Quarkus - Container Images

Quarkus provides extensions for building (and pushing) container images. Currently it supports:

- Jib
- Docker
- S2I

Container Image extensions

Jib

The extension `quarkus-container-image-jib` is powered by Jib for performing container image builds. The major benefit of using Jib with Quarkus is that all the dependencies (everything found under `target/lib`) are cached in a different layer than the actual application making rebuilds really fast and small (when it comes to pushing). Another important benefit of using this extension is that it provides the ability to create a container image without having to have any dedicated client side tooling (like Docker) or running daemon processes (like the Docker daemon) when all that is needed is the ability to push to a container image registry.

To use this feature, add the following extension to your project:

```
./mvnw quarkus:add-extension -Dextensions="container-image-jib"
```

In situations where all that is needed to build a container image and no push to a registry is necessary (essentially by having set `quarkus.container-image.build=true` and left `quarkus.container-image.push` unset - it defaults to `false`), then this extension creates a container image and registers it with the Docker daemon. This means that although Docker isn't used to build the image, it is nevertheless necessary. Also note that using this mode, the built container image **will** show up when executing `docker images`.

Docker

The extension `quarkus-container-image-docker` is using the Docker binary and the generated Dockerfiles under `src/main/docker` in order to perform Docker builds.

To use this feature, add the following extension to your project.

```
./mvnw quarkus:add-extension -Dextensions="container-image-docker"
```

S2I

The extension `quarkus-container-image-s2i` is using S2I binary builds in order to perform container builds inside the OpenShift cluster. The idea behind the binary build is that you just upload the artifact and its dependencies to the cluster and during the build they will be merged to a builder image (defaults to `fabric8/s2i-java`).

The benefit of this approach, is that it can be combined with OpenShift's `DeploymentConfig` that makes it easy to roll out changes to the cluster.

To use this feature, add the following extension to your project.

```
./mvnw quarkus:add-extension -Dextensions="container-image-s2i"
```

S2I builds require creating a `BuildConfig` and two `ImageStream` resources, one for the builder image and one for the output image. The creation of such objects is being taken care of by the Quarkus Kubernetes extension.

Building

To build a container image for your project, `quarkus.container-image.build=true` needs to be set using any of the ways that Quarkus supports.

```
./mvnw clean package -Dquarkus.container-image.build=true
```

Pushing

To push a container image for your project, `quarkus.container-image.push=true` needs to be set using any of the ways that Quarkus supports.

```
./mvnw clean package -Dquarkus.container-image.push=true
```

If no registry is set (using `quarkus.container-image.registry`) then `docker.io` will be used as the default.

Customizing

The following properties can be used to customize the container image build process.

Container Image Options

🔒 Configuration property fixed at build time - All other configuration properties are overridable at runtime

Configuration property	Type	Default
🔒 `quarkus.container-image.group` The group the container image will be part of	string	`${user.name}`
🔒 `quarkus.container-image.name` The name of the container image. If not set defaults to the application name	string	`${quarkus.application.name:unset}`
🔒 `quarkus.container-image.tag` The tag of the container image. If not set defaults to the application version	string	`${quarkus.application.version:latest}`
🔒 `quarkus.container-image.registry` The container registry to use	string	
🔒 `quarkus.container-image.username` The username to use to authenticate with the registry where the built image will be pushed	string	
🔒 `quarkus.container-image.password` The password to use to authenticate with the registry where the built image will be pushed	string	
🔒 `quarkus.container-image.insecure` Whether or not insecure registries are allowed	boolean	`false`
🔒 `quarkus.container-image.build` Whether or not a image build will be performed.	boolean	`false`
🔒 `quarkus.container-image.push` Whether or not an image push will be performed.	boolean	`false`

Jib Options

In addition to the generic container image options, the `container-image-jib` also provides the following options:

🔒 Configuration property fixed at build time - All other configuration properties are overridable at runtime

Configuration property	Type	Default
🔒 `quarkus.jib.base-jvm-image` The base image to be used when a container image is being produced for the jar build	string	`fabric8/java-alpine-openjdk11-jre`
🔒 `quarkus.jib.base-native-image` The base image to be used when a container image is being produced for the native binary build	string	`registry.access.redhat.com/ubi8/ubi-minimal`
🔒 `quarkus.jib.jvm-arguments` Additional JVM arguments to pass to the JVM when starting the application	list of string	`-Dquarkus.http.host=0.0.0.0,-Djava.util.logging.manager=org.jboss.logmanager.LogManager`
🔒 `quarkus.jib.native-arguments` Additional arguments to pass when starting the native application	list of string	`-Dquarkus.http.host=0.0.0.0`

🔒 `quarkus.jib.base-registry-username` The username to use to authenticate with the registry used to pull the base JVM image	string	
🔒 `quarkus.jib.base-registry-password` The password to use to authenticate with the registry used to pull the base JVM image	string	
🔒 `quarkus.jib.environment-variables` Environment variables to add to the container image	`Map<String,String>`	required ❗
🔒 `quarkus.jib.labels` Custom labels to add to the generated image	`Map<String,String>`	required ❗

Docker Options

In addition to the generic container image options, the `container-image-docker` also provides the following options:

🔒 Configuration property fixed at build time - All other configuration properties are overridable at runtime

Configuration property	Type	Default
🔒 `quarkus.docker.dockerfile-jvm-path` Path to the the JVM Dockerfile. If not set ${project.root}/src/main/docker/Dockerfile.jvm will be used If set to an absolute path then the absolute path will be used, otherwise the path will be considered relative to the project root	string	
🔒 `quarkus.docker.dockerfile-native-path` Path to the the JVM Dockerfile. If not set ${project.root}/src/main/docker/Dockerfile.native will be used If set to an absolute path then the absolute path will be used, otherwise the path will be considered relative to the project root	string	

S2I Options

In addition to the generic container image options, the `container-image-s2i` also provides the following options:

🔒 Configuration property fixed at build time - All other configuration properties are overridable at runtime

Configuration property	Type	Default
🔒 `quarkus.s2i.base-jvm-image` The base image to be used when a container image is being produced for the jar build	string	`fabric8/s2i-java:2.3`
🔒 `quarkus.s2i.base-native-image` The base image to be used when a container image is being produced for the native binary build	string	`quay.io/quarkus/ubi-quarkus-native-binary-s2i:19.3.0`
🔒 `quarkus.s2i.jvm-arguments` Additional JVM arguments to pass to the JVM when starting the application	list of string	`-Dquarkus.http.host=0.0.0.0,-Djava.util.logging.manager=org.jboss.logmanager.LogManager`
🔒 `quarkus.s2i.native-arguments` Additional arguments to pass when starting the native application	list of string	`-Dquarkus.http.host=0.0.0.0`
🔒 `quarkus.s2i.jar-directory` The directory where the jar is added during the assemble phase. This is dependant on the s2i image and should be supplied if a non default image is used.	string	`/deployments/`

🔒 `quarkus.s2i.jar-file-name` The resulting filename of the jar in the s2i image. This option may be used if the selected s2i image uses a fixed name for the jar.	string	
🔒 `quarkus.s2i.native-binary-directory` The directory where the native binary is added during the assemble phase. This is dependant on the s2i image and should be supplied if a non-default image is used.	string	`/home/quarkus/`
🔒 `quarkus.s2i.native-binary-file-name` The resulting filename of the native binary in the s2i image. This option may be used if the selected s2i image uses a fixed name for the native binary.	string	
🔒 `quarkus.s2i.build-timeout` The build timeout.	Duration ?	`PT5M`

About the Duration format

The format for durations uses the standard `java.time.Duration` format. You can learn more about it in the Duration#parse() javadoc.

You can also provide duration values starting with a number. In this case, if the value consists only of a number, the converter treats the value as seconds. Otherwise, `PT` is implicitly prepended to the value to obtain a standard `java.time.Duration` format.