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 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 image extension, you just need to add the following dependency into your project.

```
    <dependency>
      <groupId>io.quarkus</groupId>
      <artifactId>quarkus-container-image-jib</artifactId>
    </dependency>
```

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 image extension, you just need to add the following dependency into your project.

```
    <dependency>
      <groupId>io.quarkus</groupId>
      <artifactId>quarkus-container-image-docker</artifactId>
    </dependency>
```

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 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 rollout changes to the cluster.

To use this image extension, you just need to add the following dependency into your project.

```
    <dependency>
      <groupId>io.quarkus</groupId>
      <artifactId>quarkus-container-image-s2i</artifactId>
    </dependency>
```

S2i builds, require creating a `BuildConfig` and two `ImageStream` resources, one of 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 build 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	string	
🔒 `quarkus.container-image.password` The password to use to authenticate with the registry	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:

Docker Options

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

S2I Options

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