GraalVM™ Native Image Quick Reference v2

Tested with GraalVM 22.3.0

INSTALL

Download GraalVM JDK and install Native Image. The quickest way to get started is to run:

```
bash <(curl -sL https://get.graalvm.org/jdk)
```

BUILD

Use `bin/native-image` in the same way you would use `bin/java`. Here are some common examples:

for a JAR:

```
native-image -jar Foo.jar
```

for a module:

```
native-image -p Foo.jar -m org.foo.Main
```

for a class:

```
native-image -cp Foo.jar org.foo.Main
```

BUILD OPTIONS

Enable the quick build mode for development:

```
-Ob
```

Specify a name for the resulting binary:

```
-o <myapp>
```

Build a shared library:

```
--shared
```

Build a statically linked binary with libc implementation:

```
--static –libc=<glib | musl | bionic>
```

Pass a comma-separated list of URL protocols:

```
--enable-url-protocols=<http | https | file |
resource | JAR>
```

Specify a list of packages and classes to be linked at build time:

```
--link-at-build-time=<comma-separated list of
packages and classes>
```

Specify a garbage collector:

```
--gc=<serial | G1* | epsilon>
```

Install exit handlers:

```
--install-exit-handlers
```

Control class initialization at build or run time:

```
--initialize-at-build-time=<comma-separated list
of packages and classes>
```

or

```
--initialize-at-run-time=<comma-separated list
of packages and classes>
```

Embed a Software Bill of Materials (SBOM)*:

```
--enable-sbom
```

Enable support for GraalVM languages:

```
--language:<python | js | ruby | R | wasm>
```

Enable Profile-Guided Optimizations (PGO)* to improve performance and throughput:

1. Build an instrumented native executable:

```
native-image --pgo-instrument MyApp
```

2. Run the executable to record profiles:

```
./myapp
```

3. Build an optimized native executable:

```
native-image --pgo=default.iprof MyApp
```

List all options for `native-image`:

`--help` or `--help-extra`

INTEGRATIONS

Use the **Gradle** plugin for GraalVM Native Image:

https://graalvm.github.io/native-build-tools/latest/gradle-plugin

Use the **Maven** plugin for GraalVM Native Image:

https://graalvm.github.io/native-build-tools/latest/maven-plugin

Use the **GitHub Action** for GraalVM:

https://github.com/marketplace/actions/github-action-for-graalvm

CONFIGURE

There are three ways to configure Native Image for external Java libraries:

1. Configuration is provided by a framework such as Micronaut, Helidon, Spring, and Quarkus
2. Configuration is automatically pulled in by the Maven/Gradle plugin from GraalVM Reachability Metadata Repository
3. Configuration is provided manually. The Tracing Agent can help in this process:

a. Run a Java application with the agent to generate the configuration:

```
java -agentlib:native-image-agent=config-
output-dir=/path/to/config-dir/ -jar MyApp.jar
```

b. Build a native executable with the configuration:

```
native-image -jar MyApp.jar
```

Configuration files in META-INF/native-image on the class path or module path are included automatically.

MONITOR

Tune the garbage collector:

```
./myapp -Xmx<m> -Xmn<m> …
```

Gather garbage collection logs:

```
./myapp -XX:+PrintGC -XX:+VerboseGC
```

Prerequisite: Enable VM inspection when building a native executable with: `--enable monitoring=<all,heapdump,jfr,jvmstat>`. Defaults to `all`.

Enable JFR support and record events:

```
./myapp -XX:+FlightRecorder
-XX:StartFlightRecording="filename=recording.jfr"
```

Dump the initial heap of a native executable:

```
./myapp -XX:+DumpHeapAndExit
```

DEBUG

Build a native executable with debug information and with compiler optimizations disabled:

```
-g -O0
```

Debug a native executable with GDB or any IDE that integrates GDB.

---

* Available with GraalVM Enterprise Native Image

More info at www.graalvm.org/native-image