Xpirit

Think ahead. Act now.

Docker and Visual Studio

Marcel de Vries

Overview

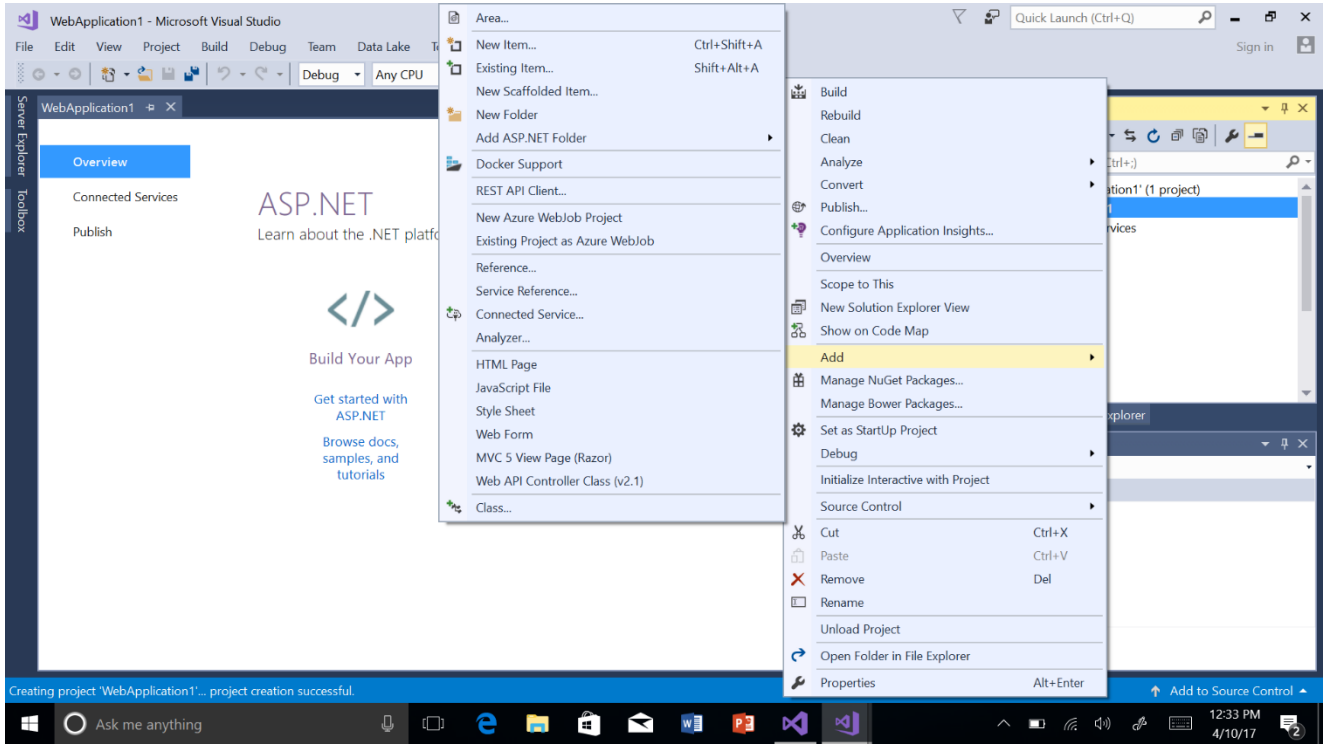
- Using the Docker tools in Visual Studio 2017
- Building your images with Docker files
- Building a group of containers using Docker compose files
- Debugging your cross-container solutions with visual Studio 2017

Using the Docker Tools in Visual Studio 2017

Xpirit
Think ahead. Act now.

Docker Tools in Visual Studio 2017

- Add Docker support to your project
- Understands ASP.NET vs. .NET Core
- Adds required Docker files to the projects
- Adds required Docker-compose Yaml files



Xpirit
Think ahead. Act now.

Docker Files Added

The Docker files added

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

```
FROM microsoft/aspnet
ARG source
WORKDIR /inetpub/wwwroot
COPY ${source:-obj/Docker/publish} .
```

```
FROM microsoft/aspnet
ARG source
WORKDIR /inetpub/wwwroot
COPY ${source:-obj/Docker/publish} .
```

The Dockerfile Debug/release Trick

Pass in argument at build

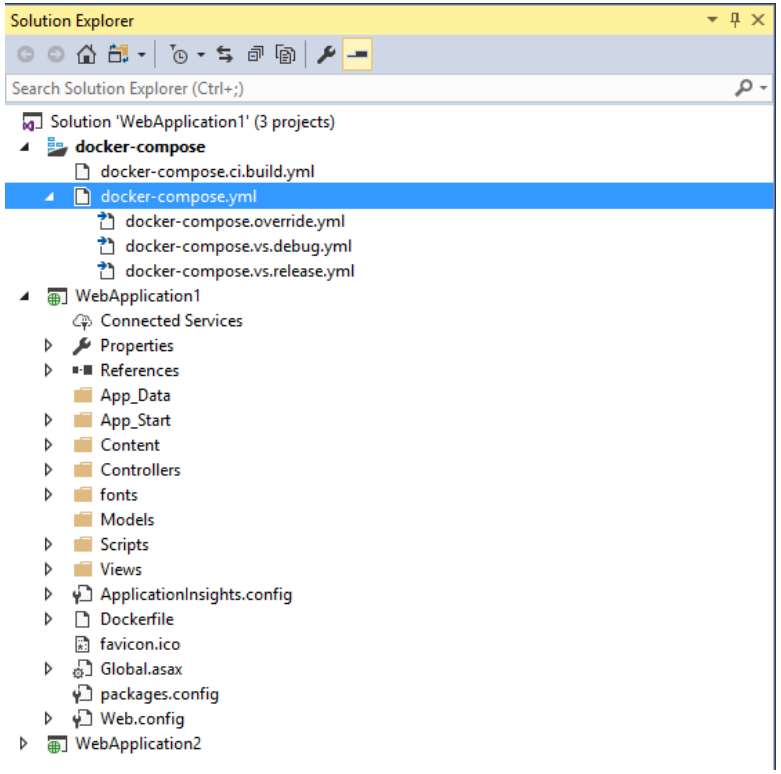
Debug -> obj/docker/empty

Release -> obj/docker/publish

Demo

Building Containers from the Command-line

Yaml Files Added



Used to run container compositions

Build

Run

Are a superset of json

You can pass multiple files on the command-line that get merged

Provide additional run information to build or start the containers

```
services:
  webapplication1:
    image: webapplication1
    build:
      context: .\WebApplication1
      dockerfile: Dockerfile
```

The Yaml File Trick

The docker-compose file defines the services

Defines what to build

{override}

```
services:
  webapplication1:
    ports:
      - "80“
networks:
  default:
    external:
      name: nat
```

The Yaml File Trick

The override file defines the ports and networking

Demo

Building Containers from the Command-line
Using Docker-compose

Xpirit
Think ahead. Act now.

{Debug}

```
services:
   webapplication1:
          image: webapplication1:dev
          build:
              args:
                  source: ${DOCKER_BUILD_SOURCE}
          volumes:
                  - .\WebApplication1:C:\inetpub\wwwroot
                  - ~\msvsmon:C:\msvsmon:ro
          labels:
              -
"com.microsoft.visualstudio.targetoperatingsystem=windows"
```

The Yaml File Trick

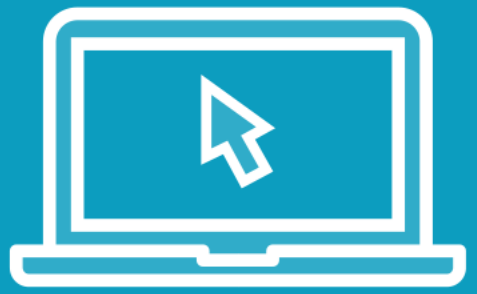
Source arg dockerfile get's the DOCKER_BUILD_SOURCE value == obj\docker\empty

Volume mapping to c:\inetpub\wwwroot

Volume mapping to local user folder (~) on host to remote debugger

Xpirit
Think ahead. Act now.

Demo



Show the Volume Maps on the Containers

Xpirit
Think ahead. Act now.

{Release}

```
services:
   webapplication1:
           image: webapplication1:dev
           build:
               args:
                   source: ${DOCKER_BUILD_SOURCE}
           volumes:
                   - ~\msvsmon:C:\msvsmon:ro
           labels:
               -
"com.microsoft.visualstudio.targetoperatingsystem=windows"
```

The Yaml File Trick

Source arg dockerfile get's the DOCKER_BUILD_SOURCE value == obj\docker\publish

Volume mapping to local user folder (~) on host to remote debugger

```
services:
  ci-build:
    image: microsoft/aspnetcore-build:1.0-1.1
    volumes:
      - .:/src
    working_dir: /src
    command: /bin/bash -c "dotnet restore ./dotnetcore.sln && dotnet
\
             publish ./dotnetcore.sln -c Release -o
./obj/Docker/publish"
```

The Compose File to Build The Source

Docker-compose.ci.build

Container image used to build the sources

Enables you to switch build environments in a very easy way

Demo

Use the Visual Studio 2017 IDE to Build Containers

Debugging Your Cross-container Solutions with Visual Studio 2017

Xpirit
Think ahead. Act now.

Debugging Containers

This uses the remote debugger capabilities of Visual Studio better known under the name msvsmon

Can be found on your local machine [osdisk]:\Users\<username>\msvsmon

Is mapped in the yaml file as volume mount on your container

Demo

Show How to Debug Cross Containers

Summary

- Using the Docker tools in Visual Studio 2017
- Building your images with Docker files
- Building a group of containers using Docker compose files
- Debugging your cross-container solutions with visual Studio 2017