Demo Script: Containers for Windows Devs

Michael S. Collier

[michael.collier@microsoft.com](mailto:michael.collier@microsoft.com) | <http://www.michaelscollier.com> | @MichaelCollier

# Deck

C:\Projects\ContainersForWindowsDevs\ContainersForWindowsDev.pptx

# Demo environment setup & prereqs

* Launch Hyper-V Manager
* Get to Windows Container support in Docker.

**& 'C:\Program Files\Docker\Docker\DockerCli.exe' -SwitchDaemon**

* Load VS2017

**C:\Projects\ContainersForWindowsDevs\HelloWorldConsole\HelloWorldConsole.sln**

**C:\Projects\ContainersForWindowsDevs\HelloWorldWeb\HelloWorldWeb.sln**

**C:\Projects\ContainersForWindowsDevs\Containers for Windows Devs\HelloWorldWithDatabase\HelloWorldWithDatabase.sln**

* Launch SQL Server Management Studio
* Launch PowerShell (set font size to 24)
* Launch PowerShell ISE

**C:\Projects\ContainersForWindowsDevs\SQL vNext Containers Demo.ps1**

**C:\Projects\ContainersForWindowsDevs\scratch.ps1**

# Demo Reset

**docker rm -f $(docker ps -a -q)**

**docker rmi $(docker images -q)**

## Cache Images

### Windows

**docker pull microsoft/windowsservercore**

**docker pull microsoft/nanoserver:latest**

**docker pull microsoft/dotnet:nanoserver**

**docker pull microsoft/dotnet-samples:dotnetapp-nanoserver**

**docker pull microsoft/iis**

**docker pull microsoft/aspnet**

**docker pull microsoft/mssql-server-windows-developer**

### Linux

**microsoft/aspnetcore**

# Demo Projects

## #1 - Show Docker running on Windows 10

In this first demo, we’ll show Docker running on Windows 10 (installed via Docker for Windows). We’ll also show the ability to switch the Docker daemon to flip between running Windows containers and Linux containers. Finally, we’ll show the MobyLinuxVM that is installed to enable us to run Linux containers.

|  |  |
| --- | --- |
| **docker version** | Show the Client and Server version for Docker.  Show the Server OS/Arch setting – either ‘windows/amd64’ or ‘linux/amd64’. |
| **& 'C:\Program Files\Docker\Docker\DockerCli.exe' -SwitchDaemon** |  |
| **docker version** | Show the Server OS/Arch setting has changed. |
|  |  |
| **Switch to Hyper-V Manager** | Show the ‘MobyLinuxVM’ running. |
|  |  |

Let’s take a quick look at a few commands to view, pull, and remove images.

|  |  |
| --- | --- |
| **Switch to Windows Containers** |  |
|  |  |
| **docker images** | List images in the local repository. |
|  |  |
| **docker pull microsoft/dotnet-samples:dotnetapp-nanoserver** | Get an image from Docker Hub. |
|  |  |
| **docker rmi microsoft/iis** | Remove a local image |
|  |  |

## #2 – Manage Containers

Let’s take a quick look at a few commands to run, start, and stop Windows containers.

|  |  |
| --- | --- |
| **docker run microsoft/dotnet-samples:dotnetapp-nanoserver** | Run a container |
|  |  |
| **docker ps** | List running containers |
|  |  |
| **docker ps -a** | List all containers, including stopped containers |
|  |  |
| **docker start -ia <container\_id>** | Run an existing, stopped |
|  |  |

## #3 – Dockerize a .NET Console Application

In this demo, we’ll show how to take a simplistic .NET 4.5.2 console application and ‘Dockerize’ it; create a docker image from the application. We’ll then run that newly created docker image as a container.

|  |  |
| --- | --- |
| Show the code in HelloWorldConsole/Program.cs | Simple .NET 4.5.2. console application.  Readline() just to keep the console application open. |
|  |  |
| Create the Dockerfile / Add the existing Dockerfile  FROM microsoft/windowsservercore  ADD publish/ /  ENTRYPOINT HelloWorldConsole.exe | The base image used for a console .NET Framework application is microsoft/windowsservercore, publicly available on Docker Hub.  The base image contains a minimal installation of Windows Server 2016, .NET Framework 4.6.2 and serves as the base OS image for Windows Containers.  Add the content in the ‘publish/’ directory.  There is a post-build event added to the project to copy the necessary files.  The ENTRYPOINT for the container is the project executable. |
|  |  |
| **cd C:\Projects\ContainersForWindowsDevs\HelloWorldConsole\HelloWorldConsole\'** | Move to the project directory. |
|  |  |
| **docker build -t helloworldconsole:** **windevug.** | Create the image  *(don’t forget the “.”)* |
|  |  |
| **docker images** | Show the image just created. |
|  |  |
| **docker run --rm helloworldconsole:windevug “mike”** | Run the image, specifying the input parameter.  Using “--rm” will remove the container after it exits. |
|  |  |
| **docker ps -a** | Show there are no containers |

## #4 – SQL Server vNext in a Container

In this demo, we’ll look at running the next version of SQL Server in a Windows container (via Docker). As we do this, we’ll make a small modification to the container, create an image, and push the image to Azure Container Registry.

|  |  |
| --- | --- |
| Run through the script located at "C:\Projects\ContainersForWindowsDevs\SQL vNext Containers Demo.ps1" | |
|  |  |

## #5 – Visual Studio 2017 (Docker Tools)

In this example, we’ll look at a basic ASP.NET MVC site and how the Visual Studio Tools for Docker can help us to quickly convert it to a Docker image, run, debug running inside the container.

|  |  |
| --- | --- |
| **In VS2017, open ‘HelloWorldWeb’ sample application.** | "C:\Projects\ContainersForWindowsDevs\HelloWorldWeb\HelloWorldWeb.sln"  Run application. Show basic ASP.NET MVC application. |
|  |  |
| **Right click on HelloWorldWeb project -> Add -> Docker Project Support** | Adds several project files to support Docker:   * docker-compose.yml   + docker-compose.override.yml   + docker-compose.vs.debug.yml   + docker-compose.vs.release.yml * Dockerfile |
|  |  |
| **Show Dockerfile** | Starts with ‘microsoft/aspnet’ as base image. This images contains:   * Windows Server Core as the base OS * IIS 10 as Web Server * .NET Framework 4.6.2 (or 3.5) * .NET Extensibility for IIS   <https://hub.docker.com/r/microsoft/aspnet/> |
|  |  |
| **Place debug point in a controller (e.g. /Controllers/HomeController/About)** |  |
|  |  |
| **Debug the CONTAINER (Docker:Debug Project)** | Show the build output and how layers are being built, along with the image.  Show the command VS2017 runs to get the IP address for the Windows container:  docker inspect -f "{{ .NetworkSettings.Networks.nat.IPAddress }}" <containerId> |
|  |  |
| **Change text on Index view** | Refresh browser.  Show the change is reflected immediately. The docker-compose.vs.debug.yml file maps the local directory to c:\inetpub\wwwroot in the container.  Look at docker-compose.vs.release.yml and notice the mapping to c:\inetpub\wwwroot is not present. |
|  |  |
| **docker images** | Show that the ‘helloworldweb:dev’ image is present |
|  |  |
| **docker ps** | Show the container is still running even after exiting the Visual Studio debugger. Subsequent debug iterations are much faster. |

## #6 – Docker Compose w/ ASP.NET MVC and SQL Server

In this example, we’ll combine two recent examples, both ASP.NET MVC and SQL Server in containers, into one solution! We’ll use Docker Compose to manage both containers and allow to work together.

|  |  |
| --- | --- |
| **In VS2017, open ‘HelloWorldWithDatabase’ sample application.** | "C:\Projects\ContainersForWindowsDevs\HelloWorldWithDatabase\HelloWorldWithDatabase.sln"  Run application. Show basic ASP.NET MVC application. |
|  |  |
| **docker-compose.yml** | Same as before with ASP.NET example, but this includes a reference to the ‘db’ service – SQL Server.  Note the dependency on the “db” service. |
|  |  |
| **db\Dockerfile** | This Dockerfile is used to build the SQL Server container. It’s very similar to what we executed previously from the docker command line.  Have a SQL file to create the database and insert some data. |
|  |  |
| **Debug the Docker project** | Show the HomeController\AboutAsync method. |
|  |  |
| **Show ‘Api’ project** | Basic ASP.NET MVC Web APi project to return answers for a magic 8 ball. |
|  |  |
| **Debug the Docker project** | Show ‘Magic’ path.  Running three containers. Ability to step-into debug into the two web projects in Visual Studio . . . . just like you would expect. |