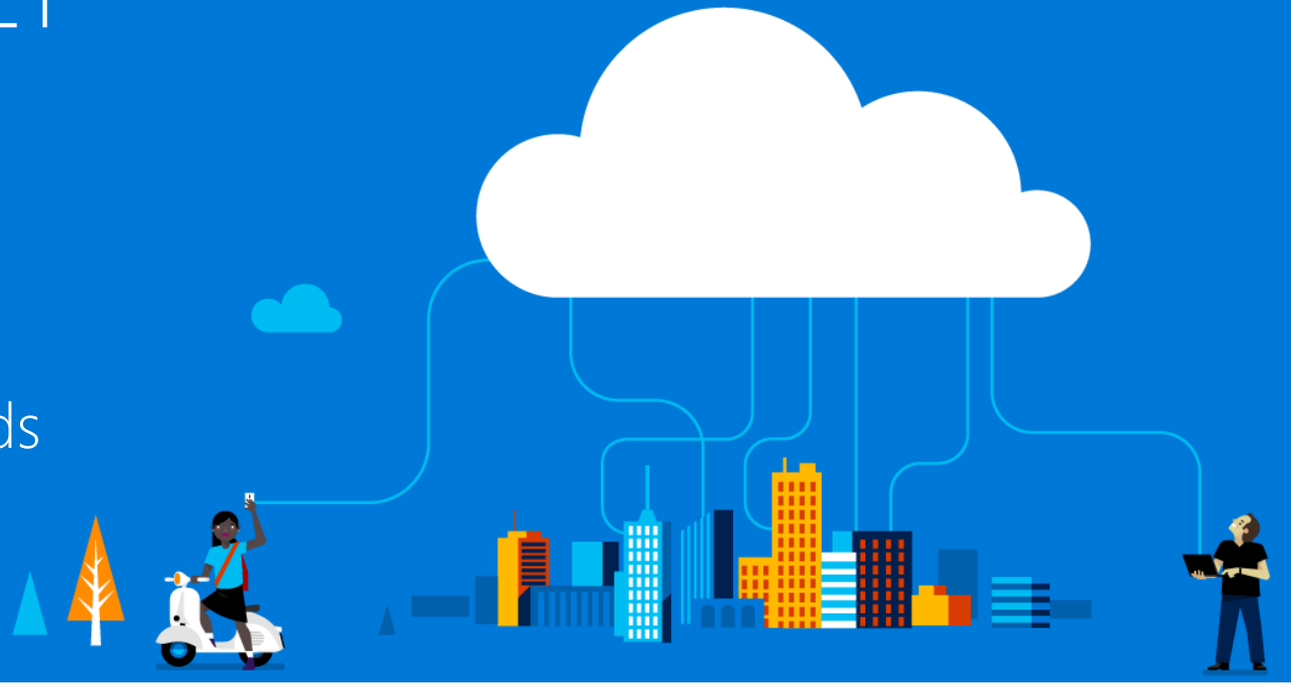
Shifting gears in .NET

Moving to containers for .NET developers

Alex Thissen
Cloud architect at Xpirit, The Netherlands
@alexthissen

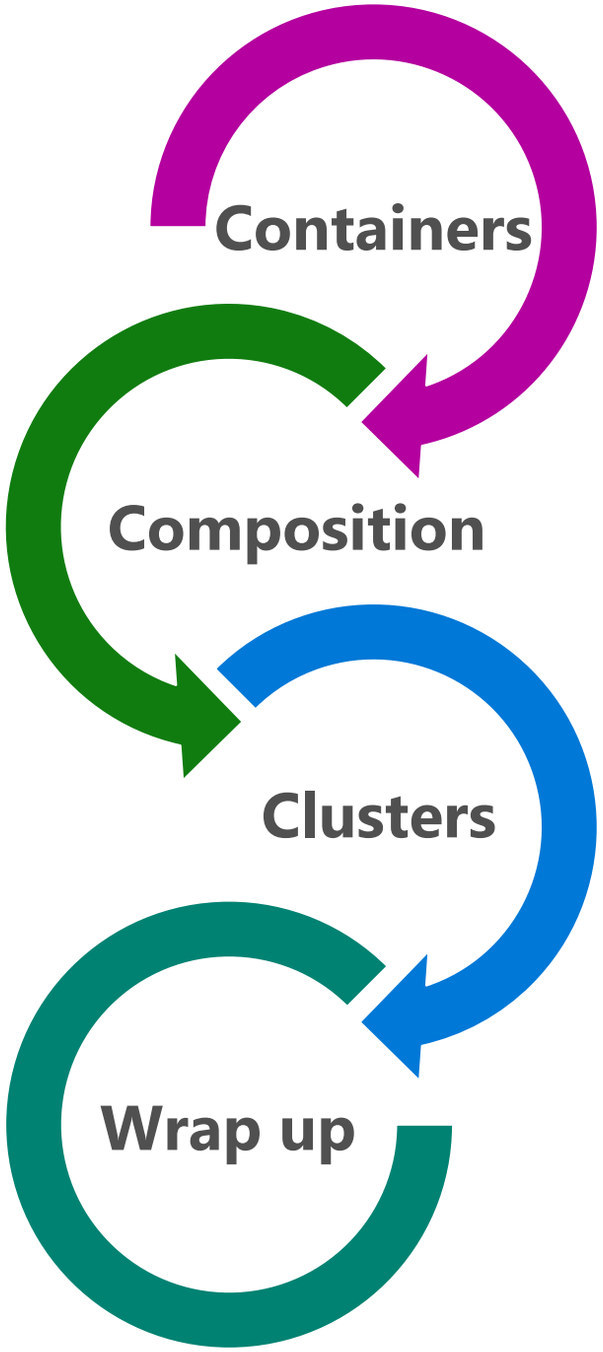


DOT NEXT

Xpirit
Think ahead. Act now.

MVP Microsoft® Most Valuable Professional

Agenda



- Container technology
- Docker
- Visual Studio 2017 tooling

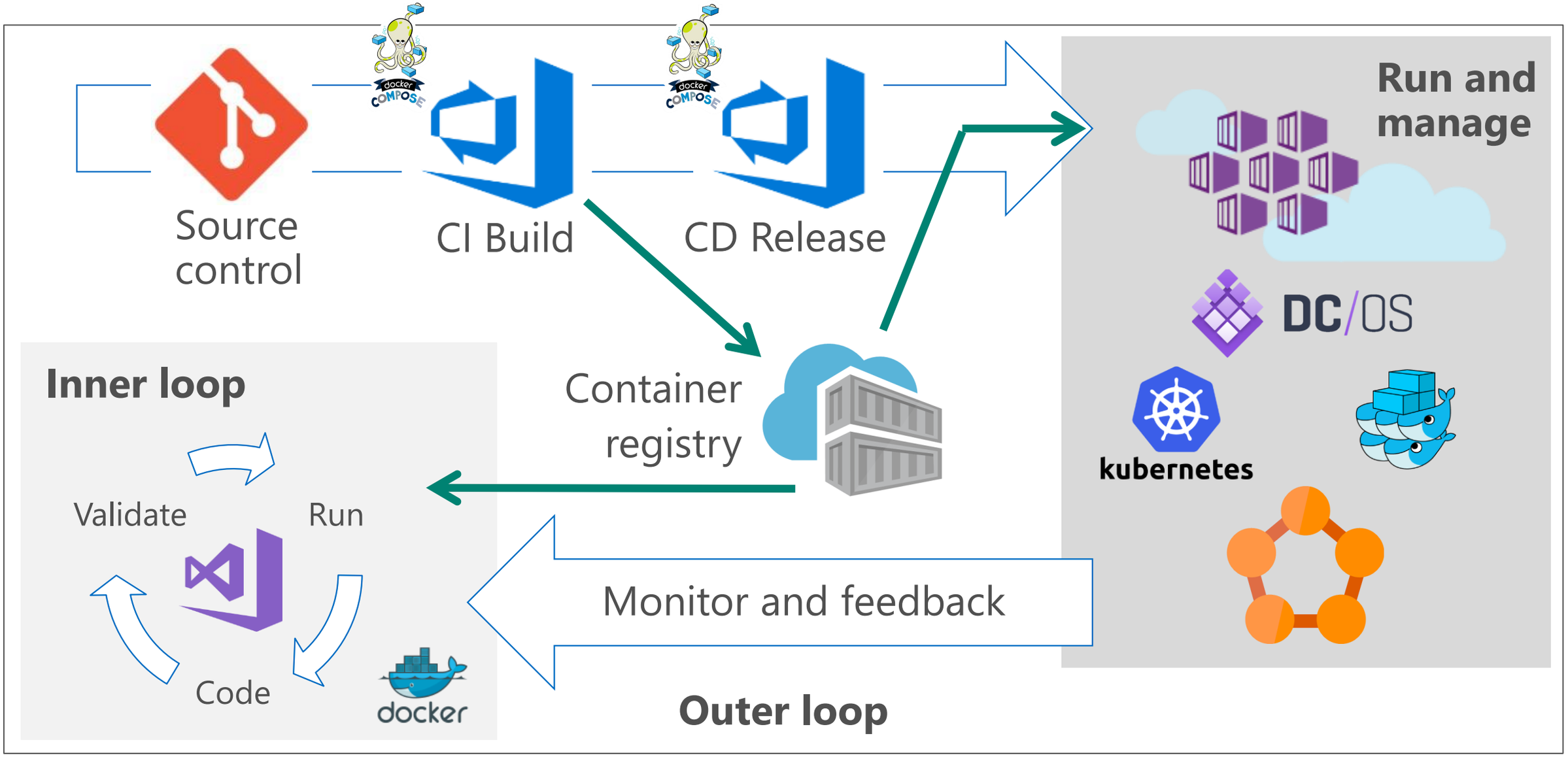
- Docker-compose
- Environments

- Orchestrators
- Containerized application lifecycle

- Summary
- Questions and answers

DOT NEXT

New developer workflow

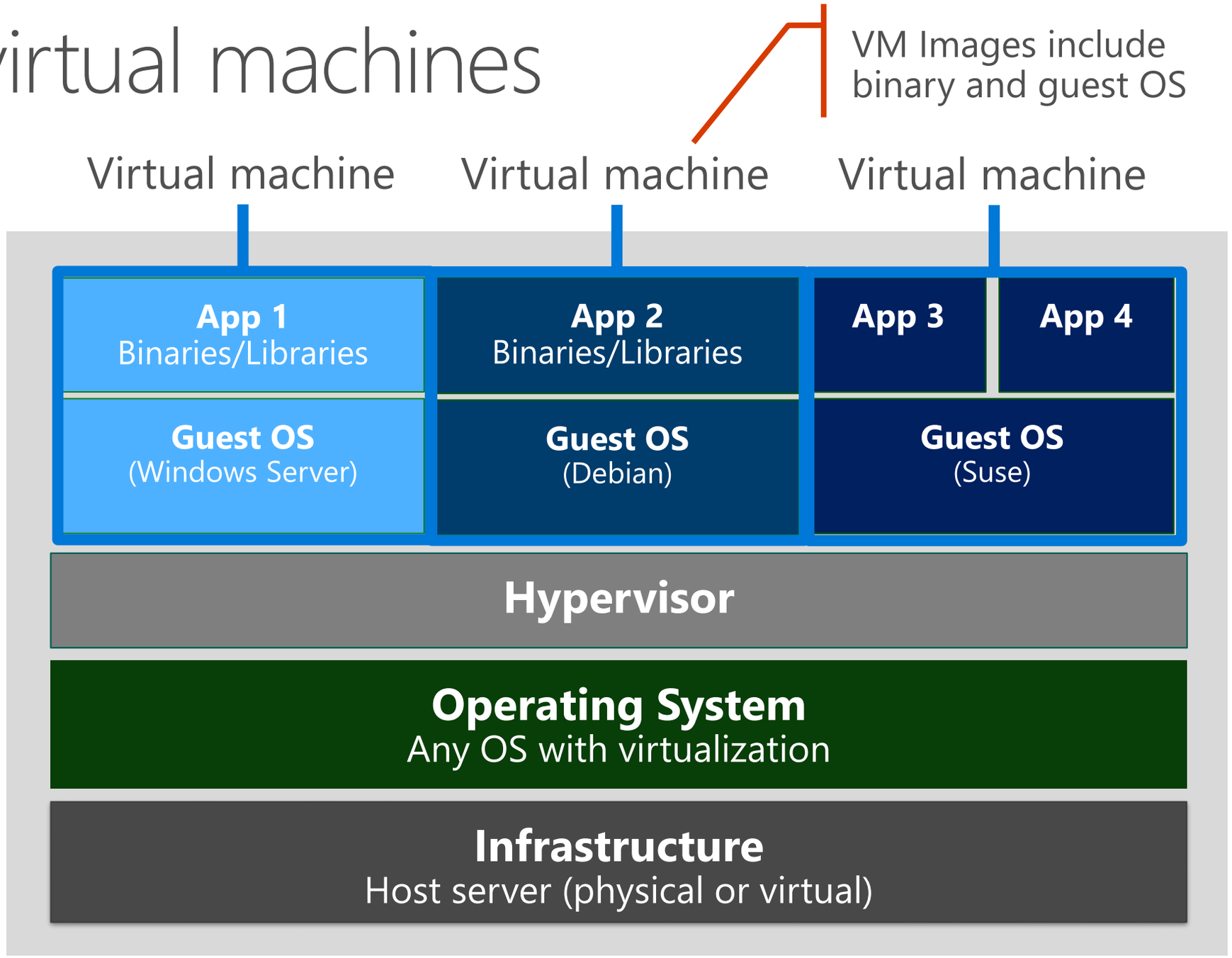


Demo #1
Diving straight into Docker containers

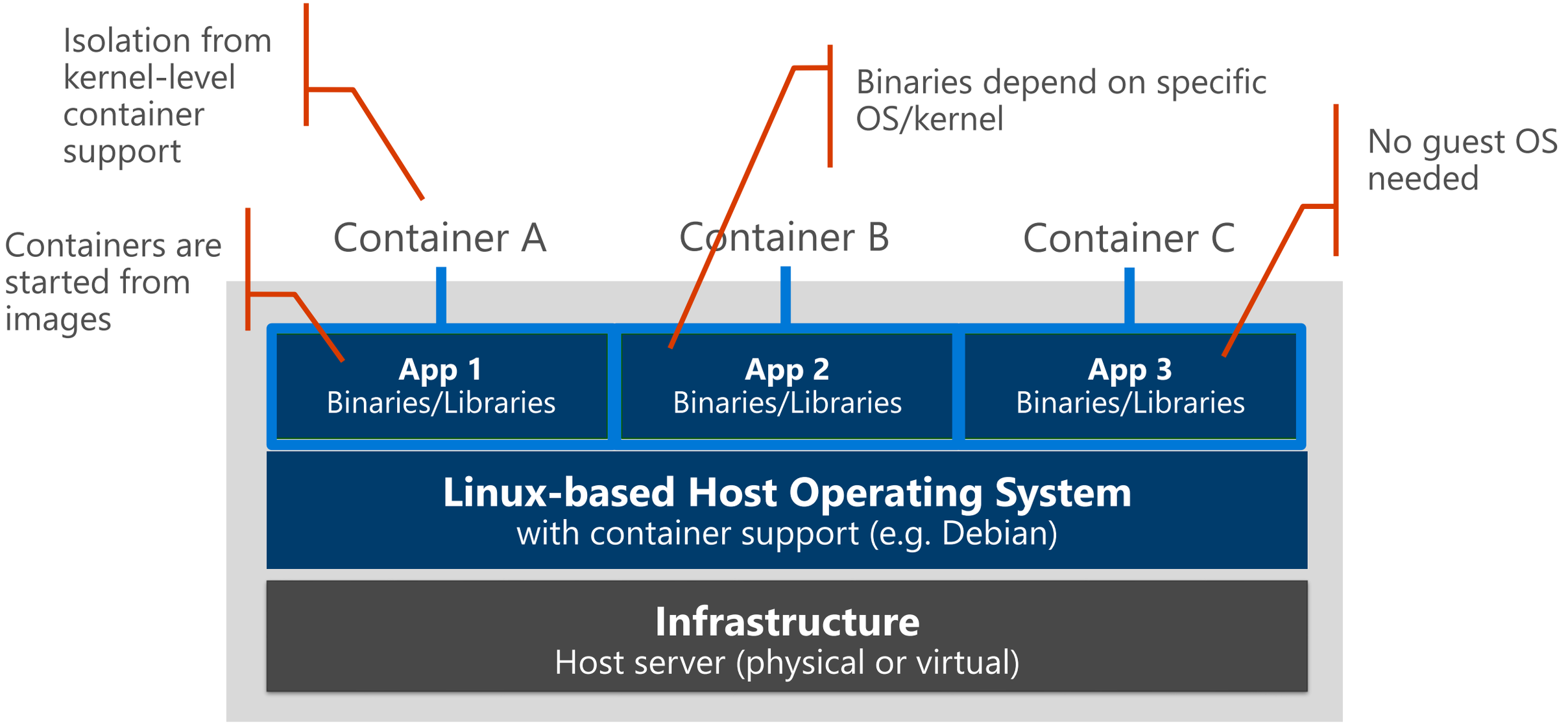
Container technology
Docker 101

From virtual machines

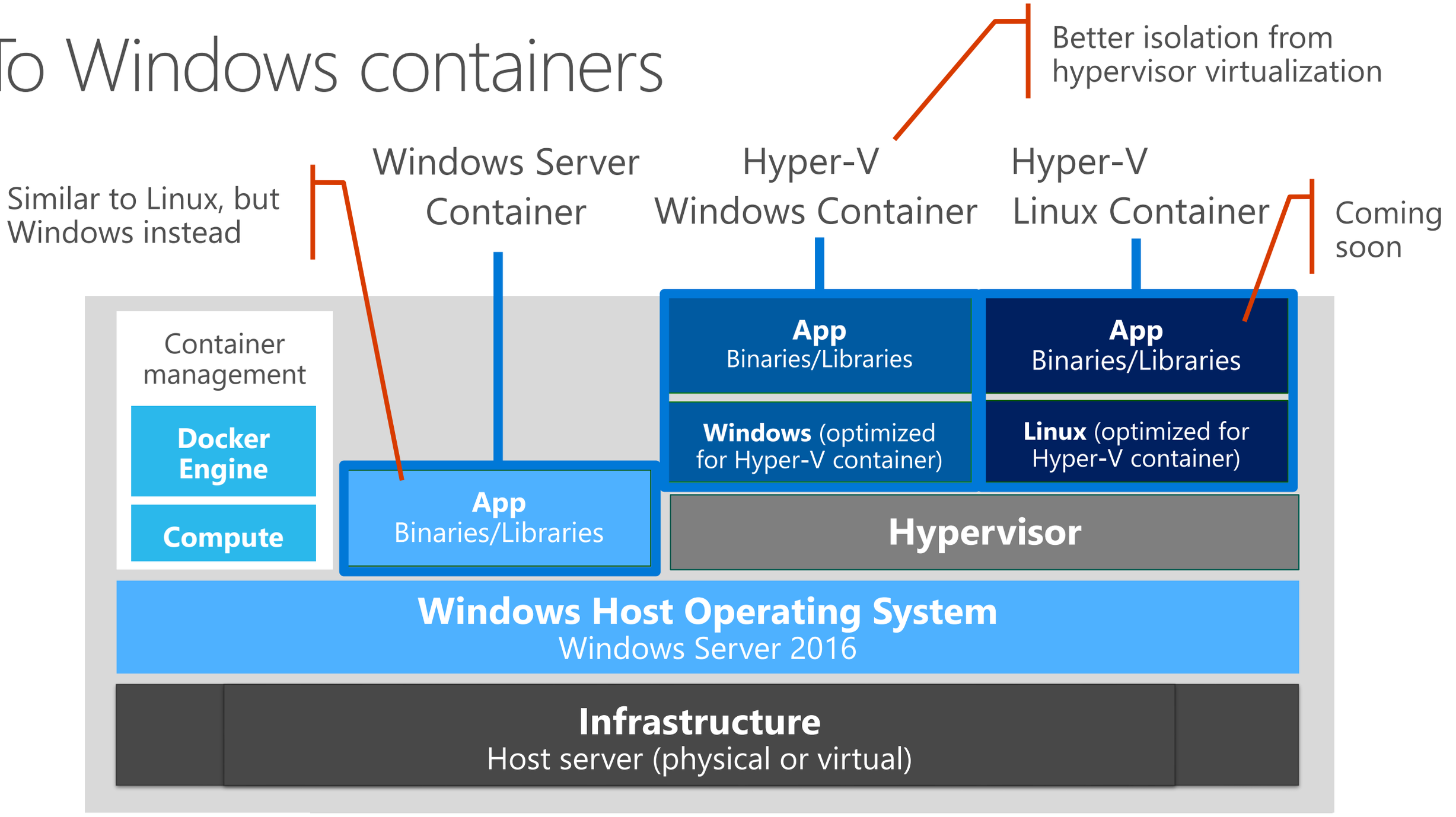
VM Images include binary and guest OS



To Linux containers



To Windows containers



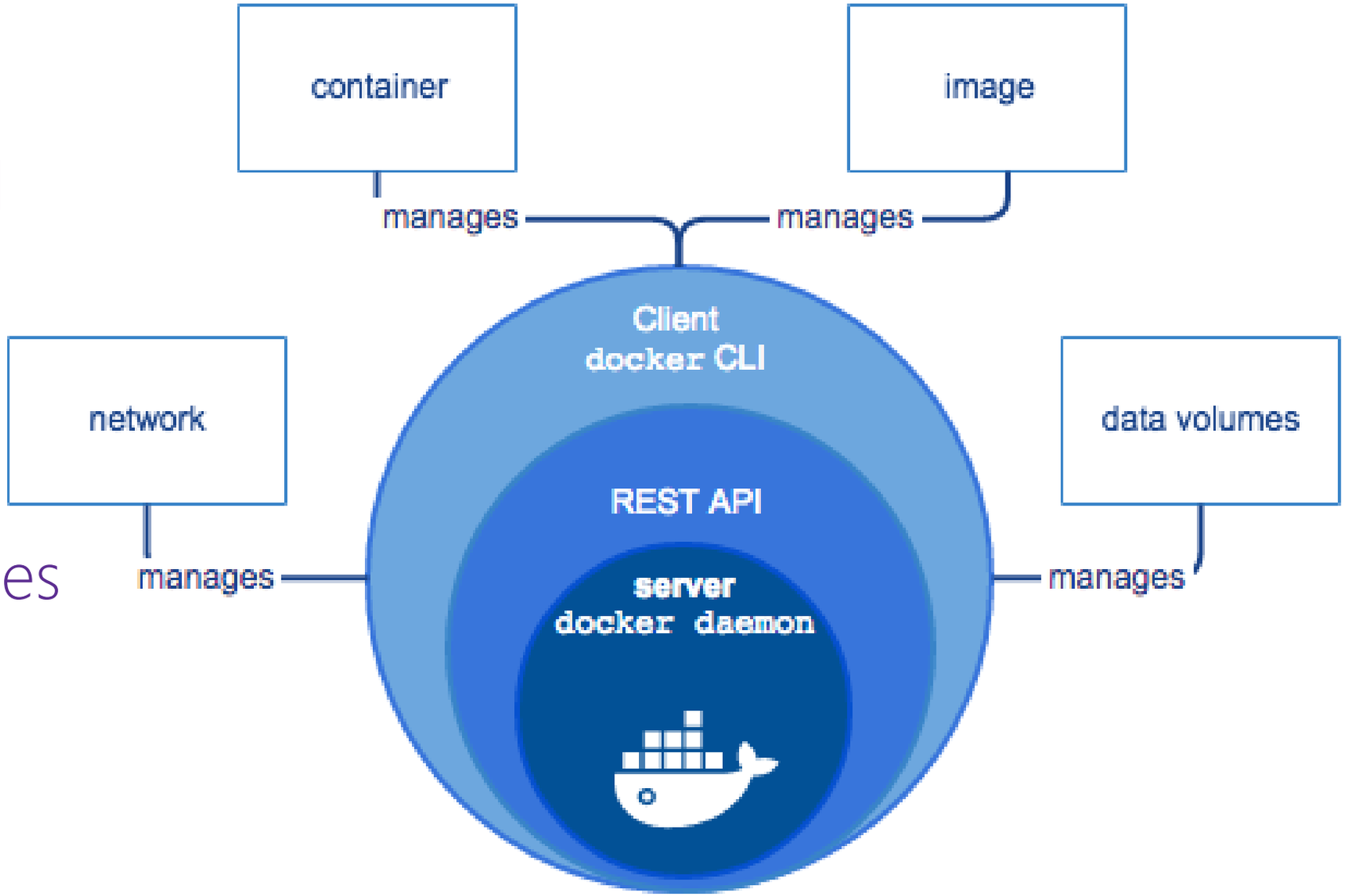
Docker

Docker CLI to control container images and instances

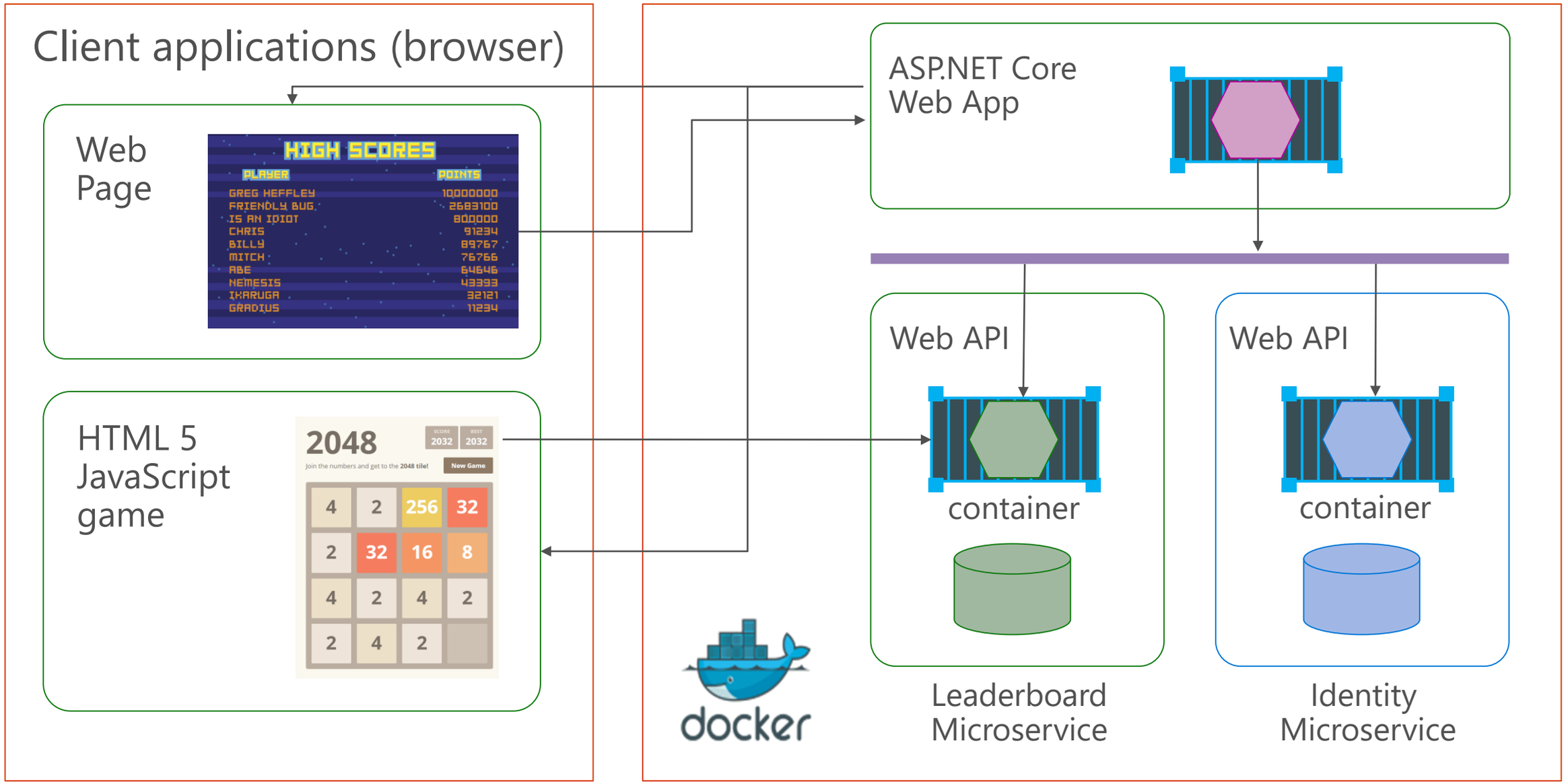
docker pull …
docker run –it …
docker commit

Builds container images

docker build
docker push



Demo application reference architecture



Demo #2
Highscore application

Docker containers and Visual Studio 2017

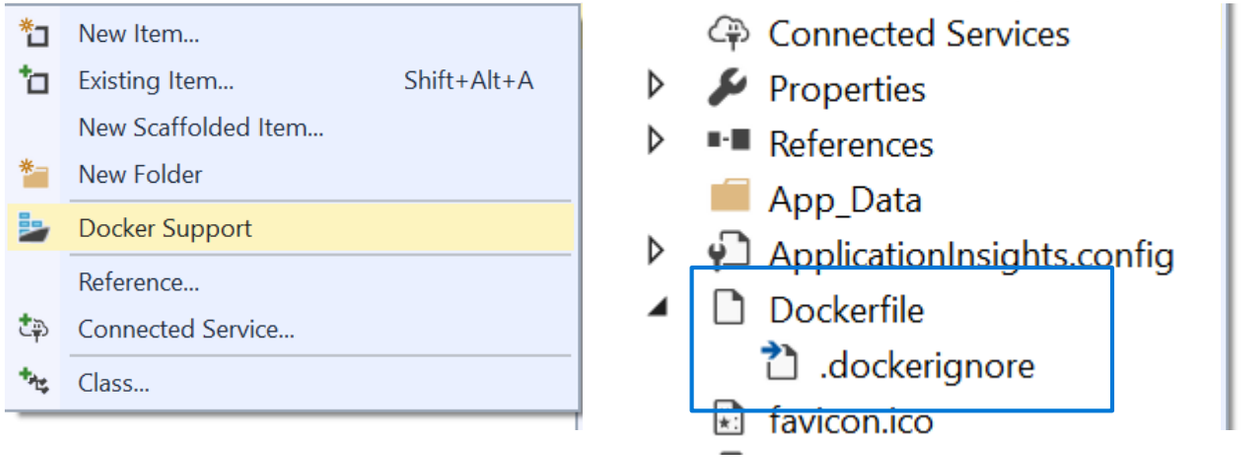
Docker support in Visual Studio 2017

Native (official) docker tooling

Docker, Compose recognized by project system.
IntelliSense for Dockerfile and docker-compose files
Artifacts work with CLI tooling as well. VS not required.

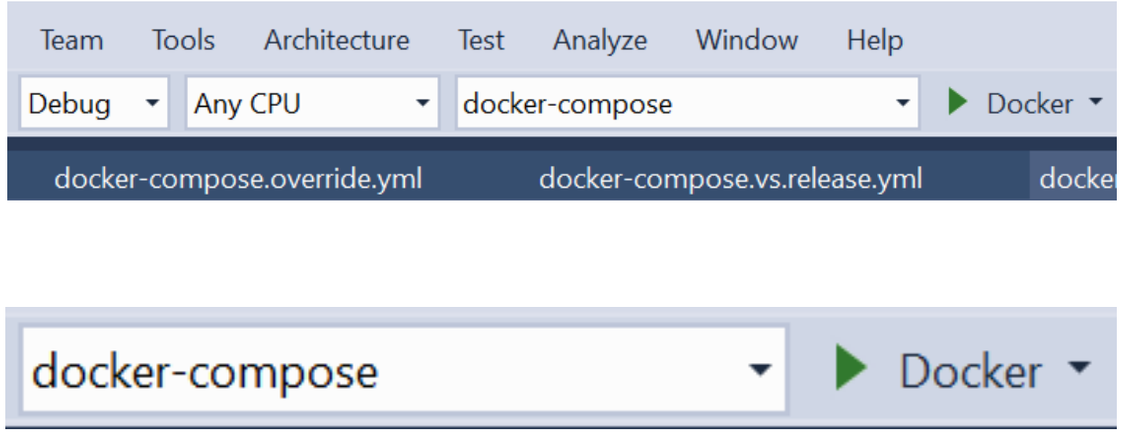
Building individual images

Container image per project from `Dockerfile`



Debugging

Remote debugging into container
Hot-editing files without rebuild of containers



VS2017 container building

Dockerfile per project

Special **`source`** argument for app binaries, or obj/Docker/publish as fallback

Expected to use docker-compose

Can be built without VS2017

Command-line

Visual Studio Code

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

Docker image layers for .NET Core

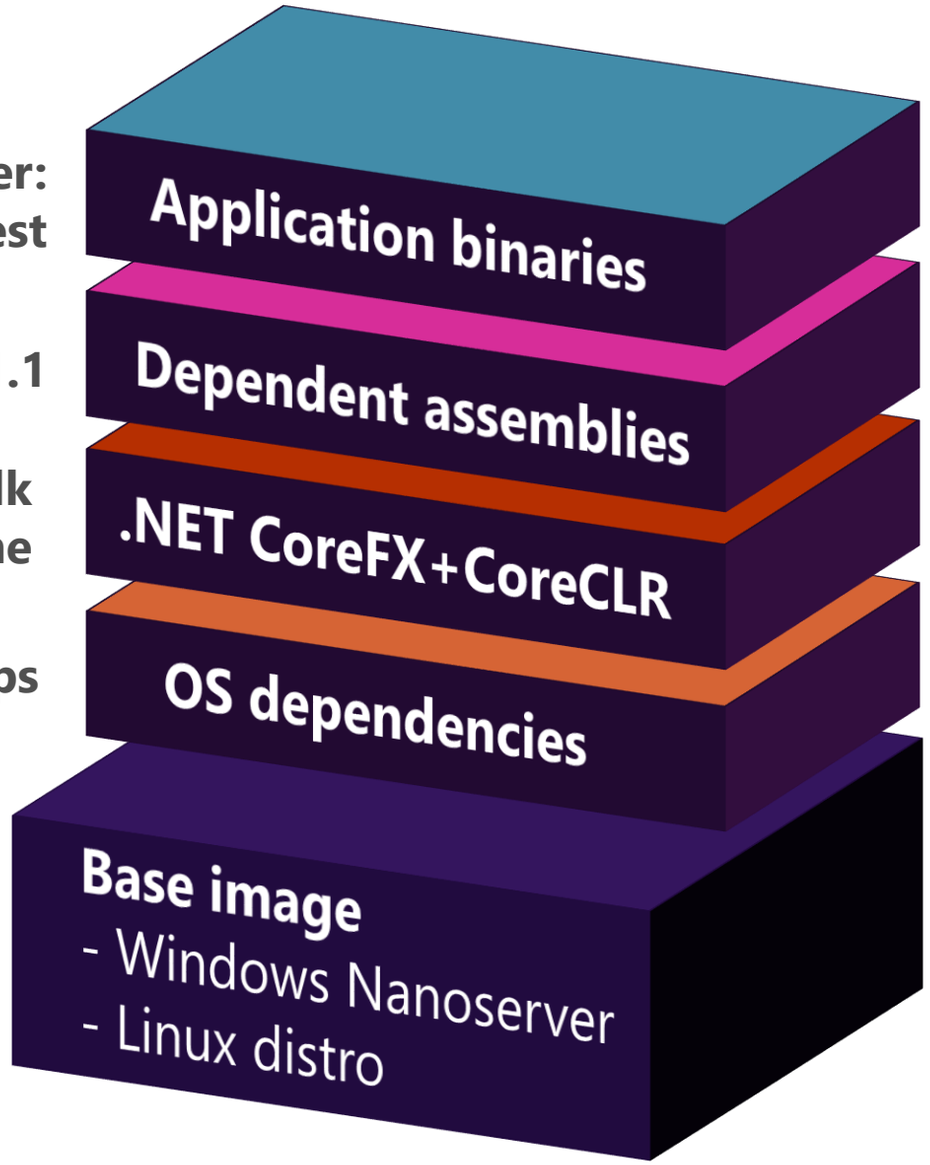
Your application layer:
e.g. dotnext/gamingwebapp:latest

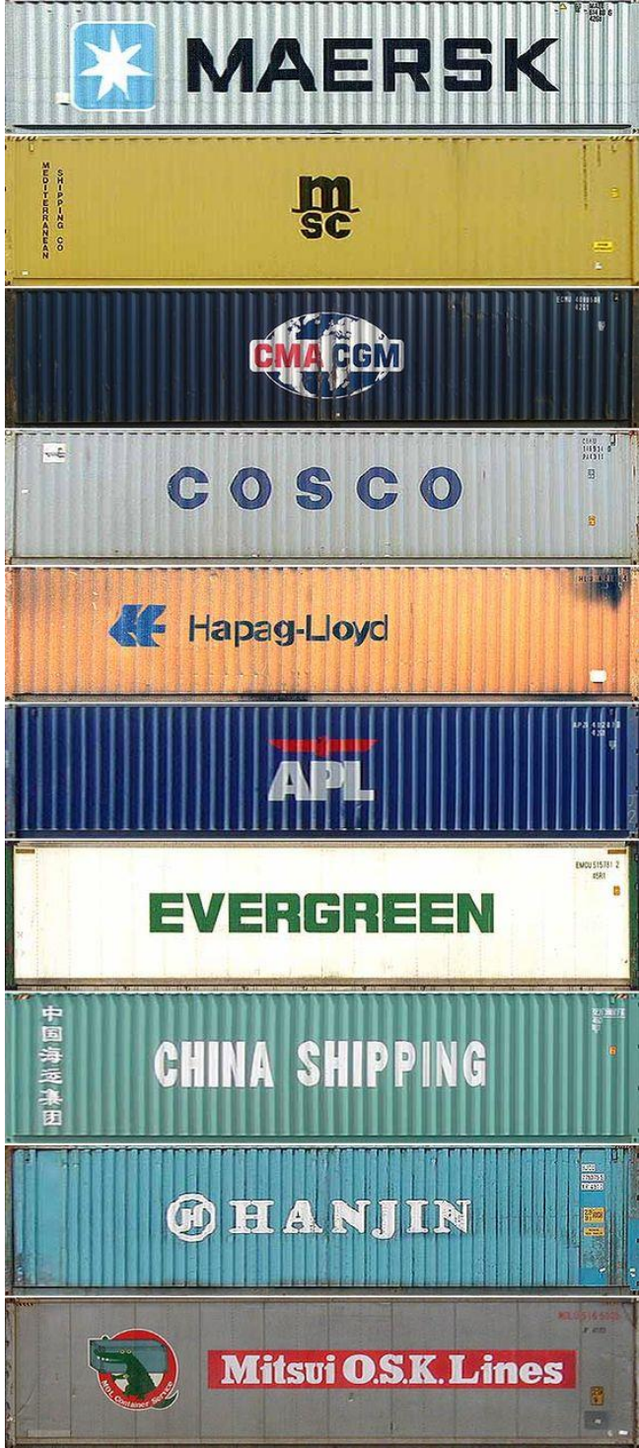
microsoft/aspnetcore:1.1.1

microsoft/dotnet:1.0.0-sdk
microsoft/dotnet:1.1.1-runtime

microsoft/dotnet:1.1.1-runtime-deps

microsoft/nanoserver:10.0.14393.1066
-or- debian/jessie





Docker image layers for .NET Core

Convention based

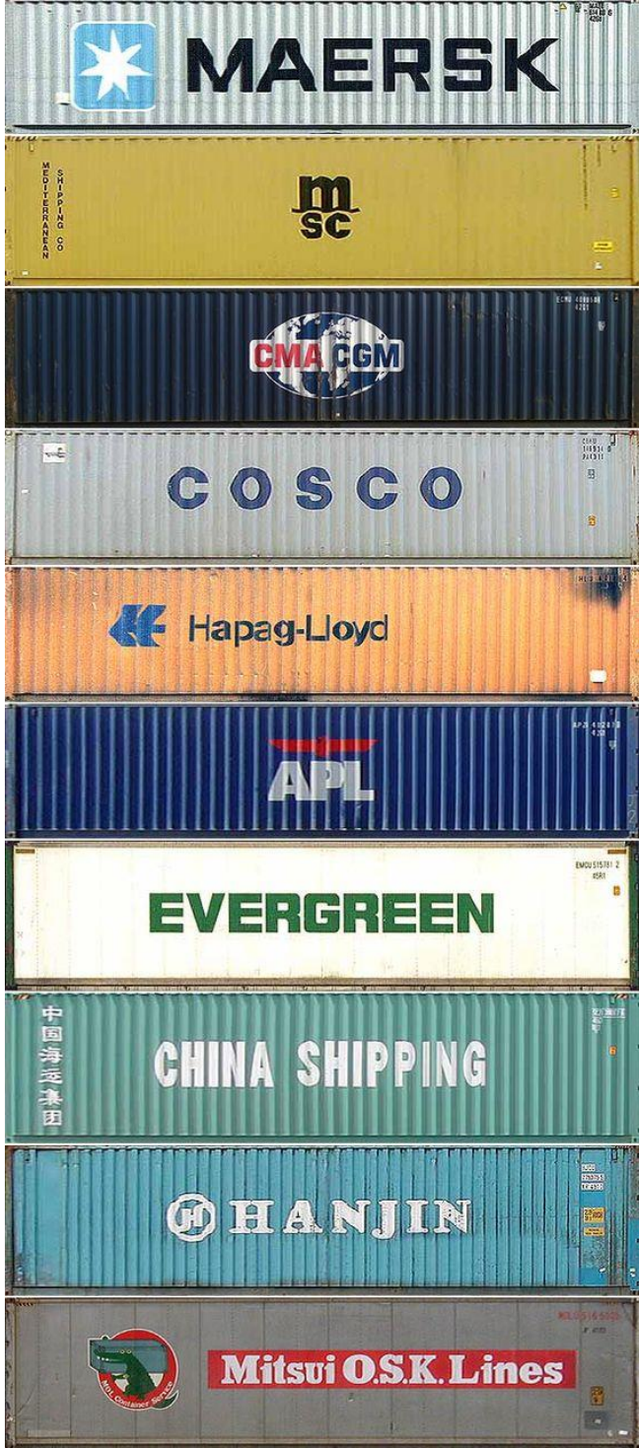
Version-less is latest
Fewer digits implies highest sub-digit
.0 for LTS version

Windows or Linux

Determined by tag:
Add **-nanoserver** for Windows Nano Server

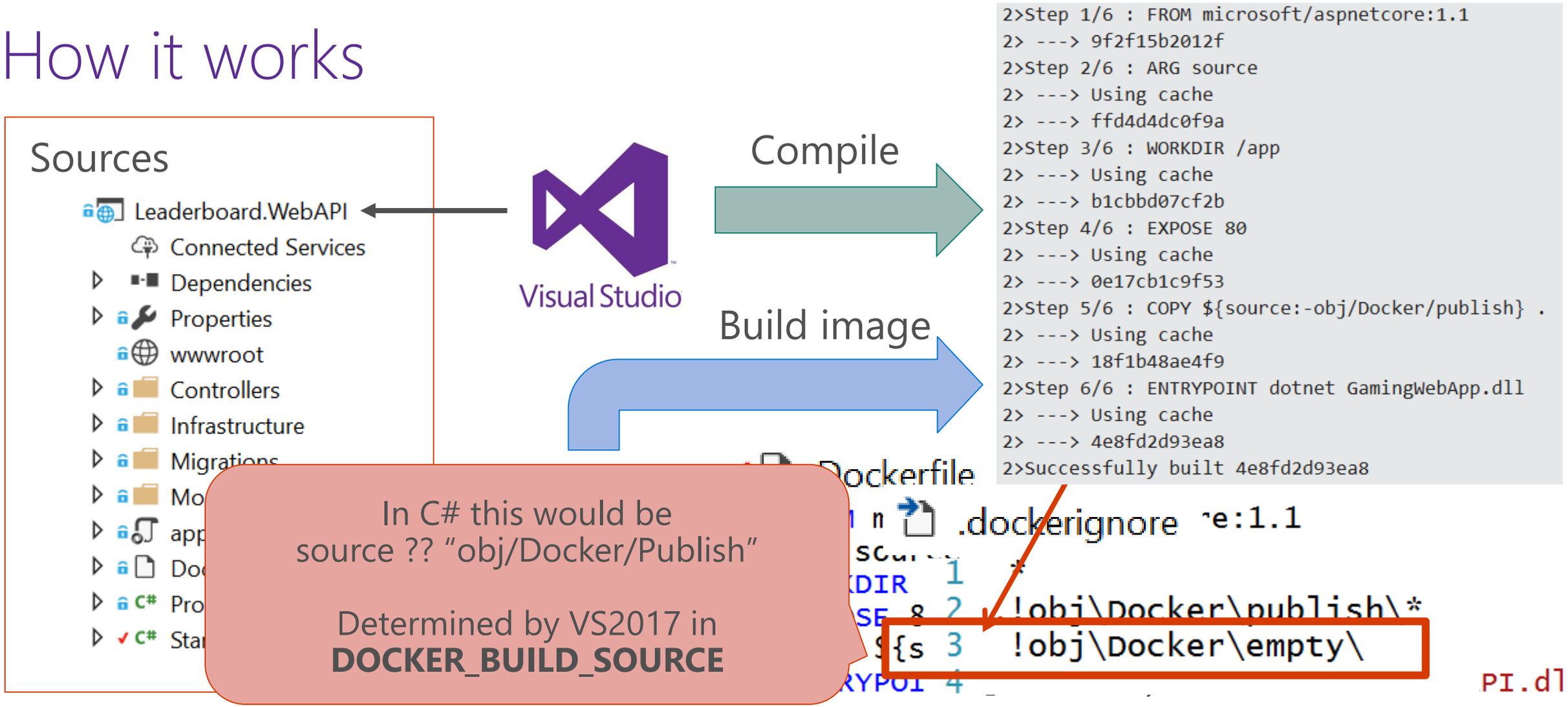
Find Dockerfile images in registries:

Official: Docker Store:
e.g. https://store.docker.com/images/dotnet
Public/private: Docker Hub or Azure Container Registry



Building containers from VS2017

How it works

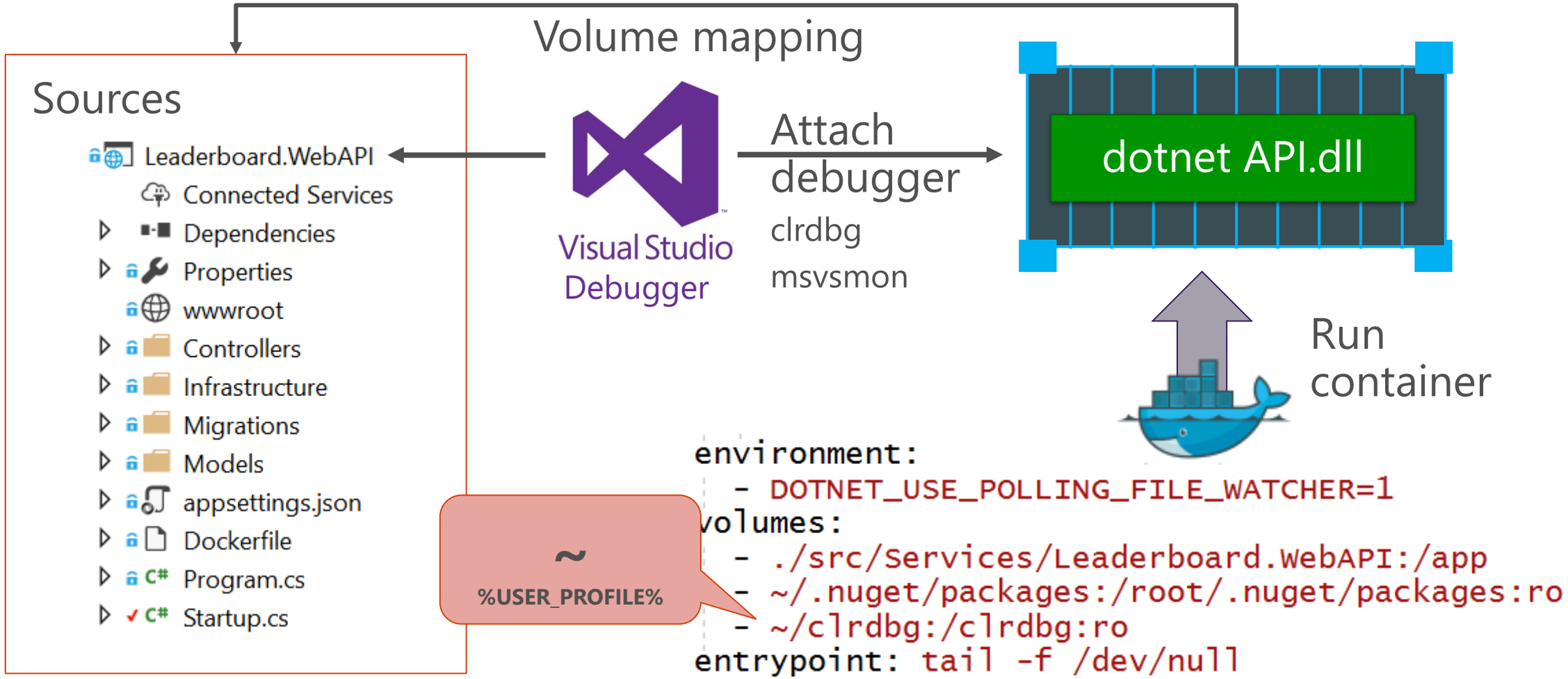


Demo #3

Running containers in VS2017

Debugging containers

How it works



Debugging .NET apps in VS2017

.NET Core

Self-hosted by dotnet.exe driver

Volume mappings:

- CLRDBG 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 #4

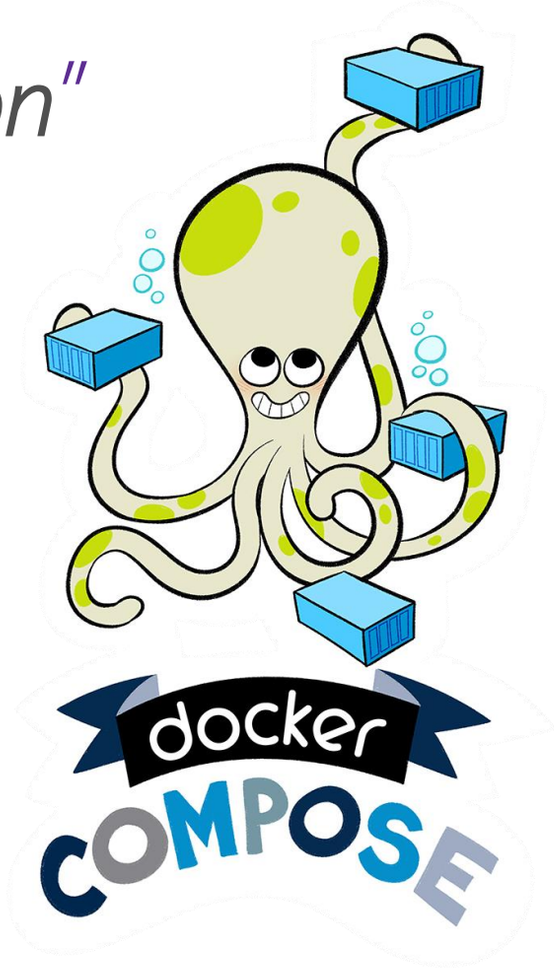
Debugging containers in VS2017

Composing applications from containers

Composing container solutions

Docker-compose:

"Orchestration tool for container automation"



Single command

Work with multiple containers: build, run, scale, heal

Scoped mostly at single-host scenarios (Use clusters for multi-host)

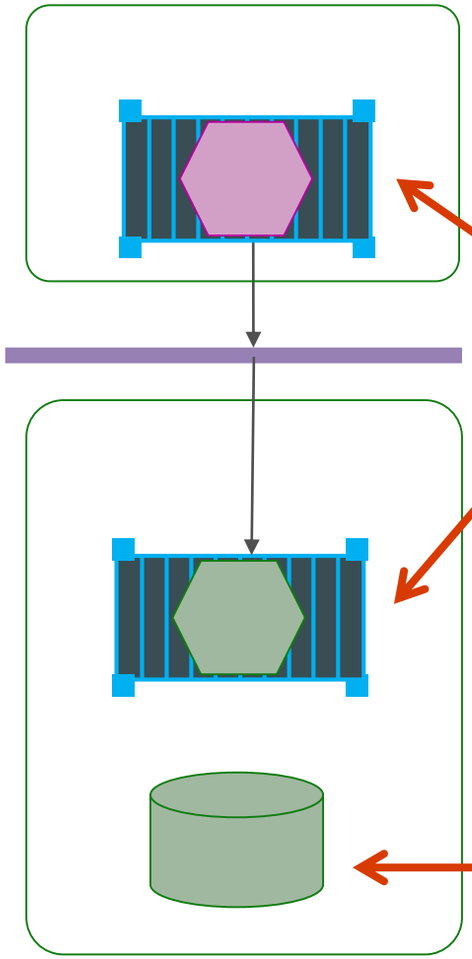
YAML files describe composition

Configuration file for images, build, services, volumes, networks, environments

Same syntax for deploying on clusters (version 3.0+)

Allows hierarchies and overriding

Docker compose demo example



```
services:
  leaderboard.webapi:
    image: leaderboard.webapi
    build: ...
    depends_on:
      - sql.data

  gamingwebapp:
    image: gamingwebapp
    build: ...
    depends_on:
      - leaderboard.webapi

  sql.data:
    image: microsoft/mssql-server-linux
```

Docker-compose structure

```
version: '2.1'
services:
  service-name:
    image: docker-image
    build: how to build image
    depends_on:
      - other services
    environment:
      - key/value pairs
    ports:
      - port mappings
```

VS2017 currently uses 2.1
Support for 3.x coming soon

```
networks:
  network-name:

volumes:
  volume-name:
```

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

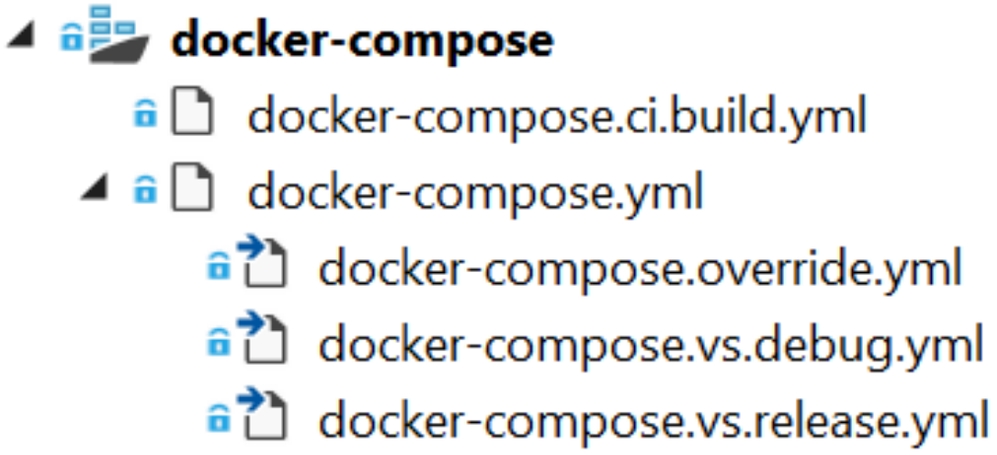
```
version: '2.1'

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

Demo #5
Building compositions

Docker-compose in Visual Studio 2017

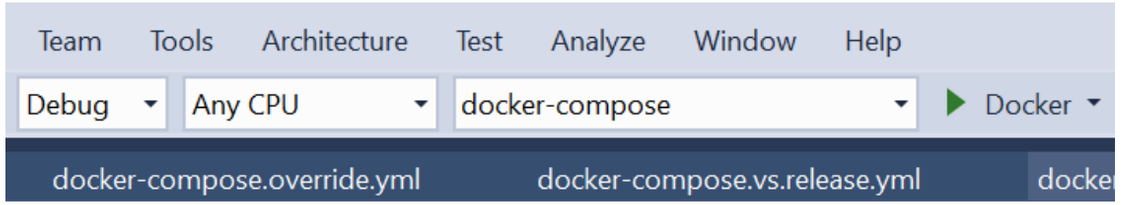
Building

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



Debugging

Remote debugging of complete composition
Attach to multiple running containers

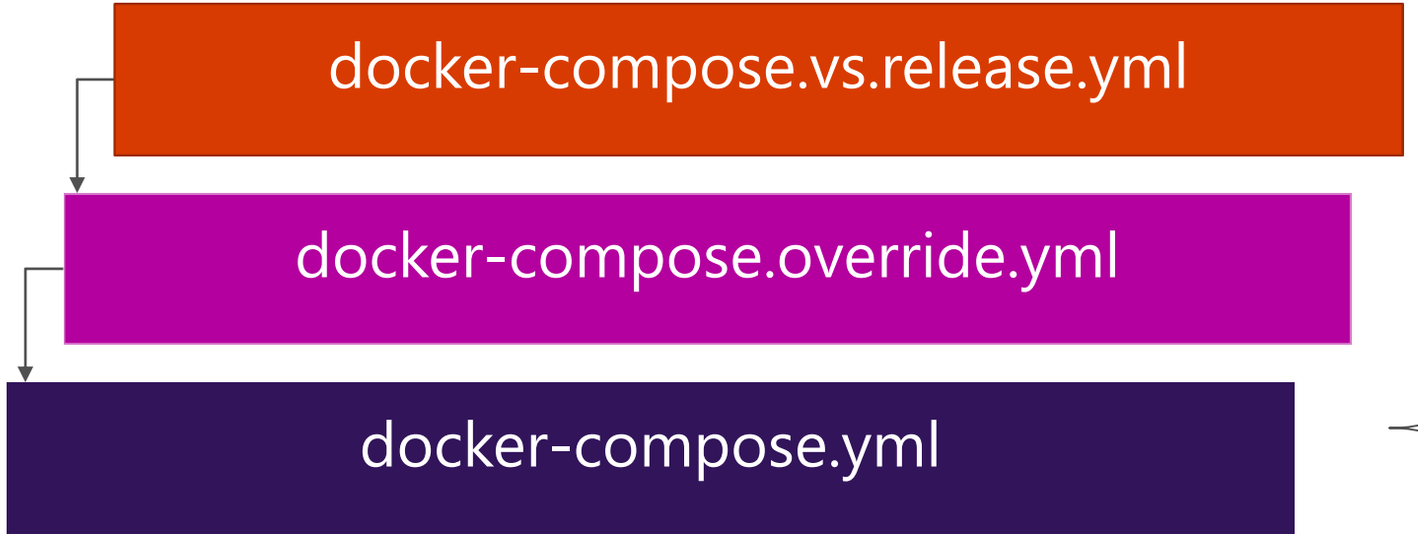


Choose your own strategy

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

Base compositions:

- Services
- Images

Environment overrides:

- Additional labels
- Port mappings
- Environment variables
- Dependencies between services
- Network

Demo #6
Working with compositions

Environments everywhere

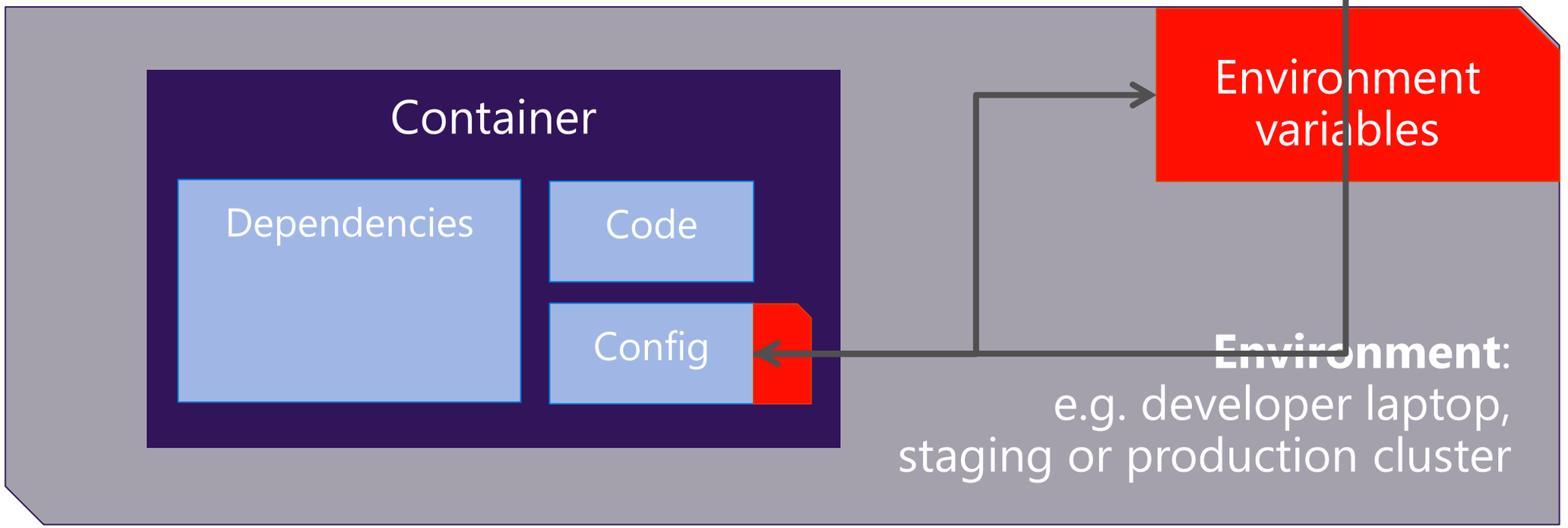
Container images are immutable

Environments are not

```
docker run -it --env key=value dotnext/app
```

appsettings.json

```
{
  "ConnectionString": "Server=tcp:…",
  "Logging": {
    "IncludeScopes": false,
    "LogLevel": {
      "Default": "Warning"
    }
  }
}
```



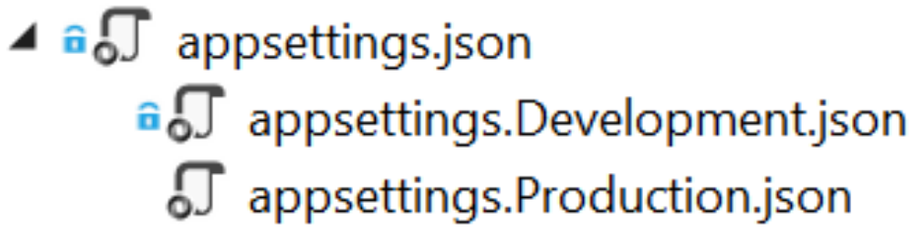
Working with environments in ASP.NET

Bootstrapping environment variables

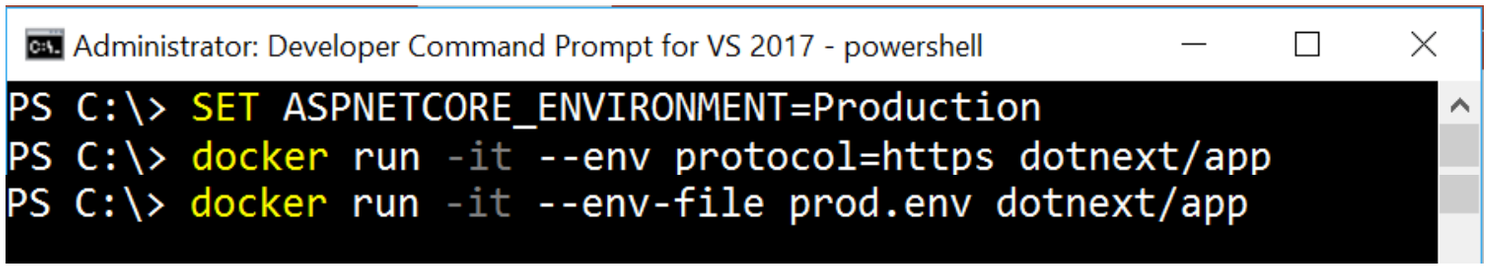
```
public Startup(IHostingEnvironment env)
        {
            var builder = new ConfigurationBuilder()
                .SetBasePath(env.ContentRootPath)
                .AddJsonFile("appsettings.json", optional: false, reloadOnChange: true)
                .AddJsonFile($"appsettings.{env.EnvironmentName}.json", optional: true)
                .AddEnvironmentVariables();
            Configuration = builder.Build();
        }
```

Adds environment variables from multiple places

Settings files



ENV variables



Environmental variables overrides

Docker-compose

Development

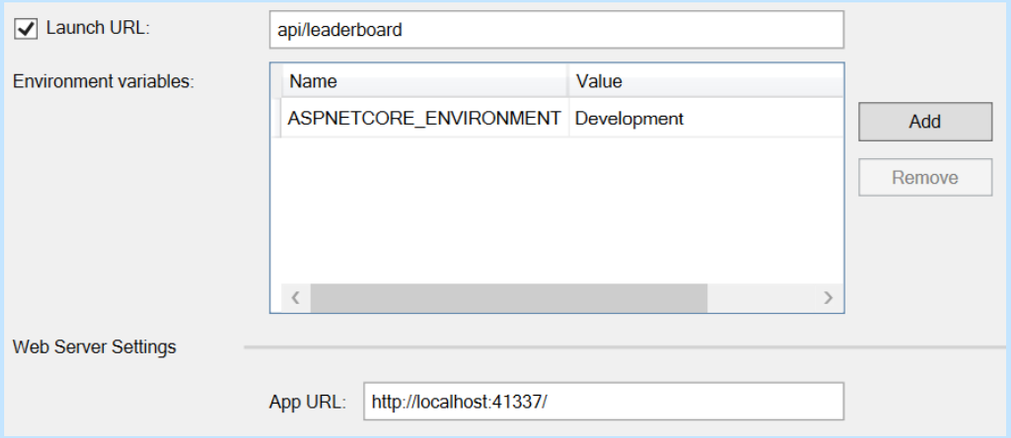
```
environment:
  - ASPNETCORE_ENVIRONMENT=Development
  - ASPNETCORE_URLS=http://0.0.0.0:1337
  - ConnectionString=Server=sql.data;…
```

Production

```
environment:
  - ASPNETCORE_ENVIRONMENT=Production
  - ASPNETCORE_URLS=http://0.0.0.0:80
  - ConnectionString=DOCKERSECRETS_KEY
```

Remember layering of compose files

Non-container development




```
{
  "ConnectionString": "Server=tcp:…",
  "Logging": {
    "IncludeScopes": false,
    "LogLevel": {
      "Default": "Warning"
    }
  }
}
```

Retrieving configuration variables (option 1)

```
// Key/value collection with indexer
Configuration["LeaderboardBaseUrl"];
Configuration["ConnectionStrings:LeaderboardContext"];
// Or special extension methods
Configuration.GetConnectionString("LeaderboardContext");
```

```
// appsettings.json
{
  "LeaderboardBaseUrl": "http://localhost:1337/api",
  "ConnectionStrings": {
    "LeaderboardContext": "Server=tcp:127.0.0.1,3433; …"
  }, …
}
```

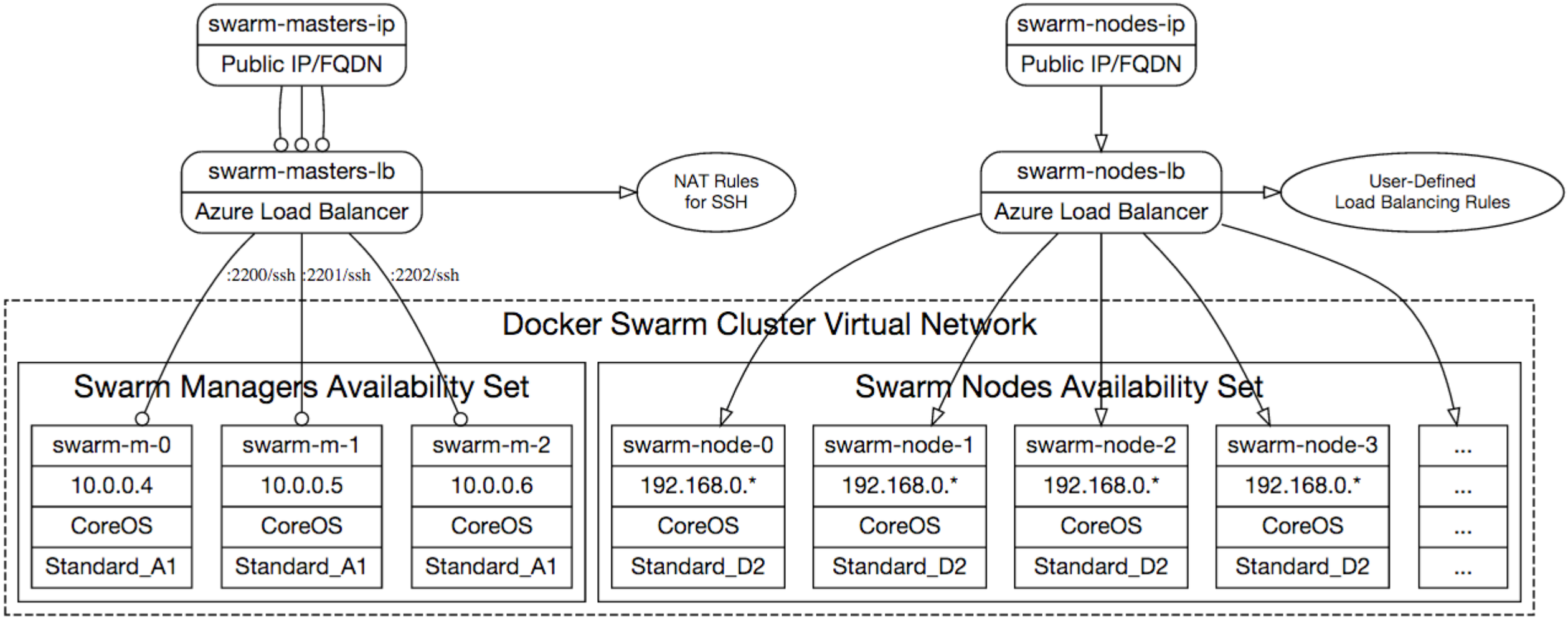
Retrieving configuration variables (option 2)

```
public void ConfigureServices(IServiceCollection services)
{
  // Add framework services.
  services.Configure<WebAppSettings>(Configuration);
  services.AddMvc();
}

public class HomeController : Controller
{
  private readonly IOptionsSnapshot<WebAppSettings> settings;

  // Inject snapshot of settings
  public HomeController(IOptionsSnapshot<WebAppSettings> settings) {
    this.settings = settings;
  }
```

```
public class WebAppSettings {
  public string Setting1 …
  public int Setting2 …
}
```

Moving to container clusters

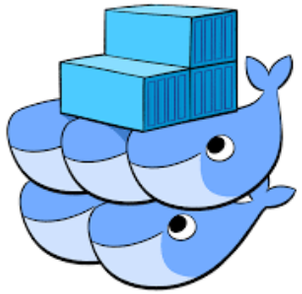
Clusters of container hosts



Cluster orchestrators

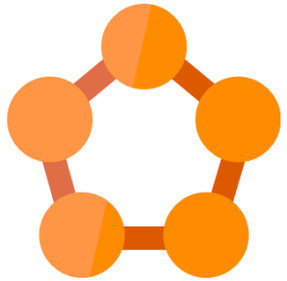
DC/OS

Mesos DC/OS



Docker Swarm

Google Kubernetes



Azure Service Fabric

Azure Container Service

Streamlined provisioning

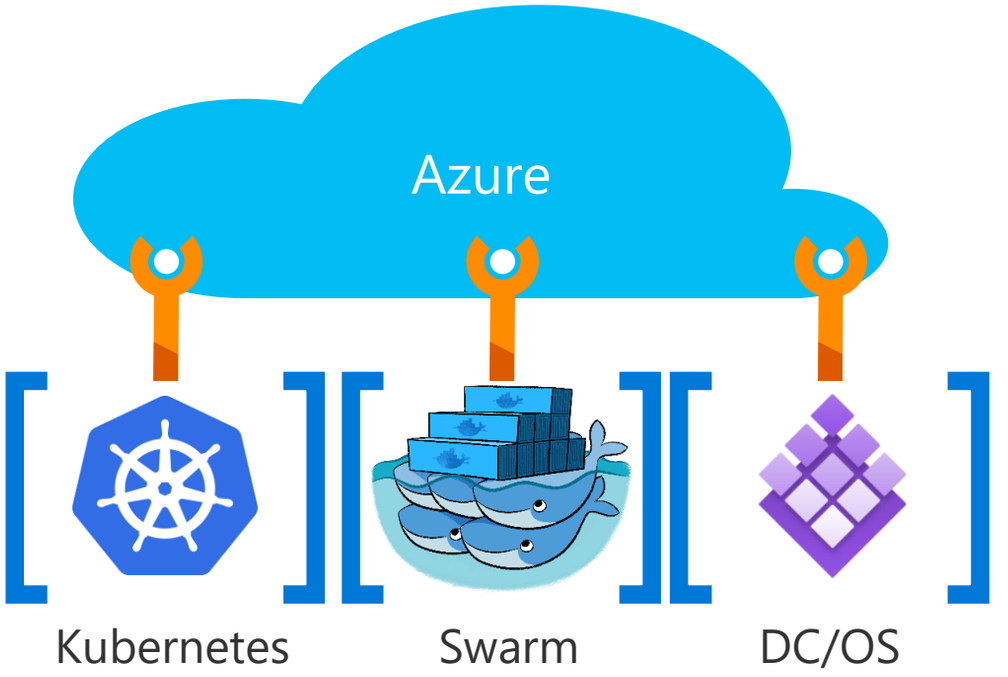
Kubernetes
Docker Swarm
DC/OS

Standard Docker tooling and API support

Linux and Windows Server containers

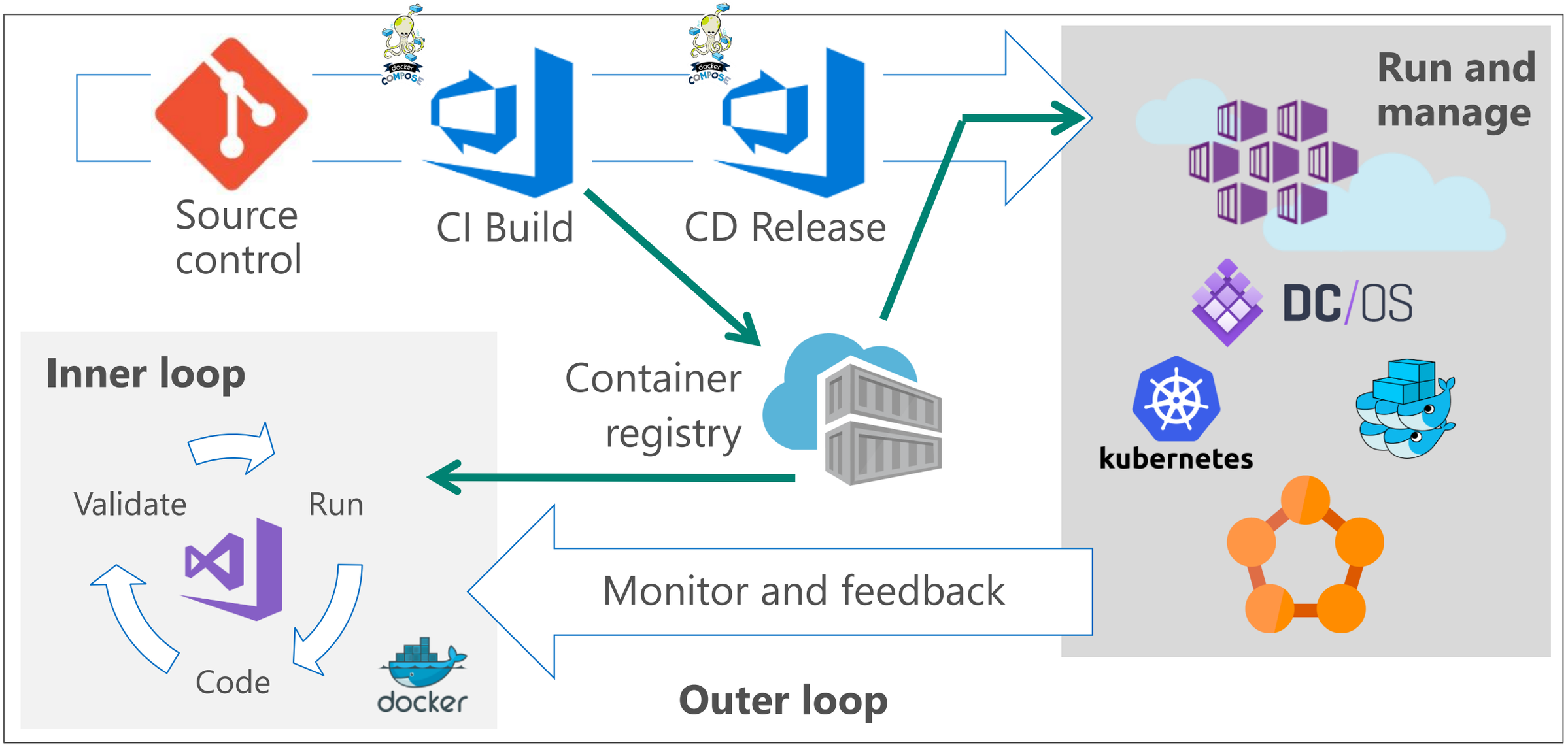
Runs on both Azure and Azure Stack

Alternative in ARM templates



Azure Container Service

Outer loop


Source control
CI Build
CD Release
Run and manage
DC/OS
kubernetes
Container registry
Inner loop
Validate
Run
Code
docker
Monitor and feedback
Outer loop

Container image registries

Push your images to centralized location

Public or private

Official registries available

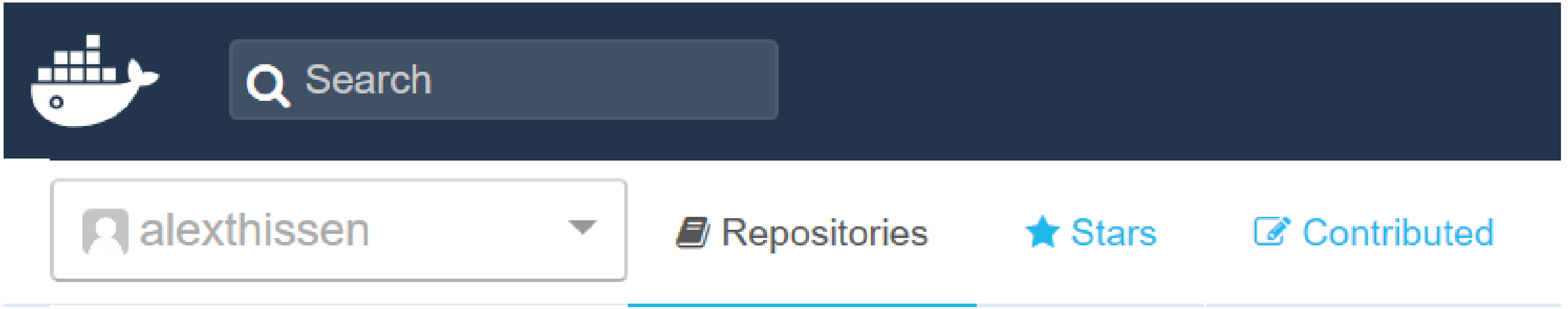


Image name corresponds to repository

Multiple images with different tags can be stored

Choose tag strategy (latest, platform, version number)

Deploy, scale and manage Docker clusters

Differs per cluster orchestrator

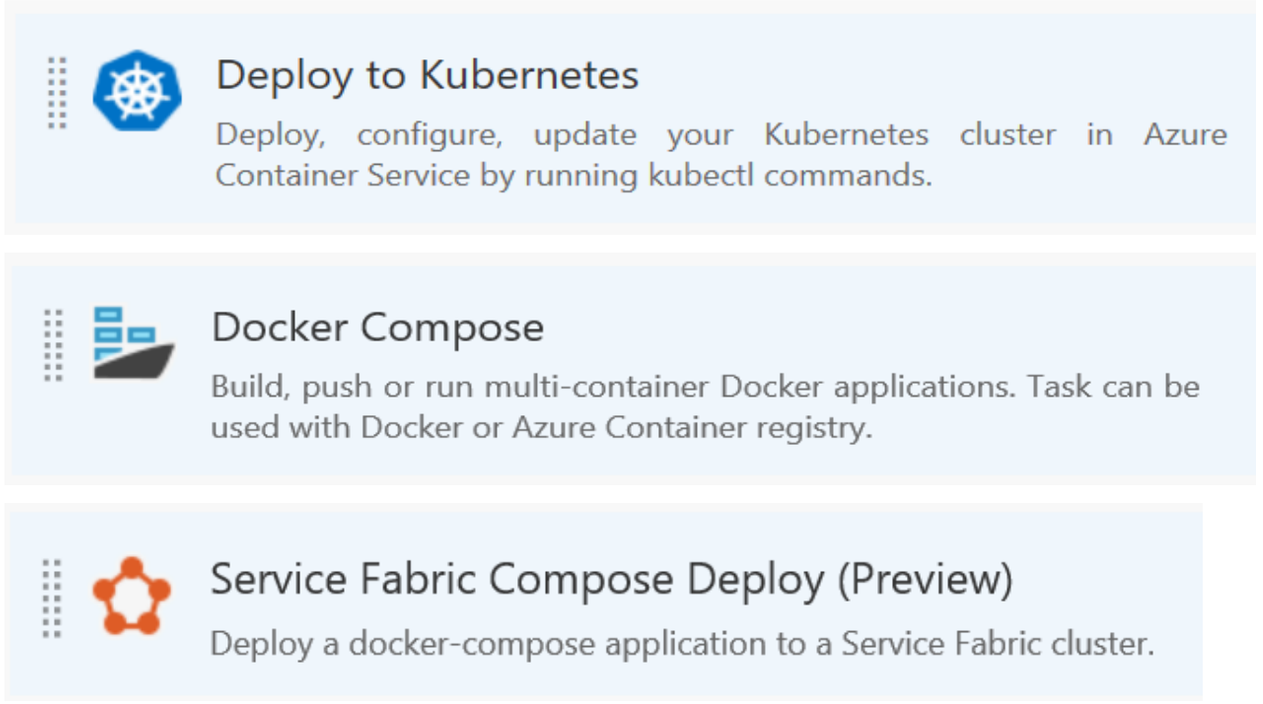
CLI tooling

Kubernetes (k8s): `kubectl`
DC/OS: `dcos`
Docker Swarm Mode: `docker`
Service Fabric: `azure servicefabric`
Azure Container Service: `az acs`

Docker Swarm Mode management commands

docker swarm
docker stack
docker service
docker node

VSTS tasks

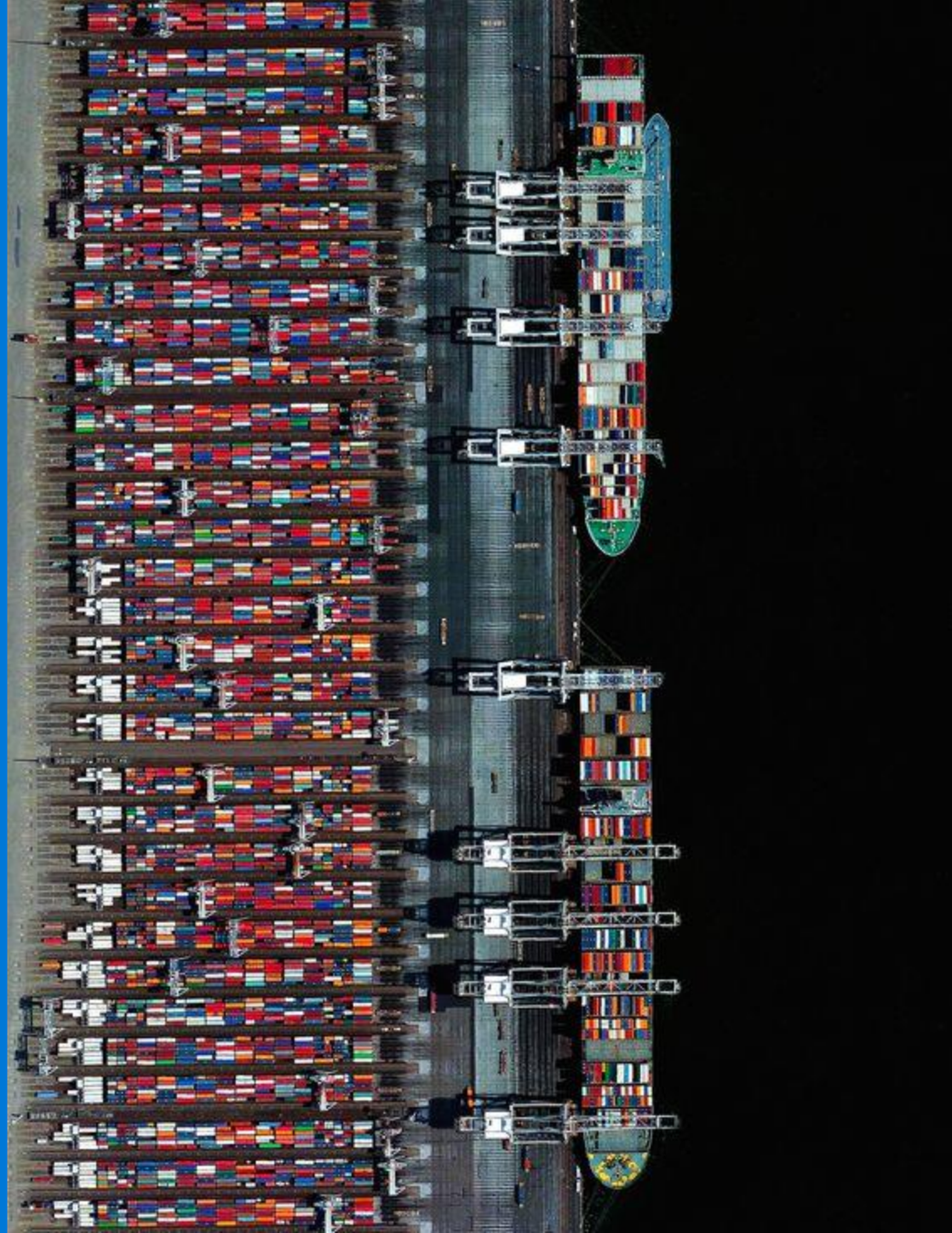


Demo #7

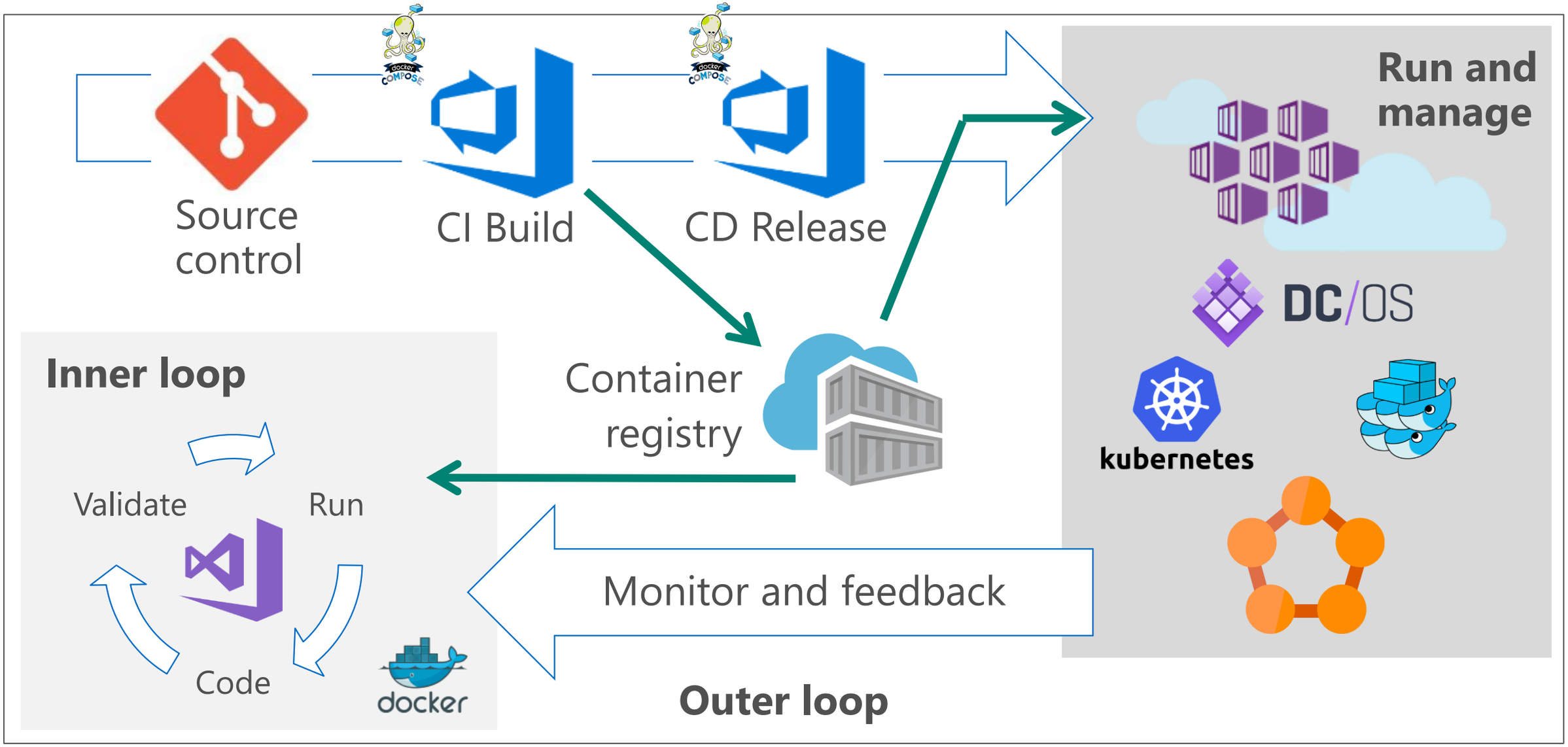
Deploying and hosting in a Docker Swarm cluster

Demo #8

Visual Studio Team Services
Build and release pipelines



Summary


Source control
CI Build
CD Release
Run and manage
DC/OS
kubernetes
Container registry
Inner loop
Validate
Run
Code
docker
Monitor and feedback
Outer loop

Resources

http://dot.net

http://docs.docker.com

http://hub.docker.com/microsoft/...

- aspnet
- dotnet
- iis

http://visualstudio.com

https://docs.microsoft.com/en-us/