AMQ Streams command cheat sheet

[Introduction](#_2rgt7a3rt5al)

[Adjust environment variables](#_50h7lw5cpwu6)

[Kafka Cluster and Kafka Broker](#_hrq5914qrz5e)

[Kafka cluster definition](#_uim135l9l0j)

[Broker configurations](#_44hz026la718)

[Show cluster-wide finalized features and features supported by a specific broker](#_noxoq56syx97)

[Show the controller Broker ID](#_2ybrzlq9vq44)

[Change the controller broker to others](#_qcadk6r3f3zx)

[Living Kafka broker list using Zookeeper](#_9rs7fg8mzn22)

[Kafka broker information on Zookeeper using Zookeeper](#_nrdolkmdshlg)

[Delete and recreate a Kafka pod and its PVC](#_a6v3pfslz4rm)

[Manual rolling update: Kafka brokers](#_1ltthpzebjuo)

[Zookeeper](#_9ncxoctmb9zu)

[Zookeeper definition](#_lto8362es38y)

[Zookeeper interactive command line](#_xs5tw91tk1h5)

[Status and environment of Zookeeper](#_k9hj6zvrhbnc)

[Dump sessions of zookeeper](#_9t1mg7vznti2)

[Manual rolling update: Zookeeper](#_2p1f2acxyw1x)

[Access Zookeepers directly for debugging : since AMQ Streams 1.6](#_84444swmglyu)

[Create a zoo-entrance to access Zookeepers directly for debugging : until AMQ Streams 1.5](#_ec5vs7lhfhnt)

[Show internal zookeeper snapshot and zookeeper log](#_hdetv5tgbqoj)

[Kafka Topic](#_hrzmakv1nf5f)

[Kafka topic definitions](#_j3f7rxb1tqmp)

[Actual Kafka topic list](#_7pixjqf1ydtw)

[Actual Kafka topic description](#_9fbewkeop154)

[Kafka topic configurations](#_yalrw55m3wnh)

[Create a Kafka topic (replication factor: 3, partitions: 5)](#_a6ggjxidy0rx)

[Delete a Kafka topic](#_jegw7b563etu)

[Topic latest offset (for each partition)](#_vgkqpwi1mlw9)

[Topic earliest offset (for each partition)](#_6ne74x3sd0v)

[Topic size (based on offsets of total partitions)](#_ll2kis8ckmy3)

[Creation timestamp of a Kafka topic](#_91swgoelm0gc)

[Disable topic deletion](#_4hys8kp9xwzb)

[Kafka Internal Topic](#_1pdqxma9txl8)

[Show \_\_consumer\_offsets contents](#_g6w0urt5w1zq)

[Show \_\_transaction\_state contents](#_slgwylrtq60p)

[Strimzi Internal Topic (since AMQ Streams 1.7)](#_xroayb78bzfq)

[Show \_\_strimzi\_store\_topic](#_nkgjj6rv4mnn)

[Show \_\_strimzi-topic-operator-kstreams-topic-store-changelog](#_vvfhefst7jg7)

[Kafka Partition and Replica](#_p01s9oy0gx51)

[Log directories and partitions](#_cfgzgykp89xx)

[Reassign partitions : since AMQ Streams 1.6](#_3nkf7dqr8g3n)

[Reassign partitions : until AMQ Streams 1.5](#_5si2j1rt14fb)

[Stop Reassigning partitions](#_7178hrc946gd)

[Topics that have under-replicated-partitions](#_so6xpbrihxg)

[Topics that have unavailable-partitions](#_ftei734er6rn)

[Preferred replica election](#_96pv85775og6)

[Kafka Producer](#_65vtazt9wzs1)

[Console producer](#_mgx3fc2migci)

[Producer performance test tool (size:10, throughput:1, records:1000)](#_ycj14eywqdz5)

[Kafka Consumer and Consumer Group](#_h9w6oq88g7zg)

[Console consumer (without a consumer group)](#_xjjep0bb0yen)

[Console consumer (with a consumer group)](#_jspq4hvqzdl)

[Consumer performance test tool](#_xe8c0kclthtu)

[Consumer group list](#_bbrsd3u4akj4)

[Consumer group description (--offset)](#_sd7zvitkerz5)

[Consumer group description (--state)](#_10bt709i0it3)

[Delete a consumer group](#_dc7t07foxae2)

[Reset the offsets for a specific consumer group](#_mymqpxtl1m3j)

[Delete offset with specific topic for a specific consumer group](#_t4l98shhrobb)

[About Kafka CLI execution environment](#_a2cj1cj1yivd)

[Executing Kafka CLI using an existing broker](#_w7z3pje6ebkv)

[Executing Kafka CLI using a new pod (do not use a pod of an existing broker)](#_gnjf6j3uohke)

[Executing Kafka CLI using a customized new pod as root](#_opvyuon5w28z)

[Strimzi Cluster Operator](#_q7kpsehn4fps)

[Stop cluster operator](#_5irmtowlcm42)

[Pause reconciliation of custom resources](#_p5iu8vpdsu9a)

[Check for duplicate cluster operators installed](#_6nkckxnvjoj0)

[Strimzi Entity Operator (comprises Topic Operator and User Operator)](#_o4bs6tp5utzk)

[Stop entity operator](#_gglj5cvjbmoh)

[Kafka Connect and Debezium](#_y4mf9apjl0rc)

[Create a MySQL DB POD for test](#_byw5rx6aliop)

[Create Kafka Connect image with Debezium: build a image on the OpenShift](#_1x6xl5ukcffj)

[Create a MySQL Debezium connector accessing to MySQL DB on the Kafka Connector POD](#_cgac8prvefv3)

[Cruise Control](#_novq6uank87g)

[Cruise Control definition](#_3f0gw0bnabq6)

[TLS](#_6us8alovic1v)

[Validity days of Cluster CA and Clients CA certifications in secrets](#_452x3mjnlfr9)

[Check actual TLS info using openssl s\_client command](#_bj8m608k239z)

[Check TLS port on a Kafka broker using openssl s\_client command](#_hlqbtrdye5tm)

[Enable ssl debug log on Kafka brokers](#_uw3bu1qkgwis)

[Renewing CA certificates manually](#_42drn4xnmjr6)

[Replace to my own cluster ca certification and its key](#_ud6h30mi4449)

[Enable SSL debug log of Kafka CLI for debugging](#_m7bnin83jgja)

[Cluster CA and truststore.p12](#_hbr7rtp1p2pe)

[KafkaUser secret and keystore.p12](#_qyy1l1ll1lmm)

[tls listener (port:9093) with one way TLS](#_fpmda74rh7lw)

[Prerequisites](#_o95gcl2f6jzi)

[Console producer](#_tfgd5n2f62uv)

[Producer performance test tool (size:10, throughput:1, records:1000)](#_7vs698hqbgus)

[Console consumer](#_191i48wpy3pv)

[tls listener (port:9093) with two-way(mutual) TLS client authentication and simple authorization](#_nv4zctr6da3f)

[Prerequisites](#_rqqft9n8je3a)

[Console producer](#_hqtbse5uihgt)

[Producer performance test tool (size:10, throughput:1, records:1000)](#_m7atgajtbmnz)

[Console consumer (with a consumer group)](#_ukb87px3mdgg)

[JMX](#_h608ejclyxga)

[Enable JMX port(9999) on the Kafka brokers](#_tng6mnjcuz85)

[Print Kakfa metrics via JMX using kafka.tools.JmxTool](#_m4qeh82di0st)

[jmxterm](#_qpl8q130ze80)

[jconsole (or jmxterm) from localhost machine to a Kafka broker](#_d4j3gw1spsk8)

[jmxterm from a new pod to a Kafka broker](#_q2t6sjurc64p)

[JMX Exporter and Kafka Exporter](#_8qcgu5luf9ma)

[Show JMX exporter metrics using curl](#_kf8pcfsaq56w)

[Show Kafka exporter metrics using curl](#_rlldszfvgja4)

[Log](#_ifjk346epf9n)

[Log of a pod](#_6dhe7np0vbzy)

[Multiple pod logs using stern](#_rsj0pfpcz3c3)

[Enable debug log : Kafka, Zookeeper, Entity Operator, Cruise Control and its tls-sidecar](#_h4v6bt7kpttl)

[Enable debug log : Cluster Operator](#_s7gjgw5gz0s9)

[Enable debug log : Kafka Bridge](#_vt1y2de8u9hx)

[Enable debug log : Kafka Connect](#_jl1ryjk8b59w)

[Enable debug log : Kafka Mirror Maker v1](#_dj16c683xmj7)

[Enable debug log : Kafka Mirror Maker v2](#_m519srnde071)

[Kafka Client Example](#_5buyequnvjfp)

[Run Kafka producer client example on the OpenShift](#_4iozrdb516f)

[Run Kafka streams client example on the OpenShift](#_c5iw5is61ahy)

[Debug](#_12i7oo31f510)

[JavaVM Info](#_tom8xympsu8h)

[Thread Dump](#_21o32nevwpg4)

[Packet capture using tcpdump](#_b8rg1bm82bpi)

[Heap Histogram & Stastics](#_jgvyyc9a69ld)

[Heap Dump](#_1vg1r3tqou6k)

[RHEL8 support tool image](#_q7d7rt2yl08i)

[Change AMQ Streams Kafka image: build a image on the local machine](#_z0kbck9kn5yw)

[Change AMQ Streams Kafka image: build a image on the OpenShift](#_cdxq1aiq81t8)

[Remote Debugging: Kafka Broker](#_vsm4r4tj5bo9)

[Remote Debugging: Cluster Operator](#_qp2yk6l6gr5x)

[Remote Debugging: Topic Operator in Entity Operator](#_615jyfloit7)

[Remote Debugging: User Operator in Entity Operator](#_bqjelxctzhfr)

[Remote Debugging: Kafka Connect](#_24zhjhnlqy0)

[Remote Debugging: Cruise Control](#_l4b384fw2rxj)

[Tips](#_3bakgia8mkdk)

[Dump OpenShift project](#_79utwm4jomh7)

[OpenShift events (sort by last timestamp)](#_xldfia30h1v1)

[Force a pod termination once](#_9t8hdfmvxyps)

[Continues to force a pod termination](#_9fbu05fe7lnr)

[Login to Code Ready Container host node using ssh](#_m5v62hcc26nk)

[no matches for kind "Deployment" in version "extensions/v1beta1" error occurs](#_vttiby27u9yd)

[Long and short names for each Strimzi resource](#_1ilwox2vyw59)

[Dump AMQ Streams project](#_m5qy1vil7xkn)

[Dump resources, status, and logs of AMQ Streams on OpenShift](#_5qqj158dwfa)

[Customizable report script](#_p6y62s559vw)

[References](#_jehaej3ehgvb)

[Unorganized](#_vax3qqax172c)

[Login to the pod’s node using debug container](#_ulb3svs2toeu)

[Execute GDB from a node](#_3pxthzdok0ww)

[Kill a connection using GDB](#_qssdbpamibc4)

[Show actual network policy](#_jt16ptyzzv8h)

##### 

### Introduction

This is a command cheat sheet for AMQ Streams to help support engineers with their work.

Basically, most of the commands were tested with AMQ Streams 1.5, but they may not work with different versions of AMQ Streams. If you find a command that does not work, please leave comments with your environment info. it may be fixed later.

You are free to add any useful commands to this cheat sheet. Or, if you leave comments about a command you want, it may be added later.

###### Adjust environment variables

| # Kafka cluster name KAFKA\_CLUSTER=my-cluster # Kafka topic name OPIC\_NAME=my-topic # consumer group name CONSUMER\_GROUP\_NAME=my-group # Kafka user name KAFKAUSER\_NAME=my-user # Kafka connect cluster  KAFKACONNECT\_CLUSTER=my-connect-cluster # Kafka broker ID BROKER\_ID=0 # Zookeeper ID ZOOKEEPER\_ID=0 # namespace name (i.e. project name) NAMESPACE=my-project |
| --- |

Adding these adjusted environment variables makes it easier to run the following commands.

### Kafka Cluster and Kafka Broker

###### Kafka cluster definition

| oc get Kafka -o yaml oc get statefulsets ${KAFKA\_CLUSTER}-kafka -o yaml  # internal Kafka configuration oc exec ${KAFKA\_CLUSTER}-kafka-0 -- cat /tmp/strimzi.properties oc exec ${KAFKA\_CLUSTER}-kafka-0 -- cat /opt/kafka/custom-config/log4j.properties oc exec ${KAFKA\_CLUSTER}-kafka-0 -- ps aux |
| --- |

###### Broker configurations

| oc exec -it ${KAFKA\_CLUSTER}-kafka-0 -c kafka -- env - bin/kafka-configs.sh --bootstrap-server localhost:9092 --entity-type brokers --entity-name ${BROKER\_ID} --describe --all |
| --- |

###### Show cluster-wide finalized features and features supported by a specific broker

| # cluster-wide finalized features  oc exec -it ${KAFKA\_CLUSTER}-kafka-0 -c kafka -- env - bin/kafka-features.sh --bootstrap-server localhost:9092 --describe  # features supported by a specific broker  oc exec -it ${KAFKA\_CLUSTER}-kafka-0 -c kafka -- env - bin/kafka-features.sh --bootstrap-server localhost:9092 --describe -- |
| --- |

For details, refer to [KIP-584](https://cwiki.apache.org/confluence/display/KAFKA/KIP-584%3A+Versioning+scheme+for+features).

###### Show the controller Broker ID

| # since AMQ Stream 1.6 oc exec -it ${KAFKA\_CLUSTER}-zookeeper-0 -- env - bin/zookeeper-shell.sh 127.0.0.1:12181 get /controller  # until AMQ Stream 1.5 oc exec -it ${KAFKA\_CLUSTER}-kafka-0 -c kafka -- env - bin/zookeeper-shell.sh 127.0.0.1:2181 get /controller |
| --- |

###### Change the controller broker to others

| # since AMQ Stream 1.6 oc exec -it ${KAFKA\_CLUSTER}-zookeeper-0 -- env - bin/zookeeper-shell.sh 127.0.0.1:12181 delete /controller  # until AMQ Stream 1.5 oc exec -it ${KAFKA\_CLUSTER}-kafka-0 -c kafka -- env - bin/zookeeper-shell.sh 127.0.0.1:2181 delete /controller |
| --- |

※cannot know which broker will be the next controller broker

###### Living Kafka broker list using Zookeeper

| # since AMQ Stream 1.6 oc exec -it ${KAFKA\_CLUSTER}-zookeeper-0 -- env - bin/zookeeper-shell.sh 127.0.0.1:12181 ls /brokers/ids  # until AMQ Stream 1.5 oc exec -it ${KAFKA\_CLUSTER}-kafka-0 -c kafka -- env - bin/zookeeper-shell.sh 127.0.0.1:2181 ls /brokers/ids |
| --- |

###### Kafka broker information on Zookeeper using Zookeeper

| # since AMQ Stream 1.6 oc exec -it ${KAFKA\_CLUSTER}-zookeeper-0 -- env - bin/zookeeper-shell.sh 127.0.0.1:12181 get /brokers/ids/${BROKER\_ID}  # until AMQ Stream 1.5 oc exec -it ${KAFKA\_CLUSTER}-kafka-0 -c kafka -- env - bin/zookeeper-shell.sh 127.0.0.1:2181 get /brokers/ids/${BROKER\_ID} |
| --- |

###### Delete and recreate a Kafka pod and its PVC

| oc annotate pod ${KAFKA\_CLUSTER}-kafka-${BROKER\_ID} strimzi.io/delete-pod-and-pvc=true |
| --- |

###### Manual rolling update: Kafka brokers

| oc annotate statefulset ${KAFKA\_CLUSTER}-kafka strimzi.io/manual-rolling-update=true  # FYI as another way to restart, using patch adding date label to change CR oc patch statefulset ${KAFKA\_CLUSTER}-kafka -p "{\"spec\":{\"template\":{\"metadata\":{\"labels\":{\"date\":\"`date +'%s'`\"}}}}}" |
| --- |

Wait for the next reconciliation to occur (every two minutes by default).

### Zookeeper

###### Zookeeper definition

| oc get Kafka -o yaml  oc get statefulsets ${KAFKA\_CLUSTER}-zookeeper -o yaml  # internal Zookeeper configuration oc exec ${KAFKA\_CLUSTER}-zookeeper-0 -- cat /tmp/zookeeper.properties oc exec ${KAFKA\_CLUSTER}-zookeeper-0 -- cat /opt/kafka/custom-config/log4j.properties oc exec ${KAFKA\_CLUSTER}-zookeeper-0 -- ps aux |
| --- |

###### Zookeeper interactive command line

| # since AMQ Stream 1.6 oc exec -it ${KAFKA\_CLUSTER}-zookeeper-0 -- env - bin/zookeeper-shell.sh 127.0.0.1:12181  # until AMQ Stream 1.5 oc exec -it ${KAFKA\_CLUSTER}-kafka-0 -c kafka -- env - bin/zookeeper-shell.sh 127.0.0.1:2181 |
| --- |

###### Status and environment of Zookeeper

| # since AMQ Stream 1.6 oc exec -it ${KAFKA\_CLUSTER}-zookeeper-0 -- env - bash -c "echo stat | nc localhost 12181" oc exec -it ${KAFKA\_CLUSTER}-zookeeper-0 -- env - bash -c "echo envi | nc localhost 12181"  # until AMQ Stream 1.5 : cannot know which Zookeeper [0 - X] to connect to oc exec -it ${KAFKA\_CLUSTER}-kafka-0 -c kafka -- bash -c "echo stat | nc localhost 2181" oc exec -it ${KAFKA\_CLUSTER}-kafka-0 -c kafka -- bash -c "echo envi | nc localhost 2181" |
| --- |

###### Dump sessions of zookeeper

| # since AMQ Stream 1.6 oc exec -it ${KAFKA\_CLUSTER}-zookeeper-0 -- env - bash -c "echo dump | nc localhost 12181"  # until AMQ Stream 1.5 : cannot know which Zookeeper [0 - X] to connect to  oc exec ${KAFKA\_CLUSTER}-kafka-0 -- bash -c "echo dump | nc localhost 2181" |
| --- |

You can confirm the controller broker and living Kafka brokers

###### Manual rolling update: Zookeeper

| oc annotate statefulset ${KAFKA\_CLUSTER}-zookeeper strimzi.io/manual-rolling-update=true  # FYI as another way to restart Zookeepers, using patch adding date label to change CR oc patch statefulset ${KAFKA\_CLUSTER}-zookeeper -p "{\"spec\":{\"template\":{\"metadata\":{\"labels\":{\"date\":\"`date +'%s'`\"}}}}}" |
| --- |

Wait for the next reconciliation to occur (every two minutes by default).

###### Access Zookeepers directly for debugging : since AMQ Streams 1.6

Since AMQ Streams 1.6, you don’t need to create zoo-entrance to access Zookeepers directly. Zookeepers expose 12181 port as plain in the pods. So you can use port-forward.

| # Step 1: create forward port to 12181 on the zookeeper oc port-forward ${KAFKA\_CLUSTER}-zookeeper-${ZOOKEEPER\_ID} 12181:12181  # Step 2: you can execute zookeeper-shell.sh from local machine via forward port ./bin/zookeeper-shell.sh 127.0.0.1:12181 get /controller |
| --- |

###### Create a zoo-entrance to access Zookeepers directly for debugging : until AMQ Streams 1.5

When Strimzi deploys the Kafka cluster, it keeps the ZooKeeper secure and inaccessible from any other application. This deploys a Stunnel proxy which exposes the Zookeeper without any authentication and encryption. For details, refer to <https://github.com/scholzj/zoo-entrance> .

| # get zoo-entrance definition from the zoo-entrance project wget -O ./zoo-entrance.yaml https://raw.githubusercontent.com/scholzj/zoo-entrance/master/deploy.yaml # change Kafka cluster name sed -i "s|my-cluster|${KAFKA\_CLUSTER}|g" zoo-entrance.yaml  # create the deployment with the proxy named zoo-entrance, a service named zoo-entrance and a Network Policy to allow the proxy to connect to the broker. oc apply -f ./zoo-entrance.yaml  # test zoo-entrance  # access from a broker oc exec -it ${KAFKA\_CLUSTER}-kafka-0 -c kafka -- bin/zookeeper-shell.sh zoo-entrance:2181 ls /brokers/ids # access from a new pod oc run kafka-exe -ti --image=registry.redhat.io/amq7/amq-streams-kafka-25-rhel7:1.5.0 --rm=true --restart=Never -- bin/zookeeper-shell.sh zoo-entrance:2181 ls /brokers/ids  # [if you need] define NodePort to access Zookeeper from external oc expose deployment/zoo-entrance --type=NodePort --name=zoo-entrance-external --port 2181  # access from external  ./bin/zookeeper-shell.sh $(crc ip):$(oc get service zoo-entrance-external -o=jsonpath='{.spec.ports[0].nodePort}') |
| --- |

###### Show internal zookeeper snapshot and zookeeper log

| # show zookeeper snapshot oc exec -it ${KAFKA\_CLUSTER}-zookeeper-${ZOOKEEPER\_ID} -- env - java -cp './libs/\*' org.apache.zookeeper.server.SnapshotFormatter /var/lib/zookeeper/data/version-2/snapshot.<snapshot number>  # show zookeeper log oc exec -it ${KAFKA\_CLUSTER}-zookeeper-${ZOOKEEPER\_ID} -- env - java -cp './libs/\*' org.apache.zookeeper.server.LogFormatter /var/lib/zookeeper/data/version-2/log.<log number> |
| --- |

###### 

### Kafka Topic

###### Kafka topic definitions

| oc get KafkaTopic -o yaml |
| --- |

If you use a name that cannot be used for the resource name like underbar of “test\_topic02” Kafka topic, the hash value will be added as a postfix as below:

| oc get KafkaTopic  ~~~ NAME PARTITIONS REPLICATION FACTOR testtopic01 5 3 test-topic02---d04f2b34bd744b57f59e223c84cfc6aedf8b5391 5 3 ~~~ |
| --- |

###### Actual Kafka topic list

| # using a broker oc exec -it ${KAFKA\_CLUSTER}-kafka-0 -c kafka -- env - bin/kafka-topics.sh --bootstrap-server localhost:9092 --list ~~~ my-topic-1 my-topic-2 ~~~  # using a new pod oc run kafka-exe -ti --image=registry.redhat.io/amq7/amq-streams-kafka-25-rhel7:1.5.0 --rm=true --restart=Never -- bin/kafka-topics.sh --bootstrap-server ${KAFKA\_CLUSTER}-kafka-bootstrap:9092 --list |
| --- |

###### Actual Kafka topic description

| oc exec -it ${KAFKA\_CLUSTER}-kafka-0 -c kafka -- env - bin/kafka-topics.sh --bootstrap-server localhost:9092 --describe ${TOPIC\_NAME} ~~~ Topic: my-topic-1 PartitionCount: 3 ReplicationFactor: 3 Configs: segment.bytes=1073741824,retention.ms=7200000,message.format.version=2.5-IV0  Topic: my-topic-1 Partition: 0 Leader: 2 Replicas: 2,1,0 Isr: 0,1,2  Topic: my-topic-1 Partition: 1 Leader: 1 Replicas: 1,0,2 Isr: 0,1,2  Topic: my-topic-1 Partition: 2 Leader: 0 Replicas: 0,2,1 Isr: 0,1,2 ~~~ |
| --- |

The ID of the first broker in “Replicas” is the preferred replica. If this is different from the "Leader", then the cluster is out of balance, [kafka-preferred-replica-election.sh](#_96pv85775og6) may need to be used.

“Isr” means in-sync-reclicas. If “Isr” is different from “Replicas”, it indicates that replication is delayed for some reason.

###### Kafka topic configurations

| oc exec -it ${KAFKA\_CLUSTER}-kafka-0 -c kafka -- env - bin/kafka-configs.sh --bootstrap-server localhost:9092 --entity-type topics --entity-name ${TOPIC\_NAME} --describe --all |
| --- |

###### Create a Kafka topic (replication factor: 3, partitions: 5)

| # using kafka-topics.sh command oc exec -it ${KAFKA\_CLUSTER}-kafka-0 -c kafka -- env - bin/kafka-topics.sh --bootstrap-server localhost:9092 --create --replication-factor 3 --partitions 5 --topic ${TOPIC\_NAME}  # using CR oc apply -f examples/topic/kafka-topic.yaml |
| --- |

The topic creation is done asynchronously.

###### Delete a Kafka topic

| # using Kafka command oc exec -it ${KAFKA\_CLUSTER}-kafka-0 -c kafka -- env - bin/kafka-topics.sh --bootstrap-server localhost:9092 --delete --topic ${TOPIC\_NAME}  # using CR oc delete KafkaTopic ${TOPIC\_NAME} |
| --- |

The [topic deletion](#_4hys8kp9xwzb) must be enabled.

The topic deletion is done asynchronously. To verify that the topic has been deleted, please refer to [How to confirm that the topic has been deleted](https://access.redhat.com/solutions/5376031).

###### Topic latest offset (for each partition)

| oc exec -it ${KAFKA\_CLUSTER}-kafka-0 -c kafka -- env - bin/kafka-run-class.sh kafka.tools.GetOffsetShell --broker-list localhost:9092 --topic ${TOPIC\_NAME} |
| --- |

###### Topic earliest offset (for each partition)

| oc exec -it ${KAFKA\_CLUSTER}-kafka-0 -c kafka -- env - bin/kafka-run-class.sh kafka.tools.GetOffsetShell --broker-list localhost:9092 --topic ${TOPIC\_NAME} --time -2 |
| --- |

###### Topic size (based on offsets of total partitions)

| oc exec -it ${KAFKA\_CLUSTER}-kafka-0 -c kafka -- env - bin/kafka-run-class.sh kafka.tools.GetOffsetShell --broker-list localhost:9092 --topic ${TOPIC\_NAME} --time -1 --offsets 1 | awk -F ":" '{sum += $3} END {print sum}' |
| --- |

###### Creation timestamp of a Kafka topic

| # using zookeeper's ctime  # until AMQ Stream 1.5 oc exec -it ${KAFKA\_CLUSTER}-kafka-0 -c kafka -- env - bin/zookeeper-shell.sh 127.0.0.1:2181 stat /brokers/topics/${TOPIC\_NAME}  # using creationTimestamp of Kafka topic CR oc get KafkaTopic/${TOPIC\_NAME} -o yaml -o=jsonpath='{.metadata.creationTimestamp}' |
| --- |

###### Disable topic deletion

| kind: Kafka  ... spec:  kafka:  config:  ...  delete.topic.enable: false |
| --- |

With this setting, Kafka will no longer accept topic deletion instructions. Sometimes, we guide this procedure for additional safety. For example, if you suspect that the Topic Operator deletes topics during a reconciliation or upgrade.

### Kafka Internal Topic

###### Show \_\_consumer\_offsets contents

| oc run kafka-consumer-offsets -ti --image=registry.redhat.io/amq7/amq-streams-kafka-25-rhel7:1.5.0 --rm=true --restart=Never -- bin/kafka-console-consumer.sh --formatter "kafka.coordinator.group.GroupMetadataManager\$OffsetsMessageFormatter" --bootstrap-server ${KAFKA\_CLUSTER}-kafka-bootstrap:9092 --from-beginning --topic \_\_consumer\_offsets |
| --- |

###### Show \_\_transaction\_state contents

| oc run kafka-transaction-state -ti --image=registry.redhat.io/amq7/amq-streams-kafka-25-rhel7:1.5.0 --rm=true --restart=Never -- bin/kafka-console-consumer.sh --formatter "kafka.coordinator.transaction.TransactionLog\$TransactionLogMessageFormatter" --bootstrap-server ${KAFKA\_CLUSTER}-kafka-bootstrap:9092 --from-beginning --topic \_\_transaction\_state |
| --- |

Producer's transaction id(transactional.id) is used to determine \_\_transaction\_state partition.

### Strimzi Internal Topic (since AMQ Streams 1.7)

###### Show \_\_strimzi\_store\_topic

| oc run strimzi-store-topic -ti --image=registry.redhat.io/amq7/amq-streams-kafka-27-rhel7:1.7.0 --rm=true --restart=Never -- bin/kafka-console-consumer.sh --bootstrap-server ${KAFKA\_CLUSTER}-kafka-bootstrap:9092 --from-beginning --topic \_\_strimzi\_store\_topic |
| --- |

###### Show \_\_strimzi-topic-operator-kstreams-topic-store-changelog

| oc run strimzi-topic-operator-kstreams-topic-store-changelog -ti --image=registry.redhat.io/amq7/amq-streams-kafka-27-rhel7:1.7.0 --rm=true --restart=Never -- bin/kafka-console-consumer.sh --bootstrap-server ${KAFKA\_CLUSTER}-kafka-bootstrap:9092 --from-beginning --topic \_\_strimzi-topic-operator-kstreams-topic-store-changelog |
| --- |

“\_\_strimzi-topic-operator-kstreams-topic-store-changelog” is used as key value store in the internal Kafka Streams in the Topic Operator. For details, refer to [TopicStoreTopologyProvider](https://github.com/strimzi/strimzi-kafka-operator/blob/main/topic-operator/src/main/java/io/strimzi/operator/topic/TopicStoreTopologyProvider.java) class.

### Kafka Partition and Replica

###### Log directories and partitions

| # using kafka-log-dirs.sh oc exec -it ${KAFKA\_CLUSTER}-kafka-0 -c kafka -- env - bin/kafka-log-dirs.sh --bootstrap-server localhost:9092 --describe ~~~ Received log directory information from brokers 0,1,2 {"version":1,"brokers":[{"broker":0,"logDirs":[{"logDir":"/var/lib/kafka/data-0/kafka-log0","error":null,"partitions":[{"partition":"my-topic-1-1","size":0,"offsetLag":0,"isFuture":false},{"partition":"my-topic-2-2","size":0,"offsetLag":0,"isFuture":false},{"partition":"my-topic-1-0","size":0,"offsetLag":0,"isFuture":false},{"partition":"my-topic-2-1","size":0,"offsetLag":0,"isFuture":false},{"partition":"my-topic-1-2","size":0,"offsetLag":0,"isFuture":false},{"partition":"my-topic-2-0","size":0,"offsetLag":0,"isFuture":false}]}]},{"broker":1,"logDirs":[{"logDir":"/var/lib/kafka/data-0/kafka-log1","error":null,"partitions":[{"partition":"my-topic-1-1","size":0,"offsetLag":0,"isFuture":false},{"partition":"my-topic-2-2","size":0,"offsetLag":0,"isFuture":false},{"partition":"my-topic-1-0","size":0,"offsetLag":0,"isFuture":false},{"partition":"my-topic-2-1","size":0,"offsetLag":0,"isFuture":false},{"partition":"my-topic-1-2","size":0,"offsetLag":0,"isFuture":false},{"partition":"my-topic-2-0","size":0,"offsetLag":0,"isFuture":false}]}]},{"broker":2,"logDirs":[{"logDir":"/var/lib/kafka/data-0/kafka-log2","error":null,"partitions":[{"partition":"my-topic-1-1","size":0,"offsetLag":0,"isFuture":false},{"partition":"my-topic-2-2","size":0,"offsetLag":0,"isFuture":false},{"partition":"my-topic-1-0","size":0,"offsetLag":0,"isFuture":false},{"partition":"my-topic-2-1","size":0,"offsetLag":0,"isFuture":false},{"partition":"my-topic-1-2","size":0,"offsetLag":0,"isFuture":false},{"partition":"my-topic-2-0","size":0,"offsetLag":0,"isFuture":false}]}]}]} ~~~  # FYI, actual log directories in a broker # JBOD : "log.dirs" option in the kafka configuration oc exec -it ${KAFKA\_CLUSTER}-kafka-${BROKER\_ID} -c kafka -- find /var/lib/kafka/data-${JBOD\_ID}/ # not JBOD: "log.dir" option in the kafka configuration oc exec -it ${KAFKA\_CLUSTER}-kafka-${BROKER\_ID} -c kafka -- find /var/lib/kafka/data/ |
| --- |

###### Reassign partitions : since AMQ Streams 1.6

| # step 0:create forward port  oc port-forward ${KAFKA\_CLUSTER}-zookeeper-0 12181:12181  # step 1: choose the partitions that should be reassigned cat << "EOF" > topicPartitionList.json {"topics": [{"topic": "my-topic"}],  "version":1 } EOF  # step 2: export proposed partitions and current partitions ./bin/kafka-reassign-partitions.sh --zookeeper=localhost:12181 --generate --topics-to-move-json-file topicPartitionList.json --broker-list 0,1,2,3 | tee >(tail -1 > proposed\_partition.json) | tee >(sed -n 2p > current\_partition\_`date "+%Y%m%d\_%H%M%S"`.json)  # step 3: modify proposed\_partition.json if you need  # step 4: reassign the partitions ./bin/kafka-reassign-partitions.sh --zookeeper=localhost:12181 --execute --reassignment-json-file proposed\_partition.json --throttle 5000000  # step 5: verify the partitions # using --verify command ./bin/kafka-reassign-partitions.sh --zookeeper=localhost:12181 --verify --reassignment-json-file proposed\_partition.json # using /admin/reassign\_partitions znode oc exec -it ${KAFKA\_CLUSTER}-kafka-0 -c kafka -- bin/zookeeper-shell.sh 127.0.0.1:2181 get /admin/reassign\_partitions |
| --- |

###### Reassign partitions : until AMQ Streams 1.5

| # step 1: choose the partitions that should be reassigned cat << "EOF" > topicPartitionList.json {"topics": [{"topic": "my-topic"}],  "version":1 } EOF # step 2: export proposed partitions and current partitions ./bin/kafka-reassign-partitions.sh --zookeeper $(crc ip):$(oc get service zoo-entrance-external -o=jsonpath='{.spec.ports[0].nodePort}') --generate --topics-to-move-json-file topicPartitionList.json --broker-list 0,1,2,3 | tee >(tail -1 > proposed\_partition.json) | tee >(sed -n 2p > current\_partition\_`date "+%Y%m%d\_%H%M%S"`.json)  # step 3: modify proposed\_partition.json if you need  # step 4: reassign the partitions ./bin/kafka-reassign-partitions.sh --zookeeper $(crc ip):$(oc get service zoo-entrance-external -o=jsonpath='{.spec.ports[0].nodePort}') --execute --reassignment-json-file proposed\_partition.json --throttle 5000000  # step 5: verify the partitions # using --verify command ./bin/kafka-reassign-partitions.sh --zookeeper $(crc ip):$(oc get service zoo-entrance-external -o=jsonpath='{.spec.ports[0].nodePort}') --verify --reassignment-json-file proposed\_partition.json # using /admin/reassign\_partitions znode oc exec -it ${KAFKA\_CLUSTER}-kafka-0 -c kafka -- bin/zookeeper-shell.sh 127.0.0.1:2181 get /admin/reassign\_partitions |
| --- |

For details, refer to the [document](https://access.redhat.com/documentation/en-us/red_hat_amq/7.7/html/using_amq_streams_on_openshift/assembly-deployment-configuration-str#con-partition-reassignment-deployment-configuration-kafka). and this example is using [zoo-entrance](#_nf3urumt7tk3) to access Zookeeper.

###### Stop Reassigning partitions

| # since AMQ Stream 1.6 oc exec -it ${KAFKA\_CLUSTER}-zookeeper-0 -- env - bin/zookeeper-shell.sh 127.0.0.1:12181 delete /admin/reassign\_partitions  # until AMQ Stream 1.5 oc exec -it ${KAFKA\_CLUSTER}-kafka-0 -c kafka -- env - bin/zookeeper-shell.sh 127.0.0.1:2181 delete /admin/reassign\_partitions |
| --- |

In some circumstances, reassignment may be halted and re-execution may not be possible. In such cases, you can stop reassignment with this command. However, general use is not recommended.

###### Topics that have under-replicated-partitions

| oc exec -it ${KAFKA\_CLUSTER}-kafka-0 -c kafka -- env - bin/kafka-topics.sh --bootstrap-server localhost:9092 --describe --under-replicated-partitions |
| --- |

###### Topics that have unavailable-partitions

| oc exec -it ${KAFKA\_CLUSTER}-kafka-0 -c kafka -- env - bin/kafka-topics.sh --bootstrap-server localhost:9092 --describe --unavailable-partitions |
| --- |

###### Preferred replica election

| oc exec -it ${KAFKA\_CLUSTER}-kafka-0 -c kafka -- env - bin/kafka-preferred-replica-election.sh --bootstrap-server localhost:9092 |
| --- |

No major side effects, so if the cluster performance is not stable, run this command anyway.

### Kafka Producer

###### Console producer

| # using a new pod oc run kafka-producer -ti --image=registry.redhat.io/amq7/amq-streams-kafka-25-rhel7:1.5.0 --rm=true --restart=Never -- bin/kafka-console-producer.sh --broker-list ${KAFKA\_CLUSTER}-kafka-bootstrap:9092 --topic ${TOPIC\_NAME}   # using kafka-0 broker oc exec -it ${KAFKA\_CLUSTER}-kafka-0 -- env - bin/kafka-console-producer.sh --broker-list localhost:9092 --topic ${TOPIC\_NAME} |
| --- |

###### Producer performance test tool (size:10, throughput:1, records:1000)

| oc run kafka-producer-perf -ti --image=registry.redhat.io/amq7/amq-streams-kafka-25-rhel7:1.5.0 --rm=true --restart=Never -- bin/kafka-producer-perf-test.sh --producer-props bootstrap.servers=${KAFKA\_CLUSTER}-kafka-bootstrap:9092 --record-size=10 --throughput 1 --num-records 1000 --topic ${TOPIC\_NAME} |
| --- |

### Kafka Consumer and Consumer Group

###### Console consumer (without a consumer group)

| # using a new pod  oc run kafka-consumer -ti --image=registry.redhat.io/amq7/amq-streams-kafka-25-rhel7:1.5.0 --rm=true --restart=Never -- bin/kafka-console-consumer.sh --bootstrap-server ${KAFKA\_CLUSTER}-kafka-bootstrap:9092 --from-beginning --topic ${TOPIC\_NAME}  # using kafka-0 broker oc exec -it ${KAFKA\_CLUSTER}-kafka-0 -- env - bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --from-beginning --topic ${TOPIC\_NAME} |
| --- |

Use the “--whitelist” option to subscribe to multiple topics using regex instead of --topic option,

for example --whitelist topicprefix.\*'

###### Console consumer (with a consumer group)

| # using a new pod  oc run kafka-consumer -ti --image=registry.redhat.io/amq7/amq-streams-kafka-25-rhel7:1.5.0 --rm=true --restart=Never -- bin/kafka-console-consumer.sh --bootstrap-server ${KAFKA\_CLUSTER}-kafka-bootstrap:9092 --from-beginning --topic ${TOPIC\_NAME} --group ${CONSUMER\_GROUP\_NAME}  # using kafka-0 broker  oc exec -it ${KAFKA\_CLUSTER}-kafka-0 -- bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --from-beginning --topic ${TOPIC\_NAME} --group ${CONSUMER\_GROUP\_NAME} |
| --- |

###### Consumer performance test tool

| oc run kafka-consumer-perf -ti --image=registry.redhat.io/amq7/amq-streams-kafka-25-rhel7:1.5.0 --rm=true --restart=Never -- bin/kafka-consumer-perf-test.sh --broker-list ${KAFKA\_CLUSTER}-kafka-bootstrap:9092 --threads 1 --from-latest --print-metrics --show-detailed-stats=true --reporting-interval 1000 --timeout 100000000 --messages 10000 --topic ${TOPIC\_NAME} |
| --- |

###### Consumer group list

| oc exec -it ${KAFKA\_CLUSTER}-kafka-0 -c kafka -- env - bin/kafka-consumer-groups.sh --bootstrap-server localhost:9092 --list |
| --- |

###### Consumer group description (--offset)

| # a specific consumer group oc exec -it ${KAFKA\_CLUSTER}-kafka-0 -c kafka -- env - bin/kafka-consumer-groups.sh --bootstrap-server localhost:9092 --describe --group ${CONSUMER\_GROUP\_NAME} ~~~ GROUP TOPIC PARTITION CURRENT-OFFSET LOG-END-OFFSET LAG CONSUMER-ID HOST CLIENT-ID my-group test\_topic 0 30 30 0 consumer-1-0f11a1f8-8f0c-4cd3-9499-471b62332d7f /10.128.0.202 consumer-1 my-group test\_topic 4 30 30 0 consumer-1-0f11a1f8-8f0c-4cd3-9499-471b62332d7f /10.128.0.202 consumer-1 ~~~  # all the consumer groups oc exec -it ${KAFKA\_CLUSTER}-kafka-0 -c kafka -- env - bin/kafka-consumer-groups.sh --bootstrap-server localhost:9092 --describe --all-groups |
| --- |

###### Consumer group description (--state)

| oc exec -it ${KAFKA\_CLUSTER}-kafka-0 -c kafka -- env - bin/kafka-consumer-groups.sh --bootstrap-server localhost:9092 --describe --group ${CONSUMER\_GROUP\_NAME} --state ~~~ GROUP COORDINATOR (ID) ASSIGNMENT-STRATEGY STATE #MEMBERS my-group my-cluster-kafka-2.my-cluster-kafka-brokers.my-project.svc:9092 (2) range Stable 1 ~~~ |
| --- |

###### Delete a consumer group

| oc exec -it ${KAFKA\_CLUSTER}-kafka-0 -c kafka -- env - bin/kafka-consumer-groups.sh --bootstrap-server localhost:9092 --delete --group ${CONSUMER\_GROUP\_NAME} |
| --- |

###### Reset the offsets for a specific consumer group

| # rest to earliest oc exec -it ${KAFKA\_CLUSTER}-kafka-0 -c kafka -- env - bin/kafka-consumer-groups.sh --bootstrap-server localhost:9092 --reset-offsets --to-earliest --group ${CONSUMER\_GROUP\_NAME} --topic ${TOPIC\_NAME} --execute  # rest to specific offset oc exec -it ${KAFKA\_CLUSTER}-kafka-0 -c kafka -- env - bin/kafka-consumer-groups.sh --bootstrap-server localhost:9092 --reset-offsets --to-offset <OFFSET\_VALUE> --group ${CONSUMER\_GROUP\_NAME} --topic ${TOPIC\_NAME}:<PARTITION\_NUMBERS> --execute |
| --- |

Don’t forget --execute option. For the other options on how to reset other than earliest, see [here](https://cwiki.apache.org/confluence/display/KAFKA/KIP-122%3A+Add+Reset+Consumer+Group+Offsets+tooling#KIP122:AddResetConsumerGroupOffsetstooling-Scenarios).

###### Delete offset with specific topic for a specific consumer group

| oc exec -it ${KAFKA\_CLUSTER}-kafka-0 -c kafka -- env - bin/kafka-consumer-groups.sh --bootstrap-server localhost:9092 --delete-offsets --group ${CONSUMER\_GROUP\_NAME} --topic ${TOPIC\_NAME} |
| --- |

### About Kafka CLI execution environment

###### Executing Kafka CLI using an existing broker

| oc exec -it ${KAFKA\_CLUSTER}-kafka-0 -c kafka -- env - <your\_command> |
| --- |

This is the easiest way to run the Kafka CLI such as bin/kafka-topics.sh. For this reason, It is also used as example commands above. However, If you don't want to burden the broker, you should create a new pod to execute commands.

[**IMPORTANT**] when running the Kafka CLI on a broker, you should add "env -" to remove the environment variable because even “-Xms” specified for the Kafka broker could be applied for the CLI commands and it can lead out of memory error.

For “oc exe” using the existing kafka broker pods, you can use localhost:9092 as a Kafka broker, or localhost:2181 to Zookeeper via tls-sidecar (until AMQ Streams 1.5) as below:

| # access Kafka oc exec -it ${KAFKA\_CLUSTER}-kafka-0 -c kafka -- env - bin/kafka-topics.sh --bootstrap-server localhost:9092 --list  # access Zookeeper # since AMQ Stream 1.6 oc exec -it ${KAFKA\_CLUSTER}-zookeeper-0 -- env - bin/zookeeper-shell.sh 127.0.0.1:12181 delete /controller # via local tls-sidecar: until AMQ Streams 1.5 oc exec -it ${KAFKA\_CLUSTER}-kafka-0 -c kafka -- env - bin/zookeeper-shell.sh zoo-entrance:2181 ls /brokers/ids |
| --- |

###### Executing Kafka CLI using a new pod (do not use a pod of an existing broker)

| oc run kafka-exe -ti --image=registry.redhat.io/amq7/amq-streams-kafka-25-rhel7:1.5.0 --rm=true --restart=Never -- <your\_command> |
| --- |

This method does not allow to use localhost:9092 as a Kafka broker, or localhost:2181 to Zookeeper via tls-sidecar. You can use service to access kafka, and use [zoo-entrance](#_nf3urumt7tk3) to access Zookeeper, and so on like below:

| # access Kafka via bootstrap service oc run kafka-exe -ti --image=registry.redhat.io/amq7/amq-streams-kafka-25-rhel7:1.5.0 --rm=true --restart=Never -- bin/kafka-topics.sh --bootstrap-server ${KAFKA\_CLUSTER}-kafka-bootstrap:9092 --describe ${TOPIC\_NAME}  # FYI, access Zookeeper via zoo-entrance # until AMQ Stream 1.5 oc run kafka-exe -ti --image=registry.redhat.io/amq7/amq-streams-kafka-25-rhel7:1.5.0 --rm=true --restart=Never -- bin/zookeeper-shell.sh zoo-entrance:2181 ls /brokers/ids |
| --- |

###### Executing Kafka CLI using a customized new pod as root

| # create imagestreams of amq-streams image oc import-image amq-streams-debug --from=registry.redhat.io/amq7/amq-streams-kafka-25-rhel7:1.5.0 --confirm  # run a new pod based on amq-streams image as root and execute your commands oc debug istag/amq-streams-debug:latest --as-root=true > <your command>  # FYI, you can delete the imagestream for cleanup after debugging oc delete imagestream amq-streams-debug |
| --- |

It can be used to execute kafka CLI such as bin/kafka-topics.sh in any namespace. You can also install various packages using microdnf.

Additionally, you can customize the amq-streams Kafka image using Dockerfile as below:

| # step 1: create Dockerfile that extends amq-streams image. vi Dockerfile ~~~ FROM registry.redhat.io/amq7/amq-streams-kafka-25-rhel7:1.5.0 USER root:root RUN microdnf install -y curl java-1.8.0-openjdk-devel  USER 1001 ~~~  # step 2: Docker build on the OpenShift oc new-build --name=amq-streams-debug --strategy=docker --binary oc start-build amq-streams-debug --from-dir=. --follow  # step 3: run the image as root and execute your commands oc debug istag/amq-streams-debug:latest --as-root=true > <your command>  # FYI, you can delete the build configuration and imagestream for cleanup after debugging oc delete all -l app=amq-streams-debug -l build=amq-streams-debug |
| --- |

### Strimzi Cluster Operator

*Strimzi Cluster Operator definition*

| oc get deployment/strimzi-cluster-operator -o yaml  # internal Cluster Operator configuration oc exec deployment/strimzi-cluster-operator -- ps aux |
| --- |

###### Stop cluster operator

| # stop the cluster operator oc scale deployment/strimzi-cluster-operator --replicas=0  # restart the cluster operator oc scale deployment/strimzi-cluster-operator --replicas=1 |
| --- |

###### Pause reconciliation of custom resources

| # pause reconciliation since AMQ Streams 1.7, for topic resource since AMQ Streams 1.8  oc annotate *KIND-OF-CUSTOM-RESOURCE* *NAME-OF-CUSTOM-RESOURCE* strimzi.io/pause-reconciliation="true"  # restart reconciliation oc annotate *KIND-OF-CUSTOM-RESOURCE* *NAME-OF-CUSTOM-RESOURCE* strimzi.io/pause-reconciliation- |
| --- |

###### Check for duplicate cluster operators installed

| oc get pods --all-namespaces oc get pods --all-namespaces -l "strimzi.io/kind=cluster-operator" -o yaml  oc get packagemanifests -n openshift-marketplace oc get packagemanifests/amq-streams -n openshift-marketplace -o yaml oc get packagemanifests/strimzi-kafka-operator -n openshift-marketplace -o yaml  oc get Subscription --all-namespaces -o yaml |
| --- |

### Strimzi Entity Operator (comprises Topic Operator and User Operator)

*Strimzi Entity Operator definition*

| oc get Kafka -o yaml  oc get deployment/${KAFKA\_CLUSTER}-entity-operator -o yaml  # internal Entity Operator configuration oc exec deployment/${KAFKA\_CLUSTER}-entity-operator -c topic-operator -- ps aux  oc exec deployment/${KAFKA\_CLUSTER}-entity-operator -c user-operator -- ps aux  oc exec deployment/${KAFKA\_CLUSTER}-entity-operator -c topic-operator -- cat /opt/topic-operator/custom-config/log4j2.properties  oc exec deployment/${KAFKA\_CLUSTER}-entity-operator -c user-operator -- cat /opt/user-operator/custom-config/log4j2.properties |
| --- |

###### Stop entity operator

| # To stop entity operator, comment out "entityOperator" and apply it. ~~~ apiVersion: kafka.strimzi.io/v1beta1 kind: Kafka spec:  kafka: # entityOperator: # topicOperator: {} # userOperator: {} ~~~  # FYI, you CANNOT stop the entity operator using the following command because CO will revert it. oc scale deployment/${KAFKA\_CLUSTER}-entity-operator --replicas=0 |
| --- |

### Kafka Connect and Debezium

###### Create a MySQL DB POD for test

| # create a mysql pod oc new-app --name=mysql debezium/example-mysql:1.0 -e MYSQL\_ROOT\_PASSWORD=debezium -e MYSQL\_USER=mysqluser -e MYSQL\_PASSWORD=mysqlpw -l name=mysql-debug  # login to mysql cli oc exec deployment/mysql -it -- mysql -umysqluser -pmysqlpw inventory  # execute mysql commands you want show tables; select \* from customers; insert into customers values (default, "Sarah", "Thompson", "[kitt@acme.com](mailto:kitt@acme.com)");  # FYI, for cleanup after debugging oc delete all -l name=mysql-debug |
| --- |

###### Create Kafka Connect image with Debezium: build a image on the OpenShift

| # step 1  mkdir ./my-plugins  # step 2: download kafka connect plugins or Debezium plugins  https://access.redhat.com/jbossnetwork/restricted/listSoftware.html?product=red.hat.integration&downloadType=distributions  # step 3: extract files and put it in ./my-plugins  $ tree ./my-plugins/ ./my-plugins/ ├── debezium-connector-db2 │ ├── ... ├── debezium-connector-mongodb │ ├── ... ├── debezium-connector-mysql │ ├── ... ├── debezium-connector-postgres │ ├── ... └── debezium-connector-sqlserver │ ├── ...  # step 4: create Dockerfile that extends amq-streams image. vi Dockerfile ~~~ FROM registry.redhat.io/amq7/amq-streams-kafka-25-rhel7:1.5.0 USER root:root COPY ./my-plugins/ /opt/kafka/plugins/ USER 1001 ~~~  # step 5: Docker build on the OpenShift oc new-build --name=my-kafka-connect --strategy=docker --binary oc start-build my-kafka-connect --from-dir=. --follow  # step 6: set KafkaConnector.spec.image to the new image path and apply. vi examples/kafka-connect/kafka-connect-single-node-kafka.yaml ~~~ apiVersion: kafka.strimzi.io/v1beta1 kind: KafkaConnector metadata:  name: my-connect-cluster spec:  #...  image: image-registry.openshift-image-registry.svc:5000/<$$namespace$$>/my-kafka-connect:latest ~~~  # FYI, you can delete the build configuration and imagestream for cleanup after debugging oc delete all -l build=my-kafka-connect |
| --- |

###### Create a MySQL Debezium connector accessing to MySQL DB on the Kafka Connector POD

| oc exec deployment/${KAFKACONNECT\_CLUSTER}-connect -- curl -s -v -X POST -H "Accept:application/json" -H "Content-Type:application/json" localhost:8083/connectors/ -d '{ "name": "inventory-connector", "config": { "connector.class": "io.debezium.connector.mysql.MySqlConnector", "tasks.max": "1", "database.hostname": "mysql", "database.port": "3306", "database.user": "debezium", "database.password": "dbz", "database.server.id": "184054", "database.server.name": "dbserver1", "database.whitelist": "inventory", "database.history.kafka.bootstrap.servers": "'${KAFKA\_CLUSTER}'-kafka-bootstrap:9092", "database.history.kafka.topic": "schema-changes.inventory" } }'  # FYI, you can subscribe "dbserver1.inventory.\*" topics using the following command. oc run kafka-consumer -ti --image=registry.redhat.io/amq7/amq-streams-kafka-25-rhel7:1.5.0 --rm=true --restart=Never -- bin/kafka-console-consumer.sh --bootstrap-server ${KAFKA\_CLUSTER}-kafka-bootstrap:9092 --from-beginning --whitelist 'dbserver1.inventory.\*'  # FYI, you can insert a row to MySQL DB using the following command. oc exec deployment/mysql -it -- mysql -umysqluser -pmysqlpw inventory -e 'insert into customers values (default, "Sarah", "Thompson", "kitt@acme.com")' |
| --- |

### Cruise Control

###### Cruise Control definition

| oc get Kafka -o yaml oc get deployment/${KAFKA\_CLUSTER}-cruise-control -o yaml  # internal Kafka configuration oc exec deployment/${KAFKA\_CLUSTER}-cruise-control -- cat /tmp/cruisecontrol.properties # log4j configuration since AMQ Streams 1.7  oc exec deployment/${KAFKA\_CLUSTER}-cruise-control -- cat /opt/cruise-control/custom-config/log4j.properties  # log4j configuration until AMQ Streams 1.6  oc exec deployment/${KAFKA\_CLUSTER}-cruise-control -- cat /opt/cruise-control/custom-config/log4j.properties oc exec deployment/${KAFKA\_CLUSTER}-cruise-control -- ps aux |
| --- |

### TLS

###### Validity days of Cluster CA and Clients CA certifications in secrets

| # Cluster CA oc get secret ${KAFKA\_CLUSTER}-cluster-ca-cert -o 'jsonpath={.data.ca\.crt}' | base64 -d | openssl x509 -subject -issuer -startdate -enddate -noout ~~~ subject=O = io.strimzi, CN = cluster-ca v0 issuer=O = io.strimzi, CN = cluster-ca v0 notBefore=Jun 30 09:43:54 2020 GMT notAfter=Jun 30 09:43:54 2021 GMT ~~~  # Clients CA oc get secret ${KAFKA\_CLUSTER}-clients-ca-cert -o 'jsonpath={.data.ca\.crt}' | base64 -d | openssl x509 -subject -issuer -startdate -enddate -noout ~~~ subject=O = io.strimzi, CN = clients-ca v0 issuer=O = io.strimzi, CN = clients-ca v0 notBefore=Jun 30 09:43:54 2020 GMT notAfter=Jun 30 09:43:54 2021 GMT ~~~  # FYI:  # Kafka Broker oc get secret ${KAFKA\_CLUSTER}-kafka-brokers -o 'jsonpath={.data.'${KAFKA\_CLUSTER}'-kafka-0\.crt}' | base64 -d | openssl x509 -subject -issuer -startdate -enddate -noout  # Zookeeper oc get secret ${KAFKA\_CLUSTER}-zookeeper-nodes -o 'jsonpath={.data.'${KAFKA\_CLUSTER}'-zookeeper-0\.crt}' | base64 -d | openssl x509 -subject -issuer -startdate -enddate -noout  # Entity Operator oc get secret ${KAFKA\_CLUSTER}-entity-operator-certs -o 'jsonpath={.data.entity-operator\.crt}' | base64 -d | openssl x509 -subject -issuer -startdate -enddate -noout |
| --- |

###### Check actual TLS info using openssl s\_client command

| # check to connect to server oc exec -it ${KAFKA\_CLUSTER}-kafka-0 -c kafka -- openssl s\_client -debug -connect <TLS\_URL\_ADDRESS>:<TLS\_PORT\_NUMBER> -servername <TLS\_URL\_ADDRESS>  # show validity days oc exec -it ${KAFKA\_CLUSTER}-kafka-0 -c kafka -- openssl s\_client -debug -connect <TLS\_URL\_ADDRESS>:<TLS\_PORT\_NUMBER> -servername <TLS\_URL\_ADDRESS> 2>/dev/null | openssl x509 -subject -issuer -startdate -enddate -noout |
| --- |

###### Check TLS port on a Kafka broker using openssl s\_client command

| oc exec -it ${KAFKA\_CLUSTER}-kafka-0 -c kafka -- openssl s\_client -debug -connect localhost:<TLS\_PORT\_NUMBER> < /dev/null |
| --- |

###### Enable ssl debug log on Kafka brokers

| ~~~  Kind: Kakfa spec:  kafka:  template:  kafkaContainer:  env:  - name: KAFKA\_DEBUG  value: "true"  - name: JAVA\_DEBUG\_OPTS  value: "-Djavax.net.debug=ssl" # or "-Djavax.net.debug=all" ~~~ |
| --- |

###### Renewing CA certificates manually

| # cluster CA oc annotate secret ${KAFKA\_CLUSTER}-cluster-ca-cert strimzi.io/force-renew=true  # clients CA oc annotate secret ${KAFKA\_CLUSTER}-clients-ca-cert strimzi.io/force-renew=true |
| --- |

for details, refer to the [document](https://access.redhat.com/documentation/en-us/red_hat_amq/7.7/html-single/using_amq_streams_on_openshift/index#proc-renewing-ca-certs-manually-str).

###### Replace to my own cluster ca certification and its key

| # I recommend recreating the kafka cluster before replacing to my own cluster ca. oc delete kafka/${KAFKA\_CLUSTER}  # set Kafka.clusterCa.generateCertificateAuthority: false # vi kafka.yaml ~~~ apiVersion: kafka.strimzi.io/v1beta1 kind: Kafka ...  clusterCa:  generateCertificateAuthority: false ~~~  # replace cluster ca mkdir cert cd cert  openssl genrsa 2024 > ca.key openssl req -new -key ca.key -subj "/C=JP/ST=Tokyo/L=Shibuya-ku/O=redhatexample/OU=cee/CN=amqstreams.redhatexample.com" > ca.csr openssl x509 -req -days 365 -signkey ca.key < ca.csr > ca.crt  oc delete secret ${KAFKA\_CLUSTER}-cluster-ca-cert oc create secret generic ${KAFKA\_CLUSTER}-cluster-ca-cert --from-file=ca.crt=./ca.crt  oc delete secret ${KAFKA\_CLUSTER}-cluster-ca oc create secret generic ${KAFKA\_CLUSTER}-cluster-ca --from-file=ca.key=./ca.key  oc label secret ${KAFKA\_CLUSTER}-cluster-ca-cert strimzi.io/kind=Kafka strimzi.io/cluster=${KAFKA\_CLUSTER} oc label secret ${KAFKA\_CLUSTER}-cluster-ca strimzi.io/kind=Kafka strimzi.io/cluster=${KAFKA\_CLUSTER}  # for cleanup cd .. rm -R ./cert  # apply kafka.yaml oc apply -f kafka.yaml |
| --- |

If you want to test cluster CA certification expiration, you can set “-days 0“.

###### Enable SSL debug log of Kafka CLI for debugging

| # Kafka console producer KAFKA\_OPTS=-Djavax.net.debug=ssl bin/kafka-console-producer.sh ...  # Kafka producer performance test tool KAFKA\_OPTS=-Djavax.net.debug=ssl bin/kafka-producer-perf-test.sh ...  # Kafka console consumer KAFKA\_OPTS=-Djavax.net.debug=ssl bin/kafka-console-consumer.sh ... |
| --- |

Use KAFKA\_OPTS=-Djavax.net.debug=ssl

###### Cluster CA and truststore.p12

| # Cluster CA oc get secret ${KAFKA\_CLUSTER}-cluster-ca-cert -o jsonpath='{.data.ca\.crt}' | base64 -d > ca.crt  # truststore.p12 keytool -keystore truststore.p12 -storepass <your truststore password> -noprompt -alias ca -import -file ca.crt -storetype PKCS12 |
| --- |

###### KafkaUser secret and keystore.p12

| # KafkaUser  oc get secret ${KAFKAUSER\_NAME} -o jsonpath='{.data.user\.p12}' | base64 -d > user.p12 oc get secret ${KAFKAUSER\_NAME} -o jsonpath='{.data.user\.password}' | base64 -d > user.password  # keystore.p12 keytool -importkeystore -destkeystore keystore.p12 -srckeystore user.p12 -srcstorepass $(cat user.password) -srcstoretype PKCS12 -deststoretype PKCS12 -destkeypass <your keystore password> -deststorepass <your keystore password as same as “destkeypass”>; |
| --- |

### tls listener (port:9093) with one way TLS

###### Prerequisites

| # modify kafka.yaml and apply it ~~~ # since AMQ Streams 1.6 kind: Kafka spec:  kafka:  listeners:  ...  - name: tls  port: 9093  type: internal  tls: true  # until AMQ Streams 1.5 kind: Kafka spec:  kafka:  listeners:  tls: {}  ... ~~~ |
| --- |

###### Console producer

| oc run kafka-producer-tls -ti --image=ignore --rm=true --restart=Never --overrides='{ "apiVersion": "v1", "spec": { "volumes": [{"name": "cluster-ca","secret": {"secretName": "'${KAFKA\_CLUSTER}'-cluster-ca-cert"}}, {"name": "workdir","emptyDir": {}}], "containers": [{ "name": "kafka-producer-tls", "image":"registry.redhat.io/amq7/amq-streams-kafka-25-rhel7:1.5.0", "tty":true, "volumeMounts": [{"mountPath": "/opt/kafka/cluster-ca-certs", "name": "cluster-ca"}, {"mountPath": "/opt/kafka/workdir", "name": "workdir"}], "command":["bash","-c", "keytool -keystore workdir/truststore.p12 -storepass password -noprompt -alias ca -import -file /opt/kafka/cluster-ca-certs/ca.crt -storetype PKCS12; bin/kafka-console-producer.sh --broker-list '${KAFKA\_CLUSTER}'-kafka-bootstrap:9093 --producer-property security.protocol=SSL --producer-property ssl.truststore.location=workdir/truststore.p12 --producer-property ssl.truststore.password=password --producer-property ssl.truststore.type=PKCS12 --topic '${TOPIC\_NAME}' "] }] }}' |
| --- |

###### Producer performance test tool (size:10, throughput:1, records:1000)

| oc run kafka-producer-tls -ti --image=ignore --rm=true --restart=Never --overrides='{ "apiVersion": "v1", "spec": { "volumes": [{"name": "cluster-ca","secret": {"secretName": "'${KAFKA\_CLUSTER}'-cluster-ca-cert"}}, {"name": "workdir","emptyDir": {}}], "containers": [{ "name": "kafka-producer-tls", "image":"registry.redhat.io/amq7/amq-streams-kafka-25-rhel7:1.5.0", "tty":true, "volumeMounts": [{"mountPath": "/opt/kafka/cluster-ca-certs", "name": "cluster-ca"}, {"mountPath": "/opt/kafka/workdir", "name": "workdir"}], "command":["bash","-c", "keytool -keystore workdir/truststore.p12 -storepass password -noprompt -alias ca -import -file /opt/kafka/cluster-ca-certs/ca.crt -storetype PKCS12; bin/kafka-producer-perf-test.sh --producer-props bootstrap.servers='${KAFKA\_CLUSTER}'-kafka-bootstrap:9093 security.protocol=SSL ssl.truststore.location=workdir/truststore.p12 ssl.truststore.password=password ssl.truststore.type=PKCS12 --record-size=10 --throughput 1 --num-records 1000 --topic '${TOPIC\_NAME}' "] }] }}' |
| --- |

###### Console consumer

| oc run kafka-consumer-tls -ti --image=ignore --rm=true --restart=Never --overrides='{ "apiVersion": "v1", "spec": { "volumes": [{"name": "cluster-ca","secret": {"secretName": "'${KAFKA\_CLUSTER}'-cluster-ca-cert"}}, {"name": "workdir","emptyDir": {}}], "containers": [{ "name": "kafka-consumer-tls", "image":"registry.redhat.io/amq7/amq-streams-kafka-25-rhel7:1.5.0", "tty":true, "volumeMounts": [{"mountPath": "/opt/kafka/cluster-ca-certs", "name": "cluster-ca"}, {"mountPath": "/opt/kafka/workdir", "name": "workdir"}], "command":["bash","-c", "keytool -keystore workdir/truststore.p12 -storepass password -noprompt -alias ca -import -file /opt/kafka/cluster-ca-certs/ca.crt -storetype PKCS12; bin/kafka-console-consumer.sh --bootstrap-server '${KAFKA\_CLUSTER}'-kafka-bootstrap:9093 --consumer-property security.protocol=SSL --consumer-property ssl.truststore.location=workdir/truststore.p12 --consumer-property ssl.truststore.password=password --consumer-property ssl.truststore.type=PKCS12 --from-beginning --topic '${TOPIC\_NAME}' "] }] }}' |
| --- |

### tls listener (port:9093) with two-way(mutual) TLS client authentication and simple authorization

###### Prerequisites

| # modify kafka.yaml and apply it ~~~ # since AMQ Streams 1.6 kind: Kafka spec:  kafka:  listeners:  ...  - name: tls  port: 9093  type: internal  tls: true  authentication:  type: tls  ...  authorization:  type: simple  # until AMQ Streams 1.5 kind: Kafka spec:  kafka:  listeners:  tls:   authentication:  type: tls  authorization:  type: simple  ... ~~~  # modify examples/user/kafka-user.yaml and apply it ~~~ apiVersion: kafka.strimzi.io/v1beta1 kind: KafkaUser metadata:  name: my-user ... spec:  authentication:  type: tls ... |
| --- |

###### Console producer

| oc run kafka-producer-tls -ti --image=ignore --rm=true --restart=Never --overrides='{ "apiVersion": "v1", "spec": { "volumes": [{"name": "cluster-ca","secret": {"secretName": "'${KAFKA\_CLUSTER}'-cluster-ca-cert"}}, {"name": "kafkauser","secret": {"secretName": "'${KAFKAUSER\_NAME}'"}}, {"name": "workdir","emptyDir": {}}], "containers": [{ "name": "kafka-producer-tls", "image":"registry.redhat.io/amq7/amq-streams-kafka-25-rhel7:1.5.0", "tty":true, "volumeMounts": [{"mountPath": "/opt/kafka/cluster-ca-certs", "name": "cluster-ca"}, {"mountPath": "/opt/kafka/kafkauser", "name": "kafkauser"}, {"mountPath": "/opt/kafka/workdir", "name": "workdir"}], "command":["bash","-c", "keytool -keystore workdir/truststore.p12 -storepass password -noprompt -alias ca -import -file /opt/kafka/cluster-ca-certs/ca.crt -storetype PKCS12; keytool -importkeystore -destkeystore workdir/keystore.p12 -srckeystore /opt/kafka/kafkauser/user.p12 -srcstorepass $(cat /opt/kafka/kafkauser/user.password) -srcstoretype PKCS12 -deststoretype PKCS12 -destkeypass password -deststorepass password; bin/kafka-console-producer.sh --broker-list '${KAFKA\_CLUSTER}'-kafka-bootstrap:9093 --producer-property security.protocol=SSL --producer-property ssl.truststore.location=workdir/truststore.p12 --producer-property ssl.truststore.password=password --producer-property ssl.truststore.type=PKCS12 --producer-property ssl.keystore.location=workdir/keystore.p12 --producer-property ssl.keystore.password=password --producer-property ssl.keystore.type=PKCS12 --topic '${TOPIC\_NAME}' "] }] }}' |
| --- |

###### Producer performance test tool (size:10, throughput:1, records:1000)

| oc run kafka-producer-tls -ti --image=ignore --rm=true --restart=Never --overrides='{ "apiVersion": "v1", "spec": { "volumes": [{"name": "cluster-ca","secret": {"secretName": "'${KAFKA\_CLUSTER}'-cluster-ca-cert"}}, {"name": "kafkauser","secret": {"secretName": "'${KAFKAUSER\_NAME}'"}}, {"name": "workdir","emptyDir": {}}], "containers": [{ "name": "kafka-producer-tls", "image":"registry.redhat.io/amq7/amq-streams-kafka-25-rhel7:1.5.0", "tty":true, "volumeMounts": [{"mountPath": "/opt/kafka/cluster-ca-certs", "name": "cluster-ca"}, {"mountPath": "/opt/kafka/kafkauser", "name": "kafkauser"}, {"mountPath": "/opt/kafka/workdir", "name": "workdir"}], "command":["bash","-c", "keytool -keystore workdir/truststore.p12 -storepass password -noprompt -alias ca -import -file /opt/kafka/cluster-ca-certs/ca.crt -storetype PKCS12; keytool -importkeystore -destkeystore workdir/keystore.p12 -srckeystore /opt/kafka/kafkauser/user.p12 -srcstorepass $(cat /opt/kafka/kafkauser/user.password) -srcstoretype PKCS12 -deststoretype PKCS12 -destkeypass password -deststorepass password; bin/kafka-producer-perf-test.sh --producer-props bootstrap.servers='${KAFKA\_CLUSTER}'-kafka-bootstrap:9093 security.protocol=SSL ssl.truststore.location=workdir/truststore.p12 ssl.truststore.password=password ssl.truststore.type=PKCS12 ssl.keystore.location=workdir/keystore.p12 ssl.keystore.password=password ssl.keystore.type=PKCS12 --record-size=10 --throughput 1 --num-records 1000 --topic '${TOPIC\_NAME}' "] }] }}' |
| --- |

###### Console consumer (with a consumer group)

| oc run kafka-consumer-tls -ti --image=ignore --rm=true --restart=Never --overrides='{ "apiVersion": "v1", "spec": { "volumes": [{"name": "cluster-ca","secret": {"secretName": "'${KAFKA\_CLUSTER}'-cluster-ca-cert"}}, {"name": "kafkauser","secret": {"secretName": "'${KAFKAUSER\_NAME}'"}}, {"name": "workdir","emptyDir": {}}], "containers": [{ "name": "kafka-consumer-tls", "image":"registry.redhat.io/amq7/amq-streams-kafka-25-rhel7:1.5.0", "tty":true, "volumeMounts": [{"mountPath": "/opt/kafka/cluster-ca-certs", "name": "cluster-ca"}, {"mountPath": "/opt/kafka/kafkauser", "name": "kafkauser"}, {"mountPath": "/opt/kafka/workdir", "name": "workdir"}], "command":["bash","-c", "keytool -keystore workdir/truststore.p12 -storepass password -noprompt -alias ca -import -file /opt/kafka/cluster-ca-certs/ca.crt -storetype PKCS12; keytool -importkeystore -destkeystore workdir/keystore.p12 -srckeystore /opt/kafka/kafkauser/user.p12 -srcstorepass $(cat /opt/kafka/kafkauser/user.password) -srcstoretype PKCS12 -deststoretype PKCS12 -destkeypass password -deststorepass password; bin/kafka-console-consumer.sh --bootstrap-server '${KAFKA\_CLUSTER}'-kafka-bootstrap:9093 --consumer-property security.protocol=SSL --consumer-property ssl.truststore.location=workdir/truststore.p12 --consumer-property ssl.truststore.password=password --consumer-property ssl.truststore.type=PKCS12 --consumer-property ssl.keystore.location=workdir/keystore.p12 --consumer-property ssl.keystore.password=password --consumer-property ssl.keystore.type=PKCS12 --from-beginning --topic '${TOPIC\_NAME}' --group '${CONSUMER\_GROUP\_NAME}' "] }] }}' |
| --- |

### JMX

Kafka provides many metrics via JMX. For details of mbeans Kafka has for metrics, refer to the [document](https://kafka.apache.org/documentation/#monitoring).

###### Enable JMX port(9999) on the Kafka brokers

| apiVersion: kafka.strimzi.io/v1beta1 kind: Kafka metadata:  name: my-cluster spec:  kafka:  ...  jmxOptions: {} |
| --- |

For details of jmxOptions, refer to the [document](https://access.redhat.com/documentation/en-us/red_hat_amq/7.6/html-single/using_amq_streams_on_openshift/index#assembly-jmx-options-deployment-configuration-kafka). jmxOptions is available since AMQ Streams 1.4.

###### Print Kakfa metrics via JMX using kafka.tools.JmxTool

| # using an existing broker oc exec -it ${KAFKA\_CLUSTER}-kafka-0 -c kafka -- env - bin/kafka-run-class.sh kafka.tools.JmxTool --reporting-interval 1000 --object-name kafka.server:type=BrokerTopicMetrics,name=MessagesInPerSec  # using a new pod (connect via headless service) oc run kafka-exe -ti --image=registry.redhat.io/amq7/amq-streams-kafka-25-rhel7:1.5.0 --rm=true --restart=Never -- bin/kafka-run-class.sh kafka.tools.JmxTool --jmx-url service:jmx:rmi:///jndi/rmi://${KAFKA\_CLUSTER}-kafka-${BROKER\_ID}.${KAFKA\_CLUSTER}-kafka-brokers:9999/jmxrmi --reporting-interval 1000 --object-name kafka.server:type=BrokerTopicMetrics,name=MessagesInPerSec |
| --- |

This method can display metrics through JMX. To output only one time you can use the ”--one-time true” option.

###### jmxterm

| # get jmxterm tool jar curl -L -O https://github.com/jiaqi/jmxterm/releases/download/v1.0.2/jmxterm-1.0.2-uber.jar  # run interactive jmxterm java -jar jmxterm-1.0.2-uber.jar -l <jmx address>:<jmx port> |
| --- |

Jmxterm is an open source command line based interactive JMX client written in Java. It lets user access a Java MBean server in command line console without graphical environment. In another word, it's a command line alternative of [jconsole](https://en.wikipedia.org/wiki/JConsole). For details of jmxterm, refer to [jmxterm web page](https://docs.cyclopsgroup.org/jmxterm).

###### jconsole (or jmxterm) from localhost machine to a Kafka broker

| # add loopback address of the same IP address as a Kafka broker on the localhost machine sudo ip addr add $(oc get pod/${KAFKA\_CLUSTER}-kafka-${BROKER\_ID} -o=jsonpath='{.status.podIP}')/32 dev lo  # open port forwarding to a Kakfa broker on the localhost machine oc port-forward pod/${KAFKA\_CLUSTER}-kafka-${BROKER\_ID} 9999:9999 --address=$(oc get pod/${KAFKA\_CLUSTER}-kafka-${BROKER\_ID} -o=jsonpath='{.status.podIP}')  # [on another terminal] execute jconsole jconsole $(oc get pod/${KAFKA\_CLUSTER}-kafka-${BROKER\_ID} -o=jsonpath='{.status.podIP}'):9999 # Or, [on another terminal] execute jmxterm  java -jar jmxterm-1.0.2-uber.jar -l $(oc get pod/${KAFKA\_CLUSTER}-kafka-${BROKER\_ID} -o=jsonpath='{.status.podIP}'):9999  # FYI, delete loopback address you've created for cleanup sudo ip addr del $(oc get pod/${KAFKA\_CLUSTER}-kafka-${BROKER\_ID} -o=jsonpath='{.status.podIP}')/32 dev lo |
| --- |

This has the advantage of not polluting the AMQ Streams environment, but you need to create a loopback address on the localhost machine. ” -Djava.rmi.server.hostname=XX.XX.XX.XX” is specified in Kafka broker so If you don't use that loopback IP address to access the Kafka broker, the connection will be denied.

###### jmxterm from a new pod to a Kafka broker

| oc run jmxterm -ti --image=registry.redhat.io/rhel8/support-tools -- bash -c "dnf install -y curl java-1.8.0-openjdk; curl -L -O https://github.com/jiaqi/jmxterm/releases/download/v1.0.2/jmxterm-1.0.2-uber.jar; java -jar jmxterm-1.0.2-uber.jar -l ${KAFKA\_CLUSTER}-kafka-${BROKER\_ID}.${KAFKA\_CLUSTER}-kafka-brokers:9999" |
| --- |

This method is easy, but it takes time because it downloads the java package.

### JMX Exporter and Kafka Exporter

To enable JMX Exporter, for details please refer to [the AMQ Streams document](https://access.redhat.com/documentation/en-us/red_hat_amq/7.7/html-single/using_amq_streams_on_openshift/index#assembly-metrics-deployment-configuration-kafka). You can also use a sample Kafka yaml (please adjust the version of the YAML file in the GitHub).

| **Component** | **Custom resource** | **Example YAML file** |
| --- | --- | --- |
| Kafka and ZooKeeper | Kafka | [kafka-metrics.yaml](https://github.com/strimzi/strimzi-kafka-operator/blob/master/examples/metrics/kafka-metrics.yaml) |
| Kafka Connect | KafkaConnect and KafkaConnectS2I | [kafka-connect-metrics.yaml](https://github.com/strimzi/strimzi-kafka-operator/blob/master/examples/metrics/kafka-connect-metrics.yaml) |
| Kafka MirrorMaker 2.0 | KafkaMirrorMaker2 | [kafka-mirror-maker-2-metrics.yaml](https://github.com/strimzi/strimzi-kafka-operator/blob/master/examples/metrics/kafka-mirror-maker-2-metrics.yaml) |
| Kafka Bridge | KafkaBridge | [kafka-bridge-metrics.yaml](https://github.com/strimzi/strimzi-kafka-operator/blob/master/examples/metrics/kafka-metrics.yaml) |
| Cruise Control | Kafka | [kafka-cruise-control-metrics.yaml](https://github.com/strimzi/strimzi-kafka-operator/blob/master/examples/metrics/kafka-cruise-control-metrics.yaml) |

###### Show JMX exporter metrics using curl

| # Kafka broker  oc exec -it ${KAFKA\_CLUSTER}-kafka-${BROKER\_ID} -c kafka -- curl <http://localhost:9404/metrics>  # Zookeeper oc exec -it ${KAFKA\_CLUSTER}-zookeeper-${ZOOKEEPER\_ID} -- curl <http://localhost:9404/metrics> |
| --- |

###### Show Kafka exporter metrics using curl

| # since AMQ Streams 1.6 oc exec -it deployment/${KAFKA\_CLUSTER}-kafka-exporter -- curl http://localhost:9404/metrics |
| --- |

### Log

###### Log of a pod

| # Kafka broker oc logs ${KAFKA\_CLUSTER}-kafka-${BROKER\_ID} -c kafka oc logs -p ${KAFKA\_CLUSTER}-kafka-${BROKER\_ID} -c kafka oc logs ${KAFKA\_CLUSTER}-kafka-${BROKER\_ID} -c tls-sidecar oc logs -p ${KAFKA\_CLUSTER}-kafka-${BROKER\_ID} -c tls-sidecar  # Zookeeper oc logs ${KAFKA\_CLUSTER}-zookeeper-${ZOOKEEPER\_ID} -c zookeeper oc logs -p ${KAFKA\_CLUSTER}-zookeeper-${ZOOKEEPER\_ID} -c zookeeper oc logs ${KAFKA\_CLUSTER}-zookeeper-${ZOOKEEPER\_ID} -c tls-sidecar oc logs -p ${KAFKA\_CLUSTER}-zookeeper-${ZOOKEEPER\_ID} -c tls-sidecar  # Cluster operator oc logs deployment/strimzi-cluster-operator oc logs -p deployment/strimzi-cluster-operator  # Entity operator oc logs deployment/${KAFKA\_CLUSTER}-entity-operator -c topic-operator oc logs -p deployment/${KAFKA\_CLUSTER}-entity-operator -c topic-operator oc logs deployment/${KAFKA\_CLUSTER}-entity-operator -c user-operator  oc logs -p deployment/${KAFKA\_CLUSTER}-entity-operator -c user-operator  oc logs deployment/${KAFKA\_CLUSTER}-entity-operator -c tls-sidecar  oc logs -p deployment/${KAFKA\_CLUSTER}-entity-operator -c tls-sidecar  # Cruise Control oc logs deployment/${KAFKA\_CLUSTER}-cruise-control -c cruise-control oc logs -p deployment/${KAFKA\_CLUSTER}-cruise-control -c cruise-control oc logs deployment/${KAFKA\_CLUSTER}-cruise-control -c tls-sidecar  oc logs -p deployment/${KAFKA\_CLUSTER}-cruise-control -c tls-sidecar   # mirror maker : It only works when there is only one pod because it grep the pod name. oc logs $(oc get pods -l strimzi.io/cluster -o name | grep mirror-maker-) oc logs -p $(oc get pods -l strimzi.io/cluster -o name | grep mirror-maker-)  # Kafka exporter : It only works when there is only one pod because it grep the pod name. oc logs $(oc get pods -l strimzi.io/cluster -o name | grep kafka-exporter-) oc logs -p $(oc get pods -l strimzi.io/cluster -o name | grep kafka-exporter-)  # Kafka connect: It only works when there is only one pod because it grep the pod name. oc logs $(oc get pods -l strimzi.io/cluster -o name | grep connect-) oc logs -p $(oc get pods -l strimzi.io/cluster -o name | grep connect-)  # Kafka bridge: It only works when there is only one pod because it grep the pod name. oc logs $(oc get pods -l strimzi.io/cluster -o name | grep bridge-) oc logs -p $(oc get pods -l strimzi.io/cluster -o name | grep bridge-) |
| --- |

Use [dump script](#_5qqj158dwfa) for a comprehensive log output.

Use the “-f” option to keep to follow log.

Use the “-p” option to get the previous log.

###### Multiple pod logs using stern

| # Kafka brokers and Zookeepers logs stern "${KAFKA\_CLUSTER}-(kafka|zookeeper)" --tail 0  # cluster operator and entity-operator logs stern "(strimzi-cluster-operator|entity-operator)" --tail 0  # cluster operator, entity-operator, Kafka brokers, Zookeepers logs stern "(strimzi-cluster-operator|entity-operator|${KAFKA\_CLUSTER}-(kafka|zookeeper))" --tail 0 |
| --- |

To install stern, refer to <https://github.com/wercker/stern>.

###### Enable debug log : Kafka, Zookeeper, Entity Operator, Cruise Control and its tls-sidecar

| apiVersion: kafka.strimzi.io/v1beta1 kind: Kafka spec:  # ...  kafka:  # ...  logging:  type: inline  loggers:  kafka.root.logger.level: "DEBUG"  log4j.logger.kafka: "DEBUG"  log4j.logger.org.apache.kafka: "DEBUG"  tlsSidecar:  # ...   logLevel: debug  # ...  zookeeper:  # ...  logging:  type: inline  loggers:  zookeeper.root.logger: "DEBUG"  tlsSidecar:  # ...   logLevel: debug  # ...  entityOperator:  # ...  topicOperator:  # ...  logging:  type: inline  loggers:  rootLogger.level: DEBUG  # ...  userOperator:  # ...  logging:  type: inline  loggers:  rootLogger.level: DEBUG  # ...  tlsSidecar:  # ...   logLevel: debug  # ...  cruiseControl:  # ...  logging:  type: inline  loggers:  # since AMQ Streams 1.7  rootLogger.level: "DEBUG"  # until AMQ Streams 1.6  # you need to restart Cruise Control pod after changing about 5 minute later.   # For details, please refer to ENTMQST-2654  cruisecontrol.root.logger: "DEBUG"  tlsSidecar:  # ...   logLevel: debug  # FYI, you can confirm actual log4j file # actual Kafka log4j file oc exec ${KAFKA\_CLUSTER}-kafka-0 -- cat /opt/kafka/custom-config/log4j.properties  # actual Zookeeper log4j file oc exec ${KAFKA\_CLUSTER}-zookeeper-0 -- cat /opt/kafka/custom-config/log4j.properties  # actual topic operator log4j file in entity operator  oc exec deployment/${KAFKA\_CLUSTER}-entity-operator -c topic-operator -- cat /opt/topic-operator/custom-config/log4j2.properties  # actual user operator log4j file in entity operator oc exec deployment/${KAFKA\_CLUSTER}-entity-operator -c user-operator -- cat /opt/usecd plr-operator/custom-config/log4j2.properties  # actual cruise contorl log4j file # until AMQ Streams 1.6 oc exec deployment/${KAFKA\_CLUSTER}-cruise-control -- cat /opt/cruise-control/custom-config/log4j.properties # since AMQ Streams 1.7 oc exec deployment/${KAFKA\_CLUSTER}-cruise-control -- cat /opt/cruise-control/custom-config/log4j2.properties |
| --- |

The above is an example of setting the log level to DEBUG for root loggers. Enabling the DEBUG log levels results in a large amount of log output. Change the log level only where necessary. Note that several loggers may not be affected by the above settings because they have a log level specified by default. For details, refer to the [document](https://access.redhat.com/documentation/en-us/red_hat_amq/7.6/html-single/using_amq_streams_on_openshift/index#con-kafka-logging-deployment-configuration-kafka) or actual log4j file.

###### Enable debug log : Cluster Operator

| # since AMQ Streams 1.6  ## pattern 1 : change STRIMZI\_LOG\_LEVEL environment variable value vi install/cluster-operator/060-Deployment-strimzi-cluster-operator.yaml ~~~ spec:  ...  template:  ...  spec:  ...  containers:  ...  env:  ...  - name: STRIMZI\_LOG\_LEVEL  value: "DEBUG" ~~~  ## pattern 2 : change log configuration on the config map vi install/cluster-operator/050-ConfigMap-strimzi-cluster-operator.yaml ~~~ data:  log4j2.properties: | ...  #rootLogger.level = ${env:STRIMZI\_LOG\_LEVEL:-INFO} # comment out  rootLogger.level = DEBUG ~~~  # until AMQ Streams 1.5 vi install/cluster-operator/050-Deployment-strimzi-cluster-operator.yaml ~~~ spec:  template:  spec:  containers:  - name: strimzi-cluster-operator  env:  - name: STRIMZI\_LOG\_LEVEL  value: "DEBUG" ~~~ |
| --- |

###### Enable debug log : Kafka Bridge

| apiVersion: kafka.strimzi.io/v1beta1 kind: KafkaBridge spec:  # ...  logging:  type: inline  loggers:  bridge.root.logger: "DEBUG"  log4j.logger.io.strimzi.kafka.bridge: "DEBUG"  # FYI, you can confirm actual log4j file oc exec $(oc get pods -l strimzi.io/cluster -o name | grep bridge- | head -n 1) -- cat /opt/strimzi/custom-config/log4j.properties |
| --- |

The default configuration for healthy and ready endpoints is WARN level.

###### Enable debug log : Kafka Connect

| # when to use CR apiVersion: kafka.strimzi.io/v1beta1 kind: KafkaConnect spec:  # ...  logging:  type: inline  loggers:  connect.root.logger.level: "DEBUG"  # ...  # FYI, you can confirm actual log4j configuration oc exec deployment/${KAFKACONNECT\_CLUSTER}-connect -- curl http://localhost:8083/admin/loggers/  # Or  oc exec deployment/${KAFKACONNECT\_CLUSTER}-connect -- cat /opt/kafka/custom-config/log4j.properties |
| --- |

Same for “Kind: KafkaConnectS2I”.

###### Enable debug log : Kafka Mirror Maker v1

| apiVersion: kafka.strimzi.io/v1beta1 kind: KafkaMirrorMaker spec:  # ...  logging:  type: inline  loggers:  mirrormaker.root.logger: "DEBUG"  # ... |
| --- |

###### Enable debug log : Kafka Mirror Maker v2

| apiVersion: kafka.strimzi.io/v1beta1 kind: KafkaMirrorMaker2 spec:  # ...  logging:  type: inline  loggers:  connect.root.logger.level: "DEBUG"  # ... |
| --- |

### Kafka Client Example

For details, refer to [Apache Kafka client examples](https://github.com/strimzi/client-examples).

###### Run Kafka producer client example on the OpenShift

| # step 1:clone Kafka Client Example git clone https://github.com/strimzi/client-examples.git cd client-examples/java/kafka/producer  # step 2: build Kafka Client Example make build  # you need scripts directory and run.sh to execute the following build because Dockerfile expects it. mkdir scripts  cp ../streams/scripts/run.sh scripts  # step 3: Docker build on the OpenShift oc new-build --name=java-kafka-producer --strategy=docker --binary oc start-build java-kafka-producer --from-dir=. --follow  # step 4: you can modify client example  # step 5: adjust image and bootstrap address in deployment yaml sed -i "s|image: .\*|image: image-registry.openshift-image-registry.svc:5000/${NAMESPACE}/java-kafka-producer:latest|g" ../java-kafka-producer.yaml sed -i "s|value: my-cluster-kafka-bootstrap:9092.\*|value: ${KAFKA\_CLUSTER}-kafka-bootstrap:9092|g" ../java-kafka-producer.yaml  # step 6: you need to also create "my-topic" and "cipot-ym" topic for Kafka streams application  # step 7: apply deployment yaml oc apply -f ../java-kafka-producer.yaml  # FYI, you can delete the build configuration and imagestream for cleanup after debugging oc delete all -l build=java-kafka-producer oc delete -f ../java-kafka-producer.yaml |
| --- |

###### Run Kafka streams client example on the OpenShift

| # step 1:clone Kafka Client Example git clone https://github.com/strimzi/client-examples.git cd client-examples/java/kafka/streams  # step 2: build Kafka Client Example make build  # step 3: Docker build on the OpenShift oc new-build --name=java-kafka-streams --strategy=docker --binary oc start-build java-kafka-streams --from-dir=. --follow  # step 4: you can modify client example  # step 5: adjust image and bootstrap address in deployment yaml sed -i "s|image: .\*|image: image-registry.openshift-image-registry.svc:5000/${NAMESPACE}/java-kafka-streams:latest|g" ../java-kafka-streams.yaml sed -i "s|value: my-source-cluster-kafka-bootstrap:9092.\*|value: ${KAFKA\_CLUSTER}-kafka-bootstrap:9092|g" ../java-kafka-streams.yaml  # step 6: you need to also create "my-topic" and "cipot-ym" topic for Kafka streams application  # step 7: apply deployment yaml oc apply -f ../java-kafka-streams.yaml  # FYI, you can delete the build configuration and imagestream for cleanup after debugging oc delete all -l build=java-kafka-streams oc delete -f ../java-kafka-streams.yaml |
| --- |

### Debug

###### JavaVM Info

| # Kafka broker # using jcmd since AMQ Streams 1.6 oc exec ${KAFKA\_CLUSTER}-kafka-${BROKER\_ID} -c kafka -- jcmd kafka.Kafka VM.info  # Zookeeper # using jcmd since AMQ Streams 1.6 oc exec ${KAFKA\_CLUSTER}-zookeeper-${ZOOKEEPER\_ID} -c zookeeper -- jcmd org.apache.zookeeper.server.quorum.QuorumPeerMain VM.info  # Cluster operator # using jcmd since AMQ Streams 1.6 oc exec deployment/strimzi-cluster-operator -- jcmd io.strimzi.operator.cluster.Main VM.info  # Topic Operator in the Entity operator # using jcmd since AMQ Streams 1.6 oc exec deployment/${KAFKA\_CLUSTER}-entity-operator -c topic-operator -- jcmd io.strimzi.operator.topic.Main VM.info  # User Operator in the Entity operator # using jcmd since AMQ Streams 1.6 oc exec deployment/${KAFKA\_CLUSTER}-entity-operator -c user-operator -- jcmd io.strimzi.operator.user.Main VM.info  # Cruise Control # using jcmd since AMQ Streams 1.6 oc exec deployment/${KAFKA\_CLUSTER}-cruise-control -c cruise-control -- jcmd com.linkedin.kafka.cruisecontrol.KafkaCruiseControlMain VM.info |
| --- |

###### Thread Dump

| # Kafka broker # using jcmd since AMQ Streams 1.6 oc exec ${KAFKA\_CLUSTER}-kafka-${BROKER\_ID} -c kafka -- jcmd kafka.Kafka Thread.print # using kill -3 <PID> #[1] oc exec ${KAFKA\_CLUSTER}-kafka-${BROKER\_ID} -c kafka -- kill -3 1   # Zookeeper # using jcmd since AMQ Streams 1.6 oc exec ${KAFKA\_CLUSTER}-zookeeper-${ZOOKEEPER\_ID} -c zookeeper -- jcmd org.apache.zookeeper.server.quorum.QuorumPeerMain Thread.print # using kill -3 <PID> #[1] oc exec ${KAFKA\_CLUSTER}-zookeeper-${ZOOKEEPER\_ID} -c zookeeper -- kill -3 1  # Cluster operator # using jcmd since AMQ Streams 1.6 oc exec deployment/strimzi-cluster-operator -- jcmd io.strimzi.operator.cluster.Main Thread.print # using kill -3 <PID> #[1] oc exec deployment/strimzi-cluster-operator -- kill -3 1   # Topic Operator in the Entity operator # using jcmd since AMQ Streams 1.6 oc exec deployment/${KAFKA\_CLUSTER}-entity-operator -c topic-operator -- jcmd io.strimzi.operator.topic.Main Thread.print # using kill -3 <PID> #[1] oc exec deployment/${KAFKA\_CLUSTER}-entity-operator -c topic-operator -- kill -3 1   # User Operator in the Entity operator # using jcmd since AMQ Streams 1.6 oc exec deployment/${KAFKA\_CLUSTER}-entity-operator -c user-operator -- jcmd io.strimzi.operator.user.Main Thread.print # using kill -3 <PID> #[1] oc exec deployment/${KAFKA\_CLUSTER}-entity-operator -c user-operator -- kill -3 1   # Kafka connect # using kill -3 <PID> #[1] # cannot know which replica to connect to. so this can be used with a single replica  oc exec deployment/${KAFKACONNECT\_CLUSTER}-connect -- kill -3 1  # or, specify pod to connect to  oc exec pod/${KAFKACONNECT\_CLUSTER}-connect-XXXXXXX -- kill -3 1   # Cruise Control # using jcmd since AMQ Streams 1.6 oc exec deployment/${KAFKA\_CLUSTER}-cruise-control -c cruise-control -- jcmd com.linkedin.kafka.cruisecontrol.KafkaCruiseControlMain Thread.print # using kill -3 <PID> #[1] oc exec deployment/${KAFKA\_CLUSTER}-cruise-control -c cruise-control -- kill -3 1 |
| --- |

[1] To get thread dump, you can also use “kill -3 <PID>”. The thread dump will be output in \*java process’s stdout\* i.e. generally pod log, not the terminal you are using as below:

| # For example: # output thread dump to \*java process's\* stdout oc exec ${KAFKA\_CLUSTER}-kafka-${BROKER\_ID} -c kafka -- kill -3 1 # confirm the thread dump from the pod log oc logs ${KAFKA\_CLUSTER}-kafka-${BROKER\_ID} -c kafka |
| --- |

###### Packet capture using tcpdump

| # multi line commands # step 1: specify namespace and pod name export NAMESPACE=my-project export POD\_NAME=${KAFKA\_CLUSTER}-kafka-0 # step 2: get pod id export POD\_ID=$(oc debug node/$(oc get pod/${POD\_NAME} -o=jsonpath='{.spec.nodeName}') -- chroot /host crictl pods --namespace ${NAMESPACE} --name ${POD\_NAME} -q) # step 3: get pod process id on the node export POD\_PID=$(oc debug node/$(oc get pod/${POD\_NAME} -o=jsonpath='{.spec.nodeName}') -- chroot /host bash -c "runc state ${POD\_ID} | jq .pid") # step 4: get packet dump using tcpdump oc debug node/$(oc get pod/${POD\_NAME} -o=jsonpath='{.spec.nodeName}') -- bash -c "nsenter -n -t ${POD\_PID} -- tcpdump -w - 2>/dev/null" | tee ${POD\_NAME}\_$(date +\%d\_%m\_%Y-%H\_%M\_%S-%Z).pcap | tcpdump -r -   # single line command # step 1: specify namespace and pod name export NAMESPACE=my-project export POD\_NAME=${KAFKA\_CLUSTER}-kafka-0 # step 2: get packet dump using tcpdump oc debug node/$(oc get pod/${POD\_NAME} -o=jsonpath='{.spec.nodeName}') -- bash -c "NAMESPACE=${NAMESPACE}; POD\_NAME=${POD\_NAME};"'POD\_ID=$(chroot /host crictl pods --namespace ${NAMESPACE} --name ${POD\_NAME} -q); POD\_PID=$(chroot /host bash -c "runc state ${POD\_ID} | jq .pid"); bash -c "nsenter -n -t ${POD\_PID} -- tcpdump -l -w - 2>/dev/null"' | tee ${POD\_NAME}\_$(date +\%d\_%m\_%Y-%H\_%M\_%S-%Z).pcap | tcpdump -r - |
| --- |

For details, refer to [“How to use tcpdump inside OpenShift v4 Pod”](https://access.redhat.com/solutions/4569211)

###### Heap Histogram & Stastics

| # Kafka broker # using jcmd since AMQ Streams 1.6 oc exec ${KAFKA\_CLUSTER}-kafka-${BROKER\_ID} -c kafka -- jcmd kafka.Kafka GC.class\_histogram oc exec ${KAFKA\_CLUSTER}-kafka-${BROKER\_ID} -c kafka -- jcmd kafka.Kafka GC.class\_stats  # Zookeeper # using jcmd since AMQ Streams 1.6 oc exec ${KAFKA\_CLUSTER}-zookeeper-${ZOOKEEPER\_ID} -c zookeeper -- jcmd org.apache.zookeeper.server.quorum.QuorumPeerMain GC.class\_histogram oc exec ${KAFKA\_CLUSTER}-zookeeper-${ZOOKEEPER\_ID} -c zookeeper -- jcmd org.apache.zookeeper.server.quorum.QuorumPeerMain GC.class\_stats  # Cluster operator # using jcmd since AMQ Streams 1.6 oc exec deployment/strimzi-cluster-operator -- jcmd io.strimzi.operator.cluster.Main GC.class\_histogram oc exec deployment/strimzi-cluster-operator -- jcmd io.strimzi.operator.cluster.Main GC.class\_stats  # Topic Operator in the Entity operator # using jcmd since AMQ Streams 1.6 oc exec deployment/${KAFKA\_CLUSTER}-entity-operator -c topic-operator -- jcmd io.strimzi.operator.topic.Main GC.class\_histogram oc exec deployment/${KAFKA\_CLUSTER}-entity-operator -c topic-operator -- jcmd io.strimzi.operator.topic.Main GC.class\_stats  # User Operator in the Entity operator # using jcmd since AMQ Streams 1.6 oc exec deployment/${KAFKA\_CLUSTER}-entity-operator -c user-operator -- jcmd io.strimzi.operator.user.Main GC.class\_histogram oc exec deployment/${KAFKA\_CLUSTER}-entity-operator -c user-operator -- jcmd io.strimzi.operator.user.Main GC.class\_stats  # Cruise Control # using jcmd since AMQ Streams 1.6 oc exec deployment/${KAFKA\_CLUSTER}-cruise-control -c cruise-control -- jcmd com.linkedin.kafka.cruisecontrol.KafkaCruiseControlMain GC.class\_histogram oc exec deployment/${KAFKA\_CLUSTER}-cruise-control -c cruise-control -- jcmd com.linkedin.kafka.cruisecontrol.KafkaCruiseControlMain GC.class\_stats |
| --- |

###### Heap Dump

<https://access.redhat.com/articles/3135421> pls check this kcs for collecting the gc, thread dump,heap dump etc.

###### RHEL8 support tool image

| oc run rhel8-support-tools -ti --image=registry.redhat.io/rhel8/support-tools |
| --- |

It can be used to execute commands such as curl in the namespace. And you will be the root user so you can install various packages using dnf or yum.

###### Change AMQ Streams Kafka image: build a image on the local machine

| # step 1: create Dockerfile that extends amq-streams image. vi Dockerfile ~~~ FROM registry.redhat.io/amq7/amq-streams-kafka-25-rhel7:1.5.0 USER root:root RUN microdnf install -y curl java-1.8.0-openjdk-devel  USER 1001 ~~~  # step 2: build image podman build . -t default-route-openshift-image-registry.apps-crc.testing/${NAMESPACE}/amq-streams-debug:latest  # step 3: login and push image to the openshift repository (example of code ready container) podman login -u kubeadmin -p $(oc whoami -t) default-route-openshift-image-registry.apps-crc.testing podman push --tls-verify=false default-route-openshift-image-registry.apps-crc.testing/${NAMESPACE}/amq-streams-debug:latest  # step 4: set Kafka.spec.kafka.image to the new image path. vi kakfa-my-cluster.yaml ~~~ kind: Kafka ... spec:  kafka:  image: image-registry.openshift-image-registry.svc:5000/<$$NAMESPACE$$>/amq-streams-debug:latest ~~~ |
| --- |

###### Change AMQ Streams Kafka image: build a image on the OpenShift

| # step 1: create Dockerfile that extends amq-streams image. vi Dockerfile ~~~ FROM registry.redhat.io/amq7/amq-streams-kafka-25-rhel7:1.5.0 USER root:root RUN microdnf install -y curl java-1.8.0-openjdk-devel  USER 1001 ~~~  # step 2: Docker build on the OpenShift oc new-build --name=amq-streams-debug --strategy=docker --binary oc start-build amq-streams-debug --from-dir=. --follow  # step 3: set Kafka.spec.kafka.image to the new image path and apply. please adjust <$$NAMESPACE$$> vi kakfa-my-cluster.yaml ~~~ kind: Kafka ... spec:  kafka:  image: image-registry.openshift-image-registry.svc:5000/<$$NAMESPACE$$>/amq-streams-debug:latest ~~~  # FYI, you can delete the build configuration and imagestream for cleanup after debugging oc delete all -l build=amq-streams-debug |
| --- |

###### Remote Debugging: Kafka Broker

| # Step 1: Set KAFKA\_DEBUG and JAVA\_DEBUG\_OPTS in the Kafka resource ~~~  Kind: Kakfa spec:  kafka:  template:  kafkaContainer:  env:  - name: KAFKA\_DEBUG  value: "true"  - name: JAVA\_DEBUG\_OPTS  value: "-agentlib:jdwp=transport=dt\_socket,address=5005,server=y,suspend=n" ~~~  # Step 2: Forward local port to the debug port on the operator pod oc port-forward pod/${KAFKA\_CLUSTER}-kafka-${BROKER\_ID} 5005:5005  # Step 3: Debug using localhost:5005 port and IDE etc remotely. |
| --- |

###### Remote Debugging: Cluster Operator

| # Step 1: Set JAVA\_OPTS and disable health check vi install/cluster-operator/050-Deployment-strimzi-cluster-operator.yaml ~~~ spec:  template:  spec:  containers:  - name: strimzi-cluster-operator  env:  - name: JAVA\_OPTS  value: "-agentlib:jdwp=transport=dt\_socket,address=5005,server=y,suspend=n" ...  livenessProbe:  timeoutSeconds: 999999999  ~~~  # Step 2: Forward local port to the debug port on the operator pod oc port-forward deployment/strimzi-cluster-operator 5005:5005  # Step 3: Debug using localhost:5005 port and IDE etc remotely. |
| --- |

###### Remote Debugging: Topic Operator in Entity Operator

| # Step 1: Set JAVA\_OPTS  ~~~  Kind: Kafka  spec:  entityOperator:  template:  topicOperatorContainer:  env:  - name: JAVA\_OPTS  value: "-agentlib:jdwp=transport=dt\_socket,address=5005,server=y,suspend=n" ~~~  # Step 2: Forward local port to the debug port on the topic operator pod oc port-forward deployment/${KAFKA\_CLUSTER}-entity-operator 5005:5005  # Step 3: Debug using localhost:5005 port and IDE etc remotely. |
| --- |

###### Remote Debugging: User Operator in Entity Operator

| # Step 1: Set JAVA\_OPTS  ~~~ Kind: Kafka spec:  entityOperator:  template:  userOperatorContainer:  env:  - name: JAVA\_OPTS  value: "-agentlib:jdwp=transport=dt\_socket,address=5005,server=y,suspend=n" ~~~  # Step 2: Forward local port to the debug port on the user operator pod oc port-forward deployment/${KAFKA\_CLUSTER}-entity-operator 5005:5005  # Step 3: Debug using localhost:5005 port and IDE etc remotely. |
| --- |

###### Remote Debugging: Kafka Connect

| # Step 1: Set JAVA\_OPTS  ~~~ Kind: KafkaConnect spec:  template:  connectContainer:  env:  - name: KAFKA\_DEBUG  value: "true"  - name: JAVA\_DEBUG\_OPTS  value: "-agentlib:jdwp=transport=dt\_socket,address=5005,server=y,suspend=n" ~~~  # Step 2: Forward local port to the debug port on the kafka connect replica pod oc port-forward pod/${KAFKACONNECT\_CLUSTER}-connect-XXXXXXX 5005:5005  # Step 3: Debug using localhost:5005 port and IDE etc remotely. |
| --- |

###### Remote Debugging: Cruise Control

| # Step 1: Set KAFKA\_OPTS  ~~~ Kind: Kakfa spec:  cruiseControl:  template:  cruiseControlContainer:  env:  - name: KAFKA\_OPTS  value: "-agentlib:jdwp=transport=dt\_socket,address=5005,server=y,suspend=n" ~~~  # Step 2: Forward local port to the debug port on the kafka connect replica pod oc port-forward deployment/${KAFKA\_CLUSTER}-cruise-control 5005:5005  # Step 3: Debug using localhost:5005 port and IDE etc remotely. |
| --- |

###### Byteman: Kafka Broker

| # step 1: create Dockerfile that extends amq-streams image. "INITIALIZE\_COMMAND" will be executed before starting Kafka. vi Dockerfile ~~~ FROM registry.redhat.io/amq7/amq-streams-kafka-25-rhel7:1.5.0 USER root:root RUN microdnf install -y unzip RUN sed -i "s|# starting Kafka server.\*|$(echo 'eval ${INITIALIZE\_COMMAND}')|g" /opt/kafka/kafka\_run.sh USER 1001 ~~~  # step 2: Docker build on the OpenShift oc new-build --name=amq-streams-debug --strategy=docker --binary oc start-build amq-streams-debug --from-dir=. --follow  # step 3: set Kafka.spec.kafka.image to the new image path and apply. please adjust <$$NAMESPACE$$>. and set byteman rule in "BYTEMAN\_BTM".  vi kakfa-my-cluster.yaml ~~~ kind: Kafka ... spec:  kafka:  image: image-registry.openshift-image-registry.svc:5000/<$$NAMESPACE$$>/amq-streams-debug:latest  template:  kafkaContainer:  env:  - name: INITIALIZE\_COMMAND  value: "curl -o /var/opt/kafka/byteman.zip https://downloads.jboss.org/byteman/4.0.15/byteman-download-4.0.15-bin.zip;unzip /var/opt/kafka/byteman.zip -d /var/opt/kafka; ls;echo \"${BYTEMAN\_BTM}\" > /var/opt/kafka/appmain.btm"  - name: KAFKA\_DEBUG  value: "true"  - name: JAVA\_DEBUG\_OPTS  value: "-agentlib:jdwp=transport=dt\_socket,address=5005,server=y,suspend=n -javaagent:/var/opt/kafka/byteman-download-4.0.15/lib/byteman.jar=script:/var/opt/kafka/appmain.btm,boot:/var/opt/kafka/byteman-download-4.0.15/lib/byteman.jar"  - name: BYTEMAN\_BTM  value: |  RULE rule101entry  CLASS GroupMetadata  METHOD completePendingTxnOffsetCommit  AT ENTRY  IF true  DO delay(500);traceln("[BYTEMAN] ENTER " + $METHOD + ":" + java.util.Arrays.asList($\*));delay(500);  ENDRULE   RULE rule101exit  CLASS GroupMetadata  METHOD onTxnOffsetCommitAppend  AT EXIT  IF true  DO delay(500);traceln("[BYTEMAN] EXIT " + $METHOD);delay(500);  ENDRULE ~~~  # FYI, you can delete the build configuration and imagestream for cleanup after debugging oc delete all -l build=amq-streams-debug |
| --- |

### Tips

###### Dump OpenShift project

| # For OCP 4 # application project issue oc adm inspect ns/myproject1 ns/myproject2 ... # non-application project issue oc adm must-gather  # For OCP 3 curl -LO https://raw.githubusercontent.com/nekop/shiftbox/master/oc-dump chmod +x ./oc-dump ./oc-dump PROJECT |
| --- |

Refer to [How to get full project information in OpenShift](https://access.redhat.com/solutions/3340581)

###### OpenShift events (sort by last timestamp)

| oc get events --sort-by '.lastTimestamp' |
| --- |

To watch changes, use the “-w” option. If error occurs, try --sort-by '.metadata.creationTimestamp'

###### Force a pod termination once

| oc delete pod ${KAFKA\_CLUSTER}-kafka-0 --grace-period=0 --force |
| --- |

You can use this command to kill a Kafka broker or Zookeeper immediately. And OpenShift restarts the pot automatically.

###### Continues to force a pod termination

| watch -n 1 oc delete pod ${KAFKA\_CLUSTER}-kafka-0 --grace-period=0 --force |
| --- |

You can use this command to continue to kill a Kafka broker or Zookeeper.

I don't know if this is a good idea.

###### Login to Code Ready Container host node using ssh

| ssh -i ~/.crc/machines/crc/id\_rsa core@$(crc ip) |
| --- |

###### no matches for kind "Deployment" in version "extensions/v1beta1" error occurs

If you use old AMQ Streams 1.2 on OpenShift 4, when to install cluster operator, the following error occurs:

| > oc apply -f install/cluster-operator/050-Deployment-strimzi-cluster-operator.yaml error: unable to recognize "install/cluster-operator/050-Deployment-strimzi-cluster-operator.yaml": no matches for kind "Deployment" in version "extensions/v1beta1" |
| --- |

Try the following apiVersion and spec.selector:

| #apiVersion: extensions/v1beta1 apiVersion: apps/v1 kind: Deployment metadata:  name: strimzi-cluster-operator  labels:  app: strimzi spec:  replicas: 1  selector:  matchLabels:  name: strimzi-cluster-operator  strimzi.io/kind: cluster-operator |
| --- |

###### Long and short names for each Strimzi resource

| **Strimzi resource** | **Long name** | **Short name** |
| --- | --- | --- |
| Kafka | kafka | k |
| Kafka Topic | kafkatopic | kt |
| Kafka User | kafkauser | ku |
| Kafka Connect | kafkaconnect | kc |
| Kafka Connect S2I | kafkaconnects2i | kcs2i |
| Kafka Connector | kafkaconnector | kctr |
| Kafka Mirror Maker | kafkamirrormaker | kmm |
| Kafka Mirror Maker 2 | kafkamirrormaker2 | kmm2 |
| Kafka Bridge | kafkabridge | kb |
| Kafka Rebalance | kafkarebalance | kr |

### Dump AMQ Streams project

###### Dump resources, status, and logs of AMQ Streams on OpenShift

| ./report.sh --namespace=${NAMESPACE} --cluster=${KAFKA\_CLUSTER} |
| --- |

The dump includes "secrets". If you provide the dump to Red Hat support services, remove reports/secrets directory if necessary.

<https://access.redhat.com/solutions/5327571>

If you would like to get the project information itself as well, please refer to [How to get full project information in OpenShift](https://access.redhat.com/solutions/3340581)

###### Customizable report script

Similar to the above dump script, but the following script can be customized. Some users prefer to be clear on what commands to execute and provide the minimum information. The following script can be trimmed leaving only a few commands. And the following commands include actual Kafka topic information, not included in the above dump script.

| #!/bin/bash # Usage:  # chmod +x ./reportscript.sh # ./reportscript.sh <KAFKA\_CLUSTER> #e.g. my-cluster # you need to login to the namespace using AMQ Streams  # prepare DEST=amqstreams-$(date +%Y%m%d%H%M%S).txt.gz set -x KAFKA\_CLUSTER=$1  ( # start commands  oc version oc whoami oc get project $(oc project -q) -o yaml oc status  # (using jcmd since AMQ Streams 1.6) oc exec ${KAFKA\_CLUSTER}-kafka-0 -c kafka -- jcmd kafka.Kafka VM.info oc exec ${KAFKA\_CLUSTER}-zookeeper-0 -c zookeeper -- jcmd org.apache.zookeeper.server.quorum.QuorumPeerMain VM.info oc exec deployment/strimzi-cluster-operator -- jcmd io.strimzi.operator.cluster.Main VM.info  oc get Kafka -o yaml oc get KafkaTopic -o yaml oc get KafkaConnect -o yaml oc get KafkaConnectS2I -o yaml oc get KafkaConnector -o yaml oc get KafkaMirrorMaker -o yaml oc get KafkaMirrorMaker2 -o yaml oc get KafkaRebalance -o yaml oc get KafkaBridge -o yaml  oc get all -l app=strimzi -o yaml   oc get all -l strimzi.io/kind -o yaml # this command includes the following info # oc get statefulsets ${KAFKA\_CLUSTER}-kafka -o yaml # oc get statefulsets ${KAFKA\_CLUSTER}-zookeeper -o yaml # oc get deployment/${KAFKA\_CLUSTER}-entity-operator -o yaml  oc get service -o yaml oc get pvc -o yaml -l app.kubernetes.io/name=strimzi oc get pv -o yaml oc get storageclasses -o yaml oc get events --sort-by '.lastTimestamp' oc get events --sort-by '.metadata.creationTimestamp' oc get events  oc exec -it ${KAFKA\_CLUSTER}-kafka-0 -c kafka -- env - bin/kafka-configs.sh --bootstrap-server localhost:9092 --entity-type brokers --entity-name ${BROKER\_ID} --describe --all  oc exec -it ${KAFKA\_CLUSTER}-kafka-0 -c kafka -- env - bin/kafka-features.sh --bootstrap-server localhost:9092 --describe  oc exec ${KAFKA\_CLUSTER}-kafka-0 -- cat /tmp/strimzi.properties oc exec ${KAFKA\_CLUSTER}-kafka-0 -- cat /opt/kafka/custom-config/log4j.properties oc exec ${KAFKA\_CLUSTER}-kafka-0 -- ps aux oc exec ${KAFKA\_CLUSTER}-zookeeper-0 -- cat /tmp/zookeeper.properties oc exec ${KAFKA\_CLUSTER}-zookeeper-0 -- cat /opt/kafka/custom-config/log4j.properties oc exec ${KAFKA\_CLUSTER}-zookeeper-0 -- ps aux  # until AMQ Stremas 1.5 oc exec ${KAFKA\_CLUSTER}-kafka-0 -- bash -c "echo dump | nc localhost 2181"  oc get secret ${KAFKA\_CLUSTER}-cluster-ca-cert -o 'jsonpath={.data.ca\.crt}' | base64 -d | openssl x509 -subject -issuer -startdate -enddate -noout oc get secret ${KAFKA\_CLUSTER}-clients-ca-cert -o 'jsonpath={.data.ca\.crt}' | base64 -d | openssl x509 -subject -issuer -startdate -enddate -noout  oc logs ${KAFKA\_CLUSTER}-kafka-0 -c kafka  oc logs -p ${KAFKA\_CLUSTER}-kafka-0 -c kafka  oc logs ${KAFKA\_CLUSTER}-kafka-0 -c tls-sidecar  oc logs -p ${KAFKA\_CLUSTER}-kafka-0 -c tls-sidecar   oc logs ${KAFKA\_CLUSTER}-zookeeper-0 -c zookeeper  oc logs -p ${KAFKA\_CLUSTER}-zookeeper-0 -c zookeeper  oc logs ${KAFKA\_CLUSTER}-zookeeper-0 -c tls-sidecar  oc logs -p ${KAFKA\_CLUSTER}-zookeeper-0 -c tls-sidecar    oc logs deployment/strimzi-cluster-operator  oc logs -p deployment/strimzi-cluster-operator  oc logs deployment/${KAFKA\_CLUSTER}-entity-operator -c topic-operator oc logs -p deployment/${KAFKA\_CLUSTER}-entity-operator -c topic-operator oc logs deployment/${KAFKA\_CLUSTER}-entity-operator -c user-operator  oc logs -p deployment/${KAFKA\_CLUSTER}-entity-operator -c user-operator   TOPICS=$(oc exec -it ${KAFKA\_CLUSTER}-kafka-0 -c kafka -- env - KAFKA\_JVM\_PERFORMANCE\_OPTS=-XX:ParallelGCThreads=1 bin/kafka-topics.sh --zookeeper localhost:2181 --list) for TOPIC in $TOPICS do   oc exec -it ${KAFKA\_CLUSTER}-kafka-0 -c kafka -- env - bin/kafka-topics.sh --zookeeper localhost:2181 --describe --topic $TOPIC done  oc exec -it ${KAFKA\_CLUSTER}-kafka-0 -c kafka -- env - bin/kafka-consumer-groups.sh --bootstrap-server localhost:9092 --describe --all-groups   ) 2>&1 | gzip > $DEST echo "Generated $DEST" |
| --- |

You can split a dump file to files per command using the following command:

| zcat \*.txt.gz | awk 'BEGIN{f=""} match($0, /^+ ((oc|date).\*)$/, a){f=a[1] ".txt"; gsub(/[ \/=]/,"\_",f);} {print $0 >> f}' |
| --- |

### References

* Red Hat AMQ Documentation
  + <https://access.redhat.com/documentation/en-us/red_hat_amq/2020.q4/>
* Red Hat AMQ 7 Supported Configuration
  + <https://access.redhat.com/articles/2791941>
* Red Hat AMQ 7 Component Details
  + <https://access.redhat.com/articles/3188232>
* Red Hat AMQ Streams Download
  + <https://access.redhat.com/jbossnetwork/restricted/listSoftware.html?downloadType=distributions&product=jboss.amq.streams>
* Kafka Documentation
  + <https://kafka.apache.org/documentation/>
* Strimzi Documentation
  + <https://strimzi.io/docs/operators/latest/using.html>
* AMQ Streams sizing recommendations
  + <https://docs.google.com/document/d/1P02sMeLLGyIcfefc6TvdLklXPwTRE61hwRE9nREHbSA/edit#>
  + <https://docs.google.com/presentation/d/1GEIvsWu9F9MJnPvZs_fpujzmyPRI2-6bUH3aAt_rP1A/edit#slide=id.ga9c3ac4421_0_58>
* Strimzi 2.4 > 10.6. Tuning Kafka configuration
  + <https://strimzi.io/docs/operators/latest/using.html#assembly-tuning-config-str>
* AMQ Streams - FAQ
  + <https://docs.google.com/document/d/1pEUSHl4IFfCgXAUiu98MtowdAmH5V4qO60vDq0-ewSw/edit?usp=sharing>
* Red Hat Integration Capacity Guidance
  + <https://docs.google.com/document/d/1CumcxzS5ykmrsGEwnaL4FaNxYx2IBag1ZNNJQ6hwmjQ/edit#heading=h.mn40jvoqksnc>
* AMQ Streams Enablement Training Agenda
  + <https://docs.google.com/spreadsheets/d/17Et8LEo7SLSxjqtEgrLyzCqiBK1sH6dizVBo_W6rDJQ/edit#gid=1411071195>
* Internal Training Sessions for Fuse and AMQ (Integration)
  + <https://source.redhat.com/groups/public/gss-fusesource/fuseeducationcenter/fuse_education_center_wiki/internal_training_sessions_for_fuse_and_amq_integration#jive_content_id_AMQStreams_Kafka>
* AMQ Streams on OpenShift staging documentation (master branch)
  + <https://gitlab.cee.redhat.com/AMQ7-documentation/amq-streams/-/tree/master/books-ocp>
* AMQ Streams on Red Hat Enterprise Linux staging documentation (master branch)
  + <https://gitlab.cee.redhat.com/AMQ7-documentation/amq-streams/-/tree/master/books-rhel>

### 

### 

### 

### 

## Unorganized

###### Login to the pod’s node using debug container

| # step 1: specify namespace and pod name NAMESPACE=my-project POD\_NAME=${KAFKA\_CLUSTER}-kafka-0 # step 2: get pod id POD\_ID=$(oc debug node/$(oc get pod/${POD\_NAME} -o=jsonpath='{.spec.nodeName}') -- chroot /host crictl pods --namespace ${NAMESPACE} --name ${POD\_NAME} -q) # step 3: get pod process id on the node POD\_PID=$(oc debug node/$(oc get pod/${POD\_NAME} -o=jsonpath='{.spec.nodeName}') -- chroot /host bash -c "runc state ${POD\_ID} | jq .pid") # step 4: login the node oc debug node/$(oc get pod/${POD\_NAME} -o=jsonpath='{.spec.nodeName}') |
| --- |

###### Execute GDB from a node

| # step 1: login to the pod’s node using debug container  # refer to the above ‘login to the pod’s node using debug container’ section # step 2: install GDB dnf install -y gdb  # step 3: execute GDB gdb -p $PID |
| --- |

###### Kill a connection using GDB

| # step 1: login to the pod's node using debug container # refer to the above 'login to the pod's node using debug container' section # step 2: install GDB dnf install -y gdb # step 3: enter to pod's network namespace nsenter -n -t <POD\_PID> # step 4: check the port/pid mapping netstat -antp # step 5: check the pid/file descriptor mapping: It has postfix of "u" like "120u" which means "120". lsof -np <pid>  # step 6: close the connection’s file descriptor using GDB gdb -p $PID --batch -ex '{call (int)close(<file descriptor id>)}' |
| --- |

| export POD\_ID=$(oc debug node/$(oc get pod/${POD\_NAME} -o=jsonpath='{.spec.nodeName}') -- chroot /host crictl pods --namespace ${NAMESPACE} --name ${POD\_NAME} -q) export POD\_PID=$(oc debug node/$(oc get pod/${POD\_NAME} -o=jsonpath='{.spec.nodeName}') -- chroot /host bash -c "runc state ${POD\_ID} | jq .pid") env | grep POD     nsenter -n -t XXXXX chroot /host iptables -I INPUT -p tcp --dport 21810 -j REJECT --reject-with tcp-reset iptables -I OUTPUT -p tcp --sport 21810 -j REJECT --reject-with tcp-reset   iptables -D INPUT -p tcp --dport 21810 -j REJECT --reject-with tcp-reset iptables -D OUTPUT -p tcp --sport 21810 -j REJECT --reject-with tcp-reset     <https://access.redhat.com/solutions/30303> |
| --- |

###### Show actual network policy

| oc get NetworkPolicy/${KAFKA\_CLUSTER}-network-policy-kafka -o yaml oc get NetworkPolicy/${KAFKA\_CLUSTER}-network-policy-zookeeper -o yaml |
| --- |

###### 

###### 