

VMware PowerCLI User's Guide

VMware PowerCLI™

You can find the most up-to-date technical documentation on the VMware website at:

<https://docs.vmware.com/>

VMware, Inc.
3401 Hillview Ave.
Palo Alto, CA 94304
www.vmware.com

Copyright © 1998-2023 VMware, Inc. All rights reserved. [Copyright and trademark information.](#)

Contents

VMware PowerCLI User's Guide 10

Updated Information 11

1 Introduction to VMware PowerCLI 12

Microsoft PowerShell Basics 12

PowerShell Command-Line Syntax 13

PowerShell Pipelines 13

PowerShell Wildcards 13

PowerShell Common Parameters 13

PowerCLI Concepts 14

PowerCLI Modules 15

Retrieving vSphere Inventory Objects from Cloud Resources 17

Selecting Objects in PowerCLI 18

Processing Non-alphanumeric Characters in PowerCLI 18

Running PowerCLI Cmdlets Asynchronously 19

Managing Default Server Connections in PowerCLI 19

Customization Specification Objects in PowerCLI 20

Using ESXCLI with PowerCLI 20

Using the PowerCLI Inventory Provider 21

Using the PowerCLI Datastore Provider 21

Using the PowerCLI About Articles 21

2 Installing VMware PowerCLI 23

Supported Operating Systems 24

Supported VMware Products 24

Supported PowerShell Versions 24

Install PowerCLI 24

Install PowerCLI Offline 25

Update PowerCLI 26

Uninstall PowerCLI 26

3 Configuring VMware PowerCLI 28

Allow Execution of Local Scripts 28

Configuring PowerCLI Response to Untrusted Certificates 28

Configure the PowerCLI Response to Untrusted Certificates 29

PowerCLI Trusted Certificate Store 30

Scoped Settings of vSphere PowerCLI 30

Configuring the Scope of the PowerCLI Settings	30
Priority of Settings Scopes in PowerCLI	31
PowerCLI Configuration Files	31
Installing and Configuring Python for PowerCLI	32
Install and Configure Python on Windows	33
Install and Configure Python on macOS	33
Install and Configure Python on Linux	34
4 Configuring Customer Experience Improvement Program	35
Categories of Information That VMware Receives	35
Join the Customer Experience Improvement Program in PowerCLI	35
5 Managing vSphere with VMware PowerCLI	36
Connect to a vCenter Server System	41
Connect to an AD FS - Federated vCenter Server System	41
Manage Virtual Machines on vSphere	43
Add a Standalone Host to a vCenter Server System	44
Set the License Key for a Host on vCenter Server	44
Activate Maintenance Mode for a Host on vCenter Server	45
Create vSphere Inventory Objects	46
Create Virtual Machines on vCenter Server Using an XML Specification File	47
Manage Virtual Machine Templates on vCenter Server	47
Create and Use Snapshots on vCenter Server	48
Update the Resource Configuration Settings of a Virtual Machine on vCenter Server	49
Get a List of Hosts on a vCenter Server System and View Their Properties	49
Change the Host Advanced Configuration Settings on vCenter Server	50
Move a Virtual Machine to a Different Host Using VMware vSphere vMotion	51
Move a Virtual Machine to a Different Datastore Using VMware vSphere Storage vMotion	51
Move a Virtual Machine to a Different vCenter Server System	52
Create a Host Profile on a vCenter Server System	52
Apply a Host Profile to a Host on vCenter Server	53
Manage Statistics and Statistics Intervals on vCenter Server	54
Modify the Settings of the NIC Teaming Policy for a Virtual Switch	54
Create a vApp on vCenter Server	55
Modify the Properties of a vApp	55
Export or Import vApps	56
Create an iSCSI Host Storage	56
Add Passthrough Devices to a Host and Virtual Machine	57
Create a Custom Property Based on an Extension Data Property	57
Create a Script-Based Custom Property for a vSphere Object	58
Apply a Customization Object to a Cloned Virtual Machine	58

Modify the Default NIC Mapping Object of a Customization Specification	59
Modify Multiple NIC Mapping Objects of a Customization Specification	60
Create Multiple Virtual Machines that Use Static IP Addresses	61
Create Multiple Virtual Machines with Two Network Adapters	62
Create a vSphere Role and Assign Permissions to a User	63
View Actions and Triggers for an Alarm on vCenter Server	64
Create Alarm Definitions on vCenter Server	65
Modify Alarm Definitions on vCenter Server	65
Create Alarm Actions and Triggers on vCenter Server	66
Remove Alarm Definitions and Actions	66
Create and Modify Advanced Settings for a Cluster	67
Modify the vCenter Server Email Configuration	67
Modify the vCenter Server SNMP Configuration	68
Use Esxtop to Get Information on the Virtual CPUs of a Virtual Machine	68
Filter vSphere Objects with Get-View	69
Populate a View Object with Get-View	70
Update the State of a Server-Side Object	70
Reboot a Host with Get-View	71
Modify the CPU Levels of a Virtual Machine with Get-View and Get-VIObjectByVIView	72
Browse the Default Inventory Drive	72
Create a New Custom Inventory Drive	73
Manage Inventory Objects Through Inventory Drives	73
Browse the Default Datastore Drives	74
Create a New Custom Datastore Drive	75
Manage Datastores Through Datastore Drives	75
Modify the Timeout Setting for Web Tasks	76
Using Tags	77
Retrieve a Tag and Save It into a Variable	77
Retrieve a Tag Category and Save It into a Variable	78
Create a Tag Category and a Tag	78
Assign a Tag to Virtual Machines	78
Retrieve Objects by Tag	79
Generate Tags Automatically by Using a Script	79
Add an Entity Type to a Tag Category	80
Retrieve Tag Assignments	80
Network Management with vSphere Distributed Switches	80
Create a Distributed Switch and Configure Networking	81
Configure a Distributed Switch	82
Migrate Virtual Machine Networking Configuration from a vSphere Standard Switch to a vSphere Distributed Switch	82
Migrate Physical and Virtual NICs to a vSphere Standard Switch	83
Migrate Physical and Virtual NICs to a vSphere Distributed Switch	84

- Configure the Traffic Shaping Policy 84
- Configure the Security Policy 85
- Create a Local Content Library 85
- Create a Subscribed Content Library 85
- Create a Content Library Item 86
- Create a Virtual Machine from a Content Library Item 86
- Create a vApp from a Content Library Item 87
- Export Content Library Item's Files to a Local Machine 87
- Create a New VM-VM DRS Rule 88
- Create a New VM-VMHost DRS Rule 88
- Managing Certificates 89
 - Add a Root Certificate to vCenter Server and ESXi 89
 - Change the Machine SSL Certificate of vCenter Server 90
 - Change the Machine SSL Certificate of an ESXi Host 91

6 Managing the vSphere Automation API with VMware PowerCLI 94

- Understanding the vSphere Automation SDK for PowerShell 94
- Navigating the vSphere Automation SDK for PowerShell 95
- Connecting to a vSphere Automation API Server 96
 - Connect Through `Connect-VIServer` 96
 - Connect Through a Configuration Object 96
- Create a Local User Account in vCenter Server 98
- Update the Local Accounts Global Password Policy in vCenter Server 99
- Create a Virtual Machine 99
- Create Tag Category, Tag, and Tag Association 100
- Create Content Library and Content Library Item 100

7 Managing vSphere Policy-Based Storage with VMware PowerCLI 102

- Create a Tag-Based Storage Policy 102
- Create a Capability-Based Storage Policy 103
- Associate a Storage Policy with a Virtual Machine and Its Hard Disk 104
- Disassociate a Storage Policy Associated with a Virtual Machine and Its Hard Disk 105
- Remove a Storage Policy 105
- Edit a Storage Policy 106
- Export and Import a Storage Policy 106
- Create a Virtual Machine in a Datastore Compatible with Storage Policy 107
- Create a vSAN Datastore 108
- Modify a vSAN Datastore 110
- Create a vSAN Stretched Cluster 110
- Enable a vSAN File Service 111
- Create a vSAN File Service Domain 112

- Create a vSAN File Share 113
- Create an NFS 4.1 Datastore 113
- Add a VASA Provider and Create a Policy 114
- Invoke a Planned Failover on a Replication Group and Reverse the Replication 115
- Attach a Flat VDisk to a Virtual Machine 117

8 Managing the vSphere Replication API with VMware PowerCLI 119

- Understanding the vSphere Replication SDK for PowerShell 119
- Navigating the vSphere Replication SDK for PowerShell 120
- Connect to a Local vSphere Replication Server 121
- Connect to a Local and to a Remote vSphere Replication Server 122
- Use the ConnectedPairings Property 123
- Replicate a VM to a Datastore on the Local vCenter Server System 123
- Replicate a VM to a Datastore on a Remote vCenter Server System 125

9 Managing VMware Site Recovery Manager (SRM) with VMware PowerCLI 128

- Connect to an SRM Server 128
- Protect a Virtual Machine 129
- Create a Report of the Protected Virtual Machines 130
- Create a Report of the Virtual Machines Associated with All Protection Groups 131

10 Managing the Site Recovery Manager (SRM) API with VMware PowerCLI 132

- Understanding the Site Recovery Manager SDK for PowerShell 132
- Navigating the Site Recovery Manager SDK for PowerShell 133
- Connect to a Local and to a Remote Site Recovery Manager (SRM) Server 134
- Host-Based Replication Scenarios 135
 - Create Protection Group for Host-Based Replication 135
 - Create Recovery Plan 136
- Array-Based Replication Scenarios 137
 - Configure Array Managers for an Array Pair 137
 - Create Replicated Array Pair 140
 - Create Protection Group and Recovery Plan for ABR 141

11 Managing the NSX Policy API with VMware PowerCLI 144

- Understanding the NSX Policy SDK for PowerShell 144
- Navigate the NSX Policy SDK for PowerShell 145
- Connect to an On-Prem NSX Server 146
- Connect to an NSX Server on VMware Cloud on AWS 146
- Create a Tier-0 Gateway 147
- Create a Tier-1 Gateway 147
- Add an Existing Tier-1 Gateway to a Specific Edge Cluster 148

- Create a Segment (On-Prem) 148
- Create a Segment (VMware Cloud on AWS) 149
- Create a Distributed Firewall Policy 150

12 Managing VMware Cloud Director with VMware PowerCLI 152

- Connect to a VMware Cloud Director Server 153
- Create and Manage Organizations 154
- Create and Manage Organization Virtual Data Centers 154
- Filter and Retrieve Organization Virtual Data Center Networks 155
- Import a vApp Template from the Local Storage 156
- Create a vApp Template from a vApp 156
- Import a vApp from vSphere 157
- Create and Modify a vApp 158
- Manage Virtual Machines with vApps 158
- Manage Virtual Machines and Their Guest Operating Systems 159
- Retrieve a List of the Internal and External IP Addresses of Virtual Machines in vApps 160
- Create and Manage Access Control Rules 161
- Filter and Retrieve vApp Networks 161
- Create vApp Networks for a Selected vApp 162
 - Create an Isolated vApp Network 162
 - Create an NAT Routed vApp Network 163
 - Create a Direct vApp Network 163
- Modify or Remove vApp Networks 164

13 Managing vSphere Update Manager with VMware PowerCLI 166

- Connect to a vCenter Server System 166
- Create Patch Baselines 167
- Attach and Detach Baselines 168
- Scan a Virtual Machine 168
- Check Virtual Machine Baseline Status 169
- Stage Patches 169
- Remediate a Virtual Machine 170
- Upgrade Virtual Machine Hardware 170
- Remediate a Cluster 171
- Remediate a Host 171
- Download Patches and Scan Objects 172

14 Managing vSphere Lifecycle Manager with VMware PowerCLI 173

- Understanding vSphere Lifecycle Manager 173
- Creating and Managing vSphere Lifecycle Manager Clusters with PowerCLI 175
 - Sample Scripts for vSphere Lifecycle Manager Cluster-Level Operations 176

Managing vSphere Lifecycle Manager Standalone Hosts with PowerCLI	180
Sample Scripts for vSphere Lifecycle Manager Host-Level Operations	181
15 Managing VMware Aria Operations with VMware PowerCLI	184
Connect to a VMware Aria Operations Server	184
Check Memory Waste Levels	185
Get Remediation Recommendations	186
Change Alert Ownership	186
Create a Report for Problematic Hosts	187
16 Managing VMware Cloud on AWS with VMware PowerCLI	188
Connecting to VMware Cloud on AWS	188
Connect to VMware Cloud on AWS by Using an API Token	189
Connect to VMware Cloud on AWS by Using an OAuth Security Context	189
Connect to VMware Cloud on AWS GovCloud (US)	189
View the Available Software-Defined Data Centers	190
Connect to a vCenter Server on VMware Cloud on AWS	190
Connect to a vCenter Server on VMware Cloud on AWS by Using an OAuth 2.0 Authentication	191
Create a Software-Defined Data Center	191
Create a Cluster in a Software-Defined Data Center	192
Set the Elastic Distributed Resource Scheduler (EDRS) Policy of a Cluster	192
Remove a Cluster from a Software-Defined Data Center	193
Add Hosts to a Software-Defined Data Center	193
Remove Hosts from a Software-Defined Data Center	194
17 Help and Support for VMware PowerCLI	195
Generate a PowerCLI Support Bundle	196

VMware PowerCLI User's Guide

The *VMware PowerCLI User's Guide* provides information about installing and using the VMware PowerCLI cmdlets (pronounced “commandlets”) for managing, monitoring, automating, and handling operations for VMware® vSphere, VMware Site Recovery Manager (SRM), vSphere Automation API, vSAN, VMware Cloud Director, vSphere Update Manager, VMware Aria Operations, VMware Horizon, VMware NSX, VMware HCX, and VMware Cloud on AWS components.

To help you start with PowerCLI, this documentation includes descriptions of specific PowerCLI concepts and features. In addition, this documentation provides a set of use case examples and sample scripts.

Intended Audience

This guide is intended for anyone who wants to install and use PowerCLI. This documentation is written for administrators and developers who are familiar with virtual machine technology and PowerShell.

- Basic administrators can use cmdlets included in PowerCLI to manage their vSphere, VMware Site Recovery Manager (SRM), vSphere Automation API, vSAN, VMware Cloud Director, vSphere Update Manager, VMware Aria Operations, VMware Horizon, VMware NSX, VMware HCX, and VMware Cloud on AWS infrastructure from the command line.
- Advanced administrators can develop PowerShell scripts that other administrators can reuse or integrate into other applications.

Updated Information

This section is updated with each release of the product or when necessary.

This table tracks record of the updated information in the *VMware PowerCLI User's Guide* starting from the 12.4.0 release.

PowerCLI Version / Publish Date	Description
13.1.0 / 04 AUGUST 2023	<ul style="list-style-type: none">■ Added a new chapter: Chapter 14 Managing vSphere Lifecycle Manager with VMware PowerCLI.■ Optimized Chapter 1 Introduction to VMware PowerCLI.
13.1.0 / 27 JUNE 2023	<ul style="list-style-type: none">■ Added a new chapter: Chapter 10 Managing the Site Recovery Manager (SRM) API with VMware PowerCLI.■ Added two new modules, <code>VMware.Sdk.Vr</code> and <code>VMware.Sdk.Srm</code> to PowerCLI Modules.■ Reorganized the chapter structure of the book.
13.1.0 / 22 MAY 2023	<ul style="list-style-type: none">■ Added a new chapter: Chapter 8 Managing the vSphere Replication API with VMware PowerCLI.
13.0.0 / 22 NOV 2022	<ul style="list-style-type: none">■ Added a new chapter: Installing and Configuring Python for PowerCLI. Starting from PowerCLI 13.0, Python is a requirement for the <code>VMware.ImageBuilder</code> module.
12.7.0 / 07 NOV 2022	<ul style="list-style-type: none">■ Added a new chapter: Chapter 11 Managing the NSX Policy API with VMware PowerCLI.
12.7.0 / 08 AUG 2022	<ul style="list-style-type: none">■ Added information on how to connect to a vCenter Server that is federated to Active Directory Federation Services (AD FS). See Connect to an AD FS - Federated vCenter Server System.
12.6.0 / 07 APR 2022	<ul style="list-style-type: none">■ Removed non-inclusive language terms from the <i>PowerCLI User's Guide</i> and replaced them with appropriate alternatives.■ Added a new module, <code>VMware.Sdk.Nsx.Policy</code>, to PowerCLI Modules.
12.5.0 / 13 JAN 2022	<ul style="list-style-type: none">■ Added information on how to manage the certificates of your vSphere environment with VMware PowerCLI. See the new chapter Managing Certificates.
12.4.0 / 28 SEP 2021	<ul style="list-style-type: none">■ Added an Updated Information topic.■ Updated the procedure for uninstalling PowerCLI from your system. See Uninstall PowerCLI.
12.4.0 / 16 SEP 2021	<ul style="list-style-type: none">■ Added information about the new VMware PowerCLI modules for the 12.4.0 release. See PowerCLI Modules.■ Updated the chapter on managing VMware Cloud on AWS with PowerCLI and added many new scripts. See Chapter 16 Managing VMware Cloud on AWS with VMware PowerCLI.■ Added information, procedures, and sample scripts for the vSphere Automation SDK for PowerShell, which was introduced in VMware PowerCLI 12.4.0. See Chapter 6 Managing the vSphere Automation API with VMware PowerCLI.

Introduction to VMware PowerCLI

1

VMware PowerCLI contains modules of cmdlets based on Microsoft PowerShell for automating vSphere, VMware Site Recovery Manager (SRM), vSphere Automation API, vSAN, VMware Cloud Director, vSphere Update Manager, VMware Aria Operations, VMware Horizon, VMware NSX, VMware HCX, and VMware Cloud on AWS administration. VMware PowerCLI provides a PowerShell interface to the VMware product APIs.

- [Microsoft PowerShell Basics](#)

PowerCLI is based on Microsoft PowerShell and uses the PowerShell basic syntax and concepts.

- [PowerCLI Concepts](#)

PowerCLI cmdlets are created to automate VMware environments administration and to introduce some specific features in addition to the PowerShell concepts.

Microsoft PowerShell Basics

PowerCLI is based on Microsoft PowerShell and uses the PowerShell basic syntax and concepts.

Microsoft PowerShell is both a command-line and scripting environment. It uses the .NET object model and provides administrators with system administration and automation capabilities. To work with PowerShell, you run commands, named cmdlets.

What to read next

- [PowerShell Command-Line Syntax](#)

PowerShell cmdlets use a consistent verb-noun structure, where the verb represents the action and the noun represents the object to operate on.

- [PowerShell Pipelines](#)

A pipeline is a series of commands separated by the pipe operator |.

- [PowerShell Wildcards](#)

PowerShell has a number of pattern-matching operators named wildcards that you can use to substitute one or more characters in a string, or substitute the complete string.

■ PowerShell Common Parameters

The PowerShell engine retains a set of parameter names, referred to as common parameters. All PowerShell cmdlets, including the PowerCLI cmdlets, support them.

PowerShell Command-Line Syntax

PowerShell cmdlets use a consistent verb-noun structure, where the verb represents the action and the noun represents the object to operate on.

PowerShell cmdlets follow consistent naming patterns, ensuring that construction of a command is easy if you know the object that you want to work with.

All command categories take parameters and arguments. A parameter starts with a hyphen and is used to control the behavior of the command. An argument is a data value consumed by the command.

A simple PowerShell command has the following syntax:

```
command -parameter1 -parameter2 argument1, argument2
```

PowerShell Pipelines

A pipeline is a series of commands separated by the pipe operator |.

Each command in the pipeline receives an object from the previous command, performs some operation on it, and then passes it to the next command in the pipeline. Objects are output from the pipeline as soon as they become available.

PowerShell Wildcards

PowerShell has a number of pattern-matching operators named wildcards that you can use to substitute one or more characters in a string, or substitute the complete string.

All wildcard expressions can be used with the PowerCLI cmdlets. For example, you can view a list of all files with a .txt extension by running `dir *.txt`. In this case, the asterisk * operator matches any combination of characters.

With wildcard patterns you can indicate character ranges as well. For example, to view all files that start with the letter S or T and have a .txt extension, you can run `dir [st]*.txt`.

You can use the question mark ? wildcard to match any single character within a sequence of characters. For example, to view all .txt files with names that consist of `string` and one more character at the end, run `dir string?.txt`.

PowerShell Common Parameters

The PowerShell engine retains a set of parameter names, referred to as common parameters. All PowerShell cmdlets, including the PowerCLI cmdlets, support them.

Some of the PowerShell common parameters are `Verbose`, `Debug`, `ErrorAction`, `ErrorVariable`, `OutVariable`, and `OutBuffer`. For a full list of the common parameters and more details on their usage, run `Get-Help about_CommonParameters`.

PowerShell offers two risk mitigation parameters: `WhatIf` and `Confirm`.

WhatIf	Displays the effects of a command without running it.
Confirm	Prompts for confirmation before running a command that stops a program or service, or deletes data.

PowerCLI Concepts

PowerCLI cmdlets are created to automate VMware environments administration and to introduce some specific features in addition to the PowerShell concepts.

- [PowerCLI Modules](#)

VMware PowerCLI consists of multiple modules that you can install and use according to your needs and environments.

- [Retrieving vSphere Inventory Objects from Cloud Resources](#)

You can retrieve vSphere inventory objects from cloud resources by using the `RelatedObject` parameter of PowerCLI cmdlets. This interoperability expands your cloud administration, automation, reporting, and troubleshooting options.

- [Selecting Objects in PowerCLI](#)

You can pass strings and wildcards to all parameters that take inventory objects, datastores, `OSCustomizationSpec` objects, and `VIserver` objects as arguments. PowerCLI then transforms the strings into PowerShell objects. This approach is called Object-by-Name (OBN) selection.

- [Processing Non-alphanumeric Characters in PowerCLI](#)

When you provide login credentials in the command prompt or in a script file, a PowerShell limitation might prevent PowerCLI from processing non-alphanumeric characters correctly. To prevent login problems, escape the non-alphanumeric characters in your credentials.

- [Running PowerCLI Cmdlets Asynchronously](#)

By default, PowerCLI cmdlets return an output only after completion of the requested tasks. If you want a cmdlet to return to the command line immediately, without waiting for the tasks to complete, you can use the `RunAsync` parameter.

- [Managing Default Server Connections in PowerCLI](#)

By default, PowerCLI cmdlets run on the vCenter Server or VMware Cloud Director systems you are connected to, if no target servers can be determined from the provided parameters. Use the `$DefaultVIServers` and `$DefaultCServers` variables to manage the default server connections.

■ Customization Specification Objects in PowerCLI

You can use customization specifications to define the configuration settings and parameters for customizing virtual machines during the deployment or cloning process. PowerCLI provides two types of objects for customization specification: persistent and nonpersistent.

■ Using ESXCLI with PowerCLI

With PowerCLI you can invoke ESXCLI commands within your PowerShell session and manage your ESXi hosts.

■ Using the PowerCLI Inventory Provider

Use the PowerCLI Inventory Provider to interact with the VMware vSphere inventory through a file system interface, so that it is easier to navigate, manipulate, and automate tasks.

■ Using the PowerCLI Datastore Provider

Use the PowerCLI Datastore Provider to access the contents of one or more datastores through a file system interface, so that it is easier to navigate, manipulate, and automate tasks..

■ Using the PowerCLI About Articles

Learn about PowerCLI concepts and features from the built-in help articles named *About Articles*. You can access them through a running PowerCLI process.

PowerCLI Modules

VMware PowerCLI consists of multiple modules that you can install and use according to your needs and environments.

The following table lists all official VMware PowerCLI modules.

Module	Description
<code>VMware.PowerCLI</code>	Provides a root module which other modules are dependent on. This module ensures the PowerCLI product can be installed, upgraded, and removed as a complete package if needed.
<code>VMware.VimAutomation.Core</code>	Provides cmdlets for automated administration of the vSphere environment.
<code>VMware.VimAutomation.Common</code>	Provides help functionalities. This module has no cmdlets, but is required for other modules to function correctly.
<code>VMware.VimAutomation.Sdk</code>	Provides SDK functionality that is needed by all PowerCLI modules. This module has no cmdlets, but is required for other modules to function correctly.
<code>VMware.VimAutomation.Vds</code>	Provides cmdlets for managing vSphere distributed switches and distributed port groups.
<code>VMware.VimAutomation.Cis.Core</code>	Provides cmdlets for managing vSphere Automation API servers.

Module	Description
<code>VMware.VimAutomation.Storage</code>	Provides cmdlets for managing vSAN and vSphere policy-based storage.
<code>VMware.VimAutomation.StorageUtility</code>	Provides utility scripts for storage.
<code>VMware.VimAutomation.License</code>	Provides the <code>Get-LicenseDataManager</code> cmdlet for managing VMware License components.
<code>VMware.ImageBuilder</code>	Provides cmdlets for managing depots, image profiles, and VIBs.
<code>VMware.DeployAutomation</code>	Provides cmdlets that provide an interface to VMware Auto Deploy for provisioning physical hosts with ESXi software.
<code>VMware.VimAutomation.Cloud</code>	Provides cmdlets for automating VMware Cloud Director features.
<code>VMware.VumAutomation</code>	Provides cmdlets for automating vSphere Update Manager features.
<code>VMware.VimAutomation.vROps</code>	Provides cmdlets for automating VMware Aria Operations features.
<code>VMware.VimAutomation.Srm</code>	Provides cmdlets for managing VMware Site Recovery Manager (SRM) features.
<code>VMware.VimAutomation.HorizonView</code>	Provides cmdlets for automating VMware Horizon features.
<code>VMware.VimAutomation.Nsxt</code>	Provides cmdlets for managing VMware NSX servers.
<code>VMware.VimAutomation.Vmc</code>	Provides cmdlets for automating VMware Cloud on AWS features.
<code>VMware.Vim</code>	Provides a module that contains the vSphere low-level binding libraries.
<code>VMware.VimAutomation.Security</code>	Provides cmdlets for managing vSphere Security, including virtual Trusted Platform Module.
<code>VMware.VimAutomation.Hcx</code>	Provides cmdlets for managing VMware HCX features.
<code>VMware.VimAutomation.WorkloadManagement</code>	Provides cmdlets for managing VMware vSphere with VMware Tanzu features.
<code>VMware.CloudServices</code>	Provides cmdlets for managing VMware Cloud Services.
<code>VMware.Sdk.VSphere.*</code>	Provides auto-generated cmdlets for managing the vSphere Automation API.
<code>VMware.Sdk.Nsx.Policy</code>	Provides auto-generated cmdlets for managing the NSX Policy API.
<code>VMware.Sdk.Vr</code>	Provides auto-generated cmdlets for managing the vSphere Replication API.
<code>VMware.Sdk.Srm</code>	Provides auto-generated cmdlets for managing the Site Recovery Manager (SRM) API.

Module	Description
<code>VMware.PowerCLI.VCenter.*</code>	Provide PowerShell-based cmdlets for automated administration of the vSphere environment.
<code>VMware.PowerCLI.Sdk.*</code>	Provide help functionalities for the PowerShell-based cmdlets. These modules have no cmdlets but are required for other modules to function correctly.

Retrieving vSphere Inventory Objects from Cloud Resources

You can retrieve vSphere inventory objects from cloud resources by using the `RelatedObject` parameter of PowerCLI cmdlets. This interoperability expands your cloud administration, automation, reporting, and troubleshooting options.

You can use the `RelatedObject` parameter of PowerCLI cmdlets to retrieve vSphere inventory objects from VMware Cloud Director objects. Passing the retrieved objects to the cmdlets of the `VMware.VimAutomation.Core` and `VMware.VimAutomation.Vds` modules, extends your administration options.

Note Use of the `VMware.VimAutomation.Core` and `VMware.VimAutomation.Vds` modules to modify the configuration of objects that are managed by VMware Cloud Director might result in unpredictable behavior of the cloud environment.

Table 1-1. List of Supported vSphere Inventory Objects You Can Retrieve from Cloud Objects

Cloud Object	Retrieved vSphere Inventory Object	Sample Script for Retrieving the vSphere Inventory Object
ProviderVdc	Datastore	<pre>Get-ProviderVdc -Name 'MyProviderVdc' Get-Datastore</pre>
CIVM	VirtualMachine	<pre>Get-CIVM -Name 'MyCloudVM' Get-VM</pre>
NetworkPool	VDSwitch	<pre>Get-NetworkPool -Name 'MyNetworkPool' Get-VDSwitch</pre>
NetworkPool	VDPortgroup	<pre>Get-NetworkPool -Name 'MyNetworkPool' Get-VDPortGroup</pre>
ExternalNetwork	VDPortgroup	<pre>Get-ExternalNetwork -Name 'MyExternalNetwork' Get-VDPortGroup</pre>

Selecting Objects in PowerCLI

You can pass strings and wildcards to all parameters that take inventory objects, datastores, `OSCustomizationSpec` objects, and `VIserver` objects as arguments. PowerCLI then transforms the strings into PowerShell objects. This approach is called Object-by-Name (OBN) selection.

Using OBN

For example, if you pass `start-vm -vm "myVM"`, PowerCLI transforms the string parameter, "myVM", into a PowerShell object. This is how OBN works.

For more details about OBN, run `help about_OBN`.

Using pipelines and variables

Instead of assigning an object name to a cmdlet parameter, you can pass the object through a pipeline or a variable. For example, the following three commands are interchangeable:

- `Remove-VM -VM "Win 7 SP1"`
- `Get-VM -Name "Win 7 SP1" | Remove-VM`
- `Remove-VM -VM (Get-VM -Name "Win 7 SP1")`

Note In PowerCLI, passing strings as pipeline input is not supported.

Example: An OBN failure

If you provide a non-existing object name, an OBN failure occurs. In such cases, PowerCLI generates a non-terminating error and runs the cmdlet ignoring the invalid name.

This example illustrates the occurrence of an OBN failure.

```
Set-VM -VM "VM1", "VM2", "VM3" -Server $server1, $server2 -MemoryGB 4
```

If the *VM2* virtual machine does not exist on either of the selected servers, PowerCLI generates a non-terminating error and applies the command only on the *VM1* and *VM3* virtual machines.

Processing Non-alphanumeric Characters in PowerCLI

When you provide login credentials in the command prompt or in a script file, a PowerShell limitation might prevent PowerCLI from processing non-alphanumeric characters correctly. To prevent login problems, escape the non-alphanumeric characters in your credentials.

To escape non-alphanumeric characters in PowerCLI, you need to place the expression that contains them in single quotes (').

Note When you provide your login credentials in the Specify Credential dialog box, you do not need to escape non-alphanumeric characters.

Example: Connecting to a vCenter Server System

This example illustrates how to escape non-alphanumeric characters when connecting to a selected vCenter Server instance with the `Adminis!ra!or` user name and the `pa$$word` password.

```
Connect-VIServer -Server 10.23.112.235 -Protocol https -Username 'Adminis!ra!or' -Password 'pa$$word'
```

Running PowerCLI Cmdlets Asynchronously

By default, PowerCLI cmdlets return an output only after completion of the requested tasks. If you want a cmdlet to return to the command line immediately, without waiting for the tasks to complete, you can use the `RunAsync` parameter.

When you use the `RunAsync` parameter, the cmdlet returns `Task` objects instead of its usual output. The `Status` property of a returned `Task` object contains a snapshot of the initial state of the task. This state is not updated automatically and has the values `Error`, `Queued`, `Running`, or `Success`. You can refresh a task state by retrieving the task object with the `Get-Task` cmdlet. If you want to observe the progress of a running task and wait for its completion before running other commands, use the `Wait-Task` cmdlet.

Note In PowerCLI, the `RunAsync` parameter affects only the invocation of a cmdlet and does not control whether the initiated tasks run consecutively or in parallel. For example, the `Remove-VM` cmdlet might remove the selected virtual machines simultaneously or consecutively depending on the internal design of PowerCLI. To make sure that tasks initiated by a cmdlet run consecutively, run the cmdlet in a loop, each time applying it to a single object.

Example: Running Remove-VM with and without the RunAsync parameter

```
Remove-VM $vmList
```

The command returns no output when all virtual machines stored in the `$vmList` variable are removed, irrespective of whether they are removed simultaneously.

```
Remove-VM $vmList -RunAsync
```

The command returns an output that consists of one or more `Task` objects immediately.

Managing Default Server Connections in PowerCLI

By default, PowerCLI cmdlets run on the vCenter Server or VMware Cloud Director systems you are connected to, if no target servers can be determined from the provided parameters. Use the `$DefaultVIServers` and `$DefaultCIServers` variables to manage the default server connections.

When you connect to a vCenter Server system by using `Connect-VIServer`, the server connection is stored in the `$DefaultVIServers` array variable. This variable contains all connected servers for the current PowerCLI session. To remove a server from the `$DefaultVIServers` variable, you can either use `Disconnect-VIServer` to close all active connections to this server, or modify the value of `$DefaultVIServers` manually.

When you connect to a VMware Cloud Director system by using `Connect-CIServer`, the server connection is stored in the `$DefaultCIServers` array variable. This variable contains all connected servers for the current session. To remove a server from the `$DefaultCIServers` variable, you can either use `Disconnect-CIServer` to close all active connections to this server, or modify the value of `$DefaultCIServers` manually.

Customization Specification Objects in PowerCLI

You can use customization specifications to define the configuration settings and parameters for customizing virtual machines during the deployment or cloning process. PowerCLI provides two types of objects for customization specification: persistent and nonpersistent.

Persistent Customization

Persistent customization specification objects are stored on the vSphere server. All persistent customization specifications created by using vSphere Client or VMware PowerCLI 4.1 or later are encrypted. Encrypted customization specifications can be applied only by the server that has encrypted them.

Nonpersistent Customization

Nonpersistent customization specification objects exist only inside the current PowerShell process. Nonpersistent customization specification objects are not encrypted, but cloning them to a vSphere server encrypts them.

Using ESXCLI with PowerCLI

With PowerCLI you can invoke ESXCLI commands within your PowerShell session and manage your ESXi hosts.

You can use PowerCLI to access ESXCLI in the following ways:

By Using the `Get-ESXcli` cmdlet

Use the `Get-ESXcli` cmdlet to invoke ESXCLI commands within your PowerShell console.

For example:

```
$vmHost = Get-VMHost <vmHostIp>
$esxcli = Get-EsxCli -VMHost $vmHost -V2
$esxcli.storage.nmp.device.list.Invoke()
```

For more information, see [Get-EsxCli](#) in the PowerCLI Reference.

By Using .NET Methods

You can use .NET methods to create managed objects that correspond to specific ESXCLI applications. To access the ESXCLI, you can call methods on these managed objects.

Note To call a method of an ESXCLI object, you must provide values for all parameters. If you want to omit a given parameter, pass *\$null* as its argument.

Using the PowerCLI Inventory Provider

Use the PowerCLI Inventory Provider to interact with the VMware vSphere inventory through a file system interface, so that it is easier to navigate, manipulate, and automate tasks.

The PowerCLI Inventory Provider allows navigation and file-style management of the VMware vSphere inventory. By creating a PowerShell drive based on a managed object (such as a data center), you can obtain a view of its contents and the relationships between the items. In addition, you can move, rename, or delete objects by running commands from the PowerShell console.

When you connect to a server with `Connect-VIServer`, the cmdlet builds two default inventory drives: `vi` and `vis`.

To access the inventory of the last connected server, run: `cd vi:\`.

To access the inventory of all vSphere servers connected within the current PowerCLI session, run `cd vis:\`.

Use the `dir` and `cd` commands to navigate the inventories.

Using the PowerCLI Datastore Provider

Use the PowerCLI Datastore Provider to access the contents of one or more datastores through a file system interface, so that it is easier to navigate, manipulate, and automate tasks..

The items in a datastore are files that contain configuration, virtual disk, and other data associated with a virtual machine.

When you connect to a server with `Connect-VIServer`, the cmdlet builds two default datastore drives: `vmstore` and `vmstores`.

To access the datastores of the last connected vCenter Server or ESXi, run: `cd vmstore:\`.

To access the datastores available on all vCenter Server and ESXi systems you are currently connected to, run: `cd vmstores:\`.

Use the `dir` and `cd` commands to navigate the inventories.

Using the PowerCLI About Articles

Learn about PowerCLI concepts and features from the built-in help articles named *About Articles*. You can access them through a running PowerCLI process.

Running `Help About_*` lists all built-in PowerShell and VMware PowerCLI *About Articles*.

Table 1-2. Built-In *About Articles* for PowerCLI

Article Title	Command	Article Description
Customer Experience Improvement Program (CEIP)	<code>Help About_CEIP</code>	Provides information about VMware's Customer Experience Improvement Program ("CEIP").
Handling Invalid Certificates	<code>Help About_Invalid_Certificates</code>	When you connect to a server, VMware PowerCLI checks if the server certificate is valid. If the certificate is not trusted for this server, by default, VMware PowerCLI fails to connect to the server.
Object-by-Name (OBN)	<code>Help About_OBN</code>	To help you save time and effort, PowerCLI lets you select objects by their names.
VMware PowerCLI Objects	<code>Help About_PowerCLI_Objects</code>	For their input and output, the PowerCLI cmdlets use a set of .NET types that reside in the <code>VMware.VimAutomation.ViCore.Types</code> namespace.
Using the RunAsync Parameter	<code>Help About_RunAsync</code>	When you set the <code>RunAsync</code> parameter, you indicate that you want to run the cmdlet asynchronously.
Authenticating with a vCenter Server System or a VMware Cloud Director Server	<code>Help About_Server_Authentication</code>	To authenticate with vCenter Server and VMware Cloud Director servers, you can provide a user name and password through the <code>User</code> and <code>Password</code> parameters, or a <code>PSCredential</code> object through the <code>Credential</code> parameter.
Unique Identifiers for PowerCLI Objects (UID)	<code>Help About_UID</code>	You can uniquely identify a PowerCLI object on a server or across multiple servers by providing its UID.
Datastore Provider (VimDatastore)	<code>Help About_VimDatastore</code>	The Datastore Provider (<code>VimDatastore</code>) provides a filesystem-style view and access to the contents of datastores.
LicenseDataManager	<code>Help About_LicenseDataManager</code>	The <code>LicenseDataManager</code> component lets you extend the vCenter Server inventory with license data.

Installing VMware PowerCLI

2

VMware PowerCLI lets you manage, monitor, automate, and handle lifecycle operations on VMware vSphere, VMware Aria Operations, vSAN, VMware Cloud Director, vSphere Update Manager, VMware NSX, VMware Cloud on AWS, VMware HCX, VMware Horizon, and VMware Site Recovery Manager (SRM) systems. You can install VMware PowerCLI modules on all supported operating systems.

After installing the package on your machine, you can connect to your ESXi, vCenter Server, VMware Aria Operations, VMware Cloud Director, VMware NSX, VMware Cloud on AWS, VMware Horizon, or VMware Site Recovery Manager (SRM) system by providing valid authentication credentials.

- **Supported Operating Systems**

You can install PowerCLI on supported operating systems. You can run guest cmdlets against virtual machines on which supported guest operating systems are installed.

- **Supported VMware Products**

You can use the PowerCLI modules to manage all supported VMware products.

- **Supported PowerShell Versions**

PowerCLI is compatible with multiple versions of PowerShell.

- **Install PowerCLI**

You can download and install PowerCLI modules from the PowerShell Gallery. You can install all VMware PowerCLI modules with a single PowerShell command, or install modules individually.

- **Install PowerCLI Offline**

You can install all VMware PowerCLI modules in offline mode by using a ZIP file.

- **Update PowerCLI**

You can update PowerCLI when a new version of the product becomes available. See the PowerShell Gallery for the latest PowerCLI versions.

- **Uninstall PowerCLI**

You can uninstall VMware PowerCLI by deleting the VMware PowerCLI modules from your system.

Supported Operating Systems

You can install PowerCLI on supported operating systems. You can run guest cmdlets against virtual machines on which supported guest operating systems are installed.

PowerCLI Local Operating Systems

For a list of operating systems on which you can install VMware PowerCLI , see the Compatibility Matrix on the [PowerCLI home page](#).

PowerCLI Guest Operating Systems

You can run VMware PowerCLI guest cmdlets against virtual machines with supported guest operating systems. For a list of supported operating systems, see the Compatibility Matrix on the [PowerCLI home page](#).

Note Guest cmdlets are not compatible with IPv6 environments.

Supported VMware Products

You can use the PowerCLI modules to manage all supported VMware products.

For a list of VMware products with which VMware PowerCLI is compatible, see [VMware Product Interoperability Matrixes](#).

Supported PowerShell Versions

PowerCLI is compatible with multiple versions of PowerShell.

For a list of PowerShell versions with which VMware PowerCLI is compatible, see the Compatibility Matrix on the [PowerCLI home page](#).

Install PowerCLI

You can download and install PowerCLI modules from the PowerShell Gallery. You can install all VMware PowerCLI modules with a single PowerShell command, or install modules individually.

Use the `Install-Module` cmdlet to download and install PowerCLI modules. See the PowerShell Gallery for the complete list of available PowerCLI modules.

Prerequisites

- Verify that your system is compatible with PowerCLI. See the Compatibility Matrix on the [PowerCLI home page](#).
- Verify that your system is connected to the Internet.
- Verify that PowerShell is available on your system. For Linux and macOS, you must install PowerShell. See [how to install PowerShell on different platforms](#).

- For Windows, if you have PowerCLI 6.5 R1 or earlier, uninstall it.

Procedure

- 1 Open PowerShell on your workstation.
- 2 To install all PowerCLI modules, run the command:

```
Install-Module VMware.PowerCLI -Scope CurrentUser
```

Alternatively, you can install individual PowerCLI modules by running the `Install-Module` cmdlet with the module name.

- 3 If you see a warning that you are installing modules from an untrusted repository, press **y** and then press Enter to confirm the installation.

What to do next

Enable execution of local scripts. See [Allow Execution of Local Scripts](#).

Install PowerCLI Offline

You can install all VMware PowerCLI modules in offline mode by using a ZIP file.

You might need to install PowerCLI on a local machine with no Internet connectivity due to security reasons and deployment restrictions. If you are using such an environment, you can download the PowerCLI ZIP file on a computer with Internet access, transfer the ZIP file to your local machine and install PowerCLI.

Prerequisites

- Verify that your system is compatible with PowerCLI. See the [Compatibility Matrix](#) on the [PowerCLI home page](#).
- Verify that PowerShell is available on your system. For Linux and macOS, you must install PowerShell. See [how to install PowerShell on different platforms](#).
- For Windows, if you have PowerCLI 6.5 R1 or earlier, uninstall it.
- Download the PowerCLI ZIP file from the [PowerCLI home page](#) and transfer the ZIP file to your local machine.

Procedure

- 1 Open PowerShell on your local machine.
- 2 To view the folder paths to which you can extract the PowerCLI ZIP file, run the command:

```
$env:PSModulePath
```

- 3 Extract the contents of the PowerCLI ZIP file to one of the listed folders.

- 4 For Windows, run the command to unblock the copied files.

```
Get-ChildItem -Path 'folder_path' -Recurse | Unblock-File
```

Replace *folder_path* with the path to the folder where you extracted the contents of the ZIP file.

- 5 (Optional) Verify that the VMware PowerCLI modules have installed successfully.

```
Get-Module VMware* -ListAvailable
```

Results

You can now run PowerCLI on your local machine.

What to do next

Enable execution of local scripts. See [Allow Execution of Local Scripts](#).

Update PowerCLI

You can update PowerCLI when a new version of the product becomes available. See the PowerShell Gallery for the latest PowerCLI versions.

Use the `Update-Module` cmdlet to update PowerCLI modules. You can update all VMware PowerCLI modules with a single command, or update modules individually.

You cannot use the `Update-Module` cmdlet in offline mode. To update PowerCLI in offline mode, you must perform a new installation. See [Install PowerCLI Offline](#).

Procedure

- 1 Open PowerShell.
- 2 To update all PowerCLI modules, run the command:

```
Update-Module VMware.PowerCLI -Scope CurrentUser
```

Alternatively, you can update individual PowerCLI modules by running the `Update-Module` cmdlet with the module name.

Uninstall PowerCLI

You can uninstall VMware PowerCLI by deleting the VMware PowerCLI modules from your system.

To uninstall VMware PowerCLI, you must locate your VMware PowerCLI modules and delete them.

Prerequisites

- Close all PowerShell sessions that are running PowerCLI.

Procedure

- 1 In PowerShell, to see the location of the `VMware.PowerCLI` modules, run `Get-Module -ListAvailable VMware*`.
- 2 In your operating system, navigate to the destination folder(s) and delete all modules that start with `VMware`.

Results

You removed PowerCLI from your system.

Configuring VMware PowerCLI

3

Configure how VMware PowerCLI responds to invalid server certificates, choose the application settings for different users and user groups, modify the script configuration file, and add custom scripts.

Read the following topics next:

- [Allow Execution of Local Scripts](#)
- [Configuring PowerCLI Response to Untrusted Certificates](#)
- [Scoped Settings of PowerCLI](#)
- [Installing and Configuring Python for PowerCLI](#)

Allow Execution of Local Scripts

If you want to run scripts and load configuration files with PowerCLI, you must set the execution policy of PowerShell to `RemoteSigned`.

For security reasons, PowerShell supports an execution policy feature. It determines whether scripts are allowed to run and whether they must be digitally signed. By default, the execution policy is set to `Restricted`, which is the most secure policy. For more information about the execution policy and script digital signing in PowerShell, run `Get-Help About_Signing`.

You can change the execution policy by using the `Set-ExecutionPolicy` cmdlet.

Procedure

- 1 Open the PowerShell console.
- 2 Run `Set-ExecutionPolicy RemoteSigned`.

Configuring PowerCLI Response to Untrusted Certificates

When you try to connect to a server, PowerCLI checks whether the server certificate is valid. You might have to configure the PowerCLI settings to be able to connect to servers with untrusted certificates.

You might not be able to connect to a server because of the default action configured for PowerCLI and untrusted certificates.

By default, PowerCLI is set to deny connections to servers with invalid certificates. You can retrieve the current setting for invalid server certificates, and configure it as required. For more information, see [Configure the PowerCLI Response to Untrusted Certificates](#).

Configure the PowerCLI Response to Untrusted Certificates

You can configure how PowerCLI responds when a server certificate is not valid.

Use the `InvalidCertificateAction` parameter of the `Set-PowerCLIConfiguration` cmdlet to configure how PowerCLI responds to invalid server certificates.

Procedure

- 1 Run `Get-PowerCLIConfiguration` and view the current setting of the `InvalidCertificateAction` parameter.
- 2 To change the setting, run `Set-PowerCLIConfiguration -InvalidCertificateAction` and specify one of the following values.
 - `Unset`. This is the default value and corresponds to `Fail`.
 - `Fail`. PowerCLI does not establish a connection if a certificate is not valid.
 - `Ignore`. PowerCLI ignores certificate validity and establishes a connection.
 - `Warn`. PowerCLI logs a warning message that a certificate is not valid, displays additional information about the certificate, and establishes the connection.
 - `Prompt`. If a server certificate is not valid, PowerCLI prompts you to choose an action before it continues. You can choose one of the following options.

Option	Action
Deny (default)	Cancel the server connection.
Accept Once	Establish the server connection and suppress further warnings for the current PowerShell session.
Accept Permanently	Save the server certificate in the PowerCLI Trusted Certificate Store for the current user and establish the server connection.
Accept For All Users	Save the server certificate both in the current user's Trusted Certificate Store and in the all users' Trusted Certificate Store and establish the server connection.

Note For Linux and macOS, only the `Fail` and `Ignore` options are supported.

A warning message prompts you to confirm the operation. Press `Y` and then press `Enter` to confirm your choice.

Results

You configured the invalid server certificate action setting. PowerCLI displays a table with the updated configuration settings.

Trusted certificates are saved in the [PowerCLI Trusted Certificate Store](#).

What to do next

Use PowerCLI to connect to a server system.

PowerCLI Trusted Certificate Store

The PowerCLI trusted certificate store is a CSV file that stores trusted server certificates for the current user or for all users.

The PowerCLI trusted certificate store is named `SslCertificateExceptions.csv` and contains server addresses and certificate thumbprints. You can edit the file with a text editor. You can transfer the CSV file to another machine if you want to use your saved certificates.

Note You must permanently accept at least one certificate as trusted for the CSV file to appear on your system.

The certificate store file is saved in a different location depending on the type of user. For more information about the exact file path of the trusted certificate store file, see [PowerCLI Configuration Files](#).

Scoped Settings of PowerCLI

In PowerCLI you can set the scope of the settings to enhance security and personalize the configuration.

What to read next

- [Configuring the Scope of the PowerCLI Settings](#)
Scoped configuration enhances system security and prevents nonadministrator users from introducing global changes to the configuration of PowerCLI.
- [Priority of Settings Scopes in PowerCLI](#)
PowerCLI loads the program configuration based on the scope that you select for each setting.
- [PowerCLI Configuration Files](#)
The copies of the `PowerCLI_settings.xml` file on your system contain `User` and `AllUsers` settings for PowerCLI.

Configuring the Scope of the PowerCLI Settings

Scoped configuration enhances system security and prevents nonadministrator users from introducing global changes to the configuration of PowerCLI.

For greater control over the PowerCLI configuration, the `Set-PowerCLIConfiguration` cmdlet provides the `Scope` parameter.

Table 3-1. Valid Values for the Scope Parameter

Parameter Value	Description
Session	Configures settings for the current PowerCLI session and does not modify any PowerCLI configuration files on your system.
User	Configures settings for the current Windows user and modifies some PowerCLI configuration files on your system.
AllUsers	Configures settings for all users and modifies some PowerCLI configuration files on your system.

Priority of Settings Scopes in PowerCLI

PowerCLI loads the program configuration based on the scope that you select for each setting.

Table 3-2. Scope Impact on the Behavior of PowerCLI

Scope	Priority	Impact
Session	High	<ul style="list-style-type: none"> ■ When started, PowerCLI tries to load settings with the <code>Session</code> scope first. ■ <code>Session</code> settings override <code>User</code> and <code>AllUsers</code> settings. ■ <code>Session</code> settings are valid for the current PowerCLI session only.
User	Medium	<ul style="list-style-type: none"> ■ When PowerCLI cannot detect <code>Session</code> settings, the program tries to load <code>User</code> settings from the PowerCLI configuration files. ■ <code>User</code> settings override <code>AllUsers</code> settings. ■ <code>User</code> settings are automatically detected from the PowerCLI configuration files.
AllUsers	Low	<ul style="list-style-type: none"> ■ When PowerCLI cannot detect <code>Session</code> and <code>User</code> settings, the program loads <code>AllUsers</code> settings. ■ <code>AllUsers</code> settings do not override <code>Session</code> and <code>User</code> settings. ■ <code>AllUsers</code> settings are automatically detected from the PowerCLI configuration files.

PowerCLI Configuration Files

The copies of the `PowerCLI_settings.xml` file on your system contain `User` and `AllUsers` settings for PowerCLI.

Configuring PowerCLI by running `Set-PowerCLIConfiguration` creates a copy of `PowerCLI_settings.xml` on your system. The location of the `PowerCLI_settings.xml` file depends on the value of the `Scope` parameter.

Note You must have administrator privileges to change the settings for `AllUsers`.

Table 3-3. Location of `PowerCLI_settings.xml`

Operating System	Scope	Location	Description
Windows	User	%APPDATA%\VMware\PowerCLI	Contains settings for the current Windows user.
	AllUsers	%SYSTEMDRIVE%\ProgramData\VMware\PowerCLI	Contains settings for all users.
Linux	User	\$HOME/.local/share/VMware/PowerCLI	Contains settings for the current Linux user.
	AllUsers	/var/opt/VMware/PowerCLI	Contains settings for all users.
macOS	User	\$HOME/Library/Preferences/VMware/PowerCLI	Contains settings for the current macOS user.
	AllUsers	/Library/Preferences/VMware/PowerCLI	Contains settings for all users.

Users with advanced knowledge and understanding of PowerShell and VMware PowerCLI can manually modify the contents of `PowerCLI_settings.xml` to change PowerCLI settings. Modifying `PowerCLI_settings.xml` might require administrator privileges.

Note If you modify the contents of `PowerCLI_settings.xml` manually while PowerCLI is running, you must restart PowerCLI for the changes to take effect.

Installing and Configuring Python for PowerCLI

Starting from VMware PowerCLI 13.0, Python 3.7 is a prerequisite for using the `VMware.ImageBuilder` module.

To use the `VMware.ImageBuilder` module, you must install Python 3.7 with four additional software packages and configure PowerCLI.

You must install the following additional packages in Python:

- `six`
- `psutil`
- `lxml`
- `pyopenssl`

You must configure PowerCLI by setting the path to the Python 3.7 installation.

Restart your PowerShell session if you tried to execute a cmdlet from the `VMware.ImageBuilder` module before setting the path to the Python executable.

Install and Configure Python on Windows

On Windows, you must install Python 3.7 with the required packages and configure PowerCLI to use the `VMware.ImageBuilder` module.

Procedure

- 1 Download Python 3.7 from <https://www.python.org/downloads/release/python-370/> and complete the installation on your machine.
- 2 Save the `get-pip.py` file from <https://bootstrap.pypa.io/get-pip.py>.
- 3 Open a terminal and install `pip` package manager.

```
<python3.7-directory>\python.exe <get-pip-directory>\get-pip.py
```

- 4 In the terminal, install the required Python modules.

```
<python3.7-directory>\Scripts\pip3.7.exe install six psutil lxml pyopenssl
```

- 5 Open PowerShell and set the path to the Python 3.7 executable.

```
Set-PowerCLIConfiguration -PythonPath <python3.7-directory>\python.exe -Scope User
```

Results

You can now use the PowerCLI `VMware.ImageBuilder` module.

Install and Configure Python on macOS

On macOS, you must install Python 3.7 with the required packages and configure PowerCLI to use the `VMware.ImageBuilder` module.

Prerequisites

Verify that you have OpenSSL version 1.1.1 installed on your machine.

Procedure

- 1 Install Python.

Option	Action
By Using the Mac Terminal	Open the Mac Terminal and run <code>brew install python@3.7</code> .
By Using the Python Installer	Download the Python installer from https://www.python.org/downloads/release/python-370/ and complete the installation on your machine.

- 2 Save the `get-pip.py` file from <https://bootstrap.pypa.io/get-pip.py>.

- 3 In the Mac Terminal, install `pip` package manager.

```
<python3.7-directory>/python3.7 <get-pip-directory>/get-pip.py
```

- 4 Install the required Python modules.

```
<python3.7-directory>/Scripts/pip3.7 install six psutil lxml pyopenssl
```

- 5 Open PowerShell and set the path to the Python 3.7 executable.

```
Set-PowerCLIConfiguration -PythonPath <python3.7-directory>/python3.7 -Scope User
```

Results

You can now use the PowerCLI `VMware.ImageBuilder` module.

Install and Configure Python on Linux

On Linux, you must install Python 3.7 with the required packages and configure PowerCLI to use the `VMware.ImageBuilder` module.

Prerequisites

Verify that you have OpenSSL version 1.1.1 installed on your machine.

Procedure

- 1 Open a terminal and install Python by running `apt-get install python3.7`.
- 2 Save the `get-pip.py` file from <https://bootstrap.pypa.io/get-pip.py>.
- 3 In the terminal, install `pip` package manager.

```
<python3.7-directory>/python3.7 <get-pip-directory>/get-pip.py
```

- 4 Install the required Python modules.

```
<python3.7-directory>/Scripts/pip3.7 install six psutil lxml pyopenssl
```

- 5 Open PowerShell and set the path to the Python 3.7 executable.

```
Set-PowerCLIConfiguration -PythonPath <python3.7-directory>/python3.7 -Scope User
```

Results

You can now use the PowerCLI `VMware.ImageBuilder` module.

Configuring Customer Experience Improvement Program

4

When you choose to participate in the Customer Experience Improvement Program (CEIP), VMware receives anonymous information to improve the quality, reliability, and functionality of VMware products and services.

Read the following topics next:

- [Categories of Information That VMware Receives](#)
- [Join the Customer Experience Improvement Program in PowerCLI](#)

Categories of Information That VMware Receives

This product participates in VMware's Customer Experience Improvement Program ("CEIP").

Details regarding the data collected through CEIP and the purposes for which it is used by VMware are set forth at the Trust & Assurance Center at <http://www.vmware.com/trustvmware/ceip.html>. To join or leave the CEIP for this product, see [Join the Customer Experience Improvement Program in PowerCLI](#).

Join the Customer Experience Improvement Program in PowerCLI

You can choose to join the Customer Experience Improvement Program (CEIP), or leave the CEIP at any time.

Procedure

- ◆ Run `Set-PowerCLIConfiguration`.
 - To join the CEIP, run the following command.

```
Set-PowerCLIConfiguration -ParticipateInCeip $true
```

- To leave the CEIP, run the following command.

```
Set-PowerCLIConfiguration -ParticipateInCeip $false
```

Managing vSphere with VMware PowerCLI

5

To help you get started with VMware PowerCLI, this documentation provides a set of sample scripts that illustrate basic and advanced tasks in vSphere administration.

- [Connect to a vCenter Server System](#)

To run PowerCLI cmdlets on vSphere and perform administration or monitoring tasks, you must establish a connection to an ESXi host or a vCenter Server system.

- [Connect to an AD FS - Federated vCenter Server System](#)

If your vCenter Server is federated to Active Directory Federation Services (AD FS), you can authenticate with PowerCLI by using the OAuth 2.0 Authorization Code grant type.

- [Manage Virtual Machines on vSphere](#)

With PowerCLI, you can automate various administration tasks on virtual machines, for example retrieving information, shutting down and powering off virtual machines.

- [Add a Standalone Host to a vCenter Server System](#)

You can add standalone hosts to a vCenter Server system by using the `Add-VMHost` cmdlet. After adding the hosts, you will be able to manage them through the vCenter Server system.

- [Set the License Key for a Host on vCenter Server](#)

You can set the license key for a host on a vCenter Server system by using the `LicenseKey` parameter of the `Set-VMHost` cmdlet.

- [Activate Maintenance Mode for a Host on vCenter Server](#)

To complete some specific administration tasks, you might need to activate maintenance mode for a host. On vCenter Server, you can activate maintenance mode by using the `Set-VMHost` cmdlet.

- [Create vSphere Inventory Objects](#)

By using PowerCLI cmdlets, you can automate creating different inventory objects on vSphere.

- [Create Virtual Machines on vCenter Server Using an XML Specification File](#)

You can use a specification provided in an XML file to automate the creation of virtual machines on vCenter Server.

- [Manage Virtual Machine Templates on vCenter Server](#)

You can use PowerCLI to create virtual machines templates and convert them to virtual machines on vCenter Server.

- [Create and Use Snapshots on vCenter Server](#)

You can use the `Snapshot` parameter of `Get-VM` to take a snapshot of virtual machines and then revert the states of the virtual machines back to the snapshot.

- [Update the Resource Configuration Settings of a Virtual Machine on vCenter Server](#)

You can use the `Set-VMResourceConfiguration` cmdlet to modify the resource configuration properties of a virtual machine, including memory, CPU shares, and other settings.

- [Get a List of Hosts on a vCenter Server System and View Their Properties](#)

With PowerCLI, you can get information about all available hosts in a data center and view their properties.

- [Change the Host Advanced Configuration Settings on vCenter Server](#)

You can modify host configuration, including advanced settings related to virtual machine migration, and apply them to another host.

- [Move a Virtual Machine to a Different Host Using VMware vSphere vMotion](#)

You can migrate a virtual machine between vCenter Server hosts by using vSphere vMotion.

- [Move a Virtual Machine to a Different Datastore Using VMware vSphere Storage vMotion](#)

You can migrate a virtual machine between datastores using the VMware Storage vMotion feature of vCenter Server.

- [Move a Virtual Machine to a Different vCenter Server System](#)

You can migrate a virtual machine from one vCenter Server system to another by using Cross vCenter Server vMotion.

- [Create a Host Profile on a vCenter Server System](#)

The VMware Host Profiles feature enables you to create standard configurations for ESXi hosts. With PowerCLI, you can automate creation and modifying of host profiles.

- [Apply a Host Profile to a Host on vCenter Server](#)

To simplify operational management of large-scale environments, you can apply standard configurations called host profiles to hosts on vCenter Server. If you want to set up a host to use the same host profile as a reference host, you can attach the host to a profile.

- [Manage Statistics and Statistics Intervals on vCenter Server](#)

You can use the PowerCLI cmdlets to automate tasks for viewing and managing statistics for vCenter Server inventory objects.

- [Modify the Settings of the NIC Teaming Policy for a Virtual Switch](#)

You can set the NIC teaming policy on a vSwitch. The NIC teaming policy determines the load balancing and failover settings of a virtual switch and lets you mark NICs as unused.

- [Create a vApp on vCenter Server](#)

With PowerCLI, you can create and manage vApps.

- [Modify the Properties of a vApp](#)

With PowerCLI, you can start and stop vApps, and modify their properties.

- [Export or Import vApps](#)

You can import and export vApps to OVA and OVF files.

- [Create an iSCSI Host Storage](#)

For a host, you can enable iSCSI, add iSCSI targets, and create new host storages.

- [Add Passthrough Devices to a Host and Virtual Machine](#)

You can get information about existing passthrough devices and add new SCSI and PCI devices to virtual machines and hosts.

- [Create a Custom Property Based on an Extension Data Property](#)

You can create custom properties to add more information to vSphere objects. Custom properties based on extension data properties correspond directly to the property of the corresponding .NET view object.

- [Create a Script-Based Custom Property for a vSphere Object](#)

You can create a custom property by writing a script and providing a name for the property. The script evaluates when the custom property is called for the first time.

- [Apply a Customization Object to a Cloned Virtual Machine](#)

You can apply a custom configuration to a cloned virtual machine by using a customization object.

- [Modify the Default NIC Mapping Object of a Customization Specification](#)

You can modify the default NIC mapping object of a customization specification and apply the specification on a newly created virtual machine.

- [Modify Multiple NIC Mapping Objects of a Customization Specification](#)

You can modify multiple NIC mapping objects of a customization specification and apply the specification to an existing virtual machine.

- [Create Multiple Virtual Machines that Use Static IP Addresses](#)

You can deploy multiple virtual machines with a single network adapter and configure the deployed virtual machines to use static IP addresses by applying a customization specification.

- [Create Multiple Virtual Machines with Two Network Adapters](#)

You can deploy multiple virtual machines with two network adapters each and configure each adapter to use specific network settings by applying a customization specification.

- [Create a vSphere Role and Assign Permissions to a User](#)

With PowerCLI, you can automate management of vSphere permissions, roles, and privileges.

- [View Actions and Triggers for an Alarm on vCenter Server](#)
You can see which actions and triggers are configured for an alarm.
- [Create Alarm Definitions on vCenter Server](#)
With PowerCLI, you can create vCenter Server alarm definitions.
- [Modify Alarm Definitions on vCenter Server](#)
With PowerCLI, you can modify vCenter Server alarm definitions.
- [Create Alarm Actions and Triggers on vCenter Server](#)
With PowerCLI, you can create vCenter Server alarm actions and triggers.
- [Remove Alarm Definitions and Actions](#)
In some cases, you might want to remove obsolete alarm definitions and actions.
- [Create and Modify Advanced Settings for a Cluster](#)
You can customize the behavior of a cluster on a vCenter Server system by creating and modifying custom advanced settings for it.
- [Modify the vCenter Server Email Configuration](#)
You can modify the email configuration settings of a vCenter Server.
- [Modify the vCenter Server SNMP Configuration](#)
To use SNMP, you must first configure the SNMP settings of the vCenter Server.
- [Use EsxTop to Get Information on the Virtual CPUs of a Virtual Machine](#)
You can use the `Get-EsxTop` cmdlet to retrieve real-time data for troubleshooting performance problems.
- [Filter vSphere Objects with Get-View](#)
You can use the `Get-View` cmdlet to filter vSphere objects before performing various actions on them.
- [Populate a View Object with Get-View](#)
To save time and efforts, you can use `Get-View` to retrieve PowerCLI views from previously retrieved view objects.
- [Update the State of a Server-Side Object](#)
You can use the `Get-View` cmdlet to update server-side objects.
- [Reboot a Host with Get-View](#)
You can reboot a host by using its corresponding view object.
- [Modify the CPU Levels of a Virtual Machine with Get-View and Get-VIObjectByVIView](#)
You can modify the CPU levels of a virtual machine using a combination of the `Get-View` and `Get-VIObjectByVIView` cmdlets.
- [Browse the Default Inventory Drive](#)
You can browse the default inventory drive and view its contents.

- [Create a New Custom Inventory Drive](#)

In addition to the default drive, you can create new custom inventory drives by using the `New-PSDrive` cmdlet.

- [Manage Inventory Objects Through Inventory Drives](#)

You can use the PowerCLI Inventory Provider to browse, modify, and remove inventory objects from inventory drives.

- [Browse the Default Datastore Drives](#)

You can use the PowerCLI Datastore Provider to browse the default datastore drives: `vmstore` and `vmstores`.

- [Create a New Custom Datastore Drive](#)

You can use the PowerCLI Datastore Provider to create custom datastore drives.

- [Manage Datastores Through Datastore Drives](#)

You can use the PowerCLI Datastore Provider to browse datastores from datastore drives.

- [Modify the Timeout Setting for Web Tasks](#)

To avoid unexpected timeouts, you can run `Set-PowerCLIConfiguration` to modify the PowerCLI settings for long-running Web tasks.

- [Using Tags](#)

You can assign tags to different types of objects, such as virtual machines, resource pools, datastores, and vSphere distributed switches. You can use tags to retrieve a specific group of objects.

- [Network Management with vSphere Distributed Switches](#)

The cmdlets provided in the `VMware.VimAutomation.VDS` module let you manage networking with vSphere distributed switches and port groups.

- [Create a Local Content Library](#)

With PowerCLI, you can create a local content library.

- [Create a Subscribed Content Library](#)

With PowerCLI, you can create a subscribed content library.

- [Create a Content Library Item](#)

With PowerCLI, you can create content library items.

- [Create a Virtual Machine from a Content Library Item](#)

You can deploy a virtual machine from a content library template.

- [Create a vApp from a Content Library Item](#)

You can deploy a vApp from a content library template.

- [Export Content Library Item's Files to a Local Machine](#)

With PowerCLI, you can export content library item's files to a local machine.

- [Create a New VM-VM DRS Rule](#)

You can create a VM-VM DRS affinity rule within a cluster.

- [Create a New VM-VMHost DRS Rule](#)

You can create a VM-VMHost DRS rule within a cluster after creating a VM DRS cluster group and a VMHost DRS cluster group.

- [Managing Certificates](#)

You can use VMware PowerCLI to manage certificates for both vCenter Server and ESXi no matter if you are using Hybrid, Subordinate CA, or Full Custom Mode.

Connect to a vCenter Server System

To run PowerCLI cmdlets on vSphere and perform administration or monitoring tasks, you must establish a connection to an ESXi host or a vCenter Server system.

You can have more than one connection to the same server. For more information, see [Managing Default Server Connections in PowerCLI](#).

If your login credentials contain non-alphanumeric characters, you might need to escape them. For more information, see [Processing Non-alphanumeric Characters in PowerCLI](#).

Prerequisites

- If the certificate of the server you want to connect to is not trusted, verify that your PowerCLI invalid server certificate action settings are configured properly. For more information, see [Configuring PowerCLI Response to Untrusted Certificates](#).
- If you use a proxy server for the connection, verify that it is configured properly, so that the connection is kept alive long enough for tasks to finish.

Note If you do not want to use a proxy server for the connection, run `Set-PowerCLIConfiguration -ProxyPolicy NoProxy`.

Procedure

- ◆ Run `Connect-VIServer` with the server name and valid credentials.

```
Connect-VIServer -Server vc1.example.com -Protocol http -User 'MyAdministratorUser'
-Password 'MyPassword'
```

Connect to an AD FS - Federated vCenter Server System

If your vCenter Server is federated to Active Directory Federation Services (AD FS), you can authenticate with PowerCLI by using the OAuth 2.0 Authorization Code grant type.

You can authenticate to a federated vCenter Server by creating a new OAuth security context and then exchanging it for a SAML security context. You create an OAuth security context for PowerCLI by using the `New-OAuthSecurityContext` cmdlet. One way to do this is to authenticate through the Authorization Code grant type, which is illustrated by this example. This workflow guarantees a substantial degree of security and can be used with multi-factor authentication.

Note You can use PowerCLI to authenticate with the other OAuth 2.0 grant types as well, such as the Client Credentials, Refresh Token, and Password grant types. For more information, run `Get-Help New-OAuthSecurityContext -full`.

Prerequisites

- Verify that your vCenter Server system is federated to AD FS. For more information, see *Federate vCenter Server to Active Directory Federation Services (AD FS)* in the *VMware vCenter Server Management Programming Guide*.
- Create an OAuth client for PowerCLI on the authentication server (AD FS). Configure the OAuth client to do the same token transformation as your vCenter Server system. You must configure a redirect URL according to the requirements in the procedure below. Save the Client ID and Client Secret that are generated by the authentication server. For more information, see the Microsoft documentation.

Procedure

- 1 Create an OAuth security context object for PowerCLI.

In the background, PowerCLI sends an authentication request to the authentication server, AD FS. A web browser opens that prompts the user to authorize the client application's request. On confirmation, PowerCLI interacts with the authentication server to obtain the access and ID tokens (JWT tokens) and creates a new OAuth security context.

```
$oauthSecContext = New-OAuthSecurityContext -TokenEndpointUrl "https://<AD FS FQDN>/adfs/
oauth2/token/" -AuthorizationEndpointUrl "https://<AD FS FQDN>/adfs/oauth2/authorize/"
-RedirectUrl "http://localhost:8844/auth" -ClientId "powercli-native" -OtherArguments
@{ "resource" = "my-vcenter" }
```

This script contains the following parameters.

Parameter	Description	Example value
<code>TokenEndpointUrl</code>	The base URL where the authentication server listens for requests to issue access tokens. You can see this value in the openid-configuration file of the authentication server under <code>token_endpoint</code> key.	<code>https://<AD FS FQDN>/adfs/oauth2/token/</code>
<code>AuthorizationEndpointUrl</code>	The base URL at the authentication server where users are redirected in order to authenticate. You can see this value in the openid-configuration file under the <code>authorization_endpoint</code> key.	<code>https://<AD FS FQDN>/adfs/oauth2/authorize/</code>

Parameter	Description	Example value
RedirectUrl	The URL where the user is redirected after he approves the authentication request. This URL must be on a localhost and a free port on the machine where PowerCLI is running. This URL must use the <code>http</code> schema.	<code>http://localhost:8844/auth</code>
ClientId	The ID that you received from the authentication server when you registered the client application.	<code>powercli-native</code>
OtherArguments	A hashtable of (String, String) pairs that represent arguments to the server-specific parameters. In the following example, it is used for an AD FS Application Group for vCenter Server with an example ID "my-vcenter".	<code>{ "resource" = "my-vcenter" }</code>

2 Exchange the OAuth security context for an SAML security context.

```
$samlSecContext = New-VISamlSecurityContext -VCenterServer 'myVC' -OAuthSecurityContext $oauthSecContext
```

3 Connect to your vCenter Server system by using the SAML security context.

```
Connect-VIServer -Server 'myVC' -SamlSecurityContext $samlSecContext
```

Manage Virtual Machines on vSphere

With PowerCLI, you can automate various administration tasks on virtual machines, for example retrieving information, shutting down and powering off virtual machines.

Procedure

1 View all virtual machines on the target system.

```
Get-VM
```

2 Save the name and the power state properties of the virtual machines in the *ResourcePool* resource pool into a file named `myVMProperties.txt`.

```
$respool = Get-ResourcePool ResourcePool
Get-VM -Location $respool | Select-Object Name, PowerState > myVMProperties.txt
```

3 Start the *VM* virtual machine.

```
Get-VM VM | Start-VM
```

4 Get information of the guest OS of the *VM* virtual machine.

```
Get-VMGuest VM | fc
```

5 Shut down the OS of the *VM* virtual machine.

```
Stop-VMGuest VM
```

- 6 Power off the *VM* virtual machine.

```
Stop-VM VM
```

- 7 Move the virtual machine *VM* from the *Host01* host to the *Host02* host.

```
Get-VM -Name VM -Location Host01 | Move-VM -Destination Host02
```

Note If the virtual machine you want to move across hosts is powered on, it must be located on a shared storage registered as a datastore on both the original and the new host.

Add a Standalone Host to a vCenter Server System

You can add standalone hosts to a vCenter Server system by using the `Add-VMHost` cmdlet. After adding the hosts, you will be able to manage them through the vCenter Server system.

Prerequisites

Verify that you are connected to a vCenter Server system.

Procedure

- 1 View all hosts on the vCenter Server system that you have established a connection with.

```
Get-VMHost
```

- 2 Add the *Host* standalone host.

```
Add-VMHost -Name Host -Location (Get-Datacenter DC) -User root -Password pass
```

Set the License Key for a Host on vCenter Server

You can set the license key for a host on a vCenter Server system by using the `LicenseKey` parameter of the `Set-VMHost` cmdlet.

Prerequisites

Verify that you are connected to a vCenter Server system.

Procedure

- 1 Save the *Host* host object as a variable.

```
$vmhost = Get-VMHost -Name Host
```

2 Set the host to evaluation mode or provide a valid license key.

- ◆ Set the host to evaluation mode by providing the evaluation key.

```
Set-VMHost -VMHost $vmhost -LicenseKey 00000-00000-00000-00000-00000
```

- ◆ Provide a valid license key.

```
Set-VMHost -VMHost $vmhost -LicenseKey Your_license_key
```

Activate Maintenance Mode for a Host on vCenter Server

To complete some specific administration tasks, you might need to activate maintenance mode for a host. On vCenter Server, you can activate maintenance mode by using the `Set-VMHost` cmdlet.

Prerequisites

Verify that you are connected to a vCenter Server system.

Procedure

- 1 Save the *Host* host object as a variable.

```
$vmhost = Get-VMHost -Name Host
```

- 2 Get the cluster to which *Host* belongs and save the cluster object as a variable.

```
$vmhostCluster = Get-Cluster -VMHost $vmhost
```

- 3 Start a task that activates maintenance mode for the *Host* host and save the task object as a variable.

```
$updateHostTask = Set-VMHost -VMHost $vmhost -State "Maintenance" -RunAsync
```

Note If the host is not automated or is partially automated and has powered-on virtual machines running on it, you must use the `RunAsync` parameter and wait until all powered-on virtual machines are relocated or powered off before applying DRS recommendations.

- 4 Get and apply the recommendations generated by DRS.

```
Get-DrsRecommendation -Cluster $vmhostCluster | where {$_.Reason -eq "Host is entering maintenance mode"} | Invoke-DrsRecommendation
```

- 5 Get the task output object and save it as a variable.

```
$myUpdatedHost = Wait-Task $updateHostTask
```

Create vSphere Inventory Objects

By using PowerCLI cmdlets, you can automate creating different inventory objects on vSphere.

Prerequisites

Verify that you are connected to a vCenter Server system.

Procedure

- 1 Get the inventory root folder and create a new folder named `Folder` in it.

```
$folder = Get-Folder -NoRecursion | New-Folder -Name Folder
```

- 2 Create a new data center named `DC` in the `Folder` folder.

```
New-Datacenter -Location $folder -Name DC
```

- 3 Create a folder named `Folder1` under `DC`.

```
Get-Datacenter DC | New-Folder -Name Folder1
$folder1 = Get-Folder -Name Folder1
```

- 4 Create a new cluster `Cluster1` in the `Folder1` folder.

```
New-Cluster -Location $folder1 -Name Cluster1 -DrsEnabled -DrsAutomationLevel
FullyAutomated
```

Distributed Resource Scheduler (DRS) is a feature that provides automatic allocation of cluster resources.

- 5 Add a host in the cluster by using the `Add-VMHost` command, and provide credentials when prompted.

```
$vmhost1 = Add-VMHost -Name 10.23.112.345 -Location (Get-Cluster Cluster1)
```

- 6 Create a resource pool in the root resource pool of the cluster.

```
$myClusterRootRP = Get-Cluster Cluster1 | Get-ResourcePool -Name Resources
New-ResourcePool -Location $myClusterRootRP -Name MyRP1 -CpuExpandableReservation
$true -CpuReservationMhz 500 -CpuSharesLevel high -MemExpandableReservation $true
-MemReservationGB 1 -MemSharesLevel high
```

- 7 Create a virtual machine asynchronously.

```
$vmCreationTask = New-VM -Name VM2 -VMHost $vmhost1 -ResourcePool MyRP01 -DiskGB 100
-MemoryGB 2 -RunAsync
```

The `RunAsync` parameter indicates that the command runs asynchronously. This means that in contrast to a synchronous operation, you do not have to wait for the process to complete before supplying the next command at the command line.

Create Virtual Machines on vCenter Server Using an XML Specification File

You can use a specification provided in an XML file to automate the creation of virtual machines on vCenter Server.

Prerequisites

Verify that you are connected to a vCenter Server system.

The `myVM.xml` file must be present with the following content:

```
<CreateVM>
<VM>
<Name>MyVM1</Name>
<HDDCapacity>100</HDDCapacity>
</VM>
<VM>
<Name>MyVM2</Name>
<HDDCapacity>100</HDDCapacity>
</VM>
</CreateVM>
```

Procedure

- 1 Read the content of the `myVM.xml` file.

```
[xml]$s = Get-Content myVM.xml
```

- 2 Create the virtual machines.

```
$s.CreateVM.VM | foreach {New-VM -VMHost $vmHost1 -Name $_.Name -DiskGB $_.HDDCapacity}
```

Manage Virtual Machine Templates on vCenter Server

You can use PowerCLI to create virtual machines templates and convert them to virtual machines on vCenter Server.

Note A virtual machine template is a reusable image created from a virtual machine. The template, as a derivative of the source virtual machine, includes virtual hardware components, an installed guest operating system, and software applications.

Prerequisites

Verify that you are connected to a vCenter Server system.

Procedure

- 1 Create a template from the *VM1* virtual machine.

```
New-Template -VM VM1 -Name VM1Template -Location (Get-Datacenter DC )
```

- 2 Convert the *VM1Template* template for use by a virtual machine named *VM3*.

```
Get-Template VM1Template | Set-Template -ToVM -Name VM3
```

- 3 Create a template from the *VM2* virtual machine.

```
New-Template -VM VM2 -Name VM2Template -Location (Get-Datacenter DC )
```

- 4 Convert the *VM2Template* template to a virtual machine named *VM4*.

```
Get-Template VM2Template | Set-Template -ToVM -Name VM4
```

- 5 Convert the *VM4* virtual machine to a template.

```
Set-VM -VM VM4 -ToTemplate -Name "VM4Template"
```

- 6 Create a template called *VM3Template* by cloning *VM2Template*.

```
Get-Template VM2Template | New-Template -Name VM3Template -VMHost $targetVMHost
```

Create and Use Snapshots on vCenter Server

You can use the `Snapshot` parameter of `Get-VM` to take a snapshot of virtual machines and then revert the states of the virtual machines back to the snapshot.

Note A snapshot captures the memory, disk, and settings state of a virtual machine at a particular moment. When you revert to a snapshot, you return all these items to the state they were in at the time you took that snapshot.

Prerequisites

Verify that you are connected to a vCenter Server system.

Procedure

- 1 Take a snapshot of all virtual machines in the *MyRP01* resource pool.

```
Get-ResourcePool MyRP01 | Get-VM | New-Snapshot -Name InitialSnapshot
```

The `Location` parameter takes arguments of the `VIContainer` type, on which `Cluster`, `Datacenter`, `Folder`, `ResourcePool`, and `VMHost` object types are based. Therefore, the `Location` parameter can use arguments of all these types.

- 2 Revert all virtual machines in the *MyRP01* resource pool to the *InitialSnapshot* snapshot.

```
$VMs = Get-ResourcePool MyRP01 | Get-VM
foreach( $vm in $VMs ) { Set-VM -VM $vm -Snapshot InitialSnapshot }
```

Update the Resource Configuration Settings of a Virtual Machine on vCenter Server

You can use the `Set-VMResourceConfiguration` cmdlet to modify the resource configuration properties of a virtual machine, including memory, CPU shares, and other settings.

Prerequisites

Verify that you are connected to a vCenter Server system.

Procedure

- 1 View the resource configuration for the *VM1* virtual machine.

```
Get-VMResourceConfiguration -VM VM1
```

- 2 View the disk share of the *VM1* virtual machine.

```
Get-VMResourceConfiguration -VM VM1 | Format-Custom -Property DiskResourceConfiguration
```

- 3 Change the memory share of the *VM1* virtual machine to low.

```
Get-VM VM1 | Get-VMResourceConfiguration | Set-VMResourceConfiguration -MemSharesLevel low
```

- 4 Change the CPU shares of the *VM1* virtual machine to high.

```
Get-VM VM1 | Get-VMResourceConfiguration | Set-VMResourceConfiguration -CpuSharesLevel high
```

- 5 Change the disk share of the *VM1* virtual machine to 100.

```
$vm1 = Get-VM VM1
$vm1disk = Get-HardDisk $vm1
Get-VMResourceConfiguration $vm1 | Set-VMResourceConfiguration -Disk $vm1disk
-DiskSharesLevel custom -NumDiskShares 100
```

Get a List of Hosts on a vCenter Server System and View Their Properties

With PowerCLI, you can get information about all available hosts in a data center and view their properties.

Prerequisites

Verify that you are connected to a vCenter Server system.

Procedure

- 1 Get a list of all hosts that are part of a data center.

```
Get-Datacenter DC | Get-VMHost | Format-Custom
```

- 2 View the properties of the first host in the data center.

```
Get-Datacenter DC | Get-VMHost | Select-Object -First 1 | Format-Custom
```

- 3 View the `Name` and the `OverallStatus` properties of the hosts in the `DC` data center.

```
Get-Datacenter DC | Get-VMHost | Get-View | Format-Table -Property Name, OverallStatus  
-AutoSize
```

- 4 View all hosts and their properties, and save the results to a file.

```
Get-Datacenter DC | Get-VMHost | Format-Custom | Out-File -FilePath hosts.txt
```

- 5 View a list of the hosts that are in maintenance mode and can be configured for vMotion operations.

```
Get-VMHost -State maintenance | Get-View | Where-Object -FilterScript { $_.capability -ne  
$null -and $_.capability.vmotionSupported }
```

Change the Host Advanced Configuration Settings on vCenter Server

You can modify host configuration, including advanced settings related to virtual machine migration, and apply them to another host.

Prerequisites

Verify that you are connected to a vCenter Server system.

Procedure

- 1 Change the migration timeout for the `ESXHost1` host.

```
Get-VMHost ESXHost1 | Set-VmHostAdvancedConfiguration -Name Migrate.NetTimeout -Value  
( [system.int32] 10 )
```

- 2 Enable creation of a checksum of the virtual machines memory during the migration.

```
Get-VMHost ESXHost1 | Set-VmHostAdvancedConfiguration -Name Migrate.MemChksum -Value  
( [system.int32] 1 )
```

- 3 Get the `ESXHost1` host migration settings.

```
$migrationSettings = Get-VMHost ESXHost1 | Get-VmHostAdvancedConfiguration -Name Migrate.*
```

4 Apply the migration settings to *ESXHost2*.

```
Set-VmHostAdvancedConfiguration -VMHost ESXHost2 -Hashtable $migrationSettings
```

Move a Virtual Machine to a Different Host Using VMware vSphere vMotion

You can migrate a virtual machine between vCenter Server hosts by using vSphere vMotion.

Note You can use vSphere vMotion to move a powered-on virtual machine from one host to another.

Prerequisites

Verify that you are connected to a vCenter Server system.

The virtual machine must be stored on a datastore shared by the current and the destination host, and the vMotion interfaces on the two hosts must be configured.

Procedure

- ◆ Get the *VM1* virtual machine and move it to a host named *ESXHost2*.

```
Get-VM VM1 | Move-VM -Destination (Get-VMHost ESXHost2)
```

Move a Virtual Machine to a Different Datastore Using VMware vSphere Storage vMotion

You can migrate a virtual machine between datastores using the VMware Storage vMotion feature of vCenter Server.

Note You can use Storage vMotion to move a powered-on virtual machine from one datastore to another.

Prerequisites

Verify that you are connected to a vCenter Server system.

The host on which the virtual machine is running must have access both to the datastore where the virtual machine is located and to the destination datastore.

Procedure

- ◆ Get the *VM1* virtual machine and move it to a datastore named *DS2*.

```
Get-VM VM1 | Move-VM -Datastore DS2
```

Move a Virtual Machine to a Different vCenter Server System

You can migrate a virtual machine from one vCenter Server system to another by using Cross vCenter Server vMotion.

You can move virtual machines between vCenter Server systems of vSphere version 6.0 and later by using the `Move-VM` cmdlet. When you move a virtual machine from one vCenter Server system to another, only datastores are supported as storage destinations.

Procedure

- 1 Connect to the *myVC1* source vCenter Server system.

```
Connect-VIServer 'myVC1' -Username MyUser1 -Password MyPass1
```

- 2 Connect to the *myVC2* destination vCenter Server system.

```
Connect-VIServer 'myVC2' -Username MyUser2 -Password MyPass2
```

- 3 Store the *MyVM* virtual machine, its network adapters, the destination host, port group, and datastore in variables.

```
$vm = Get-VM 'myVM' -Location 'myVMhostOnVC1'
$destination = Get-VMHost 'MyVMhostOnVc2'
$networkAdapter = Get-NetworkAdapter -VM $vm
$destinationPortGroup = Get-VDPortgroup -VDSwitch 'myVDSwitchOnVC2' -Name 'myPortGroup'
$destinationDatastore = Get-Datastore 'MyDatastoreOnVc2'
```

- 4 Migrate the virtual machine to the specified destination host and datastore and attach the virtual machine network adapters to the destination port group.

```
Move-VM -VM $vm -Destination $destination -NetworkAdapter $networkAdapter -PortGroup
$destinationPortGroup -Datastore $destinationDatastore
```

Create a Host Profile on a vCenter Server System

The VMware Host Profiles feature enables you to create standard configurations for ESXi hosts. With PowerCLI, you can automate creation and modifying of host profiles.

Prerequisites

Verify that you are connected to a host that runs vCenter Server 4.1 or later.

Procedure

- 1 Get the host named *Host1* and store it in the *\$vmhost* variable.

```
$vmhost = Get-VMHost Host1
```

- 2 Create a profile based on the *Host1* host.

```
New-VMHostProfile -Name MyHostProfile01 -Description "This is my test profile based on
Host1." -ReferenceHost $vmhost
```

- 3 Get the newly created host profile.

```
$hp1 = Get-VMHostProfile -Name MyHostProfile01
```

- 4 Change the description of the *HostProfile1* host profile.

```
Set-VMHostProfile -Profile $hp1 -Description "This is my old test host profile based on
Host1."
```

Apply a Host Profile to a Host on vCenter Server

To simplify operational management of large-scale environments, you can apply standard configurations called host profiles to hosts on vCenter Server. If you want to set up a host to use the same host profile as a reference host, you can attach the host to a profile.

Prerequisites

Verify that you are connected to a host that runs vCenter Server 4.1 or later.

Procedure

- 1 Get the *Host2* host.

```
$vmhost2 = Get-VMHost Host2
```

- 2 Attach the *Host2* host to the *HostProfile1* host profile.

```
Set-VMHost -VMHost $vmhost2 -Profile HostProfile1
```

- 3 Verify that the *Host2* host is compliant with the *HostProfile1* profile.

```
Test-VMHostProfileCompliance -VMHost $vmhost2
```

The output of this command contains the noncompliant settings of the host, if any.

- 4 Apply the profile to the *Host2* host.

```
$neededVariables = Invoke-VMHostProfile -Entity $vmhost2 -Profile $hp1 -Confirm:$false
```

The *\$neededVariables* variable contains the names of all required variables and their default or current values, as returned by the server. Otherwise, the *\$neededVariables* variable contains the name of the host on which the profile has been applied.

Manage Statistics and Statistics Intervals on vCenter Server

You can use the PowerCLI cmdlets to automate tasks for viewing and managing statistics for vCenter Server inventory objects.

You can modify the properties of a statistics interval and view statistics for a selected cluster.

Prerequisites

Verify that you are connected to a vCenter Server system.

Procedure

- 1 Increase the amount of time for which statistics of the previous day are stored.

```
Set-StatInterval -Interval "past day" -StorageTimeSecs 700000
```

- 2 View the available memory metric types for the *Cluster1* cluster.

```
$cluster = Get-Cluster Cluster1
$statTypes = Get-StatType -Entity $cluster -Interval "past day" -Name mem.*
```

- 3 View the cluster statistics collected for the day.

```
Get-Stat -Entity $cluster -Start ([System.DateTime]::Now.AddDays(-1)) -Finish
([System.DateTime]::Now) -Stat $statTypes
```

Modify the Settings of the NIC Teaming Policy for a Virtual Switch

You can set the NIC teaming policy on a vSwitch. The NIC teaming policy determines the load balancing and failover settings of a virtual switch and lets you mark NICs as unused.

Prerequisites

Verify that you are connected to a vCenter Server system.

Procedure

- 1 Get a list of the physical NIC objects on the host network and store them in a variable.

```
$pn = Get-VMHost 10.23.123.128 | Get-VMHostNetwork | Select -Property physicalnic
```

- 2 Store the physical NIC objects you want to mark as unused in separate variables.

```
$pn5 = $pn.PhysicalNic[2]
$pn6 = $pn.PhysicalNic[3]
$pn7 = $pn.PhysicalNic[0]
```

- 3 View the NIC teaming policy of the *VSwitch01* virtual switch.

```
$policy = Get-VirtualSwitch -VMHost 10.23.123.128 -Name VSwitch01 | Get-NicTeamingPolicy
```

- 4 Change the policy of the switch to indicate that the *\$pn5*, *\$pn6*, and *\$pn7* network adapters are unused.

```
$policy | Set-NicTeamingPolicy -MakeNicUnused $pn5, $pn6, $pn7
```

- 5 Modify the load balancing and failover settings of the virtual switch NIC teaming policy.

```
$policy | Set-NicTeamingPolicy -BeaconInterval 3 -LoadBalancingPolicy 3  
-NetworkFailoverDetectionPolicy 1 -NotifySwitches $false -FailbackEnabled $false
```

Create a vApp on vCenter Server

With PowerCLI, you can create and manage vApps.

Prerequisites

Verify that you are connected to a vCenter Server system.

Procedure

- 1 Create a new vApp named *VApp* on a host.

```
New-VApp -Name VApp -CpuLimitMhz 4000 -CpuReservationMhz 1000 -Location (Get-VMHost Host1)
```

- 2 Start the new virtual appliance.

```
Start-VApp VApp
```

Modify the Properties of a vApp

With PowerCLI, you can start and stop vApps, and modify their properties.

Prerequisites

Verify that you are connected to a vCenter Server system.

Procedure

- 1 Get the vApp named *VApp* and stop it.

```
Get-VApp VApp | Stop-VApp -Confirm:$false
```

- 2 Change the name and memory reservation for the vApp.

```
Get-VApp VApp | Set-VApp -Name OldVApp -MemReservationGB 2
```

Export or Import vApps

You can import and export vApps to OVA and OVF files.

Prerequisites

Verify that you are connected to a vCenter Server system.

Procedure

- 1 Get the vApp you want to export.

```
$oldVApp = Get-VApp OldVApp
```

- 2 Export the *OldVApp* vApp to a local directory and name the exported appliance *WebApp*.

```
Export-VApp -VApp $oldVApp -Name WebApp -Destination D:\vapps\ -CreateSeparateFolder
```

- 3 Import the *WebApp* vApp from a local directory to the *Storage2* datastore.

```
Import-VApp -Source D:\vapps\WebApp\WebApp.ovf -VMHost (Get-VMHost Host1) -Datastore (Get-Datastore -VMHost MyHost01 -Name Storage2)
```

Create an iSCSI Host Storage

For a host, you can enable iSCSI, add iSCSI targets, and create new host storages.

Prerequisites

Verify that you are connected to a vCenter Server system.

Procedure

- 1 Enable software iSCSI on a host.

```
$vmhost = Get-VMHost ESXHost1
Get-VMHostStorage $myHost | Set-VMHostStorage -SoftwareIScsiEnabled $true
```

- 2 Get the iSCSI HBA that is on the host.

```
$iscsiHba = Get-VMHostHba -Type iScsi
```

- 3 Add a new iSCSI target for dynamic discovery.

```
$iscsiHba | New-IScsiHbaTarget -Address 192.168.0.1 -Type Send
```

- 4 Rescan the HBAs on the host.

```
Get-VMHostStorage $vmhost -RescanAllHba
```


5 Get the path to the SCSI LUN.

```
$lunPath = Get-ScsiLun -VMHost $vmhost -CanonicalName ($iscsiHba.Device + "**") | Get-ScsiLunPath
```

You can provide the LUN path by using its canonical name beginning with the device name of the iSCSI HBA.

6 Create a new host storage.

```
New-Datastore -Vmfs -VMHost $vmhost -Path $lunpath.LunPath -Name iSCSI
```

Add Passthrough Devices to a Host and Virtual Machine

You can get information about existing passthrough devices and add new SCSI and PCI devices to virtual machines and hosts.

Prerequisites

Verify that you are connected to a vCenter Server system.

Procedure

1 Get a list of the PCI passthrough devices of the *VMHost* host

```
$vmhost = Get-VMHost ESXHost
Get-PassthroughDevice -VMHost $vmhost -Type Pci
```

2 Get a list of the SCSI passthrough devices of the *VM* virtual machine

```
$vm = Get-VM VM
Get-PassthroughDevice -VM $vm -Type Scsi
```

3 Add a SCSI passthrough device to the *VM* virtual machine

```
$scsiDeviceList = Get-PassthroughDevice -VMHost ESXHost -Type Scsi
Add-PassthroughDevice -VM $vm -PassthroughDevice $scsiDeviceList[0]
```

Create a Custom Property Based on an Extension Data Property

You can create custom properties to add more information to vSphere objects. Custom properties based on extension data properties correspond directly to the property of the corresponding .NET view object.

Prerequisites

Verify that you are connected to a vCenter Server system.

Procedure

- 1 Create a new custom property based on the `Guest.ToolsVersion` property.

```
New-VIProperty -ObjectType VirtualMachine -Name ToolsVersion -ValueFromExtensionProperty
'Guest.ToolsVersion'
```

- 2 View the `ToolsVersion` properties of the available virtual machines.

```
Get-VM | Select Name, ToolsVersion
```

Results

You have created a custom property named `ToolsVersion` for `VirtualMachine` objects.

Create a Script-Based Custom Property for a vSphere Object

You can create a custom property by writing a script and providing a name for the property. The script evaluates when the custom property is called for the first time.

Prerequisites

Verify that you are connected to a vCenter Server system.

Procedure

- 1 Create a new custom property named `NameOfHost` that stores the name of the host on which a virtual machine resides.

```
New-VIProperty -Name NameOfHost -ObjectType VirtualMachine -Value { return
$args[0].VMHost.Name }
```

- 2 View the `NameOfHost` properties of the available virtual machines.

```
Get-VM | select Name, NameOfHost | Format-Table -AutoSize
```

Results

You created a custom script property named `NameOfHost` for `VirtualMachine` objects.

Apply a Customization Object to a Cloned Virtual Machine

You can apply a custom configuration to a cloned virtual machine by using a customization object.

Note This feature runs only on a 32-bit PowerCLI process.

Prerequisites

Verify that you are connected to a vCenter Server system.

Procedure

- 1 Get the *Spec* customization specification and clone it for temporary use.

```
Get-OSCustomizationSpec Spec | New-OSCustomizationSpec -Type NonPersistent -Name ClientSpec
```

- 2 Change the `NamingPrefix` property of the customization object to the name of the virtual machine you want to create.

```
Set-OSCustomizationSpec -Spec ClientSpec -NamingPrefix VM1
```

- 3 Create a virtual machine named *VM1* by cloning the existing *VM* virtual machine and applying the customization specification.

```
Get-VM VM | New-VM -VMHost Host -Datastore Storage1 -OSCustomizationSpec ClientSpec -Name VM1
```

Modify the Default NIC Mapping Object of a Customization Specification

You can modify the default NIC mapping object of a customization specification and apply the specification on a newly created virtual machine.

Procedure

- 1 Create a nonpersistent customization specification for Windows operating systems.

```
New-OSCustomizationSpec -Type NonPersistent -Name Spec -OSType Windows -Workgroup Workgroup -OrgName Company -Fullname User -ProductKey "valid_key" -ChangeSid -TimeZone "Central European" -NamingScheme VM
```

- 2 View the default NIC mapping objects of the *Spec* specification.

```
Get-OSCustomizationNicMapping -Spec Spec | Set-OSCustomizationNicMapping -IpMode UseStaticIP -IpAddress 172.16.1.30 -SubnetMask 255.255.255.0 -DefaultGateway 172.16.1.1 -Dns 172.16.1.1
```

Each customization specification object has one default NIC mapping object.

- 3 Modify the default NIC mapping object of the *Spec* customization specification to use static IP.

```
Get-OSCustomizationNicMapping -Spec Spec | Set-OSCustomizationNicMapping -IpMode UseStaticIP -IpAddress 172.16.1.30 -SubnetMask 255.255.255.0 -DefaultGateway 172.16.1.1 -Dns 172.16.1.1
```

- 4 Create a new virtual machine named *VM1* from a template, and apply the static IP settings.

```
New-VM -Name VM1 -VMHost Host -Datastore Storage1 -OSCustomizationSpec Spec -Template
Template
```

Modify Multiple NIC Mapping Objects of a Customization Specification

You can modify multiple NIC mapping objects of a customization specification and apply the specification to an existing virtual machine.

Procedure

- 1 Get the network adapters of a virtual machine named *VM*.

```
Get-NetworkAdapter VM
```

When you apply a customization specification, each network adapter of the customized virtual machine must have a corresponding NIC mapping object. You can correlate network adapters and NIC mapping objects either by their position numbers, or by MAC address.

- 2 Create a customization specification named *Spec*.

```
New-OSCustomizationSpec -Type NonPersistent -Name Spec -OSType Windows -Workgroup
Workgroup -OrgName Company -Fullname User -ProductKey "valid_key" -ChangeSid -TimeZone
"Central European" -NamingScheme VM
```

- 3 Add a new NIC mapping object that uses a static IP address.

```
New-OSCustomizationNicMapping -Spec Spec -IpMode UseStaticIP -IpAddress 172.16.1.30
-SubnetMask 255.255.255.0 -DefaultGateway 172.16.1.1 -Dns 172.16.1.1
```

- 4 View the NIC mapping objects and verify that two NIC mapping objects are available.

```
Get-OSCustomizationNicMapping -Spec Spec
```

The default NIC mapping object is DHCP enabled, and the newly added one uses a static IP address.

- 5 Apply the *Spec* customization specification to the *VM* virtual machine.

```
Get-VM VM | Set-VM -OSCustomizationSpec -Spec Spec
```

- 6 Associate a network adapter from the *VMNetwork* network with the NIC mapping object that uses DHCP mode.

```
$netAdapter = Get-NetworkAdapter VM | where { $_.NetworkName -eq 'VMNetwork' }
Get-OSCustomizationNicMapping -Spec Spec | where { $_.IPMode -eq 'UseDHCP' } | Set-
OSCustomizationNicMapping -NetworkAdapterMac $netAdapter.MacAddress
```

Create Multiple Virtual Machines that Use Static IP Addresses

You can deploy multiple virtual machines with a single network adapter and configure the deployed virtual machines to use static IP addresses by applying a customization specification.

Prerequisites

Verify that you have defined a list of static IP addresses in a CSV file.

Procedure

- 1 Define the naming convention for the virtual machines.

```
$vmNameTemplate = "VM-{0:D3}"
```

- 2 Save the cluster in which the virtual machines should be created into a variable.

```
$cluster = Get-Cluster MyCluster
```

- 3 Save the template on which the virtual machines should be based into a variable.

```
$template = Get-Template MyTemplate
```

- 4 Create the virtual machines.

```
$vmList = @()

for ($i = 1; $i -le 100; $i++) {
    $vmName = $vmNameTemplate -f $i
    $vmList += New-VM -Name $vmName -ResourcePool $cluster -Template $template
}

```

- 5 Save the static IP addresses from the stored CSV file into a variable.

```
$staticIpList = Import-CSV C:\StaticIPs.csv
```

- 6 Create the customization specification.

```
$linuxSpec = New-OSCustomizationSpec -Name LinuxCustomization -Domain vmware.com -
DomainUsername "your_domain_username" -DomainPassword "your_domain_password" -DnsServer
"192.168.0.10", "192.168.0.20" -NamingScheme VM -OSType Linux

```

- 7 Clone the customization specification to a nonpersistent type.

```
$specClone = New-OSCustomizationSpec -Spec $linuxSpec -Type NonPersistent
```

- 8 Apply the customization specification to each virtual machine.

```
for ($i = 0; $i -lt $vmList.Count; $i++) {
    # Acquire a new static IP from the list
    $ip = $staticIpList[$i].IP
}

```

```

# The specification has a default NIC mapping - retrieve it and update it with the
static IP
$nicMapping = Get-OSCustomizationNicMapping -OSCustomizationSpec $specClone
$nicMapping | Set-OSCustomizationNicMapping -IpMode UseStaticIP -IpAddress $ip -
SubnetMask "255.255.252.0" -DefaultGateway "192.168.0.1"

# Apply the customization
Set-VM -VM $vmList[$i] -OSCustomizationSpec $specClone -Confirm:$false
}

```

Create Multiple Virtual Machines with Two Network Adapters

You can deploy multiple virtual machines with two network adapters each and configure each adapter to use specific network settings by applying a customization specification.

You can configure each virtual machine to have one network adapter attached to a public network and one network adapter attached to a private network. You can configure the network adapters on the public network to use static IP addresses and the network adapters on the private network to use DHCP.

Prerequisites

Verify that you have defined a list of static IP addresses in a CSV file.

Procedure

- 1 Define the naming convention for the virtual machines.

```
$vmNameTemplate = "VM-{0:D3}"
```

- 2 Save the cluster in which the virtual machines should be created into a variable.

```
$cluster = Get-Cluster MyCluster
```

- 3 Save the template on which the virtual machines should be based into a variable.

```
$template = Get-Template MyTemplate
```

- 4 Create the virtual machines.

```

$vmList = @()

for ($i = 1; $i -le 100; $i++) {
    $vmName = $vmNameTemplate -f $i
    $vmList += New-VM -Name $vmName -ResourcePool $cluster -Template $template
}

```

- 5 Save the static IP addresses from the stored CSV file into a variable.

```
$staticIpList = Import-CSV C:\StaticIPs.csv
```

- 6 Create the customization specification.

```
$linuxSpec = New-OSCustomizationSpec -Name LinuxCustomization -Domain vmware.com -
DnsServer "192.168.0.10", "192.168.0.20" -NamingScheme VM -OSType Linux -Type NonPersistent
```

- 7 Apply the customization specification to each virtual machine.

```
for ($i = 0; $i -lt $vmList.Count; $i++) {
    # Acquire a new static IP from the list
    $ip = $staticIpList[$i].IP

    # Remove any NIC mappings from the specification
    $nicMapping = Get-OSCustomizationNicMapping -OSCustomizationSpec $linuxSpec
    Remove-OSCustomizationNicMapping -OSCustomizationNicMapping $nicMapping -Confirm:$false

    # Retrieve the virtual machine's network adapter attached to the public network named
    "Public"
    $publicNIC = $vmList[$i] | Get-NetworkAdapter | where {$_.NetworkName -eq "Public"}

    # Retrieve the virtual machine's network adapter attached to the private network named
    "Private"
    $privateNIC = $vmList[$i] | Get-NetworkAdapter | where {$_.NetworkName -eq "Private"}

    # Create a NIC mapping for the "Public" NIC that should use static IP
    $linuxSpec | New-OSCustomizationNicMapping -IpMode UseStaticIP -IpAddress
    $ip -SubnetMask "255.255.252.0" -DefaultGateway "192.168.0.1" -NetworkAdapterMac
    $publicNIC.MacAddress

    # Create a NIC mapping for the "Private" NIC that should use DHCP
    $linuxSpec | New-OSCustomizationNicMapping -IpMode UseDhcp -NetworkAdapterMac
    $privateNIC.MacAddress

    # Apply the customization
    Set-VM -VM $vmList[$i] -OSCustomizationSpec $linuxSpec -Confirm:$false
}
```

Create a vSphere Role and Assign Permissions to a User

With PowerCLI, you can automate management of vSphere permissions, roles, and privileges.

Note vSphere permissions determine your level of access to vCenter Server, and ESXi hosts. Privileges define individual rights to perform actions and access object properties. Roles are predefined sets of privileges.

Prerequisites

Verify that you are connected to a vCenter Server system.

Procedure

- 1 Get the privileges of the **ReadOnly** role.

```
$readOnlyPrivileges = Get-VIPrivilege -Role ReadOnly
```

- 2 Create a new role with custom privileges.

```
$role1 = New-VIRole -Privilege $readOnlyPrivileges -Name Role1
```

- 3 Add the **PowerOn** privileges to the new role.

```
$powerOnPrivileges = Get-VIPrivilege -Name "PowerOn"
$role1 = Set-VIRole -Role $role1 -AddPrivilege $powerOnPrivileges
```

- 4 Create a permission and apply it to a vSphere root object.

```
$rootFolder = Get-Folder -NoRecursion
$permission1 = New-VIPermission -Entity $rootFolder -Principal "user" -Role readonly
-Propagate
```

The `Principal` parameter accepts both local and domain users and groups if the vCenter Server system is joined in AD.

- 5 Update the new permission with the custom role.

```
$permission1 = Set-VIPermission -Permission $permission1 -Role $role1
```

Results

You created a new role and assigned permissions to a user.

View Actions and Triggers for an Alarm on vCenter Server

You can see which actions and triggers are configured for an alarm.

Prerequisites

Verify that you are connected to a vCenter Server system.

Procedure

- 1 Get all PowerCLI supported alarm actions for the *Host Processor Status* alarm.

```
Get-AlarmDefinition -Name "Host Processor Status" | Get-AlarmAction -ActionType
"ExecuteScript", "SendSNMP", "SendEmail"
```

- 2 Get all the triggers for the first alarm definition found.

```
Get-AlarmDefinition | select -First 1 | Get-AlarmTrigger
```


Create Alarm Definitions on vCenter Server

With PowerCLI, you can create vCenter Server alarm definitions.

Prerequisites

Verify that you are connected to a vCenter Server system.

Procedure

- 1 Create an alarm action trigger.

```
$actionTrigger = New-AlarmActionTrigger -StartStatus Green -EndStatus Yellow
```

- 2 Create an alarm action email.

```
$emailAction = New-AlarmAction -Email -To 'test@vmware.com' -CC @('test1@vmware.com',
'test2@vmware.com') -Body 'Email text' -Subject 'Email subject' -AlarmActionTrigger
$actionTrigger
```

- 3 Create an event-based alarm trigger.

```
$vmReconfigEvt = Get-EventType | Where-Object {$_.Description -eq "VM reconfigured"}
$eventTrigger = New-AlarmTrigger -EventType $vmReconfigEvt -EntityStatus Yellow
-EntityType "VirtualMachine"
```

- 4 Create an alarm definition.

```
$vm = Get-VM | Select-Object -First 1
New-AlarmDefinition -Name "MyAlarm" -Description "Description" -AlarmTrigger $eventTrigger
-AlarmAction $emailAction -Entity $vm
```

Modify Alarm Definitions on vCenter Server

With PowerCLI, you can modify vCenter Server alarm definitions.

Prerequisites

Verify that you are connected to a vCenter Server system.

Procedure

- 1 For all host alarms, modify the interval after the action repeats.

```
Get-AlarmDefinition -Entity (Get-VMHost) | foreach { $_ | Set-AlarmDefinition
-ActionRepeatMinutes ($_.ActionRepeatMinutes + 1)}
```

- 2 Modify the name and the description of a selected alarm definition, and enable the alarm.

```
Get-AlarmDefinition -Name AlarmDefinition | Set-AlarmDefinition -Name AlarmDefinitionNew
-Description 'Alarm Definition Description' -Enabled:$true
```

Create Alarm Actions and Triggers on vCenter Server

With PowerCLI, you can create vCenter Server alarm actions and triggers.

Prerequisites

Verify that you are connected to a vCenter Server system.

Procedure

- 1 Create an alarm action `email` for the renamed alarm definition.

```
Get-AlarmDefinition -Name AlarmDefinitionNew | New-AlarmAction -Email -To
'test@vmware.com' -CC @('test1@vmware.com', 'test2@vmware.com') -Body 'Email text'
-Subject 'Email subject'
```

- 2 Create an `snmp` alarm action.

```
Get-AlarmDefinition -Name AlarmDefinitionNew | New-AlarmAction -Snmp
```

- 3 Create a `script` alarm action.

```
Get-AlarmDefinition -Name AlarmDefinitionNew | New-AlarmAction -Script -ScriptPath
'c:\test.ps1'
```

- 4 Create a metric-based alarm trigger.

```
$metric = Get-Metric -MetricGroup CPU -Name Usage | Where-Object {$_.Key -eq 2}
$metricTrigger = New-AlarmTrigger -Metric $metric -Red 9000 -RedIntervalSeconds 3000
-EntityType "VirtualMachine" -MetricAlarmOperator Above
```

- 5 Create a state-based alarm trigger.

```
$stateTrigger = New-AlarmTrigger -StatePath "runtime.powerState" -Value "poweredOff"
-EntityStatus Red -EntityType "VirtualMachine" -StateAlarmOperator Equal
```

Remove Alarm Definitions and Actions

In some cases, you might want to remove obsolete alarm definitions and actions.

Prerequisites

Verify that you are connected to a vCenter Server system.

Procedure

- 1 Remove all the actions for an alarm definition.

```
Get-AlarmDefinition -Name AlarmDefinition | Get-AlarmAction | Remove-AlarmAction
-Confirm:$false
```

2 Remove an alarm definition.

```
Get-AlarmDefinition -Name AlarmDefinition | Remove-AlarmDefinition
```

Create and Modify Advanced Settings for a Cluster

You can customize the behavior of a cluster on a vCenter Server system by creating and modifying custom advanced settings for it.

Prerequisites

Verify that you are connected to a vCenter Server system.

Procedure

1 Create a new cluster named *Cluster*.

```
$cluster = New-Cluster -Name Cluster -Location (Get-Datacenter Datacenter)
```

2 Create two advanced settings for the new cluster.

```
$setting1 = New-AdvancedSetting -Type "ClusterHA" -Entity $cluster -Name
'das.defaultfailoverhost' -Value '192.168.10.1'
$setting2 = New-AdvancedSetting -Type "ClusterHA" -Entity $cluster -Name
'das.isolationaddress' -Value '192.168.10.2'
```

3 Modify the value of the advanced setting stored in the *\$setting2* variable.

```
Get-AdvancedSetting -Entity $cluster -Name 'das.isolationaddress' | Set-AdvancedSetting
-Value '192.168.10.3' -Confirm:$false
```

4 Create another advanced setting.

```
New-AdvancedSetting -Entity $cluster -Name 'das.allowNetwork[Service Console]' -Value
$true -Type 'ClusterHA'
```

5 Get the *Service Console* setting and store it in a variable.

```
$setting3 = Get-AdvancedSetting -Entity $entity -Name 'das.allowNetwork `[Service Console`']'
```

The ``` character is used to escape the wildcard characters `[` and `]` in the advanced setting name.

Modify the vCenter Server Email Configuration

You can modify the email configuration settings of a vCenter Server.

Prerequisites

Verify that you are connected to a vCenter Server system.

Procedure

- 1 View the current email configuration settings of the vCenter Server from the *\$srv* variable.

```
Get-AdvancedSetting -Entity $srv -Name mail.*
```

- 2 Update the SMTP server name and port.

```
Get-AdvancedSetting -Entity $srv -Name mail.smtp.server | Set-AdvancedSetting -Value smtp.vmware.com
Get-AdvancedSetting -Entity $srv -Name mail.smtp.port | Set-AdvancedSetting -Value 25
```

Modify the vCenter Server SNMP Configuration

To use SNMP, you must first configure the SNMP settings of the vCenter Server.

Prerequisites

Verify that you are connected to a vCenter Server system.

Procedure

- 1 View the current SNMP configuration settings of the vCenter Server from the *\$srv* variable.

```
Get-AdvancedSetting -Entity $srv -Name snmp.*
```

- 2 Modify the SNMP receiver data.

```
Get-AdvancedSetting -Entity $srv -Name snmp.receiver.2.community | Set-AdvancedSetting -Value public
Get-AdvancedSetting -Entity $srv -Name snmp.receiver.2.enabled | Set-AdvancedSetting -Value $true
Get-AdvancedSetting -Entity $srv -Name snmp.receiver.2.name | Set-AdvancedSetting -Value 192.168.1.10
```

Results

Now you can use SNMP with vCenter Server.

Use EsxTop to Get Information on the Virtual CPUs of a Virtual Machine

You can use the `Get-EsxTop` cmdlet to retrieve real-time data for troubleshooting performance problems.

Prerequisites

Verify that you are connected to a server that runs ESX 4.1, vCenter Server 5.0 or later.

Procedure

- 1 Get the group to which the virtual machine belongs and save it as a variable.

```
$group = Get-EsxTop -CounterName SchedGroup | where {$_.VMName -eq $vm.Name}
```

- 2 Get the IDs of all virtual CPUs of the virtual machine and store them in an array.

```
$gr = Get-EsxTop -TopologyInfo -Topology SchedGroup | %{$_.Entries} | where
{$group.GroupID -contains $_.GroupId} $group.GroupID
$cpuIds = @()
$gr.CpuClient | %{$cpuIds += $_.CPUClientID}
```

- 3 Get the CPU statistics for the virtual machine.

```
$cpuStats = Get-EsxTop -CounterName VCPU | where {$cpuIds -contains $_.VCPUID}
```

- 4 Calculate the used and ready for use percentage by using the `UsedTimeInUsec` and `ReadyTimeInUsec` stats.

```
$result = @()
$cpuStats | %{ `
$row = "" | select VCPUID, Used, Ready; `
$row.VCPUID = $_.VCPUID; `
$row.Used = [math]::Round(((double)$_.UsedTimeInUsec/[double]$_.UpTimeInUsec)*100, 2); `
$row.Ready = [math]::Round(((double)$_.ReadyTimeInUsec/[double]$_.UpTimeInUsec)*100, 2); `
$result += $row
}
```

- 5 View the used and ready for use percentage for each virtual CPU of the virtual machine.

```
$result | Format-Table -AutoSize
```

Filter vSphere Objects with Get-View

You can use the `Get-View` cmdlet to filter vSphere objects before performing various actions on them.

The filter parameter is a `HashTable` object containing one or more pairs of filter criteria. Each of the criteria consists of a property path and a value that represents a regular expression pattern used to match the property.

Prerequisites

Verify that you are connected to a vCenter Server system.

Procedure

- 1 Create a filter by the power state and the guest operating system name of the virtual machines.

```
$filter = @{"Runtime.PowerState" = "poweredOn"; "Config.GuestFullName" = "Windows XP"}
```

- 2 Get a list of the virtual machines by using the created filter and call the `ShutdownGuest` method for each virtual machine in the list.

```
Get-View -ViewType "VirtualMachine" -Filter $filter | foreach{$_}.ShutdownGuest()
```

Results

The filter gets a list of the powered-on virtual machines whose guest OS names contain the string `Windows XP`. The `Get-View` cmdlet then initiates shutdown for each guest operating system in the list.

Populate a View Object with Get-View

To save time and efforts, you can use `Get-View` to retrieve PowerCLI views from previously retrieved view objects.

Prerequisites

Verify that you are connected to a vCenter Server system.

Procedure

- 1 Get a view of the `VM2` virtual machine by name.

```
$vm2 = Get-View -ViewType VirtualMachine -Filter @{"Name" = "VM2"}
```

- 2 Populate the `$vmhostView` object.

```
$vmhostView = Get-View -Id $vm2.Runtime.Host
```

- 3 Retrieve the runtime information for the `$vmhostView` object.

```
$vmhostView.Summary.Runtime
```

Update the State of a Server-Side Object

You can use the `Get-View` cmdlet to update server-side objects.

Prerequisites

Verify that you are connected to a vCenter Server system.

Procedure

- 1 Get the *VM2* virtual machine by name.

```
$vm2 = Get-View -ViewType VirtualMachine -Filter @{"Name" = "VM2"}
$vmhostView = Get-View -Id $vm2.Runtime.Host
```

- 2 View the current power state.

```
$vm2.Runtime.PowerState
```

- 3 Power off the virtual machine.

```
If ($vm2.Runtime.PowerState -ne "PoweredOn") {
    $vm.PowerOnVM($vm2.Runtime.Host)
} else {
    $vm2.PowerOffVM()
}
```

- 4 View the value of the *\$vm2* power state.

```
$vm2.Runtime.PowerState
```

The power state is not updated yet because the virtual machine property values are not updated automatically.

- 5 Update the view object.

```
$vm2.UpdateViewData()
```

- 6 Obtain the actual power state of the virtual machine.

```
$vm2.Runtime.PowerState
```

Reboot a Host with Get-View

You can reboot a host by using its corresponding view object.

Prerequisites

Verify that you are connected to a vCenter Server system.

Procedure

- 1 Use the `Get-VMHost` cmdlet to get a host by its name, and pass the result to the `Get-View` cmdlet to get the corresponding view object.

```
$vmhostView = Get-VMHost -Name Host | Get-View
```

- 2 Call the `reboot` method of the host view object to reboot the host.

```
$vmhostView.RebootHost()
```

Modify the CPU Levels of a Virtual Machine with Get-View and Get-VIObjectByVIView

You can modify the CPU levels of a virtual machine using a combination of the `Get-View` and `Get-VIObjectByVIView` cmdlets.

Prerequisites

Verify that you are connected to a vCenter Server system.

Procedure

- 1 Get the `VM2` virtual machine, shut it down, and pass it to the `Get-View` cmdlet to view the virtual machine view object.

```
$vmView = Get-VM VM2 | Stop-VM | Get-View
```

- 2 Create a `VirtualMachineConfigSpec` object to modify the virtual machine CPU levels and call the `ReconfigVM` method of the virtual machine view managed object.

```
$spec = New-Object VMware.Vim.VirtualMachineConfigSpec;
$spec.CPUAllocation = New-Object VMware.Vim.ResourceAllocationInfo;
$spec.CpuAllocation.Shares = New-Object VMware.Vim.SharesInfo;
$spec.CpuAllocation.Shares.Level = "normal";
$spec.CpuAllocation.Limit = -1;
$vmView .ReconfigVM_Task($spec)
```

- 3 Get the virtual machine object by using the `Get-VIObjectByVIView` cmdlet and start the virtual machine.

```
$vm = Get-VIObjectByVIView $vmView | Start-VM
```

Browse the Default Inventory Drive

You can browse the default inventory drive and view its contents.

Note For more information about the Inventory Provider and the default inventory drive, see [Using the PowerCLI Inventory Provider](#).

Prerequisites

Verify that you are connected to a vCenter Server system.

Procedure

- 1 Navigate to the `vi` inventory drive.

```
cd vi:
```


- 2 View the drive content.

```
dir
```

`dir` is an alias of the `Get-ChildItem` cmdlet.

Create a New Custom Inventory Drive

In addition to the default drive, you can create new custom inventory drives by using the `New-PSDrive` cmdlet.

Note An alternative to creating an inventory drive is to map an existing inventory path. For example, run: `New-PSDrive -Name myVi -PSProvider VimInventory -Root "vi:\Folder01\Datacenter01"`.

Prerequisites

Verify that you are connected to a vCenter Server system.

Procedure

- 1 Get the root folder of the server.

```
$root = Get-Folder -NoRecursion
```

- 2 Create a PowerShell drive named `myVi` in the server root folder.

```
New-PSDrive -Location $root -Name myVi -PSProvider VimInventory -Root '\'
```

Note You can use the `New-InventoryDrive` cmdlet, which is an alias of `New-PSDrive`. This cmdlet creates a new inventory drive using the `Name` and `Datastore` parameters. For example: `Get-Folder -NoRecursion | New-VIInventoryDrive -Name myVi`.

Manage Inventory Objects Through Inventory Drives

You can use the PowerCLI Inventory Provider to browse, modify, and remove inventory objects from inventory drives.

Prerequisites

Verify that you are connected to a vCenter Server system.

Procedure

- 1 Navigate to a host in your server inventory by running the `cd` command with the full path to the host.

```
cd Folder01\DataCenter01\host\Web\Host01
```

- 2 View the content of the host using the `ls` command.

```
ls
```

`ls` is the UNIX style alias of the `Get-ChildItem` cmdlet.

This command returns the virtual machines and the root resource pool of the host.

- 3 View only the virtual machines on the host.

```
Get-VM
```

When called within the inventory drive, `Get-VM` gets a list only of the virtual machines on the current drive location.

- 4 Delete a virtual machine named `VM1`.

```
del VM1
```

- 5 Rename a virtual machine, for example, from `VM1New` to `VM1`.

```
ren VM1New VM1
```

- 6 Start all virtual machines with names that start with `VM`.

```
dir VM* | Start-VM
```

Browse the Default Datastore Drives

You can use the PowerCLI Datastore Provider to browse the default datastore drives: `vmstore` and `vmstores`.

Note For more information about default datastore drives, see [Using the PowerCLI Datastore Provider](#).

Prerequisites

Verify that you are connected to a vCenter Server system.

Procedure

- 1 Navigate to the `vmstore` drive.

```
cd vmstore:
```

- 2 View the drive content.

```
dir
```

Create a New Custom Datastore Drive

You can use the PowerCLI Datastore Provider to create custom datastore drives.

Prerequisites

Verify that you are connected to a vCenter Server system.

Procedure

- 1 Get a datastore by its name and assign it to the *\$datastore* variable.

```
$datastore = Get-Datastore Storage1
```

- 2 Create a new PowerShell drive *ds:* in *\$datastore*.

```
New-PSDrive -Location $datastore -Name ds -PSProvider VimDatastore -Root '\'
```

Note You can use the `New-PSDrive` cmdlet, which is an alias of `New-DatastoreDrive`. It creates a new datastore drive using the `Name` and `Datastore` parameters. For example:

```
Get-Datastore Storage1 | New-DatastoreDrive -Name ds.
```

Manage Datastores Through Datastore Drives

You can use the PowerCLI Datastore Provider to browse datastores from datastore drives.

Prerequisites

Verify that you are connected to a vCenter Server system.

Procedure

- 1 Navigate to a folder on the *ds:* drive.

```
cd VirtualMachines\XPVirtualMachine
```

- 2 View the files of the folder by running the `ls` command.

```
ls
```

`ls` is the UNIX style alias of the `Get-ChildItem` cmdlet.

- 3 Rename a file by running the `Rename-Item` cmdlet or its alias `ren`.

For example, to change the name of the `vmware-3.log` file to `vmware-3old.log`, run:

```
ren vmware-3.log vmware-3old.log
```

All file operations apply only on files in the current folder.

- 4 Delete a file by running the `Remove-Item` cmdlet or its alias `del`.

For example, to remove the `vmware-3old.log` file from the `XPVirtualMachine` folder, run:

```
del ds:\VirtualMachines\XPVirtualMachine\vmware-2.log
```

- 5 Copy a file by running the `Copy-Item` cmdlet or its alias `copy`.

```
copy ds:\VirtualMachines\XPVirtualMachine\vmware-3old.log ds:\VirtualMachines\vmware-3.log
```

- 6 Copy a file to another datastore by running the `Copy-Item` cmdlet or its alias `copy`.

```
copy ds:\Datacenter01\Datastore01\XPVirtualMachine\vmware-1.log
ds:\Datacenter01\Datastore02\XPVirtualMachine02\vmware.log
```

- 7 Create a new folder by running the `New-Item` cmdlet or its alias `mkdir`.

```
mkdir -Path ds:\VirtualMachines -Name Folder01 -Type Folder
```

- 8 Download a file from the datastore drive to the local machine by running the `Copy-DatastoreItem` cmdlet.

```
Copy-DatastoreItem ds:\VirtualMachines\XPVirtualMachine\vmware-3.log C:\Temp\vmware-3.log
```

- 9 Upload a file from the local machine by running the `Copy-DatastoreItem` cmdlet.

```
Copy-DatastoreItem C:\Temp\vmware-3.log
ds:\VirtualMachines\XPVirtualMachine\vmware-3new.log
```

Modify the Timeout Setting for Web Tasks

To avoid unexpected timeouts, you can run `Set-PowerCLIConfiguration` to modify the PowerCLI settings for long-running Web tasks.

Prerequisites

Verify that you are connected to a vCenter Server system.

Procedure

- 1 (Optional) Learn more about what settings you can configure with `Set-PowerCLIConfiguration`.

```
Get-Help Set-PowerCLIConfiguration
```

- 2 Store the value of the timeout setting for the current session in the `$initialTimeout` variable.

```
$initialTimeout = (Get-PowerCLIConfiguration -Scope Session).WebOperationTimeoutSeconds
```

- 3 Set the timeout setting for the current session to 30 minutes.

```
Set-PowerCLIConfiguration -Scope Session -WebOperationTimeoutSeconds 1800
```

- 4 Run your Web task.

- You can run an `esxcli` command to install a software profile.

```
$vmHost = Get-VMHost "vmHostIp"
$esxcli = Get-EsxCli -VMHost $vmHost -V2
$arguments = $esxcli.software.profile.install.CreateArgs()
$arguments.depot = "http://mysite.com/publish/proj/index.xml"
$arguments.profile = "proj-version"
$esxcli.software.profile.install.Invoke($arguments)
```

- Alternatively, you can directly specify the arguments hash table in-line.

```
$vmHost = Get-VMHost "vmHostIp"
$esxcli = Get-EsxCli -VMHost $vmHost -V2
$esxcli.software.profile.install.Invoke(@{depot="http://mysite.com/publish/proj/
index.xml"; profile="proj-version"})
```

Note The two examples use the ESXCLI V2 interface of PowerCLI.

- 5 Revert the timeout setting for the current session to the initial value.

```
Set-PowerCLIConfiguration -Scope Session -WebOperationTimeoutSeconds $initialTimeout
```

Using Tags

You can assign tags to different types of objects, such as virtual machines, resource pools, datastores, and vSphere distributed switches. You can use tags to retrieve a specific group of objects.

Retrieve a Tag and Save It into a Variable

You can retrieve existing tags defined in vSphere and save a specific tag into a variable.

Prerequisites

Verify that you are connected to a vCenter Server system.

Procedure

- 1 Get the tag named *MyTag*.

```
Get-Tag -Name 'MyTag'
```

- 2 Save the tag into a variable.

```
$tag = Get-Tag -Name 'MyTag'
```

Retrieve a Tag Category and Save It into a Variable

You can retrieve existing tag categories defined in vSphere and save a specific tag category into a variable.

Prerequisites

Verify that you are connected to a vCenter Server system.

Procedure

- 1 Get the tag category named *MyTagCategory*.

```
Get-TagCategory -Name 'MyTagCategory'
```

- 2 Save the tag category into a variable.

```
$tagCategory = Get-TagCategory -Name 'MyTagCategory'
```

Create a Tag Category and a Tag

You can create a tag category and add a new tag in that category.

Prerequisites

Verify that you are connected to a vCenter Server system.

Procedure

- 1 Create a tag category named *Department*.

```
$departmentTagCategory = New-TagCategory -Name 'Department'
```

- 2 Create a new tag named *SalesDpt* in the *Department* category.

```
$salesDptTag = New-Tag -Name 'SalesDpt' -Category $departmentTagCategory
```

Assign a Tag to Virtual Machines

You can assign a tag to a group of virtual machines. For example, you can assign a custom tag to all virtual machines that belong to a specific department in your organization.

Prerequisites

Verify that you are connected to a vCenter Server system.

Procedure

- 1 Get the virtual machines of a department in your organization.

```
$vms = Get-VM sales-dpt*
```

- 2 Assign the custom tag to the group of virtual machines.

```
New-TagAssignment -Tag $salesDptTag -Entity $vms
```

Retrieve Objects by Tag

You can retrieve all objects that have a specific tag assigned to them.

Prerequisites

Verify that you are connected to a vCenter Server system.

Procedure

- ◆ Get all virtual machines tagged with the *salesDptTag* tag.

```
Get-VM -Tag 'salesDptTag'
```

Note You can only specify a tag filter parameter for the `VM`, `VMHost`, `Datastore`, and `VirtualPortGroup` object types.

Generate Tags Automatically by Using a Script

You can use a script to generate tags automatically. For example, you can create a virtual machine owner tag for each user account in a domain.

You must use the `Get-VIAccount` cmdlet to retrieve user accounts. For more information, see the documentation of the cmdlet.

Prerequisites

- Verify that you are connected to a vCenter Server system.
- Verify that the user accounts and the vCenter Server system are in the same domain.

Procedure

- 1 Create a new tag category and specify that tags in this category can only be assigned to entities of type `VirtualMachine`.

```
$ownerTagCategory = New-TagCategory -Name Owner -EntityType VirtualMachine
```

Note If you do not specify an entity type, tags from this category can be assigned to all entity types.

- 2 Retrieve all domain user accounts and save them in a variable.

```
$accounts = Get-VIAccount -User -Domain 'DomainName' -Category | select -ExpandProperty Id
```

- 3 Create a tag for each user account.

```
$accounts | foreach { New-Tag -Category $ownerTagCategory -Name $_ }
```

- 4 Retrieve a specific tag from the *Owner* category, so that you can later assign it to a specific virtual machine.

```
$ownerTag = Get-Tag -Category $ownerTagCategory -Name 'John_Smith'
```

Add an Entity Type to a Tag Category

You can extend the list of entity types associated with a tag category.

Prerequisites

Verify that you are connected to a vCenter Server system.

Procedure

- ◆ Add the *vApp* entity type to the *ownerTagCategory* tag category.

```
$ownerTagCategory | Set-TagCategory -AddEntityType vApp
```

Retrieve Tag Assignments

You can retrieve tag assignments by using category and entity filters.

Prerequisites

Verify that you are connected to a vCenter Server system.

Procedure

- 1 Retrieve all virtual machines that have a tag from the *ownerTagCategory* tag category assigned to them.

```
Get-TagAssignment -Category $ownerTagCategory
```

- 2 Retrieve the owner of the *MyVM* virtual machine.

```
Get-TagAssignment -Category $ownerTagCategory -Entity 'MyVM'
```

Network Management with vSphere Distributed Switches

The cmdlets provided in the `VMware.VimAutomation.VDS` module let you manage networking with vSphere distributed switches and port groups.

Create a Distributed Switch and Configure Networking

A vSphere distributed switch lets you handle networking traffic for all associated hosts in a data center. After you create a new vSphere distributed switch in PowerCLI, you can add hosts and connect virtual machines to it.

Prerequisites

Verify that you are connected to a vCenter Server system.

Procedure

- 1 Get the data center where you want to create the vSphere distributed switch.

```
$myDatacenter = Get-Datacenter -Name "MyDatacenter"
```

- 2 Get all hosts in your data center.

```
$vmHosts = $myDatacenter | Get-VMHost
```

- 3 Create a new vSphere distributed switch.

```
$myVDSwitch = New-VDSwitch -Name "MyVDSwitch" -Location $myDatacenter
```

The distributed switch is created with no port groups.

- 4 Add the hosts in your data center to the distributed switch.

```
Add-VDSwitchVMHost -VDSwitch $myVDSwitch -VMHost $vmHosts
```

- 5 Get a physical network adapter from your hosts.

```
$hostsPhysicalNic = $vmHosts | Get-VMHostNetworkAdapter -Name "vmnic2"
```

- 6 Add the physical network adapter to the distributed switch that you created.

```
Add-VDSwitchPhysicalNetworkAdapter -VMHostNetworkAdapter $hostsPhysicalNic  
-DistributedSwitch $myVDSwitch
```

- 7 Create a new distributed port group with 1000 ports and add it to the distributed switch.

```
$myVDPortGroup = New-VDPortgroup -Name "MyVMsPortGroup" -VDSwitch $myVDSwitch -NumPorts  
1000
```

- 8 Connect all virtual machines running on the hosts in your data center to the distributed port group.

```
$vmHosts | Get-VM | Get-NetworkAdapter | Set-NetworkAdapter -PortGroup $myVDPortGroup
```

What to do next

Adjust the settings of the distributed switch. See [Configure a Distributed Switch](#).

Configure a Distributed Switch

Based on your networking requirements, you can adjust the settings of a newly created or an existing distributed switch.

Prerequisites

Verify that you are connected to a vCenter Server system.

Procedure

- ◆ Modify the maximum MTU size setting for a distributed switch.

```
Get-VDSwitch -Name 'MyVDSwitch' | Set-VDSwitch -Mtu 2000
```

- ◆ Modify the number of uplink ports on a distributed switch.

```
Get-VDSwitch -Name 'MyVDSwitch' | Set-VDSwitch -NumUplinkPorts 4
```

- ◆ Modify the maximum number of ports on a distributed switch.

```
Get-VDSwitch -Name 'MyVDSwitch' | Set-VDSwitch -MaxPorts 1000
```

- ◆ Modify the discovery protocol settings on a vSphere distributed switch.

```
Get-VDSwitch -Name 'MyVDSwitch' | Set-VDSwitch -LinkDiscoveryProtocol LLDP  
-LinkDiscoveryProtocolOperation Both
```

Migrate Virtual Machine Networking Configuration from a vSphere Standard Switch to a vSphere Distributed Switch

To manage virtual machine networks on a data center level, you might need to migrate existing networks from vSphere standard switches to vSphere distributed switches.

Prerequisites

Verify that you are connected to a vCenter Server system.

Procedure

- 1 Get the source vSphere standard switch from which you want to migrate the virtual machine networking.

```
$virtualSwitch = Get-VirtualSwitch -Name 'MyVirtualSwitch'
```

- 2 Get the source standard port group to which the virtual machines are connected.

```
$vmsPortGroup = $virtualSwitch | Get-VirtualPortGroup -Name 'VM Network'
```

- 3 Get the target vSphere distributed switch to which you want to migrate the virtual machine networking.

```
$vdSwitch = Get-VDSwitch -Name 'MyTargetVDSwitch'
```

- 4 Get the target port group to which you want to connect the virtual machines.

```
$vdPortGroup = Get-VDPortGroup -VDSwitch $vdSwitch -Name 'DPortGroup'
```

- 5 Get the virtual machine network adapters connected to the source port group.

```
$vmsNetworkAdapters = Get-VM -RelatedObject $vmsPortGroup | Get-NetworkAdapter | where  
{ $_.NetworkName -eq $vmsPortGroup.Name }
```

- 6 Disconnect the retrieved network adapters from the standard port group and connect them to the distributed port group.

```
Set-NetworkAdapter -NetworkAdapter $vmsNetworkAdapters -PortGroup $vdPortGroup
```

Migrate Physical and Virtual NICs to a vSphere Standard Switch

You can migrate both physical and virtual network adapters to a vSphere standard switch simultaneously.

Prerequisites

Verify that you are connected to a vCenter Server system.

Procedure

- 1 Get the physical network adapters that you want to migrate.

```
$pNics = Get-VMHostNetworkAdapter -VMHost $vmhost -Physical
```

- 2 Get the virtual network adapters that you want to migrate.

```
$vNicManagement = Get-VMHostNetworkAdapter -VMHost $vmhost -Name vmk0  
$vNicvMotion = Get-VMHostNetworkAdapter -VMHost $vmhost -Name vmk1
```

- 3 Get the vSphere standard switch to which you want to migrate the network adapters.

```
$vSwitch = Get-VirtualSwitch -VMHost $vmhost -Name vSwitch0
```

- 4 Migrate all network adapters to the vSphere standard switch.

```
Add-VirtualSwitchPhysicalNetworkAdapter -VirtualSwitch $vSwitch -VMHostPhysicalNic $pNics  
-VMHostVirtualNic $vNicManagement,$vNicvMotion
```

Migrate Physical and Virtual NICs to a vSphere Distributed Switch

You can migrate both physical and virtual network adapters to a vSphere distributed switch simultaneously.

Prerequisites

Verify that you are connected to a vCenter Server system.

Procedure

- 1 Get the physical network adapters that you want to migrate.

```
$pNics = Get-VMHostNetworkAdapter -VMHost $vmhost -Physical
```

- 2 Get the virtual network adapters that you want to migrate.

```
$vNicManagement = Get-VMHostNetworkAdapter -VMHost $vmhost -Name vmk0
$vNicvMotion = Get-VMHostNetworkAdapter -VMHost $vmhost -Name vmk1
```

- 3 Get the port groups corresponding to the virtual network adapters that you want to migrate to the vSphere distributed switch.

```
$vdPortgroupManagement = Get-VDPortgroup -VDSwitch $vds -Name 'Management Network'
$vdPortgroupvMotion = Get-VDPortgroup -VDSwitch $vds -Name 'vMotion Network'
```

- 4 Migrate all network adapters to the vSphere distributed switch.

```
Add-VDSwitchPhysicalNetworkAdapter -DistributedSwitch $vds -VMHostPhysicalNic
$pNics -VMHostVirtualNic $vNicManagement,$vNicvMotion -VirtualNicPortGroup
$vdPortgroupManagement, $vdPortgroupvMotion
```

Results

You migrated the *\$vNicManagement* network adapter to the Management Network port group and the *\$vNicvMotion* network adapter to the vMotion Network port group.

Configure the Traffic Shaping Policy

You can modify the traffic shaping policy of a port group to limit the bandwidth of the incoming traffic and ensure that enough bandwidth is available for other port groups on the same vSphere distributed switch.

Prerequisites

Verify that you are connected to a vCenter Server system.

Procedure

- 1 Get the current traffic shaping policy of the port group.

```
$policy = Get-VDTrafficShapingPolicy -Direction In -VDPortGroup $myVDPortGroup
```

- 2 Set the peak bandwidth to 100 Mbps.

```
Set-VDTrafficShapingPolicy -Policy $policy -PeakBandwidth 104857600
```

Configure the Security Policy

You can modify the security policy of a port group to enable promiscuous mode, which allows monitoring of the traffic generated by virtual machines.

Prerequisites

Verify that you are connected to a vCenter Server system.

Procedure

- 1 Get the current security policy of the port group.

```
$policy = Get-VDSecurityPolicy -VDPortGroup $myVDPortGroup
```

- 2 Enable promiscuous mode for the port group.

```
Set-VDSecurityPolicy $policy -AllowPromiscuous $true
```

Create a Local Content Library

With PowerCLI, you can create a local content library.

Prerequisites

Verify that you are connected to a vCenter Server system.

Procedure

- 1 Get the datastore that you want to use to store files for library items in this library.

```
$datastore = Get-Datastore myDatastore
```

- 2 Create the *My Local Content Library* published local content library that uses the *myDatastore* datastore.

```
New-ContentLibrary -Name 'My Local Content Library' -Description 'Local content library description.' -Datastore $Datastore1 -Published
```

Create a Subscribed Content Library

With PowerCLI, you can create a subscribed content library.

Prerequisites

Verify that you are connected to a vCenter Server system.

Procedure

- ◆ Create the *My Subscribed Content Library* subscribed content library that uses the *myDatastore* datastore.

```
New-ContentLibrary -Name 'My Subscribed Content Library' -Description 'Subscribed content library description.' -Datastore $Datastore1 -SubscriptionUrl $Url
```

Create a Content Library Item

With PowerCLI, you can create content library items.

Prerequisites

Verify that you are connected to a vCenter Server system.

Procedure

- 1 Get the files that you want to upload to the content library.

```
$files = Get-ChildItem -File
```

- 2 Get the content library that you want to upload.

```
$contentLibrary = Get-ContentLibrary -name 'MyContentLibrary'
```

- 3 Create a content library item.

```
New-ContentLibraryItem -ContentLibrary $contentLibrary -name 'New item' -Files $files
```

Create a Virtual Machine from a Content Library Item

You can deploy a virtual machine from a content library template.

Note VMware PowerCLI cannot distinguish between OVF content library items of type virtual machine template and vApp template. As a result, `New-VM` creates a vApp if you specify a vApp template from the content library by using the `ContentLibraryItem` parameter of the cmdlet. If this happens, `New-VM` returns an error, notifying that the cmdlet produced an inventory item of the wrong type. Avoid creating vApps by using the `New-VM` cmdlet, as this behavior will be deprecated in future releases.

Prerequisites

- Verify that you are connected to a vCenter Server system.
- Verify that you have a content library with virtual machine templates available.

Procedure

- 1 Get the virtual machine host.

```
$myVMHost = Get-VMHost myVMHost
```

- 2 Create the *MyVM* virtual machine from the *MyVMContentLibrayItemName* content library item.

```
Get-ContentLibraryItem -Name MyVMContentLibrayItemName | New-VM -Name MyVM -VMHost $myVMHost
```

Create a vApp from a Content Library Item

You can deploy a vApp from a content library template.

Note VMware PowerCLI cannot distinguish between OVF content library items of type virtual machine template and vApp template. As a result, `New-VApp` creates a virtual machine if you specify a virtual machine template from the content library by using the `ContentLibraryItem` parameter of the cmdlet. If this happens, `New-VApp` returns an error, notifying that the cmdlet produced an inventory item of the wrong type. Avoid creating virtual machines by using the `New-VApp` cmdlet, as this behavior will be deprecated in future releases.

Prerequisites

- Verify that you are connected to a vCenter Server system.
- Verify that you have a content library with vApp templates available.

Procedure

- 1 Get the virtual machine host.

```
$myVMHost = Get-VMHost myVMHost
```

- 2 Create the *MyVApp* vApp from the *MyVAppContentLibrayItemName* content library item.

```
Get-ContentLibraryItem -Name MyVAppContentLibrayItemName | New-VApp -Name MyVApp -VMHost $myVMHost
```

Export Content Library Item's Files to a Local Machine

With PowerCLI, you can export content library item's files to a local machine.

Prerequisites

Verify that you are connected to a vCenter Server system.

Procedure

- 1 Get a content library item.

```
$item = Get-ContentLibraryItem -Name 'myVM'
```

- 2 Export a content library item to a local machine.

```
Export-ContentLibraryItem -ContentLibraryItem $item -Destination ./myVM-files
```

Create a New VM-VM DRS Rule

You can create a VM-VM DRS affinity rule within a cluster.

Prerequisites

- Verify that you are connected to a vCenter Server system.
- Verify that virtual machines and hosts exist within a cluster with enabled DRS in the vCenter Server environment.

Procedure

- 1 Get the virtual machines for the VM-VM DRS rule.

```
$affinityVMs = Get-VM "VM1", "VM2"
```

- 2 Get the cluster where you want to create the rule.

```
$cluster = Get-Cluster "MyCluster"
```

- 3 Create the VM-VM DRS rule within the *MyCluster* cluster.

```
New-DrsRule -Cluster $cluster -Name "AffinityRule1" -KeepTogether $true -VM $affinityVMs
```

Create a New VM-VMHost DRS Rule

You can create a VM-VMHost DRS rule within a cluster after creating a VM DRS cluster group and a VMHost DRS cluster group.

Prerequisites

- Verify that you are connected to a vCenter Server system.
- Verify that virtual machines and hosts exist within a cluster with enabled DRS in the vCenter Server environment.

Procedure

- 1 Get the virtual machines for the VM DRS cluster group.

```
$vms = Get-VM "VM1", "VM2"
```

- 2 Get the hosts for the VMHost DRS cluster group.

```
$vmHosts = Get-VMHost "hostname1", "hostname2"
```

- 3 Get the cluster where you want to create the rule.

```
$cluster = Get-Cluster "MyCluster"
```

- 4 Create a VM DRS cluster group.

```
$vmGroup = New-DrsClusterGroup -Name "MyVmsDrsClusterGroup" -VM $vms -Cluster $cluster
```

- 5 Create a VMHost DRS cluster group.

```
$vmHostGroup = New-DrsClusterGroup -Name "MyVmHostsDrsClusterGroup" -VMHost $vmHosts  
-Cluster $cluster
```

- 6 Create the VM-VMHost DRS rule by using the newly created VM DRS cluster group and VMHost DRS cluster group.

```
New-DrsVMHostRule -Name "MyDrsRule" -Cluster $cluster -VMGroup $vmGroup -VMHostGroup  
$vmHostGroup -Type "MustRunOn"
```

Managing Certificates

You can use VMware PowerCLI to manage certificates for both vCenter Server and ESXi no matter if you are using Hybrid, Subordinate CA, or Full Custom Mode.

Starting from VMware PowerCLI 12.4.0, you can do various certificate management operations in your vSphere environment. The certificate management functionality is included in the `VMware.PowerCLI.VCenter` module.

Add a Root Certificate to vCenter Server and ESXi

You can use PowerCLI to add a root certificate or certificate chain to the trusted root store of vCenter Server and to the certificate stores of the connected ESXi hosts.

Prerequisites

- Verify that you are connected to a vCenter Server system.
- Verify that the root certificate or certificate chain you want to add is available on your machine.

Procedure

- 1 (Optional) List the trusted root certificates on your vCenter Server system and the connected ESXi hosts.

```
Get-VITrustedCertificate
```

- 2 Create a variable with the root certificate or certificate chain you want to add.

```
$trustedCertChain = Get-Content "C:\Users\jdoe\Downloads\ca-chain.cert.pem" -Raw
```

- 3 Add the root certificate or certificate chain to vCenter Server and to the connected ESXi hosts

```
Add-VITrustedCertificate -PemCertificateOrChain $trustedCertChain
```

- 4 (Optional) Verify that your new root certificate or certificate chain has been added to the trusted certificate stores.

```
Get-VITrustedCertificate
```

Change the Machine SSL Certificate of vCenter Server

You can use PowerCLI to change the Machine SSL certificate of a vCenter Server system. For a custom certificate, you must generate a certificate signing request (CSR) and send it to the certificate authority (CA) of your choice.

Prerequisites

- Verify that you are connected to a vCenter Server system.
- Verify that the root certificate of the CA you are going to use is added to the trusted root store of vCenter Server.

Procedure

- 1 (Optional) Retrieve the current Machine SSL certificate of the vCenter Server system.

```
Get-VIMachineCertificate -VCenterOnly
```

- 2 Generate a CSR.

```
$csrParams = @{
    Country="US"
    Email="jdoe@vmware.com"
    Locality="San Francisco"
    Organization="My Company"
    OrganizationUnit="PowerCLI"
    StateOrProvince="California"
}
$csr = New-VIMachineCertificateSigningRequest @csrParams
```

3 Save the CSR to your system.

```
$csr.CertificateRequestPEM | Out-File "C:\Users\jdoe\Downloads\vc.csr.pem" -Force
```

4 Send the CSR to the CA of your choice.

5 Save the issued custom certificate to your system.

6 Set the new custom certificate to the vCenter Server system.

```
$vcCert = Get-Content "C:\Users\jdoe\Downloads\vc.cert.jdoe.pem" -Raw
Set-VMachineCertificate -PemCertificate $vcCert
```

Important The change of the Machine SSL certificate triggers a restart of vCenter Server. Wait for the system to reboot and login when available.

Change the Machine SSL Certificate of an ESXi Host

You can use PowerCLI to change the Machine SSL certificates of one or more ESXi hosts in your vSphere environment.

Prerequisites

- Verify that the root certificate of the CA you are going to use is added to the trusted root store of vCenter Server and to the connected ESXi hosts.

Procedure

1 Connect to the vCenter Server system.

```
$vCenterConnection = Connect-VIServer vcl.example.com `
    -User 'My User' `
    -Password 'My Password'
```

2 In the vCenter Server system, retrieve the setting for the ESXi host certificate management mode.

```
$certModeSetting = Get-AdvancedSetting "vpxd.certmgmt.mode" -Entity $vCenterConnection
$certModeSetting.Value
```

3 Change the setting to *custom*.

```
Set-AdvancedSetting $certModeSetting -Value "custom"
```

4 For the new certificate management mode to take effect, reboot your vCenter Server system.

5 Connect to the vCenter Server system.

```
$vCenterConnection = Connect-VIServer vcl.example.com `
    -User 'My User' `
    -Password 'My Password'
```

- 6 Set the ESXi host you want to manage to *Maintenance* mode.

```
$vmhost = Get-VMHost 'MyESXiHost' `
Set-VMHost -VMHost $vmhost -State Maintenance
```

- 7 Generate a certificate signing request (CSR) for the ESXi host.

```
$esxRequest = New-VIMachineCertificateSigningRequest `
-VMHost $vmhost `
-Country "US" `
-Locality "San Francisco" `
-Organization "My Company" `
-OrganizationUnit "PowerCLI" `
-StateOrProvince "California" `
-CommonName <ESXi host's FQDN> or <ESXi host's IP address>
```

Note For `CommonName`, you must use either the ESXi host's FQDN or IP address. The common name must match the identifier you use to add the host to the vCenter Server system.

- 8 Save the CSR to your system.

```
$esxRequest.CertificateRequestPEM | Out-File "C:\Users\jdoe\Downloads\esx.csr.pem" -Force
```

- 9 Send the CSR to the CA of your choice.
 10 Save the issued custom certificate to your machine.
 11 Create a variable with your issued custom certificate.

```
$esxCertificatePem = Get-Content "C:\Users\jdoe\downloads\myesxcert.pem" -Raw
```

- 12 Remove the ESXi host from the vCenter Server system.

```
Remove-VMHost $vmhost
```

- 13 Disconnect from the vCenter Server system.

```
Disconnect-VIServer $vCenterConnection
```

- 14 Connect directly to the ESXi host.

```
$esxConnection = Connect-VIServer $vmhost.Name `
-User 'My User' `
-Password 'My Password' `
-Force
```

- 15 Set the custom Machine SSL certificate to the ESXi host.

```
$targetEsxHost = Get-VMHost $vmhost.Name
Set-VIMachineCertificate -PemCertificate $esxCertificatePem -VMHost $targetEsxHost | Out-Null
```

- 16 To apply the change, restart the ESXi host.

```
Restart-VMHost $targetEsxHost
```

- 17 Disconnect from the ESXi host.

```
Disconnect-VIServer $esxConnection
```

- 18 Connect to the vCenter Server system.

```
$vCenterConnection = Connect-VIServer vcl.example.com `
    -User 'My User' `
    -Password 'My Password'
```

- 19 Add the ESXi host to the vCenter Server system.

```
$vmhost = Add-VMHost -Name <ESXi host's FQDN> or <ESXi host's IP address> `
    -Location (Get-Datacenter "My Datacenter") `
    -User "My User" `
    -Password "My Password"
```

- 20 Set the ESXi host to the *Connected* mode.

```
$vmhost = Set-VMHost -VMHost $vmhost -State Connected
```

- 21 (Optional) Verify that the Machine SSL certificate of the ESXi host is changed.

```
Get-VMachineCertificate -VMHost $vmhost
```

Managing the vSphere Automation API with VMware PowerCLI

6

You can manage the vSphere Automation API with PowerCLI by using the vSphere Automation SDK for PowerShell. Starting from VMware PowerCLI version 12.4.0, the vSphere Automation SDK provides low-level cmdlets for all available vSphere Automation API services.

Read the following topics next:

- [Understanding the vSphere Automation SDK for PowerShell](#)
- [Navigating the vSphere Automation SDK for PowerShell](#)
- [Connecting to a vSphere Automation API Server](#)
- [Create a Local User Account in vCenter Server](#)
- [Update the Local Accounts Global Password Policy in vCenter Server](#)
- [Create a Virtual Machine](#)
- [Create Tag Category, Tag, and Tag Association](#)
- [Create Content Library and Content Library Item](#)

Understanding the vSphere Automation SDK for PowerShell

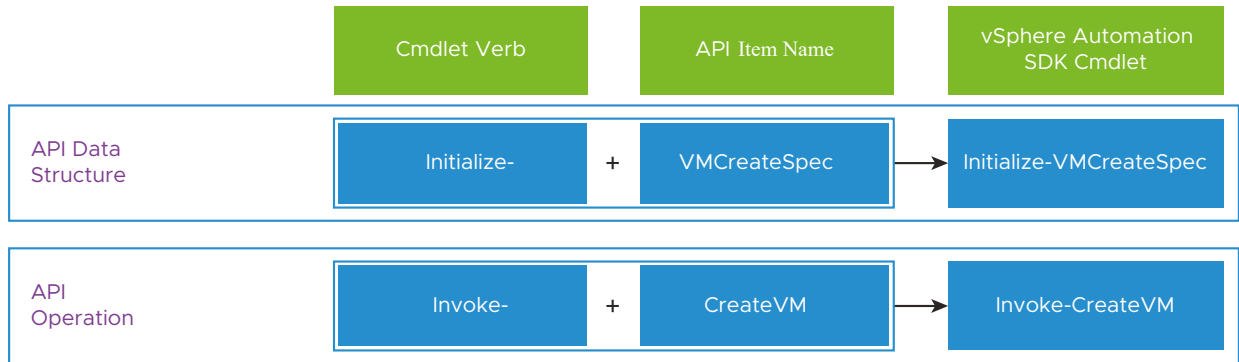
You can manage the vSphere Automation API with VMware PowerCLI by using the vSphere Automation SDK for PowerShell.

Starting from VMware PowerCLI 12.4.0, you can use the vSphere Automation SDK for PowerShell to communicate with all available vSphere Automation REST APIs.

The modules that start with the `VMware.vSphere.Sdk` prefix contain automatically-generated SDK cmdlets that bind directly to the vSphere Automation REST APIs. The modules function on a low level and communicate directly with the vSphere Automation API server.

You can construct vSphere Automation SDK cmdlets by using the `Initialize-` and `Invoke-` cmdlet verbs in combination with the names of the vSphere Automation API data structures or operations.

Figure 6-1. Constructing vSphere Automation SDK Cmdlets



- Use `Initialize-` cmdlets to prepare the data structures, if any, for your API calls. These cmdlets function on the client side and do not communicate with the API server.
- Use `Invoke-` cmdlets to call the vSphere Automation API server and invoke the operations.

Navigating the vSphere Automation SDK for PowerShell

You can navigate the vSphere Automation SDK for PowerShell by using the VMware Developer website, and the `Get-Help` and `Get-Command` cmdlets.

To use the vSphere Automation SDK for PowerShell effectively, you must be familiar with the [vSphere Automation API reference documentation](#).

How to Find a vSphere Automation SDK Cmdlet

To search for vSphere Automation SDK cmdlets:

- In your browser, navigate the vSphere Automation API reference documentation.

The vSphere Automation API reference on VMware Developer features ready-to-use vSphere Automation SDK code samples that you can paste directly into PowerShell. You can find a PowerCLI code snippet for each operation under *PowerCLI Client SDK Example* in the *Code Samples* section.
- In PowerShell, run `Get-Command -Module VMware.Sdk.vSphere.*` to list all available vSphere Automation SDK cmdlets.

How to Get Information About a Cmdlet

To get information about a vSphere Automation SDK cmdlet, run `Get-Help` with the cmdlet name. For example:

```
Get-Help Invoke-CreateVm -full
```

To open the online information in a browser, run `Get-Help` with the cmdlet name and the `Online` parameter. For example:

```
Get-Help Invoke-CreateVm -Online
```

Connecting to a vSphere Automation API Server

You can connect to the vSphere Automation API with VMware PowerCLI through the `Connect-VIServer` cmdlet or through a configuration object.

You can access a vSphere Automation API server in two ways with PowerCLI.

- Through the `Connect-VIServer` cmdlet.
- Through a configuration object.

Connect Through `Connect-VIServer`

Use the `Connect-VIServer` cmdlet to connect to a vCenter Server system and access the vSphere Automation API.

You can have more than one connection to the same server. For more information, see [Managing Default Server Connections in PowerCLI](#).

If your login credentials contain non-alphanumeric characters, you might need to escape them. For more information, see [Processing Non-alphanumeric Characters in PowerCLI](#).

Prerequisites

- If the certificate of the server you want to connect to is not trusted, verify that your PowerCLI invalid server certificate action settings are configured properly. For more information, see [Configuring PowerCLI Response to Untrusted Certificates](#).
- If you use a proxy server for the connection, verify that it is configured properly, so that the connection is kept alive long enough for tasks to finish.

Note If you do not want to use a proxy server for the connection, run `Set-PowerCLIConfiguration -ProxyPolicy NoProxy`.

Procedure

- ◆ Run `Connect-VIServer` with the server name and valid credentials.

```
Connect-VIServer -Server vc3.example.com -User 'MyUser' -Password 'MyPassword'
```

Connect Through a Configuration Object

You can access the vSphere Automation API by connecting to a vCenter Server system through a configuration object.

You can create a configuration object and pass information about the vCenter Server system on which you want to use the vSphere Automation SDK for PowerShell. You can use this connection method if you don't want to use the PowerCLI high-level cmdlets or if you don't have the `VMware.VimAutomation.Core` module installed on your system.

Procedure

- 1 Create a server configuration for a vCenter Server system.

```
$Server = 'vCenter Server address'
$User = 'username'
$Password = Read-Host -AsSecureString -Prompt "Password:"
$serverConfig = New-vSphereServerConfiguration -Server $Server -User $User -Password
$Password -SkipCertificateCheck
```

- 2 Create a vSphere Automation API session.

```
$apiSession = Invoke-CreateSession -WithHttpInfo
```

- 3 Update the server configuration with the created API session key.

```
$serverConfig | Set-vSphereServerConfigurationApiKey -SessionResponse $apiSession
```

What to do next

To learn more about using server configuration objects with the vSphere Automation SDK, see [Managing vCenter Server Connections Through Configuration Objects](#).

Managing vCenter Server Connections Through Configuration Objects

The vSphere Automation SDK for PowerShell features special configuration object cmdlets to manage vCenter Server connections. A configuration object is a client-side PowerShell object that holds information about a server that the `Invoke-` cmdlets can interact with.

Table 6-1. Configuration Object Cmdlets

Cmdlet	Description
<code>New-vSphereServerConfiguration</code>	Creates a vCenter Server configuration object.
<code>Get-vSphereServerConfiguration</code>	Retrieves the existing vCenter Server configuration objects.
<code>Set-vSphereServerConfiguration</code>	Updates a vCenter Server configuration object.
<code>Set-vSphereServerConfigurationApiKey</code>	Updates a vCenter Server configuration object with an API key for authorization.
<code>Remove-vSphereServerConfiguration</code>	Removes a vCenter Server configuration object.

Working with Configuration Objects

You can create multiple vCenter Server configuration objects and manage all of them simultaneously with the `Invoke-` cmdlets. For example, `Invoke-DeleteSession` deletes all existing vSphere Automation API sessions.

If you want to apply the `Invoke-` cmdlets to a specific configuration object, you must specify it in the command. For example,

```
Invoke-ListFolder -Server $serverConfig
```

Example Use Cases with vCenter Server Configuration Objects

- You can create a session and authenticate to a vSphere Automation API server - see [Connect Through a Configuration Object](#).
- You can retrieve the active server configurations.

```
Get-vSphereServerConfiguration
```

- You can remove a specific server configuration.

```
$serverConfig | Remove-vSphereServerConfiguration
```

Create a Local User Account in vCenter Server

You can create a local user account in vCenter Server by using the vSphere Automation SDK for PowerShell.

Prerequisites

Verify that you are connected to a vSphere Automation API server.

Procedure

- 1 Prepare the data structures for the API operation.

```
$LocalAccountsConfig = Initialize-LocalAccountsConfig -Password '$trongPa$$w0rd' -Roles "superAdmin"
$LocalAccountsCreateRequestBody = Initialize-LocalAccountsCreateRequestBody -Username "MyUsername" -Config $LocalAccountsConfig
```

- 2 Invoke the API operation.

```
Invoke-CreateLocalAccounts -LocalAccountsCreateRequestBody $LocalAccountsCreateRequestBody
```

- 3 (Optional) Retrieve the local account you created.

```
Invoke-GetUsernameLocalAccounts -Username "MyUsername"
```

Update the Local Accounts Global Password Policy in vCenter Server

You can update the local accounts global password policy in vCenter Server by using the vSphere Automation SDK for PowerShell.

Prerequisites

Verify that you are connected to a vSphere Automation API server.

Procedure

- 1 Retrieve the current local accounts global password policy.

```
Invoke-GetLocalAccountsGlobalPolicy
```

- 2 Prepare the data structures for the operation.

```
$LocalAccountsPolicyInfo = Initialize-LocalAccountsPolicyInfo -MaxDays 352 -MinDays 1  
-WarnDays 7
```

- 3 Invoke the API operation.

```
Invoke-SetLocalAccountsGlobalPolicy -LocalAccountsPolicyInfo $LocalAccountsPolicyInfo
```

- 4 (Optional) Retrieve the new local accounts global password policy.

```
Invoke-GetLocalAccountsGlobalPolicy
```

Create a Virtual Machine

You can create virtual machines in vCenter Server by using the vSphere Automation SDK for PowerShell.

Prerequisites

Verify that you are connected to a vSphere Automation API server.

Procedure

- 1 Retrieve a target datastore ID.

```
$targetDatastore = Invoke-ListDatastore -Names 'local-0'
```

- 2 Retrieve a target folder ID.

```
$targetFolder = Invoke-ListFolder -Names 'vm'
```

- 3 Retrieve a target host ID.

```
$targetHost = Invoke-ListHost -Names '10.185.41.87'
```

4 Prepare the virtual machine placement specification.

```
$placementSpec = Initialize-VMPlacementSpec -Folder $targetFolder.folder -Datastore
$targetDatastore.datastore -VarHost $targetHost.host
```

5 Prepare the virtual machine create specification.

```
$vmCreateSpec = Initialize-VMCreateSpec -Name 'TestVM' -GuestOS WIN_31 -Placement
$placementSpec
```

6 Invoke the CreateVM API operation.

```
Invoke-CreateVM -VMCreateSpec $vmCreateSpec
```

Create Tag Category, Tag, and Tag Association

You can create a tag category, tag, and tag association by using the vSphere Automation SDK for PowerShell.

Prerequisites

Verify that you are connected to a vSphere Automation API server.

Procedure

1 Create a tag category.

```
$categoryCreateSpec = Initialize-TaggingCategoryCreateSpec -Cardinality 'MULTIPLE'
-AssociableTypes 'VirtualMachine' -Name 'TestCategory' -Description 'TestDescription'
$categoryId = Invoke-CreateCategory -TaggingCategoryCreateSpec $categoryCreateSpec
```

2 Create a tag.

```
$tagCreateSpec = Initialize-TaggingTagCreateSpec -Name 'TestVMTag' -Description 'This is a
test tag for vms' -CategoryId $categoryId
$tag = Invoke-CreateTag -TaggingTagCreateSpec $tagCreateSpec
```

3 Create a tag association.

```
$vmId = Invoke-ListVM -Names 'MyVM'
$vmDynamicId = Initialize-StdDynamicID -Type 'VirtualMachine' -Id $vmId
$tagAssociationBody = Initialize-TaggingTagAssociationAttachRequestBody
-ObjectId $vmDynamicId Invoke-AttachTagIdTagAssociation -TagId $tag
-TaggingTagAssociationAttachRequestBody $tagAssociationBody
```

Create Content Library and Content Library Item

You can create a content library and content library item by using the vSphere Automation SDK for PowerShell.

Prerequisites

Verify that you are connected to a vSphere Automation API server.

Procedure

- 1 Specify a datastore for storage backing.

```
$backingDatastore = Invoke-ListDatastore -Names 'local-0'
```

- 2 Create a content library.

```
$storageBacking = Initialize-LibraryStorageBacking -DatastoreId  
$backingDatastore.datastore -Type 'DATASTORE'  
$libraryModel = Initialize-LibraryModel -Name 'TestLocalContentLibrary' -StorageBackings  
$storageBacking  
$contentLibraryId = Invoke-CreateContentLocalLibrary -LibraryModel $libraryModel
```

- 3 Create a content library item.

```
$libraryItemModel = Initialize-LibraryItemModel -Name 'TestContentLibraryItem' -LibraryId  
$contentLibraryId  
$libraryItemId = Invoke-CreateContentLibraryItem -LibraryItemModel $libraryItemModel
```

Managing vSphere Policy-Based Storage with VMware PowerCLI

7

To help you get started with VMware PowerCLI, this documentation provides a set of sample scripts that illustrate basic and advanced tasks in vSphere policy-based storage management.

Read the following topics next:

- [Create a Tag-Based Storage Policy](#)
- [Create a Capability-Based Storage Policy](#)
- [Associate a Storage Policy with a Virtual Machine and Its Hard Disk](#)
- [Disassociate a Storage Policy Associated with a Virtual Machine and Its Hard Disk](#)
- [Remove a Storage Policy](#)
- [Edit a Storage Policy](#)
- [Export and Import a Storage Policy](#)
- [Create a Virtual Machine in a Datastore Compatible with Storage Policy](#)
- [Create a vSAN Datastore](#)
- [Modify a vSAN Datastore](#)
- [Create a vSAN Stretched Cluster](#)
- [Enable a vSAN File Service](#)
- [Create a vSAN File Service Domain](#)
- [Create a vSAN File Share](#)
- [Create an NFS 4.1 Datastore](#)
- [Add a VASA Provider and Create a Policy](#)
- [Invoke a Planned Failover on a Replication Group and Reverse the Replication](#)
- [Attach a Flat VDisk to a Virtual Machine](#)

Create a Tag-Based Storage Policy

You can create storage policies by using tags from vCenter Server.

Prerequisites

- Verify that you are connected to a vCenter Server system.
- Verify that you have Profile-driven storage update privileges.
- Verify that a tag named *Tag1* exists in the vCenter Server environment.

Procedure

- 1 Get the *Tag1* tag and store it in the *\$tag* variable.

```
$tag = Get-Tag -Name 'Tag1'
```

- 2 Create a rule with the *\$tag* tag and store the rule in the *\$rule* variable.

```
$rule = New-SpbmRule -AnyOfTags $tag
```

- 3 Create a rule set by using the *\$rule* rule and store the rule set in the *\$ruleset* variable.

```
$ruleset = New-SpbmRuleSet -AllOfRules $rule
```

- 4 Create a tag-based policy named *Tag-Based-Policy* by using the *\$ruleset* rule set and store the policy in the *\$policy* variable.

```
$policy = New-SpbmStoragePolicy -Name 'Tag-Based-Policy' -Description 'This policy is created by using a tag' -AnyOfRuleSets $ruleset
```

Create a Capability-Based Storage Policy

You can create storage policies by using vendor-exposed capabilities.

Prerequisites

- Verify that you are connected to a vCenter Server system.
- Verify that you have Profile-driven storage update privileges.
- Verify that a storage provider is registered with the vCenter Server system.

Procedure

- 1 Get the *VSAN.hostFailuresToTolerate* capability and store it in the *\$cap* variable.

```
$cap = Get-SpbmCapability -Name 'VSAN.hostFailuresToTolerate'
```

- 2 Create a rule with the *\$cap* capability and store the rule in the *\$rule* variable.

```
$rule = New-SpbmRule -Capability $cap -value 1
```

- 3 Create a rule set by using the *\$rule* rule and store the rule set in the *\$ruleset* variable.

```
$ruleset = New-SpbmRuleSet -AllOfRules $rule
```

- 4 Create a capability-based policy named *Capability-Based-Policy* by using the *\$ruleset* rule set and store the policy in the *\$policy* variable.

```
$policy = New-SpbmStoragePolicy -Name 'Capability-Based-Policy' -Description 'This policy is created by using capabilities' -AnyOfRuleSets $ruleset
```

Associate a Storage Policy with a Virtual Machine and Its Hard Disk

You can associate a storage policy with a virtual machine and its hard disk and check if they are compliant with the policy.

Prerequisites

- Verify that you are connected to a vCenter Server system.
- Verify that a storage policy named *Str-Policy* exists in the vCenter Server environment.
- Verify that a virtual machine named *Target-VM* exists in the vCenter Server environment.

Procedure

- 1 Get the *Str-Policy* storage policy and store it in the *\$policy* variable.

```
$policy = Get-SpbmStoragePolicy -Name 'Str-Policy'
```

- 2 Get the *Target-VM* virtual machine and store it in the *\$vm* variable.

```
$vm = Get-VM -Name 'Target-VM'
```

- 3 Get the hard disk associated with the *\$vm* virtual machine and store it in the *\$hd* variable.

```
$hd = Get-HardDisk -VM $vm
```

- 4 Assign the *\$policy* storage policy to the *\$vm* virtual machine and the *\$hd* hard disk.

```
Set-SpbmEntityConfiguration $vm, $hd -StoragePolicy $policy
```

- 5 View the *\$policy* storage policy's compliance with the *\$vm* virtual machine and the *\$hd* hard disk.

```
Get-SpbmEntityConfiguration $vm, $hd
```

Note The storage policy can be compliant only if the datastore on which the virtual machine and hard disk are created is compliant with the storage policy.

Disassociate a Storage Policy Associated with a Virtual Machine and Its Hard Disk

You can disassociate a storage policy that is associated with a virtual machine and its hard disk.

Prerequisites

- Verify that you are connected to a vCenter Server system.
- Verify that a virtual machine named *Target-VM* exists in the vCenter Server environment.
- Verify that a storage policy is associated with the *Target-VM* virtual machine.

Procedure

- 1 Get the *Target-VM* virtual machine and store it in the *\$vm* variable.

```
$vm = Get-VM -Name 'Target-VM'
```

- 2 Get the hard disk associated with the *\$vm* virtual machine and store it in the *\$hd* variable.

```
$hd = Get-HardDisk -VM $vm
```

- 3 Disassociate all storage policies that are associated with the *\$vm* virtual machine and the *\$hd* hard disk.

```
Set-SpbmEntityConfiguration $vm, $hd -StoragePolicy $null
```

Remove a Storage Policy

You can disassociate all entities associated with a storage policy and remove the policy completely.

Prerequisites

- Verify that you are connected to a vCenter Server system.
- Verify that you have Profile-driven storage update privileges.
- Verify that a storage policy named *pol-tag* exists in the vCenter Server environment.

Procedure

- 1 Get the *pol-tag* storage policy and store it in the *\$policy* variable.

```
$policy = Get-SpbmStoragePolicy -Name 'pol-tag'
```

- 2 Disassociate all entities associated with the *\$policy* storage policy.

```
Set-SpbmEntityConfiguration (Get-SpbmEntityConfiguration -StoragePolicy $policy)
-StoragePolicy $null
```

3 Remove the *\$policy* storage policy.

```
Remove-SpbmStoragePolicy -StoragePolicy $policy
```

Edit a Storage Policy

You can modify a storage policy to replace an existing rule set with a new rule set.

Prerequisites

- Verify that you are connected to a vCenter Server system.
- Verify that you have Profile-driven storage update privileges.
- Verify that a storage provider is registered with the vCenter Server system.
- Verify that a storage policy named *pol-tag* exists in the vCenter Server environment.

Procedure

- 1 Get the *pol-tag* storage policy and store it in the *\$policy* variable.

```
$policy = Get-SpbmStoragePolicy -Name 'pol-tag'
```

- 2 Create a new rule and store it in the *\$newRule* variable.

```
$newRule = New-SpbmRule -Capability (Get-SpbmCapability -Name  
'VSAN.hostFailuresToTolerate') -Value 1
```

- 3 Create a new rule set by using the *\$newRule* rule and store it in the *\$newRuleset* variable.

```
$newRuleset = New-SpbmRuleSet -AllOfRules $newRule
```

- 4 Modify the *\$policy* storage policy by replacing the existing rule set with the newly created *\$newRuleset* rule set.

```
$modPolicy = Set-SpbmStoragePolicy -StoragePolicy $policy -AnyOfRuleSets $newRuleset
```

Export and Import a Storage Policy

You can back up a storage policy by exporting it as a file. You can later import the same storage policy.

Prerequisites

- Verify that you are connected to a vCenter Server system.
- Verify that you have Profile-driven storage update privileges.
- Verify that you have read-write permissions for the directory in which the storage policy is saved.

- Verify that a storage policy named *pol-tag* exists in the vCenter Server environment.

Procedure

- 1 Export the *pol-tag* storage policy.

```
Export-SpbmStoragePolicy -StoragePolicy 'pol-tag' -FilePath 'C:\Policy\pol-tag.xml'
```

- 2 Import the *pol-tag* storage policy and name it *Imported-Policy*.

```
Import-SpbmStoragePolicy -FilePath 'C:\Policy\pol-tag.xml' -Name 'Imported-Policy'
-Description 'Imported policy description'
```

Create a Virtual Machine in a Datastore Compatible with Storage Policy

You can retrieve a datastore compatible with storage policy and create a virtual machine in the datastore.

Prerequisites

- Verify that you are connected to a vCenter Server system.
- Verify that a tag-based storage policy named *Tag-Policy* exists in the vCenter Server environment.
- Verify that the tag of the *Tag-Policy* storage policy is associated with one of the available datastores in the vCenter Server environment.

Procedure

- 1 Get the tag-based *Tag-Policy* storage policy and store it in the *\$policy* variable.

```
$policy = Get-SpbmStoragePolicy -Name 'Tag-Policy'
```

- 2 Get the tag used in the *Tag-Policy* storage policy and store it in the *\$tag* variable.

```
$tag = (($($policy.AnyOfRulesets).AllOfRules).AnyOfTags)[0]
```

- 3 Get a datastore compatible with the *\$policy* storage policy and store it in the *\$ds* variable.

```
$ds = Get-SpbmCompatibleStorage -StoragePolicy $policy
```

- 4 Get the virtual machine host that contains the *\$ds* datastore and store it in the *\$vmhost* variable.

```
$vmHost = Get-VMHost -Datastore $ds
```

- 5 Create a virtual machine named *VM-Tag* in the *\$ds* datastore and store the virtual machine object in the *\$vm* variable.

```
$vm = New-VM -Name 'VM-Tag' -ResourcePool $vmHost -Datastore $ds -NumCPU 2 -MemoryGB 4
-DiskMB 1
```

- 6 Associate the *\$policy* storage policy with the *\$vm* virtual machine.

```
Set-SpbmEntityConfiguration $vm -StoragePolicy $policy
```

- 7 Verify that the *\$vm* virtual machine is compliant with the *\$policy* storage policy.

```
Get-SpbmEntityConfiguration $vm
```

The status should be `Compliant`.

- 8 Get the `Tag-Assignment` object for the *\$ds* datastore and store it in the *\$tagAs* variable.

```
$tagAs = Get-TagAssignment -Entity $ds
```

- 9 Remove the *\$tag* tag association from the *\$ds* datastore.

```
Remove-TagAssignment -TagAssignment $tagAs
```

- 10 Check the compliance of the *\$vm* virtual machine with the *\$policy* storage policy.

```
Get-SpbmEntityConfiguration $vm
```

The status should be `NonCompliant`.

Create a vSAN Datastore

You can create vSAN disk groups on standalone hosts and add the hosts to a vSAN-enabled cluster to form a vSAN datastore. You can then create a virtual machine on the vSAN datastore and assign a storage policy to the virtual machine and its hard disk.

Prerequisites

- Verify that you are connected to a vCenter Server system.
- Verify that you have access to at least three virtual machine hosts.
- Verify that each of the virtual machine hosts has at least one SSD and one HDD.
- Verify that the virtual machine hosts are in maintenance mode.

Procedure

- 1 Create a vSAN enabled cluster with manual disk claim mode.

```
$vsanCluster = New-Cluster -Name 'VsanCluster' -Location (Get-Datacenter) -VsanEnabled
-VsanDiskClaimMode 'Manual'
```

2 Configure a vSAN VMkernel port on each of the three hosts.

```
New-VMHostNetworkAdapter -VMHost 'Host-A' -PortGroup 'VMkernel' -VirtualSwitch 'vSwitch0'
-VsanTrafficEnabled $true
New-VMHostNetworkAdapter -VMHost 'Host-B' -PortGroup 'VMkernel' -VirtualSwitch 'vSwitch0'
-VsanTrafficEnabled $true
New-VMHostNetworkAdapter -VMHost 'Host-C' -PortGroup 'VMkernel' -VirtualSwitch 'vSwitch0'
-VsanTrafficEnabled $true
```

3 Create a vSAN disk group on each of the three hosts.

```
New-VsanDiskGroup -DataDiskCanonicalName 'HDD1-CanonicalName' -SsdCanonicalName 'SSD1-
CanonicalName' -VMHost 'Host-A'
New-VsanDiskGroup -DataDiskCanonicalName 'HDD1-CanonicalName' -SsdCanonicalName 'SSD1-
CanonicalName' -VMHost 'Host-B'
New-VsanDiskGroup -DataDiskCanonicalName 'HDD1-CanonicalName' -SsdCanonicalName 'SSD1-
CanonicalName' -VMHost 'Host-C'
```

4 Add each of the three hosts to the vSAN cluster to create a vSAN datastore.

```
Move-VMHost -VMHost 'Host-A' -Destination $vsanCluster
Move-VMHost -VMHost 'Host-B' -Destination $vsanCluster
Move-VMHost -VMHost 'Host-C' -Destination $vsanCluster
```

5 Revert the virtual machine hosts to the Connected state.

```
Set-VMHost -VMHost 'Host-A','Host-B','Host-C' -State 'Connected'
```

6 Create a virtual machine on the vSAN datastore.

```
$vsanDS = Get-Datastore -Name 'vsanDatastore'
$vm = New-VM -Name 'newVM' -DiskMB 1024 -Datastore $vsanDS -VMHost 'Host-A'
```

7 Create a storage policy by using any of the vSAN capabilities.

```
$scap = Get-SpbmCapability -Name vSAN*
$rule = New-SpbmRule $scap[1] $true
$ruleset = New-SpbmRuleSet $rule
$policy = New-SpbmStoragePolicy -Name 'vsan policy' -RuleSet $ruleset -Description 'vSAN-
based storage policy'
```

8 Assign the storage policy to the virtual machine and its hard disk.

```
$vmHdd = Get-HardDisk -VM $vm
Set-SpbmEntityConfiguration $vm, $vmHdd -StoragePolicy $policy
```

9 Check the compliance of the virtual machine and its hard disk with the storage policy.

```
Get-SpbmEntityConfiguration $vm, $vmHdd
```

The status should be Compliant.

Modify a vSAN Datastore

You can add or remove local disks from existing vSAN disk groups or remove entire vSAN disk groups.

Prerequisites

- Verify that you are connected to a vCenter Server system.
- Verify that at least one vSAN disk group exists in the cluster.

Procedure

- 1 Get the vSAN disk group from a cluster.

```
$dgs = Get-VsanDiskGroup -Cluster 'VsanCluster'
```

- 2 Get all vSAN disks from the vSAN disk group.

```
$dg = $dgs[0]
Get-VsanDisk -VsanDiskGroup $dg
```

- 3 Add a hard disk to the vSAN disk group.

```
$disk = New-VsanDisk -CanonicalName 'HDD-CanonicalName' -VsanDiskGroup $dg
```

- 4 Remove a hard disk from the vSAN disk group.

```
Remove-VsanDisk -VsanDisk $disk
```

- 5 Remove the entire vSAN disk group.

```
Remove-VsanDiskGroup -VsanDiskGroup $dg
```

Create a vSAN Stretched Cluster

You can create a vSAN stretched cluster with a witness node. You can then create a vSAN storage policy and enable performance service on the vSAN cluster.

Prerequisites

- Verify that you are connected to a vCenter Server system.
- Verify that you have access to at least two virtual machine hosts.
- Verify that each of the virtual machine hosts has at least one SSD and one HDD.
- Verify that the virtual machine hosts are in maintenance mode.
- Verify that you have access to an ESXi host that can be used as a witness host or deploy a witness appliance on any node. Ensure that the witness host or appliance is outside the vSAN cluster.

Procedure

- 1 Configure a vSAN VMkernel port on each of the two hosts.

```
New-VMHostNetworkAdapter -VMHost 'Host-A' -PortGroup 'VMkernel' -VirtualSwitch 'vSwitch0'
-VsanTrafficEnabled $true
New-VMHostNetworkAdapter -VMHost 'Host-B' -PortGroup 'VMkernel' -VirtualSwitch 'vSwitch0'
-VsanTrafficEnabled $true
```

- 2 Create a vSAN enabled cluster with automatic disk claim mode.

```
$vsanCluster = New-Cluster -Name 'VsanCluster' -Location (Get-Datacenter) -VsanEnabled
-VsanDiskClaimMode 'Automatic'
```

- 3 Add the two hosts to the vSAN cluster to create a vSAN datastore.

```
Move-VMHost -VMHost 'Host-A' -Destination $vsanCluster
Move-VMHost -VMHost 'Host-B' -Destination $vsanCluster
```

- 4 Revert the virtual machine hosts to the Connected state.

```
Set-VMHost -VMHost 'Host-A','Host-B' -State 'Connected'
```

- 5 Create two fault domains in the vSAN cluster.

```
$primaryFd = New-VsanFaultDomain -Name 'Primary' -VMHost 'Host-A'
$secondaryFd = New-VsanFaultDomain -Name 'Secondary' -VMHost 'Host-B'
```

- 6 Enable stretched cluster.

```
Set-VsanClusterConfiguration -Configuration $vsanCluster -StretchedClusterEnabled $true
-PreferredFaultDomain $primaryFd -WitnessHost 'Witness-Virtual-Appliance-IP'
```

- 7 Create a storage policy by using any vSAN capability.

```
$scap = Get-SpbmCapability -Name vSAN*
$rule = New-SpbmRule $scap[1] $true
$ruleset = New-SpbmRuleSet $rule
$policy = New-SpbmStoragePolicy -Name 'vsan policy' -RuleSet $ruleset -Description 'vSAN-
based storage policy'
```

- 8 Enable performance service on the vSAN cluster.

```
Set-VsanClusterConfiguration -Configuration $vsanCluster -PerformanceServiceEnabled $true
-StoragePolicy $policy
```

Enable a vSAN File Service

With PowerCLI, you can enable a vSAN file service.

Prerequisites

Verify that you are connected to a vCenter Server system.

Procedure

- 1 Add a vSAN file service OVF to the OVF repository of the vCenter Server system.

```
Add-VsanFileServiceOvf
```

- 2 Verify that the vSAN file service OVF is successfully added.

```
Get-VsanFileServiceOvfInfo
```

- 3 Increase the PowerCLI timeout for the current session, so that the operation does not stop.

```
Set-PowerCLIConfiguration -WebOperationTimeoutSeconds 3600 -Scope Session
```

- 4 Retrieve the network that you want to use for the vSAN file service.

```
$net = Get-VirtualNetwork -Name 'MyNetwork'
```

- 5 Modify the vSAN cluster configuration that you want to use to enable the vSAN file service.

```
$config = Get-VsanClusterConfiguration
Set-VsanClusterConfiguration $config -FileServiceEnabled $true -FileServiceNetwork $net
```

Create a vSAN File Service Domain

With PowerCLI, you can create a vSAN file service domain.

Prerequisites

- Verify that you are connected to a vCenter Server system.
- Verify that the vSAN file service is enabled.

Procedure

- 1 Configure a vSAN file server IP configuration. You can use the IPs as an IP pool for the file service domain.

```
$ipConfig = New-VsanFileServerIpConfig -Fqdn "h101.vmware.com" -Gateway "192.2.8.3"
-IPAddress "192.2.8.4" -SubnetMask "255.255.254.0" -IsPrimary
```

- 2 Create a vSAN file service domain.

```
New-VsanFileServiceDomain -DnsServerAddress "10.172.199.241" -VsanFileServerIpConfig
$ipConfig -Name "MyFileServiceDomain" -DnsSuffix "vmware.com" -Cluster $config.Cluster
```


Create a vSAN File Share

With PowerCLI, you can create a vSAN file share.

Prerequisites

- Verify that you are connected to a vCenter Server system.
- Verify that the vSAN file service is enabled.
- Verify that the vSAN file service domain is created.

Procedure

- 1 Get a vSAN file service domain.

```
$domain = Get-VsanFileServiceDomain -Name "MyFileServiceDomain"
```

- 2 Create a vSAN file share.

```
New-vsanfileshare -FileServiceDomain $domain -Name "MyVsanFileShare"
```

Create an NFS 4.1 Datastore

You can create an NFS 4.1 datastore with Kerberos authentication and multipathing.

Prerequisites

- Verify that you are connected to a vCenter Server system.
- Verify that the remote NFS share supports multipathing and Kerberos authentication.

Procedure

- 1 Get the virtual machine host where you want to create the NFS 4.1 datastore.

```
$vmhost = Get-VMHost 'hostname'
```

- 2 Set NTP servers for the virtual machine host.

```
Add-VMHostNtpServer -VMHost $vmhost -NtpServer 'ntp_server_ip'
```

- 3 Set a DNS server and search the domain for the virtual machine host.

```
$vmhostnetwork = Get-VMHostNetwork -VMHost $vmhost
Set-VMHostNetwork -Network $vmhostnetwork -DnsFromDhcp $false -DnsAddress 'dns_server_ip'
-DomainName 'domain_name' -SearchDomain 'search_domain'
```

- 4 Add the virtual machine to the Active Directory domain.

```
$vmhost | Get-VMHostAuthentication | Set-VMHostAuthentication -JoinDomain -Domain
'AD_domain_name' -Username 'AD_user_name' -Password 'AD_password'
```

- 5 Create an NFS user on the virtual machine host for Kerberos-based authentication for the NFS 4.1 datastore.

```
New-NfsUser -VMHost $vmhost -Username 'NFS_user_name' -Password 'password'
```

- 6 Create an NFS 4.1 datastore with Kerberos authentication and multipathing.

```
New-Datastore -Name 'NFS_datastore_name' -Nfs -FileSystemVersion '4.1' -VMHost $vmhost  
-NfsHost @('remote_host_1_ip', 'remote_host_2_ip') -Path 'NFS_datastore_remote_path' -  
Kerberos
```

- 7 (Optional) Retrieve the datastore.

```
$ds = Get-Datastore 'NFS_datastore_name'
```

- 8 (Optional) Remove the datastore.

```
Remove-Datastore $ds -VMHost $vmhost
```

- 9 (Optional) Get the NFS user from the virtual machine host.

```
$user = Get-NfsUser -VMHost $vmhost
```

- 10 (Optional) Update the password of the NFS user.

```
$user = Set-NfsUser -NfsUser $user -Password 'new_password'
```

- 11 (Optional) Remove the NFS user.

```
Remove-NfsUser -NfsUser $user
```

Add a VASA Provider and Create a Policy

You can add a VASA provider to a vCenter Server system and create a storage policy.

Prerequisites

- Verify that you are connected to a vCenter Server system.
- Verify that the datastore is mounted to the ESXi host.

Procedure

- 1 Add a VASA provider to the vCenter Server system.

```
$script:vasProvider = New-VasaProvider -Name 'name' -Url 'URL' -Username 'user_name'  
-Password 'password' -Description 'description' -Force
```

- 2 Get all SPBM capabilities exposed by the registered VASA provider.

```
Get-SpbmCapability
```

- 3 Create a new SPBM rule with the exposed capabilities of the registered VASA provider.

```
$rule = New-SpbmRule -Capability $capability -Value $value
```

- 4 Create a new SPBM rule set.

```
New-SpbmRuleSet -Name $ruleset -AllofRules @(($rule))
```

- 5 Create a storage policy.

```
New-SpbmStoragePolicy -Name $storagepolicy -RuleSet $ruleset
```

- 6 Refresh the VASA provider registered with the vCenter Server system.

```
$provider = Get-VasaProvider -Name $providername -Refresh
```

- 7 Verify the VASA storage array.

```
$vasaStorageArray = Get-VasaStorageArray -Provider $vasaProvider -Server $script:vcsvr
```

- 8 Refresh the VASA provider registered with the vCenter Server system.

```
$provider = Get-VasaProvider -Name $providername -Refresh
```

- 9 Get the VASA provider registered with the vCenter Server system.

```
$vasaProvider = Get-VasaProvider -Name $providername
```

- 10 (Optional) Remove the VASA provider.

```
Remove-VasaProvider -Provider $provider -Confirm:$false
```

- 11 (Optional) Verify that the VASA provider is removed.

```
$provider = Get-VasaProvider -Name $providername
```

Invoke a Planned Failover on a Replication Group and Reverse the Replication

You can invoke a planned failover on a target replication group and recover the devices on the target site. After that, you can reverse the direction of the replication and make the target site the source site.

Prerequisites

- Verify that you are connected to the vCenter Server systems of the source and target sites.
- Verify that you have access to at least one virtual machine host on each site.
- Verify that you have registered a Virtual Volume VASA provider and have access to a Virtual Volume datastore on each site.

Procedure

- 1 Create a storage policy with replication capability on the source site.

```
$replicationCapability = Get-SpbmCapability -Name *replication.RPO -Server $srcServer
$persistenceCapability = Get-SpbmCapability -Name *persistence-readLatency -Server
$srcServer
$replicationRule = New-SpbmRule -Capability $ replicationCapability -Value (New-TimeSpan
-Hours 4)
$persistenceRule = New-SpbmRule $persistenceCapability -Value 25
$ruleSet = New-SpbmRuleSet -AllOfRules $replicationRule, $persistenceRule
$replicationPolicy = New-SpbmStoragePolicy -Name cokeRep -AnyOfRuleSets $ruleSet -Server
$srcServer
```

- 2 Get a datastore compatible with the created replication storage policy and store it in the *\$ds* variable.

```
$ds = Get-SpbmCompatibleStorage -StoragePolicy $replicationPolicy
```

- 3 Create a virtual machine named *MyVM* with a hard disk in the *\$ds* datastore.

```
$vm = New-VM -Name 'MyVM' -VMHost 'Host-Source' -DiskMB 512 -Datastore $ds
$hd = Get-HardDisk -VM $vm
```

- 4 Get a replication group for the *\$ds* datastore and the *\$replicationPolicy* storage policy, and store the replication group in the *\$rg* variable.

```
$rg = Get-SpbmReplicationGroup -Datastore $ds -StoragePolicy $replicationPolicy
```

- 5 Associate the *\$vm* virtual machine and its hard disk with the *\$replicationPolicy* storage policy, and put them in the *\$rg* replication group.

```
Set-SpbmEntityConfiguration -Configuration $vm, $hd -StoragePolicy $replicationPolicy
-ReplicationGroup $rg
```

- 6 Check the compliance of the *\$vm* virtual machine and *\$hd* hard disk with the *\$replicationPolicy* storage policy.

```
Get-SpbmEntityConfiguration $vm, $hd
```

- 7 Get the replication pair corresponding to the *\$rg* source replication group, and store that pair in the *\$rgPair* variable.

```
$rgPair = Get-SpbmReplicationPair -Source $rg
```

- 8 Synchronize the target replication group.

```
Sync-SpbmReplicationGroup $rgPair.Target
```

- 9 Power off the *\$vm* virtual machine and unregister it.

```
Stop-VM $vm
Remove-VM $vm
```

- 10 Prepare the failover on the source replication group.

```
Start-SpbmReplicationPrepareFailover $rgPair.Source
```

- 11 Synchronize the target replication group again, to get the latest state of the source devices.

```
Sync-SpbmReplicationGroup $rgPair.Target
```

- 12 Invoke the planned failover on the source replication group and store the virtual machine file path on the target site in the *\$vmFilePath* variable.

```
$vmFilePath = Start-SpbmReplicationFailover $rgPair.Target
```

- 13 Register the virtual machine on the *Host-Target* host and power on the virtual machine.

```
$vm = New-VM -VMFilePath $vmFilePath -VMHost 'Host-Target'
Start-VM $vm
```

- 14 Reverse the direction of the replication.

```
Start-SpbmReplicationReverse $rgPair.Target
```

Attach a Flat VDisk to a Virtual Machine

You can create a flat `VDisk` object and attach it to a virtual machine as a hard disk. After that, you can verify whether the operation was completed successfully.

Prerequisites

- Verify that you are connected to a vCenter Server system.
- Verify that you have access to at least one virtual machine host.
- Verify that there is at least one datastore mounted on the virtual machine host.

Procedure

- 1 Mount the datastore on the virtual machine host and store the datastore in the *\$ds* variable.

```
$vmHost = Get-VMHost 'Host-A'
$ds = Get-Datastore -RelatedObject $vmHost
```

- 2 Create a flat, thin-provisioned virtual disk with 2-GB capacity on the *\$ds* datastore.

```
$vDisk = New-VDisk -Name 'VirtualDisk' -DiskType Flat -StorageFormat Thin -CapacityGB 2
-Datastore $ds
```

- 3 Create a virtual machine named *VirtualMachine* with one hard disk and store this virtual machine in the *\$vm* variable.

```
$vm = New-VM -Name 'VirtualMachine' -VMHost 'Host-A' -Datastore $ds -DiskMB 512
```

- 4 Power on the *\$vm* virtual machine.

```
Start-VM -VM $vm
```

- 5 Attach the *vDisk* object to the *\$vm* virtual machine.

```
New-HardDisk -VM $vm -VDisk $vDisk
```

- 6 Verify that the *vDisk* object has been attached and the virtual machine now has two hard disks.

```
Get-HardDisk -VM $vm
```

Managing the vSphere Replication API with VMware PowerCLI



Use the vSphere Replication SDK for PowerShell to access the vSphere Replication API with PowerCLI. Starting from VMware PowerCLI version 13.1.0, the vSphere Replication SDK provides low-level cmdlets for all available vSphere Replication API operations.

Read the following topics next:

- [Understanding the vSphere Replication SDK for PowerShell](#)
- [Navigating the vSphere Replication SDK for PowerShell](#)
- [Connect to a Local vSphere Replication Server](#)
- [Connect to a Local and to a Remote vSphere Replication Server](#)
- [Use the ConnectedPairings Property](#)
- [Replicate a VM to a Datastore on the Local vCenter Server System](#)
- [Replicate a VM to a Datastore on a Remote vCenter Server System](#)

Understanding the vSphere Replication SDK for PowerShell

You can access the vSphere Replication API with VMware PowerCLI by using the vSphere Replication SDK for PowerShell.

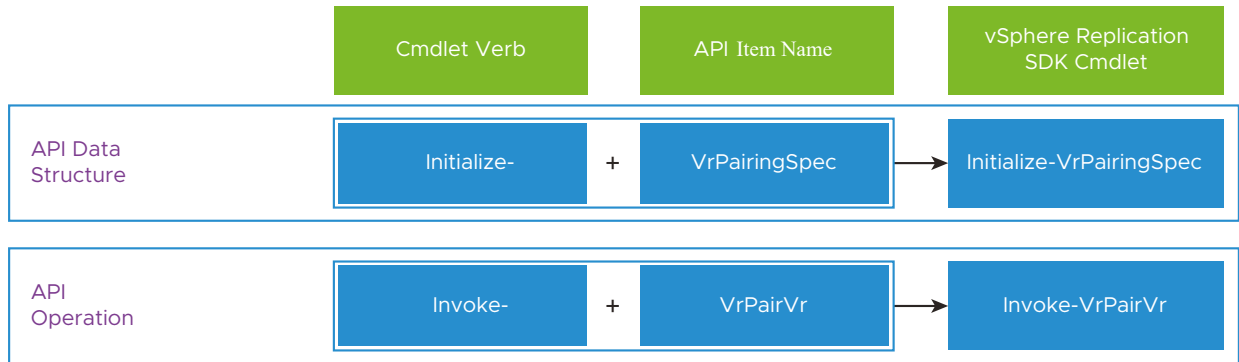
How Does the vSphere Replication SDK for PowerShell Work

Starting from VMware PowerCLI 13.1.0, you can use the vSphere Replication SDK for PowerShell to communicate with all available vSphere Replication REST APIs.

The `VMware.Sdk.Vr` module contains automatically-generated SDK cmdlets that bind directly to the vSphere Replication REST APIs. The module functions on a low level and communicates directly with the vSphere Replication API server.

You can construct vSphere Replication SDK cmdlets by using the `Initialize-` and `Invoke-` cmdlet verbs in combination with the names of the vSphere Replication API data structures or operations.

Figure 8-1. Constructing vSphere Replication SDK Cmdlets



- Use `Initialize-` cmdlets to prepare the data structures, if any, for your API calls. These cmdlets function on the client side and do not communicate with the vSphere Replication API server.
- Use `Invoke-` cmdlets to call the vSphere Replication API server and invoke the operations.

Navigating the vSphere Replication SDK for PowerShell

You can navigate the vSphere Replication SDK for PowerShell by using the VMware Developer website, and the `Get-Help`, `Get-Command`, and `Get-VrOperation` cmdlets.

To use the vSphere Replication SDK for PowerShell effectively, you must be familiar with the [vSphere Replication API reference documentation](#).

How to Find a vSphere Replication Cmdlet

There are several ways to find the vSphere Replication cmdlets you need.

- In your browser, navigate the vSphere Replication API reference documentation.

The vSphere Replication API Reference on VMware Developer features ready-to-use vSphere Replication SDK code samples that you can paste directly into PowerShell. You can find a PowerCLI code snippet for each operation under *PowerCLI Client SDK Example* in the *Code Samples* section.
- In PowerShell, run `Get-Command -Module VMware.Sdk.Vr` to list all available vSphere Replication cmdlets.
- If you want to use information from the REST API specification to find a cmdlet, you can run `Get-VrOperation` with the `Method` and `Path` parameters.

For example:

```
Get-VrOperation -Method GET -Path '/replication-servers'
```


The command returns an object with the operation details.

```
Name           : GetAllVrServers
CommandInfo    : Invoke-VrGetAllVrServers
ApiName        : ServerApi
Path           : /replication-servers
Tags           : {server}
RelatedCommandInfos : {}
Method         : GET
Parameters     : {filterProperty, filter, sortBy, orderBy...}
```

The cmdlet name is listed in the `CommandInfo` property.

How to Get Information About a Cmdlet

To get information about a vSphere Replication cmdlet, run `Get-Help` with the cmdlet name. For example:

```
Get-Help Invoke-VrConfigureReplication -full
```

Connect to a Local vSphere Replication Server

For replications within the same vCenter Server system, you must first connect to a local vSphere Replication server. You can do this by using the `Connect-VrServer` cmdlet.

Prerequisites

- Verify that a vSphere Replication appliance is deployed at the local vCenter Server system.
- If the certificate of the server you want to connect to is not trusted, verify that your PowerCLI invalid server certificate action settings are configured properly. For more information, see [Configuring PowerCLI Response to Untrusted Certificates](#).
- If you use a proxy server for the connection, verify that it is configured properly, so that the connection is kept alive long enough for tasks to finish.

Note If you do not want to use a proxy server for the connection, run `Set-PowerCLIConfiguration -ProxyPolicy NoProxy`.

Procedure

- 1 Run `Connect-VrServer` and create a variable for the connection. Use the local vSphere Replication server address with the local vCenter Server user credentials.

```
$vr1 = Connect-VrServer -Server <local_vr_address_or_fqdn> -User 'MyVCUser' -Password 'MyVCPassWord'
```

- (Optional) Print the connection variable to see the `VrServerConnection` object you created.

```
$vr1
```

The `VrServerConnection` object contains the `ConnectedPairings` property with the pairing you created. You might need this property to extract information required by some vSphere Replication operations.

Connect to a Local and to a Remote vSphere Replication Server

For replications from a source to a target vCenter Server system, you use the `Connect-VrServer` cmdlet to connect to the local and to the remote vSphere Replication server.

Prerequisites

- Verify that vSphere Replication appliances are deployed at the local and at the remote vCenter Server systems.
- If the certificate of the server you want to connect to is not trusted, verify that your PowerCLI invalid server certificate action settings are configured properly. For more information, see [Configuring PowerCLI Response to Untrusted Certificates](#).
- If you use a proxy server for the connection, verify that it is configured properly, so that the connection is kept alive long enough for tasks to finish.

Note If you do not want to use a proxy server for the connection, run `Set-PowerCLIConfiguration -ProxyPolicy NoProxy`.

Procedure

- Run `Connect-VrServer` and create a variable for the connection. Use the local vSphere Replication server address with the local vCenter Server user credentials. For the remote server connection, use the remote vCenter Server address with the remote vCenter Server user credentials.

```
$vr2 = Connect-VrServer -Server <local_vr_address_or_fqdn> -User 'MyVCUser' -Password
'MyVCPassWord' -RemoteServer <remote_vc_server_name> -RemoteUser 'MyRemoteVCUser'
-RemotePassword 'MyRemoteVCPassWord'
```

- (Optional) Print the connection variable to see the `VrServerConnection` object you created.

```
$vr2
```

The `VrServerConnection` object contains the `ConnectedPairings` property with the pairing you created. You might need this property to extract information required by some vSphere Replication operations.

Use the ConnectedPairings Property

Use the `ConnectedPairings` property to extract connection information that you might need for vSphere Replication SDK operations.

The `ConnectedPairings` property is a key-value dictionary that contains information about all available vSphere Replication pairings. Use `ConnectedPairings` to extract information such as `PairingID` and `VcServerID`, which is required for some vSphere Replication SDK operations.

Prerequisites

Verify that you have a valid `VrServerConnection`. This example uses the `$vr2` connection variable from [Connect to a Local and to a Remote vSphere Replication Server](#).

Procedure

- 1 To list the connected pairings for a vSphere Replication connection, run:

```
$vr2.ConnectedPairings
```

The system lists a key-value map with your connected pairings.

- 2 To list a pairing, run:

```
$vr2.ConnectedPairings[<vc_server_name>].Pairing
```

The system returns the `Pairing` object with all its properties. You can copy the `PairingID` property as it is required by some vSphere Replication API operations.

- 3 To list the `LocalVcServer` object, run:

```
$vr2.ConnectedPairings[<vc_server_name>].Pairing.LocalVcServer
```

The system returns a `ServerInfo` object that contains a server ID that is required by some vSphere Replication API operations.

- 4 To list the `RemoteVcServer` object, run:

```
$vr2.ConnectedPairings[<vc_server_name>].Pairing.RemoteVcServer
```

The system returns a `ServerInfo` object that contains a server ID that is required by some vSphere Replication API operations.

Replicate a VM to a Datastore on the Local vCenter Server System

You can use the vSphere Replication SDK for PowerShell to replicate a VM from a source to a target datastore on the local vCenter Server system.

This example shows how to replicate a single VM from a source to a target datastore on the same vCenter Server system.

Prerequisites

- Verify that the vSphere Replication appliance is deployed on the vCenter Server system.
- Verify that you are connected to the vCenter Server system. This example uses the `$vr1` connection variable from [Connect to a Local vSphere Replication Server](#).
- You must have the required privileges to configure replications and manage datastores on the vCenter Server system.

Procedure

- 1 Get the local VM that you want to replicate.

```
$pairing = $vr1.ConnectedPairings["<local_vc_address_or_fqdn>"].Pairing
$replicatedVm = Invoke-VrGetLocalVms -PairingId $pairing.PairingId `
    -VcenterId $pairing.LocalVcServer.Id.Guid `
    -FilterProperty "Name" `
    -Filter "<my_vm_name>" `
    -Server $vr1
```

- 2 Get the VM hard disks that you want to replicate.

```
$replicatedHdds = Invoke-VrGetLocalVmDisks -PairingId $pairing.PairingId `
    -VcenterId $pairing.LocalVcServer.Id.Guid `
    -VmId $replicatedVm.List[0].Id `
    -Server $vr1
```

- 3 Get the target datastore for the replication.

```
$targetDatastore = Invoke-VrGetVrCapableTargetDatastores -PairingId $pairing.PairingId `
    -VcenterId $pairing.PairingId.RemoteVcServer.Id.Guid `
    -FilterProperty "Name" `
    -Filter "<target_datastore_name>" `
    -Server $vr1
```

- 4 Create the configuration for disks replication.

```
$replicationVmDisks = @()
$replicatedHdds.List | ForEach-Object {
    $replicationVmDisks += Initialize-VrConfigureReplicationVmDisk -VmDisk $_ `
        -EnabledForReplication:$true `
        -DestinationDatastoreId $targetDatastore.List[0].Id `
        -DestinationDiskFormat 'SAMEASSOURCE'
```

- 5 Create the replication spec.

```
$replicationSpec = Initialize-VrConfigureReplicationSpec `
    -Rpo 60 -NetworkCompressionEnabled:$true `
    -MpitrEnabled:$true -AutoReplicateNewDisks:$true `
    -LwdEncryptionEnabled:$false `
```

```
-Disks $replicationVmDisks `
-MpitInstances 1 `
-MpitDays 1 `
-TargetVcId $pairing.RemoteVcServer.Id.Guid `
-VmId $replicatedVm.List[0].Id
```

6 Invoke the ConfigureReplication operation.

```
$task = Invoke-VrConfigureReplication -PairingId $pairing.PairingId `
-ConfigureReplicationSpec $replicationSpec -Server $vrl
```

vSphere Replication starts replicating the virtual machine files to the target datastore.

7 Monitor the task progress.

```
$t = Invoke-VrGetTaskInfo -TaskId $task.List[0].Id -Server $vrl
$t
```

Eventually, the task status changes to `SUCCESS`.

8 (Optional) Get the replication on the target datastore.

```
$repl = Invoke-VrGetAllReplications -PairingId $pairing.PairingId `
-FilterProperty "Name" -Filter <my_vm_name> -Server $vrl
```

9 Sync your virtual machine to the configured replication on the target datastore.

```
Invoke-VrSyncReplication -PairingId $pairing.PairingId `
-ReplicationId $repl.List[0].Id -Server $vrl
```

Replicate a VM to a Datastore on a Remote vCenter Server System

You can use the vSphere Replication SDK for PowerShell to replicate a VM from a source to a target vCenter Server system.

This example shows how to replicate a single VM from a datastore on the local vCenter Server system to a datastore on a remote vCenter Server system.

Prerequisites

- Verify that the vSphere Replication appliance is deployed on the source and target sites.
- Verify that your `VrServerConnection` is authenticated against the source and target vCenter Server systems. This example uses the `$vr2` connection variable from [Connect to a Local and to a Remote vSphere Replication Server](#).
- You must have the required privileges to configure replications and manage datastores on the source and target sites.

Procedure

- 1 Get the local VM that you want to replicate.

```
$pairing = $vr2.ConnectedPairings["<remote_vc_server_name>"].Pairing
$replicatedVm = Invoke-VrGetLocalVms -PairingId $pairing.PairingId `
    -VcenterId $pairing1.LocalVcServer.Id.Guid `
    -FilterProperty "Name" `
    -Filter "<my_vm_name>" `
    -Server $vr2
```

- 2 Get the VM hard disks that you want to replicate.

```
$replicatedHdds = Invoke-VrGetLocalVmDisks -PairingId $pairing.PairingId `
    -VmId $replicatedVm.List[0].Id `
    -VcenterId $pairing.LocalVcServer.Id.Guid `
    -Server $vr2
```

- 3 Get the target datastore on the remote site.

```
$targetDatastore = Invoke-VrGetVrCapableTargetDatastores -PairingId $pairing.PairingId `
    -VcenterId $pairing.RemoteVcServer.Id.Guid `
    -FilterProperty "Name" -Filter "<remote_datastore_name>" -Server $vr2
```

- 4 Create the configuration for disks replication.

```
$replicatedHdds.List | ForEach-Object {
    $replicationVmDisks += Initialize-VrConfigureReplicationVmDisk -VmDisk $_ `
        -EnabledForReplication:$true `
        -DestinationDatastoreId $targetDatastore.List[0].Id `
        -DestinationDiskFormat 'SAMEASSOURCE'
}
```

- 5 Create the replication spec.

```
$replicationSpec = Initialize-VrConfigureReplicationSpec `
    -Rpo 60 -NetworkCompressionEnabled:$true `
    -MpitEnabled:$true -AutoReplicateNewDisks:$true `
    -LwdEncryptionEnabled:$false `
    -Disks $replicationVmDisks `
    -MpitInstances 1 `
    -MpitDays 1 `
    -TargetVcId $pairing.RemoteVcServer.Id.Guid `
    -VmId $replicatedVm.List[0].Id
```

- 6 Invoke the ConfigureReplication operation.

```
$task = Invoke-VrConfigureReplication -PairingId $pairing.PairingId `
    -ConfigureReplicationSpec $replicationSpec -Server $vr2
```

vSphere Replication starts replicating the virtual machine files to the target site datastore.

7 Monitor the task progress.

```
$t = Invoke-VrGetTaskInfo -TaskId $task.List[0].Id -Server $vr2  
$t
```

Eventually, the task status must change to `SUCCESS`.

8 (Optional) Get the replication on the remote site.

```
$rep2 = Invoke-VrGetAllReplications -PairingId $pairing.PairingId `  
-FilterProperty "Name" -Filter <my_vm_name> -Server $vr2
```

9 (Optional) Sync your virtual machine to the configured replication on the target site.

```
Invoke-VrSyncReplication -PairingId $pairing.PairingId `  
-ReplicationId $rep2.List[0].Id -Server $vr2
```

Managing VMware Site Recovery Manager (SRM) with VMware PowerCLI

9

To help you get started with VMware PowerCLI, this documentation provides a set of sample scripts that illustrate basic and advanced tasks in VMware Site Recovery Manager (SRM) (SRM) administration.

Read the following topics next:

- [Connect to an SRM Server](#)
- [Protect a Virtual Machine](#)
- [Create a Report of the Protected Virtual Machines](#)
- [Create a Report of the Virtual Machines Associated with All Protection Groups](#)

Connect to an SRM Server

To use the SRM API, you must establish a connection to an SRM server.

Some of the objects returned by the SRM API are objects from the vSphere API. To use those objects in integration with the vSphere API through PowerCLI, you can connect to the vCenter Server system that the SRM server is registered with.

Prerequisites

- If the certificate of the server you want to connect to is not trusted, verify that your PowerCLI invalid server certificate action settings are configured properly. For more information, see [Configuring PowerCLI Response to Untrusted Certificates](#).
- If you use a proxy server for the connection, verify that it is configured properly, so that the connection is kept alive long enough for tasks to finish.

Note If you do not want to use a proxy server for the connection, run `Set-PowerCLIConfiguration -ProxyPolicy NoProxy`.

Procedure

- 1 To connect to the vCenter Server system that the SRM server is registered with, run `Connect-VIServer` with the server name and valid credentials.

```
Connect-VIServer -Server vc3.example.com -User 'MyAdministratorUser' -Password 'MyPassword'
```


- 2 To connect to the SRM server registered with the connected vCenter Server system, run `Connect-SrmServer`.

```
$srmConnection = Connect-SrmServer
```

Note If you have previously connected to other vCenter Server systems configured with SRM server support, this cmdlet invocation establishes a connection to their corresponding SRM servers as well.

- 3 (Optional) To use the SRM API, you can call methods of the root object and instances of the objects that those calls return.

```
$srmApi = $srmConnection.ExtensionData
```

Note The root SRM API object is the `ExtensionData` property of the `SrmServer` object.

Protect a Virtual Machine

You can protect a virtual machine by replicating it to a remote SRM site.

Procedure

- 1 Connect to the vCenter Server system that the SRM server is registered with.

```
Connect-VIServer -Server vc3.example.com -User 'MyAdministratorUser' -Password 'MyPassword'
```

- 2 Establish a connection to the local SRM server by providing credentials to the remote SRM site.

```
$srmConnection = Connect-SrmServer -RemoteUser 'MyRemoteUser' -RemotePassword 'MyRemotePassword'
```

- 3 List all protection groups associated with the SRM server.

```
$srmApi = $srmConnection.ExtensionData
$protectionGroups = $srmApi.Protection.ListProtectionGroups()
```

- 4 Associate the *TestVM* virtual machine with the *ProtGroup1* protection group and enable the protection for that virtual machine.

```
$svmToAdd = Get-VM "TestVM"

$targetProtectionGroup = $protectionGroups | where {$_.GetInfo().Name -eq "ProtGroup1" }

$targetProtectionGroup.AssociateVms(@( $svmToAdd.ExtensionData.MoRef ))

# Enable protection for that virtual machine
$protectionSpec = New-Object
```

```

VMware.VimAutomation.Srm.Views.SrmProtectionGroupVmProtectionSpec
$protectionSpec.Vm = $vmToAdd.ExtensionData.MoRef
$protectTask = $targetProtectionGroup.ProtectVms($protectionSpec)
while(-not $protectTask.IsComplete()) { sleep -Seconds 1 }

```

Create a Report of the Protected Virtual Machines

You can create a simple report containing information about the protected virtual machines associated with an SRM server.

Prerequisites

- Verify that you are connected to a vCenter Server system.
- Verify that you are connected to an SRM server.

Procedure

- 1 List all protection groups associated with the SRM server.

```

$srnApi = $srnConnection.ExtensionData
$protectionGroups = $srnApi.Protection.ListProtectionGroups()

```

- 2 Generate a report of the protected virtual machines.

```

$protectionGroups | % {
    $protectionGroup = $_

    $protectionGroupInfo = $protectionGroup.GetInfo()

    # The following command lists the virtual machines associated with a protection group
    $protectedVms = $protectionGroup.ListProtectedVms()
    # The result of the above call is an array of references to the virtual machines at
    the vSphere API
    # To populate the data from the vSphere connection, call the UpdateViewData method on
    each virtual machine view object
    $protectedVms | % { $_.Vm.UpdateViewData() }
    # After the data is populated, use it to generate a report
    $protectedVms | %{
        $output = "" | select VmName, PgName
        $output.VmName = $_.Vm.Name
        $output.PgName = $protectionGroupInfo.Name
        $output
    }
} | Format-Table @{Label="VM Name"; Expression={$_.VmName} }, @{Label="Protection group
name"; Expression={$_.PgName} }

```

Create a Report of the Virtual Machines Associated with All Protection Groups

You can create a simple report containing information about the virtual machines associated with all protection groups.

Prerequisites

- Verify that you are connected to a vCenter Server system.
- Verify that you are connected to an SRM server.

Procedure

- 1 List all protection groups associated with the SRM server.

```
$srmApi = $srmConnection.ExtensionData
$protectionGroups = $srmApi.Protection.ListProtectionGroups()
```

- 2 Generate a report of the virtual machines associated with all protection groups.

```
$protectionGroups | % {
    $protectionGroup = $_

    $protectionGroupInfo = $protectionGroup.GetInfo()

    # The following command lists the virtual machines associated with a protection group
    $vms = $protectionGroup.ListAssociatedVms()
    # The result of the above call is an array of references to the virtual machines at
    the vSphere API
    # To populate the data from the vSphere connection, call the UpdateViewData method on
    each virtual machine view object
    $vms | % { $_.UpdateViewData() }
    # After the data is populated, use it to generate a report
    $vms | %{
        $output = "" | select VmName, PgName
        $output.VmName = $_.Name
        $output.PgName = $protectionGroupInfo.Name
        $output
    }
} | Format-Table @{Label="VM Name"; Expression={$_.VmName} }, @{Label="Protection group
name"; Expression={$_.PgName} }
```

Managing the Site Recovery Manager (SRM) API with VMware PowerCLI

10

Use the Site Recovery Manager SDK for PowerShell to access the Site Recovery Manager (SRM) API with PowerCLI. Starting from VMware PowerCLI version 13.1.0, the Site Recovery Manager SDK provides low-level cmdlets for all available Site Recovery Manager (SRM) API operations.

Read the following topics next:

- [Understanding the Site Recovery Manager SDK for PowerShell](#)
- [Navigating the Site Recovery Manager SDK for PowerShell](#)
- [Connect to a Local and to a Remote Site Recovery Manager \(SRM\) Server](#)
- [Host-Based Replication Scenarios](#)
- [Array-Based Replication Scenarios](#)

Understanding the Site Recovery Manager SDK for PowerShell

You can access the Site Recovery Manager (SRM) API with VMware PowerCLI by using the Site Recovery Manager SDK for PowerShell.

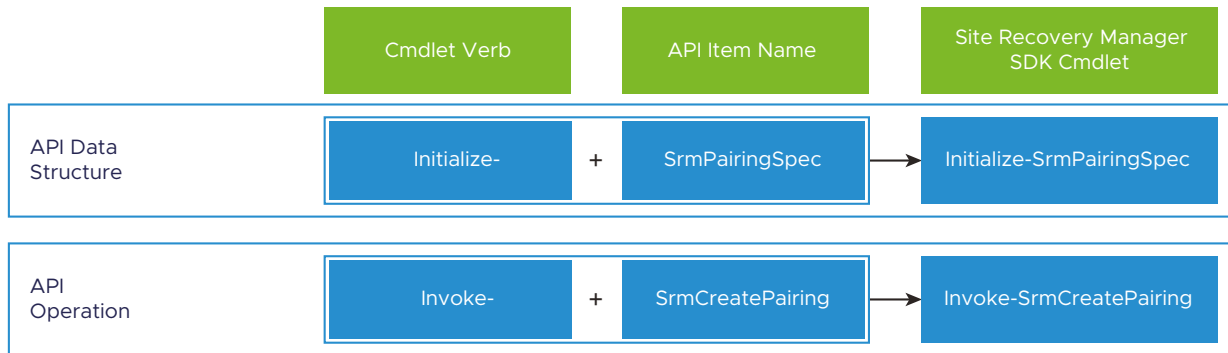
How Does the Site Recovery Manager SDK for PowerShell Work

Starting from VMware PowerCLI 13.1.0, you can use the Site Recovery Manager SDK for PowerShell to communicate with the Site Recovery Manager (SRM) REST APIs.

The `VMware.Sdk.Srm` module contains automatically-generated SDK cmdlets that bind directly to the Site Recovery Manager (SRM) REST APIs. The module functions on a low level and communicates directly with the Site Recovery Manager (SRM) API server.

You can construct Site Recovery Manager SDK cmdlets by using the `Initialize-` and `Invoke-` cmdlet verbs in combination with the names of the Site Recovery Manager (SRM) API data structures or operations.

Figure 10-1. Constructing Site Recovery Manager SDK Cmdlets



- Use `Initialize-` cmdlets to prepare the data structures, if any, for your API calls. These cmdlets function on the client side and do not communicate with the Site Recovery Manager (SRM) API server.
- Use `Invoke-` cmdlets to call the Site Recovery Manager (SRM) API server and invoke the operations.

Navigating the Site Recovery Manager SDK for PowerShell

You can navigate the Site Recovery Manager SDK for PowerShell by using the VMware Developer website, and the `Get-Help`, `Get-Command`, and `Get-SrmOperation` cmdlets.

To use the Site Recovery Manager SDK for PowerShell effectively, you must be familiar with the [VMware Site Recovery Manager REST API reference documentation](#).

How to Find a Site Recovery Manager SDK Cmdlet

There are several ways to find the Site Recovery Manager SDK cmdlets you need.

- In your browser, navigate the Site Recovery Manager (SRM) API reference documentation. The Site Recovery Manager (SRM) API Reference on VMware Developer features ready-to-use Site Recovery Manager SDK code samples that you can paste directly into PowerShell. You can find a PowerCLI code snippet for each operation under *PowerCLI Client SDK Example* in the *Code Samples* section.
- In PowerShell, run `Get-Command -Module VMware.Sdk.Srm` to list all available vSphere Replication cmdlets.
- If you want to use information from the REST API specification to find a cmdlet, you can run `Get-SrmOperation` with the `Method` and `Path` parameters.

For example:

```
Get-SrmOperation -Method GET -Path '/pairings/{pairing_id}/protection-management/groups'
```

The command returns an object with the operation details.

```
Name           : GetAllGroups
CommandInfo    : Invoke-SrmGetAllGroups
ApiName       : ProtectionApi
Path          : /pairings/{pairing_id}/protection-management/groups
Tags          : {protection}
RelatedCommandInfos : {}
Method        : GET
Parameters    : {pairingId, filterProperty, filter, sortBy...}
```

The cmdlet name is listed in the `CommandInfo` property.

How to Get Information About a Cmdlet

To get information about a Site Recovery Manager SDK cmdlet, run `Get-Help` with the cmdlet name. For example:

```
Get-Help Invoke-SrmGetAllGroups -full
```

Connect to a Local and to a Remote Site Recovery Manager (SRM) Server

To invoke operations on the Site Recovery Manager (SRM) API, you must first connect to the Site Recovery Manager (SRM) instances on the local (protected) and the remote (recovery) sites.

Prerequisites

- Verify that VMware Site Recovery Manager (SRM) is deployed to the local (protected) and to the remote (recovery) vCenter Server systems.
- Verify that the Site Recovery Manager (SRM) instances on the local and the remote sites are connected.
- If the certificate of the server you want to connect to is not trusted, verify that your PowerCLI invalid server certificate action settings are configured properly. For more information, see [Configuring PowerCLI Response to Untrusted Certificates](#).
- If you use a proxy server for the connection, verify that it is configured properly, so that the connection is kept alive long enough for tasks to finish.

Note If you do not want to use a proxy server for the connection, run `Set-PowerCLIConfiguration -ProxyPolicy NoProxy`.

Procedure

- 1 Connect to the local and to the remote Site Recovery Manager (SRM) servers.

```
$localSrm = <local_srm_address_or_fqdn>
$srmsConnection = Connect-SrmSdkServer -Server $localSrm`
```

```
-User 'MyLocalVCUser' `
-Password 'MyLocalVCPassWord' `
-RemoteUser 'MyRemoteVCUser' `
-RemotePassword 'MyRemoteVCPassWord'
```

- 2 (Optional) Print the `ConnectedPairing` property of the connection you created.

```
$srmConnection.ConnectedPairing
```

The command prints the `Pairing` object that represents the pair between the local and the remote Site Recovery Manager (SRM) sites.

You will need the `Pairing` object and its properties for various Site Recovery Manager (SRM) API operations.

Host-Based Replication Scenarios

You can use the Site Recovery Manager SDK for PowerShell to manage scenarios for host-based replication (HBR) with VMware PowerCLI. HBR operates at the hypervisor level and allows you to replicate virtual machine data across different hosts or storage systems.

Create Protection Group for Host-Based Replication

To activate the protection of virtual machines, you first create a protection group for host-based replication (HBR).

You create protection groups to enable Site Recovery Manager (SRM) to protect virtual machines. This example shows how to create a protection group for host-based replication by using the Site Recovery Manager SDK for PowerShell.

Prerequisites

- Verify that your SRM connection is authenticated against the protected (local) and recovery (remote) vCenter Server systems. This example uses the `$srmConnection` variable from [Connect to a Local and to a Remote Site Recovery Manager \(SRM\) Server](#).
- The VMs that you want to protect must be replicated by using vSphere Replication.
- You must have the required privileges to create protection groups on the recovery site.

Procedure

- 1 Get the VM that you want to add to a protection group.

```
$replicatedVm = Invoke-SrmGetReplicatedVms `
-PairingId $srmConnection.ConnectedPairing.PairingId `
-VcenterId $srmConnection.ConnectedPairing.LocalVcServer.Id.Guid `
-FilterProperty "Name" `
-Filter "<myVmName>"
```

2 Prepare the HBR protection group spec.

```
$hbrSpec = Initialize-SrmHbrProtectionGroupSpec -Vms $replicatedVm.List[0].Id
```

3 Prepare the protection group creation spec.

For the `ReplicationType` parameter, specify *HBR*.

```
$protectionGroupSpec = Initialize-SrmProtectionGroupCreateSpec -Name
"<myProtectionGroupName>" `
  -ReplicationType HBR `
  -ProtectedVcGuid $srmConnection.ConnectedPairing.LocalVcServer.Id.Guid `
  -HbrSpec $hbrSpec
```

4 Create the protection group task.

```
$task = Invoke-SrmCreateGroup -PairingId $srmConnection.ConnectedPairing.PairingId `
  -ProtectionGroupCreateSpec $protectionGroupSpec
```

5 Monitor the task progress.

```
$task = Invoke-SrmGetTaskInfo -TaskId $task.Id
```

`$task.Status` eventually changes to `SUCCESS`.

6 (Optional) Retrieve the protection group you created.

```
$protectionGroup = Invoke-SrmGetGroup -PairingId $srmConnection.ConnectedPairing.PairingId `
  -GroupId $task.Result
```

What to do next

Create a recovery plan for your protection group.

Create Recovery Plan

To ensure business continuity and facilitate efficient disaster recovery for your virtual machines, you create a recovery plan for a protection group in a given pairing.

Prerequisites

- Verify that your Site Recovery Manager (SRM) connection is authenticated against the protected (local) and recovery (remote) vCenter Server systems. This example uses the `$srmConnection` variable from [Connect to a Local and to a Remote Site Recovery Manager \(SRM\) Server](#).
- You must have created one or more protection groups on the protected (local) site. This example uses the `$protectionGroup` variable from [Create Protection Group for Host-Based Replication](#).
- You must have the required privileges to create recovery plans on the recovery site.

Procedure

1 Create the recovery plan spec.

```
$recPlanSpec = Initialize-SrmRecoveryPlanCreateSpec `
    -Name "<myRecoveryPlanName>" `
    -ProtectedVcGuid $srmConnection.ConnectedPairing.LocalVcServer.Id.Guid `
    -ProtectionGroups $protectionGroup.Id
```

2 Create the recovery plan task.

```
$task = Invoke-SrmCreatePlan -PairingId $srmConnection.ConnectedPairing.PairingId `
    -RecoveryPlanCreateSpec $recPlanSpec
```

3 Monitor the task progress.

```
$task = Invoke-SrmGetTaskInfo -TaskId $task.Id
```

Eventually, the task status changes to `SUCCESS`.

4 (Optional) Retrieve the recovery plan you created.

```
$recoveryPlan = Invoke-SrmGetRecoveryPlan -PairingId
    $srmConnection.ConnectedPairing.PairingId `
    -PlanId $task.Result
```

Array-Based Replication Scenarios

You can use the Site Recovery Manager SDK for PowerShell to manage scenarios for array-based replication (ABR) with VMware PowerCLI. With ABR, storage arrays at the protected site replicate data to peer arrays at the recovery site, typically at the block level.

Configure Array Managers for an Array Pair

Before creating a replicated array pair, you must configure array managers for the protected (local) and recovery (remote) SRM sites.

Configure array managers for the protected and recovery sites to enable the creation of an array pair for array-based replication (ABR). Array managers must be configured so that Site Recovery Manager (SRM) can discover replicated devices, compute datastore groups, and initiate storage operations.

Prerequisites

- Verify that your SRM connection is authenticated against the protected (local) and recovery (remote) vCenter Server systems. This example uses the `$srmConnection` variable from [Connect to a Local and to a Remote Site Recovery Manager \(SRM\) Server](#).
- Verify that the SRM instances on the protected (local) and recovery (remote) sites are connected. This is known as site pairing.

- Verify that storage replication adapters (SRAs) are installed at both sites.
- You must have the required privileges to create array managers on the protected and recovery sites.

Procedure

1 Retrieve the storage replication adapter (SRA) for the local array manager.

```
$localAdapter = (Invoke-SrmGetStorageAdapters -PairingId
$srmsConnection.ConnectedPairing.PairingId `
-SrmId $srmsConnection.ConnectedPairing.LocalSrmServer.Id.Guid `
-FilterProperty "Name" `
-Filter "<myAdapterName>").List[0]
```

2 Prepare the connection parameters for the local SRA.

a Retrieve the `ConnectionParamGroup` object for the local SRA.

```
$localConnectionParamGroup = (Invoke-SrmGetStorageAdapterConnectionParams `
-PairingId $srmsConnection.ConnectedPairing.PairingId `
-SrmId $srmsConnection.ConnectedPairing.LocalSrmServer.Id.Guid `
-StorageAdapterId $localAdapter.Id).List[0]
```

b Provide values to the connection parameters, available in the `$localConnectionParamGroup.ConnectionParams` list.

The set of connection parameters is specific to the used SRA. This example uses parameters specific to the NetApp SRA.

```
($localConnectionParamGroup.ConnectionParams | where {$_.key -eq "username"}).Value =
"<myNetAppUser>"
($localConnectionParamGroup.ConnectionParams | where {$_.key -eq "password"}).Value =
"<myNetAppPass>"
($localConnectionParamGroup.ConnectionParams | where {$_.key -eq
"VolumeInclusion"}).Value = "<myVolume>"
($localConnectionParamGroup.ConnectionParams | where {$_.key -eq "SVM Name"}).Value =
"<myLocalSvmName>"
($localConnectionParamGroup.ConnectionParams | where {$_.key -eq "mgmtIP"}).Value =
"<myNetAppMgmtIP>"
```

3 Prepare the local array manager creation spec.

```
$localArrayManagerSpec = Initialize-SrmArrayManagerSpec -Name "<myLocalArrayManagerName>" `
-StorageAdapter $localAdapter.Id `
-ConnectionParams $localConnectionParamGroup
```

4 Create the local array manager.

```
$task = Invoke-SrmCreateArrayManager -PairingId $srmsConnection.ConnectedPairing.PairingId `
-SrmId $srmsConnection.ConnectedPairing.LocalSrmServer.Id.Guid `
-ArrayManagerSpec $localArrayManagerSpec
```

5 Monitor the task progress.

```
$task = Invoke-SrmGetTaskInfo -TaskId $task.Id
```

\$task.Status eventually changes to SUCCESS.

6 (Optional) Retrieve the local array manager.

The `Result` property of the `$localManagerTask` object contains the ID of the created array manager.

```
$localManagerTask = Invoke-SrmGetTaskInfo -TaskId $task.Id

$localArrayManager = Invoke-SrmGetArrayManager -PairingId
$srnConnection.ConnectedPairing.PairingId `
  -SrmId $srnConnection.ConnectedPairing.LocalSrmServer.Id.Guid `
  -ArrayManagerId $localManagerTask.Result
```

7 Create the remote array manager by using the same approach.

```
$remoteAdapter = (Invoke-SrmGetStorageAdapters -PairingId
$srnConnection.ConnectedPairing.PairingId `
  -SrmId $srnConnection.ConnectedPairing.RemoteSrmServer.Id.Guid `
  -FilterProperty "Name" `
  -Filter "<myAdapterName>").List[0]

$remoteConnectionParamGroup = (Invoke-SrmGetStorageAdapterConnectionParams `
  -PairingId $srnConnection.ConnectedPairing.PairingId `
  -SrmId $srnConnection.ConnectedPairing.RemoteSrmServer.Id.Guid `
  -StorageAdapterId $remoteAdapter.Id).List[0]

($remoteConnectionParamGroup.ConnectionParams | Where-Object {$_.key -eq
"username"}).Value = "<myNetAppUser>"
($remoteConnectionParamGroup.ConnectionParams | Where-Object {$_.key -eq
"password"}).Value = "<myNetAppPass>"
($remoteConnectionParamGroup.ConnectionParams | Where-Object {$_.key -eq
"VolumeInclusion"}).Value = "<myVolume>"
($remoteConnectionParamGroup.ConnectionParams | Where-Object {$_.key -eq "SVM
Name"}).Value = "<myRemoteSvmName>"
($remoteConnectionParamGroup.ConnectionParams | Where-Object {$_.key -eq "mgmtIP"}).Value
= "myNetAppMgmtIP"

$remoteArrayManagerSpec = Initialize-SrmArrayManagerSpec -Name
"<myRemoteArrayManagerName>" `
  -StorageAdapter $remoteAdapter.Id `
  -ConnectionParams $remoteConnectionParamGroup

$task = Invoke-SrmCreateArrayManager -PairingId $srnConnection.ConnectedPairing.PairingId `
  -SrmId $srnConnection.ConnectedPairing.RemoteSrmServer.Id.Guid `
  -ArrayManagerSpec $remoteArrayManagerSpec

#Wait for the create array manager task to complete.
$remoteManagerTask = Invoke-SrmGetTaskInfo -TaskId $task.Id
```

```
$remoteArrayManager = Invoke-SrmGetArrayManager -PairingId
$srmConnection.ConnectedPairing.PairingId `
  -SrmId $srmConnection.ConnectedPairing.RemoteSrmServer.Id.Guid `
  -ArrayManagerId $remoteManagerTask.Result
```

What to do next

Create a replicated array pair for array-based replication.

Create Replicated Array Pair

Before you can create protection groups and recovery plans for ABR, you must create a replicated array pair for the protected (local) and recovery (remote) SRM sites.

Prerequisites

- Verify that your SRM connection is authenticated against the protected and recovery vCenter Server systems. This example uses the *\$srmConnection* variable from [Connect to a Local and to a Remote Site Recovery Manager \(SRM\) Server](#).
- Verify that you configured array managers for the protected (local) and recovery (remote) SRM sites. This example uses the variables from [Configure Array Managers for an Array Pair](#).
- You must have the required privileges to create replicated array pairs on the protected and recovery sites.

Procedure

- 1 Prepare the replicated array pair create spec.

You must specify the local array manager ID, and the storage array keys of the local (protected) and remote (recovery) array managers.

```
$replicatedArrayPairSpec = Initialize-SrmReplicatedArrayPairSpec -ArrayManagerId
$localArrayManager.Id `
  -StorageArrayKey $localArrayManager.StorageArrays[0].StorageArrayKey `
  -PeerStorageArrayKey $remoteArrayManager.StorageArrays[0].StorageArrayKey
```

- 2 Create the replicated array pair.

```
$task = Invoke-SrmCreateReplicatedArrayPair -PairingId
$srmConnection.ConnectedPairing.PairingId `
  -SrmId $srmConnection.ConnectedPairing.LocalSrmServer.Id.Guid `
  -ReplicatedArrayPairSpec $replicatedArrayPairSpec
```

- 3 Monitor the task progress.

```
$task = Invoke-SrmGetTaskInfo -TaskId $task.Id
```

\$task.Status eventually changes to SUCCESS.

4 (Optional) Retrieve the replicated array pair.

The `Result` property of the `$replicatedArrayPairTask` object contains the ID of the replicated array pair.

```
$replicatedArrayPairTask = Invoke-SrmGetTaskInfo -TaskId $task.Id

$replicatedArrayPair = Invoke-SrmGetReplicatedArrayPair -PairingId
$srnConnection.ConnectedPairing.PairingId `
    -SrmId $srnConnection.ConnectedPairing.LocalSrmServer.Id.Guid `
    -ArrayPairId $replicatedArrayPairTask.Result
```

What to do next

Create protection group and recovery plan by using the replicated array pair.

Create Protection Group and Recovery Plan for ABR

To ensure business continuity and facilitate efficient disaster recovery for your virtual machines with array-based replication, you create protection groups and recovery plans.

Prerequisites

- Verify that your SRM connection is authenticated against the protected and recovery vCenter Server systems. This example uses the `$srnConnection` variable from [Connect to a Local and to a Remote Site Recovery Manager \(SRM\) Server](#).
- Verify that you created a replicated array pair for the protected (local) and recovery (remote) SRM sites. This example uses the variables from [Create Replicated Array Pair](#)
- You must have the required privileges to create protection groups and recovery plans on the recovery site.

Procedure

- 1 Discover the storage devices of the replicated array pair.

```
$task = Invoke-SrmDiscoverStorageDevices -PairingId
$srnConnection.ConnectedPairing.PairingId `
    -SrmId $srnConnection.ConnectedPairing.LocalSrmServer.Id.Guid `
    -ArrayPairId $replicatedArrayPair.Id
```

- 2 Monitor the task progress.

```
$task = Invoke-SrmGetTaskInfo -TaskId $task.Id
```

`$task.Status` eventually changes to `SUCCESS`.

3 Retrieve the storage devices from the replicated array pair.

```
$storageDevices = Invoke-SrmGetStorageDevices -PairingId
$srConnection.ConnectedPairing.PairingId `
  -SrmId $srConnection.ConnectedPairing.LocalSrmServer.Id.Guid `
  -ArrayPairId $replicatedArrayPair.Id
```

4 Prepare the ABR protection group spec.

This example uses the replicated array pair with a single storage device and a single datastore created in [Create Replicated Array Pair](#).

```
$abrProtectionGroupSpec = Initialize-SrmAbrProtectionGroupSpec `
  -ReplicatedArrayPair $replicatedArrayPair.Id `
  -Datastores $storageDevices.StorageDevicesData[0].Datastores[0].Id
```

5 Prepare the protection group create spec.

For the `ReplicationType` parameter, specify *ABR*.

```
$protectionGroupSpec = Initialize-SrmProtectionGroupCreateSpec -Name
"<MyAbrProtectionGroup>" `
  -ReplicationType ABR `
  -ProtectedVcGuid $srConnection.ConnectedPairing.LocalVcServer.Id.Guid `
  -AbrSpec $abrProtectionGroupSpec
```

6 Create the ABR protection group.

```
$task = Invoke-SrmCreateGroup -PairingId $srConnection.ConnectedPairing.PairingId `
  -ProtectionGroupCreateSpec $protectionGroupSpec
```

7 Monitor the task progress.

```
$task = Invoke-SrmGetTaskInfo -TaskId $task.Id
```

`$task.Status` eventually changes to `SUCCESS`.

8 (Optional) Retrieve the ABR protection group.

```
$protectionGroup = Invoke-SrmGetGroup -PairingId $srConnection.ConnectedPairing.PairingId
`
  -GroupId $task.Result
```

9 Prepare the recovery plan create spec.

```
$recPlanSpec = Initialize-SrmRecoveryPlanCreateSpec `
  -Name "<MyAbrRecoveryPlan>" `
  -ProtectedVcGuid $srConnection.ConnectedPairing.LocalVcServer.Id.Guid `
  -ProtectionGroups @($protectionGroup.Id)
```

10 Create the recovery plan.

```
$task = Invoke-SrmCreatePlan -PairingId $srmConnection.ConnectedPairing.PairingId `
    -RecoveryPlanCreateSpec $recPlanSpec
```

11 Monitor the task progress.

```
$task = Invoke-SrmGetTaskInfo -TaskId $task.Id
```

`$task.Status` eventually changes to `SUCCESS`.

Managing the NSX Policy API with VMware PowerCLI

11

Manage your networking automation with PowerCLI and the NSX Policy SDK for PowerShell. Starting from PowerCLI 12.6, you can use low-level SDK cmdlets to communicate with all available NSX Policy REST APIs.

Read the following topics next:

- [Understanding the NSX Policy SDK for PowerShell](#)
- [Navigate the NSX Policy SDK for PowerShell](#)
- [Connect to an On-Prem NSX Server](#)
- [Connect to an NSX Server on VMware Cloud on AWS](#)
- [Create a Tier-0 Gateway](#)
- [Create a Tier-1 Gateway](#)
- [Add an Existing Tier-1 Gateway to a Specific Edge Cluster](#)
- [Create a Segment \(On-Prem\)](#)
- [Create a Segment \(VMware Cloud on AWS\)](#)
- [Create a Distributed Firewall Policy](#)

Understanding the NSX Policy SDK for PowerShell

You can manage the NSX Policy API with VMware PowerCLI by using the NSX Policy SDK for PowerShell.

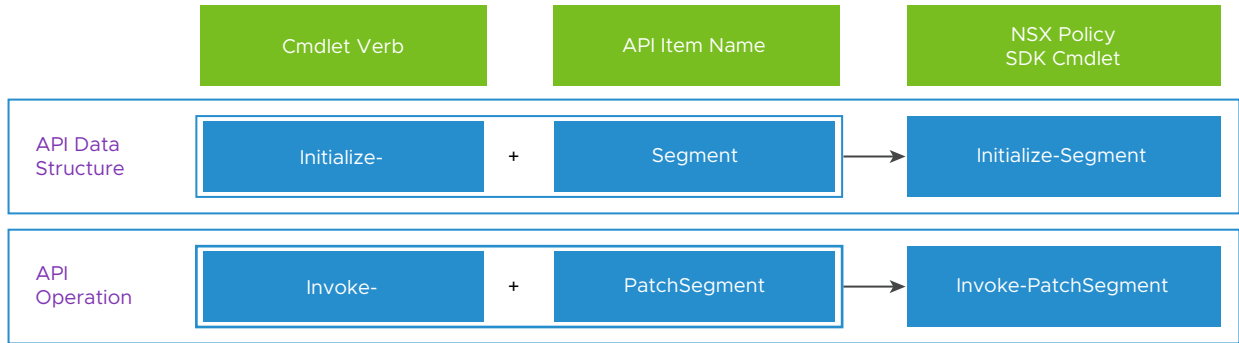
How Does the NSX Policy SDK for PowerShell Work

Starting from VMware PowerCLI 12.6, you can use the NSX Policy SDK for PowerShell to communicate with all available NSX Policy REST APIs.

The `VMware.Sdk.Nsx.Policy` module contains SDK cmdlets that are automatically generated and provide PowerShell bindings to the NSX REST APIs. The `PowerCLINSX` module functions on a low level and communicates directly with the NSX Policy API server.

You can construct NSX Policy SDK cmdlets by using the `Initialize-` and `Invoke-` cmdlet verbs in combination with the names of the API data structures or operations.

Figure 11-1. Constructing NSX Policy SDK Cmdlets



- Use `Initialize-` cmdlets to prepare the data structures, if any, for your API calls. These cmdlets function on the client side and do not communicate with the API server.
- Use `Invoke-` cmdlets to call the NSX Policy API server and invoke the operations.

Navigate the NSX Policy SDK for PowerShell

You can navigate the NSX Policy SDK for PowerShell by using the `Get-NsxOperation` and `Get-Help` cmdlets.

To navigate the NSX Policy SDK for PowerShell, you must be familiar with the NSX Policy API and its reference documentation.

You can access two NSX REST API references on the VMware Developer website:

- The [NSX REST API reference](#) that contains all NSX networking services, for both on-prem and VMware Cloud on AWS.
- The [NSX VMC Policy API reference](#) that contains only the networking services for VMware Cloud on AWS. Here, you can find ready-to-use PowerCLI samples that you can paste directly into your PowerShell console.

The following procedure is based on the NSX REST API reference.

Procedure

- 1 In the NSX REST API reference, navigate to the NSX Policy API operation you want to use. For example, go to **Segments (Fixed) > Create or update a segment**.
- 2 Copy the method and the path of the operation.
- 3 In PowerShell, run `Get-NsxOperation` with the `Method` and `Path` parameters.

For example:

```
Get-NsxOperation -Method PUT -Path '/policy/api/v1/infra/tier-1s/<tier-1-id>/segments/
<segment-id>'
```

The command retrieves a custom object with the NSX operation details.

```
Name           : CreateOrReplaceSegment
CommandInfo    : Invoke-CreateOrReplaceSegment
ApiName        : PolicyConnectivityApi
Path           : /infra/tier-1s/{tier-1-id}/segments/{segment-id}
Tags           : {Policy, Networking, Connectivity, Segments...}
RelatedCommandInfos : {Initialize-Segment}
Method         : PUT
Parameters     : {tier1Id, segmentId, segment}
```

- 4 Copy the PowerCLI cmdlet that corresponds to the API operation.

It is listed in the `CommandInfo` property.

- 5 To retrieve information about the cmdlet , run `Get-Help`.

For example:

```
Get-Help Invoke-CreateOrReplaceSegment -full
```

You can see the full information about the cmdlet such as its description, parameters, and code examples.

Connect to an On-Prem NSX Server

You can connect to an on-prem NSX server by using the `Connect-NsxServer` cmdlet.

Prerequisites

- If the certificate of the server you want to connect to is not trusted, verify that your PowerCLI invalid server certificate action settings are configured properly. For more information, see [Configuring PowerCLI Response to Untrusted Certificates](#).
- If you use a proxy server for the connection, verify that it is configured properly, so that the connection is kept alive long enough for tasks to finish.

Procedure

- ◆ Run `Connect-NsxServer` with the NSX server IP address or FQDN and valid user credentials.

```
Connect-NsxServer -Server <nsx_server_ip_address_or_fqdn> -User <my_username> -Password <my_password>
```

Connect to an NSX Server on VMware Cloud on AWS

You can connect to an NSX server on VMware Cloud on AWS by using the `Connect-NsxVmcServer` cmdlet.

Prerequisites

- Verify that you have a valid API token from VMware Cloud services.
- Verify that you know the name of the SDDC, in which the server you want to connect to resides.

Procedure

- 1 Create a variable with the VMware Cloud on AWS server connection.

```
$vmcServer = Connect-VmcServer -ApiToken $apiToken
```

- 2 Create a variable with the SDDC name.

```
$sddc = Get-VmcSddc -Name $sddcName
```

- 3 Connect to the NSX server on VMware Cloud on AWS.

```
Connect-NsxVmcServer -VmcConnection $vmcServer -VmcSddc $sddc
```

Create a Tier-0 Gateway

You can create a Tier-0 gateway with the NSX Policy SDK for PowerShell by using the `Initialize-Tier0` and `Invoke-PatchTier0` cmdlets.

Prerequisites

Verify that you are connected to an NSX server system.

Procedure

- 1 Prepare the input for the operation.

```
$t0 = Initialize-Tier0 -ArpLimit 5000 -DisplayName $T0GatewayName
```

- 2 Invoke the operation.

```
Invoke-PatchTier0 -Tier0Id $T0GatewayName -Tier0 $t0
```

Create a Tier-1 Gateway

You can create a Tier-1 gateway with the NSX Policy SDK for PowerShell by using the `Initialize-Tier1` and `Invoke-PatchTier1` cmdlets.

Prerequisites

- Verify that you are connected to an NSX server system.
- Verify that your system has Tier-0 gateway(s).

Procedure

- 1 Retrieve the Tier-0 paths.

```
$t0s = Invoke-ListTier0s
$t0 = $t0s.Results | where {$_.DisplayName -eq $T0GatewayName}
```

- 2 Prepare the input for the operation.

```
$t1 = Initialize-Tier1 -ArpLimit 5000 -DisplayName $T1GatewayName -Tier0Path $t0.Path
```

- 3 Invoke the operation.

```
Invoke-PatchTier1 -Tier1Id $T1GatewayName -Tier1 $t1
```

Add an Existing Tier-1 Gateway to a Specific Edge Cluster

You can add an existing Tier-1 gateway to a specific Edge Cluster with the NSX Policy SDK for PowerShell by using the `Invoke-PatchTier1LocaleServices` cmdlet.

Prerequisites

- Verify that you are connected to an NSX server system.
- Verify that you know the Edge Cluster name of the Edge Cluster to which you want to add the Tier-1 gateway.

Procedure

- 1 Retrieve the Edge Cluster information.

```
$edgeClusters = Invoke-ListEdgeClustersForEnforcementPoint -SiteId default
                -EnforcementPointId default
$edgeCluster = $edgeClusters.Results | where {$_.Displayname -eq $edgeClusterName}
```

- 2 Create a client-side object for locale service with the Edge Cluster path.

```
$localeService = Initialize-LocaleServices -EdgeClusterPath $edgeCluster.Path
                -DisplayName default -Id default
```

- 3 Invoke the operation.

```
Invoke-PatchTier1LocaleServices -Tier1Id $T1.Id -LocaleServices $localeService
                -LocaleServicesId default
```

Create a Segment (On-Prem)

You can create a segment on an on-prem NSX server system by using the `Invoke-PatchInfraSegment` cmdlet.

To create an on-prem (flexible) segment, you must specify a transport zone and assign a subnet.

Prerequisites

- Verify that you are connected to an on-prem NSX server system.
- Verify that your system has Tier-1 gateway(s).

Procedure

- 1 Retrieve transport zone details.

```
$tzs = Invoke-ListTransportZonesForEnforcementPoint -EnforcementpointId default -SiteId
default
$tz = $tzs.Results | where {$_.DisplayName -eq $TransportZone}
```

- 2 Prepare the input for the segment subnet.

```
$segmentSubnet = Initialize-SegmentSubnet -GatewayAddress $GatewayCIDR
```

- 3 Retrieve the Tier-1 paths.

```
$t1s = Invoke-ListTier1
$t1 = $t1s.Results | where {$_.DisplayName -eq $T1GatewayName}
```

- 4 Prepare the input for the segment.

```
$segment = Initialize-Segment -DisplayName $SegmentName -TransportZonePath $tz.Path `
-Subnets $segmentSubnet -ConnectivityPath $t1.Path
```

- 5 Invoke the operation.

```
Invoke-PatchInfraSegment -Segment $segment -SegmentId $SegmentName
```

Create a Segment (VMware Cloud on AWS)

You can create a segment for an NSX server system on VMware Cloud on AWS by using the `Invoke-CreateOrReplaceSegment` cmdlet.

Prerequisites

- Verify that you are connected to an NSX server system on VMware Cloud on AWS.
- Verify that your system has Tier-1 gateway(s).

Procedure

- 1 Prepare the input for the segment subnet.

```
$segmentSubnet = Initialize-SegmentSubnet -GatewayAddress $GatewayCIDR
```

- 2 Prepare the input for the segment.

```
$segment = Initialize-Segment -DisplayName $SegmentName -Subnets @($segmentSubnet)
```

3 Invoke the operation.

```
Invoke-CreateOrReplaceSegment -Tier1Id $T1GatewayName -SegmentId $SegmentName -Segment
$segment
```

Create a Distributed Firewall Policy

You can create a distributed firewall policy for an NSX server system by using the `Invoke-PatchSecurityPolicyForDomain` cmdlet.

Prerequisites

- Verify that you are connected to an NSX server system.

Procedure

1 Create lookup object variables.

```
$serviceList = @("SSH", "HTTP")
$sourceGroups = @("ANY")
$destinationGroups = @("MyGroupName")
```

2 Look up groups and services.

```
$allServices = Invoke-ListServicesForTenant
$ServicePathList = @()
foreach ($serv in $serviceList) {
    $s = $allServices.Results | where {$_.DisplayName -eq $serv}
    $ServicePathList += $s.Path
}

$allGroups = Invoke-ListGroupForDomain -DomainId default
$sourceGroupList = @()
foreach ($gp in $sourceGroups) {
    if ($gp -eq "ANY") {
        $SourceGroupList += "ANY"
    } else {
        $g = $allGroups.Results | where {$_.DisplayName -eq $gp}
        $SourceGroupList += $g.Path
    }
}

$destinationGroupList = @()
foreach ($gp in $destinationGroups) {
    if ($gp -eq "ANY") {
        $DestinationGroupList += "ANY"
    } else {
        $g = $allGroups.Results | where {$_.DisplayName -eq $gp}
        $DestinationGroupList += $g.Path
    }
}
```

3 Prepare the input for the policy rule.

```
$rule = Initialize-Rule -DisplayName $ruleName -Id $ruleName -SourceGroups  
$sourceGroupList -DestinationGroups $destinationGroupList -Services $servicePathList  
-Action "ALLOW"
```

4 Prepare the input for the security policy.

```
$securityPolicy = Initialize-SecurityPolicy -DisplayName $policyName -Rules @($rule)
```

5 Invoke the operation.

```
Invoke-PatchSecurityPolicyForDomain -DomainId default -SecurityPolicyId $policyName  
-SecurityPolicy $securityPolicy
```

Managing VMware Cloud Director with VMware PowerCLI

12

To help you get started with VMware PowerCLI, this documentation provides a set of scripts that illustrate basic and advanced tasks in cloud administration.

- [Connect to a VMware Cloud Director Server](#)

To run cmdlets on a VMware Cloud Director server and perform administration or monitoring tasks, you must establish a connection to the server.

- [Create and Manage Organizations](#)

Organizations provide resources to a group of users and set policies that determine how users can consume those resources. Create and manage organizations for each group of users that requires its own resources, policies, or both.

- [Create and Manage Organization Virtual Data Centers](#)

To allocate resources to an organization, you need to create an organization virtual data center (vDC). When the demands of the organization change, you can modify or remove the organization vDC.

- [Filter and Retrieve Organization Virtual Data Center Networks](#)

To generate reports about organization vDC networks, you need to retrieve the respective organization vDC networks. You can use search criteria to filter the results returned by `Get-OrgVdcNetwork`.

- [Import a vApp Template from the Local Storage](#)

To make an OVF package from your local storage available to other cloud users, you can import the package and save it as a vApp template in a catalog.

- [Create a vApp Template from a vApp](#)

Creating vApp templates from vApps in the cloud might minimize future efforts for cloning vApps. You can use the templates later to create vApps that are based on the source vApp.

- [Import a vApp from vSphere](#)

To make a virtual machine from the underlying vSphere infrastructure available to your VMware Cloud Director server, you can import it and save it as a vApp.

- [Create and Modify a vApp](#)

You can use vApp templates to instantiate vApps. After creating the vApp, you can modify its settings to minimize the consumption of computing and storage resources.

- [Manage Virtual Machines with vApps](#)

For a large-scale approach to administration, you can start, stop, or restart virtual machines or their guest operating systems by running cmdlets on the associated vApps.

- [Manage Virtual Machines and Their Guest Operating Systems](#)

For a targeted approach to administration, you can use the `CIVM` and `CIVMGuest` cmdlets to handle lifecycle operations for one or more virtual machines.

- [Retrieve a List of the Internal and External IP Addresses of Virtual Machines in vApps](#)

When managing vApps in the cloud, you might need to obtain information about the NIC settings of the associated virtual machines.

- [Create and Manage Access Control Rules](#)

By defining access control rules you can assign levels of access to separate users, user groups, or everyone in the organization. You can define access control rules for catalogs and vApps.

- [Filter and Retrieve vApp Networks](#)

To generate reports about vApp networks, you need to retrieve the respective vApp networks. You can use search criteria to filter the results returned by `Get-CIVAppNetwork`.

- [Create vApp Networks for a Selected vApp](#)

To define how the virtual machines in a vApp connect to each other and access other networks, you need to create a vApp network. When creating the vApp network, you can select the settings for the network, or adopt them from an organization policy.

- [Modify or Remove vApp Networks](#)

Based on the type of the vApp network, you can configure various network settings, such as DNS, static IP pools, and firewalls. If you no longer need a vApp network, you can remove it.

Connect to a VMware Cloud Director Server

To run cmdlets on a VMware Cloud Director server and perform administration or monitoring tasks, you must establish a connection to the server.

You can have more than one connection to the same server. For more information, see [Managing Default Server Connections in PowerCLI](#).

If your login credentials contain non-alphanumeric characters, you might need to escape them. For more information, see [Processing Non-alphanumeric Characters in PowerCLI](#).

Prerequisites

- If the certificate of the server you want to connect to is not trusted, verify that your PowerCLI invalid server certificate action settings are configured properly. For more information, see [Configuring PowerCLI Response to Untrusted Certificates](#).

- If you use a proxy server for the connection, verify that it is configured properly, so that the connection is kept alive long enough for tasks to finish.

Note If you do not want to use a proxy server for the connection, run `Set-PowerCLIConfiguration -ProxyPolicy NoProxy`.

Procedure

- ◆ Run `Connect-CIServer` with the server name and valid credentials.

```
Connect-CIServer -Server cloud.example.com -User 'MyAdministratorUser' -Password 'MyPassword'
```

Create and Manage Organizations

Organizations provide resources to a group of users and set policies that determine how users can consume those resources. Create and manage organizations for each group of users that requires its own resources, policies, or both.

Prerequisites

Verify that you are connected to a VMware Cloud Director server as a provider administrator.

Procedure

- 1 Generate a customized report for all organizations on the server.

```
Get-Org | Select Name, Enabled, StoredVMQuota, DeployedVMQuota
```

- 2 Add a new organization on the server and provide a name and a full name for it.

```
New-Org -Name 'MyOrg1' -FullName 'My Organization 1'
```

By default, the new organization is enabled. Enabling the organization lets users log in.

- 3 Add a description for the new organization.

```
Get-Org -Name 'MyOrg1' | Set-Org -Description "This organization provides resources to John Doe."
```

- 4 Deactivate and remove the new organization.

```
Get-Org -Name 'MyOrg1' | Set-Org -Enabled $false | Remove-Org
```

Create and Manage Organization Virtual Data Centers

To allocate resources to an organization, you need to create an organization virtual data center (vDC). When the demands of the organization change, you can modify or remove the organization vDC.

Prerequisites

- Verify that you are connected to a VMware Cloud Director server as a provider administrator.
- Verify that at least one enabled provider vDC is available on the server.

Procedure

- 1 Create a new organization vDC using the Pay As You Go model for resource allocation.

```
$myOrg = Get-Org -Name 'MyOrg1'
$myPVdc = Get-ProviderVdc -Name 'MyProvidervDC'
New-OrgVdc -Name 'MyOrgvDC' -AllocationModelPayAsYouGo -Org $myOrg -ProviderVdc $myPVdc
-VMCPUCoreMHz 1000
```

To create the organization vDC, VMware Cloud Director PowerCLI uses a default configuration based on the selected resource allocation model.

- VMMaxCount is set to 100
 - NetworkMaxCount is set to 1024
 - The vDC is automatically enabled
 - Thin provisioning is deactivated
 - Fast provisioning is deactivated
 - NicMaxCount is set to \$null (unlimited)
 - MemoryGuaranteedPercent is set to 100
 - CpuGuaranteedPercent is set to 0
- 2 Modify the vCPU speed setting for the virtual machines in the organization vDC.

```
Get-OrgVdc -Name 'MyOrgVdc' | Set-OrgVdc -VMCpuCoreMhz 2000
```

- 3 Enable fast provisioning for the virtual machines in the organization vDC.

```
Get-OrgVdc -Name 'MyOrgVdc' | Set-OrgVdc -UseFastProvisioning $true
```

- 4 Deactivate and remove the new organization vDC.

```
Get-OrgVdc -Name 'MyOrgVdc' | Set-OrgVdc -Enabled $false | Remove-OrgVdc
```

Filter and Retrieve Organization Virtual Data Center Networks

To generate reports about organization vDC networks, you need to retrieve the respective organization vDC networks. You can use search criteria to filter the results returned by `Get-OrgVdcNetwork`.

Prerequisites

Verify that you are connected to a VMware Cloud Director server.

Procedure

- ◆ Get all organization vDC networks for the organization named *MyOrgVdc*.

```
Get-OrgVdc -Name 'MyOrgVdc' | Get-OrgVdcNetwork
```

- ◆ Get the organization vDC network that is named *MyOrgVdcNetwork*.

```
Get-OrgVdc -Name 'MyOrgVdc' | Get-OrgVdcNetwork -Name 'MyOrgVdcNetwork'
```

Import a vApp Template from the Local Storage

To make an OVF package from your local storage available to other cloud users, you can import the package and save it as a vApp template in a catalog.

Prerequisites

Verify that you are connected to a VMware Cloud Director server.

Procedure

- 1 Retrieve the catalog to which you want to add the imported vApp template.

```
$myCatalog = Get-Catalog -Name 'MyCatalog'
```

- 2 Retrieve the organization virtual data center (vDC) to which you want to add the imported vApp template.

```
$myOrgVdc = Get-OrgVdc -Name 'MyOrgVdc'
```

- 3 Import a virtual machine from your local storage and save it as a vApp template in the cloud.

```
Import-CIVAppTemplate -SourcePath 'C:\OVFs\WindowsXP\WindowsXP.ovf' -Name  
'MyWindowsXPVAppTemplate' -OrgVdc $myOrgVdc -Catalog $myCatalog
```

Create a vApp Template from a vApp

Creating vApp templates from vApps in the cloud might minimize future efforts for cloning vApps. You can use the templates later to create vApps that are based on the source vApp.

Prerequisites

Verify that you are connected to a VMware Cloud Director server.

Procedure

- 1 Retrieve the source vApp for the vApp template that you want to create.

```
$myVApp = Get-CIVApp -Name 'MyVApp'
```

- 2 If the source vApp is running, stop it.

```
$myVApp = Stop-CIVApp -VApp $myVApp
```

- 3 Retrieve the catalog to which you want to add the new vApp template.

```
$myCatalog = Get-Catalog -Name 'MyCatalog'
```

- 4 Retrieve the organization vDC to which you want to add the new vApp template.

```
$myOrgVdc = Get-OrgVdc -Name 'MyOrgVdc'
```

- 5 Create a vApp template.

```
New-CIVAppTemplate -Name 'MyVAppTemplate' -VApp $myVApp -OrgVdc $myOrgVDC -Catalog $myCatalog
```

- 6 Start the source vApp.

```
$myVApp = Start-CIVApp -VApp $myVApp
```

What to do next

Create a vApp from the template and modify the vApp. See [Create and Modify a vApp](#).

Import a vApp from vSphere

To make a virtual machine from the underlying vSphere infrastructure available to your VMware Cloud Director server, you can import it and save it as a vApp.

Prerequisites

- Verify that you are connected to a VMware Cloud Director server as a provider administrator.
- Verify that you are connected to a vCenter Server system.

Procedure

- 1 Retrieve the vSphere virtual machine that you want to import.

```
$myVm = Get-VM -Name 'MyVMToImport'
```

- 2 Retrieve the organization vDC to which you want to import the virtual machine.

```
$myOrgVdc = Get-OrgVdc -Name 'MyOrgVdc'
```

- 3 Import the virtual machine and store it as a vApp.

```
Import-CIVApp -VM $myVm -OrgVdc $myOrgVdc
```

Create and Modify a vApp

You can use vApp templates to instantiate vApps. After creating the vApp, you can modify its settings to minimize the consumption of computing and storage resources.

Prerequisites

Verify that you are connected to a VMware Cloud Director server.

Procedure

- 1 Retrieve the organization vDC to which you want to add the new vApp.

```
$myOrgVdc = Get-OrgVdc -Name 'MyOrgVdc'
```

- 2 Retrieve the source vApp template for your new vApp.

```
$myVAppTemplate = Get-CIVAppTemplate -Name 'MyVAppTemplate'
```

- 3 Create your new vApp.

```
$myVApp = New-CIVApp -Name 'MyVApp' -VAppTemplate $myVAppTemplate -OrgVdc $myOrgVDC
```

By default, the vApp is powered off.

- 4 Renew the runtime lease for the new vApp and set it to 12 hours.

```
Set-CIVApp -VApp $myVApp -RuntimeLease "12:0:0" -RenewLease
```

To set leases, you can use the *days.hours:minutes:seconds* syntax.

- 5 Start the new vApp.

```
Start-CIVApp -VApp $myVApp
```

Manage Virtual Machines with vApps

For a large-scale approach to administration, you can start, stop, or restart virtual machines or their guest operating systems by running cmdlets on the associated vApps.

Prerequisites

Verify that you are connected to a VMware Cloud Director server.

Procedure

- 1 Power on all virtual machines in all vApps with names starting with *MyVApp*.

```
Get-CIVApp -Name 'MyVApp*' | Start-CIVApp
```

- 2 Suspend all virtual machines in all vApps with names starting with *YourVApp*.

```
Get-CIVApp -Name 'YourVApp*' | Suspend-CIVApp
```

- 3 Power off all virtual machines in the vApp named *MyVApp1*.

```
Get-CIVApp -Name 'MyVApp1' | Stop-CIVApp
```

- 4 Shut down the guest operating systems of all virtual machines in the vApp named *MyVApp2*.

```
Get-CIVApp -Name 'MyVApp2' | Stop-CIVAppGuest
```

- 5 Restart the guest operating systems of all virtual machines in the vApp named *MyVApp3*.

```
Get-CIVApp -Name 'MyVApp3' | Restart-CIVAppGuest
```

- 6 Reset all virtual machines in the vApp.

```
Get-CIVApp -Name 'MyVApp4' | Restart-CIVApp
```

Manage Virtual Machines and Their Guest Operating Systems

For a targeted approach to administration, you can use the `CIVM` and `CIVMGuest` cmdlets to handle lifecycle operations for one or more virtual machines.

Prerequisites

Verify that you are connected to a VMware Cloud Director server.

Procedure

- 1 Retrieve all virtual machines with names starting with *MyVM* and power them on.

```
Get-CIVM -Name 'MyVM*' | Start-CIVM
```

- 2 Suspend all virtual machines with names starting with *YourVM*.

```
Get-CIVM -Name 'YourVM*' | Suspend-CIVM
```

- 3 Power off the virtual machine named *MyVM1*.

```
Get-CIVM -Name 'MyVM1' | Stop-CIVM
```

- 4 Shut down the guest operating system of the virtual machine named *MyVM2*.

```
Get-CIVM -Name 'MyVM2' | Stop-CIVMGuest
```

- 5 Restart the guest operating system of the virtual machine named *MyVM3*.

```
Get-CIVM -Name 'MyVM3' | Restart-CIVMGuest
```

- 6 Reset the nonresponsive virtual machine named *MyVM4*.

```
Get-CIVM -Name 'MyVM4' | Restart-CIVM
```

Retrieve a List of the Internal and External IP Addresses of Virtual Machines in vApps

When managing vApps in the cloud, you might need to obtain information about the NIC settings of the associated virtual machines.

Prerequisites

Verify that you are connected to a VMware Cloud Director server.

Procedure

- 1 Retrieve the organization for which you want to generate the report.

```
$myOrg = Get-Org -Name 'MyOrg'
```

- 2 Retrieve all vApps in the organization.

```
$vApps = Get-CIVApp -Org $myOrg
```

- 3 Populate an array with the information that you want to report.

```
$vAppNetworkAdapters = @()
foreach ($vApp in $vApps) {
    $vms = Get-CIVM -VApp $vApp
    foreach ($vm in $vms) {
        $networkAdapters = Get-CINetworkAdapter -VM $vm
        foreach ($networkAdapter in $networkAdapters) {
            $vAppNicInfo = New-Object "PSCustomObject"
            $vAppNicInfo | Add-Member -MemberType NoteProperty -Name VAppName
            -Value $vApp.Name
            $vAppNicInfo | Add-Member -MemberType NoteProperty -Name VMName
            -Value $vm.Name
            $vAppNicInfo | Add-Member -MemberType NoteProperty -Name NIC
            -Value ("NIC" + $networkAdapter.Index)
            $vAppNicInfo | Add-Member -MemberType NoteProperty -Name
            ExternalIP -Value $networkAdapter.ExternalIpAddress
            $vAppNicInfo | Add-Member -MemberType NoteProperty -Name
            InternalIP -Value $networkAdapter.IpAddress
```



```

        $vAppNetworkAdapters += $vAppNicInfo
    }
}
}

```

Running this script retrieves the names of the virtual machines and their associated vApp, the IDs of the NICs of the virtual machines, and external, and internal IP addresses of the NICs.

- 4 Display the report on the screen.

```
$vAppNetworkAdapters
```

Create and Manage Access Control Rules

By defining access control rules you can assign levels of access to separate users, user groups, or everyone in the organization. You can define access control rules for catalogs and vApps.

Prerequisites

Verify that you are connected to a VMware Cloud Director server.

Procedure

- 1 Create a new rule for accessing the vApp named *MyVApp*.

```
New-ClAccessControlRule -Entity 'MyVApp' -EveryoneInOrg -AccessLevel "Read"
```

All users in the organization have read-only access to the vApp.

- 2 Modify the access rule for a trusted user who needs full control over *MyVApp*.

```
New-ClAccessControlRule -Entity 'MyVApp' -User "MyAdvancedUser" -AccessLevel "FullControl"
```

- 3 Restrict the full control access of *MyAdvancedUser* to read/write access.

```
$accessRule = Get-ClAccessControlRule -Entity 'MyVApp' -User 'MyAdvancedUser'
$accessRule | Set-ClAccessControlRule -AccessLevel "ReadWrite"
```

- 4 Remove the custom rule that you created earlier for *MyAdvancedUser*.

```
$accessRule | Remove-ClAccessControlRule
```

Filter and Retrieve vApp Networks

To generate reports about vApp networks, you need to retrieve the respective vApp networks. You can use search criteria to filter the results returned by `Get-ClVAppNetwork`.

Prerequisites

Verify that you are connected to a VMware Cloud Director server.

Procedure

- ◆ Get the vApp network named *MyVAppNetwork*.

```
Get-CIVAppNetwork -Name 'VAppNetwork'
```

- ◆ Get all vApp networks for the vApp named *MyVApp*.

```
Get-CIVApp -Name 'MyVApp' | Get-CIVAppNetwork
```

- ◆ Get all vApp networks of connection type direct and direct fenced.

```
Get-CIVAppNetwork -ConnectionType Direct, DirectFenced
```

- ◆ Get all direct vApp networks that connect to the organization vDC network named *MyOrgVdcNetwork*.

```
Get-OrgVdcNetwork -Name 'MyOrgVdcNetwork' | Get-CIVAppNetwork -ConnectionType Direct
```

Create vApp Networks for a Selected vApp

To define how the virtual machines in a vApp connect to each other and access other networks, you need to create a vApp network. When creating the vApp network, you can select the settings for the network, or adopt them from an organization policy.

To address multiple networking scenarios for a vApp, you can create multiple vApp networks.

What to read next

- [Create an Isolated vApp Network](#)

When you do not want the virtual machines in a vApp to connect to objects outside the vApp, you must create an isolated vApp network.

- [Create an NAT Routed vApp Network](#)

To provide a vApp network with DHCP, firewall, NAT, and VPN services, you must create it as an NAT routed vApp network.

- [Create a Direct vApp Network](#)

To establish a network connection between the virtual machines in a vApp and an organization network, you need to create a direct vApp network.

Create an Isolated vApp Network

When you do not want the virtual machines in a vApp to connect to objects outside the vApp, you must create an isolated vApp network.

Prerequisites

Verify that you are connected to a VMware Cloud Director server.

Procedure

- 1 Retrieve the vApp for which you want to create a vApp network.

```
$myVApp = Get-CIVApp -Name 'MyVApp'
```

- 2 Create the new vApp network with a selected gateway and network mask.

```
New-CIVAppNetwork -VApp $myVApp -Name 'MyVAppInternalNetwork' -Routed -Gateway  
'192.168.2.1' -Netmask '255.255.255.0' -ParentOrgVdcNetwork $null
```

By default, the vApp network has an enabled firewall.

Create an NAT Routed vApp Network

To provide a vApp network with DHCP, firewall, NAT, and VPN services, you must create it as an NAT routed vApp network.

Prerequisites

Verify that you are connected to a VMware Cloud Director server.

Procedure

- 1 Retrieve the vApp for which you want to create a vApp network.

```
$myVApp = Get-CIVApp -Name 'MyVApp'
```

- 2 Retrieve the organization vDC network to which you want to connect the vApp network.

```
$myOrgVdcNetwork = Get-OrgVdcNetwork -Name 'MyOrgVdcNetwork'
```

- 3 Create the new vApp network with a gateway and network mask, defined pool of static IP addresses, and a deactivated firewall.

```
New-CIVAppNetwork -VApp $myVApp -ParentOrgVdcNetwork $myOrgVdcNetwork -Name  
'MyVAppInternalNetwork' -Routed -Gateway '192.168.2.1' -Netmask '255.255.255.0'  
-DisableFirewall -StaticIPPool "192.168.2.100 - 192.168.2.199"
```

If you do not run `New-CIVAppNetwork` with the `DisableFirewall` parameter, the new vApp network has an active firewall by default.

Create a Direct vApp Network

To establish a network connection between the virtual machines in a vApp and an organization network, you need to create a direct vApp network.

Prerequisites

Verify that you are connected to a VMware Cloud Director server.

Procedure

- 1 Retrieve the vApp for which you want to create a vApp network.

```
$myVApp = Get-CIVApp -Name 'MyVApp'
```

- 2 Retrieve the organization vDC network that you want to connect to.

```
$myOrgVdcNetwork = Get-OrgVdcNetwork -Name 'MyOrgVdcNetwork'
```

- 3 Create a direct vApp network that connects to the selected organization vDC network.

```
New-CIVAppNetwork -VApp $myVApp -Direct -ParentOrgVdcNetwork $myOrgVdcNetwork
```

By default, the new vApp network has an enabled firewall.

Modify or Remove vApp Networks

Based on the type of the vApp network, you can configure various network settings, such as DNS, static IP pools, and firewalls. If you no longer need a vApp network, you can remove it.

Prerequisites

Verify that you are connected to a VMware Cloud Director server.

Procedure

- 1 Retrieve the vApp for which you want to modify vApp networks.

```
$myVApp = Get-CIVApp -Name 'MyVApp'
```

- 2 Modify the settings for DNS and static IP pool for the vApp network named *MyVAppNetwork*.

```
Get-CIVAppNetwork -VApp $myVApp -Name 'MyVAppNetwork' | Set-CIVAppNetwork -PrimaryDns 10.17.0.94 -SecondaryDns 10.17.0.95 -DnsSuffix 'my.domain.com' -StaticIPPool "10.151.168.1 - 10.151.169.240"
```

- 3 (Optional) Remove *MyVAppNetwork*.

```
$myVApp | Get-CIVAppNetwork -Name 'MyVAppNetwork' | Remove-CIVAppNetwork
```

- 4 (Optional) Remove all isolated vApp networks for the vApp named *MyVApp*.

```
$myVApp | Get-CIVAppNetwork -ConnectionType Isolated | Remove-CIVAppNetwork
```

- 5 Retrieve the organization vDC network named *MyOrgVdcNetwork1*.

```
$myOrgVdcNetwork1 = Get-OrgVdcNetwork -Name 'MyOrgVdcNetwork1'
```

- 6 Retrieve the organization vDC network named *MyOrgVdcNetwork2*.

```
$myOrVdcgNetwork2 = Get-OrgVdcNetwork -Name 'MyOrgVdcNetwork2'
```

- 7 Redirect all vApp networks that connect to *MyOrgVdcNetwork1* to connect to *MyOrgVdcNetwork2*.

```
Get-CIVAppNetwork -ParentOrgVdcNetwork $myOrgVdcNetwork1 | Set-CIVAppNetwork  
-ParentOrgVdcNetwork $myOrgVdcNetwork2 -NatEnabled $false -FirewallEnabled $false
```

The operation deactivates the firewall and NAT routing for all vApp networks that are connected to *MyOrgVdcNetwork1*.

Managing vSphere Update Manager with VMware PowerCLI

13

To help you get started with VMware PowerCLI, this documentation provides a set of sample scripts that illustrate basic and advanced tasks in vSphere Update Manager administration.

The vSphere Update Manager module provides a set of cmdlets for downloading software patches, creating and modifying baselines, and for scanning and remediating virtual machines or hosts.

Read the following topics next:

- [Connect to a vCenter Server System](#)
- [Create Patch Baselines](#)
- [Attach and Detach Baselines](#)
- [Scan a Virtual Machine](#)
- [Check Virtual Machine Baseline Status](#)
- [Stage Patches](#)
- [Remediate a Virtual Machine](#)
- [Upgrade Virtual Machine Hardware](#)
- [Remediate a Cluster](#)
- [Remediate a Host](#)
- [Download Patches and Scan Objects](#)

Connect to a vCenter Server System

To run Update Manager PowerCLI cmdlets on vSphere, you must establish a connection to an ESXi host or a vCenter Server system.

You can have more than one connection to the same server. For more information, see [Managing Default Server Connections in PowerCLI](#).

If your login credentials contain non-alphanumeric characters, you might need to escape them. For more information, see [Processing Non-alphanumeric Characters in PowerCLI](#).

Prerequisites

- If the certificate of the server you want to connect to is not trusted, verify that your PowerCLI invalid server certificate action settings are configured properly. For more information, see [Configuring PowerCLI Response to Untrusted Certificates](#).
- If you use a proxy server for the connection, verify that it is configured properly, so that the connection is kept alive long enough for tasks to finish.

Note If you do not want to use a proxy server for the connection, run `Set-PowerCLIConfiguration -ProxyPolicy NoProxy`.

Procedure

- ◆ Run `Connect-VIServer` with the server name and valid credentials.

```
Connect-VIServer -Server vc3.example.com -User 'MyAdministratorUser' -Password 'MyPassword'
```

Create Patch Baselines

You can apply patch baselines to hosts. Depending on the patch criteria you select, patch baselines can be either dynamic or fixed.

Patch data in dynamic baselines changes depending on the criteria you specify each time Update Manager downloads new patches. Fixed baselines contain only the patches you have selected, regardless of new patch downloads.

Prerequisites

Verify that you are connected to a vCenter Server system.

Procedure

- 1 Retrieve all host patches released after 1 Jan 2015 for ESXi products, and create a fixed baseline named *Static Baseline*, containing the retrieved patches.

```
$patches = Get-Patch -After "1 Jan 2015" -Product "ESXi"
$staticBaseline = New-PatchBaseline -Static -Name "Static Baseline" -IncludePatch $patches
```

- 2 Create a critical dynamic baseline named *Dynamic Baseline* by using a fetch-all query.

```
$criticalPatchBaseline = New-PatchBaseline -Dynamic -Name "Dynamic Baseline"
-SearchPatchSeverity Critical
```

- 3 Create an extension baseline that contains all available extensions.

```
$extensions = Get-Patch -BundleType Extension
New-PatchBaseline -Static -Name "Extension Baseline" -Extension -IncludePatch $extensions
```

Attach and Detach Baselines

You can attach baselines to individual objects and to container objects, such as folders, hosts, clusters, and data centers. Attaching a baseline to a container object attaches the baseline to all objects in the container.

Prerequisites

Verify that you are connected to a vCenter Server system.

Procedure

- 1 Attach the host patch baselines stored in the provided variables to the host named *Host*.

```
Add-EntityBaseline -Baseline $staticBaseline, $criticalPatchBaseline -Entity Host
```

- 2 Detach the two baselines from the host.

```
Remove-EntityBaseline -Baseline $dynamicBaseline, $staticBaseline -Entity Host
```

Scan a Virtual Machine

You can scan a virtual machine against the baselines attached to it or inherited by its parent object.

Prerequisites

Verify that you are connected to a vCenter Server system.

Procedure

- 1 Initialize scanning on a virtual machine that is named *VM* against baselines containing virtual machine hardware upgrades and VMware Tools upgrades.

```
$task = Test-Compliance -Entity VM -UpdateType VmHardwareUpgrade, VmToolsUpgrade -RunAsync
```

The command initializes a task on the server, returns a snapshot object of the initial state of the task, and saves it in the *\$task* variable.

- 2 View the initial status of the scanning task.

```
$task
```

Note The task object is not updated with the actual state of the task process running on the server. Even after the task is completed, the *\$task* variable value is running. To view the actual status of the tasks running on the server, use the `Get-Task` cmdlet.

- 3 (Optional) Run the `Wait-Task` cmdlet to monitor the process progress and wait for the task to complete before running other commands.

```
Wait-Task -Task $task
```

Check Virtual Machine Baseline Status

You can check whether a virtual machine has any baselines with unknown compliance status attached to it and start a scan.

Prerequisites

Verify that you are connected to a vCenter Server system.

Procedure

- 1 Retrieve the compliance statuses with the value `Unknown` for the baselines attached to the *VM* virtual machine and store them in the *\$statuses* variable.

```
$statuses = Get-Compliance -Entity VM -ComplianceStatus Unknown
```

- 2 Check whether the virtual machine has any baselines with unknown compliance status attached to it and start a scan.

```
if ($statuses.Count -gt 0) {
  Test-Compliance -Entity VM -RunAsync"
}
```

Stage Patches

Staging allows you to download patches and extensions from the Update Manager server to the ESXi hosts without applying the patches and extensions immediately.

Note Staging can be performed only for hosts, clusters, and data centers.

Prerequisites

Verify that you are connected to a vCenter Server system.

Procedure

- 1 Retrieve a host and store it in the *\$vmHost* variable.

```
$vmHost = Get-VMHost -Name 10.23.112.233
```

- 2 Stage the patches for upgrading the host.

```
Stage-Patch -Entity $vmHost
```

Remediate a Virtual Machine

You can retrieve all baselines attached to a virtual machine and remediate the virtual machine.

Prerequisites

Verify that you are connected to a vCenter Server system.

Procedure

- 1 Retrieve all baselines attached to the *VM* virtual machine.

```
$baselines = Get-Baseline -Entity VM
```

- 2 Remediate the virtual machine.

```
Update-Entity -Entity VM -Baseline $baselines
```

Upgrade Virtual Machine Hardware

You can upgrade virtual machine hardware and VMware Tools for all virtual machines in a data center.

Prerequisites

Verify that you are connected to a vCenter Server system.

Procedure

- 1 Retrieve all virtual machines in the *Datacenter* data center.

```
$vms = Get-VM -Location Datacenter
```

- 2 Retrieve all virtual machine upgrade baselines.

```
$upgradeBaselines = Get-Baseline -TargetType VM -BaselineType Upgrade
```

- 3 Remediate all virtual machines against the virtual machine upgrade baselines.

```
foreach ($vm in $vms) {
  Update-Entity -Entity $vm -Baseline $upgradeBaselines
}
```

Remediate a Cluster

You can retrieve all baselines attached to a cluster and remediate the cluster.

Note Before remediation, you must temporarily deactivate the Distributed Power Management (DPM), High Availability (HA) admission control, and Fault Tolerance (FT) features of the clusters you want to remediate. After remediation, Update Manager automatically activates the deactivated features.

Prerequisites

Verify that you are connected to a vCenter Server system.

Procedure

- 1 Retrieve all baselines attached to the *Cluster* cluster.

```
$baselines = Get-Baseline -Entity Cluster
```

- 2 Remediate the cluster.

```
Update-Entity -Entity Cluster -Baseline $baselines
-ClusterDisableDistributedPowerManagement $true -ClusterDisableHighAvailability $true
-ClusterDisableFaultTolerance $true
```

Remediate a Host

You can retrieve all baselines attached to a host and remediate the host.

Note When remediating a host, you can configure the maintenance mode settings. You can temporarily disconnect any removable media devices that might prevent the host from entering maintenance mode as well.

Prerequisites

Verify that you are connected to a vCenter Server system.

Procedure

- 1 Retrieve all baselines attached to the *Host* host.

```
$baselines = Get-Baseline -Entity Host
```

- 2 Remediate the host.

```
Update-Entity -Entity Host -Baseline $baselines -HostFailureAction Retry -
HostNumberOfRetries 2 -HostDisableMediaDevices $true
```

Download Patches and Scan Objects

You can download patches from a previously defined location.

Prerequisites

Verify that you are connected to a vCenter Server system.

Procedure

- 1 Retrieve all entities from the *Datacenter* data center and store the result in a variable.

```
$entities = Get-Inventory -Location Datacenter
```

- 2 Download all available patches and store the result in a variable.

```
$result = Sync-Patch
```

- 3 Check whether new patches are downloaded and start scanning the entities in the *Datacenter* data center.

```
if ($result.Count > 0) {  
  Test-Compliance -Entity $entities  
}
```

Managing vSphere Lifecycle Manager with VMware PowerCLI

14

To help you get started with VMware PowerCLI, this documentation provides a set of sample scripts that illustrate basic and advanced tasks in vSphere Lifecycle Manager administration.

You can use PowerCLI to manage vSphere Lifecycle Manager on the cluster and host level.

You can use cluster-level and host-level administration starting from the following vSphere and PowerCLI versions:

Table 14-1. vSphere Lifecycle Manager Availability

	Available from vSphere version	Available from PowerCLI version
Cluster-level administration	7.0	12.1.0
Host-level administration	8.0	13.1.0

Read the following topics next:

- [Understanding vSphere Lifecycle Manager](#)
- [Creating and Managing vSphere Lifecycle Manager Clusters with PowerCLI](#)
- [Managing vSphere Lifecycle Manager Standalone Hosts with PowerCLI](#)

Understanding vSphere Lifecycle Manager

With VMware vSphere® Lifecycle Manager™, you can manage ESXi clusters and hosts by using images.

Starting with vSphere 7.0 and PowerCLI 12.1.0, you can use high-level cmdlets for operations on the cluster level as a part of the `VMware.VimAutomation.Core` module. Starting with vSphere 7.0 and PowerCLI 12.4.0, you can also use low-level cmdlets for cluster-level operations.

Starting with vSphere 8.0 and PowerCLI 13.1.0, you can use high-level and low-level cmdlets for vSphere Lifecycle Manager host-level operations.

The low-level cmdlets map directly to the syntax of the [ESX REST APIs](#) and empower you to use the full API functionality within your PowerShell console.

You can construct low-level SDK cmdlets by using the `Initialize-` and `Invoke-` cmdlet verbs in combination with the names of the [ESX REST API](#) data structures or operations.

The `ESX REST API` Reference on VMware Developer features ready-to-use vSphere Lifecycle Manager SDK code samples that you can paste directly into PowerShell. You can find a PowerCLI code snippet for each operation under *PowerCLI Client SDK Example* in the *Code Samples* section.

Lifecycle Management in vSphere

Lifecycle management refers to the process of installing software, maintaining it through updates and upgrades, and decommissioning it.

In the context of maintaining a vSphere environment, your clusters and hosts in particular, lifecycle management refers to tasks such as installing ESXi and firmware on new hosts, and updating or upgrading the ESXi version and firmware when required.

What Is vSphere Lifecycle Manager

vSphere Lifecycle Manager is a service that operates within vCenter Server, by using the built-in PostgreSQL database. No additional setup is required to use vSphere Lifecycle Manager. When you deploy the vCenter Server appliance, the vSphere Lifecycle Manager user interface is automatically activated in the HTML5-based vSphere Client. You can also access vSphere Lifecycle Manager functions through PowerCLI.

vSphere Lifecycle Manager encompasses the functionality of vSphere Update Manager from previous vSphere versions and enhances it by introducing new features and options for managing the ESXi lifecycle at a cluster and host level.

Up to vSphere 7.0, the Update Manager allowed the use of baselines and baseline groups for patching and upgrading hosts. With vSphere 7.0, vSphere Lifecycle Manager introduced the alternative approach of using vSphere Lifecycle Manager images to manage the life cycle of hosts and clusters in your environment. It also makes possible the upgrade of virtual machine hardware and VMware Tools versions.

vSphere Lifecycle Manager is compatible with an internet-connected environment, either directly or through a proxy server. It can also operate in a secure network without internet access. In such cases, you can use the Update Manager Download Service (UMDS) to download updates to the vSphere Lifecycle Manager depot, or import updates manually.

vSphere Lifecycle Manager and PowerCLI

Starting with vSphere 7.0 and PowerCLI 12.1.0, you can create and manage vSphere Lifecycle Manager clusters. For more information, see [Creating and Managing vSphere Lifecycle Manager Clusters with PowerCLI](#).

Starting with vSphere 8.0 and PowerCLI 13.1.0, you can manage standalone ESXi hosts with vSphere Lifecycle Manager images. For more information, see [Managing vSphere Lifecycle Manager Standalone Hosts with PowerCLI](#).

Creating and Managing vSphere Lifecycle Manager Clusters with PowerCLI

Starting with vSphere 7.0 and VMware PowerCLI 12.1.0, you can create and manage vSphere Lifecycle Manager clusters with PowerCLI.

Cluster-level Functionality Starting from vSphere 7.0 and VMware PowerCLI 12.1.0

Starting from vSphere 7.0 and VMware PowerCLI 12.1.0, some of the cluster-level operations that you can perform are:

- Create a vSphere Lifecycle Manager cluster with a desired state.
- View desired state elements:
 - ESXi version
 - Vendor add-ons
 - Firmware/ drivers
 - Components
- View/ check recommended images.
- Add hosts to a cluster.
- Retrieve a cluster with a vLCM desired state.
- Update a cluster's ESXi base image.
- Add, remove, replace, upgrade, or downgrade a vendor add-on on a cluster.
- Update a cluster's vSphere Lifecycle Manager component list.
- Add, remove, replace, upgrade, or downgrade a firmware add-on to a cluster.
- Modify the desired state of a vSphere Lifecycle Manager cluster.
- Convert a vSphere Update Manager cluster into a vSphere Lifecycle Manager cluster.
- Check compliance of the ESXi hosts on a cluster.
- Apply the desired state on hosts in a cluster, whose current state is different from the desired specification (remediate cluster).
- Export a desired state from a cluster.
- Import a desired state to a cluster.
- View hardware compatibility.

Cluster-level Functionality Starting from vSphere 8.0 and VMware PowerCLI 13.1.0

Starting from vSphere 8.0 and VMware PowerCLI 13.1.0, some of the new cluster-level operations that you can perform are:

- Update hardware compatibility list.
- View parallel remediation settings.
- Activate/ deactivate parallel remediation.
- Create or modify a cluster by importing an ESXi base image.
- Retrieve pre-check results.
- Stage a cluster.

Sample Scripts for vSphere Lifecycle Manager Cluster-Level Operations

The following scripts demonstrate how you can use vSphere Lifecycle Manager on the cluster level with PowerCLI.

Retrieve vSphere Lifecycle Manager (vLCM) Image Profiles

You can retrieve vLCM images from a vCenter Server system based on different filters by using the `Get-LcmImage` cmdlet. You can pass IDs or a combination of name and version filters to the `Get-LcmImage` parameters.

List all types of vLCM images

```
Get-LcmImage
```

List only `BaseImage` (ESXi) vLCM images

```
Get-LCMImage -Type 'BaseImage'
```

List only `VendorAddOn` vLCM images

```
Get-LCMImage -Type 'VendorAddOn'
```

List only `Component` vLCM images

```
Get-LCMImage -Type 'Component'
```

List only `Package` (FirmwareAddOn) vLCM images

```
Get-LCMImage -Type 'Package'
```


Create a New Cluster with a vLCM Desired State

You can create a new vSphere Lifecycle Manager cluster in the following ways:

- By specifying parameters from the VMware depots:

```
$clusterName= Read-Host -Prompt 'Provide the cluster Name'
$vLCMBaseImage = Get-LCMImage -Version '7.0 GA - 15843807' -Type BaseImage
$vendorAddon = Get-LcmImage -Name 'NEC-addon-GEN' -Version '7.0.2-02'
New-Cluster -Location Datacenter -Name $clusterName -BaseImage $vLCMBaseImage
-VendorAddOn $vendorAddon -HAEnabled -DrsEnabled
```

- By importing a vLCM desired state spec.

```
New-Cluster -Name 'MyCluster' -Location <MyDatacenter> | Import-LcmClusterDesiredState
-LocalSpecLocation F:\Image\desired-state-spec.json
```

Retrieve a Cluster's Desired State

```
Get-Cluster -Name $clusterName | Select-Object -Property Name, BaseImage,
@{n='BaseImageVersion'; e={$_.BaseImage.Version}}, Components, VendorAddon, FirmwareAddons
```

Update a Cluster's vLCM Base Image

You cannot remove or downgrade an ESXi base image - you can only upgrade it. The following example demonstrates an upgrade of a cluster's desired state to ESXi 7.0 U2.

```
$vLCMBaseImageu2 = Get-LcmImage -Version '7.0 U2a - 17867351' -Type BaseImage
Get-Cluster -Name $clusterName | Set-Cluster -BaseImage $vLCMBaseImageu2
```

Remove a vLCM Vendor Add-on from a Cluster

```
Get-Cluster -Name $clusterName | Set-Cluster -VendorAddOn $null
```

Note You can add, replace, remove, upgrade, or downgrade a vendor add-on.

Update a Cluster's vLCM Component List

Replace a Component list with a new one.

```
$components = Get-LcmImage -Id 'Component-<Component1 Name>/<Component1 Version>', 'Component-
<Component2 Name>/<Component2 Version>'
Get-Cluster -Name $clusterName | Set-Cluster -Component $components
```

Note To remove all components, use `Set-Cluster -Component $null`

Add a vLCM Firmware Add-on

```
$firmwareAddOn = Get-LcmImage -Id '<FirmwareAddOn_Id>'
Get-Cluster -Name $clusterName | Set-Cluster -FirmwareAddOn $firmwareAddOn
```

Note You can add, replace, remove, upgrade, or downgrade a firmware add-on.

Modify a Cluster's vLCM Desired State

```
$vLCMBaseImage = Get-LCMImage -Version '7.0 U3f - 20036589'
$vendorAddOn = get-lcmimage -type vendoraddon -Version '703.0.0.10.9.0-11'
Set-Cluster 'vlcm-cluster-pcli' -BaseImage $vLCMBaseImage -VendorAddOn $vendorAddOn
```

Convert a vSphere Update Manager Cluster to a vSphere Lifecycle Manager Cluster

Convert a vSphere Update Manager cluster to a vSphere Lifecycle Manager cluster.

```
$vLCMBaseImage = Get-LCMImage -Version '7.0 U3f - 20036589'
$vendorAddOn = get-lcmimage -type vendoraddon -Version '703.0.0.10.9.0-11'
Set-Cluster 'vum-cluster' -BaseImage $vLCMBaseImage -VendorAddOn $vendorAddOn
```

Check Cluster Compliance

Check the compliance of the hosts on the cluster against the cluster's desired state.

```
Get-Cluster -Name $clusterName | Test-LcmClusterCompliance
```

Remediate a Cluster

Apply the desired state to the hosts in the cluster, whose current state is different from the desired software specification.

```
Get-Cluster -Name $clusterName | Set-Cluster -Remediate -AcceptEULA
```

Export a vLCM Desired State

```
Get-Cluster -Name $clusterName | Export-LcmClusterDesiredState -Destination 'F:\Image'
-ExportOfflineBundle -ExportIsoImage
```

Import a vLCM Desired State

```
Import-LcmClusterDesiredState -Cluster 'ExistingCluster' -LocalSpecLocation F:\Image\desired-
state-spec.json -Verbose
```

View / Check Recommended Images

Generate a desired state recommendation of a vSphere Lifecycle Manager cluster.

- Based on the latest ESXi base image version.

```
Get-LcmClusterDesiredStateRecommendation 'lcm-cluster' -Latest
```

- Based on the latest patch or update of the currently installed ESXi base image version.

```
Get-LcmClusterDesiredStateRecommendation 'lcm-cluster' -Current
```

View Hardware Compatibility

```
Get-LcmHardwareCompatibility -Cluster 'lcm-cluster'
```

Update Hardware Compatibility List

```
Invoke-DownloadHclCompatibilityDataAsync
```

View Parallel Remediation Settings

```
Invoke-GetDefaultsClustersPoliciesApply
Invoke-GetClusterPoliciesApply -Cluster $clusterId
```

Activate or Deactivate Parallel Remediation

Activate or deactivate parallel remediation with the maximum number of concurrent remediations.

```
$ParallelRemediationSetting = Initialize-
SettingsDefaultsClustersPoliciesParallelRemediationAction -Enabled $true -MaxHosts
$maxHostCount
$PolicySpec = Initialize-SettingsDefaultsClustersPoliciesApplyConfiguredPolicySpec
-ParallelRemediationAction $ParallelRemediationSetting
Invoke-SetDefaultsClustersPoliciesApply -Cluster $clusterID
-SettingsClustersPoliciesApplyConfiguredPolicySpec $PolicySpec
```

Create a Cluster by Importing an Image from a Standalone Host

Extract a software specification from an ESXi host and import it into the software depot.

```
$SettingsDepotsOfflineHostCredentials = Initialize-SettingsDepotsOfflineHostCredentials
-HostName "MyHostName" -UserName "MyUserName" -Password "MyPassword" -Port 0 -SslThumbPrint
"MySslThumbPrint"
$SettingsDepotsOfflineConnectionSpec = Initialize-SettingsDepotsOfflineConnectionSpec
-AuthType "USERNAME_PASSWORD" -HostCredential $SettingsDepotsOfflineHostCredentials
Invoke-CreateFromHostDepotsOfflineAsync -SettingsDepotsOfflineConnectionSpec
$SettingsDepotsOfflineConnectionSpec
```

Then use `New-Cluster` or `Set-Cluster` to create or modify a cluster's desired state from the returned software spec.

Retrieve Pre-check Results

Return the last remediation pre-check results.

```
$cluster = Get-Cluster -Name <cluster_name>
Invoke-GetClusterSoftwareReportsLastCheckResult -Cluster $cluster.ExtensionData.MoRef.Value
```

Stage a Cluster

Stage a desired software specification on a vSphere Lifecycle Manager cluster.

```
$SettingsClustersSoftwareStageSpec = Initialize-SettingsClustersSoftwareStageSpec -Commit
<myCommit> -Hosts <MyHosts>
Invoke-StageClusterSoftwareAsync -Cluster <MyCluster> -SettingsClustersSoftwareStageSpec $spec
```

Managing vSphere Lifecycle Manager Standalone Hosts with PowerCLI

Starting with vSphere 8.0 and VMware PowerCLI 13.1.0, you can manage vSphere Lifecycle Manager standalone hosts with PowerCLI.

Host-level Functionality in PowerCLI

In PowerCLI 13.1.0 and later, you can use both high-level and low-level cmdlets that bind directly to the [ESX REST APIs](#) to manage vSphere Lifecycle Manager hosts.

Starting with VMware PowerCLI 13.1.0, some of the host-level operations you can perform are:

- Activate vSphere Lifecycle Manager on a standalone host.
- Retrieve vSphere Lifecycle Manager image profiles:
 - ESXi version
 - Vendor add-ons
 - Firmware/ drivers
 - Components
- View/ check recommended images.
- Retrieve a host with a vSphere Lifecycle Manager desired state.
- Modify the desired state of a vSphere Lifecycle Manager host.
- Convert a vSphere Update Manager host into a vSphere Lifecycle Manager host.
- Update the ESXi base image of a standalone host.
- Add, remove, replace, upgrade, or downgrade a vendor add-on on a standalone host.

- Add remove, replace, upgrade, or downgrade a vSphere Lifecycle Manager firmware add-on on a standalone host.
- Check compliance of a standalone ESXi host.
- Pre-check a host.
- Stage a host.
- Apply the desired state on a standalone host, whose current state is different from the desired specification (remediate host).

Sample Scripts for vSphere Lifecycle Manager Host-Level Operations

The following scripts demonstrate how you can use vSphere Lifecycle Manager on the host level with PowerCLI.

Activate vSphere Lifecycle Manager on a Standalone Host

```
$hostName = <host_ip_address_or_fqdn>
$vmhost = Get-Vmhost $hostName
$spec=Initialize-SettingsHostsEnablementSoftwareEnableSpec -SkipSoftwareCheck $false
Invoke-SetHostEnablementSoftwareAsync -Host $vmhost.ExtensionData.MoRef.Value
-SettingsHostsEnablementSoftwareEnableSpec $spec
```

Retrieve vSphere Lifecycle Manager (vLCM) Image Profiles

You can retrieve vLCM images from a vCenter Server system based on different filters by using the `Get-LcmImage` cmdlet. You can pass IDs or a combination of name and version filters to the `Get-LcmImage` parameters.

List all types of vLCM images

```
Get-LcmImage
```

List only BaseImage (ESXi) vLCM images

```
Get-LCMImage -Type 'BaseImage'
```

List Only VendorAddOn vLCM images

```
Get-LCMImage -Type 'VendorAddOn'
```

List only Component vLCM images

```
Get-LCMImage -Type 'Component'
```

List only Package (Firmware) vLCM images

```
Get-LCMImage -Type 'Package'
```

View / Check Recommended Images

```
$vmhost = Get-Host -Name $hostName
Invoke-GetHostSoftwareRecommendations -Host $vmhost.ExtensionData.MoRef.Value
```

Retrieve a Host's Desired State

```
Get-VMHost -Name $hostName | Select-Object -Property Name, BaseImage, @{n='BaseImageVersion';
e={$_.BaseImage.Version}}, Components, VendorAddon, FirmwareAddons
```

Modify a Host's vLCM Desired State

```
$vLCMBaseImageu2 = Get-LCImage -Version '7.0 GA - 15843807' -Type BaseImage
$vendorAddon = Get-LcmImage -Name 'NEC-addon-GEN' -Version '7.0.2-02'
$components = $components = Get-LcmImage -Id 'Component-/Component1
Version>'
$firmwareAddon = Get-LcmImage -Id '<FirmwareAddOn_Id>'
Get-Host -Name $hostName | Set-Host -BaseImage $vLCMBaseImageu2 -VendorAddOn $vendorAddon
-Component $components -FirmwareAddOn $firmwareAddon
```

Update a Host's vLCM Base Image

You cannot remove or downgrade an ESXi base image - you can only upgrade it. The following example demonstrates an upgrade of a host's desired state to ESXi 7.0 U2.

```
$vLCMBaseImageu2= Get-LcmImage -Version '7.0 U2a - 17867351' -Type BaseImage
Get-Host -Name $hostName | Set-Host -BaseImage $vLCMBaseImageu2
```

Remove a vLCM Vendor Add-on from a Host

```
Get-Host -Name $hostName | Set-Host -VendorAddOn $null
```

Note You can add, replace, remove, upgrade, or downgrade a vendor add-on.

Update a Host's vLCM Component List

Replace a Component list with a new one.

```
$components = Get-LcmImage -Id 'Component-/Component1 Version>', 'Component-
Component2 Name/Component2 Version'
Get-Host -Name $hostName | Set-Host -Component $components
```

Note To remove all components, use `Set-Cluster -Component $null`

Add a vLCM Firmware Add-on

```
$firmwareAddon = Get-LcmImage -Id '<FirmwareAddOn_Id>'
Get-Host -Name $hostName | Set-Host -FirmwareAddOn $firmwareAddon
```

Note You can add, replace, remove, upgrade, or downgrade a firmware add-on.

Check Compliance of a Host

Compare the current image on the standalone host against the desired image that you specified and define the compatibility status of the host.

```
$vmhost = Get-Host -Name $hostName
Invoke-ScanHostSoftwareAsync -Host $vmhost.ExtensionData.MoRef.Value
```

Pre-check a Host

Run a remediation pre-check on the host to ensure software and hardware compatibility with the desired software specification.

```
$spec = Initialize-SettingsHostsSoftwareCheckSpec -Commit <MyCommit>
$vmhost = Get-Host -Name $hostName
Invoke-CheckHostSoftwareAsync -Host $vmhost.ExtensionData.MoRef.Value
-SettingsHostsSoftwareCheckSpec $spec
```

Stage a Host

Stage the desired state on a standalone ESXi host.

```
$spec = Initialize-SettingsHostsSoftwareStageSpec -Commit <myCommit>
$vmhost = Get-Host -Name $hostName
Invoke-StageHostSoftwareAsync -Host $vmhost.ExtensionData.MoRef.Value
-SettingsHostsSoftwareStageSpec $spec
```

Remediate a Host

Apply the desired state on a standalone host, whose current state is different from the desired software specification.

```
$spec = Initialize-SettingsHostsSoftwareApplySpec -AcceptEula $true
$vmhost = Get-Host -Name $hostName
Invoke-ApplyHostSoftwareAsync -Host $vmhost.ExtensionData.MoRef.Value
-SettingsHostsSoftwareApplySpec $spec
```

Managing VMware Aria Operations with VMware PowerCLI

15

To help you get started with VMware PowerCLI, this documentation provides a set of sample scripts that illustrate basic and advanced tasks in VMware Aria Operations.

Read the following topics next:

- [Connect to a VMware Aria Operations Server](#)
- [Check Memory Waste Levels](#)
- [Get Remediation Recommendations](#)
- [Change Alert Ownership](#)
- [Create a Report for Problematic Hosts](#)

Connect to a VMware Aria Operations Server

To run VMware Aria Operations cmdlets, you must establish a connection to a VMware Aria Operations server and a vCenter Server system that is monitored by the VMware Aria Operations instance.

You can have more than one connection to the same server. For more information, see [Managing Default Server Connections in PowerCLI](#).

If your login credentials contain non-alphanumeric characters, you might need to escape them. For more information, see [Processing Non-alphanumeric Characters in PowerCLI](#).

Prerequisites

- If the certificate of the server you want to connect to is not trusted, verify that your PowerCLI invalid server certificate action settings are configured properly. For more information, see [Configuring PowerCLI Response to Untrusted Certificates](#).
- If you use a proxy server for the connection, verify that it is configured properly, so that the connection is kept alive long enough for tasks to finish.

Note If you do not want to use a proxy server for the connection, run `Set-PowerCLIConfiguration -ProxyPolicy NoProxy`.

Procedure

- 1 Run `Connect-OMServer` with the server name and valid credentials.

```
Connect-OMServer -Server vrops3.example.com -User 'MyAdministratorUser' -Password
'MyPassword'
```

- 2 Run `Connect-VIServer` with the server name and valid credentials.

```
Connect-VIServer -Server vc3.example.com -User 'MyAdministratorUser' -Password 'MyPassword'
```

Check Memory Waste Levels

You can check the memory waste levels of a virtual machine host for a specific period of time. For example, you can check the memory waste levels in the last month.

Prerequisites

- Verify that you are connected to a VMware Aria Operations instance.
- Verify that you are connected to the vCenter Server system that is monitored by the VMware Aria Operations instance.

Procedure

- 1 Browse the vCenter Server inventory and select a virtual machine host for which you want to check the memory waste levels.

```
$vmHost = Get-VMHost 'MyHost'
```

- 2 Get the VMware Aria Operations resource that refers to this virtual machine host.

```
$hostResource = $vmHost | Get-OMResource
```

- 3 Check the defined metrics for this VMware Aria Operations resource type.

```
Get-OMStatKey -AdapterKind $hostResource.AdapterKind -ResourceKind
$hostResource.ResourceKind
```

- 4 Get data for a specific metric.

```
$hostResource | Get-OMStat -Key "mem|waste"
```

Note This command retrieves all available metric data with the highest available granularity.

- 5 Get metric data for the last month aggregated on a daily basis.

```
$hostResource | Get-OMStat -Key "mem|waste" -From ([datetime]::Now.AddMonths(-1))
-IntervalType Days -IntervalCount 1 -RollupType Avg
```

Get Remediation Recommendations

You can get remediation recommendations for a specific resource, such as a problematic virtual machine.

Prerequisites

- Verify that you are connected to a VMware Aria Operations instance.
- Verify that you are connected to the vCenter Server system that is monitored by the VMware Aria Operations instance.
- Verify that at least one alert is triggered for the virtual machine.

Procedure

- 1 Get the virtual machine you want to check for alerts.

```
$myVm = Get-VM 'MyVM'
```

- 2 Get the associated VMware Aria Operations resource and its associated active alerts.

```
$myVmAlerts = $myVm | Get-OMResource | Get-OMAlert -Status Active
```

- 3 List the remediation recommendations for the obtained alerts.

```
$myVmAlerts | Get-OMRecommendation
```

Change Alert Ownership

You can retrieve all active alerts for a specific datastore and assign the alert ownership to your user profile.

Prerequisites

- Verify that you are connected to a VMware Aria Operations instance.
- Verify that you are connected to the vCenter Server system that is monitored by the VMware Aria Operations instance.

Procedure

- 1 Get all active alerts for the datastore.

```
$alerts = Get-Datastore 'shared' | Get-OMResource | Get-OMAlert -Status Active | where  
{ ($_.AssignedUser -eq $null) -and ($_.ControlState -eq 'Open') }
```

- 2 Assign the obtained alerts to the user profile you are currently using.

```
$alerts | Set-OMAlert -TakeOwnership
```

Create a Report for Problematic Hosts

You can create a report for virtual machine hosts that have problematic health status.

Prerequisites

- Verify that you are connected to a VMware Aria Operations instance.
- Verify that you are connected to the vCenter Server system that is monitored by the VMware Aria Operations instance.

Procedure

- 1 Get all problematic host resources in VMware Aria Operations that have red or yellow health status.

```
$hosts = Get-OMResource | where { $_.ResourceKind -eq 'HostSystem' -and $_.Health -in ('Red', 'Yellow') }
```

- 2 Get the virtual machine hosts that cause the problem.

```
$hosts | Get-VmHost
```

Managing VMware Cloud on AWS with VMware PowerCLI

16

To familiarize yourself with VMware PowerCLI, you can use a set of scripts that illustrate basic and advanced tasks in the VMware Cloud on AWS administration.

Read the following topics next:

- [Connecting to VMware Cloud on AWS](#)
- [Connect to VMware Cloud on AWS by Using an API Token](#)
- [Connect to VMware Cloud on AWS by Using an OAuth Security Context](#)
- [Connect to VMware Cloud on AWS GovCloud \(US\)](#)
- [View the Available Software-Defined Data Centers](#)
- [Connect to a vCenter Server on VMware Cloud on AWS](#)
- [Connect to a vCenter Server on VMware Cloud on AWS by Using an OAuth 2.0 Authentication](#)
- [Create a Software-Defined Data Center](#)
- [Create a Cluster in a Software-Defined Data Center](#)
- [Set the Elastic Distributed Resource Scheduler \(EDRS\) Policy of a Cluster](#)
- [Remove a Cluster from a Software-Defined Data Center](#)
- [Add Hosts to a Software-Defined Data Center](#)
- [Remove Hosts from a Software-Defined Data Center](#)

Connecting to VMware Cloud on AWS

You can use VMware Cloud on AWS to provision and manage vCenter Server virtual data centers in the cloud. VMware Cloud on AWS is a cloud service that belongs to the VMware Cloud services platform.

The VMware Cloud services platform is a web-based application that supports any number of VMware Cloud services. When you sign up for VMware Cloud services, you access your services and manage all your users, organizations, and payment methods through this platform. The VMware Cloud services platform manages common requirements for all cloud services, such as authentication, user roles, and so on.

When connecting to a VMware Cloud on AWS service, you need to authenticate with the corresponding VMware Cloud services platform.

Connect to VMware Cloud on AWS by Using an API Token

To authenticate with VMware Cloud on AWS, you can use an API token from the VMware Cloud service web portal.

Prerequisites

Use the VMware Cloud service web portal to generate an API token for authentication with PowerCLI.

Procedure

- ◆ To connect to a VMware Cloud on AWS server by using the API token, run the following command.

```
Connect-Vmc -ApiToken "Your API token"
```

Connect to VMware Cloud on AWS by Using an OAuth Security Context

To connect to VMware Cloud on AWS through an OAuth 2.0 authentication mechanism, you need an OAuth security context for the VMware Cloud services platform.

You can create an OAuth security context by using the `New-VcsOAuthSecurityContext` cmdlet.

Prerequisites

Use the VMware Cloud service web portal to generate an API token for authentication with PowerCLI.

Procedure

- 1 Create an OAuth security context for the VMware Cloud services platform by using an API token.

```
$oauthSecContext = New-VcsOAuthSecurityContext -ApiToken "Your API token"
```

- 2 Connect to VMware Cloud on AWS by using an OAuth security context.

```
Connect-Vmc -OAuthSecurityContext $oauthSecContext
```

Connect to VMware Cloud on AWS GovCloud (US)

You can use PowerCLI to connect to VMware Cloud on AWS GovCloud (US).

Prerequisites

Use the VMware Cloud service web portal to generate an API token for authentication with PowerCLI.

Procedure

- 1 Create variables to store the VMware Cloud API token and the VMware Cloud on AWS GovCloud (US) endpoints.

```
$APIToken = "Your API Token"
$VCSEndpoint = "console.cloud-us-gov.vmware.com"
$VMCEndpoint = "www.vmc-us-gov.vmware.com"
```

- 2 Create an OAuth security context and login to VMware Cloud services.

```
$oauthSecContext = New-VcsOAuthSecurityContext -VcsServer $VCSEndpoint -ApiToken $APIToken
```

- 3 Connect to VMware Cloud on AWS GovCloud (US).

```
Connect-Vmc -Server $VMCEndpoint -OAuthSecurityContext $oauthSecContext
```

View the Available Software-Defined Data Centers

With PowerCLI, you can view the available Software-Defined Data Centers (SDDCs) in your VMware Cloud on AWS organization.

Prerequisites

Verify that you are connected to VMware Cloud on AWS.

Procedure

- ◆ View the available SDDCs in your VMware Cloud on AWS organization.

```
Get-VmcSddc
```

Connect to a vCenter Server on VMware Cloud on AWS

You can use PowerCLI to connect to a vCenter Server system running on VMware Cloud on AWS.

Prerequisites

- Verify that you are connected to VMware Cloud on AWS.
- Configure the VMware Cloud on AWS networking to provide access to the vCenter Server system running on the cloud.

Procedure

- 1 Create a variable with the SDDC you want to connect to.

```
$SDDC = Get-VmcSddc 'Your SDDC'
```

- 2 To connect to the vCenter Server system, provide the vCenter host name and credentials.

```
Connect-VIServer $sddc.VCenterHostName -Credential $sddc.VCenterCredentials
```

Connect to a vCenter Server on VMware Cloud on AWS by Using an OAuth 2.0 Authentication

If the vCenter Server system has an OpenID authentication enabled, you can authenticate with the vCenter Server system by using an OAuth security context for the VMware Cloud services platform.

Prerequisites

- Verify that you are connected to VMware Cloud on AWS.
- Configure the VMware Cloud on AWS networking to provide access to the vCenter Server system running on the cloud.

Procedure

- 1 Create an OAuth security context for the VMware Cloud service by using an API token.

```
$oauthSecContext = New-VcsOAuthSecurityContext -ApiToken "Your API token"
```

- 2 Create a variable with the SDDC you want to connect to.

```
$SDDC = Get-VmcSddc 'Your SDDC'
```

- 3 Exchange the OAuth security context for an SAML security context.

```
$samlSecContext = New-VISamlSecurityContext -VCenterServer $sddc.VCenterHostName  
-OAuthSecurityContext $oauthSecContext
```

- 4 Connect to a vCenter Server system by using an SAML security context.

```
Connect-VIServer -Server $sddc.VCenterHostName -SamlSecurityContext $samlSecContext
```

Create a Software-Defined Data Center

With PowerCLI, you can create a Software-Defined Data Center (SDDC) in your VMware Cloud on AWS organization.

Prerequisites

Verify that you are connected to VMware Cloud on AWS.

Procedure

- 1 Get a VMware Cloud on AWS account.

```
$awsAccount = Get-AwsAccount
```

- 2 Get the VMware Cloud on AWS Virtual Private Cloud (VPC) subnets.

```
$awsVpcSubnets = Get-AwsVpcSubnet -AwsAccount $awsAccount -Region "US_EAST_1"
```

- 3 Create an SDDC.

```
New-VmcSddc -Name "MySDDC" -HostCount 3 -Region "US_EAST_1" -AwsAccount $awsAccount  
-AwsVpcSubnet $awsVpcSubnets[0]
```

Create a Cluster in a Software-Defined Data Center

With PowerCLI, you can create a cluster in a Software-Defined Data Center (SDDC) in your VMware Cloud on AWS organization.

Prerequisites

Verify that you are connected to VMware Cloud on AWS.

Procedure

- 1 Retrieve the SDDC in which you want to create a cluster.

```
$sddc = Get-VmcSddc 'Your SDDC'
```

- 2 Create the cluster.

```
New-VmcSddcCluster -Sddc $sddc -HostCount 3
```

Set the Elastic Distributed Resource Scheduler (EDRS) Policy of a Cluster

With PowerCLI, you can set the Elastic Distributed Resource Scheduler (EDRS) policy of clusters.

Prerequisites

Verify that you are connected to VMware Cloud on AWS.

Procedure

- 1 Create a variable with the SDDCs of the clusters you want to manage.

```
$sddc = Get-VmcSddc 'Your SDDC'
```

- 2 Create a variable with the clusters you want to manage.

```
$cluster = Get-VmcSddcCluster -Name 'Your Cluster' -Sddc $sddc
```

- 3 Set the EDRS policy.

```
Set-VmcClusterEdrsPolicy -Cluster $cluster -PolicyType cost -MinHostCount 6 -MaxHostCount 15
```

Remove a Cluster from a Software-Defined Data Center

With PowerCLI, you can remove clusters from Software-Defined Data Centers (SDDCs) in your VMware Cloud on AWS organization.

Prerequisites

Verify that you are connected to VMware Cloud on AWS.

Procedure

- 1 Create a variable with the SDDCs from which you want to remove clusters.

```
$sddc = Get-VmcSddc 'Your SDDC'
```

- 2 Create a variable with the clusters you want to remove.

```
$cluster = Get-VmcSddcCluster -Name 'Your Cluster' -Sddc $sddc
```

- 3 Remove the clusters.

```
Remove-VmcSddcCluster -VmcSddcCluster $cluster
```

Add Hosts to a Software-Defined Data Center

With PowerCLI, you can add hosts to a Software-Defined Data Center (SDDC) in your VMware Cloud on AWS organization.

Add hosts to an SDDC with PowerCLI. You can use the `Cluster` parameter to indicate in which cluster you want to add the hosts. If you do not specify a cluster, the hosts are added to the default cluster of the SDDC.

Prerequisites

Verify that you are connected to VMware Cloud on AWS.

Procedure

- 1 Create a variable with the SDDCs in which you want to add hosts.

```
$sddc = Get-VmcSddc 'Your SDDC'
```

- 2 Add the hosts to the default cluster of the specified SDDCs.

```
Add-VmcSddcHost -Sddc $sddc -HostCount 2
```

Remove Hosts from a Software-Defined Data Center

With PowerCLI, you can remove hosts from a Software-Defined Data Center (SDDC) in your VMware Cloud on AWS organization.

Remove hosts from an SDDC with PowerCLI. You can use the `Cluster` parameter to indicate from which cluster you want to remove hosts. If you do not specify a cluster, the hosts are removed from the default cluster of the SDDC.

Prerequisites

Verify that you are connected to VMware Cloud on AWS.

Procedure

- 1 Create a variable with the SDDCs from which you want to remove hosts.

```
$sddc = Get-VmcSddc 'Your SDDC'
```

- 2 Remove hosts from the default cluster of the specified SDDCs.

```
Remove-VmcSddcHost -Sddc $sddc -HostCount 2
```

Help and Support for VMware PowerCLI

17

If you cannot resolve a problem with PowerCLI on your own, you can file a support request under the conditions of your VMware contract.

You can use the following options to troubleshoot problems with PowerCLI.

Self-Help

As a first step, use the existing documentation to search for a solution to your problem. You can search the following resources.

- [The PowerCLI official documentation.](#)
- [The PowerCLI community.](#)
- [The PowerCLI blog.](#)

File a Support Request Under a Basic or Production Support Contract

You can file a technical support request under the terms of a Basic or Production Support contract with VMware. You can get support for the following types of problems.

- Cmdlets are not working as documented.
- Installation problems with PowerCLI.

Note PowerCLI is not available for selection as a product in the technical support request form. You must select the product that you are having issues with, for example, vCenter Server, VMware ESXi, and so on.

File a Support Request Under an SDK and API Support Contract

The SDK and API Support package increases the scope of the professional assistance you can request. In addition to the incidents covered under Basic and Production support, you can request the following assistance.

- Clarification of the scope of tasks for which you can use PowerCLI.

- Guidance on how to use PowerCLI to accomplish tasks.
- Assistance with custom scripts.

Note The SDK and API support engineers do not write or test your code, but they might suggest sample code for a possible solution.

Read the following topics next:

- [Generate a PowerCLI Support Bundle](#)

Generate a PowerCLI Support Bundle

Generate a PowerCLI support bundle and send it to VMware support to help them resolve your problem.

If you encounter a problem in PowerCLI that you cannot resolve, you can open a support request for VMware.

Generate and download a PowerCLI support bundle in the form of a ZIP file and attach it to your VMware support request.

Procedure

- 1 Create a variable with the script that produced the error.

```
$errorcode={
$vm = Get-VMvCLS*Start-VM$vm
}
```

- 2 Generate and download the support bundle by running the `Get-ErrorReport` cmdlet.

```
Get-ErrorReport -ProblemScript$errorcode -Destination.\Downloads
-ProblemDescription"VM would not power on"
```

Optional: To limit the size of your support bundle, include the `MaxDataDepth` parameter which specifies object data depth for the report. For example, to limit object data depth to one level, add `-MaxDataDepth1` to the above example.

Results

The ZIP file that contains environmental information about the error is saved at the specified destination.

What to do next

Attach or upload the ZIP file and submit your VMware support request.