

Setting up Oracle 19c in Docker can be done using Oracle's official Docker build scripts, but due to Oracle's licensing, **Oracle does not publish official Docker images on Docker Hub**. Instead, you must **manually download the Oracle 19c installation files** and **build the image locally**.

Here's a full guide with a Dockerfile to set up Oracle Database 19c **Enterprise Edition** using Oracle's official scripts.

Prerequisites

1. **Download the Oracle 19c installation file** manually from:

👉 <https://www.oracle.com/database/technologies/oracle19c-linux-downloads.html>

- File needed (for Linux x86_64):
LINUX.X64_193000_db_home.zip

2. **Install Docker** on your system.
-

Why This Is Required

Oracle does **not allow direct Docker pulls** of its database images due to licensing restrictions. This process ensures you **accept the license** before using the software.

Step-by-step Setup

- **Clone Oracle's Docker GitHub repo**

```
git clone https://github.com/oracle/docker-images.git
cd docker-images/OracleDatabase/SingleInstance/dockerfiles
```

- **Place the ZIP file in the 19.3.0 directory**

```
mv ~/Downloads/LINUX.X64_193000_db_home.zip 19.3.0/
```

- **Build the image**

```
./buildContainerImage.sh -v 19.3.0 -s
```

- -v: version
- -s: software-only (optional, for INSTALL_DB_SWONLY)

- **Use the image locally**

Once built, Docker will tag it as:

```
oracle/database:19.3.0-se2
```

```

=====
Building image 'oracle/database:19.3.0-se2' ...
[+] Building 187.5s (16/16) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 5.06kB
=> WARN: FromAsCasing: 'as' and 'FROM' keywords' casing do not match (line 25)
=> [internal] load metadata for docker.io/library/oraclelinux:7-slim
=> [auth] library/oraclelinux:pull token for registry-1.docker.io
=> [internal] load .dockerignore
=> => transferring context: 2B
=> CACHED [base 1/4] FROM docker.io/library/oraclelinux:7-slim@sha256:1add6ed8602ea996528110fe75f4b03c2ca7ffdbe9497148
=> [internal] load build context
=> => transferring context: 3.06GB
=> [base 2/4] COPY setupLinuxEnv.sh checkSpace.sh /opt/install/
=> [base 3/4] COPY runOracle.sh startDB.sh createDB.sh createObserver.sh dbca.rsp.tmpl setPassword.sh checkDBStatus.sh
=> [base 4/4] RUN chmod ug+x /opt/install/*.sh && sync && /opt/install/checkSpace.sh && /opt/install/setup
=> [builder 1/2] COPY --chown=oracle:dba LINUX.X64_193000_db_home.zip db_inst.rsp installDBBinaries.sh /opt/install/
=> [builder 2/2] RUN chmod ug+x "/opt/install/*.sh && sync && "/opt/install"/"installDBBinaries.sh" se2
=> [stage-2 1/4] COPY --chown=oracle:dba --from=builder /opt/oracle /opt/oracle
=> [stage-2 2/4] RUN "/opt/oracle/oraInventory/orainstRoot.sh && "/opt/oracle/product/19c/dbhome_1"/root.sh
=> [stage-2 3/4] WORKDIR /home/oracle
=> [stage-2 4/4] RUN echo 'ORACLE_SID=${ORACLE_SID:-ORCLCDB}; export ORACLE_SID=${ORACLE_SID^^}' > .bashrc
=> exporting to image
=> => exporting layers
=> => writing image sha256:28ed501595cf02e875d2ccbf245f9d0fb78df586e8d78ebee9e2f977ddc6528a
=> => naming to docker.io/oracle/database:19.3.0-se2

Oracle Database container image for 'se2' version 19.3.0 is ready to be extended:

--> oracle/database:19.3.0-se2

Build completed in 188 seconds.

```

You can now use that in your Dockerfiles:

FROM oracle/database:19.3.0-se2



Use the created docker image

Great! You can fully integrate this into the Dockerfile by using a **.env file in combination with a docker-compose.yml file**—because a Dockerfile alone can't run `docker run` or define container runtime configuration like ports and environment variables.

Here's how to achieve your goal **step by step**:



Step 1: .env File

Create a file named `.env` in the same directory:

```

# .env
ORACLE_SID=ORCLCDB
ORACLE_PDB=tordb1
ORACLE_PWD=pwdtrainer
ORACLE_CHARACTERSET=AL32UTF8

```



Step 2: Create init-user.sql

```
-- init-user.sql

-- Creates user and grants privileges on first run
ALTER SESSION SET "_ORACLE_SCRIPT"=true;

CREATE USER zeus IDENTIFIED BY pwdtrainer;
GRANT CONNECT, RESOURCE, DBA TO zeus;
```



Step 3: Dockerfile (already correct)

Your Dockerfile should already include the following to copy your init SQL:

```
# Base image from Oracle
FROM oracle/database:19.3.0-se2

# Switch to root to fix permissions
USER root

# Ensure required directories exist with correct permissions
RUN mkdir -p /opt/oracle/product/19c/dbhome_1/network/log && \
    mkdir -p /opt/oracle/oradata && \
    mkdir -p /opt/oracle/cfgtoollogs && \
    chown -R oracle:oinstall /opt/oracle && \
    chmod -R 777 /opt/oracle

# Copy the startup SQL script to create user and grant privileges
COPY init-user.sql /opt/oracle/scripts/startup/

# Set correct permissions for startup script
RUN chown oracle:oinstall /opt/oracle/scripts/startup/init-user.sql && \
    chmod 775 /opt/oracle/scripts/startup/init-user.sql

# Expose ports for Oracle DB and EM Express
EXPOSE 1521 5500

# Define mountable volume
VOLUME ["/opt/oracle/oradata"]

# Switch back to oracle user
USER oracle

# Default command (will auto-run DBCA at first startup if DB not initialized)
CMD ["/bin/bash", "-c", "/opt/oracle/runOracle.sh"]
```



Step 4: docker-compose.yml

version: '3.8'

services:

oracle-db:

build:

context: .

dockerfile: Dockerfile

container_name: tordb1-container

ports:

- "1521:1521"

- "5500:5500"

env_file:

- .env

environment:

ORACLE_SID: \${ORACLE_SID}

ORACLE_PDB: \${ORACLE_PDB}

ORACLE_PWD: \${ORACLE_PWD}

ORACLE_CHARACTERSET: \${ORACLE_CHARACTERSET}

volumes:

- oracle-data:/opt/oracle/oradata

volumes:

oracle-data:



Step 5: Run Everything

Now run the following in the same folder as the Dockerfile and .env:

```
# adjust permissions
```

```
sudo chown -R 54321:54321 ./oracle-data
```

```
sudo chmod -R 775 ./oracle-data
```

#run the docker image

docker compose up --build -d

✓ Result

- Your database container (tordb1-container) will be started
- Environment variables will be taken from .env
- The database will include the PDB tordb1
- The user zeus with password pwdtrainer will be created

```
soahn@soahn-Vostro-3500:~/Desktop/Spring Boot Formation/Part-1/docker-oracle/oracle-19c$ docker compose up --build -d
WARN[0000] /home/soahn/Desktop/Spring Boot Formation/Part-1/docker-oracle/oracle-19c/docker-compose.yml: the attribute 'version' is obsolete, it will be ignored, please remove it to avoid potential confusion
[+] Building 43.7s (18/18) FINISHED
=> [oracle-db internal] load build definition from Dockerfile
=> transferring dockerfile: 924B
=> [oracle-db internal] load metadata for docker.io/oracle/database:19.3.0-se2
=> [oracle-db internal] load .dockerignore
=> transferring context: 2B
=> CACHED [oracle-db 1/4] FROM docker.io/oracle/database:19.3.0-se2
=> [oracle-db internal] load build context
=> transferring context: 34B
=> [oracle-db 2/4] RUN mkdir -p /opt/oracle/product/19c/dbhome_1/network/log && mkdir -p /opt/oracle/oradata && mkdir -p /opt/oracle/cfgtoollogs && chown -R oracle:install /opt/oracle && chmod -R 777 /opt/oracle
=> [oracle-db 3/4] COPY init-user.sql /opt/oracle/scripts/startup/
=> [oracle-db 4/4] RUN chown oracle:install /opt/oracle/scripts/startup/init-user.sql && chmod 775 /opt/oracle/scripts/startup/init-user.sql
=> [oracle-db] exporting to image
=> exporting layers
=> writing image sha256:9c337a117af8b84045a17c1ecacaddf77e618369fac2f4d36e6ee83ee5aa234f
=> naming to docker.io/library/oracle-19c-oracle-db
=> [oracle-db] resolving provenance for metadata file
[+] Running 3/3
✔ oracle-db Built
✔ Network oracle-19c_default Created
✔ Container tordb1-container Started
soahn@soahn-Vostro-3500:~/Desktop/Spring Boot Formation/Part-1/docker-oracle/oracle-19c$
```