Oracle Database 23ai + ORDS con Docker Compose

Stack completo de Oracle Database 23ai Free con Oracle REST Data Services (ORDS) completamente automatizado y listo para desarrollo.

🎯 ¿Qué incluye este proyecto?

Este stack proporciona un entorno de desarrollo completo con Oracle Database que se configura automáticamente:

- ✅ **Oracle Database 23ai Free** - La última versión de Oracle para desarrollo
- ✅ **Oracle ORDS** - REST Data Services para APIs y SQL Developer Web
- ✅ **SQL Developer Web** - Interfaz web para trabajar con la base de datos
- ✅ **Usuario de desarrollo** - Creado automáticamente con permisos completos
- ✅ **Configuración persistente** - Tus datos sobreviven a los reinicios
- ✅ **Auto-reparación** - Corrige automáticamente problemas de autenticación

🚀 Inicio rápido

Paso 1: Iniciar el stack

```
docker compose up -d
```

Eso es todo. El sistema hace el resto automáticamente.

Paso 2: Esperar a que esté listo

- **Primera vez**: 2-3 minutos (inicializa la base de datos)
- **Siguientes veces**: 30-60 segundos

Verifica que esté listo:

```
docker compose ps
```

Deberías ver ambos contenedores `Up` y oracle-db como `healthy`.

Paso 3: Acceder

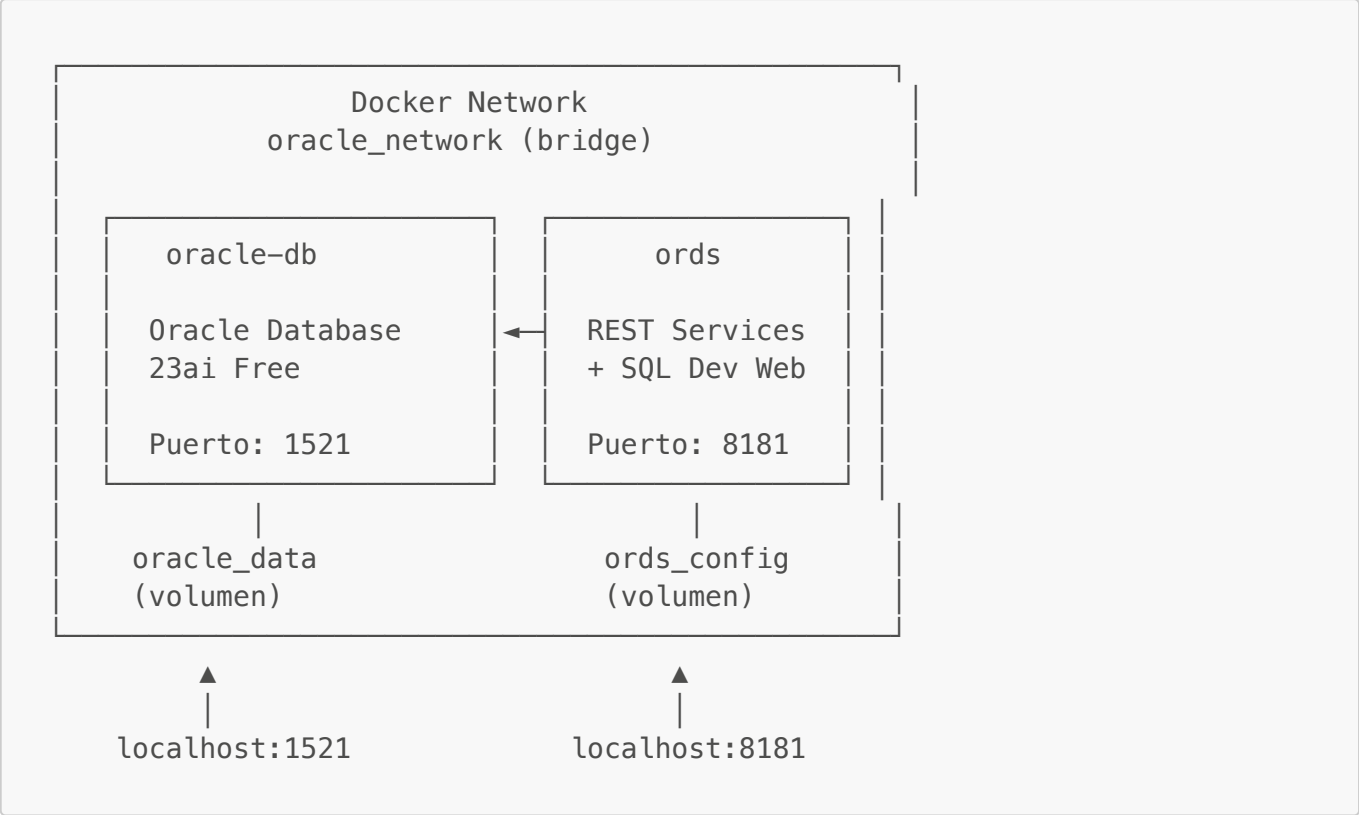
SQL Developer Web: http://localhost:8181/ords/sql-developer

```
Usuario: admin
Contraseña: Admin_123
```

¡Ya puedes empezar a trabajar! 🎉

📚 Entendiendo cómo funciona

Arquitectura del stack



Componentes principales

1. Oracle Database (oracle-db)

¿Qué es?: El motor de base de datos Oracle 23ai Free

Características:

- Versión: Oracle AI Database 26ai Free Release 23.26.0.0.0
- Arquitectura: ARM64 (optimizado para Apple Silicon)
- Contiene: 1 CDB (FREE) + 1 PDB (FREEPDB1)

Volúmenes:

- `oracle_data`: Almacena los datos de la base de datos (persiste entre reinicios)
- `./scripts/startup`: Scripts que se ejecutan en cada inicio

2. Oracle ORDS (ords)

¿Qué es?: Oracle REST Data Services - Capa de servicios REST sobre la base de datos

Proporciona:

- API REST para acceso a la base de datos

- SQL Developer Web (interfaz web tipo SQL*Plus)
- Soporte para crear APIs REST personalizadas

Volúmenes:

- `ords_config`: Configuración de ORDS (persiste entre reinicios)

🔄 Proceso de arranque explicado

¿Qué sucede cuando ejecutas `docker compose up`?

Fase 1: Preparación de la red (0-2 segundos)

Docker crea una red privada donde los contenedores se comunican:

```
✅ Network oracle_network created
```

Los contenedores pueden comunicarse usando sus nombres (`oracle-db`, `ords`).

Fase 2: Inicio de Oracle Database (0-180 segundos)

1. **Docker inicia el contenedor oracle-db**

   ```
   Container oracle-db started
   ```

2. **Oracle ejecuta su inicialización interna**

   - Si es la primera vez: Crea la base de datos desde cero (2-3 minutos)
   - Si ya existe: Arranca la base de datos existente (20-30 segundos)

3. **Se ejecuta el healthcheck cada 30 segundos**

   ```
   # El healthcheck verifica:
   lsnrctl status | grep -q 'READY'  # ¿Listener activo?
   sqlplus / as sysdba << SQL         # ¿Base de datos responde?
      SELECT 1 FROM dual;
   SQL
   ```

4. **Cuando Oracle está lista, se ejecuta el script de configuración**

   Archivo: `scripts/startup/01-configure-password.sh`

   El script hace:

```
⏳ Esperando a que Oracle esté lista...
✅ Oracle está lista

🔑 Configurando contraseña de SYS...
   → ALTER USER SYS IDENTIFIED BY Welcome_123

📝 Recreando archivo de contraseñas...
   → orapwd file=.../orapwFREE password=Welcome_123

📊 Estado de las PDBs:
   PDB$SEED → READ ONLY
   FREEPDB1 → READ WRITE ✅

👤 Configurando usuario ADMIN...
   → CREATE USER admin IDENTIFIED BY Admin_123
   → GRANT CONNECT, RESOURCE TO admin
   → GRANT CREATE TABLE, VIEW, PROCEDURE... TO admin
   → ORDS_ADMIN.ENABLE_SCHEMA(schema => 'ADMIN')

✅ Usuario ADMIN configurado

📌 Acceso a SQL Developer Web:
   URL: http://localhost:8181/ords/sql-developer
   Usuario: admin
   Contraseña: Admin_123
```

5. **Oracle marca el contenedor como "healthy"**

```
Container oracle-db is healthy
```

Fase 3: Inicio de ORDS (depende de Oracle)

ORDS espera a que Oracle esté `healthy` antes de iniciar (gracias a `depends_on`).

1. **ORDS inicia y lee las variables de entorno**

```
DBHOST: oracle-db
DBPORT: 1521
DBSERVICENAME: freepdb1  # ¡Importante! Conecta a la PDB, no al CDB
ORACLE_PWD: Welcome_123
```

2. **ORDS intenta conectarse a la base de datos**

```
Testing database connection...
Attempt 1: Connecting to sys/*****@oracle-db:1521/freepdb1 as
sysdba...
```

3. **Si la conexión es exitosa, ORDS instala sus objetos**

   Primera vez: Crea esquemas ORDS_METADATA y ORDS_PUBLIC_USER

   Siguientes veces: Verifica que todo esté correcto

4. **ORDS inicia el servidor Jetty**

```
Oracle REST Data Services initialized
Oracle REST Data Services version : 25.4.0.r3641739

Mapped local pools from /etc/ords/config/databases:
  /ords/ => default => VALID ✅
```

5. **ORDS está listo para recibir peticiones**

```
Container ords is up
```

Fase 4: Todo listo (1-3 minutos)

```
NAME        STATUS                  PORTS
oracle-db   Up (healthy)            0.0.0.0:1521->1521/tcp
ords        Up                      0.0.0.0:8181->8080/tcp
```

🔑 Credenciales y accesos

Usuario SYS (Administrador de Base de Datos)

Uso: Administración de la base de datos, tareas DBA

```
Usuario: sys
Contraseña: Welcome_123
Rol: SYSDBA
Base de datos: FREE (CDB) o FREEPDB1 (PDB)
```

Conexión:

```
# Desde tu equipo (requiere Oracle Client)
sqlplus sys/Welcome_123@localhost:1521/freepdb1 as sysdba
```

```
# Desde el contenedor (sin contraseña por ser local)
docker exec -it oracle-db sqlplus / as sysdba
```

Usuario ADMIN (Desarrollo y SQL Developer Web)

Uso: Desarrollo de aplicaciones, SQL Developer Web

```
Usuario: admin
Contraseña: Admin_123
Esquema: ADMIN
Base de datos: FREEPDB1
```

Creación automática: Se crea cada vez que inicia Oracle si no existe

Permisos otorgados:

- `CONNECT`, `RESOURCE`: Roles básicos de desarrollo
- `CREATE TABLE`, `CREATE VIEW`, `CREATE PROCEDURE`: Crear objetos
- `CREATE SEQUENCE`, `CREATE TRIGGER`: Objetos avanzados
- `UNLIMITED TABLESPACE`: Sin límite de espacio
- REST habilitado: Puede exponer objetos vía API

Conexión:

```
# Desde tu equipo
sqlplus admin/Admin_123@localhost:1521/freepdb1

# SQL Developer Web (navegador)
http://localhost:8181/ords/sql-developer
```

Endpoints ORDS

```
Servicio                    URL
─────────────────────────────────────────────────────────
Landing page                http://localhost:8181/ords/_/landing
SQL Developer Web           http://localhost:8181/ords/sql-developer
REST API (esquema ADMIN)    http://localhost:8181/ords/admin/
Documentación API           http://localhost:8181/ords/admin/metadata-
catalog/
```

🔧 Comandos útiles

 Comandos útiles

Gestión básica

```
# Iniciar todo
docker compose up -d

# Ver estado de servicios
docker compose ps

# Detener servicios (mantiene datos)
docker compose down

# Reiniciar un servicio específico
docker compose restart oracle-db
docker compose restart ords

# Reiniciar todo
docker compose restart

# ⚠ Eliminar TODO (incluyendo datos)
docker compose down -v
```

Monitoreo y logs

```
# Ver logs de todos los servicios
docker compose logs

# Seguir logs en tiempo real
docker compose logs -f

# Ver solo logs de Oracle
docker compose logs oracle-db

# Ver solo logs de ORDS
docker compose logs ords

# Ver el proceso de configuración automática
docker compose logs oracle-db | grep "Configurando"

# Ver últimas 50 líneas
docker compose logs --tail 50
```

Acceso a los contenedores

```
# Entrar a una sesión bash en Oracle
docker exec -it oracle-db bash

# Ejecutar SQL*Plus directamente
docker exec -it oracle-db sqlplus / as sysdba
```

```
# Ejecutar SQL*Plus como ADMIN
docker exec -it oracle-db sqlplus admin/Admin_123@localhost:1521/freepdb1

# Ver variables de entorno de ORDS
docker exec ords env | grep -E "(DB|ORACLE)"

# Verificar archivos de configuración de ORDS
docker exec ords ls -la /etc/ords/config/databases/
```

Verificación de conectividad

```
# Probar que Oracle responde
docker exec oracle-db sqlplus -s / as sysdba <<< "SELECT 'OK' FROM dual;"

# Ver estado de PDBs
docker exec oracle-db sqlplus -s / as sysdba <<< "SELECT name, open_mode
FROM v\$pdbs;"

# Probar conexión a ORDS via HTTP
curl http://localhost:8181/ords/

# Ver versión de ORDS
curl -s http://localhost:8181/ords/ | grep -i "version"
```

🛠️ Entendiendo la configuración automática

El script mágico: `01-configure-password.sh`

Este script se encuentra en `scripts/startup/` y es montado en el contenedor Oracle en `/opt/oracle/scripts/startup/`. Oracle ejecuta automáticamente todos los scripts `.sh` en esa ruta **cada vez que el contenedor arranca**.

¿Por qué existe este script?

Problema original:

- Oracle establece la contraseña solo en la primera inicialización
- Si el volumen persiste y reinicias con una contraseña diferente, no coincide
- ORDS no puede conectarse porque usa la nueva contraseña del `docker-compose.yaml`

Solución:

- Script que sincroniza la contraseña en cada arranque
- Crea el usuario ADMIN automáticamente
- Habilita REST services

¿Qué hace exactamente?

1. Espera a que Oracle esté lista (máximo 5 minutos)

```
until sqlplus -s / as sysdba <<< "SELECT 1 FROM dual;" > /dev/null; do
    sleep 5
done
```

2. Sincroniza la contraseña de SYS

```
ALTER USER SYS IDENTIFIED BY ${ORACLE_PASSWORD};
```

Lee `ORACLE_PASSWORD` del `docker-compose.yaml` y actualiza la contraseña.

3. Recrea el archivo de contraseñas de Oracle

```
orapwd file=${ORACLE_HOME}/dbs/orapwFREE \
       password="${ORACLE_PASSWORD}" \
       entries=10 \
       force=yes
```

Oracle necesita este archivo para autenticación remota como SYSDBA.

4. Verifica que las PDBs estén abiertas

```
SELECT name, open_mode FROM v$pdbs;
```

ORDS necesita conectarse a `FREEPDB1` (la PDB), no al CDB.

5. Crea el usuario ADMIN (si no existe)

```
-- Conectar a la PDB
ALTER SESSION SET CONTAINER = FREEPDB1;

-- Verificar si existe
SELECT COUNT(*) FROM dba_users WHERE username = 'ADMIN';

-- Si no existe, crear
CREATE USER admin IDENTIFIED BY Admin_123
  DEFAULT TABLESPACE users
  QUOTA UNLIMITED ON users;

-- Otorgar permisos de desarrollo
GRANT CONNECT, RESOURCE TO admin;
GRANT CREATE SESSION, CREATE TABLE, CREATE VIEW TO admin;
-- ... más permisos
```

6. Habilita el esquema ADMIN en ORDS

```
BEGIN
  ORDS_ADMIN.ENABLE_SCHEMA(
    p_enabled => TRUE,
    p_schema => 'ADMIN',
    p_url_mapping_type => 'BASE_PATH',
    p_url_mapping_pattern => 'admin',
    p_auto_rest_auth => FALSE
  );
END;
```

Esto permite que el esquema ADMIN sea accesible vía REST API.

Idempotencia del script

El script es **idempotente**: puede ejecutarse múltiples veces sin problemas.

- Si SYS ya tiene la contraseña correcta → No problem, la establece de nuevo
- Si el usuario ADMIN ya existe → Lo detecta y no lo recrea
- Si el esquema ya está habilitado en ORDS → No genera error

Esto significa que **no importa cuántas veces reinicies**, siempre funciona.

📂 Estructura del proyecto

```
.
├── docker-compose.yaml              # Configuración principal del stack
├── scripts/
│   └── startup/
│       └── 01-configure-password.sh # Script de auto-configuración
├── README.md                         # Esta documentación
└── QUICKSTART.md                    # Guía rápida de inicio

Volúmenes Docker (creados automáticamente):
├── oracle_data/                     # Datos de la base de datos
(persistente)
└── ords_config/                     # Configuración de ORDS (persistente)
```

Archivo: `docker-compose.yaml`

Servicio: oracle-db

```
oracle-db:
  image: container-registry.oracle.com/database/free:23.26.0.0-arm64
```

```
  ports:
    - "1521:1521"              # Puerto SQL*Net
  environment:
    - ORACLE_PASSWORD=Welcome_123    # Password inicial (sincronizada por
script)
  volumes:
    - oracle_data:/opt/oracle/oradata             # Datos persisten aquí
    - ./scripts/startup:/opt/oracle/scripts/startup # Scripts de
inicialización
  healthcheck:
    test: ["CMD-SHELL",
           "lsnrctl status | grep -q 'READY' &&
            echo 'SELECT 1 FROM dual;' | sqlplus -s / as sysdba"]
    interval: 30s              # Verifica cada 30 segundos
    timeout: 10s               # Timeout de 10 segundos
    retries: 15                # Hasta 15 reintentos (7.5 minutos)
    start_period: 60s          # Espera 60 segundos antes del primer check
```

¿Por qué estos valores de healthcheck?

- Oracle tarda en arrancar (especialmente la primera vez)
- El healthcheck verifica que el listener Y la base de datos respondan
- ORDS no intenta conectar hasta que este healthcheck pase

Servicio: ords

```
ords:
  image: container-registry.oracle.com/database/ords:latest
  ports:
    - "8181:8080"              # ORDS escucha en 8080, mapeado a 8181 en
el host
  environment:
    - DBHOST=oracle-db         # Nombre del contenedor (resuelto por
Docker)
    - DBPORT=1521
    - DBSERVICENAME=freepdb1   # ⚠ IMPORTANTE: PDB, no CDB
    - ORACLE_PWD=Welcome_123   # Misma contraseña que Oracle
    - DB_WAIT_RETRY=120        # Espera hasta 20 minutos para conectar
  depends_on:
    oracle-db:
      condition: service_healthy  # Solo inicia si Oracle está "healthy"
  volumes:
    - ords_config:/etc/ords/config    # Config persiste entre reinicios
  restart: on-failure          # Se reinicia automáticamente si falla
```

¿Por qué FREEPDB1 y no FREE?

- `FREE` = Container Database (CDB) - Base de datos raíz
- `FREEPDB1` = Pluggable Database (PDB) - Base de datos de aplicación
- ORDS debe instalarse en una PDB, no en el CDB

- Los usuarios de aplicación (como ADMIN) viven en la PDB

🔍 Diagnóstico y troubleshooting

Verificar que todo está OK

1. Verificar estado de contenedores

```
docker compose ps
```

Salida esperada:

```
NAME         IMAGE                                  STATUS
PORTS
oracle-db    oracle.../database/free:23.26...       Up (healthy)
0.0.0.0:1521->1521/tcp
ords         oracle.../database/ords:latest         Up
0.0.0.0:8181->8080/tcp
```

✅ `oracle-db` debe estar **Up** y **(healthy)**
✅ `ords` debe estar **Up**

2. Verificar logs de configuración

```
docker compose logs oracle-db | grep -A 10 "Configurando Oracle"
```

Salida esperada:

```
🔧 Configurando Oracle Database...
⏳ Esperando a que Oracle esté lista...
✅ Oracle está lista
🔑 Configurando contraseña de SYS...
📝 Recreando archivo de contraseñas...
✅ Conexión verificada exitosamente
👤 Configurando usuario ADMIN...
✅ Usuario ADMIN configurado
```

3. Verificar que ORDS conectó

```
docker compose logs ords | grep -i "initialized\|valid"
```

Salida esperada:

```
Oracle REST Data Services initialized
Mapped local pools: /ords/ => default => VALID
```

4. Probar SQL Developer Web

```
curl -I http://localhost:8181/ords/sql-developer
```

Salida esperada: HTTP/1.1 200 OK o 302 Found

Problemas comunes y soluciones

❌ Oracle no arranca (stuck en "starting")

Síntomas:

```
docker compose ps
# oracle-db   Up (starting)   # Se queda así por mucho tiempo
```

Diagnóstico:

```
docker compose logs oracle-db | tail -50
```

Posibles causas:

1. **Primera inicialización** → Es normal, espera 3-5 minutos
2. **Falta espacio en disco** → Libera espacio
3. **Corrupción de datos** → Recrea: `docker compose down -v && docker compose up -d`

❌ ORDS muestra "Database not ready"

Síntomas:

```
docker compose logs ords
# INFO: Database not ready (attempt X of 120). Retrying in 10s...
```

Diagnóstico:

```
# 1. Verificar que Oracle esté healthy
docker compose ps

# 2. Verificar que el script de configuración se ejecutó
docker compose logs oracle-db | grep "Usuario ADMIN"

# 3. Verificar contraseña manualmente
docker exec oracle-db bash -c \
  "echo 'SELECT 1 FROM dual;' | sqlplus -s
sys/Welcome_123@localhost:1521/freepdb1 as sysdba"
```

Soluciones:

1. **Oracle no está healthy todavía** → Espera más
2. **Script no se ejecutó** → Reinicia Oracle: `docker compose restart oracle-db`
3. **Contraseña incorrecta** → Verifica `docker-compose.yaml` y reinicia
4. **PDB no está abierta** →

```
docker exec oracle-db sqlplus / as sysdba <<EOF
ALTER SESSION SET CONTAINER = FREEPDB1;
ALTER DATABASE OPEN;
EOF
```

❌ "Invalid credentials" en SQL Developer Web

Síntomas: La página carga, pero al poner admin/Admin_123 dice credenciales inválidas.

Diagnóstico:

```
# Verificar que el usuario existe
docker exec oracle-db bash -c \
  "echo 'ALTER SESSION SET CONTAINER=FREEPDB1; SELECT username FROM
dba_users WHERE username='\''ADMIN'\'';' | \
   sqlplus -s / as sysdba"
```

Soluciones:

1. **Usuario no existe** → Reinicia Oracle para que el script lo cree:

```
docker compose restart oracle-db
```

2. **Contraseña incorrecta** → Restablece la contraseña:

```
```

```
docker exec oracle-db sqlplus / as sysdba <<EOF
ALTER SESSION SET CONTAINER = FREEPDB1;
ALTER USER admin IDENTIFIED BY Admin_123;
EXIT;
EOF
```

3. **Esquema no habilitado en ORDS** → Reinicia ORDS:

```
docker compose restart ords
```

❌ Puerto 1521 o 8181 ya en uso

Síntomas:

```
Error: bind: address already in use
```

Solución 1: Detén el proceso que usa el puerto

```
# En macOS/Linux
lsof -i :1521
lsof -i :8181

# Mata el proceso (busca el PID)
kill <PID>
```

Solución 2: Cambia los puertos en `docker-compose.yaml`

```
ports:
  - "1522:1521"  # Usa 1522 en lugar de 1521
  - "8182:8080"  # Usa 8182 en lugar de 8181
```

❌ Contenedor ORDS reiniciando constantemente

Síntomas:

```
docker compose ps
# ords   Restarting
```

Diagnóstico:

```
docker compose logs ords | tail -100
```

Causa común: Variables de entorno incorrectas

Solución:

```
# Verificar variables
docker exec ords env | grep -E "(DB|ORACLE)"

# Deben coincidir con docker-compose.yaml
# Si no coinciden, recrea el contenedor:
docker compose up -d --force-recreate ords
```

🎓 Conceptos clave de Oracle

CDB vs PDB: ¿Qué son y por qué importan?

Oracle usa una arquitectura **multitenant**:

```
FREE (CDB - Container Database)              ← Base de datos raíz
├── PDB$SEED (PDB - Pluggable Database)      ← Plantilla (solo lectura)
└── FREEPDB1 (PDB - Pluggable Database)      ← Tu base de datos de
aplicación
```

CDB (Container Database):

- La base de datos "contenedor"
- Almacena usuarios y configuración del sistema
- Usuario principal: SYS

PDB (Pluggable Database):

- Base de datos "enchufable"
- Donde viven tus aplicaciones y datos
- Puede ser movida, clonada, respaldada independientemente

¿Por qué ORDS conecta a FREEPDB1?

- Los usuarios de aplicación (como ADMIN) deben estar en una PDB
- ORDS es una aplicación, no una herramienta de administración
- Las APIs REST y SQL Developer Web operan en la PDB

Autenticación en Oracle

Autenticación local (dentro del contenedor):

```
sqlplus / as sysdba
```

-Sin contraseña porque el usuario del OS está en el grupo `dba`

Autenticación remota:

```
sqlplus sys/Welcome_123@localhost:1521/freepdb1 as sysdba
```

- Requiere contraseña
- Necesita el archivo de contraseñas (`orapwd`)
- Por eso el script recrea este archivo

Healthcheck: ¿Cómo sabe Docker que Oracle está lista?

El healthcheck ejecuta este comando cada 30 segundos:

```
# 1. Verificar que el listener acepta conexiones
lsnrctl status | grep -q 'READY'

# 2. Verificar que la base de datos responde
echo 'SELECT 1 FROM dual;' | sqlplus -s / as sysdba
```

Solo cuando **ambos** tienen éxito, Oracle se marca como `healthy`.

📝 Notas importantes

Persistencia de datos

Los volúmenes Docker garantizan que tus datos persistan:

```
# Ver volúmenes
docker volume ls | grep oracle

# Inspeccionar un volumen
docker volume inspect 2026_oracle_restfull_ords_oracle_data

# ⚠ Para eliminar datos (no reversible):
docker compose down -v
```

¿Qué se guarda?

- `oracle_data`: Todos los datos de tablas, índices, procedimientos
- `ords_config`: Configuración de ORDS, pools de conexión, mappings

Contraseñas en producción

Este stack usa contraseñas hardcodeadas para facilitar el desarrollo.

Para producción, usa:

- Variables de entorno externas
- Docker secrets
- Vault u otro gestor de secretos

```
# Ejemplo con archivo .env
echo "ORACLE_PASSWORD=MiPasswordSeguro123#" > .env

# En docker-compose.yaml
environment:
  - ORACLE_PASSWORD=${ORACLE_PASSWORD}
```

Requisitos de sistema

Mínimos recomendados:

- **CPU**: 2 cores
- **RAM**: 4 GB (8 GB recomendado)
- **Disco**: 10 GB libres
- **Docker**: 20.10+
- **Docker Compose**: 2.0+

Oracle Database puede usar bastante memoria. Ajusta los límites en `docker-compose.yaml` si es necesario:

```
deploy:
  resources:
    limits:
      memory: 4G
    reservations:
      memory: 2G
```

Arquitectura ARM64 vs AMD64

Esta configuración usa la imagen ARM64 (Apple Silicon):

```
image: container-registry.oracle.com/database/free:23.26.0.0-arm64
```

Para Intel/AMD:

```
image: container-registry.oracle.com/database/free:latest
```

📚 Referencias y recursos

Documentación oficial

- Oracle Database 23ai Free
- Oracle ORDS Documentation
- SQL Developer Web Guide