

# The Path Towards Spring Boot Native Applications

Sébastien Deleuze  
@sdeleuze  
VMware

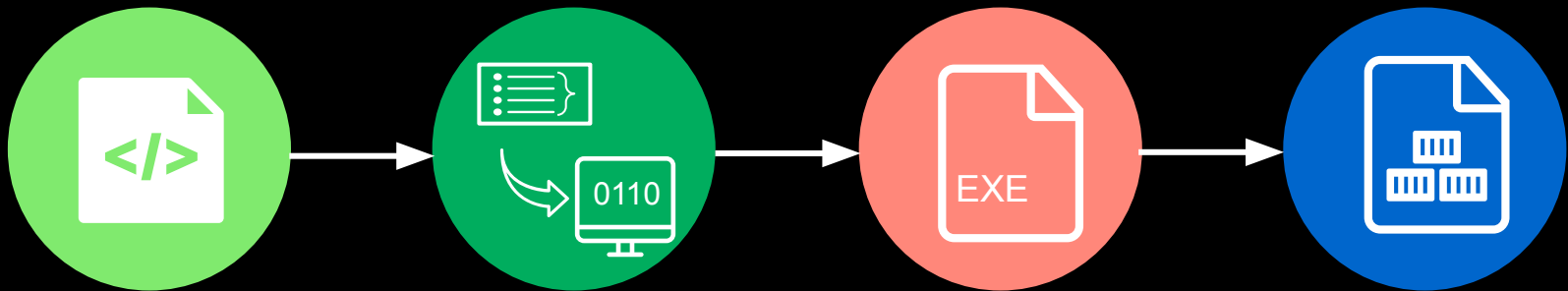
# Safe Harbor Statement

The following is intended to outline the general direction of VMware's offerings. It is intended for information purposes only and may not be incorporated into any contract. Any information regarding pre-release of VMware offerings, future updates or other planned modifications is subject to ongoing evaluation by VMware and is subject to change. This information is provided without warranty of any kind, express or implied, and is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions regarding VMware's offerings. These purchasing decisions should only be based on features currently available. The development, release, and timing of any features or functionality described for VMware's offerings in this presentation remain at the sole discretion of VMware. VMware has no obligation to update forward looking information in this presentation.

# Agenda

- Native executables
- Spring Native
- Getting started
- Demo
- The road ahead

# Native executables



Java or Kotlin  
application

GraalVM native  
compiler

Native  
executable

Lightweight  
container image

# Cheaper and more sustainable Java applications

## Instant Startup

Scale to zero

You only pay when your application is used

Serverless for any kind of workload including regular web applications

Good fit with platforms like Knative

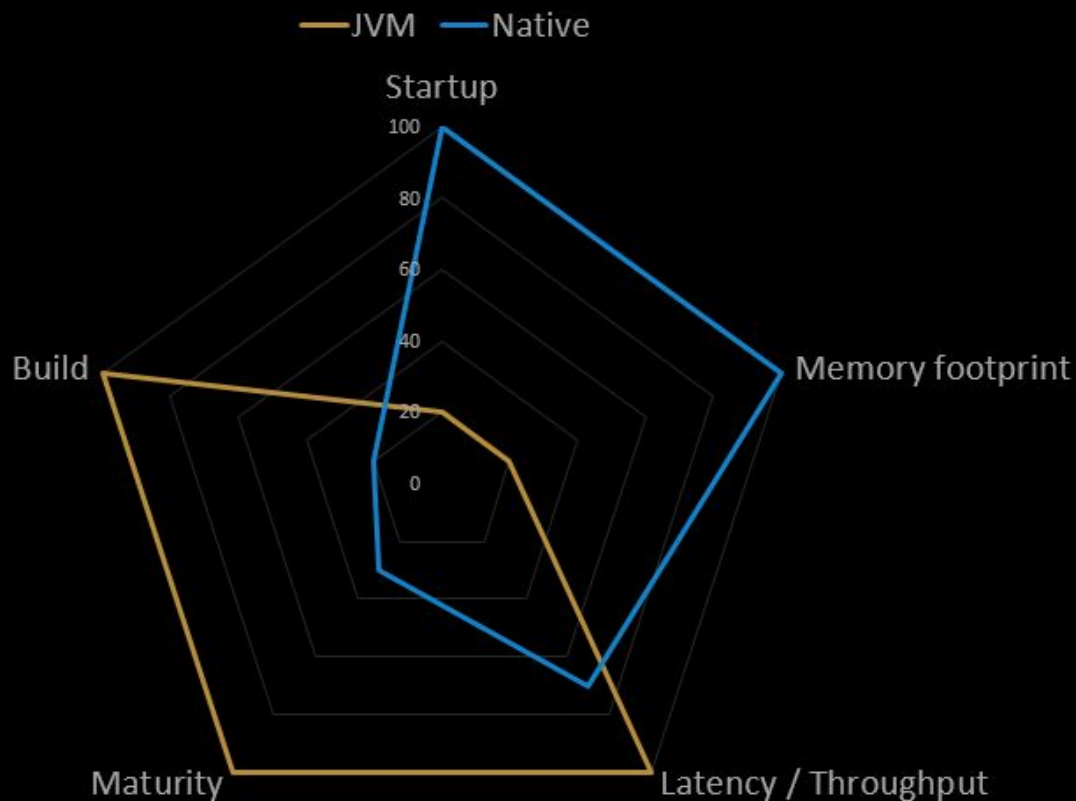
## Reduced memory consumption

4x memory reduction (RSS) on small or medium sized applications

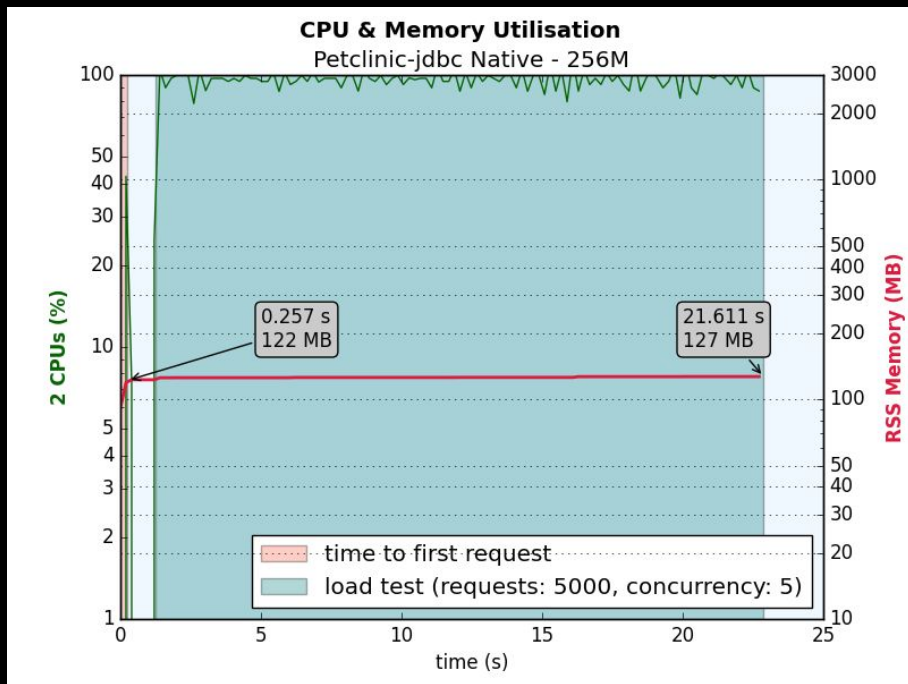
Cheaper cloud instances

Critical with systems splitted into multiple microservices

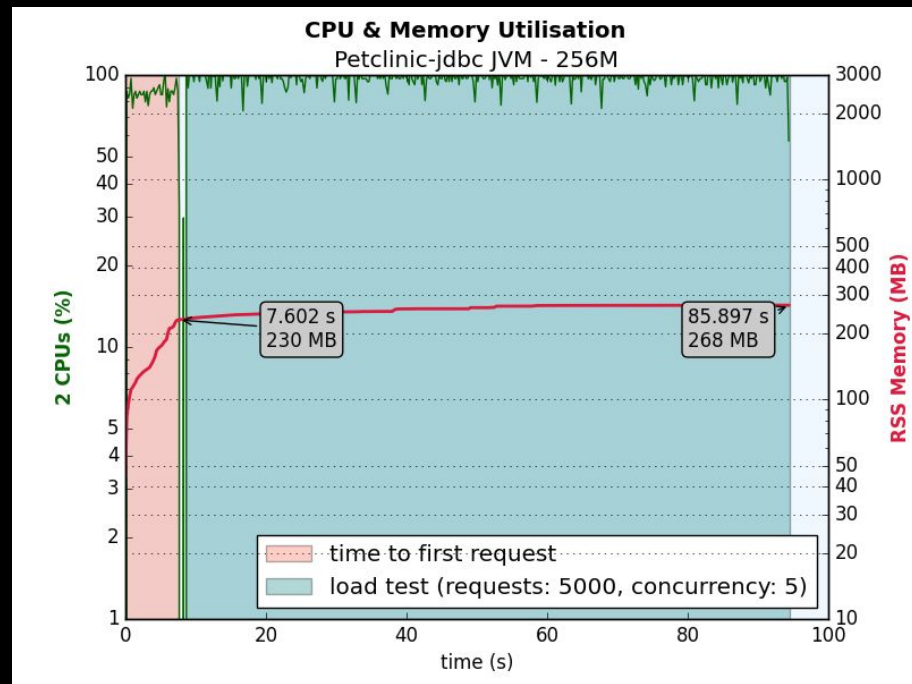
# JVM and Native trade-offs



# CPU and memory profiles



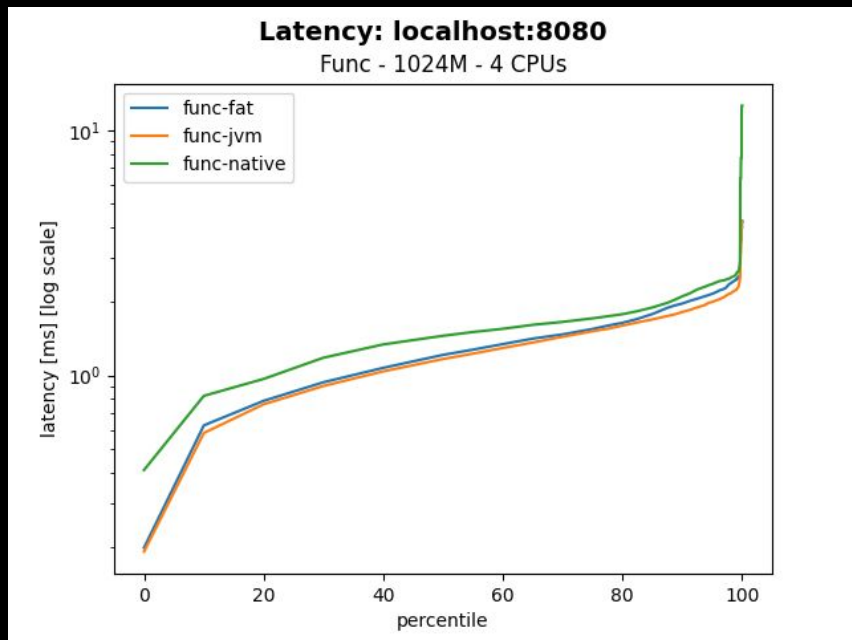
Native



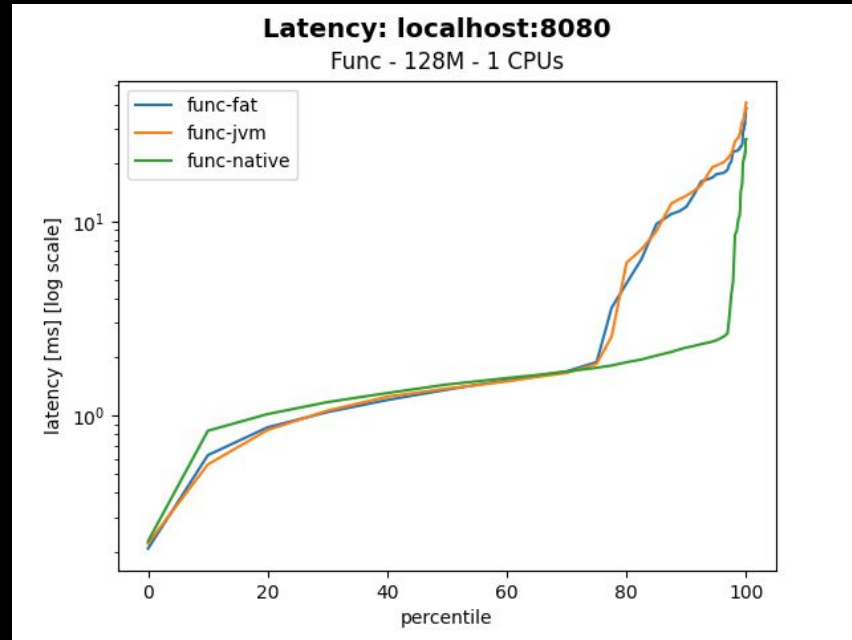
JVM



# Performance versus instance size (and cost)



**Medium  
instance**



**Small  
instance**

# Key differences between JVM and Native

- Static analysis of your application from the main entry point
- Configuration required for:
  - Reflection
  - Proxies
  - Resources
- Classpath fixed at build time
- No class lazy loading
- Optional build time class initialization

**GraalVM native is a source of inspiration for the JVM ecosystem**

# An upcoming Java platform standard with Project Leyden

**Spring Native**

**Spring Native = support for compiling Spring applications to native executables. Unchanged.**

**Spring Native for GraalVM incubates at**  
<https://github.com/spring-projects-experimental/spring-graalvm-native>

# Collaboration between Spring and GraalVM 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



# https://github.com/oracle/graal/labels/spring

oracle / graal

Watch 455 Star 13.3k Fork 963

Code Issues 672 Pull requests 78 Discussions Actions Projects 3 Security

Filters label:spring is:closed Labels 39 Milestones 4 New Issue

Clear current search query, filters, and sorts

22 Open 31 Closed Author Label Projects Milestones Assignee Sort

- Nearly Static Native Images** feature native-image spring #2589 by nebhale was closed on Jul 9 20.2 12
- Deadlock in native-image compilation** bug native-image spring #2553 by fhanik was closed on Jul 14 20.2 9
- Apply access filter to proxy interfaces.** agent native-image oracle-emp spring #2544 by peter-hofer was merged on Jun 15 20.2 1 2
- Spring app build failing when a type isn't on the classpath (with --allow-incomplete-classpath)** bug native-image spring #2529 by aclement was closed on Jul 7 20.2 2
- Agent access filter does not apply to proxy classes** agent native-image spring 1 2



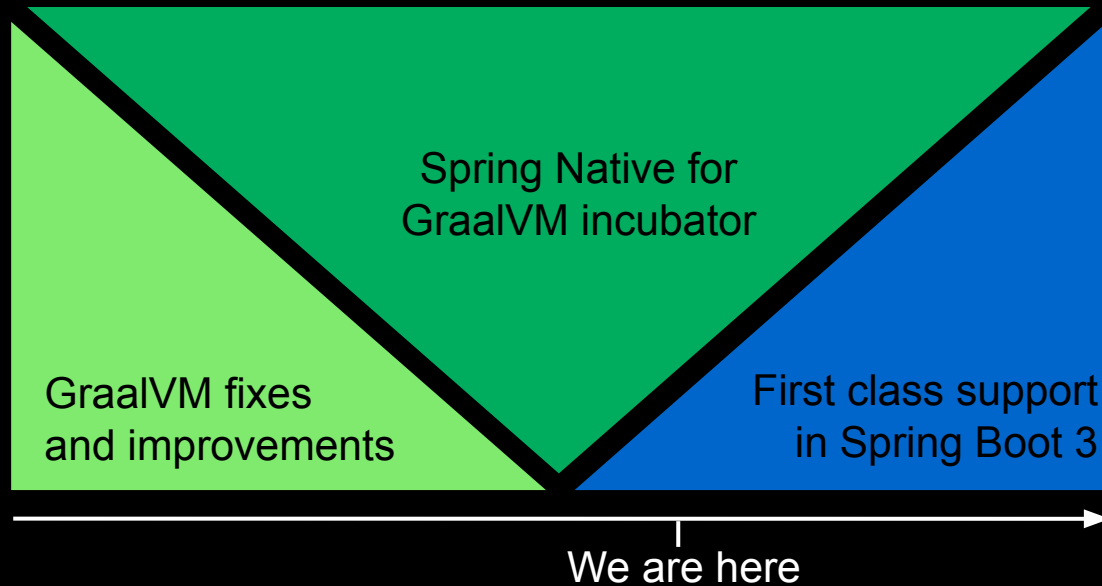
# Leverage Spring Boot container image support

```
> mvn spring-boot:build-image
```

or

```
> gradle bootBuildImage
```

# Timeline



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

spring-projects-experimental / spring-graalvm-native

Unwatch 62 Star 786 Fork 101

Code Issues 72 Pull requests Actions Security Insights Settings

master 1 branch 11 tags

Go to file Add file Code

**sdeleuze** Ignore data-elasticseach flaky sample for now ... 5f01064 3 days ago 1,282 commits

.mvn/wrapper	Upgrade Maven wrappers to 3.6.3	6 months ago
ci	Fix Artifactory credentials in Concourse CI	4 days ago
docker	Add scripts to push images	5 days ago
scripts	Use org.springframework.nativex package	4 days ago
spring-graalvm-native-buil...	Use org.springframework.nativex package	4 days ago
spring-graalvm-native-con...	Use org.springframework.nativex package in tests	4 days ago
spring-graalvm-native-docs	Use org.springframework.nativex package	4 days ago
spring-graalvm-native-feat...	Use org.springframework.nativex package in tests	4 days ago
spring-graalvm-native-ma...	Use org.springframework.nativex package	4 days ago

**About**

Spring Native for GraalVM

[spring.io/blog/2020/11/23/s...](#)

spring spring-boot graalvm serverless native

Readme

Apache-2.0 License

**Releases**

11 tags

Create a new release



# Spring Native for GraalVM

- Incubator for Spring Native
- Analyses the Spring 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

# We have just released 0.8.3

- Spring Boot 2.4.0
- GraalVM 20.3.0
- @SpringBootApplication and @Configuration support without proxyBeanMethods = false

# New flags available in Spring Framework 5.3

```
# -3.6M RSS
```

```
-Dspring.xml.ignore=true
```

```
# -0.5M RSS
```

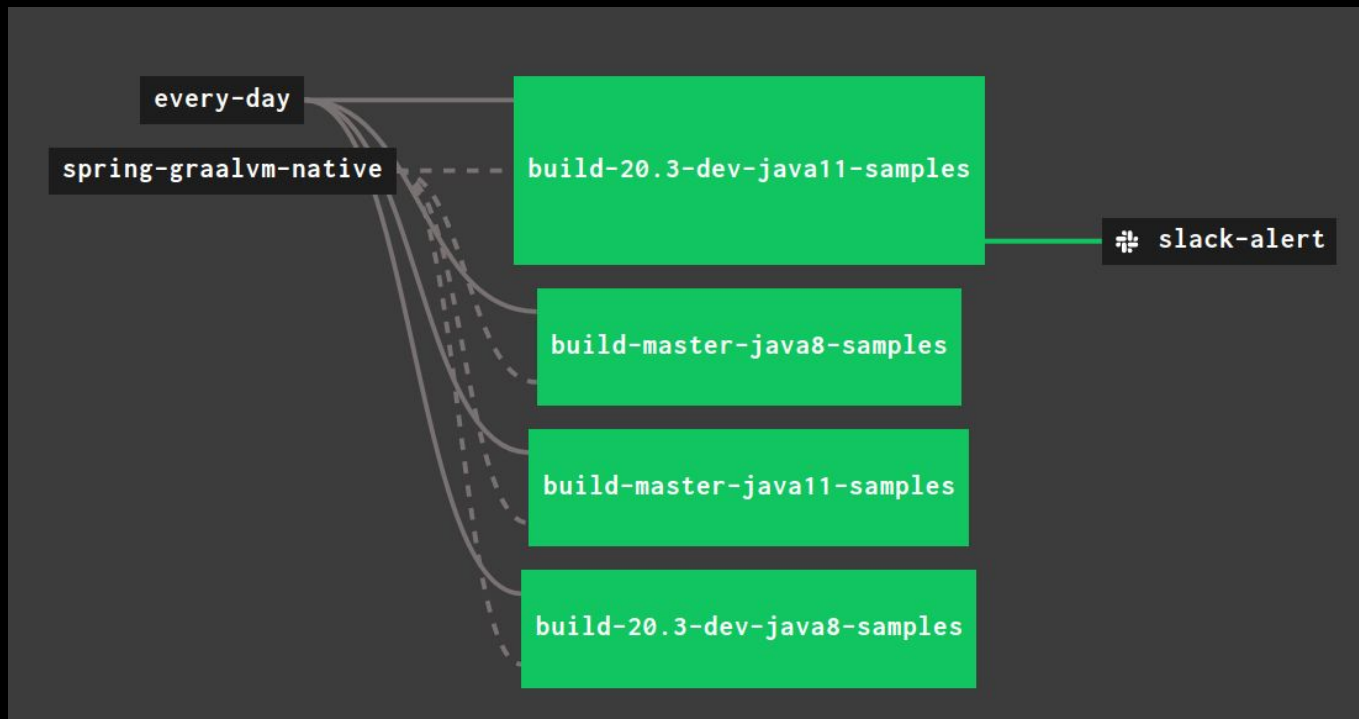
```
-Dspring.spel.ignore=true
```

# A new Tomcat artifact optimized for native apps

```
<!-- -3.5M RSS -->  
<dependency>  
  <groupId>org.apache.tomcat.experimental</groupId>  
  <artifactId>tomcat-embed-programmatic</artifactId>  
  <version>${tomcat.version}</version>  
</dependency>
```



# Our CI checks Java 8/11 x GraalVM 20.3/21.0-dev



**Getting started**

# start.spring.io or an existing project

The screenshot shows the Spring Initializr web application interface. At the top left is a hamburger menu icon. The main header features the Spring logo and the text "spring initializr". On the top right, there are icons for settings and a dark mode toggle. The interface is divided into several sections:

- Project:** Includes radio buttons for "Maven Project" (selected), "Gradle Project", "Spring Boot" (selected), and "Other".
- Language:** Includes radio buttons for "Java" (selected), "Kotlin", and "Groovy".
- Spring Boot:** Includes radio buttons for versions "2.4.1 (SNAPSHOT)", "2.4.0" (selected), "2.3.7 (SNAPSHOT)", "2.3.6", "2.2.12 (SNAPSHOT)", and "2.2.11".
- Project Metadata:** A form with input fields for "Group" (com.example), "Artifact" (demo), "Name" (demo), "Description" (Demo project for Spring Boot), and "Package name" (com.example.demo).
- Dependencies:** A section with a button "ADD DEPENDENCIES... CTRL + B". It lists "Spring Web" with a "WEB" tag and "JDBC API" with an "SQL" tag, each with a brief description.

At the bottom, there are three buttons: "GENERATE CTRL + G", "EXPLORE CTRL + SPACE", and "SHARE...". In the bottom left corner, there are icons for GitHub and Twitter.

# Configure Maven plugin

```
<plugin>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-maven-plugin</artifactId>
  <configuration>
    <image>
      <env>
        <BP_BOOT_NATIVE_IMAGE>true</BP_BOOT_NATIVE_IMAGE>
      </env>
    </image>
  </configuration>
</plugin>
```

# Add the spring-graalvm-native dependency

```
<dependency>  
  <groupId>org.springframework.experimental</groupId>  
  <artifactId>spring-graalvm-native</artifactId>  
  <version>0.8.3</version>  
</dependency>
```

# Build the native application

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

- No need for a GraalVM native local install
- Build happens in a container
- And produces a container



**You can also build a native executable without using containers**



# Configure Maven plugin in a native profile

```
<plugin>
  <groupId>org.graalvm.nativeimage</groupId>
  <artifactId>native-image-maven-plugin</artifactId>
  <version>20.3.0</version>
  <configuration>
    <mainClass>com.example.demo.DemoApplication</mainClass>
  </configuration>
  <executions>
    <execution>
      <goals>
        <goal>native-image</goal>
      </goals>
      <phase>package</phase>
    </execution>
  </executions>
</plugin>
```

# Build the native application

```
> mvn -Pnative clean package
```

```
Total time: 60 s
```

- Need for a GraalVM native local install
- Build happens directly on your host (Linux, Mac and Windows are supported)
- Produces a native executable
- No cross compilation



Demo

# How fast is your PetClinic?

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

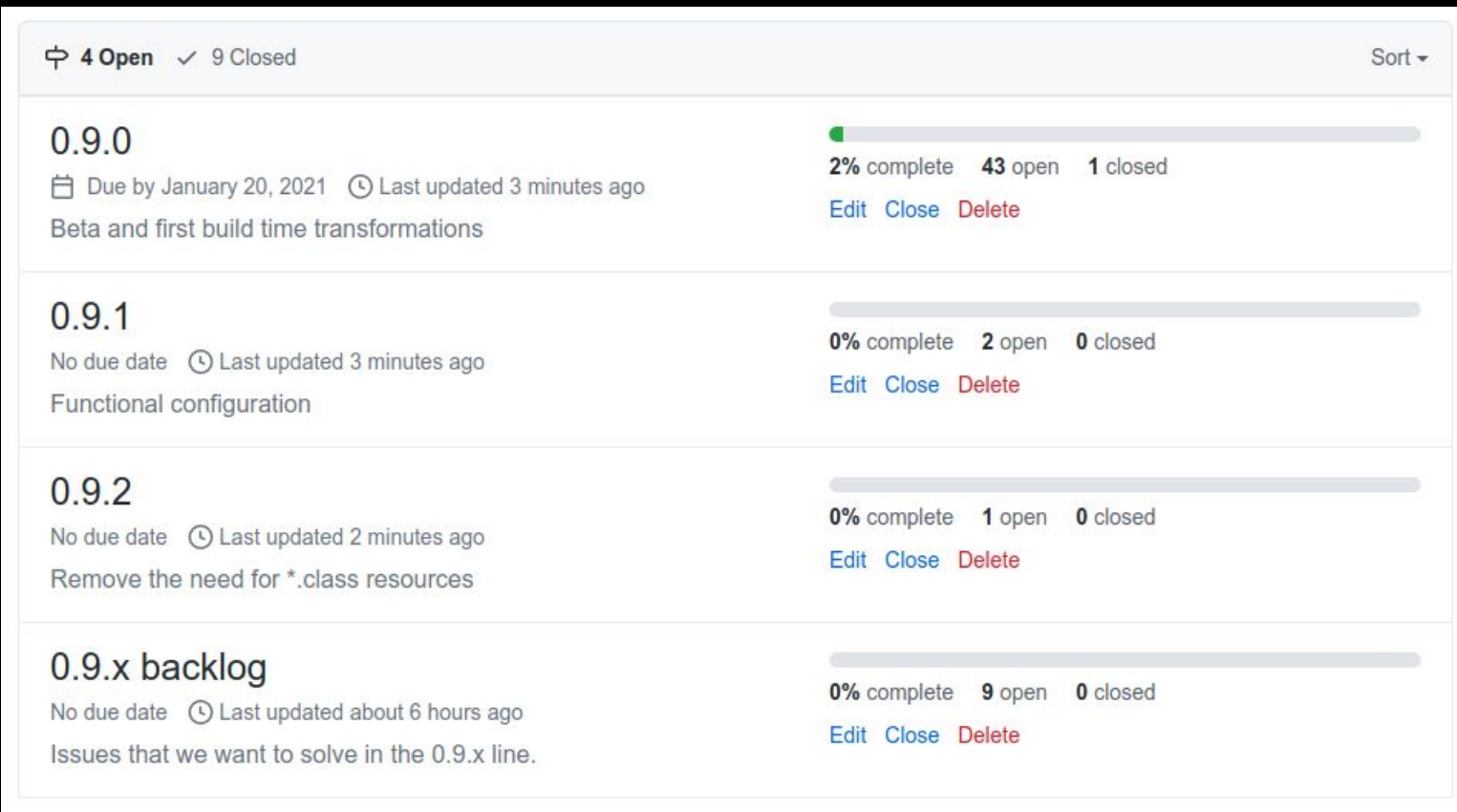
# Ongoing work on footprint

Key metrics: build time, size of resultant image, and runtime memory footprint

Sample	2019	2020
commandlinerunner	Build: 90s Exec. size: 48M Memory(RSS): 29M	Build: 50s -45% Exec. size: 22M -55% Memory(RSS): 24M -17%
webflux-netty	Build: 193s Exec. size: 81M Memory(RSS): 95M	Build: 79s -60% Exec. size: 43M -47% Memory(RSS): 47M -50%
webmvc-tomcat	Build: 203s Exec size: 105M Memory(RSS): 70M	Build: 67s -67% Exec. size: 42M -60% Memory(RSS): 51M -27%

**The road ahead**

# Rodmap





# 0.9.0 (early 2021)

- **First release with beta status**
  - Spring Boot 2.4.0 and GraalVM 20.3 baseline
  - Subset of starters supported
  - Breaking change will happen (with upgrade paths)
- Wider support
  - Spring Security
  - Spring Batch
- Introduction of build time transformations
  - Programmatic Spring factories
  - GraalVM feature to native configuration generator
  - Automatic native configurations of `src/main/resources/`\*

## 0.9.X

- **Switch at some point to Spring Framework 6 / Boot 3 milestones**
- GraalVM 21.x baseline
- More build time transformations
- Testing
- Focus on maintainability, footprint and build time reduction

# Build time transformations

☐ ⓘ 9 Open ✓ 0 Closed Author ▾ Label ▾ Projects ▾ Milestones ▾ Assignee ▾ Sort ▾

- ☐ ⓘ **Reduce the need for reflection by using functional configuration**  
type: build-time-transformation  
#386 opened 11 minutes ago by sdeleuze ↻ 0.9.1
- ☐ ⓘ **Automatically configure resources from src/main/resources** type: build-time-transformation 3  
#374 opened 2 days ago by halimpuckjava ↻ 0.9.0
- ☐ ⓘ **Create proxies at build time** type: build-time-transformation  
#356 opened 12 days ago by sdeleuze ↻ 0.9.x backlog
- ☐ ⓘ **Introduce the build time transformation infrastructure** type: build-time-transformation 3  
#346 opened 22 days ago by sdeleuze ↻ 0.9.0
- ☐ ⓘ **Generate a programmatic version of spring.factories at build time**  
type: build-time-transformation  
#345 opened 22 days ago by sdeleuze ↻ 0.9.0
- ☐ ⓘ **Turn the GraalVM feature into a pure build time transformation generating JSON configs** type: build-time-transformation  
#344 opened 22 days ago by sdeleuze ↻ 0.9.0
- ☐ ⓘ **Remove the need for \*.class resources** type: build-time-transformation  
#341 opened 22 days ago by sdeleuze ↻ 0.9.2
- ☐ ⓘ **Automatically set flags** type: build-time-transformation type: optimization  
#225 opened on Aug 13 by sdeleuze ↻ 0.9.x backlog
- ☐ ⓘ **Experiment about build time compilation of SpEL expressions** type: build-time-transformation  
type: feature  
#167 opened on Jun 8 by sdeleuze ↻ 0.9.x backlog

# Native applications functional configuration

Sample	webmvc-tomcat	webmvc-functional With build time transformation to functional configuration
Regular Spring Boot application with Spring MVC, Tomcat and Jackson	Build: 67s Exec. size: 42M Memory(RSS): 51M Startup time: 0.06s	Build: 60s -10% Exec. size: 37M -12% Memory(RSS): 26M -49% Startup time: 0.02s -66%

**Spring Boot 3, based on Spring Framework 6, is expected to provide first-class support for native application deployment, as well as an optimized footprint on the JVM.**

# Takeaways

- Spring Native is great to build lightweight containers
- Try it now with spring-graalvm-native 0.8.3
- Contribute support for your favorite starter
- Beta early 2021 with spring-graalvm-native 0.9
- Wider support and smaller footprint during 0.9.x releases
- Upcoming first-class native support in Spring Boot 3 / Framework 6

# Stay Connected.

<https://github.com/spring-projects-experimental/spring-graalvm-native>

<https://spring.io/blog>

@sdeleuze on  for fresh news