02 - Your first Quarkus application

Source code

The application is created with https://code.quarkus.io/ and the **RESTEasy Classic** extension.

Develop with developer joy

By running the maven quarku plugin:

```
mvn quarkus:dev
```

Using live coding enables us to **update Java source, resources, and configuration of a running application**. All changes are reflected in the running application automatically, enabling developers to improve the turnaround time when developing a new application. Live coding enables hot deployment via **background compilation**. Any changes to the Java source, or resources, will be reflected **as soon as the application receives a new request from the browser**. Refreshing the browser or issuing a new browser request triggers a scan of the project for any changes to then recompile and redeploy the application. If any issues arise with compilation or deployment, an error page provides details of the problem.

Add lombok support

```
<dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <version>1.18.30</version>
    <scope>provided</scope>
</dependency>
```

Resteasy client and serialization

```
<dependency>
    <groupId>io.quarkus</groupId>
    <artifactId>quarkus-resteasy</artifactId>
</dependency>
<!-- Introduces json serialization in the rest calls   -->
<dependency>
    <groupId>io.quarkus</groupId>
    <artifactId>quarkus-rest-client-jackson</artifactId>
</dependency>
<dependency>
```

Test the endpoints

```
<dependency>
 <groupId>io.quarkus</groupId>
 <artifactId>quarkus-junit5</artifactId>
 <scope>test</scope>
</dependency>
<dependency>
 <groupId>io.rest-assured</groupId>
 <artifactId>rest-assured</artifactId>
 <scope>test</scope>
</dependency>
```

The `@QuarkusTest` tests the methods in the `AccountResource` class (including the post action)

AccountResourceTest.java

Build native (GraalVM)

The `pom.xml` file of the project contains the `native` **profile** that we can use to build the native application:

```
<profiles>
    <profile>
      <id>native</id>
```

```
mvn clean install -Pnative
mvn clean install -Dquarkus.package.type=native
```

You can build Quarkus native executables in two ways:

- Use a **Container Image of GraalVM**. This option does not require installing GraalVM locally
- Install **GraalVM locally** and use it to build a native executable

Build with the container image

We need Docker running in the host

```
service docker start
```

Then we can build using the **container build** option

```
quarkus build --native -Dquarkus.native.container-build=true
```

Aleternatevely, we can set the property in the `application.properties` file

```
quarkus.native.container-build=true
```

The build will

```
[INFO] -------------< org.acme:02-your-first-quarkus-application >-------------
[INFO] Building 02-your-first-quarkus-application 1.0.0-SNAPSHOT
[INFO]   from pom.xml
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- quarkus:3.5.0:build (default-cli) @ 02-your-first-quarkus-application -
--
[INFO] [io.quarkus.deployment.pkg.steps.JarResultBuildStep] Building native image
source jar: C:\projects\personal\kubernetes-native-microservices-sources\02-your-
first-quarkus-application\target\02-your-first-quarkus-application-1.0.0-SNAPSHOT-
native-image-source-jar\02-your-first-quarkus-application-1.0.0-SNAPSHOT-
runner.jar
```

The container build option will make the build pull the `builder-image` (`ubi-quarkus-mandrel-builder-image:jdk-21`) from the Docker hub to the local Docker instance

```
[INFO] [io.quarkus.deployment.pkg.steps.NativeImageBuildContainerRunner] Using
docker to run the native image builder
[INFO] [io.quarkus.deployment.pkg.steps.NativeImageBuildContainerRunner] Checking
status of builder image 'quay.io/quarkus/ubi-quarkus-mandrel-builder-image:jdk-21'
jdk-21: Pulling from quarkus/ubi-quarkus-mandrel-builder-image
01858fc5b538: Pulling fs layer
9584a3317024: Pulling fs layer
...
...
Status: Downloaded newer image for quay.io/quarkus/ubi-quarkus-mandrel-builder-
image:jdk-21
quay.io/quarkus/ubi-quarkus-mandrel-builder-image:jdk-21
```

Once the container is running the build command is sent

```
[INFO] [io.quarkus.deployment.pkg.steps.NativeImageBuildStep] Running Quarkus
native-image plugin on MANDREL 23.1.1.0 JDK 21.1
[INFO] [io.quarkus.deployment.pkg.steps.NativeImageBuildRunner] docker run --env
LANG=C --rm -v /c/projects/personal/kubernetes-native-microservices-sources/02-
your-first-quarkus-application/target/02-your-first-quarkus-application-1.0.0-
SNAPSHOT-native-image-source-jar:/project:z...
...
==================================================================================
======================================
GraalVM Native Image: Generating '02-your-first-quarkus-application-1.0.0-
SNAPSHOT-runner' (executable)...
```

```
===================================================================================
======================================
For detailed information and explanations on the build output, visit:
https://github.com/oracle/graal/blob/master/docs/reference-manual/native-
image/BuildOutput.md
-----------------------------------------------------------------------------------
--------------------------------------
[1/8] Initializing...
(13,8s @ 0,15GB)
 Java version: 21.0.1+12-LTS, vendor version: Mandrel-23.1.1.0-Final
 Graal compiler: optimization level: 2, target machine: x86-64-v3
 C compiler: gcc (redhat, x86_64, 8.5.0)
 Garbage collector: Serial GC (max heap size: 80% of RAM)
 4 user-specific feature(s):
 - com.oracle.svm.thirdparty.gson.GsonFeature
 - io.quarkus.runner.Feature: Auto-generated class by Quarkus from the existing
extensions
 - io.quarkus.runtime.graal.DisableLoggingFeature: Disables INFO logging during
the analysis phase
 - org.eclipse.angus.activation.nativeimage.AngusActivationFeature
-----------------------------------------------------------------------------------
--------------------------------------
 4 experimental option(s) unlocked:
 - '-H:+AllowFoldMethods' (origin(s): command line)
 - '-H:BuildOutputJSONFile' (origin(s): command line)
 - '-H:-UseServiceLoaderFeature' (origin(s): command line)
 - '-H:ReflectionConfigurationResources' (origin(s): 'META-INF/native-
image/io.netty/netty-transport/native-image.properties' in
'file:///project/lib/io.netty.netty-transport-4.1.100.Final.jar')
-----------------------------------------------------------------------------------
--------------------------------------
Build resources:
 - 10,00GB of memory (64,1% of 15,60GB system memory, determined at start)
 - 12 thread(s) (100,0% of 12 available processor(s), determined at start)
[2/8] Performing analysis...
...
----------------------------------------------------------------
Produced artifacts:
 /project/02-your-first-quarkus-application-1.0.0-SNAPSHOT-runner (executable)
 /project/02-your-first-quarkus-application-1.0.0-SNAPSHOT-runner-build-output-
stats.json (build_info)
===================================================================================
======================================
Finished generating '02-your-first-quarkus-application-1.0.0-SNAPSHOT-runner' in
1m 28s.
```

The output contains also an option to run the native image in docker

```
[INFO] [io.quarkus.deployment.pkg.steps.NativeImageBuildRunner] docker run --env
LANG=C --rm -v /c/projects/personal/kubernetes-native-microservices-sources/02-
your-first-quarkus-application/target/02-your-first-quarkus-application-1.0.0-
SNAPSHOT-native-image-source-jar:/project:z --entrypoint /bin/bash
```

```
quay.io/quarkus/ubi-quarkus-mandrel-builder-image:jdk-21 -c objcopy --strip-debug
02-your-first-quarkus-application-1.0.0-SNAPSHOT-runner
```

We can also run the native app locally from

```
./target/02-your-first-quarkus-application-1.0.0-SNAPSHOT-runner
```

NOTE: GraalVM must be installed

Deploy the native applications to Kubernetes/Openshift

Quakus implement the possibility to **build kubernetes deployment application YAML files automatically**

```
<dependency>
 <groupId>io.quarkus</groupId>
 <artifactId>quarkus-kubernetes</artifactId>
</dependency>
```

When running `clean install` the dependency will generate both json and yaml config files in `/target/kubernetes`.

Add Quarkus Openshift extension

Guide

There is also an Openshift specific Quarkus extension

```
<dependency>
    <groupId>io.quarkus</groupId>
    <artifactId>quarkus-openshift</artifactId>
</dependency>
```

Build strategies

For security and convenience, OpenShift supports different build strategies that are not available in the upstream Kubernetes distributions.

Docker build

This strategy builds the artifacts (**JAR files or a native executable**) outside the OpenShift cluster, either **locally or in a CI environment**, and then **provides them to the OpenShift build system together with a `Dockerfile`**. So the produced Dockerfile is used in combination wiht the build output to create a new `ImagStream` inside the cluster.

Source to Image (S2I)

The build process is performed inside the OpenShift cluster. Using `S2I` to deploy Red Hat build of **Quarkus as a JVM application** is fully supported.

Binary S2I

This strategy uses a **JAR file as an input** to the S2I build process, which speeds up the build process and deployment of your application.

Build strategy	Support for Quarkus tooling	Support for JVM	Support for Native	Support for JVM Serverless	Support for native Serverless
Docker build	YES	YES	YES	YES	YES
S2I Binary	YES	YES	NO	NO	NO
Source S2I	NO	YES	NO	NO	NO

Deploy the native application to Openshift

If we use the `quarkus-openshift` extension, you can deploy your application to OpenShift using the **Docker build strategy**. The container is built inside the OpenShift cluster and provided as an image stream.

Building using custom `Dockerfile`

The Quarkus project includes pre-generated Dockerfiles with instructions. When you want to use a custom Dockerfile, you need to add the file in the `src/main/docker` directory or anywhere inside the module. Additionally, you need to **set the path to your Dockerfile** using the `quarkus.openshift.jvm-dockerfile` property.

Configuration

In the `application.properties`

```
quarkus.openshift.build-strategy=docker #sets the docker build strategy
quarkus.kubernetes-client.trust-certs=true #Optional, if an untrusted cert is used
quarkus.openshift.route.expose=true #expose the deployment via a route
quarkus.openshift.native-dockerfile=src/main/docker/Dockerfile.native #used for the ImageStream creation
```

This is the link to the Dockerfile

Login into the Openshift cluster

do the login action from the local machine with the desired project

Build and deploy to Openshift the application

Execute the following goal

`mvn clean package -Pnative -Dquarkus.kubernetes.deploy=true`

STEP 1: build the native image

The first step is the **native image creation** based on the Mandrel container (**docker build**) on the local Docker instance.

```
...
[INFO] [io.quarkus.deployment.pkg.steps.NativeImageBuildContainerRunner] Using
docker to run the native image builder
...
```

STEP 2: Cluster image build BuildConfig

The Dockerfile set in the `application.properties` file together with the just built native application will be used to create a **BuildConfig** generating an **ImageStream** in the cluster.

```
...
[INFO] [io.quarkus...] Starting (in-cluster) container image build for jar using:
DOCKER on server: https://api.sandbox-m4.g2pi.p1.openshiftapps.com:6443/ in
namespace:xan80-dev.
[INFO] [io.quarkus...] Applied: ImageStream s2i-java
[INFO] [io.quarkus...] Applied: ImageStream ch-02-your-first-quarkus-application
[INFO] [io.quarkus...] Applied: BuildConfig ch-02-your-first-quarkus-application
...
```

This is the generated BuildConfig in the cluster

```
kind: BuildConfig
apiVersion: build.openshift.io/v1
...
spec:
  runPolicy: Serial
  source:
    type: Dockerfile
    dockerfile: >
      <The content of the docker file in quarkus.openshift.native-dockerfile>
  strategy:
    type: Docker
    dockerStrategy: {}
  output:
    to:
      kind: ImageStreamTag
      name: 'ch-02-your-first-quarkus-application:1.0.0-SNAPSHOT'
  ...
```

STEP 3: Create the ImageStream

When the BuildConfig is applied to the cluster, the first `Build` is triggred, which will generate the `ImageStream`, using **binary s2i** build strategy. The `ImageStream` is used with the Deployment committed to the cluster.

```
[INFO] [io.quarkus...] Pulling image registry.access.redhat.com/ubi8/ubi-minimal:8.8 ...
[INFO] [io.quarkus...] STEP 1/9: FROM registry.access.redhat.com/ubi8/ubi-minimal:8.8
[INFO] [io.quarkus...] STEP 2/9: WORKDIR /work/
[INFO] [io.quarkus...] STEP 3/9: RUN chown 1001 /work     && chmod "g+rwX" /work && chown 1001:root /work
[INFO] [io.quarkus...] STEP 4/9: COPY --chown=1001:root target/*-runner /work/application
[INFO] [io.quarkus...] STEP 5/9: EXPOSE 8080
[INFO] [io.quarkus...] STEP 6/9: USER 1001
[INFO] [io.quarkus...] STEP 7/9: ENTRYPOINT ["./application","-Dquarkus.http.host=0.0.0.0"]
[INFO] [io.quarkus...] STEP 8/9: ENV "OPENSHIFT_BUILD_NAME"="ch-02-your-first-quarkus-application-1" "OPENSHIFT_BUILD_NAMESPACE"="xan80-dev"
[INFO] [io.quarkus...] STEP 9/9: LABEL "io.openshift.build.name"="ch-02-your-first-quarkus-application-1" "io.openshift.build.namespace"="xan80-dev"
...
[INFO] [io.quarkus...] Successfully pushed image-registry.openshift-image-registry.svc:5000/xan80-dev/ch-02-your-first-quarkus-application@sha256:216c50b2b129ac26d470eb7bbf7b5c8de83ac98481cbc4a307b25471744edd6c
```

This is the `ImageStream` generated:

```
kind: ImageStream
apiVersion: image.openshift.io/v1
metadata:
  ...
  name: ch-02-your-first-quarkus-application
  ...
tags:
- tag: 1.0.0-SNAPSHOT
    items:
    - created: '2023-11-07T15:34:27Z'
        dockerImageReference: >-
        image-registry.openshift-image-registry.svc:5000/xan80-dev/ch-02-your-first-quarkus-application@sha256:xxxyyy
        image: >-
        sha256:xxxyyy
        generation: 1
```

STEP 4: Deploy the application to the cluster

The process will generate the *DeploymentConfig*, the *Service* and the *Route* to expose the app endpoints externally.

```
[INFO] [io.quarkus.kubernetes.deployment.KubernetesDeployer] Applied: DeploymentConfig ch-02-your-first-quarkus-application.
[INFO] [io.quarkus.kubernetes.deployment.KubernetesDeployer] Applied: Route ch-02-your-first-quarkus-application.
[INFO] [io.quarkus.kubernetes.deployment.KubernetesDeployer] The deployed application can be accessed at: http://ch-02-your-first-quarkus-application-xan80-dev.apps.sandbox-m4.g2pi.p1.openshiftapps.com
[INFO] [io.quarkus.deployment.QuarkusAugmentor] Quarkus augmentation completed in 149895ms
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  02:45 min
[INFO] Finished at: 2023-11-07T16:34:32+01:00
[INFO] ------------------------------------------------------------------------
```

The application will respond to

http://ch-02-your-first-quarkus-application-xan80-dev.apps.sandbox-m4.g2pi.p1.openshiftapps.com/accounts