

Docker y Kubernetes Intermedio



Argentina • Chile • Colombia • México • Perú

www.netec.com | servicio@netec.com



Propiedad Intelectual

Material didáctico preparado por la empresa Global K, S.A. de C.V. Registrado en Derechos de Autor.

Todos los contenidos de este Sitio (incluyendo, pero no limitado a: texto, logotipos, contenido, fotografías, audio, botones, nombres comerciales y videos) están sujetos a derechos de propiedad por las leyes de Derechos de Autor de la empresa Global K, S.A. de C.V.

Queda prohibido copiar, reproducir, distribuir, publicar, transmitir, difundir, o en cualquier modo explotar cualquier parte de este documento sin la autorización previa por escrito de Global K, S.A. de C.V. o de los titulares correspondientes.

Descripción del Curso

El curso intermedio de Docker y Kubernetes cubre la gestión de redes y persistencia en Docker, la orquestación de aplicaciones multi-contenedor con Docker Compose, y la configuración de aplicaciones en Kubernetes utilizando ConfigMaps y Secrets. Incluye el despliegue de aplicaciones Spring en Kubernetes con balanceo de carga y configuración de entornos, así como la integración de microservicios mediante Spring Cloud Gateway para optimizar la comunicación en entornos distribuidos.



Objetivos del Curso

Al finalizar el curso, serás capaz de:

- Comprender y configurar redes de Docker para permitir la comunicación entre contenedores y resolver problemas de persistencia de datos en bases de datos Dockerizadas.
- Dockerizar microservicios y bases de datos Oracle, aplicando redes y volúmenes para gestionar la comunicación y la persistencia de datos entre contenedores.
- Utilizar Docker Compose para definir y ejecutar aplicaciones multi-contenedor, incluyendo la creación, configuración y gestión de microservicios y bases de datos en un solo archivo.
- Gestionar, ConfigMaps y Secrets en Kubernetes para gestionar configuraciones y credenciales de manera centralizada en aplicaciones distribuidas.



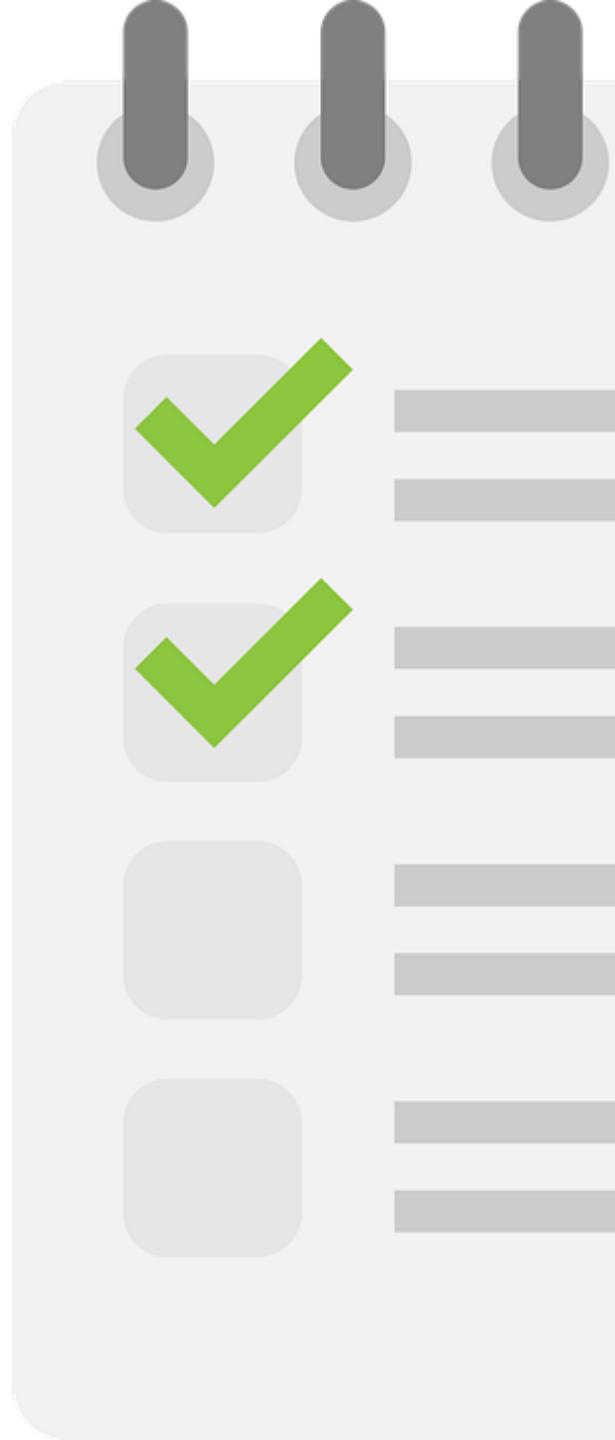
Objetivos del Curso

Al finalizar el curso, serás capaz de:

- Configurar y desplegar aplicaciones Spring en Kubernetes utilizando Spring Cloud, adaptando configuraciones específicas para entornos de desarrollo y producción.
- Implementar prácticas de balanceo de carga y monitoreo en Kubernetes mediante Liveness y Readiness Probes para mejorar la disponibilidad y resiliencia de los microservicios.
- Configurar Spring Cloud Gateway para gestionar rutas de microservicios en Kubernetes, optimizando el enrutamiento y la comunicación entre servicios.

Prerrequisitos

- Conocimientos básicos de Docker: Familiaridad con la creación de contenedores y Dockerfiles.
- Conocimientos de Kubernetes: Entender conceptos básicos como Pods, Deployments, Services, ConfigMaps y Secrets.
- Experiencia en línea de comandos: Habilidad para ejecutar comandos en Docker CLI y Kubernetes CLI (kubectl).
- Conceptos básicos de redes y comunicación entre servicios: Entender las redes de contenedores y cómo los servicios se comunican en un clúster de Kubernetes.



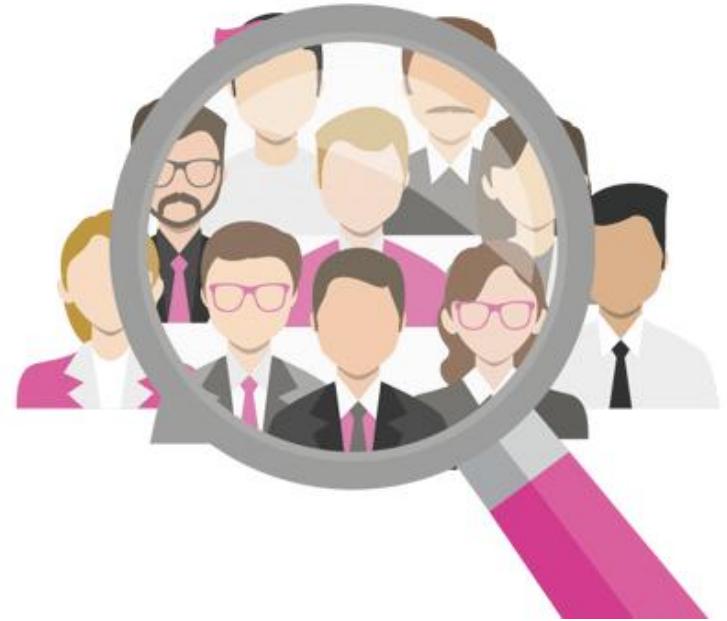
Prerrequisitos

- Conocimientos básicos en bases de datos relacionales y administración básica de Oracle Database.
- Conocimientos básicos en OOP, Java y Spring Framework.
- Conocimientos en Sistemas Operativos Linux y Windows.
- Conocimientos en básicos en Git.



Audiencia

- Este curso está dirigido a desarrolladores y profesionales de TI que desean profundizar en el despliegue, configuración y administración de aplicaciones en contenedores con Docker y Kubernetes.
- Es ideal para quienes ya tienen conocimientos básicos en estas tecnologías y buscan mejorar sus habilidades en entornos distribuidos, configuraciones centralizadas y administración de redes de contenedores.
- También está orientado a quienes trabajan con microservicios y necesitan manejar de manera eficiente la comunicación y configuración de aplicaciones en entornos de producción.



Temario

- **Unidad 1.** Docker Networks: Comunicación entre contenedores
- **Unidad 2.** Docker Compose: Orquestador para definir y ejecutar multi-contenedores
- **Unidad 3.** Kubernetes: ConfigMap y Secret
- **Unidad 4.** Kubernetes: Spring Kubernetes
- **Unidad 5.** Kubernetes: Spring Cloud Gateway

Presentación del grupo

- ¿Cuál es tu nombre?
- ¿Cuál es tu experiencia en desarrollo?
- ¿Qué tecnología/idea/software te ha impresionado?
- ¿Cuáles son tus expectativas respecto al curso?





Unidad 1

Docker Networks: Comunicación entre contenedores

Objetivos:

- Comprender y configurar redes de Docker para facilitar la comunicación entre contenedores.
- Dockerizar microservicios y bases de datos Oracle, aplicando configuraciones de red que permitan la conectividad y el intercambio de datos.
- Resolver problemas de persistencia de datos en contenedores de bases de datos Dockerizadas mediante el uso de volúmenes.
- Probar la comunicación entre contenedores utilizando herramientas de prueba como Postman.

1.1. Dockerizando microservicios, recursos y configurando la Red o Network

- ¿Cuál es la diferencia entre Red y Network?

"Red" es el término común en español que describe la conexión entre dispositivos para compartir datos o recursos. En Docker, "network" se refiere a una configuración específica que permite la comunicación entre contenedores. En otras palabras, "network" es la forma en que Docker gestiona y estructura esas conexiones o "redes"

Dockerizando | Microservicio

"Una arquitectura de microservicios es un enfoque en el que se construye una aplicación como un conjunto de servicios pequeños, autónomos y desplegables de forma independiente."

Martin Fowler

"Los microservicios son un estilo de arquitectura que estructura una aplicación como un conjunto de servicios que se pueden desplegar y escalar de manera independiente y que se comunican entre sí mediante API ligeras."

Sam Newman

La arquitectura de microservicios se basa en el concepto de desarrollar sistemas de software como conjuntos de servicios independientes y autónomos que son modelados alrededor de las capacidades de negocio de una organización."

James Lewis - Martin Fowler

- Independientes
- Flexibilidad
- Escalabilidad
- Tamaño

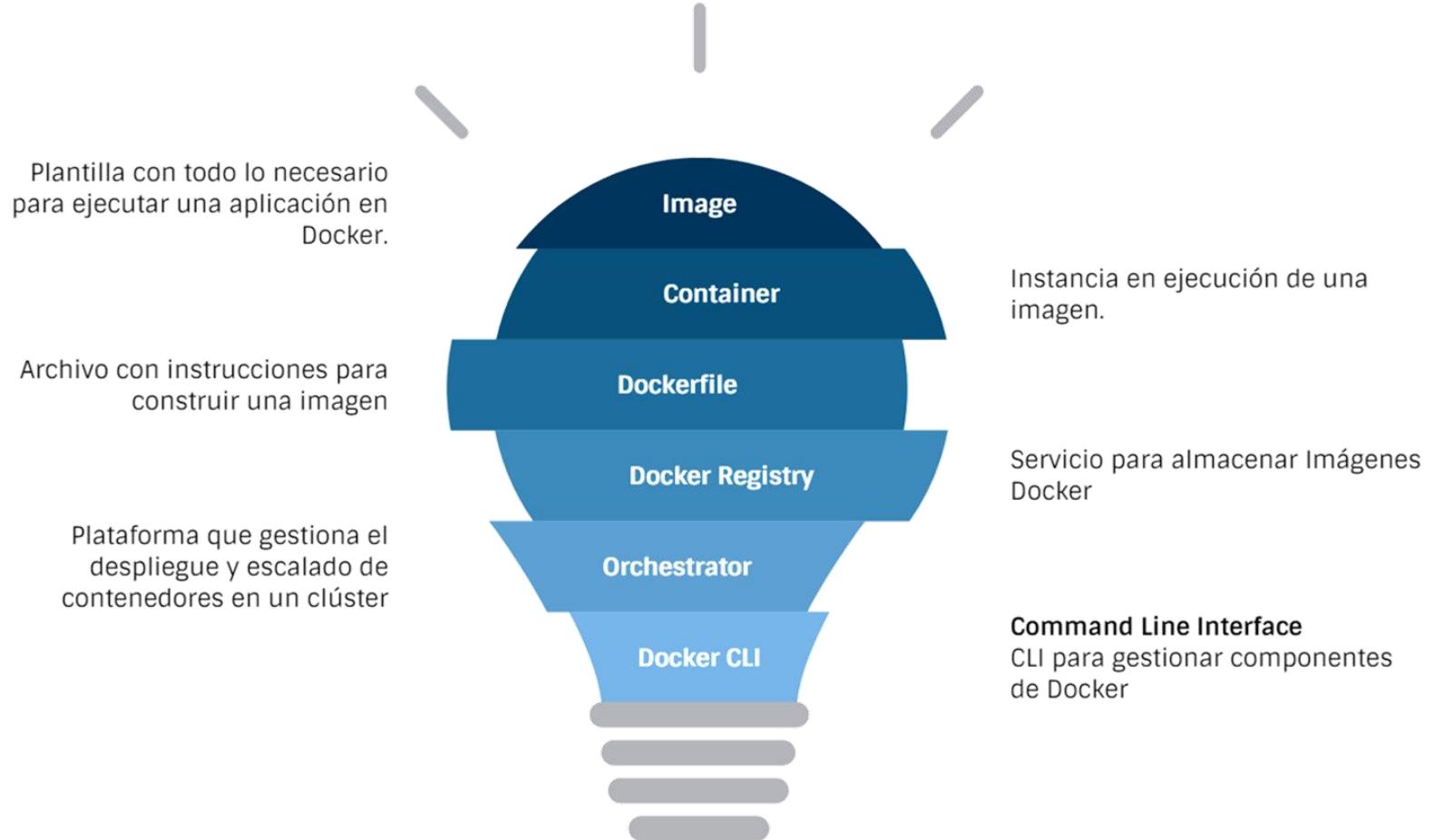
Dockerizando | Docker

- ¿Qué es Docker?

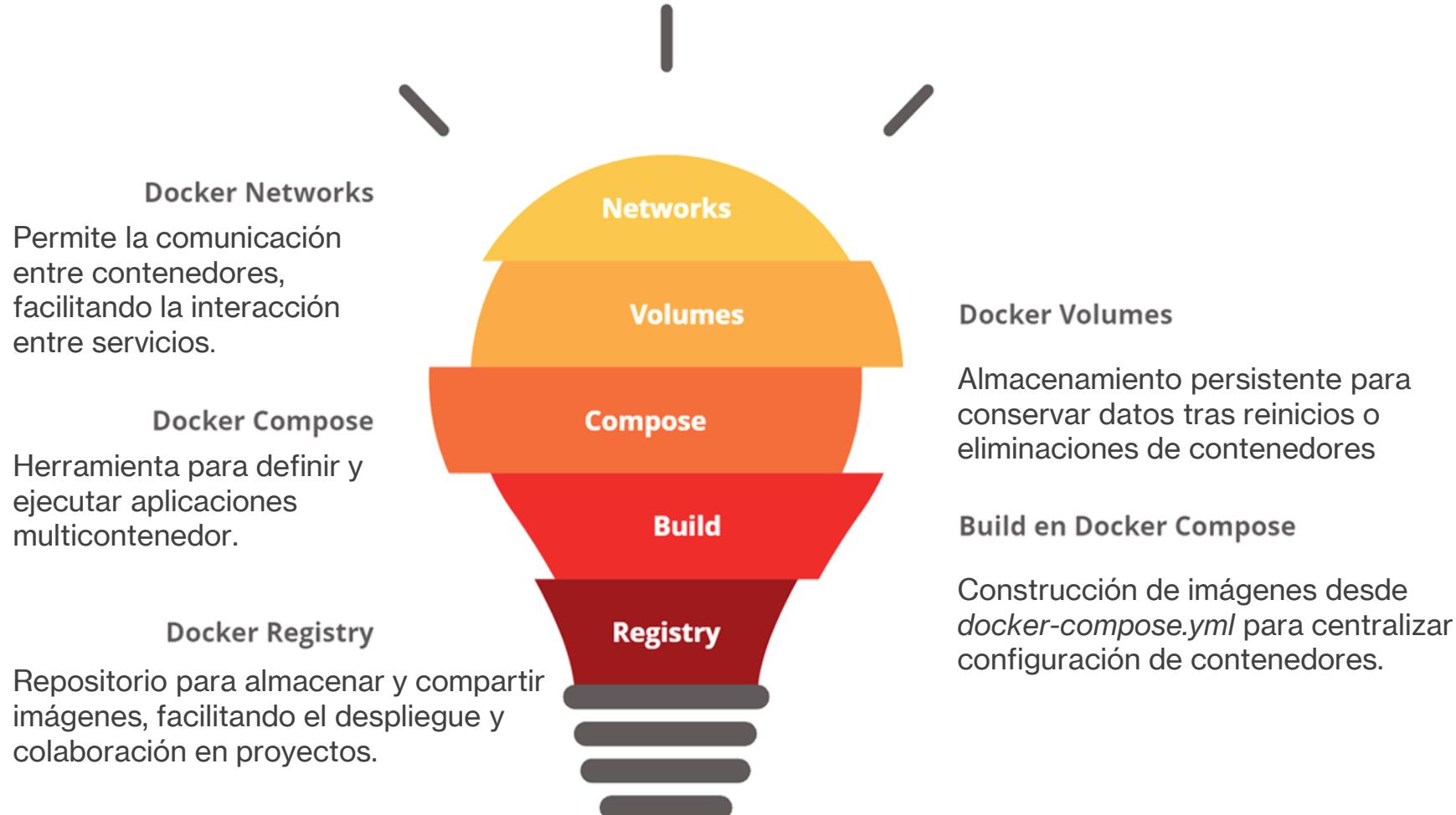
Docker es una plataforma de contenedorización que permite crear, desplegar y ejecutar aplicaciones en entornos aislados llamados **contenedores**, que incluyen todo lo necesario para funcionar de manera consistente en cualquier entorno, optimizando el desarrollo y despliegue de aplicaciones.

Pseudo-Virtualización

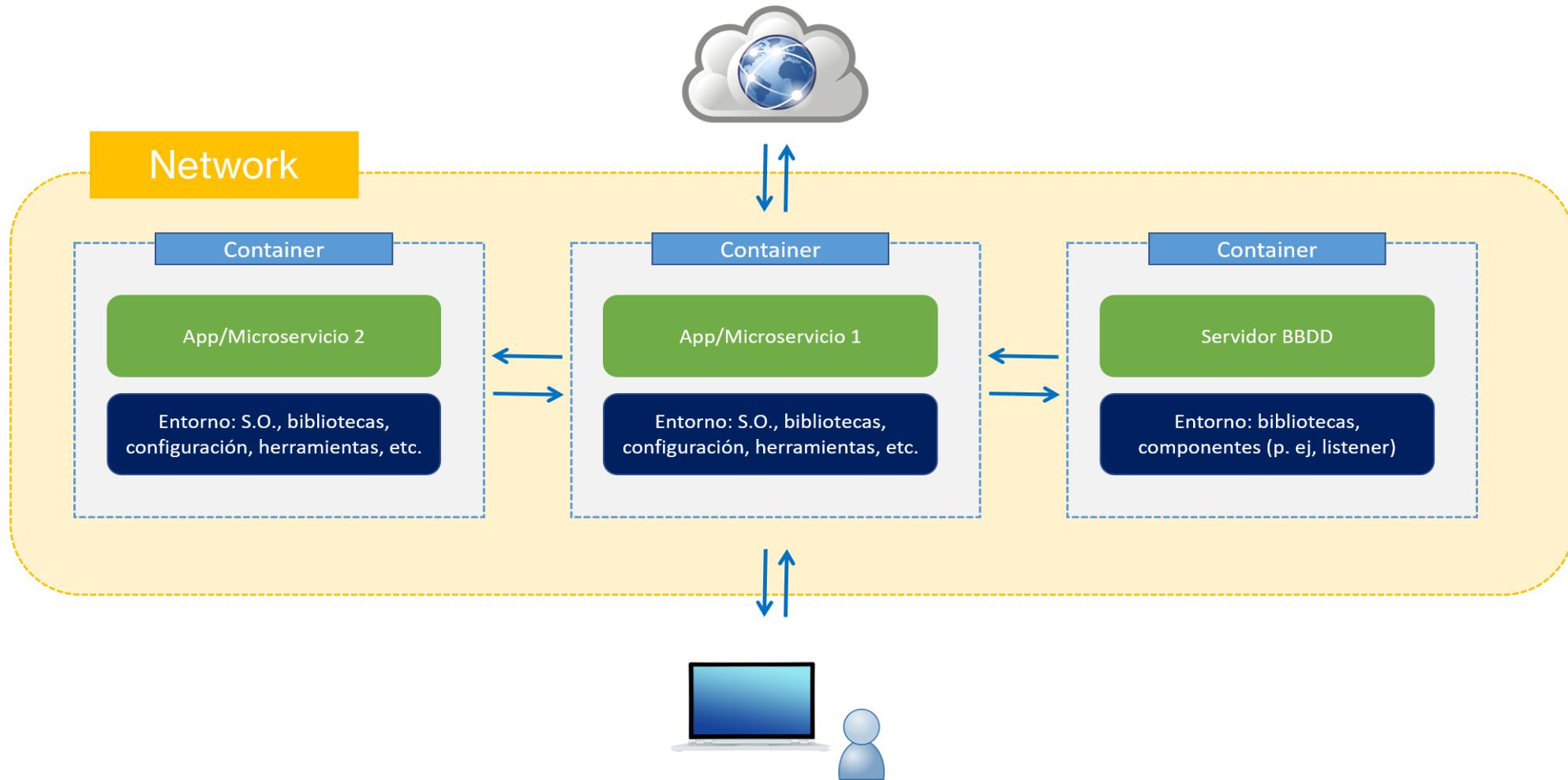
Dockerizando | Conceptos clave



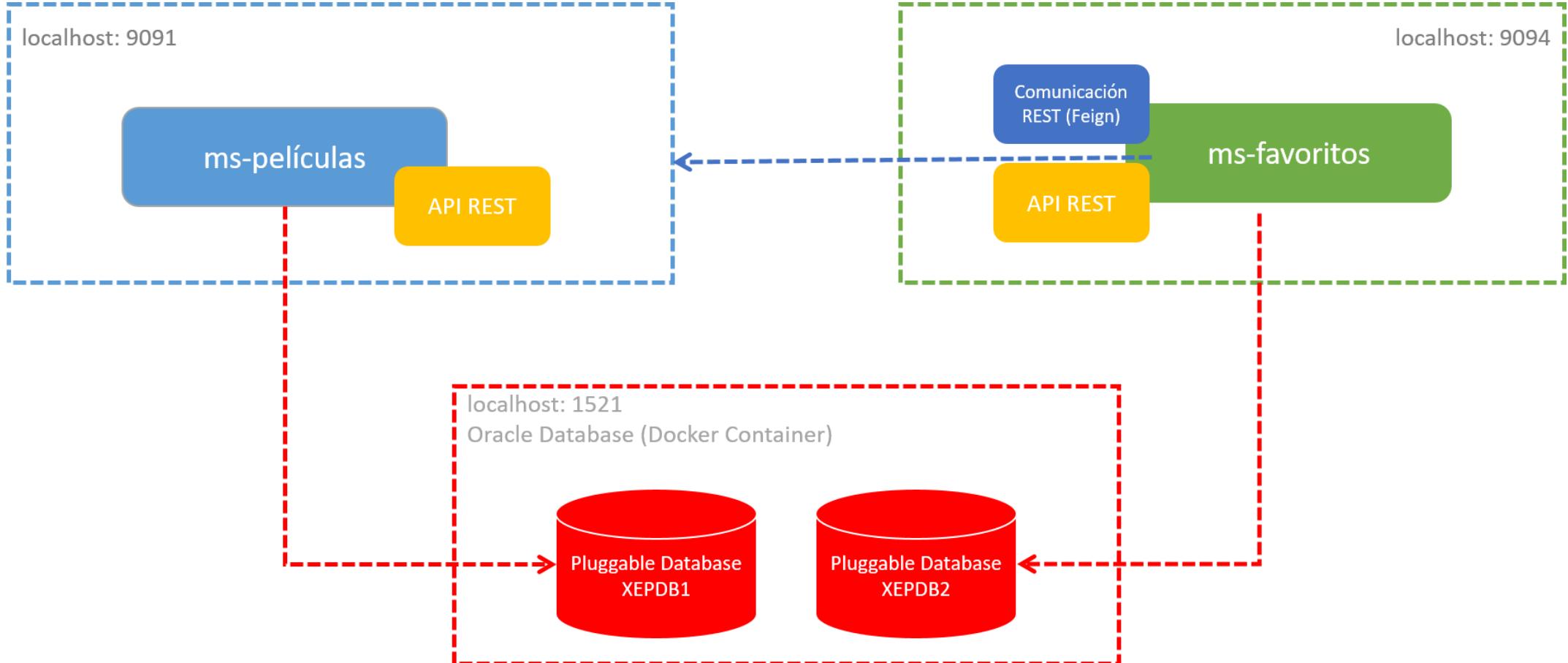
Dockerizando | Nuevos conceptos clave



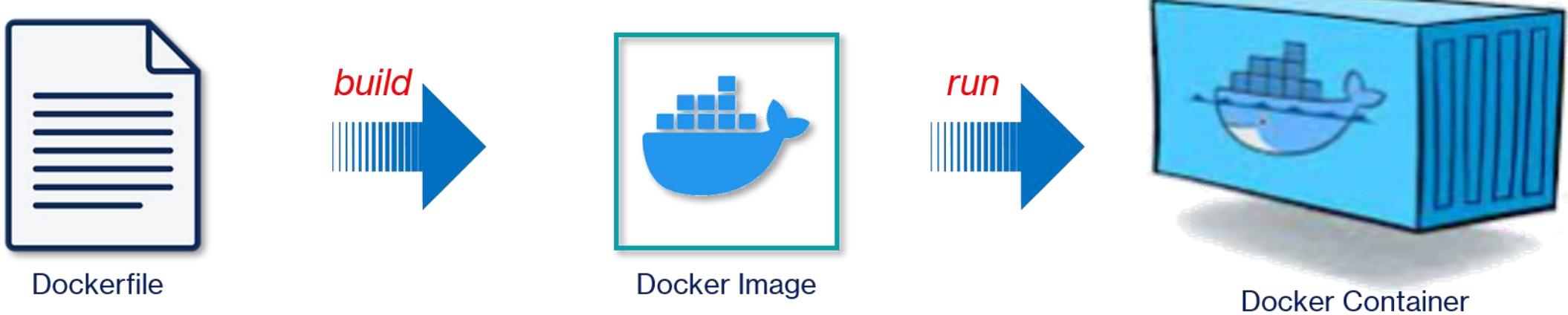
Dockerizando microservicios



Dockerizando | Caso de estudio



Dockerizando | Caso de estudio



Dockerizando microservicios | Caso de estudio

¿Qué elementos son necesarios incluir en un **Dockerfile** para poder construir una **Imagen Docker** funcional que ejecute los microservicios del caso de estudio correctamente?

FROM
WORKDIR
COPY
RUN
EXPOSE
ENTRYPOINT o CMD

Dockerizando microservicios | Caso de estudio

```
# 1. Imagen base
FROM openjdk:21-jdk-slim

# 2. Establecer el directorio de trabajo
WORKDIR /app

# 3. Copiar el archivo JAR generado
COPY target/ms-peliculas-0.0.1-SNAPSHOT.jar app.jar

# 4. Exponer el puerto utilizado por el microservicio
EXPOSE 9091

# 5. Comando de inicio
ENTRYPOINT ["java", "-jar", "app.jar"]
```

Dockerizando microservicios | Caso de estudio

```
# 1. Imagen base
FROM openjdk:21-jdk-slim

# 2. Establecer el directorio de trabajo
WORKDIR /app

# 3. Copiar el archivo JAR generado
COPY target/ms-favoritos-0.0.1-SNAPSHOT.jar app.jar

# 4. Exponer el puerto utilizado por el microservicio
EXPOSE 9094

# 5. Comando de inicio
ENTRYPOINT ["java", "-jar", "app.jar"]
```

.\mvnw o mvn
JAR

Dockerizando microservicios | Caso de estudio

```
C:\00_Material_Docker_Kubernetes\ws2\ms-peliculas>ls -l
total 33
-rw-r--r-- 1 Netec 197121 290 Nov 16 10:07 Dockerfile
-rw-r--r-- 1 Netec 197121 1685 Nov 15 12:43 HELP.md
-rwxr-xr-x 1 Netec 197121 10665 Nov 15 12:43 mvnw
-rw-r--r-- 1 Netec 197121 6912 Nov 15 12:43 mvnw.cmd
-rw-r--r-- 1 Netec 197121 2398 Nov 15 15:52 pom.xml
drwxr-xr-x 1 Netec 197121 0 Nov 15 12:43 src
drwxr-xr-x 1 Netec 197121 0 Nov 16 10:07 target

C:\00_Material_Docker_Kubernetes\ws2\ms-peliculas>.\mvnw clean install -Dmaven.test.skip=true
[INFO] Scanning for projects...
[INFO]
[INFO] -----< com.netec:ms-peliculas >-----
[INFO] Building ms-peliculas 0.0.1-SNAPSHOT
[INFO]   from pom.xml
[INFO] -----[ jar ]-----
[INFO]
[INFO] --- clean:3.3.2:clean (default-clean) @ ms-peliculas ---
[INFO] Deleting C:\00_Material_Docker_Kubernetes\ws2\ms-peliculas\target
[INFO]
[INFO] --- resources:3.3.1:resources (default-resources) @ ms-peliculas ---
[INFO] Copying 1 resource from src\main\resources to target\classes
[INFO] Copying 0 resource from src\main\resources to target\classes
[INFO]
[INFO] --- compiler:3.13.0:compile (default-compile) @ ms-peliculas ---
[INFO] Recompiling the module because of changed source code.
[INFO] Compiling 6 source files with javac [debug parameters release 21] to target\classes
[INFO] Annotation processing is enabled because one or more processors were found
on the class path. A future release of javac may disable annotation processing
unless at least one processor is specified by name (-processor), or a search
path is specified (--processor-path, --processor-module-path), or annotation
processing is enabled explicitly (-proc:only, -proc:full).
Use -Xlint:-options to suppress this message.
Use -proc:none to disable annotation processing.
[INFO]
[INFO] --- resources:3.3.1:testResources (default-testResources) @ ms-peliculas ---
```

.\mvnw clean install -Dmaven.test.skip=true

Dockerizando microservicios | Caso de estudio

```
docker build -t <nombre_microsservicio>:<tag> .
```

```
docker run -p <puerto_host>:<puerto_contenedor> --name  
<nombre_contenedor> <nombre-imagen-Docker>:<tag>
```

Host ----> Contenedor

Dockerizando microservicios | Caso de estudio

```
C:\Símbolo del sistema - .\mvnw clean install - .\mvnw clean install -Dmaven.test.skip=true - .\mvnw clean install -Dmaven.test.skip=true

C:\00_Material_Docker_Kubernetes\ws2\ms-peliculas>
C:\00_Material_Docker_Kubernetes\ws2\ms-peliculas>docker images
REPOSITORY          TAG      IMAGE ID      CREATED       SIZE
ms-favoritos        1.0.0    856d178c1471  20 minutes ago  805MB
ms-peliculas         1.0.0    f7267c1b1530  29 minutes ago  787MB

C:\00_Material_Docker_Kubernetes\ws2\ms-peliculas>docker ps -a
CONTAINER ID   IMAGE           COMMAND                  CREATED          STATUS          PORTS     NAMES
4b04e3b46fc8   ms-peliculas:1.0.0   "java -jar app.jar"   About a minute ago   Exited (1) 39 seconds ago
6b57f8300856   ms-favoritos:1.0.0    "java -jar app.jar"   15 minutes ago    Exited (1) 14 minutes ago

C:\00_Material_Docker_Kubernetes\ws2\ms-peliculas>
```

1.2. Comunicación entre contenedores

- La **comunicación entre microservicios** es el proceso mediante el cual los microservicios que componen una aplicación intercambian datos o mensajes para colaborar y cumplir con las funcionalidades del sistema.
- A diferencia de las **arquitecturas monolíticas**, donde los componentes interactúan dentro de un único espacio de **memoria**, los microservicios interactúan a través de **redes**, utilizando protocolos estandarizados.
- Estas interacciones pueden ser **sincrónicas**, **asíncronas**, o ambas, dependiendo de los requisitos del sistema.

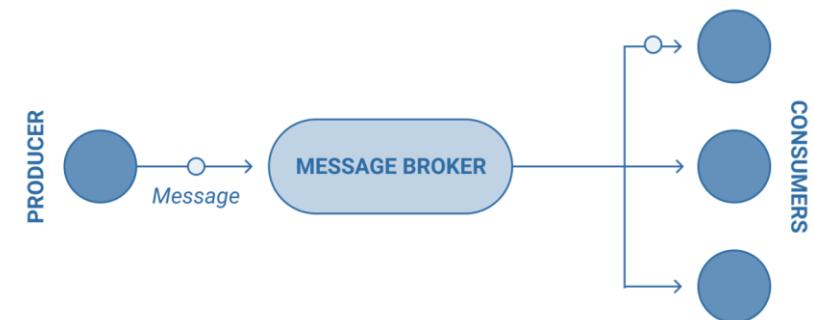
Comunicación entre contenedores

Comunicación síncrona

- Utiliza el protocolo HTTP y sigue principios REST.
- Ideal para sistemas sincrónicos, donde las solicitudes y respuestas son explícitas.
- Soporta verbos estándar como:
 - GET, POST, PUT, y DELETE
- Fácil de implementar y ampliamente adoptado en la industria.
- Algunas limitaciones: Mayor latencia debido al protocolo HTTP, menos eficiente para comunicación de alto volumen.

Comunicación asíncrona

- Los microservicios envían y reciben mensajes a través de un sistema de mensajería, desacoplando la comunicación.
- Ejemplos de sistemas:
 - RabbitMQ & Apache Kafka
- Beneficios: Escalabilidad, tolerancia a fallos, soporte para patrones pub/sub.
- Alguna Limitaciones: Mayor complejidad en la gestión de consistencia y diseño de patrones de mensajería.



Comunicación entre contenedores

Aspecto	Comunicación síncrona	Comunicación asíncrona
Definición	El emisor espera una respuesta del receptor para continuar.	El emisor no espera una respuesta inmediata; los mensajes se procesan de forma independiente.
Protocolos	HTTP/REST, gRPC (sincrónico).	RabbitMQ, Kafka, gRPC (asíncrono).
Ventajas	Simplicidad, flujo transaccional más intuitivo.	Desacoplamiento, mayor escalabilidad, menor impacto a fallos.
Desventajas	Mayor latencia, dependencia del estado del receptor.	Mayor complejidad en la gestión de consistencia eventual.
Casos de uso	Consultas de datos críticos, operaciones de negocio transaccionales.	Sistemas distribuidos con alta concurrencia, procesamiento por lotes.

Comunicación entre contenedores | CE

- En nuestro caso de estudio, donde tenemos múltiples microservicios interactuando entre sí, como **ms-películas** y **ms-favoritos**, configurarlos en la misma *red virtual* de Docker proporciona varias ventajas clave.

1. Facilita la comunicación directa.
2. Aislamiento y seguridad.
3. Reducción de latencia.
4. Escalabilidad.
5. Simplicidad en el despliegue y pruebas.

Docker crea un **DNS** interno en redes personalizadas. Esto permite a los contenedores comunicarse usando sus **nombres de servicio o contenedor** en lugar de direcciones IP estáticas.

Comunicación | Docker Network

```
docker network create [OPTIONS] NETWORK_NAME
```

- **NETWORK_NAME**: Nombre de la red que se quiere crear.
- **OPTIONS**: Opciones adicionales para personalizar la red.
 - **--driver DRIVER**: Especifica el controlador de red. Los controladores más comunes son:
 - **bridge**: Red local en el host, por defecto.
 - **overlay**: Red para servicios distribuidos, se requiere Swarm
 - **host**: usa la red del host, sin aislamiento de red,
 - **none**: Sin red, contenerización completamente aislada.
 - **--scope=local/global**: Define si la red es local al host o global (para Swarm).

Comunicación | Docker Network

```
# Crear una red virtual personalizada en Docker  
docker network create ms-network
```

```
# Ejecutar los microservicios en la misma red  
docker run --network microservices-network --name ms-peliculas -p 9091:9091  
ms-peliculas:1.0.0  
docker run --network microservices-network --name ms-favoritos -p 9093:9093  
ms-favoritos:1.0.0
```

- docker network ls
- docker network rm <NETWORK_NAME>
- docker network connect NETWORK_NAME CONTAINER_NAME
- docker network disconnect NETWORK_NAME CONTAINER_NAME

Comunicación | Microservicios Code

```
spring.application.name=ms-peliculas
server.port=9091

# Configuracion de la base de datos
spring.datasource.url=jdbc:oracle:thin:@localhost:1521/XEPDB1
spring.datasource.username=dkuser
spring.datasource.password=dkpassword
spring.datasource.driver-class-name=oracle.jdbc.OracleDriver

# Configuracion de JPA e Hibernate
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.OracleDialect
```

application.properties

Comunicación | Microservicios Code

```
spring.application.name=ms-favoritos
server.port=9094

# Configuración de la base de datos
spring.datasource.url=jdbc:oracle:thin:@localhost:1521/XEPDB2
spring.datasource.username=dkuser
spring.datasource.password=dkpassword
spring.datasource.driver-class-name=oracle.jdbc.OracleDriver

# Configuración de JPA e Hibernate
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.OracleDialect

# Perfil Docker
ms-peliculas.url=http://ms-peliculas:9091
```

application.properties

Comunicación | Microservicios Code

```
package com.netec.dk.feign;

import org.springframework.cloud.openfeign.FeignClient;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;

import com.netec.dk.model.PeliculaDTO;

@FeignClient(name = "ms-peliculas", url = "${ms-peliculas.url}")
public interface PeliculaClient {
    @GetMapping("/api/peliculas/{id}")
    PeliculaDTO obtenerPeliculaPorId(@PathVariable Long id);
}
```

PeliculaClient.java

Comunicación | Network Docker

```
C:\00_Material_Docker_Kubernetes\ws2\ms-favoritos>docker network create ms-network
4ebba80043273fdb5618d0585a2deb0d8a2101c259ae1af549f70c346da5747

C:\00_Material_Docker_Kubernetes\ws2\ms-favoritos>docker network ls
NETWORK ID      NAME      DRIVER      SCOPE
f753b405398d    bridge    bridge      local
90564692b97c    host      host       local
4ebba8004327    ms-network    bridge      local
c930d999554c    none      null       local

C:\00_Material_Docker_Kubernetes\ws2\ms-favoritos>docker network inspect ms-network
[
  {
    "Name": "ms-network",
    "Id": "4ebba80043273fdb5618d0585a2deb0d8a2101c259ae1af549f70c346da5747",
    "Created": "2024-11-16T21:52:15.729853312Z",
    "Scope": "local",
    "Driver": "bridge",
    "EnableIPv6": false,
    "IPAM": {
      "Driver": "default",
      "Options": {},
      "Config": [
        {
          "Subnet": "172.18.0.0/16",
          "Gateway": "172.18.0.1"
        }
      ]
    },
    "Internal": false,
    "Attachable": false,
    "Ingress": false,
    "ConfigFrom": {
      "Network": ""
    },
    "ConfigOnly": false,
    "Containers": {},
    "Options": {},
    "Labels": {}
  }
]
C:\00_Material_Docker_Kubernetes\ws2\ms-favoritos>
```

No hay contenedores conectados a la red ms-network., en este momento

Comunicación | Network Docker

Acción	Comando
Crear y conectar un conector a una red	<code>docker run --network <network_name> --name <container_name> -d <image_name></code>
Conectar un contenedor existente	<code>docker network connect <network_name> <container></code>
Desconectar un conector de una red	<code>docker network disconnect <network_name> <container></code>
Inspeccionar contenedores en una red	<code>docker network inspect <network_name></code>

Comunicación | Network Docker

```
C:\00_Material_Docker_Kubernetes\ws2\ms-favoritos>docker network inspect ms-network
[
    {
        "Name": "ms-network",
        "Id": "4ebba0043273fdb45618d0585a2deb0d8a2101c259ae1af549f70c346da5747",
        "Created": "2024-11-16T21:52:15.729853312Z",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": {},
            "Config": [
                {
                    "Subnet": "172.18.0.0/16",
                    "Gateway": "172.18.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {
            "4b04e3b46fc8a88c4b23f85b8682edc8a3ea2bcb9a741a2794b4bd1241df9b0f": {
                "Name": "ms-peliculas",
                "EndpointID": "12e6df2c6fa89aa097b6278e98465d75d24b2a341058faa67f4dca60acada381",
                "MacAddress": "02:42:ac:12:00:03",
                "IPv4Address": "172.18.0.3/16",
                "IPv6Address": ""
            },
            "d50a8ac2935af9b7363307463e345bdf0e80e42e5dfc5ec7ca937faa86cf0996": {
                "Name": "ms-favoritos",
                "EndpointID": "65a8ae024e3e95969080dcdaac63a949d6e252336ef12ec0546c660d30271819",
                "MacAddress": "02:42:ac:12:00:02",
                "IPv4Address": "172.18.0.2/16",
                "IPv6Address": ""
            }
        }
    }
]
```



ms-películas & ms-favoritos
en la red ms-network.

1.3. Dockerizando Oracle

Para Dockerizar Oracle se tienen varias opciones:

1. **Imagen Oficial de Oracle en Docker Hub o Oracle Container Registry**

- Ofrece imágenes oficiales en su propio registro de contenedores.
- Estás imágenes incluyen versiones como Oracle Database XE, Oracle 12c y 19c
- Las imágenes suelen ser grandes, por lo que debe de asegurarse de que el sistema anfitrión tenga suficiente espacio.
- Requiere aceptar los términos de la **licencia** antes de Poder descargar o utilizar las imágenes.



Dockerizando Oracle

2. Construcción de una imagen personalizada con los scripts de Oracle en GitHub.

- Oracle proporciona scripts en su repositorio oficial de GitHub para crear imágenes Docker de Oracle Database, lo cual no permite personalizar la imagen y elegir versiones específicas de Oracle.
- Requiere de más pasos y conocimientos técnicos.
 - Implica ejecutar y personalizar los scripts de construcción.
 - Es necesario descargar manualmente el software de Oracle desde su sitio web oficial, ya que Oracle no distribuye directamente el instalador en el repositorio de GitHub, lo cual requiere aceptar **licencias** adicionales.



Dockerizando Oracle

3. Creación manual de una imagen Docker desde cero.

- Para aquellos que necesitan un alto nivel de personalización y control, siempre existe la opción de crear una imagen Docker de Oracle Database desde cero, partiendo de una imagen base de Linux, por ejemplo, Oracle Linux o Ubuntu e instalando Oracle Database manualmente.
 - Es el método más complejo y requiere un conocimiento tanto de Docker como de la instalación de Oracle.



Dockerizando Oracle

4. Uso de imágenes de Oracle personalizadas en repositorios privados o comunitarios.

- En algunas ocasiones, otros usuarios o empresas han creado sus propias imágenes de Oracle para Docker y las han compartido en registros privados o comunitarios.
- Al ser imágenes no oficiales, deben ser revisadas y probadas antes de usar en entornos productivos.
- Pueden o cumplir con los estándares de **seguridad** y rendimiento de Oracle.



Dockerizando Oracle | Recomendaciones

- Para usuarios que buscan facilidad y compatibilidad oficial, las imágenes oficiales de **Oracle en Docker Hub** o **Oracle Container Registry** son la mejor opción.
- Para quienes necesitan personalización o versiones específicas, los **scripts de Oracle en GitHub** proporcionan una solución flexible y adaptada.
- Para entornos de prueba o aprendizaje, utilizar **imágenes de terceros** puede ser una opción rápida, aunque con precaución en cuanto a seguridad.
- Para máxima personalización y control, crear una **imagen manualmente** es una opción viable, aunque compleja y demandante en tiempo.



Dockerizando Oracle | Guía

Paso 1. Crear una cuenta y aceptar los términos de Licencia

a. [Oracle Container Registry](#)

- Busca "Oracle Database" y selecciona la versión que necesitas, p. ej, Oracle Database 21c.
- Acepta los términos de licencia necesarios.
 - Es obligatorio para que puedas descargar y ejecutar imágenes.

b. Docker Hub

- Busca "Oracle Database XE" y selecciona la versión que necesitas.
- Algunas versiones están disponibles de forma gratuita y no requieren registro adicional.

Dockerizando Oracle | Guía

Paso 2. Descargar la imagen de Oracle Database

```
# Oracle Container Registry
docker login container-registry.oracle.com
docker pull container-registry.oracle.com/database/enterprise:19.3.0
# Docker Hub
docker pull oracle/database:18.4.0-xe
```

Paso 3. Configurar y ejecutar el contenedor de Oracle Database

```
docker run --name oracle-db -p 1521:1521 -p 5500:5500 -e ORACLE_PWD=Netec_123
container-registry.oracle.com/database/express:21.3.0-xe
```

Dockerizando Oracle | Guía

Paso 4. Verificar que el contenedor de Oracle Database esté en ejecución.

```
docker ps
```

Paso 5: Acceder a Oracle Database desde una Aplicación Cliente.

- **Host:** localhost
- **Puerto:** 1521
- **SID/Service Name:** Dependerá de la versión
 - Puede ser ORCL o XE
 - Consulta la documentación específica de la versión.
- **Usuario:** SYS
- **Contraseña:** La contraseña que configuraste con la variable **ORACLE_PWD**.
 - Netec_123

Dockerizando Oracle | Verificación

```
C:\Símbolo del sistema

C:\00_Material_Docker_Kubernetes\Intermedio>
C:\00_Material_Docker_Kubernetes\Intermedio>docker images
REPOSITORY           TAG      IMAGE ID      CREATED     SIZE
container-registry.oracle.com/database/express  21.3.0-xe  dcf137aab02d  15 months ago  15.2GB

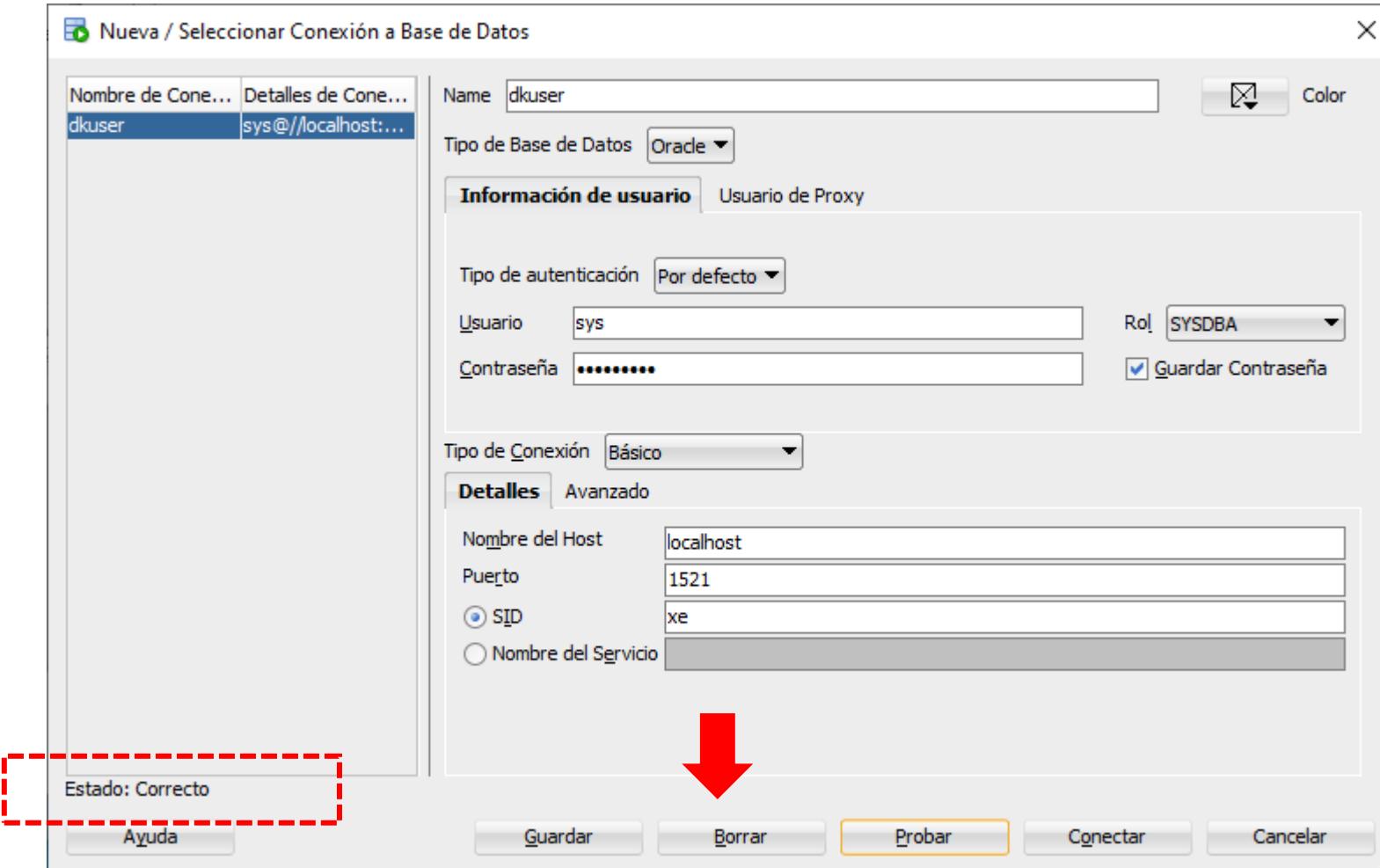
C:\00_Material_Docker_Kubernetes\Intermedio>docker ps
CONTAINER ID   IMAGE               COMMAND                  CREATED        STATUS
              NAMES
8fb8e2ac8fa7   container-registry.oracle.com/database/express:21.3.0-xe   "/bin/bash -c $ORACL..."   2 days ago   Up 2 days (healthy)
)   0.0.0.0:1521->1521/tcp, 0.0.0.0:5500->5500/tcp   oracle-db

C:\00_Material_Docker_Kubernetes\Intermedio>docker inspect oracle-db | grep -i ipaddress
    "SecondaryIPAddresses": null,
    "IPAddress": "172.17.0.2",
    "IPAddress": "172.17.0.2",

C:\00_Material_Docker_Kubernetes\Intermedio>netstat -ano | grep 1521
  TCP    0.0.0.0:1521          0.0.0.0:0                LISTENING      12136
  TCP    [::]:1521             [::]:0                 LISTENING      12136
  TCP    [::1]:1521            [::]:0                 LISTENING      15156

C:\00_Material_Docker_Kubernetes\Intermedio>
```

Dockerizando Oracle | Verificación



Dockerizando Oracle | MS Network

```
C:\Símbolo del sistema - .\mvnw clean install - .\mvnw clean install -Dmaven.test.skip=true - .\mvnw clean install -Dmaven.test.skip=true - .\mvnw clean install - .\mvnw clean install -Dmaven.test.skip=t...
C:\00_Material_Docker_Kubernetes\ws2\ms-favoritos>docker network inspect ms-network
[
  {
    "Name": "ms-network",
    "Id": "4ebba80043273fdb45618d0585a2deb0d8a2101c259ae1af549f70c346da5747",
    "Created": "2024-11-16T21:52:15.729853312Z",
    "Scope": "local",
    "Driver": "bridge",
    "EnableIPv6": false,
    "IPAM": {
      "Driver": "default",
      "Options": {},
      "Config": [
        {
          "Subnet": "172.18.0.0/16",
          "Gateway": "172.18.0.1"
        }
      ]
    },
    "Internal": false,
    "Attachable": false,
    "Ingress": false,
    "ConfigFrom": {
      "Network": ""
    },
    "ConfigOnly": false,
    "Containers": {
      "8fb8e2ac8fa71b46bcfad39b184fa14970133e872f4a7e447aa0ce05ff5c996a": {
        "Name": "oracle-db",
        "EndpointID": "772dd2c54c09b7fcfd1a0b89b16a17b0d431366addaa91192540c5817f3bbcb89",
        "MacAddress": "02:42:ac:12:00:02",
        "IPv4Address": "172.18.0.2/16",
        "IPv6Address": ""
      }
    },
    "Options": {},
    "Labels": {}
  }
]
C:\00_Material_Docker_Kubernetes\ws2\ms-favoritos>
```



1.4. Comunicación contenedores con BBDD Dockerizadas (Oracle)

- Cuando tu aplicación Spring Boot necesita conectarse a una base de datos Oracle Dockerizada, debes configurar las propiedades de conexión en el archivo: [application.properties](#) o [application.yml](#).
- Esto incluye la URL de conexión, las credenciales de la base de datos, el controlador JDBC, entre otros parámetros importantes.



Comunicación contenedores con BBDD

- Estos parámetros trabajan en conjunto para que Spring Boot pueda comunicarse correctamente con una base de datos Oracle.
 - `spring.datasource.url`
 - Define dónde se encuentra la base de datos y cómo acceder a ella.
 - `oracle-db`
 - Nombre del contenedor que hospeda Oracle Database Server.
 - `1521`
 - Puerto donde Oracle escucha conexiones.
 - `XEPDB1/XEPDB2`
 - Base de datos específica a la que conectarse.
 - `spring.datasource.username` & `spring.datasource.password`
 - Credenciales de autenticación.
 - `spring.datasource.driver-class-name`
 - Indica el driver JDBC necesario para establecer la conexión.

Comunicación contenedores con BBDD

```
spring.application.name=ms-favoritos
server.port=9094

# Configuracion de la base de datos
spring.datasource.url=jdbc:oracle:thin:@localhost:1521/XEPDB2
spring.datasource.username=dkuser
spring.datasource.password=dkpassword
spring.datasource.driver-class-name=oracle.jdbc.OracleDriver

# Configuracion de JPA e Hibernate
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.OracleDialect

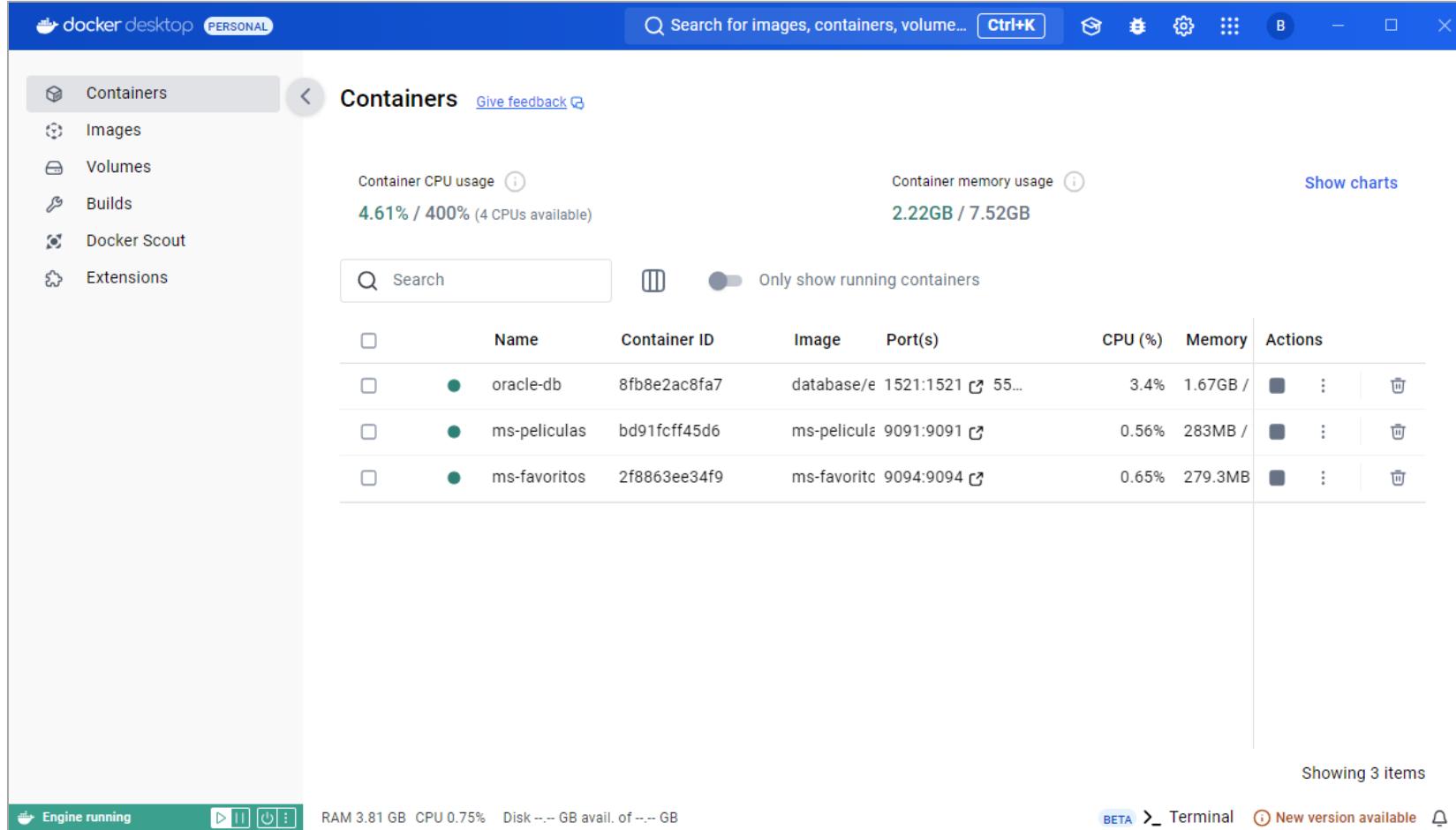
# Perfil Docker
mspeliculas.url=http://ms-peliculas:9091
```

localhost --> oracle-db

Comunicación contenedores con BBDD

1. .\mvnw clean install
2. docker rm <nombre_contenedor>
3. docker rmi <nombre_imagen> # Al menos en curso para ahorrar espacio
4. docker build -t <nombre_imagen>:<tag> .
5. docker run --name <nombre_contenedor> -p <puerto_host>:<puerto_container>
--network <nombre_red> -d <nombre_image>:<tag>

Comunicación contenedores con BBDD



The screenshot shows the Docker Desktop interface with the following details:

- Header:** docker desktop PERSONAL
- Search Bar:** Search for images, containers, volume... Ctrl+K
- Left Sidebar:** Containers (selected), Images, Volumes, Builds, Docker Scout, Extensions.
- Container CPU usage:** 4.61% / 400% (4 CPUs available)
- Container memory usage:** 2.22GB / 7.52GB
- Show charts:** Link
- Table:** Shows 3 running containers.

	Name	Container ID	Image	Port(s)	CPU (%)	Memory	Actions
<input type="checkbox"/>	oracle-db	8fb8e2ac8fa7	database/e	1521:1521	3.4%	1.67GB /	[Stop] [More] [Delete]
<input type="checkbox"/>	ms-peliculas	bd91fcff45d6	ms-pelicula	9091:9091	0.56%	283MB /	[Stop] [More] [Delete]
<input type="checkbox"/>	ms-favoritos	2f8863ee34f9	ms-favorito	9094:9094	0.65%	279.3MB	[Stop] [More] [Delete]

- Bottom Status Bar:** Engine running, RAM 3.81 GB, CPU 0.75%, Disk ... GB avail. of ... GB, Terminal, New version available



1.5. Revisando microservicios Dockerizados en Postman

- Un **endpoint** es un punto de acceso o comunicación a un sistema o servicio, que permite la interacción entre diferentes aplicaciones, servicios o clientes mediante una red.
- En el contexto de las APIs, un **endpoint** representa una URL específica que expone una funcionalidad de un servicio para que pueda ser consumida por clientes externos.

Revisando | Endpoints de caso de estudio

Método	Ruta	Descripción
GET	/api/peliculas	Listar todas las películas
GET	/api/peliculas/{id}	Obtener una película por su ID
POST	/api/peliculas	Guardar una nueva película
PUT	/api/peliculas/{id}	Actualizar una película existente
DELETE	/api/peliculas/{id}	Eliminar una película por su ID
GET	/api/peliculas/buscar	Buscar películas por título y género

Método	Ruta	Descripción
POST	/favoritos/{usuarioid}/{peliculaId}	Agregar una película a la lista de favoritos.
GET	/favoritos/{usuarioid}	Obtener la lista de favoritos de un usuario.

Revisando | Consumos caso de estudio

The screenshot shows the Postman application interface. In the top navigation bar, there are links for Home, Workspaces, API Network, and a search bar labeled "Search Postman". On the right side of the header are buttons for Invite, Upgrade, and various settings.

The main workspace displays a collection named "Collections" containing a single item: "GET http://localhost:9091/api/peliculas". The request details are shown in a red-bordered box:

- Method: GET
- URL: http://localhost:9091/api/peliculas

The response status is "200 OK" and it includes performance metrics: 17 ms, 647 B, and a link to "Save Response".

The response body is displayed in a large red-bordered box. It is a JSON array containing three objects, each representing a movie:

```
1 [  
2   {  
3     "id": 1,  
4     "titulo": "Star Wars: Episode IV - A New Hope",  
5     "director": "George Lucas",  
6     "genero": "Ciencia Ficción",  
7     "duracion": 121,  
8     "fechaEstreno": "1977-05-25"  
9   },  
10  {  
11    "id": 2,  
12    "titulo": "Star Wars: Episode V - The Empire Strikes Back",  
13    "director": "Irvin Kershner",  
14    "genero": "Ciencia Ficción",  
15    "duracion": 124,  
16    "fechaEstreno": "1980-05-21"  
17  },  
18  {  
19    "id": 3,  
20    "titulo": "Star Wars: Episode VI - Return of the Jedi",  
21    "director": "Richard Marquand",  
22    "genero": "Ciencia Ficción",  
23    "duracion": 131,  
24    "fechaEstreno": "1983-05-25"  
25  }]  
26 ]
```

At the bottom of the interface, there are buttons for Online, Find and replace, Console, Postbot, Runner, Start Proxy, Cookies, Vault, Trash, and Help.

Revisando | Consumos caso de estudio

The screenshot shows the Postman application interface. On the left, the sidebar includes 'Collections', 'Environments', 'History', and a 'Params' section. The main area displays a POST request to 'http://localhost:9091/api/peliculas'. The 'Body' tab is selected, showing a JSON payload:

```
1 {  
2   "id": 2,  
3   "titulo": "El Padrino",  
4   "director": "Francis Ford Coppola",  
5   "genero": "Drama",  
6   "duracion": 175,  
7   "fechaEstreno": "1972-03-24"  
8 }  
9
```

The response tab shows the result of the POST request:

200 OK 427 ms 288 B [Raw](#) [Pretty](#) [Preview](#) [Visualize](#) [JSON](#)

```
1 {  
2   "id": 2,  
3   "titulo": "El Padrino",  
4   "director": "Francis Ford Coppola",  
5   "genero": "Drama",  
6   "duracion": 175,  
7   "fechaEstreno": "1972-03-24"  
8 }
```

At the bottom, there are navigation links: Online, Find and replace, Console, Postbot, Runner, Start Proxy, Cookies, Vault, Trash, and a help icon.

Revisando | Consumos caso de estudio

The screenshot shows the Postman application interface. On the left, there's a sidebar with 'Collections' (highlighted with a green dashed box), 'Environments', and 'History'. The main area displays a request card for a GET request to `http://localhost:9094/favoritos/user1`. The 'Body' tab is selected, showing a JSON response with three items:

```
[{"id": 1, "usuarioId": "user1", "peliculaId": 1}, {"id": 2, "usuarioId": "user1", "peliculaId": 2}, {"id": 3, "usuarioId": "user1", "peliculaId": 3}]
```

The status bar at the bottom indicates a 200 OK response with 248 ms duration and 297 B size. The bottom navigation bar includes links for 'Postbot', 'Runner', 'Start Proxy', 'Cookies', 'Vault', 'Trash', and a help icon.

Revisando | Consumos caso de estudio

The screenshot shows the Postman interface with a successful API call. The URL is `http://localhost:9094/favoritos/user3/2`. The response status is `200 OK` with a response time of 170 ms and a size of 208 B. The response body is a JSON object:

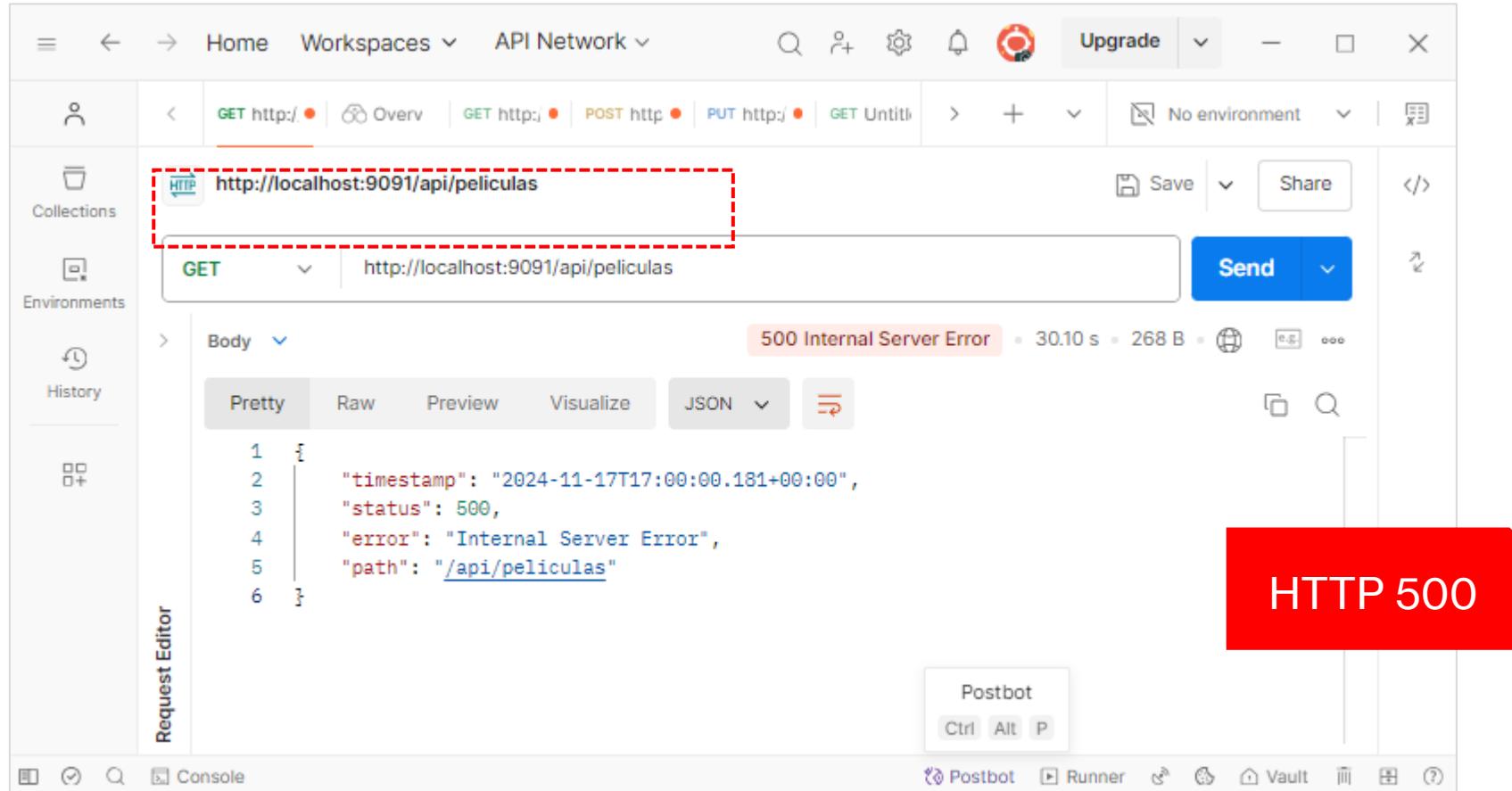
```
1 {  
2   "id": 21,  
3   "usuarioId": "user3",  
4   "peliculaId": 2  
5 }
```

1.6. Problema con persistencia de datos en Oracle al eliminar el contenedor

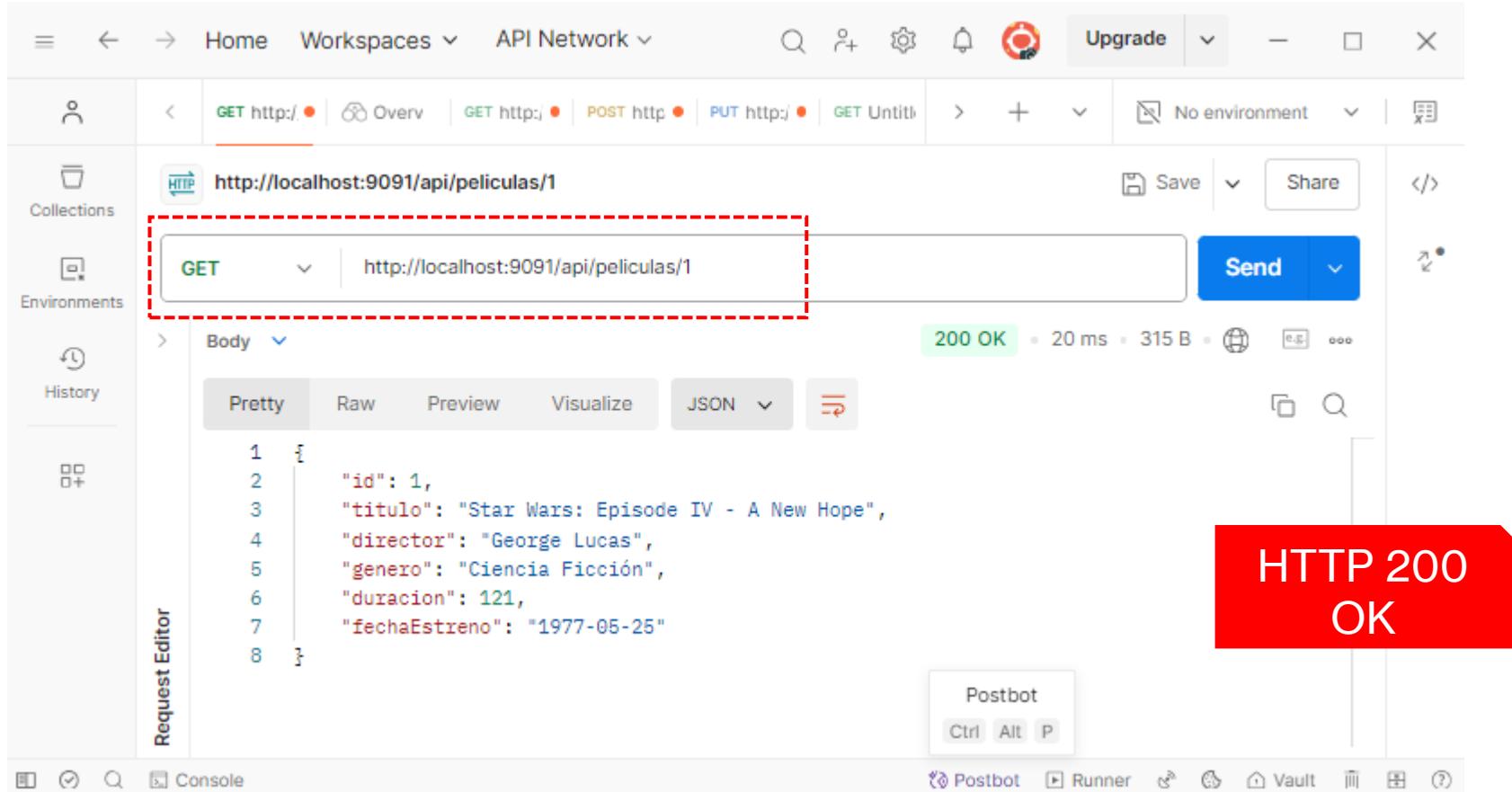
¿Por qué ocurre la pérdida de datos?

- En Docker, cada contenedor se ejecuta en un entorno aislado, que incluye su propio sistema de archivos. Este sistema de archivos es efímero, lo que significa que, por defecto, todo lo que almacena dentro del contenedor **se elimina permanentemente** cuando el contenedor se **elimina**.

Problemas con persistencia | docker stop



Problemas con persistencia | docker start



Problemas con persistencia | docker rm

- Al eliminar el contenedor con:
 - `docker rm <nombre_del_contenedor>`
 - `docker rm -f <nombre_del_contenedor>`
- Docker elimina completamente:
 - El contenedor.
 - El sistema de archivos asociados al contenedor.
- Esto significa que cualquier dato almacenado dentro del contenedor, como los datos en la base de datos Oracle, se pierde de forma definitiva. Incluso si se crea un nuevo contenedor con la misma configuración, esto no tendrá acceso a los datos del contenedor eliminado.

Problemas con persistencia | docker run

```
docker run --name oracle-db --network ms-network -p 1521:1521 -p 5500:5500 -e ORACLE_PWD=Netec_123 container-registry.oracle.com/database/express:21.3.0-xe
```

```
SQL>
Session altered.

SQL>
User altered.

SQL> Disconnected from Oracle Database 21c Express Edition Release 21.0.0.0 - Production
Version 21.3.0.0
The Oracle base remains unchanged with value /opt/oracle
#####
DATABASE IS READY TO USE!
#####
The following output is now a tail of the alert.log:
XEPDB1(3):Opening pdb with Resource Manager plan: DEFAULT_PLAN
2024-11-17T17:34:01.444247+00:00
Pluggable database XEPDB1 opened read write
Starting background process CJQ0
Completed: ALTER DATABASE OPEN
2024-11-17T17:34:01.614481+00:00
CJQ0 started with pid=63, OS id=699
2024-11-17T17:34:02.278523+00:00
TABLE AUDSYS.AUD$UNIFIED: ADDED INTERVAL PARTITION SYS_P328 (3793) VALUES LESS THAN (TIMESTAMP' 2024-11-18 00:00:00')
XEPDB1(3):TABLE AUDSYS.AUD$UNIFIED: ADDED INTERVAL PARTITION SYS_P348 (3793) VALUES LESS THAN (TIMESTAMP' 2024-11-18 00:00:00')
```

```
C:\00_Material_Docker_Kubernetes\ws2>
C:\00_Material_Docker_Kubernetes\ws2>docker ps --format "table {{.ID}}\t{{.Names}}\t{{.Status}}"
CONTAINER ID NAMES STATUS
d2295c546c1e oracle-db Up 2 minutes (healthy)
2f8863ee34f9 ms-favoritos Up 17 hours
bd91fcff45d6 ms-peliculas Up 17 hours

C:\00_Material_Docker_Kubernetes\ws2>
```

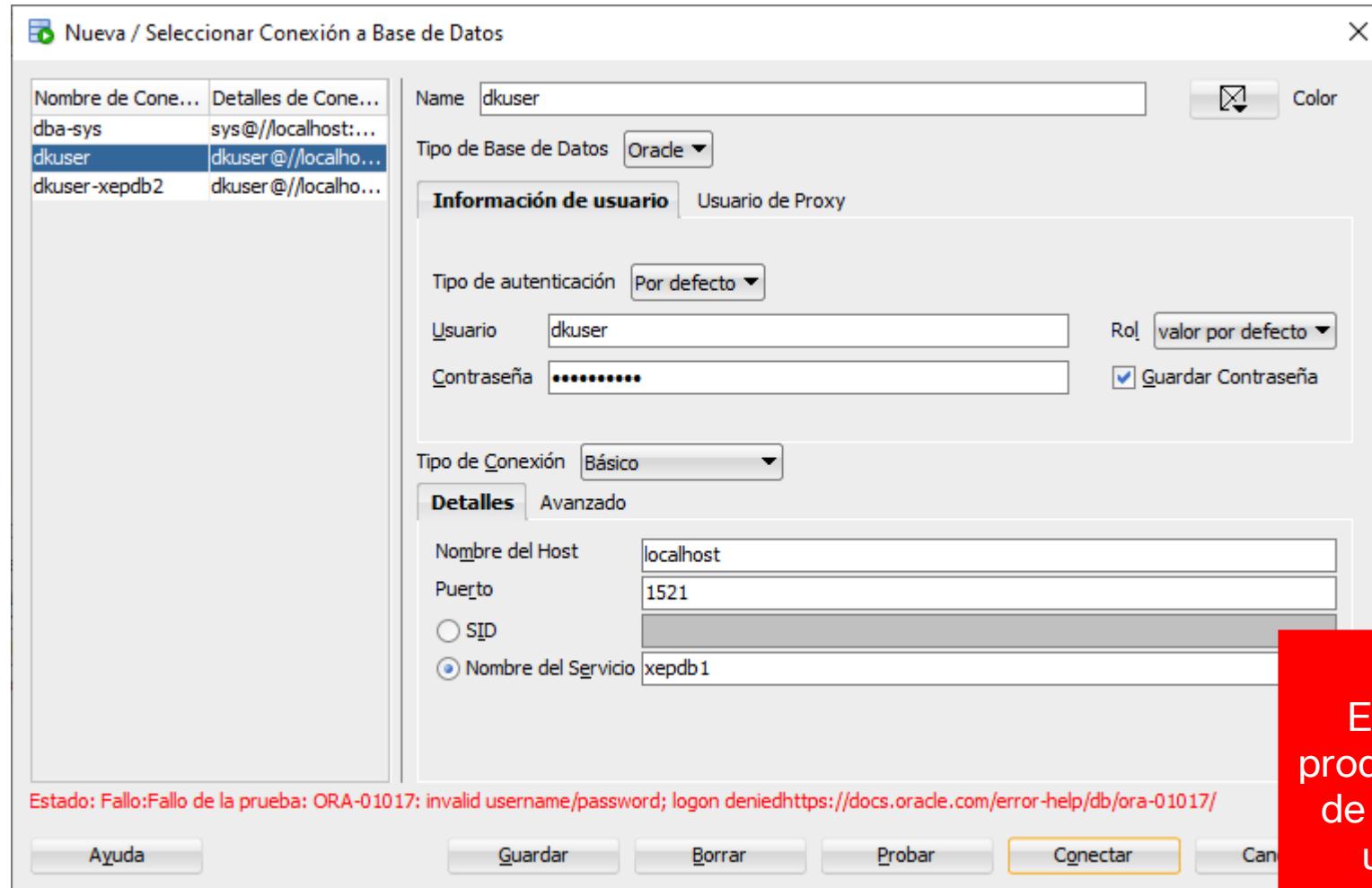
Problemas con persistencia | docker ps

```
C:\00_Material_Docker_Kubernetes\ws2>
C:\00_Material_Docker_Kubernetes\ws2>docker ps --format "table {{.ID}}\t{{.Names}}\t{{.Status}}"
CONTAINER ID NAMES STATUS
d2295c546c1e oracle-db Up 7 minutes (healthy)
2f8863ee34f9 ms-favoritos Up 17 hours
bd91fcff45d6 ms-peliculas Up 17 hours

C:\00_Material_Docker_Kubernetes\ws2>
C:\00_Material_Docker_Kubernetes\ws2>curl http://localhost:9091/api/peliculas | jq .
% Total    % Received % Xferd  Average Speed   Time     Time      Current
          Dload  Upload   Total Spent  Left Speed
100      114     0    114     0       0      3      0  --::--  0:00:30  --::--     30
{
  "timestamp": "2024-11-17T17:47:19.769+00:00",
  "status": 500,
  "error": "Internal Server Error",
  "path": "/api/peliculas"
}

C:\00_Material_Docker_Kubernetes\ws2>
```

Problemas con persistencia | Conexiones



ORA-01017

Este error generalmente se produce cuando las credenciales de inicio de sesión (nombre de usuario y contraseña) son incorrectas.

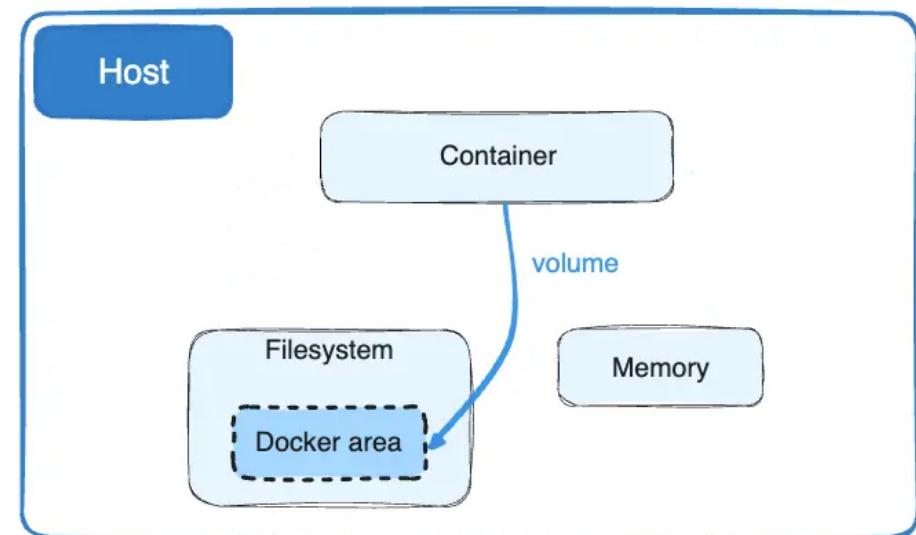
1.7. Docker Volumes: La solución al problema de persistencia de datos

¿Qué es un Volume en Docker?

- Un **volume** en Docker es un método de persistencia de datos diseñado específicamente para funcionar con contenedores.
- Es un espacio de almacenamiento que puede ser **montado** dentro de uno o varios contenedores, permitiendo que estos comparten datos o mantengan información persistente incluso si los contenedores se eliminan o reinician.

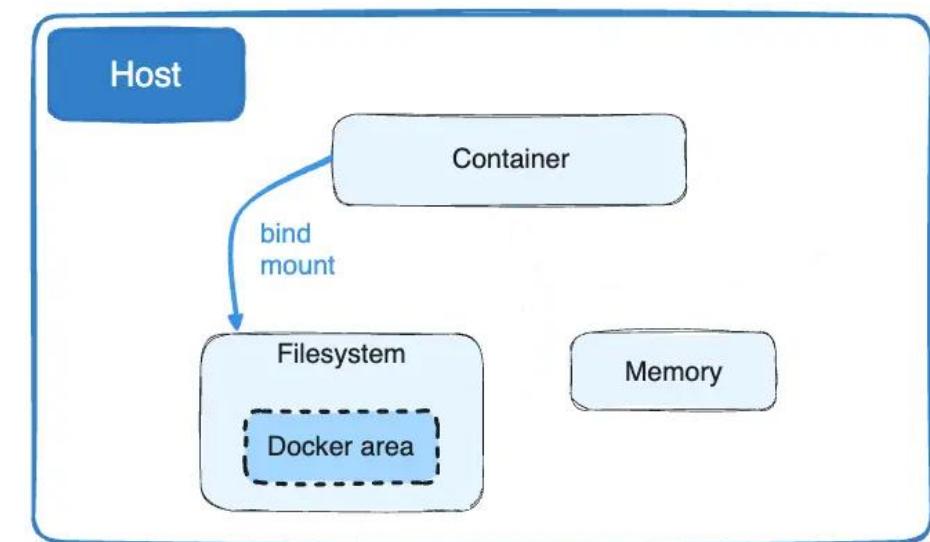
Docker volumes | Volúmenes

- Administrados por Docker.
- Almacenados fuera del sistema de archivos del contenedor.
 - En **/var/lib/docker/volumes** por defecto.
- Compatibles con sistemas operativos cruzados.
- Se usan comandos como **docker volume** para creación, inspección, eliminación, etc.
- Ventajas:
 - Completamente independientes del sistema anfitrión.
- Excelente para entornos de producción, donde se prioriza la consistencia y la seguridad.



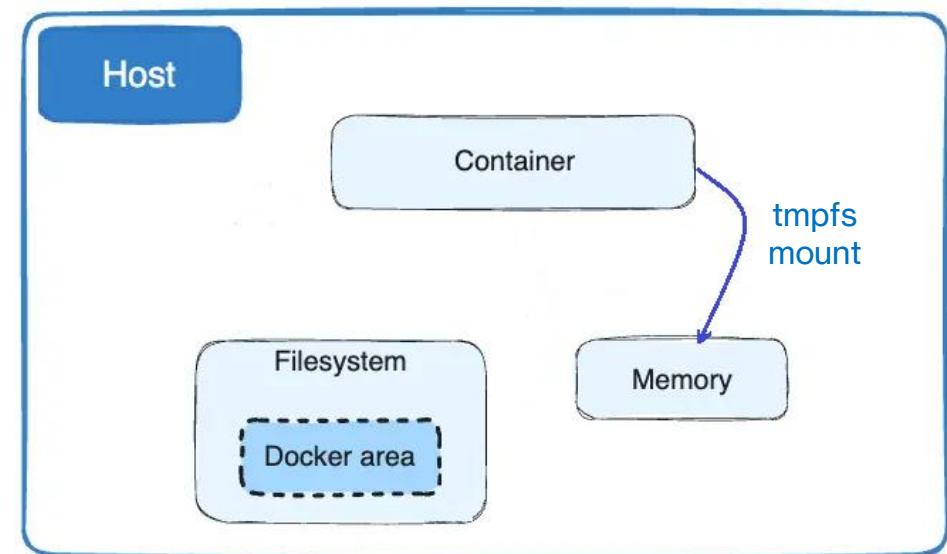
Docker volumes | Bind Mounts

- Mapean directamente un directorio o archivo del sistema de archivos anfitrión a un contenedor.
- `docker run -v /path/on/host:/path/in/container.`
- **No son administrados** por Docker; dependen de la estructura del sistema de archivos del anfitrión.
- Útiles para desarrollo, donde es necesario compartir código en tiempo real entre el host y el contenedor.
- Ventajas:
 - Ideal para desarrollo y depuración, ya que permite reflejar cambios en tiempo real.
 - Permite usar herramientas locales, como editores, IDEs, etc., directamente en el host para modificar datos.



Docker volumes | tmpfs

- Usa la **memoria volátil** del sistema anfitrión para almacenar datos.
- Los datos desaparecen cuando el contenedor se detiene o reinicia.
- `docker run --tmpfs /path/in/container`.
- Ideal para datos temporales que no necesitan persistir, como información sensible o caches.
- Ventajas:
 - Muy eficientes, ya que utiliza la memoria RAM.
 - Aumenta la seguridad al no persistir datos sensibles.



Docker volumes | Comparativa

Método	Ubicación de datos	Persistencia	Gestión	Casos de uso
Bind Mount	Sistema de archivos del host	Depende del host	No gestionado	Desarrollo o compartir archivos específicos.
Volume	Área específica de Docker	Persistente	Gestionado por Docker	Bases de datos, configuraciones críticas
tmpfs Mount	Memoria (RAM)	No persistente	Automático	Datos temporales, caches, datos sensibles

Docker volumes | Docker CLI

1 Crear un volume

```
docker volume create  
<nombre_volume>
```

2 Usar el volume en un contenedor

```
docker run -v <nombre_volume>:  
<ruta_contenedor>
```

3 Escribir Datos dentro del contenedor

```
docker exec -it <nombre_contenedor>  
bash
```

6 Volver a usar el volume

```
docker run -v <nombre_volume>:  
<ruta_contenedor>
```

5 Eliminar el Contenedor

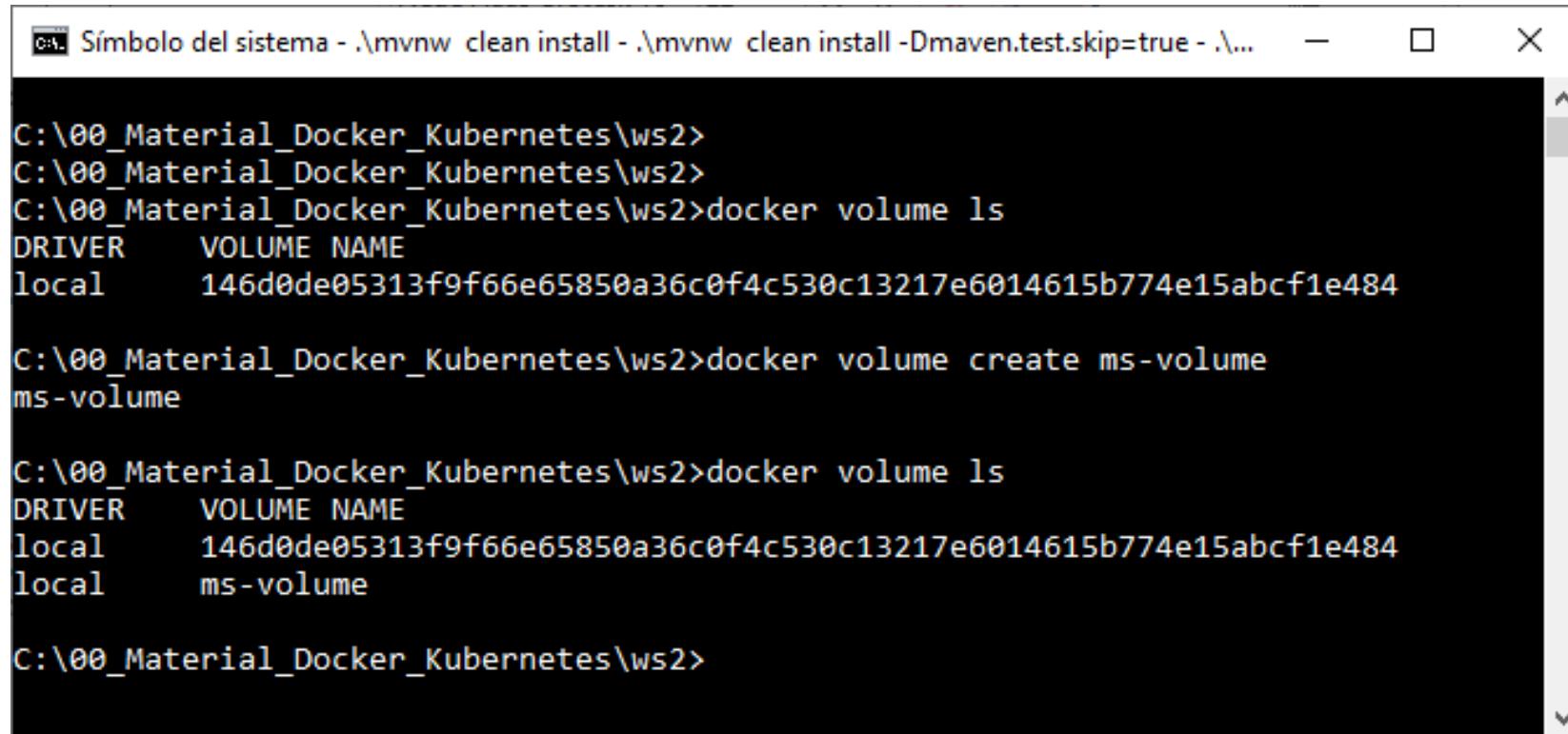
```
docker rm <nombre_contenedor>
```

4 Inspeccionar detalles del Volume

```
docker volume inspect  
<nombre_volume>
```

Docker volumes | docker volume create

docker volumen create <nombre_volumen>



```
C:\00_Material_Docker_Kubernetes\ws2>
C:\00_Material_Docker_Kubernetes\ws2>
C:\00_Material_Docker_Kubernetes\ws2>docker volume ls
DRIVER      VOLUME NAME
local      146d0de05313f9f66e65850a36c0f4c530c13217e6014615b774e15abcf1e484

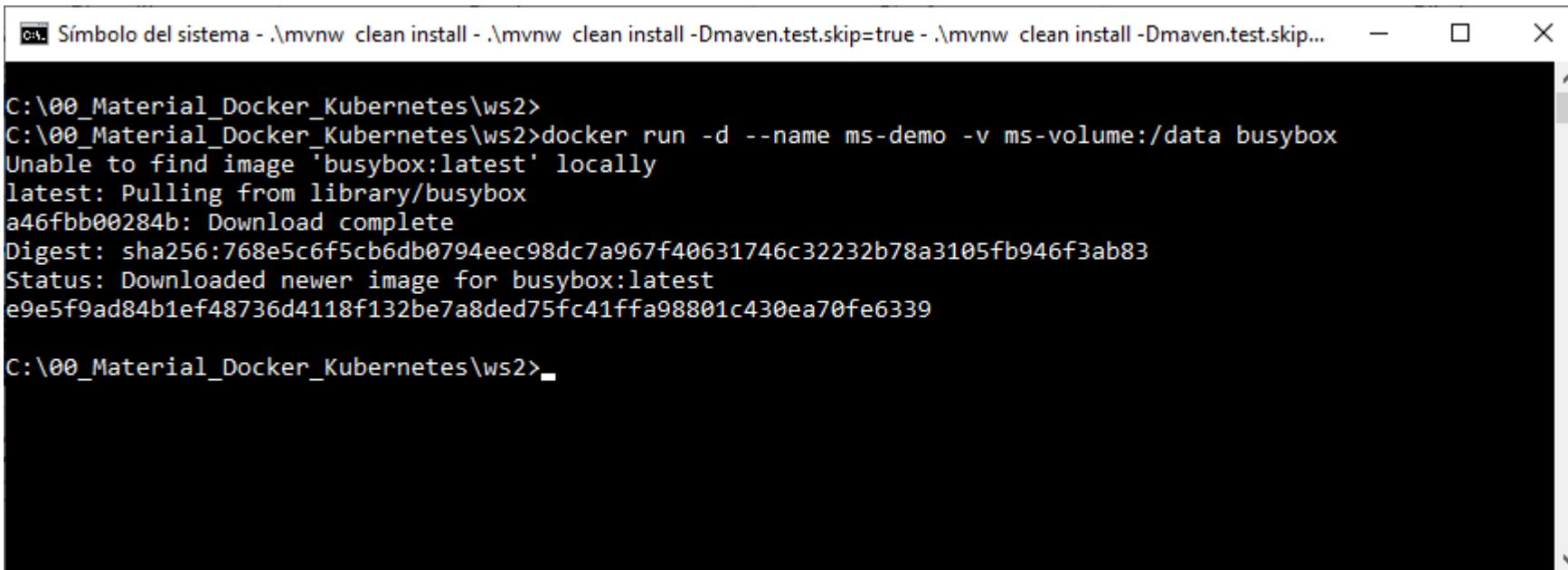
C:\00_Material_Docker_Kubernetes\ws2>docker volume create ms-volume
ms-volume

C:\00_Material_Docker_Kubernetes\ws2>docker volume ls
DRIVER      VOLUME NAME
local      146d0de05313f9f66e65850a36c0f4c530c13217e6014615b774e15abcf1e484
local      ms-volume

C:\00_Material_Docker_Kubernetes\ws2>
```

Docker volumes | docker run -v

- Podemos montar un volume en un contenedor al usar la opción `-v` o `--mount` al ejecutar docker run.
- `docker run -d --name <nombre_contenedor> -v <nombre_volume>:<ruta_contenedor> <nombre_imagen>:<tag>`



The screenshot shows a Windows Command Prompt window titled "Símbolo del sistema - .\mvnw clean install - .\mvnw clean install -Dmaven.test.skip=true - .\mvnw clean install -Dmaven.test.skip...". The command entered is:

```
C:\00_Material_Docker_Kubernetes\ws2>docker run -d --name ms-demo -v ms-volume:/data busybox
```

The output shows the Docker daemon attempting to pull the 'busybox:latest' image from a local library. It finds the image locally and downloads a newer version, resulting in a different digest and status message.

```
Unable to find image 'busybox:latest' locally
latest: Pulling from library/busybox
a46fbb00284b: Download complete
Digest: sha256:768e5c6f5cb6db0794eec98dc7a967f40631746c32232b78a3105fb946f3ab83
Status: Downloaded newer image for busybox:latest
e9e5f9ad84b1ef48736d4118f132be7a8ded75fc41ffa98801c430ea70fe6339
```

C:\00_Material_Docker_Kubernetes\ws2>

Docker volumes | docker run -v

```
C:\00_Material_Docker_Kubernetes\ws2>docker volume ls
DRIVER      VOLUME NAME
local      146d0de05313f9f66e65850a36c0f4c530c13217e6014615b774e15abcf1e484
local      ms-volume

C:\00_Material_Docker_Kubernetes\ws2>docker run -d --name nginx-container -v ms-volume:/usr/share/nginx/html -p 8090:80 nginx
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
171eebbdf235: Download complete
9b1039c85176: Download complete
9ad567d3b8a2: Download complete
773c63cd62e4: Download complete
1d2712910bdf: Download complete
2d429b9e73a6: Download complete
4b0adc47c460: Download complete
Digest: sha256:bc5eac5eafc581aeda3008b4b1f07ebba230de2f27d47767129a6a905c84f470
Status: Downloaded newer image for nginx:latest
9253ed4f2930ded9c08584779025c23cd26fc4b8fcbed4366b93dc32244603d2

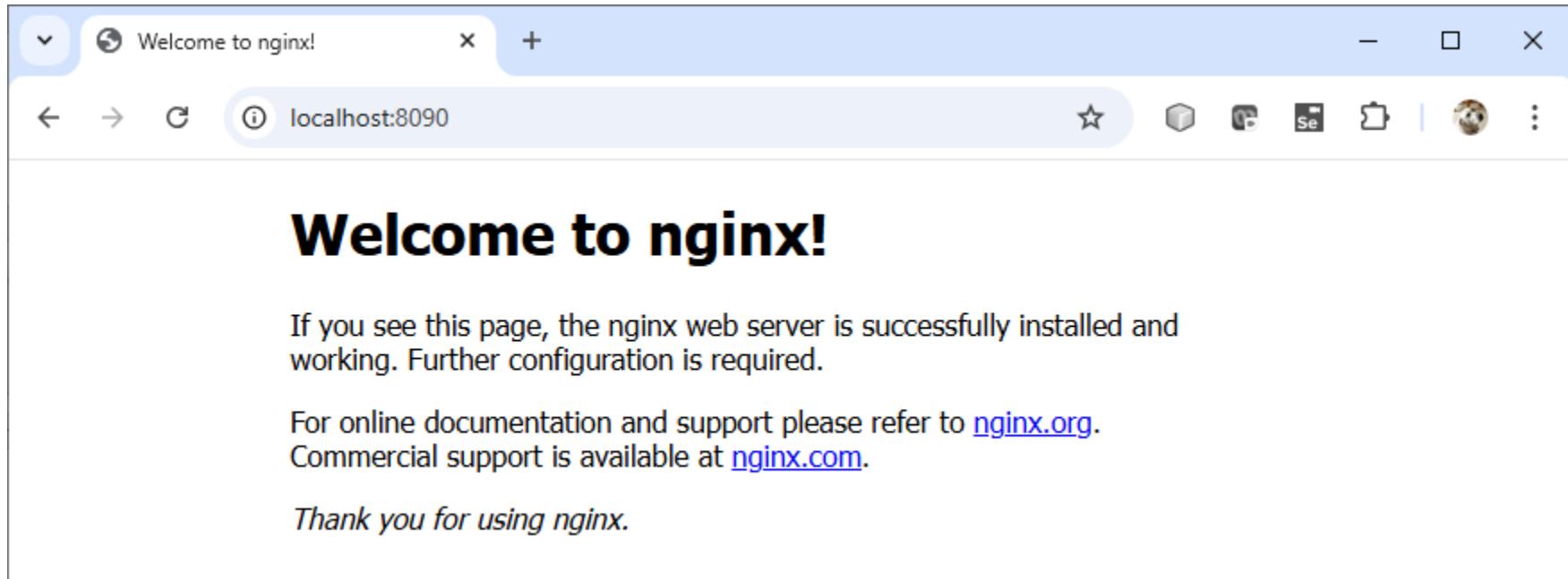
C:\00_Material_Docker_Kubernetes\ws2>docker ps -a --format "table {{.ID}}\t{{.Names}}\t{{.Status}}\t{{.Ports}}"
CONTAINER ID   NAMES          STATUS        PORTS
9253ed4f2930  nginx-container  Up 47 seconds  0.0.0.0:8090->80/tcp
d2295c546c1e  oracle-db     Up 2 hours (healthy)  0.0.0.0:1521->1521/tcp, 0.0.0.0:5500->5500/tcp
2f8863ee34f9  ms-favoritos   Up 18 hours    0.0.0.0:9094->9094/tcp
bd91fcff45d6  ms-peliculas   Up 18 hours    0.0.0.0:9091->9091/tcp

C:\00_Material_Docker_Kubernetes\ws2>docker ps -a --format "table {{.ID}}\t{{.Names}}\t{{.Status}}\t{{.Ports}}\t{{.Mounts}}"
CONTAINER ID   NAMES          STATUS        PORTS          MOUNTS
9253ed4f2930  nginx-container  Up About a minute  0.0.0.0:8090->80/tcp          ms-volume
d2295c546c1e  oracle-db     Up 2 hours (healthy)  0.0.0.0:1521->1521/tcp, 0.0.0.0:5500->5500/tcp
2f8863ee34f9  ms-favoritos   Up 18 hours    0.0.0.0:9094->9094/tcp
bd91fcff45d6  ms-peliculas   Up 19 hours    0.0.0.0:9091->9091/tcp

C:\00_Material_Docker_Kubernetes\ws2>
```



Docker volumes | nginx



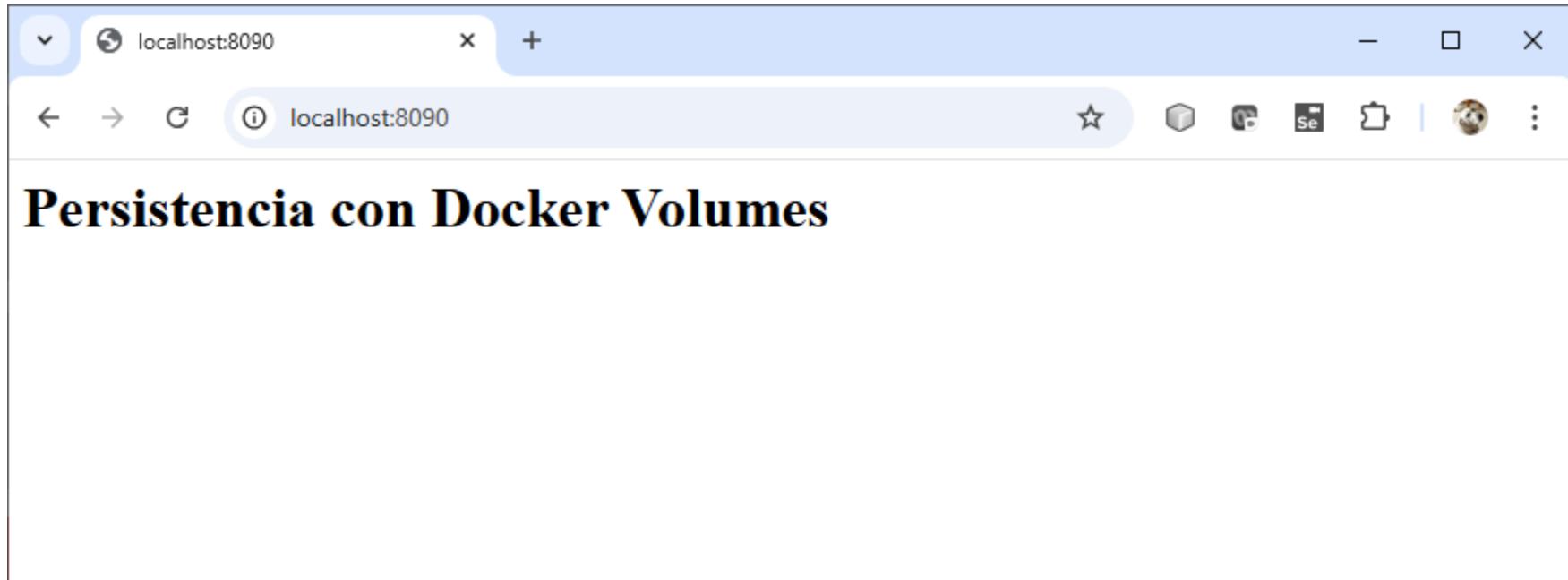
Docker volumes | docker exec

- Modifica los datos dentro del contenedor para verificar la persistencia.
- Accede al contenedor.
- `docker exec -it <nombre_contenedor> bash`
- Escribe por ejemplo un archivo HTML en la ruta donde está montado el Docker volume.
- Sal del contenedor y verifica en el navegador la salida recargando la página en el navegador.

```
C:\00_Material_Docker_Kubernetes\ws2>docker ps --format "table {{.ID}}\t{{.Names}}\t{{.Status}}\t{{.Ports}}"
CONTAINER ID NAMES STATUS PORTS
9253ed4f2930 nginx-container Up 4 hours 0.0.0.0:8090->80/tcp
d2295c546c1e oracle-db Up 5 hours (healthy) 0.0.0.0:1521->1521/tcp, 0.0.0.0:5500->5500/tcp
2f8863ee34f9 ms-favoritos Up 22 hours 0.0.0.0:9094->9094/tcp
bd91fcff45d6 ms-peliculas Up 22 hours 0.0.0.0:9091->9091/tcp

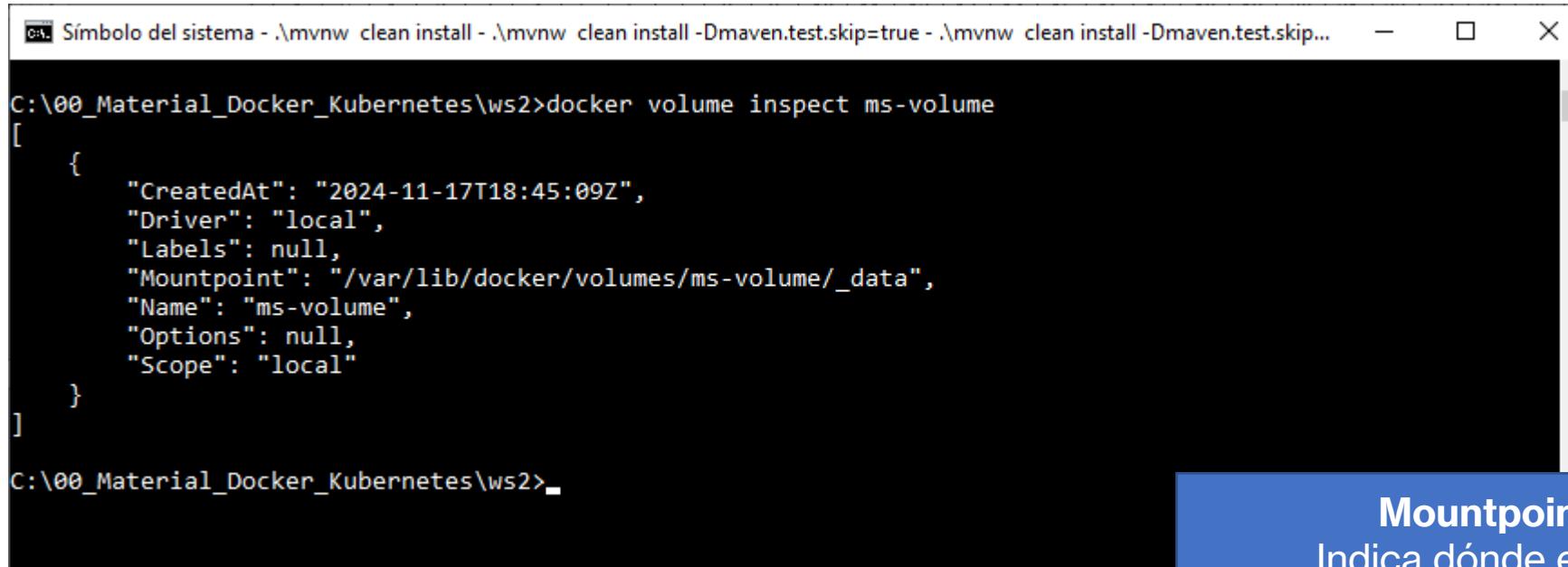
C:\00_Material_Docker_Kubernetes\ws2>docker exec -it nginx-container bash
root@9253ed4f2930:/#
root@9253ed4f2930:/# echo "<h1>Persistencia con Docker Volumes</h1>" > /usr/share/nginx/html/index.html
root@9253ed4f2930:/#
```

Docker volumes | nginx



Docker volumes | docker volume inspect

- Para verificar información sobre un volume, como se ubicación en el host, usamos el comando:
- `docker volume inspect <nombre_volume>`



```
C:\00_Material_Docker_Kubernetes\ws2>docker volume inspect ms-volume
[{"CreatedAt": "2024-11-17T18:45:09Z", "Driver": "local", "Labels": null, "Mountpoint": "/var/lib/docker/volumes/ms-volume/_data", "Name": "ms-volume", "Options": null, "Scope": "local"}]
```

Mountpoint

Indica dónde están almacenados los datos en el host.

Docker volumes | Reuso

- Elimina el contenedor.
 - `docker rm <nombre_contenedor>`
- Crea un nuevo contenedor puedes usar el mismo nombre del contenedor.
 - `docker run -d --name nginx-container-2 -v ms-volume:/usr/share/nginx/html -p 8090:80 nginx`
- Al cargar <http://localhost:8090> en el navegador. Deberíamos ver el contenido persistido.
 - Esto confirma que los datos están almacenado en el volume.

Docker volumes | Reuso

```
C:\ Símbolo del sistema - mvnw clean install - mvnw clean install -Dmaven.test.skip=true - mvnw clean install -Dmaven.test.skip=true - mvnw clean install - mvnw clea... — □ X

C:\00_Material_Docker_Kubernetes\ws2>docker ps --format "table {{.ID}}\t{{.Names}}\t{{.Status}}\t{{.Ports}}"
CONTAINER ID NAMES STATUS PORTS
9253ed4f2930 nginx-container Up 4 hours 0.0.0.0:8090->80/tcp
d2295c546c1e oracle-db Up 5 hours (healthy) 0.0.0.0:1521->1521/tcp, 0.0.0.0:5500->5500/tcp
2f8863ee34f9 ms-favoritos Up 22 hours 0.0.0.0:9094->9094/tcp
bd91fcff45d6 ms-peliculas Up 22 hours 0.0.0.0:9091->9091/tcp

C:\00_Material_Docker_Kubernetes\ws2>docker stop nginx-container
nginx-container

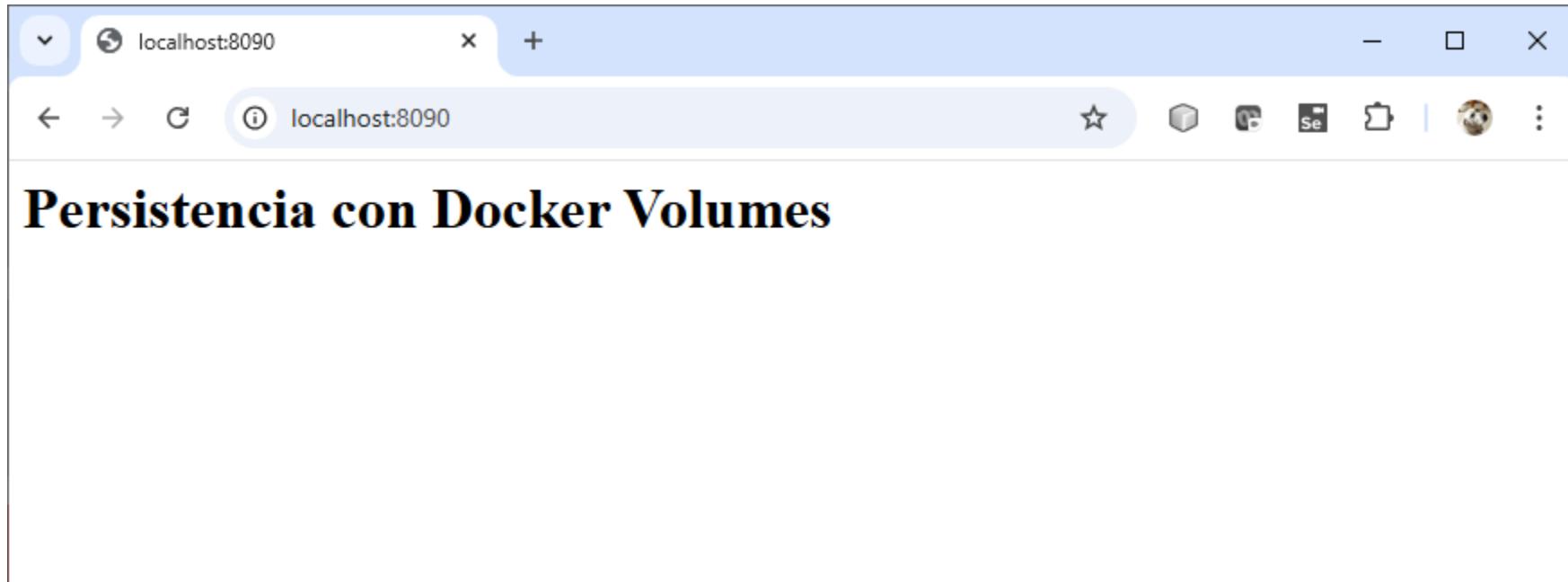
C:\00_Material_Docker_Kubernetes\ws2>docker rm nginx-container
nginx-container

C:\00_Material_Docker_Kubernetes\ws2>docker run -d --name nginx-container2 -v ms-volume:/usr/share/nginx/html -p 8090:80 nginx
67ed2e1491cc98001baadea2ecbe7e23ba3d7fd7f6c4205fac850ff41d66c7a9

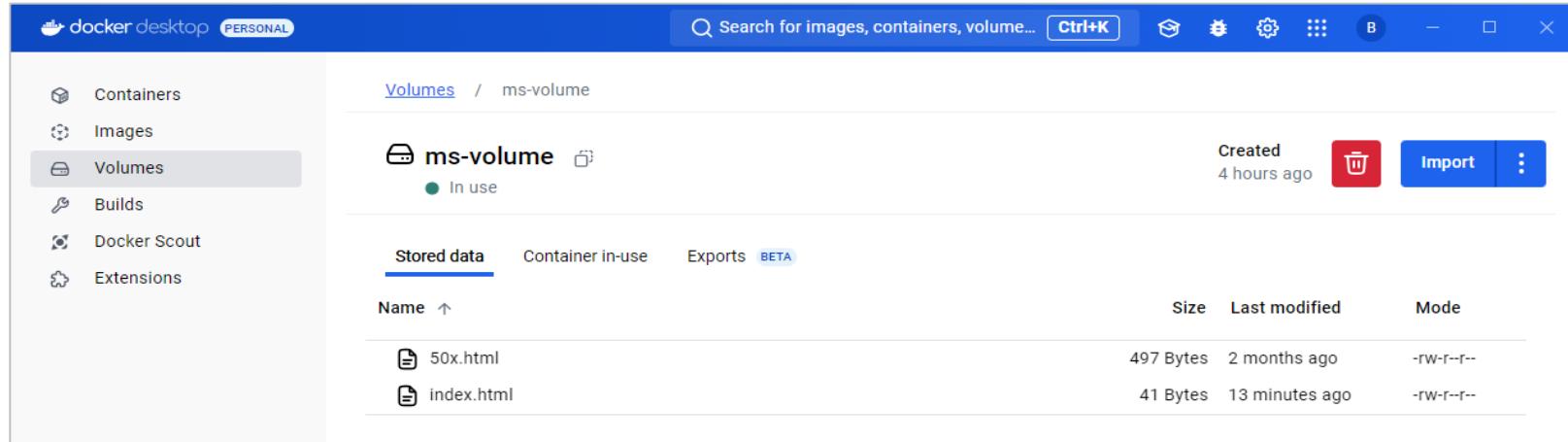
C:\00_Material_Docker_Kubernetes\ws2>docker ps --format "table {{.ID}}\t{{.Names}}\t{{.Status}}\t{{.Ports}}"
CONTAINER ID NAMES STATUS PORTS
67ed2e1491cc nginx-container2 Up 11 seconds 0.0.0.0:8090->80/tcp
d2295c546c1e oracle-db Up 5 hours (healthy) 0.0.0.0:1521->1521/tcp, 0.0.0.0:5500->5500/tcp
2f8863ee34f9 ms-favoritos Up 22 hours 0.0.0.0:9094->9094/tcp
bd91fcff45d6 ms-peliculas Up 22 hours 0.0.0.0:9091->9091/tcp

C:\00_Material_Docker_Kubernetes\ws2>
```

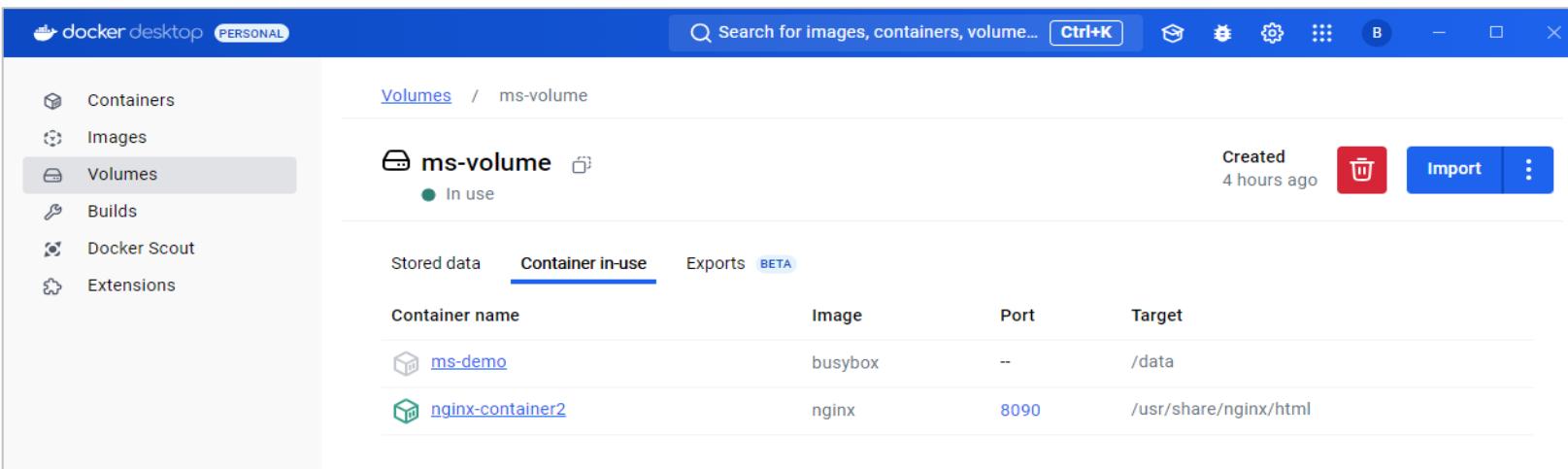
Docker volumes | Ejemplo



Docker volumes | Docker Desktop



The screenshot shows the Docker Desktop interface with the 'Volumes' tab selected in the sidebar. A volume named 'ms-volume' is listed, which was created 4 hours ago and is currently in use. The 'Stored data' tab is selected, showing two files: '50x.html' (497 Bytes, last modified 2 months ago) and 'index.html' (41 Bytes, last modified 13 minutes ago).



The screenshot shows the same Docker Desktop interface, but the 'Container in-use' tab is now selected for the 'ms-volume' volume. It lists two containers: 'ms-demo' (using the 'busybox' image) and 'nginx-container2' (using the 'nginx' image). The 'Container name' column lists the names of the containers, the 'Image' column lists the images they are using, and the 'Port' and 'Target' columns show the mapped ports and target paths.

Docker volumes | Caso estudio

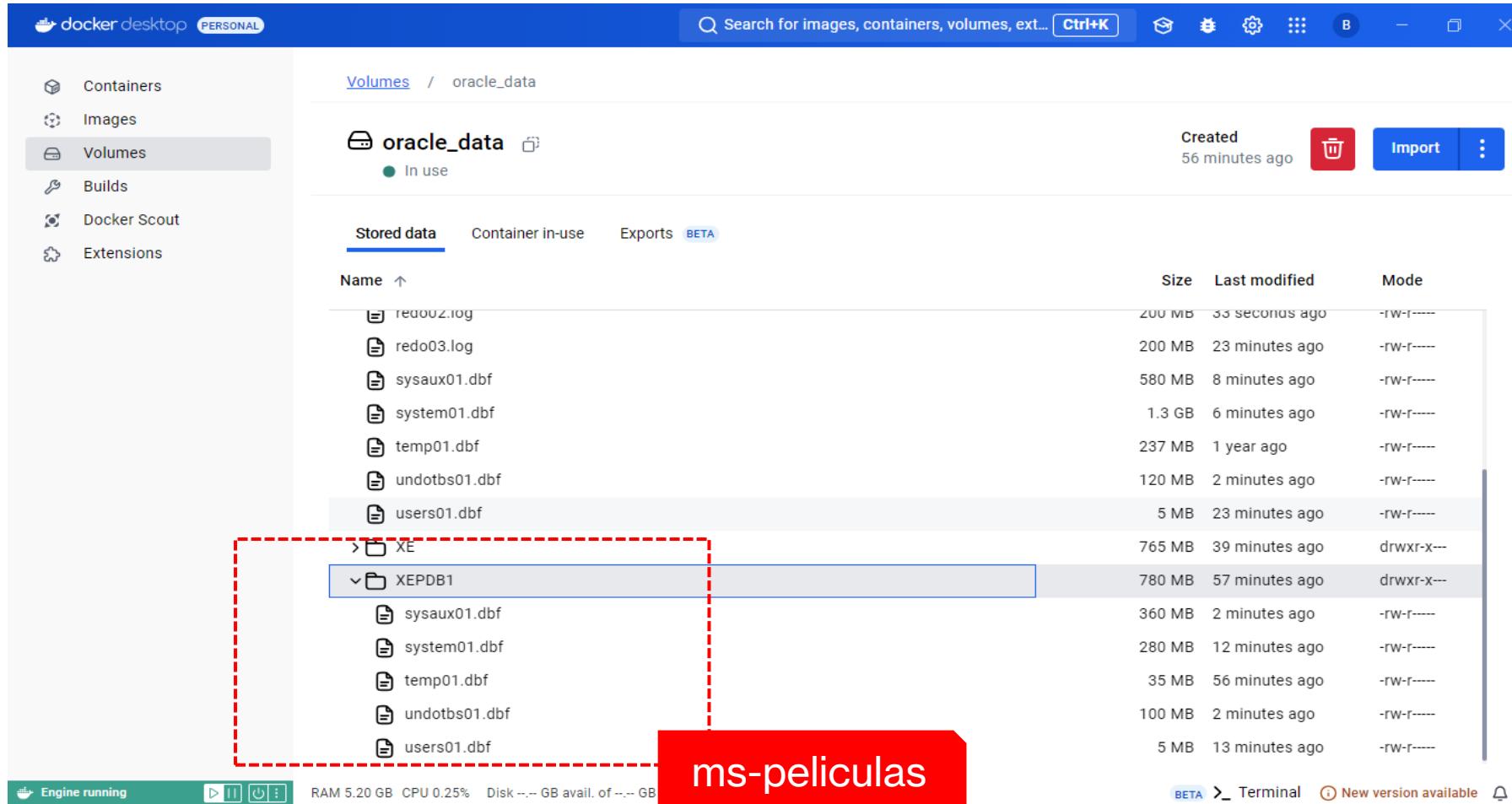
- Si deseas que los datos de Oracle Database se mantengan incluso usando el contenedor se elimine, usamos un **Docker Volume**.

```
docker run -d --name oracle-db -e ORACLE_PWD=Netec_123 -p 1521:1521  
-p 5500:5500 -v oracle_data:/opt/oracle/oradata --network ms-network  
container-registry.oracle.com/database/express:21.3.0-xe
```

- La opción **-v oracle_data:/opt/oracle/oradata**, usa un volumen llamado **oracle_data** que almacenará los datos en la carpeta donde Oracle Database guarda sus archivos de datos.

```
C:\00_Material_Docker_Kubernetes\ws2>  
C:\00_Material_Docker_Kubernetes\ws2>docker ps --format "{{.Names}}\t{{.Command}}\t{{.Ports}}\t{{.Mounts}}"  
oracle-db    "/bin/bash -c $ORACL..."  0.0.0.0:1521->1521/tcp, 0.0.0.0:5500->5500/tcp  oracle_data  
ms-favoritos "java -jar app.jar"      0.0.0.0:9094->9094/tcp  
ms-peliculas "java -jar app.jar"      0.0.0.0:9091->9091/tcp  
  
C:\00_Material_Docker_Kubernetes\ws2>
```

Docker volumes | Caso de estudio



Docker volumes | Caso de estudio

The image displays two side-by-side screenshots of the Postman application interface, illustrating a case study involving Docker volumes.

Left Screenshot: Shows a GET request to `http://localhost:9091/api/peliculas`. The response body is a JSON array containing three movie objects:

```
1 [  
2 {  
3   "id": 1,  
4   "titulo": "El Padrino",  
5   "director": "Francis Ford Coppola",  
6   "genero": "Drama",  
7   "duracion": 175,  
8   "fechaEstreno": "1972-03-24"  
9 },  
10 {  
11   "id": 2,  
12   "titulo": "El Padrino II",  
13   "director": "Francis Ford Coppola",  
14   "genero": "Drama",  
15   "duracion": 202,  
16   "fechaEstreno": "1974-12-20"  
17 },  
18 {  
19   "id": 3,  
20   "titulo": "El Padrino III",  
21   "director": "Francis Ford Coppola",  
22   "genero": "Drama",  
23   "duracion": 162,  
24   "fechaEstreno": "1990-12-25"  
25 }  
26 ]
```

Right Screenshot: Shows a GET request to `http://localhost:9094/favoritos/user2`. The response body is a JSON array containing four favorite movie objects:

```
1 [  
2 {  
3   "id": 2,  
4   "usuarioId": "user2",  
5   "peliculaId": 1  
6 },  
7 {  
8   "id": 3,  
9   "usuarioId": "user2",  
10  "peliculaId": 2  
11 },  
12 {  
13   "id": 4,  
14   "usuarioId": "user2",  
15   "peliculaId": 3  
16 }  
17 ]
```

Docker volumes | Caso de estudio

```
c:\00_Material_Docker_Kubernetes\ws2>docker ps -a --format "table {{.ID}}\t{{.Names}}\t{{.Mounts}}"
CONTAINER ID NAMES MOUNTS
2f8863ee34f9 ms-favoritos
bd91fcfff45d6 ms-peliculas

c:\00_Material_Docker_Kubernetes\ws2>docker volume ls
DRIVER VOLUME NAME
local 146d0de05313f9f66e65850a36c0f4c530c13217e6014615b774e15abcf1e484
local ms-volume
local oracle_data

c:\00_Material_Docker_Kubernetes\ws2>docker volume inspect oracle_data
[
  {
    "CreatedAt": "2024-11-18T00:28:46Z",
    "Driver": "local",
    "Labels": null,
    "Mountpoint": "/var/lib/docker/volumes/oracle_data/_data",
    "Name": "oracle_data",
    "Options": null,
    "Scope": "local"
  }
]

c:\00_Material_Docker_Kubernetes\ws2>
```

Docker volumes | Caso de estudio

```
c:\00_Material_Docker_Kubernetes\ws2>
c:\00_Material_Docker_Kubernetes\ws2>docker ps -a --format "table {{.ID}}\t{{.Names}}\t{{.Mounts}}"
CONTAINER ID NAMES MOUNTS
2f8863ee34f9 ms-favoritos
bd91fcff45d6 ms-peliculas

c:\00_Material_Docker_Kubernetes\ws2>docker run -d --name oracle-db -e ORACLE_PWD=Netec_123 -p 1521:1521 -p 5500:5500 -v oracle_data:/opt/oracle/oradata --network ms-netwokr container-registry.oracle.com/database/express:21.3.0-xe
1a8da8fd3befbc260538773d5ce30c20df5ba339f0a1c20a44a22eb698e3b4b8

c:\00_Material_Docker_Kubernetes\ws2>docker ps -a --format "table {{.ID}}\t{{.Names}}\t{{.Mounts}}"
CONTAINER ID NAMES MOUNTS
1a8da8fd3bef oracle-db oracle_data
2f8863ee34f9 ms-favoritos
bd91fcff45d6 ms-peliculas

c:\00_Material_Docker_Kubernetes\ws2>curl http://localhost:9091/api/peliculas | jq .
% Total    % Received % Xferd  Average Speed   Time   Time   Current
          Dload  Upload Total Spent   Left Speed
100  383     0  383     0      0  1817      0 --:--:-- --:--:-- 1823
[
  {
    "id": 1,
    "titulo": "El Padrino",
    "director": "Francis Ford Coppola",
    "genero": "Drama",
    "duracion": 175,
    "fechaEstreno": "1972-03-24"
  },
  {
    "id": 2,
    "titulo": "El Padrino II",
    "director": "Francis Ford Coppola",
    "genero": "Drama",
    "duracion": 202,
    "fechaEstreno": "1974-12-20"
  },
  {
    "id": 3,
    "titulo": "El Padrino III",
    "director": "Francis Ford Coppola",
    "genero": "Drama",
    "duracion": 162,
    "fechaEstreno": "1990-12-25"
  }
]
```

Docker volumes | Caso de estudio

```
"duracion": 175,  
"fechaEstreno": "1972-03-24"  
},  
{  
    "id": 2,  
    "titulo": "El Padrino II",  
    "director": "Francis Ford Coppola",  
    "genero": "Drama",  
    "duracion": 202,  
    "fechaEstreno": "1974-12-20"  
},  
{  
    "id": 3,  
    "titulo": "El Padrino III",  
    "director": "Francis Ford Coppola",  
    "genero": "Drama",  
    "duracion": 162,  
    "fechaEstreno": "1990-12-25"  
}  
]  
  
c:\00_Material_Docker_Kubernetes\ws2>curl http://localhost:9094/favoritos | jq .  
% Total    % Received % Xferd  Average Speed   Time   Time Current  
          Dload Upload   Total Spent   Left Speed  
100     98     0    98     0      0  666      0 --::-- --::-- --::--  671  
{  
    "timestamp": "2024-11-18T02:30:41.514+00:00",  
    "status": 404,  
    "error": "Not Found",  
    "path": "/favoritos"  
}  
  
c:\00_Material_Docker_Kubernetes\ws2>curl http://localhost:9094/favoritos/user1 | jq .  
% Total    % Received % Xferd  Average Speed   Time   Time Current  
          Dload Upload   Total Spent   Left Speed  
100     45     0    45     0      0  226      0 --::-- --::-- --::--  226  
[  
    {  
        "id": 1,  
        "usuarioId": "user1",  
        "peliculaId": 2  
    }  
]  
c:\00_Material_Docker_Kubernetes\ws2>
```

1.8. Conectarse desde contenedor cliente de línea de comandos a Oracle

- Un cliente de línea de comandos como **SQL*Plus** es una herramienta ligera y esencial para interactuar con bases de datos Oracle.
- Usar contenedores Docker para este propósito permite ejecutar el cliente **sin instalar** software adicional en el host, manteniendo el entorno limpio y gestionable.

Conectarse | Preparativos

1. Contenedor de Oracle se encuentre activo.
 - `docker ps`
2. Puerto expuesto
 - `netstat -ano | grep 1521 # Linux`
 - `netstat -ano | findstr 1521 # MS-DOS`
3. Credenciales y configuración, en nuestro caso de estudio:
 - Usuario: `sys/system/dkuser`
 - Contraseña: `Netec_123/dkpassword`
 - SID/Service: `xe/xepdb1/xppdb2`

Conectarse | Contenedor cliente

Oracle Instant Client

- Es una versión ligera de Oracle Database que permite a las aplicaciones conectarse a bases de datos Oracle **sin necesidad de una instalación completa** del software de base de datos.
- **Ideal para desarrolladores** que necesitan acceso rápido y simple a una base de datos Oracle desde sus aplicaciones o scripts.
- Soporta la mayoría de las funcionalidades del cliente completo, como conexión segura, autenticación y ejecución de **comandos PL/SQL**.
- Disponible para sistemas operativos como Windows, macOS, y Linux.
- Menor tamaño comparado con una instalación completa.
- Se encuentran **ímágenes Docker** en: Oracle Container Registry & Docker Hub.

Conectarse | Creación contenedor básico

```
c:\00_Material_Docker_Kubernetes\ws2\oracle_client>
c:\00_Material_Docker_Kubernetes\ws2\oracle_client>docker pull ghcr.io/oracle/oraclelinux8-instantclient:19
19: Pulling from oracle/oraclelinux8-instantclient
Digest: sha256:b4e4e6f22460c659ad7351f047736440b1a28c31e25daf221ad63560560fc3ad
Status: Image is up to date for ghcr.io/oracle/oraclelinux8-instantclient:19
ghcr.io/oracle/oraclelinux8-instantclient:19

What's next:
  View a summary of image vulnerabilities and recommendations → docker scout quickview ghcr.io/oracle/oraclelinux8-instantclient:19

c:\00_Material_Docker_Kubernetes\ws2\oracle_client>docker images
REPOSITORY                                TAG      IMAGE ID      CREATED       SIZE
ms-favoritos                             1.0.0   a2b7174d7546  41 hours ago  805MB
ms-peliculas                            1.0.0   b0aaa0b4e16f  41 hours ago  787MB
ghcr.io/oracle/oraclelinux8-instantclient    19      b4e4e6f22460  9 days ago   717MB
container-registry.oracle.com/database/express 21.3.0-xe dcf137aab02d  15 months ago  15.2GB

c:\00_Material_Docker_Kubernetes\ws2\oracle_client>docker history ghcr.io/oracle/oraclelinux8-instantclient:19
IMAGE      CREATED      CREATED BY                           SIZE      COMMENT
b4e4e6f22460  9 days ago  CMD ["sqlplus" "-v"]           0B      buildkit.dockerfile.v0 ←
<missing>  9 days ago  RUN [2 release=19 update=25 /bin/sh -c dnf -...  260MB    buildkit.dockerfile.v0
<missing>  9 days ago  ARG update=25                      0B      buildkit.dockerfile.v0
<missing>  9 days ago  ARG release=19                     0B      buildkit.dockerfile.v0
<missing>  10 days ago  CMD ["/bin/bash"]                0B      buildkit.dockerfile.v0
<missing>  10 days ago  ADD oraclelinux-8-amd64-rootfs.tar.xz / # bu...  273MB    buildkit.dockerfile.v0

c:\00_Material_Docker_Kubernetes\ws2\oracle_client>
```

```
docker pull ghcr.io/oracle/oraclelinux8-instantclient:19
```

Conectarse | Caso de estudio

```
docker run -it --rm --network=ms-network  
ghcr.io/oracle/oraclelinux8-instantclient:19 sqlplus  
SYSTEM/Netec_123@//oracle-db:1521/XE
```

```
docker run -it --rm --network=ms-network  
ghcr.io/oracle/oraclelinux8-instantclient:19 sqlplus  
SYSTEM/Netec_123@//oracle-db:1521 /<XEPDB1|XPEDB1
```

```
docker run -it --rm --network=ms-network  
ghcr.io/oracle/oraclelinux8-instantclient:19 sqlplus  
dkuser/dkpassword@//oracle-db:1521/<XEPDB1|XPEDB1>
```

Conectarse | Caso de estudio

```
c:\00_Material_Docker_Kubernetes\ws2\oracle_client>
c:\00_Material_Docker_Kubernetes\ws2\oracle_client>docker run -it --rm --network=ms-network ghcr.io/oracle/oraclelinux8-instantclient:19 sqlplus system/Netec_123@//oracle-db:1521/XE

SQL*Plus: Release 19.0.0.0.0 - Production on Mon Nov 18 18:41:08 2024
Version 19.25.0.0.0

Copyright (c) 1982, 2024, Oracle. All rights reserved.

Last Successful login time: Mon Nov 18 2024 18:40:21 +00:00

Connected to:
Oracle Database 21c Express Edition Release 21.0.0.0.0 - Production
Version 21.3.0.0.0

SQL> show user;
USER is "SYSTEM"
SQL> show con_name;

CON_NAME
-----
CDB$ROOT
SQL>
SQL> column name format a10
SQL>
SQL> select con_id, name, open_mode from v$pdbs;

  CON_ID NAME      OPEN_MODE
----- -----
    2 PDB$SEED  READ ONLY
    3 XEPDB1    READ WRITE
    4 XEPDB2    READ WRITE

SQL> desc dkuser.peliculas;
ERROR:
ORA-04043: object dkuser.peliculas does not exist

SQL> desc dkuser.favoritos;
ERROR:
ORA-04043: object dkuser.favoritos does not exist

SQL>
```

Las tablas películas y favoritos se encuentran en diferentes pdbs

Conectarse | Caso de estudio

```
SQL>
SQL> alter session set container=xepdb1;

Session altered.

SQL> desc dkuser.peliculas;
Name          Null?    Type
-----        -----
ID           NOT NULL NUMBER
TITULO       NOT NULL VARCHAR2(255)
DIRECTOR     NOT NULL VARCHAR2(255)
GENERO       VARCHAR2(100)
DURACION     NUMBER
FECHA_ESTRENO DATE

SQL> column titulo format a20
SQL> column director format a30
SQL> select id, titulo, director from dkuser.peliculas;

      ID TITULO          DIRECTOR
-----        -----
        1 El Padrino      Francis Ford Coppola
        2 El Padrino II    Francis Ford Coppola
        3 El Padrino III   Francis Ford Coppola

SQL> desc dkuser.favoritos;
ERROR:
ORA-04043: object dkuser.favoritos does not exist

SQL> alter session set container=xepdb2;

Session altered.

SQL> desc dkuser.favoritos;
Name          Null?    Type
-----        -----
ID           NOT NULL NUMBER(19)
PELICULA_ID NOT NULL NUMBER(19)
USUARIO_ID  NOT NULL VARCHAR2(255 CHAR)

SQL>
```

Conectarse | Oracle Database

1. CONNECT user/password@SID
2. CONNECT sys/password AS SYSDBA
3. SHOW USER;
4. SHOW CON_NAME;
5. SELECT con_id, name, open_mode FROM v\$pdbs;
6. ALTER SESSION SET CONTAINER = pdb_name;
7. ALTER SESSION SET CONTAINER = CDB\$ROOT;
8. SELECT username FROM dba_users;
9. SELECT table_name FROM all_tables WHERE owner = 'schema_name';
10. DESC table_name;
11. SHOW PARAMETERS;
12. SELECT * FROM v\$version;
13. EXIT



Resumen

- En esta primera unidad se trató sobre la comunicación entre contenedores Docker, abordando la *dockerización* de microservicios y Oracle, la configuración de redes para conectar ambos, y la verificación de su funcionamiento con *Postman*. Se resolvió la pérdida de datos en Oracle al eliminar contenedores mediante *Docker Volumes*, garantizando la persistencia, y se mostró como conectar clientes de línea de comandos a la base de datos desde un contenedor.



Práctica 1.1. Verificación de entorno de curso

Objetivo:

Al finalizar la práctica, serás capaz de verificar y garantizar que el entorno de trabajo esté completamente configurado y funcional, incluyendo herramientas clave, conexiones, y componentes necesarios para el desarrollo del curso.

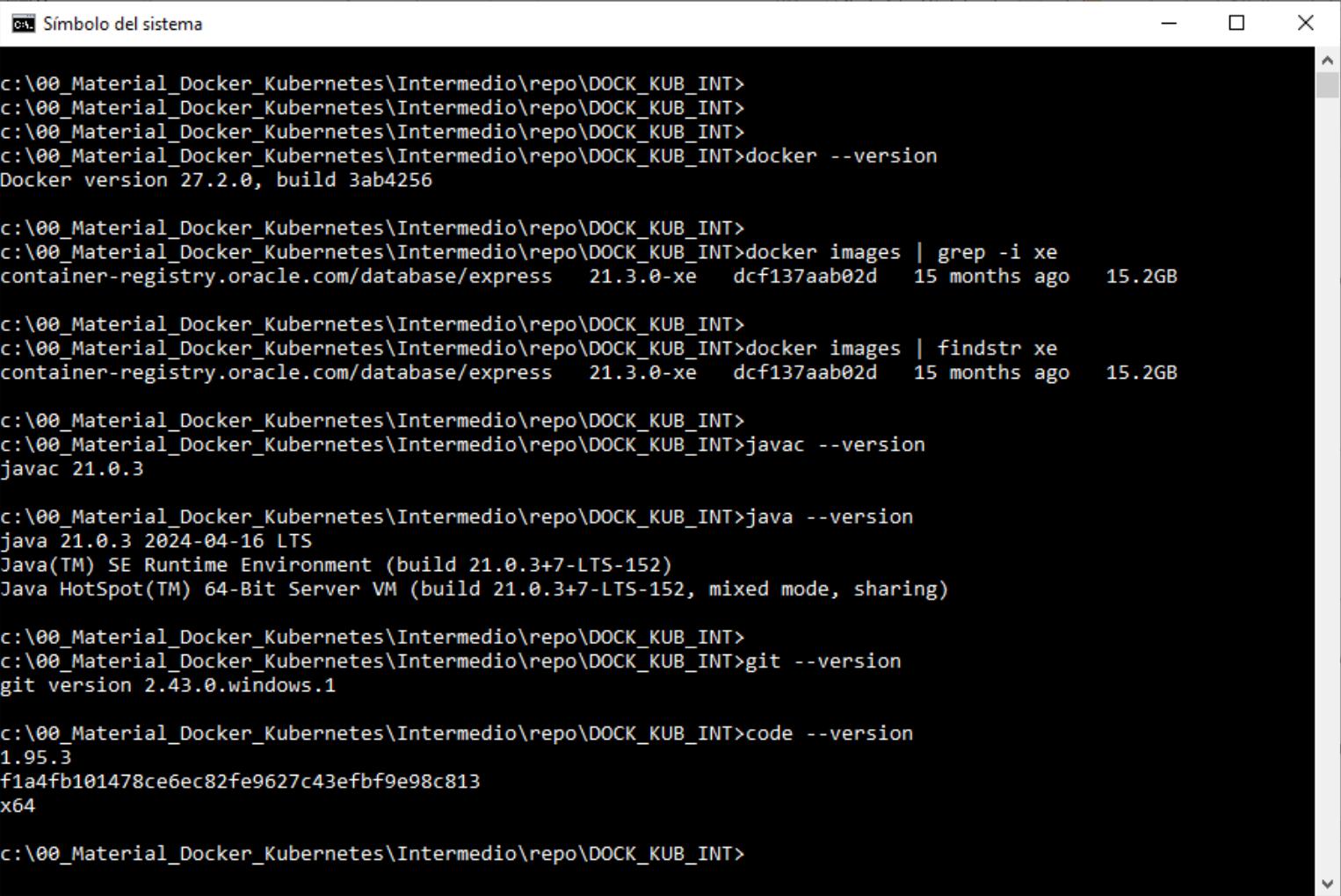
Planteamiento e instrucciones:

Realiza un seguimiento de los pasos indicados en la *Guía de Laboratorios* para llevar a cabo la tarea correspondiente en el siguiente enlace: [Plataforma de Laboratorios | DOCK_KUB_INT](#)



Tiempo para esta actividad:
25 minutos

Resultado esperado



```
c:\00_Material_Docker_Kubernetes\Intermedio\repo\DOCK_KUB_INT>
c:\00_Material_Docker_Kubernetes\Intermedio\repo\DOCK_KUB_INT>
c:\00_Material_Docker_Kubernetes\Intermedio\repo\DOCK_KUB_INT>
c:\00_Material_Docker_Kubernetes\Intermedio\repo\DOCK_KUB_INT>docker --version
Docker version 27.2.0, build 3ab4256

c:\00_Material_Docker_Kubernetes\Intermedio\repo\DOCK_KUB_INT>
c:\00_Material_Docker_Kubernetes\Intermedio\repo\DOCK_KUB_INT>docker images | grep -i xe
container-registry.oracle.com/database/express 21.3.0-xe dcf137aab02d 15 months ago 15.2GB

c:\00_Material_Docker_Kubernetes\Intermedio\repo\DOCK_KUB_INT>
c:\00_Material_Docker_Kubernetes\Intermedio\repo\DOCK_KUB_INT>docker images | findstr xe
container-registry.oracle.com/database/express 21.3.0-xe dcf137aab02d 15 months ago 15.2GB

c:\00_Material_Docker_Kubernetes\Intermedio\repo\DOCK_KUB_INT>
c:\00_Material_Docker_Kubernetes\Intermedio\repo\DOCK_KUB_INT>javac --version
javac 21.0.3

c:\00_Material_Docker_Kubernetes\Intermedio\repo\DOCK_KUB_INT>java --version
java 21.0.3 2024-04-16 LTS
Java(TM) SE Runtime Environment (build 21.0.3+7-LTS-152)
Java HotSpot(TM) 64-Bit Server VM (build 21.0.3+7-LTS-152, mixed mode, sharing)

c:\00_Material_Docker_Kubernetes\Intermedio\repo\DOCK_KUB_INT>
c:\00_Material_Docker_Kubernetes\Intermedio\repo\DOCK_KUB_INT>git --version
git version 2.43.0.windows.1

c:\00_Material_Docker_Kubernetes\Intermedio\repo\DOCK_KUB_INT>code --version
1.95.3
f1a4fb101478ce6ec82fe9627c43efbf9e98c813
x64

c:\00_Material_Docker_Kubernetes\Intermedio\repo\DOCK_KUB_INT>
```

Práctica 1.2. Docker Network

Objetivo:

Al finalizar la práctica, serás capaz de crear y configurar redes en Docker, permitiendo la comunicación eficiente entre contenedores para el correcto funcionamiento de aplicaciones distribuidas.

Planteamiento e instrucciones:

Realiza un seguimiento de los pasos indicados en la *Guía de Laboratorios* para llevar a cabo la tarea correspondiente en el siguiente enlace: [Plataforma de Laboratorios | DOCK_KUB_INT](#)



Tiempo para esta actividad:
25 minutos

Resultado esperado

```
c:\00_Material_Docker_Kubernetes>
c:\00_Material_Docker_Kubernetes>
c:\00_Material_Docker_Kubernetes>docker network ls
NETWORK ID      NAME      DRIVER      SCOPE
87a4a8211a55   bridge    bridge      local
90564692b97c   host      host       local
4ebba8004327   ms-network  bridge      local
c930d999554c   none      null       local

c:\00_Material_Docker_Kubernetes>docker network create --driver bridge ms-curso
3600d7ec2838eae053f35a5b05167def6b328d3b3adb69a7aff946efed846f4d

c:\00_Material_Docker_Kubernetes>docker network ls
NETWORK ID      NAME      DRIVER      SCOPE
87a4a8211a55   bridge    bridge      local
90564692b97c   host      host       local
3600d7ec2838   ms-curso  bridge      local
4ebba8004327   ms-network  bridge      local
c930d999554c   none      null       local

c:\00_Material_Docker_Kubernetes>
```

Práctica 1.3. Docker Volume

Objetivo:

Al finalizar esta actividad, serás capaz de utilizar volúmenes en Docker para gestionar datos persistentes, configurarlos en contenedores.

Planteamiento e instrucciones:

Realiza un seguimiento de los pasos indicados en la *Guía de Laboratorios* para llevar a cabo la tarea correspondiente en el siguiente enlace: [Plataforma de Laboratorios | DOCK_KUB_INT](#)



Tiempo para esta actividad:
25 minutos.

Resultado esperado

```
C:\Users\Netec>docker ps -a | findstr dki
C:\Users\Netec>docker volume ls | findstr dki
C:\Users\Netec>docker volume create dki-volume
dki-volume

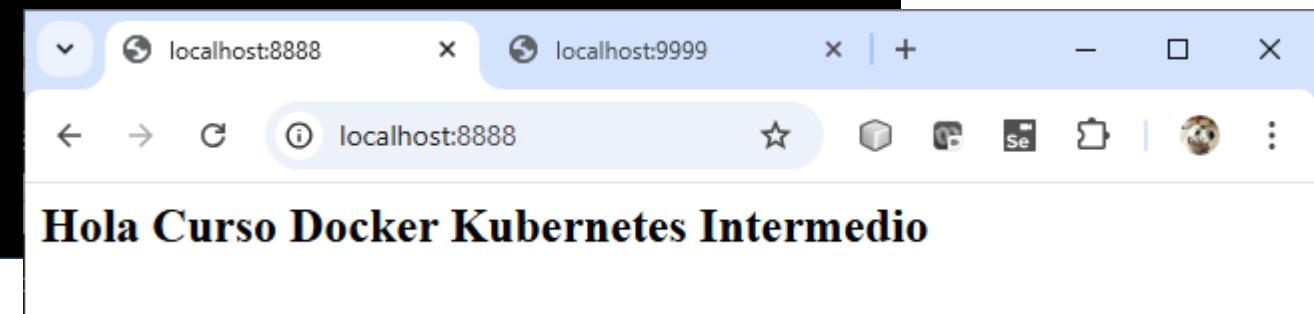
C:\Users\Netec>docker volume ls | findstr dki
local      dki-volume

C:\Users\Netec>docker run -d --name dki_nginx -v dki-volume:/usr/share/nginx/html -p 8888:80 nginx
f9add87d0608314090e9dcf5e0b3558937f59fab936f04ea4b03b6c4fbac3584

C:\Users\Netec>docker exec -it dki_nginx bash
root@f9add87d0608:#
root@f9add87d0608:/# echo "<H2>Hola Curso Docker Kubernetes Intermedio </H2>" > /usr/share/nginx/html/index.html
root@f9add87d0608:/# exit
exit

What's next:
  Try Docker Debug for seamless, persistent debugging tools in any container or image → docker debug dki\_nginx
  Learn more at https://docs.docker.com/go/debug-cli/

C:\Users\Netec>
```



Práctica 1.4. Oracle Container

Objetivo:

Al finalizar esta práctica, serás capaz de configurar y desplegar Oracle Database en un contenedor Docker, utilizando redes y volúmenes para garantizar conectividad y persistencia.

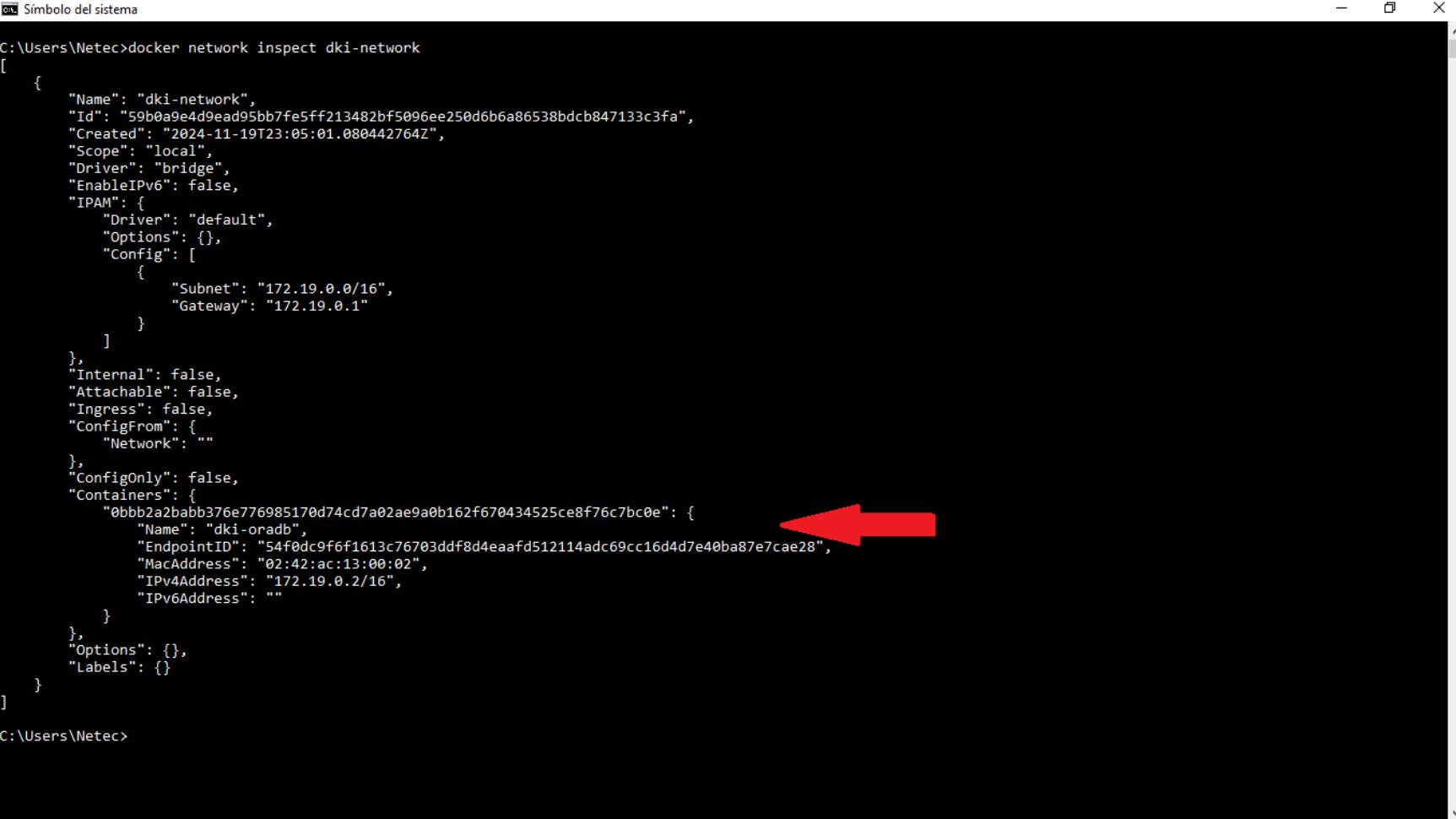
Planteamiento e instrucciones:

Realiza un seguimiento de los pasos indicados en la *Guía de Laboratorios* para llevar a cabo la tarea correspondiente en el siguiente enlace: [Plataforma de Laboratorios | DOCK_KUB_INT](#)



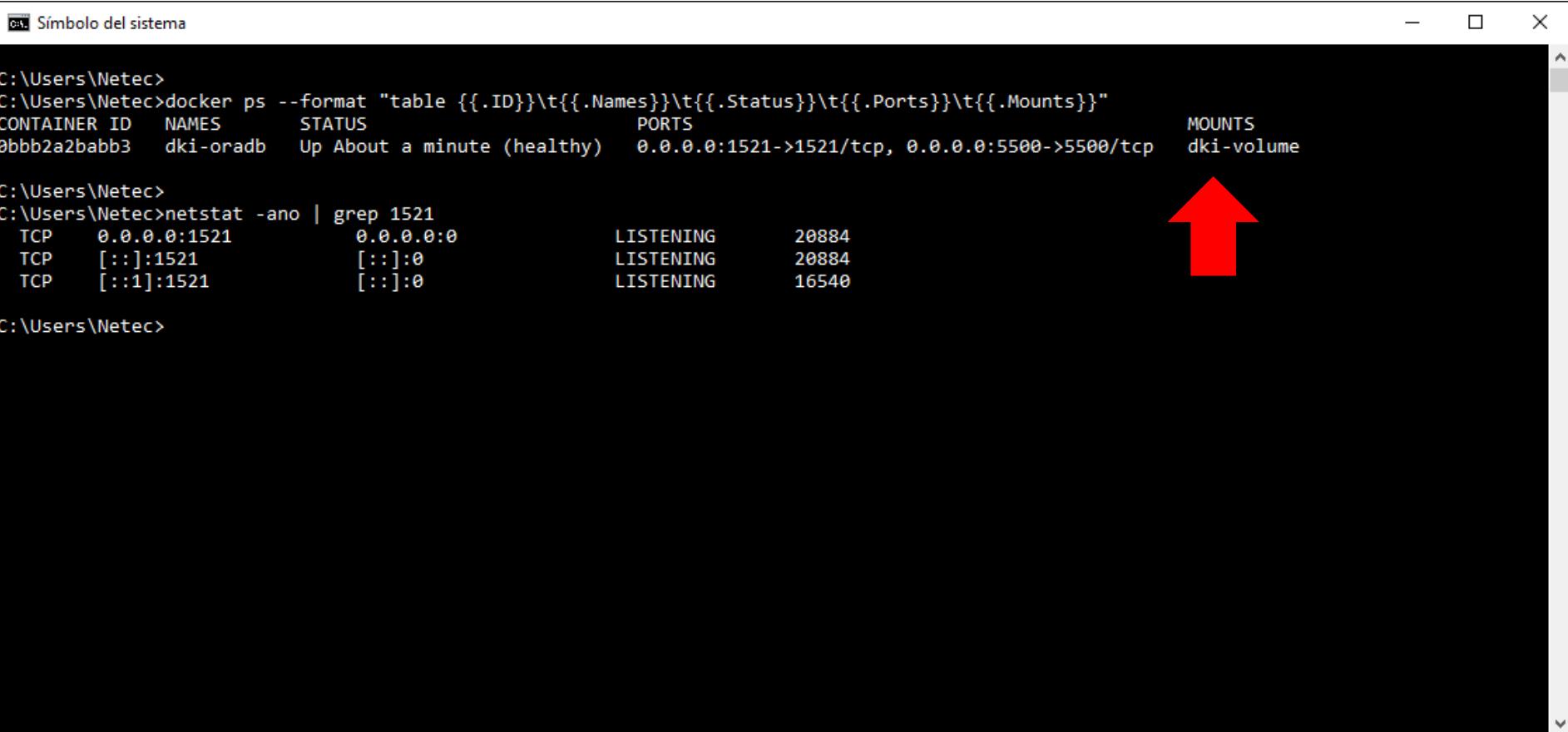
Tiempo para esta actividad:
25 minutos.

Resultado esperado



```
C:\Users\Netec>docker network inspect dki-network
[{"Name": "dki-network", "Id": "59b0a9e4d9ead95bb7fe5ff213482bf5096ee250d6b6a86538bdcb847133c3fa", "Created": "2024-11-19T23:05:01.080442764Z", "Scope": "local", "Driver": "bridge", "EnableIPv6": false, "IPAM": {"Driver": "default", "Options": {}, "Config": [{"Subnet": "172.19.0.0/16", "Gateway": "172.19.0.1"}]}, "Internal": false, "Attachable": false, "Ingress": false, "ConfigFrom": {"Network": ""}, "ConfigOnly": false, "Containers": {""0bbb2a2bab376e776985170d74cd7a02ae9a0b162f670434525ce8f76c7bc0e": {"Name": "dki-oradb", "EndpointID": "54f0dc9f6f1613c76703ddf8d4eaaf512114adc69cc16d4d7e40ba87e7cae28", "MacAddress": "02:42:ac:13:00:02", "IPv4Address": "172.19.0.2/16", "IPv6Address": ""}}, "Options": {}, "Labels": {}}]
```

Resultado esperado



```
C:\Users\Netec>
C:\Users\Netec>docker ps --format "table {{.ID}}\t{{.Names}}\t{{.Status}}\t{{.Ports}}\t{{.Mounts}}"
CONTAINER ID   NAMES      STATUS          PORTS
0bbb2a2bab3   dki-oradb  Up About a minute (healthy)  0.0.0.0:1521->1521/tcp, 0.0.0.0:5500->5500/tcp
                                                               MOUNTS
                                                               dki-volume

C:\Users\Netec>
C:\Users\Netec>netstat -ano | grep 1521
  TCP    0.0.0.0:1521        0.0.0.0:0          LISTENING      20884
  TCP    [::]:1521           [::]:0            LISTENING      20884
  TCP    [::1]:1521          [::]:0            LISTENING      16540

C:\Users\Netec>
```

Práctica 1.5. Caso de estudio (ms-productos & ms-deseos)

Objetivo:

Al finalizar esta práctica, serás capaz de implementar y consumir microservicios en una arquitectura basada en comunicación HTTP, utilizando Spring Boot y Feign como cliente declarativo, para resolver casos de estudio prácticos.

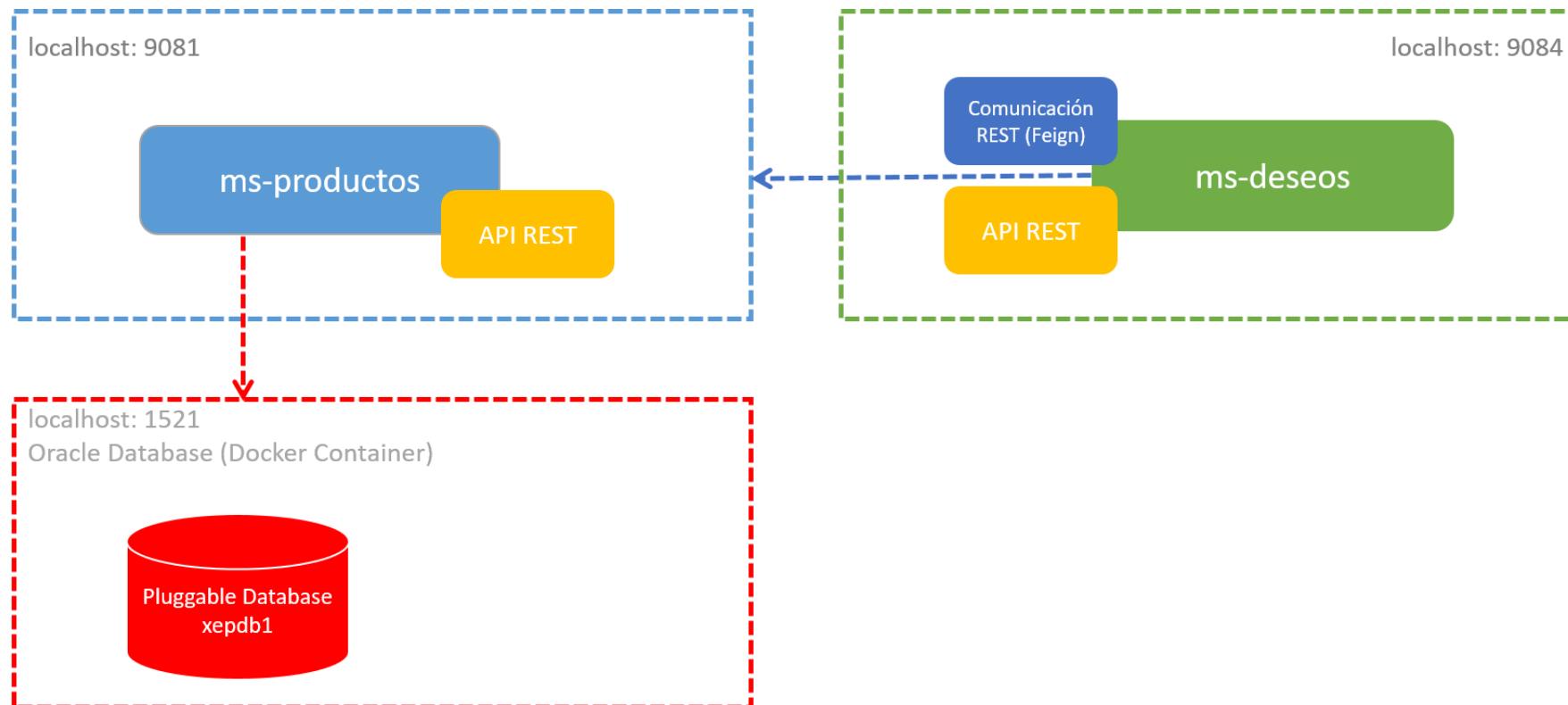
Planteamiento e instrucciones:

Realiza un seguimiento de los pasos indicados en la *Guía de Laboratorios* para llevar a cabo la tarea correspondiente en el siguiente enlace: [Plataforma de Laboratorios | DOCK_KUB_INT](#)



Tiempo para esta actividad:
80 minutos.

Resultado esperado



Resultado esperado

- **Microservicio 1: ms-productos**
- Funcionalidad principal.
 - Gestión de productos con persistencia en base de datos.

Método	Endpoint	Descripción
GET	/productos	Listar todos los productos.
GET	/productos/{id}	Obtener un producto por ID.
POST	/productos	Crear un nuevo producto.
PUT	/productos/{id}	Actualizar un producto existente.
DELETE	/productos/{id}	Eliminar un producto.

```
{  
    id: Long,  
    nombre: String,  
    descripcion: String,  
    precio: Double,  
    stock: Integer  
}
```

```
id (Primary Key, autoincremental)  
nombre (String, único)  
descripcion (String)  
precio (Decimal)  
stock (Entero)
```

Resultado esperado

Microservicio 2: ms-deseos

- Funcionalidad principal
 - Gestión temporal de una lista de deseos basada en productos.
- Servicios Utilizados
 - Consume los endpoints de ms-productos para:
 1. Validar la existencia del producto antes de agregarlo a la lista de deseos.
 2. Obtener detalles del producto para mostrar al usuario.

Método	Endpoint	Descripción
GET	/deseos	Listar todos los productos en la lista.
POST	/deseos/{idProducto}	Agregar un producto a la lista de deseos.
DELETE	/deseos/{idProducto}	Eliminar un producto de la lista

Resultado esperado

```
[  
  {  
    "idProducto": 1,  
    "nombre": "Laptop",  
    "precio": 65000.00,  
    "fechaAgregado": "2024-11-19T10:00:00"  
  },  
  {  
    "idProducto": 3,  
    "nombre": "Mouse Verbatim",  
    "precio": 800.00,  
    "fechaAgregado": "2024-11-19T12:00:00"  
  }  
]
```

Resultado esperado

Arquitectura de comunicación

- **ms-deseos** realiza llamadas HTTP al endpoint GET /productos/{id} de **ms-productos** para verificar si el producto existe y recuperar detalles al agregarlo a la lista.
- Ambos microservicios son independientes, con **ms-deseos** como consumidor.

Práctica 1.6. Contenedores Docker - Caso de estudio

Objetivo:

Al finalizar esta actividad, serás capaz de crear contenedores Docker para dos microservicios existentes a partir de sus archivos JAR.

Planteamiento e instrucciones:

Realiza un seguimiento de los pasos indicados en la *Guía de Laboratorios* para llevar a cabo la tarea correspondiente en el siguiente enlace: [Plataforma de Laboratorios | DOCK_KUB_INT](#)



Tiempo para esta actividad:
25 minutos.

Resultado esperado

```
C:\Símbolo del sistema

C:\Users\Netec>
C:\Users\Netec>docker ps --format "table {{.ID}}\t{{.Names}}\t{{.Status}}\t{{.Ports}}\t{{.Mounts}}"
CONTAINER ID NAMES STATUS PORTS MOUNTS
0bbb2a2bab3 dki-oradb Up 2 hours (healthy) 0.0.0.0:1521->1521/tcp, 0.0.0.0:5500->5500/tcp dki-volume

C:\Users\Netec>docker start ms-productos
ms-productos

C:\Users\Netec>docker start ms-deseos
ms-deseos

C:\Users\Netec>docker ps --filter volume=dki-volume
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS
NAMES
0bbb2a2bab3 container-registry.oracle.com/database/express:21.3.0-xe "/bin/bash -c $ORACL..." 2 hours ago Up 2 hours (healthy) 0.0.0.0:1521->1521/tcp, 0.0.0.0:5500->5500/tcp dki-oradb

C:\Users\Netec>
C:\Users\Netec>docker ps --format "table {{.ID}}\t{{.Names}}\t{{.Networks}}"
CONTAINER ID NAMES NETWORKS
0bbb2a2bab3 dki-oradb dki-network
8f18472e9278 ms-deseos dki-network
e3058587d09c ms-productos dki-network

C:\Users\Netec>
```

Práctica 1.7. Consumo de microservicios - Caso de estudio

Objetivo:

Al finalizar esta práctica, serás capaz de consumir microservicios mediante el uso de herramientas como Postman o implementaciones en código.

Planteamiento e instrucciones:

Realiza un seguimiento de los pasos indicados en la *Guía de Laboratorios* para llevar a cabo la tarea correspondiente en el siguiente enlace: [Plataforma de Laboratorios | DOCK_KUB_INT](#)



Tiempo para esta actividad:
25 minutos.

Resultado esperado

Método	Endpoint	Descripción
GET	/productos	Listar todos los productos.
GET	/productos/{id}	Obtener un producto por ID.
POST	/productos	Crear un nuevo producto.
PUT	/productos/{id}	Actualizar un producto existente.
DELETE	/productos/{id}	Eliminar un producto.

Método	Endpoint	Descripción
GET	/deseos	Listar todos los productos en la lista.
POST	/deseos/{idProducto}	Agregar un producto a la lista de deseos.
DELETE	/deseos/{idProducto}	Eliminar un producto de la lista

Práctica 1.8. Contenedor utilitario

Objetivo:

Al finalizar esta práctica, serás capaz de crear y utilizar un contenedor Docker utilitario con un cliente ligero para conectarte a la base de datos del microservicio ms-productos utilizando SQL*PLUS.

Planteamiento e instrucciones:

Realiza un seguimiento de los pasos indicados en la *Guía de Laboratorios* para llevar a cabo la tarea correspondiente en el siguiente enlace: [Plataforma de Laboratorios | DOCK_KUB_INT](#)



Tiempo para esta actividad:
25 minutos.

Resultado esperado

```
SQL> !clear
SQL> SQL> SQL>
SQL> desc dkuser.productos;
Name                                         Null?    Type
-----
ID                                         NOT NULL NUMBER(19)
DESCRIPCION                                VARCHAR2(255 CHAR)
NOMBRE                                      NOT NULL VARCHAR2(255 CHAR)
PRECIO                                       NOT NULL FLOAT(53)
STOCK                                       NOT NULL NUMBER(10)

SQL>
SQL> column descripcion format a35
SQL> column nombre forma a15
SQL> set line 200
SQL>
SQL> SELECT * FROM dkuser.productos;
ID  DESCRIPCION          NOMBRE      PRECIO  STOCK
-----  -----
 4  Descripcion del Producto C  Producto C    100     50
 2  Descripcion del Producto A  Producto A    100     50
 3  Descripcion del Producto B  Producto B    100     50

SQL> SELECT count(*) from dkuser.productos;
COUNT(*)
-----
 3

SQL> exit
Disconnected from Oracle Database 21c Express Edition Release 21.0.0.0.0 - Production
Version 21.3.0.0.0
C:\Users\Netec>
```

Referencias Bibliográficas

- Oracle Docker Images
<https://github.com/oracle/docker-images>
- Docker Docs <https://docs.docker.com/>
- Docker Guides <https://docs.docker.com/guides/>
- Spring Boot Reference Documentation
<https://docs.spring.io/spring-boot/docs/3.1.0/reference/html/>



Unidad 2

Docker Compose: Orquestador para definir y ejecutar multi-contenedores

Objetivos:

- Comprender los conceptos fundamentales de Docker Compose y su utilidad en la orquestación de múltiples contenedores.
- Configurar un entorno multi-contenedor mediante la creación y edición de un archivo docker-compose.yml.
- Implementar contenedores de bases de datos y microservicios utilizando Docker Compose.
- Ejecutar, detener y gestionar aplicaciones multi-contenedor con los comandos docker-compose up y docker-compose down.

2.1. Introducción Docker Compose

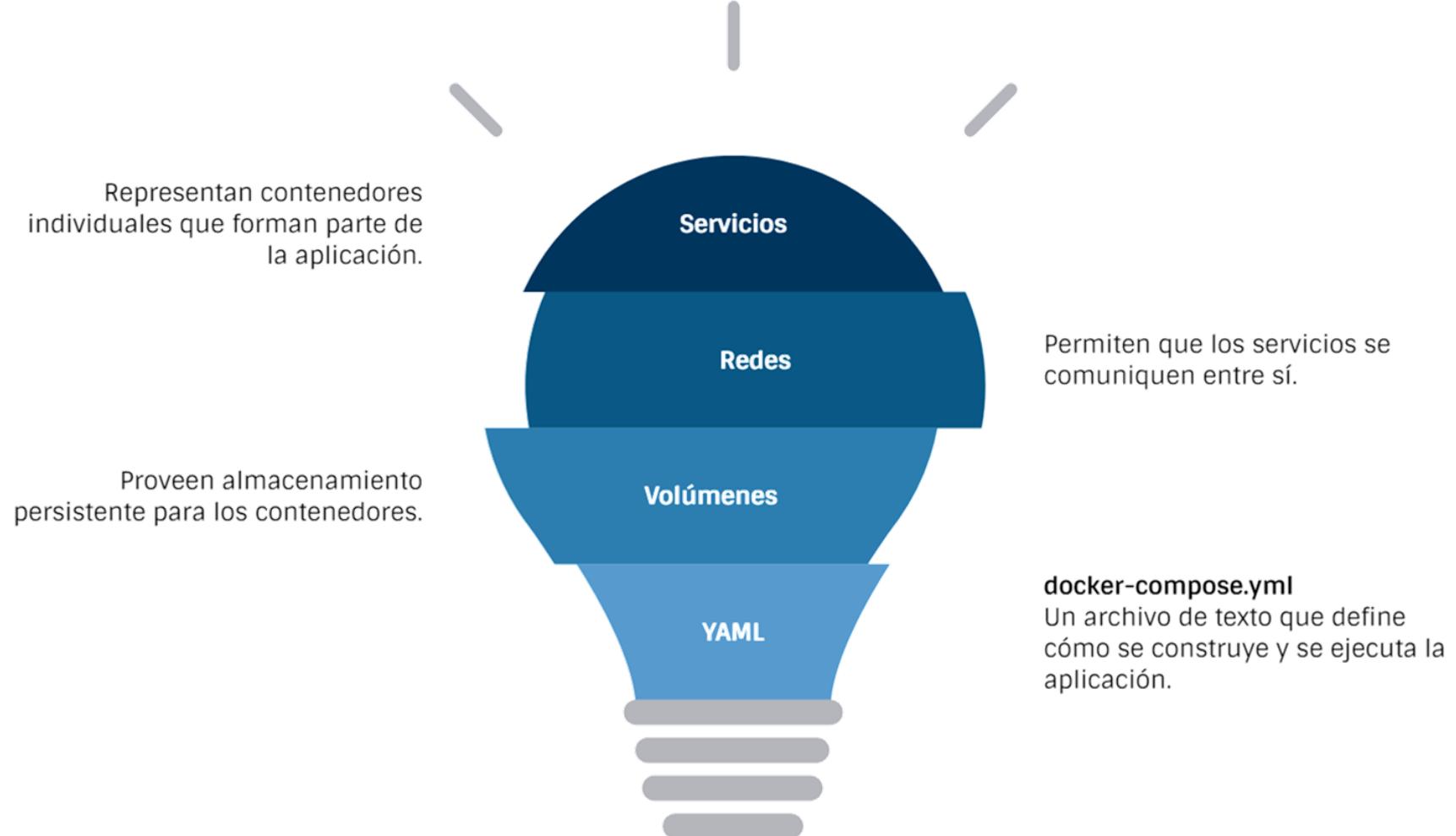
- **¿Qué es Docker Compose?**

Docker Compose es una herramienta para definir y gestionar múltiples contenedores Docker como un solo sistema. Esto permite levantar aplicaciones completas con todos sus servicios relacionados usando un único comando.

Docker Compose no solo revolucionó el uso de contenedores en el desarrollo local, sino que también inspiró conceptos más avanzados de orquestación y gestión de servicios que son la base de herramientas como Kubernetes.

Fig

Introducción a Docker Compose



Introducción a Docker Compose

```
version: "3.9"
services:
  frontend:
    image: nginx:latest
    ports:
      - "8080:80"
    networks:
      - app_network

  backend:
    image: openjdk:21-jdk-slim
    ports:
      - "8081:8080"
    networks:
      - app_network

  database:
    image: mysql:8
    environment:
      MYSQL_ROOT_PASSWORD: root
      MYSQL_DATABASE: app_db
    volumes:
      - db_data:/var/lib/mysql
    networks:
      - app_network

networks:
  app_network:
  db_data:
```



Introducción a Docker Compose | Ventajas

- Gestión centralizada
 - Un único archivo YAML para describir toda la infraestructura de la aplicación.
- Simplicidad
 - Levantar o detener todo el entorno con comandos simples:
 - `docker-compose up`
 - `docker-compose down`.
- Reproducibilidad
 - Facilita que cualquier miembro del equipo configure el mismo entorno de desarrollo.
- Escenarios de prueba
 - Útil para simular entornos productivos localmente.

2.2. Instalando Docker Compose

- **¿Es Docker Compose multiplataforma?**
- Respuesta corta si, es multiplataforma.
- La forma más sencilla y recomendada de obtener Docker Compose es instalar Docker Desktop.
- Docker Desktop incluye Docker Compose junto Docker Engine y Docker CLI, los cuales son requisitos previos de Docker Compose.



Instalando Docker Compose | Linux

Paso 0. Instalar Docker

- En sistemas basados en Debian/Ubuntu:

```
• sudo apt-get update
• sudo apt-get install -y ca-certificates curl gnupg
• sudo mkdir -m 0755 -p /etc/apt/keyrings
• curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
• echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/$(. /etc/os-release && echo "$ID") $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
• sudo apt-get update
• sudo apt-get install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin
```



Instalando Docker Compose | Linux

- Verifica la instalación

- docker --version
- sudo systemctl start docker
- sudo systemctl enable docker

```
root@8b1e8c2be7a0:/#  
root@8b1e8c2be7a0:/# id  
uid=0(root) gid=0(root) groups=0(root)  
root@8b1e8c2be7a0:/#  
root@8b1e8c2be7a0:/# hostname  
8b1e8c2be7a0  
root@8b1e8c2be7a0:/# lsb_release -a  
No LSB modules are available.  
Distributor ID: Ubuntu  
Description:    Ubuntu 20.04.6 LTS  
Release:        20.04  
Codename:       focal  
root@8b1e8c2be7a0:/#  
root@8b1e8c2be7a0:/# docker --version  
Docker version 27.3.1, build ce12230  
root@8b1e8c2be7a0:/#
```



Instalando Docker Compose | Linux

```
Setting up libperl5.30:amd64 (5.30.0-9ubuntu0.5) ...
Setting up libx11-6:amd64 (2:1.6.9-2ubuntu1.6) ...
Setting up libxmluu1:amd64 (2:1.1.3-0ubuntu1) ...
Setting up iptables (1.8.4-3ubuntu2.1) ...
update-alternatives: using /usr/sbin/iptables-legacy to provide /usr/sbin/iptables (iptables) in auto mode
update-alternatives: using /usr/sbin/ip6tables-legacy to provide /usr/sbin/ip6tables (ip6tables) in auto mode
update-alternatives: using /usr/sbin/arptables-nft to provide /usr/sbin/arptables (arptables) in auto mode
update-alternatives: using /usr/sbin/ebtables-nft to provide /usr/sbin/ebtables (ebtables) in auto mode
Setting up openssh-client (1:8.2p1-4ubuntu0.11) ...
Setting up libxext6:amd64 (2:1.3.4-0ubuntu1) ...
Setting up perl (5.30.0-9ubuntu0.5) ...
Setting up docker-ce (5:27.3.1-1~ubuntu.20.04~focal) ...
invoke-rc.d: could not determine current runlevel
invoke-rc.d: policy-rc.d denied execution of start.
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /lib/systemd/system/docker.service.
Created symlink /etc/systemd/system/sockets.target.wants/docker.socket → /lib/systemd/system/docker.socket.
Setting up xauth (1:1.1-0ubuntu1) ...
Setting up liberror-perl (0.17029-1) ...
Setting up git (1:2.25.1-1ubuntu3.13) ...
Processing triggers for libc-bin (2.31-0ubuntu9.16) ...
Processing triggers for systemd (245.4-4ubuntu3.24) ...
Processing triggers for mime-support (3.64ubuntu1) ...
root@8b1e8c2be7a0:/#
root@8b1e8c2be7a0:/#
root@8b1e8c2be7a0:/# docker --version
Docker version 27.3.1, build ce12230
root@8b1e8c2be7a0:/#
root@8b1e8c2be7a0:/#
```

Instalando Docker Compose | Linux

Paso 1: Descargar Docker Compose

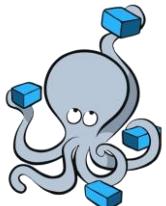
- Verifica la última versión disponible en la página oficial de Docker Compose.
- Ejecuta el siguiente comando para descargar el binario (reemplaza VERSION con la última versión disponible):
`sudo curl -L "https://github.com/docker/compose/releases/download/VERSION/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose`

Paso 2: Agregar permisos de ejecución

- `sudo chmod +x /usr/local/bin/docker-compose`

Paso 3: Verificar la instalación

- `docker-compose --version`



Instalando Docker Compose | Linux

```
root@8b1e8c2be7a0:/ws#
root@8b1e8c2be7a0:/ws# uname -s
Linux
root@8b1e8c2be7a0:/ws# uname -s
Linux
root@8b1e8c2be7a0:/ws# ls -l /usr/local/bin/docker-compose
-rwxr-xr-x 1 root root 64044374 Nov 22 20:23 /usr/local/bin/docker-compose
root@8b1e8c2be7a0:/ws# docker-compose --version
Docker Compose version v2.30.3
root@8b1e8c2be7a0:/ws#
root@8b1e8c2be7a0:/ws# docker-compose config
WARN[000] /ws/docker-compose.yml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential confusion
name: ws
services:
  test-service:
    image: nginx:latest
    networks:
      default: null
    ports:
      - mode: ingress
        target: 80
        published: "8080"
        protocol: tcp
  networks:
    default:
      name: ws_default
root@8b1e8c2be7a0:/ws#
```



Instalando Docker Compose | Windows

Paso 1: Descarga Docker Desktop

- Asegúrate de que Docker Desktop está instalado, ya que incluye Docker Compose por defecto. Puedes descargar Docker Desktop desde la página oficial.
- Durante la instalación, habilita la opción para incluir Docker Compose.

Paso 2: Verificar la instalación

- Abre PowerShell o CMD y ejecuta:
 - `docker-compose --version`
 - `docker-compose --help`

docker compose
vs.
docker-compose



Instalando Docker Compose | Windows

```
C:\Users\Netec> docker-compose --version
Docker Compose version v2.29.2-desktop.2

C:\Users\Netec> docker-compose --help
Usage: docker compose [OPTIONS] COMMAND
Define and run multi-container applications with Docker

Options:
  --all-resources           Include all resources, even those not
                           used by services
  --ansi string             Control when to print ANSI control
                           characters ("never"|"always"|"auto")
                           (default "auto")
  --compatibility          Run compose in backward compatibility mode
  --dry-run                 Execute command in dry run mode
  --env-file stringArray    Specify an alternate environment file
  -f, --file stringArray    Compose configuration files
  --parallel int            Control max parallelism, -1 for
                           unlimited (default -1)
  --profile stringArray     Specify a profile to enable
  --progress string          Set type of progress output (auto,
                           tty, plain, json, quiet) (default "auto")
  --project-directory string Specify an alternate working directory
                           (default: the path of the, first
                           specified, Compose file)
  -p, --project-name string Project name

Commands:
  attach      Attach local standard input, output, and error streams to a service's running container
  build       Build or rebuild services
  config      Parse, resolve and render compose file in canonical format
  cp          Copy files/folders between a service container and the local filesystem
  create      Creates containers for a service
  down        Stop and remove containers, networks
  events      Receive real time events from containers
  exec        Execute a command in a running container
  images      List images used by the created containers
  kill        Force stop service containers
  logs        View output from containers
  ls          List running compose projects
  pause       Pause services
```



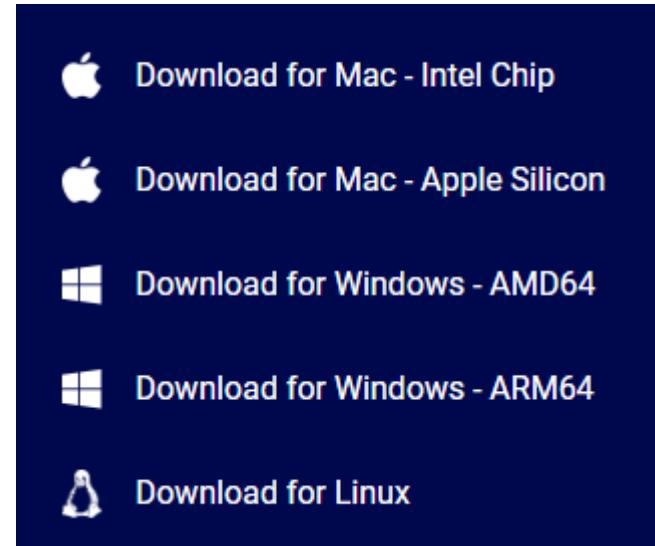
Instalando Docker Compose | macOS

Paso 1: Descargar Docker Desktop

- Descarga e instala Docker Desktop desde la página oficial.
- <https://www.docker.com/products/docker-desktop/>

Paso 2: Verificar la instalación

- Abre la Terminal y ejecuta:
 - docker-compose --version
 - docker-compose --help



Instalando Docker Compose | Problemas

Problema 1: command not found

- **Causa:** Docker Compose no se encuentra en la ruta del sistema.
- **Solución (Linux):**
 - Asegúrate de que el binario está en /usr/local/bin/.
 - Verifica los permisos con:
 - `ls -l /usr/local/bin/docker-compose`
 - Si el problema persiste, agrega /usr/local/bin al \$PATH:
 - `export PATH=$PATH:/usr/local/bin`

Instalando Docker Compose | Problemas

Problema 2: Docker Compose no está instalado aún después de configurar Docker Desktop

- **Causa:** Una instalación fallida o incompleta.
- **Solución:**
 - Reinstala Docker Desktop y asegúrate de que Docker Compose está habilitado.
 - `docker-compose --version`
 - `docker-compose --help`

Instalando Docker Compose | Problemas

Problema 3: Error de versión antigua en Linux

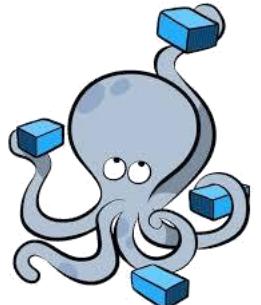
- **Causa:** Una versión anterior de Docker Compose está instalada.
- **Solución:**
 - Elimina la versión antigua con:
 - `sudo rm /usr/local/bin/docker-compose`
 - Instala la última versión siguiendo los pasos indicados.

2.3. Creando el archivo Docker Compose

-
1. ¿Qué herramienta se utiliza para definir y ejecutar múltiples contenedores Docker de manera sencilla mediante un archivo de configuración YAML, y cómo facilita la gestión de entornos de desarrollo y despliegues?
 2. ¿Cómo se llama el archivo de configuración en formato YAML que utiliza Docker Compose para orquestar servicios, volúmenes y redes?

Docker Compose

docker-compose.yml



Creando el archivo | YAML

1. YAML **no permite tabulaciones**; siempre se deben usar espacios para la indentación.
2. La cantidad de espacios debe ser uniforma para cada nivel jerárquico.
3. Los dos puntos que separan clave y valor deben ir seguidos de un espacio.
4. Las listas deben estar alineadas al mismo nivel y cada elemento debe de comenzar con un guion seguido de un espacio.
5. Mezclar sintaxis de mapas y listas en el mismo nivel puede causar errores.
6. Olvidar cerrar comillas o usarlas de forma inconsistente.
7. YAML no permite claves duplicadas en el mismo nivel.

Creando el archivo | YAML

8. Caracteres especiales como : en las cadenas deben ir entre comillas.
9. Valores como **yes**, **no**, **on**, **off** son tratados como booleanos y deben manejarse con cuidado.
10. No validar el archivo YAML puede generar errores difíciles de restear.

Creando el archivo | YAML

```
services:  
  app:  
    image: nginx # Diferente número de espacios
```

```
key: yes # Interpreta 'yes' como booleano  
key: "yes" # Correcto, es una cadena
```

```
key:value # Incorrecto  
key: value # Correcto
```

```
key:  
  - item1  
    subkey: value # Error: mapa en una lista sin indentación adecuada
```

Creando el archivo | YAML

```
key: value1  
key: value2 # Clave duplicada
```

```
- item1  
- item2 # Mal alineado
```

```
key: "value # Comillas no cerradas
```

```
key: value:123 # Error por no usar comillas  
key: "value:123" # Correcto
```

Creando el archivo | YAML

- **Recomendaciones** para evitar errores en YAML.
 - Usa un editor que resalte la sintaxis y advierta sobre errores comunes.
 - Configura el editor para usar espacios en lugar de tabulaciones.
 - Valida los archivos YAML con herramientas como [YAML Lint](#) o [linters](#) integrados en IDEs.
 - Mantén una estructura clara y consistente en la jerarquía.
 - Usa comentarios para explicar configuraciones complejas.

*Una herramienta **Lint** es un software que analiza el código fuente de un programa para identificar posibles errores, malas prácticas, inconsistencias o problemas de estilo que podrían afectar su funcionalidad, rendimiento o legibilidad.*

Creando el archivo | Services

- La sección **services** define los contenedores que se ejecutarán. Cada servicio representa una **instancia de un contenedor** y permite especificar configuraciones como:
 - **imagen**: La imagen Docker base que usará el servicio.
 - **build**: Ruta donde se encuentra el **Dockerfile** si es necesario construir la imagen.
 - **ports**: Mapea puertos entre el contenedor y el host.
 - **depends_on**: Establece el orden de inicio de los servicios.
 - **environment**: Define variables de entorno o ambiente para el contenedor.
 - **volumes**: Conecta almacenamiento persistente.
 - **healthcheck**: Se utiliza para definir comandos que verifican si un contenedor está funcionando correctamente y listo para recibir solicitudes.

Creando el archivo | Services

```
version: '3.9' # Define la versión de Docker Compose

services:
  app:
    image: nginx:latest # Imagen base del servicio
    ports:
      - "8080:80" # Mapea el puerto 8080 del host al puerto 80 del contenedor
    volumes:
      - data:/usr/share/nginx/html # Monta un volumen para persistir datos
    networks:
      - webnet # Conecta el servicio a una red personalizada
```

Creando el archivo | Volumes

- Los volúmenes se utilizan para persistir datos generados por los contenedores, evitando que se pierdan al reiniciarlos.
- Estos se declaran en una sección aparte o directamente dentro del servicio.
- Ejemplos:
 - **Volúmenes anónimos:** No tienen un nombre explícito.
 - **Volúmenes nombrados:** Son reutilizables y se nombran explícitamente.

```
volumes:  
  data: # Declara un volumen para almacenar datos persistentes
```

Creando el archivo | Networks

- Define redes virtuales para los servicios, permitiendo que los contenedores se comuniquen entre sí de manera segura y eficiente.
- Docker Compose crea una red por defecto, pero también es posible configurarlas manualmente.

Creando el archivo | Networks

Ejemplo de tipos de red:

- **Bridge (predeterminada)**: Aísla los contenedores dentro de la red.
- **Host**: El contenedor comparte la red del host.
- **Overlay**: Para comunicaciones en clústeres.

```
networks:  
  webnet: # Declara una red llamada 'webnet'  
    driver: bridge # Utiliza el driver predeterminado 'bridge'
```

```
networks:  
  dki-network: # Declara una red llamada 'dki-network'  
    external: true # No se crea en el contexto del docker-Compose.yml
```

Creando el archivo | healthcheck

- Puedes configurar un chequeo de salud para cada servicio usando la clave **healthcheck**.
- Esto permite monitorear la salud de los contenedores según una condición específica.
 - Como verificar si una aplicación está respondiendo.

```
version: "3.8"
services:
  web:
    image: my-app:latest
    ports:
      - "8080:8080"
    healthcheck:
      test: ["CMD", "curl", "-f", "http://localhost:8080"]
      interval: 30s
      timeout: 10s
      retries: 3
      start_period: 10s
```

curl o wget

Creando el archivo | healthcheck

- Una vez definido el *healthcheck*, puedes consultar el estado de los contenedores con:
 - `docker-compose ps`
 - En la columna *State*, aparecerá el estado del contenedor:
 - **healthy**
 - Indica que el contenedor pasó exitosamente el chequeo de salud definido en `docker-compose.yml`.
 - **unhealthy**
 - El contenedor falló en el chequeo de salud varias veces, según los valores configurados en *retries*.

Creando el archivo | healthcheck

- **starting**
 - El contenedor aún no ha completado los chequeos de salud iniciales. Esto ocurre dentro del período definido por *start_period*.
- Para inspeccionar más detalles sobre el estado del contenedor, puedes usar:
 - `docker inspect <nombre o id del contenedor>`
 - `docker logs <nombre o id del contenedor>`

Creando el archivo | docker-compose ps

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
abcd1234efgh	ms1-image	"java -jar app.jar"	30 seconds ago	Up 25 seconds (healthy)	0.0.0.0:8081->8080/tcp	docker_ms1_1
ijk15678mnop	ms2-image	"java -jar app.jar"	30 seconds ago	Up 25 seconds (healthy)	0.0.0.0:8082->8080/tcp	docker_ms2_1
qrst9012uvwxyz	ms3-image	"java -jar app.jar"	30 seconds ago	Up 25 seconds (healthy)	0.0.0.0:8083->8080/tcp	docker_ms3_1
yzab3456cdef	ms4-image	"java -jar app.jar"	30 seconds ago	Up 25 seconds (healthy)	0.0.0.0:8084->8080/tcp	docker_ms4_1



*El estado **healthy** aparecerá solo después de que el comando definido en `healthcheck` sea exitoso **dentro** del contenedor.*

2.4. Definiendo contenedores de BBDD Oracle

- **Oracle Database** es una solución de base de datos empresarial robusta y ampliamente utilizada. Utilizar contenedores para su despliegue permite agilizar el proceso de configuración y administración en entornos de desarrollo o pruebas.
- **Docker Compose** facilita la definición de servicios interconectados, incluyendo bases de datos, a través de un archivo YAML. Esto permite establecer configuraciones reproducibles, estandarizadas y portables.

Definiendo contenedores de BBDD Oracle

- Aunque Oracle Database en contenedores puede ser utilizado en producción, su adopción es más frecuente en entornos de desarrollo y pruebas. Esto se debe a varios factores:
 - La facilidad para crear y destruir instancias sin impactar sistemas de producción.
 - La capacidad de compartir configuraciones específicas mediante archivos YAML o Dockerfiles.
 - Uso para entrenar o validar nuevas funcionalidades sin comprometer infraestructuras críticas.
- Un aspecto importante de Oracle Database en Docker es su relación con el **licenciamiento**:
 - Oracle permite usar sus imágenes oficiales en contenedores, pero las versiones completas, como Enterprise Edition, requieren un licenciamiento adecuado incluso en entornos contenerizados.



Definiendo contenedores de BBDD Oracle

```
version: '3.9'
services:
  oracle-db:
    image: container-registry.oracle.com/database/express:21.3.0-xe
    container_name: Oracle-db
    ports:
      - "1521:1521" # Puerto para conexiones SQL*Net
      - "5500:5500" # Puerto para Oracle EM Express
    environment:
      ORACLE_SID: XE # Nombre de la instancia
      ORACLE_PDB: XEPDB1 # Nombre de la base de datos pluggable
      ORACLE_PWD: Netec_123 # Contraseña del usuario SYS y SYSTEM
    volumes:
      - ms-volume:/opt/oracle/oradata # Persistencia de datos
      - ./custom-scripts:/docker-entrypoint-initdb.d # Scripts personalizados
    networks:
      - ms-network

volumes:
  ms-volume:
    driver: local

networks:
  ms-network:
    driver: bridge
```

Definiendo contenedores de BBDD Oracle

- **Paso 1. Preparar el entorno.**
 - Instalar Docker y Docker Compose.
 - Crear el archivo [docker-compose.yml](#) en el directorio del proyecto.
- **Paso 2. Descarga la imagen.**
 - Si la imagen está en el [Oracle Container Registry](#), realizar el login y extraer la imagen.

```
docker login container-registry.oracle.com
docker pull container-registry.oracle.com/database/express:21.3.0-xe
```

Definiendo contenedores de BBDD Oracle

- **Paso 3. Levantar el servicio.**
 - Ejecutar Docker Compose para iniciar el contenedor.

```
docker-compose up -d
```

- **Paso 4. Verifica el contenedor.**
 - Confirmar que el contenedor está en ejecución.
 - Conectarse a la base de dato utilizando herramientas como SQL*Plus o SQL Developer, apuntando al puerto 1521 del host.

```
docker ps
```

Definiendo contenedores de BBDD Oracle

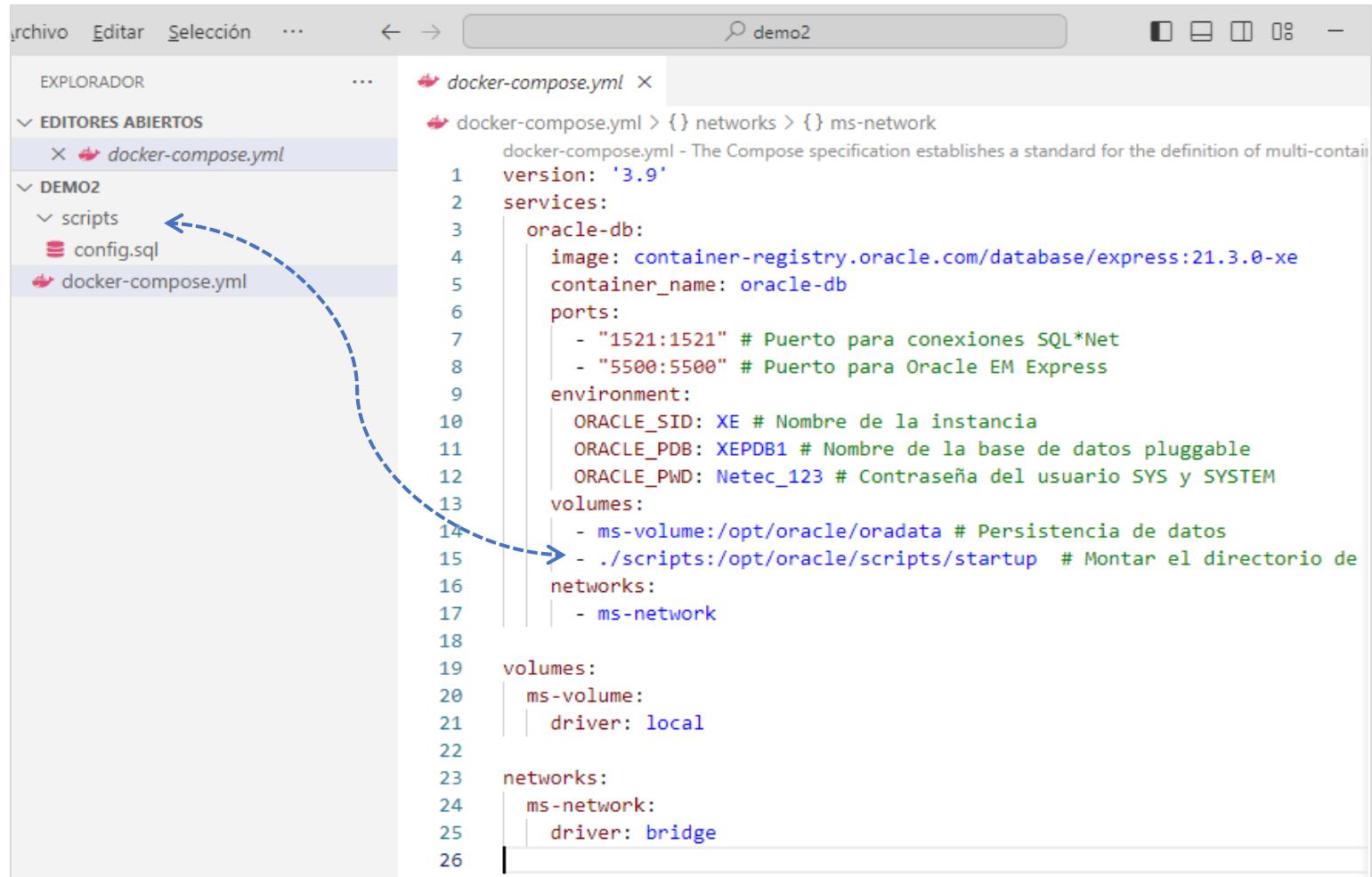
- **Paso 5. Persistencia de datos.**
 - Detener el contenedor.

```
docker-compose down
```

- Confirmar que los datos persisten reiniciando el contenedor.

```
docker-compose up -d
```

Definiendo contenedores de BBDD Oracle



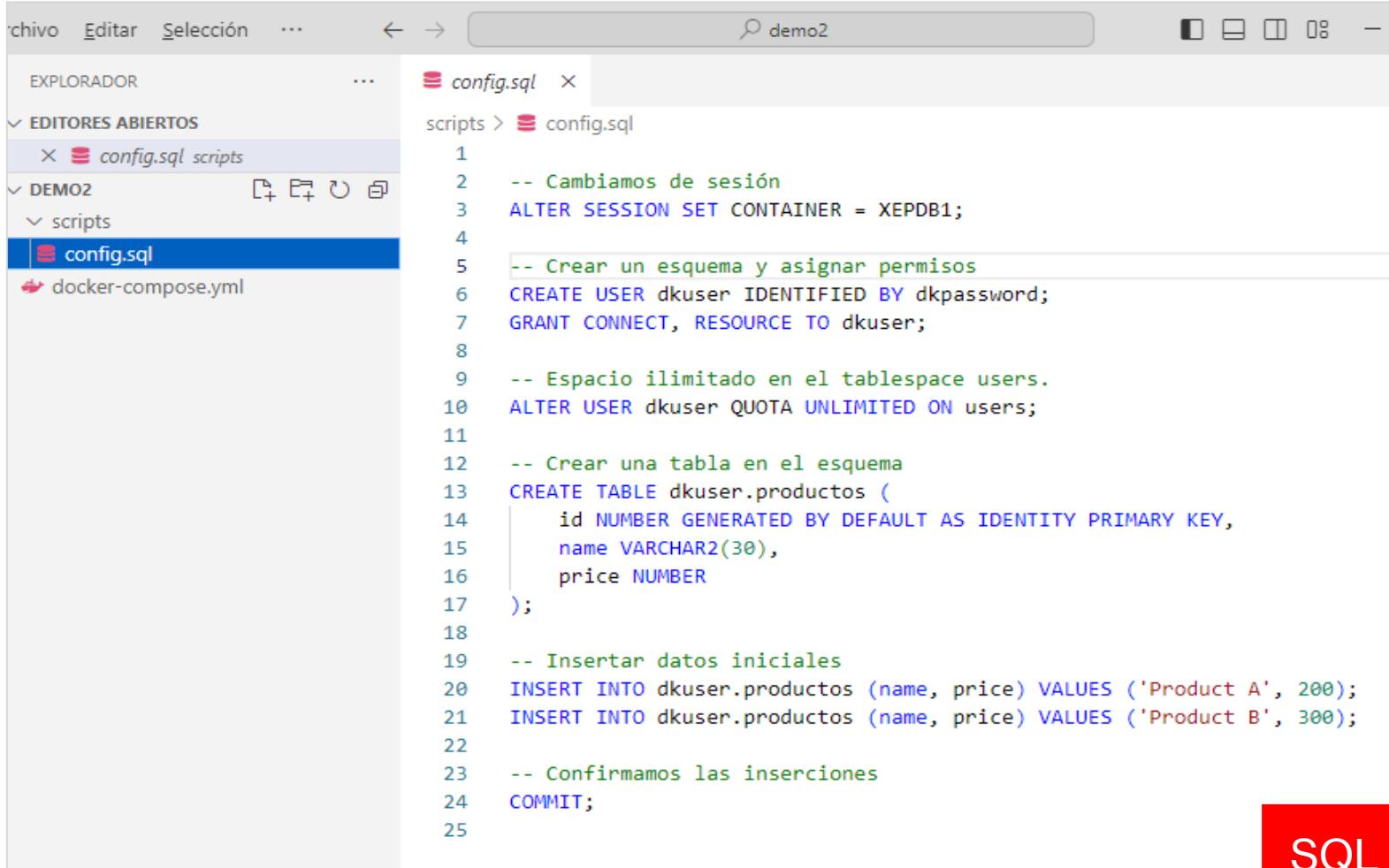
The screenshot shows a code editor interface with a navigation bar at the top. The main area displays a Docker Compose configuration file named `docker-compose.yml`. The file defines a service named `oracle-db` which runs an Oracle Express database container. The configuration includes details like port mappings, environment variables, volumes, and network settings. A dashed blue arrow points from the left margin of the code editor towards the `volumes:` section of the `oracle-db` service definition.

```
version: '3.9'
services:
  oracle-db:
    image: container-registry.oracle.com/database/express:21.3.0-xe
    container_name: oracle-db
    ports:
      - "1521:1521" # Puerto para conexiones SQL*Net
      - "5500:5500" # Puerto para Oracle EM Express
    environment:
      ORACLE_SID: XE # Nombre de la instancia
      ORACLE_PDB: XEPDB1 # Nombre de la base de datos pluggable
      ORACLE_PWD: Netec_123 # Contraseña del usuario SYS y SYSTEM
    volumes:
      - ms-volume:/opt/oracle/oradata # Persistencia de datos
      - ./scripts:/opt/oracle/scripts/startup # Montar el directorio de
        scripts
    networks:
      - ms-network

volumes:
  ms-volume:
    driver: local

networks:
  ms-network:
    driver: bridge
```

Definiendo contenedores de BBDD Oracle



The screenshot shows a code editor interface with the following details:

- Toolbar:** Archivo, Editar, Selección, ..., Back, Forward, Search bar (demo2), Window controls.
- Left Sidebar (Explorador):** EDITORES ABIERTOS (config.sql scripts), DEMO2 (scripts, config.sql selected), docker-compose.yml.
- Right Editor Area:** config.sql (scripts > config.sql)
- Script Content (config.sql):**

```
1  -- Cambiamos de sesión
2  ALTER SESSION SET CONTAINER = XEPDB1;
3
4  -- Crear un esquema y asignar permisos
5  CREATE USER dkuser IDENTIFIED BY dkpassword;
6  GRANT CONNECT, RESOURCE TO dkuser;
7
8
9  -- Espacio ilimitado en el tablespace users.
10 ALTER USER dkuser QUOTA UNLIMITED ON users;
11
12 -- Crear una tabla en el esquema
13 CREATE TABLE dkuser.productos (
14     id NUMBER GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
15     name VARCHAR2(30),
16     price NUMBER
17 );
18
19 -- Insertar datos iniciales
20 INSERT INTO dkuser.productos (name, price) VALUES ('Product A', 200);
21 INSERT INTO dkuser.productos (name, price) VALUES ('Product B', 300);
22
23 -- Confirmamos las inserciones
24 COMMIT;
25
```

SQL o PL/SQL

2.5. Añadiendo contenedores de microservicios

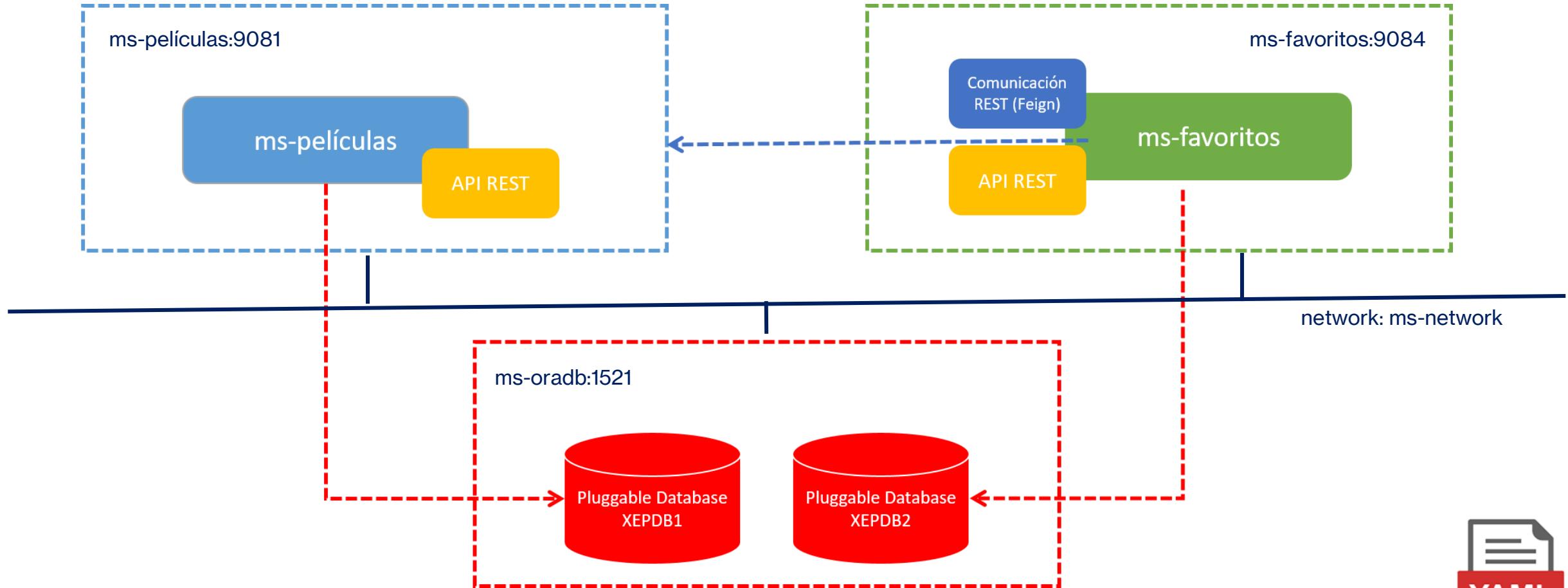
- En esta sección, el objetivo es demostrar cómo configurar un entorno en el que dos microservicios (ms-películas & ms-favoritos) interactúen entre sí y con una base de datos Oracle en un entorno contenedorizado.
- La integración de los microservicios con Docker Compose asegura una arquitectura modular y fácilmente escalable.
- La definición de redes facilita la comunicación y seguridad entre contenedores, mientras que el uso de imágenes preconstruidas o construcciones en tiempo de despliegue ofrece flexibilidad según el entorno.

Añadiendo contenedores de microservicios

- Docker Compose permite definir y ejecutar aplicaciones con **múltiples contenedores** mediante un archivo YAML.
- Este archivo especifica:
 - **Servicios:**
 - Cada contenedor se configura como un servicio.
 - **Redes:**
 - La comunicación entre contenedores se gestiona mediante redes Docker definidas explícitamente.
 - **Volúmenes:**
 - Permiten la persistencia de datos compartidos entre contenedores o en el host.



Añadiendo contenedores de microservicios



Añadiendo contenedores de microservicios

```
services:  
  ms-peliculas:  
    image: ms-peliculas:1.0.0  
    container_name: ms-peliculas  
    build:  
      context: ./ms-peliculas  
    ports:  
      - "9081:9081"  
    environment:  
      - SPRING_DATASOURCE_URL=jdbc:oracle:thin:@ms-oradb:1521/XEPDB2  
      - SPRING_DATASOURCE_USERNAME=dkuser  
      - SPRING_DATASOURCE_PASSWORD=dkpassword  
    networks:  
      - ms-network
```



Añadiendo contenedores de microservicios

```
ms-favoritos:  
  image: ms-favoritos:1.0.0  
  container_name: ms-favoritos  
  build:  
    context: ./ms-favoritos  
  ports:  
    - "9084:9084"  
  environment:  
    - MS_PELICULAS_URL=http://ms-peliculas:9081  
  networks:  
    - ms-network
```



Añadiendo contenedores de microservicios

```
oracledb:  
  image: container-registry.oracle.com/database/express:21.3.0-xe  
  container_name: ms-oradb  
  ports:  
    - "1521:1521"  
  environment:  
    - ORACLE_PDB=XEPDB1  
    - ORACLE_USER=dkuser  
    - ORACLE_PASSWORD=dkpassword  
  networks:  
    - ms-network  
  
networks:  
  ms-network:  
    driver: bridge
```



Añadiendo contenedores de microservicios

- Las redes permiten que los contenedores se comuniquen entre sí de manera segura y aislada.
- **Redes en Docker Compose:**
 - Se definen explícitamente para garantizar que solo los contenedores necesarios puedan comunicarse entre sí.
 - En este caso, la red ms-network conecta ms-peliculas, ms-favoritos y ms-oradb.
- **Resolución de nombres:**
 - Docker asigna automáticamente nombres DNS a los contenedores según sus nombres de servicio. Por ejemplo:
 - ms-peliculas se puede alcanzar como <http://ms-peliculas:9091>.
 - La base de datos se puede alcanzar como ms-oradb desde ms-peliculas o ms-favoritos.

2.6. Ejecutando todo con docker-compose up y down para detener y eliminar todo

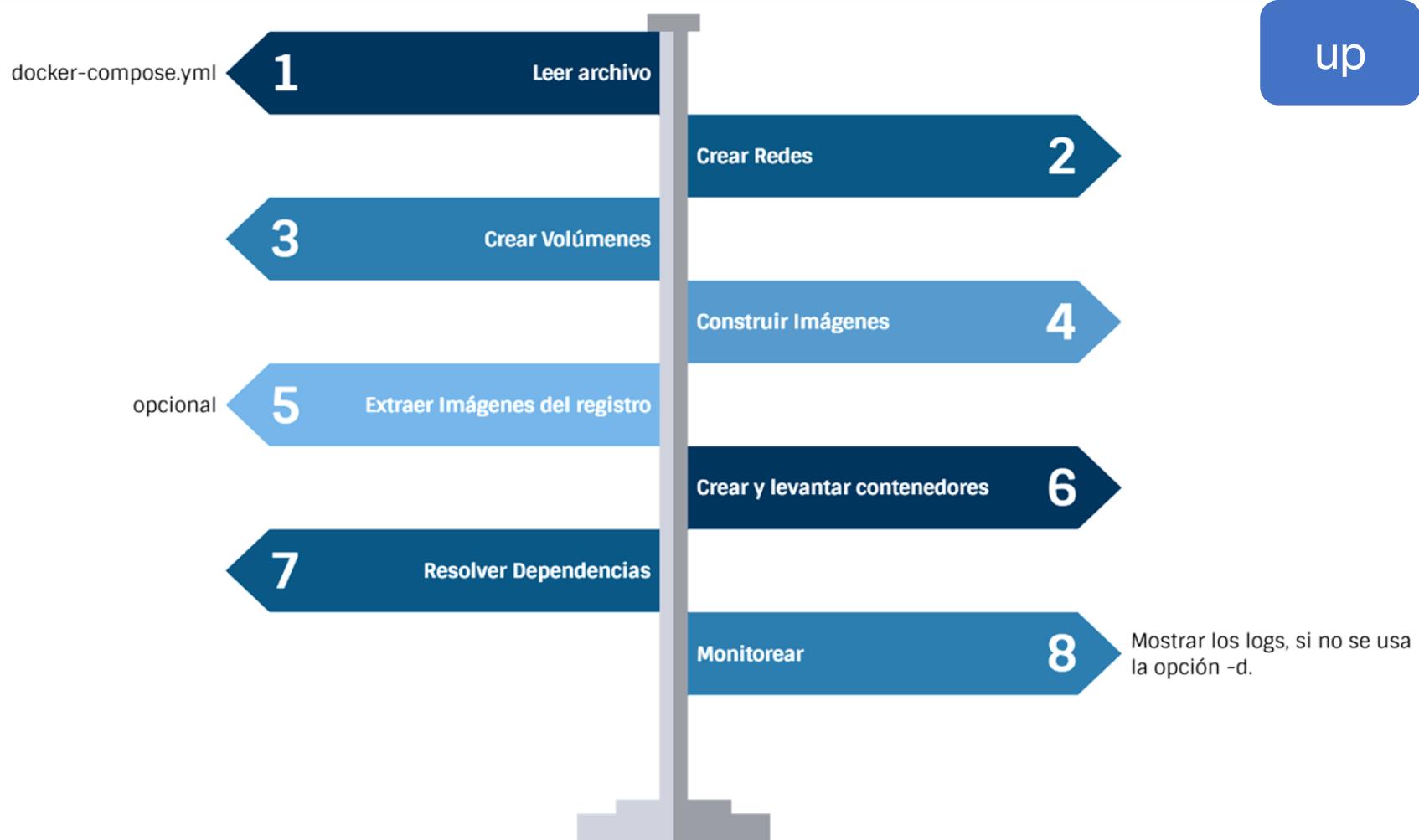
docker-compose up

- Este comando levanta todos los servicios definidos en el archivo docker-compose.yml
- Puede incluir las opciones de:
 - `-d` para ejecutar los servicios en segundo plano.
 - `--build` para forzar la construcción de las imágenes antes de iniciar los servicios.

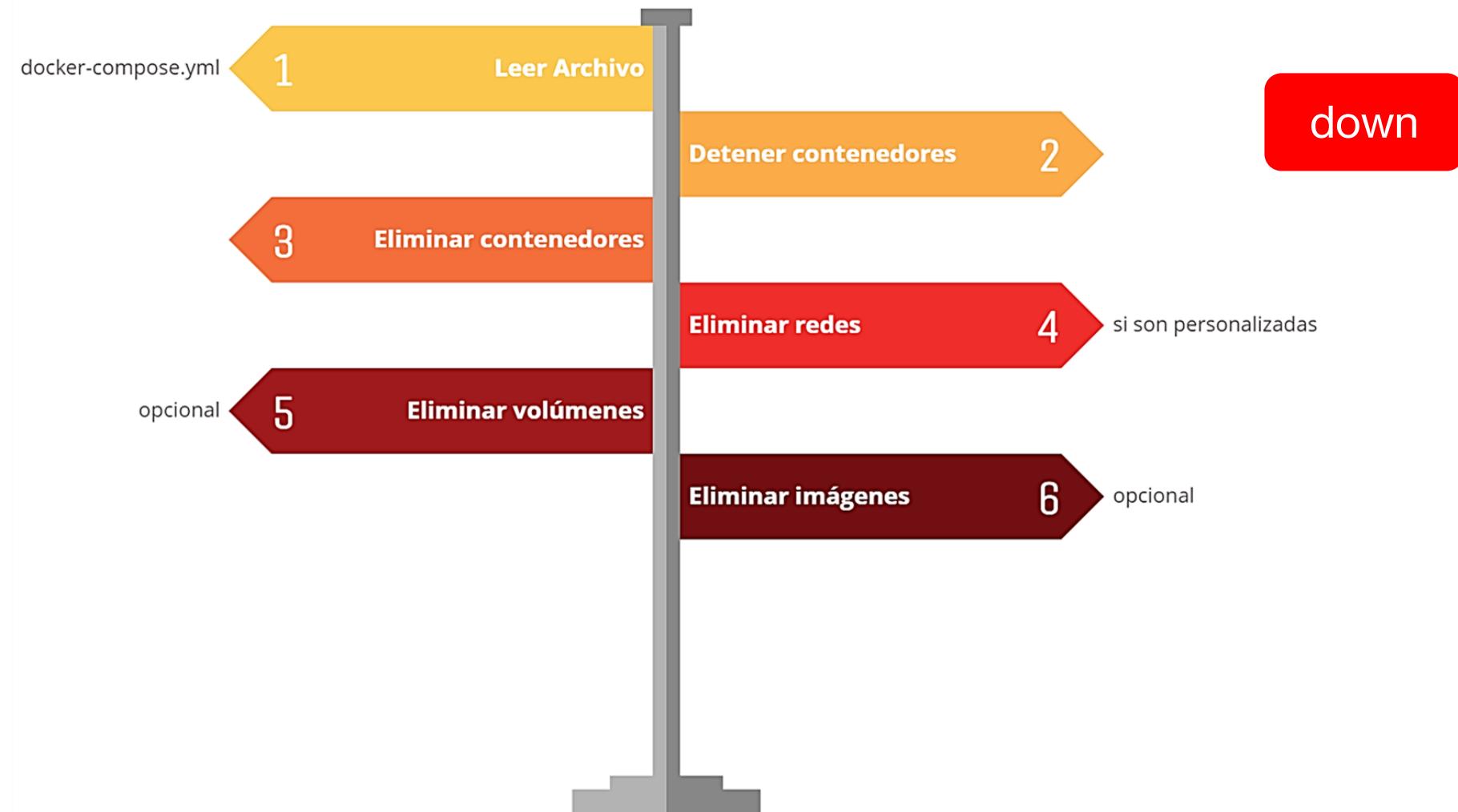
docker-compose down

- Este comando detiene todos los servicios y elimina los contenedores asociados.
- Puede incluir las opciones de:
 - `--volumes` para eliminar también los volúmenes definidos.

Ejecutando | docker-compose up



Ejecutando | docker-compose down



Ejecutando todo

```
C:\00_Material_Docker_Kubernetes\Intermedio\03_ws-dc>
C:\00_Material_Docker_Kubernetes\Intermedio\03_ws-dc>docker-compose ps
NAME      IMAGE      COMMAND     SERVICE    CREATED      STATUS      PORTS
C:\00_Material_Docker_Kubernetes\Intermedio\03_ws-dc>docker-compose up -d --dry-run
[+] Running 4/4
  ⚡ DRY-RUN MODE - Network 03_ws-dc_ms-network  Created                               0.0s
  ⚡ DRY-RUN MODE - Container ms-favoritos      Started                             0.0s
  ⚡ DRY-RUN MODE - Container ms-peliculas       Started                             0.0s
  ⚡ DRY-RUN MODE - Container ms-oradb          Started                             0.0s

C:\00_Material_Docker_Kubernetes\Intermedio\03_ws-dc>docker ps --format "table {{.ID}}\t{{.Names}}\t{{.Status}}\t{{.Ports}}\t{{.Mounts}}"
CONTAINER ID      NAMES      STATUS      PORTS      MOUNTS
C:\00_Material_Docker_Kubernetes\Intermedio\03_ws-dc>docker-compose up -d
[+] Running 4/4
  ⚡ Network 03_ws-dc_ms-network  Created                               0.1s
  ⚡ Container ms-peliculas      Started                             0.9s
  ⚡ Container ms-favoritos      Started                             1.4s
  ⚡ Container ms-oradb          Started                             1.4s

C:\00_Material_Docker_Kubernetes\Intermedio\03_ws-dc>docker ps --format "table {{.ID}}\t{{.Names}}\t{{.Status}}\t{{.Ports}}\t{{.Mounts}}"
CONTAINER ID      NAMES      STATUS      PORTS      MOUNTS
6683edcd7575    ms-favoritos Up 2 seconds           0.0.0.0:9084->9084/tcp, 9094/tcp
980b5ec426ef    ms-peliculas Up 2 seconds           0.0.0.0:9081->9081/tcp, 9091/tcp
f443742a7eeb    ms-oradb     Up 2 seconds (health: starting) 0.0.0.0:1521->1521/tcp

C:\00_Material_Docker_Kubernetes\Intermedio\03_ws-dc>docker-compose up -d
[+] Running 3/3
  ⚡ Container ms-favoritos   Running                            0.0s
  ⚡ Container ms-oradb       Running                            0.0s
  ⚡ Container ms-peliculas  Running                            0.0s

C:\00_Material_Docker_Kubernetes\Intermedio\03_ws-dc>
```

2.7. Build imagen en Docker Compose

Cuando se trabaja con Docker Compose, hay dos enfoques principales para manejar las imágenes de los servicios:

1. Usar imágenes preconstruidas.
2. Construir imágenes dentro del archivo docker-compose.yml

Build imagen en Docker Compose

Usar imágenes preconstruidas

Beneficios:

- Acelera el despliegue al evitar pasos de construcción en tiempo de ejecución.
- Ideal para entornos de producción donde las imágenes ya han sido probadas.
- Simplifica la configuración en el archivo de Docker Compose.

```
ms-peliculas:  
  image: usuario/ms-peliculas:1.0.0
```

Build imagen en Docker Compose

Construir imágenes dentro del archivo Compose

Beneficios:

- Útil en entornos de desarrollo donde el código puede cambiar con frecuencia.
- Permite asegurar que las imágenes se construyan con la configuración más reciente.

```
ms-peliculas:  
  build:  
    context: ./ms-peliculas
```

Build imagen en Docker Compose

Aspecto	Imágenes predefinidas	Construcción en Docker Compose
Tiempo de despliegue	Más rápido	Más lento
Flexibilidad	Menor, requiere subir nueva imagen.	Mátao, cambios se reflejan al construir.
Uso recomendado	Producción	Desarrollo
Dependencias externas	No depende de tener herramientas de construcción locales (p. ej., Maven, Node.js).	Requiere que el entorno de construcción tenga las herramientas necesarias configuradas.

Build imagen en Docker Compose

Aspecto	Imágenes predefinidas	Construcción en Docker Compose
Control sobre el proceso	Limitado, la construcción ya se realizó y no se puede ajustar.	Total, se puede modificar el Dockerfile y reconstruir en cualquier momento.
Uso de registries remotos	Requiere un Docker Registry para almacenar las imágenes y facilitar su despliegue en otros entornos.	No es obligatorio, ya que las imágenes se construyen localmente.

Build imagen en Docker Compose

- Ejemplo básico de un archivo `docker-compose.yml` para ejecutar una aplicación Java en un solo contenedor.
- Este ejemplo incluye la opción de construir la imagen directamente desde el archivo `docker-compse.yml` utilizando la configuración **build**.

Build imagen | Java

- Estructura del proyecto

```
my-java-app/  
  └── src/  
    └── main/  
      └── java/  
        └── com/  
          └── example/  
            └── App.java  
  └── Dockerfile  
  └── docker-compose.yml  
  └── pom.xml
```

- Estructura del proyecto

```
package com.netec;  
  
public class App {  
    public static void main(String[] args) {  
        System.out.println("Docker Compose with Build!");  
    }  
}
```

Build imagen | pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
  http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.netec</groupId>
  <artifactId>demo-app</artifactId>
  <version>1.0.0</version>
  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>3.8.1</version>
        <configuration>
          <source>21</source>
          <target>21</target>
        </configuration>
      </plugin>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-jar-plugin</artifactId>
        <version>3.2.2</version>
        <configuration>
          <archive>
            <manifest>
              <mainClass>com.netec.App</mainClass>
            </manifest>
          </archive>
        </configuration>
      </plugin>
    </plugins>
  </build>
</project>
```



Build imagen | Dockerfile

```
# Utilizar una imagen base ligera con Java 21
FROM openjdk:21-slim

# Crear un directorio de trabajo
WORKDIR /app

# Copiar el jar generado
COPY target/demo-app-1.0.0.jar app.jar

# Exponer el puerto (opcional si no hay servidor)
EXPOSE 8080

# Comando para ejecutar la aplicación
ENTRYPOINT ["java", "-jar", "app.jar"]
```

Build imagen | docker-compose.yml

```
version: '3.8'

services:
  java-app:
    build:
      context: .
      dockerfile: Dockerfile
    container_name: demo-container
    ports:
      - "8888:8080"
    command: ["java", "-jar", "app.jar"]
```

"punto"

Build imagen | Contexto

- **Context** define el directorio base desde donde Docker Compose busca los archivos necesarios para construir la imagen.
- El contexto se refiere al conjunto de archivos que Docker envía al daemon de Docker durante la construcción.
- Docker empaqueta los archivos del contexto y los pone a disposición del **Dockerfile**.
- Solo los archivos dentro de este contexto estarán disponibles para el proceso de construcción, lo cual es importante para optimizar la eficiencia y evitar incluir archivos innecesarios.

Build imagen | Ventajas

Personalización

- Puedes definir diferentes contextos y Dockerfiles para diferentes servicios en el mismo archivo docker-compose.yml.

Optimización

- Permite controlar qué archivos se incluyen en el contexto para reducir el tamaño de la imagen.

Flexibilidad

- Útil en proyectos con múltiples servicios que requieren configuraciones específicas.

Build imagen | Ejecución

1. mvn clean package

2. docker-compose up --build

3. docker logs <nombre contenedor>

4. docker ps

5. docker images

6. docker-compose down

Build imagen en Docker Compose

```
[INFO] -----[ jar ]-----  
[INFO]  
[INFO] --- clean:3.2.0:clean (default-clean) @ demo-app ---  
[INFO] Deleting C:\00_Material_Docker_Kubernetes\Intermedio\04_ws_dc\my-java-app\target  
[INFO]  
[INFO] --- resources:3.3.0:resources (default-resources) @ demo-app ---  
[WARNING] Using platform encoding (UTF-8 actually) to copy filtered resources, i.e. build is platform dependent!  
[INFO] skip non existing resourceDirectory C:\00_Material_Docker_Kubernetes\Intermedio\04_ws_dc\my-java-app\src\main\resources  
[INFO]  
[INFO] --- compiler:3.8.1:compile (default-compile) @ demo-app ---  
[INFO] Changes detected - recompiling the module!  
[WARNING] File encoding has not been set, using platform encoding UTF-8, i.e. build is platform dependent!  
[INFO] Compiling 1 source file to C:\00_Material_Docker_Kubernetes\Intermedio\04_ws_dc\my-java-app\target\classes  
[INFO]  
[INFO] --- resources:3.3.0:testResources (default-testResources) @ demo-app ---  
[WARNING] Using platform encoding (UTF-8 actually) to copy filtered resources, i.e. build is platform dependent!  
[INFO] skip non existing resourceDirectory C:\00_Material_Docker_Kubernetes\Intermedio\04_ws_dc\my-java-app\src\test\resources  
[INFO]  
[INFO] --- compiler:3.8.1:testCompile (default-testCompile) @ demo-app ---  
[INFO] No sources to compile  
[INFO]  
[INFO] --- surefire:3.0.0-M8:test (default-test) @ demo-app ---  
[INFO] No tests to run.  
[INFO]  
[INFO] --- jar:3.2.2:jar (default-jar) @ demo-app ---  
[INFO] Building jar: C:\00_Material_Docker_Kubernetes\Intermedio\04_ws_dc\my-java-app\target\demo-app-1.0.0.jar  
[INFO] -----  
[INFO] BUILD SUCCESS  
[INFO] -----  
[INFO] Total time: 3.740 s  
[INFO] Finished at: 2024-11-22T18:51:26-06:00  
[INFO] -----  
  
C:\00_Material_Docker_Kubernetes\Intermedio\04_ws_dc\my-java-app>
```

Build imagen en Docker Compose

```
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time:  3.740 s
[INFO] Finished at: 2024-11-22T18:51:26-06:00
[INFO] -----



C:\00_Material_Docker_Kubernetes\Intermedio\04_ws_dc\my-java-app>docker-compose up --build -d
time="2024-11-22T18:52:04-06:00" level=warning msg="C:\00_Material_Docker_Kubernetes\Intermedio\04_ws_dc\my-java-app\docker-compose.yml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential confusion"
[+] Building 1.6s (9/9) FINISHED                                            docker:desktop-linux
=> [java-app internal] load build definition from Dockerfile           0.1s
=> => transferring dockerfile: 374B                                     0.0s
=> [java-app internal] load metadata for docker.io/library/openjdk:21-slim   0.6s
=> [java-app internal] load .dockerignore                                0.0s
=> => transferring context: 2B                                         0.0s
=> [java-app 1/3] FROM docker.io/library/openjdk:21-slim@sha256:7072053847a8a05d7f3a14ebc778a90b38c50ce7e8f199382128a53385160688 0.1s
=> => resolve docker.io/library/openjdk:21-slim@sha256:7072053847a8a05d7f3a14ebc778a90b38c50ce7e8f199382128a53385160688 0.1s
=> [java-app internal] load build context                                0.1s
=> => transferring context: 2.19kB                                      0.0s
=> CACHED [java-app 2/3] WORKDIR /app                                    0.0s
=> [java-app 3/3] COPY target/demo-app-1.0.0.jar app.jar                0.1s
=> [java-app] exporting to image                                       0.4s
=> => exporting layers                                                 0.1s
=> => exporting manifest sha256:9e65a8a1e6cb1eb632e8dbcd880fe29cc389fca49b9308a88e65d5089baeab42 0.0s
=> => exporting config sha256:5ddb23831c81335e6810be8bdc90f7ae7d7c1c39b31bd2e1e39e3c67580499fd 0.0s
=> => exporting attestation manifest sha256:376692b97ef53a26c2b2ec7d9361c043a4a3d41f12bd856d537f5bed0718e5cd 0.1s
=> => exporting manifest list sha256:f74f722c5e11e70a2d9c2bd75e74f59a3e4ff728ae678989cc39d90a7fc379a9 0.0s
=> => naming to docker.io/library/my-java-app-jar:latest                 0.0s
=> => unpacking to docker.io/library/my-java-app-jar:latest               0.1s
=> [java-app] resolving provenance for metadata file                     0.0s
[+] Running 1/1
  Container demo-container  Started                                         1.6s



C:\00_Material_Docker_Kubernetes\Intermedio\04_ws_dc\my-java-app>
```

Build imagen en Docker Compose

```
=> => exporting layers 0.1s
=> => exporting manifest sha256:9e65a8a1e6cb1eb632e8dbcd880fe29cc389fca49b9308a88e65d5089baeab42 0.0s
=> => exporting config sha256:5ddb23831c81335e6810be8bdc90f7ae7d7c1c39b31bd2e1e39e3c67580499fd 0.0s
=> => exporting attestation manifest sha256:376692b97ef53a26c2b2ec7d9361c043a4a3d41f12bd856d537f5bed0718e5cd 0.1s
=> => exporting manifest list sha256:f74f722c5e11e70a2d9c2bd75e74f59a3e4ff728ae678989cc39d90a7fc379a9 0.0s
=> => naming to docker.io/library/my-java-app:java-app:latest 0.0s
=> => unpacking to docker.io/library/my-java-app:java-app:latest 0.1s
=> [java-app] resolving provenance for metadata file 0.0s
[+] Running 1/1
  Container demo-container  Started 1.6s

C:\00_Material_Docker_Kubernetes\Intermedio\04_ws_dc\my-java-app>docker logs demo-container
Docker Compose with Build!

C:\00_Material_Docker_Kubernetes\Intermedio\04_ws_dc\my-java-app>docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS               NAMES
ca2683c06070        my-java-app:java-app   "java -jar app.jar j..."   54 seconds ago    Exited (0)   51 seconds ago   demo-container

C:\00_Material_Docker_Kubernetes\Intermedio\04_ws_dc\my-java-app>docker images
REPOSITORY          TAG      IMAGE ID            CREATED             SIZE
my-java-app:java-app          f74f722c5e11  About a minute ago  689MB
ms-deseos           1.1      42e1fbe28348    7 hours ago       842MB
ms-productos        1.2      21d09bbd157c    7 hours ago       828MB
ms-deseos           1.0      6fb26f00cd36    2 days ago        798MB
ms-productos        1.0      7af428ad6478    3 days ago        784MB
ms-favoritos         1.0.0    a2b7174d7546    6 days ago        805MB
ms-peliculas         1.0.0    b0aaa0b4e16f    6 days ago        787MB
ghcr.io/oracle/oraclelinux8-instantclient  19      b4e4e6f22460    2 weeks ago       717MB
mongo                latest   c165af1a407e    4 weeks ago       1.14GB
nginx                latest   bc5eac5eafc5    7 weeks ago       279MB
mysql                5.7     4bc6bc963e6d    11 months ago     689MB
container-registry.oracle.com/database/express  21.3.0-xe  dcf137aab02d   15 months ago     15.2GB

C:\00_Material_Docker_Kubernetes\Intermedio\04_ws_dc\my-java-app>
```

Build imagen en Docker Compose

```
C:\00_Material_Docker_Kubernetes\Intermedio\04_ws_dc\my-java-app>
C:\00_Material_Docker_Kubernetes\Intermedio\04_ws_dc\my-java-app>docker ps -a
CONTAINER ID   IMAGE          COMMAND       CREATED      STATUS        PORTS     NAMES
ca2683c06070   my-java-app-jar   "java -jar app.jar j..."   About a minute ago   Exited (0) About a minute ago
                                                               demo-container

C:\00_Material_Docker_Kubernetes\Intermedio\04_ws_dc\my-java-app>
C:\00_Material_Docker_Kubernetes\Intermedio\04_ws_dc\my-java-app>docker-compose down
time="2024-11-22T18:54:00-06:00" level=warning msg="C:\\00_Material_Docker_Kubernetes\\Intermedio\\04_ws_dc\\my-java-app\\docker-compose.yml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential confusion"
[+] Running 2/2
  ⚡ Container demo-container    Removed                               2.1s
  ⚡ Network my-java-app_default Removed                           0.3s

C:\00_Material_Docker_Kubernetes\Intermedio\04_ws_dc\my-java-app>docker ps -a
CONTAINER ID   IMAGE          COMMAND       CREATED      STATUS        PORTS     NAMES

C:\00_Material_Docker_Kubernetes\Intermedio\04_ws_dc\my-java-app>
```

Resumen

- La unidad proporcionó una visión integral del uso de Docker Compose, desde los conceptos básicos hasta la ejecución de aplicaciones completas con servicios interdependientes. Los participantes aprendieron a gestionar múltiples contenedores, configurar servicios complejos y optimizar el flujo de desarrollo y despliegue de aplicaciones.



Práctica 2.1. Explorando Docker Compose

Objetivo:

Al finalizar la actividad serás capaz de comprender la estructura básica de Docker Compose.

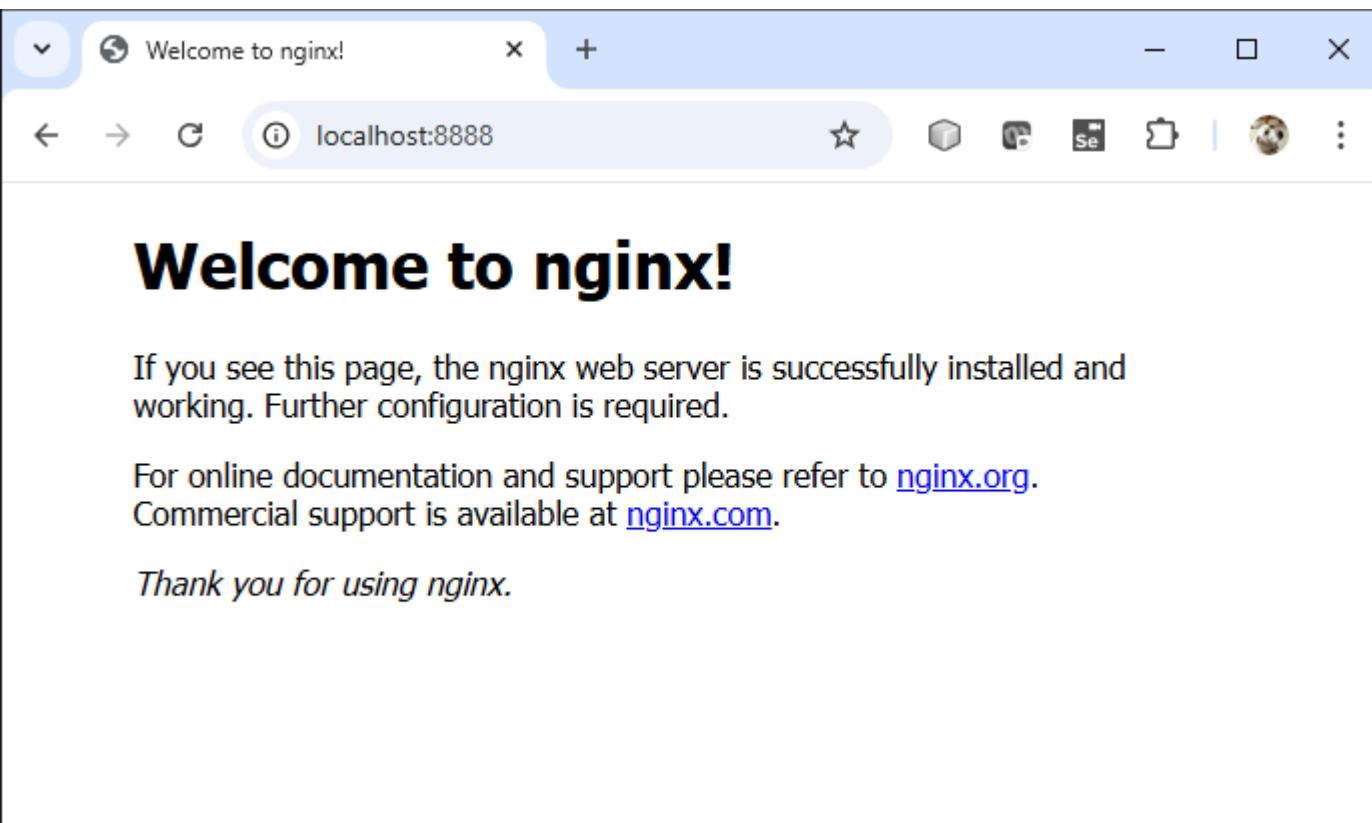
Planteamiento e instrucciones:

Realiza un seguimiento de los pasos indicados en la *Guía de Laboratorios* para llevar a cabo la tarea correspondiente en el siguiente enlace: [Plataforma de Laboratorios | DOCK_KUB_INT](#)



Tiempo para esta actividad:
25 minutos.

Resultado esperado



Resultado esperado

```
c:\ Símbolo del sistema - docker exec -it 02_ws-dc-db-1 bash
mysql>
mysql> show databases;
+-----+
| Database      |
+-----+
| information_schema |
| demo          |
| mysql          |
| performance_schema |
| sys           |
+-----+
5 rows in set (0.00 sec)

mysql> use demo
Database changed
mysql> show tables;
Empty set (0.00 sec)

mysql>
```



Resultado esperado

Los participantes podrán responder preguntas como:

1. ¿Qué ventaja tiene usar Docker Compose frente a gestionar contenedores manualmente?
2. ¿Cómo está estructurado un archivo **docker-compose.yml**?
3. ¿Cuáles son los comandos básicos de Docker Compose y para qué se usan?

Práctica 2.2. Creación de archivo Docker Compose intermedio

Objetivo:

Al finalizar esta práctica, serás capaz de crear un archivo docker-compose.yml de nivel intermedio que defina dos contenedores simples.

Planteamiento e instrucciones:

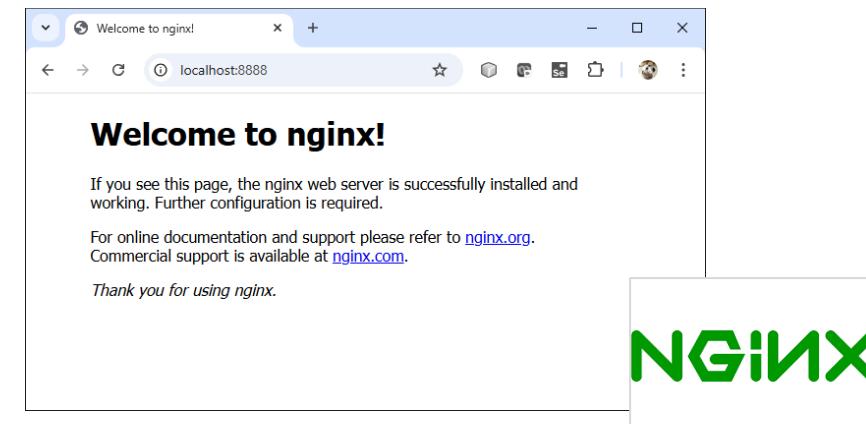
Realiza un seguimiento de los pasos indicados en la *Guía de Laboratorios* para llevar a cabo la tarea correspondiente en el siguiente enlace: [Plataforma de Laboratorios | DOCK_KUB_INT](#)



Tiempo para esta actividad:
25 minutos.

Resultado esperado

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
abc123456789	nginx:latest	/docker-entrypoint..."	1 minute ago	Up 30 seconds	0.0.0.0:8080->80/tcp	practica-nginx
def987654321	mongo:latest	"docker-entrypoint.s..."	1 minute ago	Up 30 seconds	0.0.0.0:27017->27017/tcp	practica-mongodb



Práctica 2.3. Configuración de un contenedor Oracle DB con Docker Compose

Objetivo:

Al finalizar esta práctica, serás capaz de definir un contenedor de Oracle Database en un archivo docker-compose.yml, configurando variables de entorno, volúmenes y redes personalizadas.

Planteamiento e instrucciones:

Realiza un seguimiento de los pasos indicados en la *Guía de Laboratorios* para llevar a cabo la tarea correspondiente en el siguiente enlace: [Plataforma de Laboratorios | DOCK_KUB_INT](#)

https://github.com/NetecGk/DOCK_KUB_INT



Tiempo para esta actividad:
25 minutos.

Resultado esperado

```
C:\00_Material_Docker_Kubernetes\Intermedio\practica_2_3_oracle>
C:\00_Material_Docker_Kubernetes\Intermedio\practica_2_3_oracle>
C:\00_Material_Docker_Kubernetes\Intermedio\practica_2_3_oracle>docker ps
CONTAINER ID   IMAGE      COMMAND      CREATED      STATUS      PORTS
C:\00_Material_Docker_Kubernetes\Intermedio\practica_2_3_oracle>docker volume ls
DRIVER    VOLUME NAME
local     02_ws-dc_db-data
local     146d0de05313f9f66e65850a36c0f4c530c13217e6014615b774e15abcf1e484
local     dki-volume
local     ms-volume
local     oracle_data

C:\00_Material_Docker_Kubernetes\Intermedio\practica_2_3_oracle>docker network ls
NETWORK ID   NAME      DRIVER      SCOPE
c386a1d2a105  bridge    bridge    local
59b0a9e4d9ea  dki-network    bridge    local
90564692b07c  host      host      local
4ebba8004327  ms-network    bridge    local
c930d999554c  none      null      local

C:\00_Material_Docker_Kubernetes\Intermedio\practica_2_3_oracle>docker-compose up -d
time="2024-11-21T13:39:46-06:00" level=warning msg="C:\\00_Material_Docker_Kubernetes\\\\Intermedio\\\\practica_2_3_oracle\\\\docker-compose.yml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential confusion"
[+] Running 3/3
  ▓ Network practica_2_3_oracle_dki-network      Created                               0.1s
  ▓ Volume "practica_2_3_oracle_dki-oracle-data"  Created                               0.0s
  ▓ Container dki-oracle-container               Started                             109.0s

C:\00_Material_Docker_Kubernetes\Intermedio\practica_2_3_oracle>docker ps
CONTAINER ID   IMAGE      COMMAND      CREATED      STATUS      PORTS
                  NAMES
ca7879e26199  container-registry.oracle.com/database/express:21.3.0-xe  "/bin/bash -c $ORACL..."  2 minutes ago  Up 13 seconds (health: starting)  0.0.0:1521->1521
/tcp, 0.0.0.0:5500->5500/tcp  dki-oracle-container

C:\00_Material_Docker_Kubernetes\Intermedio\practica_2_3_oracle>docker volume ls | grep dki
local     dki-volume
local     practica_2_3_oracle_dki-oracle-data

C:\00_Material_Docker_Kubernetes\Intermedio\practica_2_3_oracle>docker network ls | grep dki
59b0a9e4d9ea  dki-network    bridge    local
e8c2708f0c82  practica_2_3_oracle_dki-network    bridge    local

C:\00_Material_Docker_Kubernetes\Intermedio\practica_2_3_oracle>
```

Práctica 2.4. Integración de microservicios en Docker Compose

Objetivo:

Al finalizar esta práctica, serás capaz de añadir los microservicios del caso de estudio al archivo docker-compose.yml, conectándolos a la base de datos Oracle y definiendo redes para la comunicación entre ellos.

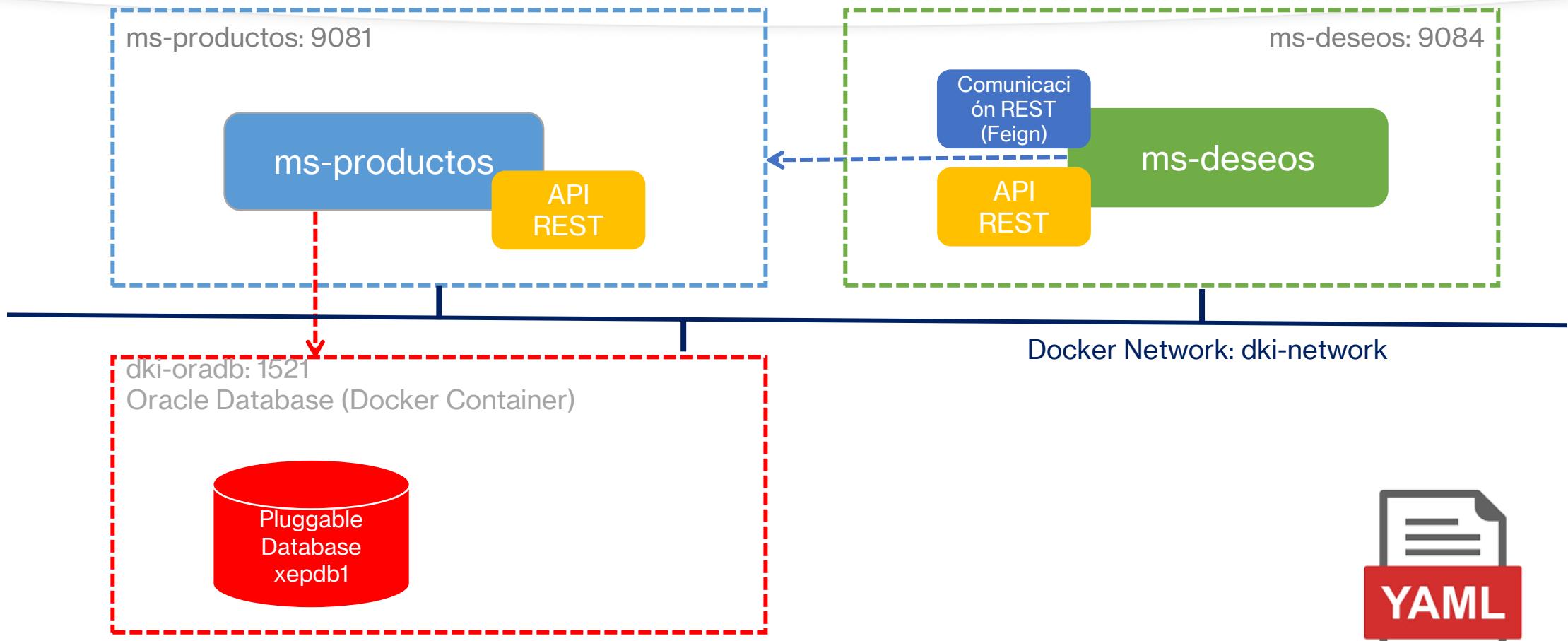
Planteamiento e instrucciones:

Realiza un seguimiento de los pasos indicados en la *Guía de Laboratorios* para llevar a cabo la tarea correspondiente en el siguiente enlace: [Plataforma de Laboratorios | DOCK_KUB_INT](#)



Tiempo para esta actividad:
50 minutos.

Resultado esperado



Resultado esperado

Método	Endpoint	Descripción
GET	/productos	Listar todos los productos.
GET	/productos/{id}	Obtener un producto por ID.
POST	/productos	Crear un nuevo producto.
PUT	/productos/{id}	Actualizar un producto existente.
DELETE	/productos/{id}	Eliminar un producto.

```
{  
    id: Long,  
    nombre: String,  
    descripcion: String,  
    precio: Double,  
    stock: Integer  
}
```

```
id (Primary Key, autoincremental)  
nombre (String, único)  
descripcion (String)  
precio (Decimal)  
stock (Entero)
```

Método	Endpoint	Descripción
GET	/deseos	Listar todos los productos en la lista.
POST	/deseos/{idProducto}	Agregar un producto a la lista de deseos.
DELETE	/deseos/{idProducto}	Eliminar un producto de la lista

Referencias Bibliográficas

- Docker-Compose: Installation scenarios
<https://docs.docker.com/compose/install/>
- Docker Compose: depends_on
https://docs.docker.com/reference/compose-file/services/#depends_on
- Docker Compose – Manuals
<https://docs.docker.com/compose/>
- Docker Compose Quickstart
<https://docs.docker.com/compose/gettingstarted/>

Objetivos:

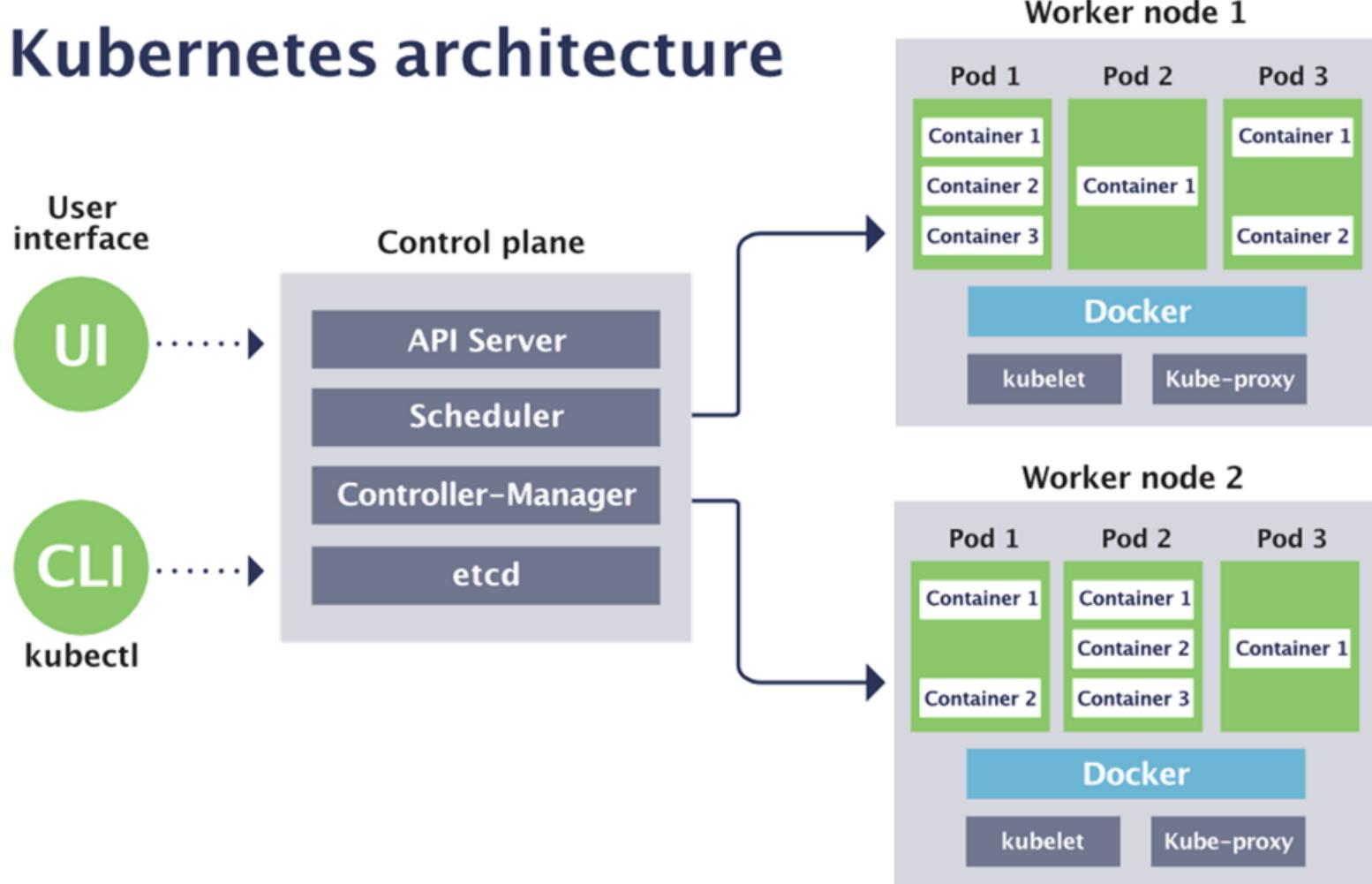
- Comprender la configuración Centralizada en Kubernetes.
- Crear y gestionar Configmaps en Kubernetes.
- Aplicar ConfigMaps en Deployments.
- Crear y Gestionar Secrets en Kubernetes.
- Aplicar Secrets en Deployments.
- Diferenciar entre ConfigMaps y Secrets.

Unidad 3

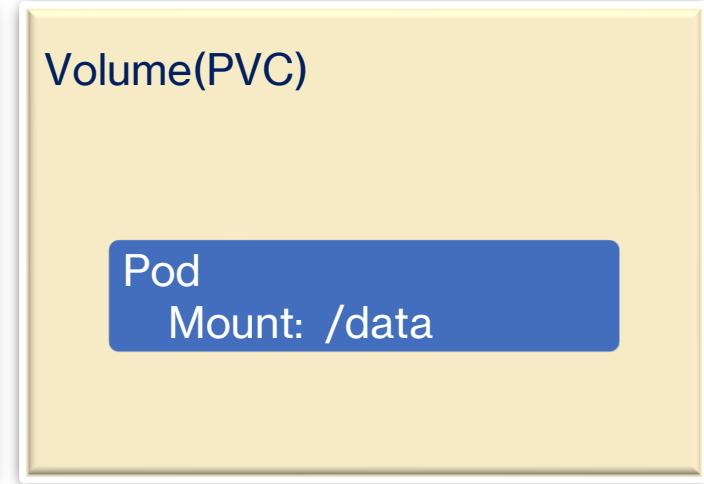
Kubernetes: ConfigMap y Secret

Kubernetes | Componentes clave

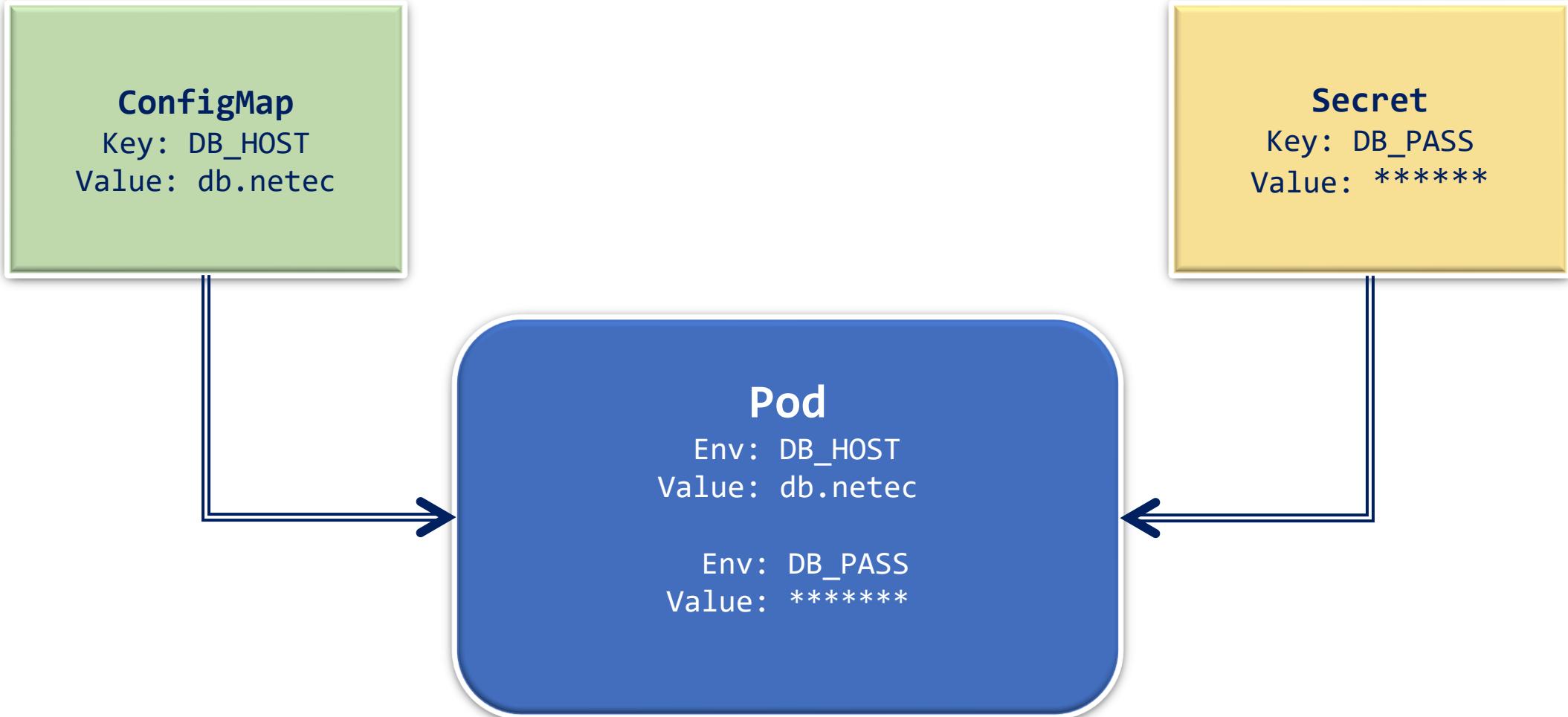
Kubernetes architecture



Kubernetes | Objetos principales



Kubernetes | Objetos principales



Kubernetes | kubectl

- Es la herramienta de línea de comandos para interactuar con Kubernetes.

```
kubectl get Pods -o wide  
kubectl get Pods -n  
<namespace>  
kubectl describe Pod <Pod-name>
```

```
kubectl apply -f file.yaml
```

```
kubectl delete -f file.yaml
```

```
kubectl get all
```

```
kubectl get namespaces
```

```
kubectl get events
```

```
kubectl get Pods -n <namespace>
```

```
kubectl logs <Pod-name>
```

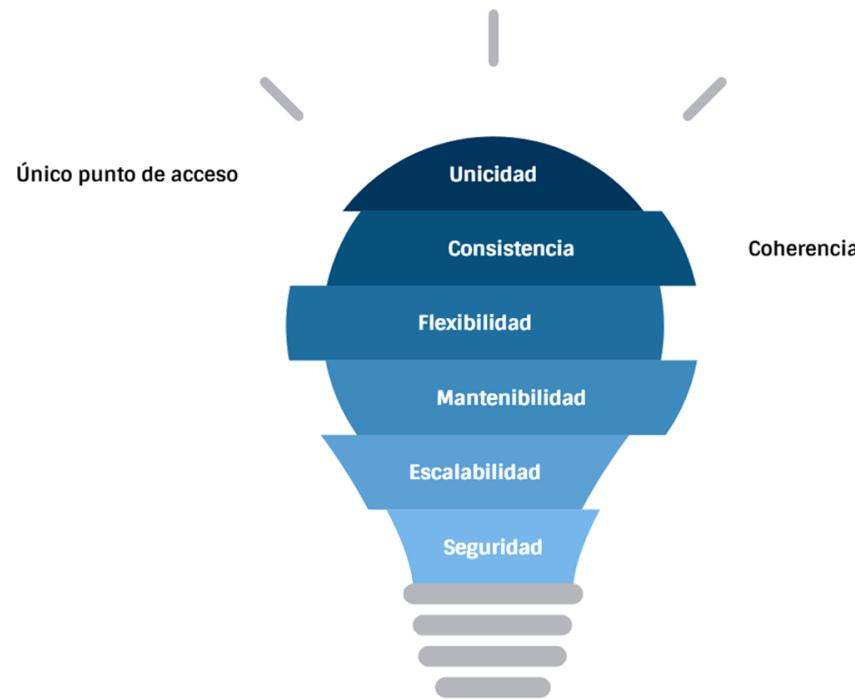
```
kubectl run my-Pod --image=nginx --port=80
```

```
kubectl exec -it <Pod-name> -- /bin/bash
```

```
kubectl scale deployment my-deployment --replicas=5
```

```
kubectl config set-context --current --namespace=<namespace>
```

3.1. Configuración centralizada



Configuración centralizada

Único punto de acceso

- Toda la configuración está almacenada y gestionada desde un único servicio o repositorio.

Consistencia

- Permite mantener configuraciones coherentes en todas las aplicaciones o servicios.

Flexibilidad

- Cambiar la configuración en un solo lugar se refleja en todas las aplicaciones que dependen de ella.

Seguridad

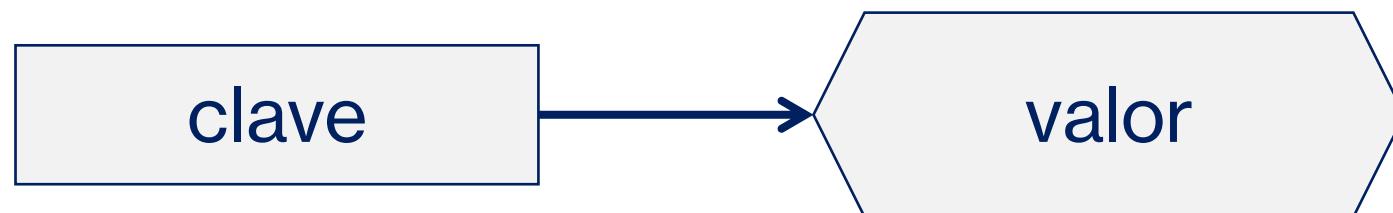
- Mejora la protección de credenciales y datos sensibles, al evitar su dispersión en múltiples archivos.

Escalabilidad

- Facilita la administración en sistemas distribuidos con muchos servicios o aplicaciones.

3.2. Agregando diccionario ConfigMaps

- Los ConfigMaps en Kubernetes son objetos que permiten almacenar datos de configuración en pares clave-valor.
- Utilizar archivos YAML para definir ConfigMaps proporciona una forma de gestionar la configuración de aplicaciones que se ejecutan en contenedores.



Agregando diccionario ConfigMaps

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: nombre-del-configmap
  namespace: nombre-del-namespace # Opcional
data:
  clave1: valor1
  clave2: valor2
  ...
```

ConfigMap no proporciona encriptación.

- **apiVersion:** Indica la versión de la API de Kubernetes.
- **kind:** Especifica el tipo de objeto, en este caso, ConfigMap.
- **metadata:** Contiene información sobre el ConfigMap, como su nombre y el namespace donde reside.
- **data:** Aquí se definen los pares clave-valor que se almacenarán en el ConfigMap.

Agregando diccionario ConfigMaps

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: nginx-env-config
data:
  NGINX_MESSAGE: "Welcome to NGINX configured with a ConfigMap!"
```

```
kubectl apply -f nginx-env-config.yaml
```

Agregando diccionario ConfigMaps

```
madmin@master:~/Intermedio/ws2$  
madmin@master:~/Intermedio/ws2$  
madmin@master:~/Intermedio/ws2$  
madmin@master:~/Intermedio/ws2$ kubectl apply -f  
configmap.yaml deployment.yaml  
madmin@master:~/Intermedio/ws2$ kubectl apply -f configmap.yaml  
configmap/nginx-env-config created  
madmin@master:~/Intermedio/ws2$  
madmin@master:~/Intermedio/ws2$ kubectl get configmap | grep nginx-env-config  
nginx-env-config 1 24s  
madmin@master:~/Intermedio/ws2$  
madmin@master:~/Intermedio/ws2$  
madmin@master:~/Intermedio/ws2$ kubectl describe configmap nginx-env-config  
Name: nginx-env-config  
Namespace: default  
Labels: <none>  
Annotations: <none>  
  
Data  
====  
NGINX_WELCOME_MESSAGE:  
----  
Welcome to NGINX configured with a ConfigMap!  
  
BinaryData  
====  
  
Events: <none>  
madmin@master:~/Intermedio/ws2$ |
```

Agregando diccionario ConfigMaps

```
# Aplicar un ConfigMap desde un YAML  
kubectl apply -f my-configmap.yaml  
  
# Crear un ConfigMap directamente desde el archivo de entorno  
kubectl create configmap my-configmap --from-env-file=configmap.env  
  
# Verificar que existen en el clúster.  
kubectl get configmaps  
  
# Ver los detalles de un Secret  
kubectl describe config my-configmap  
  
# Ver el contenido del ConfigMap en un formato legible  
kubectl get configmap my-configmap -o yaml
```

3.3. Aplicando y utilizando las configuraciones en los Deployments

- ¿Qué es un Deployment en Kubernetes?

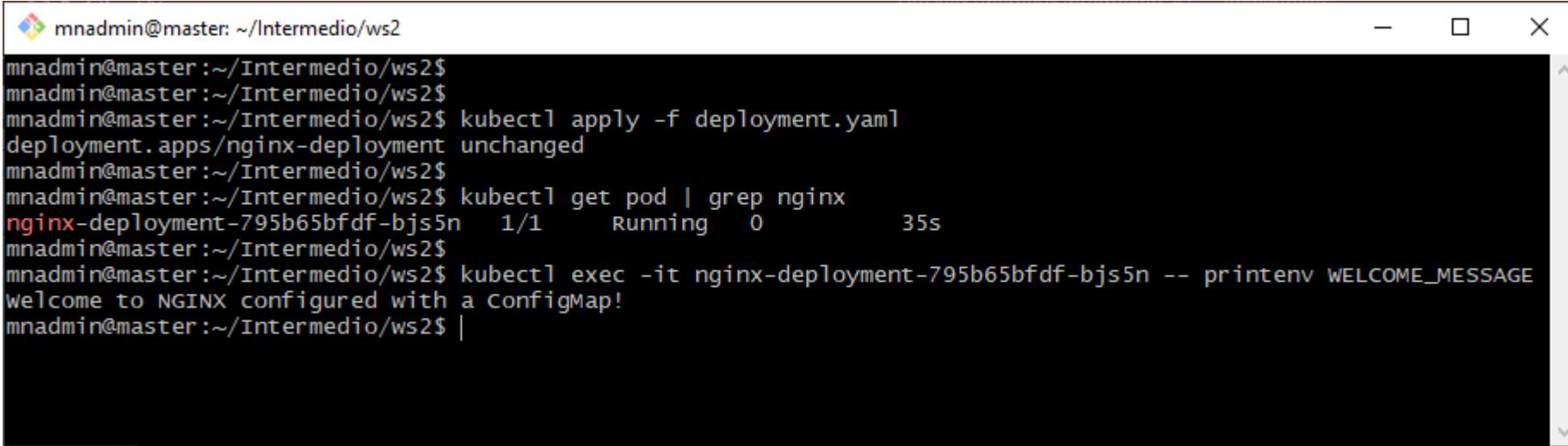
Un Deployment es una herramienta poderosa que simplifica la administración de aplicaciones en Kubernetes, combinando configuraciones declarativas, tolerancia a fallos y capacidades de actualización/escalado para aplicaciones de cualquier complejidad.

Aplicando y utilizando las configuraciones en los Deployments

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx-container
          image: nginx:1.23.4
          env:
            - name: WELCOME_MESSAGE
              valueFrom:
                configMapKeyRef:
                  name: nginx-env-config
                  key: NGINX_WELCOME_MESSAGE
```

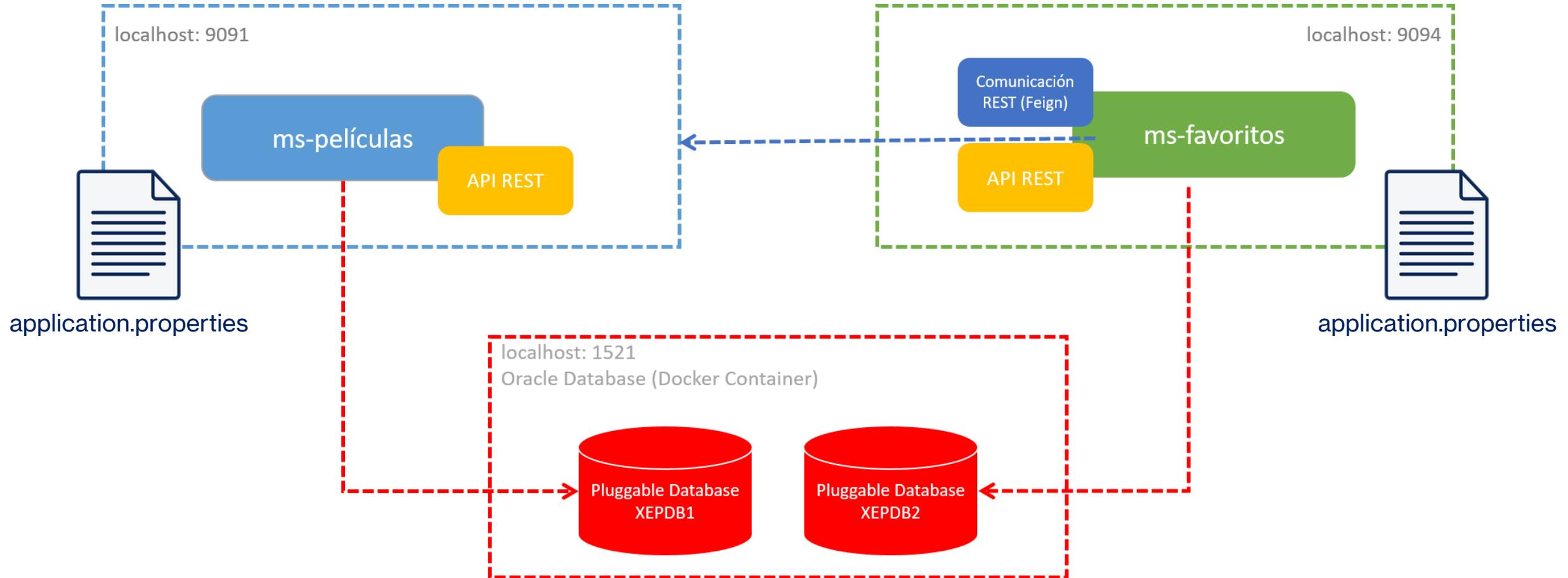
```
kubectl apply -f nginx-deployment.yaml
```

Aplicando y utilizando las configuraciones en los Deployments

A screenshot of a terminal window titled "mnadmin@master: ~/Intermedio/ws2\$". The window contains a series of commands and their outputs related to applying a deployment configuration and inspecting a pod's environment variables.

```
mnadmin@master:~/Intermedio/ws2$ mnadmin@master:~/Intermedio/ws2$ mnadmin@master:~/Intermedio/ws2$ kubectl apply -f deployment.yaml
deployment.apps/nginx-deployment unchanged
mnadmin@master:~/Intermedio/ws2$ mnadmin@master:~/Intermedio/ws2$ kubectl get pod | grep nginx
nginx-deployment-795b65bfdf-bjs5n  1/1    Running   0          35s
mnadmin@master:~/Intermedio/ws2$ mnadmin@master:~/Intermedio/ws2$ kubectl exec -it nginx-deployment-795b65bfdf-bjs5n -- printenv WELCOME_MESSAGE
Welcome to NGINX configured with a ConfigMap!
mnadmin@master:~/Intermedio/ws2$ |
```

Aplicando | Caso de estudio



Aplicando | Guía básica



Aplicando | Caso de estudio | Preparación

```
# Nombre del microservicio
spring.application.name=${SPRING_APPLICATION_NAME:ms-peliculas} ${VARIABLE:valor}

# Puerto del servidor
server.port=${SERVER_PORT:9091}

# Configuración de la base de datos
spring.datasource.url=${DB_URL:jdbc:oracle:thin:@oracle-db:1521/XEPDB1}
spring.datasource.username=${DB_USERNAME:dkuser}
spring.datasource.password=${DB_PASSWORD:dkpassword}
spring.datasource.driver-class-name=${DB_DRIVER_CLASS_NAME:oracle.jdbc.OracleDriver}

# Configuración de JPA e Hibernate
spring.jpa.hibernate.ddl-auto=${JPA_DDL_AUTO:update}
spring.jpa.show-sql=${JPA_SHOW_SQL:true}
spring.jpa.properties.hibernate.dialect=${JPA_DIALECT:org.hibernate.dialect.OracleDialect}
```

Application Properties de ms-peliculas

Aplicando | Caso de estudio | Preparación

```
# Nombre del microservicio
spring.application.name=${SPRING_APPLICATION_NAME:ms-favoritos}

# Puerto del servidor
server.port=${SERVER_PORT:9094}

# Configuración de la base de datos
spring.datasource.url=${DB_URL:jdbc:oracle:thin:@oracle-db:1521/XEPDB2}
spring.datasource.username=${DB_USERNAME:dkuser}
spring.datasource.password=${DB_PASSWORD:dkpassword}
spring.datasource.driver-class-name=${DB_DRIVER_CLASS_NAME:oracle.jdbc.OracleDriver}

# Configuración de JPA e Hibernate
spring.jpa.hibernate.ddl-auto=${JPA_DDL_AUTO:update}
spring.jpa.show-sql=${JPA_SHOW_SQL:true}
spring.jpa.properties.hibernate.dialect=${JPA_DIALECT:org.hibernate.dialect.OracleDialect}

# Comunicación con ms-peliculas
mspeliculas.url=${MSPELICULAS_URL:http://ms-peliculas:9091}
```

Application Properties de ms-favoritos

Aplicando | Caso de estudio | Preparación

```
# 1. Imagen base
FROM openjdk:21-jdk-slim

# 2. Establecer el directorio de trabajo
WORKDIR /app

# 3. Copiar el archivo JAR generado
COPY target/ms-peliculas-0.0.1-SNAPSHOT.jar app.jar

# 4. Exponer el puerto utilizado por el microservicio
EXPOSE 9091

# 5. Permitir configuración mediante variables de entorno
ENV SPRING_PROFILES_ACTIVE=docker
ENV SERVER_PORT=9091

# 6. Comando de inicio
ENTRYPOINT ["java", "-Dspring.profiles.active=${SPRING_PROFILES_ACTIVE}", "-jar", "app.jar"]
```

Aplicando | Caso de estudio | Preparación

```
# 1. Imagen base
FROM openjdk:21-jdk-slim

# 2. Establecer el directorio de trabajo
WORKDIR /app

# 3. Copiar el archivo JAR generado
COPY target/ms-favoritos-0.0.1-SNAPSHOT.jar app.jar

# 4. Exponer el puerto utilizado por el microservicio
EXPOSE 9094

# 5. Permitir configuración mediante variables de entorno
ENV SPRING_PROFILES_ACTIVE=docker
ENV SERVER_PORT=9094

# 6. Comando de inicio
ENTRYPOINT ["java", "-Dspring.profiles.active=${SPRING_PROFILES_ACTIVE}", "-jar", "app.jar"]
```

Aplicando | Caso de estudio | Preparación

Alternativas para manejar puertos dinámicos

1. Configurar el puerto en tiempo de ejecución:
 - Usa la variable de entorno **SERVER_PORT** en tu aplicación para definir dinámicamente el puerto en el que escucha la aplicación.
 - Por ejemplo, en el application.properties:
 - `server.port=${SERVER_PORT:8080}`
 - Al iniciar el contenedor, especifica la variable de entorno:
 - `docker run -e SERVER_PORT=9091 -p 9091:9091 tu-imagen`
2. No depender de **EXPOSE**:
 - Puedes omitir **EXPOSE** en el Dockerfile y configurar los puertos al ejecutar el contenedor con -p o al mapear servicios en Kubernetes:
 - `docker run -p 9091:9091 tu-imagen`

Aplicando | Caso de estudio | Preparación

3. Control dinámico en Kubernetes:

- Kubernetes usa los puertos definidos en el manifiesto del Deployment y Service, no los del Dockerfile.
- Esto permite usar puertos dinámicos configurados mediante variables de entorno en el contenedor sin depender de EXPOSE.
- EXPOSE no es imprescindible para el despliegue ni para el funcionamiento del contenedor, pero es útil como referencia o documentación.
- Podrías experimentar problemas, si la propiedad **server.port** no está configurada correctamente en la aplicación o no coincide con el puerto que estás intentando mapear o exponer.

Aplicando | Caso de estudio | Preparación

¿Por qué "jugamos" con la configuración?

- La sincronización entre:
 - El puerto que escucha la aplicación (`server.port` en Spring Boot).
 - El puerto configurado en el contenedor (`containerPort` en Kubernetes).
 - Y los puertos expuestos al exterior (`NodePort`, `LoadBalancer`, o mapeos de Docker CLI como `-p`).
- Es crucial para garantizar que las solicitudes lleguen correctamente a la aplicación.

Aplicando | Caso de estudio | Preparación

1. Crear el JAR

- `.\mvnw clean install`

2. Generar la imagen Docker

- `docker build -t <nombre>:<tag> -f Dockerfile .`
- `docker images | grep <nombre>`

3. Registrar la imagen

- `docker tag <nombre>:<version> <usuario Docker Hub>/<imagen>:<tag>`
- `docker login`
- `docker push <usuario Docker Hub>/<imagen>:<tag>`

¿Necesitarías preparar algo más?

Aplicando | Caso de estudio | ConfigMaps

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: ms-peliculas-configmap
  namespace: default
data:
  SPRING_PROFILES_ACTIVE: "docker"
  SERVER_PORT: "9091"
  JPA_DDL_AUTO: "update"
  JPA_SHOW_SQL: "true"
  JPA_DIALECT: "org.hibernate.dialect.OracleDialect"
```

ms-películas-configmap.yaml



Aplicando | Caso de estudio | ConfigMaps

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: ms-favoritos-configmap
  namespace: default
data:
  SPRING_PROFILES_ACTIVE: "docker"
  SERVER_PORT: "9091"
  JPA_DDL_AUTO: "update"
  JPA_SHOW_SQL: "true"
  JPA_DIALECT: "org.hibernate.dialect.OracleDialect"
  MSPELICULAS_URL: "http://ms-peliculas:9091"
```

ms-favoritos-configmap-yaml



Aplicando | Caso de estudio | Deployments

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: ms-peliculas
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: ms-peliculas
  template:
    metadata:
      labels:
        app: ms-peliculas
    spec:
      containers:
        - name: ms-peliculas
          image: blankiss/ms-peliculas:1.0.3
          ports:
            - containerPort: 9091
          env:
            - name: SPRING_PROFILES_ACTIVE
              valueFrom:
                configMapKeyRef:
                  name: ms-peliculas-config
                  key: SPRING_PROFILES_ACTIVE
            - name: SERVER_PORT
              valueFrom:
                configMapKeyRef:
                  name: ms-peliculas-config
                  key: SERVER_PORT
# Mas propiedades omitidas
```

ms-peliculas-deployment.yaml

Aplicando | Caso de estudio | Deployments

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: ms-favoritos
  namespace: default
spec:
  replicas: 1
  selector:
    matchLabels:
      app: ms-favoritos
  template:
    metadata:
      labels:
        app: ms-favoritos
    spec:
      containers:
        - name: ms-favoritos
          image: your-docker-repo/ms-
```

```
favoritos:1.0.3
  ports:
    - containerPort: 9094
  env:
    - name: SPRING_PROFILES_ACTIVE
      valueFrom:
        configMapKeyRef:
          name: ms-favoritos-config
          key: SPRING_PROFILES_ACTIVE
    - name: SERVER_PORT
      valueFrom:
        configMapKeyRef:
          name: ms-favoritos-config
          key: SERVER_PORT
```

ms-favoritos-deployment.yaml

Aplicando | Caso de estudio | Services

```
apiVersion: v1
kind: Service
metadata:
  name: ms-peliculas
  namespace: default
spec:
  selector:
    app: ms-peliculas
  ports:
    - protocol: TCP
      port: 9091
      targetPort: 9091
  type: ClusterIP
```

ms-peliculas-service.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: ms-favoritos
  namespace: default
spec:
  selector:
    app: ms-favoritos
  ports:
    - protocol: TCP
      port: 9094
      targetPort: 9094
  type: ClusterIP
```

ms-favoritos-service.yaml



Aplicando | Caso de estudio | Aplicar

```
# Crear ConfigMaps
kubectl apply -f ms-peliculas-configmap.yaml
kubectl apply -f ms-favoritos-configmap.yaml

# Crear Deployments
kubectl apply -f ms-peliculas-deployment.yaml
kubectl apply -f ms-favoritos-deployment.yaml

# Crear Servicios
kubectl apply -f ms-peliculas-service.yaml
kubectl apply -f ms-favoritos-service.yaml
```



Aplicando | Caso de estudio | Verificar

```
# ConfigMaps
kubectl get configmap
kubectl describe configmap <nombre-del-configmap>

# Pods
kubectl get Pods -o wide
kubectl describe Pod <nombre-del-Pod>
kubectl logs <nombre-del-Pod>
kubectl logs <nombre-del-Pod> -c <nombre-del-contenedor>

# Servicios
kubectl get services
kubectl describe service <nombre-del-servicio>

# Deployments
kubectl get deployments
kubectl describe deployment <nombre-del-deployment>
kubectl get Pods --selector=app=<nombre-del-deployment>
```



Aplicando | Caso de estudio | Verificar

```
mnadmin@master:~/Intermedio/ws$  
mnadmin@master:~/Intermedio/ws$ kubectl get services  
NAME      TYPE      CLUSTER-IP   EXTERNAL-IP   PORT(S)      AGE  
kubernetes  clusterIP  10.96.0.1    <none>        443/TCP     35d  
ms-favoritos  clusterIP  10.109.104.76  <none>        9094/TCP    28h  
ms-peliculas  clusterIP  10.106.29.200  <none>        9091/TCP    28h  
oracle-db    NodePort    10.109.200.103  <none>        1521:30011/TCP 10m  
mnadmin@master:~/Intermedio/ws$  
mnadmin@master:~/Intermedio/ws$ kubectl get pod  
NAME                  READY   STATUS    RESTARTS   AGE  
ms-favoritos-78bcdcd75dd-cs5rf  1/1     Running   0          9m17s  
ms-peliculas-d859d7c84-9rx56   1/1     Running   0          9m16s  
oracle-db-cbf654876-z5kx9     1/1     Running   0          112m  
mnadmin@master:~/Intermedio/ws$  
mnadmin@master:~/Intermedio/ws$ curl http://10.106.29.200:9091/api/peliculas | jq .  
% Total % Received % Xferd Average Speed Time Time Current  
          Dload Upload Total Spent Left Speed  
100 287 0 287 0 0 15105 0 --:--:-- --:--:-- --:--:-- 15105  
[  
  {  
    "id": 1,  
    "titulo": "El Señor de los Anillos: La Comunidad del Anillo",  
    "director": "Peter Jackson",  
    "genero": "Fantasía",  
    "duracion": 178,  
    "fechaEstreno": "2001-12-19"  
  },  
  {  
    "id": 2,  
    "titulo": "Avatar",  
    "director": "James Cameron",  
    "genero": "Ciencia Ficción",  
    "duracion": 195,  
    "fechaEstreno": "2009-12-18"  
  }]  
mnadmin@master:~/Intermedio/ws$
```



Aplicando | Caso de estudio | Verificar

```
mnadmin@master:~/Intermedio/ws$ kubectl get services
NAME           TYPE      CLUSTER-IP   EXTERNAL-IP   PORT(S)        AGE
kubernetes     ClusterIP  10.96.0.1    <none>        443/TCP       35d
ms-favoritos   ClusterIP  10.109.104.76  <none>        9094/TCP      29h
ms-peliculas   ClusterIP  10.106.29.200  <none>        9091/TCP      29h
oracle-db      NodePort   10.109.200.103  <none>        1521:30011/TCP 33m
mnadmin@master:~/Intermedio/ws$ 
mnadmin@master:~/Intermedio/ws$ kubectl get pods
NAME                           READY   STATUS    RESTARTS   AGE
ms-favoritos-78bdcd75dd-cs5rf  1/1    Running   0          32m
ms-peliculas-d859d7c84-9rx56   1/1    Running   0          32m
oracle-db-cbf654876-z5kx9     1/1    Running   0          135m
mnadmin@master:~/Intermedio/ws$ 
mnadmin@master:~/Intermedio/ws$ curl http://10.109.104.76:9094/favoritos/user1
[]mnadmin@master:~/Intermedio/ws$ 
mnadmin@master:~/Intermedio/ws$ curl -X POST http://10.109.104.76:9094/favoritos/user1/1
{"id":1,"usuarioId":"user1","peliculaId":1}mnadmin@master:~/Intermedio/ws$ 
mnadmin@master:~/Intermedio/ws$ curl -X POST http://10.109.104.76:9094/favoritos/user1/2 | jq .
  % Total    % Received % Xferd  Average Speed   Time   Time Current
          Dload  Upload   Total Spent    Left Speed
100     43    0     43    0     0   1482      0 --::-- --::-- --::-- 1482
{
  "id": 2,
  "usuarioId": "user1",
  "peliculaId": 2
}
mnadmin@master:~/Intermedio/ws$ curl http://10.109.104.76:9094/favoritos/user1 | jq .
  % Total    % Received % Xferd  Average Speed   Time   Time Current
          Dload  Upload   Total Spent    Left Speed
100     89    0     89    0     0   4684      0 --::-- --::-- --::-- 4684
[
  {
    "id": 1,
    "usuarioId": "user1",
    "peliculaId": 1
  },
  {
    "id": 2,
    "usuarioId": "user1",
    "peliculaId": 2
  }
]
```



3.4. Agregando Secrets

- **Kubernetes Secrets** son objetos que almacenan datos sensibles, como contraseñas, claves API y certificados, de manera segura.
- A diferencia de los **ConfigMaps**, que se utilizan para almacenar información *no sensible*, los **Secrets** están diseñados específicamente para gestionar información *crítica* que debe protegerse.
- Kubernetes proporciona un mecanismo para crear, almacenar y acceder a estos Secrets de forma segura en los contenedores.

Agregando Secrets

Característica	ConfigMaps	Secrets
Propósito	Almacenar datos no sensibles	Almacenar datos sensibles
Cifrado	No se cifran	Cifrados en reposo y en tránsito
Acceso	Acceso menos controlado	Acceso altamente controlado
Tamaño de datos	Hasta 1 MB por ConfigMap	Hasta 1 MB por Secret.

Usa **Secrets** cuando necesitas manejar información sensible. Esto incluye credenciales de bases de datos, claves de API, o cualquier dato que, si se expone, podría comprometer la seguridad de tu aplicación.

Agregando Secrets | Tipos

1. Opaque

- Diseñados para almacenar datos arbitrarios, como texto o datos binarios, de forma de clave-valor.
- Su uso principal es para gestionar información confidencial, como contraseñas, certificados y tokens.

2. TLS

- Se utilizan para almacenar certificados TLS y claves privadas.
- Se utilizan para proteger la comunicación entre diferentes servicios dentro del clúster, así como entre servicios y clientes externos.

3. Dockercfg

- En el entorno del contenedor de Kubernetes los utiliza para extraer imágenes de registros privados de Docker.

4. SSH

- Se utilizan para almacenar claves privadas SSH, que pueden usarse para autenticarse con otros servidores o servicios.

Agregando Secrets | Tipos

```
apiVersion: v1
kind: Secret
metadata:
  name: nginx-secret
type: Opaque
data:
  SECRET_KEY: c2VjcmV0X2tleV92YWx1ZQ== # "secret_key_value" en base64
```

Base64

Agregando Secrets | kubectl

```
# Aplicar el Secrets  
kubectl apply -f my-secret.yaml  
  
# Crear Secrets directamente desde el archivo de entorno  
kubectl create secret generic my-secrets --from-env-file=secrets.env  
  
# Verificar que existen en el clúster.  
kubectl get secrets  
  
# Ver los detalles de un Secret  
kubectl describe secret my-secret
```

Agregando Secrets | Uso

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
spec:
  containers:
    - name: nginx-container
      image: nginx:1.23.4
      env:
        - name: SECRET_KEY
          valueFrom:
            secretKeyRef:
              name: nginx-secret
              key: SECRET_KEY
```

Agregando Secrets | Uso

```
mnadmin@master:~/Intermedio/ws2$ kubectl apply -f secret.yaml
secret/nginx-secret created
mnadmin@master:~/Intermedio/ws2$ kubectl apply -f deployment2.yaml
deployment.apps/nginx-deployment created
mnadmin@master:~/Intermedio/ws2$ kubectl get pods
NAME           READY   STATUS    RESTARTS   AGE
ms-favoritos-78bdcd75dd-cs5rf   1/1     Running   0          121m
ms-peliculas-d859d7c84-9rx56   1/1     Running   0          121m
nginx-deployment-747bdb75f4-tx2gf 1/1     Running   0          7s
oracle-db-cbf654876-z5kx9     1/1     Running   0          3h44m
mnadmin@master:~/Intermedio/ws2$ kubectl exec -it nginx-deployment-747bdb75f4-tx2gf -- printenv SECRET_KEY
secret_key_value
mnadmin@master:~/Intermedio/ws2$ echo -n "secret_key_value" | base64
c2Vjcmv0x2tlev92Ywx1ZQ==
mnadmin@master:~/Intermedio/ws2$ grep SECRET secret.yaml
SECRET_KEY: c2Vjcmv0x2tlev92Ywx1ZQ== # "secret_key_value" en base64
mnadmin@master:~/Intermedio/ws2$ |
```

Aplicando Secrets | Caso de estudio

```
apiVersion: v1
kind: Secret
metadata:
  name: ms-peliculas-secret
  labels:
    app: ms-peliculas
type: Opaque
data:
  DB_USERNAME: bXNfcGVsaWN1bGFzX3VzZXI= # "ms_peliculas_user" en base64
  DB_PASSWORD: c2VjdXJlcGFzc3dvcmQ=      # "securepassword" en base64
  API_KEY: YXBpX2tleV9tcy1wZWxpY3VsYXM= # "api_key_ms-peliculas" en base64
```

```
apiVersion: v1
kind: Secret
metadata:
  name: ms-favoritos-secret
  labels:
    app: ms-favoritos
type: Opaque
data:
  API_KEY: YXBpX2tleV9tcy1mYXZvcml0b3M= # "api_key_ms-favoritos" en base64
  TOKEN: dG9rZW5fbXMtZmF2b3JpdG9zX3NlY3JldA== # "token_ms-favoritos_secret" en base64
```

Aplicando Secrets | Caso de estudio

```
env:  
  - name: DB_USERNAME  
    valueFrom:  
      secretKeyRef:  
        name: ms-peliculas-secret  
        key: DB_USERNAME  
  - name: DB_PASSWORD  
    valueFrom:  
      secretKeyRef:  
        name: ms-peliculas-secret  
        key: DB_PASSWORD  
  - name: API_KEY  
    valueFrom:  
      secretKeyRef:  
        name: ms-peliculas-secret  
        key: API_KEY
```

Aplicando Secrets | Caso de estudio

- Comando base64
 - `echo -n "tu_texto_a_codificar" | base64`
- Si tienes un texto codificado en base64 y deseas ver el contenido original.
 - `echo "texto_codificado" | base64 --decode`
- Si tienes un texto codificado en base64 y deseas ver el contenido original.
 - `base64 credentials.txt`
- Si tienes un texto codificado en base64 y deseas ver el contenido original.
 - `base64 --decode archivo_base64.txt`
- Ayuda
 - `base64 --help`

-n en echo: Asegúrate de usar `-n` con `echo` para evitar que se codifique una nueva línea extra al final.

Resumen

- En esta tercera unidad se exploraron los conceptos y herramientas fundamentales para gestionar configuraciones en Kubernetes de manera eficiente, utilizando ConfigMaps y Secrets como elementos clave. Además, se abordó cómo integrar estas configuraciones en los Deployments para aplicarlas a los contenedores de las aplicaciones.



Práctica 3.1. Verificación del clúster

Objetivo:

Al finalizar esta práctica, serás capaz de realizar una validación completa de un clúster de Kubernetes para asegurar que está configurado correctamente y es funcional.

Planteamiento e instrucciones:

Realiza un seguimiento de los pasos indicados en la *Guía de Laboratorios* para llevar a cabo la tarea correspondiente en el siguiente enlace: [Plataforma de Laboratorios | DOCK_KUB_INT](#)

https://github.com/NetecGk/DOCK_KUB_INT



Tiempo para esta actividad:
30 minutos.

Resultado esperado

- Los nodos están Ready.
- Los Pods del espacio kube-system están Running o Completed.
- Los Pods creados por el Deployment están Running.
- Un ConfigMap básico fue creado y es accesible.
- Un servicio básico expone el Deployment correctamente y permite la comunicación interna.

"Si todos estos pasos pasan, el clúster de Kubernetes está listo para manejar Deployments, ConfigMaps y Servicios".

Práctica 3.2. Customización de propiedades

Objetivo:

Al finalizar esta práctica, serás capaz de retomar los microservicios desarrollados anteriormente, ajustando su configuración mediante la definición de propiedades en archivos `application.properties` y variables de ambiente para su despliegue en un entorno basado en contenedores.

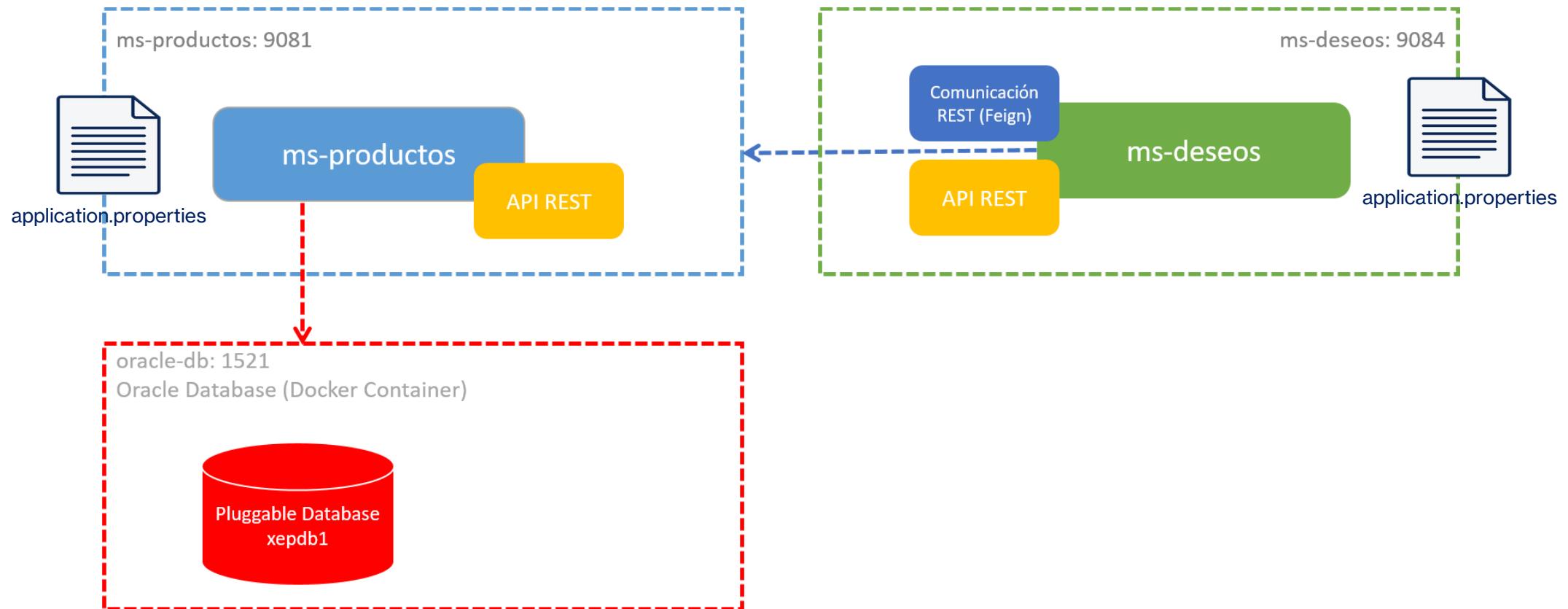
Planteamiento e instrucciones:

Realiza un seguimiento de los pasos indicados en la *Guía de Laboratorios* para llevar a cabo la tarea correspondiente en el siguiente enlace: [Plataforma de Laboratorios | DOCK_KUB_INT](#)



Tiempo para esta actividad:
20 minutos.

Resultado esperado



Resultado esperado

```
spring.application.name=ms-productos
server.port=9081

# Configuracion de la base de datos
spring.datasource.url=jdbc:oracle:thin:@dki-oradb:1521/XEPDB1
spring.datasource.username=dkuser
spring.datasource.password=dkpassword
spring.datasource.driver-class-name=oracle.jdbc.OracleDriver

# Configuracion de JPA e Hibernate
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.OracleDialect
```

Resultado esperado

```
spring.application.name=${APP_NAME:ms-productos}
server.port=${SERVER_PORT:9081}

# Configuracion de la base de datos
spring.datasource.url=${DB_URL:jdbc:oracle:thin:@<oracle-kubernetes>:1521/<PDB>}
spring.datasource.username=${DB_USERNAME:dkuser}
spring.datasource.password=${DB_PASSWORD:dkpassword}
spring.datasource.driver-class-name=${DB_DRIVER:oracle.jdbc.OracleDriver}

# Configuracion de JPA e Hibernate
spring.jpa.hibernate.ddl-auto=${JPA_DDL_AUTO:update}
spring.jpa.show-sql=${JPA_SHOW_SQL:true}
spring.jpa.properties.hibernate.dialect=${HIBERNATE_DIALECT:org.hibernate.dialect.OracleDialect}
```

Resultado esperado

```
spring.application.name=ms-deseos  
server.port=9084
```

```
spring.application.name=${APP_NAME:ms-deseos}  
server.port=${SERVER_PORT:9084}  
  
# URL del microservicio ms-productos  
productos.service.url=${PRODUCTOS_SERVICE_URL:http://ms-productos:9081}
```

Práctica 3.3. Docker Registry

Objetivo:

Al finalizar esta práctica, serás capaz de crear una imagen Docker a partir del código de los microservicios y publicarla en Docker Hub, utilizando buenas prácticas para el etiquetado y manejo de versiones.

Planteamiento e instrucciones:

Realiza un seguimiento de los pasos indicados en la *Guía de Laboratorios* para llevar a cabo la tarea correspondiente en el siguiente enlace: [Plataforma de Laboratorios | DOCK_KUB_INT](#)



Tiempo para esta actividad:
25 minutos.

Resultado esperado

- ✓ ./mvnw clean package -Dmaven.test.skip=false
- ✓ touch Dockerfile
- ✓ docker build -t <nombre>:<version> .
- ✓ docker tag <nombre> <usuario-dockerhub>/<imagen>:<tag>
- ✓ docker login
- ✓ docker push <usuario-dockerhub>/<imagen>:<tag>
- ✓ docker search <usuario-dockerhub>/<imagen>:<tag>

Resultado esperado

```
$ docker search [tu_usuario_dockerhub]/ms-productos
```

NAME	DESCRIPTION	STARS	OFFICIAL	AUTOMATED
[tú_usuario_dockerhub]/ms-productos	Microservicio para la...	0	N/A	N/A

```
$ docker search [tu_usuario_dockerhub]/ms-deseos
```

NAME	DESCRIPTION	STARS	OFFICIAL	AUTOMATED
[tú_usuario_dockerhub]/ms-deseos	Microservicio para ge...	0	N/A	N/A

Práctica 3.4. ConfigMaps & Secrets

Objetivo:

Al finalizar esta práctica, serás capaz de crear manifiestos YAML para configurar ConfigMaps y Secrets en Kubernetes.

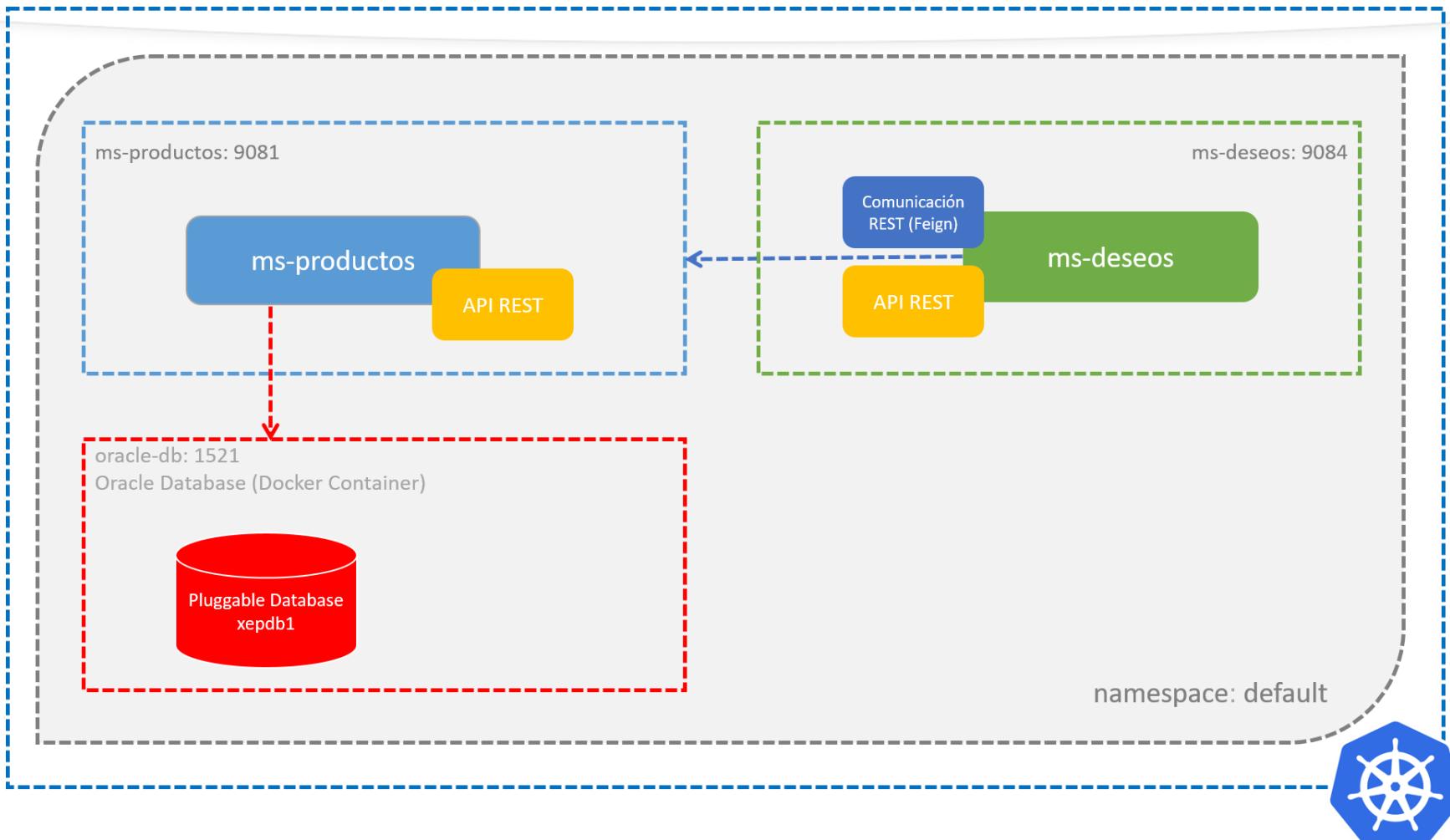
Planteamiento e instrucciones:

Realiza un seguimiento de los pasos indicados en la *Guía de Laboratorios* para llevar a cabo la tarea correspondiente en el siguiente enlace: [Plataforma de Laboratorios | DOCK_KUB_INT](#)



Tiempo para esta actividad:
30 minutos.

Resultado esperado



Resultado esperado

NAME	DATA	AGE
ms-productos-configmap	3	2d
ms-deseos-configmap	2	2d

NAME	TYPE	DATA	AGE
ms-productos-secret	Opaque	2	2d
ms-deseos-secret	Opaque	2	2d

- ✓ kubectl get configmaps
- ✓ kubectl get secrets
- ✓ kubectl describe configmap ms-productos-config
- ✓ kubectl describe configmap ms-deseos-config
- ✓ kubectl describe secret ms-productos-secrets
- ✓ kubectl describe secret ms-deseos-secrets

Resultado esperado

- ✓ `kubectl apply -f ms-productos-deployment.yaml`
- ✓ `kubectl apply -f ms-deseos-deployment.yaml`
- ✓ `kubectl apply -f ms-productos-service.yaml`
- ✓ `kubectl apply -f ms-deseos-service.yaml`

- ✓ `kubectl describe pod <número-pod>`
- ✓ `kubectl logs <número-pod>`

- ✓ `kubectl get pods`
- ✓ `kubectl get deploys`
- ✓ `kubectl get services -o wide`
- ✓ `kubectl get configmaps`
- ✓ `kubeclt get secrets`

- ✓ `curl http://<localhost|node-ip>:puerto/api-productos`
- ✓ `curl -X POST http://<localhost|node-ip>:puerto/deseos/user1/1`

Referencias Bibliográficas

- III. Config <https://12factor.net/config>
- ConfigMaps
<https://kubernetes.io/docs/concepts/configuration/configmap/>
- Configure a Pod to Use a ConfigMaps
<https://kubernetes.io/docs/tasks/configure-pod-container/configure-pod-configmap/>
- Secrets
<https://kubernetes.io/docs/concepts/configuration/secret/>



Unidad 4

Kubernetes: Spring Kubernetes

Objetivos:

- Comprende la integración de Spring Cloud con un clúster de Kubernetes.
- Configurar microservicios con Spring Cloud Kubernetes y desplegarlos en un clúster.
- Gestionar health check y recursos de los Pods.
- Realizar cambios en Kubernetes y probar aplicaciones Spring Boot.
- Visualizar y analizar el balanceo de carga en Kubernetes con metadata de los Pods.

4.1. Introducción Spring Cloud Kubernetes

¿Qué es Spring Cloud Kubernetes?

- Este proyecto integra las características nativas de Kubernetes, como la **configuración centralizada**, el **descubrimiento de servicios** y **el balanceo de carga**, con el ecosistema de Spring Cloud.
- Proporciona herramientas e integraciones específicas para aplicaciones Spring que se ejecutan en un entorno Kubernetes.

*Aunque Spring Cloud Kubernetes facilita el desarrollo de aplicaciones nativas de la nube, **no es imprescindible** para ejecutar aplicaciones Spring Boot en Kubernetes.*

Introducción Spring Cloud Kubernetes



Client Application

(Postman, Frontend, External APIx)

Spring Cloud Kubernetes

- **Discovery Client:** Descubre y registra servicios.
- **ConfigMap y Secrets:** Gestión de configuraciones.
- **Load Balancer:** Balancea tráfico entre Pods.
- **Health Check:** Liveness & Readiness Probes.



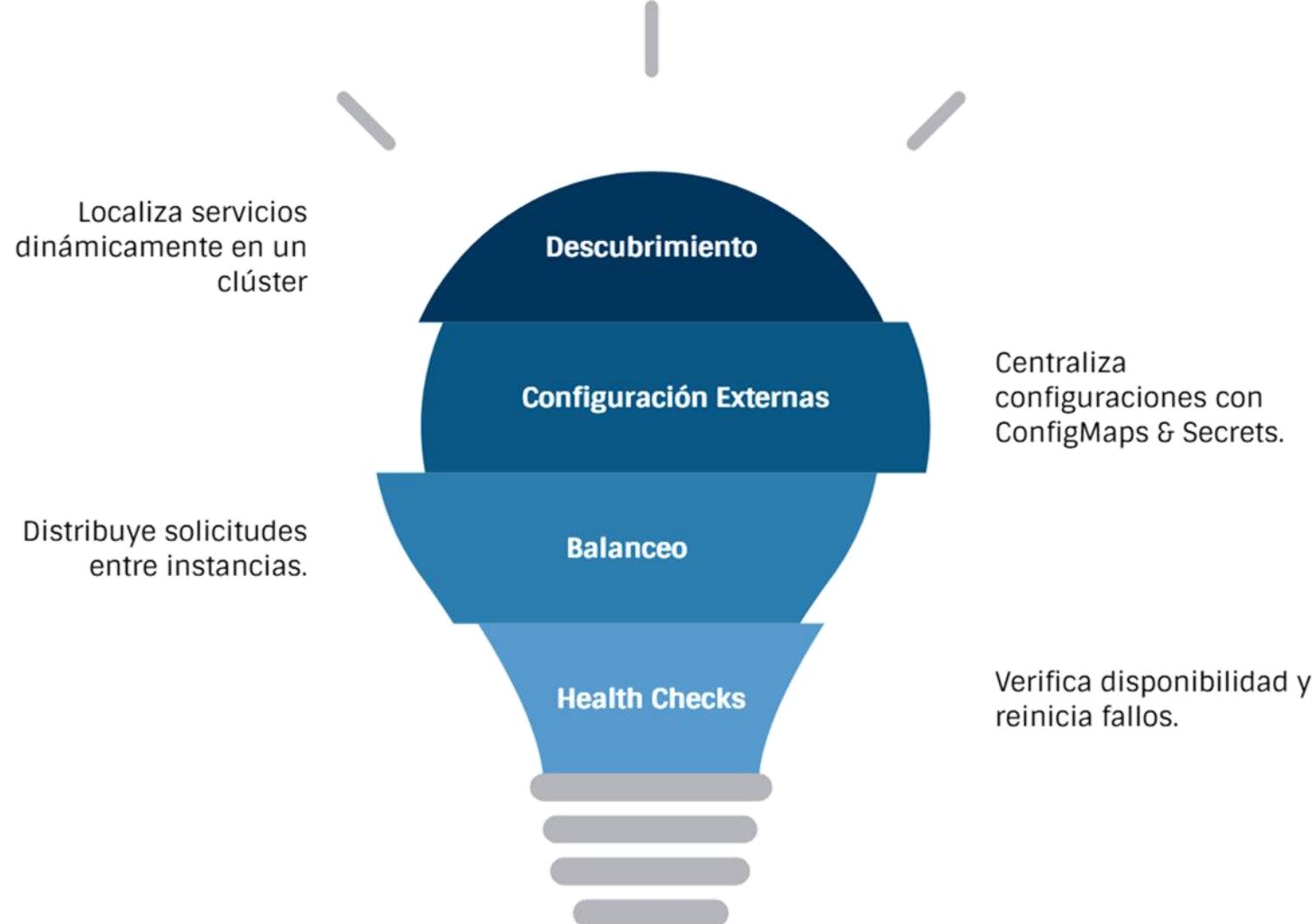
Kubernetes (Cluster Layer)

- **Pods:** Alojan contenedores con microservicios.
- **Services:** Gestionan el acceso a los Pods.
- **Deployments:** Controlan el ciclo de vida de Pods.
- **Nodes:** Infraestructura física o virtual.

Infrastructure (Cloud or On-Premise)

(AWS, Google Cloud, Azure, Bare Metal, etc.)

Introducción Spring Cloud Kubernetes



Introducción | Descubrimiento

Spring Cloud Kubernetes proporciona un cliente de descubrimiento que permite a las aplicaciones Spring identificar y comunicarse con otros servicios dentro del clúster de Kubernetes, sin necesidad de configurar manualmente direcciones IP o puertos.

- Simplificación del enrutamiento interno entre microservicios.
- Mayor resiliencia y flexibilidad al gestionar escalabilidad y despliegues.

- Utiliza el Service Discovery de Kubernetes basado en Endpoints y Labels.
- Los servicios se identifican mediante nombres configurados en Kubernetes y en los archivos application.properties de Spring Boot.
- Esto elimina la necesidad de gestionar la red directamente, ya que Kubernetes resuelve los nombres a direcciones IP y puertos dinámicos.

Introducción | Configuración externas

Permite a las aplicaciones Spring Boot consumir configuraciones almacenadas en ConfigMaps y Secrets de Kubernetes, lo que facilita la externalización y centralización de configuraciones.

- Las aplicaciones Spring pueden acceder a los valores definidos en ConfigMaps o Secrets mediante propiedades injectadas.
- Spring Cloud Kubernetes monitorea estos recursos y puede actualizar dinámicamente las configuraciones en los microservicios.

- Separación de configuración y código, mejorando la flexibilidad y seguridad.
- Soporte para múltiples entornos (desarrollo, prueba, producción) con configuraciones específicas por entorno.

Introducción | Balanceo

Implementa balanceo de carga a nivel de aplicación utilizando Spring Cloud LoadBalancer para distribuir solicitudes entre diferentes Pods que ejecutan una instancia del microservicio.

- Kubernetes asigna un servicio a los Pods mediante un selector de etiquetas, y Spring utiliza esta información para enrutar las solicitudes a los Pods adecuados.
- Los algoritmos de balanceo incluyen:
 - **Round-Robin** (por defecto): Alterna solicitudes de manera equitativa entre las instancias disponibles.
 - Algoritmos personalizados.

- Asegura una distribución uniforme del tráfico.
- Facilita la escalabilidad horizontal.

Introducción | Health Checks

Proporciona soporte integrado para implementar los Liveness y Readiness Probes, para garantizar la disponibilidad y estabilidad de los Pods en Kubernetes.

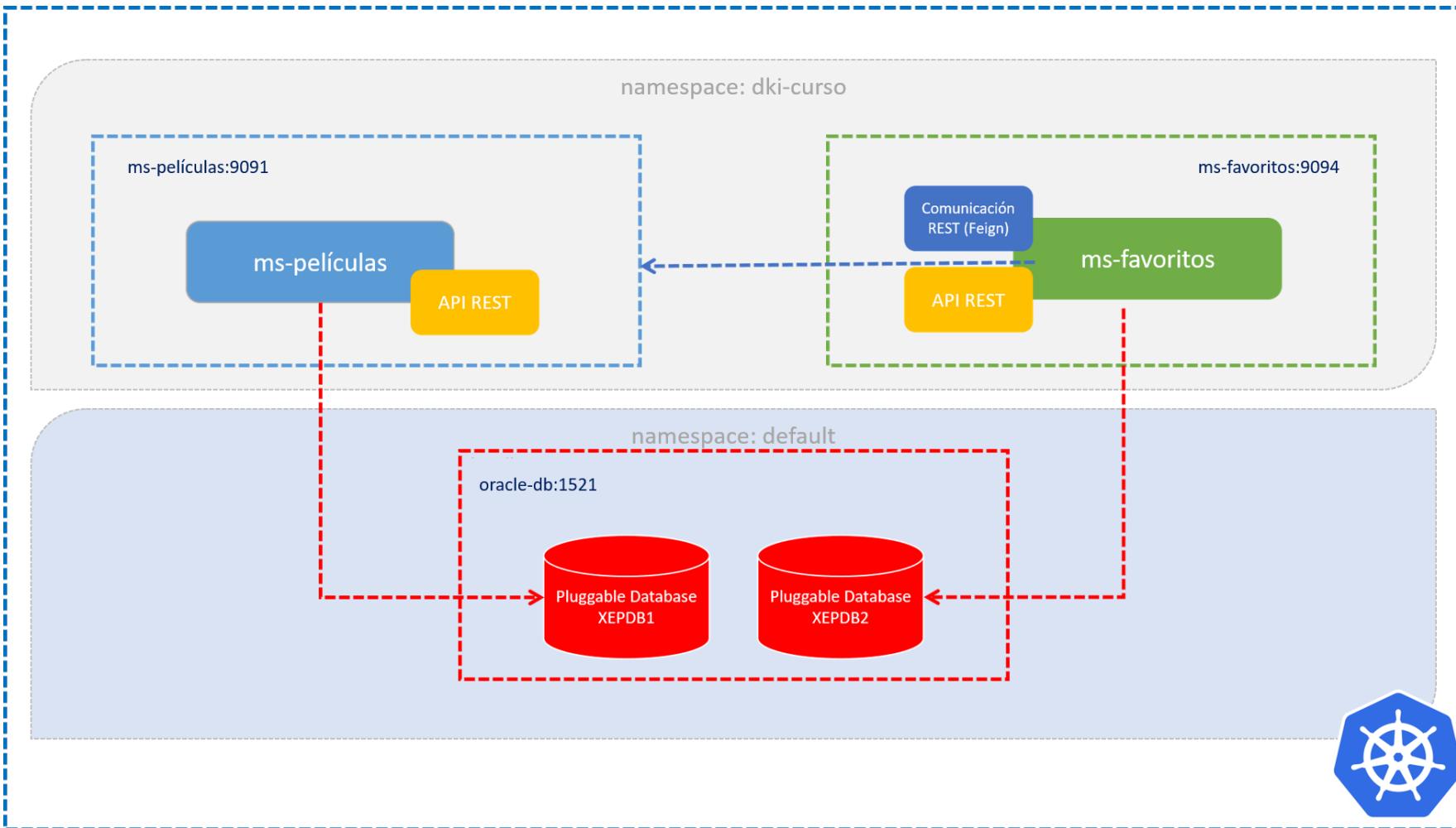
- Liveness Probe:
 - Verifica si el contenedor está vivo (es decir, si no está bloqueado o en un estado fallido).
 - Kubernetes reinicia el contenedor si este probe falla.
 - Utiliza endpoints como /actuator/health/liveness proporcionados por Spring Boot Actuator.
- Readiness Probe:
 - Indica si el contenedor está listo para recibir tráfico.
 - Si falla, Kubernetes excluye temporalmente el Pod del balanceo de carga.
 - Utiliza endpoints como /actuator/health/readiness.

- Detecta fallos de aplicaciones y garantiza alta disponibilidad.
- Reduce el tiempo de inactividad al reiniciar contenedores defectuosos automáticamente.

4.2. Configurando nuestros msvcs con Spring Cloud Kubernetes

- ¿Puedes recordar cuáles son los microservicios del caso de estudio presentado en el material y cómo se relacionan entre ellos?
- Los microservicios son **ms-películas** y **ms-favoritos**, donde **ms-favoritos** depende de **ms-películas** para verificar la existencia y obtener detalles de las películas.

Configurando nuestros MS



Configurando nuestros MS

1 Agregar Dependencias

Actualizar el pom.xml para incluir dependencias relacionadas con Spring Cloud Kubernetes.

2 Configurar Descubrimiento de Servicios

Añadir la anotación @EnableDiscoveryClient en la clase principal.

3 Configurar Feign

Personalizar los clientes Feign para comunicarse con otros microservicios utilizando únicamente el nombre registrado del servicio.

6 Registrar Imagen Docker

Crear, etiquetar y subir la imagen Docker al Docker Hub.

5 Compilar & Empaquetar

Compilar el código y generar el archivo JAR para la construcción de Imagen Docker.

4 Configurar Propiedades

Ajustar el application.properties para definir configuraciones clave.

Configurando | pom.xml

```
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-kubernetes-client</artifactId>
</dependency>
```

```
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-kubernetes-client-config</artifactId>
</dependency>
```

```
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-kubernetes-client-loadbalancer</artifactId>
</dependency>
```

<https://github.com/spring-cloud/spring-cloud-release/wiki/Supported-Versions>

Configurando | `@EnableDiscoveryClient`

- Cuando se usa junto con Spring Cloud Kubernetes, `@EnableDiscoveryClient` permite que la aplicación se conecte al entorno Kubernetes, aprovechando su capacidad para:
 - Consultar información sobre servicios expuestos en el clúster.
 - Resolver nombres de servicios en Kubernetes, en lugar de usar IPs o URLs fijas.
 - Facilitar la comunicación entre microservicios al abstraer detalles específicos como direcciones de red o puertos.

Configurando | @EnableDiscoveryClient

```
package com.netec.dk;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import
org.springframework.cloud.client.discovery.EnableDiscoveryClient;
import org.springframework.cloud.openfeign.EnableFeignClients;

@EnableDiscoveryClient
@SpringBootApplication
@EnableFeignClients(basePackages = "com.netec.dk.feign")
public class MsFavoritosApplication {
    public static void main(String[] args) {
        SpringApplication.run(MsFavoritosApplication.class, args);
    }
}
```

Configurando | Feign

- Cuando usas `@FeignClient`, no necesitas especificar la URL completa del microservicio.
- Puedes usar el nombre del servicio configurado en `spring.application.name`.
- Feign trabajará en conjunto con el Discovery Client para resolver la dirección del servicio automáticamente.

Configurando | Feign

```
package com.netec.dk.feign;

import org.springframework.cloud.openfeign.FeignClient;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import com.netec.dk.model.PeliculaDTO;

//@FeignClient(name = "ms-peliculas", url = "${ms-peliculas.url}")
@FeignClient(name = "ms-peliculas")
public interface PeliculaClient {
    @GetMapping("/api/peliculas/{id}")
    PeliculaDTO obtenerPeliculaPorId(@PathVariable Long id);
}
```

Configurando | application.properties

- Aunque `@EnableDiscoveryClient` se integra con el sistema de DNS de Kubernetes, nos asegura que las búsquedas de nombres de servicio se resuelven correctamente en todos los entornos. Puedes agregar las siguientes configuraciones:

```
# Comunicacion con ms-peliculas
# mspeliculas.url=${MSPELICULAS_URL:http://ms-peliculas:9091}

# Configuracion para Kubernetes DNS
spring.cloud.kubernetes.discovery.enabled=true
spring.cloud.kubernetes.secrets.enable-api=true
spring.cloud.kubernetes.discovery.all-namespaces=true
```

- **discovery.enabled** activa el descubrimiento de servicios en Kubernetes.
- **secrets.enable-api** habilita la lectura de secretos desde Kubernetes.
- **discovery.all-namespaces** permite descubrir servicios en todos los namespaces del clúster.

Configurando | JAR

El comando utilizado:

```
.\mvnw clean package -Dmaven.test.skip=true -U
```

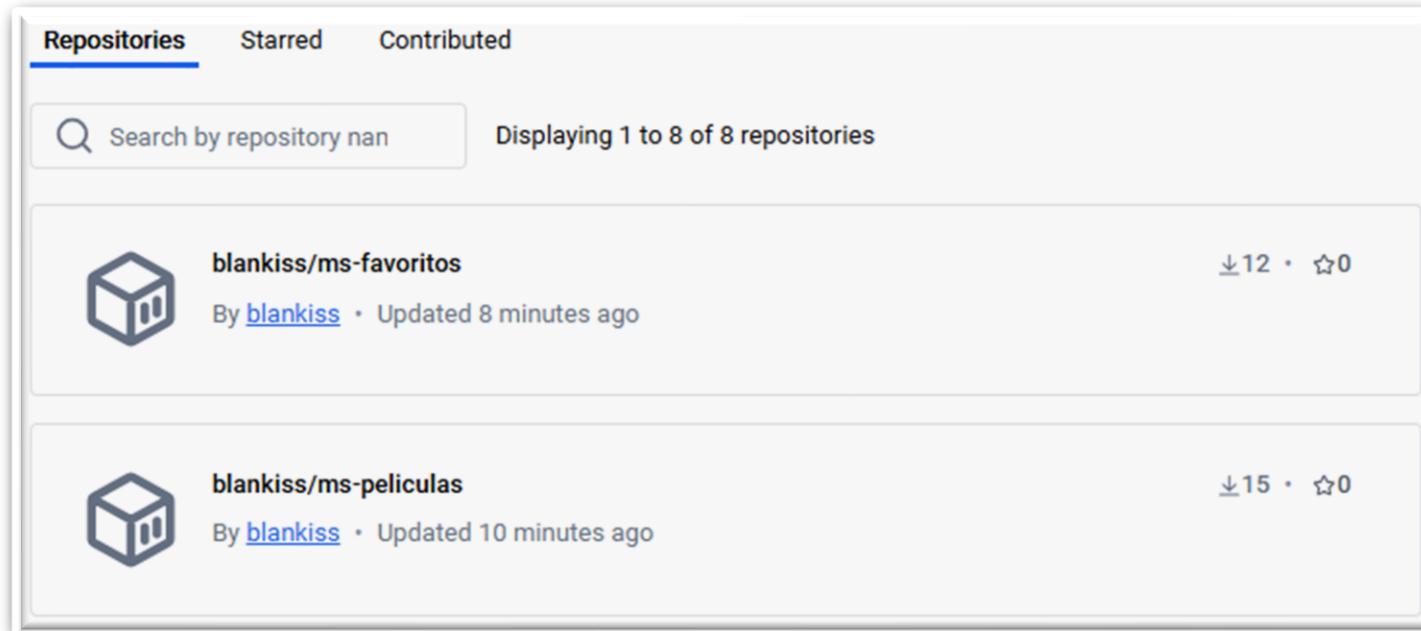
- **.\mvnw**: Utiliza el wrapper de Maven.
- **clean**: Limpia los artefactos generados en compilaciones previas.
 - Elimina la carpeta target.
- **package**: Compila el código ejecuta las etapas necesaria.
 - genera el empaquetado (JAR o WAR).

Configurando | JAR

- **-Dmaven.test.skip=true:** Omite la ejecución de las pruebas durante el proceso de empaquetado.
 - Esto es útil cuando quieras evitar que las pruebas detengan el flujo de construcción (*aunque recuerda que esto Podría ocultar errores en el código*).
- **-U:** Fuerza a Maven a actualizar las dependencias desde los repositorios remotos, incluso si las versiones locales ya existen.
 - Esto soluciona problemas cuando los artefactos no se descargan correctamente o están desactualizados.

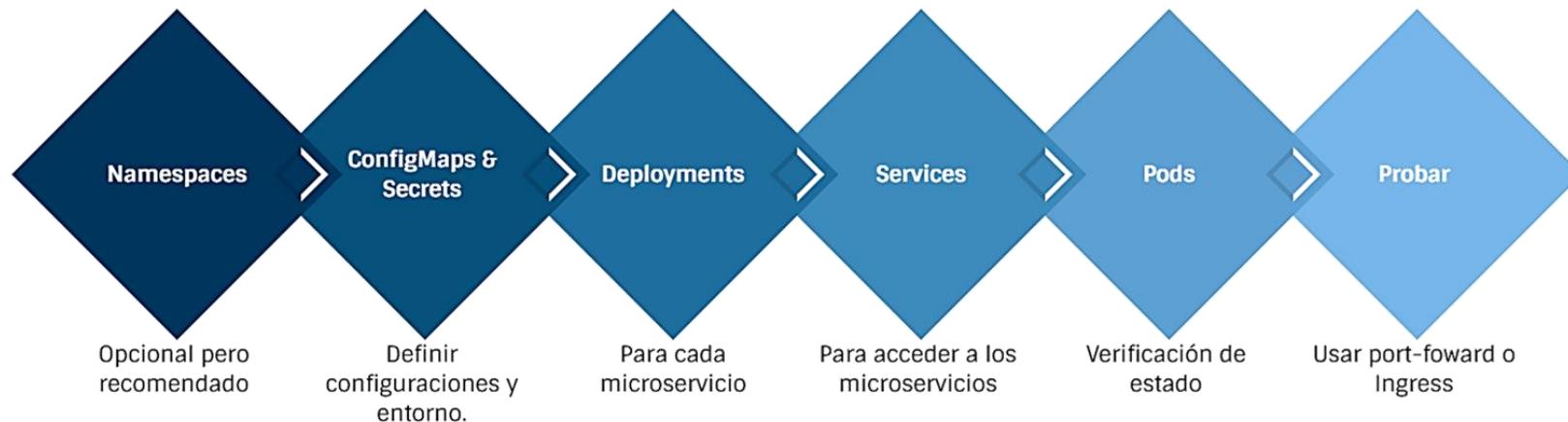
Configurando | Docker Registry

- Flujo típico:
 1. **Construcción:** Crear la imagen (docker build).
 2. **Etiqueta:** Preparar la imagen con el nombre del repositorio (docker tag).
 3. **Publicación:** Enviar la imagen al repositorio (docker push).



4.3. Desplegando una aplicación Spring en nuestro clúster

- ¿Cómo Podemos desplegar y conectar microservicios como ms-películas y ms-favoritos en un clúster de K8s?



Desplegando | Namespaces

```
mnadmin@master:~/Intermedio/ws3$ cat ms-namespace.yaml
apiVersion: v1
kind: Namespace
metadata:
  name: dki-curso

mnadmin@master:~/Intermedio/ws3$ kubectl get namespaces
NAME          STATUS    AGE
app1          Terminating   18d
default        Active     36d
development    Active     19d
dki-curso      Active     93m
ingress-nginx  Active     17d
kube-node-lease  Active     36d
kube-public    Active     36d
kube-system    Active     36d
my-namespace   Terminating 20d
mnadmin@master:~/Intermedio/ws3$ kubectl describe namespace dki-curso
Name:           dki-curso
Labels:         kubernetes.io/metadata.name=dki-curso
Annotations:   <none>
Status:        Active

No resource quota.

No LimitRange resource.
mnadmin@master:~/Intermedio/ws3$ |
```

```
apiVersion: v1
kind: Namespace
metadata:
  name: dki-curso
```

Desplegando | Namespaces

```
mnadmin@master:~/Intermedio/ws3$  
mnadmin@master:~/Intermedio/ws3$  
mnadmin@master:~/Intermedio/ws3$  
mnadmin@master:~/Intermedio/ws3$ kubectl get all,cm,secret,ing -n dki-curso  
NAME READY STATUS RESTARTS AGE  
pod/ms-peliculas-5795b5667-nddk6 1/1 Running 0 40m  
  
NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE  
service/ms-peliculas NodePort 10.111.96.188 <none> 9091:30672/TCP 72m  
  
NAME READY UP-TO-DATE AVAILABLE AGE  
deployment.apps/ms-peliculas 1/1 1 1 40m  
  
NAME DESIRED CURRENT READY AGE  
replicaset.apps/ms-peliculas-5795b5667 1 1 1 40m  
  
NAME DATA AGE  
configmap/kube-root-ca.crt 1 104m  
configmap/ms-peliculas-config 7 95m  
  
NAME TYPE DATA AGE  
secret/ms-peliculas-secrets Opaque 2 103m  
mnadmin@master:~/Intermedio/ws3$
```

Desplegando | ConfigMaps

```
mnadmin@master:~/Intermedio/ws3$  
mnadmin@master:~/Intermedio/ws3$  
mnadmin@master:~/Intermedio/ws3$ kubectl get cm -n dki-curso  
NAME          DATA   AGE  
kube-root-ca.crt    1     106m  
ms-peliculas-config 7     98m  
mnadmin@master:~/Intermedio/ws3$ cat ms-peliculas-configmaps.yaml  
apiVersion: v1  
kind: ConfigMap  
metadata:  
  name: ms-peliculas-config  
  namespace: dki-curso  
data:  
  SPRING_APPLICATION_NAME: ms-peliculas  
  SERVER_PORT: "9091"  
  DB_URL: jdbc:oracle:thin:@oracle-db.default.svc.cluster.local:1521/XEPDB1  
  DB_DRIVER_CLASS_NAME: oracle.jdbc.OracleDriver  
  JPA_DDL_AUTO: update  
  JPA_SHOW_SQL: "true"  
  JPA_DIALECT: org.hibernate.dialect.oracleDialect  
mnadmin@master:~/Intermedio/ws3$
```

DB_URL: jdbc:oracle:thin:@oracle-db.default.svc.cluster.local:1521/XEPDB1

Desplegando | ConfigMaps

```
DB_URL: jdbc:oracle:thin:@oracle-db.default.svc.cluster.local:1521/XEPDB1
```

- **jdbc:oracle:thin:**
 - Especifica el uso del controlador JDBC Thin Driver de Oracle para conectarse a la base de datos.
- **oracle-db.default.svc.cluster.local**
 - El DNS interno de Kubernetes para el servicio que expone tu base de datos Oracle.
 - oracle-db: Nombre del servicio que apunta al Pod de Oracle DB.
 - default: Namespace donde se encuentra el servicio.
 - svc.cluster.local: Sufijo estándar de DNS para servicios en Kubernetes.

Desplegando | ConfigMaps

```
DB_URL: jdbc:oracle:thin:@oracle-db.default.svc.cluster.local:1521/XEPDB1
```

- **1521**
 - Puerto estándar utilizado por Oracle Database.
- **XEPDB1**
 - Nombre de la base de datos o Pluggable Database (PDB) a la que te estás conectando.

Desplegando | Secrets

```
mnadmin@master:~/Intermedio/ws3$  
mnadmin@master:~/Intermedio/ws3$  
mnadmin@master:~/Intermedio/ws3$  
mnadmin@master:~/Intermedio/ws3$ cat ms-peliculas-secrets.yaml  
  
apiVersion: v1  
kind: Secret  
metadata:  
  name: ms-peliculas-secrets  
  namespace: dki-curso  
type: Opaque  
data:  
  DB_USERNAME: ZGt1c2Vy  # dkuser en Base64  
  DB_PASSWORD: ZGtwYXNzd29yZA== # dkpassword en Base64  
  
mnadmin@master:~/Intermedio/ws3$ kubectl get secrets -n dki-curso  
NAME          TYPE      DATA  AGE  
ms-peliculas-secrets  Opaque    2     121m  
mnadmin@master:~/Intermedio/ws3$ |
```

Desplegando | Deployments

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: ms-peliculas
  namespace: dki-curso
spec:
  replicas: 1
  selector:
    matchLabels:
      app: ms-peliculas
  template:
    metadata:
      labels:
        app: ms-peliculas
    spec:
      containers:
        - name: ms-peliculas
          image: blankiss/ms-peliculas:1.0.4
          ports:
            - containerPort: 9091
          env:
            - name: SPRING_APPLICATION_NAME
              valueFrom:
                configMapKeyRef:
                  name: ms-peliculas-config
                    key: SPRING_APPLICATION_NAME
                    - name: SERVER_PORT
                      valueFrom:
                        configMapKeyRef:
                          name: ms-peliculas-config
                          key: SERVER_PORT
                    - name: DB_URL
                      valueFrom:
                        configMapKeyRef:
                          name: ms-peliculas-config
                          key: DB_URL
                    - name: DB_DRIVER_CLASS_NAME
                      valueFrom:
                        configMapKeyRef:
                          name: ms-peliculas-config
                          key: DB_DRIVER_CLASS_NAME
                    - name: JPA_DDL_AUTO
                      valueFrom:
                        configMapKeyRef:
                          name: ms-peliculas-config
                          key: JPA_DDL_AUTO
# Parte la configuración omitida
```

Desplegando | Deployments

```
mnadmin@master:~/Intermedio/ws3$  
mnadmin@master:~/Intermedio/ws3$  
mnadmin@master:~/Intermedio/ws3$  
mnadmin@master:~/Intermedio/ws3$ kubectl get deployment -n dki-curso  
NAME      READY   UP-TO-DATE   AVAILABLE   AGE  
ms-peliculas   1/1     1          1          69m  
mnadmin@master:~/Intermedio/ws3$  
mnadmin@master:~/Intermedio/ws3$ kubectl describe deployment -n dki-curso  
Name:           ms-peliculas  
Namespace:      dki-curso  
CreationTimestamp:  Wed, 27 Nov 2024 12:26:48 -0600  
Labels:          <none>  
Annotations:    deployment.kubernetes.io/revision: 1  
Selector:        app=ms-peliculas  
Replicas:       1 desired | 1 updated | 1 total | 1 available | 0 unavailable  
StrategyType:   RollingUpdate  
MinReadySeconds: 0  
RollingUpdateStrategy: 25% max unavailable, 25% max surge  
Pod Template:  
  Labels:  app=ms-peliculas  
  Containers:  
    ms-peliculas:  
      Image:      blankiss/ms-peliculas:1.0.4  
      Port:       9091/TCP  
      Host Port:  0/TCP  
      Environment:  
        SPRING_APPLICATION_NAME: <set to the key 'SPRING_APPLICATION_NAME' of config map 'ms-peliculas-config'> Optional: false  
        SERVER_PORT: <set to the key 'SERVER_PORT' of config map 'ms-peliculas-config'> Optional: false  
        DB_URL: <set to the key 'DB_URL' of config map 'ms-peliculas-config'> Optional: false  
        DB_DRIVER_CLASS_NAME: <set to the key 'DB_DRIVER_CLASS_NAME' of config map 'ms-peliculas-config'> Optional: false  
        JPA_DDL_AUTO: <set to the key 'JPA_DDL_AUTO' of config map 'ms-peliculas-config'> Optional: false  
        JPA_SHOW_SQL: <set to the key 'JPA_SHOW_SQL' of config map 'ms-peliculas-config'> Optional: false  
        JPA_DIALECT: <set to the key 'JPA_DIALECT' of config map 'ms-peliculas-config'> Optional: false  
        DB_USERNAME: <set to the key 'DB_USERNAME' in secret 'ms-peliculas-secrets'> Optional: false  
        DB_PASSWORD: <set to the key 'DB_PASSWORD' in secret 'ms-peliculas-secrets'> Optional: false  
      Mounts:  
      Volumes:  
      Node-Selectors:  
      Tolerations:  
      Conditions:  
        Type   Status  Reason  
          
```

Desplegando | Services

```
mnadmin@master:~/Intermedio/ws3$ cat ms-peliculas-service.yaml
apiVersion: v1
kind: Service
metadata:
  name: ms-peliculas
  namespace: dki-curso
spec:
  selector:
    app: ms-peliculas
  ports:
    - protocol: TCP
      port: 9091
      targetPort: 9091
    type: NodePort

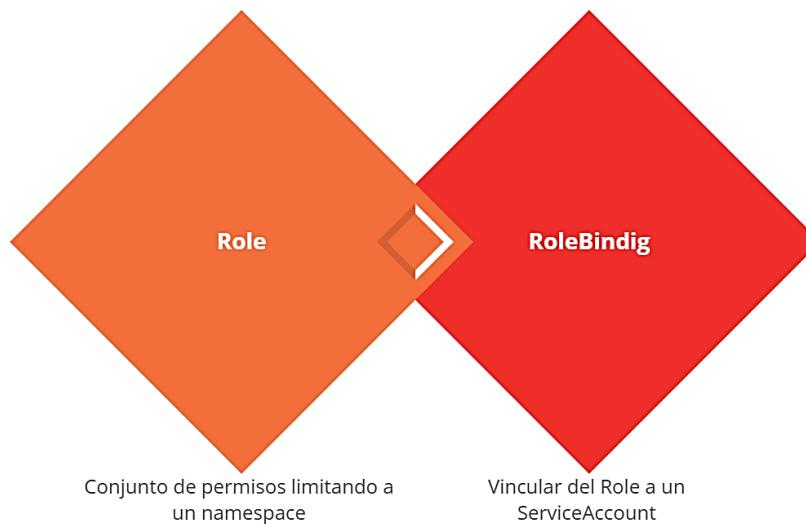
mnadmin@master:~/Intermedio/ws3$ kubectl get services -n dki-curso
NAME        TYPE      CLUSTER-IP      EXTERNAL-IP      PORT(S)          AGE
ms-peliculas   NodePort    10.111.96.188    <none>           9091:30672/TCP   112m
mnadmin@master:~/Intermedio/ws3$
mnadmin@master:~/Intermedio/ws3$ kubectl describe service ms-peliculas -n dki-curso
Name:                   ms-peliculas
Namespace:              dki-curso
Labels:                 <none>
Annotations:            <none>
Selector:               app=ms-peliculas
Type:                   NodePort
IP Family Policy:       singlestack
IP Families:            IPv4
IP:                     10.111.96.188
IPs:                    10.111.96.188
Port:                  <unset>  9091/TCP
TargetPort:              9091/TCP
NodePort:               <unset>  30672/TCP
Endpoints:              10.34.0.6:9091
Session Affinity:       None
External Traffic Policy: cluster
Events:                <none>
mnadmin@master:~/Intermedio/ws3$ |
```

Desplegando | Pods

```
mnadmin@master:~/Intermedio/ws3$  
mnadmin@master:~/Intermedio/ws3$  
mnadmin@master:~/Intermedio/ws3$ kubectl get pods -n dki-curso  
NAME           READY   STATUS    RESTARTS   AGE  
ms-peliculas-5795b5667-nddk6   1/1     Running   0          74m  
mnadmin@master:~/Intermedio/ws3$  
mnadmin@master:~/Intermedio/ws3$  
mnadmin@master:~/Intermedio/ws3$ kubectl logs ms-peliculas-5795b5667-nddk6 -n dki-curso | tail -15  
2024-11-27T18:27:06.218Z INFO 1 --- [ms-peliculas] [oller-V1Service] i.k.client.informer.cache.Controller :  
sync & reflector runnable  
2024-11-27T18:27:06.219Z INFO 1 --- [ms-peliculas] [oller-V1Service] i.k.client.informer.cache.Controller :  
due to 0 full resync period  
2024-11-27T18:27:06.228Z INFO 1 --- [ms-peliculas] [s.V1Endpoints-1] i.k.c.informer.cache.ReflectorRunnable :  
odels.V1Endpoints#Start listing and watching...  
2024-11-27T18:27:06.247Z INFO 1 --- [ms-peliculas] [els.V1Service-1] i.k.c.informer.cache.ReflectorRunnable :  
odels.V1Service#Start listing and watching...  
2024-11-27T18:27:07.209Z INFO 1 --- [ms-peliculas] [pool-5-thread-1] s.c.k.c.d.KubernetesDiscoveryClientUtils :  
o be fully loaded..  
2024-11-27T18:27:07.216Z INFO 1 --- [ms-peliculas] [main] s.c.k.c.d.KubernetesDiscoveryClientUtils :  
, discovery client is now available  
2024-11-27T18:27:07.562Z INFO 1 --- [ms-peliculas] [main] o.s.b.w.embedded.tomcat.TomcatWebServer :  
ith context path '/'  
2024-11-27T18:27:07.589Z INFO 1 --- [ms-peliculas] [main] com.netec.dk.MsPeliculasApplication :  
.502 seconds (process running for 18.063)  
2024-11-27T18:27:49.744Z INFO 1 --- [ms-peliculas] [nio-9091-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/] :  
t 'dispatcherServlet'  
2024-11-27T18:27:49.745Z INFO 1 --- [ms-peliculas] [nio-9091-exec-1] o.s.web.servlet.DispatcherServlet :  
let'  
2024-11-27T18:27:49.749Z INFO 1 --- [ms-peliculas] [nio-9091-exec-1] o.s.web.servlet.DispatcherServlet :  
Hibernate: select p1_0.id,p1_0.director,p1_0.duracion,p1_0.fecha_estreno,p1_0.genero,p1_0.titulo from peliculas p  
mnadmin@master:~/Intermedio/ws3$ |
```

4.4. Aplicando cambios en K8s y probando en Postman

- ¿Es todo lo que se necesitó para probar el microservicio?



Aplicando cambios | Role & RoleBinding

- El error 403 Forbidden al usar un **ServiceAccount** en Spring Cloud Kubernetes ocurre porque, por defecto, las cuentas de servicio **no tienen permisos** para acceder a los recursos del clúster.
- Kubernetes implementa un modelo de control de acceso basado en **Roles** y **RoleBindings** para gestionar quién puede realizar acciones sobre qué recursos.

Role

- Conjunto de permisos limitado a un namespace.
- Se utiliza para otorgar acceso a recursos específicos dentro de ese espacio de nombres, como Pods, services, o configmaps.

RoleBinding

- Un RoleBinding enlaza un Role con un ServiceAccount, otorgándole los permisos definidos en ese Role.

Aplicando cambios | Role & RoleBinding

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  namespace: dki-curso
  name: services-access
rules:
- apiGroups: []
  resources: ["services"]
  verbs: ["get", "list", "watch"]
```

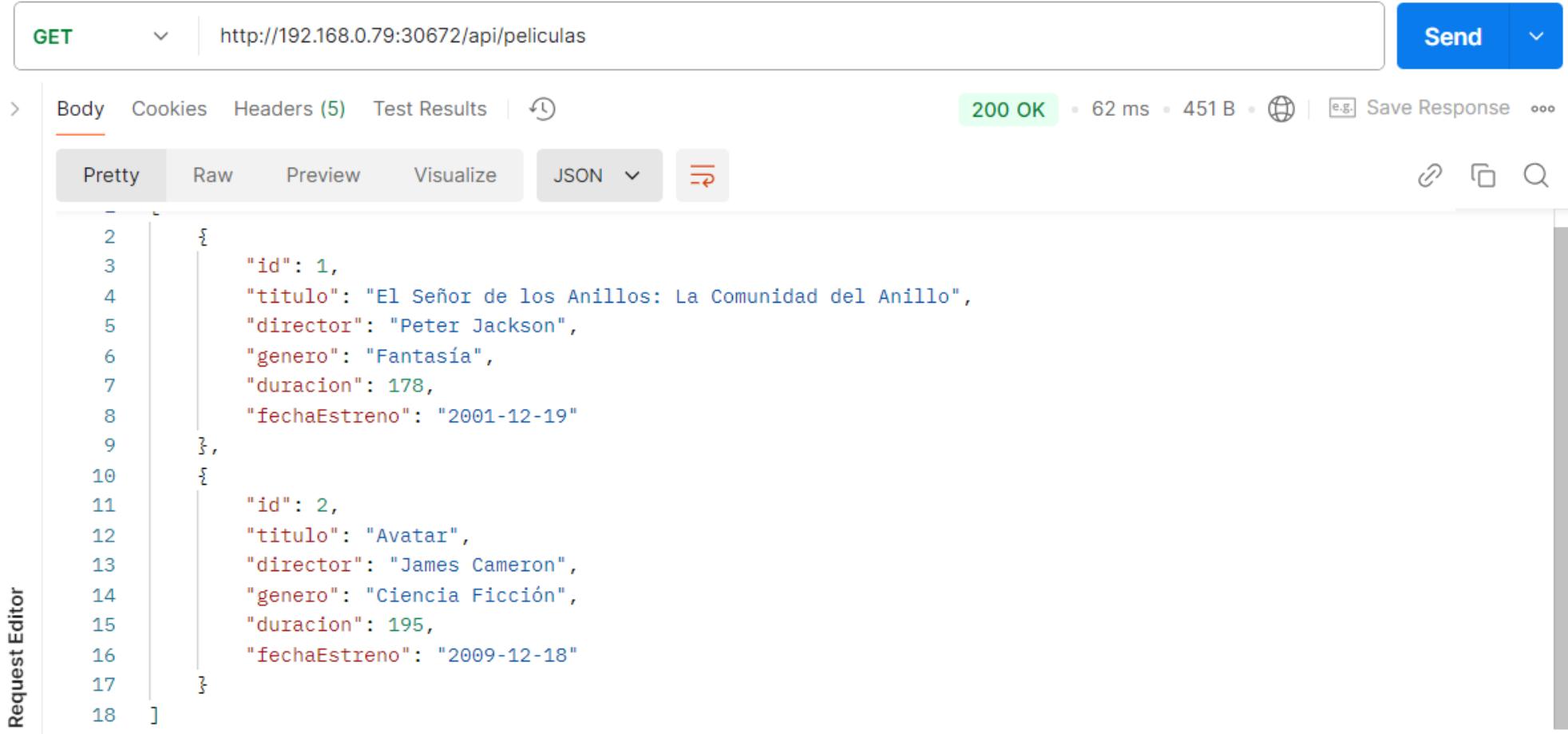
- Este RoleBinding Podría usarse para permitir que una aplicación corriendo en un Pod dentro del namespace dki-curso:
 - Descubra servicios (service discovery) mediante **Spring Cloud Kubernetes**.
 - Liste servicios disponibles en su entorno para configuraciones dinámicas.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: services-access-binding
  namespace: dki-curso
subjects:
- kind: ServiceAccount
  name: default
  namespace: dki-curso
roleRef:
  kind: Role
  name: services-access
  apiGroup: rbac.authorization.k8s.io
```

Aplicando cambios | Role & RoleBinding

```
mnadmin@master:~/Intermedio/ws3$ mnadmin@master:~/Intermedio/ws3$ kubectl get services -n dki-curso
NAME      TYPE      CLUSTER-IP      EXTERNAL-IP      PORT(S)      AGE
ms-peliculas   NodePort    10.111.96.188    <none>        9091:30672/TCP   140m
mnadmin@master:~/Intermedio/ws3$ mnadmin@master:~/Intermedio/ws3$ curl http://10.111.96.188:9091/api/peliculas | jq .
% Total    % Received % xferd  Average Speed   Time     Time      Time  Current
                                         Dload  Upload   Total   Spent    Left  Speed
100  287     0  287     0      0  11038       0 --:--:-- --:--:-- --:--:-- 11038
[
  {
    "id": 1,
    "titulo": "El Señor de los Anillos: La Comunidad del Anillo",
    "director": "Peter Jackson",
    "genero": "Fantasía",
    "duracion": 178,
    "fechaEstreno": "2001-12-19"
  },
  {
    "id": 2,
    "titulo": "Avatar",
    "director": "James Cameron",
    "genero": "Ciencia Ficción",
    "duracion": 195,
    "fechaEstreno": "2009-12-18"
  }
]
mnadmin@master:~/Intermedio/ws3$
```

Aplicando cambios | Postman



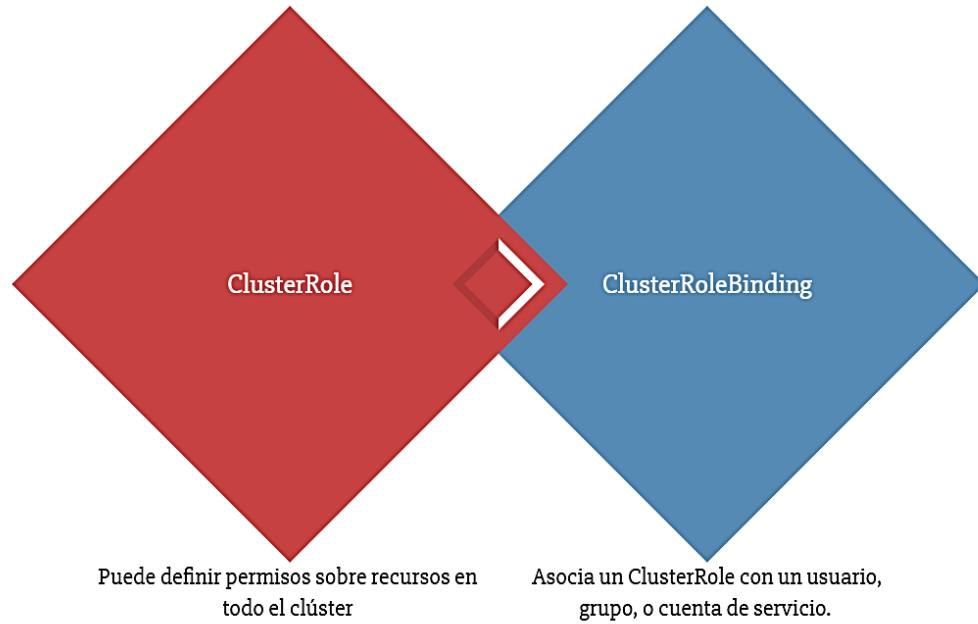
GET <http://192.168.0.79:30672/api/peliculas> Send

Body Cookies Headers (5) Test Results ⏱ 200 OK • 62 ms • 451 B • 🌐 Save Response ⚙️

Pretty Raw Preview Visualize JSON 🔍

```
1  [
2   {
3     "id": 1,
4     "titulo": "El Señor de los Anillos: La Comunidad del Anillo",
5     "director": "Peter Jackson",
6     "genero": "Fantasía",
7     "duracion": 178,
8     "fechaEstreno": "2001-12-19"
9   },
10  {
11    "id": 2,
12    "titulo": "Avatar",
13    "director": "James Cameron",
14    "genero": "Ciencia Ficción",
15    "duracion": 195,
16    "fechaEstreno": "2009-12-18"
17  }
18 ]
```

Aplicando cambios | ClusterRole



Aplicando cambios | ClusterRole

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: access-endpoints-cluster
rules:
  - apiGroups: []
    resources: ["endpoints", "services"]
    verbs: ["get", "list", "watch"]
```

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: bind-default-access-endpoints-cluster
subjects:
  - kind: ServiceAccount
    name: default
    namespace: dki-curso
roleRef:
  kind: ClusterRole
  name: access-endpoints-cluster
  apiGroup: rbac.authorization.k8s.io
```

Aplicando cambios | Revisión

- `kubectl get [roles | rolebindings] -n <namespace>`
- `kubectl describe [role | rolebindings] <role-name> -n <namespace>`
- `kubectl get [clusterroles | clusterrolebindings]`
- `kubectl describe [clusterrole | clusterrolebinding] <clusterrole-name>`
- # Comando para listar los permisos asignados (Roles & ClusterRoles)
- `kubectl auth can-i --list -n <namespace>`
- # Comando para verificar los permisos de un usuario específico.
- `kubectl auth can-i <verb> <resource> --as <user> -n <namespace>`

Aplicando cambios | Revisión

```
mnadmin@master:~/Intermedio/ws3$  
mnadmin@master:~/Intermedio/ws3$  
mnadmin@master:~/Intermedio/ws3$ kubectl get roles -n dki-curso  
NAME          CREATED AT  
endpoints-reader  2024-11-27T17:50:20Z  
services-access   2024-11-27T18:06:47Z  
mnadmin@master:~/Intermedio/ws3$  
mnadmin@master:~/Intermedio/ws3$ kubectl describe role endpoints-reader -n dki-curso  
Name:           endpoints-reader  
Labels:         <none>  
Annotations:    <none>  
PolicyRule:  
  Resources  Non-Resource URLs  Resource Names  Verbs  
  -----  -----  
  endpoints  []                  []              [get list watch]  
mnadmin@master:~/Intermedio/ws3$ kubectl describe role services-access -n dki-curso  
Name:           services-access  
Labels:         <none>  
Annotations:    <none>  
PolicyRule:  
  Resources  Non-Resource URLs  Resource Names  Verbs  
  -----  -----  
  endpoints  []                  []              [get list watch]  
  services   []                  []              [get list watch]  
mnadmin@master:~/Intermedio/ws3$ kubectl auth can-i list endpoints --as=system:serviceaccount:dki-curso:default -n dki-curso  
yes  
mnadmin@master:~/Intermedio/ws3$ |
```

Aplicando cambios | Pods & Services

```
mnadmin@master:~/Intermedio/ws3$  
mnadmin@master:~/Intermedio/ws3$  
mnadmin@master:~/Intermedio/ws3$  
mnadmin@master:~/Intermedio/ws3$ kubectl get pods  
NAME READY STATUS RESTARTS AGE  
ms-peliculas-5795b5667-hlrm5 1/1 Running 2 (14h ago) 37h  
oracle-db-cbf654876-z5kx9 1/1 Running 1 (14h ago) 2d18h  
mnadmin@master:~/Intermedio/ws3$  
mnadmin@master:~/Intermedio/ws3$ kubectl get pods -n dki-curso  
NAME READY STATUS RESTARTS AGE  
ms-favoritos-dc7b94498-nv4b6 1/1 Running 0 70m  
ms-peliculas-5795b5667-nddk6 1/1 Running 2 (14h ago) 22h  
mnadmin@master:~/Intermedio/ws3$  
mnadmin@master:~/Intermedio/ws3$ kubectl get services  
NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE  
kubernetes ClusterIP 10.96.0.1 <none> 443/TCP 37d  
ms-peliculas NodePort 10.103.255.9 <none> 9091:31754/TCP 37h  
oracle-db NodePort 10.109.200.103 <none> 1521:30011/TCP 2d16h  
mnadmin@master:~/Intermedio/ws3$  
mnadmin@master:~/Intermedio/ws3$ kubectl get services -n dki-curso  
NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE  
ms-favoritos NodePort 10.106.209.6 <none> 9094:31572/TCP 70m  
ms-peliculas NodePort 10.111.96.188 <none> 9091:30672/TCP 22h  
mnadmin@master:~/Intermedio/ws3$
```

Aplicando cambios | Postman

The screenshot shows the Postman application interface. In the top navigation bar, there are links for Home, Workspaces, API Network, and various status indicators. The main workspace displays a collection named "Collections" containing an item for "http://192.168.0.79:30672/api/peliculas". A GET request is selected, and its URL is shown in the request field: "http://192.168.0.79:30672/api/peliculas". The response status is "200 OK" with a duration of "13 ms" and a size of "451 B". The response body is displayed in JSON format, showing two movie objects:

```
1 [  
2 {  
3   "id": 1,  
4   "titulo": "El Señor de los Anillos: La Comunidad del Anillo",  
5   "director": "Peter Jackson",  
6   "genero": "Fantasía",  
7   "duracion": 178,  
8   "fechaEstreno": "2001-12-19"  
9 },  
10 {  
11   "id": 2,  
12   "titulo": "Avatar",  
13   "director": "James Cameron",  
14   "genero": "Ciencia Ficción",  
15   "duracion": 195,  
16   "fechaEstreno": "2009-12-18"  
17 }
```

The bottom of the interface includes tabs for Request Editor, Console, and various tools like Postbot, Runner, Vault, and Help.

Aplicando cambios | Postman

The screenshot shows the Postman application interface. At the top, there's a navigation bar with 'Workspaces', 'More', a search icon, a plus sign for creating new environments, a gear icon for settings, a bell icon for notifications, and the Postman logo. To the right of the logo is an 'Upgrade' button. Below the header, there's a toolbar with icons for file operations like 'New', 'Get', 'Post', 'Put', 'Delete', and a 'No environment' dropdown.

The main area displays a POST request to `http://192.168.0.79:31572/favoritos/user2/1`. The status bar indicates a `200 OK` response with `321 ms`, `208 B`, and a globe icon. The response body is shown in JSON format:

```
1 {  
2   "id": 21,  
3   "usuarioId": "user2",  
4   "peliculaId": 1  
5 }
```

The bottom of the interface includes a 'Request Editor' sidebar, a footer with icons for 'Postbot', 'Runner', 'Vault', and help, and a 'Console' tab.

4.5. Ejemplo visualizando LoadBalancer con Metadata de los Pods

- ¿Cómo Podemos configurar un microservicio en Kubernetes para que utilice un *LoadBalancer* y, además, muestre información del Pod que responde a cada solicitud, como el nombre y la IP del Pod?
 - *Esto se logrará utilizando variables de entorno injectadas por Kubernetes, retornando los datos del Pod junto con la respuesta del microservicio.*
 - *Kubernetes distribuye las solicitudes a través de un LoadBalancer y Podemos visualizar información del Pod que responde a cada solicitud.*

Ejemplo | Configuración de variables

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: ms-peliculas
spec:
  replicas: 3
  selector:
    matchLabels:
      app: ms-peliculas
  template:
    metadata:
      labels:
        app: ms-peliculas
    spec:
      containers:
        - name: ms-peliculas
          image: blankiss/ms-peliculas:1.0.5
          ports:
            - containerPort: 8080
  env:
    - name: POD_NAME
      valueFrom:
        fieldRef:
          fieldPath: metadata.name
    - name: POD_IP
      valueFrom:
        fieldRef:
          fieldPath: status.PodIP
# El resto de las variables se omite
```

POD_NAME: Obtiene el nombre del Pod desde la metadata.
POD_ID: Obtiene la IP del Pod desde el estado.

Ejemplo | ms-peliculas

```
import com.netec.dk.entities.Pelicula;
import com.netec.dk.service.PeliculaServiceImpl;

@RestController
@RequestMapping("/api/peliculas")
public class PeliculaController {
    private final PeliculaServiceImpl peliculaService;

    @Autowired
    private Environment environment;

    public PeliculaController(PeliculaServiceImpl peliculaService) {
        this.peliculaService = peliculaService;
    }

    @GetMapping
    public Map<String, Object> listarPelículas() {
        return Map.of(
            "POD_NAME", environment.getProperty("POD_NAME", "Unknown"),
            "POD_ID", environment.getProperty("POD_ID", "Unknown"),
            "películas", peliculaService.listarPelículas());
    }
    // Líneas omitidas
}
```

Ejemplo | Imagen registrada

Ejemplo | Deployment & replicas

```
mnadmin@master:~/Intermedio/ws5$ kubectl apply -f ms-peliculas-deployment.yaml
deployment.apps/ms-peliculas created
mnadmin@master:~/Intermedio/ws5$ kubectl apply -f ms-peliculas-service.yaml
service/ms-peliculas created
mnadmin@master:~/Intermedio/ws5$ kubectl get pods -n dki-curso
NAME                  READY   STATUS    RESTARTS   AGE
ms-favoritos-dc7b94498-nv4b6   1/1     Running   0          3h48m
ms-peliculas-854fd88db6-2s8th  1/1     Running   0          20s
ms-peliculas-854fd88db6-8b96r  1/1     Running   0          20s
ms-peliculas-854fd88db6-zlh5v  1/1     Running   0          20s
mnadmin@master:~/Intermedio/ws5$ kubectl get services -n dki-curso
NAME      TYPE      CLUSTER-IP      EXTERNAL-IP      PORT(S)      AGE
ms-favoritos  NodePort  10.106.209.6  <none>        9094:31572/TCP  3h49m
ms-peliculas  NodePort  10.109.0.50   <none>        9091:31394/TCP  43s
mnadmin@master:~/Intermedio/ws5$
```

Ejemplo | Consumo & verificación

The screenshot shows the Postman application interface. In the top navigation bar, 'Home' and 'Workspaces' are visible. The main area displays an 'HTTP' request to 'http://192.168.0.79:31394/api/peliculas'. The response status is '200 OK' with a duration of '12 ms' and a size of '482 B'. The response body is shown in 'Pretty' JSON format:

```
1 {  
2   "POD_NAME": "ms-peliculas-65497cff74-bfq2t",  
3   "peliculas": [  
4     {  
5       "id": 1,  
6       "titulo": "El Señor de los Anillos: La Comunidad del Anillo",  
7       "director": "Peter Jackson",  
8       "genero": "Fantasía",  
9       "duracion": 178,  
10      "fechaEstreno": "2001-12-19"  
11    },  
12    {  
13      "id": 2,  
14      "titulo": "Avatar",  
15      "director": "James Cameron",  
16      "genero": "Ciencia Ficción",  
17      "duracion": 195,  
18      "fechaEstreno": "2009-12-18"  
19    }  
20  ],  
21  "POD_ID": "10.34.0.11"  
22 }
```

Ejemplo | Consumo & verificación

```
mnadmin@master:~/Intermedio/ws5$  
mnadmin@master:~/Intermedio/ws5$ curl -s http://192.168.0.79:31394/api/peliculas | jq -r '.POD_ID,.POD_NAME'  
10.34.0.8  
ms-peliculas-65497cff74-q9zd4  
mnadmin@master:~/Intermedio/ws5$ curl -s http://192.168.0.79:31394/api/peliculas | jq -r '.POD_ID,.POD_NAME'  
10.34.0.10  
ms-peliculas-65497cff74-rhbrg  
mnadmin@master:~/Intermedio/ws5$ curl -s http://192.168.0.79:31394/api/peliculas | jq -r '.POD_ID,.POD_NAME'  
10.34.0.11  
ms-peliculas-65497cff74-bfq2t  
mnadmin@master:~/Intermedio/ws5$ curl -s http://192.168.0.79:31394/api/peliculas | jq -r '.POD_ID,.POD_NAME'  
10.34.0.10  
ms-peliculas-65497cff74-rhbrg  
mnadmin@master:~/Intermedio/ws5$ curl -s http://192.168.0.79:31394/api/peliculas | jq -r '.POD_ID,.POD_NAME'  
10.34.0.10  
ms-peliculas-65497cff74-rhbrg  
mnadmin@master:~/Intermedio/ws5$ curl -s http://192.168.0.79:31394/api/peliculas | jq -r '.POD_ID,.POD_NAME'  
10.34.0.8  
ms-peliculas-65497cff74-q9zd4  
mnadmin@master:~/Intermedio/ws5$
```

- El balanceo de carga con **Spring Cloud LoadBalancer** y Kubernetes funciona de forma transparente y eficiente.
- Este ejercicio permitió observar cómo las solicitudes se distribuyen entre las réplicas.

4.6. Configuraciones de Spring Boot en ConfigMap

- Recordemos que un **ConfigMap** en Kubernetes es un objeto diseñado para almacenar datos de configuración no confidenciales en un formato **clave:valor**.
- Los **ConfigMaps** permiten desacoplar la configuración de una aplicación del contenedor en la que se ejecuta.
- Los **ConfigMaps** facilitan el manejo y actualización sin necesidad de reconstruir imágenes.

Configuraciones | ConfigMaps

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: app-config
data:
  app_name: "MyApp"
  environment: "production"
  db_url: "jdbc:mysql://db-service:3306/mydb"
```

```
containers:
- name: app
  image: myapp:latest
  envFrom:
    - configMapRef:
        name: app-config
```

Reconfiguración de aplicaciones sin reiniciar los Pods.

- Facilitar cambios en las configuraciones sin interrumpir el servicio.
- ConfigMaps pueden actualizarse y los contenedores reflejarán esos cambios (si están programados para detectar modificaciones).

Configuraciones | pom.xml

```
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-kubernetes-client-config</artifactId>
    <version>3.1.2</version>
</dependency>
```

spring-cloud-starter-kubernetes-config

Configuraciones | Java Code

```
@Autowired  
private Environment environment;  
  
@GetMapping  
public Map<String, Object> listarPeliculas() {  
  
    return Map.of(  
        "POD_NAME", environment.getProperty("POD_NAME", "Unknown"),  
        "POD_ID", environment.getProperty("POD_ID", "Unkown"),  
        "saludo", environment.getProperty("config.saludo", "Unkown"),  
        "peliculas", peliculaService.listarPeliculas());  
}
```

Configuraciones | Properties

```
# Nombre del microservicio
spring.application.name=${SPRING_APPLICATION_NAME:ms-peliculas}

# Puerto del servidor
server.port=${SERVER_PORT:9091}

# Configuracion de la base de datos
spring.datasource.url=${DB_URL:jdbc:oracle:thin:@oracle-db:1521/XEPDB1}
spring.datasource.username=${DB_USERNAME:dkuser}
spring.datasource.password=${DB_PASSWORD:dkpassword}
spring.datasource.driver-class-name=${DB_DRIVER_CLASS_NAME:oracle.jdbc.OracleDriver}

# Configuracion de JPA e Hibernate
spring.jpa.hibernate.ddl-auto=${JPA_DDL_AUTO:update}
spring.jpa.show-sql=${JPA_SHOW_SQL:true}
spring.jpa.properties.hibernate.dialect=${JPA_DIALECT:org.hibernate.dialect.OracleDialect}
```

Configuraciones | Dockerfile

```
# 1. Imagen base
FROM openjdk:21-jdk-slim

# 2. Establecer el directorio de trabajo
WORKDIR /app

# 3. Copiar el archivo JAR generado
COPY target/ms-peliculas-0.0.1-SNAPSHOT.jar app.jar

# 4. Exponer el puerto utilizado por el microservicio
EXPOSE 9091

# 6. Comando de inicio
ENTRYPOINT ["java", "-jar", "app.jar"]
```

Configuraciones | Docker Registry

```
# build la nueva imagen  
docker build -t ms-peliculas:<versión> .  
  
# tag y push al registro  
docker tag ms-peliculas:<version> <dockerhub-usuario>/ms-peliculas:<version>  
docker push <dockerhub-usuario>/ms-peliculas:<version>
```



Configuraciones | Deployment

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: ms-peliculas
  namespace: dki-curso
spec:
  replicas: 2
  selector:
    matchLabels:
      app: ms-peliculas
  template:
    metadata:
      labels:
        app: ms-peliculas
    spec:
      containers:
        - name: ms-peliculas
          image: blankiss/ms-peliculas:1.0.6
          imagePullPolicy: Always
          ports:
            - containerPort: 9091
      volumeMounts:
        - name: config-volume
          mountPath: /config/application.yml
          subPath: application.yml
```

Configuraciones | Deployment

```
volumeMounts:
  - name: config-volume
    mountPath: /config/application.yml
    subPath: application.yml
args:
  - "--spring.config.additional-location=file:/config/application.yml"
env:
  - name: POD_NAME
    valueFrom:
      fieldRef:
        fieldPath: metadata.name
  - name: POD_ID
    valueFrom:
      fieldRef:
        fieldPath: status.podIP
  - name: SPRING_APPLICATION_NAME
    valueFrom:
      configMapKeyRef:
        name: ms-peliculas-config
        key: SPRING_APPLICATION_NAME
  - name: SERVER_PORT
    valueFrom:
      configMapKeyRef:
        name: ms-peliculas-config
        key: SERVER_PORT
```

Configuraciones | Deployment

```
- name: DB_URL
  valueFrom:
    configMapKeyRef:
      name: ms-peliculas-config
      key: DB_URL
- name: DB_DRIVER_CLASS_NAME
  valueFrom:
    configMapKeyRef:
      name: ms-peliculas-config
      key: DB_DRIVER_CLASS_NAME
- name: JPA_DDL_AUTO
  valueFrom:
    configMapKeyRef:
      name: ms-peliculas-config
      key: JPA_DDL_AUTO
- name: JPA_SHOW_SQL
  valueFrom:
    configMapKeyRef:
      name: ms-peliculas-config
      key: JPA_SHOW_SQL
- name: JPA_DIALECT
  valueFrom:
    configMapKeyRef:
      name: ms-peliculas-config
      key: JPA_DIALECT
- name: DB_USERNAME
  valueFrom:
    secretKeyRef:
      name: ms-peliculas-secrets
      key: DB_USERNAME
- name: DB_PASSWORD
  valueFrom:
    secretKeyRef:
      name: ms-peliculas-secrets
      key: DB_PASSWORD
volumes:
- name: config-volume
  configMap:
    name: ms-peliculas-config-demos
    items:
      - key: application.yml
        path: application.yml
```

Configuraciones | Verificación

```
mnadmin@master:~/Intermedio/ws6$ kubectl get all -n dki-curso
NAME                         READY   STATUS    RESTARTS   AGE
pod/ms-favoritos-dc7b94498-nv4b6   1/1     Running   0          4d7h

NAME           TYPE      CLUSTER-IP      EXTERNAL-IP      PORT(S)      AGE
service/ms-favoritos   NodePort   10.106.209.6   <none>        9094:31572/TCP   4d7h

NAME           READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/ms-favoritos   1/1     1           1          4d7h

NAME           DESIRED  CURRENT  READY   AGE
replicaset.apps/ms-favoritos-dc7b94498   1        1        1        4d7h
mnadmin@master:~/Intermedio/ws6$ |
```

```
mnadmin@master:~/Intermedio/ws6$ kubectl get all -n dki-curso
NAME                         READY   STATUS    RESTARTS   AGE
pod/ms-favoritos-dc7b94498-nv4b6   1/1     Running   0          4d10h
pod/ms-peliculas-54b4f5cfb6-5dqfq   1/1     Running   0          29m
pod/ms-peliculas-54b4f5cfb6-kp518   1/1     Running   0          29m

NAME           TYPE      CLUSTER-IP      EXTERNAL-IP      PORT(S)      AGE
service/ms-favoritos   NodePort   10.106.209.6   <none>        9094:31572/TCP   4d10h
service/ms-peliculas   NodePort   10.96.61.194   <none>        9091:31737/TCP   51m

NAME           READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/ms-favoritos   1/1     1           1          4d10h
deployment.apps/ms-peliculas   2/2     2           2          29m

NAME           DESIRED  CURRENT  READY   AGE
replicaset.apps/ms-favoritos-dc7b94498   1        1        1        4d10h
replicaset.apps/ms-peliculas-54b4f5cfb6   2        2        2        29m
mnadmin@master:~/Intermedio/ws6$
```

Configuraciones | Verificación

```
mnadmin@master:~/Intermedio/ws6$ mnadmin@master:~/Intermedio/ws6$ cat configmap-ms-peliculas.yaml

apiVersion: v1
kind: configMap
metadata:
  name: ms-peliculas-config-demos
  namespace: dki-curso
data:
  application.yml: |-  
    config:  
      saludo: "Bienvenido al microservicio de peliculas"  
  
mnadmin@master:~/Intermedio/ws6$ curl -s http://10.96.61.194:9091/api/peliculas | jq .  
{  
  "POD_ID": "10.34.0.0",  
  "saludo": "Bienvenido al microservicio de peliculas", ←  
  "peliculas": [  
    {  
      "id": 1,  
      "titulo": "El Señor de los Anillos: La Comunidad del Anillo",  
      "director": "Peter Jackson",  
      "genero": "Fantasia",  
      "duracion": 178,  
      "fechaEstreno": "2001-12-19"  
    },  
    {  
      "id": 2,  
      "titulo": "Avatar",  
      "director": "James Cameron",  
      "genero": "Ciencia Ficción",  
      "duracion": 195,  
      "fechaEstreno": "2009-12-18"  
    }  
  ],  
  "POD_NAME": "ms-peliculas-54b4f5cfb6-bt9bk"  
}  
mnadmin@master:~/Intermedio/ws6$ |
```

Configuraciones | Verificación

```
mnadmin@master:~/Intermedio/ws6$  
mnadmin@master:~/Intermedio/ws6$ cat configmap-ms-peliculas.yaml  
  
apiVersion: v1  
kind: ConfigMap  
metadata:  
  name: ms-peliculas-config-demos  
  namespace: dki-curso  
data:  
  application.yml: |-  
    config:  
      saludo: "Bienvenido al microservicio de películas MODIFICADO"  
  
mnadmin@master:~/Intermedio/ws6$  
mnadmin@master:~/Intermedio/ws6$ kubectl get services -n dki-curso  
NAME      TYPE      CLUSTER-IP   EXTERNAL-IP   PORT(S)      AGE  
ms-favoritos   NodePort   10.106.209.6 <none>        9094:31572/TCP   4d10h  
ms-peliculas   NodePort   10.96.61.194  <none>        9091:31737/TCP   22m  
mnadmin@master:~/Intermedio/ws6$  
mnadmin@master:~/Intermedio/ws6$ curl -s http://10.96.61.194:9091/api/peliculas | jq .  
{  
  "POD_ID": "10.34.0.8",  
  "POD_NAME": "ms-peliculas-54b4f5cfb6-5dqfq",  
  "peliculas": [  
    {  
      "id": 1,  
      "titulo": "El Señor de los Anillos: La Comunidad del Anillo",  
      "director": "Peter Jackson",  
      "genero": "Fantasía",  
      "duracion": 178,  
      "fechaEstreno": "2001-12-19"  
    },  
    {  
      "id": 2,  
      "titulo": "Avatar",  
      "director": "James Cameron",  
      "genero": "Ciencia Ficción",  
      "duracion": 195,  
      "fechaEstreno": "2009-12-18"  
    }  
  ],  
  "saludo": "Bienvenido al microservicio de películas MODIFICADO"  
}  
mnadmin@master:~/Intermedio/ws6$
```

Configuraciones | Beneficios

- **Centralización**
 - Facilita la gestión de configuraciones en entornos complejos.
- **Flexibilidad**
 - Permite cambiar configuraciones sin reiniciar los Pods.
- **Escalabilidad**
 - Soporta múltiples entornos con mínima intervención manual.

4.7. Configuraciones de entornos dev y prod

- La configuración de perfiles es una técnica sencilla y efectiva para gestionar diferentes entornos en aplicaciones desplegadas en Kubernetes.
- La integración con Spring Cloud y Kubernetes permite manejar configuraciones centralizadas y específicas por entorno de forma dinámica.

- Facilita la gestión de configuraciones específicas para entornos.
- Evita la duplicación de configuraciones globales.
- Promueve el desacoplamiento entre entornos.

Configuraciones | Perfiles

- Los perfiles en Spring permiten segregar configuraciones y beans según el entorno en el que se ejecuta la aplicación, como desarrollo, pruebas o producción.
- Al establecer **spring.profiles.active**, indicas cuál perfil (o perfiles) debe ser utilizado al ejecutar la aplicación.
- application.properties o application.yml
 - `spring.profiles.active=dev`
- Línea de comandos al ejecutar la aplicación.
 - `java -jar myapp.jar --spring.profilesactive=prod`
- Variables de entorno.
 - `export SPRING_PROFILES_ACTIVE=dev`

Configuraciones | Guía Express

1. Preparación de Imagen.
2. Crear ConfigMaps por entorno.
 - Opcionalmente define Secrets por entorno para almacenar información sensible.
3. Crear Deployments por entorno.
 - Define Deployments independientes para dev y prod.
 - Puedes aplicar etiquetas para diferenciar los entornos.
4. Crear Servicios por entorno
 - Define servicios específicos para cada entorno.
5. Desplegar en el espacio de nombres.
6. Verificar el despliegue.

Configuraciones | Imagen

```
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-kubernetes-client-config</artifactId>
    <version>3.1.2</version>
</dependency>
```

```
# Spring Cloud Kubernetes
spring.cloud.kubernetes.secrets.enable-api=true
spring.cloud.kubernetes.discovery.all-namespaces=true

# Perfiles
spring.profiles.active=dev
```

```
@Autowired
private Environment environment;

@GetMapping
public Map<String, Object> listarPeliculas() {

    return Map.of(
        "POD_NAME", environment.getProperty("POD_NAME", "Unknown"),
        "POD_ID", environment.getProperty("POD_ID", "Unknown"),
        "saludo", environment.getProperty("config.saludo", "Unknown"),
        "peliculas", peliculaService.listarPeliculas());
}
```

Configuraciones | ConfigMaps



```
mnadmin@master:~/Intermedio/ws10$  
mnadmin@master:~/Intermedio/ws10$ kubectl get configmaps -n dki-curso  
NAME          DATA   AGE  
configmap-dev 3      9h  
configmap-prod 3      9h  
kube-root-ca.crt 1     44h  
ms-favoritos-config 8     38h  
ms-peliculas-config 7     44h  
mnadmin@master:~/Intermedio/ws10$ kubectl describe configmap configmap-dev -n dki-curso  
Name:         configmap-dev  
Namespace:    dki-curso  
Labels:       <none>  
Annotations:  <none>  
  
Data  
=====  
config.database.url:  
----  
jdbc:mysql://dev-db:3306/mydb  
config.saludo:  
----  
Bienvenido a Ambiente de Desarrollo Curso Docker y Kubernetes II  
spring.profiles.active:  
----  
dev  
  
BinaryData  
=====  
  
Events:  <none>  
mnadmin@master:~/Intermedio/ws10$
```

Configuraciones | ConfigMaps

```
mnadmin@master:~/Intermedio/ws10$ kubectl describe configmap configmap-prod -n dki-curso
Name:           configmap-prod
Namespace:      dki-curso
Labels:         <none>
Annotations:   <none>

Data
=====
config.database.url:
-----
jdbc:mysql://dev-db:3306/mydb
config.saludo:
-----
Bienvenido a Ambiente Productivo DK II
spring.profiles.active:
-----
prod

BinaryData
=====

Events:  <none>
mnadmin@master:~/Intermedio/ws10$ |
```

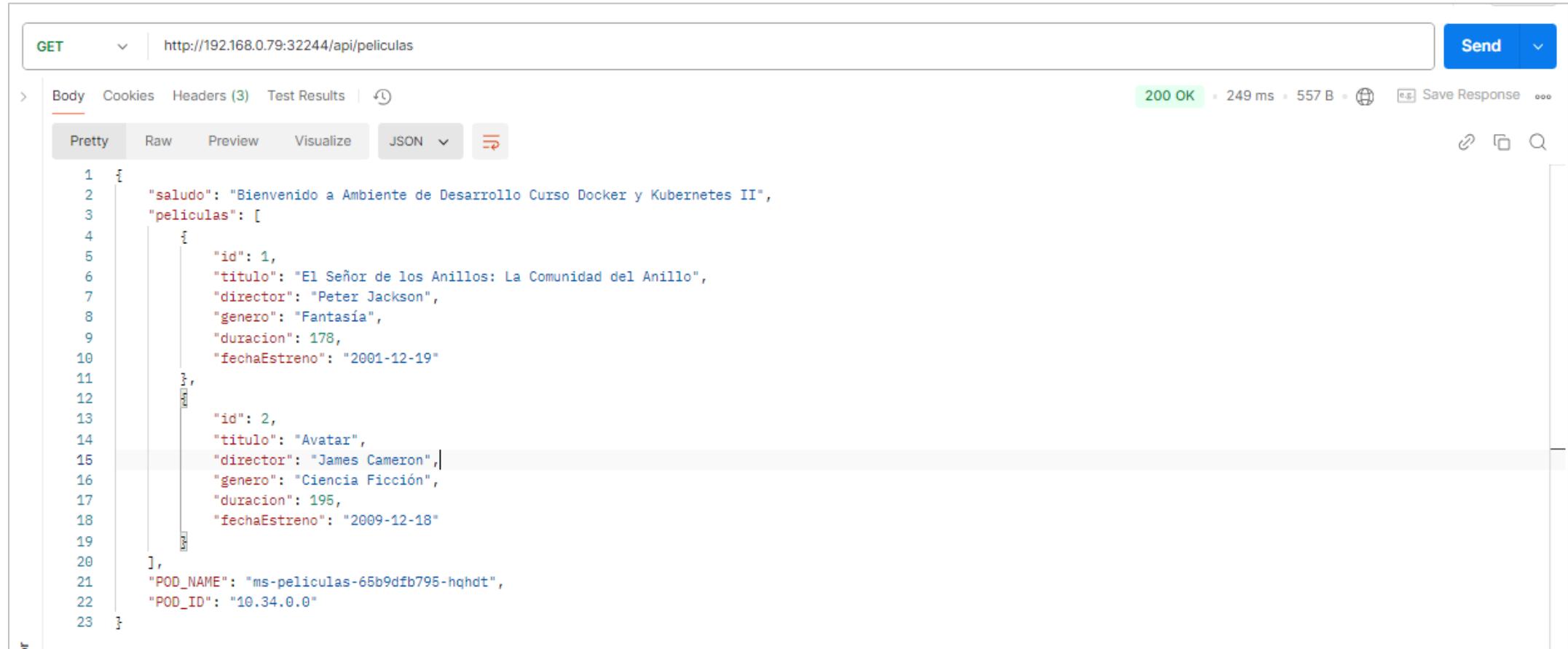
Configuraciones | ConfigMaps

```
mnadmin@master:~/Intermedio/ws10$  
mnadmin@master:~/Intermedio/ws10$ cat configmap.yaml  
  
apiVersion: v1  
data:  
  spring.profiles.active: dev  
  config.saludo: Bienvenido a Ambiente de Desarrollo Curso Docker y Kubernetes II  
  config.database.url: jdbc:mysql://dev-db:3306/mydb  
kind: ConfigMap  
metadata:  
  name: configmap-dev  
  namespace: dki-curso  
  
mnadmin@master:~/Intermedio/ws10$ cat configmap2.yaml  
  
apiVersion: v1  
data:  
  spring.profiles.active: prod  
  config.saludo: Bienvenido a Ambiente Productivo DK II  
  config.database.url: jdbc:mysql://dev-db:3306/mydb  
kind: ConfigMap  
metadata:  
  name: configmap-prod  
  namespace: dki-curso  
  
mnadmin@master:~/Intermedio/ws10$ |
```

Configuraciones | Deployments & servicios

```
mnadmin@master:~/Intermedio/ws10$  
mnadmin@master:~/Intermedio/ws10$  
mnadmin@master:~/Intermedio/ws10$ kubectl get deployments -n dki-curso -o wide  
NAME        READY   UP-TO-DATE   AVAILABLE   AGE      CONTAINERS   IMAGES          SELECTOR  
ms-favoritos  1/1     1           1           22h     ms-favoritos  blankiss/ms-favoritos:1.0.8  app=ms-favoritos  
ms-peliculas  2/2     2           2           9h      ms-peliculas  blankiss/ms-peliculas:1.0.10  app=ms-peliculas  
mnadmin@master:~/Intermedio/ws10$  
mnadmin@master:~/Intermedio/ws10$ kubectl get pods -n dki-curso -o wide  
NAME          READY   STATUS    RESTARTS   AGE      IP           NODE   NOMINATED-NODE   READINESS   GAT  
ES  
ms-favoritos-dc7b94498-nv4b6  1/1     Running   0          22h    10.34.0.2   worker4  <none>       <none>  
ms-peliculas-59bfcd9ffc-26cvl 1/1     Running   0          9h     10.34.0.8   worker4  <none>       <none>  
ms-peliculas-59bfcd9ffc-d45nr  1/1     Running   0          9h     10.34.0.0   worker4  <none>       <none>  
mnadmin@master:~/Intermedio/ws10$  
mnadmin@master:~/Intermedio/ws10$ kubectl describe deployments -n dki-curso | grep -i active  
    SPRING_PROFILES_ACTIVE:  <set to the key 'spring.profiles.active' of config map 'configmap-dev'>          Optiona  
l: false  
mnadmin@master:~/Intermedio/ws10$ kubectl get services -n dki-curso  
NAME      TYPE      CLUSTER-IP   EXTERNAL-IP   PORT(S)      AGE  
ms-favoritos  NodePort  10.106.209.6  <none>        9094:31572/TCP  22h  
ms-peliculas  NodePort  10.102.152.2  <none>        9091:32244/TCP  9h  
mnadmin@master:~/Intermedio/ws10$
```

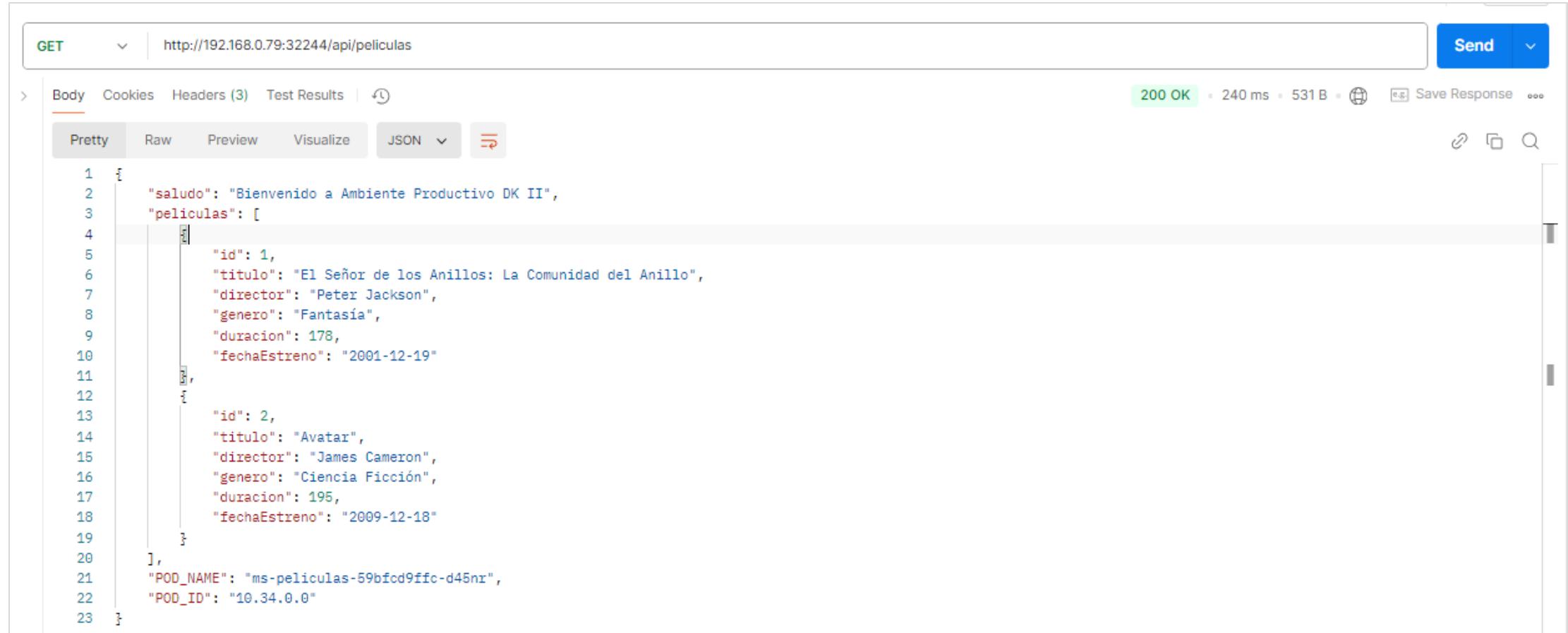
Configuraciones de entornos dev y prod



A screenshot of a Postman API request interface. The request method is GET, and the URL is `http://192.168.0.79:32244/api/peliculas`. The response status is 200 OK, with a duration of 249 ms and a size of 557 B. The response body is displayed in Pretty JSON format:

```
1 {  
2   "saludo": "Bienvenido a Ambiente de Desarrollo Curso Docker y Kubernetes II",  
3   "peliculas": [  
4     {  
5       "id": 1,  
6       "titulo": "El Señor de los Anillos: La Comunidad del Anillo",  
7       "director": "Peter Jackson",  
8       "genero": "Fantasía",  
9       "duracion": 178,  
10      "fechaEstreno": "2001-12-19"  
11    },  
12    {  
13      "id": 2,  
14      "titulo": "Avatar",  
15      "director": "James Cameron",  
16      "genero": "Ciencia Ficción",  
17      "duracion": 195,  
18      "fechaEstreno": "2009-12-18"  
19    }  
20  ],  
21  "POD_NAME": "ms-peliculas-65b9dfb795-hqhdt",  
22  "POD_ID": "10.34.0.0"  
23 }
```

Configuraciones de entornos dev y prod



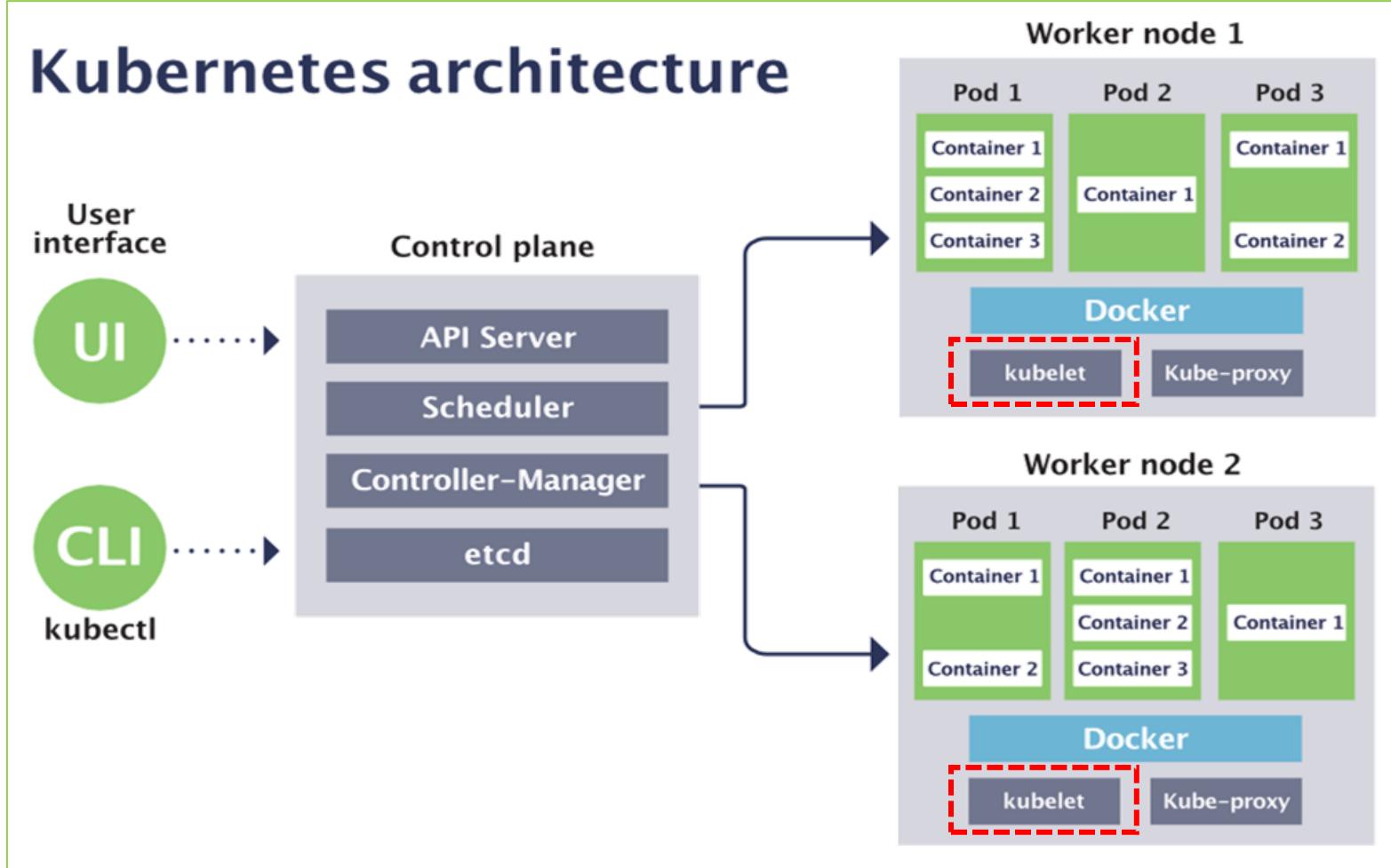
A screenshot of a Postman API request interface. The request method is GET, and the URL is `http://192.168.0.79:32244/api/peliculas`. The response status is 200 OK, with a duration of 240 ms and a size of 531 B. The response body is displayed in Pretty JSON format, showing a welcome message and a list of movies.

```
1 {  
2   "saludo": "Bienvenido a Ambiente Productivo DK II",  
3   "peliculas": [  
4     {  
5       "id": 1,  
6       "titulo": "El Señor de los Anillos: La Comunidad del Anillo",  
7       "director": "Peter Jackson",  
8       "genero": "Fantasía",  
9       "duracion": 178,  
10      "fechaEstreno": "2001-12-19"  
11    },  
12    {  
13      "id": 2,  
14      "titulo": "Avatar",  
15      "director": "James Cameron",  
16      "genero": "Ciencia Ficción",  
17      "duracion": 195,  
18      "fechaEstreno": "2009-12-18"  
19    }  
20  ],  
21  "POD_NAME": "ms-peliculas-59bfcd9ffc-d45nr",  
22  "POD_ID": "10.34.0.0"  
23 }
```

4.8. Configurando Liveness y Readiness en microservicios con Spring

1. ¿Cuál es el componente de Kubernetes responsable de comunicarse con el nodo maestro, ejecutar contenedores y garantizar que los Pods estén funcionando en cada nodo?
2. ¿Cuál es el tipo de verificación en Kubernetes que verifica si una aplicación dentro de un contenedor sigue funcionando correctamente y, en caso contrario reinicia el contenedor?
3. ¿Cuál es el tipo de verificación en Kubernetes que determina si una aplicación dentro de un contenedor está lista para recibir tráfico y solicitudes?

Configurando | kubelet



Configurando Liveness y Readiness

Característica	Liveness Probe	Readiness Probe	
Propósito	Detectar contenedores funcionales o bloqueados	no	Determinar si un contenedor puede recibir tráfico
Acción cuando falla	Kubernetes reinicia el contenedor.	el	Kubernetes remueve el Pod de los endpoints del servicio.
Uso típico	Evitar que un contenedor inestable consuma recursos		Administrar la disponibilidad de aplicaciones

Configurando | Spring Boot Actuator

Spring Boot Actuator

- Es un módulo ampliamente utilizado que proporciona una serie de endpoints y funcionalidades para monitorear, administrar y diagnosticar aplicaciones Spring Boot en tiempo de ejecución.
- Permite obtener métricas, información del estado de la aplicación, y realizar operaciones administrativas sin necesidad de construir herramientas adicionales.
 - Funcionalidades clave:
 - **Monitoreo**
 - Ofrece métricas de sistema, uso de memoria, CPU, y otros recursos.
 - **Información de diagnóstico**
 - Facilita la depuración y compresión del comportamiento de la aplicación.
 - **Endpoints listos para usar.**
 - Endpoints predefinidos que ofrecen información útil para monitoreo y administración.

Configurando | Endpoints

Endpoint	Descripción
/actuator/health	Proporciona información sobre el estado de la aplicación.
/actuator/info	Muestra información personalizada configurada en el archivo application.properties o yml.
/actuator/metrics	Ofrece métricas detalladas, como uso de CPU, memoria, hilos activos, entre otros.
/actuator/env	Expone las propiedades del entorno y variables configuradas.
/actuator/beans	Lista los beans y sus dependencias en el contenedor Spring.
/actuator/mappings	Muestra todas las rutas configuradas en la aplicación.
/actuator/threaddump	Genera un volcado del estado de los hilos en ejecución.
/actuator/loggers	Permite ver y cambiar los niveles de los logs en tiempo de ejecución.

<https://docs.spring.io/spring-boot/reference/actuator/endpoints.html>

Configurando | Dependencias

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-actuator</artifactId>
</dependency>
</dependencies>
```

<https://docs.spring.io/spring-boot/reference/actuator/index.html#actuator>

Configurando | ApplicationContext

```
// Simulación de un error en la aplicación

@Autowired
private ApplicationContext context;

@GetMapping("/error-controlado")
public void errorControlado() {
    ((ConfigurableApplicationContext) context).close();
}
```

ApplicationContext: Es la interfaz central de Spring para gestionar beans, su ciclo de vida y dependencias, además de soportar eventos y carga de recursos.

Configurando | Properties

```
# Spring Boot Actuator - application.properties - ms-peliculas
management.endpoints.web.exposure.include=*
management.endpoint.health.show-details=always
management.endpoint.health.probes.enabled=true
management.health.livenessstate.enabled=true
management.health.readinessstate.enabled=true
```

- Estas propiedades configuradas en un microservicio habilitan y exponen detalles importantes para monitoreo y diagnóstico de su estado de salud y disponibilidad.
- Estas configuraciones permiten integrar el microservicio con herramientas de orquestación como Kubernetes, que utiliza estos endpoints para gestionar la disponibilidad y el estado de las aplicaciones.

Configurando | Docker Registry

```
c:\ Símbolo del sistema - .\mvnw clean install -Dmaven.test.skip=true - .\mvnw clean install -Dmaven.test.skip=true - .\mvnw clean install -Dmave.test.s...
=> [3/3] COPY target/ms-peliculas-0.0.1-SNAPSHOT.jar app.jar 1.7s
=> exporting to image 15.2s
=> => exporting layers 11.7s
=> => exporting manifest sha256:c810e1e0f73d8f02945330e75f5f92a52afe2e2aab69672c215f2f991344480c 0.1s
=> => exporting config sha256:5e241f41317d574f04023091b144df53807701ec8de103fc5d35ac2e3ec8ae9f 0.0s
=> => exporting attestation manifest sha256:6dd6205beba739b35c534bcc2dc6b7c7c405123dab9dafcc11eb84e3c92cca37 0.2s
=> => exporting manifest list sha256:51184d92a4700e0cb087260a2df025ff9f78a1408964fc9ae26f81b4f548ef81 0.1s
=> => naming to docker.io/library/ms-peliculas:1.0.11 0.0s
=> => unpacking to docker.io/library/ms-peliculas:1.0.11 3.0s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/gapigv6pcibpbhxzc1in70a08

What's next:
    View a summary of image vulnerabilities and recommendations → docker scout quickview

C:\00_Material_Docker_Kubernetes\ws2\ms-peliculas>docker tag ms-peliculas:1.0.11 blankiss/ms-peliculas:1.0.11
C:\00_Material_Docker_Kubernetes\ws2\ms-peliculas>docker push blankiss/ms-peliculas:1.0.11
The push refers to repository [docker.io/blankiss/ms-peliculas]
a6013f34d8f6: Pushed
51a30295be38: Pushed
af800cd8441e: Layer already exists
9b2d3c368f4f: Layer already exists
a803e7c4b030: Layer already exists
b4972576c83d: Layer already exists
1.0.11: digest: sha256:51184d92a4700e0cb087260a2df025ff9f78a1408964fc9ae26f81b4f548ef81 size: 856
C:\00_Material_Docker_Kubernetes\ws2\ms-peliculas>
```

4.9. Configurando Pod con Liveness y Readiness Probe

- ¿En qué objeto de Kubernetes se configuran las comprobaciones de **Liveness** y **Readiness** para monitorear la salud y disponibilidad de los contenedores dentro de un clúster?

Pod o Deployment

Configurando Pod | ¿Deployment?

- Aunque se definen a nivel de Pod, las comprobaciones Liveness y Readiness, son aplicables en Deployments.
 - Deployment administra la creación y actualización de Pods.
- Los Pods generados por un Deployment heredan las configuraciones definidas, incluyendo estas comprobaciones.
- Se configuran dentro del archivo/manifiesto YAML de la especificación del Pod o Deployment.
 - Específicamente en la sección del contenedor (container).
 - Si estás configurando un Deployment, lo haces dentro de la plantilla del Pod `spec.template.spec`, en la definición del contenedor.



Configurando Pod | Caso de estudio

readinessProbe:

```
httpGet:  
  path: /actuator/health/readiness  
  port: 9091  
  scheme: HTTP  
initialDelaySeconds: 5  
periodSeconds: 20  
timeoutSeconds: 10
```

livenessProbe:

```
httpGet:  
  path: /actuator/health/liveness  
  port: 9091  
  scheme: HTTP  
initialDelaySeconds: 10  
periodSeconds: 30  
timeoutSeconds: 10
```

HTTP/GE
T

Esta configuración asegura que Kubernetes supervise constantemente el estado del microservicio ms-peliculas:

- **Readiness Probe:** Verifica si el contenedor está listo para recibir solicitudes.
- **Liveness Probe:** Comprueba si el contenedor sigue funcionando correctamente.

Configurando Pod | Caso de estudio

```
mnadmin@master:~/Intermedio/ws11$  
mnadmin@master:~/Intermedio/ws11$ kubectl get all -n dki-curso  
NAME                                READY   STATUS    RESTARTS   AGE  
pod/ms-favoritos-dc7b94498-nv4b6   1/1     Running   0          2d  
  
NAME           TYPE      CLUSTER-IP      EXTERNAL-IP      PORT(S)      AGE  
service/ms-favoritos   NodePort   10.106.209.6   <none>        9094:31572/TCP   2d  
  
NAME           READY   UP-TO-DATE   AVAILABLE   AGE  
deployment.apps/ms-favoritos   1/1     1           1           2d  
  
NAME           DESIRED   CURRENT   READY   AGE  
replicaset.apps/ms-favoritos-dc7b94498   1         1         1         2d  
mnadmin@master:~/Intermedio/ws11$  
mnadmin@master:~/Intermedio/ws11$ ls -l  
total 8  
-rw-rw-r-- 1 mnadmin mnadmin 3196 Nov 29 18:24 ms-peliculas-deployment.yaml  
-rw-rw-r-- 1 mnadmin mnadmin 211 Nov 29 12:44 ms-peliculas-service.yaml  
mnadmin@master:~/Intermedio/ws11$  
mnadmin@master:~/Intermedio/ws11$ kubectl apply -f ms-peliculas-deployment.yaml  
deployment.apps/ms-peliculas created  
mnadmin@master:~/Intermedio/ws11$ kubectl apply -f ms-peliculas-service.yaml  
service/ms-peliculas created  
mnadmin@master:~/Intermedio/ws11$ kubectl get pods -n dki-curso  
NAME                                READY   STATUS    RESTARTS   AGE  
ms-favoritos-dc7b94498-nv4b6   1/1     Running   0          2d  
ms-peliculas-5fb9ff7844-79hmq   0/1     Running   0          20s  
ms-peliculas-5fb9ff7844-wf95w   0/1     Running   0          20s  
mnadmin@master:~/Intermedio/ws11$
```

Configurando Pod | Caso de estudio

```
mnadmin@master:~/Intermedio/ws11$  
mnadmin@master:~/Intermedio/ws11$ kubectl get roles -n dki-curso  
NAME          CREATED AT  
endpoints-reader 2024-11-27T17:50:20Z  
pods-access    2024-11-30T16:20:06Z  
services-access 2024-11-27T18:06:47Z  
mnadmin@master:~/Intermedio/ws11$  
mnadmin@master:~/Intermedio/ws11$ kubectl describe serviceaccount default -n dki-curso  
Name:           default  
Namespace:      dki-curso  
Labels:         <none>  
Annotations:    <none>  
Image pull secrets: <none>  
Mountable secrets: <none>  
Tokens:         <none>  
Events:         <none>  
mnadmin@master:~/Intermedio/ws11$ kubectl auth can-i get pods --as=system:serviceaccount:dki-curso:default -n dki-curso  
yes  
mnadmin@master:~/Intermedio/ws11$  
mnadmin@master:~/Intermedio/ws11$
```

Configurando Pod | Caso de estudio

```
mnadmin@master:~/Intermedio/ws11$ cat ms-roles.yaml

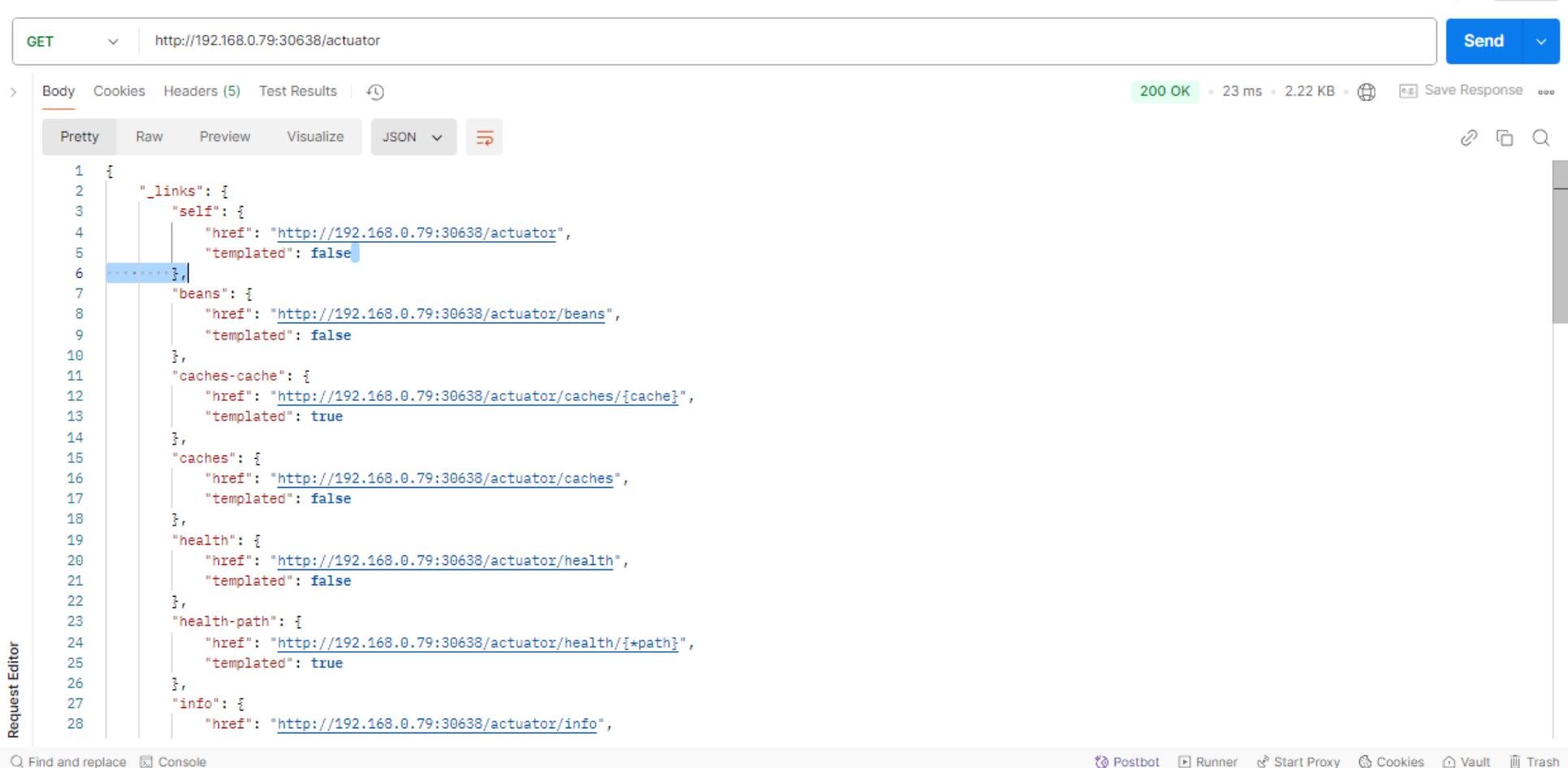
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  namespace: dki-curso
  name: pods-access
rules:
  - apiGroups: []
    resources: ["pods"]
    verbs: ["get", "list", "watch"]

---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  namespace: dki-curso
  name: pods-access-binding
subjects:
  - kind: ServiceAccount
    name: default
    namespace: dki-curso
roleRef:
  kind: Role
  name: pods-access
  apiGroup: rbac.authorization.k8s.io

mnadmin@master:~/Intermedio/ws11$
```

Al crear este rol (Pods-access) y asociarlo con el ServiceAccount (default), permitiste que Spring Boot Actuator acceda a los Pods y funciones correctamente en el entorno de Kubernetes.

Configurando Pod | Caso de estudio



The screenshot shows the Postman application interface. At the top, there is a header bar with 'GET' selected, the URL 'http://192.168.0.79:30638/actuator', and a 'Send' button. Below the header, there are tabs for 'Body', 'Cookies', 'Headers (5)', 'Test Results', and a refresh icon. To the right of the URL, it says '200 OK' with a response time of '23 ms' and a size of '2.22 KB'. There are also 'Save Response' and 'Copy' buttons.

The main area displays a JSON response with line numbers from 1 to 28. The response structure includes '_links', 'beans', 'caches-cache', 'caches', 'health', 'health-path', and 'info' sections, each containing href and templated fields. The JSON is displayed in 'Pretty' format, which includes line breaks and indentation.

```
1 {  
2   "_links": {  
3     "self": {  
4       "href": "http://192.168.0.79:30638/actuator",  
5       "templated": false  
6     }  
7   },  
8   "beans": {  
9     "href": "http://192.168.0.79:30638/actuator/beans",  
10    "templated": false  
11  },  
12  "caches-cache": {  
13    "href": "http://192.168.0.79:30638/actuator/caches/{cache}",  
14    "templated": true  
15  },  
16  "caches": {  
17    "href": "http://192.168.0.79:30638/actuator/caches",  
18    "templated": false  
19  },  
20  "health": {  
21    "href": "http://192.168.0.79:30638/actuator/health",  
22    "templated": false  
23  },  
24  "health-path": {  
25    "href": "http://192.168.0.79:30638/actuator/health/{*path}",  
26    "templated": true  
27  },  
28  "info": {  
29    "href": "http://192.168.0.79:30638/actuator/info",  
30  }
```

At the bottom of the interface, there are navigation icons for 'Find and replace', 'Console', 'Postbot', 'Runner', 'Start Proxy', 'Cookies', 'Vault', and 'Trash'.

Configurando Pod | Caso de estudio

```
mnadmin@master:~/Intermedio/ws11$  
mnadmin@master:~/Intermedio/ws11$  
mnadmin@master:~/Intermedio/ws11$ curl -s http://192.168.0.79:30638/actuator/health | jq .  
{  
    "status": "UP",  
    "components": {  
        "db": {  
            "status": "UP",  
            "details": {  
                "database": "Oracle",  
                "validationQuery": "isValid()"  
            }  
        },  
        "discoveryComposite": {  
            "status": "UP",  
            "components": {  
                "discoveryClient": {  
                    "status": "UP",  
                    "details": {  
                        "services": [  
                            "kube-dns",  
                            "oracle-db",  
                            "ms-peliculas",  
                            "ingress-nginx-controller-admission",  
                            "nginx-service",  
                            "metrics-server",  
                            "kubernetes",  
                            "ms-favoritos",  
                            "ingress-nginx-controller"  
                        ]  
                    }  
                }  
            }  
        }  
    }  
}
```

Configurando Pod | Caso de estudio

Configurando Pod | Caso de estudio

```
mnadmin@master:~/Intermedio/ws11$ curl -s http://192.168.0.79:30638/actuator/health/liveness | jq .
{
  "status": "UP"
}
mnadmin@master:~/Intermedio/ws11$ curl -s http://192.168.0.79:30638/actuator/health/readiness | jq .
{
  "status": "UP"
}
mnadmin@master:~/Intermedio/ws11$ curl -s http://192.168.0.79:30638/actuator/env | jq .
{
  "activeProfiles": [
    "kubernetes",
    "dev"
  ],
  "defaultProfiles": [
    "default"
  ],
  "propertySources": [
    {
      "name": "server.ports",
      "properties": {
        "local.server.port": {
          "value": "*****"
        }
      }
    },
    {
      "name": "servletContextInitParams",
      "properties": {}
    },
    {
      "name": "application",
      "properties": {}
    }
  ]
}
```

Configurando Pod | Caso de estudio

- La combinación de **Spring Boot Actuator** y **Kubernetes Probes** proporciona un mecanismo robusto.
 - Garantiza que los microservicios estén no solo operativos, sino también en condiciones óptimas para manejar tráfico en entornos de producción.
- **Actuator** expone endpoints como **/actuator/health**, que reportan el estado general del servicio, y métricas detalladas sobre componentes críticos.
 - Como la base de datos, conexiones, memoria y más.
- Por otro lado, Kubernetes Probes (liveness y readiness) utilizan estos endpoints para monitorear continuamente la salud y disponibilidad del microservicio.

- Alta disponibilidad
- Escalabilidad
- Integridad

4.10. Configurando Pod Container Resources

- En un entorno donde múltiples aplicaciones comparten los recursos de un clúster Kubernetes, ¿cómo garantizarías que un microservicio tenga suficientes recursos de CPU y memoria para funcionar correctamente, sin afectar a otros servicios del clúster?
- ¿Qué estrategias implementarías para evitar la sobreutilización o subutilización de recursos?"

Configurando | request & limits

request	limits
Define los recursos mínimos garantizados para un contenedor. Si se establece una solicitud de recursos, Kubernetes asegura que esa cantidad de recursos estará disponible para el contenedor en todo momento.	Define los recursos máximos que un contenedor puede utilizar. Si un contenedor intenta usar más recursos de los que ha especificado como límite Kubernetes lo limitará o podría incluso reiniciarlo si excede los límites.
requests.cpu: "2" requests.Memory: "4Gi"	limits.cpu: "4" limits.Memory: "8Gi"

Memory

Representada en MB (Mi) o GB (Gi)

Un gibibyte equivale a 2^{30} bytes, es 1,073,741,824 bytes, aproximadamente 1.07 GB

CPU

1 CPU = 1000 millicores (1000m)
200m significa 20% de un núcleo

Configurando | Guía

1. Realiza pruebas de carga

- Para entender cuántos recursos (CPU & Memory) necesita la aplicación en diferentes escenarios de uso.

2. Métricas en producción

- Para ajustar los valores basándote en el comportamiento real de la aplicación.

3. Configura request & limits

- request (garantizados)
- limits (máximos permitidos)

4. Considera la capacidad del nod

- Los valores de request & limits deben ser compatibles con los recursos disponibles en los nodos del clúster.

Configurando | Guía

5. Usa políticas de escalado

- **HPA o VPA**

6. Monitorea y ajusta periódicamente

- El uso de recursos puede cambiar con el tiempo debido a actualizaciones o cambios en la carga de trabajo.

Configurando | Worker Node

- Al configurar requests y limits para los recursos de un contenedor en un Pod, es esencial tener en cuenta las características y capacidades de los nodos del clúster.
- **Capacidad total del nodo**
 - *CPU*: Verifica cuántos núcleos de CPU están disponibles en el nodo.
 - Los requests y limits no deben superar la capacidad total de CPU del nodo, considerando otros Pods que puedan ejecutarse en el mismo nodo.
 - *Memoria*: Asegúrate de conocer la memoria total del nodo y cómo se distribuyen las cargas entre los Pods existentes.
 - Configurar límites demasiado altos puede causar una sobrecarga o impedir que Kubernetes programe el Pod.

Configurando | Worker Node

```
mnadmin@master:~/Intermedio$  
mnadmin@master:~/Intermedio$  
mnadmin@master:~/Intermedio$ kubectl get nodes -o wide  
NAME     STATUS    ROLES      AGE     VERSION   INTERNAL-IP     EXTERNAL-IP   OS-IMAGE        KERNEL-VERSION   CONTAINER-RUNTIME  
master   Ready     control-plane   39d     v1.30.5   192.168.0.3   <none>       Ubuntu 20.04.1 LTS   5.4.0-42-generic   containerd://1.7.22  
worker   NotReady  <none>      39d     v1.30.5   192.168.0.228  <none>       Ubuntu 20.04.1 LTS   5.4.0-42-generic   containerd://1.7.22  
worker2  NotReady  <none>      5d1h    v1.30.5   192.168.0.229  <none>       Ubuntu 20.04.1 LTS   5.4.0-42-generic   containerd://1.7.22  
worker4  Ready     <none>      4d22h   v1.30.7   192.168.0.79    <none>       Ubuntu 20.04.1 LTS   5.4.0-42-generic   containerd://1.7.23  
mnadmin@master:~/Intermedio$  
mnadmin@master:~/Intermedio$ kubectl describe node worker4 | wc -l  
74  
mnadmin@master:~/Intermedio$  
mnadmin@master:~/Intermedio$ kubectl get node worker4 -o jsonpath='{.metadata.name} {.status.capacity.cpu} {.status.capacity.memory} {.metadata.labels}'  
worker4 4 6747928Ki {"beta.kubernetes.io/arch":"amd64","beta.kubernetes.io/os":"linux","kubernetes.io/arch":"amd64","kubernetes.io/hostname":"worker4","kubernetes.io/os":"linux"}mnadmin@master:~/Intermedio$  
mnadmin@master:~/Intermedio$  
mnadmin@master:~/Intermedio$ kubectl get node worker4 -o jsonpath='{.metadata.name} {.status.capacity.cpu} {.status.capacity.memory} {.metadata.labels}' -o JSON | wc -l  
327  
mnadmin@master:~/Intermedio$  
mnadmin@master:~/Intermedio$ kubectl top node worker4  
NAME      CPU(cores)   CPU%      MEMORY(bytes)   MEMORY%  
worker4   99m          2%        3940Mi          60%  
mnadmin@master:~/Intermedio$  
mnadmin@master:~/Intermedio$ kubectl describe node worker4 | grep -A5 "Allocated resources"  
Allocated resources:  
(Total limits may be over 100 percent, i.e., overcommitted.)  
Resource      Requests     Limits  
-----  
cpu           300m (7%)  0 (0%)  
memory        290Mi (4%)  0 (0%)  
mnadmin@master:~/Intermedio$  
mnadmin@master:~/Intermedio$
```

- Este nodo tiene un uso de memoria relativamente alto (60%), pero el uso de CPU es bajo (2%).
- Las solicitudes (requests) de recursos son conservadoras, dejando espacio para programar más pods.
- Los límites de memoria no están configurados, lo que podría permitir a los contenedores exceder los recursos asignados.

Configurando | Deployment & Resources

```
resources:  
  requests:  
    memory: "256Mi"  
    cpu: "500m"  
  limits:  
    memory: "500Mi"  
    cpu: "800m"
```

- **request:** Garantiza 256 MB de RAM y 50% de un núcleo.
- **limits:** Permite hasta 400 MB de RAM y 70% de un núcleo en momentos de alta demanda.

Configurando | apply

```
mnadmin@master:~/Intermedio/ws12$  
mnadmin@master:~/Intermedio/ws12$  
mnadmin@master:~/Intermedio/ws12$ kubectl get all -n dki-curso  
NAME                         READY   STATUS    RESTARTS   AGE  
pod/ms-favoritos-dc7b94498-nv4b6   1/1     Running   0          2d3h  
  
NAME           TYPE      CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE  
service/ms-favoritos   NodePort   10.106.209.6 <none>        9094:31572/TCP   2d3h  
  
NAME                         READY   UP-TO-DATE   AVAILABLE   AGE  
deployment.apps/ms-favoritos   1/1     1           1          2d3h  
  
NAME                           DESIRED   CURRENT   READY   AGE  
replicaset.apps/ms-favoritos-dc7b94498   1         1         1       2d3h  
mnadmin@master:~/Intermedio/ws12$  
mnadmin@master:~/Intermedio/ws12$ kubectl apply -f ms-peliculas-deployment.yaml  
deployment.apps/ms-peliculas created  
mnadmin@master:~/Intermedio/ws12$  
mnadmin@master:~/Intermedio/ws12$ kubectl apply -f ms-peliculas-service.yaml  
service/ms-peliculas created  
mnadmin@master:~/Intermedio/ws12$
```

Configurando | Monitoreo

```
mnadmin@master:~$ kubectl get nodes -o wide
NAME      STATUS    ROLES      AGE     VERSION   INTERNAL-IP      EXTERNAL-IP   OS-IMAGE       KERNEL-VERSION   CONTAINER-RUNTIME
master    Ready     control-plane   39d    v1.30.5   192.168.0.3    <none>        Ubuntu 20.04.1 LTS   5.4.0-42-generic   containerd://1.7.22
worker    NotReady  <none>      39d    v1.30.5   192.168.0.228  <none>        Ubuntu 20.04.1 LTS   5.4.0-42-generic   containerd://1.7.22
worker2   NotReady  <none>      5d2h   v1.30.5   192.168.0.229  <none>        Ubuntu 20.04.1 LTS   5.4.0-42-generic   containerd://1.7.22
worker4   Ready     <none>      4d22h  v1.30.7   192.168.0.79   <none>        Ubuntu 20.04.1 LTS   5.4.0-42-generic   containerd://1.7.23
mnadmin@master:~$ kubectl get pods -n dki-curso -o wide
NAME                  READY   STATUS    RESTARTS   AGE     IP          NODE      NOMINATED-NODE   READINESS   GATES
ms-favoritos-dc7b94498-nv4b6  1/1    Running   0          2d3h   10.34.0.2   worker4   <none>        <none>
ms-peliculas-7f5c7f9cbf-v5qcm 1/1    Running   0          8m14s  10.34.0.8   worker4   <none>        <none>
ms-peliculas-7f5c7f9cbf-xf8lv 1/1    Running   0          8m14s  10.34.0.0   worker4   <none>        <none>
mnadmin@master:~$ kubectl top node worker4
NAME      CPU(cores)   CPU%    MEMORY(bytes)   MEMORY%
worker4   112m         2%     4627Mi        71%
mnadmin@master:~$ kubectl describe node worker4 | grep -A5 "Allocated resources"
Allocated resources:
  (Total limits may be over 100 percent, i.e., overcommitted.)
  Resource      Requests   Limits
  -----        -----      -----
  cpu           1300m (32%) 1600m (40%)
  memory        802Mi (12%) 1000Mi (15%)
mnadmin@master:~$
```

Configurando | Comparativa

Estado	CPU Request	CPU Limits	Memory Request	Memory Limits
Previo al despliegue	300m (7%)	0 (0%)	209Mi (4%)	0 (0%)
Con el despliegue ms-peliculas	1300m (32%)	1600m (40%)	802Mi (12%)	1000Mi (15%)

Estado	CPU Utilización	Memoria Utilización
Previo al despliegue	2%	60%
Con el despliegue ms-peliculas	2%	71%

Configurando | Herramientas

- **Prometheus**
 - Sistema de monitoreo y alertas diseñado para Kubernetes.
- **Grafana**
 - Herramienta de visualización de datos para métricas.
- **Kube-State-Metrics**
 - Exporta métricas específicas de Kubernetes, como el estado de los pods, despliegues y nodos.
 - Un enfoque en métricas de estado en lugar de métricas de rendimiento de sistema.

La combinación de herramientas como Prometheus, Grafana, Loki, y una solución de observabilidad como Datadog o Dynatrace, cubre la mayoría de las necesidades de monitoreo en Kubernetes.

Configurando | Herramientas

- **Lens**
 - GUI para gestionar y monitorear Kubernetes.
- **Octant**
 - Interfaz gráfica para monitoreo.
- **Datadog**
 - Monitoreo basado en SaaS para aplicaciones y Kubernetes.

Resumen

La Unidad 4 abordó la integración de Spring con Kubernetes, incluyendo la configuración de microservicios con Spring Cloud Kubernetes, despliegue en clústeres, manejo de entornos y ConfigMaps, monitoreo con Liveness y Readiness Probes, y la optimización de recursos de contenedores para entornos de desarrollo y producción.



Práctica 4. Spring Cloud Kubernetes

Objetivo:

Al finalizar esta práctica, serás capaz de integrar y aplicar todos los conocimientos adquiridos en la unidad 4 para configurar, desplegar y probar microservicios Spring Boot en un clúster de Kubernetes.

Planteamiento e instrucciones:

Realiza un seguimiento de los pasos indicados en la *Guía de Laboratorios* para llevar a cabo la tarea correspondiente en el siguiente enlace: [Plataforma de Laboratorios | DOCK_KUB_INT](#)



Tiempo para esta actividad:
180 minutos

Resultado esperado

NAME	READY	STATUS	RESTARTS	AGE
ms-productos-7d9f65f9f6-4jksm	1/1	Running	0	3m
ms-deseos-8476d9f8c7-1kqnp	1/1	Running	0	3m

Events:

Type	Reason	Age	From	Message
---	-----	----	----	-----
Normal	Scheduled	4m	default-scheduler	Successfully assigned default/ms-productos-7d9f65f9f6-4jksm to worker1
Normal	Pulled	4m	kubelet	Container image "<repo>/ms-productos:<version>" already present on machine
Normal	Created	4m	kubelet	Created container ms-productos
Normal	Started	4m	kubelet	Started container ms-productos

Resultado esperado

- ✓ `kubectl apply -f ms-productos-deployment.yaml`
- ✓ `kubectl apply -f ms-deseos-deployment.yaml`
- ✓ `kubectl apply -f ms-productos-service.yaml`
- ✓ `kubectl apply -f ms-deseos-service.yaml`

- ✓ `kubectl describe pod <nombre-pod>`
- ✓ `kubectl logs <nombre-pod>`

- ✓ `kubectl get pods`
- ✓ `kubectl get deploys`
- ✓ `kubectl get services -o wide`
- ✓ `kubectl get configmaps`
- ✓ `kubeclt get secrets`

- ✓ `curl http://<localhost|node-ip>:puerto/api-productos`
- ✓ `curl -X POST http://<localhost|node-ip>:puerto/deseos/user1/1`

Referencias Bibliográficas

- Spring Cloud Kubernetes
<https://spring.io/projects/spring-cloud-kubernetes>
- Spring Cloud Kubernetes Reference
<https://docs.spring.io/spring-cloud-kubernetes/reference/>
- Kubernetes Up & Running: Dive into the Future of Infrastructure, Kelsey Hightower, Brendan Burns, Joe Beda, editorial O'Reilly.
<https://www.oreilly.com/library/view/kubernetes-up-and/9781491935668/>
- Building a Restful Web Service with Spring Bot Actuator <https://spring.io/guides/gs/actuator-service>



Objetivos:

- Comprender el concepto y propósito de Spring Cloud Gateway.
- Configurar y desplegar un servicio de Spring Cloud Gateway.
- Escribir y aplicar configuraciones de rutas.
- Desplegar el Gateway en el clúster de Kubernetes.

Unidad 5

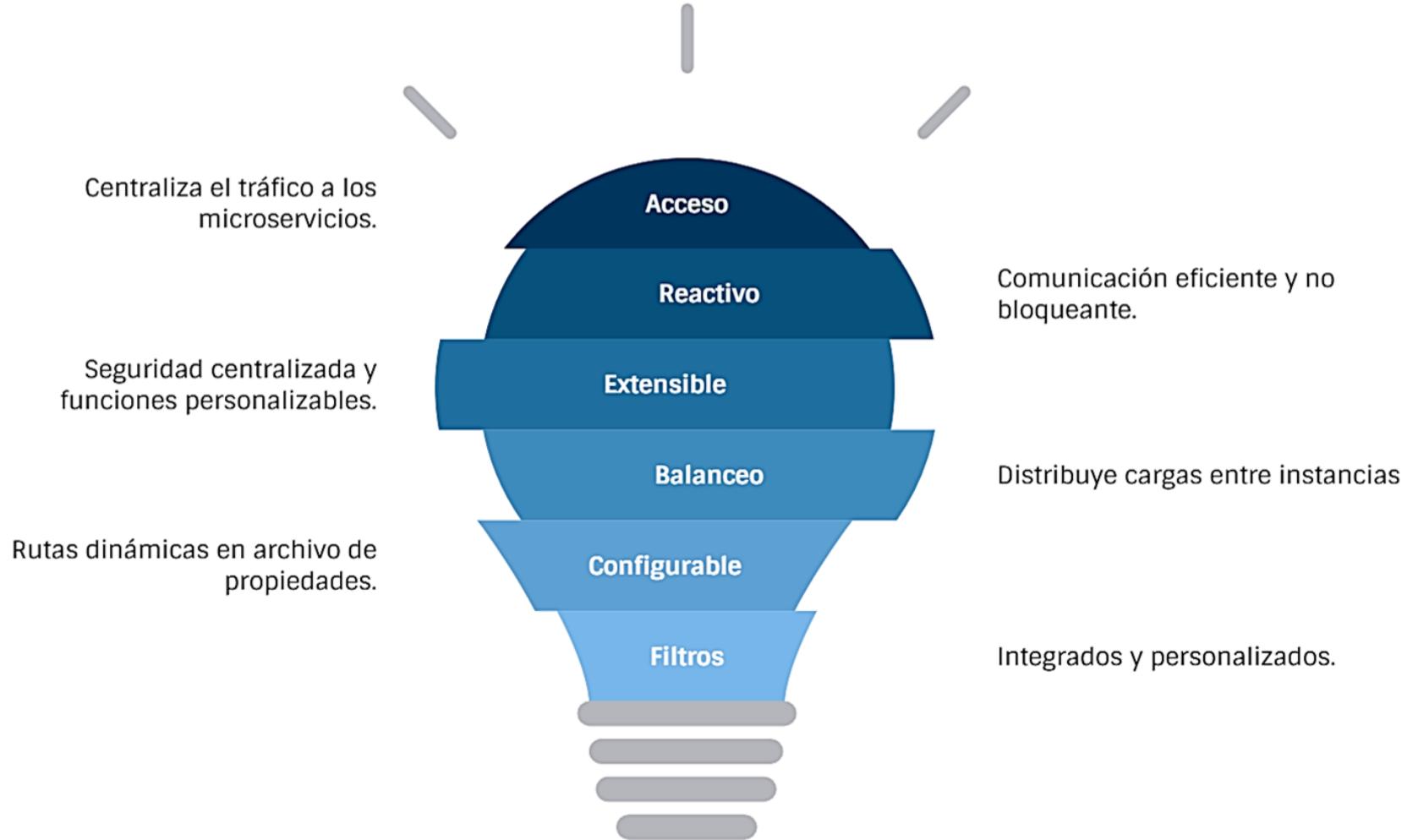
Kubernetes: Spring Cloud Gateway

5.1 Introducción a Spring Cloud Gateway

- ¿Por qué es importante contar con un enrutador API como [Spring Cloud Gateway](#) en una arquitectura de microservicios?
- ¿Cuáles son los principales beneficios que ofrece frente a otras soluciones de enrutamiento?*

Simplifica la gestión centralizada de tareas comunes, como enrutamiento, seguridad, y balanceo de carga, proporcionando escalabilidad y flexibilidad en arquitecturas basadas en microservicios.

Introducción a Spring Cloud Gateway



Introducción a Spring Cloud Gateway

1. Rutas

- Es la unidad principal de configuración en Spring Cloud Gateway.
- Representa la definición de cómo se manejará una solicitud entrante.

2. Predicates

- Son expresiones que evalúan si una solicitud entrante cumple determinadas condiciones.
- Las condiciones pueden basarse en diferentes atributos de la solicitud.
 - La URL, encabezados, métodos HTTP, entre otros.

Introducción a Spring Cloud Gateway

1. Filters

- Son responsables de modificar las solicitudes o respuestas durante su procesamiento.
- Los filtros puede ser globales o específicos de una ruta.

Una ruta es la configuración que combina predicates (para decidir si una solicitud coincide) y filters (para modificar el tráfico o agregar lógica adicional) con un destino específico.

Introducción a Spring Cloud Gateway

```
import org.springframework.cloud.gateway.route.RouteLocator;
import org.springframework.cloud.gateway.route.builder.RouteLocatorBuilder;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

@Configuration
public class GatewayConfig {

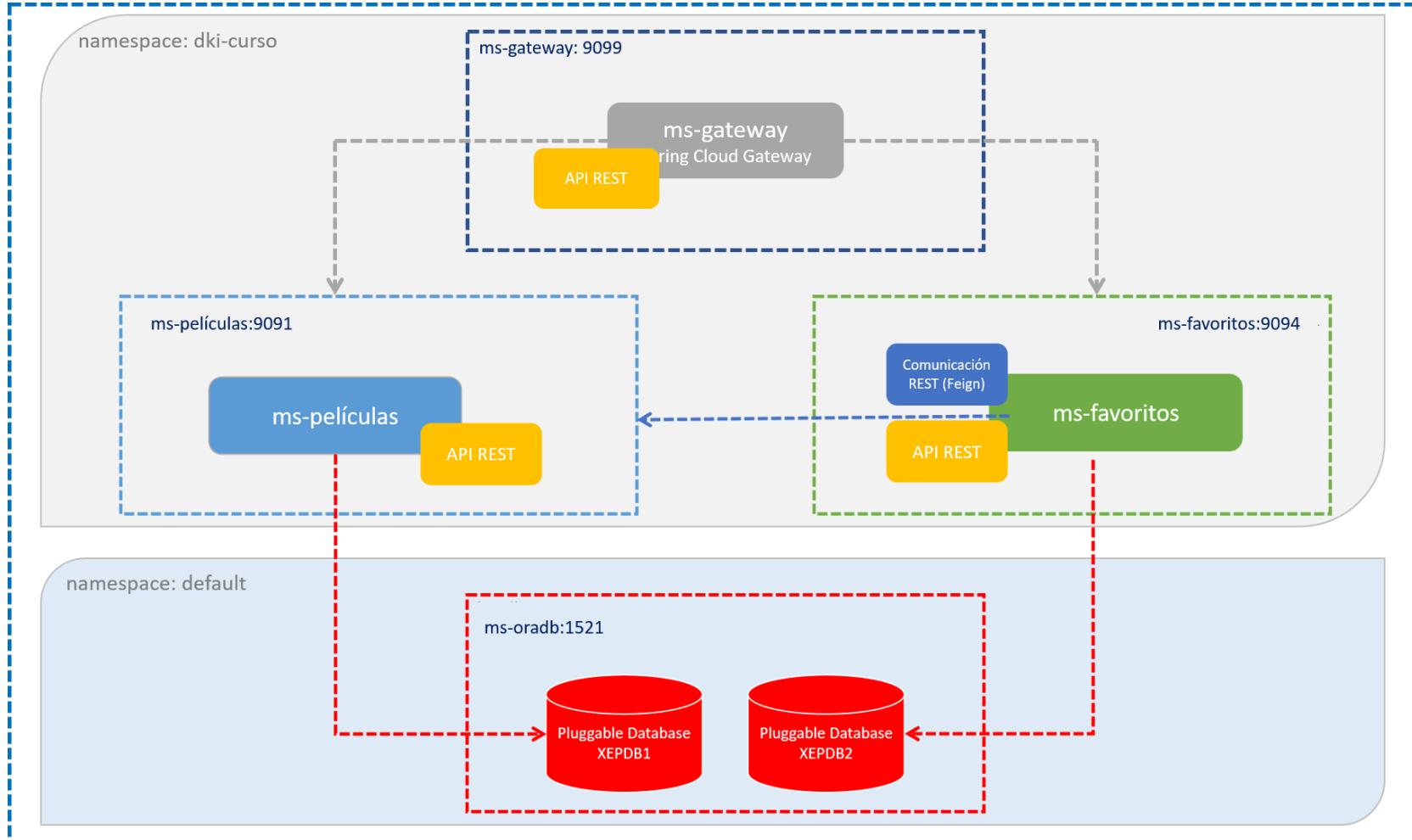
    @Bean
    public RouteLocator routeLocator(RouteLocatorBuilder builder) {
        return builder.routes()
            .route("example_route", route -> route
                .path("/api/products/**") //Predicate: La solicitud debe coincidir con este patrón
                .filters(f -> f
                    .addRequestHeader("X-Custom-Header", "CustomValue") // Filter: Agrega un encabezado
                    .rewritePath("/api/products/(?<segment>.*)", "/products/${segment}") // Filter: Reescribe la ruta
                )
                .uri("http://localhost:8081") // Destino del backend
            )
            .build();
    }
}
```

Introducción a Spring Cloud Gateway

```
spring:  
  cloud:  
    gateway:  
      routes:  
        - id: example_route  
          uri: http://localhost:8081  
          predicates:  
            - Path=/api/products/**  
          filters:  
            - AddRequestHeader=X-Custom-Header, CustomValue  
            - RewritePath=/api/products/(?<segment>.*), /products/${segment}
```

Ambos ejemplos logran lo mismo: redirigen solicitudes desde /api/products/** a http://localhost:8081 tras aplicar filtros para modificar la solicitud.

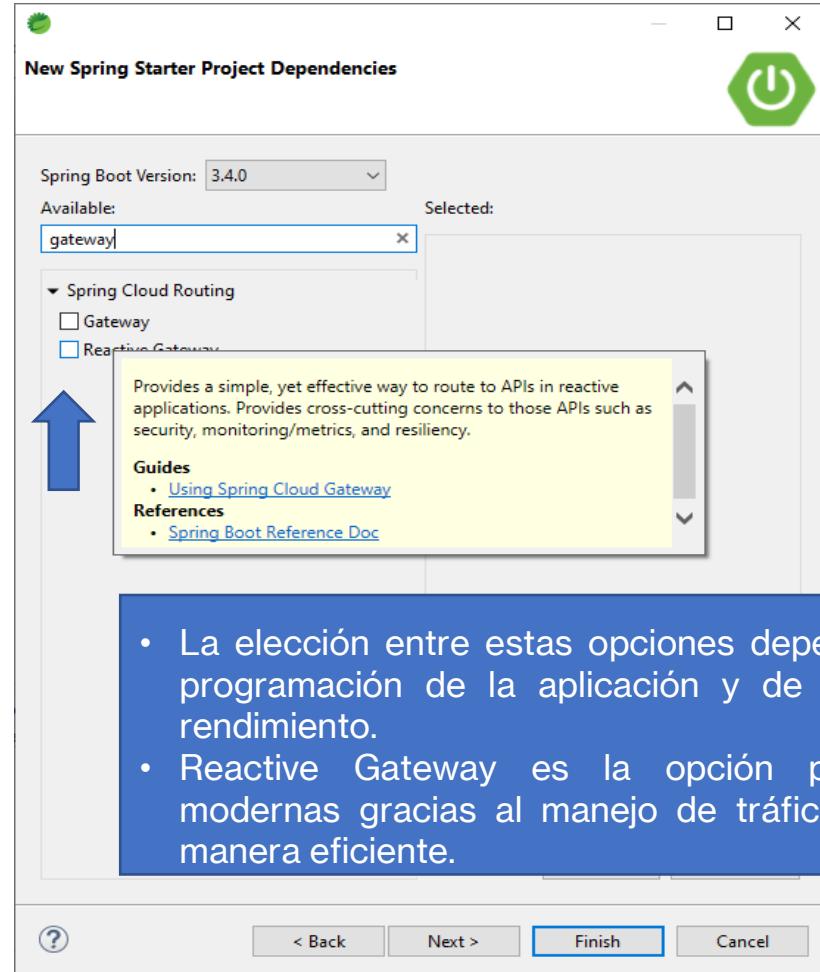
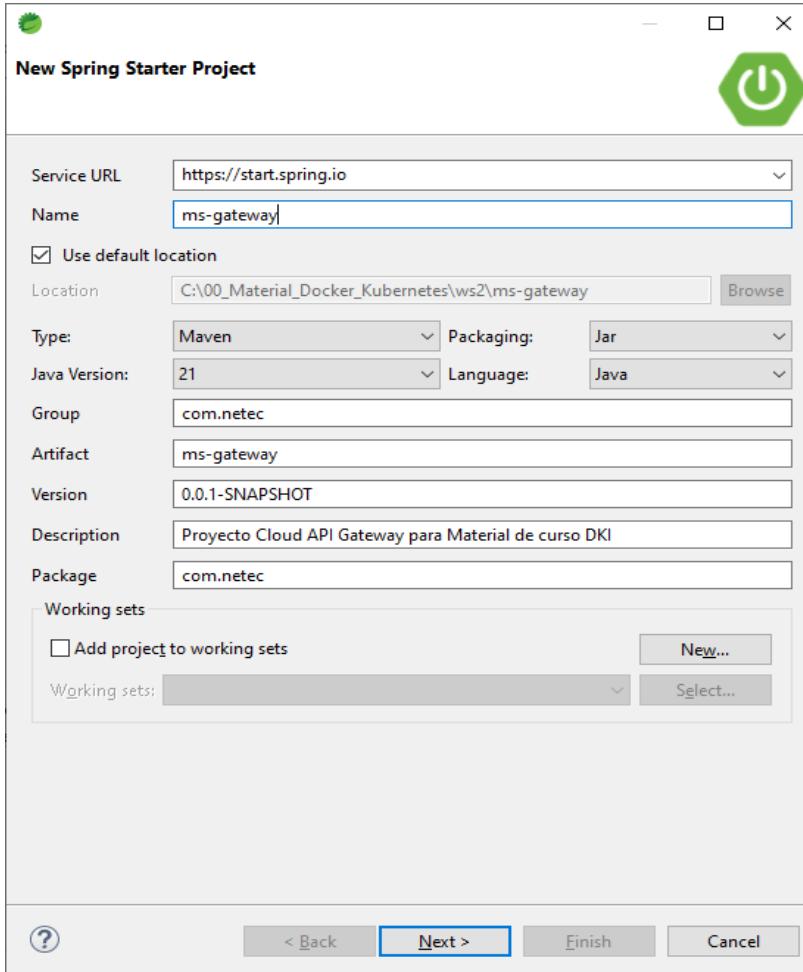
Introducción a SCG | Caso de estudio



5.2. Creando y configurando servicio Spring Cloud Gateway



Creando | Proyecto



- La elección entre estas opciones depende del estilo de programación de la aplicación y de los requisitos de rendimiento.
- Reactive Gateway es la opción para aplicaciones modernas gracias al manejo de tráfico concurrente de manera eficiente.

Creando | Dependencias

```
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-gateway</artifactId>
</dependency>
<!-- DKI -->
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-kubernetes-client</artifactId>
    <version>3.1.2</version>
</dependency>

<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-kubernetes-client-config</artifactId>
    <version>3.1.2</version>
</dependency>

<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-kubernetes-client-loadbalancer</artifactId>
    <version>3.1.2</version>
</dependency>
```

- Es la dependencia principal para crear el API Gateway reactivo. Proporciona enrutamiento dinámico, balanceo de carga, resiliencia, y otras funcionalidades.

Creando y configurando servicio SCG

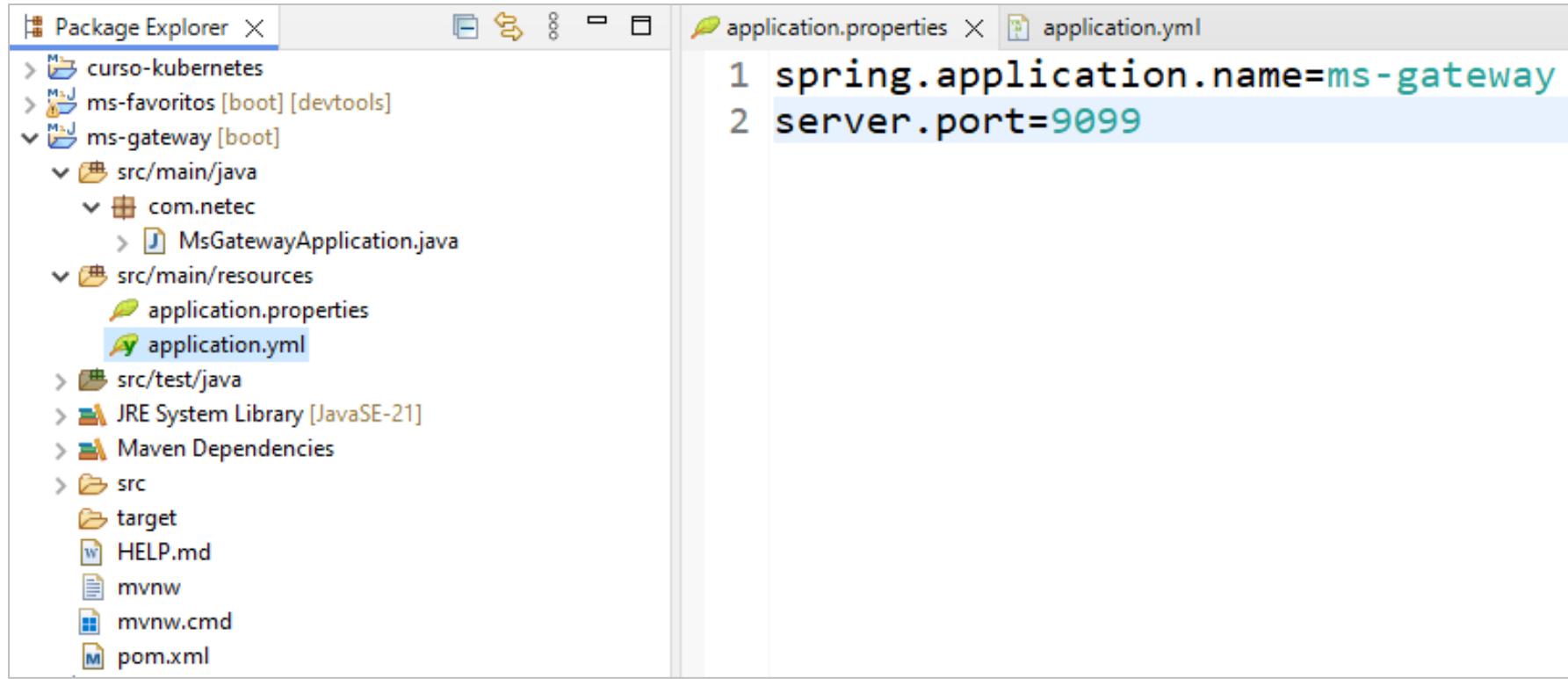
```
package com.netec;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.cloud.client.discovery.EnableDiscoveryClient;

@EnableDiscoveryClient
@SpringBootApplication
public class MsGatewayApplication {
    public static void main(String[] args) {
        SpringApplication.run(MsGatewayApplication.class, args);
    }
}
```

Creando y configurando servicio SCG

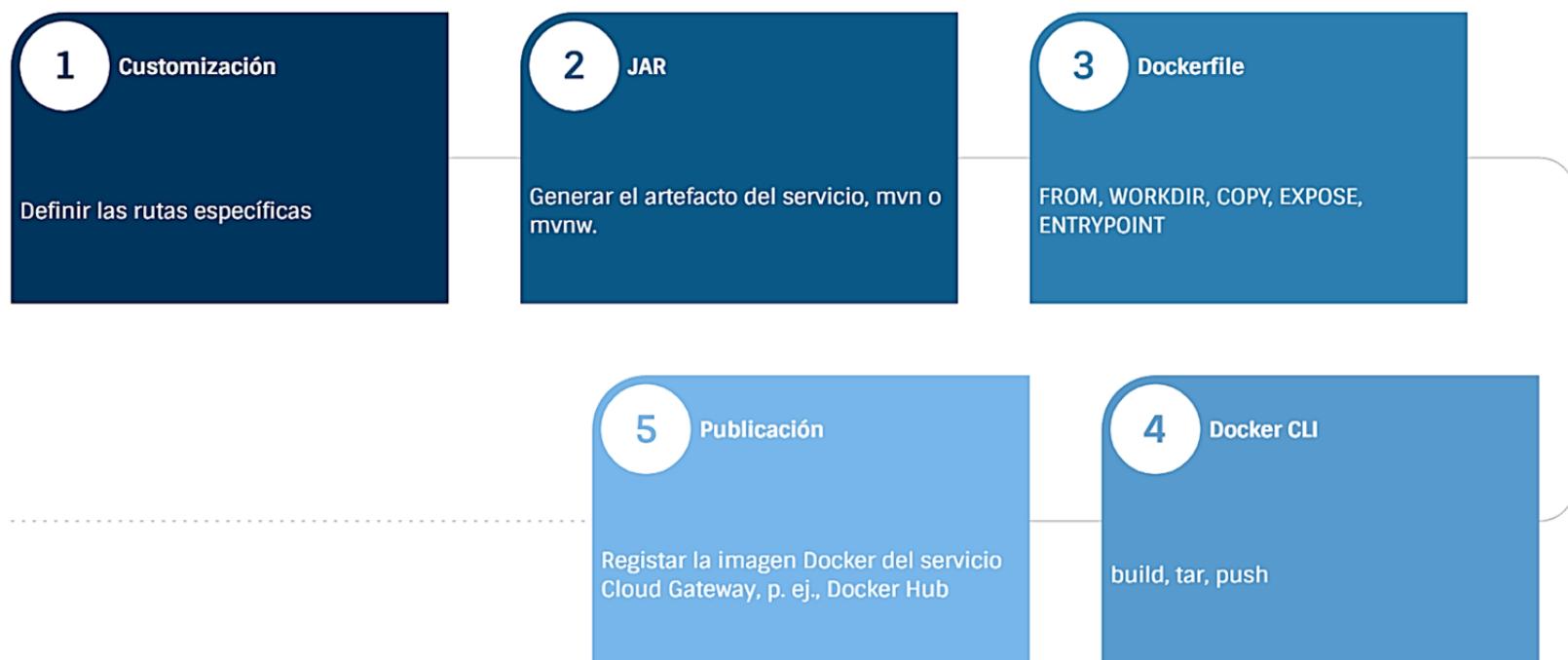


The screenshot shows a Java development environment with the following details:

- Package Explorer:** Displays the project structure:
 - curso-kubernetes
 - ms-favoritos [boot] [devtools]
 - ms-gateway [boot]
 - src/main/java
 - com.netec
 - MsGatewayApplication.java
 - src/main/resources
 - application.properties
 - application.yml
 - src/test/java
 - JRE System Library [JavaSE-21]
 - Maven Dependencies
 - src
 - target
 - HELP.md
 - mvnw
 - mvnw.cmd
 - pom.xml
- application.properties:** Configuration file content:

```
1 spring.application.name=ms-gateway
2 server.port=9099
```
- application.yml:** Configuration file content (empty)

5.3. Configurando rutas de mservicios en Gateway y Dockerizando



Configurando rutas

```
spring:  
  cloud:  
    gateway:  
      routes:  
        - id: <nombre-unico-ruta>  
          uri: <protocolo-destino>://<dirección-destino>  
          predicates:  
            - <condiciones-entrada>  
          filters:  
            - <modificaciones-a-solicitud>
```

- **id:** Identificador único para la ruta.
- **uri:** Especifica el destino al cual el Gateway debe enrutar las solicitudes.
- **predicates:** Define las condiciones que debe cumplir una solicitud entrante para coincidir con esta ruta.
- **filters:** Permite modificar solicitudes o respuestas antes de que lleguen al destino o al cliente.

Configurando rutas | Caso de estudio

```
# application.yml
# Rutas Caso de Estudio DKI
spring:
  cloud:
    gateway:
      routes:
        - id: ms-peliculas
          uri: lb://ms-peliculas
          predicates:
            - Path=/api/peliculas/**
          filters:
            - StripPrefix=2
        - id: ms-fvoritos
          uri: lb://ms-favoritos
          predicates:
            - Path=/api/favoritos/**
          filters:
            - StripPrefix=2
```

El prefijo **lb://** indica que la URI utiliza Spring Cloud LoadBalancer para resolver el servicio registrado.

Configurando rutas | Dockerfile

```
# 1. Imagen base
FROM openjdk:21-jdk-slim

# 2. Establecer el directorio de trabajo
WORKDIR /app

# 3. Copiar el archivo JAR generado
COPY target/ms-gateway-0.0.1-SNAPSHOT.jar app.jar

# 4. Exponer el puerto utilizado por el microservicio
EXPOSE 9099

# 5. Comando de inicio
ENTRYPOINT ["java", "-jar", "app.jar"]
```

Configurando rutas | Docker

```
C:\ Seleccionar Símbolo del sistema - .\mvnw clean install -Dmaven.test.skip=true - .\mvnw clean install -Dmaven.test.skip=true - .\mvnw clean install -Dmave.test.skip=tr...
=> CACHED [3/3] COPY target/ms-gateway-0.0.1-SNAPSHOT.jar app.jar 0.0s
=> exporting to image 0.4s
=> => exporting layers 0.0s
=> => exporting manifest sha256:9aed91a3f19829cb963eec4313f173cb5583d3450f826c5583f10e4b04e0ed91 0.0s
=> => exporting config sha256:1845e9167cc9d6f02f9667286510957edb438b19d51d9a399567fd448efd2293 0.0s
=> => exporting attestation manifest sha256:48e57ef12dcf4f2597454ca833cabd13a9580690fcf03267504bc2cea2f579c4 0.1s
=> => exporting manifest list sha256:f9eb86f236693a03dcf1df45ac97a0b07e3eb94fd380fea4742e76a28c1972bd 0.1s
=> => naming to docker.io/library/ms-gateway:1.0.0 0.0s
=> => unpacking to docker.io/library/ms-gateway:1.0.0 0.0s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/snn3luhrnshpkwm72yypj15tp

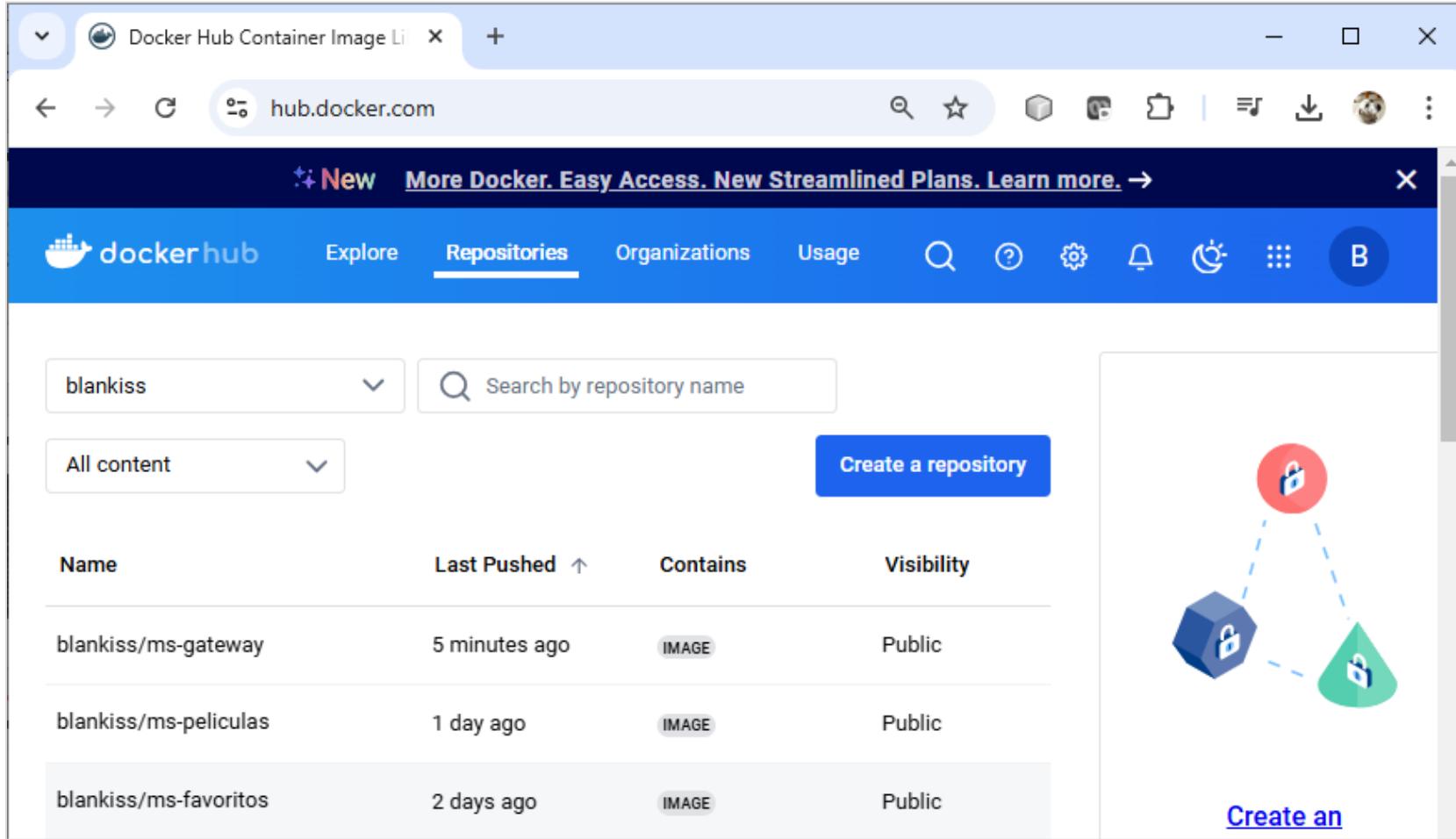
What's next:
    View a summary of image vulnerabilities and recommendations → docker scout quickview

C:\00_Material_Docker_Kubernetes\ws2\ms-gateway>docker tag ms-gateway:1.0.0 blankiss/ms-gateway:1.0.0

C:\00_Material_Docker_Kubernetes\ws2\ms-gateway>docker push blankiss/ms-gateway:1.0.0
The push refers to repository [docker.io/blankiss/ms-gateway]
9b2d3c368f4f: Layer already exists
a803e7c4b030: Layer already exists
2e05a12a4efc: Pushed
3d69f376643f: Layer already exists
b4972576c83d: Layer already exists
af800cd8441e: Layer already exists
1.0.0: digest: sha256:f9eb86f236693a03dcf1df45ac97a0b07e3eb94fd380fea4742e76a28c1972bd size: 856

C:\00_Material_Docker_Kubernetes\ws2\ms-gateway>
```

Configurando rutas | Docker Hub



The screenshot shows a web browser window for 'Docker Hub Container Image Library' at 'hub.docker.com'. The navigation bar includes 'Explore', 'Repositories' (which is selected), 'Organizations', 'Usage', and various icons for search, help, and account management. A dropdown menu shows 'blankiss' and a search bar with 'Search by repository name'. A dropdown menu for 'All content' is open. A prominent blue button says 'Create a repository'. Below, a table lists three repositories:

Name	Last Pushed	Contains	Visibility
blankiss/ms-gateway	5 minutes ago	IMAGE	Public
blankiss/ms-peliculas	1 day ago	IMAGE	Public
blankiss/ms-favoritos	2 days ago	IMAGE	Public

To the right, there's a graphic of three interconnected nodes (red circle, blue hexagon, green triangle) each with a lock icon, and a 'Create an' button.

5.4. Configuración de rutas usando application.properties y application.yml



Configuración de rutas usando application.properties y application.yml

Convertir application.properties a application.yml

1. Navega a tu archivo application.properties en el proyecto.
2. Haz clic derecho sobre el archivo.
3. Selecciona la opción:
 - **Spring > Convert to YAML.**
4. Automáticamente, se generará el archivo application.yml con las configuraciones convertidas.

Convertir application.yml a application.properties

1. Navega a tu archivo application.yml en el proyecto.
2. Haz clic derecho sobre el archivo.
3. Selecciona la opción:
 - **Spring > Convert to Properties.**
4. Automáticamente, se generará el archivo application.properties con las configuraciones convertidas.

Configuración de rutas usando application.properties y application.yml

```
# Rutas Caso de Estudio DKI
spring:
  cloud:
    gateway:
      routes:
        - id: ms-peliculas
          uri: lb://ms-peliculas
          predicates:
            - Path=/api/peliculas/**
          filters:
            - StripPrefix=2
        - id: ms-fvoritos
          uri: lb://ms-favoritos
          predicates:
            - Path=/api/favoritos/**
          filters:
            - StripPrefix=2
```

```
# Conversion to YAML from Properties formar report
# Warnings:
# - The yaml file had comments which are lost in the refactoring!

spring.cloud.gateway.routes[0].id=ms-peliculas
spring.cloud.gateway.routes[0].uri=lb\://ms-peliculas
spring.cloud.gateway.routes[0].predicates[0]=Path\=/api/peliculas/**
spring.cloud.gateway.routes[0].filters[0]=StripPrefix\=2

spring.cloud.gateway.routes[1].id=ms-fvoritos
spring.cloud.gateway.routes[1].uri=lb\://ms-favoritos
spring.cloud.gateway.routes[1].predicates[0]=Path\=/api/favoritos/**
spring.cloud.gateway.routes[1].filters[0]=StripPrefix\=2
```

5.5. Deployment y Service de Gateway

1. ¿Cuáles son los elementos clave que deben incluirse en un archivo YAML para definir un Deployment?
2. ¿Cuáles son los elementos clave que deben incluirse en un archivo YAML para definir un Service?
3. ¿Cuál es la diferencia principal entre un Deployment y un Service en Kubernetes, y cómo se complementan en un entorno de clúster?

- **Deployment:** Administra la creación y actualización de Pods.
- **Service:** Expone y conecta los Pods al tráfico interno o externo.
- **Relación:** El Service dirige el tráfico a los Pods gestionados por el Deployment.

Deployment y Service de Gateway

- **apiVersion:** Define la versión de la API de Kubernetes utilizada, generalmente `apps/v1`.
- **kind:** Especifica el tipo de objeto, en este caso, **Deployment**.
- **metadata:** Contiene información básica como el nombre y las etiquetas del Deployment.
- **spec:** Define las especificaciones del Deployment.
 - **replicas:** Número de réplicas del Pod que se desean ejecutar.
 - **selector:** Criterios para identificar los Pods gestionados por el Deployment.
 - **template:** Plantilla utilizada para definir cómo deben ser los Pods.

Deployment y Service de Gateway

- **metadata**: Etiquetas asociadas a los Pods creados.
- **spec**: Especificaciones del contenedor dentro del Pod.
 - **containers**: Lista de contenedores definidos, incluyendo:
 - **name**: Nombre del contenedor.
 - **image**: Imagen Docker que se usará para el contenedor.
 - **ports**: Puertos expuestos por el contenedor.
 - **resources**: (Opcional) Define los límites y solicitudes de recursos.
 - **readinessProbe**: (Opcional) Verifica si el Pod está listo para recibir tráfico.
 - **livenessProbe**: (Opcional) Verifica si el Pod está funcionando correctamente.

Deployment | Caso de estudio

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: ms-gateway
  namespace: dki-curso
spec:
  replicas: 1
  selector:
    matchLabels:
      role: ms-gateway
  template:
    metadata:
      labels:
        role: ms-gateway
    spec:
      containers:
        - name: ms-gateway
          image: blankiss/ms-gateway:1.0.0
      ports:
        - containerPort: 9099
```

Deployment y Service de Gateway

- **apiVersion:** Define la versión de la API de Kubernetes utilizada.
- **kind:** Especifica el tipo de objeto, en este caso, **Service**.
- **metadata:** Contiene información básica como el nombre del Service y etiquetas opcionales.
- **spec:** Define las especificaciones del Service.
 - **type:** Determina cómo el Service expone los Pods.
 - por ejemplo, ClusterIP, NodePort, LoadBalancer.
 - **ports:** Lista de puertos expuestos por el Service, incluyendo:
 - **port:** Puerto accesible en el Service.
 - **targetPort:** Puerto del contenedor al que redirige el tráfico.
 - **nodePort:** (Opcional) Puerto del nodo para el tipo NodePort.
- **selector:** Etiquetas utilizadas para identificar los Pods que el Service debe gestionar

Service de Gateway | Caso de estudio

```
apiVersion: v1
kind: Service
metadata:
  name: ms-gateway
  namespace: dki-curso
spec:
  type: LoadBalancer
  ports:
    - protocol: TCP
      port: 9099
      targetPort: 9099
  selector:
    role: ms-gateway
```

LoadBalancer: Proporciona una solución completa para acceso externo con balanceo automático de tráfico entre las réplicas del servicio.

5.6. Escribiendo Deployment y Service de Gateway y probando en Postman

1 Crear

Crear los manifiestos YAML para Deployment & Service.

2 Aplicar

```
kubectl apply -f <archivo>.yml
```

3 Verificar

```
kubectl get [deployments|pods|services]
```

6 Consumir

```
curl | Postman http://<node-ip>:<puerto>/endpoint
```

5 Inspeccionar

```
kubectl log <pod> | kubectl get pods
```

4 Acceder

```
kubectl get nodes -o wide
```

Escribiendo Deployment | Aplicar

```
madmin@master:~/Intermedio/ws13$ kubectl get all -n dki-curso
NAME                               READY   STATUS    RESTARTS   AGE
pod/ms-favoritos-dc7b94498-nv4b6  1/1    Running   0          2d10h
pod/ms-peliculas-7f5c7f9cbf-v5qcm 1/1    Running   0          6h50m
pod/ms-peliculas-7f5c7f9cbf-xf81v  1/1    Running   0          6h50m

NAME           TYPE      CLUSTER-IP      EXTERNAL-IP      PORT(S)        AGE
service/ms-favoritos  NodePort  10.106.209.6  <none>        9094:31572/TCP  2d10h
service/ms-peliculas  NodePort  10.109.78.15  <none>        9091:30181/TCP  6h49m

NAME           READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/ms-favoritos  1/1     1           1           2d10h
deployment.apps/ms-peliculas  2/2     2           2           6h50m

NAME           DESIRED  CURRENT  READY   AGE
replicaset.apps/ms-favoritos-dc7b94498  1        1        1      2d10h
replicaset.apps/ms-peliculas-7f5c7f9cbf  2        2        2      6h50m
madmin@master:~/Intermedio/ws13$ kubectl apply -f ms-gateway.yaml
deployment.apps/ms-gateway created
service/ms-gateway created
madmin@master:~/Intermedio/ws13$ |
```

Escribiendo Deployment | Verificar

```
mnadmin@master:~/Intermedio/ws13$ kubectl get all -n dki-curso
NAME                                         READY   STATUS    RESTARTS   AGE
pod/ms-favoritos-dc7b94498-nv4b6          1/1     Running   0          2d10h
pod/ms-gateway-69555cc6f9-jtdgj           1/1     Running   0          2m31s
pod/ms-peliculas-7f5c7f9cbf-v5qcm         1/1     Running   0          6h52m
pod/ms-peliculas-7f5c7f9cbf-xf81v         1/1     Running   0          6h52m

NAME            TYPE        CLUSTER-IP      EXTERNAL-IP      PORT(S)      AGE
service/ms-favoritos  NodePort    10.106.209.6   <none>        9094:31572/TCP  2d10h
service/ms-gateway   LoadBalancer 10.101.244.22  <pending>      9099:30861/TCP  2m31s
service/ms-peliculas NodePort    10.109.78.15   <none>        9091:30181/TCP  6h52m

NAME             READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/ms-favoritos  1/1     1           1           2d10h
deployment.apps/ms-gateway    1/1     1           1           2m31s
deployment.apps/ms-peliculas 2/2     2           2           6h52m

NAME            DESIRED  CURRENT  READY   AGE
replicaset.apps/ms-favoritos-dc7b94498  1        1        1      2d10h
replicaset.apps/ms-gateway-69555cc6f9   1        1        1      2m31s
replicaset.apps/ms-peliculas-7f5c7f9cbf 2        2        2      6h52m
mnadmin@master:~/Intermedio/ws13$
```

Escribiendo Deployment | Acceder

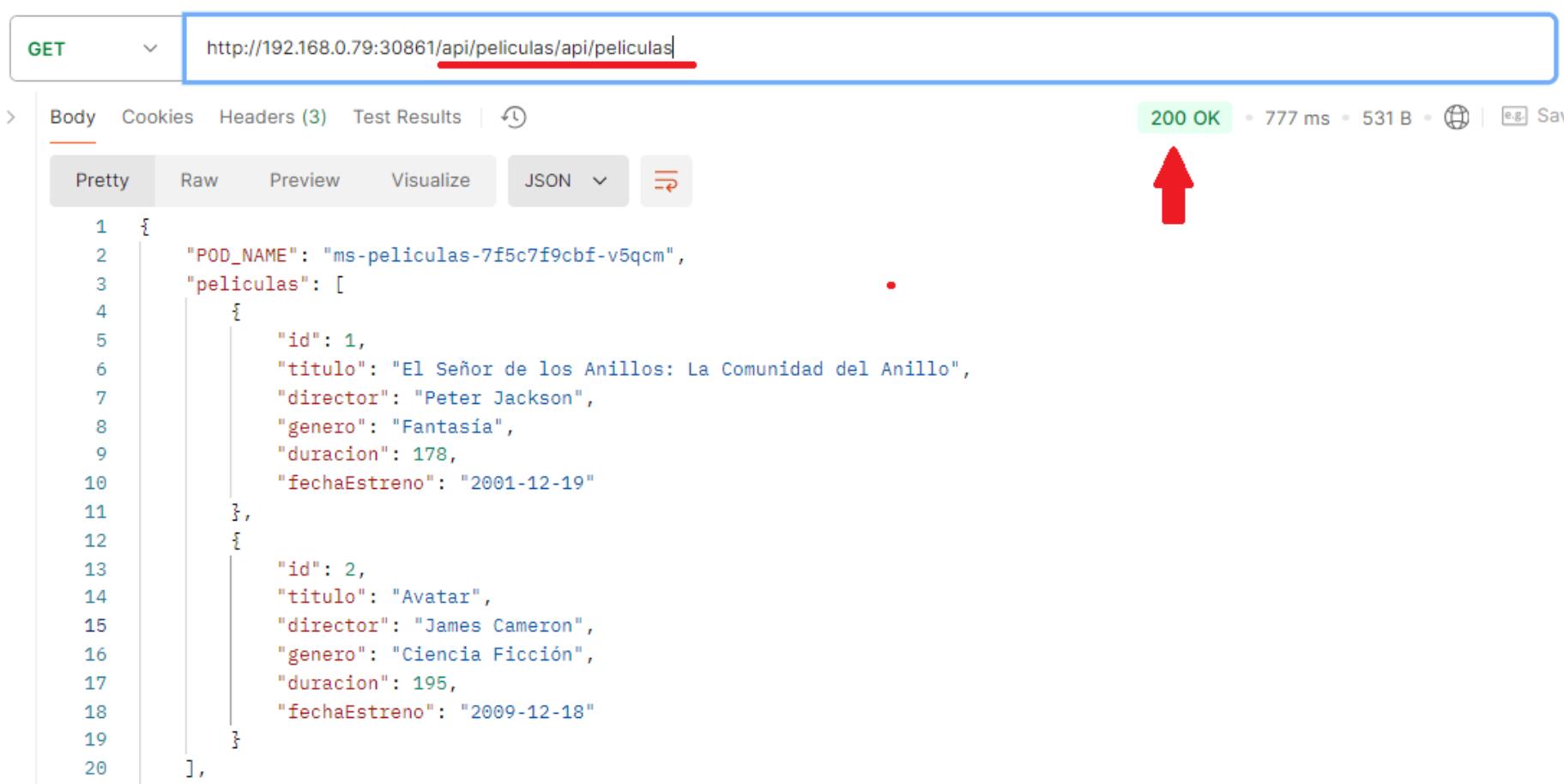
```
mnadmin@master:~/Intermedio/ws13$  
mnadmin@master:~/Intermedio/ws13$  
mnadmin@master:~/Intermedio/ws13$  
mnadmin@master:~/Intermedio/ws13$ kubectl describe service ms-gateway -n dki-curso  
Name:           ms-gateway  
Namespace:      dki-curso  
Labels:         <none>  
Annotations:    <none>  
Selector:       role=ms-gateway  
Type:          LoadBalancer  
IP Family Policy: SingleStack  
IP Families:   IPv4  
IP:            10.101.244.22  
IPs:           10.101.244.22  
Port:          <unset>  9099/TCP  
TargetPort:    9099/TCP  
NodePort:      <unset>  30861/TCP  
Endpoints:     10.34.0.9:9099  
Session Affinity: None  
External Traffic Policy: Cluster  
Events:        <none>  
mnadmin@master:~/Intermedio/ws13$ |
```

Escribiendo Deployment | Inspeccionar

```
mnadmin@master:~/Intermedio/ws13$ kubectl get services -n dki-curso -o wide
NAME           TYPE      CLUSTER-IP   EXTERNAL-IP   PORT(S)        AGE   SELECTOR
ms-favoritos   NodePort  10.106.209.6 <none>        9094:31572/TCP  2d10h  app=ms-favoritos
ms-gateway     LoadBalancer 10.101.244.22 <pending>    9099:30861/TCP  3m26s  role=ms-gateway
ms-peliculas   NodePort  10.109.78.15  <none>        9091:30181/TCP  6h53m  app=ms-peliculas
mnadmin@master:~/Intermedio/ws13$ |
```

30861

Escribiendo Deployment | Consumo

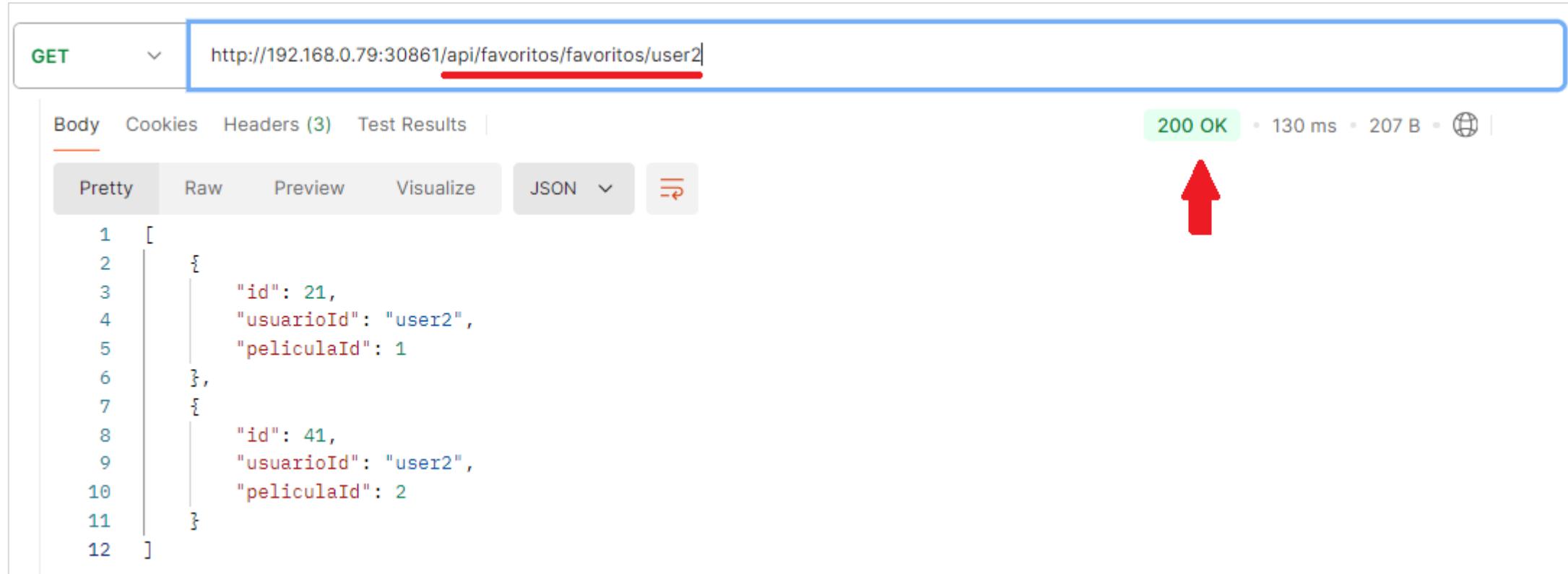


A screenshot of the Postman application interface. The top bar shows a GET request to the URL `http://192.168.0.79:30861/api/peliculas/api/peliculas`. The response status is 200 OK, with a duration of 777 ms and a body size of 531 B. A red arrow points upwards from the status bar towards the status code.

The Body tab displays the JSON response:

```
1 {  
2   "POD_NAME": "ms-peliculas-7f5c7f9cbf-v5qcm",  
3   "peliculas": [  
4     {  
5       "id": 1,  
6       "titulo": "El Señor de los Anillos: La Comunidad del Anillo",  
7       "director": "Peter Jackson",  
8       "genero": "Fantasia",  
9       "duracion": 178,  
10      "fechaEstreno": "2001-12-19"  
11    },  
12    {  
13      "id": 2,  
14      "titulo": "Avatar",  
15      "director": "James Cameron",  
16      "genero": "Ciencia Ficción",  
17      "duracion": 195,  
18      "fechaEstreno": "2009-12-18"  
19    }  
20  ],  
21 }
```

Escribiendo Deployment | Postman



GET <http://192.168.0.79:30861/api/favoritos/favoritos/user2>

Body Cookies Headers (3) Test Results | 200 OK • 130 ms • 207 B • 

Pretty Raw Preview Visualize JSON 

```
1 [  
2 {  
3   "id": 21,  
4   "usuarioId": "user2",  
5   "peliculaId": 1  
6 },  
7 {  
8   "id": 41,  
9   "usuarioId": "user2",  
10  "peliculaId": 2  
11 }  
12 ]
```

Resumen

Esta unidad proporcionó una comprensión de cómo utilizar Spring Cloud Gateway para gestionar y enrutar tráfico en un entorno de microservicios desplegado en Kubernetes, consolidando conceptos prácticos como la creación de manifiestos YAML, la configuración de rutas y la validación de servicios.



Práctica 5. Spring Cloud Gateway

Objetivo:

Al finalizar esta práctica, serás capaz de implementar y configurar un servicio de Spring Cloud Gateway en un clúster de Kubernetes.

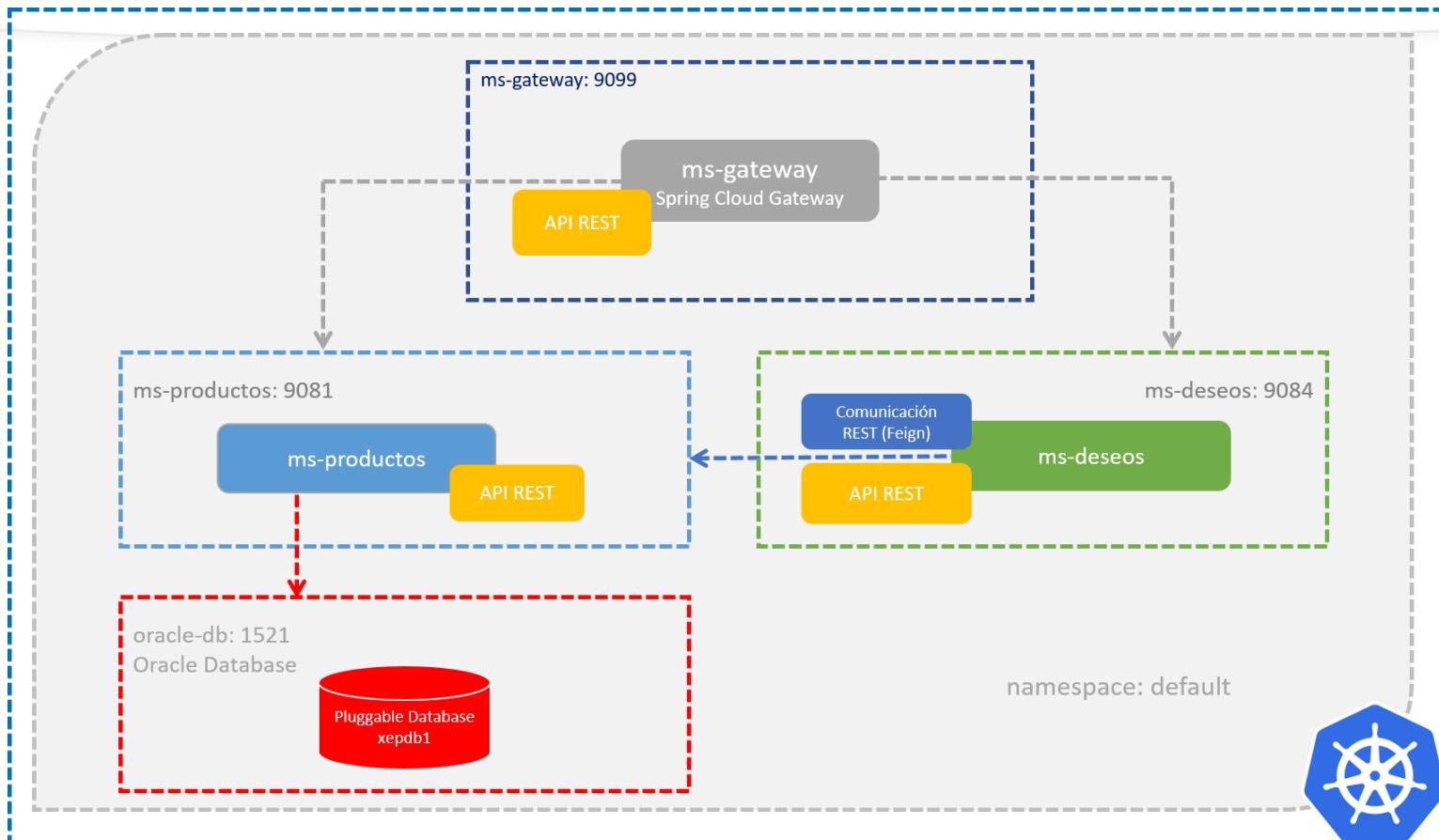
Planteamiento e instrucciones:

Realiza un seguimiento de los pasos indicados en la *Guía de Laboratorios* para llevar a cabo la tarea correspondiente en el siguiente enlace: [Plataforma de Laboratorios | DOCK_KUB_INT](#)



Tiempo para esta actividad:
180 minutos.

Resultado esperado



Resultado esperado

- ✓ `kubectl apply -f ms-productos-deployment.yaml`
- ✓ `kubectl apply -f ms-deseos-deployment.yaml`
- ✓ `kubectl apply -f ms-productos-service.yaml`
- ✓ `kubectl apply -f ms-deseos-service.yaml`

- ✓ `kubectl describe pod <nombre-pod>`
- ✓ `kubectl logs <nombre-pod>`

- ✓ `kubectl get pods`
- ✓ `kubectl get deploys`
- ✓ `kubectl get services -o wide`
- ✓ `kubectl get configmaps`
- ✓ `kubeclt get secrets`

- ✓ `curl http://<localhost|node-ip>:puerto/api-productos`
- ✓ `curl -X POST http://<localhost|node-ip>:puerto/deseos/user1/1`

Referencias Bibliográficas

- Spring Cloud Gateway
- <https://spring.io/projects/spring-cloud-gateway>
- 26 Top Kubernetes Tools for your K8s Ecosystem in 2024
- <https://spacelift.io/blog/kubernetes-tools>
- Spring Cloud Gateway Reference
- <https://docs.spring.io/spring-cloud-gateway/reference/>

Glosario Docker

- **Container:** Instancia en ejecución de una imagen. Es una unidad aislada y ligera de software.
- **Dockerfile:** Archivo de texto que contiene instrucciones para construir una imagen Docker.
- **Docker CLI:** Interfaz de línea de comandos para gestionar imágenes, contenedores, redes y volúmenes.
- **Docker Desktop:** Aplicación para desarrollar, ejecutar y gestionar contenedores Docker en sistemas operativos Windows y macOS.
- **Docker Hub:** Repositorio público donde los usuarios pueden compartir y descargar imágenes Docker.
- **Image:** Plantilla inmutable que contiene todo lo necesario para ejecutar una aplicación, incluyendo el sistema operativo, las dependencias y la propia aplicación.
- **Network:** Sistema que conecta contenedores entre sí o con el mundo exterior.
- **Volume:** Mecanismo de almacenamiento persistente para contenedores.

Glosario Docker Compose

- **docker-compose.yml**: Archivo de configuración YAML utilizado para definir y ejecutar aplicaciones multicontenedor.
- **Docker Compose CLI**: Herramienta de línea de comandos que facilita la definición y gestión de aplicaciones multicontenedor.
- **Networks**: Configuración que permite la comunicación entre servicios definidos en Docker Compose.
- **Service**: Definición de un contenedor dentro del archivo `docker-compose.yml`, incluyendo su imagen, red, volumen, entre otros.
- **Volumes**: Persistencia de datos compartida entre contenedores o con el sistema host.

Glosario Kubernetes

- **Cluster:** Conjunto de nodos (máquinas) que ejecutan aplicaciones contenidas en Pods.
- **ConfigMap:** Objeto utilizado para almacenar configuraciones en forma de pares clave-valor.
- **Deployment:** Objeto de Kubernetes que gestiona la creación y escalado de Pods.
- **Node:** Máquina física o virtual en un clúster de Kubernetes que ejecuta Pods.
- **PersistentVolume (PV):** Unidad de almacenamiento gestionada por Kubernetes para la persistencia de datos.
- **PersistentVolumeClaim (PVC):** Solicitud de un volumen persistente hecha por un usuario o aplicación
- **Pod:** Unidad más pequeña y básica en Kubernetes que encapsula uno o más contenedores.
- **Secret:** Similar a ConfigMap, pero diseñado para datos sensibles como contraseñas.
- **Service:** Punto de acceso lógico que expone los Pods al exterior o dentro del clúster.

Glosario Java Microservicios

- **Actuator**: Herramienta en Spring Boot para monitorizar y gestionar aplicaciones.
- **Controller**: Componente que gestiona las solicitudes HTTP en un microservicio.
- **Dockerizar**: Proceso de empacar un microservicio Java en un contenedor Docker.
- **DTO (Data Transfer Object)**: Objeto que encapsula datos para transferir entre capas de la aplicación o microservicios.
- **Entity**: Clase de Java que representa una tabla en la base de datos.
- **Feign**: Cliente REST declarativo que simplifica las llamadas HTTP entre microservicios.
- **JPA (Java Persistence API)**: Interfaz estándar de Java para acceder, persistir y gestionar datos en bases de datos relacionales.
- **Repository**: Interfaz que abstrae la lógica de acceso y persistencia de datos.
- **Service**: Capa de lógica de negocio en un microservicio.
- **Spring Boot**: Framework de Java que simplifica la creación de aplicaciones independientes y listas para producción.



¡Gracias!

¿Dudas o comentarios?

