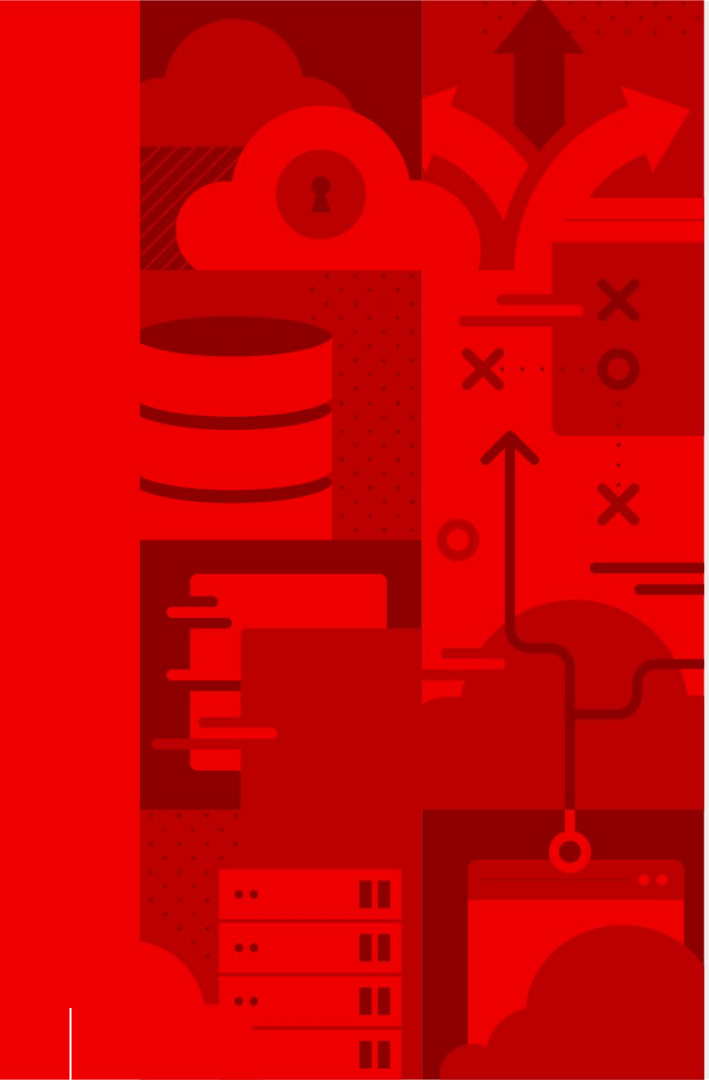


App Services

linkedin.com/company/red-hat

facebook.com/redhatinc

youtube.com/user/RedHatVideos

twitter.com/RedHat

Red Hat

Self introduction

Name: Wanja Pernath

Email: wpernath@redhat.com

Base: Germany (very close to the Alps)

Role: EMEA Technical Partner Development Manager - OpenShift and MW

Experience: Years of Consulting, Training, PreSales at Red Hat and before

Twitter: https://twitter.com/wpernath

LinkedIn:https://www.linkedin.com/in/wanjapernath/

First book just published

Getting GitOps

A technical blueprint for developing with Kubernetes and OpenShift based on a REST microservice example written with Quarkus

Technologies discussed:

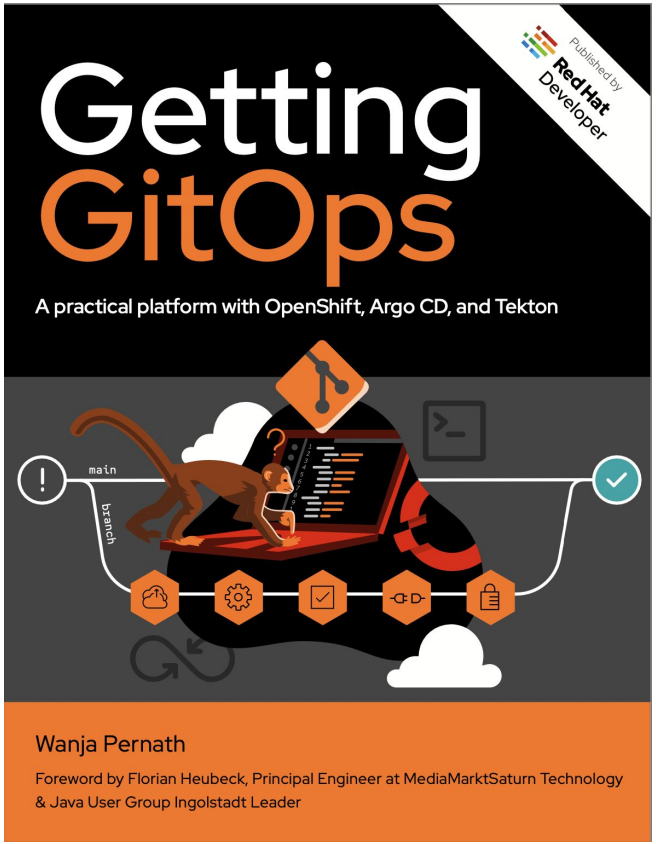
Quarkus, Helm Charts, Kustomize, Tekton Pipelines, Kubernetes Operators, OpenShift Templates, ArgoCD, CI/CD, GitOps….

Download for free at:

https://developers.redhat.com/e-books/getting-gitops-practical-platform-openshift-argo-cd-and-tekton

Interview with full GitOps Demo:

https://www.youtube.com/watch?v=znMfVqAIRzY&ab_channel=OpenShift



Quarkus Technical Value

"Historical" Enterprise Java Stack

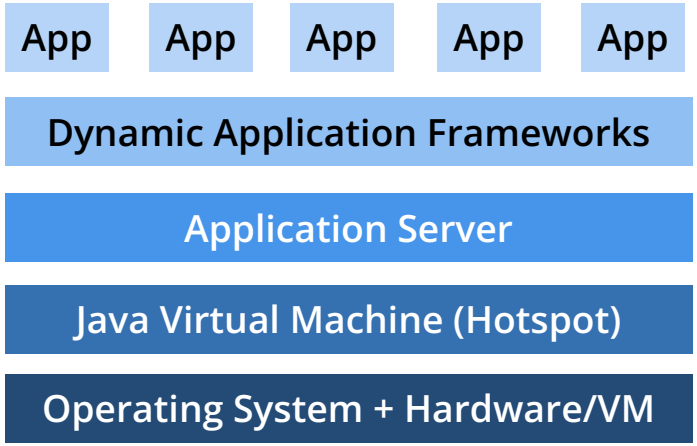
Architecture: Monoliths

Deployment: multi-app, appserver

App Lifecycle: Months

Memory: 1GB+ RAM

Startup Time: 10s of sec



"Modern" Enterprise Java Stack

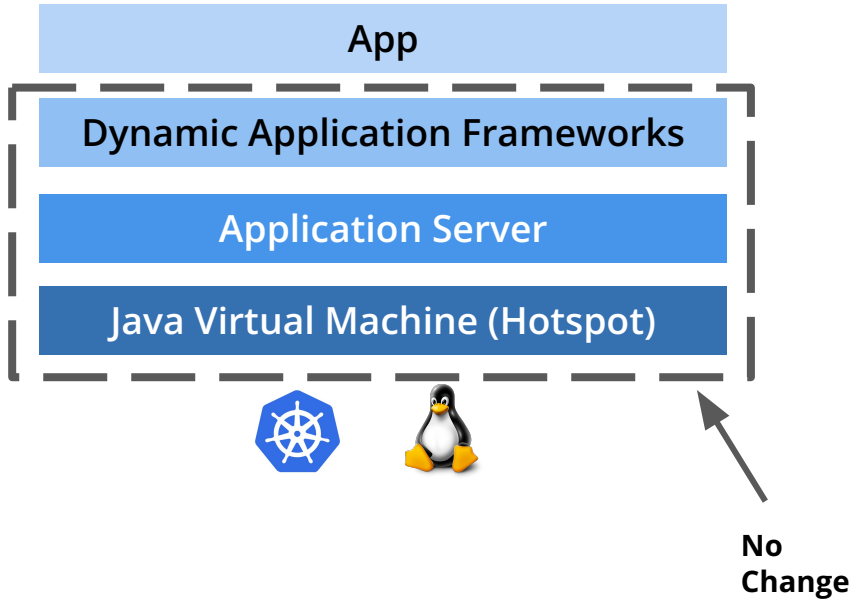
Architecture: Microservices

Deployment: Single App

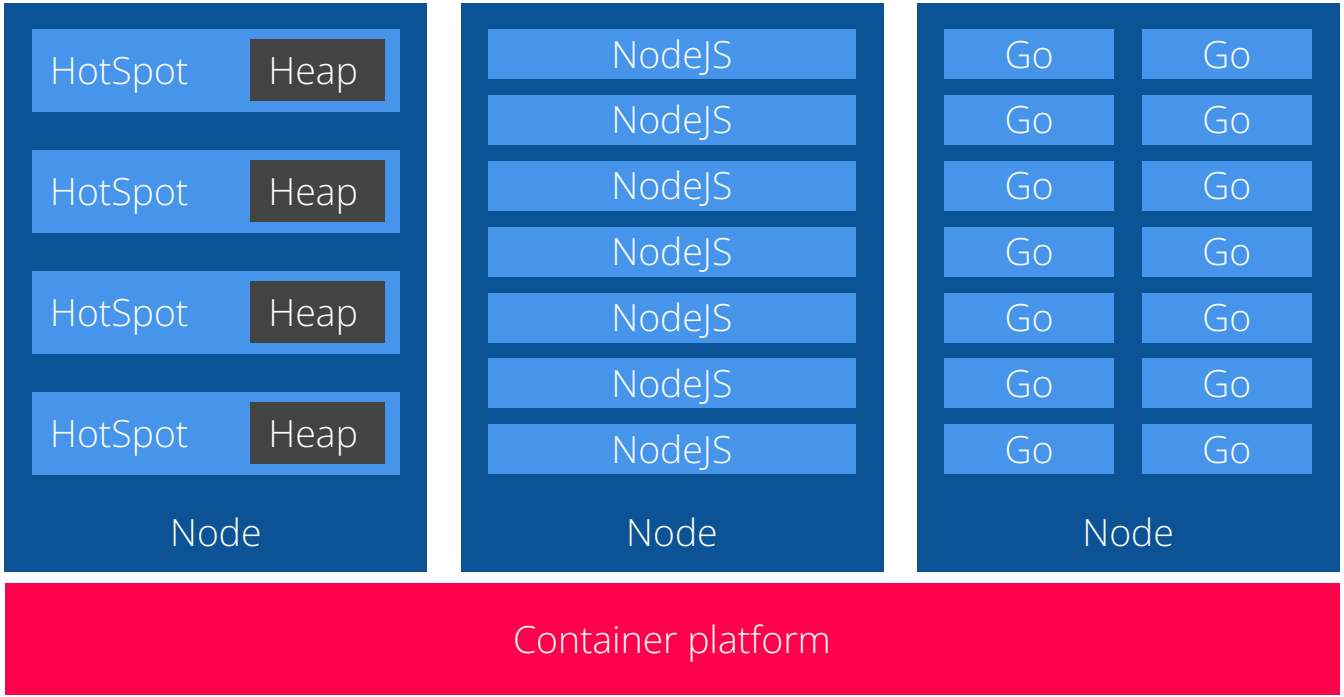
App Lifecycle: Days

Memory: 100MBs+ RAM

Startup Time: Seconds



Hidden Truth About Java + Containers



Red Hat

THERE IS A NEED FOR A NEW JAVA STACK FOR CLOUD-NATIVE AND SERVERLESS

Experts from cloud-native Java OS projects

Eclipse Vert.x | Hibernate | RESTEasy | Eclipse MicroProfile | WildFly | Undertow | OpenJDK

QUARKUS

Red Hat

Differentiators

Container First

Tailors your app for HotSpot & GraalVM

Fast boot time and low RSS memory

Serverless fit

Developer Joy

Live coding

Unified configuration

Unifies Imperative & Reactive

Combines blocking and non-blocking

Built-in event bus

Best of Breed Libraries & Standards

90+ extensions

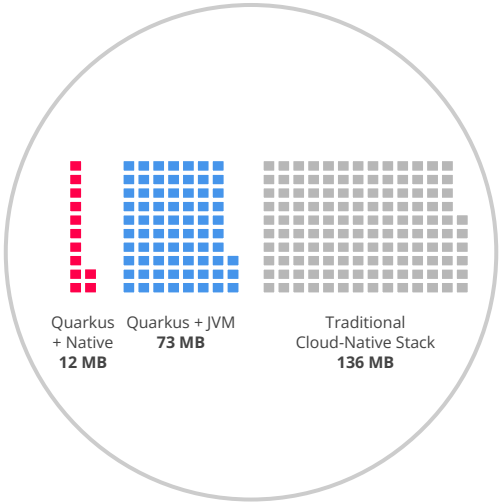
“Powered by Quarkus” applications

Red Hat

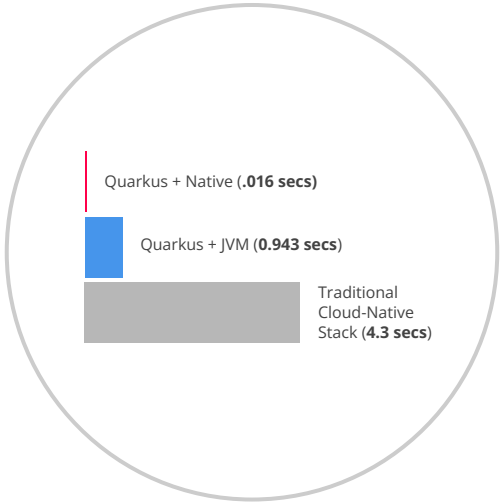
Benefit No. 1: Container First

*"We went from **1-min** startup times to **400 milliseconds**"*

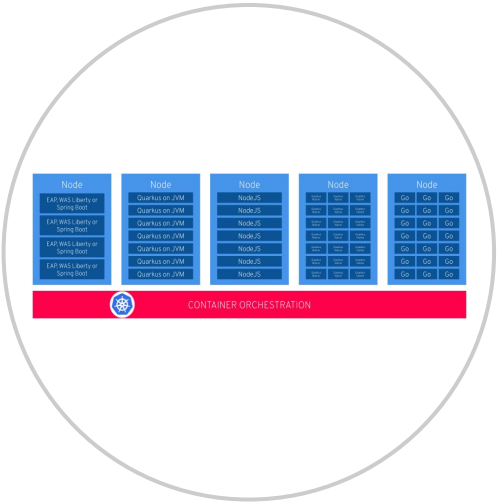
Reduced Memory Footprint



Fast Startup Time



Smaller Disk Footprint



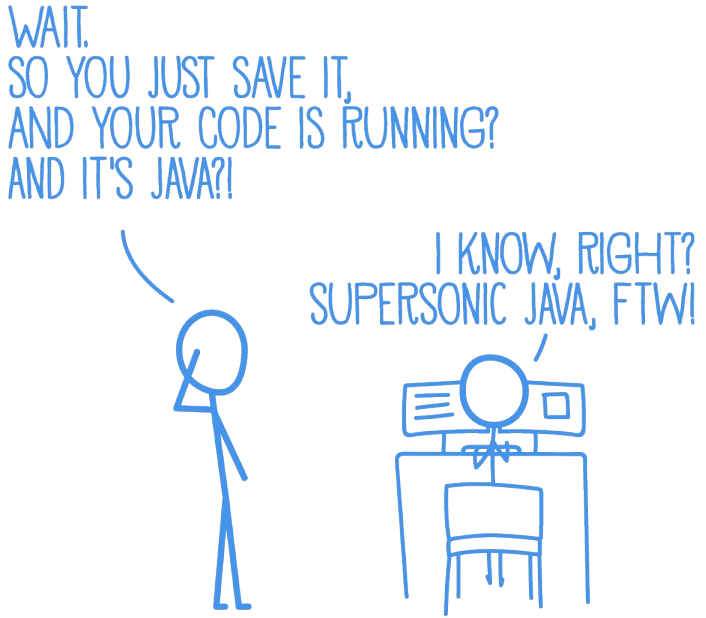
Red Hat

Benefit No. 2: Developer Joy

*"Our developers used to wait **2 to 3 mins** to see their changes. **Live coding** does away with this."*

A cohesive platform for optimized developer joy:

- Based on standards and more
- Unified configuration
- Live coding
- Streamlined code for the 80% common usages, flexible for the 20%
- No hassle native executable generation



Red Hat

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
- Inject the EventBus or the Vertx context
- 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

"When you adopt Quarkus, you will be productive from day one since you don't need to learn new technologies."

Eclipse Vert.x

Hibernate

RESTEasy

Apache Camel

Eclipse MicroProfile

Netty

Kubernetes

OpenShift

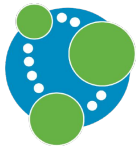
Jaeger

Prometheus

Apache Kafka

Infinispan

Flyway



Neo4j

MongoDB

MQTT

KeyCloak

Apache Tika

Red Hat

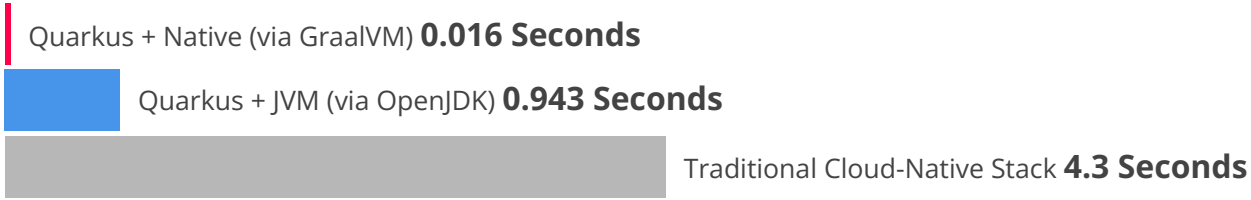
Supersonic, Subatomic

Fast.
Blazing fast to start.
Millisecond fast!

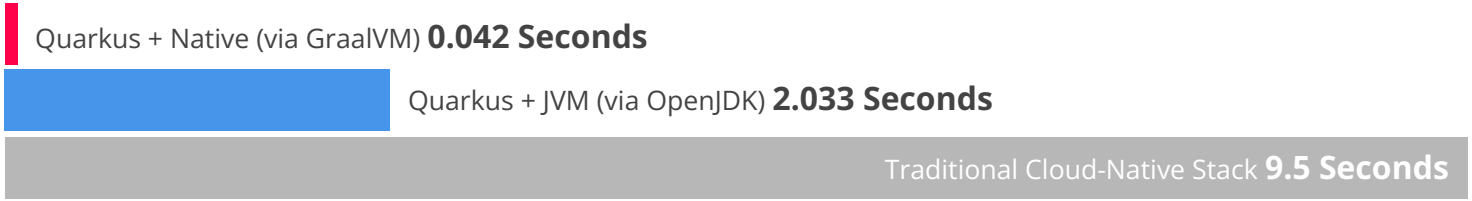
Red Hat

Supersonic, Subatomic Java

REST

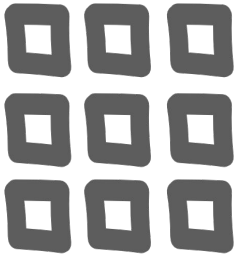


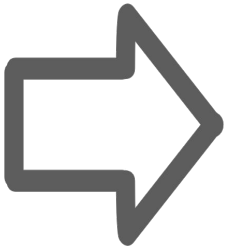
REST + CRUD



Time to first response

Red Hat





Supersonic, Subatomic

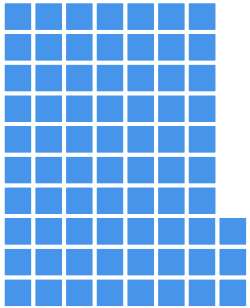
Improve memory consumption.
Increase deployment density.

Red Hat

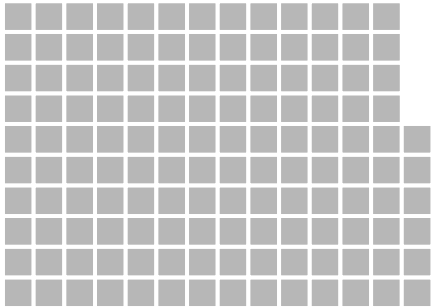
Supersonic, Subatomic Java

REST*

Quarkus + Native
(via GraalVM)
12 MB



Quarkus + JVM
(via OpenJDK)
73 MB



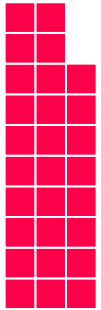
Traditional
Cloud-Native Stack
136 MB

*Memory (RSS) in Megabytes, tested on a single-core machine

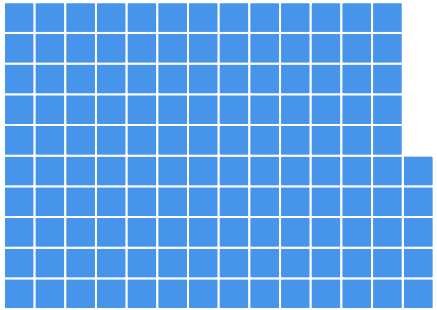
Red Hat

Supersonic, Subatomic Java

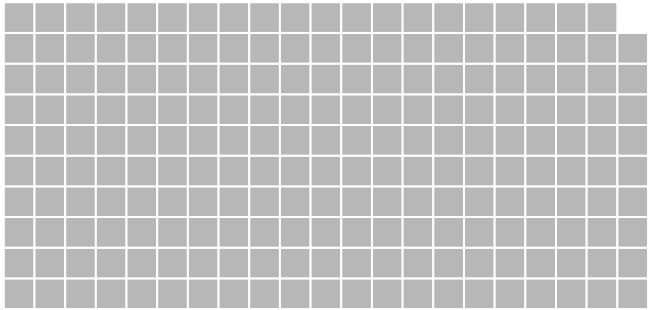
REST + CRUD*



Quarkus + Native
(via GraalVM)
28 MB



Quarkus + JVM
(via OpenJDK)
145 MB

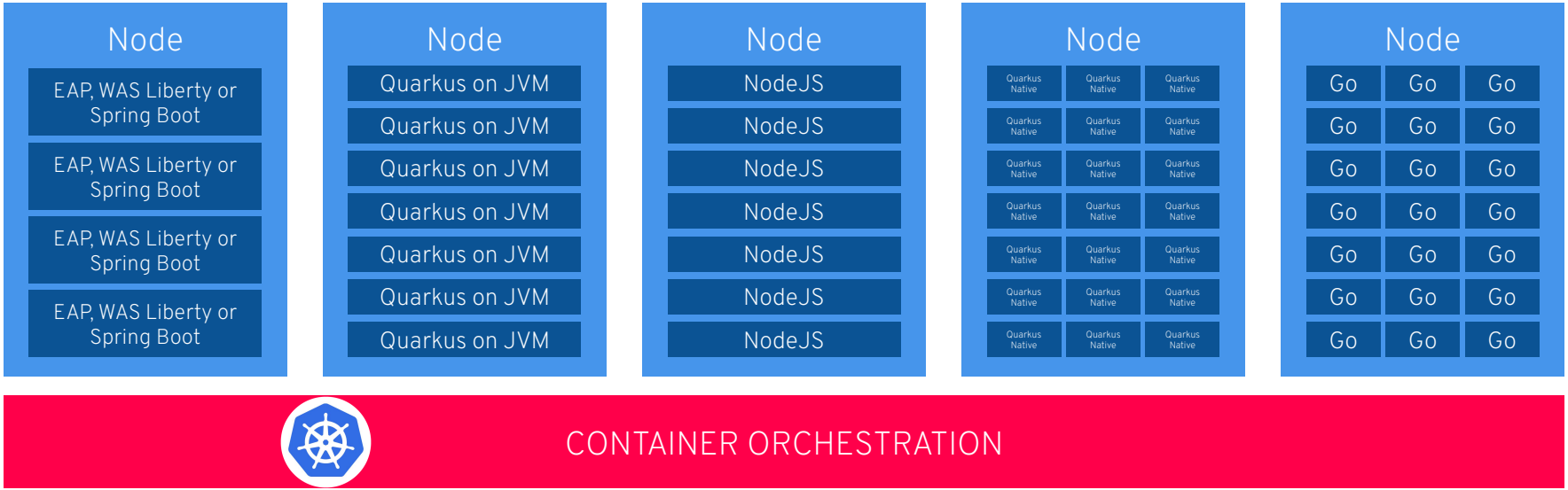


Traditional
Cloud-Native Stack
209 MB

*Memory (RSS) in Megabytes, tested on a single-core machine

Red Hat

Cloud Native Java Stack + Containers



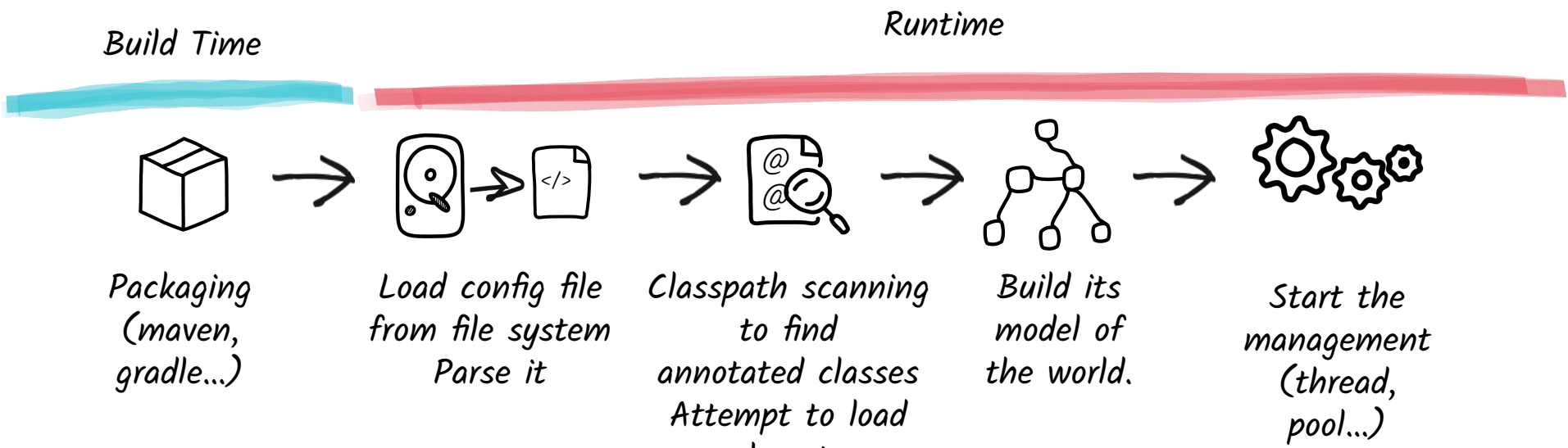
*"We could run **3 times** denser deployments without sacrificing **availability** and **response times** of services"*

https://developers.redhat.com/blog/2017/03/14/java-inside-docker/

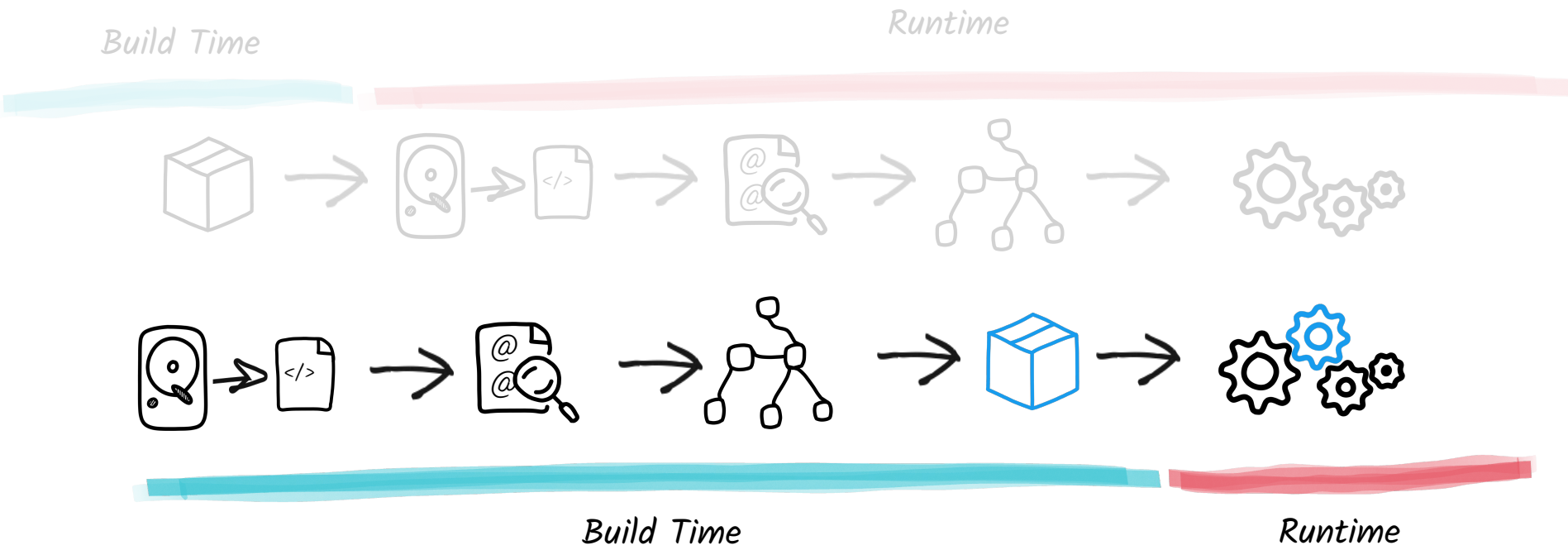
Red Hat

HOW DOES QUARKUS WORK?

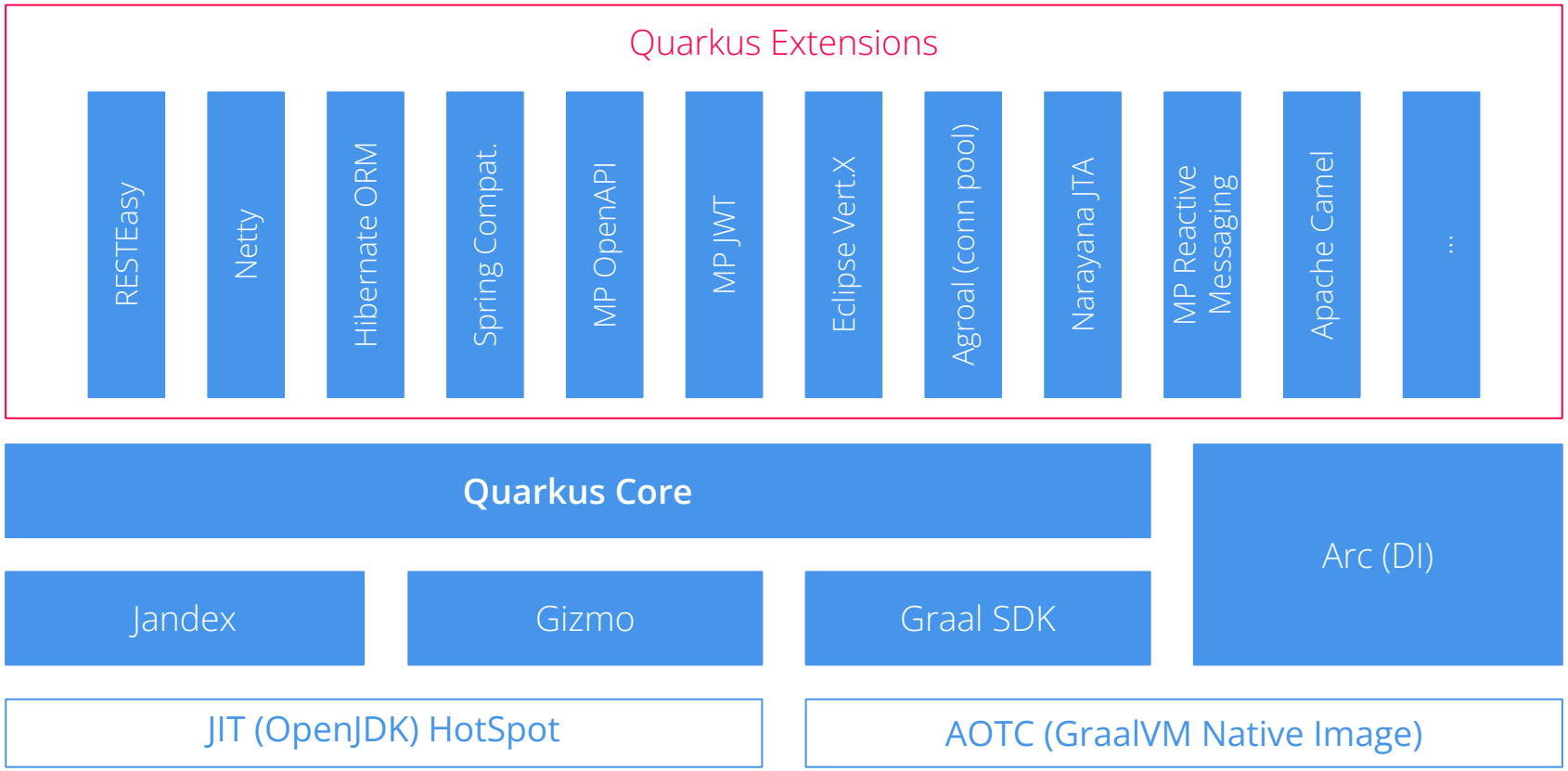
How Does a Framework Start?



The Quarkus Way



An ahead-of-time, build-time, runtime


Quarkus Extensions
RESTEasy
Netty
Hibernate ORM
Spring Compat.
MP OpenAPI
MP JWT
Eclipse Vert.X
Agroal (conn pool)
Narayana JTA
MP Reactive Messaging
Apache Camel
...
Quarkus Core
Arc (DI)
Jandex
Gizmo
Graal SDK
JIT (OpenJDK) HotSpot
AOTC (GraalVM Native Image)

Red Hat

The Right VM For the Right Deployment

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

*Currently in Tech Preview

Red Hat

Quarkus Tools - Build


```
mvn io.quarkus:quarkus-maven-plugin:1.3.2.Final-redhat-00001:create \
    -DprojectGroupId=org.acme \
    -DprojectArtifactId=getting-started \
    -DplatformGroupId=com.redhat.quarkus \
    -DplatformVersion=1.3.2.Final-redhat-00001 \
    -DclassName="org.acme.quickstart.GreetingResource" \
    -Dpath="/hello"
cd getting-started
```

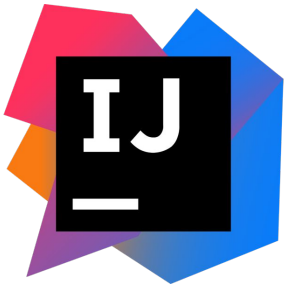
*community supported

Red Hat

Quarkus Tools - IDE

VSCode

Eclipse



IntelliJ

che.openshift.io

DEMO

THANK YOU