



Developing Applications with Containers

Microsoft Services



Agenda

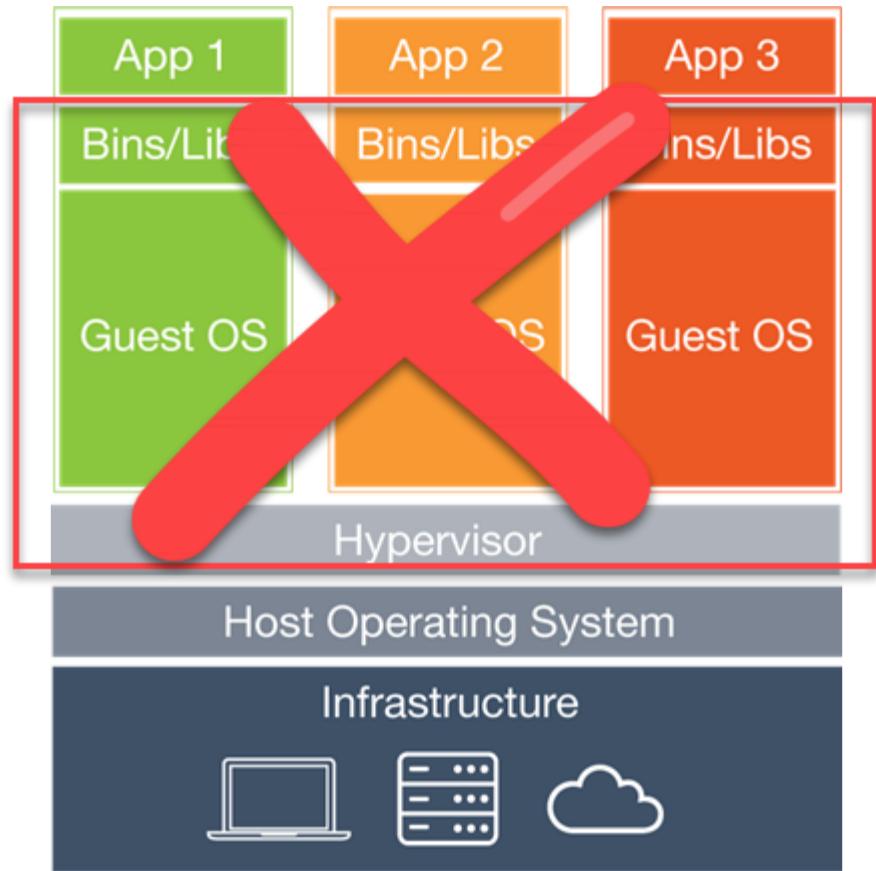
- ✓ Why do we care about containers?
- ✓ How do we run containers?
- ✓ Docker Containers
- ✓ Linux vs. Windows Containers
- ✓ Dockerfiles
- ✓ Updating base OS layer in images
- ✓ Docker-compose
- ✓ Visual Studio Tools & Docker
- ✓ Orchestration
- ✓ Review and Recap



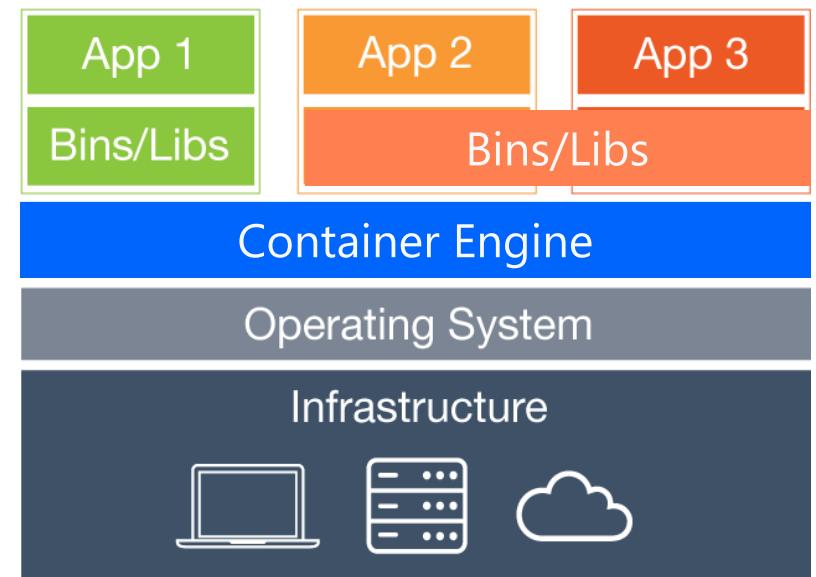
Why do we care about Containers?

Virtual Machine versus Containers

Virtual Machine



Container



Benefits of Containers

- Build it once, built it anywhere.
 - No more “It works on my machine.”
 - No environmental inconsistencies to worry about.
- Isolation and resource sharing
- Resource efficiency
- Speed: start, stop, create, and scale containers in seconds
- Operational simplicity (host updates, no licensing headaches)
- Effective DevOps pipelines
- Goes well with microservices architecture

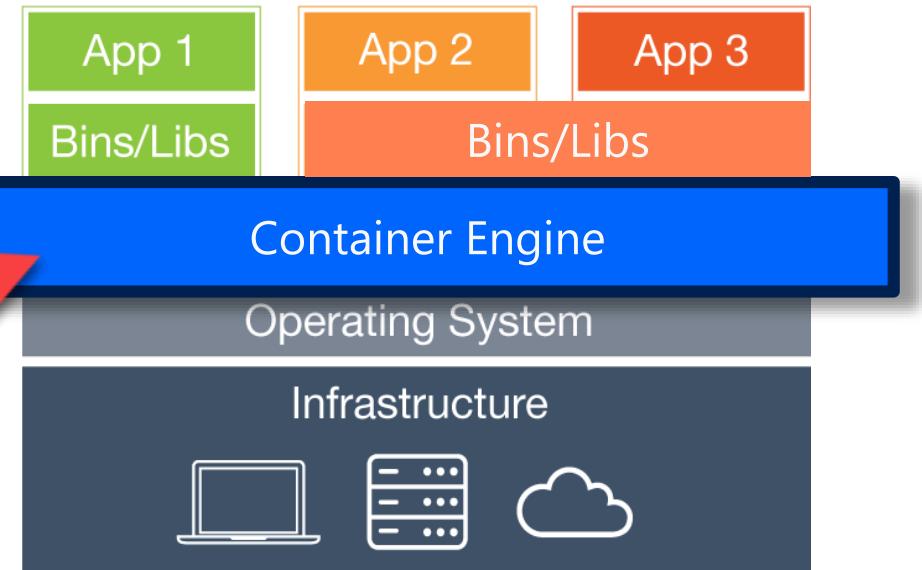
How do we run Containers?

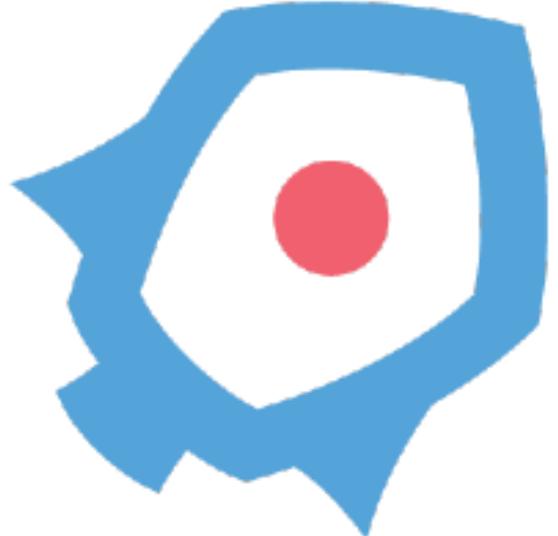
Container Runtime/Engine

- Software that executes and manages containers
- Many choices, Docker is most popular, followed by rkt

*Docker, containerd, rkt, lxd,
containerd, turbo, Clear,
runc, etc...*

Container





- Pod native approach for direct integration with Kubernetes orchestrator
- Security-focused
- Linux focused
- rkt implements a modern, open, standard container format, the App Container (appc) spec, but can also execute other container images, like those created with Docker

CoreOS
rkt (rocket)



- Container native approach, can be run inside pods for Kubernetes or as-is with Swarm orchestrator
- Build any app in any language using any stack (OS), Cross-OS platform support
- Integration with Microsoft products like Visual Studio. Microsoft direct support for Docker hub images.

Docker

Docker Containers

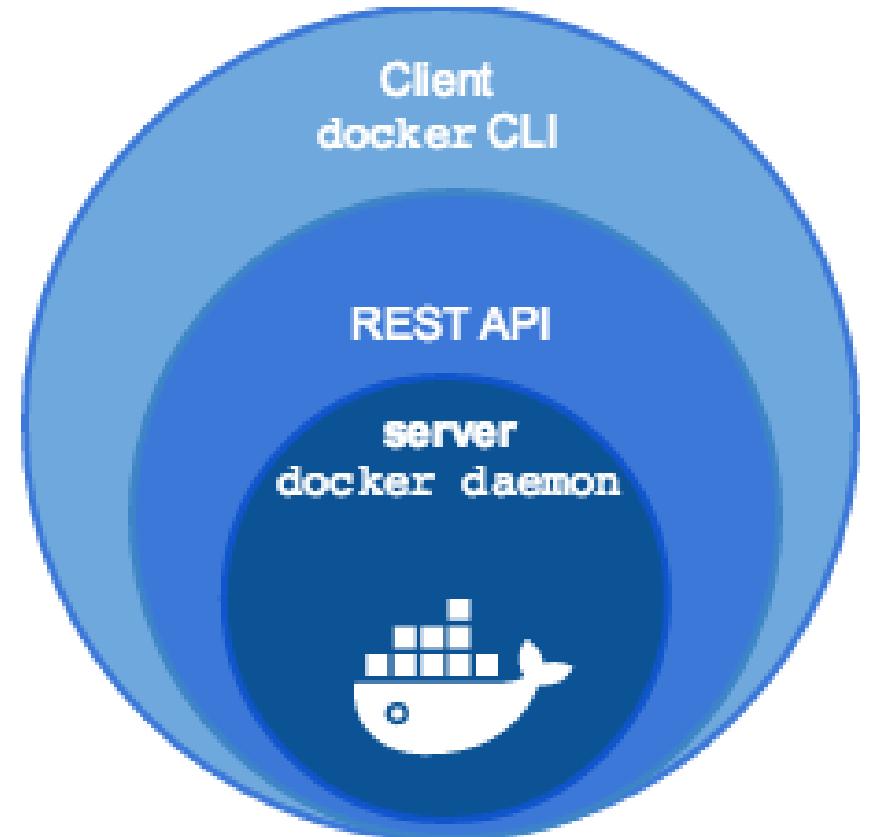
Docker Vocabulary

Host	A VM or on premise server running the Docker Daemon to host a collection of Docker Containers
Image	An <i>ordered collection of filesystems (layers)</i> to be used when instancing a container (more on it later)
Container	A runtime instance of an image
Registry	A collection of docker images

If an image is a class, then a container is an instance of a class—a runtime object.

Docker Engine

- Docker CLI
 - Allows you to issue Docker commands to create / manage containers.
- Docker API
 - Interface for interacting with the daemon
- Docker Daemon
 - The program that enables containers to be built, shipped, and run.
 - Uses Linux Kernel namespaces and control groups to give an isolated runtime environment for each application



Quick Question?

How fast you can launch a fully functional WordPress blog engine?

How about multiple WordPress blog engines running side by side on same host?

```
Razis-MacBook-Pro:~ Razi$ docker run -d -p 80:80 tutum/wordpress
Unable to find image 'tutum/wordpress:latest' locally
Latest: Pulling from tutum/wordpress

387d9ff0016: Extracting [=====] 63.5 MB/65.68 MB
3b52deaaf0ed: Download complete
4bd501fad6de: Download complete
a3ed95caeb02: Download complete
790f0e8363b9: Downloading [=====] 80.01 MB/83.82 MB
11f87572ad81: Download complete
341e06373981: Download complete
709079cecfb8: Download complete
55bf9bbb788a: Download complete
541f3cf3d47: Download complete
70789ae370c5: Download complete
1018956370c2: Download complete
1013c134941a: Download complete
100018891889: Download complete
100024011029: Download complete
```

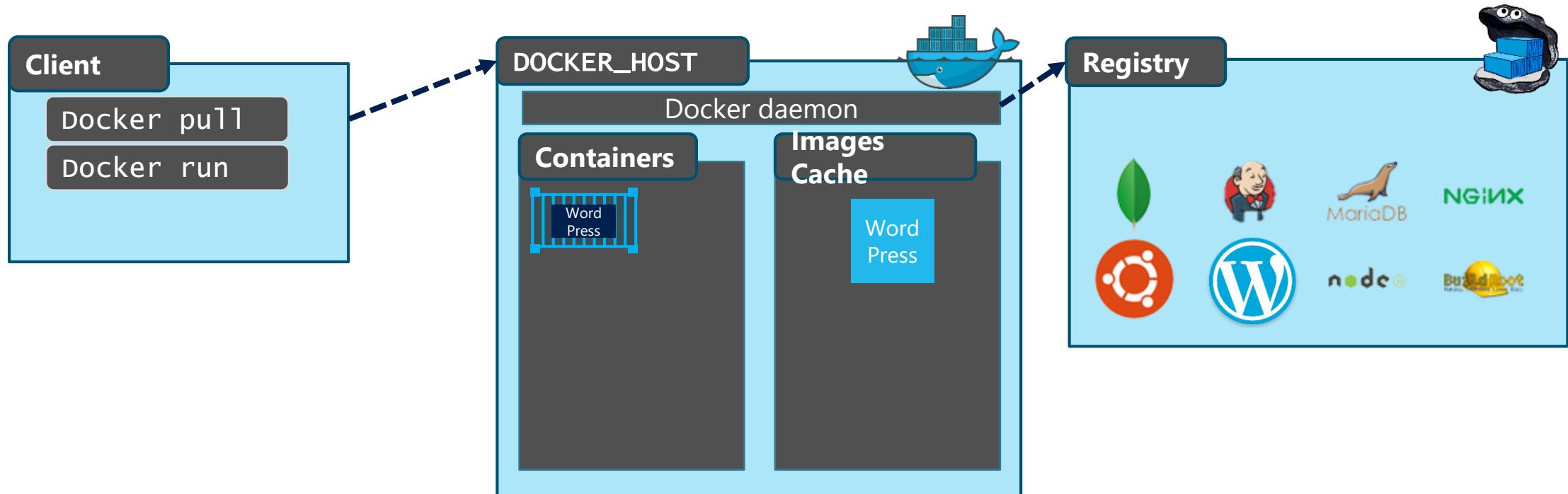
Demonstration: Running Docker Containers

Launch a single WordPress Container

Running multiple WordPress Containers side by side



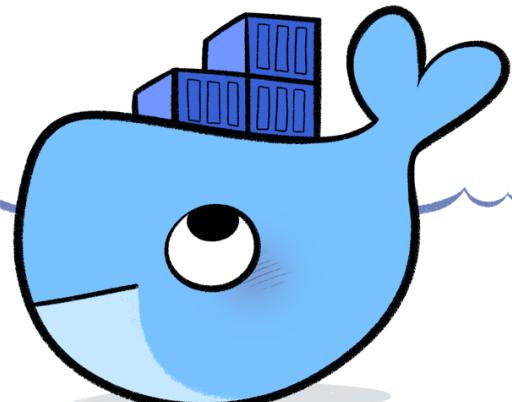
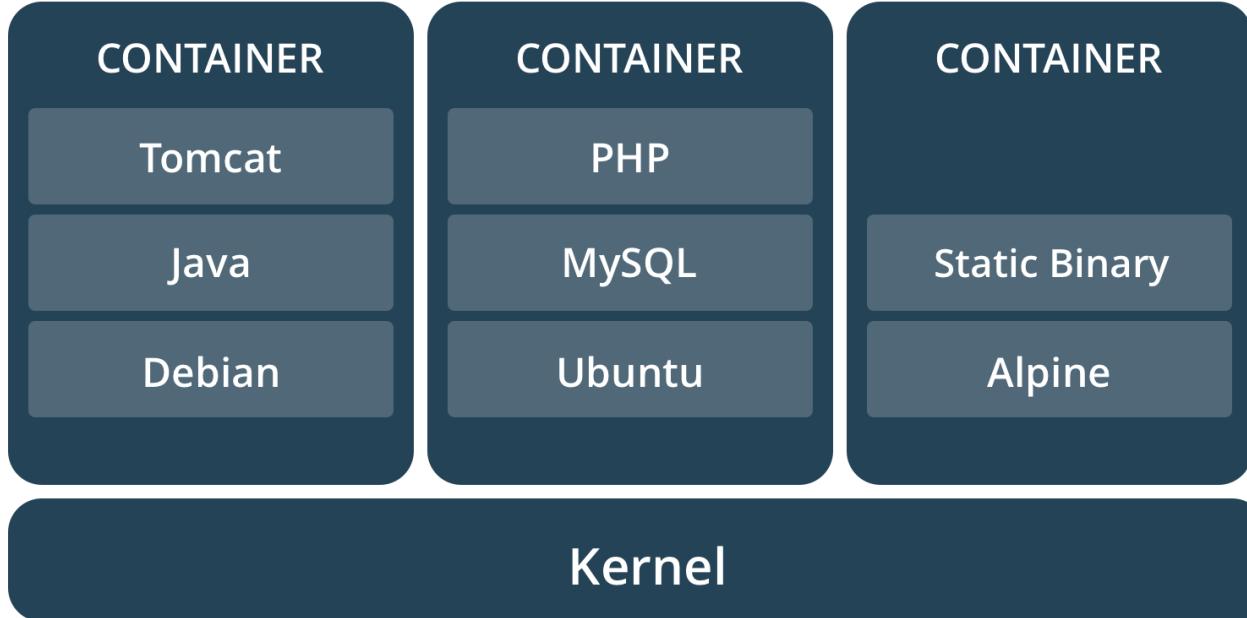
Docker In Action



Linux vs. Windows Containers

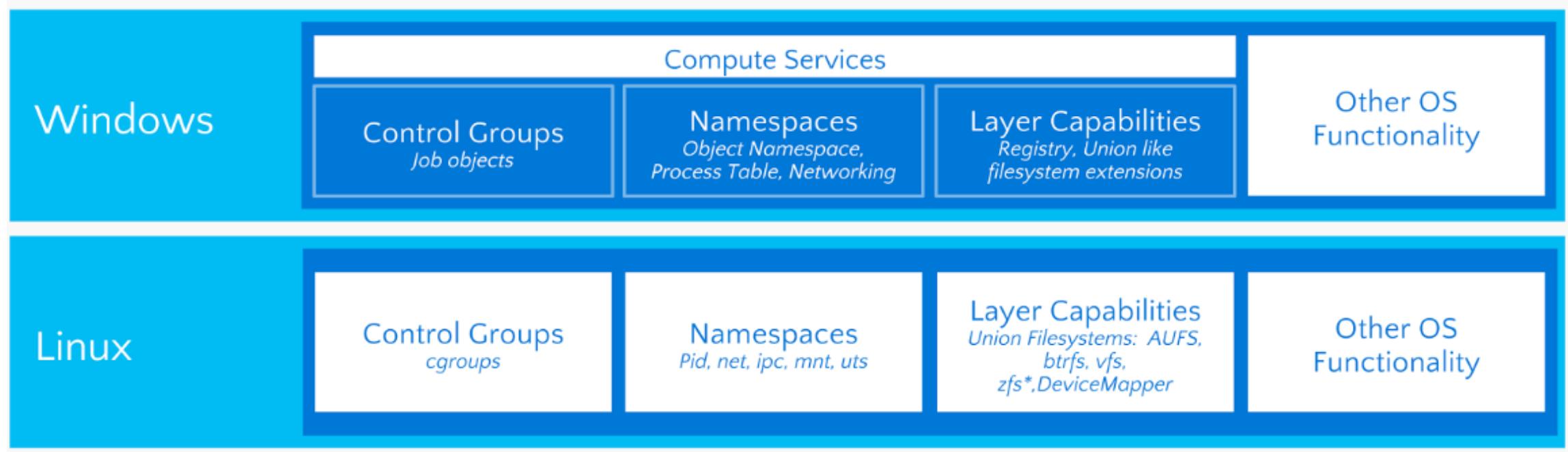
Linux vs. Windows Containers

- You can only run Windows containers on a Windows host and Linux containers on a Linux host.
- Linux
 - Containers were originally built for Linux and support tends to be more stable and better on Linux, though Windows is catching up.
- Windows
 - Docker supports only certain versions of Windows: Windows Server 2016 and Windows 10



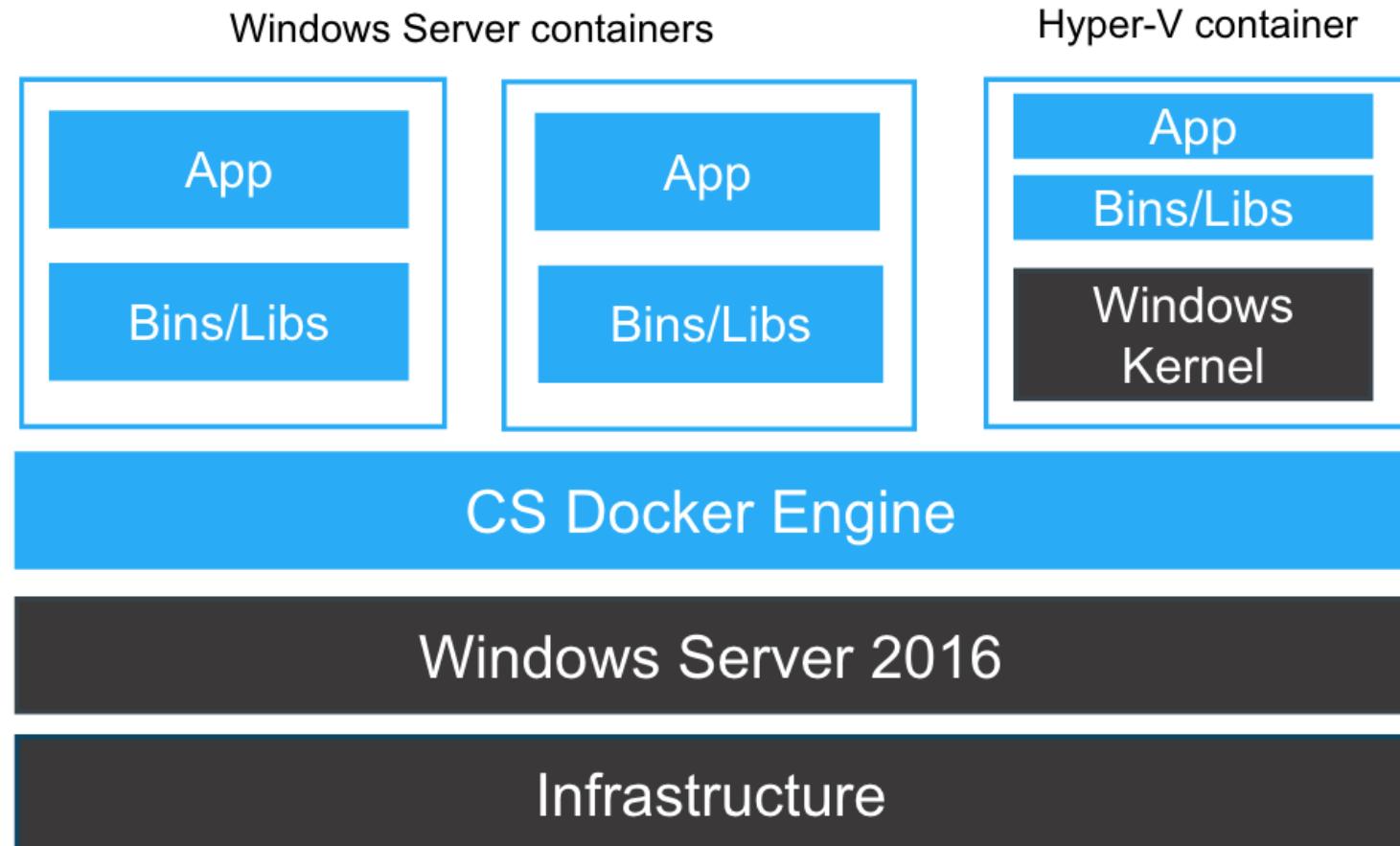
Containers share the kernel with the host OS

Linux vs. Windows Containers



Windows Hyper-V Containers

Hyper-V Containers offer both OS virtualization (container) and machine virtualization (VM) in a slightly lighter-weight configuration than a traditional VM.



Demonstration: Nano Server

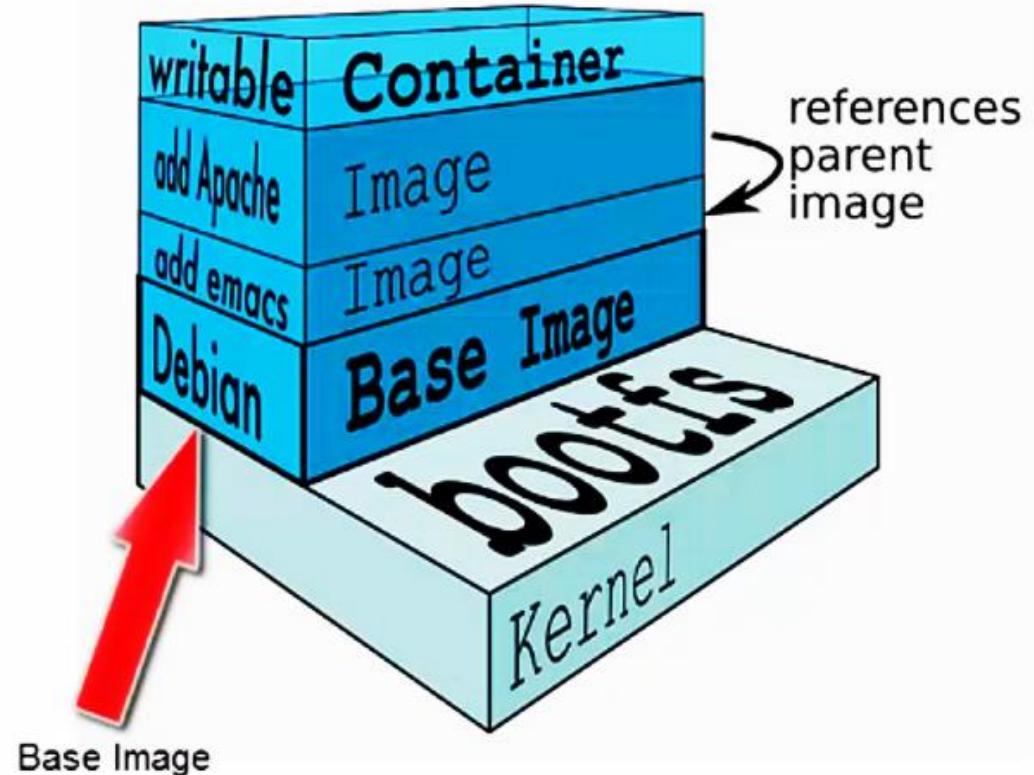
Working with Nano Server Container



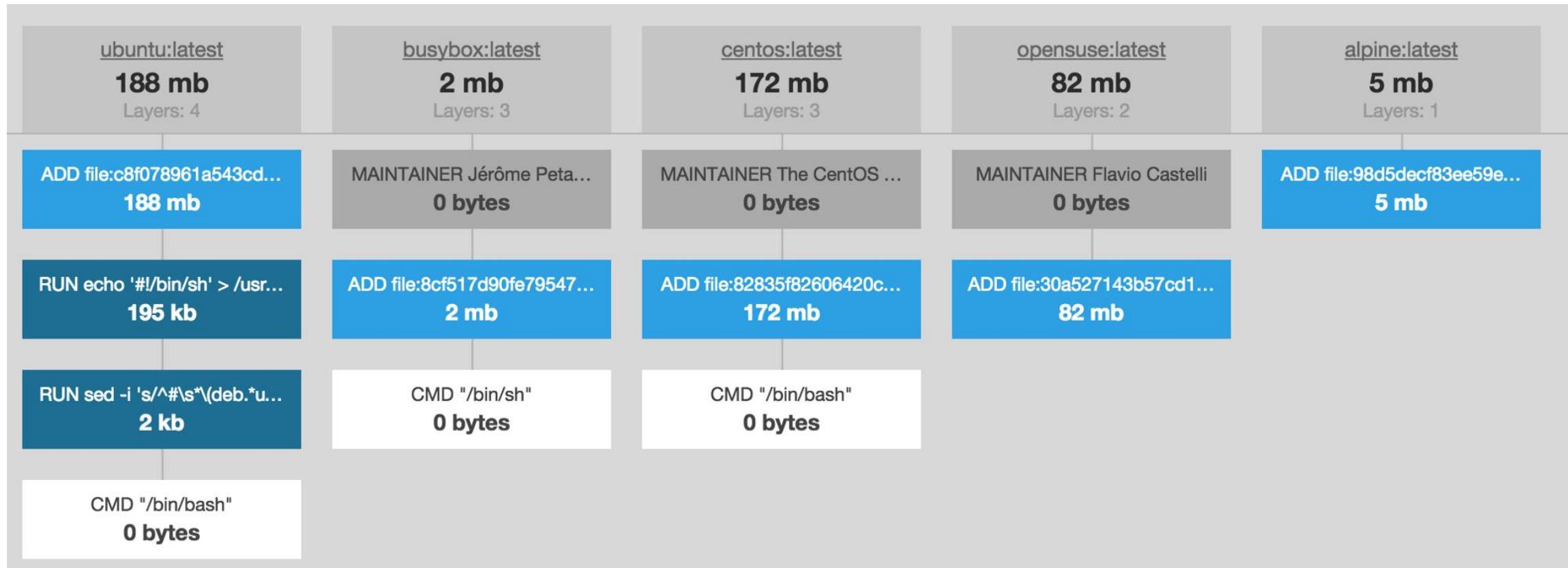
Docker Images: What are they and where to get them?

Docker Images

- A Docker image is built up from a series of layers.
- Base platform OS image is provided by vendors like Microsoft for Windows OS image, Canonical for Ubuntu image etc. These images get published to DockerHub.
- Each layer represents an instruction in the image's Dockerfile.
- Each layer except the last one is read-only.



Consider image size



Demonstration: Docker Image Layers

List All Layers for Docker Image



Manifest List – “fat manifest”

- Points to specific image manifests for one or more platforms.
- Optional.
- A client will distinguish a manifest list from an image manifest based on the Content-Type returned in the HTTP response.

```
{  
  "schemaVersion": 2,  
  "mediaType": "application/vnd.docker.distribution.manifest.list.v2+json",  
  "manifests": [  
    {  
      "mediaType": "application/vnd.docker.image.manifest.v2+json",  
      "size": 7143,  
      "digest": "sha256:e692418e4cbaf90ca69d05a66403747baa33ee08806650b51fab815ad7fc331f",  
      "platform": {  
        "architecture": "ppc64le",  
        "os": "linux",  
      },  
      {  
        "mediaType": "application/vnd.docker.image.manifest.v2+json",  
        "size": 7682,  
        "digest": "sha256:5b0bcabd1ed22e9fb1310cf6c2dec7cdef19f0ad69efa1f392e94a4333501270",  
        "platform": {  
          "architecture": "amd64",  
          "os": "linux",  
          "features": [  
            "sse4"  
          ]  
        }  
      }  
    ]  
  }  
}
```

Manifest List – “fat manifest”

ubuntu®

docker pull
microsoft/aspnetcore



docker pull
microsoft/aspnetcore



Fat Manifest

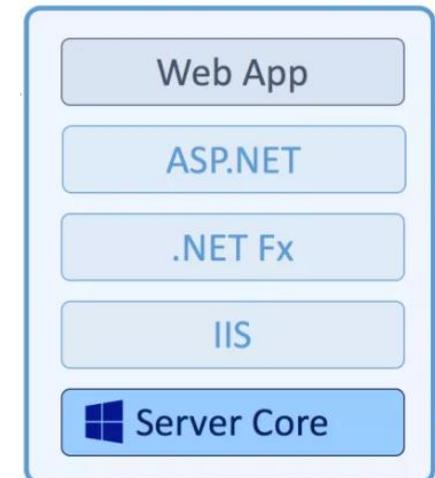
```
"manifests": [  
  {  
    "mediaType": "application/vnd.docker.  
    "size": 7143,  
    "digest": "sha256:e692418e4cbaf90ca  
    "platform": {  
      "architecture": "ppc64le",  
      "os": "linux",  
    }  
  }]
```

Compatible image

91e54dfb1179	0 B
D74508fb6632	1.895 KB
C22013c84729	194.5 KB
D3a1f33e8a5a	1881 MB



```
{  
  "mediaType":  
  "application/vnd.docker.  
  json",  
  "size": 7682,  
  "digest":  
  "sha256:5b0bcabd1ed22e9f  
  19f0ad69efaf1f392e94a4333  
  "platform": {  
    "os": "windows"  
  }  
}
```



Docker Registries

Registry - Stores docker images

- Azure Container Registry (ACR)
 - The program that enables containers to be built, shipped, and run.
 - Uses Linux Kernel namespaces and control groups to give an isolated runtime environment for each application
- Docker Hub
 - A online registry of Docker images
- Docker Trusted Registry
 - Private on-site Registry for Docker images



The Registry is open-source under the permissive Apache License.

Demonstration: Docker Registry

Search Docker Registry using Docker CLI

Search Images on DockerHub

Docker Image Naming Convention



Building Docker images with Dockerfiles

Dockerfile

- Text file with Docker commands in it to create a new image. You can think of it as a configuration file with set of instructions needed to assemble a new image.
- Docker has a docker build command that parses Dockerfile to build a new container image.

```
# Simple Dockerfile for NGINX

FROM nginx:stable-alpine

MAINTAINER Razi Rais

COPY index.html /usr/share/nginx/html/index.html

CMD ["nginx", "-g", "daemon off;"]
CWD ["/etc/nginx/conf.d"]
```

```
FROM microsoft/dotnet:1.1.0-sdk-projectjson

COPY . /app

WORKDIR /app

RUN ["dotnet", "restore"]

RUN ["dotnet", "build"]

EXPOSE 5000/tcp

CMD ["dotnet", "run", "--server.urls", "http://*:5000"]
CWD ["/src"]

EXPOSE 2000/tcp

EXPOSE 8080
```

```
# Simple Dockerfile for NodeJS

FROM node:boron

MAINTAINER Razi Rais

# Create app directory
RUN mkdir -p /usr/src/app
WORKDIR /usr/src/app

# Install app dependencies
COPY package.json /usr/src/app/
RUN npm install

# Bundle app source
COPY . /usr/src/app

EXPOSE 8080

CMD [ "npm", "start" ]
CWD [ "/app" ]

EXPOSE 8080
```

Common Dockerfile Instructions

- **FROM** instruction initializes a new build stage and sets the Base Image for subsequent instructions.
- **LABEL** is a key-value pair, stored as a string. You can specify multiple labels for an object, but each key-value pair must be unique within an object.
- **RUN** will execute any commands in a new layer on top of the current image and commit the results.
- **WORKDIR** instruction sets the working directory for any RUN, CMD, ENTRYPOINT, COPY and ADD instructions that follow it.
- **ADD** instruction copies new files, directories or remote file URLs from **<src>** and adds them to the filesystem of the image at the path **<dest>**.
- **COPY** instruction copies new files or directories from **<src>** and adds them to the filesystem of the container at the path **<dest>**.
- **CMD** provide defaults for an executing container. These defaults can include an executable.
- **ENTRYPOINT** allows you to configure a container that will run as an executable.
- **EXPOSE** instruction informs Docker that the container listens on the specified network port(s).

Image Tags

- String value that you can use to distinguish versions of your Docker images.
- PublisherName/ImageName:Tag

[microsoft/windowsservercore](#)

Last pushed: 18 days ago

Repo Info	Tags
Tag Name	
	latest
	10.0.14393.1066
	10.0.14393.1066_zh-tw
	10.0.14393.1066_zh-cn

[microsoft/dotnet](#) 

Last pushed: 4 days ago

[Repo Info](#) [Tags](#) [Dockerfile](#) [Build Details](#)

Tag Name
nanoserver-10.0.14393.1066
nanoserver
sdk-nanoserver-10.0.14393.1066
sdk-nanoserver
1-sdk-nanoserver-10.0.14393.1066
1-sdk-nanoserver

Demonstration: Dockerfile and Docker Build

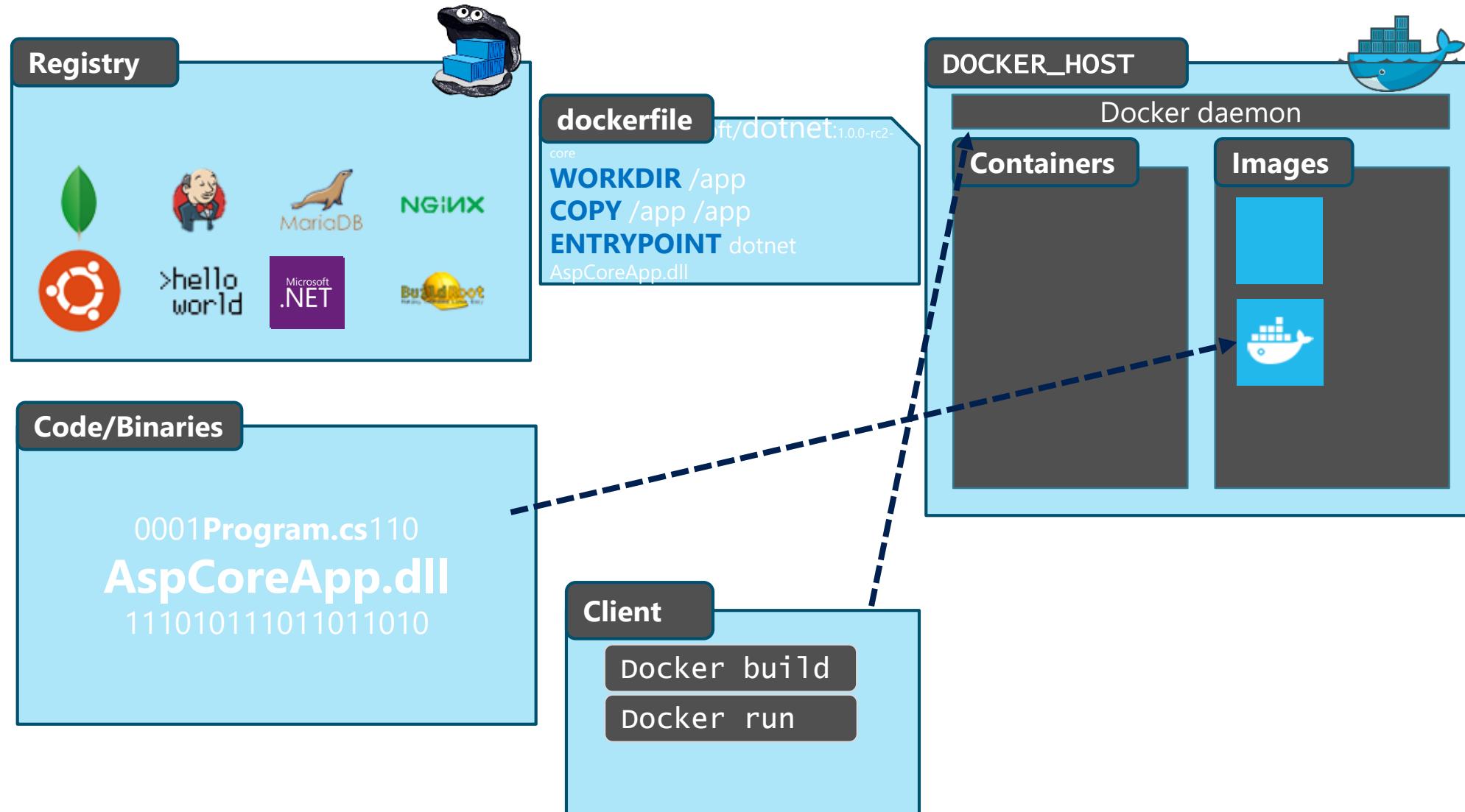
Build a Dockerfile

Build container images using Docker
build command:

- NodeJs



How does Docker build work?



Updating OS in Container Image

Update Container OS Image

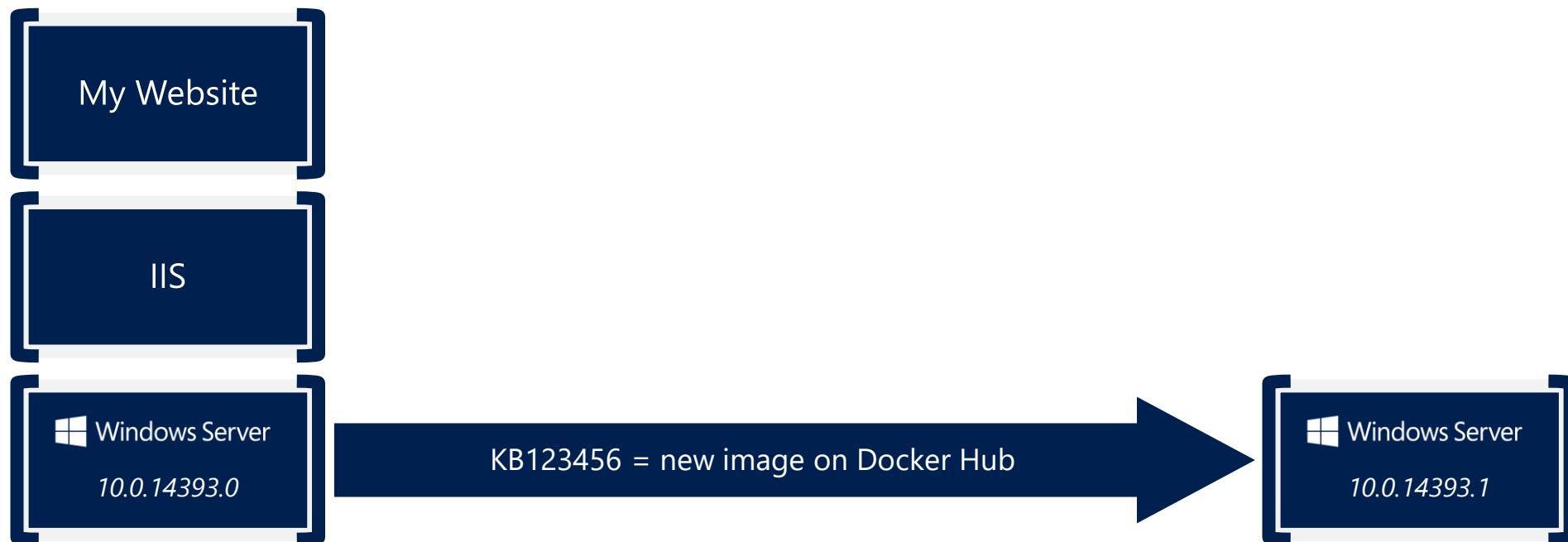
- Pull updated base image
- Rebuild containers using Dockerfile



Update Container OS Image

Pull updated base image

<https://hub.docker.com/r/microsoft/nanoserver/tags>



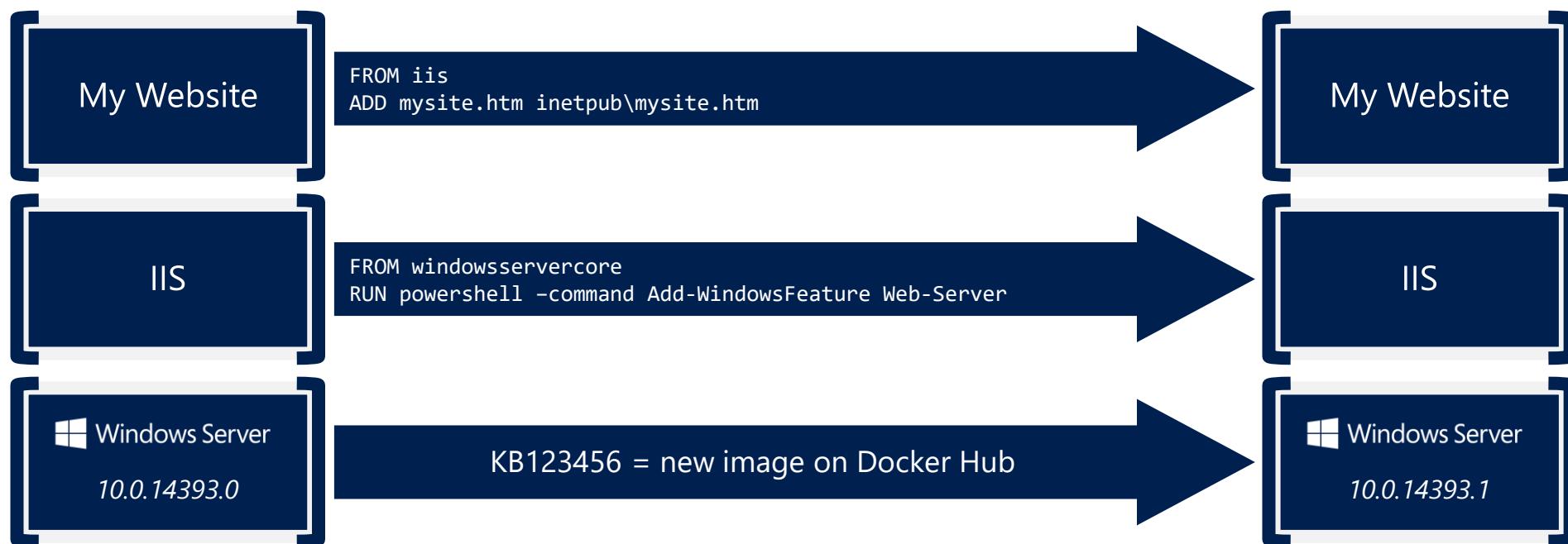
Update Container OS Image

Rebuild containers using Dockerfile



Update Container OS Image

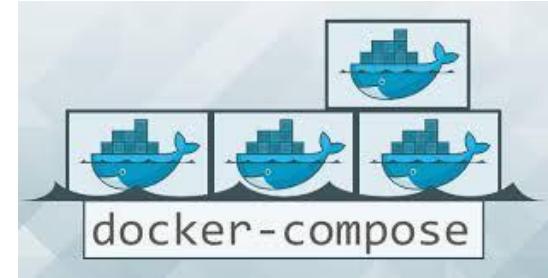
Rebuild containers using Dockerfile



Docker Compose

Docker Compose

- Compose is a tool for defining and running multi-container Docker applications.
 - Single compose file defined in yml format defines your application services
 - Single command to create and start all the services
 - Single command to stop all the services
 - Services Discoverability

A screenshot of a code editor window titled "docker-compose.yml". The file contains the following YAML configuration:

```
version : '2'
services:
  demowebapp:
    image: rzdockerregistry.azurecr.io/demo-webapp
    ports:
      - 80:80
    depends_on:
      - demowebapi
  demowebapi:
    image: rzdockerregistry.azurecr.io/demo-webapi
    ports:
      - 9000:9000
      - 8000:8000
      - 9001:9001
```

The code editor has a dark theme with icons on the left side. A status bar at the bottom right shows "100% 1000ms 1000ms 1000ms".

Docker Compose

- Using compose is a three step process:
 - Define your app's environment with a Dockerfile so it can be reproduced anywhere.
 - Define the services that make up your app in docker-compose.yml so they can be run together in an isolated environment.
 - Lastly, run docker-compose up and Compose will start and run your entire app.

```
FROM microsoft/dotnet:nanoserver
WORKDIR /app

COPY published ./

ENV ASPNETCORE_URLS http://+:80
EXPOSE 80
ENTRYPOINT ["dotnet", "mywebapp.dll"]
```

Dockerfile | webapp

```
FROM microsoft/dotnet:nanoserver
WORKDIR /app

COPY published ./

ENV ASPNETCORE_URLS http://+:9000
EXPOSE 9000

ENTRYPOINT ["dotnet", "mywebapi.dll"]
```

Dockerfile | webapi

```
version : '2'

services:
  demowebapp:
    build: ./mywebapp
    ports:
      - 80:80
    depends_on:
      - demowebapi
  demowebapi:
    build: ./mywebapi
    ports:
      - 9000:9000
networks:
  default:
    external:
      name: nat
```

Docker-compose.yml

Demonstration: Docker Compose

Launch Multi-Container Application
using Docker Compose

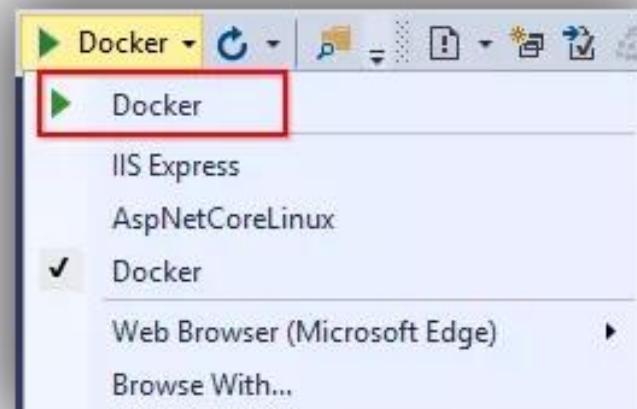
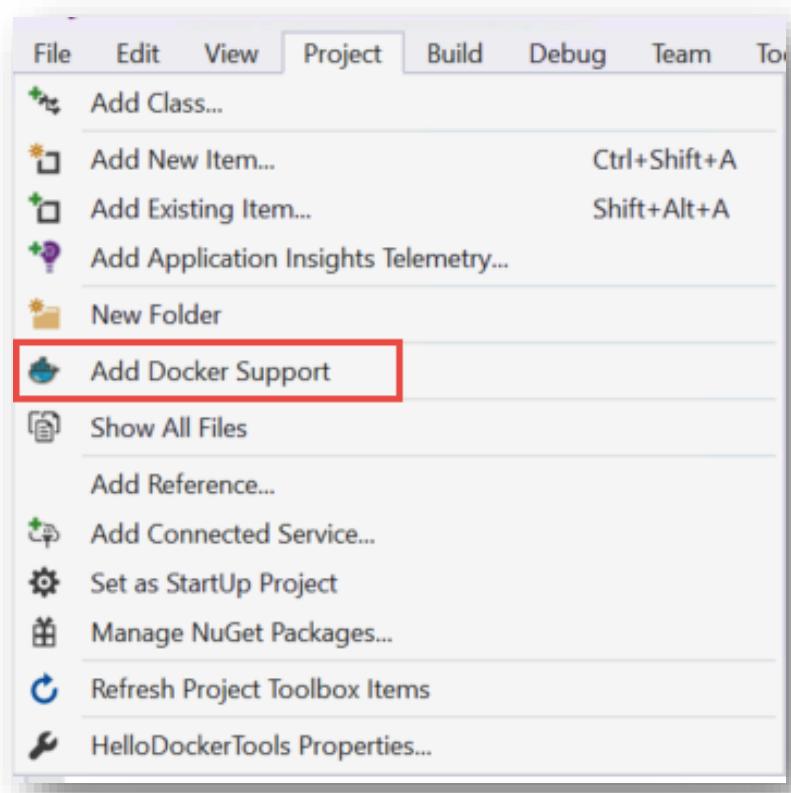


Visual Studio Tools & Docker

Visual Studio Tools for Docker

Microsoft Visual Studio 2017 provides integrated developer experience with Docker.

Building, Debugging, and Running .NET Framework and .NET Core web and console applications using Windows and Linux containers.



Demonstration: Visual Studio 2017 and Docker

Building ASP.NET Application using Visual Studio

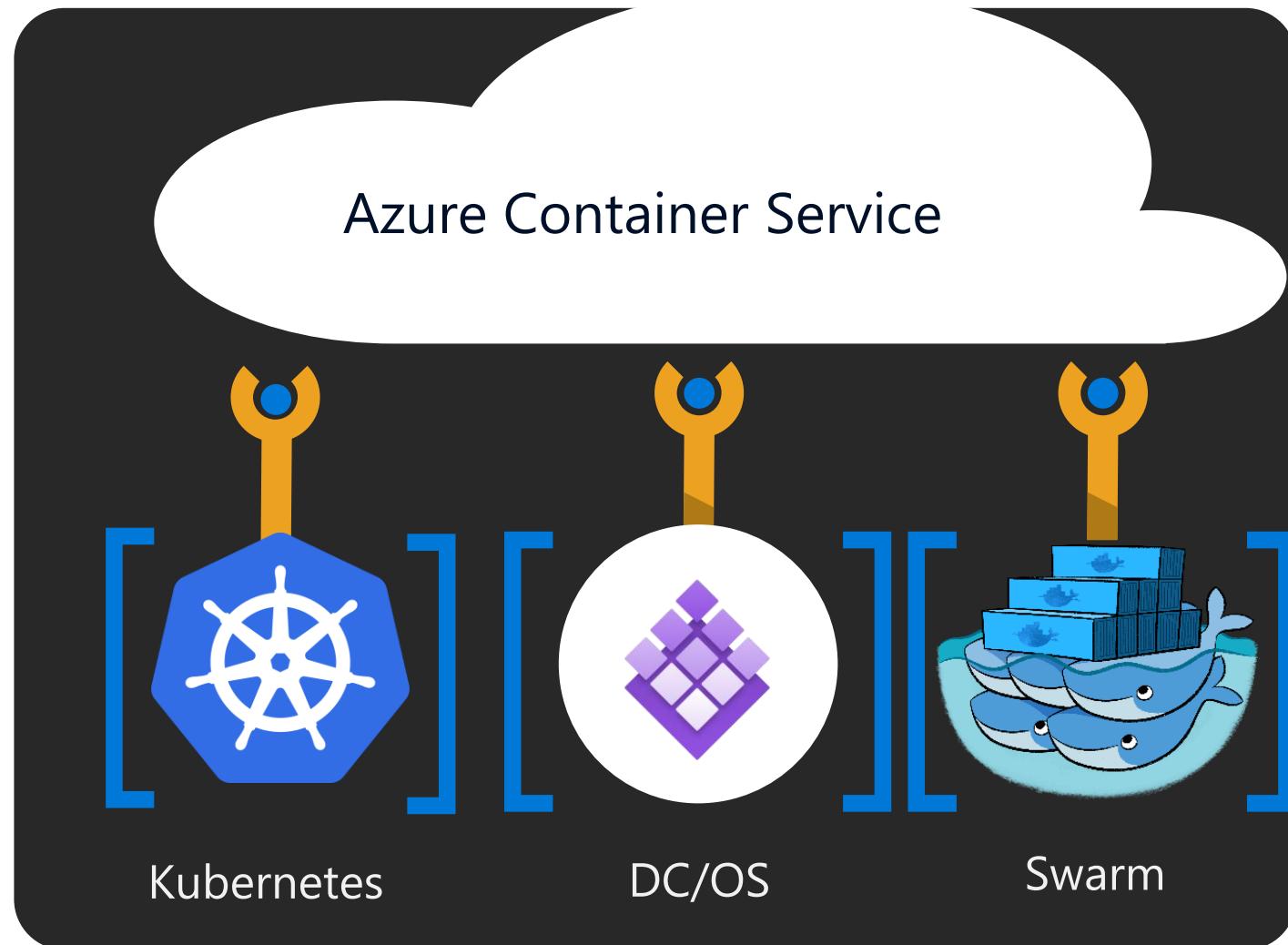
Debugging ASP.NET Application using Visual Studio



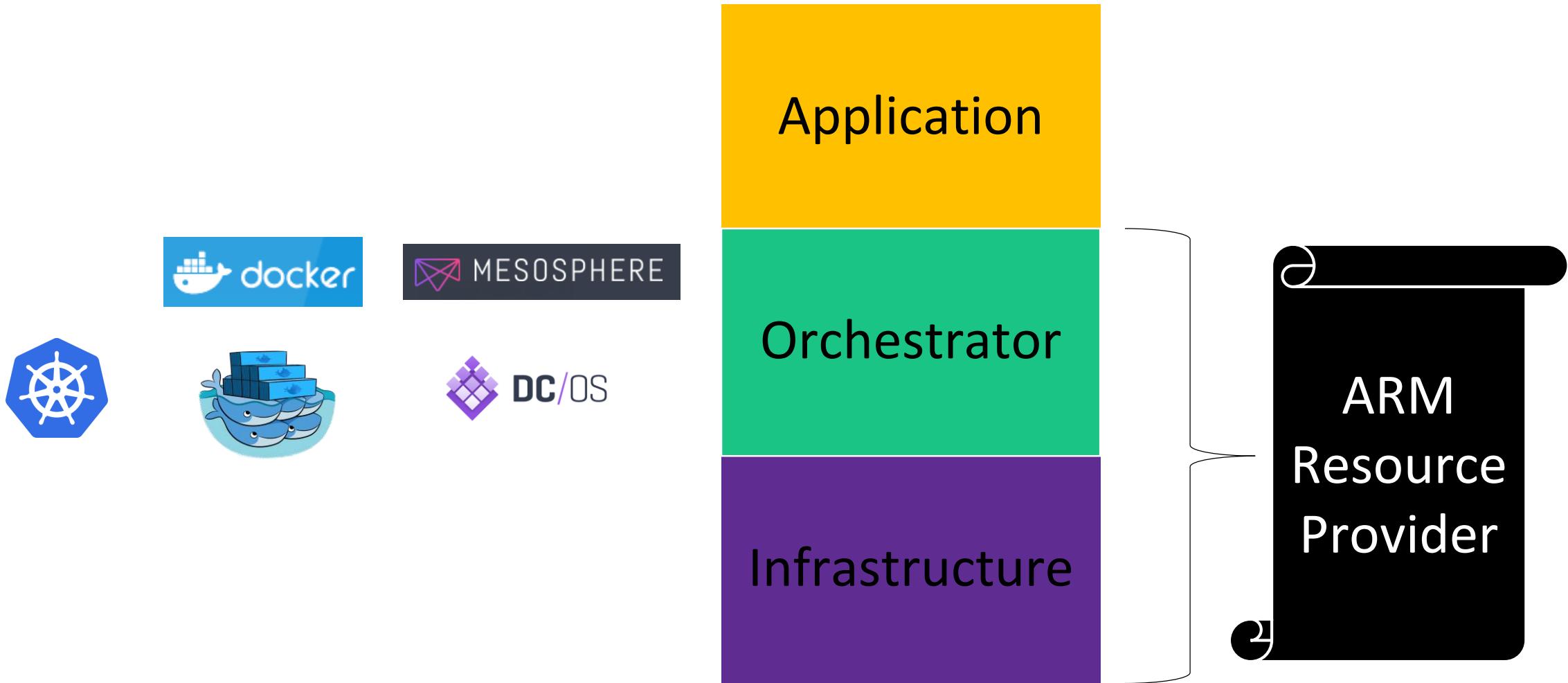
Orchestration

Azure Container Service (ACS)

- Create an optimized container hosting solution
- Standard tooling and API support
- Streamline provisioning of DC/OS, Docker Swarm, and Kubernetes (in preview)
- ACS engine is open source:
<https://github.com/Azure/acs-engine>
- Leverage Azure Platform capabilities
 - Azure Resource Manager
 - VM Scale Sets
 - Networking
 - Security



Azure Container Service (ACS)



Choosing Orchestrators in Azure

Azure Product	Orchestrator	Description	Good for	Common workloads
Azure Service Fabric 	Service Fabric 	Azure Service Fabric is a distributed systems platform that makes it easy to package, deploy, and manage scalable and reliable microservices	Production-ready & Microsoft ecosystem More mature:  Less mature: 	a) Stateful svcs & Actors b) Microservices based on plain processes c) Microservices based on containers
Azure Container Service 	Docker Swarm 	Docker Swarm is a clustering and scheduling tool for Docker containers. With Swarm, IT administrators and developers can establish and manage a cluster of Docker nodes as a single virtual system	Easy to get started Dev/Test and Production More mature:  Less mature: 	Microservices based on containers
	Kubernetes 	Kubernetes is an open-source platform for automating deployment, scaling, and operations of application containers across clusters of hosts	Production-ready & Linux ecosystem More mature:  Less mature: 	Microservices based on containers
	Mesosphere DC/OS 	As a datacenter operating system, DC/OS is itself a distributed system, a cluster manager and a container platform	Production-ready & Linux ecosystem More mature:  Less mature: 	Microservices based on containers (Including other Linux containers, not just Docker)

Demonstration: *Kubernetes Cluster in Azure Container Service*

Deploy a Kubernetes cluster in Azure Container Service

Deploy NGINX container into Kubernetes cluster

Kubernetes UI



Review and Recap

tl;dr Summary

- Containers provide better performance and a streamlined process for DevOps pipelines. No more “it works on my machine”.
- Docker is the most common container engine and supported by Microsoft.
- An image contains the base OS, application, and all bins/libs for the app. Images are made of layers.
- If an image is a class, then a container is an instance of a class—a runtime object.
- Container images can be saved in a container registry.
- Dockerfiles are an easy way to make container images.

tl;dr Summary

- Orchestrators will manage and monitor all your updates, start/stopping containers, scaling, and container lifecycle. It also handles networking, load balancing, and much more.
- Microservices go very well with containers and orchestration.
- Kubernetes is most commonly used orchestrator.
- AKS is currently in PREVIEW (6/2018) and supports only Linux containers
- ACS is available for GA (general availability) and supports both Windows and Linux containers

When to use containers

- Microservices applications that require fine-grained scaling
- Goes extremely well with ASP.NET Core because it is cross platform
- Applications requiring high performance (high user load) and uptime
- Legacy lift and shift applications
- Dev/QA databases for easy test data that can be provisioned quickly

When not to use containers

- Applications from legacy systems requiring something older than Win 10 and Server 2016
- If your application requires direct access to an IoT device that can only be reached through a native platform
- If your app requires a significant re-write to work well in containers then the cost of moving containers might not be worthwhile.
- If your apps are tightly coupled to their data, or if data management is a key focus of your application.
- Production Databases

Questions?

Thank you!