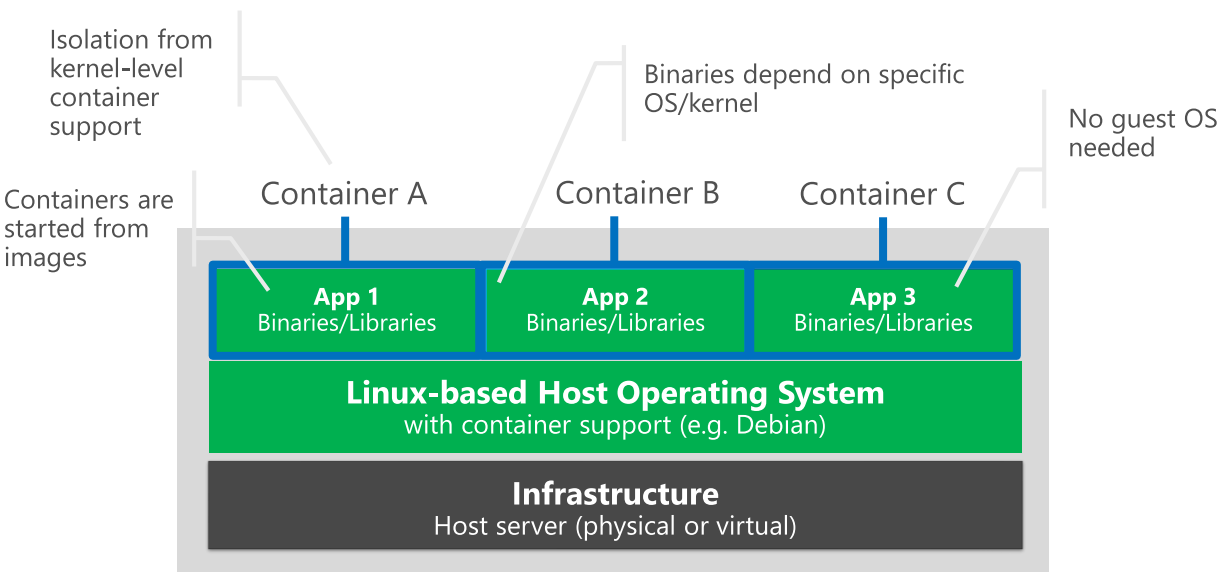
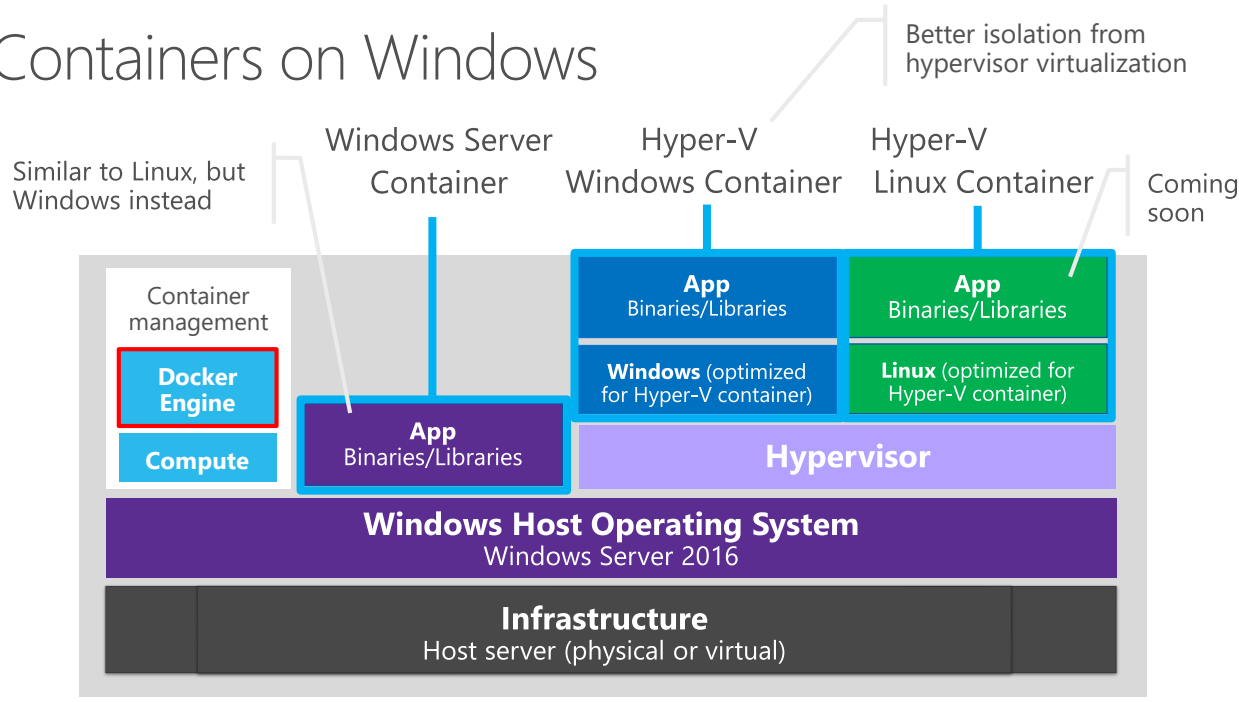




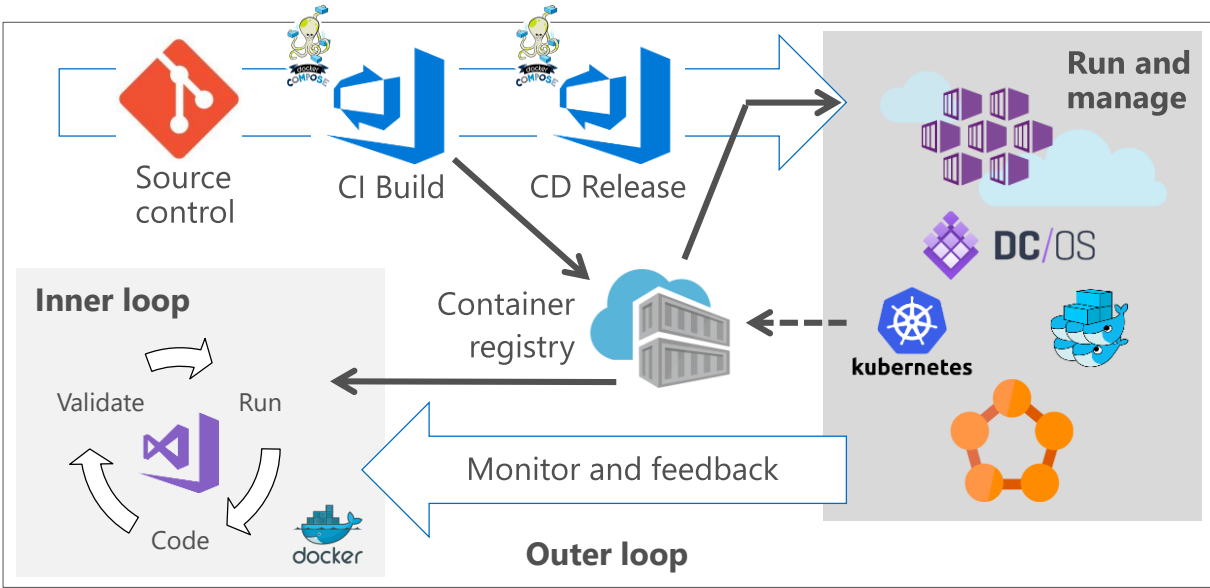
# Running containers on Linux



# Containers on Windows



# New developer workflow



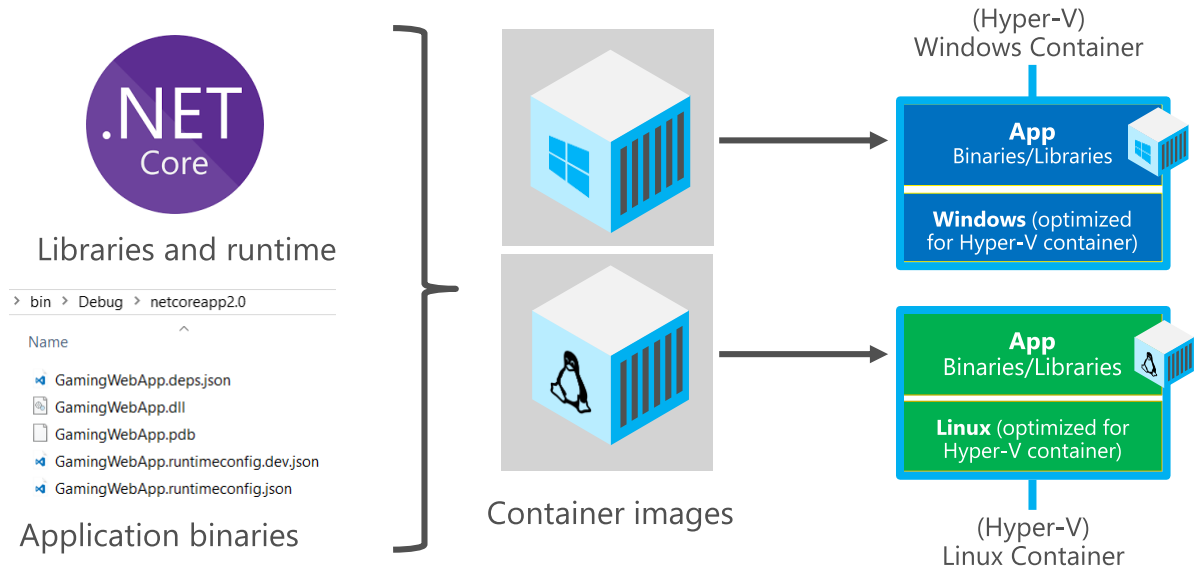
## Demo

Building and running Docker images





# Deploying your .NET apps with containers



## VS2017 container building

### Dockerfile per project

Special **source** argument for app binaries, or obj/Docker/publish as fallback  
 Expected to use docker-compose 3.0 and up

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

### Can be built without VS2017

Command-line tooling  
 Visual Studio Code

# Docker image layers .NET Core



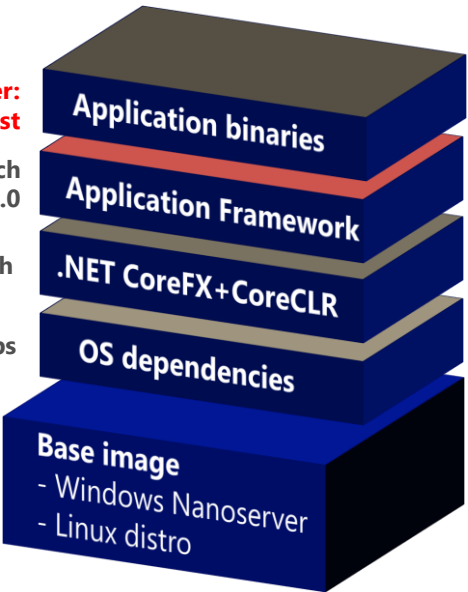
**Your application layer:**  
e.g. `techdays/gamingwebapp:latest`

`microsoft/aspnetcore:2.0.0-stretch`  
`microsoft/aspnetcore:2.0`

`microsoft/dotnet:2.0.0-runtime-stretch`

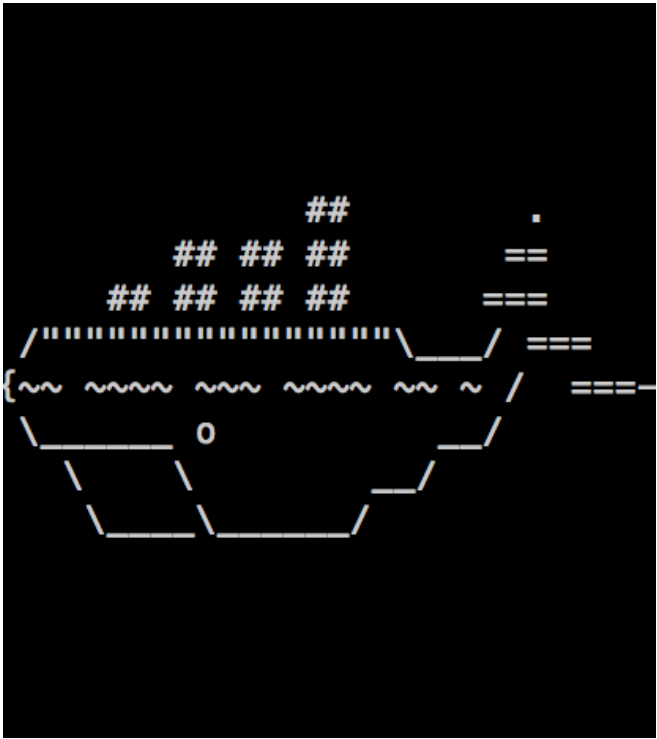
`microsoft/dotnet:2.0-runtime-deps`

`microsoft/nanoserver:10.0.14393.1715`  
-or- `debian:stretch`



## Demo

Creating Docker images




# Building containers from VS2017

## How it works

Sources

- Leaderboard.WebAPI
- Connected Services
- Dependencies
- Properties
- wwwroot
- Controllers
- Infrastructure
- Migrations
- Mo
- app
- Do
- C# Pro
- C# Sta



Visual Studio

Compile

Build image

```
2>Step 1/6 : FROM microsoft/aspnetcore:1.1
2> ----> 9f2f15b2012f
2>Step 2/6 : ARG source
2> ----> Using cache
2> ----> ffd4d4dc0f9a
2>Step 3/6 : WORKDIR /app
2> ----> Using cache
2> ----> b1cbbd07cf2b
2>Step 4/6 : EXPOSE 80
2> ----> Using cache
2> ----> 0e17cb1c9f53
2>Step 5/6 : COPY ${source:-obj/Docker/publish} .
2> ----> Using cache
2> ----> 18f1b48ae4f9
2>Step 6/6 : ENTRYPOINT dotnet GamingWebApp.dll
2> ----> Using cache
2> ----> 4e8fd2d93ea8
2>Successfully built 4e8fd2d93ea8
```

In C# this would be source ?? "obj/Docker/Publish"

Determined by VS2017 in **DOCKER\_BUILD\_SOURCE**

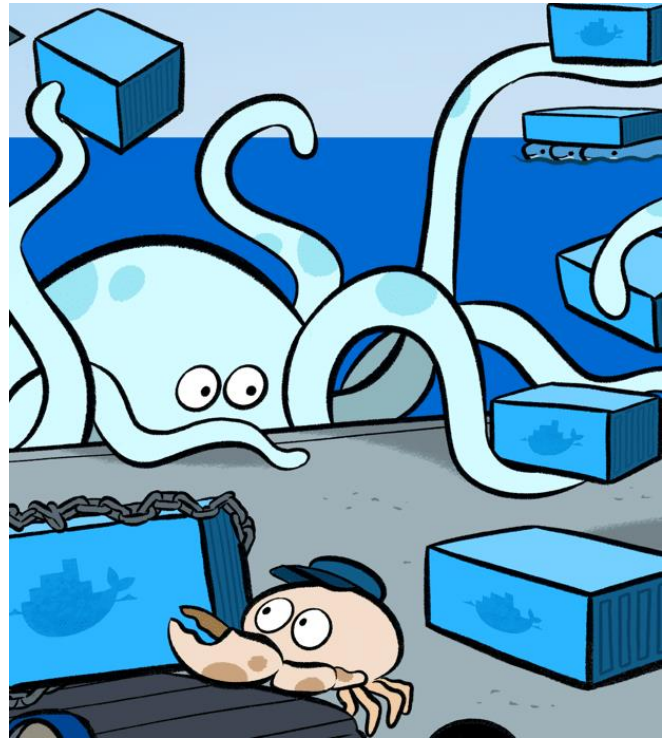
Dockerfile

```
FROM microsoft/aspnetcore:1.1
ARG source
WORKDIR /app
COPY !obj\docker\publish\*
COPY !obj\docker\empty\
ENTRYPOINT dotnet GamingWebApp.dll
```

PI.dll"

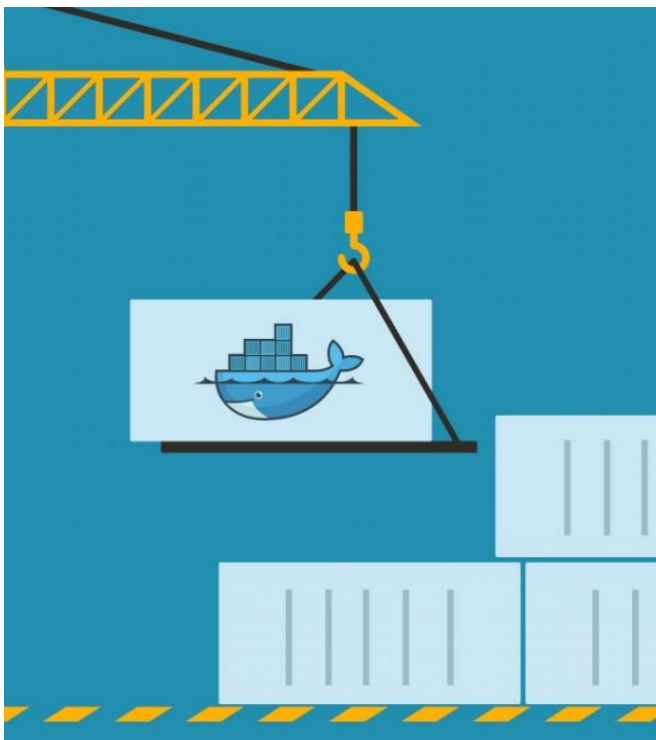
## Demo

Debugging Docker container instances



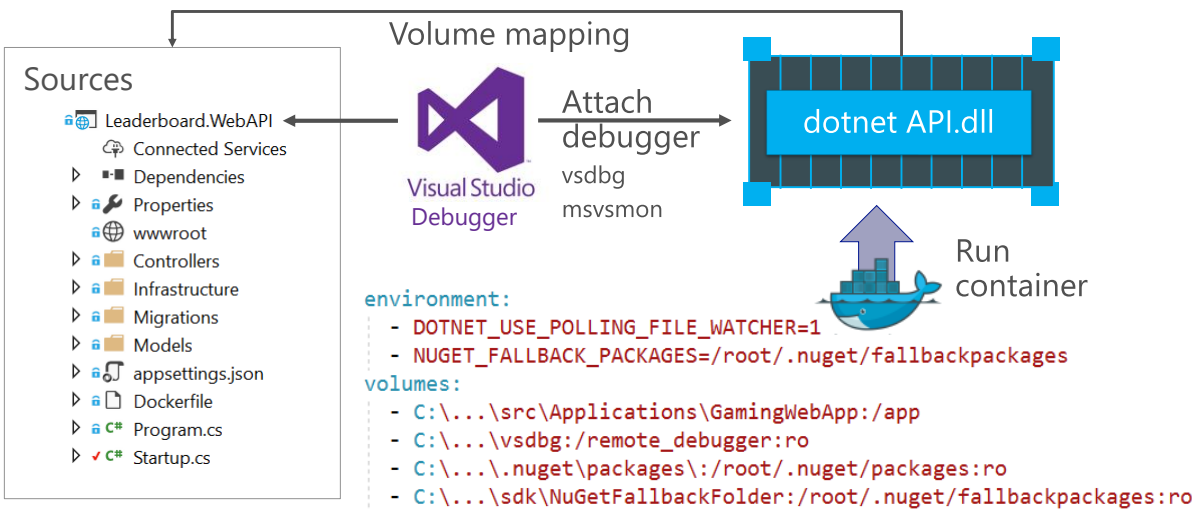
# How it works

Debugging in VS2017



## Debugging containers

### How it works





# Debugging .NET apps in VS2017

## .NET Core

Self-hosted by dotnet.exe driver

**Volume mappings:**

- VSDBG debugger
  - NuGet package cache
  - Source code in app directory
- Different entrypoint (tail -f /dev/null)

## ASP.NET 4.5+

Hosted by IIS (Express)

**Volume mappings:**

- MSVSMON debugger
  - Sources in IIS root website
- No entrypoint

## Debug configuration

**dev** image tag

**source** argument = obj/Docker/empty

- Really empty!

## Release configuration

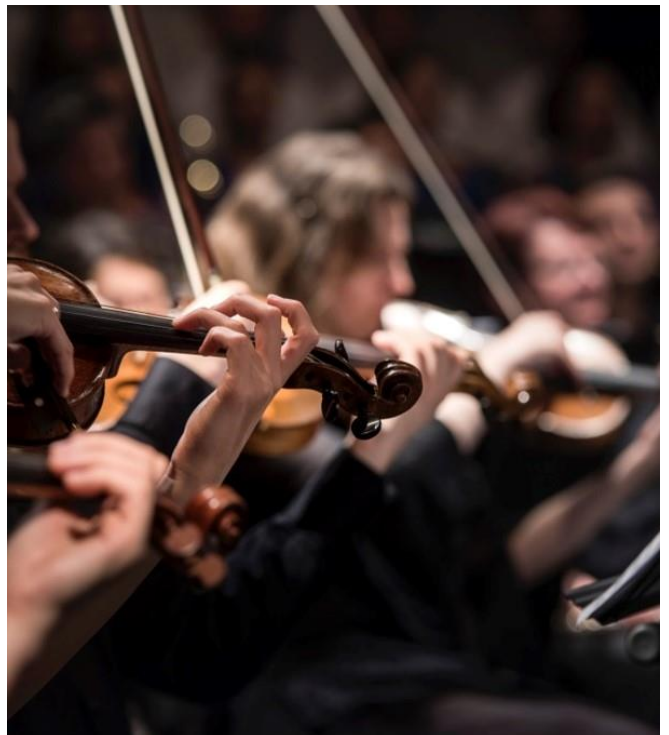
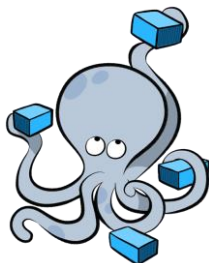
**latest** image tag

**source** argument = obj/Docker/publish

- Binaries
- Views
- Wwwroot
- Dependencies

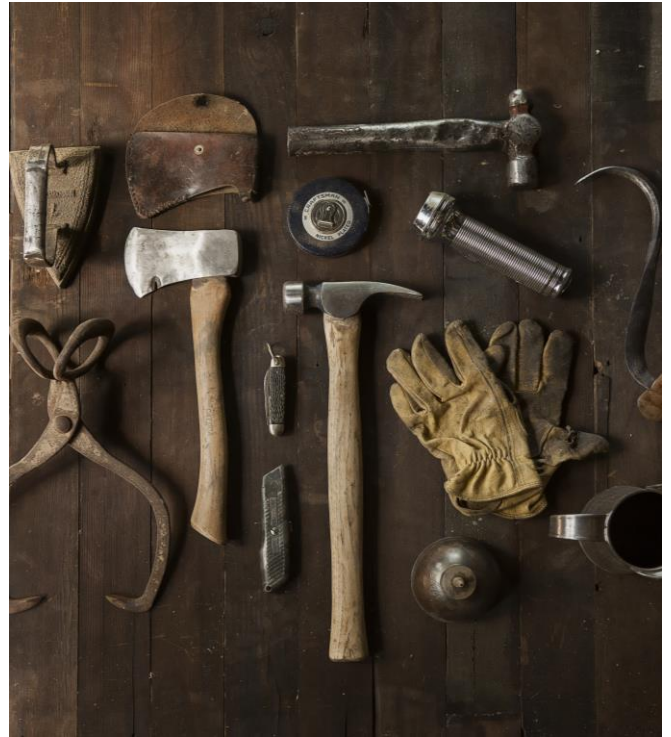
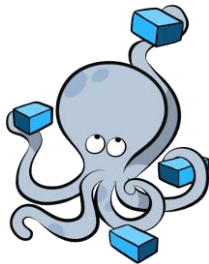
## Demo

Composing applications



# How it works

Docker-compose  
tooling



## Docker-compose structure

version: '3.0'

### services:

#### service-name:

image: ***docker-image***

build: ***how to build***

depends\_on:

- ***other services***

environment:

- ***key/value pairs***

ports:

- ***port mappings***

### networks:

#### network-name:

### volumes:

#### volume-name:



# Composing docker-compose files

```
docker-compose
  -f "docker-compose.yml"
  -f "docker-compose.override.yml"
  -p composition up -d
```

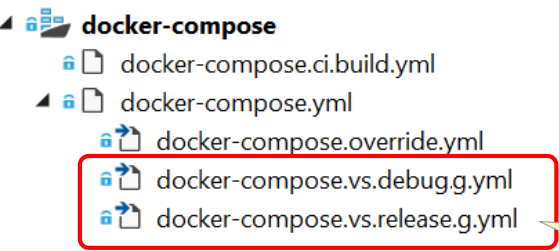
Order of files matters:  
last one wins



# Docker-compose in Visual Studio 2017

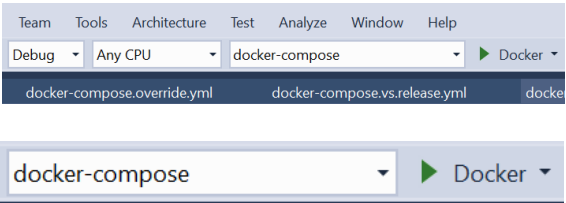
## Building

**docker-compose** builds images  
Separate compose files per configuration  
Automated editing: project-aware



## Debugging

Remote debugging of complete composition  
Attach to multiple running containers



Generated debug and release YAML files for VS2017 15.3+  
Located in obj\Docker\publish

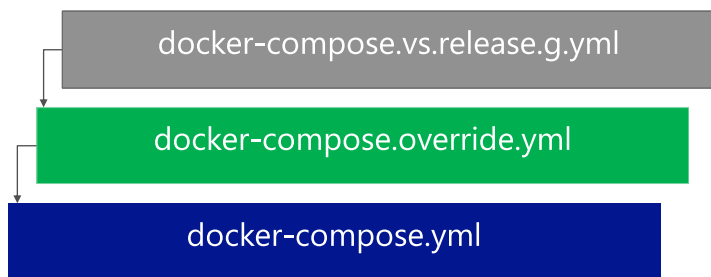
## Choose your own strategy

Visual Studio generated YAML files require some tweaking

# Composing docker-compose files

```
docker-compose  
-f "docker-compose.yml"  
-f "docker-compose.override.yml"  
-p composition up -d
```

Order of files matters:  
last one wins

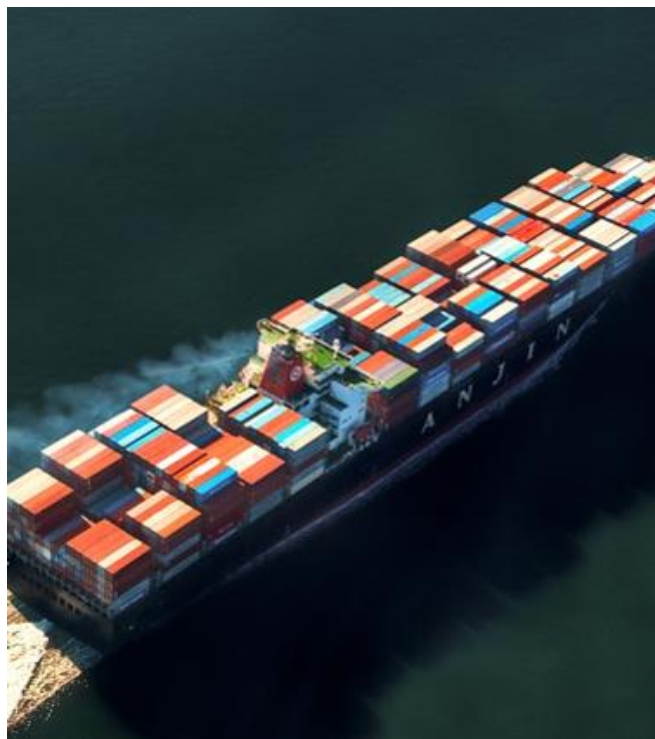


Debug or Release VS2017:

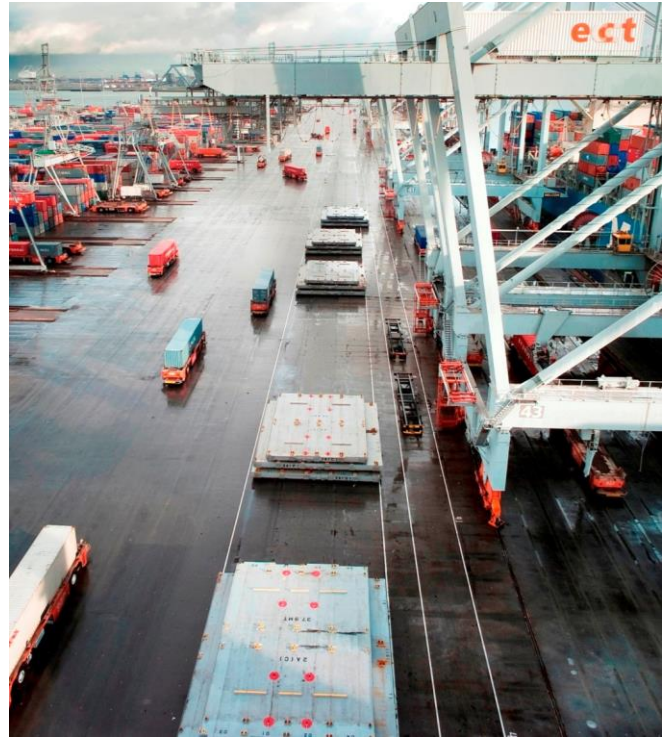
- Volume mappings
- Debugger
- Additional labels

## Demo

Deploying to a registry  
Running in a cluster



# How it works




## A container to build containers

### Specialized container can build composition

Run with command-line tooling without VS2017

Execute on build server (e.g. VSTS or Jenkins) with Docker support

Leverages solution structure

 `docker-compose.ci.build.yml`

```
version: '3'

services:
  ci-build:
    image: microsoft/aspnetcore-build:1.0-2.0
    volumes:
      - ./src
    working_dir: /src
    command: /bin/bash -c "dotnet restore ./TechDays2017Gaming.sln
      && dotnet publish ./TechDays2017Gaming.sln -c Release -o ./obj/Docker/publish"
```

Indicates which .NET Core framework versions are installed



# Summary

