

Automating VMware with PowerShell

Hands-On Lab

VMworld Europe 2008

Instructors:	3
1 Lab Summary	3
2 Lab Objectives	3
3 How to Use This Manual	3
4 Your Virtual Environment	5
5 Your Desktop Environment	6
6 An Overview of PowerShell	7
6.1 Introduction	7
6.2 A quick overview of PowerShell.	7
6.3 Getting help for a cmdlet.	7
6.4 Helpful cmdlets for this lab.	7
6.5 Pipelining	9
7 Hands-on Exercises	10
7.1 Getting a List of All cmdlets and Help on Particular cmdlets. (Beginner)	10
7.2 Listing VMs, Hosts, Datastores, And Resource Pools. (Beginner)	11
7.3 Creating a VM. (Beginner)	11
7.4 Basic VM Modification. (Beginner)	12
7.5 Powering VMs On And Off. (Beginner)	12
7.6 Creating VMs From a Template. (Intermediate)	12
7.7 VMotion a Group of VMs. (Intermediate)	13
7.8 Snapshot And Restore VMs. (Intermediate)	13
7.9 Moving VMs Among Virtual Networks. (Intermediate)	14
7.10 Removing All CD-ROM And Floppy Drives. (Intermediate)	14
7.11 Changing Resource Pool Configuration. (Intermediate)	14
7.12 Reading Logs From PowerShell. (Intermediate)	15
7.13 Making Reports From PowerShell. (Intermediate)	15
7.14 Managing Snapshots. (Advanced)	15
7.15 Ensuring All Network Adapters Start Connected. (Advanced)	16
7.16 Setting Resources Based on a Policy. (Advanced)	16
7.17 Modifying VMs based on a policy. (Advanced)	16
7.18 Identifying Over-Utilized Hosts. (Advanced)	17
7.19 Understanding Your Storage Utilization. (Advanced)	17
7.20 Creating a Set of VMs Based on a CSV File. (Advanced)	17
8 Solutions	18
8.1 Getting a List of All cmdlets and Help on Particular cmdlets. (Beginner)	18
8.2 Listing VMs, Hosts, Datastores, And Resource Pools. (Beginner)	18
8.3 Creating a VM. (Beginner)	19
8.4 Basic VM Modification. (Beginner)	19
8.5 Powering VMs On And Off. (Beginner)	19
8.6 Creating VMs From a Template. (Intermediate)	20
8.7 VMotion a Group of VMs. (Intermediate)	20
8.8 Snapshot And Restore VMs. (Intermediate)	20
8.9 Moving VMs Among Virtual Networks. (Intermediate)	21
8.10 Removing All CD-ROM And Floppy Drives. (Intermediate)	21
8.11 Changing Resource Pool Configuration. (Intermediate)	21

8.12	Reading Logs From PowerShell. (Intermediate)	22
8.13	Making Reports From PowerShell. (Intermediate).....	22
8.14	Managing Snapshots. (Advanced)	22
8.15	Ensuring All Network Adapters Start Connected. (Advanced)	23
8.16	Setting Resources Based on a Policy. (Advanced)	23
8.17	Modifying VMs based on a policy. (Advanced).....	23
8.18	Identifying Over-Utilized Hosts. (Advanced).... Error! Bookmark not defined.	
8.19	Understanding Your Storage Utilization. (Advanced).....	24
8.20	Creating a Set of VMs Based on a CSV File. (Advanced)	25
9	Appendix A: References	26

Instructors:

Aidan Dagleish - PSO Consultant, VMware Inc.

Reg Hall - Sr. Systems Engineer, VMware Inc.

Carsten Landeck - Systems Engineer, VMware Inc.

Simon Pfau - Systems Engineer, VMware Inc.

Rodolfo Rotondo - Sr. Systems Engineer, VMware Inc.

Hardev Sanghera - Sr. Systems Engineer, VMware Inc.

Carter Shanklin - Product Manager, VMware Inc.

Andrea Siviero - Channel Systems Engineer, VMware Inc.

1 Lab Summary

Welcome to the Automating VMware with PowerShell: Hands-On Lab. This lab will provide you hands-on experience in writing PowerShell scripts for managing VMware using the VI Toolkit (for Windows). Each participant will create scripts that will show you how to more effectively manage your virtual infrastructure. Some familiarity with PowerShell is helpful, but is not required.

2 Lab Objectives

The objective of this lab is to introduce you to the VI Toolkit (for Windows) and show you how easy automating VMware Virtual Infrastructure management is when you use PowerShell. We will cover a wide range of topics, including things like provisioning, storage, networking and monitoring.

3 How to Use This Manual

This manual contains 5 major sections:

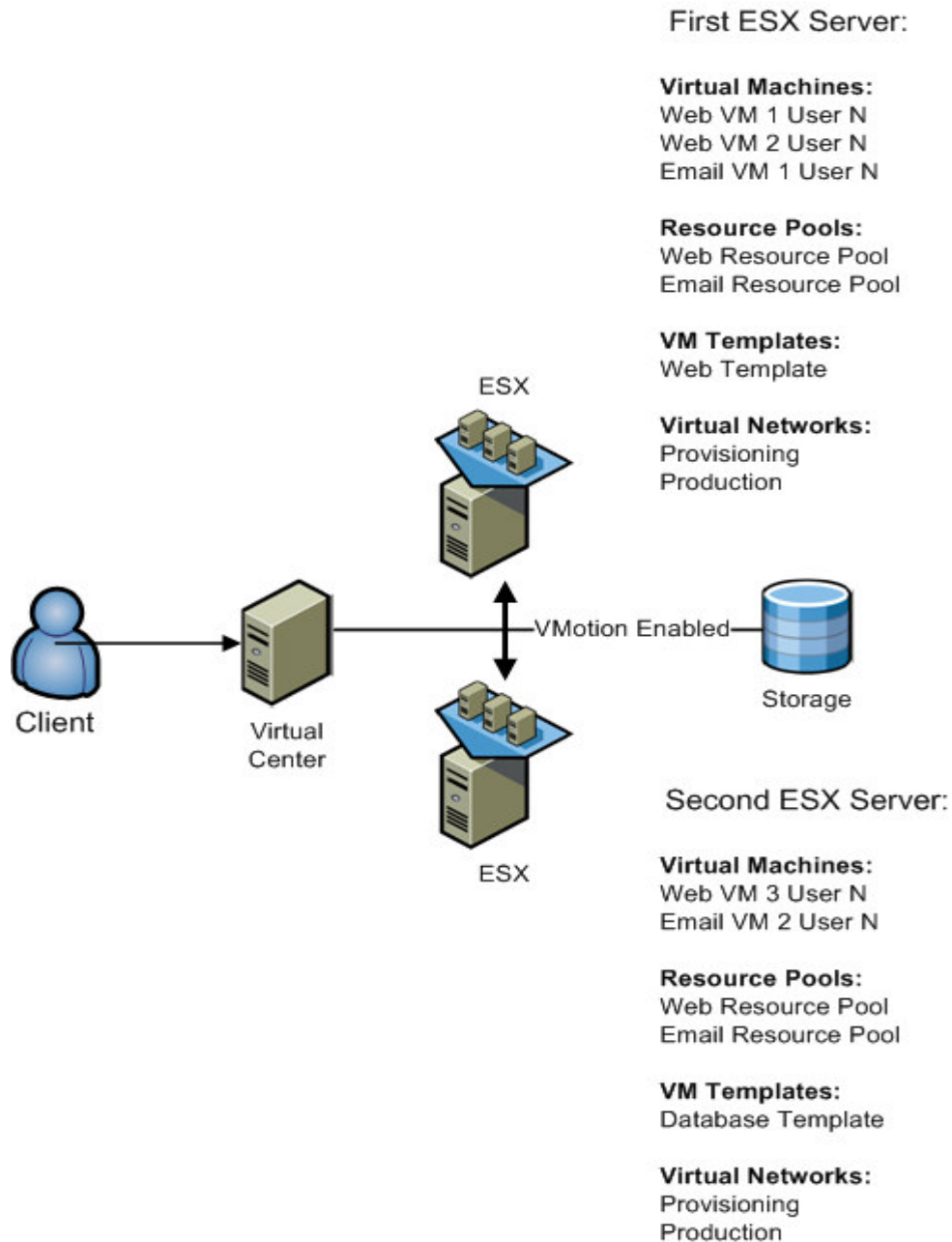
1. A diagram of your virtual environment.
2. A discussion of the desktop environment from which you will perform the exercises.
3. A quick overview of PowerShell.
4. 20 exercises, which you may do in any order.
5. Full solutions to the exercises.

You should take a moment to review sections 4 and 5 and refer back to them during the exercises. If you've never used PowerShell before, read through section 6 and execute the

samples to develop some familiarity. After that you should dive right into the exercises in section 7.

4 Your Virtual Environment

This picture of your virtual environment will help you understand what resources are available to you, and will help you decide how to perform the exercises.



Note:

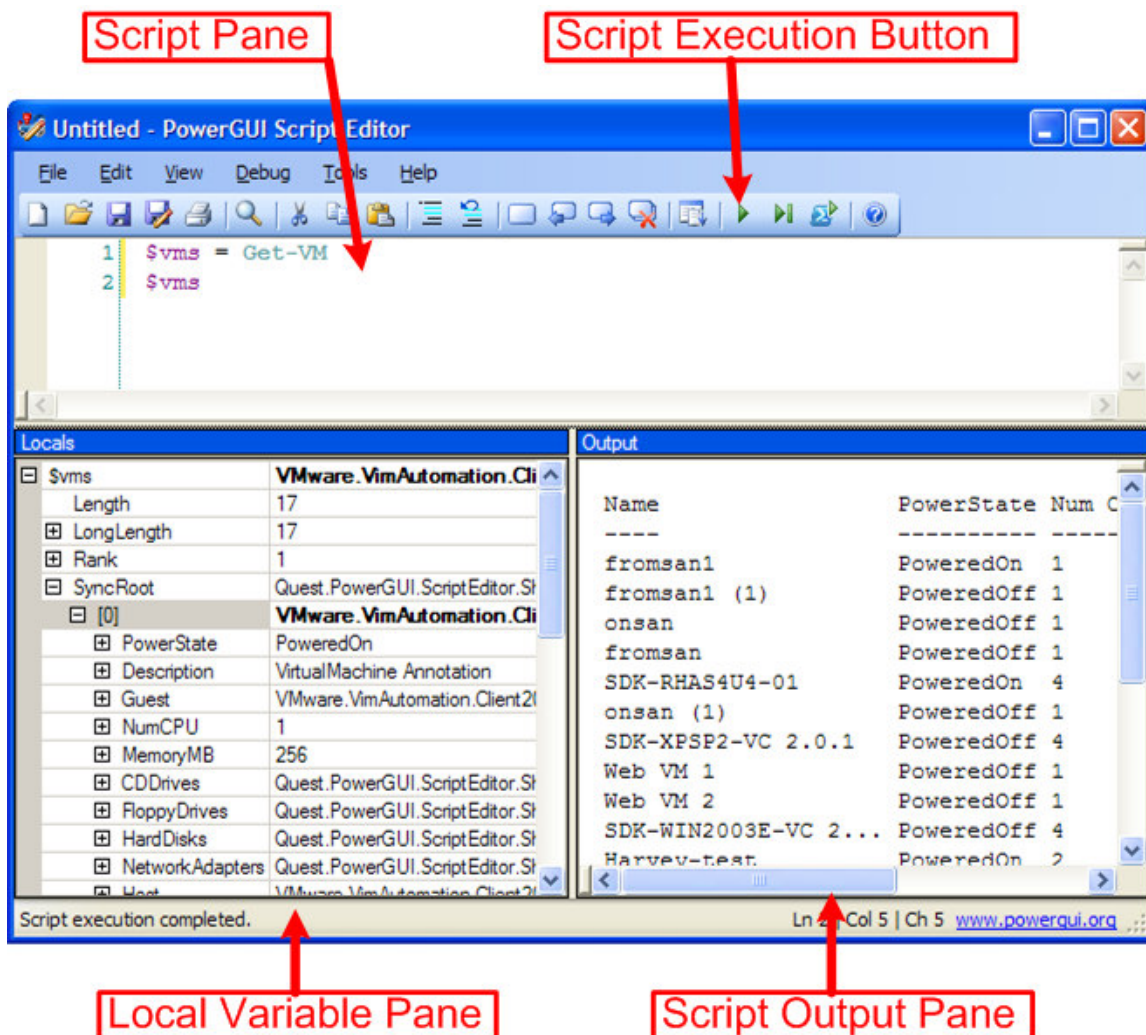
Due to our security configuration you cannot create virtual machines in the root folder. In this lab you must create virtual machines in folders and resource pools that have been configured for you. Your folder name is “studentN”, where N is your student ID, and your two resource pools are studentN-esx1 and studentN-esx2.

5 Your Desktop Environment

In this lab you have two choices for executing scripts. Either choice is available to you as a shortcut on your desktop.

The first is to execute scripts from the VI Toolkit directly, which provides an environment similar to the default PowerShell environment.

The second possible approach is to use the PowerGUI Script Editor. This environment allows you to easily write and debug scripts, as well as inspect the objects you receive as output. If you elect to use PowerGUI, this screen capture shows what your environment will look like. To use the editor, enter your script into the **script pane**, and when you are ready to execute it press the **script execution button**. After the script executes you will see the output in the **script output pane**.



The PowerGUI environment provides argument completion, syntax highlighting, debugging and other nice features, and is a good choice. It is recommended, however, that you use whichever environment is more comfortable for you.

6 An Overview of PowerShell

6.1 Introduction

If you are new to PowerShell, or if you haven't used it in a while, read this section to get a few pointers before you begin.

6.2 A quick overview of PowerShell.

A detailed exposition of PowerShell is outside the scope of this lab, but before we get into the exercises we'll cover a few high points about what PowerShell is, why it's important, and how to manage VMware with PowerShell.

PowerShell is an interactive command-line interpreter, much like the familiar DOS prompt, or the shell environment in UNIX.

What really sets PowerShell apart from other command-line environments is that applications are being built with PowerShell manageability expressly in mind, which means that in the Windows world, PowerShell will become increasingly important for managing large numbers of servers and applications.

The rest of this section covers cmdlets that are especially useful for this lab, and how you can learn more about them.

For more PowerShell resources, please refer to Appendix A.

6.3 Getting help for a cmdlet.

You can get help for a cmdlet by running a script that contains "help <cmdlet name>" from PowerGUI. The help for this cmdlet will be available in the script output pane. You can get much more detailed help for a cmdlet by typing "help <cmdlet name> -full".

6.4 Helpful cmdlets for this lab.

This section discusses the cmdlets you are most likely to see in this lab.

1. **Get-VIServer:** Before you can do any VMware management from PowerShell you must first connect to a VC server or ESX host.

Example: `Get-VIServer 192.168.1.1`

This example connects to the VI server at 192.168.1.1. In order to complete the connection you will be asked for a user name and password.

2. **ForEach-Object:** This cmdlet performs executes a script block for each input object. Input objects are usually piped in to this cmdlet. This cmdlet is usually abbreviated as either `foreach` or `%`.

Example: `dir | ForEach-Object { $_.name }`

This example prints the name of each file (including directories) in the current directory.

- 3. Get-Member:** This cmdlet allows you to inspect an object to determine what methods and properties are defined by the method. The Get-Member cmdlet is often abbreviated as “gm”.

Example: `dir | gm`

This example shows what methods and properties are define for file and/or directory objects.

- 4. Get-VM:** This VMware cmdlet retrieves managed virtual machines.

Example: `Get-VM | select name, memorymb`

This example: displays all VMs in the virtual infrastructure, along with the amount of memory allotted to the VM.

- 5. Get-VMHost:** This VMware cmdlet retrieves VM hosts (

Example: `Get-VMHost | select name`

This example creates custom report about all VM hosts that are available in the virtual infrastructure.

- 6. Measure-Object:** This cmdlet is used to

Example: `dir | Measure-Object`

This example counts the number of files (including directories) in the current directory.

- 7. Select-Object:** This cmdlet is used to select certain properties of input objects.

Select-Object also provides a means of executing a script block to populate a column. This very powerful feature allows you to execute code that uses the object as input and display the results alongside properties from the object itself. This feature is used extensively in this lab.

Example: `dir | Select-Object mode`

This example selects only the mode of the files (including directories) in the current directory.

Example:

```
get-wmiobject win32_service | `
    select Name,
           @{Name="State";
            Expression= {
```



```

        ($_.InterrogateService()).ReturnValue
    }
}

```

This example shows the advanced querying capability provided by select. In this example we display a compact report of services along with their state. The state is obtained by calling the service's InterrogateService method and using the ReturnValue property that is returned.

This capability is used extensively in this lab, so it is worthwhile to become comfortable with it if you plan to do intermediate or advanced exercises.

- 8. Where-Object:** This cmdlet executes a script block against each object, and if script block returns true, emits the input object as an output object. The Where-Object cmdlet is usually used in a pipeline. The Where-Object cmdlet is usually abbreviated as either where or ?.

Example: `dir | Where-Object { $_.mode -eq "d----" }`
 This example returns all subdirectories of the current directory.

6.5 Pipelining

One of PowerShell's most powerful features is the ability to pipe objects between cmdlets, which causes the output objects of one cmdlet to be used as the input objects of another cmdlet. This lab makes extensive use of pipelining.

Example:

```
get-vm | new-snapshot
```

This example takes a snapshot of all VMs in your virtual infrastructure.

To determine what arguments can be piped into a cmdlet, run "help <cmdlet name> -full". In the help you will see a description of all parameters. For example, in the help for the Set-VM cmdlet,

```

-VM <VirtualMachine>
  The virtual machine to update.

Required?                true
Position?                 1
Default value
Accept pipeline input?    true <ByValue>
Accept wildcard characters? false

```

we see that the -VM argument can be passed in as pipeline input. This means that a command like `Get-VM <my VM> | Set-VM -Name "New Name"` is valid, and would rename the VM "New Name".

7 Hands-on Exercises

The exercises in this section are ranked from Beginner to Advanced. The Beginner and Intermediate exercises require little or no prior experience with PowerShell. The advanced exercises require a bit of PowerShell command-line experience, and the Expert exercises require some PowerShell programming experience. If you can't figure out an exercise's solution, don't worry, complete solutions are included in this manual.

The exercises are designed to be completely independent of one another. You can choose to do as few or as many as you like, and you can do them in any order. Before you begin it may be worthwhile to look at the selection of exercises and select the exercises that sound most interesting to you, and do those first.

If you have questions or get stuck, just ask one of the lab assistants for help. Good luck and have fun!

Important: Before you can do anything you must log into the Virtual Center Server. To do this you use the command `Get-VIServer -Server <My VC Server> -user <My Username> -pass <My Password>`. Your server name, username and password have been supplied to you along with this manual. If you are using the PowerGUI Script Editor please run this command once, then delete it or comment it out. Please note, however, that sessions automatically time out after 5 minutes.

7.1 *Getting a List of All cmdlets and Help on Particular cmdlets. (Beginner)*

Scenario:

If you are new to the VI Toolkit (for Windows), or even if you are new to PowerShell, the first thing you want to do is know what cmdlets are available to you. In this exercise we will learn more about what the toolkit is capable of.

Tasks:

1. Get a listing of all cmdlets.
2. Get help on the `get-viserver` cmdlet, which allows you to connect to Virtual Center.
3. Get help on the `new-vm` cmdlet.
4. Get help on the `get-resourcepool` cmdlet that includes an explanation of all supported arguments.

Hint: Within PowerGUI press F5 or click the "Start Debugging" button to execute a script.

Solution:

The full solution is available on Page 18.

7.2 Listing VMs, Hosts, Datastores, And Resource Pools. (Beginner)

Scenario:

You are trying to get acquainted with an unfamiliar VI environment. To help you understand the system, you want to know what VI objects are available and where they are.

Tasks:

1. Connect to Virtual Center.
2. Obtain a listing of all VMs.
3. Obtain a listing of all Hosts.
4. Obtain a listing of all Datastores.
5. Obtain a listing of all Resource Pools.

Hints:

Use the `Get-VIServer` cmdlet to connect to Virtual Center.

Within PowerGUI press F5 or click the “Start Debugging” button to execute a script.

Solution:

The full solution is available on Page 18.

7.3 Creating a VM. (Beginner)

Scenario:

In this example we will go through the process of creating VMs directly without the use of templates. Creating VMs using templates is covered in a later exercise.

Tasks:

1. Create a VM called “First VM User N” on ESX1. Give the VM 256 MB of RAM and 1 GB of disk.
2. Create a VM called “Second VM User N” on ESX2. Give the VM 2 CPUs, and configure it to host Windows 2000 as the guest operating system.

Note:

Due to our security configuration you cannot create virtual machines in the root folder. In this lab you must create virtual machines in folders and resource pools that have been configured for you. Your folder name is “studentN”, where N is your student ID, and your two resource pools are studentN-esx1 and studentN-esx2.

Hints:

Use the `-pool` argument to specify your resource pool and the `-location` argument to specify your folder.

Use `get-vmhost` to specify `new-vm`'s host argument.

Use “win2000ServGuest” to specify Windows 2000 as the guest operating system.

Solution:

The full solution is available on Page 19.

7.4 Basic VM Modification. (Beginner)

Scenario:

A VM you manage, “Email VM 1” has the wrong amount of RAM and the wrong number of CPUs allotted. You need to change the VM’s configuration to the right settings.

Tasks:

1. Change the number of virtual CPUs configured for “Email VM 1” from 1 to 2.
2. Change the amount of RAM configured for “Email VM 1” from 256 MB to 512 MB.
3. Verify that your modifications were successful.

Solution:

The full solution is available on Page 19.

7.5 Powering VMs On And Off. (Beginner)

Scenario:

In this example we will learn how to perform power actions on VMs.

Tasks:

1. Determine the power status of all available VMs.
2. Power on the “Web VM 1” virtual machine.
3. Reset the “Web VM 1” virtual machine.
4. Suspend the “Web VM 1” virtual machine.
5. Resume the “Web VM 1” virtual machine.
6. Power off the “Web VM 1” virtual machine.

Solution:

The full solution is available on Page 19.

7.6 Creating VMs From a Template. (Intermediate)

Scenario:

You are responsible for deploying a new instance of an application. The application is split among two different pieces, a web server and a database server, both of which must be deployed before the application can be used.

Tasks:

1. Deploy a new virtual machine based on the template called “Web Template User N”. Call the new virtual machine “New Web VM”.
2. Deploy a new virtual machine based on the template called “Database Template User N”. Call the new virtual machine “New Database VM”.
3. Power on both virtual machines.

Solution:

The full solution is available on Page 20.

7.7 VMotion a Group of VMs. (Intermediate)

Scenario:

One of your ESX hosts needs a critical patch to be applied. To prepare for this you need to VMotion all hosted VMs to another ESX host.

Tasks:

1. Get a list of all VMs available on ESX1.
2. VMotion all VMs from this source host to ESX2.
3. Ensure that no VMs remain on the source host.
4. Get a listing of all VMs on the destination host to ensure that all VMs have been migrated.

Hint:

If you're having trouble determining the cmdlet to use for step 2, just remember that VMotion is a process for moving VMs from one host to another while they are still powered on.

Note: if you have created virtual machines on local storage on ESX1, as opposed to the shared storage, it will not be possible to VMotion them. If this happens to you, you should check to see if the VMs located on shared storage were VMotioned or not.

Solution:

The full solution is available on Page 20.

7.8 Snapshot And Restore VMs. (Intermediate)

Scenario:

It's Patch Tuesday once again, and security patches need to be deployed to a critical server. Unfortunately you haven't had enough time to fully test the compatibility of the patch, so to minimize risk you want to snapshot VMs before they are patched. If a problem occurs, you will revert to the snapshots before you call Microsoft to yell at them for a few hours.

Tasks:

1. Get a listing of all Email servers.
2. Take a snapshot of all Email servers.
3. Ensure that all snapshots were taken successfully.
4. Revert all Email servers to the snapshot just taken.

Hint:

If you are having trouble determining the cmdlet to use for step 4, think about reverting a snapshot as setting the memory and disk state for a VM to the contents of the VM.

Solution:

The full solution is available on Page 20.

7.9 Moving VMs Among Virtual Networks. (Intermediate)

Scenario:

In one datacenter you manage, VMs are provisioned on one network, and put into production on a different network. Your task is to put a set of VMs into production by moving it from the provisioning network to the production network.

Tasks:

1. Get a list of all VMs on the “Provisioning” virtual network.
2. Move all those VMs to the “Production” virtual network.
3. Ensure that no VMs remain on the “Provisioning” virtual network.

Solution:

The full solution is available on Page 21.

7.10 Removing All CD-ROM And Floppy Drives. (Intermediate)

Scenario:

Virtual CD-ROM and virtual floppies take up small amounts of resources which can make a difference when you have large numbers of VMs. In this exercise we will remove all CD-ROM and floppy drives from all managed VMs in one shot.

Tasks:

1. Obtain a count of the number of CD-ROMs allocated to virtual machines.
2. Obtain a count of the number of floppy drives allocated to virtual machines.
3. Remove all CD-ROMs and floppy drives from all virtual machines.
4. Obtain counts of CD-ROMs and floppies to ensure they are both 0.

Solution:

The full solution is available on Page 21.

7.11 Changing Resource Pool Configuration. (Intermediate)

Scenario:

Your web servers are running slowly because their resource pool does not have enough CPU resources. Your job is to increase web server availability by giving more resources to all web server resource pools

Tasks:

1. Determine the CPU reservation and shares configured for all “Web Servers” resource pools.
2. Increase the CPU reservation for all “Web Servers” resource pools to 2000 MHz.
3. Increase the CPU shares for all “Web Servers” resource pools to High.
4. Ensure your changes were successful.

Solution:

The full solution is available on Page 21.

7.12 Reading Logs From PowerShell. (Intermediate)

Scenario:

In this exercise we will learn how to read log files.

Tasks:

1. Determine what types of logs are available from ESX1.
2. Read the entire messages log file from ESX1.
3. Read the last 10 lines of the hostd log file from ESX1.

Hint:

You are logged into virtual center, but you want to read log files directly from an ESX host attached to virtual center.

`get-log` returns an object that contains all log messages in an element called "Entries".

Solution:

The full solution is available on Page 22.

7.13 Making Reports From PowerShell. (Intermediate)

Scenario:

To help you better understand your virtual environment, you want to run some reports against your Virtual Infrastructure.

Tasks:

1. Determine the average, maximum, minimum and total amounts of memory used by all your virtual machines.
2. Determine the average, maximum, minimum and total amounts of disk used by all your virtual machines.
3. Get a list of all VMs and export it to CSV format. Include the following fields: Name, Description, Host, Power State, Number of CPUs, Memory.

Hints:

The statistics can be computed with the `measure-object` cmdlet.

Use the `export-csv` cmdlet to create CSV files.

Solution:

The full solution is available on Page 22.

7.14 Managing Snapshots. (Advanced)

Scenario:

In this exercise we will go through some useful techniques for dealing with snapshots.

Tasks:

1. Get a listing of all snapshots, along with the VMs to which they belong.
2. Get a listing of all snapshots that are over 1 month old.
3. Get a listing of all VMs that have more than one snapshot.

Solution:

The full solution is available on Page 22.

7.15 Ensuring All Network Adapters Start Connected. (Advanced)

Scenario:

Some of your VMs were mis-configured when they were created. Specifically, some VMs have network adapters that are not automatically connected when the VM is powered on, which means that they are not network accessible unless manual intervention takes place. Your job is to ensure that all your VMs are network accessible when they start.

Tasks:

1. Identify which VMs have network adapters that are not automatically connected.
2. Change each affected VM so that its network adapter will start connected (Note: only change VMs that are affected.)
3. Ensure your changes were successful.

Solution:

The full solution is available on Page 23.

7.16 Setting Resources Based on a Policy. (Advanced)

Scenario:

You have established resource pools for all your virtual machines based on the type of machine used. Currently, none of your VMs conform to the policy. Your job is to ensure all VMs are in the correct resource pool.

Tasks:

1. Move all VMs with Web in the title to the “Web Resources” resource pool.
2. Move all VMs with Email in the title to the “Email Resources” resource pool.
3. Ensure that all VMs are in the correct resource pool.

Solution:

The full solution is available on Page 23.

7.17 Modifying VMs based on a policy. (Advanced)

Scenario:

According to company policy, all web servers must have at least 512 MB of RAM. Your objective is to ensure that all web servers you manage have at least 512 MB of RAM.

Tasks:

1. Identify which web servers have less than 512 MB of RAM. In this exercise, web servers are identified by the string “Web Server” in their name.
2. Reconfigure any affected web server to have 512 MB of RAM.
3. Ensure that all web servers have at least 512 MB of RAM.

Solution:

The full solution is available on Page 23.

7.18 Identifying Over-Utilized Hosts. (Advanced)

Scenario:

In this exercise we'll learn about a few tools that you can use to rank your hosts from most utilized to least utilized. After making such a ranking, the other techniques discussed in this lab can be used to better optimize overall performance, by moving VMs from heavily utilized hosts to less utilized hosts.

Tasks:

1. Sort available hosts by the number of VMs hosted.
2. Sort available hosts by average CPU utilization over the past hour.
3. Sort available hosts by average memory utilization at this moment.

Hint: For tasks 2 and 3, use the `get-stat` cmdlet.

Solution:

The full solution is available on Page **Error! Bookmark not defined.**

7.19 Understanding Your Storage Utilization. (Advanced)

Scenario:

You want to know how much storage you have available and what storage is used where.

Tasks:

1. Determine how much storage is available in your virtual infrastructure.
2. Determine how much storage is used by VMs that are powered off.
3. Determine the 3 VMs that use the most storage space.

Solution:

The full solution is available on Page 24.

7.20 Creating a Set of VMs Based on a CSV File. (Advanced)

Scenario:

In this scenario we have a template that defines what virtual machines we want, where they should reside, and the configurations of each. The objective is to create all of them in a highly automated way.

Your environment contains a CSV file called `template.csv`. A sample of the data in this file follows, though the host and VM names in your file may differ.

Name	Host	DiskMB	MemoryMB
App Server 1 User 1	ESX1	50	128
Database User 1	ESX1	50	128
App Server 2 User 1	ESX2	100	128
Web Server User 1	ESX2	50	256

Your objective is to create all these VMs automatically using a single command, based on the data available in `template.csv`.

Tasks:

1. Create all VMs listed above in an automated way.
2. Ensure your actions were successful.

Hint:

Use the `import-csv` PowerShell cmdlet to read the data in. Type “`help import-csv -full`” to learn more about importing CSV.

Solution:

The full solution is available on Page 25.

8 Solutions

This section contains the solutions to the exercises of section 6. Each task is repeated in a table below. The solution column of the table contains the PowerShell command that can be executed to accomplish the given task.

8.1 Getting a List of All cmdlets and Help on Particular cmdlets. (Beginner)

Task	Solution
Get a listing of all cmdlets.	<code>get-vicommand</code> OR (PowerGUI only) <code>get-command -pssnapin `VMware.VimAutomation.Core</code>
Get help on the <code>get-viserver</code> cmdlet, which allows you to connect to Virtual Center.	<code>help get-viserver</code>
Get help on the <code>new-vm</code> cmdlet.	<code>help new-vm</code>
Get help on the <code>get-resourcepool</code> cmdlet that includes an explanation of all supported arguments.	<code>help get-resoucepool -full</code>

8.2 Listing VMs, Hosts, Datastores, And Resource Pools. (Beginner)

Task	Solution
------	----------

Connect to Virtual Center.	<code>get-vc virtualcenter</code>
Obtain a listing of all VMs.	<code>get-vm</code>
Obtain a listing of all Hosts.	<code>get-vmhost</code>
Obtain a listing of all Datastores.	<code>get-datastore</code>
Obtain a listing of all Resource Pools.	<code>get-resourcepool</code>

8.3 Creating a VM. (Beginner)

Task	Solution
Create a VM called "First VM User N" on ESX1. Give the VM 256 MB of RAM and 1 GB of disk.	<code>get-vmhost ESX1 new-vm -name "First VM User N" -memorymb 256 -diskmb 1024 -pool (get-resourcepool studentN-esx1) -location (get-folder studentN)</code>
3. Create a VM called "Second VM User N" on ESX2. Give the VM 2 CPUs, and configure it to host Windows 2000 as the guest operating system.	<code>get-vmhost ESX2 new-vm -name "Second VM User N" -memorymb 256 -diskmb 1024 -guestid win2000ServGuest -pool (get-resourcepool studentN-esx2) -location (get-folder studentN)</code>

8.4 Basic VM Modification. (Beginner)

Task	Solution
Change the number of virtual CPUs configured for "Email VM 1" from 1 to 2.	<code>get-vm "Email VM 1 User N" set-vm -numcpu 2</code>
Change the amount of RAM configured for "Email VM 1" from 256 MB to 512 MB.	<code>get-vm "Email VM 1 User N" set-vm -memorymb 512</code>
Verify that your modifications were successful.	<code>get-vm "Email VM 1 User N"</code>

8.5 Powering VMs On And Off. (Beginner)

Task	Solution
Determine the power status of all available VMs.	<code>get-vm ft name, powerstate</code>
Power on the "Web VM 1" virtual machine.	<code>get-vm "Web VM 1 User N" start-vm</code>
Reset the "Web VM 1" virtual machine.	<code>get-vm "Web VM 1 User N" stop-vm start-vm</code>
Suspend the "Web VM 1" virtual machine.	<code>get-vm "Web VM 1 User N" suspend-vm</code>
Resume the "Web VM 1" virtual machine.	<code>get-vm "Web VM 1 User N" start-vm</code>
Power off the "Web VM 1"	<code>get-vm "Web VM 1 User N" stop-vm</code>

virtual machine.	
------------------	--

8.6 Creating VMs From a Template. (Intermediate)

Task	Solution
Deploy a new virtual machine based on the template called "Web Template User N". Call the new virtual machine "New Web VM".	<pre>new-vm -template ` (get-template "Web Template") ` -name "New Web VM User N" ` -host (get-vmhost esx1) ` -pool (get-resourcepool studentN-esx1) ` -location (get-folder studentN)</pre>
Deploy a new virtual machine based on the template called "Database Template User N". Call the new virtual machine "New Database VM".	<pre>new-vm -template ` (get-template "Database Template") ` -name "New Database VM User N" ` -host (get-vmhost esx2) ` -pool (get-resourcepool studentN-esx2) ` -location (get-folder studentN)</pre>
Power on both virtual machines.	<pre>get-vm "New Web VM" start-vm get-vm "New Database VM" start-vm</pre>

8.7 VMotion a Group of VMs. (Intermediate)

Task	Solution
Get a list of all VMs available on ESX1.	<pre>get-vmhost ESX1 get-vm</pre>
VMotion all VMs from this source host to ESX2.	<p>Note: You must disconnect CD-ROM and floppy drives before VMotion can succeed. See exercise 7.10 if you need help with removing drives.</p> <pre>get-vmhost ESX1 get-vm ` move-vm -destination (get-resourcepool ` studentN-esx2)</pre>
Ensure that no VMs remain on the source host.	<pre>get-vmhost ESX1 get-vm measure-object</pre>
Get a listing of all VMs on the destination host to ensure that all VMs have been migrated.	<pre>get-vmhost ESX2 get-vm</pre>

8.8 Snapshot And Restore VMs. (Intermediate)

Task	Solution
Get a listing of all Email servers.	<pre>get-vm *email*</pre>
Take a snapshot of all Email servers.	<pre>get-vm *email* ` foreach { ` \$_ new-snapshot -name (\$_.Name + "-` Current") ` }</pre>

Ensure that all snapshots were taken successfully.	<code>get-vm *email* get-snapshot</code>
Revert all Email servers to the snapshot just taken.	<code>get-vm *email* get-snapshot ` foreach { ` \$_.VM set-vm -snapshot \$_ ` -confirm:\$false ` }</code>

8.9 Moving VMs Among Virtual Networks. (Intermediate)

Task	Solution
Get a list of all VMs on the “Provisioning” virtual network.	<code>get-vm where { ` (\$_ ` get-networkadapter).NetworkName -eq ` “Provisioning” ` }</code>
Move all those VMs to the “Production” virtual network.	<code>get-vm where { ` (\$_ ` get-networkadapter).NetworkName ` -eq “Provisioning” ` } get-networkadapter ` set-networkadapter -networkname ` “Production”</code>
Ensure that no VMs remain on the “Provisioning” virtual network.	<code>get-vm where { ` (\$_ ` get-networkadapter).NetworkName ` -eq “Provisioning” ` } measure-object</code>

8.10 Removing All CD-ROM And Floppy Drives. (Intermediate)

Task	Solution
Obtain a count of the number of CD-ROMs allocated to virtual machines.	<code>get-vm get-cddrive measure-object</code>
Obtain a count of the number of floppy drives allocated to virtual machines.	<code>get-vm get-floppydrive measure-object</code>
Remove all CD-ROMs and floppy drives from all virtual machines.	<code>get-vm get-cddrive remove-cddrive ` -confirm:\$false</code> <code>get-vm get-floppydrive ` remove-floppydrive -confirm:\$false</code>
Obtain counts of CD-ROMs and floppies to ensure they are both 0.	<code>get-vm get-cddrive measure-object</code> <code>get-vm get-floppydrive measure-object</code>

8.11 Changing Resource Pool Configuration. (Intermediate)

Task	Solution
Determine the CPU reservation and shares configured for all	<code>get-resourcepool “Web Resource Pool” ` select Name, CpuSharesLevel, ` CpuReservationMhz</code>

“Web Servers” resource pools.	
Increase the CPU reservation for all “Web Servers” resource pools to 2000 MHz.	<pre>get-resourcepool "Web Resource Pool" ` foreach { ` set-resourcepool -ResourcePool \$_ ` -cpureservationmhz 2000 }</pre>
Increase the CPU shares for all “Web Servers” resource pools to High.	<pre>get-resourcepool "Web Resource Pool" ` foreach { ` set-resourcepool -ResourcePool \$_ ` -cpushareslevel "High" }</pre>
Ensure your changes were successful.	<pre>get-resourcepool "Web Resource Pool" ` select Name, CpuSharesLevel, ` CpuReservationMhz</pre>

8.12 Reading Logs From PowerShell. (Intermediate)

Task	Solution
Determine what types of logs are available from ESX1.	<pre>get-logtype -host (get-vmhost esx1)</pre>
4. Determine what types of logs are available from ESX1. Read the entire messages log file from ESX1.	<pre>(get-log -host (get-vmhost esx1) ` messages).Entries</pre>
Read the last 10 lines of the hostd log file from ESX1.	<pre>(get-log -host (get-vmhost esx1) ` hostd).Entries[-10..-1]</pre>

8.13 Making Reports From PowerShell. (Intermediate)

Task	Solution
Determine the average, maximum, minimum and total amounts of memory used by all your virtual machines.	<pre>get-vm ` measure-object -property Memory* ` -average -maximum -minimum</pre>
Determine the average, maximum, minimum and total amounts of disk used by all your virtual machines.	<pre>(get-vm ` foreach { ` (\$_ get-harddisk measure-object ` -property CapacityKB -sum).Sum ` } measure-object -average -sum ` -maximum -minimum</pre>
Get a list of all VMs and export it to CSV format. Include the following fields: Name, Description, Host, Power State, Number of CPUs, Memory.	<pre>get-vm select Name, Description, ` PowerState, Num*, Memory*, ` @{Name="Host"; Expression={\$_.Host.Name}} ` export-csv output.csv</pre>

8.14 Managing Snapshots. (Advanced)

Task	Solution
Get a listing of all snapshots,	<pre>get-vm get-snapshot `</pre>

along with the VMs to which they belong.	<pre>select Name, @{"Name="VM Name" ; ` Expression={\$_.VM.Name}}</pre>
Get a listing of all snapshots that are over 1 month old.	<pre>get-vm get-snapshot where { ` \$_.Created -lt (get-date).addmonths(-1) ` }</pre>
Get a listing of all VMs that have more than one snapshot.	<pre>get-vm where { ` (\$_ get-snapshot ` measure-object).count -gt 0 ` }</pre>

8.15 Ensuring All Network Adapters Start Connected. (Advanced)

Task	Solution
Identify which VMs have network adapters that are not automatically connected.	<pre>get-vm where { ` \$_ get-networkadapter where { ` \$_.ConnectionState.StartConnected -eq 0 ` } ` }</pre>
Change each affected VM so that its network adapter will start connected (Note: only change VMs that are affected.)	<pre>get-vm get-networkadapter where { ` \$_.connectionstate.startconnected -eq 0 ` } set-networkadapter -startconnected:\$true</pre>
Ensure your changes were successful.	<pre>get-vm where { ` \$_ get-networkadapter where { ` \$_.ConnectionState.StartConnected -eq 0 ` } ` } measure-object</pre>

8.16 Setting Resources Based on a Policy. (Advanced)

Task	Solution
Move all VMs with Web in the title to the "Web Resources" resource pool.	<pre>foreach (\$h in @("esx1", "esx2")) { get-vm -location (get-vmhost \$h) *web* ` move-vm -destination (get-resourcepool ` -location (get-vmhost \$h) ` "Web Resource Pool") }</pre>
Move all VMs with Email in the title to the "Email Resources" resource pool.	<pre>foreach (\$h in @("esx1", "esx2")) { get-vm -location (get-vmhost \$h) *email* ` move-vm -destination (get-resourcepool ` -location (get-vmhost \$h) ` "Email Resource Pool") }</pre>
Ensure that all VMs are in the correct resource pool.	<pre>get-resourcepool "Web Resource Pool" ` get-vm get-resourcepool "Email Resource Pool" ` get-vm</pre>

8.17 Modifying VMs based on a policy. (Advanced)

Task	Solution
Identify which web servers have less than 512 MB of RAM. In	<pre>get-vm *web* where { \$_.memorymb -lt 512 }</pre>

this exercise, web servers are identified by the string “Web Server” in their name.	
Reconfigure any affected web server to have 512 MB of RAM.	<pre>get-vm *web* where ` { \$_.memorymb -lt 512 } ` set-vm -memorymb 512</pre>
Ensure that all web servers have at least 512 MB of RAM.	<pre>get-vm *web* where ` { \$_.memorymb -lt 512 } ` measure-object</pre>

8.18 Identifying Over-Utilized Hosts. (Advanced)

Task	Solution
Sort available hosts by the number of VMs hosted.	<pre>get-vmhost select name, ` @{ name="NVMs"; ` expression = { ` (\$_ get-vm measure-object).count ` } ` } sort -property NVMs -descending</pre>
Sort available hosts by average CPU utilization over the past hour.	<pre>get-vmhost select name, ` @{ name="CPU"; ` expression = { ` (\$_ get-stat -stat "cpu.usage.average" ` -maxsamples 12 measure-object -average ` -property value).average ` } ` } sort -property CPU -descending</pre>
Sort available hosts by average memory utilization at this moment.	<pre>get-vmhost ` select name, ` @{ name="Memory"; ` expression = { (\$_ get-stat -stat ` "mem.usage.average" -maxsamples 1 ` -rt -property value).average } } ` sort -property Memory -descending</pre>

8.19 Understanding Your Storage Utilization. (Advanced)

Task	Solution
Determine how much storage is available in your virtual infrastructure.	NOTE: Though the field name is FreeSpaceMB, the value returned is actually in bytes. This bug will be addressed in our Beta release. (continued) <pre>get-datastore measure-object -sum ` -property FreeSpaceMB</pre>
Determine how much storage is used by VMs that are powered off.	<pre>get-vm where ` { \$_.PowerState -eq "PoweredOff" } ` get-harddisk measure-object ` -sum -property capacityKB</pre>
Determine the 3 VMs that use the most storage space.	<pre>get-vm select Name, @{ Name="TotalDisk"; ` Expression={ (\$_ get-harddisk ` measure-object -property CapacityKB `</pre>

	<pre>-sum).Sum }} sort -property TotalDisk ` -descending select-object -first 3</pre>
--	---

8.20 Creating a Set of VMs Based on a CSV File. (Advanced)

Task	Solution
Create all VMs listed above in an automated way.	<pre>foreach (\$f in (import-csv ` "template.csv")) { ` get-vmhost \$f.host new-vm -name ` \$f.Name -diskmb \$f.diskmb -memorymb ` \$f.memorymb ` -pool (get-resourcepool \$f.pool) ` -location (get-folder \$f.folder) }</pre>
Ensure your actions were successful.	<pre>foreach (\$f in (import-csv ` "template.csv")) { ` get-vm \$f.Name ` }</pre>

9 Appendix A: References

The following links will give you information relevant to the VI Toolkit (for Windows) as well as information about scripting in PowerShell.

VMware Resources

VMware Developer Center: <http://vmware.com/developer>

VI PowerShell Blog: <http://blogs.vmware.com/vipowershell/>

PowerShell Documentation Resources

Windows PowerShell Documentation Pack

<http://www.microsoft.com/downloads/details.aspx?FamilyId=B4720B00-9A66-430F-BD56-EC48BFCA154F&displaylang=en>

Windows PowerShell Owner's Manual

<http://www.microsoft.com/technet/scriptcenter/topics/winsh/manual/default.aspx>

Free Windows PowerShell book

<https://blogs.technet.com/chitpro-de/archive/2007/05/10/english-version-of-windows-powershellcourse-book-available-for-download.aspx>

What Can I Do With Windows PowerShell

<http://www.microsoft.com/technet/scriptcenter/topics/msh/cmdlets/index.aspx>

Mastering PowerShell in Your Lunch Break

<http://powershelllive.com/blogs/lunch/archive/2007/03/26/day-1-getting-organized.aspx>

An Introduction to Windows PowerShell

<http://www.slideshare.net/dalelane/an-introduction-to-windows-powershell/>