Red Hat AMQ Streams 2.1

Release Notes for AMQ Streams 2.1 on OpenShift

Highlights of what's new and what's changed with this release of AMQ Streams on OpenShift Container Platform
Red Hat AMQ Streams 2.1 Release Notes for AMQ Streams 2.1 on OpenShift

Highlights of what’s new and what’s changed with this release of AMQ Streams on OpenShift Container Platform
Legal Notice

Copyright © 2022 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at http://creativecommons.org/licenses/by-sa/3.0/. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux® is the registered trademark of Linus Torvalds in the United States and other countries.

Java® is a registered trademark of Oracle and/or its affiliates.

XFS® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js® is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack® Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation’s permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

Abstract

The release notes summarize the new features, enhancements, and fixes introduced in the AMQ Streams 2.1 release.
Table of Contents

MAKING OPEN SOURCE MORE INCLUSIVE .......................................................... 3

CHAPTER 1. FEATURES .................................................................................. 4
  1.1. OPENSHIFT CONTAINER PLATFORM SUPPORT ................................. 4
  1.2. KAFKA 3.1.0 SUPPORT ......................................................................... 4
  1.3. SUPPORTING THE V1BETA2 API VERSION ....................................... 4
    1.3.1. Upgrading custom resources to v1beta2 ....................................... 5
  1.4. SUPPORT FOR IBM Z AND LINUXONE ARCHITECTURE ................. 5
    1.4.1. Requirements for IBM Z and LinuxONE ....................................... 5
    1.4.2. Unsupported on IBM Z and LinuxONE ....................................... 5
  1.5. SUPPORT FOR IBM POWER ARCHITECTURE .................................. 6
    1.5.1. Requirements for IBM Power ....................................................... 6
    1.5.2. Unsupported on IBM Power ....................................................... 6
  1.6. RENEWAL OF CUSTOM CA CERTIFICATES ..................................... 6
  1.7. DEBEZIUM FOR CHANGE DATA CAPTURE INTEGRATION .................. 7
  1.8. SERVICE REGISTRY ............................................................................. 7

CHAPTER 2. ENHANCEMENTS ...................................................................... 9
  2.1. KAFKA 3.1.0 ENHANCEMENTS ......................................................... 9
  2.2. RUNNING AMQ STREAMS ON A FIPS-ENABLED CLUSTER .................. 9
  2.3. CRUISE CONTROL INTRA-BROKER DISK BALANCING ...................... 9
  2.4. FEATURE GATES MOVE TO BETA MATURITY .................................... 10
  2.5. LOADBALANCER LISTENER BOOTSTRAP SERVICE ......................... 10
  2.6. OAUTH CONFIGURATION OPTIONS ............................................... 10

CHAPTER 3. TECHNOLOGY PREVIEWS ......................................................... 12
  3.1. KAFKA STATIC QUOTA PLUGIN CONFIGURATION .............................. 12
  3.2. CRUISE CONTROL FOR CLUSTER REBALANCING ......................... 12

CHAPTER 4. DEPRECATED FEATURES ......................................................... 14
  4.1. JAVA 8 .............................................................................................. 14
  4.2. KAFKA MIRRORMAKER 1 ................................................................. 14
  4.3. IDENTITY REPPLICATION POLICY .................................................. 14
  4.4. LISTENERSTATUS TYPE PROPERTY ............................................. 14
  4.5. CRUISE CONTROL CAPACITY CONFIGURATION ............................. 14

CHAPTER 5. FIXED ISSUES .......................................................................... 16

CHAPTER 6. KNOWN ISSUES ...................................................................... 18
  6.1. AMQ STREAMS CLUSTER OPERATOR ON IPV6 CLUSTERS .............. 18
  6.2. CRUISE CONTROL CPU UTILIZATION ESTIMATION ......................... 19

CHAPTER 7. SUPPORTED INTEGRATION PRODUCTS ..................................... 22

CHAPTER 8. IMPORTANT LINKS .................................................................... 23
MAKING OPEN SOURCE MORE INCLUSIVE

Red Hat is committed to replacing problematic language in our code, documentation, and web properties. We are beginning with these four terms: master, slave, blacklist, and whitelist. Because of the enormity of this endeavor, these changes will be implemented gradually over several upcoming releases. For more details, see our CTO Chris Wright’s message.
CHAPTER 1. FEATURES

AMQ Streams 2.1 introduces the features described in this section.

AMQ Streams version 2.1 is based on Strimzi 0.28.x.

NOTE
To view all the enhancements and bugs that are resolved in this release, see the AMQ Streams Jira project.

1.1. OPENShift Container Platform SUPPORT

AMQ Streams 2.1 is supported on OpenShift Container Platform 4.6 to 4.10.

For more information about the supported platform versions, see the AMQ Streams Supported Configurations.

1.2. KAFKA 3.1.0 SUPPORT

AMQ Streams now supports Apache Kafka version 3.1.0.

AMQ Streams uses Kafka 3.1.0. Only Kafka distributions built by Red Hat are supported.

You must upgrade the Cluster Operator to AMQ Streams version 2.1 before you can upgrade brokers and client applications to Kafka 3.1.0. For upgrade instructions, see Upgrading AMQ Streams.

Refer to the Kafka 3.0.0 and Kafka 3.1.0 Release Notes for additional information.

NOTE
Kafka 3.0.x is supported only for the purpose of upgrading to AMQ Streams 2.1.

For more information on supported versions, see the AMQ Streams Component Details.

Kafka 3.1.0 requires ZooKeeper version 3.6.3, which is the same version as Kafka 3.0.0. Therefore, the Cluster Operator will not perform a ZooKeeper upgrade when you upgrade from AMQ Streams 2.0 to AMQ Streams 2.1.

1.3. SUPPORTING THE V1BETA2 API VERSION

The v1beta2 API version for all custom resources was introduced with AMQ Streams 1.7. For AMQ Streams 1.8, v1alpha1 and v1beta1 API versions were removed from all AMQ Streams custom resources apart from KafkaTopic and KafkaUser.

Upgrade of the custom resources to v1beta2 prepares AMQ Streams for a move to Kubernetes CRD v1, which is required for Kubernetes v1.22.

If you are upgrading from an AMQ Streams version prior to version 1.7:

1. Upgrade to AMQ Streams 1.7

2. Convert the custom resources to v1beta2
3. Upgrade to AMQ Streams 1.8

**IMPORTANT**

You must upgrade your custom resources to use API version **v1beta2** before upgrading to AMQ Streams version 2.1.

See [Deploying and upgrading AMQ Streams](#).

### 1.3.1. Upgrading custom resources to v1beta2

To support the upgrade of custom resources to **v1beta2**, AMQ Streams provides an **API conversion tool**, which you can download from the [AMQ Streams software downloads page](#).

You perform the custom resources upgrades in two steps.

**Step one: Convert the format of custom resources**

Using the API conversion tool, you can convert the format of your custom resources into a format applicable to **v1beta2** in one of two ways:

- Converting the YAML files that describe the configuration for AMQ Streams custom resources
- Converting AMQ Streams custom resources directly in the cluster

Alternatively, you can manually convert each custom resource into a format applicable to **v1beta2**. Instructions for manually converting custom resources are included in the documentation.

**Step two: Upgrade CRDs to v1beta2**

Next, using the API conversion tool with the **crd-upgrade** command, you must set **v1beta2** as the storage API version in your CRDs. You cannot perform this step manually.

For full instructions, see [Upgrading AMQ Streams](#).

### 1.4. SUPPORT FOR IBM Z AND LINUXONE ARCHITECTURE

AMQ Streams 2.1 is enabled to run on IBM Z and LinuxONE s390x architecture.

Support for IBM Z and LinuxONE applies to AMQ Streams running with Kafka 3.1.0 on OpenShift Container Platform 4.10. The Kafka versions shipped with AMQ Streams 2.0 and earlier versions do not contain the s390x binaries.

#### 1.4.1. Requirements for IBM Z and LinuxONE

- OpenShift Container Platform 4.10
- Kafka 3.1.0

#### 1.4.2. Unsupported on IBM Z and LinuxONE

- Kafka 3.0.0 or earlier
- AMQ Streams upgrades and downgrades since this is the first release on s390x
• AMQ Streams on disconnected OpenShift Container Platform environments
• AMQ Streams OPA integration

1.5. SUPPORT FOR IBM POWER ARCHITECTURE

AMQ Streams 2.1 is enabled to run on IBM Power ppc64le architecture.

Support for IBM Power applies to AMQ Streams running with Kafka 3.0.0 and later on OpenShift Container Platform 4.9 and later. The Kafka versions shipped with AMQ Streams 1.8 and earlier versions do not contain the ppc64le binaries.

1.5.1. Requirements for IBM Power

• OpenShift Container Platform 4.9 and later
• Kafka 3.0.0 and later

1.5.2. Unsupported on IBM Power

• Kafka 2.8.0 and earlier
• AMQ Streams on disconnected OpenShift Container Platform environments

1.6. RENEWAL OF CUSTOM CA CERTIFICATES

The Cluster Operator can now detect user-provided custom CA certificates. When you renew your custom certificates, the Cluster Operator will perform a rolling update of ZooKeeper, Kafka, and other components to trust the new CA certificate.

If you are using your own certificates, the Cluster Operator does not renew them automatically. Instead, you need to edit the existing Secret to add the new CA certificate and update a certificate generation annotation value. The annotation is set to a higher incremental value so that the Cluster Operator uses the latest certificate in the renewal process.

Example secret configuration updated with a new CA certificate

```yaml
apiVersion: v1
kind: Secret
data:
  ca.crt: GCa6LS3RTHeKFiFDGBOUDYFAZ0F... 1
metadata:
  annotations:
    strimzi.io/ca-cert-generation: "1" 2
  labels:
    strimzi.io/cluster: my-cluster
    strimzi.io/kind: Kafka
  name: my-cluster-cluster-ca-cert
#...
type: Opaque
```

1 Base64-encoded CA certificate

2 CA certificate generation annotation value
1.7. DEBEZIUM FOR CHANGE DATA CAPTURE INTEGRATION

Red Hat Debezium is a distributed change data capture platform. It captures row-level changes in databases, creates change event records, and streams the records to Kafka topics. Debezium is built on Apache Kafka. You can deploy and integrate Debezium with AMQ Streams. Following a deployment of AMQ Streams, you deploy Debezium as a connector configuration through Kafka Connect. Debezium passes change event records to AMQ Streams on OpenShift. Applications can read these change event streams and access the change events in the order in which they occurred.

Debezium has multiple uses, including:

- Data replication
- Updating caches and search indexes
- Simplifying monolithic applications
- Data integration
- Enabling streaming queries

Debezium provides connectors (based on Kafka Connect) for the following common databases:

- Db2
- MongoDB
- MySQL
- PostgreSQL
- SQL Server

For more information on deploying Debezium with AMQ Streams, refer to the product documentation.

1.8. SERVICE REGISTRY

You can use Service Registry as a centralized store of service schemas for data streaming. For Kafka, you can use Service Registry to store Apache Avro or JSON schema.

Service Registry provides a REST API and a Java REST client to register and query the schemas from client applications through server-side endpoints.

Using Service Registry decouples the process of managing schemas from the configuration of client applications. You enable an application to use a schema from the registry by specifying its URL in the client code.

For example, the schemas to serialize and deserialize messages can be stored in the registry, which are then referenced from the applications that use them to ensure that the messages that they send and receive are compatible with those schemas.

Kafka client applications can push or pull their schemas from Service Registry at runtime.
For more information on using Service Registry with AMQ Streams, refer to the Service Registry documentation.
AMQ Streams 2.1 adds a number of enhancements.

### 2.1. Kafka 3.1.0 Enhancements

For an overview of the enhancements introduced with Kafka 3.1.0, refer to the [Kafka 3.1.0 Release Notes](https://kafka.apache.org/releases/3.1.0).

### 2.2. Running AMQ Streams on a FIPS-Enabled Cluster

You can now run AMQ Streams on a FIPS-enabled cluster, although currently not in a FIPS-compliant configuration.

The OpenJDK used in AMQ Streams container images will automatically switch to FIPS mode on a FIPS-enabled cluster. This prevents AMQ Streams from running on the cluster.

To run AMQ Streams on a FIPS-enabled cluster, you disable the OpenJDK FIPS mode by setting a `FIPS_MODE` environment variable to disabled in the deployment configuration for the Cluster Operator. The AMQ Streams deployment won’t be FIPS compliant, but the AMQ Streams operators as well as all of its operands will be able to run on the FIPS-enabled Kubernetes cluster.

**Example FIPS configuration for the Cluster Operator**

```yaml
apiVersion: apps/v1
kind: Deployment
spec:
  # ...
  template:
    spec:
      serviceAccountName: strimzi-cluster-operator
      containers:
        # ...
        env:
        # ...
        - name: "FIPS_MODE"
          value: "disabled"  # "disabled"
      # ...
```

1. Disables the FIPS mode.

See [Configuring FIPS mode in the Cluster Operator](https://kafka.strimzi.io/docs/operators/cluster-operator/#configuring-fips-mode).

### 2.3. Cruise Control Intra-Broker Disk Balancing

**NOTE**

Cruise Control remains in Technology Preview.

If you are running a Kafka deployment that uses JBOD storage with multiple disks on the same broker, Cruise Control can balance partitions between the disks.
You use the `rebalanceDisk` configuration option. To perform an intra-broker disk balance, you set `rebalanceDisk` to `true` under the `KafkaRebalance.spec`.

See Rebalance performance tuning.

### 2.4. FEATURE GATES MOVE TO BETA MATURITY

The feature gates `ControlPlaneListener` and `ServiceAccountPatching` move to beta maturity. This means that they are both enabled by default.

Feature gates at the beta level of maturity are well tested and their functionality is not likely to change.

See Configuring feature gates and Feature gate releases.

**IMPORTANT**

The `ControlPlaneListener` feature gate must be disabled when upgrading from or downgrading to AMQ Streams 1.7 and earlier versions.

### 2.5. LOADBALANCER LISTENER BOOTSTRAP SERVICE

A new listener configuration property let's you control whether or not to create a bootstrap service for a `loadBalancer` type of listener. A `<cluster_name>-kafka-external-bootstrap` bootstrap service is created by default for a Kafka cluster. You can choose not to create the service for a loadbalancer by setting the `createBootstrapService` property to false in the listener configuration.

**Example configuration for a loadbalancer external listener that does not create a bootstrap service**

```yaml
listeners:
  #...
  - name: external
    port: 9094
    type: loadbalancer
    tls: true
    authentication:
      type: tls
    configuration:
      createBootstrapService: false
  # ...
```

See GenericKafkaListenerConfiguration schema properties.

### 2.6. OAUTH CONFIGURATION OPTIONS

New OAuth configuration properties have been introduced to the OAuth authentication configuration.

The properties related to timeouts and extracting groups information.

**Timeout properties**

- `connectTimeoutSeconds` specifies the maximum time in seconds to connect to an authorization server before a timeout.
- **readTimeoutSeconds** specifies the maximum time in seconds to read from an authorization server before a timeout.

The default is sixty seconds for both.

**Groups properties**

- **groupsClaim** specifies a JsonPath query to extract groups information from a JWT token or introspection endpoint response. Not set by default.

- **groupsClaimDelimiter** specifies a delimiter to parse groups information when returned as a single delimited string. The default value is ','. (comma).

**Example OAuth configuration for a Kafka broker listener**

```yaml
#...
- name: external
  port: 9094
  type: loadbalancer
  tls: true
  authentication:
    type: oauth
    # ...
    connectTimeoutSeconds: 60
    readTimeoutSeconds: 60
    groupsClaim: "$.groups"
    groupsClaimDelimiter: ","
```

See KafkaListenerAuthenticationOAuth schema reference and KafkaClientAuthenticationOAuth schema properties.
CHAPTER 3. TECHNOLOGY PREVIEWS

IMPORTANT

Technology Preview features are not supported with Red Hat production service-level agreements (SLAs) and might not be functionally complete; therefore, Red Hat does not recommend implementing any Technology Preview features in production environments. This Technology Preview feature provides early access to upcoming product innovations, enabling you to test functionality and provide feedback during the development process. For more information about support scope, see Technology Preview Features Support Scope.

3.1. KAFKA STATIC QUOTA PLUGIN CONFIGURATION

Use the Kafka Static Quota plugin to set throughput and storage limits on brokers in your Kafka cluster. You enable the plugin and set limits by configuring the Kafka resource. You can set a byte-rate threshold and storage quotas to put limits on the clients interacting with your brokers.

Example Kafka Static Quota plugin configuration

```yaml
apiVersion: kafka.strimzi.io/v1beta2
kind: Kafka
metadata:
  name: my-cluster
spec:
  kafka:
    # ...
  config:
    client.quota.callback.class: io.strimzi.kafka.quotas.StaticQuotaCallback
    client.quota.callback.static.produce: 1000000
    client.quota.callback.static.fetch: 1000000
    client.quota.callback.static.storage.soft: 400000000000
    client.quota.callback.static.storage.hard: 500000000000
    client.quota.callback.static.storage.check-interval: 5
```

See Setting limits on brokers using the Kafka Static Quota plugin.

3.2. CRUISE CONTROL FOR CLUSTER REBALANCING

You can deploy Cruise Control and use it to rebalance your Kafka cluster using optimization goals—defined constraints on CPU, disk, network load, and more. In a balanced Kafka cluster, the workload is more evenly distributed across the broker pods.

Cruise Control is configured and deployed as part of a Kafka resource. You can use the default optimization goals or modify them to suit your requirements. Example YAML configuration files for Cruise Control are provided in examples/cruise-control/.

When Cruise Control is deployed, you can create KafkaRebalance custom resources to:

- Generate optimization proposals from multiple optimization goals
- Rebalance a Kafka cluster based on an optimization proposal
Other Cruise Control features are not currently supported, including anomaly detection, notifications, write-your-own goals, and changing the topic replication factor.

See Cruise Control for cluster rebalancing.
CHAPTER 4. DEPRECATED FEATURES

The features deprecated in this release, and that were supported in previous releases of AMQ Streams, are outlined below.

4.1. JAVA 8

Support for Java 8 was deprecated in Kafka 3.0.0. Support for Java 8 is also being removed in AMQ Streams for all components, including clients.

4.2. KAFKA MIRRORMAKER 1

Kafka MirrorMaker replicates data between two or more active Kafka clusters, within or across data centers. Kafka MirrorMaker 1 is deprecated for Kafka 3.0.0 and will be removed in Kafka 4.0.0. MirrorMaker 2.0 will be the only version available. MirrorMaker 2.0 is based on the Kafka Connect framework, connectors managing the transfer of data between clusters.

As a consequence, the AMQ Streams KafkaMirrorMaker custom resource which is used to deploy Kafka MirrorMaker 1 has been deprecated. The KafkaMirrorMaker resource will be removed from AMQ Streams when Kafka 4.0.0 is adopted.

If you are using MirrorMaker 1 (referred to as just MirrorMaker in the AMQ Streams documentation), use the KafkaMirrorMaker2 custom resource with the IdentityReplicationPolicy. MirrorMaker 2.0 renames topics replicated to a target cluster. IdentityReplicationPolicy configuration overrides the automatic renaming. Use it to produce the same active/passive unidirectional replication as MirrorMaker 1.

See Kafka MirrorMaker 2.0 cluster configuration.

4.3. IDENTITY REPLICATION POLICY

Identity replication policy is used with MirrorMaker 2.0 to override the automatic renaming of remote topics. Instead of prepending the name with the name of the source cluster, the topic retains its original name. This optional setting is useful for active/passive backups and data migration.

The AMQ Streams Identity Replication Policy class (io.strimzi.kafka.connect.mirror.IdentityReplicationPolicy) is now deprecated and will be removed in the future. You can update to use Kafka’s own Identity Replication Policy (class org.apache.kafka.connect.mirror.IdentityReplicationPolicy).

See Kafka MirrorMaker 2.0 cluster configuration.

4.4. LISTENERSTATUS TYPE PROPERTY

The type property of ListenerStatus has been deprecated and will be removed in the future. ListenerStatus is used to specify the addresses of internal and external listeners. Instead of using the type, the addresses are now specified by name.

See ListenerStatus schema reference.

4.5. CRUISE CONTROL CAPACITY CONFIGURATION

The disk and cpuUtilization capacity configuration properties have been deprecated, are ignored, and will be removed in the future. The properties were used in setting capacity limits in optimization.
proposals to determine if resource-based optimization goals are being broken. Disk and CPU capacity limits are now automatically generated by AMQ Streams.

See Cruise Control configuration.
CHAPTER 5. FIXED ISSUES

The issues fixed in AMQ Streams 2.1 are shown in the following table. For details of the issues fixed in Kafka 3.1.0, refer to the Kafka 3.1.0 Release Notes.

<table>
<thead>
<tr>
<th>Issue Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENTMQST-3595</td>
<td>Cluster operator missing Java options to be passed to Kafka bridge</td>
</tr>
<tr>
<td>ENTMQST-3835</td>
<td>Connector gets restarted on every reconciliation when tasks.max is set</td>
</tr>
<tr>
<td>ENTMQST-3763</td>
<td>Errors when scaling-down ZooKeeper nodes</td>
</tr>
<tr>
<td>ENTMQST-3422</td>
<td>Failure to run on FIPS-enabled clusters</td>
</tr>
<tr>
<td>ENTMQST-3417</td>
<td>Fix leaking keystores/truststores in ZooKeeperScaler</td>
</tr>
<tr>
<td>ENTMQST-3583</td>
<td>JVM options provided by the Cluster Operator are ignored</td>
</tr>
<tr>
<td>ENTMQST-3345</td>
<td>Kafka upgrade, with inter broker protocol and log message format as M.m.p, fails with misleading error</td>
</tr>
<tr>
<td>ENTMQST-3411</td>
<td>KafkaExporter, CruiseControl and EntityOperator pods are rolled on clients CA renewal</td>
</tr>
<tr>
<td>ENTMQST-3325</td>
<td>KafkaMirrorMaker2 conditions do not reflect the state of the MM2 connectors</td>
</tr>
<tr>
<td>ENTMQST-3504</td>
<td>OptimizationFailureException due to invalid CPU utilization</td>
</tr>
<tr>
<td>ENTMQST-3585</td>
<td>Pass Java system properties to Cruise Control</td>
</tr>
<tr>
<td>ENTMQST-3856</td>
<td>Rack Awareness doesn’t work for connectors</td>
</tr>
<tr>
<td>ENTMQST-3354</td>
<td>Set the base image in Kafka Connect Build properly when it is specified in the custom resource</td>
</tr>
<tr>
<td>ENTMQST-3826</td>
<td>The /tmp volume is not big enough for the compression libraries</td>
</tr>
<tr>
<td>ENTMQST-3584</td>
<td>The strimzi_resources{kind=&quot;Kafka&quot;} metric is not removed when the Kafka related namespace is deleted</td>
</tr>
<tr>
<td>ENTMQST-3839</td>
<td>The broker is stuck in an inconsistent state after ZooKeeper disconnection</td>
</tr>
<tr>
<td>ENTMQST-2331</td>
<td>ZooKeeper, Kafka, and EntityOperator certificates are not renewed using your own cluster CA certificate</td>
</tr>
</tbody>
</table>

Table 5.1. Fixed common vulnerabilities and exposures (CVEs)
<table>
<thead>
<tr>
<th>Issue Number</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENTMQST-2851</td>
<td>CVE-2021-3520 lz4: memory corruption due to an integer overflow bug caused by memmove argument</td>
</tr>
<tr>
<td>ENTMQST-3631</td>
<td>CVE-2021-43797 netty: control chars in header names may lead to HTTP request smuggling</td>
</tr>
</tbody>
</table>
CHAPTER 6. KNOWN ISSUES

This section lists the known issues for AMQ Streams 2.1.

6.1. AMQ STREAMS CLUSTER OPERATOR ON IPV6 CLUSTERS

The AMQ Streams Cluster Operator does not start on Internet Protocol version 6 (IPv6) clusters.

Workaround

There are two workarounds for this issue.

Workaround one: Set the KUBERNETES_MASTER environment variable

1. Display the address of the Kubernetes master node of your OpenShift Container Platform cluster:

   oc cluster-info
   Kubernetes master is running at <master_address>
   # ...

   Copy the address of the master node.

2. List all Operator subscriptions:

   oc get subs -n <operator_namespace>

3. Edit the Subscription resource for AMQ Streams:

   oc edit sub amq-streams -n <operator_namespace>

4. In spec.config.env, add the KUBERNETES_MASTER environment variable, set to the address of the Kubernetes master node. For example:

   apiVersion: operators.coreos.com/v1alpha1
   kind: Subscription
   metadata:
     name: amq-streams
     namespace: OPERATOR-NAMESPACE
   spec:
     channel: amq-streams-1.8.x
     installPlanApproval: Automatic
     name: amq-streams
     source: mirror-amq-streams
     sourceNamespace: openshift-marketplace
     config:
       env:
         - name: KUBERNETES_MASTER
           value: MASTER-ADDRESS

5. Save and exit the editor.

6. Check that the Subscription was updated:
oc get sub amq-streams -n <operator_namespace>

7. Check that the Cluster Operator Deployment was updated to use the new environment variable:

oc get deployment <cluster_operator_deployment_name>

Workaround two: Disable hostname verification

1. List all Operator subscriptions:

oc get subs -n OPERATOR-NAMESPACE

2. Edit the Subscription resource for AMQ Streams:

oc edit sub amq-streams -n <operator_namespace>

3. In spec.config.env, add the KUBERNETES_DISABLE_HOSTNAME_VERIFICATION environment variable, set to true. For example:

```yaml
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: amq-streams
  namespace: OPERATOR-NAMESPACE
spec:
  channel: amq-streams-1.8.x
  installPlanApproval: Automatic
  name: amq-streams
  source: mirror-amq-streams
  sourceNamespace: openshift-marketplace
  config:
    env:
      - name: KUBERNETES_DISABLE_HOSTNAME_VERIFICATION
        value: "true"
```

4. Save and exit the editor.

5. Check that the Subscription was updated:

oc get sub amq-streams -n <operator_namespace>

6. Check that the Cluster Operator Deployment was updated to use the new environment variable:

oc get deployment <cluster_operator_deployment_name>

6.2. CRUISE CONTROL CPU UTILIZATION ESTIMATION

Cruise Control for AMQ Streams has a known issue that relates to the calculation of CPU utilization estimation. CPU utilization is calculated as a percentage of the defined capacity of a broker pod. The issue occurs when running Kafka brokers across nodes with varying CPU cores. For example, node1
might have 2 CPU cores and node2 might have 4 CPU cores. In this situation, Cruise Control can underestimate and overestimate CPU load of brokers. The issue can prevent cluster rebalances when the pod is under heavy load.

**Workaround**

There are two workarounds for this issue.

**Workaround one: Equal CPU requests and limits**

You can set CPU requests equal to CPU limits in `Kafka.spec.kafka.resources`. That way, all CPU resources are reserved upfront and are always available. This configuration allows Cruise Control to properly evaluate the CPU utilization when preparing the rebalance proposals based on CPU goals.

**Workaround two: Exclude CPU goals**

You can exclude CPU goals from the hard and default goals specified in the Cruise Control configuration.

**Example Cruise Control configuration without CPU goals**

```yaml
apiVersion: kafka.strimzi.io/v1beta2
kind: Kafka
metadata:
  name: my-cluster
spec:
  kafka:
    # ...
  zookeeper:
    # ...
  entityOperator:
    topicOperator: {}
  userOperator: {}
  cruiseControl:
    brokerCapacity:
      inboundNetwork: 10000KB/s
      outboundNetwork: 10000KB/s
    config:
      hard.goals: >
      com.linkedin.kafka.cruisecontrol.analyzer.goals.RackAwareGoal,
      com.linkedin.kafka.cruisecontrol.analyzer.goals.MinTopicLeadersPerBrokerGoal,
      com.linkedin.kafka.cruisecontrol.analyzer.goals.ReplicaCapacityGoal,
      com.linkedin.kafka.cruisecontrol.analyzer.goals.DiskCapacityGoal,
      com.linkedin.kafka.cruisecontrol.analyzer.goals.NetworkInboundCapacityGoal,
      com.linkedin.kafka.cruisecontrol.analyzer.goals.NetworkOutboundCapacityGoal
      default.goals: >
      com.linkedin.kafka.cruisecontrol.analyzer.goals.RackAwareGoal,
      com.linkedin.kafka.cruisecontrol.analyzer.goals.MinTopicLeadersPerBrokerGoal,
      com.linkedin.kafka.cruisecontrol.analyzer.goals.ReplicaCapacityGoal,
      com.linkedin.kafka.cruisecontrol.analyzer.goals.DiskCapacityGoal,
      com.linkedin.kafka.cruisecontrol.analyzer.goals.NetworkInboundCapacityGoal,
      com.linkedin.kafka.cruisecontrol.analyzer.goals.NetworkOutboundCapacityGoal,
      com.linkedin.kafka.cruisecontrol.analyzer.goals.ReplicaDistributionGoal,
      com.linkedin.kafka.cruisecontrol.analyzer.goals.PotentialNwOutGoal,
      com.linkedin.kafka.cruisecontrol.analyzer.goals.DiskUsageDistributionGoal,
      com.linkedin.kafka.cruisecontrol.analyzer.goals.NetworkInboundUsageDistributionGoal,
      com.linkedin.kafka.cruisecontrol.analyzer.goals.NetworkOutboundUsageDistributionGoal,
```

Red Hat AMQ Streams 2.1 Release Notes for AMQ Streams 2.1 on OpenShift

20
For more information, see Insufficient CPU capacity.
CHAPTER 7. SUPPORTED INTEGRATION PRODUCTS

AMQ Streams 2.1 supports integration with the following Red Hat products.

Red Hat Single Sign-On
   Provides OAuth 2.0 authentication and OAuth 2.0 authorization.

Red Hat 3scale API Management
   Secures the Kafka Bridge and provides additional API management features.

Red Hat Debezium
   Monitors databases and creates event streams.

Red Hat Service Registry
   Provides a centralized store of service schemas for data streaming.

For information on the functionality these products can introduce to your AMQ Streams deployment, refer to the product documentation.

Additional resources

- Red Hat Single Sign-On Supported Configurations
- Red Hat 3scale API Management Supported Configurations
- Red Hat Debezium Supported Configurations
- Red Hat Service Registry Supported Configurations
CHAPTER 8. IMPORTANT LINKS

- AMQ Streams Supported Configurations
- AMQ Streams Component Details

Revised on 2022-04-13 17:49:45 UTC