Hyper-V setup environment

The scope of this guide is to detail the initial configuration of a Hyper-V Server that will be used by LISAv2 in order to validate the functionality of Hyper-V as a hypervisor.

These instructions are not required for running LISAv2 to test on Azure. Please refer to LISAv2 readme document (https://github.com/LIS/LISAv2/blob/master/README.md) for LISAv2 generic setup for running against Azure.

Basic resource requirements are:

- Hyper-V host version: WS2016 or newer.
- PowerShell version: 5.0 or above, refer to the main readme document for install instructions.
- Download Latest Azure PowerShell
    1. Download Web Platform Installer from here
    2. Start Web Platform Installer and select Azure PowerShell (required 6.3.0 or above) and proceed for Azure PowerShell Installation.
- At least 8 GB of memory on the Hyper-V host.

  Most of LISAv2 tests will create and start Virtual Machines (Guests) with 3.5 GB of memory assigned.

- Hyper-V role enabled:

  ```
  Install-WindowsFeature -Name Hyper-V -IncludeManagementTools
  ```

- Git client installation:

  ```
  $GitURL = "https://github.com/git-for-windows/git/releases/download/v2.18.0.windows.1/Git-2.18.0-64-bit.exe"
  [Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::Tls12
  (New-Object System.Net.WebClient).DownloadFile($GitURL,"C:\Windows\Temp\git.exe")
  C:\Windows\Temp\git.exe /VERYSILENT
  Start-Sleep -Seconds 10
  $env:Path += ";C:\Program Files\Git\bin\"
  ```

- For test cases that run on multiple Hyper-V hosts, WinRM should be enabled and all the hosts need to be in the same trusted domain. The user under which LISAv2 runs should be an Administrator on all the hosts where LISAv2 runs.
- One External vSwitch in Hyper-V Manager/Virtual Switch Manager. This must have internet connection for LISAv2 to run correctly. We name this vSwitch “External”.

  ```
  $nicWithExternalConnectivity = Get-NetRoute | ? DestinationPrefix -eq '0.0.0.0/0' | Get-NetIPInterface | Where ConnectionState -eq 'Connected' | Get-NetAdapter -Physical | Select-Object -First 1
  ```

New-VMSwitch -Name External -NetAdapterName $nicWithExternalConnectivity.Name -AllowManagementOS $true

- For Hyper-V NETWORK tests you need 2 more vSwitch types created: Internal and Private. These 2 vSwitches will have the naming also 'Internal' and 'Private'. For the internal vSwitch we assign 192.168.0.0/24 address range.

  New-VMSwitch -Name Private -SwitchType Private -Notes 'Private network - VMs only'

  New-VMSwitch -Name Internal -SwitchType Internal -Notes 'Parent OS, and internal VMs'
  Get-NetAdapter -Name "vEthernet (Internal)" | New-NetIPAddress -AddressFamily ipv4 -IPAddress 192.168.0.1 -PrefixLength 24

- For VSS tests, lisav2 requires "Windows-Server-Backup" role installed and for live/quick migration tests, lisav2 requires "Failover-Clustering" and "RSAT-Clustering" roles

  [String[]]$Roles = @("Failover-Clustering","Windows-Server-Backup","RSAT-Clustering")
  Install-WindowsFeature -Name $Roles -IncludeAllSubFeature –IncludeManagementTools

Specialized features requirements – SR-IOV

- A pair of hosts with SR-IOV capable NICs. LISAv2 supports both Mellanox ConnectX-3 Pro cards and Intel X520 adapters. These NICs must be capable of seeing each other. All the SR-IOV tests (both functional and performance ones) are run on 2 different Hyper-V hosts and the framework expects that the physical NICs are connected to the same switch.
- For either Mellanox or Intel cards, the latest Windows drivers and firmware must be installed for that specific card. Please consult the official Mellanox or Intel websites for the latest drivers & firmware.
- We need to create, on each host, an extra vSwitch using that SR-IOV capable NIC. The naming, by default, is the following:
  - Functional testing:
    - vSwitch will be named “SRIOV_MLNX” in the case of Mellanox NICs
    - VSwitch will be named “SRIOV_INTEL” in the case of Intel NICs
  - Performance testing: vSwitch will be named “SRIOV”
- On each host, run the following cmdlet:

New-VMSwitch -Name SRIOV_NAME -NetAdapterName SR-IOV_INTERFACE_NAME -EnableIov $true

- Once the vSwitches are created, we can check if the VFs (Virtual Functions) are available on each host.

Good SR-IOV setup:

```
PS> Get-VMSwitch SRIOV_MLNX | fl *iov*
IovEnabled            : True
```

IovSupport : True
IovSupportReasons :
IovQueuePairCount : 8027
IovQueuePairsInUse : 2
IovVirtualFunctionCount : 32 – *The most important property. If it's >0, it means the host setup is good*
IovVirtualFunctionsInUse : 0

Incorrect SR-IOV setup:

PS> Get-VMSwitch External | fl *iov*
IovEnabled : False *(This can also be "True" but 'IovVirtualFunctionCount ' to be 0)*
IovSupport : False *(This can also be "True" but 'IovVirtualFunctionCount ' to be 0)*
IovSupportReasons : {….}
IovQueuePairCount : 0
IovQueuePairsInUse : 0
IovVirtualFunctionCount : 0
IovVirtualFunctionsInUse : 0

- If the IovVirtualFunctionCount is 0 or null that means something in the first 1-3 points is broken and the SR-IOV setup should be revised. Without VFs available, SR-IOV tests cannot be performed with lisav2.

Specialized features requirements – NVMe Direct

IMPORTANT NOTE: The below 4 steps are included in LISAv2 (SETUP-Add-NVME-Disk.ps1). If the host where LISAv2 runs has NVMe devices attached, then lisav2 will be able to handle all the necessary setup and run the NVMe test cases.

To get the current status of the NVMe devices (to check if they are ready to be attached to VM), we need to run this cmdlet:

Get-VMHostAssignableDevice -ComputerName localhost

After we run this cmdlet, we can have 2 cases:

1. The number of objects returned (NVMe devices) is the expected one. In this case, we need to perform an additional check:
   - We need to check if the NVMe devices are already assigned to other VMs. To do this, we run this PS cmdlet:

   *Get-VMAssignableDevice -VmName **

   - If there is any object returned, we need 2 properties from it: VMName and LocationPath. With these 2 strings, we run

   Remove-VMAssignableDevice -LocationPath LOCATION_PATH_PROPERTY -VMName VM_NAME_PROPERTY

   - The 'remove' operation will be repeated for every returned object
2. The number of objects returned (NVMe devices) is less than expected. In this case, we need to do the following
   - We need to check if all the NVMe devices are disabled and dismounted from the host. Otherwise, they will not be able be attached to a guest vm.

```
$activePnpDevs = Get-PnpDevice -PresentOnly -CimSession localhost `
    | Where-Object {$_.Class -eq "SCSIAdapter"} | Where-Object {$_.Service -eq "stornvme"}
foreach ($pnpDev in $activePnpDevs) {
    Disable-PnpDevice -InstanceId $pnpdev.InstanceId -Confirm:$false
    $locationPath = ($pnpdev | Get-PnpDeviceProperty DEVPKEY_Device_LocationPaths).data[0]
    Dismount-VMHostAssignableDevice -locationpath $locationpath
}
```

In this point, all the NVMe devices are detached from the guest VMs and from the host. We can check their location path again with this PS cmdlet:

Get-VMHostAssignableDevice -ComputerName localhost.

This will return more than one location paths. We will choose how many devices we want to attach. For each device we want to attach, we will run the following command:

Add-VMAssignableDevice -LocationPath LOCATION_PATH -VMName TEST_VM

This cmdlet will be repeated until we reach the wanted number of NVMe devices attached to the VM.

Secrets file

The easiest way to give necessary parameters for lisav2 framework is with a secrets.xml file. This contains the user/password info for logging into the test VM, database credential for uploading the results, blob location for the 3rd software party mentioned earlier that is needed in Tools folder.

The secrets.xml file has the following structure:

```
<secrets>
  <SubscriptionID> AZURE_SUBSCRIPTION_ID</SubscriptionID>
  <SubscriptionName>AZURE_SUBSCRIPTION_NAME</SubscriptionName>
  <SubscriptionServicePrincipalTenantID></SubscriptionServicePrincipalTenantID>
  <SubscriptionServicePrincipalClientID></SubscriptionServicePrincipalClientID>
  <SubscriptionServicePrincipalKey></SubscriptionServicePrincipalKey>
  <linuxTestPassword>TEST_VM_PASSWORD</linuxTestPassword>
  <linuxTestUsername>TEST_VM_USER</linuxTestUsername>
  <DatabaseServer>DB_ADDRESS</DatabaseServer>
  <DatabaseUser>DB_USER</DatabaseUser>
  <DatabasePassword>DB_PASSWORD</DatabasePassword>
  <DatabaseName>DB_TABLE</DatabaseName>
  <blobStorageLocation>3D_SOFTWARE_BLOB_LOCATION</blobStorageLocation>
</secrets>
```

Test VHD

Any Linux distro/version image can be put into testing if it follows a set of rules:

- For Hyper-V, the supported extensions are .vhd and .vhdx

- SSH service must be running - lisav2 framework will send commands and files through this service
- A non-root user must be created on the test image. The username/password pair will be provided either in the secrets file (inside linuxTestUsername and linuxPassword tags) or in GlobalConfigurations.xml (in case of missing secrets file) (inside TestCredentials tag)
- KVP daemon running – this will provide the IP address of the test VM to the Hyper-V host. Without it, lisav2 framework will not be able to contact the test VM. If the daemon is not running on the test image, it must be installed:
    - On RHEL/CentOS you can install either LIS drivers (http://aka.ms/lis) or "hyperv-daemons" package with yum
    - On Ubuntu/Debian the following packages have to be installed with apt: "linux-cloud-tools-common", "linux-tools-`uname -r`", "linux-cloud-tools-`uname -r`" (where 'uname -r' gets the kernel version)
    - On SLES KVP must be running out of the box

In case we don't have an image that meets the above requirements, we should manually create a VM and make the necessary changes. Once the changes are made, we recommend a reboot of the machine to check again the status of the changes. If everything is good, the VM can be shut down, and the OS vhd can be copied to a location from which lisav2 will use it.

Running LISAv2

A Windows PowerShell console should be started with Administrator rights. If the above setup was performed, then we should be able to clone LISAv2 on the host.

git clone https://github.com/LIS/LISAv2

In this point, LISAv2 can be run successfully if the above setup steps were executed (valid XML secrets file & test vhd).

This is a basic LISAv2 run example:

```
.\Run-LisaV2.ps1 -TestPlatform 'HyperV'  -RGIdentifier 'testIdentifier' -OsVHD
'\\test\vhd\location\image.vhdx' -TestLocation 'localhost'  -TestNames "BVT-VERIFY-DEPLOYMENT-
PROVISION" -XMLSecretFile secrets.xml
```

Parameters

- **-TestPlatform –** It will be always "HyperV" for a Hyper-V run – no need to change this at any point. Mandatory parameter.
- **-RGIdentifier –** This is where the user has the choice to put a specific identifier for the test VM(s). This will not change the entire name of the VM(s) but will change a part of the VM(s) name with the word given by the user. Useful for easily identify your test VMs in an environment where multiple VMs are present. Mandatory parameter.
- **-OsVHD –** It accepts both vhd and vhdx files. There are 3 types of locations supported and that can be given to this parameter:

- Local paths: *C:\Full\Path\To\Test_VHD.vhdx*
- Remote server paths: *\\Remote-Server\C$\Full\Path\To\Test_VHD.vhdx*
- Network shares: \\Network\Share\Full\Path\To\Test_VHD.vhdx

For all 3 cases, we can either give the full location as shown above or we can use just the "Test_VHD.vhdx" value in conjuction of –**SourceOsVHDPath** parameter. In this case, the command line will look as follows:

-OsVHD "Test_VHD.vhdx" -SourceOsVHDPath "C:\Full\Path\To\" (for local paths)

- **-TestLocation –** It usually takes "localhost" value, but if a run is wanted on another host, then the value should be changed to that server name.

For SR-IOV functional and Network Performance tests, this parameter must include both test servers in the following form:

-TestLocation "server_1,server_2"

.\Run-LisaV2.ps1 -TestPlatform 'HyperV' -RGIdentifier 'testIdentifier' -OsVHD '\\test\vhd\location\image.vhdx' -TestLocation 'localhost,server_2' -TestNames "SRIOV-VERIFY-VF-BASIC-CONNECTION" -XMLSecretFile secrets.xml

- -**TestNames –** This is the used parameter when we want to run a specific test case. YOu can run 1 or more test cases (e.g: -TestNames "Test1,Test2, …,TestN"). There are other 3 ways to load the test cases: **-TestCategory, -TestArea, -TestTag:**
  - Using just -**TestCategory**, you can run an entire category of tests (e.g: Functional ctests ontains KVP, Storage, KDUMP, SR-IOV, FCOPY, etc and all of them will be run)

    .\Run-LisaV2.ps1 -TestPlatform 'HyperV' -RGIdentifier 'testIdentifier' -OsVHD '\\test\vhd\location\image.vhdx' -TestLocation 'localhost' -TestCategory 'Functional' -XMLSecretFile secrets.xml

  - Using -**TestCategory** and **-TestArea** you can narrow down the tests you want to run. (e.g. You can select Functional category and KVP Area). Also, multiple areas from that category can be run ("area_1,area_2,etc")

    .\Run-LisaV2.ps1 -TestPlatform 'HyperV' -RGIdentifier 'testIdentifier' -OsVHD '\\test\vhd\location\image.vhdx' -TestLocation 'localhost' -TestCategory 'Functional' -TestArea 'KVP,FCOPY,LIS' -XMLSecretFile secrets.xml

  - Using –**TestTag** will run all the tests that have the specified tag inside the test definition.

    .\Run-LisaV2.ps1 -TestPlatform 'HyperV' -RGIdentifier 'testIdentifier' -OsVHD '\\test\vhd\location\image.vhdx' -TestLocation 'localhost' -TestTag 'memory' -XMLSecretFile secrets.xml

- **-XmlSecretFile –** provide the secrets file created earlier. If this is not provided, you need to do additional steps before being able to run LISAv2 framework:
  - In XML\GlobalConfigurations.xml the "LinuxUsername" and "LinuxPassword" must be filled with the test vhd credentials

- Download the 3d software party in the Tools folder (those tools are available in Azure blob and network share, \\redmond\wsscfs\OSTC\LIS\LISAv2 . Contact to lisasupport@microsoft.com)

Additional parameters

- **-ResultDBTable –** If you want to upload the results to a database table, you can add this parameter. Only needed for Performance test cases. In this case, database credentials will be added either in secrets file or in GlobalConfigurations.xml

.\Run-LisaV2.ps1 -TestPlatform 'HyperV' -RGIdentifier 'testIdentifier' -OsVHD '\\test\vhd\location\image.vhdx' -TestLocation 'localhost' -TestNames "PERF-NETWORK-TCP-THROUGHPUT-MULTICONNECTION-NTTTCP-SRIOV" -XMLSecretFile secrets.xml -ResultDBTable "Perf_Network_TCP_Azure_DefaultKernel"

- -**CustomKernel –** If you want the framework to install a custom kernel on top of the test vhd before all the tests are performed, use this parameter. The supported custom kernels are: proposed, proposed-azure, proposed-edge, ppa, latest, netnext, *.deb, *.rpm, linuxnext

.\Run-LisaV2.ps1 -TestPlatform 'HyperV' -RGIdentifier 'testIdentifier' -OsVHD '\\test\vhd\location\ubuntu_image.vhdx' -TestLocation 'localhost' -TestNames "BVT-VERIFY-DEPLOYMENT-PROVISION" -XMLSecretFile secrets.xml -CustomKernel "proposed-azure"

Known behavior that can cause issues

- Framework modules that remain loaded after a test run. This can cause issues if a new lisav2 test run is performed in the same PowerShell console using different parameters. In this case, there are 2 solutions:
  - Open a new PS console for each test run
  - After a test run, remove the lisav2 modules: e.g. *Remove-Module XmlProcessing*. Repeat this for every lisav2 module loaded (modules are in Libraries folder).
- VHD not being copied & used in the test framework if the –OsVHD param is a network share.

  The framework is set to copy the VHD from the share to the test host (In the default Hyper-V VHD path). But to speed things up, it will first look if the VHD already exists on the host. It will look into the DiskIdentifier and if they match, the copy will be skipped and the existing VHD will be used. But this may not be wanted in this situation:

  - A test has been run on the localhost, using a vhd file located on a network share. In the first run, the VHD will be copied
  - Let's assume we spot an issue with the VHD (e.g KVP not running). We made changes to the VHD and upload it to the network share.
  - We start lisav2 again. In this case, the VHD will not be copied.

  If a similar situation occurs, we must reset the vhd disk identifier for the vhd located on the network share. We can do this with this cmdlet:

  - Set-VHD Test_VHD.vhdx -ResetDiskIdentifier

After doing this step, lisav2 will compare the disk identifiers and detect that they are changed. In this case, it will copy again the VHD from the share.