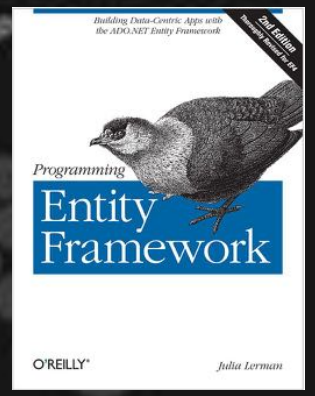
SQL Server in Containers with Docker

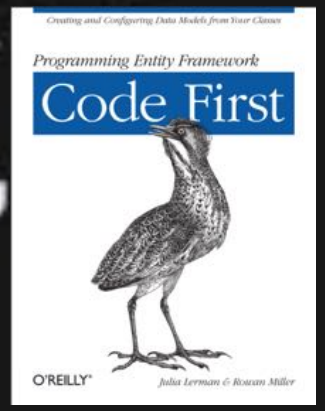
Julie Lerman TheDataFarm.com @julielerman

COURSES AUTHORED

20

All time





Julie Lerman

Pluralsight Author

✓ Following

928 Followers

Julie Lerman has been programming and mentoring developers for over 25 years and brings a wealth of experience and knowledge to share with Pluralsight subscribers. She is the leading independent authority on the Entity Framework and has been using and teaching EF since its inception.

Courses by Julie

Entity Framework Core 2: Mappings

by Julie Lerman · Intermediate · 1h 52m

Entity Framework Core 2: Getting Started

by Julie Lerman · Beginner · 2h 42m · 6 February 2018

Cross-platform SQL Server Management for Developers Using VS Code

by Julie Lerman · Intermediate · 2h 21m · 8 September 2017

GOALS

Understand what containers are

Use SQL Server containers as context for first experience

See how SQL Server in a container can be amazing for dev & test

Container

Application + resources
combined into a
self-contained execution
environment

Not a Virtual Machine

- ✓ Runs directly on OS kernel
- ✓ Does not involve Hyper-V
- ✓ Isolation allows you to run concurrent multiple instances, even different versions

Docker Engine

KüchenApplianzichtImDerContainerzeigenfeld

Dockerfile	Docker Image	Docker Container
Text file Defines image contents & runtime instrux	Binary file composed of app and dependencies	Running instance of an image
Class definition	Binary form of class	Object instance

docker build

docker run

Why?

SQL Server in a Container?

- ✓ Quickly & easily spin up a SQL Server instance for dev or test
- ✓ Pre-configure database(s) and server to share across your team
- ✓ Run different instances/versions side by side
- ✓ Create new clean instances as needed
- ✓ Save resources: shut down SQL Server without losing data*

cont….

- ✓Maximize density in test or production environments, especially in microservice architectures
- ✓Isolate and control applications in a multi-tenant infrastructure

Why?

SQL Server in a Container?

SQL Server for Linux

All SQL Server 2017 SKUs

Enterprise, Standard, Web (Cloud only), Developer, Express

Runs on

Red Hat Enterprise (RHEL), SUSE, Ubuntu, **Docker**

Licensing for SQL Server in Docker: Regardless of where you run it - VM, Docker, physical, cloud, on prem - the licensing model is the same and it depends on which edition of SQL Server you're using.

Express & Developer Editions are free.

Standard & Enterprise have a cost.

github.com/Microsoft/mssql-docker/ReadMe

SQL Server Free Images on Docker Hub

For Dev & Test Only

SQL Server 2017 Developer Edition for Linux Docker Container
(Docker for Linux, Mac or Windows)

SQL Server 2017 Developer Edition, SQL Server 2017 Express
(Docker for Windows in Windows Container mode,
runs on Windows 10 , Windows Server 2016)

SQL Server Licensed Images on Docker Store

SQL Server 2017 Developer Edition for Linux Docker Container
(Docker for Linux, Mac or Windows)
store.docker.com/images/mssql-server-linux (Requires a log in)
Configure it to run as Standard or Enterprise if you have a license
docs.microsoft.com/en-us/sql/linux/sql-server-linux-configure-docker#production

Will come on board when tooling matures (apparently we're close now)

Pulling Images

microsoft/mssql-server-linux

microsoft/mssql-server-windows-developer

microsoft/mssql-server-windows-express

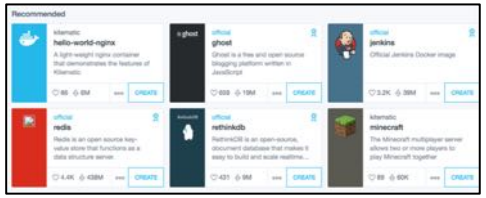
Dockerfile

```
FROM image
```

Command line

```
docker pull
```

Kitematic



etc…

```
➜  ~ docker pull microsoft/mssql-server-linux
```

```
→  ~ docker pull microsoft/mssql-server-linux
Using default tag: latest
latest: Pulling from microsoft/mssql-server-linux
aed15891ba52: Downloading  11.17MB/50.07MB
773ae8583d14: Download complete
d1d48771f782: Download complete
cd3d6cd6c0cf: Download complete
8ff6f8a9120c: Download complete
1fd7e8b10447: Downloading  11.22MB/29.16MB
bd485157db89: Downloading  7.871MB/38.8MB
273a1970ce9c: Waiting
006581b3a024: Waiting
25c54ac351f0: Waiting
```

```
→  ~ docker pull microsoft/mssql-server-linux
Using default tag: latest
latest: Pulling from microsoft/mssql-server-linux
aed15891ba52: Pull complete
773ae8583d14: Pull complete
d1d48771f782: Pull complete
cd3d6cd6c0cf: Pull complete
8ff6f8a9120c: Pull complete
1fd7e8b10447: Extracting  12.39MB/29.16MB
bd485157db89: Download complete
273a1970ce9c: Download complete
006581b3a024: Downloading   17.3MB/272.6MB
25c54ac351f0: Downloading  15.68MB/88.24MB
```

```
→  ~ docker pull microsoft/mssql-server-linux
Using default tag: latest
latest: Pulling from microsoft/mssql-server-linux
aed15891ba52: Pull complete
773ae8583d14: Pull complete
d1d48771f782: Pull complete
cd3d6cd6c0cf: Pull complete
8ff6f8a9120c: Pull complete
1fd7e8b10447: Pull complete
bd485157db89: Pull complete
273a1970ce9c: Pull complete
006581b3a024: Pull complete
25c54ac351f0: Extracting   9.47MB/88.24MB
```

```
➜  ~ docker pull microsoft/mssql-server-linux
Using default tag: latest
latest: Pulling from microsoft/mssql-server-linux
aed15891ba52: Pull complete
773ae8583d14: Pull complete
d1d48771f782: Pull complete
cd3d6cd6c0cf: Pull complete
8ff6f8a9120c: Pull complete
1fd7e8b10447: Pull complete
bd485157db89: Pull complete
273a1970ce9c: Pull complete
006581b3a024: Pull complete
25c54ac351f0: Pull complete
Digest: sha256:77ebcec549076994f93ab85c5ce194e85366d9bcd124c53e1347660edd315666
Status: Downloaded newer image for microsoft/mssql-server-linux:latest
➜  ~
```

```
→  ~ docker pull microsoft/mssql-server-linux
Using default tag: latest
latest: Pulling from microsoft/mssql-server-linux
aed15891ba52: Pull complete
773ae8583d14: Pull complete
d1d48771f782: Pull complete
cd3d6cd6c0cf: Pull complete
8ff6f8a9120c: Pull complete
1fd7e8b10447: Pull complete
bd485157db89: Pull complete
273a1970ce9c: Pull complete
006581b3a024: Pull complete
25c54ac351f0: Pull complete
Digest: sha256:77ebcec549076994f93ab85c5ce194e85366d9bcd124c53e1347660edd315666
Status: Downloaded newer image for microsoft/mssql-server-linux:latest
→  ~ docker images
REPOSITORY                    TAG                 IMAGE ID            CREATED             SIZE
microsoft/mssql-server-linux  latest              694950fbcb3f        7 weeks ago         1.41GB
→  ~
```

Starting the Container with docker run

Always start with docker run

Starting the Container with docker run

```
docker run microsoft/mssql-server-linux
```

Name of image to run.
If not found in local repo,
docker will download it first.

Starting the Container with docker run

```
docker run                    microsoft/mssql-server-linux
```

Options go here

Starting a SQL Server Container with docker run

Required

```
docker run -e 'ACCEPT_EULA=Y' microsoft/mssql-server-linux
```

-e (environment variable)

Image requires you to accept the EULA

Starting a SQL Server Container with docker run

Required

Required

```
docker run -e 'ACCEPT_EULA=Y' -e 'SA_PASSWORD=Passw0rd'
  microsoft/mssql-server-linux
```

Image requires you set a password

"8+ characters of at least 3 of these 4 categories:
UPPERCASE letters, lowercase letters, numbers and non-alphanumeric symbols"

Starting a SQL Server Container with docker run

```
docker run -e 'ACCEPT_EULA=Y' -e 'SA_PASSWORD=Passw0rd'
  -p 1433:1433 microsoft/mssql-server-linux
```

-p port to expose the container from : port for SQL Server

Starting a SQL Server Container with docker run

```
docker run -e 'ACCEPT_EULA=Y' -e 'SA_PASSWORD=Passw0rd'
  -p 1433:1433 -d microsoft/mssql-server-linux
```

-d (detached mode)
Runs in background. Prompt returns and you continue to work at command line

Starting a SQL Server Container with docker run

```
docker run -e 'ACCEPT_EULA=Y' -e 'SA_PASSWORD=Passw0rd'
  -p 1433:1433 -d --name juliesqllinux
  microsoft/mssql-server-linux
```

--name : name to use for the container

Otherwise, a name will be auto-generated

Container also gets a GUID id

One more option: attach_dbs

```
docker run -e 'ACCEPT_EULA=Y' -e 'SA_PASSWORD=Passw0rd'
   -e attach_dbs="[{'dbName':'mydb'
                    'dbFiles':'mydb.mdf'}]"
   -p 1433:1433 -d
   --name juliesqllinux
   microsoft/mssql-server-linux
```

Starting the Windows Container with docker run

```
docker run -e ACCEPT_EULA=Y -e SA_PASSWORD=Passw0rd
  -p 1433:1433 -d --name sqlcontainer
  microsoft/mssql-server-windows-developer
```

(No quotes around the environment variables)

Using the Server: Same As Any SQL Server

Command line

sqlcmd (Windows, newly for Linux & macOS, from Microsoft)

sql-cli (open source, Windows, macOS & Linux)

IDEs

SQL Server Management Studio

SQL Operations Studio (cross-platform)

Visual Studio Code mssql Extension

JetBrains DataGrip (cross-platform)

Visual Studio SSDT (Windows only)

Others ….

Using the Server: Same As Any SQL Server

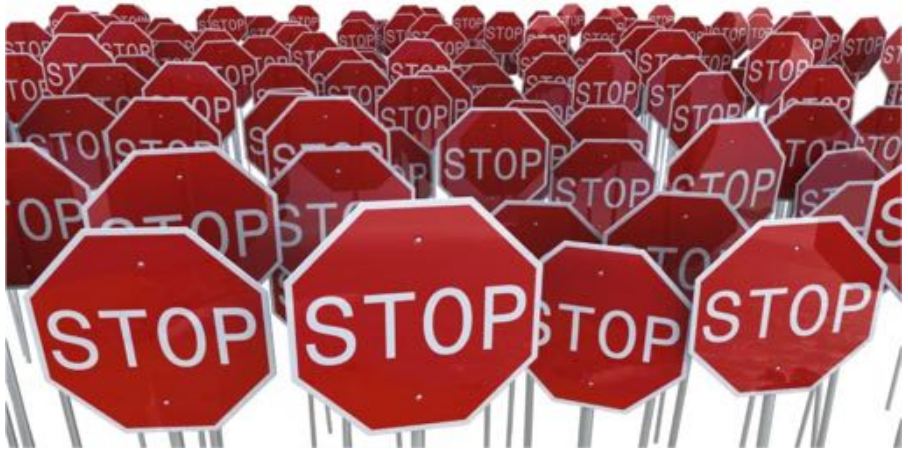
From your apps

“Updating” an Image

```
→  ~ docker ps -a
CONTAINER ID        IMAGE                                                COMMAND                  CREATED             STATUS
      PORTS                    NAMES
d397ed268a52        microsoft/mssql-server-linux                         "/bin/sh -c /opt/m..."   13 days ago         Exited (137) 4 seconds
ago                           juliesqllinux
3b77f21150be        julielerman/mssql-sqlserver-linux-adventureworkslt   "/bin/sh -c '/bin/..."   3 months ago        Exited (255) 2 months a
go    0.0.0.0:1433->1433/tcp   sqlserverlinuxawdb
→  ~ docker images
REPOSITORY                                          TAG                 IMAGE ID            CREATED             SIZE
julielerman/mssql-sqlserver-linux-adventureworkslt  latest              2fa5d0d85ea5        4 months ago        1.4GB
microsoft/mssql-server-linux                        latest              5985832855cf        5 months ago        1.38GB
→  ~ docker pull microsoft/mssql-server-linux
Using default tag: latest
latest: Pulling from microsoft/mssql-server-linux
aed15891ba52: Already exists
773ae8583d14: Already exists
d1d48771f782: Already exists
cd3d6cd6c0cf: Already exists
8ff6f8a9120c: Already exists
1fd7e0b10447: Already exists
bd4[illegible]157db89: Already exists
273a[illegible]970ce9c: Already exists
006581b3a024: Downloading [=====>                                  ]  28.65MB/272.6MB
25c54ac351f0: Downloading [=================>                    ]  29.73MB/88.24MB
```

Requires a completely new container instance to use the new image

Container Persistence (or Lack Thereof)



docker stop

PERSISTED

docker rm*

NOT PERSISTED

*remove

“Docker Volumes

allow you to upgrade containers, restart machines and share data without data loss. This is essential when updating database or application versions.”

-katacoda.com

Use Separate Volumes to Persist Data

Mapped volumes
Files stored directly on host (your computer/network)*

Data volumes
Data stored in a separate running container

**mssql-server-linux image cannot yet support mapped volumes on macOS*
Experience mapped volumes at katacoda.com/courses/docker/persisting-data-using-volumes

Creating and Binding to a Volume

```
docker create -v /var/opt/mssql --name mydatavolume
   microsoft/mssql-server-linux /bin/true

docker run -e 'ACCEPT_EULA=Y' -e 'SA_PASSWORD=Passw0rd'
  -p 1520:1433 -d  microsoft/mssql-server-linux
   --name juliesqllinux
   --volumes-from mydatavolume
   microsoft/mssql-server-linux

docker run ... -p 1602:1433 ... --name anothercontainer -
   --volumes-from mydatavolume
```

Build Your Own Image

docker build

docker run

Dockerfile

```
# using vNext image
FROM microsoft/mssql-server-linux

# set environment variables
ENV SA_PASSWORD=Passw0rd
ENV ACCEPT_EULA=Y

COPY entrypoint.sh entrypoint.sh
COPY SqlCmdStartup.sh SqlCmdStartup.sh
COPY SqlCmdScript.sql SqlCmdScript.sql

# Grant permissions for the SqlCmdStartup.sh
# script to be executable
RUN chmod +x ./SqlCmdStartup.sh
#start the waterfall of commands
#when started via docker run
CMD /bin/bash ./entrypoint.sh
```

Dockerfile

```
# using vNext image
FROM microsoft/mssql-server-linux

# set environment variables
ENV SA_PASSWORD=Passw0rd
ENV ACCEPT_EULA=Y

COPY entrypoint.sh entrypoint.sh
COPY SqlCmdStartup.sh SqlCmdStartup.sh
COPY SqlCmdScript.sql SqlCmdScript.sql

# Grant permissions for the SqlCmdStartup.sh
# script to be executable
RUN chmod +x ./SqlCmdStartup.sh
#start the waterfall of commands
#when started via docker run
CMD /bin/bash ./entrypoint.sh
```

entrypoint.sh

```
#start the script to create the DB and data
#then re-start the sqlserver
./SqlCmdStartup.sh & /opt/mssql/bin/sqlservr.sh
```

Dockerfile

```
# using vNext image
FROM microsoft/mssql-server-linux

# set environment variables
ENV SA_PASSWORD=Passw0rd
ENV ACCEPT_EULA=Y

COPY entrypoint.sh entrypoint.sh
COPY SqlCmdStartup.sh SqlCmdStartup.sh
COPY SqlCmdScript.sql SqlCmdScript.sql

# Grant permissions for the SqlCmdStartup.sh
# script to be executable
RUN chmod +x ./SqlCmdStartup.sh
#start the waterfall of commands
#when started via docker run
CMD /bin/bash ./entrypoint.sh
```

entrypoint.sh

```
#start the script to create the DB and data
#then re-start the sqlserver
./SqlCmdStartup.sh & /opt/mssql/bin/sqlservr.sh
```

SqlCmdStartup.sh

```
#wait for the SQL Server to come up
sleep 20s
#run setup script to create DB & its schema
/opt/mssql-tools/bin/sqlcmd -S localhost -U sa
-P Passw0rd -d master -i SqlCmdScript.sql
```

Dockerfile

```
# using vNext image
FROM microsoft/mssql-server-linux

# set environment variables
ENV SA_PASSWORD=Passw0rd
ENV ACCEPT_EULA=Y

COPY entrypoint.sh entrypoint.sh
COPY SqlCmdStartup.sh SqlCmdStartup.sh
COPY SqlCmdScript.sql SqlCmdScript.sql

# Grant permissions for the SqlCmdStartup.sh
# script to be executable
RUN chmod +x ./SqlCmdStartup.sh
#start the waterfall of commands
#when started via docker run
CMD /bin/bash ./entrypoint.sh
```

entrypoint.sh

```
#start the script to create the DB and data
#then re-start the sqlserver
./SqlCmdStartup.sh & /opt/mssql/bin/sqlservr.sh
```

SqlCmdStartup.sh

```
#wait for the SQL Server to come up
sleep 20s
#run setup script to create DB & its schema
/opt/mssql-tools/bin/sqlcmd -S localhost -U sa
-P Passw0rd -d master -i SqlCmdScript.sql
```

SqlCmdScript.sql

```
create database juliedb;
GO
use juliedb;
create table people (PersonId int Primary Key, Name nvarchar(50));
insert into people values (1,'julie');
insert into people values (2,'giantpuppy');
select * from people
GO
```

Important Links

github.com/Microsoft/mssql-docker

Official repository where docker files are maintained

hub.docker.com/r/microsoft/mssql-server-linux

mssql-server-windows-developer

mssql-server-windows-express

Resources

Data Points - On-the-Fly SQL Servers with Docker
msdn.microsoft.com/magazine/mt784660

Pluralsight: Cross-Platform SQL Server Management for Developers:
bit.ly/PS_MSSQL

SQL Server Lab : Docker for Windows
github.com/docker/labs/tree/master/windows/sql-server

Run the SQL Server 2017 container image with Docker
docs.microsoft.com/en-us/sql/linux/quickstart-install-connect-docker

Docker Volumes: docs.docker.com/engine/admin/volumes/volumes/

KataCoda hands on walkthrough on data volumes:
katacoda.com/courses/docker/persisting-data-using-volumes

SQL Server in Containers with Docker

Julie Lerman TheDataFarm.com @julielerman