLogging, **ExternalLogging**	
jvmOptions	JVM Options for pods.
JvmOptions	

CHAPTER 48. ENTITYUSEROPERATORSPEC SCHEMA REFERENCE

Used in: **EntityOperatorSpec**

Full list of **EntityUserOperatorSpec** schema properties

Configures the User Operator.

48.1. LOGGING

The User Operator has a configurable logger:

- **rootLogger.level**

The User Operator uses the Apache **log4j2** logger implementation.

Use the **logging** property in the **entityOperator.userOperator** field of the **Kafka** resource to configure loggers and logger levels.

You can set the log levels by specifying the logger and level directly (inline) or use a custom (external) ConfigMap. If a ConfigMap is used, you set **logging.valueFrom.configMapKeyRef.name** property to the name of the ConfigMap containing the external logging configuration. Inside the ConfigMap, the logging configuration is described using **log4j2.properties**. Both **logging.valueFrom.configMapKeyRef.name** and **logging.valueFrom.configMapKeyRef.key** properties are mandatory. A ConfigMap using the exact logging configuration specified is created with the custom resource when the Cluster Operator is running, then recreated after each reconciliation. If you do not specify a custom ConfigMap, default logging settings are used. If a specific logger value is not set, upper-level logger settings are inherited for that logger. For more information about log levels, see Apache logging services.

Here we see examples of **inline** and **external** logging. The **inline** logging specifies the **rootLogger.level**. You can also set log levels for specific classes or loggers by adding them to the loggers property.

Inline logging

```
apiVersion: kafka.strimzi.io/v1beta2
kind: Kafka
metadata:
  name: my-cluster
spec:
  kafka:
    # ...
  zookeeper:
    # ...
  entityOperator:
    # ...
    userOperator:
      watchedNamespace: my-topic-namespace
      reconciliationIntervalSeconds: 60
      logging:
        type: inline
        loggers:
          rootLogger.level: INFO
          logger.uop.name: io.strimzi.operator.user 1
```

```
      logger.uop.level: DEBUG 2
      logger.abstractcache.name: io.strimzi.operator.user.operator.cache.AbstractCache 3
      logger.abstractcache.level: TRACE 4
      logger.jetty.level: DEBUG 5

  # ...
```

1. Creates a logger for the **user** package.
2. Sets the logging level for the **user** package.
3. Creates a logger for the **AbstractCache** class.
4. Sets the logging level for the **AbstractCache** class.
5. Changes the logging level for the default **jetty** logger. The **jetty** logger is part of the logging configuration provided with AMQ Streams. By default, it is set to **INFO**.

NOTE

When investigating an issue with the operator, it's usually sufficient to change the **rootLogger** to **DEBUG** to get more detailed logs. However, keep in mind that setting the log level to **DEBUG** may result in a large amount of log output and may have performance implications.

External logging

```
apiVersion: kafka.strimzi.io/v1beta2
kind: Kafka
metadata:
  name: my-cluster
spec:
  kafka:
    # ...
  zookeeper:
    # ...
  entityOperator:
    # ...
    userOperator:
      watchedNamespace: my-topic-namespace
      reconciliationIntervalSeconds: 60
      logging:
        type: external
        valueFrom:
          configMapKeyRef:
            name: customConfigMap
            key: user-operator-log4j2.properties
  # ...
```

Garbage collector (GC)

Garbage collector logging can also be enabled (or disabled) using the **jvmOptions** property.

48.2. ENTITYUSEROPERATORSPEC SCHEMA PROPERTIES

Property	Description
watchedNamespace string	The namespace the User Operator should watch.
image string	The image to use for the User Operator.
reconciliationIntervalSeconds integer	Interval between periodic reconciliations.
zookeeperSessionTimeoutSeconds integer	**The zookeeperSessionTimeoutSeconds property has been deprecated.** This property has been deprecated because ZooKeeper is not used anymore by the User Operator. Timeout for the ZooKeeper session.
secretPrefix string	The prefix that will be added to the KafkaUser name to be used as the Secret name.
livenessProbe **Probe**	Pod liveness checking.
readinessProbe **Probe**	Pod readiness checking.
resources ResourceRequirements	CPU and memory resources to reserve. For more information, see the external documentation for core/v1 resourcerequirements.
logging **InlineLogging**, **ExternalLogging**	Logging configuration. The type depends on the value of the **logging.type** property within the given object, which must be one of [inline, external].
jvmOptions **JvmOptions**	JVM Options for pods.

CHAPTER 49. TLSSIDECAR SCHEMA REFERENCE

Used in: **CruiseControlSpec**, **EntityOperatorSpec**

Full list of **TlsSidecar** schema properties

Configures a TLS sidecar, which is a container that runs in a pod, but serves a supporting purpose. In AMQ Streams, the TLS sidecar uses TLS to encrypt and decrypt communication between components and ZooKeeper.

The TLS sidecar is used in the Entity Operator.

The TLS sidecar is configured using the **tlsSidecar** property in **Kafka.spec.entityOperator**.

The TLS sidecar supports the following additional options:

- **image**
- **resources**
- **logLevel**
- **readinessProbe**
- **livenessProbe**

The **resources** property specifies the memory and CPU resources allocated for the TLS sidecar.

The **image** property configures the container image which will be used.

The **readinessProbe** and **livenessProbe** properties configure healthcheck probes for the TLS sidecar.

The **logLevel** property specifies the logging level. The following logging levels are supported:

- emerg
- alert
- crit
- err
- warning
- notice
- info
- debug

The default value is *notice*.

Example TLS sidecar configuration

```
apiVersion: kafka.strimzi.io/v1beta2
kind: Kafka
metadata:
```

```
  name: my-cluster
spec:
  # ...
  entityOperator:
    # ...
    tlsSidecar:
      resources:
        requests:
          cpu: 200m
          memory: 64Mi
        limits:
          cpu: 500m
          memory: 128Mi
    # ...
```

49.1. TLSSIDECAR SCHEMA PROPERTIES

Property	Description
image	The docker image for the container.
string	
livenessProbe	Pod liveness checking.
Probe	
logLevel	The log level for the TLS sidecar. Default value is **notice**.
string (one of [emerg, debug, crit, err, alert, warning, notice, info])	
readinessProbe	Pod readiness checking.
Probe	
resources	CPU and memory resources to reserve. For more information, see the external documentation for core/v1 resourcerequirements.
ResourceRequirements	

CHAPTER 50. ENTITYOPERATORTEMPLATE SCHEMA REFERENCE

Used in: **EntityOperatorSpec**

Property	Description
deployment	Template for Entity Operator **Deployment**.
DeploymentTemplate	
pod	Template for Entity Operator **Pods**.
PodTemplate	
topicOperatorContainer	Template for the Entity Topic Operator container.
ContainerTemplate	
userOperatorContainer	Template for the Entity User Operator container.
ContainerTemplate	
tlsSidecarContainer	Template for the Entity Operator TLS sidecar container.
ContainerTemplate	
serviceAccount	Template for the Entity Operator service account.
ResourceTemplate	
entityOperatorRole	Template for the Entity Operator Role.
ResourceTemplate	
topicOperatorRoleBinding	Template for the Entity Topic Operator RoleBinding.
ResourceTemplate	
userOperatorRoleBinding	Template for the Entity Topic Operator RoleBinding.
ResourceTemplate	

CHAPTER 51. DEPLOYMENTTEMPLATE SCHEMA REFERENCE

Used in: **CruiseControlTemplate**, **EntityOperatorTemplate**, **JmxTransTemplate**, **KafkaBridgeTemplate**, **KafkaConnectTemplate**, **KafkaExporterTemplate**, **KafkaMirrorMakerTemplate**

Full list of **DeploymentTemplate** schema properties

Use **deploymentStrategy** to specify the strategy used to replace old pods with new ones when deployment configuration changes.

Use one of the following values:

- **RollingUpdate**: Pods are restarted with zero downtime.
- **Recreate**: Pods are terminated before new ones are created.

Using the **Recreate** deployment strategy has the advantage of not requiring spare resources, but the disadvantage is the application downtime.

Example showing the deployment strategy set to Recreate.

```
# ...
template:
  deployment:
    deploymentStrategy: Recreate
# ...
```

This configuration change does not cause a rolling update.

51.1. DEPLOYMENTTEMPLATE SCHEMA PROPERTIES

Property	Description
metadata	Metadata applied to the resource.
MetadataTemplate	
deploymentStrategy	Pod replacement strategy for deployment configuration changes. Valid values are **RollingUpdate** and **Recreate**. Defaults to **RollingUpdate**.
string (one of [RollingUpdate, Recreate])	

CHAPTER 52. CERTIFICATEAUTHORITY SCHEMA REFERENCE

Used in: **KafkaSpec**

Configuration of how TLS certificates are used within the cluster. This applies to certificates used for both internal communication within the cluster and to certificates used for client access via **Kafka.spec.kafka.listeners.tls**.

Property	Description
generateCertificateAuthority boolean	If true then Certificate Authority certificates will be generated automatically. Otherwise the user will need to provide a Secret with the CA certificate. Default is true.
generateSecretOwnerReference boolean	If **true**, the Cluster and Client CA Secrets are configured with the **ownerReference** set to the **Kafka** resource. If the **Kafka** resource is deleted when **true**, the CA Secrets are also deleted. If **false**, the **ownerReference** is disabled. If the **Kafka** resource is deleted when **false**, the CA Secrets are retained and available for reuse. Default is **true**.
validityDays integer	The number of days generated certificates should be valid for. The default is 365.
renewalDays integer	The number of days in the certificate renewal period. This is the number of days before the a certificate expires during which renewal actions may be performed. When **generateCertificateAuthority** is true, this will cause the generation of a new certificate. When **generateCertificateAuthority** is true, this will cause extra logging at WARN level about the pending certificate expiry. Default is 30.
certificateExpirationPolicy string (one of [replace-key, renew-certificate])	How should CA certificate expiration be handled when **generateCertificateAuthority=true**. The default is for a new CA certificate to be generated reusing the existing private key.

CHAPTER 53. CRUISECONTROLSPEC SCHEMA REFERENCE

Used in: **KafkaSpec**

Full list of **CruiseControlSpec** schema properties

Configures a Cruise Control cluster.

Configuration options relate to:

- Goals configuration
- Capacity limits for resource distribution goals

53.1. CONFIG

Use the **config** properties to configure Cruise Control options as keys.

The values can be one of the following JSON types:

- String
- Number
- Boolean

Exceptions

You can specify and configure the options listed in the Cruise Control documentation.

However, AMQ Streams takes care of configuring and managing options related to the following, which cannot be changed:

- Security (encryption, authentication, and authorization)
- Connection to the Kafka cluster
- Client ID configuration
- ZooKeeper connectivity
- Web server configuration
- Self healing

Properties with the following prefixes cannot be set:

- **bootstrap.servers**
- **capacity.config.file**
- **client.id**
- **failed.brokers.zk.path**
- **kafka.broker.failure.detection.enable**

- **metric.reporter.sampler.bootstrap.servers**
- **network.**
- **request.reason.required**
- **security.**
- **self.healing.**
- **ssl.**
- **topic.config.provider.class**
- **two.step.**
- **webserver.accesslog.**
- **webserver.api.urlprefix**
- **webserver.http.**
- **webserver.session.path**
- **zookeeper.**

If the **config** property contains an option that cannot be changed, it is disregarded, and a warning message is logged to the Cluster Operator log file. All other supported options are forwarded to Cruise Control, including the following exceptions to the options configured by AMQ Streams:

- Any **ssl** configuration for supported TLS versions and cipher suites
- Configuration for **webserver** properties to enable Cross-Origin Resource Sharing (CORS)

Example Cruise Control configuration

```
apiVersion: kafka.strimzi.io/v1beta2
kind: Kafka
metadata:
  name: my-cluster
spec:
  # ...
  cruiseControl:
    # ...
    config:
      # Note that `default.goals` (superset) must also include all `hard.goals` (subset)
      default.goals: >
        com.linkedin.kafka.cruisecontrol.analyzer.goals.RackAwareGoal,
        com.linkedin.kafka.cruisecontrol.analyzer.goals.ReplicaCapacityGoal
      hard.goals: >
        com.linkedin.kafka.cruisecontrol.analyzer.goals.RackAwareGoal
      cpu.balance.threshold: 1.1
      metadata.max.age.ms: 300000
      send.buffer.bytes: 131072
      webserver.http.cors.enabled: true
```

```
    webserver.http.cors.origin: "*"
    webserver.http.cors.exposeheaders: "User-Task-ID,Content-Type"
  # ...
```

53.2. CROSS-ORIGIN RESOURCE SHARING (CORS)

Cross-Origin Resource Sharing (CORS) is a HTTP mechanism for controlling access to REST APIs. Restrictions can be on access methods or originating URLs of client applications. You can enable CORS with Cruise Control using the **webserver.http.cors.enabled** property in the **config**. When enabled, CORS permits read access to the Cruise Control REST API from applications that have different originating URLs than AMQ Streams. This allows applications from specified origins to use **GET** requests to fetch information about the Kafka cluster through the Cruise Control API. For example, applications can fetch information on the current cluster load or the most recent optimization proposal. **POST** requests are not permitted.

NOTE

For more information on using CORS with Cruise Control, see REST APIs in the Cruise Control Wiki.

Enabling CORS for Cruise Control

You enable and configure CORS in **Kafka.spec.cruiseControl.config**.

```
apiVersion: kafka.strimzi.io/v1beta2
kind: Kafka
metadata:
  name: my-cluster
spec:
  # ...
  cruiseControl:
    # ...
    config:
      webserver.http.cors.enabled: true (1)
      webserver.http.cors.origin: "*" (2)
      webserver.http.cors.exposeheaders: "User-Task-ID,Content-Type" (3)

    # ...
```

(1) Enables CORS.

(2) Specifies permitted origins for the **Access-Control-Allow-Origin** HTTP response header. You can use a wildcard or specify a single origin as a URL. If you use a wildcard, a response is returned following requests from any origin.

(3) Exposes specified header names for the **Access-Control-Expose-Headers** HTTP response header. Applications in permitted origins can read responses with the specified headers.

53.3. CRUISE CONTROL REST API SECURITY

The Cruise Control REST API is secured with HTTP Basic authentication and SSL to protect the cluster against potentially destructive Cruise Control operations, such as decommissioning Kafka brokers. We recommend that Cruise Control in AMQ Streams is **only used with these settings enabled**.

However, it is possible to disable these settings by specifying the following Cruise Control configuration:

- To disable the built-in HTTP Basic authentication, set **webserver.security.enable** to **false**.
- To disable the built-in SSL, set **webserver.ssl.enable** to **false**.

Cruise Control configuration to disable API authorization, authentication, and SSL

```
apiVersion: kafka.strimzi.io/v1beta2
kind: Kafka
metadata:
  name: my-cluster
spec:
  # ...
  cruiseControl:
    config:
      webserver.security.enable: false
      webserver.ssl.enable: false
# ...
```

53.4. BROKERCAPACITY

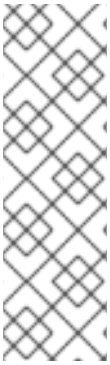
Cruise Control uses capacity limits to determine if optimization goals for resource capacity limits are being broken. There are four goals of this type:

- **DiskCapacityGoal** - Disk utilization capacity
- **CpuCapacityGoal** - CPU utilization capacity
- **NetworkInboundCapacityGoal** - Network inbound utilization capacity
- **NetworkOutboundCapacityGoal** - Network outbound utilization capacity

You specify capacity limits for Kafka broker resources in the **brokerCapacity** property in **Kafka.spec.cruiseControl** . They are enabled by default and you can change their default values. Capacity limits can be set for the following broker resources:

- **cpu** - CPU resource in millicores or CPU cores (Default: 1)
- **inboundNetwork** - Inbound network throughput in byte units per second (Default: 10000KiB/s)
- **outboundNetwork** - Outbound network throughput in byte units per second (Default: 10000KiB/s)

For network throughput, use an integer value with standard OpenShift byte units (K, M, G) or their bibyte (power of two) equivalents (Ki, Mi, Gi) per second.



NOTE

Disk and CPU capacity limits are automatically generated by AMQ Streams, so you do not need to set them. In order to guarantee accurate rebalance proposals when using CPU goals, you can set CPU requests equal to CPU limits in **Kafka.spec.kafka.resources**. That way, all CPU resources are reserved upfront and are always available. This configuration allows Cruise Control to properly evaluate the CPU utilization when preparing the rebalance proposals based on CPU goals. In cases where you cannot set CPU requests equal to CPU limits in **Kafka.spec.kafka.resources**, you can set the CPU capacity manually for the same accuracy.

Example Cruise Control brokerCapacity configuration using bibyte units

```
apiVersion: kafka.strimzi.io/v1beta2
kind: Kafka
metadata:
  name: my-cluster
spec:
  # ...
  cruiseControl:
    # ...
    brokerCapacity:
      cpu: "2"
      inboundNetwork: 10000KiB/s
      outboundNetwork: 10000KiB/s
    # ...
```

53.5. CAPACITY OVERRIDES

Brokers might be running on nodes with heterogeneous network or CPU resources. If that's the case, specify **overrides** that set the network capacity and CPU limits for each broker. The overrides ensure an accurate rebalance between the brokers. Override capacity limits can be set for the following broker resources:

- **cpu** - CPU resource in millicores or CPU cores (Default: 1)
- **inboundNetwork** - Inbound network throughput in byte units per second (Default: 10000KiB/s)
- **outboundNetwork** - Outbound network throughput in byte units per second (Default: 10000KiB/s)

An example of Cruise Control capacity overrides configuration using bibyte units

```
apiVersion: kafka.strimzi.io/v1beta2
kind: Kafka
metadata:
  name: my-cluster
spec:
  # ...
  cruiseControl:
    # ...
    brokerCapacity:
      cpu: "1"
      inboundNetwork: 10000KiB/s
```

```
    outboundNetwork: 10000KiB/s
    overrides:
    - brokers: [0]
      cpu: "2.755"
      inboundNetwork: 20000KiB/s
      outboundNetwork: 20000KiB/s
    - brokers: [1, 2]
      cpu: 3000m
      inboundNetwork: 30000KiB/s
      outboundNetwork: 30000KiB/s
```

CPU capacity is determined using configuration values in the following order of precedence, with the highest priority first:

1. **Kafka.spec.cruiseControl.brokerCapacity.overrides.cpu** that define custom CPU capacity limits for individual brokers
2. **Kafka.cruiseControl.brokerCapacity.cpu** that defines custom CPU capacity limits for all brokers in the kafka cluster
3. **Kafka.spec.kafka.resources.requests.cpu** that defines the CPU resources that are reserved for each broker in the Kafka cluster.
4. **Kafka.spec.kafka.resources.limits.cpu** that defines the maximum CPU resources that can be consumed by each broker in the Kafka cluster.

This order of precedence is the sequence in which different configuration values are considered when determining the actual capacity limit for a Kafka broker. For example, broker-specific overrides take precedence over capacity limits for all brokers. If none of the CPU capacity configurations are specified, the default CPU capacity for a Kafka broker is set to 1 CPU core.

For more information, refer to the BrokerCapacity schema reference.

53.6. LOGGING

Cruise Control has its own configurable logger:

- **rootLogger.level**

Cruise Control uses the Apache **log4j2** logger implementation.

Use the **logging** property to configure loggers and logger levels.

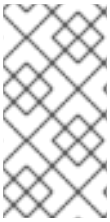
You can set the log levels by specifying the logger and level directly (inline) or use a custom (external) ConfigMap. If a ConfigMap is used, you set **logging.valueFrom.configMapKeyRef.name** property to the name of the ConfigMap containing the external logging configuration. Inside the ConfigMap, the logging configuration is described using **log4j.properties**. Both **logging.valueFrom.configMapKeyRef.name** and **logging.valueFrom.configMapKeyRef.key** properties are mandatory. A ConfigMap using the exact logging configuration specified is created with the custom resource when the Cluster Operator is running, then recreated after each reconciliation. If you do not specify a custom ConfigMap, default logging settings are used. If a specific logger value is not set, upper-level logger settings are inherited for that logger.

Here we see examples of **inline** and **external** logging. The **inline** logging specifies the root logger level. You can also set log levels for specific classes or loggers by adding them to the loggers property.

Inline logging

```
apiVersion: kafka.strimzi.io/v1beta2
kind: Kafka
# ...
spec:
  cruiseControl:
    # ...
    logging:
      type: inline
      loggers:
        rootLogger.level: INFO
        logger.exec.name: com.linkedin.kafka.cruisecontrol.executor.Executor 1
        logger.exec.level: TRACE 2
        logger.go.name: com.linkedin.kafka.cruisecontrol.analyzer.GoalOptimizer 3
        logger.go.level: DEBUG 4
    # ...
```

1. Creates a logger for the Cruise Control **Executor** class.
2. Sets the logging level for the **Executor** class.
3. Creates a logger for the Cruise Control **GoalOptimizer** class.
4. Sets the logging level for the **GoalOptimizer** class.



NOTE

When investigating an issue with Cruise Control, it's usually sufficient to change the **rootLogger** to **DEBUG** to get more detailed logs. However, keep in mind that setting the log level to **DEBUG** may result in a large amount of log output and may have performance implications.

External logging

```
apiVersion: kafka.strimzi.io/v1beta2
kind: Kafka
# ...
spec:
  cruiseControl:
    # ...
    logging:
      type: external
      valueFrom:
        configMapKeyRef:
          name: customConfigMap
          key: cruise-control-log4j.properties
    # ...
```

Garbage collector (GC)

Garbage collector logging can also be enabled (or disabled) using the **jvmOptions** property.

53.7. CRUISECONTROLSPEC SCHEMA PROPERTIES

Property	Description
image string	The docker image for the pods.
tlsSidecar **TlsSidecar**	**The tlsSidecar property has been deprecated.** TLS sidecar configuration.
resources ResourceRequirements	CPU and memory resources to reserve for the Cruise Control container. For more information, see the external documentation for core/v1 resourcerequirements.
livenessProbe **Probe**	Pod liveness checking for the Cruise Control container.
readinessProbe **Probe**	Pod readiness checking for the Cruise Control container.
jvmOptions **JvmOptions**	JVM Options for the Cruise Control container.
logging **InlineLogging**, **ExternalLogging**	Logging configuration (Log4j 2) for Cruise Control. The type depends on the value of the **logging.type** property within the given object, which must be one of [inline, external].
template **CruiseControlTemplate**	Template to specify how Cruise Control resources, **Deployments** and **Pods**, are generated.
brokerCapacity **BrokerCapacity**	The Cruise Control **brokerCapacity** configuration.

Property	Description
config map	The Cruise Control configuration. For a full list of configuration options refer to https://github.com/linkedin/cruise-control/wiki/Configurations. Note that properties with the following prefixes cannot be set: bootstrap.servers, client.id, zookeeper., network., security., failed.brokers.zk.path,webserver.http., webserver.api.urlprefix, webserver.session.path, webserver.accesslog., two.step., request.reason.required,metric.reporter.sampler.bootstrap.servers, capacity.config.file, self.healing., ssl., kafka.broker.failure.detection.enable, topic.config.provider.class (with the exception of: ssl.cipher.suites, ssl.protocol, ssl.enabled.protocols, webserver.http.cors.enabled, webserver.http.cors.origin, webserver.http.cors.exposeheaders, webserver.security.enable, webserver.ssl.enable).
metricsConfig **JmxPrometheusExporterMetrics**	Metrics configuration. The type depends on the value of the **metricsConfig.type** property within the given object, which must be one of [jmxPrometheusExporter].

CHAPTER 54. CRUISECONTROLTEMPLATE SCHEMA REFERENCE

Used in: **CruiseControlSpec**

Property	Description
deployment	Template for Cruise Control **Deployment**.
DeploymentTemplate	
pod	Template for Cruise Control **Pods**.
PodTemplate	
apiService	Template for Cruise Control API **Service**.
InternalServiceTemplate	
podDisruptionBudget	Template for Cruise Control **PodDisruptionBudget**.
PodDisruptionBudgetTemplate	
cruiseControlContainer	Template for the Cruise Control container.
ContainerTemplate	
tlsSidecarContainer	**The tlsSidecarContainer property has been deprecated.** Template for the Cruise Control TLS sidecar container.
ContainerTemplate	
serviceAccount	Template for the Cruise Control service account.
ResourceTemplate	

CHAPTER 55. BROKERCAPACITY SCHEMA REFERENCE

Used in: **CruiseControlSpec**

Property	Description
disk string	**The disk property has been deprecated.** The Cruise Control disk capacity setting has been deprecated, is ignored, and will be removed in the future Broker capacity for disk in bytes. Use a number value with either standard OpenShift byte units (K, M, G, or T), their bibyte (power of two) equivalents (Ki, Mi, Gi, or Ti), or a byte value with or without E notation. For example, 100000M, 100000Mi, 104857600000, or 1e+11.
cpuUtilization integer	**The cpuUtilization property has been deprecated.** The Cruise Control CPU capacity setting has been deprecated, is ignored, and will be removed in the future Broker capacity for CPU resource utilization as a percentage (0 - 100).
cpu string	Broker capacity for CPU resource in cores or millicores. For example, 1, 1.500, 1500m. For more information on valid CPU resource units see https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/#meaning-of-cpu.
inboundNetwork string	Broker capacity for inbound network throughput in bytes per second. Use an integer value with standard OpenShift byte units (K, M, G) or their bibyte (power of two) equivalents (Ki, Mi, Gi) per second. For example, 10000KiB/s.
outboundNetwork string	Broker capacity for outbound network throughput in bytes per second. Use an integer value with standard OpenShift byte units (K, M, G) or their bibyte (power of two) equivalents (Ki, Mi, Gi) per second. For example, 10000KiB/s.
overrides **BrokerCapacityOverride** array	Overrides for individual brokers. The **overrides** property lets you specify a different capacity configuration for different brokers.

CHAPTER 56. BROKERCAPACITYOVERRIDE SCHEMA REFERENCE

Used in: **BrokerCapacity**

Property	Description
brokers integer array	List of Kafka brokers (broker identifiers).
cpu string	Broker capacity for CPU resource in cores or millicores. For example, 1, 1.500, 1500m. For more information on valid CPU resource units see https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/#meaning-of-cpu.
inboundNetwork string	Broker capacity for inbound network throughput in bytes per second. Use an integer value with standard OpenShift byte units (K, M, G) or their bibyte (power of two) equivalents (Ki, Mi, Gi) per second. For example, 10000KiB/s.
outboundNetwork string	Broker capacity for outbound network throughput in bytes per second. Use an integer value with standard OpenShift byte units (K, M, G) or their bibyte (power of two) equivalents (Ki, Mi, Gi) per second. For example, 10000KiB/s.

CHAPTER 57. JMXTRANSSPEC SCHEMA REFERENCE

The type JmxTransSpec has been deprecated.

Used in: **KafkaSpec**

Property	Description
image string	The image to use for the JmxTrans.
outputDefinitions **JmxTransOutputDefinitionTemplate** array	Defines the output hosts that will be referenced later on. For more information on these properties see, **JmxTransOutputDefinitionTemplate** schema reference.
logLevel string	Sets the logging level of the JmxTrans deployment.For more information see, JmxTrans Logging Level.
kafkaQueries **JmxTransQueryTemplate** array	Queries to send to the Kafka brokers to define what data should be read from each broker. For more information on these properties see, **JmxTransQueryTemplate** schema reference.
resources ResourceRequirements	CPU and memory resources to reserve. For more information, see the external documentation for core/v1 resourcerequirements.
template **JmxTransTemplate**	Template for JmxTrans resources.

CHAPTER 58. JMXTRANSOUTPUTDEFINITIONTEMPLATE SCHEMA REFERENCE

Used in: **JmxTransSpec**

Property	Description
outputType string	Template for setting the format of the data that will be pushed.For more information see JmxTrans OutputWriters.
host string	The DNS/hostname of the remote host that the data is pushed to.
port integer	The port of the remote host that the data is pushed to.
flushDelayInSeconds integer	How many seconds the JmxTrans waits before pushing a new set of data out.
typeNames string array	Template for filtering data to be included in response to a wildcard query. For more information see JmxTrans queries.
name string	Template for setting the name of the output definition. This is used to identify where to send the results of queries should be sent.

CHAPTER 59. JMXTRANSQUERYTEMPLATE SCHEMA REFERENCE

Used in: **JmxTransSpec**

Property	Description
targetMBean string	If using wildcards instead of a specific MBean then the data is gathered from multiple MBeans. Otherwise if specifying an MBean then data is gathered from that specified MBean.
attributes string array	Determine which attributes of the targeted MBean should be included.
outputs string array	List of the names of output definitions specified in the spec.kafka.jmxTrans.outputDefinitions that have defined where JMX metrics are pushed to, and in which data format.

CHAPTER 60. JMXTRANSTEMPLATE SCHEMA REFERENCE

Used in: **JmxTransSpec**

Property	Description
deployment	Template for JmxTrans **Deployment**.
DeploymentTemplate	
pod	Template for JmxTrans **Pods**.
PodTemplate	
container	Template for JmxTrans container.
ContainerTemplate	
serviceAccount	Template for the JmxTrans service account.
ResourceTemplate	

CHAPTER 61. KAFKAEXPORTERSPEC SCHEMA REFERENCE

Used in: **KafkaSpec**

Property	Description
image	The docker image for the pods.
string	
groupRegex	Regular expression to specify which consumer groups to collect. Default value is **.***.
string	
topicRegex	Regular expression to specify which topics to collect. Default value is **.***.
string	
groupExcludeRegex	Regular expression to specify which consumer groups to exclude.
string	
topicExcludeRegex	Regular expression to specify which topics to exclude.
string	
resources	CPU and memory resources to reserve. For more information, see the external documentation for core/v1 resourcerequirements.
ResourceRequirements	
logging	Only log messages with the given severity or above. Valid levels: [**info**, **debug**, **trace**]. Default log level is **info**.
string	
enableSaramaLogging	Enable Sarama logging, a Go client library used by the Kafka Exporter.
boolean	
template	Customization of deployment templates and pods.
KafkaExporterTemplate	
livenessProbe	Pod liveness check.
Probe	
readinessProbe	Pod readiness check.

Property	Description
Probe	

CHAPTER 62. KAFKAEXPORTERTEMPLATE SCHEMA REFERENCE

Used in: **KafkaExporterSpec**

Property	Description
deployment	Template for Kafka Exporter **Deployment**.
DeploymentTemplate	
pod	Template for Kafka Exporter **Pods**.
PodTemplate	
service	**The service property has been deprecated.** The Kafka Exporter service has been removed. Template for Kafka Exporter **Service**.
ResourceTemplate	
container	Template for the Kafka Exporter container.
ContainerTemplate	
serviceAccount	Template for the Kafka Exporter service account.
ResourceTemplate	

CHAPTER 63. KAFKASTATUS SCHEMA REFERENCE

Used in: **Kafka**

Property	Description
conditions **Condition** array	List of status conditions.
observedGeneration integer	The generation of the CRD that was last reconciled by the operator.
listeners **ListenerStatus** array	Addresses of the internal and external listeners.
kafkaNodePools **UsedNodePoolStatus** array	List of the KafkaNodePools used by this Kafka cluster.
clusterId string	Kafka cluster Id.
operatorLastSuccessfulVersion string	The version of the AMQ Streams Cluster Operator which performed the last successful reconciliation.
kafkaVersion string	The version of Kafka currently deployed in the cluster.

CHAPTER 64. CONDITION SCHEMA REFERENCE

Used in: **KafkaBridgeStatus**, **KafkaConnectorStatus**, **KafkaConnectStatus**, **KafkaMirrorMaker2Status**, **KafkaMirrorMakerStatus**, **KafkaNodePoolStatus**, **KafkaRebalanceStatus**, **KafkaStatus**, **KafkaTopicStatus**, **KafkaUserStatus**, **StrimziPodSetStatus**

Property	Description
type	The unique identifier of a condition, used to distinguish between other conditions in the resource.
string	
status	The status of the condition, either True, False or Unknown.
string	
lastTransitionTime	Last time the condition of a type changed from one status to another. The required format is 'yyyy-MM-ddTHH:mm:ssZ', in the UTC time zone.
string	
reason	The reason for the condition's last transition (a single word in CamelCase).
string	
message	Human-readable message indicating details about the condition's last transition.
string	

CHAPTER 65. LISTENERSTATUS SCHEMA REFERENCE

Used in: **KafkaStatus**

Property	Description
type string	**The type property has been deprecated.** The **type** property is not used anymore. Use the **name** property with the same value. The name of the listener.
name string	The name of the listener.
addresses **ListenerAddress** array	A list of the addresses for this listener.
bootstrapServers string	A comma-separated list of **host:port** pairs for connecting to the Kafka cluster using this listener.
certificates string array	A list of TLS certificates which can be used to verify the identity of the server when connecting to the given listener. Set only for **tls** and **external** listeners.

CHAPTER 66. LISTENERADDRESS SCHEMA REFERENCE

Used in: **ListenerStatus**

Property	Description
host	The DNS name or IP address of the Kafka bootstrap service.
string	
port	The port of the Kafka bootstrap service.
integer	

CHAPTER 67. USEDNODEPOOLSTATUS SCHEMA REFERENCE

Used in: **KafkaStatus**

Property	Description
name string	The name of the KafkaNodePool used by this Kafka resource.

CHAPTER 68. KAFKACONNECT SCHEMA REFERENCE

Property	Description
spec	The specification of the Kafka Connect cluster.
KafkaConnectSpec	
status	The status of the Kafka Connect cluster.
KafkaConnectStatus	

CHAPTER 69. KAFKACONNECTSPEC SCHEMA REFERENCE

Used in: **KafkaConnect**

Full list of **KafkaConnectSpec** schema properties

Configures a Kafka Connect cluster.

69.1. CONFIG

Use the **config** properties to configure Kafka Connect options as keys.

The values can be one of the following JSON types:

- String
- Number
- Boolean

Certain options have default values:

- **group.id** with default value **connect-cluster**
- **offset.storage.topic** with default value **connect-cluster-offsets**
- **config.storage.topic** with default value **connect-cluster-configs**
- **status.storage.topic** with default value **connect-cluster-status**
- **key.converter** with default value **org.apache.kafka.connect.json.JsonConverter**
- **value.converter** with default value **org.apache.kafka.connect.json.JsonConverter**

These options are automatically configured in case they are not present in the **KafkaConnect.spec.config** properties.

Exceptions

You can specify and configure the options listed in the Apache Kafka documentation.

However, AMQ Streams takes care of configuring and managing options related to the following, which cannot be changed:

- Kafka cluster bootstrap address
- Security (encryption, authentication, and authorization)
- Listener and REST interface configuration
- Plugin path configuration

Properties with the following prefixes cannot be set:

- **bootstrap.servers**
- **consumer.interceptor.classes**

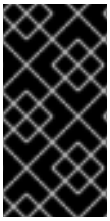
- **listeners.**
- **plugin.path**
- **producer.interceptor.classes**
- **rest.**
- **sasl.**
- **security.**
- **ssl.**

If the **config** property contains an option that cannot be changed, it is disregarded, and a warning message is logged to the Cluster Operator log file. All other supported options are forwarded to Kafka Connect, including the following exceptions to the options configured by AMQ Streams:

- Any **ssl** configuration for supported TLS versions and cipher suites

Example Kafka Connect configuration

```
apiVersion: kafka.strimzi.io/v1beta2
kind: KafkaConnect
metadata:
  name: my-connect
spec:
  # ...
  config:
    group.id: my-connect-cluster
    offset.storage.topic: my-connect-cluster-offsets
    config.storage.topic: my-connect-cluster-configs
    status.storage.topic: my-connect-cluster-status
    key.converter: org.apache.kafka.connect.json.JsonConverter
    value.converter: org.apache.kafka.connect.json.JsonConverter
    key.converter.schemas.enable: true
    value.converter.schemas.enable: true
    config.storage.replication.factor: 3
    offset.storage.replication.factor: 3
    status.storage.replication.factor: 3
  # ...
```



IMPORTANT

The Cluster Operator does not validate keys or values in the **config** object provided. If an invalid configuration is provided, the Kafka Connect cluster might not start or might become unstable. In this case, fix the configuration so that the Cluster Operator can roll out the new configuration to all Kafka Connect nodes.

69.2. LOGGING

Kafka Connect has its own configurable loggers:

- **connect.root.logger.level**

- **log4j.logger.org.reflections**

Further loggers are added depending on the Kafka Connect plugins running.

Use a curl request to get a complete list of Kafka Connect loggers running from any Kafka broker pod:

```
curl -s http://<connect-cluster-name>-connect-api:8083/admin/loggers/
```

Kafka Connect uses the Apache **log4j** logger implementation.

Use the **logging** property to configure loggers and logger levels.

You can set the log levels by specifying the logger and level directly (inline) or use a custom (external) ConfigMap. If a ConfigMap is used, you set **logging.valueFrom.configMapKeyRef.name** property to the name of the ConfigMap containing the external logging configuration. Inside the ConfigMap, the logging configuration is described using **log4j.properties**. Both **logging.valueFrom.configMapKeyRef.name** and **logging.valueFrom.configMapKeyRef.key** properties are mandatory. A ConfigMap using the exact logging configuration specified is created with the custom resource when the Cluster Operator is running, then recreated after each reconciliation. If you do not specify a custom ConfigMap, default logging settings are used. If a specific logger value is not set, upper-level logger settings are inherited for that logger. For more information about log levels, see Apache logging services.

Here we see examples of **inline** and **external** logging. The **inline** logging specifies the root logger level. You can also set log levels for specific classes or loggers by adding them to the loggers property.

Inline logging

```
apiVersion: kafka.strimzi.io/v1beta2
kind: KafkaConnect
spec:
  # ...
  logging:
    type: inline
    loggers:
      connect.root.logger.level: INFO
      log4j.logger.org.apache.kafka.connect.runtime.WorkerSourceTask: TRACE
      log4j.logger.org.apache.kafka.connect.runtime.WorkerSinkTask: DEBUG
  # ...
```

NOTE

Setting a log level to **DEBUG** may result in a large amount of log output and may have performance implications.

External logging

```
apiVersion: kafka.strimzi.io/v1beta2
kind: KafkaConnect
spec:
  # ...
  logging:
    type: external
    valueFrom:
```

```
    configMapKeyRef:
      name: customConfigMap
      key: connect-logging.log4j
# ...
```

Any available loggers that are not configured have their level set to **OFF**.

If Kafka Connect was deployed using the Cluster Operator, changes to Kafka Connect logging levels are applied dynamically.

If you use external logging, a rolling update is triggered when logging appenders are changed.

Garbage collector (GC)

Garbage collector logging can also be enabled (or disabled) using the **jvmOptions** property.

69.3. KAFKACONNECTSPEC SCHEMA PROPERTIES

Property	Description
version string	The Kafka Connect version. Defaults to 3.6.0. Consult the user documentation to understand the process required to upgrade or downgrade the version.
replicas integer	The number of pods in the Kafka Connect group. Defaults to **3**.
image string	The docker image for the pods.
bootstrapServers string	Bootstrap servers to connect to. This should be given as a comma separated list of *<hostname>*:_<port>_ pairs.
tls **ClientTls**	TLS configuration.
authentication **KafkaClientAuthenticationTls**, **KafkaClientAuthenticationScramSha256**, **KafkaClientAuthenticationScramSha512**, **KafkaClientAuthenticationPlain**, **KafkaClientAuthenticationOAuth**	Authentication configuration for Kafka Connect. The type depends on the value of the **authentication.type** property within the given object, which must be one of [tls, scram-sha-256, scram-sha-512, plain, oauth].

Property	Description
config map	The Kafka Connect configuration. Properties with the following prefixes cannot be set: ssl., sasl., security., listeners, plugin.path, rest., bootstrap.servers, consumer.interceptor.classes, producer.interceptor.classes (with the exception of: ssl.endpoint.identification.algorithm, ssl.cipher.suites, ssl.protocol, ssl.enabled.protocols).
resources ResourceRequirements	The maximum limits for CPU and memory resources and the requested initial resources. For more information, see the external documentation for core/v1 resourcerequirements.
livenessProbe **Probe**	Pod liveness checking.
readinessProbe **Probe**	Pod readiness checking.
jvmOptions **JvmOptions**	JVM Options for pods.
jmxOptions **KafkaJmxOptions**	JMX Options.
logging **InlineLogging**, **ExternalLogging**	Logging configuration for Kafka Connect. The type depends on the value of the **logging.type** property within the given object, which must be one of [inline, external].
clientRackInitImage string	The image of the init container used for initializing the **client.rack**.
rack **Rack**	Configuration of the node label which will be used as the **client.rack** consumer configuration.

Property	Description
tracing **JaegerTracing**, **OpenTelemetryTracing**	The configuration of tracing in Kafka Connect. The type depends on the value of the **tracing.type** property within the given object, which must be one of [jaeger, opentelemetry].
template **KafkaConnectTemplate**	Template for Kafka Connect and Kafka Mirror Maker 2 resources. The template allows users to specify how the **Deployment**, **Pods** and **Service** are generated.
externalConfiguration **ExternalConfiguration**	Pass data from Secrets or ConfigMaps to the Kafka Connect pods and use them to configure connectors.
build **Build**	Configures how the Connect container image should be built. Optional.
metricsConfig **JmxPrometheusExporterMetrics**	Metrics configuration. The type depends on the value of the **metricsConfig.type** property within the given object, which must be one of [jmxPrometheusExporter].

CHAPTER 70. CLIENTTLS SCHEMA REFERENCE

Used in: **KafkaBridgeSpec**, **KafkaConnectSpec**, **KafkaMirrorMaker2ClusterSpec**, **KafkaMirrorMakerConsumerSpec**, **KafkaMirrorMakerProducerSpec**

Full list of **ClientTls** schema properties

Configures TLS trusted certificates for connecting KafkaConnect, KafkaBridge, KafkaMirror, KafkaMirrorMaker2 to the cluster.

70.1. TRUSTEDCERTIFICATES

Provide a list of secrets using the **trustedCertificates** property.

70.2. CLIENTTLS SCHEMA PROPERTIES

Property	Description
trustedCertificates	Trusted certificates for TLS connection.
CertSecretSource array	

CHAPTER 71. KAFKACLIENTAUTHENTICATIONTLS SCHEMA REFERENCE

Used in: **KafkaBridgeSpec**, **KafkaConnectSpec**, **KafkaMirrorMaker2ClusterSpec**, **KafkaMirrorMakerConsumerSpec**, **KafkaMirrorMakerProducerSpec**

Full list of **KafkaClientAuthenticationTls** schema properties

To configure mTLS authentication, set the **type** property to the value **tls**. mTLS uses a TLS certificate to authenticate.

71.1. CERTIFICATEANDKEY

The certificate is specified in the **certificateAndKey** property and is always loaded from an OpenShift secret. In the secret, the certificate must be stored in X509 format under two different keys: public and private.

You can use the secrets created by the User Operator, or you can create your own TLS certificate file, with the keys used for authentication, then create a **Secret** from the file:

```
oc create secret generic MY-SECRET \
--from-file=MY-PUBLIC-TLS-CERTIFICATE-FILE.crt \
--from-file=MY-PRIVATE.key
```

NOTE

mTLS authentication can only be used with TLS connections.

Example mTLS configuration

```
authentication:
  type: tls
  certificateAndKey:
    secretName: my-secret
    certificate: my-public-tls-certificate-file.crt
    key: private.key
```

71.2. KAFKACLIENTAUTHENTICATIONTLS SCHEMA PROPERTIES

The **type** property is a discriminator that distinguishes use of the **KafkaClientAuthenticationTls** type from **KafkaClientAuthenticationScramSha256**, **KafkaClientAuthenticationScramSha512**, **KafkaClientAuthenticationPlain**, **KafkaClientAuthenticationOAuth**. It must have the value **tls** for the type **KafkaClientAuthenticationTls**.

Property	Description
certificateAndKey **CertAndKeySecretSource**	Reference to the **Secret** which holds the certificate and private key pair.
type	Must be **tls**.

Property	Description
string	

CHAPTER 72. KAFKACLIENTAUTHENTICATIONSCRAMSHA256 SCHEMA REFERENCE

Used in: **KafkaBridgeSpec**, **KafkaConnectSpec**, **KafkaMirrorMaker2ClusterSpec**, **KafkaMirrorMakerConsumerSpec**, **KafkaMirrorMakerProducerSpec**

Full list of **KafkaClientAuthenticationScramSha256** schema properties

To configure SASL-based SCRAM-SHA-256 authentication, set the **type** property to **scram-sha-256**. The SCRAM-SHA-256 authentication mechanism requires a username and password.

72.1. USERNAME

Specify the username in the **username** property.

72.2. PASSWORDSECRET

In the **passwordSecret** property, specify a link to a **Secret** containing the password.

You can use the secrets created by the User Operator.

If required, you can create a text file that contains the password, in cleartext, to use for authentication:

```
echo -n PASSWORD > MY-PASSWORD.txt
```

You can then create a **Secret** from the text file, setting your own field name (key) for the password:

```
oc create secret generic MY-CONNECT-SECRET-NAME --from-file=MY-PASSWORD-FIELD-NAME=./MY-PASSWORD.txt
```

Example Secret for SCRAM-SHA-256 client authentication for Kafka Connect

```
apiVersion: v1
kind: Secret
metadata:
  name: my-connect-secret-name
type: Opaque
data:
  my-connect-password-field: LFTIyFRFlMmU2N2Tm
```

The **secretName** property contains the name of the **Secret**, and the **password** property contains the name of the key under which the password is stored inside the **Secret**.

IMPORTANT

Do not specify the actual password in the **password** property.

Example SASL-based SCRAM-SHA-256 client authentication configuration for Kafka Connect

```
authentication:
```

```
type: scram-sha-256
username: my-connect-username
passwordSecret:
  secretName: my-connect-secret-name
  password: my-connect-password-field
```

72.3. KAFKACLIENTAUTHENTICATIONSCRAMSHA256 SCHEMA PROPERTIES

Property	Description
passwordSecret	Reference to the **Secret** which holds the password.
PasswordSecretSource	
type	Must be **scram-sha-256**.
string	
username	Username used for the authentication.
string	

CHAPTER 73. PASSWORDSECRETSOURCE SCHEMA REFERENCE

Used in: **KafkaClientAuthenticationOAuth**, **KafkaClientAuthenticationPlain**, **KafkaClientAuthenticationScramSha256**, **KafkaClientAuthenticationScramSha512**

Property	Description
password	The name of the key in the Secret under which the password is stored.
string	
secretName	The name of the Secret containing the password.
string	

CHAPTER 74. KAFKACLIENTAUTHENTICATIONSCRAMSHA512 SCHEMA REFERENCE

Used in: **KafkaBridgeSpec**, **KafkaConnectSpec**, **KafkaMirrorMaker2ClusterSpec**, **KafkaMirrorMakerConsumerSpec**, **KafkaMirrorMakerProducerSpec**

Full list of **KafkaClientAuthenticationScramSha512** schema properties

To configure SASL-based SCRAM-SHA-512 authentication, set the **type** property to **scram-sha-512**. The SCRAM-SHA-512 authentication mechanism requires a username and password.

74.1. USERNAME

Specify the username in the **username** property.

74.2. PASSWORDSECRET

In the **passwordSecret** property, specify a link to a **Secret** containing the password.

You can use the secrets created by the User Operator.

If required, you can create a text file that contains the password, in cleartext, to use for authentication:

```
echo -n PASSWORD > MY-PASSWORD.txt
```

You can then create a **Secret** from the text file, setting your own field name (key) for the password:

```
oc create secret generic MY-CONNECT-SECRET-NAME --from-file=MY-PASSWORD-FIELD-NAME=./MY-PASSWORD.txt
```

Example Secret for SCRAM-SHA-512 client authentication for Kafka Connect

```
apiVersion: v1
kind: Secret
metadata:
  name: my-connect-secret-name
type: Opaque
data:
  my-connect-password-field: LFTIyFRFlMmU2N2Tm
```

The **secretName** property contains the name of the **Secret**, and the **password** property contains the name of the key under which the password is stored inside the **Secret**.

IMPORTANT

Do not specify the actual password in the **password** property.

Example SASL-based SCRAM-SHA-512 client authentication configuration for Kafka Connect

```
authentication:
```

```
type: scram-sha-512
username: my-connect-username
passwordSecret:
  secretName: my-connect-secret-name
  password: my-connect-password-field
```

74.3. KAFKACLIENTAUTHENTICATIONSCRAMSHA512 SCHEMA PROPERTIES

Property	Description
passwordSecret	Reference to the **Secret** which holds the password.
PasswordSecretSource	
type	Must be **scram-sha-512**.
string	
username	Username used for the authentication.
string	

CHAPTER 75. KAFKACLIENTAUTHENTICATIONPLAIN SCHEMA REFERENCE

Used in: **KafkaBridgeSpec**, **KafkaConnectSpec**, **KafkaMirrorMaker2ClusterSpec**, **KafkaMirrorMakerConsumerSpec**, **KafkaMirrorMakerProducerSpec**

Full list of **KafkaClientAuthenticationPlain** schema properties

To configure SASL-based PLAIN authentication, set the **type** property to **plain**. SASL PLAIN authentication mechanism requires a username and password.

WARNING

The SASL PLAIN mechanism will transfer the username and password across the network in cleartext. Only use SASL PLAIN authentication if TLS encryption is enabled.

75.1. USERNAME

Specify the username in the **username** property.

75.2. PASSWORDSECRET

In the **passwordSecret** property, specify a link to a **Secret** containing the password.

You can use the secrets created by the User Operator.

If required, create a text file that contains the password, in cleartext, to use for authentication:

```
echo -n PASSWORD > MY-PASSWORD.txt
```

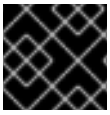
You can then create a **Secret** from the text file, setting your own field name (key) for the password:

```
oc create secret generic MY-CONNECT-SECRET-NAME --from-file=MY-PASSWORD-FIELD-NAME=./MY-PASSWORD.txt
```

Example Secret for PLAIN client authentication for Kafka Connect

```
apiVersion: v1
kind: Secret
metadata:
  name: my-connect-secret-name
type: Opaque
data:
  my-password-field-name: LFTIyFRFlMmU2N2Tm
```

The **secretName** property contains the name of the **Secret** and the **password** property contains the name of the key under which the password is stored inside the **Secret**.



IMPORTANT

Do not specify the actual password in the **password** property.

An example SASL based PLAIN client authentication configuration

```
authentication:
  type: plain
  username: my-connect-username
  passwordSecret:
    secretName: my-connect-secret-name
    password: my-password-field-name
```

75.3. KAFKACLIENTAUTHENTICATIONPLAIN SCHEMA PROPERTIES

The **type** property is a discriminator that distinguishes use of the **KafkaClientAuthenticationPlain** type from **KafkaClientAuthenticationTls**, **KafkaClientAuthenticationScramSha256**, **KafkaClientAuthenticationScramSha512**, **KafkaClientAuthenticationOAuth**. It must have the value **plain** for the type **KafkaClientAuthenticationPlain**.

Property	Description
passwordSecret	Reference to the **Secret** which holds the password.
PasswordSecretSource	
type	Must be **plain**.
string	
username	Username used for the authentication.
string	

CHAPTER 76. KAFKACLIENTAUTHENTICATIONOAUTH SCHEMA REFERENCE

Used in: **KafkaBridgeSpec**, **KafkaConnectSpec**, **KafkaMirrorMaker2ClusterSpec**, **KafkaMirrorMakerConsumerSpec**, **KafkaMirrorMakerProducerSpec**

Full list of **KafkaClientAuthenticationOAuth** schema properties

To configure OAuth client authentication, set the **type** property to **oauth**.

OAuth authentication can be configured using one of the following options:

- Client ID and secret
- Client ID and refresh token
- Access token
- Username and password
- TLS

Client ID and secret

You can configure the address of your authorization server in the **tokenEndpointUri** property together with the client ID and client secret used in authentication. The OAuth client will connect to the OAuth server, authenticate using the client ID and secret and get an access token which it will use to authenticate with the Kafka broker. In the **clientSecret** property, specify a link to a **Secret** containing the client secret.

An example of OAuth client authentication using client ID and client secret

```
authentication:
  type: oauth
  tokenEndpointUri: https://sso.myproject.svc:8443/auth/realms/internal/protocol/openid-connect/token
  clientId: my-client-id
  clientSecret:
    secretName: my-client-oauth-secret
    key: client-secret
```

Optionally, **scope** and **audience** can be specified if needed.

Client ID and refresh token

You can configure the address of your OAuth server in the **tokenEndpointUri** property together with the OAuth client ID and refresh token. The OAuth client will connect to the OAuth server, authenticate using the client ID and refresh token and get an access token which it will use to authenticate with the Kafka broker. In the **refreshToken** property, specify a link to a **Secret** containing the refresh token.

An example of OAuth client authentication using client ID and refresh token

```
authentication:
  type: oauth
  tokenEndpointUri: https://sso.myproject.svc:8443/auth/realms/internal/protocol/openid-connect/token
  clientId: my-client-id
```

```
  refreshToken:
    secretName: my-refresh-token-secret
    key: refresh-token
```

Access token

You can configure the access token used for authentication with the Kafka broker directly. In this case, you do not specify the **tokenEndpointUri**. In the **accessToken** property, specify a link to a **Secret** containing the access token.

An example of OAuth client authentication using only an access token

```
authentication:
  type: oauth
  accessToken:
    secretName: my-access-token-secret
    key: access-token
```

Username and password

OAuth username and password configuration uses the OAuth *Resource Owner Password Grant* mechanism. The mechanism is deprecated, and is only supported to enable integration in environments where client credentials (ID and secret) cannot be used. You might need to use user accounts if your access management system does not support another approach or user accounts are required for authentication.

A typical approach is to create a special user account in your authorization server that represents your client application. You then give the account a long randomly generated password and a very limited set of permissions. For example, the account can only connect to your Kafka cluster, but is not allowed to use any other services or login to the user interface.

Consider using a refresh token mechanism first.

You can configure the address of your authorization server in the **tokenEndpointUri** property together with the client ID, username and the password used in authentication. The OAuth client will connect to the OAuth server, authenticate using the username, the password, the client ID, and optionally even the client secret to obtain an access token which it will use to authenticate with the Kafka broker.

In the **passwordSecret** property, specify a link to a **Secret** containing the password.

Normally, you also have to configure a **clientId** using a public OAuth client. If you are using a confidential OAuth client, you also have to configure a **clientSecret**.

An example of OAuth client authentication using username and a password with a public client

```
authentication:
  type: oauth
  tokenEndpointUri: https://sso.myproject.svc:8443/auth/realms/internal/protocol/openid-connect/token
  username: my-username
  passwordSecret:
    secretName: my-password-secret-name
    password: my-password-field-name
  clientId: my-public-client-id
```

An example of OAuth client authentication using a username and a password with a confidential client

```
authentication:
  type: oauth
  tokenEndpointUri: https://sso.myproject.svc:8443/auth/realms/internal/protocol/openid-connect/token
  username: my-username
  passwordSecret:
    secretName: my-password-secret-name
    password: my-password-field-name
  clientId: my-confidential-client-id
  clientSecret:
    secretName: my-confidential-client-oauth-secret
    key: client-secret
```

Optionally, **scope** and **audience** can be specified if needed.

TLS

Accessing the OAuth server using the HTTPS protocol does not require any additional configuration as long as the TLS certificates used by it are signed by a trusted certification authority and its hostname is listed in the certificate.

If your OAuth server is using certificates which are self-signed or are signed by a certification authority which is not trusted, you can configure a list of trusted certificates in the custom resource. The **tlsTrustedCertificates** property contains a list of secrets with key names under which the certificates are stored. The certificates must be stored in X509 format.

An example of TLS certificates provided

```
authentication:
  type: oauth
  tokenEndpointUri: https://sso.myproject.svc:8443/auth/realms/internal/protocol/openid-connect/token
  clientId: my-client-id
  refreshToken:
    secretName: my-refresh-token-secret
    key: refresh-token
  tlsTrustedCertificates:
    - secretName: oauth-server-ca
      certificate: tls.crt
```

The OAuth client will by default verify that the hostname of your OAuth server matches either the certificate subject or one of the alternative DNS names. If it is not required, you can disable the hostname verification.

An example of disabled TLS hostname verification

```
authentication:
  type: oauth
  tokenEndpointUri: https://sso.myproject.svc:8443/auth/realms/internal/protocol/openid-connect/token
  clientId: my-client-id
  refreshToken:
    secretName: my-refresh-token-secret
    key: refresh-token
  disableTlsHostnameVerification: true
```

76.1. KAFKACLIENTAUTHENTICATIONOAUTH SCHEMA PROPERTIES

The **type** property is a discriminator that distinguishes use of the **KafkaClientAuthenticationOAuth** type from KafkaClientAuthenticationTls, KafkaClientAuthenticationScramSha256, KafkaClientAuthenticationScramSha512, KafkaClientAuthenticationPlain. It must have the value **oauth** for the type **KafkaClientAuthenticationOAuth**.

Property	Description
accessToken GenericSecretSource	Link to OpenShift Secret containing the access token which was obtained from the authorization server.
accessTokenIsJwt boolean	Configure whether access token should be treated as JWT. This should be set to **false** if the authorization server returns opaque tokens. Defaults to **true**.
audience string	OAuth audience to use when authenticating against the authorization server. Some authorization servers require the audience to be explicitly set. The possible values depend on how the authorization server is configured. By default, **audience** is not specified when performing the token endpoint request.
clientId string	OAuth Client ID which the Kafka client can use to authenticate against the OAuth server and use the token endpoint URI.
clientSecret GenericSecretSource	Link to OpenShift Secret containing the OAuth client secret which the Kafka client can use to authenticate against the OAuth server and use the token endpoint URI.
connectTimeoutSeconds integer	The connect timeout in seconds when connecting to authorization server. If not set, the effective connect timeout is 60 seconds.
disableTlsHostnameVerification boolean	Enable or disable TLS hostname verification. Default value is **false**.
enableMetrics boolean	Enable or disable OAuth metrics. Default value is **false**.
httpRetries integer	The maximum number of retries to attempt if an initial HTTP request fails. If not set, the default is to not attempt any retries.

Property	Description
httpRetryPauseMs integer	The pause to take before retrying a failed HTTP request. If not set, the default is to not pause at all but to immediately repeat a request.
includeAcceptHeader boolean	Whether the Accept header should be set in requests to the authorization servers. The default value is **true**.
maxTokenExpirySeconds integer	Set or limit time-to-live of the access tokens to the specified number of seconds. This should be set if the authorization server returns opaque tokens.
passwordSecret **PasswordSecretSource**	Reference to the **Secret** which holds the password.
readTimeoutSeconds integer	The read timeout in seconds when connecting to authorization server. If not set, the effective read timeout is 60 seconds.
refreshToken **GenericSecretSource**	Link to OpenShift Secret containing the refresh token which can be used to obtain access token from the authorization server.
scope string	OAuth scope to use when authenticating against the authorization server. Some authorization servers require this to be set. The possible values depend on how authorization server is configured. By default **scope** is not specified when doing the token endpoint request.
tlsTrustedCertificates **CertSecretSource** array	Trusted certificates for TLS connection to the OAuth server.
tokenEndpointUri string	Authorization server token endpoint URI.
type string	Must be **oauth**.
username	Username used for the authentication.

Property	Description
string	

CHAPTER 77. JAEGERTRACING SCHEMA REFERENCE

The type JaegerTracing has been deprecated.

Used in: **KafkaBridgeSpec**, **KafkaConnectSpec**, **KafkaMirrorMaker2Spec**, **KafkaMirrorMakerSpec**

The **type** property is a discriminator that distinguishes use of the **JaegerTracing** type from **OpenTelemetryTracing**. It must have the value **jaeger** for the type **JaegerTracing**.

Property	Description
type	Must be **jaeger**.
string	

CHAPTER 78. OPENTELEMETRYTRACING SCHEMA REFERENCE

Used in: **KafkaBridgeSpec**, **KafkaConnectSpec**, **KafkaMirrorMaker2Spec**, **KafkaMirrorMakerSpec**

The **type** property is a discriminator that distinguishes use of the **OpenTelemetryTracing** type from **JaegerTracing**. It must have the value **opentelemetry** for the type **OpenTelemetryTracing**.

Property	Description
type string	Must be **opentelemetry**.

CHAPTER 79. KAFKACONNECTTEMPLATE SCHEMA REFERENCE

Used in: **KafkaConnectSpec**, **KafkaMirrorMaker2Spec**

Property	Description
deployment	Template for Kafka Connect **Deployment**.
DeploymentTemplate	
podSet	Template for Kafka Connect **StrimziPodSet** resource.
ResourceTemplate	
pod	Template for Kafka Connect **Pods**.
PodTemplate	
apiService	Template for Kafka Connect API **Service**.
InternalServiceTemplate	
headlessService	Template for Kafka Connect headless **Service**.
InternalServiceTemplate	
connectContainer	Template for the Kafka Connect container.
ContainerTemplate	
initContainer	Template for the Kafka init container.
ContainerTemplate	
podDisruptionBudget	Template for Kafka Connect **PodDisruptionBudget**.
PodDisruptionBudgetTemplate	
serviceAccount	Template for the Kafka Connect service account.
ResourceTemplate	
clusterRoleBinding	Template for the Kafka Connect ClusterRoleBinding.
ResourceTemplate	

Property	Description
buildPod **PodTemplate**	Template for Kafka Connect Build **Pods**. The build pod is used only on OpenShift.
buildContainer **ContainerTemplate**	Template for the Kafka Connect Build container. The build container is used only on OpenShift.
buildConfig **BuildConfigTemplate**	Template for the Kafka Connect BuildConfig used to build new container images. The BuildConfig is used only on OpenShift.
buildServiceAccount **ResourceTemplate**	Template for the Kafka Connect Build service account.
jmxSecret **ResourceTemplate**	Template for Secret of the Kafka Connect Cluster JMX authentication.

CHAPTER 80. BUILDCONFIGTEMPLATE SCHEMA REFERENCE

Used in: **KafkaConnectTemplate**

Property	Description
metadata **MetadataTemplate**	Metadata to apply to the **PodDisruptionBudgetTemplate** resource.
pullSecret string	Container Registry Secret with the credentials for pulling the base image.

CHAPTER 81. EXTERNALCONFIGURATION SCHEMA REFERENCE

Used in: **KafkaConnectSpec**, **KafkaMirrorMaker2Spec**

Full list of **ExternalConfiguration** schema properties

Configures external storage properties that define configuration options for Kafka Connect connectors.

You can mount ConfigMaps or Secrets into a Kafka Connect pod as environment variables or volumes. Volumes and environment variables are configured in the **externalConfiguration** property in **KafkaConnect.spec** or **KafkaMirrorMaker2.spec**.

When applied, the environment variables and volumes are available for use when developing your connectors.

For more information, see Loading configuration values from external sources .

81.1. EXTERNALCONFIGURATION SCHEMA PROPERTIES

Property	Description
env	Makes data from a Secret or ConfigMap available in the Kafka Connect pods as environment variables.
ExternalConfigurationEnv array	
volumes	Makes data from a Secret or ConfigMap available in the Kafka Connect pods as volumes.
ExternalConfigurationVolumeSource array	

CHAPTER 82. EXTERNALCONFIGURATIONENV SCHEMA REFERENCE

Used in: **ExternalConfiguration**

Property	Description
name	Name of the environment variable which will be passed to the Kafka Connect pods. The name of the environment variable cannot start with **KAFKA_** or **STRIMZI_**.
string	
valueFrom	Value of the environment variable which will be passed to the Kafka Connect pods. It can be passed either as a reference to Secret or ConfigMap field. The field has to specify exactly one Secret or ConfigMap.
ExternalConfigurationEnvVarSource	

CHAPTER 83. EXTERNALCONFIGURATIONENVVARSOURCE SCHEMA REFERENCE

Used in: **ExternalConfigurationEnv**

Property	Description
configMapKeyRef	Reference to a key in a ConfigMap. For more information, see the external documentation for core/v1 configmapkeyselector.
ConfigMapKeySelector	
secretKeyRef	Reference to a key in a Secret. For more information, see the external documentation for core/v1 secretkeyselector.
SecretKeySelector	

CHAPTER 84. EXTERNALCONFIGURATIONVOLUMESOURCE SCHEMA REFERENCE

Used in: **ExternalConfiguration**

Property	Description
configMap ConfigMapVolumeSource	Reference to a key in a ConfigMap. Exactly one Secret or ConfigMap has to be specified. For more information, see the external documentation for core/v1 configmapvolumesource.
name string	Name of the volume which will be added to the Kafka Connect pods.
secret SecretVolumeSource	Reference to a key in a Secret. Exactly one Secret or ConfigMap has to be specified. For more information, see the external documentation for core/v1 secretvolumesource.

CHAPTER 85. BUILD SCHEMA REFERENCE

Used in: **KafkaConnectSpec**

Full list of **Build** schema properties

Configures additional connectors for Kafka Connect deployments.

85.1. OUTPUT

To build new container images with additional connector plugins, AMQ Streams requires a container registry where the images can be pushed to, stored, and pulled from. AMQ Streams does not run its own container registry, so a registry must be provided. AMQ Streams supports private container registries as well as public registries such as Quay or Docker Hub. The container registry is configured in the **.spec.build.output** section of the **KafkaConnect** custom resource. The **output** configuration, which is required, supports two types: **docker** and **imagestream**.

Using Docker registry

To use a Docker registry, you have to specify the **type** as **docker**, and the **image** field with the full name of the new container image. The full name must include:

- The address of the registry
- Port number (if listening on a non-standard port)
- The tag of the new container image

Example valid container image names:

- **docker.io/my-org/my-image/my-tag**
- **quay.io/my-org/my-image/my-tag**
- **image-registry.image-registry.svc:5000/myproject/kafka-connect-build:latest**

Each Kafka Connect deployment must use a separate image, which can mean different tags at the most basic level.

If the registry requires authentication, use the **pushSecret** to set a name of the Secret with the registry credentials. For the Secret, use the **kubernetes.io/dockerconfigjson** type and a **.dockerconfigjson** file to contain the Docker credentials. For more information on pulling an image from a private registry, see Create a Secret based on existing Docker credentials .

Example output configuration

```
apiVersion: kafka.strimzi.io/v1beta2
kind: KafkaConnect
metadata:
  name: my-connect-cluster
spec:
  #...
  build:
    output:
      type: docker 1
```

```
    image: my-registry.io/my-org/my-connect-cluster:latest 2
    pushSecret: my-registry-credentials 3
  #...
```

1 (Required) Type of output used by AMQ Streams.

2 (Required) Full name of the image used, including the repository and tag.

3 (Optional) Name of the secret with the container registry credentials.

Using OpenShift ImageStream

Instead of Docker, you can use OpenShift ImageStream to store a new container image. The ImageStream has to be created manually before deploying Kafka Connect. To use ImageStream, set the **type** to **imagestream**, and use the **image** property to specify the name of the ImageStream and the tag used. For example, **my-connect-image-stream:latest**.

Example output configuration

```
apiVersion: kafka.strimzi.io/v1beta2
kind: KafkaConnect
metadata:
  name: my-connect-cluster
spec:
  #...
  build:
    output:
      type: imagestream 1
      image: my-connect-build:latest 2
  #...
```

1 (Required) Type of output used by AMQ Streams.

2 (Required) Name of the ImageStream and tag.

85.2. PLUGINS

Connector plugins are a set of files that define the implementation required to connect to certain types of external system. The connector plugins required for a container image must be configured using the **.spec.build.plugins** property of the **KafkaConnect** custom resource. Each connector plugin must have a name which is unique within the Kafka Connect deployment. Additionally, the plugin artifacts must be listed. These artifacts are downloaded by AMQ Streams, added to the new container image, and used in the Kafka Connect deployment. The connector plugin artifacts can also include additional components, such as (de)serializers. Each connector plugin is downloaded into a separate directory so that the different connectors and their dependencies are properly *sandboxed*. Each plugin must be configured with at least one **artifact**.

Example plugins configuration with two connector plugins

```
apiVersion: kafka.strimzi.io/v1beta2
kind: KafkaConnect
metadata:
```

```
  name: my-connect-cluster
spec:
  #...
  build:
    output:
      #...
    plugins: 1
      - name: connector-1
        artifacts:
          - type: tgz
            url: <url_to_download_connector_1_artifact>
            sha512sum: <SHA-512_checksum_of_connector_1_artifact>
      - name: connector-2
        artifacts:
          - type: jar
            url: <url_to_download_connector_2_artifact>
            sha512sum: <SHA-512_checksum_of_connector_2_artifact>
  #...
```

1 (Required) List of connector plugins and their artifacts.

AMQ Streams supports the following types of artifacts:

- JAR files, which are downloaded and used directly
- TGZ archives, which are downloaded and unpacked
- ZIP archives, which are downloaded and unpacked
- Maven artifacts, which uses Maven coordinates
- Other artifacts, which are downloaded and used directly

IMPORTANT

AMQ Streams does not perform any security scanning of the downloaded artifacts. For security reasons, you should first verify the artifacts manually, and configure the checksum verification to make sure the same artifact is used in the automated build and in the Kafka Connect deployment.

Using JAR artifacts

JAR artifacts represent a JAR file that is downloaded and added to a container image. To use a JAR artifacts, set the **type** property to **jar**, and specify the download location using the **url** property.

Additionally, you can specify a SHA-512 checksum of the artifact. If specified, AMQ Streams will verify the checksum of the artifact while building the new container image.

Example JAR artifact

```
apiVersion: kafka.strimzi.io/v1beta2
kind: KafkaConnect
metadata:
  name: my-connect-cluster
spec:
```

```
#...
build:
  output:
    #...
  plugins:
    - name: my-plugin
      artifacts:
        - type: jar (1)
          url: https://my-domain.tld/my-jar.jar (2)
          sha512sum: 589...ab4 (3)
        - type: jar
          url: https://my-domain.tld/my-jar2.jar
#...
```

(1) (Required) Type of artifact.

(2) (Required) URL from which the artifact is downloaded.

(3) (Optional) SHA-512 checksum to verify the artifact.

Using TGZ artifacts

TGZ artifacts are used to download TAR archives that have been compressed using Gzip compression. The TGZ artifact can contain the whole Kafka Connect connector, even when comprising multiple different files. The TGZ artifact is automatically downloaded and unpacked by AMQ Streams while building the new container image. To use TGZ artifacts, set the **type** property to **tgz**, and specify the download location using the **url** property.

Additionally, you can specify a SHA-512 checksum of the artifact. If specified, AMQ Streams will verify the checksum before unpacking it and building the new container image.

Example TGZ artifact

```
apiVersion: kafka.strimzi.io/v1beta2
kind: KafkaConnect
metadata:
  name: my-connect-cluster
spec:
  #...
  build:
    output:
      #...
    plugins:
      - name: my-plugin
        artifacts:
          - type: tgz (1)
            url: https://my-domain.tld/my-connector-archive.tgz (2)
            sha512sum: 158...jg10 (3)
#...
```

(1) (Required) Type of artifact.

(2) (Required) URL from which the archive is downloaded.

3 (Optional) SHA-512 checksum to verify the artifact.

Using ZIP artifacts

ZIP artifacts are used to download ZIP compressed archives. Use ZIP artifacts in the same way as the TGZ artifacts described in the previous section. The only difference is you specify **type: zip** instead of **type: tgz**.

Using Maven artifacts

maven artifacts are used to specify connector plugin artifacts as Maven coordinates. The Maven coordinates identify plugin artifacts and dependencies so that they can be located and fetched from a Maven repository.

NOTE

The Maven repository must be accessible for the connector build process to add the artifacts to the container image.

Example Maven artifact

```
apiVersion: kafka.strimzi.io/v1beta2
kind: KafkaConnect
metadata:
  name: my-connect-cluster
spec:
  #...
  build:
    output:
      #...
    plugins:
      - name: my-plugin
        artifacts:
          - type: maven 1
            repository: https://mvnrepository.com 2
            group: org.apache.camel.kafkaconnector 3
            artifact: camel-kafka-connector 4
            version: 0.11.0 5
  #...
```

1 (Required) Type of artifact.

2 (Optional) Maven repository to download the artifacts from. If you do not specify a repository, Maven Central repository is used by default.

3 (Required) Maven group ID.

4 (Required) Maven artifact type.

5 (Required) Maven version number.

Using other artifacts

other artifacts represent any kind of file that is downloaded and added to a container image. If you want to use a specific name for the artifact in the resulting container image, use the **fileName** field. If a file name is not specified, the file is named based on the URL hash.

Additionally, you can specify a SHA-512 checksum of the artifact. If specified, AMQ Streams will verify the checksum of the artifact while building the new container image.

Example other artifact

```
apiVersion: kafka.strimzi.io/v1beta2
kind: KafkaConnect
metadata:
  name: my-connect-cluster
spec:
  #...
  build:
    output:
      #...
    plugins:
      - name: my-plugin
        artifacts:
          - type: other (1)
            url: https://my-domain.tld/my-other-file.ext (2)
            sha512sum: 589...ab4 (3)
            fileName: name-the-file.ext (4)
  #...
```

(1) (Required) Type of artifact.

(2) (Required) URL from which the artifact is downloaded.

(3) (Optional) SHA-512 checksum to verify the artifact.

(4) (Optional) The name under which the file is stored in the resulting container image.

85.3. BUILD SCHEMA PROPERTIES

Property	Description
output **DockerOutput**, **ImageStreamOutput**	Configures where should the newly built image be stored. Required. The type depends on the value of the **output.type** property within the given object, which must be one of [docker, imagestream].
resources ResourceRequirements	CPU and memory resources to reserve for the build. For more information, see the external documentation for core/v1 resourcerequirements.
plugins **Plugin** array	List of connector plugins which should be added to the Kafka Connect. Required.

CHAPTER 86. DOCKEROUTPUT SCHEMA REFERENCE

Used in: **Build**

The **type** property is a discriminator that distinguishes use of the **DockerOutput** type from **ImageStreamOutput**. It must have the value **docker** for the type **DockerOutput**.

Property	Description
image string	The full name which should be used for tagging and pushing the newly built image. For example **quay.io/my-organization/my-custom-connect:latest**. Required.
pushSecret string	Container Registry Secret with the credentials for pushing the newly built image.
additionalKanikoOptions string array	Configures additional options which will be passed to the Kaniko executor when building the new Connect image. Allowed options are: --customPlatform, --insecure, --insecure-pull, --insecure-registry, --log-format, --log-timestamp, --registry-mirror, --reproducible, --single-snapshot, --skip-tls-verify, --skip-tls-verify-pull, --skip-tls-verify-registry, --verbosity, --snapshotMode, --use-new-run. These options will be used only on OpenShift where the Kaniko executor is used. They will be ignored on OpenShift. The options are described in the Kaniko GitHub repository. Changing this field does not trigger new build of the Kafka Connect image.
type string	Must be **docker**.

CHAPTER 87. IMAGESTREAMOUTPUT SCHEMA REFERENCE

Used in: **Build**

The **type** property is a discriminator that distinguishes use of the **ImageStreamOutput** type from **DockerOutput**. It must have the value **imagestream** for the type **ImageStreamOutput**.

Property	Description
image	The name and tag of the ImageStream where the newly built image will be pushed. For example **my-custom-connect:latest**. Required.
string	
type	Must be **imagestream**.
string	

CHAPTER 88. PLUGIN SCHEMA REFERENCE

Used in: **Build**

Property	Description
name string	The unique name of the connector plugin. Will be used to generate the path where the connector artifacts will be stored. The name has to be unique within the KafkaConnect resource. The name has to follow the following pattern: **^[a-z][-_a-z0-9]*[a-z]$**. Required.
artifacts **JarArtifact**, **TgzArtifact**, **ZipArtifact**, **MavenArtifact**, **OtherArtifact** array	List of artifacts which belong to this connector plugin. Required.

CHAPTER 89. JARARTIFACT SCHEMA REFERENCE

Used in: **Plugin**

Property	Description
url	URL of the artifact which will be downloaded. AMQ Streams does not do any security scanning of the downloaded artifacts. For security reasons, you should first verify the artifacts manually and configure the checksum verification to make sure the same artifact is used in the automated build. Required for **jar**, **zip**, **tgz** and **other** artifacts. Not applicable to the **maven** artifact type.
string	
sha512sum	SHA512 checksum of the artifact. Optional. If specified, the checksum will be verified while building the new container. If not specified, the downloaded artifact will not be verified. Not applicable to the **maven** artifact type.
string	
insecure	By default, connections using TLS are verified to check they are secure. The server certificate used must be valid, trusted, and contain the server name. By setting this option to **true**, all TLS verification is disabled and the artifact will be downloaded, even when the server is considered insecure.
boolean	
type	Must be **jar**.
string	

CHAPTER 90. TGZARTIFACT SCHEMA REFERENCE

Used in: **Plugin**

Property	Description
url string	URL of the artifact which will be downloaded. AMQ Streams does not do any security scanning of the downloaded artifacts. For security reasons, you should first verify the artifacts manually and configure the checksum verification to make sure the same artifact is used in the automated build. Required for **jar**, **zip**, **tgz** and **other** artifacts. Not applicable to the **maven** artifact type.
sha512sum string	SHA512 checksum of the artifact. Optional. If specified, the checksum will be verified while building the new container. If not specified, the downloaded artifact will not be verified. Not applicable to the **maven** artifact type.
insecure boolean	By default, connections using TLS are verified to check they are secure. The server certificate used must be valid, trusted, and contain the server name. By setting this option to **true**, all TLS verification is disabled and the artifact will be downloaded, even when the server is considered insecure.
type string	Must be **tgz**.

CHAPTER 91. ZIPARTIFACT SCHEMA REFERENCE

Used in: **Plugin**

Property	Description
url string	URL of the artifact which will be downloaded. AMQ Streams does not do any security scanning of the downloaded artifacts. For security reasons, you should first verify the artifacts manually and configure the checksum verification to make sure the same artifact is used in the automated build. Required for **jar**, **zip**, **tgz** and **other** artifacts. Not applicable to the **maven** artifact type.
sha512sum string	SHA512 checksum of the artifact. Optional. If specified, the checksum will be verified while building the new container. If not specified, the downloaded artifact will not be verified. Not applicable to the **maven** artifact type.
insecure boolean	By default, connections using TLS are verified to check they are secure. The server certificate used must be valid, trusted, and contain the server name. By setting this option to **true**, all TLS verification is disabled and the artifact will be downloaded, even when the server is considered insecure.
type string	Must be **zip**.

CHAPTER 92. MAVENARTIFACT SCHEMA REFERENCE

Used in: **Plugin**

The **type** property is a discriminator that distinguishes use of the **MavenArtifact** type from **JarArtifact**, **TgzArtifact**, **ZipArtifact**, **OtherArtifact**. It must have the value **maven** for the type **MavenArtifact**.

Property	Description
repository string	Maven repository to download the artifact from. Applicable to the **maven** artifact type only.
group string	Maven group id. Applicable to the **maven** artifact type only.
artifact string	Maven artifact id. Applicable to the **maven** artifact type only.
version string	Maven version number. Applicable to the **maven** artifact type only.
insecure boolean	By default, connections using TLS are verified to check they are secure. The server certificate used must be valid, trusted, and contain the server name. By setting this option to **true**, all TLS verification is disabled and the artifacts will be downloaded, even when the server is considered insecure.
type string	Must be **maven**.

CHAPTER 93. OTHERARTIFACT SCHEMA REFERENCE

Used in: **Plugin**

Property	Description
url string	URL of the artifact which will be downloaded. AMQ Streams does not do any security scanning of the downloaded artifacts. For security reasons, you should first verify the artifacts manually and configure the checksum verification to make sure the same artifact is used in the automated build. Required for **jar**, **zip**, **tgz** and **other** artifacts. Not applicable to the **maven** artifact type.
sha512sum string	SHA512 checksum of the artifact. Optional. If specified, the checksum will be verified while building the new container. If not specified, the downloaded artifact will not be verified. Not applicable to the **maven** artifact type.
fileName string	Name under which the artifact will be stored.
insecure boolean	By default, connections using TLS are verified to check they are secure. The server certificate used must be valid, trusted, and contain the server name. By setting this option to **true**, all TLS verification is disabled and the artifact will be downloaded, even when the server is considered insecure.
type string	Must be **other**.

CHAPTER 94. KAFKACONNECTSTATUS SCHEMA REFERENCE

Used in: **KafkaConnect**

Property	Description
conditions **Condition** array	List of status conditions.
observedGeneration integer	The generation of the CRD that was last reconciled by the operator.
url string	The URL of the REST API endpoint for managing and monitoring Kafka Connect connectors.
connectorPlugins **ConnectorPlugin** array	The list of connector plugins available in this Kafka Connect deployment.
labelSelector string	Label selector for pods providing this resource.
replicas integer	The current number of pods being used to provide this resource.

CHAPTER 95. CONNECTORPLUGIN SCHEMA REFERENCE

Used in: **KafkaConnectStatus**, **KafkaMirrorMaker2Status**

Property	Description
type	The type of the connector plugin. The available types are **sink** and **source**.
string	
version	The version of the connector plugin.
string	
class	The class of the connector plugin.
string	

CHAPTER 96. KAFKATOPIC SCHEMA REFERENCE

Property	Description
spec	The specification of the topic.
KafkaTopicSpec	
status	The status of the topic.
KafkaTopicStatus	

CHAPTER 97. KAFKATOPICSPEC SCHEMA REFERENCE

Used in: **KafkaTopic**

Property	Description
partitions integer	The number of partitions the topic should have. This cannot be decreased after topic creation. It can be increased after topic creation, but it is important to understand the consequences that has, especially for topics with semantic partitioning. When absent this will default to the broker configuration for **num.partitions**.
replicas integer	The number of replicas the topic should have. When absent this will default to the broker configuration for **default.replication.factor**.
config map	The topic configuration.
topicName string	The name of the topic. When absent this will default to the metadata.name of the topic. It is recommended to not set this unless the topic name is not a valid OpenShift resource name.

CHAPTER 98. KAFKATOPICSTATUS SCHEMA REFERENCE

Used in: **KafkaTopic**

Property	Description
conditions **Condition** array	List of status conditions.
observedGeneration integer	The generation of the CRD that was last reconciled by the operator.
topicName string	Topic name.

CHAPTER 99. KAFKAUSER SCHEMA REFERENCE

Property	Description
spec	The specification of the user.
KafkaUserSpec	
status	The status of the Kafka User.
KafkaUserStatus	

CHAPTER 100. KAFKAUSERSPEC SCHEMA REFERENCE

Used in: **KafkaUser**

Property	Description
authentication **KafkaUserTlsClientAuthentication**, **KafkaUserTlsExternalClientAuthentication**, **KafkaUserScramSha512ClientAuthentication**	Authentication mechanism enabled for this Kafka user. The supported authentication mechanisms are **scram-sha-512**, **tls**, and **tls-external**. • **scram-sha-512** generates a secret with SASL SCRAM-SHA-512 credentials. • **tls** generates a secret with user certificate for mutual TLS authentication. • **tls-external** does not generate a user certificate. But prepares the user for using mutual TLS authentication using a user certificate generated outside the User Operator. ACLs and quotas set for this user are configured in the **CN=<username>** format. Authentication is optional. If authentication is not configured, no credentials are generated. ACLs and quotas set for the user are configured in the **<username>** format suitable for SASL authentication. The type depends on the value of the **authentication.type** property within the given object, which must be one of [tls, tls-external, scram-sha-512].
authorization **KafkaUserAuthorizationSimple**	Authorization rules for this Kafka user. The type depends on the value of the **authorization.type** property within the given object, which must be one of [simple].
quotas **KafkaUserQuotas**	Quotas on requests to control the broker resources used by clients. Network bandwidth and request rate quotas can be enforced.Kafka documentation for Kafka User quotas can be found at http://kafka.apache.org/documentation/#design_quotas.
template **KafkaUserTemplate**	Template to specify how Kafka User **Secrets** are generated.

CHAPTER 101. KAFKAUSERTLSCLIENTAUTHENTICATION SCHEMA REFERENCE

Used in: **KafkaUserSpec**

The **type** property is a discriminator that distinguishes use of the **KafkaUserTlsClientAuthentication** type from **KafkaUserTlsExternalClientAuthentication**, **KafkaUserScramSha512ClientAuthentication**. It must have the value **tls** for the type **KafkaUserTlsClientAuthentication**.

Property	Description
type	Must be **tls**.
string	

CHAPTER 102. KAFKAUSERTLSEXTERNALCLIENTAUTHENTICATION SCHEMA REFERENCE

Used in: **KafkaUserSpec**

The **type** property is a discriminator that distinguishes use of the **KafkaUserTlsExternalClientAuthentication** type from **KafkaUserTlsClientAuthentication**, **KafkaUserScramSha512ClientAuthentication**. It must have the value **tls-external** for the type **KafkaUserTlsExternalClientAuthentication**.

Property	Description
type string	Must be **tls-external**.

CHAPTER 103. KAFKAUSERSCRAMSHA512CLIENTAUTHENTICATION SCHEMA REFERENCE

Used in: **KafkaUserSpec**

The **type** property is a discriminator that distinguishes use of the **KafkaUserScramSha512ClientAuthentication** type from **KafkaUserTlsClientAuthentication**, **KafkaUserTlsExternalClientAuthentication**. It must have the value **scram-sha-512** for the type **KafkaUserScramSha512ClientAuthentication**.

Property	Description
password	Specify the password for the user. If not set, a new password is generated by the User Operator.
Password	
type	Must be **scram-sha-512**.
string	

CHAPTER 104. PASSWORD SCHEMA REFERENCE

Used in: **KafkaUserScramSha512ClientAuthentication**

Property	Description
valueFrom	Secret from which the password should be read.
PasswordSource	

CHAPTER 105. PASSWORDSOURCE SCHEMA REFERENCE

Used in: **Password**

Property	Description
secretKeyRef	Selects a key of a Secret in the resource's namespace. For more information, see the external documentation for core/v1 secretkeyselector.
SecretKeySelector	

CHAPTER 106. KAFKAUSERAUTHORIZATIONSIMPLE SCHEMA REFERENCE

Used in: **KafkaUserSpec**

The **type** property is a discriminator that distinguishes use of the **KafkaUserAuthorizationSimple** type from other subtypes which may be added in the future. It must have the value **simple** for the type **KafkaUserAuthorizationSimple**.

Property	Description
type string	Must be **simple**.
acls **AclRule** array	List of ACL rules which should be applied to this user.

CHAPTER 107. ACLRULE SCHEMA REFERENCE

Used in: **KafkaUserAuthorizationSimple**

Full list of **AclRule** schema properties

Configures access control rules for a **KafkaUser** when brokers are using **simple** authorization.

Example KafkaUser configuration with authorization

```
apiVersion: kafka.strimzi.io/v1beta2
kind: KafkaUser
metadata:
  name: my-user
  labels:
    strimzi.io/cluster: my-cluster
spec:
  # ...
  authorization:
    type: simple
    acls:
      - resource:
          type: topic
          name: my-topic
          patternType: literal
        operations:
          - Read
          - Describe
      - resource:
          type: group
          name: my-group
          patternType: prefix
        operations:
          - Read
```

107.1. RESOURCE

Use the **resource** property to specify the resource that the rule applies to.

Simple authorization supports four resource types, which are specified in the **type** property:

- Topics (**topic**)
- Consumer Groups (**group**)
- Clusters (**cluster**)
- Transactional IDs (**transactionalId**)

For Topic, Group, and Transactional ID resources you can specify the name of the resource the rule applies to in the **name** property.

Cluster type resources have no name.

A name is specified as a **literal** or a **prefix** using the **patternType** property.

- Literal names are taken exactly as they are specified in the **name** field.
- Prefix names use the **name** value as a prefix and then apply the rule to all resources with names starting with that value.

When **patternType** is set as **literal**, you can set the name to * to indicate that the rule applies to all resources.

Example ACL rule that allows the user to read messages from all topics

```
acls:
  - resource:
      type: topic
      name: "*"
      patternType: literal
    operations:
      - Read
```

107.2. TYPE

The **type** of rule, which is to **allow** or **deny** (not currently supported) an operations.

The **type** field is optional. If **type** is unspecified, the ACL rule is treated as an **allow** rule.

107.3. OPERATIONS

Specify a list of **operations** for the rule to allow or deny.

The following operations are supported:

- Read
- Write
- Delete
- Alter
- Describe
- All
- IdempotentWrite
- ClusterAction
- Create
- AlterConfigs
- DescribeConfigs

Only certain operations work with each resource.

For more details about **simple** authorization, ACLs, and supported combinations of resources and operations, see Authorization and ACLs.

107.4. HOST

Use the **host** property to specify a remote host from which the rule is allowed or denied.

Use an asterisk (*) to allow or deny the operation from all hosts. The **host** field is optional. If **host** is unspecified, the * value is used by default.

107.5. ACLRULE SCHEMA PROPERTIES

Property	Description
host string	The host from which the action described in the ACL rule is allowed or denied.
operation string (one of [Read, Write, Delete, Alter, Describe, All, IdempotentWrite, ClusterAction, Create, AlterConfigs, DescribeConfigs])	**The operation property has been deprecated, and should now be configured using spec.authorization.acls[*].operations.** Operation which will be allowed or denied. Supported operations are: Read, Write, Create, Delete, Alter, Describe, ClusterAction, AlterConfigs, DescribeConfigs, IdempotentWrite and All.
operations string (one or more of [Read, Write, Delete, Alter, Describe, All, IdempotentWrite, ClusterAction, Create, AlterConfigs, DescribeConfigs]) array	List of operations which will be allowed or denied. Supported operations are: Read, Write, Create, Delete, Alter, Describe, ClusterAction, AlterConfigs, DescribeConfigs, IdempotentWrite and All.
resource **AclRuleTopicResource**, **AclRuleGroupResource**, **AclRuleClusterResource**, **AclRuleTransactionalIdResource**	Indicates the resource for which given ACL rule applies. The type depends on the value of the **resource.type** property within the given object, which must be one of [topic, group, cluster, transactionalId].
type string (one of [allow, deny])	The type of the rule. Currently the only supported type is **allow**. ACL rules with type**allow** are used to allow user to execute the specified operations. Default value is **allow**.

CHAPTER 108. ACLRULETOPICRESOURCE SCHEMA REFERENCE

Used in: **AclRule**

The **type** property is a discriminator that distinguishes use of the **AclRuleTopicResource** type from **AclRuleGroupResource**, **AclRuleClusterResource**, **AclRuleTransactionalIdResource**. It must have the value **topic** for the type **AclRuleTopicResource**.

Property	Description
type string	Must be **topic**.
name string	Name of resource for which given ACL rule applies. Can be combined with **patternType** field to use prefix pattern.
patternType string (one of [prefix, literal])	Describes the pattern used in the resource field. The supported types are **literal** and **prefix**. With **literal** pattern type, the resource field will be used as a definition of a full topic name. With **prefix** pattern type, the resource name will be used only as a prefix. Default value is **literal**.

CHAPTER 109. ACLRULEGROUPRESOURCE SCHEMA REFERENCE

Used in: **AclRule**

The **type** property is a discriminator that distinguishes use of the **AclRuleGroupResource** type from **AclRuleTopicResource**, **AclRuleClusterResource**, **AclRuleTransactionalIdResource**. It must have the value **group** for the type **AclRuleGroupResource**.

Property	Description
type string	Must be **group**.
name string	Name of resource for which given ACL rule applies. Can be combined with **patternType** field to use prefix pattern.
patternType string (one of [prefix, literal])	Describes the pattern used in the resource field. The supported types are **literal** and **prefix**. With **literal** pattern type, the resource field will be used as a definition of a full topic name. With **prefix** pattern type, the resource name will be used only as a prefix. Default value is **literal**.

CHAPTER 110. ACLRULECLUSTERRESOURCE SCHEMA REFERENCE

Used in: **AclRule**

The **type** property is a discriminator that distinguishes use of the **AclRuleClusterResource** type from **AclRuleTopicResource**, **AclRuleGroupResource**, **AclRuleTransactionalIdResource**. It must have the value **cluster** for the type **AclRuleClusterResource**.

Property	Description
type	Must be **cluster**.
string	

CHAPTER 111. ACLRULETRANSACTIONALIDRESOURCE SCHEMA REFERENCE

Used in: **AclRule**

The **type** property is a discriminator that distinguishes use of the **AclRuleTransactionalIdResource** type from **AclRuleTopicResource**, **AclRuleGroupResource**, **AclRuleClusterResource**. It must have the value **transactionalId** for the type **AclRuleTransactionalIdResource**.

Property	Description
type string	Must be **transactionalId**.
name string	Name of resource for which given ACL rule applies. Can be combined with **patternType** field to use prefix pattern.
patternType string (one of [prefix, literal])	Describes the pattern used in the resource field. The supported types are **literal** and **prefix**. With **literal** pattern type, the resource field will be used as a definition of a full name. With **prefix** pattern type, the resource name will be used only as a prefix. Default value is **literal**.

CHAPTER 112. KAFKAUSERQUOTAS SCHEMA REFERENCE

Used in: **KafkaUserSpec**

Full list of **KafkaUserQuotas** schema properties

Kafka allows a user to set **quotas** to control the use of resources by clients.

112.1. QUOTAS

You can configure your clients to use the following types of quotas:

- *Network usage* quotas specify the byte rate threshold for each group of clients sharing a quota.
- *CPU utilization* quotas specify a window for broker requests from clients. The window is the percentage of time for clients to make requests. A client makes requests on the I/O threads and network threads of the broker.
- *Partition mutation* quotas limit the number of partition mutations which clients are allowed to make per second.

A partition mutation quota prevents Kafka clusters from being overwhelmed by concurrent topic operations. Partition mutations occur in response to the following types of user requests:

- Creating partitions for a new topic
- Adding partitions to an existing topic
- Deleting partitions from a topic

You can configure a partition mutation quota to control the rate at which mutations are accepted for user requests.

Using quotas for Kafka clients might be useful in a number of situations. Consider a wrongly configured Kafka producer which is sending requests at too high a rate. Such misconfiguration can cause a denial of service to other clients, so the problematic client ought to be blocked. By using a network limiting quota, it is possible to prevent this situation from significantly impacting other clients.

AMQ Streams supports user-level quotas, but not client-level quotas.

Example Kafka user quota configuration

```
spec:
  quotas:
    producerByteRate: 1048576
    consumerByteRate: 2097152
    requestPercentage: 55
    controllerMutationRate: 10
```

For more information about Kafka user quotas, refer to the Apache Kafka documentation.

112.2. KAFKAUSERQUOTAS SCHEMA PROPERTIES

Property	Description
consumerByteRate integer	A quota on the maximum bytes per-second that each client group can fetch from a broker before the clients in the group are throttled. Defined on a per-broker basis.
controllerMutationRate number	A quota on the rate at which mutations are accepted for the create topics request, the create partitions request and the delete topics request. The rate is accumulated by the number of partitions created or deleted.
producerByteRate integer	A quota on the maximum bytes per-second that each client group can publish to a broker before the clients in the group are throttled. Defined on a per-broker basis.
requestPercentage integer	A quota on the maximum CPU utilization of each client group as a percentage of network and I/O threads.

CHAPTER 113. KAFKAUSERTEMPLATE SCHEMA REFERENCE

Used in: **KafkaUserSpec**

Full list of **KafkaUserTemplate** schema properties

Specify additional labels and annotations for the secret created by the User Operator.

An example showing the KafkaUserTemplate

```
apiVersion: kafka.strimzi.io/v1beta2
kind: KafkaUser
metadata:
  name: my-user
  labels:
    strimzi.io/cluster: my-cluster
spec:
  authentication:
    type: tls
  template:
    secret:
      metadata:
        labels:
          label1: value1
        annotations:
          anno1: value1
  # ...
```

113.1. KAFKAUSERTEMPLATE SCHEMA PROPERTIES

Property	Description
secret	Template for KafkaUser resources. The template allows users to specify how the **Secret** with password or TLS certificates is generated.
ResourceTemplate	

CHAPTER 114. KAFKAUSERSTATUS SCHEMA REFERENCE

Used in: **KafkaUser**

Property	Description
conditions **Condition** array	List of status conditions.
observedGeneration integer	The generation of the CRD that was last reconciled by the operator.
username string	Username.
secret string	The name of **Secret** where the credentials are stored.

CHAPTER 115. KAFKAMIRRORMAKER SCHEMA REFERENCE

The type KafkaMirrorMaker has been deprecated. Please use **KafkaMirrorMaker2** instead.

Property	Description
spec	The specification of Kafka MirrorMaker.
KafkaMirrorMakerSpec	
status	The status of Kafka MirrorMaker.
KafkaMirrorMakerStatus	

CHAPTER 116. KAFKAMIRRORMAKERSPEC SCHEMA REFERENCE

Used in: **KafkaMirrorMaker**

Full list of **KafkaMirrorMakerSpec** schema properties

Configures Kafka MirrorMaker.

116.1. INCLUDE

Use the **include** property to configure a list of topics that Kafka MirrorMaker mirrors from the source to the target Kafka cluster.

The property allows any regular expression from the simplest case with a single topic name to complex patterns. For example, you can mirror topics A and B using **A|B** or all topics using ***. You can also pass multiple regular expressions separated by commas to the Kafka MirrorMaker.

116.2. KAFKAMIRRORMAKERCONSUMERSPEC AND KAFKAMIRRORMAKERPRODUCERSPEC

Use the **KafkaMirrorMakerConsumerSpec** and **KafkaMirrorMakerProducerSpec** to configure source (consumer) and target (producer) clusters.

Kafka MirrorMaker always works together with two Kafka clusters (source and target). To establish a connection, the bootstrap servers for the source and the target Kafka clusters are specified as comma-separated lists of ***HOSTNAME:PORT*** pairs. Each comma-separated list contains one or more Kafka brokers or a **Service** pointing to Kafka brokers specified as a ***HOSTNAME:PORT*** pair.

116.3. LOGGING

Kafka MirrorMaker has its own configurable logger:

- **mirrormaker.root.logger**

MirrorMaker uses the Apache **log4j** logger implementation.

Use the **logging** property to configure loggers and logger levels.

You can set the log levels by specifying the logger and level directly (inline) or use a custom (external) ConfigMap. If a ConfigMap is used, you set **logging.valueFrom.configMapKeyRef.name** property to the name of the ConfigMap containing the external logging configuration. Inside the ConfigMap, the logging configuration is described using **log4j.properties**. Both **logging.valueFrom.configMapKeyRef.name** and **logging.valueFrom.configMapKeyRef.key** properties are mandatory. A ConfigMap using the exact logging configuration specified is created with the custom resource when the Cluster Operator is running, then recreated after each reconciliation. If you do not specify a custom ConfigMap, default logging settings are used. If a specific logger value is not set, upper-level logger settings are inherited for that logger. For more information about log levels, see Apache logging services.

Here we see examples of **inline** and **external** logging. The **inline** logging specifies the root logger level. You can also set log levels for specific classes or loggers by adding them to the loggers property.

```
apiVersion: kafka.strimzi.io/v1beta2
```

```
kind: KafkaMirrorMaker
spec:
  # ...
  logging:
    type: inline
    loggers:
      mirrormaker.root.logger: INFO
      log4j.logger.org.apache.kafka.clients.NetworkClient: TRACE
      log4j.logger.org.apache.kafka.common.network.Selector: DEBUG
  # ...
```

NOTE

Setting a log level to **DEBUG** may result in a large amount of log output and may have performance implications.

```
apiVersion: kafka.strimzi.io/v1beta2
kind: KafkaMirrorMaker
spec:
  # ...
  logging:
    type: external
    valueFrom:
      configMapKeyRef:
        name: customConfigMap
        key: mirror-maker-log4j.properties
  # ...
```

Garbage collector (GC)

Garbage collector logging can also be enabled (or disabled) using the **jvmOptions** property.

116.4. KAFKAMIRRORMAKERSPEC SCHEMA PROPERTIES

Property	Description
version string	The Kafka MirrorMaker version. Defaults to 3.6.0. Consult the documentation to understand the process required to upgrade or downgrade the version.
replicas integer	The number of pods in the **Deployment**.
image string	The docker image for the pods.
consumer	Configuration of source cluster.

Property	Description
KafkaMirrorMakerConsumerSpec	
producer	Configuration of target cluster.
KafkaMirrorMakerProducerSpec	
resources	CPU and memory resources to reserve. For more information, see the external documentation for core/v1 resourcerequirements.
ResourceRequirements	
whitelist	**The whitelist property has been deprecated, and should now be configured using spec.include.** List of topics which are included for mirroring. This option allows any regular expression using Java-style regular expressions. Mirroring two topics named A and B is achieved by using the expression **A|B**. Or, as a special case, you can mirror all topics using the regular expression *****. You can also specify multiple regular expressions separated by commas.
string	
include	List of topics which are included for mirroring. This option allows any regular expression using Java-style regular expressions. Mirroring two topics named A and B is achieved by using the expression **A|B**. Or, as a special case, you can mirror all topics using the regular expression *****. You can also specify multiple regular expressions separated by commas.
string	
jvmOptions	JVM Options for pods.
JvmOptions	
logging	Logging configuration for MirrorMaker. The type depends on the value of the **logging.type** property within the given object, which must be one of [inline, external].
InlineLogging, **ExternalLogging**	
metricsConfig	Metrics configuration. The type depends on the value of the **metricsConfig.type** property within the given object, which must be one of [jmxPrometheusExporter].
JmxPrometheusExporterMetrics	
tracing	The configuration of tracing in Kafka MirrorMaker. The type depends on the value of the **tracing.type** property within the given object, which must be one of [jaeger, opentelemetry].
JaegerTracing, **OpenTelemetryTracing**	

Property	Description
template **KafkaMirrorMakerTemplate**	Template to specify how Kafka MirrorMaker resources, **Deployments** and **Pods**, are generated.
livenessProbe **Probe**	Pod liveness checking.
readinessProbe **Probe**	Pod readiness checking.

CHAPTER 117. KAFKAMIRRORMAKERCONSUMERSPEC SCHEMA REFERENCE

Used in: **KafkaMirrorMakerSpec**

Full list of **KafkaMirrorMakerConsumerSpec** schema properties

Configures a MirrorMaker consumer.

117.1. NUMSTREAMS

Use the **consumer.numStreams** property to configure the number of streams for the consumer.

You can increase the throughput in mirroring topics by increasing the number of consumer threads. Consumer threads belong to the consumer group specified for Kafka MirrorMaker. Topic partitions are assigned across the consumer threads, which consume messages in parallel.

117.2. OFFSETCOMMITINTERVAL

Use the **consumer.offsetCommitInterval** property to configure an offset auto-commit interval for the consumer.

You can specify the regular time interval at which an offset is committed after Kafka MirrorMaker has consumed data from the source Kafka cluster. The time interval is set in milliseconds, with a default value of 60,000.

117.3. CONFIG

Use the **consumer.config** properties to configure Kafka options for the consumer as keys.

The values can be one of the following JSON types:

- String
- Number
- Boolean

Exceptions

You can specify and configure the options listed in the Apache Kafka configuration documentation for consumers.

However, AMQ Streams takes care of configuring and managing options related to the following, which cannot be changed:

- Kafka cluster bootstrap address
- Security (encryption, authentication, and authorization)
- Consumer group identifier
- Interceptors

Properties with the following prefixes cannot be set:

- **bootstrap.servers**
- **group.id**
- **interceptor.classes**
- **sasl.**
- **security.**
- **ssl.**

If the **config** property contains an option that cannot be changed, it is disregarded, and a warning message is logged to the Cluster Operator log file. All other supported options are forwarded to MirrorMaker, including the following exceptions to the options configured by AMQ Streams:

- Any **ssl** configuration for supported TLS versions and cipher suites

IMPORTANT

The Cluster Operator does not validate keys or values in the **config** object provided. If an invalid configuration is provided, the MirrorMaker cluster might not start or might become unstable. In this case, fix the configuration so that the Cluster Operator can roll out the new configuration to all MirrorMaker nodes.

117.4. GROUPID

Use the **consumer.groupId** property to configure a consumer group identifier for the consumer.

Kafka MirrorMaker uses a Kafka consumer to consume messages, behaving like any other Kafka consumer client. Messages consumed from the source Kafka cluster are mirrored to a target Kafka cluster. A group identifier is required, as the consumer needs to be part of a consumer group for the assignment of partitions.

117.5. KAFKAMIRRORMAKERCONSUMERSPEC SCHEMA PROPERTIES

Property	Description
numStreams	Specifies the number of consumer stream threads to create.
integer	
offsetCommitInterval	Specifies the offset auto-commit interval in ms. Default value is 60000.
integer	
bootstrapServers	A list of host:port pairs for establishing the initial connection to the Kafka cluster.
string	

Property	Description
groupId	A unique string that identifies the consumer group this consumer belongs to.
string	
authentication	Authentication configuration for connecting to the cluster. The type depends on the value of the **authentication.type** property within the given object, which must be one of [tls, scram-sha-256, scram-sha-512, plain, oauth].
KafkaClientAuthenticationTls, **KafkaClientAuthenticationScramSha256**, **KafkaClientAuthenticationScramSha512**, **KafkaClientAuthenticationPlain**, **KafkaClientAuthenticationOAuth**	
config	The MirrorMaker consumer config. Properties with the following prefixes cannot be set: ssl., bootstrap.servers, group.id, sasl., security., interceptor.classes (with the exception of: ssl.endpoint.identification.algorithm, ssl.cipher.suites, ssl.protocol, ssl.enabled.protocols).
map	
tls	TLS configuration for connecting MirrorMaker to the cluster.
ClientTls	

CHAPTER 118. KAFKAMIRRORMAKERPRODUCERSPEC SCHEMA REFERENCE

Used in: **KafkaMirrorMakerSpec**

Full list of **KafkaMirrorMakerProducerSpec** schema properties

Configures a MirrorMaker producer.

118.1. ABORTONSENDFAILURE

Use the **producer.abortOnSendFailure** property to configure how to handle message send failure from the producer.

By default, if an error occurs when sending a message from Kafka MirrorMaker to a Kafka cluster:

- The Kafka MirrorMaker container is terminated in OpenShift.
- The container is then recreated.

If the **abortOnSendFailure** option is set to **false**, message sending errors are ignored.

118.2. CONFIG

Use the **producer.config** properties to configure Kafka options for the producer as keys.

The values can be one of the following JSON types:

- String
- Number
- Boolean

Exceptions

You can specify and configure the options listed in the Apache Kafka configuration documentation for producers.

However, AMQ Streams takes care of configuring and managing options related to the following, which cannot be changed:

- Kafka cluster bootstrap address
- Security (encryption, authentication, and authorization)
- Interceptors

Properties with the following prefixes cannot be set:

- **bootstrap.servers**
- **interceptor.classes**
- **sasl.**

- **security.**
- **ssl.**

If the **config** property contains an option that cannot be changed, it is disregarded, and a warning message is logged to the Cluster Operator log file. All other supported options are forwarded to MirrorMaker, including the following exceptions to the options configured by AMQ Streams:

- Any **ssl** configuration for supported TLS versions and cipher suites

IMPORTANT

The Cluster Operator does not validate keys or values in the **config** object provided. If an invalid configuration is provided, the MirrorMaker cluster might not start or might become unstable. In this case, fix the configuration so that the Cluster Operator can roll out the new configuration to all MirrorMaker nodes.

118.3. KAFKAMIRRORMAKERPRODUCERSPEC SCHEMA PROPERTIES

Property	Description
bootstrapServers string	A list of host:port pairs for establishing the initial connection to the Kafka cluster.
abortOnSendFailure boolean	Flag to set the MirrorMaker to exit on a failed send. Default value is **true**.
authentication **KafkaClientAuthenticationTls**, **KafkaClientAuthenticationScramSha256**, **KafkaClientAuthenticationScramSha512**, **KafkaClientAuthenticationPlain**, **KafkaClientAuthenticationOAuth**	Authentication configuration for connecting to the cluster. The type depends on the value of the **authentication.type** property within the given object, which must be one of [tls, scram-sha-256, scram-sha-512, plain, oauth].
config map	The MirrorMaker producer config. Properties with the following prefixes cannot be set: ssl., bootstrap.servers, sasl., security., interceptor.classes (with the exception of: ssl.endpoint.identification.algorithm, ssl.cipher.suites, ssl.protocol, ssl.enabled.protocols).
tls **ClientTls**	TLS configuration for connecting MirrorMaker to the cluster.

CHAPTER 119. KAFKAMIRRORMAKERTEMPLATE SCHEMA REFERENCE

Used in: **KafkaMirrorMakerSpec**

Property	Description
deployment	Template for Kafka MirrorMaker **Deployment**.
DeploymentTemplate	
pod	Template for Kafka MirrorMaker **Pods**.
PodTemplate	
podDisruptionBudget	Template for Kafka MirrorMaker **PodDisruptionBudget**.
PodDisruptionBudgetTemplate	
mirrorMakerContainer	Template for Kafka MirrorMaker container.
ContainerTemplate	
serviceAccount	Template for the Kafka MirrorMaker service account.
ResourceTemplate	

CHAPTER 120. KAFKAMIRRORMAKERSTATUS SCHEMA REFERENCE

Used in: **KafkaMirrorMaker**

Property	Description
conditions	List of status conditions.
Condition array	
observedGeneration	The generation of the CRD that was last reconciled by the operator.
integer	
labelSelector	Label selector for pods providing this resource.
string	
replicas	The current number of pods being used to provide this resource.
integer	

CHAPTER 121. KAFKABRIDGE SCHEMA REFERENCE

Property	Description
spec	The specification of the Kafka Bridge.
KafkaBridgeSpec	
status	The status of the Kafka Bridge.
KafkaBridgeStatus	

CHAPTER 122. KAFKABRIDGESPEC SCHEMA REFERENCE

Used in: **KafkaBridge**

Full list of **KafkaBridgeSpec** schema properties

Configures a Kafka Bridge cluster.

Configuration options relate to:

- Kafka cluster bootstrap address
- Security (encryption, authentication, and authorization)
- Consumer configuration
- Producer configuration
- HTTP configuration

122.1. LOGGING

Kafka Bridge has its own configurable loggers:

- **rootLogger.level**
- **logger.*<operation-id>***

You can replace ***<operation-id>*** in the **logger.*<operation-id>*** logger to set log levels for specific operations:

- **createConsumer**
- **deleteConsumer**
- **subscribe**
- **unsubscribe**
- **poll**
- **assign**
- **commit**
- **send**
- **sendToPartition**
- **seekToBeginning**
- **seekToEnd**
- **seek**
- **healthy**

- **ready**
- **openapi**

Each operation is defined according OpenAPI specification, and has a corresponding API endpoint through which the bridge receives requests from HTTP clients. You can change the log level on each endpoint to create fine-grained logging information about the incoming and outgoing HTTP requests.

Each logger has to be configured assigning it a **name** as **http.openapi.operation.*<operation-id>***. For example, configuring the logging level for the **send** operation logger means defining the following:

```
logger.send.name = http.openapi.operation.send
logger.send.level = DEBUG
```

Kafka Bridge uses the Apache **log4j2** logger implementation. Loggers are defined in the **log4j2.properties** file, which has the following default configuration for **healthy** and **ready** endpoints:

```
logger.healthy.name = http.openapi.operation.healthy
logger.healthy.level = WARN
logger.ready.name = http.openapi.operation.ready
logger.ready.level = WARN
```

The log level of all other operations is set to **INFO** by default.

Use the **logging** property to configure loggers and logger levels.

You can set the log levels by specifying the logger and level directly (inline) or use a custom (external) ConfigMap. If a ConfigMap is used, you set **logging.valueFrom.configMapKeyRef.name** property to the name of the ConfigMap containing the external logging configuration. The **logging.valueFrom.configMapKeyRef.name** and **logging.valueFrom.configMapKeyRef.key** properties are mandatory. Default logging is used if the **name** or **key** is not set. Inside the ConfigMap, the logging configuration is described using **log4j.properties**. For more information about log levels, see Apache logging services.

Here we see examples of **inline** and **external** logging.

Inline logging

```
apiVersion: kafka.strimzi.io/v1beta2
kind: KafkaBridge
spec:
  # ...
  logging:
    type: inline
    loggers:
      rootLogger.level: INFO
      # enabling DEBUG just for send operation
      logger.send.name: "http.openapi.operation.send"
      logger.send.level: DEBUG
  # ...
```

External logging

```
apiVersion: kafka.strimzi.io/v1beta2
kind: KafkaBridge
```

```
spec:
  # ...
  logging:
    type: external
    valueFrom:
      configMapKeyRef:
        name: customConfigMap
        key: bridge-logj42.properties
  # ...
```

Any available loggers that are not configured have their level set to **OFF**.

If the Kafka Bridge was deployed using the Cluster Operator, changes to Kafka Bridge logging levels are applied dynamically.

If you use external logging, a rolling update is triggered when logging appenders are changed.

Garbage collector (GC)

Garbage collector logging can also be enabled (or disabled) using the **jvmOptions** property.

122.2. KAFKABRIDGESPEC SCHEMA PROPERTIES

Property	Description
replicas integer	The number of pods in the **Deployment**. Defaults to **1**.
image string	The docker image for the pods.
bootstrapServers string	A list of host:port pairs for establishing the initial connection to the Kafka cluster.
tls **ClientTls**	TLS configuration for connecting Kafka Bridge to the cluster.
authentication **KafkaClientAuthenticationTls**, **KafkaClientAuthenticationScramSha256**, **KafkaClientAuthenticationScramSha512**, **KafkaClientAuthenticationPlain**, **KafkaClientAuthenticationOAuth**	Authentication configuration for connecting to the cluster. The type depends on the value of the **authentication.type** property within the given object, which must be one of [tls, scram-sha-256, scram-sha-512, plain, oauth].
http	The HTTP related configuration.

Property	Description
KafkaBridgeHttpConfig	
adminClient	Kafka AdminClient related configuration.
KafkaBridgeAdminClientSpec	
consumer	Kafka consumer related configuration.
KafkaBridgeConsumerSpec	
producer	Kafka producer related configuration.
KafkaBridgeProducerSpec	
resources	CPU and memory resources to reserve. For more information, see the external documentation for core/v1 resourcerequirements.
ResourceRequirements	
jvmOptions	**Currently not supported** JVM Options for pods.
JvmOptions	
logging	Logging configuration for Kafka Bridge. The type depends on the value of the **logging.type** property within the given object, which must be one of [inline, external].
InlineLogging, **ExternalLogging**	
clientRackInitImage	The image of the init container used for initializing the **client.rack**.
string	
rack	Configuration of the node label which will be used as the client.rack consumer configuration.
Rack	
enableMetrics	Enable the metrics for the Kafka Bridge. Default is false.
boolean	
livenessProbe	Pod liveness checking.
Probe	
readinessProbe	Pod readiness checking.

Property	Description
Probe	
template **KafkaBridgeTemplate**	Template for Kafka Bridge resources. The template allows users to specify how a **Deployment** and **Pod** is generated.
tracing **JaegerTracing**, **OpenTelemetryTracing**	The configuration of tracing in Kafka Bridge. The type depends on the value of the **tracing.type** property within the given object, which must be one of [jaeger, opentelemetry].

CHAPTER 123. KAFKABRIDGEHTTPCONFIG SCHEMA REFERENCE

Used in: **KafkaBridgeSpec**

Full list of **KafkaBridgeHttpConfig** schema properties

Configures HTTP access to a Kafka cluster for the Kafka Bridge.

The default HTTP configuration is for the Kafka Bridge to listen on port 8080.

123.1. CORS

As well as enabling HTTP access to a Kafka cluster, HTTP properties provide the capability to enable and define access control for the Kafka Bridge through Cross-Origin Resource Sharing (CORS). CORS is a HTTP mechanism that allows browser access to selected resources from more than one origin. To configure CORS, you define a list of allowed resource origins and HTTP access methods. For the origins, you can use a URL or a Java regular expression.

Example Kafka Bridge HTTP configuration

```
apiVersion: kafka.strimzi.io/v1beta2
kind: KafkaBridge
metadata:
  name: my-bridge
spec:
  # ...
  http:
    port: 8080
    cors:
      allowedOrigins: "https://strimzi.io"
      allowedMethods: "GET,POST,PUT,DELETE,OPTIONS,PATCH"
  # ...
```

123.2. KAFKABRIDGEHTTPCONFIG SCHEMA PROPERTIES

Property	Description
port	The port which is the server listening on.
integer	
cors	CORS configuration for the HTTP Bridge.
KafkaBridgeHttpCors	

CHAPTER 124. KAFKABRIDGEHTTPCORS SCHEMA REFERENCE

Used in: **KafkaBridgeHttpConfig**

Property	Description
allowedOrigins string array	List of allowed origins. Java regular expressions can be used.
allowedMethods string array	List of allowed HTTP methods.

CHAPTER 125. KAFKABRIDGEADMINCLIENTSPEC SCHEMA REFERENCE

Used in: **KafkaBridgeSpec**

Property	Description
config	The Kafka AdminClient configuration used for AdminClient instances created by the bridge.
map	

CHAPTER 126. KAFKABRIDGECONSUMERSPEC SCHEMA REFERENCE

Used in: **KafkaBridgeSpec**

Full list of **KafkaBridgeConsumerSpec** schema properties

Configures consumer options for the Kafka Bridge as keys.

The values can be one of the following JSON types:

- String
- Number
- Boolean

Exceptions

You can specify and configure the options listed in the Apache Kafka configuration documentation for consumers.

However, AMQ Streams takes care of configuring and managing options related to the following, which cannot be changed:

- Kafka cluster bootstrap address
- Security (encryption, authentication, and authorization)
- Consumer group identifier

Properties with the following prefixes cannot be set:

- **bootstrap.servers**
- **group.id**
- **sasl.**
- **security.**
- **ssl.**

If the **config** property contains an option that cannot be changed, it is disregarded, and a warning message is logged to the Cluster Operator log file. All other supported options are forwarded to Kafka Bridge, including the following exceptions to the options configured by AMQ Streams:

- Any **ssl** configuration for supported TLS versions and cipher suites

Example Kafka Bridge consumer configuration

```
apiVersion: kafka.strimzi.io/v1beta2
kind: KafkaBridge
metadata:
  name: my-bridge
spec:
  # ...
  consumer:
```

```
config:
  auto.offset.reset: earliest
  enable.auto.commit: true
# ...
```

IMPORTANT

The Cluster Operator does not validate keys or values in the **config** object. If an invalid configuration is provided, the Kafka Bridge deployment might not start or might become unstable. In this case, fix the configuration so that the Cluster Operator can roll out the new configuration to all Kafka Bridge nodes.

126.1. KAFKABRIDGECONSUMERSPEC SCHEMA PROPERTIES

Property	Description
config	The Kafka consumer configuration used for consumer instances created by the bridge. Properties with the following prefixes cannot be set: ssl., bootstrap.servers, group.id, sasl., security. (with the exception of: ssl.endpoint.identification.algorithm, ssl.cipher.suites, ssl.protocol, ssl.enabled.protocols).
map	

CHAPTER 127. KAFKABRIDGEPRODUCERSPEC SCHEMA REFERENCE

Used in: **KafkaBridgeSpec**

Full list of **KafkaBridgeProducerSpec** schema properties

Configures producer options for the Kafka Bridge as keys.

The values can be one of the following JSON types:

- String
- Number
- Boolean

Exceptions

You can specify and configure the options listed in the Apache Kafka configuration documentation for producers.

However, AMQ Streams takes care of configuring and managing options related to the following, which cannot be changed:

- Kafka cluster bootstrap address
- Security (encryption, authentication, and authorization)
- Consumer group identifier

Properties with the following prefixes cannot be set:

- **bootstrap.servers**
- **sasl.**
- **security.**
- **ssl.**

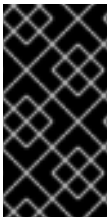
If the **config** property contains an option that cannot be changed, it is disregarded, and a warning message is logged to the Cluster Operator log file. All other supported options are forwarded to Kafka Bridge, including the following exceptions to the options configured by AMQ Streams:

- Any **ssl** configuration for supported TLS versions and cipher suites

Example Kafka Bridge producer configuration

```
apiVersion: kafka.strimzi.io/v1beta2
kind: KafkaBridge
metadata:
  name: my-bridge
spec:
  # ...
  producer:
    config:
```

```
    acks: 1
    delivery.timeout.ms: 300000
  # ...
```



IMPORTANT

The Cluster Operator does not validate keys or values in the **config** object. If an invalid configuration is provided, the Kafka Bridge deployment might not start or might become unstable. In this case, fix the configuration so that the Cluster Operator can roll out the new configuration to all Kafka Bridge nodes.

127.1. KAFKABRIDGEPRODUCERSPEC SCHEMA PROPERTIES

Property	Description
config	The Kafka producer configuration used for producer instances created by the bridge. Properties with the following prefixes cannot be set: ssl., bootstrap.servers, sasl., security. (with the exception of: ssl.endpoint.identification.algorithm, ssl.cipher.suites, ssl.protocol, ssl.enabled.protocols).
map	

CHAPTER 128. KAFKABRIDGETEMPLATE SCHEMA REFERENCE

Used in: **KafkaBridgeSpec**

Property	Description
deployment	Template for Kafka Bridge **Deployment**.
DeploymentTemplate	
pod	Template for Kafka Bridge **Pods**.
PodTemplate	
apiService	Template for Kafka Bridge API **Service**.
InternalServiceTemplate	
podDisruptionBudget	Template for Kafka Bridge **PodDisruptionBudget**.
PodDisruptionBudgetTemplate	
bridgeContainer	Template for the Kafka Bridge container.
ContainerTemplate	
clusterRoleBinding	Template for the Kafka Bridge ClusterRoleBinding.
ResourceTemplate	
serviceAccount	Template for the Kafka Bridge service account.
ResourceTemplate	
initContainer	Template for the Kafka Bridge init container.
ContainerTemplate	

CHAPTER 129. KAFKABRIDGESTATUS SCHEMA REFERENCE

Used in: **KafkaBridge**

Property	Description
conditions	List of status conditions.
Condition array	
observedGeneration	The generation of the CRD that was last reconciled by the operator.
integer	
url	The URL at which external client applications can access the Kafka Bridge.
string	
labelSelector	Label selector for pods providing this resource.
string	
replicas	The current number of pods being used to provide this resource.
integer	

CHAPTER 130. KAFKACONNECTOR SCHEMA REFERENCE

Property	Description
spec	The specification of the Kafka Connector.
KafkaConnectorSpec	
status	The status of the Kafka Connector.
KafkaConnectorStatus	

CHAPTER 131. KAFKACONNECTORSPEC SCHEMA REFERENCE

Used in: **KafkaConnector**

Property	Description
class string	The Class for the Kafka Connector.
tasksMax integer	The maximum number of tasks for the Kafka Connector.
autoRestart **AutoRestart**	Automatic restart of connector and tasks configuration.
config map	The Kafka Connector configuration. The following properties cannot be set: connector.class, tasks.max.
pause boolean	**The pause property has been deprecated.** Deprecated in AMQ Streams 2.6, use state instead. Whether the connector should be paused. Defaults to false.
state string (one of [running, paused, stopped])	The state the connector should be in. Defaults to running.

CHAPTER 132. AUTORESTART SCHEMA REFERENCE

Used in: **KafkaConnectorSpec**, **KafkaMirrorMaker2ConnectorSpec**

Full list of **AutoRestart** schema properties

Configures automatic restarts for connectors and tasks that are in a **FAILED** state.

When enabled, a back-off algorithm applies the automatic restart to each failed connector and its tasks. An incremental back-off interval is calculated using the formula **n * n + n** where **n** represents the number of previous restarts. This interval is capped at a maximum of 60 minutes. Consequently, a restart occurs immediately, followed by restarts after 2, 6, 12, 20, 30, 42, 56 minutes, and then at 60-minute intervals. By default, AMQ Streams initiates restarts of the connector and its tasks indefinitely. However, you can use the **maxRestarts** property to set a maximum on the number of restarts. If **maxRestarts** is configured and the connector still fails even after the final restart attempt, you must then restart the connector manually.

For Kafka Connect connectors, use the **autoRestart** property of the **KafkaConnector** resource to enable automatic restarts of failed connectors and tasks.

Enabling automatic restarts of failed connectors for Kafka Connect

```
apiVersion: kafka.strimzi.io/v1beta2
kind: KafkaConnector
metadata:
  name: my-source-connector
spec:
  autoRestart:
    enabled: true
```

If you prefer, you can also set a maximum limit on the number of restarts.

Enabling automatic restarts of failed connectors for Kafka Connect with limited number of restarts

```
apiVersion: kafka.strimzi.io/v1beta2
kind: KafkaConnector
metadata:
  name: my-source-connector
spec:
  autoRestart:
    enabled: true
    maxRestarts: 10
```

For MirrorMaker 2, use the **autoRestart** property of connectors in the **KafkaMirrorMaker2** resource to enable automatic restarts of failed connectors and tasks.

Enabling automatic restarts of failed connectors for MirrorMaker 2

```
apiVersion: kafka.strimzi.io/v1beta2
kind: KafkaMirrorMaker2
metadata:
  name: my-mm2-cluster
spec:
```

```
mirrors:
- sourceConnector:
    autoRestart:
      enabled: true
    # ...
  heartbeatConnector:
    autoRestart:
      enabled: true
    # ...
  checkpointConnector:
    autoRestart:
      enabled: true
    # ...
```

132.1. AUTORESTART SCHEMA PROPERTIES

Property	Description
enabled boolean	Whether automatic restart for failed connectors and tasks should be enabled or disabled.
maxRestarts integer	The maximum number of connector restarts that the operator will try. If the connector remains in a failed state after reaching this limit, it must be restarted manually by the user. Defaults to an unlimited number of restarts.

CHAPTER 133. KAFKACONNECTORSTATUS SCHEMA REFERENCE

Used in: **KafkaConnector**

Property	Description
conditions	List of status conditions.
Condition array	
observedGeneration	The generation of the CRD that was last reconciled by the operator.
integer	
autoRestart	The auto restart status.
AutoRestartStatus	
connectorStatus	The connector status, as reported by the Kafka Connect REST API.
map	
tasksMax	The maximum number of tasks for the Kafka Connector.
integer	
topics	The list of topics used by the Kafka Connector.
string array	

CHAPTER 134. AUTORESTARTSTATUS SCHEMA REFERENCE

Used in: **KafkaConnectorStatus**, **KafkaMirrorMaker2Status**

Property	Description
count	The number of times the connector or task is restarted.
integer	
connectorName	The name of the connector being restarted.
string	
lastRestartTimestamp	The last time the automatic restart was attempted. The required format is 'yyyy-MM-ddTHH:mm:ssZ' in the UTC time zone.
string	

CHAPTER 135. KAFKAMIRRORMAKER2 SCHEMA REFERENCE

Property	Description
spec	The specification of the Kafka MirrorMaker 2 cluster.
KafkaMirrorMaker2Spec	
status	The status of the Kafka MirrorMaker 2 cluster.
KafkaMirrorMaker2Status	

CHAPTER 136. KAFKAMIRRORMAKER2SPEC SCHEMA REFERENCE

Used in: **KafkaMirrorMaker2**

Property	Description
version string	The Kafka Connect version. Defaults to 3.6.0. Consult the user documentation to understand the process required to upgrade or downgrade the version.
replicas integer	The number of pods in the Kafka Connect group. Defaults to **3**.
image string	The docker image for the pods.
connectCluster string	The cluster alias used for Kafka Connect. The value must match the alias of the **target** Kafka cluster as specified in the **spec.clusters** configuration. The target Kafka cluster is used by the underlying Kafka Connect framework for its internal topics.
clusters **KafkaMirrorMaker2ClusterSpec** array	Kafka clusters for mirroring.
mirrors **KafkaMirrorMaker2MirrorSpec** array	Configuration of the MirrorMaker 2 connectors.
resources ResourceRequirements	The maximum limits for CPU and memory resources and the requested initial resources. For more information, see the external documentation for core/v1 resourcerequirements.
livenessProbe **Probe**	Pod liveness checking.
readinessProbe **Probe**	Pod readiness checking.
jvmOptions	JVM Options for pods.

Property	Description
JvmOptions	
jmxOptions	JMX Options.
KafkaJmxOptions	
logging	Logging configuration for Kafka Connect. The type depends on the value of the **logging.type** property within the given object, which must be one of [inline, external].
InlineLogging, **ExternalLogging**	
clientRackInitImage	The image of the init container used for initializing the **client.rack**.
string	
rack	Configuration of the node label which will be used as the **client.rack** consumer configuration.
Rack	
tracing	The configuration of tracing in Kafka Connect. The type depends on the value of the **tracing.type** property within the given object, which must be one of [jaeger, opentelemetry].
JaegerTracing, **OpenTelemetryTracing**	
template	Template for Kafka Connect and Kafka Mirror Maker 2 resources. The template allows users to specify how the **Deployment**, **Pods** and **Service** are generated.
KafkaConnectTemplate	
externalConfiguration	Pass data from Secrets or ConfigMaps to the Kafka Connect pods and use them to configure connectors.
ExternalConfiguration	
metricsConfig	Metrics configuration. The type depends on the value of the **metricsConfig.type** property within the given object, which must be one of [jmxPrometheusExporter].
JmxPrometheusExporterMetrics	

CHAPTER 137. KAFKAMIRRORMAKER2CLUSTERSPEC SCHEMA REFERENCE

Used in: **KafkaMirrorMaker2Spec**

Full list of **KafkaMirrorMaker2ClusterSpec** schema properties

Configures Kafka clusters for mirroring.

137.1. CONFIG

Use the **config** properties to configure Kafka options.

Standard Apache Kafka configuration may be provided, restricted to those properties not managed directly by AMQ Streams.

For client connection using a specific *cipher suite* for a TLS version, you can configure allowed **ssl** properties. You can also configure the **ssl.endpoint.identification.algorithm** property to enable or disable hostname verification.

137.2. KAFKAMIRRORMAKER2CLUSTERSPEC SCHEMA PROPERTIES

Property	Description
alias string	Alias used to reference the Kafka cluster.
bootstrapServers string	A comma-separated list of **host:port** pairs for establishing the connection to the Kafka cluster.
tls **ClientTls**	TLS configuration for connecting MirrorMaker 2 connectors to a cluster.
authentication **KafkaClientAuthenticationTls**, **KafkaClientAuthenticationScramSha256**, **KafkaClientAuthenticationScramSha512**, **KafkaClientAuthenticationPlain**, **KafkaClientAuthenticationOAuth**	Authentication configuration for connecting to the cluster. The type depends on the value of the **authentication.type** property within the given object, which must be one of [tls, scram-sha-256, scram-sha-512, plain, oauth].

Property	Description
config	The MirrorMaker 2 cluster config. Properties with the following prefixes cannot be set: ssl., sasl., security., listeners, plugin.path, rest., bootstrap.servers, consumer.interceptor.classes, producer.interceptor.classes (with the exception of: ssl.endpoint.identification.algorithm, ssl.cipher.suites, ssl.protocol, ssl.enabled.protocols).
map	

CHAPTER 138. KAFKAMIRRORMAKER2MIRRORSPEC SCHEMA REFERENCE

Used in: **KafkaMirrorMaker2Spec**

Property	Description
sourceCluster string	The alias of the source cluster used by the Kafka MirrorMaker 2 connectors. The alias must match a cluster in the list at **spec.clusters**.
targetCluster string	The alias of the target cluster used by the Kafka MirrorMaker 2 connectors. The alias must match a cluster in the list at **spec.clusters**.
sourceConnector **KafkaMirrorMaker2ConnectorSpec**	The specification of the Kafka MirrorMaker 2 source connector.
heartbeatConnector **KafkaMirrorMaker2ConnectorSpec**	The specification of the Kafka MirrorMaker 2 heartbeat connector.
checkpointConnector **KafkaMirrorMaker2ConnectorSpec**	The specification of the Kafka MirrorMaker 2 checkpoint connector.
topicsPattern string	A regular expression matching the topics to be mirrored, for example, "topic1|topic2|topic3". Comma-separated lists are also supported.
topicsBlacklistPattern string	**The topicsBlacklistPattern property has been deprecated, and should now be configured using .spec.mirrors.topicsExcludePattern.** A regular expression matching the topics to exclude from mirroring. Comma-separated lists are also supported.
topicsExcludePattern string	A regular expression matching the topics to exclude from mirroring. Comma-separated lists are also supported.
groupsPattern string	A regular expression matching the consumer groups to be mirrored. Comma-separated lists are also supported.

Property	Description
groupsBlacklistPattern string	**The groupsBlacklistPattern property has been deprecated, and should now be configured using .spec.mirrors.groupsExcludePattern.** A regular expression matching the consumer groups to exclude from mirroring. Comma-separated lists are also supported.
groupsExcludePattern string	A regular expression matching the consumer groups to exclude from mirroring. Comma-separated lists are also supported.

CHAPTER 139. KAFKAMIRRORMAKER2CONNECTORSPEC SCHEMA REFERENCE

Used in: **KafkaMirrorMaker2MirrorSpec**

Property	Description
tasksMax integer	The maximum number of tasks for the Kafka Connector.
config map	The Kafka Connector configuration. The following properties cannot be set: connector.class, tasks.max.
autoRestart **AutoRestart**	Automatic restart of connector and tasks configuration.
pause boolean	**The pause property has been deprecated.** Deprecated in AMQ Streams 2.6, use state instead. Whether the connector should be paused. Defaults to false.
state string (one of [running, paused, stopped])	The state the connector should be in. Defaults to running.

CHAPTER 140. KAFKAMIRRORMAKER2STATUS SCHEMA REFERENCE

Used in: **KafkaMirrorMaker2**

Property	Description
conditions **Condition** array	List of status conditions.
observedGeneration integer	The generation of the CRD that was last reconciled by the operator.
url string	The URL of the REST API endpoint for managing and monitoring Kafka Connect connectors.
autoRestartStatuses **AutoRestartStatus** array	List of MirrorMaker 2 connector auto restart statuses.
connectorPlugins **ConnectorPlugin** array	The list of connector plugins available in this Kafka Connect deployment.
connectors map array	List of MirrorMaker 2 connector statuses, as reported by the Kafka Connect REST API.
labelSelector string	Label selector for pods providing this resource.
replicas integer	The current number of pods being used to provide this resource.

CHAPTER 141. KAFKAREBALANCE SCHEMA REFERENCE

Property	Description
spec	The specification of the Kafka rebalance.
KafkaRebalanceSpec	
status	The status of the Kafka rebalance.
KafkaRebalanceStatus	

CHAPTER 142. KAFKAREBALANCESPEC SCHEMA REFERENCE

Used in: **KafkaRebalance**

Property	Description
mode string (one of [remove-brokers, full, add-brokers])	Mode to run the rebalancing. The supported modes are **full**, **add-brokers**, **remove-brokers**. If not specified, the **full** mode is used by default. • **full** mode runs the rebalancing across all the brokers in the cluster. • **add-brokers** mode can be used after scaling up the cluster to move some replicas to the newly added brokers. • **remove-brokers** mode can be used before scaling down the cluster to move replicas out of the brokers to be removed.
brokers integer array	The list of newly added brokers in case of scaling up or the ones to be removed in case of scaling down to use for rebalancing. This list can be used only with rebalancing mode **add-brokers** and **removed-brokers**. It is ignored with**full** mode.
goals string array	A list of goals, ordered by decreasing priority, to use for generating and executing the rebalance proposal. The supported goals are available at https://github.com/linkedin/cruise-control#goals. If an empty goals list is provided, the goals declared in the default.goals Cruise Control configuration parameter are used.
skipHardGoalCheck boolean	Whether to allow the hard goals specified in the Kafka CR to be skipped in optimization proposal generation. This can be useful when some of those hard goals are preventing a balance solution being found. Default is false.
rebalanceDisk boolean	Enables intra-broker disk balancing, which balances disk space utilization between disks on the same broker. Only applies to Kafka deployments that use JBOD storage with multiple disks. When enabled, inter-broker balancing is disabled. Default is false.
excludedTopics string	A regular expression where any matching topics will be excluded from the calculation of optimization proposals. This expression will be parsed by the java.util.regex.Pattern class; for more information on the supported format consult the documentation for that class.

Property	Description
concurrentPartitionMovementsPerBroker integer	The upper bound of ongoing partition replica movements going into/out of each broker. Default is 5.
concurrentIntraBrokerPartitionMovements integer	The upper bound of ongoing partition replica movements between disks within each broker. Default is 2.
concurrentLeaderMovements integer	The upper bound of ongoing partition leadership movements. Default is 1000.
replicationThrottle integer	The upper bound, in bytes per second, on the bandwidth used to move replicas. There is no limit by default.
replicaMovementStrategies string array	A list of strategy class names used to determine the execution order for the replica movements in the generated optimization proposal. By default BaseReplicaMovementStrategy is used, which will execute the replica movements in the order that they were generated.

CHAPTER 143. KAFKAREBALANCESTATUS SCHEMA REFERENCE

Used in: **KafkaRebalance**

Property	Description
conditions **Condition** array	List of status conditions.
observedGeneration integer	The generation of the CRD that was last reconciled by the operator.
sessionId string	The session identifier for requests to Cruise Control pertaining to this KafkaRebalance resource. This is used by the Kafka Rebalance operator to track the status of ongoing rebalancing operations.
optimizationResult map	A JSON object describing the optimization result.

CHAPTER 144. KAFKANODEPOOL SCHEMA REFERENCE

Property	Description
spec	The specification of the KafkaNodePool.
KafkaNodePoolSpec	
status	The status of the KafkaNodePool.
KafkaNodePoolStatus	

CHAPTER 145. KAFKANODEPOOLSPEC SCHEMA REFERENCE

Used in: **KafkaNodePool**

Property	Description
replicas integer	The number of pods in the pool.
storage **EphemeralStorage**, **PersistentClaimStorage**, **JbodStorage**	Storage configuration (disk). Cannot be updated. The type depends on the value of the **storage.type** property within the given object, which must be one of [ephemeral, persistent-claim, jbod].
roles string (one or more of [controller, broker]) array	The roles that the nodes in this pool will have when KRaft mode is enabled. Supported values are 'broker' and 'controller'. This field is required. When KRaft mode is disabled, the only allowed value if **broker**.
resources ResourceRequirements	CPU and memory resources to reserve. For more information, see the external documentation for core/v1 resourcerequirements.
jvmOptions **JvmOptions**	JVM Options for pods.
template **KafkaNodePoolTemplate**	Template for pool resources. The template allows users to specify how the resources belonging to this pool are generated.

CHAPTER 146. KAFKANODEPOOLTEMPLATE SCHEMA REFERENCE

Used in: **KafkaNodePoolSpec**

Property	Description
podSet	Template for Kafka **StrimziPodSet** resource.
ResourceTemplate	
pod	Template for Kafka **Pods**.
PodTemplate	
perPodService	Template for Kafka per-pod **Services** used for access from outside of OpenShift.
ResourceTemplate	
perPodRoute	Template for Kafka per-pod **Routes** used for access from outside of OpenShift.
ResourceTemplate	
perPodIngress	Template for Kafka per-pod **Ingress** used for access from outside of OpenShift.
ResourceTemplate	
persistentVolumeClaim	Template for all Kafka **PersistentVolumeClaims**.
ResourceTemplate	
kafkaContainer	Template for the Kafka broker container.
ContainerTemplate	
initContainer	Template for the Kafka init container.
ContainerTemplate	

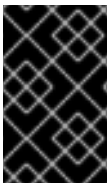
CHAPTER 147. KAFKANODEPOOLSTATUS SCHEMA REFERENCE

Used in: **KafkaNodePool**

Property	Description
conditions **Condition** array	List of status conditions.
observedGeneration integer	The generation of the CRD that was last reconciled by the operator.
nodeIds integer array	Node IDs used by Kafka nodes in this pool.
clusterId string	Kafka cluster ID.
replicas integer	The current number of pods being used to provide this resource.
labelSelector string	Label selector for pods providing this resource.

CHAPTER 148. STRIMZIPODSET SCHEMA REFERENCE

Full list of **StrimziPodSet** schema properties



IMPORTANT

StrimziPodSet is an internal AMQ Streams resource. Information is provided for reference only. Do not create, modify or delete **StrimziPodSet** resources as this might cause errors.

148.1. STRIMZIPODSET SCHEMA PROPERTIES

Property	Description
spec	The specification of the StrimziPodSet.
StrimziPodSetSpec	
status	The status of the StrimziPodSet.
StrimziPodSetStatus	

CHAPTER 149. STRIMZIPODSETSPEC SCHEMA REFERENCE

Used in: **StrimziPodSet**

Property	Description
selector LabelSelector	Selector is a label query which matches all the pods managed by this **StrimziPodSet**. Only **matchLabels** is supported. If **matchExpressions** is set, it will be ignored. For more information, see the external documentation for meta/v1 labelselector.
pods Map array	The Pods managed by this StrimziPodSet. For more information, see the external documentation for core/v1 pods.

CHAPTER 150. STRIMZIPODSETSTATUS SCHEMA REFERENCE

Used in: **StrimziPodSet**

Property	Description
conditions **Condition** array	List of status conditions.
observedGeneration integer	The generation of the CRD that was last reconciled by the operator.
pods integer	Number of pods managed by this **StrimziPodSet** resource.
readyPods integer	Number of pods managed by this **StrimziPodSet** resource that are ready.
currentPods integer	Number of pods managed by this **StrimziPodSet** resource that have the current revision.

APPENDIX A. USING YOUR SUBSCRIPTION

AMQ Streams is provided through a software subscription. To manage your subscriptions, access your account at the Red Hat Customer Portal.

Accessing Your Account

1. Go to access.redhat.com.
2. If you do not already have an account, create one.
3. Log in to your account.

Activating a Subscription

1. Go to access.redhat.com.
2. Navigate to **My Subscriptions**.
3. Navigate to **Activate a subscription** and enter your 16-digit activation number.

Downloading Zip and Tar Files

To access zip or tar files, use the customer portal to find the relevant files for download. If you are using RPM packages, this step is not required.

1. Open a browser and log in to the Red Hat Customer Portal **Product Downloads** page at access.redhat.com/downloads.
2. Locate the **AMQ Streams for Apache Kafka** entries in the **INTEGRATION AND AUTOMATION** category.
3. Select the desired AMQ Streams product. The **Software Downloads** page opens.
4. Click the **Download** link for your component.

Installing packages with DNF

To install a package and all the package dependencies, use:

```
dnf install <package_name>
```

To install a previously-downloaded package from a local directory, use:

```
dnf install <path_to_download_package>
```

Revised on 2023-12-06 17:39:54 UTC