GraalVM™ Native Image Quick Reference – GraalVM for JDK 23

BUILDING

Use `native-image` in the same way you would use `java` to execute a class:

```
native-image [options] <mainclass> [args...]
```

to execute a JAR file:

```
native-image [options] -jar <jarfile> [args...]
```

to execute the main class in a module:

```
native-image [options] -m <module>[/<mainclass>]
[args...]
```

Build a shared library:

`--shared`

Build a statically linked binary with the `libc` implementation [2]:

`--static --libc=musl`

Build a mostly static binary [2]:

`--static-nolibc`

Maven

Add Native Build Tools plugin to the `<plugins>` section of your *pom.xml*:

```
<plugin>
   <groupId>org.graalvm.buildtools</groupId>
   <artifactId>native-maven-plugin</artifactId>
   <version>${buildtools.version}</version>
</plugin>
```

Build:

`./mvnw -Pnative package`

Pass options to `native-image` in the plugin configuration:

```
<configuration>
   <buildArgs>
       <buildArg>--option</buildArg>
   </buildArgs>
</configuration>
```

Run JUnit tests:

`./mvnw -Pnative test`

More at:

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

Gradle

Add Native Build Tools plugin to plugins in *build.gradle*:

```
plugins {
  id 'org.graalvm.buildtools.native'
version'${buildtools.version}'
}
```

Build:

`./gradlew nativeCompile`

Pass options to `native-image` in the plugin configuration:

```
graalvmNative {
  binaries {
    main {
      buildArgs.add("--option")
    }
  }
}
```

Run JUnit tests:

`./gradlew nativeTest`

More at:

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

OPTIMIZING

Improved performance

Enable Profile-Guided Optimizations (PGO) [1]:

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`

Enable the G1 garbage collector [1] [2]:

`--gc=G1`

Enable more CPU features to match the architecture of the build machine (use `-march=list` to list all types):

`-march=native`

Optimize for the best performance (without using PGO):

`-O3`

Faster build time

Enable the quick build mode (for development only):

`-Ob`

Smaller binary size

Create the smallest possible image at the cost of reduced performance:

`-Os`

Reduced memory footprint

Set the max heap size for more predictable memory usage:

`./myapp -Xmx<m>…`

DEVELOPER TOOLING

Generate and explore a Build Report [1]:

`--emit build-report`

Embed a Software Bill of Materials (SBOM) to identify application dependencies [1]:

`--enable-sbom`

Enable application monitoring features (defaults to `all`):

`--enable-monitoring=heapdump, jfr, jvmstat, jmxserver, jmxclient, nmt, threaddump`

Gather statistics with the Linux `perf` profiler [2]:

`perf stat ./myapp`

[1] Available with Oracle GraalVM

[2] Linux only

More info at www.graalvm.org