# Microsoft *Application Containers for Geeks

# Mike Nelson

Infrastructure Architect – SECURA Insurance

Microsoft MVP – Cloud & Datacenter
Microsoft Azure Advisor
Citrix CTP
VMware vExpert

@nelmedia
nelmedia@gmail.com

# Discuss

What Containers Are

Container Images

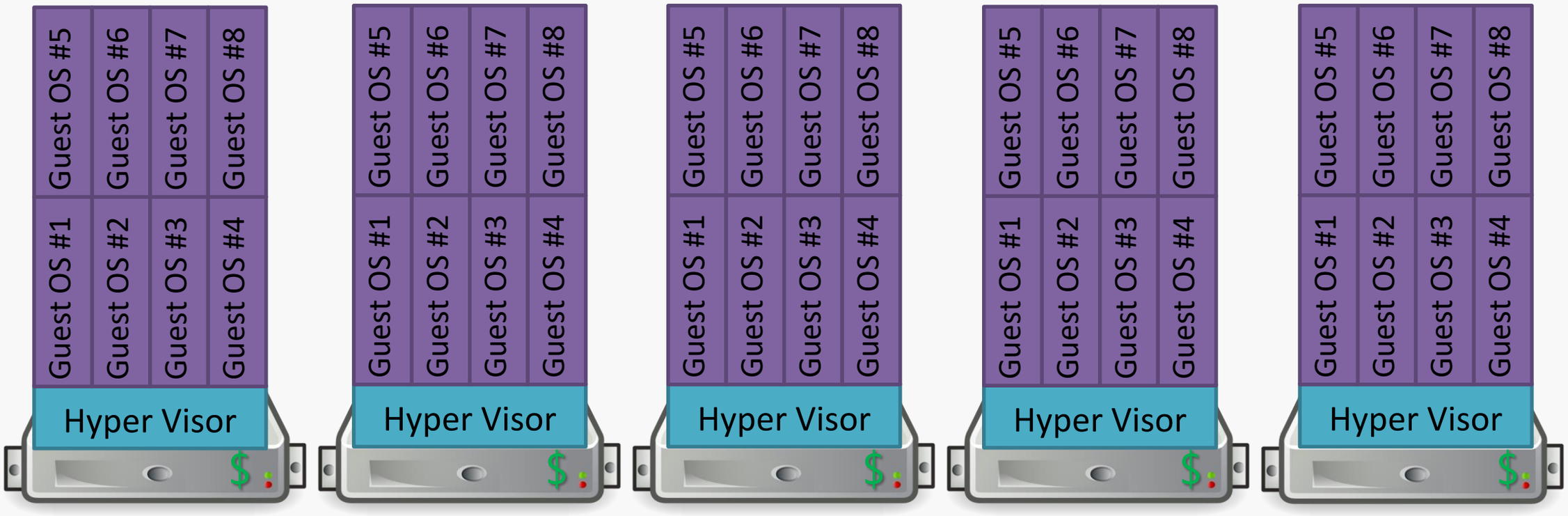
Why Use Them?

But What About…

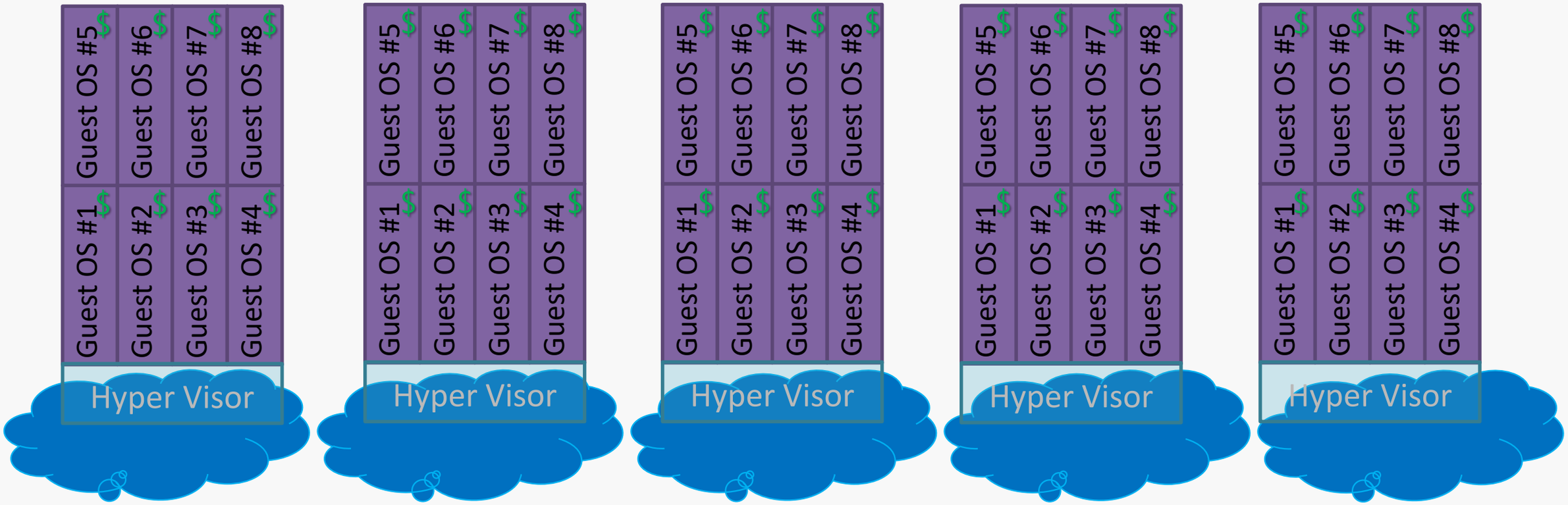
Demo's & Labs

# Mainframe / Mini → Server Revolution

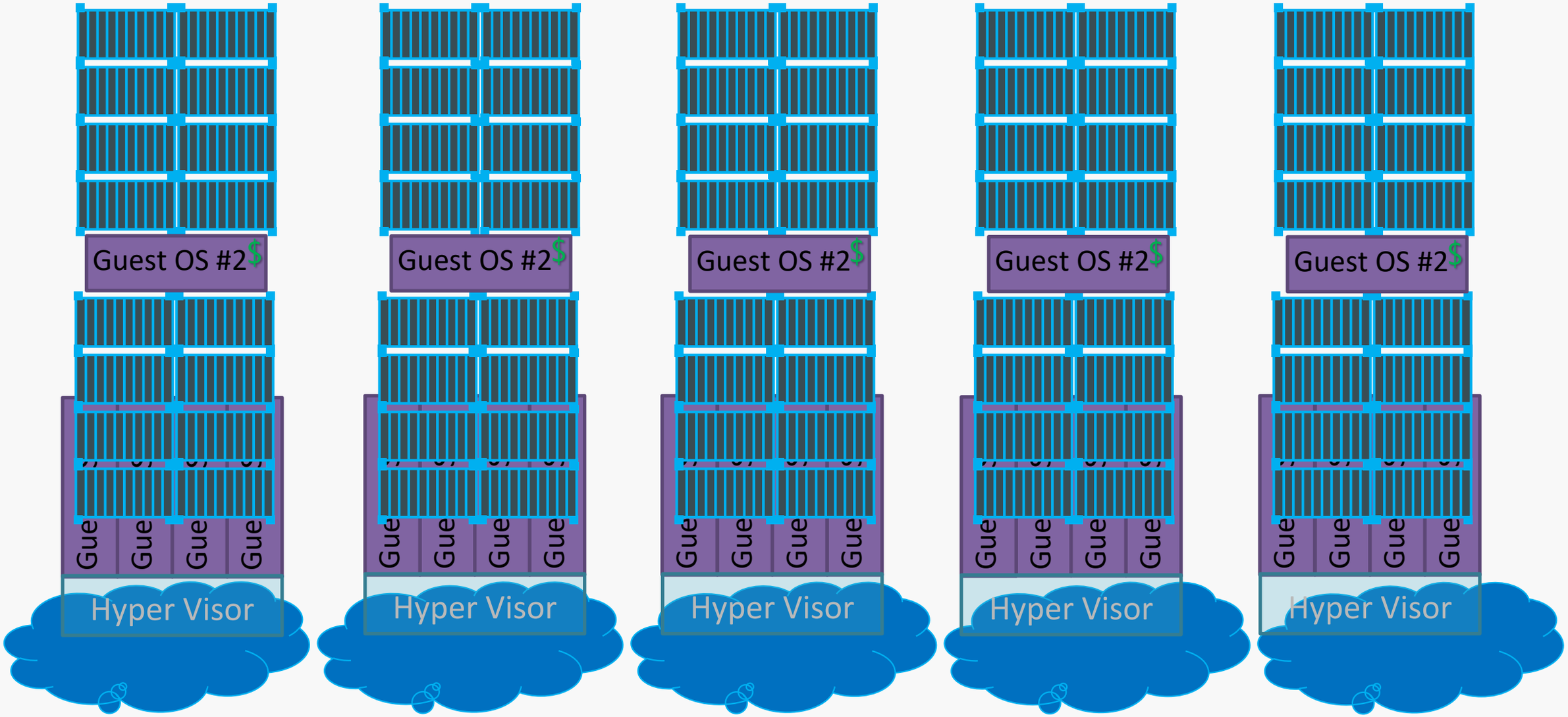
# Server → VM Evolution



# VM → Cloud Evolvement



# Cloud → Containerization




@nelmedia 2016

1979
CHROOT
UNIX V7

2000 BSD
JAILS &
SOLARIS
ZONES

2006
PROCESS
CONTAINERS

2008 LXC

2013
DOCKER

2015 AZURE
CONTAINERS

2016
ROCKET -
PHOTON

2016
WINDOWS
CONTAINERS

Now on GitHub!

# Microsoft ❤ Linux

Server 2016 GA!

System Center 2016 GA!

CS Docker for Server 2016 GA!

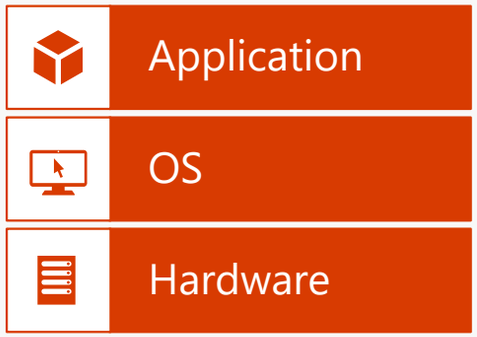
# What is a container?

# What is a container?

Will containers replace VM's? No.

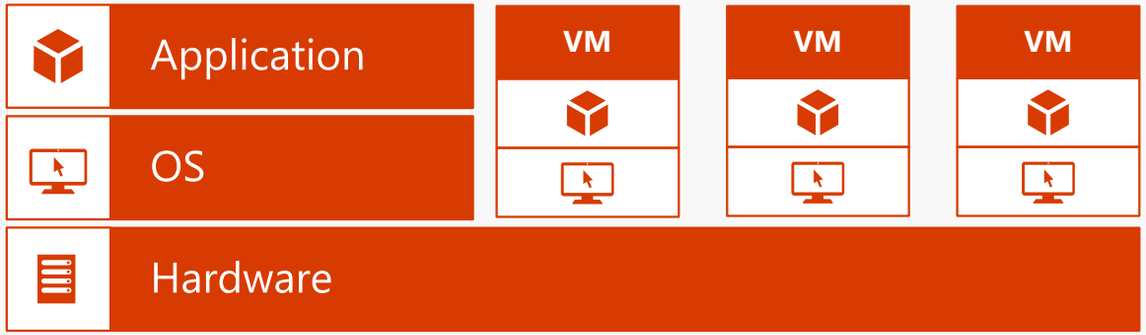
Containers are not a virtualization technology.
They are an application delivery technology.

# What is a container?



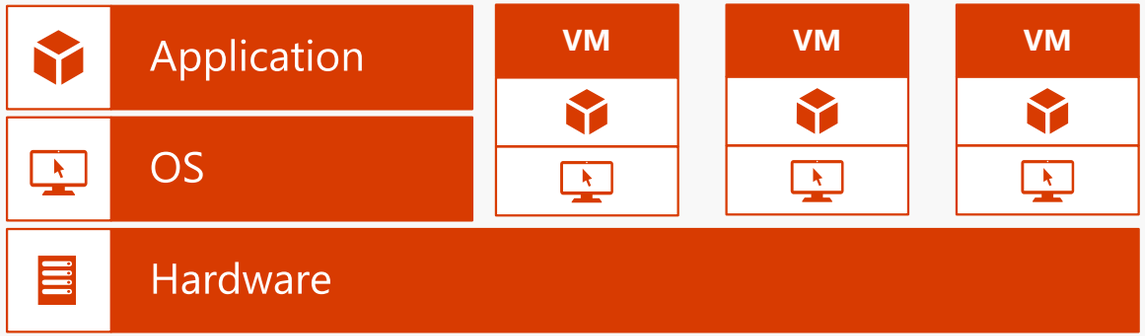
# What is a container?

**Traditional virtual machines** = hardware virtualization



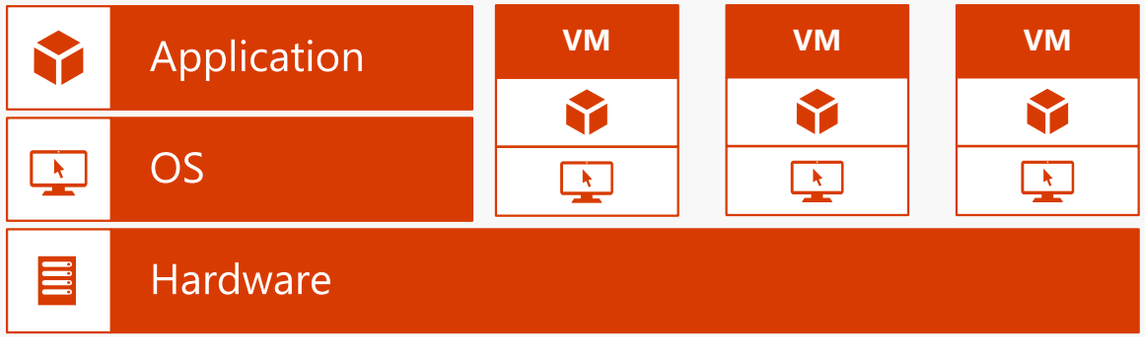
# What is a container?

**Traditional virtual machines** = hardware virtualization



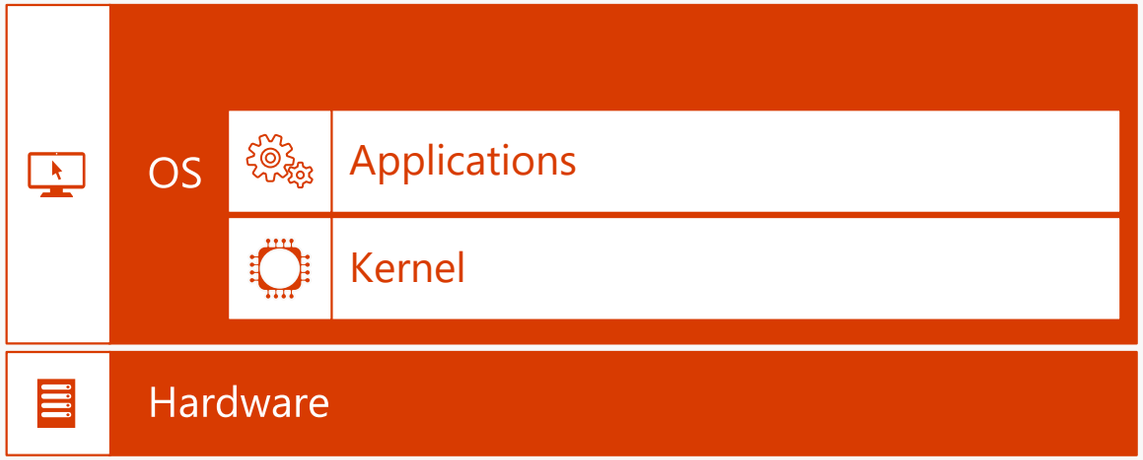
# What is a container?

**Traditional virtual machines** = hardware virtualization

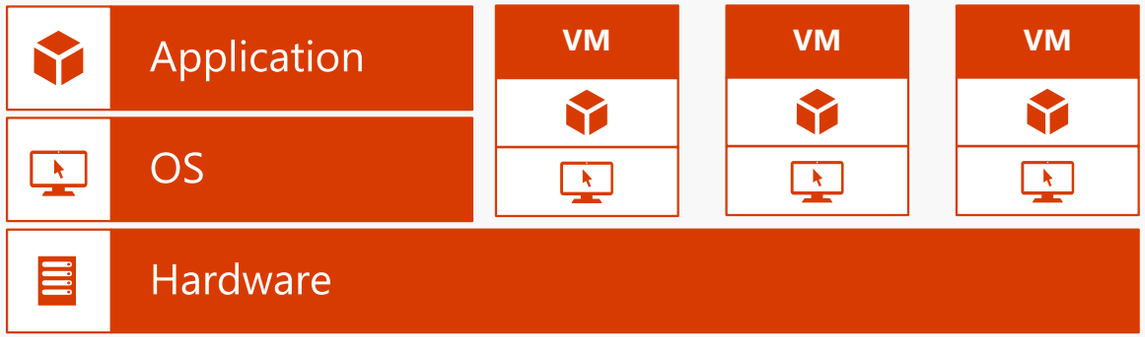


# What is a container?

**Containers** = Operating system virtualization

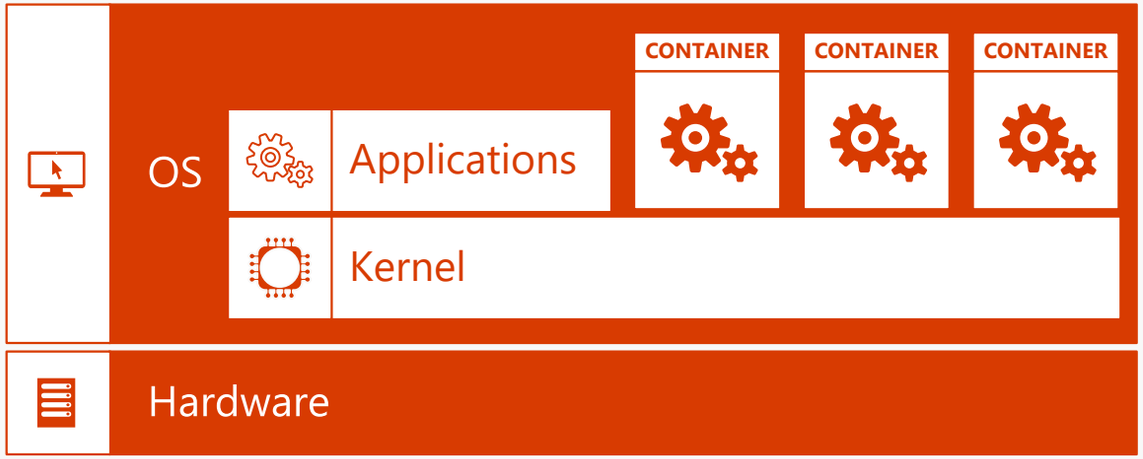



**Traditional virtual machines** = hardware virtualization

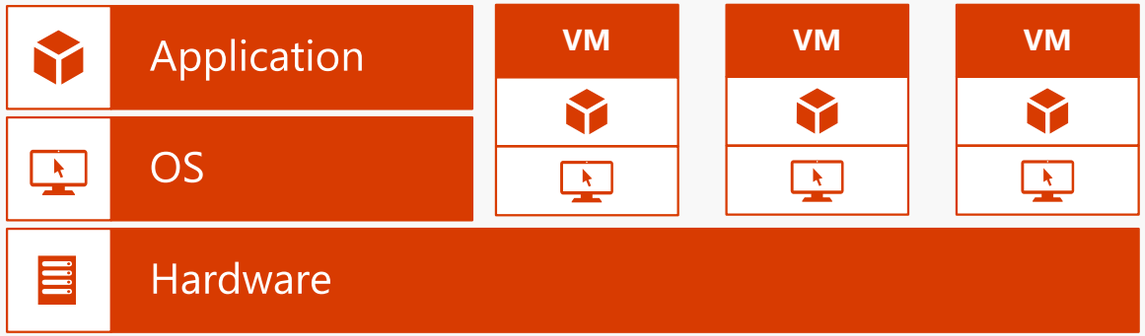


# What is a container?

**Containers** = Operating system virtualization

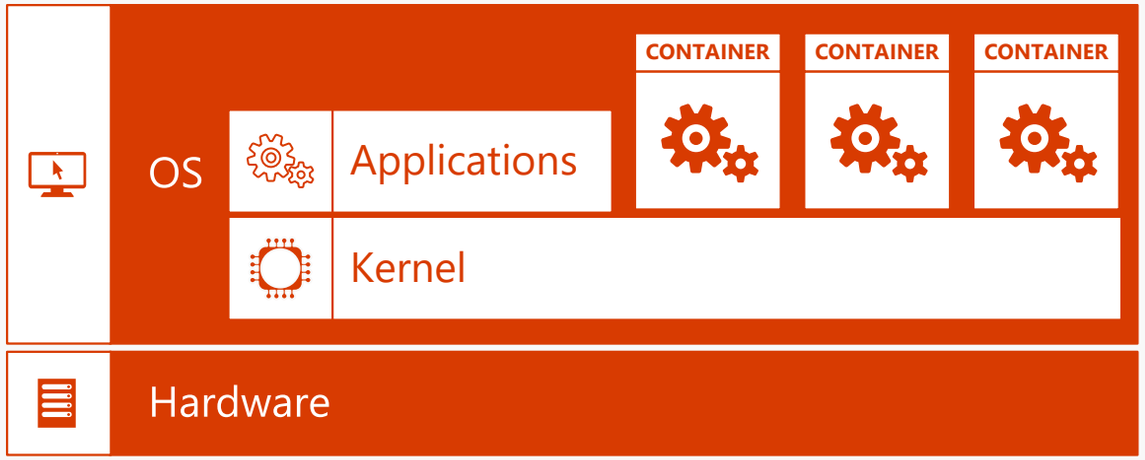



**Traditional virtual machines** = hardware virtualization

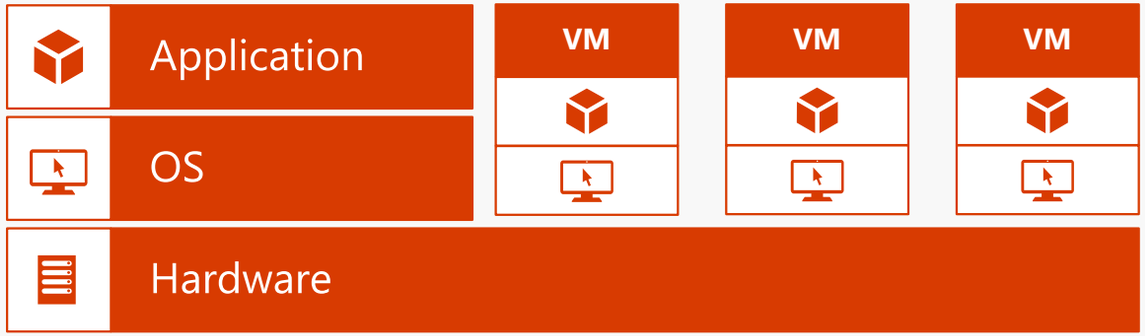


# What is a container?

**Containers** = Operating system virtualization

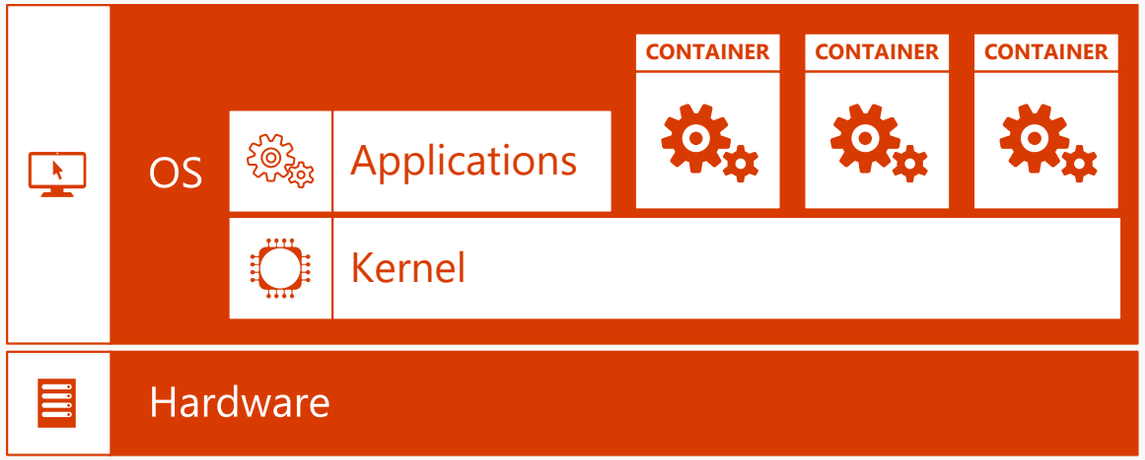



**Traditional virtual machines** = hardware virtualization

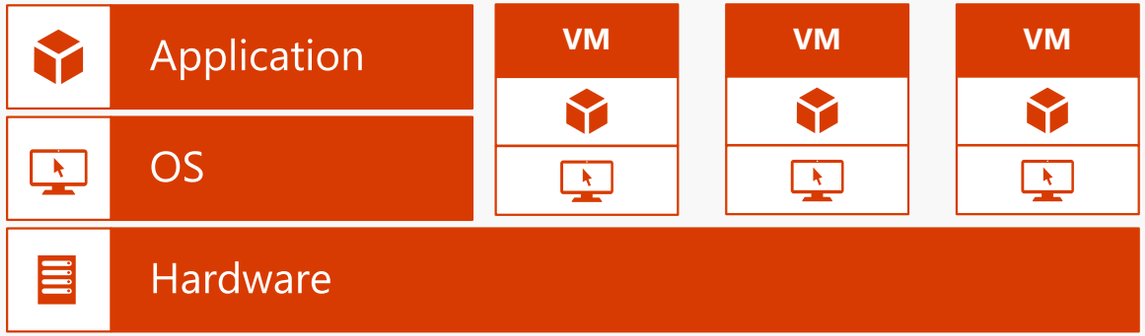


# What is a container?

**Containers** = Operating system virtualization

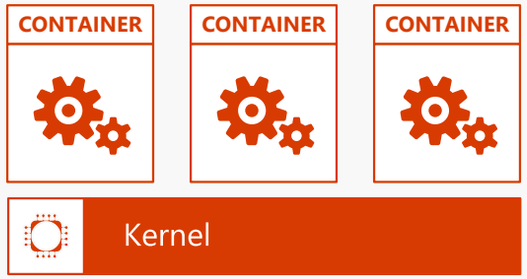



**Traditional virtual machines** = hardware virtualization



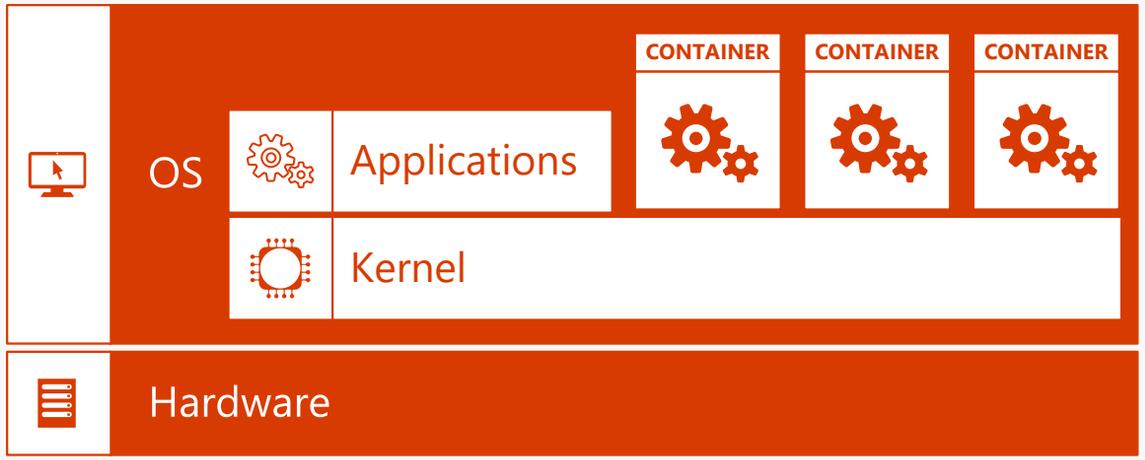

**Windows Server Containers**

Maximum speed and density

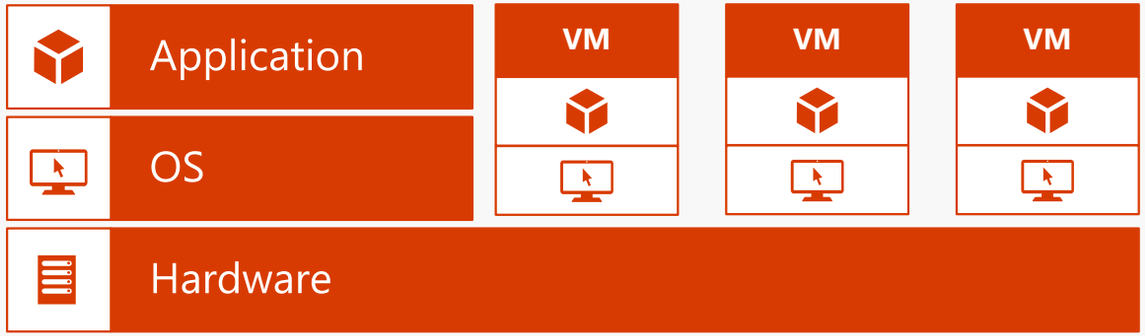


# What is a container?

**Containers** = Operating system virtualization

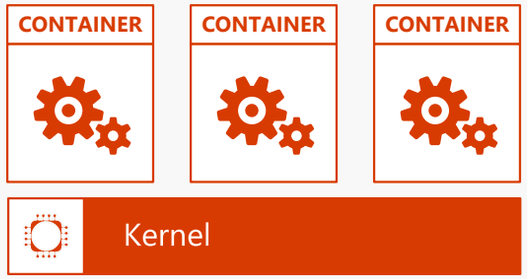



**Traditional virtual machines** = hardware virtualization



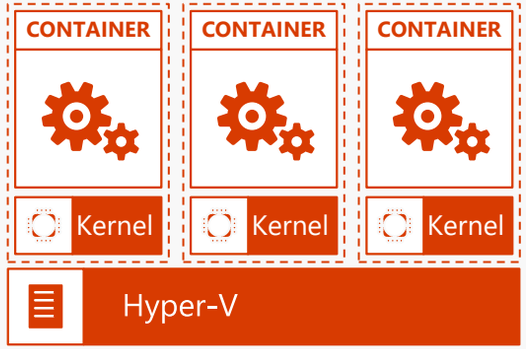

**Windows Server Containers**

Maximum speed and density

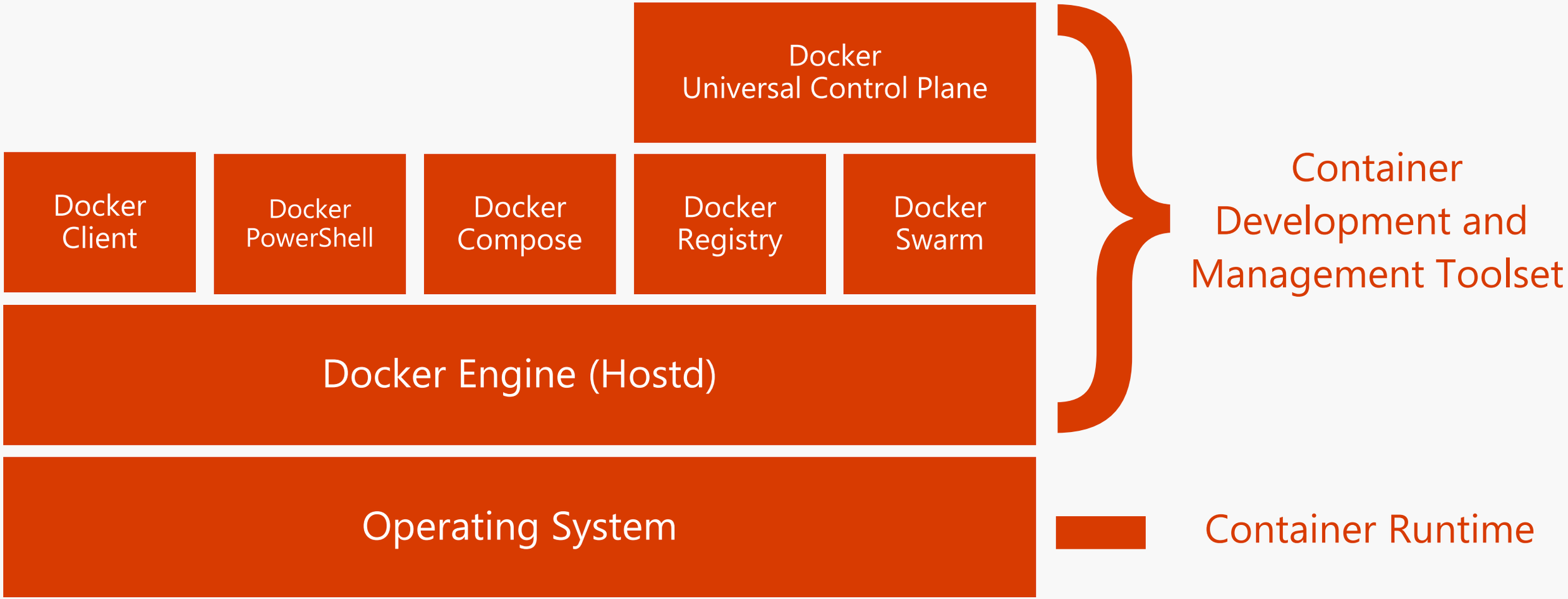



**Hyper-V Containers**

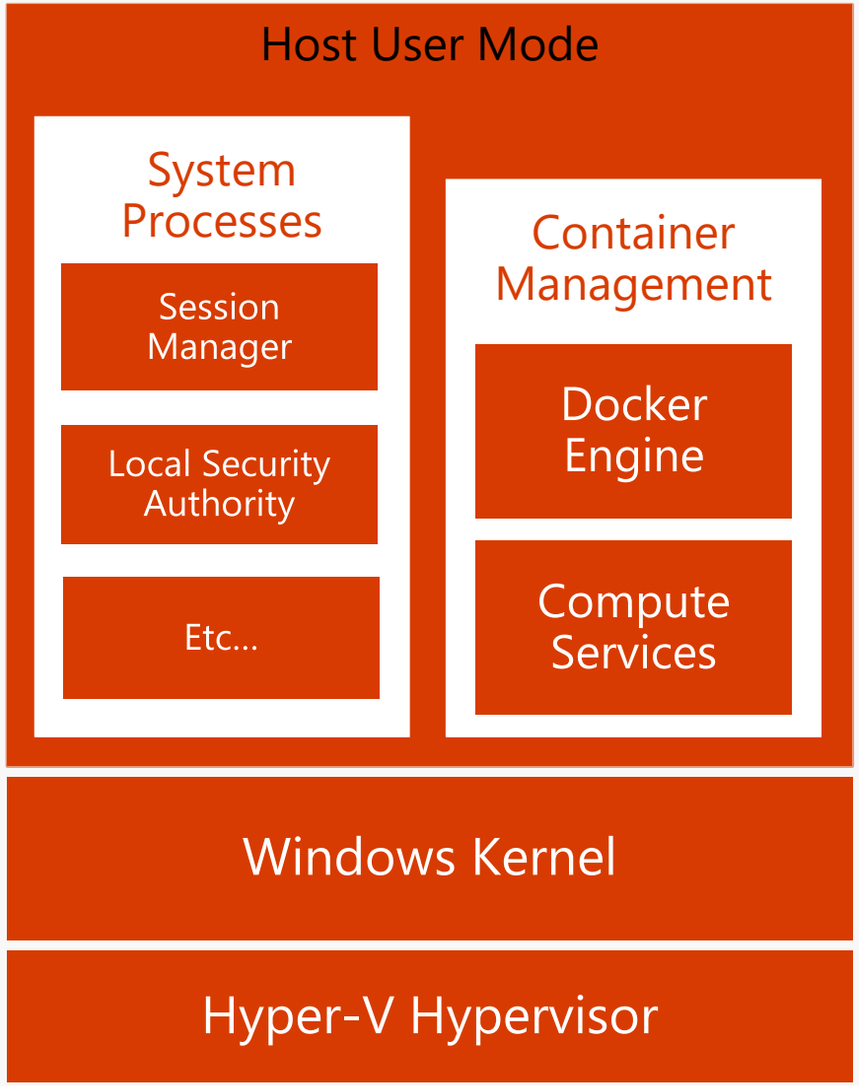
Isolation plus performance



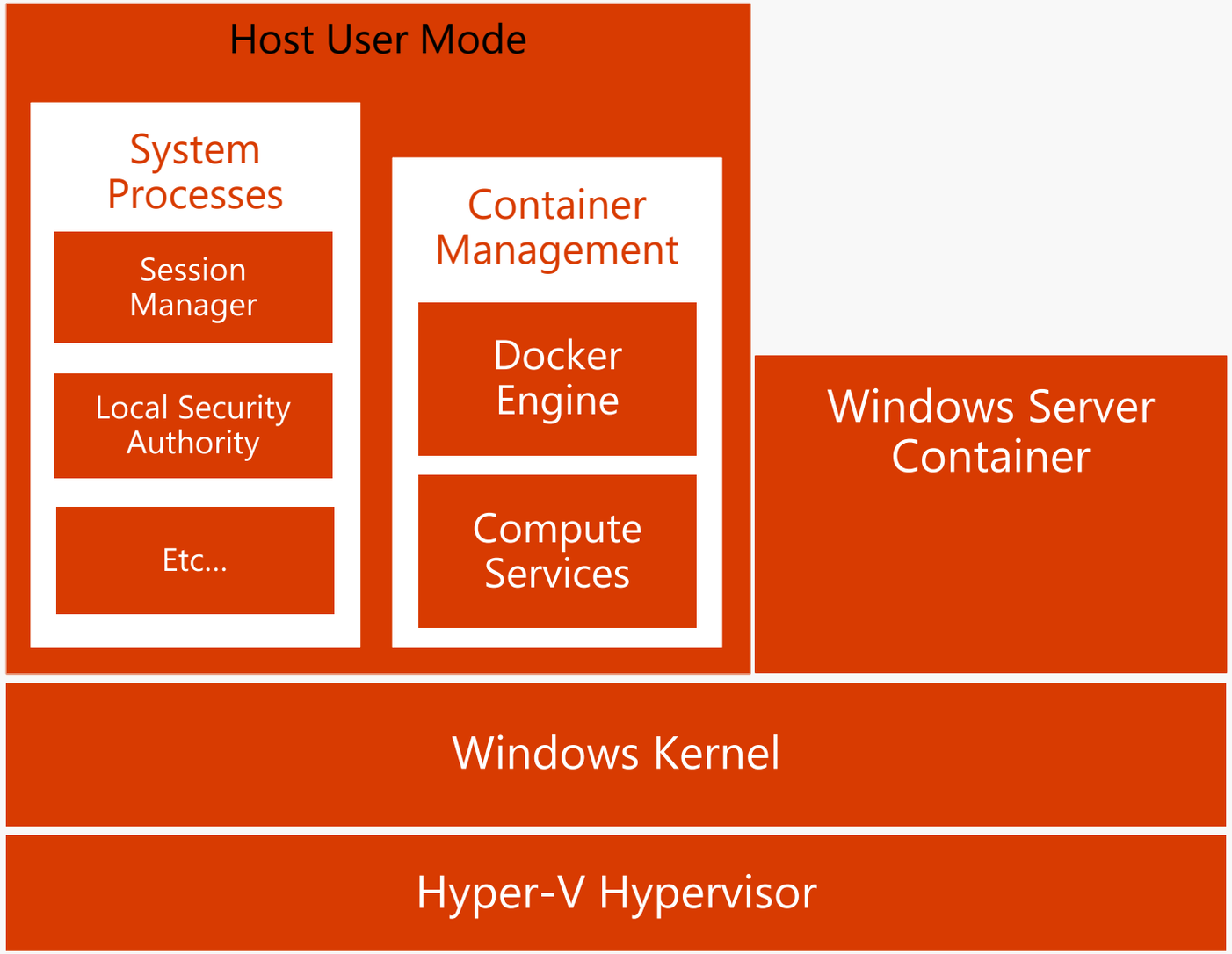
# High Level Architecture



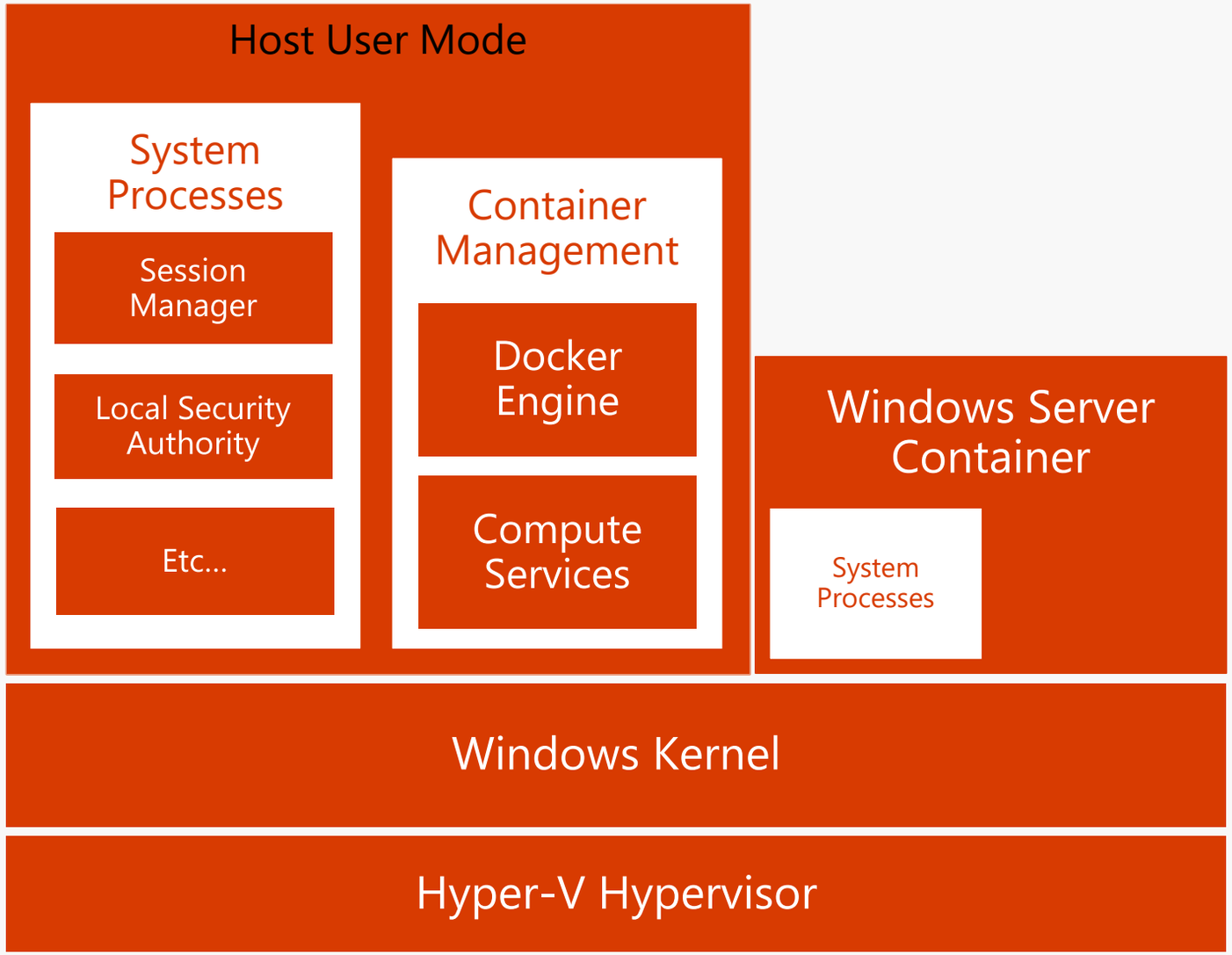
# Windows Containers



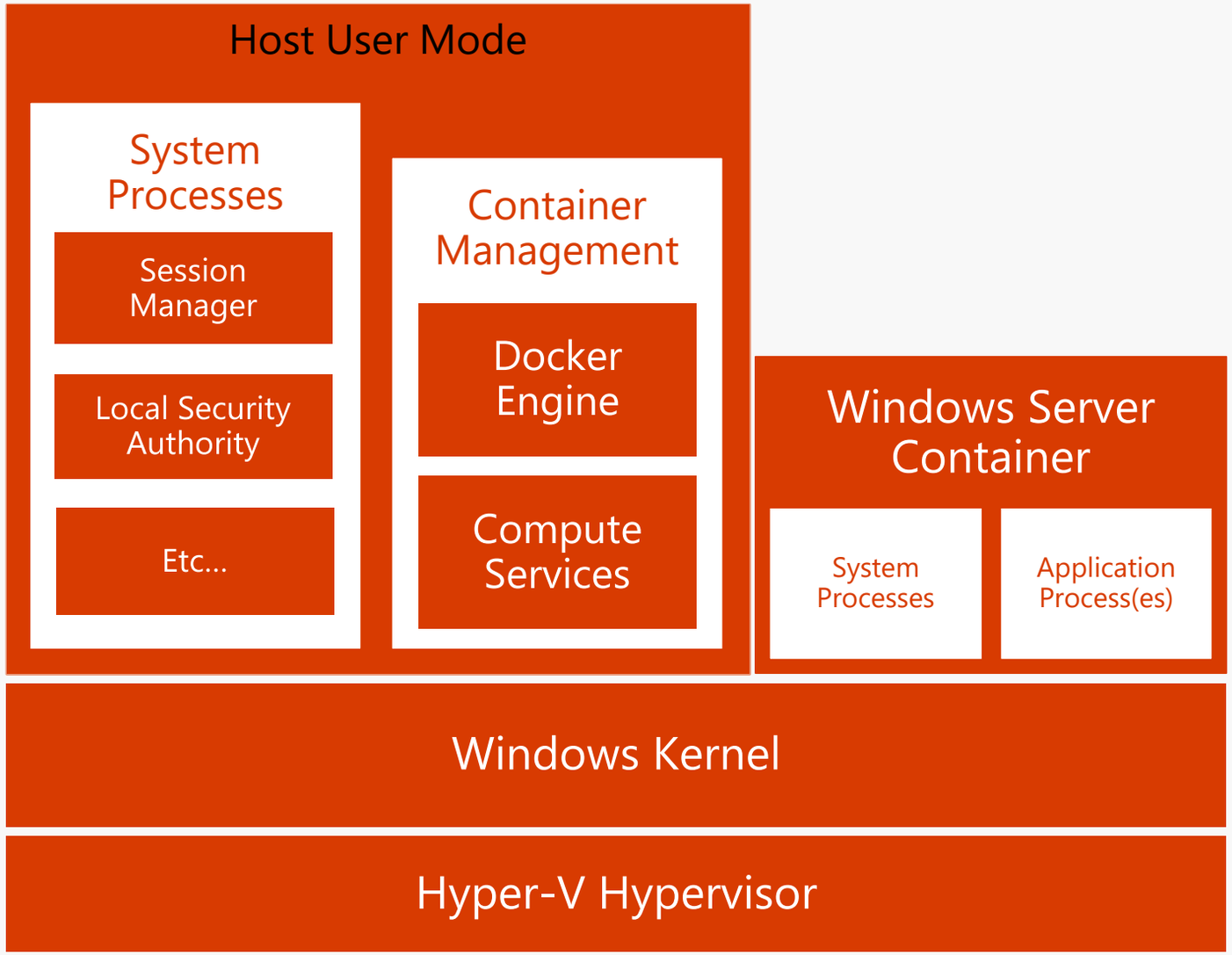
# Windows Containers



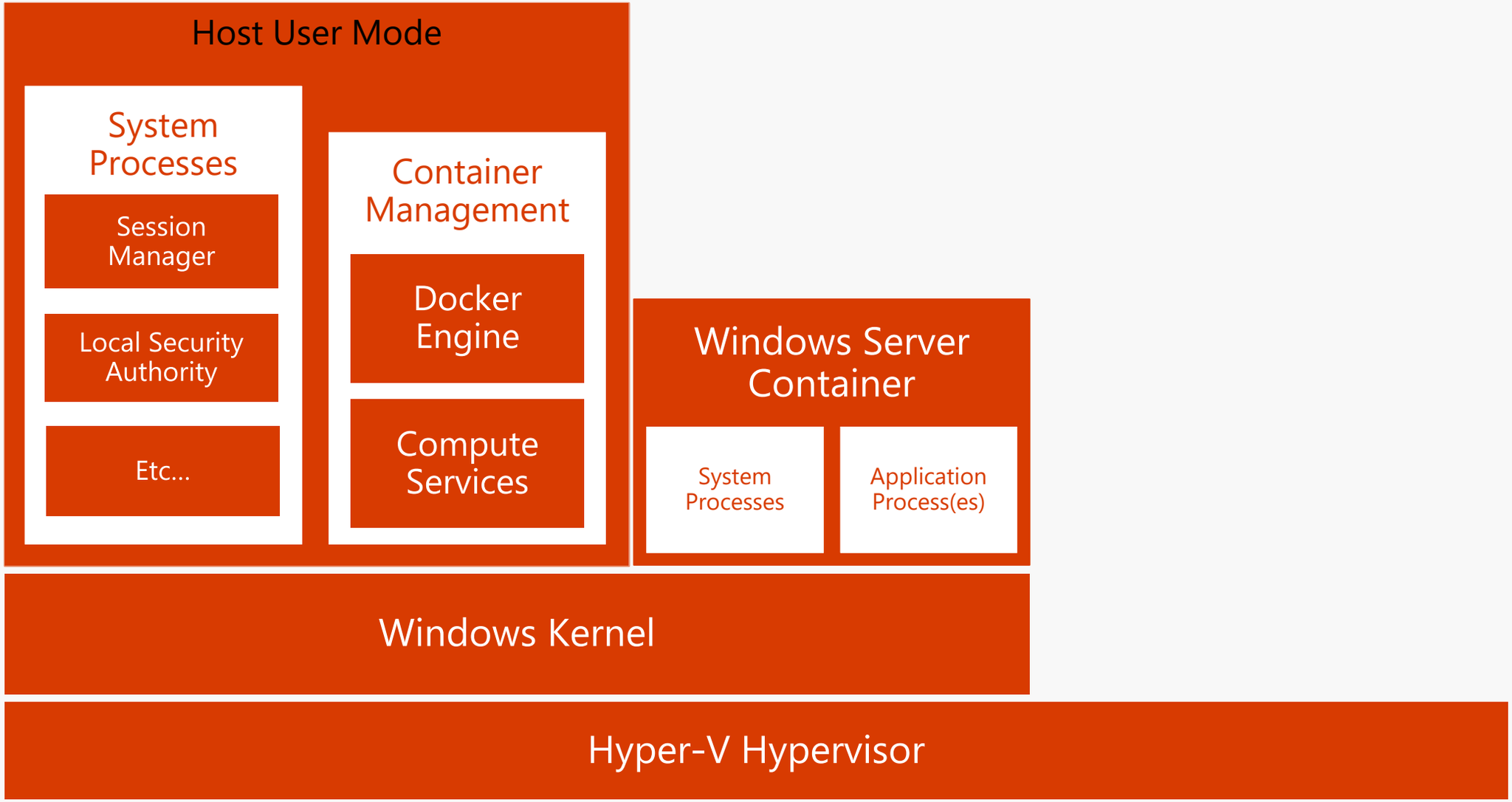
# Windows Containers



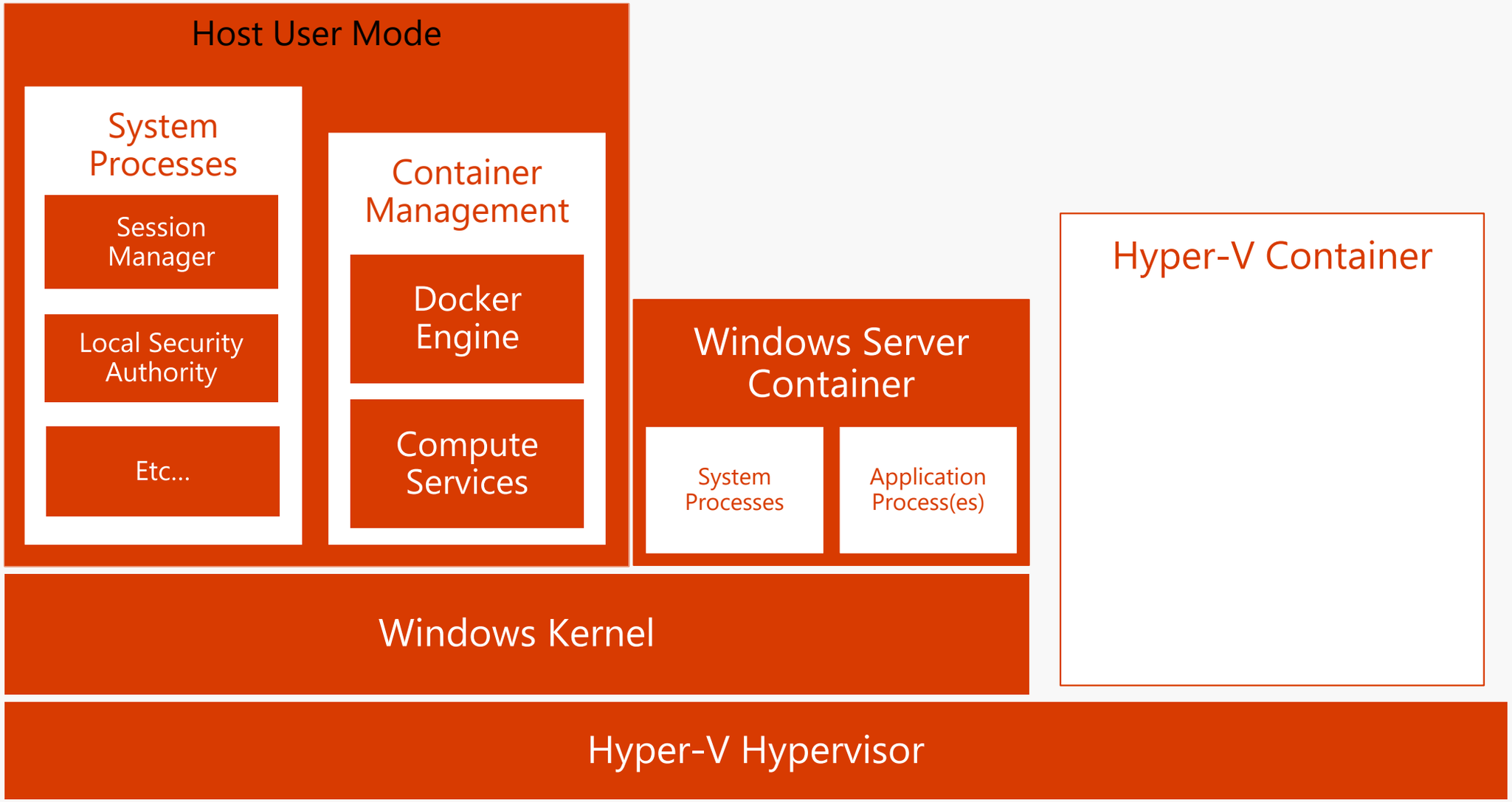
# Windows Containers



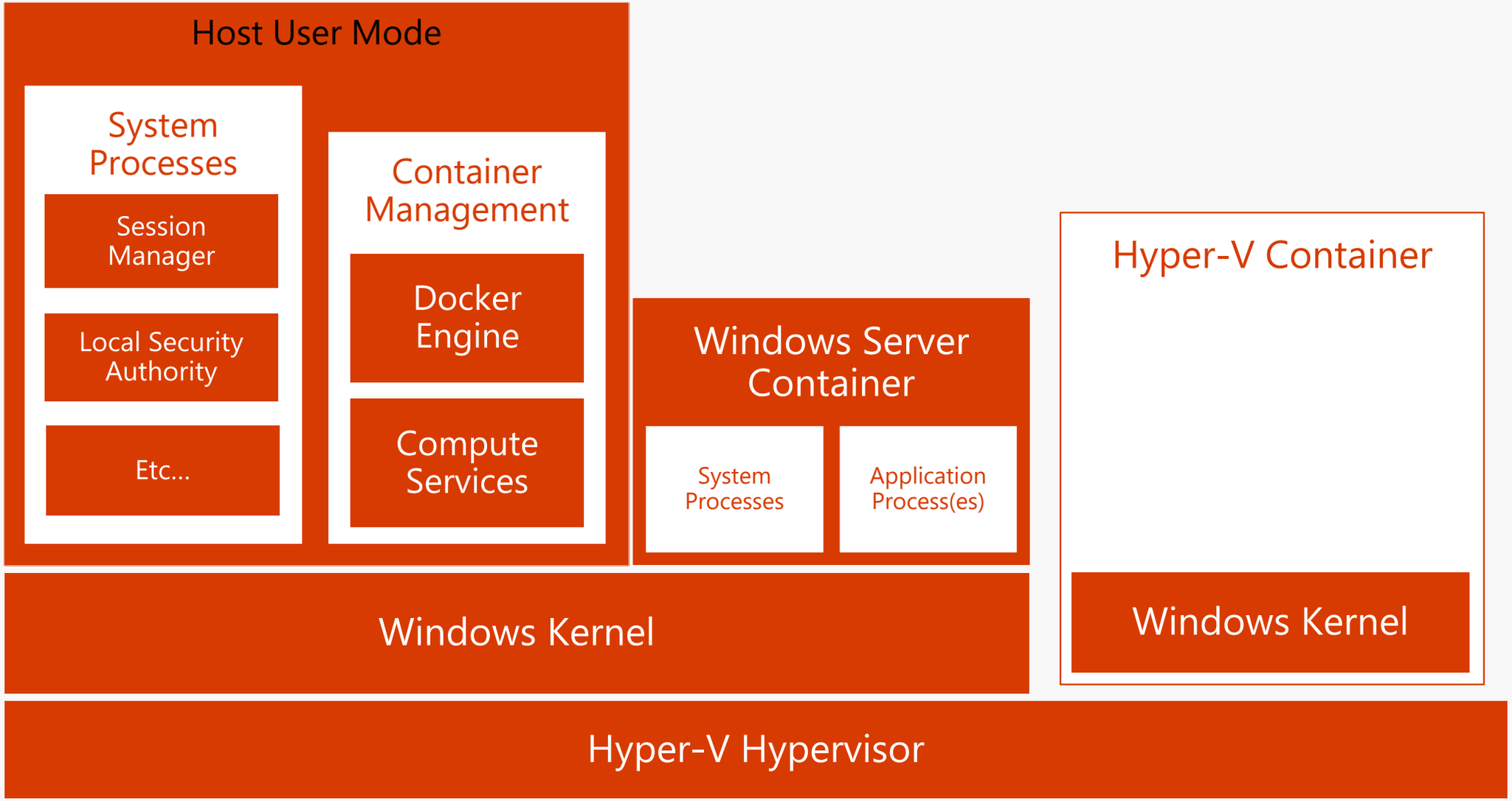
# Windows Containers



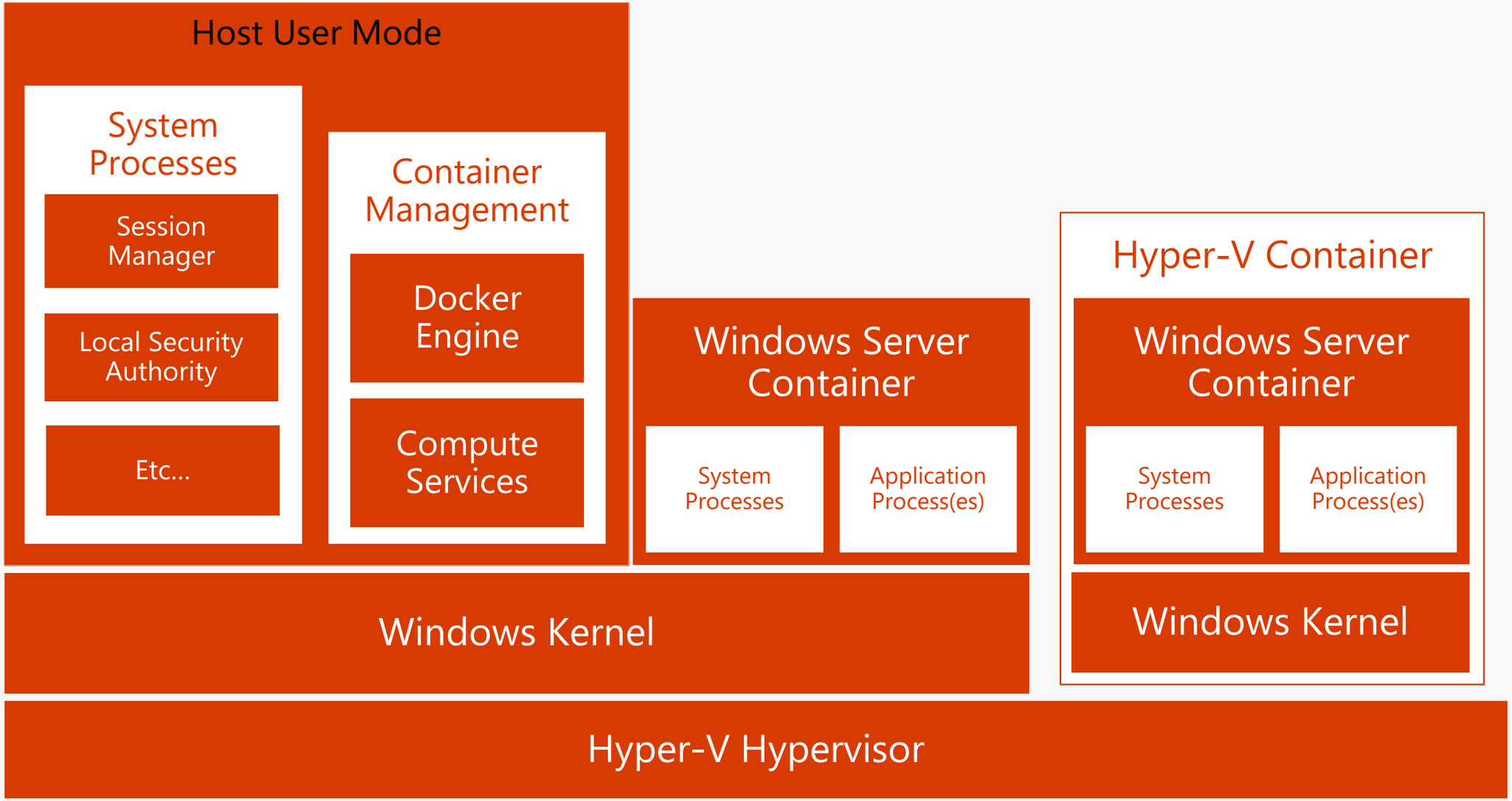
# Windows Containers



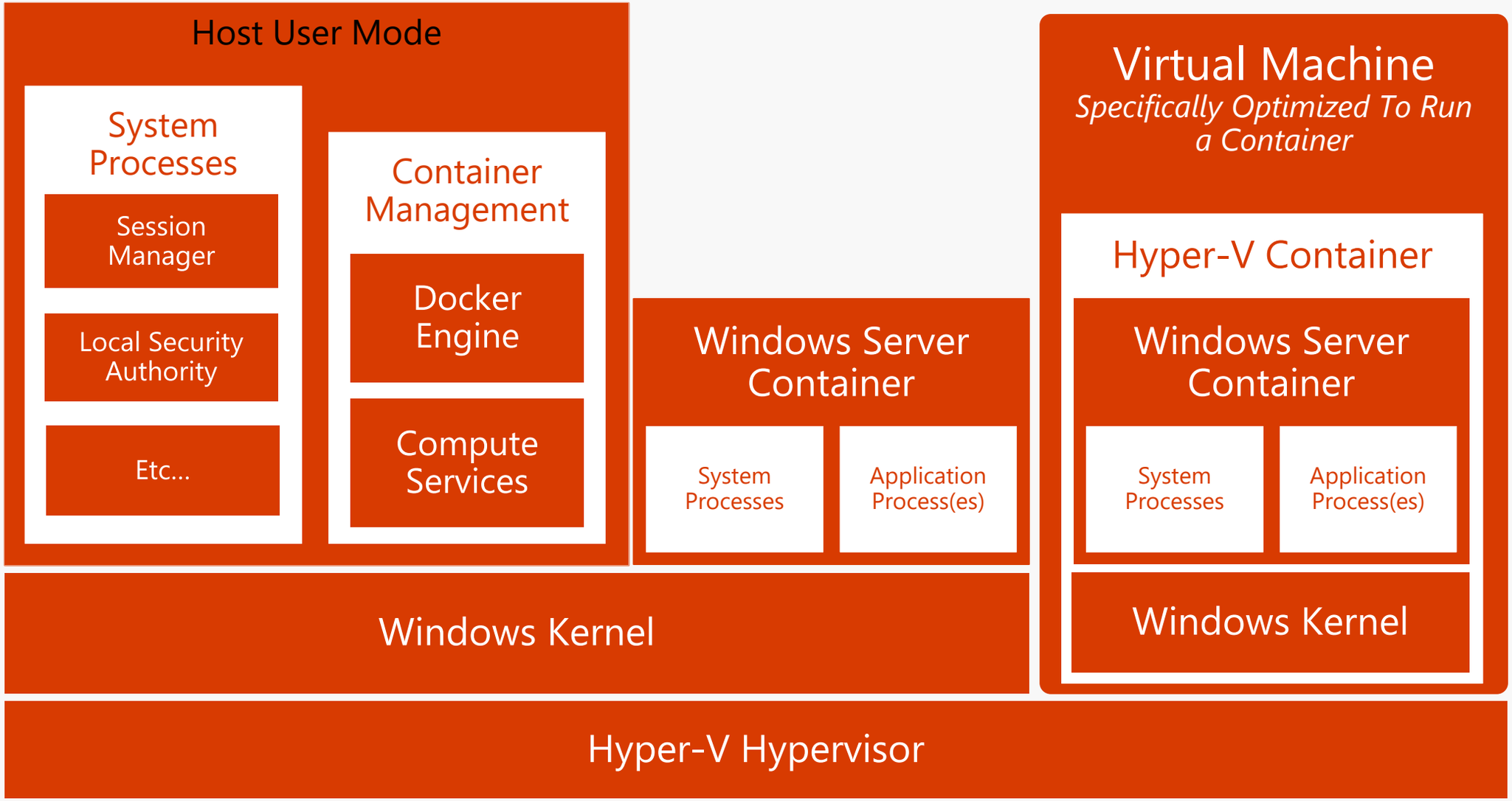
# Windows Containers



# Windows Containers



# Windows Containers



# Startup Performance

## NodeJS with Windows Server Core

Windows Server Container
**~1 second**

Hyper-V Container
**~3.3 seconds**

A virtual machine takes ~**5 seconds to over a min**

*Includes initial boot provisioning phase (out-of-box setup etc…)

**Startup time after initial container start.

Testing performed on HP ProLiant SL250s Gen8, E5-2600, 2 Socket, 8 Core, 128GB RAM, HP SATA SSD - results may vary based on hardware and software configurations.

# Startup Performance

## NodeJS with Nano Server

Windows Server Container
**Under 600 Milliseconds!**

Hyper-V Container
**~1.75 seconds**

A virtual machine takes ~**3 seconds**

## NodeJS with Windows Server Core

Windows Server Container
**~1 second**

Hyper-V Container
**~3.3 seconds**

A virtual machine takes ~**5 seconds to over a min**

# Density

## NodeJS with Windows Server Core

| Windows Server Container | Hyper-V Container |
| --- | --- |
| First Container **~150MB** | First Container **~555MB** |
| Additional Containers **~75MB** | Additional Containers **~280MB** |

Testing performed on HP ProLiant SL250s Gen8, E5-2600, 2 Socket, 8 Core, 128GB RAM, HP SATA SSD - results may vary based on hardware and software configurations.

# Density

## NodeJS with Nano Server

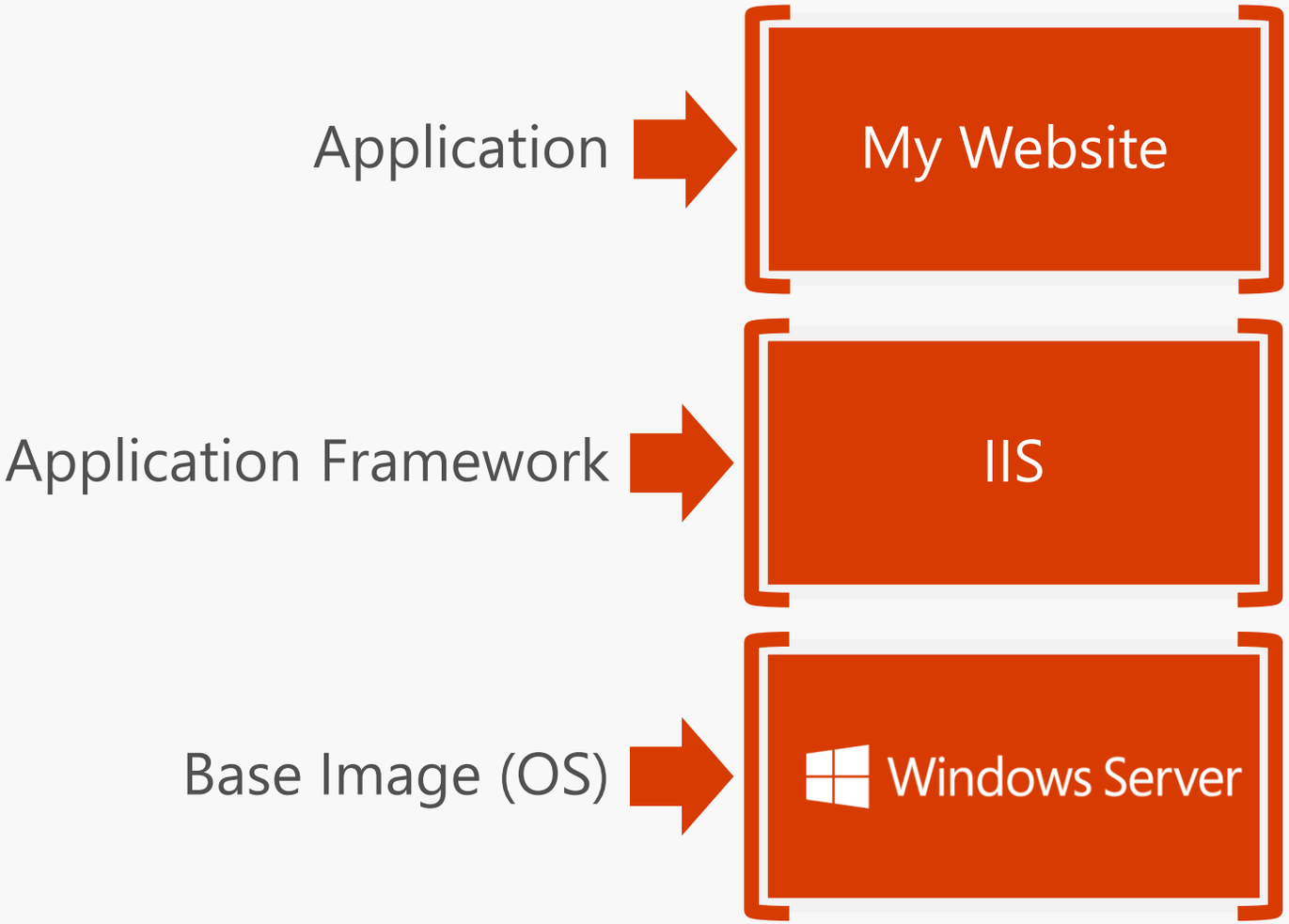
| Windows Server Container | Hyper-V Container |
| --- | --- |
| First Container **~120MB** | First Container **~340MB** |
| Additional Containers **~75MB** | Additional Containers **~150MB** |

## NodeJS with Windows Server Core

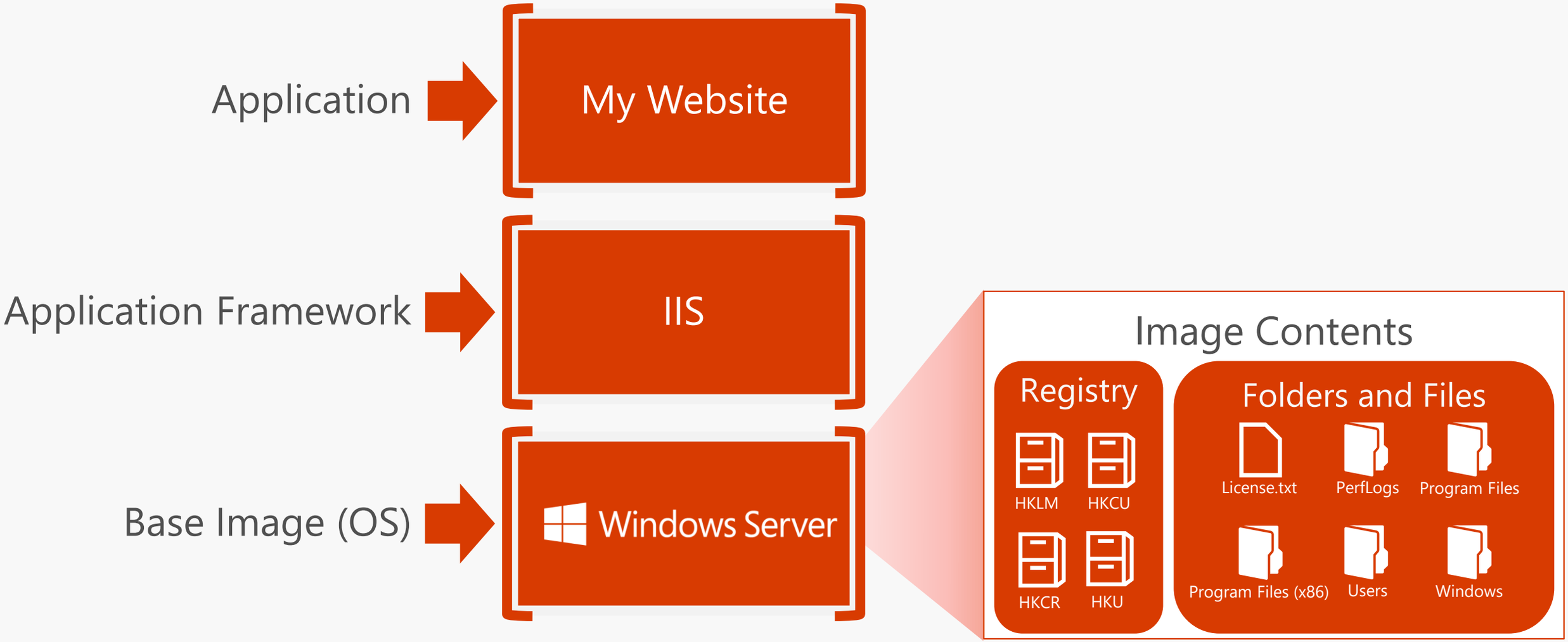
| Windows Server Container | Hyper-V Container |
| --- | --- |
| First Container **~150MB** | First Container **~555MB** |
| Additional Containers **~75MB** | Additional Containers **~280MB** |

# What is a container image?

# Container Images



# Container Images



# Container Images

Application → My Website

Application Framework → IIS

Base Image (OS) → Windows Server

## Image Contents

### Registry

HKLM

SOFTWARE/
mykey

### Folders and Files

inetpub

mysite.html

## Image Contents

### Registry

HKLM

HKCU

HKCR

HKU

### Folders and Files

License.txt

PerfLogs

Program Files

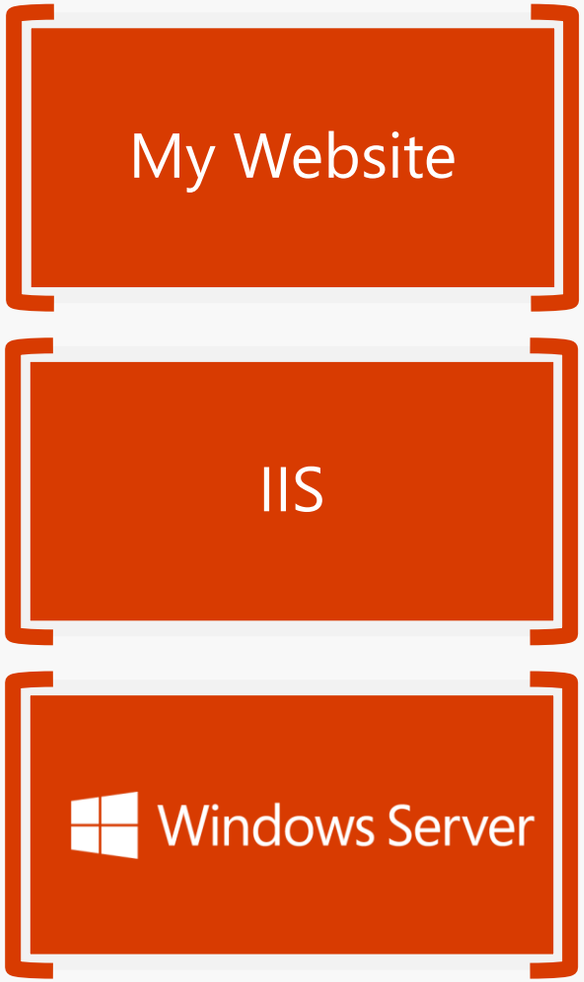
Program Files (x86)

Users

Windows

# Docker Build and Dockerfiles

Method for automated container image build
Consumed when running “docker build”
Caches unchanged commands
Integrates into Docker Hub




# Examples

IIS

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

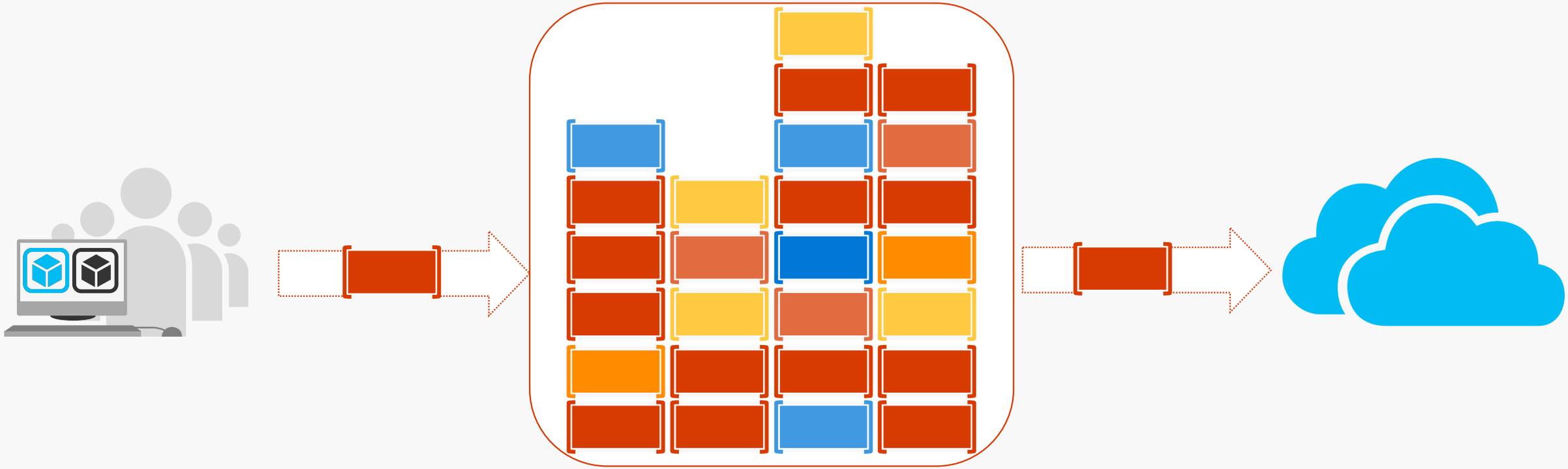
Website

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

# What is a registry?

Stores container images

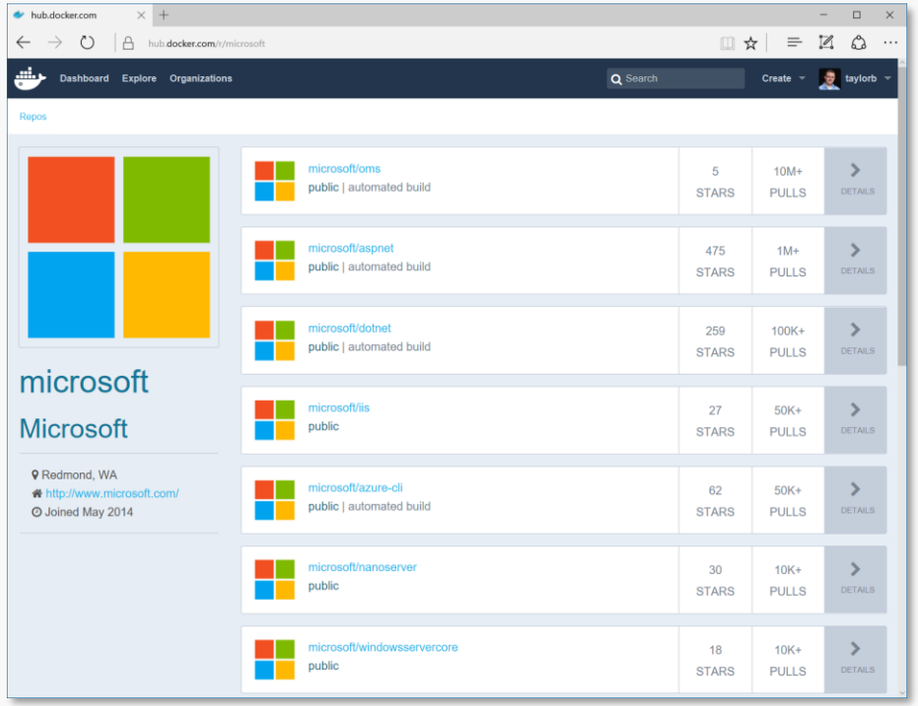
- Images are **Pushed** into a registry
- Images are **Pulled** from a registry
- Images are **Searched** for within a registry



# 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

Enterprise Grade Private Registries
Runs on your infrastructure (on-prem or cloud)
Active Directory and Role Based Access Controls

## Docker Registry

Open source foundation of Hub and DTR
Runs on your infrastructure (on-prem or cloud) as a container
https://docs.docker.com/registry and or https://github.com/docker/distribution

# Why Containers?

"It's all about the apps."

"Containers make it easier."

# How do containers make it easier?

Consistency

Efficiency

# How do containers make it easier?

Consistency

Efficiency

# How do containers make it easier?

## Consistency

- Rapid Deployment
- Fast Iterations
- Reproducibility
- On Any Cloud
- Immutability

## Efficiency

- Service Separation
- High Availability
- Scalability
- Test at Scale
- Reduced Costs

# Today's Typical IT Services Creation Process

# Why Containers?

**Developers**

Containers unlock ultimate productivity and freedom

Enable 'write-once, run-anywhere' apps

Can be deployed as multi-tier distributed apps in IaaS/PaaS models

Containers offers powerful abstraction for **microservices**

**Operations**

Provide standardized environments for dev, QA, and prod

Abstract differences in OS distributions and underlying infrastructure

Higher utilization, compute density. & fast startup

Rapid scale-up and scale-down in response to changing business needs

**DevOps**

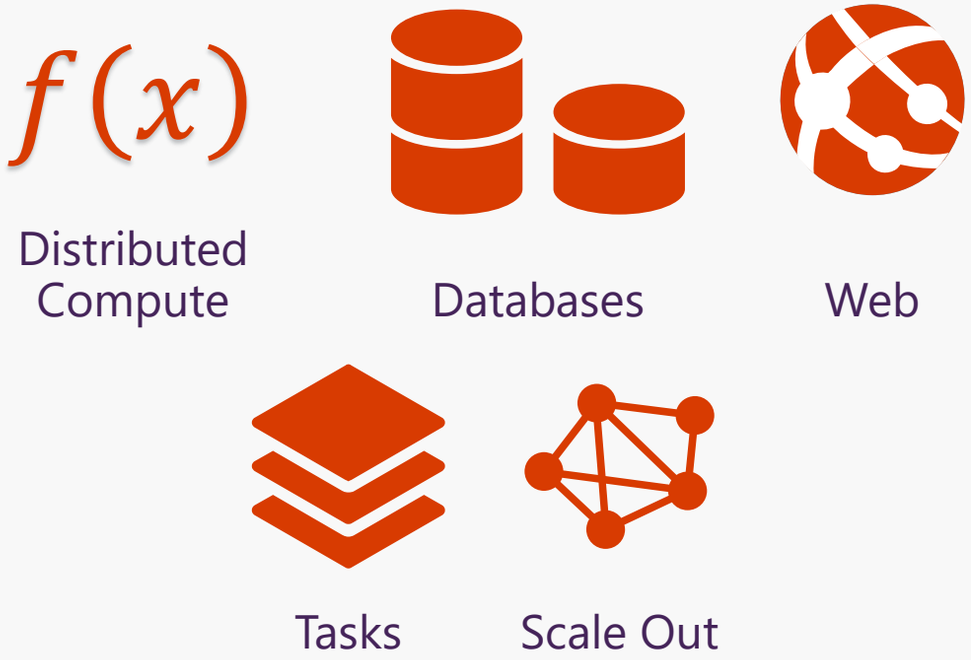
Integrate people, process, and tools for an optimized app development process

Operations focus on standardized infrastructure

Developers focus on building, deploying and testing apps

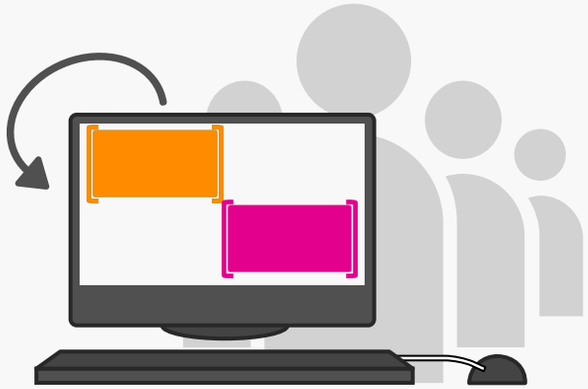
# Container use cases

- Workload Characteristics
  - Scale out
  - Distributed
  - State separated
  - Rapid (re)start
- Deployment Characteristics
  - Efficient hosting
  - Multitenancy
  - Rapid deployment
  - Highly automatable
  - Rapid scaling

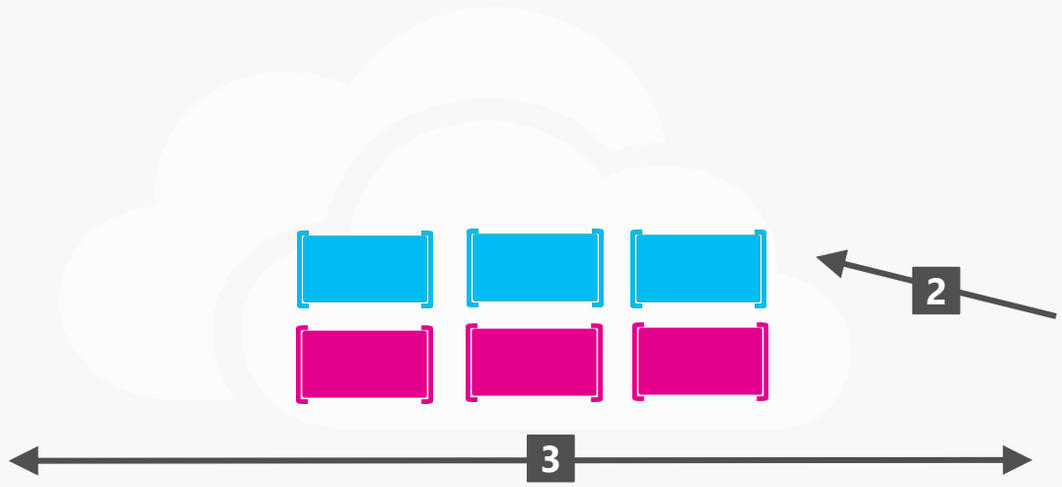


# DevOps Process with Containers

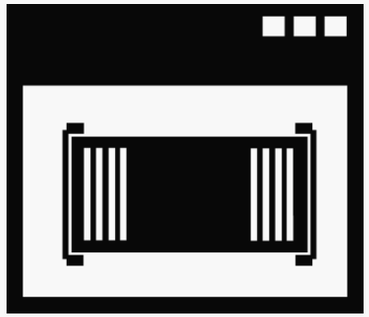
**Developers** update, iterate, and deploy updated containers

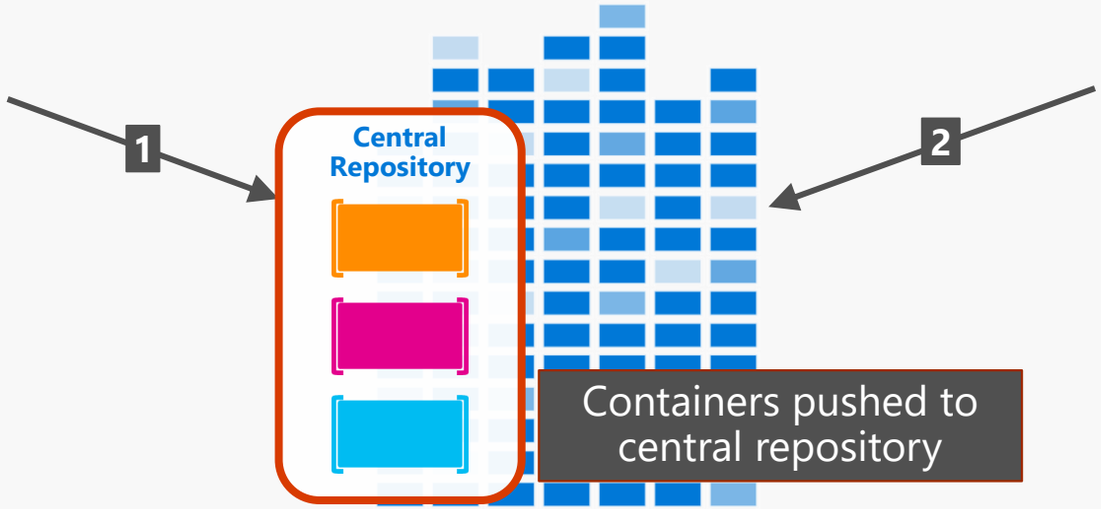


**Developers** build and test apps in containers, using development environment i.e. Visual Studio




**Operations** collaborates with **developers** to provide app metrics and insights






**Operations** automates deployment and monitors deployed apps from central repository

# Not Into DevOps?

- DNS / DHCP – Simple Services
- Websites
- SQL / MySQL / NoSQL / Postgres / SQLite

# But What About…

# The Desktop?

Docker
Rocket
Turbo.net
Virtuozzo

# Licensing

## Windows Server 2016 Licensing

| Windows Server 2016 feature differentiation and core-based pricing | | |
|---|---|---|
| Feature | Datacenter | Standard |
| Core functionality of Windows Server | • | • |
| OSEs / Hyper-V containers | Unlimited | 2 |
| Windows Server containers | Unlimited | Unlimited |
| Host Guardian Service | • | • |
| Nano Server* | • | • |
| Storage features including Storage Spaces Direct and Storage Replica | • | |
| Shielded Virtual Machines | • | |
| Networking stack | • | |
| Core-based pricing** | $6,155 | $882 |

*Software Assurance is required to deploy and operate Nano Server in production.
**Pricing for Open (NL) ERP license for 16 core licenses. Actual customer prices may vary.

*https://www.microsoft.com/en-us/cloud-platform/windows-server-pricing*

# Update Container OS Image

Pull updated base image

Rebuild containers using dockerfiles

# Update Container OS Image

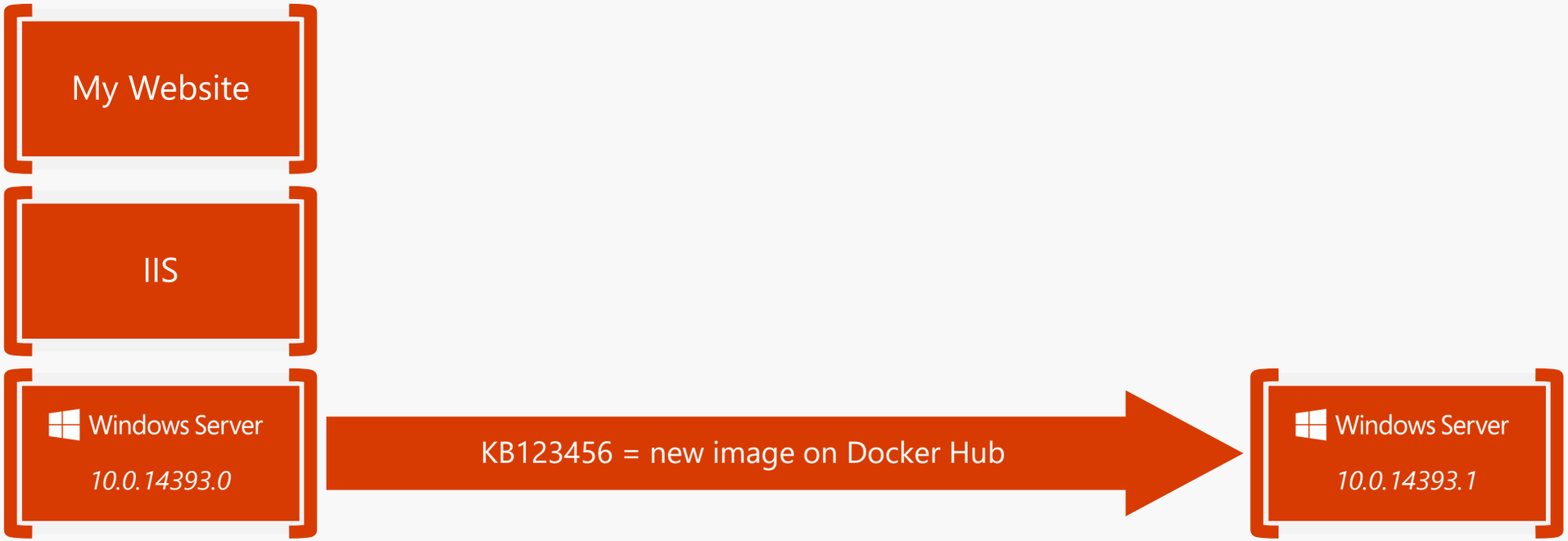
Pull updated base image

Rebuild containers using dockerfiles

# Update Container OS Image

Pull updated base image

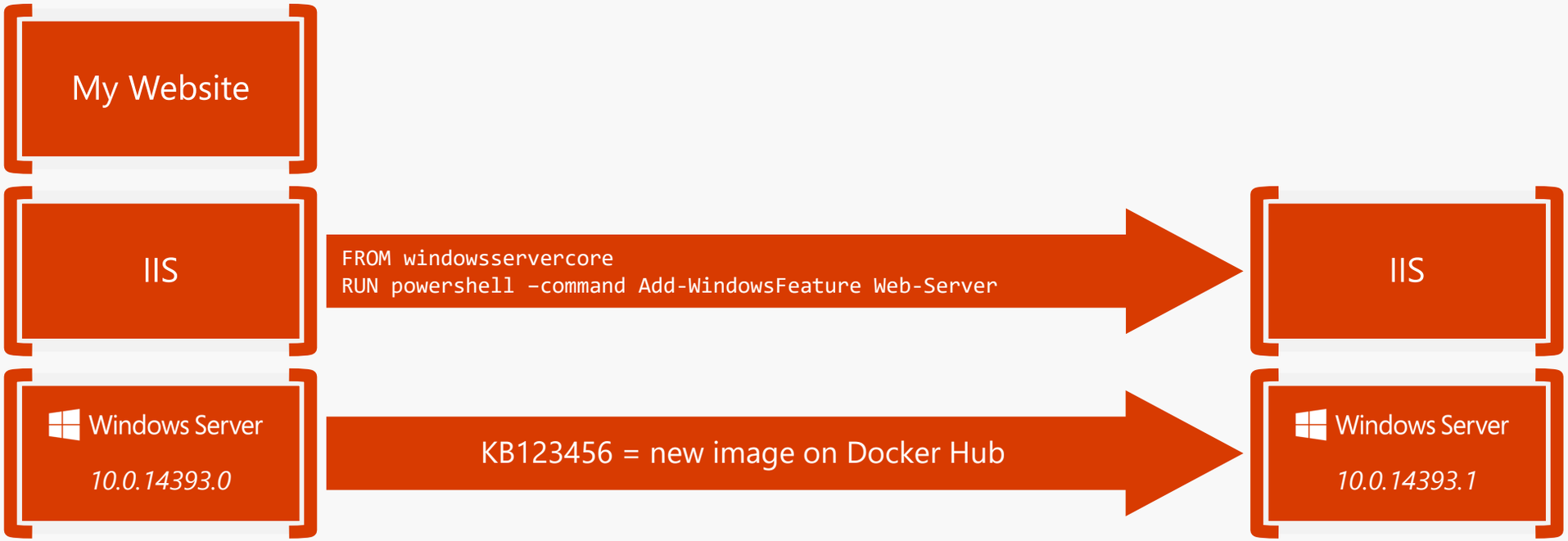
Rebuild containers using dockerfiles



# Update Container OS Image

Pull updated base image

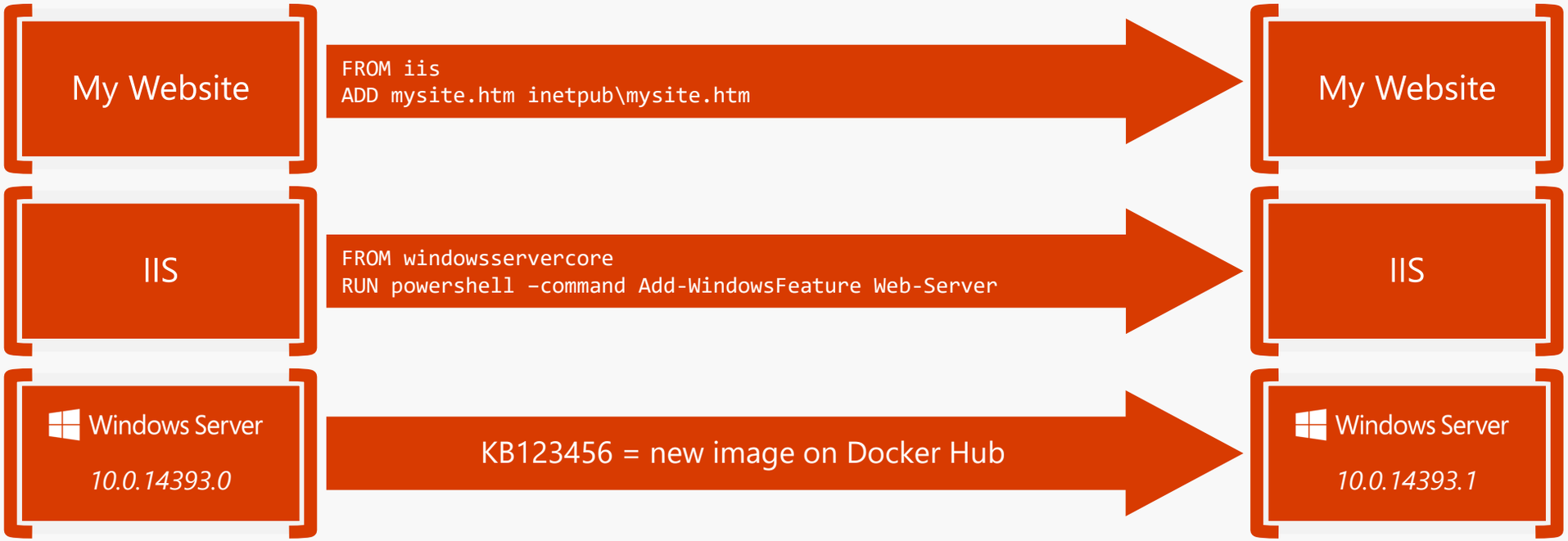
Rebuild containers using dockerfiles



# Update Container OS Image

Pull updated base image

Rebuild containers using dockerfiles



# Update as a new layer

Download update in container (ala run Windows Update in the container)

When container is stopped update is applied as a new layer

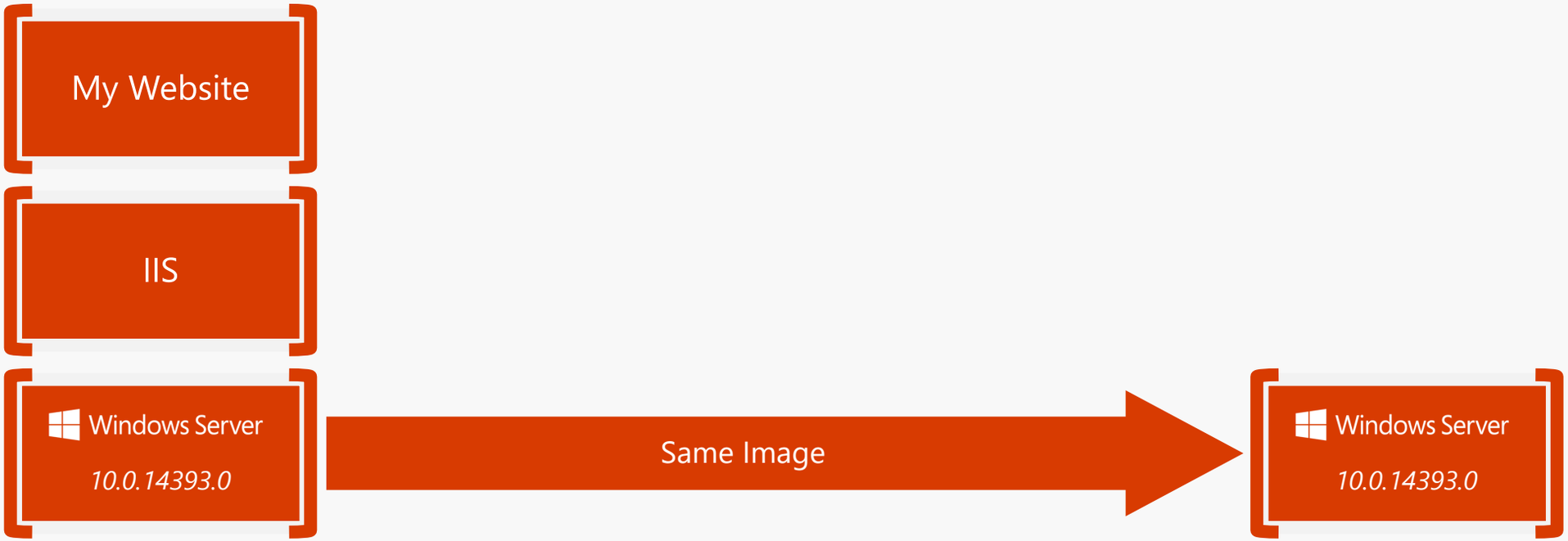
# Update as a new layer

Download update in container (ala run Windows Update in the container)

When container is stopped update is applied as a new layer

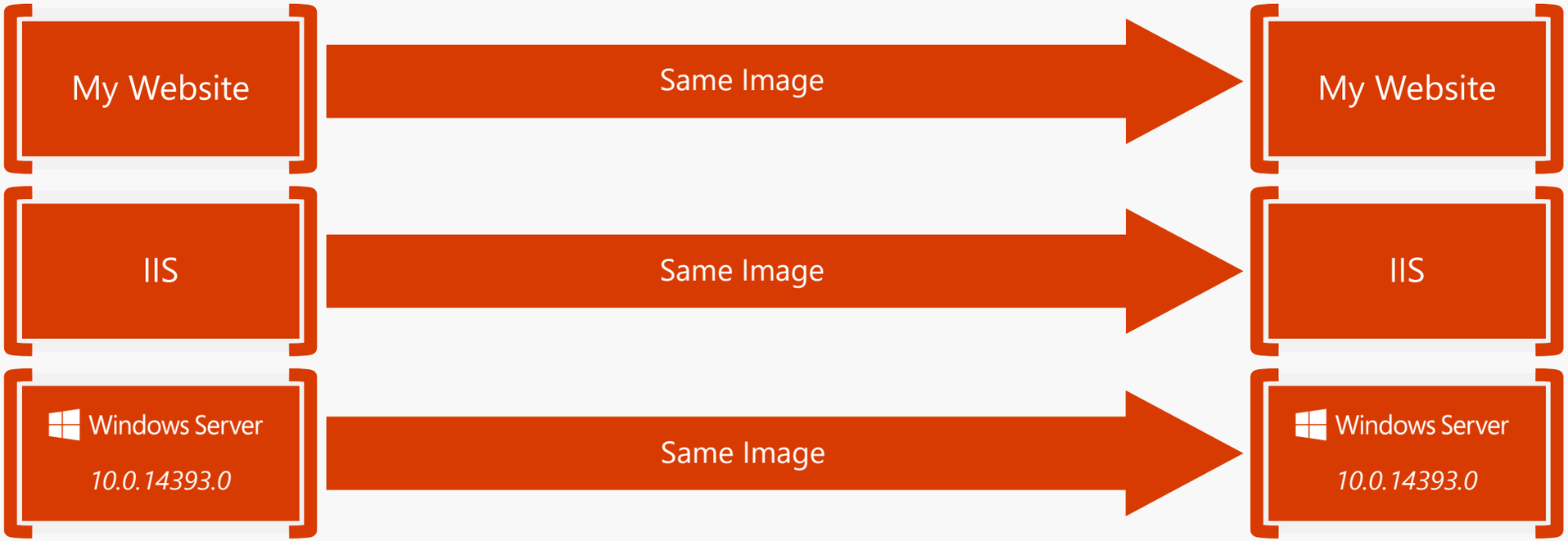
# Update as a new layer

Download update in container (ala run Windows Update in the container)
When container is stopped update is applied as a new layer



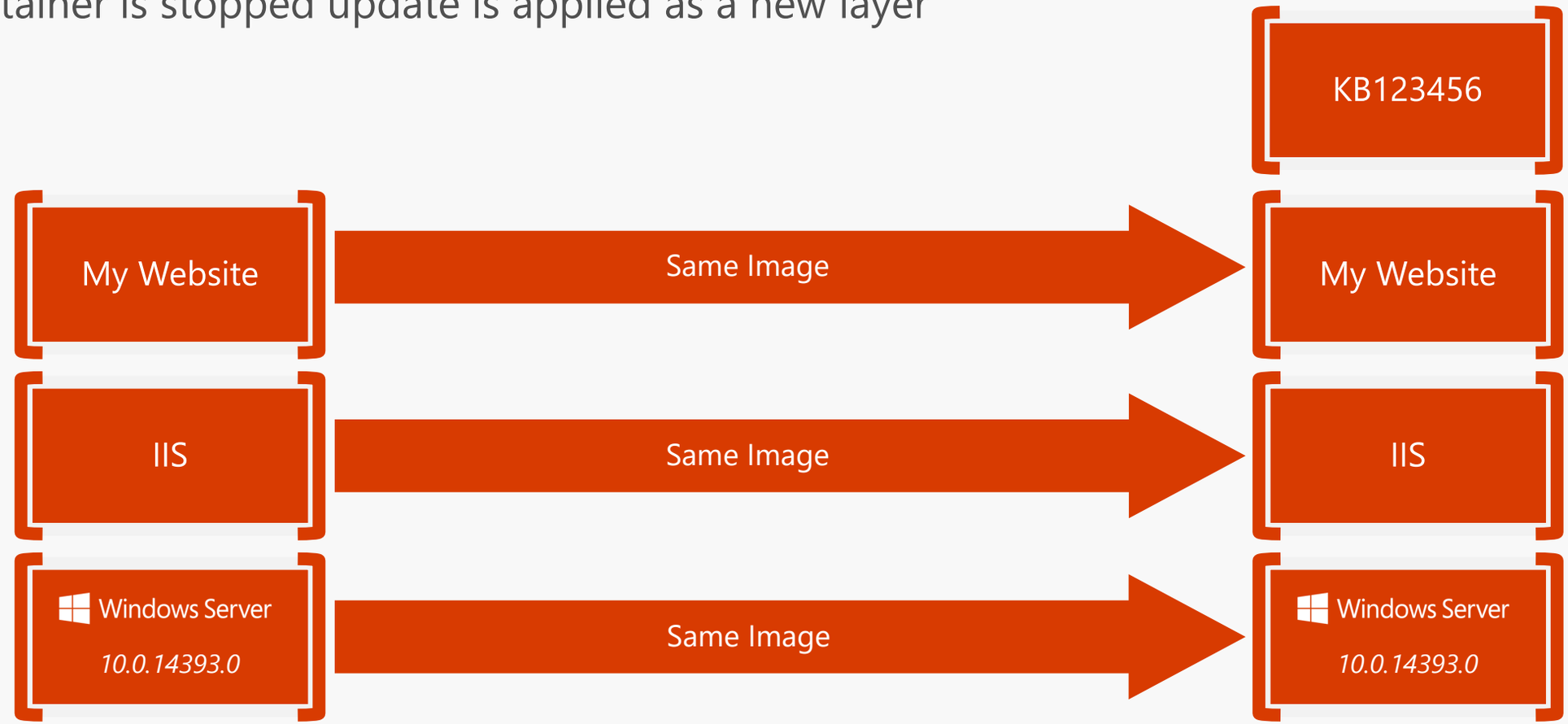
# Update as a new layer

Download update in container (ala run Windows Update in the container)
When container is stopped update is applied as a new layer



# Update as a new layer

Download update in container (ala run Windows Update in the container)
When container is stopped update is applied as a new layer



# Resource Controls

## CPU

Percent of host CPU container can consume

## Memory

Maximum memory container can consume

## Disk

Maximum IO bandwidth on the system drive
Maximum IOPs limit on the system drive

## Network

Platform support for egress caps

Networking

# Fully Manageable with Docker

Network creation/enumeration
Service Discovery

# Optimized for Microsoft Cloud Stack

Advanced network policy (ACLs, QoS) can be assigned per container endpoint
Load Balancing can be handled through the Microsoft Software Load Balancer (Coming Soon)

# Area of Regular and Continuous Innovation

Docker tooling support (Compose) for networking (Limited Support at GA)
Integration with other Orchestrators (Kubernetes, Swarm, etc.)
Native Overlay network driver
Multiple networks (NAT and overlay) per host

# Container Image

Not designed for persistent data
Not designed for secrets

# "Volumes"

Enables storage persistence
Enables mapping of storage into containers
Read-Only or Read/Write
Multiple containers on the same host can access the same location
Plug-In Architecture

# Network Storage

Containers access SMB shares
Accessed though the containers network

# Azure Container Service

Streamlined provisioning of Apache Mesos Clusters

VM Scale Sets

Layered support of Swarm, Marathon, and Chronos

Docker tooling and API support

Integrated app management and scaling

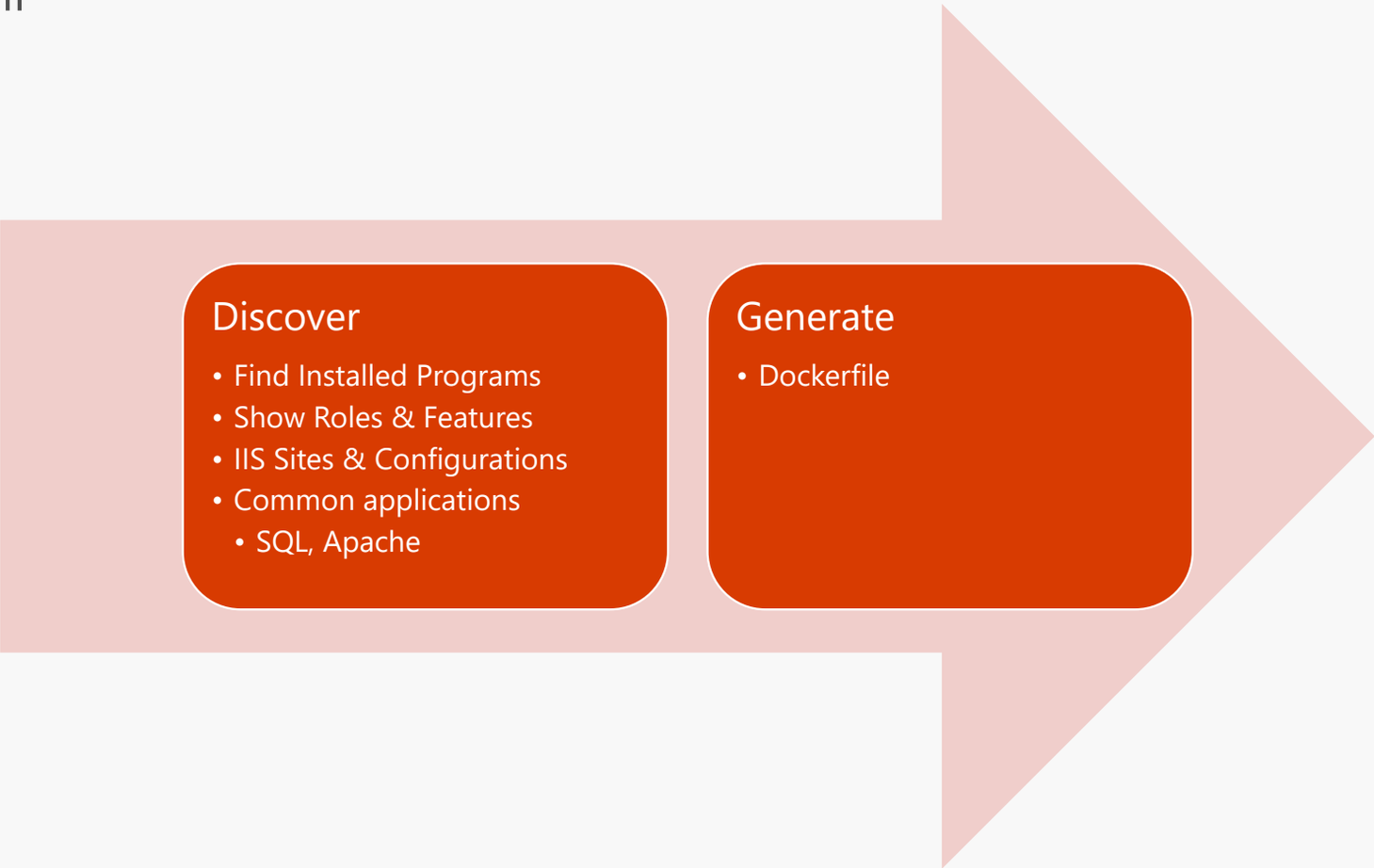
Linux and Windows Server containers

Public and private cloud

# Building a Container

# Image2Docker

Convert WIM/VHD to Dockerfile
Written in PowerShell
Open Source



# Container Tools

## Container Management

PowerShell

Docker

Others

## Service Fabric

Microservices

Hybrid Operations
Data Partitioning
Service Fabric
Health Monitoring
Container Orchestration & lifecycle management
High Density
Rolling Upgrades
Low Latency
Placement Constraints
Fast startup & shutdown
Load balancing
Hyper-Scaling
Actor programming model
Automated Rollback
Stateful services
High Availability
Self-healing
Auto Replication & Failover

## Container Technologies

Windows Server

Linux

## Development Environments

Visual Studio

eclipse

Others...

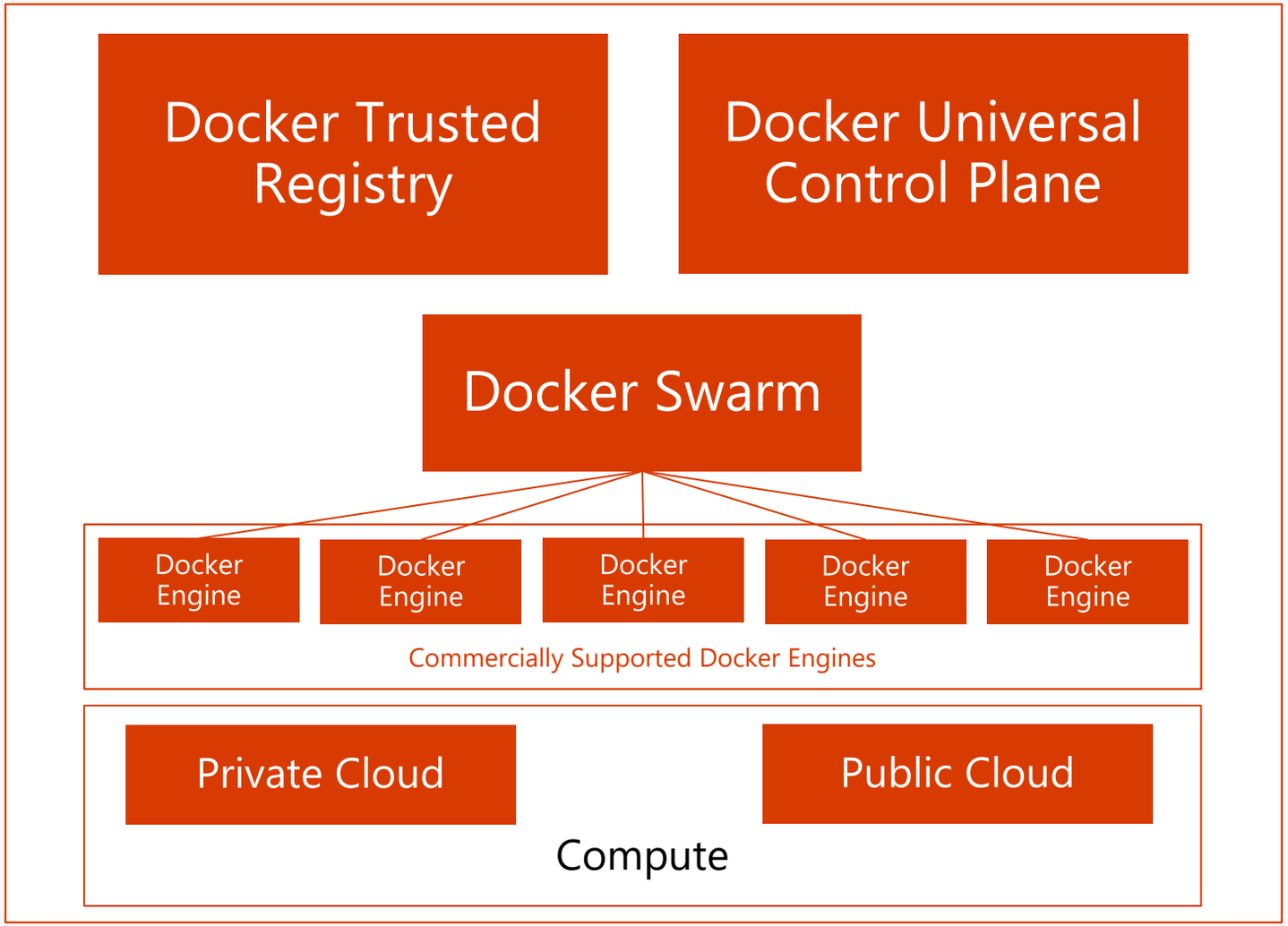
## Microsoft Cloud

Azure

On Premises

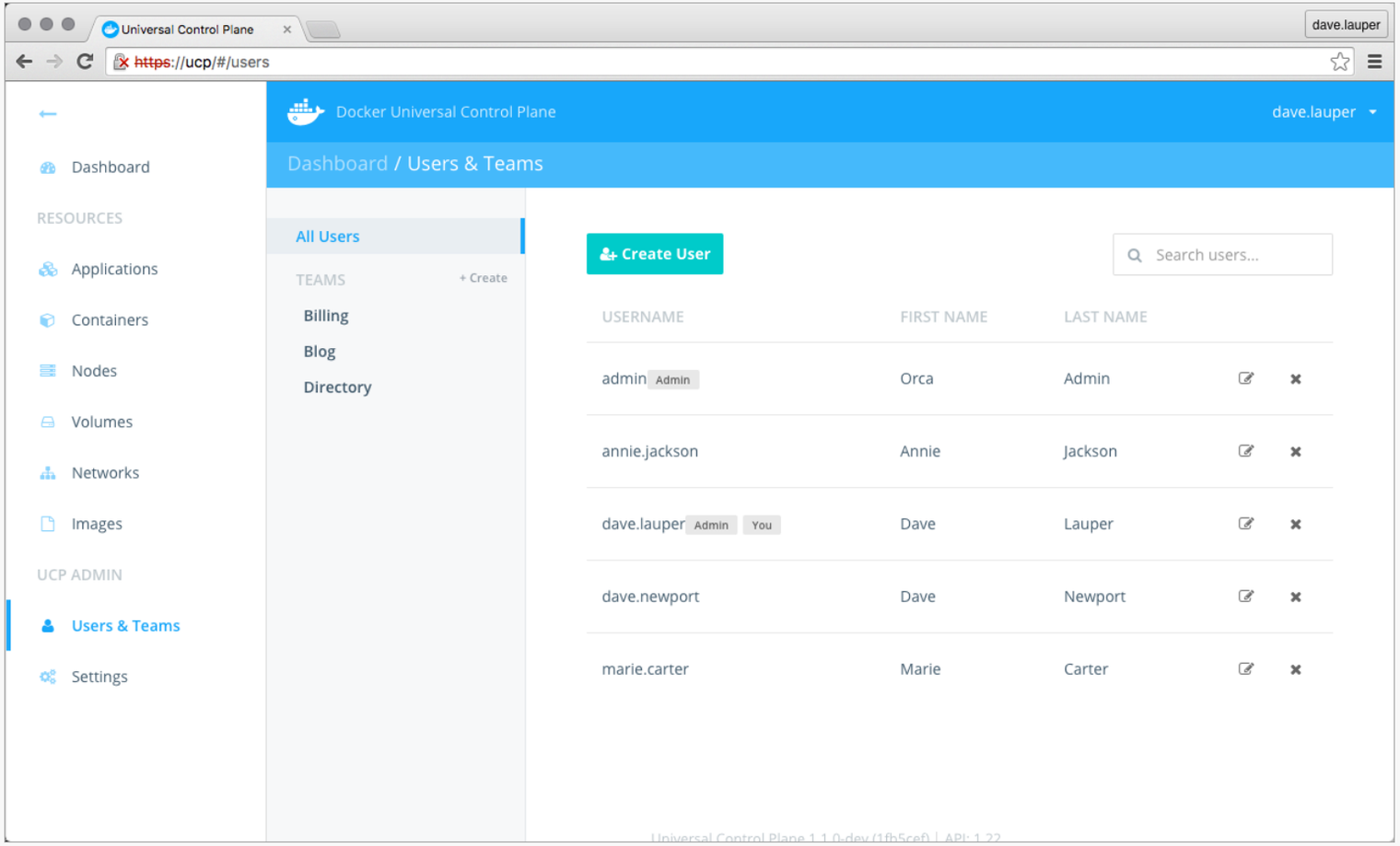
Service Provider

# Docker Datacenter



# Management and Monitoring Tools

## Docker Datacenter

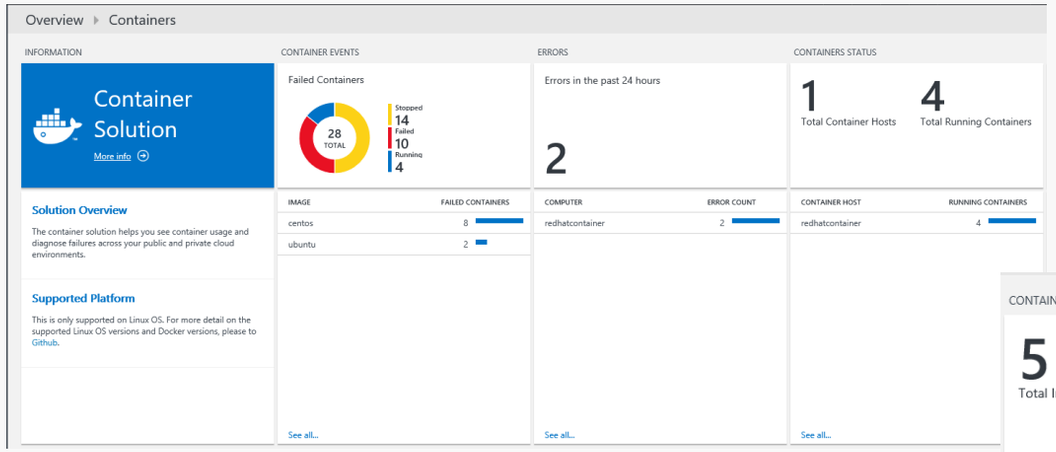


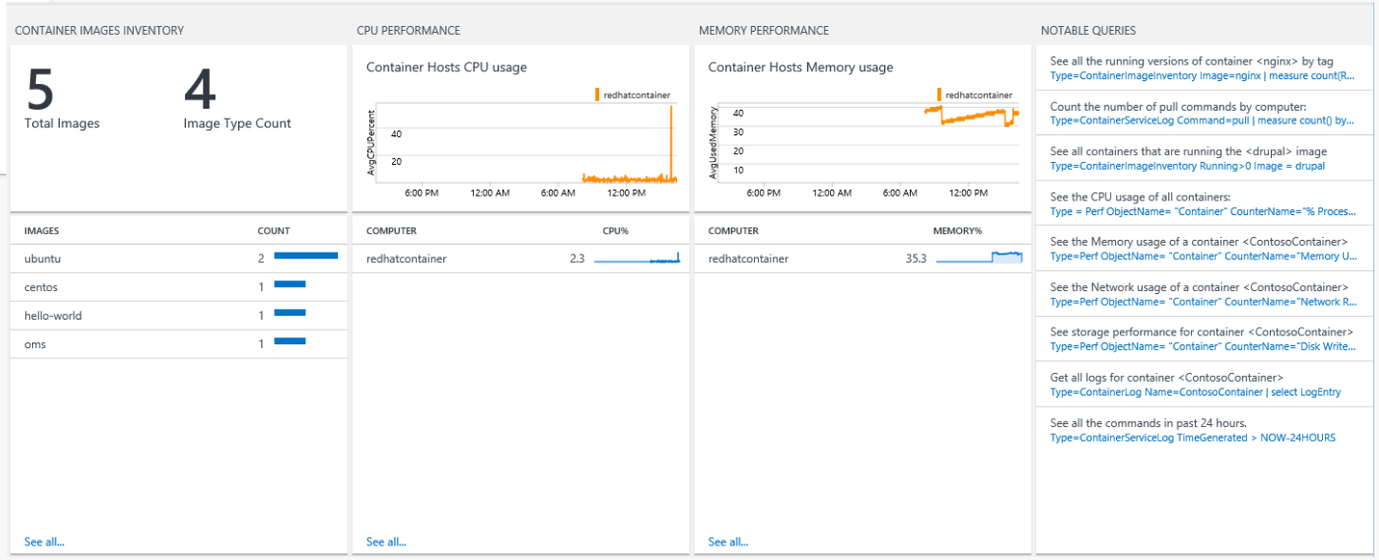
# Monitoring Solution for Docker

Syslog events

Performance metrics

Container data



# Build Once, Run Anywhere

*http://aka.ms/containers*

# Thank you!

# Demo's & Labs

Microsoft

# Lab 1

## Installing Containers

# Installing the container service

1. Open PowerShell in administrative mode
2. Install Container Service
   1. Install-windowsfeature containers
3. Restart server
   1. Restart-Computer –force

# Install Docker

1. Open PowerShell in administrative mode
2. Create directory for Docker
   1. New-item –type directory –path “c:\program files\docker\”
3. Download Docker Daemon
   1. Invoke-WebRequest https://aka.ms/tp5/b/dockerd -OutFile $env:ProgramFiles\docker\dockerd.exe -UseBasicParsing
4. Download Docker Client
   1. Invoke-WebRequest https://aka.ms/tp5/b/docker -OutFile $env:ProgramFiles\docker\docker.exe –UseBasicParsing
5. Create environment variable for Docker
   1. [Environment]::SetEnvironmentVariable("Path", $env:Path + ";C:\Program Files\Docker", [EnvironmentVariableTarget]::Machine)
6. Register service
   1. dockerd --register-service
7. Start service
   1. Start-Service docker

# 1. Update container modules for PowerShell

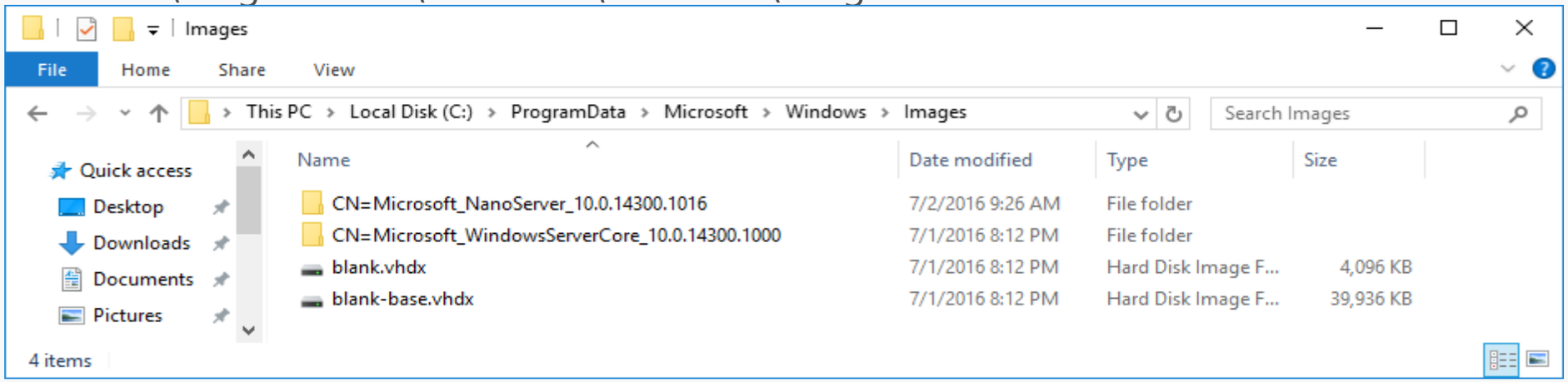
1. http://github.com/microsoft/Docker-PowerShell
    1. Register-PSRepository -Name DockerPS-Dev -SourceLocation https://ci.appveyor.com/nuget/docker-powershell-dev
    2. Install-Module Docker -Repository DockerPS-Dev
    3. Update-Module Docker

# Install base images

1. Open PowerShell in administrative mode
2. Install package provider
   1. Install-PackageProvider ContainerImage –Force
3. Method 1 – download and install images via Internet
   1. Install-ContainerImage -Name WindowsServerCore
   2. Install-ContainerImage -Name NanoServer
4. Method 2 – download and install later
   1. Find-ContainerImage
   2. Save-ContainerImage -Name NanoServer -Path c:\container-image
   3. Install-ContainerOSImage -WimPath C:\container-image\NanoServer.wim –Force
5. Restart Docker after image installation
   1. Restart-Service docker

# Verify installed images

- Open Explorer
  - C:\ProgramData\Microsoft\Windows\images




- Visualize using PowerShell or Docker

```
PS C:\Windows\system32> get-containerimage

Name              Publisher    Version         IsOSImage
----              ---------    -------         ---------
NanoServer        CN=Microsoft 10.0.14300.1016 True
WindowsServerCore CN=Microsoft 10.0.14300.1000 True


PS C:\Windows\system32> docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
windowsservercore   latest              5bc36a335344        7 weeks ago         9.354 GB
nanoserver          latest              3f5112ddd185        7 weeks ago         810.2 MB
microsoft/iis       windowsservercore   c26f4ceb81db        8 weeks ago         9.49 GB
PS C:\Windows\system32>
```

# Prepare images

- Add TAG for image to be used
  - docker tag windowsservercore:10.0.14300.1000 windowsservercore:latest
  - docker tag windowsservercore:10.0.14300.1016 nanoserver:latest
- Visualize existent images
  - Docker images
  - Get-containerimage

# Deploy Containers

1. Open PowerShell in administrative mode
    1. Docker search Microsoft
    2. OR
    3. Docker search *
2. Downloading container
    1. docker pull microsoft/iis:windowsservercore
3. Start container in background (-d), map port to host and container (-p) and execute PING to remain the container running
    1. docker run -d -p 80:80 microsoft/iis:windowsservercore ping -t localhost
4. Visualize container in execution (ps)
    1. Docker ps

# Visualize Web Site

1. Verify wich IP Address used by host
   1. Ipconfig
2. Use another computer on your network e type the IP Address on browser (step 1)

# Shutdown container

1. Visualize container in execution and record the name

```
PS C:\Windows\system32> docker ps
CONTAINER ID        IMAGE                           COMMAND              CREATED             STATUS              PORT
S                   NAMES
f33158537e11        microsoft/iis:windowsservercore   "ping -t localhost"   18 seconds ago      Up 13 seconds       0.0.
0.0:80->80/tcp   fervent_wright
PS C:\Windows\system32>
```

2. Remove container (rm) forcibly (-f)
   1. Docker rm –f fervent_wright
3. Visualize active containers
   1. Docker ps

# Execute containers “on demand”

1. Open PowerShell in Administrative Mode
2. Start container in interactive mode (-it), map host  port to container (-p) and visualize the command
   1. docker run -it -p 80:80 microsoft/iis:windowsservercore ping -t localhost
3. Open another PowerShell in Administrative Mode and visualize containers
   1. Docker ps
4. Return to the 1st PowerShell window (w/ PING command) and shutdown the container
   1. CTRL + C
5. Return to the 2nd PowerShell window (w/ Docker PS command) and execute again

# Edit content inside the container

1. Open PowerShell in administrative mode
2. Create new container and execute CMD
   1. docker run -it -p 80:80 microsoft/iis:windowsservercore cmd
3. Delete default page from IIS
   1. del C:\inetpub\wwwroot\iisstart.htm
4. Create new page
   1. echo “Teste do @fabiohara" > C:\inetpub\wwwroot\index.html
5. Closing container session
   1. Exit

# Edit content inside the container (cont…)

1. Verify the name of the container
   1. Docker ps –a
2. Create new container image (commit)
   1. Docker commit nome_container modificado-iis
3. Visualize images created
   1. Docker images

Microsoft

# Optional Lab

## Dockerfile

Syntax (example)

# 1. Iinform the container image to be used

1. FROM windowsservercore

# 2. Execute command (exec form or shell form)

1. RUN [“powershell”, “New-item”, “c:\\teste”]
2. OR
3. RUN powershell new-item c:\teste
4. RUN dism.exe /online /enable-feature /all /featurename:iis-webserver /norestart
5. RUN powershell.exe –command c:\vcredist_x86.exe /quiet

# 3. Copy data

1. COPY teste1.txt c:/temp/
2. COPY diretorio1 /sqldemo/

# 4. ADD (similar to COPY)

1. ADD teste2.txt c:/temp/
2. ADD https://www.python.org/ftp/python/3.5.1/python-3.5.1.exe /temp/python-3.5.1.exe

# Sintax (example)(cont…)

1. Work directory
   1. WORKDIR c:\\windows
   2. WORKDIR c:\\app1\\dev
2. Default command to be executed during deployment of the container (exec form OR shell form)
   1. CMD [“c:\\app1\\dev\\httpd.exe”, “-x”]
   2. OR
   3. CMD c:\\app1\\dev\\httpd.exe –x

# Sintax (examples)(cont…)

## 1. Command lines to be executed via ESCAPE (character \ or `)

```
RUN powershell.exe -Command \
    $ErrorActionPreference = 'Stop'; \
    wget https://www.python.org/ftp/python/3.5.1/python-3.5.1.exe -OutFile c:\python-3.5.1.exe ; \
    Start-Process c:\python-3.5.1.exe -ArgumentList '/quiet InstallAllUsers=1 PrependPath=1' -Wait ; \
    Remove-Item c:\python-3.5.1.exe -Force
```

## 2. Using character `

```
FROM windowsservercore
RUN powershell.exe -Command `
    $ErrorActionPreference = 'Stop'; `
    wget https://www.python.org/ftp/python/3.5.1/python-3.5.1.exe -OutFile c:\python-3.5.1.exe ; `
    Start-Process c:\python-3.5.1.exe -ArgumentList '/quiet InstallAllUsers=1 PrependPath=1' -Wait ; `
    Remove-Item c:\python-3.5.1.exe -Force
```

Syntax (examples)(cont…)

# 1. PowerShell commands on Dockerfile

1. FROM windowsservercore
2. RUN powershell -command Expand-Archive -Path c:\apache.zip -DestinationPath c:\

# 2. REST Call

1. FROM windowsservercore
2. RUN powershell.exe -Command \
3. $ErrorActionPreference = 'Stop'; \
4. Invoke-WebRequest https://www.python.org/ftp/python/3.5.1/python-3.5.1.exe -OutFile c:\python\python-3.5.1.exe ; \
5. Start-Process c:\python\python-3.5.1.exe -ArgumentList '/quiet InstallAllUsers=1 PrependPath=1' -Wait ; \
6. Remove-Item c:\python\python-3.5.1.exe -Force

# Syntax (examples)(cont…)

## 1. PowerShell command on Dockerfile

```
1. FROM windowsservercore
2. RUN powershell -command Expand-Archive -Path c:\apache.zip -DestinationPath c:\
```

## 2. REST call (using invoke-webrequest)

```
1. FROM windowsservercore
2. RUN powershell.exe -Command \
3.   $ErrorActionPreference = 'Stop'; \
4.   Invoke-WebRequest https://www.python.org/ftp/python/3.5.1/python-3.5.1.exe -OutFile c:\python\python-3.5.1.exe ; \
5.   Start-Process c:\python\python-3.5.1.exe -ArgumentList '/quiet InstallAllUsers=1 PrependPath=1' -Wait ; \
6.   Remove-Item c:\python\python-3.5.1.exe –Force
```

## 3. REST call (using .Net library - WebClient)

```
1. FROM windowsservercore
2. RUN powershell.exe -Command \
3.   $ErrorActionPreference = 'Stop'; \
4.   (New-Object System.Net.WebClient).DownloadFile('https://www.python.org/ftp/python/3.5.1/python-3.5.1.exe','c:\python\python-3.5.1.exe') ; \
5.   Start-Process c:\python\python-3.5.1.exe -ArgumentList '/quiet InstallAllUsers=1 PrependPath=1' -Wait ; \
6.   Remove-Item c:\python\python-3.5.1.exe -Force
```

# 1. PowerShell script on Dockerfile

1. FROM windowsservercore
2. ADD script.ps1 /windows/temp/script.ps1
3. RUN powershell.exe -executionpolicy bypass c:\windows\temp\script.ps1

# Creating images via Dockerfile

1. Create directory to store Dockerfile
   1. Powershell new-item c:\build\dockerfile –force
2. Edit Dockerfile with Notepad
   1. Notepad c:\build\dockerfile
3. Create new image (using microsoft/iis:windowsservercore) and execute command
   1. Copy and paste the following text
      1. FROM microsoft/iis:windowsservercore
      2. RUN echo "Hello World – teste de Dockerfile" > c:\inetpub\wwwroot\index.html
   2. Save and close notepad
4. Create new image from Dockerfile
   1. CD build
   2. Docker build –t teste2 .
   3. OR
   4. Docker build –t iis-dockerfile c:\build

# Creating images via Dockerfile (cont…)

- Visualize available images
  - Docker images
- Vsualize image dependency
  - Docker history name_image
- Deploy new container based on image
  - docker run -d -p 80:80 iis-dockerfile ping -t localhost
- Test: verify the host IP and open browser in another machine using this IP
- Verify the name of the container in execution
  - Docker ps
- Remove container (rm)
  - Docker rm –f nome_container
- Extra: remove image (rmi)
  - Docker rmi –f id_do_container

Microsoft

# Lab 2

## Installing images via Docker Hub

# Downloading images from Docker Hub

1. Open PowerShell in administrative mode and list available images
   1. Find-containerimage
2. List images from Docker Hub
   1. Docker search *
3. Download image with Apache for Windows Server Core
   1. Docker pull microsoft/sample-httpd:windowsservercore
4. Download image with Apache for Nano Server
   1. Docker pull microsoft/sample-httpd:nanoserver
5. List local images
   1. Docker images

Microsoft

# Optional Lab

## Using VM as container Host

# Enable Hyper-V Container in Hyper-V VM

1. Install VM with Windows Server 2016 TP5 and shutdown
2. Execute the following commands (Host)

```
#Name of the VM
$vm = "<virtual-machine>"
#configure the virtual processor
Set-VMProcessor -VMName $vm -ExposeVirtualizationExtensions $true -Count 2
#disable dynamic memory
Set-VMMemory $vm -DynamicMemoryEnabled $false
#enable mac spoofing
Get-VMNetworkAdapter -VMName $vm | Set-VMNetworkAdapter -MacAddressSpoofing On
```

Microsoft

# Lab 3

## Deploy Windows Containers and Hyper-V Containers

# Prepare the Host

1. Verify if the host support virtualization
    1. Mínimum 4GB RAM for labs
    2. Windows Server 2016 TP5 or Windows 10 build 10565 (or superior)
    3. Processor supporting Intel-VT or AMD-V
        1. Msinfo32 to verify the processor support virtualization
2. Install Hyper-V role
    1. Install-windowsfeature hyper-v
3. Restart server
    1. Restart-computer -force

# (Optional) Enable Hyper-V Container inside Hyper-V VM

1. Install the VM with Windows Server 2016 TP5 and shutdown
2. Execute the following commands (Host)

```
#name of the VM
$vm = "<virtual-machine>"
#configure virtual processor
Set-VMProcessor -VMName $vm -ExposeVirtualizationExtensions $true -Count 2
#disable dynamic memory
Set-VMMemory $vm -DynamicMemoryEnabled $false
#enable mac spoofing
Get-VMNetworkAdapter -VMName $vm | Set-VMNetworkAdapter -MacAddressSpoofing On
```

# Using Nano Server VM to host Containers

1. Copy ISSO file of Windows Server 2016 TP5 locally and mount volume
   1. Double click on ISSO file and access the directory \NanoServer
   2. Copy the directory NanoServerImageGenerator to c:\Nano
2. Importe module
   1. Import-module .\NanoServerImageGenerator -verbose
3. Open PowerShell in administrative mode and create Nano Server image
   1. New-NanoServerImage -Edition Standard -DeploymentType Guest -MediaPath e:\ -TargetPath .\Nano\Nano.vhd -ComputerName Nano -containers -storage

# Create Hyper-V VM for Nano

1. Create directory to store the VM (eg: c:\VM\Nano)
2. Copy the file Nano.VHD and paste on c:\vm\nano
3. Create VM with the name Nano and using the existent VHD (Nano.VHD)

# Create Hyper-V container

1. Isolated mode (do not execute)
    1. Docker run –it --isolation=hyperv nanoserver cmd
    2. Powershell prompt will open CMD
2. Regular mode
    1. Docker run –d windowsservercore ping localhost –t
    2. Copy ID
3. Visualize ping process via Docker
    1. Docker top “ID_copy”
    2. You can see the Process ID and the command (ping)
4. Visualize ping process via Windows
    1. Get-process –name ping
    2. You can see the same process ID above

# Create Hyper-V container (cont…)

1. Start container in isolated mode
   1. docker run -d --isolation=hyperv nanoserver ping -t localhost
   2. Copy ID
2. Visualize ping process via Docker
   1. Docker top “ID_copy”
3. Visualize process via Windows
   1. Get-process –name ping
   2. The process ID will not be shown in “docker top” command
4. Visualize the isolated container
   1. Get-process –name vmwp or via Task Manager

Microsoft

# Lab 4

## Managing data

# Create data volumes

1. Execute container and create volume
   1. Docker run –it –v c:\new-data-volume windowsservercore cmd
2. Create file on c:\new-data-volume
   1. Echo “teste de volume de dados” > c:\novo-volume-dados\testevolume1.txt
   2. Dir
   3. Exit
   4. Visualize volume on c:\ProgramData\docker\volumes

# Mount data volume

1. Create folder on Host and copy files
   1. Create folder c:\origem
   2. Copy small files and paste inside this folder
2. Execute container mounting existing data volume on Host (c:\origem) to inside this container (c:\destino)
   1. Docker run –it –v c:\origem:c:\destino windowservercore cmd
   2. Command Prompt will be open
3. Visualize content
   1. Using Command Prompt: enter on this folder c:\destino and visualize content (DIR)

# Mounting single file

## 1. Create folder and test file on host

1. New-item -type directory –path c:\origem2
2. Echo “test with single file1” > c:\origem2\teste1.txt
3. Echo “test with single file2” > c:\origem2\teste2.txt
4. Echo “test with single file3” > c:\origem2\teste3.txt

## 2. Execute container and map file

1. Docker run –it –v c:\origem2\teste1.txt windowsservercore cmd

## 3. Visualize the directory content inside the container

1. Cd origem2
2. Dir

## 4. Inspect shared volume

1. Docker inspect nome_container

Microsoft

# Optional Lab

## Managing resources

# Managing resources

1. Controlling CPU resources (1-10000, default 5000) for 2 virtual processors
   1. Docker run –it --cpu-shares 2 –name dockerdemo windowsservercore cmd

OBS: CPU and IO still not yet supported in Hyper-V Containers
IO still not yet supported using data volume

Microsoft

# Optional Lab

Nano VM using Hyper-V containers
OBS: installing new images are still not available (TBD)

# Optional –Nano VM with Hyper-V containers

1. Also know as Nested Virtualization
2. If you want to create hyper-v containers inside Nano VM just execute the following PowerShell script:

```
1. #name of the VM
2. $vm = "<virtual-machine>"
3. #configure the virtual processor
4. Set-VMProcessor -VMName $vm -ExposeVirtualizationExtensions $true -Count 2
5. #disable dynamic memory
6. Set-VMMemory $vm -DynamicMemoryEnabled $false
7. #enable mac spoofing
8. Get-VMNetworkAdapter -VMName $vm | Set-VMNetworkAdapter -MacAddressSpoofing On
```

# Enabling Firewall rules on Nano Server

1. Logon on Nano Server console and access “Inbound Firewall Rules”
2. Enable each rule of “File and Printer Sharing”
    1. Press F4 to enable
    2. Status will be “Allow”
3. Optional – enable remote management via WinRM
    1. Select option “Windows remote Management” e press ENTER twice
4. Verify the IP address used by Nano
    1. Select “Networking” and select “Ethernet Adapter”

# Install Docker on Nano Server VM

- Nano Server TP5 not support Invoke-WebRequest
- Download locally (eg: via Host) Docker Daemon and Client
  - Invoke-WebRequest https://aka.ms/tp5/b/dockerd -OutFile .\dockerd.exe
  - Invoke-WebRequest https://aka.ms/tp5/b/docker -OutFile .\docker.exe
- Remote connect on Nano Server VM
  - Open PowerShell in administrative mode and type the following command:

```
PS C:\software\NanoServerImageGenerator> enter-pssession -vmname nano

cmdlet Enter-PSSession at command pipeline position 1
Supply values for the following parameters:
Credential
[NANO]: PS C:\Users\administrator\Documents>
```

- Create folder C:\Docker, share and copy the content downloaded
  - New-item –type directory c:\docker
  - New-SmbShare –name DOCKER –path c:\docker
  - Grant-SmbShareAccess –name DOCKER –AccountName Administrators –AccessRight Full -force

# Install Docker on Nano Server VM(cont..)

- Register Docker Service on Windows
  - & “c:\docker\dockerd.exe” --register-service
- Inicialize Docker service
  - Start-service docker
- Install Package Provider for Windows Containers
  - Install-packageprovider ContainerImage –force –verbose
- Download and install base image of Nano Server
  - Install-containerimage –name NanoServer –version 10.0.14300.1010
  - **Obs: command is not yet available for Nano**

Microsoft

# Optional Lab

## Configuring Docker

# Configuring Docker on Windows

- Daemon.json
  - C:\ProgramData\docker\config\daemon.json
  - If the file does not exist then you can create
  - Obs: not all parameters on daemon.json are valid on Docker for Windows
- Enable connection to Docker daemon on TCP 2375
  - Option 1 – daemon.json
    - "hosts": ["tcp://0.0.0.0:2375", "npipe://"]
  - Option 2 – SC Config (via CMD.exe)
    - sc config docker binpath= "\"C:\docker\dockerd.exe\" --run-service -H tcp://0.0.0.0:2375"

Microsoft

# Optional Lab

## Remove Images

# Remove installed images

- Powershell
  - Uninstall-ContainerOSImage –fullname CN=Microsoft_NanoServer_10.0.14300.1016
- Docker
  - Remove images
    - Docker rmi windowsservercoreiis
  - Remove containers
    - Docker rm

# Ready for Enterprise Applications

- No infrastructure changes to Active Directory required
- Leverages Group Managed Service Accounts
- Enables containerized applications to authenticate with SQL Server, file shares, and more

# Simple and Secure

- Identity specified at container start time
- Credentials are not stored within the container image
- No changes required to move across environments i.e. dev, test, prod, cloud

# Breaking down larger applications

## Development

Makes development teams faster
Decoupled development, less coordination
Freedom to use the best tools for the job
Shorter builds & faster deployment

## Operations

Deploy faster with automation
Reliable and deterministic
Clear list of running containers
Less risk, less downtime

# Our Journey To Nano Server – Docker CI

## What is it?

Part of the Docker Project

For Every Code Submission

- Compiles the change for both Linux and Windows
- Uses Jenkins to execute tests against Linux and Windows hosts

## Windows Server Core Image

Azure D3 VM (4 cores, 14GB RAM, 200GB SSD)

Takes ~31 min and 21 sec to run

## Nano Image

Early prototype (not checked in yet)

Azure D2 VM (2 cores, 7GB, 100GB SSD)

Takes ~21min and 39 sec to run

Benefit of Nano

**30%** Reduction in Execution Time!

**50%** Reduction in Cost

Let's Get Moving

# Start your journey

Learn by doing

Many apps require few to no changes

# Take advantage of opportunities

Start breaking larger applications down into microservices

Achieve the benefits DevOps delivers

# Have Fun!

Quick start guide at http://aka.ms/containers