Spring Aot – GraalVm

**Definition:**

Ahead-of-Time (AOT) compilation is a technique that pre-compiles bytecode into native machine code before the application runs.(from Java 9)

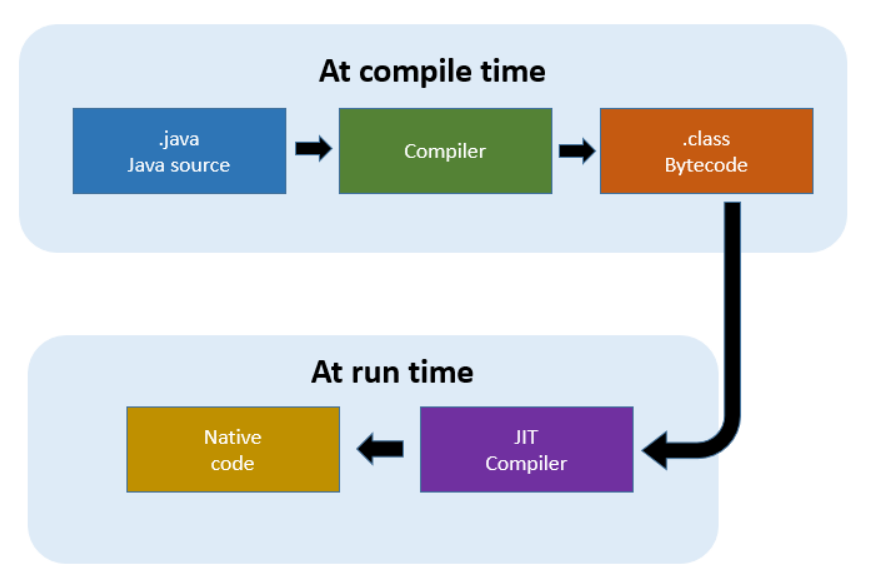
Java Virtual Machines do not commonly support this feature. However, Oracle has released an experimental AOT feature for the HotSpot JVM in the OpenJDK project called “GraalVM Native Image” that allows for the ahead-of-time compilation.

After pre-compiling the code, the computer’s processor can execute it directly, eliminating the need for the JVM to interpret the bytecode and improving the start-up time.

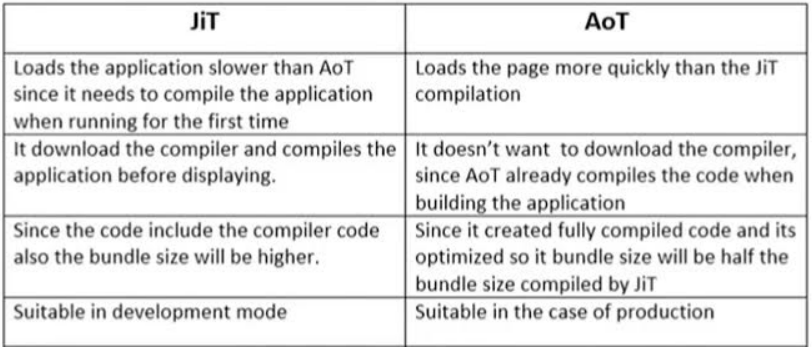
**JIT (Just In Time):**

 when we compile the source code (.java file), the produced bytecode is stored in .class files. This way, **the JVM uses a JIT compiler to convert bytecode into machine code.**

JIT compilation involves the interpretation of the bytecode by the JVM and the dynamic compilation of frequently executed code into native machine code during runtime**.**



Difference between JIT and AOT



**Pre-requisites:**

**For Windows:**

* Install the Visual Studio Build Tools with the Windows 11 SDK (or later version)
* Install Visual Studio with the Windows 11 SDK (or later version)
* We have to install desktop development with C++

**Why we need..?**

=> During native image generation (native code), the system links Java code with native libraries (like system libraries or those used by GraalVM).

=> Visual Studio tools handle this linking process on Windows.

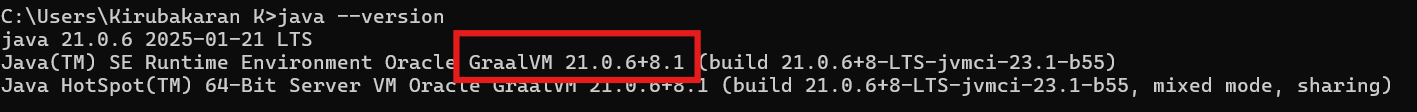
**For Mac:**

xcode-select –install

**For Linux:**

sudo apt-get install build-essential zlib1g-dev

**GraalVm:**

We have to install Jdk based on GraalVM and cross verify with checking version

**Maven:**

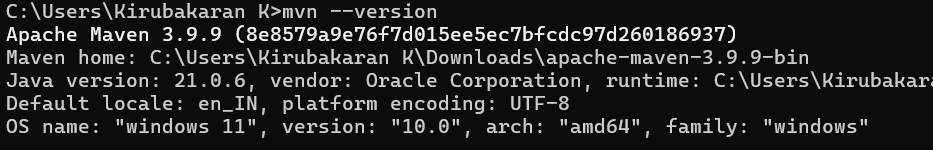
1. Install the Maven with this following Link:

https://maven.apache.org/download.cgi

1. Unzip it
2. Add Path variable:
   1. Variable Name -- Maven\_home
   2. Variable value – path/to/maven bin

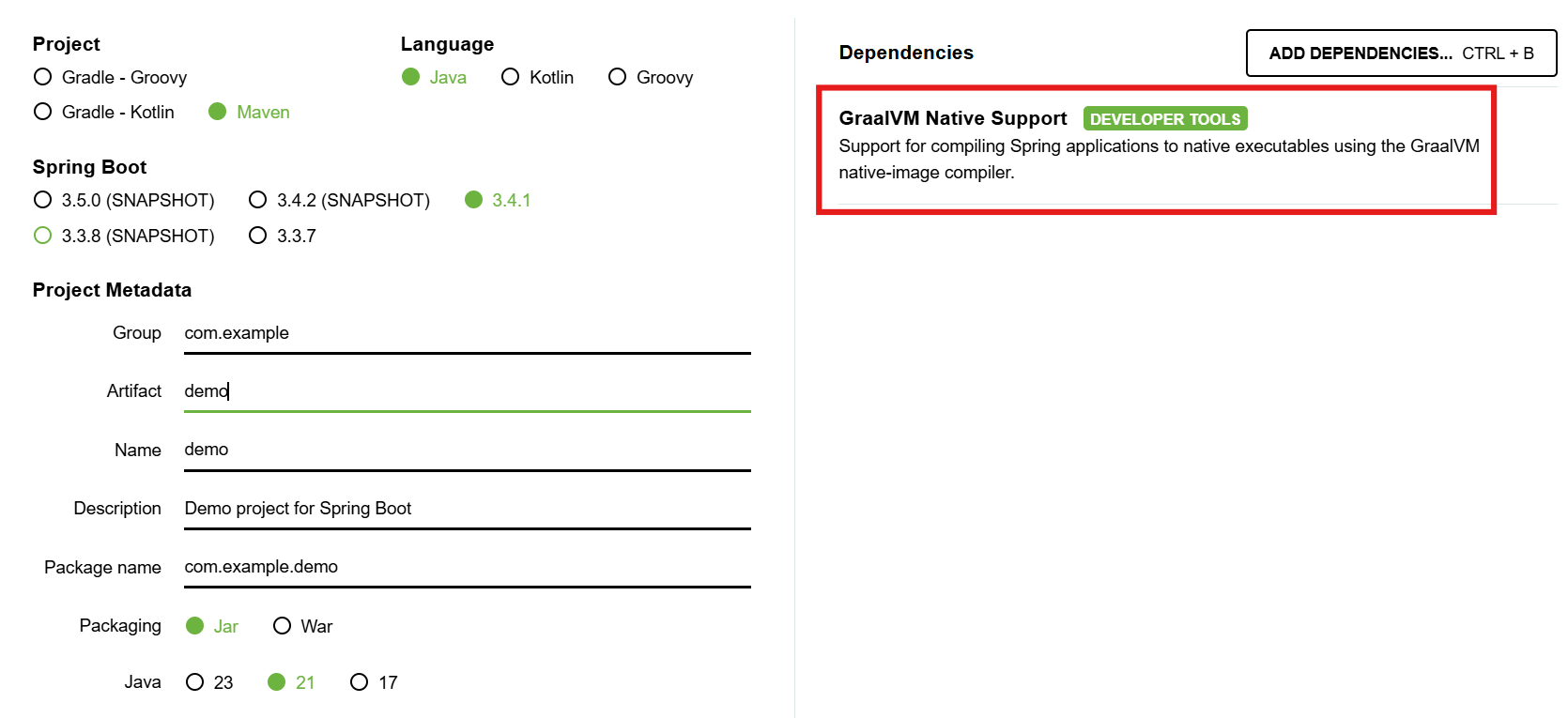
As well as add in the path %MAVEN\_HOME%\bin



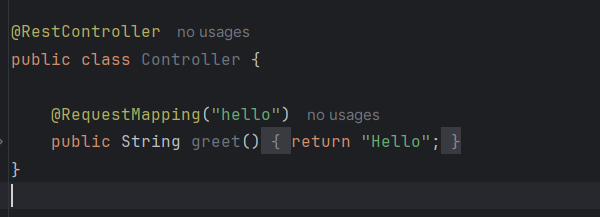
1. Check version: 

**Create Project in spring Boot:**

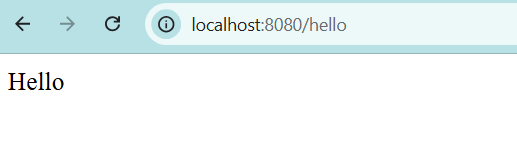
**1)**

****

2) Create Simple Controller class:



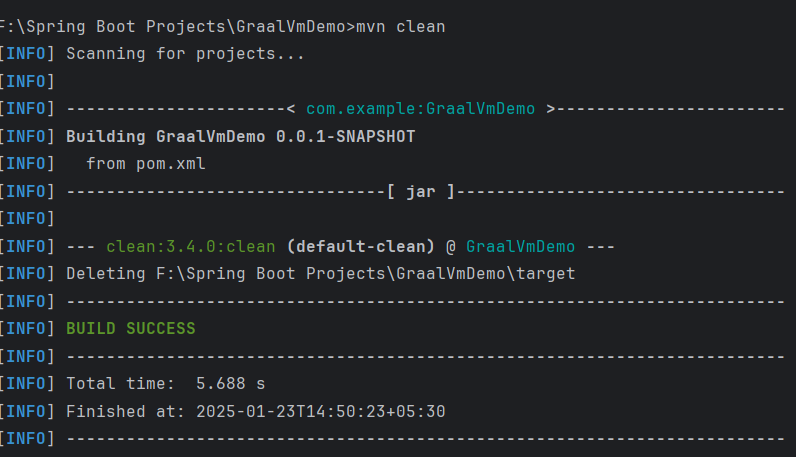
3) Run the application :



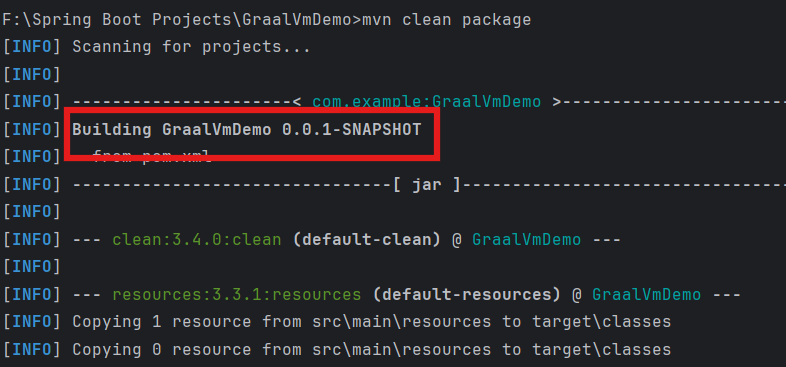
**Convert into Native image:**

1. Clean the previous execution if it happens:

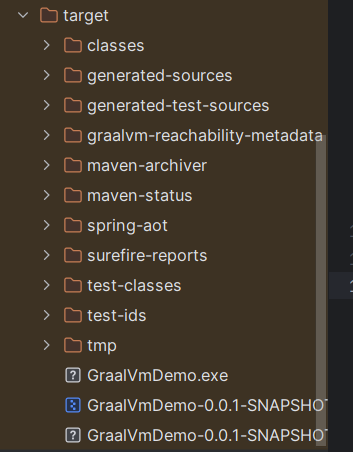
Execute the Command: mvn clean



1. Create target file and the jar file:

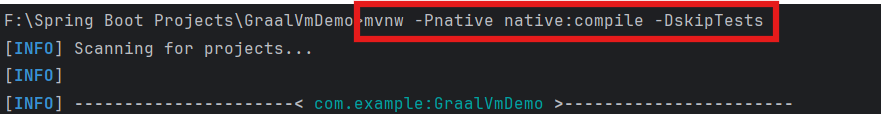
Execute the command: mvn clean package

After running this commad: Target folder created

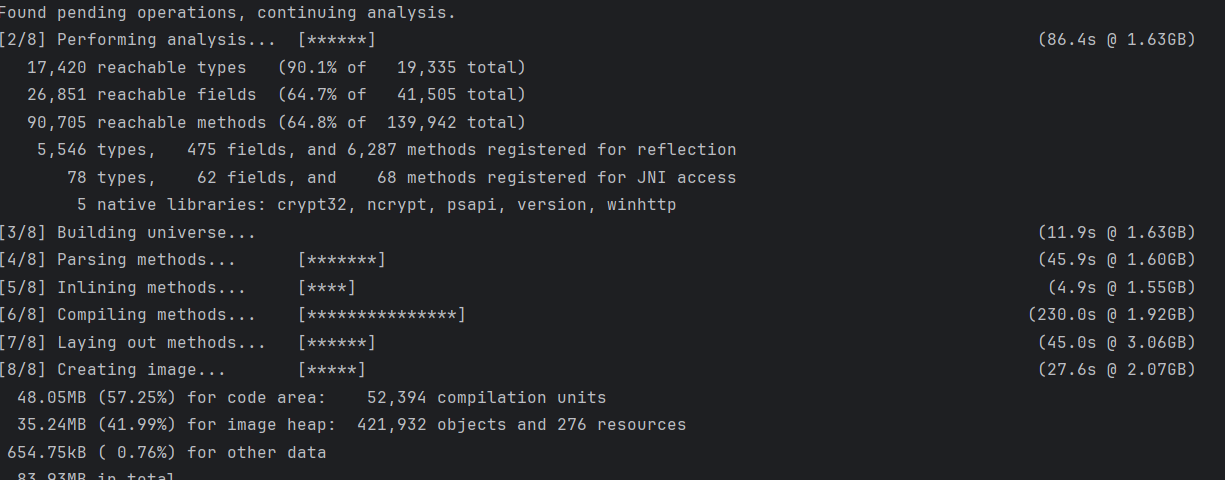


1. **Build image:**

Command: mvnw -Pnative native:compile -DskipTests



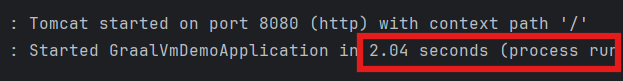
* It build the image with several stages and it took long time to built:

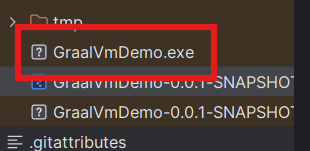


1. **After run the jar file**:

**Command**: java -jar ./target/GraalVmDemo-0.0.1-SNAPSHOT.jar

It takes **2.04 sec** to start the server :

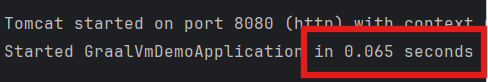


1. **But,**
2. We can see the executable file 
3. To run the image (windows):

Change the directory: cd targer

Run the image: GraalVmDemo.exec

It takes only **0.065s** to start the server



**Limitations of AOT:**

**Larger Binary Size:**

AOT compilation typically results in a larger executable file because the entire program, including runtime and libraries, is compiled into a single native binary.

**Impact:** This can increase the disk space required and make distribution less efficient.

**Slower Compilation Time:**

The AOT compilation process involves complex optimizations and static analysis, which can take significantly longer than Just-in-Time (JIT) compilation.

**Impact**: This increases the build time, making iterative development and testing slower

**Limited Runtime Optimizations:**

AOT compilation optimizes the code at build time based on static information, whereas JIT compilers optimize dynamically at runtime based on actual execution patterns.

**Impact**: AOT might miss certain performance improvements that JIT could achieve by adapting to runtime conditions.