Red Hat Summit

Connect

Quarkus

Overview

Waeil Eldoamiry Principal Solutions Architect

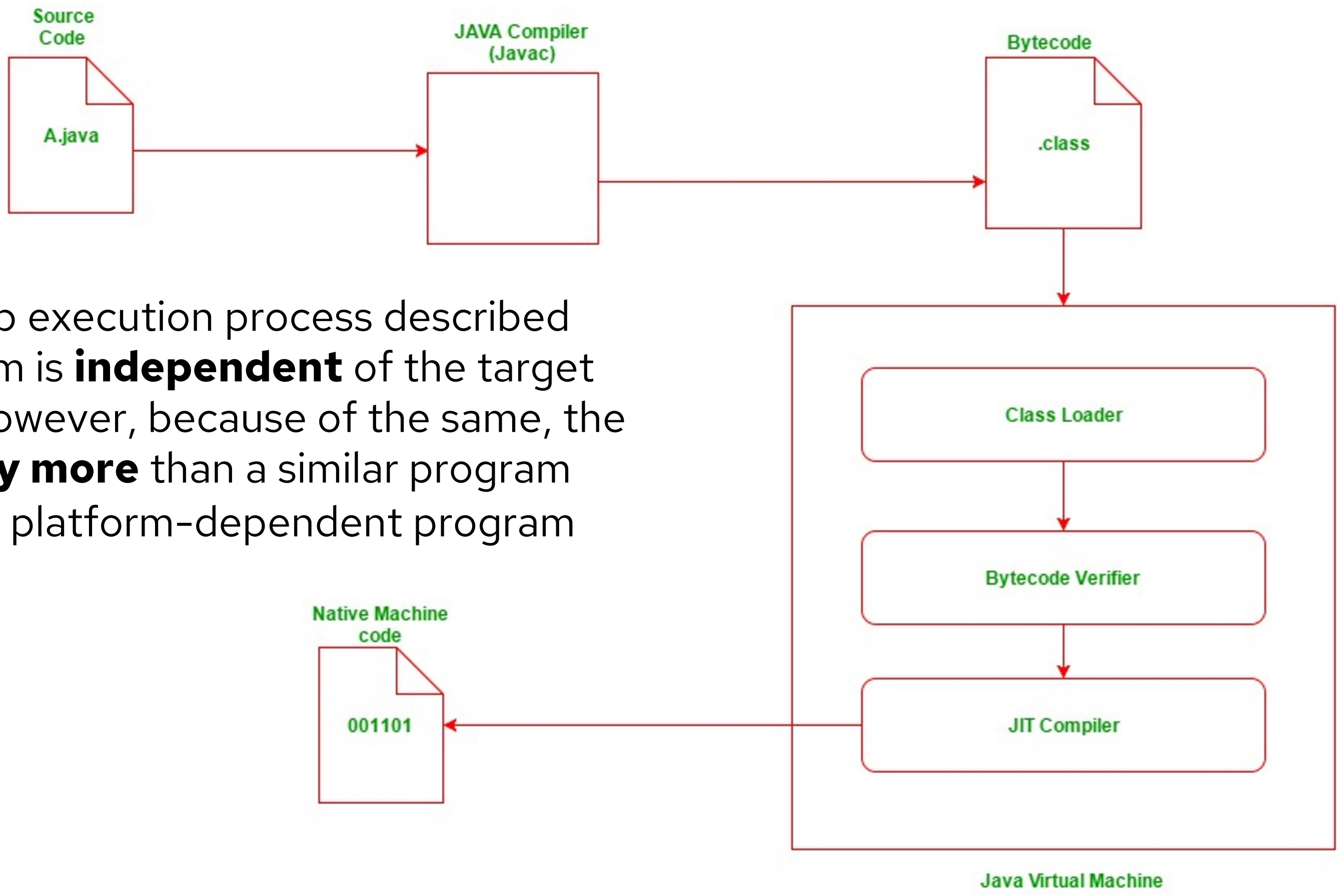
Red Hat | intel.

Is Java dead!?

But, Java is slow?

But, Java is slow?

Multi-step runtime process



Due to the multi-step execution process described above, a java program is **independent** of the target operating system. However, because of the same, the execution time is **way more** than a similar program written in a compiled platform-dependent program

Multi Step runtime process

ClassLoader loads the main class and all dependencies.
Remember Dependency resolution happens at the byte code level

What you might not know!

Dependency Injection

Java developers heavily rely on DI applying patterns like dynamic proxies and IoC.

What does this mean?
Developers declaratively specify what should happen and the implementation makes sure it does.

- Dependency resolution happens at **runtime** that results in **heavy lifting** and **long start up** time.
- Is there a chance that dependency resolution **fails**? **Yes**, what is the impact? Application will not start "very famous class not found exception"
- JEE has CDI specs - Context and dependency injection - and **Weld** provides the reference implementation and It is integrated in most Application servers if not all.

Weld

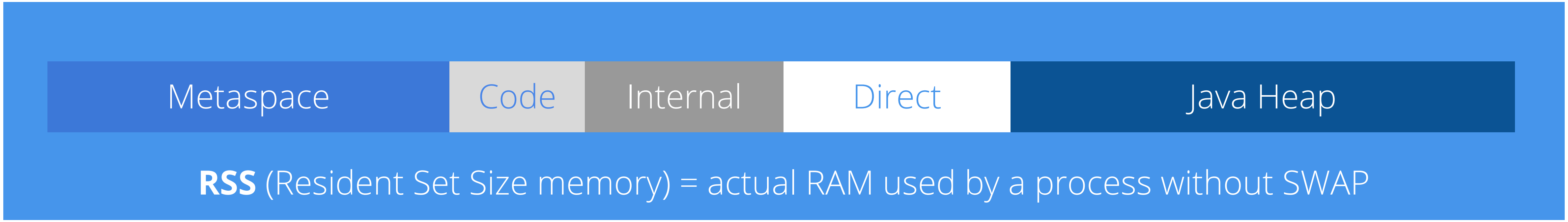
CDI reference implementation

- All beans are discovered at startup
- Proxies are dynamically generated
- Extensive use of reflection
- Expensive to start

http://cdi-spec.org/
http://weld.cdi-spec.org/

The hidden truth about Java

- Startup overhead
  - # of classes, bytecode, JIT
- Memory overhead
  - # of classes, metadata, compilation

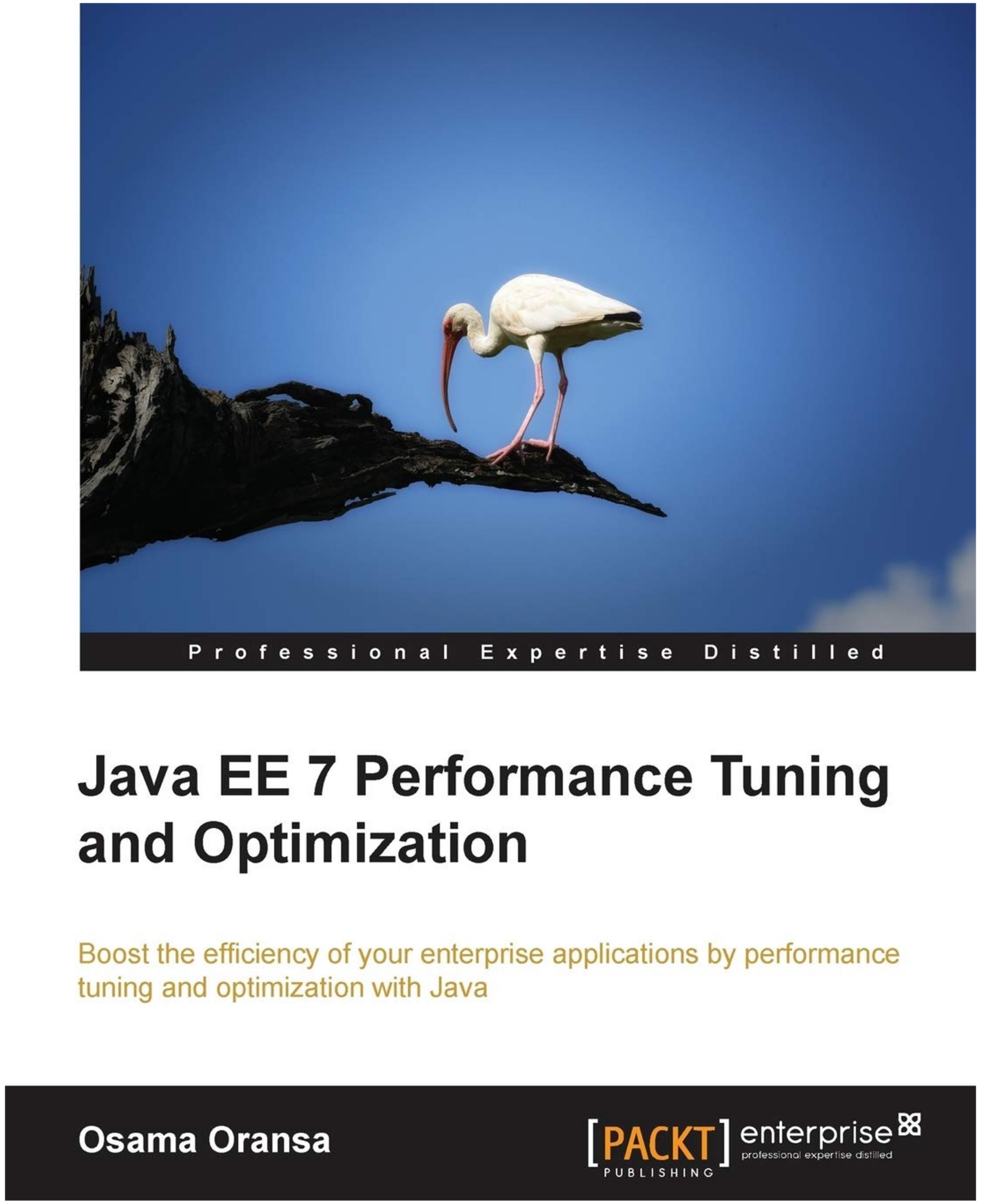


RSS (Resident Set Size memory) = actual RAM used by a process without SWAP

Classes are indexed, Metadata about annotation is created, injections and dependency resolution happens. This all is waste of memory and time.

Resident **S**et **S**ize is the amount of physical memory currently allocated and used by a process (without swapped out pages)

Experts did a great job addressing Java performance. But no matter what **experts** do, Java is still slow :)



Professional Expertise Distilled

Java EE 7 Performance Tuning and Optimization

Boost the efficiency of your enterprise applications by performance tuning and optimization with Java

Osama Oransa

[PACKT] PUBLISHING enterprise professional expertise distilled

Cloud and Java don’t mix

They mix, and produced !!

QUARKUS

Whats is Quarkus?

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

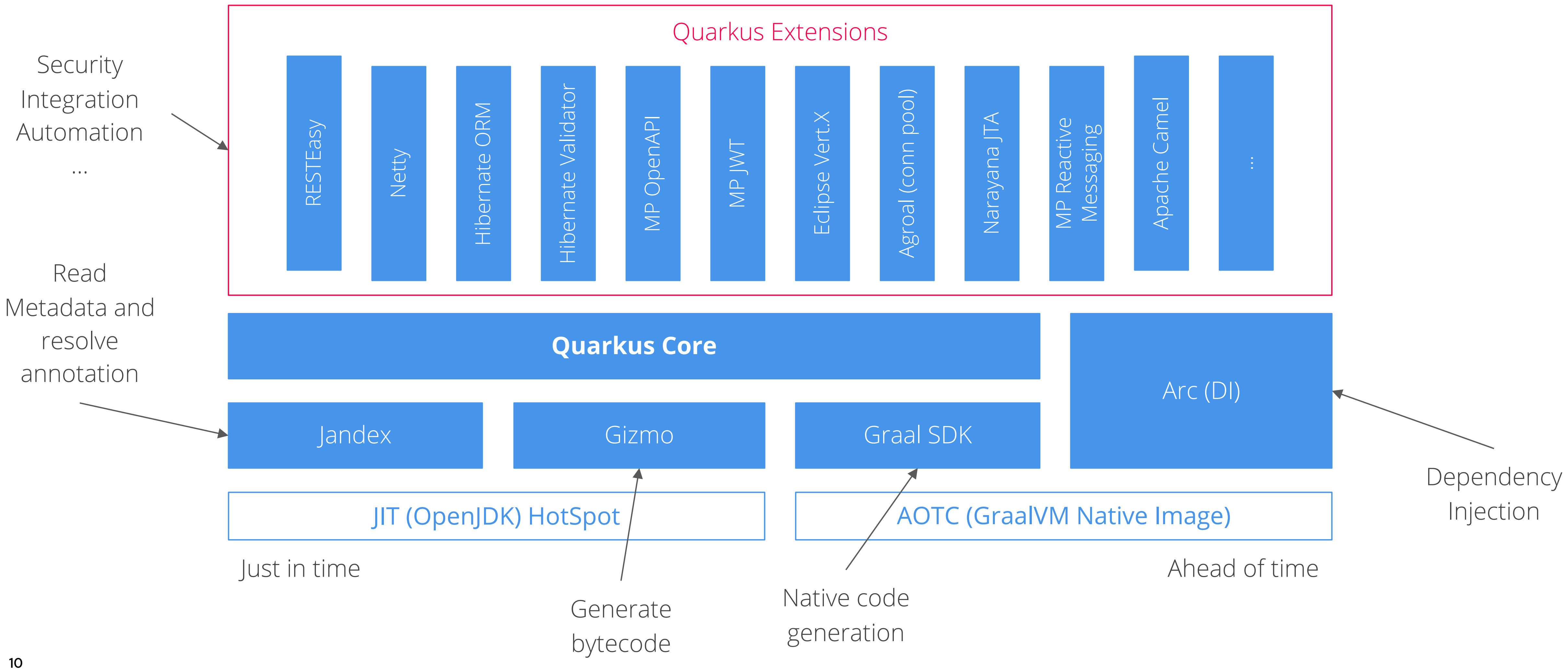
Quarkus is a K8 native **java stack**.

Subatomic because It is very small and lightweight

Supersonic because it is fast with unbeatable ignition time

Supported on **OpenJDK** and **GraalVM**

Quarkus Architecture



How does It work?

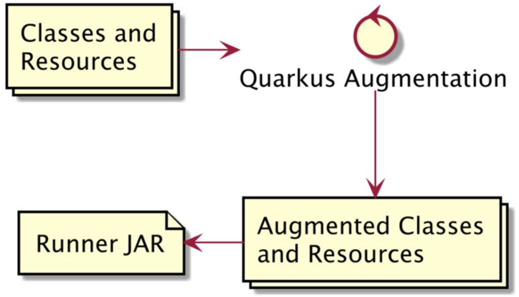
Augmentation

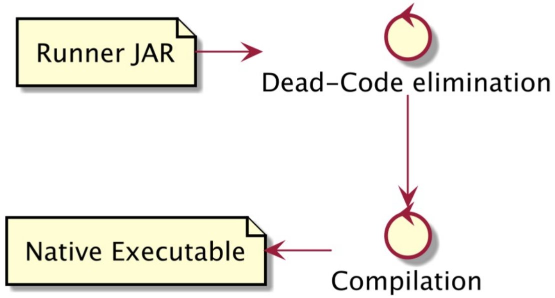
Ahead-of-time techniques vs **Just-in-time**

During the build, some work like annotation scanning, XML parsing, resolving dependencies, declares which classes need reflection at runtime and generates static proxies to avoid reflection, and more is pre-computed.

Quarkus can also use GraalVM to generate native executables using native-image.

This has two direct benefits: **faster startup** time and **lower memory** consumption.





Move Forward!

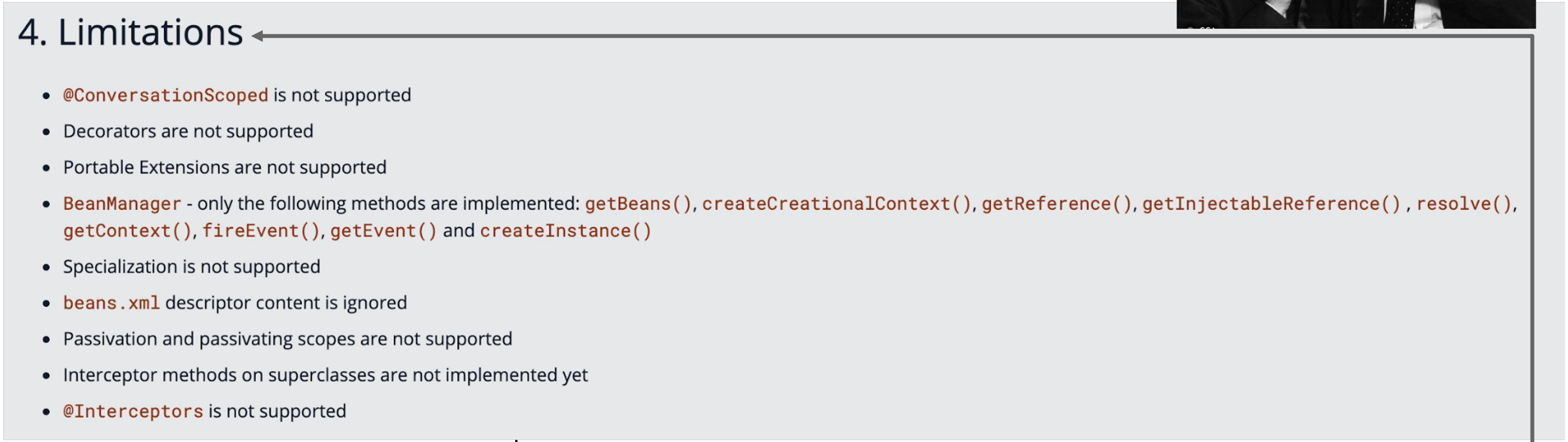
Quarkus Core Philosophy

Quarkus aims to do as much work as possible at build time, to keep the resulting application as small and fast as possible = **Fast Ignition**

Runtime should only contain classes that are needed to actually run the application. = **Min footprint**

CDI - The foundation

Don’t tell anyone



4. Limitations

- `@ConversationScoped` is not supported
- Decorators are not supported
- Portable Extensions are not supported
- `BeanManager` - only the following methods are implemented: `getBeans()`, `createCreationalContext()`, `getReference()`, `getInjectableReference()` , `resolve()`, `getContext()`, `fireEvent()`, `getEvent()` and `createInstance()`
- Specialization is not supported
- `beans.xml` descriptor content is ignored
- Passivation and passivating scopes are not supported
- Interceptor methods on superclasses are not implemented yet
- `@Interceptors` is not supported

- **ArC** doesn’t fully implement CDI, only most commonly used **subset** of the specification is implemented.

CDI - The foundation

- Context Dependency Injection - CDI
  - Injecting bean into another
  - Injecting configuration
  - Injecting resources to a component
- CDI is built on the concept of "loose coupling, strong typing", meaning that beans are loosely coupled, but in a strongly-typed way.
- CDI is also bringing interceptors, decorators and events to DI.
- Quarkus is based on a CDI implementation called **ArC**
- **ArC** doesn’t fully implement CDI, only most commonly used **subset** of the specification is implemented.

ArC - The magic

- ArC is a build-time oriented dependency injection based on CDI 2.0
- Beans and proxies generated at build time
- Removing Unused Beans (In standard CDI, all beans are retained by the container no matter whether they're needed or not)
- Minimal reflection (private members only)
- Startup is very fast

ArC Supported features
https://quarkus.io/guides/cdi-reference#supported_features

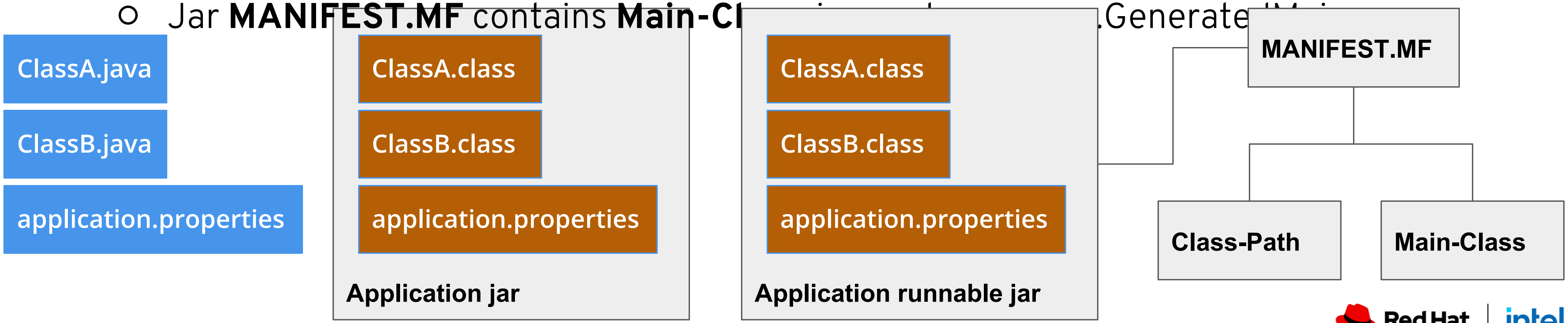
ArC plus integration runtime consist of 72 classes and occupies ~ 140 KB in jars.

Weld 3.1.1 (CDI Reference Implementation) core is roughly 1200 classes and approx. 2 MB jar.

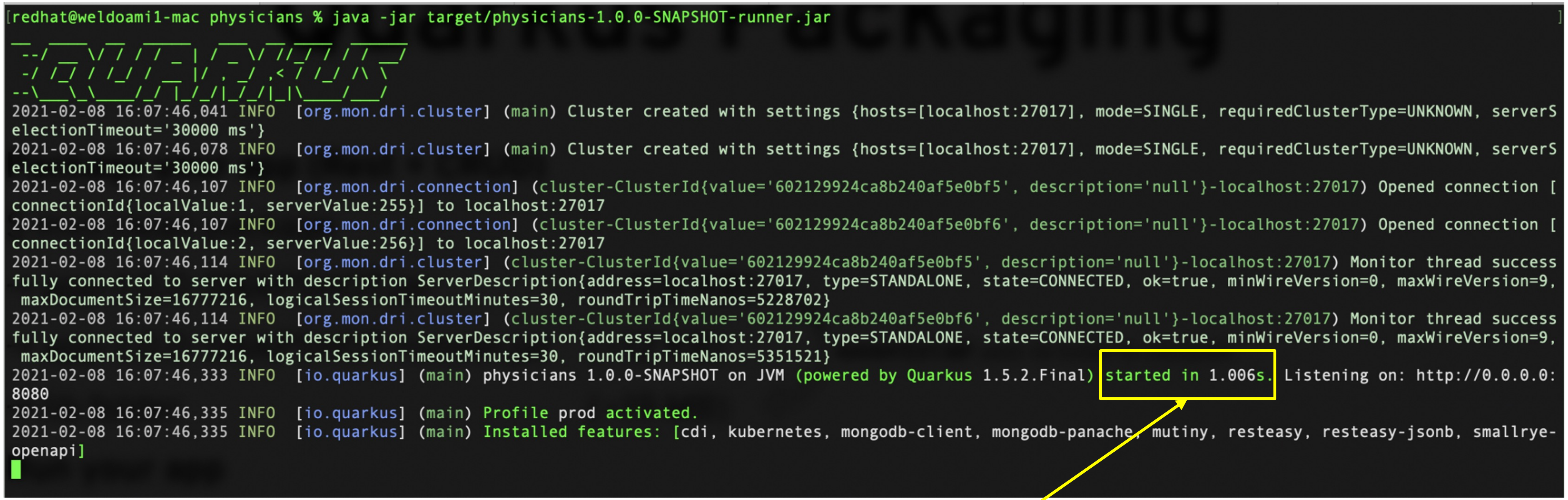
In other words, ArC runtime takes approx. 7% of the Weld runtime in terms of number of classes and jar footprint.

Quarkus Packaging

- **Application** code only Jar
- **Executable** (Runnable) Jar
  - It is an executable JAR, **not an Uber-JAR**
  - Quarkus copies all the dependencies into the **target/quarkus-app/lib** directory
  - All dependencies are listed under **target/quarkus-app/quarkus-app-dependencies.txt**
  - Jar **MANIFEST.MF** contains **Class-Path** pointing to all jars under target/lib directory
  - Jar **MANIFEST.MF** contains **Main-Cl** .Generate



Quarkus Packaging



```
[redhat@weldoami1-mac physicians % java -jar target/physicians-1.0.0-SNAPSHOT-runner.jar
__  ____  __  _____   ___  __ ____  ______
 --/ __ \/ / / / _ | / _ \/ //_/ / / / __/
 -/ /_/ / /_/ / __ |/ , _/ ,< / /_/ /\ \
--\___\_\____/_/ |_/_/|_/_/|_|\____/___/
2021-02-08 16:07:46,041 INFO  [org.mon.dri.cluster] (main) Cluster created with settings {hosts=[localhost:27017], mode=SINGLE, requiredClusterType=UNKNOWN, serverS
electionTimeout='30000 ms'}
2021-02-08 16:07:46,078 INFO  [org.mon.dri.cluster] (main) Cluster created with settings {hosts=[localhost:27017], mode=SINGLE, requiredClusterType=UNKNOWN, serverS
electionTimeout='30000 ms'}
2021-02-08 16:07:46,107 INFO  [org.mon.dri.connection] (cluster-ClusterId{value='602129924ca8b240af5e0bf5', description='null'}-localhost:27017) Opened connection [
connectionId{localValue:1, serverValue:255}] to localhost:27017
2021-02-08 16:07:46,107 INFO  [org.mon.dri.connection] (cluster-ClusterId{value='602129924ca8b240af5e0bf6', description='null'}-localhost:27017) Opened connection [
connectionId{localValue:2, serverValue:256}] to localhost:27017
2021-02-08 16:07:46,114 INFO  [org.mon.dri.cluster] (cluster-ClusterId{value='602129924ca8b240af5e0bf5', description='null'}-localhost:27017) Monitor thread success
fully connected to server with description ServerDescription{address=localhost:27017, type=STANDALONE, state=CONNECTED, ok=true, minWireVersion=0, maxWireVersion=9,
 maxDocumentSize=16777216, logicalSessionTimeoutMinutes=30, roundTripTimeNanos=5228702}
2021-02-08 16:07:46,114 INFO  [org.mon.dri.cluster] (cluster-ClusterId{value='602129924ca8b240af5e0bf6', description='null'}-localhost:27017) Monitor thread success
fully connected to server with description ServerDescription{address=localhost:27017, type=STANDALONE, state=CONNECTED, ok=true, minWireVersion=0, maxWireVersion=9,
 maxDocumentSize=16777216, logicalSessionTimeoutMinutes=30, roundTripTimeNanos=5351521}
2021-02-08 16:07:46,333 INFO  [io.quarkus] (main) physicians 1.0.0-SNAPSHOT on JVM (powered by Quarkus 1.5.2.Final) started in 1.006s. Listening on: http://0.0.0.0:
8080
2021-02-08 16:07:46,335 INFO  [io.quarkus] (main) Profile prod activated.
2021-02-08 16:07:46,335 INFO  [io.quarkus] (main) Installed features: [cdi, kubernetes, mongodb-client, mongodb-panache, mutiny, resteasy, resteasy-jsonb, smallrye-
openapi]
```

```
$ java -jar target/physicians-1.0.0-SNAPSHOT-runner.jar
```

Quarkus Native Executable

Rest + JPA CRUD ~ **1 second** startup time, **is this enough!**

Do you think 1 second is CLOUD ENOUGH!

Quarkus Native Executable

Rest + JPA CRUD ~ **1 second** startup time, **is this enough!**

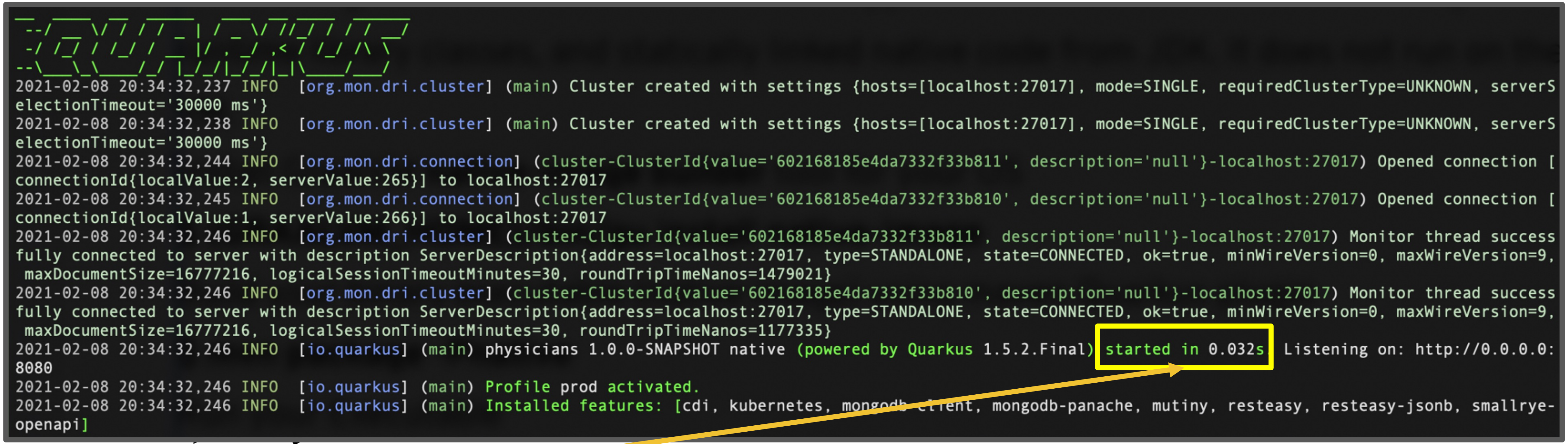
Go Native!

Native Image

Rest + JPA CRUD (Native) ~ **36 milliseconds**

Quarkus on GraalVM

Native Image



```
__  ____  __  _____   ___  __ ____  ______
 --/ __ \/ / / / _ | / _ \/ //_/ / / / __/
 -/ /_/ / /_/ / __ |/ , _/ ,< / /_/ /\ \
--\___\_\____/_/ |_/_/|_/_/|_|\____/___/
2021-02-08 20:34:32,237 INFO  [org.mon.dri.cluster] (main) Cluster created with settings {hosts=[localhost:27017], mode=SINGLE, requiredClusterType=UNKNOWN, serverS
electionTimeout='30000 ms'}
2021-02-08 20:34:32,238 INFO  [org.mon.dri.cluster] (main) Cluster created with settings {hosts=[localhost:27017], mode=SINGLE, requiredClusterType=UNKNOWN, serverS
electionTimeout='30000 ms'}
2021-02-08 20:34:32,244 INFO  [org.mon.dri.connection] (cluster-ClusterId{value='602168185e4da7332f33b811', description='null'}-localhost:27017) Opened connection [
connectionId{localValue:2, serverValue:265}] to localhost:27017
2021-02-08 20:34:32,245 INFO  [org.mon.dri.connection] (cluster-ClusterId{value='602168185e4da7332f33b810', description='null'}-localhost:27017) Opened connection [
connectionId{localValue:1, serverValue:266}] to localhost:27017
2021-02-08 20:34:32,246 INFO  [org.mon.dri.cluster] (cluster-ClusterId{value='602168185e4da7332f33b811', description='null'}-localhost:27017) Monitor thread success
fully connected to server with description ServerDescription{address=localhost:27017, type=STANDALONE, state=CONNECTED, ok=true, minWireVersion=0, maxWireVersion=9,
 maxDocumentSize=16777216, logicalSessionTimeoutMinutes=30, roundTripTimeNanos=1479021}
2021-02-08 20:34:32,246 INFO  [org.mon.dri.cluster] (cluster-ClusterId{value='602168185e4da7332f33b810', description='null'}-localhost:27017) Monitor thread success
fully connected to server with description ServerDescription{address=localhost:27017, type=STANDALONE, state=CONNECTED, ok=true, minWireVersion=0, maxWireVersion=9,
 maxDocumentSize=16777216, logicalSessionTimeoutMinutes=30, roundTripTimeNanos=1177335}
2021-02-08 20:34:32,246 INFO  [io.quarkus] (main) physicians 1.0.0-SNAPSHOT native (powered by Quarkus 1.5.2.Final) started in 0.032s. Listening on: http://0.0.0.0:
8080
2021-02-08 20:34:32,246 INFO  [io.quarkus] (main) Profile prod activated.
2021-02-08 20:34:32,246 INFO  [io.quarkus] (main) Installed features: [cdi, kubernetes, mongodb-client, mongodb-panache, mutiny, resteasy, resteasy-jsonb, smallrye-
openapi]
```

Run your Executable

Quarkus on GraalVM

Native Image

It is a technology to **ahead-of-time** compile Java code to a **standalone executable**, called a native image. This executable includes the application classes, classes from its dependencies, runtime library classes, and statically linked native code from JDK. It does not run on the Java VM.

Install GraalVM **native-image builder** tool for your OS

$ {GRAALVM_HOME}/bin/gu install native-image

Build your binary executable (native image) using maven Quarkus plugin

$ mvn package -Pnative

Run your Executable

Deploy on Openshift

Kubernetes

When bootstrapping Quarkus application, Two Docker files are generated

- **Dockerfile.jvm**: To containerize the application using the generated JAR
- **Dockerfile.native**: To containerize the application using the native executable

For **Openshift** deployment

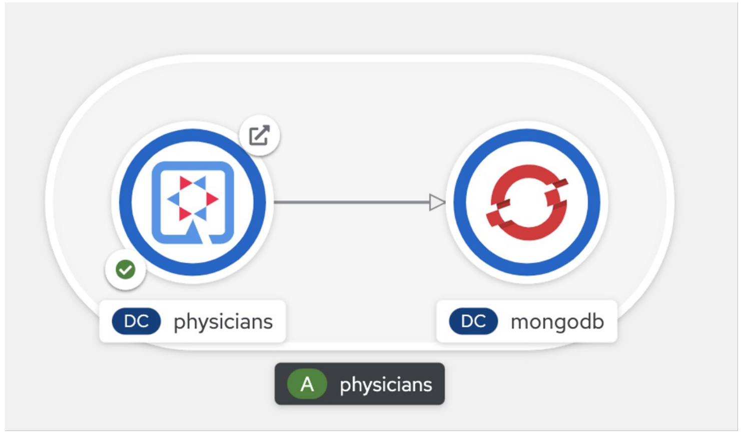
Use the magic of **S2I**

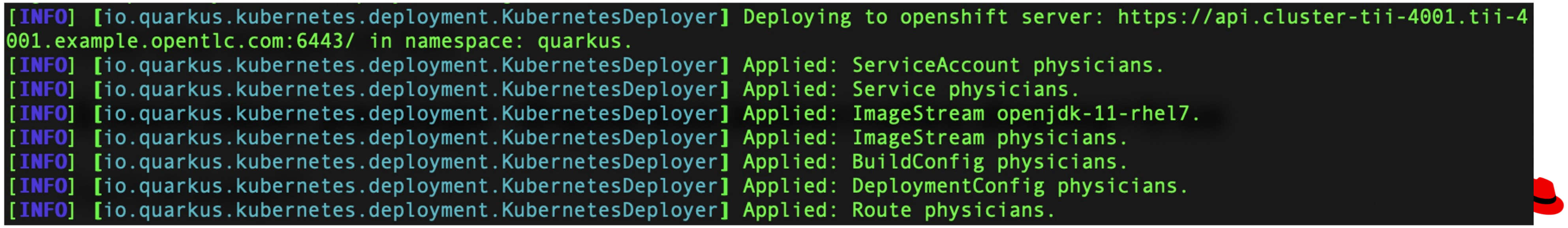
$ mvn clean package -Dquarkus.kubernetes.deploy=true

Add **-Dquarkus.kubernetes-client.trust-certs=true** to accept self-signed certs

Isn't it easy!!!

But What is happening?



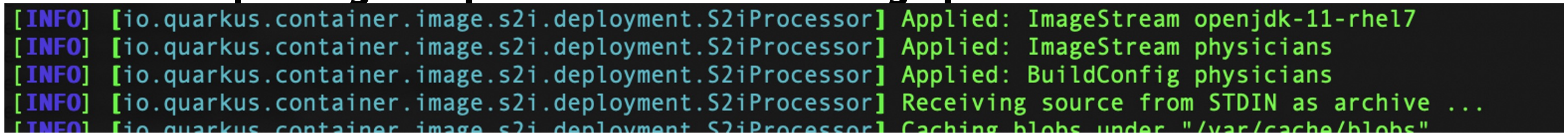


```
[INFO] [io.quarkus.kubernetes.deployment.KubernetesDeployer] Deploying to openshift server: https://api.cluster-tii-4001.tii-4
001.example.opentlc.com:6443/ in namespace: quarkus.
[INFO] [io.quarkus.kubernetes.deployment.KubernetesDeployer] Applied: ServiceAccount physicians.
[INFO] [io.quarkus.kubernetes.deployment.KubernetesDeployer] Applied: Service physicians.
[INFO] [io.quarkus.kubernetes.deployment.KubernetesDeployer] Applied: ImageStream openjdk-11-rhel7.
[INFO] [io.quarkus.kubernetes.deployment.KubernetesDeployer] Applied: ImageStream physicians.
[INFO] [io.quarkus.kubernetes.deployment.KubernetesDeployer] Applied: BuildConfig physicians.
[INFO] [io.quarkus.kubernetes.deployment.KubernetesDeployer] Applied: DeploymentConfig physicians.
[INFO] [io.quarkus.kubernetes.deployment.KubernetesDeployer] Applied: Route physicians.
```

Deploy on Openshift

Push image

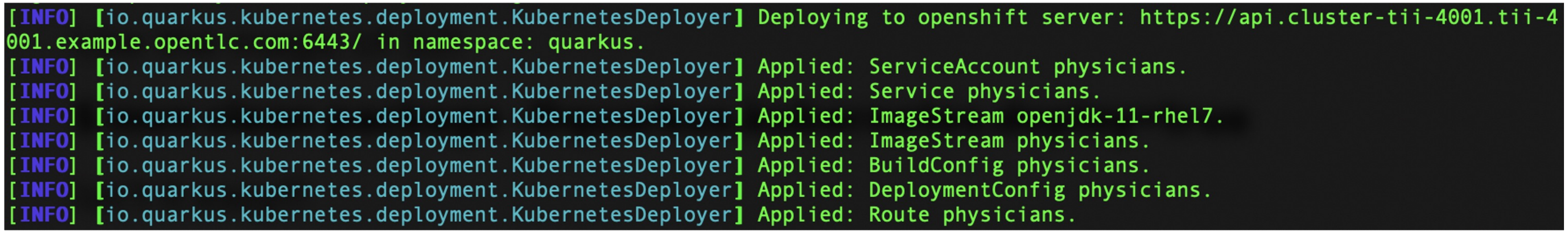
$ mvn clean package -Dquarkus.container-image.push=true



```
[INFO] [io.quarkus.container.image.s2i.deployment.S2iProcessor] Applied: ImageStream openjdk-11-rhel7
[INFO] [io.quarkus.container.image.s2i.deployment.S2iProcessor] Applied: ImageStream physicians
[INFO] [io.quarkus.container.image.s2i.deployment.S2iProcessor] Applied: BuildConfig physicians
[INFO] [io.quarkus.container.image.s2i.deployment.S2iProcessor] Receiving source from STDIN as archive ...
[INFO] [io.quarkus.container.image.s2i.deployment.S2iProcessor] Caching blobs under "/var/cache/blobs"
```

Push image and deploy the app

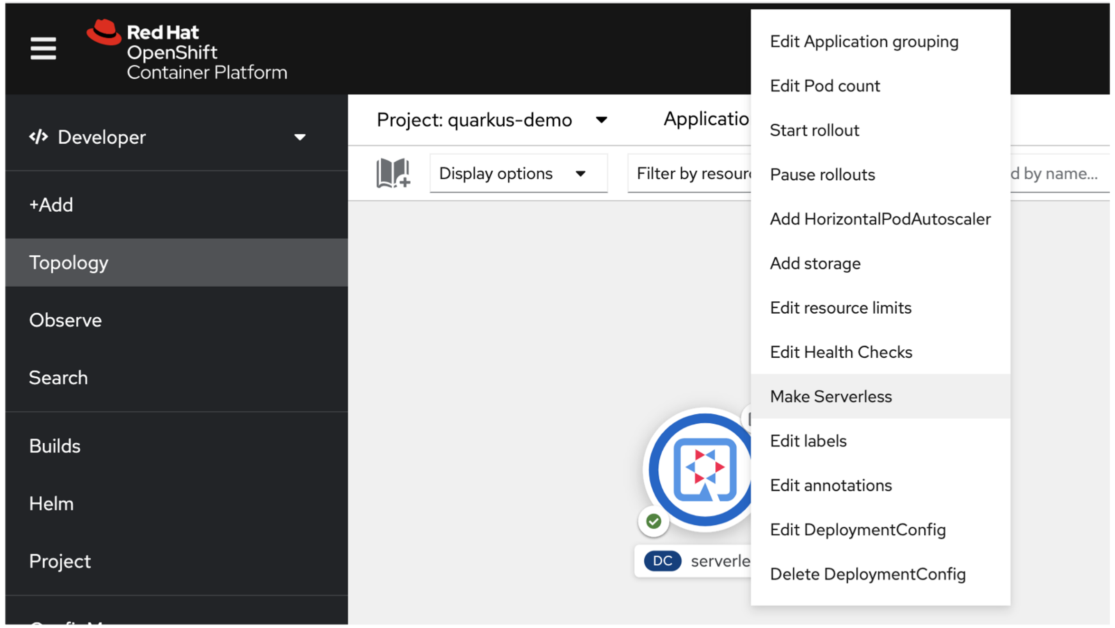
$mvn clean package -Dquarkus.kubernetes.deploy=true

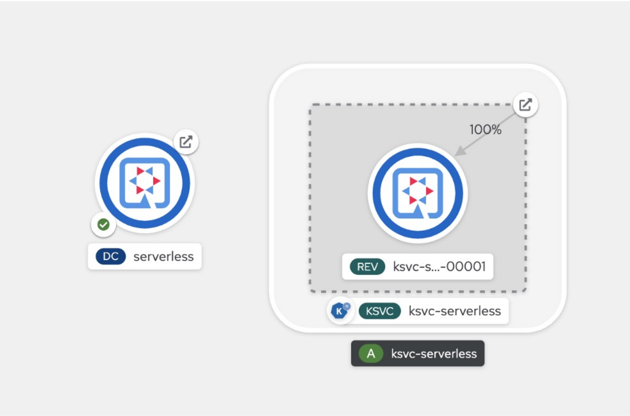


```
[INFO] [io.quarkus.kubernetes.deployment.KubernetesDeployer] Deploying to openshift server: https://api.cluster-tii-4001.tii-4
001.example.opentlc.com:6443/ in namespace: quarkus.
[INFO] [io.quarkus.kubernetes.deployment.KubernetesDeployer] Applied: ServiceAccount physicians.
[INFO] [io.quarkus.kubernetes.deployment.KubernetesDeployer] Applied: Service physicians.
[INFO] [io.quarkus.kubernetes.deployment.KubernetesDeployer] Applied: ImageStream openjdk-11-rhel7.
[INFO] [io.quarkus.kubernetes.deployment.KubernetesDeployer] Applied: ImageStream physicians.
[INFO] [io.quarkus.kubernetes.deployment.KubernetesDeployer] Applied: BuildConfig physicians.
[INFO] [io.quarkus.kubernetes.deployment.KubernetesDeployer] Applied: DeploymentConfig physicians.
[INFO] [io.quarkus.kubernetes.deployment.KubernetesDeployer] Applied: Route physicians.
```

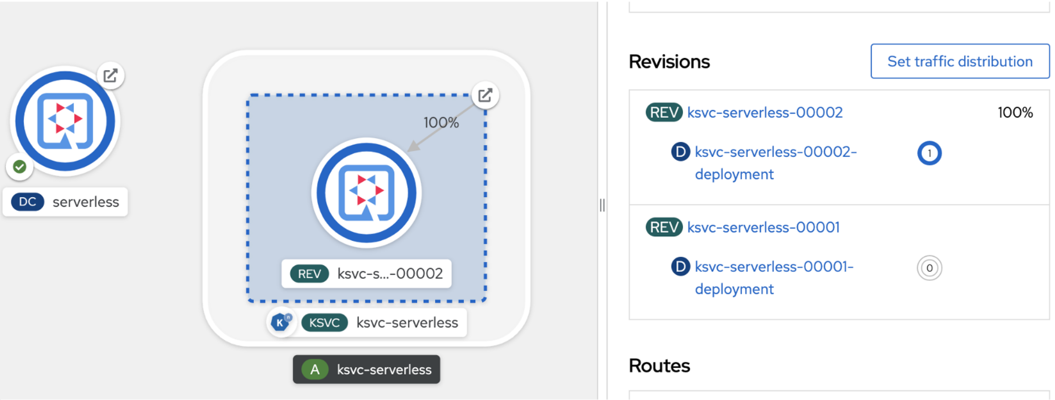
Serverless it!

1 - Create Knative svc

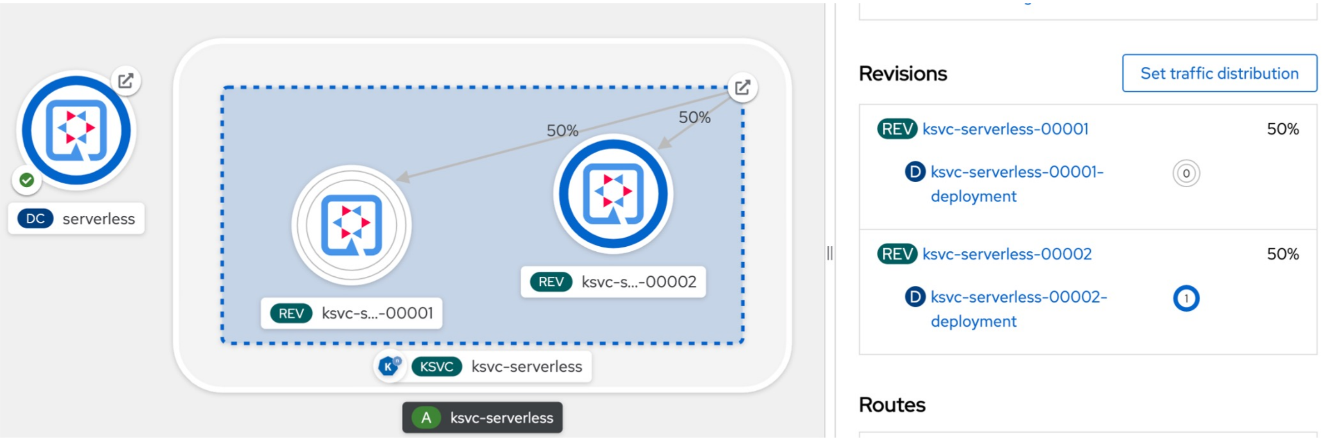




2 - Create revision


100%
DC serverless
REV ksvc-s...-00002
KSVC ksvc-serverless
A ksvc-serverless
Revisions
Set traffic distribution
REV ksvc-serverless-00002 100%
D ksvc-serverless-00002-deployment 1
REV ksvc-serverless-00001
D ksvc-serverless-00001-deployment 0
Routes

3 - Split the traffic



Profiles

Quarkus supports the notion of configuration profiles. This allows you to have multiple configurations in the same file and to select them via a profile name.

By default, Quarkus has three profiles, although it is possible to create your own and use as many as you like. The built-in profiles are:

- dev: Activated when in development mode (when running mvn quarkus:dev).
- test: Activated when running tests.
- prod: The default profile when not running in development or test mode.

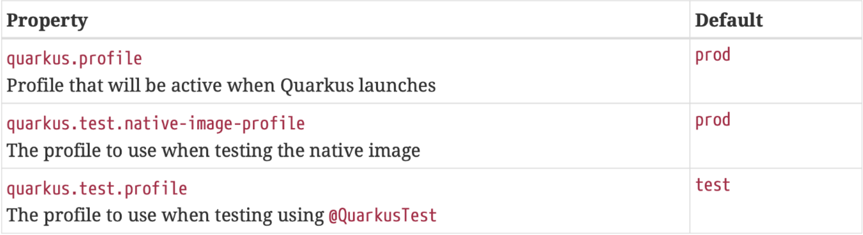
The syntax is **%{profile}**.config.key=value in the application.properties file.

For example

%dev.quarkus.mongodb.connection-string = mongodb://localhost:27017/persons

If profile is omitted, then the property works for all

Some Quarkus Profile Configuration Properties



Property	Default
quarkus.profile Profile that will be active when Quarkus launches	prod
quarkus.test.native-image-profile The profile to use when testing the native image	prod
quarkus.test.profile The profile to use when testing using @QuarkusTest	test

Then, you set the system variable depending on your needs:

- Use **mvn -Dquarkus.profile=staging quarkus:dev** if you are developing,
- Or **java -Dquarkus.profile=staging -jar profiles-1.0-runner.jar** if you are running your executable JAR.

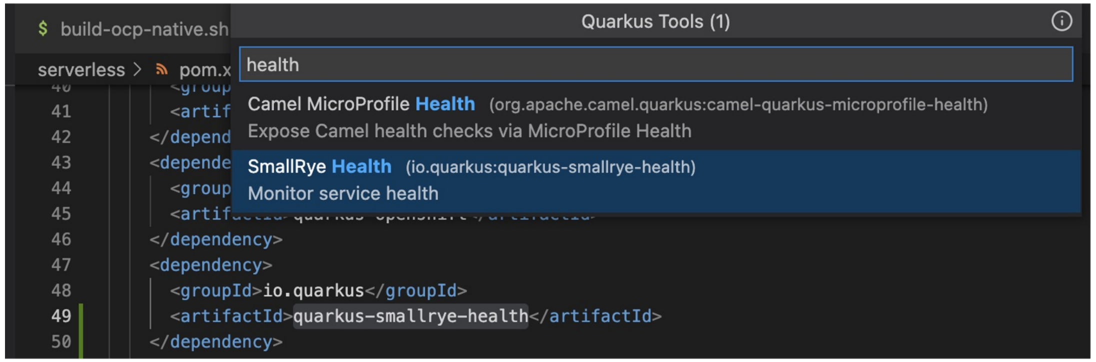
Metrics and Health Check

Metrics in two steps

- Install Quarkus Prometheus extension
- Access your metrics http://URL:PORT/**q/metrics**

Health in two steps

- Install Quarkus smallrye health extension



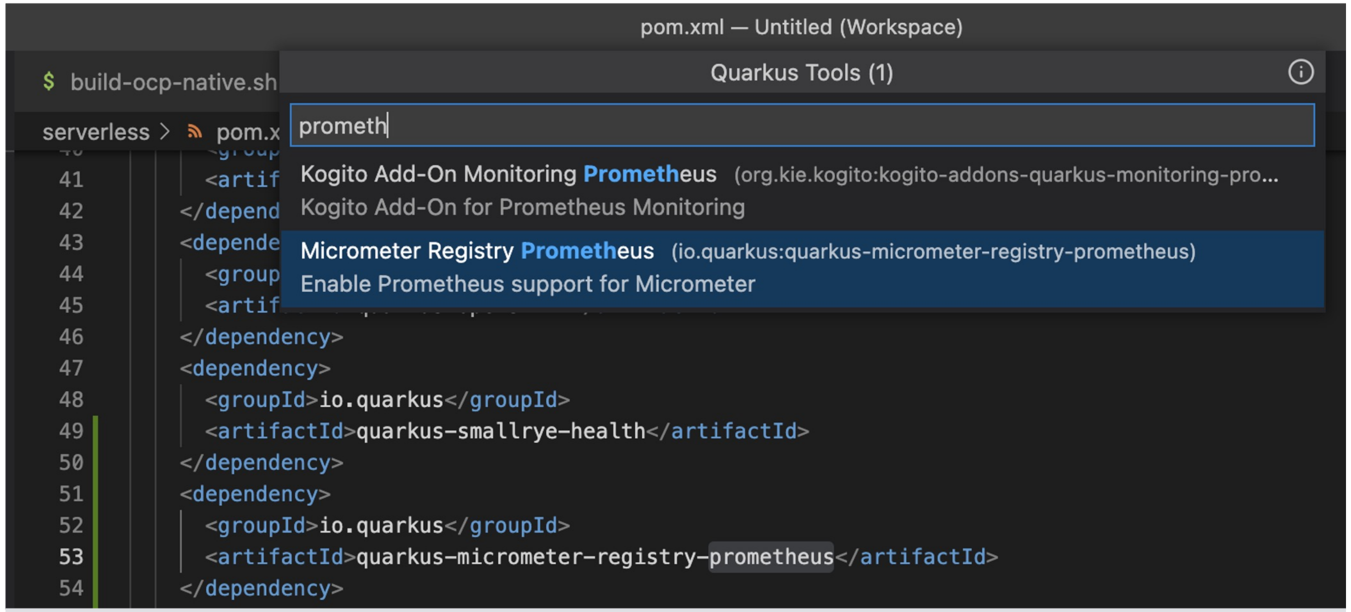
health

Camel MicroProfile **Health** (org.apache.camel.quarkus:camel-quarkus-microprofile-health)
Expose Camel health checks via MicroProfile Health

SmallRye **Health** (io.quarkus:quarkus-smallrye-health)
Monitor service health

```
    <artif
  </depend
  <depende
    <group
    <artif
  </dependency>
  <dependency>
    <groupId>io.quarkus</groupId>
    <artifactId>quarkus-smallrye-health</artifactId>
  </dependency>
```

- Access your health probes
  - **/q/health/live** - The application is up and running.
  - **/q/health/ready** - The application is ready to serve requests.
  - **/q/health/started** - The application is started.
  - **/q/health** - Accumulating all health check procedures in the application.

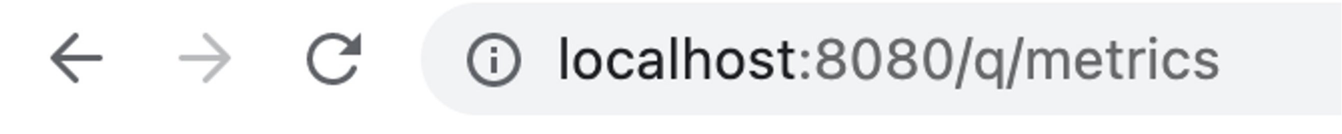


prometh

Kogito Add-On Monitoring **Prometheus** (org.kie.kogito:kogito-addons-quarkus-monitoring-pro...
Kogito Add-On for Prometheus Monitoring

Micrometer Registry **Prometheus** (io.quarkus:quarkus-micrometer-registry-prometheus)
Enable Prometheus support for Micrometer

```
    <artif
  </depend
  <depende
    <group
    <artif
  </dependency>
  <dependency>
    <groupId>io.quarkus</groupId>
    <artifactId>quarkus-smallrye-health</artifactId>
  </dependency>
  <dependency>
    <groupId>io.quarkus</groupId>
    <artifactId>quarkus-micrometer-registry-prometheus</artifactId>
  </dependency>
```

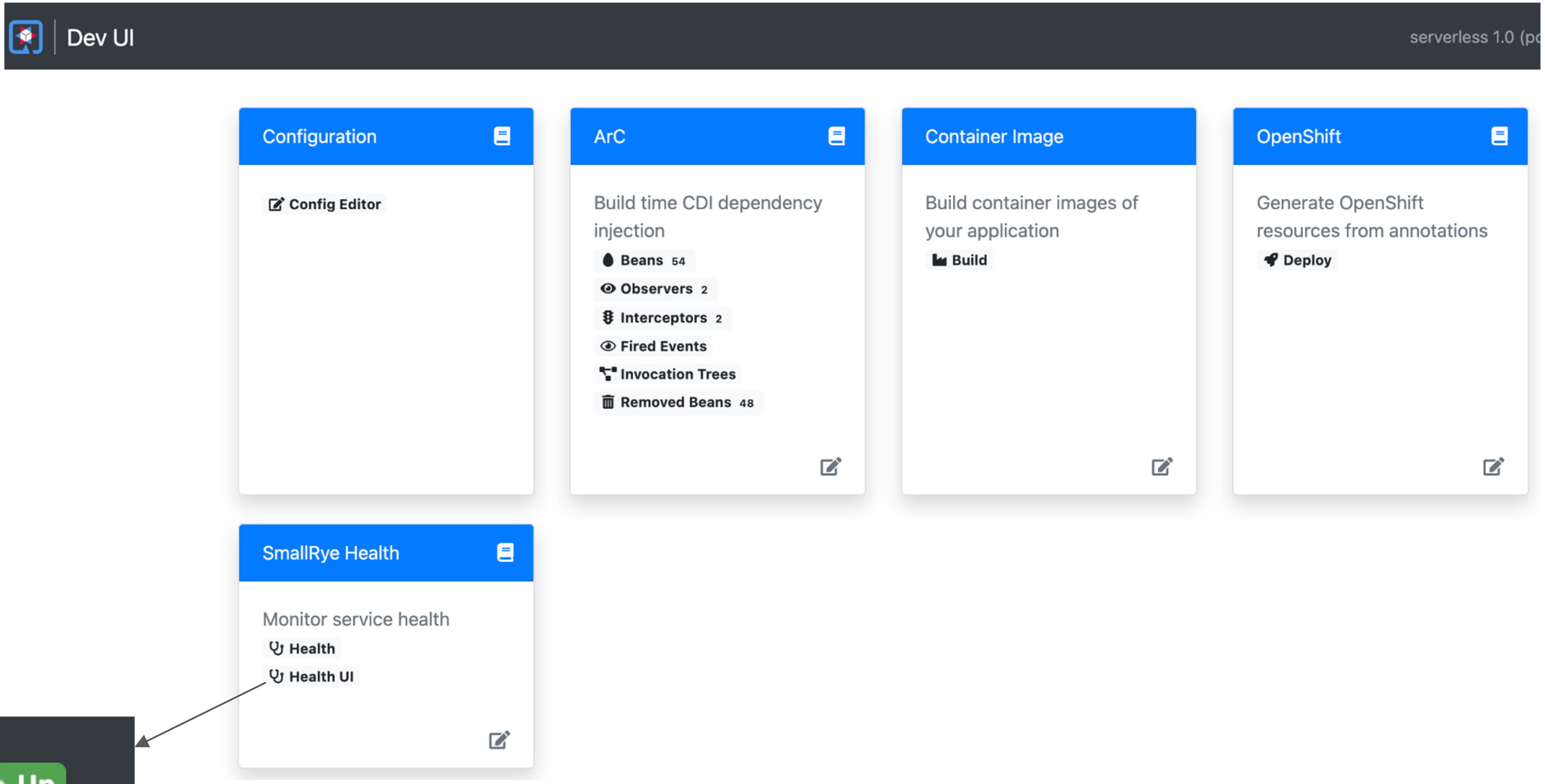


localhost:8080/q/metrics

Dev UI

Start you app in Dev mode

- mvn quarkus:dev
- Access dev UI /**q/dev**



Dev UI

serverless 1.0 (pc

Configuration

Config Editor

ArC

Build time CDI dependency injection

Beans 54

Observers 2

Interceptors 2

Fired Events

Invocation Trees

Removed Beans 48

Container Image

Build container images of your application

Build

OpenShift

Generate OpenShift resources from annotations

Deploy

SmallRye Health

Monitor service health

Health

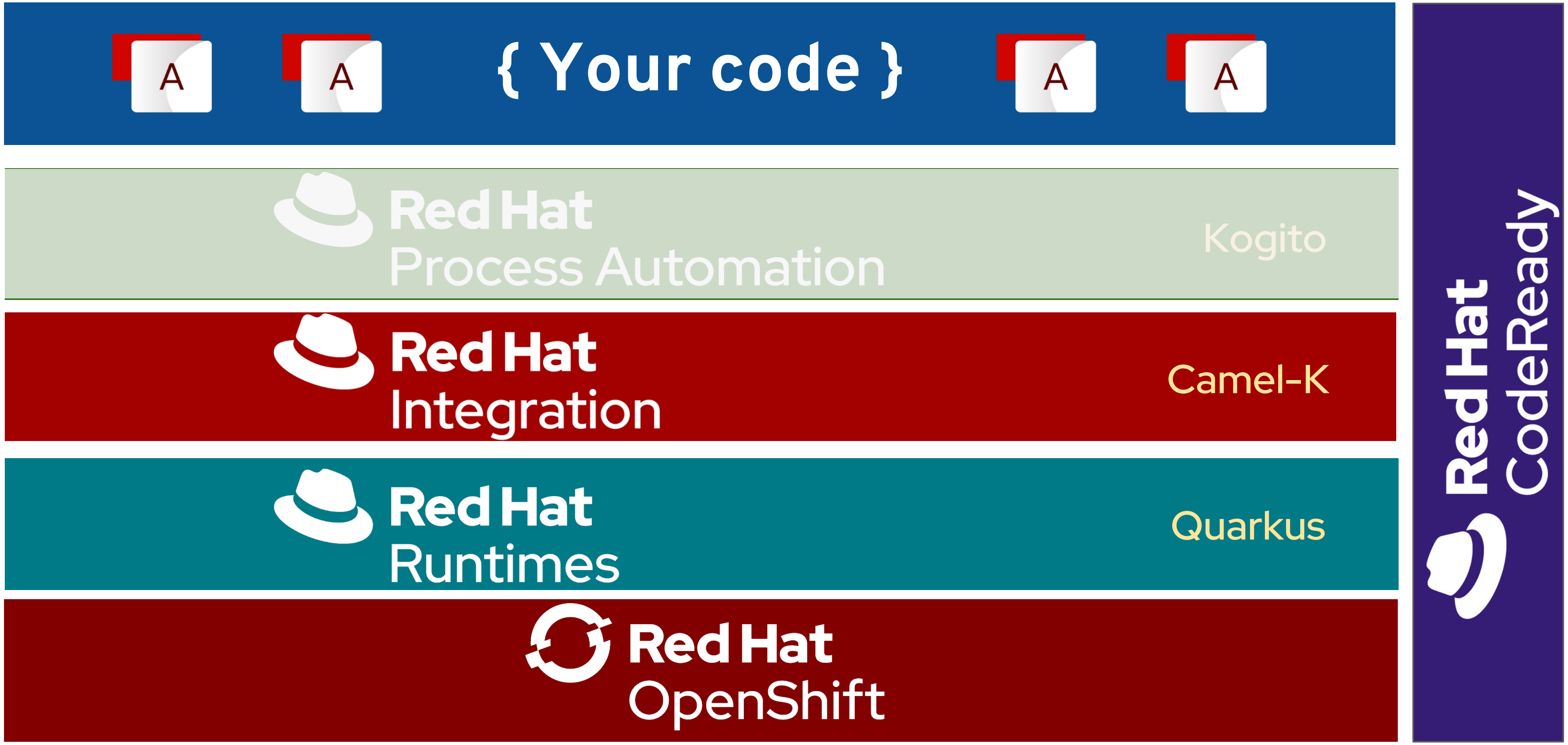
Health UI

Health UI

Up

Application Environment with Red Hat

"Quarkus powers the next-generation Red Hat stack for hybrid-cloud applications"



A A { Your code } A A

Red Hat Process Automation — Kogito

Red Hat Integration — Camel-K

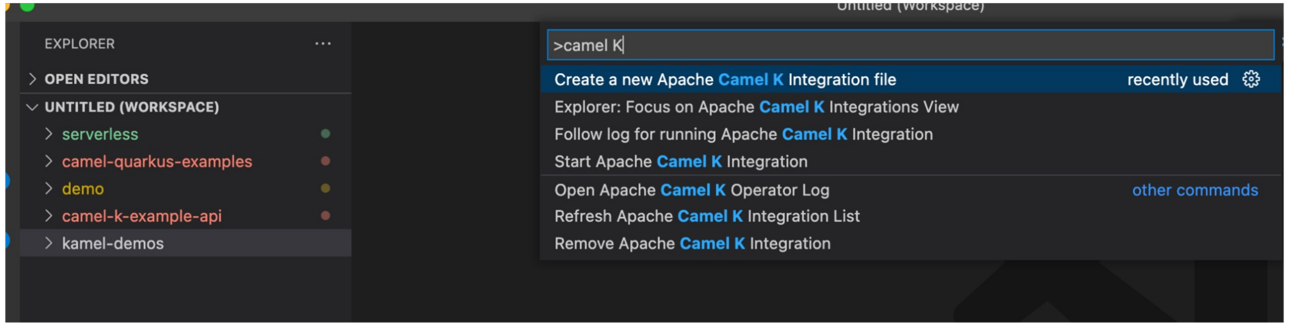
Red Hat Runtimes — Quarkus

Red Hat OpenShift

Red Hat CodeReady

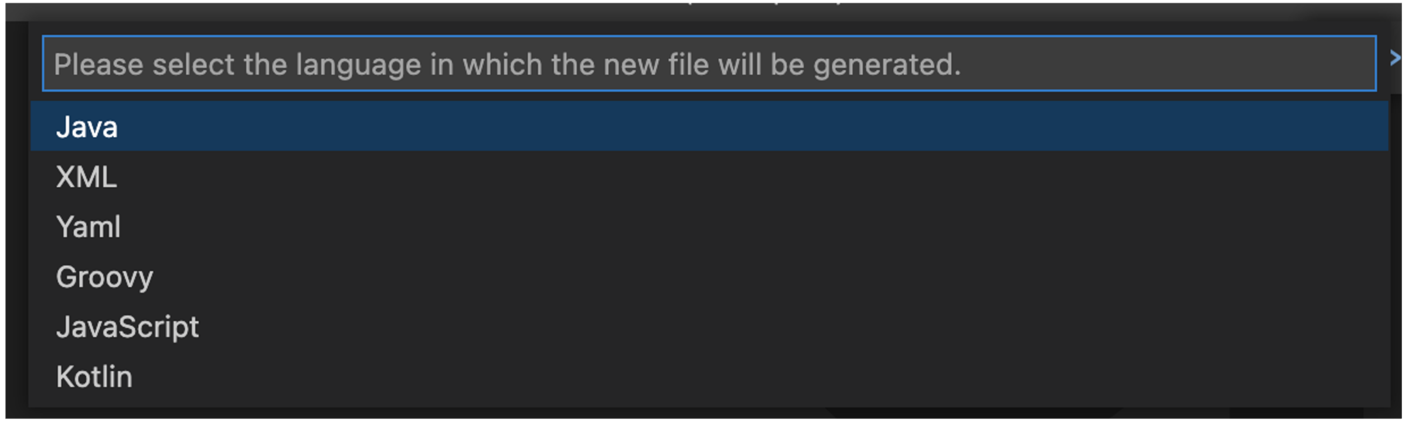
CamelK

- Build and run your camel routes natively on Kubernetes on suing serverless and microservice architectures
- Architectured by Kubernetes CRDs and Operators
- Part of Apache Camel. Started on August 31st, 2018



>camel K

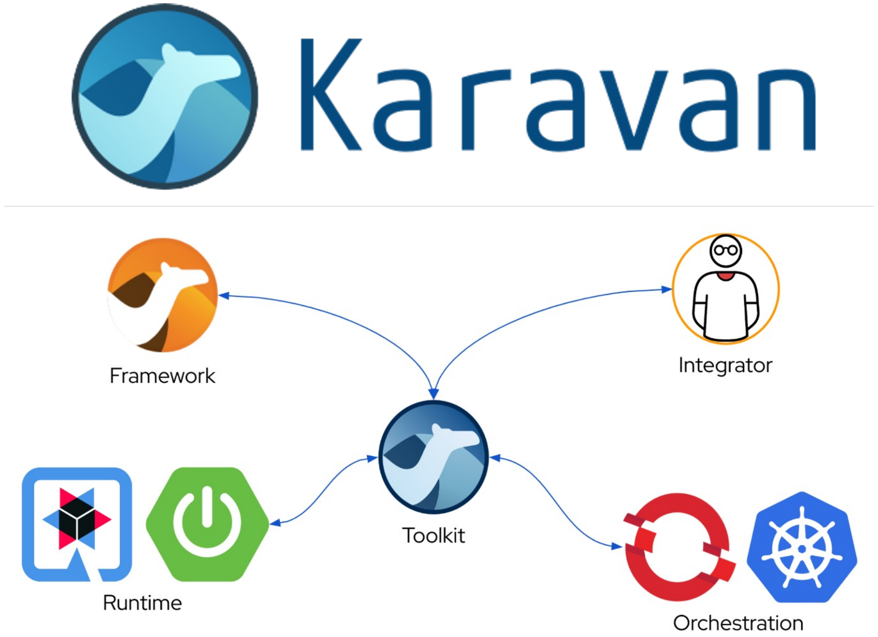
- Create a new Apache Camel K Integration file — recently used
- Explorer: Focus on Apache Camel K Integrations View
- Follow log for running Apache Camel K Integration
- Start Apache Camel K Integration
- Open Apache Camel K Operator Log — other commands
- Refresh Apache Camel K Integration List
- Remove Apache Camel K Integration



Please select the language in which the new file will be generated.

- Java
- XML
- Yaml
- Groovy
- JavaScript
- Kotlin

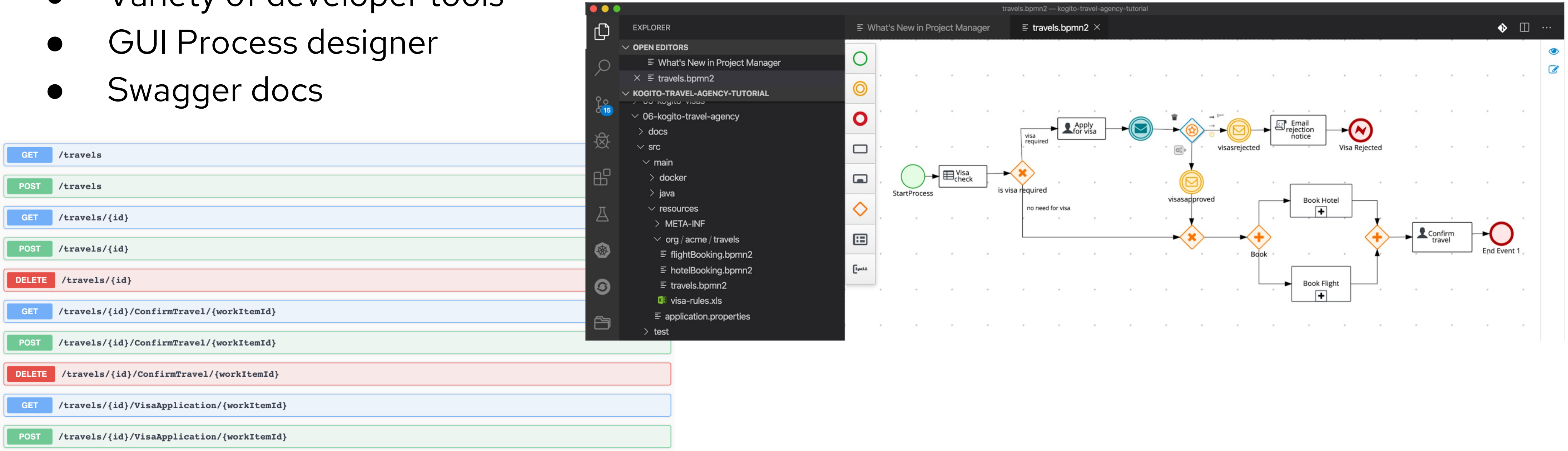
VSCode | RED HAT CODEREADY



Kogito

VSCode RED HAT CODEREADY

- Encapsulate your business processes/rules into your microservices
- Fit into Knative serverless
- Superfast boot time, low footprint (GraalVM native image)
- Operator-driven service lifecycle management
- Leveraging / integrating many other (cloud) technologies
- Variety of developer tools
- GUI Process designer
- Swagger docs



GET /travels
POST /travels
GET /travels/{id}
POST /travels/{id}
DELETE /travels/{id}
GET /travels/{id}/ConfirmTravel/{workItemId}
POST /travels/{id}/ConfirmTravel/{workItemId}
DELETE /travels/{id}/ConfirmTravel/{workItemId}
GET /travels/{id}/VisaApplication/{workItemId}
POST /travels/{id}/VisaApplication/{workItemId}

travels.bpmn2 — kogito-travel-agency-tutorial

EXPLORER
OPEN EDITORS
What's New in Project Manager
travels.bpmn2
KOGITO-TRAVEL-AGENCY-TUTORIAL
06-kogito-travel-agency
docs
src
main
docker
java
resources
META-INF
org / acme / travels
flightBooking.bpmn2
hotelBooking.bpmn2
travels.bpmn2
visa-rules.xls
application.properties
test

Red Hat intel.

https://github.com/wael2000/quarkus-hackathon

Demo Time

Red Hat Summit

Connect

Thank you

in linkedin.com/company/red-hat

f facebook.com/redhatinc

youtube.com/user/RedHatVideos

twitter.com/RedHat

Red Hat | intel.