PowerCLI

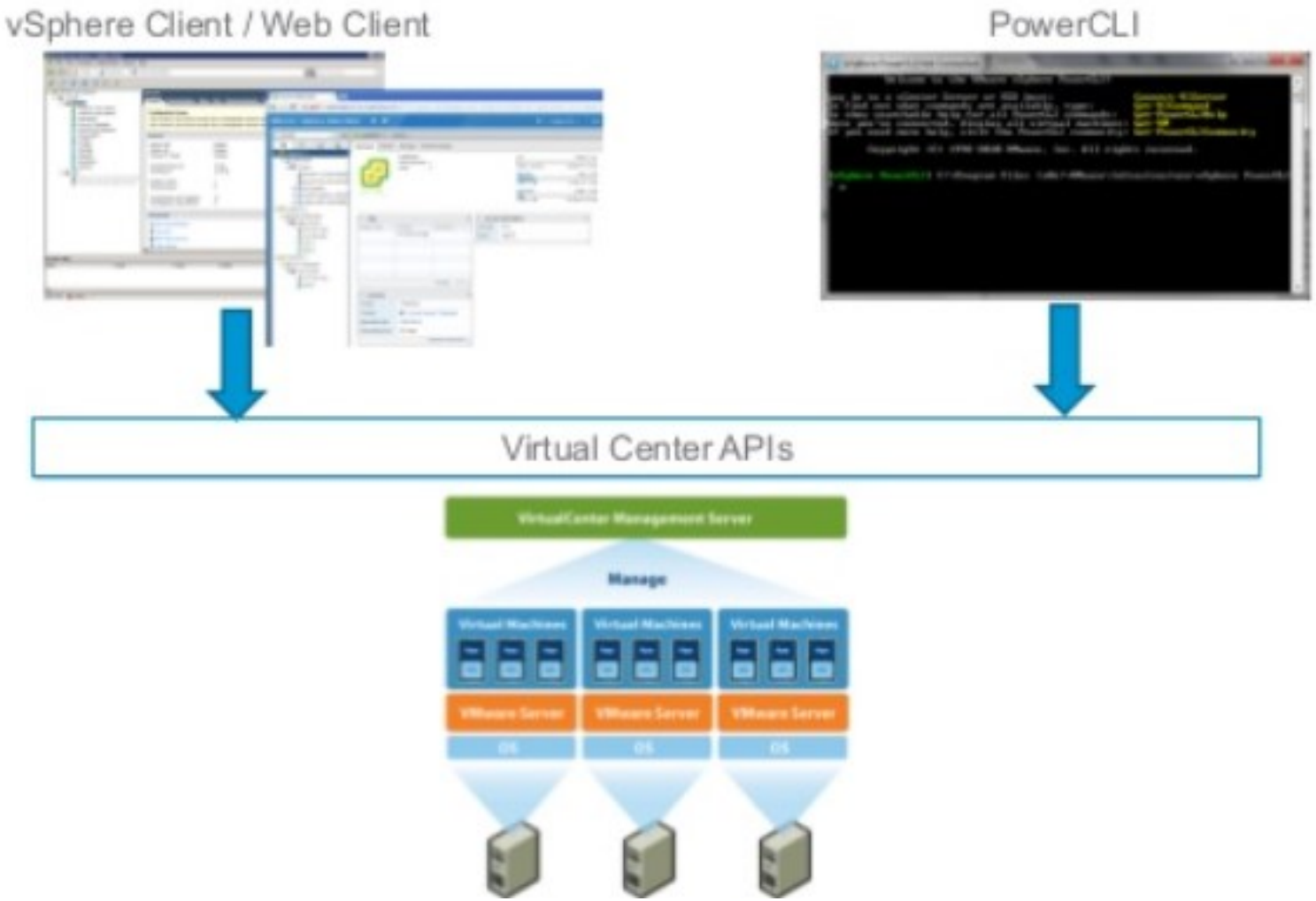
проф. д-р инж. Христо Вълчанов

http://cs.tu-varna.bg

VMware PowerCLI

- Средство за автоматизиране на управлението на Vmware виртуални инфрастуктури.
- Командите на PowerCLI се изпълняват в Windows PowerShell като командлети.

VMware PowerCLI



Инсталиране на PowerCLI

Изисквания:

- Microsoft .NET Framework (в зависимост от версията на vSphere, за vSphere 6.0 -> v.4.5).
- Windows PowerShell минимално 3.0.
- Изисква се Интернет връзка.

Инсталира се чрез

```
Install-Module –Name VMware.PowerCLI
```

Инсталиране на PowerCLI

Administrator: Windows PowerShell

```
PS C:\WINDOWS\system32> Install-Module -Name VMware.PowerCLI

Untrusted repository
You are installing the modules from an untrusted repository. If you trust this repository, change its
InstallationPolicy value by running the Set-PSRepository cmdlet. Are you sure you want to install the modules from
'PSGallery'?
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is "N"): A
PS C:\WINDOWS\system32>
```

Проверка за инсталиране на PowerCLI

Administrator: Windows PowerShell

```
PS C:\WINDOWS\system32> Get-Module VMware.PowerCLI -ListAvailable


    Directory: C:\Program Files\WindowsPowerShell\Modules


ModuleType Version    Name                                ExportedCommands
---------- -------    ----                                ----------------
Manifest   12.1.0.... VMware.PowerCLI


PS C:\WINDOWS\system32>
```

Политика на изпълнение на PowerShell

- Извеждане на текущата политика:

```
Get-ExecutionPolicy
```

Възможни политики:

Restricted - не се разрешава изпълнение на скриптове.

AllSigned - може да се изпълняват скриптове само от доверени разработчици

RemoteSigned – могат да се стартират собствени скриптове и такива от доверени разработчици

Unrestricted - могат да се изпълняват всякакви скриптове. Не се препоръчва.

Политика на изпълнение

- Установяване на текуща политика:

```
Set-ExecutionPolicy политика
```

Най-често се използва **Remote-Signed.**

Синтаксис на командлет

`VERB-NOUN параметри`

VERB – действие (Get, Set, Start …)

NOUN – предмет на действието (ВМ, сървъри, обекти …)

Команлетите връщат като резултат обекти, които могат да се използват в последствие.

Свързване към сървър

- Първоначална стъпка.
- Ако е налице vCenter, може да се осъществи връзка към него.
- Може да се осъществи връзка и само към ESXi хост.

```
Connect-VIServer [-Server] <String[]>
      [-Port <Int32>] [-Protocol <String>]
      [-Credential <PSCredential>] [-User <String>]
      [-Password <String>] [-Session <String>]
      [-NotDefault] [-SaveCredentials] [-AllLinked]
      [-Force] [<CommonParameters>]
```

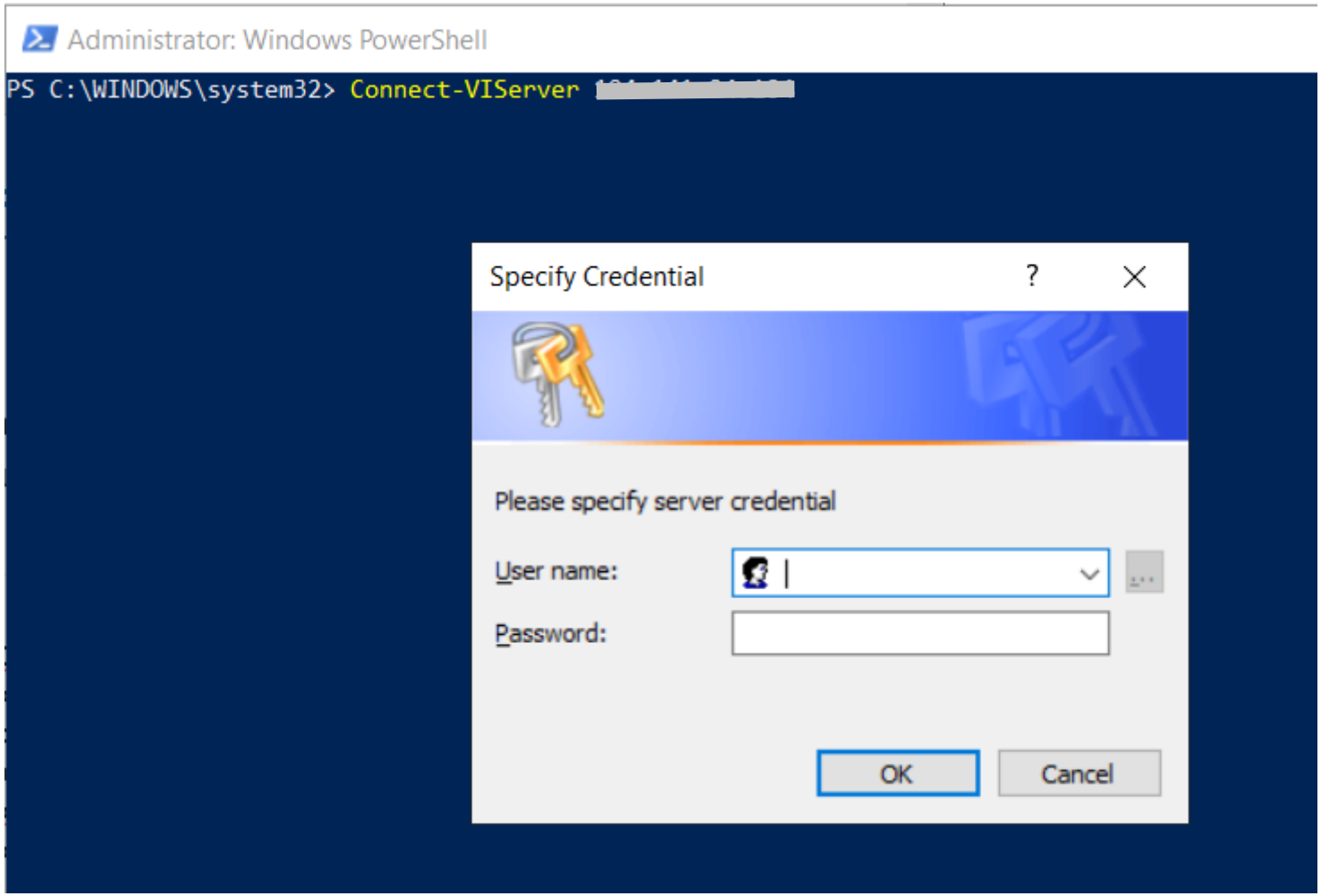
Игнориране на сертификат

Ако vCenter или ESX хост нямат валидни сертификати, ще се изведат предупредителни съобщения.

Тези съобщения могат да се игнорират.

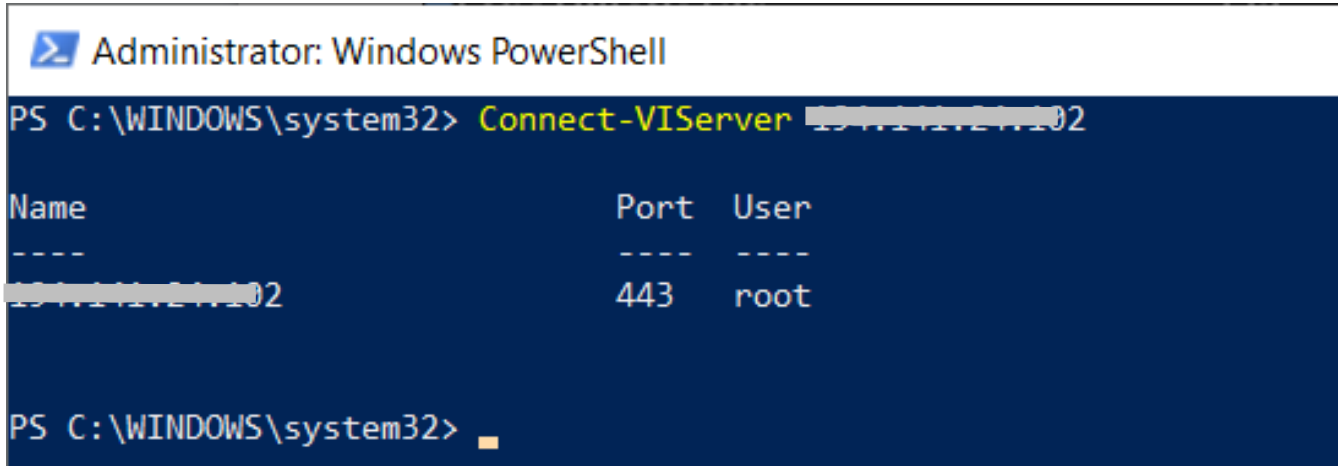
```
Set-PowerCLIConfiguration -InvalidCertificateAction
                Ignore
```

Свързване към сървър



Задаване на кредeнциали

```
Connect-VIServer IP_server -User admin -Password pass
```



Откачане от сървър

```
Disonnect-VIServer IP_server
```

Select Administrator: Windows PowerShell

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\WINDOWS\system32> Connect-VIServer [redacted]

Name                           Port  User
----                           ----  ----
194.141.30.125                 443   root


PS C:\WINDOWS\system32> Disconnect-VIServer [redacted]

Confirm
Are you sure you want to perform this action?
Performing the operation "Disconnect VIServer" on target "User: root, Server: [redacted], Port: 443".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is "Y"): y
PS C:\WINDOWS\system32>
```

Свързване от скрипт

При използване на скриптове диалогът с командлетите трябва да се реализира по следния начин:

- Игнорират се потвържденията (указва се като параметър).
- Извежданите съобщения от командлетите трябва да се пренасочат към NULL устройството с *Out-null*.

```
Set-PowerCLIConfiguration -InvalidCertificateAction
        Ignore -Confirm:$false | Out-null
```

Пример за свързване

```
# Load VMware modules
Get-Module -Name VMware* -ListAvailable | Import-Module | Out-null

# Ignore Invalid certificate message
Set-PowerCLIConfiguration -InvalidCertificateAction Ignore -Confirm:$false `
          | Out-null

$VMHost = "192.168.0.100"
$Password = „MyPassword"
$User="root"

$connected = $false

try {
  Connect-VIserver -Server $VMHost -User $User -Password $Password  `
          -WarningAction SilentlyContinue -errorAction "Stop" | Out-null
  $connected = $true
  write-host "Connected."
}
catch {
  Exit 1
}

if ($connected) {
  Disconnect-VIserver -Server * -Force -Confirm:$false
  write-host "Disconnected."
}
```

Пример за свързване

```
PS C:\WINDOWS\system32> C:\Users\hwalchan\Documents\Labs\Virtualization\LabsRemote\test1.ps1
Connected.
Disconnected

PS C:\WINDOWS\system32> |
```

Пример

Извеждане на информация за хост

```
Get-VMHost
```

```
PS C:\WINDOWS\system32> C:\Users\hwalchan\Documents\Labs\Virtualization\LabsRemote\test2.ps1

Name                ConnectionState PowerState NumCpu CpuUsageMhz CpuTotalMhz   MemoryUsageGB   MemoryTotalGB Version
----                --------------- ---------- ------ ----------- -----------   -------------   ------------- -------
194.141.30.125      Connected       PoweredOn       6         795       11394          29,742          31,967   5.5.0


PS C:\WINDOWS\system32> |
```

Пример

Извеждане списък от виртуални машини

Get-VM

```
PS C:\WINDOWS\system32> C:\Users\hwalchan\Documents\Labs\Virtualization\LabsRemote\test1.ps1
Connected.

Name                 PowerState Num CPUs MemoryGB
----                 ---------- -------- --------
ESXi                 PoweredOff 4        7,473
Zabbix               PoweredOff 1        1,000
WServer2008Zabbix    PoweredOff 1        4,000
ESXi_Ivo             PoweredOff 4        3,879
VT vCenter Server... PoweredOff 2        8,000
PowerManager         PoweredOff 1        4,000
vCenter 6.0          PoweredOn  4        16,000
RouterA              PoweredOn  1        2,000
HostA                PoweredOn  1        2,000
RouterB              PoweredOn  1        2,000
HostB                PoweredOn  1        2,000
RouterC              PoweredOn  1        2,000
HostC                PoweredOn  1        2,000
Disconnected



PS C:\WINDOWS\system32>
```

Пример

Извеждане списък от виртуални машини с филтър

```
Get-VM | Where-Object NumCPU -GT 2
```

```
PS C:\WINDOWS\system32> C:\Users\hwalchan\Documents\Labs\Virtualization\LabsRemote\test1.ps1
Connected.

Name                PowerState Num CPUs MemoryGB
----                ---------- -------- --------
ESXi                PoweredOff 4        7,473
ESXi_Ivo            PoweredOff 4        3,879
vCenter 6.0         PoweredOn  4        16,000
Disconnected


PS C:\WINDOWS\system32> |
```

Пример

Стартиране на виртуална машина

```
StartVM –VM VM_name
Get-VM –Name VM_name | StartVM
```

Ако се стартират няколко ВМ, за да се избегне изчакването (командлетите се изпълняват един след друг), може те да се стартират асинхронно:

```
StartVM –VM * -RunAsync
```

Пример за стартиране

```
...

try {
  Connect-VIserver -Server $VMHost -User $User -Password $Password  `
        -WarningAction SilentlyContinue -errorAction "Stop" | `
        Out-null
  $connected = $true
}
catch {
  Exit 1
}

if ($connected) {
  $vms = Get-VM

  Foreach ($vm in $vms) {
    # The VM is powered OFF
    if ($vm.PowerState -eq "PoweredOff") {
            Start-VM -VM $vm.name -Confirm:$false | Out-null
    }
  }
  Disconnect-VIserver -Server * -Force -Confirm:$false
}
```

Пример

Спиране на виртуална машина

- Ако има инсталирани VMware Tools:

```
Shutdown-VMGuest -VM VM_name
```

- Ако няма инсталирани VMware Tools:

```
Stop-VM -Name VM_name
Stop-VM -Name VM_name -Kill
```

Проверка за инсталирани VMware Tools:

```
if ($vm.config.Tools.ToolsVersion -ne 0) { ...
```

Пример

Спиране на ВМ, която не е стартирана води до грешка, затова трябва да се прави проверка:

```
Get-VM | Where-object powerstate -eq “PoweredOn” |
                Shutdown-VMGuest


if ($vm.PowerState -eq "PoweredOn") {
      Shutdown-VMGuest –VM $vm.Name
}
```

Пример за спиране

```
...

try {
  Connect-VIserver -Server $VMHost -User $User `
        -Password $Password  `
        -WarningAction SilentlyContinue -errorAction "Stop" | `
        Out-null
  $connected = $true
}
catch {
  Exit 1
}

if ($connected) {
  $vms = Get-VM | Where-object powerstate -eq "PoweredOn"

  Foreach ($vm in $vms) {
    Shutdown-VMGuest -VM $vm.name -Confirm:$false | Out-null
  }
  Disconnect-VIserver -Server * -Force -Confirm:$false
}
```

Пример

Информация за виртуални комутатори

```
Get-VirtualSwitch | Format-List
```

```
Id                : key-vim.host.VirtualSwitch-vSwitch0
Key               : key-vim.host.VirtualSwitch-vSwitch0
Name              : vSwitch0
NumPorts          : 2352
NumPortsAvailable : 2333
Nic               : {vmnic0}
Mtu               : 1500
VMHostId          : HostSystem-ha-host
VMHost            : 194.141.30.125
VMHostUid         : /VIServer=root@194.141.30.125:443/VMHost=HostSystem-ha-host/
Uid               : /VIServer=root@194.141.30.125:443/VMHost=HostSystem-ha-host/VirtualS
                    Switch0/
ExtensionData     : VMware.Vim.HostVirtualSwitch

Id                : key-vim.host.VirtualSwitch-vSwitch1
Key               : key-vim.host.VirtualSwitch-vSwitch1
Name              : vSwitch1
NumPorts          : 2352
```

Пример

Извеждане на мрежовите адаптери

```
Get-NetworkAdapter -VM vm
```

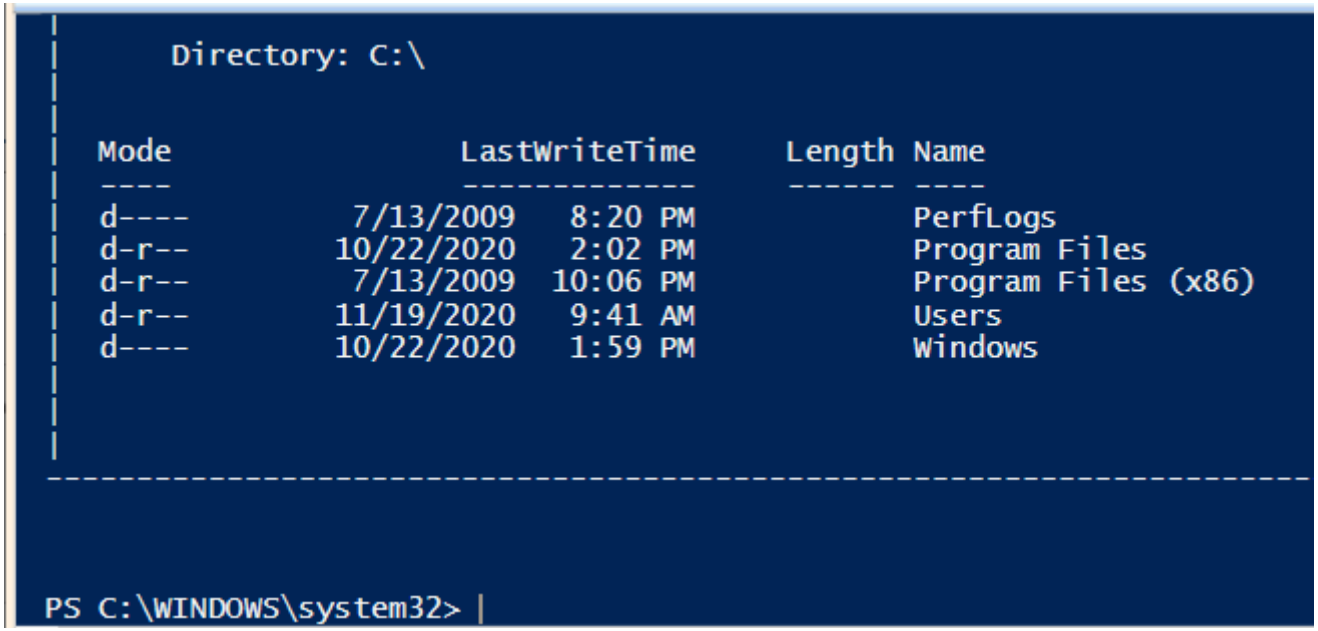
```
PS C:\WINDOWS\system32> C:\Users\hwalchan\Documents\Labs\Virtualization\LabsRemote\
Connected.

Name                 Type            NetworkName     MacAddress         WakeOnLan
                                                                          Enabled
----                 ----            -----------     ----------         ---------
Network adapter 1    e1000           VM Network      00:0c:29:8c:1a:91       True
Network adapter 1    e1000           VM Network      00:0c:29:73:e3:99       True
Network adapter 2    e1000           Internal net    00:0c:29:73:e3:a3       True
Network adapter 1    e1000           Internal net    00:0c:29:c0:9b:7b       True
Network adapter 1    e1000           VM Network      00:0c:29:fd:45:7b       True
Network adapter 2    e1000           Internal net 2  00:0c:29:fd:45:85       True
Network adapter 1    e1000           Internal net 2  00:0c:29:f8:fe:6f       True
Network adapter 1    e1000           VM Network      00:0c:29:d8:c2:4a       True
Network adapter 2    e1000           Internal net 3  00:0c:29:d8:c2:54       True
Network adapter 1    e1000           Internal net 3  00:0c:29:ac:94:52       True
Disconnected
```

Пример

Изпълнение на скрипт на виртуална машина

```
$script = "dir c:\"
Invoke-VMScript -VM vm -ScriptText $script `
      -GuestUser "admin" -GuestPassword „xxxxx"
```



Пример

Създаване на виртуална машина

```
New-VM –Name 'VM_name'
New-VM -Name 'VMname' -Datastore 'Datastore‘
New-VM -Name 'VMname' -DiskStorageFormat thin
      -NumCpu 4 -MemoryMB 2048 -DiskMB 40960 -
      NetworkName ‘Business network‘
```

По подразбиране се създава ВМ с 1 CPU, 256Mb RAM 4Gb диск и NIC свързана към „Internal network”

Клониране на ВМ:

```
New-VM -Name 'VMname' -VM 'VMtoClone' -Datastore
      'Datastore' -DiskStorageFormat thin
```

Пример

Изтриване на виртуална машина

```
$Exists = Get-VM -name 'VMname' -ErrorAction
                    SilentlyContinue
if ($Exists) {
      Remove-VM -VM 'VMName' -DeleteFromDisk
      -Confirm:$false -RunAsync | Out-Null
}
```

Изпълнение на esxcli от PowerCLI

Получаване на обект esxcli

```
Get-ESXCLI
```

```
Connect-VIserver -Server 192.168.0.100

$vmh = Get-VMHost

$esxcli = Get-EsxCli -VMHost $vmh -V2
```

V2 – използване на новите функционалности със стари скриптове

Изпълнение на esxcli от PowerCLI

Изпълнение на команда под CLI:

```
$ esxcli network nic list
```

Изпълнение чрез PowerCLI:

```
esxcli.network.nic.list
```

```
PS C:\WINDOWS\system32> C:\Users\hwalchan\Documents\Labs\Virtualization\LabsRemote

===================
EsxCliMethodElement: list

   Methods:
   --------
   Nic[] Invoke()
   string Help()



PS C:\WINDOWS\system32>
```

Изпълнение на esxcli от PowerCLI

```
esxcli.network.nic.list.Invoke()
```

```
Description : Intel Corporation 82574L Gigabit Network Connection
Driver      : e1000e
Duplex      : Full
Link        : Up
MACAddress  : 00:19:99:e3:6e:cc
MTU         : 1500
Name        : vmnic0
PCIDevice   : 0000:006:00.0
Speed       : 100

Description : Intel Corporation 82574L Gigabit Network Connection
Driver      : e1000e
Duplex      : Half
Link        : Down
MACAddress  : 00:19:99:e3:6e:ce
MTU         : 1500
Name        : vmnic1
PCIDevice   : 0000:007:00.0
```

Предаване на параметри към esxcli

Когато методът **Invoke()** изисква параметри е необходимо създаване на атрибутен обект (хеш таблица) с необходимите параметри.

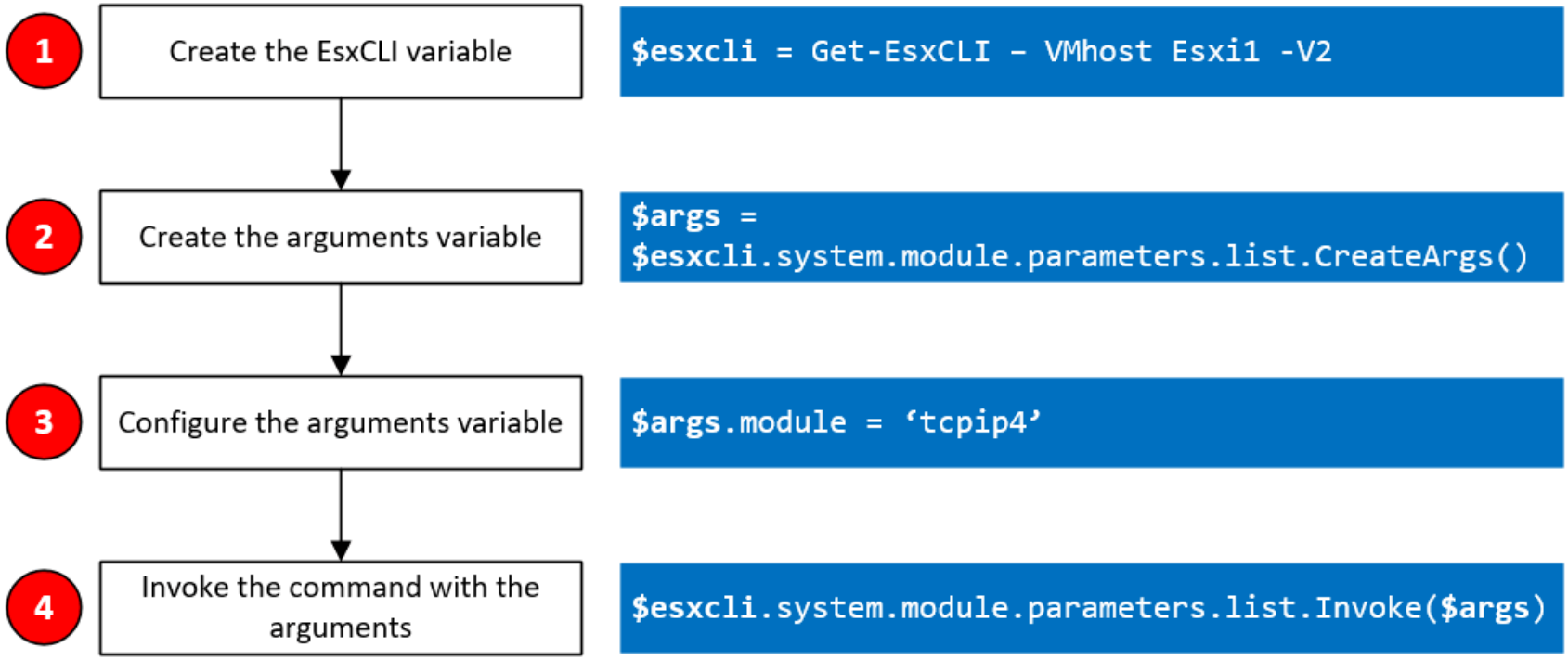
```
CreateArgs()

esxcli ... Invoke($MyArgs)
```

Предаване на параметри към esxcli

<u>Пример:</u>
Проверка дали IPv6 е разрешен на хоста

```
$esxcli system module parameters list -m tcpip4
```



Предаване на параметри към esxcli

```
...
if ($connected) {
  $vmh = Get-VMHost
  $esxcli = Get-EsxCli -VMHost $vmh -V2
  $arguments = $esxcli.system.module.parameters.list.CreateArgs()
  $arguments.module = “tcpip4”
  $esxcli.system.module.parameters.list.Invoke($arguments)
  ...
}
```

Предаване на параметри към esxcli

```
PS C:\WINDOWS\system32> C:\Users\hwalchan\Documents\Labs\Virtualization\

Description                      Name                 Type Value
-----------                      ----                 ---- -----
# of provisioned H/W iSCSI ports ipportIscsiReserved int
Enable/Disable IPv6              ipv6                 int  0


PS C:\WINDOWS\system32> |
```

Въпроси?