Bring the Action: Using GraalVM in Production

Alina Yurenko
Oracle

About me

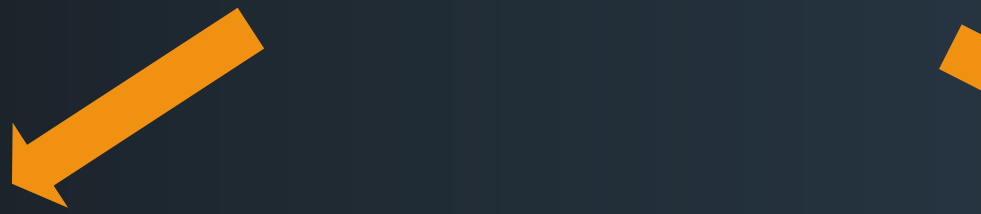
- Alina Yurenko / @alina_yurenko
- Developer Advocate for GraalVM at Oracle
- Love open source and communities 🤝
- Love both programming 👩‍💻 & natural languages 🗣️

GRAALVM 

GraalVM™

An advanced JDK with AOT

Native Image compilation



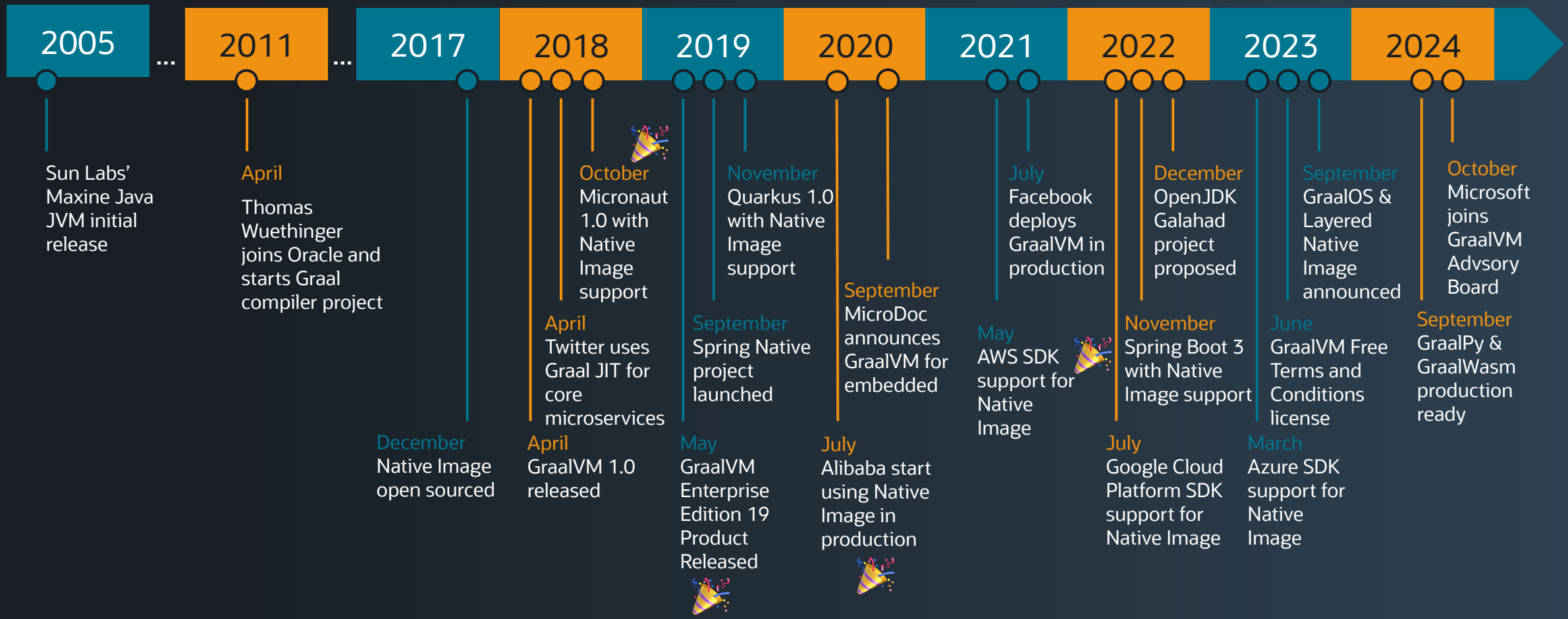
JIT

```
java MyMainClass
```

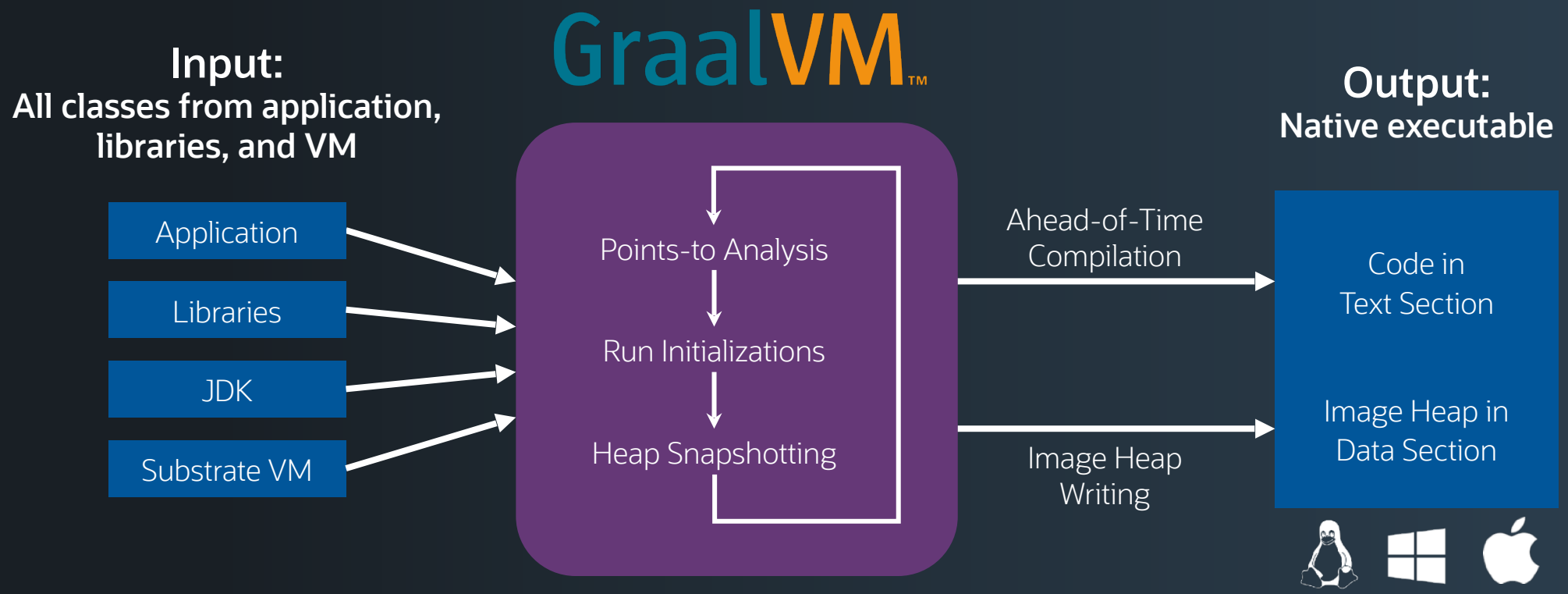
AOT

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

Graal History



GraalVM Native Image AOT Compilation



DEMO 

GraalVM Native Image—Fast Start Up and Much More

Fast Start & Scale

Lower Resource Usage

Predictable Performance

Improved Security

Compact Packaging

MICRONAUT
spring
QUARKUS
helidon.io

Azure
AWS
GCP
OCI

Wide Ecosystem

@alina_yurenko

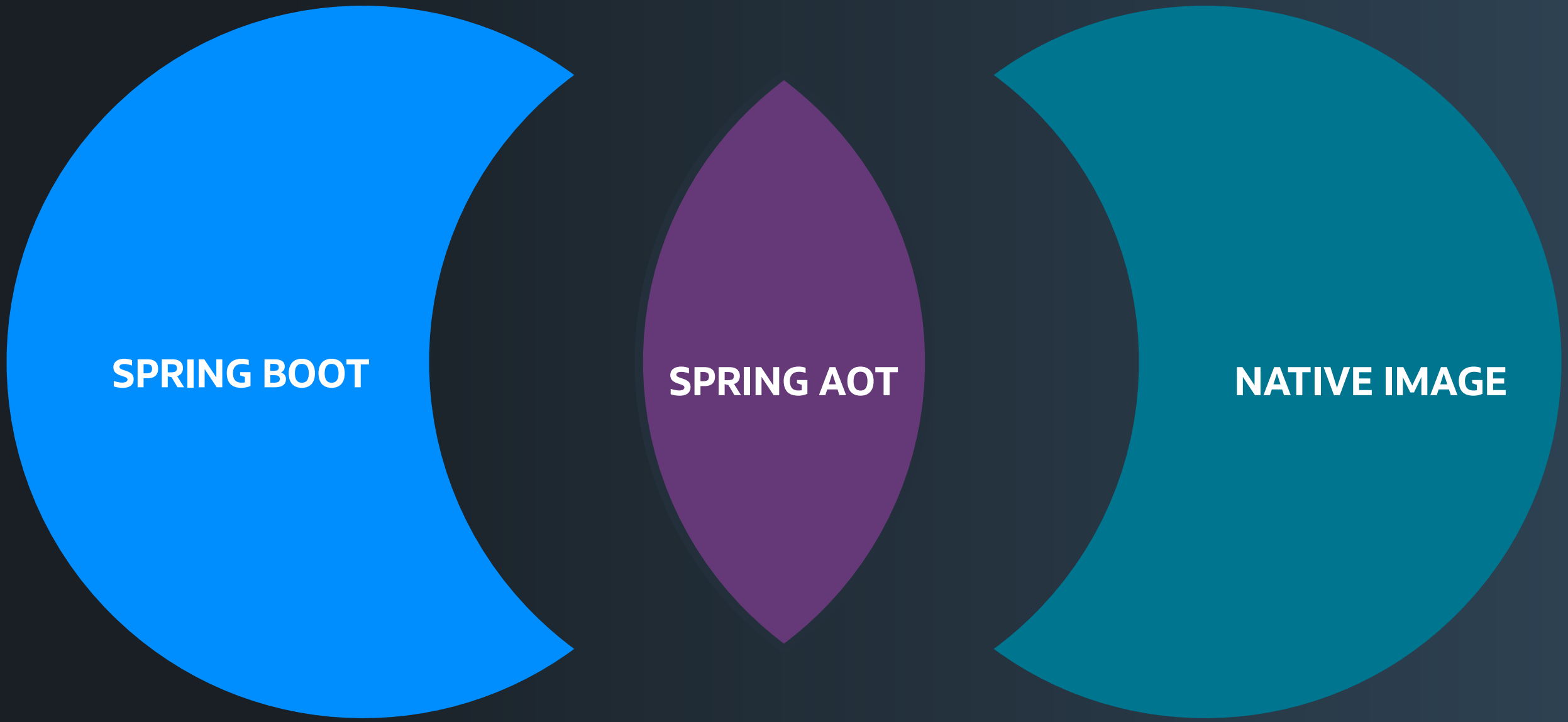
FRAMEWORKS AND LIBRARIES

SPRING BOOT

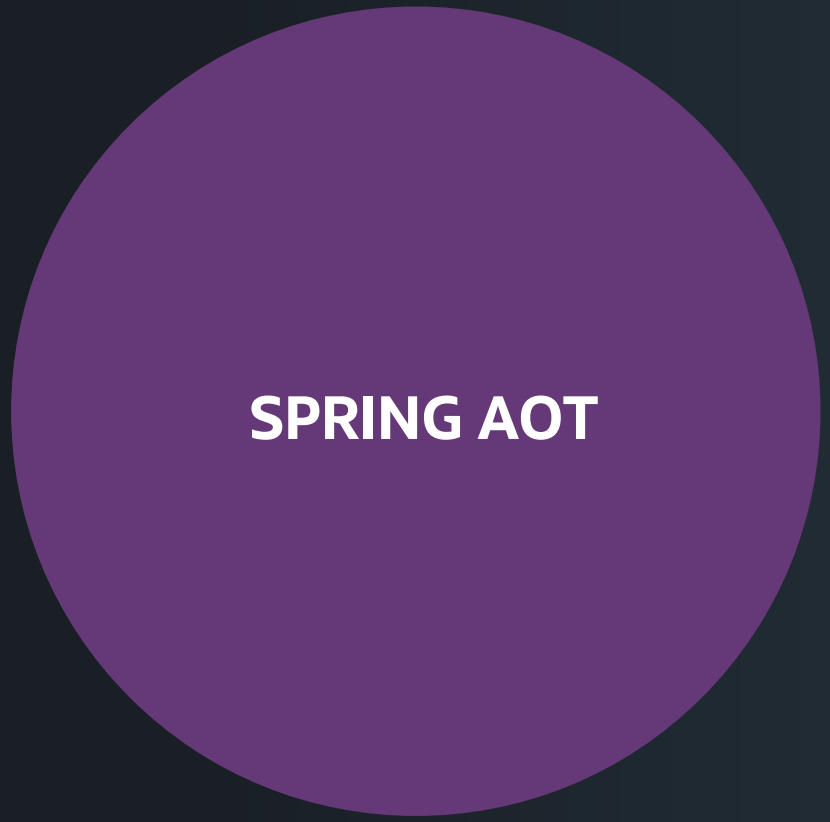
RUN-TIME DISCOVERY
AND CONFIGURATION

NATIVE IMAGE

BUILD-TIME DISCOVERY
AND CONFIGURATION


SPRING BOOT
SPRING AOT
NATIVE IMAGE

SPRING AOT



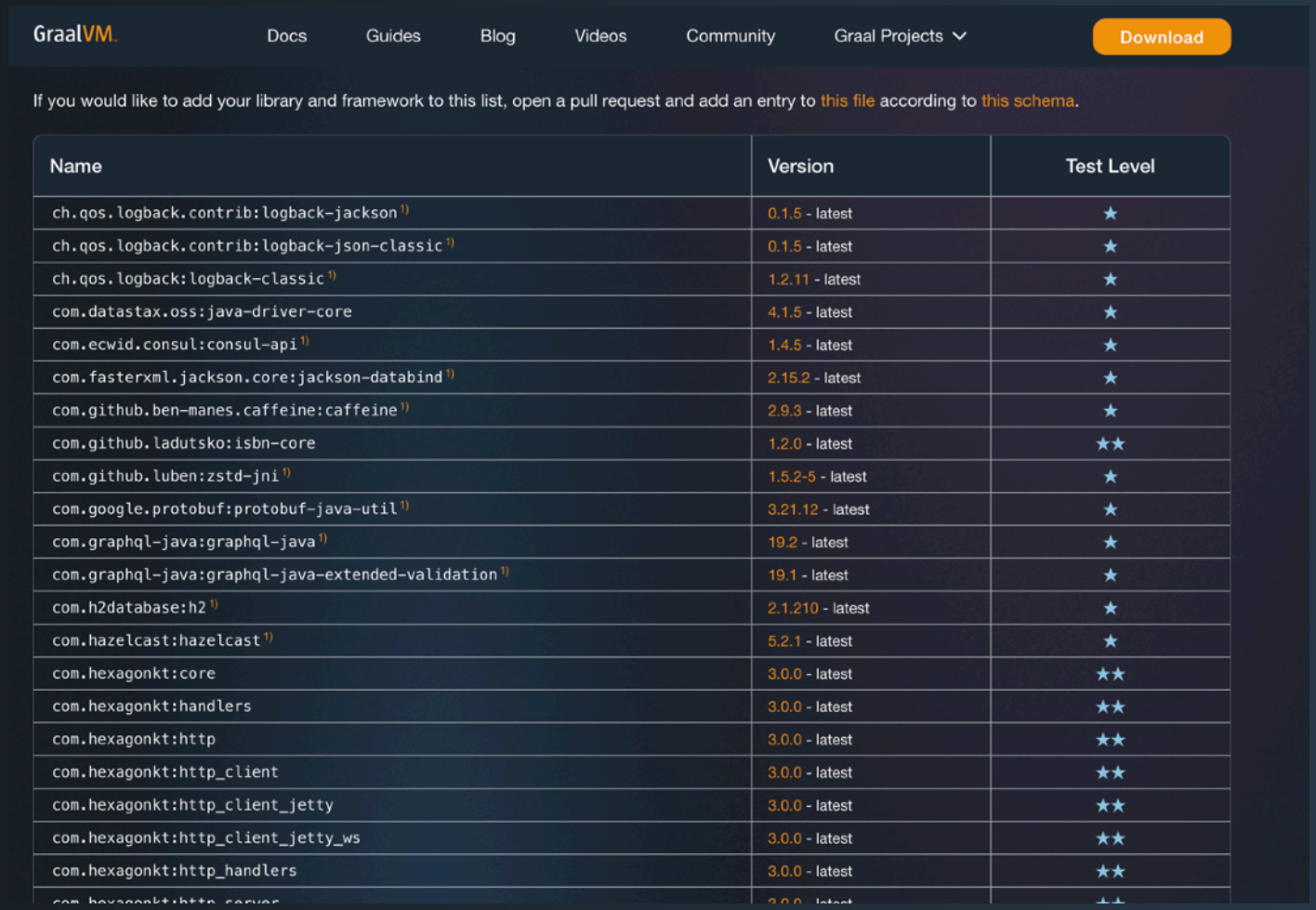
JAVA SOURCE FILES

BYTECODE

HINT FILES

Ready for GraalVM Native Image

200+ libraries and frameworks integrating and testing with Native Image



GraalVM. Docs Guides Blog Videos Community Graal Projects Download

If you would like to add your library and framework to this list, open a pull request and add an entry to this file according to this schema.

Name	Version	Test Level
ch.qos.logback.contrib:logback-jackson [1]	0.1.5 - latest	★
ch.qos.logback.contrib:logback-json-classic [1]	0.1.5 - latest	★
ch.qos.logback:logback-classic [1]	1.2.11 - latest	★
com.datastax.oss:java-driver-core	4.1.5 - latest	★
com.ecwid.consul:consul-api [1]	1.4.5 - latest	★
com.fasterxml.jackson.core:jackson-databind [1]	2.15.2 - latest	★
com.github.ben-manes.caffeine:caffeine [1]	2.9.3 - latest	★
com.github.ladutsko:isbn-core	1.2.0 - latest	★★
com.github.luben:zstd-jni [1]	1.5.2-5 - latest	★
com.google.protobuf:protobuf-java-util [1]	3.21.12 - latest	★
com.graphql-java:graphql-java [1]	19.2 - latest	★
com.graphql-java:graphql-java-extended-validation [1]	19.1 - latest	★
com.h2database:h2 [1]	2.1.210 - latest	★
com.hazelcast:hazelcast [1]	5.2.1 - latest	★
com.hexagonkt:core	3.0.0 - latest	★★
com.hexagonkt:handlers	3.0.0 - latest	★★
com.hexagonkt:http	3.0.0 - latest	★★
com.hexagonkt:http_client	3.0.0 - latest	★★
com.hexagonkt:http_client_jetty	3.0.0 - latest	★★
com.hexagonkt:http_client_jetty_ws	3.0.0 - latest	★★
com.hexagonkt:http_handlers	3.0.0 - latest	★★

graalvm.org/native-image/libraries-and-frameworks

Libraries, dynamic Java features, and Native Image

- Libraries might be Native-Image friendly out of the box
  - twitter.com/YunaMorgenstern/status/1729039787351536084
- Libraries might include config for Native Image:
  - github.com/h2database/h2database/blob/master/h2/src/main/META-INF/native-image/
- Libraries might contain config in the Reachability Metadata Repository
  - github.com/oracle/graalvm-reachability-metadata/tree/master/metadata/io.netty

Happy path; most of the cases

- You can use framework support to produce custom “hints” for Native Image

```
runtimeHints.resources().registerPattern(“config/app.properties”);
```

- You can use the Tracing Agent to produce the necessary config automatically
  - graalvm.org/latest/reference-manual/native-image/metadata/AutomaticMetadataCollection/
- You can provide/extend config for reflection, JNI, resources, serialization, and predefined classes manually in JSON:
  - graalvm.org/latest/reference-manual/native-image/metadata/#specifying-metadata-with-json

Custom code/ libraries

PERFORMANCE

DEMO 

AOT at the speed of JIT 🚀

- Profile-guided optimizations

Collect and use profiles to optimize for the specific runtime behaviour of your application

- ML-enabled PGO

Use a ML model to automatically predict the profile of the application

- G1 GC

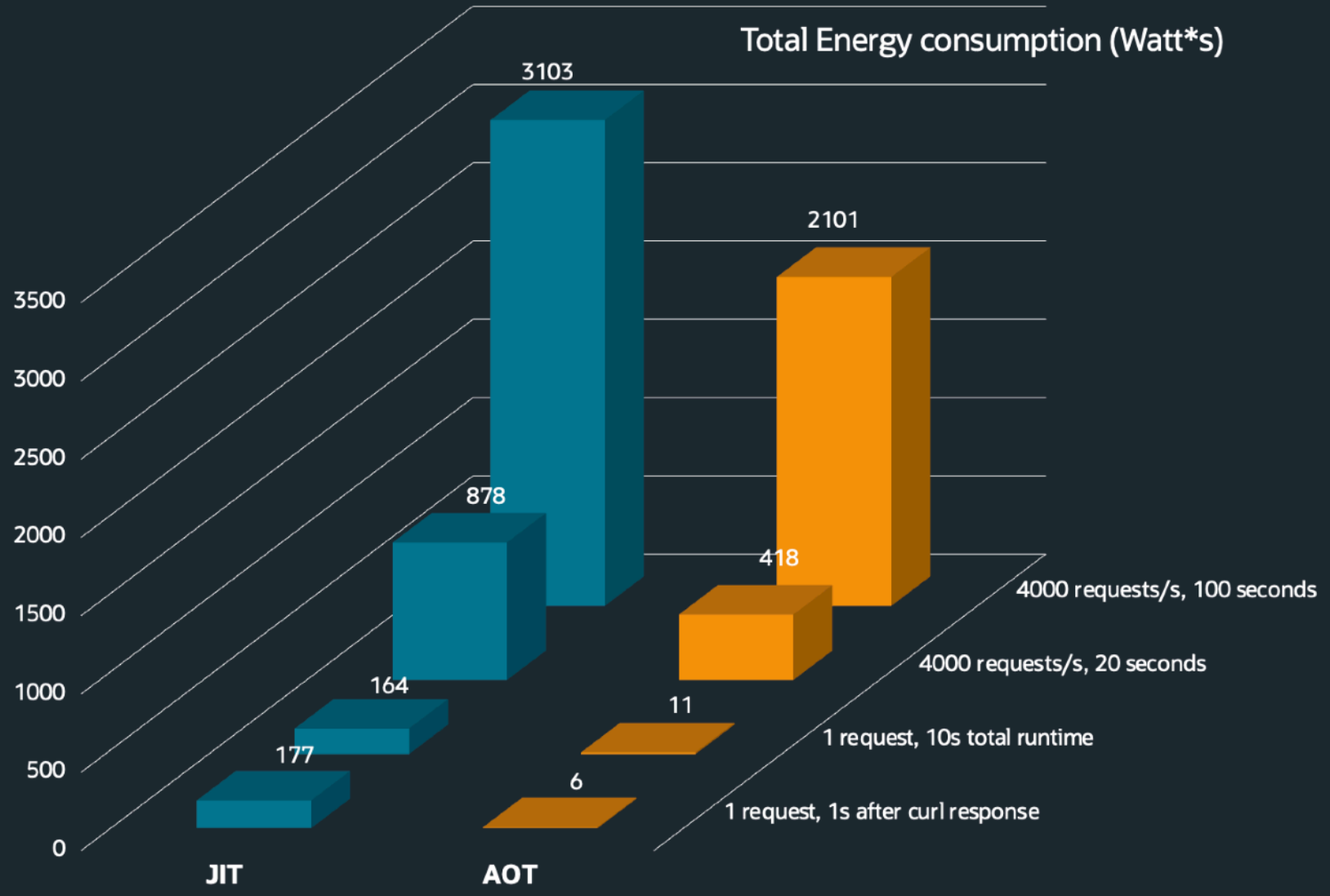
Optimize GC for peak throughput and improved latency

- `-march=native`

Optimize for the specific hardware features of the machine you'll be running on

Spring PetClinic— Energy Consumption

Energy efficient

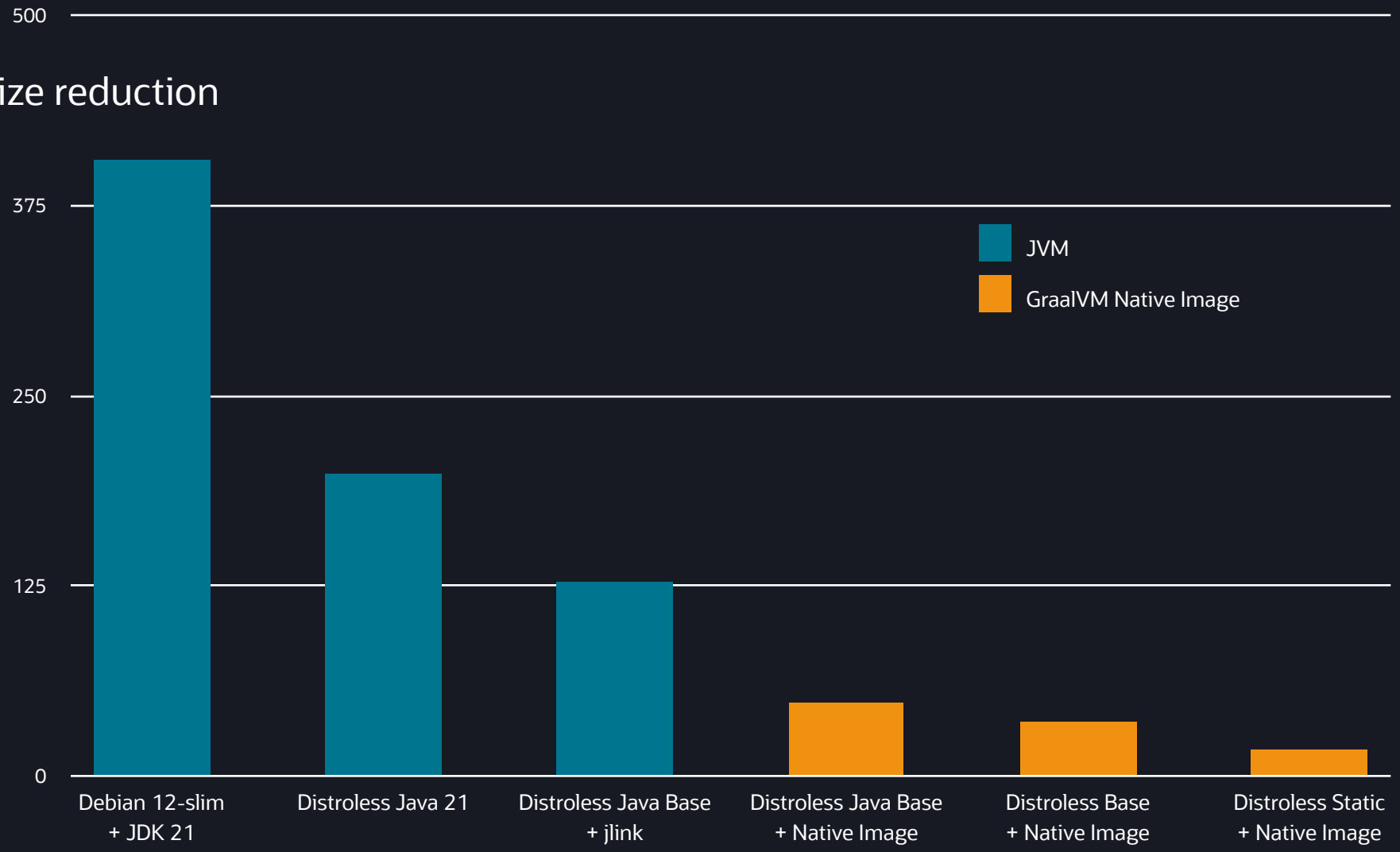


BEST PRACTICES

JDK's Simple Web Server

95.5% container image size reduction

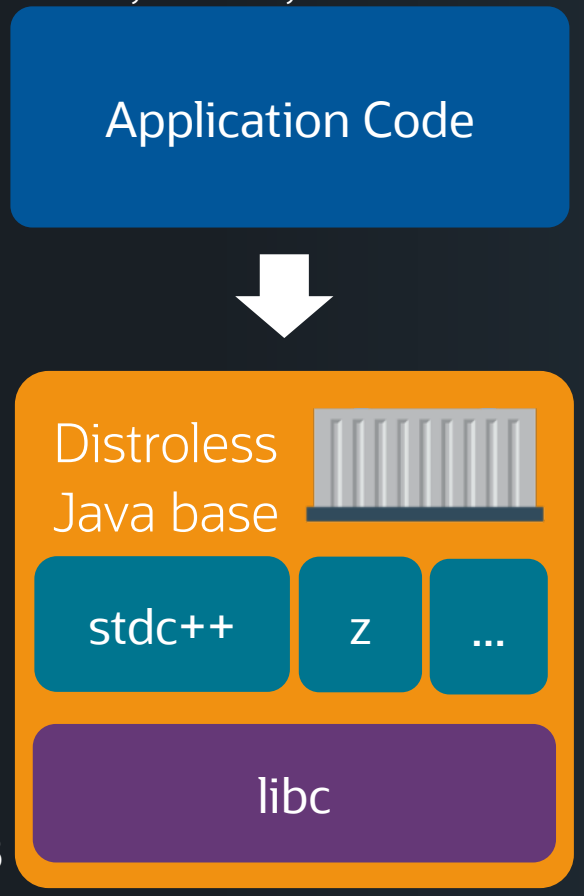
Compact Packaging



Native Linking and Containerization Options

Fully Dynamic

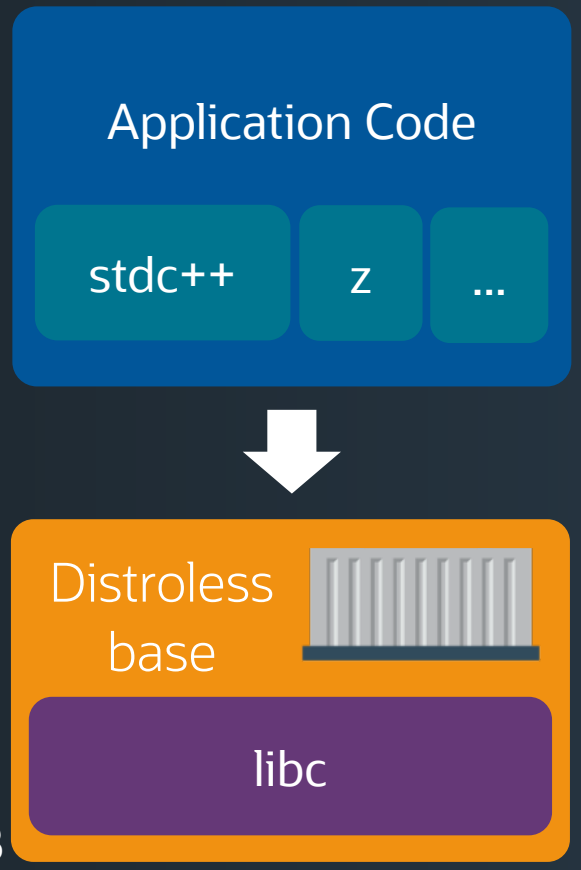
OS must include all dynamically linked libs



33 MB

Mostly Static

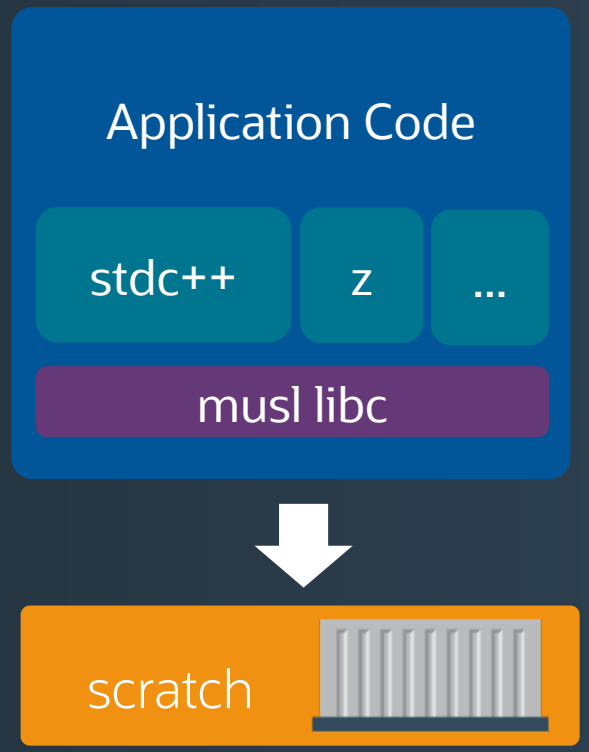
OS only need provide libc.



20 MB

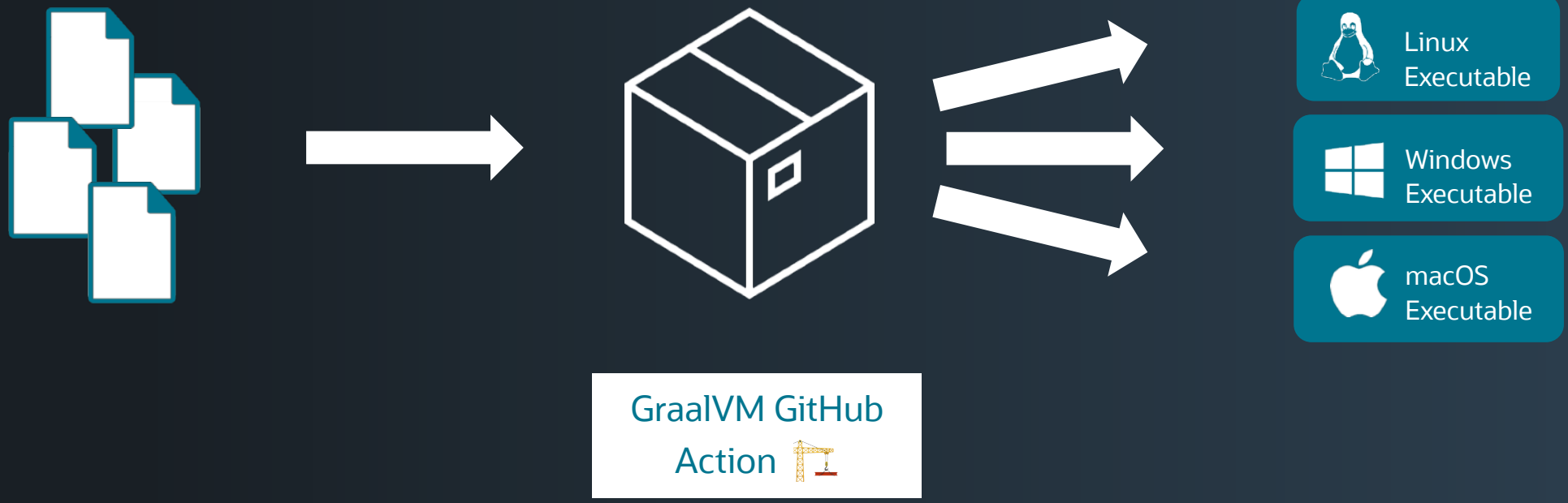
Fully Static

No libs provided by OS



~0 MB

Cross-Platform Builds on GitHub Actions

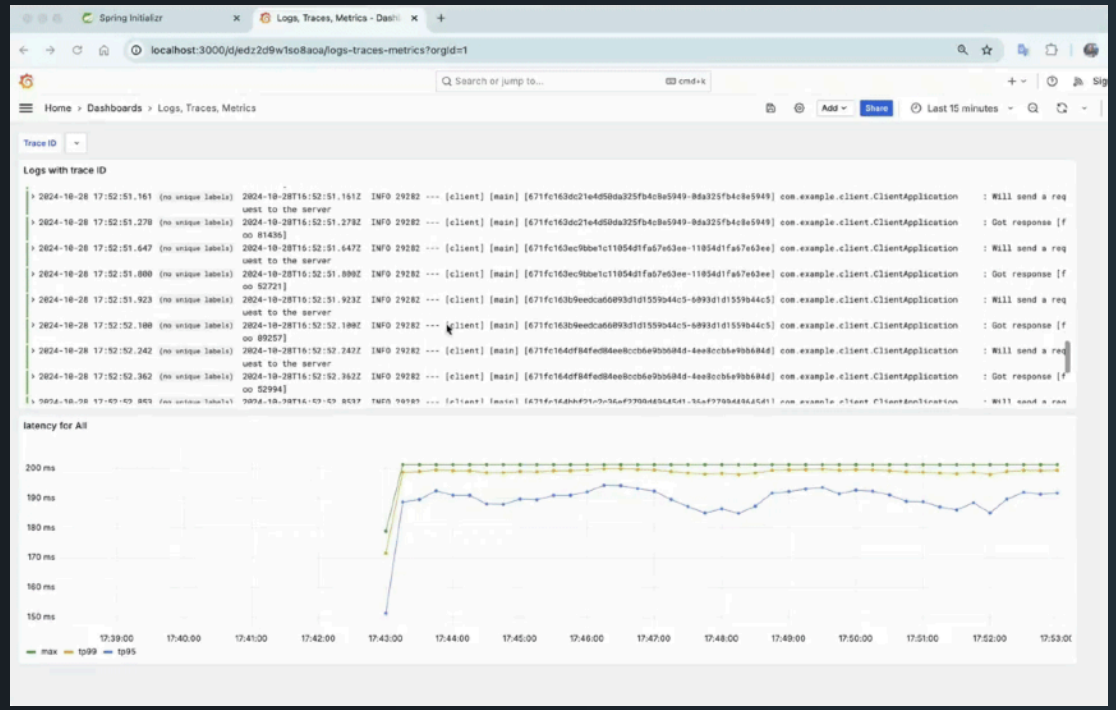


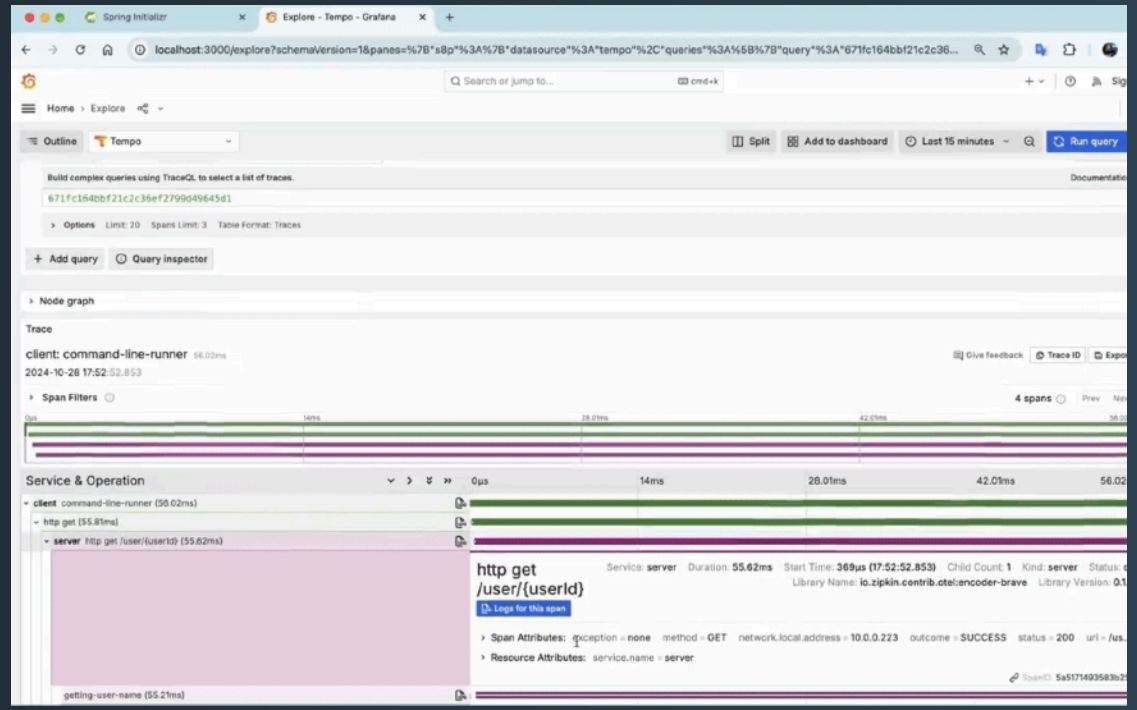
Recommendations

- Migrate 🚀
  - Add Native Build Tools
  - Alternatively, use recent versions of frameworks
  - Evaluate libraries: graalvm.org/native-image/libraries-and-frameworks
- Build and deploy 👷
  - Build and test on GraalVM as the JVM, build with Native Image closer to the deployment
  - Quick build mode with `-Ob`
  - Use CI/CD systems (e.g. GitHub actions) for deployment and cross-platform builds
- Monitor
  - jvmstat, JFR, JMX, Micrometer, cloud vendor solutions
- Run faster 🚀
  - PGO
  - ML-enabled PGO
  - G1 GC
  - `-march=native`

MONITORING 

Monitoring native images with Micrometer





https://spring.io/blog/2022/10/12/observability-with-spring-boot-3

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

WHAT'S NEXT?

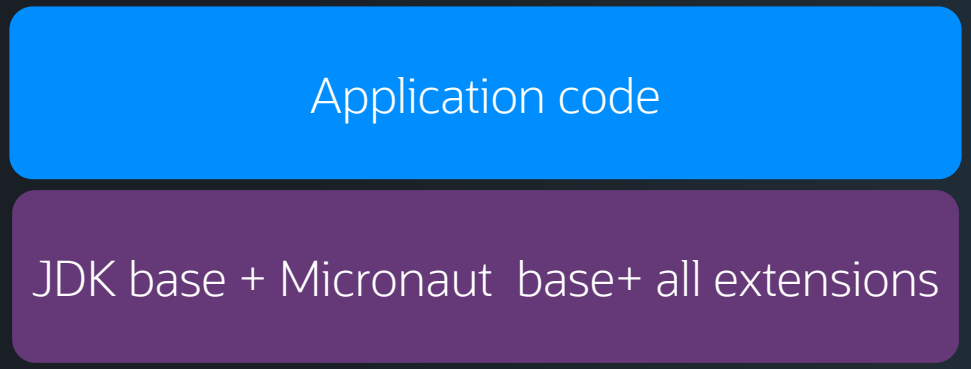
GraalVM for JDK 24 🚀

- Java 24 features 😍
- The fastest GraalVM yet :)
- New ML profile inference for even higher performance out of the box 🤖
- Smaller executables 📦
- Vector API support in Native Image 🚀
- New SBOM features 🛡️
- Developer experience improvements 👩‍💻

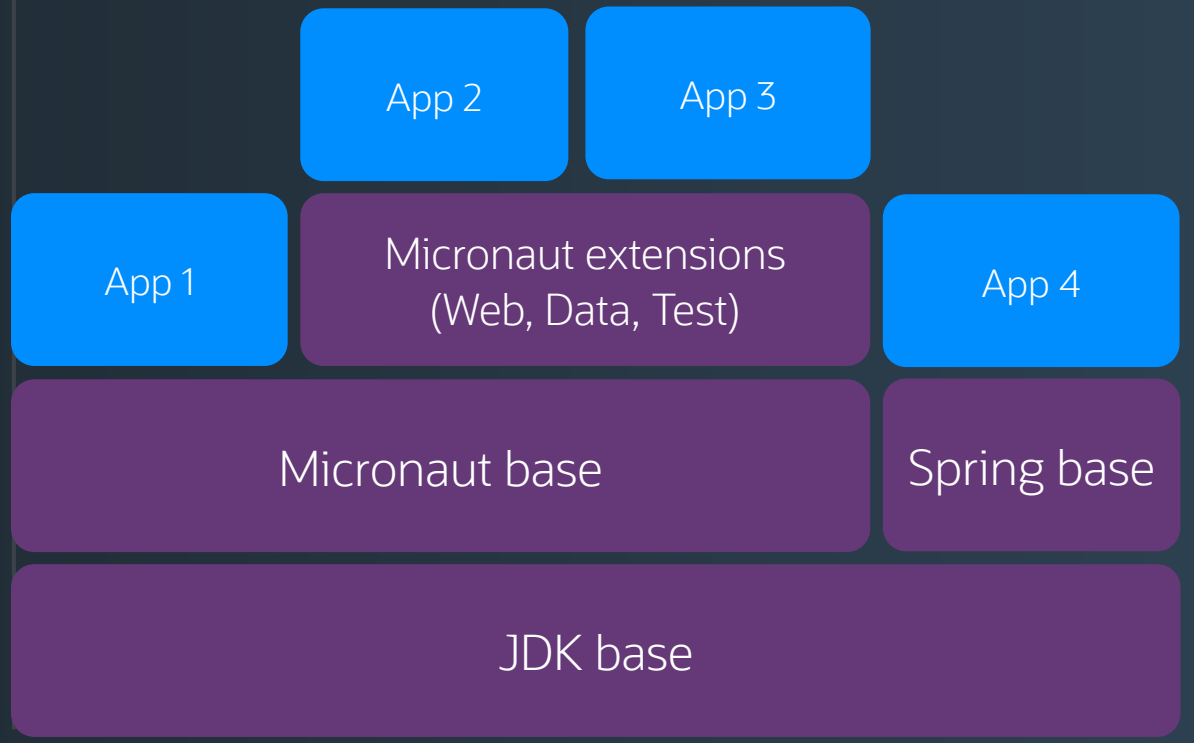
Learn more: medium.com/graalvm

Native Images Layers

Development: fast recompilation 🚀

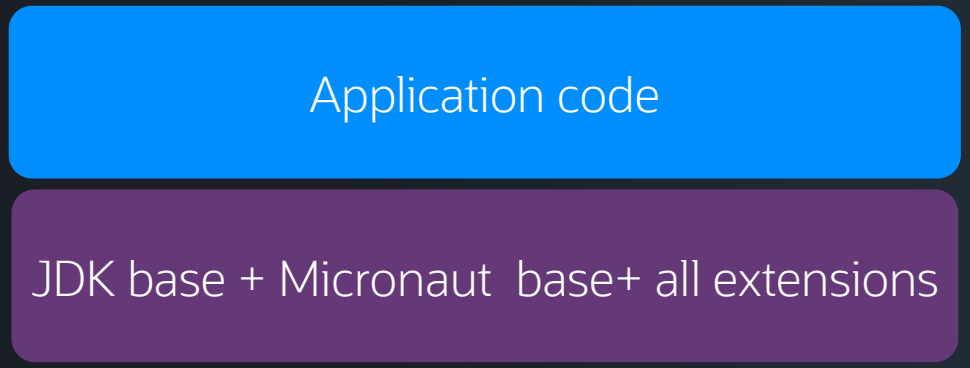


Deployment: resources sharing ☁️

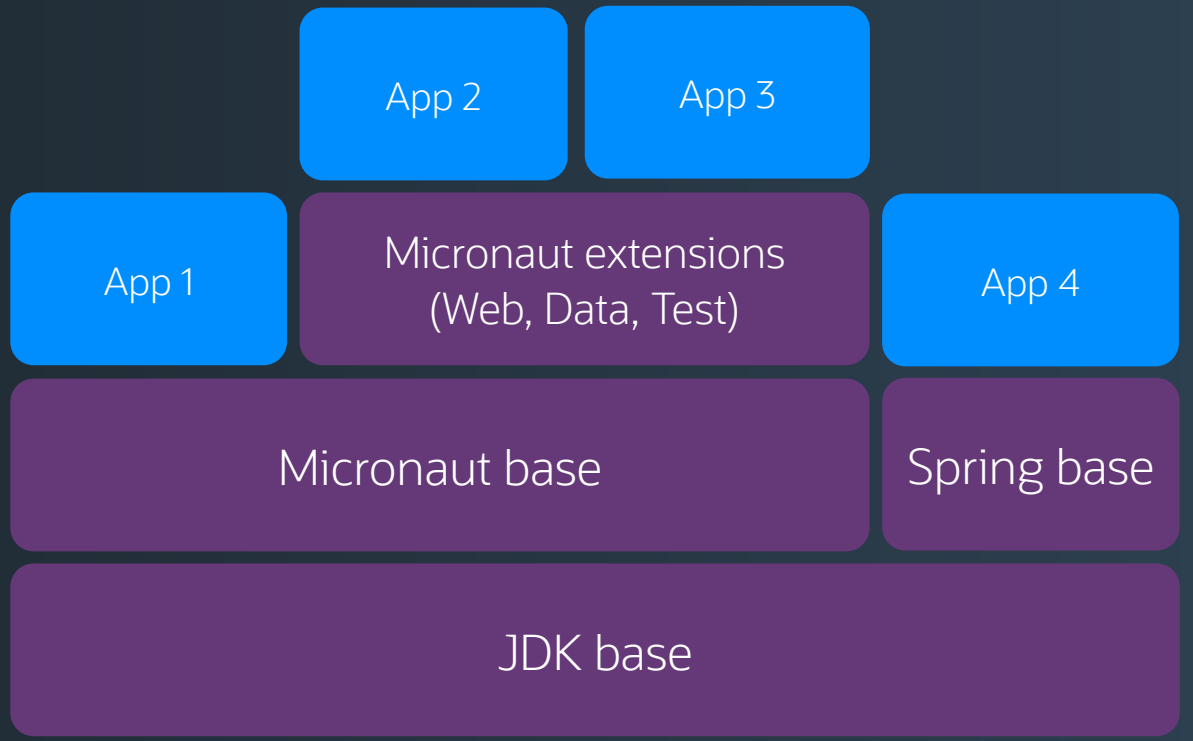


Native Images Layers

Development: fast recompilation 🚀



Deployment: resources sharing ☁️



Why GraalVM Native Image

Fast Start & Scale

Lower Resource Usage

Predictable Performance

Improved Security

Compact Packaging

MICRONAUT
spring
QUARKUS
helidon.io

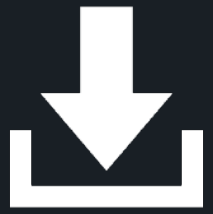
Azure
AWS
GCP
OCI

Supported

Get started with GraalVM 🚀

graalvm.org

docker pull container-registry.oracle.com/graalvm/jdk:24



sdk install java 24-graal

github.com/graalvm/graalvm-demos

Questions & let's connect!

Demos 👩‍💻


Thank
you


ORACLE