Red Hat build of OpenJDK 21

Using jlink to customize Java runtime environment
Red Hat build of OpenJDK 21 Using jlink to customize Java runtime environment
Abstract

Red Hat build of OpenJDK 21 is a Red Hat offering on the Red Hat Enterprise Linux platform. The Using jlink to customize Java runtime images guide provides an overview of Jlink, and explains how to create a customized Java runtime image by using jlink.
# Table of Contents

MAKING OPEN SOURCE MORE INCLUSIVE .................................................. 3

PROVIDING FEEDBACK ON RED HAT DOCUMENTATION ................................. 4

CHAPTER 1. OVERVIEW OF JLINK .......................................................... 5

CHAPTER 2. CREATING A CUSTOM JAVA RUNTIME ENVIRONMENT FOR NON-MODULAR APPLICATIONS .......................................................... 6

CHAPTER 3. CREATING A CUSTOM JAVA RUNTIME ENVIRONMENT FOR MODULAR APPLICATIONS .......................................................... 10
Red Hat is committed to replacing problematic language in our code, documentation, and web properties. We are beginning with these four terms: master, slave, blacklist, and whitelist. Because of the enormity of this endeavor, these changes will be implemented gradually over several upcoming releases. For more details, see our CTO Chris Wright’s message.
PROVIDING FEEDBACK ON RED HAT DOCUMENTATION

We appreciate your feedback on our documentation. To provide feedback, you can highlight the text in a document and add comments.

This section explains how to submit feedback.

Prerequisites

- You are logged in to the Red Hat Customer Portal.
- In the Red Hat Customer Portal, view the document in Multi-page HTML format.

Procedure

To provide your feedback, perform the following steps:

1. Click the Feedback button in the top-right corner of the document to see existing feedback.

   NOTE

   The feedback feature is enabled only in the Multi-page HTML format.

2. Highlight the section of the document where you want to provide feedback.

3. Click the Add Feedback pop-up that appears near the highlighted text. A text box appears in the feedback section on the right side of the page.

4. Enter your feedback in the text box and click Submit. A documentation issue is created.

5. To view the issue, click the issue tracker link in the feedback view.
Jlink is a Java command line tool that is used to generate a custom Java runtime environment (JRE). You can use your customized JRE to run Java applications.

Using jlink, you can create a custom runtime environment that only includes the relevant class file.
CHAPTER 2. CREATING A CUSTOM JAVA RUNTIME ENVIRONMENT FOR NON-MODULAR APPLICATIONS

You can create a custom Java runtime environment from a non-modular application by using the `jlink` tool.

Prerequisites

- Install [Installing Red Hat build of OpenJDK on RHEL using an archive](#).

**NOTE**

For best results, use portable Red Hat binaries as a basis for a Jlink runtime, because these binaries contain bundled libraries.

Procedure

1. Create a simple Hello World application by using the `Logger` class.
   a. Check the base Red Hat build of OpenJDK 21 binary exists in the `jdk-17` folder:

   ```bash
   $ 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. Create a directory for your application:

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

   c. Create `hello-example/sample/HelloWorld.java` file with the following content:

   ```java
   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. Compile your application:

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

   e. Run your application **without** a custom JRE:
The previous example shows the base Red Hat build of OpenJDK requiring 311 MB to run a single class.

f. (Optional) You can inspect the Red Hat build of OpenJDK and see many non-required modules for your application:

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

```bash
$ ./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
```
This sample **Hello World** application has very few dependencies. You can use jlink to create custom runtime images for your application. With these images you can run your application with only the required Red Hat build of OpenJDK dependencies.

2. Determine module dependencies of your application using **jdeps** command:

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

3. Build a custom java runtime image for your application:

```
$ ./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
```

**NOTE**

Red Hat build of OpenJDK reduces the size of your custom Java runtime image from a 313 M runtime image to a 50 M runtime image.
4. You can verify the reduced runtime of your application:

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

The generated JRE with your sample application does not have any other dependencies.

You can distribute your application together with your custom runtime for deployment.

**NOTE**

You must rebuild the custom Java runtime images for your application with every security update of your base Red Hat build of OpenJDK.
CHAPTER 3. CREATING A CUSTOM JAVA RUNTIME ENVIRONMENT FOR MODULAR APPLICATIONS

You can create a custom Java runtime environment from a modular application by using the `jlink` tool.

**Prerequisites**

- Install [Installing Red Hat build of OpenJDK on RHEL using an archive](#).

**NOTE**

For best results, use portable Red Hat binaries as a basis for a Jlink runtime, because these binaries contain bundled libraries.

**Procedure**

1. Create a simple Hello World application by using the `Logger` class.
   a. Check the base Red Hat build of OpenJDK 21 binary exists in the `jdk-17` folder:

   ```bash
   $ 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. Create a directory for your application:

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

   c. Create `hello-example/sample/HelloWorld.java` file with the following content:

   ```java
   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. Create a file called `hello-example/module-info.java` and include the following code in the file:

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

   e. Compile your application:
$ ./jdk-17/bin/javac -d example $(find hello-example -name ".java")

f. Run your application without a custom JRE:

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

The previous example shows the base Red Hat build of OpenJDK requiring 311 MB to run a single class.

g. (Optional) You can inspect the Red Hat build of OpenJDK and see many non-required modules for your application:

$ 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.charset@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
This sample Hello World application has very few dependencies. You can use jlink to create custom runtime images for your application. With these images you can run your application with only the required Red Hat build of OpenJDK dependencies.

2. Create your application module:

   $ 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. Create a custom JRE with the required modules and a custom application launcher for your application:

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

4. List the modules of the produced custom JRE.
   Observe that only a fraction of the original Red Hat build of OpenJDK remains.

   $ 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

NOTE

Red Hat build of OpenJDK reduces the size of your custom Java runtime image from a 313 M runtime image to a 50 M runtime image.

5. Launch the application using the **hello** launcher:

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

The generated JRE with your sample application does not have any other dependencies besides **java.base**, **java.logging**, and **sample** module.

You can distribute your application that is bundled with the custom runtime in **custom-runtime**. This custom runtime includes your application.

NOTE

You must rebuild the custom Java runtime images for your application with every security update of your base Red Hat build of OpenJDK.

*Revised on 2023-11-14 21:12:47 UTC*