Strimzi Custom Resource API Reference

Table of Contents

Chapter 1. Using schema properties to configure custom resources

Custom resources offer a flexible way to manage and fine-tune the operation of Strimzi 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 Strimzi. 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 Strimzi 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 Kubernetes it may not be necessary to run multiple replicas for high availability. When the node where the component is deployed crashes, Kubernetes 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 Kubernetes cluster. They can also refer to a Kafka cluster not deployed by Strimzi.

If on the same Kubernetes 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 Strimzi but on different Kubernetes 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 Strimzi, 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 2, and HTTP 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`

Use the `tls` and `trustedCertificates` properties to enable TLS encryption and specify secrets under which TLS certificates are stored in X.509 format. You can add this configuration to the Kafka Connect, Kafka MirrorMaker, and HTTP Bridge components for TLS connections to the Kafka cluster.

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:

Creating a secret

```
kubectl create secret generic <my_secret> \
--from-file=<my_tls_certificate_file.crt>
```

- Replace `<my_secret>` with your secret name.
- Replace `<my_tls_certificate_file.crt>` with the path to your TLS certificate file.

Use the `pattern` property to include all files in the secret that match the pattern. Using the `pattern` property means that the custom resource does not need to be updated if certificate file names change. However, you can specify a specific file using the `certificate` property instead of the `pattern` property.

Example TLS encryption configuration for components

```
tls:
  trustedCertificates:
    - secretName: my-cluster-cluster-cert
      pattern: "*.crt"
    - secretName: my-cluster-cluster-cert
      certificate: ca2.crt
```

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: []
```

Similarly, you can use the `tlsTrustedCertificates` property in the configuration for `oauth` and `keycloak` authentication and authorization types that integrate with authorization servers. The configuration sets up encrypted TLS connections to the authorization server.

Example TLS encryption configuration for authentication types

```
tlsTrustedCertificates:
  - secretName: oauth-server-ca
    pattern: "*.crt"
```

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

2.5. `resources`

Configure resource *requests* and *limits* to control resources for Strimzi containers. You can specify requests and limits for `memory` and `cpu` resources. The requests should be enough to ensure a stable

performance of Kafka.

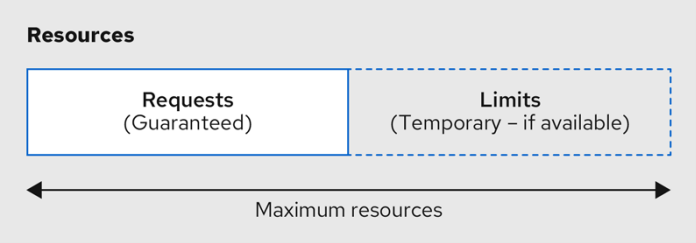
How you configure resources in a production environment depends on a number of factors. For example, applications are likely to be sharing resources in your Kubernetes cluster.

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, Kubernetes uses the same value for both. Setting equal requests and limits for resources guarantees quality of service, as Kubernetes will not kill containers unless they exceed their limits.

Configure resource requests and limits for components using `resources` properties in the `spec` of following custom resources:

Use the `KafkaNodePool` custom resource for Kafka nodes (`spec.resources`)

Use the `Kafka` custom resource for the following components:

- Topic Operator (`spec.entityOperator.topicOperator.resources`)
- User Operator (`spec.entityOperator.userOperator.resources`)
- Cruise Control (`spec.cruiseControl.resources`)
- Kafka Exporter (`spec.kafkaExporter.resources`)

For other components, resources are configured in the corresponding custom resource. For example:

- `KafkaConnect` resource for Kafka Connect (`spec.resources`)

- `KafkaMirrorMaker2` resource for MirrorMaker (`spec.resources`)
- `KafkaBridge` resource for HTTP Bridge (`spec.resources`)

Example resource configuration for a node pool

```
apiVersion: kafka.strimzi.io/v1
kind: KafkaNodePool
metadata:
  name: pool-a
  labels:
    strimzi.io/cluster: my-cluster
spec:
  replicas: 3
  roles:
    - broker
  resources:
      requests:
        memory: 64Gi
        cpu: "8"
      limits:
        memory: 64Gi
        cpu: "12"
  # ...
```

Example resource configuration for the Topic Operator

```
apiVersion: kafka.strimzi.io/v1
kind: Kafka
metadata:
  name: my-cluster
spec:
  # ..
  entityOperator:
    #...
    topicOperator:
      #...
      resources:
        requests:
          memory: 512Mi
          cpu: "1"
        limits:
          memory: 512Mi
          cpu: "1"
```

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

NOTE Strimzi uses the Kubernetes syntax for specifying `memory` and `cpu` resources. For more information about managing computing resources on Kubernetes, 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 Kubernetes 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 custom image.

For example, if your network does not allow access to the container repository used by Strimzi, you can copy the Strimzi images or build them from the source. However, if the configured image is not compatible with Strimzi 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.entityOperator.topicOperator`
- `Kafka.spec.entityOperator.userOperator`
- `Kafka.spec.cruiseControl`
- `Kafka.spec.kafkaExporter`
- `Kafka.spec.kafkaBridge`
- `KafkaConnect.spec`
- `KafkaMirrorMaker2.spec`
- `KafkaBridge.spec`

NOTE	Changing the Kafka image version does not automatically update the image versions for other Kafka components, such as Kafka Exporter. These components are not version dependent, so no additional configuration is necessary when updating the Kafka image version.

Setting Kafka component images

Strimzi supports multiple Kafka versions across Kafka, Kafka Connect, and Kafka MirrorMaker 2 components. Each component requires a specific container image, which can be configured in two places:

1. Cluster Operator environment variables (Default image mappings)
2. Custom resource configuration

Each environment variable maps Kafka versions to container images. These mappings are used when a custom resource does not explicitly specify an image. You can override the default image by specifying the image and the matching version in the custom resource.

Table 1. Cluster Operator environment variables

Component	Environment variable
Kafka	`STRIMZI_KAFKA_IMAGES`
Kafka Connect	`STRIMZI_KAFKA_CONNECT_IMAGES`
MirrorMaker 2	`STRIMZI_KAFKA_MIRROR_MAKER2_IMAGES`

Table 2. Custom resource configuration

Custom resource	Image property	Version property
`Kafka`	`spec.kafka.image`	`spec.kafka.version`
`KafkaConnect`	`spec.image`	`spec.version`
`KafkaMirrorMaker2`	`spec.image`	`spec.version`

The values for the environment variables and those specified for `image` and `version` in the component configuration determines the image and Kafka version used:

Table 3. Values specified for image and Kafka version

`image` set?	`version` set?	Result
☐	☐	Uses Cluster Operator's default image and corresponding Kafka version
✔	☐	Uses specified image and default Kafka version
☐	✔	Uses image from environment variable for specified Kafka version
✔	✔	Uses specified image and assumes specified Kafka version matches

NOTE To avoid Kafka version and image mismatches, set the `version` property and allow the Cluster Operator to select the matching image from its mappings. If you need to change the default image mapping for a given Kafka version, configure the Cluster Operator's environment variables.

Even if the configuration is syntactically correct, it can still be invalid if the image and version mismatch. To ensure a valid configuration:

- The specified `version` **must** match the Kafka version that the image is built for.
- The specified `version` must be one of the versions supported by the operator.
- If you set a custom `image`, always set `version` to the Kafka version of that image.

Here we can see what happens with versions 4.0.0 and 4.1.1 (the default).

Table 4. Valid and invalid configuration examples

`image` set?	`version` set?	Image used	Kafka version	Valid?
☐	☐	4.1.1 (default)	4.1.1	✔
☐	4.0.0	4.0.0 (from mapping)	4.0.0	✔
Custom 4.0.0	4.0.0	Custom 4.0.0	4.0.0	✔
Custom 4.0.0	☐	Custom 4.0.0	4.1.1 (default)	☐
Custom 4.0.0	4.1.1	Custom 4.0.0	4.1.1	☐

NOTE *Custom 4.0.0* refers to a custom user-provided container image built against Kafka 4.0.0. Setting a **custom image** means you set the `image` property to a user-provided image. The operator does not automatically change this value during upgrades.

Handling upgrades with custom images

When you set a custom image through the `image` property in a custom resource, you must keep the `image` and version in `sync` during upgrades. The Cluster Operator does not automatically update images defined this way. (This limitation does not apply when you use Kafka Connect Build or the default image mappings defined in the Cluster Operator environment variables.)

To avoid version mismatches when changing the Kafka version:

- Pause reconciliation of the custom resource.
- Upgrade the Cluster Operator to a release that supports a new Kafka version.
- Update the custom resource:
  - Set `spec.*.version` to the target Kafka version
  - Set `spec.*.image` to the custom image built for that version
- Unpause the reconciliation.

If you don't follow these steps, the operator might attempt to upgrade the resource before the correct image is specified, leading to mismatches.

Configuring the `image` property in other resources

For the `image` property in the custom resources for other components, the given value is used during deployment. If the `image` property is not set, the container `image` specified as an environment variable in the Cluster Operator configuration is used. If an `image` name is not defined in the Cluster Operator configuration, then a default value is used.

For more information on image environment variables, see Configuring the Cluster Operator.

Table 5. Image environment variables and defaults

Component	Environment variable	Default image
Topic Operator	`STRIMZI_DEFAULT_TOPIC_OPERATOR_IMAGE`	`quay.io/strimzi/operator:0.50.0`

Component	Environment variable	Default image
User Operator	STRIMZI_DEFAULT_USER_OPERATOR_IMAGE	quay.io/strimzi/operator:0.50.0
Kafka Exporter	STRIMZI_DEFAULT_KAFKA_EXPORTER_IMAGE	quay.io/strimzi/kafka:0.50.0-kafka-4.1.1
Cruise Control	STRIMZI_DEFAULT_CRUISE_CONTROL_IMAGE	quay.io/strimzi/kafka:0.50.0-kafka-4.1.1
HTTP Bridge	STRIMZI_DEFAULT_KAFKA_BRIDGE_IMAGE	quay.io/strimzi/kafka-bridge:0.33.1
Kafka initializer	STRIMZI_DEFAULT_KAFKA_INIT_IMAGE	quay.io/strimzi/operator:0.50.0

Example container image configuration

```
apiVersion: kafka.strimzi.io/v1
kind: Kafka
metadata:
  name: my-cluster
spec:
  kafka:
    # ...
    image: my-org/my-image:latest
    # ...
```

2.7. `livenessProbe` and `readinessProbe` healthchecks

Use the `livenessProbe` and `readinessProbe` properties to configure healthcheck probes supported in Strimzi.

Healthchecks are periodical tests which verify the health of an application. When a Healthcheck probe fails, Kubernetes 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. Strimzi provides support for Prometheus JMX Exporter and Strimzi Metrics Reporter. Only one of these can be selected at any given time.

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

When the `metricsConfig` property is not defined in the resource, the Prometheus metrics are not enabled.

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

Using Prometheus JMX Exporter

The `metricsConfig` property contains a reference to a `ConfigMap` that has additional configurations for the Prometheus JMX Exporter. When configured to use Prometheus JMX Exporter, Strimzi converts the JMX metrics provided by Apache Kafka into a Prometheus-compatible format.

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/v1
kind: Kafka
metadata:
  name: my-cluster
spec:
  kafka:
    # ...
    metricsConfig:
      type: jmxPrometheusExporter
      valueFrom:
        configMapKeyRef:
          name: my-config-map
          key: my-key
    # ...
```

Using Strimzi Metrics Reporter

The `metricsConfig` property contains configurations for the Strimzi Metrics Reporter. The Strimzi Metrics Reporter offers a lightweight solution for exposing Kafka metrics in Prometheus format, and avoiding complex mapping rules that can introduce latency.

To enable Strimzi Metrics Reporter, set the type to `strimziMetricsReporter`. The `allowList` configuration is a comma-separated list of regex patterns to filter the metrics that are collected. This defaults to `.*`, which allows all metrics.

NOTE Using `strimziMetricsReporter` is only supported in the Kafka brokers and controllers at the moment.

Example metrics configuration for Kafka

```
apiVersion: kafka.strimzi.io/v1
kind: Kafka
metadata:
  name: my-cluster
spec:
  kafka:
    # ...
    metricsConfig:
      type: strimziMetricsReporter
      values:
        allowList:
          key: ".*"
    # ...
```

2.9. jvmOptions

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

- Apache Kafka
- Apache Kafka Connect
- Apache Kafka MirrorMaker
- HTTP Bridge

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

- `Kafka.spec.kafka`
- `Kafka.spec.entityOperator.userOperator`
- `Kafka.spec.entityOperator.topicOperator`
- `Kafka.spec.cruiseControl`
- `KafkaNodePool.spec`
- `KafkaConnect.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 Kubernetes 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 6. Default heap settings for components

Component	Percent of available memory allocated to the heap	Maximum limit
Kafka	50%	5 GB
Kafka Connect	75%	None
MirrorMaker 2	75%	None
MirrorMaker	75%	None
Cruise Control	75%	None
HTTP 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:

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

2.11. Additional volumes

Strimzi supports specifying additional volumes and volume mounts in the following components:

- Kafka
- Kafka Connect
- HTTP Bridge
- Kafka MirrorMaker2
- Entity Operator
- Cruise Control
- Kafka Exporter
- User Operator
- Topic Operator

All additional mounted paths are located inside `/mnt` to ensure compatibility with future Kafka and Strimzi updates.

Supported Volume Types

- Secret
- ConfigMap
- EmptyDir
- PersistentVolumeClaims
- CSI Volumes
- Image Volumes

Example configuration for additional volumes

```
kind: Kafka
spec:
  kafka:
    # ...
    template:
      pod:
        volumes:
          - name: example-secret
            secret:
```

```
              secretName: secret-name
          - name: example-configmap
            configMap:
              name: config-map-name
          - name: temp
            emptyDir: {}
          - name: example-pvc-volume
            persistentVolumeClaim:
              claimName: myclaim
          - name: example-csi-volume
            csi:
              driver: csi.cert-manager.io
              readOnly: true
              volumeAttributes:
                csi.cert-manager.io/issuer-name: my-ca
                csi.cert-manager.io/dns-names:
${POD_NAME}.${POD_NAMESPACE}.svc.cluster.local
          - name: example-oci-plugin
            image:
              reference: my-registry.io/oci-artifacts/example-plugin:latest
      kafkaContainer:
        volumeMounts:
          - name: example-secret
            mountPath: /mnt/secret-volume
          - name: example-configmap
            mountPath: /mnt/cm-volume
          - name: temp
            mountPath: /mnt/temp
          - name: example-pvc-volume
            mountPath: /mnt/data
          - name: example-csi-volume
            mountPath: /mnt/certificate
          - name: example-oci-plugin
            mountPath: /mnt/example-plugin
```

You can use volumes to store files containing configuration values for a Kafka component and then load those values using a configuration provider. For more information, see Loading configuration values from external sources.

You can also use additional volumes to mount custom plugins:

- To include custom plugins in the User Operator and Topic Operator, set the `JAVA_CLASSPATH` environment variable to modify the Java classpath.
- To include custom plugins in the Kafka operands and Cruise Control, set the `CLASSPATH` environment variable to modify the Java classpath.
- To add Kafka Connect connectors, see Adding Kafka Connect connectors.
- Some plugins, such as the Tiered Storage plugins, may require their own classpath configuration.

Chapter 3. Kafka schema reference

Property	Property type	Description
spec	KafkaSpec	The specification of the Kafka cluster.
status	KafkaStatus	The status of the Kafka cluster.

Chapter 4. KafkaSpec schema reference

Used in: Kafka

Property	Property type	Description
kafka	KafkaClusterSpec	Configuration of the Kafka cluster.
zookeeper	ZookeeperClusterSpec	**The zookeeper property has been deprecated.** ZooKeeper-based Apache Kafka clusters are not supported anymore since Strimzi 0.46.0. As of Strimzi 0.46.0, ZooKeeper-based Apache Kafka clusters are not supported anymore and this option is ignored.
entityOperator	EntityOperatorSpec	Configuration of the Entity Operator.
clusterCa	CertificateAuthority	Configuration of the cluster certificate authority.
clientsCa	CertificateAuthority	Configuration of the clients certificate authority.
cruiseControl	CruiseControlSpec	Configuration for Cruise Control deployment. Deploys a Cruise Control instance when specified.
jmxTrans	JmxTransSpec	**The jmxTrans property has been deprecated.** JMXTrans is deprecated and related resources removed in Strimzi 0.35.0. As of Strimzi 0.35.0, JMXTrans is not supported anymore and this option is ignored.
kafkaExporter	KafkaExporterSpec	Configuration of the Kafka Exporter. Kafka Exporter can provide additional metrics, for example lag of consumer group at topic/partition.
maintenanceTimeWindows	string array	A list of time windows for maintenance tasks (that is, certificates renewal). Each time window is defined by a cron expression.

Chapter 5. KafkaClusterSpec schema reference

Used in: KafkaSpec

Full list of KafkaClusterSpec schema properties

Configures a Kafka cluster using the Kafka custom resource.

The config properties are one part of the overall configuration for the resource. Use the config properties to configure Kafka broker options as keys.

Example Kafka configuration

```
apiVersion: kafka.strimzi.io/v1
kind: Kafka
metadata:
  name: my-cluster
spec:
  kafka:
    version: 4.1.1
    metadataVersion: 4.1
    # ...
    config:
      auto.create.topics.enable: "false"
      offsets.topic.replication.factor: 3
      transaction.state.log.replication.factor: 3
      transaction.state.log.min.isr: 2
      default.replication.factor: 3
      min.insync.replicas: 2
# ...
```

The values can be one of the following JSON types:

- String
- Number
- Boolean

Exceptions

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

However, Strimzi 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

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`

NOTE	Strimzi supports only KRaft-based Kafka deployments. As a result, ZooKeeper-related configuration options are not supported.

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 Strimzi:

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

5.1. Configuring rack awareness and init container images

Rack awareness is enabled using the rack property. When rack awareness is enabled, Kafka broker pods use init container to collect the labels from the Kubernetes cluster nodes. The container image for this init container can be specified using the brokerRackInitImage property. If the brokerRackInitImage field is not provided, the images used are prioritized as follows:

1. Container image specified in STRIMZI_DEFAULT_KAFKA_INIT_IMAGE environment variable in the Cluster Operator configuration.
2. quay.io/strimzi/operator:0.50.0 container image.

Example brokerRackInitImage *configuration*

```
apiVersion: kafka.strimzi.io/v1
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, such as when your network does not allow access to the container registry used by Strimzi. In such cases, you should either copy the Strimzi images or build them from the source. Be aware that if the configured image is not compatible with Strimzi images, it might not work properly.

5.2. Logging

Kafka has its own preconfigured loggers:

Logger	Description	Default Level
`rootLogger`	Default logger for all classes	INFO
`kafka`	Logs Kafka node classes	INFO
`orgapachekafka`	Logs Kafka library classes	INFO
`requestlogger`	Logs client request details	WARN
`requestchannel`	Logs request handling in the broker	WARN
`controller`	Logs controller activity, such as leadership changes	INFO
`logcleaner`	Logs log compaction and cleanup processes	INFO
`statechange`	Logs broker and partition state transitions	INFO
`authorizer`	Logs access control decisions	INFO

Kafka uses the Apache `log4j2` logger implementation. Use the `logging` property to configure loggers and logger levels.

You can set log levels using either the `inline` or `external` logging configuration types.

Specify loggers and levels directly in the custom resource for inline configuration:

Example inline logging configuration

```
apiVersion: kafka.strimzi.io/v1
kind: Kafka
spec:
  # ...
  kafka:
    # ...
    logging:
      type: inline
      loggers:
        rootLogger.level: INFO
        logger.kafka.level: DEBUG
        logger.logcleaner.level: DEBUG
        logger.authorizer.level: TRACE
  # ...
```

You can define additional loggers by specifying the full class or package name using `logger.<name>.name`. For example, to configure logging for OAuth components inline:

Example custom inline loggers

```
# ...
logger.oauth.name: io.strimzi.kafka.oauth ①
logger.oauth.level: DEBUG ②
```

① Creates a logger for the `io.strimzi.kafka.oauth` package.

② Sets the logging level for the OAuth package.

Alternatively, you can reference an external `ConfigMap` containing a complete `log4j2.properties` file that defines your own log4j2 configuration, including loggers, appenders, and layout configuration:

Example external logging configuration

```
apiVersion: kafka.strimzi.io/v1
kind: Kafka
spec:
  # ...
  logging:
    type: external
    valueFrom:
      configMapKeyRef:
        # name and key are mandatory
        name: customConfigMap
        key: log4j2.properties
  # ...
```

Garbage collector (GC)

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

5.3. `KafkaClusterSpec` schema properties

Property	Property type	Description
version	string	The Kafka broker version. Defaults to the latest version. Consult the user documentation to understand the process required to upgrade or downgrade the version.
metadataVersion	string	Added in Strimzi 0.39.0. The KRaft metadata version used by the Kafka cluster. This property is ignored when running in ZooKeeper mode. If the property is not set, it defaults to the metadata version that corresponds to the `version` property.
replicas	integer	**The `replicas` property has been deprecated.** Use `KafkaNodePool` resources. Replicas are now configured in `KafkaNodePool` resources and this option is ignored.

Property	Property type	Description
image	string	The container image used for Kafka pods. If the property is not set, the default Kafka image version is determined based on the `version` configuration. The image names are specifically mapped to corresponding versions in the Cluster Operator configuration. Changing the Kafka image version does not automatically update the image versions for other components, such as Kafka Exporter.
listeners	`GenericKafkaListener` array	Configures listeners to provide access to Kafka brokers.

Property	Property type	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, zookeeper.metadata.migration.enable, client.quota.callback.static.kafka.admin., client.quota.callback.static.produce, client.quota.callback.static.fetch, client.quota.callback.static.storage.per.volume.limit.min.available., client.quota.callback.static.excluded.principal.name.list, prometheus.metrics.reporter. (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, broker.session.timeout.ms, broker.heartbeat.interval.ms, controller.socket.timeout.ms, controller.quorum.election.backoff.max.ms, controller.quorum.election.timeout.ms, controller.quorum.fetch.timeout.ms).

Property	Property type	Description
storage	EphemeralStorage, PersistentClaimStorage, JbodStorage	**The storage property has been deprecated.** Use KafkaNodePool resources. Storage is now configured in the KafkaNodePool resources and this option is ignored.
authorization	KafkaAuthorizationSimple, KafkaAuthorizationOpa, KafkaAuthorizationKeycloak, KafkaAuthorizationCustom	Authorization configuration for Kafka brokers.
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	Probe	Pod readiness checking.
jvmOptions	JvmOptions	JVM Options for pods.
jmxOptions	KafkaJmxOptions	JMX Options for Kafka brokers.
resources	ResourceRequirements	**The resources property has been deprecated.** Use KafkaNodePool custom resources to configure CPU and memory. CPU and memory resources to reserve.
metricsConfig	JmxPrometheusExporterMetrics, StrimziMetricsReporter	Metrics configuration.
logging	InlineLogging, ExternalLogging	Logging configuration for Kafka.
template	KafkaClusterTemplate	Template for Kafka cluster resources. The template allows users to specify how the Kubernetes resources are generated.
tieredStorage	TieredStorageCustom	Configure the tiered storage feature for Kafka brokers.
quotas	QuotasPluginKafka, QuotasPluginStrimzi	Quotas plugin configuration for Kafka brokers allows setting quotas for disk usage, produce/fetch rates, and more. Supported plugin types include kafka (default) and strimzi. If not specified, the default kafka quotas plugin is used.

Chapter 6. `GenericKafkaListener` schema reference

Used in: `KafkaClusterSpec`

Full list of `GenericKafkaListener` schema properties

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

Configure Kafka broker listeners using the `listeners` property in the `Kafka` resource. Listeners are defined as an array.

Example `Kafka` resource showing listener configuration

```
apiVersion: kafka.strimzi.io/v1
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
  #...
```

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.

6.1. Specifying a port number

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 Kubernetes

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

```
kubectl 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).

6.2. Specifying listener types

Set the type to `internal` for internal listeners. For external listeners, choose from `route`, `loadbalancer`, `nodeport`, or `ingress`. You can also configure a `cluster-ip` listener, which is an internal type used for building 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 hostname used by the bootstrap service using `GenericKafkaListenerConfigurationBootstrap` property. And you must also specify the hostnames used by the per-broker services using `GenericKafkaListenerConfigurationBroker` or `hostTemplate` properties. With the `hostTemplate` property, you don't need to specify the configuration for every broker.

Example `ingress` listener configuration

```
#...
spec:
  kafka:
    #...
    listeners:
      #...
      - name: external2
        port: 9095
        type: ingress
        tls: true
        authentication:
          type: tls
        configuration:
          hostTemplate: broker-{nodeId}.myingress.com
          bootstrap:
            host: bootstrap.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 Kubernetes. 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, Strimzi 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 Kubernetes 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
    #...
```

6.3. Configuring network policies to restrict listener access

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
# ...
```

6.4. `GenericKafkaListener` schema properties

Property	Property type	Description
name	string	Name of the listener. The name will be used to identify the listener and the related Kubernetes 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	Property type	Description
type	string (one of [ingress, internal, route, loadbalancer, cluster-ip, nodeport])	Type of the listener. The supported types are as follows: • `internal` type exposes Kafka internally only within the Kubernetes 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 Kubernetes 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. For `route` and `ingress` type listeners, TLS encryption must be always enabled.
authentication	`KafkaListenerAuthenticationTls`, `KafkaListenerAuthenticationScramSha512`, `KafkaListenerAuthenticationOAuth`, `KafkaListenerAuthenticationCustom`	Authentication configuration for this listener.
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.

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	Property type	Description
type	string	Must be tls.

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

Chapter 9. KafkaListenerAuthenticationOAuth schema reference

The type KafkaListenerAuthenticationOAuth has been deprecated. Please use KafkaListenerAuthenticationCustom instead.

Used in: GenericKafkaListener

The type KafkaListenerAuthenticationOAuth is supported only in the Strimzi API version(s) v1beta2.

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	Property type	Description
type	string	Must be oauth.
clientId	string	OAuth Client ID which the Kafka broker can use to authenticate against the authorization server and use the introspect endpoint URI.
clientSecret	GenericSecretSource	Link to Kubernetes Secret containing the OAuth client secret which the Kafka broker can use to authenticate against the authorization server and use the introspect endpoint URI.
validIssuerUri	string	URI of the token issuer used for authentication.
checkIssuer	boolean	Enable or disable issuer checking. By default issuer is checked using the value configured by validIssuerUri. Default value is 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.

Property	Property type	Description
jwksEndpointUri	string	URI of the JWKS certificate endpoint, which can be used for local JWT validation.
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.
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.
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`.
introspectionEndpointUri	string	URI of the token introspection endpoint which can be used to validate opaque non-JWT tokens.
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`.
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	Property type	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).
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.
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`.
validTokenType	string	Valid value for the `token_type` attribute returned by the Introspection Endpoint. No default value, and not checked by default.
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`.
tlsTrustedCertificates	`CertSecretSource` array	Trusted certificates for TLS connection to the OAuth server.
disableTlsHostnameVerification	boolean	Enable or disable TLS hostname verification. Default value is `false`.

Property	Property type	Description
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 Strimzi. Value is ignored.
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`).
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`.
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).

Property	Property type	Description
enableOauthBearer	boolean	Enable or disable OAuth authentication over SASL_OAUTHBEARER. Default value is `true`.
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.
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.
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.
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.
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.
clientGrantType	string	The grant type to use when making requests to the authorization server's token endpoint. Used for `OAuth over PLAIN` when `username` and `password` passed via SASL_PLAIN client authentication are passed on to the authorization server as `clientId` and `secret`.

Property	Property type	Description
enableMetrics	boolean	Enable or disable OAuth metrics. 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`.
includeAcceptHeader	boolean	Whether the Accept header should be set in requests to the authorization servers. The default value is `true`.
serverBearerTokenLocation	string	Path to the file on the local filesystem that contains a bearer token to be used instead of client ID and secret when authenticating to authorization server.
userNamePrefix	string	The prefix to use with the value of `userNameClaim` to construct the user ID. This only takes effect if `userNameClaim` is specified and the value is present for the claim. When used in combination with `fallbackUserNameClaims`, it ensures consistent mapping of usernames and client IDs into the same user ID space and prevents name collisions.

Chapter 10. GenericSecretSource schema reference

Used in: KafkaClientAuthenticationOAuth, KafkaListenerAuthenticationCustom, KafkaListenerAuthenticationOAuth

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

Chapter 11. `CertSecretSource` schema reference

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

Property	Property type	Description
secretName	string	The name of the Secret containing the certificate.
certificate	string	The name of the file certificate in the secret.
pattern	string	Pattern for the certificate files in the secret. Use the *glob syntax* for the pattern. All files in the secret that match the pattern are used.

Chapter 12. KafkaListenerAuthenticationCustom schema reference

Used in: GenericKafkaListener

Full list of KafkaListenerAuthenticationCustom schema properties

Configures custom authentication for listeners.

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 ;
    template:
      pod:
        volumes:
          - name: example-secret
            secret:
              secretName: example
      kafkaContainer:
        volumeMounts:
          - name: example-secret
            mountPath: /mnt/secret-volume
```

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

Secrets are mounted to the `/mnt` directory in the Kafka broker nodes' containers. For example, the mounted secret (`example`) in the example configuration would be located at `/mnt/secret-volume`.

12.1. Configuring customized TLS Client Authentication

You can also use the `custom` authentication to configure customized TLS client authentication. This allows configuration options that are not permissible with `type: tls` authentication. For example, it's possible to configure a custom truststore with multiple trusted CAs or options such as `ssl.principal.mapping.rules`.

Example custom TLS Client Authentication configuration

```
spec:
  kafka:
    listeners:
      - name: tls
        port: 9093
        tls: true
        type: internal
        authentication:
          type: custom
          sasl: false
          listenerConfig:
            ssl.client.auth: required
            ssl.principal.mapping.rules: RULE:^CN=(.*?),(.*)$/$1@my-cluster.com/
            ssl.truststore.location: /mnt/my-truststore/ca.crt
            ssl.truststore.type: PEM
    template:
      pod:
        volumes:
          - name: my-truststore
            secret:
              secretName: custom-truststore
      kafkaContainer:
        volumeMounts:
          - name: my-truststore
            mountPath: /mnt/my-truststore
```

12.2. Setting a custom 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 Strimzi 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 Strimzi.

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

A custom OAuth principal builder might be identical or very similar to the Strimzi OAuth principal builder.

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 Strimzi.

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
        ...
    }
}
```

12.3. 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	Property type	Description
type	string	Must be custom.
sasl	boolean	Enable or disable SASL on this listener.
listenerConfig	map	Configuration to be used for a specific listener. All values are prefixed with listener.name.<listener_name>.
secrets	GenericSecretSource array	**The secrets property has been deprecated.** Please use the template section to configure additional volumes instead. Secrets to be mounted to /opt/kafka/custom-authn-secrets/custom-listener-<listener_name>-<port>/<secret_name>.

Chapter 13. GenericKafkaListenerConfiguration schema reference

Used in: GenericKafkaListener

Full list of GenericKafkaListenerConfiguration schema properties

Configures Kafka listeners.

13.1. Providing your own listener certificates

The brokerCertChainAndKey property is for listeners that have TLS encryption enabled only. Use this property to provide your own Kafka listener certificates.

Example loadbalancer listener configuration to provide certificates

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

When the certificate or key in the brokerCertChainAndKey secret is updated, the operator automatically detects it in the next reconciliation and triggers a rolling update of the Kafka brokers to reload the certificate.

NOTE The private key referenced in brokerCertChainAndKey must be in an unencrypted PKCS #8 format. If you obtain the certificate is obtained using the cert-manager project, you can set key encoding to use PKCS#8.

13.2. Avoiding hops to other nodes

The externalTrafficPolicy property is used with loadbalancer and nodeport listeners. When exposing Kafka outside of Kubernetes, 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.

Example loadbalancer listener configuration avoiding hops

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

```
  - name: external3
    port: 9094
    type: loadbalancer
    tls: true
    configuration:
      externalTrafficPolicy: Local
# ...
```

13.3. Providing CIDR source ranges for a loadbalancer

The `loadBalancerSourceRanges` property is for `loadbalancer` listeners only. When exposing Kafka outside of Kubernetes, use CIDR (Classless Inter-Domain Routing) source ranges in addition to labels and annotations to customize how a service is created.

Example `loadbalancer` listener configuration to provide source ranges

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

13.4. Specifying a preferred node port address type

The `preferredNodePortAddressType` property is for `nodeport` listeners only. Use this 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, Strimzi proceeds through the types in the standard order of priority:

- ExternalDNS
- ExternalIP
- Hostname
- InternalDNS
- InternalIP

Example `nodeport` listener using a preferred node port address type

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

13.5. Using fully-qualified DNS names

The `useServiceDnsDomain` property is for `internal` and `cluster-ip` listeners. It defines whether the fully-qualified DNS names that include the cluster service suffix (usually `.cluster.local`) are used.

- Set to `false` (default) to generate advertised addresses without the service suffix; for example, `my-cluster-kafka-0.my-cluster-kafka-brokers.myproject.svc`.
- Set to `true` to generate advertised addresses with the service suffix; for example, `my-cluster-kafka-0.my-cluster-kafka-brokers.myproject.svc.cluster.local`.

Example `internal` listener using the service DNS domain

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

13.6. Specifying the hostname

To specify the hostname used for the bootstrap resource or brokers, use the `host` property. The `host` property is for `route` and `ingress` listeners only.

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

Example `ingress` listener with host configuration

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

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

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

Example `route` listener with host configuration

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

Instead of specifying the `host` property for every broker, you can also use a `hostTemplate` to generate them automatically. The `hostTemplate` supports the following variables:

- The `{nodeId}` variable is replaced with the ID of the Kafka node to which the template is applied.
- The `{nodePodName}` variable is replaced with the Kubernetes pod name for the Kafka node where the template is applied.

The `hostTemplate` property applies only to per-broker values. The bootstrap `host` property must always be specified.

Example `ingress` listener with `hostTemplate` configuration

```
#...
spec:
  kafka:
    #...
    listeners:
      #...
      - name: external2
        port: 9095
        type: ingress
        tls: true
        authentication:
          type: tls
        configuration:
          hostTemplate: broker-{nodeId}.myingress.com
          bootstrap:
            host: bootstrap.myingress.com
  #...
```

13.7. Overriding assigned node ports

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

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

Example `nodeport` listener configuration with overrides for node ports

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

```
        nodePort: 32002
# ...
```

13.8. Requesting a specific loadbalancer IP address

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

Example `loadbalancer` listener with specific IP addresses

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

13.9. Adding listener annotations to Kubernetes resources

Use the `annotations` property to add annotations to Kubernetes resources related to the listeners. These annotations can be used, for example, to instrument DNS tooling such as External DNS, which automatically assigns DNS names to the loadbalancer services.

Example `loadbalancer` listener using `annotations`

```
# ...
listeners:
  #...
  - name: external3
    port: 9094
    type: loadbalancer
    tls: true
    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"
# ...
```

13.10. `GenericKafkaListenerConfiguration` schema properties

Property	Property type	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.
class	string	Configures a specific class for `Ingress` and `LoadBalancer` that defines which controller is used. If not specified, the default controller is used. • For an `ingress` listener, the operator uses this property to set the `ingressClassName` property in the `Ingress` resources. • For a `loadbalancer` listener, the operator uses this property to set the `loadBalancerClass` property in the `Service` resources. For `ingress` and `loadbalancer` listeners only.

Property	Property type	Description
externalTrafficPolicy	string (one of [Local, Cluster])	Specifies whether the service routes external traffic to cluster-wide or node-local 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, Kubernetes uses `Cluster` as the default. For `loadbalancer` or `nodeport` listeners only.
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 loadbalancer 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. For `loadbalancer` listeners only.
bootstrap	`GenericKafkaListenerConfigurationBootstrap`	Bootstrap configuration.
brokers	`GenericKafkaListenerConfigurationBroker` array	Per-broker configurations.

Property	Property type	Description
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, Kubernetes 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, Kubernetes 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` listener.
finalizers	string array	A list of finalizers 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. For `loadbalancer` listeners only.

Property	Property type	Description
useServiceDnsDomain	boolean	Configures whether the Kubernetes service DNS domain should be included in the generated addresses. • If set to `false`, the generated addresses do not contain the service DNS domain suffix. For example, `my-cluster-kafka-0.my-cluster-kafka-brokers.myproject.svc`. • If set to `true`, the generated addresses contain the service DNS domain suffix. For example, `my-cluster-kafka-0.my-cluster-kafka-brokers.myproject.svc.cluster.local`. The default is `.cluster.local`, but this is customizable using the environment variable `KUBERNETES_SERVICE_DNS_DOMAIN`. For `internal` and `cluster-ip` listeners only.
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.
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.

Property	Property type	Description
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 are used in the following order (the first one found is used): • `ExternalDNS` • `ExternalIP` • `InternalDNS` • `InternalIP` • `Hostname` This property 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.For `nodeport` listeners only.
publishNotReadyAddresses	boolean	Configures whether the service endpoints are considered "ready" even if the Pods themselves are not. Defaults to `false`. This field can not be used with `internal` listeners.
hostTemplate	string	Configures the template for generating the hostnames of the individual brokers. Valid placeholders that you can use in the template are `{nodeId}` and `{nodePodName}`.
advertisedHostTemplate	string	Configures the template for generating the advertised hostnames of the individual brokers. Valid placeholders that you can use in the template are `{nodeId}` and `{nodePodName}`.
allocateLoadBalancerNodePorts	boolean	Configures whether to allocate NodePort automatically for the `Service` with type `LoadBalancer`. This is a one to one with the `spec.allocateLoadBalancerNodePorts` configuration in the `Service` type For `loadbalancer` listeners only.

Chapter 14. `CertAndKeySecretSource` schema reference

Used in: `GenericKafkaListenerConfiguration`, `KafkaClientAuthenticationTls`

Property	Property type	Description
secretName	string	The name of the Secret containing the certificate.
certificate	string	The name of the file certificate in the Secret.
key	string	The name of the private key in the secret. The private key must be in unencrypted PKCS #8 format. For more information, see RFC 5208: https://datatracker.ietf.org/doc/html/rfc5208.

Chapter 15. GenericKafkaListenerConfigurationBootstrap schema reference

Used in: GenericKafkaListenerConfiguration

Full list of GenericKafkaListenerConfigurationBootstrap schema properties

Configures bootstrap service settings for listeners.

Example configuration for the host, nodePort, loadBalancerIP, and annotations properties is shown in the GenericKafkaListenerConfiguration schema section.

15.1. Specifying alternative bootstrap addresses

To specify alternative names for the bootstrap address, use the alternativeNames property. This property is applicable to all types of listeners. The names are added to the broker certificates and can be used for TLS hostname verification.

Example route listener configuration with additional bootstrap addresses

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

15.2. GenericKafkaListenerConfigurationBootstrap schema properties

Property	Property type	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.

Property	Property type	Description
host	string	Specifies the hostname used for the bootstrap resource. For `route` (optional) or `ingress` (required) listeners only. Ensure the hostname resolves to the Ingress endpoints; no validation is performed by Strimzi.
nodePort	integer	Node port for the bootstrap service. For `nodeport` listeners only.
loadBalancerIP	string	The loadbalancer is requested with the IP address specified in this property. This feature depends on whether the underlying cloud provider supports specifying the `loadBalancerIP` when a load balancer is created. This property is ignored if the cloud provider does not support the feature. For `loadbalancer` listeners only.
annotations	map	Annotations added to `Ingress`, `Route`, or `Service` resources. You can use this property to configure DNS providers such as External DNS. For `loadbalancer`, `nodeport`, `route`, or `ingress` listeners only.
labels	map	Labels added to `Ingress`, `Route`, or `Service` resources. For `loadbalancer`, `nodeport`, `route`, or `ingress` listeners only.
externalIPs	string array	External IPs associated to the nodeport service. These IPs are used by clients external to the Kubernetes cluster to access the Kafka brokers. This property is helpful when `nodeport` without `externalIP` is not sufficient. For example on bare-metal Kubernetes clusters that do not support Loadbalancer service types. For `nodeport` listeners only.

Chapter 16. GenericKafkaListenerConfigurationBroker schema reference

Used in: GenericKafkaListenerConfiguration

Full list of GenericKafkaListenerConfigurationBroker schema properties

Configures broker settings for listeners.

Example configuration for the host, nodePort, loadBalancerIP, and annotations properties is shown in the GenericKafkaListenerConfiguration schema section.

16.1. Overriding advertised addresses for brokers

By default, Strimzi 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 Strimzi 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. Strimzi 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
    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
# ...
```

Instead of specifying the advertisedHost field for every broker, you can also use an

`advertisedHostTemplate` to generate them automatically. The `advertisedHostTemplate` supports the following variables:

- The `{nodeId}` variable is replaced with the ID of the Kafka node to which the template is applied.
- The `{nodePodName}` variable is replaced with the Kubernetes pod name for the Kafka node where the template is applied.

Example `route` listener with `advertisedHostTemplate` configuration

```
listeners:
  #...
  - name: external1
    port: 9094
    type: route
    tls: true
    configuration:
      advertisedHostTemplate: example.hostname.{nodeId}
# ...
```

16.2. `GenericKafkaListenerConfigurationBroker` schema properties

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

Property	Property type	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` 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.
externalIPs	string array	External IPs associated to the nodeport service. These IPs are used by clients external to the Kubernetes cluster to access the Kafka brokers. This field is helpful when `nodeport` without `externalIP` is not sufficient. For example on bare-metal Kubernetes clusters that do not support Loadbalancer service types. This field can only be used with `nodeport` type listener.

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	Property type	Description
id	integer	Storage identification number. It is mandatory only for storage volumes defined in a storage of type 'jbod'.
sizeLimit	string	When type=ephemeral, defines the total amount of local storage required for this EmptyDir volume (for example 1Gi).
type	string	Must be ephemeral.
kraftMetadata	string (one of [shared])	Specifies whether this volume should be used for storing KRaft metadata. This property is optional. When set, the only currently supported value is shared. At most one volume can have this property set.

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	Property type	Description
id	integer	Storage identification number. It is mandatory only for storage volumes defined in a storage of type 'jbod'.
type	string	Must be persistent-claim.
size	string	When type=persistent-claim, defines the size of the persistent volume claim, such as 100Gi. Mandatory when type=persistent-claim.
kraftMetadata	string (one of [shared])	Specifies whether this volume should be used for storing KRaft metadata. This property is optional. When set, the only currently supported value is shared. At most one volume can have this property set.
class	string	The storage class to use for dynamic volume allocation.
volumeAttributesClass	string	Specifies VolumeAttributeClass name for dynamically configuring storage attributes.
selector	map	Specifies a specific persistent volume to use. It contains key:value pairs representing labels for selecting such a volume.
deleteClaim	boolean	Specifies whether the persistent volume claim is deleted when a Kafka node is deleted. Optional. Defaults to false.
overrides	PersistentClaimStorageOverride array	**The overrides property has been deprecated.** The storage overrides for individual brokers are not supported anymore since Strimzi 0.46.0. As of Strimzi 0.46.0, the storage overrides for individual brokers are not supported anymore and this option is ignored.

Chapter 19. PersistentClaimStorageOverride schema reference

Used in: PersistentClaimStorage

Property	Property type	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	Property type	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

Configures the Kafka custom resource to use simple authorization and define Access Control Lists (ACLs).

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

Strimzi uses Kafka's built-in StandardAuthorizer authorization plugin.

Set the type property in the authorization section to the value simple, and configure a list of super users. Super users are always allowed without querying ACL rules.

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

Example simple authorization configuration

```
apiVersion: kafka.strimzi.io/v1
kind: Kafka
metadata:
  name: my-cluster
  namespace: myproject
spec:
  kafka:
    # ...
    authorization:
      type: simple
      superUsers:
        - CN=user-1
        - user-2
        - CN=user-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.

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

Chapter 22. KafkaAuthorizationOpa schema reference

The type KafkaAuthorizationOpa has been deprecated. Please use KafkaAuthorizationCustom instead.

Used in: KafkaClusterSpec

Full list of KafkaAuthorizationOpa schema properties

Configures the Kafka custom resource to use Open Policy Agent authorization.

To use Open Policy Agent authorization, set the type property in the authorization section to the value opa, and configure OPA properties as required. Strimzi uses the 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.

The type: opa authorization is now deprecated and will be removed in the future. If you want to use the Open Policy Agent authorizer, you should use the type: custom authorization.

Example Open Policy Agent authorizer configuration using the type: custom API

```
apiVersion: kafka.strimzi.io/v1
kind: Kafka
metadata:
  name: my-cluster
  namespace: myproject
spec:
  kafka:
    # ...
    authorization:
      type: custom
      authorizerClass: org.openpolicyagent.kafka.OpaAuthorizer
      superUsers:
        - CN=user-1
        - user-2
        - CN=user-3
    config:
      # OPA authorization options
      opa.authorizer.url: http://opa:8181/v1/data/kafka/allow
      opa.authorizer.cache.expire.after.seconds: 60
      opa.authorizer.allow.on.error: false
      opa.authorizer.cache.initial.capacity: 1000
      opa.authorizer.cache.maximum.size: 10000
    # ...
```

22.1. KafkaAuthorizationOpa schema properties

The type KafkaAuthorizationOpa is supported only in the Strimzi API version(s) v1beta2.

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	Property type	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`.
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

The type KafkaAuthorizationKeycloak has been deprecated. Please use KafkaAuthorizationCustom instead.

Used in: KafkaClusterSpec

The type KafkaAuthorizationKeycloak is supported only in the Strimzi API version(s) v1beta2.

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	Property type	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 Keycloak 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.
grantsMaxIdleTimeSeconds	integer	The time, in seconds, after which an idle grant can be evicted from the cache. The default value is 300.

Property	Property type	Description
grantsGcPeriodSeconds	integer	The time, in seconds, between consecutive runs of a job that cleans stale grants from the cache. The default value is 300.
grantsAlwaysLatest	boolean	Controls whether the latest grants are fetched for a new session. When enabled, grants are retrieved from Keycloak 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`.

Chapter 24. KafkaAuthorizationCustom schema reference

Used in: KafkaClusterSpec

Full list of KafkaAuthorizationCustom schema properties

Configures the Kafka custom resource to use a custom authorizer and 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 specify an authorizer class that implements the org.apache.kafka.server.authorizer.Authorizer interface to support custom ACLs. Set the type property in the authorization section to the value custom, and configure a list of super users. Super users are always allowed without querying ACL rules. Add additional configuration for initializing the custom authorizer using Kafka.spec.kafka.config.

Example custom *authorization configuration*

```
apiVersion: kafka.strimzi.io/v1
kind: Kafka
metadata:
  name: my-cluster
  namespace: myproject
spec:
  kafka:
    # ...
    authorization:
      type: custom
      authorizerClass: io.mycompany.CustomAuthorizer
      superUsers:
        - CN=user-1
        - user-2
        - CN=user-3
    # ...
    config:
      authorization.custom.property1=value1
      authorization.custom.property2=value2
    # ...
```

NOTE The super.user configuration option in the config property in Kafka.spec.kafka is ignored. Designate super users in the authorization property instead.

24.1. Adding custom authorizer JAR files to the container image

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

You can add them by building Strimzi from the source-code. The Strimzi 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/artifacts/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 Strimzi build process.

Alternatively, you can add the JARs to an existing Strimzi container image:

```
FROM quay.io/strimzi/kafka:0.50.0-kafka-4.1.1
USER root:root
COPY ./my-authorizer/ /opt/kafka/libs/
USER 1001
```

24.2. Using custom authorizers with OAuth authentication

When using `oauth` authentication with a `groupsClaim` configuration to extract user group information from JWT tokens, group information can be used in custom authorization calls. Groups are accessible through the `OAuthKafkaPrincipal` object during custom authorization calls, 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	Property type	Description
type	string	Must be `custom`.
authorizerClass	string	Authorization implementation class, which must be available in classpath.
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 Kubernetes 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 Kubernetes versions), which contains the name of the availability zone in which the Kubernetes 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 Kubernetes node labels, see Well-Known Labels, Annotations and Taints. Consult your Kubernetes 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, Strimzi 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/v1
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, Strimzi adds a node affinity rule to ensure Kafka nodes are

scheduled on Kubernetes workers that include the specified `topologyKey` label. It does not configure additional rules to distribute nodes across racks or zones. Depending on your exact Kubernetes and Kafka configurations, you should add additional `affinity` rules or configure `topologySpreadConstraints` for Kafka to make sure the nodes are properly distributed across 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/v1
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.

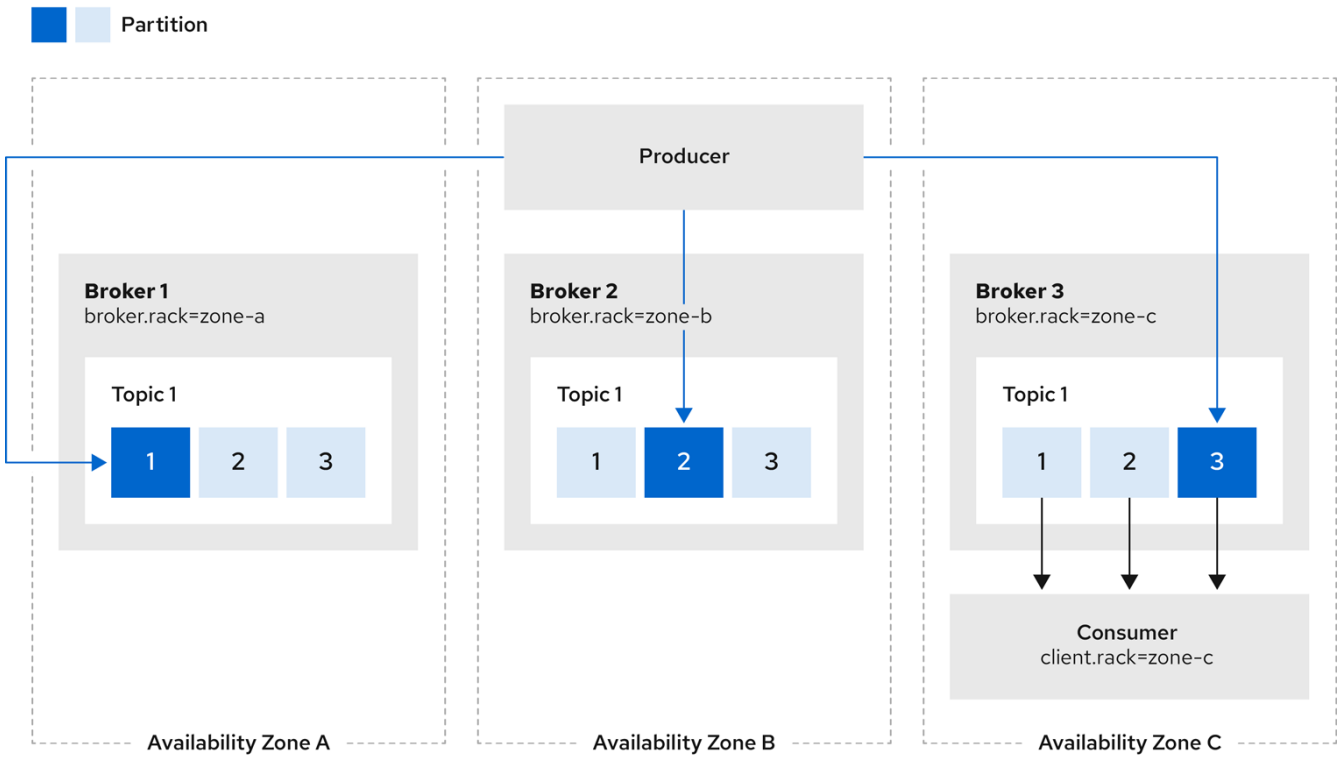


Figure 1. Example showing client consuming from replicas in the same availability zone

You can also configure Kafka Connect, MirrorMaker 2 and HTTP 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 Strimzi, 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/v1
kind: KafkaConnect
# ...
spec:
  # ...
  rack:
    topologyKey: topology.kubernetes.io/zone
  # ...
```

When deploying MirrorMaker 2 using Strimzi, 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/v1
```

```
kind: KafkaMirrorMaker2
# ...
spec:
  # ...
  rack:
    topologyKey: topology.kubernetes.io/zone
  # ...
```

When deploying HTTP Bridge using Strimzi, you can use the `rack` section in the `KafkaBridge` custom resource to automatically configure the `client.rack` option.

Example `rack` configuration for HTTP Bridge

```
apiVersion: kafka.strimzi.io/v1
kind: KafkaBridge
# ...
spec:
  # ...
  rack:
    topologyKey: topology.kubernetes.io/zone
  # ...
```

25.3. Rack schema properties

Property	Property type	Description
topologyKey	string	A key that matches labels assigned to the Kubernetes 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, TlsSidecar, ZookeeperClusterSpec

Property	Property type	Description
initialDelaySeconds	integer	The initial delay before first the health is first checked. Default to 15 seconds. Minimum value is 0.
timeoutSeconds	integer	The timeout for each attempted health check. Default to 5 seconds. Minimum value is 1.
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.
failureThreshold	integer	Minimum consecutive failures for the probe to be considered failed after having succeeded. Defaults to 3. Minimum value is 1.

Chapter 27. JvmOptions schema reference

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

Property	Property type	Description
-XX	map	A map of -XX options to the JVM.
-Xmx	string	-Xmx option to to the JVM.
-Xms	string	-Xms 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	Property type	Description
name	string	The system property name.
value	string	The system property value.

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, 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

```
apiVersion: kafka.strimzi.io/v1
kind: Kafka
metadata:
  name: my-cluster
spec:
  kafka:
    # ...
    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

```
apiVersion: kafka.strimzi.io/v1
kind: Kafka
metadata:
  name: my-cluster
spec:
  kafka:
    # ...
    jmxOptions: {}
    # ...
```

NOTE The `jmxOptions` configuration enables direct access to Java Management Extensions (JMX) from Kafka components. It is not required for the Prometheus JMX Exporter, which collects and converts JMX metrics to Prometheus metrics without direct JMX access.

Additional resources

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

29.1. `KafkaJmxOptions` schema properties

Property	Property type	Description
authentication	`KafkaJmxAuthenticationPassword`	Authentication configuration for connecting to the JMX port.

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	Property type	Description
type	string	Must be password.

Chapter 31. JmxPrometheusExporterMetrics schema reference

Used in: CruiseControlSpec, KafkaBridgeSpec, KafkaClusterSpec, KafkaConnectSpec, KafkaMirrorMaker2Spec, ZookeeperClusterSpec

The type property is a discriminator that distinguishes use of the JmxPrometheusExporterMetrics type from StrimziMetricsReporter. It must have the value jmxPrometheusExporter for the type JmxPrometheusExporterMetrics.

Property	Property type	Description
type	string	Must be jmxPrometheusExporter.
valueFrom	ExternalConfigurationReference	ConfigMap entry where the Prometheus JMX Exporter configuration is stored.

Chapter 32. ExternalConfigurationReference schema reference

Used in: ExternalLogging, JmxPrometheusExporterMetrics

Property	Property type	Description
configMapKeyRef	ConfigMapKeySelector	Reference to the key in the ConfigMap containing the configuration.

Chapter 33. StrimziMetricsReporter schema reference

Used in: CruiseControlSpec, KafkaBridgeSpec, KafkaClusterSpec, KafkaConnectSpec, KafkaMirrorMaker2Spec, ZookeeperClusterSpec

The type property is a discriminator that distinguishes use of the StrimziMetricsReporter type from JmxPrometheusExporterMetrics. It must have the value strimziMetricsReporter for the type StrimziMetricsReporter.

Property	Property type	Description
type	string	Must be strimziMetricsReporter.
values	StrimziMetricsReporterValues	Configuration values for the Strimzi Metrics Reporter.

Chapter 34. StrimziMetricsReporterValues schema reference

Used in: StrimziMetricsReporter

Property	Property type	Description
allowList	string array	A list of regex patterns to filter the metrics to collect. Should contain at least one element.

Chapter 35. InlineLogging schema reference

Used in: CruiseControlSpec, EntityTopicOperatorSpec, EntityUserOperatorSpec, KafkaBridgeSpec, KafkaClusterSpec, KafkaConnectSpec, KafkaMirrorMaker2Spec, 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	Property type	Description
type	string	Must be inline.
loggers	map	A Map from logger name to logger level.

Chapter 36. ExternalLogging schema reference

Used in: CruiseControlSpec, EntityTopicOperatorSpec, EntityUserOperatorSpec, KafkaBridgeSpec, KafkaClusterSpec, KafkaConnectSpec, KafkaMirrorMaker2Spec, 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	Property type	Description
type	string	Must be external.
valueFrom	ExternalConfigurationReference	ConfigMap entry where the logging configuration is stored.

Chapter 37. KafkaClusterTemplate schema reference

Used in: KafkaClusterSpec

Property	Property type	Description
statefulset	StatefulSetTemplate	**The statefulset property has been deprecated.** Support for StatefulSets was removed in Strimzi 0.35.0. 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 Kubernetes.
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 Kubernetes.
persistentVolumeClaim	ResourceTemplate	Template for all Kafka PersistentVolumeClaims.
podDisruptionBudget	PodDisruptionBudgetTemplate	Template for Kafka PodDisruptionBudget.
kafkaContainer	ContainerTemplate	Template for the Kafka broker container.
initContainer	ContainerTemplate	Template for the Kafka init container.
clusterCaCert	ResourceTemplate	Template for Secret with Kafka Cluster certificate public key.
serviceAccount	ResourceTemplate	Template for the Kafka service account.
jmxSecret	ResourceTemplate	Template for Secret of the Kafka Cluster JMX authentication.

Property	Property type	Description
clusterRoleBinding	`ResourceTemplate`	Template for the Kafka ClusterRoleBinding.
podSet	`ResourceTemplate`	Template for Kafka `StrimziPodSet` resource.

Chapter 38. `StatefulSetTemplate` schema reference

Used in: `KafkaClusterTemplate`, `ZookeeperClusterTemplate`

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

Chapter 39. 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 Strimzi and cannot be configured.

39.1. MetadataTemplate schema properties

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

Chapter 40. PodTemplate schema reference

Used in: CruiseControlTemplate, EntityOperatorTemplate, JmxTransTemplate, KafkaBridgeTemplate, KafkaClusterTemplate, KafkaConnectTemplate, KafkaExporterTemplate, 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
    hostAliases:
      - ip: "192.168.1.86"
        hostnames:
        - "my-host-1"
        - "my-host-2"
    #...
```

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.

40.1. PodTemplate schema properties

Property	Property type	Description
metadata	MetadataTemplate	Metadata applied to the resource.

Property	Property type	Description
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.
securityContext	PodSecurityContext	Configures pod-level security attributes and common container settings.
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.
tolerations	Toleration array	The pod's tolerations.
topologySpreadConstraints	TopologySpreadConstraint array	The pod's topology spread constraints.
priorityClassName	string	The name of the priority class used to assign priority to the pods.
schedulerName	string	The name of the scheduler used to dispatch this `Pod`. If not specified, the default scheduler will be used.
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.

Property	Property type	Description
dnsPolicy	string (one of [ClusterFirstWithHostNet, ClusterFirst, Default, None])	The pod's DNSPolicy. Defaults to `ClusterFirst`. Valid values are `ClusterFirstWithHostNet`, `ClusterFirst`, `Default` or `None`.
dnsConfig	PodDNSConfig	The pod's DNSConfig. If specified, it will be merged to the generated DNS configuration based on the DNSPolicy.
enableServiceLinks	boolean	Indicates whether information about services should be injected into Pod's environment variables.
tmpDirSizeLimit	string	Defines the total amount of pod memory allocated for the temporary `EmptyDir` volume `/tmp`. Specify the allocation in memory units, for example, `100Mi` for 100 mebibytes. Default value is `5Mi`. The `/tmp` volume is backed by pod memory, not disk storage, so avoid setting a high value as it consumes pod memory resources.
volumes	`AdditionalVolume` array	Additional volumes that can be mounted to the pod.
hostUsers	boolean	Use the host user namespace. Optional. Defaults to `true`. When `true` or not set, the pod runs in the host user namespace. This is required when the pod needs features available only in the host namespace, such as loading kernel modules with `CAP_SYS_MODULE`.When set to `false`, the pod runs in a new user namespace. Setting `false` helps mitigate container breakout vulnerabilities and allows containers to run as `root` without granting `root` privileges on the host. This property is alpha-level in Kubernetes and is supported only by Kubernetes clusters that enable the `UserNamespacesSupport` feature.

Chapter 41. `AdditionalVolume` schema reference

Used in: `PodTemplate`

Property	Property type	Description
name	string	Name to use for the volume. Required.
secret	SecretVolumeSource	`Secret` to use to populate the volume.
configMap	ConfigMapVolumeSource	`ConfigMap` to use to populate the volume.
emptyDir	`EmptyDirVolume`	`EmptyDir` to use to populate the volume.
persistentVolumeClaim	PersistentVolumeClaimVolumeSource	`PersistentVolumeClaim` object to use to populate the volume.
csi	CSIVolumeSource	`CSIVolumeSource` object to use to populate the volume.
image	ImageVolumeSource	`ImageVolumeSource` object to use to populate the volume.

Chapter 42. `EmptyDirVolume` schema reference

Used in: `AdditionalVolume`

Property	Property type	Description
medium	string (one of [Memory])	Medium represents the type of storage medium should back this volume. Valid values are unset or `Memory`. When not set, it will use the node's default medium.
sizeLimit	string	The total amount of local storage required for this EmptyDir volume (for example 1Gi).

Chapter 43. InternalServiceTemplate schema reference

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

Property	Property type	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, Kubernetes 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, Kubernetes will choose the default value based on the ipFamilyPolicy setting.

Chapter 44. `ResourceTemplate` schema reference

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

Property	Property type	Description
metadata	`MetadataTemplate`	Metadata applied to the resource.

Chapter 45. PodDisruptionBudgetTemplate schema reference

Used in: CruiseControlTemplate, EntityOperatorTemplate, KafkaBridgeTemplate, KafkaClusterTemplate, KafkaConnectTemplate, KafkaExporterTemplate, ZookeeperClusterTemplate

Full list of PodDisruptionBudgetTemplate schema properties

A PodDisruptionBudget is a Kubernetes resource that helps maintain high availability during planned maintenance or upgrades. It defines the minimum number of pods that must remain available at all times.

Strimzi creates one PodDisruptionBudget per component:

- Kafka cluster
- Kafka Connect
- MirrorMaker 2
- HTTP Bridge
- Entity Operator
- Kafka Exporter
- Cruise Control

Each budget applies across all pods deployed for that component.

A single PodDisruptionBudget applies to all associated node pool pods managed by the Kafka resource. This is true regardless of how many StrimziPodSet resources are created for that component. Disruption limits are applied at the component level, not per node pool.

By default, only one pod can be unavailable at a time. You can modify this limit using the maxUnavailable property.

StrimziPodSet custom resources are managed by a controller that does not use maxUnavailable directly. Instead, the value is automatically converted to minAvailable, which serves the same purpose.

For example:

- 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.

The above examples also applies for the EntityOperator, Cruise Control, KafkaExporter deployments.

Example `PodDisruptionBudget` template configuration

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

Custom disruption budget behavior

To define custom disruption budget behavior, disable automatic `PodDisruptionBudget` generation. Set the `STRIMZI_POD_DISRUPTION_BUDGET_GENERATION` environment variable to `false` in the Cluster Operator.

You can then create your own `PodDisruptionBudget` resources. For example, apply separate budgets to specific node pools using label selectors with `StrimziPodSet` resources.

45.1. `PodDisruptionBudgetTemplate` schema properties

Property	Property type	Description
metadata	`MetadataTemplate`	Metadata to apply to the `PodDisruptionBudgetTemplate` resource.
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 46. ContainerTemplate schema reference

Used in: CruiseControlTemplate, EntityOperatorTemplate, JmxTransTemplate, KafkaBridgeTemplate, KafkaClusterTemplate, KafkaConnectTemplate, KafkaExporterTemplate, 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 Strimzi and should be avoided. If you set a custom environment variable that is already in use by Strimzi, it is ignored and a warning is recorded in the log.

46.1. ContainerTemplate schema properties

Property	Property type	Description
env	ContainerEnvVar array	Environment variables which should be applied to the container.
securityContext	SecurityContext	Security context for the container.
volumeMounts	VolumeMount array	Additional volume mounts which should be applied to the container.

Chapter 47. `ContainerEnvVar` schema reference

Used in: `ContainerTemplate`

Property	Property type	Description
name	string	The environment variable key.
value	string	The environment variable value.
valueFrom	`ContainerEnvVarSource`	Reference to the secret or config map property to which the environment variable is set.

Chapter 48. `ContainerEnvVarSource` schema reference

Used in: `ContainerEnvVar`

Property	Property type	Description
secretKeyRef	SecretKeySelector	Reference to a key in a secret.
configMapKeyRef	ConfigMapKeySelector	Reference to a key in a config map.

Chapter 49. TieredStorageCustom schema reference

Used in: KafkaClusterSpec

Full list of TieredStorageCustom schema properties

Enables custom tiered storage for Kafka.

If you want to use custom tiered storage, you must first add a tiered storage for Kafka plugin to the Strimzi image by building a custom container image.

Custom tiered storage configuration enables the use of a custom RemoteStorageManager configuration. RemoteStorageManager is a Kafka interface for managing the interaction between Kafka and remote tiered storage.

If custom tiered storage is enabled, Strimzi uses the TopicBasedRemoteLogMetadataManager for Remote Log Metadata Management (RLMM).

NOTE	Tiered storage is a production-ready feature in Kafka since version 3.9.0, and it is also supported in Strimzi. Before introducing tiered storage to your environment, review the known limitations of this feature.

Example custom tiered storage configuration

```
kafka:
  tieredStorage:
    type: custom
    remoteStorageManager:
      className: com.example.kafka.tiered.storage.s3.S3RemoteStorageManager
      classPath: /opt/kafka/plugins/tiered-storage-s3/*
      config:
        # A map with String keys and String values.
        # Key properties are automatically prefixed with `rsm.config.`
        # and appended to Kafka broker config.
        storage.bucket.name: my-bucket
  config:
    ...
    # Additional RLMM configuration can be added through the Kafka config
    # under `spec.kafka.config` using the `rlmm.config.` prefix.
    rlmm.config.remote.log.metadata.topic.replication.factor: 1
```

49.1. TieredStorageCustom schema properties

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

Property	Property type	Description
type	string	Must be `custom`.
remoteStorageManager	`RemoteStorageManager`	Configuration for the Remote Storage Manager.

Chapter 50. `RemoteStorageManager` schema reference

Used in: `TieredStorageCustom`

Property	Property type	Description
className	string	The class name for the `RemoteStorageManager` implementation.
classPath	string	The class path for the `RemoteStorageManager` implementation.
config	map	The additional configuration map for the `RemoteStorageManager` implementation. Keys will be automatically prefixed with `rsm.config.`, and added to Kafka broker configuration.

Chapter 51. QuotasPluginKafka schema reference

Used in: KafkaClusterSpec

The type property is a discriminator that distinguishes use of the QuotasPluginKafka type from QuotasPluginStrimzi. It must have the value kafka for the type QuotasPluginKafka.

Property	Property type	Description
type	string	Must be kafka.
producerByteRate	integer	The default client quota on the maximum bytes per-second that each client can publish to each broker before it is throttled. Applied on a per-broker basis.
consumerByteRate	integer	The default client quota on the maximum bytes per-second that each client can fetch from each broker before it is throttled. Applied on a per-broker basis.
requestPercentage	integer	The default client quota limits the maximum CPU utilization of each client as a percentage of the network and I/O threads of each broker. Applied on a per-broker basis.
controllerMutationRate	number	The default client quota on the rate at which mutations are accepted per second for create topic requests, create partition requests, and delete topic requests, defined for each broker. The mutations rate is measured by the number of partitions created or deleted. Applied on a per-broker basis.

Chapter 52. QuotasPluginStrimzi schema reference

Used in: KafkaClusterSpec

The type property is a discriminator that distinguishes use of the QuotasPluginStrimzi type from QuotasPluginKafka. It must have the value strimzi for the type QuotasPluginStrimzi.

Property	Property type	Description
type	string	Must be strimzi.
producerByteRate	integer	A per-broker byte-rate quota for clients producing to a broker, independent of their number. If clients produce at maximum speed, the quota is shared equally between all non-excluded producers. Otherwise, the quota is divided based on each client's production rate.
consumerByteRate	integer	A per-broker byte-rate quota for clients consuming from a broker, independent of their number. If clients consume at maximum speed, the quota is shared equally between all non-excluded consumers. Otherwise, the quota is divided based on each client's consumption rate.
minAvailableBytesPerVolume	integer	Stop message production if the available size (in bytes) of the storage is lower than or equal to this specified value. This condition is mutually exclusive with minAvailableRatioPerVolume.
minAvailableRatioPerVolume	number	Stop message production if the percentage of available storage space falls below or equals the specified ratio (set as a decimal representing a percentage). This condition is mutually exclusive with minAvailableBytesPerVolume.
excludedPrincipals	string array	List of principals that are excluded from the quota. The principals have to be prefixed with User:, for example User:my-user;User:CN=my-other-user.

Chapter 53. ZookeeperClusterSpec schema reference

The type ZookeeperClusterSpec has been deprecated.

Used in: KafkaSpec

Property	Property type	Description
replicas	integer	The number of pods in the cluster.
image	string	The container image used for ZooKeeper pods. If no image name is explicitly specified, it is determined based on the Kafka version set in spec.kafka.version. The image names are specifically mapped to corresponding versions in the Cluster Operator configuration.
storage	EphemeralStorage, PersistentClaimStorage	Storage configuration (disk). Cannot be updated.
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.
metricsConfig	JmxPrometheusExporterMetrics, StrimziMetricsReporter	Metrics configuration.

Property	Property type	Description
logging	`InlineLogging`, `ExternalLogging`	Logging configuration for ZooKeeper.
template	`ZookeeperClusterTemplate`	Template for ZooKeeper cluster resources. The template allows users to specify how the Kubernetes resources are generated.

Chapter 54. ZookeeperClusterTemplate schema reference

The type ZookeeperClusterTemplate has been deprecated.

Used in: ZookeeperClusterSpec

Property	Property type	Description
statefulset	StatefulSetTemplate	**The statefulset property has been deprecated.** Support for StatefulSets was removed in Strimzi 0.35.0. This property is ignored. Template for ZooKeeper StatefulSet.
podSet	ResourceTemplate	Template for ZooKeeper StrimziPodSet resource.
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.

Chapter 55. EntityOperatorSpec schema reference

Used in: KafkaSpec

Property	Property type	Description
topicOperator	EntityTopicOperatorSpec	Configuration of the Topic Operator.
userOperator	EntityUserOperatorSpec	Configuration of the User Operator.
tlsSidecar	TlsSidecar	**The tlsSidecar property has been deprecated.** TLS sidecar was removed in Strimzi 0.41.0. This property is ignored. TLS sidecar configuration.
template	EntityOperatorTemplate	Template for Entity Operator resources. The template allows users to specify how a Deployment and Pod is generated.

Chapter 56. EntityTopicOperatorSpec schema reference

Used in: EntityOperatorSpec

Full list of EntityTopicOperatorSpec schema properties

Configures the Topic Operator.

56.1. Logging

The Topic Operator has its own preconfigured loggers:

Logger	Description	Default Level
rootLogger	Default logger for all classes	INFO
jetty	Logs HTTP server activity	INFO

The Topic Operator uses the Apache log4j2 logger implementation. Use the logging property to configure loggers and logger levels.

You can set log levels using either the inline or external logging configuration types.

Specify loggers and levels directly in the custom resource for inline configuration:

Example inline logging configuration

```
apiVersion: kafka.strimzi.io/v1
kind: Kafka
metadata:
  name: my-cluster
spec:
  kafka:
    # ...
  entityOperator:
    # ...
    topicOperator:
      watchedNamespace: my-topic-namespace
      reconciliationIntervalMs: 60000
      logging:
        type: inline
        loggers:
          rootLogger.level: INFO
          logger.jetty.level: WARN
  # ...
```

You can define additional loggers by specifying the full class or package name using logger.<name>.name. For example, to configure more detailed logging for the Topic Operator inline:

Example custom inline loggers

```
# ...
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
```

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.

Alternatively, you can reference an external `ConfigMap` containing a complete `log4j2.properties` file that defines your own log4j2 configuration, including loggers, appenders, and layout configuration:

Example external logging configuration

```
apiVersion: kafka.strimzi.io/v1
kind: Kafka
metadata:
  name: my-cluster
spec:
  kafka:
    # ...
  entityOperator:
    # ...
    topicOperator:
      watchedNamespace: my-topic-namespace
      reconciliationIntervalMs: 60000
      logging:
        type: external
        valueFrom:
          configMapKeyRef:
            # name and key are mandatory
            name: customConfigMap
            key: log4j2.properties
  # ...
```

Garbage collector (GC)

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

56.2. `EntityTopicOperatorSpec` schema properties

Property	Property type	Description
watchedNamespace	string	The namespace the Topic Operator should watch.

Property	Property type	Description
image	string	The image to use for the Topic Operator.
reconciliationIntervalSeconds	integer	**The `reconciliationIntervalSeconds` property has been deprecated, and should now be configured using `.spec.entityOperator.topicOperator.reconciliationIntervalMs`.** Interval between periodic reconciliations in seconds. Ignored if reconciliationIntervalMs is set.
reconciliationIntervalMs	integer	Interval between periodic reconciliations in milliseconds.
zookeeperSessionTimeoutSeconds	integer	**The `zookeeperSessionTimeoutSeconds` property has been deprecated.** This property is not used anymore in Strimzi 0.41.0 and it is ignored. Timeout for the ZooKeeper session.
startupProbe	`Probe`	Pod startup checking.
livenessProbe	`Probe`	Pod liveness checking.
readinessProbe	`Probe`	Pod readiness checking.
resources	ResourceRequirements	CPU and memory resources to reserve.
topicMetadataMaxAttempts	integer	**The `topicMetadataMaxAttempts` property has been deprecated.** This property is not used anymore in Strimzi 0.41.0 and it is ignored. The number of attempts at getting topic metadata.
logging	`InlineLogging`, `ExternalLogging`	Logging configuration.
jvmOptions	`JvmOptions`	JVM Options for pods.

Chapter 57. EntityUserOperatorSpec schema reference

Used in: EntityOperatorSpec

Full list of EntityUserOperatorSpec schema properties

Configures the User Operator.

57.1. Logging

The User Operator has its own preconfigured loggers:

Logger	Description	Default Level
rootLogger	Default logger for all classes	INFO
jetty	Logs HTTP server activity	INFO

The User Operator uses the Apache log4j2 logger implementation. Use the logging property to configure loggers and logger levels.

You can set log levels using either the inline or external logging configuration types.

Specify loggers and levels directly in the custom resource for inline configuration:

Example inline logging configuration

```
apiVersion: kafka.strimzi.io/v1
kind: Kafka
metadata:
  name: my-cluster
spec:
  kafka:
    # ...
  entityOperator:
    # ...
    userOperator:
      watchedNamespace: my-topic-namespace
      reconciliationIntervalMs: 60000
      logging:
        type: inline
        loggers:
          rootLogger.level: INFO
          logger.jetty.level: WARN
  # ...
```

You can define additional loggers by specifying the full class or package name using logger.<name>.name. For example, to configure more detailed logging for the User Operator inline:

Example custom inline loggers

```
# ...
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
```

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.

Alternatively, you can reference an external `ConfigMap` containing a complete `log4j2.properties` file that defines your own log4j2 configuration, including loggers, appenders, and layout configuration:

Example external logging configuration

```
apiVersion: kafka.strimzi.io/v1
kind: Kafka
metadata:
  name: my-cluster
spec:
  kafka:
    # ...
  entityOperator:
    # ...
    userOperator:
      watchedNamespace: my-topic-namespace
      reconciliationIntervalMs: 60000
      logging:
        type: external
        valueFrom:
          configMapKeyRef:
            # name and key are mandatory
            name: customConfigMap
            key: log4j2.properties
   # ...
```

Garbage collector (GC)

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

57.2. `EntityUserOperatorSpec` schema properties

Property	Property type	Description
watchedNamespace	string	The namespace the User Operator should watch.

Property	Property type	Description
image	string	The image to use for the User Operator.
reconciliationIntervalSeconds	integer	**The `reconciliationIntervalSeconds` property has been deprecated, and should now be configured using `.spec.entityOperator.userOperator.reconciliationIntervalMs`.** Interval between periodic reconciliations in seconds. Ignored if reconciliationIntervalMs is set.
reconciliationIntervalMs	integer	Interval between periodic reconciliations in milliseconds.
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.
logging	`InlineLogging`, `ExternalLogging`	Logging configuration.
jvmOptions	`JvmOptions`	JVM Options for pods.

Chapter 58. TlsSidecar schema reference

The type TlsSidecar has been deprecated.

Used in: CruiseControlSpec, EntityOperatorSpec

Full list of TlsSidecar schema properties

The TLS sidecar type is not used anymore. If set, it will be ignored

58.1. TlsSidecar schema properties

Property	Property type	Description
image	string	The docker image for the container.
resources	ResourceRequirements	CPU and memory resources to reserve.
livenessProbe	Probe	Pod liveness checking.
readinessProbe	Probe	Pod readiness checking.
logLevel	string (one of [emerg, debug, crit, err, alert, warning, notice, info])	The log level for the TLS sidecar. Default value is notice.

Chapter 59. `EntityOperatorTemplate` schema reference

Used in: `EntityOperatorSpec`

Property	Property type	Description
deployment	`DeploymentTemplate`	Template for Entity Operator `Deployment`.
pod	`PodTemplate`	Template for Entity Operator `Pods`.
topicOperatorContainer	`ContainerTemplate`	Template for the Entity Topic Operator container.
userOperatorContainer	`ContainerTemplate`	Template for the Entity User Operator container.
tlsSidecarContainer	`ContainerTemplate`	**The `tlsSidecarContainer` property has been deprecated.** TLS sidecar was removed in Strimzi 0.41.0. This property is ignored. Template for the Entity Operator TLS sidecar container.
serviceAccount	`ResourceTemplate`	Template for the Entity Operator service account.
podDisruptionBudget	`PodDisruptionBudgetTemplate`	Template for the Entity Operator Pod Disruption Budget.
entityOperatorRole	`ResourceTemplate`	Template for the Entity Operator Role.
topicOperatorRoleBinding	`ResourceTemplate`	Template for the Entity Topic Operator RoleBinding.
userOperatorRoleBinding	`ResourceTemplate`	Template for the Entity Topic Operator RoleBinding.

Chapter 60. DeploymentTemplate schema reference

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

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.

60.1. DeploymentTemplate schema properties

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

Chapter 61. `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	Property type	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.

Property	Property type	Description
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 62. `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

The `config` properties are one part of the overall configuration for the resource. Use the `config` properties to configure Cruise Control options as keys.

Example Cruise Control configuration

```
apiVersion: kafka.strimzi.io/v1
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"
    # ...
```

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, Strimzi 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
- 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 Strimzi:

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

62.1. 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 Strimzi. 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/v1
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.

62.2. 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 Strimzi 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/v1
kind: Kafka
metadata:
  name: my-cluster
spec:
  # ...
  cruiseControl:
    config:
      webserver.security.enable: false
      webserver.ssl.enable: false
# ...
```

62.3. API users

With the necessary permissions, create REST API users to safely access a secured Cruise Control REST API directly.

This allows roles and permissions to be defined to allow advanced users and third-party applications to access the Cruise Control REST API without having to disable basic HTTP authentication.

The following use cases would benefit from accessing the Cruise Control API without disabling API security:

- Monitoring a Strimzi-managed Kafka cluster with the Cruise Control user interface.
- Gathering Cruise Control-specific statistical information not available through Strimzi or Cruise Control sensor metrics, such as detailed information surrounding cluster and partition load and user tasks.
- Debugging Cruise Control in a secured environment.

Cruise Control reads authentication credentials for API users in Jetty's `HashLoginService` file format.

Standard Cruise Control `USER` and `VIEWER` roles are supported.

- `USER` has access to all the `GET` endpoints except `bootstrap` and `train`.
- `VIEWER` has access to `kafka_cluster_state`, `user_tasks`, and `review_board` endpoints.

In this example, we define two custom API users in the supported format in a text file called `cruise-control-auth.txt`:

```
userOne: passwordOne, USER
userTwo: passwordTwo, VIEWER
```

Then, use this file to create a secret with the following command:

```
kubectl create secret generic cruise-control-api-users-secret  --from-file=cruise
-control-auth.txt=cruise-control-auth.txt
```

Next, we reference the secret in the `spec.cruiseControl.apiUsers` section of the Kafka resource:

Example Cruise Control apiUsers configuration

```
apiVersion: kafka.strimzi.io/v1
kind: Kafka
metadata:
  name: my-cluster
spec:
  # ...
  cruiseControl:
    # ...
    apiUsers:
      type: hashLoginService
      valueFrom:
        secretKeyRef:
          name: cruise-control-api-users-secret
          key: cruise-control-auth.txt
     ...
```

Strimzi then decodes and uses the contents of this secret to populate Cruise Control's API authentication credentials file.

62.4. Configuring capacity limits

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 Kubernetes 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 Strimzi, 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/v1
kind: Kafka
metadata:
  name: my-cluster
spec:
  # ...
  cruiseControl:
    # ...
    brokerCapacity:
      cpu: "2"
      inboundNetwork: 10000KiB/s
      outboundNetwork: 10000KiB/s
    # ...
```

62.5. Configuring 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/v1
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.

62.6. Logging

Cruise Control has its own preconfigured logger:

Logger	Description	Default Level
`rootLogger`	Default logger for all classes	INFO

Cruise Control uses the Apache `log4j2` logger implementation. Use the `logging` property to configure loggers and logger levels.

You can set log levels using either the `inline` or `external` logging configuration types.

Specify loggers and levels directly in the custom resource for inline configuration:

Example inline logging configuration

```
apiVersion: kafka.strimzi.io/v1
kind: Kafka
# ...
spec:
  cruiseControl:
    # ...
    logging:
      type: inline
      loggers:
        rootLogger.level: INFO
    # ...
```

You can define additional loggers by specifying the full class or package name using `logger.<name>.name`. For example, to configure more detailed logging for Cruise Control inline:

Example custom inline loggers

```
# ...
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.

Alternatively, you can reference an external `ConfigMap` containing a complete `log4j2.properties` file that defines your own log4j2 configuration, including loggers, appenders, and layout configuration:

Example external logging configuration

```
apiVersion: kafka.strimzi.io/v1
kind: Kafka
# ...
spec:
  cruiseControl:
    # ...
    logging:
      type: external
      valueFrom:
        configMapKeyRef:
          # name and key are mandatory
          name: customConfigMap
          key: log4j2.properties
```

```
    # ...
```

Garbage collector (GC)

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

62.7. `CruiseControlSpec` schema properties

Property	Property type	Description
image	string	The container image used for Cruise Control pods. If no image name is explicitly specified, the image name corresponds to the name specified in the Cluster Operator configuration. If an image name is not defined in the Cluster Operator configuration, a default value is used.
tlsSidecar	`TlsSidecar`	**The `tlsSidecar` property has been deprecated.** TLS sidecar configuration.
resources	ResourceRequirements	CPU and memory resources to reserve for the Cruise Control container.
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.
template	`CruiseControlTemplate`	Template to specify how Cruise Control resources, `Deployments` and `Pods`, are generated.
brokerCapacity	`BrokerCapacity`	The Cruise Control `brokerCapacity` configuration.

Property	Property type	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`, `StrimziMetricsReporter`	Metrics configuration. Only `jmxPrometheusExporter` can be configured, as this component does not support `strimziMetricsReporter`.
apiUsers	`HashLoginServiceApiUsers`	Configuration of the Cruise Control REST API users.
autoRebalance	`KafkaAutoRebalanceConfiguration` array	Auto-rebalancing on scaling related configuration listing the modes, when brokers are added or removed, with the corresponding rebalance template configurations.If this field is set, at least one mode has to be defined.

Chapter 63. `CruiseControlTemplate` schema reference

Used in: `CruiseControlSpec`

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

Chapter 64. BrokerCapacity schema reference

Used in: CruiseControlSpec

Property	Property type	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 Kubernetes 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 Kubernetes 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 Kubernetes byte units (K, M, G) or their bibyte (power of two) equivalents (Ki, Mi, Gi) per second. For example, 10000KiB/s.

Property	Property type	Description
overrides	`BrokerCapacityOverride` array	Overrides for individual brokers. The `overrides` property lets you specify a different capacity configuration for different brokers.

Chapter 65. BrokerCapacityOverride schema reference

Used in: BrokerCapacity

Property	Property type	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 Kubernetes 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 Kubernetes byte units (K, M, G) or their bibyte (power of two) equivalents (Ki, Mi, Gi) per second. For example, 10000KiB/s.

Chapter 66. HashLoginServiceApiUsers schema reference

Used in: CruiseControlSpec

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

Property	Property type	Description
type	string	Must be hashLoginService.
valueFrom	PasswordSource	Secret from which the custom Cruise Control API authentication credentials are read.

Chapter 67. PasswordSource schema reference

Used in: HashLoginServiceApiUsers, Password

Property	Property type	Description
secretKeyRef	SecretKeySelector	Selects a key of a Secret in the resource's namespace.

Chapter 68. KafkaAutoRebalanceConfiguration schema reference

Used in: CruiseControlSpec

Property	Property type	Description
mode	string (one of [remove-brokers, add-brokers])	Specifies the mode for automatically rebalancing when brokers are added or removed. Supported modes are add-brokers and remove-brokers.
template	LocalObjectReference	Reference to the KafkaRebalance custom resource to be used as the configuration template for the auto-rebalancing on scaling when running for the corresponding mode.

Chapter 69. `LocalObjectReference` schema reference

Used in: `AlterOffsets`, `KafkaAutoRebalanceConfiguration`, `ListOffsets`

Property	Property type	Description
name	string	

Chapter 70. JmxTransSpec schema reference

The type JmxTransSpec has been deprecated.

Used in: KafkaSpec

Property	Property type	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.
template	JmxTransTemplate	Template for JmxTrans resources.

Chapter 71. JmxTransOutputDefinitionTemplate schema reference

Used in: JmxTransSpec

Property	Property type	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 72. JmxTransQueryTemplate schema reference

Used in: JmxTransSpec

Property	Property type	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 73. `JmxTransTemplate` schema reference

Used in: `JmxTransSpec`

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

Chapter 74. KafkaExporterSpec schema reference

Used in: KafkaSpec

Property	Property type	Description
image	string	The container image used for the Kafka Exporter pods. If no image name is explicitly specified, the image name corresponds to the version specified in the Cluster Operator configuration. If an image name is not defined in the Cluster Operator configuration, a default value is used.
groupRegex	string	Regular expression to specify which consumer groups to collect. Default value is .*.
topicRegex	string	Regular expression to specify which topics to collect. Default value is .*.
groupExcludeRegex	string	Regular expression to specify which consumer groups to exclude.
topicExcludeRegex	string	Regular expression to specify which topics to exclude.
resources	ResourceRequirements	CPU and memory resources to reserve.
logging	string	Only log messages with the given severity or above. Valid levels: [info, debug, trace]. Default log level is info.
livenessProbe	Probe	Pod liveness check.
readinessProbe	Probe	Pod readiness check.
enableSaramaLogging	boolean	Enable Sarama logging, a Go client library used by the Kafka Exporter.
showAllOffsets	boolean	Whether show the offset/lag for all consumer group, otherwise, only show connected consumer groups.
template	KafkaExporterTemplate	Customization of deployment templates and pods.

Chapter 75. KafkaExporterTemplate schema reference

Used in: KafkaExporterSpec

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

Chapter 76. KafkaStatus schema reference

Used in: Kafka

Property	Property type	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.
registeredNodeIds	integer array	**The registeredNodeIds property has been deprecated.** The registeredNodeIds property is deprecated and it is not used anymore. It will be removed in the future. Registered node IDs used by this Kafka cluster. This field is used for internal purposes only and will be removed in the future.
clusterId	string	Kafka cluster Id.
operatorLastSuccessfulVersion	string	The version of the Strimzi Cluster Operator which performed the last successful reconciliation.
kafkaVersion	string	The version of Kafka currently deployed in the cluster.
kafkaMetadataVersion	string	The KRaft metadata.version currently used by the Kafka cluster.
kafkaMetadataState	string (one of [PreKRaft, ZooKeeper, KRaftMigration, KRaftDualWriting, KRaftPostMigration, KRaft])	**The kafkaMetadataState property has been deprecated.** The kafkaMetadataState property is deprecated and will be removed in the future. Defines where cluster metadata are stored. This property is deprecated and not used anymore.
autoRebalance	KafkaAutoRebalanceStatus	The status of an auto-rebalancing triggered by a cluster scaling request.

Chapter 77. Condition schema reference

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

Property	Property type	Description
type	string	The unique identifier of a condition, used to distinguish between other conditions in the resource.
status	string	The status of the condition, either True, False or Unknown.
lastTransitionTime	string	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.
reason	string	The reason for the condition's last transition (a single word in CamelCase).
message	string	Human-readable message indicating details about the condition's last transition.

Chapter 78. `ListenerStatus` schema reference

Used in: `KafkaStatus`

Property	Property type	Description
type	string	**The `type` property has been deprecated, and should now be configured using `.status.listeners.name`.** 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 79. `ListenerAddress` schema reference

Used in: `ListenerStatus`

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

Chapter 80. UsedNodePoolStatus schema reference

Used in: KafkaStatus

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

Chapter 81. KafkaAutoRebalanceStatus schema reference

Used in: KafkaStatus

Property	Property type	Description
state	string (one of [RebalanceOnScaleUp, Idle, RebalanceOnScaleDown])	The current state of an auto-rebalancing operation. Possible values are: • Idle as the initial state when an auto-rebalancing is requested or as final state when it completes or fails. • RebalanceOnScaleDown if an auto-rebalance related to a scale-down operation is running. • RebalanceOnScaleUp if an auto-rebalance related to a scale-up operation is running.
lastTransitionTime	string	The timestamp of the latest auto-rebalancing state update.
modes	KafkaAutoRebalanceStatusBrokers array	List of modes where an auto-rebalancing operation is either running or queued. Each mode entry (add-brokers or remove-brokers) includes one of the following: • Broker IDs for a current auto-rebalance. • Broker IDs for a queued auto-rebalance (if a previous rebalance is still in progress).

Chapter 82. KafkaAutoRebalanceStatusBrokers schema reference

Used in: KafkaAutoRebalanceStatus

Property	Property type	Description
mode	string (one of [remove-brokers, add-brokers])	Mode for which there is an auto-rebalancing operation in progress or queued, when brokers are added or removed. The possible modes are `add-brokers` and `remove-brokers`.
brokers	integer array	List of broker IDs involved in an auto-rebalancing operation related to the current mode. The list contains one of the following: • Broker IDs for a current auto-rebalance. • Broker IDs for a queued auto-rebalance (if a previous auto-rebalance is still in progress).

Chapter 83. KafkaConnect schema reference

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

Chapter 84. KafkaConnectSpec schema reference

Used in: KafkaConnect

Full list of KafkaConnectSpec schema properties

Configures a Kafka Connect cluster.

84.1. Group ID and internal topics

The group ID and the internal topics used by the Kafka Connect cluster are configured in the .spec section of the KafkaConnect resource

Example group ID and internal topics configuration

```
apiVersion: kafka.strimzi.io/v1
kind: KafkaConnect
metadata:
  name: my-connect
spec:
  # ...
  groupId: my-connect-group
  configStorageTopic: my-config-topic
  offsetStorageTopic: my-offset-topic
  statusStorageTopic: my-status-topic
  # ...
```

These fields are required in the v1 CRD API version. In the v1beta2 version, these properties are optional.

If you do not set them, the following default values apply:

- groupId defaults to connect-cluster
- configStorageTopic defaults to connect-cluster-configs
- offsetStorageTopic defaults to connect-cluster-offsets
- statusStorageTopic defaults to connect-cluster-status

84.2. Additional configuration

The config properties are one part of the overall configuration for the resource. Use the config properties to configure additional Kafka Connect options as keys.

Example Kafka Connect configuration

```
apiVersion: kafka.strimzi.io/v1
```

```
kind: KafkaConnect
metadata:
  name: my-connect
spec:
  # ...
  config:
    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
  # ...
```

The values can be one of the following JSON types:

- String
- Number
- Boolean

Certain options have default values:

- `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 Kafka Connect configuration documentation.

However, Strimzi 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 Strimzi:

- 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 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.

84.3. Logging

Kafka Connect has its own preconfigured loggers:

Logger	Description	Default Level
`rootLogger`	Default logger for all classes	INFO
`reflections`	Logs classpath scanning and metadata discovery used to find plugins	ERROR

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 `log4j2` logger implementation. Use the `logging` property to configure loggers and logger levels.

You can set log levels using either the `inline` or `external` logging configuration types.

Specify loggers and levels directly in the custom resource for inline configuration:

Example inline logging configuration

```
apiVersion: kafka.strimzi.io/v1
kind: KafkaConnect
spec:
```

```
  # ...
  logging:
    type: inline
    loggers:
      rootLogger.level: INFO
      logger.reflections.level: DEBUG
  # ...
```

You can define additional loggers by specifying the full class or package name using `logger.<name>.name`. For example, to configure logging for Kafka Connect runtime classes inline:

Example custom inline loggers

```
# ...
logger.sourcetask.name: org.apache.kafka.connect.runtime.WorkerSourceTask ①
logger.sourcetask.level: TRACE ②
logger.sinktask.name: org.apache.kafka.connect.runtime.WorkerSinkTask ③
logger.sinktask.level: DEBUG ④
```

① Creates a logger for the runtime `WorkerSourceTask` class.

② Sets the logging level for `WorkerSourceTask`.

③ Creates a logger for the runtime `WorkerSinkTask` class.

④ Sets the logging level for `WorkerSinkTask`.

Alternatively, you can reference an external `ConfigMap` containing a complete `log4j2.properties` file that defines your own log4j2 configuration, including loggers, appenders, and layout configuration:

Example external logging configuration

```
apiVersion: kafka.strimzi.io/v1
kind: KafkaConnect
spec:
  # ...
  logging:
    type: external
    valueFrom:
      configMapKeyRef:
        # name and key are mandatory
        name: customConfigMap
        key: log4j2.properties
  # ...
```

Garbage collector (GC)

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

84.4. KafkaConnectSpec schema properties

Property	Property type	Description
version	string	The Kafka Connect version. Defaults to the latest version. 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. Required in the `v1` version of the Strimzi API. Defaults to `3` in the `v1beta2` version of the Strimzi API.
image	string	The container image used for Kafka Connect pods. If no image name is explicitly specified, it is determined based on the `spec.version` configuration. The image names are specifically mapped to corresponding versions in the Cluster Operator configuration.
bootstrapServers	string	Bootstrap servers to connect to. This should be given as a comma separated list of <*hostname*>:_<port>_ pairs.
groupId	string	A unique ID that identifies the Connect cluster group.
configStorageTopic	string	The name of the Kafka topic where connector configurations are stored.
statusStorageTopic	string	The name of the Kafka topic where connector and task status are stored.
offsetStorageTopic	string	The name of the Kafka topic where source connector offsets are stored.
tls	`ClientTls`	TLS configuration.
authentication	`KafkaClientAuthenticationTls`, `KafkaClientAuthenticationScramSha256`, `KafkaClientAuthenticationScramSha512`, `KafkaClientAuthenticationPlain`, `KafkaClientAuthenticationOAuth`, `KafkaClientAuthenticationCustom`	Authentication configuration for Kafka Connect.

Property	Property type	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, prometheus.metrics.reporter. (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.
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.
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.
metricsConfig	`JmxPrometheusExporterMetrics`, `StrimziMetricsReporter`	Metrics configuration.
tracing	`JaegerTracing`, `OpenTelemetryTracing`	The configuration of tracing in Kafka Connect.
template	`KafkaConnectTemplate`	Template for Kafka Connect and Kafka MirrorMaker 2 resources. The template allows users to specify how the `Pods`, `Service`, and other services are generated.

Property	Property type	Description
externalConfiguration	`ExternalConfiguration`	**The `externalConfiguration` property has been deprecated.** The external configuration is deprecated and will be removed in the future. Please use the template section instead to configure additional environment variables or volumes. 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.
plugins	`MountedPlugin` array	List of connector plugins to mount into the `KafkaConnect` pod.

Chapter 85. `ClientTls` schema reference

Used in: `KafkaBridgeSpec`, `KafkaConnectSpec`, `KafkaMirrorMaker2ClusterSpec`, `KafkaMirrorMaker2TargetClusterSpec`

Full list of `ClientTls` schema properties

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

85.1. `ClientTls` schema properties

Property	Property type	Description
trustedCertificates	`CertSecretSource` array	Trusted certificates for TLS connection.

Chapter 86. KafkaClientAuthenticationTls schema reference

Used in: KafkaBridgeSpec, KafkaConnectSpec, KafkaMirrorMaker2ClusterSpec, KafkaMirrorMaker2TargetClusterSpec

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.

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

Example mTLS configuration

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

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:

```
kubectl create secret generic <my_tls_secret> \
--from-file=<my_public_tls_certificate>.crt \
--from-file=<my_private_key>.key
```

Example secret for mTLS client authentication

```
apiVersion: v1
kind: Secret
metadata:
  name: my-tls-secret
type: Opaque
data:
  tls.crt: LS0tLS1CRUdJTiBDRVJ...
  tls.key: LS0tLS1CRUdJTiBQUkl...
```

NOTE mTLS authentication can only be used with TLS connections.

86.1. KafkaClientAuthenticationTls schema properties

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

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

Chapter 87. KafkaClientAuthenticationScramSha256 schema reference

Used in: KafkaBridgeSpec, KafkaConnectSpec, KafkaMirrorMaker2ClusterSpec, KafkaMirrorMaker2TargetClusterSpec

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.

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
```

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:

```
kubectl 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.

87.1. `KafkaClientAuthenticationScramSha256` schema properties

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

Chapter 88. PasswordSecretSource schema reference

Used in: KafkaClientAuthenticationOAuth, KafkaClientAuthenticationPlain, KafkaClientAuthenticationScramSha256, KafkaClientAuthenticationScramSha512

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

Chapter 89. KafkaClientAuthenticationScramSha512 schema reference

Used in: KafkaBridgeSpec, KafkaConnectSpec, KafkaMirrorMaker2ClusterSpec, KafkaMirrorMaker2TargetClusterSpec

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.

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
```

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:

```
kubectl 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.

89.1. `KafkaClientAuthenticationScramSha512` schema properties

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

Chapter 90. KafkaClientAuthenticationPlain schema reference

Used in: KafkaBridgeSpec, KafkaConnectSpec, KafkaMirrorMaker2ClusterSpec, KafkaMirrorMaker2TargetClusterSpec

Full list of KafkaClientAuthenticationPlain schema properties

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

An example SASL-based PLAIN client authentication configuration for Kafka Connect

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

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.

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:

```
kubectl 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.

90.1. `KafkaClientAuthenticationPlain` schema properties

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

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

Chapter 91. KafkaClientAuthenticationOAuth schema reference

The type KafkaClientAuthenticationOAuth has been deprecated. Please use KafkaClientAuthenticationCustom instead.

Used in: KafkaBridgeSpec, KafkaConnectSpec, KafkaMirrorMaker2ClusterSpec, KafkaMirrorMaker2TargetClusterSpec

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.

Example client ID and client secret configuration

```
authentication:
  type: oauth
  tokenEndpointUri: https://<auth_server_address>/<path_to_token_endpoint>
  clientId: my-client-id
  clientSecret:
    secretName: my-client-oauth-secret
    key: client-secret
```

Optionally, scope and audience can be specified if needed. grantType can also be specified for custom client credentials implementations.

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.

Example client ID and refresh token configuration

```
authentication:
  type: oauth
  tokenEndpointUri: https://<auth_server_address>/<path_to_token_endpoint>
  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. Alternatively, use `accessTokenLocation` property, and specify a path to the token file.

Example access token only configuration

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

Example (service account) access token configuration specifying a mounted file

```
authentication:
  type: oauth
  accessTokenLocation: `/var/run/secrets/kubernetes.io/serviceaccount/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`.

Example username and password configuration with a public client

```
authentication:
  type: oauth
  tokenEndpointUri: https://<auth_server_address>/<path_to_token_endpoint>
  username: my-username
  passwordSecret:
    secretName: my-password-secret-name
    password: my-password-field-name
  clientId: my-public-client-id
```

Example username and password configuration with a confidential client

```
authentication:
  type: oauth
  tokenEndpointUri: https://<auth_server_address>/<path_to_token_endpoint>
  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 uses self-signed certificates or certificates signed by an untrusted certification authority, use the `tlsTrustedCertificates` property to specify the secrets containing them. The certificates must be in X.509 format.

Example configuration specifying TLS certificates

```
authentication:
  type: oauth
  tokenEndpointUri: https://<auth_server_address>/<path_to_token_endpoint>
```

```
  clientId: my-client-id
  refreshToken:
    secretName: my-refresh-token-secret
    key: refresh-token
  tlsTrustedCertificates:
    - secretName: oauth-server-ca
      pattern: "*.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.

Example configuration to disable TLS hostname verification

```
authentication:
  type: oauth
  tokenEndpointUri: https://<auth_server_address>/<path_to_token_endpoint>
  clientId: my-client-id
  refreshToken:
    secretName: my-refresh-token-secret
    key: refresh-token
  disableTlsHostnameVerification: true
```

91.1. KafkaClientAuthenticationOAuth schema properties

The type KafkaClientAuthenticationOAuth is supported only in the Strimzi API version(s) v1beta2.

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

Property	Property type	Description
type	string	Must be `oauth`.
clientId	string	OAuth Client ID which the Kafka client can use to authenticate against the OAuth server and use the token endpoint URI.
username	string	Username used for the authentication.
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.

Property	Property type	Description
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.
tokenEndpointUri	string	Authorization server 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.
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.
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.
clientSecret	`GenericSecretSource`	Link to Kubernetes Secret containing the OAuth client secret which the Kafka client can use to authenticate against the OAuth server and use the token endpoint URI.
passwordSecret	`PasswordSecretSource`	Reference to the `Secret` which holds the password.
accessToken	`GenericSecretSource`	Link to Kubernetes Secret containing the access token which was obtained from the authorization server.
refreshToken	`GenericSecretSource`	Link to Kubernetes Secret containing the refresh token which can be used to obtain access token from the authorization server.
tlsTrustedCertificates	`CertSecretSource` array	Trusted certificates for TLS connection to the OAuth server.

Property	Property type	Description
disableTlsHostnameVerification	boolean	Enable or disable TLS hostname verification. Default value is `false`.
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.
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`.
enableMetrics	boolean	Enable or disable OAuth metrics. Default value is `false`.
includeAcceptHeader	boolean	Whether the Accept header should be set in requests to the authorization servers. The default value is `true`.
accessTokenLocation	string	Path to the token file containing an access token to be used for authentication.
clientAssertion	`GenericSecretSource`	Link to Kubernetes secret containing the client assertion which was manually configured for the client.
clientAssertionLocation	string	Path to the file containing the client assertion to be used for authentication.
clientAssertionType	string	The client assertion type. If not set, and either `clientAssertion` or `clientAssertionLocation` is configured, this value defaults to `urn:ietf:params:oauth:client-assertion-type:jwt-bearer`.
saslExtensions	map	SASL extensions parameters.
grantType	string	A custom OAuth grant type to use when authenticating against the authorization server with `clientId` and one of `clientSecret` or `clientAssertion`. The value defaults to `client_credentials` in these cases. This is optional configuration, only used with custom authorization server implementations.

Chapter 92. KafkaClientAuthenticationCustom schema reference

Used in: KafkaBridgeSpec, KafkaConnectSpec, KafkaMirrorMaker2ClusterSpec, KafkaMirrorMaker2TargetClusterSpec

Full list of KafkaClientAuthenticationCustom schema properties

To configure custom client authentication, set the type property to custom. Custom client authentication allows you to use any type of Kafka-supported authentication mechanism. This authentication option is especially useful with third-party Apache Kafka services, such as Amazon MSK, that require their own authentication mechanisms.

Example custom client authentication configuration using the AWS_MSK_IAM authentication

```
  authentication:
    type: custom
    sasl: true
    config:
      sasl.mechanism: AWS_MSK_IAM
      sasl.jaas.config: software.amazon.msk.auth.iam.IAMLoginModule required;
      sasl.client.callback.handler.class:
software.amazon.msk.auth.iam.IAMClientCallbackHandler
```

The config section accepts only options that start with the prefixes sasl. and ssl.keystore.. All other options are ignored. To configure additional options, use the related properties of the Strimzi custom resource, such as .spec.config or .spec.tls. The security.protocol `setting is generated automatically from the `sasl property (when you use custom authentication) and the tls configuration in the same Strimzi custom resource:

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

92.1. Using secrets in custom authentication

If your custom authentication mechanism requires additional information from Secret or ConfigMap resources, use the Additional Volumes feature to mount them into the operands. Kafka configuration providers can be used to load them from the disk.

92.2. Using additional authentication plugins

Custom authentication mechanisms might require additional plugins for the operand. You can add them using the Additional Volumes feature. Alternatively, you can build a custom container image.

92.3. `KafkaClientAuthenticationCustom` schema properties

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

Property	Property type	Description
type	string	Must be `custom`.
sasl	boolean	Enable or disable SASL on this authentication mechanism.
config	map	Configuration for the custom authentication mechanism. Only properties with the `sasl.` and `ssl.keystore.` prefixes are allowed. Specify other options in the regular configuration section of the custom resource.

Chapter 93. JaegerTracing schema reference

The type JaegerTracing has been deprecated.

Used in: KafkaBridgeSpec, KafkaConnectSpec, KafkaMirrorMaker2Spec

The type JaegerTracing is supported only in the Strimzi API version(s) v1beta2.

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	Property type	Description
type	string	Must be jaeger.

Chapter 94. OpenTelemetryTracing schema reference

Used in: KafkaBridgeSpec, KafkaConnectSpec, KafkaMirrorMaker2Spec

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	Property type	Description
type	string	Must be opentelemetry.

Chapter 95. KafkaConnectTemplate schema reference

Used in: KafkaConnectSpec, KafkaMirrorMaker2Spec

Property	Property type	Description
deployment	DeploymentTemplate	**The deployment property has been deprecated.** Kafka Connect and MirrorMaker 2 operands do not use Deployment resources anymore. This field will be ignored. Template for Kafka Connect Deployment.
podSet	ResourceTemplate	Template for Kafka Connect StrimziPodSet resource.
pod	PodTemplate	Template for Kafka Connect Pods.
apiService	InternalServiceTemplate	Template for Kafka Connect API Service.
headlessService	InternalServiceTemplate	Template for Kafka Connect headless Service.
connectContainer	ContainerTemplate	Template for the Kafka Connect container.
initContainer	ContainerTemplate	Template for the Kafka init container.
podDisruptionBudget	PodDisruptionBudgetTemplate	Template for Kafka Connect PodDisruptionBudget.
serviceAccount	ResourceTemplate	Template for the Kafka Connect service account.
clusterRoleBinding	ResourceTemplate	Template for the Kafka Connect ClusterRoleBinding.
buildPod	PodTemplate	Template for Kafka Connect Build Pods. The build pod is used only on Kubernetes.
buildContainer	ContainerTemplate	Template for the Kafka Connect Build container. The build container is used only on Kubernetes.
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.

Property	Property type	Description
jmxSecret	`ResourceTemplate`	Template for Secret of the Kafka Connect Cluster JMX authentication.

Chapter 96. BuildConfigTemplate schema reference

Used in: KafkaConnectTemplate

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

Chapter 97. ExternalConfiguration schema reference

The type ExternalConfiguration has been deprecated. Please use KafkaConnectTemplate instead.

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.

97.1. ExternalConfiguration schema properties

Property	Property type	Description
env	ExternalConfigurationEnv array	**The env property has been deprecated.** The external configuration environment variables are deprecated and will be removed in the future. Please use the environment variables in a container template instead. Makes data from a Secret or ConfigMap available in the Kafka Connect pods as environment variables.
volumes	ExternalConfigurationVolumeSource array	**The volumes property has been deprecated.** The external configuration volumes are deprecated and will be removed in the future. Please use the additional volumes and volume mounts in pod and container templates instead to mount additional secrets or config maps. Makes data from a Secret or ConfigMap available in the Kafka Connect pods as volumes.

Chapter 98. ExternalConfigurationEnv schema reference

The type ExternalConfigurationEnv has been deprecated. Please use ContainerEnvVar instead.

Used in: ExternalConfiguration

Property	Property type	Description
name	string	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_.
valueFrom	ExternalConfigurationEnvVarSource	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.

Chapter 99. ExternalConfigurationEnvVarSource schema reference

Used in: ExternalConfigurationEnv

Property	Property type	Description
secretKeyRef	SecretKeySelector	Reference to a key in a Secret.
configMapKeyRef	ConfigMapKeySelector	Reference to a key in a ConfigMap.

Chapter 100. `ExternalConfigurationVolumeSource` schema reference

The type `ExternalConfigurationVolumeSource` has been deprecated. Please use `AdditionalVolume` instead.

Used in: `ExternalConfiguration`

Property	Property type	Description
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.
configMap	ConfigMapVolumeSource	Reference to a key in a ConfigMap. Exactly one Secret or ConfigMap has to be specified.

Chapter 101. `Build` schema reference

Used in: `KafkaConnectSpec`

Full list of `Build` schema properties

Configures additional connectors for Kafka Connect deployments.

101.1. Configuring container registries

To build new container images with additional connector plugins, Strimzi requires a container registry where the images can be pushed to, stored, and pulled from. Strimzi does not run its own container registry, so a registry must be provided. Strimzi 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/v1
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 Strimzi.

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/v1
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 Strimzi.

② (Required) Name of the ImageStream and tag.

101.2. Configuring connector 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 Strimzi, 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/v1
```

```
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.

Strimzi 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	Strimzi 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, Strimzi will verify the checksum of the artifact while building the new container image.

Example JAR artifact

```
apiVersion: kafka.strimzi.io/v1
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 Strimzi 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, Strimzi will verify the checksum before unpacking it and building the new container image.

Example TGZ artifact

```
apiVersion: kafka.strimzi.io/v1
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/v1
kind: KafkaConnect
metadata:
  name: my-connect-cluster
spec:
  #...
  build:
    output:
      #...
    plugins:
      - name: my-plugin
        artifacts:
          - type: maven ①
            repository: https://mvnrepository.com ②
            group: <maven_group> ③
            artifact: <maven_artifact> ④
            version:  <maven_version_number> 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, Strimzi will verify the checksum of the artifact while building the new container image.

Example `other` artifact

```
apiVersion: kafka.strimzi.io/v1
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.

101.3. `Build` schema properties

Property	Property type	Description
output	`DockerOutput`, `ImageStreamOutput`	Configures where should the newly built image be stored. Required.
plugins	`Plugin` array	List of connector plugins which should be added to the Kafka Connect. Required.
resources	ResourceRequirements	CPU and memory resources to reserve for the build.

Chapter 102. 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	Property type	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	**The additionalKanikoOptions property has been deprecated, and should now be configured using .spec.build.output.additionalBuildOptions.** The additionalKanikoOptions configuration is deprecated and will be removed in the v1 CRD API. Configures additional options which will be passed to the Kaniko executor when building the new Connect image. Allowed options are: --customPlatform, --custom-platform, --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, --registry-certificate, --registry-client-cert, --ignore-path. These options will be used only on Kubernetes 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.

Property	Property type	Description
additionalBuildOptions	string array	Configures additional options to pass to the `build` command of either Kaniko or Buildah (depending on the feature gate setting) when building a new Kafka Connect image. Allowed Kaniko options: --customPlatform, --custom-platform, --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, --registry-certificate, --registry -client-cert, --ignore-path. Allowed Buildah `build` options: --authfile, --cert -dir, --creds, --decryption-key, --retry, --retry-delay, --tls-verify. Those options are used only on Kubernetes, where Kaniko and Buildah are available. They are ignored on OpenShift. For more information, see the Kaniko GitHub repository or the Buildah build document. Changing this field does not trigger a rebuild of the Kafka Connect image.
additionalPushOptions	string array	Configures additional options to pass to the Buildah `push` command when pushing a new Connect image. Allowed options: --authfile, --cert-dir, --creds, --quiet, --retry, --retry-delay, --tls-verify. Those options are used only on Kubernetes, where Buildah is available. They are ignored on OpenShift. For more information, see the Buildah push document. Changing this field does not trigger a rebuild of the Kafka Connect image.
type	string	Must be `docker`.

Chapter 103. 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	Property type	Description
type	string	Must be imagestream.
image	string	The name and tag of the ImageStream where the newly built image will be pushed. For example my-custom-connect:latest. Required.

Chapter 104. `Plugin` schema reference

Used in: `Build`

Property	Property type	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 105. `JarArtifact` schema reference

Used in: `Plugin`

Property	Property type	Description
type	string	Must be `jar`.
url	string	URL of the artifact which will be downloaded. Strimzi 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.

Chapter 106. `TgzArtifact` schema reference

Used in: `Plugin`

Property	Property type	Description
type	string	Must be `tgz`.
url	string	URL of the artifact which will be downloaded. Strimzi 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.

Chapter 107. `ZipArtifact` schema reference

Used in: `Plugin`

Property	Property type	Description
type	string	Must be `zip`.
url	string	URL of the artifact which will be downloaded. Strimzi 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.

Chapter 108. `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	Property type	Description
type	string	Must be `maven`.
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.

Chapter 109. `OtherArtifact` schema reference

Used in: `Plugin`

Property	Property type	Description
type	string	Must be `other`.
url	string	URL of the artifact which will be downloaded. Strimzi 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.

Chapter 110. MountedPlugin schema reference

Used in: KafkaConnectSpec

Full list of MountedPlugin schema properties

IMPORTANT This feature requires Kubernetes support for the Image Volume feature.

110.1. MountedPlugin schema properties

Property	Property type	Description
name	string	A unique name for the connector plugin. This name is used to generate the mount path for the connector artifacts. The name has to be unique within the KafkaConnect resource. The name must be unique within the KafkaConnect resource and match the pattern: ^[a-z][-_a-z0-9]*[a-z]$. Required.
artifacts	ImageArtifact array	List of artifacts associated with this connector plugin. Required.

Chapter 111. `ImageArtifact` schema reference

Used in: `MountedPlugin`

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

Property	Property type	Description
type	string	Must be `image`.
reference	string	Reference to the container image (OCI artifact) containing the Kafka Connect plugin. The image is mounted as a volume and provides the plugin binary. Required.
pullPolicy	string	Policy that determines when the container image (OCI artifact) is pulled. Possible values are: • `Always`: Always pull the image. If the pull fails, container creation fails. • `Never`: Never pull the image. Use only a locally available image. Container creation fails if the image isn't present. • `IfNotPresent`: Pull the image only if it's not already available locally. Container creation fails if the image isn't present and the pull fails. Defaults to `Always` if `:latest` tag is specified, or `IfNotPresent` otherwise.

Chapter 112. KafkaConnectStatus schema reference

Used in: KafkaConnect

Property	Property type	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.
replicas	integer	The current number of pods being used to provide this resource.
labelSelector	string	Label selector for pods providing this resource.

Chapter 113. `ConnectorPlugin` schema reference

Used in: `KafkaConnectStatus`, `KafkaMirrorMaker2Status`

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

Chapter 114. KafkaTopic schema reference

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

Chapter 115. KafkaTopicSpec schema reference

Used in: KafkaTopic

Property	Property type	Description
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 Kubernetes resource name.
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.

Chapter 116. KafkaTopicStatus schema reference

Used in: KafkaTopic

Property	Property type	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.
topicId	string	The topic's id. For a KafkaTopic with the ready condition, this will change only if the topic gets deleted and recreated with the same name.
replicasChange	ReplicasChangeStatus	Replication factor change status.

Chapter 117. ReplicasChangeStatus schema reference

Used in: KafkaTopicStatus

Property	Property type	Description
targetReplicas	integer	The target replicas value requested by the user. This may be different from .spec.replicas when a change is ongoing.
state	string (one of [ongoing, pending])	Current state of the replicas change operation. This can be pending, when the change has been requested, or ongoing, when the change has been successfully submitted to Cruise Control.
message	string	Message for the user related to the replicas change request. This may contain transient error messages that would disappear on periodic reconciliations.
sessionId	string	The session identifier for replicas change requests pertaining to this KafkaTopic resource. This is used by the Topic Operator to track the status of ongoing replicas change operations.

Chapter 118. KafkaUser schema reference

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

Chapter 119. KafkaUserSpec schema reference

Used in: KafkaUser

Property	Property type	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.
authorization	KafkaUserAuthorizationSimple	Authorization rules for this Kafka user.
quotas	KafkaUserQuotas	Quotas on requests to control the broker resources used by clients. Network bandwidth and request rate quotas can be enforced. For more information, see the Apache Kafka design documentation about quotas.
template	KafkaUserTemplate	Template to specify how Kafka User Secrets are generated.

Chapter 120. 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	Property type	Description
type	string	Must be tls.

Chapter 121. 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	Property type	Description
type	string	Must be tls-external.

Chapter 122. 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	Property type	Description
type	string	Must be scram-sha-512.
password	Password	Specify the password for the user. If not set, a new password is generated by the User Operator.

Chapter 123. `Password` schema reference

Used in: `KafkaUserScramSha512ClientAuthentication`

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

Chapter 124. 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	Property type	Description
type	string	Must be simple.
acls	AclRule array	List of ACL rules which should be applied to this user.

Chapter 125. 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 simple authorization

```
apiVersion: kafka.strimzi.io/v1
kind: KafkaUser
metadata:
  name: my-user
  labels:
    strimzi.io/cluster: my-cluster
spec:
  # ...
  authorization:
    type: simple
    acls:
      - resource:
          type: topic
          name: "*"
          patternType: literal
        operations:
          - Read
          - Describe
      - resource:
          type: group
          name: my-group
          patternType: prefix
        operations:
          - Read
```

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.

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

125.1. `AclRule` schema properties

Property	Property type	Description
type	string (one of [allow, deny])	The type of the rule. ACL rules with type `allow` are used to allow user to execute the specified operations. ACL rules with type `deny` are used to deny user to execute the specified operations. Default value is `allow`.
resource	`AclRuleTopicResource`, `AclRuleGroupResource`, `AclRuleClusterResource`, `AclRuleTransactionalIdResource`	Indicates the resource for which given ACL rule applies.
host	string	The host from which the action described in the ACL rule is allowed or denied. If not set, it defaults to `*`, allowing or denying the action from any host.
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.

Property	Property type	Description
operations	string (one or more of [Read, Write, Delete, Alter, Describe, All, IdempotentWrite, ClusterAction, Create, AlterConfigs, DescribeConfigs]) array	List of operations to allow or deny. Supported operations are: Read, Write, Create, Delete, Alter, Describe, ClusterAction, AlterConfigs, DescribeConfigs, IdempotentWrite and All. Only certain operations work with the specified resource.

Chapter 126. 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	Property type	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 127. 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	Property type	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 128. `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	Property type	Description
type	string	Must be `cluster`.

Chapter 129. 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	Property type	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 130. KafkaUserQuotas schema reference

Used in: KafkaUserSpec

Full list of KafkaUserQuotas schema properties

Configure clients to use quotas so that a user does not overload Kafka brokers.

Example Kafka user quota configuration

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

For more information, see the Apache Kafka design documentation.

130.1. KafkaUserQuotas schema properties

Property	Property type	Description
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.
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.
requestPercentage	integer	A quota on the maximum CPU utilization of each client group as a percentage of network and I/O threads.
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.

Chapter 131. 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/v1
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
  # ...
```

131.1. KafkaUserTemplate schema properties

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

Chapter 132. KafkaUserStatus schema reference

Used in: KafkaUser

Property	Property type	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 133. KafkaBridge schema reference

Property	Property type	Description
spec	KafkaBridgeSpec	The specification of the HTTP Bridge.
status	KafkaBridgeStatus	The status of the HTTP Bridge.

Chapter 134. KafkaBridgeSpec schema reference

Used in: KafkaBridge

Full list of KafkaBridgeSpec schema properties

Configures a HTTP Bridge cluster.

Configuration options relate to:

- Kafka cluster bootstrap address
- Security (encryption, authentication, and authorization)
- Consumer configuration
- Producer configuration
- HTTP configuration

134.1. Logging

HTTP Bridge has its own preconfigured loggers:

Logger	Description	Default Level
rootLogger	Default logger for all classes	INFO
bridge	All classes in the io.strimzi.kafka.bridge package	INFO
healthy	Uses http.openapi.operation.healthy endpoint	WARN
ready	Uses http.openapi.operation.ready endpoint	WARN

You can configure log levels directly for these loggers. For example:

```
apiVersion: kafka.strimzi.io/v1
kind: KafkaBridge
spec:
  # ...
  logging:
    type: inline
    loggers:
      rootLogger.level: INFO
      logger.ready.level: DEBUG
  # ...
```

You can also specify custom loggers for other operations:

- createConsumer

- `deleteConsumer`
- `subscribe`
- `unsubscribe`
- `poll`
- `assign`
- `commit`
- `send`
- `sendToPartition`
- `seekToBeginning`
- `seekToEnd`
- `seek`
- `openapi`

Each operation maps to an API endpoint, defined according to the OpenAPI specification, and supports fine-grained logging via `http.openapi.operation.<operation_id>`.

For example, to set the logging level for the `send` operation logger:

```
apiVersion: kafka.strimzi.io/v1
kind: KafkaBridge
spec:
  # ...
  logging:
    type: inline
    loggers:
      logger.send.name: http.openapi.operation.send
      logger.send.level: DEBUG
  # ...
```

HTTP Bridge uses the Apache `log4j2` logger implementation. Use the `logging` property to configure loggers and logger levels.

You can set log levels using either the `inline` or `external` logging configuration types.

Specify loggers and levels directly in the custom resource for inline configuration:

Example inline logging configuration

```
apiVersion: kafka.strimzi.io/v1
kind: KafkaBridge
spec:
  # ...
  logging:
    type: inline
    loggers:
```

```
      rootLogger.level: INFO
      # enabling DEBUG for send operation
      logger.send.name: http.openapi.operation.send
      logger.send.level: DEBUG
  # ...
```

Alternatively, you can reference an external `ConfigMap` containing a complete `log4j2.properties` file that defines your own log4j2 configuration, including loggers, appenders, and layout configuration:

Example external logging configuration

```
apiVersion: kafka.strimzi.io/v1
kind: KafkaBridge
spec:
  # ...
  logging:
    type: external
    valueFrom:
      configMapKeyRef:
        # name and key are mandatory
        name: customConfigMap
        key: log4j2.properties
  # ...
```

Garbage collector (GC)

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

134.2. `KafkaBridgeSpec` schema properties

Property	Property type	Description
replicas	integer	The number of pods in the `Deployment`. Required in the `v1` version of the Strimzi API. Defaults to `1` in the `v1beta2` version of the Strimzi API.
image	string	The container image used for HTTP Bridge pods. If no image name is explicitly specified, the image name corresponds to the image specified in the Cluster Operator configuration. If an image name is not defined in the Cluster Operator configuration, a default value is used.
bootstrapServers	string	A list of host:port pairs for establishing the initial connection to the Kafka cluster.

Property	Property type	Description
tls	ClientTls	TLS configuration for connecting HTTP Bridge to the cluster.
authentication	KafkaClientAuthenticationTls, KafkaClientAuthenticationScramSha256, KafkaClientAuthenticationScramSha512, KafkaClientAuthenticationPlain, KafkaClientAuthenticationOAuth, KafkaClientAuthenticationCustom	Authentication configuration for connecting to the cluster.
http	KafkaBridgeHttpConfig	The HTTP related configuration.
adminClient	KafkaBridgeAdminClientSpec	Kafka AdminClient related configuration.
consumer	KafkaBridgeConsumerSpec	Kafka consumer related configuration.
producer	KafkaBridgeProducerSpec	Kafka producer related configuration.
resources	ResourceRequirements	CPU and memory resources to reserve.
jvmOptions	JvmOptions	JVM Options for pods.
logging	InlineLogging, ExternalLogging	Logging configuration for HTTP Bridge.
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.
enableMetrics	boolean	**The enableMetrics property has been deprecated, and should now be configured using .spec.metricsConfig.** The enableMetrics configuration is deprecated and will be removed in the future. Enable the metrics for the HTTP Bridge. Default is false.
metricsConfig	JmxPrometheusExporterMetrics, StrimziMetricsReporter	Metrics configuration.
livenessProbe	Probe	Pod liveness checking.
readinessProbe	Probe	Pod readiness checking.
template	KafkaBridgeTemplate	Template for HTTP Bridge resources. The template allows users to specify how a Deployment and Pod is generated.

Property	Property type	Description
tracing	`JaegerTracing`, `OpenTelemetryTracing`	The configuration of tracing in HTTP Bridge.

Chapter 135. KafkaBridgeHttpConfig schema reference

Used in: KafkaBridgeSpec

Full list of KafkaBridgeHttpConfig schema properties

Configures HTTP access to a Kafka cluster for the HTTP Bridge. The default HTTP configuration is for the HTTP Bridge to listen on port 8080.

Example HTTP Bridge HTTP configuration

```
apiVersion: kafka.strimzi.io/v1
kind: KafkaBridge
metadata:
  name: my-bridge
spec:
  # ...
  http:
    port: 8080
    cors:
      allowedOrigins: "https://strimzi.io"
      allowedMethods: "GET,POST,PUT,DELETE,OPTIONS,PATCH"
  # ...
```

As well as enabling HTTP access to a Kafka cluster, HTTP properties provide the capability to enable and define access control for the HTTP 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.

135.1. KafkaBridgeHttpConfig schema properties

Property	Property type	Description
port	integer	The port which is the server listening on.
cors	KafkaBridgeHttpCors	CORS configuration for the HTTP Bridge.

Chapter 136. KafkaBridgeHttpCors schema reference

Used in: KafkaBridgeHttpConfig

Property	Property type	Description
allowedOrigins	string array	List of allowed origins. Java regular expressions can be used.
allowedMethods	string array	List of allowed HTTP methods.

Chapter 137. KafkaBridgeAdminClientSpec schema reference

Used in: KafkaBridgeSpec

Property	Property type	Description
config	map	The Kafka AdminClient configuration used for AdminClient instances created by the bridge.

Chapter 138. KafkaBridgeConsumerSpec schema reference

Used in: KafkaBridgeSpec

Full list of KafkaBridgeConsumerSpec schema properties

Configures consumer options for the HTTP Bridge.

Example HTTP Bridge consumer configuration

```
apiVersion: kafka.strimzi.io/v1
kind: KafkaBridge
metadata:
  name: my-bridge
spec:
  # ...
  consumer:
    enabled: true
    timeoutSeconds: 60
    config:
      auto.offset.reset: earliest
      enable.auto.commit: true
    # ...
```

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, Strimzi 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 HTTP Bridge, including the following exceptions to the options configured by Strimzi:

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

IMPORTANT

The Cluster Operator does not validate keys or values in the `config` object. If an invalid configuration is provided, the HTTP 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 HTTP Bridge nodes.

138.1. `KafkaBridgeConsumerSpec` schema properties

Property	Property type	Description
enabled	boolean	Whether the HTTP consumer should be enabled or disabled. The default is enabled (`true`).
timeoutSeconds	integer	The timeout in seconds for deleting inactive consumers, default is -1 (disabled).
config	map	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).

Chapter 139. KafkaBridgeProducerSpec schema reference

Used in: KafkaBridgeSpec

Full list of KafkaBridgeProducerSpec schema properties

Configures producer options for the HTTP Bridge.

Example HTTP Bridge producer configuration

```
apiVersion: kafka.strimzi.io/v1
kind: KafkaBridge
metadata:
  name: my-bridge
spec:
  # ...
  producer:
    enabled: true
    config:
      acks: 1
      delivery.timeout.ms: 300000
    # ...
```

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, Strimzi 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 HTTP Bridge, including the following exceptions to the options configured by Strimzi:

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

IMPORTANT	The Cluster Operator does not validate the keys or values of `config` properties. If an invalid configuration is provided, the HTTP 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 HTTP Bridge nodes.

139.1. `KafkaBridgeProducerSpec` schema properties

Property	Property type	Description
enabled	boolean	Whether the HTTP producer should be enabled or disabled. The default is enabled (`true`).
config	map	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).

Chapter 140. KafkaBridgeTemplate schema reference

Used in: KafkaBridgeSpec

Property	Property type	Description
deployment	DeploymentTemplate	Template for HTTP Bridge Deployment.
pod	PodTemplate	Template for HTTP Bridge Pods.
apiService	InternalServiceTemplate	Template for HTTP Bridge API Service.
podDisruptionBudget	PodDisruptionBudgetTemplate	Template for HTTP Bridge PodDisruptionBudget.
bridgeContainer	ContainerTemplate	Template for the HTTP Bridge container.
clusterRoleBinding	ResourceTemplate	Template for the HTTP Bridge ClusterRoleBinding.
serviceAccount	ResourceTemplate	Template for the HTTP Bridge service account.
initContainer	ContainerTemplate	Template for the HTTP Bridge init container.

Chapter 141. KafkaBridgeStatus schema reference

Used in: KafkaBridge

Property	Property type	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 at which external client applications can access the HTTP Bridge.
replicas	integer	The current number of pods being used to provide this resource.
labelSelector	string	Label selector for pods providing this resource.

Chapter 142. KafkaConnector schema reference

Property	Property type	Description
spec	KafkaConnectorSpec	The specification of the Kafka Connector.
status	KafkaConnectorStatus	The status of the Kafka Connector.

Chapter 143. KafkaConnectorSpec schema reference

Used in: KafkaConnector

Property	Property type	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.
version	string	Desired version or version range to respect when starting the Kafka Connector. This is only supported when using Kafka Connect version 4.1.0 and higher.
config	map	The Kafka Connector configuration. The following properties cannot be set: name, connector.class, tasks.max, connector.plugin.version.
pause	boolean	**The pause property has been deprecated.** Deprecated in Strimzi 0.38.0, 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.
listOffsets	ListOffsets	Configuration for listing offsets.
alterOffsets	AlterOffsets	Configuration for altering offsets.

Chapter 144. 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. The restart counter is automatically reset once the connector has been running continuously for longer than the next backoff interval. For example, if the connector has restarted 4 times, it must run without errors for 20 minutes before the restart count resets to 0. By default, Strimzi 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/v1
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/v1
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/v1
kind: KafkaMirrorMaker2
metadata:
  name: my-mm2-cluster
spec:
  mirrors:
  - sourceConnector:
      autoRestart:
        enabled: true
      # ...
    checkpointConnector:
      autoRestart:
        enabled: true
      # ...
```

144.1. `AutoRestart` schema properties

Property	Property type	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 145. `ListOffsets` schema reference

Used in: `KafkaConnectorSpec`, `KafkaMirrorMaker2ConnectorSpec`

Property	Property type	Description
toConfigMap	`LocalObjectReference`	Reference to the ConfigMap where the list of offsets will be written to.

Chapter 146. AlterOffsets schema reference

Used in: KafkaConnectorSpec, KafkaMirrorMaker2ConnectorSpec

Property	Property type	Description
fromConfigMap	LocalObjectReference	Reference to the ConfigMap where the new offsets are stored.

Chapter 147. KafkaConnectorStatus schema reference

Used in: KafkaConnector

Property	Property type	Description
conditions	Condition array	List of status conditions.
observedGeneration	integer	The generation of the CRD that was last reconciled by the operator.
autoRestart	AutoRestartStatus	The auto restart status.
connectorStatus	map	The connector status, as reported by the Kafka Connect REST API.
tasksMax	integer	The maximum number of tasks for the Kafka Connector.
topics	string array	The list of topics used by the Kafka Connector.

Chapter 148. AutoRestartStatus schema reference

Used in: KafkaConnectorStatus, KafkaMirrorMaker2Status

Property	Property type	Description
count	integer	The number of times the connector or task is restarted.
connectorName	string	The name of the connector being restarted.
lastRestartTimestamp	string	The last time the automatic restart was attempted. The required format is 'yyyy-MM-ddTHH:mm:ssZ' in the UTC time zone.

Chapter 149. KafkaMirrorMaker2 schema reference

Property	Property type	Description
spec	KafkaMirrorMaker2Spec	The specification of the Kafka MirrorMaker 2 cluster.
status	KafkaMirrorMaker2Status	The status of the Kafka MirrorMaker 2 cluster.

Chapter 150. KafkaMirrorMaker2Spec schema reference

Used in: KafkaMirrorMaker2

Property	Property type	Description
version	string	The Kafka Connect version. Defaults to the latest version. 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. Required in the v1 version of the Strimzi API. Defaults to 3 in the v1beta2 version of the Strimzi API.
image	string	The container image used for Kafka Connect pods. If no image name is explicitly specified, it is determined based on the spec.version configuration. The image names are specifically mapped to corresponding versions in the Cluster Operator configuration.
connectCluster	string	**The connectCluster property has been deprecated, and should now be configured using .spec.target.** The connectCluster property is deprecated and will be removed in the v1 CRD API. 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	**The clusters property has been deprecated.** The clusters section is deprecated and will be removed in the v1 CRD API. Please use the .spec.target and .spec.mirrors[].source sections instead. Kafka clusters for mirroring.
target	KafkaMirrorMaker2TargetClusterSpec	The target Apache Kafka cluster. The target Kafka cluster is used by the underlying Kafka Connect framework for its internal topics.

Property	Property type	Description
mirrors	KafkaMirrorMaker2MirrorSpec array	Configuration of the MirrorMaker 2 connectors.
resources	ResourceRequirements	The maximum limits for CPU and memory resources and the requested initial resources.
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.
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.
metricsConfig	JmxPrometheusExporterMetrics, StrimziMetricsReporter	Metrics configuration.
tracing	JaegerTracing, OpenTelemetryTracing	The configuration of tracing in Kafka Connect.
template	KafkaConnectTemplate	Template for Kafka Connect and Kafka MirrorMaker 2 resources. The template allows users to specify how the Pods, Service, and other services are generated.
externalConfiguration	ExternalConfiguration	**The externalConfiguration property has been deprecated.** The external configuration is deprecated and will be removed in the future. Please use the template section instead to configure additional environment variables or volumes. Pass data from Secrets or ConfigMaps to the Kafka Connect pods and use them to configure connectors.

Chapter 151. KafkaMirrorMaker2ClusterSpec schema reference

Used in: KafkaMirrorMaker2MirrorSpec, KafkaMirrorMaker2Spec

Full list of KafkaMirrorMaker2ClusterSpec schema properties

Configures Kafka clusters for mirroring.

Use the config properties to configure Kafka options, restricted to those properties not managed directly by Strimzi.

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.

151.1. KafkaMirrorMaker2ClusterSpec schema properties

Property	Property type	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, KafkaClientAuthenticationCustom	Authentication configuration for connecting to the cluster.

Property	Property type	Description
config	map	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).

Chapter 152. KafkaMirrorMaker2TargetClusterSpec schema reference

Used in: KafkaMirrorMaker2Spec

Full list of KafkaMirrorMaker2TargetClusterSpec schema properties

Configures the target Apache Kafka cluster for a MirrorMaker 2 deployment. The target Kafka cluster is used by the underlying Kafka Connect framework for its internal topics.

152.1. Group ID and internal topics

The group ID and the internal topics used by the Kafka Connect cluster are configured in the `.spec.target` section of the KafkaMirrorMaker2 resource. These properties are required.

Example group ID and internal topics configuration

```
apiVersion: kafka.strimzi.io/v1
kind: KafkaMirrorMaker2
metadata:
  name: my-mirrormaker-2
spec:
  # ...
  target:
    # ...
    groupId: my-connect-group
    configStorageTopic: my-config-topic
    offsetStorageTopic: my-offset-topic
    statusStorageTopic: my-status-topic
  # ...
```

152.2. Additional configuration

Use the `config` properties to configure Kafka options, restricted to those properties not managed directly by Strimzi.

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.

Example target cluster configuration

```
apiVersion: kafka.strimzi.io/v1
kind: KafkaMirrorMaker2
metadata:
```

```
  name: my-mirrormaker-2
spec:
  # ...
  target:
    # ...
    config:
      config.storage.replication.factor: -1
      offset.storage.replication.factor: -1
      status.storage.replication.factor: -1
  # ...
```

152.3. KafkaMirrorMaker2TargetClusterSpec schema properties

Property	Property type	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.
groupId	string	A unique ID that identifies the Connect cluster group. Required.
configStorageTopic	string	The name of the Kafka topic where connector configurations are stored. Required.
statusStorageTopic	string	The name of the Kafka topic where connector and task statuses are stored. Required.
offsetStorageTopic	string	The name of the Kafka topic where source connector offsets are stored. Required.
tls	ClientTls	TLS configuration for connecting MirrorMaker 2 connectors to a cluster.
authentication	KafkaClientAuthenticationTls, KafkaClientAuthenticationScramSha256, KafkaClientAuthenticationScramSha512, KafkaClientAuthenticationPlain, KafkaClientAuthenticationOAuth, KafkaClientAuthenticationCustom	Authentication configuration for connecting to the cluster.

Property	Property type	Description
config	map	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).

Chapter 153. KafkaMirrorMaker2MirrorSpec schema reference

Used in: KafkaMirrorMaker2Spec

Property	Property type	Description
sourceCluster	string	**The sourceCluster property has been deprecated, and should now be configured using .spec.mirrors.source.** The sourceCluster property is deprecated and will be removed in the v1 CRD API. 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 targetCluster property has been deprecated, and should now be configured using .spec.target.** The targetCluster property is deprecated and will be removed in the v1 CRD API. 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.
source	KafkaMirrorMaker2ClusterSpec	The source Apache Kafka cluster. The source Kafka cluster is used by the Kafka MirrorMaker 2 connectors.
sourceConnector	KafkaMirrorMaker2ConnectorSpec	The specification of the Kafka MirrorMaker 2 source connector.
heartbeatConnector	KafkaMirrorMaker2ConnectorSpec	**The heartbeatConnector property has been deprecated.** The heartbeatConnector property is deprecated and will be removed in the v1 CRD API. 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.

Property	Property type	Description
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.
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 154. KafkaMirrorMaker2ConnectorSpec schema reference

Used in: KafkaMirrorMaker2MirrorSpec

Property	Property type	Description
tasksMax	integer	The maximum number of tasks for the Kafka Connector.
pause	boolean	**The pause property has been deprecated.** Deprecated in Strimzi 0.38.0, use state instead. Whether the connector should be paused. Defaults to false.
version	string	Desired version or version range to respect when starting the Kafka Connector. This is only supported when using Kafka Connect version 4.1.0 and higher.
config	map	The Kafka Connector configuration. The following properties cannot be set: name, connector.class, tasks.max, connector.plugin.version.
state	string (one of [running, paused, stopped])	The state the connector should be in. Defaults to running.
autoRestart	AutoRestart	Automatic restart of connector and tasks configuration.
listOffsets	ListOffsets	Configuration for listing offsets.
alterOffsets	AlterOffsets	Configuration for altering offsets.

Chapter 155. KafkaMirrorMaker2Status schema reference

Used in: KafkaMirrorMaker2

Property	Property type	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.
connectors	map array	List of MirrorMaker 2 connector statuses, as reported by the Kafka Connect REST API.
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.
labelSelector	string	Label selector for pods providing this resource.
replicas	integer	The current number of pods being used to provide this resource.

Chapter 156. KafkaRebalance schema reference

Property	Property type	Description
spec	KafkaRebalanceSpec	The specification of the Kafka rebalance.
status	KafkaRebalanceStatus	The status of the Kafka rebalance.

Chapter 157. KafkaRebalanceSpec schema reference

Used in: KafkaRebalance

Property	Property type	Description
mode	string (one of [remove-disks, 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. • `remove-disks` mode can be used to move data across the volumes within the same broker .
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.

Property	Property type	Description
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.
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.
moveReplicasOffVolumes	`BrokerAndVolumeIds` array	List of brokers and their corresponding volumes from which replicas need to be moved.

Chapter 158. `BrokerAndVolumeIds` schema reference

Used in: `KafkaRebalanceSpec`

Property	Property type	Description
brokerId	integer	ID of the broker that contains the disk from which you want to move the partition replicas.
volumeIds	integer array	IDs of the disks from which the partition replicas need to be moved.

Chapter 159. KafkaRebalanceStatus schema reference

Used in: KafkaRebalance

Property	Property type	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.
progress	KafkaRebalanceProgress	A reference to Config Map with the progress information.
optimizationResult	map	A JSON object describing the optimization result.

Chapter 160. KafkaRebalanceProgress schema reference

Used in: KafkaRebalanceStatus

Property	Property type	Description
rebalanceProgressConfigMap	string	The name of the ConfigMap containing information related to the progress of a partition rebalance.

Chapter 161. KafkaNodePool schema reference

Property	Property type	Description
spec	KafkaNodePoolSpec	The specification of the KafkaNodePool.
status	KafkaNodePoolStatus	The status of the KafkaNodePool.

Chapter 162. KafkaNodePoolSpec schema reference

Used in: KafkaNodePool

Property	Property type	Description
replicas	integer	The number of pods in the pool.
storage	EphemeralStorage, PersistentClaimStorage, JbodStorage	Storage configuration (disk). Cannot be updated.
roles	string (one or more of [controller, broker]) array	The roles assigned to the node pool. Supported values are broker and controller. This property is required.
resources	ResourceRequirements	CPU and memory resources to reserve.
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 163. KafkaNodePoolTemplate schema reference

Used in: KafkaNodePoolSpec

Property	Property type	Description
podSet	ResourceTemplate	Template for Kafka StrimziPodSet resource.
pod	PodTemplate	Template for Kafka Pods.
perPodService	ResourceTemplate	Template for Kafka per-pod Services used for access from outside of Kubernetes.
perPodRoute	ResourceTemplate	Template for Kafka per-pod Routes used for access from outside of OpenShift.
perPodIngress	ResourceTemplate	Template for Kafka per-pod Ingress used for access from outside of Kubernetes.
persistentVolumeClaim	ResourceTemplate	Template for all Kafka PersistentVolumeClaims.
kafkaContainer	ContainerTemplate	Template for the Kafka broker container.
initContainer	ContainerTemplate	Template for the Kafka init container.

Chapter 164. KafkaNodePoolStatus schema reference

Used in: KafkaNodePool

Property	Property type	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.
roles	string (one or more of [controller, broker]) array	Added in Strimzi 0.39.0. The roles currently assigned to this pool.
replicas	integer	The current number of pods being used to provide this resource.
labelSelector	string	Label selector for pods providing this resource.

Chapter 165. StrimziPodSet schema reference

Full list of StrimziPodSet schema properties

IMPORTANT

StrimziPodSet is an internal Strimzi resource. Information is provided for reference only. Do not create, modify or delete StrimziPodSet resources as this might cause errors.

165.1. StrimziPodSet schema properties

Property	Property type	Description
spec	StrimziPodSetSpec	The specification of the StrimziPodSet.
status	StrimziPodSetStatus	The status of the StrimziPodSet.

Chapter 166. `StrimziPodSetSpec` schema reference

Used in: `StrimziPodSet`

Property	Property type	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.
pods	Map array	The Pods managed by this StrimziPodSet.

Chapter 167. StrimziPodSetStatus schema reference

Used in: StrimziPodSet

Property	Property type	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.