GraalVM™

GraalVM in a Nutshell

Alina Yurenko
Shaun Smith

Jfokus 2024

GraalVM™

An advanced JDK with ahead-of-time
Native Image compilation

JIT

```
java MyMainClass
```

AOT

```
native-image MyMainClass
./mymainclass
```

GraalVM and OpenJDK

OpenJDK

Galahad

Galahad's primary goal is to contribute Java-related GraalVM technologies to the OpenJDK Community and prepare them for possible incubation in a JDK main-line release.

This Project is sponsored by the HotSpot Group.

Community

- Members
- Mailing list
  - galahad-dev (archives)
- Links
  - Repository

OpenJDK

Project Leyden

The primary goal of this Project is to improve the startup time, time to peak performance, and footprint of Java programs.

This Project is sponsored by the HotSpot and Core Libraries Groups.

Design notes

- Project Leyden: Beginnings
- Selectively Shifting and Constraining Computation
- Toward Condensers
- Condensing Indy Bootstraps

GraalVM Editions

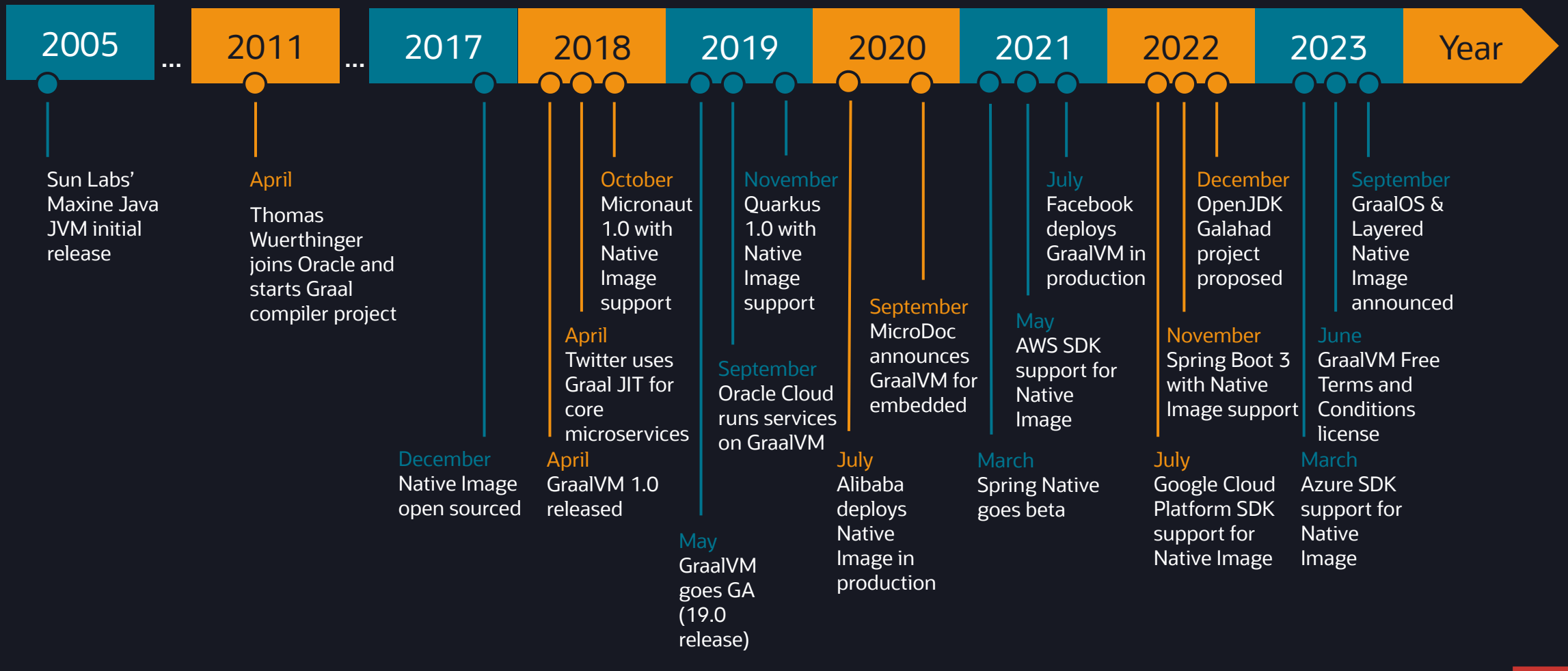
GraalVM Community Edition (CE)

- Built on OpenJDK and follows its release model
- Open-source on GitHub

Oracle GraalVM

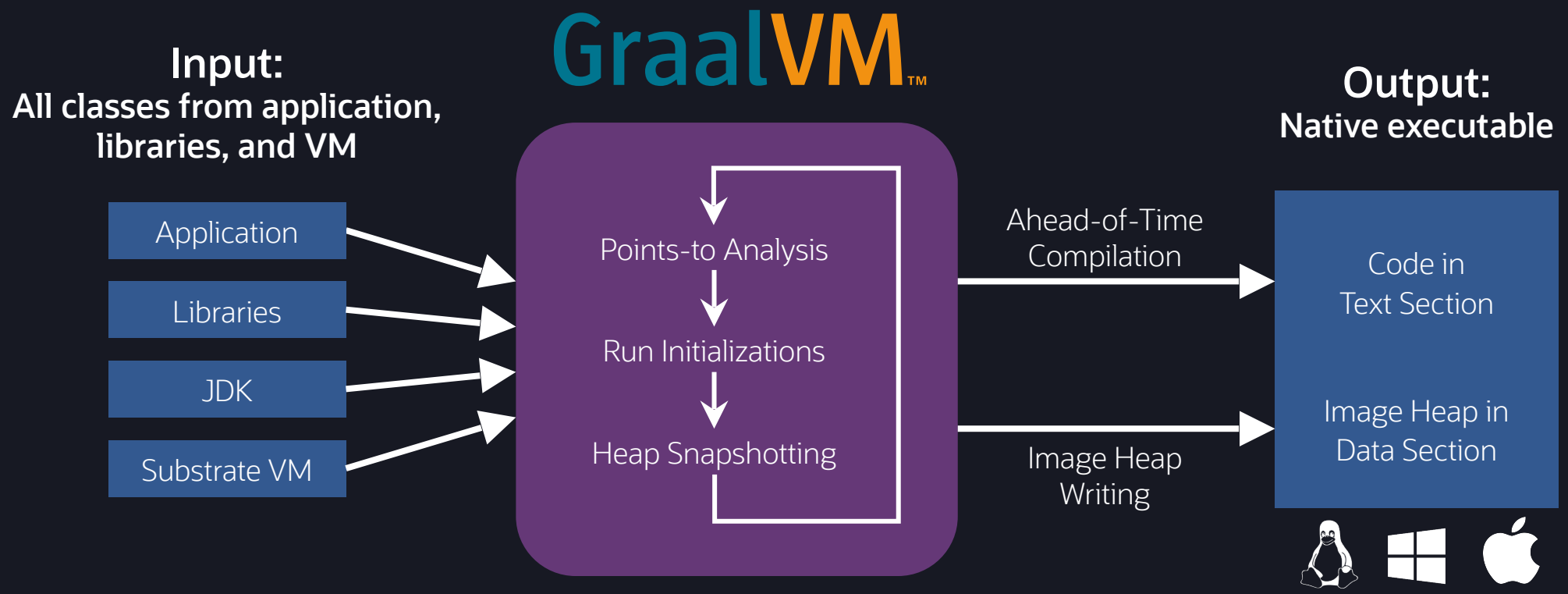
- Built on Oracle JDK and follows its release model
- Additional compiler optimizations
- G1 GC tailored for AOT compiled applications
- Profile guided optimization
- ML profile-based optimization

Graal History



GraalVM Native Image

GraalVM Native Image AOT Compilation



Demo – Build Your First App

Demo – AOT at the speed of JIT 🚀

Spring PetClinic on Oracle GraalVM for JDK 21

	GraalVM CE with C2 JIT	Oracle GraalVM Native Image	
Memory Usage (max RSS)	1,029 MB	641 MB	-38% lower
Peak throughput	11,066 req/s	11,902 req/s	+8% higher
Throughput per memory	12,488 req/(GB*s)	18,569 req/(GB*s)	+49% better
Tail latency (P99)	7.2ms	5.15ms	-28% lower
Startup	7,090ms	210ms	34x faster

GraalVM Native Image—Ideal for Cloud Native Applications

Fast Start & Scale

Lower Resource Usage

Predictable Performance

Improved Security

Compact Packaging

MICRONAUT
spring
QUARKUS
helidon.io

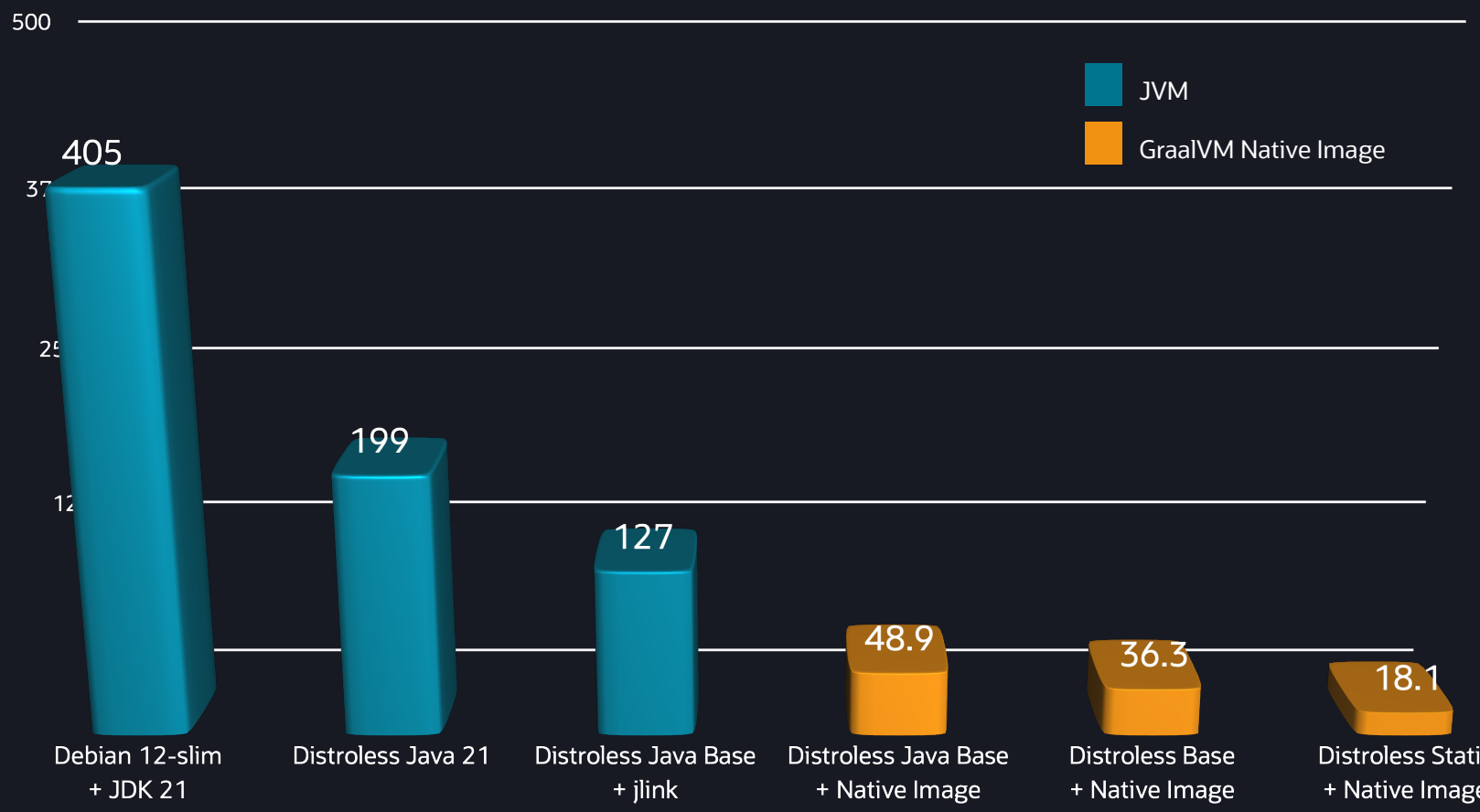
Azure
AWS
GCP
OCI

Supported

JDK's Simple Web Server

95.5% container image size reduction

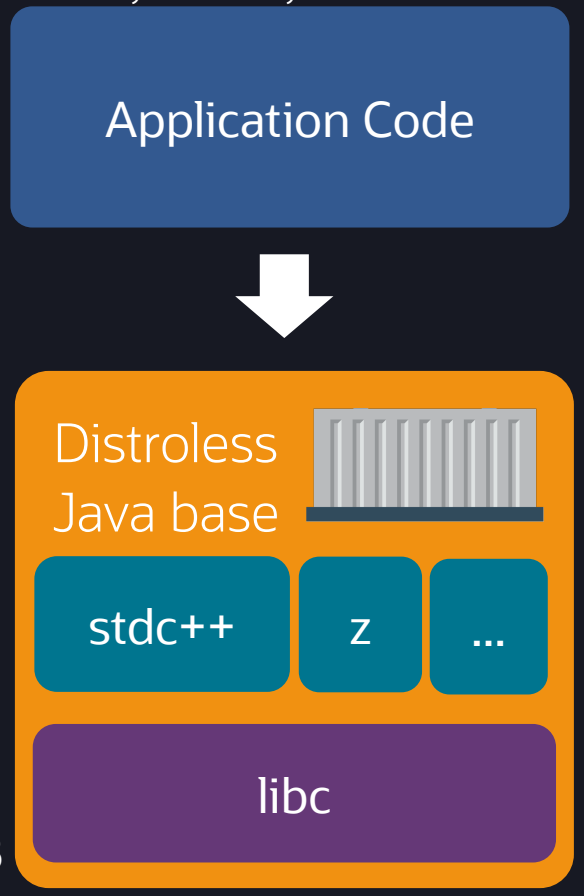
Compact Packaging



Native Linking and Containerization Options

Fully Dynamic

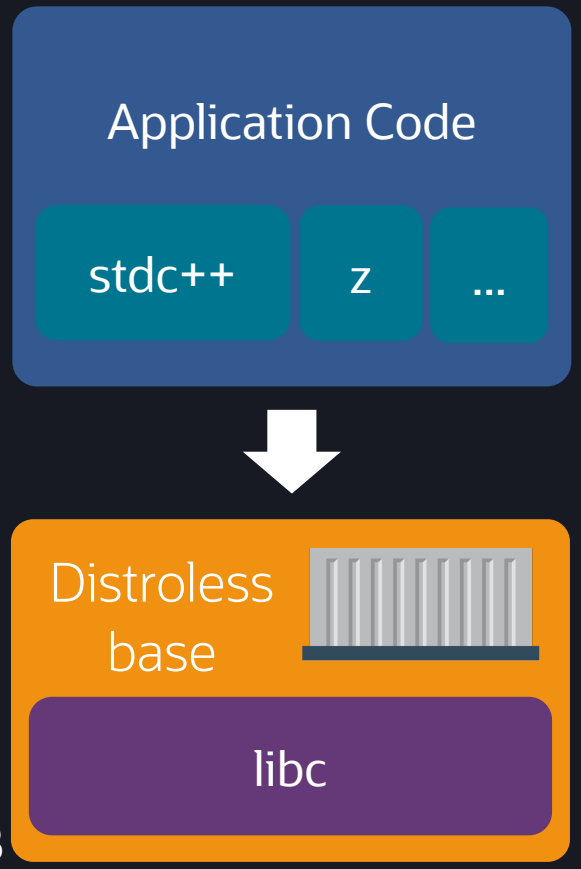
OS must include all dynamically linked libs



33 MB

Mostly Static

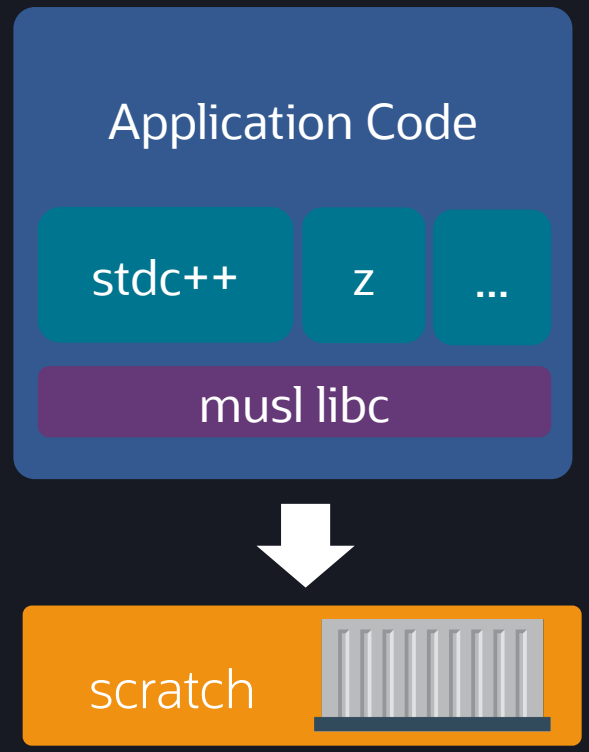
OS only need provide libc.



20 MB

Fully Static

No libs provided by OS



~0 MB

Cross-Platform Builds on GitHub Actions

Reduced Attack Surface

Improved Security

- Reduced attack surface area due to dead code removal—unused classes, methods, and fields not included in executable
- SBOM supporting industry standards
  Embedded in executables
  CycloneDX format
- Not vulnerable to deserialization attacks via class loading—executable includes only required and specified classes
- Not vulnerable to JIT compiler attacks
  all code is AOT compiled

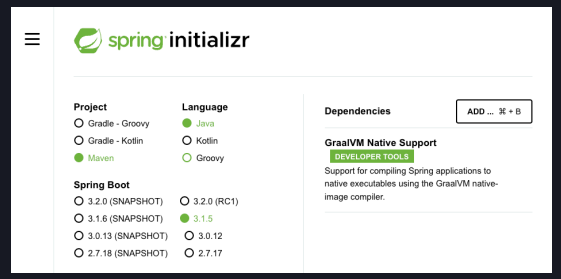
Framework Support Out of the Box 📦

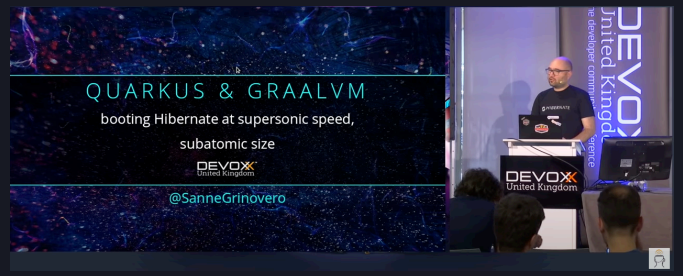
GraalVM Community Meetup

Devoxx Belgium

MICRONAUT
spring
QUARKUS
helidon.io

Supported

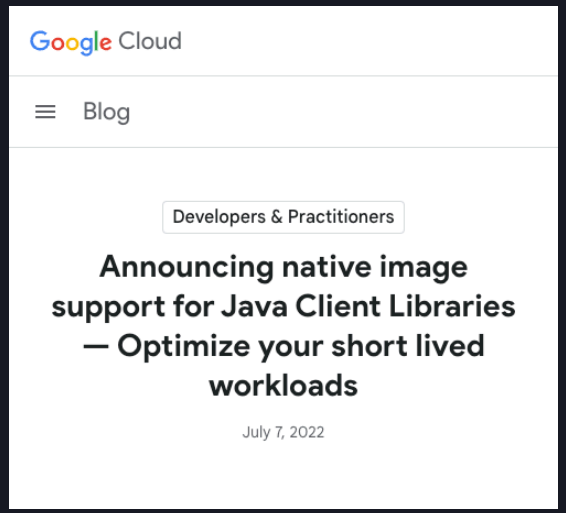


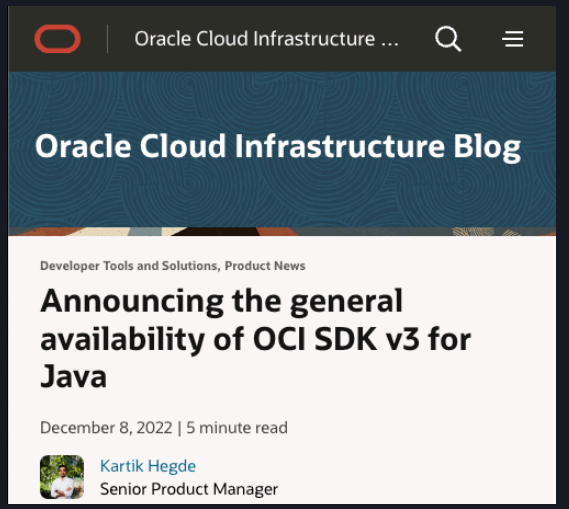


Native Image—Cloud Platform SDK Support

Microsoft Azure
Amazon Web Services
Google Cloud Platform
Oracle Cloud Infrastructure

Supported





GraalVM at Disney Streaming

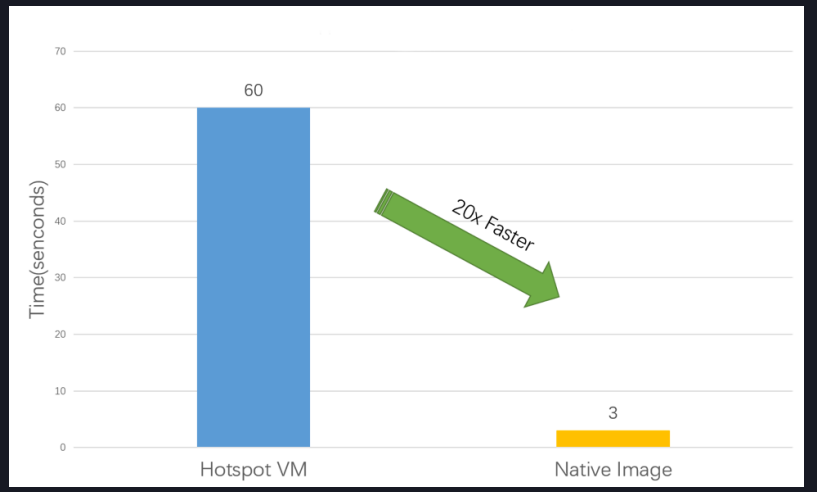
- The Messaging team at Disney Streaming is responsible for engaging our customers through messaging such as email or push notifications. They send millions of messages to the customers for a variety of use cases including password recovery, account changes and purchase confirmations.
- They moved to GraalVM Native Image to solve cold start issues of their serverless Java workloads: the same Java function with 3008 MB of memory that took 3.6 seconds to start with the JVM, started in under 100 milliseconds once compiled to a native executable using GraalVM Native Image.

full story: aws.amazon.com/blogs/opensource/improving-developer-productivity-at-disney-with-serverless-and-open-source/

GraalVM at Alibaba

- Alibaba moved a number of their production microservices and serverless workloads to GraalVM Native Image for performance gains:
  - The startup time decreased from 60 seconds to 3 seconds, i.e., 20X speedup in the starting up time comparing to the JVM. In addition, the GC pause time was controlled to under 100 milliseconds. 100x faster
  - A serverless function was at startup with 1/6 memory cost, which can help customers save 80% of the cost ("billed duration").

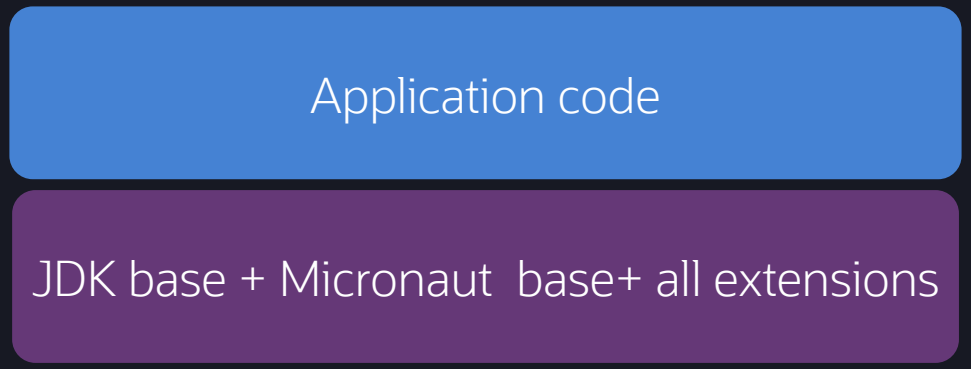
full story: https://www.alibabacloud.com/blog/static-compilation-of-java-applications-at-alibaba-at-scale_596246



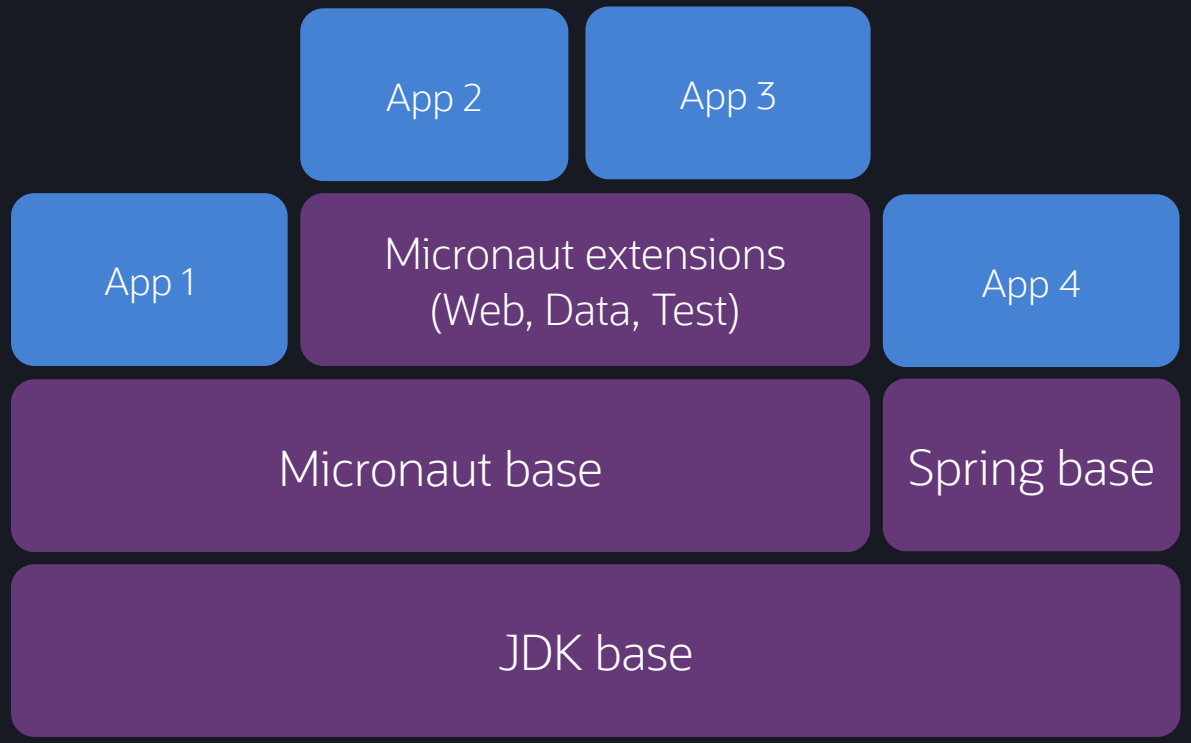
What’s next for GraalVM

Layered Native Images

Development: fast recompilation 🚀



Deployment: resources sharing ☁️



GraalOS—Advanced cloud native application deployment platform

Runs applications as small, fast GraalVM Native Image compiled machine executables

Fast Start

GraalOS applications start fast with virtually no cold start cost

Low Latency

Excellent 99th percentile latency makes GraalOS applications highly responsive

Reduced Memory

GraalOS applications require significantly less memory resulting in reduced operating costs

Run On Demand

GraalOS applications are automatically suspended and resumed on demand —with no idle cost

Applications, not Containers

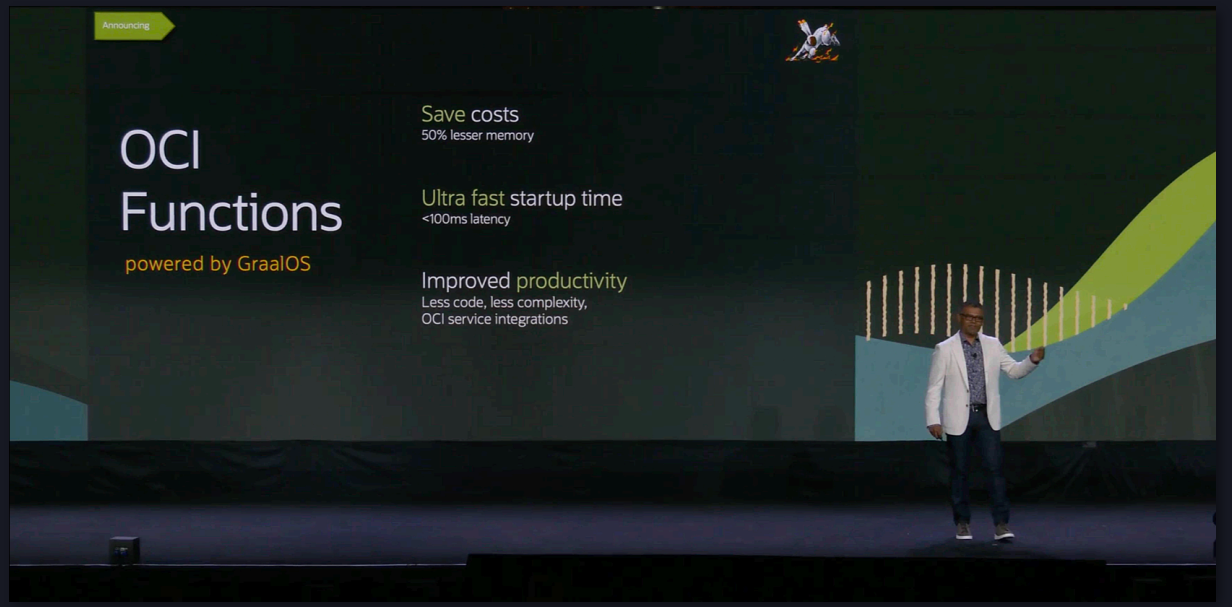
GraalOS uses the latest advances in x86 and AArch64 processor architectures for hardware enforced application isolation without containers

Cloud Native

With support for stateful and stateless services and functions, GraalOS is ideal for cloud native applications

OCI Functions powered by GraalOS

Announced at Oracle Cloud World 2023



	Function (Fn)		Functions powered by GraalOS
Total Time	3.88 seconds		1.93 seconds
Application Time	2.05 seconds		1.91 seconds
Startup Time	1.8 seconds	100x faster	0.018 seconds
Memory	512MB	2x smaller	256MB

GraalVM Native Image—Ideal for Cloud Native Applications

Fast Start & Scale

Lower Resource Usage

Predictable Performance

Improved Security

Compact Packaging

MICRONAUT
spring
QUARKUS
helidon.io

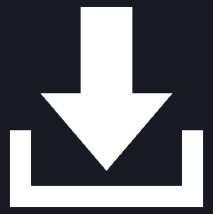
Azure
AWS
GCP
OCI

Supported

Get started with GraalVM 🚀

graalvm.org

docker pull container-registry.oracle.com/graalvm/jdk:21



sdk install java 21.0.2-graal

github.com/graalvm/graalvm-demos

Thank you!

@alina_yurenko
@shaunmsmith


ORACLE