BUILD NATIVE JAVA APPS WITH GRAALVM

FU CHENG

Build Native Java Apps with GraalVM

Java Going Native in Cloud-native era

Fu Cheng

This book is available at
https://leanpub.com/build-native-java-apps-graalvm

This version was published on 2025-12-08

Leanpub

This is a Leanpub book. Leanpub empowers authors and publishers with the Lean Publishing process. Lean Publishing is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.

Also By Fu Cheng

Exploring Java 25
Text-to-SQL, Spring AI Implementation with RAG
Understanding Java Virtual Threads
From Java 21 to Java 25
Build AI Applications with Spring AI
From Java 17 to Java 21
From Java 11 to Java 17
ES6 Generators
A Practical Guide for Java 8 Lambdas and Streams
Lodash 4 Cookbook
JUnit 5 Cookbook

Contents

Introduction

Going native is a necessary step for Java apps in cloud-native era.

Introduction to Native Java Apps

Before building native Java apps, we need to understand why we need to build them.

Traditional Java Apps

Write once, run anywhere is a famous slogan of Java programming language. Java source code is compiled to platform-neutral bytecode first, then bytecode is executed on Java Virtual Machine (JVM).

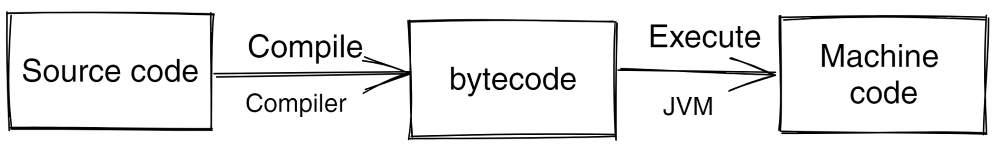

Figure 1. Java source code to bytecode

JVM provides runtime components to execute Java programs and encapsulates implementation details for different platforms. The same bytecode can be executed on different platforms without any changes. This is one of the important reasons why Java is so popular in enterprise applications development.

In the traditional deployment mode of Java apps, JDK/JRE is installed on top of the operating systems running on physical or virtual machines. A Java app is started by launching a JVM process using the `java` command. Each Java app has its own JVM process.

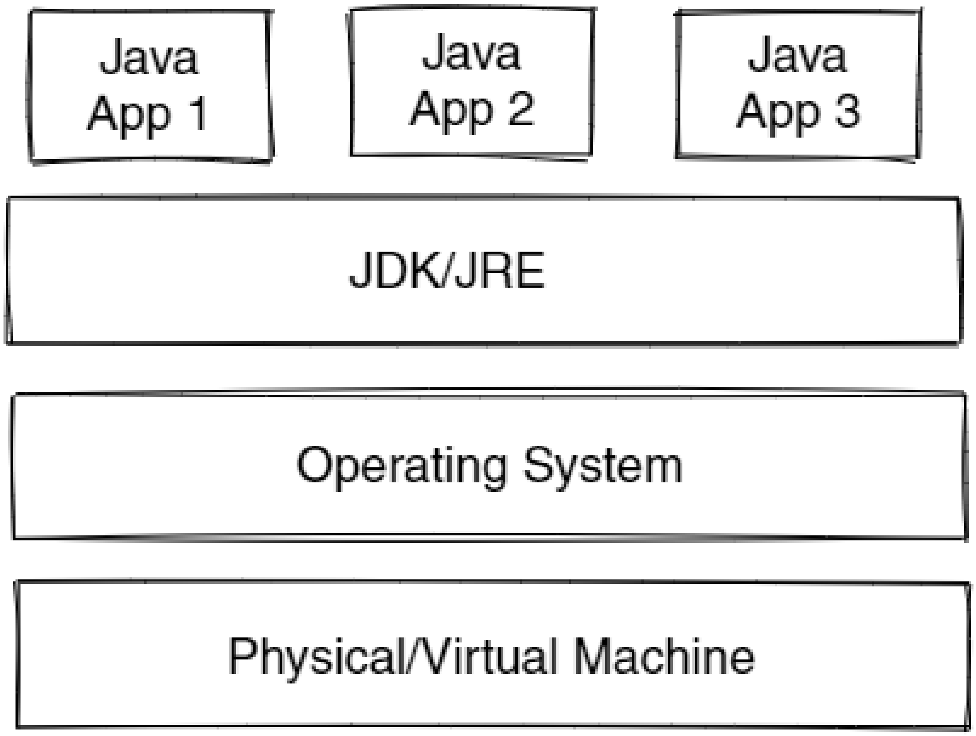

Figure 2. Java traditional deployment model

Installed JDKs can be used by many Java apps. A single installation is usually enough when all Java apps support the same Java version.

Cloud-Native Java Apps

Cloud-native apps are packages as OCI/Docker container images. Container images are immutable and self-contained. After the container image is built, the execution platform of the app has already been frozen. This means that cloud-native apps can always target a certain platform. Platform independence provided by JVM is actually unnecessary.

When Java apps are packaged as container images, a compatible JDK or JRE is required to be bundled with the app. When running the container image, a JVM process is launched to run the app.

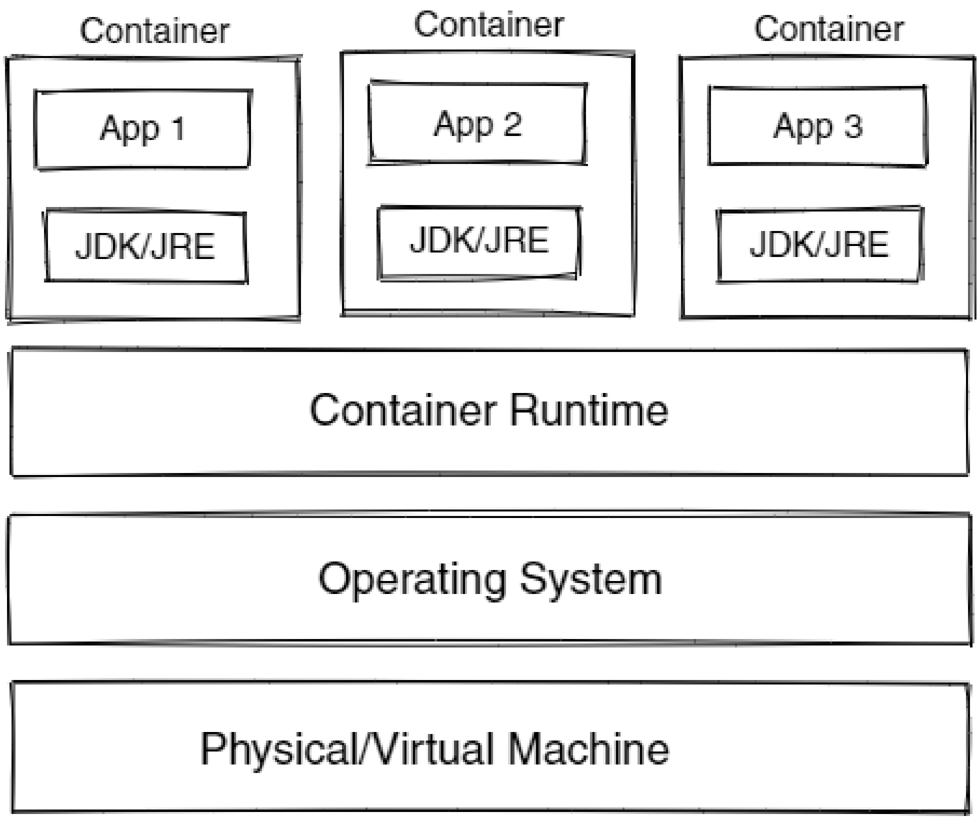

Figure 3. Java cloud deployment model

The key issue is that **Java bytecode is not executable on its own**. Bytecode is a platform-neutral intermediate representation (IR), which requires a JVM to interpret in the runtime.

Even a simple Java application requires a complete JVM to run it. This will dramatically increase the size of container images.

Java Platform Module System

With the introduction of Java Platform Module System (JPMS) in Java 9, it's now possible to customize the JDK with `jlink` to only include required modules. This can reduce the size of JDK, but requires extra efforts to implement.

Comparing to other programming languages, Java is less promising due to its limitations on container image size, app startup time, and memory consumption.

Container Image Size

Let's use the simplest "Hello World" program as the example to demonstrate container image size. Java, Go, and Rust are used to implement this program. These programs are all very simple to write.

The code below is the Java version of "Hello World" program.

Figure 4. Hello World in Java

```java
public class Main {
  public static void main(String[] args) {
    System.out.println("Hello, world!");
  }
}
```

The code below is the Go version of "Hello World" program.

Figure 5. Hello World in Go

```go
package main

import "fmt"

func main() {
  fmt.Println("Hello, world!")
}
```

The code below is the Rust version of "Hello World" program.

Figure 6. Hello World in Rust

```rust
fn main() {
  println!("Hello, world!");
}
```

When creating container images for this app, Docker multi-stage builds are used to create container images for different programming languages. For Java, both JVM and native versions are provided.

The container image build process has two stages:

1. The first stage builds the app. For Java, the output is class files; for Go and Rust, the output is a native executable binary file.
2. The second stage combines the output from the first stage with a runtime environment. For Java, the runtime environment includes the JRE; for Go and Rust, the runtime environment is just the operating system.

Below is the Dockerfile of a Java application running using JVM. This Dockerfile uses Eclipse Temurin 25 to build the application.

Figure 7. Dockerfile for Java (JVM)

```
FROM eclipse-temurin:25-jdk AS builder

RUN mkdir /build && mkdir /build/source && mkdir /build/target

COPY Main.java ./build/source

RUN javac /build/source/Main.java -d /build/target

#############

FROM eclipse-temurin:25-jre-alpine

RUN mkdir /app

COPY --from=builder /build/target/*.class /app

ENTRYPOINT [ "java", "-cp", "/app", "Main" ]
```

Below is the Dockerfile of a Java application running as a native executable.

Figure 8. Dockerfile for Java (Native)

```
FROM ghcr.io/graalvm/native-image-community:25.0.1 AS builder

RUN mkdir /build && mkdir /build/source && mkdir /build/target

WORKDIR /build

COPY Main.java ./source

RUN javac ./source/Main.java -d ./target

RUN native-image -cp ./target -H:+UnlockExperimentalVMOptions -H:Name=hellowo\
rld -H:Class=Main

#############
```

```

FROM debian:bookworm-slim

WORKDIR /

COPY --from=builder /build/helloworld /helloworld

ENTRYPOINT ["/helloworld"]
```

Below is the Dockerfile of a Go application.

Figure 9. Dockerfile for Go

```
FROM golang:1.17-buster AS builder

WORKDIR /app

COPY go.mod ./
RUN go mod download

COPY *.go ./

RUN go build -o /helloWorld

#############

FROM gcr.io/distroless/base-debian10

WORKDIR /

COPY --from=builder /helloWorld /helloWorld

USER nonroot:nonroot

ENTRYPOINT ["/helloWorld"]
```

Below is the Dockerfile of a Rust application.

Figure 10. Dockerfile for Rust

```
FROM rust AS builder

WORKDIR /usr/app

COPY . .

RUN cargo build --release

#############

FROM debian:buster-slim

WORKDIR /

COPY --from=builder /usr/app/target/release/helloworld /helloworld

ENTRYPOINT ["/helloworld"]
```

The table below shows the size of container images for different languages. Container images with native apps can save a lot of space.

Figure 11. Sizes of container images

Language	Binary size (MB)	Container image size (MB)
Java (JVM)	N/A	223.7
Java (Native)	13	110.3
Go	1.8	21
Rust	3.7	72.9

In the container image of a native Java app, most of the space is occupied by the operation system. We can further reduce the container image size by switching to a different base image.

App Startup Time

Java programs are executed by launching the JVM with a specified entry point, which is the class with `public static void main(String[])` method.

During the execution of the program, more classes will be loaded dynamically. Classing loading is a powerful feature in Java. However, it may impact the runtime performance, especially the startup time.

Startup time is an important metrics for cloud-native apps. In a typical microservice architecture, these services use horizontal scaling to serve more requests. When the system is under heavy load, it's crucial that service replicas can start and serve requests quickly.

The extensive usage of frameworks slows down the startup of Java apps. During application startup, many frameworks will scan the classpath and perform initialization tasks, which will generate new classes or enhance existing classes. For example, Hibernate will enhance entity classes annotated with `@Entity`.

[Quarkus] provides some metrics about the startup time of native Java apps and traditional Java apps.

Figure 12. Startup times of Quarkus apps

App type	Quarkus + Native	Quarkus + JIT	Traditional
REST	0.016s	0.943s	4.3s
REST + CRUD	0.042s	2.033s	9.5s

Memory Consumption

Cloud native apps have many replicas in the runtime. If we can reduce the memory consumption of an app, the memory savings could be huge. This can cut down the cost of the whole infrastructure.

In JVM mode, Java apps may waste memory in certain cases. A typical scenario is classes metadata. For example, an app uses YAML as the format of configuration files. It requires YAML libraries to parse configuration files into Java objects in the runtime. This usually happens at the initialization phase. After the parsing is done, only the result Java objects will be used. YAML libraries won't be used any more, but their classes metadata is still in the memory.

Quarkus provides some metrics about the memory of native Java apps and traditional Java apps.

Figure 13. Memory consumption of Quarkus apps

App type	Quarkus + Native	Quarkus + JIT	Traditional
REST	12MB	73MB	136MB
REST + CRUD	28MB	145MB	209MB

Class unloading

JVM may unload classes to reduce memory use. However, the actual behavior is implementation specific and transparent to the program.

GraalVM

> GraalVM is a high-performance JDK distribution designed to accelerate the execution of applications written in Java and other JVM languages along with support for JavaScript, Ruby, Python, and a number of other popular languages.
>
> Introduction to GraalVM

GraalVM provides the `native-image` tool to build native executables for Java apps. This is the primary tool for building native Java apps.

JIT vs. AOT

When building native images using GraalVM, an important concept to understand is AOT.

JIT

Bytecode is a intermediate representation, which needs to be translated to machine code for execution. JVM can simply interpret the bytecode and translate to machine code directly. However, this approach may be less performant in certain cases. A typical example is loops. The loop body may be executed many times.

Most JVM implementations use a technology called Just-in-time (JIT) compilation to improve the performance. The JIT compiler dynamically compiles bytecode into machine code and caches the result for later use. For the loop example, the bytecode of loop body can be compiled and cached. When executing the loop, the cached machine code can be reused to speed up the execution.

JIT compilation also incurs runtime cost. JIT compiler usually employees a complicated strategy to determine which part of bytecode should be compiled.

AOT

The term Ahead-of-time (AOT) compilation may have different meanings in different contexts. For GraalVM, AOT means compiling bytecode into native machine code at build-time. This allows Java programs to run as native executables.

GraalVM AOT compilation starts from the `main` method of the main class. The executable includes the application classes, classes from third-party dependencies, classes from Java runtime libraries, and statically linked native code from JDK.

To enable AOT compilation, GraalVM runs an aggressive static analysis that requires a **closed-world assumption**. If this assumption doesn't hold, native image won't work as expected. When developing native images with GraalVM, the key point is to remove *dynamic* parts of an application.

Native Image

The native executable built by GraalVM doesn't run on the JVM, but includes necessary components from a runtime system called "Substrate VM". Substrate VM includes following components:

- Memory management - Garbage collectors
- Thread scheduling
- Java Native Interface (JNI) support
- Deoptimizer

Limitations

Due to the nature of AOT compilation, there are some limitations of GraalVM native image.

Some features require configurations.

- **Dynamic class loading**. Any class to be accessed by name at image run time must be enumerated at image build time.

- **Reflection**. Individual classes, methods, and fields that should be accessible via reflection need to be known ahead-of-time.
- **Dynamic proxy**. The list of interfaces that define dynamic proxies needs to be known at image build time.
- **JNI**. Any Java artifacts accessed by name via JNI must be specified during a native image generation.
- **Serialization**. Java serialization requires class metadata information in order to function and must be specified during a native image generation.

Some features are not compatible.

- `invokedynamic` bytecode and method handles.
- Security manager.

Some features have different behavior.

- **Signal handlers**. No signal handlers are registered by default when building a native image, unless the `--install-exit-handlers` option is used.
- **Class initializers**. Classes can be initialized at image build time.
- **Finalizers**. Finalizers are not invoked.
- **Threads**. Native Image does not implement long-deprecated methods in `java.lang.Thread`.
- **Unsafe memory access**. Fields accessed using `sun.misc.Unsafe` need to be marked if classes are initialized at image build time.
- **Debugging and monitoring**. JVMTI and other bytecode-based tools are not supported with Native Image. Debugging can only be done using native debuggers like GDB.

GraalVM and OpenJDK

Oracle is contributing GraalVM Community Edition Java code to OpenJDK to more closely align the development of the GraalVM technologies with that of Java. Oracle plans to contribute the most applicable portions of the GraalVM just-in-time (JIT) compiler and Native Image to the OpenJDK Community.

Installation

To build native Java apps, GraalVM needs to be installed on the local machine, or runs in containers.

There are several GraalVM distributions to use, see the table below:

Figure 14. GraalVM distributions

Distribution	Description	License
Oracle GraalVM	New product name after Oracle contributing GraalVM to OpenJDK	Free to use in production and free to redistribute, at no cost, under the GraalVM Free Terms and Conditions.
GraalVM Community Edition (CE)	Community edition on GitHub	Version 2 of the GNU General Public License with the “Classpath” Exception
Oracle GraalVM Enterprise Edition (EE)	Subscription required for production	Oracle Technology Network License Agreement for GraalVM Enterprise Edition for developing, testing, prototyping, and demonstrating
Liberica Native Image Kit	Based on GraalVM CE	EULA
Mandrel	Quarkus specific distribution of GraalVM CE	Same as GraalVM Community Edition

This book uses Oracle GraalVM as the base version.

Local Installation

There are different versions of Oracle GraalVM based on the OpenJDK versions.

- Java 25 based (Current Feature Release)
- Java 21 based (LTS)
- Java 17 based (LTS)

> Oracle GraalVM `25.0.1` is used in this book.

GraalVM Versioning

After GraalVM `22.3.3` release, GraalVM switched to a different versioning schema to align with OpenJDK versions. There are no specific GraalVM versions, only OpenJDK versions. For each OpenJDK version, there will be a matching GraalVM version. Old GraalVM releases can still be downloaded.

Install GraalVM

Linux / macOS

SDKMAN

Oracle GraalVM and GraalVM CE can be installed with SDKMAN.

Figure 15. Install Oracle GraalVM 25

```
sdk install java 25.0.1-graal
```

Figure 16. Install GraalVM CE 25

```
sdk install java 25.0.1-graalce
```

Manual

Depends on the GraalVM version,

- Download Oracle GraalVM releases from GraalVM.org and extract files to a proper location.
- Download GraalVM CE releases from GitHub and extract files to a proper location.

Windows

Depends on the GraalVM version,

- Download Oracle GraalVM releases from GraalVM.org and extract files to a proper location.
- Download GraalVM CE Windows releases from GitHub and extract files to a proper location.

Using WSL

Another option is to install WSL first, then install Ubuntu. This can make installation of GraalVM much easier.

Configure GraalVM

Set the `GRAALVM_HOME` environment variable to the installation location.

Figure 17. Set GRAALVM_HOME environment variable

```
export GRAALVM_HOME=<graalvm_directory>
```

Add GraalVM `bin` directory to the `PATH` environment variable.

Figure 18. Update PATH environment variable

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

Use GraalVM as the default JDK

If you want to use GraalVM as the default JDK, environment variable `JAVA_HOME` can be set to GraalVM's installation directory.

Install Native Image

Oracle GraalVM 25 already has Native Image feature bundled.

> For old versions, Native Image feature needs to be installed with GraalVM Updater using `gu install native-image`.

The `native-image` executable can be found in the `${GRAALVM_HOME}/bin` directory.

`native-image` requires the local toolchain to run.

To verify the installation of `native-image`, run the command below to check the version.

Figure 19. Check version of native-image

```
native-image --version
```

Output likes below should be printed:

Figure 20. Output of the version of native-image

```
native-image 25.0.1 2025-10-21
GraalVM Runtime Environment Oracle GraalVM 25.0.1+8.1 (build 25.0.1+8-LTS-jvm\
ci-b01)
Substrate VM Oracle GraalVM 25.0.1+8.1 (build 25.0.1+8-LTS, serial gc, compre\
ssed references)
```

Run in Containers

GraalVM CE container images are published to GitHub Container Registry. There are different packages and tags to choose for different GraalVM versions. GraalVM CE packages have the `-community` suffix.

To use Native Image, package `native-image-community` should be used.

The command below pulls container image of GraalVM CE `25.0.1`.

Figure 21. Pull container image

```
docker pull ghcr.io/graalvm/graalvm-community:25.0.1
```

A container can be started using the pulled image.

Figure 22. Start a container image

```
docker run -it --rm ghcr.io/graalvm/native-image-community:25.0.1 bash
```

Oracle GraalVM container images can be pulled from Oracle Container Registry.

Figure 23. Pull Oracle GraalVM image

```
docker pull container-registry.oracle.com/graalvm/native-image:25
```

Native Image

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

`native-image`

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Basic Usage

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Fallback Image

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Build Output

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Command Line Options

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Basic Options

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Non-standard Options

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Reflection

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Automatic Detection

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Elements With Constant Names

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Elements With Non-constant Names

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Manual Configuration

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Reflection at Image Build Time

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Class Initialization

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Class Initialization Strategy

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Debug Class Initialization

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Common Errors

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Wrong Initialization Stage

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Dynamic Proxy

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Automatic Detection

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Manual Configuration

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Resources

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Resource Files

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Configuration

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Debug Resource Registration

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Locales

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Default Locale

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Include Locales

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Resource Bundles

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Memory Management

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Garbage Collectors

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Serial GC

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Epsilon GC

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

G1 GC

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Memory Configurations

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

GC Logs

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Tracing Agent

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Basic Usage

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Build Configuration

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Advanced Usage

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Filters

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Filter File

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Caller-based Filters

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Access Filters

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Trace Files

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Static and Mostly Static Images

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Static Images

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Mostly Static Images

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Container Image

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Developer Guide

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

GraalVM Integration

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

GraalVM SDK

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Feature

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

`Feature` Interface

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Write Custom Features

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

A Simple Feature

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Enable with `--features`

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Enable with `@AutomaticFeature`

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Provide Runtime Configurations

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Class Initialization

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Reflection

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Serialization

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Resources

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Dynamic Proxy

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Example

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Advanced Features

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Feature Inclusion

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Native Image Debug Info Feature

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

JavaFX Feature

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Java Net HTTP Feature

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Feature Dependencies

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Substitution

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Target Class

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Alias

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

`@Alias`

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

`@Delete`

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

`@Substitute`

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

`@AnnotateOriginal`

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Delete

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Replace

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Target Element

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Recompute Field Value

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Target Class

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Kind

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

`Reset`

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

`NewInstance`

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

`FromAlias`

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

`FieldOffset`

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

`ArrayBaseOffset` and `ArrayIndexShift`

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

`AtomicFieldUpdaterOffset`

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

`Manual`

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

`Custom`

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Inject Accessor

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Inject

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

JDK Flight Recorder

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Basic Usage

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Custom Events

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Continuous Integration

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Maven

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Debug Native Image

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Debug Info

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Debugging

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Frameworks

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Quarkus

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Build Container Images

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Native Image

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Configurations

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Resources

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Reflections

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Class Initialization

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Dynamic Proxy

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Internals

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Tools

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.

Native Image Build Report

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/build-native-java-apps-graalvm.