The Path Towards Spring Boot Native Applications

Sébastien Deleuze
@sdeleuze
VMware

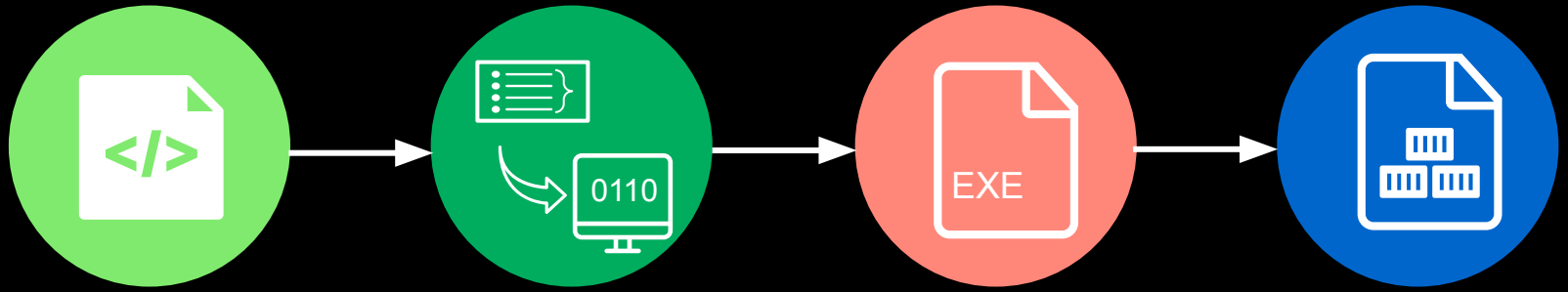
Safe Harbor Statement

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 or 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

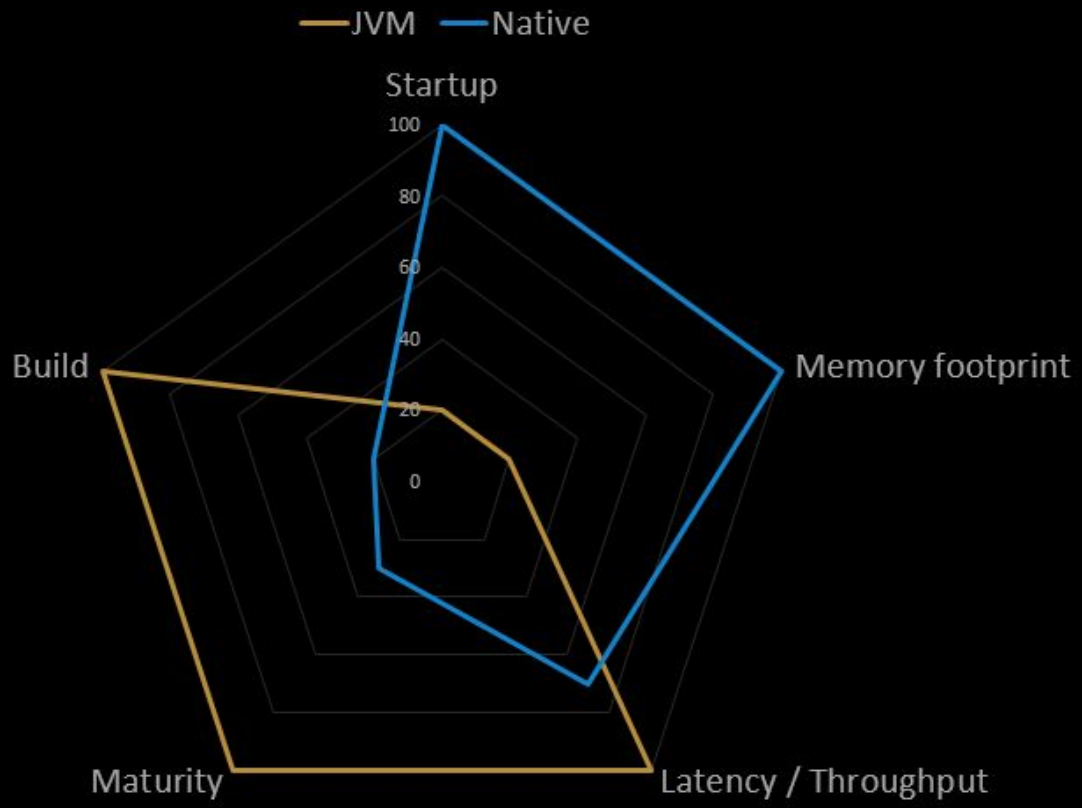
Reduced memory consumption

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

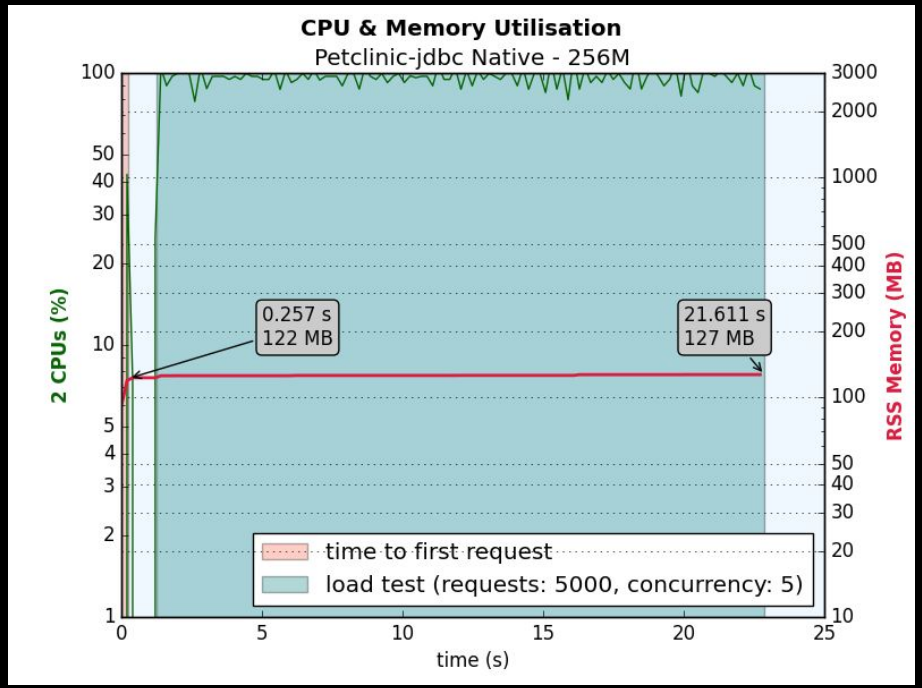
Cheaper cloud instances

Critical with systems splitted into multiple microservices

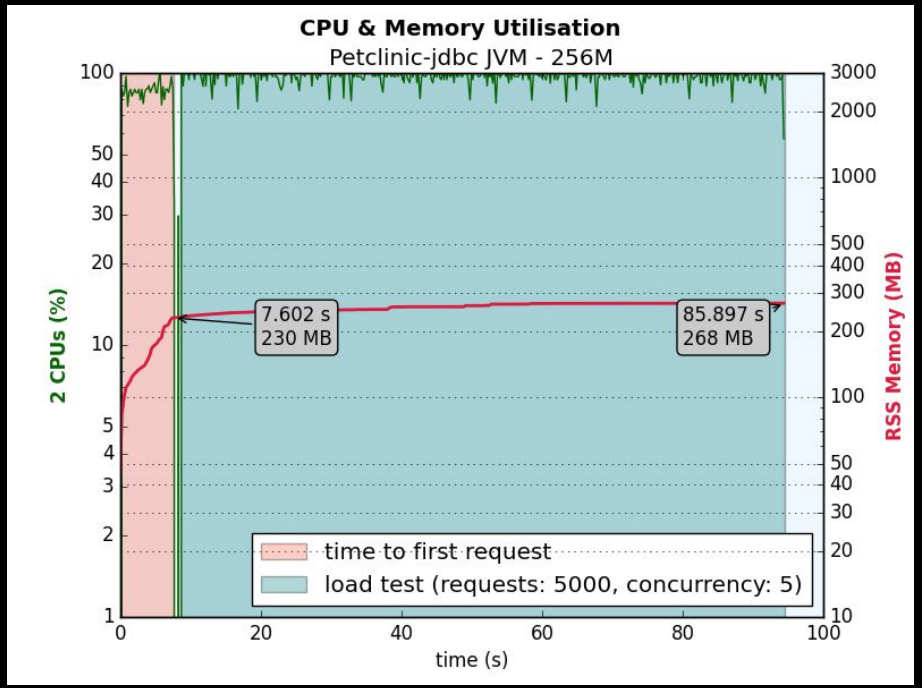
JVM and Native trade-offs



CPU and memory profiles

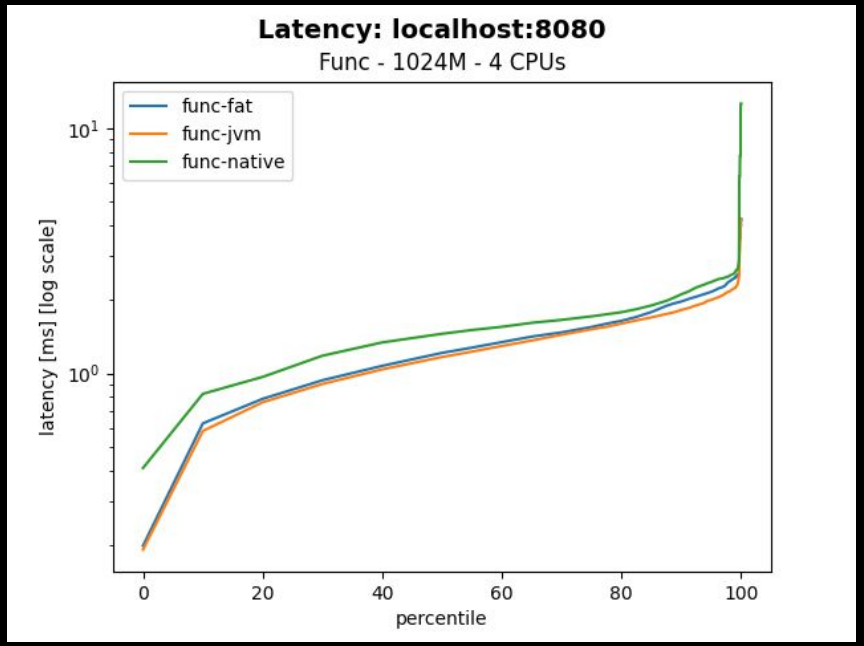


Native

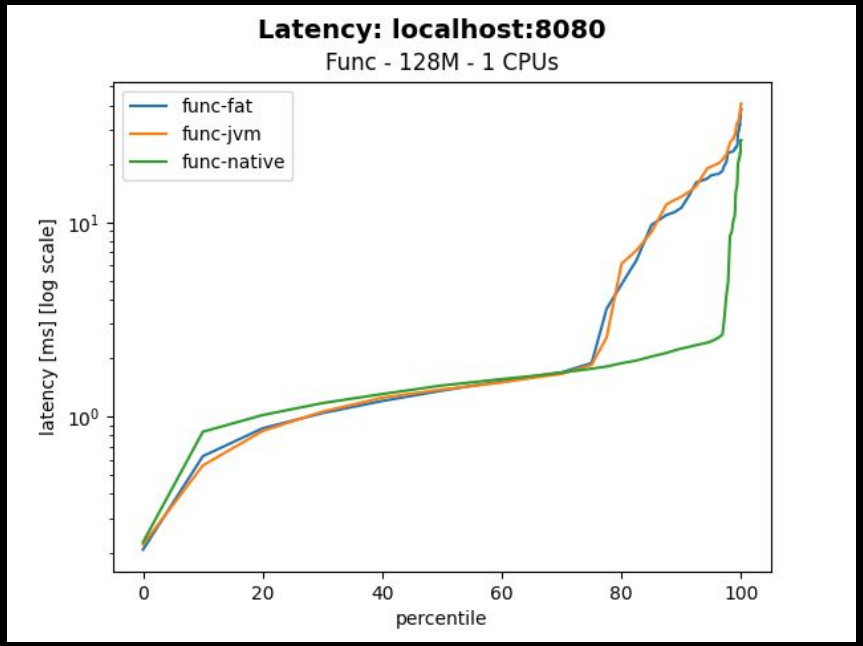


JVM

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

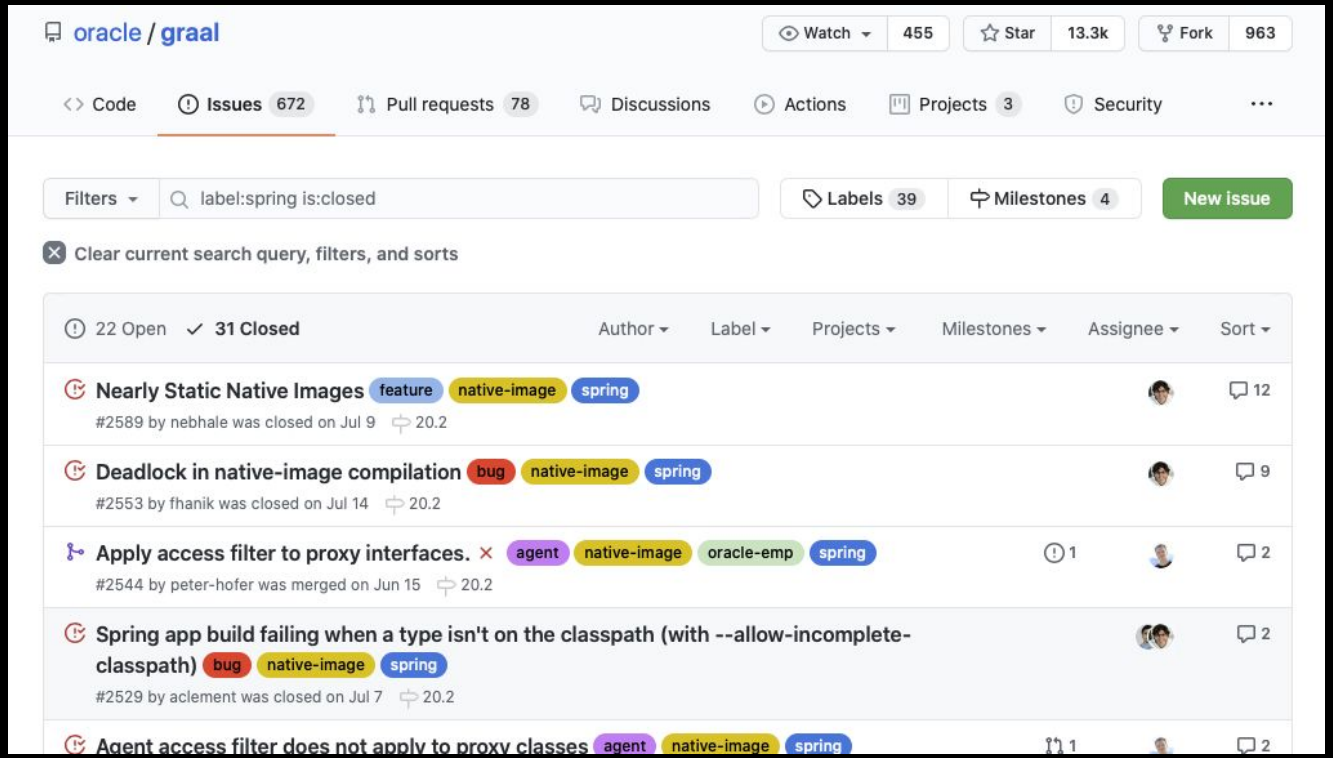
https://github.com/spring-projects-experimental/spring-graalvm-native

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

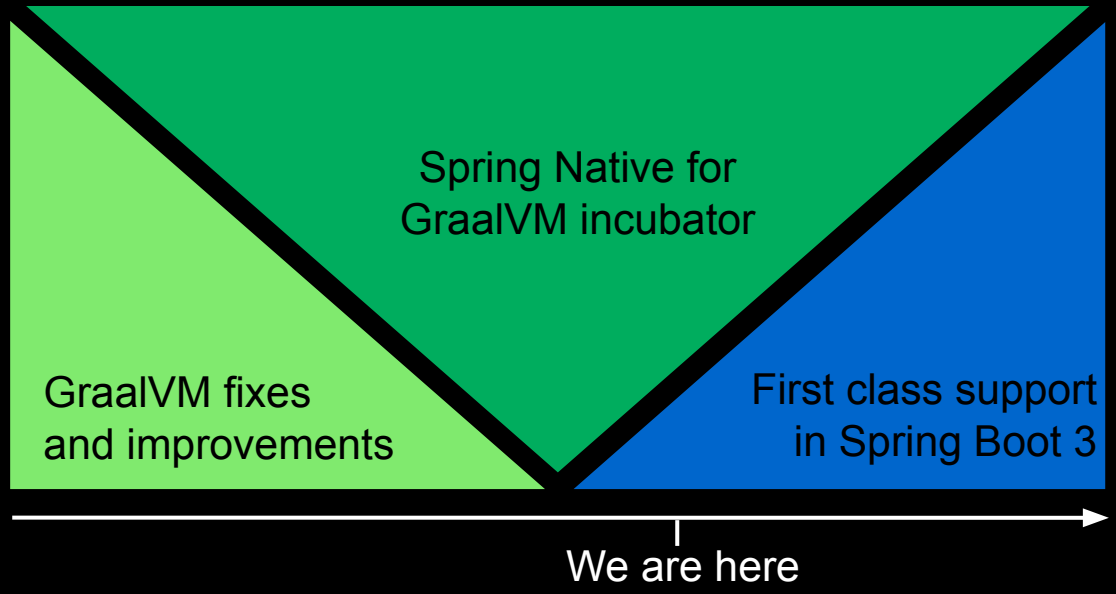
https://github.com/oracle/graal/labels/spring



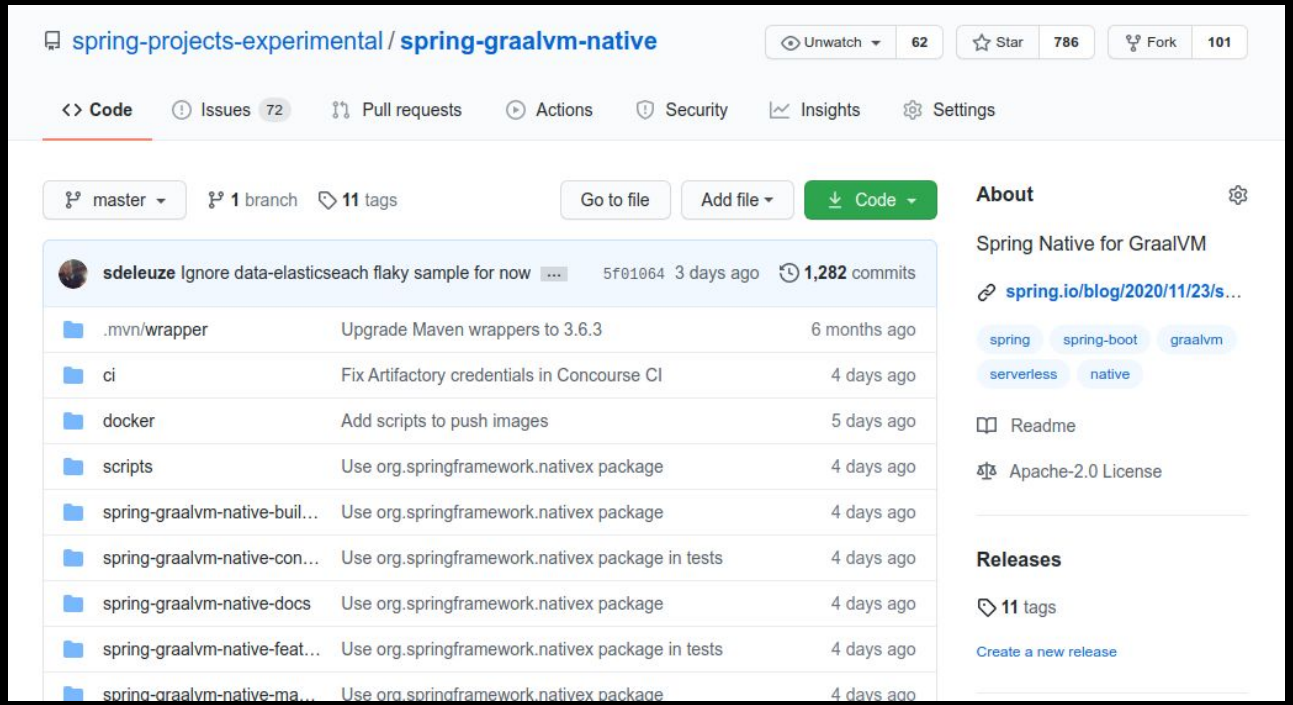
Leverage Spring Boot container image support

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

                or

> gradle bootBuildImage
```

Timeline


Spring Native for
GraalVM incubator
GraalVM fixes
and improvements
First class support
in Spring Boot 3
We are here

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



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

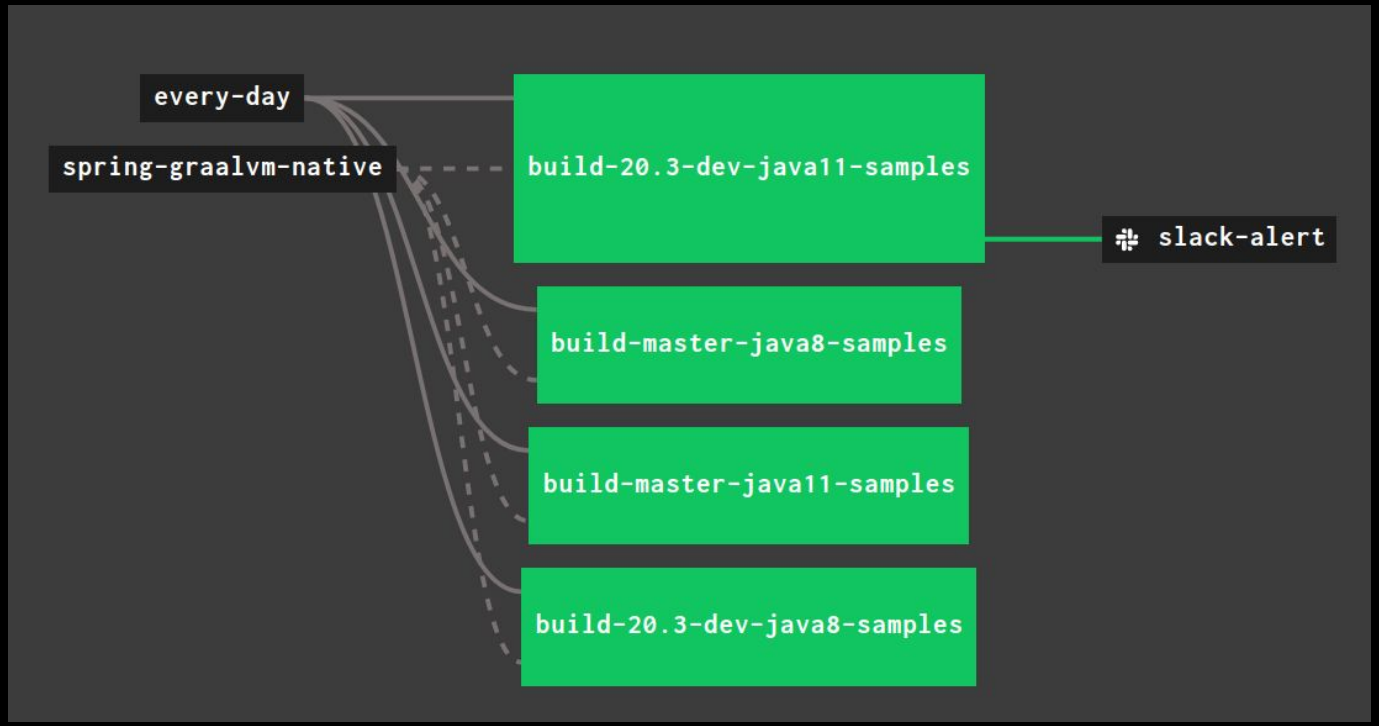
New flags available in Spring Framework 5.3

```
# -3.6M RSS
-Dspring.xml.ignore=true

# -0.5M RSS
-Dspring.spel.ignore=true
```

A new Tomcat artifact optimized for native apps

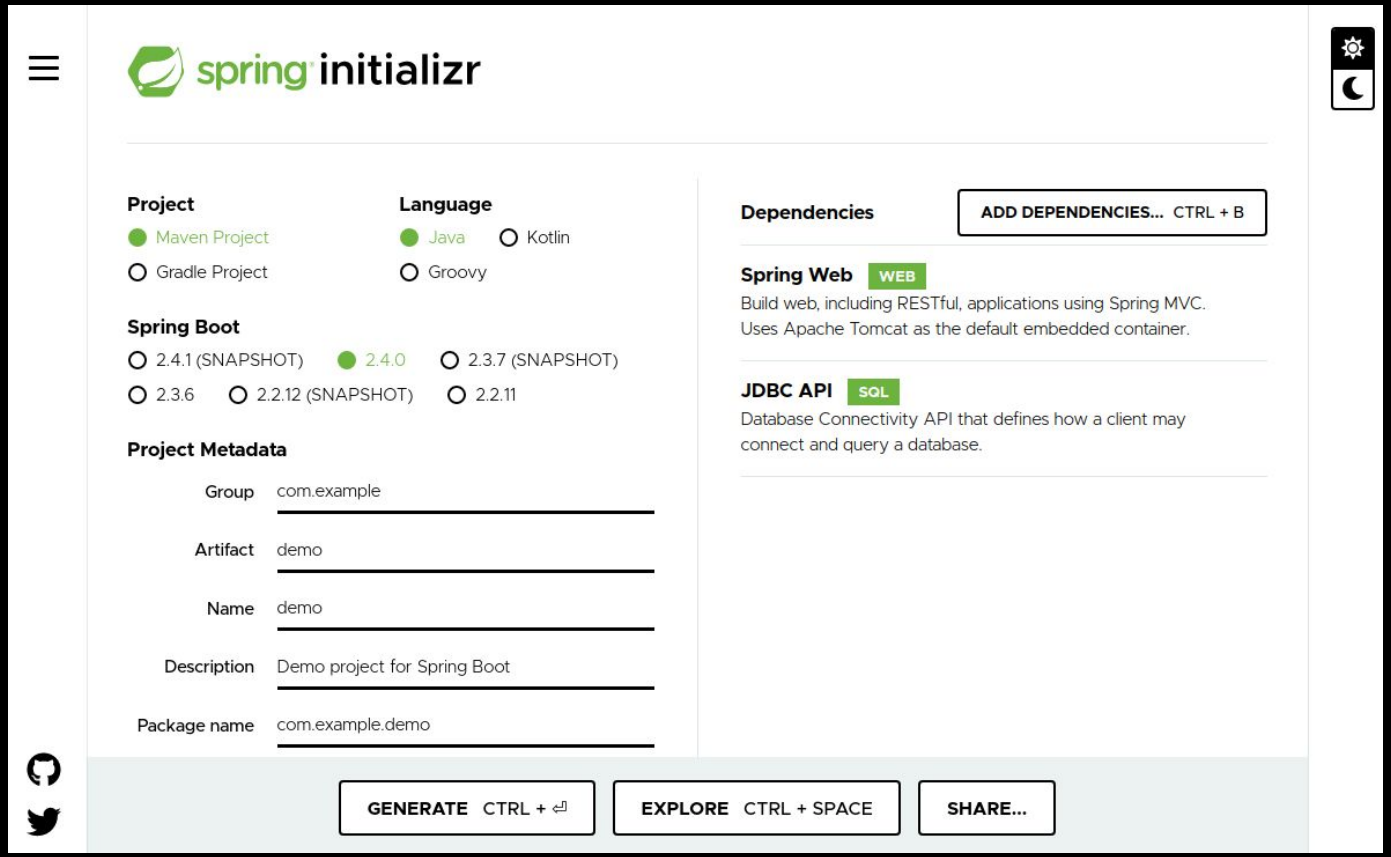
```
<!-- -3.5M RSS -->
<dependency>
  <groupId>org.apache.tomcat.experimental</groupId>
  <artifactId>tomcat-embed-programmatic</artifactId>
  <version>${tomcat.version}</version>
</dependency>
```

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



Getting started

start.spring.io or an existing project



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

Run the native application

```
> 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)
```

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

Run the native application

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


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

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

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%

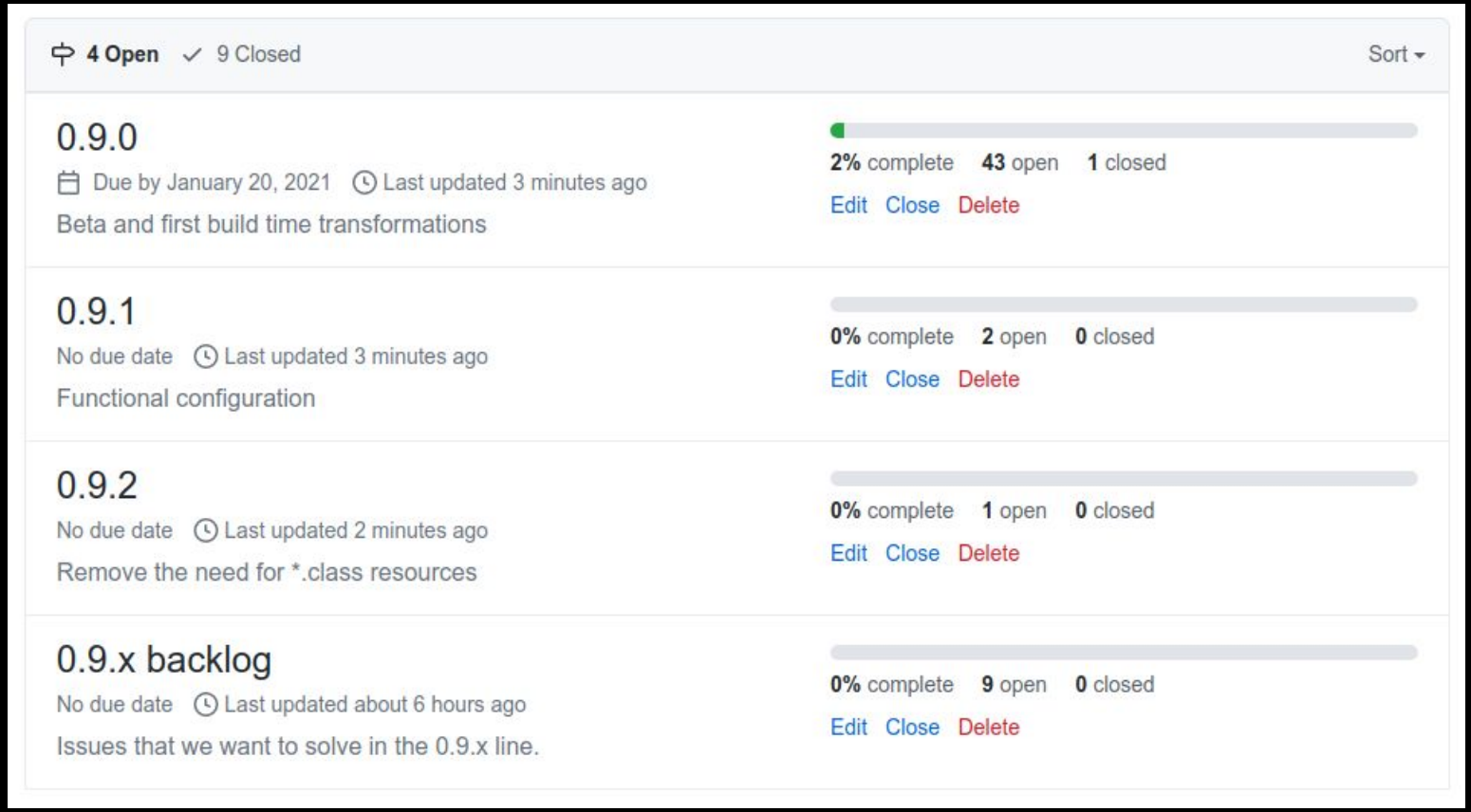
Ongoing work on footprint

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


4 Open
9 Closed
Sort
0.9.0
Due by January 20, 2021
Last updated 3 minutes ago
Beta and first build time transformations
2% complete 43 open 1 closed
Edit Close Delete
0.9.1
No due date
Last updated 3 minutes ago
Functional configuration
0% complete 2 open 0 closed
Edit Close Delete
0.9.2
No due date
Last updated 2 minutes ago
Remove the need for *.class resources
0% complete 1 open 0 closed
Edit Close Delete
0.9.x backlog
No due date
Last updated about 6 hours ago
Issues that we want to solve in the 0.9.x line.
0% complete 9 open 0 closed
Edit Close Delete

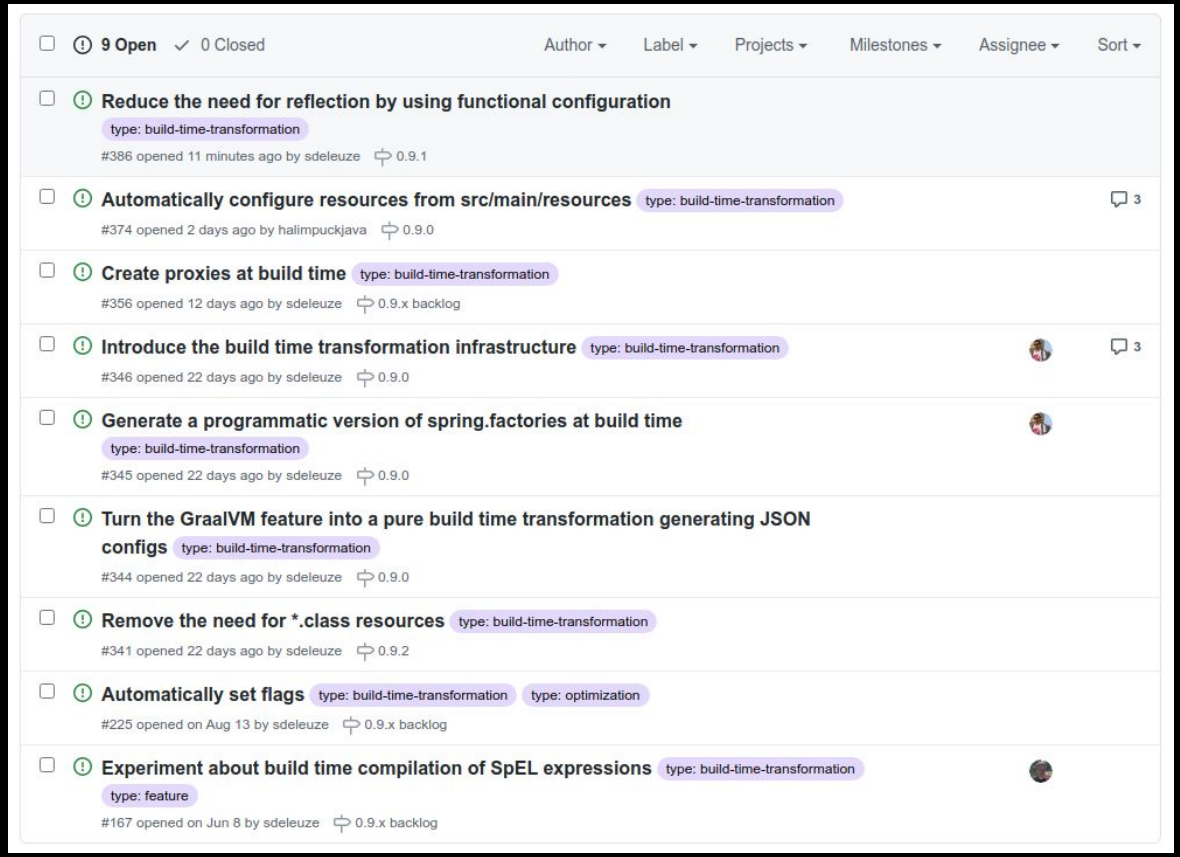
0.9.0 (early 2021)

- **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



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