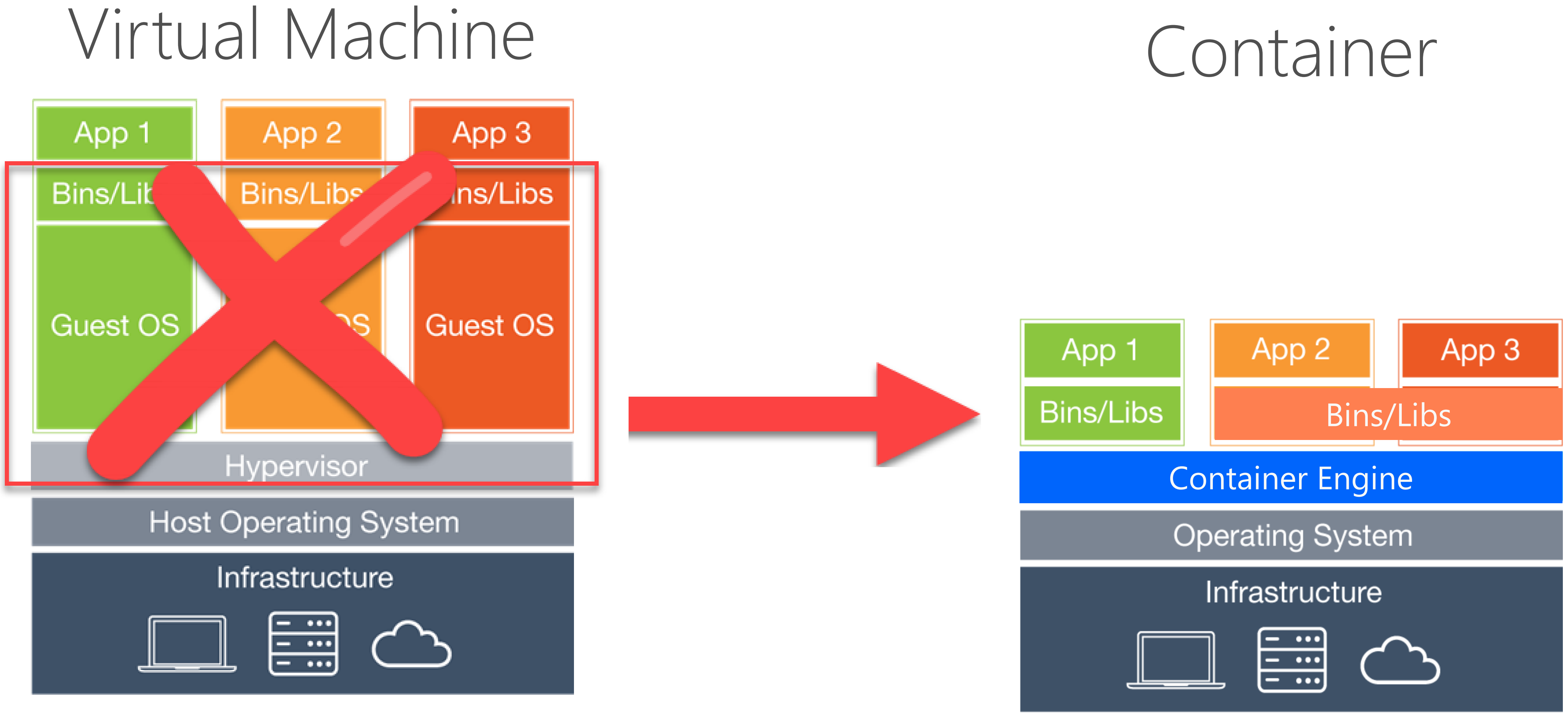
Microsoft

Developing Applications with Containers

Microsoft Services

Why do we care about Containers?

Virtual Machine versus Containers



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

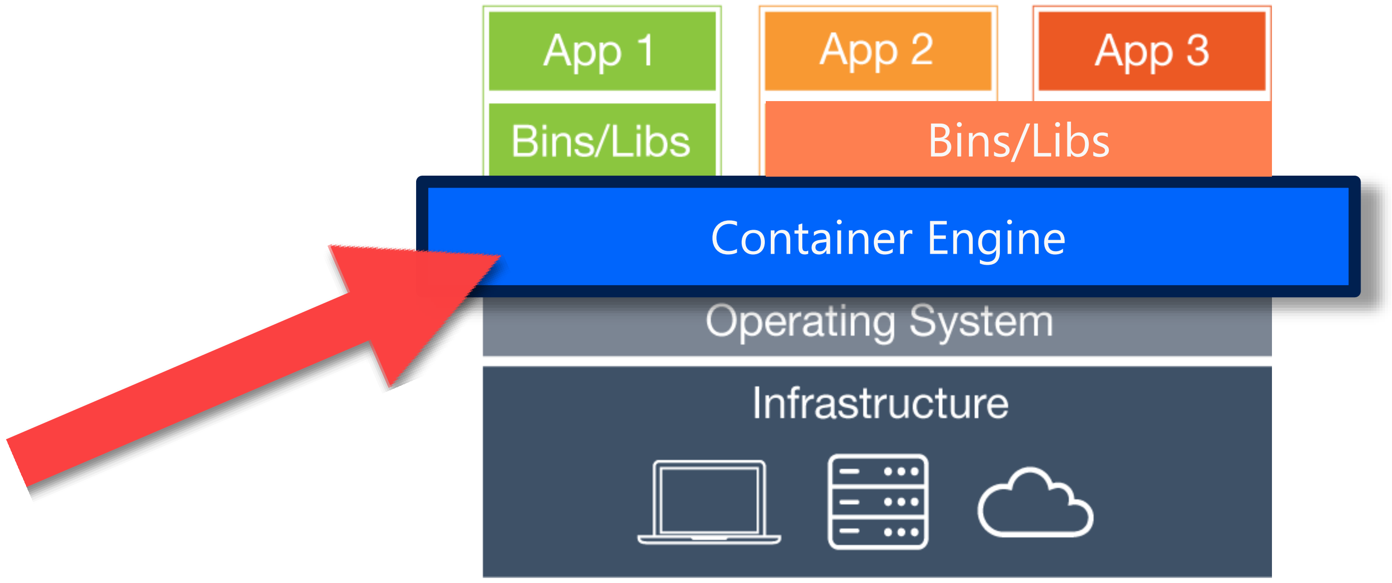
How do we run Containers?

Container Runtime/Engine

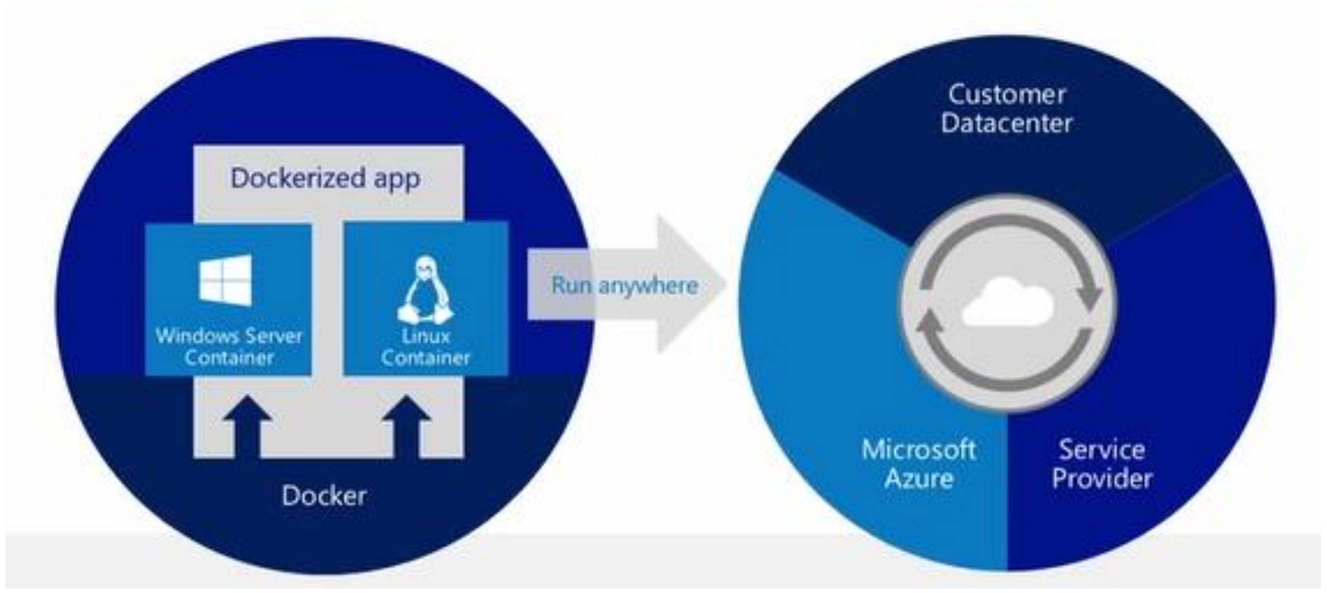
- Software that executes and manages containers
- Many choices, Docker is most popular, followed by rkt

Docker, containerd, rkt, lxd, containerd, turbo, Clear, runc, etc…

Container



Docker



- 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 Containers

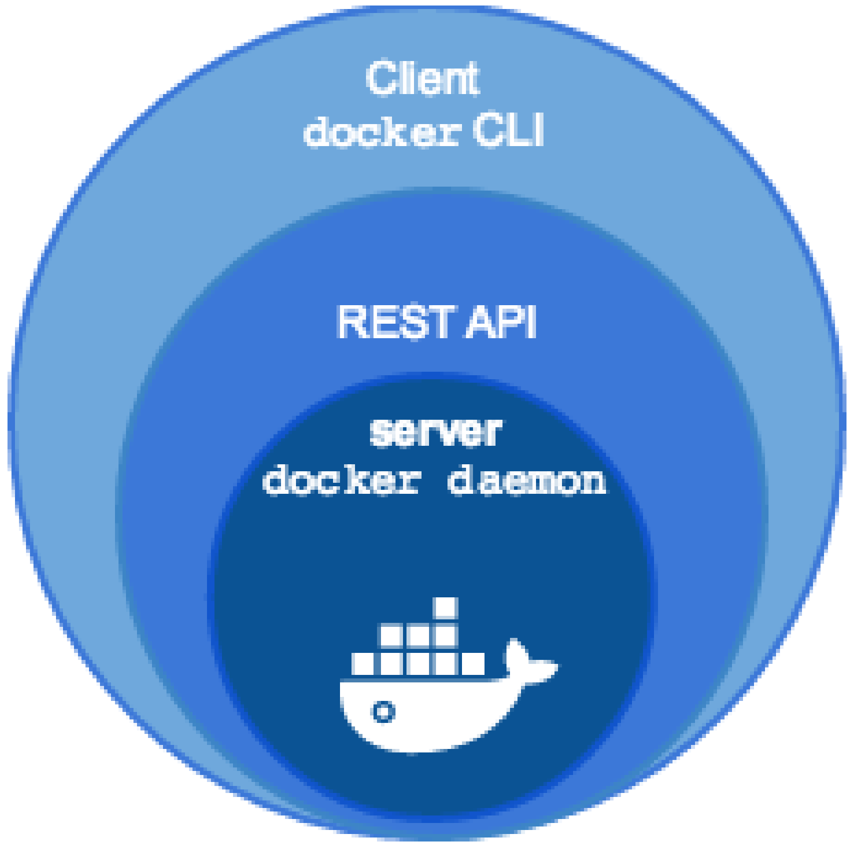
Docker Vocabulary

Host	A VM or on premise server running the Docker Daemon to host a collection of Docker Containers
Image	An *ordered collection of filesystems (layers)* 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

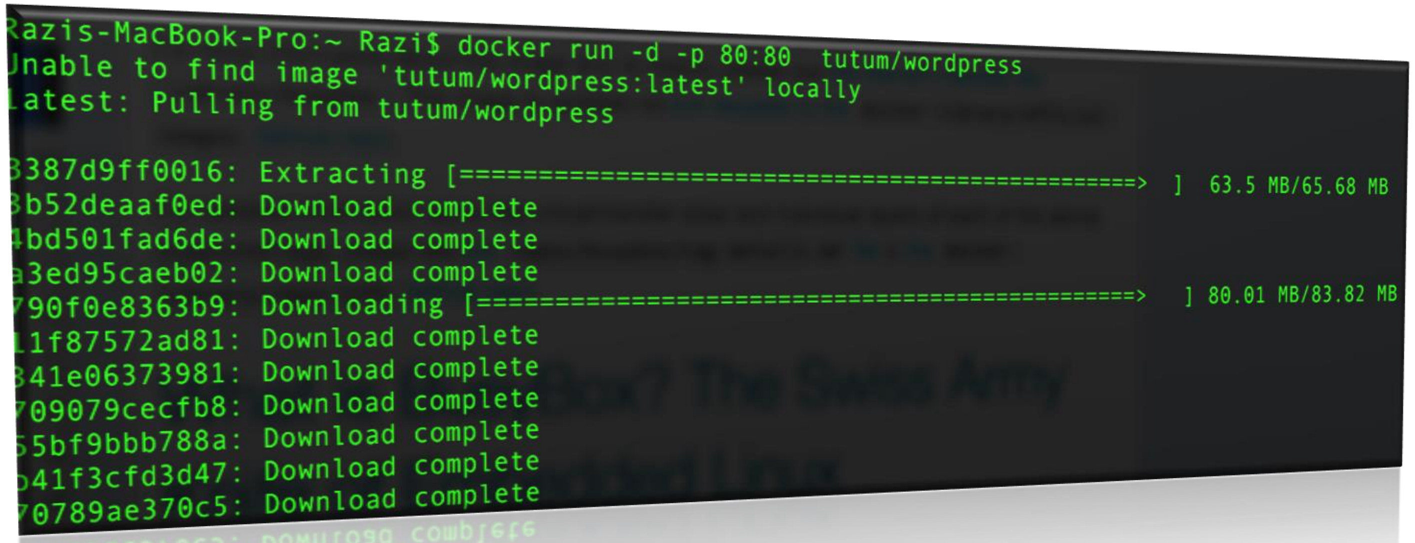
- Docker CLI
  - Allows you to issue Docker commands to create / manage containers.
- Docker API
  - Interface for interacting with the daemon
- Docker Daemon
  - The program that enables containers to be built, shipped, and run.
  - Uses Linux Kernel namespaces and control groups to give an isolated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



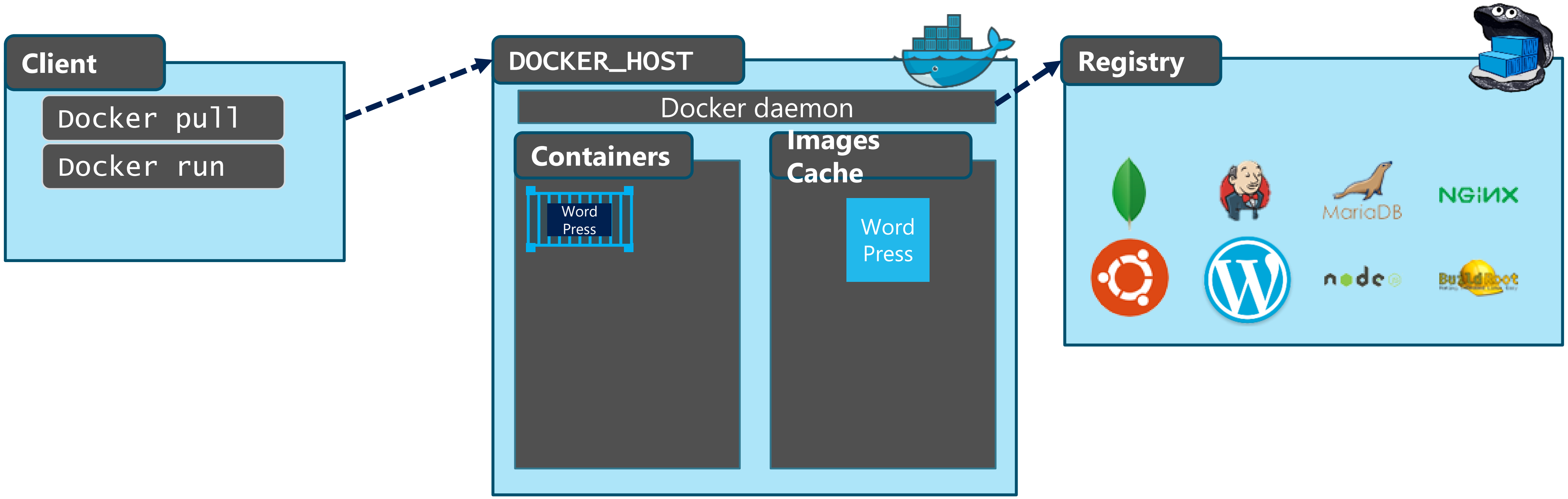
```
azis-MacBook-Pro:~ Razi$ docker run -d -p 80:80  tutum/wordpress
nable to find image 'tutum/wordpress:latest' locally
atest: Pulling from tutum/wordpress

387d9ff0016: Extracting [=================================================>  ]  63.5 MB/65.68 MB
b52deaaf0ed: Download complete
bd501fad6de: Download complete
3ed95caeb02: Download complete
90f0e8363b9: Downloading [================================================>   ] 80.01 MB/83.82 MB
1f87572ad81: Download complete
41e06373981: Download complete
09079cecfb8: Download complete
5bf9bbb788a: Download complete
41f3cfd3d47: Download complete
0789ae370c5: Download complete
```

Demonstration: Running Docker Containers

Launch a single WordPress Container

Running multiple WordPress Containers side by side

Docker In Action

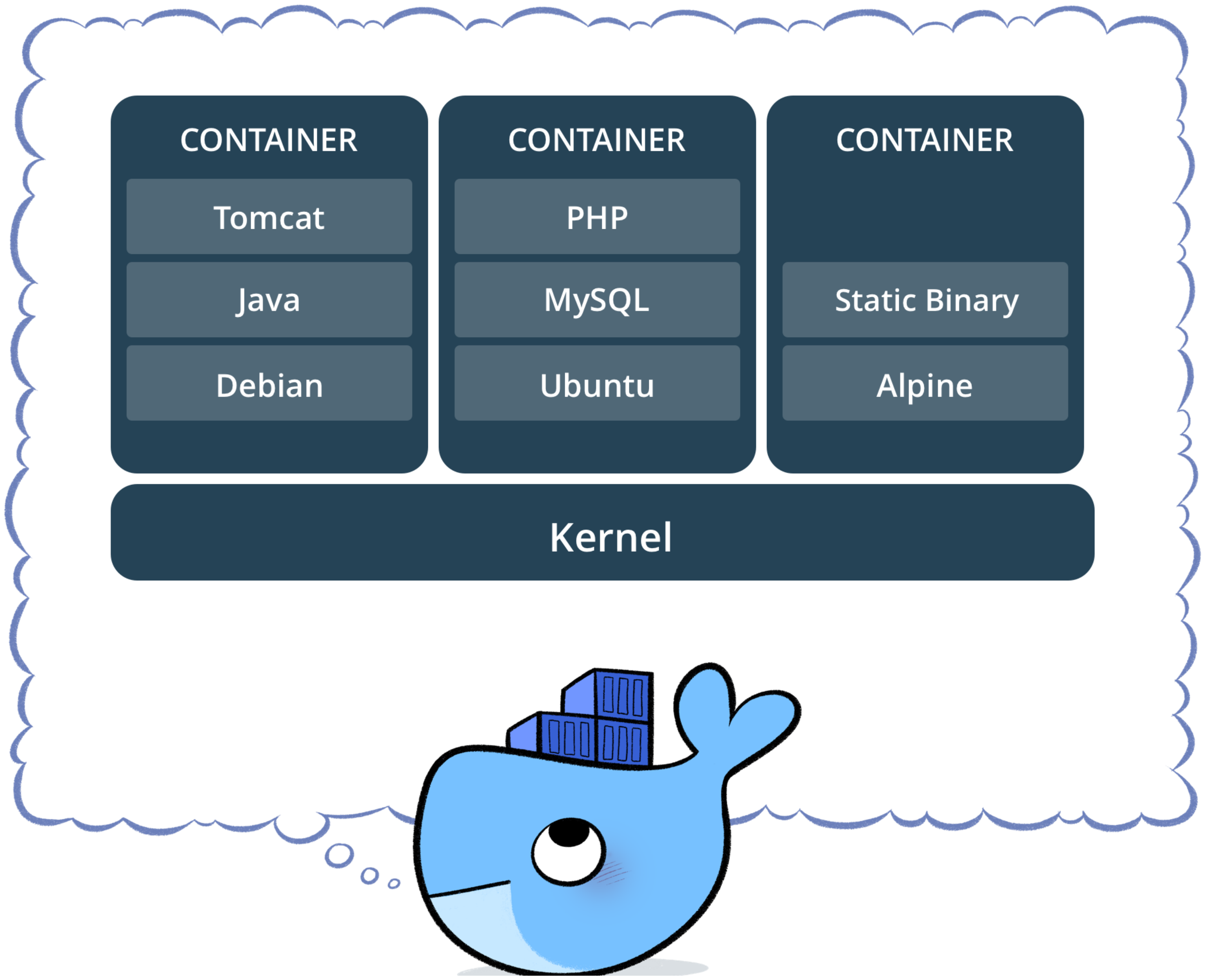


Linux vs. Windows Containers

Linux vs. Windows Containers

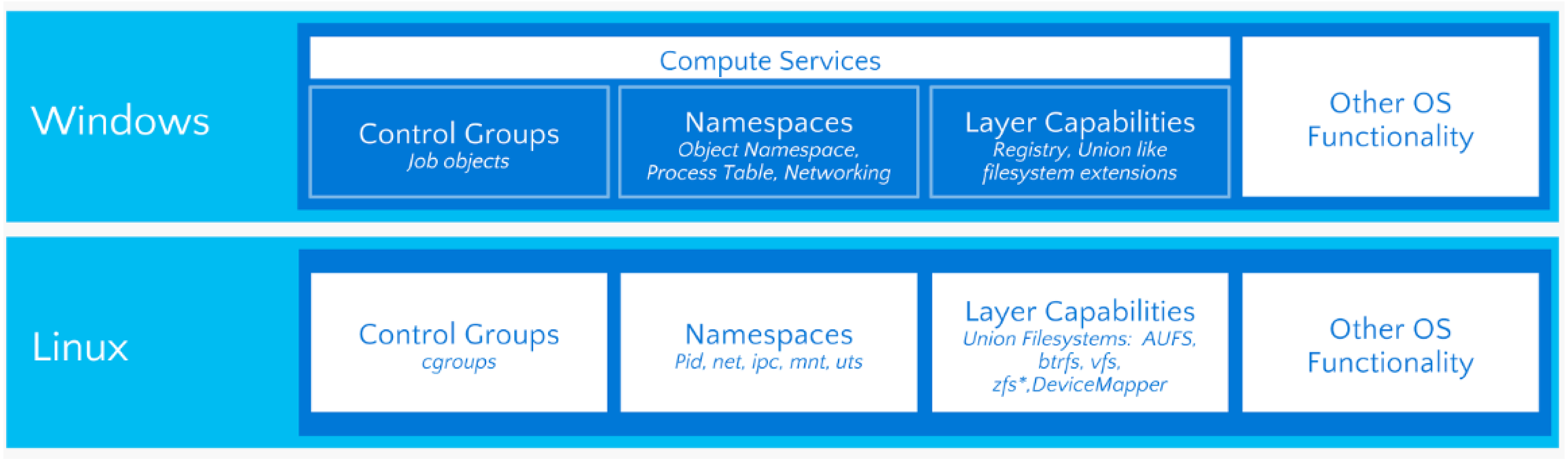
- You can only run Windows containers on a Windows host and Linux containers on a Linux host.

- 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



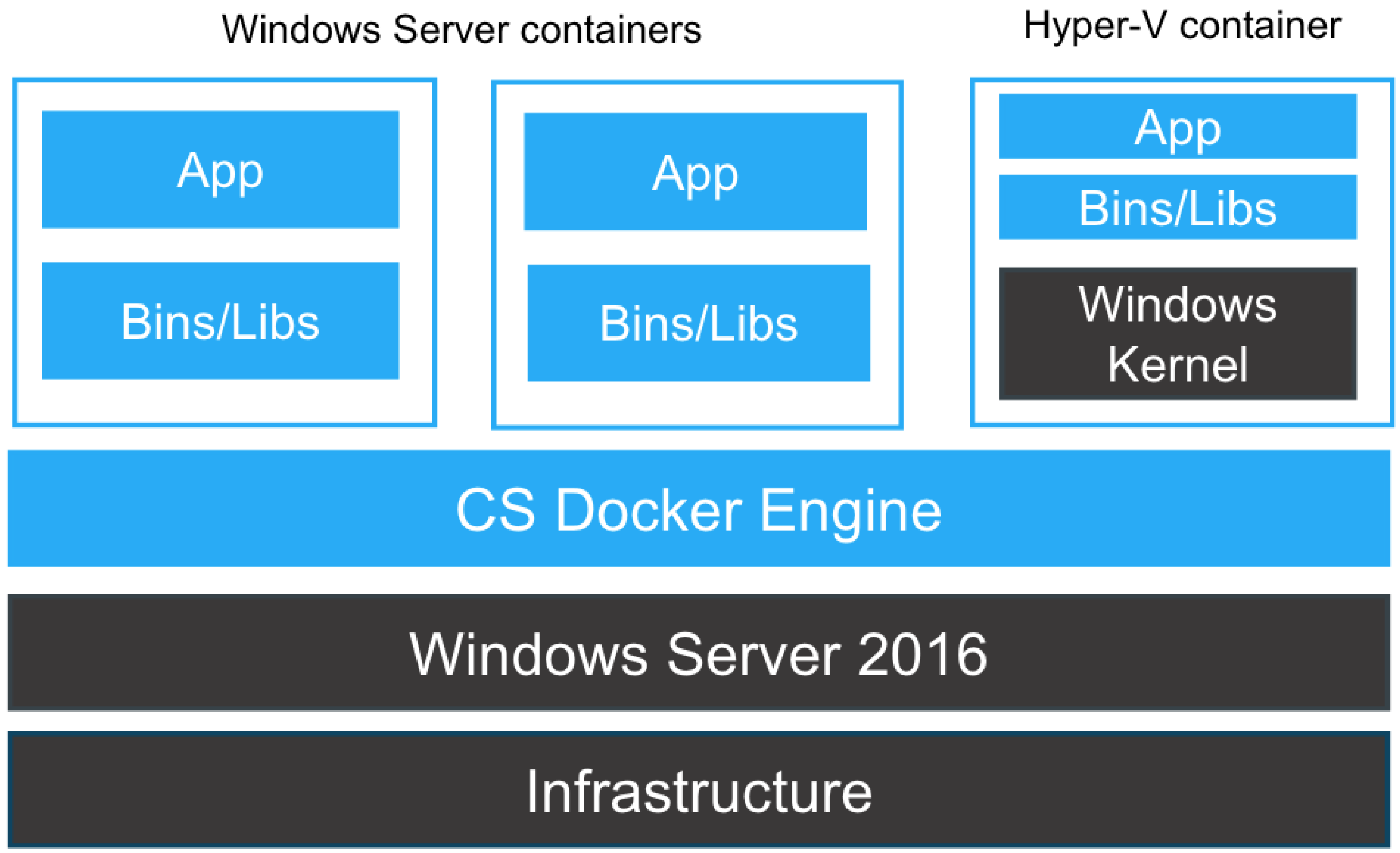
Containers share the kernel with the host OS

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.



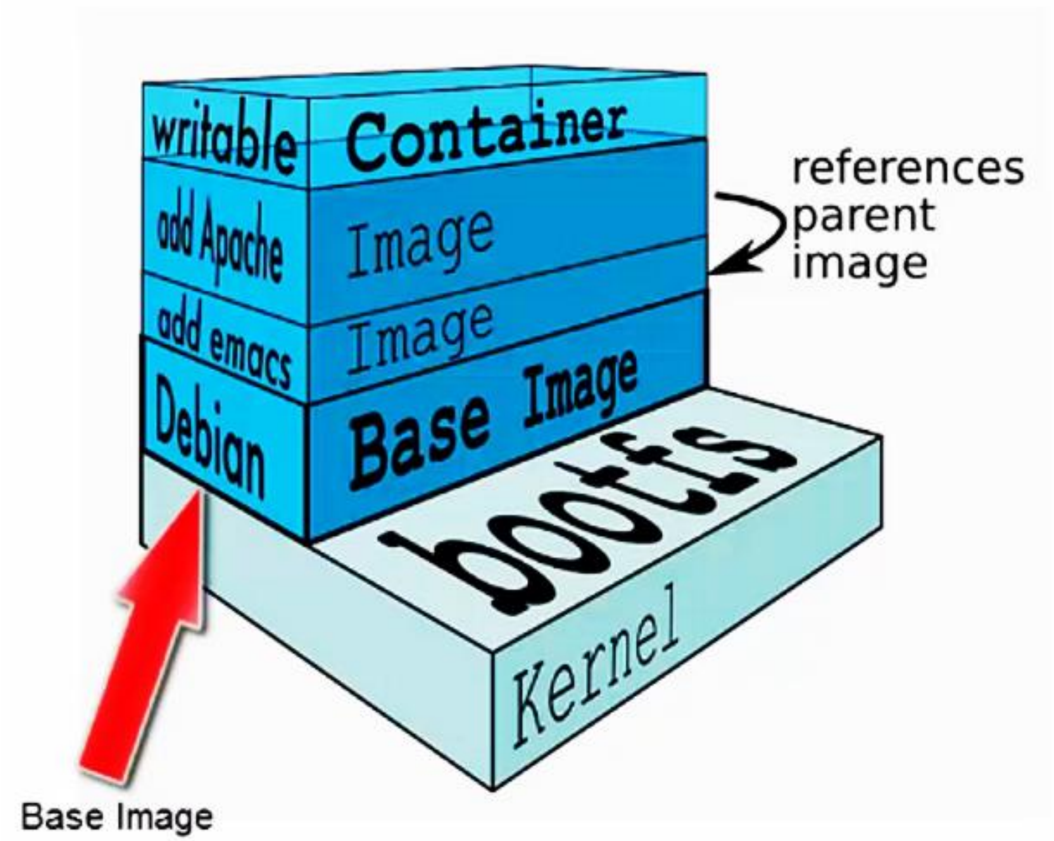
Demonstration: Nano Server

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.



Demonstration: Docker Image Layers

List All Layers for Docker Image

Manifest List – “fat manifest”

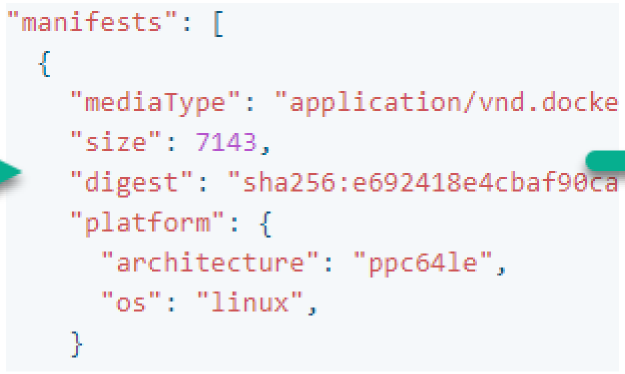
- Points to specific image manifests for one or more platforms.
- Optional.
- A client will distinguish a manifest list from an image manifest based on the Content-Type returned in the HTTP response.

```
{
  "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”

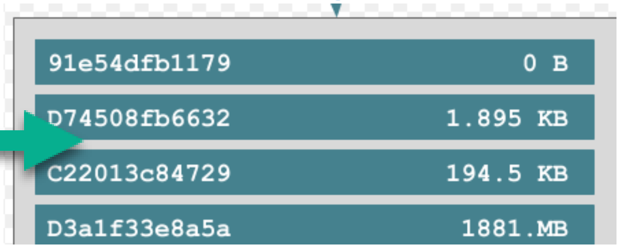
docker pull microsoft/aspnetcore

Fat Manifest



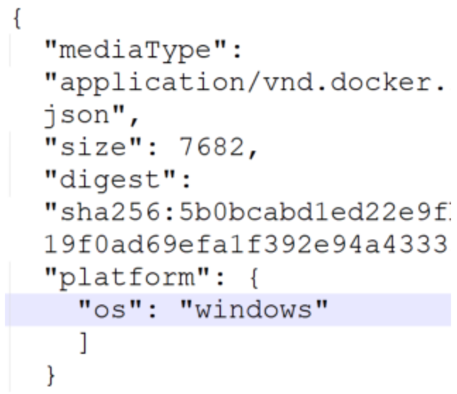
```
"manifests": [
  {
    "mediaType": "application/vnd.docke
    "size": 7143,
    "digest": "sha256:e692418e4cbaf90ca
    "platform": {
      "architecture": "ppc64le",
      "os": "linux",
    }
```

Compatible image

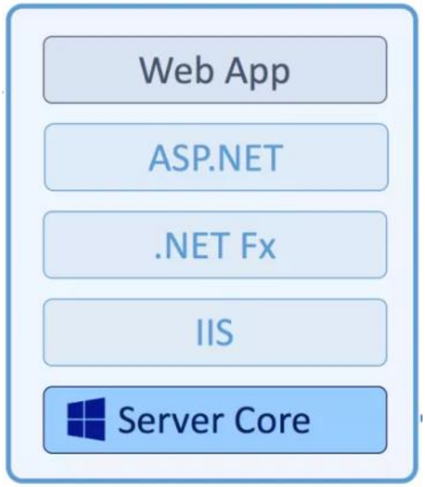


```
91e54dfb1179           0 B
D74508fb6632       1.895 KB
C22013c84729       194.5 KB
D3a1f33e8a5a        1881.MB
```

docker pull microsoft/aspnetcore



```
{
  "mediaType":
  "application/vnd.docker.
  json",
  "size": 7682,
  "digest":
  "sha256:5b0bcabd1ed22e9f
  19f0ad69efa1f392e94a4333
  "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

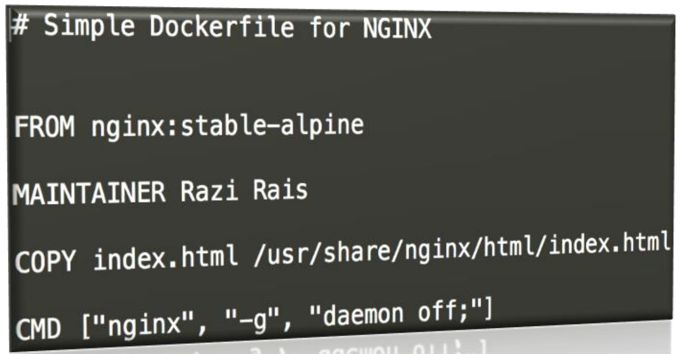
Docker Image Naming Convention

Building Docker images with Dockerfiles

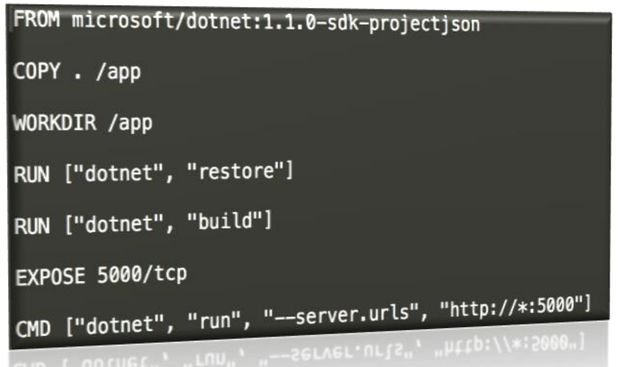
Dockerfile

- 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.

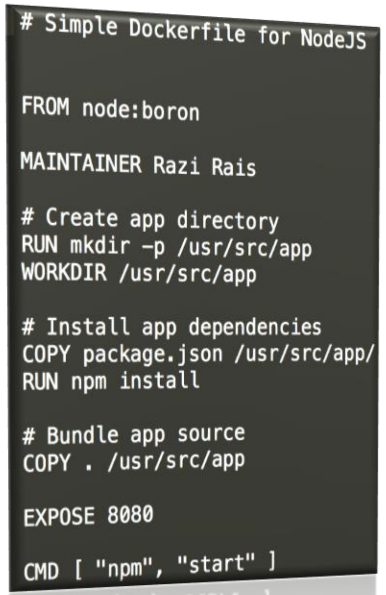
- Docker has a docker build command that parses Dockerfile to build a new container image.



```
# 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;"]
```



```
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"]
```



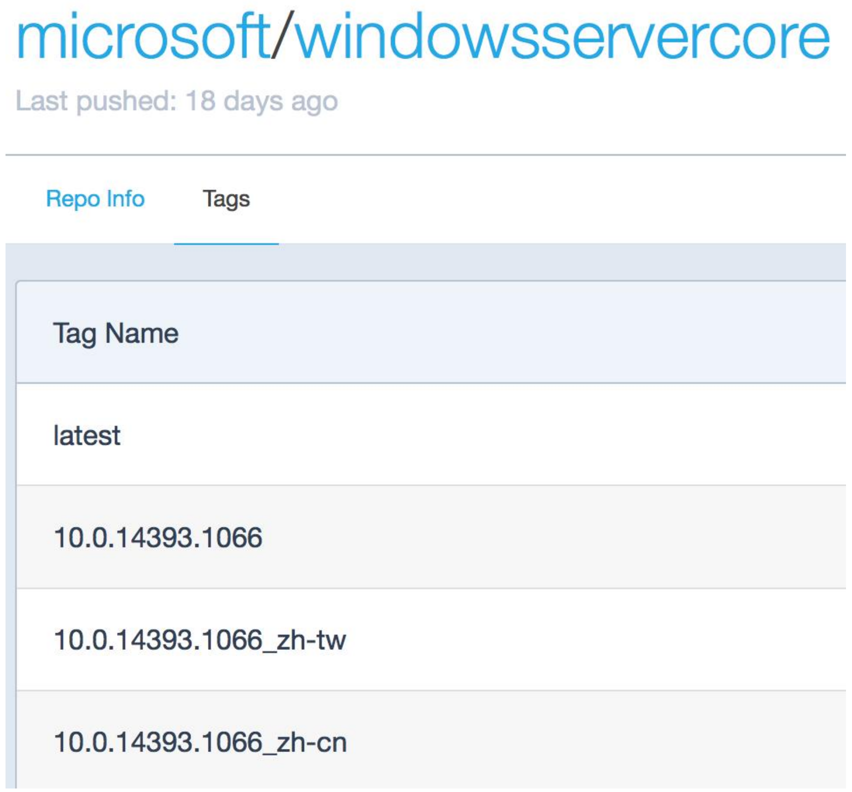
```
# Simple Dockerfile for NodeJS

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" ]
```

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

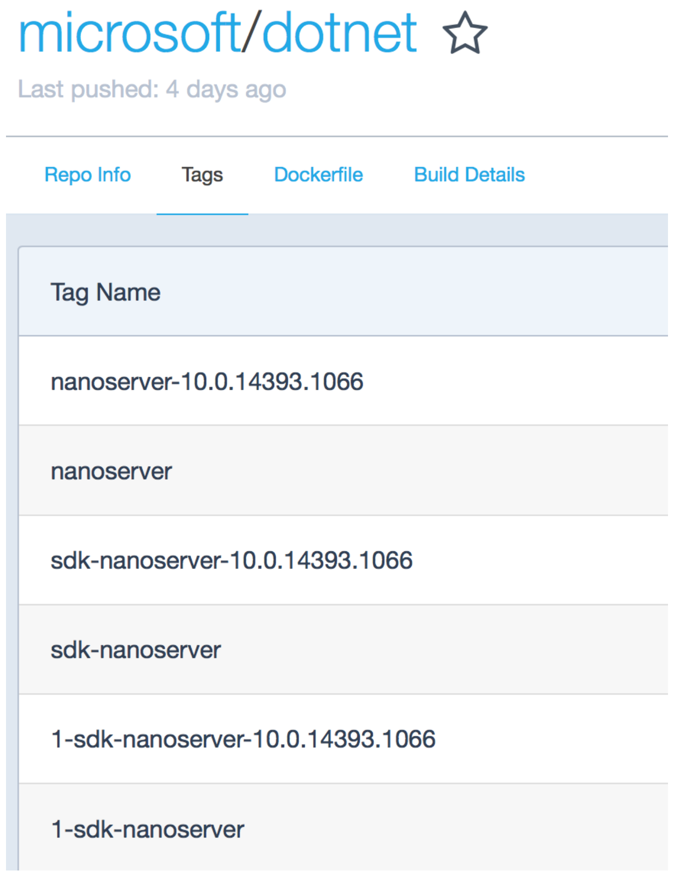
- String value that you can use to distinguish versions of your Docker images.
- PublisherName/ImageName:Tag



microsoft/windowsservercore

Last pushed: 18 days ago

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

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

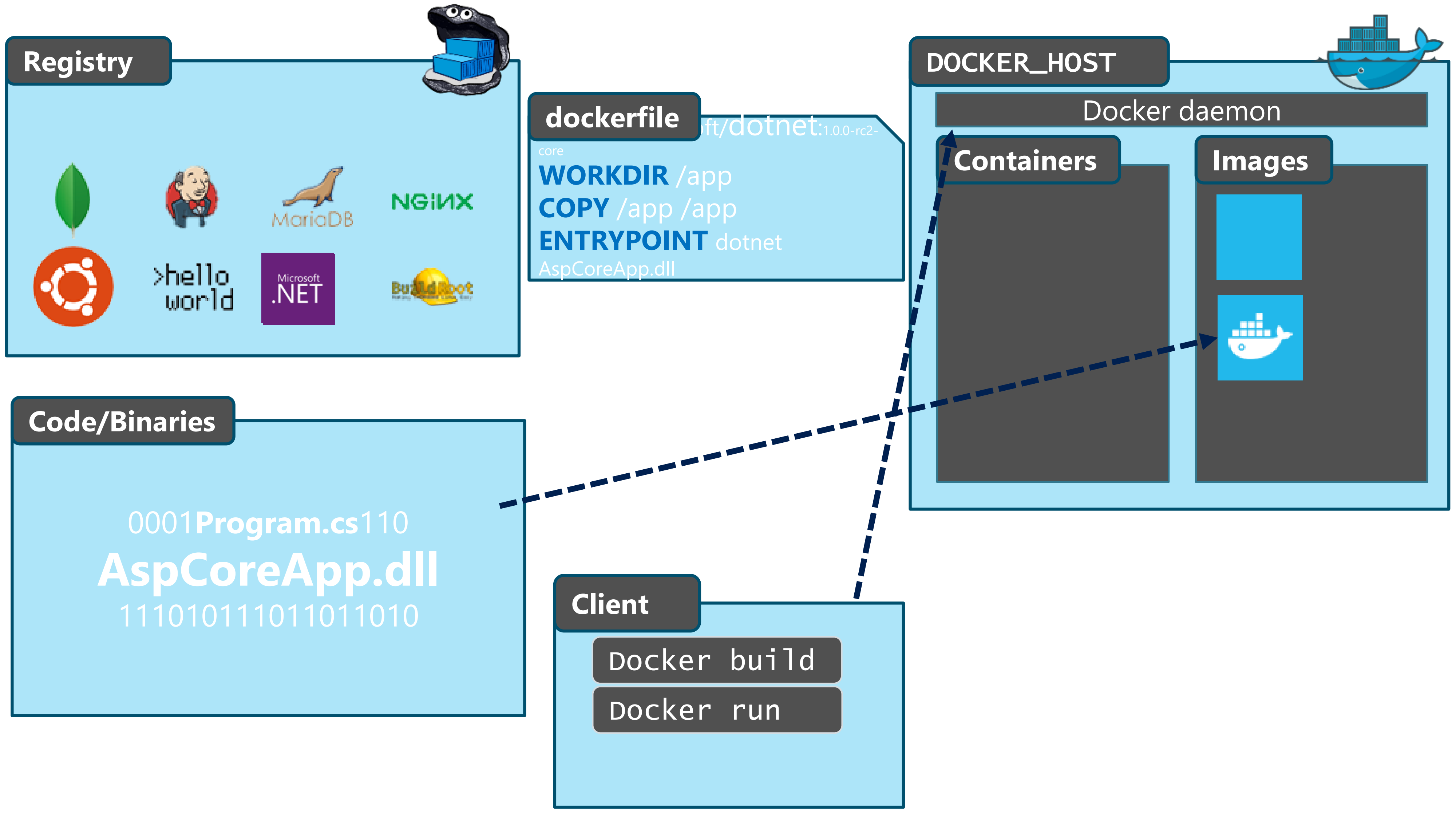
Demonstration: Dockerfile and Docker Build

Build a Dockerfile

Build container images using Docker build command:

- NodeJs

How does Docker build work?



dockerfile

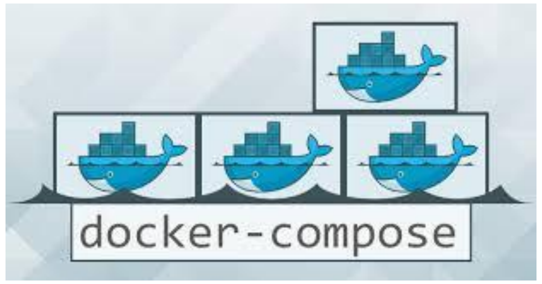
```
ft/dotnet:1.0.0-rc2-
core
WORKDIR /app
COPY /app /app
ENTRYPOINT dotnet
AspCoreApp.dll
```

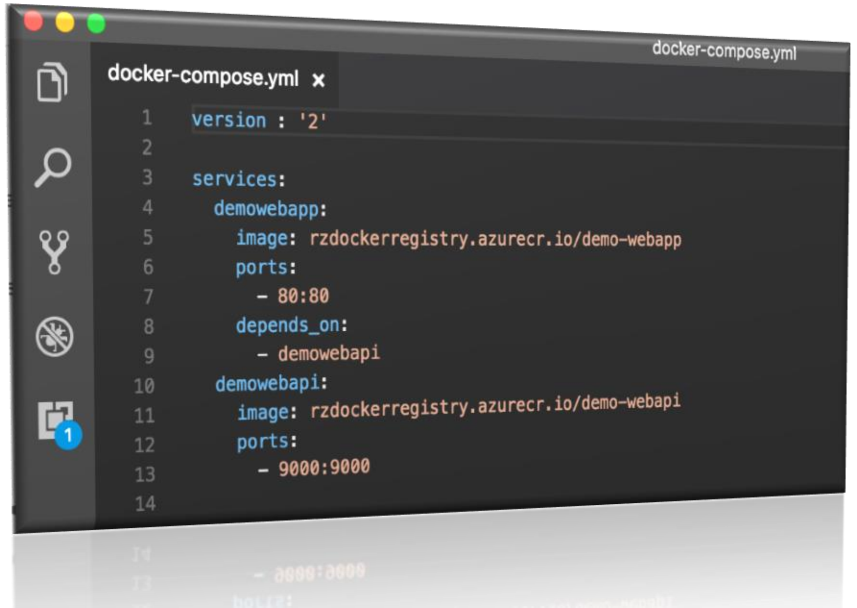
```
Docker build
Docker run
```

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

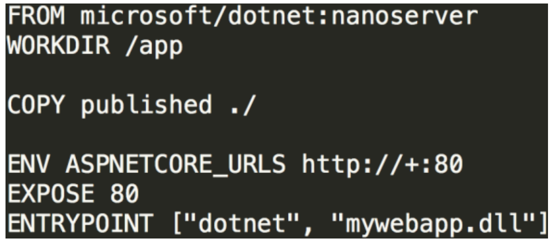




```
version : '2'

services:
  demowebapp:
    image: rzdockerregistry.azurecr.io/demo-webapp
    ports:
      - 80:80
    depends_on:
      - demowebapi
  demowebapi:
    image: rzdockerregistry.azurecr.io/demo-webapi
    ports:
      - 9000:9000

```

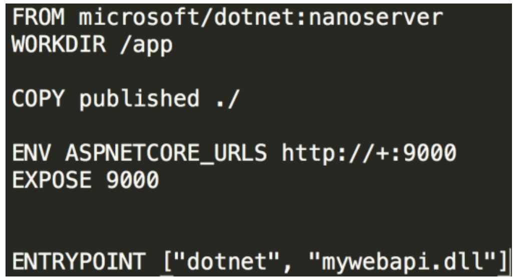
Docker Compose

- 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.



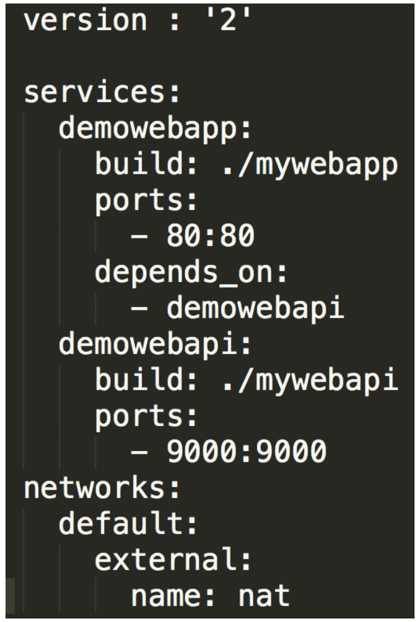
```
FROM microsoft/dotnet:nanoserver
WORKDIR /app

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
```

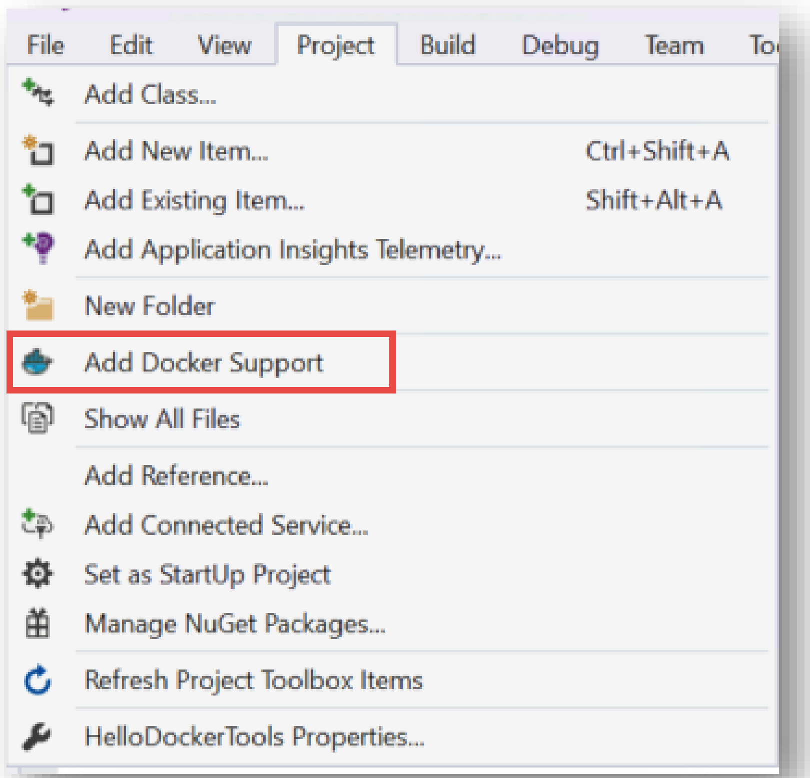
Docker-compose.yml

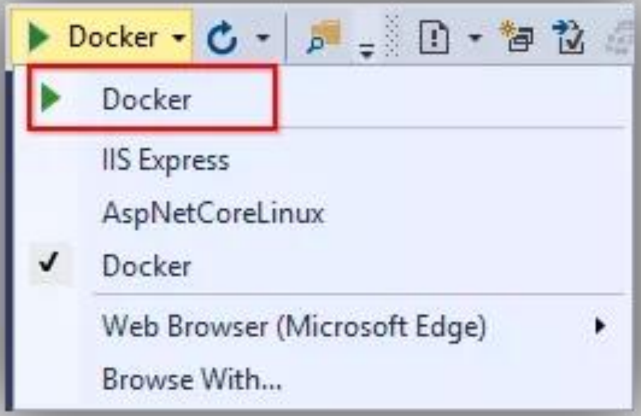
Visual Studio Tools & Docker

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.





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.

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 you 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!