Red Hat AMQ 7.5

Release Notes for AMQ Streams 1.3 on OpenShift

For use with AMQ Streams on OpenShift Container Platform

Last Updated: 2022-06-29

Red Hat AMQ 7.5 Release Notes for AMQ Streams 1.3 on OpenShift

For use with AMQ Streams on OpenShift Container Platform

Legal Notice

Abstract

These release notes contain the latest information about new features, enhancements, fixes, and issues contained in the AMQ Streams 1.3 release.

Table of Contents

CHAPTER 1. FEATURES

The features added in this release, and that were not in previous releases of AMQ Streams, are outlined below.

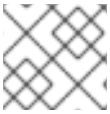
1.1. KAFKA 2.3.0 SUPPORT

AMQ Streams now supports Apache Kafka version 2.3.0.

AMQ Streams is based on Kafka 2.3.0. Only Kafka distributions built by Red Hat are supported.

You must upgrade the Cluster Operator to AMQ Streams version 1.3 before you can upgrade brokers and client applications to Kafka 2.3.0. For instructions, see AMQ Streams and Kafka upgrades .

Refer to the Kafka 2.2.1 and Kafka 2.3.0 Release Notes for additional information.



NOTE

Kafka 2.2.x is supported in AMQ Streams only for upgrade purposes.

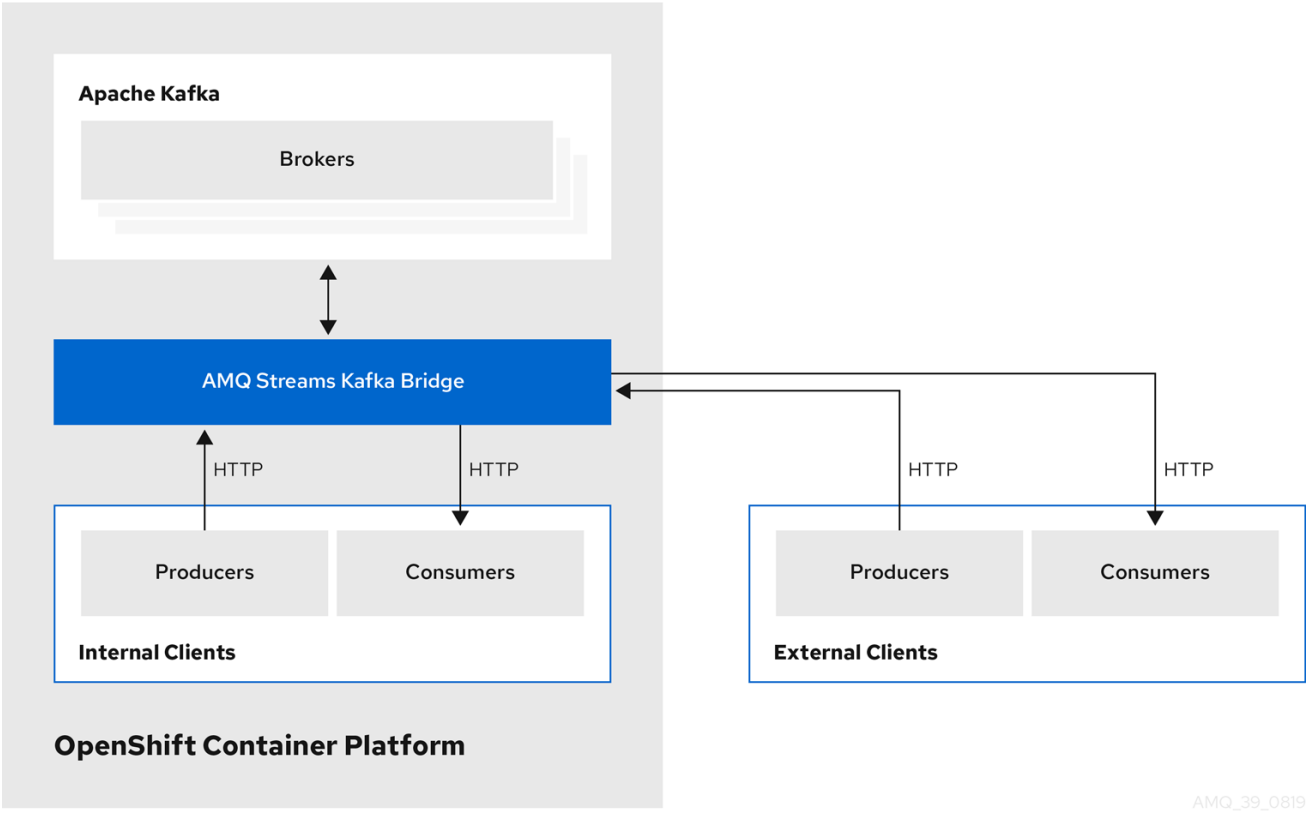
For more information on supported versions, see the Red Hat AMQ 7 Component Details Page on the Customer Portal.

1.2. AMQ STREAMS KAFKA BRIDGE

The AMQ Streams Kafka Bridge moves from a Technology Preview to a generally available component of AMQ Streams.

1.2.1. Overview

The Kafka Bridge provides a RESTful interface to AMQ Streams, offering the advantages of a web API that is easy to use and connect with AMQ Streams without the need for client applications to interpret the Kafka protocol.



The API has two main resources – **consumers** and **topics** – that are exposed through endpoints that allow you to interact with consumers and producers in your Kafka cluster. The resources relate only to the Kafka Bridge, not the consumers and producers connected directly to Kafka.

The Kafka Bridge supports HTTP requests to:

- Send messages to topics.
- Retrieve messages from topics.
- Create and delete consumers.
- Subscribe consumers to topics, so that they start receiving messages from those topics.
- Retrieve a list of topics that a consumer is subscribed to.
- Unsubscribe consumers from topics.
- Assign partitions to consumers.
- Commit consumer offsets.
- Perform seek operations on a partition.

The methods provide JSON responses and HTTP response code error handling. Messages can be sent in JSON or binary formats.

See Kafka Bridge overview and Kafka Bridge configuration. For API documentation, see the Kafka Bridge API reference.

To try out the API from your local machine, see Kafka Bridge quickstart.

1.2.2. API changes since the Technology Preview

The following changes have been made to the Kafka Bridge API since the Technology Preview release:

- The **openapi** endpoint has been added. This retrieves the OpenAPI 2.0 specification for the Kafka Bridge in JSON format.
- The **subscription** endpoint now supports **GET** requests to retrieve a list of all topics that the consumer is subscribed to.

See the openapi and subscription endpoints in the Kafka Bridge API reference.

Also, there is a **breaking change** to the data types declared in the OpenAPI Specification (OAS) for the following consumer configuration settings:

Consumer configuration setting	Previously declared as...	Now declared as...
consumer.request.timeout.ms	String	Integer
enable.auto.commit	String	Boolean
fetch.min.bytes	String	Integer

The data types for these settings are now validated by the OAS. As a result, if invalid data types are submitted to the /**consumers** endpoint, the OAS returns a **400 Bad Request** code (containing details of the invalid settings). Previously, a general **500 Internal Server Error** code was returned.

1.2.3. 3scale integration with the Kafka Bridge

You can now integrate Red Hat 3scale API Management with the Kafka Bridge.

3scale can secure the Kafka Bridge with TLS, and provide authentication and authorization. Integration with 3scale also means that additional features like metrics, rate limiting and billing are now available.

See Using the Kafka Bridge with 3scale .

1.2.4. 3scale service discovery annotations and labels

When the Kafka Bridge is deployed, the service that exposes the REST interface of the Kafka Bridge has the annotations and labels required for discovery by 3scale.

See Kafka Bridge service discovery .

1.2.5. Kafka Bridge quickstart

A new quickstart guide for the Kafka Bridge helps you to try out the API from your local machine. The quickstart provides example curl requests for the most common methods in the API.

See Kafka Bridge quickstart .

1.3. STATUS PROPERTIES FOR CUSTOM RESOURCES

Status properties for AMQ Streams custom resources moves from a Technology Preview to a generally available feature.

You can check the current status of a custom resource by querying its **status** property. A resource's **status** publishes information about the resource to users and tools that need the information. The current state and last **observedGeneration** are available for every resource. Some resources also publish resource-specific information.

Status information is available for the following resources:

- **Kafka**
- **KafkaTopic**
- **KafkaConnect**
- **KafkaConnectS2I**
- **KafkaMirrorMaker**
- **KafkaUser**
- **KafkaBridge**

To check the status of a resource, use the **oc get** command and apply a JSONPath expression, for example:

oc get kafka KafkaTopic -o jsonpath='{.status}'

oc get kafka KafkaBridge -o jsonpath='{.status.observedGeneration}'

See AMQ Streams custom resource status and Checking the status of a custom resource.

1.4. KAFKA EXPORTER SUPPORT

Kafka Exporter is an open source project to enhance monitoring of Apache Kafka brokers and clients. Kafka Exporter is provided with AMQ Streams for deployment with a Kafka cluster to extract additional metrics data from Kafka brokers related to offsets, consumer groups, consumer lag, and topics.

The metrics data is used, for example, to help identify slow consumers.

Lag data is exposed as Prometheus metrics, which can then be presented in Grafana for analysis.

If you are already using Prometheus and Grafana for monitoring of built-in Kafka metrics, you can configure Prometheus to also scrape the Kafka Exporter Prometheus endpoint.

See Kafka Exporter.

CHAPTER 2. ENHANCEMENTS

The enhancements added in this release are outlined below.

2.1. KAFKA 2.3.0 ENHANCEMENTS

For an overview of the enhancements introduced with Kafka 2.3.0, refer to the Kafka 2.3.0 Release Notes.

2.2. METRICS

The example metrics configuration provided with AMQ Streams has been updated. You can find the example metrics configuration files in the installation ZIP file available for this release from the AMQ Streams download site.

You can set up for monitoring using Prometheus to provide monitoring data for Grafana dashboards. The example configuration is supplied for:

- Adding metrics configuration to your Kafka cluster
- Deploying Prometheus and the Prometheus Alertmanager plugin
- Deploying Grafana

See Introducing Metrics.

2.3. MIRROR MAKER HEALTHCHECKS

You can now configure liveness and readiness probes for Mirror Maker resources. These Healthchecks allow you to check the status of Mirror Maker replicas in your AMQ Streams deployment.

See Kafka Mirror Maker configuration and Healthcheck configurations.

2.4. CUSTOM ENVIRONMENT VARIABLES

You can now set custom environment variables for Kafka broker, Mirror Maker, and Kafka Connect containers by setting template properties in the resource. For example, you can use the **kafkaContainer** template property in **Kafka.spec.kafka**:

```
# ...
spec:
  kafka:
    template:
      kafkaContainer:
        env:
          - name: TEST_ENV_1
            value: test.env.one
          - name: TEST_ENV_2
            value: test.env.two
# ...
```

You set custom environment variables when configuring distributed tracing, which is a Technology Preview feature in this release.

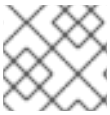
See Customizing containers with environment variables.

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. DISTRIBUTED TRACING WITH JAEGER



NOTE

This is a Technology Preview feature.

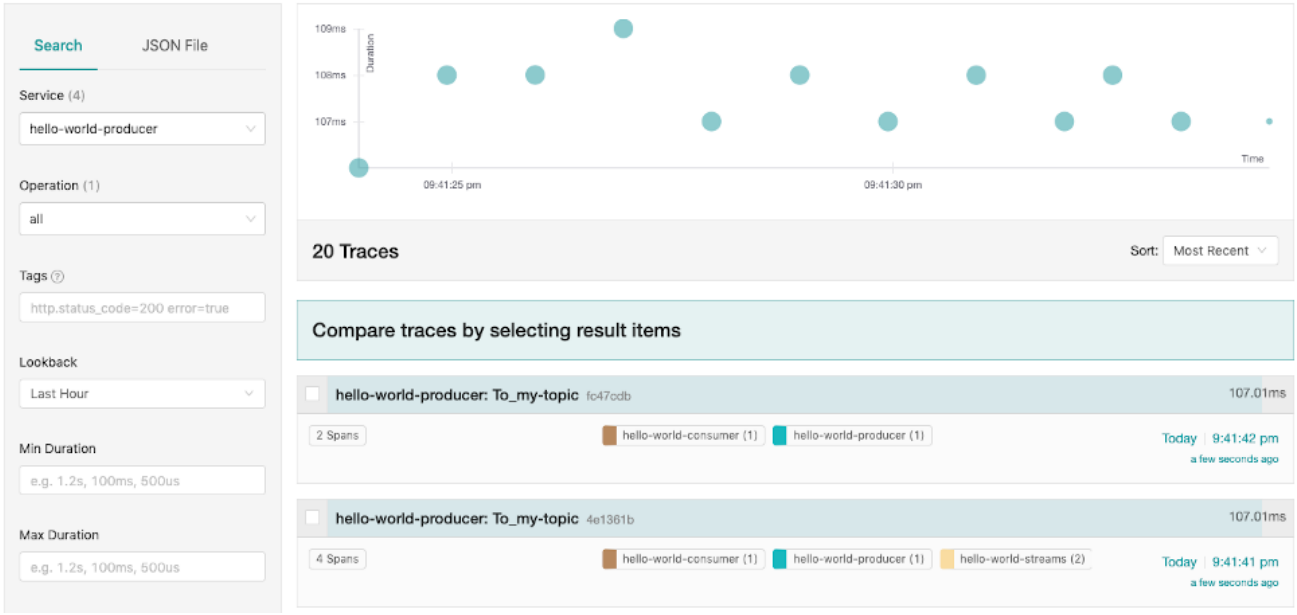
This release adds support for the distributed tracing of transactions within a typical Kafka architecture. Using an included OpenTracing Java library, you can instrument your client applications to generate traces for transactions, for example, producing and consuming messages.

Distributed tracing is supported in the following components:

- Kafka clusters
- Producers and consumers
- Kafka Streams applications
- Mirror Maker
- Kafka Connect

Trace data is visualized in a user interface using Jaeger. You can use this information to monitor the operation of your Kafka cluster from end-to-end, and debug performance issues with target systems and applications.

An example of a query in the Jaeger user interface



See Distributed tracing.

3.2. OAUTH 2.0 AUTHENTICATION

NOTE

This is a Technology Preview feature.

AMQ Streams supports the use of OAuth 2.0 authentication using the *SASL OAUTHBEARER* mechanism.

Using OAuth 2.0 token based authentication, application clients can access resources on application servers (called 'resource servers') without exposing account credentials. The client presents an access token as a means of authenticating, which application servers can also use to find more information about the level of access granted. The authorization server handles the granting of access and inquiries about access.

In the context of AMQ Streams:

- Kafka brokers act as resource servers
- Kafka clients act as resource clients

The brokers and clients communicate with the OAuth 2.0 authorization server, as necessary, to obtain or validate access tokens.

For a deployment of AMQ Streams, OAuth 2.0 integration provides:

- Server-side OAuth 2.0 support for Kafka brokers
- Client-side OAuth 2.0 support for Kafka Mirror Maker, Kafka Connect and the Kafka Bridge

Red Hat Single Sign-On integration

You can deploy Red Hat Single Sign-On as an authorization server and configure it for integration with AMQ Streams.

You can use Red Hat Single Sign-On to:

- Configure authentication for Kafka brokers
- Configure and authorize clients
- Configure users and roles
- Obtain access and refresh tokens

See Using OAuth 2.0 token based authentication .

3.3. CHANGE DATA CAPTURE

NOTE

This is a Technology Preview feature.

Red Hat Change Data Capture is a distributed platform that monitors databases and creates change event streams. Change Data Capture is built on Apache Kafka and can be deployed and integrated with AMQ Streams.

Following a deployment of AMQ Streams, you deploy Change Data Capture as a connector configuration through Kafka Connect.

Change Data Capture captures row-level changes to a database table and passes corresponding change events to AMQ Streams on OpenShift. Applications can read these *change event streams* and access the change events in the order in which they occurred.

Change Data Capture has multiple uses, including:

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

Change Data Capture provides connectors (based on Kafka Connect) for the following common databases:

- MySQL
- PostgreSQL
- SQL Server
- MongoDB

Debezium is the open source project for Change Data Capture.

For more information on deploying Change Data Capture with AMQ Streams, see Deploying the Debezium 1.0 Technology Preview with AMQ Streams.

CHAPTER 4. DEPRECATED FEATURES

There are no deprecated features for AMQ Streams 1.3.

CHAPTER 5. FIXED ISSUES

The following table lists the issues fixed in AMQ Streams 1.3.

Issue Number	Description
ENTMQST-1287	The operation to create consumer should not contain HTTP code 500 in response
ENTMQST-1236	Missing the dashboard for Kafka Connect S2I
ENTMQST-1216	[Labels] - Modification of labels
ENTMQST-1194	Unhandled exception when decoding invalid JSON
ENTMQST-1158	NPE when deploying a Kafka cluster with an unsupported version of Kafka
ENTMQST-1138	Fix null check in EU/UO role binding management
ENTMQST-1137	Fix deprecated Topic Operator to work with new images
ENTMQST-1100	Topic Operator should handle deletion of Custom Resource when **delete.topic.enable=false**

CHAPTER 6. KNOWN ISSUES

The following table lists the known issues for AMQ Streams 1.3.

Issue Number	Description

ENTMQST-1402	When accessing status information for the AMQ Streams Kafka Bridge resource, the URL for external client access to the Kafka Bridge service is incorrect. The **-bridge** part of the URL is duplicated. For example:
	
	`#...`
	`status:`
	`  conditions:`
	`  - lastTransitionTime: "2019-10-15T14:21:40.520Z"`
	`    status: "True"`
	`    type: Ready`
	`  observedGeneration: 1`
	`  url: http://my-bridge-bridge-bridge-service.myproject.svc:8080`
	
	The correct URL in the above example is: **http://my-bridge-bridge-service.myproject.svc:8080**.

CHAPTER 7. SUPPORTED INTEGRATION PRODUCTS

AMQ Streams 1.3 supports integration with the following Red Hat products.

- **Red Hat Single Sign-On 7.3.0** for OAuth 2.0 support (as a Technology Preview)
- **Red Hat 3scale API Management 2.6** to secure the Kafka Bridge and provide additional API management features

For information on the functionality these products can introduce to your AMQ Streams deployment, refer to the AMQ Streams 1.3 documentation.

CHAPTER 8. IMPORTANT LINKS

- Red Hat AMQ 7 Supported Configurations
- Red Hat AMQ 7 Component Details

Revised on 2022-06-29 08:45:59 UTC