Red Hat build of Quarkus 3.8

红帽构建的 Quarkus 3.8 发行注记

Last Updated: 2024-06-21

法律通告

摘要

发行注记提供有关新功能、重要的技术更改、技术预览、错误修复、已知问题和相关公告的信息。

目录

提供有关红帽构建的 QUARKUS 文档的反馈

要报告错误或改进文档，请登录到 Red Hat JIRA 帐户并提交问题。如果您没有 Red Hat Jira 帐户，则会提示您创建一个帐户。

流程

1. 单击以下链接 **以创建 ticket**。
2. 在 **Summary** 中输入问题的简短描述。
3. 在 **Description** 中提供问题或功能增强的详细描述。包括一个指向文档中问题的 URL。
4. 点 **Submit** 创建问题，并将问题路由到适当的文档团队。

使开源包含更多

红帽致力于替换我们的代码、文档和 Web 属性中有问题的语言。我们从这四个术语开始：master、slave、黑名单和白名单。由于此项工作十分艰巨，这些更改将在即将推出的几个发行版本中逐步实施。详情请查看 CTO Chris Wright 的信息。

第 1 章 RED HAT BUILD OF QUARKUS 3.8 发行注记

发行注记提供有关红帽构建的 Quarkus 3.8 的新功能、重要的技术更改、技术预览功能、错误修复、已知问题以及相关公告的信息。

另外还提供了有关升级和向后兼容性的信息，可帮助您从早期版本中过渡。

1.1. 关于红帽构建的 QUARKUS

Red Hat build of Quarkus 是一个 Kubernetes 原生 Java 堆栈，针对容器和 Red Hat OpenShift Container Platform 进行了优化。Quarkus 设计为使用流行的 Java 标准、框架和库，如 Eclipse MicroProfile、Eclipse Vert.x、Apache Camel、Apache Kafka、Hibernate ORM 和 RESTEasy Reactive (Jakarta REST)。

作为开发人员，您可以选择 Java 应用所需的 Java 框架，您可以在 Java 虚拟机(JVM)模式下运行，或者以原生模式运行。Quarkus 提供了构建 Java 应用程序的容器优先方法。容器先行方法促进微服务和功能的容器化和高效执行。因此，Quarkus 应用程序具有较小的内存空间和更快的启动时间。

Quarkus 还通过统一配置、自动配置未配置的服务、实时编码和持续测试等功能优化应用程序开发流程，为您提供对代码更改的即时反馈。

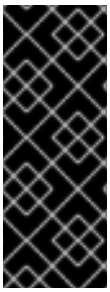
1.2. QUARKUS 社区版本和红帽构建的 QUARKUS 之间的区别

作为应用程序开发人员，您可以访问两个不同的 Quarkus 版本： Quarkus 社区版本和产品化版本，Red Hat build of Quarkus。

下表描述了 Quarkus 社区版本和红帽构建的 Quarkus 之间的区别。

功能	Quarkus 社区版本	Red Hat build of Quarkus 版本	描述
访问最新的社区功能	是	否	使用 Quarkus 社区版本，您可以访问最新的功能开发。 红帽没有发布红帽构建的 Quarkus，以与社区版本的每个版本对应。红帽构建的 Quarkus 功能版本的节奏大约每 6 个月。
来自红帽的企业支持	否	是	红帽只为红帽构建的 Quarkus 提供企业支持。要报告 Quarkus 社区版本的问题，请参阅 quarkusio/quarkus - issues。
使用 Red Hat OpenShift Container Platform 和 Red Hat Enterprise Linux (RHEL)测试并验证	否	是	Red Hat build of Quarkus 使用 Red Hat OpenShift Container Platform 和 RHEL 构建、测试并验证。红帽根据您的订阅协议为支持的配置和经过测试的集成提供生产和开发支持。如需更多信息，请参阅 Red Hat build of Quarkus 支持的配置。

功能	Quarkus 社区版本	Red Hat build of Quarkus 版本	描述
使用安全构建系统从源构建	否	是	在红帽构建的 Quarkus 中，红帽使用安全软件交付提供核心平台和所有支持的扩展，这意味着它们从源构建、扫描以了解安全问题，以及验证的许可证使用情况。
访问 JDK 和红帽构建的 Quarkus 原生构建程序分发	否	是	Red Hat build of Quarkus 支持经过认证的 OpenJDK 构建和经认证的原生可执行构建器。请参阅以下接受。如需更多信息，请参阅 支持的配置。



重要

Red Hat build of Quarkus 支持使用 红帽构建的 Quarkus 原生构建器镜像来构建原生 Linux 可执行文件，该镜像基于 Mandrel 并由红帽分发。

如需更多信息，请参阅将 Quarkus 应用程序编译到原生可执行文件。使用红帽构建的 Quarkus 不支持使用 Oracle GraalVM 社区版(CE)、Mael 社区版本或任何其他 GraalVM 发行版构建原生可执行文件。

1.3. 新功能、增强功能和技术变化

本节概述红帽构建的 Quarkus 3.8 中引入的新功能、增强功能和技术变化。

1.3.1. Core

1.3.1.1. 支持 Java 21

Java 21 现在是推荐的版本，但支持 Java 17。

1.3.1.2. 删除了对 Java 11 的支持

在这个 3.8 发行版本中，删除了对版本 3.2 中弃用的 Java 11 的支持。

1.3.1.3. Red Hat build of Quarkus 添加了对虚拟线程的支持

使用带有虚拟线程(VT)的 Red Hat build of Quarkus 会有以下几项：

- 增强并发任务的管理，提高可扩展性和资源效率。
- 通过提高资源效率提高虚拟线程的资源效率，这降低了块。
- 简化并发模式，简化代码库的维护。
- 减少线程上下文切换开销，从而降低延迟和更高的吞吐量。
- 启用更好的多核处理器利用率，允许更多的并发任务，而无需大量上下文切换。

注意

虚拟线程仅在 Java 21 JVM 上被支持。如需更多信息，请参阅 Oracle *Java Core Libraries 开发者指南中的* 虚拟线程 部分，以及 OpenJDK 的 JEP 444: Virtual Threads 。

虚拟线程限制：

- 固定载体线程的库可能会延迟采用，直到 Java 生态系统完全采用虚拟线程兼容性。
- 冗长的计算需要仔细分析，以防止对资源进行大量处理。
- 载体线程池的弹性可能会导致内存消耗增加。
- thread-local 对象轮询模式可能会影响分配和内存用量。
- 虚拟线程本身并不会解决线程安全问题，需要减少管理。

1.3.2. data

1.3.2.1. Hibernate ORM 升级到 6.4

在红帽构建的 Quarkus 3.8 中，Hibernate Object-Relational Mapping (ORM)升级到 6.4 版本。

如需更多信息，请参阅以下资源：

- 影响与早期版本兼容性的更改
- Hibernate ORM 文档 6.4
- 使用 Hibernate ORM 和 Jakarta Persistence 指南进行 Quarkus

1.3.2.2. Hibernate Reactive 操作以及 Agroal

在红帽构建的 Quarkus 3.8 中，Hibernate Reactive 可以与 Agroal 共存，这意味着您可以在应用程序中使用 Flyway 或 Liquibase，同时将 Hibernate Reactive 用作对象-Relational Mapping (ORM)。

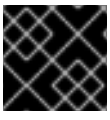
注意

红帽构建的 Quarkus 中存在一个限制，其中您不能在同一应用程序中同时存在 Hibernate ORM 和 Hibernate Reactive。

如需更多信息，请参阅使用 Hibernate Reactive 的 Quarkus 指南。

1.3.2.3. Hibernate Reactive 升级到 2.2 版本

在 Red Hat build of Quarkus 3.8 中，Hibernate Reactive 扩展升级到版本 2.2，它与 Hibernate ORM 6.4.0 兼容。



重要

Hibernate Reactive 扩展在红帽构建的 Quarkus 3.8 中作为技术预览提供。

如需更多信息，请参阅 Hibernate Reactive 2.2.0 文档。

1.3.2.4. Hibernate Search 升级到 7.0 版本

在 Red Hat build of Quarkus 3.8 中，Hibernate Search 被升级到 7.0 版本。

Hibernate Search 向红帽构建 Quarkus 应用程序提供索引和全文本搜索功能。版本 7.0 引入了对 geo-point 字段的默认配置的改进、新功能以及一些显著变化。

如需了解更多详细信息，请参阅 影响与早期版本兼容性的更改。

要了解更多有关 Hibernate Search 7.0 的信息，请参阅以下资源：

- Quarkus Hibernate 搜索 指南
- Hibernate Search 7.0.1 参考文档

1.3.2.5. 新的 OpenSearch Dev Service

Red Hat build of Quarkus 3.8 引入了一个新的 OpenSearch Dev Service。

当您使用 Hibernate Search 时，Dev Services 默认为根据 Hibernate Search 配置启动 Elasticsearch 或 OpenSearch。

要将 Dev Services 配置为使用 OpenSearch，请指定以下设置：

quarkus.elasticsearch.devservices.distribution=opensearch

如需更多信息，请参阅 Quarkus "Dev Services for Elasticsearch" 指南中的 配置 镜像部分。

1.3.3. Observability（可观察性）

1.3.3.1. 使用 MeterRegistry 自定义 Micrometer

红帽构建的 Quarkus 3.8 引进了许多使用作为上下文和依赖注入(CDI) Bean 实施的新 **MeterRegistryCustomizer** 接口来自定义 Micrometerterter 的方法。

您可以使用以下方法自定义 Micrometer ：

- 通过使用 **MeterFilter** 实例自定义 **MeterRegistry** 实例发送的指标。**Micrometer** 扩展检测到 **MeterFilter** CDI Bean，并在初始化 **MeterRegistry** 实例时使用它们。
- 通过将 **HttpServerMetricsTagsContributor** 用于服务器 HTTP 请求。通过提供实现 **io.quarkus.micrometer.runtime.HttpServerMetricsTagsContributor** 的 CDI Bean，用户代码可以根据 HTTP 请求的详细信息提供任意标签。
- 通过使用 **MeterRegistryCustomizer** 进行计量 registry 的任意自定义。用户代码可以通过提供实现 **io.quarkus.micrometer.runtime.MeterRegistryCustomizer** 的 CDI Bean 来更改激活的任何 **MeterRegistry** 的配置。

如需更多信息，请参阅 Quarkus " Micrometer Metrics" 指南中的自定义 Micrometer 部分。

1.3.3.2. Micrometer @MeterTag 支持

Micrometer 定义两个注释，**@Counted** 和 **@Timed**，您可以添加到方法中。

使用红帽构建的 Quarkus 3.8，微主题器可将 **@Meter** 注释添加到标上 **@Counted** 和 **@Timed** 的方法的参数。

@MeterTag 注释使用 **io.micrometer.common.annotation** 软件包中的 **ValueResolver** 或 **ValueExpressionResolver** 解析器动态地为方法计数器或计时器分配额外的标签值。

如需更多信息，请参阅 Quarkus Micrometer Metrics 指南。

1.3.3.3. Micrometer 支持的 netty 指标

红帽构建的 Quarkus 3.8 引入了对从 Micrometer 指标库收集 Netty 分配器指标的支持。

引入了 **quarkus.micrometer.binder.netty.enabled** 属性，允许在启用了 Micrometer 支持时收集 Netty 指标。

Netty 分配器指标可以深入了解您的 Netty 框架内的内存分配和使用，这有助于了解使用 Netty 的红帽构建的 Quarkus 应用程序的性能。

收集以下指标：

指标	描述
netty.allocator.memory.used	分配器使用的内存大小，以字节为单位
netty.allocator.memory.pinned	分配的缓冲区使用的内存的大小（以字节为单位）。
netty.allocator.pooled.arenas	池分配器的 arenas 数
netty.allocator.pooled.cache.size	池分配器的缓存的大小（以字节为单位）
netty.allocator.pooled.threadlocal.caches	池 **分配器** 的线程本地缓存数量
netty.allocator.pooled.chunk.size	池分配器的内存块的大小，以字节为单位
netty.eventexecutor.tasks.pending	事件执行器中待处理的任务数量

有关 Micrometer 的更多信息，请参阅 Quarkus Micrometer 指标 指南。

1.3.3.4. 将 OkHttp tracing gRPC exporter 替换为 Vert.x

在 Red Hat build of Quarkus 3.8 中，OpenTelemetry (OTel)扩展 **quarkus-opentelemetry** 被改进，将默认的 OTel exporter 替换为基于 Vert.x 构建的 Quarkus 实现。

这消除了对 OkHttp 库的依赖项。导出器仍然会自动与上下文和依赖注入(CDI)连接，因此 **quarkus.otel.traces.exporter** 属性默认为 **cdi**。

如需更多信息，请参阅使用 OpenTelemetry 的 Quarkus 指南。

1.3.4. 安全性

1.3.4.1. 分割超过 4KB 的 OIDC 会话 Cookie 的功能

使用红帽构建的 Quarkus 3.8，如果其内容大小超过 4KB，您可以将 OpenID Connect (OIDC)会话 Cookie 分成较小的 Cookie。

通常，默认情况下，会话 Cookie 由三个令牌连接组成，包括三个令牌，即 ID、访问和刷新令牌。如果它的大小大于 4KB，一些浏览器可能无法处理它。

在这个版本中，一个超过 4KB 的会话 Cookie 会自动分成多个块。

如需更多信息，请参阅 用于保护 Web 应用程序的 Quarkus OIDC 授权代码流机制 。

1.3.4.2. 在 HTTP 请求完成后创建 OIDC SecurityIdentity 实例

使用红帽构建的 Quarkus 3.8，您可以在 HTTP 请求完成后为身份验证目的创建 OIDC **SecurityIdentity** 实例。

在这个版本中，**quarkus-oidc** 扩展包括 **io.quarkus.oidc.TenantIdentityProvider** 接口，您可以注入并调用在 HTTP 请求完成后将令牌转换为 **SecurityIdentity** 实例。

如需更多信息，请参阅以下 Quarkus 资源：

- OIDC 授权代码流机制，用于保护 Web 应用程序 指南。
- HTTP 请求完成 部分"OIDC bearer 令牌身份验证"指南中的身份验证。

1.3.4.3. 自定义 OIDC JavaScript 请求检查

Red Hat build of Quarkus 3.8 引入了 OIDC **JavaScriptRequestChecker** bean，可用于自定义 JavaScript 请求检查。

如果您使用带有红帽构建的 Quarkus Web 应用的单页应用程序(SPAs)和 JavaScript API，如 **Fetch** 或 **XMLHttpRequest**(XHR)，您必须在浏览器脚本中设置标头，将请求标识为 JavaScript 请求。但是，脚本引擎也可以设置特定于引擎的请求标头本身。

在这个版本中，您可以注册自定义 **io.quarkus.oidc.JavaScriptRequestChecker** bean，它会在当前请求是 JavaScript 请求时通知 Red Hat build of Quarkus，这有助于避免创建冗余标头。

1.3.4.4. 现在支持延迟 OIDC JWK 解析

Red Hat build of Quarkus 3.8 引入了对延迟 OIDC JSON Web 密钥(JWK)解析的支持，您现在可以解决当前令牌可用的密钥。

此发行版本添加了 **quarkus.oidc.jwks.resolve-early** 配置属性。默认情况下，此属性设置为 **true**，这意味着当您建立 OIDC 供应商连接时，JWK 密钥会解决。

但是，您可以将其设置为 **false**，同时启用密钥延迟解析令牌验证。延迟 JWK 解析在初始化时使用当前令牌而不是读写方法。例如，令牌可能会提供有关如何正确解析密钥的信息。

1.3.4.5. 使用 mTLS 和 HTTP Restrictions 增强安全性

当将 mTLS 客户端身份验证(**quarkus.http.ssl.client-auth**)设置为 **required** 时，Red Hat build of Quarkus 会自动禁用普通的 HTTP 端口，以确保只接受安全 HTTPS 请求。要启用普通的 HTTP，请将 **quarkus.http.ssl.client-auth** 配置为 **请求** 或设置 **quarkus.http.ssl.client-auth=required** 和 **quarkus.http.insecure-requests=enabled**。

1.3.4.6. HTTP 权限和角色移到运行时配置

Red Hat build of Quarkus 已更新，允许运行时配置 HTTP 权限和角色，跨配置集启用灵活的安全设置。这解决了原生可执行文件锁定到构建时安全配置的问题。现在，针对每个配置文件动态调整安全性，适用于 JVM 和原生模式。

1.3.4.7. 将 OIDC 范围属性映射到 Bearer 令牌身份验证中的 SecurityIdentity 权限

如果您在红帽构建的 Quarkus 中使用 Bearer 令牌身份验证，您可以从验证的 JWT 访问令牌中映射 **SecurityIdentity** 角色。Red Hat build of Quarkus 3.8 引入了将 OIDC scope 参数映射到 **SecurityIdentity** 对象的权限的功能。

例如，您可以使用 **@PermissionAllowed ("orders_read")** 来请求 JWT 令牌具有 **orders_read** 值的范围声明。

如需更多信息，请参阅 Quarkus OIDC Bearer 令牌身份验证 指南。

1.3.4.8. 使用 CDI 观察安全事件

使用红帽构建的 Quarkus 3.8，您可以使用上下文和依赖注入(CDI)来观察身份验证和授权安全事件。

CDI 观察程序可以是同步或异步的，并支持以下安全事件的报告：

- **io.quarkus.security.spi.runtime.AuthenticationFailureEvent**
- **io.quarkus.security.spi.runtime.AuthenticationSuccessEvent**
- **io.quarkus.security.spi.runtime.AuthorizationFailureEvent**
- **io.quarkus.security.spi.runtime.AuthorizationSuccessEvent**
- **io.quarkus.oidc.SecurityEvent**

如需更多信息，请参阅 Quarkus"安全提示和技巧"指南中的 观察安全事件 部分。

1.3.4.9. 不支持 OIDC 授权代码流

Red Hat build of Quarkus 3.8 引入了对 OpenID Connect (OIDC)授权代码流的支持。

当 OIDC 授权服务器发出一个 ID 令牌以响应授权请求时，ID 令牌包含一个 非ce 声明，它必须与非ce 身份验证请求查询参数匹配。此功能通过确保返回 ID 令牌来响应原始授权请求，且不是重播响应，这有助于缓解重播攻击。

1.3.4.10. 支持 OIDC 请求过滤器

使用红帽构建的 Quarkus 3.8，您可以通过注册一个或多个 OidcRequestFilter 实现来自定义由 quarkus-oidc-client 或 quarkus-oidc 扩展提供的 OIDC 客户端请求。

例如，OIDC 请求过滤器可以分析请求正文，并将其摘要添加为新的标头值。

如需更多信息，请参阅 Quarkus "OIDC authorization code flow mechanism for protect web application" 指南中的 OIDC request filters 部分。

1.3.4.11. 引入了新的 OIDC @TenantFeature 注解，将 OIDC 功能绑定到租户

在红帽构建的 Quarkus 3.8 中，引入了一个新的 @TenantFeature 注解，将 OpenID Connect (OIDC)功能绑定到 OIDC 租户。

io.quarkus.oidc.Tenant 注解现在用于解析租户配置。

1.3.4.12. 支持 OIDC 令牌传播

红帽构建的 Quarkus 3.8 引入了对 OIDC 令牌传播的支持。

在这个版本中，红帽构建的 Quarkus 端点使用 REST 客户端将传入的 OIDC 访问令牌传播到预期访问令牌的其他安全端点。

1.3.4.13. 客户端证书的角色映射

在使用 Mutual TLS (mTLS)身份验证机制时，红帽构建的 Quarkus 3.8 现在支持将通用名称(CN)属性从客户端的 X.509 证书映射到角色。此功能在特定条件下激活：

此功能在特定条件下激活：

- 如果使用 quarkus.http.ssl.client-auth=required 或 quarkus.http.ssl.client-auth=request启用 mTLS 身份验证机制
- application.properties 文件引用带有 quarkus.http.auth.certificate-role-properties 属性的角色映射文件。

角色映射文件应该具有 CN=role1,role,...,roleN 格式，并使用 UTF-8 进行编码。

1.3.4.14. 支持使用内联证书链进行令牌验证

Red Hat build of Quarkus 3.8 引入了使用令牌中内联的 X.509 证书链来验证 OIDC bearer 访问令牌。

这意味着，您在从叶证书中提取公钥前验证证书链。leaf 证书指的是 X.509 证书，它位于证书链的末尾。要验证令牌的签名，您可以使用这个公钥。

1.3.5. 工具

1.3.5.1. 使用 OpenRewrite 扩展更新功能

Quarkus 更新 现在支持 OpenRewrite recipes 用于外部红帽构建的 Quarkus 扩展，使其功能只扩展超过内置扩展的功能。引进了新的方法，增强对外部扩展的迁移支持。

请注意，红帽支持使用 Quarkus 开发工具的开发支持，包括 Quarkus CLI 以原型、开发、测试和部署 Quarkus 应用程序的红帽构建。https://access.redhat.com/support/offerings/developer/红帽不支持在生产环境中使用 Quarkus 开发工具。

1.3.6. Web

1.3.6.1. 通过 CDI 集成增强 /info 端点

使用 quarkus-info 的应用程序现在可以通过 CDI 集成通过额外的数据增强 /info 端点。此功能增强了自定义和扩展应用程序诊断和元数据可见性的功能。

注意

如需更多信息，请参阅 Quarkus 社区的 CDI 集成指南。

1.3.6.2. 对 REST 客户端被动中的 SSE 支持的改进

使用红帽构建的 Quarkus 3.8，REST 客户端的服务器事件(SSE)功能得到了增强，启用完整的事件返回和过滤。REST 客户端中的这些更新和新描述为开发人员提供了管理实时数据流的控制和灵活性。

1.3.6.3. REST Client Reactive Jackson 中的 ObjectMapper 自定义

使用红帽构建的 Quarkus 3.8，您可以在使用 rest-client-reactive-jackson 扩展时自定义 ObjectMapper。您可以使用注解 @Client ObjectMapper 添加仅客户端使用的自定义 ObjectMapper。

重要

对于您要继承默认设置的任何自定义操作，您不必修改默认的对象映射器 defaultObjectMapper。您必须改为创建一个副本。defaultObjectMapper 是红帽构建的 Quarkus 本身配置的 ObjectMapper 实例，作为 CDI bean 提供，以及 RESTEasy Reactive 和 REST 客户端（默认应用程序）使用。

如需更多信息，请参阅 Quarkus "Using the REST client" 指南中的自定义 ObjectMapper in REST Client Reactive Jackson 部分。

1.3.6.4. @TestHTTPResource中的 path 参数支持

@TestHTTPResource 注释现在支持路径参数。由于 URI 格式的不合规，不再应用作为 URI 字符串的验证。

1.4. 支持和兼容性

您可以找到有关与 Red Hat build of Quarkus 3.8 以及红帽客户门户网站中的高级别支持生命周期政策兼容的支持的配置和工件的详细信息，如下所示：

- 有关支持的配置、OpenJDK 版本和经过测试的集成列表，请参阅 Red Hat build of Quarkus 支持的配置。
-

有关 Red Hat build of Quarkus 支持的 Maven 工件、扩展和 BOM 列表，请参阅 红帽构建的 Quarkus 组件详情。

- 有关所有红帽产品的正式发布、全面支持和维护支持日期，请参阅 Red Hat Application Services 产品更新和支持政策。

1.4.1. 产品更新和支持生命周期政策

在 Red Hat build of Quarkus 中，功能版本可以是主版本，也可以是引入新功能或支持的次版本。红帽构建的 Quarkus 发行版本号与 Quarkus 社区项目的 Long-Term Support (LTS)版本直接一致。红帽构建的 Quarkus 功能发行版本的版本号与它所基于的 Quarkus 社区版本匹配。如需更多信息，请参阅 Quarkus 博客文章的 Long-Term Support (LTS)。

重要

红帽不会为社区版本的每个版本发布 Quarkus 版本。红帽构建的 Quarkus 功能版本的节奏大约每六个月。

Red Hat build of Quarkus 为功能发行版本提供完全支持，直到后续版本发布为止。当一个功能版本被一个新版本取代时，红帽将继续为这个版本提供后续的 6 个月的维护支持，如以下支持生命周期图表概述 [Fig. 1]。

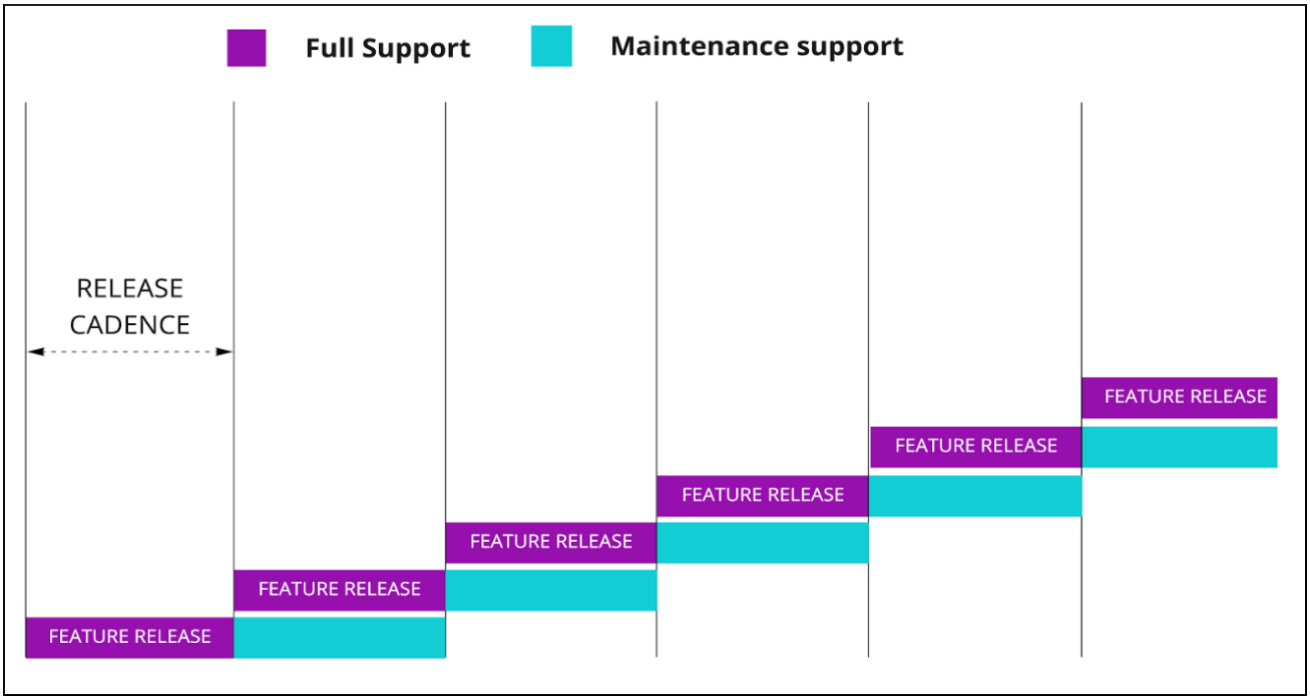


图 1.发行节奏和支持 Red Hat build of Quarkus 的生命周期

在版本的完全支持阶段和维护支持阶段中，红帽还提供 'service-pack (SP)' 更新和"micro"版本来修

复错误以及常见漏洞和暴露(CVE)。

红帽构建的 Quarkus 新功能可能会引入对底层技术或平台中依赖项的增强、创新和更改。有关连续功能发行版中的新或更改内容的详细信息，请参阅 新功能、增强功能和技术变化。

虽然在升级到最新版本后，Red Hat build of Quarkus 的大多数功能都会继续按预期工作，但在某些情况下，您可能需要更改现有应用程序，或对您的环境或依赖项执行一些额外的配置。因此，在将 Red Hat build of Quarkus 升级到最新版本前，请仔细检查影响与之前版本的兼容性以及发行注记中 已弃用的 组件和功能部分的更改。

1.4.2. 经过测试并验证的环境

Red Hat build of Quarkus 3.8 包括在以下 Red Hat OpenShift Container Platform 版本： 4.15、4.12 和 Red Hat Enterprise Linux 8.9 中。

有关支持的配置列表，登录到红帽客户门户网站并查看红帽知识库解决方案 Red Hat build of Quarkus 支持的配置。

1.4.3. 开发支持

红帽为以下红帽构建的 Quarkus 功能、插件、扩展和依赖项提供了 开发支持 ：

功能

- 持续测试
- Dev Services
- Dev UI
- 本地开发模式
- 远程开发模式

plugins

- Maven 协议缓冲器插件

1.4.3.1. 开发工具

红帽提供对使用 Quarkus 开发工具的 开发支持，包括 Quarkus CLI 和 Maven 和 Gradle 插件，以原型、开发、测试和部署红帽构建的 Quarkus 应用程序。

红帽不支持在生产环境中使用 Quarkus 开发工具。如需更多信息，请参阅红帽知识库文章 开发支持覆盖范围。

1.5. 弃用的组件和功能

本节中列出的组件和功能在 Red Hat build of Quarkus 3.8 中被弃用。这个版本包括了并提供支持。但是，不会对这些组件和功能进行任何增强，以后可能会删除它们。

有关本发行版本中弃用的组件和功能列表，请登录到红帽客户门户网站并查看 Red Hat build of Quarkus 组件详情页面。

1.5.1. 弃用 DeploymentConfig

在 Red Hat build of Quarkus 3.8 中，DeploymentConfig 对象在 OpenShift 中已弃用，在 Red Hat build of Quarkus 中也被弃用。现在，Deployment 是 quarkus-openshift 扩展的默认和首选部署类型。

如果您使用 DeploymentConfig 重新部署之前部署的应用，默认情况下，这些应用将使用 Deployment，但不会删除前面的 DeploymentConfig。这会导致部署新的和旧应用程序，因此您必须手动删除 DeploymentConfig。

但是，如果要继续使用 DeploymentConfig，仍可以通过将 quarkus.openshift.deployment-kind 设置为 DeploymentConfig 来完成此操作。

如需更多信息，请参阅 将红帽构建的 Quarkus 应用程序部署到 OpenShift Container Platform。

1.5.2. OpenShift Service Binding Operator 已弃用

OpenShift Service Binding Operator 在 OpenShift Container Platform (OCP) 4.13 及更高版本中已弃用，计划在以后的 OCP 发行版本中删除。

1.5.3. 弃用 quarkus-reactive-routes

从 2.13 版本 2.13 开始，quarkus-reactive-routes 已被弃用，计划在以后的版本中删除。smallrye JWT 不再包含 quarkus-reactive-routes ；因此，其自动包含将停用。要维护功能，请在构建配置中添加 quarkus-reactive-routes。

1.5.4. quarkus-test-infinispan-client 工件的停用

quarkus-test-infinispan-client 工件已被停用，不再是 Red Hat build of Quarkus 的一部分。此更改遵循其冗余性，因为它没有在 Quarkus 核心存储库之外使用，并且已被 Dev Services for Infinispan 替代。

1.6. 技术预览

本节列出了在红帽构建的 Quarkus 3.8 中作为技术预览提供的功能和扩展。

重要

技术预览功能不被红帽产品服务级别协议(SLA)支持，且其功能可能并不完善，红帽建议您不要在生产环境中使用它们。这些技术预览功能可以使用户提早试用新的功能，并有机会在开发阶段提供反馈意见。

有关红帽技术预览功能的更多信息，请参阅 技术预览功能范围。

1.6.1. 引入了 Hibernate 搜索管理端点

红帽构建的 Quarkus 3.8 会公开 Hibernate Search 的 HTTP 管理端点作为技术预览功能。

通过此功能，您可以触发数据索引和其他维护任务。默认情况下，此端点不会被启用。要启用它，请将以下配置属性设置为 true ：

- quarkus.management.enabled=true

- quarkus.hibernate-search-orm.management.enabled=true

红帽构建的 Quarkus 会在管理界面的 /q/hibernate-search/reindex 下公开管理端点，该端点默认在端口 9000 上公开。

如需更多信息，请参阅以下资源：

- Quarkus Hibernate 搜索 指南
- Quarkus 管理界面参考指南

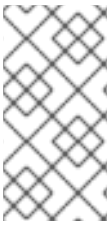
1.6.2. 处于技术预览的扩展列表

- RESTEasy Reactive JAXB，quarkus-resteasy-reactive-jaxb。JAXB 序列化支持 RESTEasy Reactive。此扩展与 quarkus-resteasy 扩展或依赖于它的任何扩展不兼容。
- smallrye Stork，quarkus-smallrye-stork。smallrye Stork 是一个动态服务发现和选择框架，用于查找和选择服务实例。
- Elasticsearch REST 客户端 quarkus-elasticsearch-rest-client.使用 REST 低级别客户端连接到 Elasticsearch 集群。
- Hibernate Reactive、quarkus-hibernate-reactive.Hibernate ORM 的被动 API，支持非阻塞数据库驱动程序以及与数据库交互的被动样式。
- MongoDB 客户端，quarkus-mongodb-client.以 imperative 或 reactive 样式连接到 MongoDB。
- 主动 MS SQL 客户端, quarkus-reactive-mssql-client.使用被动模式连接到 Microsoft SQL Server 数据库。
- 主动 Oracle 客户端, quarkus-reactive-oracle-client.使用被动模式连接到 Oracle 数据库。

- Apache Kafka Streams, quarkus-kafka-streams.根据 Apache Kafka 实施流处理应用程序。

- Kubernetes Service Binding， quarkus-kubernetes-service-binding。根据 Kubernetes 服务绑定规范读取运行时配置。

- OpenShift Client, quarkus-openshift-client.与 OpenShift 交互并开发 OpenShift Operator。



注意

虽然 OpenTelemetry 扩展 quarkus-opentelemetry 可能会被标记为 "Tech-Preview"，但它被完全支持。该标签正在更新。

1.7. 影响与早期版本兼容性的更改

这部分论述了红帽构建的 Quarkus 3.8 中的更改，它会影响与之前产品版本构建的应用程序的兼容性。

查看这些中断更改，并执行必要的步骤，以确保应用程序在将其更新至红帽构建的 Quarkus 3.8 后继续运行。

要自动执行许多这些更改，请使用 quarkus update 命令将项目更新至最新的红帽构建的 Quarkus 版本。

1.7.1. Core

1.7.1.1. Stork 负载平衡器配置的更改

您不再使用以前的配置名称 stork."service-name".load-balancer 和 quarkus.stork."service-name".load-balancer 来配置 Stork 负载均衡器。反之，使用 quarkus.stork."service-name".load-balancer.type 进行配置。

1.7.1.2. OkHttp 和 Okio 的依赖项管理更新

OkHttp 和 Okio 已从 Quarkus Platform BOM 中删除，它们的版本不再被强制执行，从而解决了与过时的依赖项相关的问题。这个更改会影响测试框架依赖项，并简化了运行时依赖项。使用这些依赖项的

开发人员现在在构建文件中指定其版本。另外，quarkus-test-infinispan-client 工件已被删除，因为对 Infinispan 提供强大的 Dev Services 支持。

1.7.1.3. Java 版本要求更新

从此版本 Red Hat build of Quarkus 开始，删除了对 Java 11 的支持（在以前的版本中被弃用）。Java 21 现在是推荐的版本，但支持 Java 17。

1.7.1.4. RESTEasy Reactive 中的集合的 JAXB 限制

在红帽构建的 Quarkus 中，将 RESTEasy Reactive 与 XML Binding (JAXB)的 Java 架构搭配使用，不支持将集合、数组和映射用作 REST 方法中的参数或返回类型。为克服 JAXB 的这一限制，请在带有 @XmlRootElement 的类内封装这些类型。

1.7.1.5. 构建时 @StaticInitSafe 的强制规格

在静态初始化阶段，红帽构建的 Quarkus 会收集要注入 CDI Bean 的配置。然后，收集的值会与其运行时初始化对应的部分进行比较，如果检测到不匹配，应用程序启动会失败。使用红帽构建的 Quarkus 3.8，您现在可以使用 @io.quarkus.runtime.annotations.StaticInitSafe 注解来告知用户注入的配置：

- 在构建时设置
- 无法更改
- 在运行时安全地使用，指示红帽构建的 Quarkus 无法在配置不匹配时启动失败

1.7.1.6. Qute ：默认情况下标签模板的隔离执行

模板中的用户标签现在默认以隔离方式执行，从而限制对调用模板的上下文的访问。在这个版本中，可以在标签模板中更改数据处理，可能会影响其当前功能。要绕过此隔离并保持对父上下文的访问权限，请在标签调用中包含 _isolated=false 或 _unisolated，例如：NT itemDetail item showImage=true _isolated=false。这种方法允许标签从父上下文访问数据，如以前一样。此更改可最小化从父上下文到标签的意外数据暴露，从而增强模板数据完整性。但是，可能需要对现有模板的依赖共享上下文访问更新，代表一个显著的更改，可能会影响与这种隔离机制不熟悉的用户。

1.7.1.7. Qute ：解决类型轮询问题

ResultNode 类已更新为抽象类，而不是接口，尽管在公共 API 中不应该被用户实现。Qute API 现在将 CompletionStage 实现限制为 java.util.concurrent.CompletableFuture 和

io.quarkus.qute.CompletedStage，这是可通过 -Dquarkus.qute.unrestricted-completion-stage-support=true 的一个限制。

1.7.1.8. Quarkus-rest-client 扩展重命名为 quarkus-resteasy-client

在 Red Hat build of Quarkus 3.8 中，以下 quarkus-rest-client 扩展被重命名：

旧名称	新名称
quarkus-rest-client	quarkus-resteasy-client
quarkus-rest-client-mutiny	quarkus-resteasy-client-mutiny
quarkus-rest-client-jackson	quarkus-resteasy-client-jackson
quarkus-rest-client-jaxb	quarkus-resteasy-client-jaxb
quarkus-rest-client-jsonb	quarkus-resteasy-client-jsonb

1.7.1.9. 在注入 @TestHTTPResource 时删除 URI 验证

@TestHTTPResource 注释现在支持路径参数。由于 URI 格式的不合规，不再应用作为 URI 字符串的验证。

1.7.1.10. 使用依赖项调整对 GraalVM SDK 23.1.2 的更新

GraalVM SDK 版本在 Red Hat build of Quarkus 3.8 中已更新至 23.1.2。使用需要 GraalVM 替换扩展的开发人员应该从 org.graalvm.sdk:graal-sdk 切换到 org.graalvm.sdk:nativeimage 以访问必要的类。对于使用 org.graalvm.js:js 的用户，请将这个依赖项替换为 org.graalvm.polyglot:js-community 用于社区版本。对于企业版本，请将此依赖项替换为 org.graalvm.polyglot:js。使用 quarkus update 自动调整 graal-sdk。但是，必须手动更改 js 依赖项。虽然这个变化不太可能，但这个更改可能会影响依赖的用户：

- org.graalvm.sdk:collections
- org.graalvm.sdk:word

1.7.1.11. 对 QuarkusComponentTest的各种调整

在本发行版本中，QuarkusComponentTest 有几个调整。它仍然是实验性的，不受 Red Hat build

of Quarkus 支持。此实验状态表示 API 可能会随时更改，以反映收到的反馈。

QuarkusComponentTestExtension 现在不可变，需要通过简化的 constructor QuarkusComponentTestExtension (Class…) 或 QuarkusComponentTestExtension.builder （） 方法进行编程注册。测试实例生命周期（可以是 Lifecycle#PER_METHOD （默认）或 Lifecycle#PER_CLASS 规定，决定 CDI 容器何时启动和停止； PER_METHOD 会在每个测试后启动容器并停止它，而 PER_CLASS 在所有测试之前都会启动它，并在所有测试后停止它。这代表了之前版本的变化，容器总是在所有测试之前启动并停止。

1.7.2. data

1.7.2.1. Hibernate ORM 升级到 6.4

在红帽构建的 Quarkus 3.8 中，Hibernate Object-Relational Mapping (ORM)已升级到版本 6.4，并引进了以下破坏更改：

- 与一些旧的数据库版本兼容将被丢弃。有关支持的版本的更多信息，请参阅支持的结束。
- 现在，数字文字被解释为 Jakarta Persistence 3.2 中定义的。

如需更多信息，请参阅 Hibernate ORM 6.4 迁移指南。

1.7.2.2. 在启动时 Hibernate ORM 数据库版本验证

当在红帽构建的 Quarkus 3.8 上使用 Hibernate ORM 时，您可以在应用程序启动时验证指定的数据库版本。

要让 Hibernate ORM 生成效率更高的 SQL 并利用更多数据库功能，您可以在 applications.properties 文件中为 Hibernate 数据库设置特定的数据库版本。

例如： quarkus.datasource.db-version = 14.0

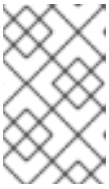
在这个 3.8 发行版本中，红帽构建的 Quarkus 会根据应用程序要连接到的实际数据库版本验证指定的数据库版本，并在出现不匹配时在启动时抛出异常。

如需更多信息，请参阅 Quarkus "使用 Hibernate ORM 和 Jakarta persistence" 指南中的 支持的数据库 部分。

1.7.2.3. Hibernate Search 升级到 7.0

在 Red Hat build of Quarkus 3.8 中，Hibernate Search 被升级到 7.0 版本，并引入了以下破坏更改：

- quarkus.hibernate-search-orm.coordination.entity-mapping.outbox-event.uuid-type 和 quarkus.hibernate-search-orm.coordination.entity-mapping.agent.uuid-type 配置属性已更改：
  - uuid-binary 已被弃用，而是 二进制
  - uuid-char 已弃用，而是使用 char
- quarkus.hibernate-search-orm.elasticsearch.query.shard-failure.ignore 属性的默认值从 true 改为 false，这意味着如果搜索操作中至少有一个分片失败，则 Hibernate Search 现在会抛出异常。要获得前面的行为，请将此配置属性设为 true。



注意

如果定义多个后端，您必须为每个 Elasticsearch 后端设置这个配置属性。

- 正则表达式 predicate 中的补充运算符(~)被移除，没有替换它的替代方法。
- Hibernate 搜索依赖项在其工件 ID 中不再有 -orm6 后缀；例如，应用程序现在依赖于 hibernate-search-mapper-orm 模块，而不是 hibernate-search-mapper-orm-orm6。

如需更多信息，请参阅以下资源：

- Hibernate 搜索文档

- Hibernate Search 7.0.0.Final: Migration guide from 6.2

1.7.2.4. SQL Server Dev Services 升级到 2022-latest

SQL Server 的 dev Services 将其默认镜像从 mcr.microsoft.com/mssql/server:2019-latest 更新至 mcr.microsoft.com/mssql/server:2022-latest。

用户更喜欢使用 Red Hat build of Quarkus "Configure data sources" 指南中的 References 部分中详述的配置属性来指定一个替代版本。

1.7.2.5. 升级到 Flyway 会为 Oracle 用户添加额外的依赖项

在 Red Hat build of Quarkus 3.8 中，Flyway 扩展被升级到 Flyway 9.20.0，它为 Oracle 用户提供额外的依赖项 flyway-database-oracle。

Oracle 用户必须更新 pom.xml 文件，使其包含 flyway-database-oracle 依赖项。要做到这一点，请执行以下操作：

```
<dependency>
  <groupId>org.flywaydb</groupId>
  <artifactId>flyway-database-oracle</artifactId>
</dependency>
```

如需更多信息，请参阅 使用 Flyway 的 Quarkus 指南。

1.7.3. 原生

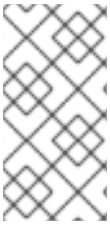
1.7.3.1. Kafka 扩展中的 strimzi OAuth 支持问题

quarkus-bom 中的 Kafka 扩展的 Strimzi OAuth 支持现在使用 io.strimzi:strimzi-kafka-oauth 版本 0.14.0，引入一个已知问题，导致原生构建失败。'io.smallrye.reactive.kafka.graal.Target_com_jayway_jsonpath_internal_DefaultsImpl 的错误不会被加载，方法是将 io.strimzi:kafka-oauth-common 添加到项目的 classpath 中。

1.7.4. Observability（可观察性）

1.7.4.1. 添加了 @AddingSpanAttributes 注释

当在 Quarkus 3.8 中使用 Opentelemetry (oTel)检测时，您现在可以使用 io.opentelemetry.instrumentation.annotations.AddingSpanAttributes 注解来在任何上下文依赖注入 (CDI)-aware bean 中添加注解的方法。



注意

如果您错误地给方法标上 @AddingSpanAttributes 和 @WithSpan 注释，则 @WithSpan 注释将具有优先权。

如需更多信息，请参阅 Quarkus "Using OpenTelemetry" 指南中的 CDI 部分。

1.7.4.2. quarkus-smallrye-metrics 扩展不再被支持

使用红帽构建的 Quarkus 3.8 时，quarkus-smallrye-metrics 扩展不再被支持。现在，它只作为社区扩展提供。不建议在生产环境中使用它。

在 Red Hat build of Quarkus 3.8 中，quarkus-smallrye-metrics 被完全支持的 quarkus-micrometer 扩展替代。

1.7.4.3. quarkus-smallrye-opentracing 扩展不再被支持

使用红帽构建的 Quarkus 3.8 时，不再支持 SmallRye OpenTracing。要继续使用分布式追踪，请将您的应用程序迁移到 SmallRye OpenTelemetry，现在在这个发行版本中被完全支持，不再是一个技术预览功能。如果您仍然需要使用 quarkus-smallrye-opentracing，请通过更新 groupId 并手动指定版本，调整应用程序以使用 Quarkiverse 中的扩展。

1.7.4.4. 重构调度程序和 OpenTelemetry Tracing 扩展

在 Red Hat build of Quarkus 3.8 中，已重构 OpenTelemetry Tracing 和 quarkus-scheduler 扩展的集成。

在此次更新之前，只有 @Scheduled 方法具有一个新的 io.opentelemetry.api.trace.Span 类，该类在启用追踪时自动关联。也就是说，当 quarkus.scheduler.tracing.enabled 配置属性设为 true 时，quarkus-opentelemetry 扩展可用。

在这个 3.8 发行版本中，所有调度的作业（包括以编程方式调度的作业）在启用追踪时自动关联 Span。每个调度方法的唯一作业标识符都是生成的，通过设置

io.quarkus.scheduler.Scheduled#identity 属性或使用 JobDefinition 方法来指定。在此次更新之前，范围名称后是 < simpleclassname>.<methodName> 格式。

如需更多信息，请参阅以下 Quarkus 资源：

- 调度程序参考
- 使用 OpenTelemetry

1.7.5. 安全性

1.7.5.1. 使用 mTLS 和 HTTP Restrictions 增强安全性

当将 mTLS 客户端身份验证(quarkus.http.ssl.client-auth)设置为 required 时，Red Hat build of Quarkus 会自动禁用普通的 HTTP 端口，以确保只接受安全 HTTPS 请求。要启用普通的 HTTP，请将 quarkus.http.ssl.client-auth 配置为 请求 或设置 quarkus.http.ssl.client-auth=required 和 quarkus.http.insecure-requests=enabled。

1.7.5.2. JWT 扩展会删除不必要的 Reactive 路由依赖项

JWT 扩展不再依赖于主动路由扩展。如果您的应用程序同时使用 JWT 和 Reactive Routes 功能，但没有声明对 Reactive Routes 的明确依赖项，您必须添加此依赖项。

1.7.5.3. Keycloak 授权丢弃了 keycloak-adapter-core 依赖项

因为更新到 Keycloak 22.0.0 及其对扩展功能的更新，quarkus-keycloak-authorization 扩展不再包含 org.keycloak:keycloak-adapter-core 依赖项。在以后的 Keycloak 版本中，计划删除 Keycloak Java 适配器代码。如果您的应用程序需要这个依赖项，请手动将它添加到项目的 pom.xml 中。

1.7.5.4. 使用 CDI 拦截器解析 RESTEasy Classic 中的 OIDC 租户不再被支持

您不再使用 Context 和 Dependency Injection (CDI)注解和拦截器来解析 RESTEasy Classic 应用的租户 OIDC 配置。

由于在 CDI 拦截器和需要身份验证的检查之前强制进行安全检查，使用 CDI 拦截器解析多个 OIDC 供应商配置标识符不再可以正常工作。

使用 @Tenant 注释或自定义 io.quarkus.oidc.TenantResolver。

如需更多信息，请参阅 Quarkus" 使用 OIDC 多租户指南"中的解决注解 部分。

1.7.5.5. 使用 OIDC @Tenant 注解将 OIDC 功能绑定到租户不再可能

在 Red Hat build of Quarkus 3.8 中，现在使用 quarkus.oidc.TenantFeature 注解而不是 quarkus.oidc.Tenant 将 OpenID Connect (OIDC)功能绑定到 OIDC 租户。

quarkus.oidc.Tenant 注解现在用于解析租户配置。

1.7.5.6. 安全配置集灵活性增强

Red Hat build of Quarkus 3.8 允许运行时配置 HTTP 权限和角色，跨配置集启用灵活的安全设置。这解决了原生可执行文件锁定到构建时安全配置的问题。现在，针对每个配置文件动态调整安全性，适用于 JVM 和原生模式。

1.7.6. Standards

1.7.6.1. 修正 GraphQL 指令应用程序

已修正了基于注解的 GraphQL 指令的应用程序，以确保它们只应用于声明它们的 schema 元素类型。

例如，如果声明了一个指令应用到 GraphQL 元素类型 FIELD，但错误地应用到不同的元素类型，它仍然在不应该适用的元素的 schema 中可见，从而导致一个无效的模式。现在，这个问题已被修正，指令会根据其适用性声明进行了检查。

如果您以这种方式应用了指令，它们将不再出现在架构中，红帽构建的 Quarkus 3.8 会在构建期间记录警告。

1.7.7. OpenAPI 标准化 POJO 和原语的内容类型默认值

当未提供 @ContentType 注释时，此更改已标准化了用于生成 OpenAPI 文档的默认内容类型。在以前的版本中，默认内容类型在不同的扩展间有所变化，如 RestEasy Reactive、RestEasy Classic、

Spring Web 和 OpenAPI。例如，OpenAPI 始终使用 JSON 作为默认值，而 RestEasy 将 JSON 用于对象类型，以及用于 primitive 类型的文本。现在，所有扩展都已使用统一的默认设置，确保一致性：

- 原语类型 现在统一设置为 text/plain。
- 复杂的 POJO (Plain Old Java Object)类型 默认为 application/json。

虽然跨扩展的行为是一致的，但它根据数据类型来适当地区分，并使用 application /plain 和 POJO 使用 text/plain 和 POJO。这种方法并不意味着相同的内容类型适用于所有 Java 类型，而所有扩展现在都以同样的方式处理内容类型，并针对数据的性质量身定制。

1.7.8. Web

1.7.8.1. 改进了 REST 客户端中的 SSE 处理

红帽构建的 Quarkus 3.8 增强了其 REST 客户端的 Server-Sent Events (SSE)功能，从而实现了完整的事件返回和过滤。REST 客户端中的这些更新和新描述为开发人员提供了管理实时数据流的控制和灵活性。

1.7.8.2. 手动添加 Reactive Routes 依赖项

在版本 3.8 之前，红帽构建的 Quarkus SmallRye JWT 会自动纳入 quarkus-reactive-routes，此功能从版本 3.8 开始停用。为确保继续功能，请手动添加 quarkus-reactive-routes 作为构建配置中的依赖项。

1.8. 已知问题

查看以下已知问题，深入了解红帽构建的 Quarkus 3.8 限制和临时解决方案。

1.8.1. Strimzi OAuth 客户端更新到 0.14.0 的原生构建失败

因为将 io.strimzi:strimzi-kafka-oauth 依赖项更新到 0.14.0，导致 Strimzi OAuth 客户端遇到一个已知问题，从而导致 没有加载 io.smallrye.reactive.kafka.graal.Target_com_jayway_jsonpath_DefaultsImpl 指示的原生构建失败。

临时解决方案： 要解决这个问题，请在 classpath 中包含 io.strimzi:kafka-oauth-common 依赖项。

1.8.2. 在 AArch64 上缺少 Kafka Streams 扩展的原生库

由于没有原生库 **librocksdbjni-linux-AArch64.so**，使用 **quarkus-kafka-streams** 扩展的应用程序在 **AArch64** 系统上具有运行时失败。这个问题会抛出 **java.lang.RuntimeException: librocksdbjni-linux-AArch64.so**，在应用程序启动过程中没有找到 **JAR** 错误。此错误可防止成功初始化 **RocksDB** 组件，这对 **Kafka Streams** 应用程序至关重要。

临时解决方案：目前还没有可用的临时解决方案。

java.lang.RuntimeException 示例： librocksdbjni-linux-AArch64.so 错误

```
09:32:54,059 INFO  [app] ERROR: Failed to start application (with profile [prod])
09:32:54,059 INFO  [app] java.lang.RuntimeException: Failed to start quarkus
09:32:54,060 INFO  [app]  at io.quarkus.runner.ApplicationImpl.doStart(Unknown Source)
09:32:54,060 INFO  [app]  at io.quarkus.runtime.Application.start(Application.java:101)
09:32:54,060 INFO  [app]  at
io.quarkus.runtime.ApplicationLifecycleManager.run(ApplicationLifecycleManager.java:111)
09:32:54,061 INFO  [app]  at io.quarkus.runtime.Quarkus.run(Quarkus.java:71)
09:32:54,061 INFO  [app]  at io.quarkus.runtime.Quarkus.run(Quarkus.java:44)
09:32:54,061 INFO  [app]  at io.quarkus.runtime.Quarkus.run(Quarkus.java:124)
09:32:54,062 INFO  [app]  at io.quarkus.runner.GeneratedMain.main(Unknown Source)
09:32:54,062 INFO  [app] Caused by: java.lang.ExceptionInInitializerError
09:32:54,063 INFO  [app]  at
io.quarkus.kafka.streams.runtime.KafkaStreamsRecorder.loadRocksDb(KafkaStreamsRecorder.java:14
)
09:32:54,063 INFO  [app]  at
io.quarkus.deployment.steps.KafkaStreamsProcessor$loadRocksDb1611413226.deploy_0(Unknown
Source)
09:32:54,063 INFO  [app]  at
io.quarkus.deployment.steps.KafkaStreamsProcessor$loadRocksDb1611413226.deploy(Unknown
Source)
09:32:54,064 INFO  [app]  ... 7 more
09:32:54,064 INFO  [app] Caused by: java.lang.RuntimeException: librocksdbjni-linux-AArch64.so
was not found inside JAR.
09:32:54,065 INFO  [app]  at
org.rocksdb.NativeLibraryLoader.loadLibraryFromJarToTemp(NativeLibraryLoader.java:118)
09:32:54,065 INFO  [app]  at
org.rocksdb.NativeLibraryLoader.loadLibraryFromJar(NativeLibraryLoader.java:102)
09:32:54,065 INFO  [app]  at
org.rocksdb.NativeLibraryLoader.loadLibrary(NativeLibraryLoader.java:82)
09:32:54,066 INFO  [app]  at org.rocksdb.RocksDB.loadLibrary(RocksDB.java:70)
09:32:54,066 INFO  [app]  at org.rocksdb.RocksDB.<clinit>(RocksDB.java:39)
09:32:54,067 INFO  [app]  ... 10 more
```

1.8.3. Microsoft Windows 上 Kafka Streams 扩展缺少原生库

在 Microsoft Windows 上使用 quarkus-kafka-streams 扩展的应用程序会因为没有原生库 librocksdbjni-win64.dll 而有运行时失败。这个问题会在应用程序启动过程中 没有找到 java.lang.RuntimeException: librocksdbjni-win64.dll in JAR 错误。

此错误可防止成功初始化 RocksDB 组件，这对 Kafka Streams 应用程序至关重要。

临时解决方案：目前还没有可用的临时解决方案。

java.lang.RuntimeException: librocksdbjni-win64.dll 错误

```
13:07:08,118 INFO  [app] ERROR: Failed to start application (with profile [prod])
13:07:08,118 INFO  [app] java.lang.RuntimeException: Failed to start quarkus
13:07:08,118 INFO  [app]  at io.quarkus.runner.ApplicationImpl.doStart(Unknown Source)
13:07:08,118 INFO  [app]  at io.quarkus.runtime.Application.start(Application.java:101)
13:07:08,118 INFO  [app]  at
io.quarkus.runtime.ApplicationLifecycleManager.run(ApplicationLifecycleManager.java:111)
13:07:08,118 INFO  [app]  at io.quarkus.runtime.Quarkus.run(Quarkus.java:71)
13:07:08,118 INFO  [app]  at io.quarkus.runtime.Quarkus.run(Quarkus.java:44)
13:07:08,118 INFO  [app]  at io.quarkus.runtime.Quarkus.run(Quarkus.java:124)
13:07:08,118 INFO  [app]  at io.quarkus.runner.GeneratedMain.main(Unknown Source)
13:07:08,118 INFO  [app] Caused by: java.lang.ExceptionInInitializerError
13:07:08,118 INFO  [app]  at
io.quarkus.kafka.streams.runtime.KafkaStreamsRecorder.loadRocksDb(KafkaStreamsRecorder.java:14
)
13:07:08,118 INFO  [app]  at
io.quarkus.deployment.steps.KafkaStreamsProcessor$loadRocksDb1611413226.deploy_0(Unknown
Source)
13:07:08,118 INFO  [app]  at
io.quarkus.deployment.steps.KafkaStreamsProcessor$loadRocksDb1611413226.deploy(Unknown
Source)
13:07:08,118 INFO  [app]  ... 11 more
13:07:08,118 INFO  [app] Caused by: java.lang.RuntimeException: librocksdbjni-win64.dll was not
found inside JAR.
13:07:08,118 INFO  [app]  at
org.rocksdb.NativeLibraryLoader.loadLibraryFromJarToTemp(NativeLibraryLoader.java:118)
13:07:08,118 INFO  [app]  at
org.rocksdb.NativeLibraryLoader.loadLibraryFromJar(NativeLibraryLoader.java:102)
13:07:08,118 INFO  [app]  at
org.rocksdb.NativeLibraryLoader.loadLibrary(NativeLibraryLoader.java:82)
13:07:08,118 INFO  [app]  at org.rocksdb.RocksDB.loadLibrary(RocksDB.java:70)
13:07:08,118 INFO  [app]  at org.rocksdb.RocksDB.<clinit>(RocksDB.java:39)
13:07:08,118 INFO  [app]  ... 14 more
```

1.8.4. 明确说明原生构建期间缺少 Vert.x 类

在原生构建期间，开发人员可能会获取 Vert.x 类的 java.lang.ClassNotFoundException 错误，如 io.vertx.core.http.impl.Http1xServerResponse 和 io.vertx.core.parsetools.impl.RecordParserImpl。这些错误在构建应用程序时发生，包括使用 quarkus-qpid-jms 扩展的应用程序，而无需包括 Vert.x 作为直接或传输的依赖项。

阐明 quarkus-qpid-jms 不直接使用 Vert.x 非常重要。这个问题来自 quarkus-netty 扩展，quarkus-qpid-jms 使用它。quarkus-netty 扩展负责在原生构建期间为运行时初始化注册这些 Vert.x 类，而无需验证其存在。当构建中没有引入 Vert.x 的扩展时，这会导致注意的例外。

这些 ClassNotFoundException 错误会在构建过程中记录，但不影响应用程序的功能。它们是原生构建过程的结果，以及依赖项在红帽构建的 Quarkus 中处理的方式，特别是通过 quarkus-netty 模块。

java.lang.ClassNotFoundException 错误示例

```
java.lang.ClassNotFoundException: io.vertx.core.http.impl.Http1xServerResponse
 at java.base/jdk.internal.loader.BuiltinClassLoader.loadClass(BuiltinClassLoader.java:641)
 at java.base/jdk.internal.loader.ClassLoaders$AppClassLoader.loadClass(ClassLoaders.java:188)
 at java.base/java.lang.ClassLoader.loadClass(ClassLoader.java:526)
 at
org.graalvm.nativeimage.builder/com.oracle.svm.hosted.NativeImageClassLoader.loadClass(NativeIma
geClassLoader.java:652)
 ... (further stack trace details)
java.lang.ClassNotFoundException: io.vertx.core.parsetools.impl.RecordParserImpl
 at java.base/jdk.internal.loader.BuiltinClassLoader.loadClass(BuiltinClassLoader.java:641)
 at java.base/jdk.internal.loader.ClassLoaders$AppClassLoader.loadClass(ClassLoaders.java:188)
 at java.base/java.lang.ClassLoader.loadClass(ClassLoader.java:526)
 at
org.graalvm.nativeimage.builder/com.oracle.svm.hosted.NativeImageClassLoader.loadClass(NativeIma
geClassLoader.java:652)
 ... (further stack trace details)
```

临时解决方案：要防止这些日志条目并确保正确识别所有依赖项，您可以选择将 quarkus-vertx 扩展添加到项目中。

1.8.5. 在 OpenShift 上测试 JVM 模式中的 AArch64 支持限制

从 Red Hat build of Quarkus 3.2 开始，在带有 AArch64 的 Red Hat OpenShift Container Platform 上进行 JVM 的测试管道会有一些已知的限制：

- AArch64 不支持 Red Hat Serverless。在 AArch64 架构上运行的 OpenShift 集群上支持 Red Hat Serverless 的功能请求在 SRVCOM-2472 中进行跟踪，计划在 Serverless 1.33 中包含支持。
- AArch64 不支持 Red Hat AMQ Streams。因为 AArch64 上还不支持 AMQ Streams，所以此集成的支持还没有被测试。这个问题目前没有在红帽问题管理系统中跟踪。
- AArch64 不支持 Red Hat Single Sign-On。因为 Red Hat Single Sign-On 和 Red Hat build of Keycloak 尚不在 AArch64 上被支持，所以与 Red Hat build of Quarkus 应用程序的集成还没有被测试。
- AArch64 不支持服务绑定。因为 AArch64 还不支持与 Red Hat build of Quarkus 进行服务绑定集成的绑定服务，所以还没有测试此集成。另外，OpenShift Service Binding Operator 在 OpenShift Container Platform (OCP) 4.13 及更高版本中已弃用，计划在以后的 OCP 发行版本中删除。

AArch64 支持仅限于 Red Hat Universal Base Image (UBI)容器，且不会扩展到裸机环境。

临时解决方案：目前还没有可用的临时解决方案。

1.8.6. 对 org.apache.maven:maven:pom:3.6.3 的依赖项可能会导致代理问题

在使用某些 Quarkus 扩展时，可能会解决对 org.apache.maven:maven:pom:3.6.3 的依赖。这不适用于 Gradle 插件，但会影响到任何具有 io.smallrye:smallrye-parent:pom:37 的项目，在其父项目对象模型(POM)层次结构中。对于代理后面的环境，这个依赖关系可能会导致构建失败，以限制对版本 3.6.x 的 org.apache.maven 工件的访问。Maven 3.6.3 中的二进制软件包都作为 Quarkus 核心框架或支持的 Quarkus 扩展的依赖项下载。

临时解决方案：目前还没有可用的临时解决方案。

如需更多信息，请参阅 QUARKUS-1025 - Gradle 插件在 maven core 3.6.x 中拖动。

1.9. 红帽构建的 QUARKUS 3.8.4 的更新

红帽构建的 Quarkus 3.8 提供了更高的稳定性，包括对用户有严重影响的错误的修复。

要获得针对红帽构建的 Quarkus 的最新修复，请确保您使用的是最新可用版本，即 3.8.4。

1.9.1. 程序错误修复

要查看已针对此版本解决的问题列表，请参阅 Red Hat build of Quarkus 3.8.4 bug 修复。

1.9.2. 安全修复

- CVE-2024-2700 io.quarkus/quarkus-core: 将本地配置属性调整到 Quarkus 应用程序
- CVE-2024-29025 io.netty/netty-codec-http: Allocation of resources Without Limits 或 Throttling

1.9.3. 公告

在开始使用和部署红帽构建的 Quarkus 3.8.4 之前，请查看以下与发行版本相关的公告。

- RHSA-2024:2106

1.10. 红帽构建的 QUARKUS 3.8.3 的更新

红帽构建的 Quarkus 3.8 提供了更高的稳定性，包括对用户有严重影响的错误的修复。

要获得针对红帽构建的 Quarkus 的最新修复，请确保您使用的是最新可用版本，即 3.8.4。

1.10.1. 程序错误修复

- QUARKUS-2289 Unable 将文件挂载到 MacOS 上的容器中

- 在使用 Quarkus 的 Dev Mode Live-Reload 功能时，QUARKUS-3206 RabbitMQ TCP 连接将无法关闭

- 在 Vert.x 中使用域套接字时，QUARKUS-3804 Regression in 2.13.9

- QUARKUS-4061 Quarkus maven 插件创建具有非并置依赖项的项目

- 在使用 RHBQ 而不是上游 Quarkus 时，QUARKUS-4065 HTTP 端点会返回空的正文

1.10.2. 公告

在开始使用和部署红帽构建的 Quarkus 3.8.3 之前，请查看以下与发行版本相关的公告。

- RHEA-2024:2057

1.11. 其他资源

- 将应用程序迁移到红帽构建的 Quarkus 3.8 指南。

- 红帽构建的 Quarkus 入门

更新于 2024-05-08