Red Hat build of Quarkus 3.8

配置数据源

Last Updated: 2024-05-18

法律通告

摘要

使用统一配置模型，在红帽构建的 Quarkus 中定义 JDBC 和 Reactive 驱动程序数据源。

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 在红帽构建的 QUARKUS 中配置数据源

使用统一配置模型定义 Java 数据库连接(JDBC)和被动驱动程序的数据源。

应用使用数据源访问关系数据库。Quarkus 提供了一个统一配置模型，用于定义用于 Java 数据库连接(JDBC)和被动数据库驱动程序的数据源。

Quarkus 使用 Agroal 和 Vert.x 为 JDBC 和被动驱动程序提供高性能、可扩展的数据源连接池。**quarkus-jdbc114** 和 **quarkus-reactive114-client** 扩展提供构建时间优化，并将配置的数据源与 security、健康检查和指标等 Quarkus 功能集成。

有关消耗和使用被动数据源的更多信息，请参阅 Quarkus Reactive SQL 客户端 指南。

另外，请参阅 Quarkus Hibernate ORM 指南，以了解有关消耗和使用 JDBC 数据源的信息。

1.1. 开始在 QUARKUS 中配置 数据源

对于熟悉基本权限的用户，本节提供了快速设置数据源的概述和代码示例。

有关示例的更多高级配置，请参阅 参考。

1.1.1. 开发模式中的零配置设置

Quarkus 通过提供 Dev Services 功能来简化数据库配置，为测试或开发(dev)模式启用零配置数据库设置。在 dev 模式中，建议的方法是使用 DevServices，并让 Quarkus 处理数据库，而对于生产环境模式，您可以提供指向 Quarkus 外部管理的数据库配置详情。

要使用 Dev Services，请将所需的数据库类型的适当驱动程序扩展（如 **jdbc-postgresql** ）添加到 **pom.xml** 文件中。在 dev 模式中，如果您不提供任何显式数据库连接详情，Quarkus 会自动处理数据库设置，并提供应用程序和数据库之间的 wiring。

如果您提供用户凭证，则底层数据库将配置为使用它们。如果要使用外部工具连接到数据库，这非常有用。

要使用此功能，请确保安装了 Docker 或 Podman 容器运行时，具体取决于数据库类型。某些数据库，如 H2，在内存模式下操作，不需要容器运行时。

提示

使用 **%prod.** 为 prod 模式提供实际连接详情，以确保它们不会以 dev 模式应用。如需更多信息，请参阅"配置参考"指南中的配置集部分。https://quarkus.io/version/3.8/guides/config-reference#profiles

有关 Dev 服务的更多信息，请参阅 Dev 服务概述。

如需了解更多详细信息和可选配置，请参阅数据库 Dev 服务。

1.1.2. 配置 JDBC 数据源

1. 为您选择的数据库添加正确的 JDBC 扩展。
   - **quarkus-jdbc-db2**
   - **quarkus-jdbc-derby**
   - **quarkus-jdbc-h2**

- **quarkus-jdbc-mariadb**
- **quarkus-jdbc-mssql**
- **quarkus-jdbc-mysql**
- **quarkus-jdbc-oracle**
- **quarkus-jdbc-postgresql**

2. 配置 JDBC 数据源：

```
quarkus.datasource.db-kind=postgresql (1)
quarkus.datasource.username=<your username>
quarkus.datasource.password=<your password>

quarkus.datasource.jdbc.url=jdbc:postgresql://localhost:5432/hibernate_orm_test
quarkus.datasource.jdbc.max-size=16
```

(1) 只有在 classpath 上有多个数据库扩展时才需要此配置值。

如果只有一个可行的扩展可用，则 Quarkus 会假定这是正确的扩展。当您向测试范围中添加驱动程序时，Quarkus 会自动在测试中包含指定的驱动程序。

1.1.2.1. JDBC 连接池大小调整

为了保护您的数据库在负载高峰期间被过载，请适当调整池的大小，以节流数据库负载。最佳池大小取决于许多因素，如并行应用程序用户的数量或工作负载的性质。

请注意，设置池大小太大可能会导致一些请求在等待连接时超时。

有关池大小调整属性的更多信息，请参阅 JDBC 配置参考部分。

1.1.3. 配置被动数据源

1. 为您选择的数据库添加正确的被动扩展。
   - **quarkus-reactive-db2-client**
   - **quarkus-reactive-mssql-client**
   - **quarkus-reactive-mysql-client**
   - **quarkus-reactive-oracle-client**
   - **quarkus-reactive-pg-client**
2. 配置被动数据源：

```
quarkus.datasource.db-kind=postgresql (1)
quarkus.datasource.username=<your username>
quarkus.datasource.password=<your password>

quarkus.datasource.reactive.url=postgresql:///your_database
quarkus.datasource.reactive.max-size=20
```

1 只有在 classpath 中有多个 Reactive 驱动程序扩展时才需要此配置值。

1.2. 配置数据源

以下章节描述了单个或多个数据源的配置。为了简单起见，我们将引用单个数据源作为默认数据源（未命名）数据源。

1.2.1. 配置单个数据源

数据源可以是 JDBC 数据源、被动或两者。这取决于配置和选择项目扩展。

1. 使用以下配置属性定义数据源，其中 **db-kind** 定义要连接到哪个数据库平台，例如 **h2** ：

```
quarkus.datasource.db-kind=h2
```

Quarkus 会从 **db-kind** 数据库平台属性的指定的值减少需要使用的 JDBC 驱动程序类。

注意

只有在应用程序依赖于多个数据库驱动程序时，才需要执行此步骤。如果应用程序使用单个驱动程序运行，则会自动检测到此驱动程序。

Quarkus 当前包括以下内置数据库类型：

- DB2: **db2**
- Derby: **derby**
- H2: **h2**
- mariadb: **mariadb**
- Microsoft SQL Server: **mssql**
- MySQL: **mysql**
- Oracle: **oracle**
- postgresql: **postgresql**,**pgsql** 或 **pg**
- 要使用不是内置的数据库类型，请使用 **other** 并明确定义 JDBC 驱动程序

注意

您可以在 JVM 模式中使用 Quarkus 应用程序中的任何 JDBC 驱动程序，如使用其他数据库 中所述。但是，当将应用程序编译到原生可执行文件时，使用非内置数据库类型不太可能可以正常工作。

对于原生可执行构建，建议使用可用的 JDBC Quarkus 扩展，或为您的特定驱动程序提供自定义扩展。

2. 配置以下属性以定义凭证：

```
quarkus.datasource.username=<your username>
quarkus.datasource.password=<your password>
```

您还可以使用数据源 的凭证供应商 从 Vault 检索密码。

到目前为止，无论您使用的是 JDBC 还是被动驱动程序，配置都相同。当您定义了数据库类型和凭证时，其余则取决于您所使用的驱动程序类型。可以同时使用 JDBC 和被动驱动程序。

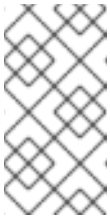
1.2.1.1. JDBC 数据源

JDBC 是最常见的数据库连接模式，通常与非主动 Hibernate ORM 一起使用时需要。

1. 要使用 JDBC 数据源，请从添加所需的依赖项开始：

   a. 对于内置 JDBC 驱动程序，请从下面的列表中选择并添加相关数据库驱动程序的 Quarkus 扩展：

   - Derby - **quarkus-jdbc-derby**
   - H2 - **quarkus-jdbc-h2**

   注意

   H2 和 Derby 数据库可以配置为以"embedded 模式"运行，但 Derby 扩展不支持将嵌入式数据库引擎编译到原生可执行文件中。

   使用内存数据库读取测试，以了解与集成测试相关的建议。

   - DB2 - **quarkus-jdbc-db2**
   - MariaDB - **quarkus-jdbc-mariadb**
   - Microsoft SQL Server - **quarkus-jdbc-mssql**
   - MySQL - **quarkus-jdbc-mysql**
   - Oracle - **quarkus-jdbc-oracle**
   - PostgreSQL - **quarkus-jdbc-postgresql**
   - 其他 JDBC 扩展（如 SQLite 及其 文档 ）可在 Quarkiverse 中找到。
     例如，添加 PostgreSQL 驱动程序依赖项：

     ```
     ./mvnw quarkus:add-extension -Dextensions="jdbc-postgresql"
     ```

     注意

     使用内置 JDBC 驱动程序扩展会自动包含 Agroal 扩展，该扩展是适用于自定义和内置 JDBC 驱动程序的 JDBC 连接池实施。但是，对于自定义驱动程序，需要明确添加 Agroal。

   b. 对于自定义 JDBC 驱动程序，请将 **quarkus-agroal** 依赖项添加到项目中，以及您的相关数据库驱动程序的扩展：

```
./mvnw quarkus:add-extension -Dextensions="agroal"
```

要将 JDBC 驱动程序用于其他 数据库，请使用没有内置扩展或具有不同驱动程序 的数据库。

2. 通过定义 JDBC URL 属性来配置 JDBC 连接：

```
quarkus.datasource.jdbc.url=jdbc:postgresql://localhost:5432/hibernate_orm_test
```

注意

注意属性名称中的 **jdbc** 前缀。特定于 JDBC 的所有配置属性都具有 **jdbc** 前缀。对于被动数据源，前缀是 **被动**。

有关配置 JDBC 的更多信息，请参阅 JDBC URL 格式参考 和 Quarkus 扩展和数据库驱动程序参考。

1.2.1.1.1. 自定义数据库和驱动程序

如果您需要连接到 Quarkus 没有提供 JDBC 驱动程序的扩展数据库，您可以使用自定义驱动程序。例如，如果您在项目中使用 OpenTracing JDBC 驱动程序。

如果没有扩展，驱动程序将在 JVM 模式下运行的任何 Quarkus 应用程序中正常工作。但是，当将应用程序编译到原生可执行文件时，驱动程序不太可能工作。如果您计划生成原生可执行文件，请使用现有的 JDBC Quarkus 扩展，或为您的驱动程序贡献一个。

警告

OpenTracing 已被弃用，而被 OpenTelemetry 替代。如需追踪信息，请检查有关 数据源追踪的相关部分，bellow。

带有旧 **OpenTracing** 驱动程序的自定义驱动程序定义示例：

```
quarkus.datasource.jdbc.driver=io.opentracing.contrib.jdbc.TracingDriver
```

定义在 **JVM** 模式中不支持内置支持的数据库的访问示例：

```
quarkus.datasource.db-kind=other
quarkus.datasource.jdbc.driver=oracle.jdbc.driver.OracleDriver
quarkus.datasource.jdbc.url=jdbc:oracle:thin:@192.168.1.12:1521/ORCL_SVC
quarkus.datasource.username=scott
quarkus.datasource.password=tiger
```

有关 JDBC 配置选项和配置其他方面的所有详细信息，如连接池大小，请参阅 JDBC 配置参考部分。

1.2.1.1.2. 消耗数据源

使用 Hibernate ORM 时，Hibernate 层会自动获取数据源并使用它。

对于对数据源的 in-code 访问，请作为任何其他 bean 获取，如下所示：

```
@Inject
AgroalDataSource defaultDataSource;
```

在上例中，type 是 **AgroalDataSource**，一个 **javax.sql.DataSource** 子类型。因此，您也可以使用 **javax.sql.DataSource** 作为注入的类型。

1.2.1.2. 主动数据源

Quarkus 提供多个被动客户端，用于被动数据源。

1. 在应用程序中添加对应的扩展：
   - DB2: **quarkus-reactive-db2-client**
   - mariadb/MySQL: **quarkus-reactive-mysql-client**
   - Microsoft SQL Server: **quarkus-reactive-mssql-client**
   - Oracle: **quarkus-reactive-oracle-client**
   - postgresql: **quarkus-reactive-pg-client**
     安装的扩展必须与您在数据源配置中定义的 **quarkus.datasource.db-kind** 一致。
2. 添加驱动程序后，配置连接 URL，并为您的连接池定义正确的大小。

```
quarkus.datasource.reactive.url=postgresql:///your_database
quarkus.datasource.reactive.max-size=20
```

1.2.1.2.1. 重新调整连接池大小

为了保护您的数据库在负载高峰期间被过载，请适当调整池的大小，以节流数据库负载。正确的大小总是取决于许多因素，如并行应用程序用户的数量或工作负载的性质。

请注意，设置池大小太大可能会导致一些请求在等待连接时超时。

有关池大小调整属性的更多信息，请参阅 主动数据源配置参考部分。

1.2.1.3. JDBC 和被动数据源同时

当 JDBC 扩展 - 以及 Agroal - 以及处理给定数据库类型的被动数据源扩展时，将默认创建这两个数据库。

- 明确禁用 JDBC 数据源：

```
quarkus.datasource.jdbc=false
```

- 明确禁用被动数据源：

```
quarkus.datasource.reactive=false
```

提示

在大多数情况下，上述配置将作为 JDBC 驱动程序或被动数据源扩展是可选的，而不是同时存在。

1.2.2. 配置多个数据源

注意

Hibernate ORM 扩展支持使用配置属性定义 持久性单元。对于每个持久性单元，指向您选择的数据源。

定义多个数据源的工作方式类似于定义一个数据源，具有一个重要更改 - 您必须为每个数据源指定一个名称（配置属性）。

以下示例提供三个不同的数据源：

- 默认 1
- 名为 **users**的数据源
- 名为 **inventory**的数据源

每个配置都有其配置：

```
quarkus.datasource.db-kind=h2
quarkus.datasource.username=username-default
quarkus.datasource.jdbc.url=jdbc:h2:mem:default
quarkus.datasource.jdbc.max-size=13

quarkus.datasource.users.db-kind=h2
quarkus.datasource.users.username=username1
quarkus.datasource.users.jdbc.url=jdbc:h2:mem:users
quarkus.datasource.users.jdbc.max-size=11

quarkus.datasource.inventory.db-kind=h2
quarkus.datasource.inventory.username=username2
quarkus.datasource.inventory.jdbc.url=jdbc:h2:mem:inventory
quarkus.datasource.inventory.jdbc.max-size=12
```

请注意，配置属性中有一个额外的部分。语法如下： **quarkus.datasource.[optional name.][datasource property]**.

注意

即使只安装了一个数据库扩展，命名的数据库还需要指定一个 build-time 属性，以便 Quarkus 能够检测它们。通常，这是 **db-kind** 属性，但您也可以指定 Dev Services 属性，以根据 Dev Services for Databases 指南创建名为 datasources。

1.2.2.1. 命名数据源注入

使用多个数据源时，每个 **DataSource** 也具有 **io.quarkus.agroal.DataSource** 限定符，其名称为数据源的名称。

通过使用上一节中提到的属性来配置三个不同的数据源，请按如下所示注入每个数据源：

```
@Inject
AgroalDataSource defaultDataSource;
```

```
@Inject
@DataSource("users")
AgroalDataSource usersDataSource;

@Inject
@DataSource("inventory")
AgroalDataSource inventoryDataSource;
```

1.2.3. 激活/停用数据源

如果在构建时配置了数据源，默认情况下，它在运行时处于活跃状态，即 Quarkus 将在应用启动时启动对应的 JDBC 连接池或重新主动客户端。

要在运行时取消激活数据源，请将 **quarkus.datasource[.optional name].active** 设置为 **false**。然后 Quarkus 不会在应用启动时启动对应的 JDBC 连接池或被动客户端。在运行时使用对应的数据源的任何尝试都会失败，并显示清晰的错误消息。

当您希望应用程序在运行时使用预先确定的数据源集合时，这特别有用。

警告

如果另一个 Quarkus 扩展依赖于不活跃的数据源，则该扩展可能无法启动。

在这种情况下，您还需要取消激活其他扩展。例如，请参见此处的 Hibernate ORM。

例如，使用以下配置：

```
quarkus.datasource."pg".db-kind=postgres
quarkus.datasource."pg".active=false
quarkus.datasource."pg".jdbc.url=jdbc:postgresql:///your_database

quarkus.datasource."oracle".db-kind=oracle
quarkus.datasource."oracle".active=false
quarkus.datasource."oracle".jdbc.url=jdbc:oracle:///your_database
```

在运行时设置 **quarkus.datasource."pg".active=true** 将使 PostgreSQL 数据源可用，并在运行时 设置 **quarkus.datasource."oracle".active=true** 将使 Oracle 数据源可用。

提示

自定义配置文件 可帮助简化此类设置。通过将以下特定于配置集的配置附加到以上内容，只需通过 设置 **quarkus.profile**: **quarkus.profile** =prod,pg 或 **quarkus.profile=prod,oracle** 即可在运行时选择持久性单元/数据源。

```
%pg.quarkus.hibernate-orm."pg".active=true
%pg.quarkus.datasource."pg".active=true
# Add any pg-related runtime configuration here, prefixed with "%pg."

%oracle.quarkus.hibernate-orm."oracle".active=true
%oracle.quarkus.datasource."oracle".active=true
# Add any pg-related runtime configuration here, prefixed with "%pg."
```

提示

它也可用于定义 CDI bean 生成者 重定向到当前活跃的数据源，如下所示：

```
public class MyProducer {
    @Inject
    DataSourceSupport dataSourceSupport;

    @Inject
    @DataSource("pg")
    AgroalDataSource pgDataSourceBean;

    @Inject
    @DataSource("oracle")
    AgroalDataSource oracleDataSourceBean;

    @Produces
    @ApplicationScoped
    public AgroalDataSource dataSource() {
        if (dataSourceSupport.getInactiveNames().contains("pg")) {
            return oracleDataSourceBean;
        } else {
            return pgDataSourceBean;
        }
    }
}
```

1.3. 数据源集成

1.3.1. 数据源健康检查

如果您使用 **quarkus-smallrye-health** 扩展，**quarkus-agroal** 和 reactive 客户端扩展会自动添加就绪度健康检查来验证数据源。

当您默认访问应用程序的健康状况就绪端点 **/q/health/ready** 时，您会收到数据源验证状态的信息。如果您有多个数据源，则会检查所有数据源验证失败，如果发生单一数据源验证失败，则状态将变为 **DOWN**。

可以使用 **quarkus.datasource.health.enabled** 属性禁用此行为。

要只从健康检查中排除特定的数据源，请使用：

```
quarkus.datasource."datasource-name".health-exclude=true
```

1.3.2. 数据源指标

如果您使用 **quarkus-micrometer** 或 **quarkus-smallrye-metrics** 扩展，**quarkus-agroal** 可以为指标 registry 提供一些与数据源相关的指标。这可以通过将 **quarkus.datasource.metrics.enabled** 属性设置为 **true** 来激活。

要使公开的指标包含任何实际值，则必须通过 Agroal 机制在内部启用指标集合。默认情况下，当指标扩展存在时，为所有数据源启用此指标集合机制，并且启用了 Agroal 扩展的指标。

要禁用特定数据源的指标，请将 **quarkus.datasource.jdbc.enable-metrics** 设置为 **false**，或为命名数据源应用 **quarkus.datasource.<datasource name>.jdbc.enable-metrics**。这将禁用收集指标，并在 **/q/metrics** 端点中公开它们（如果禁用了收集它们的机制）。

相反，将 **quarkus.datasource.jdbc.enable-metrics** 设置为 **true**，或 **quarkus.datasource.<datasource name>.jdbc.enable-metrics** 用于一个命名的数据源，即使指标扩展没有被使用。如果您需要以编程方式访问收集的指标，这非常有用。在注入的 **AgroalDataSource** 实例上调用 **dataSource.getMetrics （）** 后可用。

如果禁用了此数据源的指标集合，则所有值都会为零。

1.3.3. 数据源追踪

要将追踪与数据源搭配使用，您需要为项目添加 **quarkus-opentelemetry** 扩展。

您不需要声明不同的驱动程序，因为您需要追踪。如果使用 JDBC 驱动程序，您需要按照 此处 OpenTelemetry 扩展中的说明进行操作。

即使所有追踪基础架构都默认不启用数据源追踪，而且您需要通过设置此属性来启用它：

```
# enable tracing
quarkus.datasource.jdbc.telemetry=true
```

1.3.4. Narayana 事务管理器集成

如果 Narayana JTA 扩展也可用，则会自动集成。

您可以通过设置 **transactions** 配置属性来覆盖它：

- **quarkus.datasource.jdbc.transactions** 用于默认的未命名数据源
- **Quarkus.datasource. &*lt;datasource-name&* gt; .jdbc.transactions** for named datasource

如需更多信息，请参阅下面的 配置参考部分。

要使用 JDBC 促进数据库中存储事务日志，请参阅 使用 Quarkus 指南中的将事务日志配置为存储在使用事务 的数据源 部分中。

1.3.4.1. 命名数据源

使用 Dev Services 时，将始终创建默认数据源，但要指定命名数据源，您需要至少有一个构建时间属性，以便 Quarkus 能够检测如何创建数据源。

您通常会指定 **db-kind** 属性，或通过设置 **quarkus.datasource."name".devservices.enabled=true** 来显式启用 Dev Services。

1.3.5. 使用内存数据库进行测试

H2 和 Derby 等一些 *数据库* 通常用作快速运行集成测试的工具。

推荐的方法是使用您要在生产环境中使用的实际数据库，特别是 Dev Services 为测试提供零配置数据库，并对容器运行测试相对快速，并在实际环境中生成预期结果。但是，在需要运行简单集成测试时，也可以使用 JVM 支持的数据库。

1.3.5.1. 支持和限制

嵌入式数据库(H2 和 Derby)在 JVM 模式中工作。对于原生模式，会有以下限制：

- Derby 无法嵌入到原生模式的应用程序中。但是，Quarkus Derby 扩展允许对 Derby JDBC **客户端** 进行原生编译，支持 **远程连接**。
- 不建议将 H2 嵌入到您的原生镜像中。考虑使用替代方法，例如改为使用远程连接单独的数据库。

1.3.5.2. 运行集成测试

1. 添加对工件的依赖，提供以下 Maven 协调下的额外工具：
   - **io.quarkus:quarkus-test-h2** for H2
   - **io.quarkus:quarkus-test-derby** for Derby
     这将允许您测试应用，即使它被编译到原生可执行文件中，而数据库将作为 JVM 进程运行。
2. 在集成测试中任何类添加以下特定注解，以便在 JVM 或原生可执行文件中运行集成测试：
   - **@QuarkusTestResource(H2DatabaseTestResource.class)**
   - **@QuarkusTestResource(DerbyDatabaseTestResource.class)**
     这样可确保测试套件根据测试执行所需的单独进程启动并终止受管数据库。

     H2 示例

     ```
     package my.app.integrationtests.db;

     import io.quarkus.test.common.QuarkusTestResource;
     import io.quarkus.test.h2.H2DatabaseTestResource;

     @QuarkusTestResource(H2DatabaseTestResource.class)
     public class TestResources {
     }
     ```
3. 配置与受管数据库的连接：

   ```
   quarkus.datasource.db-kind=h2
   quarkus.datasource.jdbc.url=jdbc:h2:tcp://localhost/mem:test
   ```

1.4. 参考

1.4.1. 常见数据源配置参考

构建时修复的配置属性 - 所有其他配置属性可在运行时覆盖

Configuration 属性	类型	default
quarkus.datasource.health.enabled 如果存在 smallrye-health 扩展，是否发布健康检查。 这是一个全局设置，不特定于数据源。 环境变量：**QUARKUS_DATASOURCE_HEALTH_ENABLED**	布尔值	**true**
quarkus.datasource.metrics.enabled 如果存在指标扩展，是否发布数据源指标。 这是一个全局设置，不特定于数据源。 注意 这与需要在 JDBC 数据源级别上设置的 "jdbc.enable-metrics" 属性不同，以便为该数据源启用指标集合。 环境变量：**QUARKUS_DATASOURCE_METRICS_ENABLED**	布尔值	**false**
quarkus.datasource.db-kind **quarkus.datasource."datasource-name".db-kind** 我们将连接到的数据库类型（例如 h2、postgresql...）。 环境变量：**QUARKUS_DATASOURCE_DB_KIND**	string	

quarkus.datasource.db-version **quarkus.datasource."datasource-name".db-version** 我们将连接到的数据库版本（例如'10.0'）. 小心 此处设置的版本号应该遵循与数据库 JDBC 驱动程序的 **java.sql.DatabaseMetaData="getDatabaseProductVersion （）** 返回的字符串相同的编号方案。此编号方案可能与您的数据库最流行的方案不同；例如，Microsoft SQL Server 2016 版本为 **13**。 作为规则，这里设置的版本应尽可能高，但必须小于或等于应用程序要连接到的任何数据库的版本。 高版本将提高性能和使用更多功能（例如 Hibernate ORM 可能会生成效率更高的 SQL，避免临时解决方案并利用更多数据库功能），但如果它比您要连接的数据库版本高，则可能会导致运行时异常（例如 Hibernate ORM 可能会生成无效的 SQL，以便您数据库将拒绝）。 某些扩展（如 Hibernate ORM 扩展）将尝试在启动时针对实际数据库版本检查这个版本，从而导致实际版本较低或只是在数据库无法访问时发出警告。 此属性的默认值特定于每个扩展；Hibernate ORM 扩展将默认为它支持的最旧的版本。 环境变量： **QUARKUS_DATASOURCE_DB_VERSION**	string	
quarkus.datasource.devservices.enabled **quarkus.datasource."datasource-name".devservices.enabled** 此 Dev Service 是否应该以 dev 模式为应用程序启动，还是测试。 除非明确设置了 JDBC URL 或被动客户端 URL，否则 dev Services 会被默认启用。 环境变量： **QUARKUS_DATASOURCE_DEVSERVICES_ENABLED**	布尔值	
quarkus.datasource.devservices.image-name **quarkus.datasource."datasource-name".devservices.image-name** 基于容器的 Dev Service 供应商的容器镜像名称。 如果提供程序不是基于容器的数据库，如 H2 或 Derby，则这不起作用。 环境变量： **QUARKUS_DATASOURCE_DEVSERVICES_IMAGE_NAME**	string	

quarkus.datasource.devservices.container-env quarkus.datasource."datasource-name".devservices.container-env 传递给容器的环境变量。 环境变量： **QUARKUS_DATASOURCE_DEVSERVICES_CONTAINER_ENV**	**Map<String,String>**	
quarkus.datasource.devservices.container-properties quarkus.datasource."datasource-name".devservices.container-properties 为其他容器配置传递的通用属性。 此处定义的属性是特定于数据库的属性，特别是在每个数据库开发服务实施中。 环境变量： **QUARKUS_DATASOURCE_DEVSERVICES_CONTAINER_PROPERTIES**	**Map<String,String>**	
quarkus.datasource.devservices.properties quarkus.datasource."datasource-name".devservices.properties 添加到数据库连接 URL 的通用属性。 环境变量： **QUARKUS_DATASOURCE_DEVSERVICES_PROPERTIES**	**Map<String,String>**	
quarkus.datasource.devservices.port quarkus.datasource."datasource-name".devservices.port dev 服务将侦听的可选固定端口。 如果没有定义，则会随机选择端口。 环境变量： **QUARKUS_DATASOURCE_DEVSERVICES_PORT**	int	
quarkus.datasource.devservices.command quarkus.datasource."datasource-name".devservices.command 用于基于容器的 Dev Service 供应商的容器 start 命令。 如果提供程序不是基于容器的数据库，如 H2 或 Derby，则这不起作用。 环境变量： **QUARKUS_DATASOURCE_DEVSERVICES_COMMAND**	string	

quarkus.datasource.devservices.db-name **quarkus.datasource."datasource-name".devservices.db-name** 如果此 Dev 服务支持覆盖它，则使用数据库名称。 环境变量： **QUARKUS_DATASOURCE_DEVSERVICES_DB_NAME**	string	
quarkus.datasource.devservices.username **quarkus.datasource."datasource-name".devservices.username** 如果此 Dev Service 支持覆盖它，则使用的用户名。 环境变量： **QUARKUS_DATASOURCE_DEVSERVICES_USERNAME**	string	
quarkus.datasource.devservices.password **quarkus.datasource."datasource-name".devservices.password** 如果此 Dev 服务支持覆盖它，则使用密码。 环境变量： **QUARKUS_DATASOURCE_DEVSERVICES_PASSWORD**	string	
quarkus.datasource.devservices.init-script-path **quarkus.datasource."datasource-name".devservices.init-script-path** 要从 classpath 加载的 SQL 脚本的路径，并应用到 Dev Service 数据库。 如果提供程序不是基于容器的数据库，如 H2 或 Derby，则这不起作用。 环境变量： **QUARKUS_DATASOURCE_DEVSERVICES_INIT_SCRIPT_PATH**	string	
quarkus.datasource.devservices.volumes **quarkus.datasource."datasource-name".devservices.volumes** 要映射到容器的卷。 map 键对应于主机位置；映射值是容器位置。如果主机位置以 "classpath:" 开头，则映射会从具有只读权限的 classpath 中加载资源。 使用文件系统位置时，将生成具有读写权限的卷，可能会导致文件系统中的数据丢失或修改。 如果提供程序不是基于容器的数据库，如 H2 或 Derby，则这不起作用。 环境变量： **QUARKUS_DATASOURCE_DEVSERVICES_VOLUMES**	**Map<String,String>**	

quarkus.datasource.devservices.reuse **quarkus.datasource."datasource-name".devservices.reuse** 在 **dev 模式会话或测试套件执行后，** 是否保持 Dev Service 容器运行，以便在下一个 dev 模式会话或测试套件执行中重复使用它们。 在 dev 模式会话或测试套件执行中，如果它们的配置（用户名、密码、环境、端口绑定、...）没有改变，则 Quarkus 将始终重复使用 Dev Services。此功能专用于在 **Quarkus 未运行时保持容器运行**，以便在运行之间重复使用容器。 **警告** 这个功能需要在 **testcontainers.properties** 中明确启用，可能需要更改您在 dev 模式中配置数据初始化，并可能会无限期地运行容器，强制您停止并手动删除它们。如需更多信息，请参阅文档中的 此部分。 默认情况下，此配置属性设为 **true**，因此如果它在 **testcontainers.properties** 中启用它，但只想将其用于某些 Quarkus 应用程序或数据源，则最好 **禁用** 重复使用。 环境变量：**QUARKUS_DATASOURCE_DEVSERVICES_REUSE**	布尔值	**true**
quarkus.datasource.health-exclude **quarkus.datasource."datasource-name".health-exclude** 如果启用了数据源的常规健康检查，是否应从健康检查中排除此特定数据源。 默认情况下，健康检查包含所有配置的数据源（如果已启用）。 环境变量：**QUARKUS_DATASOURCE_HEALTH_EXCLUDE**	布尔值	**false**
quarkus.datasource.active **quarkus.datasource."datasource-name".active** 此数据源是否应该在运行时处于活动状态。 请参阅文档中的此部分。 如果数据源未激活，则不会从应用程序开始，访问对应的 Datasource CDI bean 将失败，这意味着此数据源的特定消费者（如 Hibernate ORM 持久性单元）将无法启动，除非它们也不活跃。 环境变量：**QUARKUS_DATASOURCE_ACTIVE**	布尔值	**true**

quarkus.datasource.username **quarkus.datasource."datasource-name".username** 数据源用户名 环境变量： **QUARKUS_DATASOURCE_USERNAME**	string	
quarkus.datasource.password **quarkus.datasource."datasource-name".password** 数据源密码 环境变量： **QUARKUS_DATASOURCE_PASSWORD**	string	
quarkus.datasource.credentials-provider **quarkus.datasource."datasource-name".credentials-provider** 凭证供应商名称 环境变量： **QUARKUS_DATASOURCE_CREDENTIALS_PROVIDER**	string	
quarkus.datasource.credentials-provider-name **quarkus.datasource."datasource-name".credentials-provider-name** 凭据 provider bean 名称。 这是实施 **CredentialsProvider** 的 bean 名称（如**@Named**中）。它可用于在多个存在时选择凭据提供程序 bean。当只有一个凭证供应商可用时，这不需要。 对于 Vault，凭据提供程序 bean 名称为 **vault-credentials-provider**。 环境变量： **QUARKUS_DATASOURCE_CREDENTIALS_PROVIDER_NAME**	string	

1.4.2. JDBC 配置参考

构建时修复的配置属性 - 所有其他配置属性可在运行时覆盖

Configuration 属性	类型	default
quarkus.datasource.jdbc **quarkus.datasource."datasource-name".jdbc** 如果我们为此数据源创建 JDBC 数据源。 环境变量： **QUARKUS_DATASOURCE_ JDBC**	布尔值	**true**

quarkus.datasource.jdbc.driver quarkus.datasource."datasource-name".jdbc.driver 数据源驱动程序类名称 环境变量： **QUARKUS_DATASOURCE_ JDBC_DRIVER**	string	
quarkus.datasource.jdbc.transactions quarkus.datasource."datasource-name".jdbc.transactions 是否要使用常规 JDBC 事务、XA 或禁用所有事务功能。 启用 XA 时，您将需要实施 **javax.sql.XADataSource** 的驱动程序。 环境变量： **QUARKUS_DATASOURCE_ JDBC_TRANSACTIONS**	**enabled**,**xa**, **disabled**	**enabled**
quarkus.datasource.jdbc.enable-metrics quarkus.datasource."datasource-name".jdbc.enable-metrics 启用数据源指标集合。如果未指定，如果指标扩展处于活跃状态，则默认启用收集指标。 环境变量： **QUARKUS_DATASOURCE_ENABLE_METRICS**	布尔值	
quarkus.datasource.jdbc.tracing quarkus.datasource."datasource-name".jdbc.tracing 启用 JDBC 追踪。默认禁用此选项。 环境变量： **QUARKUS_DATASOURCE_ JDBC_TRACING**	布尔值	**false**
quarkus.datasource.jdbc.telemetry quarkus.datasource."datasource-name".jdbc.telemetry 启用 OpenTelemetry JDBC 检测。 环境变量： **QUARKUS_DATASOURCE_ JDBC_TELEMETRY**	布尔值	**false**
quarkus.datasource.jdbc.url quarkus.datasource."datasource-name".jdbc.url 数据源 URL 环境变量： **QUARKUS_DATASOURCE_ branch_URL**	string	

quarkus.datasource.jdbc.initial-size **quarkus.datasource."datasource-name".jdbc.initial-size** 池的初始大小。通常，您希望将初始大小设置为至少与最小大小匹配，但不强制这样做，以便允许在启动时具有延迟初始化连接的架构，同时可以在启动后保持最小池大小。 环境变量：**QUARKUS_DATASOURCE_ JDBC_INITIAL_SIZE**	int	
quarkus.datasource.jdbc.min-size **quarkus.datasource."datasource-name".jdbc.min-size** 数据源池最小大小 环境变量：**QUARKUS_DATASOURCE_ JDBC_MIN_SIZE**	int	**0**
quarkus.datasource.jdbc.max-size **quarkus.datasource."datasource-name".jdbc.max-size** 数据源池最大大小 环境变量：**QUARKUS_DATASOURCE_ JDBC_MAX_SIZE**	int	**20**
quarkus.datasource.jdbc.background-validation-interval **quarkus.datasource."datasource-name".jdbc.background-validation-interval** 我们在后台验证闲置连接的间隔。 设置为 **0** 可禁用后台验证。 环境变量： **QUARKUS_DATASOURCE_BACKGROUND_VALIDATION_INTERVAL**	duration	**2M**
quarkus.datasource.jdbc.foreground-validation-interval **quarkus.datasource."datasource-name".jdbc.foreground-validation-interval** 在闲置的时间超过指定间隔的连接上执行前台验证。 环境变量： **QUARKUS_DATASOURCE_FOREGROUND_VALIDATION_INTERVAL**	duration	
quarkus.datasource.jdbc.acquisition-timeout **quarkus.datasource."datasource-name".jdbc.acquisition-timeout** 取消获取新连接前的超时时间 环境变量：**QUARKUS_DATASOURCE_ JDBC_ACQUISITION_TIMEOUT**	duration	**5S**

quarkus.datasource.jdbc.leak-detection-interval quarkus.datasource."datasource-name".jdbc.leak-detection-interval 我们检查连接泄漏的时间间隔。 环境变量：**QUARKUS_DATASOURCE_ JDBC_LEAK_DETECTION_INTERVAL**	duration	**此功能默认为禁用。**
quarkus.datasource.jdbc.idle-removal-interval quarkus.datasource."datasource-name".jdbc.idle-removal-interval 我们尝试删除闲置连接的间隔。 环境变量：**QUARKUS_DATASOURCE_ JDBC_IDLE_REMOVAL_INTERVAL**	duration	**5M**
quarkus.datasource.jdbc.max-lifetime quarkus.datasource."datasource-name".jdbc.max-lifetime 连接的最大生命周期。 环境变量：**QUARKUS_DATASOURCE_ JDBC_MAX_LIFETIME**	duration	**默认情况下，连接的生命周期没有限制。**
quarkus.datasource.jdbc.transaction-isolation-level quarkus.datasource."datasource-name".jdbc.transaction-isolation-level 事务隔离级别。 环境变量：**QUARKUS_DATASOURCE_ JDBC_TRANSACTION_ISOLATION_LEVEL**	**undefined,none,read-uncommitted,read-committed,repeatable-read,serializable**	
quarkus.datasource.jdbc.extended-leak-report quarkus.datasource."datasource-name".jdbc.extended-leak-report 收集并显示有关泄漏连接的额外故障排除信息。 环境变量：**QUARKUS_DATASOURCE_EXTENDED_LEAK_REPORT**	布尔值	**false**

quarkus.datasource.jdbc.flush-on-close **quarkus.datasource."datasource-name".jdbc.flush-on-close** 允许在返回到池时清空连接。默认情况下不启用它。 环境变量：**QUARKUS_DATASOURCE_ JDBC_FLUSH_ON_CLOSE**	布尔值	**false**
quarkus.datasource.jdbc.detect-statement-leaks **quarkus.datasource."datasource-name".jdbc.detect-statement-leaks** 启用后，当连接返回池时，如果没有关闭所有 open 语句的应用程序，则 Agroal 将能够生成警告。这与跟踪开放连接无关。禁用峰值性能，但仅在有高信任时不会发生泄漏。 环境变量： **QUARKUS_DATASOURCE_JDBC_DETECT_STATEMENT_LEAKS**	布尔值	**true**
quarkus.datasource.jdbc.new-connection-sql **quarkus.datasource."datasource-name".jdbc.new-connection-sql** 首次使用连接时执行的查询。 环境变量：**QUARKUS_DATASOURCE_ JDBC_NEW_CONNECTION_SQL**	string	
quarkus.datasource.jdbc.validation-query-sql **quarkus.datasource."datasource-name".jdbc.validation-query-sql** 执行查询以验证连接。 环境变量：**QUARKUS_DATASOURCE_ JDBC_VALIDATION_QUERY_SQL**	string	
quarkus.datasource.jdbc.pooling-enabled **quarkus.datasource."datasource-name".jdbc.pooling-enabled** 禁用池以防止重复使用连接。当外部池管理 Connections 的生命周期时，请使用此选项。 环境变量：**QUARKUS_DATASOURCE_ JDBC_POOLING_ENABLED**	布尔值	**true**
quarkus.datasource.jdbc.transaction-requirement **quarkus.datasource."datasource-name".jdbc.transaction-requirement** 在获取连接时需要活跃的事务。建议用于生产环境。警告：有些扩展会获取连接，而不会为架构更新和架构验证等操作留出一个事务。将此设置设置为 STRICT 可能会导致在这种情况下失败。 环境变量： **QUARKUS_DATASOURCE_PRESS_TRANSACTION_REQUIREMENT**	**off,warn,strict**	

quarkus.datasource.jdbc.additional-jdbc-properties **quarkus.datasource."datasource-name".jdbc.additional-jdbc-properties** 创建新连接时要传递给 JDBC 驱动程序的其他未指定属性。 环境变量：**QUARKUS_DATASOURCE_ JDBC_ADDITIONAL_ JDBC_PROPERTIES**	**Map< String ,Strin g>**	
quarkus.datasource.jdbc.tracing.enabled **quarkus.datasource."datasource-name".jdbc.tracing.enabled** 启用 JDBC 追踪。 环境变量：**QUARKUS_DATASOURCE_ JDBC_TRACING_ENABLED**	布尔值	**如果 quark us.dat asour ce.jdb c.traci ng=fal se 和 true，** 则为 **false （如果 quark us.dat asour ce.jdb c.traci ng=tr ue）**
quarkus.datasource.jdbc.tracing.trace-with-active-span-only **quarkus.datasource."datasource-name".jdbc.tracing.trace-with-active-span-only** 仅跟踪带有活跃 Span 的调用 环境变量：**QUARKUS_DATASOURCE_ JDBC_TRACING_TRACE_WITH_ACTIVE_SPAN_ONLY**	布尔值	**false**
quarkus.datasource.jdbc.tracing.ignore-for-tracing **quarkus.datasource."datasource-name".jdbc.tracing.ignore-for-tracing** 忽略要跟踪的特定查询 环境变量：**QUARKUS_DATASOURCE_TRACING_IGNORE_FOR_TRACING**	string	**忽略要跟踪的特定查询，多个查询可以通过分号分隔，使用 \ 转义双引号。**

quarkus.datasource.jdbc.telemetry.enabled **quarkus.datasource."datasource-name".jdbc.telemetry.enabled** 启用 OpenTelemetry JDBC 检测。 环境变量： **QUARKUS_DATASOURCE_ JDBC_TELEMETRY_ENABLED**	布尔值	**如果 quarkus.datasource.jdbc.telemetry=false 和 true，则为 false（如果 quarkus.datasource.jdbc.telemetry=true）**

关于 DURATION 格式

要写入持续时间值，请使用标准的 **java.time.Duration** 格式。如需更多信息，请参阅 Duration#parse （）Java API 文档。

您还可以使用简化的格式，从数字开始：

- 如果该值只是一个数字，则代表以秒为单位。
- 如果值是数字，后跟 **ms**，则表示时间（毫秒）。

在其他情况下，简化的格式转换为 **java.time.Duration** 格式进行解析：

- 如果该值是一个数字，后跟 **h**、**m** 或 **s**，则使用 **PT** 前缀。
- 如果该值是一个数字，后跟 **d**，则前缀为 **P**。

1.4.3. JDBC URL 参考

每个支持的数据库都包含不同的 JDBC URL 配置选项。以下部分介绍了每个数据库 URL 和官方文档的链接。

1.4.3.1. DB2

jdbc:db2://<serverName>[:<portNumber>]/<databaseName>[:<key1>=<value>;[<key2>=<value2>;]]

Example

jdbc:db2://localhost:50000/MYDB:user=dbadm;password=dbadm;

有关 URL 语法和其他支持的选项的更多信息，请参阅 官方文档。

1.4.3.2. Derby

jdbc:derby:[//serverName[:portNumber]/]
[memory:]databaseName[;property=value[;property=value]]

Example

jdbc:derby://localhost:1527/myDB, **jdbc:derby:memory:myDB;create=true**

Derby 是一个嵌入式数据库，可作为服务器运行（基于文件），也可以完全在内存中运行。上面列出的所有选项都可用。

如需更多信息，请参阅 官方文档。

1.4.3.3. H2

jdbc:h2:{ {.|mem:}[name] | [file:]fileName | {tcp|ssl}:[//]server[:port][,server2[:port]]/name }
[;key=value...]

Example

jdbc:h2:tcp://localhost/~/test, **jdbc:h2:mem:myDB**

H2 是一个可在嵌入式或服务器模式下运行的数据库。它可以使用文件存储或完全在内存中运行。上面列出的所有选项都可用。

如需更多信息，请参阅 官方文档。

1.4.3.4. MariaDB

JDBC:mariadb:[replication:|failover:|sequential:|aurora:]//<hostDescription>[,
<hostDescription>...]/[database][?<key1>=<value1>[&<key2>=<value2>]] hostDescription:: < **host>**
[:<portnumber>] or address=(host=<host>)[(port=<portnumber>)][(type=(master|slave))]

Example

jdbc:mariadb://localhost:3306/test

如需更多信息，请参阅 官方文档。

1.4.3.5. Microsoft SQL 服务器

jdbc:sqlserver://[serverName[\instanceName][:portNumber]][;property=value[;property=value]]

Example

jdbc:sqlserver://localhost:1433;databaseName=AdventureWorks

Microsoft SQL Server JDBC 驱动程序的工作方式基本上与其它驱动程序相同。

如需更多信息，请参阅 官方文档。

1.4.3.6. MySQL

jdbc:mysql:[replication:|failover:|sequential:|aurora:]//<hostDescription>[,<hostDescription>...
]/[database][?<key1>=<value1>[&<key2>=<value2>]] hostDescription:: <host>[:< **portnumber>] 或**
address=(host=<host>)[(port=<portnumber>)[(port=<portnumber>)][（type=(master|slave)]

Example

jdbc:mysql://localhost:3306/test

如需更多信息，请参阅 官方文档。

1.4.3.6.1. MySQL 限制

当将 Quarkus 应用程序编译到原生镜像时，对 JMX 和 Oracle Cloud Infrastructure (OCI)集成的 MySQL 支持被禁用，因为它们与 GraalVM 原生镜像不兼容。

- 缺少 JMX 支持是以原生模式运行的自然结果，不太可能解决。
- 不支持与 OCI 集成。

1.4.3.7. Oracle

jdbc:oracle:driver_type:@database_specifier

Example

jdbc:oracle:thin:@localhost:1521/ORCL_SVC

如需更多信息，请参阅 官方文档。

1.4.3.8. PostgreSQL

jdbc:postgresql://[host][:port][/database][?key=value...]

Example

jdbc:postgresql://localhost/test

不同部分的默认值如下：

主机

localhost

port

5432

database

与用户名相同

有关附加参数的更多信息，请参阅 官方文档。

1.4.4. Quarkus 扩展和数据库驱动程序参考

下表列出了内置的 **db-kind** 值、相应的 Quarkus 扩展以及这些扩展使用的 JDBC 驱动程序。

使用其中一个内置数据源类型时，会自动解析 JDBC 和 Reactive 驱动程序，以匹配这些表中的值。

表 1.1. 数据库平台类型到 JDBC 驱动程序映射

数据库类型	Quarkus 扩展	驱动程序

数据库类型	Quarkus 扩展	驱动程序
db2	**quarkus-jdbc-db2**	• JDBC: **com.ibm.db2.jcc.DB2Driver** • XA: **com.ibm.db2.jcc.DB2XADataSource**
Derby	**quarkus-jdbc-derby**	• JDBC: **org.apache.derby.jdbc.ClientDriver** • XA: **org.apache.derby.jdbc.ClientXADataSource**
h2	**quarkus-jdbc-h2**	• jdbc: **org.h2.Driver** • XA: **org.h2.jdbcx.JdbcDataSource**
mariadb	**quarkus-jdbc-mariadb**	• jdbc: **org.mariadb.jdbc.Driver** • XA: **org.mariadb.jdbc.MySQLDataSource**
mssql	**quarkus-jdbc-mssql**	• JDBC: **com.microsoft.sqlserver.jdbc.SQLServerDriver** • XA: **com.microsoft.sqlserver.jdbc.SQLServerXADataSource**
mysql	**quarkus-jdbc-mysql**	• JDBC: **com.mysql.cj.jdbc.Driver** • XA: **com.mysql.cj.jdbc.MysqlXADataSource**
oracle	**quarkus-jdbc-oracle**	• JDBC: **oracle.jdbc.driver.OracleDriver** • XA: **oracle.jdbc.xa.client.OracleXADataSource**
postgresql	**quarkus-jdbc-postgresql**	• jdbc: **org.postgresql.Driver** • XA: **org.postgresql.xa.PGXADataSource**

表 1.2. 数据库类型到 Reactive 驱动程序映射

数据库类型	Quarkus 扩展	驱动
oracle	reactive-oracle-client	io.vertx.oracleclient.spi.OracleDriver
mysql	reactive-mysql-client	io.vertx.mysqlclient.spi.MySQLDriver
mssql	reactive-mssql-client	io.vertx.mssqlclient.spi.MSSQLDriver
postgresql	reactive-pg-client	io.vertx.pgclient.spi.PgDriver
db2	reactive-db2-client	io.vertx.db2client.spi.DB2Driver

提示

在大多数情况下，这个自动解析都适用，因此不需要驱动程序配置。

1.4.5. 主动数据源配置参考

构建时修复的配置属性 - 所有其他配置属性可在运行时覆盖

Configuration 属性	类型	default
quarkus.datasource.reactive 如果我们为此数据源创建主动数据源。 环境变量：**QUARKUS_DATASOURCE_REACTIVE**	布尔值	**true**
quarkus.datasource.reactive.cache-prepared-statements 准备的语句是否应在客户端上缓存。 环境变量：**QUARKUS_DATASOURCE_REACTIVE_CACHE_PREPARED_STATEMENTS**	布尔值	**false**
quarkus.datasource.reactive.url 数据源 URL。 如果设置了多个值，则此数据源会创建一个包含服务器列表而不是单个服务器的池。该池在连接建立过程中对服务器选择使用循环负载平衡。请注意，某些驱动程序可能在此上下文中不容纳多个值。 环境变量：**QUARKUS_DATASOURCE_REACTIVE_URL**	字符串列表	

quarkus.datasource.reactive.max-size 数据源池最大大小。 环境变量：**QUARKUS_DATASOURCE_REACTIVE_MAX_SIZE**	int	**20**
quarkus.datasource.reactive.event-loop-size 创建新连接对象时，池会为它分配一个事件循环。 当 **#event-loop-size** 设置为一个严格的正值时，池会以轮循方式分配任意数量的事件循环。默认情况下，使用 Quarkus 配置或计算的事件循环数量。如果 **#event-loop-size** 设置为零或负值，池会将当前事件循环分配给新连接。 环境变量：**QUARKUS_DATASOURCE_REACTIVE_EVENT_LOOP_SIZE**	int	
quarkus.datasource.reactive.trust-all 所有服务器证书是否应该被信任。 环境变量：**QUARKUS_DATASOURCE_REACTIVE_TRUST_ALL**	布尔值	**false**
quarkus.datasource.reactive.trust-certificate-pem 默认情况下禁用 PEM Trust 配置。 环境变量： **QUARKUS_DATASOURCE_REACTIVE_TRUST_CERTIFICATE_PEM**	布尔值	**false**
quarkus.datasource.reactive.trust-certificate-pem.certs 以逗号分隔的信任证书文件列表(Pem 格式)。 环境变量： **QUARKUS_DATASOURCE_REACTIVE_TRUST_CERTIFICATE_PEM_CERTS**	字符串列表	
quarkus.datasource.reactive.trust-certificate-jks JKS 配置默认为禁用。 环境变量： **QUARKUS_DATASOURCE_REACTIVE_TRUST_CERTIFICATE_JKS**	布尔值	**false**
quarkus.datasource.reactive.trust-certificate-jks.path 密钥文件的路径(JKS 格式)。 环境变量： **QUARKUS_DATASOURCE_REACTIVE_TRUST_CERTIFICATE_JKS_PATH**	string	

quarkus.datasource.reactive.trust-certificate-jks.password 密钥文件的密码。 环境变量： **QUARKUS_DATASOURCE_REACTIVE_TRUST_CERTIFICATE_JKS_PASSWORD**	string	
quarkus.datasource.reactive.trust-certificate-pfx 默认情况下禁用 PFX 配置。 环境变量： **QUARKUS_DATASOURCE_REACTIVE_TRUST_CERTIFICATE_PFX**	布尔值	**false**
quarkus.datasource.reactive.trust-certificate-pfx.path 密钥文件的路径(PFX 格式)。 环境变量： **QUARKUS_DATASOURCE_REACTIVE_TRUST_CERTIFICATE_PFX_PATH**	string	
quarkus.datasource.reactive.trust-certificate-pfx.password 密钥的密码。 环境变量： **QUARKUS_DATASOURCE_REACTIVE_TRUST_CERTIFICATE_PFX_PASSWORD**	string	
quarkus.datasource.reactive.key-certificate-pem 默认情况下禁用 PEM Key/cert 配置。 环境变量：**QUARKUS_DATASOURCE_REACTIVE_KEY_CERTIFICATE_PEM**	布尔值	**false**
quarkus.datasource.reactive.key-certificate-pem.keys 以逗号分隔的密钥文件的路径列表(Pem 格式)。 环境变量： **QUARKUS_DATASOURCE_REACTIVE_KEY_CERTIFICATE_PEM_KEYS**	字符串列表	
quarkus.datasource.reactive.key-certificate-pem.certs 以逗号分隔的证书文件的路径列表(Pem 格式)。 环境变量： **QUARKUS_DATASOURCE_REACTIVE_KEY_CERTIFICATE_PEM_CERTS**	字符串列表	
quarkus.datasource.reactive.key-certificate-jks JKS 配置默认为禁用。 环境变量：**QUARKUS_DATASOURCE_REACTIVE_KEY_CERTIFICATE_JKS**	布尔值	**false**

quarkus.datasource.reactive.key-certificate-jks.path 密钥文件的路径(JKS 格式)。 环境变量： **QUARKUS_DATASOURCE_REACTIVE_KEY_CERTIFICATE_JKS_PATH**	string	
quarkus.datasource.reactive.key-certificate-jks.password 密钥文件的密码。 环境变量： **QUARKUS_DATASOURCE_REACTIVE_KEY_CERTIFICATE_JKS_PASSWORD**	string	
quarkus.datasource.reactive.key-certificate-pfx 默认情况下禁用 PFX 配置。 环境变量：**QUARKUS_DATASOURCE_REACTIVE_KEY_CERTIFICATE_PFX**	布尔值	**false**
quarkus.datasource.reactive.key-certificate-pfx.path 密钥文件的路径(PFX 格式)。 环境变量： **QUARKUS_DATASOURCE_REACTIVE_KEY_CERTIFICATE_PFX_PATH**	string	
quarkus.datasource.reactive.key-certificate-pfx.password 密钥的密码。 环境变量： **QUARKUS_DATASOURCE_REACTIVE_KEY_CERTIFICATE_PFX_PASSWORD**	string	
quarkus.datasource.reactive.reconnect-attempts 当无法在第一次尝试建立连接时重新连接尝试的数量。 环境变量： **QUARKUS_DATASOURCE_REACTIVE_RECONNECT_ATTEMPTS**	int	**0**
quarkus.datasource.reactive.reconnect-interval 在第一次尝试时无法建立池连接时重新连接尝试的时间间隔。 环境变量：**QUARKUS_DATASOURCE_REACTIVE_RECONNECT_INTERVAL**	duration	**PT1S**

quarkus.datasource.reactive.hostname-verification-algorithm 应该检查服务器身份时使用的主机名验证算法。应该是 **HTTPS**、**LDAPS** 或 **NONE**。**NONE** 是默认值，禁用验证。 环境变量： **QUARKUS_DATASOURCE_REACTIVE_HOSTNAME_VERIFICATION_ALGORITHM**	string	**NONE**
quarkus.datasource.reactive.idle-timeout 连接在池关闭前未使用的最长时间。 环境变量：**QUARKUS_DATASOURCE_REACTIVE_IDLE_TIMEOUT**	duration	**没有超时**
quarkus.datasource.reactive.max-lifetime 连接在池中保留的最长时间，之后将根据需要关闭并替换它。 环境变量：**QUARKUS_DATASOURCE_REACTIVE_MAX_LIFETIME**	duration	**没有超时**
quarkus.datasource.reactive.shared 设置为 true，以在数据源间共享池。当未设置特定名称时，可以使用 **__vertx.DEFAULT** 名称来区分多个共享池。 环境变量：**QUARKUS_DATASOURCE_REACTIVE_SHARED**	布尔值	**false**
quarkus.datasource.reactive.name 设置池名称，在池在数据源之间共享时使用，否则忽略。 环境变量：**QUARKUS_DATASOURCE_REACTIVE_NAME**	string	
quarkus.datasource.reactive.additional-properties 在启动新连接时，要通过 Reactive SQL 客户端直接传递给数据库的其他未指定属性。 环境变量： **QUARKUS_DATASOURCE_REACTIVE_ADDITIONAL_PROPERTIES**	**Map<String,String>**	
Additional named datasources	**类型**	**default**
quarkus.datasource."datasource-name".reactive 如果我们为此数据源创建主动数据源。 环境变量：**QUARKUS_DATASOURCE__DATASOURCE_NAME__REACTIVE**	布尔值	**true**

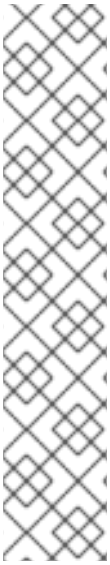
quarkus.datasource."datasource-name".reactive.cache-prepared-statements 准备的语句是否应在客户端上缓存。 环境变量： **QUARKUS_DATASOURCE__DATASOURCE_NAME__REACTIVE_CACHE_PREPARED_STATEMENTS**	布尔值	**false**
quarkus.datasource."datasource-name".reactive.url 数据源 URL。 如果设置了多个值，则此数据源会创建一个包含服务器列表而不是单个服务器的池。该池在连接建立过程中对服务器选择使用循环负载平衡。请注意，某些驱动程序可能在此上下文中不容纳多个值。 环境变量： **QUARKUS_DATASOURCE__DATASOURCE_NAME__REACTIVE_URL**	字符串列表	
quarkus.datasource."datasource-name".reactive.max-size 数据源池最大大小。 环境变量： **QUARKUS_DATASOURCE__DATASOURCE_NAME__REACTIVE_MAX_SIZE**	int	**20**
quarkus.datasource."datasource-name".reactive.event-loop-size 创建新连接对象时，池会为它分配一个事件循环。 当 **#event-loop-size** 设置为一个严格的正值时，池会以轮循方式分配任意数量的事件循环。默认情况下，使用 Quarkus 配置或计算的事件循环数量。如果 **#event-loop-size** 设置为零或负值，池会将当前事件循环分配给新连接。 环境变量： **QUARKUS_DATASOURCE__DATASOURCE_NAME__REACTIVE_EVENT_LOOP_SIZE**	int	
quarkus.datasource."datasource-name".reactive.trust-all 所有服务器证书是否应该被信任。 环境变量： **QUARKUS_DATASOURCE__DATASOURCE_NAME__REACTIVE_TRUST_ALL**	布尔值	**false**
quarkus.datasource."datasource-name".reactive.trust-certificate-pem 默认情况下禁用 PEM Trust 配置。 环境变量： **QUARKUS_DATASOURCE__DATASOURCE_NAME__REACTIVE_TRUST_CERTIFICATE_PEM**	布尔值	**false**

quarkus.datasource."datasource-name".reactive.trust-certificate-pem.certs 以逗号分隔的信任证书文件列表(Pem 格式)。 环境变量： **QUARKUS_DATASOURCE__DATASOURCE_NAME__REACTIVE_TRUST_CERTIFICATE_PEM_CERTS**	字符串列表	
quarkus.datasource."datasource-name".reactive.trust-certificate-jks JKS 配置默认为禁用。 环境变量： **QUARKUS_DATASOURCE__DATASOURCE_NAME__REACTIVE_TRUST_CERTIFICATE_JKS**	布尔值	**false**
quarkus.datasource."datasource-name".reactive.trust-certificate-jks.path 密钥文件的路径(JKS 格式)。 环境变量： **QUARKUS_DATASOURCE__DATASOURCE_NAME__REACTIVE_TRUST_CERTIFICATE_JKS_PATH**	string	
quarkus.datasource."datasource-name".reactive.trust-certificate-jks.password 密钥文件的密码。 环境变量： **QUARKUS_DATASOURCE__DATASOURCE_NAME__REACTIVE_TRUST_CERTIFICATE_JKS_PASSWORD**	string	
quarkus.datasource."datasource-name".reactive.trust-certificate-pfx 默认情况下禁用 PFX 配置。 环境变量： **QUARKUS_DATASOURCE__DATASOURCE_NAME__REACTIVE_TRUST_CERTIFICATE_PFX**	布尔值	**false**
quarkus.datasource."datasource-name".reactive.trust-certificate-pfx.path 密钥文件的路径(PFX 格式)。 环境变量： **QUARKUS_DATASOURCE__DATASOURCE_NAME__REACTIVE_TRUST_CERTIFICATE_PFX_PATH**	string	

quarkus.datasource."datasource-name".reactive.trust-certificate-pfx.password 密钥的密码。 环境变量： **QUARKUS_DATASOURCE__DATASOURCE_NAME__REACTIVE_TRUST_CERTIFICATE_PFX_PASSWORD**	string	
quarkus.datasource."datasource-name".reactive.key-certificate-pem 默认情况下禁用 PEM Key/cert 配置。 环境变量： **QUARKUS_DATASOURCE__DATASOURCE_NAME__REACTIVE_KEY_CERTIFICATE_PEM**	布尔值	**false**
quarkus.datasource."datasource-name".reactive.key-certificate-pem.keys 以逗号分隔的密钥文件的路径列表(Pem 格式)。 环境变量： **QUARKUS_DATASOURCE__DATASOURCE_NAME__REACTIVE_KEY_CERTIFICATE_PEM_KEYS**	字符串列表	
quarkus.datasource."datasource-name".reactive.key-certificate-pem.certs 以逗号分隔的证书文件的路径列表(Pem 格式)。 环境变量： **QUARKUS_DATASOURCE__DATASOURCE_NAME__REACTIVE_KEY_CERTIFICATE_PEM_CERTS**	字符串列表	
quarkus.datasource."datasource-name".reactive.key-certificate-jks JKS 配置默认为禁用。 环境变量： **QUARKUS_DATASOURCE__DATASOURCE_NAME__REACTIVE_KEY_CERTIFICATE_JKS**	布尔值	**false**
quarkus.datasource."datasource-name".reactive.key-certificate-jks.path 密钥文件的路径(JKS 格式)。 环境变量： **QUARKUS_DATASOURCE__DATASOURCE_NAME__REACTIVE_KEY_CERTIFICATE_JKS_PATH**	string	

quarkus.datasource."datasource-name".reactive.key-certificate-jks.password 密钥文件的密码。 环境变量： **QUARKUS_DATASOURCE__DATASOURCE_NAME__REACTIVE_KEY_CERTIFICATE_JKS_PASSWORD**	string	
quarkus.datasource."datasource-name".reactive.key-certificate-pfx 默认情况下禁用 PFX 配置。 环境变量： **QUARKUS_DATASOURCE__DATASOURCE_NAME__REACTIVE_KEY_CERTIFICATE_PFX**	布尔值	**false**
quarkus.datasource."datasource-name".reactive.key-certificate-pfx.path 密钥文件的路径(PFX 格式)。 环境变量： **QUARKUS_DATASOURCE__DATASOURCE_NAME__REACTIVE_KEY_CERTIFICATE_PFX_PATH**	string	
quarkus.datasource."datasource-name".reactive.key-certificate-pfx.password 密钥的密码。 环境变量： **QUARKUS_DATASOURCE__DATASOURCE_NAME__REACTIVE_KEY_CERTIFICATE_PFX_PASSWORD**	string	
quarkus.datasource."datasource-name".reactive.reconnect-attempts 当无法在第一次尝试建立连接时重新连接尝试的数量。 环境变量： **QUARKUS_DATASOURCE__DATASOURCE_NAME__REACTIVE_RECONNECT_ATTEMPTS**	int	**0**
quarkus.datasource."datasource-name".reactive.reconnect-interval 在第一次尝试时无法建立池连接时重新连接尝试的时间间隔。 环境变量： **QUARKUS_DATASOURCE__DATASOURCE_NAME__REACTIVE_RECONNECT_INTERVAL**	duration	**PT1S**

quarkus.datasource."datasource-name".reactive.hostname-verification-algorithm 应该检查服务器身份时使用的主机名验证算法。应该是 **HTTPS**、**LDAPS** 或 **NONE**。**NONE** 是默认值，禁用验证。 环境变量： **QUARKUS_DATASOURCE__DATASOURCE_NAME__REACTIVE_HOSTNAME_VERIFICATION_ALGORITHM**	string	**NONE**
quarkus.datasource."datasource-name".reactive.idle-timeout 连接在池关闭前未使用的最长时间。 环境变量： **QUARKUS_DATASOURCE__DATASOURCE_NAME__REACTIVE_IDLE_TIMEOUT**	duration	**没有超时**
quarkus.datasource."datasource-name".reactive.max-lifetime 连接在池中保留的最长时间，之后将根据需要关闭并替换它。 环境变量： **QUARKUS_DATASOURCE__DATASOURCE_NAME__REACTIVE_MAX_LIFETIME**	duration	**没有超时**
quarkus.datasource."datasource-name".reactive.shared 设置为 true，以在数据源间共享池。当未设置特定名称时，可以使用 **__vertx.DEFAULT** 名称来区分多个共享池。 环境变量： **QUARKUS_DATASOURCE__DATASOURCE_NAME__REACTIVE_SHARED**	布尔值	**false**
quarkus.datasource."datasource-name".reactive.name 设置池名称，在池在数据源之间共享时使用，否则忽略。 环境变量： **QUARKUS_DATASOURCE__DATASOURCE_NAME__REACTIVE_NAME**	string	
quarkus.datasource."datasource-name".reactive.additional-properties 在启动新连接时，要通过 Reactive SQL 客户端直接传递给数据库的其他未指定属性。 环境变量： **QUARKUS_DATASOURCE__DATASOURCE_NAME__REACTIVE_ADDITIONAL_PROPERTIES**	**Map<String,String>**	



关于 DURATION 格式

要写入持续时间值，请使用标准的 **java.time.Duration** 格式。如需更多信息，请参阅 Duration#parse （）Java API 文档。

您还可以使用简化的格式，从数字开始：

- 如果该值只是一个数字，则代表以秒为单位。
- 如果值是数字，后跟 **ms**，则表示时间（毫秒）。

在其他情况下，简化的格式转换为 **java.time.Duration** 格式进行解析：

- 如果该值是一个数字，后跟 **h**、**m** 或 **s**，则使用 **PT** 前缀。
- 如果该值是一个数字，后跟 **d**，则前缀为 **P**。

1.4.5.1. 被动 DB2 配置

构建时修复的配置属性 - 所有其他配置属性可在运行时覆盖

Datasources	类型	default
quarkus.datasource.reactive.db2.ssl **quarkus.datasource."datasource-name".reactive.db2.ssl** 是否启用 SSL/TLS。 环境变量： **QUARKUS_DATASOURCE_REACTIVE_DB2_SSL**	布尔值	**false**

1.4.5.2. 主动 MariaDB/MySQL 特定配置

构建时修复的配置属性 - 所有其他配置属性可在运行时覆盖

Additional named datasources	类型	default
quarkus.datasource.reactive.mysql.charset **quarkus.datasource."datasource-name".reactive.mysql.charset** 连接的 charset。 环境变量： **QUARKUS_DATASOURCE_REACTIVE_MYSQL_CHARSET**	string	

quarkus.datasource.reactive.mysql.collation **quarkus.datasource."datasource-name".reactive.mysql.collation** 连接合并。 环境变量： **QUARKUS_DATASOURCE_REACTIVE_MYSQL_COLLATION**	string	
quarkus.datasource.reactive.mysql.ssl-mode **quarkus.datasource."datasource-name".reactive.mysql.ssl-mode** 与服务器连接的所需安全状态。 请参阅 MySQL 参考手册。 环境变量： **QUARKUS_DATASOURCE_REACTIVE_MYSQL_SSL_MODE**	**禁用,preferred,required,verify-ca,verify-identity**	**disabled**
quarkus.datasource.reactive.mysql.connection-timeout **quarkus.datasource."datasource-name".reactive.mysql.connection-timeout** 连接超时（以秒为单位） 环境变量： **QUARKUS_DATASOURCE_REACTIVE_MYSQL_CONNECTION_TIMEOUT**	int	
quarkus.datasource.reactive.mysql.authentication-plugin **quarkus.datasource."datasource-name".reactive.mysql.authentication-plugin** 客户端应使用的身份验证插件。默认情况下，它使用由服务器在初始握手数据包中指定的插件名称。 环境变量： **QUARKUS_DATASOURCE_REACTIVE_MYSQL_AUTHENTICATION_PLUGIN**	**默认,mysql-clear-password,mysql-native-password,sha256-password,caching-sha2-password**	**default**

quarkus.datasource.reactive.mysql.pipelining-limit **quarkus.datasource."datasource-name".reactive.mysql.pipelining-limit** 可以管道的 inflight 数据库命令的最大数量。默认情况下禁用 pipelining。 环境变量： **QUARKUS_DATASOURCE_REACTIVE_MYSQL_PIPELINING_LIMIT**	int	
quarkus.datasource.reactive.mysql.use-affected-rows **quarkus.datasource."datasource-name".reactive.mysql.use-affected-rows** 是否返回与 *UPDATE* 语句中 *WHERE* 子句匹配的行数，而不是实际更改的行数。 环境变量： **QUARKUS_DATASOURCE_REACTIVE_MYSQL_USE_AFFECTED_ROWS**	布尔值	**false**

1.4.5.3. 主动 Microsoft SQL 服务器特定配置

构建时修复的配置属性 - 所有其他配置属性可在运行时覆盖

Datasources	类型	default
quarkus.datasource.reactive.mssql.packet-size **quarkus.datasource."datasource-name".reactive.mssql.packet-size** TDS 数据包所需的大小（以字节为单位）。 环境变量：**QUARKUS_DATASOURCE_REACTIVE_MSSQL_PACKET_SIZE**	int	
quarkus.datasource.reactive.mssql.ssl **quarkus.datasource."datasource-name".reactive.mssql.ssl** 是否启用 SSL/TLS。 环境变量：**QUARKUS_DATASOURCE_REACTIVE_MSSQL_SSL**	布尔值	**false**

1.4.5.4. 主动 Oracle 特定的配置

构建时修复的配置属性 - 所有其他配置属性可在运行时覆盖

Datasources	类型	default

1.4.5.5. 主动 PostgreSQL 特定配置

构建时修复的配置属性 - 所有其他配置属性可在运行时覆盖

Datasources	类型	default
quarkus.datasource.reactive.postgresql.pipelining-limit **quarkus.datasource."datasource-name".reactive.postgresql.pipelining-limit** 可以管道的 inflight 数据库命令的最大数量。 环境变量： **QUARKUS_DATASOURCE_REACTIVE_POSTGRESQL_PIPELINING_LIMIT**	int	
quarkus.datasource.reactive.postgresql.ssl-mode **quarkus.datasource."datasource-name".reactive.postgresql.ssl-mode** 客户端的 SSL 操作模式。 请参阅 不同模式中提供的保护。 环境变量： **QUARKUS_DATASOURCE_REACTIVE_POSTGRESQL_SSL_MODE**	**禁用，允许，首选，需要,verify-ca,verify-full**	**disable**
quarkus.datasource.reactive.postgresql.use-layer7-proxy **quarkus.datasource."datasource-name".reactive.postgresql.use-layer7-proxy** 级别 7 代理可以在多个连接上负载平衡到实际数据库的查询。当发生时，客户端可以被缺少会话关联和不需要的错误混淆，如 ERROR: unnamed prepared 语句不存在 (26000)。请参阅使用级别 7 代理 环境变量： **QUARKUS_DATASOURCE_REACTIVE_POSTGRESQL_USE_LAYER7_PROXY**	布尔值	**false**

1.4.6. 主动数据源 URL 参考

1.4.6.1. DB2

db2://[user[:[password]]@]host[:port][/database][?<key1>=<value1>[&<key2>=<value2>]]

Example

db2://dbuser:secretpassword@database.server.com:50000/mydb

目前，客户端支持以下参数键：

- **主机**
- **port**

- **user**
- **password**
- **database**

注意

在连接 URL 中配置参数会覆盖默认属性。

1.4.6.2. Microsoft SQL 服务器

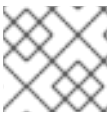
sqlserver://[user[:[password]]@]host[:port][/database][?<key1>=<value1>[&<key2>=<value2>]]

Example

sqlserver://dbuser:secretpassword@database.server.com:1433/mydb

目前，客户端支持以下参数键：

- **主机**
- **port**
- **user**
- **password**
- **database**



注意

在连接 URL 中配置参数会覆盖默认属性。

1.4.6.3. MySQL / MariaDB

mysql://[user[:[password]]@]host[:port][/database][?<key1>=<value1>[&<key2>=<value2>]]

Example

mysql://dbuser:secretpassword@database.server.com:3211/mydb

目前，客户端支持以下参数键（不区分大小写）：

- **主机**
- **port**
- **user**
- **password**
- **schema**
- **socket**
- **useAffectedRows**

注意

在连接 URL 中配置参数会覆盖默认属性。

1.4.6.4. Oracle

1.4.6.4.1. EZConnect 格式

oracle:thin:@[[protocol:]//]host[:port][/service_name][:server_mode][/instance_name][?connection properties]

Example

oracle:thin:@mydbhost1:5521/mydbservice?connect_timeout=10sec

1.4.6.4.2. TNS 别名格式

oracle:thin:@<alias_name>[?connection properties]

Example

oracle:thin:@prod_db?TNS_ADMIN=/work/tns/

1.4.6.5. PostgreSQL

postgresql://[user[:[password]]@]host[:port][/database][?<key1>=<value1>[&<key2>=<value2>]]

Example

postgresql://dbuser:secretpassword@database.server.com:5432/mydb

目前，客户端支持：

- 以下参数键：
  - **主机**
  - **port**
  - **user**
  - **password**
  - **dbname**
  - **sslmode**
- 其他属性，例如：
  - **application_name**
  - **fallback_application_name**
  - **search_path**
  - **options**

注意

在连接 URL 中配置参数会覆盖默认属性。