OPEN NETWORKING
SUMMIT 2016
MARCH 14-17, 2016 | SANTA CLARA, CA

# Power-up Networking for Containers

Jason Messer, *Microsoft*

# Focus and Key Takeaways

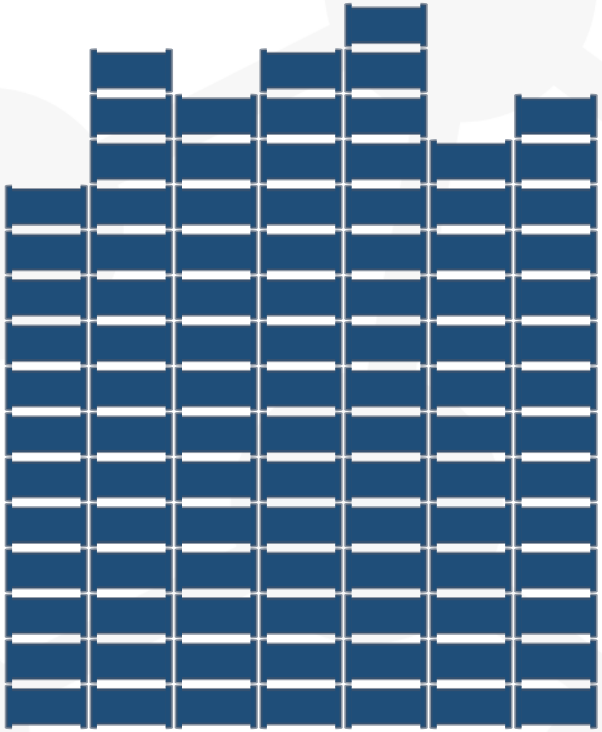
- Microsoft is active in and engaged with the FOSS community
- Microsoft understands DevOps and Microservice architectures
- Developers and IT Pros have a different set of concerns
  - Developers want to be agile in development and innovation
  - IT Pros want to provide security and flexibility in deployments
- With Windows Server 2016 and Microsoft Azure Stack (MAS), Software-Defined Networking stacks are improving these experiences and addressing concerns to create synergies between Developers and IT Pros
- Containers and Microsoft Software Defined Networking (SDN) enable “Write once, run anywhere”

# Microsoft FOSS Investments

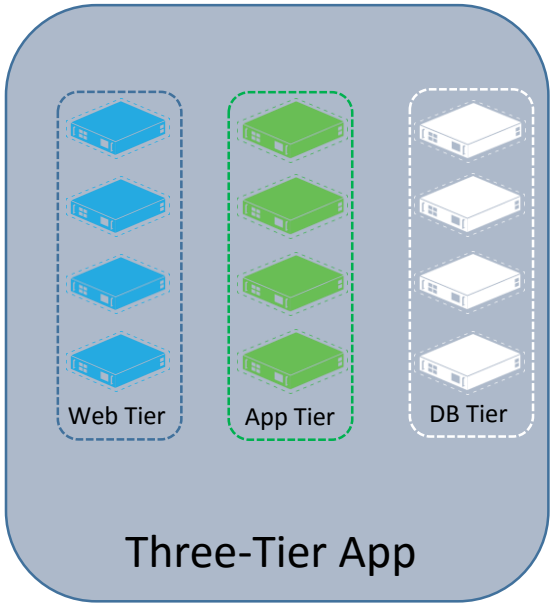
- Engaged in over a dozen open source projects and standards groups
- Employees are in leadership roles in the Open Source community
  - President of the Apache Software Foundation
  - Co-Chair of the W3C HTML5 Working Group
- Released key projects as open source (e.g. .NET, VS code, etc.)
- Employees are among top contributors to open source (Docker)
- Co-Founder with Docker for Open Container Initiative (OCI)
- AllJoy / AllSeen Alliance (IoT home networking)

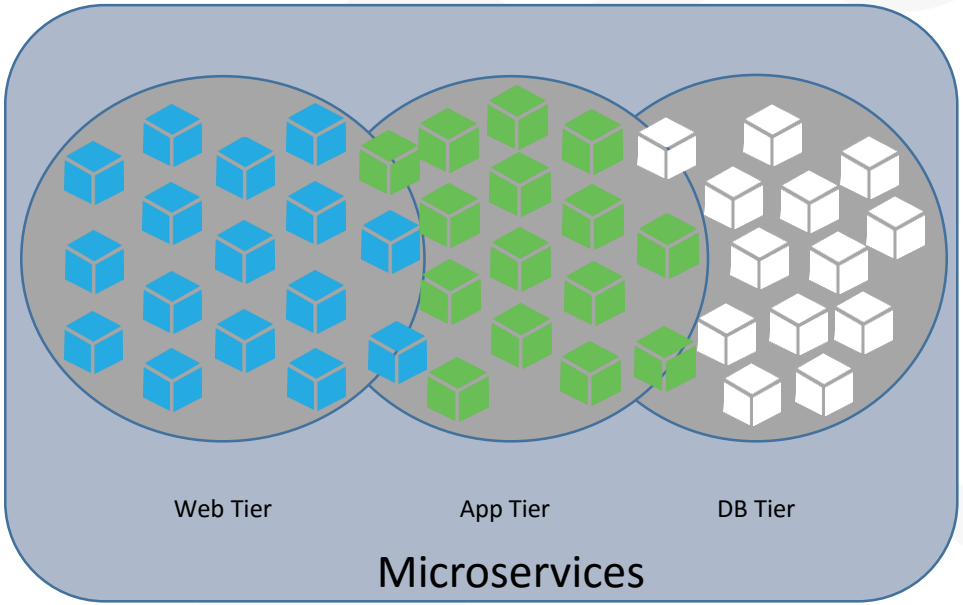
# Microservices

- “Born-in-the-Cloud” applications
- Highly Scalable
- Highly Available
- Modular
- Example
  - Receive request, process request, generate response

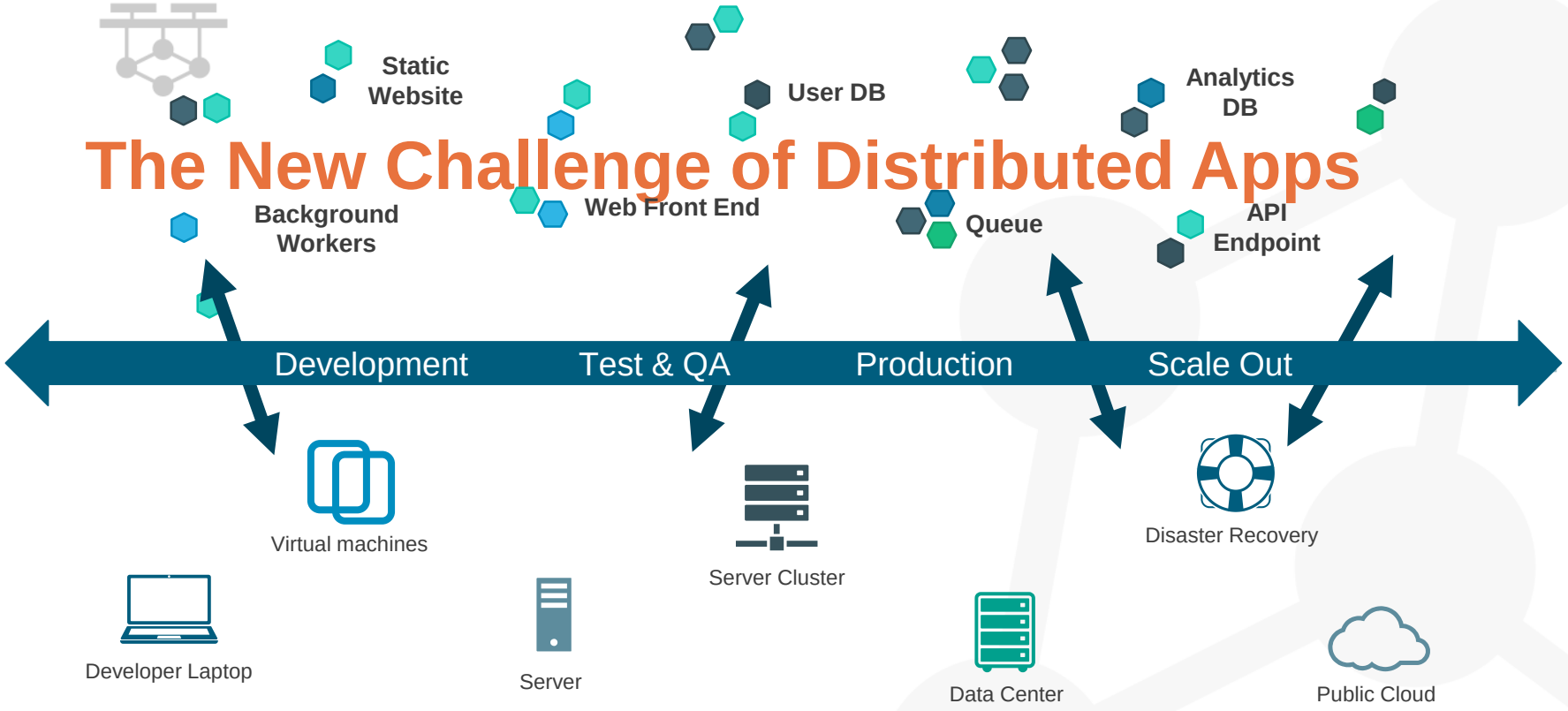


# Three-tier applications vs Micro-services



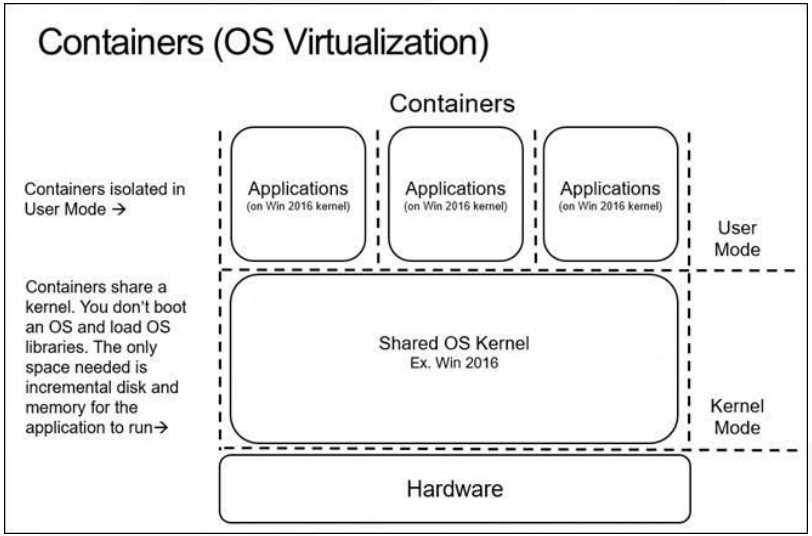



# The New Challenge of Distributed Apps



# Container Technology

- Virtual Machines : Hardware Virtualization
- Containers : OS Virtualization
- Isolation
  - Namespaces
  - Resource Control (CGroups)
- Layering
  - Union filesystems (UnionFS)
- Containers
  - Package layers into an image



# Windows Server Containers

## Anatomy and key capabilities

**Build:** Developers will use familiar development tools, such as emacs, vi, Visual Studio, to write apps to run within containers.

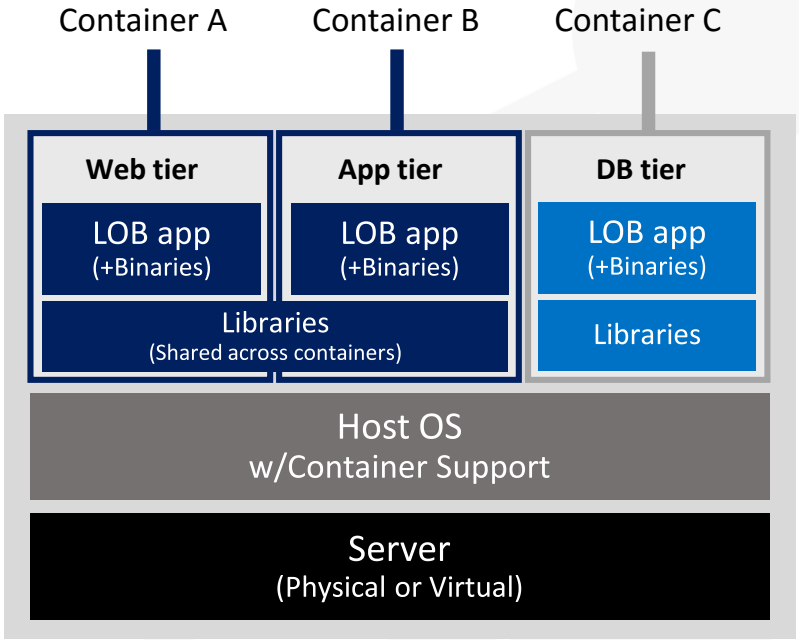
By building modular apps leveraging containers, modules can scale independently, and be updated on independent cadences.

**Run:** Container capabilities built into Windows Server

**Manage:** Deploy and manage containers using PowerShell, or using Docker.

**Resources:** Define CPU and memory resources per container along with storage and network throughput.

**Network:** Provide four modes of network connectivity (e.g. NAT)



# Hyper-V Containers

## Anatomy and key capabilities

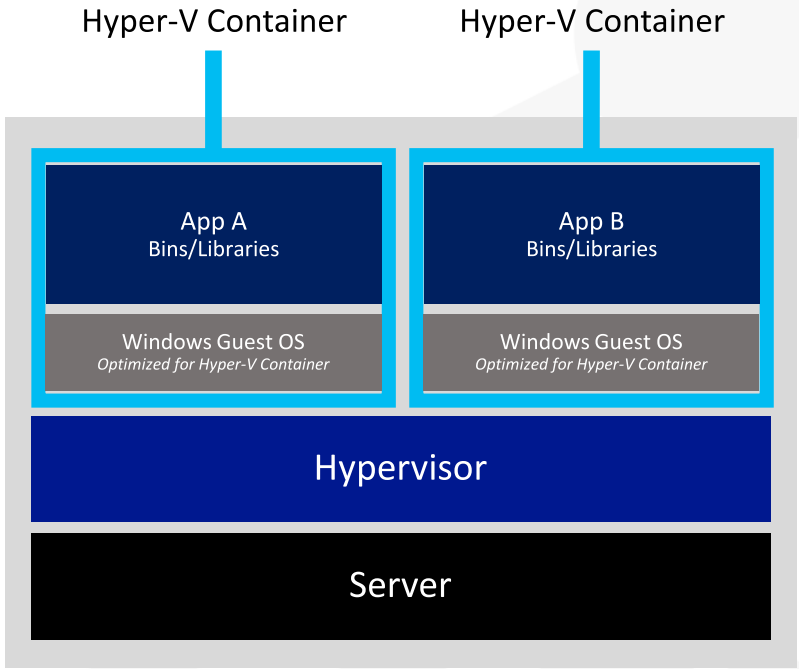
**Consistency:** Hyper-V Containers use the same APIs as Windows Server Containers ensuring consistency across management and deployment toolsets.

**Compatibility:** Hyper-V Containers use the exact same images as Windows Server Containers.

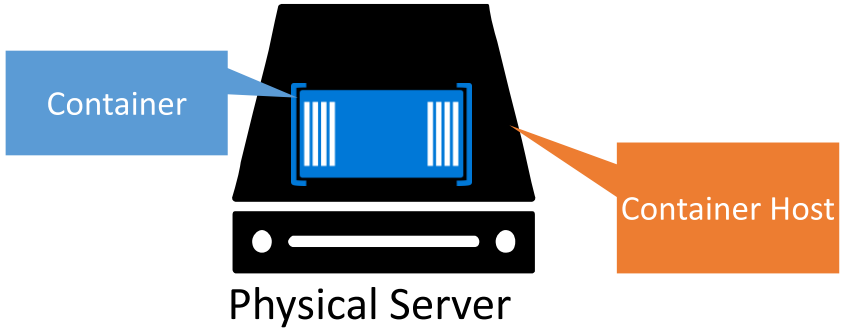
**Strong Isolation:** Each Hyper-V container has it's own dedicated copy of the kernel

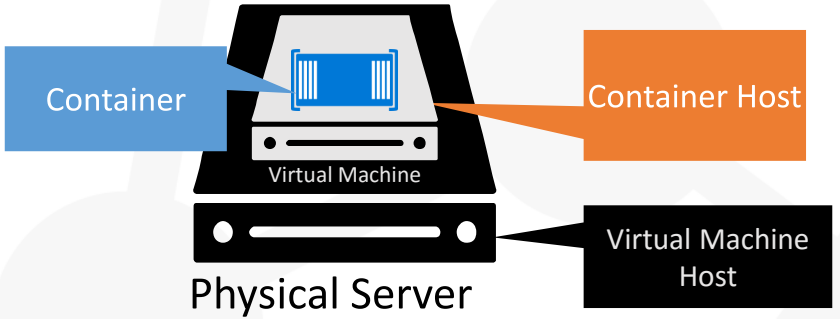
**Highly Trusted:** Built with proven Hyper-V virtualization technology.

**Optimized:** The virtualization layer and the operating system have been specifically optimized for containers



| | | Container Deployment | |
|---|---|---|---|
| Container Host Deployment | | **Server Core** | **Nano Server** |
| | **Server with UI** | Windows Server Containers<br>*Hyper-V Containers Not Supported In TP4* | Hyper-V Container<br>*Windows Server Containers Not Supported In TP4* |
| | **Server Core** | Windows Server Containers<br>*Hyper-V Containers Not Supported In TP4* | Hyper-V Container<br>*Windows Server Containers Not Supported In TP4* |
| | **Nano Server** | *Hyper-V Containers Not Supported In TP4* | Windows Server Containers<br>*or* Hyper-V Container |



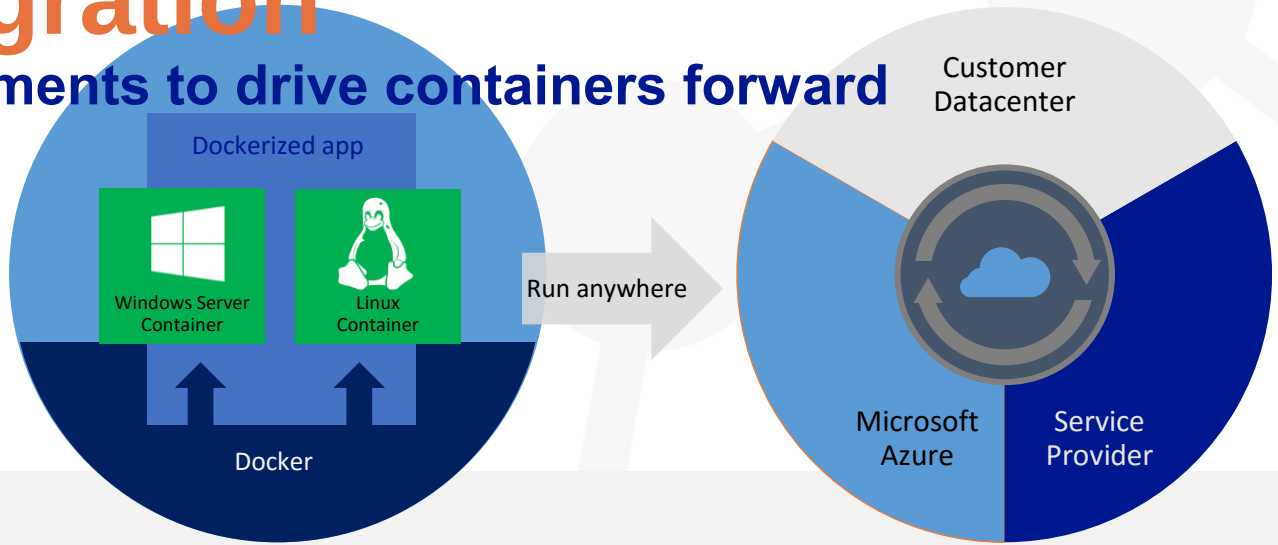



# Docker integration

## Joint strategic investments to drive containers forward

**Docker:** An open source engine that automates the deployment of any application as a portable, self-sufficient container that can run almost anywhere.

**Partnership:** Enable the Docker toolset to manage multi-container applications using both Linux and Windows containers, regardless of the hosting environment or cloud provider.




Strategic investments

Investments in the next wave of Windows Server

Open source development of the Docker Engine for Windows Server

Azure support for the Docker Open Orchestration APIs

Federation of Docker Hub images into the Azure Gallery and Portal

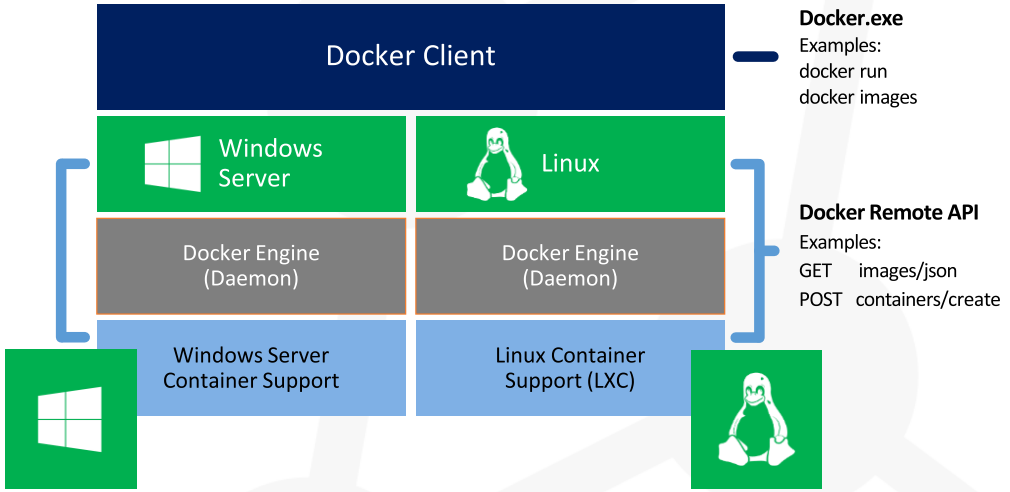
# Docker integration

**Docker Engine:** Docker Engine for Windows Server containers will be developed under the aegis of the Docker open source project.

**Docker client:** Windows customers will be able to use the same standard Docker client and interface on multiple development environments.

**Docker Hub:** Huge collection of open and curated applications available for download.

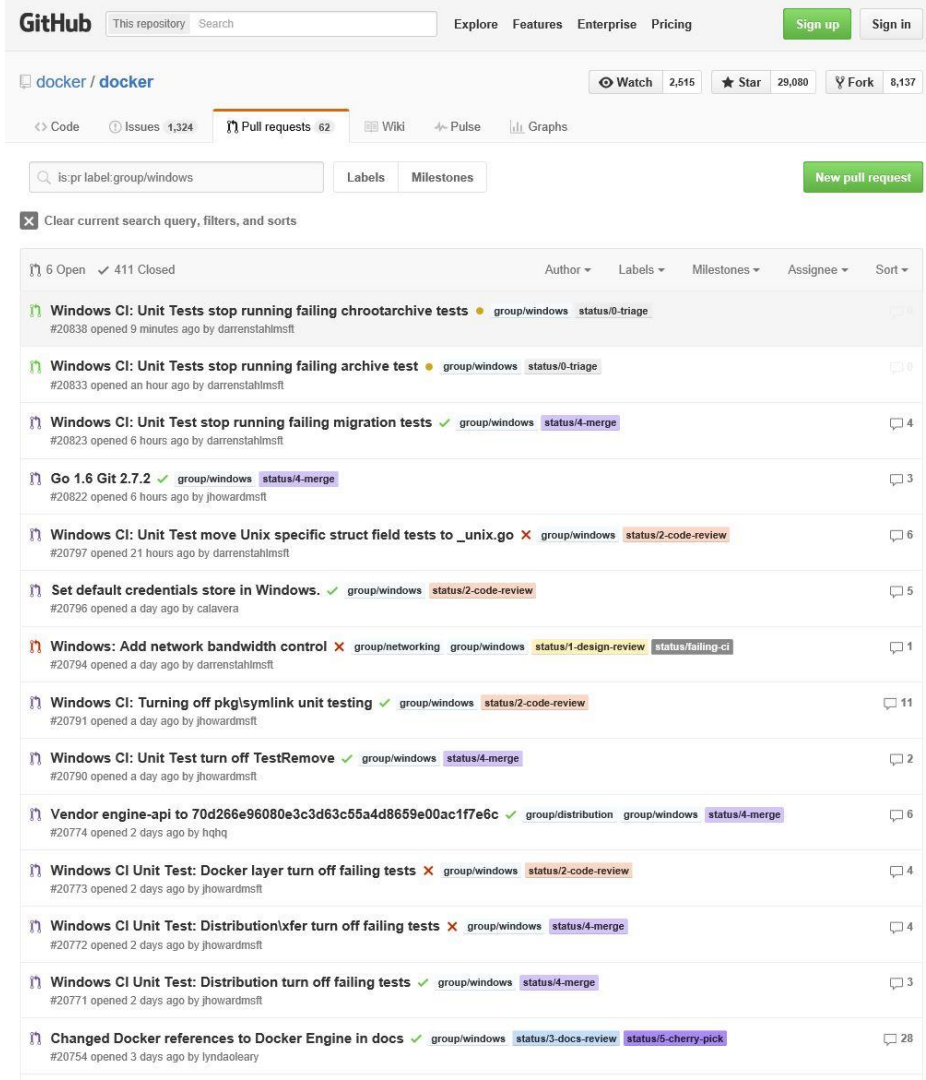
**Collaboration:** Bring Windows Server containers to the Docker ecosystem to expand the reach of both developer communities.



OPEN NETWORKING SUMMIT 2016
MARCH 14-17, 2016 | SANTA CLARA, CA

# Sample Pull Requests

- Windows CI Tests
- Network bandwidth
- Go / Git updates
- Docker engine updates



# Quick Windows Container Demo

- Create Windows Container using Docker
- Create Windows Container using PowerShell

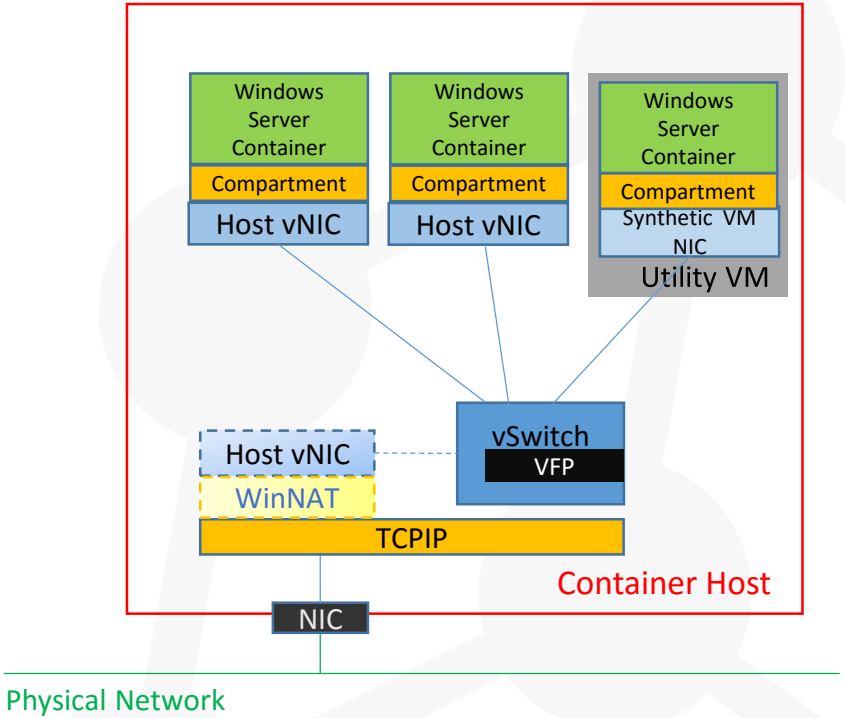
```
C:\> docker run --it --name=Container1 windowsservercore cmd.exe

PS > $container1 = New-Container -Name Container 2 windowsservercore
PS > Start-Container $container1
PS > Enter-PSSession -ContainerId $container1.Id -RunAsAdministrator
```

# Brief Intro to Windows Container Networking

- Windows Container Networking Modes
  - NAT – analogous to Docker “bridge” driver on Linux
  - Transparent – similar to Docker “host” driver on Linux
  - L2 Bridge – used in Microsoft private cloud to **bridge** network traffic and re-write container MAC addresses on the uplink port to the fabric (physical) host – *enables overlay networks*
  - L2 Tunnel – used in Azure public cloud to **forward** all traffic to the fabric (physical) host – *enables overlay networks*
- Docker libnetwork plug-in for Windows new in Technical Preview 5
  - Supports ***docker network*** commands for Cloud Network Model (CNM)
  - Creates IP endpoints with static and dynamic (ephemeral) port forwarding rules
  - Pull Requests:
    - https://github.com/docker/docker/pull/20478
    - https://github.com/docker/libnetwork/pull/973

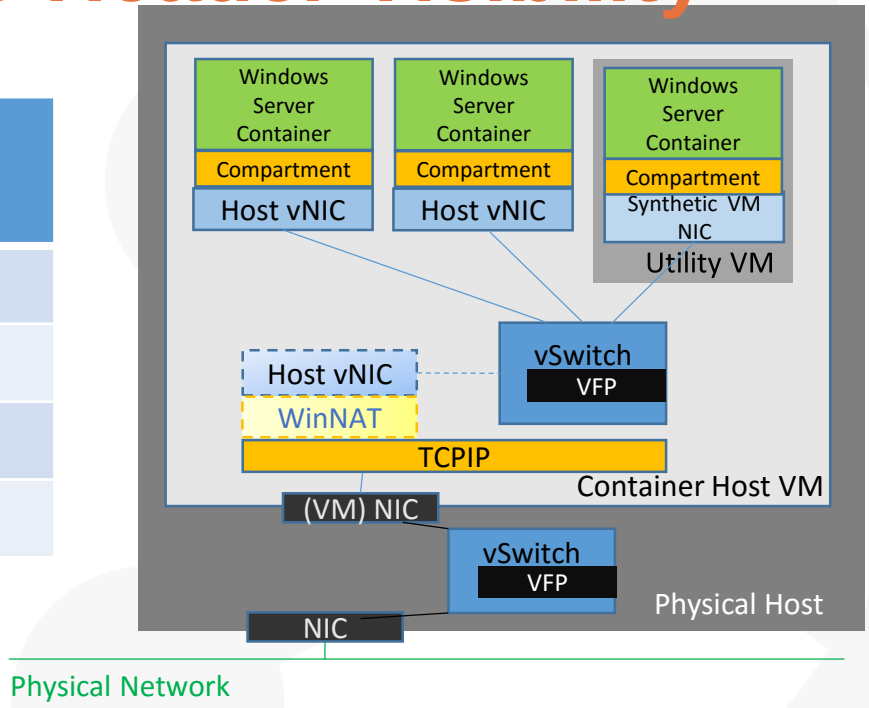
# Network Stack

- Containers connect to a Hyper-V Virtual Switch over a Host vNIC (Windows Server Container) or Synthetic VM NIC (Hyper-V Containers)
- The Host vNIC / Synthetic VM NIC sits within its own Network Compartment to provide isolation
- Network connectivity to Hyper-V Containers through synthetic VM NIC is transparent to the Utility VM
- (Optional) Host vNIC assigned default gateway IP from WinNAT which binds to TCPIP

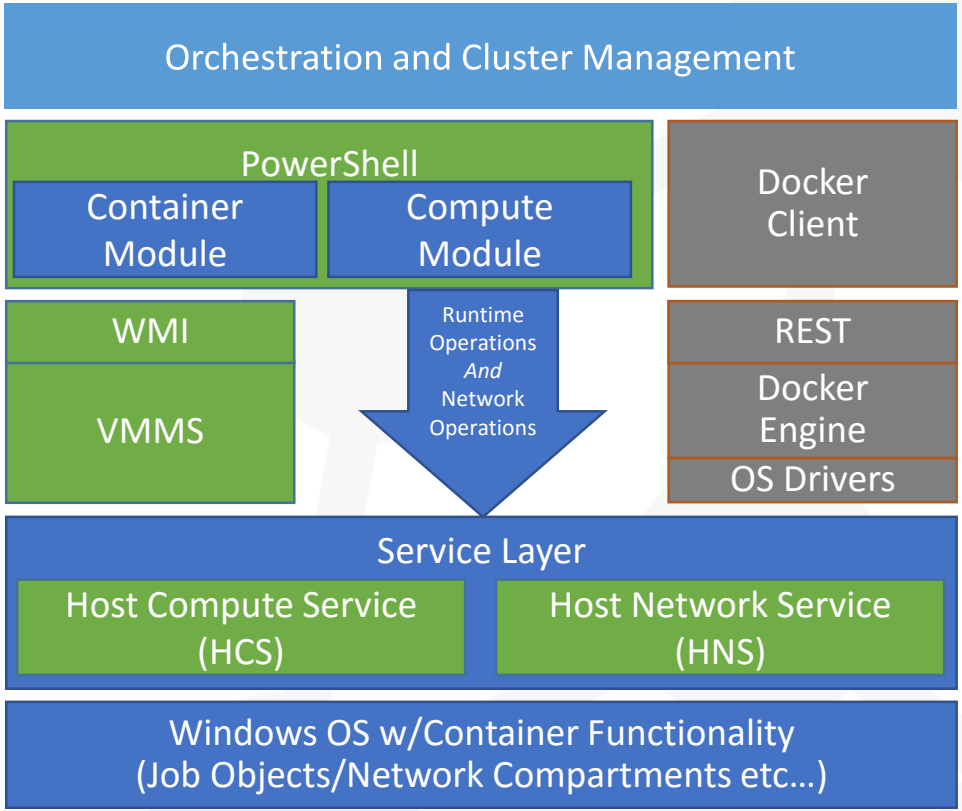


# Traffic Flow and L2- L3-Header Visibility

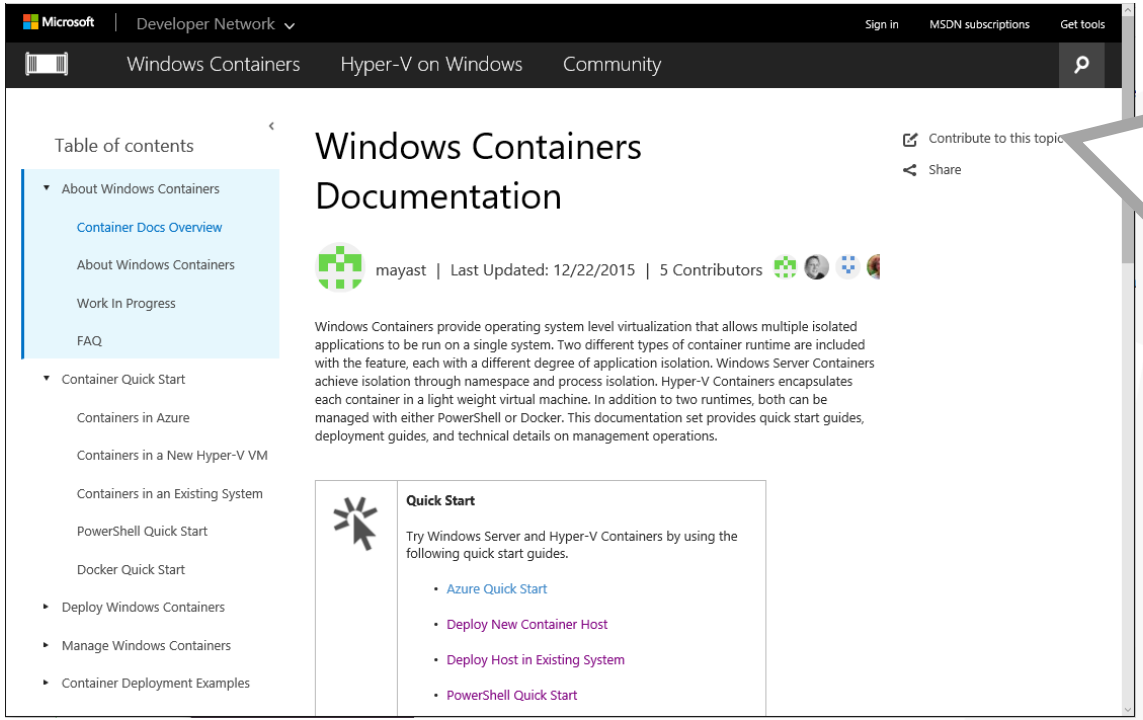
| Networking Mode | Layer-2 Visibility in Physical Host | Layer-3 Visibility in Physical Host |
|---|---|---|
| NAT | 1 MAC | 1 IP (Container Host) |
| Transparent | N MACs | N IPs |
| L2 Bridge | 1 MAC | N IPs |
| L2 “Tunnel” | 1 MAC | N IPs |



# Management Architecture



# http://aka.ms/WindowsContainers




GitHub

Documentation

Quick Start Guides

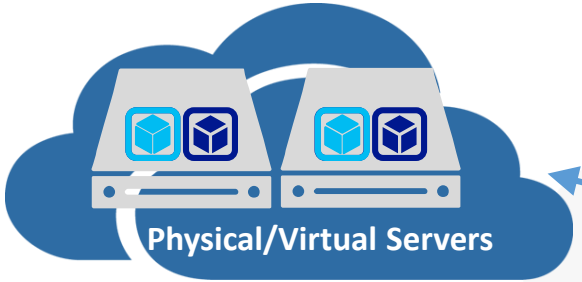
Setup Scripts

Samples
- Dockerfiles

# Container Lifecycle

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

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

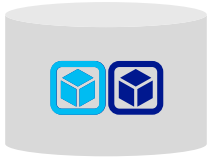



2

3

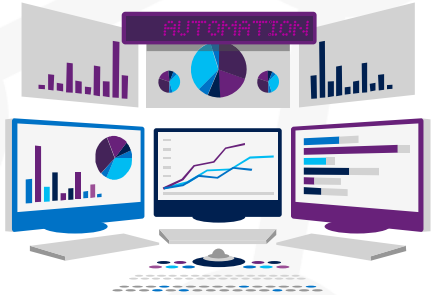
**Operations** collaborates with **developers** to provide app metrics and insights

1



Containers pushed to central repository

2




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

# Developer Concerns

- *"It works on my machine…"*

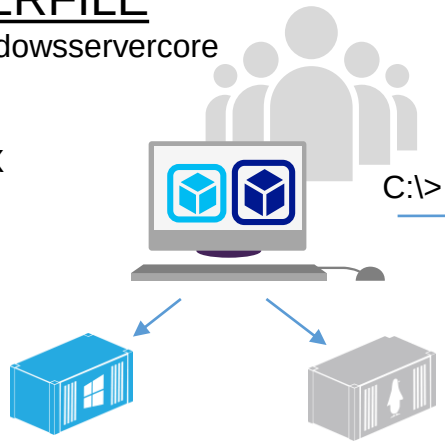
- How do I iteratively build my application?
- How do I package my application?
- How do I test my application?
- How do I provide networking?

# Developer Demo

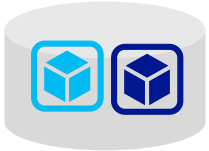
- Web Server Containers on Linux and Windows

## DOCKERFILE

FROM windowsservercore
RUN …
MKDIR …
EXPOSE X
…



C:\> Docker push

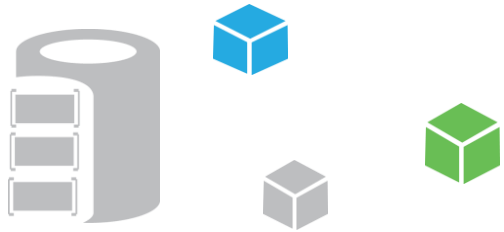


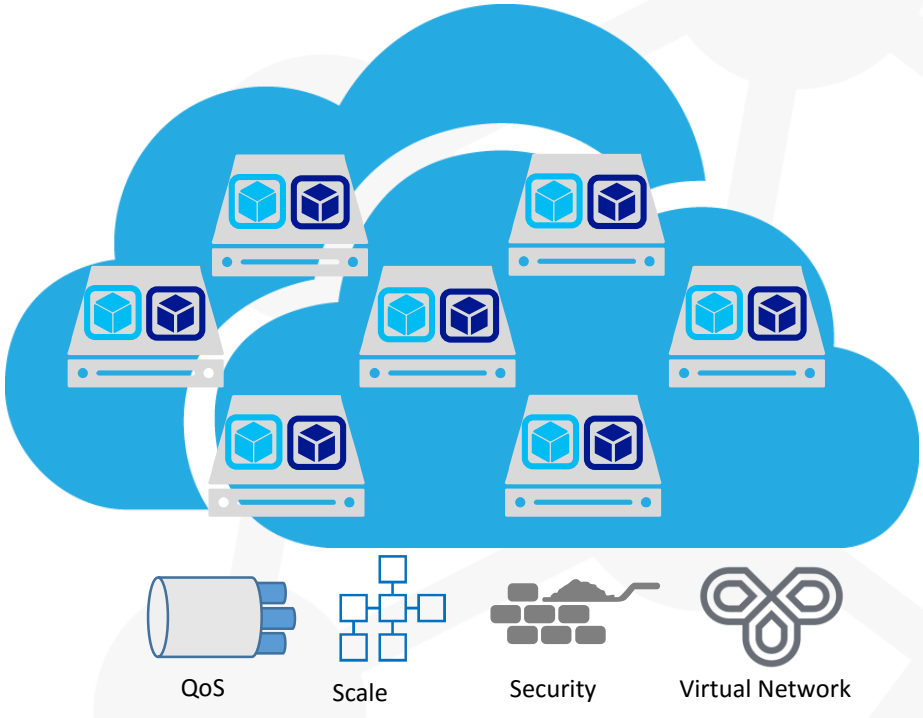
C:\> Docker pull

## Create Network

C:\> docker network create -d NAT –subnet= 172.18.0.0/24 MyNatNetwork
C:\> docker run –itd --name=Container1 --net=MyNatNetwork MyWebImage

# IT Pro Perspective





# IT Pro Concerns and Challenges

- How do I quickly deploy new microservices?
- How can I guarantee continuous availability?
- How can I be flexible with placement?
- How do I secure the microservices?
- What if I want to scale-up?

- *Does the dev have any requirements or intent I need to meet?*

# Developer + IT Pro

- We need to merge the Dev with the Ops to satisfy concerns of both
- Maintain developer intent with “Write once, run anywhere…”
- Help IT become the hero

# How can we solve these challenges?

# Software-Defined Networking (SDN)

- What is SDN?
  - Separate the control-planes from the data-planes
  - Based on Layers and Abstractions
- Overlay Virtual Networks
  - Layer-2 Encapsulation (e.g. VxLAN or NVGRE)
- Network policy for virtual networks and endpoints
  - Access Control Lists
  - Quality of Service queues
  - IP Address assignment
- **Goal: *Treat Infrastructure as Code***

# Correct Layering

Management

## Fabric Administration

- Deployment of Windows Server 2016 Networking Stack
- Deployment of Network Controller, Software Load Balancer Multiplexer & Unified Edge GW VMs

## Tenant Administration

- Self-service tenant portal
- Web based UI, REST APIs, PS Cmdlets for tenant workflows
- Azure Resource Manager (ARM)

Control

## Network Controller

- Programmatic interface (NorthBound API invoked through PowerShell and REST Wrappers)
- Logically centralized control plane for:
- Comprehensive monitoring of network health

Data

## Hyper-V vSwitch/SDN Extension

- Network Policy Enforcement
  - Encapsulation (e.g. VXLAN)
  - QoS max egress caps
  - ACLs

## Inbox Virtualized Network Functions

- Multi-tenant gateways: S2S IPSec (VPN) and GRE, L3 Forwarding
- Route Reflector (BGP)
- Software Load Balancer (SLB)

## Physical Network Gear

- OMI/DSC based configuration of switches (NetConf coming soon)
- 3rd-party Hardware VXLAN Gateway

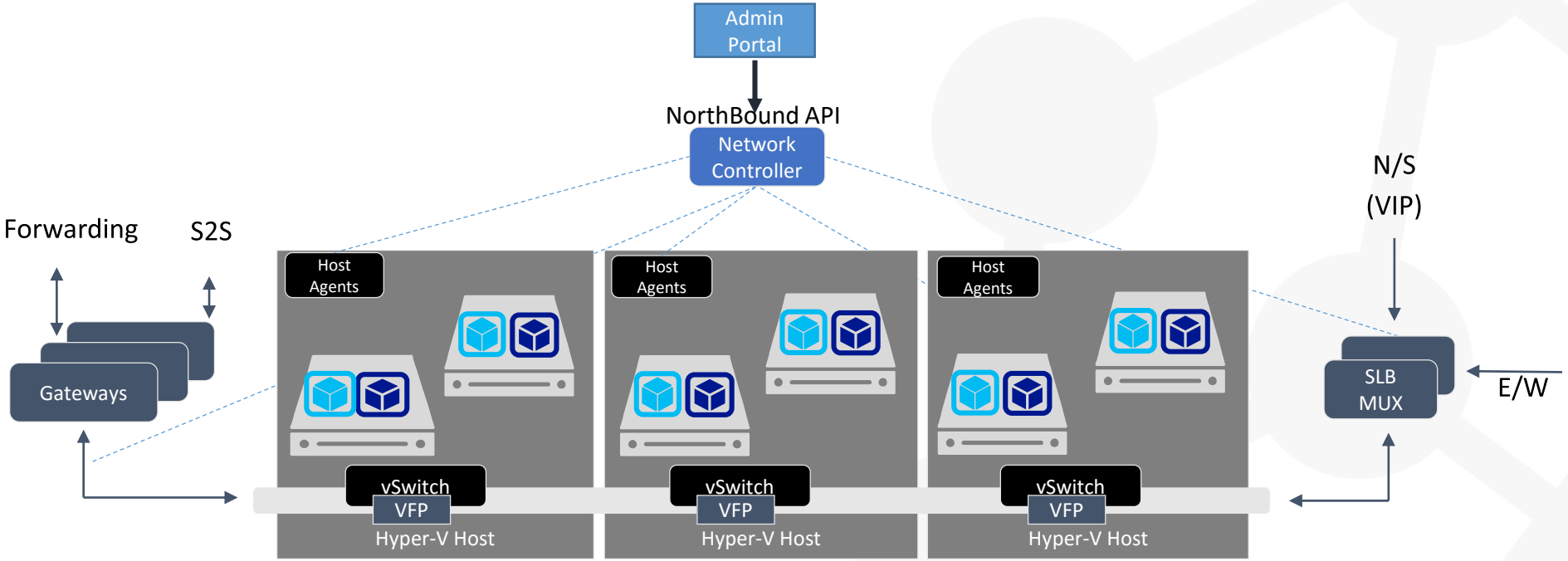
# Overlay Virtual Networks

- **Encapsulation**
  - ***Layer-2 Frame encapsulated with header (e.g. VxLAN) and wrapped in outer IP header***
- **Network Policy pushed down from the Network Controller**
- **Network Policy enforced at the vSwitch (Host networking)**

# Journey to Microsoft Azure Stack

- Microsoft SDN offering in Windows Server 2012R2 with System Center
  - Policy programmed through PowerShell and System Center Virtual Machine Manager (SCVMM)
  - Overlay Virtual Networks with NVGRE encapsulation
  - Support for ACLs and QoS
  - Extensible Virtual Switch
  - Gateways
- Microsoft Azure Stack with Windows Server 2016
  - Joint-engineering and consistency with Azure
  - SDN Features
    - Network Controller with open and RESTful NorthBound API
    - Overlay Virtual Networks with VxLAN encapsulation
    - Distributed Firewall for ACLs
    - Quality of Service queues
  - Network Function Virtualization (NFV) Features
    - In-Box L4 Software Load Balancer
    - Site-to-Site (S2S) Gateways: IPSec and GRE Tunnels
    - User-Defined Routing for Service Appliance chaining

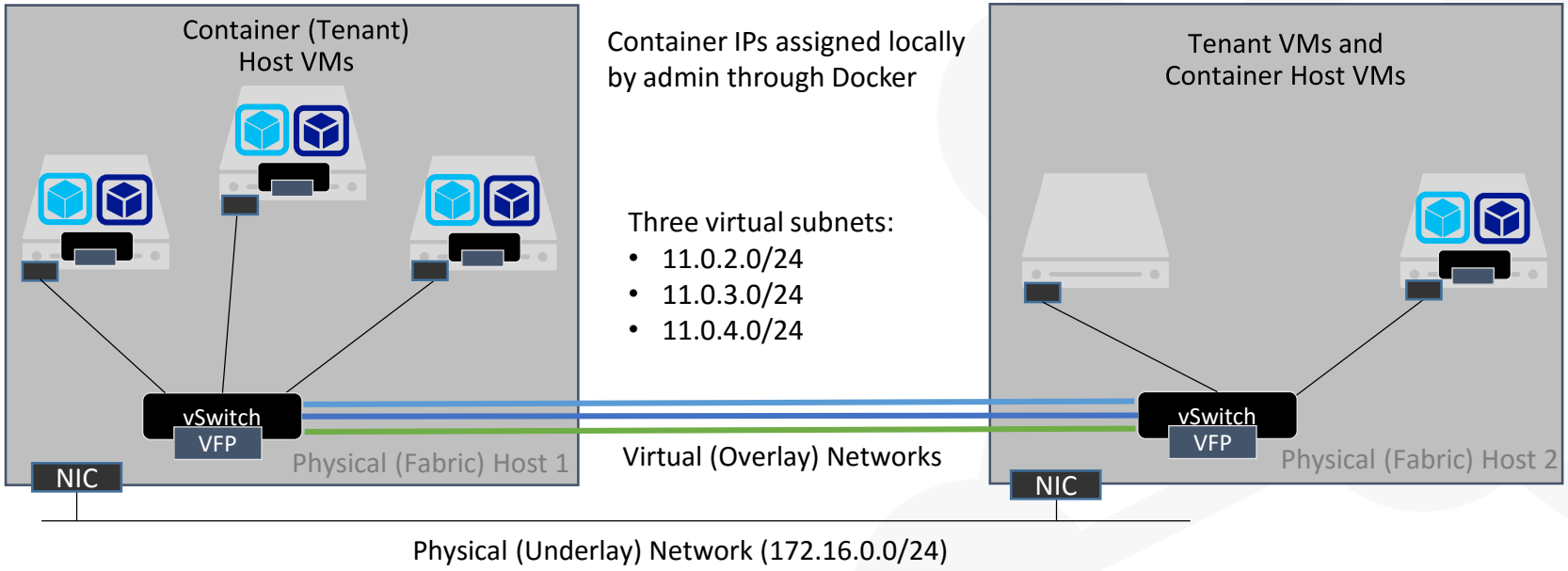
# Windows Server 2016 SDN Stack



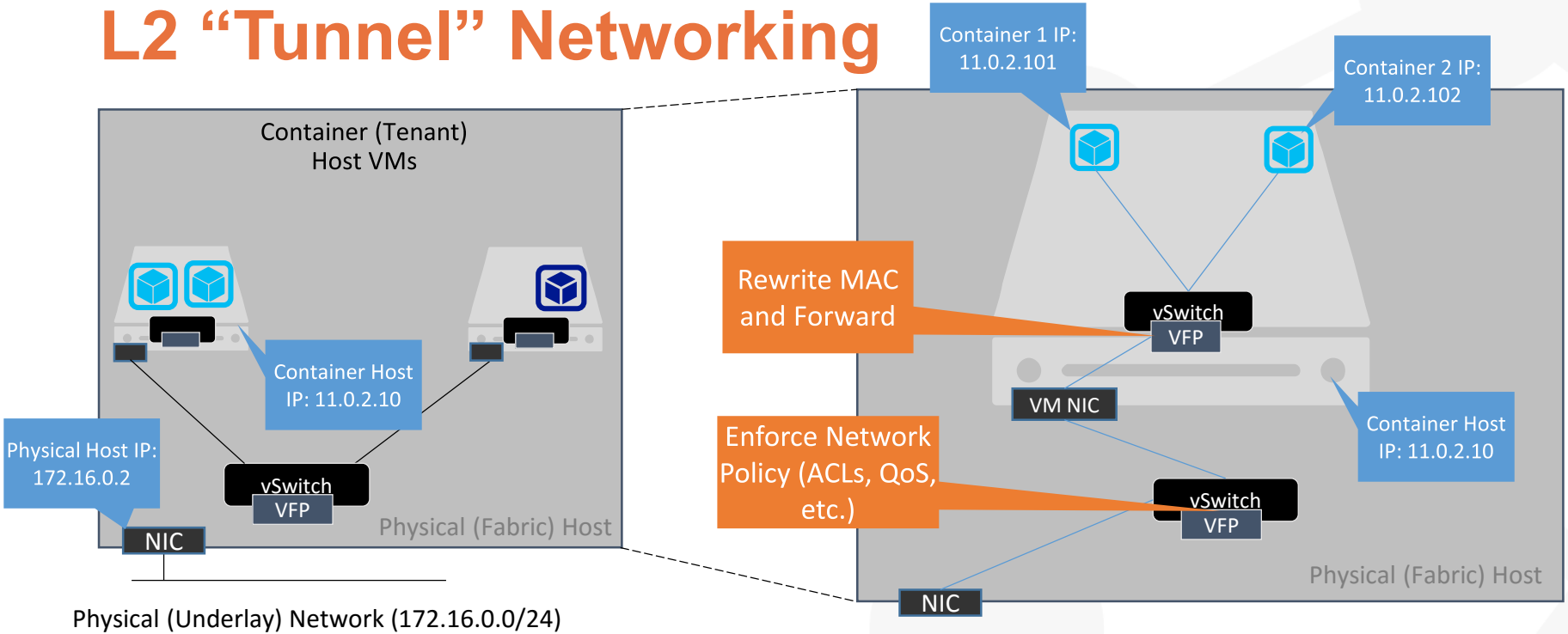
# Infrastructure as Code

- Using SDN Technology with the Microsoft Network Controller, we can treat infrastructure as code to dynamically create networks and assign policy
- Windows containers can join the overlay virtual networks created through the Microsoft Network Controller
  - L2 Bridge / Tunnel Networking Modes
  - IP-level granularity for assigning network policy

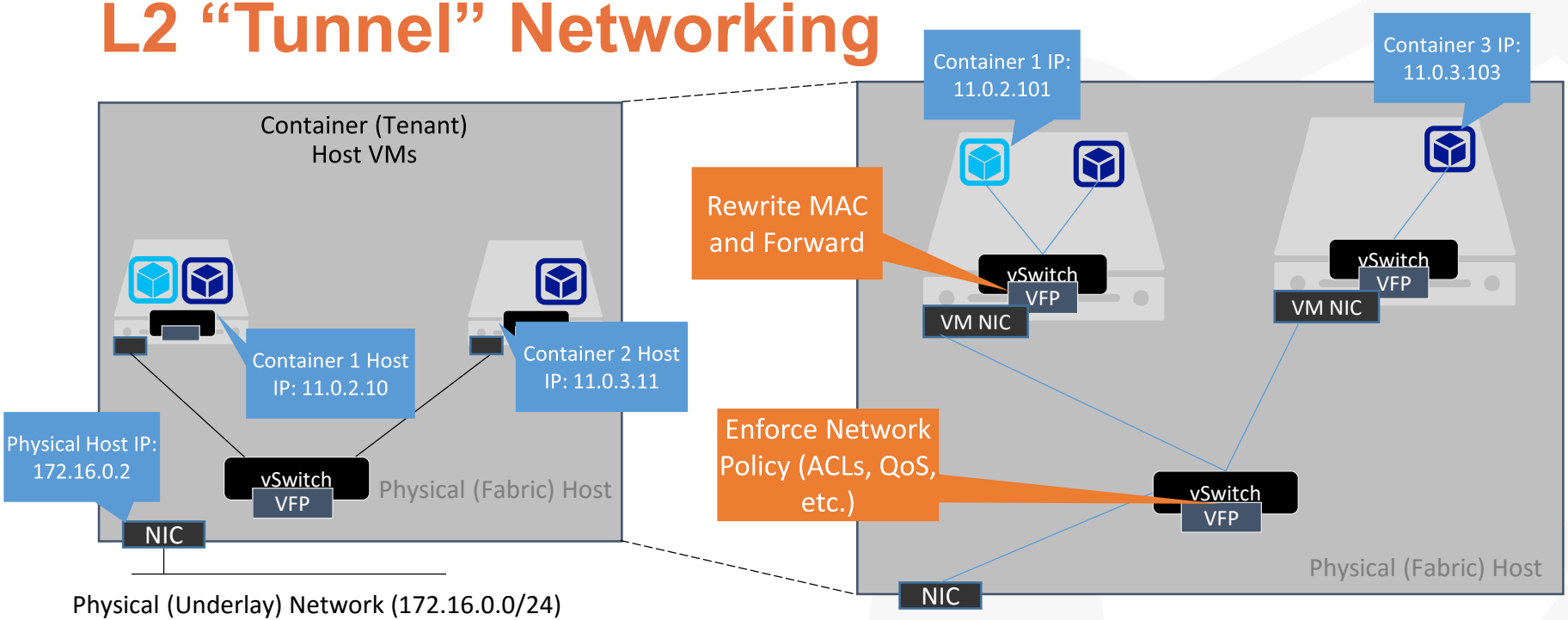
# L2 “Tunnel” Networking



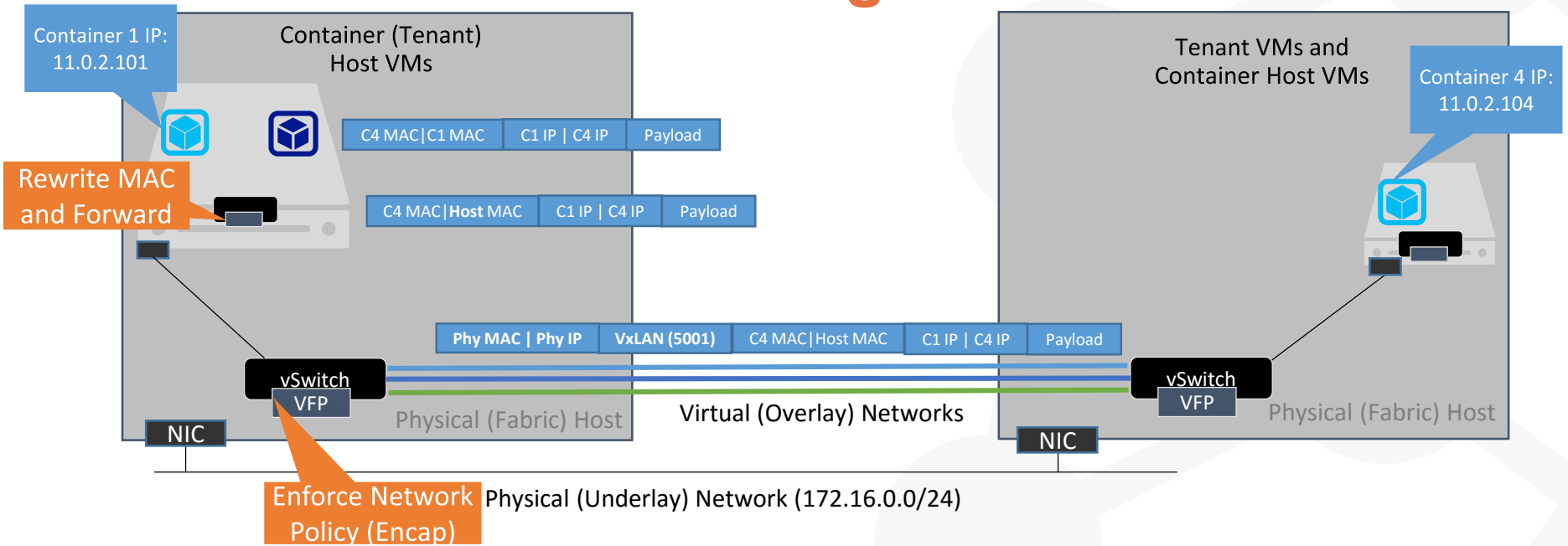
# L2 "Tunnel" Networking



# L2 "Tunnel" Networking



# L2 "Tunnel" Networking



# IT Pro Demo

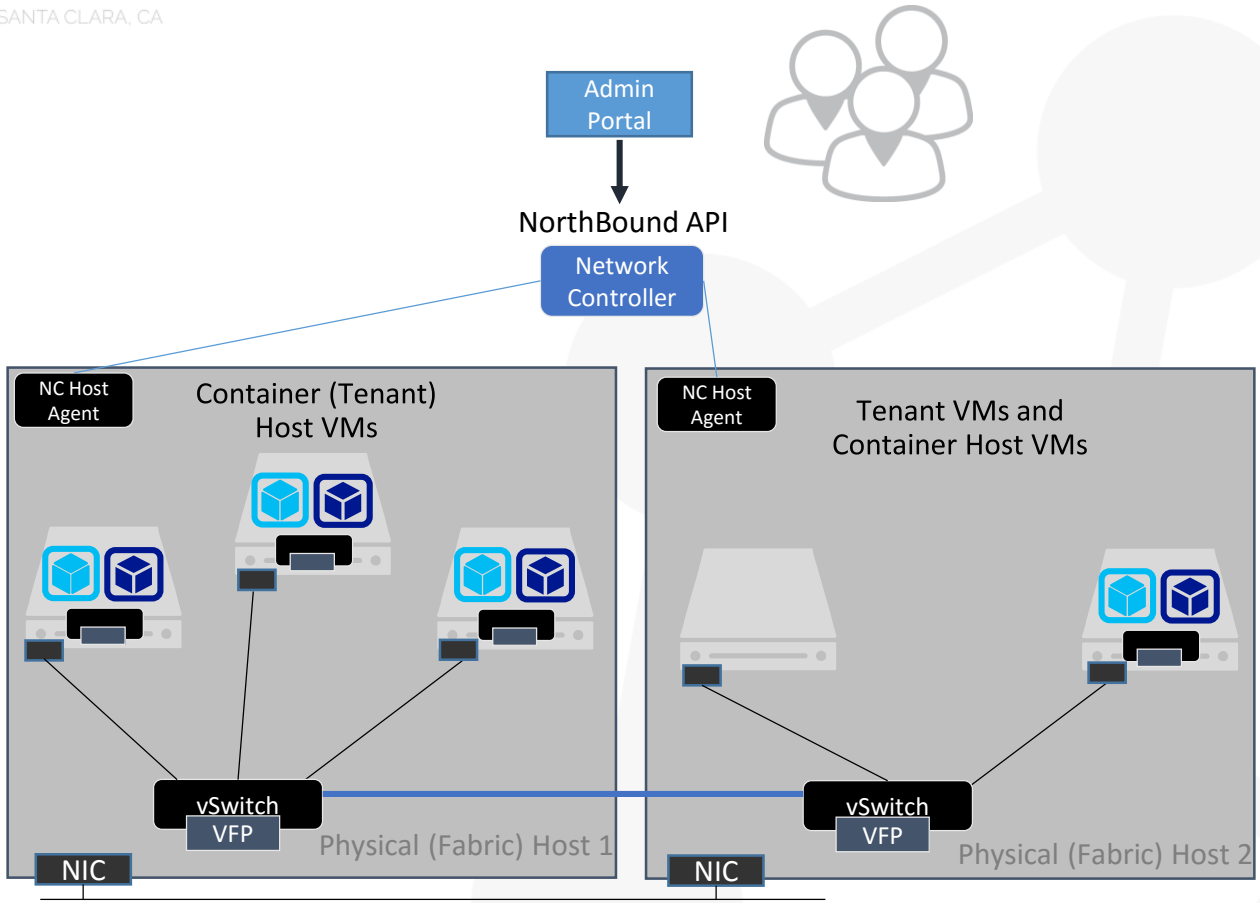
- Create virtual subnets
- Create Access Control Lists
- Create VMs and assign IP Addresses
- Create containers inside Container Host VM

# Network Policy Provisioning Approaches

- Private and Public Cloud Approaches for Container Networking
- Top-Down
  - Admin defines network policy (e.g. IP addresses, ACLs, etc.) in admin portal
  - SDN Stack sends policy from admin portal through Network Controller to Container Host
  - Container network policy “ready” to be enforced on physical host
  - Local container host admin creates containers and endpoints
- Bottom-Up
  - Container Host admin creates containers and specifies network policy for endpoints
  - Network policy applied at container host
  - Network policy communicated up the stack

## Top-Down Approach

1) Create virtual subnet
2) Create VMs
3) Assign IPs to VMs
4) Create containers
5) Assign IPs to Containers
6) Start Containers



# Innovations

- Incorporate developer intent into higher-level orchestration
- New network modes
  - Transparent
  - L2 Bridge
  - L2 “Tunnel”
- Access Control Lists (ACL)
  - [Available NOW] 5-tuple (Protocol, SRC/DEST IP, SRC/DEST Port) ACL rules
- QoS
  - [Future] Queue assignment on an IP-level granularity
  - [Available NOW] Maximum network bandwidth (egress caps)
  - [Future] Inbound port reservations
- Load Balancing
  - [Future] Layer-4 Software Load Balancer

# Q&A

- Your Questions!

- My Questions! ☺
  - How do we meet you where you are? Best engagement forum and medium?
  - Do you see a need for enforcing network policy directly on a container host?
  - Will you be deploying containers with other (non-Microsoft) SDN solutions?
  - How important is native support for Docker overlay driver on Windows?
  - What features are we missing?

# Additional Resources

- Containers
  - MSDN Documentation Hub For Windows Containers
  - The Container's Channel of Channel 9
  - Containers: Docker, Windows and Trends by Mark Russinovich
- Software Defined Networking
  - Microsoft TechNet Documentation for SDN
  - https://Github.com/Microsoft/sdn
  - Blogs
    - Four Datacenter Challenges and how Windows Server 2016 SDN can help
    - From Zero to SDN in Under Five Minutes
- Development Tools
  - Docker Tools for Visual Studio
  - Docker Tools for Visual Studio Code
  - yo docker

OPEN NETWORKING
SUMMIT 2016
MARCH 14-17, 2016 | SANTA CLARA, CA

# Thank You

Jason Messer (jmesser@microsoft.com)