Deploying to Containers using TFS Build and Release Management

Benjamin Day
TRAINER | COACH | DEVELOPER
@benday www.benday.com

Overview

What is a Container?

What is Docker?

Crash course on Containers & Docker

Container-izing ASP.NET Core

Deploy ASP.NET Core to Docker from TFS Build

Container & Docker Overview

Disclaimer:
This is not a Docker “deep dive”.

Want more detail on Docker?

Introduction to Docker on Windows with Visual Studio 2017

by Marcel de Vries

Learn how to use Docker on Windows to containerize your application delivery. Learn full cycle CI/CD application delivery using Docker containers, all up to production clusters running on either Azure ACS with Kubernetes or on Service Fabric.

Course author

Marcel de Vries

Marcel is the co-founder and CTO of Xpirit, a high-end consulting firm based in Hilversum, Netherlands. Helping organizations transform towards a high speed, innovative, and productive organization...

Course info

Level	Beginner
Rating	★★★★★

https://www.pluralsight.com/courses/docker-visual-studio-2017-windows

What is a Container?

Infrastructure as code + application deployment & hosting

Similar to a virtual machine

Lightweight operating system hosted on another machine

Just enough to run your application

Why are containers interesting for DevOps?

Why Containers + DevOps?

Extremely repeatable

Describe your deployment environment and deployment at the same time

- Configuration file
- Goes into version control

Big Mindset Change:
No more application updates

New build? → New container

Need to patch the OS? → New container

What is Docker?

Docker is a company

- http://www.docker.com

Docker is a product

Set of tools for managing containers

Windows or Linux

Images vs. Containers

Class vs. Object

- Class is an description of data & methods
- Object is an instance of a class

Image

- Definition of the container

Container

- Running instance of an image

Windows Containers vs. Docker Containers?

Docker Images

Each image is a "layer" built on other images

Exist either locally and/or in a registry

- http://hub.docker.com

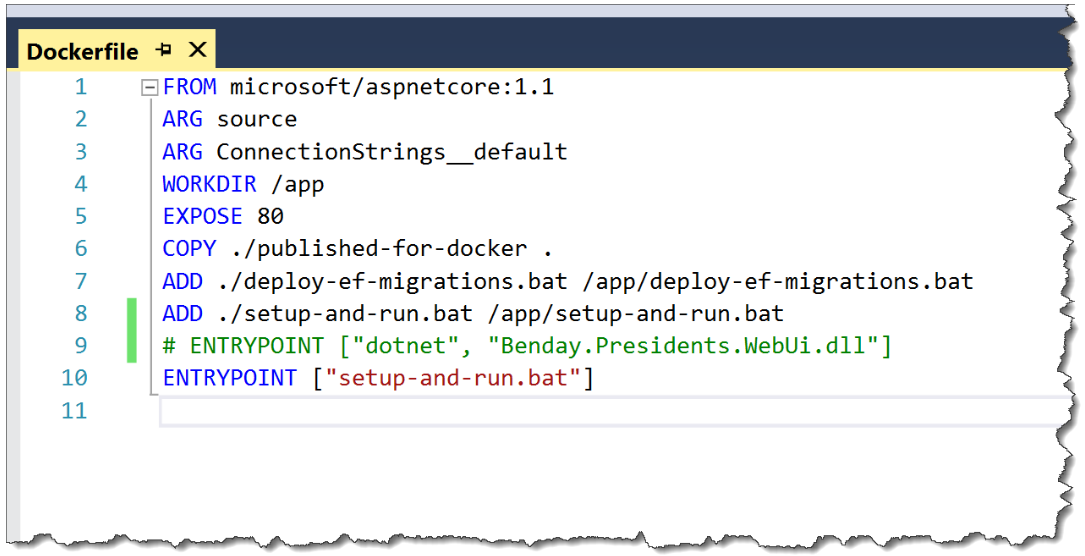
Choose an image

- https://hub.docker.com/u/microsoft/
- microsoft/aspnetcore:1.1
- microsoft/mssql-server-windows

Run the image…

…or create your own image using a Dockerfile

Dockerfile



```
FROM microsoft/aspnetcore:1.1
ARG source
ARG ConnectionStrings__default
WORKDIR /app
EXPOSE 80
COPY ./published-for-docker .
ADD ./deploy-ef-migrations.bat /app/deploy-ef-migrations.bat
ADD ./setup-and-run.bat /app/setup-and-run.bat
# ENTRYPOINT ["dotnet", "Benday.Presidents.WebUi.dll"]
ENTRYPOINT ["setup-and-run.bat"]
```

Describes the image

Refers to a base image

- FROM

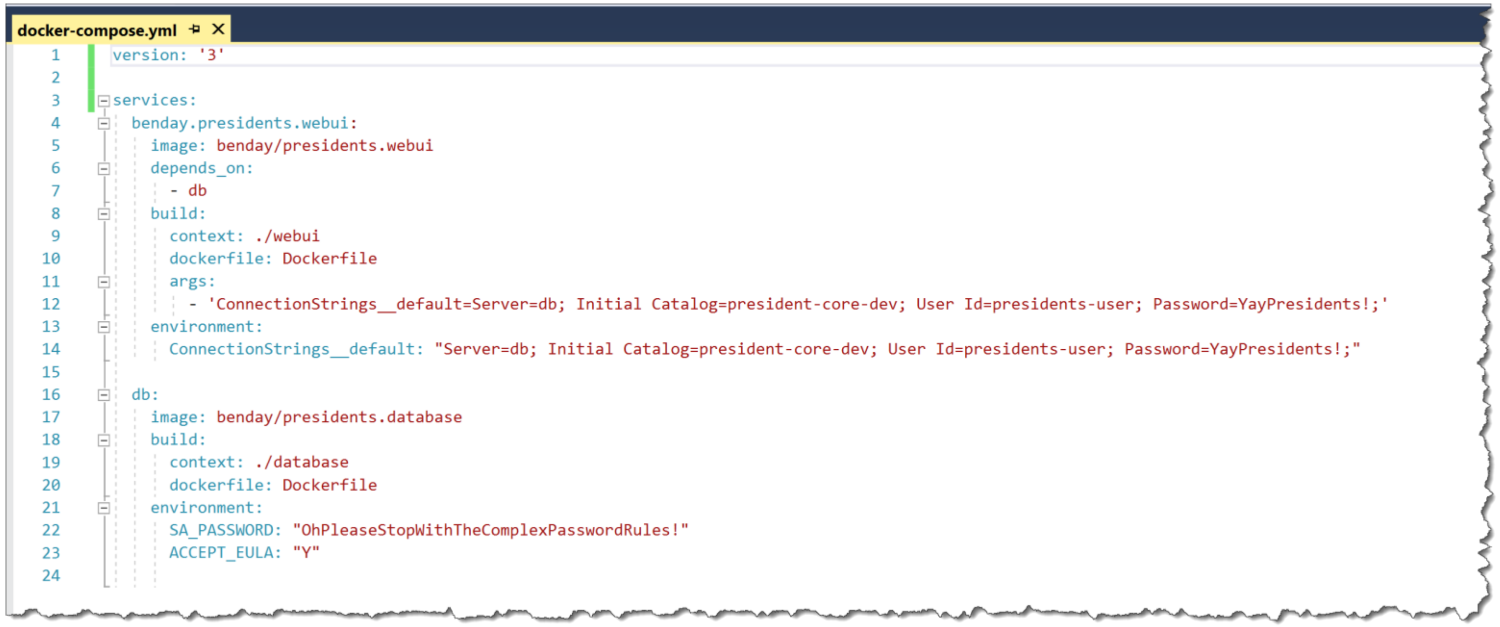
Has configuration

- Your files
- Network ports
- Environment variables

Runs something

- ENTRYPOINT

Docker Compose


docker-compose.yml

```yaml
version: '3'

services:
  benday.presidents.webui:
    image: benday/presidents.webui
    depends_on:
      - db
    build:
      context: ./webui
      dockerfile: Dockerfile
      args:
        - 'ConnectionStrings__default=Server=db; Initial Catalog=president-core-dev; User Id=presidents-user; Password=YayPresidents!;'
    environment:
      ConnectionStrings__default: "Server=db; Initial Catalog=president-core-dev; User Id=presidents-user; Password=YayPresidents!;"

  db:
    image: benday/presidents.database
    build:
      context: ./database
      dockerfile: Dockerfile
    environment:
      SA_PASSWORD: "OhPleaseStopWithTheComplexPasswordRules!"
      ACCEPT_EULA: "Y"

```

Describe multiple containers working together

docker-compose.yml

Container = service

Wildly Over-simplified Description of Docker's Structure

Docker Service / Daemon

- Does the work

Docker Command Line Interface (CLI)

- Talks to the service
- Administer containers & images
- Knows about Dockerfile
- Knows how to build images
- Knows how to run containers

Docker Compose

- docker-compose.yml
- docker-compose build
- docker-compose up

Next up:
Docker-izing ASP.NET Core for DevOps

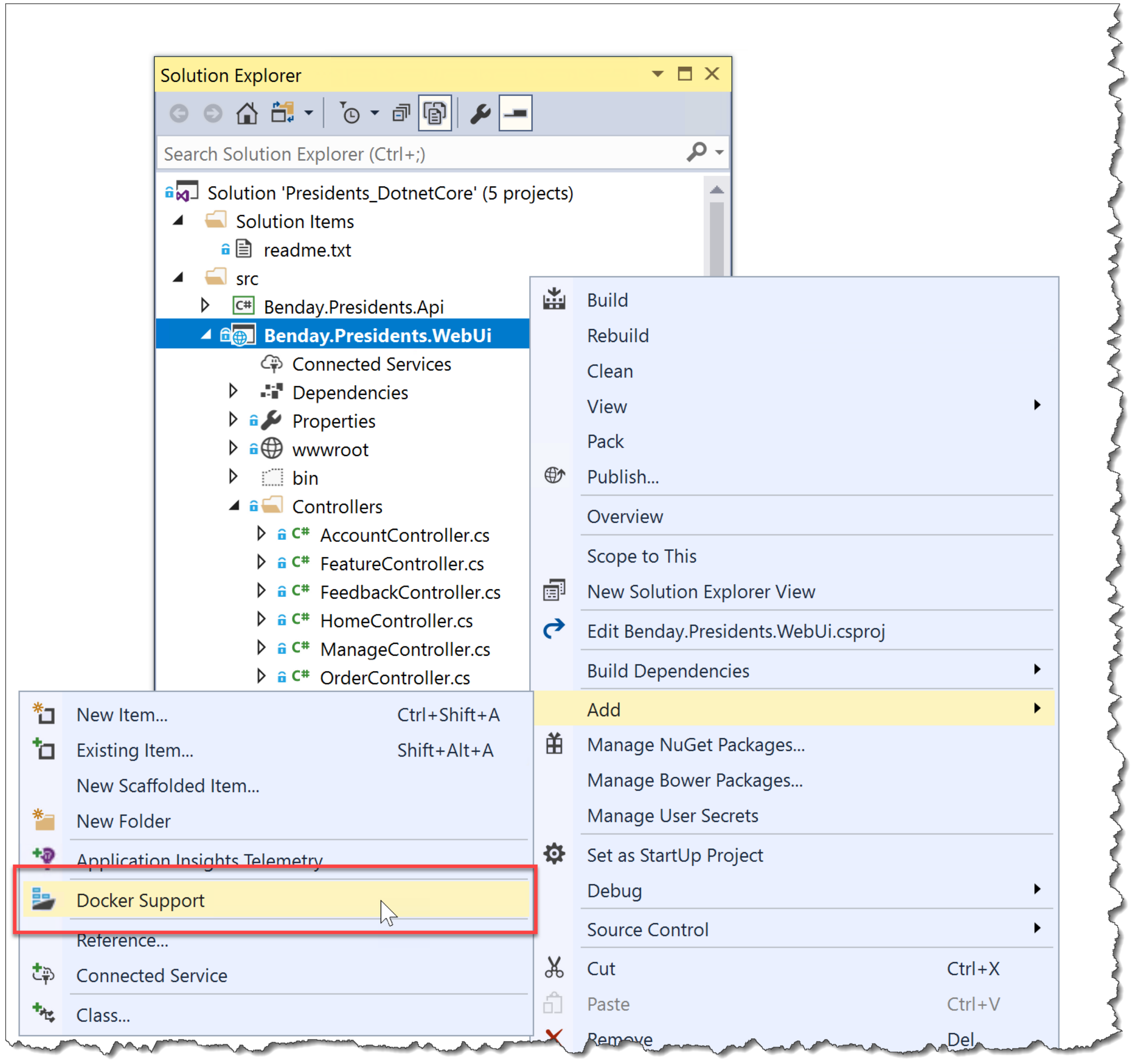
Container-ize / Docker-ize an ASP.NET Core application

Two ways to Docker-ize:

1) Visual Studio

2) Notepad & Command Line

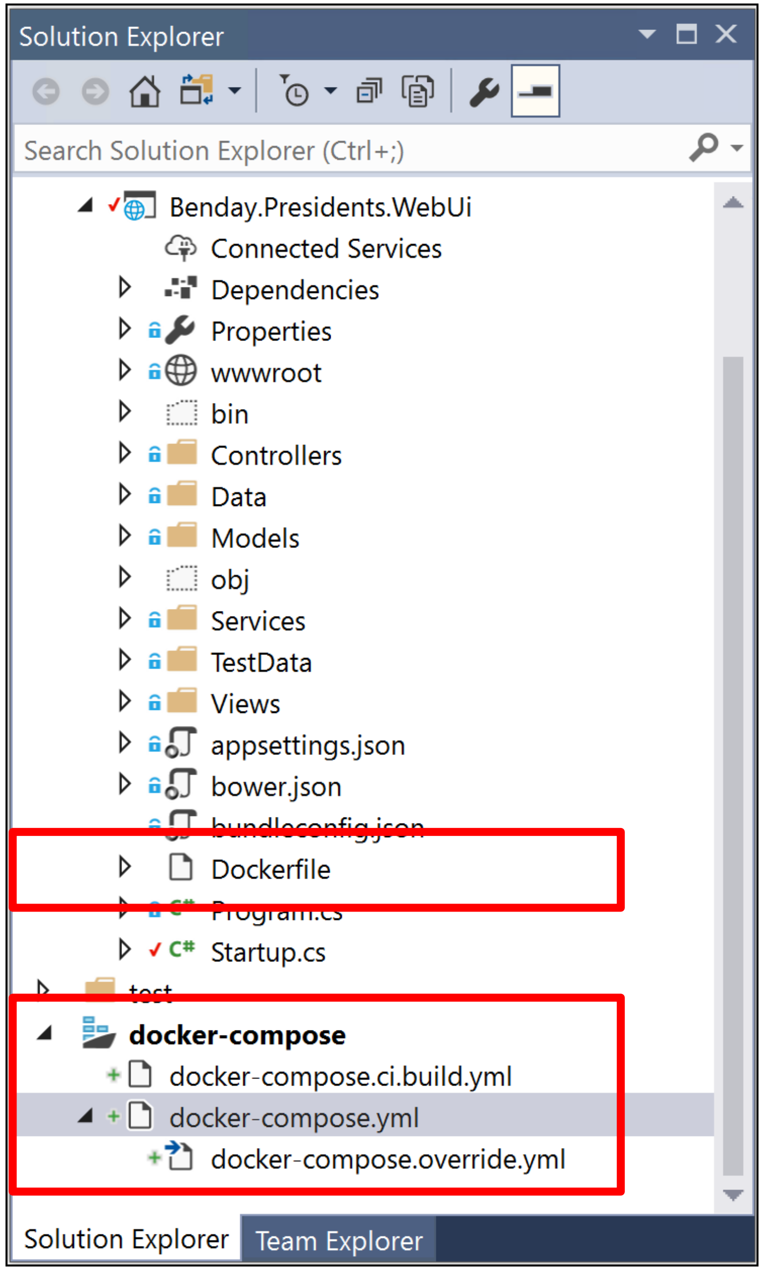
Add Docker Support using Visual Studio 2017



Right-click the project

Add → Docker Support

Windows or Linux?



Solution Explorer

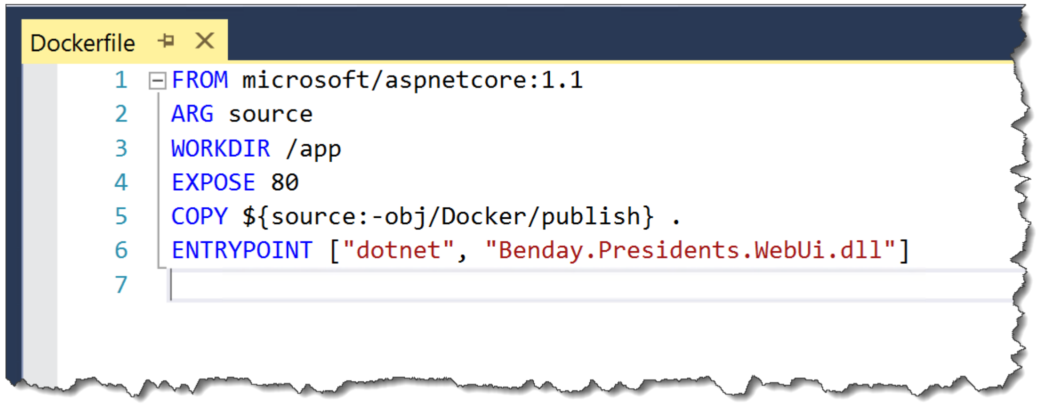
Search Solution Explorer (Ctrl+;)

- Benday.Presidents.WebUi
  - Connected Services
  - Dependencies
  - Properties
  - wwwroot
  - bin
  - Controllers
  - Data
  - Models
  - obj
  - Services
  - TestData
  - Views
  - appsettings.json
  - bower.json
  - bundleconfig.json
  - Dockerfile
  - Program.cs
  - Startup.cs
- test
- **docker-compose**
  - docker-compose.ci.build.yml
  - docker-compose.yml
    - docker-compose.override.yml

Solution Explorer | Team Explorer

Dockerfile in the web project

New “docker-compose” project



```
FROM microsoft/aspnetcore:1.1
ARG source
WORKDIR /app
EXPOSE 80
COPY ${source:-obj/Docker/publish} .
ENTRYPOINT ["dotnet", "Benday.Presidents.WebUi.dll"]
```

Dockerfile

Describes a Docker image

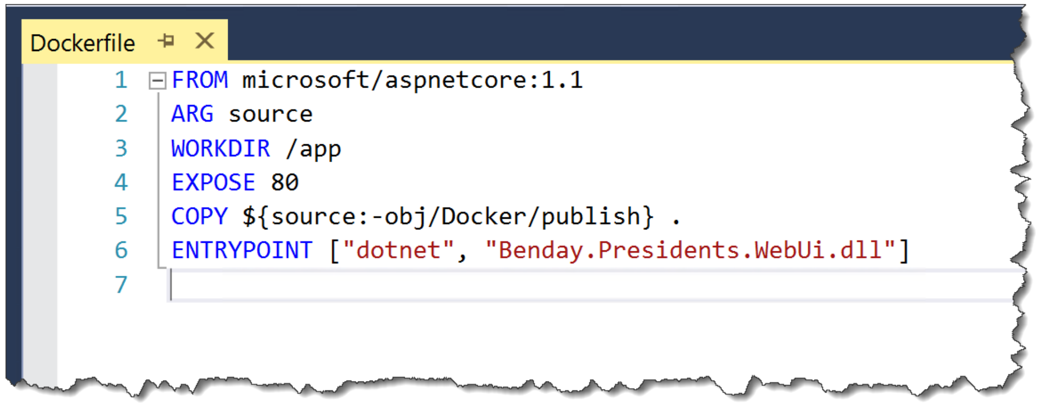
Starts from a base image

- FROM

Adds in your content & config

Run “docker build”

→ Your image



Dockerfile

```
FROM microsoft/aspnetcore:1.1
ARG source
WORKDIR /app
EXPOSE 80
COPY ${source:-obj/Docker/publish} .
ENTRYPOINT ["dotnet", "Benday.Presidents.WebUi.dll"]
```

FROM

- Base image

WORKDIR

- Path inside of container

EXPOSE

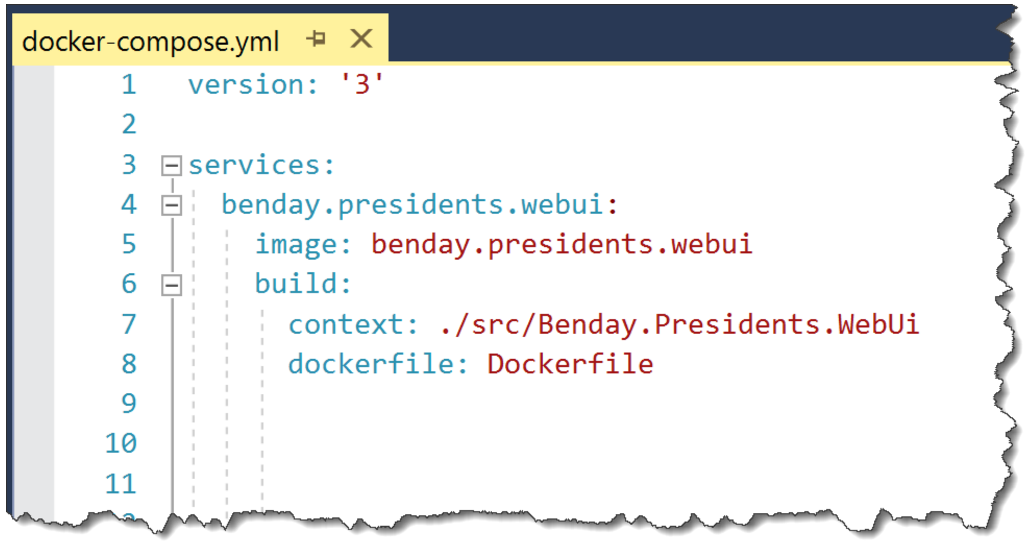
- Network port

COPY

- Build output into image

ENTRYPOINT

- What to run?



```yaml
version: '3'

services:
  benday.presidents.webui:
    image: benday.presidents.webui
    build:
      context: ./src/Benday.Presidents.WebUi
      dockerfile: Dockerfile
```

docker-compose.yml

Describes images that will work together

In Visual Studio terms…

- docker-compose.yml = "solution file"
- Dockerfile = "project file"

Interesting commands:

- "docker-compose build"
- "docker-compose up"
- "docker-compose down"

For DevOps, I prefer coding Docker by hand rather than using Visual Studio.

“dotnet build” vs. “dotnet publish”

Docker using Notepad or Visual Studio?

Visual Studio

Pros:

- Right there in Visual Studio
- Great for development

Cons:

- Docker gets mingled with your code
- Images are based on “build” rather than “publish”
- Requires Docker on your dev workstation

Notepad / by hand

Pros:

- Docker stuff stays separated
- Easier to handle published code
- More control → more repeatable
- Not everyone needs Docker installed

Cons:

- You’re doing it by hand

Next up:
Demos

Demo Overview

Demo Goal 1:
Run the application & SQL Server database in Containers

Demo Goal 2:
The app is fully configured

The Docker Demos

Note: These demos don’t use TFS

Run SQL Server in a container

Tour of the database image

Tour of the webui image

Tour of the docker-compose.yml file

Build & run the containers

Demo

Run SQL Server in a Docker container

Demo

Tour of the database image Dockerfile

Demo

Tour of the ASP.NET Core app's Dockerfile

Demo

Tour of docker-compose.yml

Build & Run using “docker-compose build” & “docker-compose up”

Demo

Docker-ize the Presidents ASP.NET Core Application

ASP.NET Core app in a container

SQL Server in a container

Use docker-compose to connect the two

Setup SQL users & permissions in database

Deploy Entity Framework Migrations

Next up:
Docker Lessons Learned

Docker Lessons Learned

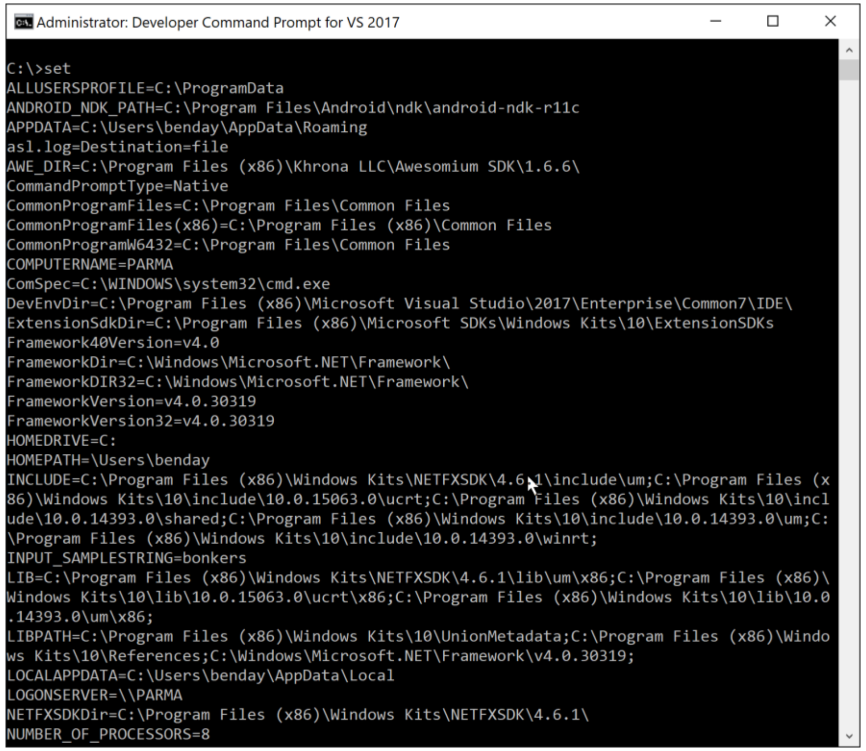
Docker Lessons Learned the Hard Way

Environment variables are everything.

How to Debug Your Environment Variables on Windows

Command Prompt

Run “set”

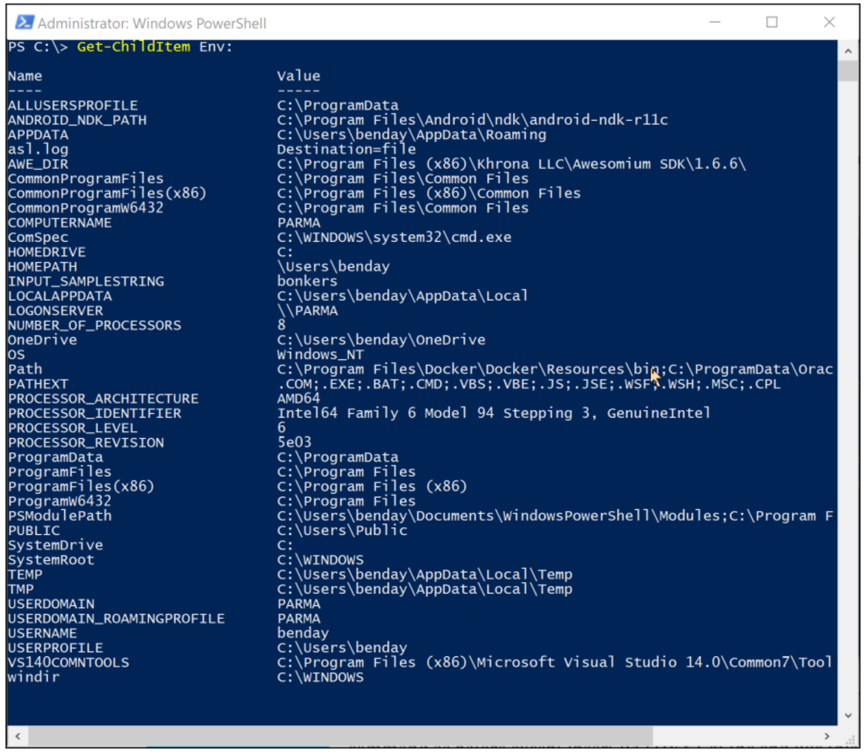


Administrator: Developer Command Prompt for VS 2017

```
C:\>set
ALLUSERSPROFILE=C:\ProgramData
ANDROID_NDK_PATH=C:\Program Files\Android\ndk\android-ndk-r11c
APPDATA=C:\Users\benday\AppData\Roaming
asl.log=Destination=file
AWE_DIR=C:\Program Files (x86)\Khrona LLC\Awesomium SDK\1.6.6\
CommandPromptType=Native
CommonProgramFiles=C:\Program Files\Common Files
CommonProgramFiles(x86)=C:\Program Files (x86)\Common Files
CommonProgramW6432=C:\Program Files\Common Files
COMPUTERNAME=PARMA
ComSpec=C:\WINDOWS\system32\cmd.exe
DevEnvDir=C:\Program Files (x86)\Microsoft Visual Studio\2017\Enterprise\Common7\IDE\
ExtensionSdkDir=C:\Program Files (x86)\Microsoft SDKs\Windows Kits\10\ExtensionSDKs
Framework40Version=v4.0
FrameworkDir=C:\Windows\Microsoft.NET\Framework\
FrameworkDIR32=C:\Windows\Microsoft.NET\Framework\
FrameworkVersion=v4.0.30319
FrameworkVersion32=v4.0.30319
HOMEDRIVE=C:
HOMEPATH=\Users\benday
INCLUDE=C:\Program Files (x86)\Windows Kits\NETFXSDK\4.6.1\include\um;C:\Program Files (x
86)\Windows Kits\10\include\10.0.15063.0\ucrt;C:\Program Files (x86)\Windows Kits\10\incl
ude\10.0.14393.0\shared;C:\Program Files (x86)\Windows Kits\10\include\10.0.14393.0\um;C:
\Program Files (x86)\Windows Kits\10\include\10.0.14393.0\winrt;
INPUT_SAMPLESTRING=bonkers
LIB=C:\Program Files (x86)\Windows Kits\NETFXSDK\4.6.1\lib\um\x86;C:\Program Files (x86)\
Windows Kits\10\lib\10.0.15063.0\ucrt\x86;C:\Program Files (x86)\Windows Kits\10\lib\10.0
.14393.0\um\x86;
LIBPATH=C:\Program Files (x86)\Windows Kits\10\UnionMetadata;C:\Program Files (x86)\Windo
ws Kits\10\References;C:\Windows\Microsoft.NET\Framework\v4.0.30319;
LOCALAPPDATA=C:\Users\benday\AppData\Local
LOGONSERVER=\\PARMA
NETFXSDKDir=C:\Program Files (x86)\Windows Kits\NETFXSDK\4.6.1\
NUMBER_OF_PROCESSORS=8
```

Powershell

Run “Get-ChildItem Env:”



Administrator: Windows PowerShell

```
PS C:\> Get-ChildItem Env:

Name                           Value
----                           -----
ALLUSERSPROFILE                C:\ProgramData
ANDROID_NDK_PATH               C:\Program Files\Android\ndk\android-ndk-r11c
APPDATA                        C:\Users\benday\AppData\Roaming
asl.log                        Destination=file
AWE_DIR                        C:\Program Files (x86)\Khrona LLC\Awesomium SDK\1.6.6\
CommonProgramFiles             C:\Program Files\Common Files
CommonProgramFiles(x86)        C:\Program Files (x86)\Common Files
CommonProgramW6432             C:\Program Files\Common Files
COMPUTERNAME                   PARMA
ComSpec                        C:\WINDOWS\system32\cmd.exe
HOMEDRIVE                      C:
HOMEPATH                       \Users\benday
INPUT_SAMPLESTRING             bonkers
LOCALAPPDATA                   C:\Users\benday\AppData\Local
LOGONSERVER                    \\PARMA
NUMBER_OF_PROCESSORS           8
OneDrive                       C:\Users\benday\OneDrive
OS                             Windows_NT
Path                           C:\Program Files\Docker\Docker\Resources\bin;C:\ProgramData\Orac
PATHEXT                        .COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC;.CPL
PROCESSOR_ARCHITECTURE         AMD64
PROCESSOR_IDENTIFIER           Intel64 Family 6 Model 94 Stepping 3, GenuineIntel
PROCESSOR_LEVEL                6
PROCESSOR_REVISION             5e03
ProgramData                    C:\ProgramData
ProgramFiles                   C:\Program Files
ProgramFiles(x86)              C:\Program Files (x86)
ProgramW6432                   C:\Program Files
PSModulePath                   C:\Users\benday\Documents\WindowsPowerShell\Modules;C:\Program F
PUBLIC                         C:\Users\Public
SystemDrive                    C:
SystemRoot                     C:\WINDOWS
TEMP                           C:\Users\benday\AppData\Local\Temp
TMP                            C:\Users\benday\AppData\Local\Temp
USERDOMAIN                     PARMA
USERDOMAIN_ROAMINGPROFILE      PARMA
USERNAME                       benday
USERPROFILE                    C:\Users\benday
VS140COMNTOOLS                 C:\Program Files (x86)\Microsoft Visual Studio 14.0\Common7\Tool
windir                         C:\WINDOWS
```

The docker-compose.yml file format *REALLY* cares about whitespace & indentation.

When you build an image, it tries to run the “intermediate” container.

Environment variables are different between the build and run phases.

If you aren’t careful, these two phases will <u>eat your soul</u> while trying to debug “docker-compose build” versus “docker-compose up”.

In docker-compose.yml, remember that…

…environment variables during *build* are specified using “args”
but
environment variables at *runtime* are specified using “environment”.

[show what I’m talking about]

In your Dockerfile,
the RUN command executes at
build and not at container startup.

Your container does one thing and has one entry point.

Do you need your container to do multiple things?

You’ll need a script as your entry point.

Save typing.
Script everything.

Next up:
Run this from a TFS Build

Demo

Build & Run Images from a TFS Build

Summary

What is a Container?

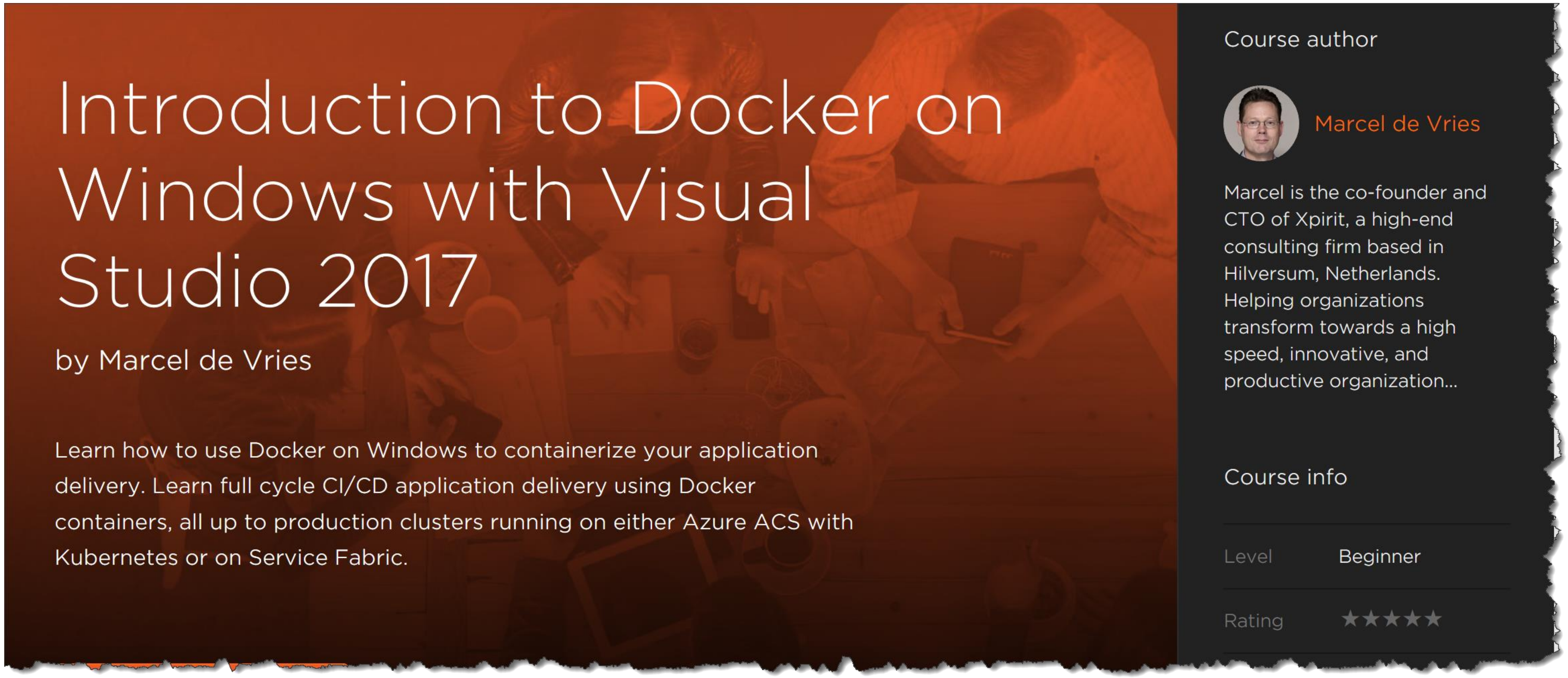
What is Docker?

Crash course on Containers & Docker

Container-izing ASP.NET Core

Deploy ASP.NET Core to Docker from TFS Build

Want more detail on Docker?



Introduction to Docker on Windows with Visual Studio 2017

by Marcel de Vries

Learn how to use Docker on Windows to containerize your application delivery. Learn full cycle CI/CD application delivery using Docker containers, all up to production clusters running on either Azure ACS with Kubernetes or on Service Fabric.

Course author

Marcel de Vries

Marcel is the co-founder and CTO of Xpirit, a high-end consulting firm based in Hilversum, Netherlands. Helping organizations transform towards a high speed, innovative, and productive organization...

Course info

Level	Beginner
Rating	★★★★★

https://www.pluralsight.com/courses/docker-visual-studio-2017-windows

Next up:
Testing & DevOps