



Red Hat build of Quarkus 3.8

データソースの設定

法律上の通知

Copyright © 2024 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux[®] is the registered trademark of Linus Torvalds in the United States and other countries.

Java[®] is a registered trademark of Oracle and/or its affiliates.

XFS[®] is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL[®] is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js[®] is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack[®] Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

概要

統合設定モデルを使用して、Red Hat build of Quarkus で JDBC およびリアクティブドライバー データソースを定義します。

目次

RED HAT BUILD OF QUARKUS ドキュメントへのフィードバックの提供	3
多様性を受け入れるオープンソースの強化	4
第1章 RED HAT BUILD OF QUARKUS でのデータソースの設定	5
1.1. QUARKUS で DATASOURCES の設定を開始する	5
1.2. データソースを設定する	7
1.3. データソースのインテグレーション	14
1.4. 参考資料	17

RED HAT BUILD OF QUARKUS ドキュメントへのフィードバックの提供

エラーを報告したり、ドキュメントを改善したりするには、Red Hat Jira アカウントにログインし、課題を送信してください。Red Hat Jira アカウントをお持ちでない場合は、アカウントを作成するように求められます。

手順

1. 次のリンクをクリックして [チケットを作成します](#)。
2. **Summary** に課題の簡単な説明を入力します。
3. **Description** に課題や機能拡張の詳細な説明を入力します。問題があるドキュメントのセクションへの URL を含めてください。
4. **Submit** をクリックすると、課題が作成され、適切なドキュメントチームに転送されます。

多様性を受け入れるオープンソースの強化

Red Hat では、コード、ドキュメント、Web プロパティにおける配慮に欠ける用語の置き換えに取り組んでいます。まずは、マスター (master)、スレーブ (slave)、ブラックリスト (blacklist)、ホワイトリスト (whitelist) の 4 つの用語の置き換えから始めます。この取り組みは膨大な作業を要するため、今後の複数のリリースで段階的に用語の置き換えを実施して参ります。詳細は、[Red Hat CTO である Chris Wright のメッセージ](#) をご覧ください。

第1章 RED HAT BUILD OF QUARKUS でのデータソースの設定

統合設定モデルを使用して、Java Database Connectivity (JDBC) とリアクティブデータベースドライバーのデータソースを定義できます。

アプリケーションは、データソースを使用してリレーショナルデータベースにアクセスします。Quarkus では、Java Database Connectivity (JDBC) とリアクティブデータベースドライバーのデータソースを定義するための統合設定モデルが提供されています。

Quarkus では、[Agroal](#) と [Vert.x](#) を使用することで、JDBC とリアクティブドライバーで高性能かつスケラブルなデータソース接続プールが可能になります。`quarkus-jdbc-*` および `quarkus-reactive*-client` エクステンションは、ビルド時間の最適化を提供し、設定されたデータソースをセキュリティー、ヘルスチェック、メトリクスなどの Quarkus 機能と統合します。

リアクティブデータソースの消費と使用の詳細は、Quarkus [Reactive SQL clients](#) ガイドを参照してください。

さらに、JDBC データソースの消費と使用については、Quarkus [Hibernate ORM](#) ガイドを参照してください。

1.1. QUARKUS で DATASOURCES の設定を開始する

このセクションでは、基本的な部分を理解しているユーザー向けに、データソースのセットアップを容易にするための概要とコードサンプルについて説明します。

より高度な設定例については、[参考資料](#) を参照してください。

1.1.1. 開発モードでのゼロコンフィグセットアップ

Quarkus では Dev Services 機能を使用することで容易にデータベースを設定でき、開発 (dev) モードでのテストや稼働を目的としたデータベースのゼロコンフィグセットアップが可能になります。開発モードでは、Dev Services を使用して Quarkus にデータベースを処理させることが推奨されます。一方、実稼働モードでは、Quarkus 外で管理されるデータベースを指す明示的なデータベース設定の詳細を指定します。

Dev Services を使用するには、目的のデータベース型に適したドライバーエクステンション (`jdbc-postgresql` など) を `pom.xml` ファイルに追加します。開発モードでは、データベース接続の詳細を明示的に指定しない場合、Quarkus が自動的にデータベースのセットアップを処理し、アプリケーションとデータベース間の接続を指定します。

ユーザー認証情報を指定すると、基礎となるデータベースがその認証情報を使用するように設定されます。これは、外部ツールを使用してデータベースに接続する場合に便利です。

この機能を使用するには、データベース型に応じて Docker または Podman コンテナランタイムがインストールされていることを確認してください。H2 などの特定のデータベースは in-memory モードで動作し、コンテナランタイムを必要としません。

ヒント

本番モードで使用する実際の接続詳細の前に `%prod` を付けて、開発モードに適用されないようにしてください。詳細は、"設定リファレンス" ガイドの [プロファイル](#) セクションを参照してください。

Dev Services の詳細は、[Dev Services の概要](#) を参照してください。

詳細とオプションの設定については、[Dev Services for databases](#) を参照してください。

1.1.2. JDBC データソースを設定する

1. 選択したデータベースに応じて、適切な JDBC エクステンションを追加します。

- **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. JDBC データソースを設定します。

```
quarkus.datasource.db-kind=postgresql ❶
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
```

❶ この設定値は、クラスパス上に複数のデータベースエクステンションがある場合にのみ必要です。

利用可能なエクステンションが1つだけの場合、Quarkus はそれが適切であると想定します。ドライバーをテストスコープに追加すると、Quarkus は指定されたドライバーをテストに自動的に含めます。

1.1.2.1. JDBC 接続プールのサイズ調整

負荷のピーク時の過負荷からデータベースを保護するには、データベースの負荷にスロットリングを適用できるようにプールサイズを適切に設定します。最適なプールサイズは、並列アプリケーションユーザーの数やワークロードの性質など、多くの要因により異なります。

プールサイズの設定が小さすぎると、接続の待機中に一部のリクエストがタイムアウトになる可能性があることに注意してください。

プールサイズ調整プロパティの詳細は、[JDBC 設定リファレンス](#) セクションを参照してください。

1.1.3. リアクティブデータソースを設定する

1. 選択したデータベースに応じて、適切なリアクティブエクステンションを追加します。

- **quarkus-reactive-db2-client**
- **quarkus-reactive-mssql-client**
- **quarkus-reactive-mysql-client**

- **quarkus-reactive-oracle-client**
- **quarkus-reactive-pg-client**

2. リアクティブデータソースを設定します。

```
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 この設定値は、クラスパス上に複数のリアクティブドライバーエクステンションがある場合にのみ必要です。

1.2. データソースを設定する

次のセクションでは、単一または複数データソースの設定について説明します。説明をシンプルにするために、単一データソースをデフォルトデータソース (名前なし) とします。

1.2.1. 単一データソースを設定する

データソースは、JDBC データソース、リアクティブ、またはその両方になります。これは、プロジェクトエクステンションの設定と選択に応じて決定されます。

1. 次の設定プロパティを使用してデータソースを定義します。この場合の **db-kind** は、接続先のデータベースプラットフォーム (例: **h2**) を定義します。

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

Quarkus は、**db-kind** データベースプラットフォーム属性に指定された値から、使用するべき JDBC ドライバークラスを推定します。



注記

この手順は、アプリケーションが複数のデータベースドライバーに依存する場合にのみ必要です。アプリケーションが単一のドライバーで動作する場合、ドライバーは自動的に検出されます。

Quarkus には現在、次の種類のビルトインデータベースがあります。

- DB2: **db2**
- Derby: **derby**
- H2: **h2**
- MariaDB: **mariadb**
- Microsoft SQL Server: **mssql**
- MySQL: **mysql**

- Oracle: **oracle**
- PostgreSQL: **postgresql**、**pgsql** または **pg**
- ビルトインではない種類のデータベースを使用する場合は **other** を使用し、JDBC ドライバーを明示的に定義します。



注記

[他のデータベースを使用する](#) で説明されているように、JVM モードの Quarkus アプリケーションでは任意の JDBC ドライバーを使用できます。ただし、ビルトイン以外の種類のデータベースを使用すると、アプリケーションをネイティブ実行可能ファイルにコンパイルする際に機能しない可能性が高くなります。

ネイティブ実行可能ファイルのビルドの場合、利用可能な JDBC Quarkus エクステンションを使用するか、使用する特定のドライバー用のカスタムエクステンションを提供することが推奨されます。

2. 次のプロパティを設定して認証情報を定義します。

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

データソースの [認証情報プロバイダー](#) を使用して、Vault からパスワードを取得することもできます。

これまでの、JDBC とリアクティブドライバーのどちらを使用しているかにかかわらず、設定は同じでした。データベースの種類と認証情報の定義以外は、使用しているドライバーの種類により異なります。JDBC とリアクティブドライバーは、同時に使用できます。

1.2.1.1. JDBC データソース

JDBC は最も一般的なデータベース接続パターンであり、通常は非リアクティブな Hibernate ORM と組み合わせて使用する場合に必要です。

1. JDBC データソースを使用する場合は、まず必要な依存関係を追加します。
 - a. ビルトイン JDBC ドライバーで使用する場合は、以下のリストからリレーショナルデータベースドライバーの Quarkus エクステンションを選択して追加します。
 - Derby - **quarkus-jdbc-derby**
 - H2 - **quarkus-jdbc-h2**



注記

H2 データベースと Derby データベースは、"組み込みモード" で実行するように設定できます。ただし、Derby エクステンションでは、組み込みデータベースエンジンのネイティブ実行可能ファイルへのコンパイルはサポートされていません。

結合テストに関する提案事項については、[in-memory データベースを使用してテストする](#) を参照してください。

- 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**
- 他の JDBC エクステンション ([SQLite](#) やその [ドキュメント](#) など) は、[Quarkiverse](#) にあります。
たとえば、PostgreSQL ドライバーの依存関係を追加するには、以下を実行します。

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



注記

ビルトイン JDBC ドライバーエクステンションを使用すると、Agroal エクステンションが自動的に組み込まれます。これは、カスタムおよびビルトインの JDBC ドライバーに適用できる JDBC 接続プール実装です。ただし、カスタムドライバーの場合は Agroal を明示的に追加する必要があります。

- カスタム JDBC ドライバーで使用するには、リレーショナルデータベースドライバーのエクステンションとともに **quarkus-agroal** 依存関係をプロジェクトに追加します。

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

別のデータベースの JDBC ドライバーを使用するには、[ビルトインエクステンションがない](#)、または別のドライバーがあるデータベースを使用します。

- JDBC URL プロパティを定義して JDBC 接続を設定します。

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



注記

プロパティ名の接頭辞 **jdbc** に注意してください。JDBC 固有のすべての設定プロパティには、接頭辞として **jdbc** が付いています。リアクティブデータソースの場合、接頭辞は **reactive** です。

JDBC の設定の詳細は、[JDBC URL フォーマットのリファレンス](#) および [Quarkus エクステンションとデータベースドライバーのリファレンス](#) を参照してください。

1.2.1.1.1. カスタムデータベースとドライバー

JDBC ドライバーを使用して、Quarkus がエクステンションを提供しないデータベースに接続する必要がある場合は、代わりにカスタムドライバーを使用できます。たとえば、プロジェクトで OpenTracing JDBC ドライバーを使用している場合です。

エクステンションがなければ、ドライバーは JVM モードで実行されている Quarkus アプリケーションで正常に動作します。しかし、アプリケーションをネイティブ実行可能ファイルにコンパイルする際に、ドライバーが機能しない可能性があります。ネイティブ実行可能ファイルを作成する予定がある場合は、既存の JDBC Quarkus エクステンションを使用するか、ドライバー用にエクステンションを提供してください。



警告

OpenTelemetry を優先し、OpenTracing が非推奨になりました。トレーシング情報については、以下の [データソースのトレース](#) に関する関連セクションを参照してください。

従来の OpenTracing ドライバーを使用したカスタムドライバー定義の例:

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

JVM モードでビルトインサポートがないデータベースへのアクセスを定義した例:

```
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
```

JDBC 設定オプションや、接続プールサイズなどの他の側面の設定に関する詳細は、[JDBC 設定リファレンス](#) セクションを参照してください。

1.2.1.1.2. データソースを消費する

Hibernate ORM を使用すると、Hibernate レイヤーが自動的にデータソースを取得して使用します。

データソースへの in-code アクセスでは、次のように他の Bean としてデータソースを取得する必要があります。

```
@Inject
AgroalDataSource defaultDataSource;
```

上記の例のタイプは **AgroalDataSource**、つまり **javax.sql.DataSource** サブタイプです。そのため、注入されたタイプとして **javax.sql.DataSource** も使用できます。

1.2.1.2. リアクティブデータソース

Quarkus では、リアクティブデータソースで使用できるリアクティブクライアントがいくつか提供されています。

1. 対応するエクステンションをアプリケーションに追加します。
 - 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**
インストールされたエクステンションは、データソース設定で定義した **quarkus.datasource.db-kind** と一致する必要があります。

2. ドライバーを追加した後、接続 URL を設定し、接続プールの適切なサイズを定義します。

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

1.2.1.2.1. リアクティブ接続プールのサイズ調整

負荷のピーク時の過負荷からデータベースを保護するには、データベースの負荷にスロットリングを適用できるようにプールサイズを適切に設定します。適切なサイズは、並列アプリケーションユーザーの数やワークロードの性質など、多くの要因により異なります。

プールサイズの設定が小さすぎると、接続の待機中に一部のリクエストがタイムアウトになる可能性があることに注意してください。

プールサイズ調整プロパティの詳細は、[リアクティブデータソース設定リファレンス](#) セクションを参照してください。

1.2.1.3. JDBC とリアクティブデータソースの同時使用

JDBC エクステンション (と Agroal) および特定の種類のデータベースを処理するリアクティブデータソースエクステンションが含まれている場合、デフォルトで両方が作成されます。

- JDBC データソースを明示的に無効にするには、以下を実行します。

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

- リアクティブデータソースを明示的に無効にするには、以下を実行します。

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

ヒント

ほとんどの場合、JDBC ドライバーとリアクティブデータソースエクステンションは、両方ではなくいずれか一方のみ存在するため、上記の設定はオプションです。

1.2.2. 複数のデータソースを設定する



注記

Hibernate ORM エクステンションは、設定プロパティを使用した [永続化ユニット](#) の定義をサポートします。永続ユニットごとに、選択したデータソースを参照できます。

複数のデータソースを定義することは、単一のデータソースを定義するのと同じように機能しますが、重要な相違点として名前 (設定プロパティ) を定義する必要があります。

次の例では、3つの異なるデータソースを示しています。

- デフォルトのデータソース
- **users** という名前のデータソース
- **inventory** という名前のデータソース

それぞれの設定は次のとおりです。

```
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
```

設定プロパティに追加のセクションがあることに注意してください。構文は **quarkus.datasource.[optional name].[datasource property]** です。



注記

データベースエクステンションが1つだけインストールされている場合でも、Quarkus がそれを検出できるように、名前付きデータベースは少なくとも1つのビルドタイムプロパティを指定する必要があります。通常これは **db-kind** プロパティですが、[Dev Services for Databases](#) ガイドに従って、Dev Services プロパティを指定して名前付きデータソースを作成することもできます。

1.2.2.1. 名前付きデータソースの注入

複数のデータソースを使用する場合、各 **DataSource** には、データソースの名前を値として持つ **io.quarkus.agroal.DataSource** 修飾子もあります。

前のセクションで説明したプロパティを使用して3つの異なるデータソースを設定し、それぞれを次のように注入します。

```
@Inject
AgroalDataSource defaultDataSource;

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

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

1.2.3. データソースのアクティブ化/非アクティブ化

データソースがビルド時に設定されている場合、デフォルトでは実行時にアクティブになります。つまり、Quarkus はアプリケーションの起動時に対応する JDBC 接続プールまたはリアクティブクライアントを起動します。

実行時にデータソースを非アクティブ化するには、**quarkus.datasource[.optional name].active** を **false** に設定します。その後、Quarkus はアプリケーションの起動時に対応する JDBC 接続プールまたはリアクティブクライアントを開始しません。実行時に対応するデータソースを使用しようとすると、明確なエラーメッセージが表示されて失敗します。

これは、アプリケーションが事前に決定された一連のデータソースを実行時に使用できるようにする場合に特に便利です。



警告

別の Quarkus エクステンションが非アクティブなデータソースに依存する場合、そのエクステンションは起動に失敗する可能性があります。

この場合は、他のエクステンションも非アクティブ化する必要があります。たとえば、[here for Hibernate ORM](#) を参照してください。

たとえば、以下の設定を使用します。

```
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
```

実行時に **quarkus.datasource."pg".active=true** を設定すると、PostgreSQL データソースのみが利用でき、実行時に **quarkus.datasource."oracle".active=true** を設定すると Oracle データソースのみが利用可能になります。

ヒント

[カスタム設定プロファイル](#) は、このような設定を単純化するのに役立ちます。上記のプロファイル固有の設定を上記の設定に追加することで、[quarkus.profile](#) を設定するだけで、実行時に永続ユニット/データソースを選択できます (`quarkus.profile=prod,pg` または `quarkus.profile=prod,oracle`)。

```
%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."
```

ヒント

また、次のように、現在アクティブなデータソースにリダイレクトする [CDI bean プロデューサー](#) を定義すると便利なこともあります。

```
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. データソースのインテグレーション

1.3.1. データソースヘルスチェック

[quarkus-smallrye-health](#) エクステンションを使用する場合は、[quarkus-agroal](#) エクステンションおよびリアクティブクライアントエクステンションが自動的にレディネスヘルスチェックを追加してデータソースを検証します。

アプリケーションのヘルスレディネスエンドポイント (デフォルトは `/q/health/ready`) にアクセスすると、データソースの検証ステータスに関する情報を受け取ります。複数のデータソースがある場合はすべてのデータソースがチェックされ、そのうちの1つでもデータソース検証に失敗すると、ステータスが **DOWN** に遷移します。

この動作は、**quarkus.datasource.health.enabled** プロパティを使用して無効にできます。

特定のデータソースのみをヘルスチェックから除外するには、以下を使用します。

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

1.3.2. データソースメトリクス

quarkus-micrometer または **quarkus-smallrye-metrics** エクステンションを使用している場合、**quarkus-agroal** は一部のデータソース関連のメトリクスをメトリクスレジストリーに提供できます。これは、**quarkus.datasource.metrics.enabled** プロパティを **true** に設定することでアクティブ化できます。

メトリクスに実際の値を含むメトリクスを公開するには、Agroal メカニズムを使用して内部でメトリクスを有効にする必要があります。メトリクスエクステンションが存在し、Agroal エクステンションのメトリクスが有効になっている場合、このメトリクス収集メカニズムはデフォルトですべてのデータソースに対して有効になります。

特定のデータソースのメトリクスを無効にするには、**quarkus.datasource.jdbc.enable-metrics** を **false** に設定するか、名前付きデータソースに **quarkus.datasource.<datasource name>.jdbc.enable-metrics** を適用します。これにより、収集メカニズムが無効の場合は、メトリクスの収集と `/q/metrics` エンドポイントへの公開が無効になります。

逆に、**quarkus.datasource.jdbc.enable-metrics** を **true** に設定するか、名前付きデータソースに **quarkus.datasource.<datasource name>.jdbc.enable-metrics** を設定すると、メトリクスエクステンションが使用されていない場合でもメトリクス収集が明示的に有効になります。これは、収集されたメトリクスにプログラムを使用してアクセスする必要がある場合に役立ちます。これらは、注入された **AgroalDataSource** インスタンスで **dataSource.getMetrics()** を呼び出すと使用可能になります。

このデータソースのメトリクス収集が無効になっている場合、すべての値はゼロになります。

1.3.3. データソースのトレース

データソースでトレースを使用するには、**quarkus-opentelemetry** エクステンションをプロジェクトに追加する必要があります。

トレーシングが必要なため、別のドライバーを宣言する必要はありません。JDBC ドライバーを使用する場合は、[ここ](#) で OpenTelemetry エクステンションの手順に従う必要があります。

すべてのトレースインフラストラクチャーを使用しても、データソーストレースはデフォルトで有効になっていません。そのため、このプロパティを設定して有効にする必要があります。

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

1.3.4. Narayana Transaction Manager のインテグレーション

Narayana JTA エクステンションも使用できる場合、インテグレーションは自動的行われます。

これは、**transactions** 設定プロパティを設定することでオーバーライドできます。

- デフォルトの名前が指定されていないデータソースの **quarkus.datasource.jdbc.transactions**
- 名前付きデータソースの **quarkus.datasource.<datasource-name>.jdbc.transactions**

詳細は、後述の [設定リファレンス](#) セクションを参照してください。

データベースへのトランザクションログの保存を JDBC を使用して容易にするには、[Quarkus でのトランザクションの使用](#) ガイドの [データソースに保存されるトランザクションログの設定](#) セクションを参照してください。

1.3.4.1. 名前付きデータソース

Dev Services を使用する場合、必ずデフォルトのデータソースが作成されます。しかし、名前付きデータソースを指定するには、Quarkus がデータソースの作成方法を検出できるように、少なくとも1つのビルドタイムプロパティが必要です。

通常は、**db-kind** プロパティを指定する

か、**quarkus.datasource."name".devservices.enabled=true** を設定して Dev Services を明示的に有効にします。

1.3.5. in-memory データベースを使用したテスト

H2 や Derby などの一部のデータベースは、結合テストを迅速に実行するファシリティーとして頻繁に組み込みモードで使用されます。

この場合、実稼働環境で使用する予定の実際のデータベースを使用することが推奨されます。特に [Dev Services がテスト用にゼロコンフィグデータベースを提供](#) しており、比較的迅速にコンテナをテストでき、実際の環境で期待どおりの結果が得られる場合に当てはまります。ただし、単純な結合テストを実行する機能が必要なシナリオでは、JVM 対応データベースを使用することもできます。

1.3.5.1. サポートと制限事項

組み込みデータベース (H2 および Derby) は JVM モードで動作します。ネイティブモードの場合、次の制限が適用されます。

- ネイティブモードでは、Derby をアプリケーションに組み込めません。ただし、Quarkus Derby エクステンションを使用すると、Derby JDBC クライアントのネイティブコンパイルが可能になり、リモート接続がサポートされます。
- ネイティブイメージに H2 を組み込むことは推奨されません。代わりに別のデータベースへのリモート接続を使用するなど、別の方法を検討してください。

1.3.5.2. 結合テストを実行する

1. 次の Maven コーディネートの下にある追加ツールを提供するアーティファクトに依存関係を追加します。
 - H2 の場合は **io.quarkus:quarkus-test-h2**
 - Derby の場合は **io.quarkus:quarkus-test-derby**
これにより、データベースが JVM プロセスとして実行されている間、アプリケーションがネイティブ実行可能ファイルにコンパイルされていても、アプリケーションをテストできるようになります。
2. JVM またはネイティブ実行可能ファイルの両方で結合テストを実行するには、結合テストの任意のクラスに次のアノテーションを追加します。
 - **@QuarkusTestResource(H2DatabaseTestResource.class)**
 - **@QuarkusTestResource(DerbyDatabaseTestResource.class)**

そうすることで、テストの実行で必要なとおり、テストスイートはマネージドデータベースを必ず別のプロセスで開始および終了します。

H2 の例

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

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

@QuarkusTestResource(H2DatabaseTestResource.class)
public class TestResources {
}
```

3. マネージドデータベースへの接続を設定します。

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

1.4. 参考資料

1.4.1. 共通データソース設定リファレンス

 ビルド時に固定された設定プロパティ: その他の設定プロパティはすべて実行時にオーバーライド可能

設定プロパティ	型	デフォルト
 quarkus.datasource.health.enabled <small>rye-health</small> エクステンションが存在する場合にヘルスチェックを公開するかどうか。 これはグローバル設定であり、データソース固有の設定ではありません。 環境変数: QUARKUS_DATASOURCE_HEALTH_ENABLED	boolean	true

<p> quarkus.datasource.metrics.enabled</p> <p>メトリクスエクステンションが存在する場合にデータソースメトリクスを公開するかどうか。</p> <p>これはグローバル設定であり、データソース固有の設定ではありません。</p> <p> 注記</p> <p>これは、データソースのメトリクス収集を有効にするために JDBC データソースレベルで設定する必要がある "jdbc.enable-metrics" プロパティとは異なります。</p> <p>環境変数: QUARKUS_DATASOURCE_METRICS_ENABLED</p>	boolean	false
<p> quarkus.datasource.db-kind</p> <p>quarkus.datasource."datasource-name".db-kind</p> <p>接続先のデータベースの種類 (h2、postgresql など)。</p> <p>環境変数: QUARKUS_DATASOURCE_DB_KIND</p>	string	

<p> quarkus.datasource.db-version</p> <p>quarkus.datasource."datasource-name".db-version</p> <p>接続先のデータベースのバージョン (例: '10.0').</p> <p>注意</p> <p>ここで設定するバージョン番号は、データベースの JDBC ドライバーの java.sql.DatabaseMetaData#getDatabaseProductVersion() で返される文字列と同じ番号付けスキームに従う必要があります。この番号付けスキームは、データベースで最も一般的に使用される番号付けスキームとは異なる場合があります。たとえば、Microsoft SQL Server 2016 はバージョン 13 になります。</p> <p>原則として、ここでは可能な限り高いバージョンを設定する必要がありますが、アプリケーションの接続先となるデータベースのバージョン以下でなければなりません。</p> <p>バージョンが高いほどパフォーマンスが向上し、より多くの機能を使用できます (たとえば、Hibernate ORM はより効率的な SQL を生成し、回避策を回避し、より多くのデータベース機能を活用できます)。しかし、接続先のデータベースよりバージョンが高くと、ランタイム例外が発生する (たとえば、Hibernate ORM がデータベースが拒否する無効な SQL を生成する) 可能性があります。</p> <p>一部のエクステンション (Hibernate ORM エクステンションなど) は、起動時にこのバージョンを実際のデータベースバージョンと照合しようとします。そのため、実際のバージョンが低いために起動に失敗するか、データベースにアクセスできずに警告が表示されます。</p> <p>このプロパティのデフォルトは各エクステンションにより異なり、Hibernate ORM エクステンションのデフォルトは最も古いサポート対象のバージョンです。</p> <p>環境変数: QUARKUS_DATASOURCE_DB_VERSION</p>	string	
<p> quarkus.datasource.devservices.enabled</p> <p>quarkus.datasource."datasource-name".devservices.enabled</p> <p>この Dev Service を dev モードまたはテストでアプリケーションで始まるかどうか。</p> <p>接続設定 (JDBC URL やリアクティブクライアント URL など) が明示的に設定されていない限り、Dev Service はデフォルトで有効になります。</p> <p>環境変数: QUARKUS_DATASOURCE_DEVSERVICES_ENABLED</p>	boolean	
<p> quarkus.datasource.devservices.image-name</p> <p>quarkus.datasource."datasource-name".devservices.image-name</p> <p>コンテナベースの Dev Service プロバイダーのコンテナイメージ名。</p> <p>プロバイダーが H2 や Derby などのコンテナベースのデータベースでない場合、これは効果がありません。</p> <p>環境変数: QUARKUS_DATASOURCE_DEVSERVICES_IMAGE_NAME</p>	string	

 quarkus.datasource.devservices.container-env quarkus.datasource."datasource-name".devservices.container-env コンテナに渡される環境変数。 環境変数: QUARKUS_DATASOURCE_DEVSERVICES_CONTAINER_ENV	Map<String, String>	
 quarkus.datasource.devservices.container-properties quarkus.datasource."datasource-name".devservices.container-properties 追加のコンテナ設定に渡される汎用プロパティ。 ここで定義されるプロパティはデータベース固有であり、各データベースの Dev Service 実装で解釈されます。 環境変数: QUARKUS_DATASOURCE_DEVSERVICES_CONTAINER_PROPERTIES	Map<String, String>	
 quarkus.datasource.devservices.properties quarkus.datasource."datasource-name".devservices.properties データベース接続 URL に追加される汎用プロパティ。 環境変数: QUARKUS_DATASOURCE_DEVSERVICES_PROPERTIES	Map<String, String>	
 quarkus.datasource.devservices.port quarkus.datasource."datasource-name".devservices.port 開発サービスがリッスンするオプションの固定ポート。 定義されていない場合、ポートはランダムに選択されます。 環境変数: QUARKUS_DATASOURCE_DEVSERVICES_PORT	int	
 quarkus.datasource.devservices.command quarkus.datasource."datasource-name".devservices.command コンテナベースの Dev Service プロバイダーに使用するコンテナ起動コマンド。 プロバイダーが H2 や Derby などのコンテナベースのデータベースでない場合、これは効果がありません。 環境変数: QUARKUS_DATASOURCE_DEVSERVICES_COMMAND	string	

 quarkus.datasource.devservices.db-name quarkus.datasource."datasource-name".devservices.db-name この Dev Service がオーバーライドをサポートする場合に使用するデータベース名。 環境変数: QUARKUS_DATASOURCE_DEVSERVICES_DB_NAME	string	
 quarkus.datasource.devservices.username quarkus.datasource."datasource-name".devservices.username この Dev Service がオーバーライドをサポートする場合に使用するユーザー名。 環境変数: QUARKUS_DATASOURCE_DEVSERVICES_USERNAME	string	
 quarkus.datasource.devservices.password quarkus.datasource."datasource-name".devservices.password この Dev Service がオーバーライドをサポートする場合に使用するパスワード。 環境変数: QUARKUS_DATASOURCE_DEVSERVICES_PASSWORD	string	
 quarkus.datasource.devservices.init-script-path quarkus.datasource."datasource-name".devservices.init-script-path クラスパスからロードされ、Dev Service データベースに適用される SQL スクリプトへのパス。 プロバイダーが H2 や Derby などのコンテナベースのデータベースでない場合、これは効果がありません。 環境変数: QUARKUS_DATASOURCE_DEVSERVICES_INIT_SCRIPT_PATH	string	
 quarkus.datasource.devservices.volumes quarkus.datasource."datasource-name".devservices.volumes コンテナにマッピングされるボリューム。 マップキーはホストの場所に対応します。マップ値はコンテナの場所です。ホストの場所が "classpath:" で始まる場合、マッピングは読み取り専用権限でクラスパスからリソースをロードします。 ファイルシステムの場所を使用する場合、ボリュームは読み取り/書き込み権限で生成されるため、ファイルシステムでデータの損失や変更が発生する可能性があります。 プロバイダーが H2 や Derby などのコンテナベースのデータベースでない場合、これは効果がありません。 環境変数: QUARKUS_DATASOURCE_DEVSERVICES_VOLUMES	Map<String, String>	

<p> quarkus.datasource.devservices.reuse</p> <p>quarkus.datasource."datasource-name".devservices.reuse</p> <p>dev モードセッションまたはテストスイートの実行後に Dev Service コンテナを実行したままにして、次の dev モードセッションまたはテストスイートの実行で再利用するかどうか。</p> <p>dev モードセッションまたはテストスイート実行中、Quarkus は、設定 (ユーザー名、パスワード、環境、ポートバインディングなど) が変更されていない限り、常に Dev Services を再利用します。この機能は、Quarkus が実行していないときにコンテナを実行したままにして、実行間で再利用できるようにすることを目的としています。</p> <div data-bbox="164 613 1166 1025" style="background-color: #fff9c4; padding: 10px; border: 1px solid #ccc;"> <p> 警告</p> <p>この機能は、testcontainers.properties で明示的に有効にする必要があります。dev モードおよびテストでデータ初期化を設定する方法の変更が必要となる場合があります、コンテナが無期限に実行されたままになり、手動による停止および削除が必要になることがあります。詳細は、このドキュメントのセクションを参照してください。</p> </div> <p>この設定プロパティはデフォルトで true に設定されているため、testcontainers.properties で有効にしたが、Quarkus アプリケーションまたはデータソースの一部にのみ使用したい場合は、再利用を 無効にする と便利です。</p> <p>環境変数: QUARKUS_DATASOURCE_DEVSERVICES_REUSE</p>	boolean	true
<p> quarkus.datasource.health-exclude</p> <p>quarkus.datasource."datasource-name".health-exclude</p> <p>データソースの一般的なヘルスチェックが有効になっている場合に、この特定のデータソースをヘルスチェックから除外するかどうか。</p> <p>デフォルトでは、ヘルスチェックには設定されているすべてのデータソースが含まれます (有効になっている場合)。</p> <p>環境変数: QUARKUS_DATASOURCE_HEALTH_EXCLUDE</p>	boolean	false

<p>quarkus.datasource.active</p> <p>quarkus.datasource."datasource-name".active</p> <p>実行時にこのデータソースをアクティブにするかどうか。</p> <p>ドキュメントのこのセクション を参照してください。</p> <p>データソースがアクティブでない場合、アプリケーションでは起動されず、対応するデータソース CDI Bean へのアクセスは失敗します。つまり、特にこのデータソースのコンシューマー (Hibernate ORM 永続性ユニットなど) は、非アクティブでない限り起動に失敗します。</p> <p>環境変数: QUARKUS_DATASOURCE_ACTIVE</p>	boolean	true
<p>quarkus.datasource.username</p> <p>quarkus.datasource."datasource-name".username</p> <p>データソースのユーザー名</p> <p>環境変数: QUARKUS_DATASOURCE_USERNAME</p>	string	
<p>quarkus.datasource.password</p> <p>quarkus.datasource."datasource-name".password</p> <p>データソースのパスワード</p> <p>環境変数: QUARKUS_DATASOURCE_PASSWORD</p>	string	
<p>quarkus.datasource.credentials-provider</p> <p>quarkus.datasource."datasource-name".credentials-provider</p> <p>認証情報プロバイダー名</p> <p>環境変数: QUARKUS_DATASOURCE_CREDENTIALS_PROVIDER</p>	string	
<p>quarkus.datasource.credentials-provider-name</p> <p>quarkus.datasource."datasource-name".credentials-provider-name</p> <p>認証情報プロバイダーの Bean 名。</p> <p>これは、CredentialsProvider を実装する Bean の名前 (例:@Named) です。認証情報プロバイダー Bean が複数存在する場合に、それを選択するために使用されます。使用可能な認証情報プロバイダーが1つだけの場合は不要です。</p> <p>Vault の場合、認証情報プロバイダー Bean 名は vault-credentials-provider です。</p> <p>環境変数: QUARKUS_DATASOURCE_CREDENTIALS_PROVIDER_NAME</p>	string	

1.4.2. JDBC 設定リファレンス

 ビルド時に固定された設定プロパティ: その他の設定プロパティはすべて実行時にオーバーライド可能

設定プロパティ	型	デフォルト
<p> quarkus.datasource.jdbc</p> <p>quarkus.datasource."datasource-name".jdbc</p> <p>このデータソースの JDBC データソースを作成する場合。</p> <p>環境変数: QUARKUS_DATASOURCE_JDBC</p>	boolean	true
<p> quarkus.datasource.jdbc.driver</p> <p>quarkus.datasource."datasource-name".jdbc.driver</p> <p>データソースドライバーのクラス名</p> <p>環境変数: QUARKUS_DATASOURCE_JDBC_DRIVER</p>	string	
<p> quarkus.datasource.jdbc.transactions</p> <p>quarkus.datasource."datasource-name".jdbc.transactions</p> <p>通常の JDBC トランザクション、XA を使用するか、すべてのトランザクション機能を無効にするか。</p> <p>XA を有効にする場合は、javax.sql.XADataSource を実装するドライバーが必要です。</p> <p>環境変数: QUARKUS_DATASOURCE_JDBC_TRANSACTIONS</p>	enabled, xa, disabled	enabled
<p> quarkus.datasource.jdbc.enable-metrics</p> <p>quarkus.datasource."datasource-name".jdbc.enable-metrics</p> <p>データソースメトリクスの収集を有効にします。指定しない場合、メトリクスエクステンションがアクティブであれば、デフォルトでメトリクス収集が有効になります。</p> <p>環境変数: QUARKUS_DATASOURCE_JDBC_ENABLE_METRICS</p>	boolean	
<p> quarkus.datasource.jdbc.tracing</p> <p>quarkus.datasource."datasource-name".jdbc.tracing</p> <p>JDBC トレーシングを有効にします。デフォルトでは無効になっています。</p> <p>環境変数: QUARKUS_DATASOURCE_JDBC_TRACING</p>	boolean	false

 quarkus.datasource.jdbc.telemetry quarkus.datasource."datasource-name".jdbc.telemetry OpenTelemetry JDBC インストルメンテーションを有効にします。 環境変数: QUARKUS_DATASOURCE_JDBC_TELEMETRY	boolean	false
quarkus.datasource.jdbc.url quarkus.datasource."datasource-name".jdbc.url データソースの URL 環境変数: QUARKUS_DATASOURCE_JDBC_URL	string	
quarkus.datasource.jdbc.initial-size quarkus.datasource."datasource-name".jdbc.initial-size プールの初期サイズ。通常、初期サイズは少なくとも最小サイズに一致するように設定する必要がありますが、強制ではありません。これは、起動時に接続の遅延初期化を優先するアーキテクチャーを可能にし、一方で起動後に最小プールサイズを維持できるようにするためです。 環境変数: QUARKUS_DATASOURCE_JDBC_INITIAL_SIZE	int	
quarkus.datasource.jdbc.min-size quarkus.datasource."datasource-name".jdbc.min-size データソースプールの最小サイズ 環境変数: QUARKUS_DATASOURCE_JDBC_MIN_SIZE	int	0
quarkus.datasource.jdbc.max-size quarkus.datasource."datasource-name".jdbc.max-size データソースプールの最大サイズ 環境変数: QUARKUS_DATASOURCE_JDBC_MAX_SIZE	int	20
quarkus.datasource.jdbc.background-validation-interval quarkus.datasource."datasource-name".jdbc.background-validation-interval バックグラウンドでアイドル状態の接続を検証する間隔。 バックグラウンド検証を無効にするには、 0 に設定します。 環境変数: QUARKUS_DATASOURCE_JDBC_BACKGROUND_VALIDATION_INTERVAL	Duration 	2M

<p>quarkus.datasource.jdbc.foreground-validation-interval</p> <p>quarkus.datasource."datasource-name".jdbc.foreground-validation-interval</p> <p>指定された間隔よりも長くアイドル状態になっている接続に対し、フォアグラウンド検証を実行します。</p> <p>環境変数: QUARKUS_DATASOURCE_JDBC_FOREGROUND_VALIDATION_INTERVAL</p>	<p>Duratio n </p>	
<p>quarkus.datasource.jdbc.acquisition-timeout</p> <p>quarkus.datasource."datasource-name".jdbc.acquisition-timeout</p> <p>新しい接続の取得をキャンセルするまでのタイムアウト</p> <p>環境変数: QUARKUS_DATASOURCE_JDBC_ACQUISITION_TIMEOUT</p>	<p>Duratio n </p>	5S
<p>quarkus.datasource.jdbc.leak-detection-interval</p> <p>quarkus.datasource."datasource-name".jdbc.leak-detection-interval</p> <p>接続リークをチェックする間隔。</p> <p>環境変数: QUARKUS_DATASOURCE_JDBC_LEAK_DETECTION_INTERVAL</p>	<p>Duratio n </p>	デフォルトでは無効になっています。
<p>quarkus.datasource.jdbc.idle-removal-interval</p> <p>quarkus.datasource."datasource-name".jdbc.idle-removal-interval</p> <p>アイドル状態の接続の削除を試行する間隔。</p> <p>環境変数: QUARKUS_DATASOURCE_JDBC_IDLE_REMOVAL_INTERVAL</p>	<p>Duratio n </p>	5 M
<p>quarkus.datasource.jdbc.max-lifetime</p> <p>quarkus.datasource."datasource-name".jdbc.max-lifetime</p> <p>接続の最大存続期間。</p> <p>環境変数: QUARKUS_DATASOURCE_JDBC_MAX_LIFETIME</p>	<p>Duratio n </p>	デフォルトでは、接続の存続時間に制限はありません。

<p>quarkus.datasource.jdbc.transaction-isolation-level</p> <p>quarkus.datasource."datasource-name".jdbc.transaction-isolation-level</p> <p>トランザクション分離レベル。</p> <p>環境変数: QUARKUS_DATASOURCE_JDBC_TRANSACTION_ISOLATION_LEVEL</p>	<p>undefined, none, read-uncommitted, read-committed, repeatable-read, serializable</p>	
<p>quarkus.datasource.jdbc.extended-leak-report</p> <p>quarkus.datasource."datasource-name".jdbc.extended-leak-report</p> <p>リークされた接続に関する追加のトラブルシューティング情報を収集して表示します。</p> <p>環境変数: QUARKUS_DATASOURCE_JDBC_EXTENDED_LEAK_REPORT</p>	<p>boolean</p>	<p>false</p>
<p>quarkus.datasource.jdbc.flush-on-close</p> <p>quarkus.datasource."datasource-name".jdbc.flush-on-close</p> <p>プールに戻すときに接続をフラッシュできるようにします。デフォルトでは有効になっていません。</p> <p>環境変数: QUARKUS_DATASOURCE_JDBC_FLUSH_ON_CLOSE</p>	<p>boolean</p>	<p>false</p>
<p>quarkus.datasource.jdbc.detect-statement-leaks</p> <p>quarkus.datasource."datasource-name".jdbc.detect-statement-leaks</p> <p>有効にすると、開いているすべてのステートメントがアプリケーションにより閉じられないまま接続がプールに戻された場合に、Agroal は警告を生成できるようになります。これは、オープン接続の追跡とは関係ありません。パフォーマンスを最大化するために無効にしますが、リークが発生していないという高い信頼性がある場合に限りません。</p> <p>環境変数: QUARKUS_DATASOURCE_JDBC_DETECT_STATEMENT_LEAKS</p>	<p>boolean</p>	<p>true</p>
<p>quarkus.datasource.jdbc.new-connection-sql</p> <p>quarkus.datasource."datasource-name".jdbc.new-connection-sql</p> <p>接続の初回使用時に実行されるクエリ。</p> <p>環境変数: QUARKUS_DATASOURCE_JDBC_NEW_CONNECTION_SQL</p>	<p>string</p>	

<p>quarkus.datasource.jdbc.validation-query-sql</p> <p>quarkus.datasource."datasource-name".jdbc.validation-query-sql</p> <p>接続を検証するために実行されるクエリー。</p> <p>環境変数: QUARKUS_DATASOURCE_JDBC_VALIDATION_QUERY_SQL</p>	string	
<p>quarkus.datasource.jdbc.pooling-enabled</p> <p>quarkus.datasource."datasource-name".jdbc.pooling-enabled</p> <p>接続の再利用を防ぐためにプーリングを無効にします。外部プールが接続のライフサイクルを管理する場合に使用します。</p> <p>環境変数: QUARKUS_DATASOURCE_JDBC_POOLING_ENABLED</p>	boolean	true
<p>quarkus.datasource.jdbc.transaction-requirement</p> <p>quarkus.datasource."datasource-name".jdbc.transaction-requirement</p> <p>接続の取得時にアクティブなトランザクションを要求します。実稼働環境で推奨されま す。警告: 一部のエクステンションは、スキーマ更新やスキーマ検証などのトランザク ションを保持せずに接続を取得します。その場合、これを STRICT に設定すると失敗す る可能性があります。</p> <p>環境変数: QUARKUS_DATASOURCE_JDBC_TRANSACTION_REQUIREMENT</p>	off, warn, strict	
<p>quarkus.datasource.jdbc.additional-jdbc-properties</p> <p>quarkus.datasource."datasource-name".jdbc.additional-jdbc-properties</p> <p>新しい接続の作成時に JDBC ドライバーに渡される、その他の未指定のプロパ ティー。</p> <p>環境変数: QUARKUS_DATASOURCE_JDBC_ADDITIONAL_JDBC_PROPERTIES</p>	Map< String ,Strin g>	

<p>quarkus.datasource.jdbc.tracing.enabled</p> <p>quarkus.datasource."datasource-name".jdbc.tracing.enabled</p> <p>JDBC トレーシングを有効にします。</p> <p>環境変数: QUARKUS_DATASOURCE_JDBC_TRACING_ENABLED</p>	boolean	<p>quarkus.datasource.jdbc.tracing=false の場合は false、quarkus.datasource.jdbc.tracing=true の場合は true</p>
<p>quarkus.datasource.jdbc.tracing.trace-with-active-span-only</p> <p>quarkus.datasource."datasource-name".jdbc.tracing.trace-with-active-span-only</p> <p>アクティブなスパンを持つコールのみをトレースします</p> <p>環境変数: QUARKUS_DATASOURCE_JDBC_TRACING_TRACE_WITH_ACTIVE_SPAN_ONLY</p>	boolean	false

<p>quarkus.datasource.jdbc.tracing.ignore-for-tracing</p> <p>quarkus.datasource."datasource-name".jdbc.tracing.ignore-for-tracing</p> <p>特定のクエリーがトレースされないようにします。</p> <p>環境変数: QUARKUS_DATASOURCE_JDBC_TRACING_IGNORE_FOR_TRACING</p>	string	<p>特定のクエリーをトレースしないよう無視します。複数のクエリーを、セミコロンの区切って指定できます。二重引用符は\でエスケープする必要があります。</p>
<p>quarkus.datasource.jdbc.telemetry.enabled</p> <p>quarkus.datasource."datasource-name".jdbc.telemetry.enabled</p> <p>OpenTelemetry JDBC インstrumentationを有効にします。</p> <p>環境変数: QUARKUS_DATASOURCE_JDBC_TELEMETRY_ENABLED</p>	boolean	<p>quarkus.datasource.jdbc.telemetry=false の場合は false、quarkus.datasource.jdbc.telemetry=true の場合は true</p>



DURATION のフォーマット

duration の値を書き込むには、標準の **java.time.Duration** フォーマットを使用します。詳細は、[Duration#parse\(\) Java API ドキュメント](#) を参照してください。

数字で始まる簡略化されたフォーマットも使用できます。

- 値が数値のみの場合は、秒単位の時間を表します。
- 数字の後に **ms** が続く値は、ミリ秒単位の時間を表します。

その他の場合は、解析のために簡略化されたフォーマットが **java.time.Duration** フォーマットに変換されます。

- 数字の後に **h**、**m**、または **s** が続く値には、接頭辞 **PT** が付きます。
- 数字の後に **d** が続く値は、接頭辞 **P** が付きます。

1.4.3. JDBC URL リファレンス

サポートされている各データベースには、異なる JDBC URL 設定オプションが含まれています。次のセクションでは、各データベース URL の概要と公式ドキュメントへのリンクを示します。

1.4.3.1. DB2

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

例

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

URL 構文とその他のサポートされるオプションの詳細は、[公式ドキュメント](#) を参照してください。

1.4.3.2. Derby

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

例

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

Derby は、ファイルに基づきサーバーとして実行することも、完全にインメモリーで実行することもできる組み込みデータベースです。上記のオプションはすべて利用可能です。

詳細は、[公式ドキュメント](#) を参照してください。

1.4.3.3. H2

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

例

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

H2 は、組み込みモードまたはサーバーモードで実行できるデータベースです。ファイルストレージを使用することも、完全にインメモリーで実行することも可能です。上記のオプションはすべて利用可能です。

詳細は、[公式ドキュメント](#) を参照してください。

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))]
```

例

```
jdbc:mariadb://localhost:3306/test
```

詳細は、[公式ドキュメント](#) を参照してください。

1.4.3.5. Microsoft SQL サーバー

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

例

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

Microsoft SQL Server JDBC ドライバーは、基本的に他のドライバーと同じように動作します。

詳細は、[公式ドキュメント](#) を参照してください。

1.4.3.6. MySQL

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

例

```
jdbc:mysql://localhost:3306/test
```

詳細は、[公式ドキュメント](#) を参照してください。

1.4.3.6.1. MySQL の制限事項

Quarkus アプリケーションをネイティブイメージにコンパイルする場合、JMX および Oracle Cloud Infrastructure (OCI) のインテグレーションに対する MySQL サポートは、GraalVM ネイティブイメージと互換性がないため無効になります。

- ネイティブモードで実行する場合に JMX サポートがないのは当然の結果であり、解決される見込みはありません。
- OCI とのインテグレーションはサポートされていません。

1.4.3.7. Oracle

```
jdbc:oracle:driver_type:@database_specifier
```

例

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

詳細は、[公式ドキュメント](#) を参照してください。

1.4.3.8. PostgreSQL

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

例

```
jdbc:postgresql://localhost/test
```

各部分のデフォルトは次のとおりです。

host

localhost

port

5432

database

ユーザー名と同じ名前

追加パラメーターの詳細は、[公式ドキュメント](#) を参照してください。

1.4.4. Quarkus エクステンションとデータベースドライバーのリファレンス

次の表は、ビルトイン **db-kind** 値、対応する Quarkus エクステンション、およびそれらのエクステンションで使用される JDBC ドライバーを示しています。

ビルトインデータソースの種類の内いずれかを使用する場合、JDBC ドライバーとリアクティブドライバーはこの表の値と一致するように自動的に解決されます。

表1.1 データベースプラットフォームの種類と JDBC ドライバーのマッピング

データベースの種類	Quarkus エクステンション	ドライバー
db2	quarkus-jdbc-db2	<ul style="list-style-type: none"> JDBC: com.ibm.db2.jcc.DB2Driver XA: com.ibm.db2.jcc.DB2XADataSource
derby	quarkus-jdbc-derby	<ul style="list-style-type: none"> JDBC: org.apache.derby.jdbc.ClientDriver XA: org.apache.derby.jdbc.ClientXADataSource
h2	quarkus-jdbc-h2	<ul style="list-style-type: none"> JDBC: org.h2.Driver XA: org.h2.jdbcx.JdbcDataSource

データベースの種類	Quarkus エクステンション	ドライバー
maria db	quarkus-jdbc-mariadb	<ul style="list-style-type: none"> JDBC: org.mariadb.jdbc.Driver XA: org.mariadb.jdbc.MySQLDataSource
mssql	quarkus-jdbc-mssql	<ul style="list-style-type: none"> JDBC: com.microsoft.sqlserver.jdbc.SQLServerDriver XA: com.microsoft.sqlserver.jdbc.SQLServerXADataSource
mysql	quarkus-jdbc-mysql	<ul style="list-style-type: none"> JDBC: com.mysql.cj.jdbc.Driver XA: com.mysql.cj.jdbc.MysqlXADataSource
oracle	quarkus-jdbc-oracle	<ul style="list-style-type: none"> JDBC: oracle.jdbc.driver.OracleDriver XA: oracle.jdbc.xa.client.OracleXADataSource
postgres	quarkus-jdbc-postgresql	<ul style="list-style-type: none"> JDBC: org.postgresql.Driver XA: org.postgresql.xa.PGXDataSource

表1.2 データベースの種類とリアクティブドライバーのマッピング

データベースの種類	Quarkus エクステンション	ドライバー
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
postgres	reactive-pg-client	io.vertx.pgclient.spi.PgDriver
db2	reactive-db2-client	io.vertx.db2client.spi.DB2Driver

ヒント

ほとんどの場合、この自動解決を適用できるため、ドライバーの設定は必要ありません。

1.4.5. リアクティブデータソース設定リファレンス

 ビルド時に固定された設定プロパティ: その他の設定プロパティはすべて実行時にオーバーライド可能

設定プロパティ	型	デフォルト
 quarkus.datasource.reactive このデータソースに対してリアクティブデータソースを作成する場合。 環境変数: QUARKUS_DATASOURCE_REACTIVE	boolean	true
quarkus.datasource.reactive.cache-prepared-statements 準備されたステートメントをクライアント側でキャッシュするかどうか。 環境変数: QUARKUS_DATASOURCE_REACTIVE_CACHE_PREPARED_STATEMENTS	boolean	false
quarkus.datasource.reactive.url データソースの URL。 複数の値が設定されている場合、このデータソースは1つのサーバーではなくサーバーリストを含むプールを作成します。プールは、接続確立時のサーバー選択にラウンドロビン負荷分散を使用します。一部のドライバーは、このコンテキストで複数の値をサポートできない場合があることに注意してください。 環境変数: QUARKUS_DATASOURCE_REACTIVE_URL	文字列のリスト	
quarkus.datasource.reactive.max-size データソースプールの最大サイズ。 環境変数: QUARKUS_DATASOURCE_REACTIVE_MAX_SIZE	int	20

<p>quarkus.datasource.reactive.event-loop-size</p> <p>新しい接続オブジェクトが作成されると、プールはそれにイベントループを割り当てます。</p> <p>#event-loop-size が正の値に厳密に設定されている場合、プールは指定された数のイベントループをラウンドロビン方式で割り当てます。デフォルトでは、Quarkus によって設定または計算されたイベントループの数を使用されます。#event-loop-size がゼロまたは負の値に設定されている場合、プールは現在のイベントループを新しい接続に割り当てます。</p> <p>環境変数: QUARKUS_DATASOURCE_REACTIVE_EVENT_LOOP_SIZE</p>	int	
<p>quarkus.datasource.reactive.trust-all</p> <p>すべてのサーバー証明書を信頼するかどうか。</p> <p>環境変数: QUARKUS_DATASOURCE_REACTIVE_TRUST_ALL</p>	boolean	false
<p>quarkus.datasource.reactive.trust-certificate-pem</p> <p>デフォルトで、PEM Trust 設定は無効になっています。</p> <p>環境変数: QUARKUS_DATASOURCE_REACTIVE_TRUST_CERTIFICATE_PEM</p>	boolean	false
<p>quarkus.datasource.reactive.trust-certificate-pem.certs</p> <p>信頼証明書ファイルのコンマ区切りリスト (Pem フォーマット)。</p> <p>環境変数: QUARKUS_DATASOURCE_REACTIVE_TRUST_CERTIFICATE_PEM_CERTS</p>	文字列のリスト	
<p>quarkus.datasource.reactive.trust-certificate-jks</p> <p>デフォルトで、JKS 設定は無効になっています。</p> <p>環境変数: QUARKUS_DATASOURCE_REACTIVE_TRUST_CERTIFICATE_JKS</p>	boolean	false
<p>quarkus.datasource.reactive.trust-certificate-jks.path</p> <p>キーファイルのパス (JKS フォーマット)。</p> <p>環境変数: QUARKUS_DATASOURCE_REACTIVE_TRUST_CERTIFICATE_JKS_PATH</p>	string	

<p>quarkus.datasource.reactive.trust-certificate-jks.password</p> <p>キーファイルのパスワード。</p> <p>環境変数: QUARKUS_DATASOURCE_REACTIVE_TRUST_CERTIFICATE_JKS_PASSWORD</p>	string	
<p>quarkus.datasource.reactive.trust-certificate-pfx</p> <p>デフォルトで、PFX 設定は無効になっています。</p> <p>環境変数: QUARKUS_DATASOURCE_REACTIVE_TRUST_CERTIFICATE_PFX</p>	boolean	false
<p>quarkus.datasource.reactive.trust-certificate-pfx.path</p> <p>キーファイルへのパス (PFX フォーマット)。</p> <p>環境変数: QUARKUS_DATASOURCE_REACTIVE_TRUST_CERTIFICATE_PFX_PATH</p>	string	
<p>quarkus.datasource.reactive.trust-certificate-pfx.password</p> <p>キーのパスワード。</p> <p>環境変数: QUARKUS_DATASOURCE_REACTIVE_TRUST_CERTIFICATE_PFX_PASSWORD</p>	string	
<p>quarkus.datasource.reactive.key-certificate-pem</p> <p>デフォルトで、PEM キー/証明書設定は無効になっています。</p> <p>環境変数: QUARKUS_DATASOURCE_REACTIVE_KEY_CERTIFICATE_PEM</p>	boolean	false
<p>quarkus.datasource.reactive.key-certificate-pem.keys</p> <p>キーファイルへのパスのコンマ区切りリスト (Pem フォーマット)。</p> <p>環境変数: QUARKUS_DATASOURCE_REACTIVE_KEY_CERTIFICATE_PEM_KEYS</p>	文字列のリスト	
<p>quarkus.datasource.reactive.key-certificate-pem.certs</p> <p>証明書ファイルへのパスのコンマ区切りリスト (Pem フォーマット)。</p> <p>環境変数: QUARKUS_DATASOURCE_REACTIVE_KEY_CERTIFICATE_PEM_CERTS</p>	文字列のリスト	

<p>quarkus.datasource.reactive.key-certificate-jks</p> <p>デフォルトで、JKS 設定は無効になっています。</p> <p>環境変数: QUARKUS_DATASOURCE_REACTIVE_KEY_CERTIFICATE_JKS</p>	boolean	false
<p>quarkus.datasource.reactive.key-certificate-jks.path</p> <p>キーファイルのパス (JKS フォーマット)。</p> <p>環境変数: QUARKUS_DATASOURCE_REACTIVE_KEY_CERTIFICATE_JKS_PATH</p>	string	
<p>quarkus.datasource.reactive.key-certificate-jks.password</p> <p>キーファイルのパスワード。</p> <p>環境変数: QUARKUS_DATASOURCE_REACTIVE_KEY_CERTIFICATE_JKS_PASSWORD</p>	string	
<p>quarkus.datasource.reactive.key-certificate-pfx</p> <p>デフォルトで、PFX 設定は無効になっています。</p> <p>環境変数: QUARKUS_DATASOURCE_REACTIVE_KEY_CERTIFICATE_PFX</p>	boolean	false
<p>quarkus.datasource.reactive.key-certificate-pfx.path</p> <p>キーファイルへのパス (PFX フォーマット)。</p> <p>環境変数: QUARKUS_DATASOURCE_REACTIVE_KEY_CERTIFICATE_PFX_PATH</p>	string	
<p>quarkus.datasource.reactive.key-certificate-pfx.password</p> <p>キーのパスワード。</p> <p>環境変数: QUARKUS_DATASOURCE_REACTIVE_KEY_CERTIFICATE_PFX_PASSWORD</p>	string	
<p>quarkus.datasource.reactive.reconnect-attempts</p> <p>プールされた接続が初回試行時に確立できなかった場合の再接続の試行回数。</p> <p>環境変数: QUARKUS_DATASOURCE_REACTIVE_RECONNECT_ATTEMPTS</p>	int	0
<p>quarkus.datasource.reactive.reconnect-interval</p> <p>プールされた接続が初回試行時に確立できなかった場合の再接続の試行間隔。</p> <p>環境変数: QUARKUS_DATASOURCE_REACTIVE_RECONNECT_INTERVAL</p>	Duration 	PT1S

<p>quarkus.datasource.reactive.hostname-verification-algorithm</p> <p>サーバーのアイデンティティをチェックする必要がある場合に使用するホスト名検証アルゴリズム。HTTPS、LDAPS、または NONE にする必要があります。NONE はデフォルト値であり、検証を無効にします。</p> <p>環境変数: QUARKUS_DATASOURCE_REACTIVE_HOSTNAME_VERIFICATION_ALGORITHM</p>	string	NONE
<p>quarkus.datasource.reactive.idle-timeout</p> <p>プール内で未使用の接続が閉じられるまでの、未使用状態の最大継続時間。</p> <p>環境変数: QUARKUS_DATASOURCE_REACTIVE_IDLE_TIMEOUT</p>	Duration n 	タイムアウトなし
<p>quarkus.datasource.reactive.max-lifetime</p> <p>接続がプール内に留まる最大時間。この時間が経過すると接続は戻ったときに閉じられ、必要に応じて置き換えられます。</p> <p>環境変数: QUARKUS_DATASOURCE_REACTIVE_MAX_LIFETIME</p>	Duration n 	タイムアウトなし
<p>quarkus.datasource.reactive.shared</p> <p>データソース間でプールを共有する場合は true に設定します。複数の共有プールを名前前で区別できます。特定の名前を設定しない場合、__vertx.DEFAULT の名前が使用されます。</p> <p>環境変数: QUARKUS_DATASOURCE_REACTIVE_SHARED</p>	boolean	false
<p>quarkus.datasource.reactive.name</p> <p>プール名を設定します。この名前は、プールがデータソース間で共有される場合に使用され、それ以外では無視されます。</p> <p>環境変数: QUARKUS_DATASOURCE_REACTIVE_NAME</p>	string	
<p>quarkus.datasource.reactive.additional-properties</p> <p>新しい接続の開始時に、リアクティブ SQL クライアントを介してデータベースに直接渡されるその他の未指定のプロパティ。</p> <p>環境変数: QUARKUS_DATASOURCE_REACTIVE_ADDITIONAL_PROPERTIES</p>	Map<String, String>	
<p>追加の名前付きデータソース</p>	型	デフォルト
<p> quarkus.datasource."datasource-name".reactive</p> <p>このデータソースに対してリアクティブデータソースを作成する場合。</p> <p>環境変数: QUARKUS_DATASOURCE__DATASOURCE_NAME__REACTIVE</p>	boolean	true

<p>quarkus.datasource."datasource-name".reactive.cache-prepared-statements</p> <p>準備されたステートメントをクライアント側でキャッシュするかどうか。</p> <p>環境変数: QUARKUS_DATASOURCE__DATASOURCE_NAME__REACTIVE_CACHE_PREPARED_STATEMENTS</p>	boolean	false
<p>quarkus.datasource."datasource-name".reactive.url</p> <p>データソースの URL。</p> <p>複数の値が設定されている場合、このデータソースは1つのサーバーではなくサーバーリストを含むプールを作成します。プールは、接続確立時のサーバー選択にラウンドロビン負荷分散を使用します。一部のドライバーは、このコンテキストで複数の値をサポートできない場合があることに注意してください。</p> <p>環境変数: QUARKUS_DATASOURCE__DATASOURCE_NAME__REACTIVE_URL</p>	文字列のリスト	
<p>quarkus.datasource."datasource-name".reactive.max-size</p> <p>データソースプールの最大サイズ。</p> <p>環境変数: QUARKUS_DATASOURCE__DATASOURCE_NAME__REACTIVE_MAX_SIZE</p>	int	20
<p>quarkus.datasource."datasource-name".reactive.event-loop-size</p> <p>新しい接続オブジェクトが作成されると、プールはそれにイベントループを割り当てます。</p> <p>#event-loop-size が正の値に厳密に設定されている場合、プールは指定された数のイベントループをラウンドロビン方式で割り当てます。デフォルトでは、Quarkus によって設定または計算されたイベントループの数を使用されます。#event-loop-size がゼロまたは負の値に設定されている場合、プールは現在のイベントループを新しい接続に割り当てます。</p> <p>環境変数: QUARKUS_DATASOURCE__DATASOURCE_NAME__REACTIVE_EVENT_LOOP_SIZE</p>	int	
<p>quarkus.datasource."datasource-name".reactive.trust-all</p> <p>すべてのサーバー証明書を信頼するかどうか。</p> <p>環境変数: QUARKUS_DATASOURCE__DATASOURCE_NAME__REACTIVE_TRUST_ALL</p>	boolean	false

<p>quarkus.datasource."datasource-name".reactive.trust-certificate-pem</p> <p>デフォルトで、PEM Trust 設定は無効になっています。</p> <p>環境変数: QUARKUS_DATASOURCE__DATASOURCE_NAME__REACTIVE_TRUST_CERTIFICATE_PEM</p>	boolean	false
<p>quarkus.datasource."datasource-name".reactive.trust-certificate-pem.certs</p> <p>信頼証明書ファイルのコンマ区切りリスト (Pem フォーマット)。</p> <p>環境変数: QUARKUS_DATASOURCE__DATASOURCE_NAME__REACTIVE_TRUST_CERTIFICATE_PEM_CERTS</p>	文字列のリスト	
<p>quarkus.datasource."datasource-name".reactive.trust-certificate-jks</p> <p>デフォルトで、JKS 設定は無効になっています。</p> <p>環境変数: QUARKUS_DATASOURCE__DATASOURCE_NAME__REACTIVE_TRUST_CERTIFICATE_JKS</p>	boolean	false
<p>quarkus.datasource."datasource-name".reactive.trust-certificate-jks.path</p> <p>キーファイルのパス (JKS フォーマット)。</p> <p>環境変数: QUARKUS_DATASOURCE__DATASOURCE_NAME__REACTIVE_TRUST_CERTIFICATE_JKS_PATH</p>	string	
<p>quarkus.datasource."datasource-name".reactive.trust-certificate-jks.password</p> <p>キーファイルのパスワード。</p> <p>環境変数: QUARKUS_DATASOURCE__DATASOURCE_NAME__REACTIVE_TRUST_CERTIFICATE_JKS_PASSWORD</p>	string	
<p>quarkus.datasource."datasource-name".reactive.trust-certificate-pfx</p> <p>デフォルトで、PFX 設定は無効になっています。</p> <p>環境変数: QUARKUS_DATASOURCE__DATASOURCE_NAME__REACTIVE_TRUST_CERTIFICATE_PFX</p>	boolean	false

<p>quarkus.datasource."datasource-name".reactive.trust-certificate-pfx.path</p> <p>キーファイルへのパス (PFX フォーマット)。</p> <p>環境変数: QUARKUS_DATASOURCE__DATASOURCE_NAME__REACTIVE_TRUST_CERTIFICATE_PFX_PATH</p>	string	
<p>quarkus.datasource."datasource-name".reactive.trust-certificate-pfx.password</p> <p>キーのパスワード。</p> <p>環境変数: QUARKUS_DATASOURCE__DATASOURCE_NAME__REACTIVE_TRUST_CERTIFICATE_PFX_PASSWORD</p>	string	
<p>quarkus.datasource."datasource-name".reactive.key-certificate-pem</p> <p>デフォルトで、PEM キー/証明書設定は無効になっています。</p> <p>環境変数: QUARKUS_DATASOURCE__DATASOURCE_NAME__REACTIVE_KEY_CERTIFICATE_PEM</p>	boolean	false
<p>quarkus.datasource."datasource-name".reactive.key-certificate-pem.keys</p> <p>キーファイルへのパスのコンマ区切りリスト (Pem フォーマット)。</p> <p>環境変数: QUARKUS_DATASOURCE__DATASOURCE_NAME__REACTIVE_KEY_CERTIFICATE_PEM_KEYS</p>	文字列のリスト	
<p>quarkus.datasource."datasource-name".reactive.key-certificate-pem.certs</p> <p>証明書ファイルへのパスのコンマ区切りリスト (Pem フォーマット)。</p> <p>環境変数: QUARKUS_DATASOURCE__DATASOURCE_NAME__REACTIVE_KEY_CERTIFICATE_PEM_CERTS</p>	文字列のリスト	
<p>quarkus.datasource."datasource-name".reactive.key-certificate-jks</p> <p>デフォルトで、JKS 設定は無効になっています。</p> <p>環境変数: QUARKUS_DATASOURCE__DATASOURCE_NAME__REACTIVE_KEY_CERTIFICATE_JKS</p>	boolean	false

<p>quarkus.datasource."datasource-name".reactive.key-certificate-jks.path</p> <p>キーファイルのパス (JKS フォーマット)。</p> <p>環境変数: QUARKUS_DATASOURCE__DATASOURCE_NAME__REACTIVE_KEY_CERTIFICATE_JKS_PATH</p>	string	
<p>quarkus.datasource."datasource-name".reactive.key-certificate-jks.password</p> <p>キーファイルのパスワード。</p> <p>環境変数: QUARKUS_DATASOURCE__DATASOURCE_NAME__REACTIVE_KEY_CERTIFICATE_JKS_PASSWORD</p>	string	
<p>quarkus.datasource."datasource-name".reactive.key-certificate-pfx</p> <p>デフォルトで、PFX 設定は無効になっています。</p> <p>環境変数: QUARKUS_DATASOURCE__DATASOURCE_NAME__REACTIVE_KEY_CERTIFICATE_PFX</p>	boolean	false
<p>quarkus.datasource."datasource-name".reactive.key-certificate-pfx.path</p> <p>キーファイルへのパス (PFX フォーマット)。</p> <p>環境変数: QUARKUS_DATASOURCE__DATASOURCE_NAME__REACTIVE_KEY_CERTIFICATE_PFX_PATH</p>	string	
<p>quarkus.datasource."datasource-name".reactive.key-certificate-pfx.password</p> <p>キーのパスワード。</p> <p>環境変数: QUARKUS_DATASOURCE__DATASOURCE_NAME__REACTIVE_KEY_CERTIFICATE_PFX_PASSWORD</p>	string	
<p>quarkus.datasource."datasource-name".reactive.reconnect-attempts</p> <p>プールされた接続が初回試行時に確立できなかった場合の再接続の試行回数。</p> <p>環境変数: QUARKUS_DATASOURCE__DATASOURCE_NAME__REACTIVE_RECONNECT_ATTEMPTS</p>	int	0

<p>quarkus.datasource."datasource-name".reactive.reconnect-interval</p> <p>プールされた接続が初回試行時に確立できなかった場合の再接続の試行間隔。</p> <p>環境変数: QUARKUS_DATASOURCE__DATASOURCE_NAME__REACTIVE_RECONNECT_INTERVAL</p>	Duration n 	PT1S
<p>quarkus.datasource."datasource-name".reactive.hostname-verification-algorithm</p> <p>サーバーのアイデンティティをチェックする必要がある場合に使用するホスト名検証アルゴリズム。HTTPS、LDAPS、または NONE にする必要があります。NONE はデフォルト値であり、検証を無効にします。</p> <p>環境変数: QUARKUS_DATASOURCE__DATASOURCE_NAME__REACTIVE_HOSTNAME_VERIFICATION_ALGORITHM</p>	string	NONE
<p>quarkus.datasource."datasource-name".reactive.idle-timeout</p> <p>プール内で未使用の接続が閉じられるまでの、未使用状態の最大継続時間。</p> <p>環境変数: QUARKUS_DATASOURCE__DATASOURCE_NAME__REACTIVE_IDLE_TIMEOUT</p>	Duration n 	タイムアウトなし
<p>quarkus.datasource."datasource-name".reactive.max-lifetime</p> <p>接続がプール内に留まる最大時間。この時間が経過すると接続は戻ったときに閉じられ、必要に応じて置き換えられます。</p> <p>環境変数: QUARKUS_DATASOURCE__DATASOURCE_NAME__REACTIVE_MAX_LIFETIME</p>	Duration n 	タイムアウトなし
<p>quarkus.datasource."datasource-name".reactive.shared</p> <p>データソース間でプールを共有する場合は true に設定します。複数の共有プールを名前前で区別できます。特定の名前を設定しない場合、__vertx.DEFAULT の名前が使用されます。</p> <p>環境変数: QUARKUS_DATASOURCE__DATASOURCE_NAME__REACTIVE_SHARED</p>	boolean n	false
<p>quarkus.datasource."datasource-name".reactive.name</p> <p>プール名を設定します。この名前は、プールがデータソース間で共有される場合に使用され、それ以外では無視されます。</p> <p>環境変数: QUARKUS_DATASOURCE__DATASOURCE_NAME__REACTIVE_NAME</p>	string	

quarkus.datasource."datasource-name".reactive.additional-properties	Map<String,String>
新しい接続の開始時に、リアクティブ SQL クライアントを介してデータベースに直接渡されるその他の未指定のプロパティ。	
環境変数: QUARKUS_DATASOURCE__DATASOURCE_NAME__REACTIVE_ADDITIONAL_PROPERTIES	



DURATION のフォーマット

duration の値を書き込むには、標準の **java.time.Duration** フォーマットを使用します。詳細は、[Duration#parse\(\) Java API ドキュメント](#) を参照してください。

数字で始まる簡略化されたフォーマットも使用できます。

- 値が数値のみの場合は、秒単位の時間を表します。
- 数字の後に **ms** が続く値は、ミリ秒単位の時間を表します。

その他の場合は、解析のために簡略化されたフォーマットが **java.time.Duration** フォーマットに変換されます。

- 数字の後に **h**、**m**、または **s** が続く値には、接頭辞 **PT** が付きます。
- 数字の後に **d** が続く値は、接頭辞 **P** が付きます。

1.4.5.1. リアクティブ DB2 設定

 ビルド時に固定された設定プロパティ: その他の設定プロパティはすべて実行時にオーバーライド可能

Databases	型	デフォルト
quarkus.datasource.reactive.db2.ssl quarkus.datasource."datasource-name".reactive.db2.ssl SSL/TLS が有効かどうか。 環境変数: QUARKUS_DATASOURCE_REACTIVE_DB2_SSL	boolean	false

1.4.5.2. リアクティブ MariaDB/MySQL 固有の設定

 ビルド時に固定された設定プロパティ: その他の設定プロパティはすべて実行時にオーバーライド可能

Additional named datasources	型	デフォルト

<p>quarkus.datasource.reactive.mysql.charset</p> <p>quarkus.datasource."datasource-name".reactive.mysql.charset</p> <p>接続の charset。</p> <p>環境変数: QUARKUS_DATASOURCE_REACTIVE_MYSQL_CHARSET</p>	string	
<p>quarkus.datasource.reactive.mysql.collation</p> <p>quarkus.datasource."datasource-name".reactive.mysql.collation</p> <p>接続の照合。</p> <p>環境変数: QUARKUS_DATASOURCE_REACTIVE_MYSQL_COLLATION</p>	string	
<p>quarkus.datasource.reactive.mysql.ssl-mode</p> <p>quarkus.datasource."datasource-name".reactive.mysql.ssl-mode</p> <p>サーバーへの接続に必要なセキュリティー状態。</p> <p>MySQL Reference Manual を参照してください。</p> <p>環境変数: QUARKUS_DATASOURCE_REACTIVE_MYSQL_SSL_MODE</p>	disabled, preferred, required, verify-ca, verify-identity	disabled
<p>quarkus.datasource.reactive.mysql.connection-timeout</p> <p>quarkus.datasource."datasource-name".reactive.mysql.connection-timeout</p> <p>SSH 接続タイムアウト (秒単位)</p> <p>環境変数: QUARKUS_DATASOURCE_REACTIVE_MYSQL_CONNECTION_TIMEOUT</p>	int	

<p>quarkus.datasource.reactive.mysql.authentication-plugin</p> <p>quarkus.datasource."datasource-name".reactive.mysql.authentication-plugin</p> <p>クライアントが使用する必要がある認証プラグイン。デフォルトでは、最初のハンドシェイクパケットでサーバーが指定したプラグイン名が使用されます。</p> <p>環境変数: QUARKUS_DATASOURCE_REACTIVE_MYSQL_AUTHENTICATION_PLUGIN</p>	<p>default, mysql-clearpassword, mysql-native-password, sha256-password, caching-sha2-password</p>	<p>default</p>
<p>quarkus.datasource.reactive.mysql.pipelining-limit</p> <p>quarkus.datasource."datasource-name".reactive.mysql.pipelining-limit</p> <p>パイプライン化できる未完了のデータベースコマンドの最大数。デフォルトでは、パイプライン化は無効になっています。</p> <p>環境変数: QUARKUS_DATASOURCE_REACTIVE_MYSQL_PIPELINING_LIMIT</p>	<p>int</p>	
<p>quarkus.datasource.reactive.mysql.use-affected-rows</p> <p>quarkus.datasource."datasource-name".reactive.mysql.use-affected-rows</p> <p>実際に変更された行数ではなく、UPDATE ステートメントの WHERE 句に一致する行数を返すかどうか。</p> <p>環境変数: QUARKUS_DATASOURCE_REACTIVE_MYSQL_USE_AFFECTED_ROWS</p>	<p>boolean</p>	<p>false</p>

1.4.5.3. リアクティブ Microsoft SQL サーバー固有の設定



ビルド時に固定された設定プロパティ: その他の設定プロパティはすべて実行時にオーバーライド可能

Datasources	型	デフォルト
-------------	---	-------

<p>quarkus.datasource.reactive.mssql.packet-size</p> <p>quarkus.datasource."datasource-name".reactive.mssql.packet-size</p> <p>TDS パケットの必要なサイズ (バイト単位)。</p> <p>環境変数: QUARKUS_DATASOURCE_REACTIVE_MSSQL_PACKET_SIZE</p>	int	
<p>quarkus.datasource.reactive.mssql.ssl</p> <p>quarkus.datasource."datasource-name".reactive.mssql.ssl</p> <p>SSL/TLS が有効かどうか。</p> <p>環境変数: QUARKUS_DATASOURCE_REACTIVE_MSSQL_SSL</p>	boolean	false

1.4.5.4. リアクティブ Oracle 固有の設定



ビルド時に固定された設定プロパティ: その他の設定プロパティはすべて実行時にオーバーライド可能

Datasources	型	デフォルト

1.4.5.5. リアクティブ PostgreSQL 固有の設定



ビルド時に固定された設定プロパティ: その他の設定プロパティはすべて実行時にオーバーライド可能

Datasources	型	デフォルト
<p>quarkus.datasource.reactive.postgresql.pipelining-limit</p> <p>quarkus.datasource."datasource-name".reactive.postgresql.pipelining-limit</p> <p>パイプライン化できる未完了のデータベースコマンドの最大数。</p> <p>環境変数: QUARKUS_DATASOURCE_REACTIVE_POSTGRESQL_PIPELINING_LIMIT</p>	int	

<p>quarkus.datasource.reactive.postgresql.ssl-mode</p> <p>quarkus.datasource."datasource-name".reactive.postgresql.ssl-mode</p> <p>クライアントの SSL 動作モード。</p> <p>Protection Provided in Different Modes を参照してください。</p> <p>環境変数: QUARKUS_DATASOURCE_REACTIVE_POSTGRESQL_SSL_MODE</p>	<p>disabl e, allow, prefer, requir e, verify- ca, verify- full</p>	<p>disabl e</p>
<p>quarkus.datasource.reactive.postgresql.use-layer7-proxy</p> <p>quarkus.datasource."datasource-name".reactive.postgresql.use-layer7-proxy</p> <p>レベル7のプロキシは、実際のデータベースへの複数の接続でクエリーの負荷を分散できます。これが起こると、クライアントはセッションアフィニティーの欠如によって混乱し、ERROR: unnamed prepared statement does not exist (26000) のような望ましくないエラーが発生する可能性があります。 Using a level 7 proxy を参照してください。</p> <p>環境変数: QUARKUS_DATASOURCE_REACTIVE_POSTGRESQL_USE_LAYER7_PROXY</p>	<p>boolea n</p>	<p>false</p>

1.4.6. リアクティブデータソース URL リファレンス

1.4.6.1. DB2

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

例

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

現在、クライアントは次のパラメーターキーをサポートしています。

- **host**
- **port**
- **user**
- **password**
- **database**



注記

接続 URL でパラメーターを設定すると、デフォルトのプロパティーがオーバーライドされます。

1.4.6.2. Microsoft SQL サーバー

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

例

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

現在、クライアントは次のパラメーターキーをサポートしています。

- **host**
- **port**
- **user**
- **password**
- **database**



注記

接続 URL でパラメーターを設定すると、デフォルトのプロパティーがオーバーライドされます。

1.4.6.3. MySQL / MariaDB

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

例

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

現在、クライアントは次のパラメーターキーをサポートしています (大文字と小文字は区別されません)。

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



注記

接続 URL でパラメーターを設定すると、デフォルトのプロパティーがオーバーライドされます。

1.4.6.4. Oracle

1.4.6.4.1. EZConnect フォーマット

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

例

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

1.4.6.4.2. TNS エイリアスフォーマット

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

例

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

1.4.6.5. PostgreSQL

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

例

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

現在、クライアントは以下をサポートしています。

- パラメーターキー:
 - **host**
 - **port**
 - **user**
 - **password**
 - **dbname**
 - **sslmode**
- 以下のような追加プロパティ:
 - **application_name**
 - **fallback_application_name**
 - **search_path**
 - **options**



注記

接続 URL でパラメーターを設定すると、デフォルトのプロパティがオーバーライドされます。

