Alpaquita Linux
Building Java applications with Cloud Native Buildpacks

Alpaquita Linux
Revision 1.0
November 2024

bell/soft

THIS INFORMATION MAY CHANGE WITHOUT NOTICE. TO THE EXTENT PERMITTED BY APPLICABLE LAW, BELLSOFT PROVIDES THIS DOCUMENTATION “AS IS” WITHOUT WARRANTY OF ANY KIND, INCLUDING WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NONINFRINGEMENT. IN NO EVENT WILL BELLSOFT BE LIABLE TO YOU OR ANY THIRD PARTY FOR ANY LOSS OR DAMAGE, DIRECT OR INDIRECT, FROM THE USE OF THIS DOCUMENTATION, INCLUDING WITHOUT LIMITATION, LOST PROFITS, LOST INVESTMENT, BUSINESS INTERRUPTION, GOODWILL, OR LOST DATA, EVEN IF BELLSOFT IS EXPRESSLY ADVISED IN ADVANCE OF THE POSSIBILITY OF SUCH LOSS OR DAMAGE.

The use of any software product referenced in this document is governed by the applicable license agreement, which is not modified in any way by the terms of this notice.

Contents

1. Introduction

BellSoft provides Cloud Native Buildpacks images that contain the latest versions of Liberica JDK Lite and Liberica NIK products specifically built and optimized for Alpaquita Linux. Alpaquita Linux is a full-featured operating system shipped in two libc variants based on two different libc implementations, namely `glibc` and `musl`. BellSoft provides images for both `glibc` and `musl` libc libraries.

Using Alpaquita Linux (musl) as a target image helps you create lightweight Docker images. For example, the size of a Docker image for a basic Spring Boot application running Java 21, if using standard `paketobuildpacks/builder-jammy-tiny` is about 250 MB, while using `bellsoft/buildpacks.builder:musl` saves about 140 MB.

Cloud Native Buildpacks

Cloud Native Buildpacks (CNBs) transform your application source code into container images that can run on any cloud.

Liberica JDK Lite

Liberica JDK Lite is a JDK optimized for cloud instances with a minimal footprint. It is a full-fledged, Java SE-compliant runtime but much smaller than any standard Java distribution. Liberica JDK Lite has a higher compression ratio for modules than a classic JDK, thus reducing static footprint. For more information, see Liberica JDK Lite in the *Choosing Liberica JDK flavor* document.

Liberica NIK

Liberica Native Image Kit is a utility capable of converting your JVM-based application into a fully compiled native executable ahead-of-time under the closed-world assumption with an almost instant startup time. It optimizes resource consumption, minimizes the static footprint, and works on various platforms, including lightweight musl-based Alpaquita Linux.

2. Image flavors

When building images from BellSoft buildpacks, you can choose one of the latest JDK releases of 8, 11, 17 (default), and 21 JDK versions; 22 and 23 NIK versions as well as Alpaquita Linux based on glibc and musl libraries. For the latest product release versions, see the following links:

- Liberica JDK Download Center.
- Liberica Native Image Kit Download Center.
- Alpaquita Linux Download Center.

Choosing libc library

If the size of the resulting image is important and the final application does not have native components that require `glibc`, then choose `musl`. If you need higher performance with a slight increase in size of the final image, it makes sense to consider using the `glibc` library. In addition, there might be other considerations when choosing C runtime, such as licensing.

3. Building a Java application with Alpaquita Linux

To build a Java application for Alpaquita Linux with the default JDK version, run one of the following commands within the root of your project. Make sure you choose the `bellsoft` builder and specify the target C library.

> **Note:**
>
> A builder includes the buildpacks that will be used as well as the environment for building your app. For more information about getting started with buildpacks, see Build your very first application with buildpacks.

- Building a Java application with Alpaquita Linux (glibc)

  ```
  pack build demo-app --builder bellsoft/buildpacks.builder:glibc --path .
  ```

- Building a Java application with Alpaquita Linux (musl)

  ```
  pack build demo-app --builder bellsoft/buildpacks.builder:musl --path .
  ```

Where `bellsoft/buildpacks.builder:<glibc/musl>` is the BellSoft builder producing an Alpaquita Linux image with the specified C library. You can also specify BellSoft builder as the default one as follows:

```
pack config default-builder bellsoft/buildpacks.builder:musl
```

If you want to install a specific JVM feature version, use the `BP_JVM_VERSION` environment variable that accepts the following JDK version values: 8, 11, 17, 21 (default is 17). For example:

```
pack build demo-app --builder bellsoft/buildpacks.builder:glibc --path . -env
BP_JVM_VERSION=21
```

To inspect the resulting image, check the stack, buildpacks participated in the build, etc. Run the following command:

```
pack inspect demo-app
```

You can also check the Linux distro inside your Docker image as follows:

```
docker run --rm -it --entrypoint /bin/cat demo-app /etc/os-release
```

Then you can check the image size as follows:

```
docker image ls demo-app
```

To run the application, use the following Docker command:

```
docker run --rm -it demo-app
```

4. Optimizing image with JLink

The JRE inside the resulting image can be optimized in size by using the `jlink` tool. To enable `jlink`, set the environment variable `BP_JVM_JLINK_ENABLED` to `true` (default is `false`). In this case, jlink generates a custom JRE with the following default options: `--no-man-pages`, `--no-header-files`, `--strip-debug`, `--compress=1`. To change the options, use `BP_JVM_JLINK_ARGS` environment variable. Check the jlink documentation for further information.

For example, the following command creates a new application image on Alpaquita Linux with a custom JRE 21:

```
pack build demo-app --builder bellsoft/buildpacks.builder:musl --path . -env
BP_JVM_VERSION=21 -env BP_JVM_JLINK_ENABLED=true
```

The resulting image is usually smaller than the one with the default JRE.

5. Building native image

To build the native image application, set the `BP_NATIVE_IMAGE` environment variable to `true` as in the following example:

```
pack -v build demo-native-app --builder bellsoft/buildpacks.builder:musl --path
. --env BP_NATIVE_IMAGE=true --env BP_JVM_VERSION=21
```

You can check the size of the resulting image as follows:

```
docker image ls demo-native-app
```

To run the application, use the following Docker command:

```
docker run --rm -it demo-native-app
```

6. Configuration options

The following table lists pack builder configuration options for BellSoft Liberica JDK Lite and Native Image Kit Cloud Native Buildpacks ver. 1.1.0.

Variable	Default value	Description
Build configuration options		
$BP_JVM_JLINK_ARGS	• `--no-man-pages` • `--no-header-files` • `--strip-debug` • `--compress=1`	Configures custom jlink arguments (`--output` must be omitted).
$BP_JVM_JLINK_ENABLED	`false`	Enables the jlink tool to generate custom JRE.
$BP_JVM_TYPE	JRE	Specifies the JVM type - JDK or JRE.
$BP_JVM_VERSION	17	Specifies a Java version.
Launch configuration options		
$BPL_DEBUG_ENABLED	false	Enables Java remote debugging support.
$BPL_DEBUG_PORT	8000	Specifies the remote debugging port.
$BPL_DEBUG_SUSPEND	false	Specifies whether to suspend execution until a debugger has attached.

Variable	Default value	Description
$BPL_HEAP_DUMP_PATH	N/A	Specifies the path to write heap dumps on error.
$BPL_JAVA_NMT_ENABLED	true	Enables Java Native Memory Tracking (NMT).
$BPL_JAVA_NMT_LEVEL	summary	Configures the level of NMT, `summary` or `detail`.
$BPL_JFR_ARGS	N/A	Configures custom Java Flight Recording (JFR) arguments.
$BPL_JFR_ENABLED	false	Enables Java Flight Recorder (JFR).
$BPL_JMX_ENABLED	false	Enables Java Management Extensions (JMX).
$BPL_JMX_PORT	5000	Specifies the JMX port.
$BPL_JVM_HEAD_ROOM	0	Configures the headroom in memory calculation.
$BPL_JVM_LOADED_CLASS_COUNT	35% of classes	Sets the number of loaded classes in memory calculation.
$BPL_JVM_THREAD_COUNT	250	Sets the number of threads in memory calculation.
$JAVA_TOOL_OPTIONS	N/A	Specifies the JVM launch flags.

Alpaquita Linux

Building Java applications with Cloud Native Buildpacks