Departamento de Ciencias de la Computación(DCCO)

Carrera de Ingeniería de Software

Curso de Aplicaciones Distribuidas

Contenerización y Despliegue

Presentado por: Marley Almeida, Sebastian Bolaños, Nicole Lara, Axel Pullaguari

Tutor: Morales, Dario.

Ciudad: Sangolquí, Ecuador

Fecha: 01/02/2025

Contenido

4. Conteneriazión de Docker

4.1. Contenerización con Docker

La contenerización con Docker es una tecnología que permite empaquetar aplicaciones y sus dependencias en un entorno aislado llamado contenedor. Esto facilita la portabilidad, escalabilidad y consistencia en diferentes entornos, desde desarrollo hasta producción

4.1.1 Empleado

Para el servicio employee-service. Se revisan aspectos como el uso de imágenes base, la eficiencia del tamaño, la gestión de versiones y las mejores prácticas para optimizar el despliegue.

2. Construcción de la Imagen Docker
La imagen se generó utilizando el siguiente comando en el directorio del servicio:

docker build -t employee-service:latest .

El proceso de construcción muestra los siguientes detalles:

- **Base utilizada:** eclipse-temurin:17-jdk-alpine, que es una imagen oficial con JDK 17 en un entorno Alpine Linux, optimizado para producción.
- **Capas descargadas:** Se observa la descarga de múltiples capas (sha256) que corresponden al sistema base y librerías necesarias.
- **Archivo copiado:** employee-service-0.0.1-SNAPSHOT.jar es copiado dentro de la imagen, lo que sugiere que el servicio es una aplicación Java basada en Spring Boot o similar.

3. Gestión de la Imagen en Docker
Se observa que la imagen employee-service ha sido generada con los siguientes detalles:

- Tag: latest (sin especificar versión).
- Estado: Unused (no ha sido ejecutada).
- Tamaño: 389.3 MB.

4. Publicación en Docker Hub
La imagen ha sido subida al repositorio lynmom/employee-service en Docker Hub.
Tiene una única etiqueta (latest).
Se proporciona el comando para subir nuevas versiones:
docker push lynmom/employee-service:tag

4.1.2 Administrador

Para el servicio de autenticación (**auth-service**) en un entorno Windows PowerShell. Se describe la ejecución del comando docker build, el significado de sus argumentos y los pasos involucrados en la construcción de la imagen a partir de un Dockerfile.

2. Ejecución del Comando Docker Build El comando ejecutado en la terminal de Windows PowerShell es:

docker build -t auth-service:latest .

Este comando indica a Docker que construya una imagen a partir del Dockerfile ubicado en el directorio actual (.) y le asigne la etiqueta auth-service:latest.

3. Descripción del Proceso de Construcción Durante la ejecución del comando, se registran los siguientes pasos:

1. **Carga del Dockerfile**: Se detecta y transfiere el Dockerfile, con un tamaño de 34 bytes.
2. **Carga de metadatos de la imagen base**: Se recuperan los metadatos de la imagen base openjdk:17-jdk-slim, que proporciona un entorno ligero para ejecutar aplicaciones Java.
3. **Carga del contexto de construcción**: El contexto de construcción incluye todos los archivos necesarios para crear la imagen. En este caso, el tamaño del contexto es de 60.48MB.
4. **Ejecución de instrucciones del Dockerfile**:
    - FROM openjdk:17-jdk-slim: Especifica la imagen base, una versión optimizada de OpenJDK 17.
    - WORKDIR /app: Define /app como el directorio de trabajo dentro del contenedor.
    - COPY target/auth-service-0.0.1-SNAPSHOT.jar app.jar: Copia el archivo auth-service-0.0.1-SNAPSHOT.jar desde la carpeta target del host al contenedor, renombrándolo como app.jar.

5. Finalización de la Construcción Una vez ejecutadas todas las instrucciones del Dockerfile:

- Se exportan las capas de la imagen.
- Se genera una imagen con el identificador sha256:2649785575c...b7ca4.
- La imagen es etiquetada como auth-service:latest, lo que facilita su referencia en futuros despliegues.

4.1.3 Recursos Humanos

Para el servicio de asistencia (**attendance-service**) en un entorno **Docker Desktop en Windows**, se describe la ejecución del comando `docker build`, el significado de sus argumentos y los pasos involucrados en la construcción de la imagen a partir de un **Dockerfile**.

2. Ejecución del Comando Docker Build

El comando ejecutado en la terminal de **Windows PowerShell** es:

docker build -t attendance-service:latest .

- docker build: Inicia el proceso de construcción de una imagen Docker.
- -t attendance-service:latest: Asigna la etiqueta **latest** a la imagen generada.
- . (punto): Indica que el **Dockerfile** se encuentra en el directorio actual.

3. Descripción del Proceso de Construcción

Durante la ejecución del comando, se registran los siguientes pasos:

1. Carga del Dockerfile

- Se detecta el **Dockerfile** en el directorio actual.
- El archivo es transferido a Docker con un tamaño de **34 bytes**.

2. Carga de metadatos de la imagen base

- Se recuperan los metadatos de la imagen base **eclipse-temurin:17-jdk-alpine**.
- Esta imagen proporciona un entorno **ligero y optimizado** para ejecutar aplicaciones **Java 17**.

3. Carga del contexto de construcción

- El **contexto de construcción** incluye todos los archivos necesarios para crear la imagen.
- En este caso, el tamaño del contexto es de **60.48 MB**.

4. Ejecución de instrucciones del Dockerfile

1. **Imagen Base:**

   FROM eclipse-temurin:17-jdk-alpine

   - Se usa una imagen de **Eclipse Temurin (Adoptium) con Alpine Linux**, que ofrece una implementación optimizada de **OpenJDK 17**.

2. **Definición del Directorio de Trabajo:**

   WORKDIR /app

   - Define /app como el **directorio de trabajo** dentro del contenedor.
   - Todos los comandos siguientes se ejecutarán en esta ubicación.

3. **Copia del Archivo JAR al Contenedor:**

   COPY target/attendance-service-0.0.1-SNAPSHOT.jar app.jar

   - Copia el archivo attendance-service-0.0.1-SNAPSHOT.jar desde la carpeta target/ del host al contenedor.
   - Lo renombra como app.jar dentro del contenedor.

Una vez ejecutadas todas las instrucciones del **Dockerfile**, se generan las capas de la imagen y se registran los siguientes eventos:

Exportación de capas:

- Se generan **múltiples capas SHA256** correspondientes a la imagen base y a los archivos copiados.

Generación de la imagen final:

- Se crea una imagen con el identificador único:

  sha256:a7c0de485618dc1d2f3512c5110f746da11b2dde05ad4a6f4471

- El tamaño final de la imagen es de **464.92 MB**.

Etiquetado de la imagen:

- La imagen es etiquetada como attendance-service:latest, lo que facilita su referencia en futuros despliegues y ejecuciones con Docker

4.2. Despliegue en Azure

Se ha configurado el despliegue de una aplicación web en Azure App Services utilizando **Azure Container Registry (ACR)** como origen de imagen.

Configuración de la imagen:

- **Registro de contenedor:** mirecurso
- **Método de autenticación:** Identidad administrada (**Managed identity**)
- **Identidad asignada:** (Nuevo) ua-id-8aab
- **Imagen:** mirecurso.azurecr.io/mirecurso/employee-service:latest
- **Etiqueta de la imagen:** latest

2. Guía de Despliegue en Azure

Paso 1: Crear un Azure Container Registry (ACR)

Si aún no tienes un ACR, sigue estos pasos:

1. Configura los siguientes parámetros:
    - **Nombre:** mirecurso
    - **Suscripción:** Selecciona la suscripción activa.
    - **Grupo de recursos:** Crea uno nuevo o usa uno existente.
    - **Ubicación:** Selecciona la más cercana a tus usuarios.
    - **SKU:** Basic o superior.
2. Clic en **Revisar y crear**, luego **Crear**.

Paso 2: Construcción y carga de la imagen en ACR

1. Inicia sesión en Azure con CLI:

   az login

2. Autenticar con ACR:

   az acr login --name mirecurso

3. Construir la imagen Docker:

   docker build -t mirecurso.azurecr.io/mirecurso/employee-service:latest .

4. Subir la imagen a ACR:

   docker push mirecurso.azurecr.io/mirecurso/employee-service:latest

Paso 3: Crear y Configurar la Aplicación Web en Azure

1. Seleccionamos el grupo de recursos y asigna un nombre a la aplicación.
2. En la sección **Origen de imagen**, selecciona **Azure Container Registry**.
3. Configuramos los siguientes parámetros:
    - **Registro:** mirecurso
    - **Autenticación:** Managed Identity
    - **Imagen:** mirecurso.azurecr.io/mirecurso/employee-service:latest
    - **Etiqueta:** latest

4. Clic en **Revisar y crear** y luego **Crear**.