IPG_LOGO_DEF

**TEMPLATE OFFERTE**

**PNERSHIP EL**

Docker

**Inhoudstafel**

[1 Docker Overview 3](#_Toc58419590)

[**1.1** Why using Docker 3](#_Toc58419591)

[1.1.1 As a user 3](#_Toc58419592)

[1.1.2 As a developer 3](#_Toc58419593)

[2 Info 4](#_Toc58419594)

[3 Using Docker 5](#_Toc58419595)

[**3.1** Install Docker for Windows 5](#_Toc58419596)

[**3.2** Linux subsystem for windows 5](#_Toc58419597)

[**3.3** Example 5](#_Toc58419598)

[4 Creating an image 6](#_Toc58419599)

[**4.1** Info 6](#_Toc58419600)

[**4.2** Templates : 6](#_Toc58419601)

[4.2.1 Dockerfile 6](#_Toc58419602)

[4.2.2 .dockerignore 6](#_Toc58419603)

[4.2.3 Build 7](#_Toc58419604)

[**4.3** Visual Studio Docker support 7](#_Toc58419605)

[4.3.1 New project 7](#_Toc58419606)

[4.3.2 Existing project 7](#_Toc58419607)

# Docker Overview

## Why using Docker

### As a user

You can use Docker as lightweight virtual machines that you can turn on and off at the blink of an eye. There’s a wide range of Docker images available to you. For example: in a couple of minutes you have a hole new Sql Server instance up and running. Stopping and restarting just takes a couple of seconds. You can easily have multiple database servers side by side. If you don’t need one anymore, just remove the container and/or image, using one or two statements

### As a developer

You can pack your apps into an image. At that point, there are no dependencies to underlying hardware and/or software. Wherever you deploy your app you can be sure everything is as you tested it.   
You then share your image on (for example) Docker Hub. There’s a free version for public images. If you need private images, you’ll have to pay (starts at $5/user/month).

Once the image is hosted, you can then pull and run it on any machine

# Info

<https://www.docker.com/get-started>

<https://www.docker.com/pricing>

https://docs.docker.com/engine/reference/commandline/docker/

# Using Docker

## Install Docker for Windows

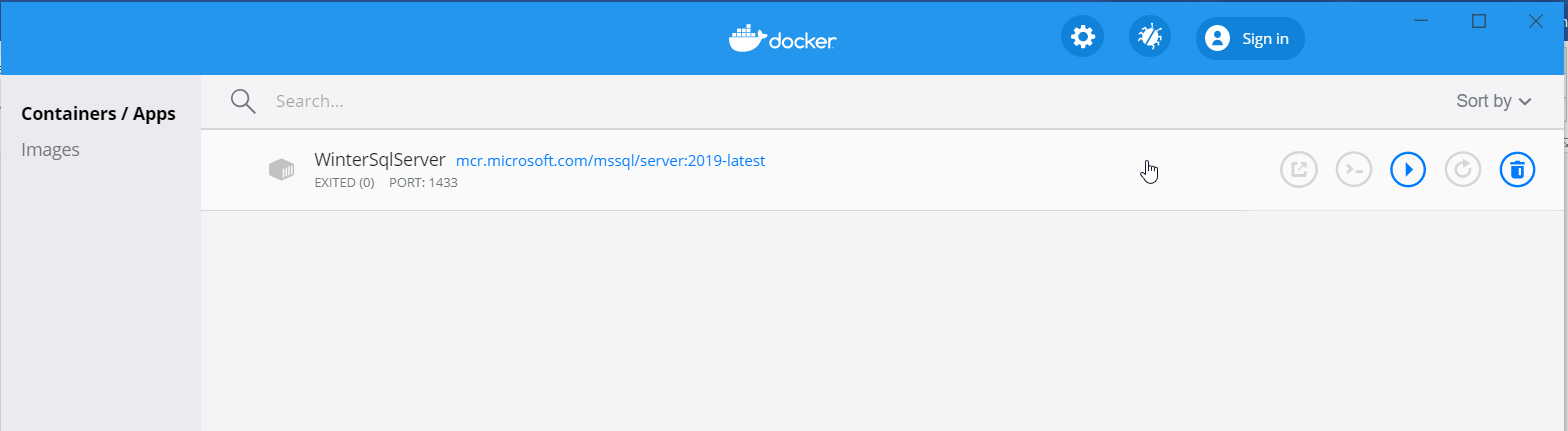
See: <https://www.docker.com/get-started>

## Linux subsystem for windows

You might need to install WSL2 (or higher), just follow the instructions when prompted. At my machine, it was nothing more than typing ‘wsl --set-default-version 2’ into powershell

## Example

In this walk-trough, we will get a SQL Server instance up and running

* Open a command prompt/bash window
* In order to get the image onto your machine, run:
  + docker pull mcr.microsoft.com/mssql/server:2019-latest
    - I’m using SQL Server 2019 because that is the version on P01**V**BCPSQL01 at the time of writing
* In order to get the image running, the first time, you need to run:
  + docker run -e "ACCEPT\_EULA=Y" -e "SA\_PASSWORD=<StrongPassword>" -p 1433:1433 --name <nameOfYourLiking> -h <nameOfYourLiking> -d mcr.microsoft.com/mssql/server:2019-latest
    - For ease of use: make ‘nameOfYourLiking’ the same both times
    - If you need multiple instances of SQL Server, you need to change the port (first value)
    - !! The password should follow the SQL Server default password policy, otherwise the container can NOT setup SQL server and will stop working.
    - For more info: <https://docs.microsoft.com/en-us/sql/linux/quickstart-install-connect-docker?view=sql-server-ver15&pivots=cs1-bash>
* Run 'docker ps -a' for checking if it is up and running
* Using SSMS, you can now connect the your SQL Server instance
  + SQL Server Authentication
  + SA + your password
  + Server name: localhost or . (and port if you changed the default port)
* For stopping/starting your container
  + Using the command line:
    - docker stop nameOfYourLiking
    - docker run nameOfYourLiking
  + Using Docker Desktop

# Creating an image

## Info

As there are a lot of options, a good ‘getting started’ place is: <https://github.com/dotnet/dotnet-docker>

## Templates :

### Dockerfile

# https://hub.docker.com/\_/microsoft-dotnet

FROM mcr.microsoft.com/dotnet/sdk:5.0 AS build

WORKDIR /source

# copy csproj and restore as distinct layers

COPY \*.sln .

COPY aspnetapp/\*.csproj ./aspnetapp/

RUN dotnet restore

# copy everything else and build app

COPY aspnetapp/. ./aspnetapp/

WORKDIR /source/aspnetapp

RUN dotnet publish -c release -o /app --no-restore

# final stage/image

FROM mcr.microsoft.com/dotnet/aspnet:5.0

WORKDIR /app

COPY --from=build /app ./

ENTRYPOINT ["dotnet", "aspnetapp.dll"]

### .dockerignore

# directories

\*\*/bin/

\*\*/obj/

\*\*/out/

# files

Dockerfile\*

\*\*/\*.md

\*\*/appsettings.Development.json

### Build

Run the following in the command line :

* docker build --pull -t <somename\_only\_lowercase> .
* docker run --rm -it -p 8000:80 aspnetapp

## Visual Studio Docker support

### New project

If you create a new project in Visual Studio, check ‘Docker support’. It’s that simple 😊. Once you do that, debugging takes place inside the container.

### Existing project

Right-click the project in the solution explorer and choose Add -> Docker Support