Red Hat build of Quarkus 3.8

红帽构建的 Quarkus 入门

Last Updated: 2024-05-18

法律通告

摘要

本指南论述了如何使用 Apache Maven 创建简单 Quarkus 应用程序。

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 红帽构建的 QUARKUS 入门

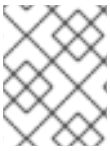
作为应用程序开发人员，您可以使用红帽构建的 Quarkus 创建使用 Java 在 OpenShift 环境中运行的基于微服务的应用程序。Quarkus 应用程序可以在 Java 虚拟机(JVM)上运行，或者编译到原生可执行文件。原生应用的内存占用量较小，启动时间比其 JVM 对应的启动速度要快。

您可以使用以下方法之一创建 Quarkus 应用程序：

- 使用 Apache Maven 和 Quarkus Maven 插件
- 使用 code.quarkus.redhat.com
- 使用 Quarkus 命令行界面(CLI)

您可以开始使用 Quarkus，并创建、测试、软件包并运行一个简单的 Quarkus 项目来公开 **hello** HTTP 端点。为演示依赖项注入，**hello** HTTP 端点使用 **问候** Bean。

78_OpenShift_0420



注意

对于入门练习的已完成示例，请下载 Quarkus Quickstarts 归档或克隆 Quarkus Quickstarts Git 存储库，再前往 **getting-started** 目录。

1.1. 关于红帽构建的 QUARKUS

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

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

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

有关 Quarkus 社区版本和红帽构建的 Quarkus 之间的差异的详情，请参考 Quarkus 社区版本和红帽构建的 Quarkus 之间的 Differences。

1.2. 准备您的环境

在使用 Quarkus 之前，您必须准备您的环境。

流程

- 确认您的系统中已完成以下安装：
  - 已安装 OpenJDK 17 或 21，并设置 **JAVA_HOME** 环境变量来指定 Java SDK 的位置。

- 要下载红帽构建的 OpenJDK，请登录到红帽客户门户网站，再进入 Software Downloads。

- 已安装 Apache Maven 3.8.6 或更高版本。Apache Maven 位于 Apache Maven Project 网站。

- **可选** ：如果要使用 Quarkus 命令行界面(CLI)，请确保已安装它。

  - 有关如何安装 Quarkus CLI 的说明，请参考 Quarkus CLI 中的特定于社区的信息。

重要

Quarkus CLI 仅用于 dev 模式。红帽不支持在生产环境中使用 Quarkus CLI。

1.2.1. 关于红帽构建的 Quarkus BOM

从红帽构建的 Quarkus 2.2 中，所有核心 Quarkus 扩展的依赖项版本都由使用 **com.redhat.quarkus.platform:quarkus-bom** 文件进行管理。

Bill of Materials (BOM)文件的目的是管理项目中的 Quarkus 工件的依赖项版本，以便在项目中使用 BOM 时，您不需要指定哪些依赖项版本协同工作。相反，您可以将 Quarkus BOM 文件导入到 **pom.xml** 配置文件，其中依赖项版本包含在 < **dependencyManagement>** 部分中。因此，您不需要列出由 **pom.xml** 文件中指定 BOM 管理的单个 Quarkus 依赖项版本。

要查看红帽构建的 Quarkus 中支持的扩展特定 BOM 的信息，请参阅 红帽构建的 Quarkus 组件详情。

您只需要为应用程序中使用的平台成员扩展导入特定于成员的 BOM。因此，与单个 BOM 相比，您需要管理较少的依赖项。因为每个特定于成员的 BOM 是通用 Quarkus BOM 的片段，所以您可以以任何顺序导入成员 BOM，而不创建冲突。

1.2.2. 关于 Apache Maven 和红帽构建的 Quarkus

Apache Maven 是一个分布式构建自动化工具，用于 Java 应用程序开发，用于创建、管理和构建软件项目。

要了解有关 Apache Maven 的更多信息，请参阅 Apache Maven 文档。

Maven 存储库

Maven 存储库存储 Java 库、插件和其他构建构件。默认公共存储库是 Maven 2 Central Repository，但存储库可以是私有和内部的，以在开发团队之间共享通用工件。也可从第三方提供存储库。

您可以将 Red Hat-hosted Maven 存储库与 Quarkus 项目搭配使用，也可以下载红帽构建的 Quarkus Maven 存储库。

Maven 插件

Maven 插件是 POM 文件的定义部分，用于运行一个或多个任务。红帽构建的 Quarkus 应用程序使用以下 Maven 插件：

- *Quarkus Maven 插件(**quarkus-maven-plugin**)* ：启用 Maven 来创建 Quarkus 项目，将应用程序打包到 JAR 文件中，并提供 dev 模式。

- *Maven Surefire 插件(**maven-surefire-plugin**)*: 当 Quarkus 启用 **测试** 配置集时，会在构建生命周期的测试阶段使用 Maven Surefire 插件在应用程序上运行单元测试。**该插件生成包含测试报告的文本和 XML 文件。**

其他资源

- 配置红帽构建的 Quarkus 应用程序

1.2.3. 为在线存储库配置 Maven settings.xml 文件

要将 Red Hat-hosted Quarkus 存储库与您的 Quarkus Maven 项目搭配使用，请为您的用户配置 settings.xml 文件。与存储库管理器或共享服务器上的存储库一起使用的 Maven 设置可以提供更好的项目控制和易管理性。

注意

当您修改 Maven settings.xml 文件来配置存储库时，这些更改将应用到所有 Maven 项目。如果只想将配置应用到特定的项目，请使用 **-s** 选项并指定特定于项目的 settings.xml 文件的路径。

流程

1. 在文本编辑器中或集成开发环境(IDE)中打开 Maven $HOME/.m2/settings.xml 文件。

注意

如果 $HOME/.m2/ 目录中没有 settings.xml 文件，请将 settings.xml 文件从 $MAVEN_HOME/conf/ 目录中复制到 $HOME/.m2/ 目录中。

2. 在 settings.xml 文件的 <profiles > 元素中添加以下行：

```
<!-- Configure the Red Hat build of Quarkus Maven repository -->
<profile>
 <id>red-hat-enterprise-maven-repository</id>
 <repositories>
  <repository>
   <id>red-hat-enterprise-maven-repository</id>
   <url>https://maven.repository.redhat.com/ga/</url>
   <releases>
    <enabled>true</enabled>
   </releases>
   <snapshots>
    <enabled>false</enabled>
   </snapshots>
  </repository>
 </repositories>
```

```
    <pluginRepositories>
      <pluginRepository>
        <id>red-hat-enterprise-maven-repository</id>
        <url>https://maven.repository.redhat.com/ga/</url>
        <releases>
          <enabled>true</enabled>
        </releases>
        <snapshots>
          <enabled>false</enabled>
        </snapshots>
      </pluginRepository>
    </pluginRepositories>
  </profile>
```

3. 在 settings.xml 文件的 < activeProfiles > 元素中添加以下行并保存文件。

```
<activeProfile>red-hat-enterprise-maven-repository</activeProfile>
```

1.2.4. 将 Maven 项目重新配置为红帽构建的 Quarkus

您可以通过更改项目 POM 文件中的 Maven 配置，将 Quarkus 社区项目迁移到红帽构建的 Quarkus。

先决条件

- 您有一个使用 Maven 构建的 Quarkus 项目，它依赖于 pom.xml 文件中的 Quarkus 社区工件。

流程

- 更改项目的 pom.xml 文件的 <properties > 部分中的以下值：
  - 将 < quarkus.platform.group-id> 属性的值更改为 com.redhat.quarkus.platform。
  - 将 < quarkus.platform.version> 属性的值改为 3.8.4.redhat-00002。

pom.xml

```
<project>
  ...
```

```
  <properties>
    ...
    <quarkus.platform.group-id>com.redhat.quarkus.platform</quarkus.platform.group-
id>
    <quarkus.platform.artifact-id>quarkus-bom</quarkus.platform.artifact-id>
    <quarkus.platform.version>3.8.4.redhat-00002</quarkus.platform.version>
    ...
  </properties>
  ...
</project>
```

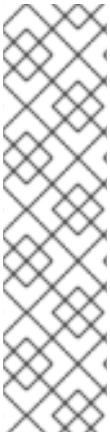
1.3. 配置红帽构建的 QUARKUS 开发人员工具

通过使用 Quarkus 开发人员工具，您可以完成以下任务，例如：

- 为您的应用程序创建一个 Maven 项目
- 添加并配置要在应用程序中使用的扩展
- 在 OpenShift 集群中部署应用程序

1.3.1. 配置红帽构建的 Quarkus 扩展 registry 客户端

扩展 registry registry.quarkus.redhat.com 托管红帽提供的 Quarkus 扩展。您可以通过在 registry 客户端配置文件中添加 registry 来配置 Quarkus 开发人员工具来访问此 registry 中的扩展。registry 客户端配置文件是一个 YAML 文件，其中包含一个 registry 列表。



注意

- 默认 Quarkus registry 是 registry.quarkus.io; 通常，您不需要配置它。但是，如果用户提供自定义 registry 列表，并且 registry.quarkus.io 没有在其中，则不会启用 registry.quarkus.io。
- 确保您喜欢的 registry 首先出现在 registry 列表中。当 Quarkus 开发人员工具搜索 registry 时，它们从列表的顶部开始。

流程

1. 打开包含扩展 registry 配置的 config.yaml 文件。当您首次配置扩展 registry 时，您可能需要在机器的 $HOME/.quarkus 目录中创建 config.yaml 文件。

2. 将新 registry 添加到 config.yaml 文件中。例如：

config.yaml

```
registries:
 - registry.quarkus.redhat.com
 - registry.quarkus.io
```

1.4. 创建 GETTING STARTED 项目

通过创建一个 get -started 项目，您可以使用简单的 Quarkus 应用程序启动并运行。您可以使用以下方法之一 创建一个 get-started 项目：

- 使用 Apache Maven 和 Quarkus Maven 插件
- 使用 code.quarkus.redhat.com 生成 Quarkus Maven 项目
- 使用 Quarkus 命令行界面(CLI)

先决条件

- 您已准备了您的环境。如需更多信息，请参阅准备您的环境。

流程

- 根据您的要求，选择您要用来创建 get -started 项目的方法。

1.4.1. 使用 Apache Maven 创建 Getting Started 项目

您可以使用 Apache Maven 和 Quarkus Maven 插件创建一个 get -started 项目。通过此 入门 项目，您可以使用简单的 Quarkus 应用程序启动并运行。

先决条件

- 您已准备了环境以使用 Maven。如需更多信息，请参阅准备您的环境。
- 您已配置了 Quarkus Maven 存储库。要使用 Maven 创建 Quarkus 应用程序，请使用 Red Hat-hosted Quarkus 存储库。如需更多信息，请参阅为在线存储库配置 Maven settings.xml 文件。

流程

1. 要验证 Maven 是否使用 OpenJDK 17 或 21，Maven 版本是否为 3.8.6 或更高版本，且 mvn 可从 PATH 环境变量访问，请输入以下命令：

```
mvn --version
```

2. 如果前面的命令没有返回 OpenJDK 17 或 21，请将到 OpenJDK 17 或 21 的路径添加到 PATH 环境变量中，然后再次输入前面的命令。

3. 要生成项目，请输入以下命令之一：

   - 如果使用 Linux 或 Apple macOS，请输入以下命令：

```
mvn com.redhat.quarkus.platform:quarkus-maven-plugin:3.8.4.redhat-00002:create \
    -DprojectGroupId=org.acme \
    -DprojectArtifactId=getting-started \
    -DplatformGroupId=com.redhat.quarkus.platform \
    -DplatformVersion=3.8.4.redhat-00002 \
    -DclassName="org.acme.quickstart.GreetingResource" \
    -Dpath="/hello"
cd getting-started
```

   - 如果使用 Microsoft Windows 命令行，请输入以下命令：

```
mvn com.redhat.quarkus.platform:quarkus-maven-plugin:3.8.4.redhat-
00002:create
-DprojectGroupId=org.acme -DprojectArtifactId=getting-started
-DplatformGroupId=com.redhat.quarkus.platform
-DplatformVersion=3.8.4.redhat-00002
-DclassName="org.acme.quickstart.GreetingResource"
-Dpath="/hello"
```

- 如果使用 Microsoft Windows PowerShell，请输入以下命令：

```
mvn com.redhat.quarkus.platform:quarkus-maven-plugin:3.8.4.redhat-
00002:create
"-DprojectGroupId=org.acme"
"-DprojectArtifactId=getting-started"
"-DplatformVersion=3.8.4.redhat-00002"
"-DplatformGroupId=com.redhat.quarkus.platform"
"-DclassName=org.acme.quickstart.GreetingResource"
"-Dpath=/hello"
```

这些命令在 ./get-started 目录中创建以下元素 ：

- Maven 项目目录结构
- org.acme.quickstart.GreetingResource 资源在 /hello上公开
- 用于以原生模式和 JVM 模式测试应用程序的相关单元测试
- 启动应用程序后，可在 http://localhost:8080 上访问的登录页面
- src/main/docker 目录中的 Dockerfile 示例
- 应用程序配置文件

注意

因为 **Mandrel** 不支持 **macOS**，所以您可以使用 **Oracle GraalVM** 在此操作系统上构建原生可执行文件。

您还可以在裸机 **Linux** 或 **Windows** 发行版上直接使用 **Oracle GraalVM** 来构建原生可执行文件。有关此过程的更多信息，请参阅 **Oracle GraalVM README** 和发行注记。

有关支持的配置的更多信息，请参阅 **Red Hat build of Quarkus** 支持的配置。

4. 创建目录结构后，在文本编辑器中打开 **pom.xml** 文件并检查文件的内容：

pom.xml

```
<project>
  ...
  <properties>
    ...
    <quarkus.platform.artifact-id>quarkus-bom</quarkus.platform.artifact-id>
    <quarkus.platform.group-id>com.redhat.quarkus.platform</quarkus.platform.group-id>
    <quarkus.platform.version>3.8.4.redhat-00002</quarkus.platform.version>
    ...
  </properties>
  ...
  <dependencyManagement>
    <dependencies>
      <dependency>
        <groupId>${quarkus.platform.group-id}</groupId>
        <artifactId>${quarkus.platform.artifact-id}</artifactId>
        <version>${quarkus.platform.version}</version>
        <type>pom</type>
        <scope>import</scope>
      </dependency>
    </dependencies>
  </dependencyManagement>
  ...
  <build>
    ...
    <plugins>
      ...
      <plugin>
        <groupId>${quarkus.platform.group-id}</groupId>
        <artifactId>quarkus-maven-plugin</artifactId>
```

```
        <version>${quarkus.platform.version}</version>
        <extensions>true</extensions>
        <executions>
          <execution>
            <goals>
              <goal>build</goal>
              <goal>generate-code</goal>
              <goal>generate-code-tests</goal>
            </goals>
          </execution>
        </executions>
      </plugin>
      ...
    </plugins>
    ...
  </build>
 ...
</project>
```

pom.xml 文件的 <dependencyManagement > 部分包含 Quarkus BOM。因此，您不需要列出 pom.xml 文件中的单个 Quarkus 依赖项版本。在这个配置文件中，您还可以找到负责打包应用程序的 quarkus-maven-plugin 插件。

5. 查看 pom.xml 文件中的 quarkus-resteasy-reactive 依赖项。这个依赖项允许您开发 REST 应用程序：

```
<dependency>
  <groupId>io.quarkus</groupId>
  <artifactId>quarkus-resteasy-reactive</artifactId>
</dependency>
```

6. 查看 src/main/java/org/acme/quickstart/GreetingResource.java 文件：

```
import jakarta.ws.rs.GET;
import jakarta.ws.rs.Path;
import jakarta.ws.rs.Produces;
import jakarta.ws.rs.core.MediaType;

@Path("/hello")
public class GreetingResource {

  @GET
  @Produces(MediaType.TEXT_PLAIN)
  public String hello() {
```

```
        return "Hello from RESTEasy Reactive";
    }
}
```

此文件包含一个简单的 REST 端点，它将 Hello from RESTEasy Reactive 返回为发送到 /hello 端点的请求。

注意

使用 Quarkus 时，支持 Jakarta REST （以前称为 JAX-RS）的应用程序类，但不是必需的。另外，只创建一个 GreetingResource 类的实例，而不是每个请求的一个实例。您可以使用不同的 *Scoped 注解来配置它，如 ApplicationScoped、RequestScoped 等等。

1.4.2. 使用 code.quarkus.redhat.com 创建 Getting Started 项目

作为应用程序开发人员，您可以使用 code.quarkus.redhat.com 生成 Quarkus Maven 项目，并自动添加并配置要在应用程序中使用的扩展。另外，code.quarkus.redhat.com 会自动管理将项目编译为原生可执行文件所需的配置参数。

您可以生成 Quarkus Maven 项目，包括以下活动：

- 指定应用程序的基本详情
- 选择您要包含在项目中的扩展
- 使用项目文件生成可下载归档
- 使用自定义命令编译和启动应用程序

先决条件

- 您有一个 Web 浏览器。
- 您已准备了环境以使用 Apache Maven。如需更多信息，请参阅准备您的环境。

- 您已配置了 Quarkus Maven 存储库。要使用 Maven 创建 Quarkus 应用程序，请使用 Red Hat-hosted Quarkus 存储库。如需更多信息，请参阅为在线存储库配置 Maven settings.xml 文件。

- 可选 ： 已安装 Quarkus 命令行界面(CLI)，这是您可以在 dev 模式中启动 Quarkus 的方法之一。

  如需更多信息，请参阅安装 Quarkus CLI。

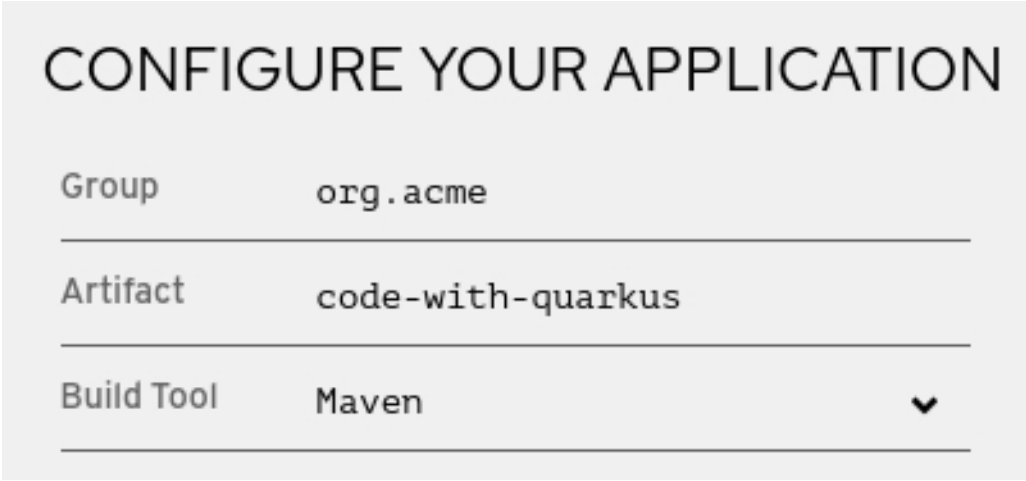
注意

Quarkus CLI 仅用于 dev 模式。红帽不支持在生产环境中使用 Quarkus CLI。

流程

1. 在您的 Web 浏览器中，访问 https://code.quarkus.redhat.com。

2. 指定项目的基本详情：



   a. 输入项目的组名称。name 格式遵循 Java 软件包命名规则，例如 org.acme。

   b. 输入项目生成的 Maven 工件的名称，如 code-with-quarkus。

   c. 选择您要用来编译和启动应用程序的构建工具。您选择的构建工具决定了以下设置：

- 生成的项目的目录结构
- 您生成的项目中使用的配置文件格式
- 在生成项目后，会显示用于编译和启动 code.quarkus.redhat.com 的自定义构建脚本和命令。

注意

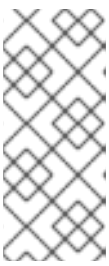
红帽仅支持使用 code.quarkus.redhat.com 创建 Quarkus Maven 项目。

3. 指定应用程序项目的更多详情：

a. 要显示包含更多应用程序详情的字段，请选择 ***More options***。

b. 输入您要用于项目生成的工件的版本。此字段的默认值为 **1.0.0-SNAPSHOT**。使用 语义版本 是首选的；但是，您可以选择指定不同类型的版本控制。

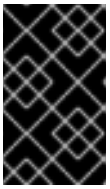
c. 选择是否希望 code.quarkus.redhat.com 将初学者代码添加到项目中。当您添加标记为 "***STARTER-CODE***" 的扩展时，您可以启用此选项在生成项目时自动为这些扩展创建示例类和资源文件。但是，如果您没有添加提供示例代码的任何扩展，这个选项不会影响生成的项目。



注意

code.quarkus.redhat.com 应用程序自动使用最新版本的 Red Hat build of Quarkus。但是，如果需要，可以在生成项目后手动更改为 pom.xml 文件中的较早 BOM 版本，但不建议这样做。

4. 选择要使用的扩展。Quarkus 应用程序包含您选择为依赖项的扩展。Quarkus 平台还确保这些扩展与将来的版本兼容。



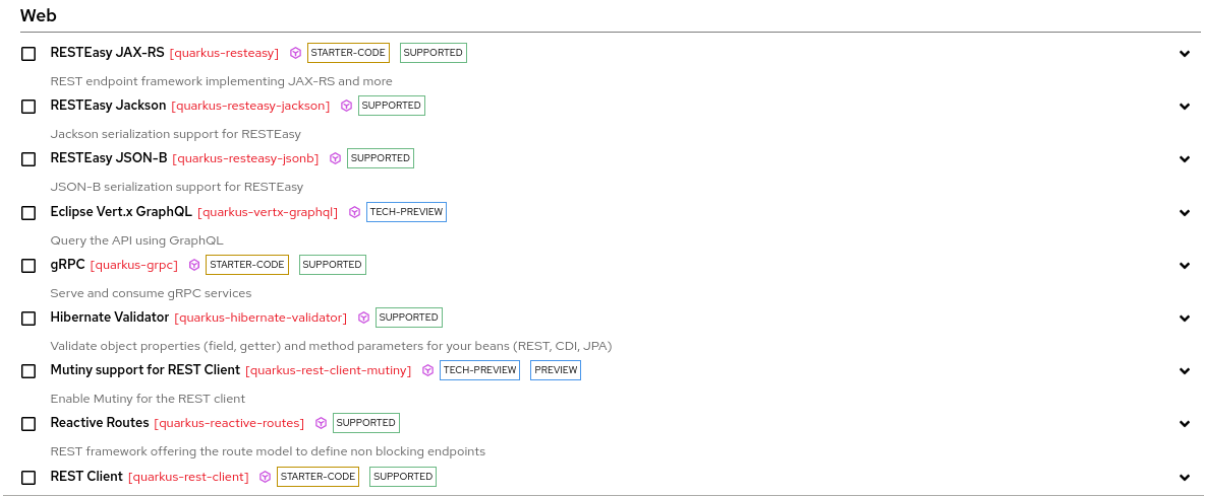
重要

不要在同一项目中使用 RESTEasy 和 RESTEasy 主动扩展。

扩展旁边的 quark 图标(

)表示扩展是 Red Hat build of Quarkus 平台发行版本的一部分。红帽更喜欢使用同一平台中的扩展，因为它们被一起测试和验证，因此更易于使用和升级。

您可以启用选项，为标记为"***STARTER-CODE***"的扩展自动生成初学程序代码。



5. 要确认您的选择，请选择 **Generate your application**。显示的对话框显示以下项目：

- 下载包含您生成的项目的存档的链接
- 可用于编译和启动应用程序的命令

6. 要将带有生成的项目文件的存档保存到机器中，请选择 **Download the ZIP**。

7. 提取存档的内容。

8. 进入包含您提取的项目文件的目录：

```
cd <directory_name>
```

9. 要在 **dev** 模式中编译并启动应用程序，请使用以下方法之一：

- 使用 **Maven**：

```
./mvnw quarkus:dev
```

- 使用 **Quarkus CLI**：

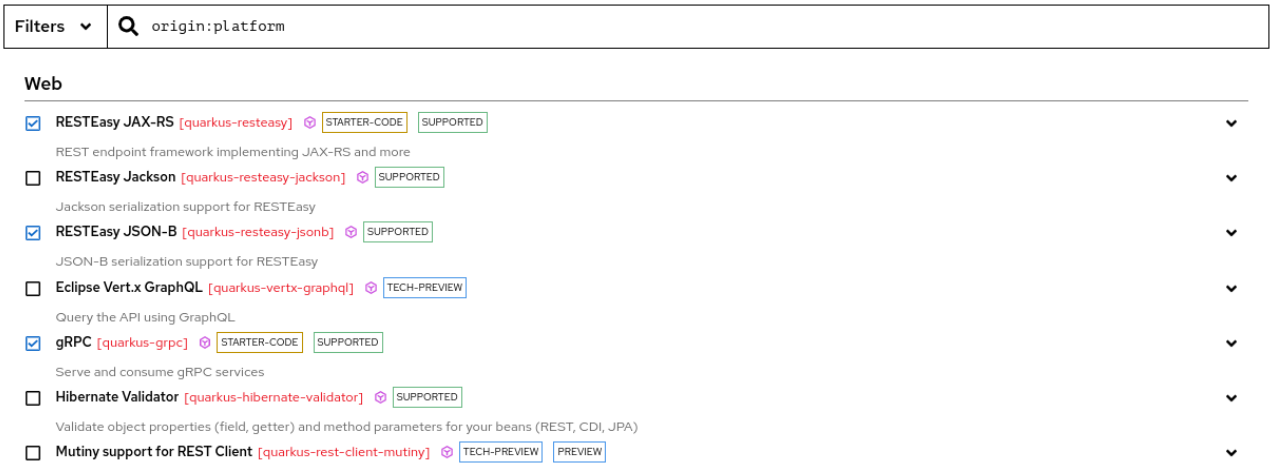
```
quarkus dev
```

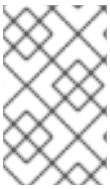
其他资源

- " Red Hat build of Quarkus"指南中的红帽构建的 Quarkus 扩展支持级别。

1.4.2.1. 红帽构建的 Quarkus 扩展的支持等级

红帽为 **code.quarkus.redhat.com** 上可用的扩展提供了 不同级别的支持，您可以添加到 **Quarkus** 项目。每个扩展名称旁的标签代表支持级别。





注意

红帽不支持在生产环境中使用未标记扩展。

红帽为 **Red Hat build of Quarkus** 扩展提供以下级别的支持：

表 1.1. 红帽为 Quarkus 扩展提供支持级别

支持级别	描述
支持	红帽完全支持在生产环境中使用企业应用程序的扩展。
TECH-PREVIEW	红帽在 技术预览功能支持范围 下，对生产环境提供有限的支持。
预览	红帽不支持在生产环境中使用扩展。扩展只在 Quarkus 社区版本中提供。
DEV-SUPPORT	红帽不支持在生产环境中使用扩展，但红帽开发人员支持它们在开发新应用程序时提供的核心功能。
弃用	红帽计划将扩展替换为提供相同功能的最新技术或实施。
STARTER-CODE	您可以自动生成扩展示例代码。

点击每个扩展旁的箭头图标(DESTINATION)，您可以扩展 overflow 菜单来访问该扩展的进一步操作。例如：

- **在命令行中使用 Quarkus Maven 插件将扩展添加到现有项目中**

- 复制 XML 片段，将扩展添加到项目的 pom.xml 文件中
- 获取每个扩展的 groupId、artifactId、和版本
- 打开扩展指南

1.4.3. 使用 Red Hat build of Quarkus CLI 创建 Getting Started 项目

您可以使用 Quarkus 命令行界面(CLI)创建 get -started 项目。

使用 Quarkus CLI，您可以创建项目、管理扩展并运行构建和开发命令。

重要

Quarkus CLI 仅用于 dev 模式。红帽不支持在生产环境中使用 Quarkus CLI。

先决条件

- 已安装 Quarkus CLI。如需更多信息，请参阅准备您的环境。
- 您已将 Quarkus 开发人员工具配置为访问扩展 registry 中的扩展。如需更多信息，请参阅配置红帽构建的 Quarkus 扩展 registry 客户端。

流程

1. 要生成项目，在命令终端中输入以下命令：

```
quarkus create && cd code-with-quarkus
```

注意

您还可以指定 'app' 子命令，例如 quarkus create app。但是，这不是强制要求，因为如果没有指定 'app' 子命令就意味着它。

使用此命令，您可以在当前工作目录的名为 'code-with-quarkus' 的文件夹中创建 Quarkus 项目。

2. 默认情况下，groupId、artifactId 和 version 属性使用以下默认值指定：

- groupId='org.acme'
- artifactId='code-with-quarkus'
- version='1.0.0-SNAPSHOT'

要更改 groupId,artifactId, 和 version 属性的值，请发出 quarkus create 命令并在 CLI 中指定以下语法：

```
groupId:artifactId:version
```

例如，quarkus create app mygroupId:myartifactid:version

注意

要查看所有可用 Quarkus 命令的信息，请指定 help 参数：

```
quarkus --help
```

3. 在文本编辑器中查看 src/main/java/org/acme/GreetingResource.java 文件：

```
package org.acme;

import jakarta.ws.rs.GET;
import jakarta.ws.rs.Path;
import jakarta.ws.rs.Produces;
import jakarta.ws.rs.core.MediaType;

@Path("/hello")
public class GreetingResource {
```

```
    @GET
    @Produces(MediaType.TEXT_PLAIN)
    public String hello() {
        return "Hello from RESTEasy Reactive";
    }
}
```

此文件包含一个简单的 REST 端点，它将 Hello from RESTEasy Reactive 返回为发送到 /hello 端点的请求。

验证

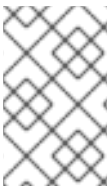
1. 在 dev 模式中编译并启动您的应用程序。如需更多信息，请参阅 编译并启动红帽构建的 Quarkus Getting Started 项目。

2. 从 Quarkus CLI 软件包并运行您的 Getting Started 项目。如需更多信息，请参阅打包并运行红帽构建的 Quarkus 入门应用程序。

1.5. 编译并启动红帽构建的 QUARKUS 入门项目

创建 Quarkus Getting Started 项目后，您可以编译 Hello 应用，并验证 hello 端点是否返回 "Hello from RESTEasy Reactive。

此流程使用 Quarkus 内置 dev 模式，因此您可以在应用程序运行时更新应用程序源和配置。您所做的更改会出现在正在运行的应用程序中。



注意

用于编译 Quarkus Hello 应用的命令取决于您在机器上安装的开发人员工具。

先决条件

- 您已创建了 Quarkus Getting Started 项目。

流程

1. 前往 项目目录。

2. 要在 dev 模式中编译 Quarkus Hello 应用程序，请使用以下方法之一，具体取决于您要使用的开发人员工具：

- 如果您希望使用 Apache Maven，请输入以下命令：

```
mvn quarkus:dev
```

- 如果要使用 Quarkus 命令行界面(CLI)，请输入以下命令：

```
quarkus dev
```

- 如果要使用 Maven 包装程序，请输入以下命令：

```
./mvnw quarkus:dev
```

预期输出

以下提取显示了预期输出的示例：

```
INFO  [io.quarkus] (Quarkus Main Thread) Profile dev activated. Live Coding activated.
INFO  [io.quarkus] (Quarkus Main Thread) Installed features: [cdi, resteasy-reactive,
smallrye-context-propagation, vertx]
```

验证

- 要将请求发送到应用程序提供的端点，请在一个新的终端窗口中输入以下命令：

```
curl -w "\n" http://localhost:8080/hello
Hello from RESTEasy Reactive
```

注意

"\n" 属性会在命令的输出前自动添加新行，这样可防止您的终端打印 '%' 字符，或者将结果和下一个 shell 提示符放在同一行中。

1.6. 使用红帽构建的 QUARKUS 依赖项注入

通过依赖项注入，您可以以完全独立于任何客户端使用的方式使用服务。它将客户端依赖项的创建与客

户端的行为分开，这使得程序设计能够松散耦合。

红帽构建的 Quarkus 中的依赖注入基于 Quarkus ArC，它是基于上下文和依赖注入(CDI)的基于构建时导向的依赖项注入解决方案，专为 Quarkus 架构量身定制。因为 ArC 是 quarkus-resteasy-reactive 的传输依赖项，并且 quarkus-resteasy-reactive 是项目的依赖项，所以 ArC 已下载。

先决条件

- 您已创建了 Quarkus Getting Started 项目。

流程

1. 要修改应用程序并添加 companion bean，请使用以下内容创建 src/main/java/org/acme/quickstart/GreetingService.java 文件：

```
package org.acme.quickstart;

import jakarta.enterprise.context.ApplicationScoped;

@ApplicationScoped
public class GreetingService {

    public String greeting(String name) {
        return "hello " + name;
    }

}
```

2. 编辑 src/main/java/org/acme/quickstart/GreetingResource.java 以注入 GreetingService，并使用它来创建新端点：

```
package org.acme.quickstart;

import jakarta.inject.Inject;
import jakarta.ws.rs.GET;
import jakarta.ws.rs.Path;
import jakarta.ws.rs.Produces;
import jakarta.ws.rs.core.MediaType;

@Path("/hello")
public class GreetingResource {

    @Inject
    GreetingService service;

    @GET
```

```
    @Produces(MediaType.TEXT_PLAIN)
    @Path("/greeting/{name}")
    public String greeting(String name) {
        return service.greeting(name);
    }

    @GET
    @Produces(MediaType.TEXT_PLAIN)
    public String hello() {
        return "Hello from RESTEasy Reactive";
    }
}
```

3. 如果您停止了应用程序，请输入以下命令重启它：

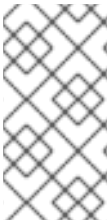
```
./mvnw quarkus:dev
```

4. 要验证端点是否返回 hello quarkus，请在一个新的终端窗口中输入以下命令：

```
curl -w "\n" http://localhost:8080/hello/greeting/quarkus
hello quarkus
```

1.7. 测试您的红帽构建的 QUARKUS 应用程序

编译 Quarkus Getting Started 项目后，您可以使用 JUnit 5 框架测试应用程序来验证其是否如预期运行。



注意

另外，您可以启用对 Quarkus 应用程序的持续测试。如需更多信息，请参阅启用和运行持续测试。

Quarkus 项目在 pom.xml 文件中生成以下两个测试依赖项：

- quarkus-junit5: 测试需要，因为它提供了控制 JUnit 5 测试框架的 @QuarkusTest 注解。
- rest-assured: 不需要 rest-assured 依赖项，因为它提供了一种便捷的方式来测试 HTTP 端点，因此集成它。rest-assured 依赖项会自动设置正确的 URL，因此不需要配置。

pom.xml 文件示例：

```
<dependency>
    <groupId>io.quarkus</groupId>
    <artifactId>quarkus-junit5</artifactId>
    <scope>test</scope>
</dependency>
<dependency>
    <groupId>io.rest-assured</groupId>
    <artifactId>rest-assured</artifactId>
    <scope>test</scope>
</dependency>
```

注意

这些测试使用 REST-Assured 框架，但如果您愿意，您可以使用不同的库。

先决条件

- 您已编译了 Quarkus Getting Started 项目。如需更多信息，请参阅 编译并启动红帽构建的 Quarkus Getting Started 项目。

流程

1. 打开生成的 pom.xml 文件并查看内容：

```
<plugin>
    <artifactId>maven-surefire-plugin</artifactId>
    <version>${surefire-plugin.version}</version>
    <configuration>
      <systemPropertyVariables>

<java.util.logging.manager>org.jboss.logmanager.LogManager</java.util.logging.manager
>
        <maven.home>${maven.home}</maven.home>
      </systemPropertyVariable>
    </configuration>
</plugin>
```

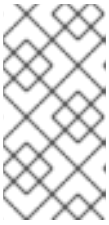
请注意以下属性的值：

- 设置 java.util.logging.manager 系统属性，以确保您的应用程序使用正确的日志管理器进行测试。

- maven.home 属性指向 settings.xml 文件的位置，您可以在其中存储您要应用到项目的自定义 Maven 配置。

2. 编辑 src/test/java/org/acme/quickstart/GreetingResourceTest.java 文件，使其与以下内容匹配：

```
package org.acme.quickstart;

import io.quarkus.test.junit.QuarkusTest;
import org.junit.jupiter.api.Test;

import java.util.UUID;

import static io.restassured.RestAssured.given;
import static org.hamcrest.CoreMatchers.is;

@QuarkusTest
public class GreetingResourceTest {

    @Test
    public void testHelloEndpoint() {
        given()
          .when().get("/hello")
          .then()
             .statusCode(200)
             .body(is("Hello from RESTEasy Reactive"));
    }

    @Test
    public void testGreetingEndpoint() {
        String uuid = UUID.randomUUID().toString();
        given()
          .pathParam("name", uuid)
          .when().get("/hello/greeting/{name}")
          .then()
            .statusCode(200)
            .body(is("hello " + uuid));
    }

}
```



注意

通过使用 QuarkusTest 运行程序，您可以指示 JUnit 在启动测试前启动应用。

3. 要从 Maven 运行测试，请输入以下命令：

```
./mvnw test
```

注意

您还可以从 IDE 运行测试。如果您这样做，请首先停止应用程序。

默认情况下，测试在端口 8081 上运行，因此它们不会与正在运行的应用程序冲突。在 Quarkus 中，RestAssured 依赖项配置为使用此端口。

注意

如果要使用其他客户端，请使用 @TestHTTPResource 注释，将测试应用的 URL 直接注入 Test 类中的字段。此字段可以是 字符串、URL 或 URI 类型。您也可以在 @TestHTTPResource 注释中输入测试路径。例如，要测试向 /foo 公开的资源，请在测试中添加以下行：

```
@TestHTTPResource("/foo")
URL testUrl;
```

4. 如有必要，在 quarkus.http.test-port 配置属性中指定测试端口。

1.8. 启用并运行持续测试

使用红帽构建的 Quarkus，您可以在开发应用程序时持续测试代码更改。Quarkus 提供了一个连续的测试功能，您可以在进行并保存代码更改后立即运行。

运行持续测试时，测试会在启动应用程序后暂停。您可以在应用程序启动时立即恢复测试。Quarkus 应用程序决定运行哪个测试，以便测试仅在更改的代码中运行。

Quarkus 的持续测试功能会被默认启用。您可以通过将 src/main/resources/application.properties 文件中的 quarkus.test.continuous-testing 属性设置为 disabled 来禁用持续测试。

注意

如果您之前禁用了持续测试并希望再次启用它，则必须在开始测试前重启 Quarkus 应用程序。

先决条件

- 您已编译了 Quarkus Getting Started 应用程序（或任何其他应用程序）。如需更多信息，请参阅 编译并启动红帽构建的 Quarkus Getting Started 项目。

流程

1. 启动 Quarkus 应用程序。

   - 如果您使用 code.quarkus.redhat.com 或 Quarkus CLI 创建 Getting Project 项目，则生成项目时会提供 Maven 包装器。在项目目录中输入以下命令：

     ```
     ./mvnw quarkus:dev
     ```

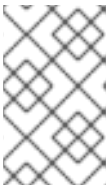
   - 如果您使用安装在您的机器上的 Apache Maven 创建 Getting Project 项目，请输入以下命令：

     ```
     mvn quarkus:dev
     ```

   - 如果您在 dev 模式下运行持续测试并使用 Quarkus CLI，请输入以下命令：

     ```
     quarkus dev
     ```

2. 查看生成的输出日志中测试状态的详情。



注意

要查看输出日志，您可能需要滚动到屏幕的底部。

- 启用持续测试时，会显示以下信息：

```
Press [e] to edit command line args (currently ''), [r] to re-run, [o] Toggle test output, [:] for the terminal, [h] for more options>
```

- 当持续测试暂停时，会显示以下信息：

```
Press [e] to edit command line args (currently ''), [r] to resume testing, [o] Toggle test output, [:] for the terminal, [h] for more options>
```

注意

默认情况下，当启用持续测试时，在启动应用程序后会暂停测试。要查看可用于控制测试方式的键盘命令，请参阅控制连续测试的命令。

3. 要开始运行测试，请在键盘上按 r。

4. 查看更新后的输出日志，以监控测试状态和测试结果、检查测试统计信息以及获取后续操作的指导。例如：

```
All 2 tests are passing (0 skipped), 2 tests were run in 2094ms. Tests completed at 14:45:11.
Press [e] to edit command line args (currently ''), [r] to re-run, [o] Toggle test output, [:] for the terminal, [h] for more options>
```

验证

1. 进行代码更改。例如，在文本编辑器中，打开 src/main/java/org/acme/quickstart/GreetingsResource.java 文件。

2. 更改"hello"端点，以返回"Hello world"并保存文件。

```
import jakarta.ws.rs.GET;
import jakarta.ws.rs.Path;
import jakarta.ws.rs.Produces;
import jakarta.ws.rs.core.MediaType;

@Path("/hello")
public class GreetingResource {

    @GET
    @Produces(MediaType.TEXT_PLAIN)
    public String hello() {
```

```
        return "Hello world";
    }
}
```

3. 验证 **Quarkus** 是否立即重新运行测试来测试更改的代码。

4. 查看输出日志以检查测试结果。在本例中，测试会检查更改的字符串是否包含值 **"Hello from RESTEasy Reactive"**。测试会失败，因为字符串已更改为 **"Hello world"**。

```
2024-02-26 15:03:45,911 ERROR [io.qua.test] (Test runner thread) Test
GreetingResourceTest#testHelloEndpoint() failed: java.lang.AssertionError: 1 expectation
failed.
Response body doesn't match expectation.
Expected: is "Hello from RESTEasy Reactive"
  Actual: Hello world
        at
io.restassured.internal.ValidatableResponseOptionsImpl.body(ValidatableResponseOptionsImp
.java:238)
        at
org.acme.quickstart.GreetingResourceTest.testHelloEndpoint(GreetingResourceTest.java:20)

--
1 test failed (1 passing, 0 skipped), 2 tests were run in 2076ms. Tests completed at 15:03:45.
Press [e] to edit command line args (currently ''), [r] to re-run, [o] Toggle test output, [:] for the
terminal, [h] for more options>
```

5. 要退出持续测试，请在键盘上按 **Ctrl-C** 或 **q**。

注意

如果再次将值更改回 **"Hello from RESTEasy Reactive"**，则测试会再次运行。

1.8.1. 控制持续测试的命令

您可以使用键盘上的热键命令来控制用于持续测试的选项。要查看命令的完整列表，请在键盘中按 **'h'**。可用的选项如下：

命令	描述
r	重新运行所有测试。
f	重新运行失败的所有测试。

命令	描述
b	切换 'broken only' 模式。只有之前失败的测试才会运行，即使其他测试会受到您的代码更改的影响。如果您更改许多测试使用的代码，但您想要查看失败的测试，这个选项可能很有用。
v	输出详细描述了从上次测试运行到控制台的测试失败。如果在上次测试运行后有相当数量的控制台输出，则此选项可能很有用。
p	临时暂停正在运行的测试。如果您要进行许多代码更改，这可能很有用，但您不想在完成更改前获得测试反馈。
q	退出持续测试。
o	将测试输出输出到控制台。这默认是禁用的。当禁用测试输出时，输出会被过滤并保存，但不会在控制台中显示。您可以在 Development UI 中查看测试输出。
i	切换基于检测的重新加载。使用这个选项不会影响测试，但允许进行实时重新加载。如果更改不会影响类的结构，这可能会很有用。
l	切换实时重新加载。使用此选项不会影响测试，但您可以打开和关闭实时重新加载。
s	强制重启。使用这个选项，您可以强制扫描更改的文件以及包含更改的实时重新加载。请注意，即使没有代码更改，并且禁用实时重新加载，应用仍会重新启动。

1.9. 打包并运行红帽构建的 QUARKUS GETTING STARTED 应用程序

编译 Quarkus 入门项目后，您可以将其打包在 JAR 文件中，并从命令行运行它。

注意

用于打包和运行 Quarkus Getting Started 应用程序的命令取决于您在机器上安装的开发人员工具。

先决条件

- 您已编译了 Quarkus Getting Started 项目。

流程

1. 前往 getting-started 项目目录。
2. 要打包 Quarkus 入门项目，请使用以下方法之一，具体取决于您要使用的开发人员工具：
   - 如果您希望使用 Apache Maven，请输入以下命令：

```
mvn package
```

   - 如果要使用 Quarkus 命令行界面(CLI)，请输入以下命令：

```
quarkus build
```

   - 如果要使用 Maven 包装程序，请输入以下命令：

```
./mvnw package
```

     这个命令在 /target 目录中生成以下 JAR 文件：

     - getting-started-1.0-0-SNAPSHOT.jar ：包含项目的类和资源。这是 Maven 构建生成的常规工件。
     - quarkus-app/quarkus-run.jar: 是一个可执行 JAR 文件。此文件不是 uber-JAR 文件。依赖项被复制到 target/quarkus-app/lib 目录中。
3. 要启动应用程序，请输入以下命令：

```
java -jar target/quarkus-app/quarkus-run.jar
```

注意

- 在运行应用程序前，请确保停止 dev 模式（按 CTRL+C），或者您将存在端口冲突。
- quarkus-run.jar 文件中的 MANIFEST.MF 文件的 Class-Path 条目明确列出 lib 目录中的 JAR 文件。如果要从另一个位置部署应用程序，您必须部署整个 quarkus-app 目录。

重要

各种红帽构建的 Quarkus 扩展贡献了非应用程序端点，它们提供有关应用程序的不同信息。例如，quarkus-smallrye-health、quarkus-micrometer-registry-prometheus 和 quarkus-smallrye-openapi 扩展。

您可以通过指定 /q 前缀来访问这些非应用程序端点。例如：/q/health,/q/metrics,/q/openapi。

对于可能会存在安全风险的非应用程序端点，您可以选择使用专用管理界面在不同的 TCP 端口下公开这些端点。如需更多信息，请参阅 Quarkus 管理界面参考指南。

1.10. JVM 和原生构建模式

您可以编译典型的 Java 虚拟机(JVM)应用程序，或使用 Mandrel 或 GraalVM 的 native-image 工具编译原生应用程序。

1.10.1. 将应用程序编译为典型的 JVM 应用程序

您可以将应用程序编译为 JVM 应用程序。这个选项基于 quarkus.package.type 配置属性，并生成以下文件之一：

- fast-jar ：针对 Quarkus 和默认配置选项优化的 JAR 文件。可以更快地启动时间，并稍微减少内存用量。
- legacy-jar ：典型的 JAR 文件。

- uber-jar ：单个独立 JAR 文件。

  这些 JAR 文件可用于所有操作系统，构建速度比原生镜像快。

1.10.2. 将应用程序编译到原生镜像中

您可以将应用程序编译到原生镜像。为此，您可以将 quarkus.package.type 配置属性设置为 native。

使用此属性，您可以创建一个为您选择的操作系统（如 Windows 的 .exe 文件）编译的可执行二进制文件。这些文件的启动时间比 JAVA JAR 文件要快得多，但其编译需要几分钟时间。此外，使用原生二进制文件可达到的最大吞吐量比常规 JVM 应用低，因为缺少配置文件指南的优化。

- 使用 Mandrel

  Mandrel 是 GraalVM for Red Hat build of Quarkus 的专用发行版，也是构建针对 Linux 容器化环境的原生可执行文件的首选方法。虽然 Mandrel 方法非常适合将编译输出嵌入到容器化环境中，但只提供了 Linux64-bit 原生可执行文件。因此， .exe 等结果不是选项。

  我们鼓励 Mandrel 用户使用容器来构建其原生可执行文件。

  要使用官方 Mandrel 镜像使用本地 Docker 或 Podman 安装将应用程序编译到原生模式，请输入具有以下属性的 mvn package 命令：

```
-Dquarkus.package.type=native
-Dquarkus.native.container-build=true
-Dquarkus.native.builder-image=registry.access.redhat.com/quarkus/mandrel-for-jdk-21-rhel8:23.1
```

  - 有关如何使用 Mandrel 构建原生可执行文件的详情，请参考 将红帽构建的 Quarkus 应用程序编译到原生可执行文件

  - 有关可用 Mandrel 镜像的列表，请参阅 可用的 Mandrel 镜像

- 使用 GraalVM

因为 Mandrel 不支持 macOS，所以您可以使用 Oracle GraalVM 在此操作系统上构建原生可执行文件。

您还可以在裸机 Linux 或 Windows 发行版上直接使用 Oracle GraalVM 来构建原生可执行文件。

有关如何使用 Oracle GraalVM 构建原生可执行文件的详情，请参考 将红帽构建的 Quarkus 应用程序编译到原生可执行文件。

其他资源

- 有关构建、编译、打包和调试原生可执行文件的更多信息，请参阅 构建原生可执行文件。
- 有关帮助排除在尝试以原生可执行文件运行 Java 应用程序时可能出现的问题 的提示，请参阅提示编写原生应用程序。

1.11. 以原生模式打包并运行红帽构建的 QUARKUS GETTING STARTED 应用程序

在原生模式中，应用构建的输出是一个独立于平台的原生二进制文件，而不是压缩或存档 JAR 文件。有关原生模式与 Java 虚拟机(JVM)不同方式的更多信息，请参阅入门指南中的 JVM 和原生构建模式 章节。

先决条件

- 已安装 OpenJDK 17 或 21，并设置 JAVA_HOME 环境变量来指定 Java SDK 的位置。
- 已安装 Apache Maven 3.8.6 或更高版本。
- 您有一个正常工作的 C 开发环境。
- 您有一个正常工作的容器运行时，如 Docker 或 Podman。
- 可选 ： 如果要使用 Quarkus 命令行界面(CLI)，请确保已安装它。

- 有关如何安装 Quarkus CLI 的说明，请参考 Quarkus CLI 中的特定于社区的信息。

- 您已克隆并编译了 Quarkus Getting Started 项目。
- 您已下载并安装了一个社区或企业级的 GraalVM 版本。
  - 要下载和安装社区或 GraalVM 的企业级版本，请参阅 GraalVM 的官方入门 文档。
  - 或者，使用特定于平台的安装工具，如 sdkman、homebrew 或 scoop。

注意

虽然您可以使用 GraalVM 的社区版本完成入门指南中的所有流程，但红帽构建的 Quarkus 生产环境不支持 GraalVM 社区版本。如需更多信息，请参阅将 红帽构建的 Quarkus 应用程序编译到原生可执行文件。

流程

1. 通过将 GRAALVM_HOME 环境变量设置为 GraalVM 安装目录来配置运行时环境。例如：

```
export GRAALVM_HOME=$HOME/Development/graalvm/
```

   - 在 macOS 中，将变量指向 Home 子目录：

```
export GRAALVM_HOME=$HOME/Development/graalvm/Contents/Home/
```

   - 在 Windows Server 上，使用 Control Panel 设置环境变量。

2. 安装 native-image 工具：

```
${GRAALVM_HOME}/bin/gu install native-image
```

3. 将 JAVA_HOME 环境变量设置为 GraalVM 安装目录：

```
export JAVA_HOME=${GRAALVM_HOME}
```

4. 在路径中添加 GraalVM bin 目录：

```
export PATH=${GRAALVM_HOME}/bin:$PATH
```

5. 进入 Getting Started 项目文件夹：

```
cd getting-started
```

6. 使用以下方法之一编译原生镜像：

- 使用 Maven：

```
mvn clean package -Pnative
```

- 使用 Quarkus CLI：

```
quarkus build --native
```

验证

1. 启动应用程序：

```
./target/getting-started-1.0.0-SNAPSHOT-runner
```

2. 观察日志消息，并验证它是否包含 原生 词语：

```
2024-02-15 09:51:51,505 INFO [io.quarkus] (main) getting-started 1.0.0-SNAPSHOT native (powered by Red Hat build of Quarkus 3.8.4.redhat-00002) started in 0.043s. Listening on: http://0.0.0.0:8080
```

其他资源

- 有关其他提示或故障排除信息，请参阅 Quarkus 构建原生可执行文件 指南。

1.12. 其他资源

- 将红帽构建的 Quarkus 应用程序部署到 OpenShift Container Platform

更新于 2024-05-10