Red Hat AMQ 2021.q2

Release Notes for AMQ Streams 1.7 on RHEL

For use with AMQ Streams on Red Hat Enterprise Linux

Last Updated: 2021-05-17

Red Hat AMQ 2021.q2 Release Notes for AMQ Streams 1.7 on RHEL

For use with AMQ Streams on Red Hat Enterprise Linux

Legal Notice

Abstract

These release notes contain the latest information about new features, enhancements, fixes, and issues contained in the AMQ Streams 1.7 release.

Table of Contents

MAKING OPEN SOURCE MORE INCLUSIVE

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

CHAPTER 1. FEATURES

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

NOTE

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

1.1. KAFKA 2.7.0 SUPPORT

AMQ Streams now supports Apache Kafka version 2.7.0.

AMQ Streams uses Kafka 2.7.0. Only Kafka distributions built by Red Hat are supported.

For upgrade instructions, see AMQ Streams and Kafka upgrades .

Refer to the Kafka 2.6.0 and Kafka 2.7.0 Release Notes for additional information.

NOTE

Kafka 2.6.x is supported only for the purpose of upgrading to AMQ Streams 1.7.

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

Kafka 2.7.0 requires the same ZooKeeper version as Kafka 2.6.x (ZooKeeper version 3.5.8). Therefore, you do *not* need to upgrade ZooKeeper when upgrading from AMQ Streams 1.6 to AMQ Streams 1.7.

CHAPTER 2. ENHANCEMENTS

The enhancements added in this release are outlined below.

2.1. KAFKA 2.7.0 ENHANCEMENTS

For an overview of the enhancements introduced with Kafka 2.7.0, refer to the Kafka 2.7.0 Release Notes.

2.2. OAUTH 2.0 AUTHENTICATION AND AUTHORIZATION

This release includes the following enhancements to OAuth 2.0 token-based authentication and authorization.

Checks on JWT access tokens

You can now configure two additional checks on JWT access tokens. Both of these checks are configured in the OAuth 2.0 authentication listener configuration.

Custom claim checks

Custom claim checks impose custom rules on the validation of JWT access tokens by Kafka brokers. They are defined using JsonPath filter queries.

If an access token does not contain the necessary data, it is rejected. When using *introspection endpoint* token validation, the custom check is applied to the introspection endpoint response JSON.

To configure custom claim checks, add the **oauth.custom.claim.check** option to the **server.properties** file and define a JsonPath filter query. Custom claim checks are disabled by default.

See Configuring OAuth 2.0 support for Kafka brokers

Audience checks

Your authorization server might provide **aud** (audience) claims in JWT access tokens.

When audience checks are enabled, the Kafka broker rejects tokens that do not contain the broker's **clientId** in their **aud** claims.

To enable audience checks, set the **oauth.check.audience** option to **true**. Audience checks are disabled by default.

See Configuring OAuth 2.0 support for Kafka brokers

Support for OAuth 2.0 over SASL PLAIN authentication

You can now configure the PLAIN mechanism for OAuth 2.0 authentication between Kafka clients and Kafka brokers. Previously, the only supported authentication mechanism was OAUTHBEARER.

PLAIN is a simple authentication mechanism supported by all Kafka client tools (including developer tools such as kafkacat). AMQ Streams includes server-side callbacks that enable PLAIN to be used with OAuth 2.0 authentication. These capabilities are referred to as *OAuth 2.0 over PLAIN*.

NOTE

Red Hat recommends using OAUTHBEARER authentication for clients whenever possible. OAUTHBEARER provides a higher level of security than PLAIN because client credentials are *never* shared with Kafka brokers. Consider using PLAIN only with Kafka clients that do not support OAUTHBEARER.

When used with the provided *OAuth 2.0 over PLAIN* callbacks, Kafka clients can authenticate with Kafka brokers using either of the following methods:

- Client ID and secret (by using the OAuth 2.0 client credentials mechanism)
- A long-lived access token, obtained manually at configuration time

To use PLAIN, you must enable it in the **server.properties** file, in the OAuth authentication listener configuration.

See OAuth 2.0 authentication mechanisms and Configuring OAuth 2.0 support for Kafka brokers

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. CRUISE CONTROL FOR CLUSTER REBALANCING

NOTE

Cruise Control remains in Technology Preview, with some new enhancements.

You can install 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 helps to reduce the time and effort involved in running an efficient and balanced Kafka cluster.

A zipped distribution of Cruise Control is available for download from the Customer Portal. To install Cruise Control, you configure each Kafka broker to use the provided Metrics Reporter. Then, you set Cruise Control properties, including optimization goals, and start Cruise Control using the provided script.

The Cruise Control server is hosted on a single machine for the whole Kafka cluster.

When Cruise Control is running, you can use the REST API to:

- Generate *dry run* optimization proposals from multiple optimization goals
- Initiate an optimization proposal to rebalance the Kafka cluster

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.

3.1.1. Enhancements to the Technology Preview

The following enhancements have been added to the Technology Preview of Cruise Control for cluster rebalancing.

New goal: Minimum topic leaders per broker

You can use a new goal named **MinTopicLeadersPerBrokerGoal**.

For each topic in a defined group of topics, the goal ensures that each active broker has at least a certain number of leader replicas.

MinTopicLeadersPerBrokerGoal is a default goal and is preset as a hard goal.

See Optimization goals overview

Logging enhancement: Log4j 2

Log4j 2 is now used for Cruise Control logging.

You **must** update existing configurations for Cruise Control logging (in **/opt/cruise-control/config/log4j.properties**) from **Log4j** to **Log4j 2** compatible syntax.

See Cruise Control configuration

CHAPTER 4. DEPRECATED FEATURES

There are no deprecated features for AMQ Streams 1.7.

CHAPTER 5. FIXED ISSUES

The issues fixed in AMQ Streams 1.7 on RHEL are shown in the following table. For details of the issues fixed in Kafka 2.7.0, refer to the Kafka 2.7.0 Release Notes .

Issue Number	Description
ENTMQST-2030	If the **bin/kafka-acls.sh** utility is used to add or remove an ACL, the operation is successful but a warning is generated
ENTMQST-2188	MirrorMaker: Enable synchronization of offsets to the consumer group on the target cluster
ENTMQST-2269	kafka-configs.sh is deprecating **--zookeeper** option but does not provide alternative functionality to list configs for users

CHAPTER 6. KNOWN ISSUES

There are no known issues for AMQ Streams 1.7 on RHEL.

CHAPTER 7. SUPPORTED INTEGRATION PRODUCTS

AMQ Streams 1.7 supports integration with the following Red Hat products.

- **Red Hat Single Sign-On 7.4 and later** for OAuth 2.0 authentication and OAuth 2.0 authorization
- **Red Hat Debezium 1.4 and later** for monitoring databases and creating event streams

For information on the functionality these products can introduce to your AMQ Streams deployment, refer to the AMQ Streams 1.7 documentation.

CHAPTER 8. IMPORTANT LINKS

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

Revised on 2021-05-17 14:47:27 UTC