

# GOTO **AARHUS 2023**

**#GOTOaar**

# Java in the Cloud with GraalVM

**Alina Yurenko**

Developer Advocate for GraalVM

Oracle Labs

GOTO Aarhus

Steffen Muldbjerg @ Unsplash

# What GraalVM offers

## More performance with the Graal compiler

- Run your Java application faster
- New JIT compiler optimizations

## Fast startup with Native Image

- Create standalone binaries with low footprint
- Instant performance

## Polyglot VM

- Interop: extend your Java application with libraries from JavaScript, Python, R...
- High performance for all languages
- Polyglot tooling

The logo for GraalVM, featuring the word "Graal" in blue and "VM" in orange, with a small "TM" trademark symbol to the right.

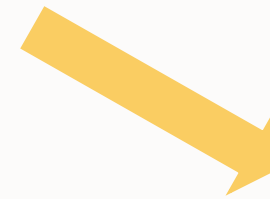


**GraalVM™**



**JIT**

`java MyMainClass`



**AOT**

`native-image MyMainClass  
./mymainclass`



## GraalVM 🤝 Java microservice frameworks



# GraalVM™

## Java in the Cloud - Goals



**Start Fast**



**Low Resource  
Usage**



**Minimize  
Vulnerability**



**Compact  
Packaging**

# Java in the Cloud - Goals



**Start Fast**



**Low Resource  
Usage**



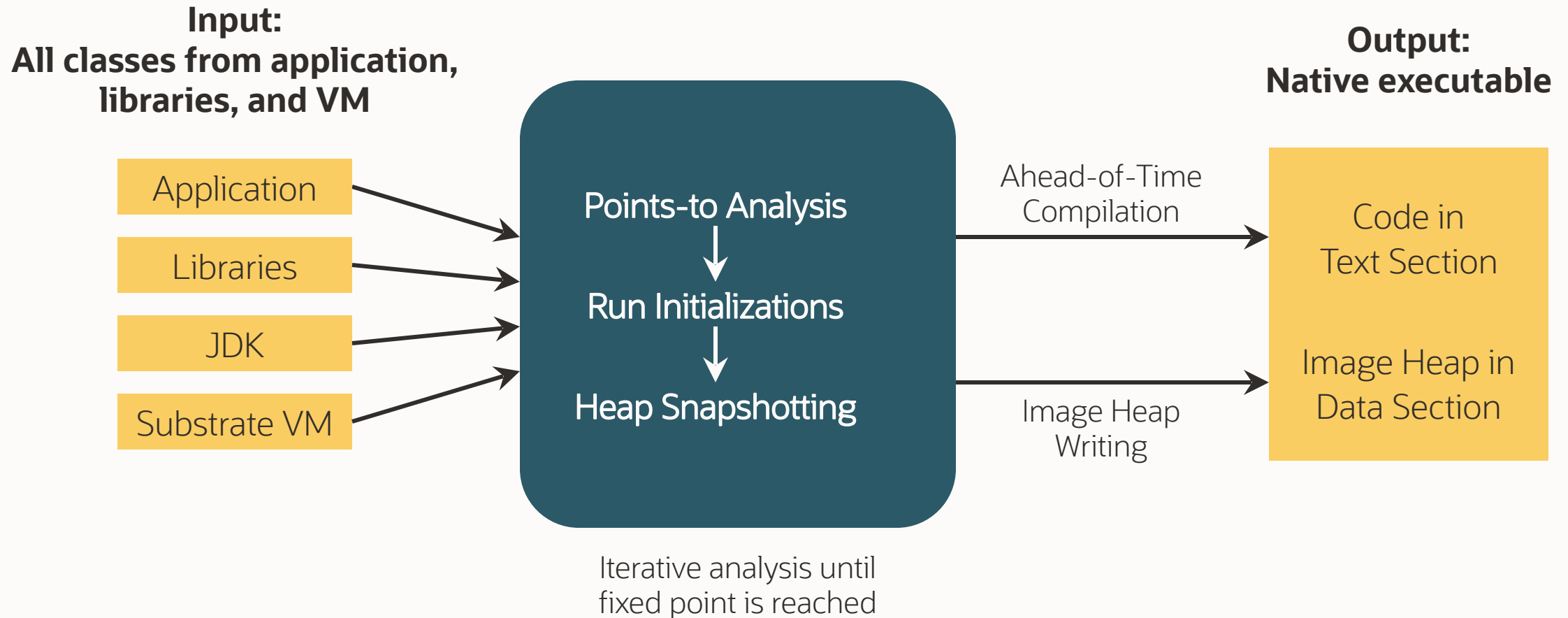
**Minimize  
Vulnerability**



**Compact  
Packaging**

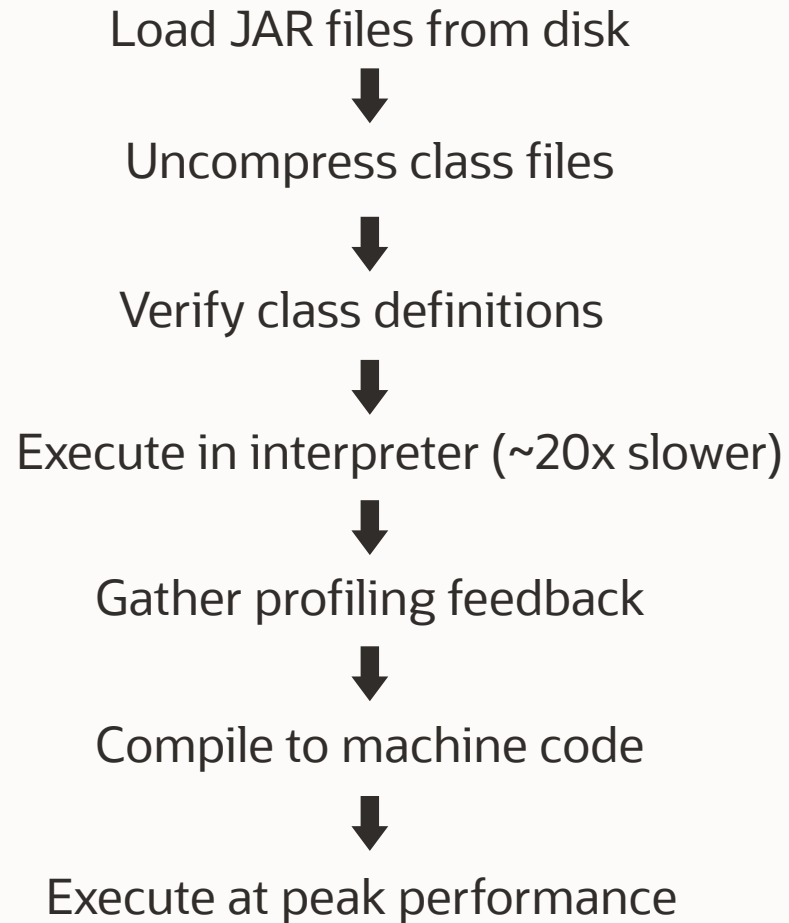


# Native Image Build Process

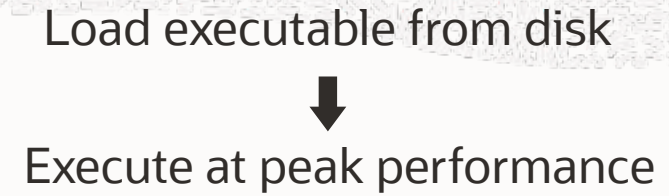




# JIT



# AOT



# Java in the Cloud - Goals



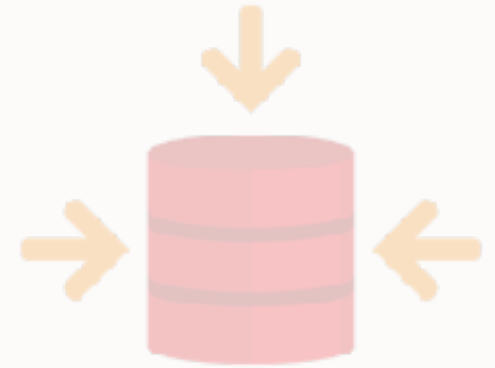
Start Fast



**Low Resource  
Usage**



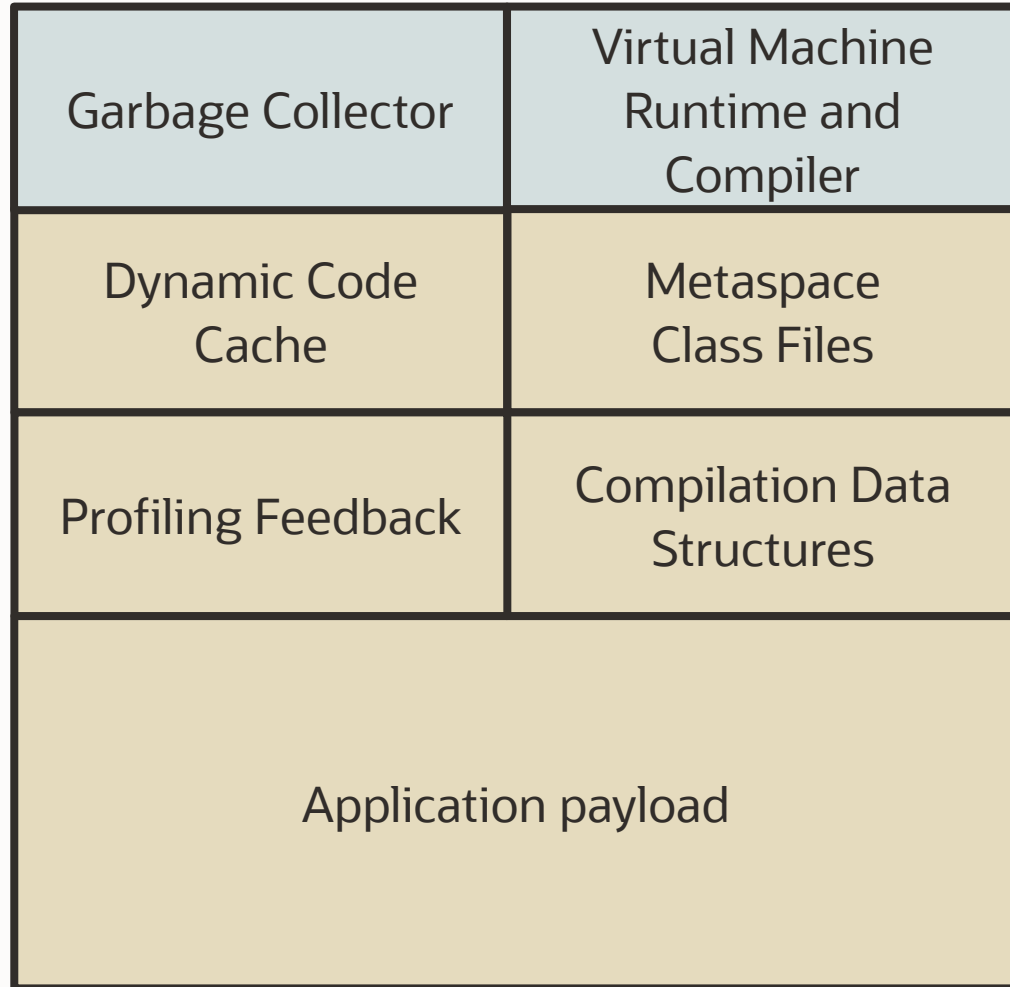
Minimize  
Vulnerability



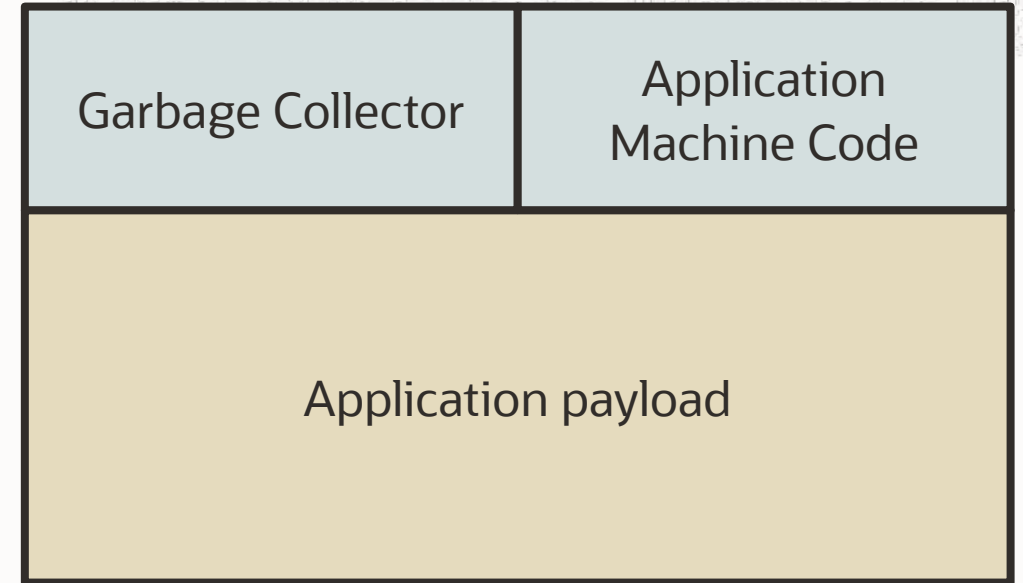
Compact  
Packaging

# JIT

## Memory

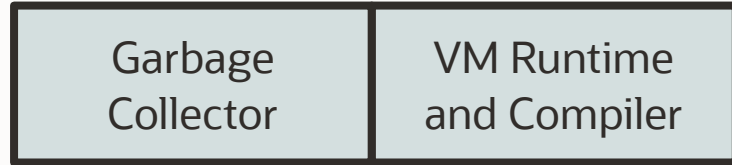


# AOT



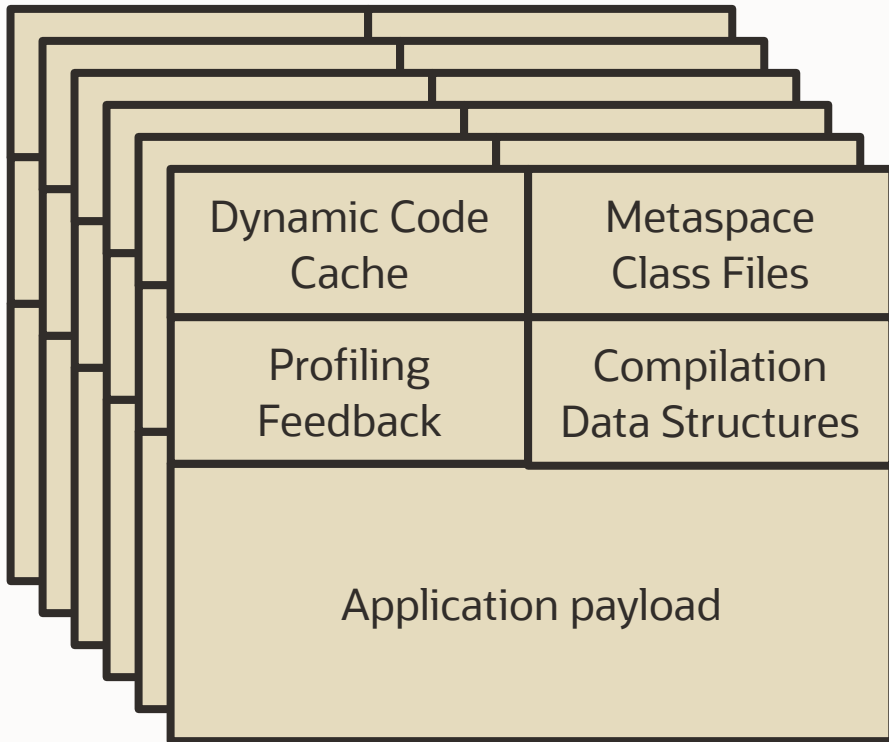
# Memory Scalability

## JIT

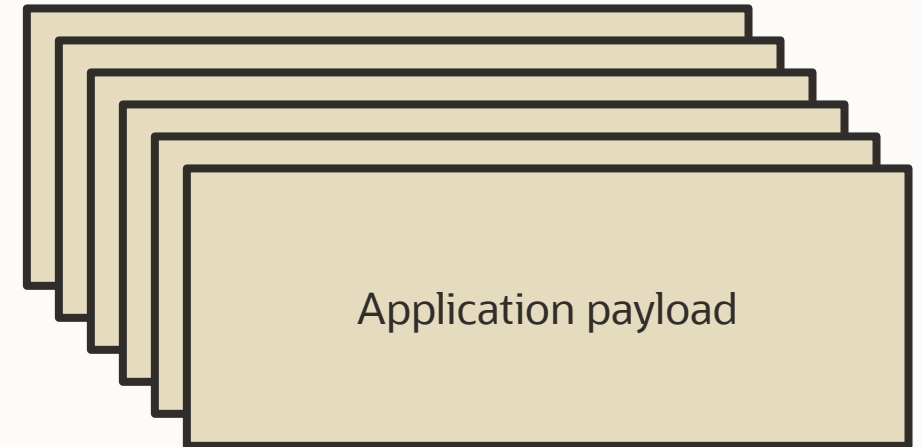
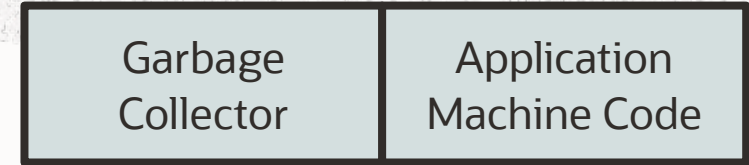
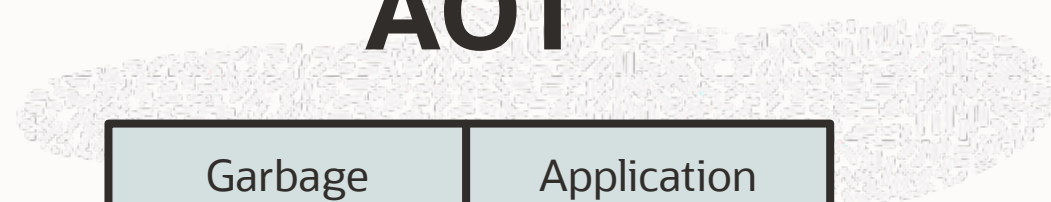


**shared**

**duplicated  
per process**



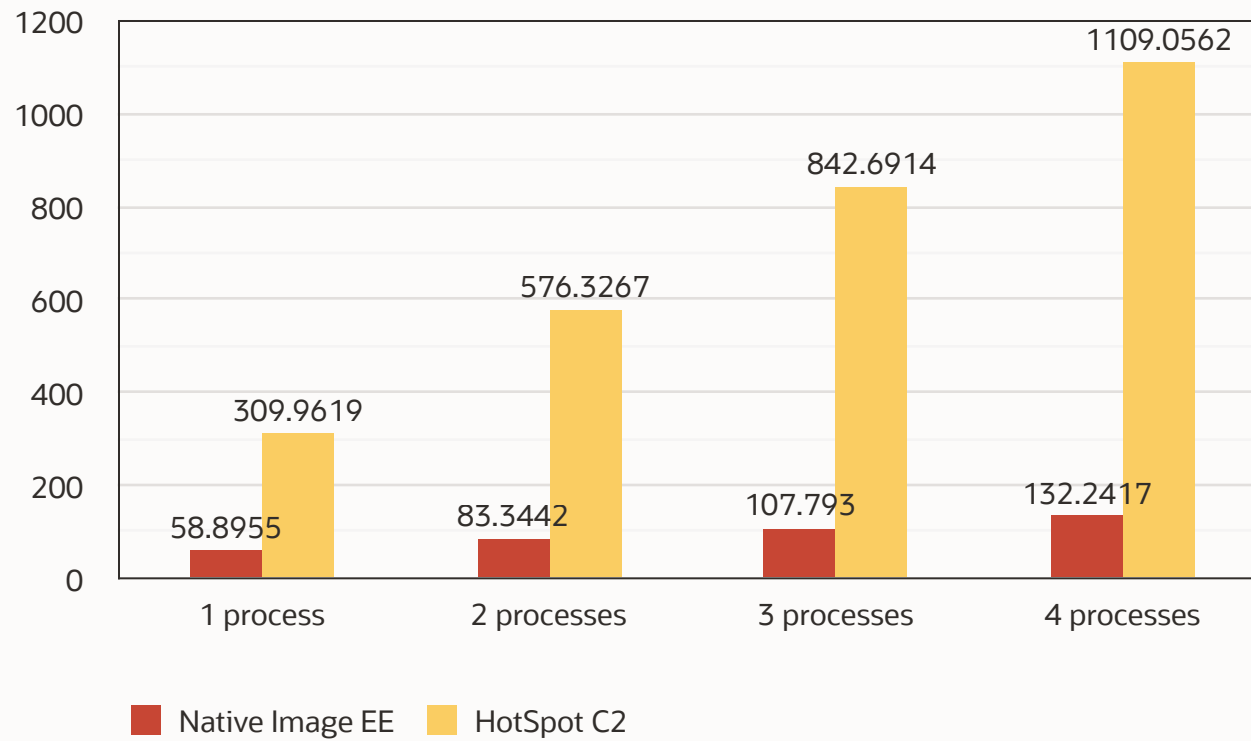
## AOT



# Example: horizontal scaling of microservices

## Memory Usage in MB

Quarkus Apache Tika ODT in a “tiny” configuration and with the serial GC  
(1 CPU core per process, -Xms32m -Xmx128m) – JDK 11



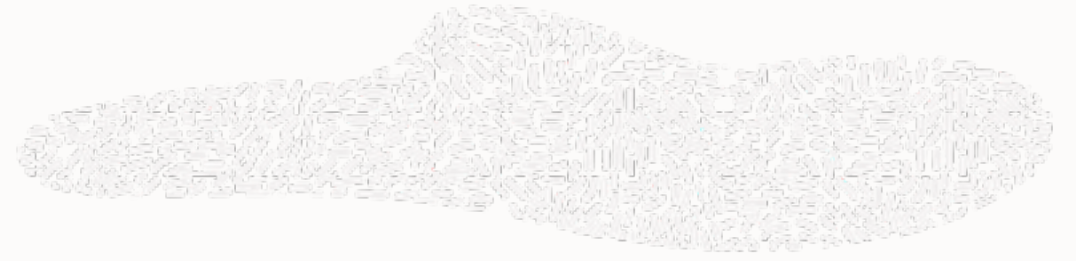
## Java HotSpot VM

- **4 VM instances = 4 times the memory**

## Native Image

- **4 VM instances = 2 times the memory**
- Image heap shared between processes
- Machine code shared between processes

## Demo: startup and performance



# Java in the Cloud - Goals



**Start Fast**



**Low Resource  
Usage**



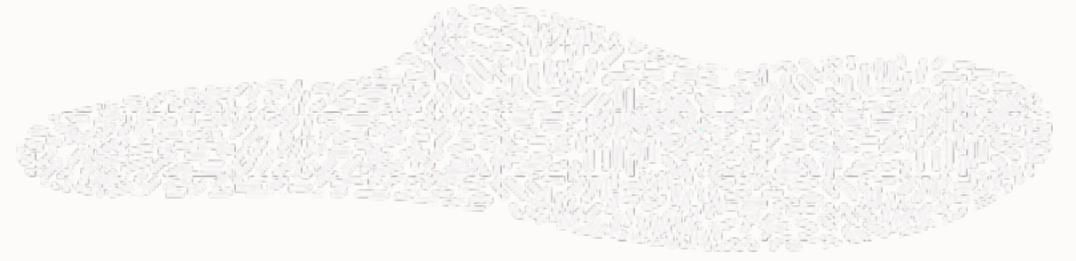
**Minimize  
Vulnerability**



**Compact  
Packaging**

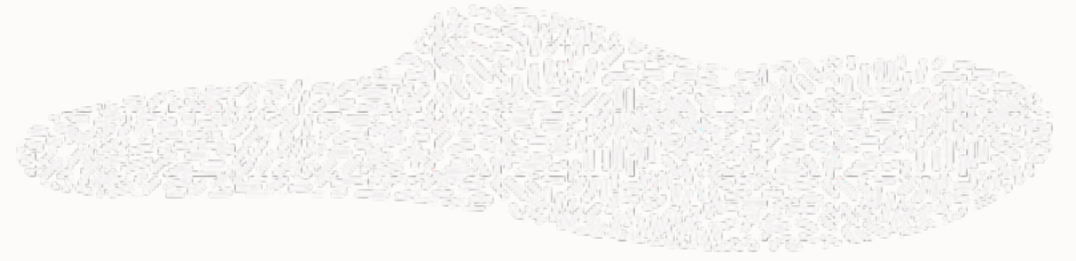


# Reduced Attack Surface



- No new unknown code can be loaded at run time
- Only paths proven reachable by the application are included in the image
- Reflection is disabled by default and needs an explicit include list
- Deserialization only enabled for specified list of classes
- Just-in-time compiler crashes, wrong compilations, or “JIT spraying” to create machine code gadgets are impossible

## Demo: reducing attack surface



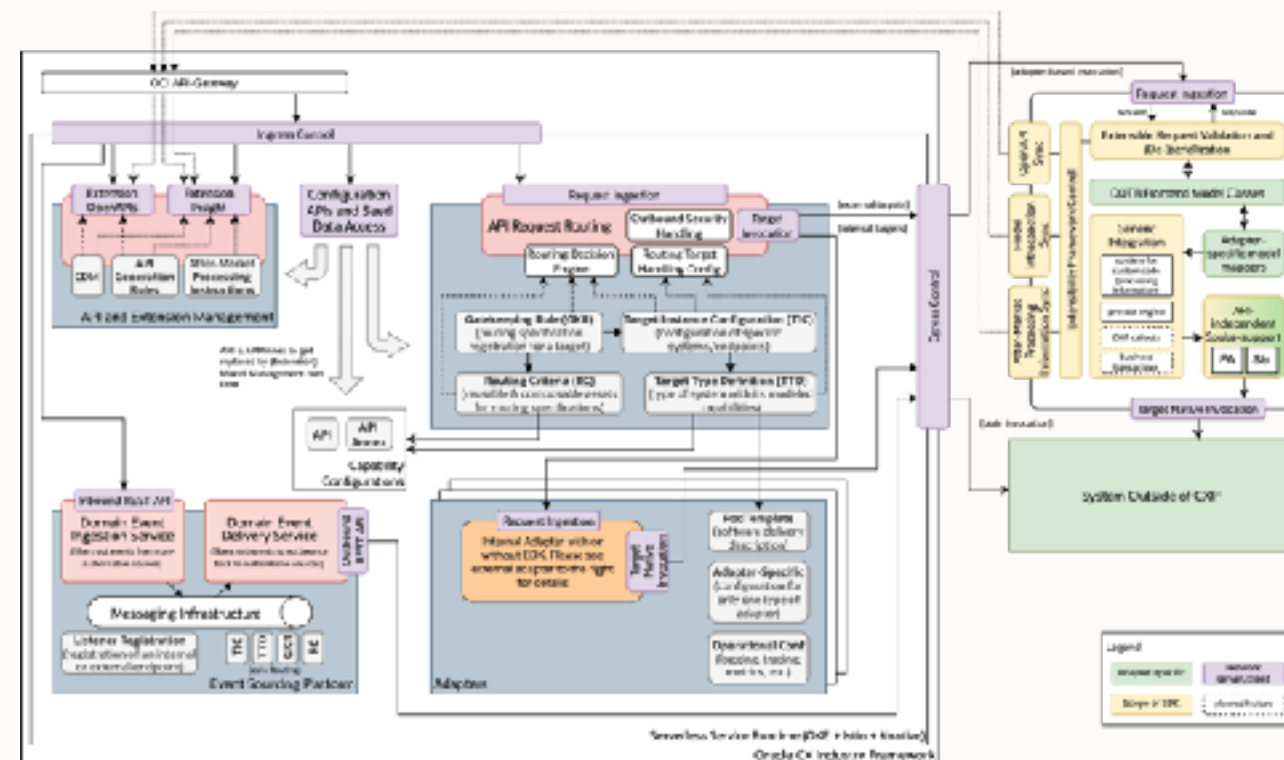
# and Native Image



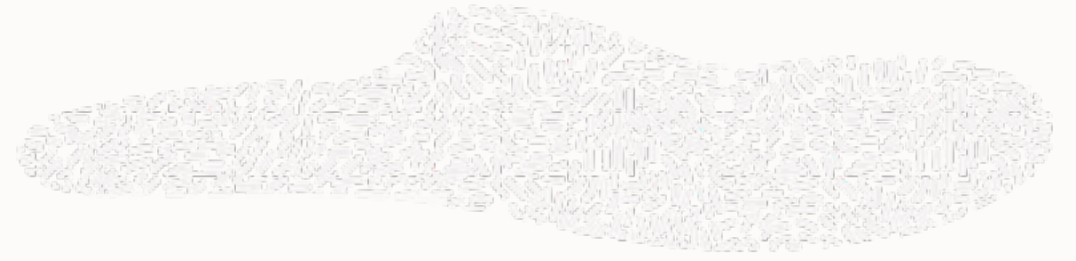
Native Image

- Use GraalVM Native Image to create minimum-size, precompiled executable images of its microservices: container images of <50MB
- “It’s a killer feature for security”

[medium.com/helidon/oracle-cx-industry-framework-a-helidon-flight-with-aerobatic-stunts-4666683d5176](https://medium.com/helidon/oracle-cx-industry-framework-a-helidon-flight-with-aerobatic-stunts-4666683d5176)



# Java in the Cloud - Goals



**Start Fast**



**Low Resource  
Usage**



**Minimize  
Vulnerability**



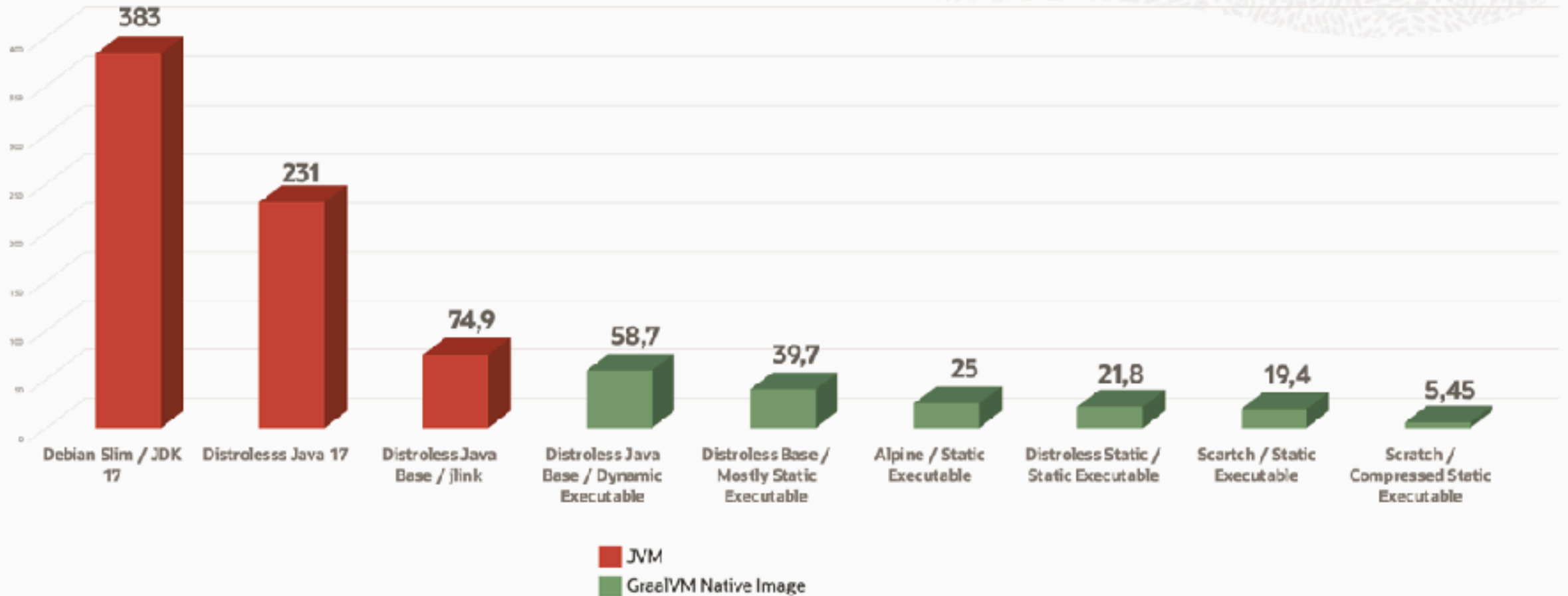
**Compact  
Packaging**

# Single Native Executable



- All relevant JVM runtime and JDK library code is included
- Unreachable paths (i.e., dead code) in the application and its dependencies eliminated
- Disadvantage that Java runtime installation cannot be shared, but also advantage that applications can be patched/updated individually

# Lightweight containerized applications



YouTube: A 1.5MB Java Container App? Yes you can! by Shaun Smith

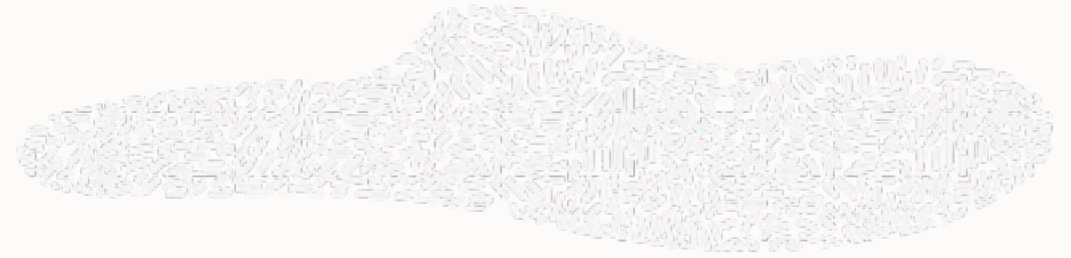




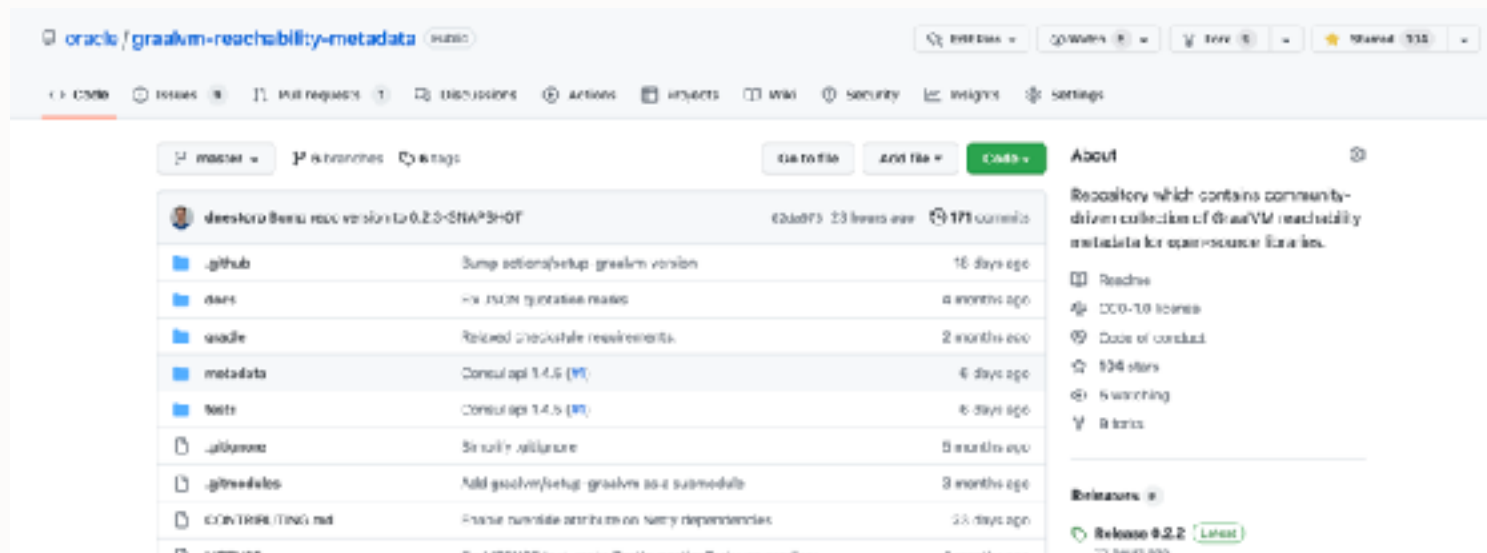
**What's the catch?**



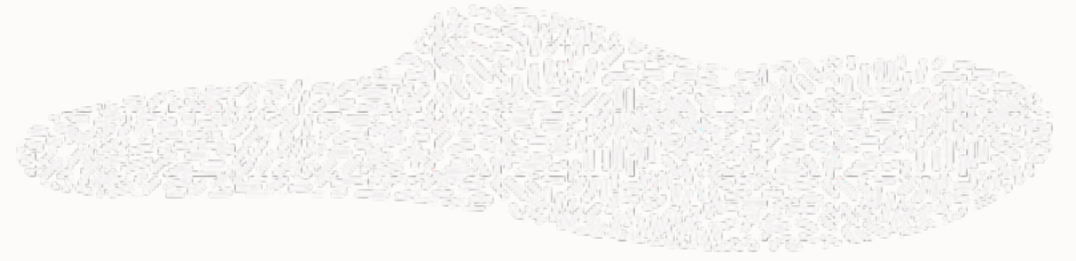
# GraalVM & Reflection?



- GraalVM 🤝 Reflection!
- Native Image tries to resolve the target elements through a static analysis that detects calls to the Reflection API
  - If the analysis can not automatically detect your use of reflection, you might need additional configuration
- Trace reflection, JNI, resource usage on the JVM with the tracing agent
  - Manual adjustment / addition might still be necessary



# Required Build Time Step



- Computational effort necessary at build time
- Need a powerful machine with the same target architecture & OS
  - Use GraalVM with GitHub Actions: [github.com/marketplace/actions/github-action-for-graalvm](https://github.com/marketplace/actions/github-action-for-graalvm)
  - Many larger apps can build with 2 GB of memory
- Develop in JIT mode for fast development, only use AOT for final deployment
- For best throughput, use profile-guided optimizations

# What's new in GraalVM

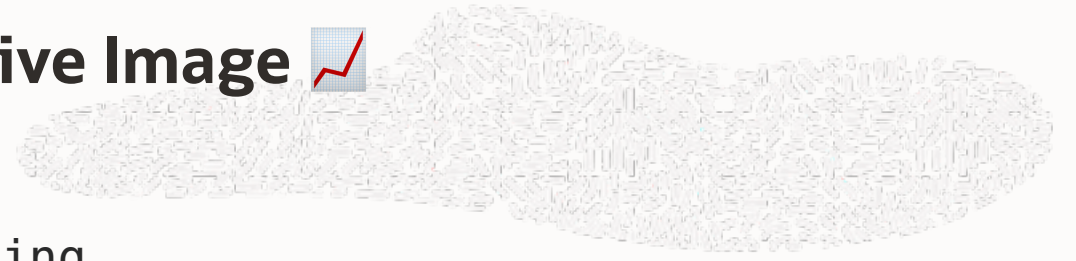
# GraalVM™

# JDK 20

dev builds

now available! 🎉

# New monitoring features in GraalVM Native Image



- `-H:+AllowVMInspection -> --enable-monitoring`
  - `--enable-monitoring=<all,heapdump,jfr,jvmstat>`
- added support for `jvmstat` in Native Image
- keep building out the JFR support in Native Image (thanks to Red Hat for their contributions!)



# GraalVM Community roadmap on GitHub

GraalVM Community Roadmap

Native Image + Compiler

Language Runtimes

+ New view

focus-area:Native Image + Compiler

19

X

Title	Assignees	Status	Labels	Notes	
22.3.0 Release (October 25, 2022) 7					
1  GraalVM JDK19 Builds #4957	gilles-duboscq	In Progress	feature		
2  Support Virtual Threads (Project Loom) in Native Image #4920	peter-hofer	Done	feature		
3  Complete the Third-Party API in Native Image #4919	christianwimmer and  olpaw	In Progress	feature		
4  [GR-40674] Perf support #4811	adinn and  olpaw	Done	native-image-debuginfo	Contributions from Red H	
5  [GR-40264] Introduce <code>--enable-monitoring</code> option. #4823	friephaus	Done	OCA Verified		
6  [GR-39475] [GR-40095] Initial jvmsat support in GraalVM Na #4803	christianhaeubl	Done	OCA Verified		
8  [GR-39497] Add support for JSON build output to GraalVM Na #4685	jerboaa	Done	oca-signed	Contributions from Red H	
+ Cannot add items when grouped by milestone					
23.0.0 Release (January 24, 2023) 3					
15  [GR-40463] Add initial support for remote management over J #4732	roberttoyonaga	In Progress	feature native-image	Contributions from Red H	
18  Add support for ZGC on HotSpot #5050	tkrodriguez	In Progress	compiler feature		
19  Add support for JDK19 and retire JDK11 support #5063		In Progress	feature		

<https://github.com/orgs/oracle/projects/6>



# What's next for GraalVM

## Add support for ZGC on HotSpot #5050

Open

tkrodriguez opened this issue on Sep 22, 2022 · 0 comments



tkrodriguez commented on Sep 22, 2022 · edited by fniephaus

Member



### TL;DR

Add support for [Z Garbage Collector](#) to the Graal compiler.

### Goals

Add required ZGC barriers on HotSpot along with any relevant performance optimizations, allowing the use of ZGC when the Graal is used as a JIT compiler.

### Non-Goals

- Add support for ZGC to GraalVM Native Image
- Add support for Shenandoah GC (although ZGC support will make it easier to support other GCs in the future)





## What's next for Native Image

- Simplifying configuration and compatibility for Java libraries
- Continuing with peak performance improvements
- Keep working with Java framework teams to leverage all Native Image features, develop new ones, improve performance, and ensure a great developer experience
- Further reduce build time and footprint of the Native Image builder
- IDE support for Native Image configuration and agent-based configuration
- Further improving GC performance and adding new GC implementations

# Get started with GraalVM

## Get started with GraalVM

```
bash <(curl -sL https://get.graalvm.org/jdk)\  
    graalvm-ce-java19-22.3.0
```

```
    sdk install java 22.3.r19-grl
```

## Java in the Cloud - Goals



**Start Fast**



**Low Resource  
Usage**



**Minimize  
Vulnerability**



**Compact  
Packaging**

📅 Tuesday May 23 ⌚ 10:50 – 11:40

📍 Lille Sal

# Next-Generation Cloud Native Apps with Spring Boot 3

The recent release of Spring Boot 3 laid the foundation for the next generation of modern Java applications. This session will highlight what's new, and demonstrate hands-on patterns and techniques for cloud native development.

Support for generating native executables with GraalVM is now part of the core framework, making it straightforward to build efficient applications with instant startup time and reduced memory consumption. The new Java 17 baseline and support for Jakarta EE 10 unlock many new features and integrations.



Thomas Vitale

*Software Architect, author of "Cloud Native Spring in Action" and certified Red Hat enterprise application developer*

# Thank you!

Alina Yurenko

[@alina\\_yurenko](https://twitter.com/alina_yurenko)





Don't forget to  
**vote for this session**  
in the **GOTO Guide app**