Microsoft

Containers as Infrastructure

Introduction to Docker Containers

Microsoft Services

Objectives

- Learn how containers work
- Learn about all the benefits of containers
- Learn about Docker core concepts

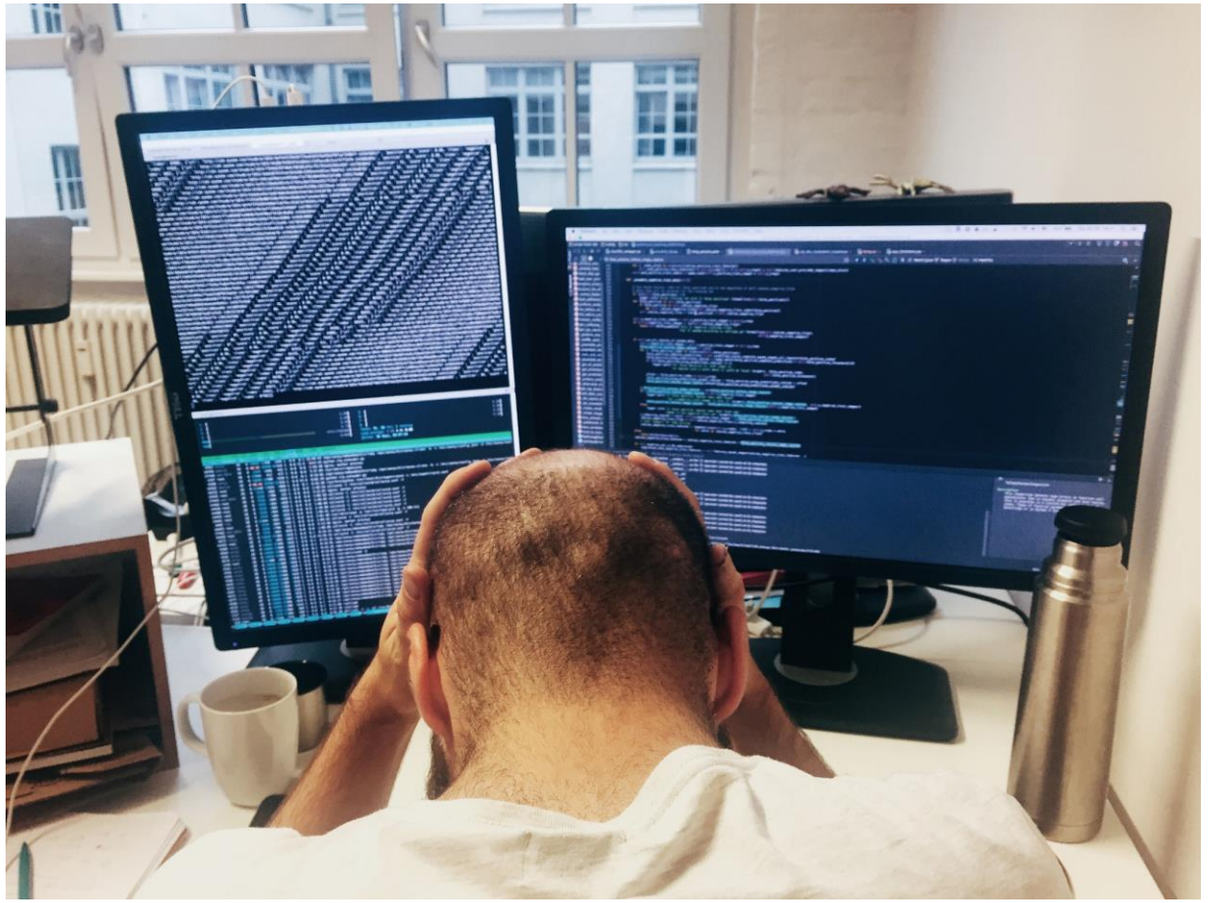
Microsoft

Introduction to Docker Containers

Overview of Containers

Microsoft Services

Why containers?



Transforming Existing Applications into Cloud Applications Is Hard!

Building Hybrid Cloud Applications Is Hard!

Why containers?

Containers make it much easier…

How do containers make it easier?

Agility

Fast Interaction

Portability

Easily move workloads

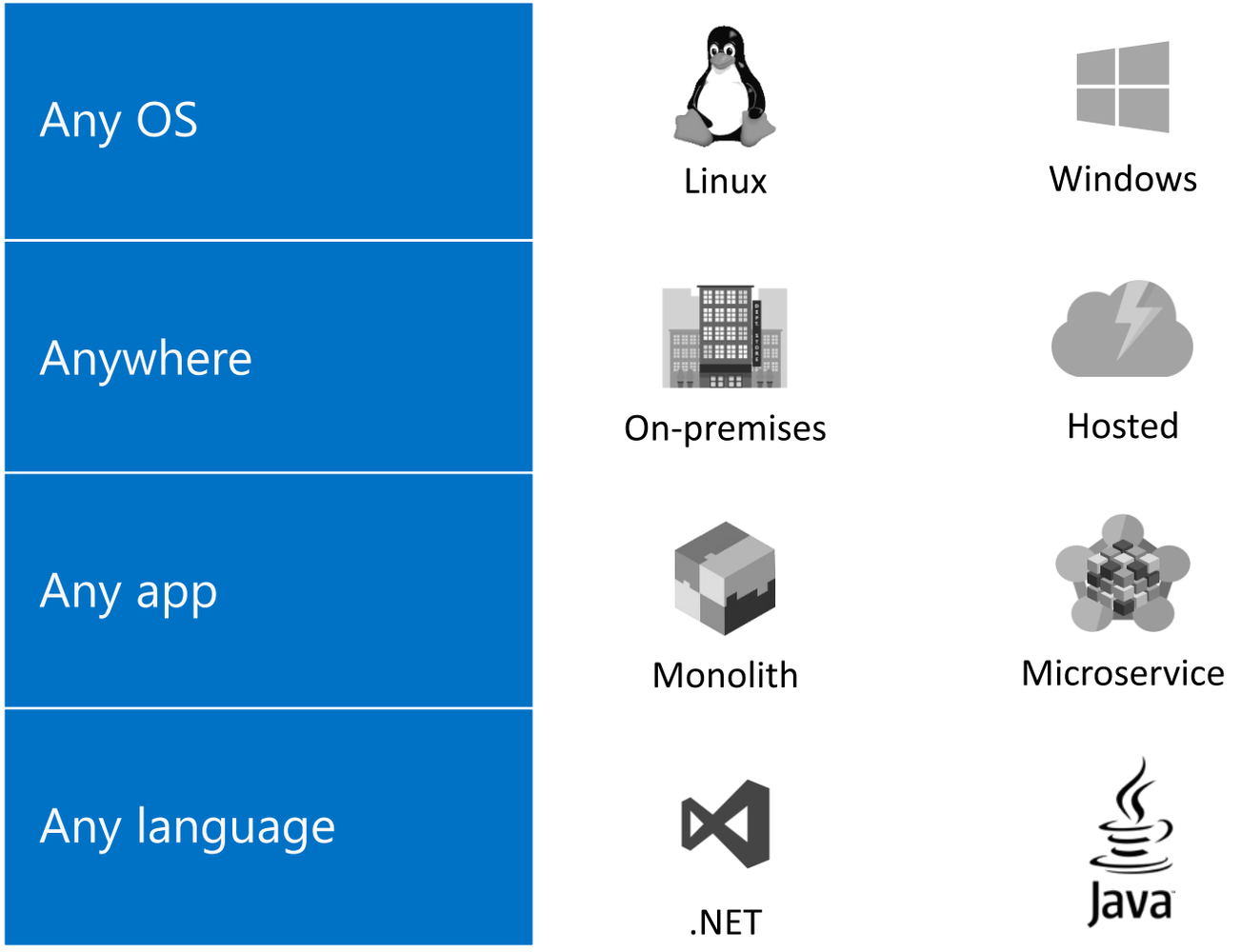
Density

Achieve resource efficiency

Rapid scale

Scale easily to meet demand

The Benefits of Using Containers



The Benefits to Developers and IT

Developers

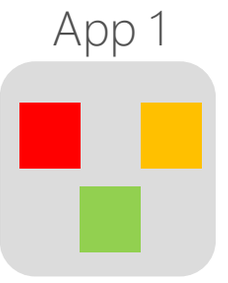
IT Operations

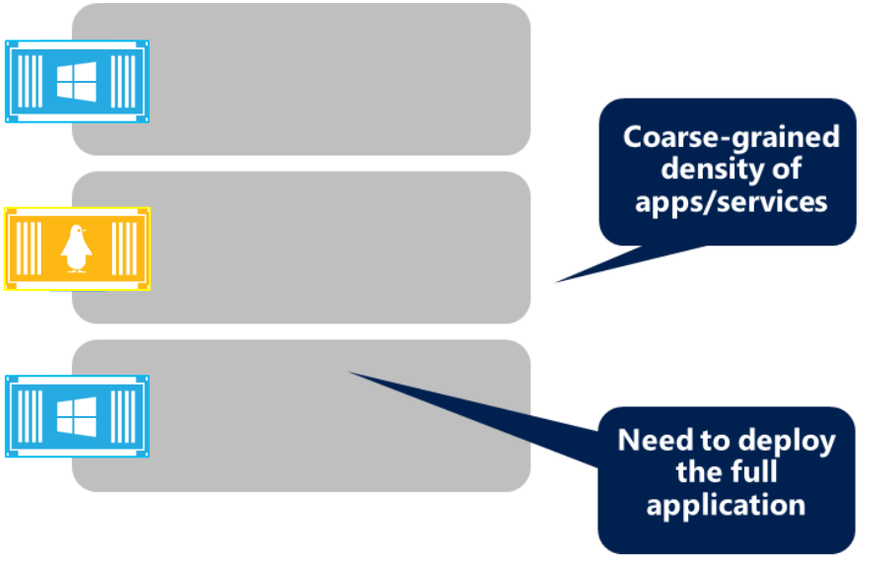
Developers	IT Operations
Easily contain a local development environment	Containers are a single OS and are more efficient at resource utilization
Containers enable microservices architectures	Low overhead, allow for better performance and higher density
Containers allow for efficiency and agility	More secure by default and isolates resources
Empower CI/CD integration	Are faster to provision

The Benefits for Applications

Traditional application approach

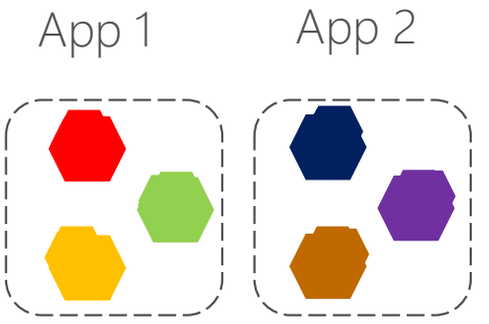
- A traditional application has most of its functionality within a few processes that are componentized with layers and libraries
- Scales by cloning the app on multiple servers and VMs

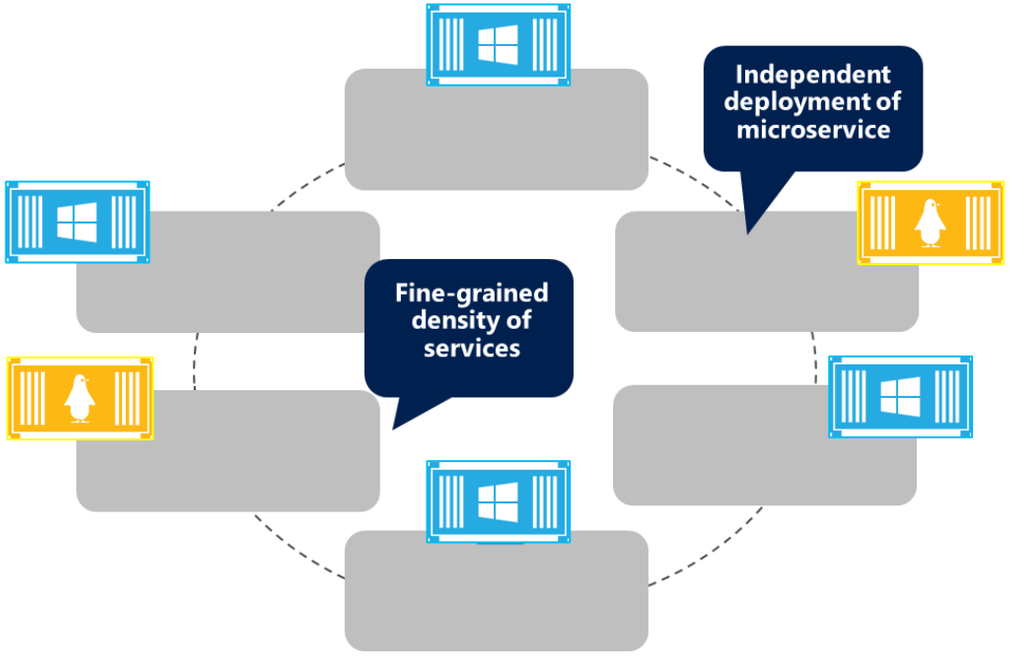




Microservices application approach

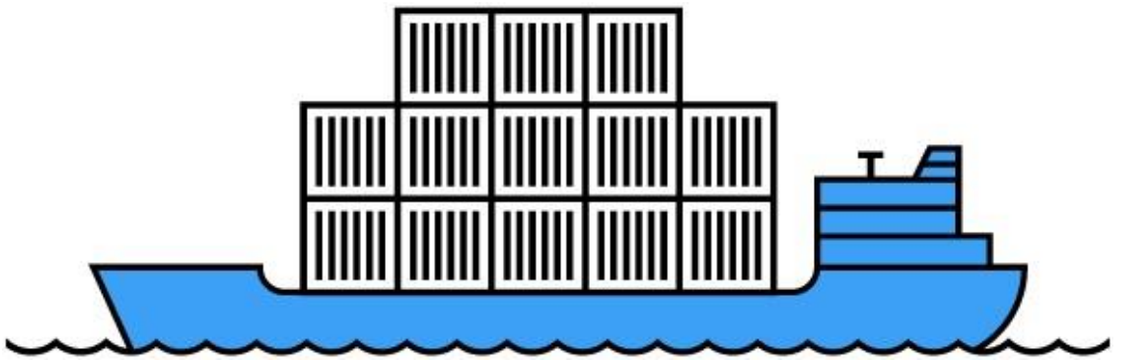
- A microservice application segregates functionality into separate smaller services
- Scales out by deploying each service independently with multiple instances across servers and VMs



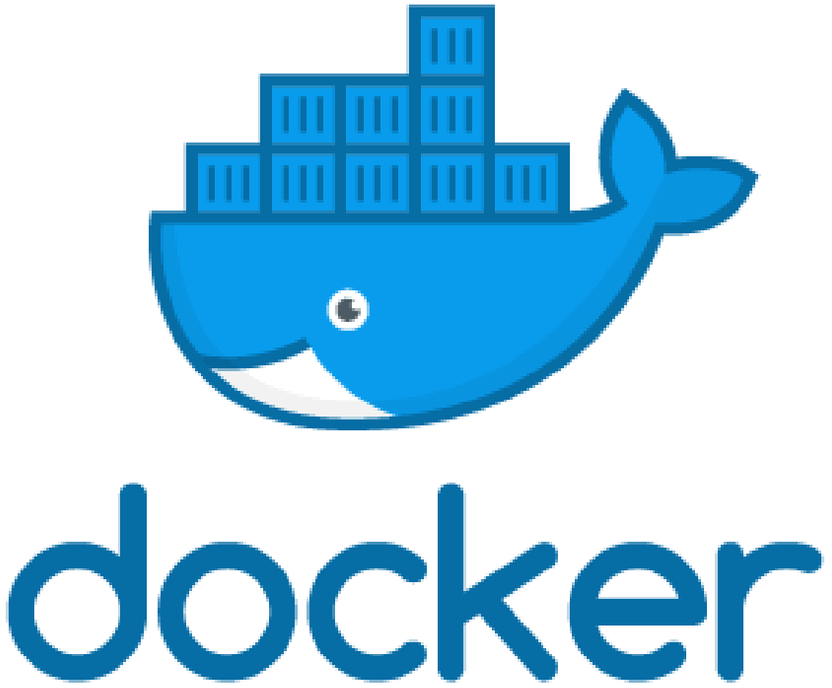


What is a container?

- A method of operation system virtualization
- A way to wrap an application into its own isolated box
- Includes only the binaries needed to support the application
- Isolates an app with its own view of the host from the perspectives of memory, CPU and network



What main options are available for containers?



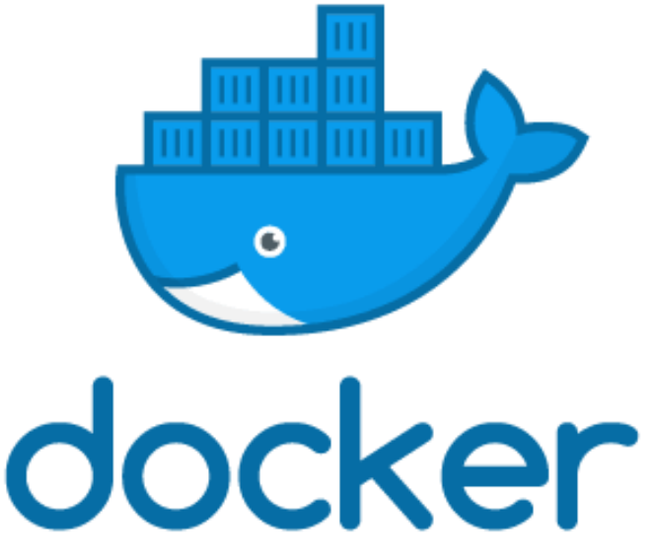
Microsoft

Introduction to Docker Containers

Docker Fundamentals

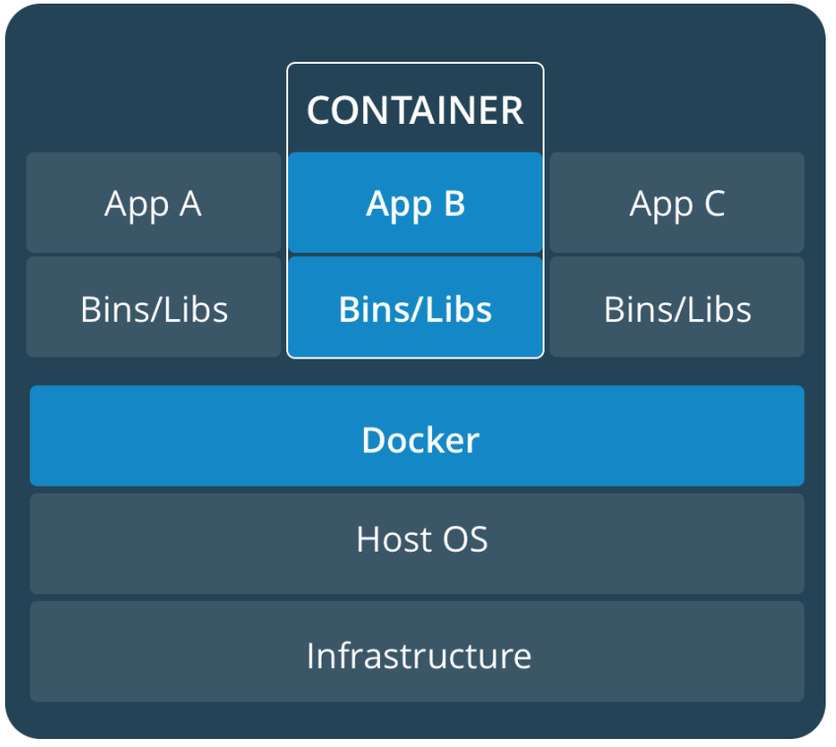
Microsoft Services

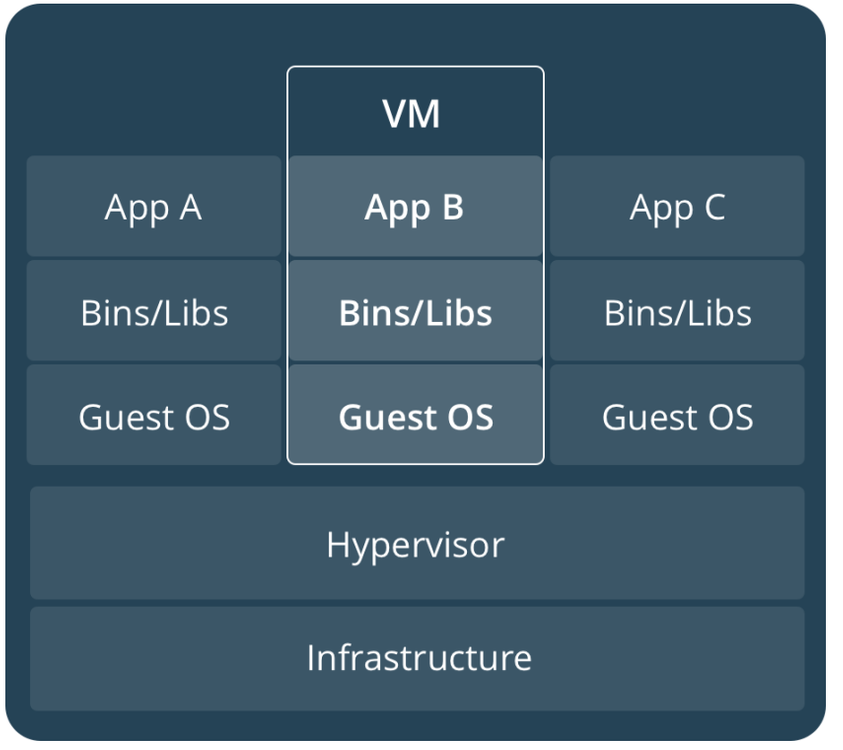
What is Docker?



- Open-source software to build and manage containers.
- Docker separates the application from the infrastructure using container technology
- “Dockerized” apps can run anywhere on anything
- No more dependency daemons so developers and system admins unite

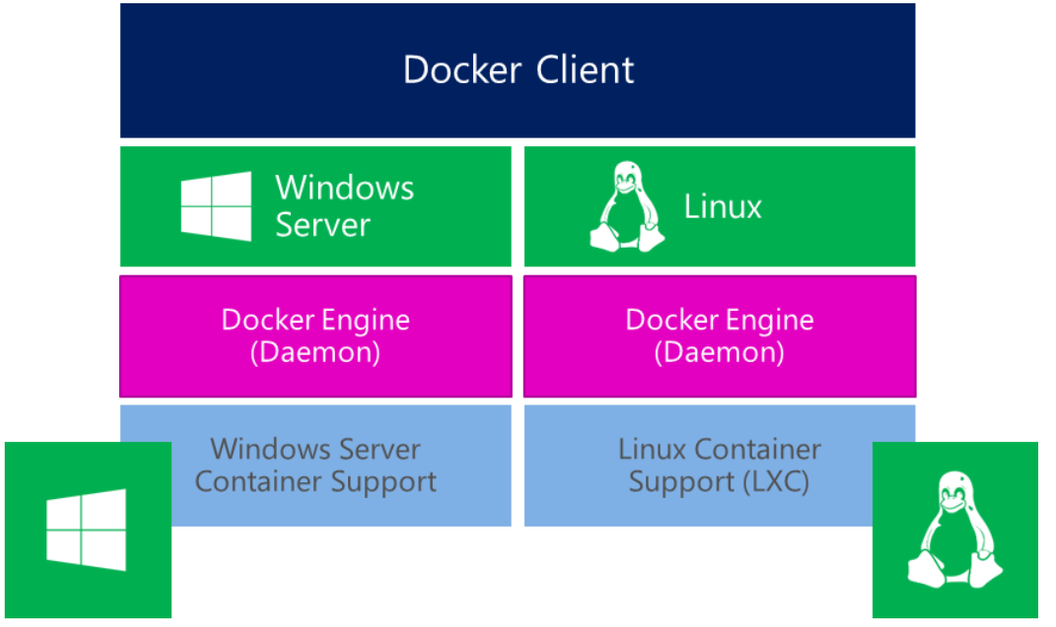
Docker Containers vs Virtual Machines





Manage Docker

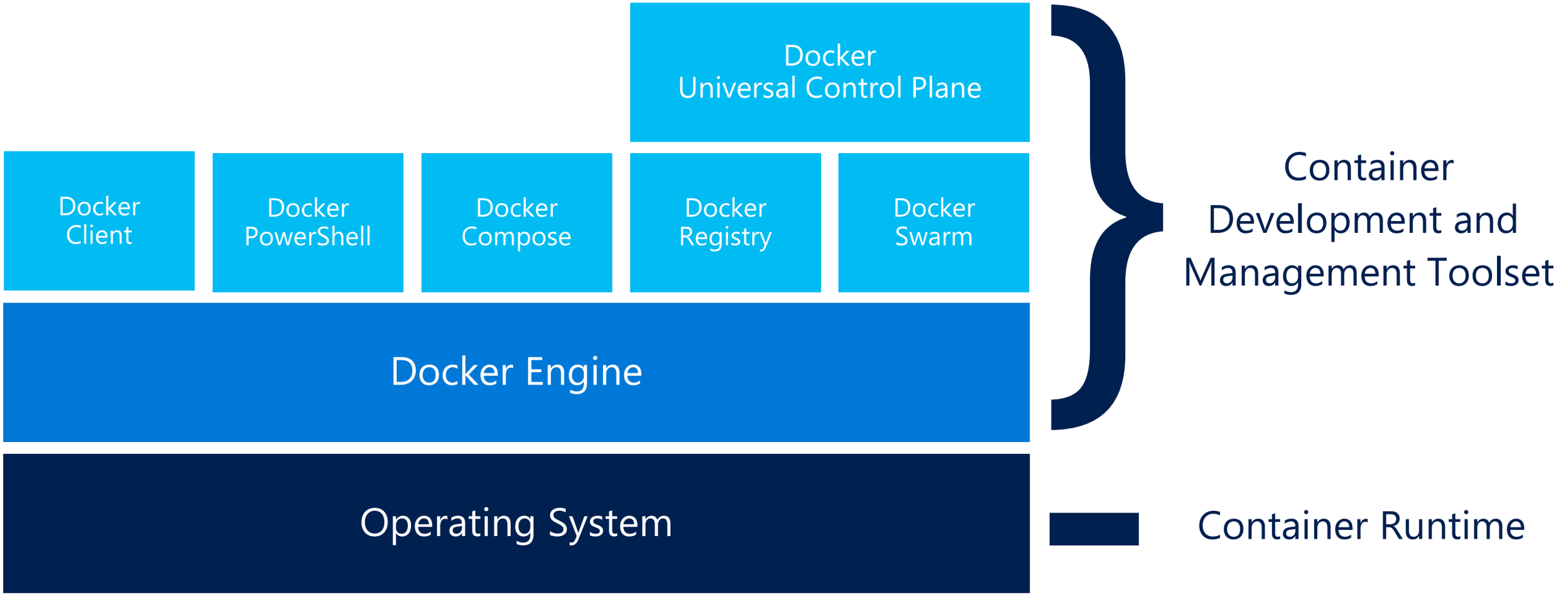
- Docker has been available for Linux Containers since 2008
- Open source engine that automates deployment of portable, self-sufficient apps
- Microsoft partnered with Docker to add support for Windows Containers in Windows Server 2016
- Docker experience maintained across both Linux and Windows platforms



Docker Terminology

Host	A VM running the Docker daemon to host a collection of Docker containers
Client	Where Docker commands are executed
Daemon	The background service running on the host that manages building, running and distributing 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 service that provides access to repositories, either through Docker Hub or Azure Container Registry
Azure Container Registry	A public resource for working with Docker images and its components in Azure

High Level Architecture



Docker Engine

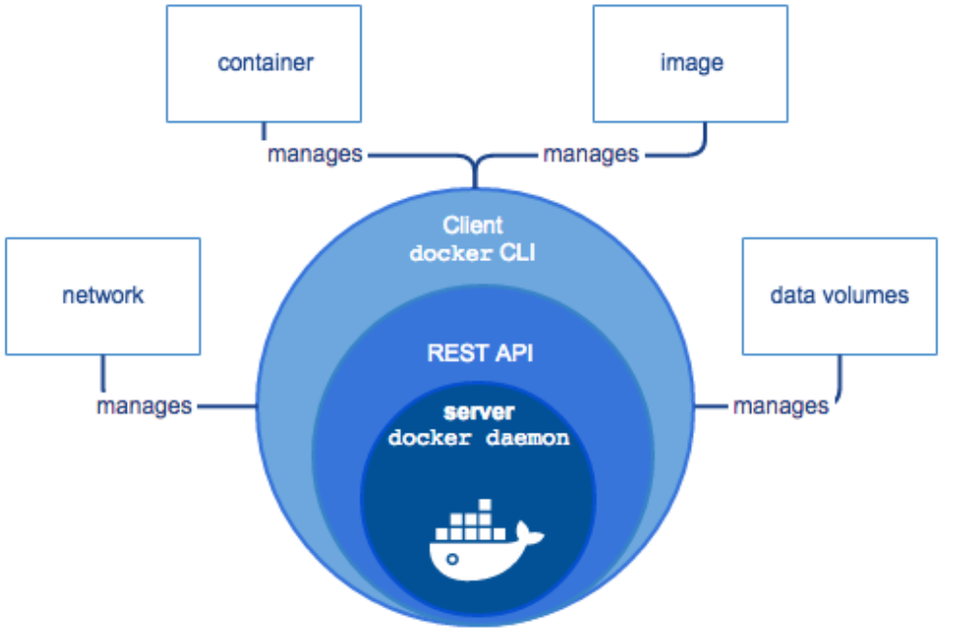
Client-Server application used for managing containers

Docker Daemon: runs on a container host

- Builds, runs and distributes containers
- Client accesses via Sockets through REST API

Docker Client: interface to docker daemon

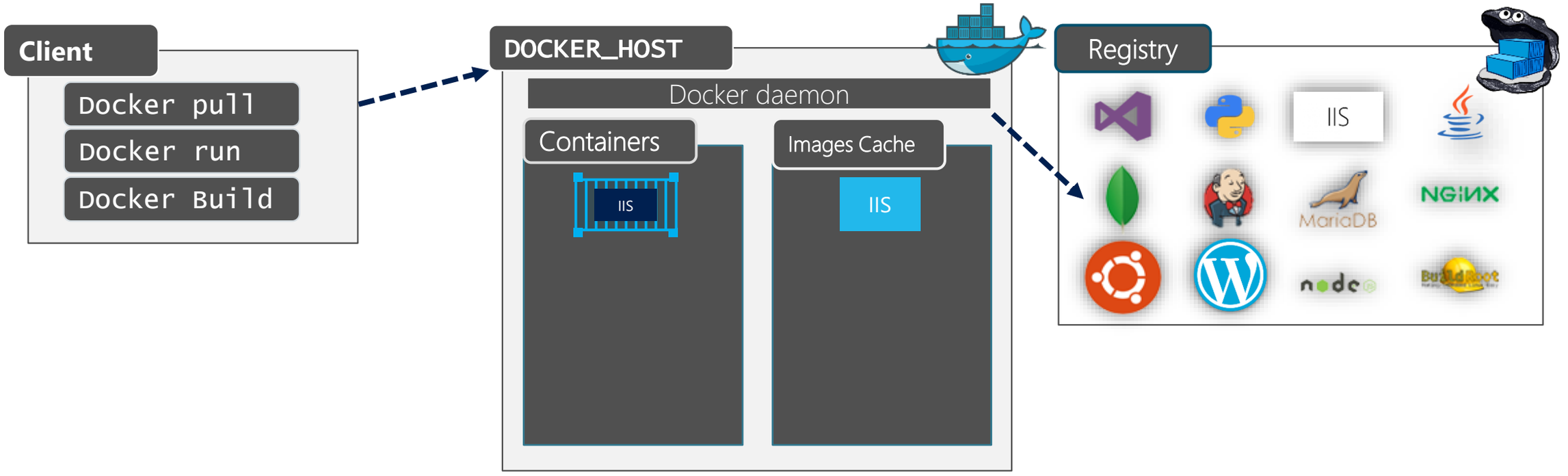
- Uses command line interface (i.e. docker.exe)
- Client can connect to a local or remote docker daemon



Basic Docker Commands

- docker run -> Runs a command in new container
- docker start -> Start one or more stopped containers
- docker stop -> Stop one or more running containers
- docker build -> Build an image from a Dockerfile
- docker pull -> Pull an image or a repository from a registry
- docker push -> Push an image or a repository to a registry
- docker version -> Show the Docker version information.
- docker search -> Search the Docker Hub for images
- docker images -> List images
- docker ps -> List Docker containers.
- docker rm -> Remove one or more containers
- docker rmi -> Remove one or more images

Docker in Action



Demonstration: Running Docker Containers

Running your First Container

Working with Docker Command Line Interface (CLI)

Microsoft

Introduction to Docker Containers

Docker Image

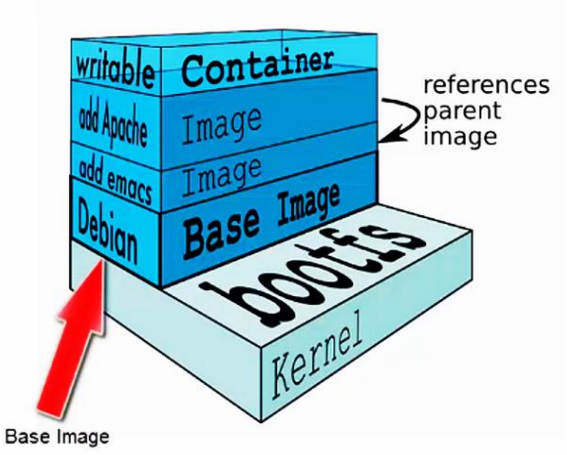
Microsoft Services

Docker Image

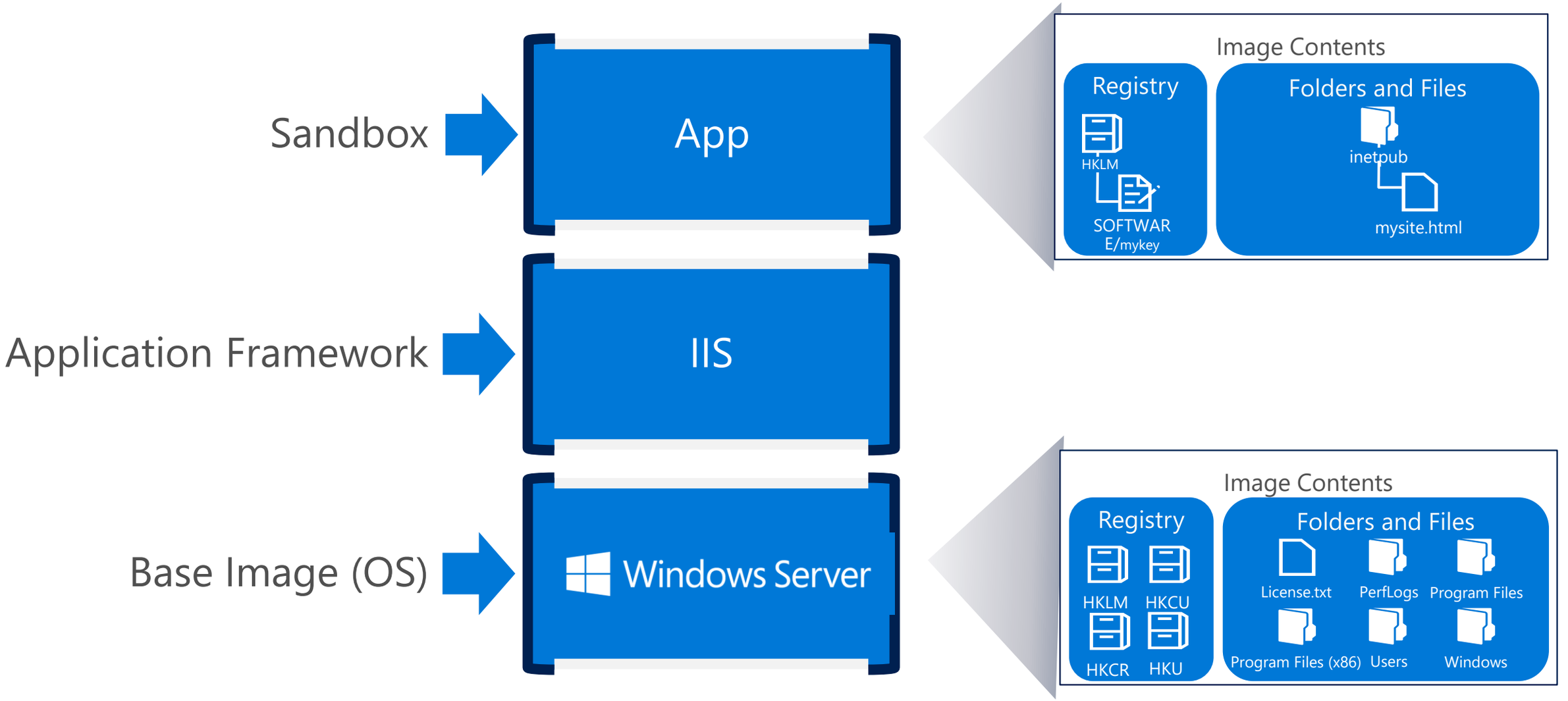
A container image is a read-only template from which containers are created.

A container image:

- Starts from a base Operating System image
- Has additional layers that include application binaries
- Can automatically be built using a *Dockerfile* text file
- Are distributed from a Registry and stored locally



Docker Container Image layers



Building images

- Docker Build and Dockerfiles
  - Method for automated container image build
  - Consumed when running “docker build”
  - Caches unchanged commands
  - Integrates into Docker Hub or Azure Container Registry

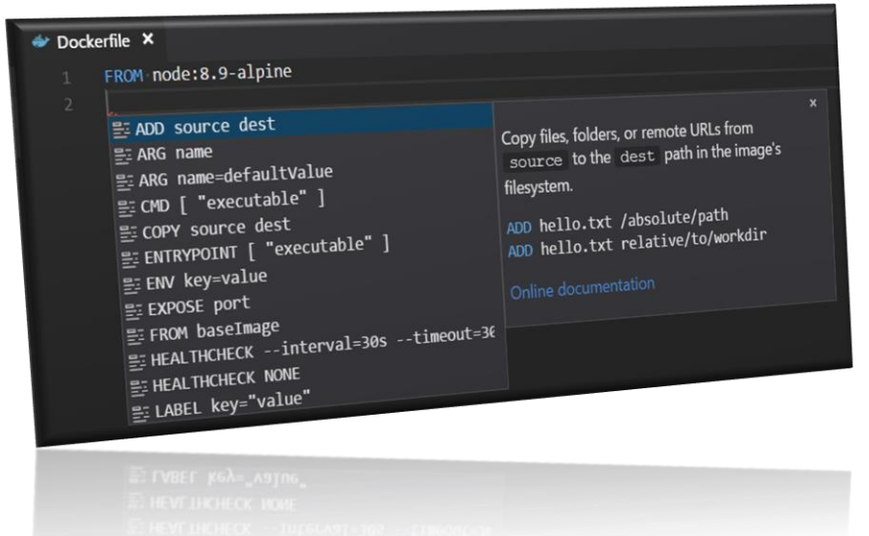
Examples

IIS Image

```
FROM windowsservercore
RUN powershell –command Add-WindowsFeature  Web-Server
```

WebApp Image

```
FROM iis
ADD mysite.htm inetpub\mysite.htm
```

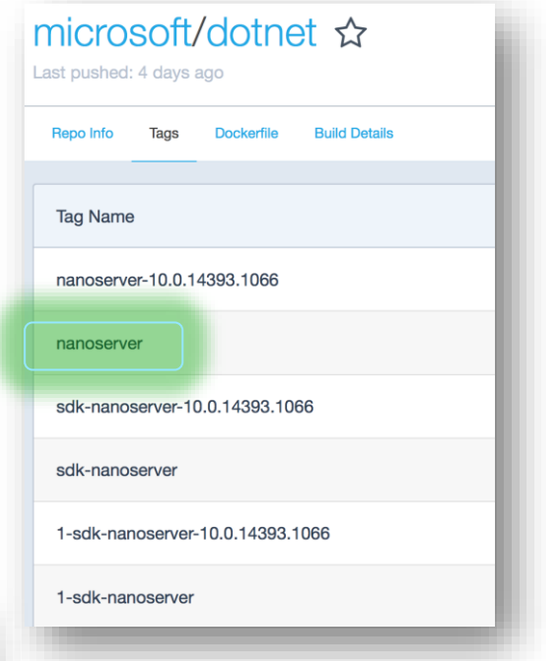


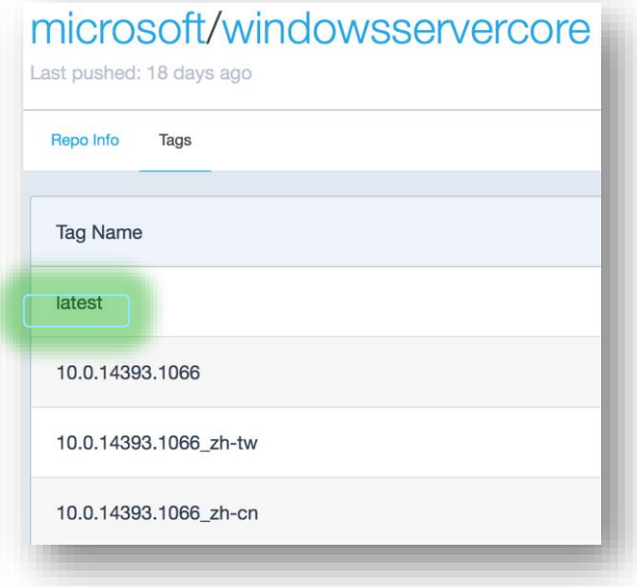
Common Dockerfile Instructions

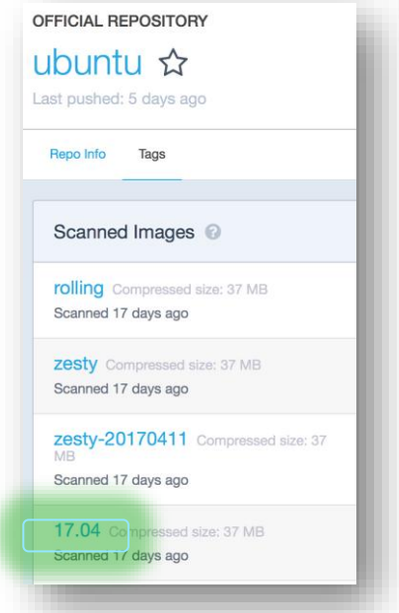
- FROM instruction initializes a new build stage and sets the Base Image for subsequent instructions.
- LABEL is a key-value pair, stored as a string. You can specify multiple labels for an object, but each key-value pair must be unique within an object.
- RUN will execute any commands in a new layer 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).

Images Tags

- A tag name is a string value that you can use to distinguish versions of your Docker images allowing you to 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)







Demonstration: Building your image

Building Custom Container Images with dockerfile

Interaction with a Running Container

Microsoft

Introduction to Docker Containers

Image Registries

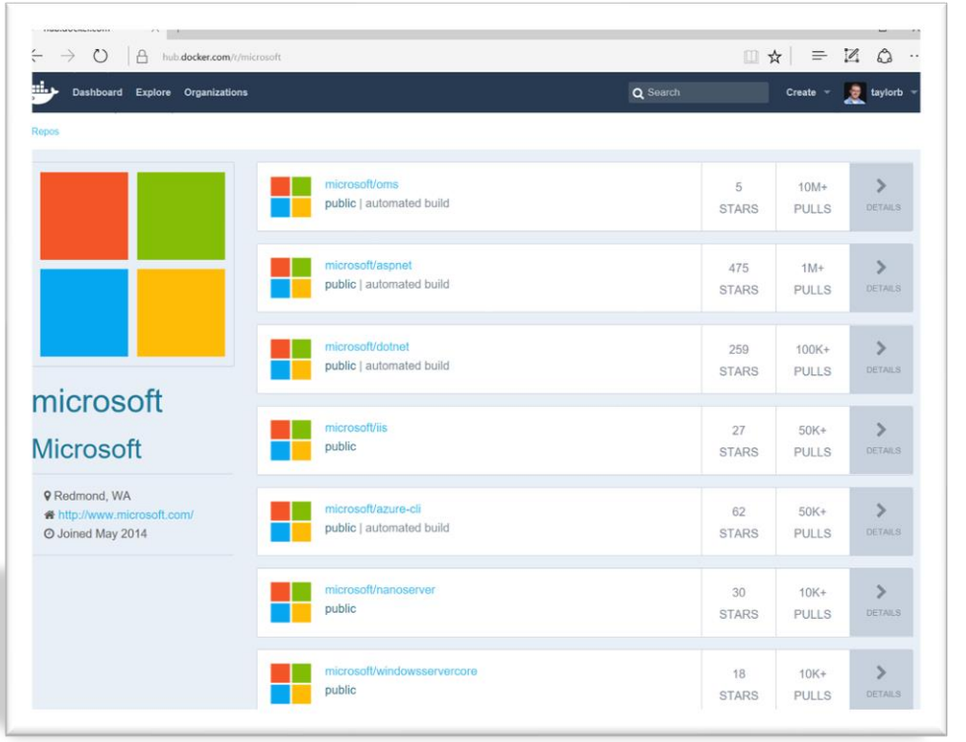
Microsoft Services

Image Registries

- What is a registry?
  - Registries are a storage system for container images
    - Images are **Pushed** into a registry
    - Images are **Pulled** from a registry
    - Images are **Searched** for within a registry

Image Registries

- Docker Hub and Docker Store
  - Public, official and private image repositories
  - Granular access controls with organization support
  - Automated image build support
- Docker Trusted Registry (DTR)
  - Enterprise grade private registries
  - Runs on your infrastructure (on-prem or cloud)
  - Active Directory and Role Based Access Controls
- Azure Container Registry (ACR)
  - Store and manage container images across Azure deployments
  - Maintain Windows and Linux container images
  - Same API and Tools as Docker Hub/Store/Registry
- Docker Registry
  - Open source foundation of Hub and Docker Trusted Registry
  - Runs your infrastructure (on-prem or cloud) as a container
  - https://docs.docker.com/registry and/or https://github.com/docker/distribution



Microsoft

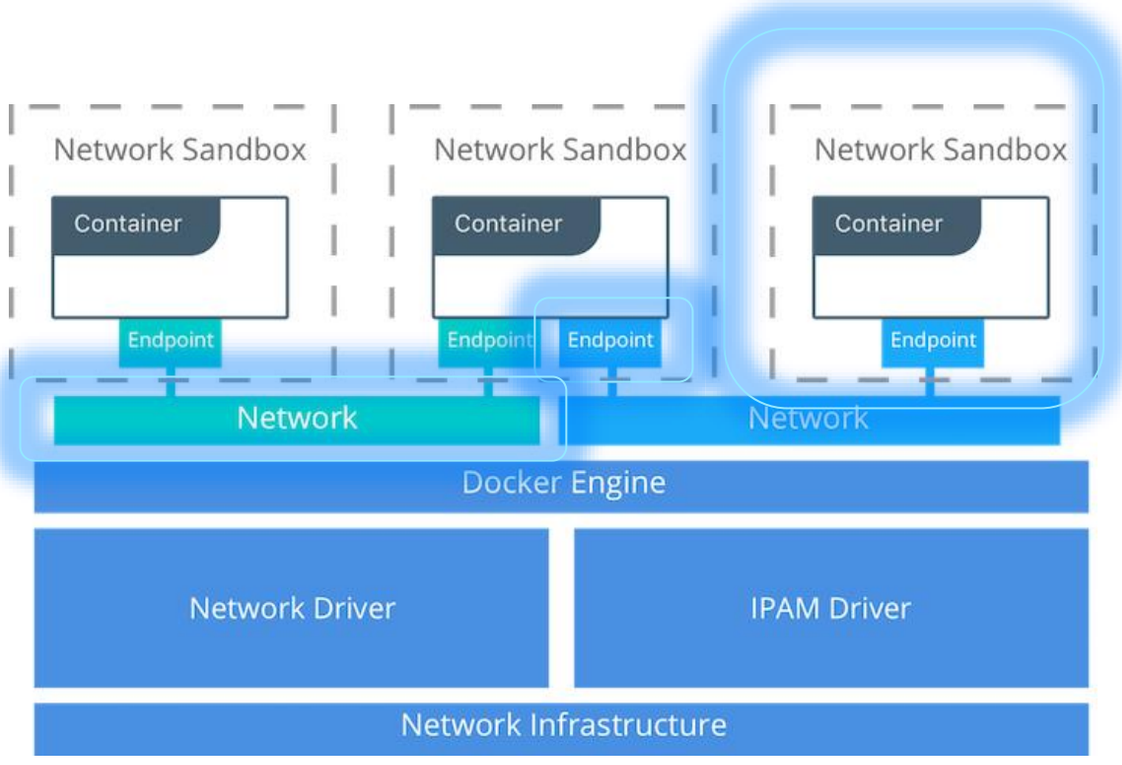
Introduction to Docker Containers

Docker Network

Microsoft Services

Docker Network Overview

- Docker networking is based on a pluggable networking model called Container Networking Model (CNM)
- Each Docker network can only have a single interface per container
- CNM is divided in three blocks:
  - Sandbox
  - Endpoint
  - Network



Docker Network Overview

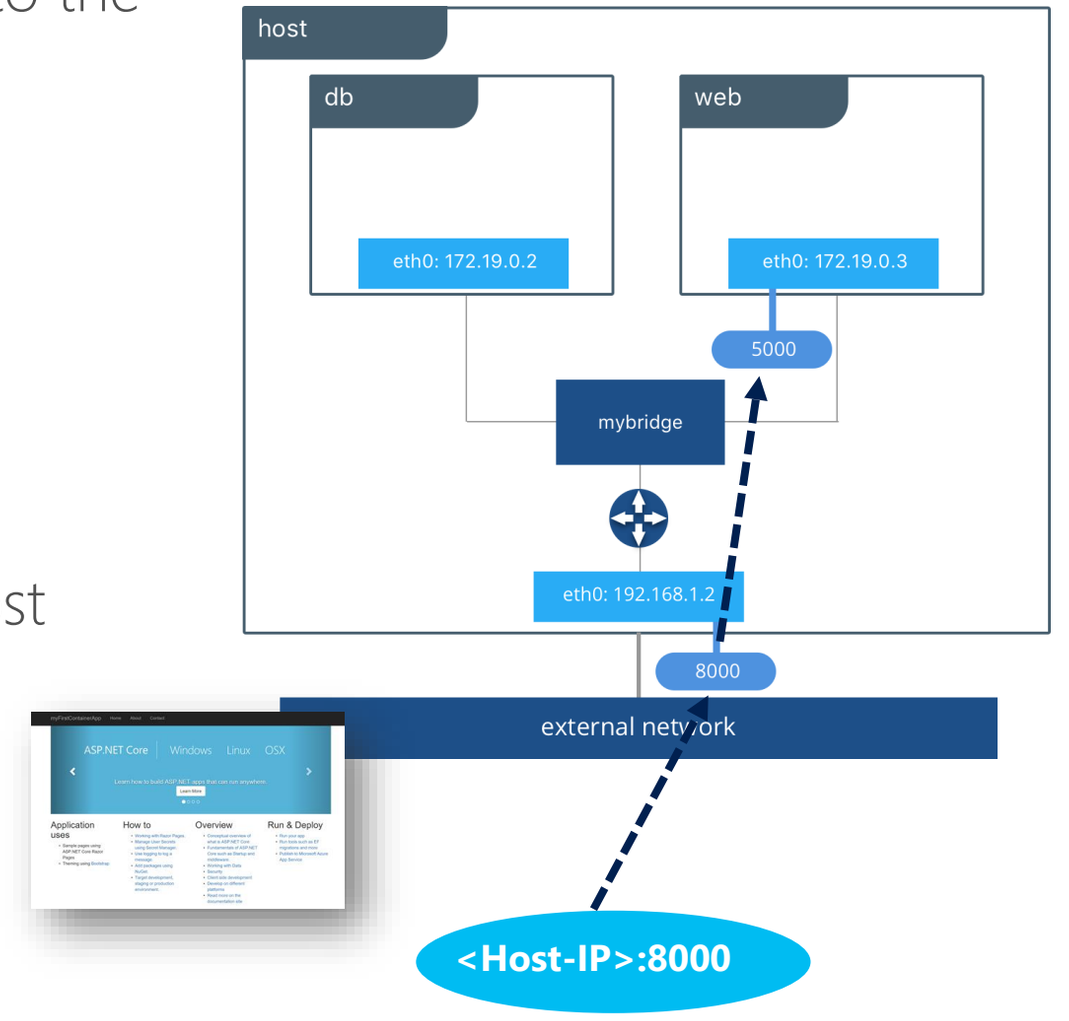
- The Docker engine creates several networks using different drivers when it starts the first time.
- Each driver has different benefits and limitations.

```
C:\>docker network ls

NETWORK ID          NAME                DRIVER              SCOPE
9c783e1b2ed6        bridge              bridge              local
5009517d1247        host                host                local
bea80cb94c27        none                null                local
Vlujsum8myOu        ingress             overlay             swarm
ed60df3f6402        macvlan             macvlan             local
```

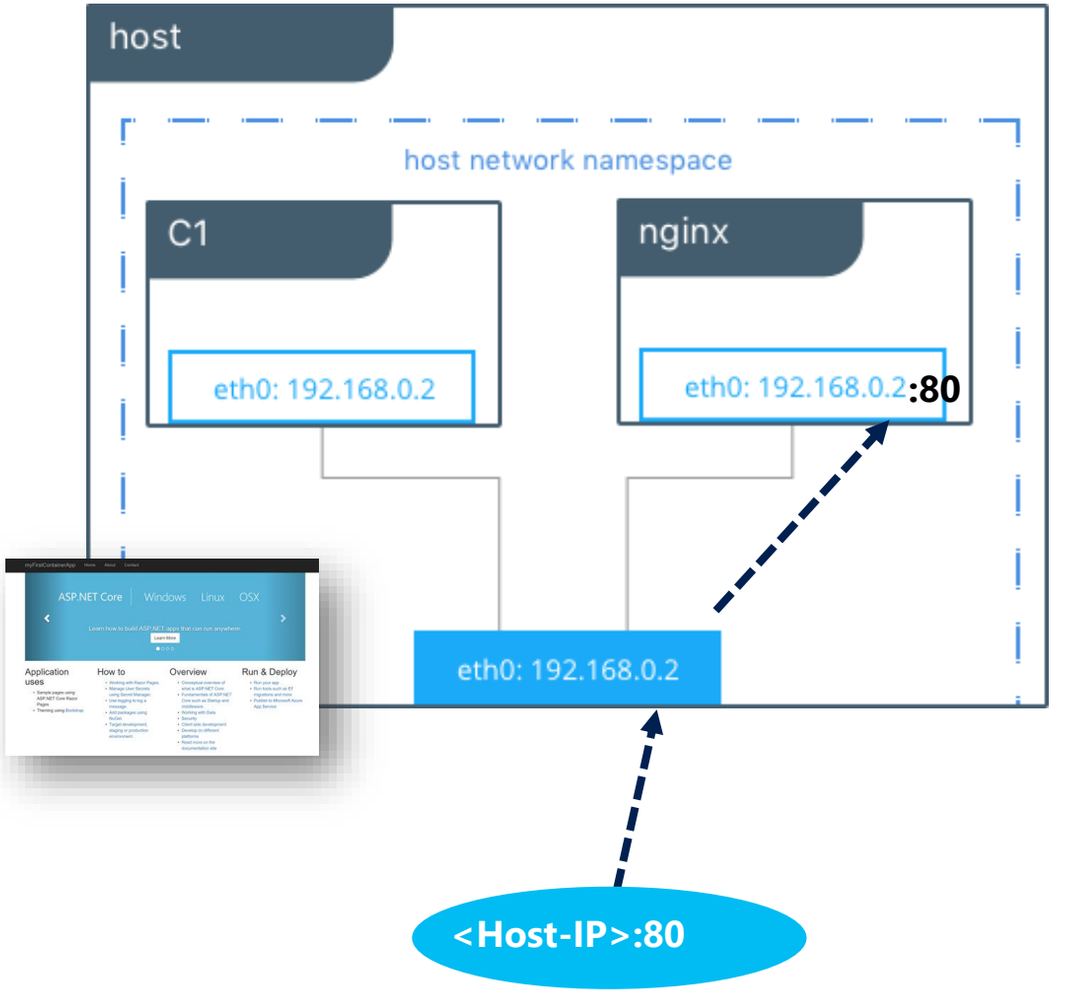
Bridge Mode

- Containers are started and connected by default to the internal bridge network
- The bridge driver is a local scope driver
- External access is granted by exposing ports to containers.
- The –p flag will expose an external port on the host and map it to a port on the container



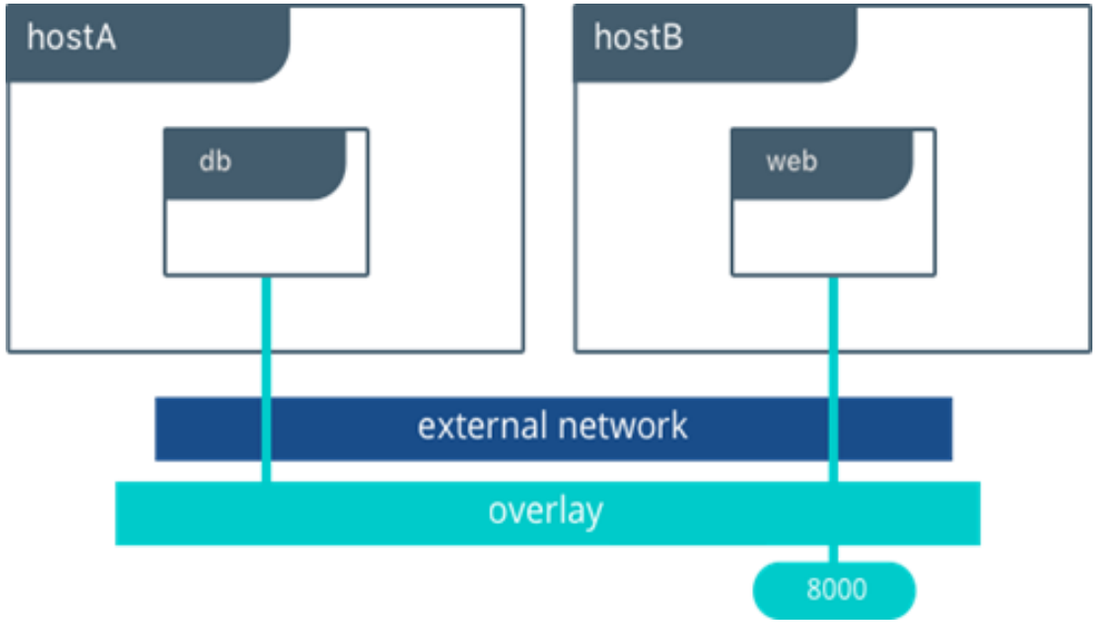
Host Mode

- The host driver will start the container in the same namespace as the host itself, allowing a container to use the host networking stack directly
- Better performance, but can result in port conflicts
- In host mode, a container uses the host IP address



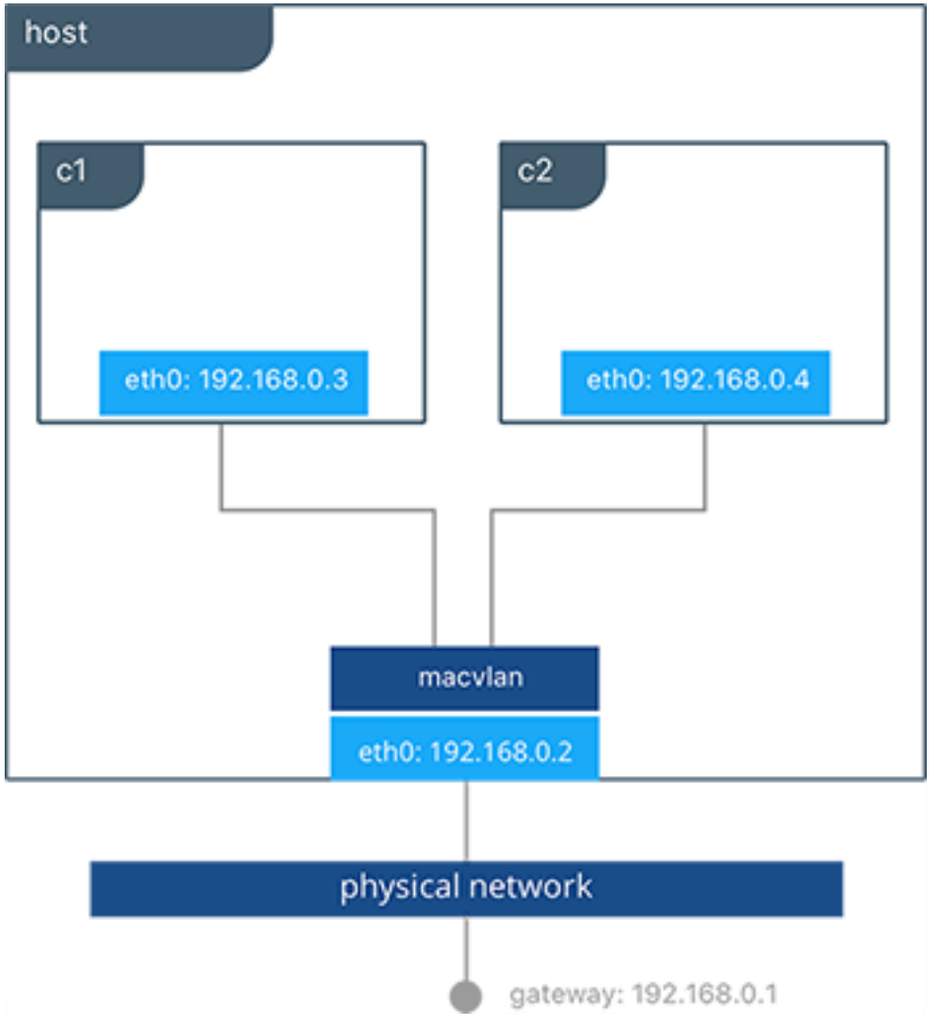
Overlay Network

- The overlay network makes use of VXLAN in order to create an overlay network over the underlying network
- The tunnel allows containers across hosts to communicate
- Each container gets a pair of IP addresses
- The overlay mode is used by Docker Swarm orchestrator



Macvlan Network

- Macvlan mode provides a MAC Address for each container
- Allows containers to become part of the traditional network
- When starting a container, you can apply a physical IP address on that network



Microsoft

Introduction to Docker Containers

Docker Storage

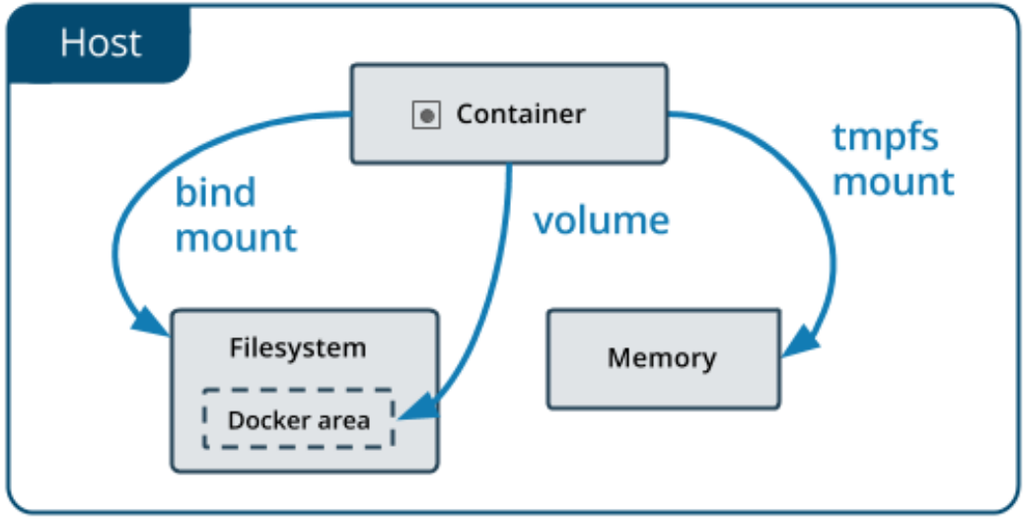
Microsoft Services

Docker Storage - Overview

- Docker Containers are immutable by design
- To save data, you must provide external storage
- Two options for data:
  - Persistent
  - Non-persistent → Every container automatically gets local storage

Docker Storage – Persistent Data

- **Volumes** are stored in an area of the host filesystem that is managed by Docker
- **Bind** mounts can map to any folder in the host filesystem
- **tmpfs** mounts are like virtual folders that only exist in the host's memory and are never written to the filesystem
- **Remote Storage**: Azure Storage or relational databases: Azure SQL or Cosmo DB



Lab: Introduction to Docker Containers

Running Your First Container

Working with Docker Command Line Interface (CLI)

Building Custom Container Images with Dockerfile

Knowledge Check

1. What are the components of Docker architecture?
2. What is a Dockerfile?
3. What platforms does Docker run on?