



QUARKUS

Supersonic. Subatomic. Java.

Scott Seighman
Solutions Architect / Red Hat
sseighma@redhat.com

Red Hat & Java



Red Hat Now Leading OpenJDK Efforts

- Red Hat will now serve as the steward of the OpenJDK 8 and OpenJDK 11 projects
- Red Hat will work with the community to enable continued innovation in Java
- Red Hat will ensure that key production versions of the Java platform are actively managed and will continue to be updated on a regular cadence
- Red Hat also leads the upstream development of Shenandoah, a high-performance garbage collector that is now part of OpenJDK 12



Red Hat's Java Leadership

- Red Hat is a member of the OpenJDK Vulnerability Group Governing Board
- Pioneered Java and Kubernetes (OpenShift)
- Developed and lead the 64-bit ARMv8 port
- Developed and lead the ultra-low pause time Shenandoah garbage collector project
- Lead and collaborate on many Java-based projects:
 - ActiveMQ, Eclipse CHE, Hibernate, Camel, Drools, WildFly, Strimzi ...
- Established the Quarkus project



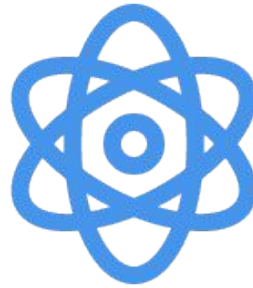
WHY QUARKUS?



An Open Source Stack to Write Java Apps



Cloud Native



Microservices



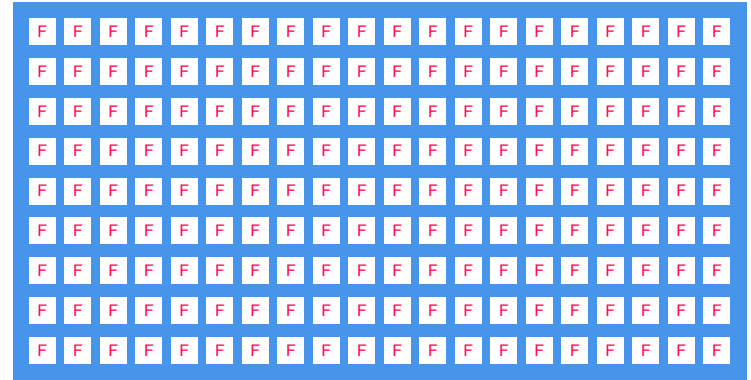
Serverless



From monolith to...



MONOLITH



1 monolith \approx 20 microservices \approx 200 functions



WHAT IS QUARKUS?

QUARK: elementary particle / **US:** hardest thing in computer science



Benefits of Quarkus

Four primary benefits:

- Developer Joy
- Supersonic. Subatomic. Java.
- Unifies Imperative & Reactive
- Best of Breed Frameworks & Standards



QUARKUS



Benefit No. 1: Developer Joy

A cohesive platform for optimized developer joy:

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

WAIT.
SO YOU JUST SAVE IT,
AND YOUR CODE IS RUNNING?
AND IT'S JAVA?!



I KNOW, RIGHT?
SUPERSONIC JAVA, FTW!



Benefit No. 1: Developer Joy

Improved Java Development Workflow

From this:

Write Code → Compile → Deploy → Refresh Browser → Repeat

To this:

Write Code → Refresh Browser → Repeat

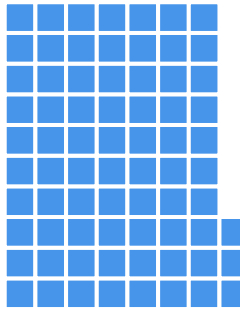


Benefit No. 2: Supersonic Subatomic Java

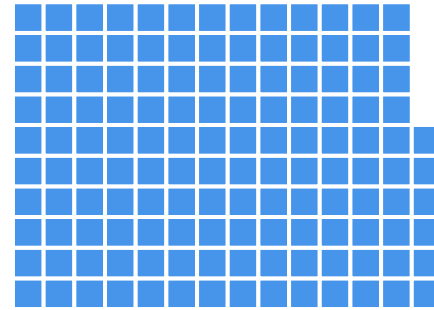
REST



Quarkus + Native (via GraalVM)
12 MB



Quarkus + JDK (via OpenJDK)
73 MB

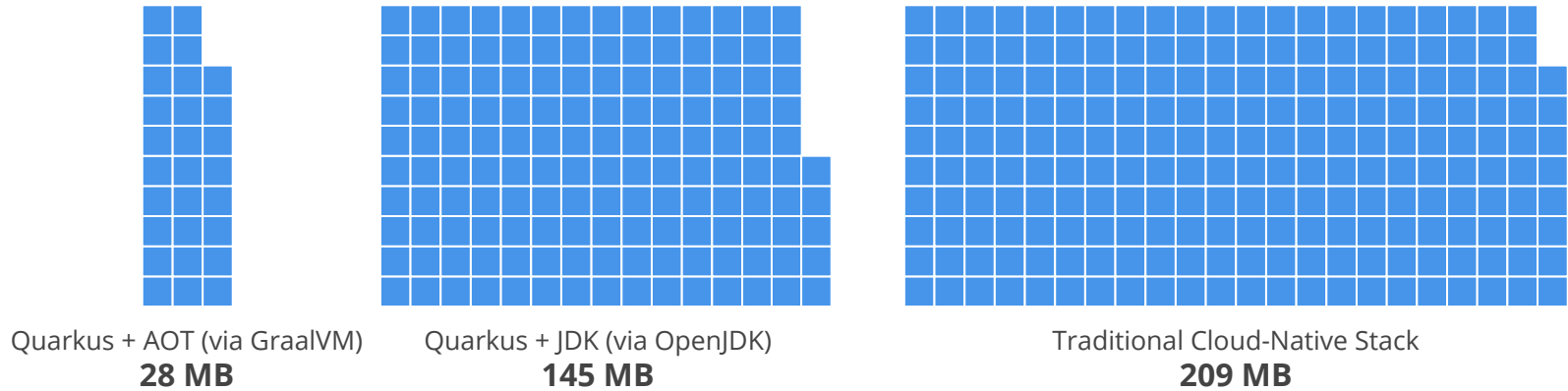


Traditional Cloud-Native Stack
136 MB

Memory RSS

Benefit No. 2: Supersonic Subatomic Java

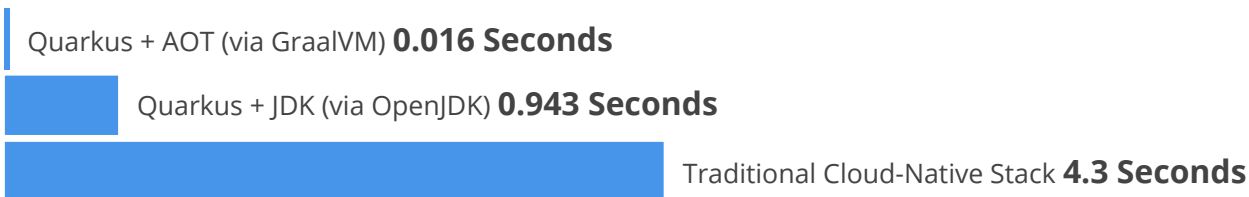
REST + CRUD



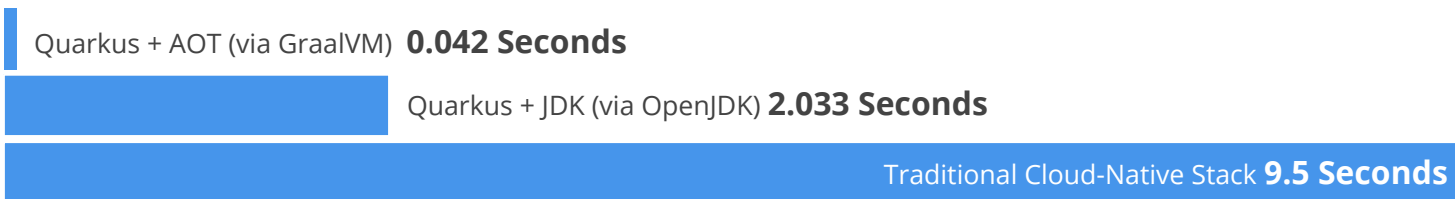
Memory RSS

Benefit No. 2: Supersonic Subatomic Java

REST

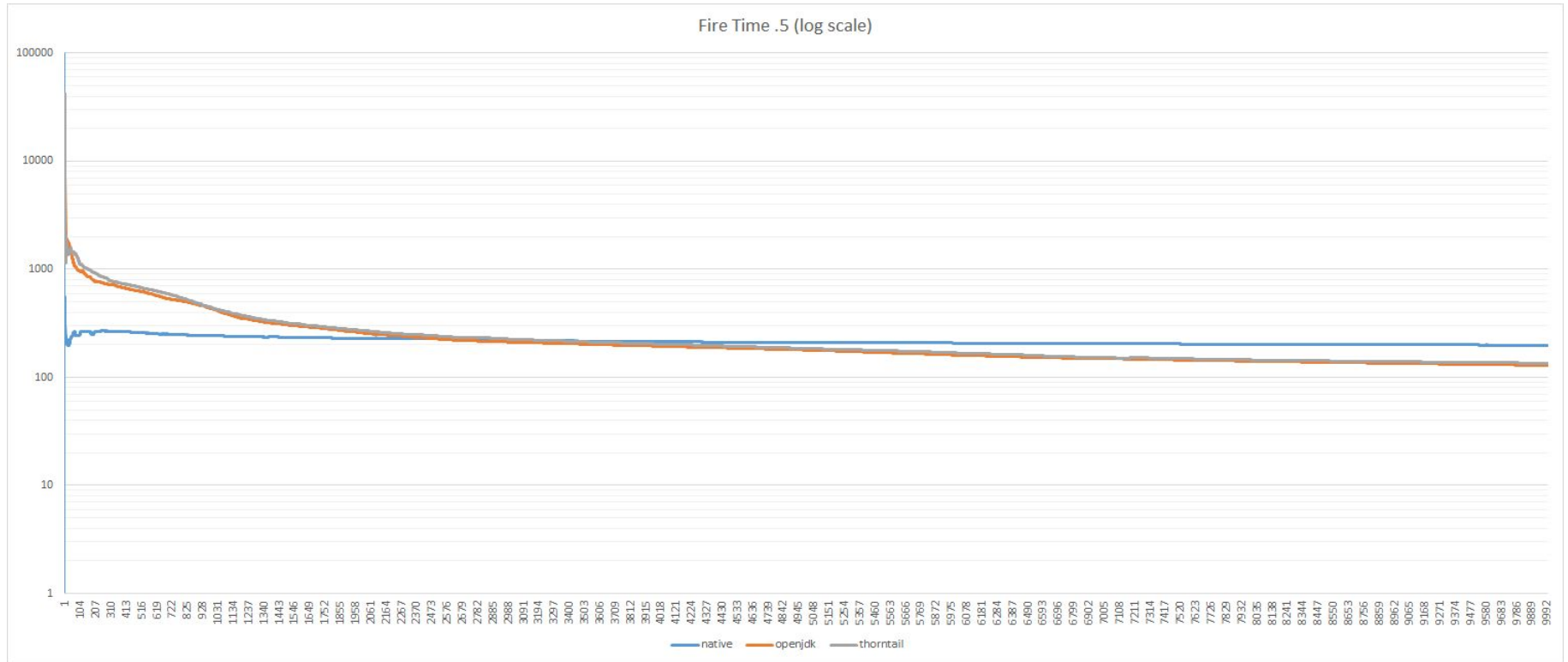


REST + CRUD



Time to first **response**

JVM vs Native Response Time



Benefit No. 3: Unifies Imperative and Reactive

```
@Inject
SayService say;

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

```
@Inject @Stream("kafka")
Publisher<String> reactiveSay;

@GET
@Produces(MediaType.SERVER_SENT_EVENTS)
public Publisher<String> stream() {
    return reactiveSay;
}
```

- 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



Eclipse Vert.x



Eclipse MicroProfile



Spring Compat



Hibernate



RESTEasy



Apache Camel



Kubernetes



OpenShift



Jaeger



Prometheus



Apache Kafka



Netty



HOW QUARKUS WORKS



Move Startup Time to Build Time

What does a framework do at startup time

- Parse config files
- Classpath & classes scanning
 - For annotations, getters or other metadata
- Build framework metamodel objects
- Prepare reflection and build proxies
- *Start and open IO, threads etc*



Framework Optimizations

- Moved everything possible to build phase
- Minimized runtime dependencies
- Maximize dead code elimination
- Introduced clear metadata contracts
- Spectrum of optimization levels
(all → some → no runtime reflection)

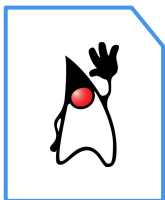
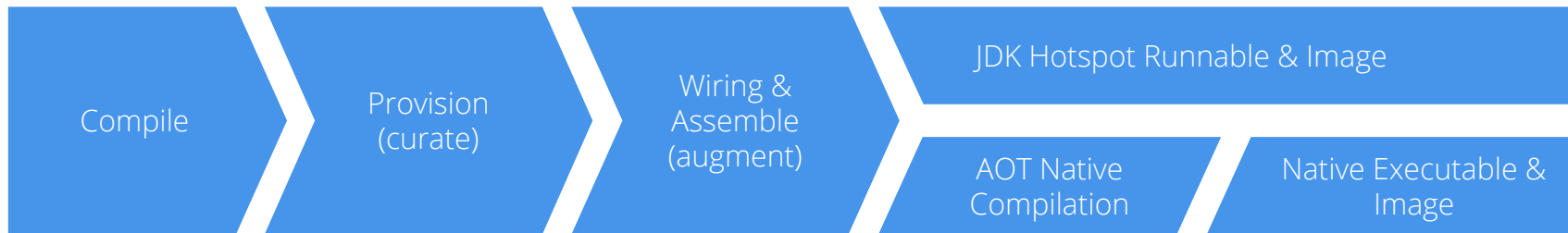


Build Time Benefits

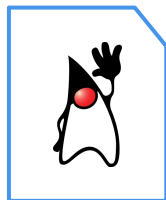
- Do the work once, not at each start
- All the bootstrap classes are no longer loaded
- Less time to start, less memory used
- Less or no reflection or dynamic proxy



An Ahead-of-Time, Build-Time, Runtime



app.jar



frameworks



Runnable java app



native-app

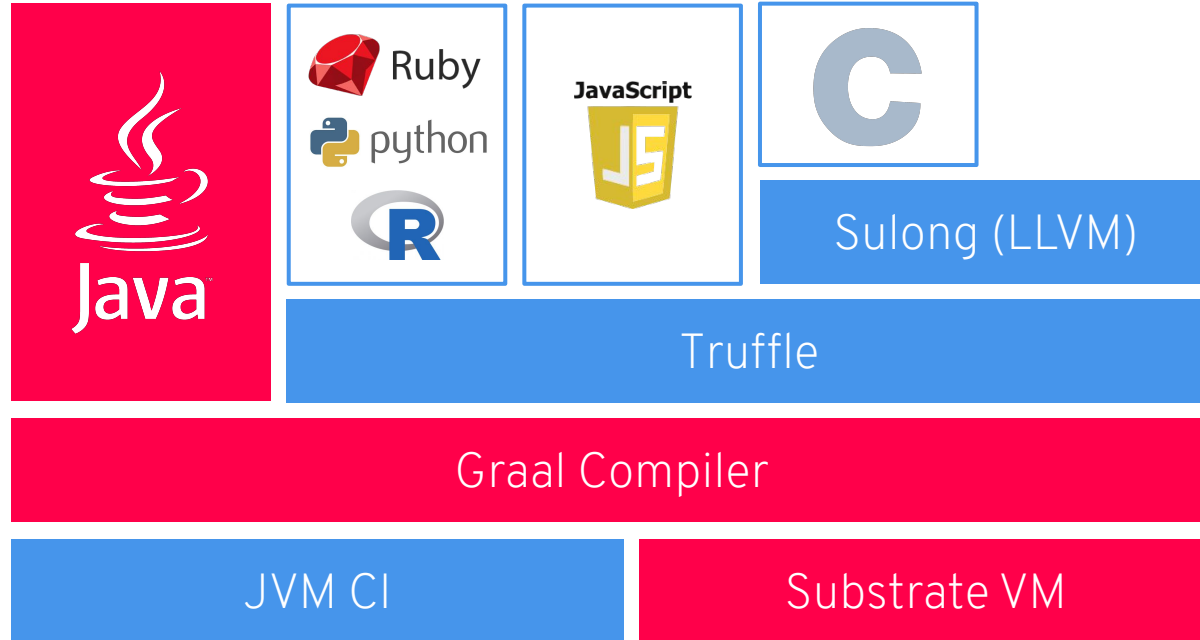


GraalVM Overview

- A polyglot VM with cross-language JIT supporting:
 - Java Bytecode and JVM languages
 - Interop with different languages
 - Dynamic languages through Truffle API
- Cross-language interop out of the box
 - Simple AST-based interpreter
 - JIT across language boundaries
- Support for native binary compilation (SubstrateVM)
 - Faster boot-up



GraalVM



The Dark Side

The "Good" Parts

Not supported

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

OK with caveats in usage

- **Reflection (manual list)**
- **Dynamic proxy (manual list)**
- JNI (manual list)
- **Static initializers (eager)**
- References (similar)



Quarkus Specific Benefits for GraalVM

100% of the ecosystem supported on GraalVM

Drives the gathering of metadata needed by GraalVM

- Based on framework knowledge
- Classes using reflection, resources, etc
- No need for agent + prerun, long JSON metadata or manual command lines

Minimize dependencies

Help dead code elimination



Building a Native Executable

Prerequisites

- JDK 8 installed with JAVA_HOME configured appropriately
- A working C development environment
- GraalVM installed and configured appropriately
 - Set GRAALVM_HOME
 - `gu install native-image`
- A working container runtime (Docker, podman)
- The code of the application developed

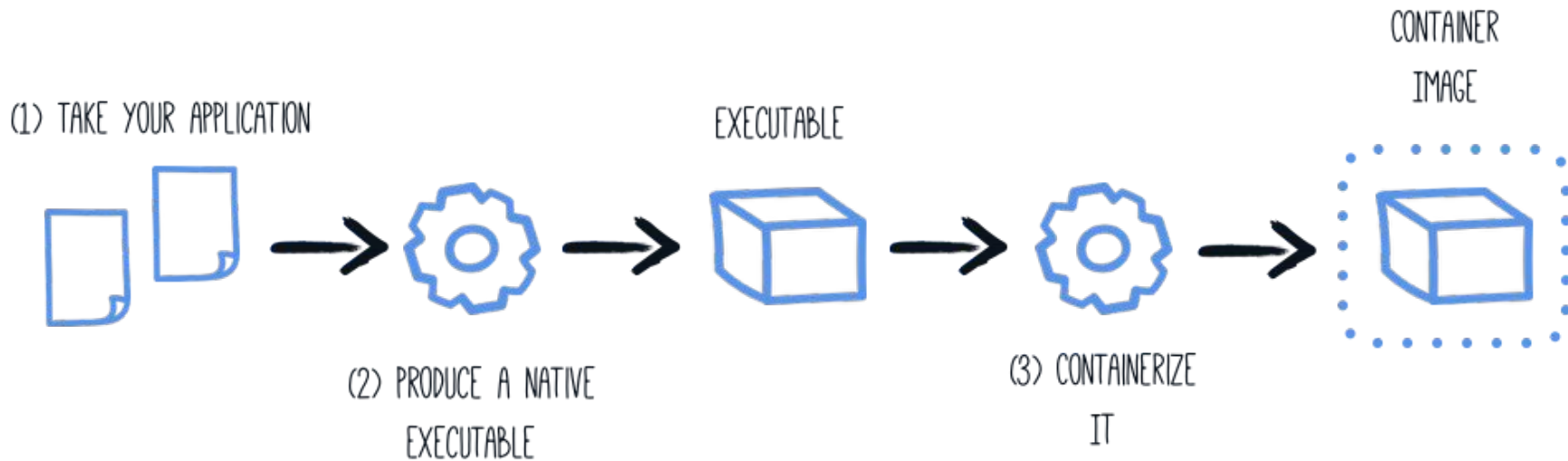
```
$ ./mvnw package -Pnative
```

```
$ ./mvnw package -Pnative
```

```
-Dquarkus.native.container-runtime=podman
```



Building a Native Executable



End Results of Native Executable

- Quarkus allows generation of operating system specific (native) executable from your Java code
- Trade-offs: lack of build-once-run-anywhere and hot reload capabilities VS. lower on-disk footprint and quicker startup time
- Perfect fit for serverless/event-driven environments
- Statically compiled executables benefit from closed-world optimizations:
 - Fine-grained dead-code elimination, where only the portions of frameworks (including the JDK itself) actually in use by the service are included in the resulting image



DEMO

Next-Gen Tools: Buildah, Podman, Skopeo

Providing stability, flexibility and performance with containers and images

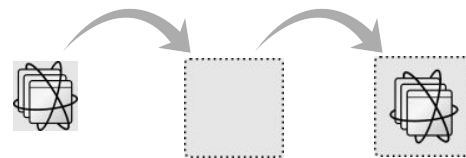
Container-tools:

OCI tooling to create, run, and manage, Linux Containers with an enterprise life cycle

- Conform to the OCI image and runtime specifications
- Daemon-less, OS-native container tooling
- Separation of concerns



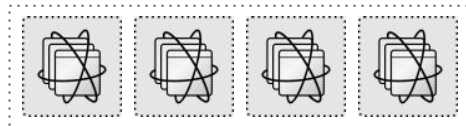
buildah



Build OCI/docker Images



podman



RHEL

Run, manage, debug containers



skopeo



Inspect, copy, & sign Images





cri-o

Project Charter and Goals:

- A lightweight, OCI-compliant container runtime designed for Kubernetes
- Runs any OCI / Docker container from any OCI / Docker registry
- Focus on stability and life cycle *with* the OpenShift & Kubernetes
- Improve container security & performance at scale



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 works in standard JDK

AOT - GraalVM native image

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



Quarkus Extensions

RESTEasy

Netty

Hibernate

Spring Compat.

MP OpenAPI

MP JWT

Eclipse Vert.X

Agroal (conn pool)

Narayana JTA

MP Reactive
Messaging

Apache Camel

...

Quarkus Core

Jandex

Gizmo

Graal SDK

Arc (DI)

JDK JIT - HotSpot

AOTC - GraalVM Native Image

MORE QUARKUS



Getting Started

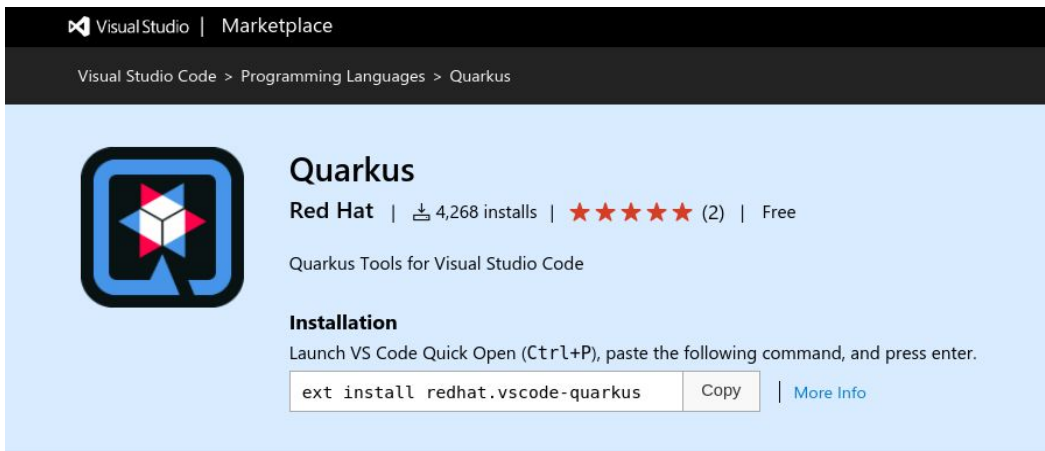
- An IDE (VS Code Extension)
- JDK 8 or 11+ installed with JAVA_HOME configured appropriately
- Apache Maven 3.5.3+
- Optionally: GraalVM



QUARKUS




Quarkus VS Code Extension



VisualStudio | Marketplace

Visual Studio Code > Programming Languages > Quarkus

 **Quarkus**

Red Hat | 📄 4,268 installs | ★★★★★ (2) | Free

Quarkus Tools for Visual Studio Code

Installation

Launch VS Code Quick Open (Ctrl+P), paste the following command, and press enter.

```
ext install redhat.vscode-quarkus
```

[Copy](#) | [More Info](#)

The Quarkus extension is dependent on a couple of Java extensions for VS Code, so it is recommended that you have the [Java Extension Pack](#) installed



Quickly Bootstrap Quarkus Applications

QUARKUS GET STARTED GUIDES COMMUNITY BLOG **START CODING**

Application Info

Group:

Artifact:

Build Tool:

CONFIGURE MORE OPTIONS

Generate your application (alt + ⌘)

Extensions

Selected Extensions

Core

- Logging GELF PREVIEW Log using the Graylog Extended Log Format and centralize your...
- Logging JSON PREVIEW Add JSON formatter for console logging
- Logging Sentry PREVIEW Use Sentry, a self-hosted or cloud-based error monitoring solut...
- YAML Configuration Use YAML to configure your Quarkus application

Web

- RESTEasy JAX-RS INCLUDED REST framework implementing JAX-RS and more
- RESTEasy JSON-B JSON-B serialization support for RESTEasy
- RESTEasy Jackson Jackson serialization support for RESTEasy



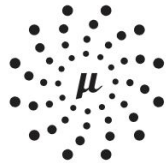
Others ...



Google Cloud Functions



**Azure
Functions**



M I C R O N A U T™



helidon.io





Red Hat Runtimes

Offering lightweight runtimes and frameworks for highly-distributed **cloud native** architectures such as microservices or serverless, with distributed in-memory caching for fast data access, single sign-on for authentication and authorization, and durable messaging for reliable data transfer between existing and new applications.

LAUNCH SERVICE	
Red Hat JBoss Enterprise Application Platform	Red Hat Data Grid
OpenJDK™	Red Hat AMQ
RED HAT™ SSO	Red Hat Application Migration Toolkit

- Best-of-breed runtimes, frameworks and languages
- OpenShift & Kubernetes Services native integration
- Modernization and optimization initiatives
- Established middleware technologies (EAP)
- In-memory data grid
- Standards-based enterprise messaging
- SSO authentication



Guided Choice Of Runtimes & Languages

ENTERPRISE JAVA



SPRING APPS



JAVA MICROSERVICES



JAVASCRIPT FLEXIBILITY



REACTIVE SYSTEMS



TOMCAT SIMPLICITY



Runtimes Summary

- Support from **one vendor** for all runtimes
 - **Production bug fix resolution for critical issues**
- **Standardized** set of best-of-breed containerized runtimes for microservices
- **Polyglot, poly-architecture** for developer choice
- **Pre-integration** with OpenShift, Middleware **simplifies** architecture
- Supports **existing** and net-new apps
- The future is container native Java
- Getting started experience
- Great value (subscription)



Resources

- Quarkus
 - <https://quarkus.io/>
- Quarkus Forum
 - <https://quarkusio.zulipchat.com/login/>
- Quarkus Cheat Sheet
 - <https://lordofthejars.github.io/quarkus-cheat-sheet/#quarkuscheatsheet>
- GraalVM
 - <https://www.graalvm.org/>



Resources

- Quarkus Roadmap
 - <https://github.com/orgs/quarkusio/projects/5>



Summary: Quarkus Benefits

Developer Joy

Supersonic Subatomic Java

Unifies
Imperative and Reactive




Best of breed
Libraries and Standards



Thank you



QUARKUS

-  <https://quarkus.io>
-  <https://quarkusio.zulipchat.com>
-  [@quarkusio](https://twitter.com/quarkusio)

Scott Seighman
Solutions Architect / Red Hat
sseighma@redhat.com

