QUARKUS

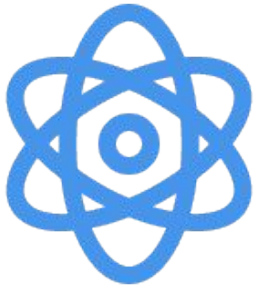
Supersonic Subatomic Java @ FOSSCOMM 2021

Dimitris Andreadis
Engineering Director
Quarkus Team, Red Hat

@dandreadis

QUARKUS

A stack to write Java apps

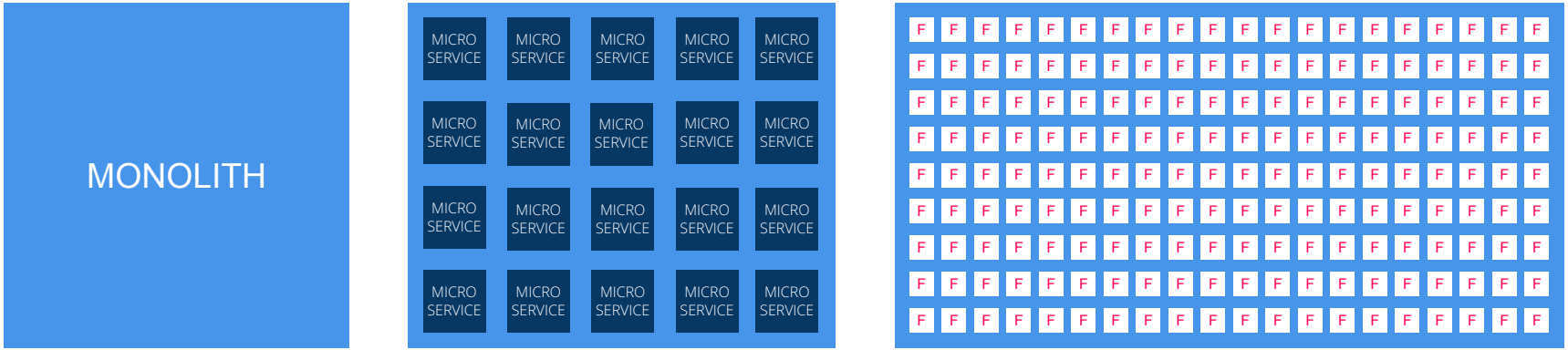


Cloud Native, Microservices, Serverless

What is the best way to write Cloud Native Applications

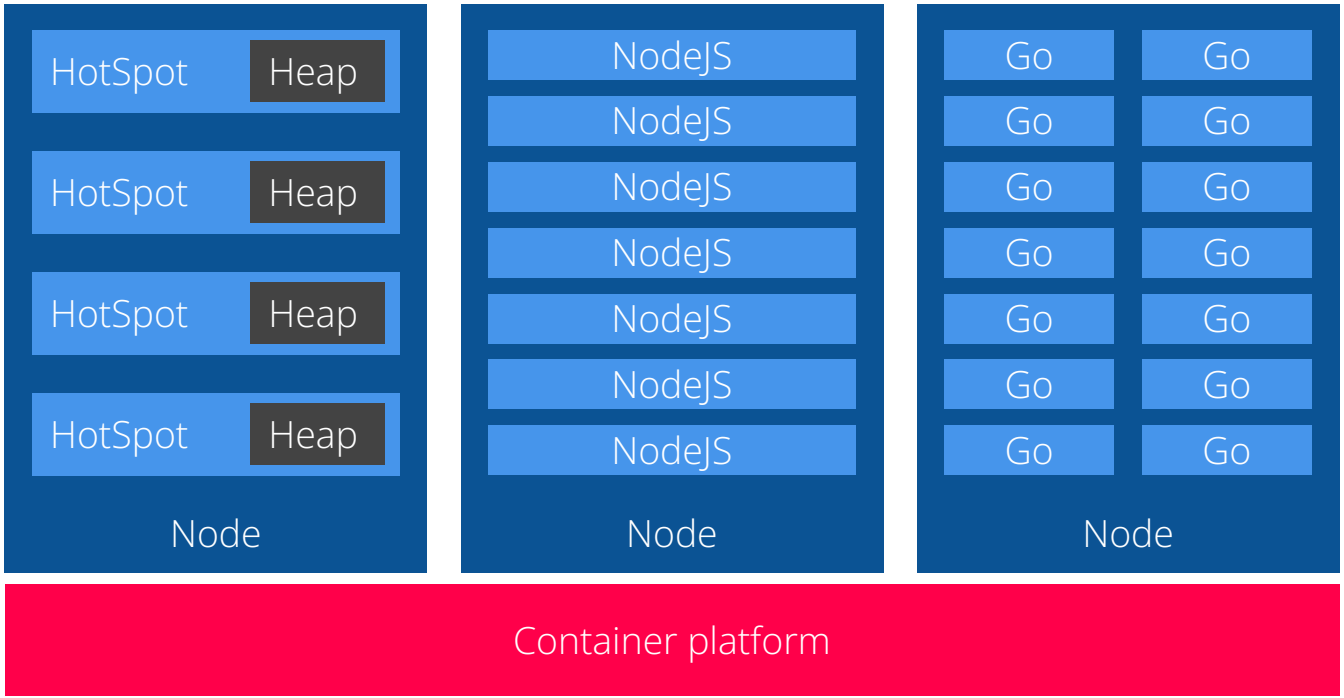
in Java

In a Kubernetes Native world, size matters



- 1 monolith ≈ 20 microservices ≈ 200 functions
- Long running process(es), vs scale up/down, vs scale infinitely and back to 0
- Start up time and density become key

Deployment density matters



Java frameworks suffer from the same problems

What is Wrong with Java Frameworks

They load way too many classes

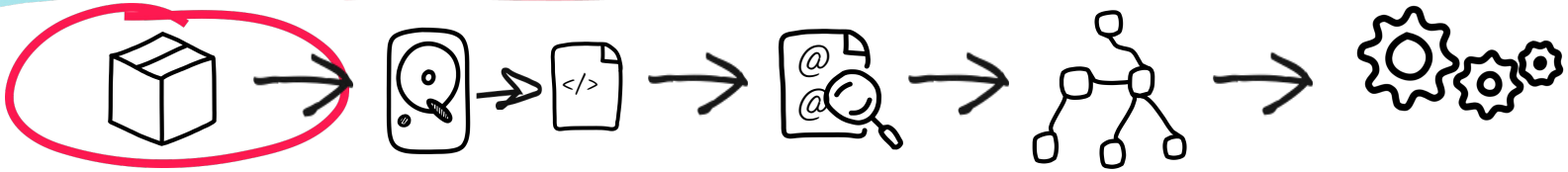
They are way too dynamic / reflective

They perform a lot of initialization at Runtime

How does a framework start?

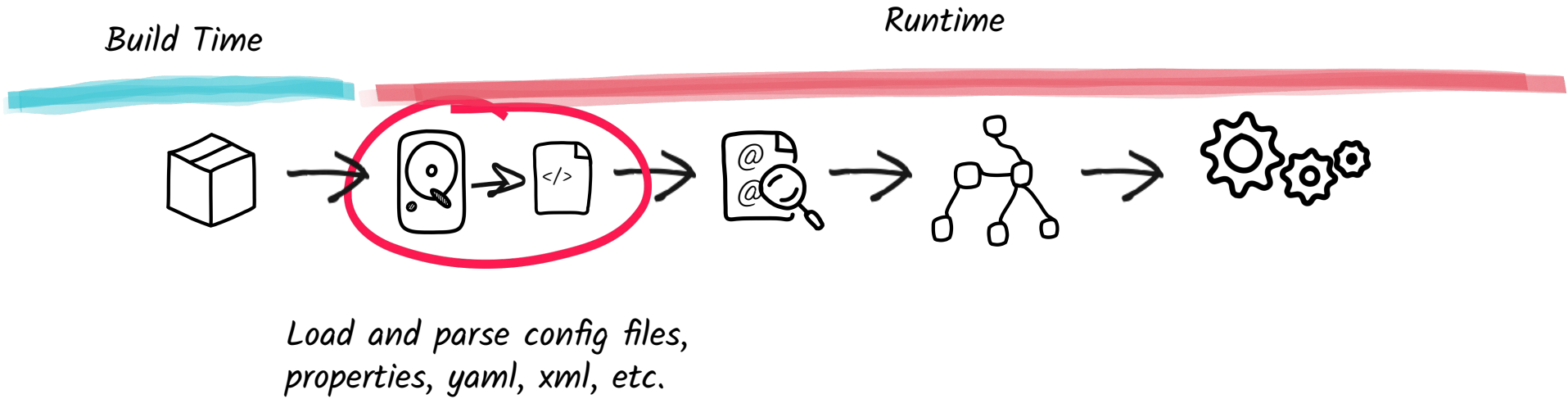
Build Time

Runtime



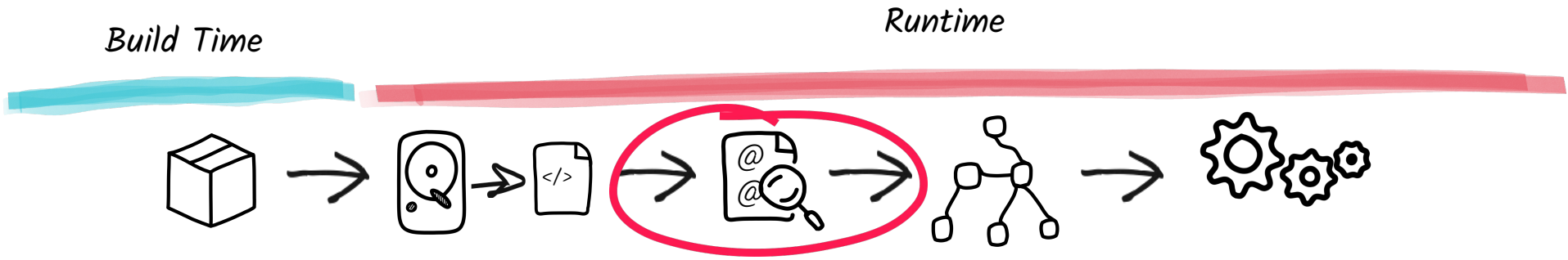
Packaging
(maven, gradle...)

How does a framework start?



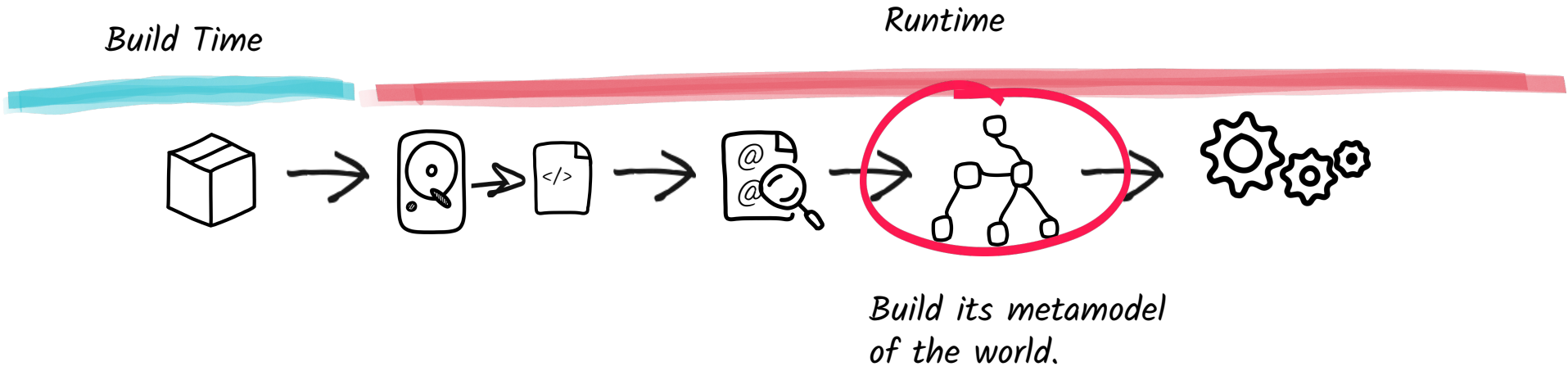
Load and parse config files,
properties, yaml, xml, etc.

How does a framework start?

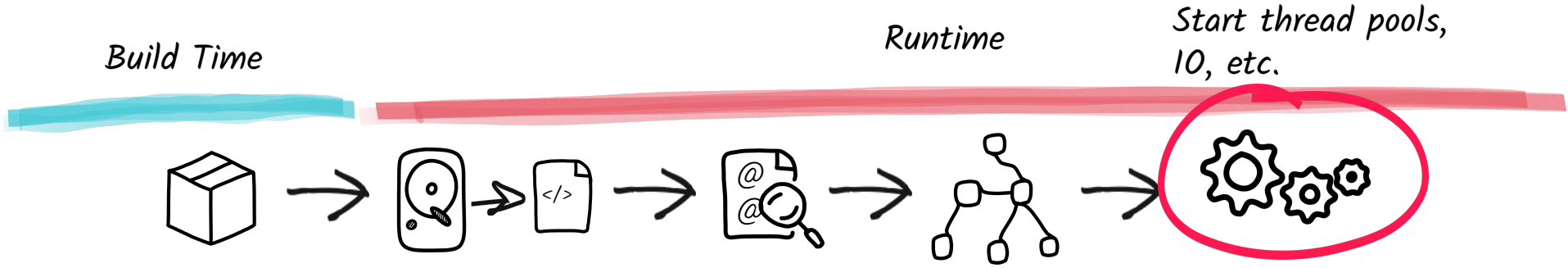


Classpath scanning and
annotation discovery
Attempt to load class to
enable/disable features

How does a framework start?

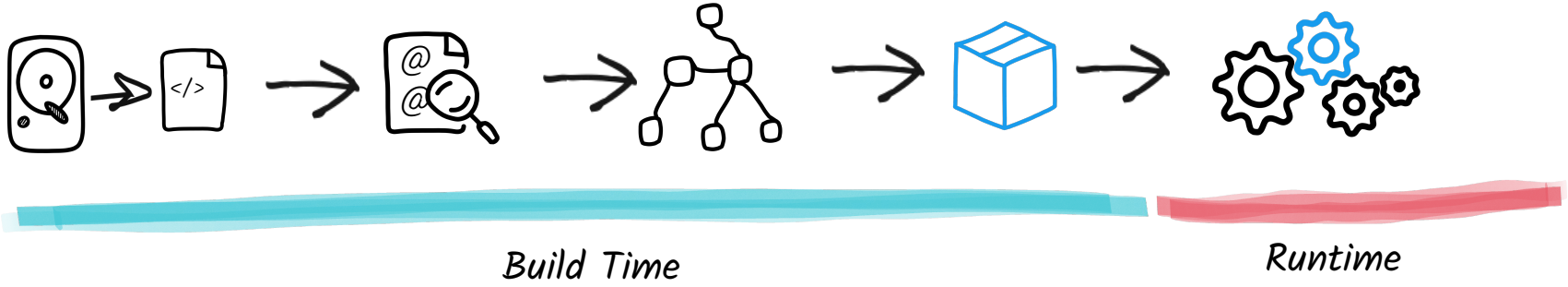


How does a framework start?



The basic idea behind Quarkus:

What if we perform Initialization at Build time?



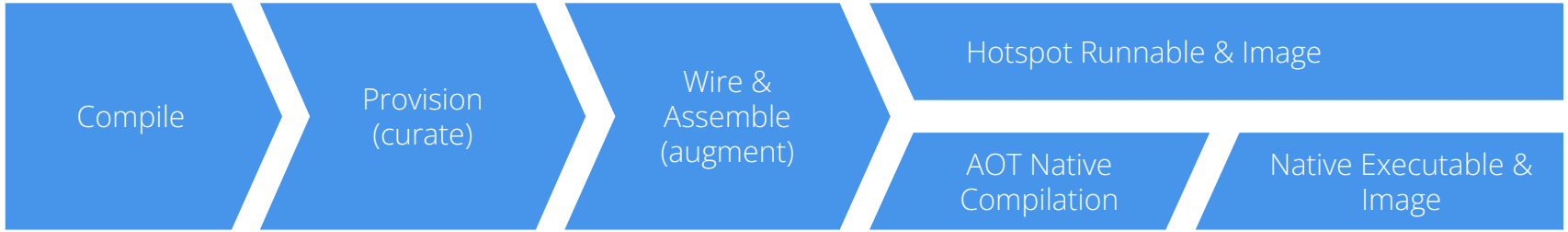
Do the work once, not at each start

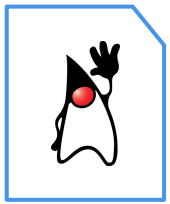
Get rid of all bootstrap classes

Less time to start, less memory needed

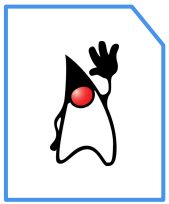
Less or no reflection nor dynamic proxies

Quarkus Build Process





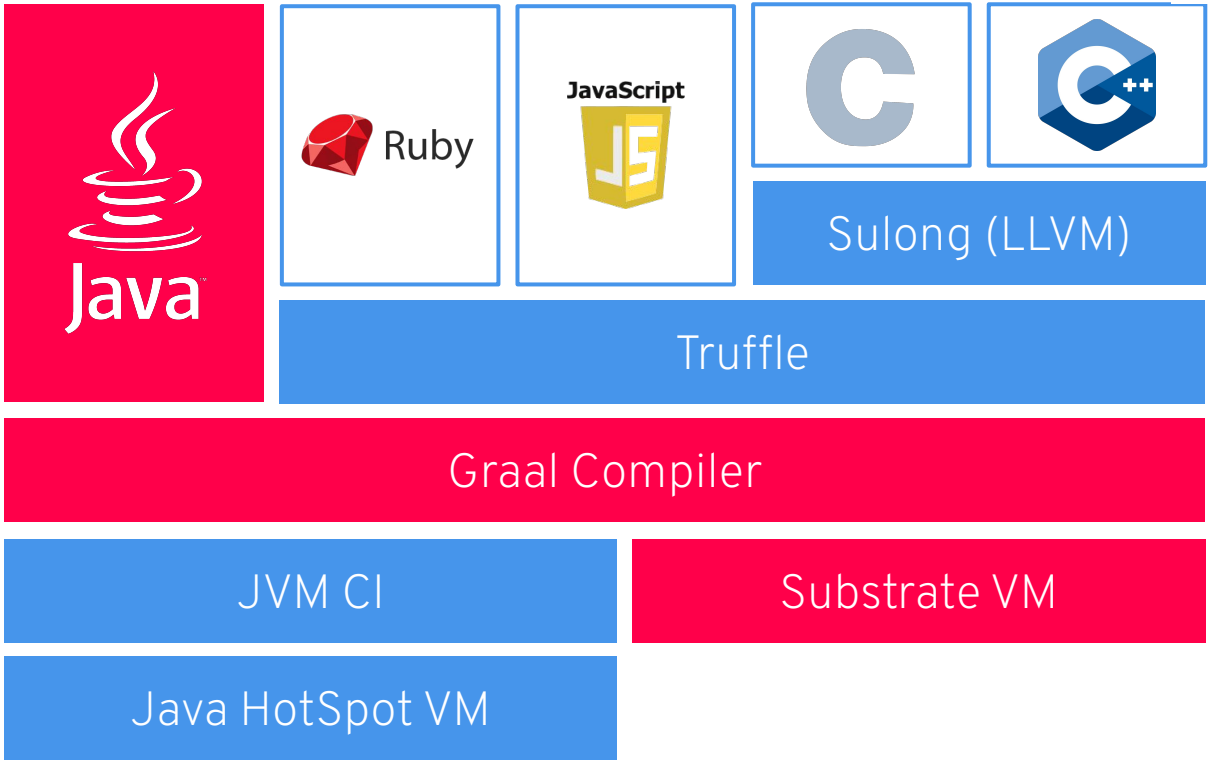
app.jar



frameworks

Runnable java app

native-app


Java
Ruby
JavaScript
C
C++
Sulong (LLVM)
Truffle
Graal Compiler
JVM CI
Substrate VM
Java HotSpot VM

The Dark Side

AOT is not That Simple

Not supported

- Dynamic classloading
- InvokeDynamic & Method handles
- Finalizers
- Security manager
- JVMTI, JMX, native VM Interfaces

OK with caveats in usage

- Reflection (requires configuration)
- Dynamic proxy (requires configuration)
- JNI (requires configuration)
- Static initializers (eager)
- Lambda, Threads (Okay)
- References (mostly supported)

https://github.com/oracle/graal/blob/master/substratevm/LIMITATIONS.md

How faster/smaller? Rule of Thumb

Hotspot optimized Quarkus App

⇒ ½ the RSS space
⇒ x5 boot speed

Native optimized Quarkus App

⇒ ⅕ the RSS space
⇒ x50 boot speed

When to use which VM with Quarkus

JIT - OpenJDK HotSpot

High memory density requirements
High request/s/MB
Fast startup time

Best raw performance (CPU)
Best garbage collectors
Higher heap size usage

Known monitoring tools
Compile Once, Run anywhere
Libraries that only work in standard JDK

AOT - GraalVM native image

Highest memory density requirements
Highest request/s/MB
    for low heap size usage
Faster startup time
    10s of ms for Serverless

More consistent CPU performance
    No JIT spikes
    Simpler GC

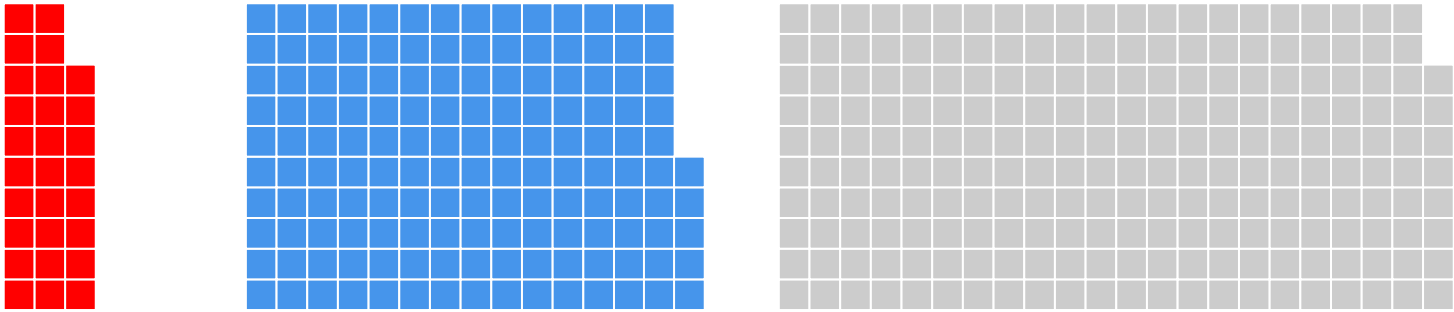
Show me some Code!

Recap:

Why QUARKUS?

Benefit No. 1: Supersonic Subatomic Java

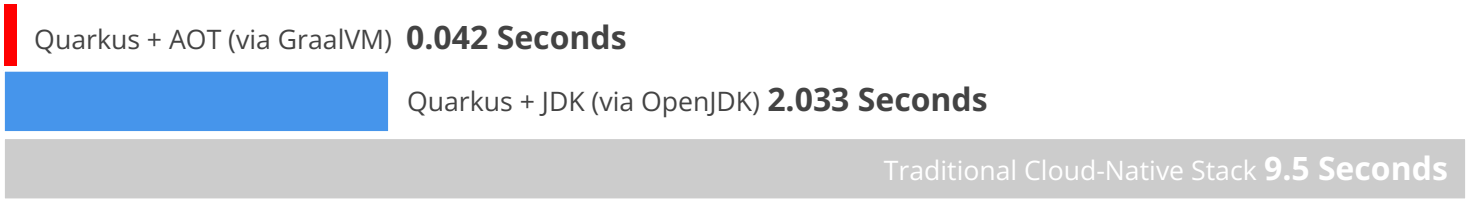
REST + CRUD



Quarkus + AOT (via GraalVM)
28 MB

Quarkus + JDK (via OpenJDK)
145 MB

Traditional Cloud-Native Stack
209 MB

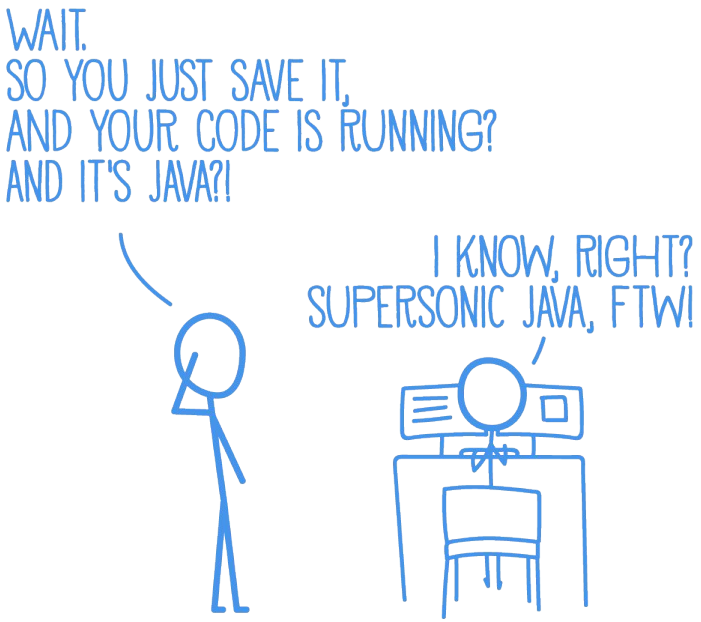


Time to first **response**

Benefit No. 2: Developer Joy

A cohesive platform for optimized developer joy:

- Based on standards, but not limited
- Unified configuration
- Zero config, live reload in the blink of an eye
- Streamlined code for the 80% common usages, flexible for the 20%
- No hassle native executable generation
- Live testing!



Benefit No. 3: Unifies Imperative and Reactive

```
@Inject
SayService say;

@GET
@Produces(MediaType.TEXT_PLAIN)
public String hello() {
      return say.hello();
}
```

```
@Channel("kafka") Multi<String> events;

@GET
@Produces(MediaType.SERVER_SENT_EVENTS)
public Multi<String> events() {
      return events;
}
```

- Combine both Reactive and imperative development in the same application
- Use the technology that fits your use-case
- Key for reactive systems based on event driven apps

Benefit No. 4: Best of Breed Frameworks & Standards

Quarkus provides a cohesive, fun to use, full-stack framework by leveraging a growing list of over fifty best-of-breed libraries that you love and use. All wired on a standard backbone.

Benefit No. 5: Continuous Innovation

Beyond support for popular and de-facto frameworks and standards, Quarkus is breaking ground with constant innovation in new APIs and implementations.

New Quarkus APIs & Impls

- Panache - Simplified Hibernate ORM
- Qute - New Templating Engine
- Funqy - Portable Functions API
- Mutiny - Reactive Programming Library
- RestEasy Reactive - Reactive JAX-RS variant
- Hibernate Reactive
- ...more to come

Want to learn more?

QUARKUS

https://quarkus.io

https://quarkusio.zulipchat.com

https://youtube.com/quarkusio

@quarkusio

If you like Quarkus, star it on GitHub!
https://github.com/quarkusio/quarkus