The Path Towards Spring Native Applications

Yogi Rampuria, Advisory Platform Architect, VMware
March 2021
@yogendra | github.com/yogendra | yogendra.me

(based on the work from Martin Lippert, Sébastien Deleuze, Andy Clement, and others)

Welcome to Spring Native Beta

(0.9.0 released on March 11, 2021)

spring

Agenda

- GraalVM native - the secret superpower behind Spring Native
- What is Spring Native?
- Getting Started with Spring Native
- Compatibility and Support
- Early numbers
- Q & A

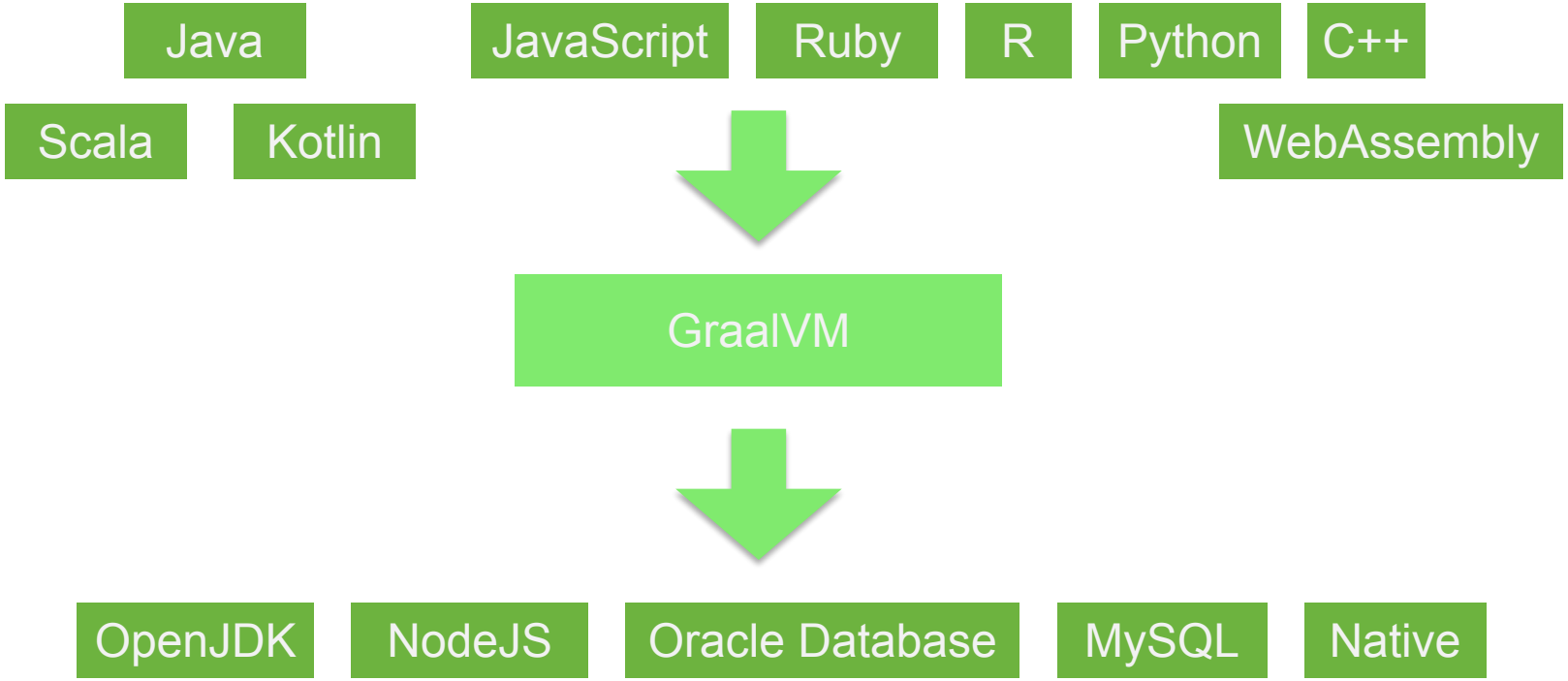
spring

GraalVM native - the secret superpower behind Spring Native

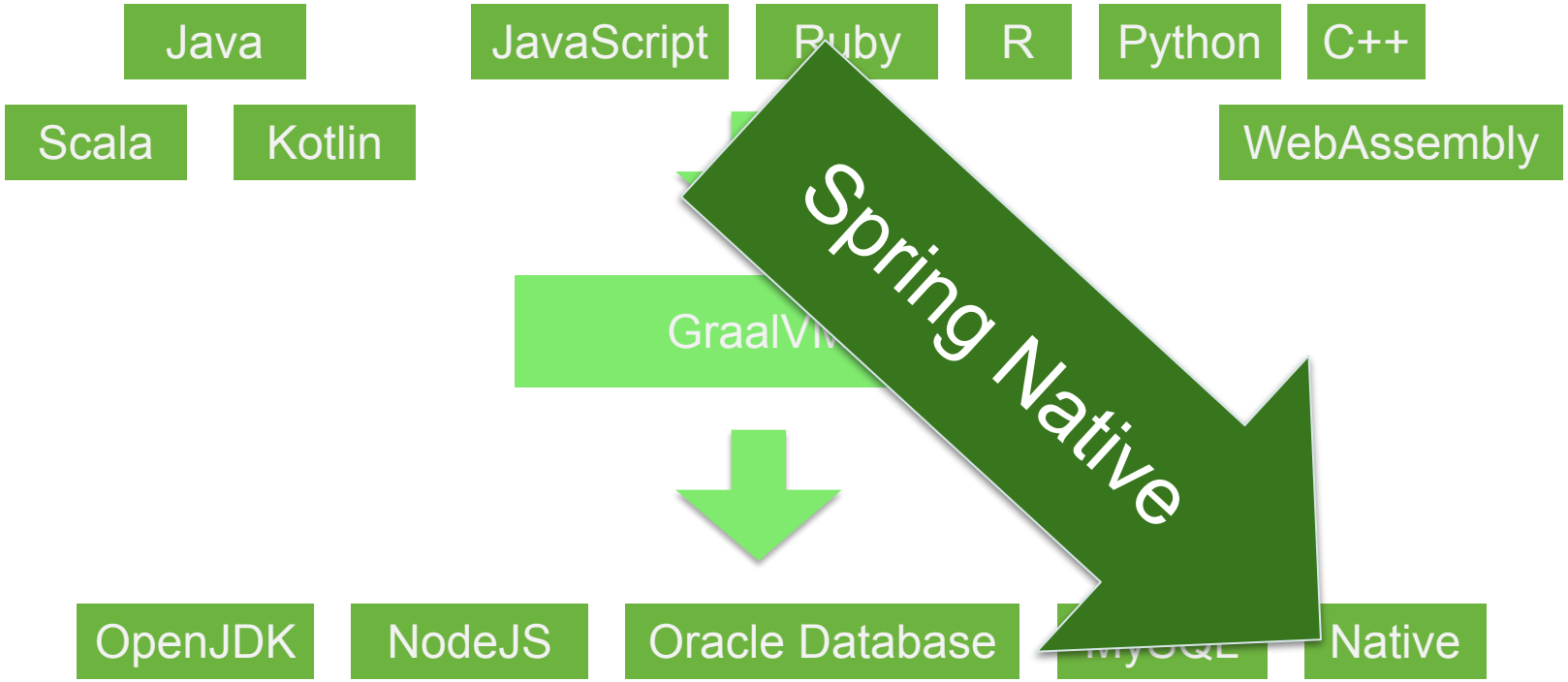
GraalVM - a high-performance polyglot VM

Basic idea: One virtual machine that can execute all languages (Java, JavaScript, R, Ruby, Python, C, C++, Kotlin, Scala, etc.)

- https://graalvm.org/
- You can run all those languages without any boundaries between them

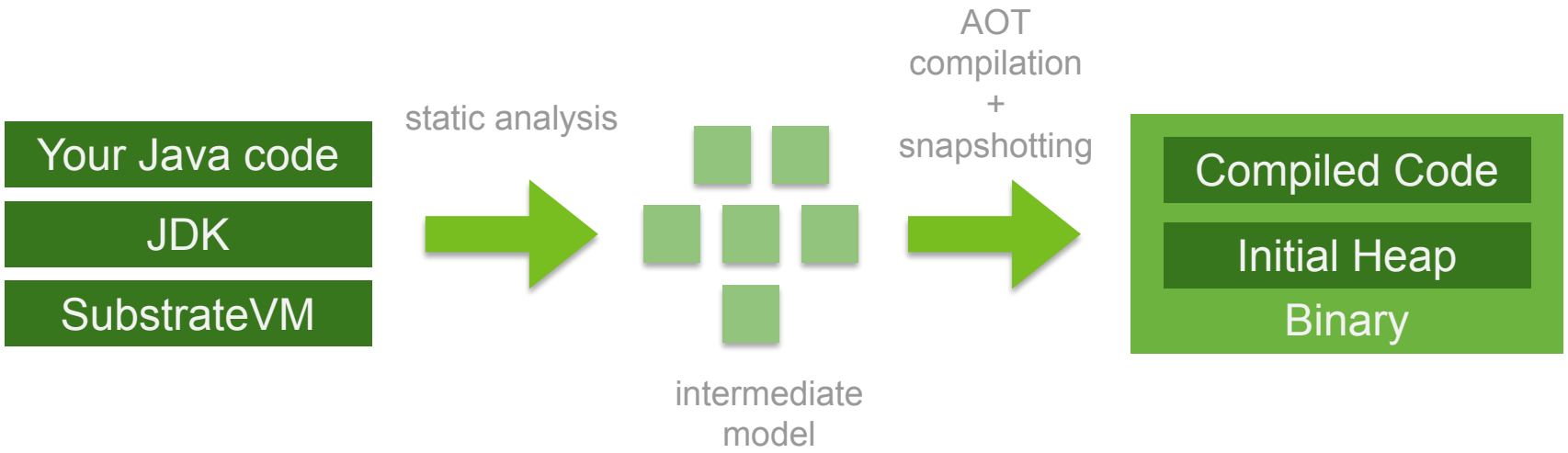

Java
JavaScript
Ruby
R
Python
C++
Scala
Kotlin
WebAssembly
GraalVM
OpenJDK
NodeJS
Oracle Database
MySQL
Native

spring


Java
JavaScript
Ruby
R
Python
C++
Scala
Kotlin
WebAssembly
Spring Native
OpenJDK
NodeJS
Oracle Database
Native

spring

GraalVM Native Image Technology



Two main benefits

Reduced memory consumption

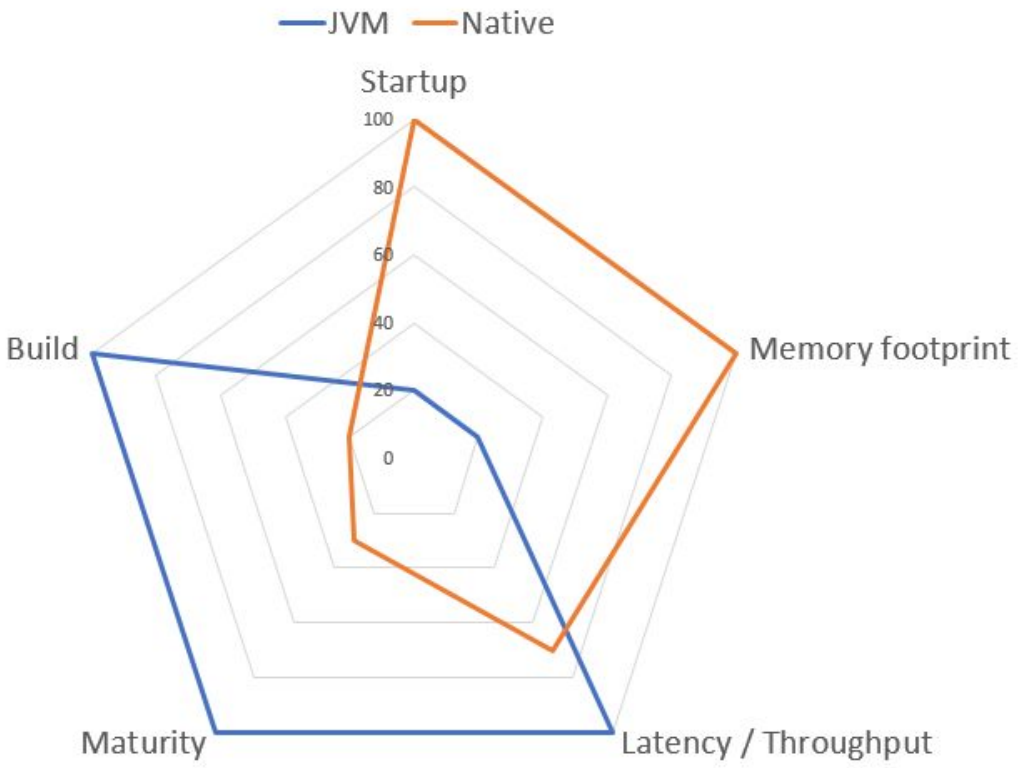
- 3x - 5x memory reduction (RSS)
- That means cheaper cloud instances
- Great for systems that are divided into many small microservices

Instant startup time

- Scale to zero (run your app only when it is used)
- Serverless for any kind of workload
- Good fit for platforms like Knative, etc.

spring

Trade Offs



spring

Key differences between JVM and Native Images

Native apps are different from JVM apps

- Static analysis of the app at build-time using a specific entry point (including aggressive removal of code)
- Upfront configuration required for proxies, reflection, resources
- Classpath is fixed at build-time
- No lazy class-loading
- Some code will run at build-time
- No runtime optimizations

GraalVM native is a great source of inspiration for the JVM ecosystem (e.g. Project Leyden)

Still in “early adopter” mode, but matures quickly

GraalVM in Action

yogendra/native-spring-workshop

Spring Native

Our goal is to support compilation of existing Spring Boot applications into native executables - <u>unchanged</u>

Key observations

The Spring Boot standalone deployment model is a great fit with GraalVM native image technology

- But it requires configuration for reflection/proxies/resources
- Spring Boot usually uses a lot of those technologies
- The Spring team is doing the hard work for you:
  - Auto-generate the required configs
  - Reduce the use of the above mentioned technologies using AOT techniques

Collaboration between the GraalVM and Spring teams

“We are excited about the great partnership between the Spring and GraalVM engineering teams to support native ahead-of-time compilation for millions of Spring Boot applications. This is a game changer enabling low memory footprint and instant startup for these workloads.”

Thomas Wuerthinger, GraalVM founder & project lead

spring

Collaboration between the GraalVM and Spring teams

“It is a great joy to collaborate with the Spring team on crafting the native JVM ecosystem: their deep technical knowledge, wrapped with sensitive touch for the community always leads to the best solutions. The latest Spring Native release, and its numerous usages in the JVM ecosystem, pave the way for the wide adoption of native compilation.”

Vojin Jovanovic, Principal Researcher at Oracle Labs, GraalVM

spring

Spring Native

https://github.com/spring-projects-experimental/spring-native

Search or jump to... / Pull requests Issues Marketplace Explore

spring-projects-experimental / spring-native

Watch 77 Star 1k Fork 134

Code Issues 68 Pull requests 1 Actions Security Insights

master 8 branches 13 tags Go to file Add file Code

sdeleuze Add missing Cloud Discovery hints 338d53a 2 hours ago 1,829 commits

.mvn/wrapper	Upgrade Maven wrappers to 3.6.3	9 months ago
ci	Bring back Slack notification for failed builds	5 hours ago
docker	Make docker work in container with vfs	last month
org.springframework.experimental....	Harmonize pom.xml format	9 days ago
samples	Update BP environment variables	5 hours ago
scripts	Add sample application for Gradle AOT plugin	10 days ago
spring-aot-gradle-plugin	Set consistently the copyright in Java sources	6 hours ago
spring-aot-maven-plugin	Set consistently the copyright in Java sources	6 hours ago
spring-aot	Set consistently the copyright in Java sources	6 hours ago
spring-native-configuration	Add missing Cloud Discovery hints	2 hours ago
spring-native-docs	Clarify Spring Boot support policy.	3 hours ago
spring-native-tools	Set consistently the copyright in Java sources	6 hours ago
spring-native	Set consistently the copyright in Java sources	6 hours ago
.gitignore	Add flatten maven plugin	9 days ago
LICENSE.txt	First commit	2 years ago

About

Spring Native provides an incubating support for compiling Spring applications to native executables using GraalVM native-image compiler.

spring.io/blog/2020/11/23/spring-na...

spring spring-boot serverless native graalvm

Readme

Apache-2.0 License

Releases

13 tags

Contributors 22

+ 11 contributors

spring

What is Spring Native?

Beta Spring Boot native application support

- It includes a plugin for the GraalVM native image builder
- Analyses the Spring Boot application at build time
  - Computes the most optimal native image configuration
  - Challenge is doing that with static analysis
- Also perform some build time transformation for:
  - Optimized footprint
  - Compatibility

Reference documentation

https://docs.spring.io/spring-native/docs/current/reference/htmlsingle/#overview

spring

Spring Native documentation

Version 0.9.0-SNAPSHOT - Andy Clement · Sébastien Deleuze · Filip Hanik · Dave Syer · Esteban Ginez · Jay Bryant · Brian Clozel · Stéphane Nicoll

1. Overview

Spring Native provides support for compiling Spring applications to native executables using the GraalVM native-image compiler.

Compared to the Java Virtual Machine, native images can enable cheaper and more sustainable hosting for many types of workload. These include microservices, function workloads, well suited to containers, and Kubernetes

Using native image have key advantages, such as instant startup, instant peak performance, and reduced memory consumption.

There are also some drawbacks and trade-offs that the GraalVM native project expect to improve on over time. Building a native image is a heavy process that is slower than a regular application. A native image has fewer runtime optimizations after warmup. Finally, it is less mature than the JVM with some different behaviors.

The key differences between a regular JVM and this native image platform are:

- A static analysis of your application from the main entry point is performed at build time.
- The unused parts are removed at build time.
- Configuration is required for reflection, resources, and dynamic proxies.

spring

Two ways to use Spring Native

Via Buildpacks

- Configure your build to use the Paketo Buildpacks
- Tell the buildpack to produce a native image
- The result is a small container image with the compiled native executable inside
- No local GraalVM installation needed
- Super easy to use

Via GraalVM native image Maven plugin

- Configure your build to compile to a native executable
- Produces a native executable for the platform you are running on
- Requires GraalVM locally installed
- Also super easy to use

spring

Getting Started

https://start.spring.io

spring initializr

Project

- (●) Maven Project
- ( ) Gradle Project

Language

- (●) Java
- ( ) Kotlin
- ( ) Groovy

Spring Boot

- ( ) 2.5.0 (SNAPSHOT)
- ( ) 2.5.0 (M2)
- ( ) 2.4.4 (SNAPSHOT)
- (●) 2.4.3
- ( ) 2.3.10 (SNAPSHOT)
- ( ) 2.3.9

Project Metadata

Group	com.example
Artifact	demo
Name	demo
Description	Demo project for Spring Boot
Package name	com.example.demo
Packaging	(●) Jar ( ) War
Java	( ) 15 (●) 11 ( ) 8

Dependencies ADD DEPENDENCIES... ⌘ + B

Spring Native [Experimental] DEVELOPER TOOLS

Incubating support for compiling Spring applications to native executables using the GraalVM native-image compiler.

GENERATE ⌘ + ↵ | EXPLORE CTRL + SPACE | SHARE...

Spring Native via start.spring.io

Ready to go

- Generates projects that include all the required dependencies
- Readily configured for Paketo Buildpacks to produce native images
- HELP.MD with pointers to additional resources and documentation

Build Spring Native application using the Paketo Buildpacks

spring

Use Spring Boot 2.4.3

```
<parent>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-parent</artifactId>
  <version>2.4.3</version>
  <relativePath/> <!-- lookup parent from repository -->
</parent>
```

spring

Configure Maven build to use buildpacks + Spring native

```
<plugin>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-maven-plugin</artifactId>
  <configuration>
    <image>
      <builder>paketobuildpacks/builder:tiny</builder>
      <env>
        <BP_NATIVE_IMAGE>true</BP_NATIVE_IMAGE>
      </env>
    </image>
  </configuration>
</plugin>
```

spring

Add dependency to Spring Native

```
<dependency>
  <groupId>org.springframework.experimental</groupId>
  <artifactId>spring-native</artifactId>
  <version>0.9.0</version>
</dependency>
```

spring

Add Spring AOT plugin to the build

```
<plugin>
  <groupId>org.springframework.experimental</groupId>
  <artifactId>spring-aot-maven-plugin</artifactId>
  <version>0.9.0</version>
  <executions>
    <execution>
      <id>test-generate</id>
      <goals>
        <goal>test-generate</goal>
      </goals>
    </execution>
    <execution>
      <id>generate</id>
      <goals>
         <goal>generate</goal>
      </goals>
    </execution>
  </executions>
</dependency>
```

spring

Spring AOT

Ahead-of-time transformations for your Spring application

- Incubating in spring-native
- A general purpose framework for performing tasks, like code analysis/generation at project build time (not native image build time, regular project build time)
- Currently being used for:
  - generating code to represent what is configured in spring.factories code, optimizing away entries that aren't valid in this system (i.e. their ConditionalOnClass or similar conditions can't pass)
  - generating reflection/resource/proxy configuration for later use by native-image

Spring AOT

Ahead-of-time transformations for your Spring application

- Extensible by other portfolio projects (for example: Spring Data)
- Key benefits:
  - push code from startup time to build time
  - if what's happening is using java constructs (method references, lambdas) rather than reflection, the native-image static analysis understands the system more easily -> producing more optimal images.
- More to follow here!

spring

Run the build

```
> mvn spring-boot:build-image
Successfully built image 'docker.io/library/demo:0.0.1-SNAPSHOT'
Total time:  60 s
```

spring

Run the native app in the container

```
> docker run -p 8080:8080 docker.io/library/demo:0.0.1-SNAPSHOT

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::

Started application in 0.05 seconds (JVM running for 0.009)
```

spring

Build Spring Native application directly using Maven

Configure Maven Plugin

```
<profiles>
  <profile>
    <id>native-image</id>
    <build>
      <plugins>
        <plugin>
          <groupId>org.graalvm.nativeimage</groupId>
          <artifactId>native-image-maven-plugin</artifactId>
          <version>21.0.0</version>
          <configuration>
            <mainClass>com.example.restservice.RestServiceApplication</mainClass>
          </configuration>
          <executions>
            <execution>
              <goals>
                <goal>native-image</goal>
              </goals>
              <phase>package</phase>
            </execution>
          </executions>
          ...
```

spring

Set a classifier to avoid a clash

```
<plugin>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-maven-plugin</artifactId>
  <configuration>
    <classifier>exec</classifier>
  </configuration>
</plugin>
```

spring

Add Spring AOT plugin to the build

```
<plugin>
  <groupId>org.springframework.experimental</groupId>
  <artifactId>spring-aot-maven-plugin</artifactId>
  <version>0.9.0</version>
  <executions>
    <execution>
      <id>test-generate</id>
      <goals>
        <goal>test-generate</goal>
      </goals>
    </execution>
    <execution>
      <id>generate</id>
      <goals>
         <goal>generate</goal>
      </goals>
    </execution>
  </executions>
</dependency>
```

spring

Run the build

```
> mvn -Pnative clean package
Total time:  60 s
```

spring

Run the native executable directly

```
> target/com.example.demo.demoapplication

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::

Started application in 0.05 seconds (JVM running for 0.009)
```

spring

What does that mean?

Show me the numbers

Sample	On the JDK	Native application
actuator-r2dbc-webflux	Build: 8s Memory(RSS): 640M Startup time: 3.0s	Build: 141s +1700% Memory(RSS): 86M -87% Startup time: 0.094s -97%

Sample	On the JDK	
petclinic-jdbc	Build: 9s Memory(RSS): 417M Startup time: 2.6s	Build: 194s +2050% Memory(RSS): 101M -75% Startup time: 0.158s -94%

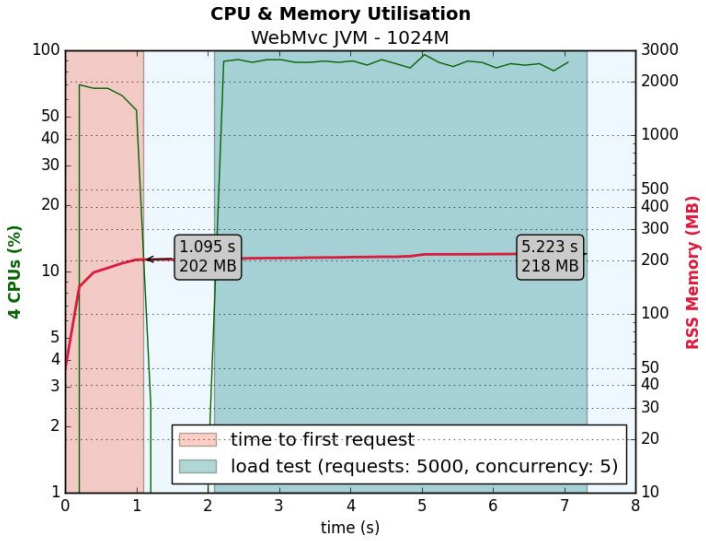
spring

Ongoing improvements in reducing footprint

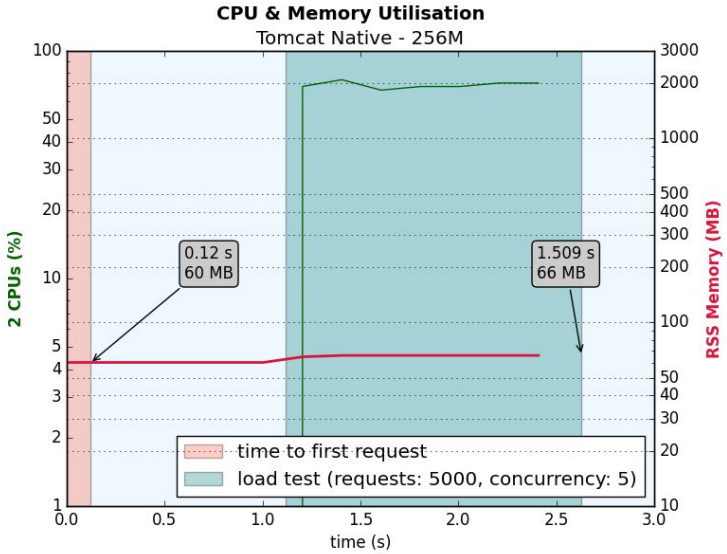
Sample	Sep 2019	Mar 2021
commandlinerunner	Build: 90s Exec. size: 48M Memory(RSS): 29M	Build: 50s Exec. size: 21M Memory(RSS): 22M
webflux-netty	Build: 193s Exec. size: 81M Memory(RSS): 95M	Build: 79s Exec. size: 47M Memory(RSS): 48M
webmvc-tomcat	Build: 203s Exec size: 105M Memory(RSS): 70M	Build: 67s Exec. size: 45M Memory(RSS): 51M

spring

Instant startup, less resources



Spring Boot on JVM,
4 vCPU, 1G RAM



Spring Boot on Native,
2 vCPU, 256M RAM

spring

The road ahead

Just released: Spring Native 0.9.0

New and noteworthy

- GraalVM 21.0.0 baseline
- Spring Boot 2.4.3
- Significant footprint improvements
- Wider range of supported technology
- Uses AOT code generation

Supported Starters

Starters

- Actuator
- Data (JDBC, JPA, MongoDB, Neo4J, R2DBC, Redis)
- JDBC
- Logging (Logback)
- Mail
- Thymeleaf
- RSocket
- Security, OAuth2

Starters (cont.)

- Validation
- Web (Spring MVC with Tomcat)
- Webflux (Netty)
- Wavefront
- Websocket
- Cloud Config
- Cloud Config (Client + Server)
- Cloud Function (Web, WebFlux, AWS)
- Cloud Netflix Eureka Client

Additionally supported

- Spring Kafka
- GPRC
- H2
- MySQL JDBC driver
- PostgreSQL JDBC driver

spring

Coming up: Spring Native 0.9.1

Bugfix release

- Bugfixes
- Update to Spring Boot 2.4.4

Coming up: Spring Native 0.10.0+

Continuous Updates to latest Spring Boot version

Native Testing

Improved AOT features

- E.g. converting existing into functional bean definitions

spring

Resources

All about Spring Native

- Beta release (0.9.0) announcement:
  https://spring.io/blog/2021/03/11/announcing-spring-native-beta
  https://www.youtube.com/watch?v=96n_YpGx-JU
- GitHub:
  https://github.com/spring-projects-experimental/spring-native/
- Reference Documentation:
  https://docs.spring.io/spring-native/docs/current/reference/htmlsingle/
- Roadmap:
  https://github.com/spring-projects-experimental/spring-native/milestones

Thank you

Contact me at yrampuria@vmware.com
@yogendra