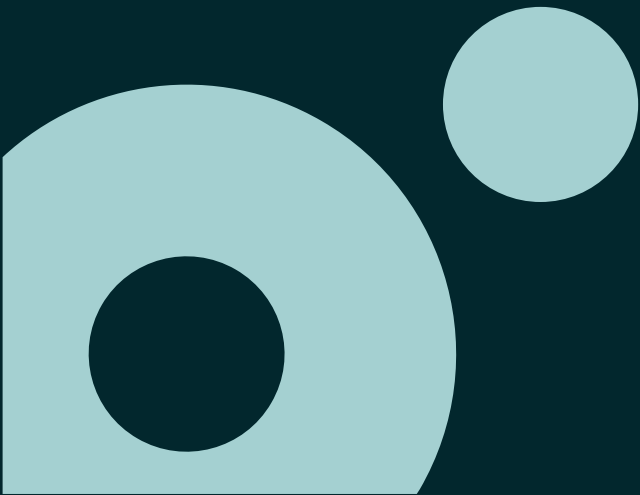


moOngy.  
Minds on the move

# Containers & Kubernetes

Session #01





Containers: Introduction

Images: Introduction

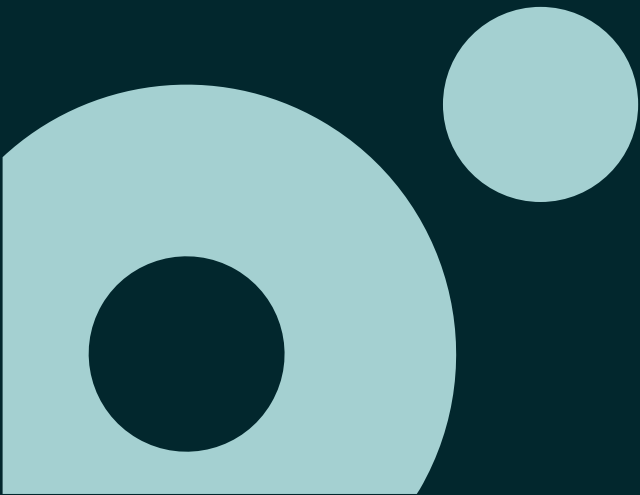
Registry: Introduction

Container Lifecycle

Linux vs Windows Containers

Lab

# Containers: Introduction



# What is a container?

## Containers: Introduction

A method of operation system virtualization

A way to wrap an application into its own isolated box

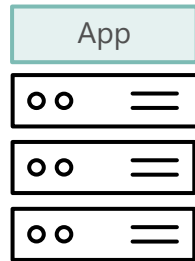
Includes only the binaries needed to support the application

Isolates an app with its own view of the host from the perspectives of memory, CPU and network



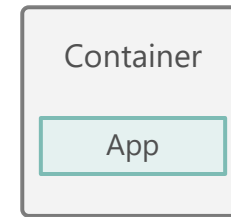
# VM vs Containers

## Containers: Introduction



### Virtual machines

Virtualize the hardware  
VMs as units of scaling



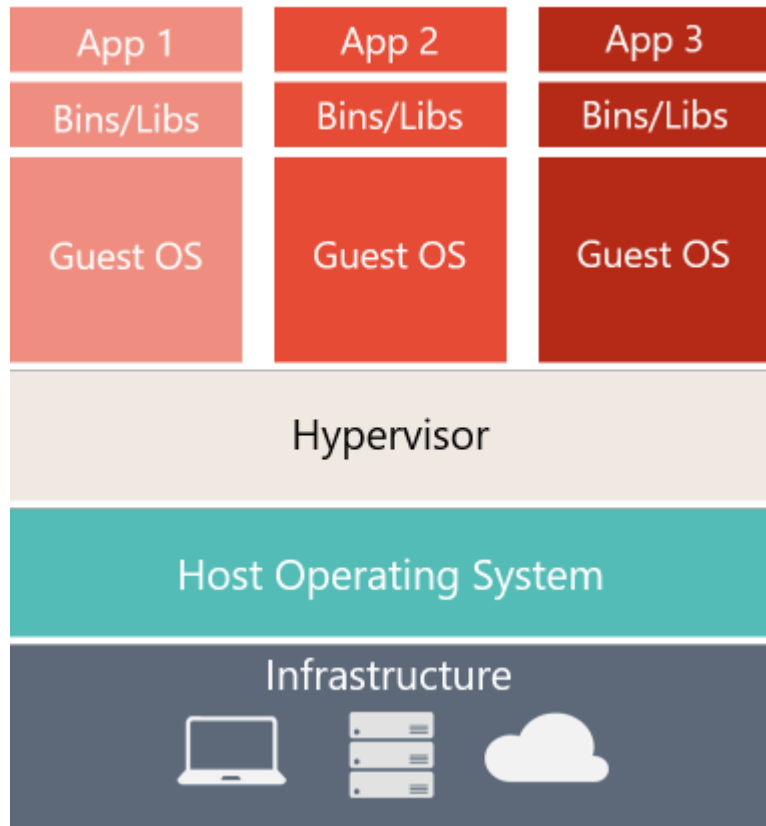
### Containers

Virtualize the operating system  
Applications as units of scaling

# VM vs Containers

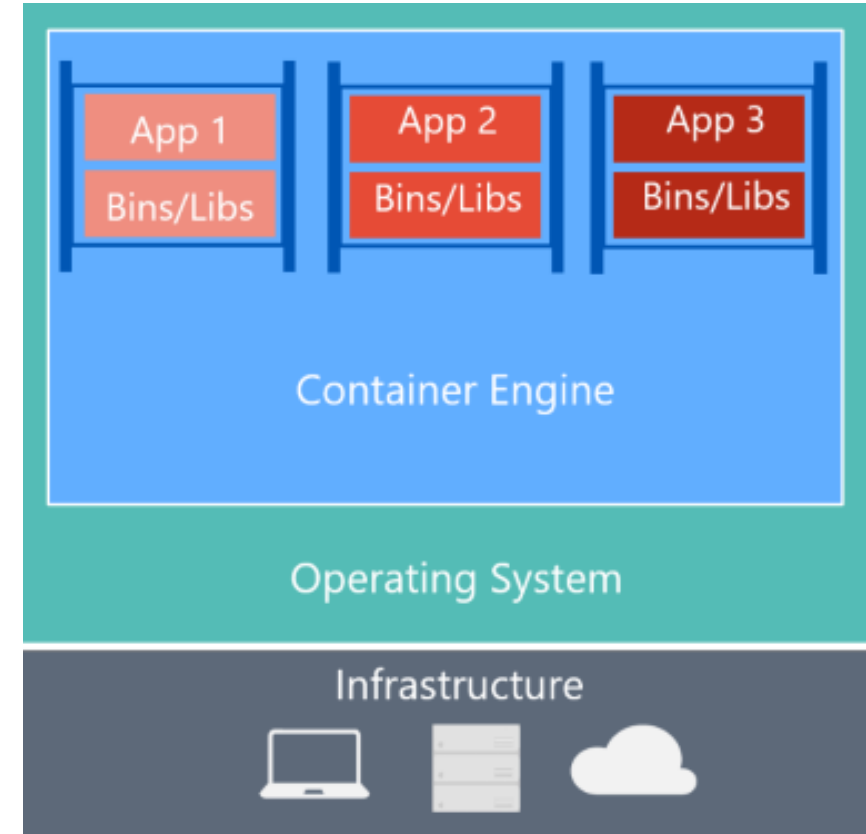
## Containers: Introduction

### Virtual Machine



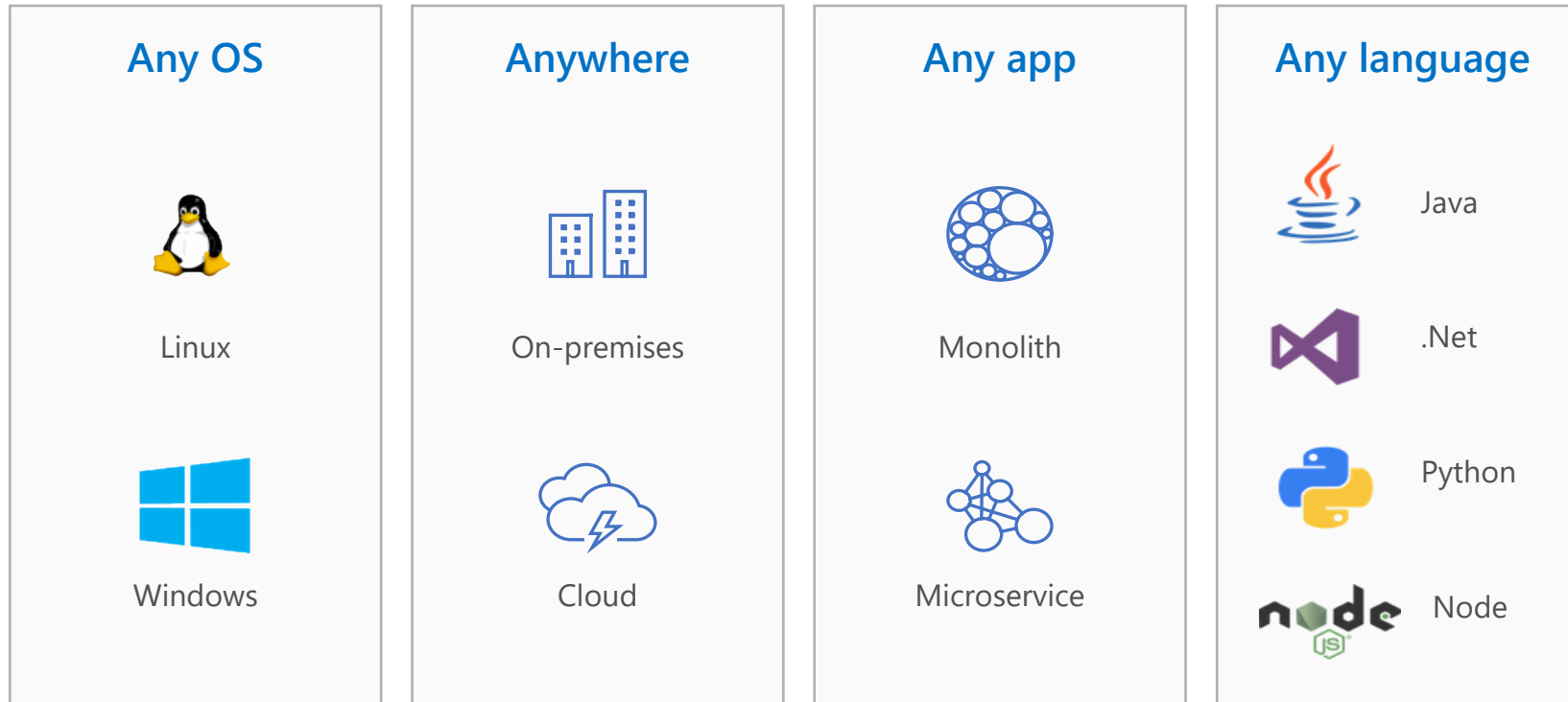
MoOngy 2021

### Container



# Benefits of using containers

## Containers: Introduction





# Benefits of using containers

## Containers: Introduction



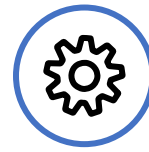
Agility

Ship apps  
faster



Portability

Easily move  
workloads



Density

Achieve  
resource  
efficiency

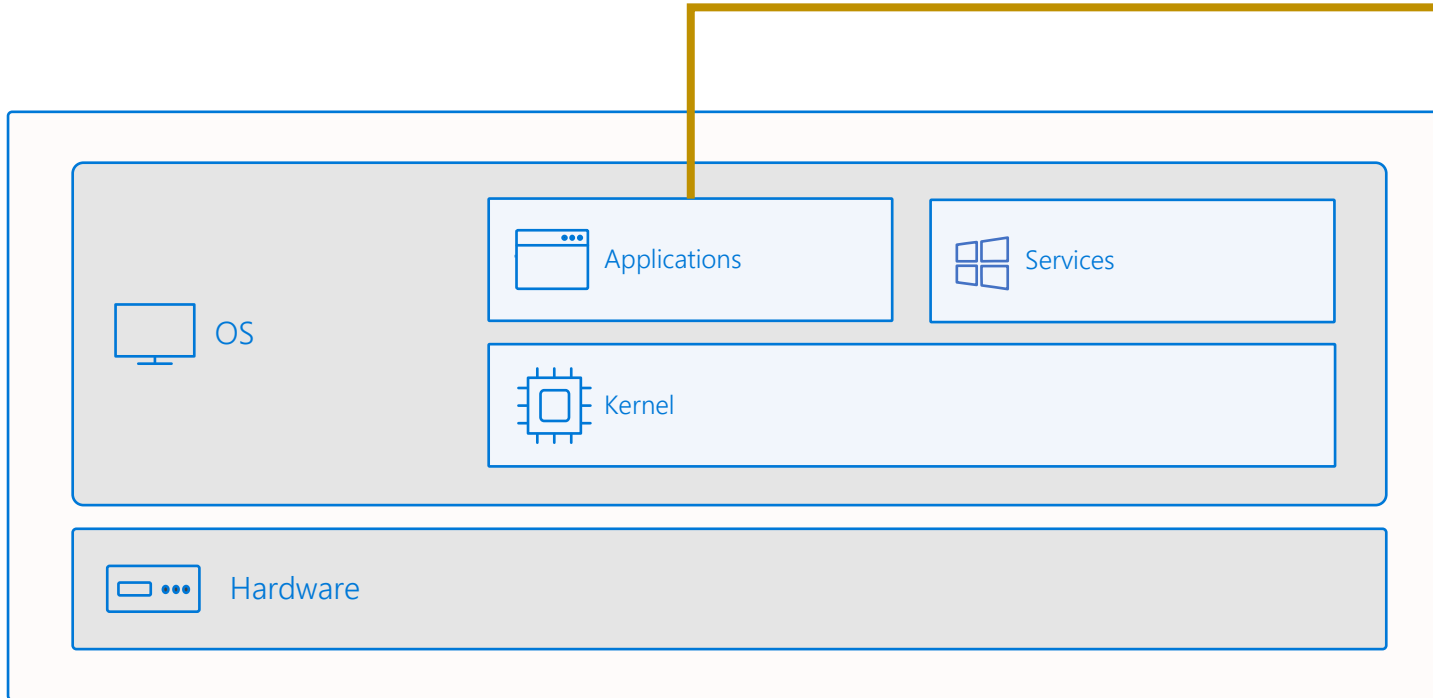


Rapid scale

Scale easily  
to meet  
demand

# Architecture: Legacy Apps

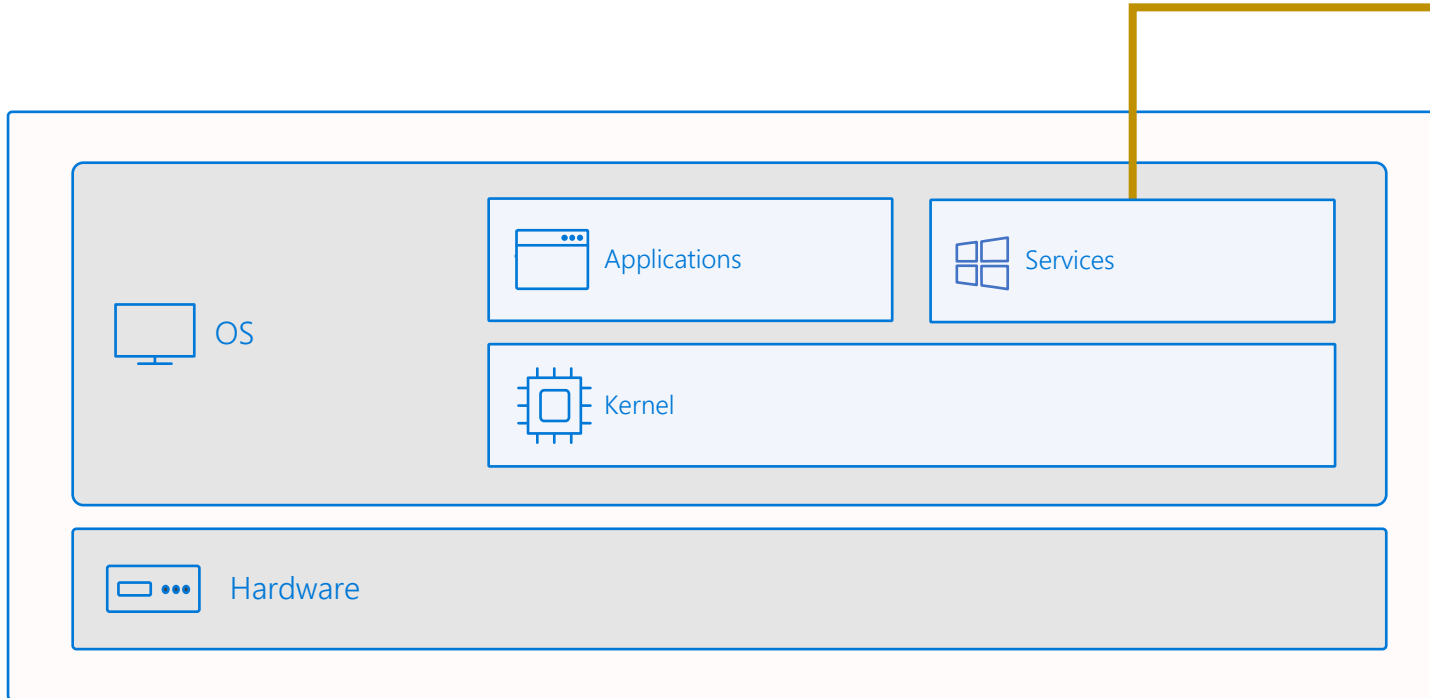
## Containers: Introduction



- Your application, could be written in a variety of languages:
  - Python
  - NodeJS
  - .NET full
  - .NET core

# Architecture: Legacy Apps

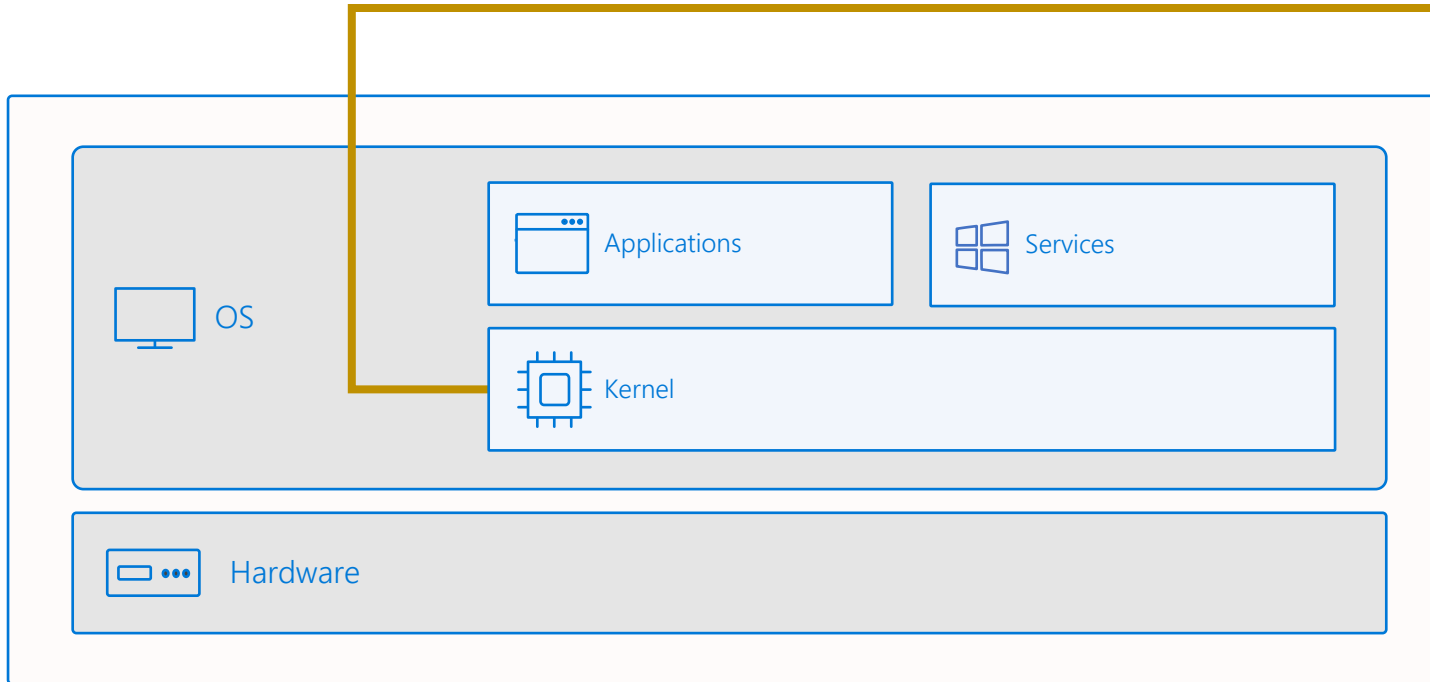
## Containers: Introduction



- Long-running processes
- Can provide additional functionality to your app
- Can assist with interacting with the kernel

# Architecture: Legacy Apps

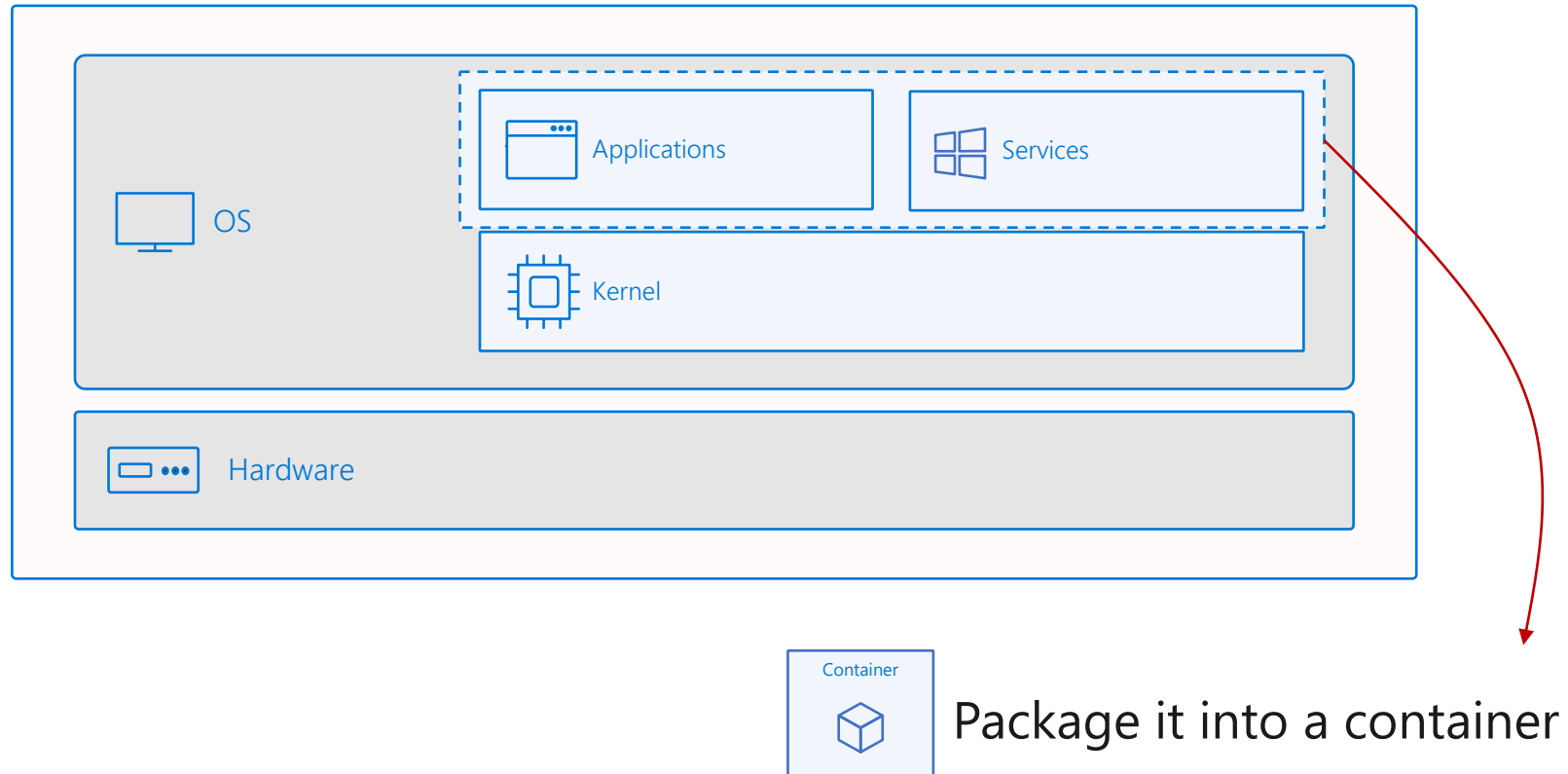
## Containers: Introduction



- Provides low-level operations for your app.
- Talks to the hardware on behalf of your app
- Manages system resources
- Drivers

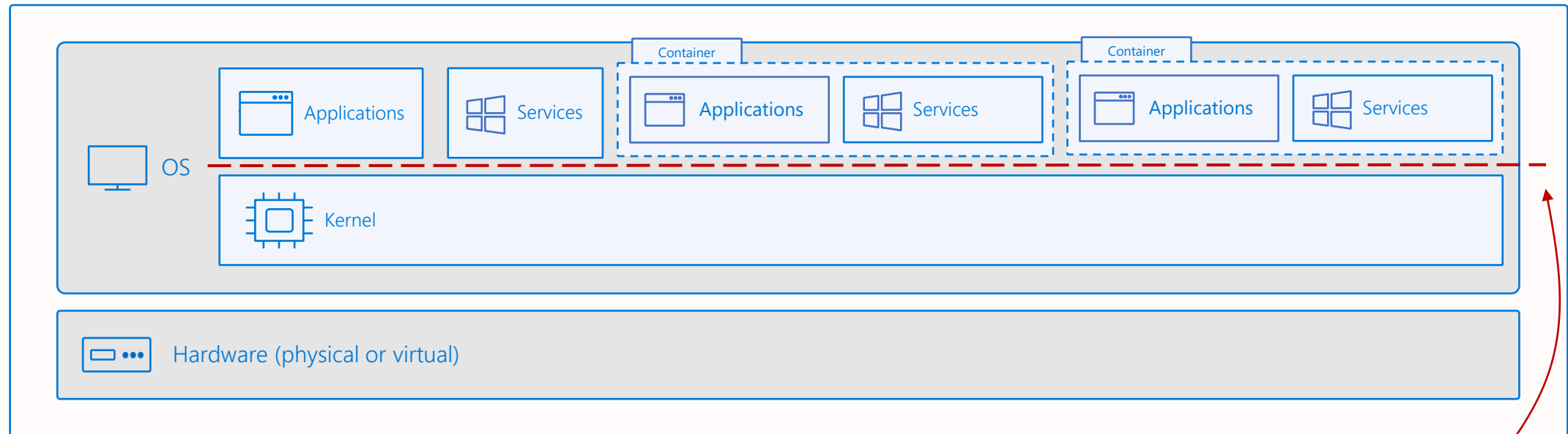
# Architecture: Legacy Apps → Containers

## Containers: Introduction



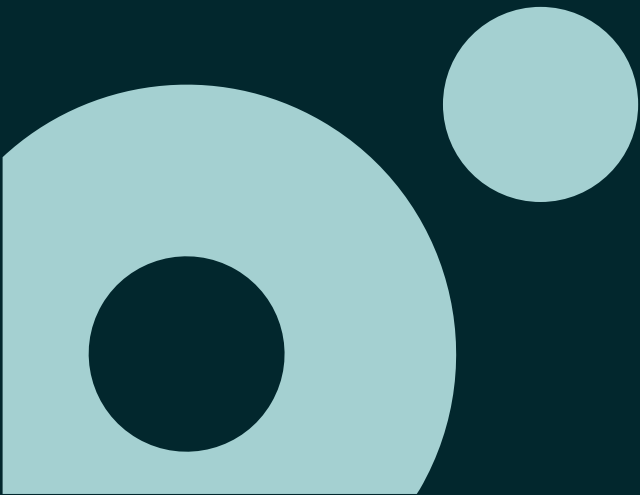
# Architecture: Legacy Apps → Containers

## Containers: Introduction



This is our virtualization boundary.

# Images: Introduction



# What is a container image?

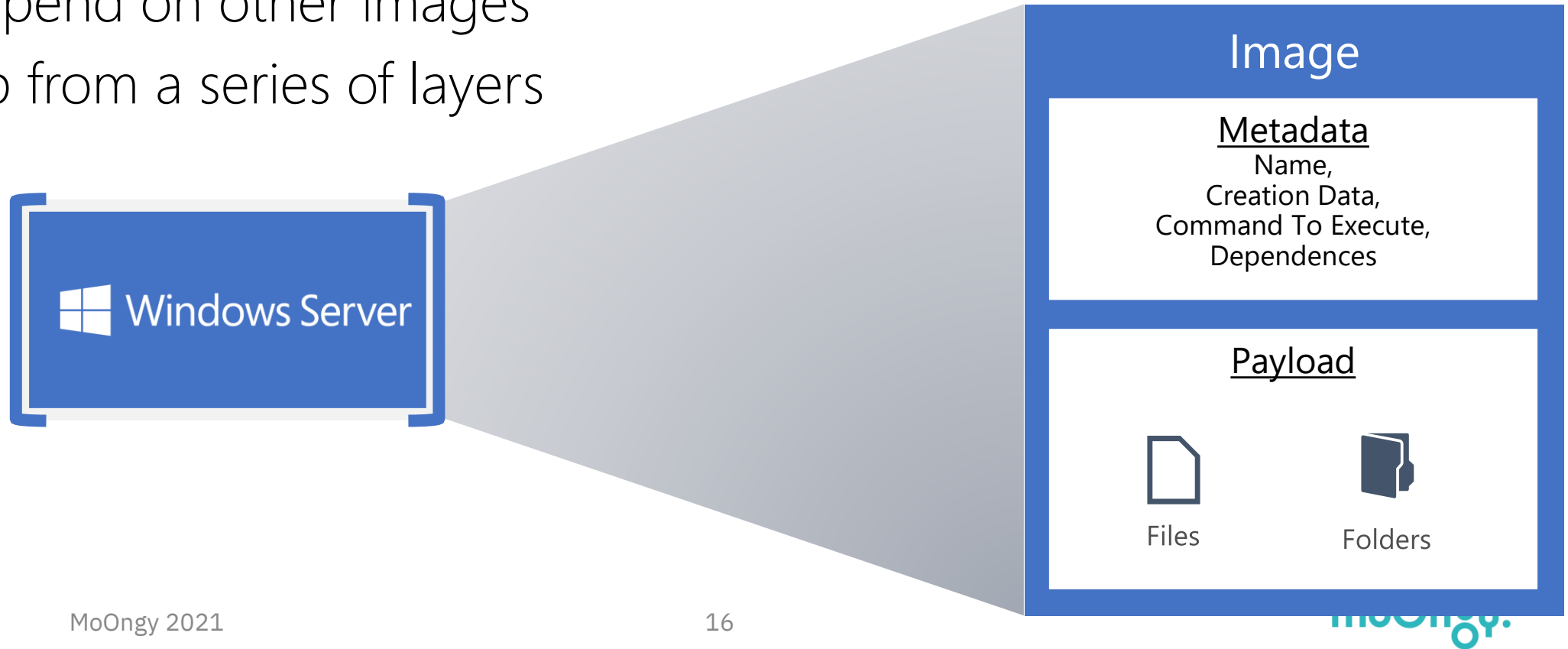
## Images: Introduction

Analogous to a VHD + Config for Virtual Machines

Read-only templates for containers

Can depend on other images

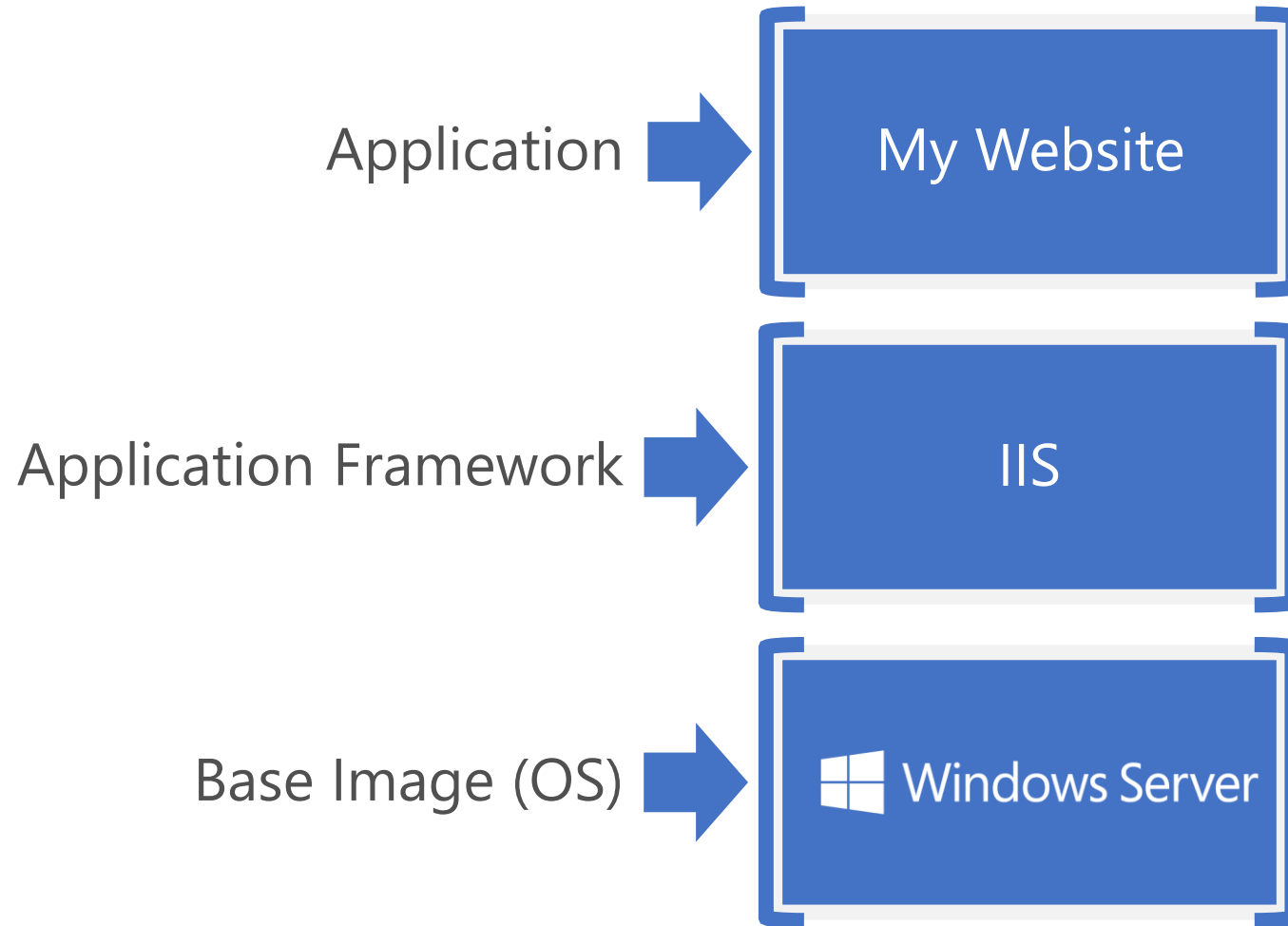
Built up from a series of layers





# What is a container image?

## Images: Introduction



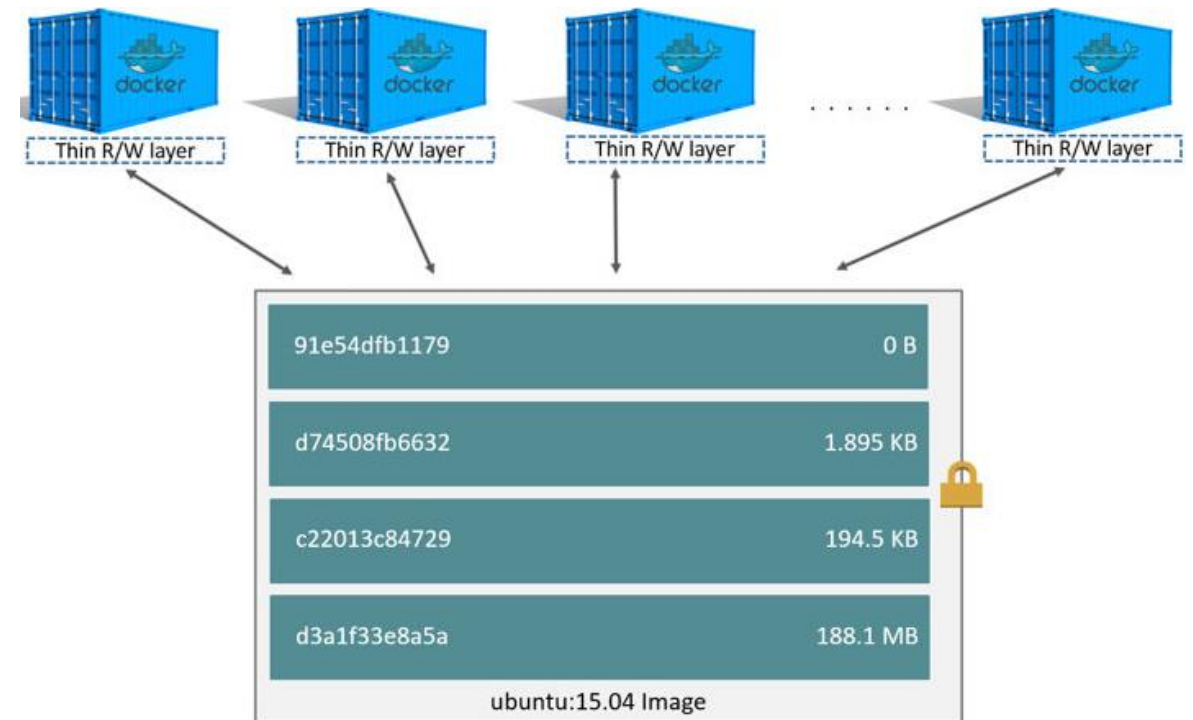
# What is a container image?

## Images: Introduction

Each container has its own writable container layer

All changes are stored in this container layer

Multiple containers can share access to the same underlying image but have their own data state



# How Image and Container relates?

## Images: Introduction

Image is a template for the container

Container is a running instance of the workload

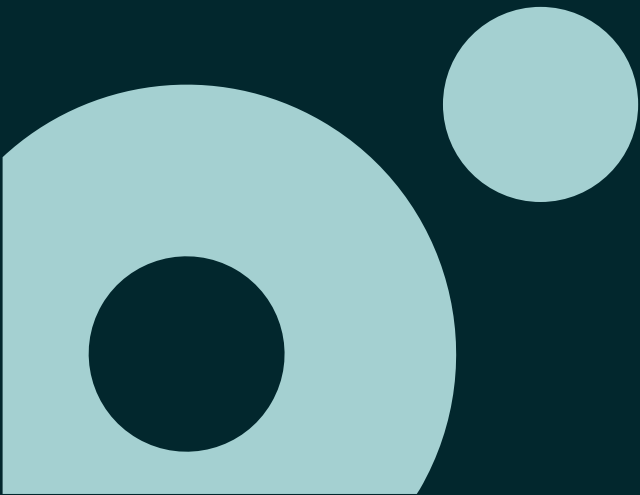
Making OOP comparison

- Image is a class

- Container is an instance of the class (i.e. an object)

Using one Image you can instantiate several containers

# Registry: Introduction



# What is a Registry?

## Registry: Introduction

Registry is a stateless, highly scalable server side application that stores and lets you distribute images



 Search		Explore	Help	Sign up	Sign in
Explore Official Repositories					
	nginx official	9.5K STARS	10M+ PULLS	> DETAILS	
	alpine official	4.2K STARS	10M+ PULLS	> DETAILS	
	busybox official	1.4K STARS	10M+ PULLS	> DETAILS	
	httpd official	2.0K STARS	10M+ PULLS	> DETAILS	
	redis official	5.7K STARS	10M+ PULLS	> DETAILS	

# How to use a Registry?

## Registry: Introduction

Tightly control where your images are being stored

Fully own your images distribution pipeline

Integrate image storage and distribution tightly into your in-house development workflow

Public registry and/or Private registry

# Public vs Private

## Registry: Introduction

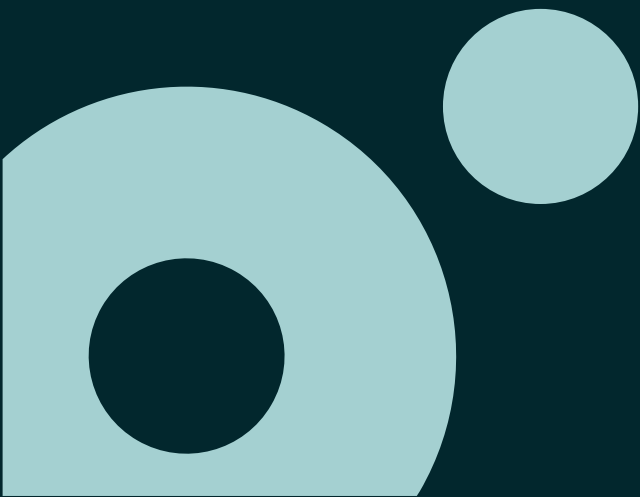
### Docker Hub and Docker Store

- Public, official and private image repositories
- Granular access controls with organization support
- Automated image build support

### 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
- Supports hosting private registry with fine grain Role Based Access Control for management
- Can be geo-redundant making it faster to download/upload images based on client location

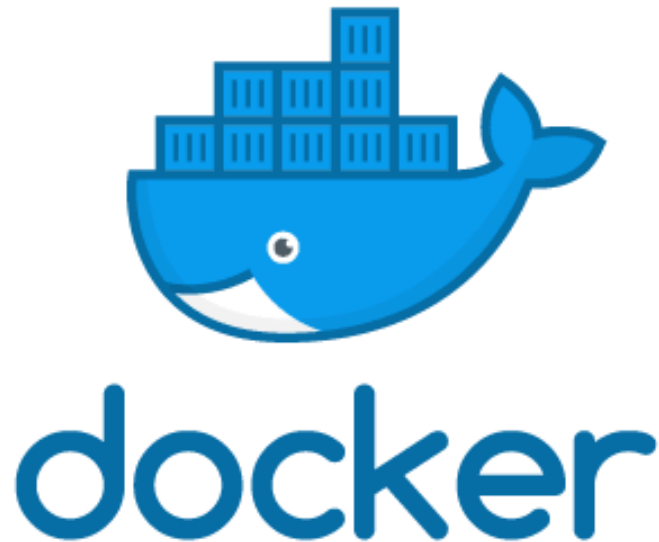
# Docker: Container Lifecycle





# What is Docker?

## Docker: Container Lifecycle



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

# What is Docker?

## Docker: Container Lifecycle



# Docker concepts

## Docker: Container Lifecycle

### 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 instantiating a container (more on it later)

### Container

A runtime instance of an image

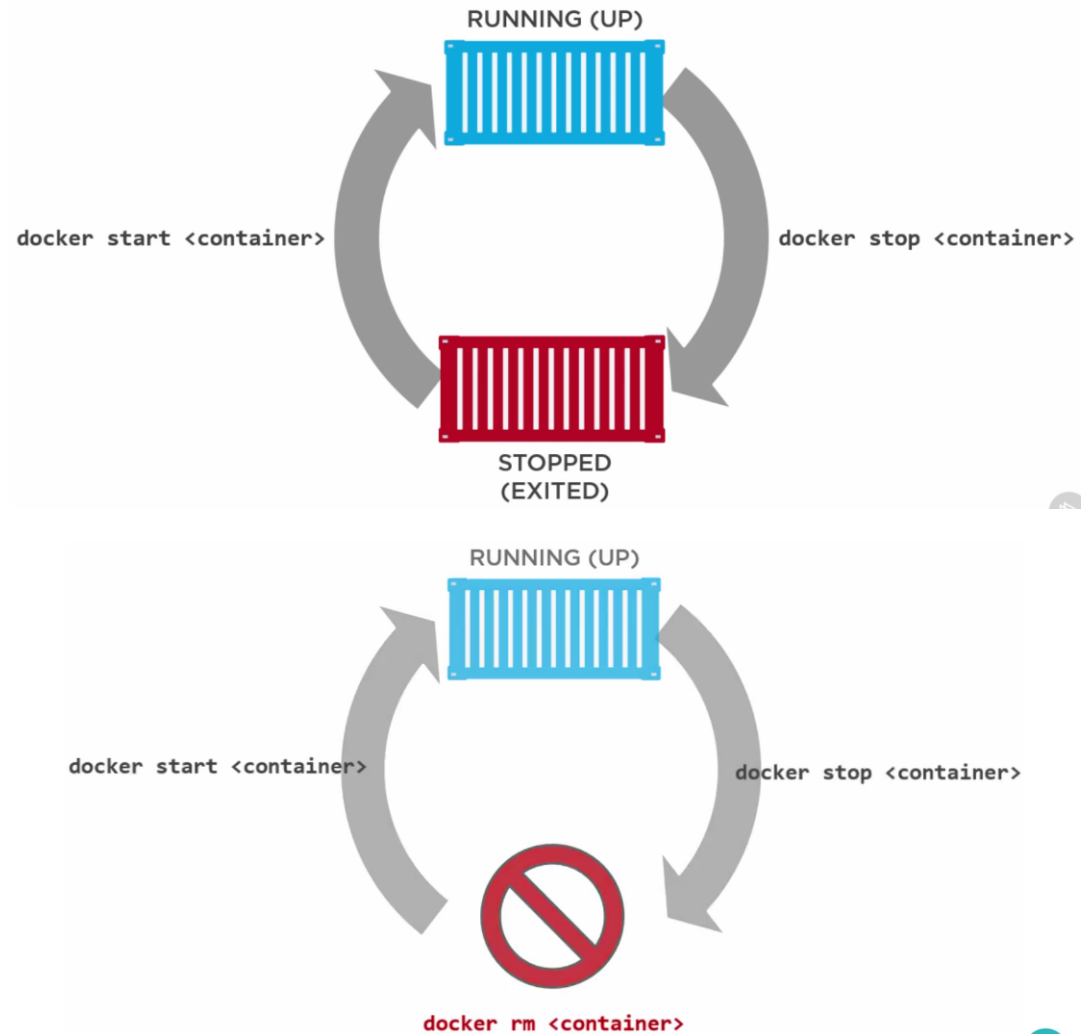
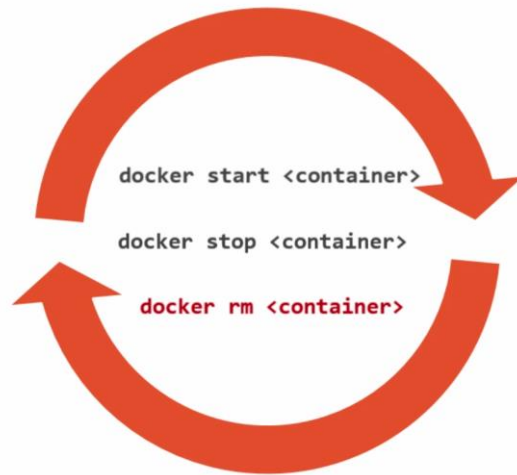
### Registry

A service that provides access to repositories, either through Docker Hub or Azure Container Registry

# Docker: Container Lifecycle

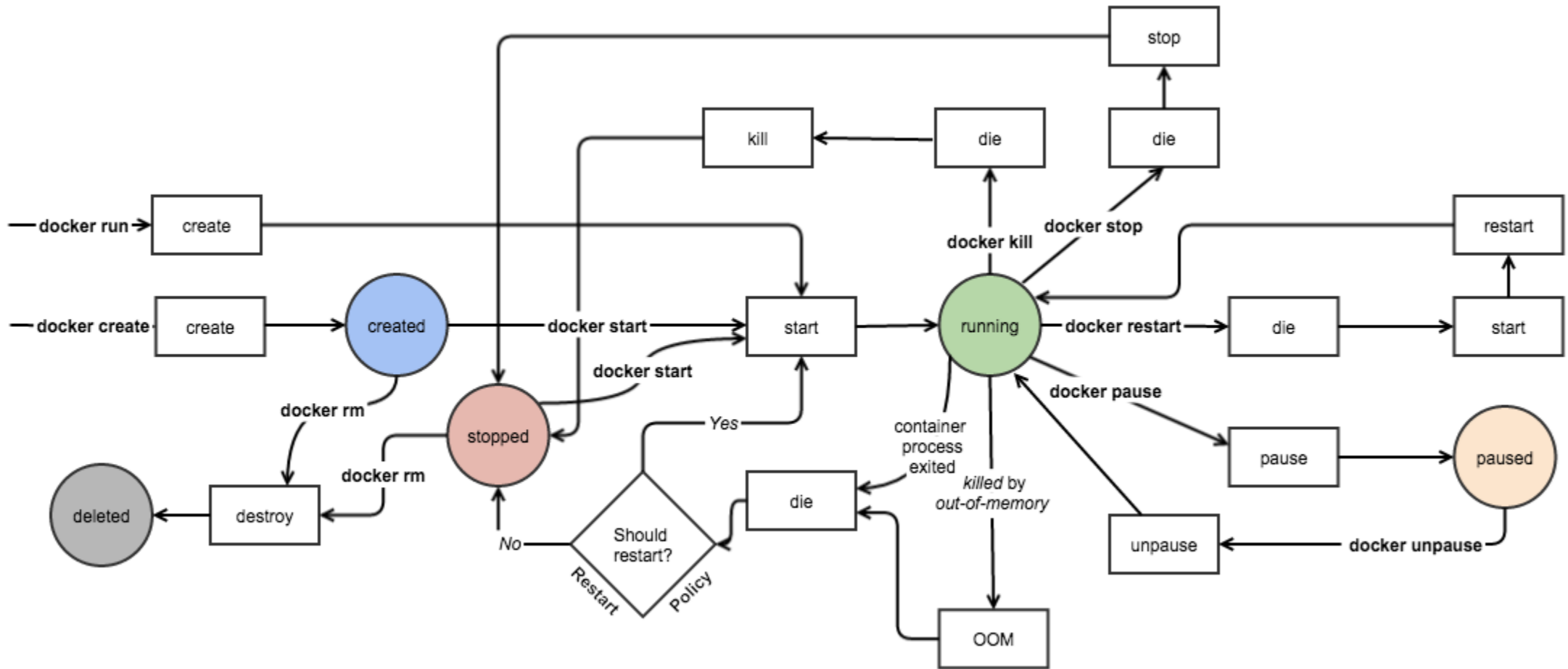
## Docker: Container Lifecycle

Container lifecycle ~ VM lifecycle



# Docker: Container Lifecycle

## Docker: Container Lifecycle



# Docker commands

## Docker: Container Lifecycle

`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 images` -> List images

`docker ps` -> List Docker containers.

`docker rm` -> Remove one or more containers

`docker rmi` -> Remove one or more images

`docker pull` -> Pull an image or a repository from a registry

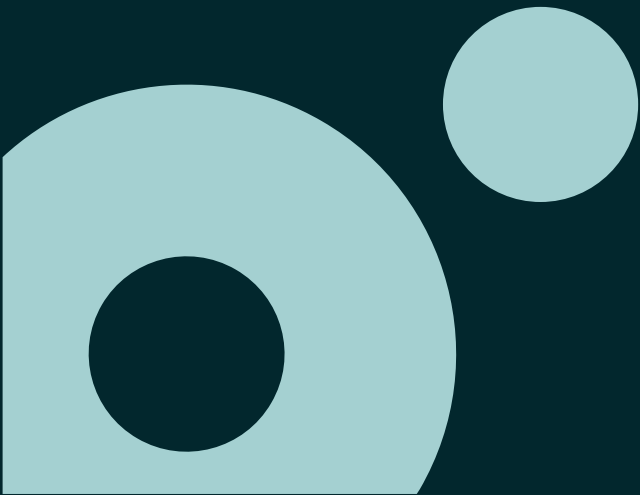
`docker push` -> Push an image or a repository to a registry

`docker search` -> Search the Docker Hub for images



| Demo: Container Lifecycle

# Linux vs Windows Containers





# Linux Containers

## Linux vs. Windows Containers

Containers started to be available only on Linux hosts with Linux Containers

Now you may use Docker Desktop to manage and handle containers on Windows Host

Windows Host can run Linux Containers using VMs or (better approach) WSL 2

Windows Subsystem for Linux 2 allow you to run Linux inside Windows

# Windows Containers

## Linux vs. Windows Containers

For running Windows Containers you need to have docker running on Windows Host

Docker Desktop is a standard solution for developer machine (now with licensing...)

For production environments you need to enable Containers feature on Windows Server (native on 2019 and 2022)

Windows Container version needs to be equal or less than Windows Host Machine Kernel

# Windows Containers

## Linux vs. Windows Containers

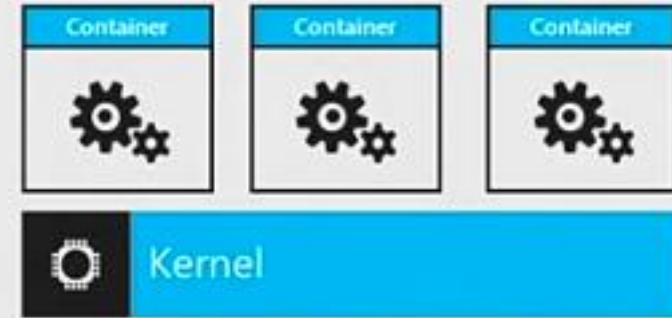
### Windows Server Container

applications placed in either a regular Windows Server Container or a Hyper-V Container without any modification.

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

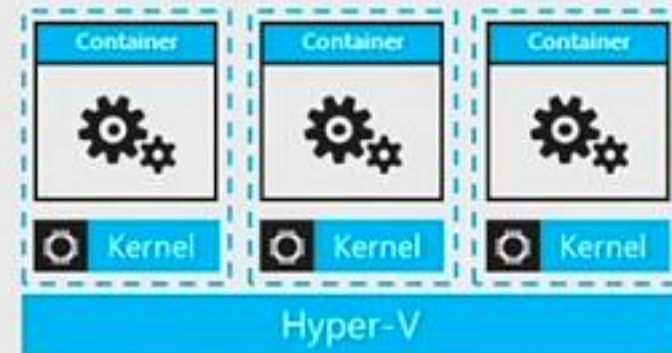
### Windows Server containers

Maximum speed and density



### Hyper-V containers

Isolation plus performance



# Windows Containers

## Linux vs. Windows Containers

**Windows** ([https://hub.docker.com/\\_/microsoft-windows](https://hub.docker.com/_/microsoft-windows)) \*New in Windows Server 2019

- Automation workloads

- Carries most Windows OSS components

**Windows Server Core** ([https://hub.docker.com/\\_/microsoft-windows-servercore](https://hub.docker.com/_/microsoft-windows-servercore))

- Minimal installation of Windows Server 2016

- Contains only core OS features

- Command-line access only

**Nano Server** ([https://hub.docker.com/\\_/microsoft-windows-nanoserver](https://hub.docker.com/_/microsoft-windows-nanoserver))

- Available only as container base OS image (no VM support)

- 20 times smaller than Server Core

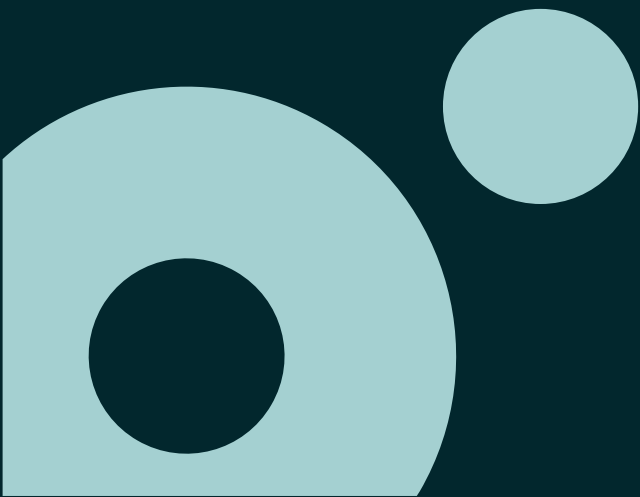
- Headless – no logon or GUI

- Optimized for .NET Core applications



| Demo: Windows Containers

# Lab



# Lab 1: Container Lifecycle

## Github

Navigate to <https://github.com/tasb/docker-kubernetes-training>

Read README.md for more details about the repo

Run Lab#01: <https://github.com/tasb/docker-kubernetes-training/blob/main/labs/lab01.md>



● Rua Sousa Martins, nº 10  
1050-218 Lisboa | Portugal