Introduction

Virtualization is a technique that allows multiple virtual machine instances to run independently on the same hardware. In a cloud environment, this is useful to help maximize resource usage. A server instance often has periods of high and low usage. By combining the ability to host multiple instances on a single physical host with automatic deployment of instances, it is possible to use a host's physical resources more efficiently.

Virtual machine instances are managed by a hypervisor. There are two main types of hypervisors. A *type 1* hypervisor runs directly on the physical machine. It manages the use of the physical resources among the virtual machine instances. A *type 2* hypervisor runs as a program on top of a separate operating system. Hyper-V is a virtualization platform from Microsoft that always runs as a type 1 hypervisor. There is a standalone Hyper-V server that is intended to only host virtual machine instances. It is also possible to run Hyper-V on machines with Windows Server or Professional editions. In these cases, Hyper-V runs on the bare metal of the machine and the Windows operating system runs as a special privileged virtual machine with direct access to the hardware.

Storage for virtual machines can be set up in many ways. The hypervisor can manage virtual drives as files on the local hardware. These drives can be a fixed size or dynamically expanding. It is also possible to configure virtual machines to directly use storage attached to the host machine. In larger deployments, storage is sometimes hosted in separate storage pools. In this configuration, there are usually high-speed links between the machines hosting the instances and the machines hosting the storage. In this lab, we will use the first option with virtual hard drives stored as files on the host computer.

Virtual machines can also be connected to networks. Each virtual machine can be configured to have zero or more network adapters. The hypervisor is used to manage connections for those adapters. Each virtualization system has its own method of configuring virtual networks. For Hyper-V, each VM adapter can be connected to a virtual switch. In this lab, you will explore some of the options for these switches. Connecting the VM adapters to the virtual network is like connecting real hardware with switches and cables: it is only part of the configuration. Each VM also needs to have its network interfaces configured as if they were real machines.

Lab Overview

This lab has three parts, which should be completed in the order specified.

1. In the first part of the lab, you will learn how to manage a Hyper-V server using Hyper-V Manager.

2. In the second part of the lab, you will learn how to manage a Hyper-V server using PowerShell.

3. In the third part of the lab, you will use both techniques to change networking configurations.

Finally, you will explore the virtual environment on your own. You will answer questions and complete challenges that allow you to use the skills you learned in the lab to conduct independent, unguided work - similar to what you will encounter in a real-world situation.

Learning Objectives

Upon completing this lab, you will be able to:

1. Describe what a Hypervisor does.

2. Use Hyper-V Manager to manage a Hyper-V Server.

3. Use PowerShell to manage a Hyper-V Server.

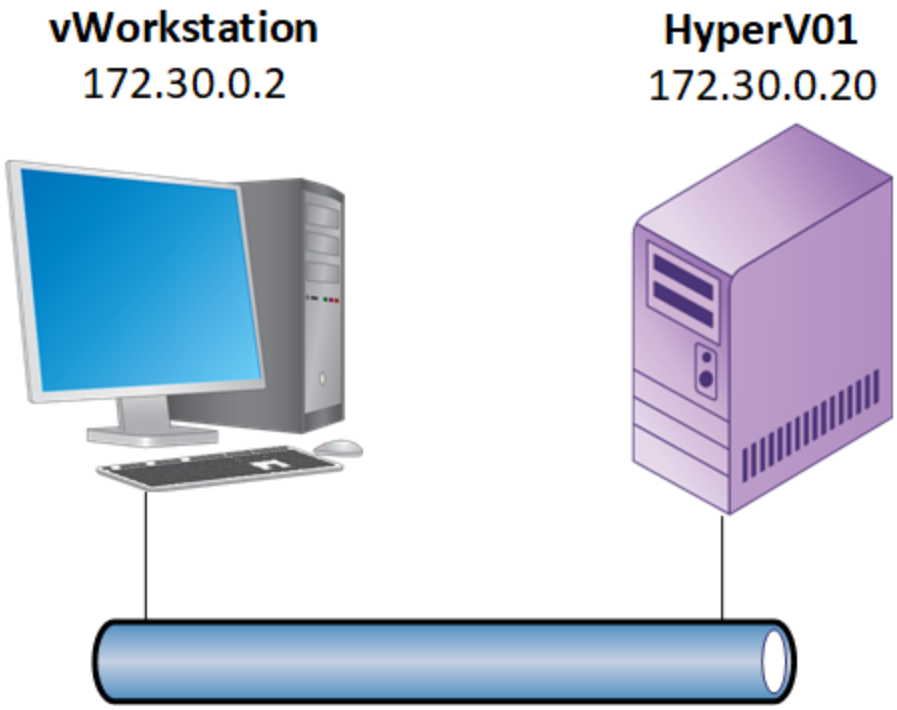
4. Describe virtual switches used by Hyper-V servers.

5. Add a virtual switch and connect it to existing VMs.

Topology

This lab contains the following virtual machines. Please refer to the network topology diagram below.

- vWorkstation
- HyperV01
    - Ubuntu 20
    - Windows 2019



Tools and Software

The following software and/or utilities are required to complete this lab. Students are encouraged to explore the Internet to learn more about the products and tools used in this lab.

- Hyper-V Manager
- PowerShell

Deliverables

Upon completion of this lab, you are required to provide the following deliverables to your instructor:

Hands-On Demonstration

1. Lab Report file, including screen captures of the following:

- Network interface configuration on the Ubu20 virtual machine.
- Virtual machine list with the new name and memory allocation.
- New memory allocation.
- Get-VM output indicating the machine is using two processors.
- The successful pings.
- VM list with the new name.
- Successful ping from the Windows VM to the Linux VM.
- Successful ping from the Linux VM to the Windows VM.

2. Any additional information as directed by the lab:

- None.

Challenge and Analysis

1. Lab Report file, including screen captures of the following:

- state of all the MicroService VM's Integration Services in the cmdlet output.
- the snapshots for each VM.

Hands-On Demonstration

Part 1: Basic Operations using Hyper-V Manager

Note: When a standalone Hyper-V server is first set up, there are no virtual machines. Each virtual machine instance must be created. Creating a virtual machine involves configuring the virtual hardware, storage, and networking; starting the machine; and installing an operating system.

You are a part of a development team setting up the infrastructure for a new project. The project involves multiple microservices running on different servers. The microservices are being developed to run on Windows Server 2019. There is also an end user application that runs on Linux. Your team needs to set up a couple of virtual machines so that the software can be deployed and tested on them. One member of your team has installed Hyper-V on the development server and created two virtual machine instances: one with Windows Server 2019 and one with Ubuntu Desktop 20.04.

You are going to start by reviewing the machines that have been set up; then you will do some basic configuration based on the anticipated CPU and memory loads. Hyper-V Manager is a GUI application on Windows that can be used to interact with a Hyper-V server and create, configure, or connect to virtual instances on that server.

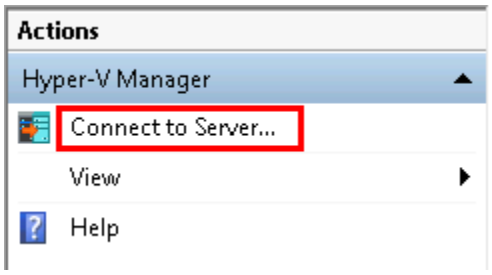
1. In the vWorkstation taskbar, **click** the **Hyper-V Manager icon** to open the Hyper-V Manager.

Hyper-V Manager Icon

Note: The Hyper-V manager allows you to connect to a Hyper-V server and manage or interact with the virtual machines on that server. The left pane will show a list of connected servers. The large central pane will show details about the currently selected server. The right pane contains a set of actions that can be used to manage servers.

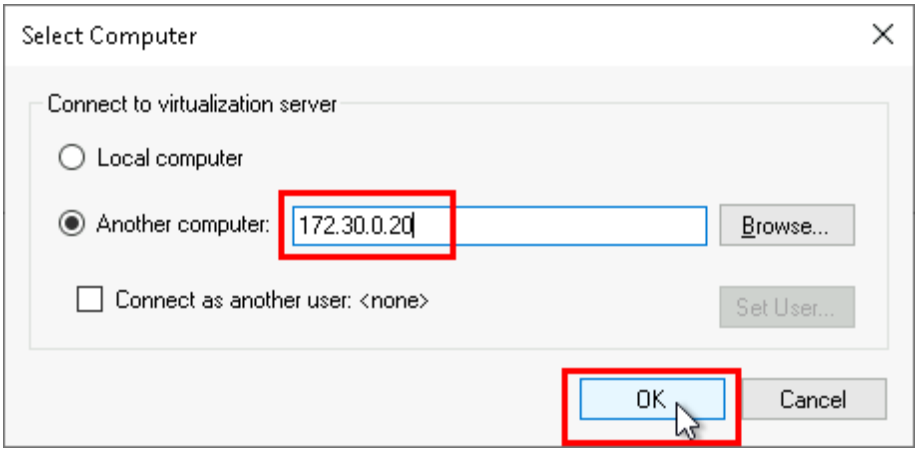
Your development team has already set up the Hyper-V server for testing. It is available at 172.30.0.20. You need to connect the Hyper-V manager to the server.

2. In the Actions pane, **click Connect to server…** to open the Select Computer dialog box.



Connect to Server Action

3. In the Select Computer dialog box, **type** `172.30.0.20` in the Another Computer: radio text field, and then **click** the **OK button**.



Select Computer Dialog

Note: You should see a server labeled 172.30.0.20 under the Hyper-V Manager server node in the left pane. The center pane should show three widgets:

- Virtual Machines widget
- Checkpoints widget

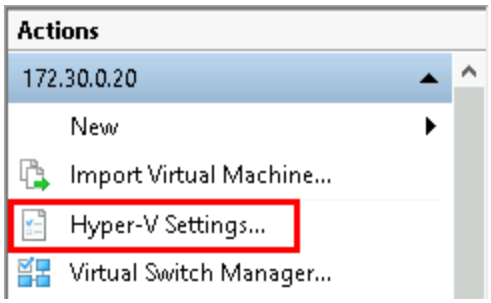
- Ubu20 widget

The Virtual Machines widget should display a list of virtual machines with two entries: Ubu20 and Win2019. These represent the Linux (Ubu20) and Windows (Win2019) virtual machine instances that your team has already created. By default, the top machine listed in the Virtual Machine widget is selected and populates the information in the other two widgets. The Checkpoints widget displays a list of any saved states on the selected virtual machine. The bottom widget displays details about the selected virtual machine's configuration. Take a moment to select the tabs labeled *Summary*, *Memory*, *Networking*, and *Replication* to view the Ubu20 virtual machine's configuration.

You should notice that the Actions pane has changed. It now contains two sections. The top section labeled 172.30.0.20 contains a list of possible actions that you can perform on the Hyper-V server. The bottom section labeled Ubu20 contains actions that you can perform on the selected virtual machine instance.

In the next steps, you will examine the settings on the Hyper-V server.

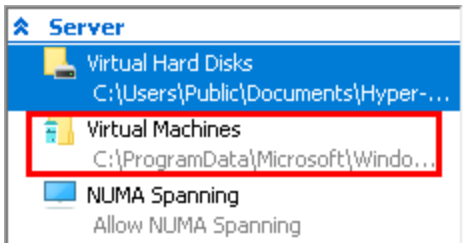
4. In the 172.30.0.20 section of the Actions panel, **click Hyper-V Settings…** to open the Hyper-V Settings for 172.30.0.20 dialog box.



Hyper-V Settings Action

Note: The left pane of the Hyper-V Settings dialog box contains a Server section and a User section. Each section contains a list of settings groups. The right panel displays information related to the selected settings group. You should see that the Virtual Hard Disks settings group is selected and that the right panel displays the physical location of the virtual hard disks on the Hyper-V server as *C:\Users\Public\Documents\Hyper-V\Virtual Hard Disks*. This is a path on the Hyper-V server, not vWorkstation.

5. In the Server panel, **click** the **Virtual Machines** settings group to reveal the physical storage location of the virtual machine configuration files.



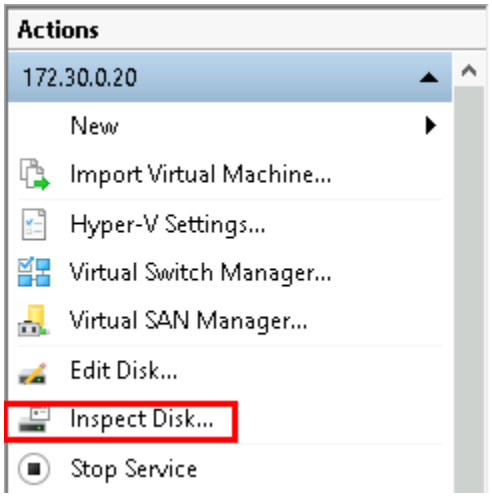
Virtual Machines Settings Group

Note: You should see that the path to the Virtual Machines configuration files is *C:\ProgramData\Microsoft\Windows\Hyper-V*. This is also a path on the Hyper-V server, not vWorkstation. This path specifies where the files required by the hypervisor to manage the virtual machines are stored. The other settings groups are outside the scope of this lab, but you should take a moment to examine the information that they reveal.

6. Click the **Cancel button** to close the Hyper-V Settings for 172.30.0.20 dialog box.

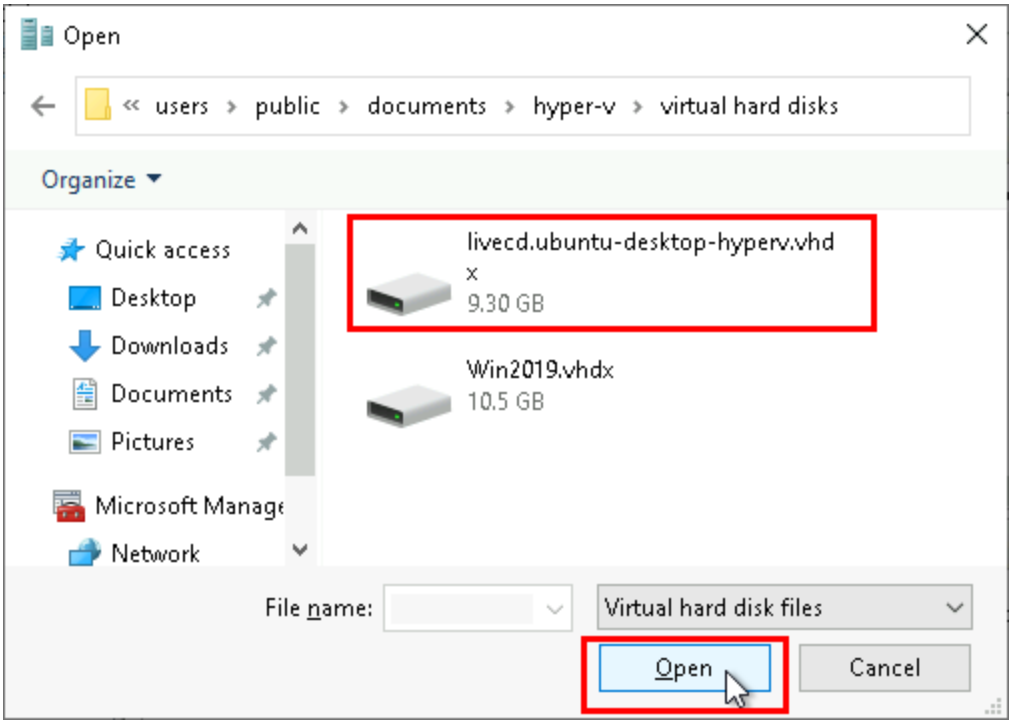
Note: Your team has configured the Hyper-V server to store the virtual hard disks as files. In a production environment with many virtual machines, the storage could be located on separate machines. In the next steps, you will examine the properties of the virtual hard disks on each virtual machine.

7. In the 172.30.0.20 section of the Actions panel, **click Inspect disk…** to open the Open virtual hard disk dialog box.



Inspect Disk Action

8. In the Open dialog box, **select livecd.ubuntu-desktop-hyperv.vhdx** and **click** the **Open button** to open the Virtual Hard Disk Properties box.



Open Hard Disk Properties Dialog

9. **Repeat steps 7–8** to examine the Windows VM disk space (*Win2019.vhdx*).

Note: Your team configured these virtual hard disks as dynamically expanding disks. That means that the actual size of the files on the Hyper-V server can initially be smaller than the capacity of the virtual disk when the full size is not needed. This can be helpful when it is unlikely that all the images will be full, but occasionally a few will grow to contain temporary data. Other virtualization systems like VMware and Citrix XenServer typically offer similar options for storage.

A configuration option that will not be used in this lab is to use fixed virtual hard disk sizes. While a fixed size will immediately consume the full size of the virtual disk from the storage available, it also opens up options for more advanced backup mechanisms.

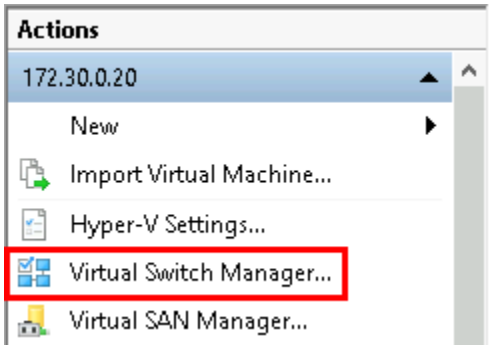
10. **Close** both of the **Virtual Hard Disk Properties dialog boxes**.

Note: The Ubu20 and Win2019 virtual machine instances can also be connected to virtual networks. Hyper-V provides a mechanism to create virtual switches. There are three different options.

- An external switch is connected to a physical network adapter on the host machine, so it allows the virtual machines connected to the switch to communicate with the external network.
- An internal switch allows communication among virtual machines on the server that are connected to the switch. The host can also communicate on this network.
- A private switch only allows communication among the virtual machines on the server that are connected to the switch.

In the next steps, you will use the virtual switch manager to examine the network that your team has set up.

11. In the 172.30.0.20 section of the Actions panel, **click Virtual Switch Manager…** to open the Virtual Switch Manager for 172.30.0.20.



Virtual Switch Manager Action

12. In the Virtual Switches panel, **click Ext** to view the Virtual Switch Properties.

Select Virtual Switch Ext

Note: You should notice that the Name field of the Virtual Switch Properties is *Ext*. The Hyper-V server has been configured with a single external virtual switch called *Ext*. When configuring machine instances, this switch name will be available as a connection point for virtual network adapters on the instance. You should also notice that the Connection type radio button is set to *External network*. This indicates that virtual machines configured to use the Ext virtual switch can access the internet through it.

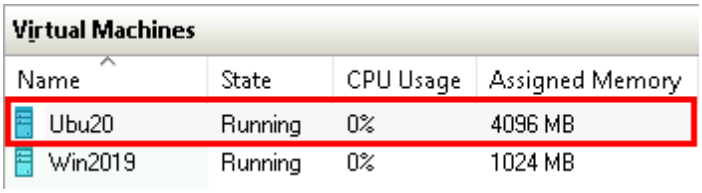
13. Click the **Cancel button** to close the Virtual Switch Manager window without making any changes.

Note: Now turn your attention to the list of virtual machines in the Virtual Machines widget. The state for each machine should be *Running*. Take a moment to examine the information given in the CPU Usage, Assigned Memory, and Uptime columns.

In the next steps, you will use the Ubu20 widget to examine the network connections on the Linux

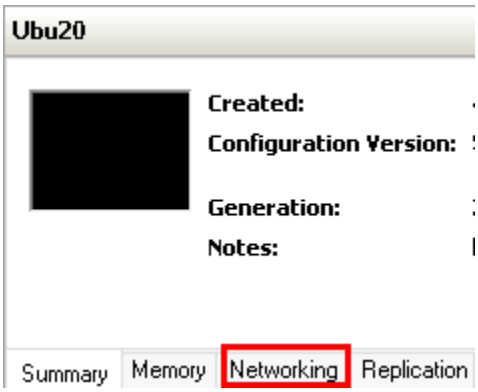
virtual machine.

14. In the Virtual Machines widget, **click** the **Ubu20 row** to select that instance.



Select Ubu20 Virtual Machine

15. In the Ubu20 widget, **click** the **Networking tab** to view the networking summary.



Virtual Machine Networking Tab

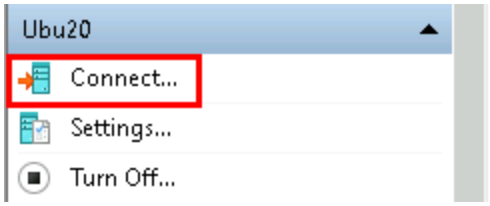
Note: You should notice that the Ubu20 virtual machine has a single network interface, which is connected to the *Ext* virtual switch. This indicates that the Ubu20 virtual machine will be able to access the external network.

16. **Repeat steps 14–15** for the Win2019 virtual machine to view its networking summary.

Note: You should notice that the Win2019 virtual machine also has internet access through its connection to the Ext virtual switch. Now that you have examined the physical properties of the virtual machines, you will use the Hyper-V manager to connect to test-run the Linux virtual machine.

17. In the Virtual Machines widget, **click** the **Ubu20 row** to select the Linux virtual machine instance.

18. In the Ubu20 section of the Actions panel, click **Connect…** to open a virtual machine connection in a new window.



Virtual Machine Connect Action

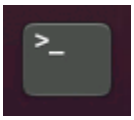
Note: You should see a Virtual Machine Connection window that looks like a normal Ubuntu login screen. You can interact with the virtual machine using the mouse and keyboard as usual.

19. In the Ubu20 on 172.30.0.20 Virtual Machine Connection window, **click** the **user icon** and then **type** `password` in the Password field and **press Enter** to log in to the Ubu20 VM.

Note: It may take a minute or two for the Linux Desktop to appear. In the context of this lab, the virtual machine loading process and interactions may seem slow because of the layers of virtual machines that are being used. You are accessing the vWorkstation virtual machine through your web browser. In turn, the vWorkstation virtual machine is running a Linux virtual machine. This complex setup with one virtual machine containing another virtual machine is called *nested virtualization*.

In the next steps, you will examine the network from the perspective of the Linux virtual machine.

20. In the Linux VM Taskbar, **click** the **Terminal icon** to open a terminal window.



Terminal Icon

21. At the command prompt, execute **ip a** to view the network interface configuration.

Note: You should see two network interfaces. The first, named “lo,” is the loopback interface. This interface allows processes executing on the Ubu20 virtual machine to communicate using network protocols to other processes on the same machine. The second, named “eth0,” is the connection to the Ext virtual switch. The interface has been assigned an IPv4 address, 172.30.0.102, as well as an IPv6 address.

22. **Make a screen capture** showing the network interface configuration on the Ubu20 virtual machine.

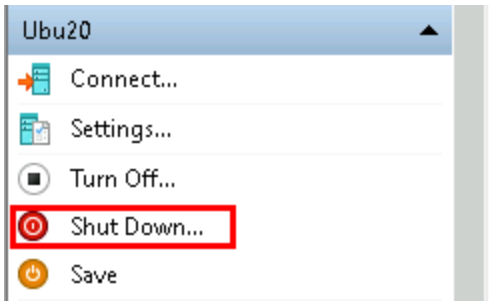
23. At the command prompt, **execute** the **exit command** to close the Terminal session.

24. In the Ubu20 on 172.30.0.20 Virtual Machine Connection, **select File > Exit** to close the Linux Virtual Machine Connection window.

25. **Restore** the **Hyper-V Manager window**.

Note: Each VM consumes resources from the host Hyper-V server. Tuning the VMs to use only the resources they need can help your servers run more efficiently. The two primary resources that are consumed are the processors and memory. Your development team has noticed that the integration testing on the Linux machine is both processor- and memory-intensive. You have been tasked with increasing the memory and processor usage through the Hyper-V Manager. Changing these settings while the VM is running can be problematic, so first you will shut down the instance.

26. In the Ubu20 section of the Actions pane, **click Shut Down...** to shut down the Ubu20 virtual machine.

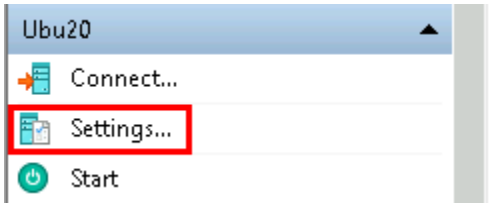


Virtual Machine Shut Down Action

Note: You should notice the State change to *Off* in the Ubu20 row of the Virtual Machines widget.

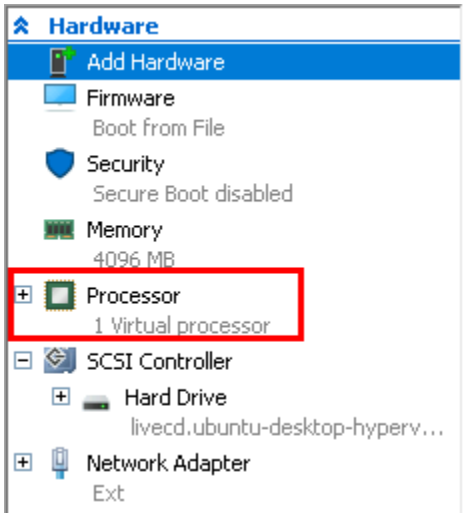
In the next steps, you will increase the processor and memory usage of the Ubu20 instance and then restart the instance.

27. In the Ubu20 section of the Actions pane, **click Settings...** to open the Settings for Ubu20 on 172.30.0.20 window.



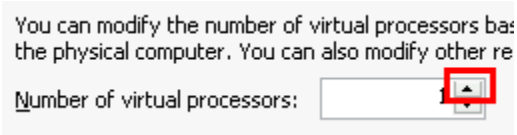
Virtual Machine Settings Action

28. In the Hardware pane of the Settings window, **click Processor** to view the Processor configuration options.



Select the Processor configuration options

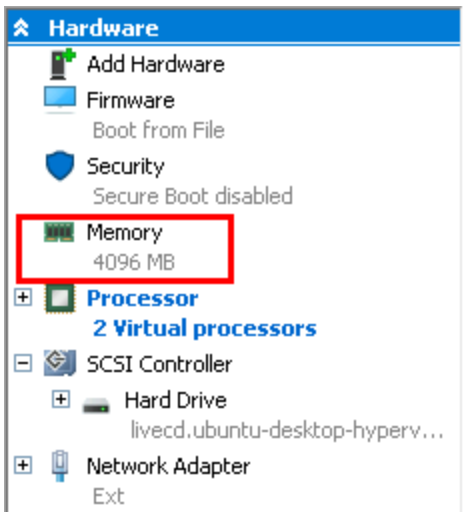
29. In the Number of virtual processors: field, **click** the **up arrow** to change the number of processors from 1 to 2.



Increase the Processor count

30. In the Hardware pane of the Settings window, **click Memory** and then **change RAM** from

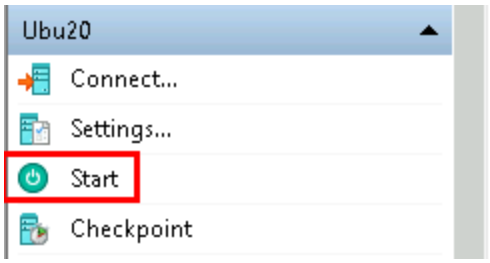
4096 MB to **5120 MB**.



Select the Memory configuration options

31. In the Settings for Ubu20 on 172.30.0.20 window, **click** the **OK button** to save the new settings.

32. In the Ubu20 part of the actions panel, **click Start** to restart the Linux VM.



Virtual Machine Start Action

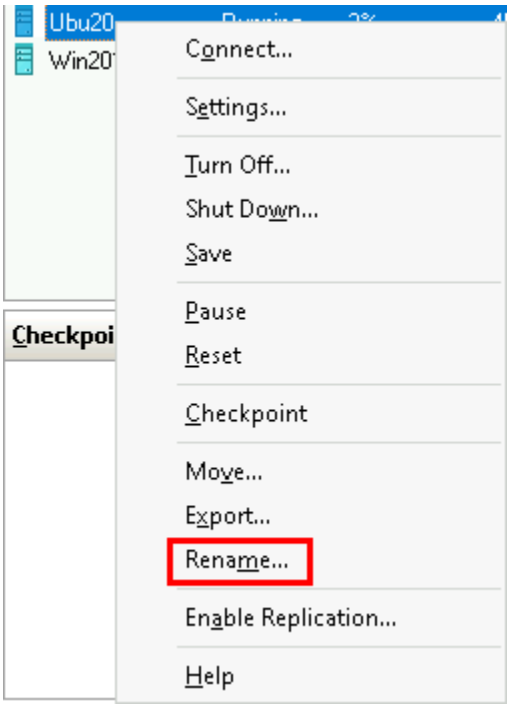
Note: You should see an error dialog box that indicates that the Ubu20 virtual machine failed to start because there was not enough memory in the system to start it. Although your team had hoped to increase the memory to 5120 MB, you will adjust the configuration to a slightly smaller capacity and then inform the team.

33. In the Ubu20 section of the Actions pane, **click Settings...** to open the Settings window.

34. **Repeat Steps 30–32** to **change the RAM to 4500 MB** and then **restart** the **Linux VM**.

Note: The Ubu20 row in the Virtual Machines widget will reflect your changes. Your next task is to rename the virtual machines to reflect their purpose in the production application. Meaningful names are almost always a good idea. Since the Linux machine is intended to run the user application for integration testing, your team has determined that its name will be *UserApp*.

35. In the Virtual Machines list, **right-click Ubu20 in the Name column** and then **select Rename...** from the context window.



Select Rename from the context menu

36. In the active Name field, **replace Ubu20** with **UserApp** and **press Enter** to rename the VM.

37. **Make a screen capture** showing the virtual machine list with the new name and memory allocation.

38. In the Hyper-V Manager, **select File > Exit** to close the Hyper-V Manager application.

Part 2: Basic Operations Using PowerShell

Note: In Part 1 of the lab, you examined and modified the configuration of virtual machines (VMs) through the graphical user interface of the Hyper-V Manager. For exploration of a new system, using a graphical interface can be helpful. However, for deploying systems using automation tools, it is often helpful to be able to use a command line interface.

The Hyper-V module available for Windows PowerShell allows you to manage Hyper-V servers and instances from the command line. This module has already been installed on vWorkstation. The module's standard installation includes many cmdlets (pronounced *command-lets*), which are commands that can be used at the PowerShell prompt.

Over the course of the next steps, you will use a command line interface to examine similar information about the Hyper-V server and its virtual machines that you accessed in Part 1 of the lab. The first command that you will use is *Get-VM.* This is the name of a PowerShell cmdlet that lists the virtual machines on a Hyper-V server. The -ComputerName option to the *Get-VM* cmdlet takes an argument that provides the name (or address) of the Hyper-V server.

1. On the vWorkstation Taskbar, **click** the **PowerShell icon** to open a PowerShell.

PowerShell Icon

2. At the command prompt, **execute** `Get-VM -ComputerName 172.30.0.20` to display a list of VMs running on the Hyper-V server.

```
PS C:\Users\Administrator> Get-VM -ComputerName 172.30.0.20

Name    State   CPUUsage(%) MemoryAssigned(M) Uptime
----    -----   ----------- ----------------- ------
UserApp Running 0           4500              00:04:47.1...
Win2019 Running 0           1486              00:00:00
```

Execute Get-VM command

Note: You should see a list of the two VMs. Now the Linux instance is named UserApp because you changed it through the Hyper-V Manager. You can also use PowerShell to make changes to the settings. The microservice that is being developed to run on Windows Server 2019 requires very little memory, so your team has decided to reduce the Win2019 VM memory requirements. You are curious about the limitations of the virtual machines and decide to try to increase the memory beyond what the server can handle to see what happens. To change memory settings, you will use the Set-VM cmdlet. The required arguments are as follows:

- -ComputerName 172.30.0.20
  -- This option directs Set-VM to use the Hyper-V server at IP address 172.30.0.20.
- -VMName Win2019
  -- This option and its argument set the instance on the Hyper-V server.

- -MemoryStartupBytes 2147483648
  -- This option and its argument instruct Set-VM to use 2 GB when the Win2019 VM starts up.

Note, that for any PowerShell cmdlets that query a remote Hyper-V server, the -ComputerName option is required; otherwise, PowerShell assumes the host on which you are executing the command is the target Hyper-V server.

The MemoryStartupBytes option requires that its argument (the number of bytes) be a power of 2. If you try to use a “nicer” number like 2,000,000,000 for the memory, the Hyper-V Manager will return an error indicating that the size is not aligned correctly. The value that you are supplying in your command is the result of the calculation 2*1024*1024*1024 (2 GB).

3. At the command prompt, **execute** **`Set-VM -ComputerName 172.30.0.20 -VMName Win2019 -MemoryStartupBytes 2147483648`**.

```
PS C:\Users\Administrator> Set-VM -ComputerName 172.30.0.20
-VMName Win2019 -MemoryStartupBytes 2147483648
Set-VM : Failed to modify device 'Memory'.
'Win2019' failed to modify device 'Memory'. (Virtual
machine ID 2F1899C4-F905-42A5-94A8-D1557311B4E6)
Changing memory size of running virtual machine 'Win2019'
failed. The system doesn't have enough memory to complete
the operation.
At line:1 char:1
+ Set-VM -ComputerName 172.30.0.20 -VMName Win2019
-MemoryStartupBytes  ...
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~
    + CategoryInfo          : FromStdErr: (:) [Set-VM], Vi
   rtualizationException
    + FullyQualifiedErrorId : OutOfMemory,Microsoft.HyperV
   .PowerShell.Commands.SetVM
```

Execute Set-VM command

Note: The command should fail with an error that Set-VM failed to modify the device “Memory.” Run the command again with a reduced memory footprint of 768 MB = 7*1024*1024 = 805306368 bytes.

4. At the command prompt, **execute** **`Set-VM -ComputerName 172.30.0.20 -VMName Win2019 -MemoryStartupBytes 805306368`** to allocate 768 MB to the Windows VM.

```
PS C:\Users\Administrator> Set-VM -ComputerName 172.30.0.20
-VMName Win2019 -MemoryStartupBytes 805306368
```

Execute Set-VM command

Note: You can access and modify previously run commands in the PowerShell by using the Up and Down arrows on your keyboard.

5. **Repeat Step 2** to confirm that the MemoryAssigned(M) field now displays 768 MB.

6. **Make a screen capture** showing the new memory allocation.

Note: Your development team expects the service being developed for the Windows server to be processor intensive. Your next task is to upgrade the Win2019 VM to two CPUs. You will use the Set-VM cmdlet again with the -ProcessorCount flag to change the number of processors.

7. At the command prompt, **execute** `Set-VM -ComputerName 172.30.0.20 -VMName Win2019 -ProcessorCount 2` to upgrade the Windows VM to two processors.

```
PS C:\Users\Administrator> Set-VM -ComputerName 172.30.0.20
-VMName Win2019 -ProcessorCount 2
Set-VM : Failed to modify device 'Processor'.
Cannot change the number of processors now.
'Win2019' failed to modify device 'Processor'. (Virtual
machine ID 2F1899C4-F905-42A5-94A8-D1557311B4E6)
Cannot change the number of processors of virtual machine
'Win2019' while it is running. (Virtual machine ID
2F1899C4-F905-42A5-94A8-D1557311B4E6)
At line:1 char:1
+ Set-VM -ComputerName 172.30.0.20 -VMName Win2019
-ProcessorCount 2
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~
    + CategoryInfo          : InvalidOperation: (:) [Set-V
   M], VirtualizationException
    + FullyQualifiedErrorId : InvalidState,Microsoft.Hyper
   V.PowerShell.Commands.SetVM
```

Execute Set-VM command

Note: As you might expect, changing the number of processors of a virtual machine while it is running is not supported. You need to shut down the virtual machine in order to change the number of processors. You will use the Stop-VM cmdlet to stop a virtual machine instance. It uses the familiar options -ComputerName and -VMName.

8. At the command prompt, **execute** **`Stop-VM -ComputerName 172.30.0.20 -VMName Win2019 -TurnOff`** to shut down the Windows VM.

```
PS C:\Users\Administrator> Stop-VM -ComputerName 172.30.0.20
 -VMName Win2019 -TurnOff
```

Execute Stop-VM command

Note: The default behavior of Stop-VM is to shut down the machine via the guest operating system, often referred to as a soft shutdown. The -TurnOff flag specifies a hard shutdown, which is equivalent to disconnecting the machine from its power source. A soft shutdown is preferred whenever possible to avoid corruption and data loss, though it requires that the guest operating system have the corresponding Hyper-V integration service installed. These services allow the guest operating system to communicate with the Hyper-V host, including via shutdown signals. In this lab, you are using -TurnOff for brevity.

9. **Repeat Step 2** to confirm that the State of the Win2019 VM is *Off*.

10. **Repeat Step 7** to upgrade the Windows VM to two processors.

Note: A successful processor change will show no output messages and simply return the command prompt. Now you can restart the VM using the Start-VM cmdlet. This will use the familiar parameter -ComputerName, but the virtual machine is selected using the -Name option.

11. At the command prompt, **execute** **`Start-VM -ComputerName 172.30.0.20 -Name Win2019`** to restart the Win2019 VM.

```
PS C:\Users\Administrator> Start-VM -ComputerName 172.30.0.2
0 -Name Win2019
```

Execute Start-VM command

12. **Repeat Step 2** to confirm that the State of the Win2019 VM is *Running*.

Note: You should notice that the Get-VM command does not display the number of processors that a virtual machine uses.

In the next step, you will use the Get-VMProcessor cmdlet to confirm that the Win2019 virtual machine is now running with two CPUs.

13. At the command prompt, **execute** `Get-VMProcessor -ComputerName 172.30.0.20 -VMName Win2019` to show the processor information.

Execute Get-VMProcessor command

14. **Make a screen capture** showing the Get-VM output indicating the machine is using two processors.

Note: Now that you have adjusted the Win2019 VM to the settings that your team expects will be needed during the development, you will use the command line interface to connect to VM instances. The Hyper-V PowerShell module does not have a cmdlet for this. Instead, you will use the vmconnect.exe application. This is the same application that the Hyper-V Manager application uses to connect with virtual machines.

15. At the command prompt, **execute** `vmconnect.exe 172.30.0.20 Win2019`.

```
PS C:\Users\Administrator> vmconnect 172.30.0.20 Win2019
```

Execute vmconnect command

16. In the Win2019 VM menubar, **click** the **CTRL+ALT+DEL icon** to open the Windows login screen.

Control Alt Delete Icon

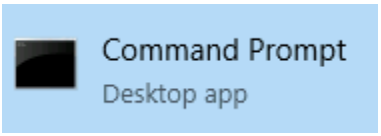
17. In the Administrator's Password field, **type** `P@ssw0rd!` and **press Enter** to log in to the Win2019 VM as the Administrator user.

Note: In the next steps, you will verify that the virtual machine is connected to the external network.

18. In the Win2019 VM Taskbar, **click the Magnifying glass** to open a Search box.

Search Box Icon

19. In the Search box, **type** `cmd` and then **select** the **Command Prompt** Desktop app from the Best match pane to open a Command Prompt window.



Command Prompt Search Result

20. At the Win2019 VM command prompt, **execute** `ipconfig /all` to display all TCP/IP connections.

Note: In the lists, you should see an ethernet adapter that has been assigned an IPv4 address 172.30.0.101 and also an IPv6 address.

In the next steps, you will verify connectivity using simple ping commands.

21. At the Win2019 VM command prompt, **execute** `ping 172.30.0.102` to verify connectivity to the Linux VM.

22. **Repeat step 21** with address 172.30.0.20 to verify connectivity to the Hyper-V server.

23. **Make a screen capture** showing the successful pings.

Note: You should see ping replies after each ping command. This indicates that the Win2019 VM can send (and receive) network traffic to (and from) both the Linux VM and the Hyper-V server over the virtual switch *Ext*.

24. At the Win2019 VM command prompt, **execute** `exit` to close the Command Prompt window.

25. In the Win2019 VM Connection window, **select File > Exit** to close the window.

Note: Now that you have confirmed that the Win2019 virtual machine runs with the new configuration settings, you will change its name using PowerShell. The Windows Server is going to be used to run a microservice that is being developed, so your team has decided to use *MicroService* as the name for the Win2019 VM.

In the next steps, you will use the Rename-VM cmdlet to change the Win2019 virtual machine's name.

26. At the vWorkstation command prompt, **execute** `Rename-VM -ComputerName 172.30.0.20 -Name Win2019 -NewName MicroService` to rename the Win2019 VM.

```
PS C:\Users\Administrator> Rename-VM -ComputerName 172.30.0.
20 -Name Win2019 -NewName MicroService
```

Execute Rename-VM command

27. **Repeat Step 2** to confirm that the Name field now displays MicroService.

28. **Make a screen capture** showing the VM list with the new name.

29. **Close** the **PowerShell** window.

Part 3: Add a New Private Network

Note: The network configuration required for any machine should be determined by its expected environment and usage. When your development team's application is deployed, the communication between the user application on the UserApp virtual machine and the microservice on the MicroService virtual machine needs to be encrypted and travel over a virtual private network (VPN). However, that VPN is not going to be set up in the development environment. Instead, you will emulate the VPN by creating a new internal virtual network. This network will primarily be used for data transport in the application, so your development team has decided to name the switch *Data*.

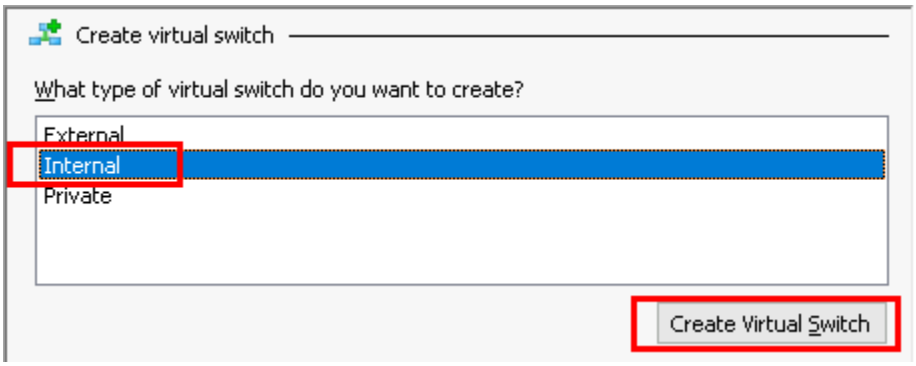
In the next steps, you will use the Hyper-V Manager to create a virtual switch.

1. In the vWorkstation taskbar, **click** the **Hyper-V Manager icon** to open the Hyper-V Manager.

Note: You should notice that the configuration changes that you made through the command line in Part 2 of the lab are reflected in the information displayed through the widgets in the Hyper-V Manager.

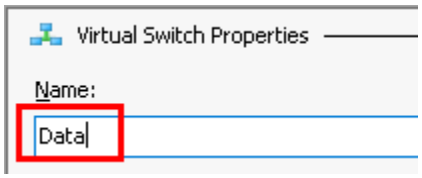
2. In the 172.30.0.20 section of the Actions pane, **click Virtual Switch Manager...** to open the Virtual Switch Manager for 172.30.0.20 window.

3. In the Create virtual switch pane, **select Internal** and then **click** the **Create Virtual Switch button** to open the new Virtual Switch Properties pane.



Create a new internal Virtual Switch

4. In the Name: field, **replace** the text with **Data** and then **click** the **OK button** to save the Internal Virtual Switch configuration.



Set the switch name

Note: You have created a new Internal switch, but the virtual machines are not yet connected to it. Next, you will update each VM to add a network interface. Because of the magic of virtualization, you can do this without shutting down the machines!

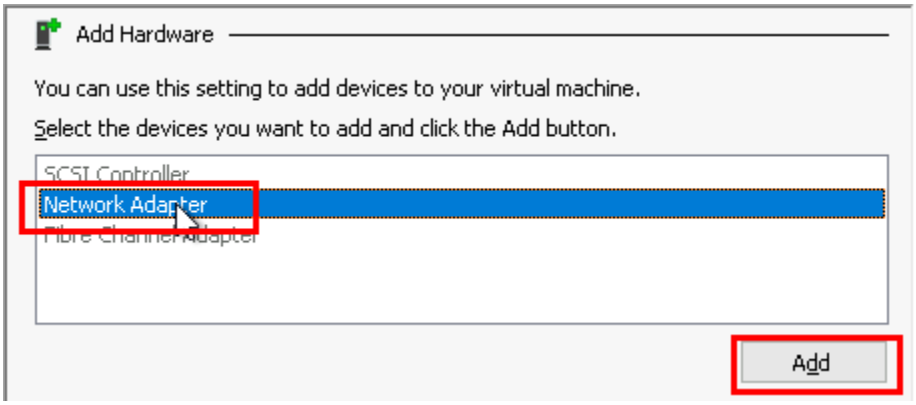
In the next steps, you will add a new interface to the MicroService VM using Hyper-V Manager.

5. In the Virtual Machines widget, **select** the **MicroService VM**.

Note: You should notice that the bottom widget and the bottom section of the Actions pane have changed to reflect information and actions related to the MicroService VM.

6. In the MicroService section of the Actions pane, **select Settings...** to open the Settings for MicroService on 172.30.0.20 dialog box.

7. In the Add Hardware pane, **select Network Adapter** and then click the **Add button** to add a new interface.

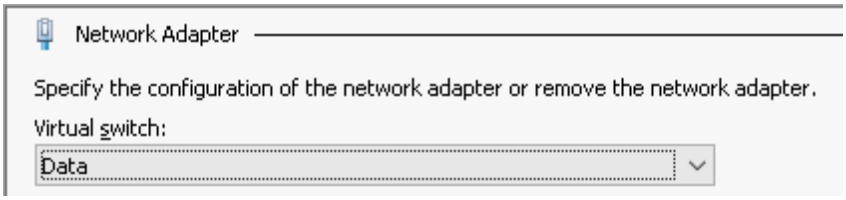


Add a new Network Adapter

Note: You should notice that a new Network Adapter settings group has been added at the bottom of the Hardware pane and that it is automatically selected.

In the next steps, you will indicate which virtual switch the adapter should be connected to.

8. In the Virtual Switch: dropdown, **select** the **Data** virtual switch and then **click** the **Apply button** to save the new hardware configuration.



Choose the Virtual Switch

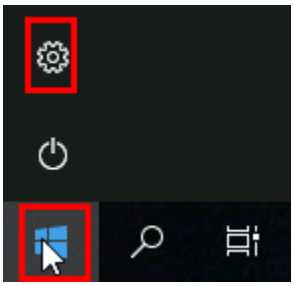
9. In the Settings window, **click** the **OK button** to return to the Hyper-V Manager.

Note: Your next task is to configure this interface. In a production environment, the interface may need to use DHCP to assign its IP address, even within a virtual network. Accessing a DHCP server is outside the scope of this lab. Your team has decided to give each virtual machine a static IP address

on the emulated virtual network. This configuration needs to be done from within the virtual machine.

10. In the MicroService section of the Actions panel, **click Connect...** to open a connection to the MicroService VM.

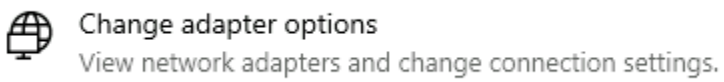
11. In the MicroService Virtual Machine Connection window, **click the start icon** and **select the gear icon** to open the Windows Settings window.



Open Windows Settings

12. In the Settings window, **click Network & Internet** to open the Network & Internet settings dialog box.

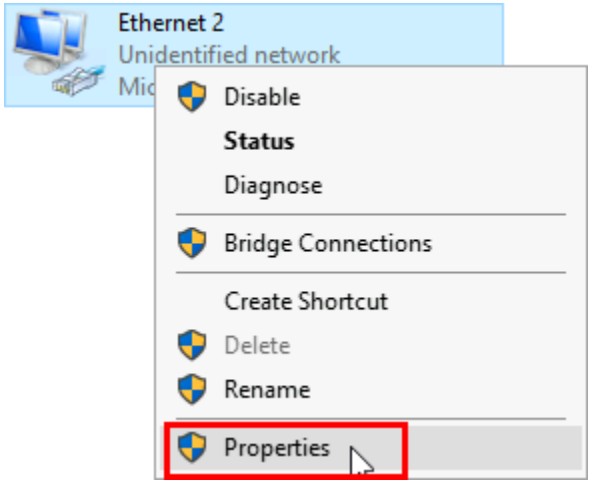
13. On the right side of the Network & Internet settings dialog box, **click Change adapter options** to open the Network Connections window.



Change adapter options heading

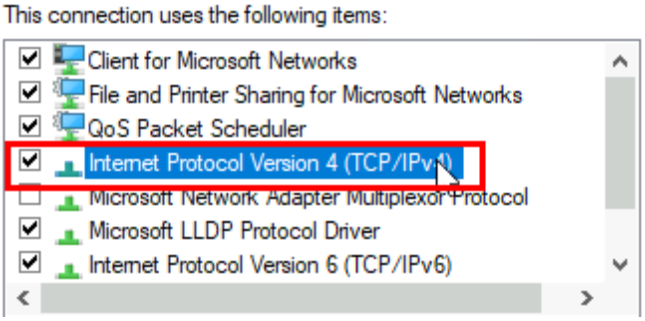
Note: This window can also be opened by executing ncps.cpl from a command prompt. You should see two ethernet adapters. Ethernet 2 is the new adapter that was added. You need to update the TCP/IP settings to assign a manual address.

14. In the Network Connections window, **right-click Ethernet 2** and then **select Properties from the context menu** to open the Ethernet 2 Properties window.



Ethernet 2 Conext Menu

15. In the Networking tab, **double-click Internet Protocol Version 4 (TCP/IPv4)** to open the Internet Protocol Version 4 (TCP/IPv4) Properties dialog box.



TCP/IPv4 Properties Heading

16. In the TCP/IPv4 Properties dialog box, **select** the **Use the following IP address: radio button** and then complete its section with the following information and **click** the **OK button** to save it.

IP address: **`10.10.10.1`**
Subnet mask: **`255.255.255.0`**
Default gateway: (leave blank)

17. In the Ethernet 2 Properties dialog box, **click** the **OK button** to close the dialog box.

18. **Close** the **Ethernet 2 Properties window**.

19. **Close** the **Network connections window**.

20. **Close** the **Windows Settings window**.

Note: Now that you have configured the second network interface in the MicroService virtual machine, you will verify the configuration changes by using a command prompt.

21. In the MicroService VM Taskbar, **click the Magnifying glass** to open a Search box.

22. In the Search box, **type** `cmd` and then **select** the **Command Prompt Desktop app** from the Best match pane to open a command Prompt window.

23. At the command prompt, **execute** `ipconfig` to display the Windows IP Configuration for the Ethernet adapters.

Note: You should see the two ethernet interfaces. The original interface still has the 172.30.0.101 IP address. The new interface should display the IP address 10.10.10.1 and Subnet Mask 255.255.255.0.

24. **Minimize** the **MicroService on 172.30.0.20 VM Connection window**.

Note: Your next task is to add an interface to the UserApp VM. This time, you will use PowerShell on vWorkstation to access the Hyper-V Manager command line interface. The Add-VMNetworkAdapter cmdlet allows you to add an adapter. It takes the familiar arguments of ComputerName and VMName to specify the Hyper-V server and instance. You will use the argument - *SwitchName Data* to specify which virtual switch to connect to.

25. At the vWorkstation PowerShell prompt, **execute** `Add-VMNetworkAdapter -ComputerName 172.30.0.20 -VMName UserApp -SwitchName Data`.

```
PS C:\Users\Administrator> Add-VMNetworkAdapter -ComputerNa
me 172.30.0.20 -VMName UserApp -SwitchName Data
```

Execute Add-VMNetworkAdapter command

Note: The Add-VMNetworkAdapter command should complete without any output. As with the Windows VM, the operating system on the VM needs to be used to configure the settings of the network adapter.

In the next steps, you will connect to the UserApp VM to configure its network settings, using Hyper-V's provided connection utility.

26. At the vWorkstation command prompt, **execute** `vmconnect.exe 172.30.0.20 UserApp`.

If prompted, log in with the following credentials:

Username: `user`
Password: `password`

Note: You may notice a popup window with the message “Connection failed - Activation of network connection failed” before the following steps are completed. The Ubuntu operating system has a network manager running that detects the presence of the new network adapter. However, since it has not yet been configured, the network manager notifies the user that the adapter configuration is incomplete. You can ignore these messages.

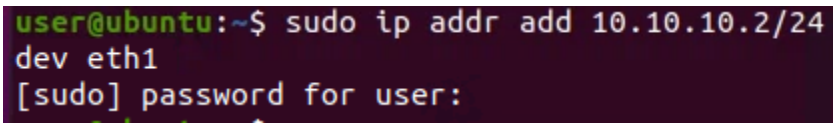
27. In the Linux VM taskbar, **click the terminal icon** to open a terminal if needed.

28. At the command prompt, **execute** `ip a` to view the network adapters.

Note: You should see a new adapter eth1 that has no IPv4 address assigned. A temporary static address can be configured using the ip command.

29. At the command prompt, **execute** `sudo ip addr add 10.10.10.2/24 dev eth1` to assign a static IP address.

When prompted, **type** `password` and **press Enter** to elevate permissions



Execute ip addr command

Note: Now that you have configured both VMs to connect to the internal switch named Data with valid IPv4 addresses, you will ping each VM from the other to confirm the network is working.

30. In the MicroService VM window, at the command prompt, **execute** `ping 10.10.10.2` to try to ping the Linux VM.

31. In the UserApp VM window, at the command prompt, **execute** `ping -w 5 10.10.10.1` to try to ping the Windows VM.

Note: Although the ping from the Windows VM to the Linux VM will work, the other direction does not. Windows Server defaults to having the firewall block incoming packets. Therefore, the ICMP packets used by the ping command are rejected by the firewall and no reply is sent from MicroService to UserApp. To ensure that the ping command from UserApp works, you will update the firewall settings on the Windows VM.

32. In the MicroService VM Taskbar, **click the Magnifying glass** to open a Search box.

33. In the Search box, **type** `firewall` and then **select** the **Windows Defender Firewall with Advanced Security** from the Apps pane to open the Windows Defender Firewall with Advanced Security configuration application window.

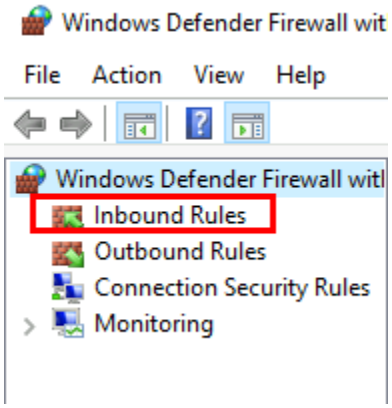
Firewall Search Result

Note: This application allows many things to be configured for the firewall. To enable ping, you need

to enable a preexisting rule. By default, the machine is rejecting network packets that are sent to the machine.

In the next steps, you will enable the rule that will allow ICMP packets for IPv4.

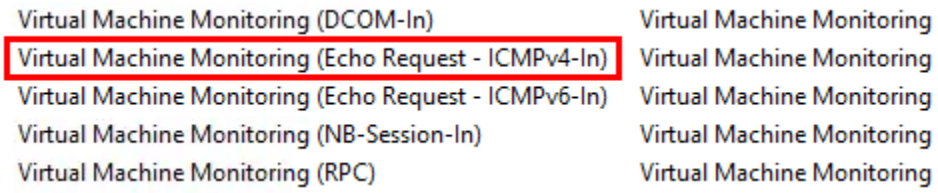
34. In the left pane, **select Inbound Rules** to reveal an alphabetized list of predefined rules.



Select Inbound Rules

35. In the Inbound Rules pane, **scroll down** to reveal the **Virtual Machine Monitoring rules**.

36. In the Inbound Rules pane, **click the rule "Virtual Machine Monitoring (Echo Request - ICMPv4-In)** and then **select Enable Rule** in the Virtual Machine Monitoring… Actions pane.



Echo Request Rule for IPv4

37. **Close** the **Windows Defender Firewall with Advanced Security window**.

Note: You will see a green checkmark appear to the left of the rule. That means the rule is now enabled. You can go back and try both ping commands to verify the network connectivity.

38. **Repeat steps 30–31** to check network connectivity on the Data network.

39. **Make a screen capture** showing the successful ping from the Windows VM to the Linux VM.

40. **Make a screen capture** showing the successful ping from the Linux VM to the Windows VM.

Note: Now that you have confirmed connectivity between the virtual machines through the Data switch, you will close the connections to the virtual machines.

41. **Close** the **UserApp Virtual Machine Connection** window.

42. **Close** the **MicroService Virtual Machine Connection window**.

Challenge and Analysis

Note: The following scenario is provided to allow independent, unguided work – similar to what you will encounter in a real-world situation.

Part 1: Determine State of Hyper-V Integration Services

Your initial proof has gained some traction, and now the senior system administrator is inquiring about the current guest-to-host/host-to-guest functionality enabled on the MicroService VM. They have begun to consider how they might monitor and maintain this application, so are interested in the level of interaction permitted between the guest operating system and the Hyper-V host. You know these capabilities are provided through Hyper-V's Integration Services, so you immediately load up your favorite browser and dig into the documentation.

1. On your own computer, use the Internet to find the PowerShell cmdlet used for managing Hyper-V Integration Services.

2. On the vWorkstation, open PowerShell and then execute the cmdlet you discovered above to show the current state of Integration Services on the MicroService VM.
   Hint: Don't forget to specify the target Hyper-V server in your command with the -ComputerName option! The documentation assumes you are executing these commands directly on the Hyper-V server itself, but vWorkstation is not the Hyper-V server in this environment! Review your commands from Section 1/Part 2 if necessary.

Make a screen capture showing the state of all the MicroService VM's Integration Services in the cmdlet output.

3. Determine which integration service is currently disabled.

4. On your own computer, use the Internet to determine what functionality the disabled service provides, and so is not present on this system.

Describe what guest-to-host functionality is currently unavailable for the MicroService VM.

Part 2: Create Checkpoints of Hyper-V VMs

The development team has been able to start testing with the current configuration of the virtual machines. During a recent test, a bug that seems to be intermittent was discovered. During some manual testing, you were able to reproduce the bug. Now you need to create a checkpoint of the virtual machines so that other people on the team can see the same results.

1. The development team has been able to start testing with the current configuration of the virtual machines. During a recent test, a bug that seems to be intermittent was discovered. During some manual testing, you were able to reproduce the bug. Now you need to create a checkpoint of the virtual machines so that other people on the team can see the same results.

2. Use PowerShell to create a check point of the MicroService VM. A bug was discovered during testing, and we want to be able to reproduce that bug. The first step is to create a checkpoint of each machine.

3. Use Hyper-V Manager to create a check point of UserApp

4. Use PowerShell to create a check point of MicroService.

5. Use PowerShell to list the check points for each VM.

Make a screen capture showing the snapshots for each VM.