Automation of Virtual Route Reflector Topologies using the ESXCLI

July 2014

Deepti Chandra

Table of Contents

White Paper ............................................................................................................................................................ 1 Automation of Virtual Route Reflector Topologies using the ESXCLI ………................................ 1 Executive Summary.............................................................................................................................................. 3 Introduction............................................................................................................................................................. 3 VMware command line scripting tools ........................................................................................................ 4 ESXCLI Architecture.............................................................................................................................................. 5 ESXCLI Command Overview.............................................................................................................................. 6 VRR topology creation using ESXCLI ............................................................................................................ 9 Virtual machine creation using ESXCLI ........................................................................................................11 Hybridging using ESXCLI ....................................................................................................................................13 Advantages…………………………………………………...........................................................................................15 Limitations ................................................................................................................................................................15 Conclusion .................................................................................................................................................................15

Executive Summary

This paper provides an overview of the ESXCLI architecture for the creation of virtual machines and hybridging the same with physical devices. Additionally, it demonstrates the use of a tool that can create a vRR topology, optionally with loading a baseline Junos configuration. It can be used for quick and easy deployments during initial rollouts for internal testing and customer demonstrations, reducing the learning curve overhead. The detailed procedure and the theoretical reasons are described here.

Introduction

With the increase in virtual infrastructure deployments, there has been an increase in the setup of scaled test beds that can mirror production networks for effective testing and initial field rollouts. Most often, these test beds are a combination of virtual and physical devices, interconnecting which can be complicated. New hypervisors/orchestration mechanisms often come with an associated overhead of getting up to speed with new technologies and relevant terminology, all of which can increase the time and effort put into testing. In large customer networks, where topologies consist of multiple virtual machines, using a graphical interface is not a feasible solution for provisioning or management of network infrastructure. Graphical user interfaces (GUIs) provide intuitive and conceptual ways to manage VMware environments, but they generally don't scale well for large or repetitive tasks. Activities such as mass provisioning and configuring of virtual machines (VMs) are best suited for VMware scripts and commands. The method described here explores one of the many VMware command line scripting interfaces, the ESXCLI and how it can be utilized to eliminate many of those caveats.

VMware command line scripting tools

VMware vSphere has several command line and scripting tools, some of which are listed below:

* ESXCLI : Run directly from the ESXi server shell
* vSphere Command-Line Interface (vCLI) : This command set allows you to run common system administration commands against ESXi systems from any machine with network access to those systems.
* PowerShell command-line Interface (PowerCLI): VMware vSphere PowerCLI provides a Windows PowerShell interface as an alternative to the vSphere API. Being built on top of the Power Shell, it is a popular choice amongst most Windows administrators. vCLI on the other hand tends to be more popular with Linux administrators because it has a Linux look and feel.

Apart from the above, there are numerous other command line interfaces, APIs and SDKs that are deployed for VMware automation in many customer networks based on the requirement and problem statement, in hand.

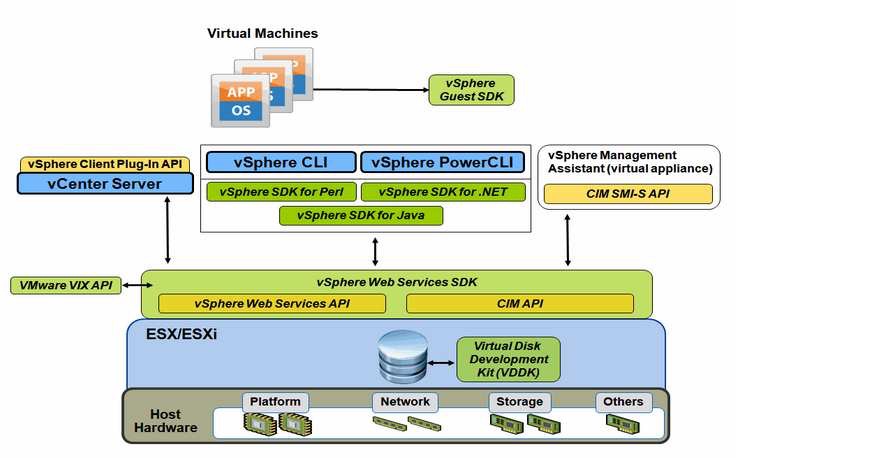


Figure 1 – vSphere CLI and API

ESXCLI Architecture

ESXCLI is a complete set of commands available in various formats:

* Run directly from the ESXi Server shell
* From the vSphere CLI package, available for Windows and Linux
* As part of the commands in the vMA
* From the Get-EsxCli cmdlet available with vSphere PowerCLI

ESXi 5.0 comes with a new ESXCLI command set that runs from the ESXi shell. This command set comes with an improved syntax and support for many additional namespaces that can be used for both local and remote command line administration. Every ESXi host comes with a version of the ESXCLI command in its ESXi shell. This command can only be used to perform local administration of the host, that is local host is always assumed.

There are two versions of the command – one that runs in the ESXi Shell, as described above and one that runs remotely as part of vCLI. Along with the local version of the ESXCLI command there is a remote version provided with the vCLI that is used for remote administration. While the command syntax is largely the same as the local version, when working remotely you need to provide additional connection options to specify the target ESXi host along with the user credentials.

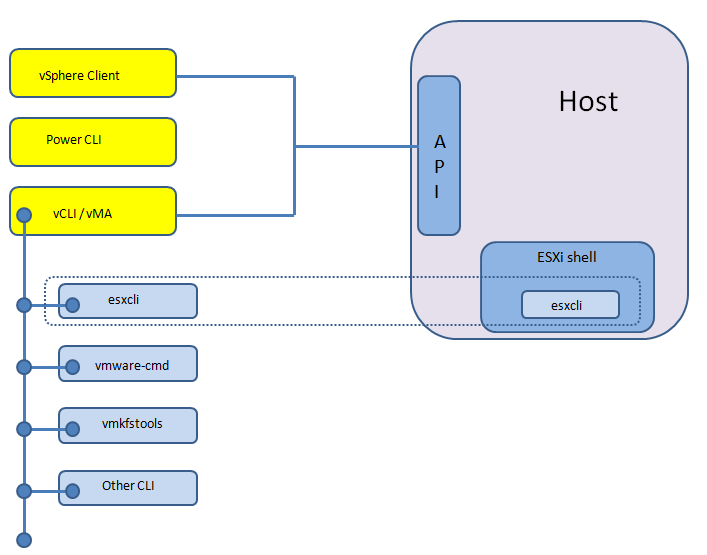


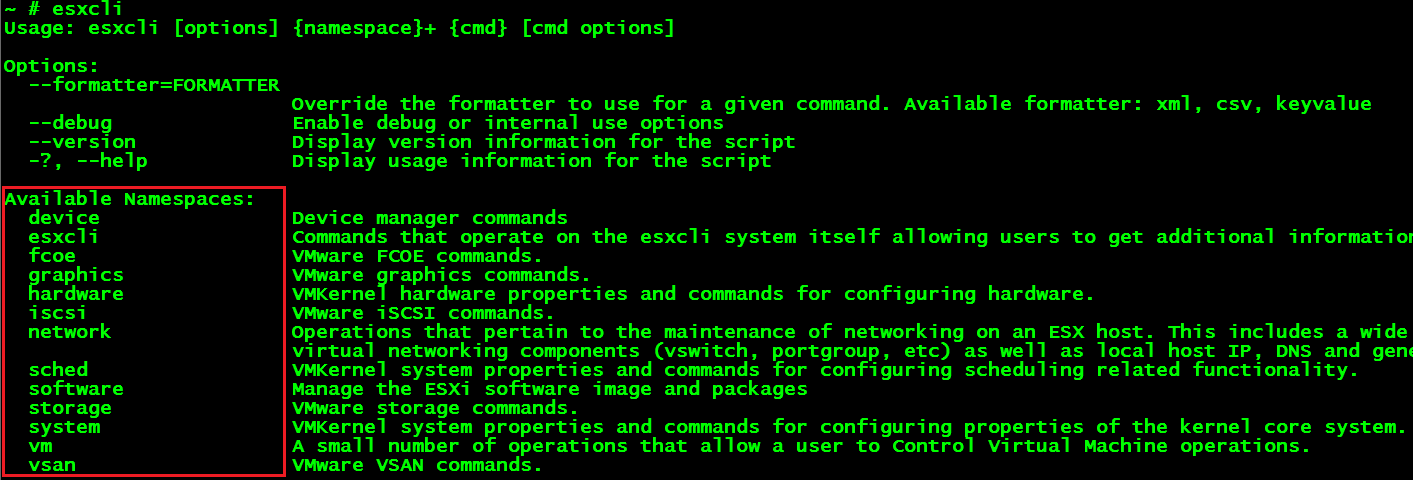
Figure 2 – ESXCLI Architecture

ESXCLI Command Overview

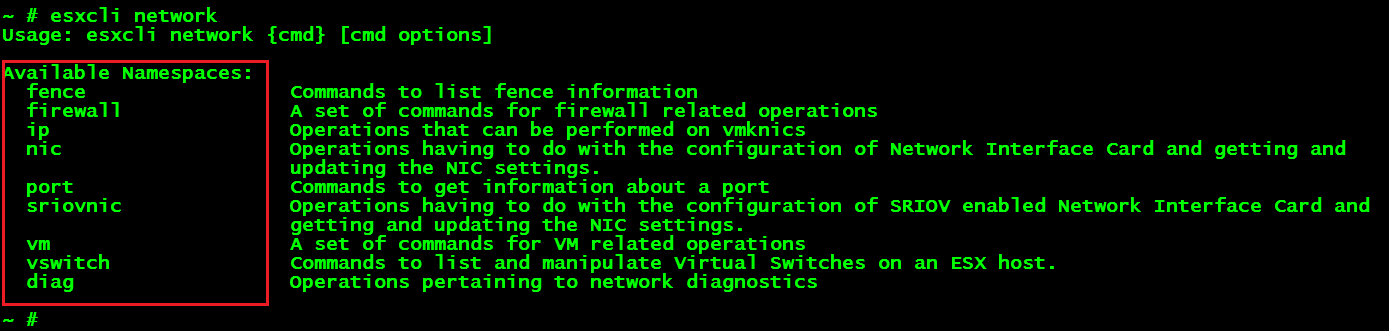
General syntax of the ESXCLI command is:

Usage: esxcli [options] {namespace}+ {cmd} [cmd options]

* Options : For connection information such as target host, user name etc
* Namespace : Namespaces group ESXCLI commands. Nested namespaces are also supported.
* Command : Can have additional options. Provides reports or modifies state on the system.

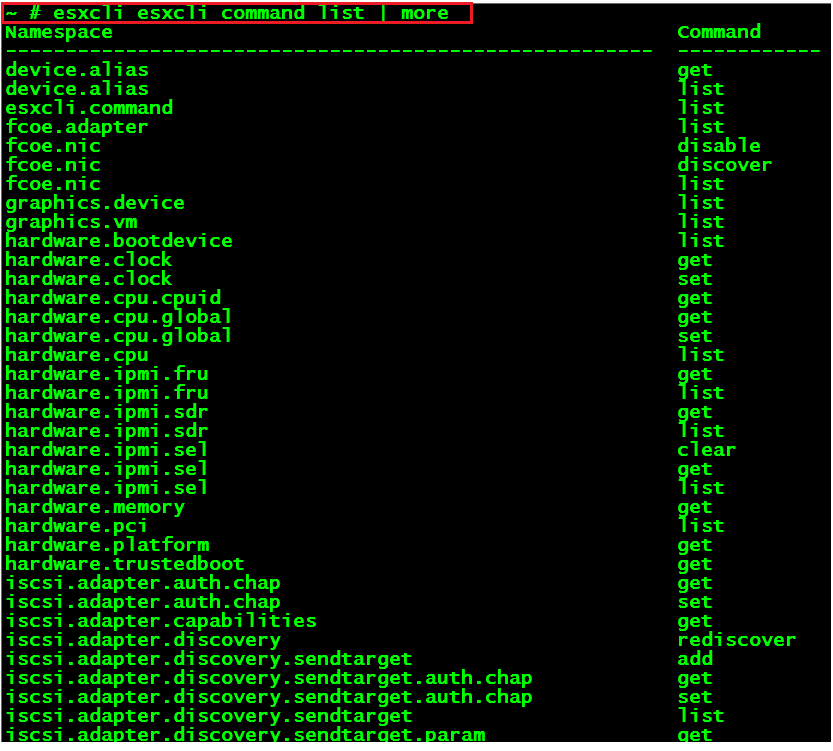


Highlighted above is the first level of namespaces. Nested namespaces include additional namespaces and commands within them.



Above is how you can navigate through nested namespaces.

To get a complete list of all namespaces and commands, the command ‘esxcli esxcli command list | more’ can be used.



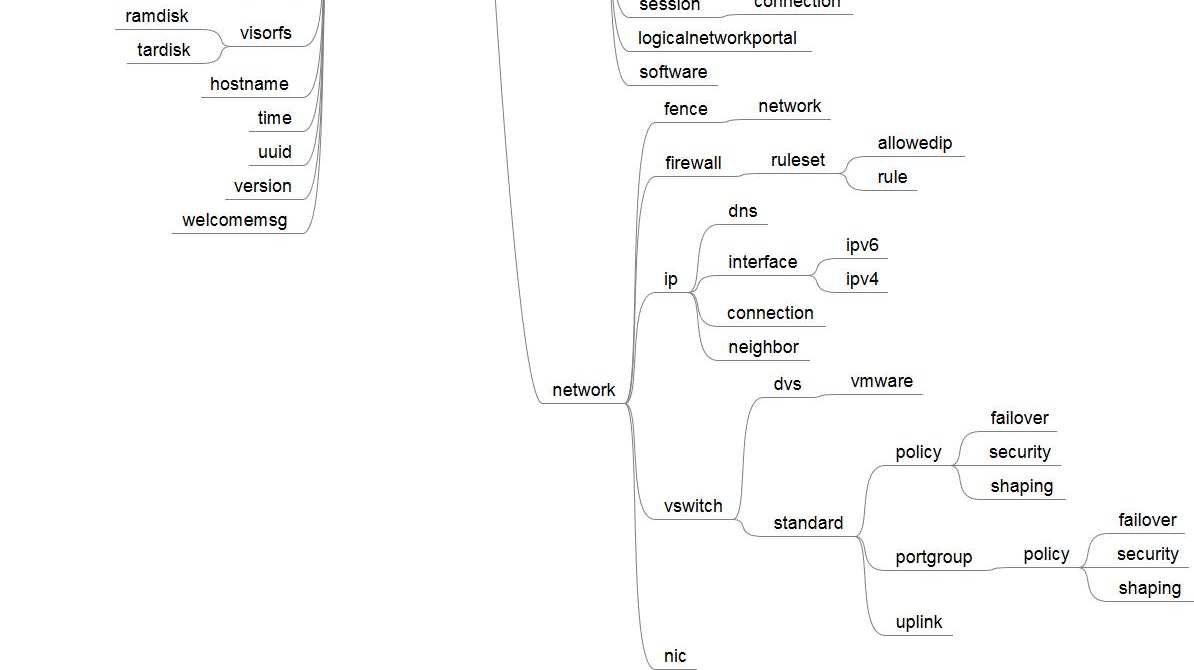
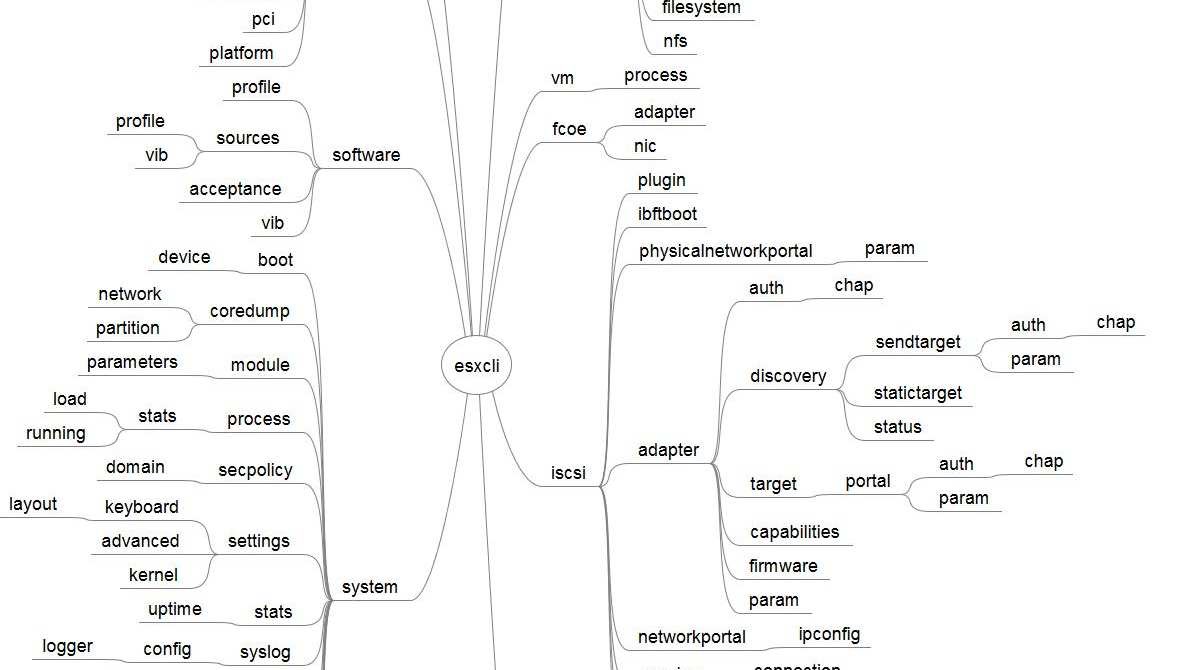
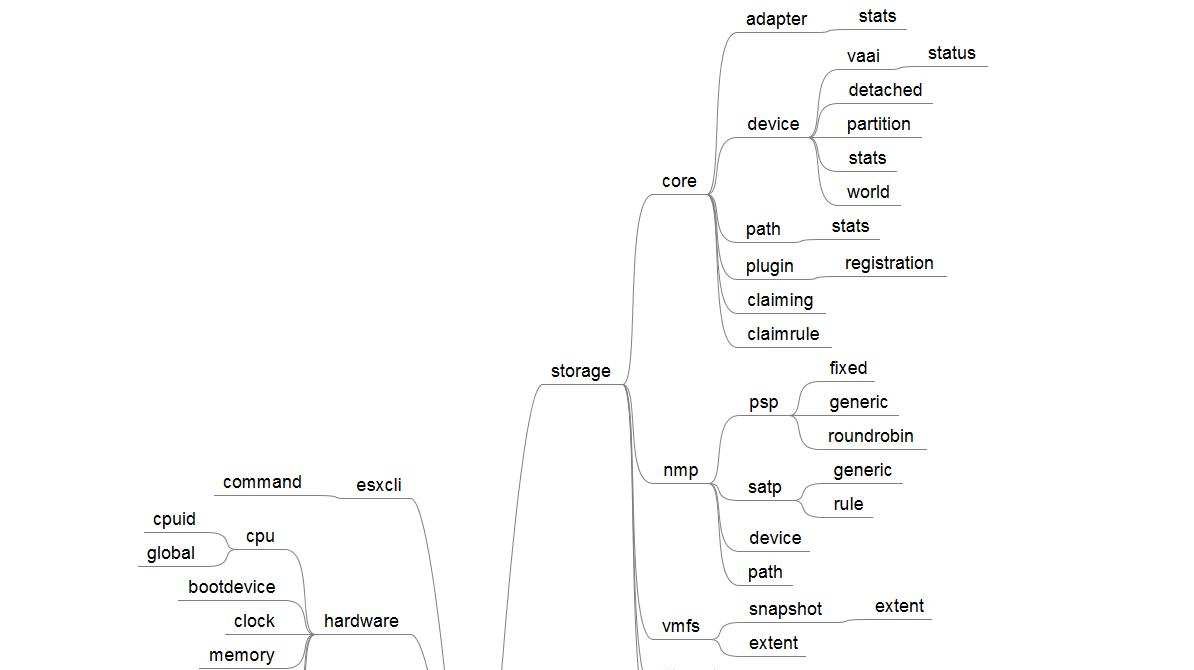


Figure 3 – ESXCLI complete hierarchy overview

VRR topology creation using ESXCLI

Every GUI operation on the vSphere interface gets translated to an underlying system operation on the ESXi server. This applies to all Virtual machine creation, hybridging and similar other operations. Using CLI operations allows the user to integrate scripting abilities, automating these operations for different topologies with the added advantage of running Junos specific operations while still using a virtual environment, which in this case is VMware.

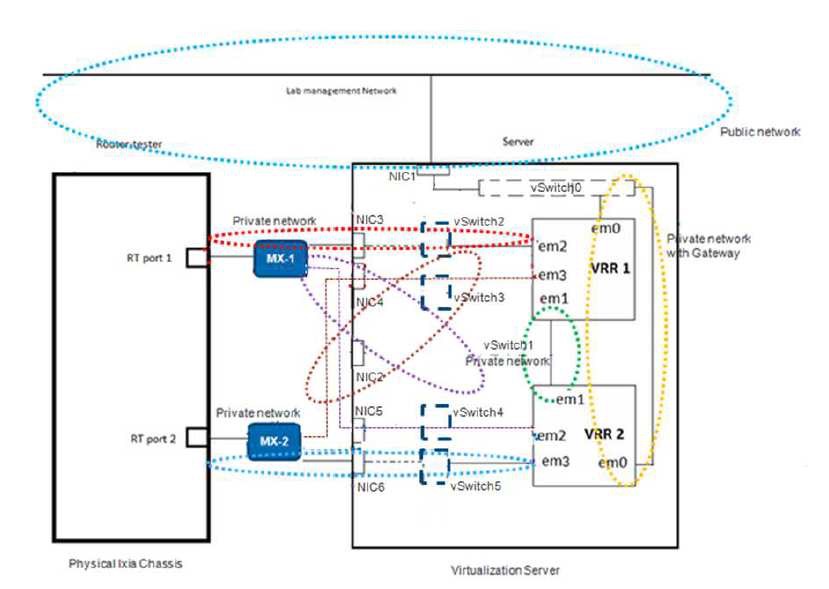


Figure 4 – Baseline testbed topology diagram

Brief summary of some the key file is as below:

.log Virtual machine log file used for troubleshooting

.nvram Stores the state of the virtual machine's BIOS

.vmdk The Virtual disk file stores the contents of the virtual machine’s hard disk drive. A virtual disk is made up of one or more .vmdk files.

.vmem The virtual machine's paging file, which backs up the guest main memory on the host file system. This file exists only when the virtual machine is running, or if the virtual machine has crashed.

.vmsd Centralized file for storing information and metadata about snapshots.

.vmsn Snapshot state file which stores the running state of a virtual machine

.vmss Suspended state file which stores the state of a suspended virtual machine

.vmtm Configuration file containing team data

.vmx Primary configuration file, which stores settings chosen in the New Virtual Machine Wizard or virtual machine settings editor.

.vmxf Supplemental configuration file for virtual machines that are in a team

There are multiple files that compose a virtual machine, management of which is done internally. A virtual machine typically is stored on the host in a set of files, usually in a directory created specifically for that specific virtual machine.

VRR topology creation can be broadly classified as:

* Creation of virtual machines
* Creation of vSwitch and configuring the same for hybridging
* Copying Junos baseline configuration to created virtual machines

Virtual machine creation using ESXCLI

Three main steps are involved to create a virtual machine utilizing the command-line tools :

* Creation of a virtual machine configuration file
* Creation of a virtual machine disk file
* Registering the virtual machine with the ESXi Server

Creation of a virtual machine configuration file

The .vmx file is just a text file with specific fields that define the virtual machine's hardware configuration. Not all parameters in a vmx file need to be specified before virtual machine creation. The system also generates and adds data to the vmx file.

Creation of a virtual machine disk file

VMware has a command-line utility, called vmkfstools, which can be used for the creation of VMFS file systems and virtual machine disk files. It is possible to either create a VMDK file or clone an existing image for VM creation.

vmkfstools  
OPTIONS FOR VIRTUAL DISKS:  
vmkfstools -c --createvirtualdisk #[gGmMkK]  
-d --diskformat [zeroedthick|eagerzeroedthick|thick|thin] -a --adapterType [buslogic|lsilogic]  
-w --writezeros  
-j --inflatedisk  
-U --deletevirtualdisk  
-E --renamevirtualdisk srcDisk  
-i --clonevirtualdisk srcDisk  
-d --diskformat [rdm:|rdmp:|raw:|thin|2gbsparse]  
-X --extendvirtualdisk #[gGmMkK]  
-M --migratevirtualdisk  
-r --createrdm /vmfs/devices/disks/...  
-q --queryrdm  
-z --createrdmpassthru /vmfs/devices/disks/...  
-Q --createrawdevice /vmfs/devices/generic/...  
-v --verbose #  
-g --geometry  
vmfsPath

For the VRR topology creation, the approach of cloning a base VMDK file for VRR to create an individual .vmdk copy, for each virtual machine, has been taken.

Usage: vmkfstools -i <Base-VMDK> <VM-VMDK>

Registering the virtual machine with the ESXi Server

For the virtual machine to be operational, as a last step in the VM creation process it is necessary to register the guest OS with the ESXi host. The return value is a VM Identifier which is a unique ID assigned to each operational virtual machine.

Usage : vim-cmd solo/registervm <.vmx>

Powering the virtual machine up/down

After successful creation and operation of the virtual machine, the below commands can be used to power on/off the VM.

~ # vim-cmd vmsvc/getallvms

Vmid Name

277 vRR1

~ # vim-cmd

Retrieved runtime info

Powered on

~ # vim-cmd vmsvc/power.off 277

Powering off VM:

~ # vim-cmd vmsvc/power.getstate 277 Retrieved runtime info  
Powered off

~ # vim-cmd vmsvc/power.on 277

Powering on VM:

~ # vim-cmd vmsvc/power.getstate 277 Retrieved runtime info  
Powered on

Hybridging using ESXCLI

The interconnection of physical and virtual interfaces is a two-step process:

* Assigning the port-groups to the VM interfaces in the .vmx file  
  Usage: This maps the VM interface to portgroup "vRR1 - MX1" ethernet2.pciSlotNumber = "34"  
  ethernet2.virtualDev = "e1000"  
  ethernet2.networkName = "vRR1 - MX1"  
  ethernet2.addressType = "generated"  
  ethernet2.present = "TRUE"
* Creation of vSwtich and mapping the same to relevant port-groups and uplinks  
  esxcli network vswitch standard uplink remove --uplink-name=*vmnic* –-vswitch-name=vSwitch

# unlink an uplink

esxcli network vswitch standard uplink add --uplink-name=*vmnic* –-vswitch-name=vSwitch # add an uplink

Usage: creates vSwtich2 and maps portgroup "VRR1 - MX1" to interconnect ethernet2 of the VM above to the physical port vmnic4 on the ESXi host

# esxcli network vswitch standard add --vswitch-name=vSwitch2  
# esxcli network vswitch standard portgroup add --portgroup-name="VRR1 - MX1"  
# esxcli network vswitch standard uplink add --uplink-name=vmnic4 --vswitch-name=vSwitch2

Python script for VRR topology automation

Though existing VMware command-line tools and APIs help the automation and management of virtual environment operations, it is complicated to get a level deeper to access the guest OS and automate guest OS specific operations. Thus network devices, running Junos in a virtual environment cannot benefit from automation resources that benefit only virtual infrastructure specific operations. A local standalone tool has been developed in Python that helps integrate automation for both the creation of the virtual environment with hybridging and loading the Junos configuration.

This tool simulates the static baseline VRR testbed with the objective of bringing it up quickly which can help expedite testing and customer demonstrations. The source code is available here along with the instructions on how to run the script :

<https://github.com/deeptic/Virtualization-Automation/blob/master/vRR-Automate-ESXi.py>

Below is a summary of the algorithm the script uses to accomplish this objective:

1. Get user preference of creating only the VM topology for VRR or loading a baseline Junos configuration along with VM topology creation. A log file is also created to capture all session logs.
2. Baseline VMDK has pre-build dummy management IP and services config enabled which is effectively used only when the user chooses to load the Junos configuration on created VMs. This base VMDK file is cloned for each individual VM.
3. For VM and vSwitch creation, esxcli commands as described above are invoked from the appropriate functions.
4. In order to load Junos configuration on the created VMs, the base VMDK file has dummy management IP and services configured to allow access to the guest OS from the ESXi host. The script ensures that multiple VMs with the same IP do not come up at the same time to prevent any IP conflicts.
5. The script also ensures that the created VRRs are accessible from the external network should the user select the option of loading Junos configuration.
6. The end result are VRRs powered on and ready for use, with the connections that emulate the VRR baseline physical testbed as shown in Figure 4.

Advantages

* Leveraging command-line interface for virtual environment facilitates easy automation and management
* Enhanced flexibility to achieve powerful operations
* Scalable solution
* Allows for integration of VM and Junos automation
* Easy rollout of test beds useful for customer demonstrations and testing teams
* Tool can be uplifted to incorporate support for more dynamic topologies and network scripts

Limitations

Understanding use cases of taking advantage of this approach in a testing environment. The tool is a starting point on how system calls & APIs can be leveraged to integrate virtual infrastructure usage with Junos, which would be essential in network environments.

Conclusion

The benefits of Network Function Virtualization have been enumerated numerous times with those ranging from cost benefits to new product deployments. Customers are looking at solutions that would be in line with production networks in terms of scalability and complexity. Automation is a critical component of network provisioning and management. With the integration of virtual and network functions, automation also needs to leverage the same functionality. Command-line tools and APIs serve a beneficial purpose and can be customized to involve their use for not only virtual infrastructure specific operations but also for network related functions.