3.2 – Contenedores y Orquestadores

Tema 1 - Docker con Quarkus

Universidad Rey Juan Carlos

Micael Gallego
micael.gallego@urjc.es
@micael_gallego

code URJC

Google jib

- Para ciertas aplicaciones de **tipos concretos** se han creado herramientas más optimizadas para crear las imágenes Docker
- **jib** es un plugin de Maven y Gradle desarrollado por Google que empaqueta aplicaciones Java directamente como contenedores Docker (sin pasar por un .jar)
- Las capas optimizadas para cachear librerías
- Al no generar el .jar envía sólo los .class de la aplicación

Google jib

- **jib** es un **plugin de Maven y Gradle** que empaqueta aplicaciones Java directamente como contenedores Docker (**sin generar el .jar**)
- Las capas **optimizadas** para cachear librerías
- Al no generar el .jar **envía sólo los .class** de la aplicación (muy poco tamaño > poco tiempo de transferencia)
- La aplicación **arranca más rápido** (*exploded jar*)

https://github.com/GoogleContainerTools/jib

Google jib

- jib **no necesita el docker engine** para generar las imágenes Docker, todo lo hace con Java
- **Aumenta la seguridad** en el entorno de CI porque no necesita permisos de administración (necesarios para Docker) para crear una imagen
- Para Quarkus debemos añadir la extensión específica que nos indican
- Por defecto se utiliza la imagen base **ubi8/openjdk-17**

https://quarkus.io/guides/container-image#jib

https://catalog.redhat.com/software/containers/ubi8/openjdk-17/618bdbf34ae3739687568813

Quarkus y Google jib

- **Crear una imagen utilizando jib**
  - Debemos añadir la dependencia de jib al proyecto Quarkus

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

  - Indicamos en el **application.properties** la creación de la imagen, definimos el nombre y habilitamos la subida automática a un registro remoto

```
quarkus.container-image.build=true
quarkus.container-image.group=miusuario
quarkus.container-image.name=repositorio
quarkus.container-image.tag=version
quarkus.container-image.push=true
```

Quarkus y Google jib

- **Crear una imagen utilizando jib**
  - Para crear la imagen ejecutamos

```
$ ./mvnw install
```

- Por defecto Quarkus compila la imagen utilizando el formato **fast-jar**, aunque podemos indicar que utilice el formato **uber-jar** en el **application.properties**

```
quarkus.package.type=uber-jar
quarkus.container-image.build=true
quarkus.container-image.group=miusuario
quarkus.container-image.name=repositorio
quarkus.container-image.tag=version
quarkus.container-image.push=true
```

Quarkus y Google jib

- **Crear una imagen utilizando jib**
  - Al igual que en Spring Boot podremos almacenar la imagen en el **docker engine** local sin pasar por un registro remoto
  - Para ello tendremos que cambiar la propiedad **quarkus.container-image.push** a **false**

```
quarkus.container-image.build=true
quarkus.container-image.group=miusuario
quarkus.container-image.name=repositorio
quarkus.container-image.tag=version
quarkus.container-image.push=false
```

Ejercicio 1 Quarkus y Google jib

- **Crea una imagen Docker con jib**
  - Utiliza la aplicación **"aplicacion-quarkus-enunciado"**

https://github.com/MasterCloudApps/3.2.Contenedores-y-orquestadores/tree/master/docker/aplicacion-quarkus-enunciado

Quarkus y Docker

- Ficheros **Dockerfile** proporcionados en la ruta **"src/main/docker"**
- Los ficheros están optimizados para crear una imagen de la aplicación
- Construcción con **fast-jar** o **imagen nativa**
- Es necesario **Docker** para crear la imagen
- Por defecto se utilizan las imágenes **ubi8/openjdk-17** y **ubi8/ubi-minimal:8.6**

https://quarkus.io/guides/container-image#docker

https://catalog.redhat.com/software/containers/ubi8/openjdk-17/618bdbf34ae3739687568813

https://catalog.redhat.com/software/containers/ubi8/ubi-minimal/5c359a62bed8bd75a2c3fba8

Quarkus y Docker

- **Crear una imagen utilizando Docker**
  - Debemos añadir la dependencia de Docker al proyecto Quarkus

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

  - Indicamos en el **application.properties** la creación de la imagen y definimos el nombre

```
quarkus.container-image.build=true
quarkus.container-image.group=miusuario
quarkus.container-image.name=repositorio
quarkus.container-image.tag=version
```

Quarkus y Docker

- **Crear una imagen utilizando Docker**
  - Para crear la imagen ejecutamos

```
$ ./mvnw install
```

- Al igual que con **jib**, podremos indicar a Quarkus que publique automáticamente la imagen en un repositorio remoto con la propiedad **quarkus.container-image.push** puesta a **true**

```
quarkus.container-image.build=true
quarkus.container-image.group=miusuario
quarkus.container-image.name=repositorio
quarkus.container-image.tag=version
quarkus.container-image.push=true
```

Quarkus y Docker

- **Crear una imagen GraalVM native utilizando Docker**
  - Quarkus trae por defecto habilitado el soporte para compilación nativa
  - Para crear una imagen nativa utilizaremos los mismos pasos vistos anteriormente (añadir el **plugin de Docker** al proyecto y definir las **propiedades** necesarias en el fichero **application.properties**)
  - Ejecutamos el siguiente comando

```
$ ./mvnw install -Dnative
```

Ejercicio 2 Quarkus y Docker

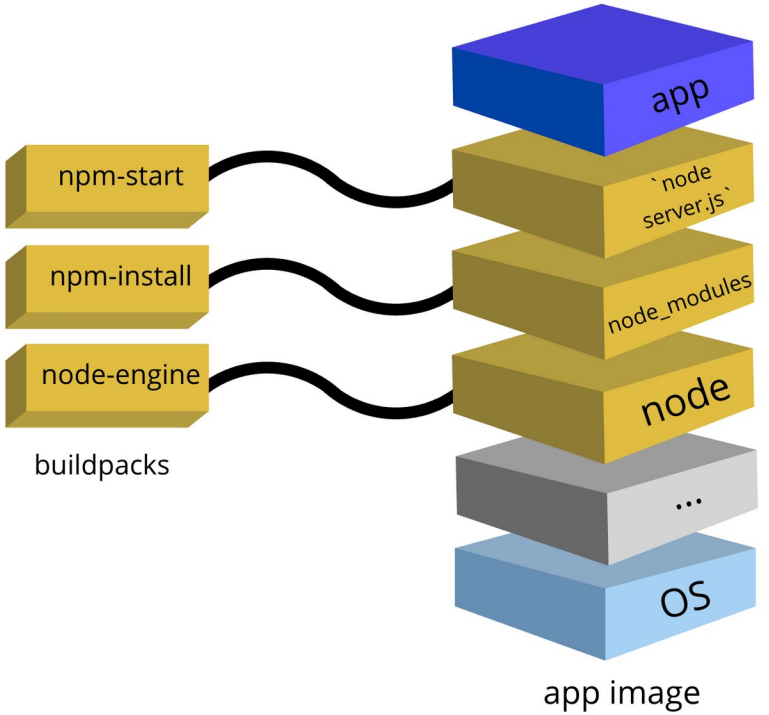
- **Crea una imagen Docker**
  - Utiliza la aplicación **"aplicacion-quarkus-enunciado"**

https://github.com/MasterCloudApps/3.2.Contenedores-y-orquestadores/tree/master/docker/aplicacion-quarkus-enunciado

Buildpacks

- Traduce el **código** fuente a **imagénes**
- No hay que utilizar Dockerfiles
- Cacheo de capas de forma optimizada
- Multilenguaje
- Imagenes minimas
- Builder y builpacks
- **Pack** CLI utiliza **Docker**

Buildpacks.io



https://buildpacks.io/
https://buildpacks.io/docs/tools/pack/

Paketo

paketo
buildpacks

- Implementación de Buildpacks
- Nos proporciona diferentes Builders para diferentes lenguajes (Java, Node, python, Go…)
- Existen diferentes implementaciones de otros proveedores (Google, Heroku)

https://paketo.io/

https://cloud.google.com/docs/buildpacks

https://github.com/heroku/builder

Quarkus y Buildpacks

- **Crear una imagen utilizando Buildpacks**
  - Debemos añadir la dependencia de Buildpack al proyecto Quarkus

```
$ ./mvnw quarkus:add-extension -Dextensions='container-image-buildpack'
```

  - Indicamos en el **application.properties** la creación de la imagen y definimos el nombre

```
quarkus.container-image.build=true
quarkus.container-image.group=miusuario
quarkus.container-image.name=repositorio
quarkus.container-image.tag=version
```

Quarkus y Buildpacks

- **Crear una imagen utilizando Buildpacks**
  - Para crear la imagen ejecutamos

```
$ ./mvnw install
```

Actualmente **buildpacks** en Quarkus sólo soporta la versión de **Java 11,** no es posible crear una imagen de un proyecto con **Java 17** utilizando este método

Quarkus y Buildpacks

- **Crear una imagen utilizando Buildpacks**
  - Al igual que con **jib y Docker**, podremos indicar a Quarkus que publique automáticamente la imagen en un repositorio remoto con la propiedad **quarkus.container-image.push** puesta a **true**

```
quarkus.container-image.build=true
quarkus.container-image.group=miusuario
quarkus.container-image.name=repositorio
quarkus.container-image.tag=version
quarkus.container-image.push=true
```

Quarkus y Buildpacks

- **Crear una imagen GraalVM native utilizando Buildpacks**
  - Quarkus trae por defecto habilitado el soporte para compilación nativa
  - Para crear una imagen nativa utilizaremos los mismos pasos vistos anteriormente (añadir el **plugin de Buildpack** al proyecto y definir las **propiedades** necesarias en el fichero **application.properties**)
  - Ejecutamos el siguiente comando

```
$ ./mvnw install -Dnative
```