Red Hat build of Quarkus 1.3

Release Notes for Red Hat build of Quarkus 1.3

Last Updated: 2020-12-08

Red Hat build of Quarkus 1.3 Release Notes for Red Hat build of Quarkus 1.3

Legal Notice

Abstract

This document contains release notes for Red Hat build of Quarkus 1.3.

Table of Contents

PREFACE

These release notes list new features, features in technology preview, known issues, and issues fixed in Red Hat build of Quarkus 1.3.

CHAPTER 1. RED HAT BUILD OF QUARKUS

Red Hat build of Quarkus is a Kubernetes-native Java stack that is optimized for use with containers and Red Hat OpenShift Container Platform. Quarkus is designed to work with popular Java standards, frameworks, and libraries such as Eclipse MicroProfile, Apache Kafka, RESTEasy (JAX-RS), Hibernate ORM (JPA), Spring, Infinispan, and Apache Camel.

The Quarkus dependency injection solution is based on CDI (contexts and dependency injection) and includes an extension framework to expand functionality and to configure, boot, and integrate a framework into your application.

Quarkus provides a container-first approach to building Java applications. This approach makes it much easier to build microservices-based applications written in Java as well as enabling those applications to invoke functions running on serverless computing frameworks. For this reason, Quarkus applications have small memory footprints and fast start-up times.

CHAPTER 2. RED HAT BUILD OF QUARKUS SUPPORTED PLATFORMS, CONFIGURATIONS, EXTENSIONS, AND DEPENDENCIES

This section lists supported environments, configurations, extensions, and dependencies in Red Hat build of Quarkus.

2.1. TESTED AND VERIFIED ENVIRONMENTS

Red Hat build of Quarkus is supported on the following platforms:

- Red Hat Enterprise Linux 8
- Red Hat OpenShift Container Platform 3.11 on x86_64
- Red Hat OpenShift Container Platform 4.3 on x86_64
- Red Hat OpenShift Container Platform 4.3 on IBM Z (supported with Quarkus 1.3.4)

NOTE

Only the database client is tested and verified, not the server.

2.2. SUPPORTED CONFIGURATIONS

- For a list of supported configurations, see the Red Hat build of Quarkus Supported Configurations page (login required).
- For a list of supported Maven artifacts see the Red Hat build of Quarkus Component Details page (login required).

2.3. SUPPORTED EXTENSIONS AND DEPENDENCIES

Red Hat provides production support for the following Red Hat build of Quarkus extensions and dependencies:

- **quarkus-agroal**
- **quarkus-config-yaml**
- **quarkus-core**
- **quarkus-hibernate-orm**
- **quarkus-hibernate-orm-panache**
- **quarkus-hibernate-validator**
- **quarkus-jackson**
- **quarkus-jaxb**
- **quarkus-jdbc-mariadb**

- **quarkus-jdbc-mssql**
- **quarkus-jdbc-mysql**
- **quarkus-jdbc-postgresql**
- **quarkus-jsonb**
- **quarkus-jsonp**
- **quarkus-kafka-client**
- **quarkus-logging-json**
- **quarkus-narayana-jta**
- **quarkus-oidc**
- **quarkus-quartz**
- **quarkus-reactive-pg-client**
- **quarkus-rest-client**
- **quarkus-resteasy**
- **quarkus-resteasy-jackson**
- **quarkus-resteasy-jaxb**
- **quarkus-resteasy-jsonb**
- **quarkus-scheduler**
- **quarkus-smallrye-context-propagation**
- **quarkus-smallrye-fault-tolerance**
- **quarkus-smallrye-health**
- **quarkus-smallrye-jwt**
- **quarkus-smallrye-metrics**
- **quarkus-smallrye-openapi**
- **quarkus-smallrye-opentracing**
- **quarkus-smallrye-reactive-messaging**
- **quarkus-smallrye-reactive-messaging-amqp**
- **quarkus-smallrye-reactive-messaging-kafka**
- **quarkus-smallrye-reactive-streams-operators**
- **quarkus-spring-boot-properties**

- **quarkus-spring-data-jpa**
- **quarkus-spring-di**
- **quarkus-spring-security**
- **quarkus-spring-web**
- **quarkus-undertow**
- **quarkus-undertow-websockets**
- **quarkus-vertx**
- **quarkus-vertx-web**

2.4. DEVELOPMENT SUPPORT

Red Hat provides development support for the following Red Hat build of Quarkus features, plug-ins, extensions, and dependencies:

Features

- Live development mode
- Debugging

Plug-ins

- Quarkus Maven plug-in (**quarkus-maven-plugin)**
- Maven Surefire plug-in (**maven-surefire-plugin**)

Extensions and dependencies

- **quarkus-jdbc-derby**
- **quarkus-jdbc-h2**
- **quarkus-openshift**
- **quarkus-junit5**
- **rest-assured**

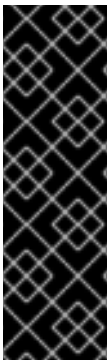
CHAPTER 3. DEPRECATED COMPONENTS AND FEATURES

The components and features listed in this section are deprecated with Red Hat build of Quarkus 1.3. They are included and supported in this release, however no enhancements will be made to these components and features and they might be removed in the future.

- The **quarkus-smallrye-opentracing** extension
- The use of ReactiveX APIs
- The **reactive-streams-operators** API

CHAPTER 4. TECHNOLOGY PREVIEW

This section lists features and extensions that are in Technology Preview in Red Hat build of Quarkus 1.3.



IMPORTANT

These features are for Technology Preview only. Technology Preview features are not supported with Red Hat production service level agreements (SLAs), might not be functionally complete, and Red Hat does not recommend to use them for production. These features provide early access to upcoming product features, enabling customers to test functionality and provide feedback during the development process.

For more information on Red Hat Technology Preview features, see Technology Preview Features Scope.

Features

- Native compilation using GraalVM
- Remote development

Extensions and dependencies

- **quarkus-mutiny**
- **quarkus-resteasy-mutiny**
- **quarkus-keycloak-authorization**

CHAPTER 5. KNOWN ISSUES

This section lists known issues with Red Hat build of Quarkus 1.3.

- If you add the **smallrye-reactive-messaging** extension, a warning message about ReactiveX appears when you close Quarkus even if you are not using Reactive APIs in your code. For more information, see Red Hat article 4954651.

- The **quarkus-container-image-s2** extension, which is typically used indirectly through the **quarkus-openshift** extension, treats the Red Hat OpenJ9 images recommended for s390x architecture as if they do not include the **run-java.sh** script. This affects the generated **openshift.yml** file.
  If the image uses the **run-java.sh** script, the container definition only includes several environment variables. It does not include a command that should be executed in the container. Instead, the default S2I **run** script is used which executes the **run-java.sh** script.

  If the image does not use the **run-java.sh** script, the container definition includes the **java** command that should be executed in the container.

  One difference between these scenarios is that the **run-java.sh** script changes the current working directory to the directory where the application JAR file is located, typically **/deployments**. This is important when mounting a ConfigMap with the **application.properties** configuration file into the container filesystem. With the **run-java.sh** script, the ConfigMap typically must be mounted into the **/deployments/config** directory, while with the other images, the ConfigMap must be mounted into the default working directory defined by the image.

  NOTE

  This issue does not affect all applications that use the Red Hat OpenJ9 images recommended for s390x architecture.

  ### Cause

  The **quarkus-container-image-s2i** extension has a list of known images that contain the **run-java.sh** script. All other images are treated as if they do not contain the **run-java.sh** script. This issue will be fixed in a future release.

  ### Workaround

  - If your application is affected by the current working directory issue, you can define the current working directory of the container in the pod definition. This overrides the default working directory defined in the image. If you use the **quarkus-openshift** extension to generate the **openshift.yml** file, use the following configuration property:

    ```
    quarkus.openshift.working-dir=/deployments
    ```

  - If your application is affected by the current working directory issue specifically when mounting a ConfigMap with the **application.properties** configuration file, another option is to mount the ConfigMap into a specific directory. If you use the **quarkus-openshift** extension, use following configuration property, assuming that the default working directory defined by the image is **/home/jboss**:

    ```
    quarkus.openshift.mounts.<mount name>.path=/home/jboss/config
    ```

CHAPTER 6. FIXED ISSUES IN QUARKUS 1.3.4

Quarkus 1.3.4 provides increased stability and fixed issues listed in this section.

6.1. MAJOR CHANGES

- Performance improvements on multi-core systems:
  Several scalability issues were identified and fixed in JBoss Threads. JBoss Threads has been upgraded to version 3.1.1.Final which significantly improves scalability on multi-core systems.
- Performance improvements related to suboptimal bean scopes:
  - Add the singleton scope to beans produced by the **JsonbProducer** class.
  - Fix a Wildfly Elytron security framework performance issue due to missing bean scope .
- Upgraded to Quarkus HTTP 3.0.7.Final which fixes several issues related to websockets and HTTP/2:
  - Maximum frame size fixed.
  - Set host and port on the **SSLEngine** class.
  - Use the correct port.
  - Fixed potential security issue.
- The environment variables from secrets were generated in such a way that Quarkus was incompatible with OpenShift 3.11.
- The OpenID Connect extension was not removing cookies if the cookie path was set . This issue was fixed here.
- Several fixes were made to clustering support for the Quartz extension:
  - Cron executions could be duplicated in a clustered Quartz environment .
  - Transactions were not properly handled when storing the Quartz jobs in a database .

6.2. MINOR CHANGES

- The PostgreSQL JDBC driver was upgraded to 42.2.12 because several regressions were spotted in 42.2.11.
- The health probes were missing from the descriptor generated by the container image extension if a container image name was specified.
- In certain cases, when a configuration error occurred at bootstrap, the exception was swallowed which made issues difficult to diagnose. The error is now properly logged in all cases.
- The Spring Data compatibility extension now supports primitive types .
- Updated SnakeYAML to 1.26 to fix CVE (security issue) .

6.3. FIXED ISSUES IN QUARKUS 1.3.4 SP1

The Quarkus 1.3.4 SP1 release contains the following bug fixes.

- For Quarkus 1.3.4 SP1, the PostgreSQL JDBC driver (aka PgJDBC) before version 42.2.13 allows XXE.
- For Quarkus 1.3.4 SP1, a vulnerability was fixed in RESTEasy, where RootNode incorrectly caches routes.

Revised on 2020-12-08 19:29:33 UTC