Компилируем ahead-of-time с GraalVM™

Oleg Šelajev
Developer Advocate, GraalVM team, Oracle Labs
@shelajev

ORACLE®

Safe Harbor Statement

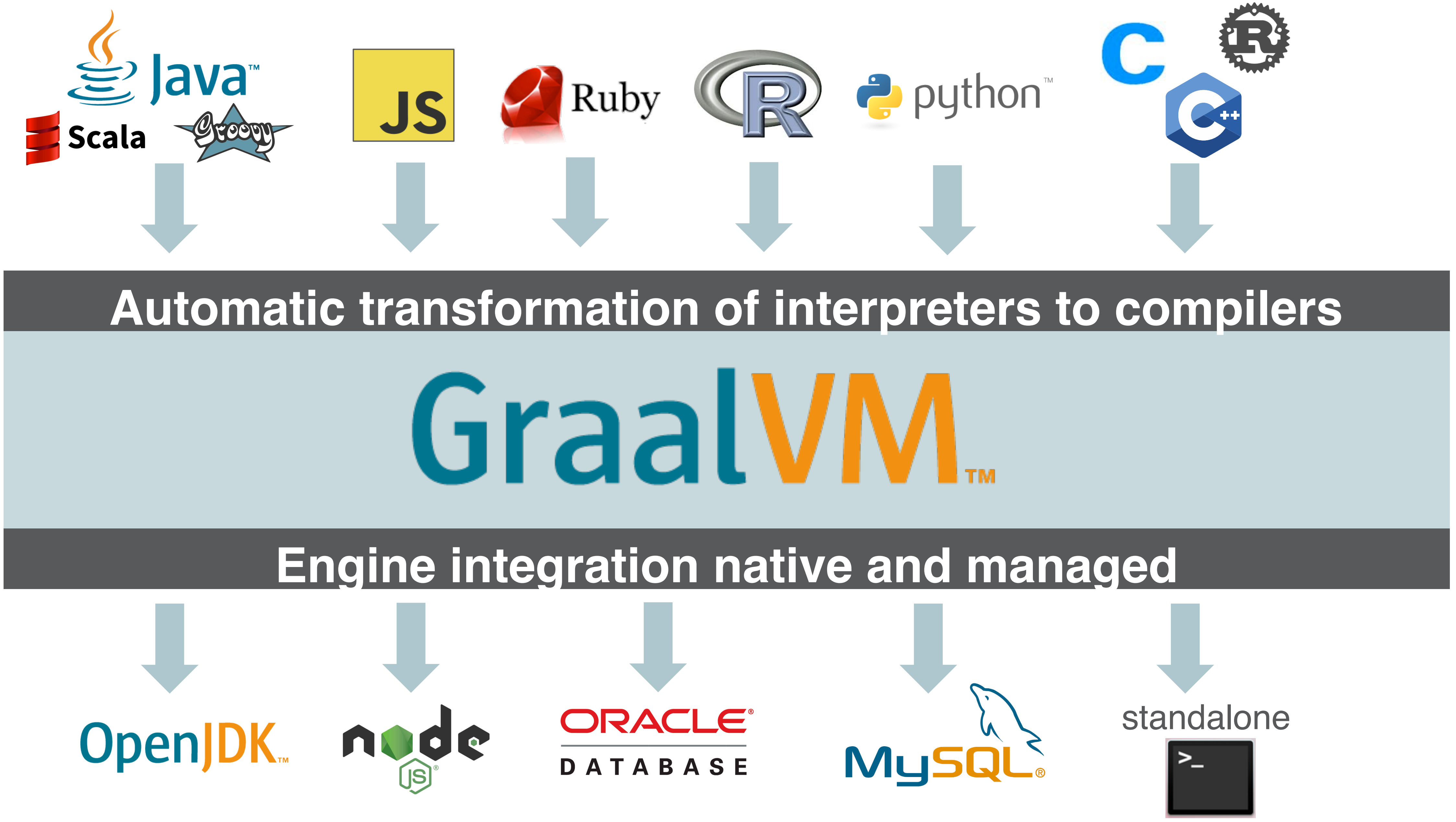
The following is intended to provide some insight into a line of research in Oracle Labs. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described in connection with any Oracle product or service remains at the sole discretion of Oracle. Any views expressed in this presentation are my own and do not necessarily reflect the views of Oracle.

spring

by Pivotal™

GraalVM™

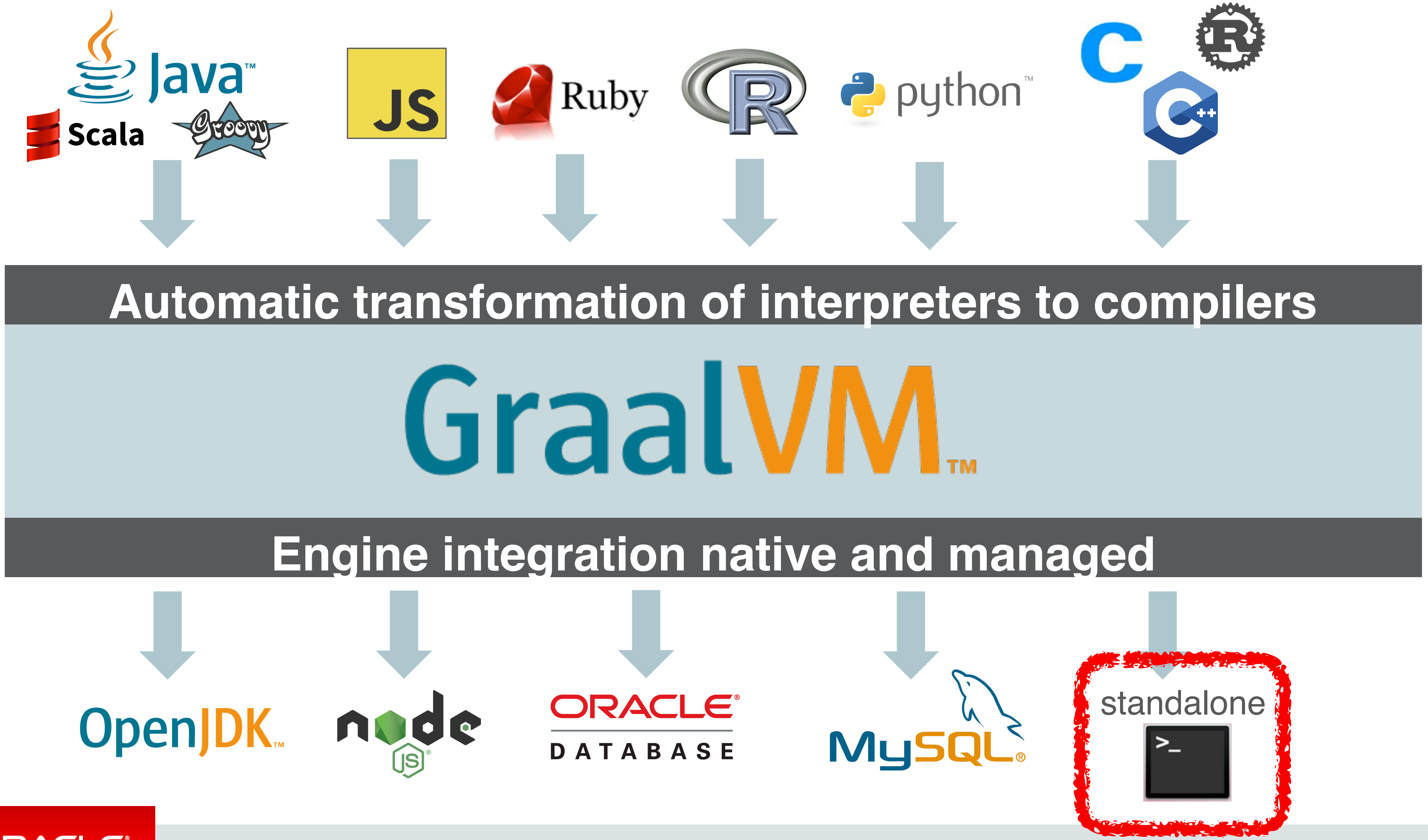
ORACLE®

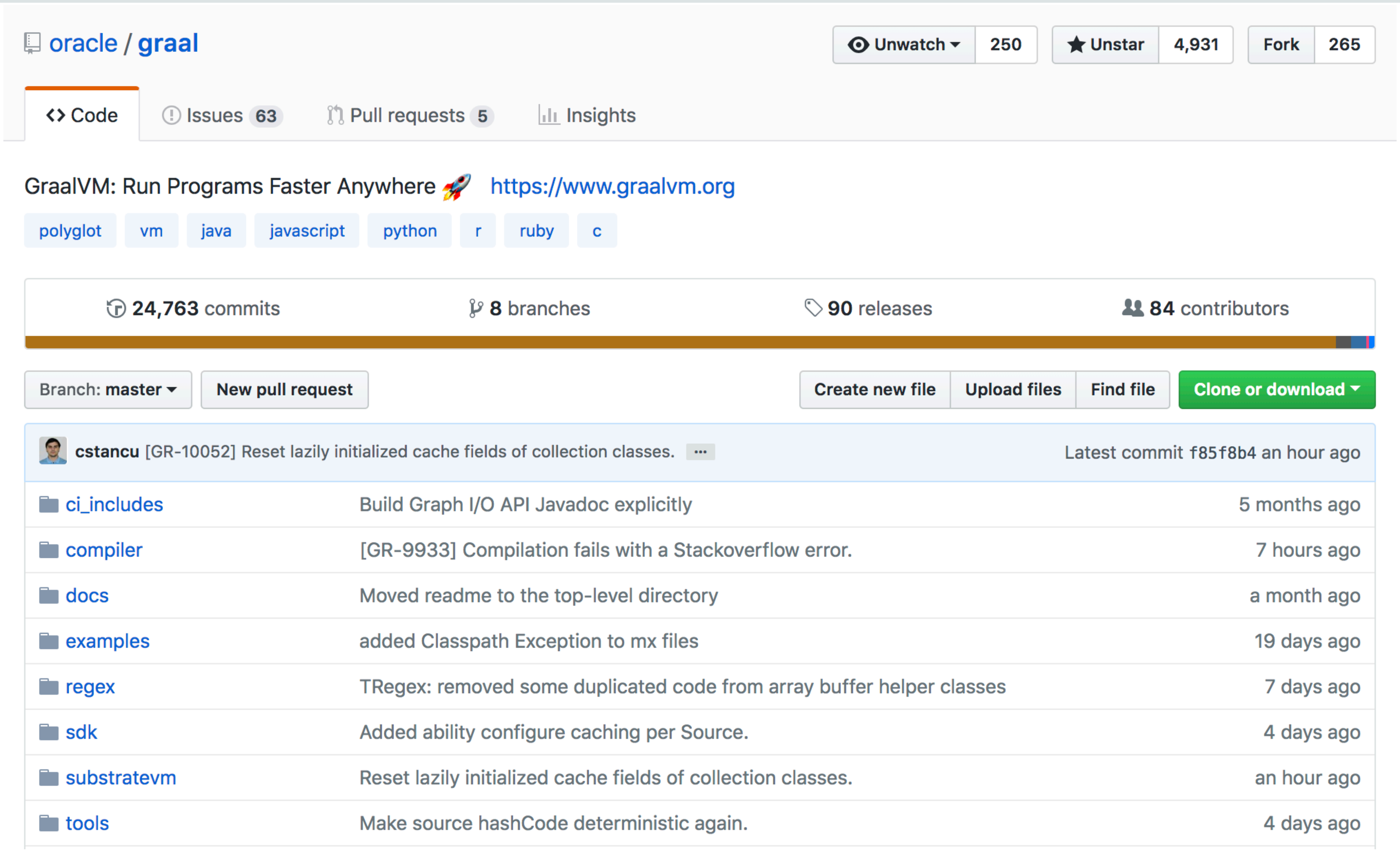

Java
Scala
Groovy
JS
Ruby
R
python
C
C++
Automatic transformation of interpreters to compilers
GraalVM
Engine integration native and managed
OpenJDK
node JS
ORACLE DATABASE
MySQL
standalone
>_

Top 10 Things To Do With GraalVM

1. High-performance modern Java
2. Low-footprint, fast-startup Java
3. Combine JavaScript, Java, Ruby, and R
4. Run native languages on the JVM
5. Tools that work across all languages
6. Extend a JVM-based application
7. Extend a native application
8. Java code as a native library
9. Polyglot in the database
10. Create your own language

medium.com/graalvm/graalvm-ten-things-12d9111f307d


Java
Scala
Groovy
JS
Ruby
R
python
C
C++
Automatic transformation of interpreters to compilers
GraalVM
Engine integration native and managed
OpenJDK
node JS
ORACLE DATABASE
MySQL
standalone
>_



oracle / graal

Unwatch 250 | Unstar 4,931 | Fork 265

Code | Issues 63 | Pull requests 5 | Insights

GraalVM: Run Programs Faster Anywhere 🚀 https://www.graalvm.org

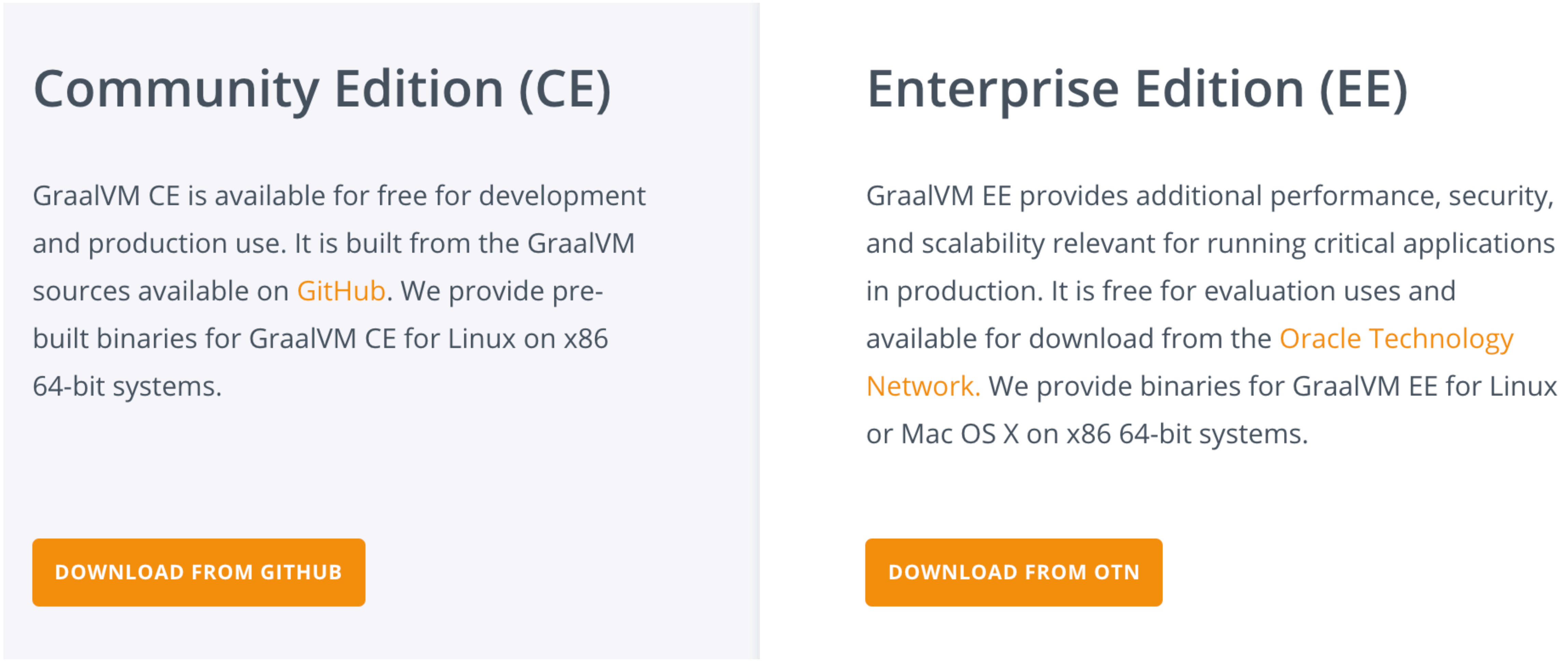
polyglot vm java javascript python r ruby c

24,763 commits | 8 branches | 90 releases | 84 contributors

Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

cstancu [GR-10052] Reset lazily initialized cache fields of collection classes. … Latest commit f85f8b4 an hour ago

ci_includes	Build Graph I/O API Javadoc explicitly	5 months ago
compiler	[GR-9933] Compilation fails with a Stackoverflow error.	7 hours ago
docs	Moved readme to the top-level directory	a month ago
examples	added Classpath Exception to mx files	19 days ago
regex	TRegex: removed some duplicated code from array buffer helper classes	7 days ago
sdk	Added ability configure caching per Source.	4 days ago
substratevm	Reset lazily initialized cache fields of collection classes.	an hour ago
tools	Make source hashCode deterministic again.	4 days ago



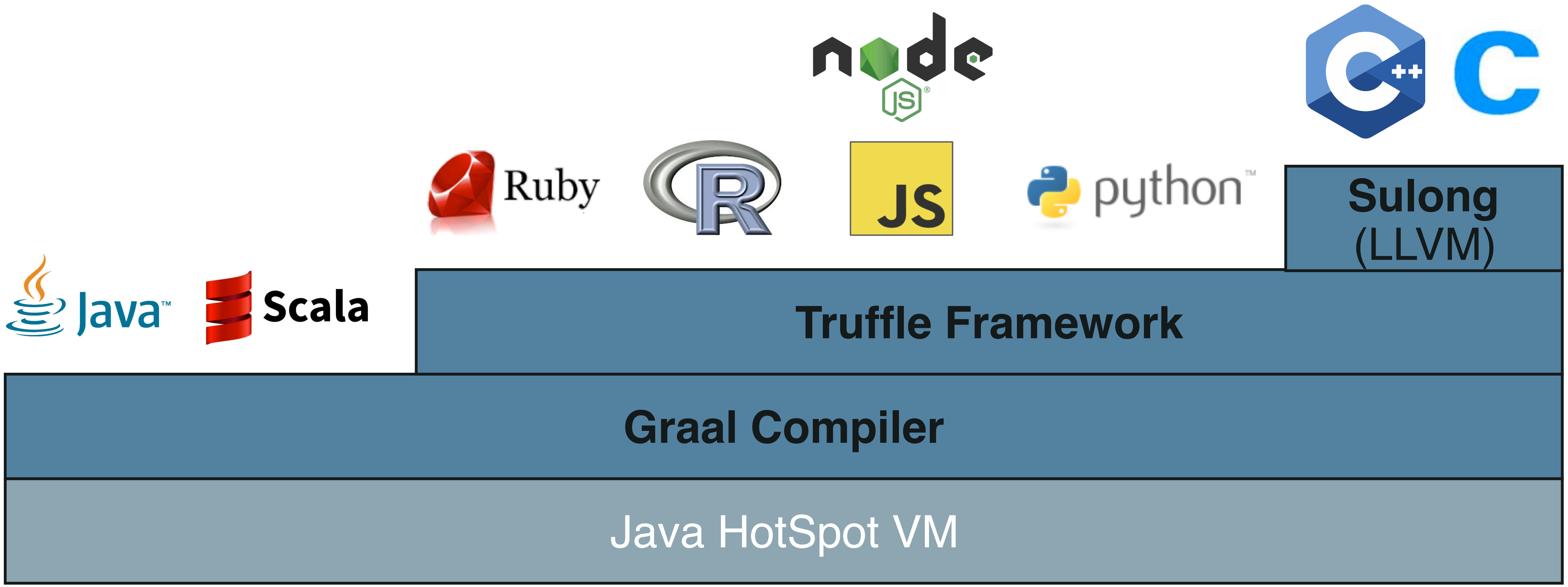
Community Edition (CE)

GraalVM CE is available for free for development and production use. It is built from the GraalVM sources available on GitHub. We provide pre-built binaries for GraalVM CE for Linux on x86 64-bit systems.

Enterprise Edition (EE)

GraalVM EE provides additional performance, security, and scalability relevant for running critical applications in production. It is free for evaluation uses and available for download from the Oracle Technology Network. We provide binaries for GraalVM EE for Linux or Mac OS X on x86 64-bit systems.

www.graalvm.org/downloads


Java
Scala
Ruby
R
node
JS
JS
python
C++
C
Sulong
(LLVM)
Truffle Framework
Graal Compiler
Java HotSpot VM

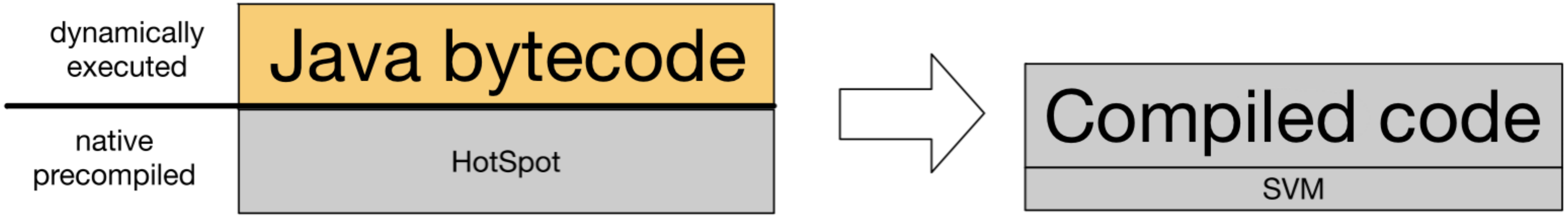
GraalVM native images

Native images

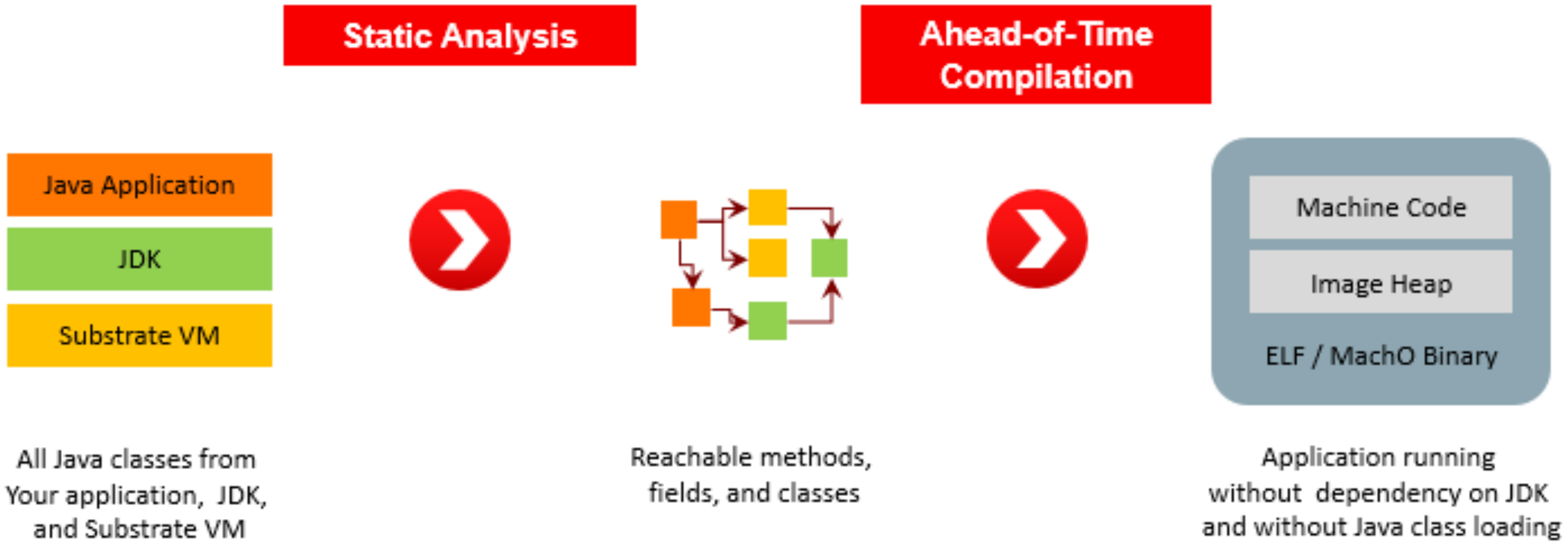
- Full AOT compilation to machine code
- Works with memory management
- Secure execution (e.g., bounds checks)
- **Embeddable with native applications**

Native images

- Full AOT compilation to machine code
- Works with memory management
- Secure execution (e.g., bounds checks)
- **Embeddable with native applications**



Jenga


Static Analysis
Ahead-of-Time Compilation
Java Application
JDK
Substrate VM
Machine Code
Image Heap
ELF / MachO Binary
All Java classes from Your application, JDK, and Substrate VM
Reachable methods, fields, and classes
Application running without dependency on JDK and without Java class loading

Static initializers

Static class initialization blocks, pre-initialized static variables.

Static initializers

By **default** static class initialization is done **during native image construction**.

Large static data structures are pre-allocated allowing faster startup of the generated image.

No **instance-specific initializations** can be done in static initializers.

Static initializers - things not to do

- start application **threads** that continue to run in the background
- **load** native libraries using `java.lang.Runtime.load(String)`
- open **files or sockets**, or
- allocate C memory, e.g., `java.nio.ByteBuffer.allocateDirect(int)`.

Static initializers

Write your **own initialization methods**
and **call them explicitly** from your main entry point.

```java
import java.util.Date;

class HelloCachedTime {
  static final Date CACHED_TIME = Startup.TIME;

  public static void main(String args[]) {
    System.out.println("Startup: " + CACHED_TIME);
    System.out.println("Now:     " + new Date());
  }
}

class Startup {
  static final Date TIME = new Date();
}
```

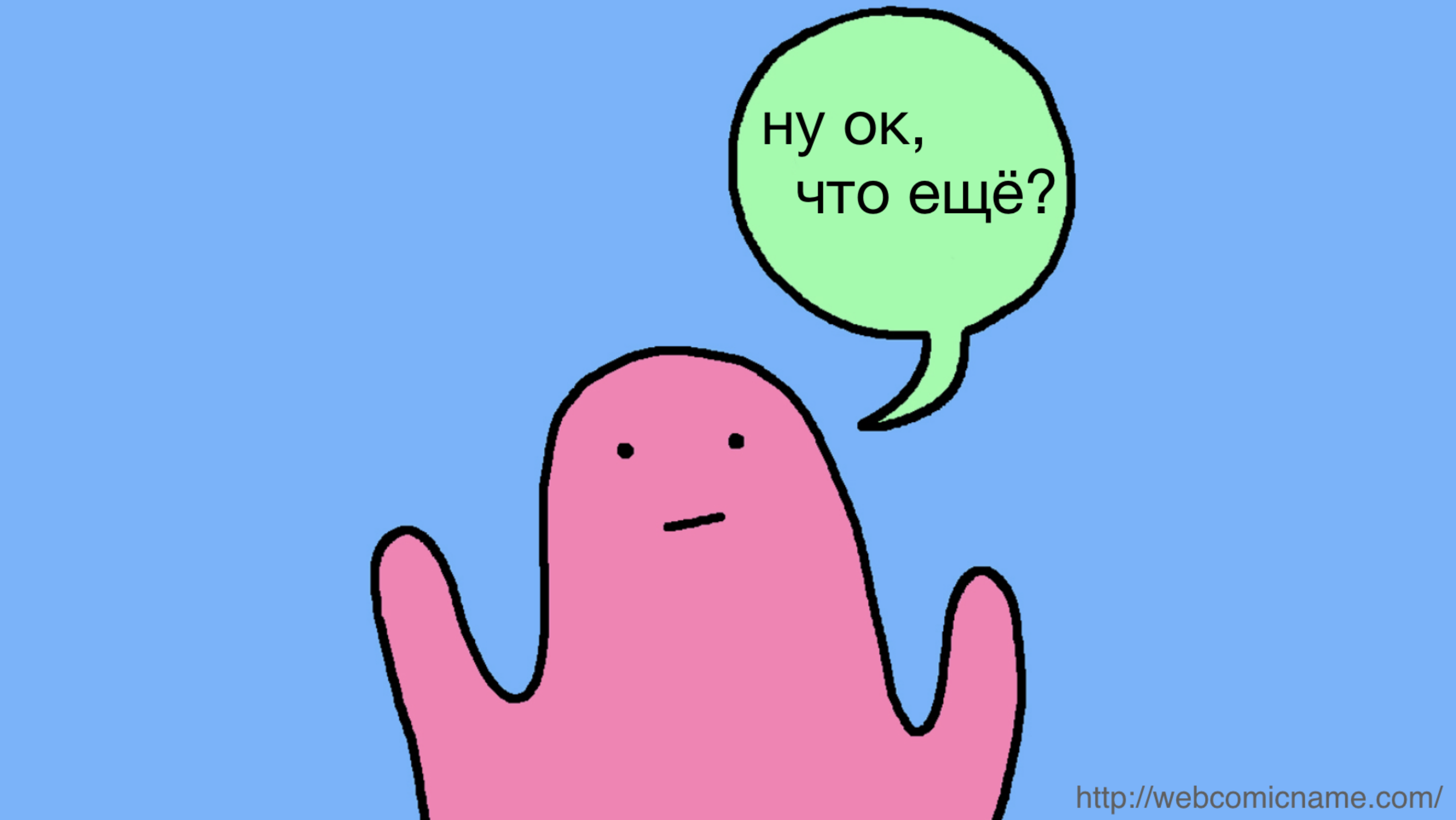
```
> $JAVA_HOME/bin/javac HelloStartupTime.java


> $JAVA_HOME/bin/java HelloStartupTime
Startup: Fri Aug 31 13:17:05 PDT 2018
Now:     Fri Aug 31 13:17:05 PDT 2018
```

```
> $JAVA_HOME/bin/javac HelloStartupTime.java


> $JAVA_HOME/bin/java HelloStartupTime
Startup: Fri Aug 31 13:17:05 PDT 2018
Now:     Fri Aug 31 13:17:05 PDT 2018
```

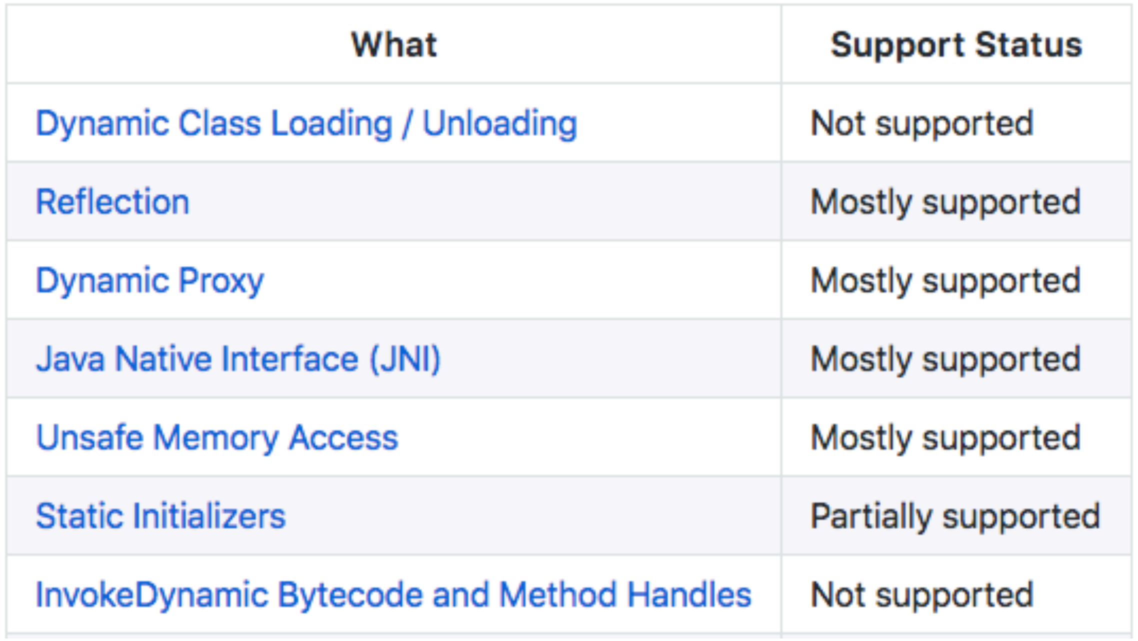
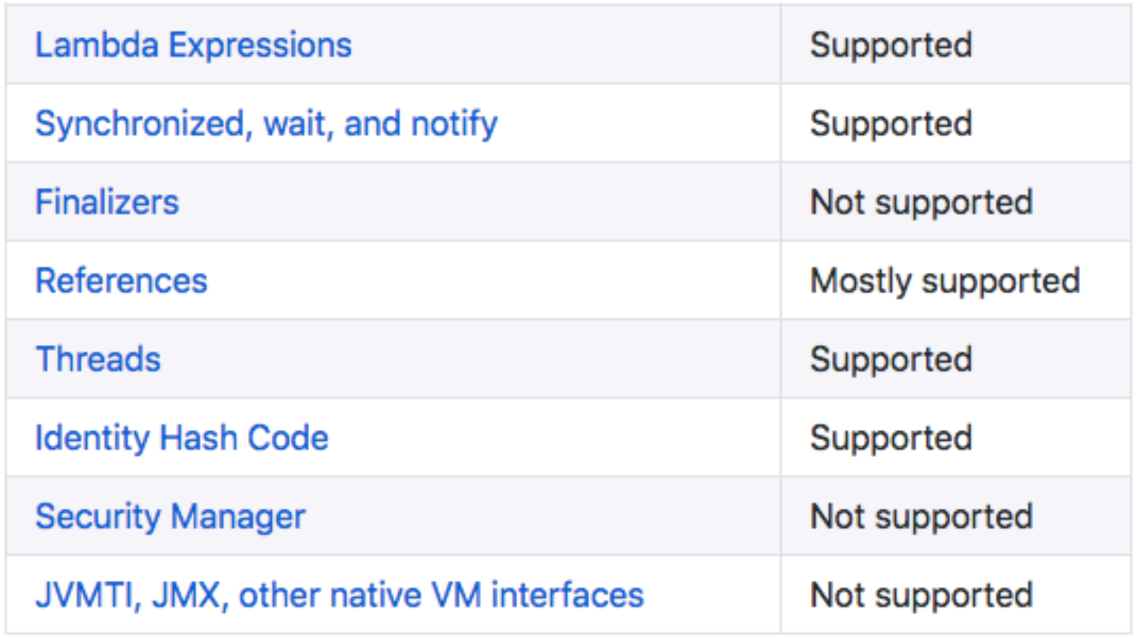
```
> $JAVA_HOME/bin/javac HelloStartupTime.java


> $JAVA_HOME/bin/java HelloStartupTime
Startup: Fri Aug 31 13:17:05 PDT 2018
Now:     Fri Aug 31 13:17:05 PDT 2018

> $JAVA_HOME/bin/native-image HelloStartupTime

> ./hellostartuptime
Startup: Fri Aug 31 13:22:12 PDT 2018
Now:     Fri Aug 31 14:35:42 PDT 2018
```

Static initializers delay

```
--delay-class-initialization-to-runtime=
 class,list
```


ну ок, что ещё?
http://webcomicname.com/

Limitations

What	Support Status
Dynamic Class Loading / Unloading	Not supported
Reflection	Mostly supported
Dynamic Proxy	Mostly supported
Java Native Interface (JNI)	Mostly supported
Unsafe Memory Access	Mostly supported
Static Initializers	Partially supported
InvokeDynamic Bytecode and Method Handles	Not supported
Lambda Expressions	Supported
Synchronized, wait, and notify	Supported
Finalizers	Not supported
References	Mostly supported
Threads	Supported
Identity Hash Code	Supported
Security Manager	Not supported
JVMTI, JMX, other native VM interfaces	Not supported

Dynamic classloading

Dynamic classloading

Image run time

Dynamic classloading

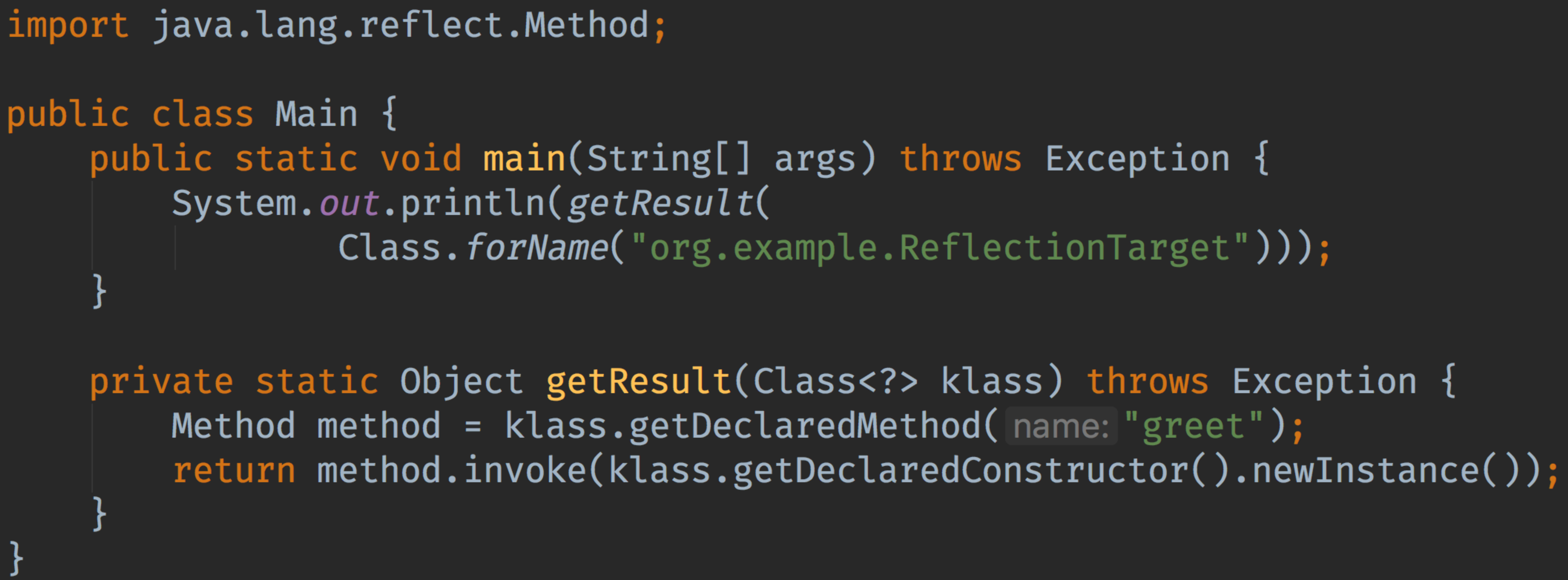
Image build time

Image run time

Reflection

```
-H:ReflectionConfigurationFiles=
```

```java
package org.example;

public class ReflectionTarget {
    public String greet() {
        return "Hello World!";
    }
}
```



```java
import java.lang.reflect.Method;

public class Main {
    public static void main(String[] args) throws Exception {
        System.out.println(getResult(
                Class.forName("org.example.ReflectionTarget")));
    }

    private static Object getResult(Class<?> klass) throws Exception {
        Method method = klass.getDeclaredMethod( name: "greet");
        return method.invoke(klass.getDeclaredConstructor().newInstance());
    }
}
```

```json
[
  {
    "name": "org.example.ReflectionTarget",
    "methods":  [
      {
        "name": "<init>",
        "parameterTypes":  []
      },
      {
        "name": "greet",
        "parameterTypes":  []
      }
    ]
  }
]
```

Java Native Interface (JNI)

```
-H:+JNI
-H:JNIConfigurationFiles=
```

Java Native Interface (JNI)

```
// Java declaration
native int[] sort0(int[] array);
// native declaration with JNI name mangling
jintArray JNICALL Java_org_example_sorter_IntSorter_sort0(
                        JNIEnv *env, jobject this, jintArray array) {...}
```

Finalizers

Finalizers

Use References and ReferenceQueues

Reference Queues

$$\texttt{Weak / Soft / Phantom} \longrightarrow \texttt{Feeble}$$

Reference Queues

`Weak / Soft / Phantom` $\longrightarrow$ `Feeble`

`Reference.enqueue() / Reference.isEnqueued()`

~~No~~

Resources

```
-H:IncludeResources=<regexp>
-H:IncludeResources=
     "application.yml|META-INF/services/*.*"
```

Unsafe

```
static final long fieldOffset = Unsafe.getUnsafe().objectFieldOffset(X.class.getDeclaredField("f"));

static final long arrayBaseOffsets = Unsafe.getUnsafe().arrayBaseOffset(byte[].class);

static final long byteArrayIndexScale = Unsafe.getUnsafe().arrayIndexScale(byte[].class);
```

Substitutions

```
Core classes (annotations):
com.oracle.svm.core.annotate.TargetClass
com.oracle.svm.core.annotate.Substitute
```

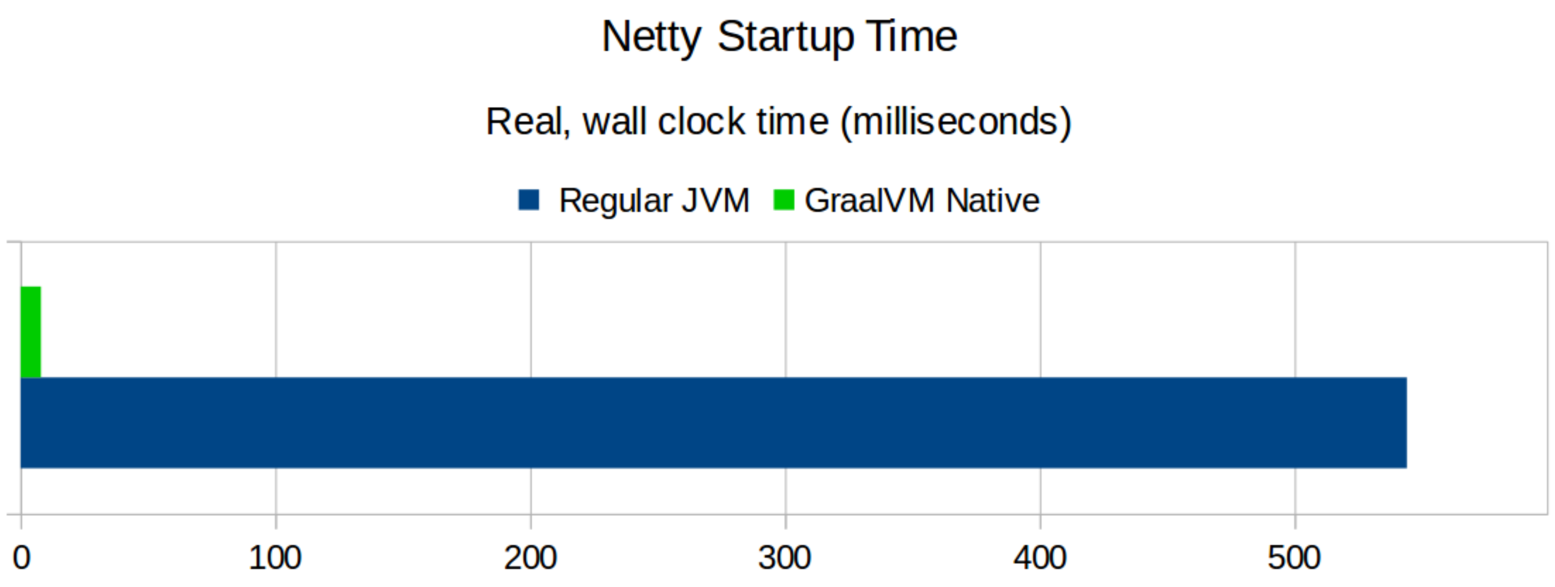
Image build time vs runtime

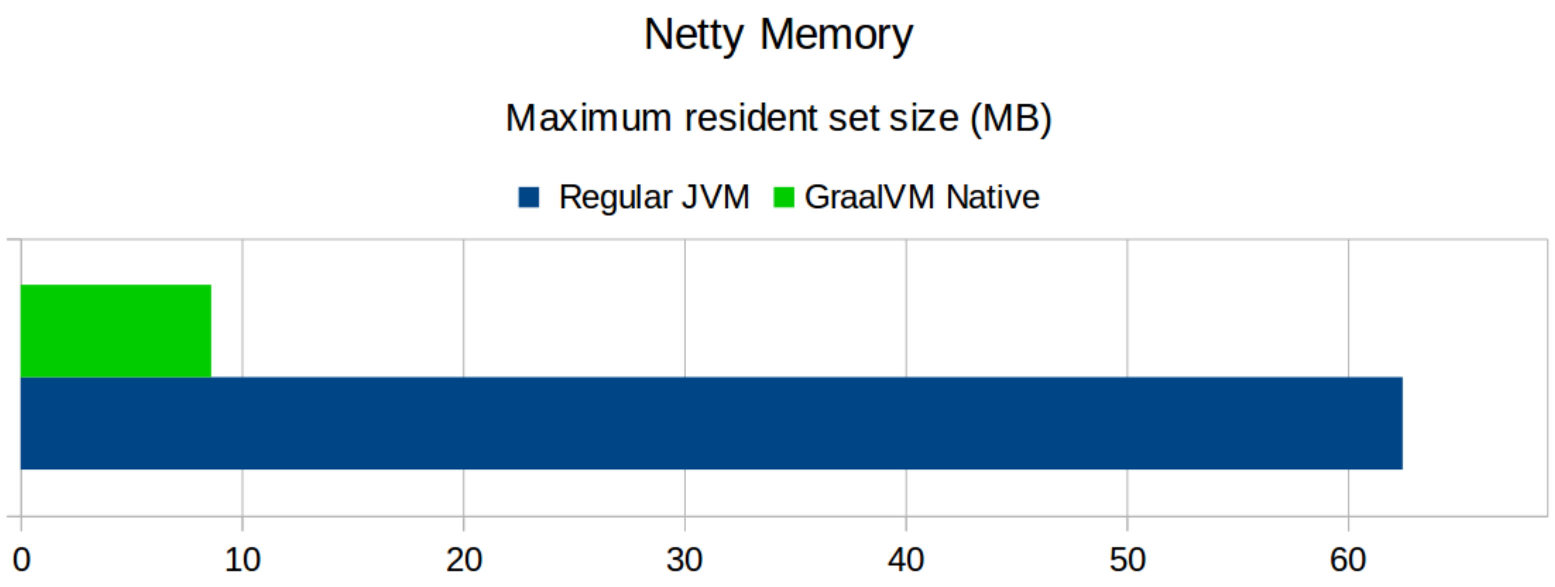
```
if(!ImageInfo.inImageCode()) {
 // JVM specific code here
}

if(ImageInfo.inImageBuildtimeCode()) {
 // we're building the image, let's go crazy with classloading
 // but not read any local configuration
}

if(ImageInfo.inImageRuntimeCode()) {
 // we're actually at runtime
}
```

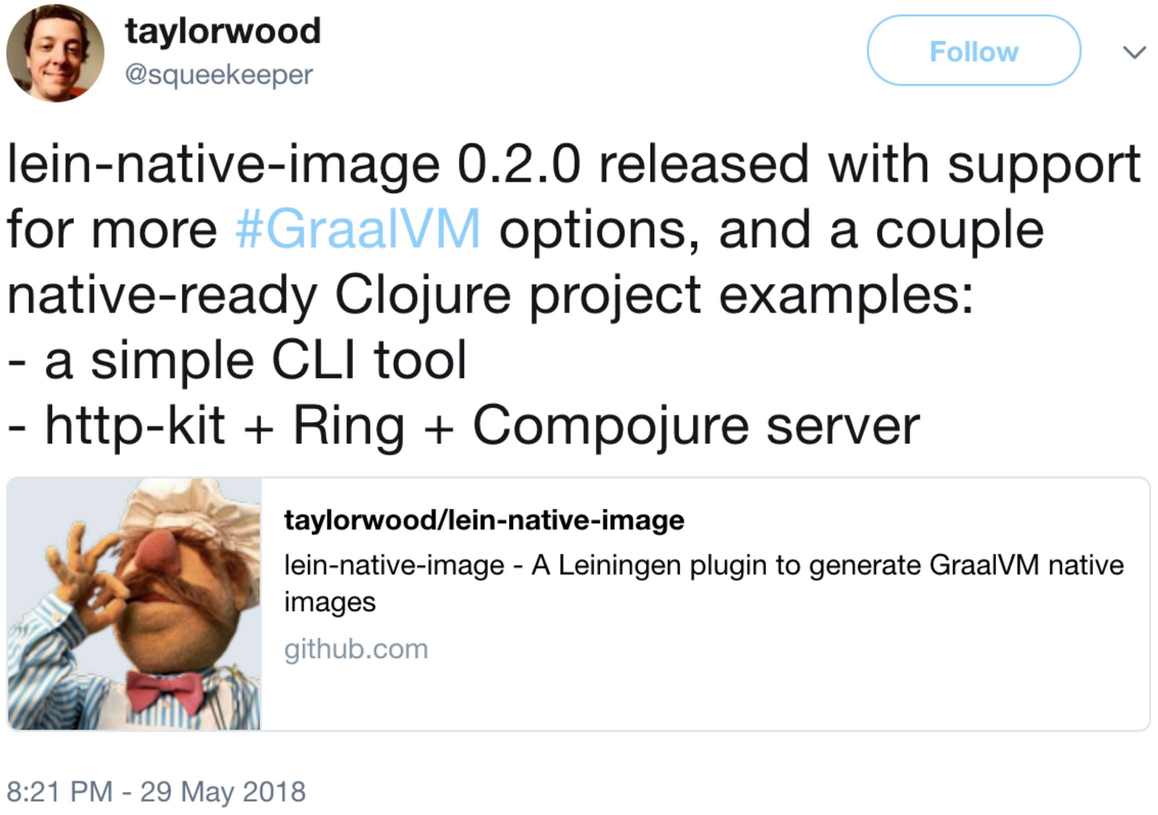
java.lang.NoClassDefFoundError

```
-H:+ReportUnsupportedElementsAtRuntime
```

https://medium.com/graalvm/instant-netty-startup-using-graalvm-native-image-generation-ed6f14ff7692


Netty Startup Time
Real, wall clock time (milliseconds)
Regular JVM
GraalVM Native
0
100
200
300
400
500


Netty Memory
Maximum resident set size (MB)
Regular JVM
GraalVM Native
0
10
20
30
40
50
60



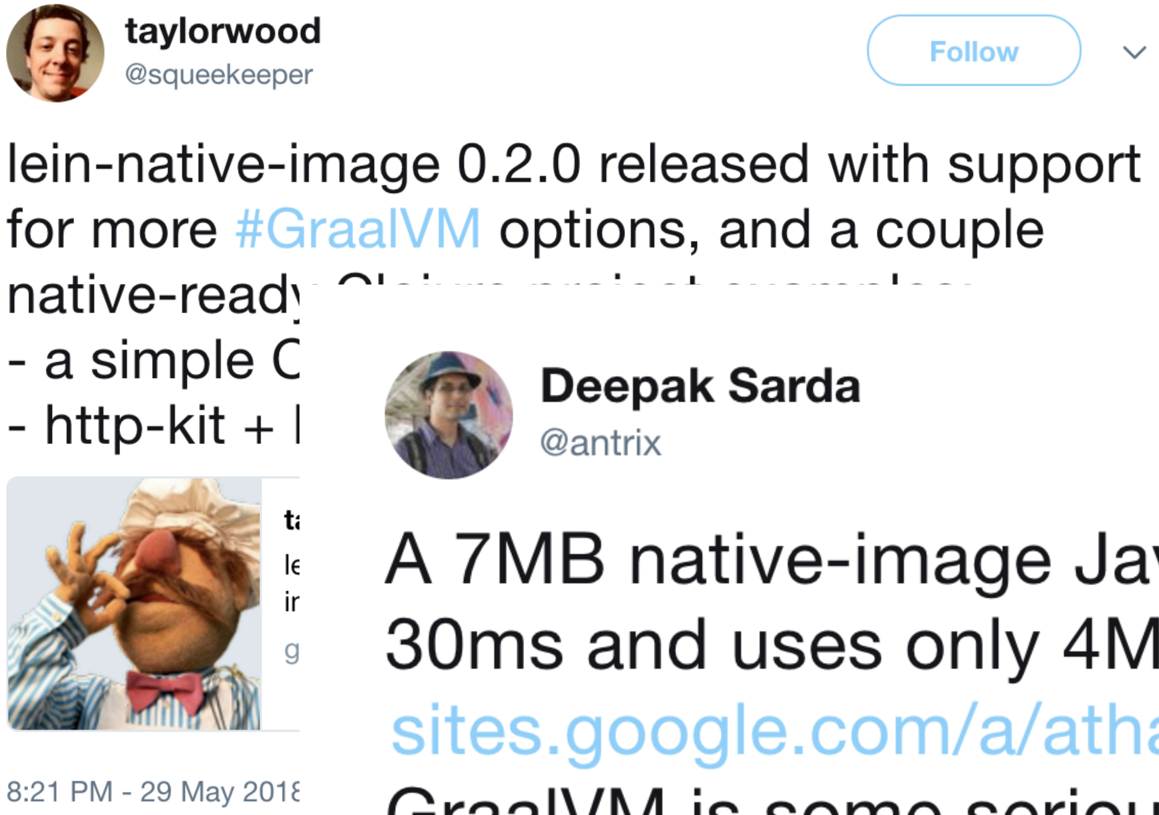
taylorwood
@squeekeeper

Follow

lein-native-image 0.2.0 released with support for more #GraalVM options, and a couple native-ready Clojure project examples:
- a simple CLI tool
- http-kit + Ring + Compojure server

taylorwood/lein-native-image
lein-native-image - A Leiningen plugin to generate GraalVM native images
github.com

8:21 PM - 29 May 2018



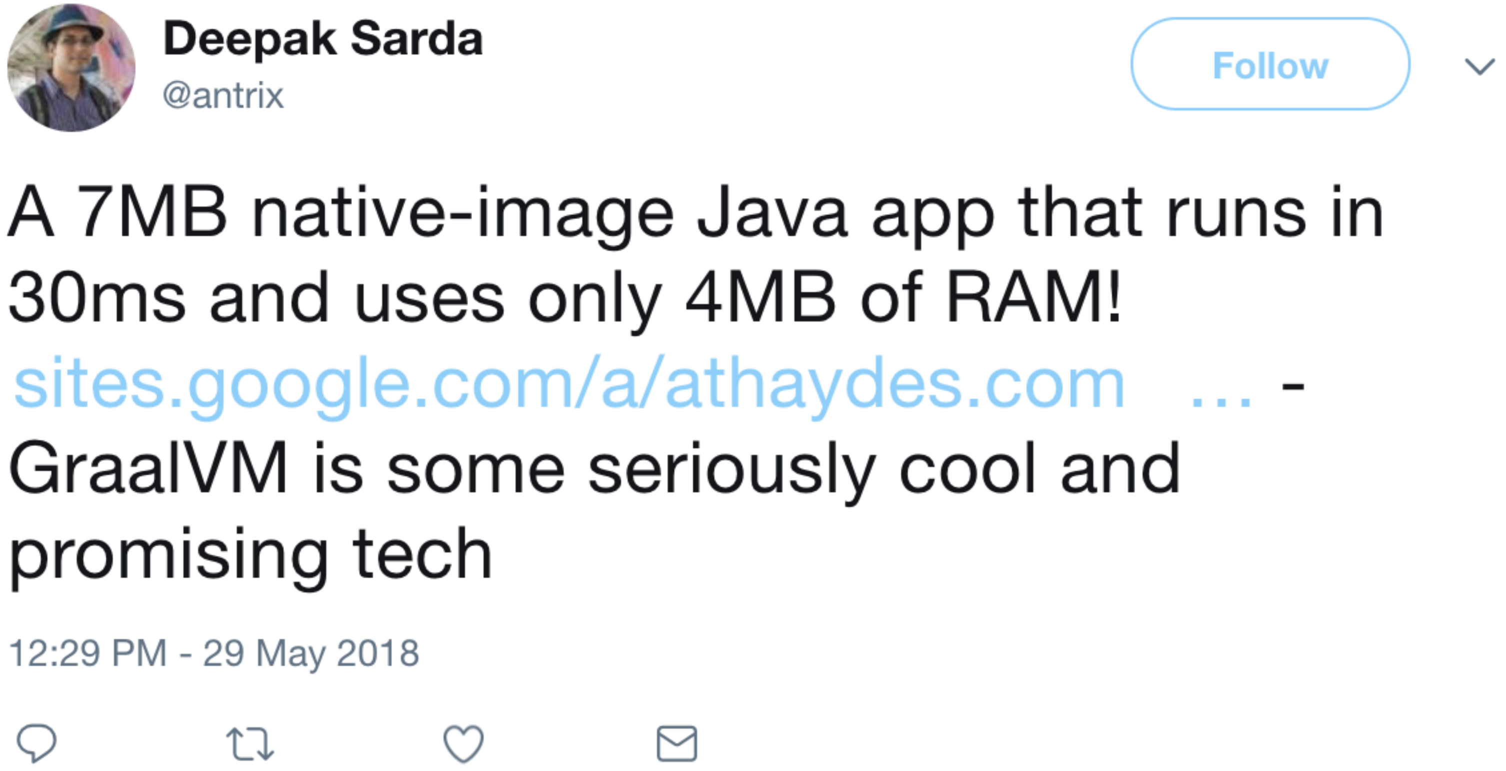
taylorwood
@squeekeeper

Follow

lein-native-image 0.2.0 released with support for more #GraalVM options, and a couple native-read
- a simple C
- http-kit + l

ta
le
ir
g

8:21 PM - 29 May 201

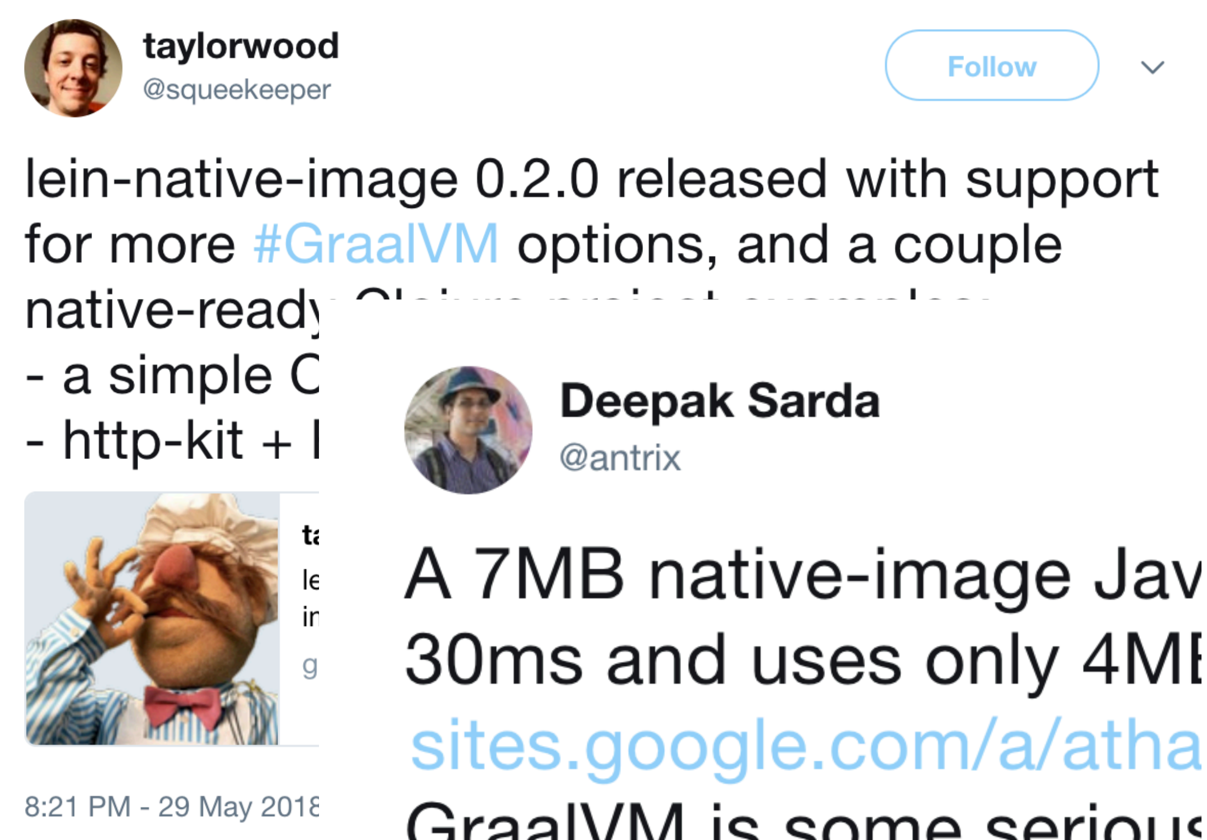


Deepak Sarda
@antrix

Follow

A 7MB native-image Java app that runs in 30ms and uses only 4MB of RAM! sites.google.com/a/athaydes.com … - GraalVM is some seriously cool and promising tech

12:29 PM - 29 May 2018



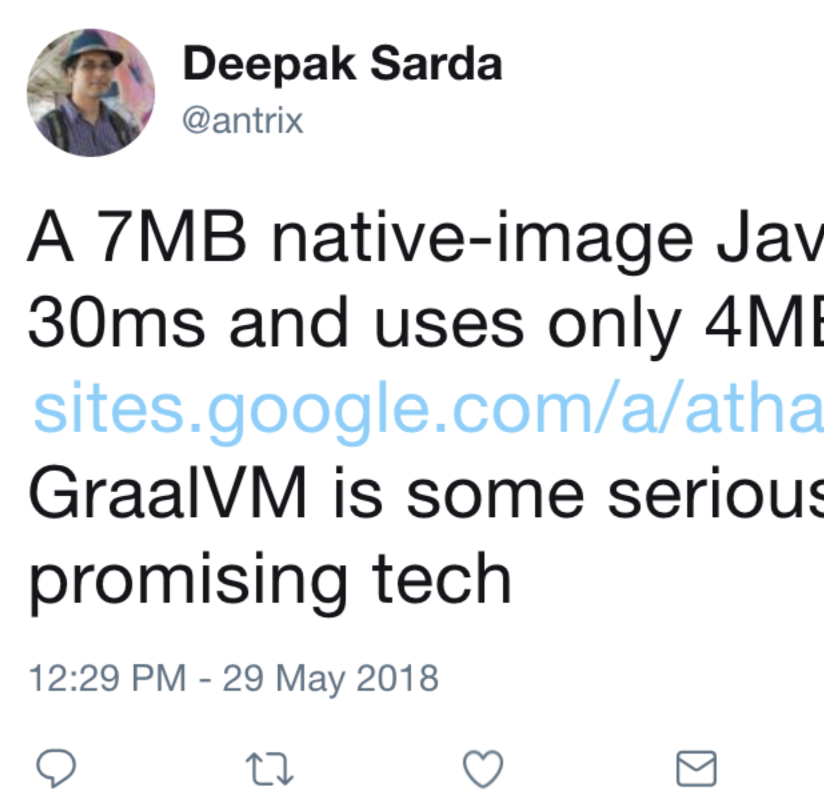
taylorwood
@squeekeeper

Follow

lein-native-image 0.2.0 released with support for more #GraalVM options, and a couple native-ready
- a simple C
- http-kit + l

ta
le
ir
g

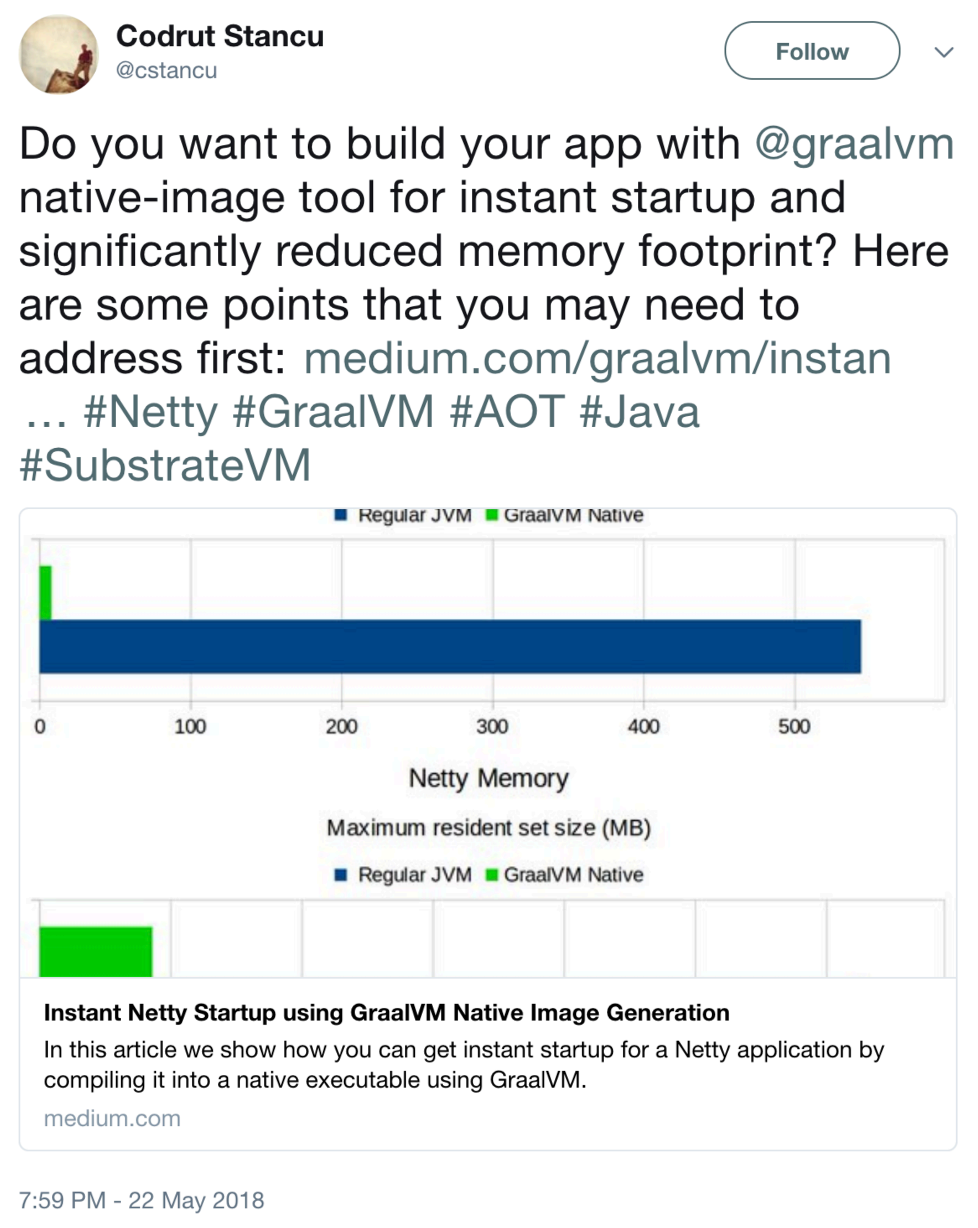
8:21 PM - 29 May 2018



Deepak Sarda
@antrix

A 7MB native-image Jav
30ms and uses only 4MI
sites.google.com/a/atha
GraalVM is some serious promising tech

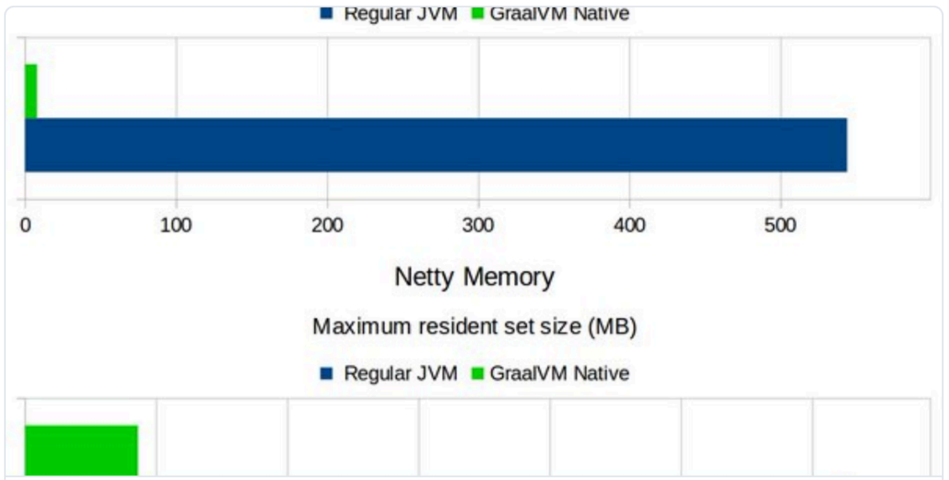
12:29 PM - 29 May 2018



Codrut Stancu
@cstancu

Follow

Do you want to build your app with @graalvm native-image tool for instant startup and significantly reduced memory footprint? Here are some points that you may need to address first: medium.com/graalvm/instan … #Netty #GraalVM #AOT #Java #SubstrateVM

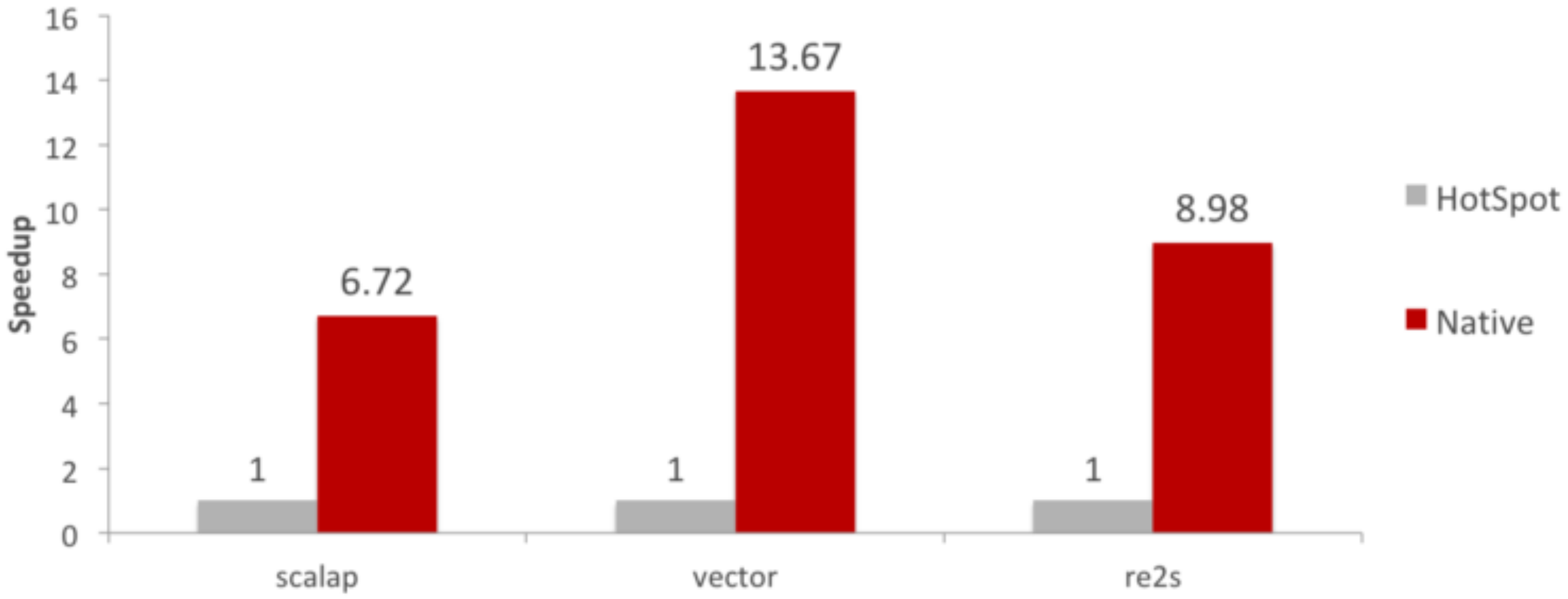


Instant Netty Startup using GraalVM Native Image Generation
In this article we show how you can get instant startup for a Netty application by compiling it into a native executable using GraalVM.
medium.com

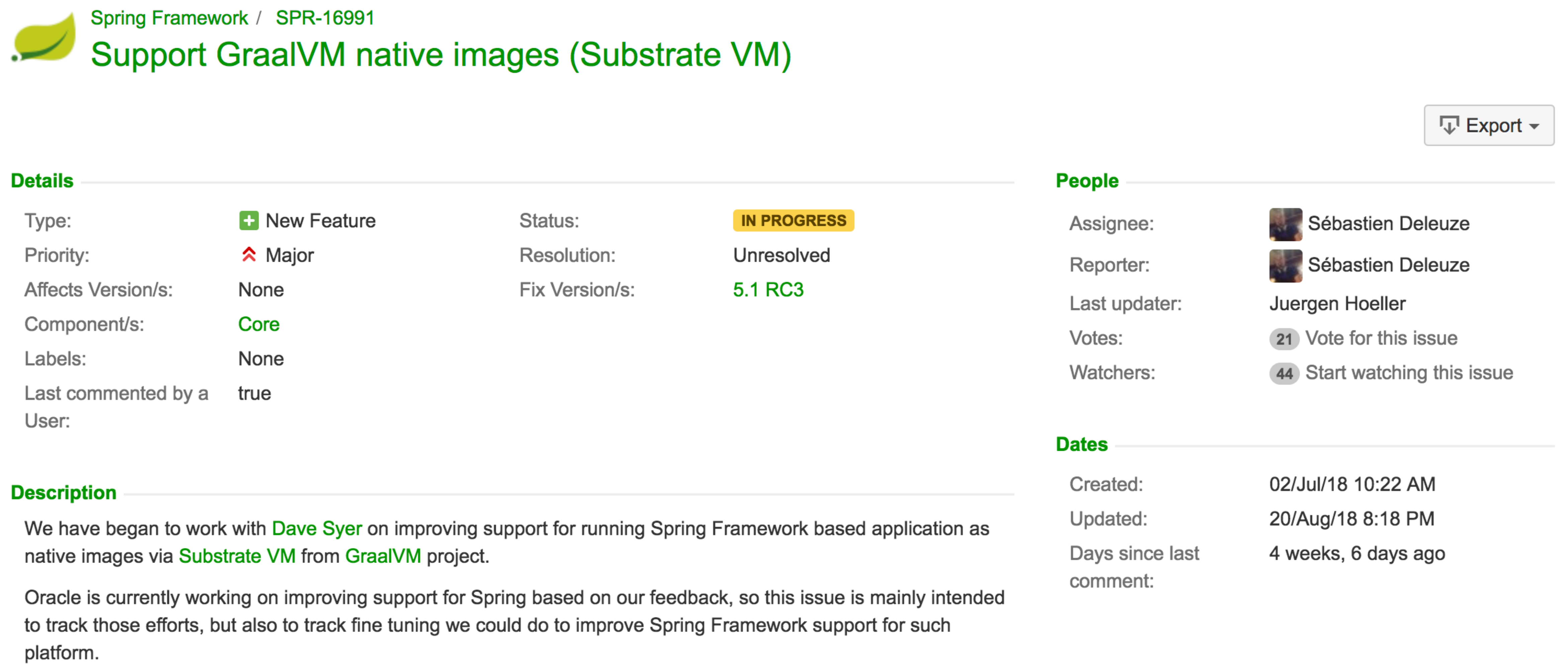
7:59 PM - 22 May 2018

Native scalac

```
git clone https://github.com/graalvm/graalvm-demos

cd graalvm-demos/scala-days-2018/scalac-native/scala-substitutions

sbt package

cd ../


$GRAALVM_HOME/bin/native-image -cp $SCALA_HOME/lib/scala-compiler.jar:
$SCALA_HOME/lib/scala-library.jar:$SCALA_HOME/lib/scala-reflect.jar:$PWD/
scalac-substitutions/target/scala-2.12/scalac-substitutions_2.12-0.1.0-
SNAPSHOT.jar \

  -H:SubstitutionResources=substitutions.json,substitutions-2.12.json \

  -H:ReflectionConfigurationFiles=scalac-substitutions/reflection-
config.json \

  -H:Class=scala.tools.nsc.Main \

  -H:Name=scalac
```



https://medium.com/graalvm/compiling-scala-faster-with-graalvm-86c5c0857fa3



Spring Framework / SPR-16991

Support GraalVM native images (Substrate VM)

Export

Details

Type:	New Feature	Status:	IN PROGRESS
Priority:	Major	Resolution:	Unresolved
Affects Version/s:	None	Fix Version/s:	5.1 RC3
Component/s:	Core		
Labels:	None		
Last commented by a User:	true		

People

Assignee:	Sébastien Deleuze
Reporter:	Sébastien Deleuze
Last updater:	Juergen Hoeller
Votes:	21 Vote for this issue
Watchers:	44 Start watching this issue

Description

We have began to work with Dave Syer on improving support for running Spring Framework based application as native images via Substrate VM from GraalVM project.

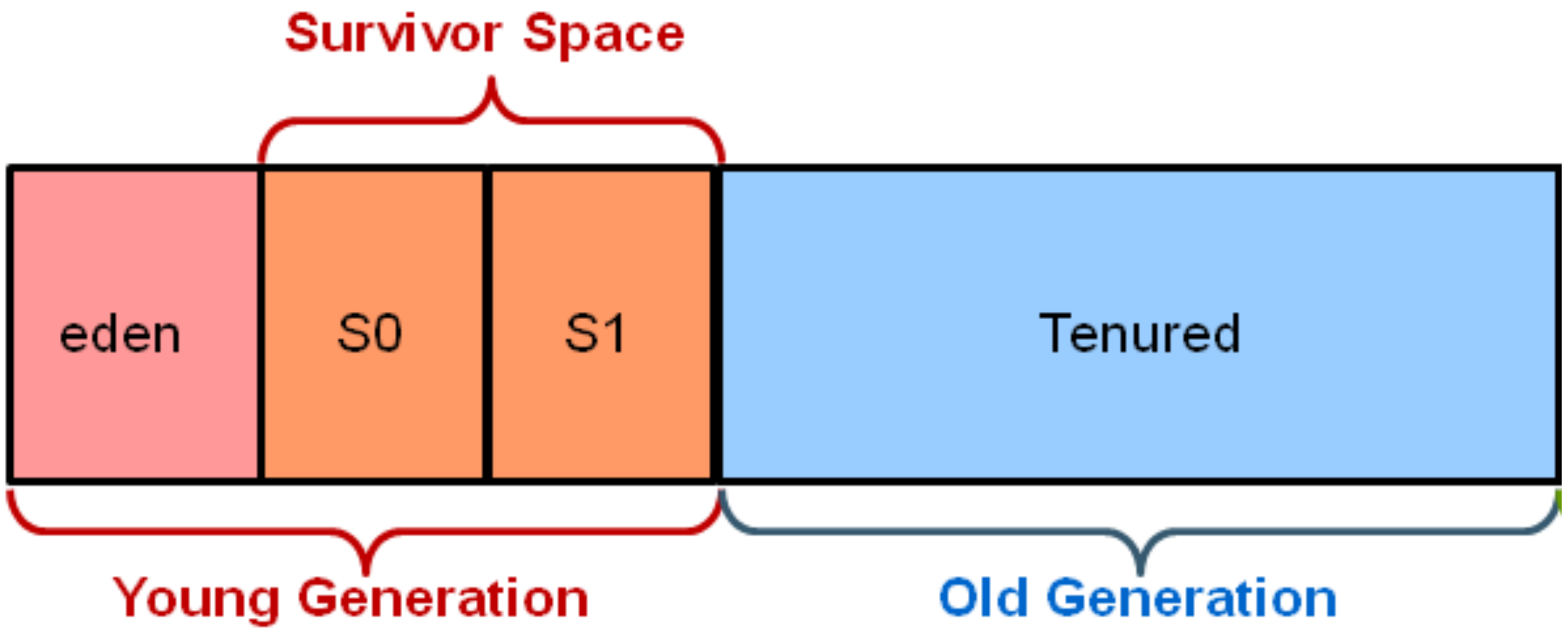
Oracle is currently working on improving support for Spring based on our feedback, so this issue is mainly intended to track those efforts, but also to track fine tuning we could do to improve Spring Framework support for such platform.

Dates

Created:	02/Jul/18 10:22 AM
Updated:	20/Aug/18 8:18 PM
Days since last comment:	4 weeks, 6 days ago

https://jira.spring.io/browse/SPR-16991

GC

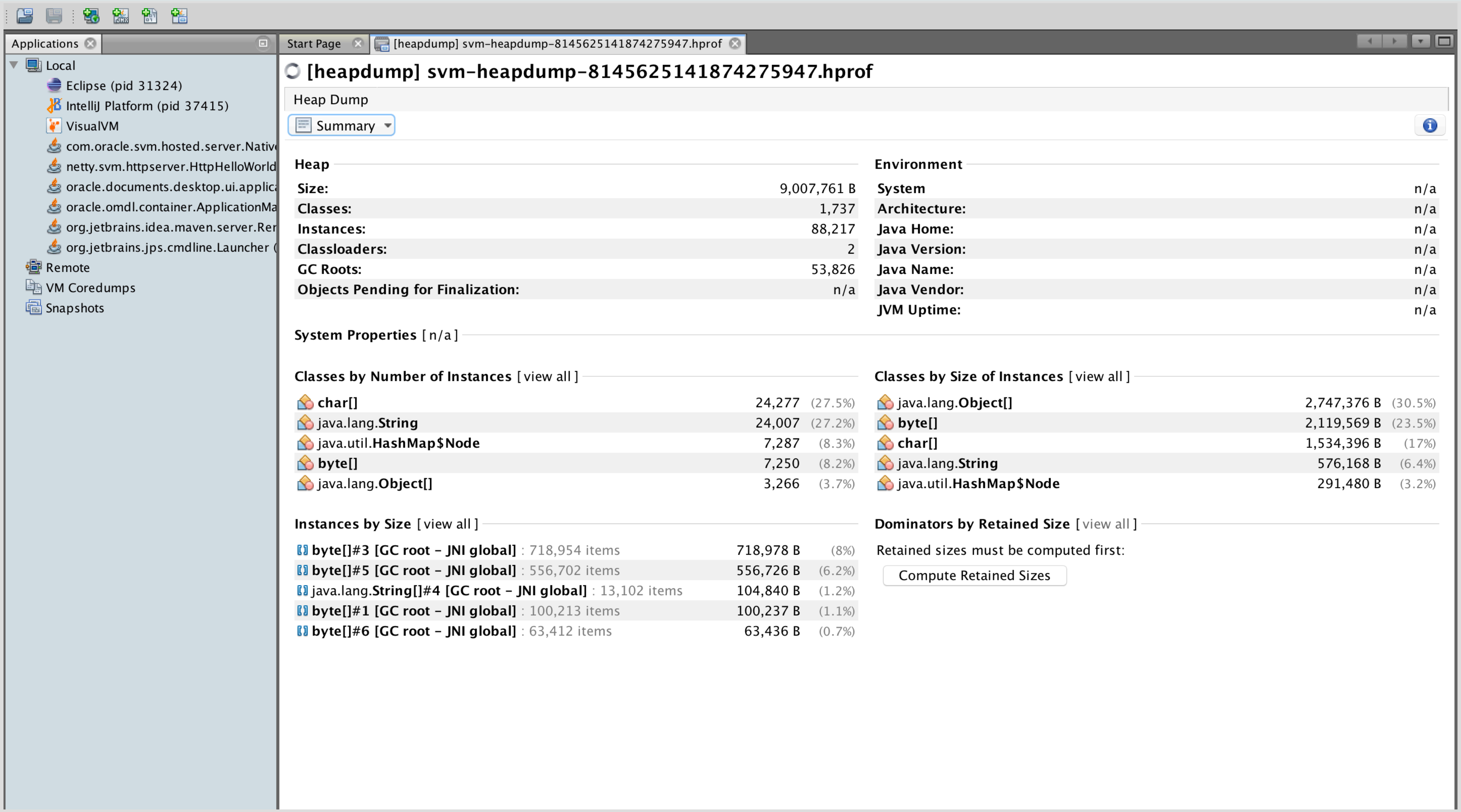

Survivor Space
eden
S0
S1
Tenured
Young Generation
Old Generation

Garbage Collection Options

- `-Xmn=` Set the size of the young generation (the amount of memory that can be allocated without triggering a GC). Value is specified in bytes, suffix `k`, `m`, or `g` can be used for scaling.
- `-Xmx=` Set the maximum heap size in bytes. Value is specified in bytes, suffix `k`, `m`, or `g` can be used for scaling. Note that this is not the maximum amount of consumed memory, because during GC the system can request more temporary memory.
- `-Xms=` Set the minimum heap size in bytes. Value is specified in bytes, suffix `k`, `m`, or `g` can be used for scaling. Heap space that is unused will be retained for future heap usage, rather than being returned to the operating system.
- `-R:[+|-]PrintGC` Print summary GC information after each collection.
- `-R:[+|-]VerboseGC` Print more information about the heap before and after each collection.

Heapdump

```
-H:+AllowVMInspection

$ ps -e | grep native-name
$ kill -SIGUSR1 <pid>
```


Applications
Start Page
[heapdump] svm-heapdump-8145625141874275947.hprof
Local
Eclipse (pid 31324)
IntelliJ Platform (pid 37415)
VisualVM
com.oracle.svm.hosted.server.Nativ
netty.svm.httpserver.HttpHelloWorld
oracle.documents.desktop.ui.applic
oracle.omdl.container.ApplicationMa
org.jetbrains.idea.maven.server.Rer
org.jetbrains.jps.cmdline.Launcher (
Remote
VM Coredumps
Snapshots
Heap Dump
Summary

[heapdump] svm-heapdump-8145625141874275947.hprof

Heap

Size:	9,007,761 B
Classes:	1,737
Instances:	88,217
Classloaders:	2
GC Roots:	53,826
Objects Pending for Finalization:	n/a

Environment

System	n/a
Architecture:	n/a
Java Home:	n/a
Java Version:	n/a
Java Name:	n/a
Java Vendor:	n/a
JVM Uptime:	n/a

System Properties [n/a]

Classes by Number of Instances [view all]

char[]	24,277	(27.5%)
java.lang.String	24,007	(27.2%)
java.util.HashMap$Node	7,287	(8.3%)
byte[]	7,250	(8.2%)
java.lang.Object[]	3,266	(3.7%)

Classes by Size of Instances [view all]

java.lang.Object[]	2,747,376 B	(30.5%)
byte[]	2,119,569 B	(23.5%)
char[]	1,534,396 B	(17%)
java.lang.String	576,168 B	(6.4%)
java.util.HashMap$Node	291,480 B	(3.2%)

Instances by Size [view all]

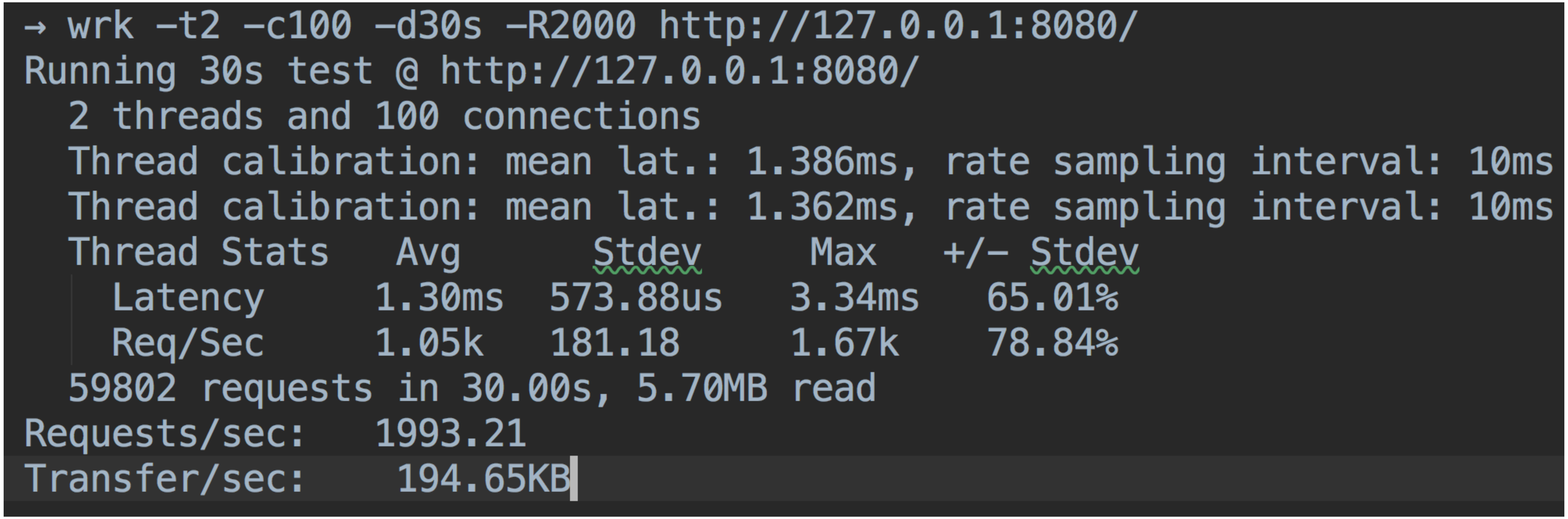
byte[]#3 [GC root – JNI global] : 718,954 items	718,978 B	(8%)
byte[]#5 [GC root – JNI global] : 556,702 items	556,726 B	(6.2%)
java.lang.String[]#4 [GC root – JNI global] : 13,102 items	104,840 B	(1.2%)
byte[]#1 [GC root – JNI global] : 100,213 items	100,237 B	(1.1%)
byte[]#6 [GC root – JNI global] : 63,412 items	63,436 B	(0.7%)

Dominators by Retained Size [view all]

Retained sizes must be computed first:

Compute Retained Sizes

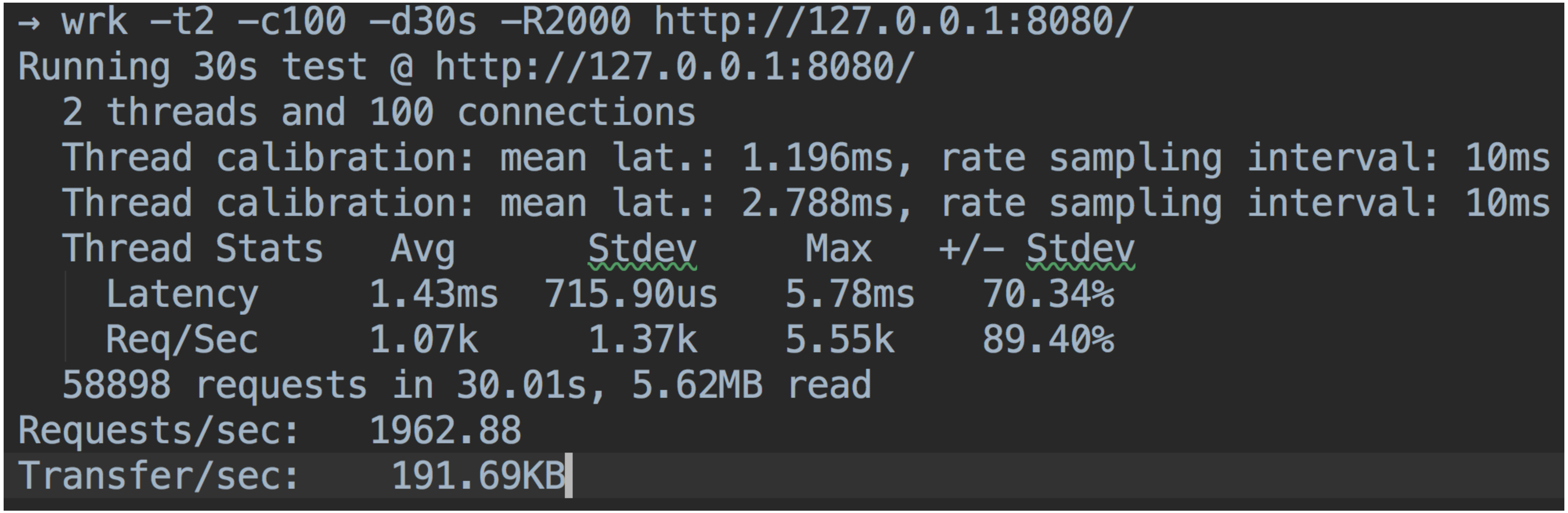
Runtime



```
→ wrk -t2 -c100 -d30s -R2000 http://127.0.0.1:8080/
Running 30s test @ http://127.0.0.1:8080/
  2 threads and 100 connections
  Thread calibration: mean lat.: 1.386ms, rate sampling interval: 10ms
  Thread calibration: mean lat.: 1.362ms, rate sampling interval: 10ms
  Thread Stats   Avg      Stdev     Max   +/- Stdev
    Latency     1.30ms  573.88us   3.34ms   65.01%
    Req/Sec     1.05k   181.18     1.67k    78.84%
  59802 requests in 30.00s, 5.70MB read
Requests/sec:   1993.21
Transfer/sec:    194.65KB
```

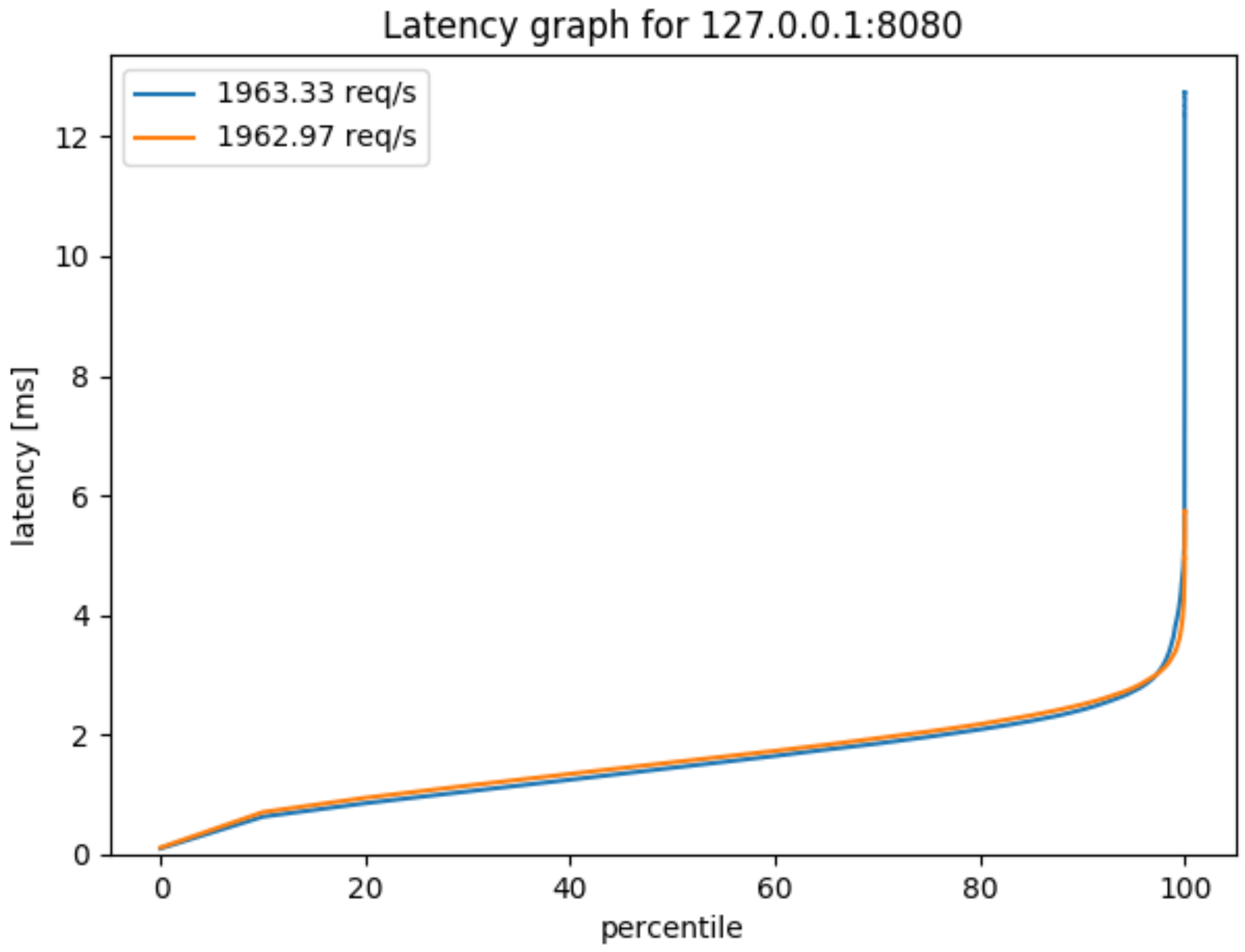
https://github.com/giltene/wrk2

Runtime

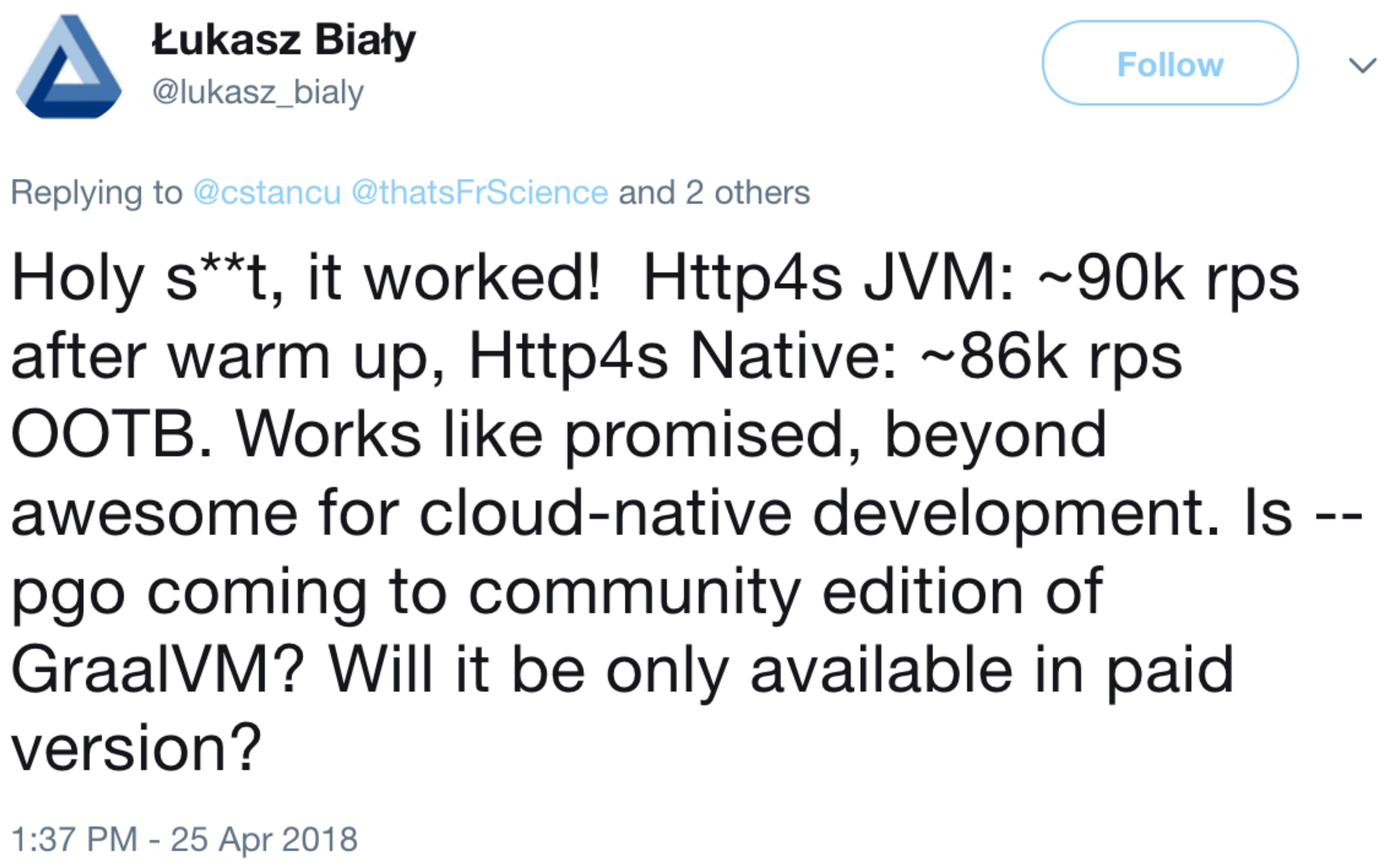


```
→ wrk -t2 -c100 -d30s -R2000 http://127.0.0.1:8080/
Running 30s test @ http://127.0.0.1:8080/
  2 threads and 100 connections
  Thread calibration: mean lat.: 1.196ms, rate sampling interval: 10ms
  Thread calibration: mean lat.: 2.788ms, rate sampling interval: 10ms
  Thread Stats   Avg      Stdev     Max   +/- Stdev
    Latency     1.43ms  715.90us   5.78ms   70.34%
    Req/Sec     1.07k     1.37k    5.55k    89.40%
  58898 requests in 30.01s, 5.62MB read
Requests/sec:   1962.88
Transfer/sec:    191.69KB
```

https://github.com/giltene/wrk2



https://github.com/PPACI/wrk2img


Łukasz Biały
@lukasz_bialy

Replying to @cstancu @thatsFrScience and 2 others

Holy s**t, it worked! Http4s JVM: ~90k rps after warm up, Http4s Native: ~86k rps OOTB. Works like promised, beyond awesome for cloud-native development. Is --pgo coming to community edition of GraalVM? Will it be only available in paid version?

1:37 PM - 25 Apr 2018

GraalVM™

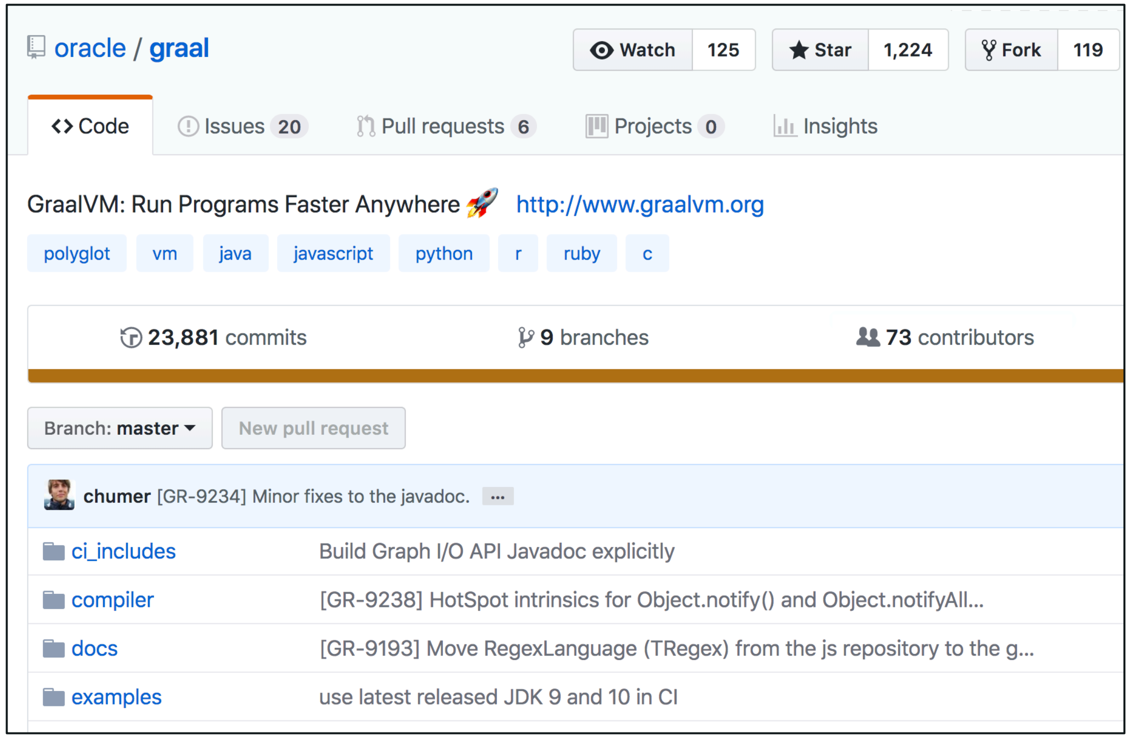
High performance, polyglot, language-level virtualization layer…

embeddable across the stack

in native and JVM-based applications.

Building a Universal VM is a Community Effort

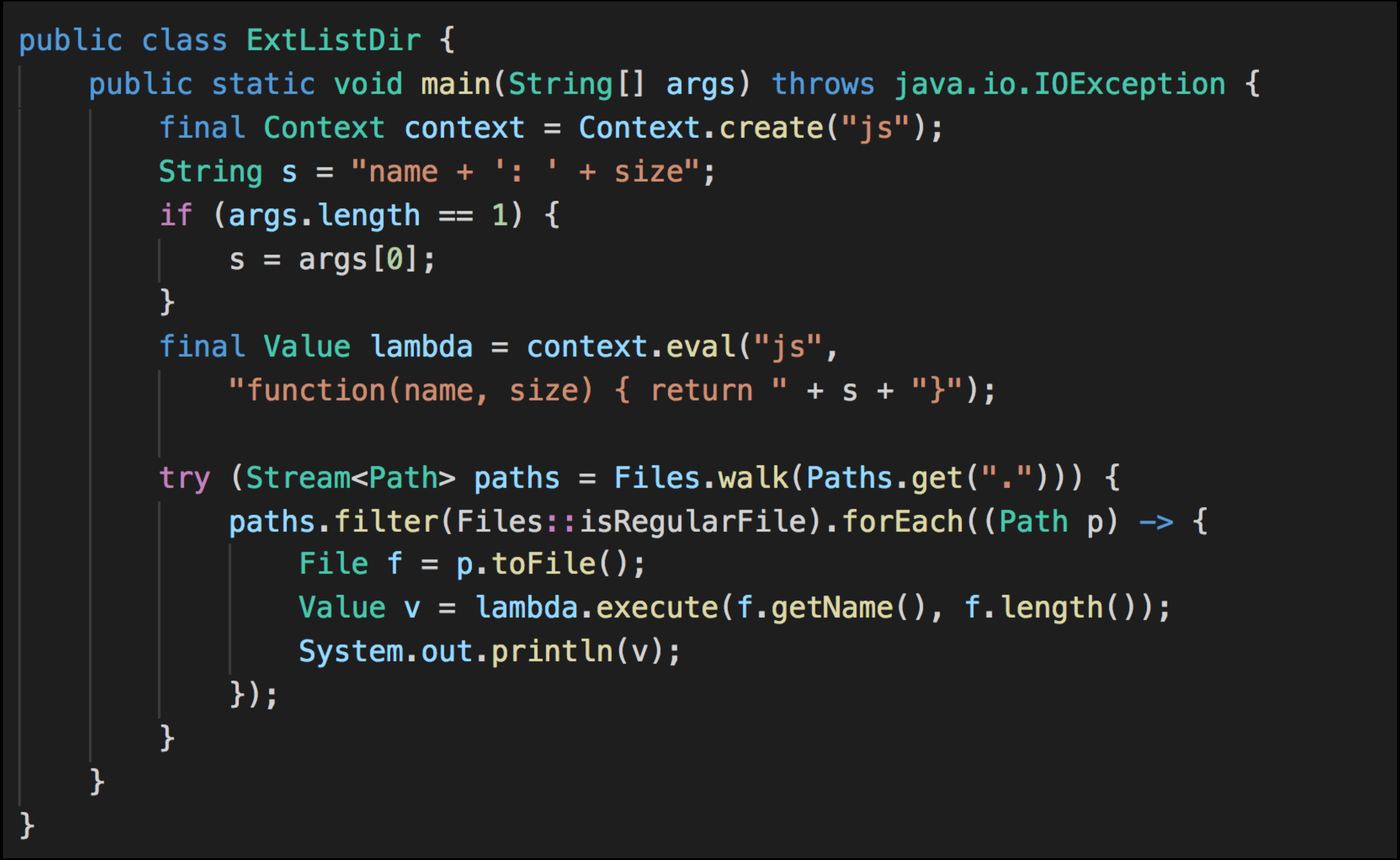
- Test your applications with GraalVM
  - Documentation and downloads at https://www.graalvm.org
- Connect your technology with GraalVM
  - Integrate GraalVM into your application
  - Run your own programming language or DSL
  - Build language-agnostic tools
- Join the conversation
  - Report issues or pull requests on GitHub
  - graalvm-users@oss.oracle.com
  - Follow @graalvm



GraalVM: Run Programs Faster Anywhere 🚀 http://www.graalvm.org

chumer [GR-9234] Minor fixes to the javadoc.

ci_includes	Build Graph I/O API Javadoc explicitly
compiler	[GR-9238] HotSpot intrinsics for Object.notify() and Object.notifyAll...
docs	[GR-9193] Move RegexLanguage (TRegex) from the js repository to the g...
examples	use latest released JDK 9 and 10 in CI



```
public class ExtListDir {
    public static void main(String[] args) throws java.io.IOException {
        final Context context = Context.create("js");
        String s = "name + ': ' + size";
        if (args.length == 1) {
            s = args[0];
        }
        final Value lambda = context.eval("js",
            "function(name, size) { return " + s + "}");

        try (Stream<Path> paths = Files.walk(Paths.get("."))) {
            paths.filter(Files::isRegularFile).forEach((Path p) -> {
                File f = p.toFile();
                Value v = lambda.execute(f.getName(), f.length());
                System.out.println(v);
            });
        }
    }
}
```