How to run Microsoft SQL (Database Engine) on a Mac with Apple silicon chip using Docker Desktop

- Parallels Desktop for Mac Standard Edition
- Parallels Desktop for Mac Pro Edition
- Parallels Desktop for Mac Business Edition

As a developer or a knowledge worker, you might need a local database to work with. However, many databases are based on x86_64 architecture, which means they might not run on the latest Mac with Apple silicon; this is the case with Microsoft SQL where it will not yet run in Windows 11 on Mac with Apple silicon. In this article, we will explore how to run Microsoft SQL, one of the most common database engines, on the newest highly performant machines using Docker Desktop. This will only be the SQL server and none of the other tools that come with the Microsoft SQL Server.

Prerequisites

- Mac with Apple silicon
- Docker Desktop

Note: These instructions are based on Docker Desktop v.4.1.7 installed on macOS 13.3.1, but they should work on previous versions if Rosetta is enabled and available in Docker. At the moment of writing, this feature was still in beta.

Steps

Step 1. Install Docker Desktop

Follow the instructions using the Apple Silicon version.

Step 2. Configure Docker Desktop to run using Rosetta

Go to Settings > Features in Development, and select the “Use Rosetta for x86/amd64 emulation on Apple silicon” option.

Step 3. Create a container using Microsoft SQL for Linux

To start the container, run the following command in Terminal (Finder > Applications > Utilities > Terminal):

```
docker run -e "ACCEPT_EULA=Y" -e "MSSQL_SA_PASSWORD=VeryStr0ngP@ssw0rd" -p
1433:1433 --name sql --hostname sql --platform linux/amd64 -d
mcr.microsoft.com/mssql/server:2022-latest
```

This command creates a new container and forces the platform to be amd64. Docker uses the Rosetta library to emulate the x86 and run the SQL server. After this step, you can access the SQL server using any client, including from within a VM like the ones Parallels makes available to you.

Note: if the container starts and immediately stops, check if your password is complex enough or if the Rosetta setting is indeed enabled.

If you want to persist the data even if the container is deleted, add a volume mount. This allows you to store the data locally on your macOS and create or delete containers as needed. To do this, add the following just after the platform flag:

```
docker run -e "ACCEPT_EULA=Y" -e "MSSQL_SA_PASSWORD=VeryStr0ngP@ssw0rd" -p
1433:1433 --name sql --hostname sql --platform linux/amd64 -v
/sql_server:/var/opt/mssql -d mcr.microsoft.com/mssql/server:2022-latest
```

You can set the folder where you want to store the data by changing the initial section of the mount.

Alternatively, you can automate this process by using Docker Compose and creating a run file with all of these instructions. Here's an example:

```
version: '3.7
name: sql_server
services:
  server:
    image: mcr.microsoft.com/mssql/server:2022-latest
    volumes:
      - / sql_server:/var/opt/mssql
    environment:
      ACCEPT_EULA: Y
      MSSQL_SA_PASSWORD: VeryStr0ngP@ssw0rd
    ports:
      - "1433:1433"'
```

Save this code in a file, and then run **docker-compose up** in the root directory where you saved the file. This will create the container and start it up for you.

With a few commands and the use of Docker, you can access the SQL Server from any place, including Windows virtual machines available in your Parallels Desktop. You will need your IP address, and your username/password will be *sa/VeryStr0ngP@ssw0rd* or any other that you defined in your command.

Congratulations! You have successfully set up Microsoft SQL on your Apple Silicon-based Mac using Docker Desktop. Now you can work.
