Script to setup mssql inside docker on ubuntu

Created	@September 4, 2025 2:52 PM
Class	career ready
Last edited time	@September 4, 2025 5:29 PM

How to use

1. create a file using

```
nano mssql_docker.sh
```

Script:

```
#!/bin/bash

# ==========================
# Configuration
# ==========================
CONTAINER_NAME="mssql2019"
MSSQL_IMAGE="mcr.microsoft.com/mssql/server:2019-latest"
VOLUME_NAME="mssql2019_data"

# ==========================
# Check Docker installation
# ==========================
if ! command -v docker &> /dev/null; then
    echo "🚀 Docker not found. Installing..."
    sudo apt update
    sudo apt install -y docker.io
    sudo systemctl enable --now docker
    sudo usermod -aG docker $USER
    echo "✅ Docker installed successfully. Please log out and log back in if yo
u want to use docker without sudo."
else
    echo "✅ Docker is already installed."
fi

# ==========================
# Check Docker volume
# ==========================
if ! docker volume ls | grep -q "$VOLUME_NAME"; then
    echo "📦 Creating Docker volume: $VOLUME_NAME"
    docker volume create "$VOLUME_NAME"
else
    echo "📦 Docker volume '$VOLUME_NAME' already exists."
fi

# ==========================
# Check container
# ==========================
if docker container inspect "$CONTAINER_NAME" > /dev/null 2>&1; then
    RUNNING=$(docker inspect -f '{{.State.Running}}' "$CONTAINER_NAME")
    if [ "$RUNNING" = "true" ]; then
        echo "✅ Container '$CONTAINER_NAME' is already running."
    else
        echo "▶️ Starting existing container '$CONTAINER_NAME'..."
        docker start "$CONTAINER_NAME"
        echo "✅ Container '$CONTAINER_NAME' started."
    fi
else
    echo "🚧 Creating MSSQL container '$CONTAINER_NAME'..."
    echo "⚠️ Please set SA_PASSWORD environment variable before running t
his script."
    echo "Example: export SA_PASSWORD='YourStrong!Passw0rd'"
    if [ -z "$SA_PASSWORD" ]; then
```

```
        echo "❌ SA_PASSWORD not set. Aborting."
        exit 1
    fi
    docker run -e "ACCEPT_EULA=Y" \
            -e "SA_PASSWORD=$SA_PASSWORD" \
            -e "TZ=$(cat /etc/timezone)" \
            -p 11433:1433 \
            --name "$CONTAINER_NAME" \
            --volume "$VOLUME_NAME:/var/opt/mssql" \
            --restart always \
            -d "$MSSQL_IMAGE"
    echo "✅ Container '$CONTAINER_NAME' created and running."
fi

# =========================
# Check/install sqlcmd on host
# =========================
if ! command -v sqlcmd &> /dev/null; then
    echo "🚀 Installing sqlcmd tools on host..."
    curl https://packages.microsoft.com/keys/microsoft.asc | sudo apt-key add
-
    curl https://packages.microsoft.com/config/ubuntu/22.04/prod.list | sudo te
e /etc/apt/sources.list.d/mssql-release.list
    sudo apt update
    sudo ACCEPT_EULA=Y apt install -y mssql-tools unixodbc-dev
    echo 'export PATH="$PATH:/opt/mssql-tools/bin:$PATH"' >> ~/.bashrc
    source ~/.bashrc
    echo "✅ sqlcmd installed successfully."
else
    echo "✅ sqlcmd is already installed."
fi

# =========================
# Prompt to connect (manual)
# =========================
echo "ℹ️ To connect to SQL Server, run:"
echo "sqlcmd -S localhost,11433 -U SA -P \"<YourPassword>\""
```

2. Make it executable

```
chmod +x mssql_docker.sh
```

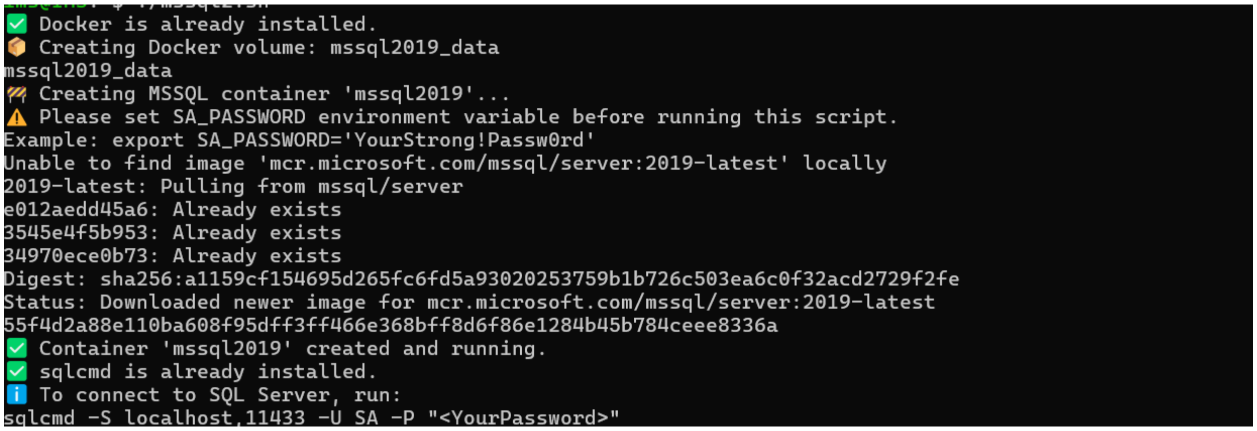
3. Set the SA password

```
export SA_PASSWORD='YourStrong!Passw0rd'
```

4. then you can run your script

```
./mssql_docker.sh
```

Output:

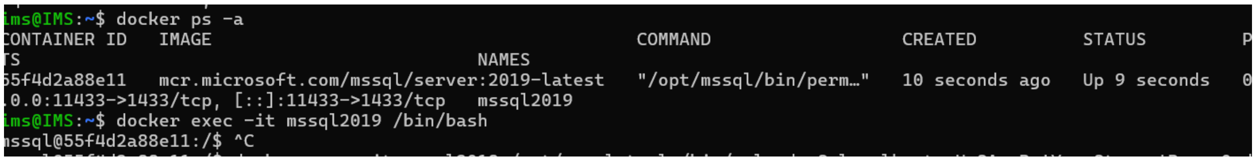


```
✅ Docker is already installed.
📦 Creating Docker volume: mssql2019_data
mssql2019_data
🚧 Creating MSSQL container 'mssql2019'...
⚠️ Please set SA_PASSWORD environment variable before running this script.
Example: export SA_PASSWORD='YourStrong!Passw0rd'
Unable to find image 'mcr.microsoft.com/mssql/server:2019-latest' locally
2019-latest: Pulling from mssql/server
e012aedd45a6: Already exists
3545e4f5b953: Already exists
34970ece0b73: Already exists
Digest: sha256:a1159cf154695d265fc6fd5a93020253759b1b726c503ea6c0f32acd2729f2fe
Status: Downloaded newer image for mcr.microsoft.com/mssql/server:2019-latest
55f4d2a88e110ba608f95dff3ff466e368bff8d6f86e1284b45b784ceee8336a
✅ Container 'mssql2019' created and running.
✅ sqlcmd is already installed.
ℹ️ To connect to SQL Server, run:
sqlcmd -S localhost,11433 -U SA -P "<YourPassword>"
```

You can see :

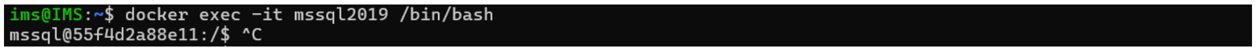
The script first checks if Docker is installed on the system and installs it if necessary. It then ensures that a Docker volume is available to persist the SQL

Server data. Next, it checks whether the container already exists; if not, it prompts the user to set the `SA_PASSWORD` environment variable before proceeding. The script then pulls the SQL Server image from Docker Hub (if not already available) and creates a new container using that image. After the container is running, it installs `sqlcmd` on the host, which is an essential tool for interacting with SQL Server. Finally, the script provides clear instructions on how to connect to the SQL Server instance using `sqlcmd`



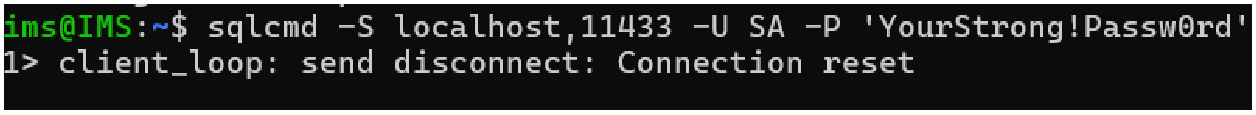
```
ims@IMS:~$ docker ps -a
CONTAINER ID   IMAGE                                        COMMAND                  CREATED          STATUS          P
TS                                             NAMES
55f4d2a88e11   mcr.microsoft.com/mssql/server:2019-latest   "/opt/mssql/bin/perm…"   10 seconds ago   Up 9 seconds    0
.0.0:11433->1433/tcp, [::]:11433->1433/tcp    mssql2019
ims@IMS:~$ docker exec -it mssql2019 /bin/bash
mssql@55f4d2a88e11:/$ ^C
```

here you can see the container is created and



```
ims@IMS:~$ docker exec -it mssql2019 /bin/bash
mssql@55f4d2a88e11:/$ ^C
```

Runs a command inside a running Docker container.



```
ims@IMS:~$ sqlcmd -S localhost,11433 -U SA -P 'YourStrong!Passw0rd'
1> client_loop: send disconnect: Connection reset
```

connect with sql server

What the Script Does (Briefly)

1. **Docker Installation Check:**
    - Checks if Docker is installed.
    - If missing, installs Docker, enables it, and adds the current user to the Docker group.
2. **Persistent Volume Setup:**
    - Creates a Docker volume (`mssql2019_data`) to store SQL Server data persistently, so data remains safe even if the container is removed.
3. **Container Management:**

    - Starts the container if it already exists.
    - Creates a new SQL Server container if it doesn't exist, applying environment variables, port mapping (`11433:1433`), timezone, volume mount, and auto-restart.
4. `sqlcmd` **Installation:**
    - Installs the SQL Server command-line tool (`sqlcmd`) on the host if not present, enabling easy interaction with the SQL Server container.
5. **Connection Instructions:**
    - Provides a simple command to connect to the SQL Server container after setup.

Benefits

- Fully automated SQL Server setup on Docker.
- Persistent data storage with Docker volumes.
- Easy connection to the database using `sqlcmd`.
- Reusable script for multiple environments or setups.