



Windows PowerShell 使用说明

Windows PowerShell 使用说明

Windows PowerShell 是一种命令行外壳程序和脚本环境，使命令行用户和脚本编写者可以利用 .NET Framework 的强大功能。本期 TT 虚拟化技术手册介绍 Windows PowerShell 的使用方法，还会涉及到 VMware PowerCLI 的相关信息。

PowerShell 与 Hyper-V

SCVMM 和 Hyper-V PowerShell cmdlets 帮助虚拟机环境的管理员开启了新的管理模式。PowerShell 不仅能对 Hyper-V 进行监控与测试，还能克服 Hyper-V 热迁移限制，您需要掌握哪些命令呢？

- ❖ Hyper-V PowerShell 中的常用命令简介
- ❖ 使用 PowerShell 进行 Hyper-V 监控与测试
- ❖ 使用 PowerShell 克服 Hyper-V 热迁移限制

PowerShell 与 PowerCLI

拿 VMware 的 PowerCLI 与微软的 PowerShell 作比较似乎也太可能，因为前者需要后者先安装好，且它仅是一个针对核心 PowerShell 环境的供应商指定附加物。两者结合能发挥很大作用。

- ❖ VMware PowerCLI 和微软 PowerShell 谁好？
- ❖ 使用 PowerShell 与 PowerCLI 自动化主机服务器任务
- ❖ 在 vSphere PowerCLI PowerShell 界面使用 host profiles
- ❖ 跳出框外 巧解 PowerShell 与 PowerCLI 难题

PowerShell 管理技巧

Windows 7 远程桌面管理? Citrix XenDesktop 5 管理? PowerShell 都能掌控。

- ❖ PowerShell 管理 Windows 7 远程桌面的技巧
- ❖ 使用 Windows PowerShell 管理 Citrix XenDesktop 5
- ❖ 使用 Windows PowerShell 管理远程桌面设备

Hyper-V PowerShell 中的常用命令简介

对于虚拟机、宿主机和任务进程而言，微软 Hyper-V 管理控制台和 SCVMM (System Center Virtual Machine Manager) 都是很不错的管理工具。但是和大多数的图形界面接口 (GUI) 工具一样，都无法避免在客户定制化和权限上受到限制。但是 PowerShell 的使用可以很好的弥补 Hyper-V 在管理方面的缺陷。

SCVMM 和 Hyper-V PowerShell cmdlets 帮助虚拟机环境的管理员开启了新的管理模式。在本文中，TechTarget 中国的特约专家 Rob McShinsky 讲述一些 PowerShell cmdlets 和命令。即使您没有购买 SCVMM，依然可以使用 PowerShell 来完成大多数的任务，包括：获得基本的系统信息、创建新的虚拟机和完成在线迁移过程等。首先，下载并安装 Hyper-V PowerShell 模块。（或许某些时候您可能需要更改 PowerShell 的执行模式，以保证该模块可以正确运行。在 Windows Server 2008 中它可以正常工作，但是在 Windows 7 中，您可能需要对执行模式做更改以减轻其受到的限制。这并不是最佳的办法，但是请注意有些时候我们必须减轻执行模式的限制。）

在完成了 Hyper-V PowerShell 模块的安装后，接下来就可以尝试一些基本的功能。在你对这些 PowerShell cmdlets 和命令足够熟悉之后，还可以组合一些命令编写成更加复杂和实用的脚本。但是这个过程并不像“Hello World”那么简单，在我们学习跑步之前先来学习走路。

Get-VM。这个 cmdlets 对于很多脚本程序而言，都是最基本的常用命令之一，而且从字面上也很容易理解。Get-VM 检索虚拟机相关的信息，然后快速呈现出某台特定主机上的虚拟机状态。我经常这样使用该命令：

```
Get-VM --Server
```

图 1



Host	VMElementName	State	Up-Time (mS)	Owner
	test16	Running	724322080	
	test14	Running	724337150	
	CentOS_5_4	Running	40968155	
	robtest	Stopped	0	
	Test5	Running	724276402	
	test17	Running	724291753	
	test18	Running	724439800	

请注意，您必须通过-Server 辅助命令来锁定某一台特殊的服务器操作，这点在使用 Hyper-V 管理控制台时也是一样的。

New-VM, Set-VMemory, Set-VMCPUCount, Add-NewVirtualHardDisk. 这一组 Hyper-V PowerShell cmdlets 用于创建新虚拟机。虽然在 Hyper-V Manager GUI 中通过配置向导来完成更加简单一些，但是在需要同时创建多个虚拟机的时候，命令行的方式会更加好用。

图 2

```

PS C:\> New-VM -Server Host1 -Name RobTest2 -Path d:

Host
-----
Host1

UMElementName
-----
RobTest2

State
-----
Stopped

Up-Time (mS)
-----
0

Owner
-----

PS C:\> Set-VMemory -Server Host1 -VM RobTest2 1024

UMElementName
-----
RobTest2

VirtualQuantity
-----
1024

Limit
-----
1024

Reservation
-----
1024

PS C:\> Set-VMCPUCount -Server Host1 -VM RobTest2 4

UMElementName
-----
RobTest2

Quantity
-----
4

Limit
-----
1000000

Reservation
-----
0

Weight
-----
100

Cores/Socket
-----
4

SocketCount
-----
1

PS C:\> Add-NewVirtualHardDisk -Server Host1 -VM RobTest2 -Size 24GB

Mode
-----
darhs

LastWriteTime
-----
12/31/1600 7:00 PM

Length
-----

Name
-----
RobTest2.VHD

UMElementName
-----
RobTest2

Element Name
-----
Hard Drive

Resource Sub type
-----
Microsoft Synthetic Disk Drive

UMElementName
-----
RobTest2

Element Name
-----
Hard Disk Image

Resource Sub type
-----
Microsoft Virtual Hard Disk

PS C:\>

```

通过这些命令做简单调整，还可以增加或修改多台虚拟机。

```

New-VM -Server Host1 -Name RobTest2,RobTest3,RobTest4 -Path d:
Set-VMemory -Server Host1 -VM RobTest2,RobTest3,RobTest4 1024
Set-VMCPUCount -Server Host1 -VM RobTest2,RobTest3,RobTest4 4

```

```
Add-NewVMHardDisk -Server Host1 -VM RobTest2,RobTest3,RobTest4 -Size 24GB
```

现在您可以看到，在同时创建多个虚拟机时这样可以节省时间。

Start-VM, Stop-VM, Save-VM, Shutdown-VM. 用于完成虚拟机状态调整的命令也非常的好用。有一些操作无法通过 GUI 来实现的，我经常使用这些命令来关闭虚拟机或是保存虚拟机状态以用于故障诊断。和同时创建和调整多个虚拟机的命令相似，通过这些命令可以同时改变多台虚拟机的运行状态。

```
Start-VM -Server Host1 -VM RobTest2,RobTest3,RobTest4
Stop-VM -Server Host1 -VM RobTest2,RobTest3,RobTest4
Save-VM -Server Host1 -VM RobTest2,RobTest3,RobTest4
Shutdown-VM -Server Host1 -VM RobTest2,RobTest3,RobTest4
```

注： 如果要使用 Shutdown-VM 命令，需要安装 Hyper-V 集成组件。

Move-VM. PowerShell 还可以用于帮助完成 Hyper-V 集群的某些工作。我曾经在 Failover Cluster Administrator 无法正确工作的情况下，使用如下的命令迁移虚拟机以用于故障的诊断。当然，使用 cluster.exe 命令也可以在节点间迁移虚拟机，但是我更喜欢使用 PowerShell 命令。

```
Move-VM -Server Host1 -VM RobTest2 -Destination Host2 --force
```

当然，多台虚拟机的迁移也是可以实现的，但是需要连续地操作。

```
Move-VM -Server Host1 -VM RobTest2,RobTest3,RobTest4 -Destination Host2 --force
```

或者借助通配符，您也可以把 Host1 上的所有以 RobTest 开头的虚拟机迁移到 Host2 上。

```
Move-VM -Server Host1 -VM RobTest% -Destination Host2 --force
```

（注：为了避免出现提示信息：“WARNING: Cluster commands not loaded. Import-Module FailoverClusters and try again”，运行“Run, Import-Module FailoverCluster”命令完成 Failover Cluster 模块的装载，并启用 PowerShell 的集群属性。）

在您对这些 PowerShell cmdlets 和命令逐渐熟悉以后，可以尝试浏览一下 PowerShell Management Library for Hyper-V 手册。虽然手册的内容本身让人昏昏欲睡，但是我们可以一边阅读一边实践一些新的 Hyper-V PowerShell 命令，之后您可能会发现它提供了一种通用的语法模式。那么，您就可以编写脚本来借助 PowerShell 完成一些更加复杂的辅助性管理工作，相信您会慢慢发现这样做要比完全使用 Hyper-V 向导来完成虚拟机管理任务快地多。

(来源: TechTarget 中国)

使用 PowerShell 进行 Hyper-V 监控与测试

通过 PowerShell 可以实现 VMware 中的很多管理功能，因此 Hyper-V 也可以通过类似的方式去管理。然而微软在通过 PowerShell 命令（或称为 cmdlet）管理 Hyper-V 方面做的工作并不多。

如果您的版本是 Windows Server 2008 R2，借助文件库可以和 VMware 一样通过免费的 PowerShell cmdlet 来监控和管理 Hyper-V。该文本库俗称“PowerShell Module for Hyper-V”，其中包括了用于完成虚拟机状态监控、检索硬件配置和获取其它虚拟机相关数据的 cmdlet。通过这些 Hyper-V 便捷管理方式，您可以检查虚拟机是否健康、内存容量甚至是功能状态。

安装和测试用于 Hyper-V 管理的 PowerShell

PowerShell 模块需要简单几步安装过程，你需要主机安装 2.0 版的 .NET Framework 和启用 Windows PowerShell。请注意安装程序需要调用 PowerShell 脚本，因此您只有在系统中启用 PowerShell 执行策略为无限制模式（微软通常不推荐这么做）才可以完成安装。通过输入 Set-ExecutionPolicy Unrestricted cmdlet 可以更改策略。如果您不希望做更改，可以手动完成各个部分的注册。

完成 PowerShell 模块安装后下一步是检查您的用户是否具备在服务器上的管理员权限，否则部分 Hyper-V 管理用 cmdlet 可能无法工作。输入 Test-Admin cmdlet 可以检查权限。图 1 中所示 Windows 通过 true 或 false 给出是否具备管理权限的回答。

图 1



在您安装完 PowerShell 模块后，检查账户是否具备管理员权限。

监控虚拟机状态

PowerShell 可以监控在 Hyper-V 服务器上的多个虚拟机的运行状态。通过如下 cmdlet 获取虚拟机参数：

Get-VM.Windows 返回某台服务器上的每台虚拟机的名称、基本状态和正常运行时间等信息。

Get-VMSummary。输入该命令，并附加您希望检查的机器名称，可以获得某个指定虚拟机的更多细节。Windows 会返回很多，包括虚拟机使用的虚拟 CPU 数量、CPU 加载历史记录、操作系统、创建时间以及被系统认证过的域名全称等信息。（参考图 2）

图 2

```
Administrator: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
PS C:\> Get-UM

Host                                UMElementName                    State      Up-Time (mS) Owner
-----
HYPER-U                            Lab-E2K10                         Running    1488560
HYPER-U                            ESA                               Stopped    0
HYPER-U                            Tasmania                         Stopped    0
HYPER-U                            Lab2-DC                          Running    1484519
HYPER-U                            Storage                          Stopped    0
HYPER-U                            Server Core                      Stopped    0
HYPER-U                            32                               Stopped    0
HYPER-U                            SharePoint                      Stopped    0
HYPER-U                            DFS2                             Stopped    0
HYPER-U                            DFS1                             Stopped    0
HYPER-U                            Lab-E2K7                        Stopped    0
HYPER-U                            Lab2-E2K7                       Stopped    0
HYPER-U                            DHCP                             Stopped    0
HYPER-U                            Server 2008 R2                  Stopped    0
HYPER-U                            Lab-DC                           Running    1491766

PS C:\> Get-UMSummary Lab-DC

CPUCount           : 1
Snapshots          :
GuestOS            : Windows Server 2008 R2 Enterprise
Uptime             : 1501796
UptimeFormatted    : if (<$_.uptime -gt 0> <<[datetime]0>.addmilliseconds(<$_UpTime>.tostr
m:ss">)} else {0}
CPULoadHistory     : {0, 0, 0, 0...}
Host               : HYPER-U
IpAddress          : www180.sedoparking.com
Jobs               :
UMElementName      : Lab-DC
Notes              : #CLUSTER-INVARIANT#:160529e6-346b-4d08-a8e1-5f456dcceb3a}
Heartbeat          : OK
FQDN               : Lab-DC.lab.com
CPULoad            : 0
CreationTime       : 20091207154633.707200-000
EnabledState       : Running
Name               : F92B74E1-A569-4DC4-AEF5-201EE1090E1A
MemoryUsage        : 1024

PS C:\>
```

通过命令行可以获取虚拟机的健康信息。

测试虚拟机功能

您还可以了解虚拟机是否按照指定的方式工作。使用 Test-VMHeartbeat cmdlet 来检查虚拟机是否响应。Windows 返回 OK 表示指定的虚拟机心跳信息正常。（参考图 3）

图 3



```
Administrator: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
PS C:\> Test-UMHeartbeat Lab-DC

elementName
-----
Lab-DC

PS C:\>
```

测试虚拟机心跳信息检查硬件配置

很多 cmdlet 可以返回虚拟机的硬件资源分配情况。如下的硬件专用 cmdlet 可以帮助改善 Hyper-V 的管理。

Get-VMSettingData.Windows 返回每个虚拟机硬件配置的详细信息。如图 4，我指定了用列表输出的方式。另外，还可以以概要方式输出。如果您没有指定虚拟机，cmdlet 会提供所有该服务器上的虚拟机硬件配置。为了补充您的 Hyper-V 管理策略，可以把该信息输出为 CSV 文件并在 Excel 中编辑。

图 4

```
Administrator: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
PS C:\> Get-UMSettingData Lab-DC | FL

UMElementName      : Lab-DC
GENUS               : 2
CLASS               : Msvm_VirtualSystemSettingData
SUPERCLASS          : CIM_VirtualSystemSettingData
DYNASTY              : CIM_ManagedElement
RELPATH             : Msvm_VirtualSystemSettingData.InstanceID="Microsoft:F92B74E1-A569-4DC4-AEF5-201EE1090E1A"
PROPERTY_COUNT      : 21
DERIVATION           : <CIM_VirtualSystemSettingData, CIM_SettingData, CIM_ManagedElement>
SERVER              : HYPER-U
NAMESPACE           : root\virtualization
PATH                : \\HYPER-U\root\virtualization:Msvm_VirtualSystemSettingData.InstanceID="Microsoft:F92B74E1-A569-4DC4-AEF5-201EE1090E1A"
AutoActivate        :
BaseBoardSerialNumber : 1942-2052-8254-9404-7839-4307-14
BIOSGUID            : <93985F3A-7D27-4990-89E1-932A925D77A0>
BIOSNumLock         : False
BIOSSerialNumber    : 1942-2052-8254-9404-7839-4307-14
BootOrder           : <1, 2, 3, 0>
Caption             : Virtual Machine Settings
ChassisAssetTag      : 6577-1801-7689-2506-7686-9012-54
ChassisSerialNumber  : 1942-2052-8254-9404-7839-4307-14
CreationTime         : 20091207154633.707200-000
Description          : Active settings for the virtual machine.
ElementName         : Lab-DC
InstanceID          : Microsoft:F92B74E1-A569-4DC4-AEF5-201EE1090E1A
Notes                : #CLUSTER-INVARIANT#: <160529e6-346b-4d08-a8e1-5f456dcceb3a>
NumaNodeList         : <>
NumaNodesAreRequired : False
OtherVirtualSystemType :
Parent              :
SettingType          : 3
SystemName           : F92B74E1-A569-4DC4-AEF5-201EE1090E1A
VirtualSystemType    : 301

PS C:\>
```

您可以获得虚拟机的硬件配置信息

Get-VMMemory. 这个 cmdlet 提供了为每台虚拟机保留的内存数量（参考图 5）。这对于资源规划来说是非常方便的，您在 Hyper-V 主机中添加虚拟机前，必须确切知道空闲的内存数量。

图 5


```
Administrator: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
PS C:\> Get-UMMemory
```

UMElementName	VirtualQuantity	Limit	Reservation
Lab-E2K10	2048	2048	2048
ESB	1024	1024	1024
Tazmania	2048	2048	2048
Lab2-DC	2048	2048	2048
Storage	2048	2048	2048
Server Core	2048	2048	2048
32	1024	1024	1024
SharePoint	2048	2048	2048
DFS2	2048	2048	2048
DFS1	2048	2048	2048
Lab-E2K7	2048	2048	2048
Lab2-E2K7	2048	2048	2048
DHCP	2048	2048	2048
Server 2008 R2	2048	2048	2048
Lab-DC	1024	1024	1024

```
PS C:\>
```

查看每台虚拟机分配的内存空间。

Get-VMCPUCount 使用该 cmdlet 可以识别为每台虚拟机分配的 CPU 核心数量。

Get-VMProcessor 准确的讲这个 cmdlet 返回的是性能状态，图 6 中显示了正在工作中的虚拟机 CPU 占用率。

图 6

```
Administrator: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
PS C:\> Get-UMCPUCount
```

UMElementName	Quantity	Limit	Reservation	Weight	Cores/Socket	Socket
Lab-E2K10	1	1000000	0	100	4	1
ESA	1	1000000	0	100	4	1
Tazmania	1	1000000	0	100	4	1
Lab2-DC	1	1000000	0	100	4	1
Storage	1	1000000	0	100	4	1
Server Core	1	1000000	0	100	4	1
32	1	1000000	0	100	4	1
SharePoint	1	1000000	0	100	4	1
DFS2	1	1000000	0	100	4	1
DFS1	1	1000000	0	100	4	1
Lab-E2K7	1	1000000	0	100	4	1
Lab2-E2K7	1	1000000	0	100	4	1
DHCP	1	1000000	0	100	4	1
Server 2008 R2	1	1000000	0	100	4	1
Lab-DC	1	1000000	0	100	4	1

```
PS C:\> Get-UMProcessor
```

UM Name	ID	Family	Speed MHz	Load %
Lab-E2K10	0	6	2400	0
Lab2-DC	0	6	2400	0
Lab-DC	0	6	2400	0

```
PS C:\>
```

Windows 窗口中提供了 CPU 的配置和性能情况

结论

请了解，PowerShell 模块是成熟的 Hyper-V 管理解决方案。如果您需要的功能比如上列举的要多，我建议自行下载和测试该模块。库体中包含的 PowerShell cmdlet 可以帮助提高您的系统管理能力。

(来源: TechTarget 中国)

使用 PowerShell 克服 Hyper-V 热迁移限制

微软在 Windows Server 2008R2 系统里改进了 [Hyper-V 的热迁移](#)，这也拉近了和对手 (Vmware) 产品的差距，但是它仍然有许多的限制。不过你可以使用 PowerShell 命令行工具来热迁移，从而突破这些限制。

管理员利用 Hyper-V 的热迁移，可以几乎不间断地把一个虚拟机从 Hyper-V 群集节点转移到另一个节点。当然，还会有少许的中断系统时间，各种各样的虚拟机热迁移都不能避免，这是热迁移的通病，而且各种虚拟化产品都是以虚拟机的迁移中断时间来作标准。新的 Hyper-V 3.0 改进了 Hyper-V 热迁移的一些限制，现在通过 [PowerShell](#) 命令行工具的确可以看到这些改进。

使用 PowerShell 故障转移命令行工具

大多数的 Hyper-V 群集都用 System Center Virtual Machine Manager (简称 [SCVMM](#)) 来管理，不过在只有一个或者极少量的 Hyper-V 服务器的群集的时候，PowerShell 命令行工具可以避免使用 SCVMM 热迁移时带来的额外中断时间，让热迁移的过程自动化和策略化。多数 Windows 群集管理员都熟练使用 cluster.exe 去管理群集资源，不过 PowerShell 群集故障转移命令行工具还可以用来进行系统的热迁移。

在同一群集中单独迁移虚拟机到另一个节点

下面的 PowerShell 脚本可以让你灵活地迁移虚拟机而不受群集的约束。你所要做的，是确认好群集的命名，群集虚拟机资源，还有你即将要迁移虚拟机到哪一个目标节点。你也可以修改这个脚本用于迁移多个虚拟机，这可以让你自动完成热迁移时节点的准备。

有一个要点，如果你要迁移一个 [Windows 7](#) 系统的虚拟机，你需要安装远程服务管理工具 (Remote Server Administration Tools-RSAT) 和确认启动了群集的故障转移。一般情况下，启用群集的时候服务器就会自动启懂 RSAT。

下面的 PowerShell 脚本可以对群集里的虚拟机热迁移：

```
# -----  
-----  
# Migrate Single Virtual Machine With Failover Cluster CMDLet
```

```
# -----  
-----  
# Necessary to enable failover cluster functions  
Import-Module FailoverClusters  
$CL = Read-Host "Enter Cluster Alias Name"  
$VM = Read-Host "Enter Full Cluster Name Resource Name of VM to Migrate"  
$DH = Read-Host "Enter Destination Host Name"  
get-cluster "$CL" | Move-ClusterVirtualMachineRole -name "SCVMM $VM  
Resources" -node "$DH"
```

确认你要迁移的虚拟机是启动状态(否则你将会收到一个错误信息), 下面是使用脚本的步骤:

1. 复制上面的脚本, 保存为。ps1 的脚本文件(如:VM.ps1)

2. 打开 PowerShell。(开始菜单-程序)

3. 运行之前保存的脚本。(如 VM.ps1)

4. 填入处于同一群集中的各项应答必填项, 群集名称提示, 虚拟机群集资源名称和目标节点。



5. 图 1 填好群集属性等提示。

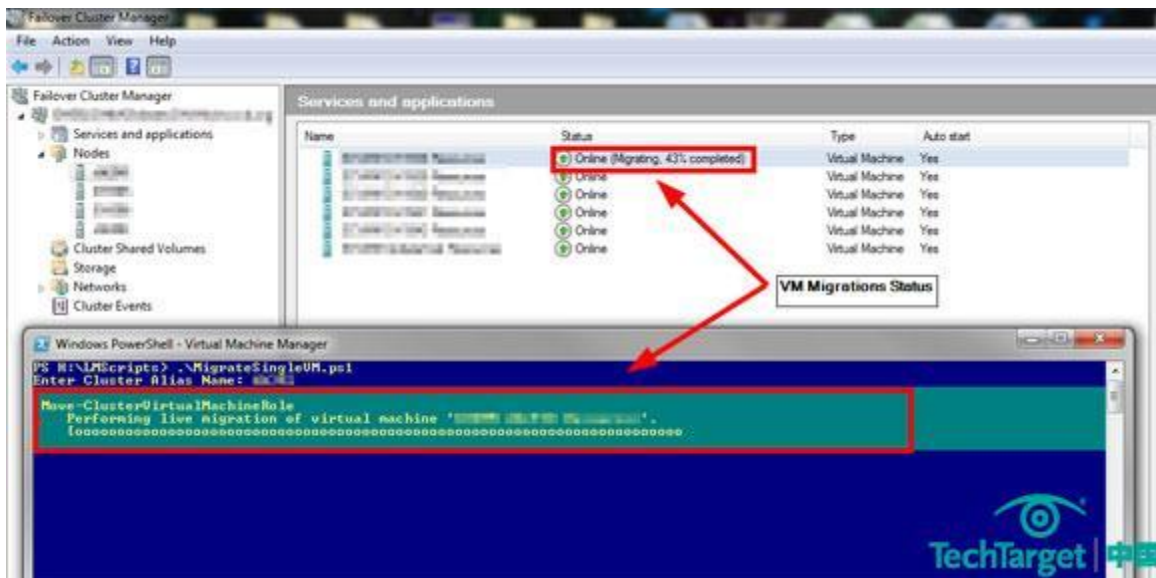


图 2 从群集故障转移管理或者命令状态查看热迁移进程。

在同一个群集里从同一个节点迁移所有的虚拟机到另一个节点

为了把维护工作放在单独一个服务器上，你可能要做一个热迁移，把所有的虚拟机从一个服务器移动到同个群集里的另一个服务器。你可以使用 PowerShell 命令行代替 Hyper-V 管理器快速地执行虚拟机迁移，或者做一个小小的修改以适应维护事件或者对节点的失败迁移作出快速的处理。相对于 Hyper-V 一次只能迁移一个服务器，PowerShell 命令行简直神了。

这里是同时热迁移多个虚拟机的脚本：

```
# -----
# Migrate All Virtual Machines on One Node to Another with Failover
Cluster CMDLet
# -----

# Necessary to enable failover cluster functions
Import-Module FailoverClusters
$CL = Read-Host "Enter Cluster Alias Name"
$SH = Read-Host "Enter Source Host Name"
$DH = Read-Host "Enter Destination Host Name"
```

这脚本的执行步骤和迁移单个虚拟机一样(见上面单个虚拟机脚本运用的说明)。把脚本保存起来，运行之，根据提示填入相应的属性。

现在你应该会觉得 PowerShell 命令行工具非常有用了吧，稍作修改就能完成你用 Hyper-V 热迁移做不到的事情。到 VirtuallyAware.com 分享你的热迁移心得吧。

(来源: TechTarget 中国)

VMware PowerCLI 和微软 PowerShell 谁好？

拿 VMware 的 PowerCLI 与微软的 PowerShell 作比较似乎是太可能，因为前者需要后者先安装好，且它仅是一个针对核心 PowerShell 环境的供应商指定附加物。

某人仅仅基于其脚本功能而去选择一个虚拟化平台也是不可能的。因此，我这里的比较并不是为了设置一个让两家公司竞争的恶意比赛，我将告诉你各个方案中我喜欢和不喜欢的地方。希望我可以解释两家公司提高其脚本产品的方法。让我们微软开始吧。

微软的 PowerShell

关于 SCVMM，我喜欢虚拟化管理的一个方面是其内置的观察脚本窗口。这就像一个脚本记录器，它赋予你如同 PowerShell 脚本一样的抓取管理行为的能力。当你正在 SCVMM 内执行一个把一个虚拟机转移到另一个 Hyper-V 主机的转移任务时，这是这点的很好举例。在这个向导结束时，将出现一个“查看脚本”按钮，显示一个脚本需要的确切 PowerShell 命令。

从某种程度上讲，这完全是一个来自于 SCVMM 研发方式的福音。最初，SCVMM 是由 PowerShell 管理的，后来一个图形管理壳的研发围绕它开展起来，因此 Windows 管理员并不需要了解它背后的 PowerShell。

对于微软来说，揭示在后台运行的 PowerShell 并不难。虽然这不利于管理员使用每个循环、错误检查以及日志的变量来编写 PowerShell 脚本，但是这对接触 PowerShell 的新手们来说是非常有用的，至少可以让他们克服在学习如何通过基本的 PowerShell 命令集来执行普通的管理任务时的困难阶段。

目前，Hyper-V 还没有一个官方的命令集，但是 James O'Neill 创建了一个集合并存储在 Codeplex 网站。O'Neill 采用了应用程序接口并编写了基于此的命令集。随着 SCVMM 2012 的发布，命令集的数量将会显著增加，人们在 SCVMM 中可以做的任务量通常也会大幅增加。

VMware 的 PowerCLI

VMware 已经大步前进地在 PowerShell 中增加了许多新功能，它被称为 PowerCLI。在连续的发布中，公司已经显著增加了产品中命令集的数量，差不多有

250 个命令集。之前的补充缺乏控制 ESX 主机 iSCSI 堆栈的命令集，虽然最近的 vSphere 4.1 版本已经填补了这一缺口。

但仍然到处存在奇怪的漏洞。例如，没有任何有意义的有关 VMware 分布式虚拟网络切换器的命令集。看来公司赞同将其“Host Profiles”功能作为新创建的 ESX 主机的主要配置工具。购买 Enterprise 和 SKU 的客户有权使用分布式虚拟网络切换器和 Host Profiles。

未来版本中会有新的部署方法，它们强调会在部署新的 ESX 主机时在 PowerCLI 上使用 host profiles。值得一提的是，虚拟化专家 Luc Dekens 已经研发出考虑到 DvSwitches 管理的功能。如此说来，PowerCLI 在 VMware 社区的主要用途是报告和检查 vSphere 环境。诸如 virtu-al.com 的“健康检查”脚本也已经非常受欢迎。

VMware 的软件开发包就是一切。这个极其丰富的接口让访问人几乎可以执行每一个想要的行动，无论是在主机上还是 vCenter 管理服务器上。这与微软唯一官方支持的命令集只针对 SCVMM 形成了对比。VMware 和微软之间存在一些共同之处：都对其管理程序所谓的“免费”版本提供极其有限的 PowerShell 支持。对于客户可能得到一个免费的管理程序，并用命令行工具管理它而不是为这个管理块支付溢价这一点，两家公司都持谨慎态度。

vCenter 自身没有将 PowerCLI 直接集成到 vSphere 客户端。想要寻找一些将管理“记录”成 PowerCLI 代码的方法，你可以查看 VMware Onyx，它是一个放置在 vSphere 客户端和 vCenter 之间的免费设备，它将行为以原始的 PowerCLI SDK 代码形式输出。附带说一下，它也把行为以适合 VMware 的 Orchestrator 的 Java 脚本格式输出。

Onyx 的输出可能会有点让人不知所措，但请不要被吓到。通过一个图形用户界面和在任务完成之后收集的 PowerCLI 代码，我们用于执行管理任务的方式更为友好。人们应该利用 PowerGUI 的前端以及可扩展其功能的各种 PowerPacks。PowerGUI 对包括活动目录在内的种类繁多的管理任务有效，但是从虚拟化的角度来看，为 VMware 的 vSphere、Citrix 的 XenServer、Microsoft 的 HyperV、虚拟磁盘分析、HP 的虚拟连接以及 Quest 的 vWorkspace 准备了 PowerPacks。

与此同时，VMware 正在试图将 PowerCLI 支持扩展到其他技术，这也是微软已经准备要在今后做的事情。这里有一个 PowerCLI 作为 VMware 的虚拟桌面解决方案，称为 View，但是这个实施方案与 vSphere PowerCLI 的实施大不相同。感觉起来它更像一个 DOS 命令行系统而不是 PowerShell，并且它允许流水线技术以及查

询对象的属性和特性。截至目前，虽然像虚拟机站点恢复管理这样的技术不具有 PowerCLI，但是它很有可能出现在虚拟机的雷达屏幕上。

对于那些没有生活在 Windows 世界的人来讲，PowerShell 和 PowerCLI 已经在系统管理世界引起了巨大冲击波。但是基于 Linux 的脚本编写者也许会觉得遭受了冷落。VMware 提供了 Perl 语言，而微软没有，但这是指日可待的。

相似的老版 CLI 和远程 CLI 似乎也正在消亡，就像 VMware 到 vCLI 和 vCLI 设备的“ESX”命令行端口一样。现在，它们想要一个中间步骤，想让 VMware 客户对老版“服务控制台”环境的依赖。现在，似乎 PowerShell 和 PowerCLI 为可预见的未来规定系统管理员的休息场所。

(来源: TechTarget 中国)

使用 PowerShell 与 PowerCLI 自动化主机服务器任务

这是关于 PowerCLI 的另一篇技巧，你还记得 PowerCLI 吗？它是为 Windows PowerShell 脚本语言而创建的嵌入单元。这个嵌入单元能让你快速轻松地将 vSphere 环境的所有方面进行自动化。今天的文章介绍如何使用 PowerShell 和 PowerCLI 对主机服务器进行自动化操作。

VMware 说法中的主机服务器是个提供虚拟化服务的系统。也就是所说的 hypervisor，当然这种情况下就是 VMware 的产品 ESX Server。PowerCLI 能管理 ESX Server 的各种版本，从 3.0 开始，包括“瘦”版本 ESXi。以前你可以使用 PowerCLI 管理 VMware Server（非裸金属 hypervisor），但那个功能处于某种原因消失了。

那么我们来看看使用 PowerCLI 能对主机服务器做什么。写这篇文章的时候，装载的是 PowerCLI is 4.0 Update 1。这个版本是对先前版本的重大更新，因为它所包含的 cmdlets 清单比之前的多。（PowerShell cmdlets 类似于如 cmd.exe 或 bash 等 command-shells 中的内部命令。）

有个关于 PowerShell 的笑话，它只列出了 PowerCLI cmdlets，在名称中含有“host”。

(Note that "PS >" indicates the shell prompt, and the text that immediately follows is what I typed. Everything else is output from a previously typed command.)

```
PS > get-command *host* -pssnapin vmware* | format-wide
Add-VmHostNtpServerApply-VMHostProfile                                Export-
VMHostProfileGet-VMHost                                              Get-
VMHostAccountGet-VMHostAdvancedConfiguration                       Get-
VMHostAvailableTimeZoneGet-VMHostDiagnosticPartition               Get-
VMHostFirewallDefaultPolicyGet-VMHostFirewallException
Get-VMHostFirmwareGet-VMHostHba                                     Get-
VMHostModuleGet-VMHostNetwork                                       Get-
VMHostNetworkAdapterGet-VMHostNtpServer                            Get-
VMHostProfileGet-VMHostService                                      Get-
VMHostSnmpGet-VMHostStartPolicy                                     Get-
VMHostStorageGet-VMHostSysLogServer                                Import-
VMHostProfileInstall-VMHostPatch                                    Move-
```

VMHostNew-VMHostAccount	New-	
VMHostNetworkAdapterNew-VMHostProfile		Remove-
VMHostRemove-VMHostAccount	Remove-	
VMHostNetworkAdapterRemove-VMHostNtpServer		Remove-
VMHostProfileRestart-VMHost	Restart-	
VMHostServiceSet-VMHost	Set-	
VMHostAccountSet-VMHostAdvancedConfiguration	Set-	
VMHostDiagnosticPartitionSet-VMHostFirewallDefaultPolicy		
Set-VMHostFirewallExceptionSet-VMHostFirmware		
Set-VMHostHbaSet-VMHostModule	Set-	
VMHostNetworkSet-VMHostNetworkAdapter	Set-	
VMHostProfileSet-VMHostService	Set-	
VMHostSnmpSet-VMHostStartPolicy	Set-	
VMHostStorageSet-VMHostSysLogServer	Start-	
VMHostStart-VMHostService	Stop-VMHostStop-	
VMHostService	Suspend-VMHostTest-	
VMHostProfileCompliance	Test-VMHostSnmp	

那是 60 个 cmdlets。不过我不能对每个 cmdlet 的细节加以描述，但我们能重点介绍一些 cmdlets。同样，不要忘记 PowerShell 里的每个 cmdlet 都有内置帮助。需要的话输入“get-help”加上 cmdlet 名称，或者 cmdlet 名称加上“-?”。

首先我想介绍的是 Get-VMHost。与所有 get-cmdlets 一样，着重在于从容器检索对象。如果使用 Connect-VIServer cmdlet 连接到 vCenter 服务器，然后自己运行 Get-VMHost，那么结果可能如下：

```
PS > Connect-VIServer vlab.halr9000.comName
Port                               User-----
----vlab.halr9000.com             443
halPS >
Get-VMHostName                     State      PowerState    Id CpuUsage
CpuTotal  Memory  Memory
Mhz       Mhz  UsageMB TotalMB-----
-----atlesx01.hal... Connected
PoweredOn ...-232    168    8000    2295    4094atlesx02.hal...
Connected PoweredOn ...-225    366    8000    2645
4094atlesx03.hal... Connected PoweredOn ...t-10    350    8000
3264    4091
```


如果你没有 vCenter 备份，不用担心，可以用 PowerCLI 所有的信息。事实上，PowerCLI 在没有 vCenter 的情况下能一次管理多台 ESX 服务器。

Connect-VIServer 命令似乎不同，但最终结果是一样的。（没有 vMotion 或模板，你在使用某些功能时仍然需要 vCenter。）

现在来分析上面命令的输出结果，然后我们将用一些其他方式使用这个 cmdlet。这里执行了两个命令。第一个是建立到 vCenter 服务器的连接。第二个是不通过参数自己运行 Get-VMHost。这样，你可以看见所有连接到 vCenter 的主机服务器清单（ESX or ESXi）。这个输出表分为几列，包括主机名称、连接状态、电源状态。从 informational ID 查看，有四个参数能让你快速诊断系统健康状况。

如同 PowerShell 里的其他对象，我们不能立即看到所有东西。所以，我们来看看其他一些有用的信息：

```
PS > get-vmhost | format-table name, manufacturer, model, numcpu,
version, build -autosize
```

Name	Manufacturer	Model	Num Cpu
Version Build			
-----	-----	-----	-----
atlesx01.halr9000.com	Dell Inc.	PowerEdge	
SC1435 4 4.0.0 208167			
atlesx02.halr9000.com	Dell Inc.	PowerEdge	
SC1435 4 4.0.0 219382			
atlesx03.halr9000.com	Dell Inc.	PowerEdge	
SC1435 4 4.0.0 219382			

因此问题在于：你如何知道对象里的哪个领域可用？首先我会说在 PowerShell 中，每样事物都是个对象。对象包含成员，成员能包含属性和方式。属性相当于上面表格中输出信息栏所见的域名。方法我会在另一篇文章中介绍，但总之，它们能定义对对象所作的事情，就是行动。

回到属性。有种方法能列出所有属性。注意，使用 Format-Wide cmdlet 出来的信息横跨两个输出栏。


```
PS > get-vmhost | get-member -MemberType property | format-wide
```

Build	ConnectionState
CpuTotalMhz	CpuUsageMhz
CustomFields	HyperthreadingActive
Id	Manufacturer
MemoryTotalMB	MemoryUsageMB
Model	Name
NumCpu	ParentId
PowerState	ProcessorType
State	TimeZone
Version	VMSwapfileDatastoreId
VMSwapfilePolicy	

你可使用 Get-VMHost cmdlet 上的 Name 属性，只对那些匹配的服务器进行输出，你可以使用通配符。由于 PowerShell 管道工作的方式，你能将 VMHost 对象作为输入传送到 Get-VM cmdlet。最终结果将会显示宿主在某台主机服务器上的虚拟机列表。例如：

```
PS > get-vmhost -name atlesx01* | get-vm
```

Name	PowerState	Num CPUs	Memory (MB)

ELGFIL01	PoweredOn	1	384
VMGVIC01	PoweredOn	1	512
VMGADC01	PoweredOn	1	384
VMGADC02	PoweredOn	1	384
ELGADC01	PoweredOn	1	512

在文章最后，我将介绍如何使用主机服务器防火墙。使用 Get-VMHostFirewallException cmdlet 列出防火墙允许的活动。如果输入没有参数的 cmdlet，会反馈错误信息，因为你必须制定查询的具体主机服务器。下面是步骤：

```
PS > $vmhost = get-vmhost atlesx01*
PS > Get-VMHostFirewallException -Name 'FTP Server' -VMHost $vmhost
```

Name	Enabled
IncomingPorts	OutgoingPorts
Protocols	ServiceRunning

FTP Server	False	21	TCP
------------	-------	----	-----

在这个例子中，我也将通过指定防火墙规则名称过滤输出。

一旦你明白如何进行防火墙配置，使用 `Set-VMHostFirewallException` cmdlet 进行更改就很是容易的事。这个 cmdlet 有两个参数：

- **Exception:** 这个相当于需要更改的防火墙规则。参数能在管道上具体说明。
- **Enabled:** 这是个 Boolean 旗帜，如果真实，将启用规则（因为已经防火墙允许）。设置不正确将禁用规则并关闭任何响应的端口。

下图是启用 FTP 服务器防火墙规则的例子：

```
PS > $ftp = Get-VMHostFirewallException -Name 'FTP Server' -VMHost $vmhost
PS > $ftp | Set-VMHostFirewallException -Enabled:$true
```

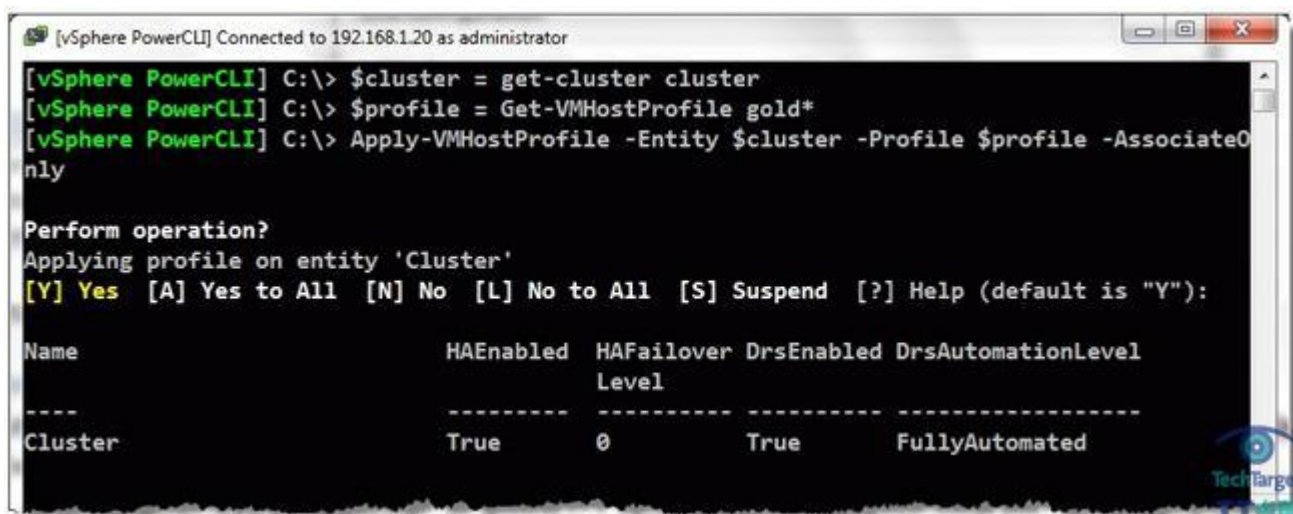
Name	Enabled		
IncomingPorts	OutgoingPorts	Protocols	ServiceRunning
FTP Server	True	21	TCP

在 vSphere PowerCLI PowerShell 界面使用 host profiles

在本文的上半部分中，我们介绍了 [VMware host profiles 的概念以及使用 PowerCLI 创建一个全新的 host profiles 的方法](#)。下面我们将挂载主机并检查 host profiles 的一致性。

把某个主机或集群系统挂载到 host profiles

如我之前所讲，把主机挂载到 host profiles 并且应用这些更改的方法非常简单。只需在 PowerCLI 中运行一个简单的 cmdlets: Apply-VMHostProfile。在我们开始举例之前，先来简单解释一些这个 cmdlets 中各个参数所代表的含义。



```
[vSphere PowerCLI] Connected to 192.168.1.20 as administrator
[vSphere PowerCLI] C:\> $cluster = get-cluster cluster
[vSphere PowerCLI] C:\> $profile = Get-VMHostProfile gold*
[vSphere PowerCLI] C:\> Apply-VMHostProfile -Entity $cluster -Profile $profile -AssociateOnly

Perform operation?
Applying profile on entity 'Cluster'
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is "Y"):
```

Name	HAEnabled	HAFailover Level	DrsEnabled	DrsAutomationLevel
Cluster	True	0	True	FullyAutomated

名称	描述
Entity	将要涉及的主机或集群目录。
Profile	将要被采用的 host profiles 名称或者是目标地
Variable	接受哈希算法，可以附加变量条件。例如把 vSwitch 的 IP 地址作为条件。
AssociateOnly	通过该命令把 profile 跟某个实体挂载，但是不应用更改。
ApplyOnly	通过该命令把 profile 更改应用到某个实体而不执行挂载。
RunAsync	通过该命令，允许在 vCenter 在后台启用某些特殊操作的时候，脚本程序可以迅速释放对资源的控制权。
Server	选择性地建立到一台或多台 vCenter 服务器的连接。如果您没有指定服务器，该操作将会应用到所有活动的 vCenter 服务器。

注：在使用 Apply-VMHostProfile 时还有很多需要特别注意的地方。请一定要注意参考 PowerCLI 中的 help 相关内容来获取更多信息。

让我们继续实现 profile 跟集群的连接：

如果连接成功了，您将会重新获得对目标实体的控制权。而在现有的 Get-VMHostProfile cmdlet 版本下，您无法轻易地看到哪些实体已经被挂载，但是接下来我将会向您展示如何借助 Get-View 和 Get-VIObjectByVIView cmdlets 来获得这些详细信息。

```

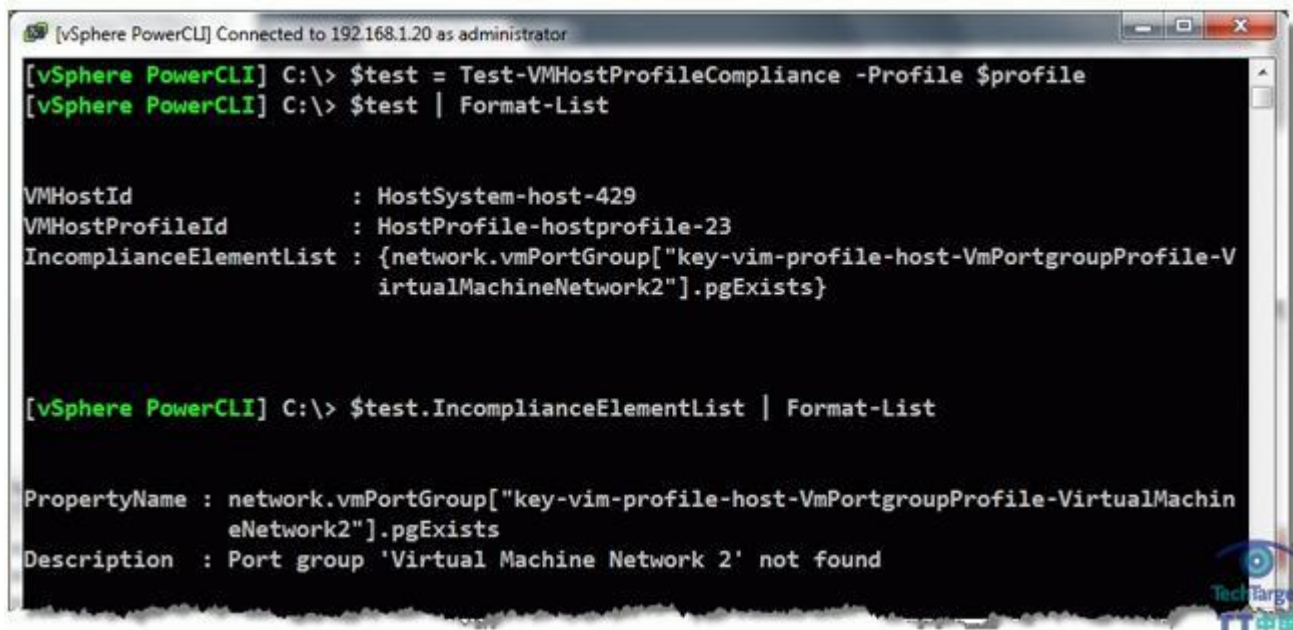
[vSphere PowerCLI] Connected to 192.168.1.20 as administrator
[vSphere PowerCLI] C:\> $hpview = $profile | Get-View
[vSphere PowerCLI] C:\> $hpview.Entity | Get-VIObjectByVIView | Select-Object Name, Id

Name                                Id
----                                --
Cluster                             ClusterComputeResource-domain-c278
192.168.1.11                         HostSystem-host-427
192.168.1.12                         HostSystem-host-429
  
```

检查 host profiles 的一致性

接下来，让我们一起来检查 \$profile 变量指代的所有 host profiles 的一致性情况。这个操作是通过 Test-VMHostProfileCompliance cmdlet 实现的。

Test cmdlet 可以返回很多的相关信息，因此我强烈建议您通过变量来指定所需的结果。下面是如何进行操作的演示：



```
[vSphere PowerCLI] C:\> $test = Test-VMHostProfileCompliance -Profile $profile
[vSphere PowerCLI] C:\> $test | Format-List

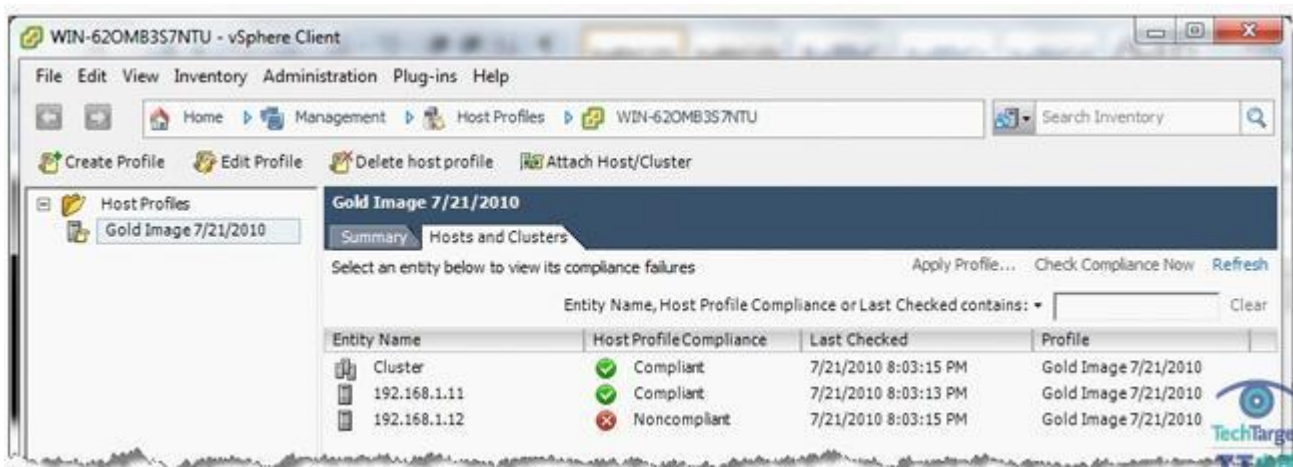
VMHostId           : HostSystem-host-429
VMHostProfileId     : HostProfile-hostprofile-23
IncomplianceElementList : {network.vmPortGroup["key-vim-profile-host-VmPortgroupProfile-V
                          irtualMachineNetwork2"].pgExists}

[vSphere PowerCLI] C:\> $test.IncomplianceElementList | Format-List

PropertyName : network.vmPortGroup["key-vim-profile-host-VmPortgroupProfile-VirtualMachineNetwork2"].pgExists
Description  : Port group 'Virtual Machine Network 2' not found
```

在第二行中，我在 Format-List cmdlet 中加入了 \$test 变量。这么做的目的是因为在默认的结果列表中，很多相关的信息被隐藏了。而很多有价值的信息都是来自于对 \$test 变量 IncomplianceElementList 属性的核对过程。在截图的最下方，您可以看到很简单地英文提示说明了兼容性检查失败的原因。对于实现 vSphere 环境中配置变化的检查而言，这是一个非常实用的工具。

请参考如下我在使用 vSphere Client 时看到的内容：



应用 host profiles 设置

接下来，让我们开始对 ESX 主机做一致性检查。如上所提到的，主机必须首先被设置为维护模式。您可以通过 PowerCLI 中的 Set-VMHost 以及状态参数 cmdlets 实现这个操作。在开始前，您还需要做好把某台主机从集群中断开的准备，正如我们在使用 vSphere client 时所做的那样。

```
[vSphere PowerCLI] C:\> Get-VMHost 192.168.1.12 | Set-VMHost -State maintenance
```

Name	State	PowerState	Id	CpuUsage Mhz	CpuTotal Mhz	Memory UsageMB	Memory TotalMB
192.168.1.12	Mainten...	PoweredOn	...-429	65	2802	760	2044

```
[vSphere PowerCLI] C:\> Apply-VMHostProfile -Profile $profile -Entity 192.168.1.12
```

Perform operation?
Applying profile on entity '192.168.1.12'
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is "Y"):

Name	State	PowerState	Id	CpuUsage Mhz	CpuTotal Mhz	Memory UsageMB	Memory TotalMB
192.168.1.12	Mainten...	PoweredOn	...-429	26	2802	736	2044

在第二行命令中，我应用了 profile。换句话说，vSphere 创建了丢失的 portgroup。接下来让我们重新检查一次一致性。

```
[vSphere PowerCLI] C:\> Test-VMHostProfileCompliance -Profile $profile
```

如果此时您在 vSphere Client 中点击刷新，就可以看到所有的项目都已经处于绿色状态。

看过本文之后，您应该已经对于如何配合 host profiles 使用 PowerCLI 有了一定的了解。现在让我们抛开这些理论的东西，开始对系统的一致性检查吧！

跳出框外 巧解 PowerShell 与 PowerCLI 难题

在这篇文章中，我想着重讲解一下在我浏览 PowerCLI 论坛时，看到的一些很有用的技巧。下面我将向你介绍这些技巧，会谈到 PowerCLI 是什么，以及我发现的一些特点。我还将提供涉及到的技术的一些背景资料。

PowerShell 以外的思路

首先介绍的一个帖子是，“[在 ESX 中，需要借助 PowerShell 脚本来找到某个虚拟机 \(VM\) 的进程 ID \(PID\)](#)。”此贴的前提是，作者需要找到一些信息，但他唯一知道获取到这些信息的途径就是登录到 ESX 操作系统控制台 (COS)。这就是为什么当前需求与他需要进程 ID 不符的原因了，但这足以证明，有时你为了完成某些诊断任务而必须登录到控制台。

这个帖子的特点在于，它似乎暴露了 vSphere 应用程序编程接口 (API) 上的某些不足，并需要以 PowerCLI 为辅助。就如所有的 APIs 一样，功能都是创建了一个抽象层，可以对某一产品或某一产品套件进行尽可能灵活的一致性编程，而且可以引入新的功能来降低软件开发任务的难度。问题是，与 ESX 虚拟化层进行沟通工作的操作系统控制台 (COS) 实际上也是一个操作系统，而操作系统往往都相当复杂，并且在某些情况下，用户接口也非常分散。

回到 PowerCLI 和 vSphere API 上来。架构在 VMware 最底层架构之上的 vSphere API 层是非常有层次架构，并提供了最外层的功能。操作系统却有着不一样的架构！很不幸的是，vSphere API 并不能构建一张完美的流程图以使你可以登录到操作系统控制台 (COS)。然而，PowerCLI 也是基于某些 API 之上，这意味着你不能在操作系统 (OS) 控制台之内使用 PowerCLI 来执行相关任务——至少不是直接。这正是发布到论坛上的解决方案的用武之地。

对我们和原始帖子的作者来说幸运的事情是，PowerCLI 是建立在 PowerShell 内，并且 PowerShell 可以调用一个外部进程作为脚本的一部分。该解决方案是在 LucD 的回贴中提到的。在这种情况下，外部进程是 plink.exe，它是一个非常流行的安全命令行 shell (SSH) 客户端—PuTTY 中的一部分。该脚本实现使用 plink 登录到 ESX 主机管理控制台，然后执行远程控制命令“vm-support”。此实用工具是一种收集信息的方法，正如原帖中提到的。

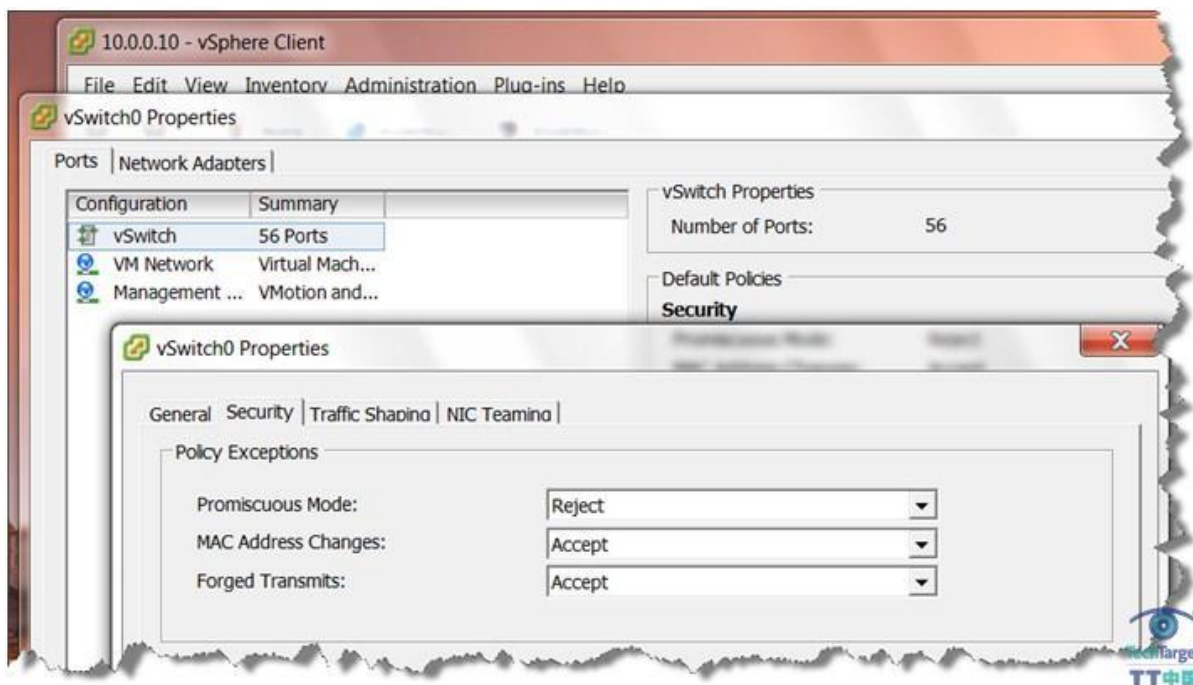
你可能会说这种行为有点像一个黑客，因为你不是通过“被认可的接口”来访问数据，但有些时候，你不得不去找到一个方法。这个特别帖子的寓意是：不要害

怕，要跳出思想的框框。虽然 vSphere API 是一个相当大的框框，但它的目的不是为了解决你可能遇到的每一个 vSphere 系统的问题。事实上，对于它准确的理解是，它确实不适合用于诊断。

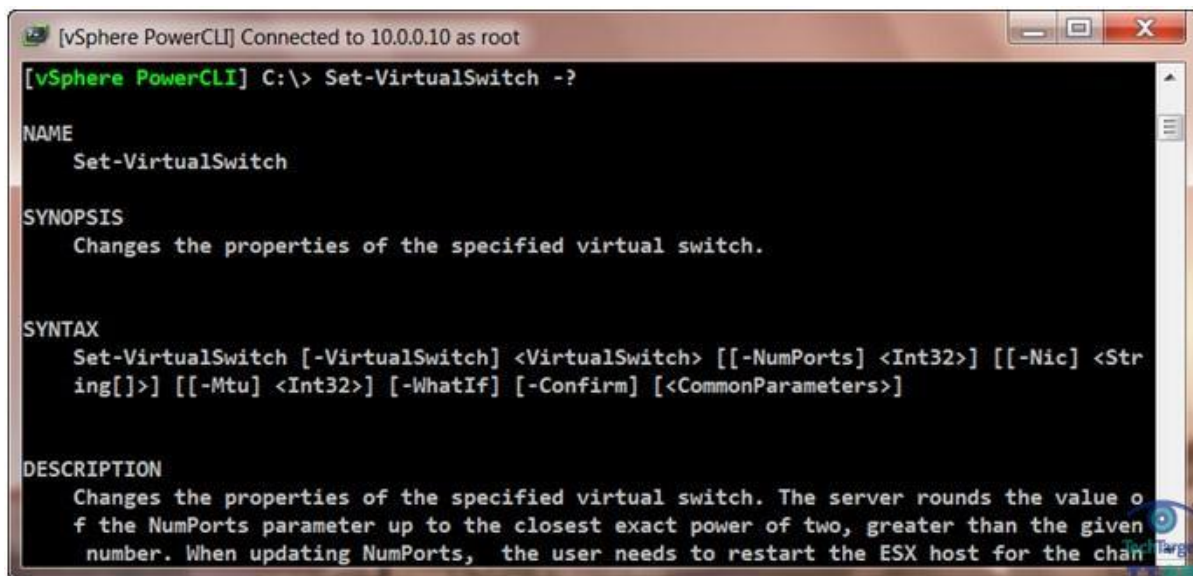
当 PowerCLI 的内置命令不起作用

这第二个帖子标题是，“[使用脚本来配置 vSwitch 上的安全参数](#)。”虽然在这个帖子中的脚本非常高深，但是我认为这将会是一个很好的例子，因为我认为它描述了一种常见的场景，当你没有使用 PowerCLI 内置命令而使用到 vSphere API 的功能。其基本前提是内置命令（在以前的文章我也谈到过）与 API 的功能没有 100% 的对等。然而，通过使用 Get-View 内置命令，再加上其他一些技巧，你就可以完成很多的需要做的事情。

此贴的作者想要说的是关于设置 vSwitch 安全方面的设置。我打开安全选项卡中的一个选项页面，当你打开一台虚拟交换机的性能属性时就可以看到，如下图：



不幸的是，你不能通过内置命令设置这些属性，下图为 Set-VirtualSwitch 命令的帮助文档：



```
[vSphere PowerCLI] Connected to 10.0.0.10 as root
[vSphere PowerCLI] C:\> Set-VirtualSwitch -?

NAME
    Set-VirtualSwitch

SYNOPSIS
    Changes the properties of the specified virtual switch.

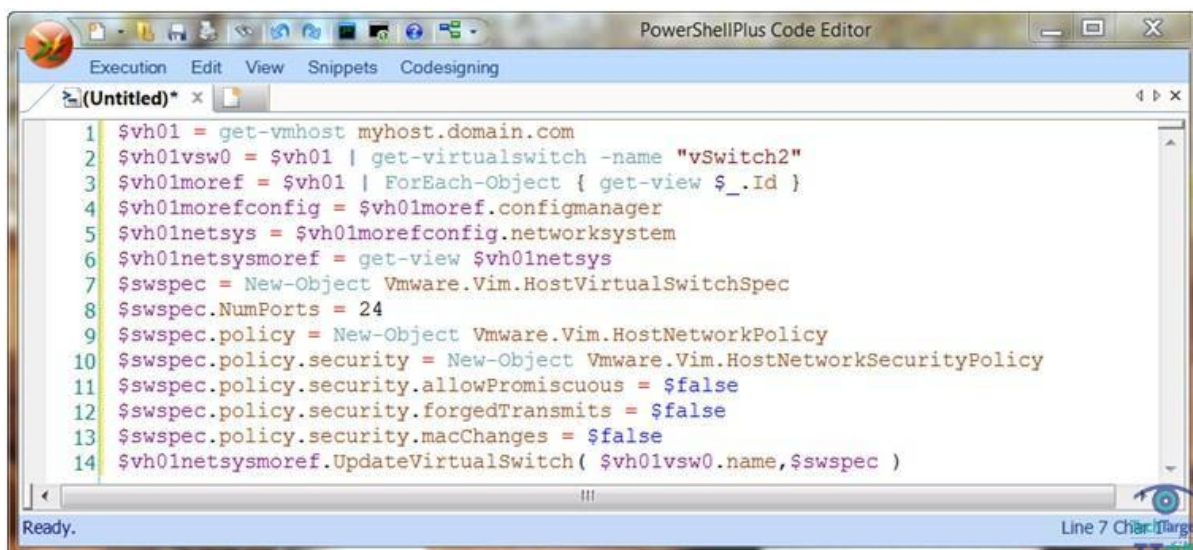
SYNTAX
    Set-VirtualSwitch [-VirtualSwitch] <VirtualSwitch> [[-NumPorts] <Int32>] [[-Nic] <String[]>] [[-Mtu] <Int32>] [-WhatIf] [-Confirm] [<CommonParameters>]

DESCRIPTION
    Changes the properties of the specified virtual switch. The server rounds the value of the NumPorts parameter up to the closest exact power of two, greater than the given number. When updating NumPorts, the user needs to restart the ESX host for the change to take effect.
```

该内置命令缺乏安全设置参数。也有可能，VMware 正在并将继续缩短内置命令和 API 之间的差距，但今天它确实是帮不了我们了。

为了弥补暂时的不足，你必须使用 Get-View 的内置命令。Get-View 是非常有用的——它就像释放出一种技能以让你闯过难关。我不打算详细介绍有关的内置命令，但你需要知道的是，目前 PowerCLI 内置命令给虚拟化管理员开发了一个可以与 PowerShell 协调工作的可定制接口。当它起作用时，这对于管理员来说是好消息，但是当你的操作受到了内置命令的制约，那么你必须采用底层 API 对象。换句话说，你需要理解一个完全不同的接口——这与程序员为 vSphere 开发软件时看到的一样。

回到帖子，帖子中有段代码，如下所示（为便于阅读，已稍微编辑）：



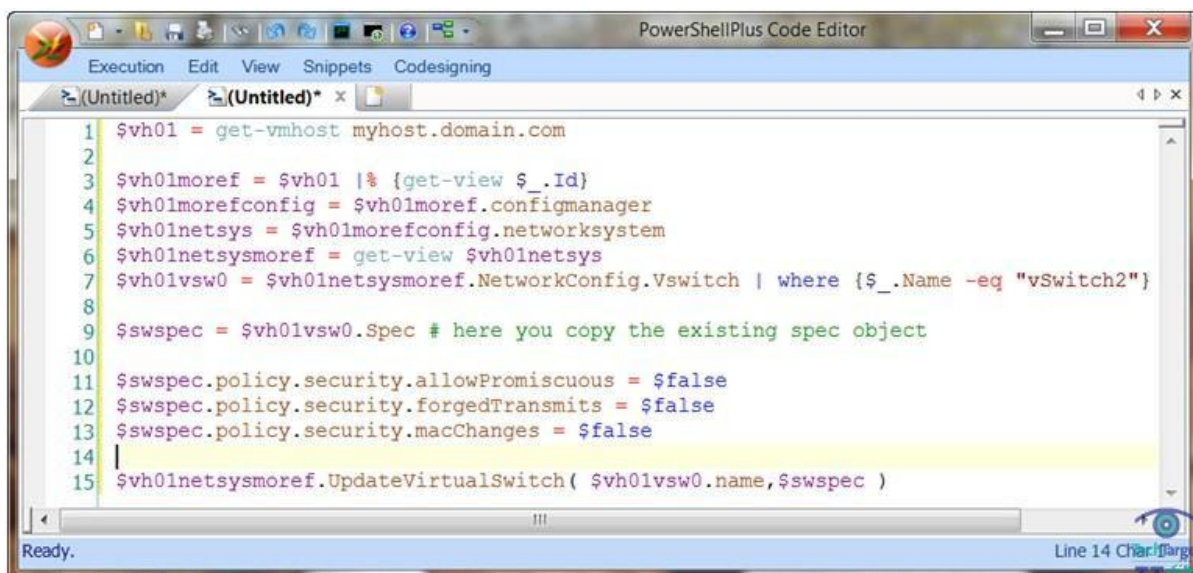
```
1 $vh01 = get-vmhost myhost.domain.com
2 $vh01vsw0 = $vh01 | get-virtualswitch -name "vSwitch2"
3 $vh01moref = $vh01 | ForEach-Object { get-view $_.Id }
4 $vh01morefconfig = $vh01moref.configmanager
5 $vh01netsys = $vh01morefconfig.networksystem
6 $vh01netsysmoref = get-view $vh01netsys
7 $swwspec = New-Object VMware.Vim.HostVirtualSwitchSpec
8 $swwspec.NumPorts = 24
9 $swwspec.policy = New-Object VMware.Vim.HostNetworkPolicy
10 $swwspec.policy.security = New-Object VMware.Vim.HostNetworkSecurityPolicy
11 $swwspec.policy.security.allowPromiscuous = $false
12 $swwspec.policy.security.forgedTransmits = $false
13 $swwspec.policy.security.macChanges = $false
14 $vh01netsysmoref.UpdateVirtualSwitch( $vh01vsw0.name, $swwspec )
```

第 1 行检索了主机列表并将它们分配给一个变量。第 2 行遍历该变量中包含的每个主机服务器，并找到名为“vSwitch2”的交换机。第 3 行在这里你可以看到 Get-View 内置命令的使用以获取原始对象。

4-6 行，你会得到一个以一定方式命名的 vSphere API，这些将为更新虚拟交换机做好准备——“艰难的方式”。

在第 7 行会报错。为了达到编辑虚拟交换机的目的，作者给出了非常有效的假设，所以你将使用到一个新的对象（这些通常是 vSphere API 使用来描述一个正在创建或修改的对象）。

但问题是，就像 LucD 在回复中做出的非常有用的解释，该 API 有时是相当挑剔。希望看到这个规范的对象是，它其中的每一个参数字段都被明确定义并赋予了默认值。在这一点上，你可以找出这些默认值应该是什么，也可以简单地复制一个已有的，之后再修改成你的内容。更新后的代码如下所示：



```
1 $vh01 = get-vmhost myhost.domain.com
2
3 $vh01moref = $vh01 |% {get-view $_.Id}
4 $vh01morefconfig = $vh01moref.configmanager
5 $vh01netsys = $vh01morefconfig.networksystem
6 $vh01netsysmoref = get-view $vh01netsys
7 $vh01vsw0 = $vh01netsysmoref.NetworkConfig.Vswitch | where {$_.Name -eq "vSwitch2"}
8
9 $swspec = $vh01vsw0.Spec # here you copy the existing spec object
10
11 $swspec.policy.security.allowPromiscuous = $false
12 $swspec.policy.security.forgedTransmits = $false
13 $swspec.policy.security.macChanges = $false
14 |
15 $vh01netsysmoref.UpdateVirtualSwitch( $vh01vsw0.name,$swspec )
```

我想强调这个论坛帖子的目的是想强调什么方式才是将 PowerShell 和 vSphere API 对象协调工作的最佳方式。当你修改一个现有对象的时候，复制该对象的属性，然后再修改对象的属性。这 will 为你节省不必再通过 API 文档找到对象的正确默认值，因为那可能是一件头痛的事情，尤其当你去尝试使用对象的新属性的时候。但是，要记住，当你花时间来从头创建新对象的时候，那么这种便捷的方式对你不会有任何帮助。

PowerShell 管理 Windows 7 远程桌面的技巧

通过 PowerShell 脚本和 cmdlets 管理 Windows 7 远程桌面并不神秘，但是很多人在系统初始配置时容易出错。下面，我将讲述如何通过 PowerShell 执行远程 Windows 7 桌面管理任务和故障诊断。

启用 WinRM

为支持 [PowerShell](#) 管理远程桌面，管理主机上需要启用 WS-Management (Windows Remote Management) 服务。该服务默认的启动方式为手动，所以您需要更改为自动方式并启用服务。还可以通过在 PowerShell 中输入如下命令来确认服务是否启动：

```
Get-Service WinRM
```

在需要被管理的机器上也需要完成一些准备工作，第一件事情就是在所有需要管理的桌面上安装 IIS (Internet Information Service)。打开 Control Panel 点击 Programs，找到 Turn Windows Features On or Off。在接下来的画面中选择 Internet Information Services，并点击 OK。

在 IIS 安装完成后，启用 WS-Management (Windows Remote Management) 服务。接下来，您需要把 IIS 配置为 WinRM (Windows Remote Management Service) 服务的接收器。[Windows 7](#) 提供了自动完成方式，只需打开命令窗，输入如下命令：

```
WinRM QuickConfig
```



```
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\Administrator>winrm quickconfig
WinRM is not set up to receive requests on this machine.
The following changes must be made:

Set the WinRM service type to delayed auto start.
Make these changes [y/n]? y
WinRM has been updated to receive requests.
WinRM service type changed successfully.
WinRM is not set up to allow remote access to this machine for management.
The following changes must be made:

Create a WinRM listener on HTTP://* to accept WS-Man requests to any IP on this machine.
Enable the WinRM firewall exception.
Make these changes [y/n]? y
WinRM has been updated for remote management.
Created a WinRM listener on HTTP://* to accept WS-Man requests to any IP on this machine.
WinRM firewall exception enabled.
C:\Users\Administrator>_
```

图 A: 在远端 PC 上快速配置 WinRM 接收器

该命令自动修改 WinRM 服务启动方式、在防火墙中添加额外项目以及配置 WinRM 接收器。如图 A 中看到，可以远程确认所有动作。

测试远程 PowerShell

最简单的测试通过 PowerShell 命令管理远程 Windows 7 PC 是否可行的方式是用如下命令返回主机名称：

```
Gc env:computername
```

如果您想通过同一条命令管理远程主机，命令方式如下：

```
Invoke-command -ComputerName <the name of the remote computer> -
ScriptBlock {GC ENV:computername}
```



```
PS C:\Users\Administrator> invoke-command -computername lab-u7 -scriptblock{gc env:computername}
LAB-U7
PS C:\Users\Administrator> _
```

图 B: 该命令返回远程计算机名

例如，我对名为 Lab-W7（图 B）的远端主机运行该命令，可以看到成功返回主机名。显然这个命令在实际中没有用处，因为您已经知道主机名。实际上，在该命令本身就包含了计算机名。不过，运行该命令可以很好地说明您是否已经连接到远程系统中。

更多 PowerShell 命令

虽然只演示了如何返回计算机名的命令，实际上您可以通过运行任意的命令执行对远程计算机的管理。只需把两个大括号 {} 之间的命令进行替换就可以了。

这项技术在只需运行少量命令时非常管用，如果需要执行的是一系列命令，最好使用 New-PSSession cmdlet。

另外，关于我介绍的这个方法您还需要弄清楚，它可以适用于对多个远程计算机的管理。您需要更改的只是输入所有要被管理的计算机名，名字之间键入一个空格分隔。

远程 PowerShell 中存在的陷阱

本文开头提过，在对远程桌面使用 PowerShell 的时候可能遇到问题。当我在写这篇文章的时候，WinRM QuickConfig 命令被锁定，最终返回错误。在经历了六个小时的故障诊断后发现防火墙阻止了到远程 PowerShell 的访问。

您可能还会遇到权限问题。您登陆的计算机账户需要具备对所有被管理远程主机的管理员权限才可以。

另外一点，本地计算机需要可以解析远程计算机的 IP 地址。如果 DNS 解析不能正常工作，您需要输入的就是不是计算机名而是它们的 IP 地址。

防火墙也会导致 WinRM 失效。防火墙需要可以接受界内和界外的 HTTP 访问。WinRM QuickConfig 命令可以自动在 Windows 防火墙内添加例外，但是如果您使用了第三方防火墙就需要手动添加。请注意 WinRM1.1 和早期版本使用了 TCP 80 端口，而 WinRM 2.0（Window 7 中使用的版本）使用了 5985 端口代替。

最后，如果您遇到认证问题，那么需要制定远程计算机为被信任主机。例如，我需要指定名为 Lab-W7 的主机被信任。在本地计算机上输入如下命令：

```
WinRM s winrm/config/client '@{TrustedHosts="Lab-W7"}'
```

可以看到，在通过 PowerShell 命令管理远程桌面时是有一些问题需要考虑清楚的。尽管如此，一旦配置正确，这种管理方式还是很稳定的。

(来源: TechTarget 中国)

使用 Windows PowerShell 管理 Citrix XenDesktop 5

在创建 [XenDesktop 5](#) 时，Citrix 便完全支持 [PowerShell](#)。实际上，XenDesktop 5 SDK 包括了 100 多个 PowerShell cmdlets，为数不尽的管理任务提供了帮助。

让我们一起来了解一些对管理虚拟桌面环境特别有帮助的 cmdlets，这其中包括中断虚拟桌面会话、增加管理员、创建虚拟机快照。

创建虚拟机快照

在很多情况下都要使用虚拟机快照。在 XenDesktop 5 中，能够使用 PowerShell 创建[虚拟机快照](#)。如果正在创建操作脚本，想在脚本运行时创建快照的话是非常便利的。可以使用 New-HypVMSnapshot cmdlet 创建快照。该命令完整的语法如下：

```
New-HypVMSnapshot [-LiteralPath] <String> [-SnapshotName] <String> [-AdminAddress <String>] [[-SnapshotDescription] <String>] [<CommonParameters>]
```

尽管这个 cmdlet 看起来很恐怖，但是使用却相当简单。下面这个例子在名为 MyHV 的 hypervisor 中创建虚拟机 MyVM 的快照：

```
New-HypVMSnapshot -LiteralPath XDHyp:\Connections\MyHV\MyVm.vm -SnapshotName "New snapshot" -SnapshotDescription "Example snapshot" XDHyp:\Connections\MyHV\MyVm.vm\New snapshot.snapshot
```

获取管理员账户列表

有时需要检索配置为主机服务的管理员账号列表，这时 Get-HypAdministrator cmdlet 便派上用场了。和之前的 cmdlet 类似，Get-HypAdministrator cmdlet 有一些可选的参数。该命令完整的语法如下：

```
Get-HypAdministrator [-AccountSid <String>] [-ReadOnly] [-ReturnTotalRecordCount] [-MaxRecordCount <Int32>] [-Skip <Int32>] [-
```



```
SortBy <String>] [-Filter <String>] [-AdminAddress <String>]  
[<CommonParameters>]
```

这些参数同样可以用来过滤管理员列表。例如，如果你只想查看具有只读权限的管理员，那么可以使用 Read Only 参数。但是，如果想查看所有管理员账户的列表，只需要使用不带任何参数的 Get-HypAdministrator cmdlet 即可。

检索任务的历史记录

你可能也想查看任务的历史记录，这时 Get-HypTask cmdlet 便派上用场了。该命令完整的语法如下：

```
Get-HypTask [[-TaskId] <Guid>] [-Type <JobType>] [-Active <Boolean>] [-  
ReturnTotalRecordCount] [-MaxRecordCount <Int32>] [-Skip <Int32>] [-  
SortBy <String>] [-Filter <String>] [-AdminAddress <String>]  
[<CommonParameters>]
```

和 Get-HypAdministrator cmdlet 类似，Get-HypTask cmdlet 所有的参数都是可选的。如果想查看完整的任务历史记录，只需要输入不带任何参数的 Get-HypTask cmdlet 即可。该命令所提供的参数用于过滤输出结果或返回额外的信息。例如增加 ReturnTotalRecordCount 参数将显示返回记录的总数。

增加配置管理员

通过 PowerShell 能够完成的另一个任务就是在配置管理员列表中增加活动目录用户或组。可以使用 New-ConfigAdministrator cmdlet 命令添加配置管理员。该命令完整的语法如下：

```
New-ConfigAdministrator [-Account] <String> [-ReadOnly] [-AdminAddress  
<String>] [<CommonParameters>]
```

在所有参数当中，唯一的必选参数就是 Account。该参数后面跟的是域名以及你想授予访问权限的用户名或组名。也可以使用 Read Only 参数将用户或组配置为只读的配置管理员。下面的这个例子为 Lab 域中的用户 User1 分配管理员权限：
New-ConfigAdministrator -Account Lab\User1

中断会话

有时，你可能需要中断处于活动状态的会话，这时 Disconnect-BrokerSession cmdlet 便派上用场了，该命令完整的语法如下：

```
Disconnect-BrokerSession [-InputObject] <Session[]> [-AdminAddress  
<String>] [<CommonParameters>]
```

唯一的必选参数是 InputObject，该参数后面需要跟一个数值。通常使用管道输入而不是手动指定一个输入对象将更容易。例如，如果你想中断 Lab 域中的用户 User1，可以通过输入管道 cmdlets 实现（注意：即使从技术角度来看，InputObject 参数是必须的，但是该参数并不是必须的）：

```
Get-BrokerSession -UserName Lab\User1 | Disconnect-BrokerSession
```

以上只是众多 PowerShell cmdlet 中很小的一部分。如果你对 PowerShell 脚本不太熟悉，如下方式可以提供帮助：

首先，Citrix Desktop Studio（为 XenDesktop 提供的图形用户界面）显示 PowerShell 和在图形用户界面中执行的众多操作是等价的。这意味着在执行管理任务时，有时可以使用 PowerShell 达到同样的效果。

Citrix 还提供了所有 XenDesktop PowerShell cmdlet 的命令参考。你可以在 Citrix 的支持页面找到该命令参考。

(来源: TechTarget 中国)

使用 Windows PowerShell 管理远程桌面设备

微软提供了图形用户界面和命令行界面协助虚拟桌面部署。如果你已经在 Windows 2008 R2 里安装了远程桌面设备，我们现在来看看如何使用 Windows PowerShell 对其进行管理。

与远程桌面服务（RDS）PowerShell 模块一起安装的管理界面由两部分组成：

1. **Providers** 使用树形架构呈现数据结构（类似于文件系统）的逻辑视图，以便处理 RDS。Providers 也利用像 `get-item`、`get-childitem`、`get-acl`、`set-acl` 和 `new-item` 这样的通用 cmdlets。微软在网站上提供了[更多关于供应商的信息](#)。

2. 与 RDS 相关的 **Cmdlets** 提供了超出 Providers 提供范围的管理。

步骤一、安装远程桌面服务管理

第一步很简单，事实上，你不需要做任何事。

provider 和 cmdlets 默认下是与 RDS 组件一起安装的。

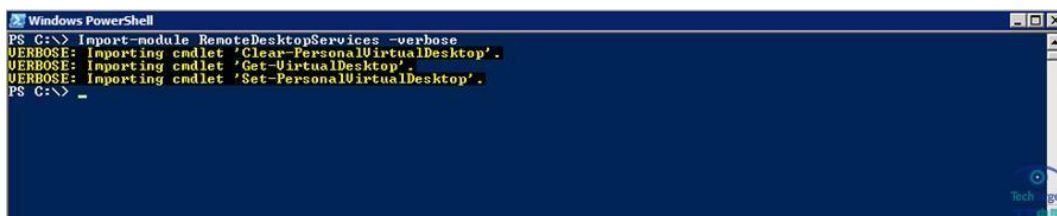
步骤二、运行 PowerShell RDS 模块

启用 PowerShell RDS 模块有两个选项：

1. 从 Start Menu——All Programs——Administrative Tools——Remote Desktop Services——Remote Desktop Services PowerShell 启动。
2. 通过 Import-Module 内置到 cmdletPowerShell 添加模块到现有 PowerShell 模块。

`Import-module RemoteDesktopServices -- verbose`。

注意：如果没有禁用用户账户控制，你可以以管理员身份登录。



```
PS C:\> Import-module RemoteDesktopServices -verbose
VERBOSE: Importing cmdlet 'Clear-PersonalVirtualDesktop'.
VERBOSE: Importing cmdlet 'Get-VirtualDesktop'.
VERBOSE: Importing cmdlet 'Set-PersonalVirtualDesktop'.
PS C:\>
```

点击图片本身就能放大

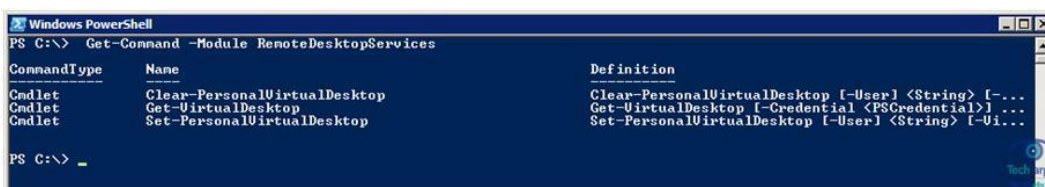
步骤三、查看目前资料

启动模块的最大障碍在于计算你拥有什么。有几种方式解决，包括 `get-command` 和默认的 provider cmdlets。

Get-command

`Get-command` 是一个内置的 cmdlet，说明哪些命令是可用的。 `get-command` 的一个最有用的参数是 `-module`，在模块里启用这个命令。

看看由 RDS 模块提供的 cmdlets。



注意：关于更多 `get-command` 的信息，请参看 `get-command` 上的 `get-help get-command -- full` 或者微软帮助页面。

Default Provider cmdlets

有许多内置的 provider cmdlets 旨在与所有 providers 协调工作（如注册表、文件系统和 RDS）。

- **Get-Childitem** `Get-Childitem (dir:)` 在目前为止获取所有子项目。例如： `PS> get-childitem RDS:\ConnectionBroker`
- **Get-Item** `(gi:)` 获取供应商的某个项目。例如， `PS> get-item RDS:\ConnectionBroker\DisplayName`
- **Get-Member** `(gm:)` 列出某个项目的属性。例如， `PS> get-item RDS:\ConnectionBroker\DisplayName | get-member`
- **Set-item**：用于给某个项目设置属性。例如， `PS> get-item RDS:\ConnectionBroker\DisplayName | set-item -value "RDS CB"`

总的说来，PowerShell 是管理远程桌面服务的强大工具。关于虚拟桌面部署的更多信息，请参看[微软的远程桌面服务团队博客](#)。

(来源: TechTarget 中国)