



# Module 1 - Introduction to Containers

Microsoft Services

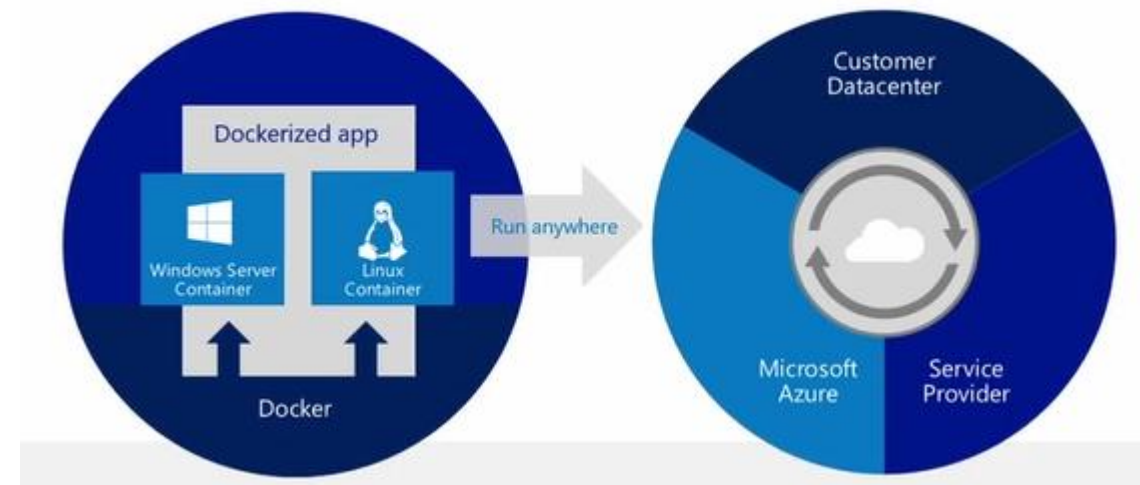


# Objectives

- Understand What Containers Are
- Learn Docker Fundamentals (Docker Engine and Client)
- Understand Container Images and Docker Registry
- Learn How to Build Container Image using Dockerfile
- Learn how to Start, Stop, and Remove Docker Containers
- Understand use of Tags for Versioning Images
- Microsoft Partnership with Docker Inc.

# Why Docker?

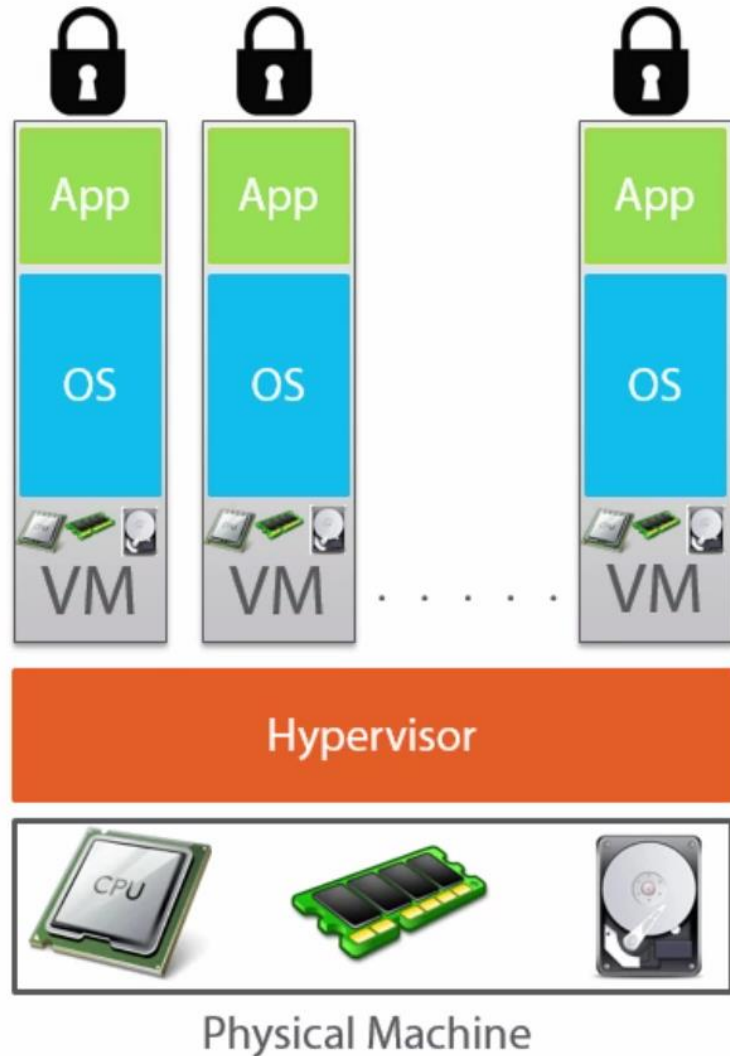
- Build any app in any language using any stack (OS)
- Dockerized apps can run anywhere on anything
- No more “It works on my machine”
- No more dependency daemons so Developers and System admins unite



# Docker Vocabulary

Host	A machine running the Docker Daemon to host a collection of Docker Containers
Client	Where docker commands are executed (client/server)
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

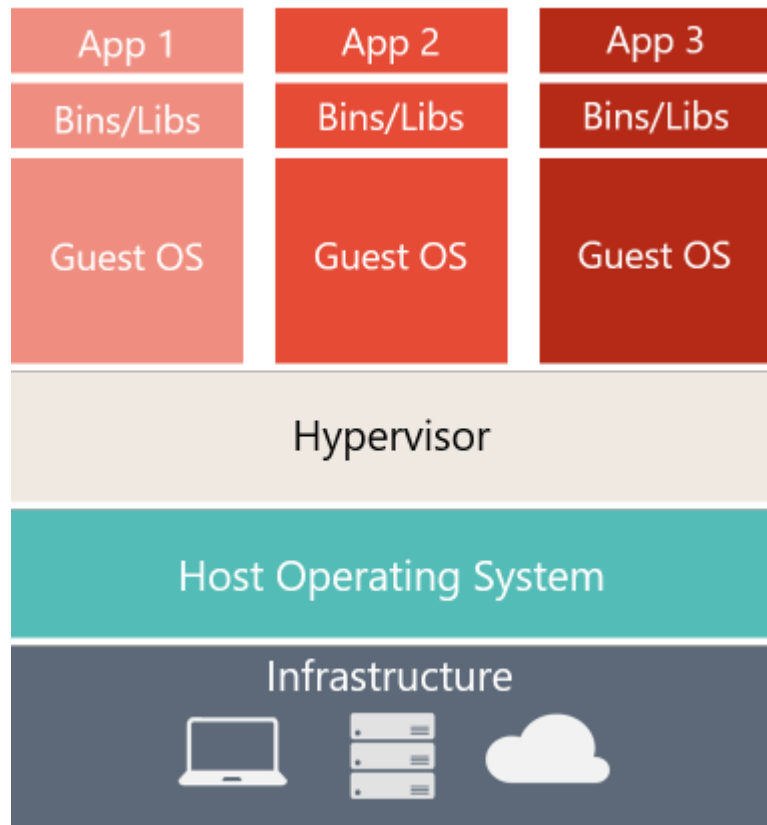
# Challenges with Virtualization



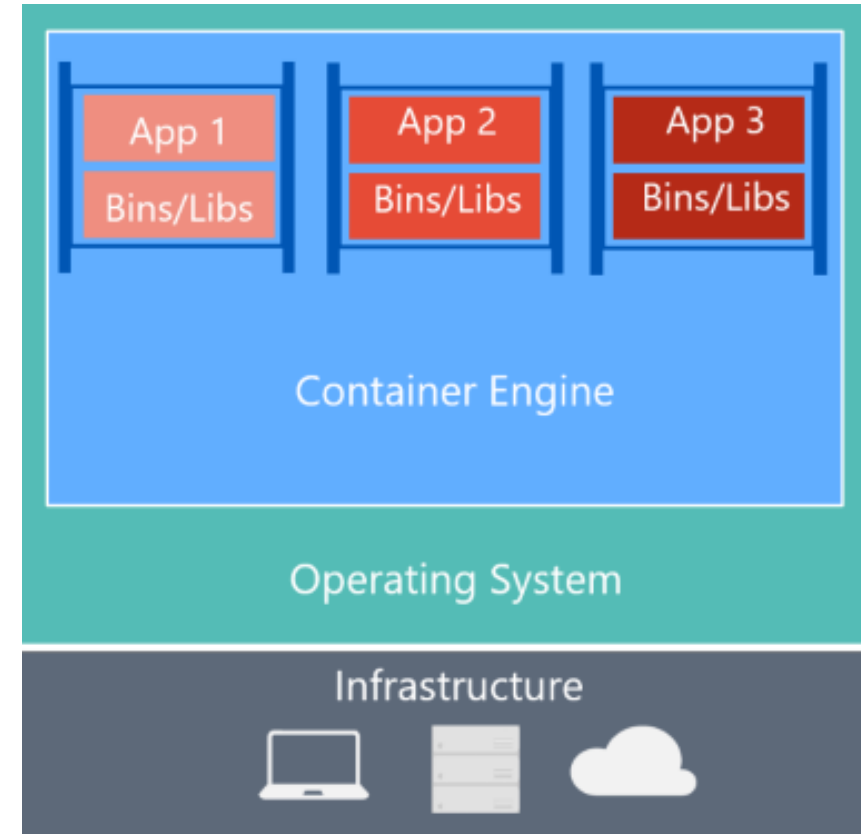
> OS != Business Value

# Virtual Machines versus Containers

## Virtual Machine



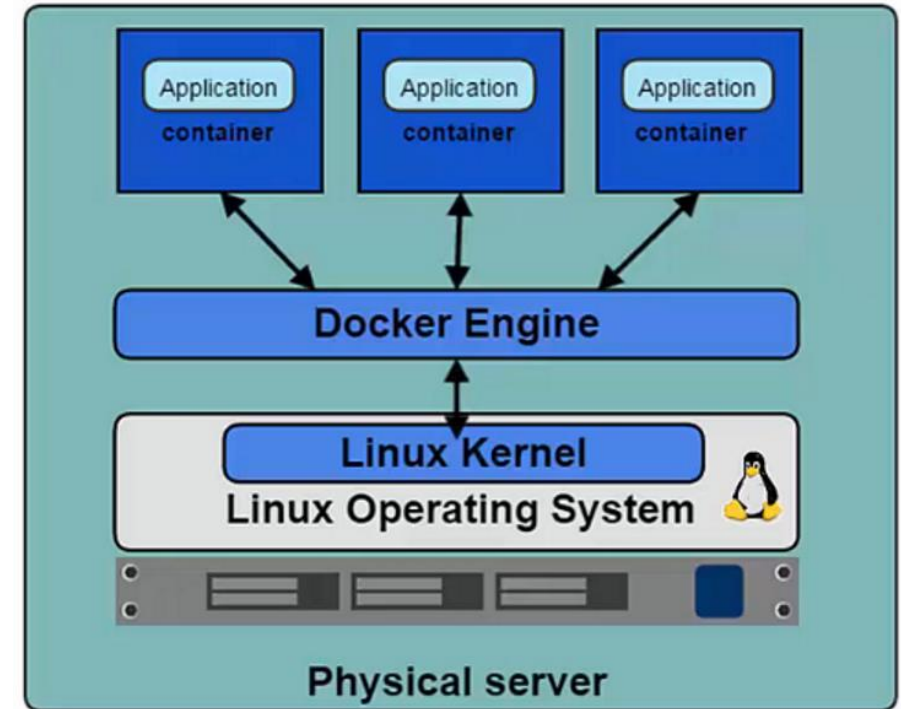
## Container





# Docker Platform

- Docker Engine (a.k.a. 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
- Docker Hub
  - A online registry of Docker images
- Docker Trusted Registry
  - Private on-site Registry for Docker images

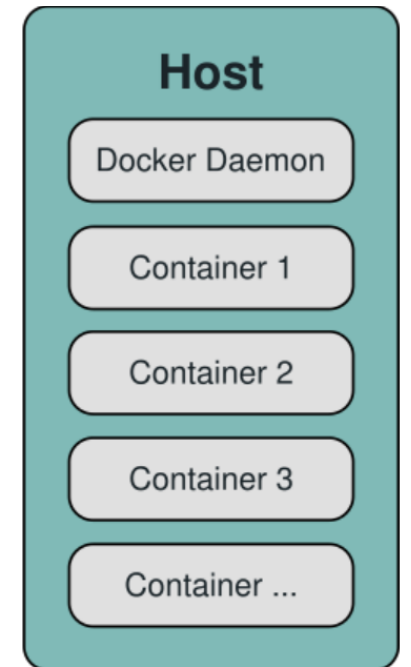


# Docker Platform (Cont.)

- Docker Client
  - Takes user inputs and sends them to the Daemon.
  - Client and Daemon can run on the same host or on different hosts.
- Docker Images
  - Read-only template used to create containers.
  - Contains a set of instructions for creating the containers.
- Docker Containers
  - Isolated application platform based on one or more images.
  - Contains everything needed to run your application.

## Docker Client

`docker pull`  
`docker run`  
`docker ...`





# 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

3387d9ff0016: Extracting [=====> ] 63.5 MB/65.68 MB
8b52deaaf0ed: Download complete
4bd501fad6de: Download complete
a3ed95caeb02: Download complete
790f0e8363b9: Downloading [=====> ] 80.01 MB/83.82 MB
11f87572ad81: Download complete
841e06373981: Download complete
709079cecfb8: Download complete
55bf9bbb788a: Download complete
b41f3cfd3d47: Download complete
70789ae370c5: Download complete
1018096310c2: DOMUJ09q c0mbj6t6
041t3c1q3q41: DOMUJ09q c0mbj6t6
2P10P0P1889: DOMUJ09q c0mbj6t6
00010C6C109: DOMUJ09q c0mbj6t6
```

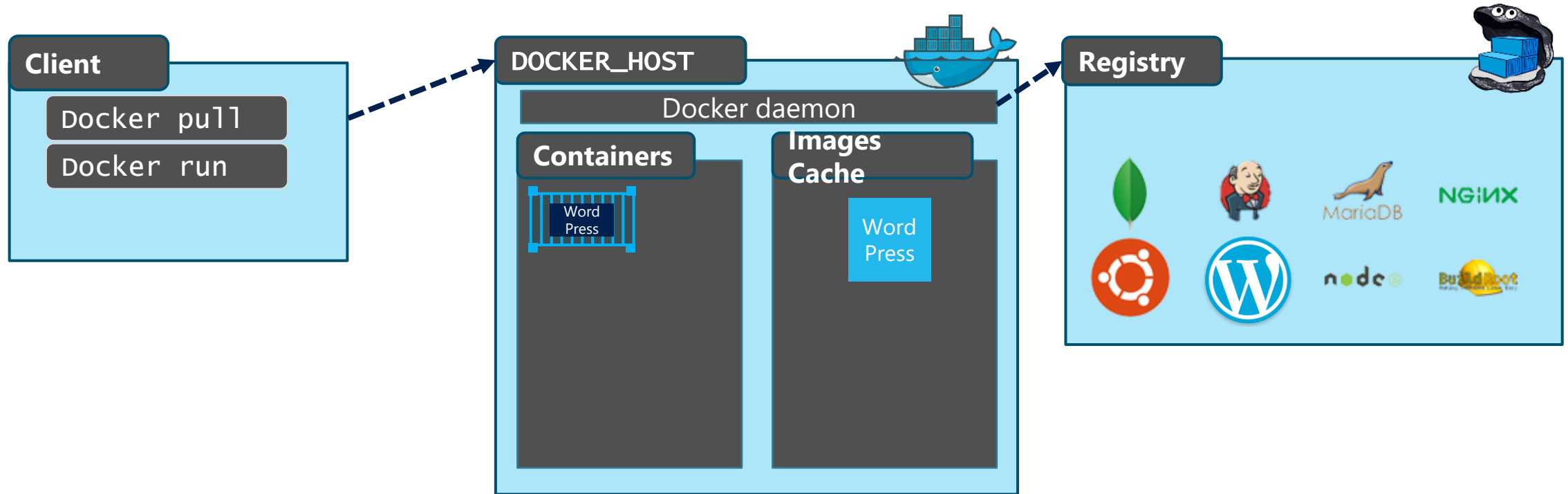
# Demonstration: Running Docker Containers

Launch a single WordPress  
Container

Running multiple WordPress  
Containers side by side

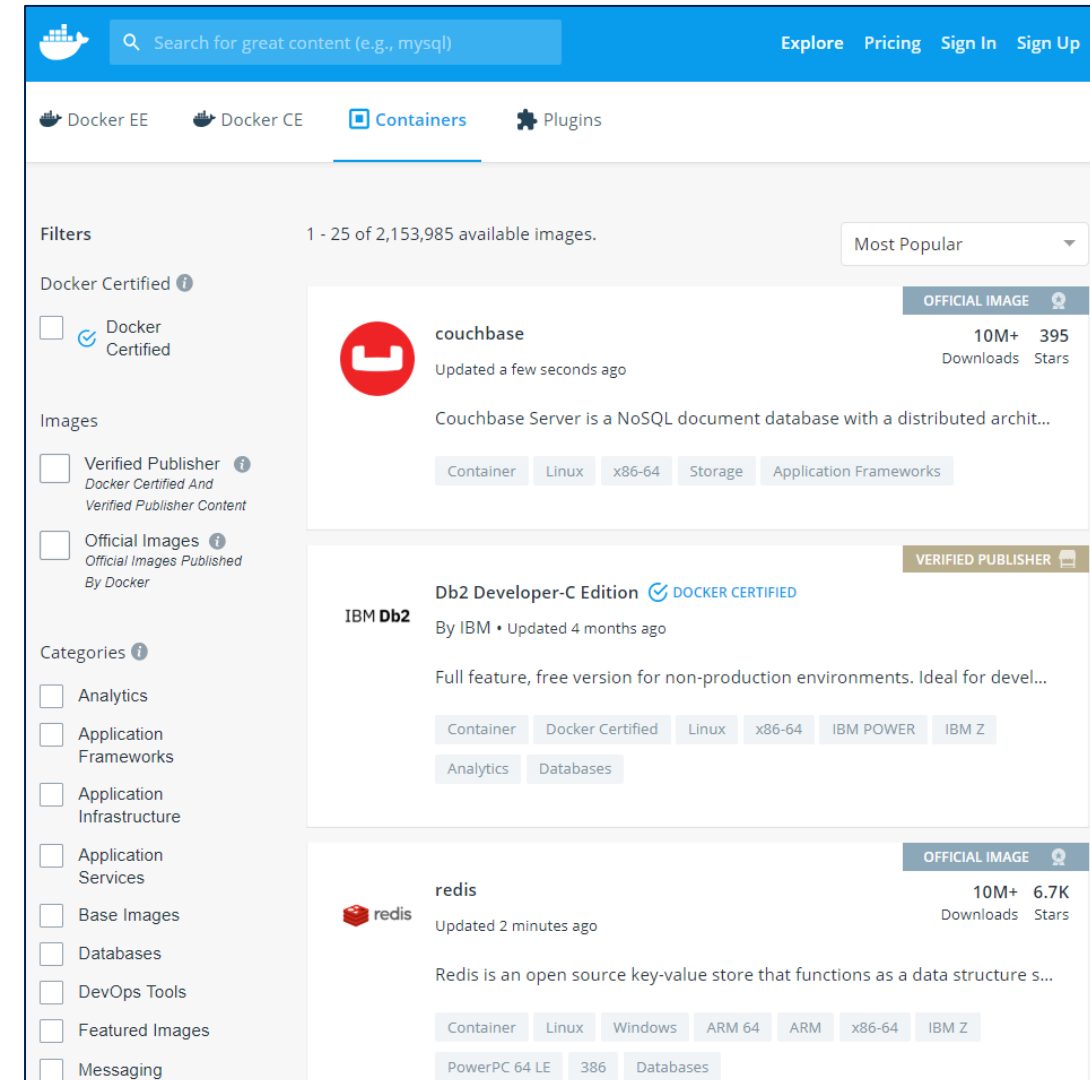


# Docker In Action



# Docker Registry

- Stores docker images
- Searchable
- Public Registry – [hub.docker.com](https://hub.docker.com)
- Private Registries – Instanced for you. E.g. Azure Container Registry
- The Registry is open-source under the permissive Apache License





# Demonstration: Docker Registry

Search Docker Registry using  
Docker CLI

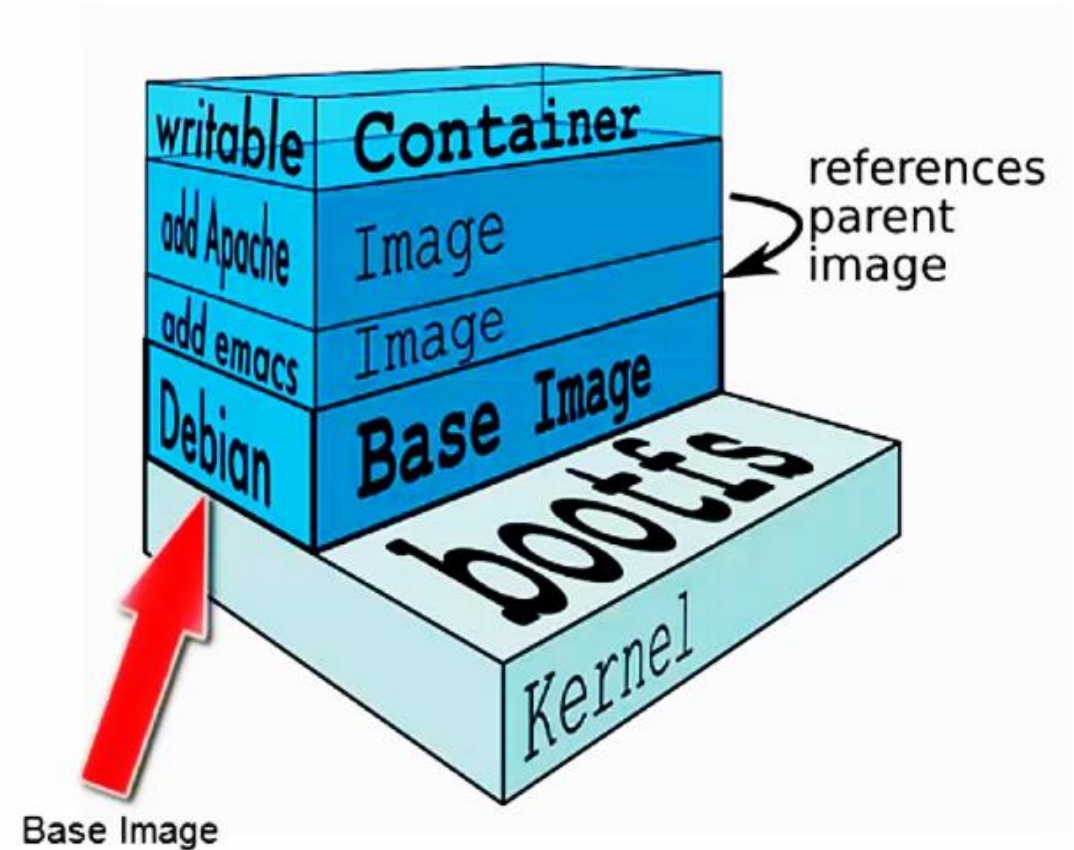
Search Images on DockerHub

Docker Image Naming  
Convention



# 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





# 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 ["/usr/src/app"]
```

```
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 ["/usr/src/app"]
```

```
# 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" ]

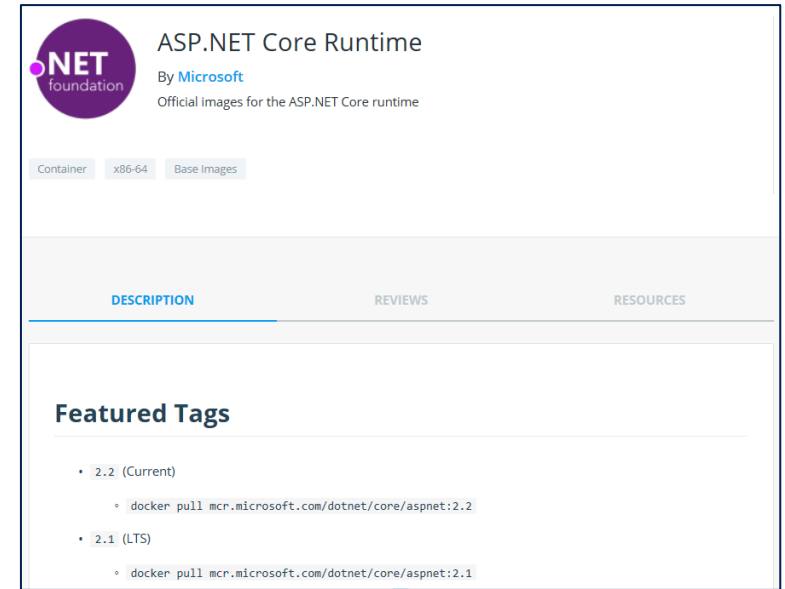
CWD [ "/usr/src/app" ]
```

# 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

- A Tag name is a string value that you can use to distinguish versions of your Docker images so you can preserve older copies or variants of a primary build.
- You can group your images together using names and tags (if you don't provide any tag default value of latest is assumed)



ASP.NET Core Runtime

By Microsoft

Official images for the ASP.NET Core runtime

Container x86-64 Base Images

DESCRIPTION REVIEWS RESOURCES

**Featured Tags**

- 2.2 (Current)
  - `docker pull mcr.microsoft.com/dotnet/core/aspnet:2.2`
- 2.1 (LTS)
  - `docker pull mcr.microsoft.com/dotnet/core/aspnet:2.1`

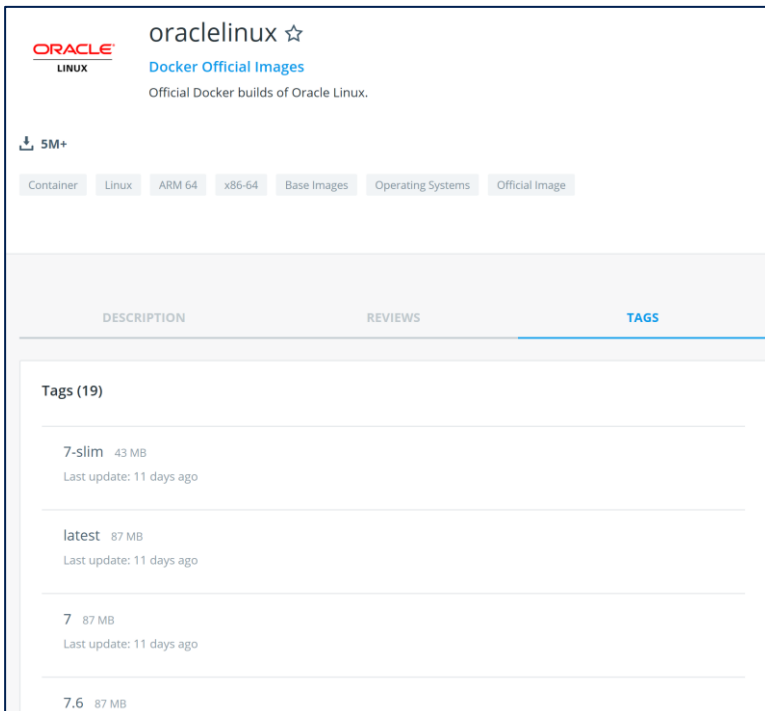


**Linux amd64 tags**

- 2.2.4-stretch-slim, 2.2-stretch-slim, 2.2.4, 2.2, latest (Dockerfile)
- 2.2.4-alpine3.9, 2.2-alpine3.9 (Dockerfile)
- 2.2.4-alpine3.8, 2.2-alpine3.8, 2.2.4-alpine, 2.2-alpine (Dockerfile)
- 2.2.4-bionic, 2.2-bionic (Dockerfile)
- 2.1.10-stretch-slim, 2.1-stretch-slim, 2.1.10, 2.1 (Dockerfile)
- 2.1.10-alpine3.9, 2.1-alpine3.9 (Dockerfile)
- 2.1.10-alpine3.7, 2.1-alpine3.7, 2.1.10-alpine, 2.1-alpine (Dockerfile)
- 2.1.10-bionic, 2.1-bionic (Dockerfile)

**Windows Server, version 1809 amd64 tags**

- 2.2.4-nanoserver-1809, 2.2-nanoserver-1809, 2.2.4, 2.2, latest (Dockerfile)
- 2.1.10-nanoserver-1809, 2.1-nanoserver-1809, 2.1.10, 2.1 (Dockerfile)



ORACLE LINUX

oraclelinux ☆

Docker Official Images

Official Docker builds of Oracle Linux.

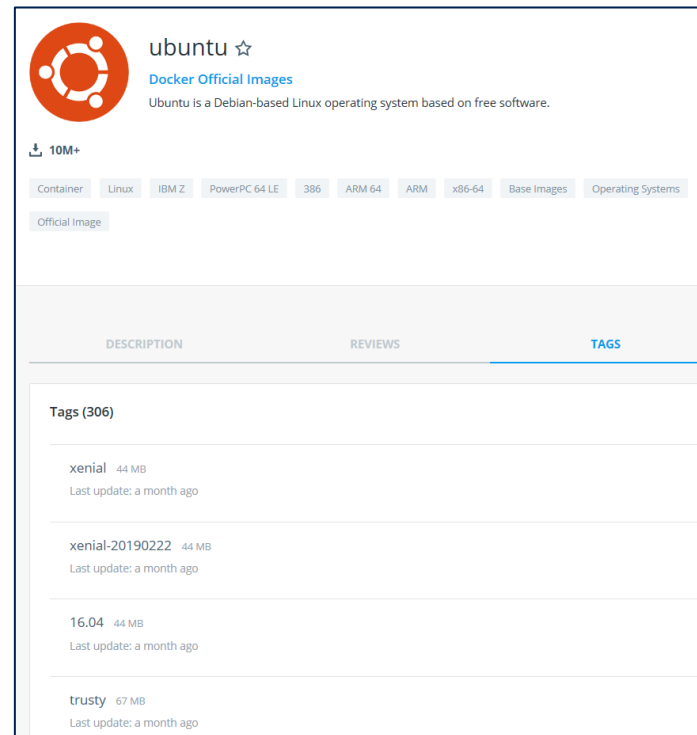
5M+

Container Linux ARM 64 x86-64 Base Images Operating Systems Official Image

DESCRIPTION REVIEWS TAGS

**Tags (19)**

- 7-slim 43 MB  
Last update: 11 days ago
- latest 87 MB  
Last update: 11 days ago
- 7 87 MB  
Last update: 11 days ago
- 7.6 87 MB



ubuntu ☆

Docker Official Images

Ubuntu is a Debian-based Linux operating system based on free software.

10M+

Container Linux IBM Z PowerPC 64 LE 386 ARM 64 ARM x86-64 Base Images Operating Systems

Official Image

DESCRIPTION REVIEWS TAGS

**Tags (306)**

- xenial 44 MB  
Last update: a month ago
- xenial-20190222 44 MB  
Last update: a month ago
- 16.04 44 MB  
Last update: a month ago
- trusty 67 MB  
Last update: a month ago

# Demonstration: Dockerfile and Docker Build

Working with Dockerfile

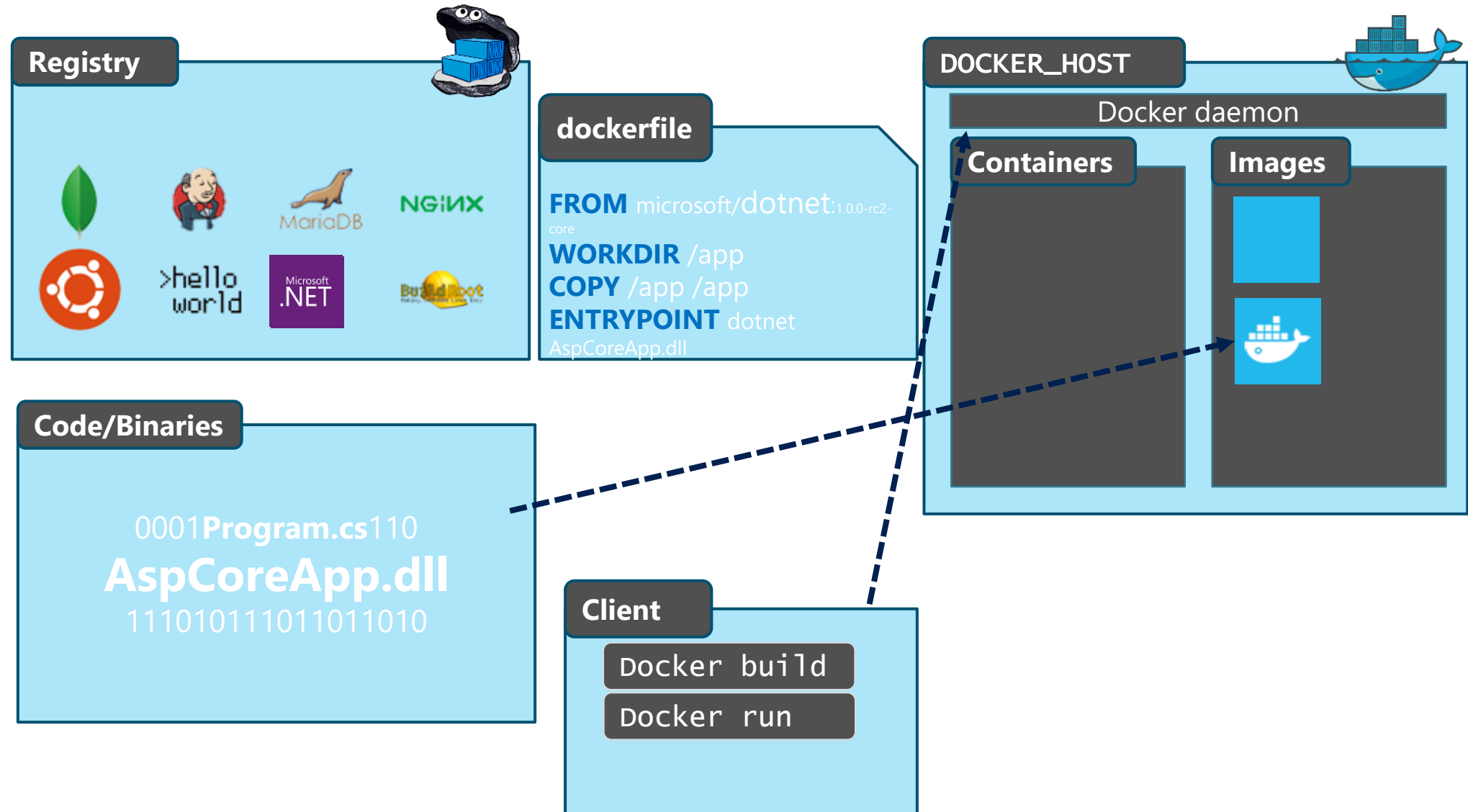
Build container images using Docker  
build command:

- NodeJs
- Nginx
- ASP.NET Core

Using Image Tags

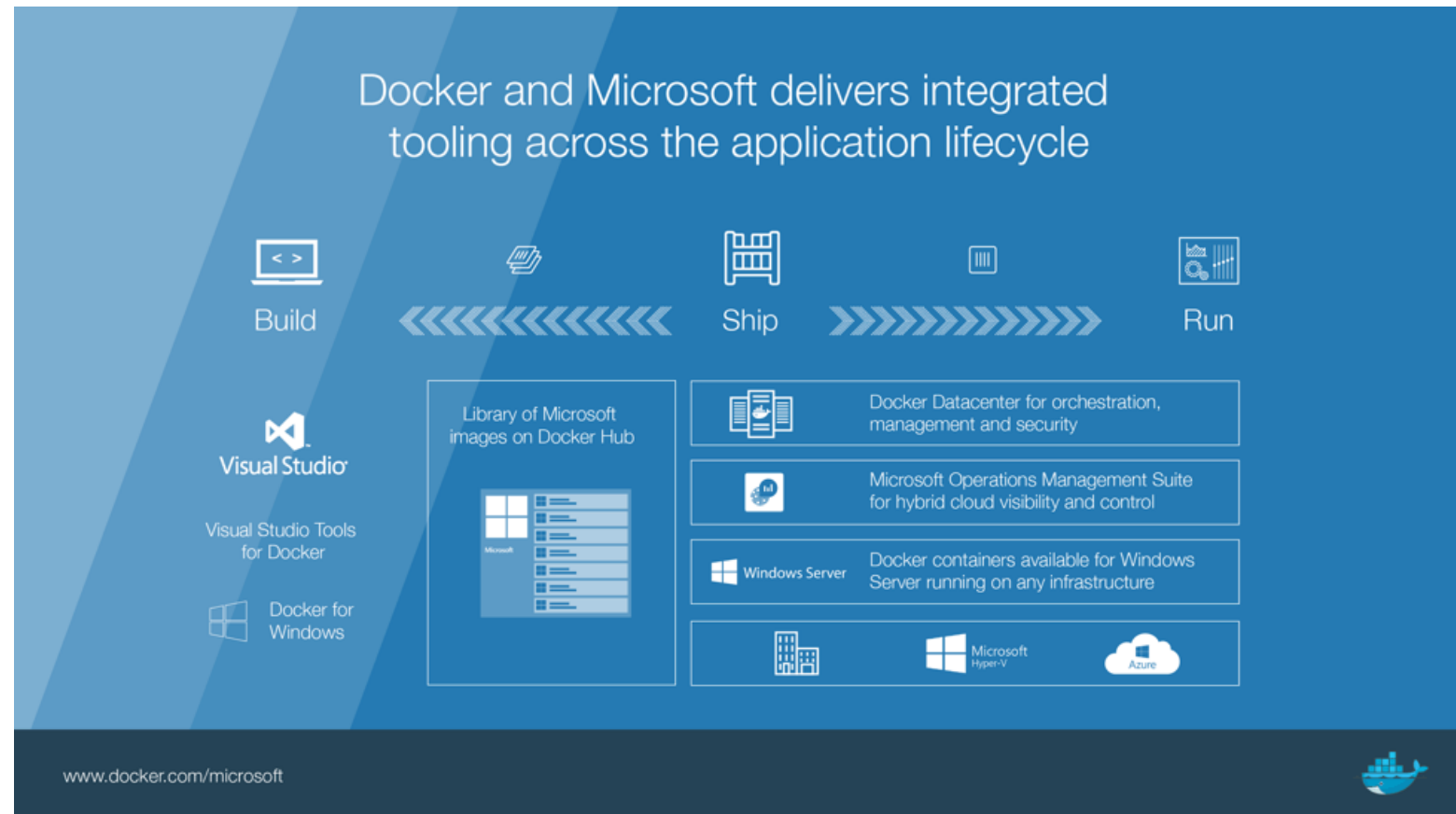


# How does Docker build work?



# Docker and Microsoft Partnership

- Docker Engine is tested, validated, and supported on Windows Server 2016/2019 and Windows 10 at no additional cost.
- Microsoft provides Windows Server 2016/2019 customers enterprise support for Docker Enterprise Edition.
- Docker is supported throughout Microsoft Cloud and on-premises ecosystem.





# Docker and Microsoft Partnership (Cont.)

- For developers, the integration of Visual Studio Tools for Docker and Docker Desktop provides complete desktop development environments for building Dockerized Windows apps
- To jumpstart app development, Microsoft has contributed Windows Server container base images and apps to Docker Hub
- For IT pros, Docker Enterprise for Azure is designed to manage Windows Server environments in addition to the Linux environments with the Docker Universal Control Plane

