SQL Server in Docker

This is a summarization of the full article found here.

Setup

Get the SQL Server 2017 image from Docker Hub:

```
sudo docker pull mcr.microsoft.com/mssql/server:2017-latest
```

Start up the container image, and init SQL Server setup:

```
sudo docker run -e "ACCEPT_EULA=Y" -e
"MSSQL_SA_PASSWORD=<YourStrong@Passw0rd>" \
   -p 1433:1433 --name sql1 --hostname sql1 \
   -d \
   mcr.microsoft.com/mssql/server:2017-latest
```

The password must follow SQL Server default password policy, or the setup will fail: at least 8 characters long, containing characters from three of the following four sets: Uppercase letters, Lowercase letters, Base 10 digits, and Symbols.

By default, this creates a container with the Developer edition of SQL Server 2017.

Detailed description of parameters from the previous example:

Parameter	Description
-e 'ACCEPT_EULA=Y'	Set the ACCEPT_EULA variable to any value to confirm your acceptance of the End-User Licensing Agreement. Required setting for the SQL Server image.
-e 'MSSQL_SA_PASSWORD='	Specify your own strong password that is at least 8 characters and meets the SQL Server password requirements. Required setting for the SQL Server image.
-p 1433:1433	Map a TCP port on the host environment (first value) with a TCP port in the container (second value). In this example, SQL Server is listening on TCP 1433 in the container and this is exposed to the port, 1433, on the host.

Parameter	Description
–name sql1	Specify a custom name for the container rather than a randomly generated one. If you run more than one container, you cannot reuse this same name.
microsoft/mssql-server-linux:2017-latest	The SQL Server 2017 Linux container image.

Check the status of your Docker containers:

```
sudo docker ps -a
```

Change the SA password

After initial setup, the MSSQL_SA_PASSWORD environment variable you specified is discoverable by running echo $MSSQL_SA_PASSWORD in the container. For security purposes, change your SA password.

Use docker exec to run sqlcmd to change the password using Transact-SQL.

```
sudo docker exec -it sql1 /opt/mssql-tools/bin/sqlcmd \
-S localhost -U SA \
 -P "$(read -sp "Enter current SA password: "; echo "${REPLY}")" \
 -Q "ALTER LOGIN SA WITH PASSWORD=\"$(read -sp "Enter new SA password: ";
echo "${REPLY}")\""
```

Connect to SQL Server

Start an interactive bash shell inside your running container. In the following example sql1 is name specified by the –name parameter when you created the container.

```
sudo docker exec -it sql1 "bash"
```

Once inside the container, connect locally with sqlcmd. Sqlcmd is not in the path by default, so you have to specify the full path.

```
/opt/mssql-tools/bin/sqlcmd -S localhost -U SA -P '<YourNewStrong!Passw0rd>'
```

After running SqlCmd, you can do normal database stuff: create databases, query data, etc.

Connect from outside the container

You can also connect to the SQL Server instance on your Docker machine from any external Linux, Windows, or macOS tool that supports SQL connections.

The following steps use sqlcmd outside of your container to connect to SQL Server running in the container. These steps assume that you already have the SQL Server command-line tools installed

outside of your container. The same principals apply when using other tools, but the process of connecting is unique to each tool.

Find the IP address for the machine that hosts your container. On Linux, use **ifconfig** or **ip addr**. On Windows, use **ipconfig**.

Run sqlcmd specifying the IP address and the port mapped to port 1433 in your container. In this example, that is the same port, 1433, on the host machine. If you specified a different mapped port on the host machine, you would use it here.

```
sqlcmd -S <ip_address>,1433 -U SA -P '<YourNewStrong!Passw0rd>'
```

database, docker