Red Hat build of Quarkus 1.11

Red Hat ビルドの Quarkus 1.11 の リリースノート

Last Updated: 2021-05-07

Red Hat build of Quarkus 1.11 Red Hat ビルドの Quarkus 1.11 の リリースノート

法律上の通知

概要

本ガイドには、Red Hat ビルドの Quarkus 1.11 のリリースノートが含まれています。

目次

前書き

本リリースノートには、新機能、テクノロジープレビューの機能、既知の問題、および Red Hat ビルドの Quarkus 1.11 で修正された問題が記載されています。

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 プロパティー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、弊社 の CTO、Chris Wright のメッセージ を参照してください。

第1章 RED HAT ビルドの QUARKUS

Red Hat ビルドの Quarkus は、コンテナーおよび Red Hat OpenShift Container Platform と使用するために最適化された Kubernetes ネイティブ Java スタックです。Quarkus は、Eclipse MicroProfile、Apache Kafka、RESTEasy (JAX-RS)、Hibernate ORM (JPA)、Spring、Infinispan、Apache Camel などの一般的な Java 標準、フレームワーク、およびライブラリーと連携するように設計されています。

Quarkus のディペンデンシーインジェクション (依存性の注入) ソリューションは、CDI (コンテキストとディペンデンシーインジェクション) をベースとし、エクステンションフレームワークを備えているので、機能の拡張、およびフレームワークの設定、起動、アプリケーションへの統合が可能です。

Quarkus は、コンテナーファーストという手法で Java アプリケーションをビルドします。この手法により、Java で書かれたマイクロサービスベースのアプリケーションのビルドが大幅に容易になるほか、これらのアプリケーションがサーバーレスコンピューティングフレームワークで実行している関数を呼び出すことができるようになります。これにより、Quarkus アプリケーションのメモリーフットプリントは小さくなり、起動時間は高速化されます。

第2章 RED HAT OPENSHIFT の QUARKUS メータリングラベル

メータリングラベルを Quarkus Pod に追加し、OpenShift Metering Operator を使用して Red Hat サブスクリプションの詳細を確認できます。

注記

メータリングラベルは、Operator またはテンプレートがデプロイおよび管理する Pod に追加しないでください。

Quarkus は以下のメータリングラベルを使用できます。

- **com.redhat.component-name: "Quarkus"**
- **com.redhat.component-type: application**
- **com.redhat.component-version: 1.11.6**
- **com.redhat.product-name: "Red_Hat_Runtimes"**
- **com.redhat.product-version: 2021-Q1**

その他のリソース

- 『OpenShift Container Platform でのメータリングの設定および使用』

第3章 新機能および変更された機能

本セクションでは、Red Hat ビルドの Quarkus 1.11.6 で導入された新機能および変更について概説します。

3.1. CODE.QUARKUS.REDHAT.COM プロジェクトジェネレーター

Red Hat では、新しい Web ベースのプロジェクトジェネレーターを導入しています。これを使用して、Red Hat ビルドの Quarkus の最新リリースに基づいて、アプリケーション用のプロジェクトを作成できます。プロジェクトジェネレーターは、Quarkus を使用した新しいアプリケーションの開発が容易となる複数の機能を提供します。以下に例を示します。

- プロジェクトに追加するエクステンションを選択するための Web ブラウザーベースのインターフェース。
- 選択したビルドツールに基づくプロジェクトディレクトリー構造の自動生成。
- アプリケーションで使用するために選択したエクステンションの自動インポートおよび設定。
- アプリケーションのスターターコードの自動生成。

プロジェクトジェネレーターは、code.quarkus.redhat.com でアクセスできます。Red Hat は、code.quarkus.redhat.com を使用した Maven ベースのアプリケーションプロジェクトの作成のみをサポートする点に留意してください。

code.quarkus.redhat.com の使用方法についての詳細は、**「code.quarkus.redhat.com を使用した Quarkus Maven プロジェクトの作成」** を参照してください。

3.2. OPENJDK 11 UNIVERSAL BASE IMAGE を SOURCE-TO-IMAGE ビルドの新しいデフォルトベースイメージとして使用

Red Hat ビルドの Quarkus 1.11 は、Source-to-Image (S2I) ツールを使用したアプリケーションの Red Hat OpenShift Container Platform へのビルドおよびデプロイに対して、OpenJDK 11 Universal Base Image の使用をサポートしています。Red Hat Ecosystem Catalog から最新バージョンのイメージをダウンロードできます。

3.3. PROMETHEUS を使用して QUARKUS アプリケーションをモニタリングする新しい MICROMETER メトリクスエクステンション

Red Hat ビルドの Quarkus 1.11 では、Micrometer ライブラリーを使用してランタイムアプリケーションのメトリクスを収集し、アプリケーションを Prometheus でモニタリングするための新しいエクステンションが導入されました。エクステンションを使用すると、アプリケーションから、そしてアプリケーションが使用するエクステンションからも、ランタイムアプリケーションメトリクスを収集し、Micrometer と統合することができます (Apache Kafka の Quarkus エクステンション、HTTP、Resteasy など)。Red Hat は、Quarkus Micrometer メトリクスの実稼働環境での使用をサポートします。**『Quarkus アプリケーションでのメトリクスの収集』** を参照してください。

3.4. 複数の HIBERNATE ORM 永続ユニットへのサポート

Red Hat ビルドの Quarkus 1.11 では、Hibernate ORM を使用してアプリケーションでデータソースを管理する際に、複数のデータソースを永続ユニットとして定義することができます。詳細は、**『Quarkus アプリケーションでのデータソースの設定』** の **「複数の JDBC データソースの設定」** を参照してください。

3.5. 生成された OPENAPI スキーマの保存へのサポート

Red Hat ビルドの Quarkus 1.11 では、Quarkus Smallrye OpenAPI エクステンションを使用してアプリケーションに生成された OpenAPI スキーマを保存するためのサポートが導入されました。**quarkus-smallrye-openapi_quarkus.smallrye-openapi.store-schema-directory** の値は、アプリケーションのコンパイル時に OpenAPI スキーマを含む YAML ファイルと JSON ファイルが保存されるディレクトリーのパスに設定できます。例:

application.properties

```
quarkus-smallrye-openapi_quarkus.smallrye-openapi.store-schema-directory=/path/to/schema/directory
```

3.6. QUARKUS QUARTZ エクステンションでの ARC コンテキストおよび依存性注入のサポート

Quarkus Quartz エクステンションを使用して、コンテキストおよび依存性注入に依存する定期的なタスクをスケジュールできるようになりました。詳細は、Quarkus Quartz エクステンションに関する コミュニティードキュメント を参照してください。

3.7. SMALLRYE REACTIVE MESSAGING のバージョン 2.7.1 へのアップグレード

Red Hat ビルドの Quarkus 1.11 では、AMQP および Apache Kafka の Quarkus Reactive Messaging エクステンションで使用される SmallRye Reactive Messaging API がバージョン 2.7.1 にアップグレードされました。詳細は、SmallRye Reactive Messaging API ドキュメント を参照してください。

3.8. MUTINY REACTIVE API のバージョン 0.12.5 へのアップグレード

Red Hat ビルドの Quarkus 1.11 では、Quarkus のリアクティブエクステンションで使用される Mutiny イベント駆動型ライブラリーがバージョン 0.12.5 にアップグレードされました。

3.9. リアクティブルートでの BEAN VALIDATION のサポート

Red Hat ビルドの Quarkus 1.11 におけるリアクティブルートのエクステンションは、Java Bean の制約検証をサポートします。

3.10. QUARKUS REST アプリケーションのデフォルトの JSON シリアライゼーションおよびデシリアライゼーションツールとしての JACKSON への変更

警告

この変更により、アプリケーションを Red Hat ビルドの Quarkus 1.7 から Red Hat ビルドの Quarkus 1.11 にアップグレードする際に、アプリケーションの REST エンドポイントが破損する可能性があります。アプリケーションのアップグレード後にオブジェクトマッピングが正しく機能するように、アプリケーションコードを更新します。

Red Hat ビルドの Quarkus 1.11 リリースでは、Jackson は Quarkus REST JSON エクステンションによって使用されるデフォルトの **ObjectMapper** ツールとして設定されます。コンテキストと依存性注入を使用して、Jackson をアプリケーションの REST コントローラークラスに注入し、REST アプリケーションデータの JSON 形式への変換および JSON 形式からの変換をサポートすることができます。Red Hat ビルドの Quarkus 1.11 の Jackson で **DeserializationFeature.FAIL_ON_UNKNOWN_PROPERTIES** 機能を無効化することで発生する互換性を破る変更に関する詳細は、**「Red Hat ビルドの Quarkus 1.7 から Red Hat ビルドの Quarkus 1.11 へのアプリケーションのアップグレード」** を参照してください。

3.11. アプリケーション以外のユーザーインターフェースの有効化の新しいオプション (実稼働モードでアプリケーションを起動する場合)

Red Hat ビルドの Quarkus 1.11 では、**-Dquarkus.<ui-name>.always-include=true** を指定して、アプリケーションを実稼働モードで起動する場合に JAR の一部であるユーザーインターフェースを有効化することができます。このオプションは、以下のインターフェースで利用できます。

- Swagger UI
- OpenAPI
- SmallRye Health UI
- GraphQL UI

たとえば、SwaggerUI インターフェースが含まれる REST アプリケーションに JAR を作成すると、アプリケーションの起動時にこのインターフェースはデフォルトで無効になります。アプリケーションの起動時にインターフェースを有効にするために、起動コマンドに **-Dquarkus.swagger-ui.always-include=true** オプションを追加することができます。

```
java -jar -Dquarkus.swagger-ui.enable=true target/<application-name>-1.0.0-SNAPSHOT-runner.jar
```

<application-name> は JAR の名前に置き換える必要があることに注意してください。

3.12. QUARKUS REST クライアントのセキュリティー更新による CVE-2020-25633 の解決

Red Hat ビルドの Quarkus 1.11 で利用可能な **quarkus-rest-client** エクステンションは、CVE-2020-25633 セキュリティー問題を解決する更新の一部として導入された、MicroProfile REST クライアントおよび JAX-RS クライアントによる **WebApplicationException** の処理の変更の影響を受けます。

セキュリティー更新により、RESTEasy バージョン 4.5.9 では、クライアントアプリケーションが

WebApplicationException を返す際の **Response** の処理方法が変更されます。4.5.9 の更新前は、ローカルドメインのリモートサーバーから送信された **Response** には、承認されていないユーザーがアクセスする可能性があるリモートサーバー (Cookie など) に関する機密情報が含まれていました。

RESTEasy 4.5.9 の更新により、**Response** の内容の処理方法が変更されます。ローカルサーバーが **Response** を受け取ると、RESTEasy は Response を保存する前にすべての機密コンテンツを削除しますが、ローカルサーバーが必要に応じて **Response** の元のコンテンツにアクセスできる方法を保持します。

この例外処理の変更により、ローカルサーバーに機密性の高いコンテンツを格納することに関連するセキュリティーリスクを回避できるようになりましたが、RESTEasy バージョン 4.5.9 を使用するクライアントは引き続き JAX-RS 仕様と互換性を保つことができます。

RESTEasy バージョン 4.5.9 への **Response** コンテンツの保存の変更に関する詳細は、RESTEasy 4.5.9 ドキュメントの **WebApplicationException** のセクションを参照してください。

3.13. ネイティブ実行可能ファイルをコンパイルするためのデフォルトの MANDREL ベースイメージのバージョン 20.3 へのアップグレード

Red Hat ビルドの Quarkus 1.11 では、ネイティブ実行可能ファイルをコンパイルするためのデフォルトのベースイメージが Mandrel 20.3 にアップグレードされます。その結果、Quarkus アプリケーションのネイティブ実行可能ファイルへのコンパイル の **quarkus.native.builder-image** 設定プロパティー のデフォルト値が、**quay.io/quarkus/ubi-quarkus-mandrel:20.3-java11** に変更されます。

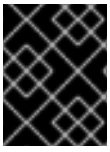
3.14. バージョン 5.X にアップグレードされた QUARKUS KUBERNETES クライアント

重要

Red Hat は、実稼働環境で Quarkus Kubernetes クライアントを使用する場合のサポートは提供していません。

Red Hat ビルドの Quarkus 1.11 には、Quarkus Kubernetes クライアントの新しいバージョンが含まれています。Quarkus Kubernetes クライアントバージョン 4.x を開発環境で使用し、既存のアプリケーションをバージョン 5.x にアップグレードする場合は、Kubernetes Client Migration Guide を参照してください。

3.15. QUARKUS DEV UI



重要

Red Hat では、実稼働環境での Quarkus Dev UI の使用に対するサポートは提供していません。

Red Hat ビルドの Quarkus 1.11 には、Quarkus Dev UI の最初のリリースが含まれます。Dev UI は実験的なインターフェースで、以下のような場合に使用できます。

- アプリケーションで現在使用しているエクステンションの一覧の表示
- アプリケーションでのエクステンションのステータスの確認
- アプリケーションで使用しているエクステンションのドキュメントへのアクセス

アプリケーションを開発モードで起動して、Web ブラウザーで **localhost:8080/q/dev** に移動すると、Dev UI にアクセスできます。

第4章 RED HAT ビルドの QUARKUS 1.7 から RED HAT ビルドの QUARKUS 1.11 へのアプリケーションのアップグレード

本セクションでは、Red Hat ビルドの Quarkus 1.7 から Red Hat ビルドの Quarkus 1.11 にアプリケーションをアップグレードする際に対応が必要な互換性を破る変更の概要を説明します。

4.1. QUARKUS QUARTZ エクステンションの設定プロパティーの変更

Red Hat ビルドの Quarkus 1.11 リリースでは、Quarkus Quartz エクステンション で利用可能な設定プロパティーにいくつかの変更が加えられました。

Red Hat ビルドの Quarkus 1.11 で導入された新しい設定プロパティー

- **quarkus.quartz.cluster-checkin-interval**
- **quarkus.quartz.instance-name**
- **quarkus.quartz.start-mode**

Red Hat ビルドの Quarkus 1.11 で削除された設定プロパティー

- **quarkus.quartz.force-start**

表4.1 Red Hat ビルドの Quarkus 1.11 で名前が変更された設定プロパティー

Quarkus 1.7 のプロパティー名	Quarkus 1.11 のプロパティー名
quarkus.quartz.triggerListener."namedTriggerListener".class	quarkus.quartz.trigger-listeners."listener-name".class
quarkus.quartz.triggerListener."namedTriggerListener"	quarkus.quartz.trigger-listeners."listener-name".properties
quarkus.quartz.jobListener."namedJobListener".class	quarkus.quartz.job-listeners."listener-name".class
quarkus.quartz.jobListener."namedJobListener"	quarkus.quartz.job-listeners."listener-name".properties
quarkus.quartz.plugin."namedPlugin".class	quarkus.quartz.plugins."plugin-name".class
quarkus.quartz.plugin."namedPlugin"	quarkus.quartz.plugins."plugin-name".properties

4.2. SPRING BOOT 設定プロパティーの命名ストラテジーの変更

Red Hat ビルドの Quarkus 1.11 では、Quarkus アプリケーションの大文字および小文字の組み合わせが含まれる Spring Boot 設定プロパティーに使用される命名ストラテジーは、**verbatim** に設定されなくなりました。

代わりに、**quarkus.arc.config-properties-default-naming-strategy** プロパティーを、プロジェクトの **application.properties** ファイルの以下のいずれかの値に設定できます。

from-config

命名規則はアプリケーション設定で指定されます。

verbatim

設定プロパティーの名前は、プロパティーが適用されるフィールドまたはメソッドの名前と一致します。

kebab

設定プロパティーの名前は小文字を使用し、スペースはハイフンに置き換えます。例: **application-name**

アプリケーションの **quarkus.arc.config-properties-default-naming-strategy** プロパティーを設定しない場合は、 **kebab** がデフォルト値として使用されます。

アプリケーションの **verbatim** 命名ストラテジーに従ってフォーマットされた Spring Boot 設定プロパティーを使用している場合は、以下のいずれかの変更を行います。

- プロジェクトの **application.properties** または **application.yml** ファイルで、**quarkus.arc.config-properties-default-naming-strategy** の値を **verbatim** に設定します。例:

  application.properties

  ```
  quarkus.arc.config-properties-default-naming-strategy=verbatim
  ```

- アプリケーションで使用する設定プロパティーの名前を変換し、**kebab** 命名ストラテジーと一致するようにします。

4.3. QUARKUS.DATASOURCE.URL および QUARKUS.DATASOURCE.DRIVER データソース設定プロパティーのサポートの削除

Red Hat ビルドの Quarkus 1.11 では、Red Hat ビルドの Quarkus 1.3 で導入された接続 URL およびデータソースのドライバーを設定するプロパティーをサポートしないことになりました。

サポート対象外の設定プロパティーを Quarkus 1.11 と互換性のあるプロパティーに置き換えて、アプリケーションを Quarkus 1.11 にアップグレードする際にデータソース設定が正しく機能するようにします。

Quarkus 1.11 のアプリケーションでデータソースを設定する方法の詳細は、**『Quarkus アプリケーションでのデータソースの設定』** を参照してください。

4.4. QUARKUS アプリケーションのデフォルトのメディアタイプの JSON への変更

警告

この変更により、アプリケーションを Red Hat ビルドの Quarkus 1.7 から Red Hat ビルドの Quarkus 1.11 にアップグレードする際に、アプリケーションの REST エンドポイントが破損する可能性があります。アプリケーションの REST エンドポイントが使用する戻り値の型を更新し、アプリケーションのアップグレード後も引き続き REST エンドポイントが正しく動作するようにします。

Red Hat ビルドの Quarkus 1.11 リリースでは、アプリケーションデータをシリアライズするデフォルト形式は JSON に変更されます。

アプリケーションの REST エンドポイントのコンテンツタイプ形式として JSON をデフォルトで使用することを無効にし、JSON を使用するインターフェースのみに対してアノテーションを使用して JSON の仕様を明示的に有効化することができます。

1. アプリケーションの **application.properties** ファイルで、**quarkus.resteasy-json.default-json** プロパティーの値を **false** に設定します。

   application.properties

   ```
   quarkus.resteasy-json.default-json=false
   ```

2. JSON をコンテンツタイプ形式として使用するアプリケーションの REST エンドポイントに **@Produces(MediaType.APPLICATION_JSON)** および **@Consumes(MediaType.APPLICATION_JSON)** アノテーションを追加します。

4.5. JACKSON の FAIL_ON_UNKNOWN_PROPERTIES 機能はデフォルトで無効化

Red Hat ビルドの Quarkus 1.11 では、Jackson はデフォルトで **DeserializationFeature.FAIL_ON_UNKNOWN_PROPERTIES** を無効にし、認識されないプロパティーを無視するように設定されています。これは、アプリケーションによって認識されないプロパティーが含まれる JSON データオブジェクトをデシリアライズしようとする際に失敗しないようにするためです。

アプリケーションの **application.properties** ファイルで **quarkus.jackson.fail-on-unknown-properties** の値を **true** に設定して、**FAIL_ON_UNKNOWN_PROPERTIES** 機能を有効にします。この機能は、アプリケーションのクラスごとに個別に有効にする必要があります。

application.properties

```
<fully-qualified-class-name>.quarkus.jackson.fail-on-unknown-properties=true
```

REST アプリケーションでの Jackson の使用およびカスタマイズに関する詳細は、Quarkus REST JSON extension guide を参照してください。

4.6. DEBUG および TRACE レベルでの最小ロギングレベルの設定への変更

Quarkus 1.11 では、**DEBUG** および **TRACE** レベルでメッセージをログに記録するようにロガーを設定する際に、最小ロギングレベルを設定する必要があります。アプリケーションを起動する前に、最小ロギ

ングレベルを設定する必要があります。ロガー設定に最小ロギングレベルを設定しない場合、**DEBUG** および **TRACE** ログレベルのメッセージはログ出力に表示されません。たとえば、**TRACE** レベルでログを記録するようにロガーを設定する場合は、最小ログレベルを **TRACE** にする必要があります。そうしないと、**TRACE** レベルのメッセージはログ出力に表示されません。

アプリケーションの **application.properties** ファイルで、特定カテゴリーの最小ログレベルを設定できます。**application.properties**。たとえば、ロガー設定の **io.quarkus.smallrye.jwt** および **io.undertow.request.security** カテゴリーの **TRACE** レベルでロギングを有効にするには、以下のプロパティーを設定します。

application.properties

```
quarkus.log.file.enable=true
# Send output to a trace.log file under the /tmp directory
quarkus.log.file.path=/tmp/trace.log
quarkus.log.file.level=TRACE
quarkus.log.file.format=%d{HH:mm:ss} %-5p [%c{2.}] (%t) %s%e%n
# Set 2 categories (io.quarkus.smallrye.jwt, io.undertow.request.security) to TRACE level
quarkus.log.min-level=TRACE
quarkus.log.category."io.quarkus.smallrye.jwt".level=TRACE
quarkus.log.category."io.undertow.request.security".level=TRACE
```

4.7. RED HAT ビルドの QUARKUS BOM の内部構造への変更

Red Hat ビルドの Quarkus 1.11 では、**com.redhat.quarkus:quarkus-universe-bom** にすべてのエクステンションおよび依存関係の **goupId**、**artifactID**、および **version** を直接含まなくなりました。代わりに、**com.redhat.quarkus:quarkus-universe-bom** は、Red Hat がサポートするすべての Red Hat ビルドの Quarkus エクステンションおよびすべてのコミュニティー Quarkus エクステンションを含む **io.quarkus:quarkus-universe-bom** の依存関係宣言が含まれる **com.redhat.quarkus quarkus-product-bom** をインポートします。

この変更は、Maven プロジェクトで Red Hat ビルドの Quarkus BOM を使用する方法には影響しません。ただし、カスタムスクリプトを使用して BOM を解析する場合は、アプリケーションを Red Hat ビルドの Quarkus 1.11 にアップグレードした後も引き続き解析が正しく機能するように、カスタムスクリプトを更新する必要があります。

4.8. REST エンドポイントパス解決における変更

警告

この変更により、アプリケーションを Red Hat ビルドの Quarkus 1.7 から Red Hat ビルドの Quarkus 1.11 にアップグレードする際に、アプリケーションの REST エンドポイントが破損する可能性があります。アプリケーションの移行後にエンドポイントパスを更新するようにしてください。

Red Hat ビルドの Quarkus 1.11 では、アプリケーションとアプリケーション以外の REST エンドポイントのパスは、一般的な絶対ルートパスと相対的に解決されます。REST エンドポイントのデフォルトの一般的なルートパスは、以下のように設定されます。

- /: アプリケーションのメイン REST コントローラークラスによって直接公開される REST エンドポイント。アプリケーションエンドポイントのデフォルトパスを変更するには、プロジェクトの **application.properties** ファイルの **quarkus.http.root-path** プロパティーの値を変更します。

- **q**: アプリケーションと統合されるツールが提供するサービスの REST エンドポイント (アプリケーションのヘルスモニタリングやメトリクスコレクションなどを目的とする)。アプリケーションエンドポイントのデフォルトパスを変更するには、プロジェクトの **application.properties** ファイルの **quarkus.http.non-application-root-path** プロパティーの値を変更します。

相対ルートパスは、**quarkus.http.root-path** プロパティーによって定義されるルートパスの下にネストされる点に留意してください。たとえば、**quarkus.http.root-path** プロパティーに定義されたルートパスが / に設定され、**quarkus.http.non-applicationroot-path** プロパティーに定義されたアプリケーション以外のエンドポイントのルートパスが **q** に設定されている場合、アプリケーション以外のエンドポイントの絶対エンドポイントパスは **/q/<non-application-endpoint-name>** になります。

ただし、アプリケーション以外の個々のエンドポイントのパスを、**/q/<non-application-endpoint-name>** に配置するように明示的に設定することもできます。

エンドポイントパスは **quarkus.http.root-path** および **quarkus.http.non-application-root-path** によって設定されるルートパスに相対的と解釈されるため、アプリケーションのエンドポイントに設定するカスタムパスおよびサブパスから先頭のスラッシュ (/) を除外する必要があります。

たとえば、アプリケーションの REST コントローラーで Prometheus のメトリクスエンドポイントを公開する場合、エンドポイントが **/q/metrics** で公開されるように、**@Path** アノテーションのエンドポイントパスを **metrics** に設定する必要があります。同じパス値を **/metrics** に設定すると、メトリクスエンドポイントは **/metrics** で公開されます。

アプリケーション以外のエンドポイントを別の namespace の下に設定する例
たとえば、プロジェクトの **application.properties** ファイルに以下のプロパティーを設定し、**/api** ルートパスで **hello** エンドポイントアプリケーションを、そして **/api/q** パスで **metrics** エンドポイントを公開することができます。

application.properties

```
quarkus.http.root-path=/api
quarkus.http.non-application-root-path=q
```

この設定では、**/api/hello** と **/api/q/metrics** の両方ともパブリックです。つまり、**/api/hello** にアクセスするパーミッションを持つユーザーは、**/api/q/metrics** エンドポイントにもリクエストを送信して、有効な Response を受け取ることができます。

health エンドポイントをパブリック以外にする場合は、**application.properties** のアプリケーション以外のエンドポイントのルートパスを **/q** namespace に設定できます。

application.properties

```
quarkus.http.root-path=/api
quarkus.http.non-application-root-path=/q
```

この設定では、**/api/hello** エンドポイントはパブリックですが、**/q/metrics** は異なるアクセスパーミッションを設定できる別の namespace に公開されています。

Red Hat ビルドの Quarkus 1.11 では、元のアプリケーション以外のエンドポイントパスに送信されたリクエストは、**/q** namespace の新しいパスに自動的にリダイレクトされます。

プロジェクトの **application.properties** ファイルで以下の属性を設定して、アプリケーション以外のエンドポイントパスの自動リダイレクトを無効にすることができます。

```
quarkus.http.redirect-to-non-application-root-path=false
```

quarkus.http.non-application-root-path の値を、絶対アプリケーションエンドポイントルートの値に解決する変数に設定し、すべてのエンドポイントの絶対エンドポイントのルートパスを使用するようにアプリケーションを切り替えることができます。

```
quarkus.http.non-application-root-path=${quarkus.http.root-path}
```

4.9. REST アプリケーションを RED HAT OPENSHIFT CONTAINER PLATFORM にデプロイする CONFIGMAP オブジェクトの処理時に、追加の設定プロパティーが必要

警告

この変更により、アプリケーションを Red Hat ビルドの Quarkus 1.7 から Red Hat ビルドの Quarkus 1.11 にアップグレードする際に、アプリケーションの OpenShift へのデプロイに使用する設定に不具合が生じる可能性があります。ConfigMap で提供される設定パラメーターがアプリケーションによって確実に認識されるように、アプリケーションの **applications.properties** ファイルを更新する必要があります。

Red Hat ビルドの Quarkus 1.11 では、Resteasy をベースとする Quarkus アプリケーションを Red Hat OpenShift Container Platform に設定し、**quarkus.openshift** で指定された ConfigMap にアプリケーションの設定パラメーターを提供する場合は、アプリケーションが ConfigMap を認識および処理するように、**application.properties** ファイルの **quarkus.openshift.app-secret** および **quarkus-openshift.app-configmap** プロパティーも指定する必要があります。

- アプリケーションの **application.properties** ファイルに以下のプロパティーを追加して、アプリケーションが ConfigMap を認識できるようにします。

  application.properties

  ```
  quarkus.openshift.app-secret=<secret-name>
  quarkus-openshift.app-configmap=<configmap-name>
  ```

  <secret-name> を使用するシークレットの名前に置き換え、**<configmap-name>** を使用する ConfigMap の名前に置き換える必要があります。

第5章 RED HAT ビルドの QUARKUS 対応プラットフォーム、設定、エクステンション、および依存関係

- サポートされる設定とテスト済みの統合の一覧は、「Red Hat ビルドの Quarkus でサポートされる構成」 のページを参照してください (ログインが必要です)。
- サポートされる Maven アーティファクトの一覧は、「Red Hat build of Quarkus Component Details」 のページを参照してください (ログインが必要です)。

Red Hat ビルドの Quarkus 1.11 でテクノロジープレビューとして利用可能なエクステンションおよび依存関係の一覧は、「Component details overview」 を参照してください。

5.1. サポートされるエクステンション、依存関係、およびプラグイン

Red Hat ビルドの Quarkus 1.11 では、以下のサポートされるエクステンションが追加されています。

- Quarkus Micrometer
- Quarkus OpenID Connect クライアント
- Quarkus OpenID Connect クライアントフィルター
- Quarkus Resteasy Multipart

Red Hat ビルドの Quarkus にて、Red Hat が 本番環境でサポートするエクステンション、依存関係、およびプラグインの一覧は、「Red Hat build of Quarkus Component Details」 のページを参照してください (ログインが必要です)。

5.2. 開発サポート

Red Hat は、以下に示す Red Hat ビルドの Quarkus の機能、プラグイン、エクステンション、および依存関係に対して 開発サポート を提供します。

機能

- ライブ開発モード
- リモート開発モード
- Dev UI

プラグイン

- **protobuf-maven-plugin**

第6章 非推奨のコンポーネントおよび機能

本セクションに記載するコンポーネントおよび機能は、Red Hat ビルドの Quarkus 1.11 で非推奨となりました。これらは、本リリースに含まれており、サポートもされますが、機能拡張の予定はなく、今後削除される可能性があります。

Red Hat ビルドの Quarkus 1.11 で非推奨となったコンポーネントおよび機能の一覧は、Red Hat build of Quarkus Component Details のページを参照してください。

第7章 テクノロジープレビュー

本セクションでは、Red Hat ビルドの Quarkus 1.11 でテクノロジープレビューとして利用可能な機能およびエクステンションについて記載しています。

重要

これらの機能は、テクノロジープレビュー機能としてのみ利用可能です。テクノロジープレビュー機能は、Red Hat の実稼働環境のサービスレベルアグリーメント (SLA) ではサポートされず、機能的に完全ではないことがあるため、Red Hat では実稼働環境での使用を推奨していません。テクノロジープレビュー機能は、最新の製品機能をいち早く提供し、お客様には開発段階で機能性をテストし、フィードバックをお寄せいただくことができます。

Red Hat のテクノロジープレビュー機能についての詳細は 「テクノロジープレビュー機能のサポート範囲」 を参照してください。

7.1. テクノロジープレビュー機能

7.1.1. fast-jar としての Quarkus アプリケーションのパッケージ化

fast-jar パッケージ形式は、アプリケーションの起動時間を高速化するデフォルトの JAR パッケージ形式に代わるものです。プロジェクトの **application.properties** ファイルに以下のプロパティーを設定して、fast-jar パッケージを有効にすることができます。

application.properties

```
quarkus.package.type=fast-jar
```

また、アプリケーションのパッケージ化に使用するコマンドに **-Dquarkus.package.type=fast-jar** プロパティーを追加できます。

```
mvn clean package -Dquarkus.package.type=fast-jar
```

7.2. テクノロジープレビューの拡張機能および依存関係

Red Hat ビルドの Quarkus 1.11 でテクノロジープレビューとして利用可能なエクステンションおよび依存関係の一覧は、「Red Hat build of Quarkus Component Details」 のページを参照してください (ログインが必要です)。

第8章 既知の問題

本セクションでは、Red Hat ビルドの Quarkus 1.11 の既知の問題について記載しています。

- Issue #11633: OpenShift Serverless のゼロ設定ソリューションがありません。この問題は、Quarkus ネイティブ Serverless アプリケーションのデプロイメントのみに影響があります。
- QUARKUS-695: Quarkus Keycloak Authorization は、Red Hat Single Sign-On 7.4 の最新バージョンと互換性がありません。
- QUARKUS-697: ネイティブノードで MicroProfile HTTP クライアントを使用する場合に、デフォルトメソッドのフォールバックが動作しません。
- QUARKUS-719: Quarkus Reactive PG クライアントは、接続が利用可能な場合でも pgPool からのデータベース接続を使用しようとする場合に **Fail to read any response from the server, the underlying connection might get lost unexpectedly** のエラーメッセージを表示します。

改訂日時: 2021-05-06 00:10:46 UTC