Red Hat Enterprise Linux 8

Installing, updating, and configuring Java on RHEL 8

An introduction to Java application development in RHEL 8

Last Updated: 2020-04-28

Red Hat Enterprise Linux 8 Installing, updating, and configuring Java on RHEL 8

An introduction to Java application development in RHEL 8

Legal Notice

Abstract

This document describes the different features and utilities that make Red Hat Enterprise Linux 8 an ideal enterprise platform for Java application development.

Table of Contents

PROVIDING FEEDBACK ON RED HAT DOCUMENTATION

We appreciate your input on our documentation. Please let us know how we could make it better. To do so:

- For simple comments on specific passages:
  1. Make sure you are viewing the documentation in the *Multi-page HTML* format. In addition, ensure you see the **Feedback** button in the upper right corner of the document.
  2. Use your mouse cursor to highlight the part of text that you want to comment on.
  3. Click the **Add Feedback** pop-up that appears below the highlighted text.
  4. Follow the displayed instructions.
- For submitting more complex feedback, create a Bugzilla ticket:
  1. Go to the Bugzilla website.
  2. As the Component, use **Documentation**.
  3. Fill in the **Description** field with your suggestion for improvement. Include a link to the relevant part(s) of documentation.
  4. Click **Submit Bug**.

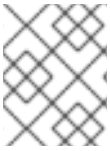
CHAPTER 1. INSTALLING JAVA ON RHEL 8

Java is an environment for developing and running a wide range of platform-agnostic applications, from mobile applications to desktop and web applications and enterprise systems. Java is a general-purpose programming language. Red Hat provides an open-source implementation of the Java Platform SE (Standard Edition) called OpenJDK. It is included in the RHEL repository.

Applications are developed using the JDK (Java Development Kit). Applications are run on a JVM (Java Virtual Machine), which is included in the JRE (Java Runtime Environment) and the JDK. There is also a headless version of Java which has the smallest footprint and does not include the libraries needed for a user interface. The headless version is packaged in the **headless subpackage**.

NOTE

If you are unsure whether you need the JRE or the JDK, it is recommended that you install the JDK.



NOTE

To get short-term supported Java versions, use Extra Packages for Enterprise Linux (EPEL). For information see EPEL. The name of package is **java-latest-openjdk**.

The following sections provide instructions for installing Java on RHEL.

1.1. INSTALLING A JRE ON RHEL USING YUM

You can install an OpenJDK Java Runtime Environment (JRE) using the system package manager, **yum**. This requires root privileges.

The JRE is part of the OpenJDK.

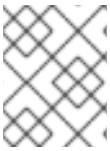
Procedure

1. Run the **yum** command, specifying the package you want to install:
   - For JRE 8: **java-1.8.0-openjdk**
     $ sudo yum install java-1.8.0-openjdk
   - For JRE 11: **java-11-openjdk**
     $ sudo yum install java-11-openjdk
2. Check that the installation works:

```
$ java -version
openjdk version "1.8.0_242"
OpenJDK Runtime Environment (build 1.8.0_242-b08)
OpenJDK 64-Bit Server VM (build 25.242-b08, mixed mode)
```

1.2. INSTALLING A JRE ON RHEL USING AN ARCHIVE PORTABLE BUNDLE

You can install an OpenJDK Java Runtime Environment (JRE) using the ZIP bundle. This is useful if the Java administrator does not have root privileges.



NOTE

It is recommended that you create a parent directory to contain your JREs and create a symbolic link to the latest JRE using a generic path. This eases upgrades to later versions.

Procedure

1. Download the latest version of the JRE ZIP bundle for Linux .
2. Extract the contents of the ZIP bundle to a directory of your choice:

```
$ mkdir ~/jres
$ cd ~/jres
$ tar -xf java-1.8.0-openjdk-1.8.0.242.b08-1.static.jre.openjdkportable.x86_64.tar.xz
```

3. Create a generic path by using symbolic links to your JRE for easier upgrades:
$ ln -s ~/jres/java-1.8.0-openjdk-1.8.0.242.b08-1.static.jre.openjdkportable.x86_64 ~/jres/java-8
4. Configure the **JAVA_HOME** environment variable:

```
$ export JAVA_HOME=~/jres/java-8
```

5. Verify that **JAVA_HOME** environment variable is set correctly:

```
$ printenv | grep JAVA_HOME
JAVA_HOME=~/jres/java-8
```

NOTE

When installed using this method, Java will only be available for the current user.

6. Add the **bin** directory of the generic JRE path to the **PATH** environment variable:

```
$ export PATH="$JAVA_HOME/bin:$PATH"
```

7. Verify that **java -version** works without supplying the full path:

```
$ java -version
openjdk version "1.8.0_242"
OpenJDK Runtime Environment (build 1.8.0_242-b08)
OpenJDK 64-Bit Server VM (build 25.242-b08, mixed mode)
```

NOTE

You can ensure that **JAVA_HOME** environment variable persists for the current user by exporting the environment variable in ~/**.bashrc**.

1.3. INSTALLING A JDK ON RHEL 8 USING YUM

You can install an OpenJDK (Open Java Development Kit) using the system package manager, **yum**. This requires root privileges.

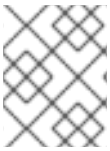
Procedure

1. Run the **yum** command, specifying the package you want to install:
   - For JDK 8: **java-1.8.0-openjdk-devel**
     $ sudo yum install java-1.8.0-openjdk-devel
   - For JDK 11: **java-11-openjdk-devel**
     $ sudo yum install java-11-openjdk-devel
2. Check that the installation works:

```
$ javac -version
javac 1.8.0_242
```

1.4. INSTALLING A JDK ON RHEL 8 USING AN ARCHIVE PORTABLE BUNDLE

You can install an OpenJDK (Open Java Development Kit) using the ZIP bundle. This is useful if the Java administrator does not have root privileges.



NOTE

It is recommended that you create a parent directory to contain your JDKs and create a symbolic link to the latest JDK using a generic path. This eases upgrades to later versions.

Procedure

1. Download the latest version of the JDK ZIP bundle for Linux .
2. Extract the contents of the ZIP bundle to a directory of your choice:

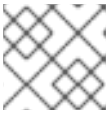
```
$ mkdir ~/jdks
$ cd ~/jdks
$ tar -xf java-1.8.0-openjdk-1.8.0.242.b08-1.static.jdk.openjdkportable.x86_64.tar.xz
```

3. Create a generic path by using symbolic links to your JRE for easier upgrades:
   $ ln -s ~/jdks/java-1.8.0-openjdk-1.8.0.242.b08-1.static.jdk.openjdkportable.x86_64 ~/jdks/java-8
4. Configure the **JAVA_HOME** environment variable:

```
$ export JAVA_HOME=~/jdks/java-8
```

5. Verify that **JAVA_HOME** environment variable is set correctly:

```
$ printenv | grep JAVA_HOME
JAVA_HOME=~/jdks/java-8
```



NOTE

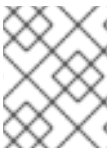
When installed using this method, Java will only be available for the current user.

6. Add the **bin** directory of the generic JDK path to the **PATH** environment variable:

```
$ export PATH="$JAVA_HOME/bin:$PATH"
```

7. Verify that **javac -version** works without supplying the full path:

```
$ javac -version
javac "1.8.0_242"
```



NOTE

You can ensure that **JAVA_HOME** environment variable persists for the current user by exporting the environment variable in ~/**.bashrc**.

1.5. INSTALLING MULTIPLE MAJOR VERSIONS OF OPENJDK ON RHEL USING YUM

You can install multiple versions of openJDK using the system package manager, **yum**. This requires root privileges.

Prerequisites

- A Red Hat Subscription Manager (RHSM) account with an active subscription that provides access to a repository that provides the OpenJDK versions you want to install.

Procedure

1. Run the **yum** command, specifying the package you want to install:
   - For JRE 8: **java-1.8.0-openjdk**
     $ sudo yum install java-1.8.0-openjdk
   - For JDK 11: **java-11-openjdk**
     $ sudo yum install java-11-openjdk
2. Check that the installation works:

```
$ java -version
openjdk version "1.8.0_242"
OpenJDK Runtime Environment (build 1.8.0_242-b08)
OpenJDK 64-Bit Server VM (build 25.242-b08, mixed mode)
```

You can configure the default Java version to use by using **java --alternatives**. For more information see Selecting a system-wide java version .

1.6. INSTALLING MULTIPLE MAJOR VERSIONS OF OPENJDK ON RHEL USING AN ARCHIVE PORTABLE BUNDLE

You can install multiple major versions of OpenJDK by using the same procedures found in Installing a JRE on RHEL using an archive portable bundle or Installing a JDK on RHEL 8 using an archive portable bundle using multiple major versions.

NOTE

For instructions how to configure the default Java version for the system, see Selecting a system-wide zip-bundle java version.

1.7. INSTALLING MULTIPLE MINOR VERSIONS OF OPENJDK ON RHEL USING YUM

You can install multiple minor versions of OpenJDK on RHEL. This is done by preventing the installed minor versions from being updated.

Procedure

- Add the **installonlypkgs** option in **/etc/yum.conf** to specify the Java packages that **yum** can install but not update.
  installonlypkgs=java-*<version>*--openjdk,java-*<version>*--openjdk-headless,java-*<version>*--openjdk-devel

  Updates will install new packages while leaving the old versions on the system:

  ```
  $ rpm -qa | grep java-1.8.0-openjdk
  java-1.8.0-openjdk-1.8.0.242.b08-0.el8_1.x86_64
  java-1.8.0-openjdk-headless-1.8.0.242.b08-0.el8_1.x86_64
  ```

- The different minor versions of OpenJDK can be found in the **/usr/lib/jvm/*<minor version>*** files.
  For example, the following shows part of **/usr/lib/jvm/java-1.8.0-openjdk-1.8.0**:

  ```
  $ /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.242.b08-0.el8_1.x86_64/bin/java -version
  openjdk version "1.8.0_242"
  OpenJDK Runtime Environment (build 1.8.0_242-b08)
  OpenJDK 64-Bit Server VM (build 25.242-b08, mixed mode)

  $ /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.172-7.b11.el7.x86_64/bin/java -version
  openjdk version "1.8.0_172"
  OpenJDK Runtime Environment (build 1.8.0_172-b11)
  OpenJDK 64-Bit Server VM (build 25.172-b11, mixed mode)
  ```

You can choose system-wide using version by following Selecting a system-wide java version .

1.8. INSTALLING MULTIPLE MINOR VERSIONS OF OPENJDK ON RHEL USING AN ARCHIVE PORTABLE BUNDLE

Installing multiple minor versions is the same as Installing a JRE on RHEL using an archive portable bundle or Installing a JDK on RHEL 8 using an archive portable bundle using multiple minor versions.

NOTE

For instructions how to choose a default minor version for the system, see Selecting a system-wide zip-bundle java version.

CHAPTER 2. UPDATING JAVA ON RHEL 8

The following sections provide instructions for updating Java on RHEL.

2.1. UPDATING JAVA ON RHEL 8 USING YUM

The installed Java packages can be updated using the **yum** system package manager. This requires root privileges.

Procedure

1. Check the current Java version:
   $ sudo yum list installed "java*"

   A list of installed Java packages appears.

   ```
   Installed Packages
   java-1.8.0-openjdk.x86_64              1:1.8.0.131.b08-0.fc31       @updates
   java-1.8.0-openjdk-headless.x86_64     1:1.8.0.131.b08-0.fc31       @updates
   ```

2. Update a specific package. For example:
   $ sudo yum update java-1.8.0-openjdk

3. Check the current Java versions:

   ```
   # java -version
   openjdk version "1.8.0_242"
   OpenJDK Runtime Environment (build 1.8.0_242-b08)
   OpenJDK 64-Bit Server VM (build 25.242-b08, mixed mode)
   ```

2.2. UPDATING JAVA ON RHEL 8 USING AN ARCHIVE

You can update Java using the ZIP bundle. This is useful if the Java administrator does not have root privileges.

Prerequisites

- Know the generic path pointing to your JDK or JRE installation. For example, **~/jdks/java-11**

Procedure

1. Remove the existing symbolic link of the generic path to your JDK or JRE.
   For example:

   $ unlink ~/jdks/java-11

2. Install the latest version of the JDK or JRE in your installation location.

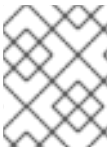
   - For instructions on installing a JRE, see Installing a JRE on RHEL 8 .
   - For instructions on installing a JDK, see Installing a JDK on RHEL 8 .

CHAPTER 3. CONFIGURING JAVA ON RHEL 8

The following sections provide instructions for configuring Java on RHEL.

3.1. INTERACTIVELY SELECTING A SYSTEM-WIDE JAVA VERSION ON RHEL

If you have multiple versions of Java installed on RHEL, you can interactively select the default Java version to use system-wide. This requires root privileges.



NOTE

If you do not have root privileges, you can select a Java version by configuring the **JAVA_HOME** environment variable.

Prerequisites

- You must have root privileges on the system.
- Multiple versions of Java were installed using the **yum** package manager.

Procedure

1. View the Java versions installed on the system.
   $ yum list installed "java*"

   A list of installed Java packages appears.

```
Installed Packages
java-1.8.0-openjdk.x86_64                1:1.8.0.242.b08-0.el8_1                @rhel-8-
appstream-rpms
java-1.8.0-openjdk-headless.x86_64       1:1.8.0.242.b08-0.el8_1                @rhel-8-
appstream-rpms
java-11-openjdk.x86_64                   1:11.0.6.10-0.el8_1                    @rhel-8-
appstream-rpms
java-11-openjdk-headless.x86_64          1:11.0.6.10-0.el8_1                    @rhel-8-
appstream-rpms
javapackages-filesystem.noarch           5.3.0-1.module+el8+2447+6f56d9a6       @rhel-
8-appstream-rpms
```

2. Display the Java versions that can be used for a specific Java command and select the one to use:

```
$ sudo alternatives --config java
There are 2 programs which provide 'java'.

  Selection    Command
-----------------------------------------------
*+ 1           java-1.8.0-openjdk.x86_64 (/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.242.b08-
0.el8_1.x86_64/jre/bin/java)
   2           java-11-openjdk.x86_64 (/usr/lib/jvm/java-11-openjdk-11.0.6.10-
0.el8_1.x86_64/bin/java)
```

```
Enter to keep the current selection[+], or type selection number: 1
```

- The current system-wide Java version is marked with an asterisk.
- The current Java version for the specified Java command is marked with a plus sign.

3. Press **Enter** to keep the current selection or enter the **Selection** number of the Java version you want to select followed by the **Enter** key.
The default Java version for the system is the selected version.

4. Verify that the selected binary is selected.

```
$ java -version
openjdk version "1.8.0_242"
OpenJDK Runtime Environment (build 1.8.0_242-b08)
OpenJDK 64-Bit Server VM (build 25.242-b08, mixed mode)
```

NOTE

This procedure configures the **java** command. Then **javac** command can be set up in a similar way, but it operates independently.

If you have a JDK installed, **alternatives** provides more possible selections. In particular, the **javac** master alternative switches many binaries provided by the **-devel** sub-package. Even if you have JDK installed, java (and other JRE masters) and javac (and other JDK masters) still operate separately, so you can have different selections for JRE and JDK.

The **alternatives --config java** command affects the **jre** and its associated slaves. If you want to change the **JDK**, use the **javac alternatives** command. The **--config javac** utility configures the **SDK** and related slaves. To see all possible masters, use **alternatives --list** and check all of the **java**,**javac**, **jre**, and **sdk** masters.

3.2. NON-INTERACTIVELY SELECTING A SYSTEM-WIDE JAVA VERSION ON RHEL

If you have multiple versions of Java installed on RHEL, you can select the default Java version to use system-wide in a non-interactive way. This is useful for administrators who have root privileges on a Red Hat Enterprise Linux system and need to switch the default Java on many systems in an automated way.

NOTE

If you do not have root privileges, you can select a Java version by configuring the **JAVA_HOME** environment variable.

Prerequisites

- You must have root privileges on the system.
- Multiple versions of Java were installed using the **yum** package manager.

Procedure

1. Select the major Java version to switch to. For example, for Java 11, use java-11-openjdk.

```
# PKG_NAME=java-11-openjdk`
# JAVA_TO_SELECT=$(alternatives --display java | grep "family $PKG_NAME" | cut -d' ' -f1)`
# alternatives --set java $JAVA_TO_SELECT`
```

2. Verify that the active Java version is the one you specified.

```
$ java -version
openjdk version "11.0.3" 2019-04-16 LTS
OpenJDK Runtime Environment 18.9 (build 11.0.3+7-LTS)
OpenJDK 64-Bit Server VM 18.9 (build 11.0.3+7-LTS, mixed mode, sharing)
```

NOTE

For Java 8, set **PKG_NAME** to **java-1.8.0-openjdk**.

NOTE

A similar approach can be followed for **javac**.

3.3. SELECTING AN INSTALLED JAVA VERSION FOR A SPECIFIC APPLICATION

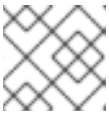
Some applications require a specific Java version to run. If multiple versions of Java are installed on the system using the **yum** package manager or portable bundle, you can select a Java version for each application where necessary by setting the value of the **JAVA_HOME** environment variable or using a wrapper script.

Prerequisites

- Multiple versions of Java installed on the machine.
- Ensure that the application you want to run is installed.

Procedure

1. Set the **JAVA_HOME** environment variable. For example, if openjdk-11 was installed using **yum**:
$ JAVA_HOME=/usr/lib/jvm/java-11-openjdk



NOTE

The symbolic link **java-11-openjdk** is controlled by the **alternatives** command.

2. Do one of the following:
   - Launch the application using the default, system-wide configuration.

```
$ mvn --version
Apache Maven 3.5.4 (Red Hat 3.5.4-5)
Maven home: /usr/share/maven
Java version: 1.8.0_242, vendor: Oracle Corporation, runtime: /usr/lib/jvm/java-1.8.0-
```

```
openjdk-1.8.0.242.b08-0.el8_1.x86_64/jre
Default locale: en_US, platform encoding: UTF-8
OS name: "linux", version: "4.18.0-147.3.1.el8_1.x86_64", arch: "amd64", family: "unix"
```

- Launch the application specifying the **JAVA_HOME** variable:

```
$ JAVA_HOME=/usr/lib/jvm/java-11-openjdk mvn --version

Apache Maven 3.5.4 (Red Hat 3.5.4-5)
Maven home: /usr/share/maven
Java version: 11.0.5, vendor: Oracle Corporation, runtime: /usr/lib/jvm/java-11-openjdk-11.0.5.10-0.el8_1.x86_64
Default locale: en_US, platform encoding: UTF-8
OS name: "linux", version: "5.4.12-200.el8_1.x86_64", arch: "amd64", family: "unix"
```

3.4. SELECTING A SYSTEM-WIDE ZIP-BUNDLE JAVA VERSION

If you have multiple version of Java installed on RHEL 8 using **ZIP** bundles, you can select a specific Java version to use system-wide.

Prerequisites

- Know the locations of the Java versions installed using **ZIP** bundles.

Procedure

To specify the Java version to use for a single session:

1. Configure **JAVA_HOME** with the path to the Java version you want used system-wide.
   $ export JAVA_HOME=/opt/jdk/jdk-11.0.3

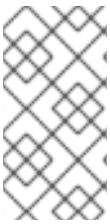
2. Add **$JAVA_HOME/bin** to the **PATH** environment variable.
   $ export PATH="$JAVA_HOME/bin:$PATH"

To specify the Java version to use permanently for a single user, add these commands into **~/.bashrc**:

```
export JAVA_HOME=/opt/jdk/jdk-11.0.3
export PATH="$JAVA_HOME/bin:$PATH"
```

To specify the Java version to use permanently for all users, add these commands into **/etc/bashrc**:

```
export JAVA_HOME=/opt/jdk/jdk-11.0.3
export PATH="$JAVA_HOME/bin:$PATH"
```



NOTE

Be aware of the exact meaning of **JAVA_HOME**. For more information, see Changes/Decouple system java setting from java command setting . If you do not want to redefine **JAVA_HOME**, add only the PATH command to **bashrc**, specifying the path to the Java binary. For example, **export PATH="/opt/jdk/jdk-11.0.3/bin:$PATH"**.

3.5. CONFIGURING THE JAVA_HOME ENVIRONMENT VARIABLE ON RHEL

Some applications require you to set the **JAVA_HOME** environment variable so that they can find the Java installation.

Prerequisites

- Know where Java is installed on your system. For example, /**opt/jdk/11**.

Procedure

1. Set the value of **JAVA_HOME**.
   $ export JAVA_HOME=/opt/jdk/11
2. Verify that **JAVA_HOME** is set correctly.

```
$ printenv | grep JAVA_HOME
JAVA_HOME=/opt/jdk/11
```

NOTE

You can make the value of **JAVA_HOME** persistent by exporting the environment variable in ~/**.bashrc** for single users or /**etc/bashrc** for system wide settings.

NOTE

Be aware of the exact meaning of **JAVA_HOME**. For more information, see Changes/Decouple system java setting from java command setting .

3.6. CONFIGURING THE HEAP SIZE FOR A JAVA APPLICATION ON RHEL

OpenJDK can be configured to use a customized heap size.

Procedure

- Add the maximum heap size option to the Java command when running your application
  For example to set the maximum heap size to 100 megabytes use the **-Xmx100m** option.

  For OpenJDK 8:

  $ java -Xmx100m *<your-application>*

  For OpenJDK 11 it is recommended to use a new way:

  $ java -Xmn100m *<your-application>*

  For more information about the **Xmx** option, see **-Xmxsize** in Java documentation.