ORACLE

New Opportunities for Developers with GraalVM

Alina Yurenko

GraalVM Developer Advocate

Oracle Labs

November 01, 2019

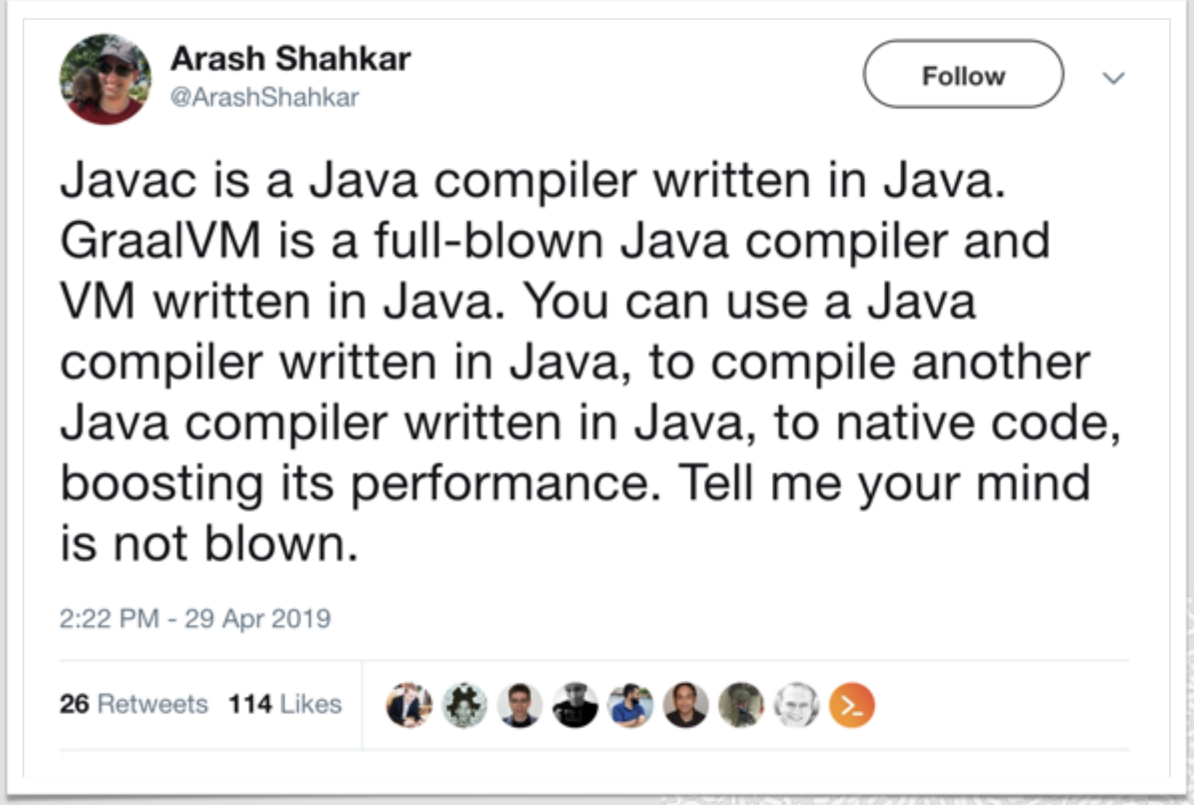
Safe harbor statement

The following is intended to outline our general product direction. 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, timing, and pricing of any features or functionality described for Oracle's products may change and remains at the sole discretion of Oracle Corporation.

GraalVM Native Image Early Adopter Status

GraalVM Native Image technology (including SubstrateVM) is early adopter technology. It is available only under an early adopter license and remains subject to potentially significant further changes, compatibility testing and certification

GraalVM magic in one tweet



Arash Shahkar
@ArashShahkar

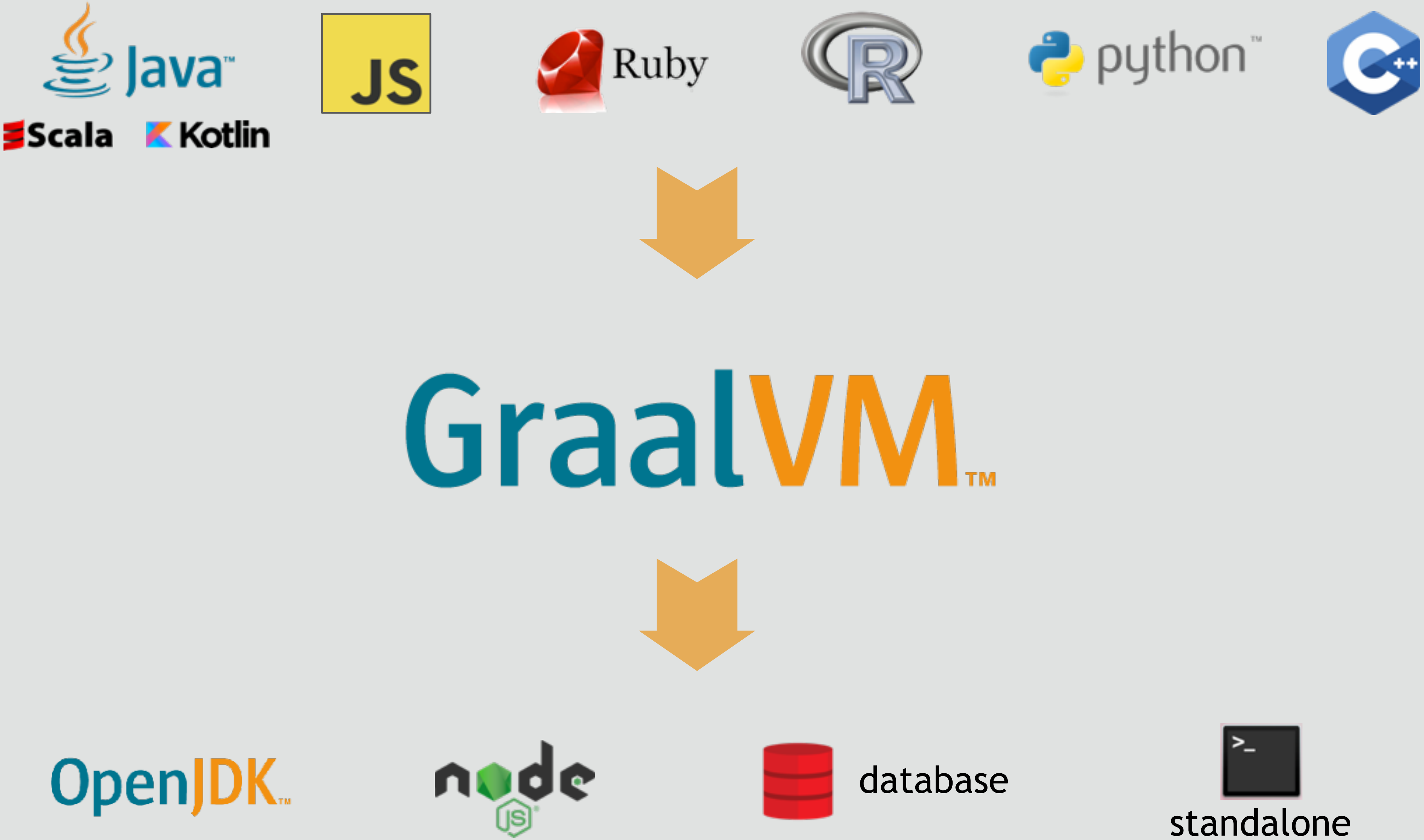
Javac is a Java compiler written in Java. GraalVM is a full-blown Java compiler and VM written in Java. You can use a Java compiler written in Java, to compile another Java compiler written in Java, to native code, boosting its performance. Tell me your mind is not blown.

2:22 PM - 29 Apr 2019

26 Retweets 114 Likes

GraalVM Project Goals

- High performance for abstractions of any language
- Low-footprint ahead-of-time mode for JVM-based languages
- Convenient language interoperability and polyglot tooling
- Simple embeddability in native and managed programs



database

standalone

What GraalVM offers

High Performance

Optimize application performance with GraalVM compiler

Fast Startup

Compile your application AOT and start instantly

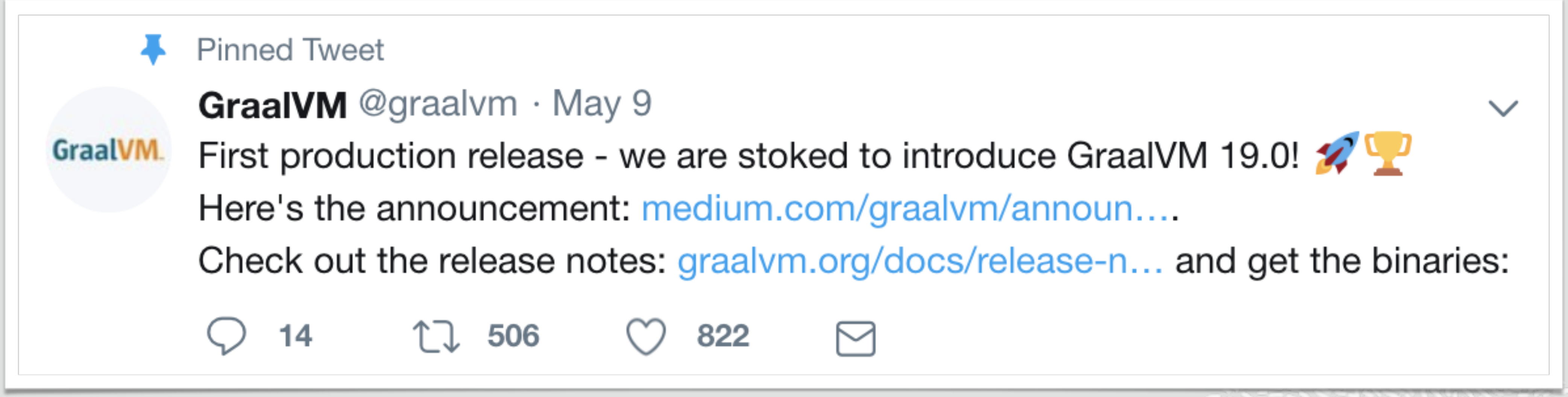
Polyglot

Mix & match languages with seamless interop

Open Source

See what’s inside, track features progress, contribute

Production-ready!🎉



📌 Pinned Tweet

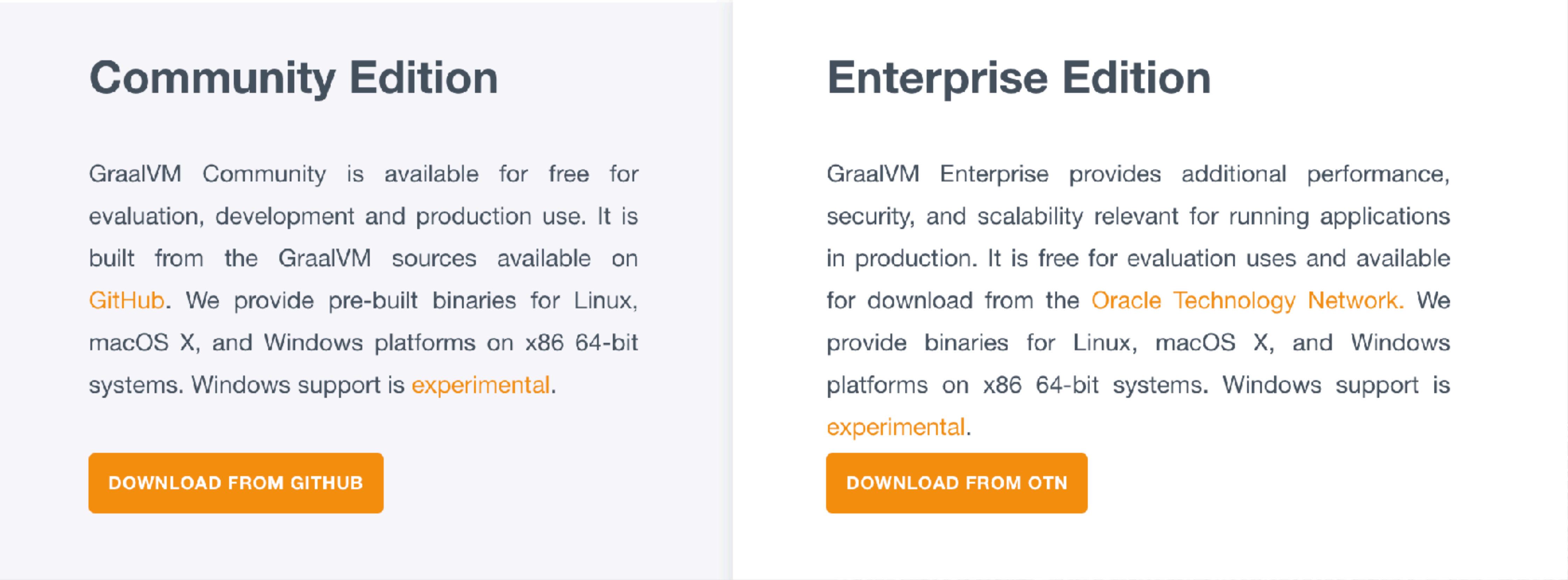
GraalVM @graalvm · May 9

First production release - we are stoked to introduce GraalVM 19.0! 🚀🏆

Here's the announcement: medium.com/graalvm/announ….

Check out the release notes: graalvm.org/docs/release-n… and get the binaries:

GraalVM Versions



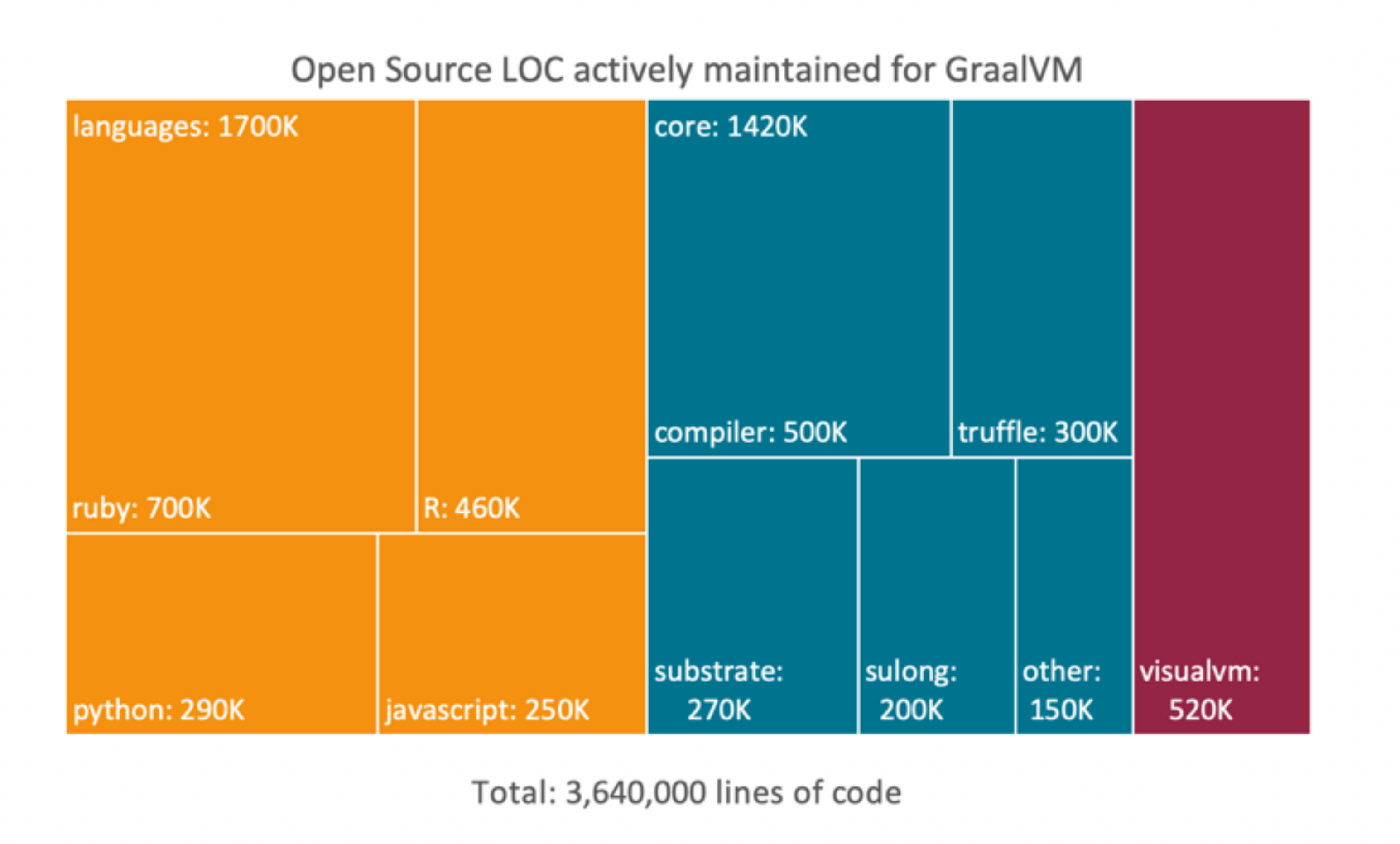
Community Edition

GraalVM Community is available for free for evaluation, development and production use. It is built from the GraalVM sources available on GitHub. We provide pre-built binaries for Linux, macOS X, and Windows platforms on x86 64-bit systems. Windows support is experimental.

Enterprise Edition

GraalVM Enterprise provides additional performance, security, and scalability relevant for running applications in production. It is free for evaluation uses and available for download from the Oracle Technology Network. We provide binaries for Linux, macOS X, and Windows platforms on x86 64-bit systems. Windows support is experimental.

GraalVM Open Source

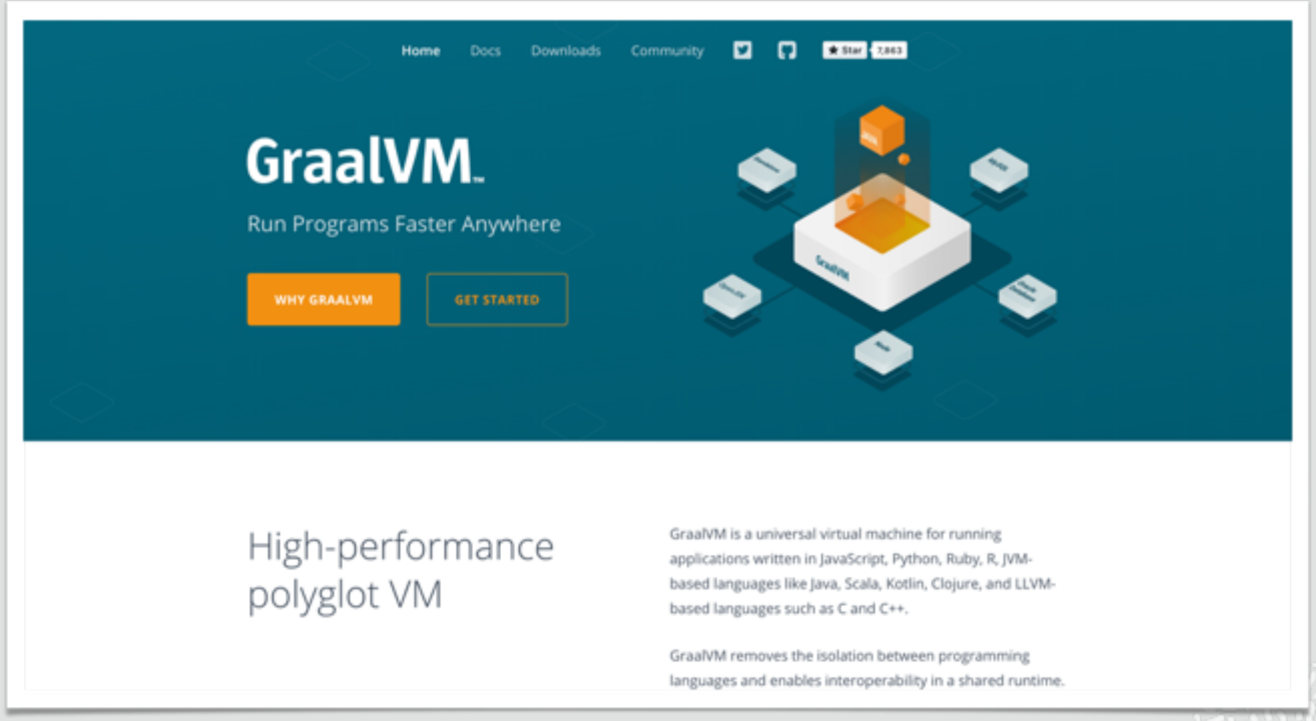


Contributions are welcome!

How to contribute:

- Report an issue: https://github.com/oracle/graal/issues
- Submit your PR: https://github.com/oracle/graal/pulls
- Extend libraries support: graalvm.org/docs/reference-manual/compatibility/
- Contribute to documentation: https://www.graalvm.org/docs/

Get Started



GraalVM

Run Programs Faster Anywhere

High-performance polyglot VM

GraalVM is a universal virtual machine for running applications written in JavaScript, Python, Ruby, R, JVM-based languages like Java, Scala, Kotlin, Clojure, and LLVM-based languages such as C and C++.

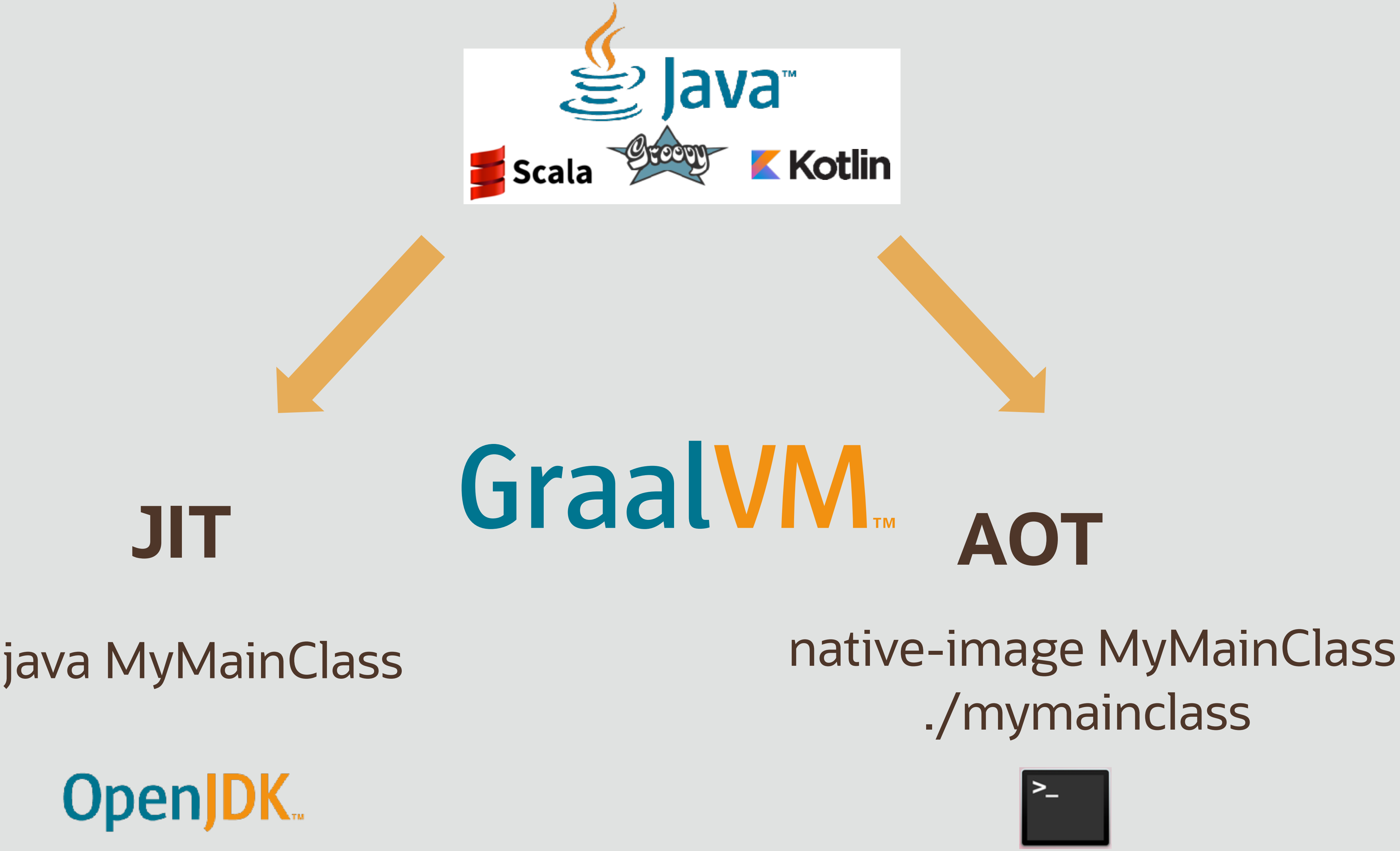
GraalVM removes the isolation between programming languages and enables interoperability in a shared runtime.

- Downloads
- Documentation
- Community support

For Java & JVM programs

GraalVM Compiler

- Brand new compiler written itself in Java;
- Adds new optimizations on top of traditional ones;
- Supports multiple languages and platforms;
- Can work in JIT & AOT modes.



Java

Scala Groovy Kotlin

GraalVM

JIT

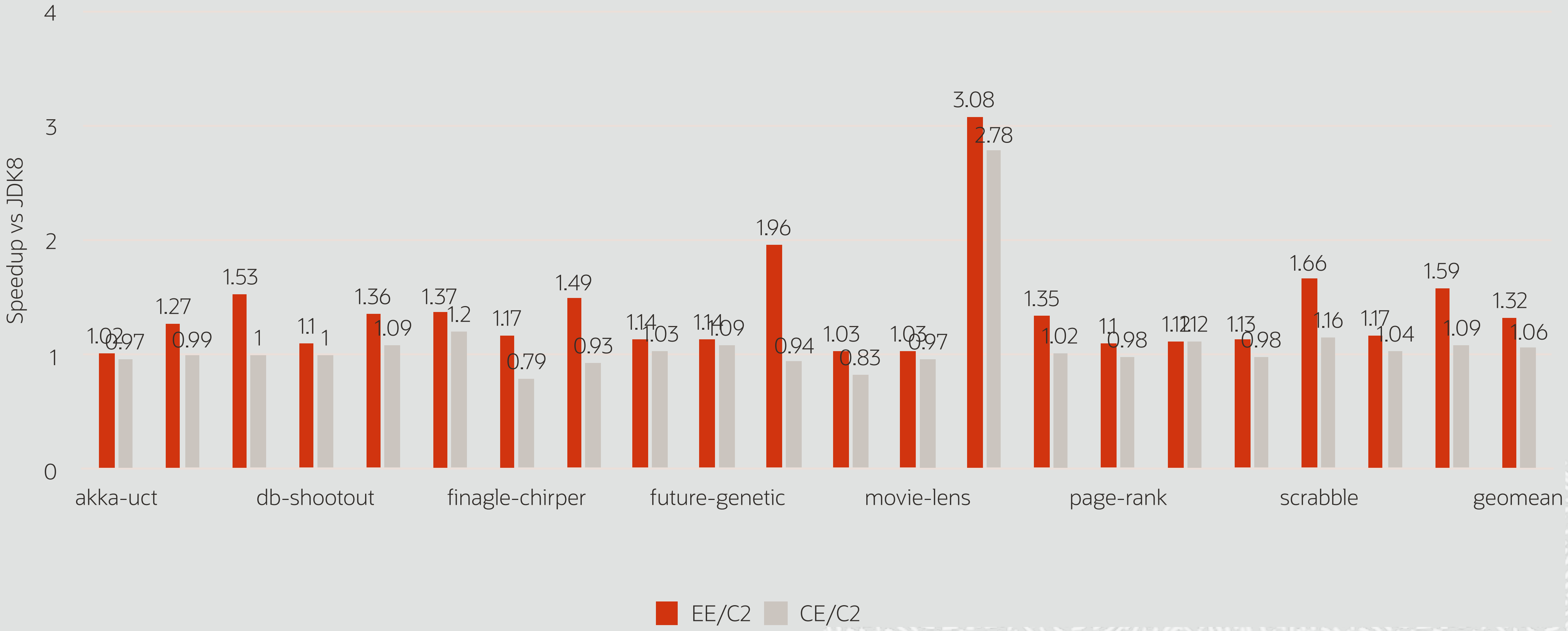
java MyMainClass

OpenJDK

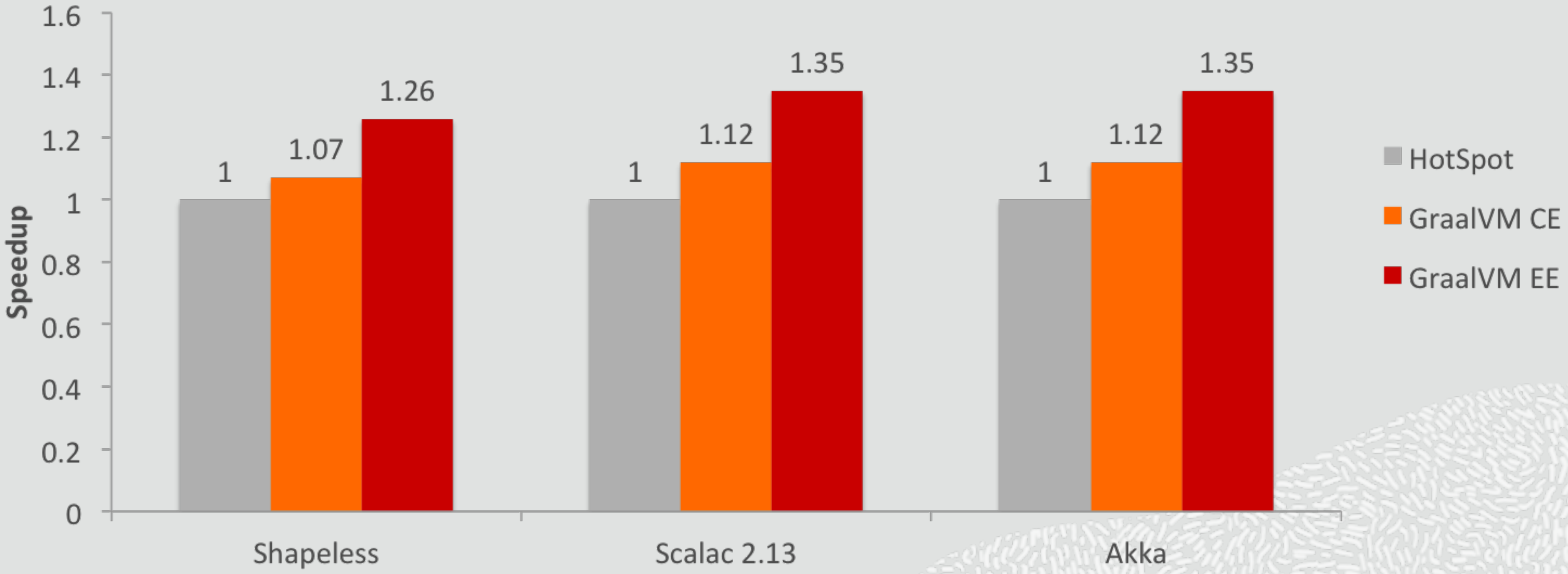
AOT

native-image MyMainClass
./mymainclass

GraalVM JIT Performance: Renaissance.dev



Scala performance



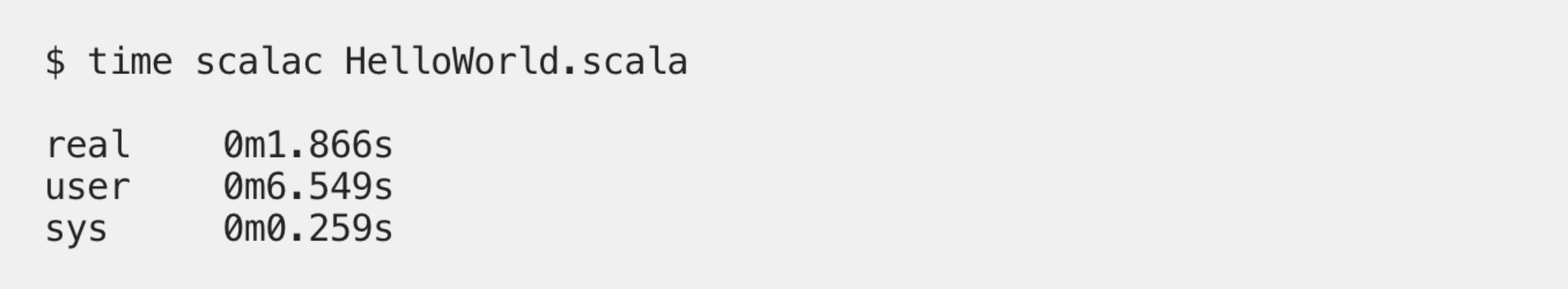
https://medium.com/graalvm/compiling-scala-faster-with-graalvm-86c5c0857fa3

GraalVM Native Images

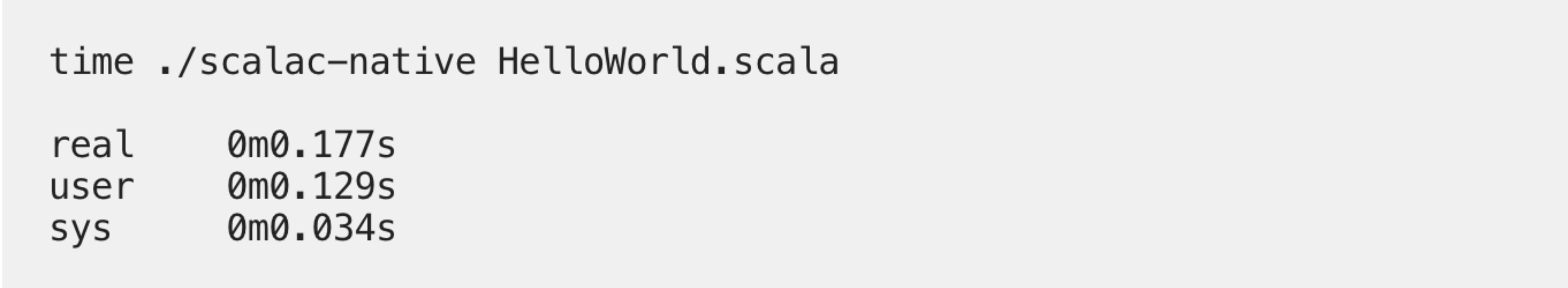
- Instant startup;
- Low memory footprint;
- AOT-compiled using the GraalVM compiler;
- Great for microservices.

Demo time

Native Image startup time example

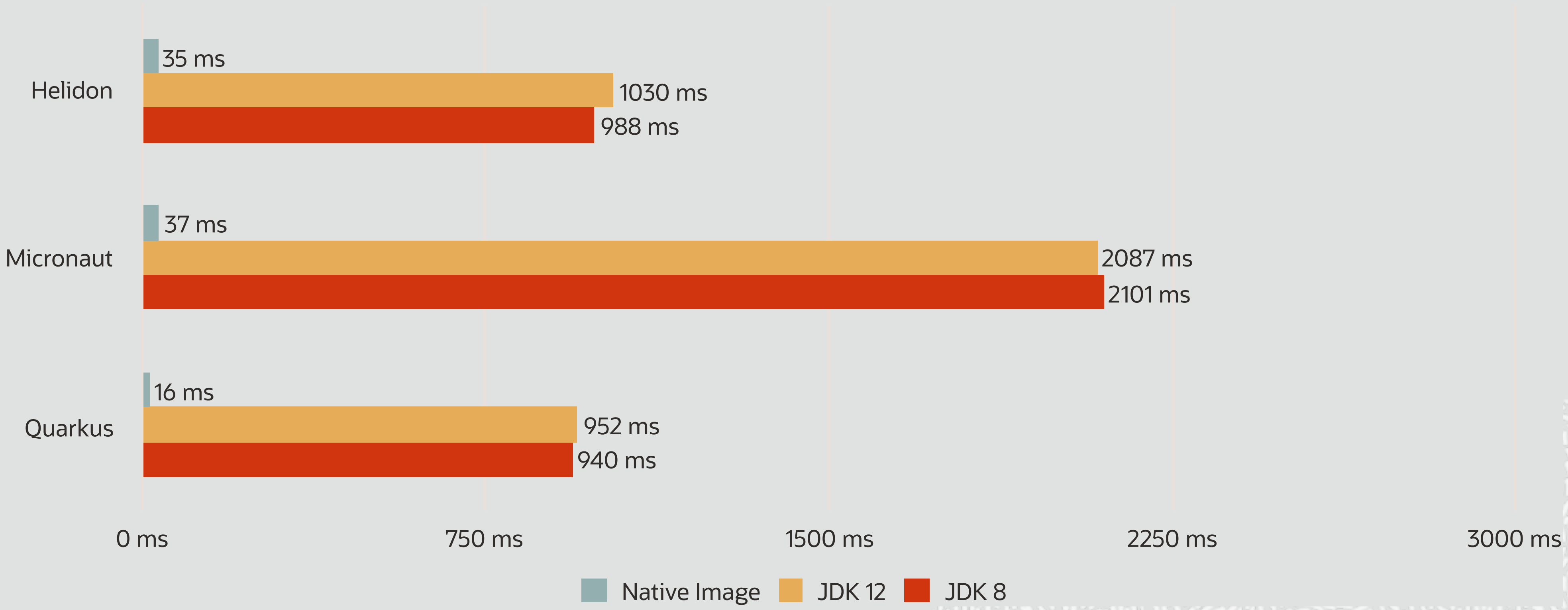


```
$ time scalac HelloWorld.scala

real    0m1.866s
user    0m6.549s
sys     0m0.259s
```

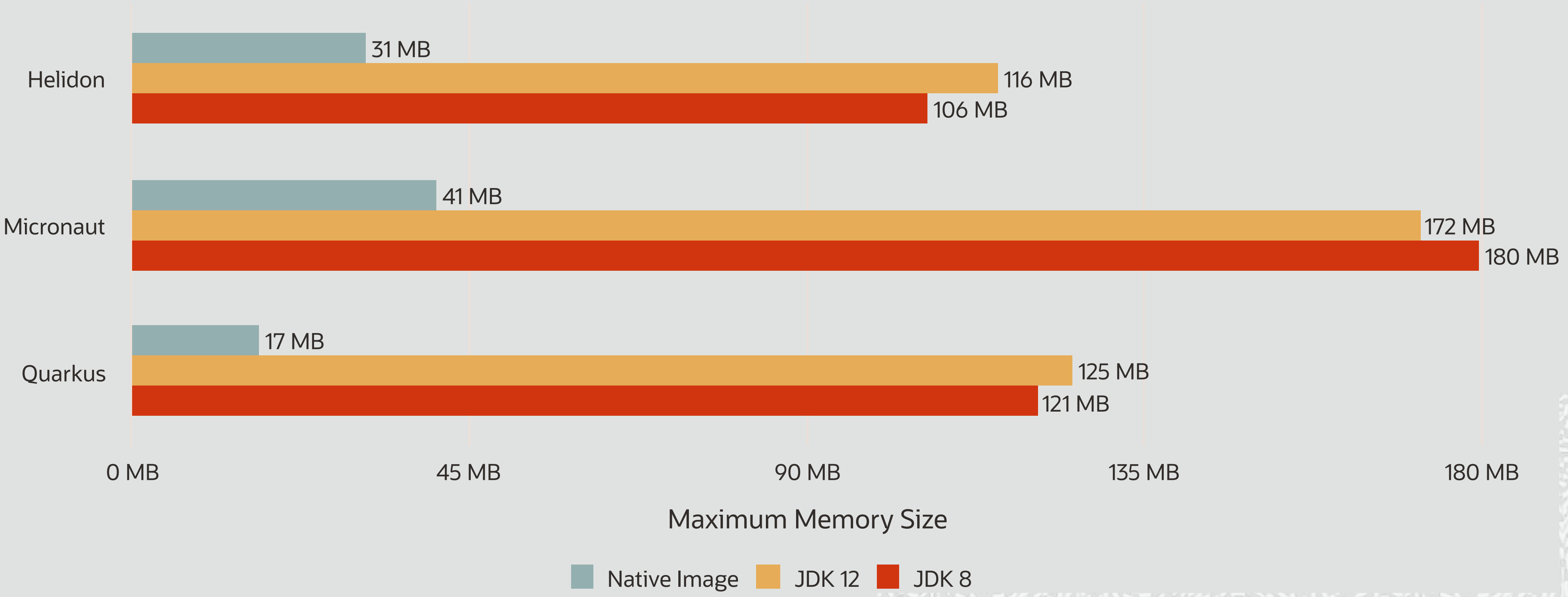


```
time ./scalac-native HelloWorld.scala

real    0m0.177s
user    0m0.129s
sys     0m0.034s
```

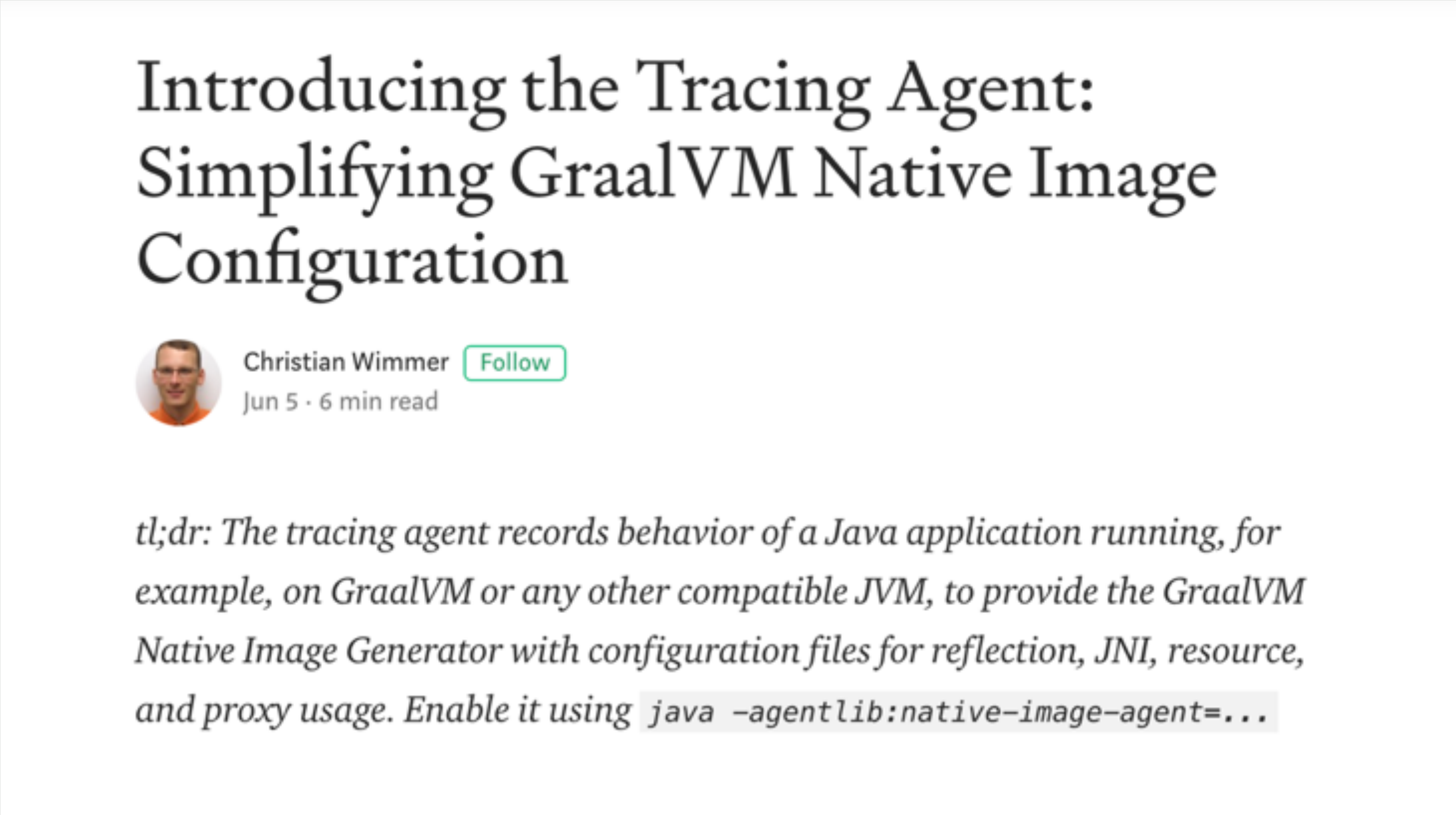
Microservice Frameworks: Startup Time



Microservice Frameworks: Memory Usage



Simplifying the Native Image Configuration



Introducing the Tracing Agent: Simplifying GraalVM Native Image Configuration

Christian Wimmer

Jun 5 · 6 min read

tl;dr: The tracing agent records behavior of a Java application running, for example, on GraalVM or any other compatible JVM, to provide the GraalVM Native Image Generator with configuration files for reflection, JNI, resource, and proxy usage. Enable it using `java -agentlib:native-image-agent=...`

Continue Learning About GraalVM Native Images

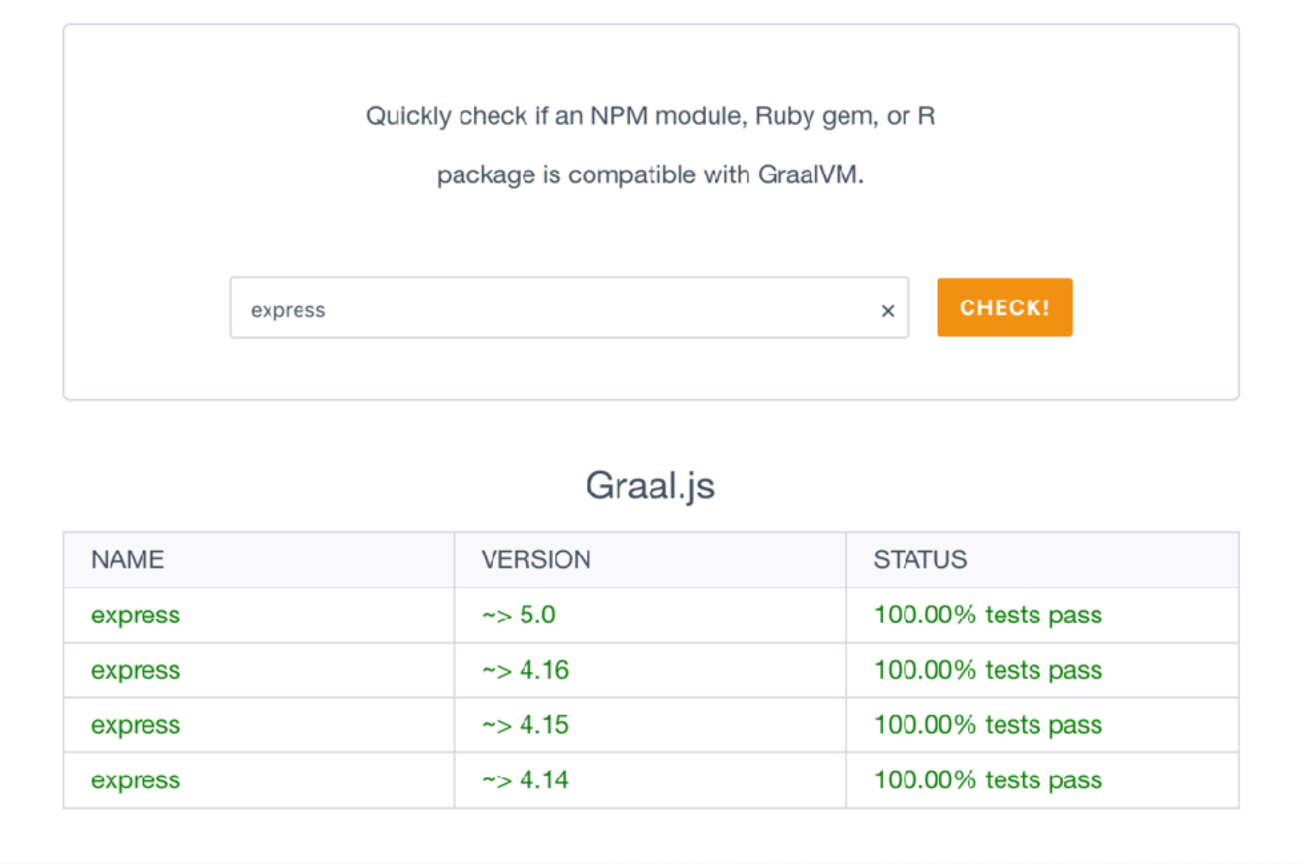
- Reference manual: graalvm.org/docs/reference-manual/aot-compilation/
- Improving performance of GraalVM native images with PGO: https://medium.com/graalvm/improving-performance-of-graalvm-native-images-with-profile-guided-optimizations-9c431a834edb
- GraalVM Native Images: The Best Startup Solution for Your Applications: https://www.youtube.com/watch?v=z0jedLjcWjI

JavaScript & Node.js programs

JavaScript & Node.js

- ECMAScript 2019 complaint JavaScript engine;
- Access to GraalVM language interoperability and common tooling;
- Constantly tested against 100,000+ npm modules, `including express, react, async, request`

Compatibility Tool



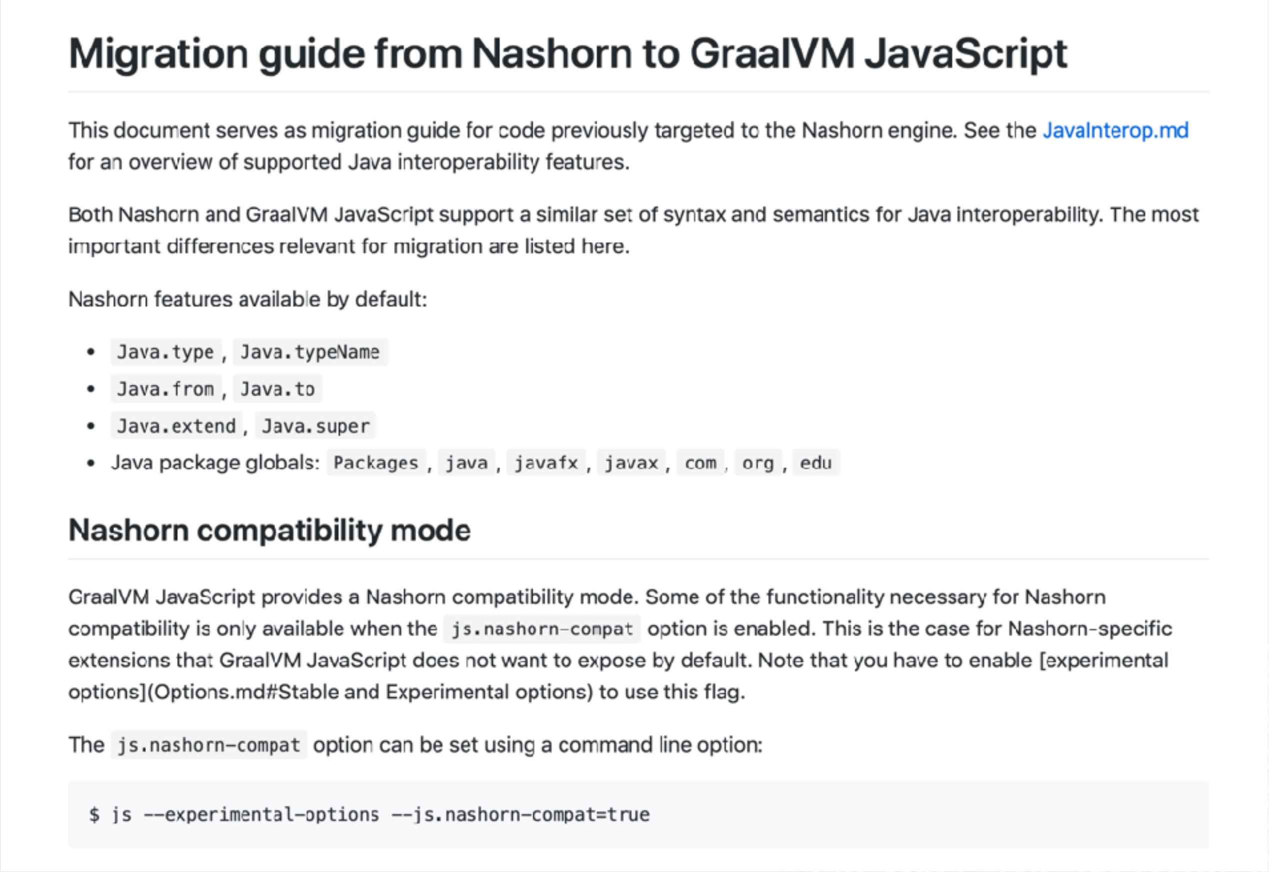
Quickly check if an NPM module, Ruby gem, or R package is compatible with GraalVM.

Graal.js

NAME	VERSION	STATUS
express	~> 5.0	100.00% tests pass
express	~> 4.16	100.00% tests pass
express	~> 4.15	100.00% tests pass
express	~> 4.14	100.00% tests pass

https://www.graalvm.org/docs/reference-manual/compatibility

Nashorn Migration Guide



Migration guide from Nashorn to GraalVM JavaScript

This document serves as migration guide for code previously targeted to the Nashorn engine. See the JavaInterop.md for an overview of supported Java interoperability features.

Both Nashorn and GraalVM JavaScript support a similar set of syntax and semantics for Java interoperability. The most important differences relevant for migration are listed here.

Nashorn features available by default:

- `Java.type`, `Java.typeName`
- `Java.from`, `Java.to`
- `Java.extend`, `Java.super`
- Java package globals: `Packages`, `java`, `javafx`, `javax`, `com`, `org`, `edu`

Nashorn compatibility mode

GraalVM JavaScript provides a Nashorn compatibility mode. Some of the functionality necessary for Nashorn compatibility is only available when the `js.nashorn-compat` option is enabled. This is the case for Nashorn-specific extensions that GraalVM JavaScript does not want to expose by default. Note that you have to enable [experimental options](Options.md#Stable and Experimental options) to use this flag.

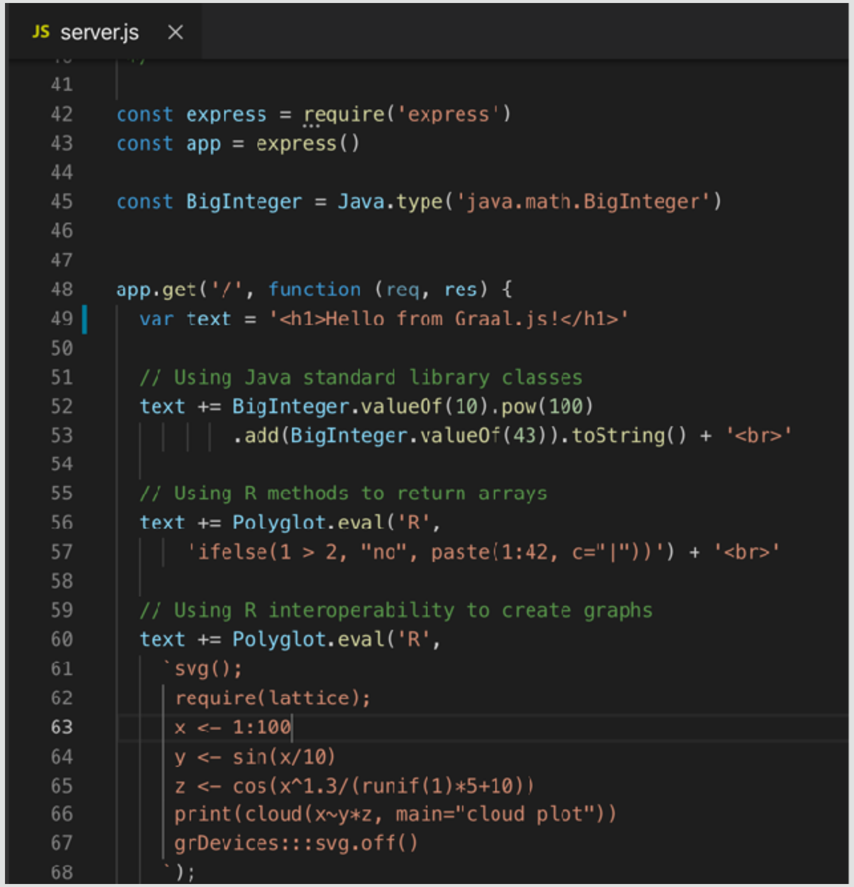
The `js.nashorn-compat` option can be set using a command line option:

```
$ js --experimental-options --js.nashorn-compat=true
```

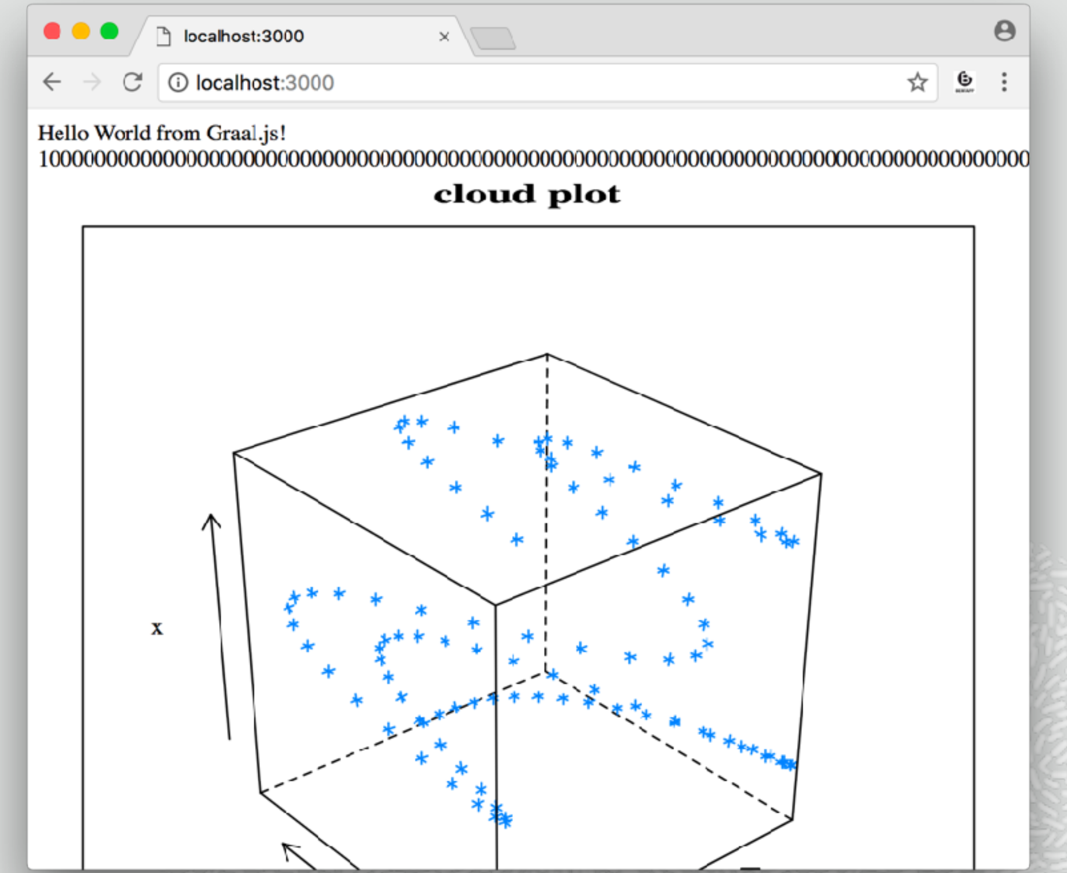
https://github.com/graalvm/graaljs/blob/master/docs/user/NashornMigrationGuide.md

Polyglot programs

JavaScript + Java + R



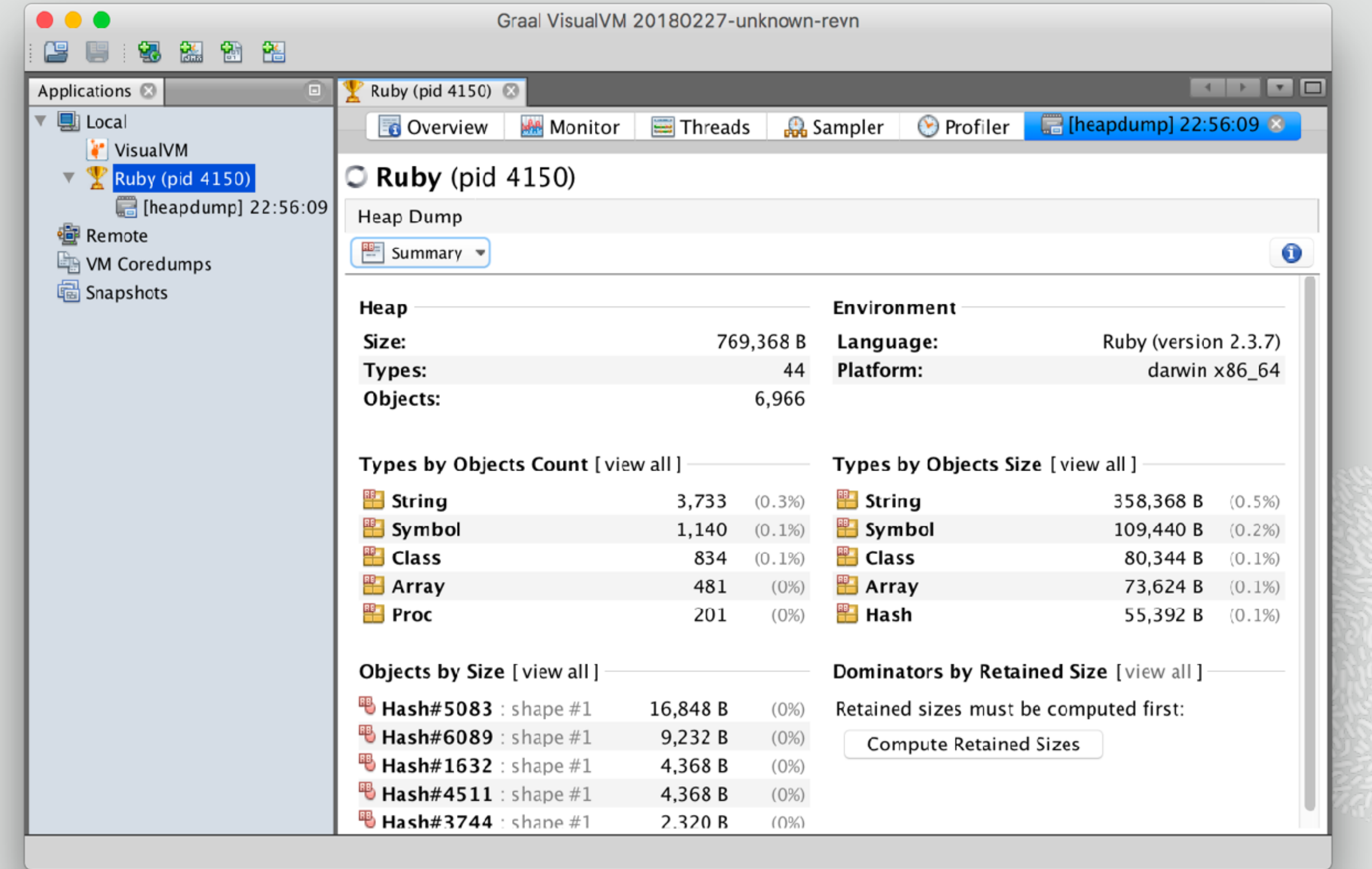
```
const express = require('express')
const app = express()

const BigInteger = Java.type('java.math.BigInteger')


app.get('/', function (req, res) {
  var text = '<h1>Hello from Graal.js!</h1>'

  // Using Java standard library classes
  text += BigInteger.valueOf(10).pow(100)
          .add(BigInteger.valueOf(43)).toString() + '<br>'

  // Using R methods to return arrays
  text += Polyglot.eval('R',
      'ifelse(1 > 2, "no", paste(1:42, c="|"))') + '<br>'

  // Using R interoperability to create graphs
  text += Polyglot.eval('R',
    `svg();
     require(lattice);
     x <- 1:100
     y <- sin(x/10)
     z <- cos(x^1.3/(runif(1)*5+10))
     print(cloud(x~y*z, main="cloud plot"))
     grDevices:::svg.off()
    `);
```



Hello World from Graal.js!

100000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000

Polyglot tools: GraalVM VisualVM



Ruby (pid 4150)

Heap Dump

Heap

Size:	769,368 B
Types:	44
Objects:	6,966

Environment

Language:	Ruby (version 2.3.7)
Platform:	darwin x86_64

Types by Objects Count [view all]

String	3,733	(0.3%)
Symbol	1,140	(0.1%)
Class	834	(0.1%)
Array	481	(0%)
Proc	201	(0%)

Types by Objects Size [view all]

String	358,368 B	(0.5%)
Symbol	109,440 B	(0.2%)
Class	80,344 B	(0.1%)
Array	73,624 B	(0.1%)
Hash	55,392 B	(0.1%)

Objects by Size [view all]

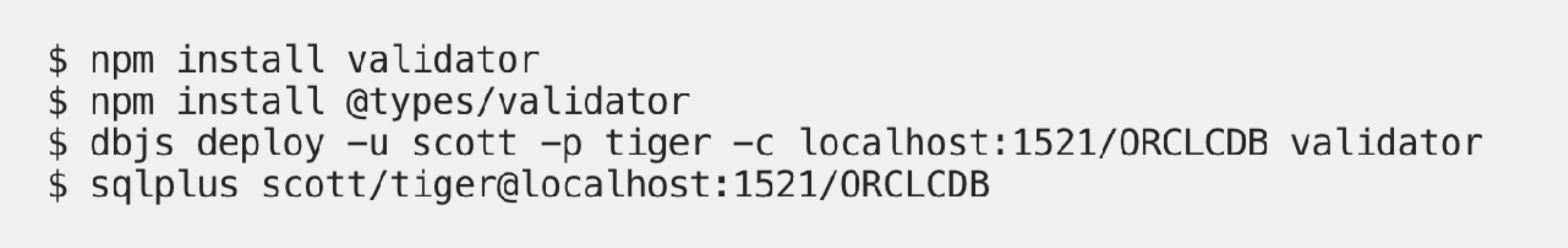
Hash#5083 : shape #1	16,848 B	(0%)
Hash#6089 : shape #1	9,232 B	(0%)
Hash#1632 : shape #1	4,368 B	(0%)
Hash#4511 : shape #1	4,368 B	(0%)
Hash#3744 : shape #1	2,320 B	(0%)

Dominators by Retained Size [view all]

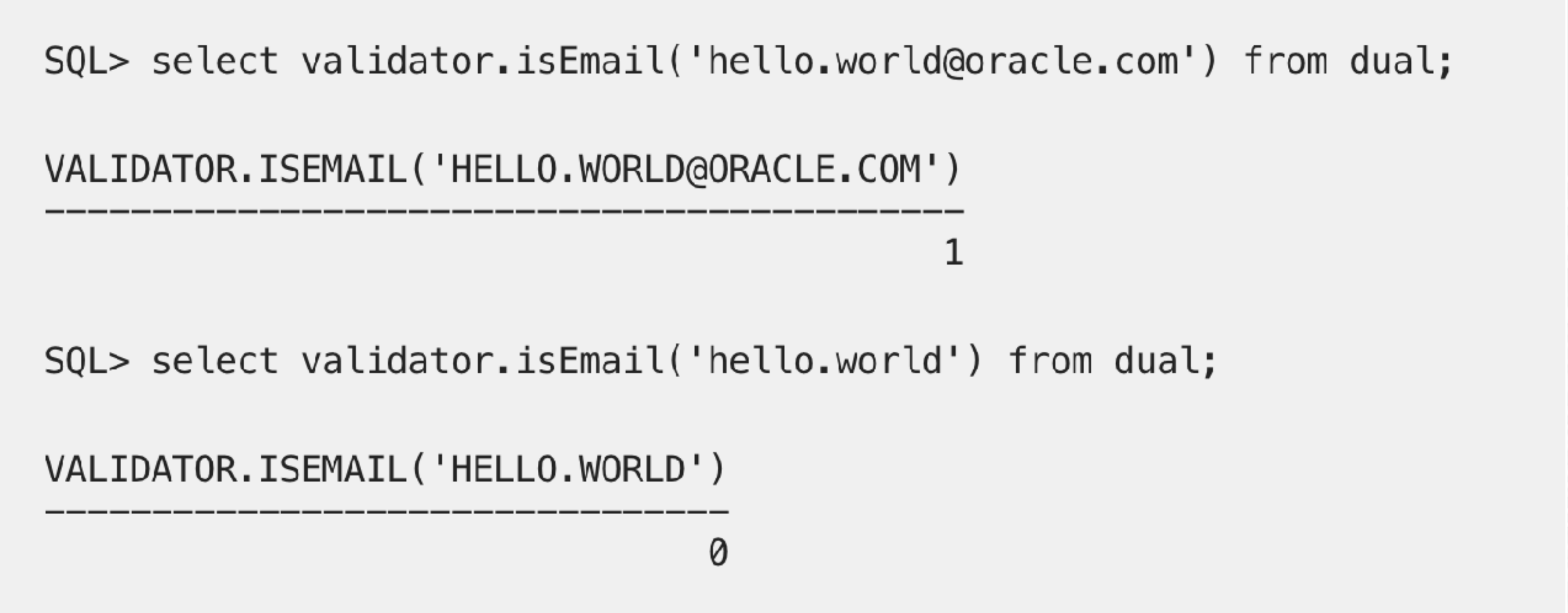
Retained sizes must be computed first:

Compute Retained Sizes

Polyglot in a Database

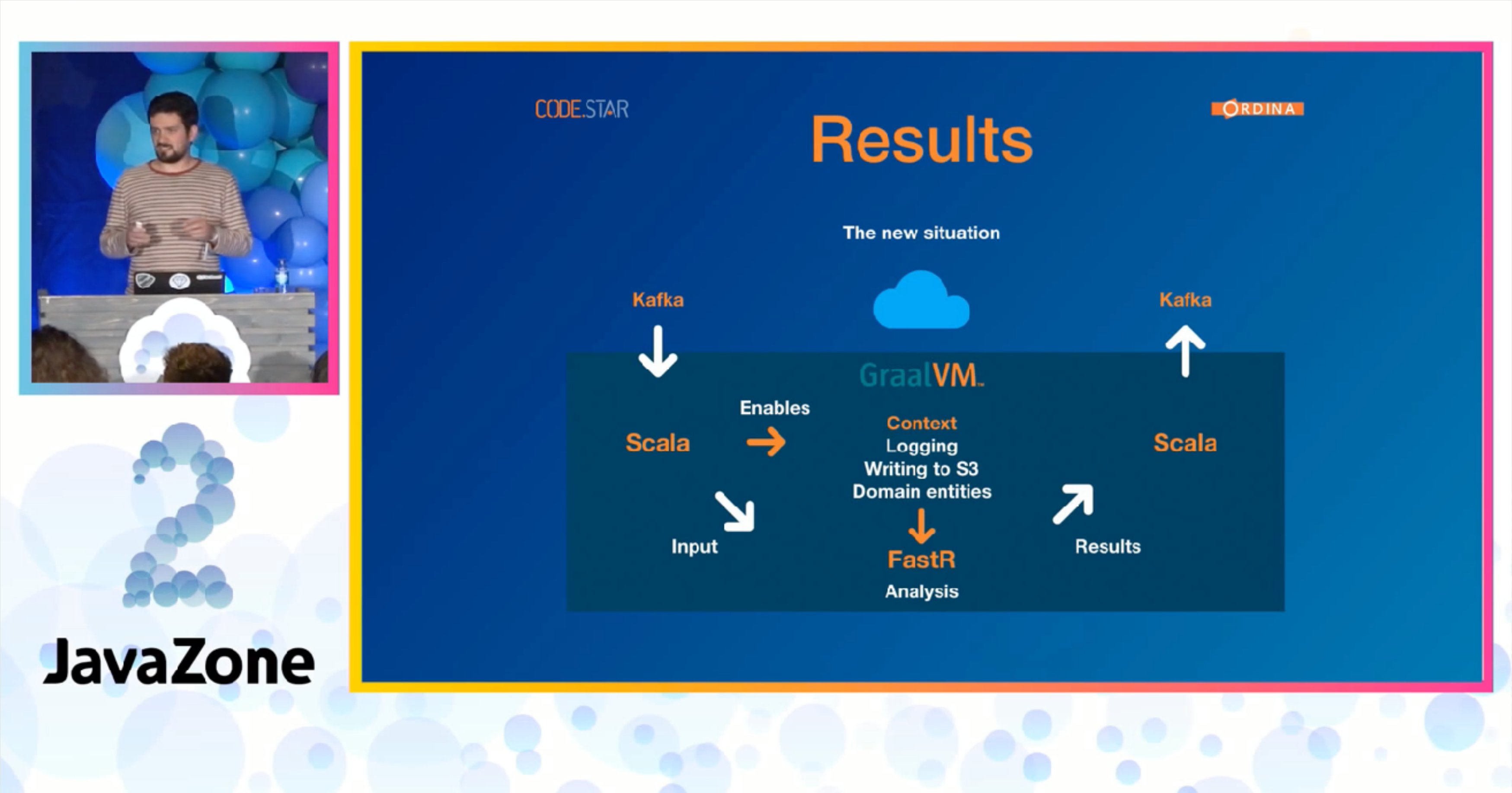


```
$ npm install validator
$ npm install @types/validator
$ dbjs deploy -u scott -p tiger -c localhost:1521/ORCLCDB validator
$ sqlplus scott/tiger@localhost:1521/ORCLCDB
```



```
SQL> select validator.isEmail('hello.world@oracle.com') from dual;

VALIDATOR.ISEMAIL('HELLO.WORLD@ORACLE.COM')
-------------------------------------------
                                          1

SQL> select validator.isEmail('hello.world') from dual;

VALIDATOR.ISEMAIL('HELLO.WORLD')
--------------------------------
                               0
```

GraalVM in practice at the Dutch National Police



Results

The new situation

Kafka

GraalVM

Kafka

Scala

Enables

Context
Logging
Writing to S3
Domain entities

Scala

Input

FastR

Analysis

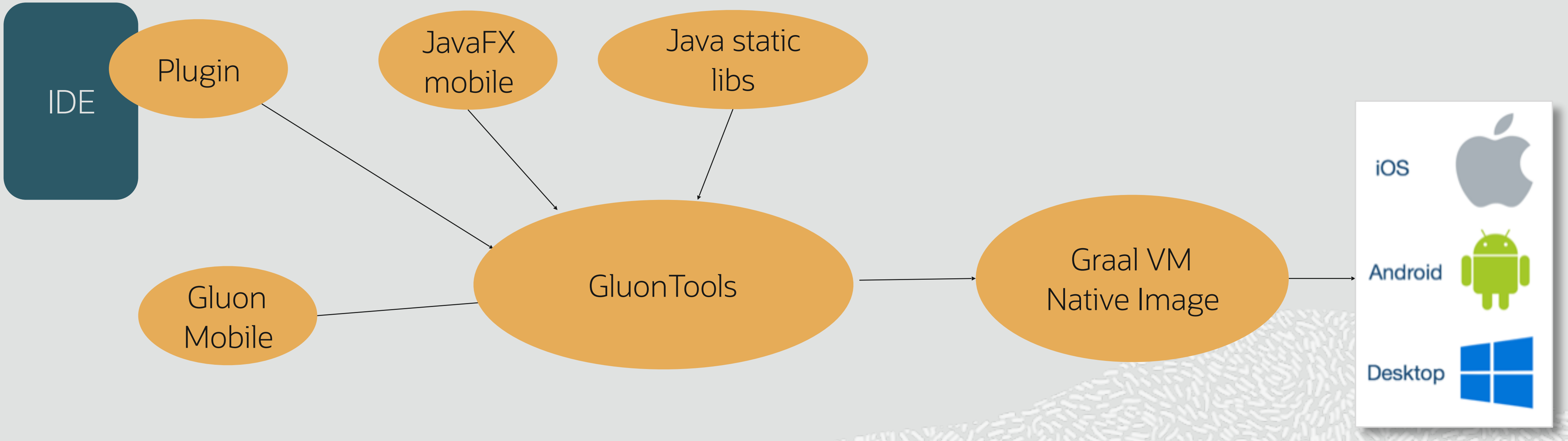
Results

https://vimeo.com/360837119

Do even more with GraalVM: polyglot demo

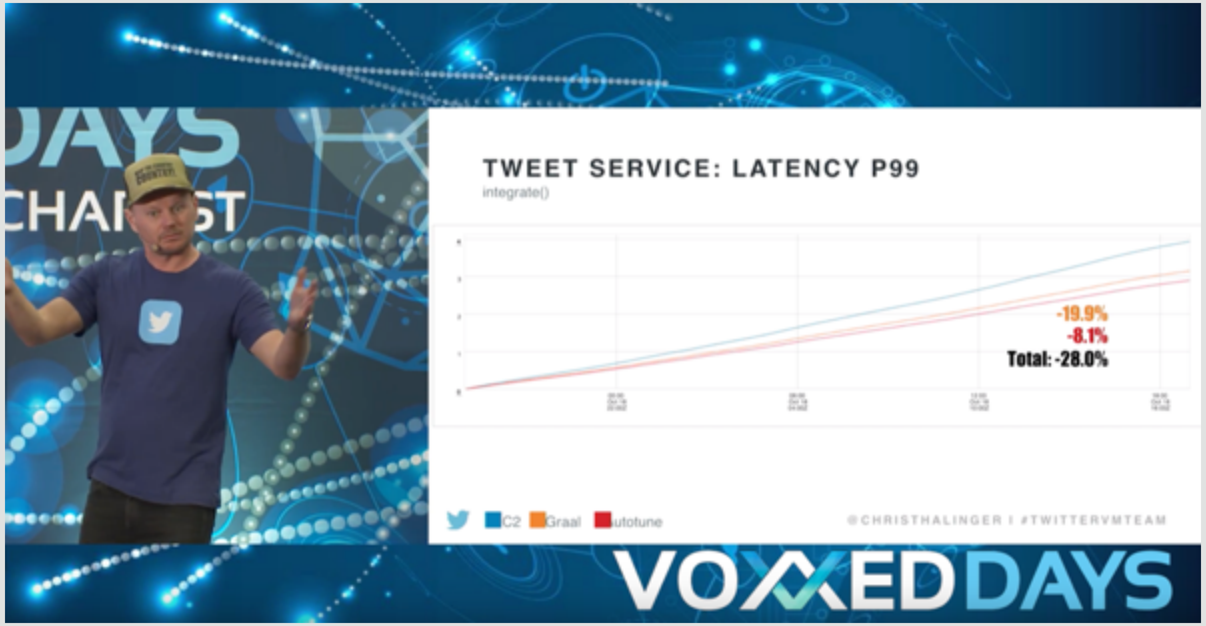
Zero overhead interoperability between programming languages allows you to write polyglot applications and select the best language for your task.

Do even more with GraalVM: Cross-Platform Development



Industry Use Cases

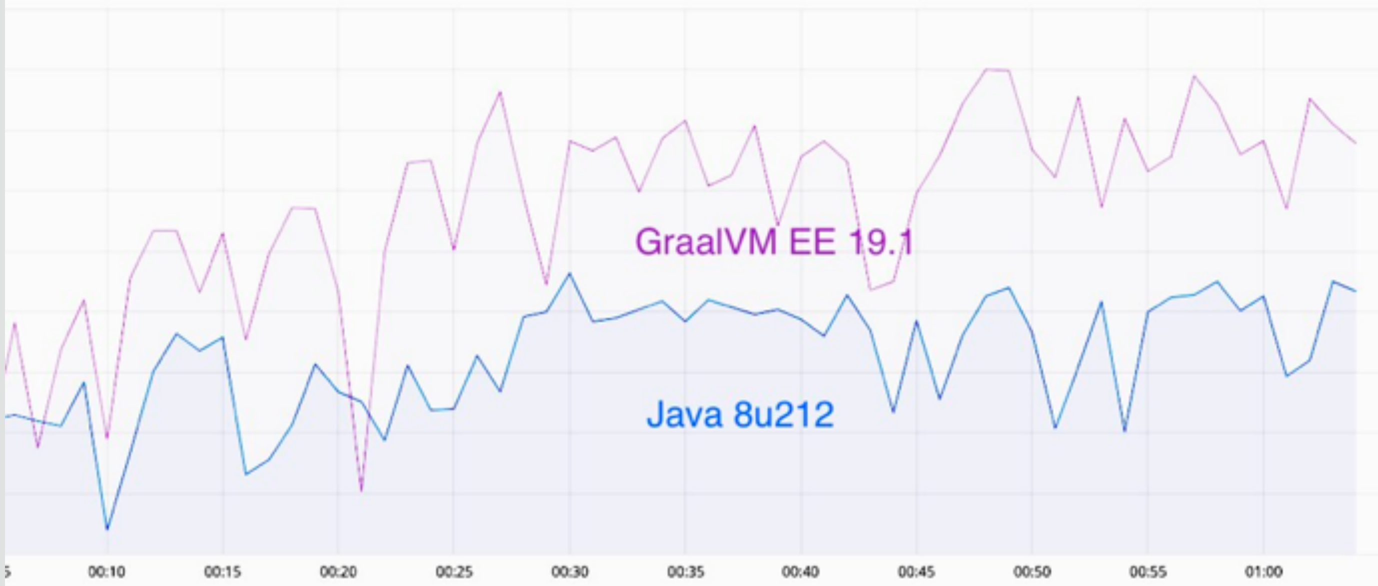
Twitter uses GraalVM compiler in production to run their Scala microservices



- Peak performance: +10%
- Garbage collection time: -25%
- Seamless migration

ORACLE

Cloud Infrastructure



The rich ecosystem of CUDA-X libraries is now available for GraalVM applications.

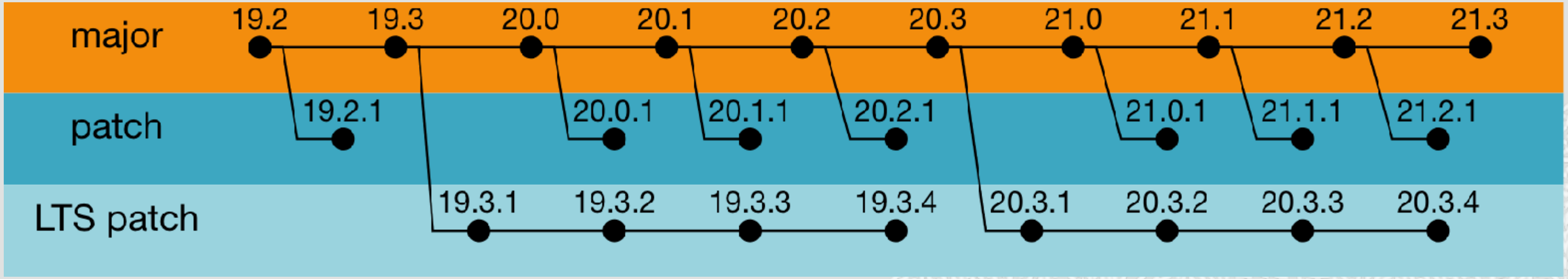
GPU kernels can be directly launched from GraalVM languages such as R, JavaScript, Scala and other JVM-based languages.

NVIDIA®

What’s next

Version Roadmap

- Predictable release schedule;
- LTS releases: last major release of the year.



https://www.graalvm.org/docs/release-notes/version-roadmap

Recent Updates: Class Initialization in Native Images

- Since GraalVM 19.0, application classes in native images are by default initialized at run time and no longer at image build time.

- Configure class initialization behavior:

```
--initialize-at-build-time=...
```

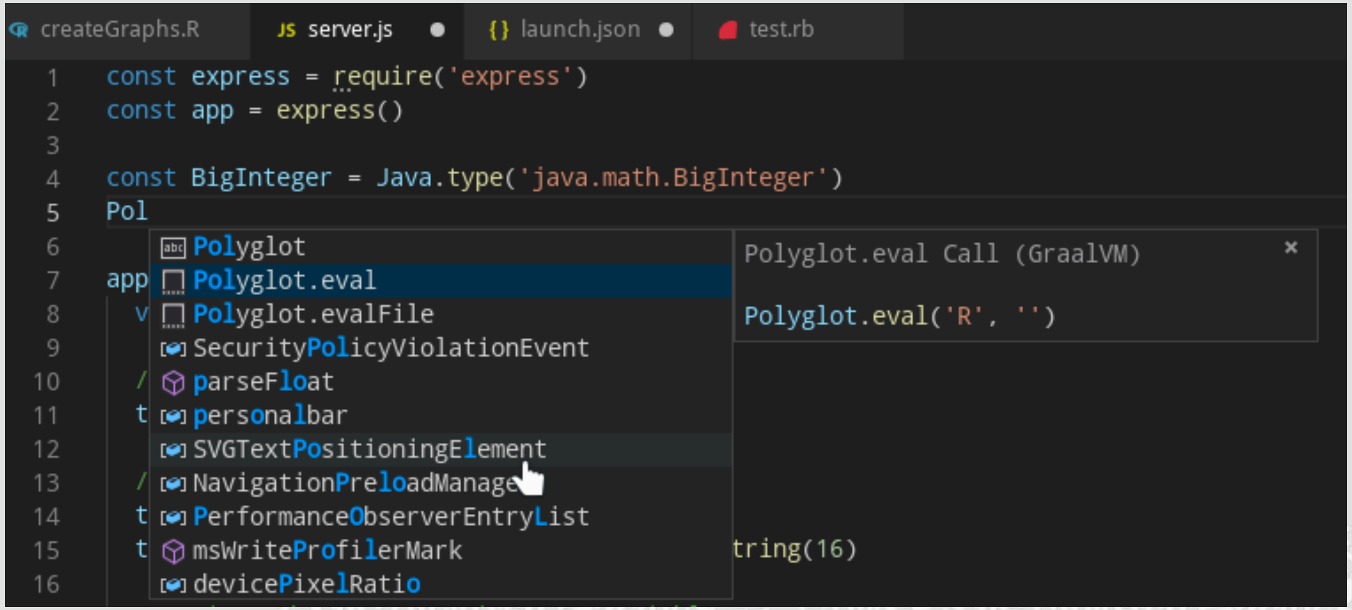
```
--initialize-at-run-time=...
```

- To debug and understand class initialization problems:

```
-H:+TraceClassInitialization
```

Recent Updates: VS Code Plugin

Basic support for editing and debugging programs running on GraalVM



```
const express = require('express')
const app = express()

const BigInteger = Java.type('java.math.BigInteger')
Pol

app
  v

  /
  t

  /
  t
  t                                                  tring(16)
```

```
Polyglot
Polyglot.eval
Polyglot.evalFile
SecurityPolicyViolationEvent
parseFloat
personalbar
SVGTextPositioningElement
NavigationPreloadManager
PerformanceObserverEntryList
msWriteProfilerMark
devicePixelRatio
```

```
Polyglot.eval Call (GraalVM)

Polyglot.eval('R', '')
```

What’s next for GraalVM

- JDK-11 based builds;
- ARM64 and Windows support;
- Low-latency, high-throughput, and parallel GC for native images;
- Work with the community to support important libraries;
- New languages and platforms;
- Your choice – contribute!

Use GraalVM:

- Run your applications faster;
- For fast startup & low memory footprint;
- Write polyglot apps;
- Embed in your platform.

What’s next for you

- Download:

  graalvm.org/downloads

- Follow updates:

  @GraalVM / #GraalVM

- If you need help:
- graalvm.org/community/
- graalvm-users @oss.oracle.com

Thank you!

Alina Yurenko / @alina_yurenko

GraalVM Developer Advocate
Oracle Labs