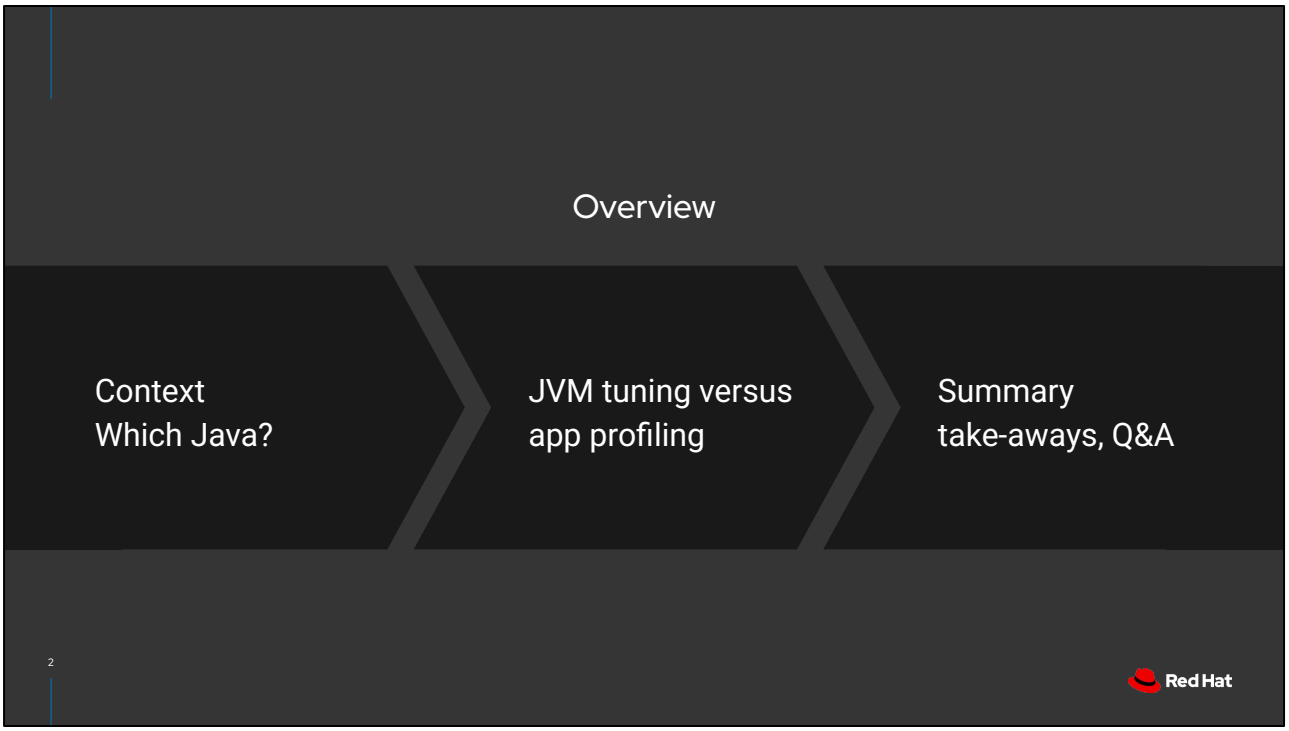
It’s an honour to be the warm-up act :)

Poll: first FOSDEM?
(approx half of audience!)



Three-parter:

1. Background, scene-setting. Which Java?
2. Java memory; JVM tuning; GC selection
3. Application profiling
4. Wrap-up

EnterpriseFactoryFactory?

```
properties:
  spec:
    properties:
      data:
        properties:
          mapping:
            properties:
              components:
                type: array
                items:
                  properties:
                    pushSourceContainer:
                      enum:
                        - true
```

Assume you aren't "Java people"
Confession: I wasn't a Java person
Java is a Unique OSS success story

—

Let's get this out of the way right at the beginning

We've all heard the jokes about the quality of Enterprise Java code, FactoryFactories and what-not. I'm lucky to have never experienced such horrors personally

Let’s not throw stones in glass houses: here’s some actual YAML from an actual container pipeline system (witholding the names of the guilty) that I have to work with

Besides verbose, enterprise, let’s not forget: Java is a unique open source success story. It was not a greenfield open source development like Linux. It was a proprietary source until ~2010 (fin), when it was open sourced despite it underpinning billion dollar business, completed _after_ Oracle acquired them. Today it’s developed in cooperation between Oracle, Amazon, Red Hat, SAP, etc – astonishing

OpenJDK is a *source* distribution

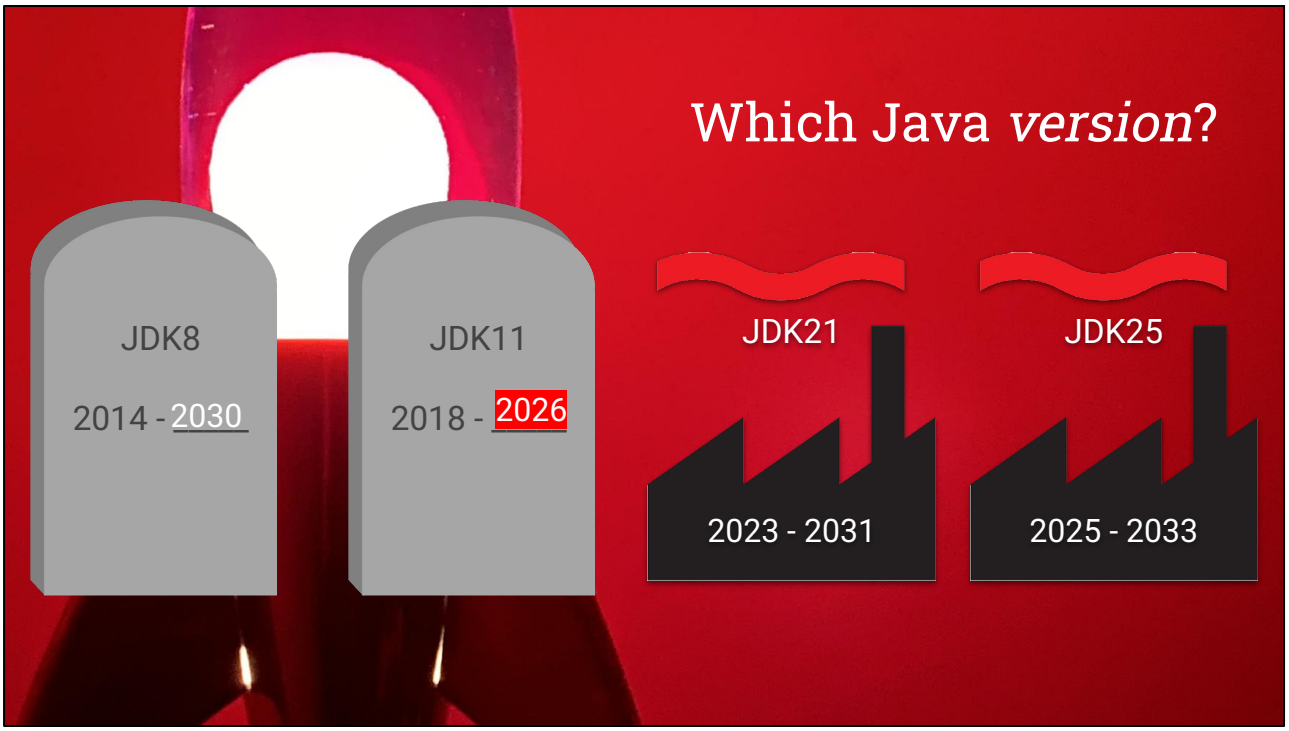
Independent vendors provide *builds* of OpenJDK – features may be turned on or off; extra patches etc., so vendor matters

If in doubt/no idea, try Temurin

other JVMs (not OpenJDK/hotspot) out of scope today

—

https://adoptium.net/en-GB/temurin



New feature release every 6 months
New LTS every 2 years
New patch release every quarter

Support is a separate question

—

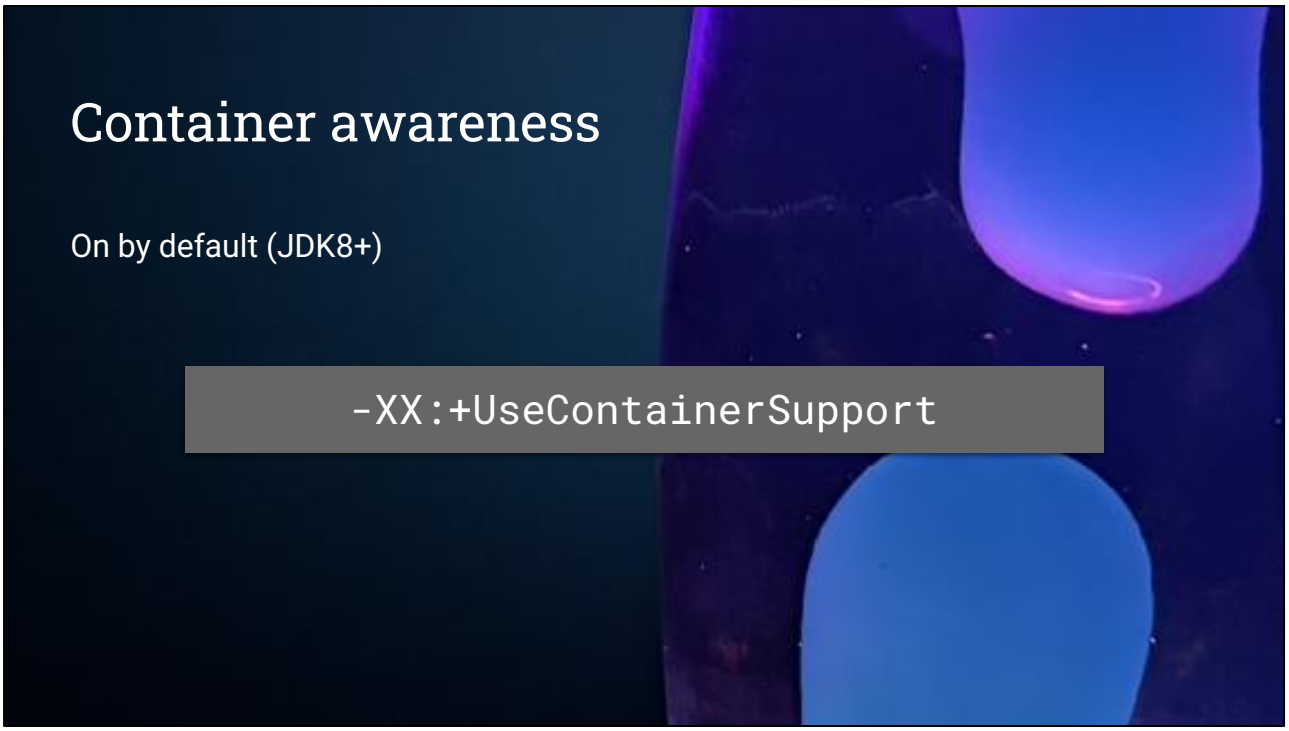
https://access.redhat.com/articles/1299013
https://mreinhold.org/blog/forward-faster

(clipart: openclipart.org/217613)

One way to manage memory: compile to native
GraalVM, Oracle Labs
Quarkus (Java batteries included framework) makes it easier
This is all I will say about it today

—

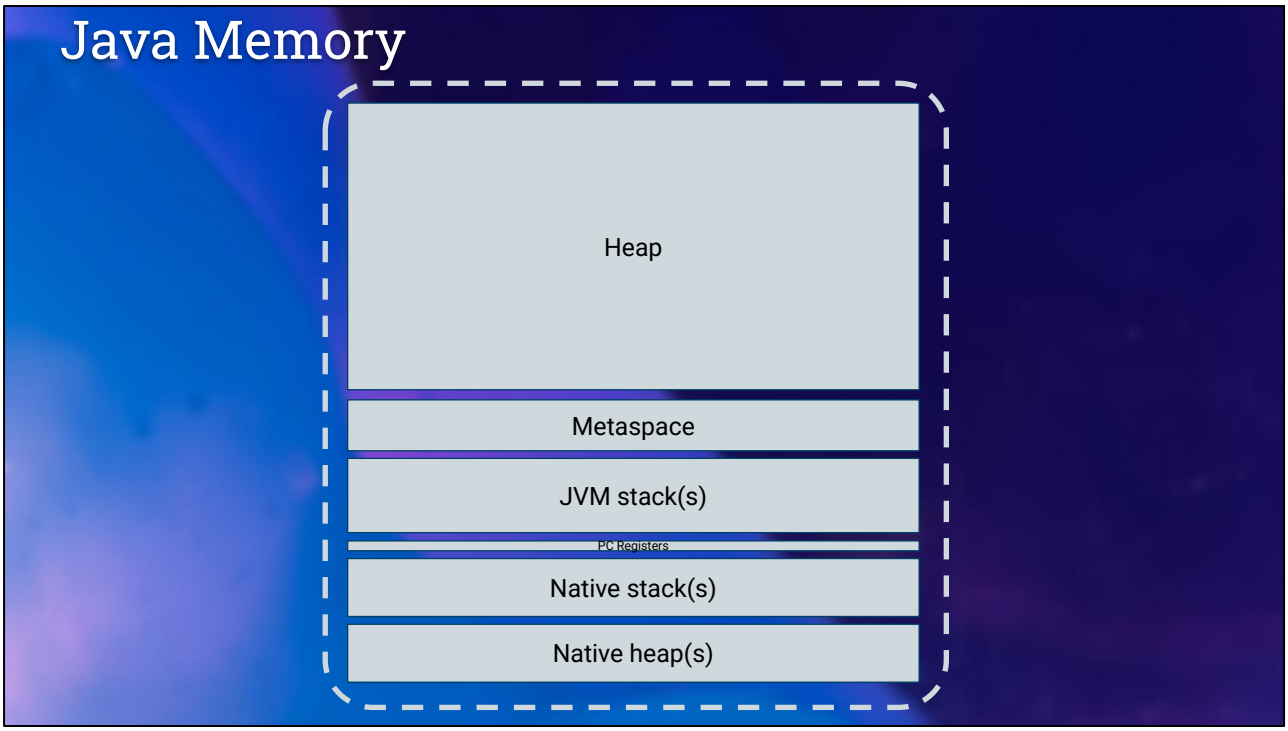
https://archive.fosdem.org/2024/schedule/event/fosdem-2024-1876-exploring-quarkus-native-choices-and-implementation/



Reads memory limit from cgroups v2 or v1
(cgroups: so good we did it twice)

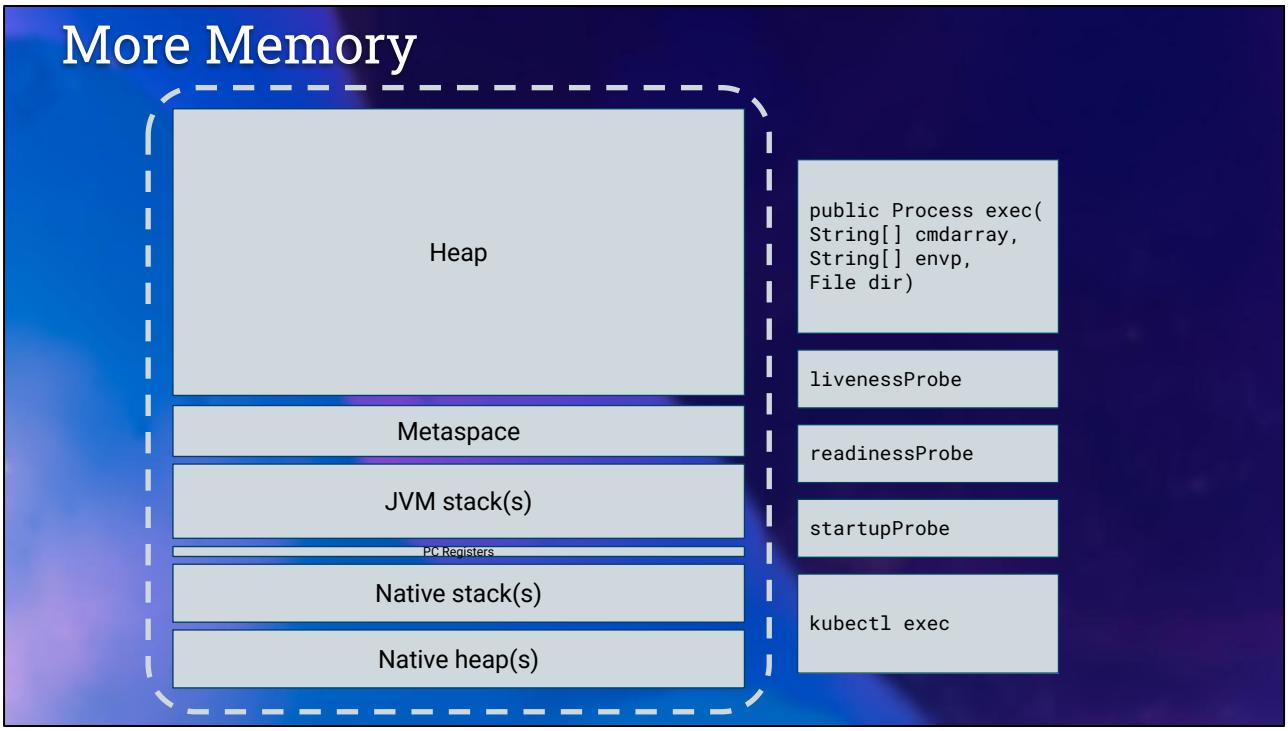
Old docs may have this flag, no longer needed

Backported to JDK8 and newer.



Approximation (some stuff omitted)

- Heap - GC operates on this. Likely the majority
- Metaspace: class metadata; static variables; not GC'd.
- JVM stack(s): one per JVM thread
- Program counters for JVM
- Native stack(s): one per OS thread
- Native heap! Netty uses a lot of this



In-container memory the JVM is not aware of

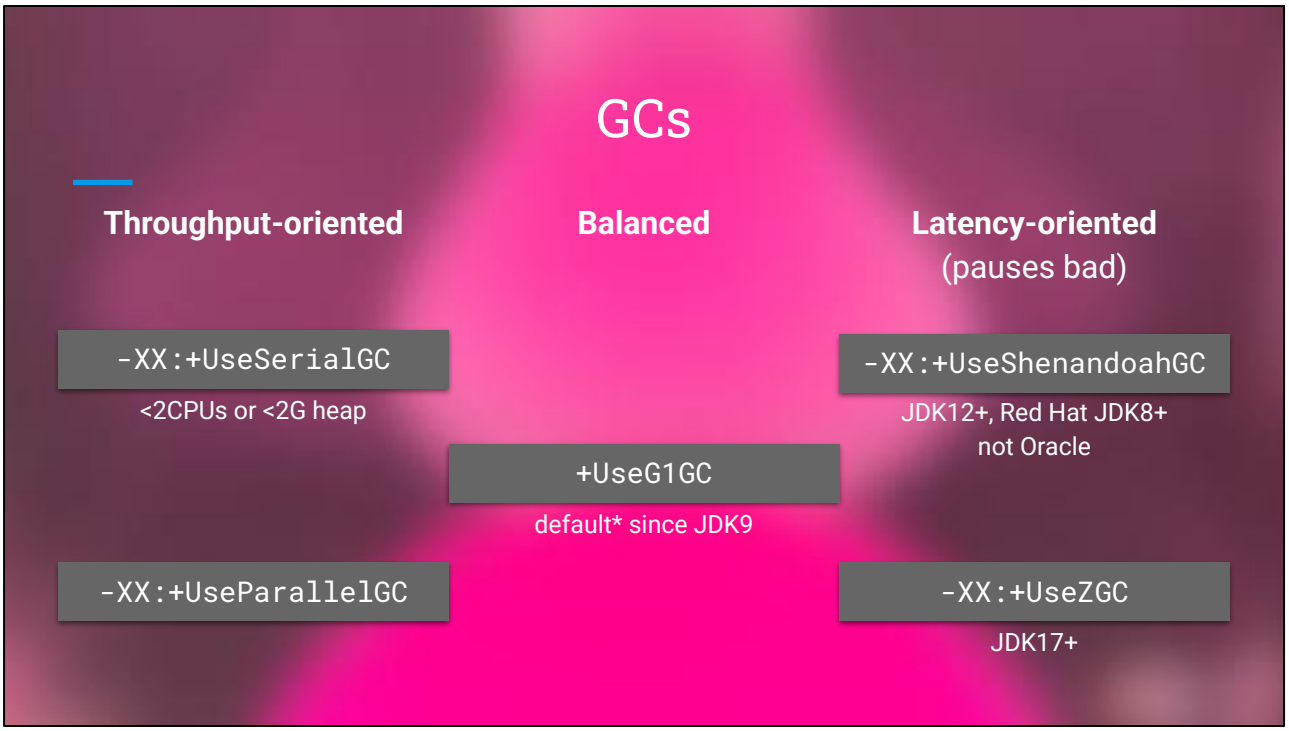
- Sub-processes spawned by the JVM
- Various probes from kubernetes
- Shell processes from sysops
- Perhaps more

So the JVM cannot be aware or in control of all memory in the container.

Need headroom for non-Heap RAM.

Are we in a container? What is a container? (leaky abstraction)

Can also define MaxMetaspaceSize (absolute values)



Lots of GCs!
Throughput: minimize time spent in GC (versus application time)
Latency: application response

G1 will be default all the time soon (JEP-523)

No GC pauses at all

EOM = kill (let external scheduler handle it)

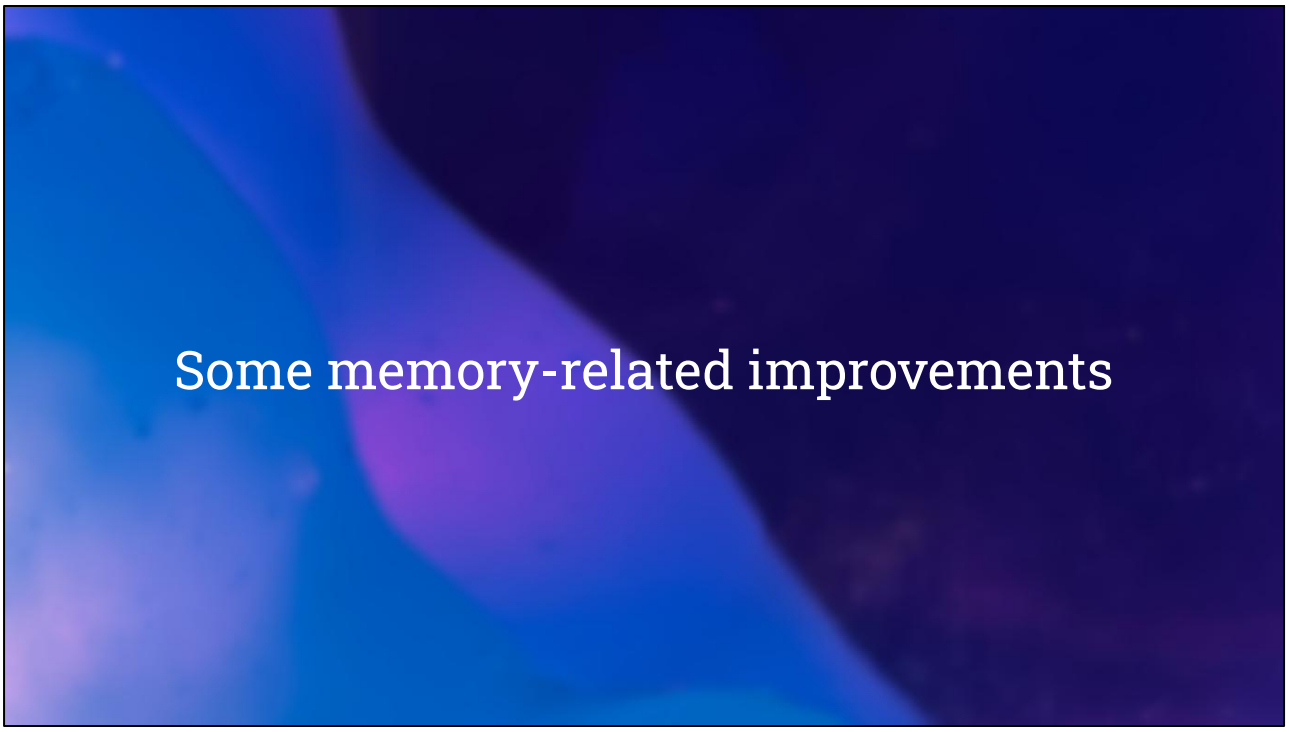
FaaS?

—

https://shipilev.net/jvm/diy-gc/

https://openjdk.org/jeps/318

https://archive.fosdem.org/2019/schedule/event/build_gc_minutes/



Use a recent JVM! Here’s why

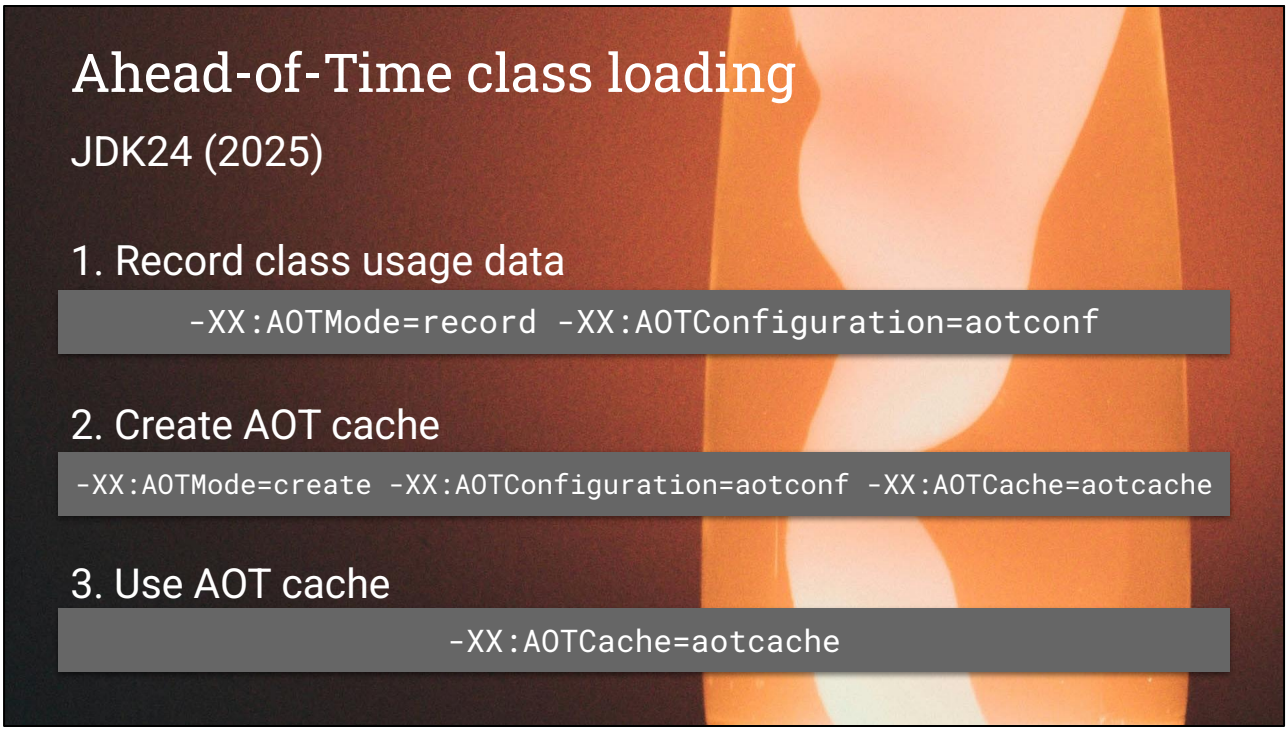
In more recent JDKs (>=21) option went away (balanced is the default)

—

https://community.sap.com/t5/technology-blog-posts-by-sap/jep-387-quot-elastic-metaspace-quot-a-new-classroom-for-the-java-virtual/ba-p/13497081

https://stuefe.de/posts/fosdem2020-metaspace-talk/fosdem2020-metaspace-talk/

https://archive.fosdem.org/2020/schedule/event/metaspace/



Some class initialisation (etc) can be cached to speed up future executions
Steps 1 & 2 may coalesce in the future

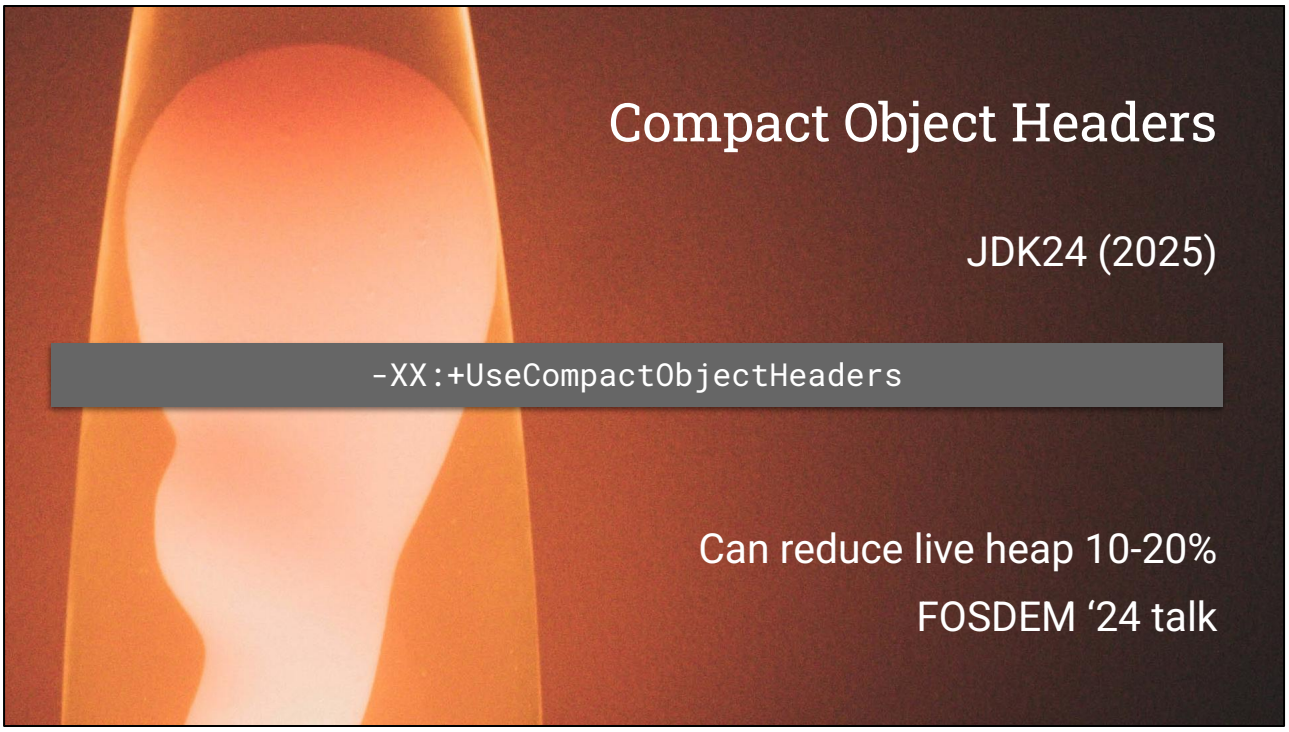
FOSDEM ‘25 talk

—

https://openjdk.org/projects/leyden/

https://archive.fosdem.org/2025/schedule/event/fosdem-2025-5469-project-leyden-past-and-the-future/

https://www.morling.dev/blog/jep-483-aot-class-loading-linking/

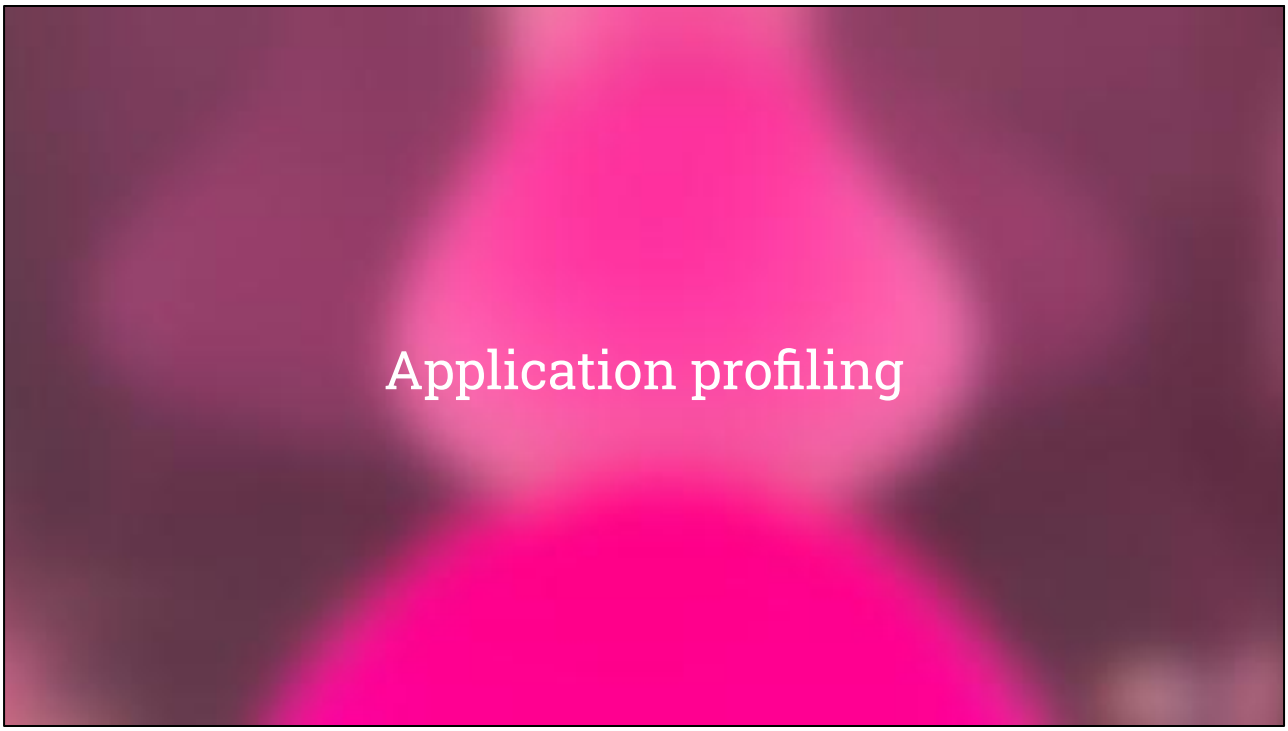


Typical workloads have lots of small objects. Reducing the per-object footprint has large memory gains

—

https://archive.fosdem.org/2024/schedule/event/fosdem-2024-3015-project-lilliput-compact-object-headers/

JEP 450 “Project Lilliput”


Application profiling

Java Flight Recorder

(& Java Mission Control)

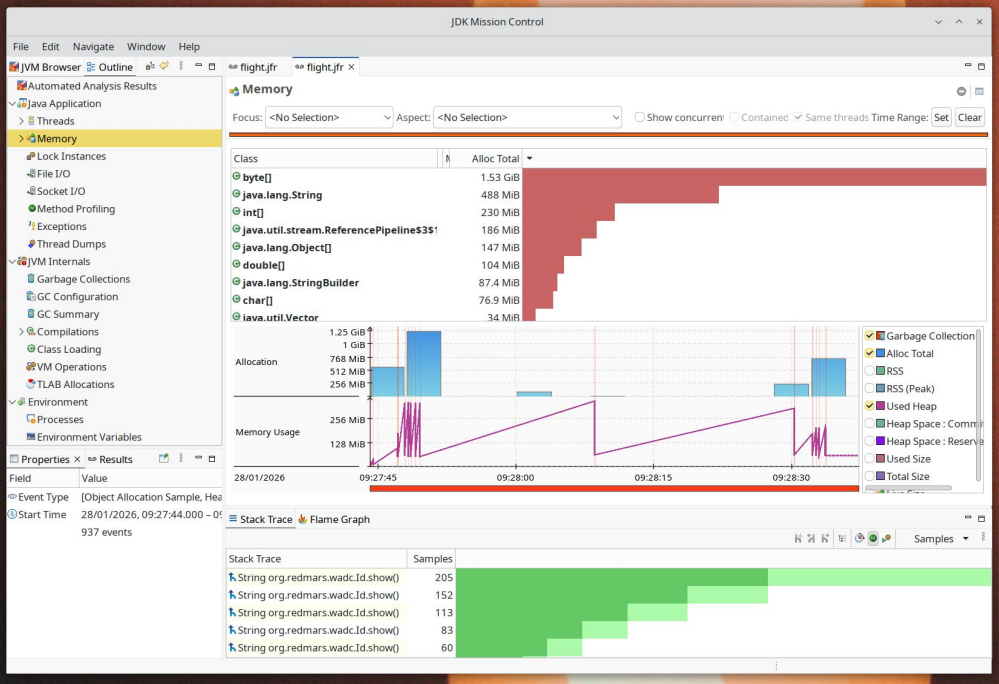
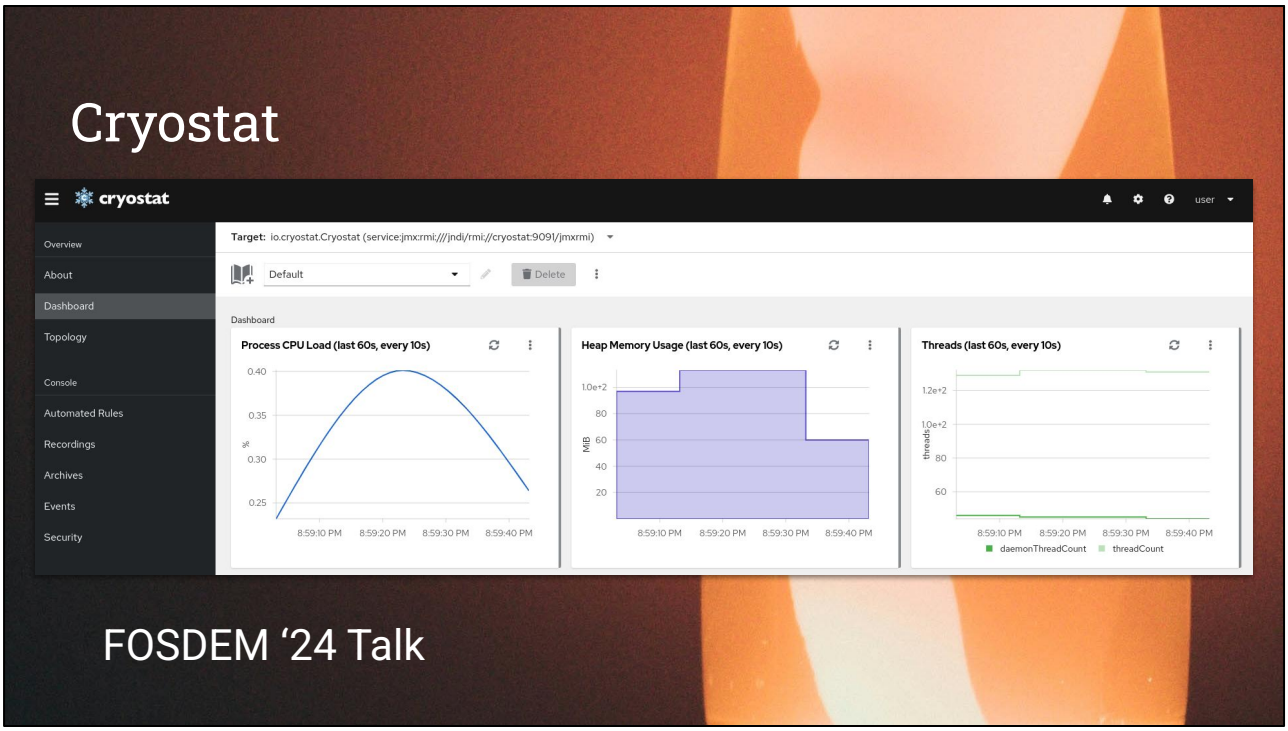


Illustration of Java Mission Control (jmc)
Browsing data collected by Java Flight Recorder (jfr)
In JDK8 onwards



Manage JFR (and more) securely in container deployments

—

https://archive.fosdem.org/2024/schedule/event/fosdem-2024-2336-cryostat-jfr-in-the-cloud/

https://cryostat.io/

MY question for the audience: other managed languages / GC languages, interesting strategies?