Red Hat build of OpenJDK 21

使用 jlink 自定义 Java 运行时环境

Last Updated: 2024-05-10

法律通告

摘要

红帽构建的 OpenJDK 21 是 Red Hat Enterprise Linux 平台上的红帽产品。使用 jlink 自定义 Java 运行时镜像 指南提供了 Jlink 的概述，并解释了如何使用 jlink 创建自定义 Java 运行时镜像。

目录

提供有关红帽构建的 OPENJDK 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 属性中存在问题的语言。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 JLINK 概述

Jlink 是一个 Java 命令行工具，用于生成自定义 Java 运行时环境(JRE)。您可以使用自定义 JRE 来运行 Java 应用程序。

使用 jlink，您可以创建一个仅包含相关类文件的自定义运行时环境。

第 2 章 为非模块化应用程序创建自定义 JAVA 运行时环境

您可以使用 **jlink** 工具从非模块化应用程序创建自定义 Java 运行时环境。

先决条件

- 使用存档 在 RHEL 上安装红帽构建的 OpenJDK 。

注意

为获得最佳结果，请使用可移植的红帽二进制文件作为 Jlink 运行时的基础，因为这些二进制文件包含捆绑的库。

流程

1. 使用 **Logger** 类创建一个简单的 Hello World 应用。
   a. 检查 **jdk-17** 文件夹中是否存在 OpenJDK 21 二进制文件的基本红帽构建：

```
$ ls jdk-17
bin conf demo include jmods legal lib man NEWS release
$ ./jdk-17/bin/java -version
openjdk version "17.0.10" 2021-01-19 LTS
OpenJDK Runtime Environment 18.9 (build 17.0.10+9-LTS)
OpenJDK 64-Bit Server VM 18.9 (build 17.0.10+9-LTS, mixed mode)
```

   b. 为应用程序创建一个目录：

```
$ mkdir -p hello-example/sample
```

   c. 使用以下内容创建 **hello-example/sample/HelloWorld.java** 文件：

```
package sample;

import java.util.logging.Logger;

public class HelloWorld {
    private static final Logger LOG = Logger.getLogger(HelloWorld.class.getName());
    public static void main(String[] args) {
        LOG.info("Hello World!");
    }
}
```

   d. 编译应用程序：

```
$ ./jdk-17/bin/javac -d . $(find hello-example -name \*.java)
```

   e. **在没有自定义 JRE 的情况下运行** 您的应用程序：

```
$ ./jdk-17/bin/java sample.HelloWorld
Mar 09, 2021 10:48:59 AM sample.HelloWorld main
INFO: Hello World!
```

前面的例子显示了需要 311 MB 的 OpenJDK 基础构建来运行单个类。

f. *（可选）* 您可以检查红帽构建的 OpenJDK，并为您的应用程序查看许多非必需的模块：

```
$ du -sh jdk-17/
313M jdk-17/
```

```
$ ./jdk-17/bin/java --list-modules
java.base@17.0.1
java.compiler@17.0.1
java.datatransfer@17.0.1
java.desktop@17.0.1
java.instrument@17.0.1
java.logging@17.0.1
java.management@17.0.1
java.management.rmi@17.0.1
java.naming@17.0.1
java.net.http@17.0.1
java.prefs@17.0.1
java.rmi@17.0.1
java.scripting@17.0.1
java.se@17.0.1
java.security.jgss@17.0.1
java.security.sasl@17.0.1
java.smartcardio@17.0.1
java.sql@17.0.1
java.sql.rowset@17.0.1
java.transaction.xa@17.0.1
java.xml@17.0.1
java.xml.crypto@17.0.1
jdk.accessibility@17.0.1
jdk.attach@17.0.1
jdk.charsets@17.0.1
jdk.compiler@17.0.1
jdk.crypto.cryptoki@17.0.1
jdk.crypto.ec@17.0.1
jdk.dynalink@17.0.1
jdk.editpad@17.0.1
jdk.hotspot.agent@17.0.1
jdk.httpserver@17.0.1
jdk.incubator.foreign@17.0.1
jdk.incubator.vector@17.0.1
jdk.internal.ed@17.0.1
jdk.internal.jvmstat@17.0.1
jdk.internal.le@17.0.1
jdk.internal.opt@17.0.1
jdk.internal.vm.ci@17.0.1
jdk.internal.vm.compiler@17.0.1
jdk.internal.vm.compiler.management@17.0.1
jdk.jartool@17.0.1
jdk.javadoc@17.0.1
jdk.jcmd@17.0.1
jdk.jconsole@17.0.1
jdk.jdeps@17.0.1
jdk.jdi@17.0.1
jdk.jdwp.agent@17.0.1
```

```
jdk.jfr@17.0.1
jdk.jlink@17.0.1
jdk.jpackage@17.0.1
jdk.jshell@17.0.1
jdk.jsobject@17.0.1
jdk.jstatd@17.0.1
jdk.localedata@17.0.1
jdk.management@17.0.1
jdk.management.agent@17.0.1
jdk.management.jfr@17.0.1
jdk.naming.dns@17.0.1
jdk.naming.rmi@17.0.1
jdk.net@17.0.1
jdk.nio.mapmode@17.0.1
jdk.random@17.0.1
jdk.sctp@17.0.1
jdk.security.auth@17.0.1
jdk.security.jgss@17.0.1
jdk.unsupported@17.0.1
jdk.unsupported.desktop@17.0.1
jdk.xml.dom@17.0.1
jdk.zipfs@17.0.1
```

这个示例 **Hello World** 应用有非常少的依赖项。您可以使用 jlink 为应用程序创建自定义运行时镜像。使用这些镜像，您只能使用所需的红帽构建 OpenJDK 依赖项来运行应用程序。

2. 使用 **jdeps** 命令确定应用程序的模块依赖项：

```
$ ./jdk-17/bin/jdeps -s ./sample/HelloWorld.class
HelloWorld.class -> java.base
HelloWorld.class -> java.logging
```

3. 为您的应用程序构建自定义 java 运行时镜像：

```
$ ./jdk-17/bin/jlink --add-modules java.base,java.logging --output custom-runtime
$ du -sh custom-runtime
50M custom-runtime/
$ ./custom-runtime/bin/java --list-modules
java.base@17.0.10
java.logging@17.0.10
```

注意

红帽构建的 OpenJDK 将自定义 Java 运行时镜像的大小从 313 M 运行时镜像减小到 50 M 运行时镜像。

4. 您可以验证应用程序的运行时减少：

```
$ ./custom-runtime/bin/java sample.HelloWorld
Jan 14, 2021 12:13:26 PM HelloWorld main
INFO: Hello World!
```

生成的 JRE 和您的示例应用程序没有任何其他依赖项。

您可以将应用程序与自定义运行时一起分发用于部署。

注意

您必须使用基本红帽构建 OpenJDK 的基本安全更新，为应用程序重建自定义 Java 运行时镜像。

第 3 章 为模块化应用程序创建自定义 JAVA 运行时环境

您可以使用 **jlink** 工具从模块化应用程序创建自定义 Java 运行时环境。

先决条件

- 使用存档 在 RHEL 上安装红帽构建的 OpenJDK 。

注意

为获得最佳结果，请使用可移植的红帽二进制文件作为 Jlink 运行时的基础，因为这些二进制文件包含捆绑的库。

流程

1. 使用 **Logger** 类创建一个简单的 Hello World 应用。
   a. 检查 **jdk-17** 文件夹中是否存在 OpenJDK 21 二进制文件的基本红帽构建：

```
$ ls jdk-17
bin  conf  demo  include  jmods  legal  lib  man  NEWS  release
$ ./jdk-17/bin/java -version
openjdk version "17.0.10" 2021-01-19 LTS
OpenJDK Runtime Environment 18.9 (build 17.0.10+9-LTS)
OpenJDK 64-Bit Server VM 18.9 (build 17.0.10+9-LTS, mixed mode)
```

   b. 为应用程序创建一个目录：

```
$ mkdir -p hello-example/sample
```

   c. 使用以下内容创建 **hello-example/sample/HelloWorld.java** 文件：

```
package sample;

import java.util.logging.Logger;

public class HelloWorld {
    private static final Logger LOG = Logger.getLogger(HelloWorld.class.getName());
    public static void main(String[] args) {
        LOG.info("Hello World!");
    }
}
```

   d. 创建名为 **hello-example/module-info.java** 的文件，并在文件中包含以下代码：

```
module sample
{
    requires java.logging;
}
```

   e. 编译应用程序：

```
$ ./jdk-17/bin/javac -d example $(find hello-example -name \*.java)
```

f. *在没有自定义 JRE 的情况下运行* 您的应用程序：

```
$ ./jdk-17/bin/java -cp example sample.HelloWorld
Mar 09, 2021 10:48:59 AM sample.HelloWorld main
INFO: Hello World!
```

前面的例子显示了需要 311 MB 的 OpenJDK 基础构建来运行单个类。

g. *（可选）* 您可以检查红帽构建的 OpenJDK，并为您的应用程序查看许多非必需的模块：

```
$ du -sh jdk-17/
313M jdk-17/
```

```
$ ./jdk-17/bin/java --list-modules
java.base@17.0.1
java.compiler@17.0.1
java.datatransfer@17.0.1
java.desktop@17.0.1
java.instrument@17.0.1
java.logging@17.0.1
java.management@17.0.1
java.management.rmi@17.0.1
java.naming@17.0.1
java.net.http@17.0.1
java.prefs@17.0.1
java.rmi@17.0.1
java.scripting@17.0.1
java.se@17.0.1
java.security.jgss@17.0.1
java.security.sasl@17.0.1
java.smartcardio@17.0.1
java.sql@17.0.1
java.sql.rowset@17.0.1
java.transaction.xa@17.0.1
java.xml@17.0.1
java.xml.crypto@17.0.1
jdk.accessibility@17.0.1
jdk.attach@17.0.1
jdk.charsets@17.0.1
jdk.compiler@17.0.1
jdk.crypto.cryptoki@17.0.1
jdk.crypto.ec@17.0.1
jdk.dynalink@17.0.1
jdk.editpad@17.0.1
jdk.hotspot.agent@17.0.1
jdk.httpserver@17.0.1
jdk.incubator.foreign@17.0.1
jdk.incubator.vector@17.0.1
jdk.internal.ed@17.0.1
jdk.internal.jvmstat@17.0.1
jdk.internal.le@17.0.1
jdk.internal.opt@17.0.1
jdk.internal.vm.ci@17.0.1
jdk.internal.vm.compiler@17.0.1
jdk.internal.vm.compiler.management@17.0.1
```

```
jdk.jartool@17.0.1
jdk.javadoc@17.0.1
jdk.jcmd@17.0.1
jdk.jconsole@17.0.1
jdk.jdeps@17.0.1
jdk.jdi@17.0.1
jdk.jdwp.agent@17.0.1
jdk.jfr@17.0.1
jdk.jlink@17.0.1
jdk.jpackage@17.0.1
jdk.jshell@17.0.1
jdk.jsobject@17.0.1
jdk.jstatd@17.0.1
jdk.localedata@17.0.1
jdk.management@17.0.1
jdk.management.agent@17.0.1
jdk.management.jfr@17.0.1
jdk.naming.dns@17.0.1
jdk.naming.rmi@17.0.1
jdk.net@17.0.1
jdk.nio.mapmode@17.0.1
jdk.random@17.0.1
jdk.sctp@17.0.1
jdk.security.auth@17.0.1
jdk.security.jgss@17.0.1
jdk.unsupported@17.0.1
jdk.unsupported.desktop@17.0.1
jdk.xml.dom@17.0.1
jdk.zipfs@17.0.1
```

这个示例 **Hello World** 应用有非常少的依赖项。您可以使用 jlink 为应用程序创建自定义运行时镜像。使用这些镜像，您只能使用所需的红帽构建 OpenJDK 依赖项来运行应用程序。

2. 创建应用程序模块：

```
$ mkdir sample-module
$ ./jdk-17/bin/jmod create --class-path example/ --main-class sample.HelloWorld --module-version 1.0.0 -p example sample-module/hello.jmod
```

3. 使用所需模块创建自定义 JRE，并为您的应用程序创建自定义应用程序启动程序：

```
$ ./jdk-17/bin/jlink --launcher hello=sample/sample.HelloWorld --module-path sample-module --add-modules sample --output custom-runtime
```

4. 列出生成的自定义 JRE 的模块。
请注意，只有一部分原始红帽构建的 OpenJDK 会保留。

```
$ du -sh custom-runtime
50M custom-runtime/
$ ./custom-runtime/bin/java --list-modules
java.base@17.0.10
java.logging@17.0.10
sample@1.0.0
```

注意

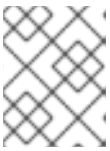
红帽构建的 OpenJDK 将自定义 Java 运行时镜像的大小从 313 M 运行时镜像减小到 50 M 运行时镜像。

5. 使用 **hello** launcher 启动应用程序：

```
$ ./custom-runtime/bin/hello
Jan 14, 2021 12:13:26 PM HelloWorld main
INFO: Hello World!
```

您的示例应用程序生成的 JRE 没有除 **java.base**、**java.logging** 和 **sample** 模块之外的任何其他依赖项。

您可以在 **custom-runtime** 中发布与自定义运行时捆绑的应用程序。此自定义运行时包括您的应用程序。



注意

您必须使用基本红帽构建 OpenJDK 的基本安全更新，为应用程序重建自定义 Java 运行时镜像。

更新于 2024-05-10