Module 5: Hyper-V virtualization and containers in Windows Server

The concept of virtualization has evolved to include many aspects within the network environment. What started off primarily as virtual machines (VMs) has expanded to include virtual networking, virtual applications, and containers to comprise what's referred to as a software-defined infrastructure. As a server administrator, you must determine the server and service workloads that might run effectively in a virtual environment and which workloads should remain in a traditional, physical environment.

This module discusses the key features of the **Hyper-V server role** in Windows Server 2022. You'll learn how to configure Hyper-V networking and storage and manage a VM's state. You also learn how to help secure the Hyper-V host and associated VMs using security features within a guarded fabric that Windows Server provides.

This module's final lessons introduce you to the concept of using and managing containers, which are used to virtualize and package application code and processes to provide a protected and isolated environment and to an overview for Kubernetes.

By completing this module, you'll achieve the knowledge and skills to:

- Describe key features of Hyper-V in Windows Server.
- Deploy and configure VMs in Hyper-V.
- Help secure a virtual infrastructure using Hyper-V.
- Deploy containers using Windows Server.
- Describe container orchestration using Kubernetes.

SAMPLE

Lesson 1: Hyper-V in Windows Server

The Hyper-V server role provides a virtualized computing environment to create, configure, and manage VMs. To effectively plan and implement a VM environment, it's essential to understand key features that Hyper-V provides in Windows Server.

In this lesson, you'll learn how to use Hyper-V to implement virtualization and best practices for configuring Windows server hosts; factors to consider with respect to deployment scenarios such as nested virtualization; and considerations, requirements, and processes for migrating on-premises Hyper-V VMs to Microsoft Azure.

By completing this lesson, you'll achieve the knowledge and skills to:

- Describe how Hyper-V provides virtualization capabilities to Windows Server.
- Use Hyper-V Manager to manage virtualization.
- Identify best practices for configuring Hyper-V hosts.

- Explain nested virtualization.
- Describe how to migrate on-premises Hyper-V VMs to Azure.

Overview of the Hyper-V virtualization platform

Hyper-V is a server role in Windows Server x64 OS versions that leverages the capabilities of the underlying hardware to host multiple operating systems that are running simultaneously, but independently, of each other.

When you install the Hyper-V server role, it implements a software layer referred to as the *hypervisor*, which controls access to the physical hardware. Access is based on the concept of *partition isolation*, where each individual partition hosts an OS. Hyper-V has one designated partition known as the parent (or *root*) partition. The parent partition runs the Windows Server OS and hosts the virtualization management stack.

The Hyper-V root partition enables you to create child partitions, which can run any OS that the Hyper-V virtualization platform supports. These partitions are referred to VMs. The operating systems that are running in VMs are *guest operating systems*. The maximum number of child partitions is largely dependent on the capacity of the underlying physical resources, although Hyper-V in Windows Server 2022 does impose some scalability limits:

- The maximum number of running VMs is set to 1024.
- The maximum number of virtual processors is set to 2048.
- The maximum amount of memory is set to 48 terabytes (TB).

Individual VMs operate just like their physical counterparts, with an OS largely unaware of the abstraction layer that the hypervisor provides. The partition isolation facilitates sharing and dynamic allocation of computing, storage, and networking resources to accommodate changing usage patterns of virtualized workloads.

Hyper-V is used to support various scenarios from simple VMs to complex software-defined infrastructures. You can use Hyper-V to:

- Consolidate your server infrastructure.
- Provide a virtual development or test environment.
- Establish a virtual desktop infrastructure (VDI).
- Implement a private cloud infrastructure.

The ability to install Hyper-V depends on hardware support and sufficient capacity of computing resources. At a minimum, a Hyper-V host should have the following hardware components:

- A 64-bit processor with Second-Level Address Translation (SLAT).
- A processor with VM Monitor Mode extensions.
- A minimum of 4 gigabytes (GB) of memory.
- Intel Virtualization Technology (Intel VT) or AMD Virtualization (AMD-V) enabled.
- Hardware-enforced, Data Execution Prevention (DEP)-enabled (Intel XD bit, AMD NX bit).

SAMPLE

> **Note:** Windows Server contains the **Systeminfo.exe** command-line utility, which includes a check that automatically determines whether the local hardware meets the Hyper-V installation prerequisites.

As part of your Hyper-V deployment planning, you should also evaluate the resource requirements for the virtualized workloads you intend to deploy to a given host. Besides capacity and scalability, you should consider criteria such as resiliency and access to underlying hardware devices. In particular, you should consider:

- **The number of physical processor cores.** Make sure that you can allocate sufficient processing resources to each VM.
- **The amount of physical memory.** Ensure that there's enough physical memory to support the total number of VMs that you intend to run. The minimum amounts are OS- and workload-specific. In situations where usage patterns differ across VMs, you can utilize the Hyper-V support for dynamic memory allocation.
- **The amount of physical storage.** Ensure that there's enough storage to accommodate the disk space you'll allocate to the virtual hard disks (VHDs) attached to your VMs. Additionally, you should consider throughput and the number of input/output (I/O) operations per second (IOPS) required to support concurrent access from multiple VMs.
- **Network throughput.** Ensure that there's enough available network bandwidth to accommodate traffic generated and received by VMs.
- **Support for discrete device assignment.** This includes graphics processing units (GPUs) and non-volatile memory express (NVMe).

To install the Hyper-V server role, you can use Server Manager or the `Install-WindowsFeature` cmdlet in Windows PowerShell.

> **Learn more:** For more information about Hyper-V in Windows Server, refer to Hyper-V on Windows Server.

SAMPLE

Manage Hyper-V with Hyper-V Manager

Hyper-V Manager is often used as a graphical user interface (GUI) to manage both local and remote Hyper-V hosts. Hyper-V Manager is available when you install the Hyper-V Management Tools, which a complete Hyper-V server role installation includes or which you can install as a tools-only installation.

Hyper-V Manager supports the following general features:

- **Previous version support**. When using Hyper-V Manager on Windows Server 2022 or Windows 10, you can still manage hosts installed with previous operating systems such as Windows Server 2019, Windows Server 2016, or Windows Server 2012 R2.
- **Support for WS-Management protocol**. Hyper-V Manager supports connections to Hyper-V hosts over the Web Services Management Protocol (WS-Management Protocol). This allows Hyper-V Manager to communicate by using the Kerberos protocol NTLM or Credential Security Support

Provider (CredSSP). When using CredSSP, you remove the need for Active Directory Domain Services (AD DS) delegation. This makes it easier to enable remote administration because WS-Management Protocol communicates over either ports 80 or 443, which are the default open ports.

- **Alternate credential support**. Communicating over the WS-Management Protocol enables you to use different credentials in Hyper-V Manager and to save those credentials for ease of management. However, alternative credentials only work with Windows 10 and newer and Windows Server 2016 and newer hosts. Older servers installed with Hyper-V don't support the WS-Management Protocol for Hyper-V Manager communication.

Hyper-V Manager is the most common interface for managing VMs in Hyper-V, but there are other tools that provide similar features for specific management scenarios, including:

- **Windows PowerShell.** The Hyper-V module for Windows PowerShell provides PowerShell cmdlets that you can use for scripting or command-line administrative scenarios.
- **PowerShell Direct.** PowerShell Direct allows you to use Windows PowerShell inside a VM, regardless of the network configuration or remote-management settings on either the Hyper-V host machine or the VM.
- **Windows Admin Center.** The Windows Admin Center is a browser-based application that remotely manages Windows Servers, clusters, and Windows 10 and newer PCs.
- **System Center Virtual Machine Manager (SCVMM).** SCVMM is part of the System Center suite, used to configure, manage, and transform traditional datacenters, and helps to provide a unified management experience across on-premises, service providers, and the Azure cloud.

SAMPLE

Best practices and considerations for configuring Hyper-V hosts

You should consider the following best practices when provisioning Windows Server to function as a Hyper-V host:

- Provision the host with adequate hardware.
- Deploy VMs on separate disks, solid state drives (SSDs), or Cluster Shared Volumes (CSVs) if using shared storage.
- Don't collocate other server roles.
- Manage Hyper-V remotely.
- Run Hyper-V by using a Server Core configuration.
- Run the Best Practices Analyzer and resource metering.
- Use generation 2 VMs if the guest OS supports them.

Provision the host with adequate hardware

Perhaps the most critical best practice is to provision your Hyper-V host with adequate hardware, ensuring appropriate processing capacity and amount of random access memory (RAM), and that it has fast and redundant storage. Additionally, ensure that you provision the Hyper-V host with

multiple network adapters configured as a team. Inadequately provisioning a Hyper-V host with hardware affects the performance of all VMs that the server hosts.

Deploy VMs on separate disks

You should use separate disks to host VM files rather than store VM files on the same disk as the host OS files. Doing this minimizes contention and ensures that read/write operations on VM files don't conflict with read/write operations at the host OS level.

It also minimizes the chance that VM hard disks will grow to consume all available space on an OS volume. We also recommend using SSDs, which have much faster read-write speed and consume less power than standard hard-disk drives.

You can reduce the effect on performance if you deploy to a disk that uses striping, such as a Redundant Array of Independent Disks (RAID) 1+0 array. Using shared storage means you can provision multiple VMs on the same logical unit number (LUN) if you use CSVs. However, choosing between separate LUNs for each VM or a shared LUN depends heavily on VM workload and host configuration.

Don't collocate other server roles

You should ensure that Hyper-V is the only server role installed on the host server. Don't collocate the Hyper-V role with other roles, such as the domain controller (DC) or the file server role. Each role that you deploy on a server requires resources, and when deploying Hyper-V, you want to ensure that VMs have access to as much of a host server's resources as possible. If locating these roles on the same hardware is necessary, deploy them as VMs rather than installing them on the physical host.

Manage Hyper-V remotely

When you sign in locally to a server, your session consumes server resources. If you configure remote management of a Hyper-V server instead of performing administrative tasks by signing in locally, you can ensure that all possible resources on the Hyper-V host are available to the hosted VMs. You should also restrict access to the Hyper-V server so only administrators responsible for VM management can make connections. A configuration error on a Hyper-V host can cause downtime for all guest VMs.

Run Hyper-V by using a Server Core configuration

You should manage Hyper-V by using the Server Core configuration, which provides the following benefits:

- The Server Core configuration of Windows Server minimizes hardware-resource utilization for the host OS. As a result, the hosted VMs have more hardware resources.
- The Server Core requires fewer software updates, requiring fewer restarts.

SAMPLE

- When you restart a Hyper-V host and it's unavailable, all VMs that the server hosts also become inaccessible. To prevent this issue, consider using a clustered environment. Because a Hyper-V host can host many critical servers as VMs, ensure that you minimize downtime.

Run the Best Practices Analyzer and use resource metering

You can use the Best Practices Analyzer to determine specific configuration issues you should address. You can also use resource metering to monitor how hosted VMs use server resources and determine if specific VMs use a disproportionate amount of a host server's resources. If the performance characteristics of one VM erode the performance of other VMs hosted on the same server, consider migrating that VM to another Hyper-V host.

Use generation 2 VMs if the guest OS supports them

Generation 2 VMs have slightly faster start times than Generation 1 VMs. Generation 2 VMs use a simplified hardware model and allow advanced features such as:

- Pre-Boot Execution Environment (PXE) boot from a standard network adapter.
- Hot add/removal of virtual network adapters.
- Small Computer System Interface (SCSI) controller boot.
- Secure boot.
- Shielded VMs.
- Support for use of the Storage Spaces direct service.

SAMPLE

Overview of nested virtualization

The term *nested virtualization* refers to when the Hyper-V server role is installed within a VM's OS, thereby enabling you to provision "nested" VMs within that VM. While this is rarely suitable in production environments, it's useful for development and testing.

When implementing nested virtualization, there are additional requirements beyond the prerequisites for traditional Hyper-V server role installations. However, physical processor requirements are vendor dependent, such as:

- When using an Intel processor with VT-x and EPT technology, the Hyper-V host must run Windows Server 2016 or newer, and the VM configuration must be version 8.0 or newer.
- When using AMD EPYC or AMD Ryzen processors or newer, the Hyper-V host must be Windows Server 2022, and the VM configuration must be version 10.0 or newer.

> **Note:** You'll learn about the VM configuration versions later in this module.

Before you configure nested virtualization in a VM, you must enable virtualization extensions for its virtual processor. To accomplish this, ensure that the VM is stopped, and then from the Hyper-V

host, run the following Windows PowerShell command, where <*vmname*> is the placeholder designating the name of the target VM:

```
Set-VMProcessor -VMName <vmname> -ExposeVirtualizationExtensions $true
```

You also must consider the number of virtual central processing units (vCPUs) assigned to the VM, the amount of memory allocated to it, and its network configuration. While 4 GB of RAM should suffice to install the Hyper-V server role, the optimal amount will depend on the planned number of nested VMs and their memory demands.

> **Note:** Dynamic memory and runtime memory resize aren't available on VMs running Hyper-V.

From the networking standpoint, establishing connectivity between nested VMs and external networks typically involves enabling the **MAC Address Spoofing** feature on the network adapter of the VM configured as the Hyper-V host. You can perform this task directly from the **Hyper-V Manager** console or by using the `Set-NetworkAdapter` PowerShell cmdlet. This feature isn't necessary to allow connectivity between nested VMs.

To enable MAC address spoofing, run the following command from the physical Hyper-V host machine:

```
Set-VMNetworkAdapter -VMName <VMName> | Set-VMNetworkAdapter -
MacAddressSpoofing On
```

Alternatively, you can provide connectivity between nested VMs and external networks by implementing network address translation (NAT) for the VM serving the role of the Hyper-V host. This is a common approach in environments where access to the Hyper-V host isn't available, such as Azure.

After you enable virtualization extensions and satisfy all other prerequisites, the rest of the setup process is the same as when you configure Hyper-V on a physical host.

> **Learn more:** For more information about nested virtualization in Hyper-V, refer to Run Hyper-V in a Virtual Machine with Nested Virtualization.

Integrate Hyper-V platform with Azure IaaS

Many organizations decide to move some or all their server infrastructure to cloud-based platforms, such as Azure. These organizations realize the benefits and take advantage of the decreased cost of infrastructure maintenance, increased scalability, and high availability of a cloud-based infrastructure.

Using the Azure Migrate service, you can discover, assess, and migrate many of your on-premises workloads, apps, and VMs to Azure.

Azure Migrate is a service that Microsoft Azure includes that provides the following benefits:

SAMPLE

- **A single migration platform.** Provides a single portal to start, run, and track your migration to Azure.
- **Assessment and migration tools.** Includes several tools to assist in your migration tasks, including Azure Migrate: Server Assessment and Azure Migrate: Server Migration.
- **Assess and migrate multiple object types.** The Azure Migrate hub portal allows you to assess and migrate:
  - Servers.
  - Databases.
  - Web applications.
  - Virtual desktops.
  - Data.

How does Hyper-V migration to Azure work

To migrate VMs from local Hyper-V to Azure, you can use the Server Migration tool, which provides agentless replication to Azure for on-premises Hyper-V VMs. Software-agent components are only installed on the Hyper-V hosts or cluster nodes, and no agents must be installed on the Hyper-V guest VMs.

The Server Migration tool shares common technology with the Microsoft Azure Site Recovery tool, which had been the primary solution for replicating VMs to Azure before the release of the Azure Migrate service. The components used for replication include:

- **Replication provider**. Installed on Hyper-V hosts and registered with Azure Migration Server Migration and used to orchestrate replication for Hyper-V VMs.
- **Recovery Services agent**. Works with the provider to replicate data from Hyper-V VMs to Azure. Replicated data is migrated to a storage account in your Azure subscription. The Server Migration tool then processes the replicated data and applies it to replica disks used to create the Azure VMs during migration.

SAMPLE

Lesson 2: Configure VMs

After installing the Hyper-V server role on your host server, your next step is configuring your virtual infrastructure, which consists of several components and tasks, including configuring virtual networks, creating virtual disks, and creating and managing VMs that contain supported operating systems.

In this lesson, you'll learn the concepts related to VM configurations and generation versions, VM settings, storage options, and virtual disk types. You'll also learn about the types of virtual networks and how to create and manage a VM.

By completing this lesson, you'll achieve the knowledge and skills to:

- Describe VM configuration and generation versions.
- Explain operating systems in Hyper-V VMs.

- Describe VM settings.
- Describe storage options in Hyper-V.
- Describe VHD formats and types.
- Describe Shared VHDX and VHD Set files.
- Explain Hyper-V networking.
- Describe networking features for Hyper-V.
- Manage VM checkpoints.
- Import and export VMs.

VM configuration and generation versions

A Hyper-V VM's configuration version determines its capabilities and affects the range of options you can use to configure its settings, saved state, and checkpoints. The initial value of the configuration version is set during VM creation. By default, it depends on the OS of the underlying Hyper-V host.

Prior to Windows Server 2016 Hyper-V, upgrading the OS would automatically result in upgrading the configuration version of all VMs on that host. However, starting with Windows Server 2016, you must explicitly involve a configuration-version upgrade. This behavior enables more administrative control over the upgrade process and facilitates a fallback in case an issue occursof an issue.

Table 1 lists the VM configuration version numbers and the corresponding Hyper-V host OS versions that support them:

Table 1: VM configuration versions

Windows version of the Hyper-V host	Configuration versions
Windows Server 2022	10.0, 9.3, 9.2, 9.1, 9.0, 8.3, 8.2, 8.1, 8.0
Windows Server 2019	9.0, 8.3, 8.2, 8.1, 8.0, 7.1 ,7.0 ,6.2, 5.0
Windows Server 2016	8.0, 7.1 ,7.0 ,6.2, 5.0
Windows Server 2012 R2	5.0

SAMPLE

Identify the VM configuration version

To identify the configuration version of VMs on a Hyper-V host, run the following PowerShell command from the host OS:

```
Get-VM * | Format-Table Name, Version
```

Update a single VM

To update a VM's configuration version, run the following PowerShell commands from the host OS, where <*vmname*> is the placeholder designating the target VM's name:

```
Stop-VM -Name <vmname>
Update-VMVersion <vmname> -Confirm $false
Start-VM -Name <vmname>
```

> **Note:** The VM must be in the stopped state for the update to succeed.

Updating a VM configuration version automatically sets it to the highest value supported by the Hyper-V host OS. For example, if you update the configuration version of a VM on a Hyper-V host running Windows Server 2022, the configuration version is updated to version 10.0.

Hyper-V VM generations

Windows Server Hyper-V supports two generations of VMs: Generation 1 and Generation 2. A VM's generation determines the virtual hardware and functionality available to it. Generation 2 VMs use a different virtualized hardware model and therefore, no longer support several legacy devices such as COM ports, the emulated floppy disk drive, and integrated drive electronics (IDE) controllers. Generation 2 VMs also use Unified Extensible Firmware Interface (UEFI) firmware, unlike Generation 1 VMs, which rely on the basic input/output system (BIOS).

You must choose a VM's generation during its provisioning. You can deploy and run a mix of Generation 1 and Generation 2 VMs on the same Hyper-V host.

Generation 2 VMs offer several advantages over their Generation 1 counterparts, including:

- Secure boot leveraging UEFI firmware support.
- Boot from a VHD connected to a virtual SCSI controller.
- Boot from a virtual DVD connected to a virtual SCSI controller.
- PXE boot by using a synthetic Hyper-V network adapter.
- Larger boot volume of up to 64 TBs when using the .vhdx file format.
- Slightly shorter boot and installation times.

Because of the Generation 2 VM enhancements, you should use it, but only if your guest OS supports it. Also, because Generation 2 VMs rely on UEFI firmware rather than BIOS, they don't support 32-bit operating systems.

Operating systems in Hyper-V VMs

Every currently supported version of the Windows Server OS includes the option to install the Hyper-V server role. Previous Windows Server versions (up to Windows Server 2019) also support the free Hyper-V Server edition, which includes the Windows hypervisor with the required driver

SAMPLE

and virtualization components, providing a specialized virtualization solution that offers optimized server utilization at reduced cost.

With respect to guest operating systems, in general, any OS that uses an x86/x64 architecture can run in a Hyper-V VM, although Microsoft doesn't officially support all of them.

Hyper-V in Windows Server 2022 does support the following guest operating systems:

- Windows Server 2008 with Service Pack 2 (SP2) or newer.
- Windows 7 with Service Pack 1 (SP1) or newer.
- Main Linux distributions including CentOS, Red Hat Enterprise Linux, Debian, Oracle Linux, SUSE, and Ubuntu.
- FreeBSD.

VM settings overview

In the Hyper-V Manager, the VM settings are grouped into two main sections: Hardware and Management. The configuration files that store hardware and management information are separated into two formats: .vmcx and .vmrs. The .vmcx format is used for configuring VMs, and the .vmrs format is used for runtime data. This helps decrease the chance of data corruption during storage failure.

SAMPLE

Hardware

VMs use simulated hardware, and Hyper-V uses this virtual hardware to mediate access to actual hardware. Depending on the scenario, you might not need to use all available simulated hardware.

Generation 1 VMs have the following hardware by default:

- **BIOS.** Virtual hardware that simulates a computer's BIOS. You can configure a VM to switch Num Lock on or off. You can also choose the startup order for a VM's virtual hardware and start a VM from a DVD drive, an IDE device, a legacy network adapter, or a floppy disk.
- **Memory.** You can allocate memory resources to a VM. An individual VM can allocate as much as 1 TB of memory. You can also configure the Dynamic Memory feature to allow for dynamic memory allocation based upon resource requirements.
- **Processor.** You can allocate processor resources to a VM. You can allocate up to 64 virtual processors to a single VM.
- **IDE controller.** A VM can support only two IDE controllers and, by default, allocates two IDE controllers to a VM: IDE controller 0 and IDE controller 1. Each IDE controller can support two devices. You can connect VHDs or virtual DVD drives to an IDE controller. The boot device must be connected to an IDE controller if starting from a hard-disk drive or DVD-ROM. IDE controllers are the only way to connect VHDs and DVD-ROMs to VMs that use operating systems that don't support integration services.
- **SCSI controller.** You can use SCSI controllers on VMs you deploy with operating systems supporting integration services. SCSI controllers allow you to support up to 256 disks by using

four controllers with a maximum of 64 connected disks each. You can add and remove virtual SCSI disks while a VM is running.

- **Network adapter.** Network adapters specific to Hyper-V represent virtualized network adapters. You can only use network adapters with supported VM guest operating systems that support integration services.
- **COM port.** A COM port enables connections to a simulated serial port on a VM.
- **Diskette drive.** You can map a .vfd floppy disk file to a virtual floppy drive.

Generation 2 VMs have the following hardware by default:

- **Firmware.** UEFI allows all the features of the BIOS in Generation 1 VMs. However, it also allows secure boot, which is enabled by default.
- **Memory.** Same as Generation 1 VMs.
- **Processor.** Same as Generation 1 VMs.
- **SCSI controller.** Generation 2 VMs can use a SCSI controller for a boot device.
- **Network adapter.** Generation 2 VMs support hot add/removal of virtual network adapters.

You can add the following hardware to a VM by editing its properties, and then selecting **Add Hardware**:

- **SCSI controller.** You can add up to four virtual SCSI devices. Each controller supports up to 64 disks.
- **Network adapter.** A single VM can have a maximum of eight Hyper-V–specific network adapters.
- **Fibre Channel adapter.** This adapter allows a VM to connect directly to a Fibre Channel storage area network (SAN). For this adapter, the Hyper-V host should have a Fibre Channel host-bus adapter (HBA) with a Windows Server driver that supports virtual Fibre Channels.

SAMPLE

Management

Use management settings to configure how a VM behaves on a Hyper-V host, including:

- **Name.** Configure the display name of the VM on a Hyper-V host. Doing this doesn't alter the actual VM's computer name.
- **Integration Services.** Configure which VM integration settings are enabled.
- **Checkpoints.** Specify a location for storing VM checkpoints.
- **Smart Paging File Location.** Use this location when Smart Paging is required to start a VM.
- **Automatic Start Action.** Use to manage how a VM responds when a Hyper-V host is powered on.
- **Automatic Stop Action.** Use to manage how a VM responds when a Hyper-V host shuts down gracefully.

Storage options in Hyper-V

Just as a physical computer has a hard disk for storage, VMs also require storage. Hyper-V provides many different VM storage options. If you know which option is appropriate for a given

situation, you can ensure that a VM performs well and doesn't consume unnecessary space or place an unnecessary performance burden on the Hyper-V host server. You need to understand the various storage options for VHDs so you can select one that meets your performance and high-availability requirements.

A key factor when provisioning VMs is to ensure correct VHD placement and storage. Servers that are otherwise well-provisioned with RAM and processor capacity can still experience poor performance if the storage system is overwhelmed or inadequate. You can store VHDs on local disks, a SAN, or Server Message Block (SMB) version 3.0 file shares.

Consider the following factors when you plan VHD file storage:

- **High-performance connection to storage.** You can locate VHD files on local or remote storage. When you locate them on remote storage, you must ensure there's adequate bandwidth and minimal latency between the host and the remote storage. Slow network connections to storage or connections where there's latency result in poor VM performance.
- **Redundant storage.** The volume on which the VHD files are stored should be fault tolerant whether the VHD is stored on a local disk or on a remote network-attached storage (NAS) or SAN device. Hard disks often fail, so it's important that your VM and Hyper-V host remain in operation after a disk failure. Replacing failed disks shouldn't affect the operation of the Hyper-V host or VMs.
- **High-performance storage.** The storage device on which you store VHD files should have excellent I/O characteristics. Many enterprises use hybrid SSDs in RAID 1+0 arrays to achieve maximum performance and redundancy. Multiple VMs that are running simultaneously on the same storage can place a tremendous I/O burden on a disk subsystem. Therefore, you must ensure that you choose high-performance storage. If you don't, VM performance suffers.
- **Adequate growth space.** If you configure your VHDs to grow automatically, ensure that there's adequate space into which the files can grow. You should carefully monitor growth so that you aren't surprised when a VHD fills the volume that you allocated to host it.

SAMPLE

Fibre Channel support for SANs

The Hyper-V virtual Fibre Channel adapter is a virtual hardware component that you can add to a VM to enable access to Fibre Channel storage on SANs. To deploy a virtual Fibre Channel:

- You must configure the Hyper-V host with a Fibre Channel host bus adapter (HBA) or Fibre Channel over Ethernet (FCoE) network adapter.
- The Fibre Channel HBA must have a driver that supports virtual Fibre Channel.

Virtual Fibre Channel adapters support port virtualization by exposing HBA ports in the guest OS. Doing so allows a VM to access a SAN by using a standard World Wide Name that's associated with the VM. You can deploy up to four virtual Fibre Channel adapters on each VM.

Store VMs on SMB 3.0 file shares

Hyper-V supports storing VM data, such as VM configuration files, checkpoints, and VHD files on SMB 3.0 file shares. The file share must support SMB 3.0.

> **Note:** The recommended bandwidth for network connectivity to an SMB file share is 1 gigabit per second (Gbps) or more.

SMB 3.0 file shares provide an alternative to storing VM files on Internet Small Computer Interface (iSCSI) or Fibre Channel SAN devices. When creating a VM in Hyper-V, you can specify a network share as the VM location and the VHD location or attach disks that are stored on SMB 3.0 file shares. You can use .vhd, .vhdx, and .vhds files with SMB 3.0 file shares.

When you use SMB 3.0 file shares, you should separate network traffic that goes to the file shares that contain the VM files. Client network traffic shouldn't be on the same virtual LAN (VLAN) as SMB traffic.

To provide high availability for file shares that store VM files, use Scale-Out File Server (SOFS), which provides redundant servers for accessing a file share. SOFS also provides faster performance than when you're accessing files through a single share, because all servers in the SOFS are active at the same time. Windows Server 2016 and newer can use Storage Quality of Service (QoS) to manage QoS policies for Hyper-V and SOFS. This allows deployment of QoS policies for SMB 3.0 storage.

VHD formats and types

Starting with Windows Server 2016, you can choose three virtual hard disk formats: .vhd, .vhdx, and .vhds.

.vhd

The .vhd format provides the necessary functionality for VM storage, but it lacks several performance-optimization and resiliency features that exist in the other two formats. Additionally, its maximum disk size is limited to 2 TB.

.vhdx

The .vhdx format was introduced in Windows Server 2012 and then enhanced in Windows Server 2016 and newer versions. The .vhdx format provides several advantages compared to the .vhd format, including that it:

- Has a larger maximum disk size of 64 TB.
- Provides built-in protection that helps avoid data corruption resulting from an unexpected power outages.
- Offers larger block sizes on dynamic and differential disks, which results in performance improvements.
- Has more efficient data representation, which leads to smaller file sizes along with a mechanism that facilitates reclaiming unused disk space.

Based on these benefits, we recommend the .vhdx format when implementing VM disk storage on Windows Server 2016 or newer.

SAMPLE

> **Note:** Microsoft supports conversion between .vhd and .vhdx formats. Converting formats results in a new disk with content copied from the source disk. However, before you run a conversion, ensure that you have sufficient disk space to complete it.

.vhds

Windows Server 2016 also introduced the .vhds format, which is optimized for sharing disks among multiple VMs. This format is commonly used to provide shared storage for VMs operating as nodes in a failover cluster.

Virtual hard-disk types in Hyper-V

Windows Server also supports multiple virtual hard-disk types, and each has its own set of strengths and optimal-use cases, including:

- **Fixed size.** Fixed-size VHDs preallocate the underlying storage equal to the disk's full size during disk creation. This approach's primary benefit is that it minimizes fragmentation, which, in turn, enhances performance. Another benefit is a minimized CPU overhead associated with reads and writes, because the need for metadata lookups, which is required when using other disk types, is eliminated.
- **Dynamically expanding.** Dynamically expanding VHDs allocate the underlying storage as required, which optimizes its usage The .vhdx disk format also has the ability to dynamically shrink if you remove data and stop the VM. However, writing data to a dynamically expanding disk requires metadata updates and subsequent lookups, which increases CPU utilization.
- **Differencing.** Differencing VHDs rely on a parent disk for any existing read-only content. Writes are allocated to a dynamically expanding disk. Subsequent reads are serviced from the same disk, which introduces a CPU overhead required to look up the mapping of data blocks. However, in scenarios where multiple VMs rely on the same parent disk, this approach can lead to significant space savings.
- **Pass-through.** Pass-through VHDs map directly to a physical disk or an iSCSI or Fibre Channel LUN rather than a virtual disk file. While this might improve the impact of storage I/O on CPU usage, it also introduces several limitations in VM migration scenarios. This is because it's not easy, or sometimes not possible, to ensure the same pass-through disk configuration on more than one physical server.

Consider the following factors when choosing virtual-disk formats and types:

- When running Hyper-V on Windows Server 2016 or mewer, we recommend using VHDs in the .vhdx format.
- Dynamically expanding VHDs in the .vhdx format offer almost the same performance level as those in the .vhd format and are supported for production workloads.
- Dynamically expanding VHDs report free space availability based on the maximum size set during their creation, not the actual amount of underlying storage available. Therefore, it's important to monitor the availability of physical storage when using dynamically expanding VHDs.

SAMPLE

- While it's possible to link multiple differencing disks, the resulting performance decreases as the number of linked disks increases.
- Any modification to a parent VHD automatically invalidates all corresponding differencing disks.

Shared VHDX and VHD Set files

In some scenarios, you might need to share a single VHD between multiple VMs, such as when you incorporate high availability using VMs configured to support failover clustering. Hyper-V in Windows Server 2022 supports two methods for sharing a VHD between multiple VMs, shared VHDs, and VHD Sets.

Shared VHDs

Windows Server 2012 R2 and newer operating systems support the ability to create a VHD in the .VHDX format and connect the file to the SCSI controllers of multiple VMs. The shared VHDs must be stored on cluster-shared volumes or a file server with SMB version 3.0 file-based storage.

Using a shared VHD with guest failover clustering does pose limitations, such as:

- The .VHDX disk format doesn't support resizing the file while the cluster is running. You must shut down the cluster to resize the disk.
- Shared VHDs don't support Hyper-V Replica to replicate the VM failover cluster.
- Backing up the VMs from the host machine isn't supported.

To address these limitations, Windows Server 2016 and newer provides the ability to create VHD Sets.

SAMPLE

VHD Sets

A VHD Set provides the next evolution for sharing virtual disk files with multiple VMs. Consider the following factors involved in using a VHD Set:

- Requires Windows Server 2016 or newer, Windows 10 or newer.
- Uses the .VHDS file format for the shared disk along with an .AVHDX file that's used as a checkpoint file.
- Supports both fixed size and dynamically expanding disk type.
- Supports dynamic resizing, backup at the host level, and the ability to use Hyper-V replica.

> **Tip:** You can convert from a Shared VHDX to a VHD Set by using the `Convert-VHD` PowerShell cmdlet.

Overview of Hyper-V networking

Networking in Hyper-V consists of two primary components: a virtual network adapter and a virtual switch. You configure the virtual network adapter on the VM and connect it to a port on the virtual switch to communicate on a network.

Hyper-V supports three basic network configurations that accommodate the most common virtualization scenarios. You can further customize these scenarios by leveraging performance, resiliency, and scalability capabilities included in the Hyper-V network-related feature set. These include VLAN integration and Switch Embedded Teaming (SET).

A Hyper-V virtual switch is a software-based, layer-2 Ethernet network switch, and its functionality becomes available when you install the Hyper-V server role. Similar to its physical counterparts, the purpose of a Hyper-V virtual switch is to enable connectivity between different networking endpoints. In the case of a Hyper-V switch, these endpoints can include the local Hyper-V host, its VMs, and physical networks to which the Hyper-V host is connected.

Hyper-V supports the three types of virtual switches, which Table 2 lists:

Table 2: Virtual switches

Type	Description
External	External virtual switches provide connectivity between VMs on a Hyper-V host, between the VMs and the Hyper-V host itself, and between the VMs and a physical network. Connectivity relies on a wired or wireless network adapter attached to the Hyper-V host. The adapter can be shared with the host or used exclusively by the VMs that connect to the switch.
Internal	Internal virtual switches provide connectivity between the VMs on the Hyper-V host and between the VMs and the Hyper-V host itself. Connectivity doesn't rely on a physical network adapter.
Private	Private virtual switches provide connectivity between the VMs on the Hyper-V host. Similar to internal virtual switches, connectivity doesn't rely on a physical network adapter.

SAMPLE

When configuring virtual switches, you can also assign a VLAN ID to associate with the management OS. When used with an external switch, you can extend existing VLANs on an external network to include VMs connected to the same switch.

> **Note:** You can use an internal switch to implement NAT for VMs on a Hyper-V host. NAT maps an external IP address assigned to the host's physical network adapter to a set of internal IP addresses assigned to VMs. As a result, NAT provides VMs with access to external network resources without exposing them directly via an external IP address.

Virtual switches also support extensions that further enhance their capabilities, including:

- **Network Driver Interface Specification (NDIS) Capture.** Enables you to capture network traffic that traverses the virtual switch.
- **Windows Filtering Platform.** Enables you to filter traffic that traverses the virtual switch.

Networking features for Hyper-V

Windows Server offers a wide range of features that further enhance the core functionality built into the Hyper-V networking model. Most of these features increase networking performance by minimizing utilization of server-processing resources and maximizing network connectivity's reliability and resiliency. They also tend to minimize the administrative overhead required to implement and maintain them. These Windows Server features include QoS, Dynamic Virtual Machine Multiple Queues (VMMQ), Remote Direct Memory Access (RDMA), and SET.

QoS

QoS is a set of networking and storage technologies that enables you to control network-traffic flow based on its characteristics, resulting in optimized system functionality and workload performance. QoS also facilitates implementing provisions that allow you to satisfy performance requirements of workloads that rely on shared infrastructure. To accomplish this, QoS:

- Monitors and measures the utilization of shared resources.
- Detects changing resource utilization.
- Prioritizes or throttles designated types of workloads, depending on a configuration you specify.

For example, to address temporary network congestion, QoS would prioritize traffic for latency-sensitive services by throttling latency-insensitive ones.

SAMPLE

Dynamic VMMQ

Dynamic VMMQ (also known as *d.VMMQ*) is a feature of the Hyper-V network stack. It dynamically manages processing of VM network traffic in response to changes in network conditions and utilization of system resources. Dynamic VMMQ optimizes the Hyper-V host efficiency and accommodates seamless processing of burst workloads.

When network throughput is low, Dynamic VMMQ coalesces traffic received on a VM network adapter to utilize as few processors as possible. This is referred to as *queue packing,* which is beneficial from the compute standpoint because there's an overhead associated with managing packet distribution across multiple processors. To proactively prepare for sudden bursts of network traffic, Dynamic VMMQ preallocates resources to idle workloads, which is known as *queue parking*.

Dynamic VMMQ is enabled by default, and it doesn't require any explicit setup beyond installing a supporting driver. It'll autotune the existing workload to maintain optimal throughput for each VM.

RDMA support for virtual switches

RDMA is a networking technology that provides high-throughput, low-latency communication to minimize CPU usage. RDMA accomplishes this by leveraging the capabilities of physical network adapters to perform direct data transfers to and from a computer's memory without relying on the CPU hardware.

The original RDMA implementation in Windows Server 2012 supported RDMA only when using physical adapters without a virtualization layer. Windows Server 2016 extends this support to Hyper-V external switches associated with RDMA adapters. You could also use SET to combine multiple RDMA adapters into a team for bandwidth aggregation and failover protection. Starting with Windows Server 2019, you can configure RDMA in Hyper-V VMs.

> **Note:** Windows Server 2022 introduces another performance-optimization technology referred to as *Receive Segment Coalescing* (RSC) built into Hyper-V switches. Its role is to coalesce incoming network packets and process them as a single segment, which increases the efficiency of traffic delivery for external, internal, and private switches.

SET

Most production deployments require some degree of high availability and performance provisions. These provisions apply to every infrastructure component, including network connectivity. To account for such requirements, server hardware typically includes multiple physical network adapters. To deliver the expected functionality, such adapters are then configured as a network interface card (NIC) teams through software or firmware to provide failover and load-balancing functionality.

Microsoft introduced built-in NIC teaming in Windows Server 2012 in the form of the load balancing and failover (LBFO) OS component. LBFO enables the configuration of up to 32 network adapters as a team, with automatic failover and bandwidth aggregation. Adapters can be of a different make and model and support different network speeds.

LBFO-based teaming is dynamic, which means that it continuously monitors network data flows across all team members and redistributes them if needed to maximize load-balancing behavior.

Starting with Windows Server 2016, Microsoft also provides an alternative teaming solution in the form of SET, which integrates NIC teaming functionality directly into the Hyper-V virtual switch. It also enables grouping between one and eight physical Ethernet network interfaces into a single team.

SET architecture doesn't expose individual team interfaces. Instead, the Hyper-V switch ports provide automatic load balancing and fault tolerance if a network-interface failure occurs.

SET requires all network interfaces that are members of the same team to have a matching manufacturer, model, firmware, and driver. However, it offers better stability and performance than LBFO. It also addresses several of LBFO's limitations that stem from its incompatibility with more recently developed networking technologies such as RDMA and Dynamic VMMQ.

SAMPLE

You must create a SET team at the same time that you create the Hyper-V virtual switch. You can accomplish this by using the `New-VMSwitch` PowerShell cmdlet with the *EnableEmbeddedTeaming* parameter, as the following example depicts:

```
New-VMSwitch -Name TeamedvSwitch -NetAdapterName "NIC 1","NIC 2" -
EnableEmbeddedTeaming $true
```

Manage VM checkpoints

One of the significant advantages of virtualization is that you can save the current state of VMs almost instantly. On the Hyper-V platform, this capability is referred to as *checkpoints*.

> **Note:** This capability originally was referred to as *snapshots*, but it was changed starting with Windows Server 2012 R2. You might still occasionally encounter the term *snapshot* when dealing with checkpoints, such as when running PowerShell checkpoint-related cmdlets such as `Export-VMSnapshot`.

When you create a checkpoint, Hyper-V creates a differencing disk, in the form of a .avhd file, which stores changes from the previous, most recent checkpoint, or in the case of an initial checkpoint, from the parent VHD.

Windows Server 2016 introduced two types of checkpoints: standard and production:

- **Standard checkpoints.** Contain the state of a VM's disks and memory at the point in time when the checkpoint was initiated. Using such checkpoints for restoring a VM and its contents might result in data-consistency issues. Stateful distributed workloads that have their internal change-tracking mechanism are particularly vulnerable.
- **Production checkpoints.** Rely on Volume Shadow Copy Service (VSS) on Windows and File System Freeze on Linux. These software components attempt to quiesce all workload activity prior to creating a checkpoint, which makes such checkpoints suitable for a reliable restore. To restore a production checkpoint, the VM must be turned off. This checkpoint type is used by default.

You can create a checkpoint from the **Hyper-V Manager** console or the **Actions** pane of Virtual Machine Connection. Hyper-V supports up to 50 checkpoints per VM. When creating a checkpoint, be sure to determine whether the target VM has any external dependencies that might lead to data-integrity issues when the checkpoint is restored. If such dependencies exist, consider creating checkpoints for that VM and their dependencies simultaneously.

The most common checkpoint-management tasks involve applying, renaming, deleting, and exporting checkpoints:

- Applying a checkpoint reverts the VM's status to the point in time at which it was created, and it preserves any existing points. You also can create a checkpoint that captures the VM's current state before you revert to an earlier checkpoint.
- Renaming a checkpoint allows you to apply a meaningful naming convention that describes the corresponding VM state. Checkpoint names are limited to 100 characters.

SAMPLE

- Deleting a checkpoint means its content is either discarded or merged into the previous checkpoint or parent VHD. For example, if you delete the most recent checkpoint, its content is discarded. However, if you delete the previous checkpoint, the content of the corresponding differencing VHD is merged with its parent to preserve the integrity of the most recent checkpoint.
- Exporting a checkpoint produces a fully functional export, which you can move and import to another location. This is a viable backup method, allowing you to restore a VM from a checkpoint.

Import and export VMs

Hyper-V import and export tasks enable you to transfer VMs between Hyper-V hosts and create image-based VM backups.

An export of a VM captures all its VHD files, configuration files, and any existing checkpoint files. The VM being exported can be in a running or stopped state.

Starting with Windows Server 2016, you can perform a VM import from an exported VM, and from copies of VM configuration files, VHD files, and checkpoints. This is helpful in recovery scenarios in which an OS volume failed but other VM files remained intact.

When using the **Import Virtual Machine Wizard**, you also have the option to fix incompatibilities that might exist between the source and target Hyper-V hosts, such as different virtual switches or processors.

When performing an import, you need to choose one of the following options:

- **Register the VM in-place (use the existing unique ID).** Recreates a VM without changing the location of its files or its ID.
- **Restore the VM (use the existing unique ID).** Copies the VM files either to their original file system path or to another location you specify, and reuses the original ID. This option is equivalent to restoring a VM from a backup.
- **Copy the VM (create a new unique ID).** Is similar to a restore, but the imported machine gets assigned a new, automatically generated ID.

When exporting a VM, you have two options:

- **Export a specific checkpoint.** Enables you to create an exported VM as it existed at the point of checkpoint creation. The exported VM will have no checkpoints. Select the checkpoint to export, and then select **Export**.
- **Export a VM with all checkpoints**. Exports the VM and all checkpoints that are associated with the VM. From the **Virtual Machine** list in Hyper-V Manager, select the VM to export, and then select **Export**.

> **Note:** Hyper-V in Windows Server 2016 and newer supports exporting VMs and checkpoints while a VM is running.

SAMPLE

Demonstration: Create and manage a VM

For high-level demonstration steps, refer to the **Notes**, which are in the accompanying Microsoft PowerPoint presentation.

Lesson 3: Secure virtualization in Windows Server

Most organizations use Hyper-V to virtualize network-infrastructure services that traditionally are hosted on physical servers. Virtualization provides many benefits related to consolidation, portability, and ease of management. However, these benefits also introduce unique security concerns. Unlike a physical server that might be protected in a secure datacenter, VMs can simply be exported, copied offsite, and then imported to run on any Hyper-V host server. These VMs, which might consist of services such as domain controllers (DCs), human-resource systems, or sensitive file servers, often contain confidential information that must be protected from malicious use.

Hyper-V supports the concept of a *guarded fabric* to provide a more secure environment for VMs. This lesson introduces the concept of implementing a guarded fabric, including the Host Guardian Service (HGS), guarded host servers, and shielded VMs.

By completing this lesson, you'll achieve the knowledge and skills to:

- Describe the guarded fabric.
- Describe attestation modes in a guarded fabric.
- Explain the HGS.
- Explain the types of protected VMs in a guarded fabric.
- Describe the general process for creating a shielded VM.
- Describe the process for when a shielded VM is powered on in a guarded fabric.

SAMPLE

What is a guarded fabric?

A guarded fabric in Hyper-V is a security solution used to help protect VMs against inspection, theft, and tampering from either malware or malicious system administrators. The VMs that are part of a guarded fabric are called *shielded VMs* and are protected both at rest and during runtime. A shielded VM is encrypted and can only run on healthy and approved hosts within the guarded fabric infrastructure.

The guarded fabric provides several security benefits to a virtual infrastructure, such as:

- **Secure and authorized Hyper-V hosts.** Hyper-V hosts that are part of a guarded fabric are *guarded hosts*, which can run a shielded VM only if it can prove that it's in a known, trusted state. A guarded host provides health information and requests permission to start a shielded VM from an external authority called the HGS.

- **Verification that a host is in a healthy state.** The HGS performs attestation, which means that the service can measure the health of a guarded host and provide certification that the host is healthy and authorized to run the shielded VM.
- **Secure method to release keys to healthy hosts.** After a guarded host is verified as healthy and authorized, the HGS releases a secure key, which is used to unlock and start a shielded VM.

To summarize, a guarded fabric is comprised of the following components:

- **Guarded Hyper-V hosts.** You might have one or more guarded Hyper-V hosts running Windows Server Datacenter.
- **Host Guardian Service.** Typically, a three-node cluster running the HGS server role.
- **Shielded virtual machines.** A VM that has a virtual trusted platform module (TPM) and is encrypted using BitLocker.

> **Note:** A guarded fabric can run a normal VM with no protection, similar to a standard Hyper-V environment. You can also implement encryption-supported VMs that are secured by encryption but don't have the same restrictions in place as shielded VMs.

Tools used to automate and manage a guarded fabric

To help manage and automate processes within the guarded fabric infrastructure, you can use the following tools:

- **SCVMM.** Provides a unified management solution for virtualization hosts, virtual networking, and storage so you can effectively maintain a virtual enterprise environment.
- **Windows Azure Pack.** A graphical web-portal interface that enables you to manage a guarded fabric and shielded VMs.
- **PowerShell.** If you don't have a fabric manager such as SCVMM, you can use PowerShell to create the components needed for a new shielded VM manually. You can also use PowerShell to provision the shielded VM to a guarded host.

SAMPLE

> **Learn more:** For more information about Guarded fabric and shielded VMs, refer to Guarded fabric and shielded VMs overview.

Attestation modes for guarded fabric

Hyper-V hosts can only run protected VMs in the guarded fabric infrastructure after HGS has validated them. *Attestation* is the process of evaluating and validating Hyper-V hosts. Protected VMs can't be powered-on or live migrated to a Hyper-V host that hasn't yet attested or has failed attestation.

The HGS supports the following attestation modes:

- **Trusted platform module (TPM)-trusted attestation.** A hardware-based attestation method that offers the strongest protections but does require a more complex configuration and higher host

hardware requirements. These requirements include TPM 2.0 and UEFI 2.3.1 with Secure Boot enabled. A guarded Hyper-V host is approved and validated based upon its TPM identity, Measured Boot sequence, and code-integrity policies to ensure that it only runs approved code.

- **Host key attestation.** This mode is based upon asymmetric key pairs and is used when existing Hyper-V host machines don't support TPM 2.0. A guarded Hyper-V host is approved and validated based upon key possession.

> **Note:** Windows Server 2016 included another mode, *Admin-trusted attestation*. However, beginning with Windows Server 2019, it's been deprecated.

Host Guardian Service overview

The HGS contains two important components that ensure the validity of guarded hosts and protection for VMs:

- **Attestation Service.** Ensures that only trusted Hyper-V hosts can run protected VMs.
- **Key Protection Service (KPS).** Provides the keys necessary to power on protected VMs and to permit live migration to other guarded Hyper-V hosts.

The HGS provides authority for the guarded fabric and helps to enforce and assure the following:

- Protected VMs contain BitLocker encrypted disks.
- Protected VMs use BitLocker to help protect both the OS disk and data disks.
- The virtual TPM helps protect the BitLocker keys needed to boot the VM and decrypt the disks.
- Shielded VMs are deployed from trusted template disks and images.
- When deploying new shielded VMs, the VM owner can specify which template disks they trust.
- Shielded template disks have signatures computed when their content is deemed trustworthy. The disk signatures are then stored in a signature catalog and securely provided to the fabric when creating shielded VMs.
- During the provisioning of shielded VMs, the signature of the disk is computed again and compared to the catalog's trusted signatures. If the signatures match, the shielded VM is deployed. If the signatures don't match, the shielded template disk is deemed untrustworthy, and deployment fails.
- Passwords and other secrets are protected when a shielded VM is created.
- When creating VMs, it's necessary to ensure that VM secrets, such as the trusted disk signatures, Remote Desktop Protocol (RDP) certificates, and the password of the VM's local Administrator account, are not divulged to the fabric. These secrets are stored in an encrypted file called a *shielding data file* (.PDK file), which is protected by certificate keys and uploaded to the fabric.
- When you create a shielded VM, you must select which shielding data file to use so that these secrets can only be provided to the trusted components within the guarded fabric.
- Control of where the shielded VM can be started.

SAMPLE

- The shielding data file also contains a list of the guarded fabrics on which a particular shielded VM is permitted to run. This is useful in cases where a shielded VM typically resides in an on-premises private cloud but might need to be migrated to another public or private cloud.
- The target cloud or fabric must support shielded VMs, and the shielded VM's shielding data file must permit that fabric to run it.

Factors to consider when planning an HGS implementation

You should carefully consider the following factors when you're planning an HGS implementation:

- **Hardware requirements.** The HGS can be run on both physical computers and VMs. However, if you want to run HGS as a three-node cluster, we strongly recommend that you use three physical servers.
- **OS requirements.** If you plan to use TPM trusted attestation, the HGS can run on Windows Server 2022, 2019, or Windows Server 2016 Standard or Datacenter edition. Host key attestation requires Windows Server 2019 Standard or Datacenter edition operating with v2 attestation, which requires a TPM certificate to be present to add the TPM identifier information to the HGS.
- **Certificate requirements.** When you deploy HGS, you'll be asked to provide signing and encryption certificates that KPS uses to help protect the sensitive information needed to start a shielded VM. These certificates never leave the HGS and are only used to decrypt shielded VM keys when the host on which they're running has proven it's healthy. Although not required, you should obtain the certificates from a trusted certificate authority.
- **Server Roles required.** The HGS and supporting server roles are installed on the server. The default installation establishes the server in a new Active Directory forest dedicated to HGS, which ensures that all sensitive key information is as secure as possible from the rest of the organizational network environment.

SAMPLE

Types of protected VMs in a guarded fabric

A guarded fabric can run the following types of VMs:

- **Shielded VM.** Used in fabrics where the data and state of the VM must be protected from both fabric administrators and untrusted software that might be running on the Hyper-V hosts.
- **Encryption-supported VM**. Used where the fabric administrators are fully trusted. For example, an enterprise might deploy a guarded fabric to ensure VM disks are encrypted at rest for compliance purposes. Fabric administrators can continue to use convenient management features, such as VM console connections, PowerShell Direct, and other day-to-day management and troubleshooting tools.
- **Normal VM.** Unprotected VMs can be run in the fabric if needed. However, a normal VM doesn't offer any of the security benefits of a shielded or encryption-supported VM.

Table 3 provides a summary of the differences between encryption supported and shielded VMs:

Table 3: Comparison of encryption-supported and shielded VMs

Capability	Encryption-supported	Shielded
Secure boot	Yes. Required but configurable.	Yes. Required and enforced.
Virtual TPM	Yes. Required but configurable.	Yes. Required and enforced.
Encrypt VM state and live-migration traffic	Yes. Required but configurable.	Yes, required but not enforced.
Integration components	Configurable by a fabric admin.	Specific components blocked, such as PowerShell Direct (enabled for Windows Server v1803 and newer) and data exchange.
Virtual Machine Connection, Human Interface Devices (HIDs) such as a keyboard and mouse	On and can't be disabled.	Enabled for hosts beginning with Windows Server v1803, but disable on earlier hosts.
COM/Serial ports	Supported.	Disabled (can't be enabled).
Attach a debugger to the VM process	Supported.	Disabled (can't be enabled).

SAMPLE

> **Note:** You must ensure that you configure both encryption-supported and shielded VMs as Generation 2 VMs.

General process for creating shielded VMs

To create a shielded VM, there are three general steps:

1. Create a shielded VM template disk.
2. Create a shielded data file.
3. Deploy the shielded VM.

Create a shielded VM template disk

A VM template disk is used as a basis for all future VMs deployed within the fabric. A shielded VM template disk provides additional security by hiding all confidential information, such as passwords, certificates, and similar information from fabric administrators.

To create a template disk, the first step is to prepare an OS disk using your standard processes. The goal is to set up a VM with a blank VHDX, and then install an OS onto that disk.

The disk you configure must meet several requirements, which Table 4 lists:

Table 4: Requirements for a template disk

VHDX requirement	Description
Must be a GUID Partition Table (GPT) disk.	Needed for Generation 2 VMs to support UEFI.
The disk type must be **Basic**.	Disks can't be of type **Dynamic** as BitLocker doesn't support dynamic disks.
The disk must have at least two partitions.	One partition must include the drive on which you'll install Windows. BitLocker encrypts this drive. The other partition is the active partition, which contains the bootloader and remains unencrypted so the computer can start.
The file system must be NTFS.	Required for BitLocker.
The OS must be supported.	Supported operating systems include Windows 10 or Windows Server 2012 or newer. Must support Generation 2 VMs and the Microsoft Secure Boot process.
The OS must be generalized.	You must generalize the template disk by running **sysprep.exe**.

SAMPLE

> **Note:** After installing the OS, verify and install all of the latest Windows updates, which often are released to improve reliability of the end-to-end shielded process.

Before using the template disk to create shielded VMs, you must prepare the generalized disk and encrypt it with BitLocker. You perform this task by using the **Shielded Template Disk Creation Wizard**, which generates a hash for the disk and adds it to a volume signature catalog (VSC). The VSC is signed using a certificate you specify and is used during the provisioning process to ensure the disk being deployed in the fabric hasn't been altered or replaced with an untrusted disk. Finally, BitLocker is installed on the disk's OS to prepare it for encryption during VM provisioning.

The **Shielded Template Disk Creation Wizard** is part of the **Shielded VM Tools** available from the **Remote Server Administration Tools** feature.

Create a shielded data file

A shielded data file, also called a *provisioning data file* or *PDK* file, is an encrypted file that a VM owner creates to help protect important VM configuration information such as the administrator password, Remote Desktop Protocol (RDP) and other identity-related certificates, and domain-join credentials. Before creating this file, you must create or obtain a shielded template disk as described previously.

You will use the **Shielding Data File Wizard**, which the **Shielded VM Tools** provides from within the **Remote Server Administration Tools** feature.

Deploy a shielded VM

The method used to deploy a shielded VM depends on your management process for the guarded fabric, and common methods include:

- Deploying using SCVMM.
- Using the Windows Azure Pack to provide a web-based portal to simplify shielded VM deployments.
- Using Windows PowerShell.

SAMPLE

Process for powering-on shielded VMs

When you power on a shielded VM within a guarded fabric, several processes occur that validate the guarded host, unlock the shielded VM, and allow the protected VM to start. The general process is:

1. **User requests to start a shielded VM.** When a fabric administrator attempts to start a shielded VM, the guarded host can't power on the VM until it attests that it's healthy.
2. **Host requests attestation.** To prove that it's healthy, the guarded host requests attestation, and then the HGS determines the mode of attestation. The mode used might be either TPM-trusted attestation or host-key attestation.
3. **Attestation succeeds or fails.** The attestation mode determines which checks are needed to successfully ensure the host is healthy. The host's TPM identity, boot measurements, and code integrity policy are validated with TPM-trusted attestation. With host-key attestation, only registration of the host key is validated.
4. **Attestation certificate sent to host.** If attestation is successful, a health certificate is sent to the host, and the host is considered *guarded* and authorized to run shielded VMs. The host uses the health certificate to request the KPS to release the keys needed to work with shielded VMs securely.
5. **Host requests VM key.** Guarded hosts must request the keys from the KPS on the HGS. To obtain the necessary keys, the guarded host provides the current health certificate and an encrypted secret file that only the KPS can decrypt.

6. **Key is released.** The KPS examines the health certificate that the guarded host provides to determine its validity. The certificate must not have expired, and KPS must trust the attestation service that issued it.
7. **Key is returned to the host.** If the health certificate is valid, the KPS attempts to decrypt the provided secret from the guarded host and then securely returns the keys needed to power on the VM.
8. **Host powers on shielded VM.** The guarded host can now unlock and power on the shielded VM.

Lesson 4: Containers in Windows Server

Windows Server 2022 supports the development, packaging, and deployment of apps and their dependencies in Windows containers. You can use container technology to package, provision, and run applications across diverse environments on-premises or in the cloud. Windows containers provide a lightweight and isolated virtualization environment at the operating system (OS) level to make apps easier to develop, deploy, and manage.

By completing this lesson, you'll achieve the knowledge and skills to:

- Describe containers and how they work.
- Explain the difference between containers and VMs.
- Describe the difference between Process Isolation and Hyper-V isolation modes.
- Describe Docker and how it's used to manage Windows containers.
- Pull and push container base images.
- Understand the process for running a Windows container.
- Explain how to manage containers using the Windows Admin Center.
- Deploy Windows containers by using Docker and Windows Admin Center.

SAMPLE

What are containers?

Traditionally, a software application is developed to run only on a supported processor, hardware, and OS platform. Software applications typically can't move from one computing platform to another without extensive recoding to support the intended platform. With many diverse computing systems, a more efficient software-development and management platform is often needed to support portability between multiple computing environments.

A *container* is used to package an application with all its dependencies and abstract it from the host OS in which it is to run. A container is isolated not only from the host OS, but also from other containers. Isolated containers provide a virtual runtime, thereby improving the security and reliability of the apps that run within them.

The benefits of using containers include:

- **The ability to run anywhere.** Containers can run on various platforms such as Linux, Windows, and Mac operating systems. You can host them on a local workstation, on servers in on-premises datacenters, or provisioned in the cloud.

- **Isolation.** To an application, a container appears to be a complete OS. The CPU, memory, storage, and network resources are virtualized within the container isolated from the host platform and other applications.
- **Increased efficiency.** You can quickly deploy, update, and scale containers to support a more agile development, test, and production life cycle.
- **A consistent development environment.** Developers have a consistent and predictable development environment that supports various development languages such as Java, .NET, Python, and Node. Developers know that no matter where the application is deployed, the container will ensure that the application runs as intended.

How do containers work?

The processor in a standard Windows computer has two different modes: a user mode and a kernel mode. Core OS components and most device drivers run in kernel mode, whereas applications run in user mode.

When you install container technology on a computer, each container creates an isolated, lightweight silo for running an app on the host OS. A container builds upon and shares most of the host operating system's kernel to gain access to the file system and registry.

Each container has its own copy of the user mode system files, isolated from other containers and the host's own user mode environment.

A container is created from a container image. You can imagine a container image as a set of files organized in layers, where each layer contains a set of files that, when overlaid together, represent your container image. These images can be located on your local server or some remote container store. The container image consists of the user mode operating system files needed to support your app, any runtimes or dependencies for your app, and any other miscellaneous configuration files your app needs to run properly. Container base image templates provide a foundational layer of OS services used by the containerized app that aren't provided (or restricted) from the host's kernel mode layer.

SAMPLE

Container-related terms

The following are common terms that relate to containers:

- **Container host.** Consists of the physical computer or VM that's installed with the Windows containers feature.
- **Sandbox.** Consists of all changes made to a container, including:
  - File system.
  - Registry.
  - Software installations.
- **Container image.** Enables you to capture changes made to a container in its sandbox. For example, if you install an app in a container and capture its image, you can create new containers based on that new image. Each will have the app installed and include any related changes.

- **Container OS image.** Provides the OS environment and is immutable. This image is the first layer of potentially several image layers that make a container.
- **Container repository.** Stores the container image and its dependencies when you make a container image.
- **Kubernetes**. An open-source orchestration software for deploying, managing, and scaling containers. When you install applications that span multiple containers deployed over multiple servers, it might be difficult to manage this infrastructure. To address this, Kubernetes provides an open source API that controls how and where those containers will run.

> **Learn more:** For more information about containers in Windows Server, refer to Windows containers.

Containers vs. VMs

Both VMs and containers are virtualization technologies that provide isolated and portable computing environments for applications and services.

As previously described, containers build upon the host operating system's kernel and contain an isolated user mode process for the packaged app. This helps to make containers very lightweight and quick to launch.

SAMPLE

Figure 1 and Figure 2 display the architectural differences between containers and VMs:

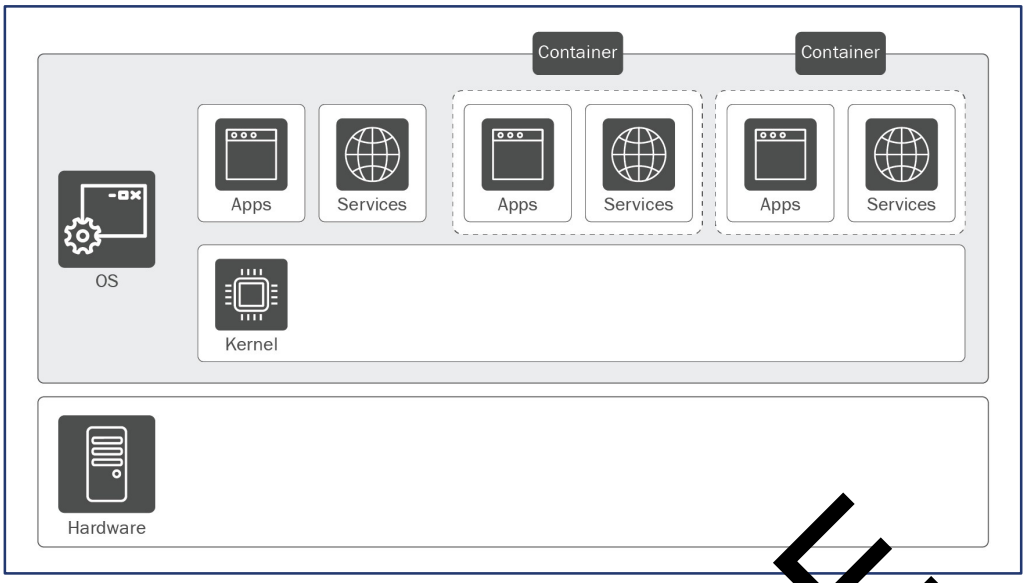
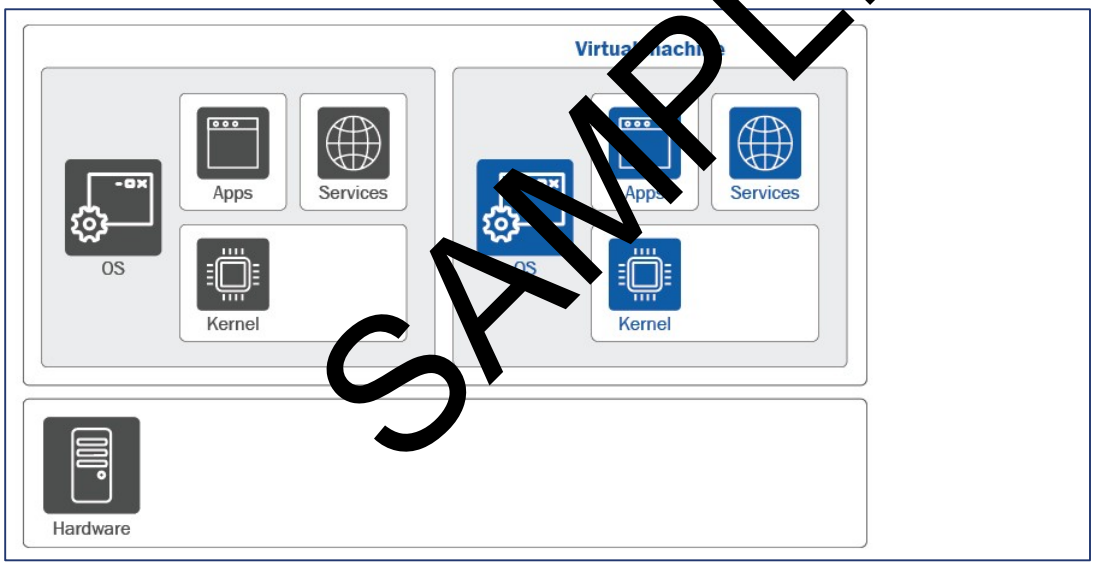


Figure 1: Container architecture



SAMPLE

Figure 2: VM architecture

VMs simulate an entire computer, including the virtualized hardware, OS, user mode, and its own kernel mode. A VM is quite agile and provides tremendous support for applications; however, VMs tend to be large and can take up a lot of resources from the host machine.

Table 5 summarizes the similarities and differences between containers and VMs:

Table 5: Containers versus VMs

Feature	VM	Container
Isolation	Provides complete isolation from the host OS and other VMs.	Provides lightweight isolation from the host and other containers.
OS	Runs a complete OS, including the kernel.	Only runs the user-mode portion of an OS.
Guest compatibility	Is able to run any supported OS inside the VM.	If running in Process isolation mode, you must run on the same type of kernel (OS) as the host; Hyper-V isolation mode provides more flexibility in running non-Windows containers on Windows hosts.
Deployment	Is deployed using Hyper-V manager or other VM management tools.	Deployed and managed using Docker. Deploy multiple containers using an orchestrator such as Azure Kubernetes Service.
Persistent storage	Uses VHD files or an SMB share.	Uses Azure disks for local storage and Azure files (SMB shares) for storage shared by multiple containers.
Load balancing	Uses Windows failover cluster to move VMs as needed.	Uses an orchestrator to start and stop containers automatically.
Networking	Uses virtual network adapters.	Creates a default NAT network, which uses an internal vSwitch and a Windows component named Windows NAT (WinNAT).

SAMPLE

Use a VM when you:

- Need to manage several operating systems.
- Need to run an app that requires all the resources and services of an entire OS, such as a GUI.
- Need an environment that preserves changes and is persistent.
- Require complete isolation and security.

Use a container when you:

- Need a lightweight application package that quickly starts.
- Need to deploy multiple instances of a single app.
- Need to run an app or process that's nonpersistent on an on-demand basis.
- Need to deploy an app that can run on any underlying infrastructure.

> **Note:** It's common to provision containers within a highly optimized VM to provide enhanced isolation and security. The next topic describes isolation modes, which provide an option to provision container runtime isolation.

Overview of container isolation modes

Containers on the Windows operating system can run in one of two distinct isolation modes: process isolation and Hyper-V isolation. Both modes support identical processes for creating, managing, and running containers. However, there's a difference between the degree of isolation and security between the container, other containers, and the host OS.

Process Isolation

Considered the traditional isolation mode for Windows containers, process isolation allows multiple container instances to run concurrently on a host. When running in this mode, containers share the same kernel and host OS. Each provisioned container features its own user mode to allow Windows and app processes to run in isolation from other containers. When you configure Windows containers to use process isolation, containers can run multiple apps in isolated states on the same computer, but they don't provide security-enhanced isolation, because containers share the same kernel with the host as well as each other.

This is very similar to how Linux containers run. In Windows containers, there is virtualized access to several operating system namespaces. A namespace provides access to information, objects, or resources via a name. A good example of a well-known namespace is the file system. However, there are also other namespaces that are isolated on a per-container basis, such as registry, network ports, process and thread ID space, and others. Containers deployed with process isolation are also called Windows Server containers.

SAMPLE

Hyper-V Isolation

With Hyper-V isolation, each container runs inside a highly optimized VM. The advantage of this mode is that each container effectively gets its own kernel, providing an enhanced level of stability and security. The VM provides an additional layer of hardware-level isolation between each container and the host computer. When deployed, a container using Hyper-V isolation mode starts in seconds, much faster than a VM with a complete Windows OS. Containers deployed with Hyper-V isolation mode are also called Hyper-V containers.

When it comes to management, it is nearly identical to manage process-isolated containers and Hyper-V isolated containers. When using Docker, you need to set the **–isolation** switch to either process or isolation value.

> **Note:** Windows containers running on Windows Server default to using process isolation. Windows containers running on Windows 11 Pro and Enterprise default to running in Hyper-V isolation mode.

Manage containers using Docker

Docker is a collection of open-source tools, solutions, and cloud-based services that provide a common model for packaging or containerizing app code into a standardized unit for software development. Called a Docker container, this standardized unit is software that's wrapped in a complete file system that includes everything it needs to run: code, runtime, and system tools libraries.

At the heart of Docker is the Docker engine, which is a runtime environment that runs on Windows, macOS, or Linux operating systems. The Docker client provides a command-line interface for management.

Docker containers are standards-based, which enables them to run on all major Linux distributions and Microsoft operating systems. In fact, Docker containers can run on any computer, infrastructure, and in the cloud.

With Docker, you can create, remove, and manage containers. You can also browse the Docker Hub to access and download prebuilt images. In most organizations, the most common management tasks that use Docker include:

- Automating the creation of container images using Dockerfile on a Windows OS.
- Managing containers by using Docker.

Windows Server provides a built-in, native Docker daemon for Windows Server hosts, so you can use Docker containers, tools, and workflows in production Windows environments.

SAMPLE

Docker components

It's important you're familiar with basic Docker terminology, including the following:

- **Image.** A stateless collection of root-file system changes in the form of layered file systems that are stacked on one another.
- **Container.** A runtime instance of an image, consisting of:
  - The image.
  - The operating environment.
  - A standard set of instructions.
- **Dockerfile.** A text file that contains the commands you need to build a Docker image.
- **Build.** The process of building Docker images from a Dockerfile and any other files in the directory where the image is being built.
- **Docker toolbox.** A collection of platform tools including:
  - **Docker Engine.** Use for building and running Docker containers.

- **Docker Compose.** Enables you to define a multiple-container app together with any dependencies so that you can run it with a single command.
- **Docker Machine.** Enables you to provision Docker hosts by installing the Docker Engine on a computer in your datacenter or on a cloud provider.
- **Docker client.** Includes a command shell that's preconfigured as a Docker command-line environment.
- **Docker Registry.** Forms the basis for the Docker Hub and Docker Trusted Registry.
- **Docker Swarm.** Enables you to combine multiple Docker Engines into a single virtual Docker Engine.

In addition to the components and tools in the Docker toolbox, the following Docker solutions are also important to understand:

- **Docker Hub.** A cloud-hosted service for registering and sharing your Docker images.
- **Docker Trusted Registry.** Enables you to store and manage images in on-premises environments or in a virtual private cloud.
- **Universal Control Panel.** Enables you to manage Docker apps.
- **Docker Cloud.** Enables you to deploy and manage your Docker apps.
- **Docker Datacenter.** Is an integrated platform for deploying CaaS in on-premises environments and in a virtual private cloud.

> To learn more about Docker Engine on Windows, visit: Configure Docker in Windows.

Run Docker on Windows Server

To install the Container extension and Docker on Windows Server, follow the instructions on Prepare Windows operating system containers.

Overview of management with Docker

You can use Docker to support a container environment. After you install Docker, use the following commands to manage your containers:

- `docker images`. This lists the available images on your container host. You might recall that you use container images as a base for new containers.
- `docker run`. This creates a container by using a container image. For example, the following command creates a container using the default process isolation mode, named IIS, and based on the Windows Server Core container image:

```
docker run --name IIS -it windowsservercore
```

- `docker commit`. This commits the changes you made to a container to a new container image. The commit operation doesn't include data contained in volumes mounted within the container. Note that the container will be paused by default as the new container image is created.

SAMPLE

- `docker stop`. This stops a running container.
- `docker rm`. This removes an existing container.

Overview of Docker Desktop

The Docker Desktop is available for both Mac and Windows 10 or newer desktop environments. This toolset enables you to build and distribute containerized applications and services, and it includes:

- **Docker Engine.** Is a lightweight runtime environment for building and running Docker containers.
- **Docker Compose.** Enables you to define a multiple-container app together with any dependencies to run it with a single command. Docker Compose lets you specify the images your app will use along with any necessary volumes or networks.
- **Docker CLI client.** Includes a preconfigured command shell as a Docker command-line environment used to interface with the Docker Engine.
- **Kubernetes.** Use to manage and orchestrate containerized applications across multiple hosts. It helps to deploy, maintain, and scale applications.
- **Credential Helper.** Use to help safely store Docker login credentials.

The Docker Hub

The Docker Hub is a web-based online library service with which you can:

- Register, store, and manage your own Docker images in an online repository and share them with others.
- Access over 100,000 container images from software vendors, open-source projects, and other community members.
- Download the latest versions of the Docker Desktop.

Docker Hub provides the following major features and functions:

- **Image repositories.** Docker Hub contains images stored in repositories, within which you can work with image libraries to build your containers. Repositories contain:
  - Images.
  - Layers.
  - Metadata about those images.
- **Organizations and teams.** Enables you to create organizations or work groups where you can collaborate.
- **Automated builds.** Help automate the build and update of images from GitHub or Bitbucket, directly on Docker Hub.
- **Webhooks.** Attaches to your repositories and enables you to trigger an event or action when an image or updated image successfully pushes to the repository.

SAMPLE

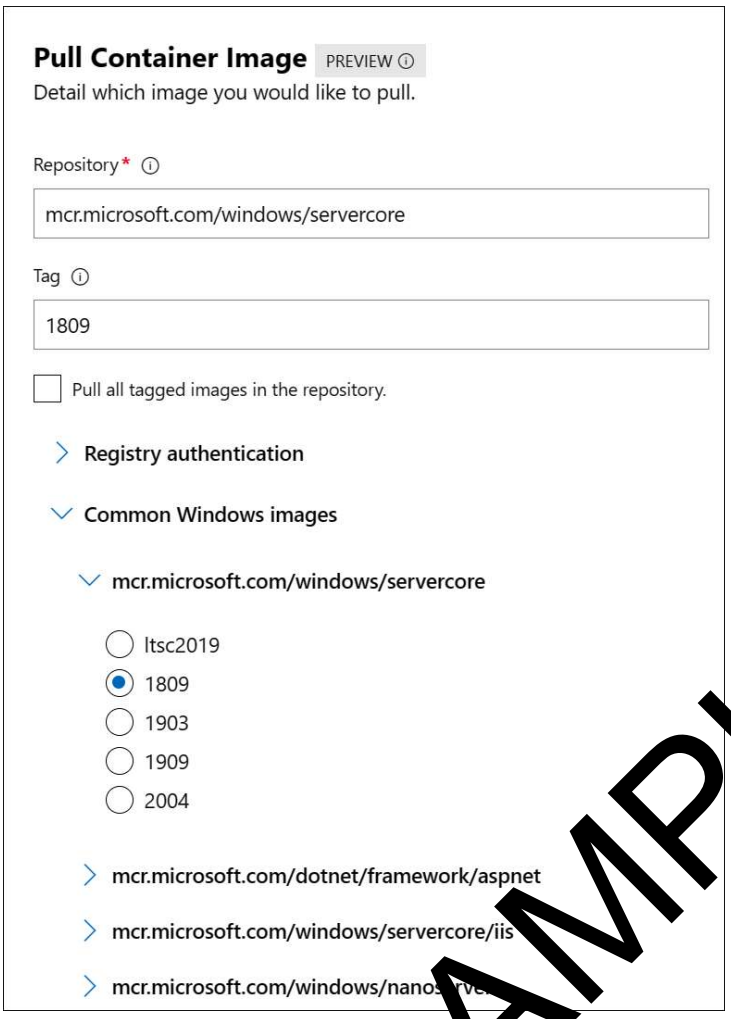
- **GitHub and Bitbucket integration.** Enables you to add the Docker Hub and your Docker images to your current workflows.

Pull and push container base images

After you install Docker Engine, the next step is to pull a base image, which is used to provide a foundational layer of OS services for your container. You can then create and run a container based on the base image.

You can pull container base images either by using Docker or through the Windows Admin Center. When using Windows Admin Center, with the container extension enabled, you should select the **Images** tab within the **Container** extension, and then select **Pull**. The **Pull Container Image** window will appear, where you should provide the URL for the image you want to pull, as well as the tag for the image, as depicted in the following screenshot.

SAMPLE



SAMPLE

Figure 3: Pull Container Image window

For images that are on a Microsoft repository, you don't have to provide user name and password, but if you want to pull a container image from another private repository, you need to provide credentials for authentication under the **Registry authentication** node.

If you're not sure which image to pull, you can get a list of common images from Microsoft in Windows Admin Center. Select the **Common Windows images** dropdown to review a list of base images that are commonly pulled. Select the image you want to pull, and Windows Admin Center will fill in the repository and tag fields.

When you create a container image, it is recommended to push that image to a centralized repository to allow other container hosts or cloud services to pull the image. You can do that by using the **Push** option on the **Containers** pane.

On the **Images** tab in the **Containers** extension of Windows Admin Center, select the image you want to push, and then select **Push**. You can select to push your container image to a generic repository or a repository on Azure Container Registry. For a generic repository, you need to provide a username and password. For Azure Container Registry, the integrated authentication on Windows Admin Center can be used.

A container base image includes:

- The user mode OS files needed to support the provisioned application.
- Any runtime files or dependencies required by the application.
- Any other miscellaneous configuration files the app needs to provision and run properly.

> **Note:** The Windows host OS version must match the container OS version. To run a container based on a newer Windows build, you must ensure that an equivalent OS version is installed on the host. If your host server contains a newer OS version, you can use Hyper-V isolation mode to run an older version of Windows containers. To determine the version of Windows installed, run the `ver` command from the command prompt.

The Windows container base images are discoverable through the Docker Hub and are downloaded from the Microsoft Container Registry (MCR). Besides using Windows Admin Center, you can use the Docker pull command to download a specific base image. When you enter the pull command, you specify the version that matches the version of the host machine.

If you want to pull a 2019 LTSC Server core image, run the following command:

```
docker pull mcr.microsoft.com/windows/servercore:ltsc2019
```

After you download the base images needed for your containers, you can verify the locally available images and display metadata information by entering the following command:

```
docker image ls
```

SAMPLE

Run a Windows container

After you set up your Windows Server environment for containers, you can create and run your first container. As discussed previously, before you run a container, you should pull the appropriate base container image. You can do this by using Windows Admin Center, or by using Docker.

For example, let's presume that you pulled the nanoserver:ltsc2022 container base image. To create and deploy the container image, you should run the following Docker command:

```
docker run -it mcr.microsoft.com/windows/nanoserver:ltsc2022 cmd.exe
```

When you run this command, container will start, and the command prompt context will change to the container. For example, if you want to create a new text file within a container, run the following command:

```
echo "Hello World!" > Hello.txt
```

This command will create a new text file within a container. With the **exit** command, you can leave the context of a container.

If you want to commit changes to the container you ran, run the following command:

```
Docker commit <containerid> helloworld
```

Replace the <containerid> variable with the actual container ID, which you can get by running the following command:

```
Docker ps -a
```

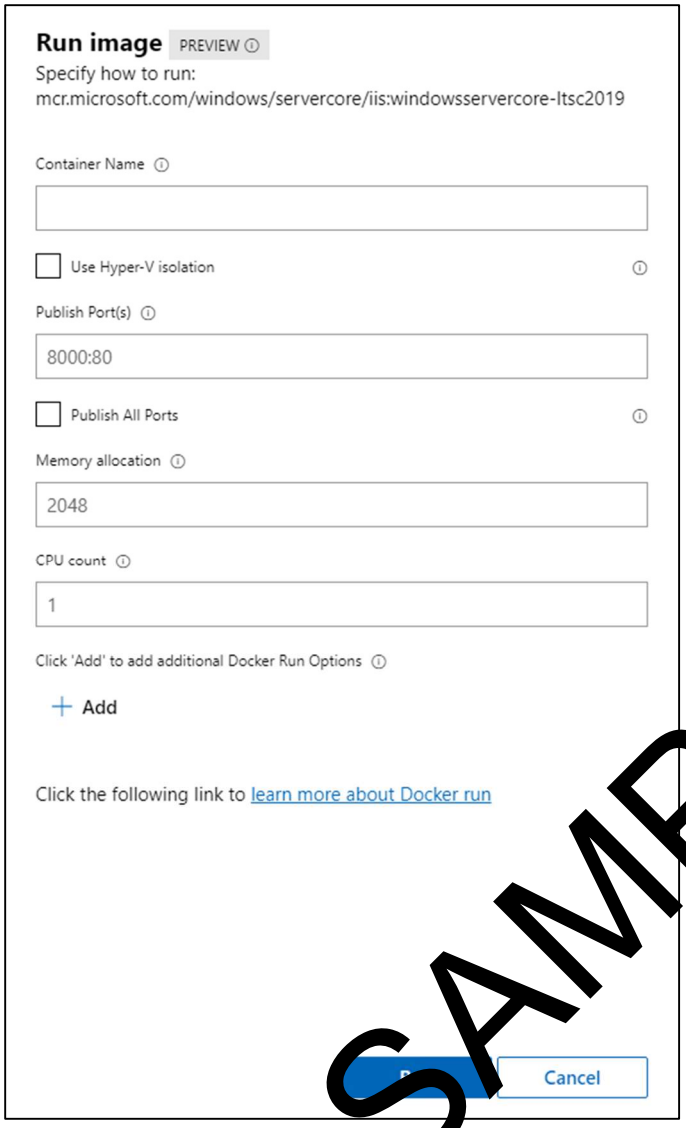
To run a newly created container, run the following command:

```
Docker run –rm helloworld cmd.exe /s /c type Hello.txt
```

The result of this command is a container created from the 'HelloWorld' image, Docker started an instance of cmd.exe in the container, and the cmd.exe read the file and output the contents to the shell. As the final step, Docker stopped and removed the container.

As an alternative to the Docker command line interface, you can use Windows Admin Center to run containers on Windows. After you pull the appropriate container image, as discussed earlier, you should select that image from the **Images** pane, and then select **Run**. This command will open a **Run image** window, which the following screenshot depicts.

SAMPLE



SAMPLE

Figure 4: Run image window

In this window, you can set up the configuration for the container, such as the container name, the isolation type, which ports to publish, and memory and CPU allocation. Additionally, you can add Docker run commands that are not in the UI, such as **-v** for persistent volume.

> **Note**: If you select to use Hyper-V isolation here, you're actually running a Hyper-V container.

Once you've finished the configuration for the container, select **Run**. You can review the status of the running containers on the **Containers** tab, as depicted in the following screenshot.

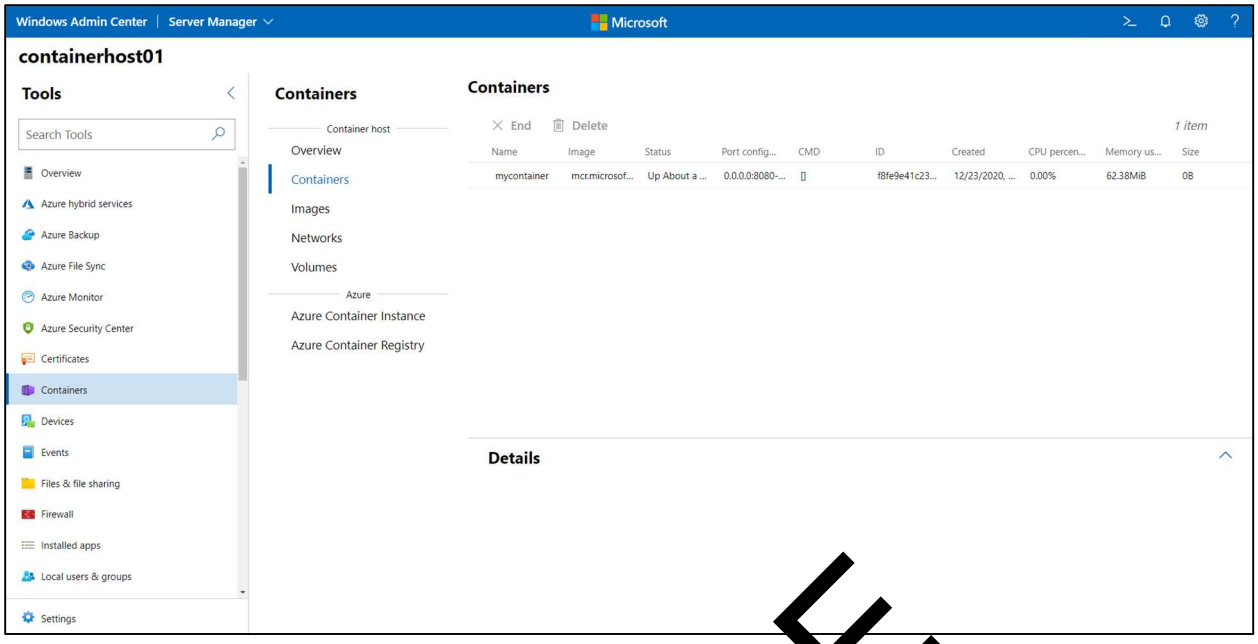


Figure 5: Container status

SAMPLE

Manage containers using Windows Admin Center

Windows Admin Center is a browser-based GUI that you can use to manage Windows servers, clusters, hyperconverged infrastructure (HCI), and Windows 10 or newer PCs. It's a single administrative tool that you can use to perform many of the tasks that were commonly performed using a variety of consoles, tools, or processes. Another powerful aspect of the Windows Admin Center is that it's extensible and supports third-party hardware manufacturers for configuration or monitoring capabilities related to their specific hardware.

After you install the Windows Admin Center, you might need to add additional extensions to allow you to manage the services you intend to use with the tool. You can add extensions by selecting the **Settings** button, and then selecting **Extensions**. Windows Admin Center pulls the latest extension list from its default feed to display the available extensions that can be installed.

Overview of the Containers extension

The Windows Admin Center Containers extension performs remote management and monitoring of the containers running on the targeted host machine and displays the following information and management tabs:

- **Summary.** Provides a summary of the status of the containers, including how many are running, which images are available, the networks that have been configured, and the volumes used.
- **Containers.** Provides an extensive amount of status information in both text and graphical format. You can also obtain details, logs, and events from specific containers and perform tasks such as End or Delete containers.

- **Images.** Displays and provides details on the images available in your local repository. You can also use this tab to delete images from the repository.
- **Networks.** Displays the networks that have been configured for the containers. Includes networking information such as subnet and gateway addresses used for the network connection.
- **Volumes.** Displays volume information used by containers on the host machine.

Demonstration: Deploy containers by using Docker and Windows Admin Center

For high-level demonstration steps, refer to the **Notes**, which are in the accompanying PowerPoint presentation.

Lesson 5: Overview of Kubernetes

Building and managing modern applications using containers requires methods and processes that allow efficient container deployment to both on-premises and cloud-based resources such as Microsoft Azure. Kubernetes is open-source orchestration software used to efficiently deploy, manage, and scale containers in a hosted environment.

In this lesson, we'll introduce Kubernetes and its benefits for managing container technology. By completing this lesson, you'll achieve the knowledge and skills to:

- Describe container orchestration.
- Explain Kubernetes.
- List high-level steps for deploying Kubernetes resources.

SAMPLE

What is Windows container orchestration?

Containers provide many benefits for managing applications and supporting agile delivery environments and microservice-based architectures. However, application components often grow to span multiple containers spread across multiple host servers. If you don't have automation processes in place, managing and operating these as a distributed architecture in an efficient and scalable manner can be difficult.

Automating the processes within a containerized environment is the job of an orchestrator, which automates and manages large numbers of containers and controls how they interact with one another.

A typical orchestrator performs the following tasks:

- **Scheduling.** Finds a suitable machine for running the container when given a container image and a resource request.
- **Affinity/Anti-affinity.** Specifies whether a set of containers should run near each other for performance or far apart for availability.

- **Health monitoring.** Watches for container failures and automatically reschedules them.
- **Failover.** Keeps track of what's running on each machine and reschedules containers from failed machines to healthy nodes.
- **Scaling**. Manually or automatically adds or removes container instances to match demand.
- **Networking.** Provides an overlay network that coordinates containers to communicate across multiple host machines.
- **Service discovery.** Enables containers to locate each other automatically even as they move between host machines and change IP addresses.
- **Coordinated application upgrades.** Manages container upgrades to avoid application downtime and enables rollback if something goes wrong.

Several container-orchestration tools are available depending upon the specific needs of the architecture you're managing, including:

- **Kubernetes.** Considered the main standard for container orchestration, Kubernetes is an open-source platform used for deploying and managing containers at scale. It's important to note that Kubernetes is often abbreviated to "K8s" with the "8" representing the eight characters between the K and the s of the word "Kubernetes".
- **Docker Swarm.** Docker's own fully integrated container orchestration tool. Considered less extensible and complex than Kubernetes, it's a good choice for Docker-specific enthusiasts. Docker bundles both Swarm and Kubernetes with the Docker Desktop.
- **Apache Mesos.** Open-source software that can provide management of a container cluster. Requires additional add-on frameworks to support full orchestration tasks.

Kubernetes is typically the clear standard for container orchestration. Most cloud providers offer Kubernetes-as-a-service to help manage the deployment and management of containerized applications for you. For example, Azure Kubernetes Service (AKS) integrates with Azure Container Registry (ACR). It provides its own provisioning portal where you can help secure your container clusters with Azure's Active Directory and deploy apps across Azure's datacenter offerings. Using AKS, you can leverage Azure's enterprise-grade features while maintaining application portability through Kubernetes and the Docker image format.

SAMPLE

Overview of Kubernetes on Windows

Kubernetes is based upon common cluster technology in which a set of resources works together and is managed as a single system that performs similar tasks.

A cluster consists of centralized software, also known as the *master* or *control plane*, that's responsible for scheduling and controlling tasks within the cluster. The components that run the cluster's tasks are *nodes,* and a cluster typically contains multiple nodes that the control plane manages.

A Kubernetes cluster contains a Master/Control plane and a combination of either Windows or Linux-based worker node instances, as follows:

- **At least one Master/Control plane.** Runs several services used to manage the orchestration in Kubernetes. Currently, only the Linux OS is supported as the host OS for the Kubernetes

master. Components that make up the master include kube-api-server, controller, and scheduler services.

- **One or more Linux-based node instances.** Worker nodes based on the Linux OS. Includes components called the Kubelet, Kube-proxy, and the Container runtime services.
- **One or more Windows-based node instances.** Worker nodes based on Windows Server 2022. Includes components called the Kubelet, Kube-proxy, and the Container runtime services.

A Kubernetes workload typically includes several Docker-based containers that are often dispersed throughout multiple worker nodes within the cluster. A Kubernetes object called a *pod* groups one or more containers to represent a single application instance.

> **Note:** A single pod can hold one or more containers. However, a pod usually doesn't contain multiple versions of the same application.

A pod includes information about the shared storage and network configuration, and a specification on how to run its packaged containers. You use pod templates to define the information about the pods that run in your cluster.

> **Learn more:** For more information about Kubernetes for Windows, refer to Kubernetes on Windows.

SAMPLE

Deploy Kubernetes resources

As of Kubernetes version 1.14, Windows-based nodes are supported for both worker nodes and scheduling Windows containers. Windows-based worker nodes must be on Windows Server 2019 or newer and Windows Server version 1809 or newer.

The process for creating a Kubernetes orchestration solution includes the following general steps:

1. **Create a Kubernetes master.** The Kubernetes master is configured using the Linux OS, using the **kubeadm** tool, which initializes the master and provides all the associated tools used to administer cluster nodes. You must install Docker to allow for container support.
2. **Configure a network solution.** The network solution typically used to create routable cluster subnets is a Linux-based Container Network Interface (CNI) plugin called Flannel. Other solutions include configuring a smart top-of-rack (ToR) switch or using Open vSwitch (OvS) or Open Virtual Network (OVN) technology.
3. **Join worker nodes.** After creating the Kubernetes master and configuring the network solution, you can join Windows Server and Linux-based worker nodes to the cluster. Joining the worker nodes consists of configuring OS-specific binaries and connecting to the network solution.
4. **Manage Kubernetes resources.** You use the `Kubectl` command to deploy and manage Kubernetes pods containing the containers which comprise the application.

> **Tip:** Cloud services such as Azure Kubernetes Service (AKS) reduce many challenges of manually configuring Kubernetes clusters by providing a hosted Kubernetes environment.

> These services also simplify the deployment and management of containerized applications in Azure. With AKS, you gain the benefits of open-source Kubernetes without the complexity or operational overhead of running your own custom Kubernetes cluster.

Lab 5: Implement and configure virtualization in Windows Server

Please refer to the online lab to supplement your learning experience with exercises.

Knowledge check

Check your knowledge by answering these questions:

1. Which of the following are requirements for installing the Hyper-V server role? Choose two.
   a. A 32-bit processor
   b. Minimum 32 GB of memory
   c. A 64-bit processor
   d. BitLocker enabled
   e. Intel VT or AMD-V enabled
2. You plan to enable nested virtualization on a Hyper-V host. What do you need to do to ensure that the nested VM can route to external destinations?
   a. Enable BitLocker
   b. Enable MAC address spoofing
   c. Enable Device Guard
   d. Configure a switch with the External Network type
   e. Configure a switch with the Private Network type.
3. Which of the following are true about implementing a Host Guardian service? Choose two.
   a. A new Active Directory forest is created dedicated to the Host Guardian service.
   b. The Host Guardian service must be installed on a server containing the Linux OS.
   c. The Host Guardian service must be installed in a VM.
   d. The Host Guardian service uses certificates for signing and encryption tasks.
   e. The Host Guardian service must be installed in the same domain as the Hyper-V guarded hosts.
4. Which of the following are requirements for creating a shielded template disk? Choose two.
   a. A Generation 2 VM
   b. A basic disk
   c. A Generation 1 VM
   d. A dynamic disk

SAMPLE

e. Must be generalized

5. You download a container base image. When you attempt to create and run a container using the base image, you get an error message that relates to incompatibility with the host machine. What should you do?
   a. Download a new container base image that matches the version of the OS installed on the host machine.
   b. Run the container using the --isolation=process switch.
   c. Update the version of Docker installed on the host machine.
   d. Install a self-signed authentication certificate on the host machine.
   e. Use BitLocker to encrypt the OS drive of the host machine.
6. Which of the following can be used as worker nodes in a Kubernetes cluster? Choose two.
   a. Nano Server
   b. Windows Server 2019 or newer
   c. MacOS
   d. Linux

> **Note:** To find the answers, refer to the **Knowledge check** slides in the PowerPoint presentation.

Learn more

For more information, refer to:

- Hyper-V on Windows Server
- Run Hyper-V in a Virtual Machine with Nested Virtualization
- Guarded fabric and shielded VMs overview
- Windows and containers
- Kubernetes on Windows

SAMPLE