ORACLE

Introduction to GraalVM

Best practices for Building and Operating Java Apps

Faster. Smarter. Leaner.

Elvadas Nono
Solution Architect , Oracle,
@nelvadas

Nov 9th 2023

Java Ecosystem

OpenJDK Develop latest Java reference implementations

GraalVM Innovative compiler technologies

Java Community Process 1000+ participants developing technical specifications for Java

350+ groups

Java magazine Over 250,000 subscribers

ORACLE

300+ members

Java Champions

ORACLE Academy Educating the next generation

1M certifications

ORACLE UNIVERSITY

JDuchess Program
550+ organizations in 60 countries
focused on women-in-tech

GraalVM Features

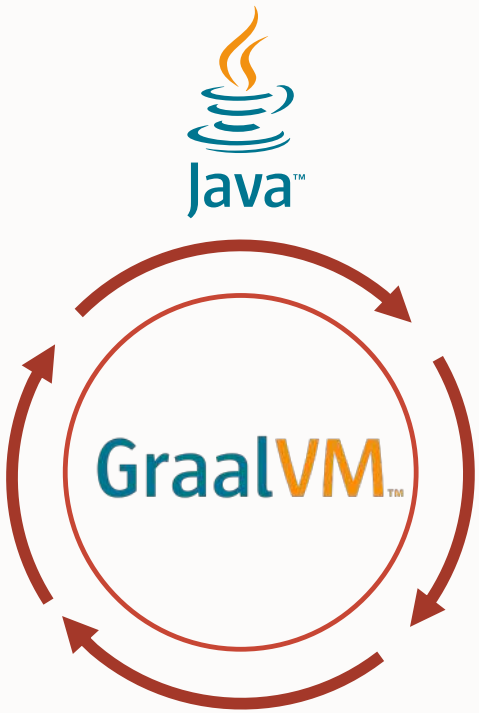
A high-performance optimizing Just-in-Time (JIT) compiler

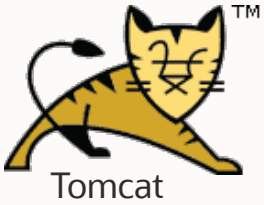
An Ahead-of-Time (AOT) “native image” compiler

Multilingual Virtual Machine

GraalVM Enterprise : High Performance JIT Compiler

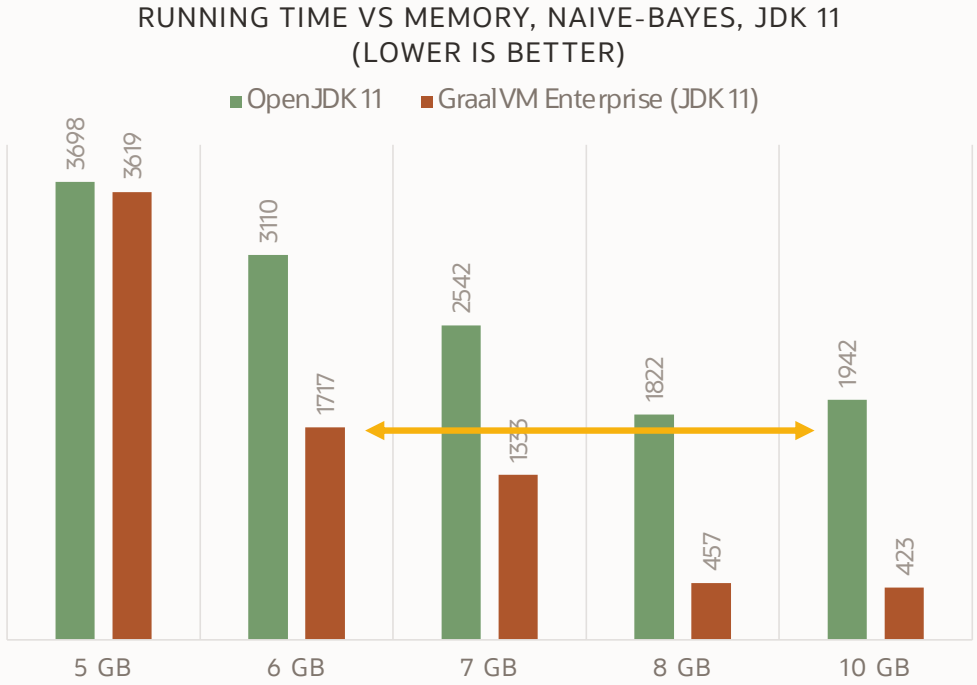
GraalVM Enterprise JIT Compiler—Ideal for traditional Java workloads





GraalVM Enterprise—Leaner

Higher performance with less memory

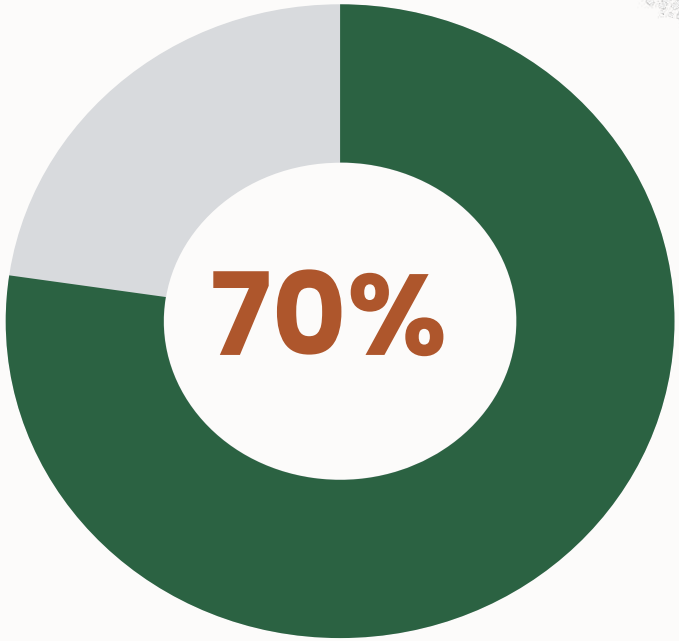


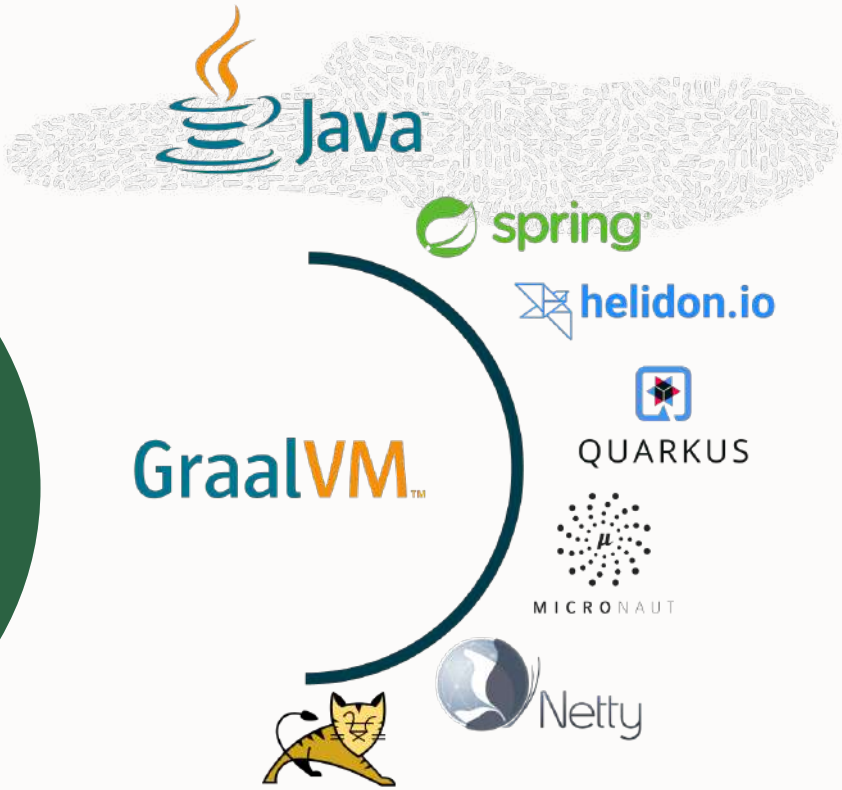
On the Renaissance “naïve-bayes” benchmark, GraalVM Enterprise outperforms OpenJDK 11—regardless of the amount of available RAM.

GraalVM Enterprise with 6GB outperforms OpenJDK with 10GB

Source: https://blogs.oracle.com/graalvm/apache-spark%e2%80%94lightning-fast-on-graalvm-enterprise

Java and Container





70% Java Applications reported to New Relic are running in containers, **New Relic State of the Java Ecosystem – April 2022**

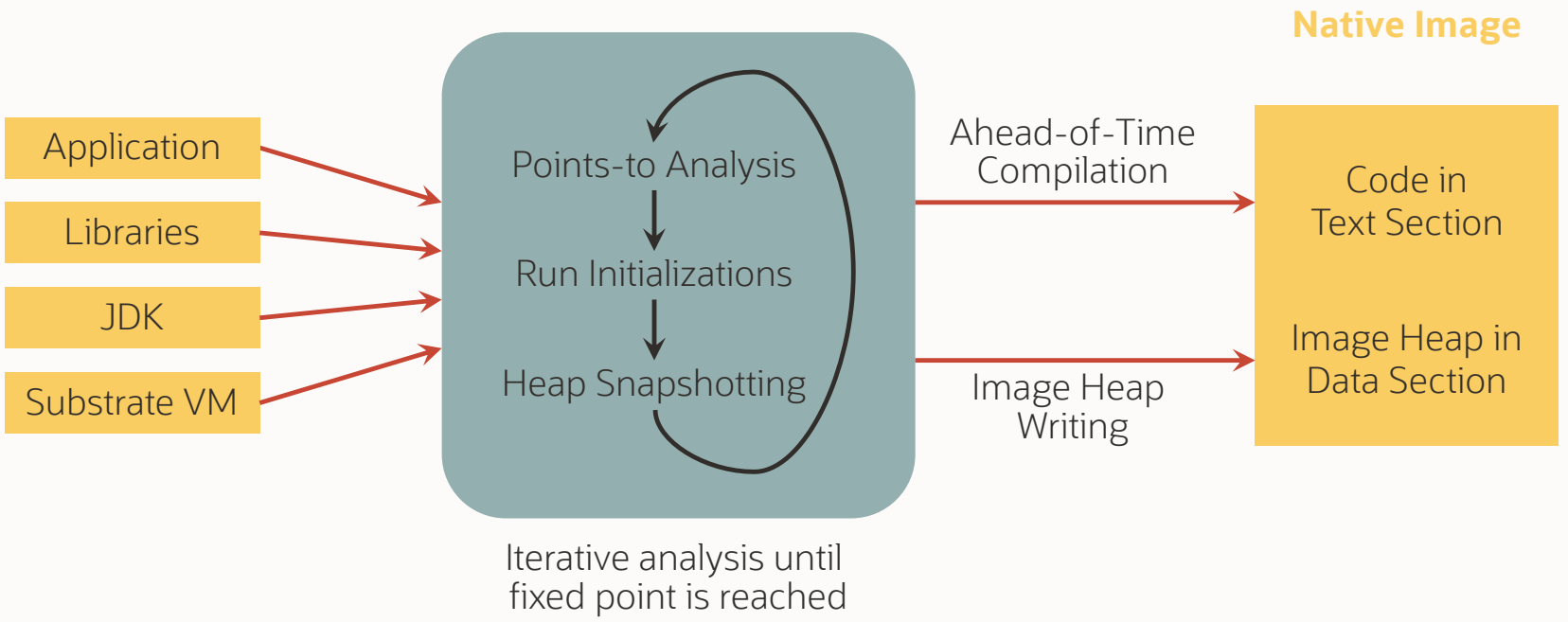
GraalVM Enterprise Native Image—Ahead-of-time compiler & runtime

Microservices and Containers

Up to 5x less memory
100x faster startup

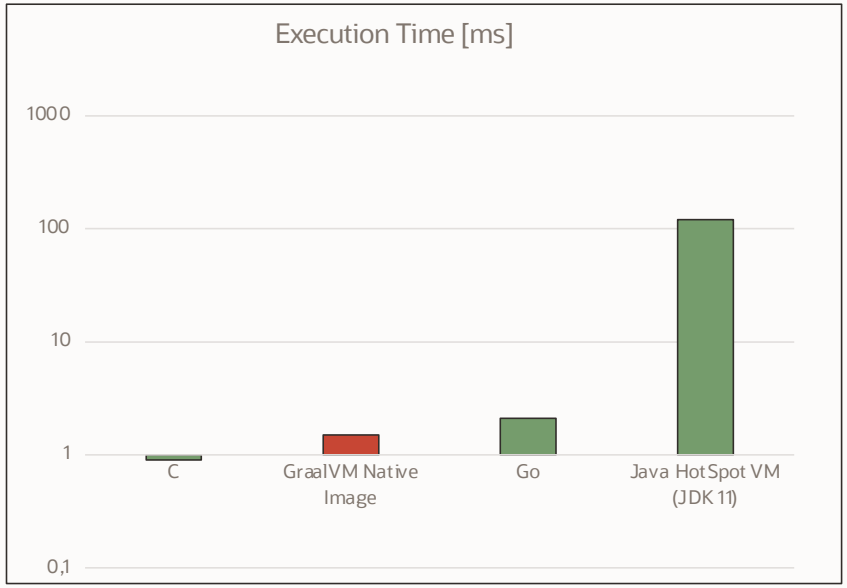
GraalVM Enterprise Native Image—Ahead-of-time compiler & runtime

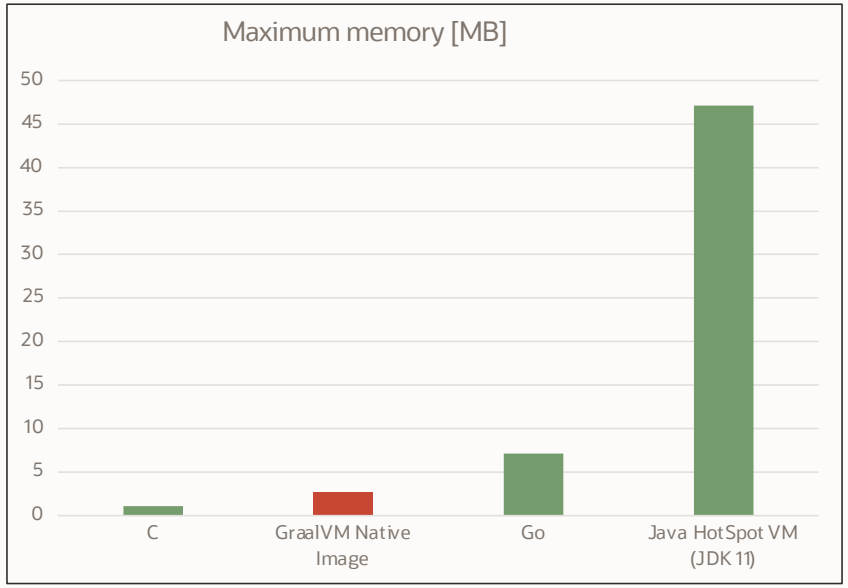
How it works?



GraalVM Enterprise Native Image—Java productivity with C-like performance

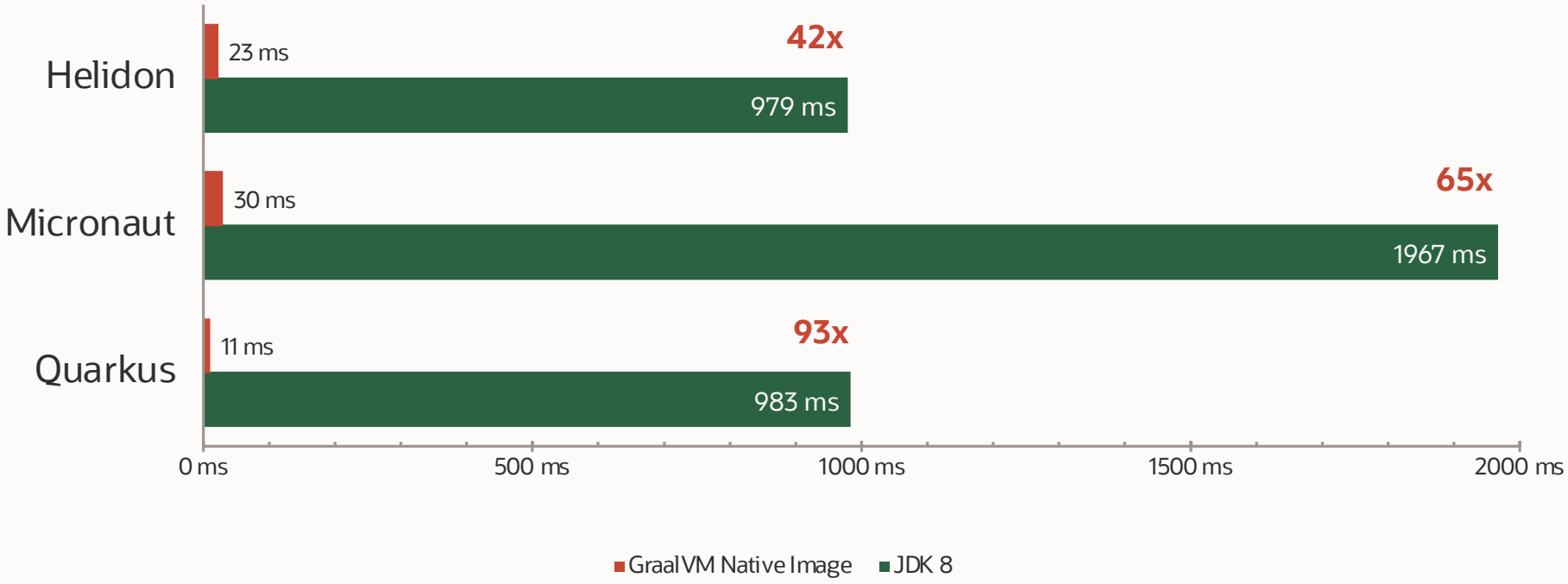
Microservices and Containers



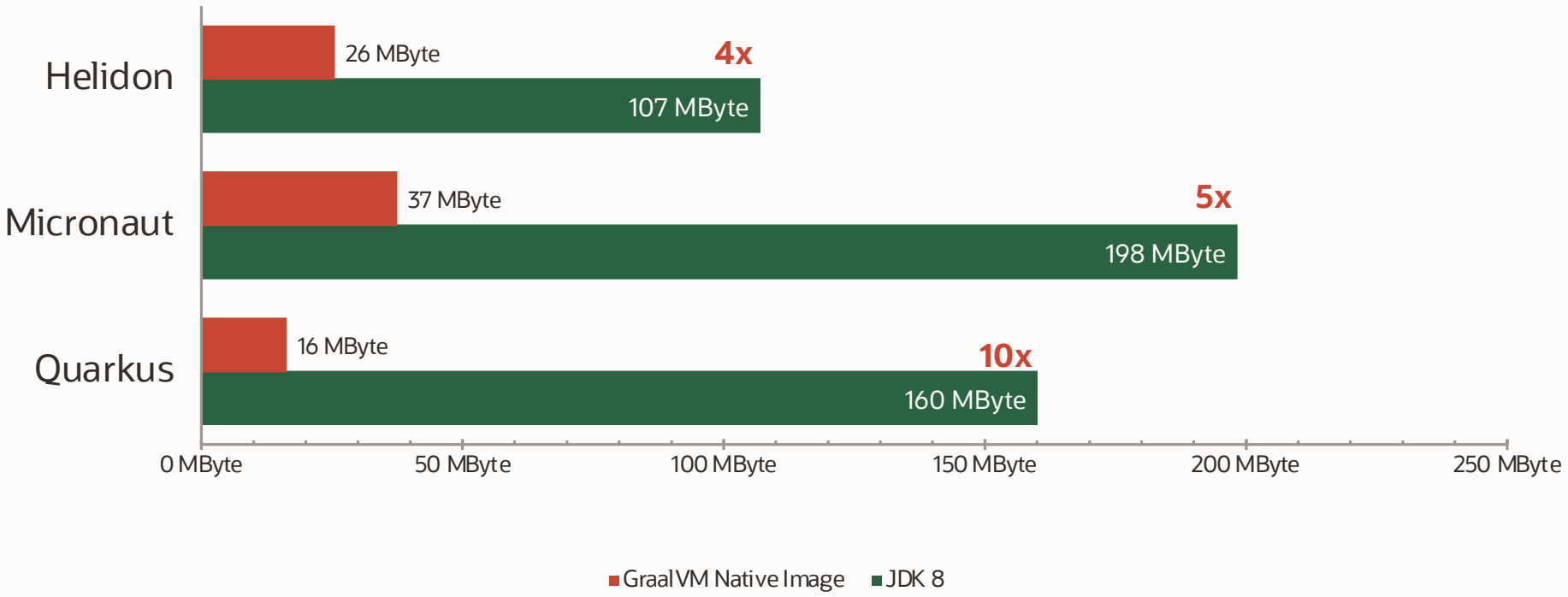


Lower compute requirements and faster execution reduces infrastructure/cloud costs

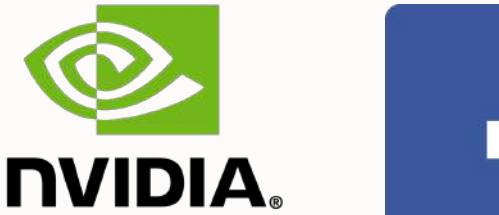
Microservices – Faster Startups

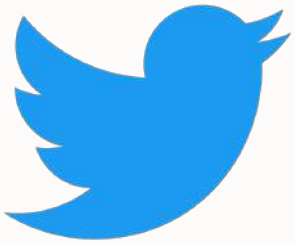


Microservices – Memory Footprint



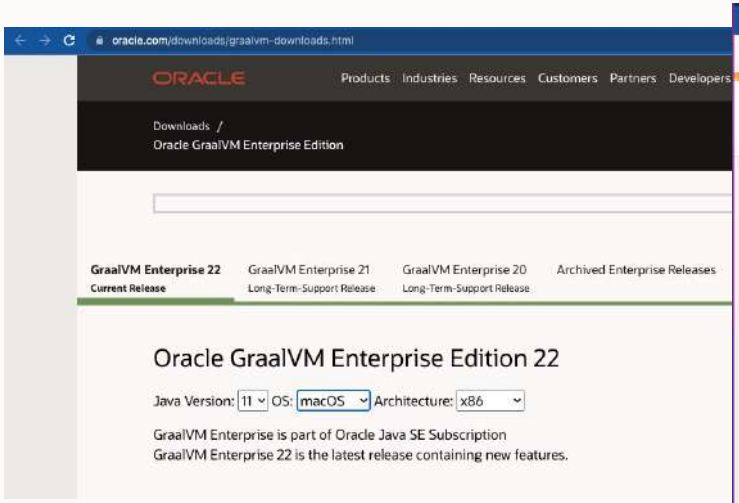
Some GraalVM References

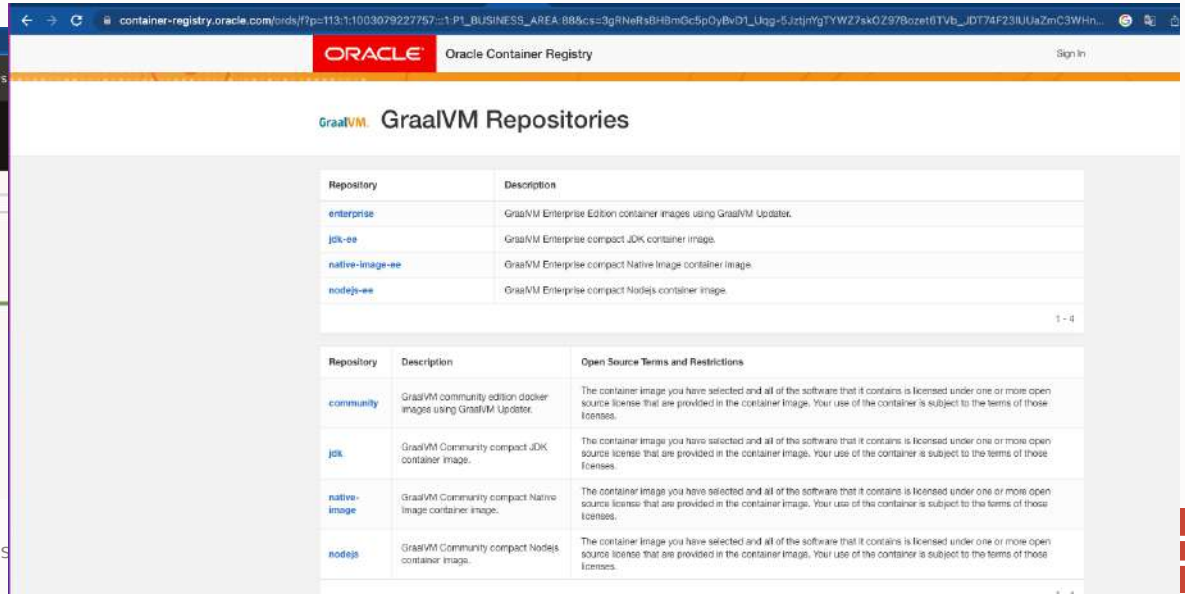




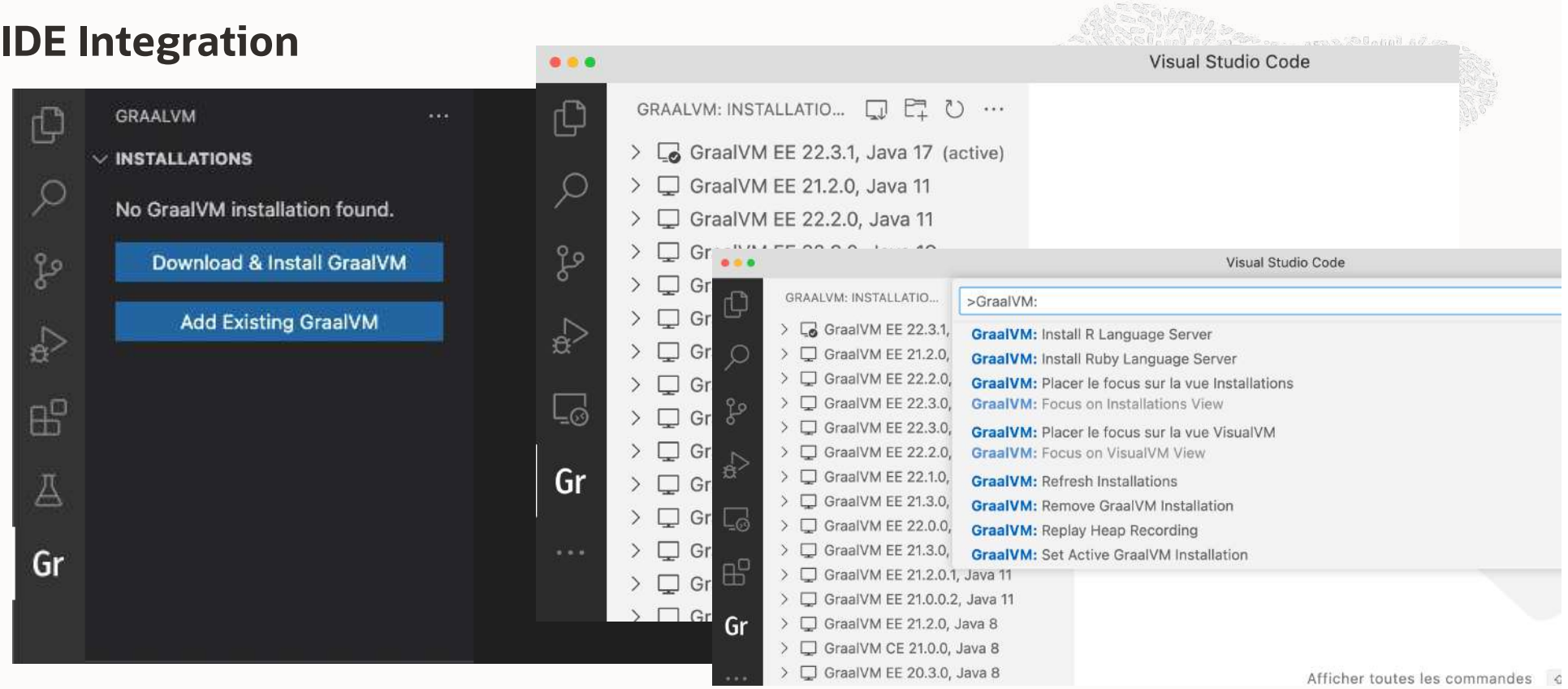
GraalVM Enterprise : Downloads

Format	Links
Packages	https://www.oracle.com/downloads/graalvm-downloads.html
Containers Images	**https://container-registry.oracle.com/**



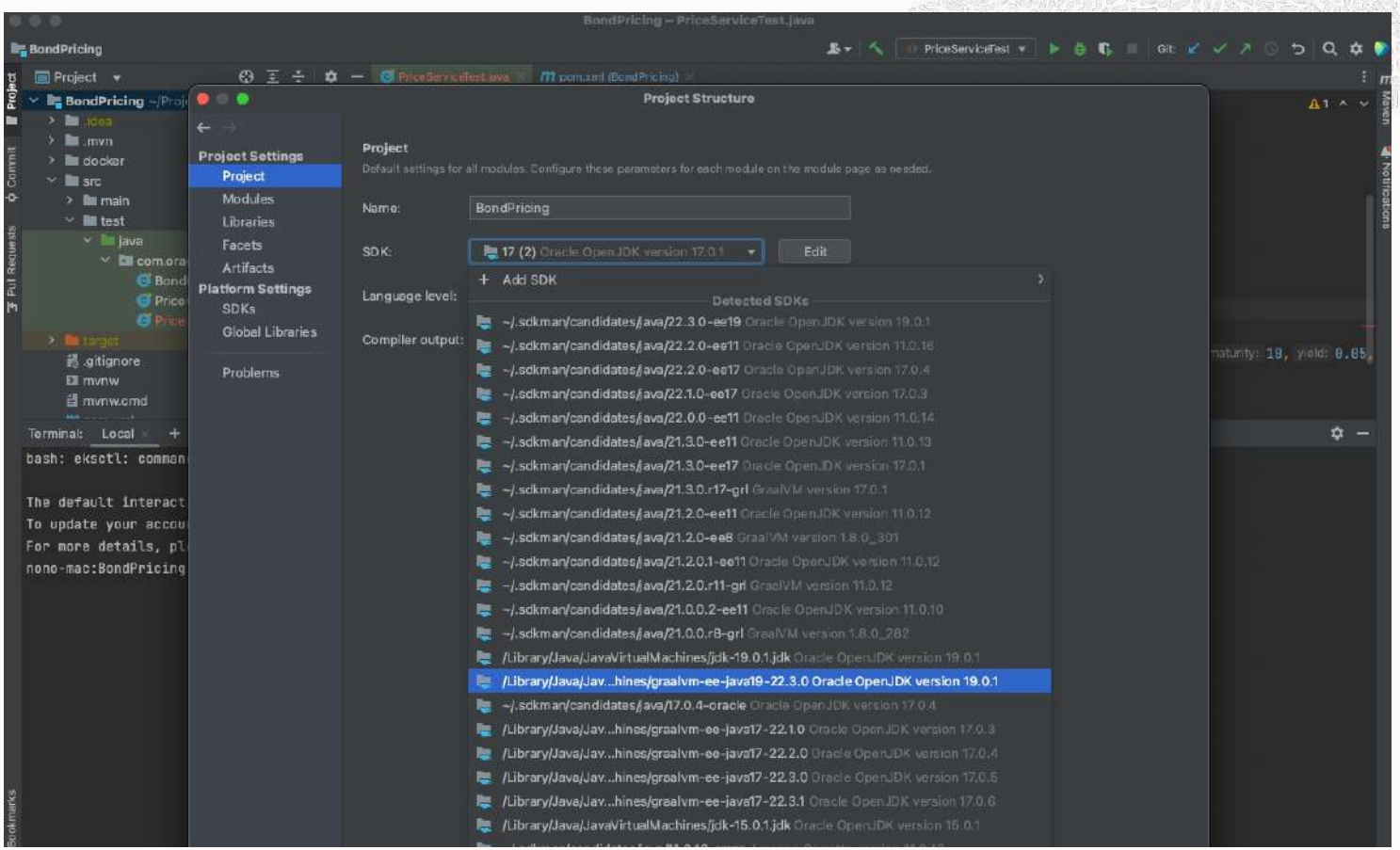


IDE Integration

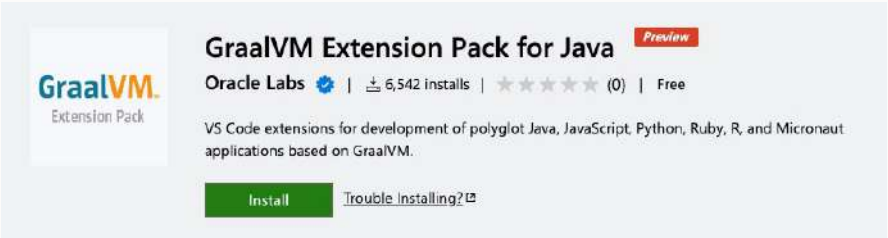


Install GraalVM in VS Code by using the **Gr** Installation Wizard

GraalVM Intellij



GraalVM IDE Add-On



GraalVM Plugins

Plugin	Platforms	Link
Native Build Tools	Maven Gradle	https://graalvm.github.io/native-build-tools/
Paketo Build Pack	GraalVM	https://github.com/paketo-buildpacks/graalvm
Spring AOT Plugins	Spring Boot	
Micronaut GraalVM	Micronaut	https://plugins.gradle.org/plugin/io.micronaut.graalvm

Containerizing Hello World

src > main > java > com > oracle > graalvm > demos > hellonative > J App.java > Java Languag

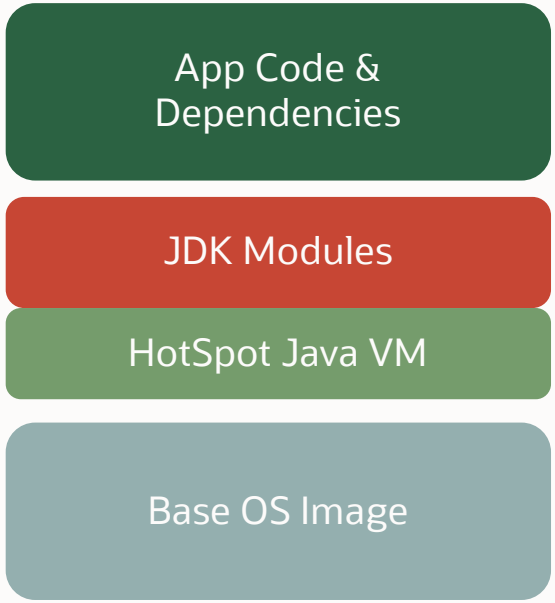
```
1   package com.oracle.graalvm.demos.hellonative;
2
3   /**
4    * Hello world!
5    *
6    */
7   public class App
8   {
        Run main | Debug main
9       public static void main( String[] args )
10      {
11          System.out.println( "Hello World!" );
12      }
13  }
14
```


Java Native Executable w GraalVM Native Image

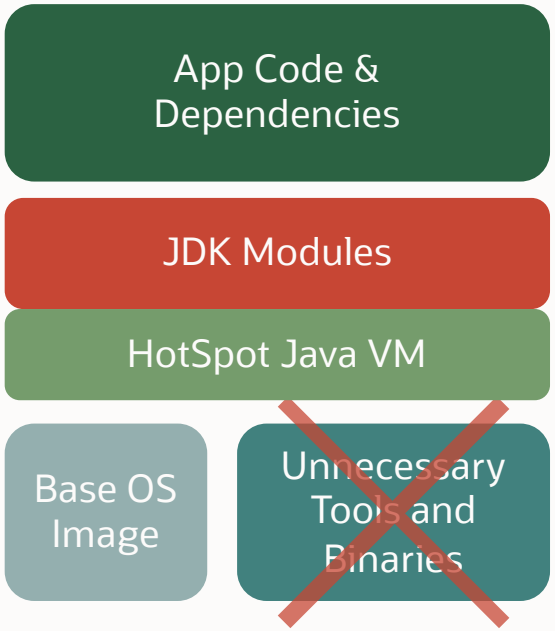
Compile

```
$ git clone https://github.com/nelvadas/graalvm-helloworld-nativeimage.git
$ mvn clean package

$ native-image -cp target/*.jar com.oracle.graalvm.demos.hellonative.App HelloApp
-rwxr-xr-x   1 nono  staff   4,6M 22 fév 08:14 HelloApp

$ upx helloworld
-rwxr-xr-x   1 nono  staff   1,7M 22 fév 08:14 HelloApp

$
```

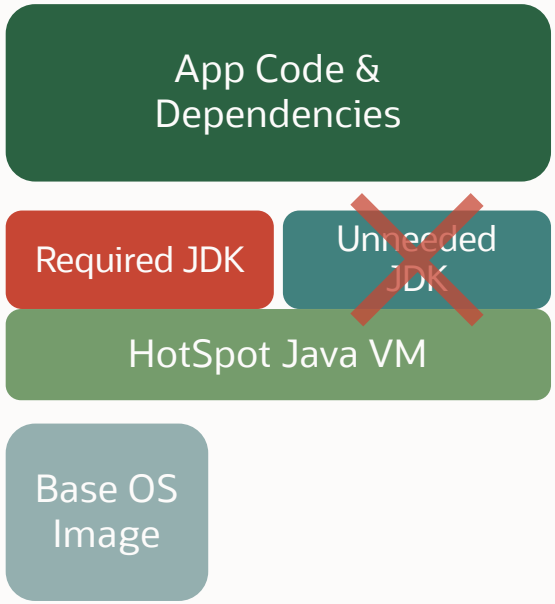

Java in Container



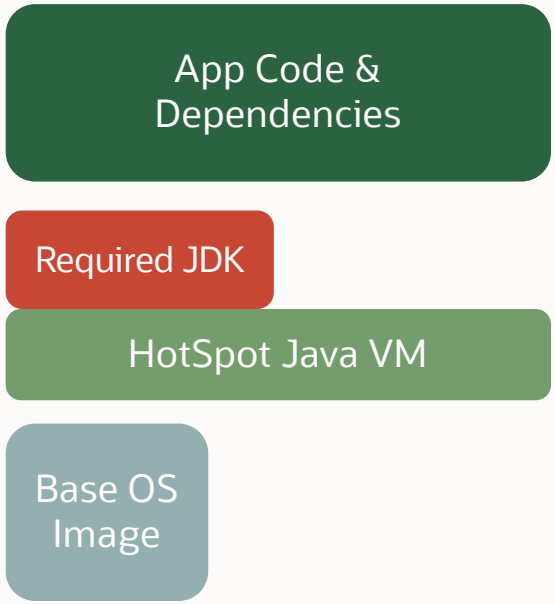
Java in a Slim/Distroless Container



Java using jlink in a Slim/Distroless Container



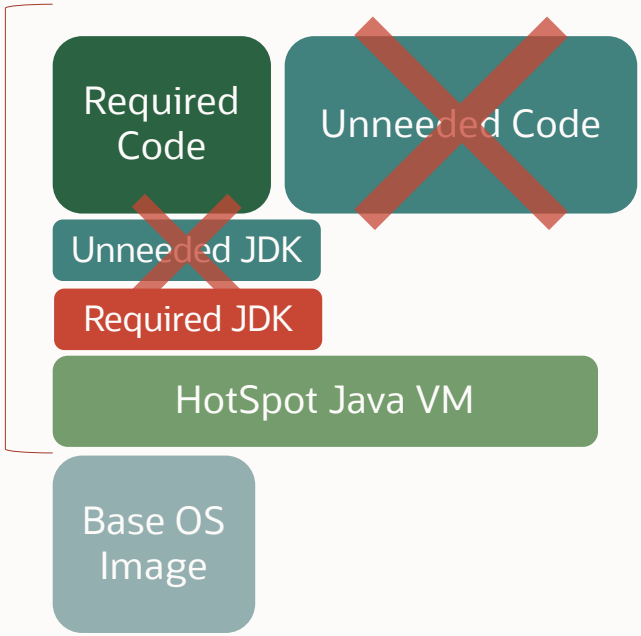
Java using jlink in a Slim/Distroless Container



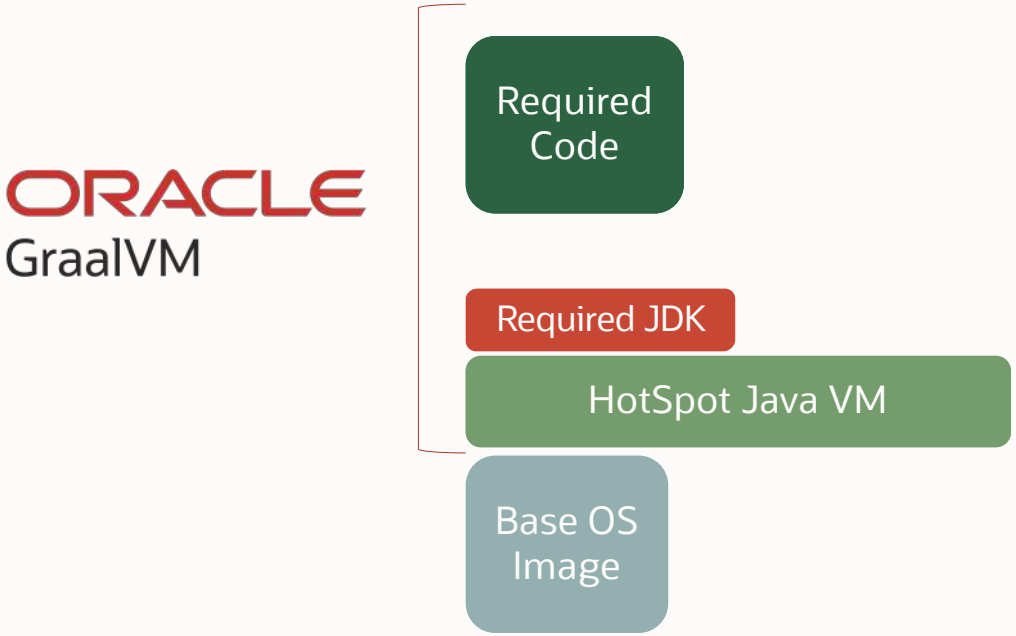
Java Native Executable in Scratch Container w/ GraalVM Native Image

```
native-image --static -jar <jar> <app name>
```

ORACLE GraalVM



Java Native Executable in Scratch Container w/ GraalVM Native Image



Java Native Executable in Scratch Container w/ GraalVM Native Image

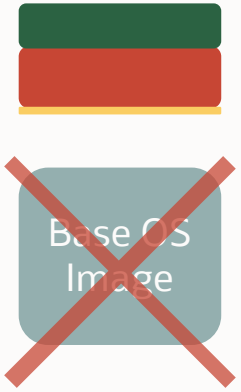
ORACLE GraalVM

libc statically linked into executable

HotSpot replaced by thin runtime services layer

Java Native Executable in Scratch Container w/ GraalVM Native Image

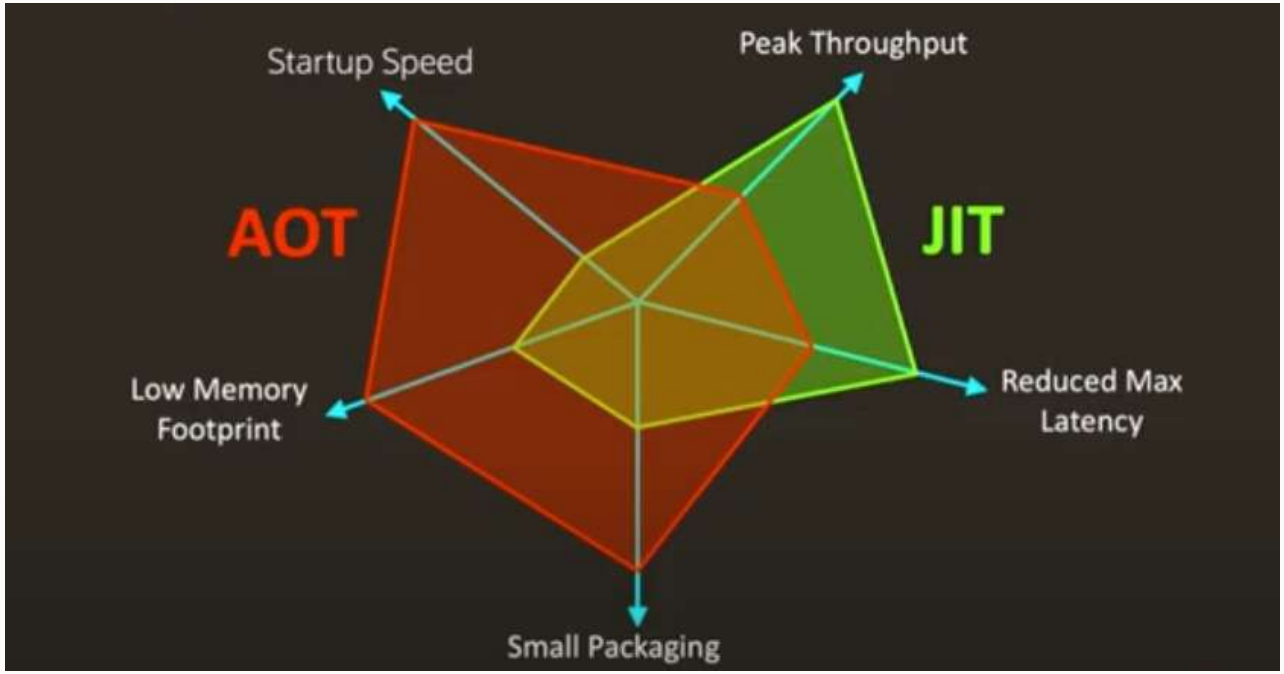
```
FROM scratch
COPY helloworld app
ENTRYPOINT ["/app"]
```



Java Native Executable in Scratch Container w/ GraalVM Native Image

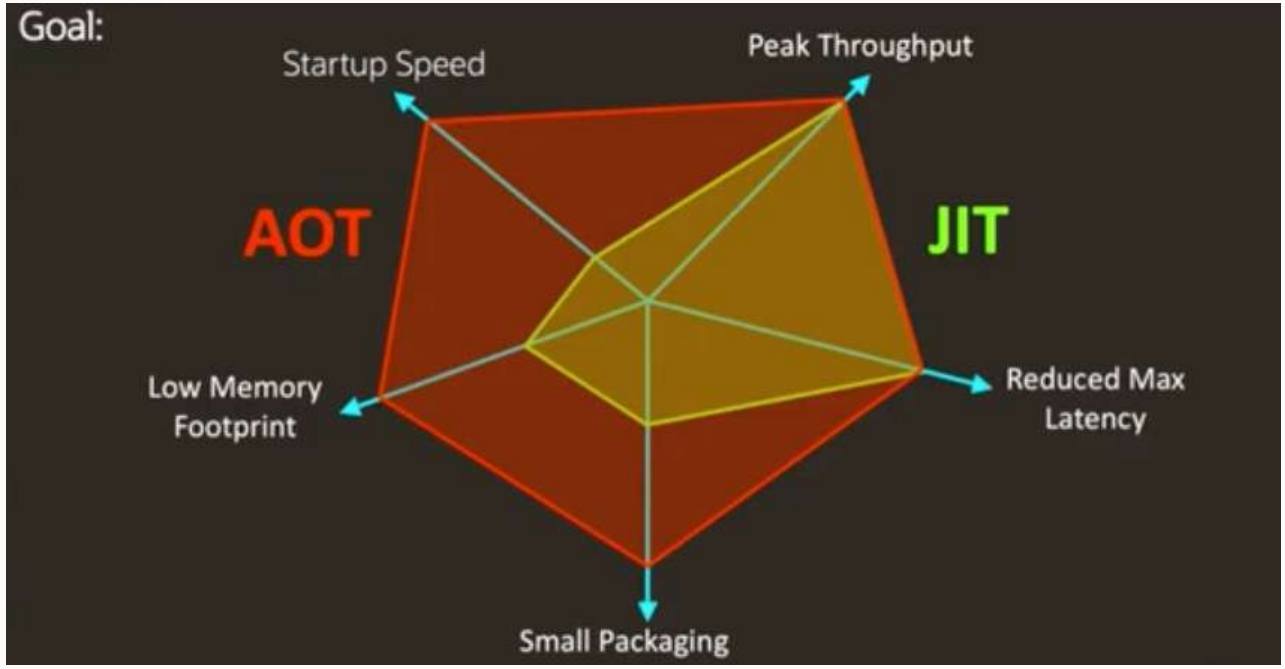
Java Hello World container image < ~15 MB

Benefits of GraalVM Enterprise Edition
Performance

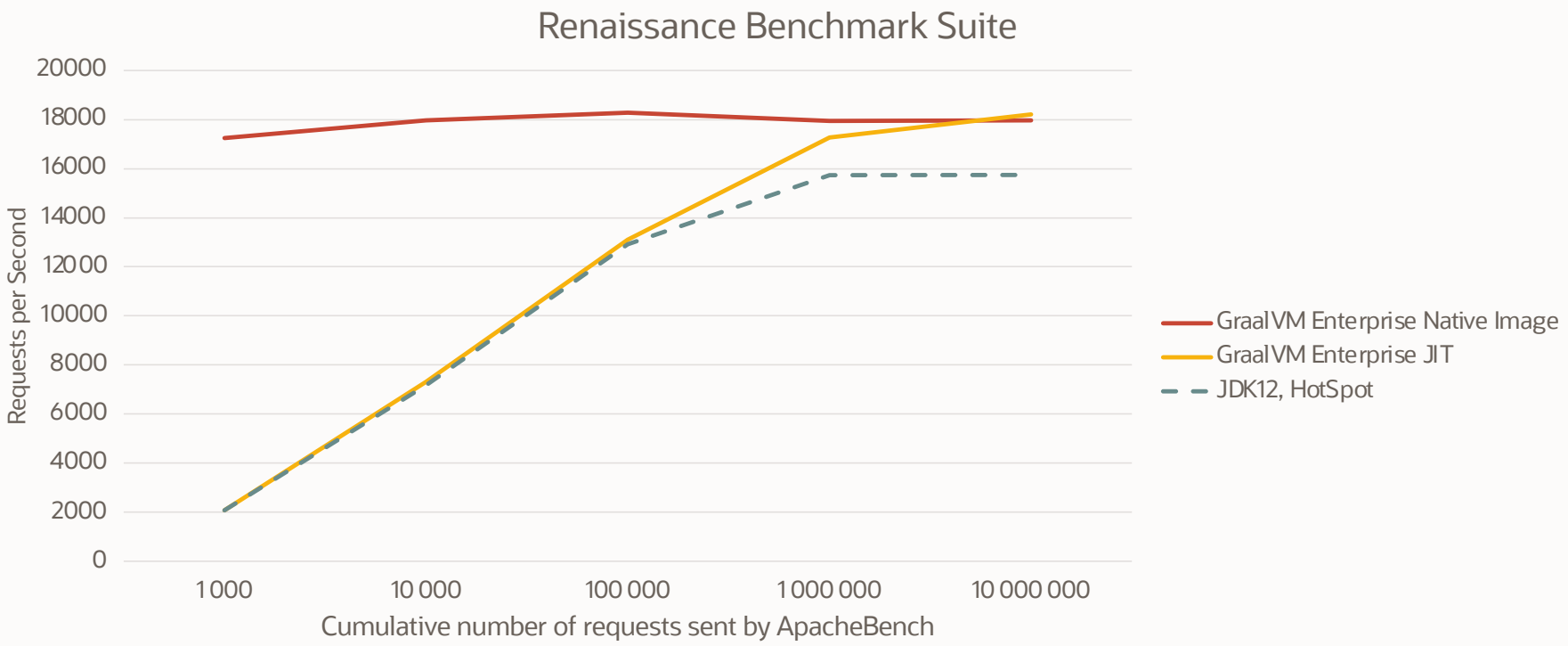


Benefits of GraalVM Enterprise Edition

Peak Throughput with EE Edition



GraalVM Enterprise throughput—Profile Guided Optimization Goal + G1GC



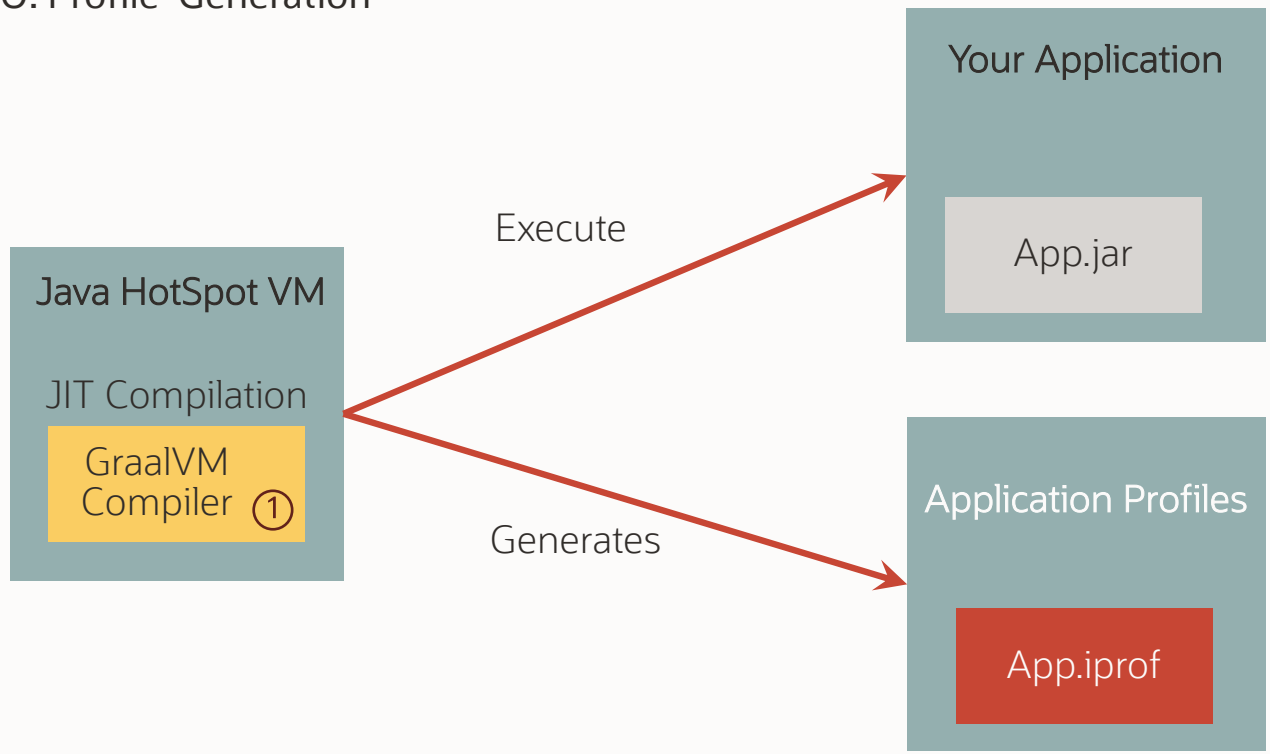
Java Native Executable w GraalVM Native Image

Compile

```
$ native-image --pgo-instrument HelloWorld native1
$ $ ./native1
$ ls -rtl
-rw-rw-r--. 1 opc opc      451 May 24 14:43 HelloWorld.java
-rw-rw-r--. 1 opc opc      451 May 24 14:43 HelloWorld.class
-rwxrwxr-x. 1 opc opc  4999592 May 24 14:47 helloworld
-rwxrwxr-x. 1 opc opc 52264896 May 24 14:54 native1
-rw-------. 1 opc opc  4090664 May 24 14:59 default.iprof
$ native-image  --pgo=default.iprof HelloWorld -H:Name=native2



$
```

Native Image

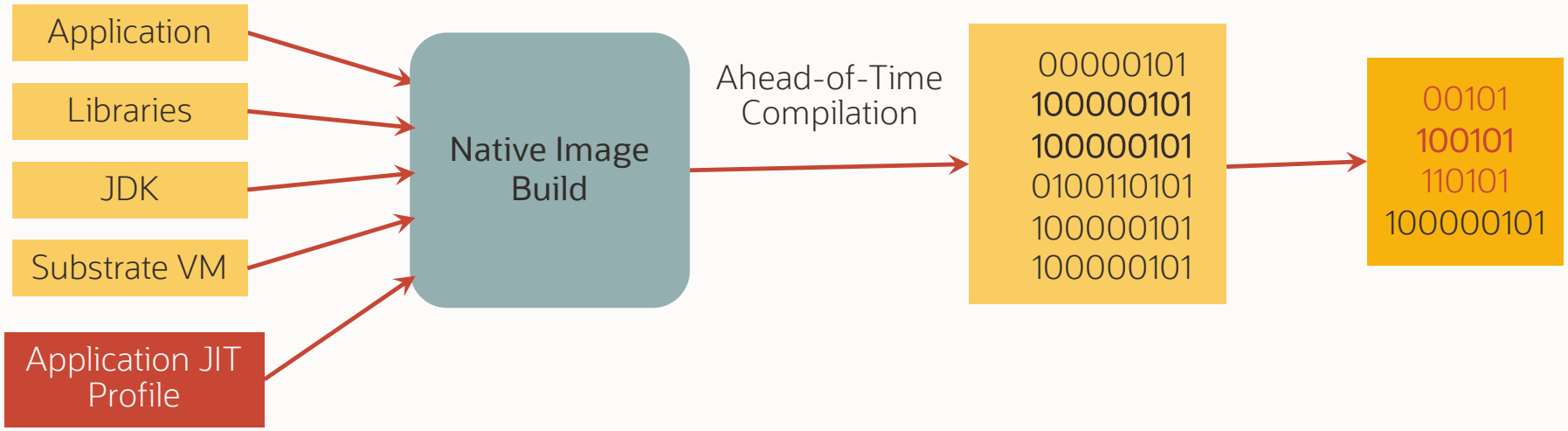
PGO: Profile Generation



1 Compiler run the application in JIT mode with. **`--pgo-instrument`**

Native Image

PGO: Native image Optimizations



Run with **`--pgo=default.iprof`**

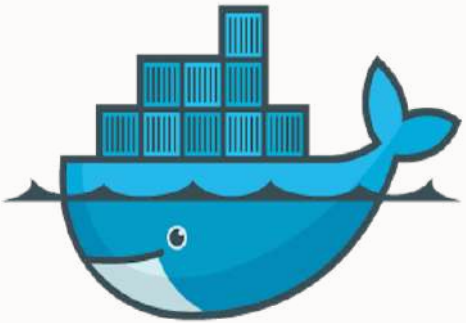
- Compress pointers
- Increased Througpouts

1

GraalVM Enterprise Native Image

Supported by microservice frameworks and platforms

Ideal for Containers



GraalVM on AWS

https://maxday.github.io/lambda-perf/

Lambda Cold Starts analysis by maxday

Visualize 10 Cold Starts for each runtime, updated daily.

[How to deploy a Rust Lambda function?]

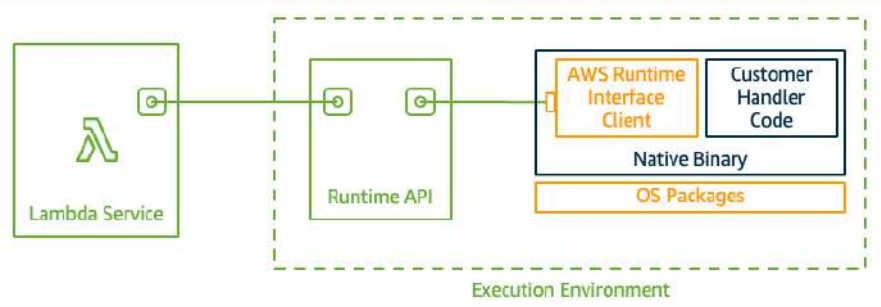
[How does it work?]

us-east1 | x86_64 | 128MB

Last update: 2023-02-20T14:00:06.298Z

Avg Cold Start duration | Avg Memory Used | Avg duration

rust (prov.al2)	go on (provided)
15.958ms 14.1MB 1.299ms	53.39ms 19MB 1.518ms
golx	graalvm java17 (prov.al2)
85.428ms 29.1MB 3.419ms	112.733ms 32.1MB 102.263ms
python37	dotnet7 aot (prov.al2)
113.221ms 36MB 1.806ms	115.011ms 34MB 33.219ms
python39	python38
117.149ms 36.2MB 2.824ms	124.655ms 38MB 6.814ms
nodejs16x	ruby27
150.212ms 57.2MB 11.086ms	150.653ms 30.1MB 12.133ms
nodejs12x	dotnetcore31
156.285ms 55MB 8.061ms	179.447ms 50.4MB 223.059ms
nodejs14x	java11 snapstart
180.9ms 56.6MB 12.166ms	209.733ms 77.7MB



Source : https://github.com/aws-samples/serverless-graalvm-demo

Potential Issues ?

- Longer Native Build Times for some Apps
- Frameworks Specifics
- Dependencies Reachability
  - Proxies
  - Reflection
  - JNI
    - @see Native Image Agent + Reachability Metadata

Lab

BondPricer Application : Java Stream to compute Bond Price

What is the price of 3-year Maturity UnitCredit bond giving a 5% annual Coupon with a 100€ Face Value?
Using the Present Value Model

$$PV = \sum_{t=1}^{T} \frac{C}{(1+r)^t} + \frac{V}{(1+r)^T}$$

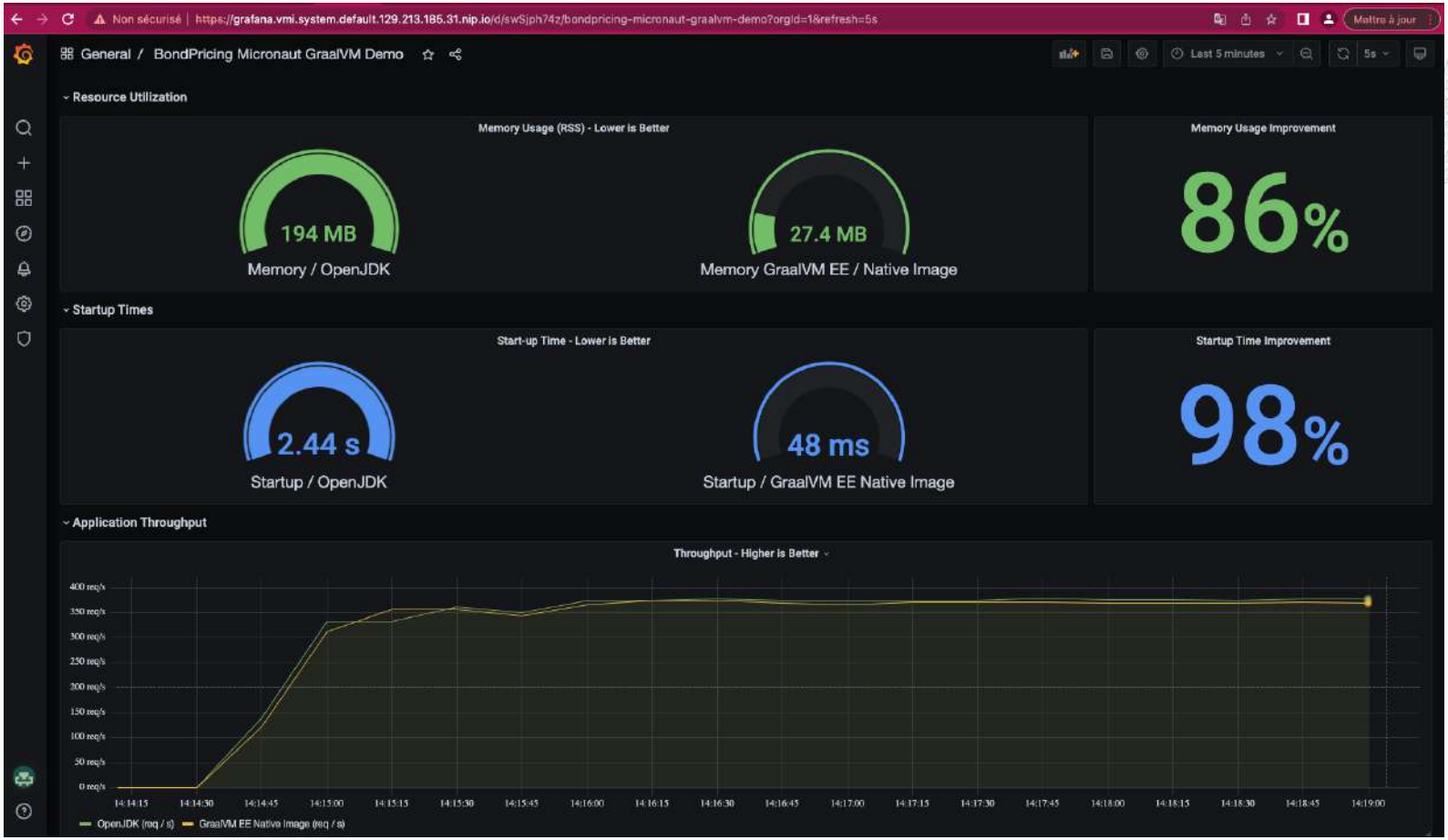
GraalVM Hands on Labs (1) - Overview

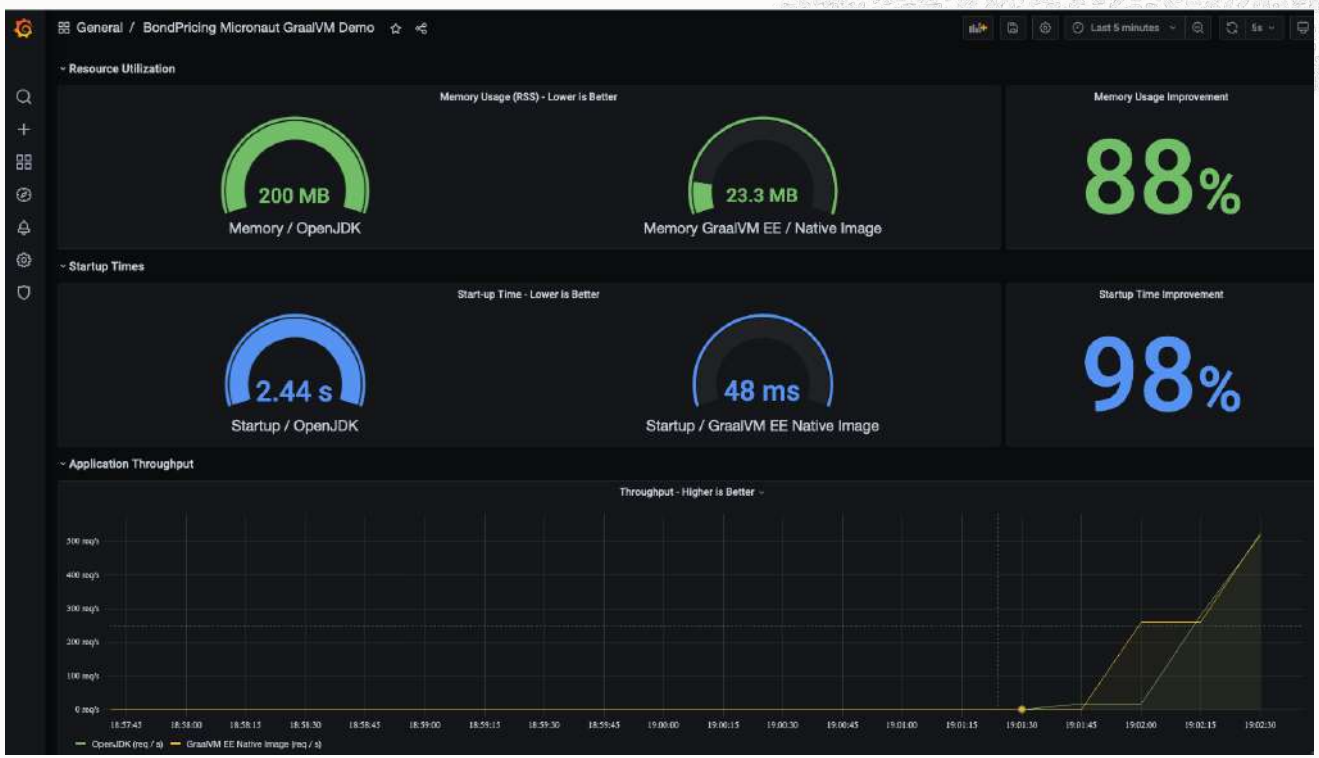
https://github.com/nelvadas/GraalVM101

GraalVM 101 - Spring Boot + Micronaut

GraalVM 101: *Practical Workshop to Get Started with GraalVM Enterprise Edition.*

Table of Contents





Q&A

Additional Resources

Visit Oracle GraalVM homepage:
oracle.com/graalvm

Connect with us on Twitter:
@GraalVM

Contact us:
graalvmcomms_ww@oracle.com

ORACLE
GraalVM

This concludes our presentation.

Thank you for tuning in!

GraalVM Webcast Series
oracle.com/graalvm/webcasts


ORACLE®